PostgreSQL 9.5 新機能解説

Noriyoshi Shinoda

December 5, 2015

自己紹介
篠田典良（しのだのりよし）

- 所属
  - 日本ヒューレット・パッカード株式会社 テクノロジーコンサルティング事業統括
- 現在の業務
  - PostgreSQLをはじめOracle Database, Microsoft SQL Server, Vertica, Sybase ASE等 RDBMS全般に関するシステムの設計、チューニング、コンサルティング
  - オープンソース製品に関する調査、検証
  - Oracle Database関連書籍の執筆
  - 弊社講習「Oracle DatabaseエンジニアのためのPostgreSQL入門」講師
- 関連する URL
  - 「PostgreSQL 虎の巻」シリーズ
    - http://h30507.www3.hp.com/t5/user/viewprofilepage/user-id/838802
  - Oracle ACEってどんな人？
    - http://www.oracle.com/technetwork/jp/database/articles/vivadeveloper/index-1838335-ja.html

Agenda
PostgreSQL 9.5新機能解説

1. PostgreSQL 9.5新機能概要
2. 大規模環境に対応する機能
3. 運用を容易にする機能
4. アプリケーション開発関連
5. アーキテクチャの変更
6. PostgreSQL 10?
7. まとめ

1. PostgreSQL 9.5概要

1. PostgreSQL 9.5概要
PostgreSQLの歴史

- 1974年 Ingres プロトタイプ
  - HP NonStop SQL, SAP Sybase ASE, Microsoft SQL Serverの元になる
- 1989年 POSTGRES 1.0～
- 1995年 Postgres95
  - SQLのサポート
- 1997年 PostgreSQL 6.0～
  - GEQO, MVCC, マルチバイト
- 2000年 PostgreSQL 7.0～
  - WAL, TOAST
- 2005年 PostgreSQL 8.0～
  - 自動VACUUM, HOT, PITR
- 2010年 PostgreSQL 9.0～
  - レプリケーション, 外部表, JSON, マテリアライズド・ビュー

1. PostgreSQL 9.5概要
開発ステータス

- 2015年12月現在
  - PostgreSQL 9.5 Beta 2
- リリース状況
  - 2015年6月 Alpha 1
  - 2015年8月 Alpha 2
  - 2015年10月 Beta 1
  - 2015年11月 Beta 2　　← いまここ
  - 2015年12月 RC ?
  - 2016年1月 リリース?
- 次期バージョン（PostgreSQL 10?）も絶賛開発中

1. PostgreSQL 9.5概要
本日説明する新機能

- 大規模環境に対応する機能
  - BRIN Index
  - CREATE FOREIGN TABLE INHERITS文
  - SELECT TABLESAMPLE文
- 運用を容易にする機能
  - pg_rewindコマンド
  - ALTER TABLE SET UNLOGGED文
- アプリケーション開発関連
  - INSERT ON CONFLICT文
  - Row Level Security機能
  - JSONBに関する新機能
  - UPDATE SET文
  - SELECT SKIP LOCK文
  - etc

1. PostgreSQL 9.5概要
本日説明しない主な新機能

- パフォーマンスの向上
  - ロック制御の改善によるマルチ・プロセッサ環境におけるスループット向上
  - ソート処理の高速化
- ユーティリティの改善
  - vacuumdbコマンドの--jobsパラメータ
  - pgbenchコマンドの--latency-limitパラメータ
- SQL文の新機能
  - IMPORT FOREIGN SCHEMA文
  - CREATE EVENT TRIGGERの拡張
  - REINDEX SCHEMA文

