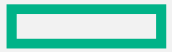


Hewlett Packard
Enterprise

PostgreSQL 進化と将来

Noriyoshi Shinoda

January 12, 2020

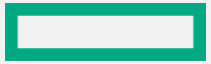
SPEAKER
篠田典良（しのだのりよし）

- 所属
  - 日本ヒューレット・パッカード株式会社
- 現在の業務
  - PostgreSQL をはじめ、Oracle Database, Microsoft SQL Server, Vertica 等 RDBMS 全般に関するシステムの設計、移行、チューニング、コンサルティング
  - Oracle ACE（2009 年 4 月～）
  - Oracle Database 関連書籍15冊の執筆
  - オープンソース製品に関する調査、検証
- 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 概要
- パラレル・クエリー
- パーティショニング
- ロジカル・レプリケーション
- その他
- 非互換

PostgreSQL 概要



PostgreSQL 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
  - 特定の企業に所属していない
- 最新バージョン
  - PostgreSQL 12 (12.1)
  - 原則として 1 年毎に新バージョンが公開

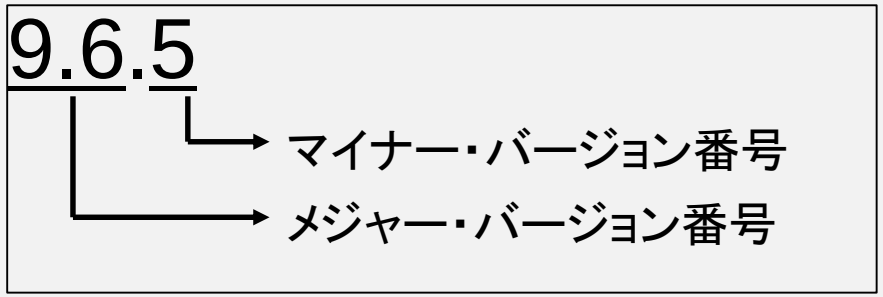
PostgreSQL 概要
PostgreSQL の歴史

- 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～
- 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
- 2017年10月 PostgreSQL 10
- 2018年10月 PostgreSQL 11
- 2019年10月 PostgreSQL 12

