

メジャーバージョンアップした PostgreSQL 10の機能紹介

Noriyoshi Shinoda

November 26, 2017



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・パカード株式会社 Pointnext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
 - <https://community.hpe.com/t5/forums/searchpage/tab/message?q=%E8%99%8E%E3%81%AE%E5%B7%BB>
- Oracle ACEってどんな人？
 - <http://www.oracle.com/technetwork/jp/database/articles/vivadeveloper/index-1838335-ja.html>

Agenda

PostgreSQL 10の機能紹介

1. PostgreSQL 10概要
2. バージョン表記の変更
3. Logical Replication
4. パーティショニング
5. パラレル・クエリーの拡張
6. SQL文
7. その他
8. 非互換

まとめ





1. PostgreSQL 10概要



1. PostgreSQL 10概要

PostgreSQLの概要

- オープンソースで開発されているRDBMS
 - MySQLやFirebird等の仲間
- ライセンスはPostgreSQL License
 - ≒BSD License
- 活発な開発コミュニティ
 - The PostgreSQL Global Developer Group (<http://www.postgresql.org/>)
 - 日本PostgreSQLユーザ会 (<http://www.postgresql.jp/>)
 - PostgreSQL Enterprise Consortium (<http://www.pgecons.org/>)
 - Commitfests (<https://commitfest.postgresql.org/>)
- 最新バージョン
 - PostgreSQL 10 (10.1) ← 今日のお話



1. PostgreSQL 10概要

PostgreSQLの歴史

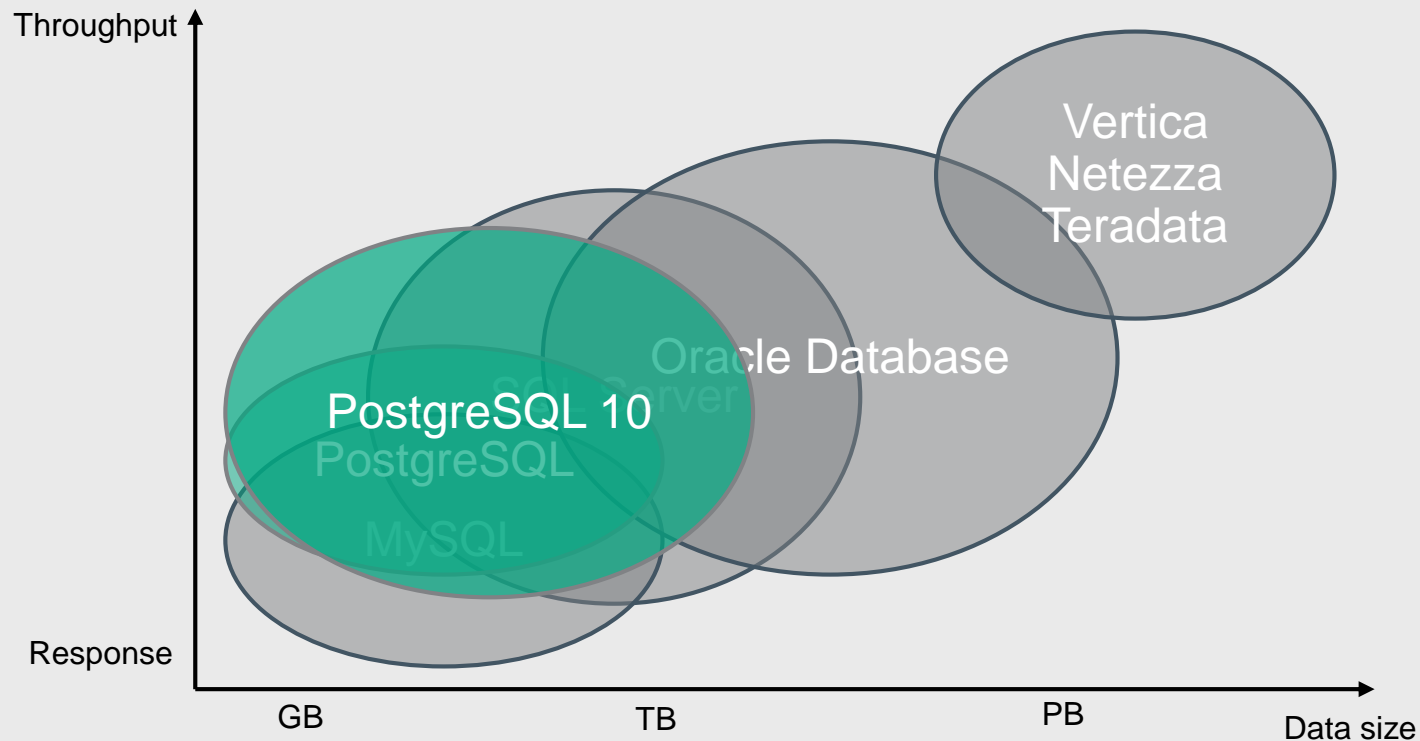
- 1974年 Ingres プロトタイプ (UCB)
 - HPE NonStop SQL, SAP Sybase ASE, Microsoft SQL Serverの元になる
- 1989年 POSTGRES 1.0～
- 1997年 PostgreSQL 6.0～
 - GEQO, MVCC, マルチバイト, Netezza (PureData) の分離
- 2005年 PostgreSQL 8.0～
 - 自動VACUUM, HOT, PITR, Vertica, ParAccel (RedShift) の分離
- 2010年 PostgreSQL 9.0～
 - レプリケーション, 外部表, JSON, マテリアライズド・ビュー
- 2016年5月 PostgreSQL 9.6
 - パラレル・クエリー, マルチ・インスタンス同期レプリケーション
- 2017年10月 PostgreSQL 10