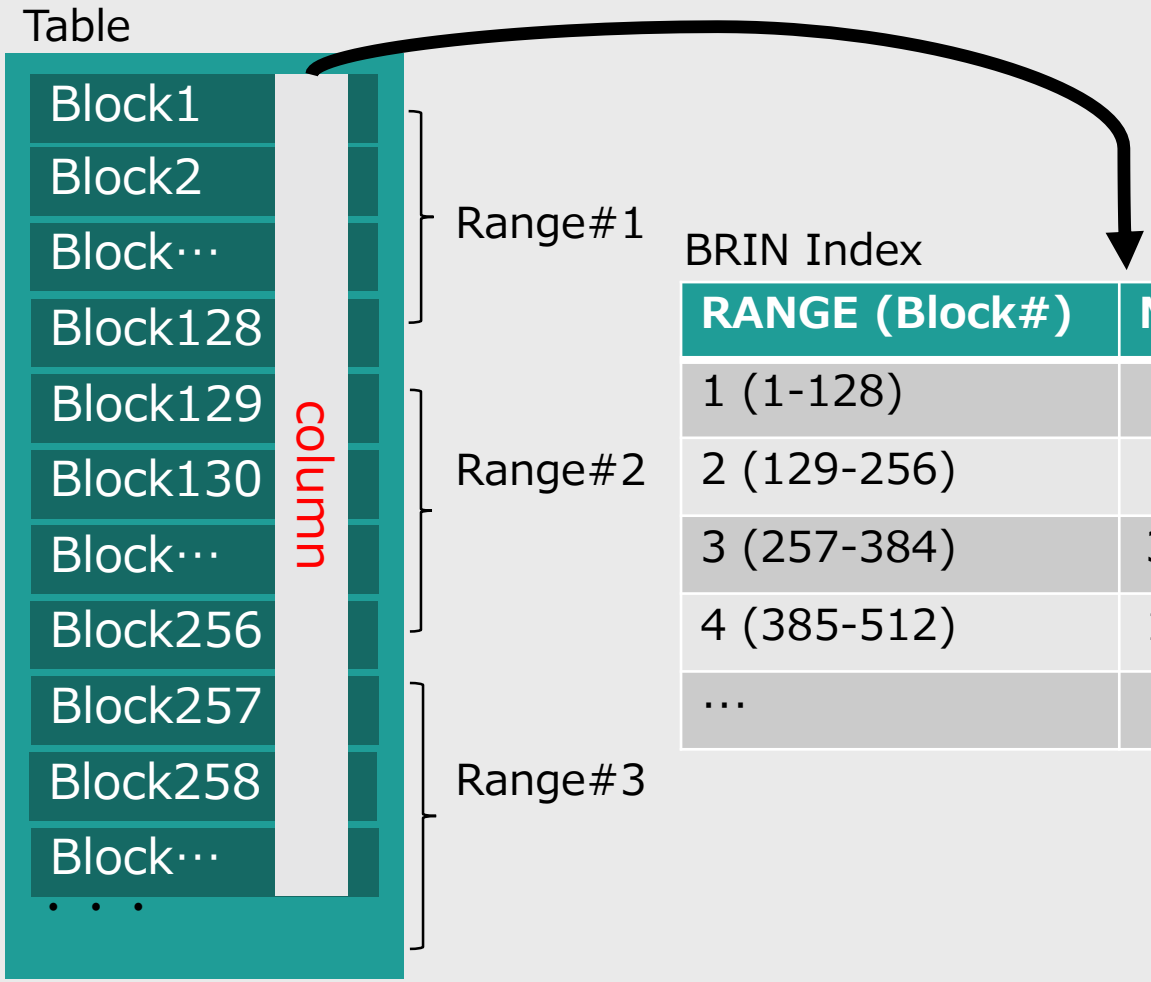
2. 大規模環境に対応する機能

BRIN Index (Block Range Index)
大規模環境におけるストレージ量とパフォーマンスのバランス

- B-treeインデックス
  - B-treeインデックスは、列値とタプルID（TID）のセットをソートして保存
  - 範囲検索は非常に高速
  - 大規模環境では、ストレージ使用量が多い
- BRINインデックス
  - 隣接する複数ブロックを１レンジとして、レンジ単位に最大値／最小値／NULL値の有無を保持
  - レンジ内のブロック数はパラメーターpages_per_rangeで決定（デフォルト値128）
  - ストレージ使用量が非常に少ない
  - B-treeインデックスよりも低速だが、全件検索よりもはるかに高速

```
CREATE INDEX インデックス名 ON テーブル名 USING BRIN (列名)
[ WITH (pages_per_range = ページ数) ]
```

BRIN Index (Block Range Index)
BRIN Indexの構造



BRIN Index

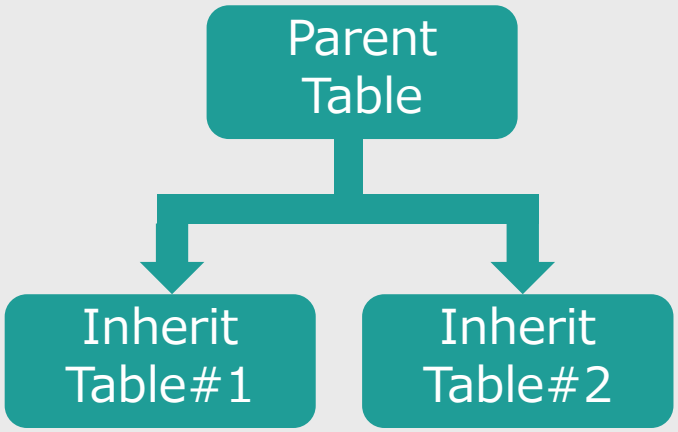
RANGE (Block#)	MIN	MAX	NULL?
1 (1-128)	100	20,000	Yes
2 (129-256)	10	3,000	No
3 (257-384)	3,000	40,000	Yes
4 (385-512)	1,000	100,000	No
…			

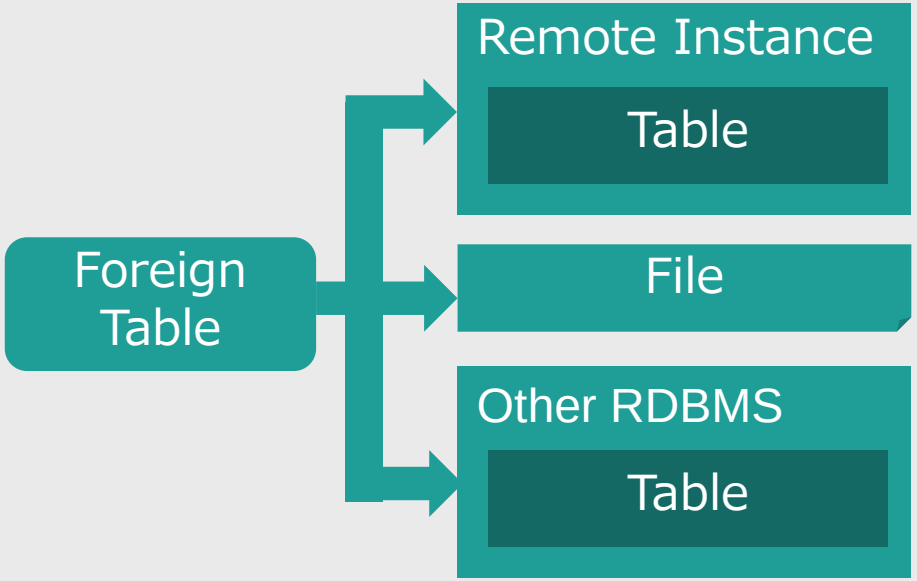
BRIN Index (Block Range Index)
実行例

```
postgres=> CREATE TABLE brin1(c1 NUMERIC, c2 NUMERIC) ;
CREATE TABLE
postgres=> INSERT INTO brin1 VALUES (generate_series(1, 100000000),
      generate_series(1, 100000000)) ;
INSERT 0 100000000
postgres=> CREATE INDEX idx_btree ON brin1 (c1) ;
CREATE INDEX
postgres=> CREATE INDEX idx_brin ON brin1 USING BRIN (c2) ;
CREATE INDEX
postgres=> SELECT relname, pg_size_pretty(pg_relation_size(oid)) FROM
      pg_class ;
relname    | pg_size_pretty
-----------+----------------
 brin1     | 4223 MB
idx_btree  | 2142 MB
idx_brin   | 160 kB
```