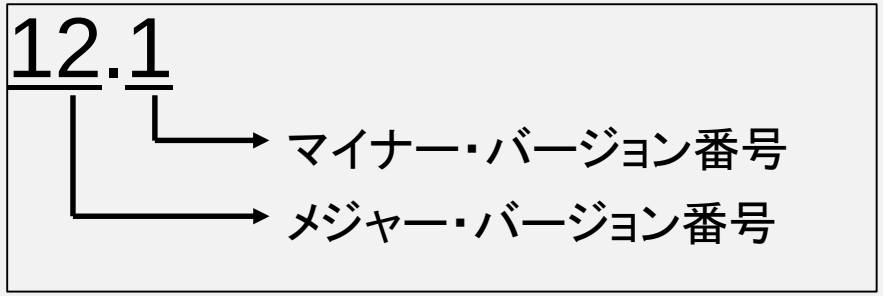
今日お話しする範囲

PostgreSQL 概要
バージョン番号

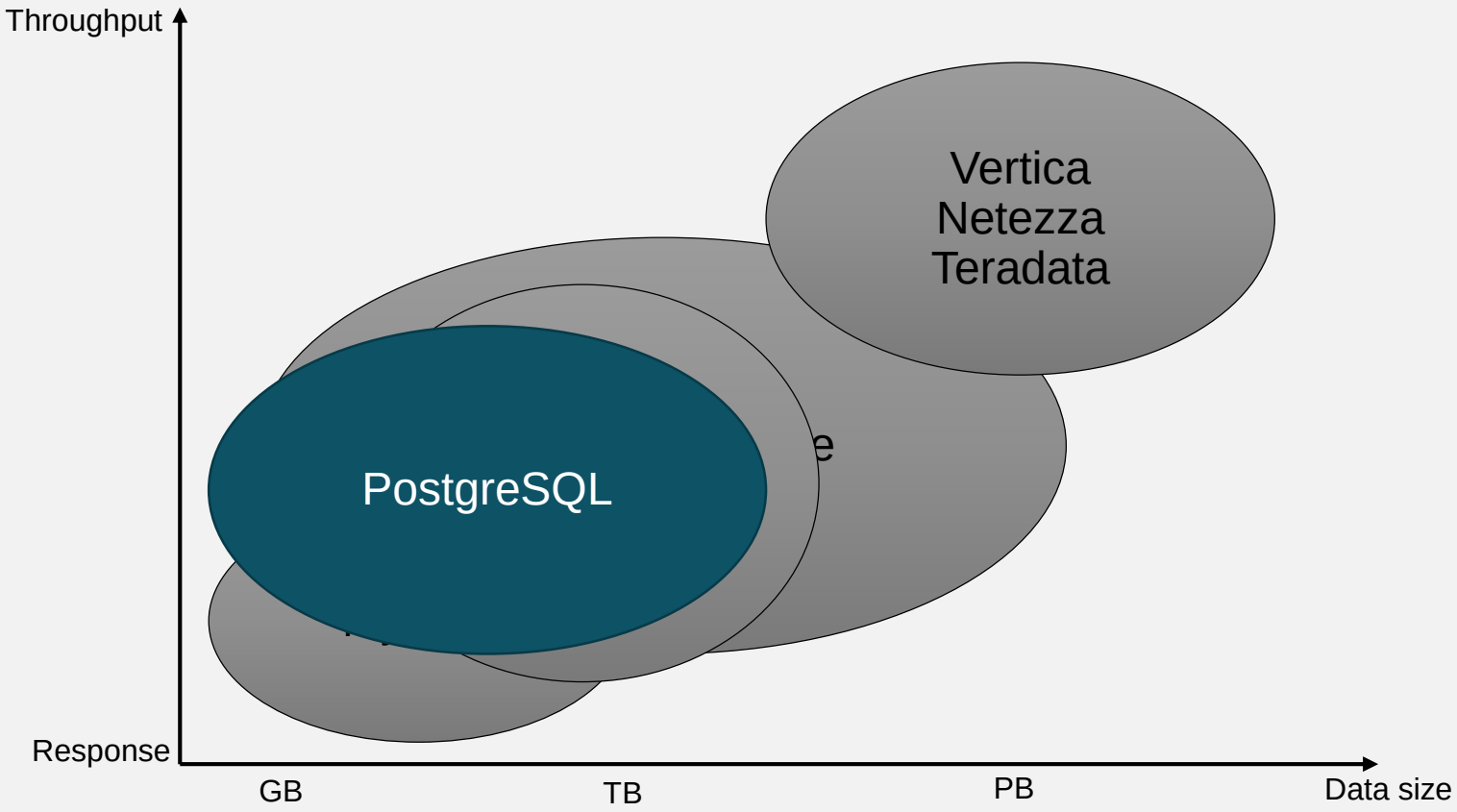
- PostgreSQL 9.6 まで
  - 2つの数字がメジャー、最後の数字がマイナー



- PostgreSQL 10 以降
  - 最初の数がメジャー、最後の数がマイナー
  - マイナー・バージョンはバグ・フィックスおよび脆弱性対策バージョン



PostgreSQL 概要
PostgreSQL の適用範囲



PostgreSQL 概要
PostgreSQL から派生／拡張した製品

- EDB Postgres Enterprise Edition (EnterpriseDB)
  - Oracle Database 互換機能
- Amazon Aurora PostgreSQL (AWS)
  - ストレージ・エンジンの機能拡張＋スケールアウト
- Azure Database for PostgreSQL – Hyperscale (Citus)
  - エクステンションによるスケールアウト
- Amazon Redshift (AWS)
  - DWH 用機能拡張
- Vertica (Micro Focus Enterprise)
  - DWH 用機能拡張
- Pivotal Greenplum (Pivotal)
  - DWH 用機能拡張

PostgreSQL 概要
PostgreSQL 進化の例

- 大規模データの並列処理
  - パラレル・クエリー
- データの分割による I/O 削減
  - パーティショニング
  - ロジカル・レプリケーション
- プロセッサの効率的な使用
  - LLVM 統合
- 信頼性の向上
  - pg_checksums ユーティリティ