1. PostgreSQL 10概要

PostgreSQL 10の特徴

– レプリケーション機能と大規模環境への適用機能が拡張されました





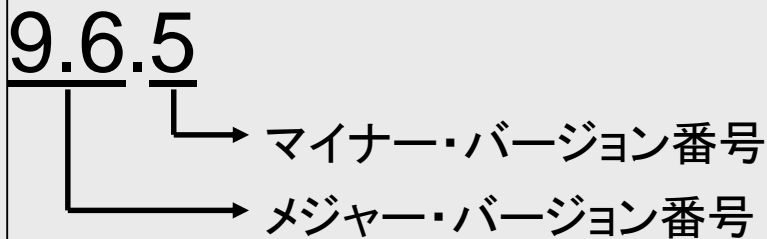
2. バージョン表記



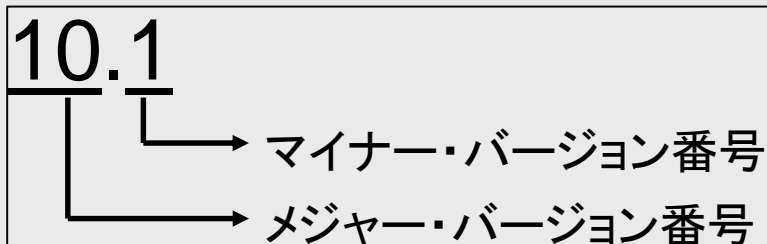
2. バージョン表記

メジャー・バージョンの変更

- これまでは
 - 2つの数字がメジャー、最後の数字がマイナー



- これからは
 - 最初の数がメジャー、最後の数がマイナー





3. Logical Replication

3. Logical Replication

Logical Replicationとは？

– Logical Replicationとは？

- テーブル単位のレプリカ作成機能
- レプリケーション先のテーブルもRead / Write可能
- SQL文の結果が同一であることを保証(=Logical)
- ≡ Slony-I
- ≡ MySQLのRow-based Replication (RBR)

– Streaming Replication (Physical Replication) とは？

- PostgreSQL 9.0以降利用可能
- データベース・クラスタ全体のレプリカ作成機能
- レプリケーション先インスタンスは更新不可 (INSERT / UPDATE / DELETE実行不可)
- 物理的に同一ブロックを作成(=Physical)



3. Logical Replication

Logical Replicationとは？

－ 利用シナリオ

- － 拠点ごとにデータベースを持つ構成でマスター・テーブルだけ共有したい
- － 分析用データベースにインデックスを追加したい



3. Logical Replication

Logical Replicationとは？

- 同じである必要があること

- スキーマ名
- テーブル名
- 列名
- 列データ型(暗黙の型変換ができれば違っていても可)

- 違っていても良いこと

- データベース名
- 文字エンコーディング(UTF-8, 日本語EUC等)
- 列の定義順序
- インデックスの追加
- 制約の追加
- 列の追加(レプリケーション先)



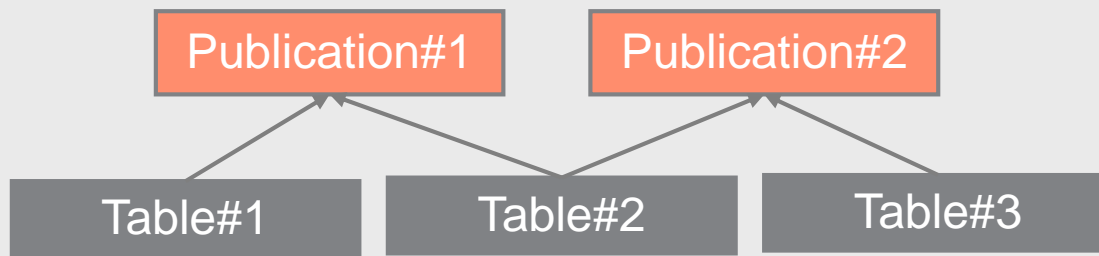
3. Logical Replication

新しいオブジェクト

– PUBLICATION

- データを提供するデータベースに作成
- レプリケーション対象テーブルを指定(複数可、全体指定可)
- レプリケーション対象DML (INSERT, UPDATE, DELETE)を指定
 - デフォルトは全DMLをレプリカ
- CREATE PUBLICATION文を実行
- データベースに対するCREATE権限+レプリカ対象テーブルの所有者であるか、またはSUPERUSER属性が必要