CREATE FOREIGN TABLE INHERITS
複数リモート・インスタンスに処理をオフロード

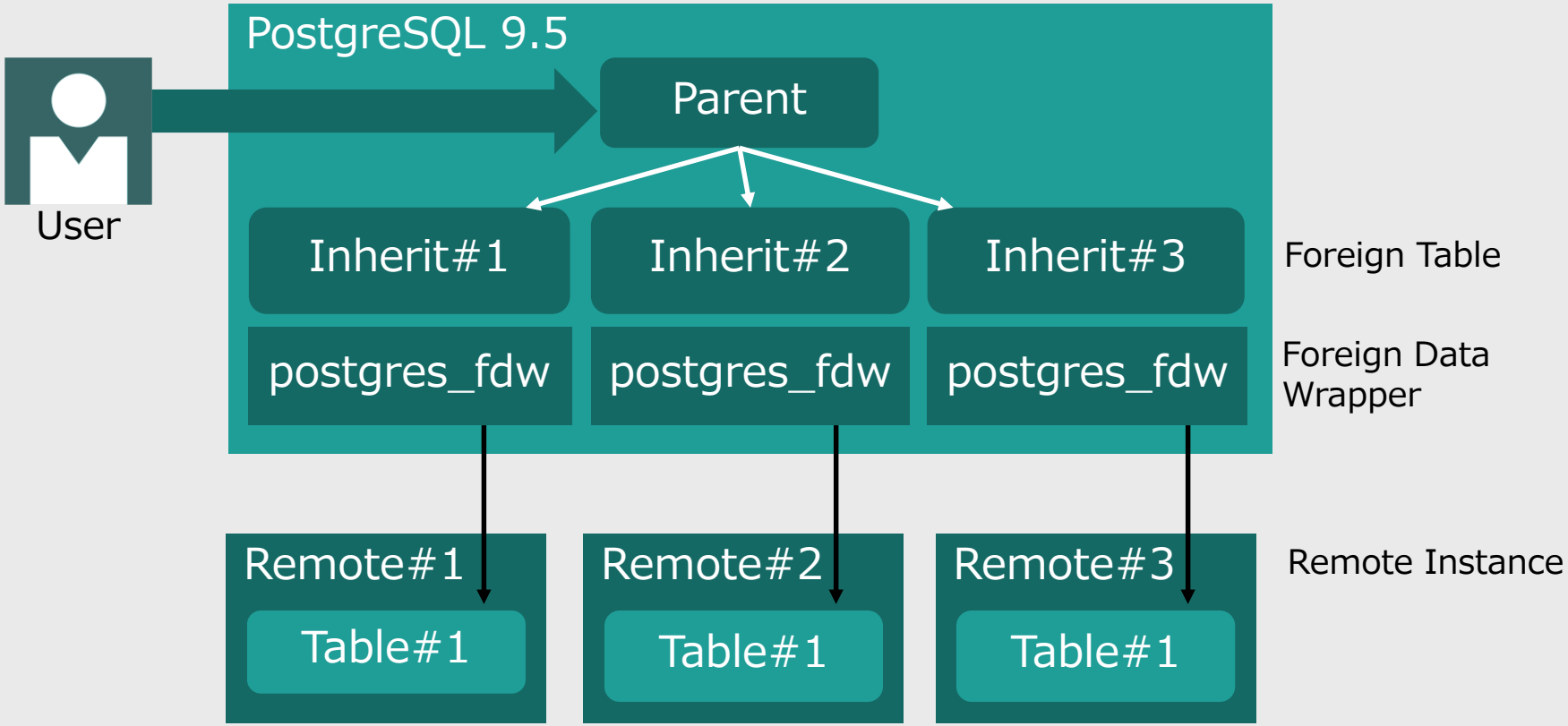
- 継承テーブル（INHERITS）
  - 親子関係を持つテーブル（パーティション・テーブルと呼ぶことも）
  - WHERE句により自動的に子テーブルを参照することで負荷分散が可能
- 外部テーブル（FOREIGN TABLE / FOREIGN DATA WRAPPER)
  - リモート・インスタンスのテーブルやファイルをテーブルとして参照できる機能（ファイルや他システム等）





CREATE FOREIGN TABLE INHERITS
複数リモート・インスタンスに処理をオフロード

– PostgreSQL 9.5では、外部テーブルと継承テーブルを混在可能に



CREATE FOREIGN TABLE INHERITS
実行例

```
postgres=> CREATE TABLE parent_table (col1 NUMERIC, …) ;

postgres=# CREATE SERVER remote1 FOREIGN DATA WRAPPER
      postgres_fdw OPTIONS (host 'remsvr1', dbname 'userdb1', port '5432') ;

postgres=# CREATE USER MAPPING FOR public SERVER remote1 OPTION
      (user 'demo', password 'secret') ;

postgres=> CREATE TABLE inherit_table1 INHERITS (parent_table) ;
      SERVER remote1 ;
```