パラレル・クエリー

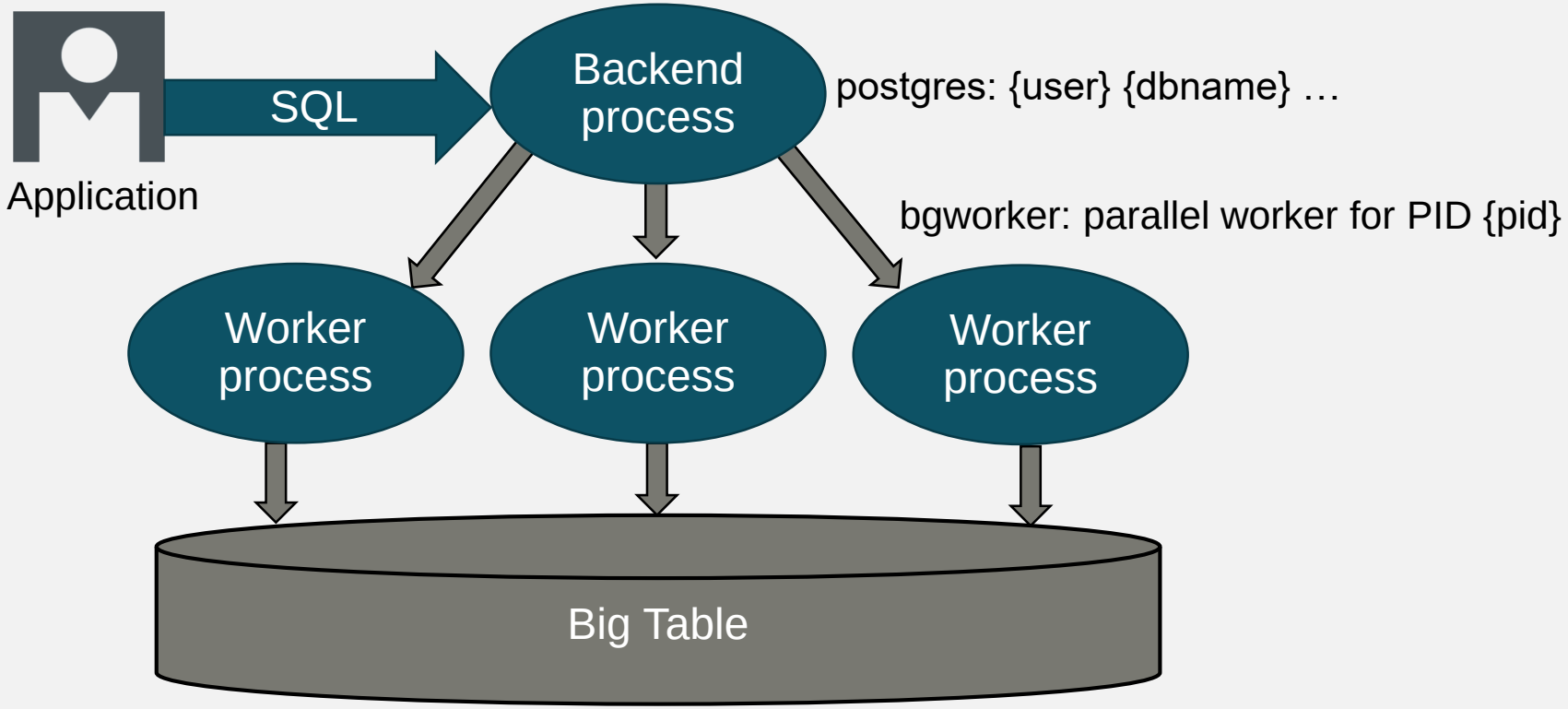
パラレル・クエリー
概要

- 概要説明
  - パラレル・クエリーは単一の SQL 文を複数のバックエンド・プロセスで並列に処理を行う機能
  - パラレル／シリアルの選択はコスト量により自動的に決定

- アーキテクチャ
  - パラレル処理は Background Worker (9.3～) を使う
  - プロセス間通信には Dynamic Shared Memory (9.4～) を使う
  - パラレル処理 API (9.5～) を使う

パラレル・クエリー
処理概要図

– SQL文を複数のプロセスで平行に処理を行う



パラレル・クエリー
実行計画

- 実行計画

```
postgres=> EXPLAIN ANALYZE SELECT COUNT(*) FROM data1;
                                                              QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------
 Finalize Aggregate  (cost=11614.55..11614.56 rows=1 width=8) (actual time=1106.746..1106.747 rows=1 loops=1)
   ->  Gather  (cost=11614.33..11614.54 rows=2 width=8) (actual time=1105.972..1106.766 rows=3 loops=1)
         Workers Planned: 2
         Workers Launched: 2
         ->  Partial Aggregate  (cost=10614.33..10614.34 rows=1 width=8) (actual time=1087.334..1087.335 rows=1 loops=3)
               ->  Parallel Seq Scan on data1  (cost=0.00..9572.67 rows=416667 width=0) (actual time=0.018..591.216 rows=333333 loops=3)
 Planning Time: 0.030 ms
 Execution Time: 1106.803 ms
(8 rows)
```