– PUBLICATIONとテーブルの関係は柔軟に構成できる

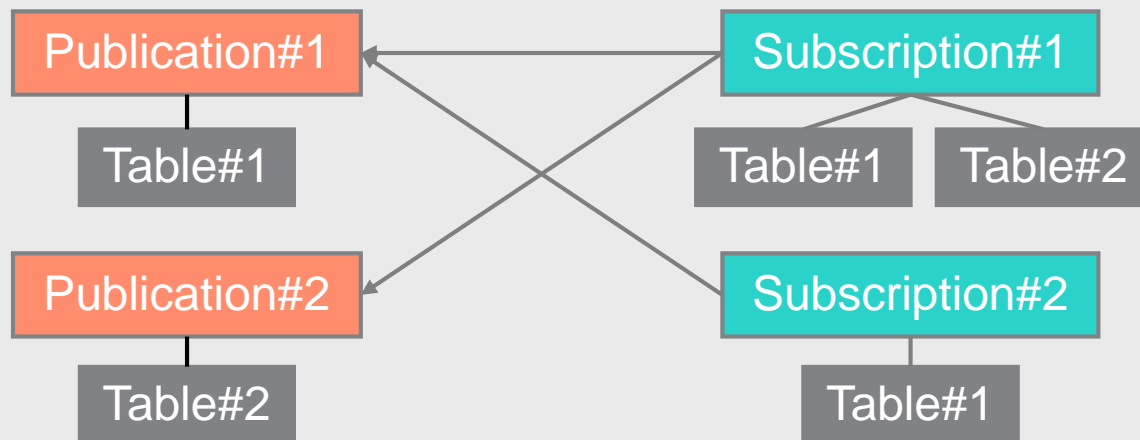


3. Logical Replication

新しいオブジェクト

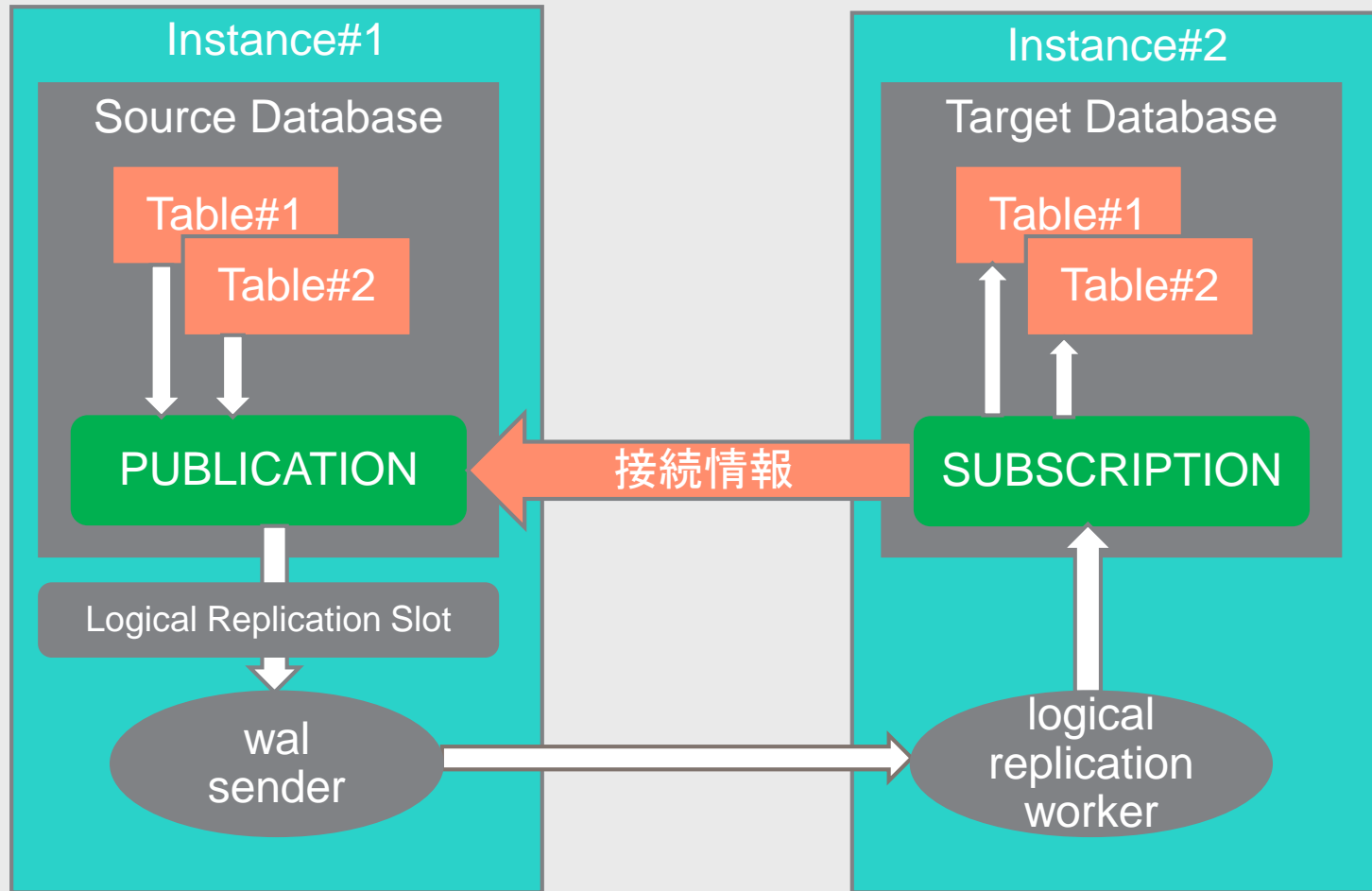
– SUBSCRIPTION

- データ受信するデータベースに作成
 - PUBLICATION側インスタンスへの接続情報を指定
 - 接続先PUBLICATION名を指定
 - CREATE SUBSCRIPTION文を実行
 - SUPERUSER属性が必要
- PUBLICATIONとSUBSCRIPTIONの関係は柔軟に構成できる



3. Logical Replication

オブジェクト関連図



3. Logical Replication

作成例(PUBLICATION側の操作)

– レプリケーション用ユーザー作成

- REPLICATION属性LOGIN属性を持つロールを作成。テーブル所有者にCREATE権限付与

```
postgres=# CREATE ROLE repusr PASSWORD 'passwd' REPLICATION LOGIN;  
CREATE ROLE  
postgres=# GRANT CREATE ON DATABASE postgres TO srcusr ;  
GRANT
```

– pg_hba.confファイルの編集

- レプリケーション用ユーザーの接続を許可

```
host all repusr 192.168.1.100/32 trust
```

– パラメーターの設定

```
postgres=> SHOW wal_level ;  
wal_level  
-----  
logical
```

3. Logical Replication

作成例(PUBLICATION側の操作)

- アプリケーション用テーブルを作成

- 主キーの指定を推奨

```
postgres=> CREATE TABLE data1(c1 INT PRIMARY KEY, c2 VARCHAR(10));  
CREATE TABLE
```



- 接続ユーザーにテーブルの検索を許可(初期データ移行用)

```
postgres=> GRANT SELECT ON data1 TO repuser ;  
GRANT
```

- PUBLICATIONを作成

- レプリケーション対象テーブルを指定

```
postgres=> CREATE PUBLICATION pub1 FOR TABLE data1 ;  
CREATE PUBLICATION
```

3. Logical Replication

作成例 (SUBSCRIPTION側の操作)

- アプリケーション用テーブルを作成
 - 原則としてレプリケーション元と同一構成

```
postgres=> CREATE TABLE data1 (c1 INT PRIMARY KEY, c2 VARCHAR(10));  