– 集計処理（MAX / MIN / SUM / AVG / GROUP BY etc）はローカル・インスタンスで実施

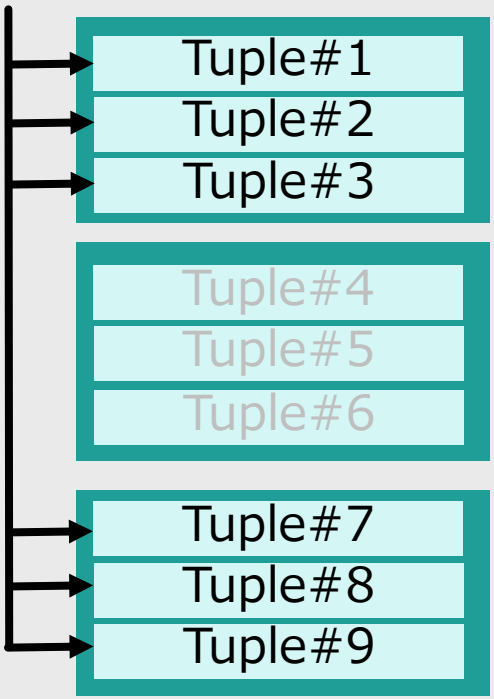
SELECT TABLESAMPLE
サンプリング検索

- テーブル内の一部をサンプリング
  - percent でサンプリング割合をパーセンテージで指定する（0～100）
  - WHERE句を指定した場合は、サンプリング後に評価される
  - SYSTEM
    - ブロック単位でサンプリング（ランダム・スキャン）
    - ブロック内の全タプルを使用
  - BERNOULLI
    - タプルの単位でサンプリング（シーケンシャル・スキャン）
    - SYSTEMよりも正確だがI/O負荷が高い
  - REPEATABLE
    - サンプリング・アルゴリズムに使用する数値を指定
    - 省略時は random(3) 関数による乱数が使用される

```
SELECT … FROM table_name
TABLESAMPLE {SYSTEM | BERNOULLI} (percent) [ REPEATABLE (seed) ]
```

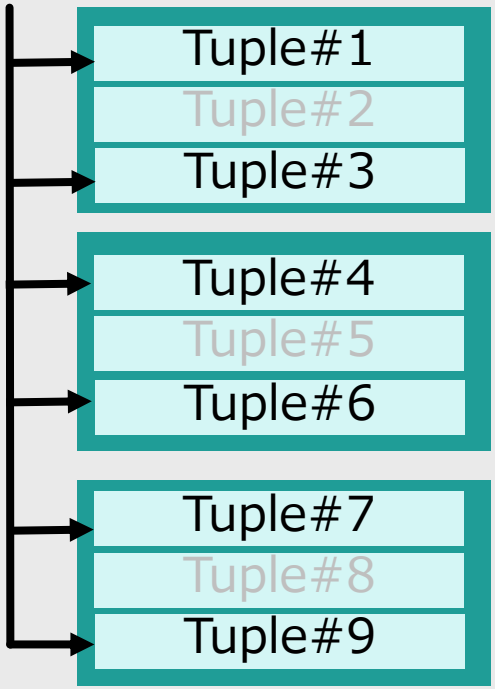
SELECT TABLESAMPLE
サンプリング検索

SYSTEM



サンプル率が1% 以上の場合、Bulk Read

BERNOULLI



常にBulk Read

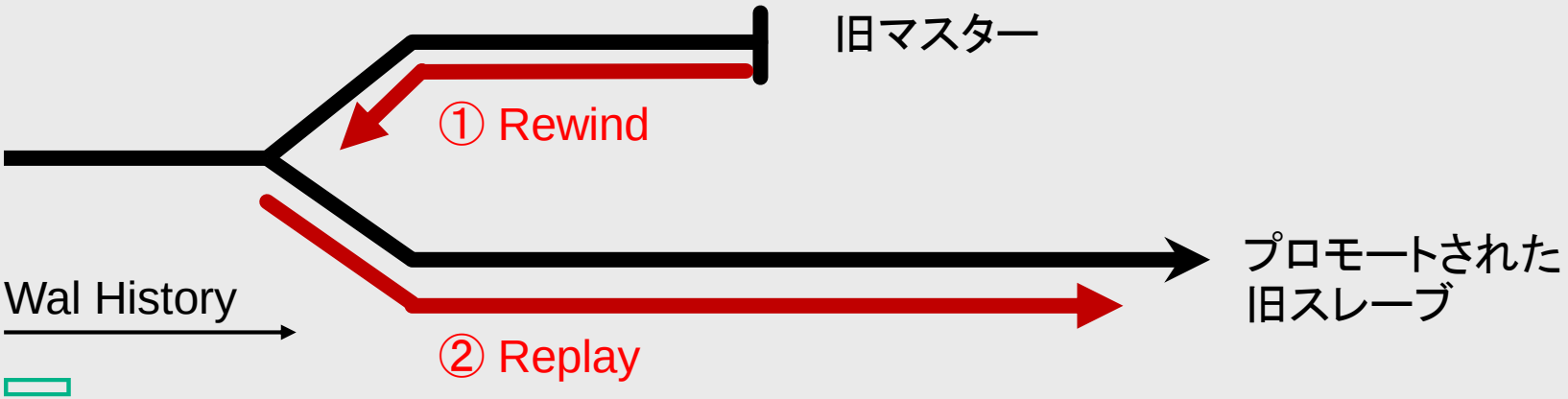
3. 運用を容易にする機能

pg_rewind

レプリケーション環境の再同期

– PostgreSQL 9.4まで
  ① マスター・インスタンスの異常終了
  ② スレーブ・インスタンスを昇格
  ③ 旧マスター・インスタンスのデータを削除し、全データをコピーし再設定

– PostgreSQL 9.5
  ① マスター・インスタンスの異常終了
  ② スレーブ・インスタンスを昇格
  ③ pg_rewindコマンドで旧マスターを差分更新



Hewlett Packard Enterprise

pg_rewind

レプリケーション環境の再同期

– 実行条件
  – 旧マスター・インスタンス側で起動
  – wal_log_hints = on（デフォルトoff）
  – full_page_writes = on （デフォルトon）またはチェックサムの有効化
  – 旧マスター（ターゲット）インスタンスが正常終了していること

```
$ pg_rewind
      --target-pgdata={旧マスターのクラスター}
      --source-server= {新マスター接続情報}
      --dry-run シミュレーション実行
      --progress 進捗状況の出力
      --debug 追加情報の出力
      --help    使用方法の表示
```