パラレル・クエリー
関連するパラメーター

– 関連するパラメーター（PostgreSQL 12）

パラメーター名	説明	デフォルト値　備考
max_parallel_workers	パラレルワーカーの最大数	8
max_parallel_maintenance_workers	ユーティリティの最大ワーカー数	2
max_parallel_workers_per_gather	SQL 内のワーカー最大数	2
min_parallel_table_scan_size	最小テーブルサイズ	8MB
min_parallel_index_scan_size	最大インデックスサイズ	512kB
enable_parallel_append	並列 Append 実行可否	on
enable_parallel_hash	並列 Hash 実行可否	on
parallel_setup_cost	並列化初期コスト	1000
parallel_tuple_cost	並列化タプルコスト	0.1
parallel_leader_participation	リーダー・プロセスがタプルを処理	on

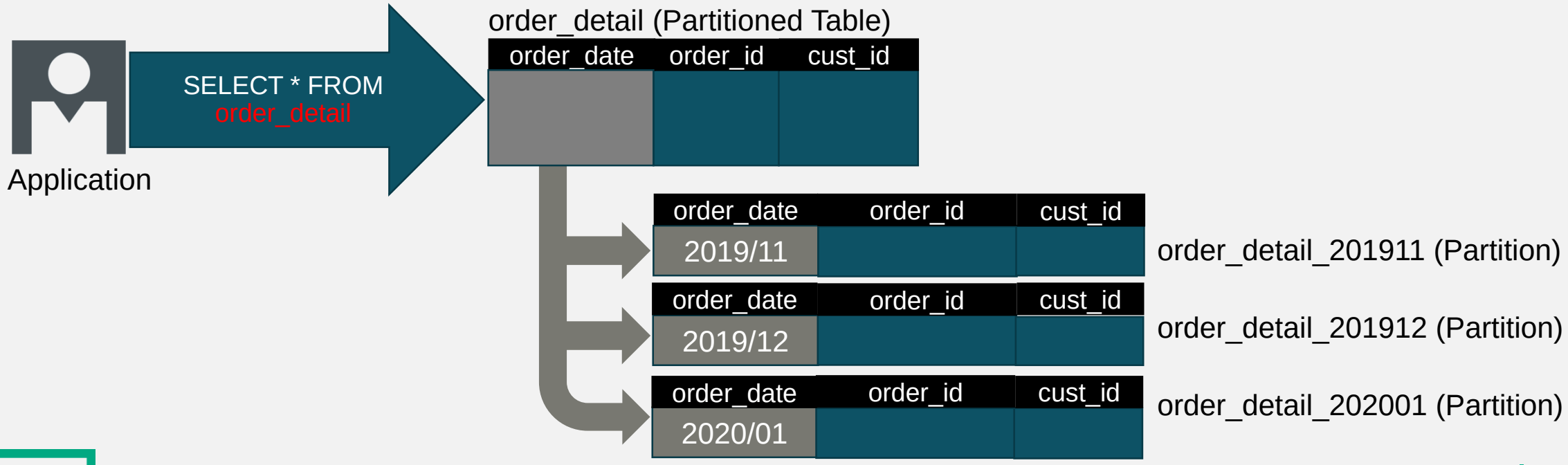
パラレル・クエリー
バージョン間の差異

構文／環境	9.6	10	11	12	備考
全件検索（Seq Scan）と集約（Aggregate)	●				
インデックス検索（Index Scan）		●			
結合（Nest Loop / Merge Join）		●			
ビットマップ・スキャン（Bitmap Heap Scan）		●			
PREPARE / EXECUTE 文		●			
サブクエリー（Sub Plan）		●			
COPY 文		●			
結合（Hash Join）			●		
UNION 文（Append）			●		
CREATE 文（TABLE AS SELECT / MATERIALIZED VIEW / INDEX）			●		
SELECT INTO 文			●		
SERIALIZABLE トランザクション分離レベル				●	

パーティショニング

パーティショニング
概要

- 概要説明
  - 大規模なテーブルを物理的に分割する機能
  - 一般的には列値を使って自動的に分割先を決定
  - パーティションもテーブルとしてアクセス可能



パーティショニング
分割方法

- LIST Partition
  - 特定の値でパーティション化
  - 列値に一致するパーティションが選択される
- RANGE Partition
  - 値の範囲でパーティション化
  - 「下限値 <= 列値 < 上限値」によりパーティションが選択される
- HASH Partition
  - 値のハッシュ値でパーティション化
  - 分割数を指定する
  - PostgreSQL 11 から利用可能

パーティショニング
パーティション・テーブル作成例

- LIST パーティションの作成例

```
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.)
```

パーティショニング
パーティションの追加／削除

- パーティションのアタッチ／デタッチ例

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

- パーティションのアタッチ時の動作と制約
  - ATTACH PARTITION 句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。
  - ATTACH PARTITION 句実行時に格納済のデータは FOR VALUES 句に合致しているかチェックされる。