CREATE TABLE
```

- SUBSCRIPTIONを作成
 - PUBLICATION側インスタンスへの接続情報とPUBLICATION名を指定
 - PUBLICATION側のテーブルから初期データのコピーが行われる

```
postgres=# CREATE SUBSCRIPTION sub1  
CONNECTION 'host=srchost1 port=5432 dbname=postgres user=repusr'  
PUBLICATION pub1 ;  
NOTICE: created replication slot "sub1" on publisher  
CREATE SUBSCRIPTION
```

3. Logical Replication

作成例(レプリケーションの確認)

– PUBLICATION側

– pg_stat_replicationビューを確認

```
postgres=# SELECT application_name, state FROM pg_stat_replication;  
application_name | username | state  
-----+-----+-----  
sub1              | repusr   | streaming
```

– SUBSCRIPTION側

– pg_stat_subscriptionビューを確認(PostgreSQL 10新機能)

```
postgres=> SELECT subname, received_lsn FROM pg_stat_subscription ;  
subname | received_lsn  
-----+-----  
sub1    | 0/25434768
```

3. Logical Replication

制約

- レプリケーションできないオブジェクト
 - SEQUENCE
 - MATERIALIZED VIEW
 - INDEX (レプリカ先テーブルが更新されることで自動メンテナンスされる)
 - SERIAL列 (列の値はレプリケーションされる)
 - UNLOGGED TABLE
 - Large Object
- レプリケーションできない構文
 - TRUNCATE文
 - ALTER TABLE文等のDDL
- 構成上の制約
 - 同一名テーブル間の相互レプリケーション不可
 - 単一インスタンス内のレプリケーションは可能だが構成時に制限あり (マニュアルに記載)
 - デフォルトでは非同期レプリケーションになる (同期設定は可能)

3. Logical Replication

制約

– トリガー

- レプリケーションによる更新ではSTATEMENT TRIGGERが実行できない
- デフォルトではROW TRIGGERが実行されない
 - ALTER TABLE ENABLE REPLICA TRIGGERを実行する必要がある