ALTER TABLE SET UNLOGGED / LOGGED
更新時のWAL出力量を制御

- 更新処理（INSERT, UPDATE, DELETE）実行時にはWALが出力
  - データベース障害時の復旧に使用
  - pg_xlogディレクトリ内の16MBのファイル群
  - OLTP環境ではパフォーマンス・ボトルネック
- WALを出力しないテーブルも作成できる
  - CREATE TEMPORARY TABLE文（PostgreSQL 9.0～）
  - TEMPORARY句以外は通常のテーブルと同じ
- PostgreSQL 9.5ではLOGGED / UNLOGGED を切り替え可能に
  - 内部的には新規テーブルの作成とデータのコピーを実行

```
ALTER TABLE テーブル名 SET UNLOGGED
ALTER TABLE テーブル名 SET LOGGED
```

4. アプリケーション開発を容易にする機能

INSERT ON CONFLICT
INSERT文で制約違反が発生したらUPDATE文を実行

```
INSERT INTO emploees VALUES (1000, 'Shinoda', 'shinoda@hpe.com')
ON CONFLICT (empid)
DO UPDATE SET ename = EXCLUDED.ename, email = EXCLUDED.email
```

- 主キー列empid = 1000のタプルが存在しなければ
  - INSERT INTO employees VALUES (1000, 'Shinoda', 'shinoda@hpe.com') が実行される
- 主キー列empid = 1000のタプルが存在すれば
  - UPDATE employees SET ename='Shinoda', email='shinoda@hpe.com' WHERE empid=1000 が実行される
- EXCLUDED句は、INSERT INTOで指定した値を指す
- その他の構文
  - 「DO NOTHING」を記述すると、制約違反が発生しても何もしない（エラーが発生しない）
  - 「DO NOTHING」指定時は制約列名を省略できる
  - 「ON CONFLICT ON CONSTRAINT 制約名」を記述すると、制約名を指定できる

INSERT ON CONFLICT
トリガーの動作

– INSERT ON CONFLICT文はINSERT文なのか？UPDATE文なのか？

– トリガーが特殊な動作になる

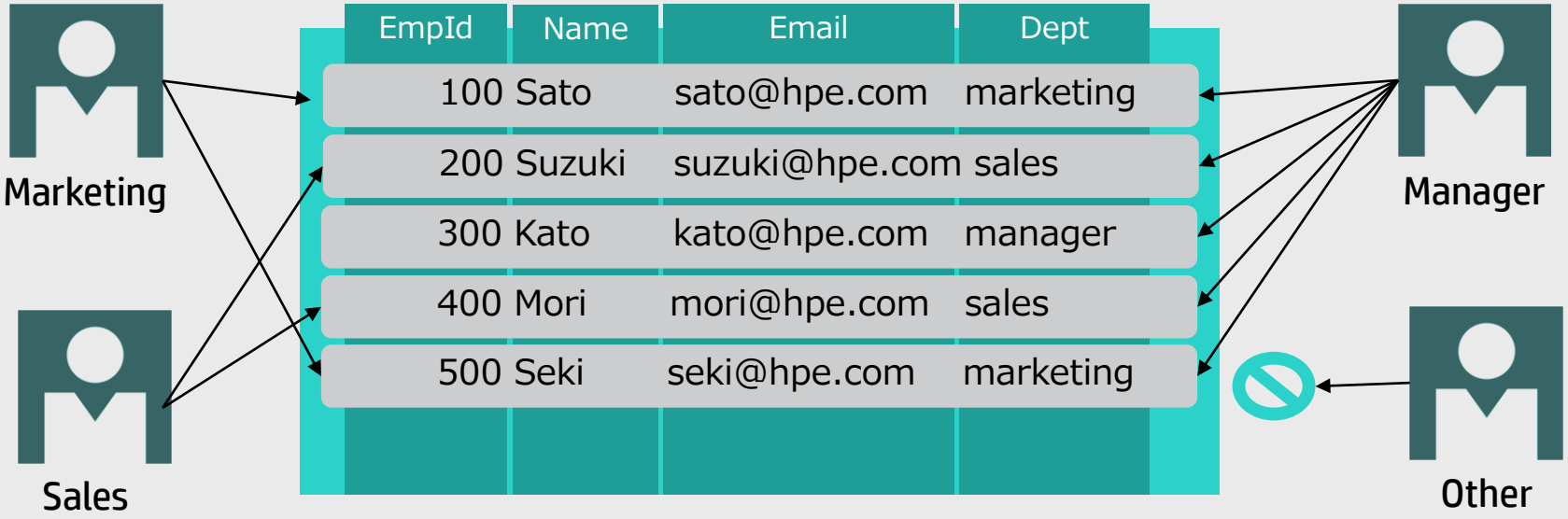
Trigger	INSERT 成功	DO NOTHING	DO UPDATE（更新あり）	DO UPDATE（更新なし）
BEFORE INSERT	実行	実行	実行	実行
AFTER INSERT	実行	-	-	-
BEFORE UPDATE	-	-	実行	-
AFTER UPDATE	-	-	実行	-

– DO UPDATE (更新なし)

  – DO UPDATE句にWHERE句を指定し、更新されなかった場合。

Row Level Security
タプル単位のデータ参照設定

- GRANT文によるアクセス制御
  - テーブル単位
  - 列単位
- Row Level Security (RLS)
  - タプル単位のアクセス制御
  - GRANTによる制限を置き換えるものではなく、追加するもの