パーティショニング
実行計画

```
postgres=> CREATE TABLE measurement (city_id int not null, logdate date not null,unitsales int)
    PARTITION BY RANGE (logdate);
CREATE TABLE
postgres=> CREATE TABLE measurement_y2019m02 PARTITION OF measurement
    FOR VALUES FROM ('2019-02-01') TO ('2019-03-01');
CREATE TABLE
postgres=> CREATE TABLE measurement_y2020m12 PARTITION OF measurement
    FOR VALUES FROM ('2020-12-01') TO ('2021-01-01');
CREATE TABLE
...
postgres=> EXPLAIN SELECT * FROM measurement WHERE logdate = '2020-12-02';
                              QUERY PLAN
-----------------------------------------------------------------------
 Seq Scan on measurement_y2020m12  (cost=0.00..33.12 rows=9 width=16)
   Filter: (logdate = '2020-12-02'::date)
(2 rows)
```

パーティショニング
関連するパラメーター

– 関連するパラメーター（PostgreSQL 12）

パラメーター名	説明	デフォルト値　備考
enable_partition_pruning	パーティション・プルーニング実施	on
enable_partitionwise_aggregate	パーティションワイズ集約の実施	off
enable_partitionwise_join	パーティションワイズ結合の実施	off

パーティショニング
パーティション機能の拡充

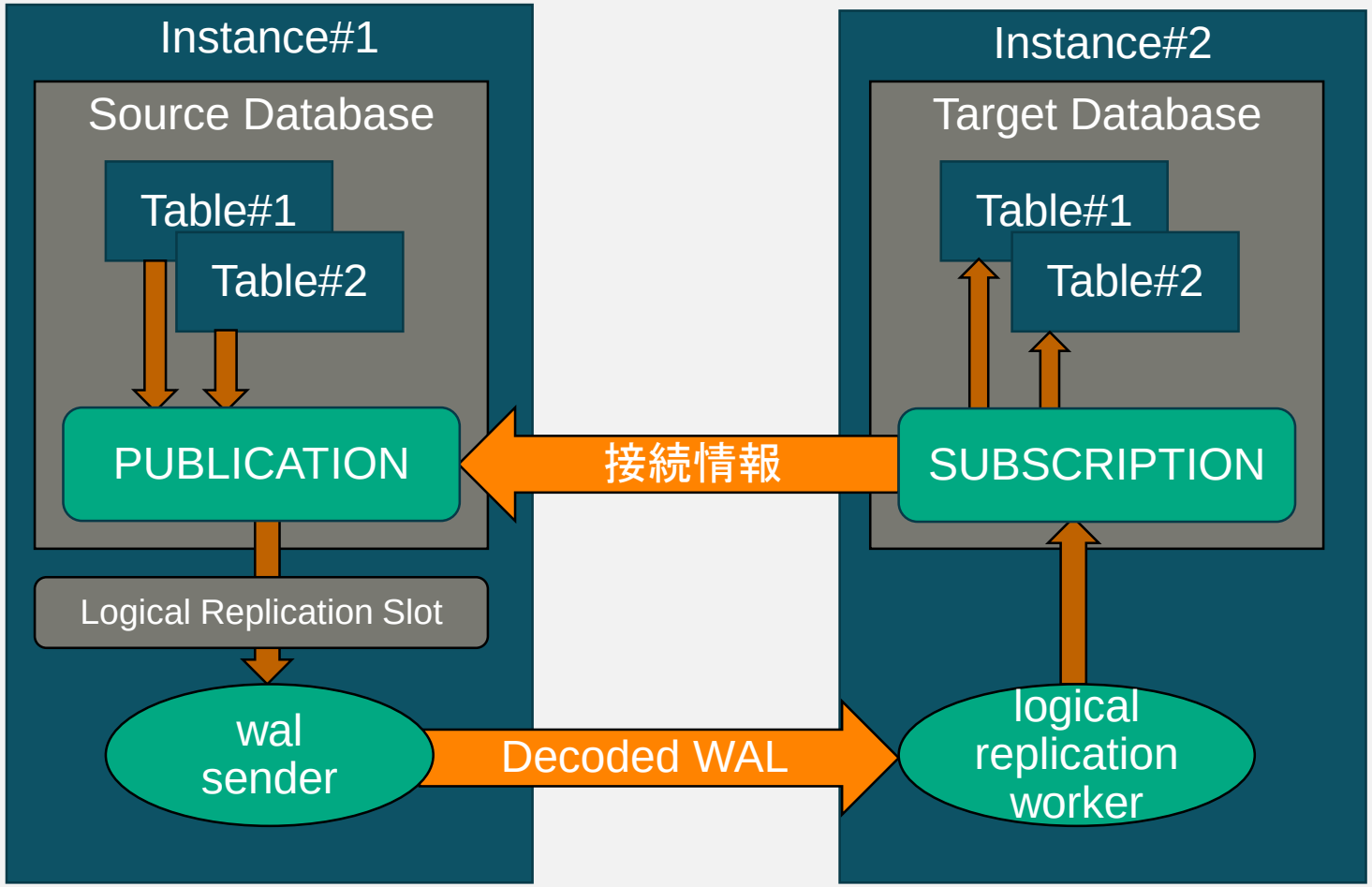
構文／環境	10	11	12	備考
範囲によるパーティション（RANGE PARTITION）	●			
値によるパーティション（LIST PARTITION）	●			
ハッシュ値によるパーティション（HASH PARTITION）		●		
その他の値が格納されるパーティション（DEFAULT PARTITION）		●		
パーティションを移動する UPDATE 文の実行		●		
親パーティション・テーブルに対するインデックス作成と伝播		●		
親パーティションに対する一意制約の作成		●		
パーティション・ワイズ結合		●		
INSERT ON CONFLICT 文の対応		●		
計算値によるパーティション			●	
外部キーとしてパーティション・テーブルの参照			●	