3. Logical Replication

衝突

- 双方のテーブルは更新可能であるため衝突が発生する可能性がある
 - 主キー／一意キー違反
 - CHECK制約違反
- 衝突が発生しないパターン
 - PUBLICATION側でUPDATE/DELETE文を実行したがSUBSCRIPTION側にデータ無し
- 待機が発生するパターン
 - LOCK TABLE
 - トランザクションの確定待ち
- 衝突が発生すると
 - SUBSCRIPTION側のlogical replication workerプロセスが停止
 - 5秒待機
 - 再起動

```
ERROR: duplicate key value violates unique constraint "data1_pkey"
```

```
DETAIL: Key (c1)=(14) already exists.
```

```
LOG: worker process: logical replication worker for subscription 16399 (PID 3626)  
exited with exit code 1
```

3. Logical Replication

衝突

- 衝突を解消する方法
 - SUBSCRIPTION側で衝突したデータを削除
 - レプリケーション開始位置 (LSN) を衝突後の位置に進める
 - `pg_replication_origin_advance`関数を使う
 - 衝突が解消されるとレプリケーションが自動的に再開される





4. パーティショニング

4. パーティショニング

概要

– パーティショニングとは

- 大規模テーブルを複数に物理分割する機能（分割方法は一般的には列値）
- アプリケーションからは単独のテーブルに見える
- テーブルに対してSQL文が発行されると自動的に参照するパーティションを特定する（**パーティション・プルーニング**）
- PostgreSQL 10では**Declarative Table Partitioning**と呼ばれる

– 分割方法はテーブル単位に選択できる

- 列値の範囲で分割
 - RANGEパーティション
- 列値の値で分割
 - LISTパーティション
- ハッシュ値で分割
 - PostgreSQL 10では未実装（PostgreSQL 11で実装予定）

4. パーティショニング

新旧バージョンの比較

- PostgreSQL 9.6まで
 - アプリケーションからアクセスする親テーブルを作成
 - 親テーブルを継承(**INHERIT**)する子テーブルを複数作成
 - 親テーブルに対する**INSERTトリガー**を使って子テーブルにデータを分散INSERT
 - パフォーマンスの問題あり
 - 子テーブルの列に**CHECK制約**を付与することより、格納する値を制限
 - 変更運用の問題あり
- PostgreSQL 10以降
 - 親テーブルに**分割方法**(RANGEまたはLIST)と列を指定
 - 子テーブルに**分割条件**(列値)を指定(INHERIT指定は不要)
- 子テーブルにも直接アクセスできるのは従来と同様



4. パーティショニング


LISTパーティション作成例

```
postgres=> CREATE TABLE plist1 (c1 NUMERIC, c2 VARCHAR(10))
           PARTITION BY LIST (c1) ;
CREATE TABLE
postgres=> CREATE TABLE plist1_p1 PARTITION OF plist1 FOR VALUES IN (100) ;
CREATE TABLE
postgres=> CREATE TABLE plist1_p2 PARTITION OF plist1 FOR VALUES IN (200) ;
CREATE TABLE
postgres=> \d plist1
Table "public.plist1"
 Column |          Type          | Collation | Nullable | Default
-----+-----+-----+-----+-----
 c1     | numeric                |           |          |
 c2     | character varying(10)  |           |          |
Partition key: LIST (c1)
Number of partitions: 2 (Use \d+ to list them.)
```

4. パーティショニング

RANGEパーティション作成例

```
postgres=> CREATE TABLE prange1 (c1 NUMERIC, c2 VARCHAR(10)) PARTITION BY RANGE (c1) ;
CREATE TABLE
postgres=> CREATE TABLE prange1_p1 PARTITION OF prange1 FOR VALUES FROM (100) TO (200) ;
CREATE TABLE
postgres=> CREATE TABLE prange1_p2 PARTITION OF prange1 FOR VALUES FROM (200) TO (300) ;
CREATE TABLE
postgres=> \d prange1
Table "public.prange1"
Column |          Type          | Collation | Nullable | Default
-----+-----+-----+-----+-----
c1      | numeric                |           |          |
c2      | character varying(10)  |           |          |
Partition key: RANGE (c1)
Number of partitions: 2 (Use \d+ to list them.)
```



- パーティションにはFROM 以上、TO未満の値が格納される
- 上限や下限を規定しない場合はMINVALUE / MAXVALUE を指定

4. パーティショニング

既存テーブルをパーティション登録／解除

```
postgres=> ALTER TABLE plist1 ATTACH PARTITION plist1_p3 FOR VALUES IN  
(300) ;  
ALTER TABLE  
postgres=> ALTER TABLE plist1 DETACH PARTITION plist1_p3 ;  
ALTER TABLE
```


- ATTACH PARTITIONで指定するテーブルは、他のパーティションと同一構造(列名、データ型)が一致している必要がある
- ATACH PARTITION実行時には格納済のデータはFOR VALUES句に合致しているかチェックされる