Row Level Security
利用方法

– テーブルに対してRLSの有効化

```
ALTER TABLE table_name ENABLE ROW LEVEL SECURITY
```

– ポリシーの作成
  – 対象となるテーブル（ON）
  – 対象となる操作（FOR）
  – 対象となるロール（TO）
  – 許可する検索条件（USING）→ WHERE句条件
  – 許可する更新条件（WITH CHECK）　→ WHERE句条件

```
CREATE POLICY policy_name ON table_name
      [ FOR { ALL | SELECT | INSERT | UPDATE | DELETE } ]
      [ TO role_name | PUBLIC [, …] ]
      [ USING (expression) ]
      [ WITH CHECK (expression) ]
```

Row Level Security
実行例

– ポリシーの作成とRLS有効化

```
postgres(owner)=> ALTER TABLE employees ENABLE ROW LEVEL SECURITY ;
ALTER TABLE
postgres(owner)=> GRANT SELECT ON employees TO PUBLIC ;
postgres(owner)=> CREATE POLICY pol1 ON employees
                  FOR SELECT TO PUBLIC
                  USING (dept = current_user) ;
CREATE POLICY
```

```
postgres(marketing)=> SELECT name FROM emlpoyees ;
 name
---------
 Sato
 Seki
(2 rows)
```

JSONB
演算子と関数の追加

- 「||」演算子
  - 要素の追加／更新を行う

```
postgres=> SELECT '{"key":"key1", "val1":"1000"}'::jsonb ||
     '{"val2":"2000"}' ;
                  ?column?
-----------------------------------------------
 {"key": "key1", "val1": "1000", "val2": "2000"}
(1 row)
```

- 「-」演算子
  - 要素の削除を行う
  - 入れ子構造の要素を削除する「#-」演算子も追加

```
postgres=> SELECT '{"key":"key1", "val1":"1000"}'::jsonb - 'key' ;
     ?column?
------------------
 {"val1": "1000"}
(1 row)
```

JSONB
演算子と関数の追加

- jsonb_set関数
  - 要素の置換／更新を行う

```
postgres=> SELECT
      jsonb_set('{"key":"key1", "val1":"1000"}'::jsonb, '{"val1"}','2000') ;
        jsonb_set
-----------------------------
 {"key": "key1", "val1": 2000}
(1 row)
```

JSONB
演算子と関数の追加

– その他

関数名	機能	備考
jsonb_pretty	整形	
jsonb_strip_nulls	NULL要素の削除	
jsonb_concat	結合	|| 演算子
jsonb_delete	削除	-演算子

UPDATE SET
結合結果による複数列の同時更新構文

– PostgreSQL 9.4 まで

```
UPDATE upd2
SET c2 = upd1.c2, c3 = upd1.c3
FROM
      (SELECT * FROM upd1) AS upd1 WHERE upd1.c1 = upd2.c1
```

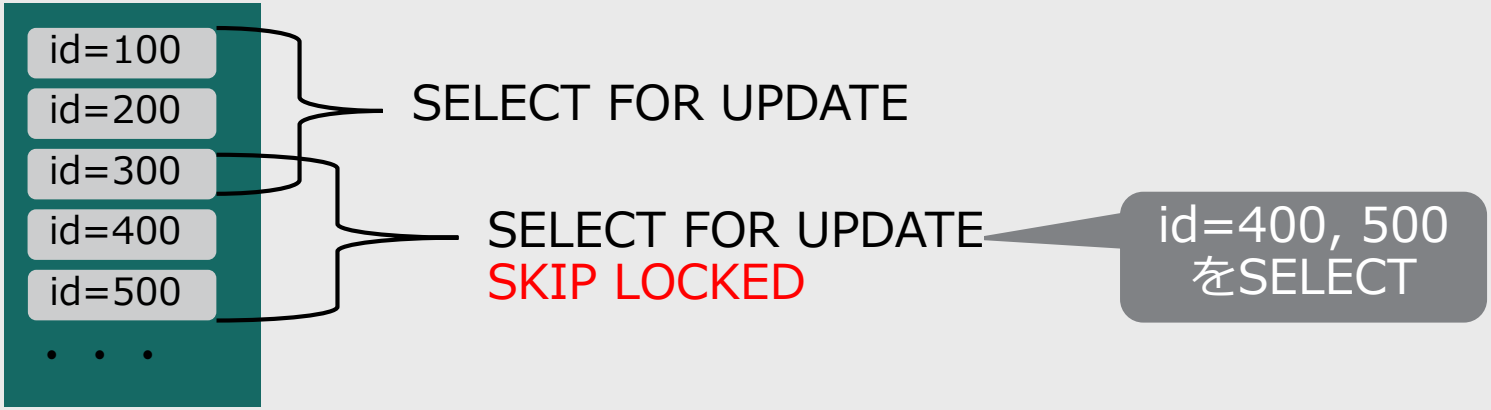
– PostgreSQL 9.5

```
UPDATE upd2
SET (c2, c3) =
      (SELECT c2, c3 FROM upd1 WHERE upd1.c1 = upd2.c1)
```

SELECT SKIP LOCKED
ロックされていないタプルのみ検索

- ロックが競合する場合の動作
  - 待機するかエラーにする
    - SELECT FOR UPDATE同士
    - SELECT FOR UPDATEとSELECT FOR SHARE
    - NOWAITを指定するとエラー
  - ロックしていないタプルのみ検索する
    - PostgreSQL 9.5新機能

```
SELECT … FROM table_name FOR UPDATE SKIP LOCKED
```