ロジカル・レプリケーション

ロジカル・レプリケーション
部分レプリケーション機能

- ロジカル・レプリケーションとは？
  - PostgreSQL 10 以降で利用可能
  - 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)
- ストリーミング・レプリケーション (Physical Replication) とは？
  - PostgreSQL 9.0 以降で利用可能
  - 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）

PostgreSQL 10
ロジカル・レプリケーション



ロジカル・レプリケーション
オブジェクト

- PUBLICATION オブジェクト
  - データ提供側データベースに作成
  - 一般ユーザー権限で作成可能
  - レプリケーション対象テーブルを決定
  - CREATE PUBLICATION 文で作成

- SUBSCRIPTION オブジェクト
  - データ受信側データベースに作成
  - SUPERUSER 権限が必要
  - CREATE SUBSCRIPTION 文で作成
  - 作成時に接続先インスタンスの接続情報と PUBLICATION 名を指定

ロジカル・レプリケーション
作成例(データ提供元)

- レプリケーション対象テーブルの作成

```
pubdb=> CREATE TABLE data1 (c1 INT PRIMARY KEY, c2 VARCHAR(5));
CREATE TABLE
```

- レプリケーション対象テーブルの参照を接続ユーザーに許可

```
pubdb=> GRANT SELECT ON data1 TO repusr1;
GRANT
```

- PUBLICATION オブジェクトの作成

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

ロジカル・レプリケーション
作成例（データ受信先）

－レプリケーション対象テーブルの作成

```
subdb=> CREATE TABLE data1 (c1 INT PRIMARY KEY, c2 VARCHAR(5));
CREATE TABLE
```

－SUBSCRIPTIONオブジェクトの作成（SUPERUSER）

```
subdb=# CREATE SUBSCRIPTION sub1 CONNECTION
  'host=pubhost1 dbname=pubdb user=repusr1 password=******' PUBLICATION pub1;
CREATE SUBSCRIPTION
```

ロジカル・レプリケーション
制約

- 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）

ロジカル・レプリケーション
関連するパラメーター

– 関連するパラメーター（PostgreSQL 12）

パラメーター名	説明	デフォルト値　備考
max_logical_replication_workers	レプリケーション・ワーカーの最大数	4
max_sync_workers_per_subscription	サブスクリプション単位の同期ワーカー数	2
max_worker_processes	ワーカー・プロセスの最大値	8

ロジカル・レプリケーション
バージョン間の違い

構文／環境	10	11	備考
PUBLISHER / SUBSCRIBER によるレプリケーション	●		
テーブル単位の伝播	●		
全テーブルの伝播	●		
文字コード変換	●		
TRUNCATE 文の伝播		●	

- 制約
  - DDL は伝播不可
  - UNLOGGED TABLE ／ TEMPORARY TABLE は伝播不可
  - SEQUENCE ／ MATERIALIZED VIEW ／ INDEX は伝播不可