4. パーティショニング 制約

- 親テーブルに対するインデックスは作成できない
- 親テーブルに対する主キー制約／一意キー制約を指定できない
- 更新値がパーティションをまたがるUPDATE文を実行できない(PostgreSQL 11で実装されるかも=Needs Review Status)
- 「その他」の値を格納するパーティションを定義できない(PostgreSQL 11で実装予定)
 - パーティションに含まれない値を持つINSERT文はエラー



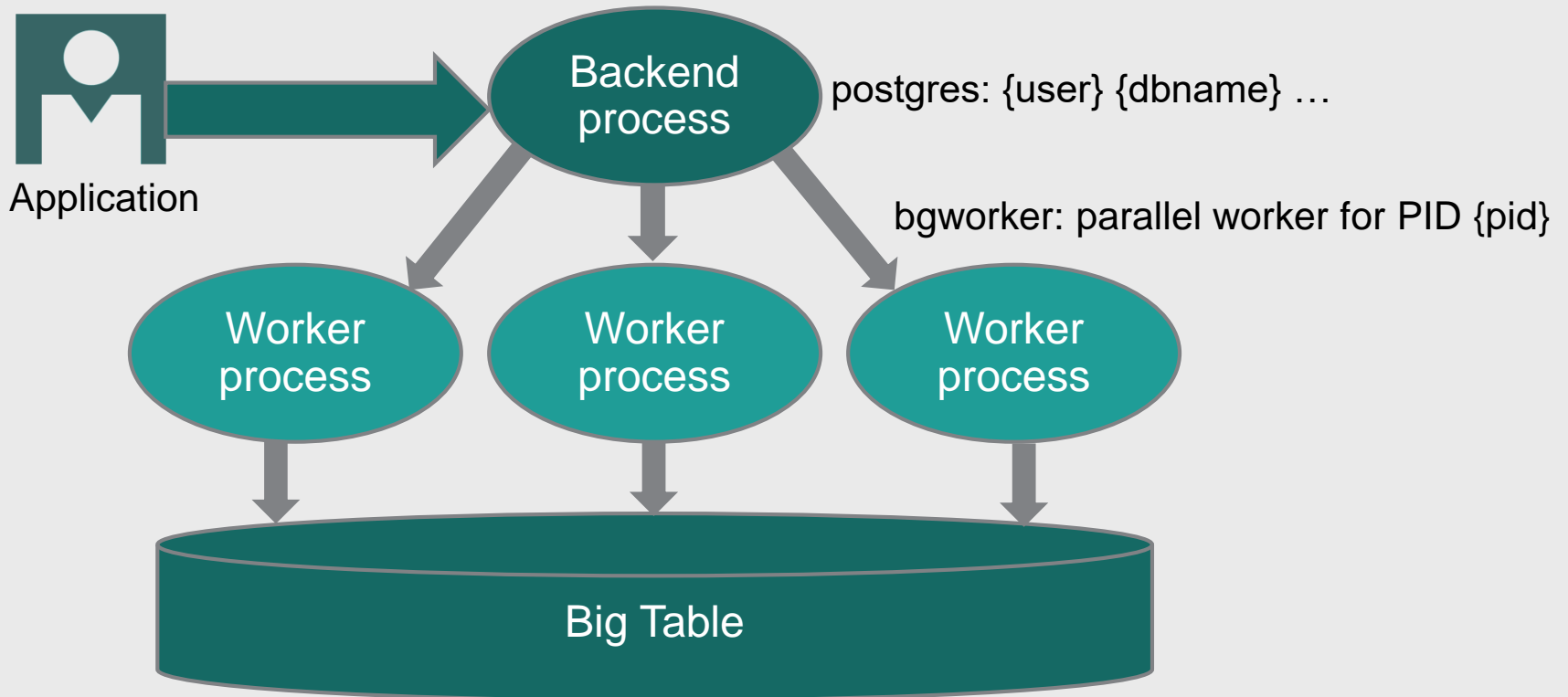


5. パラレル・クエリーの拡張

5. パラレル・クエリーの拡張

パラレル・クエリーとは

- マルチ・プロセスによるSQL文の並列処理
- PostgreSQL 9.6で実装



5. パラレル・クエリーの拡張


利用可能な条件の追加

– 実行可能な条件

- Seq Scan (PostgreSQL 9.6～)
- **PREPARE / EXECUTE (追加)**
- **Index Scan / Index Only Scan / Nested Loop Join / Hash Join (追加)**
- **Merge Join / Gather Merge (追加)**
- **Bitmap Heap Scan (追加)**
- **SubPlan (追加)**
- **COPY文 (追加)**

– パラメーターの追加／変更(デフォルト値)

- max_parallel_workers (8)
- max_parallel_workers_per_gather (2) **デフォルト値が変更された**
- min_parallel_table_scan_size (8MB)
- min_parallel_index_scan_size (512kb)

 min_parallel_relation_size (廃止)