Hewlett Packard Enterprise

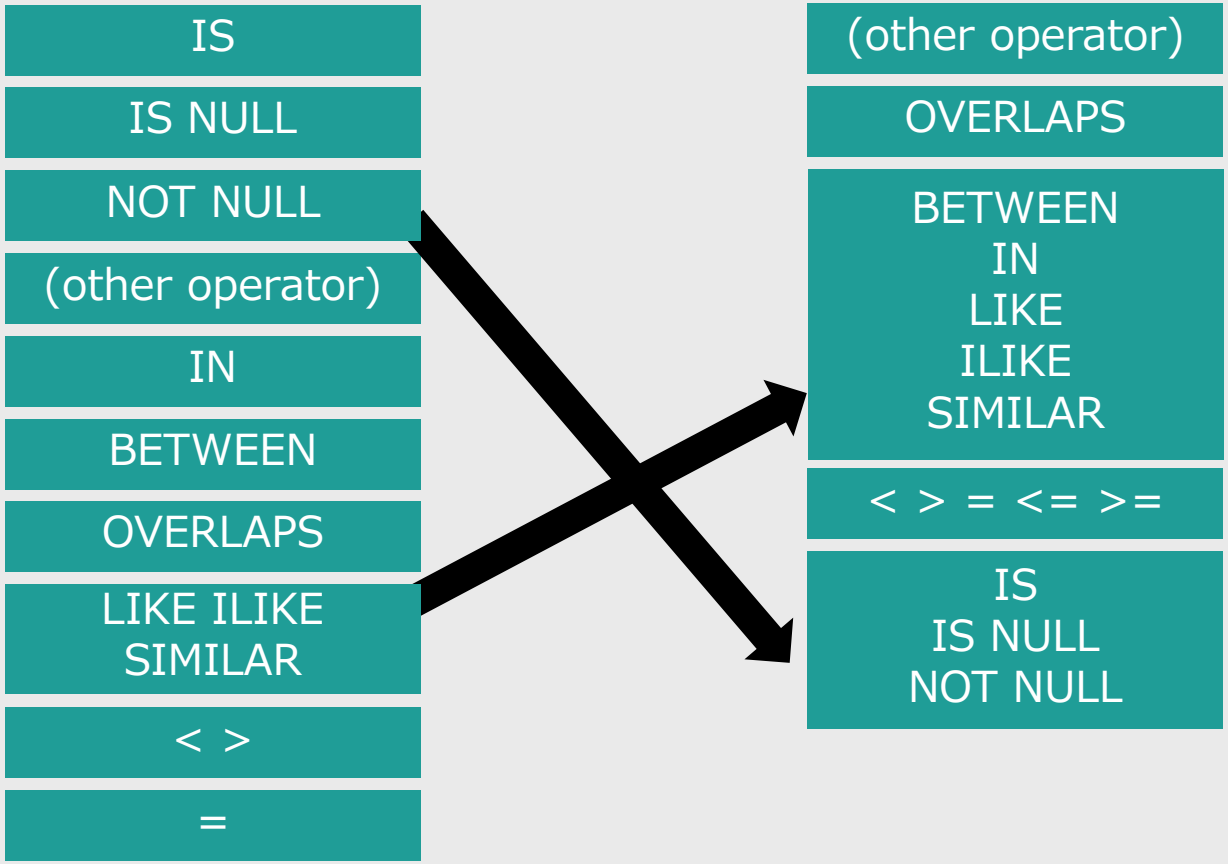
GROUPING SETS / CUBE / ROLLUP
複数の集計単位を一括検索

- GROUP BY句に追加することで複数の集計単位を出力可能
  - GROUPING SETS（集計レコードの指定）
  - CUBE（クロス集計レポート）
  - ROLLUP（小計の出力）
- CUBE, ROLLUPはGROUPING SETSの短縮形

```
GROUP BY GROUPING SETS ((1, 2), 1, 2, ())
= GROUP BY CUBE (1, 2)

GROUP BY GROUPING SETS ((1, 2), 1, ())
= GROUP BY ROLLUP(1, 2)
```

優先順位の変更
演算子の優先順位が変更された



優先順位の変更
演算子の優先順位が変更された

– 優先順位の変更により影響を受けるSQLに警告を出力するパラメーター

operator_precedence_warning （デフォルトoff）

– 実行例

```
postgres=> SET operator_precedence_warning = on;
SET
postgres=> SELECT COUNT(*) FROM sample1 WHERE c1 > 10 IS true  ;
WARNING:  operator precedence change: IS is now lower precedence than >
LINE 1: SELECT COUNT(*) FROM sample1 WHERE c1 > 10 IS true ;

count
--------
 999990
(1 row)
```

PL/pgSQL ASSERT
アサーション

– 優先順位の変更により影響を受けるSQLに警告を出力するパラメーター

```
ASSERT condition [, message ]
```

– アサーション
  – condition部分がFalseまたはNullになると例外（ASSERT_EXCEPTION）が発生する
  – FUNCTIONのパラメータ・チェック
  – デバッグ
– パラメーターplpgsql.check_asserts
  – ASSERT文有効（デフォルト）
  – 無効にするにはoffに指定

5. アーキテクチャの変更

パラメーターの変更
追加されたパラメーター

パラメータ名	説明
max_wal_size	チェックポイントの開始サイズ（checkpoint_segments廃止）
min_wal_size	WALリサイクルを行うサイズ
cluster_name	プロセス名の指定
gin_pending_list_limit	GINインデックスの待機リスト最大値
row_security	Row Level Security機能の有効化
track_commit_timestamp	トランザクションのコミット時間の出力
wal_compression	WAL圧縮機能の有効化
log_replication_commands	レプリケーション関連ログの出力
operator_precedence_warning	優先順位の変更影響に関する警告
wal_retrieve_retry_interval	WALデータの再取得間隔の指定

WAL圧縮
WAL出力量の削減

– WALを圧縮して出力する機能
  – Full Page Write （ページに対するチェックポイント後の最初の書き込み）時に圧縮
  – パラメーターwal_compressionをonに指定することで有効化（デフォルトoff）