その他

その他
JSON 対応

- JSON
  - データ型として json 型（9.2～）と jsonb 型（9.4～）を使用

構文／環境	10	11	12	備考
to_tsvector / to_headline 関数による型変換	●			
配列からの要素削除	●			
jsonb_plpython / jsonb_plperl モジュール		●		
json[b]_to_tsvector 関数		●		
bool 型／数値型へのキャスト		●		
jsonb_path_exits 関数			●	
jsonb_path_match 関数			●	
jsonb_path_query 関数			●	
jsonb_path_query_arrary 関数			●	
jsonb_path_query_first 関数			●	

その他
JIT機能(PostgreSQL 11)

- SQL 文の実行(Executer)をネイティブ・コンパイルしたコードで実行
  - LLVM を統合(https://llvm.org/)
  - プロセッサ・ネックとなる長時間実行される SQL 文のパフォーマンスを向上
  - 一定コスト(jit_above_cost)以上の SQL 文に対して実行される
  - JIT 機能を使う SQL 文の実行計画

```
postgres=> EXPLAIN SELECT COUNT(*) FROM data1;
                                  QUERY PLAN
-------------------------------------------------------------------------------
 Aggregate  (cost=17906.00..17906.01 rows=1 width=8)
   ->  Seq Scan on data1  (cost=0.00..15406.00 rows=1000000 width=0)
 JIT:
   Functions: 2
   Options: Inlining false, Optimization false, Expressions true, Deforming true
(5 rows)
```

その他

PROCEDURE オブジェクト（PostgreSQL 11）

- 戻り値が無い FUNCTION オブジェクトと同じ
- CREATE PROCEDURE 文で作成
- EXCEPTION ブロックが無い場合、トランザクション制御（COMMIT / ROLLBACK）を実行可能

```
postgres=> CREATE PROCEDURE proc01() AS $$
        BEGIN
                RAISE NOTICE 'Procedure Demo';
        END; $$
        LANGUAGE plpgsql;
CREATE PROCEDURE
postgres=> CALL proc01();
```

非互換

非互換
設定ファイルの統合（PostgreSQL 9.6⇒10）

- データベース・クラスタ内のディレクトリ名変更
  - ログ・ファイル用ディレクトリはパラメーター 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

非互換
設定ファイルの統合（PostgreSQL 9.6⇒10）

– パラメーターのデフォルト値変更

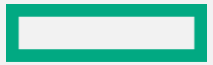
パラメーター名	PostgreSQL 9.6	PostgreSQL 10
hot_standby	off	on
log_directory	pg_log	log
log_line_prefix	''	%m [%p]
max_parallel_workers_per_gather	0	2
max_replication_slots	0	10
max_wal_senders	0	10
password_encryption	on	md5
wal_level	minimal	replica

非互換
設定ファイルの統合（PostgreSQL 11⇒12）

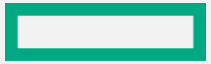
- postgresql.conf と recovery.conf ファイルの統合
  - 原則として recovery.conf ファイル内の同一名称のパラメーターを postgresql.conf に設定
  - 以下は例外パラメーター

説明	recovery.conf	postgresql.conf	備考
スタンバイ・モード	standby_mode	廃止	hot_standby で代替
昇格指示ファイル名	trigger_file	promote_trigger_file	

  - リカバリ（ストリーミング・レプリケーション含む）場合は recovery.signal ファイルを作成
  - スタンバイ・インスタンスの自動昇格を行うクラスタウェアや、リカバリ手順の変更が必要