5. Parallel Queryの拡張

利用可能な条件の追加

– 実行計画の例

```
postgres=# EXPLAIN COPY (SELECT COUNT(*) FROM copy1) TO '/tmp/copy1.csv' ;
```

```
LOG:  duration: 42.880 ms  plan:
```

```
Query Text: COPY (SELECT COUNT(*) FROM copy1) TO '/tmp/copy1.csv' ;
```

```
Finalize Aggregate  (cost=11614.55..11614.56 rows=1 width=8)
```

```
    -> Gather  (cost=11614.33..11614.54 rows=2 width=8)
```

```
        Workers Planned: 2
```

```
            -> Partial Aggregate  (cost=10614.33..10614.34 rows=1
```

```
width=8)
```

```
                -> Parallel Seq Scan on copy1  (cost=0.00..9572.67
```

```
rows=416667 width=0)
```



6. SQL文

6. SQL文

GENERATED句

- 整数自動生成列
 - CREATE TABLE文の列定義に指定
 - データ型に指定できるのはsmallint, integer, bigintのみ
 - GENERATED BY DEFAULT
 - 自動生成された列は更新可
 - GENERATED ALWAYS
 - 自動生成された列は更新不可
- 内部的にはSERIAL列と同じ
 - SEQUENCEオブジェクトが作成される
- 主キー制約やインデックスは自動設定されない
- SQL標準構文に準拠する目的
 - DB2 / Oracle Databaseでは実装済



6. SQL文 GENERATED句

– GENERATED BY DEFAULT

```
postgres=> CREATE TABLE gen2 (c1 BIGINT GENERATED BY DEFAULT AS IDENTITY, c2 VARCHAR(10));  
CREATE TABLE
```

```
postgres=> \d gen2
```

Table "public.gen2"				
Column	Type	Collation	Nullable	Default
c1	bigint		not null	generated by default as identity
c2	character varying(10)			

```
postgres=> INSERT INTO gen2(c2) VALUES ('data1') ;
```

```
INSERT 0 1
```

```
postgres=> SELECT * FROM gen2 ;
```

```
 c1 | c2  
----+-----  
  1 | data1
```

```
postgres=> UPDATE gen2 SET c1=100 WHERE c2='data1' ;
```

```
UPDATE 1
```

6. SQL文 GENERATED句

– GENERATED ALWAYS

```
postgres=> CREATE TABLE gen1 (c1 BIGINT GENERATED ALWAYS AS IDENTITY, c2 VARCHAR(10)) ;  
CREATE TABLE
```

```
postgres=> \d gen1
```

Table "public.gen1"				
Column	Type	Collation	Nullable	Default
c1	bigint		not null	generated always as identity
c2	character varying(10)			

```
postgres=> INSERT INTO gen1(c2) VALUES ('data1') ;
```

```
INSERT 0 1
```

```
postgres=> UPDATE gen1 SET c1=100 WHERE c2='data1' ;
```

```
ERROR: column "c1" can only be updated to DEFAULT
```

```
DETAIL: Column "c1" is an identity column defined as GENERATED ALWAYS.
```

6. SQL文

CREATE STATISTICS文

- 複数列の相関を示す統計情報の作成
- 実際に計算されるのはANALYZE文実行時
- 一意な値 (ndistinct)、相関係数 (dependencies) を収集可能
- 取得された情報はpg_statistic_extカタログで参照

```
postgres=> CREATE STATISTICS stat1 ON c1, c2 FROM data1 ;
```

```
CREATE STATISTICS
```

```
postgres=> \d data1
```

Table "public.data1"

Column	Type	Collation	Nullable	Default
c1	numeric			
c2	numeric			
c3	character varying(10)			

```
Statistics objects:
```

```
"public"."stat1" (ndistinct, dependencies) ON c1, c2 FROM data1
```




7. その他

7. その他

同期レプリケーションの拡張

- synchronous_standby_names パラメーター
 - PostgreSQL 9.5 以前
 - 同期レプリケーションは単一インスタンスのみ
 - 設定方法 *application_name1, application_name2, ...*
 - PostgreSQL 9.6
 - 複数の同期レプリケーション・インスタンス数を指定可能
 - パラメータ記述順で同期レプリケーション対象を決定
 - 設定方法 *num_sync (application_name1, application_name2, ...)*
 - num_sync は同期レプリケーション数
 - PostgreSQL 10
 - ANY を指定するとアプリケーション名の記述順は関係なくなる
 - FIRST または省略した場合は PostgreSQL 9.6 と同じ動作
 - 設定方法 **FIRST** | **ANY** *num_sync (application_name1, application_name2, ...)*
 - num_sync は同期レプリケーション数