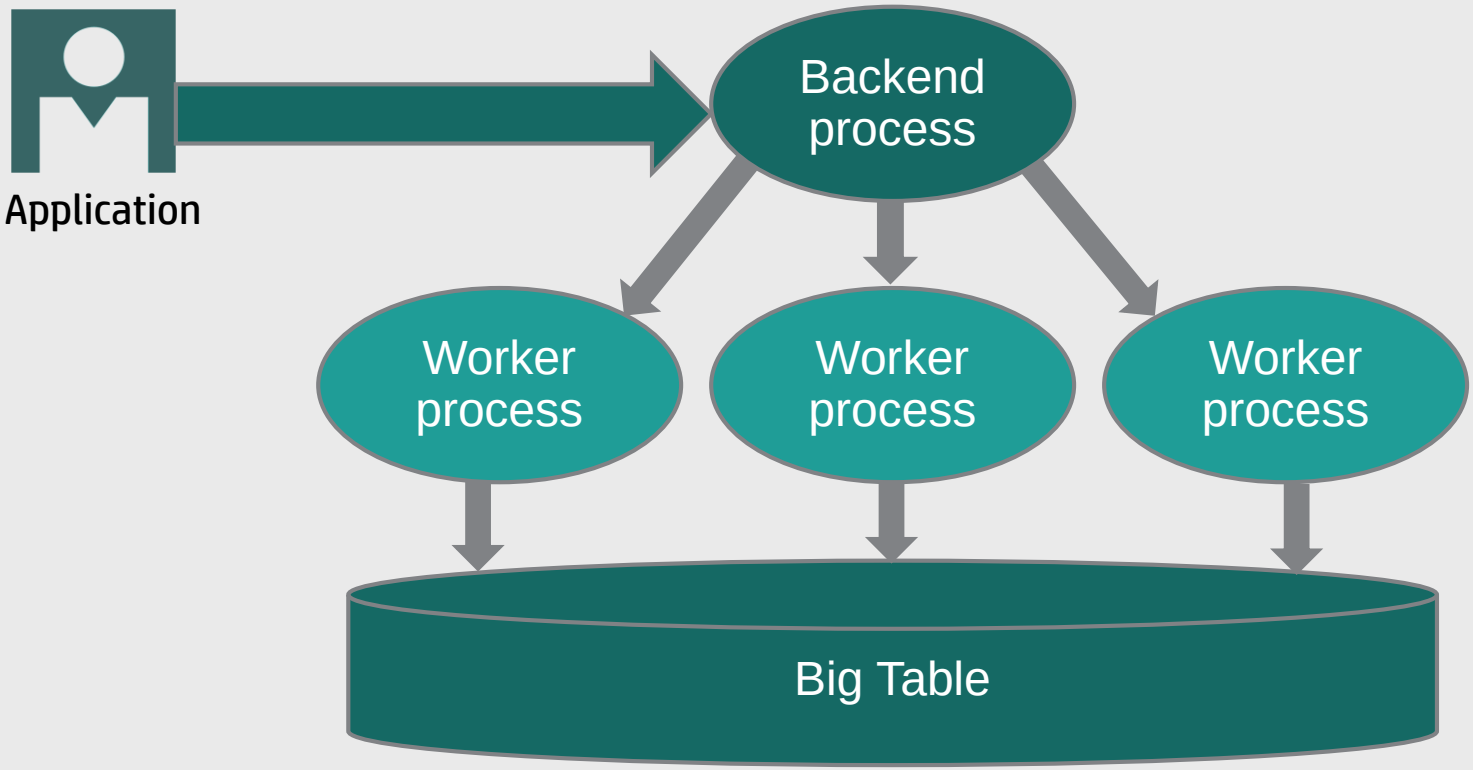
パッケージの変更
Contribモジュールからbinへ

– 以下のコマンドはContribモジュールからPostgreSQL本体へ移動された

コマンド名	説明
pg_archivecleanup	不要なアーカイブログの削除
pg_test_fsync	wal_sync_methodの最適解をチェック
pg_test_timing	時間計測のオーバーヘッドをチェック
pg_upgrade	データベース・クラスタのバージョンアップ
pg_xlogdump	WALのダンプ
pgbench	簡易ベンチマーク・プログラム

6. PostgreSQL 10?

Parallel Seq Scan
並列検索



Parallel Seq Scan
並列検索

– 実行計画

```
postgres=# SET max_parallel_degree = 10 ;
SET
postgres=# EXPLAIN SELECT * FROM data1 WHERE col2='1000' ;
                           QUERY PLAN
-----------------------------------------------------------------------
Gather  (cost=1000.00..40789.71 rows=1 width=11)
  Number of Workers: 4
  ->  Parallel Seq Scan on data1  (cost=0.00..39789.61 rows=1 width=11)
        Filter: ((c2)::text = '1000'::text)
```

– 永安さんのブログで紹介

– http://pgsqldeepdive.blogspot.jp/2015/12/parallel-seq-scan.html

Native Partition Table
継承を使用しないパーティション・テーブル

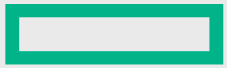
– CREATE TABLE table_name PARTITION BY RANGE (column) …

– CREATE TABLE table_name PARTITION BY LIST(column) …

7. まとめ

まとめ

- PostgreSQL 9.5には、魅力的な新機能が数多く採用された
  - パフォーマンスの向上
  - 大規模環境に対応した新機能
  - アプリケーション開発を容易にする新機能
- 参考URL
  - Commitfests
    http://commitfest.postgresql.org/
  - PostgreSQL 9.5新機能紹介（澤田さん）
    http://www.slideshare.net/hadoopxnttdata/postgresql-95-new-features-nttdata
  - Michael Paquierさんのブログ
    http://michael.otacoo.com/
  - ぬこ@横浜さんのブログ
    http://d.hatena.ne.jp/nuko_yokohama/
  - Performance improvements in PostgreSQL 9.5 (and beyond)
    http://www.slideshare.net/fuzzycz/performance-improvements-in-postgresql-95-and-beyond



Hewlett Packard
Enterprise

Thank you

noriyoshi.shinoda@hpe.com