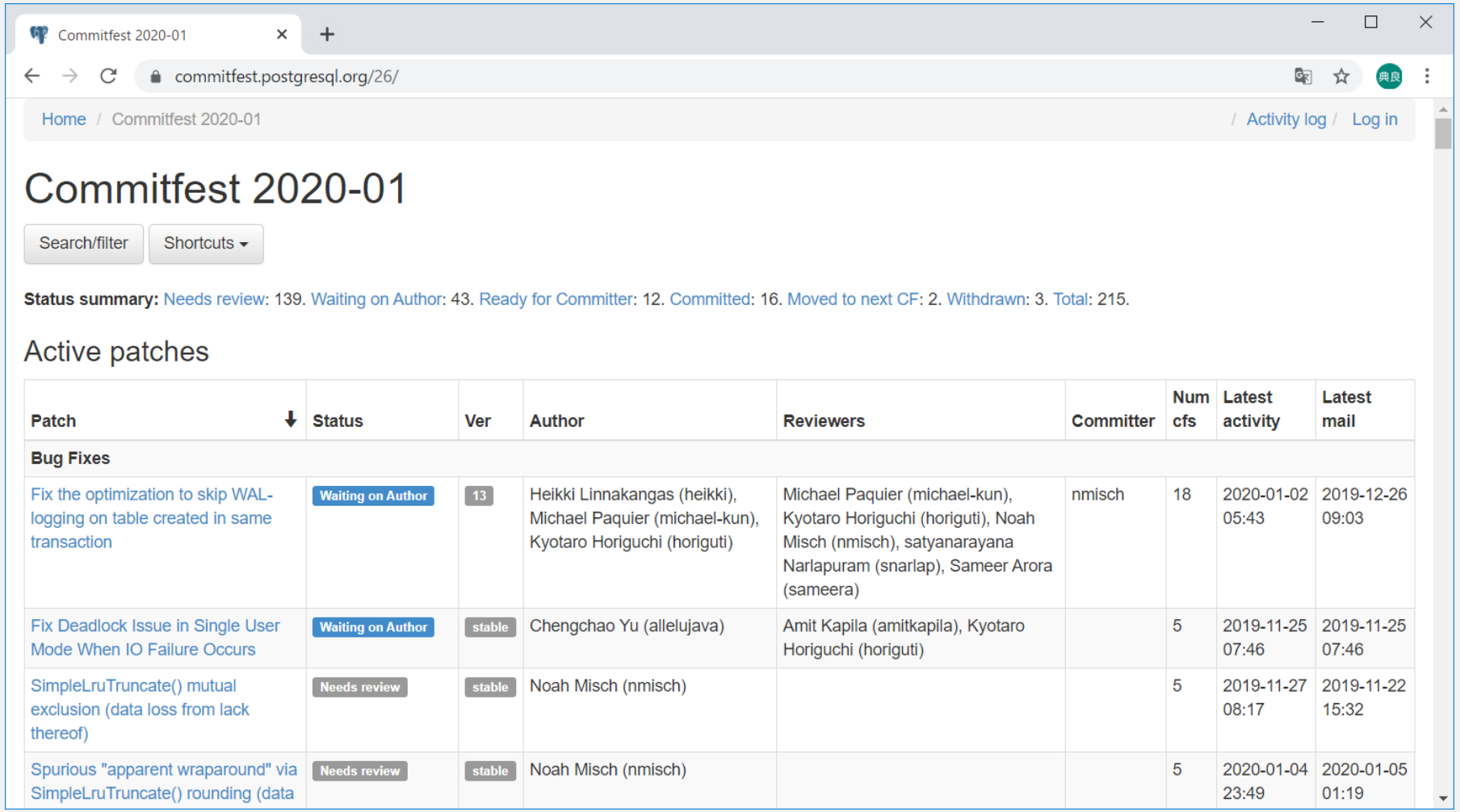
PostgreSQL の将来



PostgreSQL の将来
新機能の開発

- メーリングリストによる開発
  - pgsql-hackers@postgresql.org にパッチを添付して送付
- Commitfest
  - https://commitfest.postgresql.org/
  - メーリングリストに添付したパッチを登録する
  - 3か月単位でステータスが管理される
  - レビューアが明示される
  - メーリングリストによる議論が一覧化される
- GIT
  - git://git.postgresql.org/git/postgresql.git

PostgreSQL の将来
Commitfest



PostgreSQL の将来
開発中の機能

- Pluggable Storage Engine (PostgreSQL 12)
  - 複数のストレージ・エンジンを選択可能に
  - テーブル単位にストレージエンジンを指定
  - zHeap (EnterpriseDB)
    - 追記型アーキテクチャからの脱却
    - VACUUM 不要
    - テーブルの肥大化を避けられる
    - https://github.com/EnterpriseDB/zheap
- Transaction ID の 64 bit 化
  - 現状では 32 bit のため、周回対策（FREEZE）が必要
  - 64 bit 化により FREEZE 不要に
  - PostgreSQL 12 では一部の API の提供に留まる

PostgreSQL の将来
開発中の機能

- ロジカル・レプリケーション
  - テーブルの特定のレコードのみをレプリケーション
  - パーティション・テーブルのレプリケーション
  - ストリーミング対応（現状ではトランザクション単位）
- パラレル・クエリー
  - GROUPING SETS 句のパラレル化
  - パラレル VACUUM

参考資料
参考になる情報

- Web サイト
  - 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/
  - PostgreSQL マニュアル日本語化プロジェクト
    https://github.com/pgsql-jp/jpug-doc
- Twitter
  @snaga
  @nuko_yokohama
  @masahiko_sawada
  @fujii_masao
  @kkaigai
  @michaelpq
  @tatsuo_ishii
  @soudai1025

THANK YOU

Mail: noriyoshi.shinoda@hpe.com
Twitter: @nori_shinoda