7. その他

HASHインデックスの拡張

– HASHインデックスとは？

- 列値のハッシュ値を元に作成されるインデックス
- 同値検索に使用できる
- ストレージ要求がBtreeインデックスよりも小さくできる

– PostgreSQL 10の拡張

- トランザクション・ログが出力される
- ストリーミング・レプリケーション環境で使用可能になった



7. その他

Foreign Data Wrapperの拡張

- Foreign Data Wrapperとは？
 - PostgreSQL以外の外部システムに対して処理を自動転送
 - リモートのPostgreSQL / Oracle Database / ファイル等にアクセス可能
- PostgreSQL 10の拡張
 - 集約処理(COUNT / SUM / AVG / GROUP BY等)をリモートで実行可能
 - postgres_fdwモジュールが対応済

```
statement: START TRANSACTION ISOLATION LEVEL REPEATABLE READ
execute <unnamed>: DECLARE c1 CURSOR FOR
        SELECT count(*), avg(c1), sum(c1) FROM public.datar1
statement: FETCH 100 FROM c1
statement: CLOSE c1
statement: COMMIT TRANSACTION
```



8. 非互換

8. 非互換 ディレクトリ名の変更

- データベース・クラスタ内のディレクトリ名変更
 - ログ・ファイルはパラメーターlog_directoryで変更可能

カテゴリー	PostgreSQL 9.6	PostgreSQL 10
トランザクション・ログ	pg_xlog	pg_wal
ログ・ファイル	pg_log	log
コミット・ログ	pg_clog	pg_xact

- 名称の統一
 - xlogからwalへ
 - locationからlsnへ



8. 非互換 コマンド名の変更

– コマンド名変更

- コマンドおよびパラメーター名指定の変更
- 一部の関数でも「xlog」から「wal」への変更が行われた

PostgreSQL 9.6	PostgreSQL 10
pg_receivexlog	pg_receivewal
pg_resetxlog	pg_resetwal
pg_xlogdump	pg_waldump
pg_basebackup --xlog-method	pg_basebackup --wal-method
pg_basebackup --xlogdir	pg_basebackup --waldir
initdb --xlogdir	initdb --waldir
initdb --noclean / --nosync	initdb --no-clean / --no-sync
createlang / droplang	廃止
createuser	-Nオプション廃止

8. 非互換

ユーティリティの動作変更

– pg_ctlコマンド

- インスタンス起動時を含めすべての処理で待機(--wait)がデフォルトに
- インスタンス起動時にアドレス(ポート)情報が表示される

```
$ pg_ctl -D data start
waiting for server to start....
2017-11-12 13:12:57.323 JST [4297] LOG:  listening on IPv6 address ":::1", port 5432
2017-11-12 13:12:57.323 JST [4297] LOG:  listening on IPv4 address "127.0.0.1", port 5432
2017-11-12 13:12:57.325 JST [4297] LOG:  listening on Unix socket "/tmp/.s.PGSQL.5432"
2017-11-12 13:12:57.339 JST [4297] HINT:  Future log output will appear in directory "log".
done
server started
```



8. 非互換

ユーティリティの動作変更

- pg_basebackupコマンド
 - -xパラメーター廃止
 - --wal-method=streamがデフォルトに
 - 複数のwal senderプロセスを使う(max_wal_sendersパラメーターの不足に注意)



8. 非互換

パラメーターのデフォルト値

– パラメーターのデフォルト値変更

パラメーター名	PostgreSQL 9.6	PostgreSQL 10
hot_standby	off	on
log_directory	pg_log	log
log_line_prefix	"	%m [%p]
max_parallel_workers_per_gather	0	2
max_replication_slots	0	10
max_wal_senders	0	10
password_encryption	on	md5
wal_level	minimal	replica



8. 非互換

pg_hba.conf

– Replication用設定はローカルホストはtrustがデフォルト

```
$ cat data/pg_hba.conf
# TYPE      DATABASE      USER      ADDRESS      METHOD

# "local" is for Unix domain socket connections only
local      all              all              trust
# IPv4 local connections:
host       all              all          127.0.0.1/32  trust
# IPv6 local connections:
host       all              all          ::1/128      trust
# Allow replication connections from localhost, by a user with the
# replication privilege.
local      replication      all              trust
host       replication      all          127.0.0.1/32  trust
host       replication      all          ::1/128      trust
```

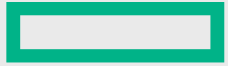


まとめ



まとめ

- PostgreSQL 10には、魅力的な新機能が数多く採用されました
 - Logical Replication
 - パーティショニング
 - パラレル・クエリーの拡張
 - その他(同期レプリケーションの拡張、HASHインデックの拡張等)
- 参考URL
 - オンライン・マニュアル
<https://www.postgresql.org/docs/10/static/index.html>
 - Slack
<https://postgresql-hackers-jp.herokuapp.com/>
 - ぬこ@横浜さんのブログ
https://qiita.com/nuko_yokohama
 - Michael Paquierさんのブログ
<http://paquier.xyz/>
 - PostgreSQL 10 Beta1 新機能検証結果(PostgreSQL 虎の巻 その7)
<https://www.slideshare.net/noriyoshishinoda/postgresql-10-beta1-new-features-japanese>



Hewlett Packard
Enterprise

Thank you

noriyoshi.shinoda@hpe.com