

# Logical Replication Internals

Noriyoshi Shinoda

December 6, 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
- Oracle ACE
- 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

- Logical Replicationとは？
- 試してみよう
- アーキテクチャ
- 制約
- トラブルシューティング





# Logical Replicationとは？

# Logical Replicationとは？

## Logical Replicationとは？

### – Is

- PostgreSQL 10の新機能
- テーブル単位のレプリケーション
- トランザクション単位のレプリケーション
- レプリケーション先のテーブルも更新可能
- SQL文の結果が同一になることを保証(=Logical)
- Publish / Subscribeモデルを使用
- ≒ Slony-I

### – Is Not

- SQLの再実行
- 物理的なページ・フォーマットの一致



---

# Logical Replicationとは？

## レプリケーション条件

- レプリケーション可能なテーブルの条件
  - 同一のスキーマ名
  - 同一のテーブル名
  - 同一の列名
  - 同一の列データ型
    - 暗黙の型変換できれば異なるデータ型利用可能





試してみよう

# 試してみよう

## Publisherインスタンス上の操作

- pubdbデータベースのdata1テーブルをsubdbデータベースへレプリケーション
- REPLICATION属性／LOGIN属性を持つロール作成
  - Subscriberインスタンスから接続されるロール

```
pubdb=# CREATE ROLE repusr1 PASSWORD 'Passw0rd' LOGIN REPLICATION ;  
CREATE ROLE
```

- テーブル所有者(pubusr1)にデータベースのCREATE権限を付与

```
pubdb=# GRANT CREATE ON DATABASE pubdb TO pubusr1 ;  
GRANT
```

- pg\_hba.conf ファイルの修正

#	TYPE	DATABASE	USER	ADDRESS	METHOD
host		pubdb	repusr1	192. 168. 1. 100/32	md5

- DATABASE=replication 項目不要



# 試してみよう

## Publisherインスタンス上の操作

- postgresql.conf ファイルの修正

```
wal_level = logical
```

# 試してみよう

## Publisherデータベース上の操作

### – 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
```

# 試してみよう

## Subscriberデータベース上の操作

### – 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=Passw0rd'  
      PUBLICATION pub1 ;  
CREATE SUBSCRIPTION
```



# 試してみよう

## 確認

### – Publisherインスタンス

```
pubdb=# SELECT application_name, state, sync_state FROM  
       pg_stat_replication ;
```

application_name	state	sync_state
sub1	streaming	async

(1 row)

### – Subscriberインスタンス

```
subdb=> SELECT subname, received_lsn FROM pg_stat_subscription ;
```

subname	received_lsn
sub1	0/1C8FF738

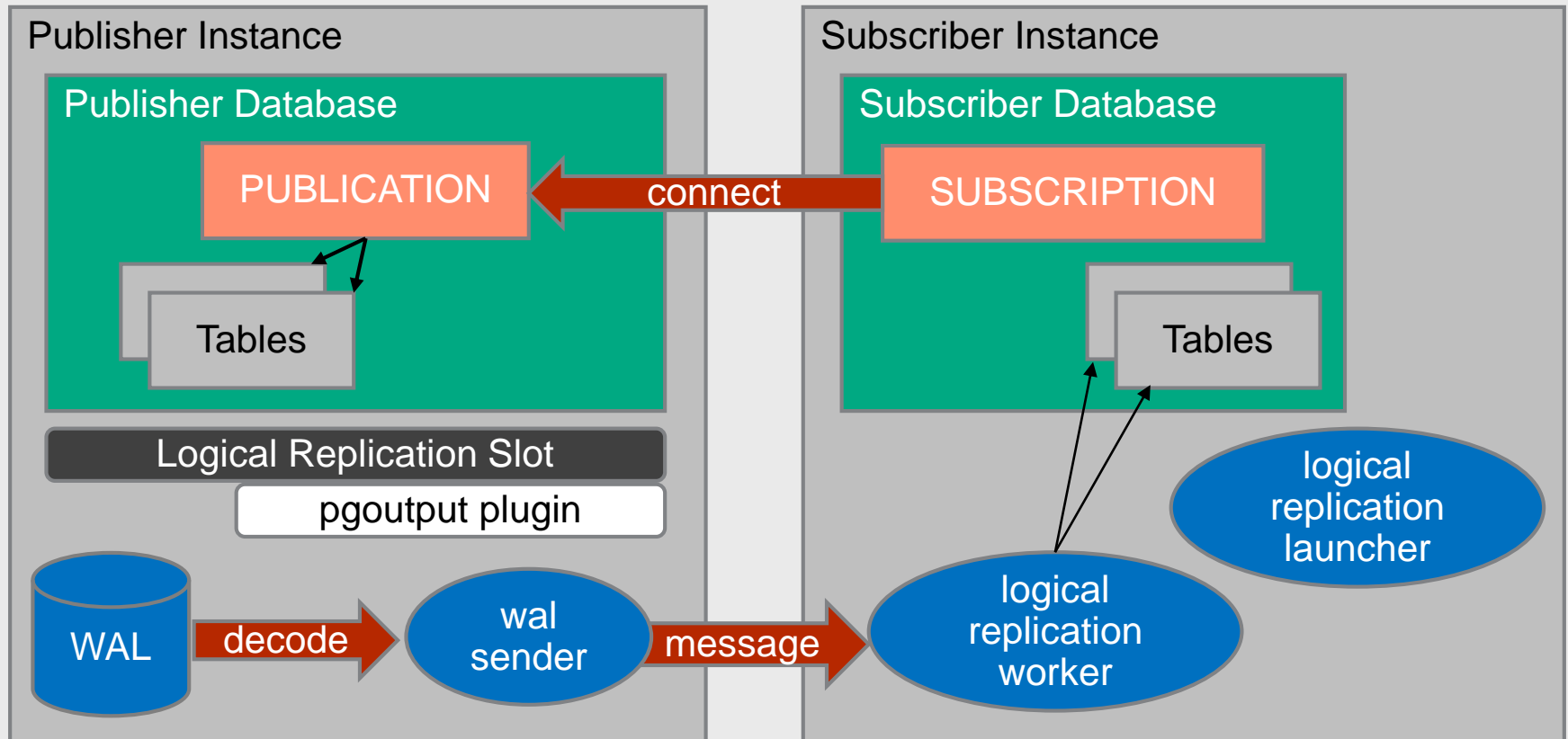




# アーキテクチャー

# アーキテクチャ

## Logical Replicationの構成要素



# アーキテクチャー

## プロセス

- wal sender process {user} {client ip} {state}
  - PUBLICATIONインスタンスで稼働
  - WALデコード・メッセージ送信プロセス
  - SUBSCRIPTIONからの接続単位に起動
- bgworker: logical replication launcher
  - PUBLICATION / SUBSCRIPTIONインスタンスで稼働
  - logical replication workerプロセスを起動
- bgworker: logical replication worker for subscription {oid}
  - SUBSCRIPTIONインスタンスで稼働
  - wal sender processプロセスに接続
  - WALデコード・メッセージを受信し、テーブルを更新
  - SUBSCRIPTION単位に起動



# アーキテクチャー

## PUBLICATION作成時の動作

- **pg\_publication**カタログに情報格納
  - PUBLICATION名
  - 伝播するDML (INSERT / UPDATE / DELETE)
  - 全テーブルを対象とするか (FOR ALL TABLES)
- **pg\_publication\_rel**カタログに情報格納
  - レプリケーション対象のテーブルOIDとPUBLICATIONのOID
  - FOR ALL TABLES指定の場合はこのカタログには格納されない
- **pg\_publication\_tables**カタログはテーブル名を参照できる





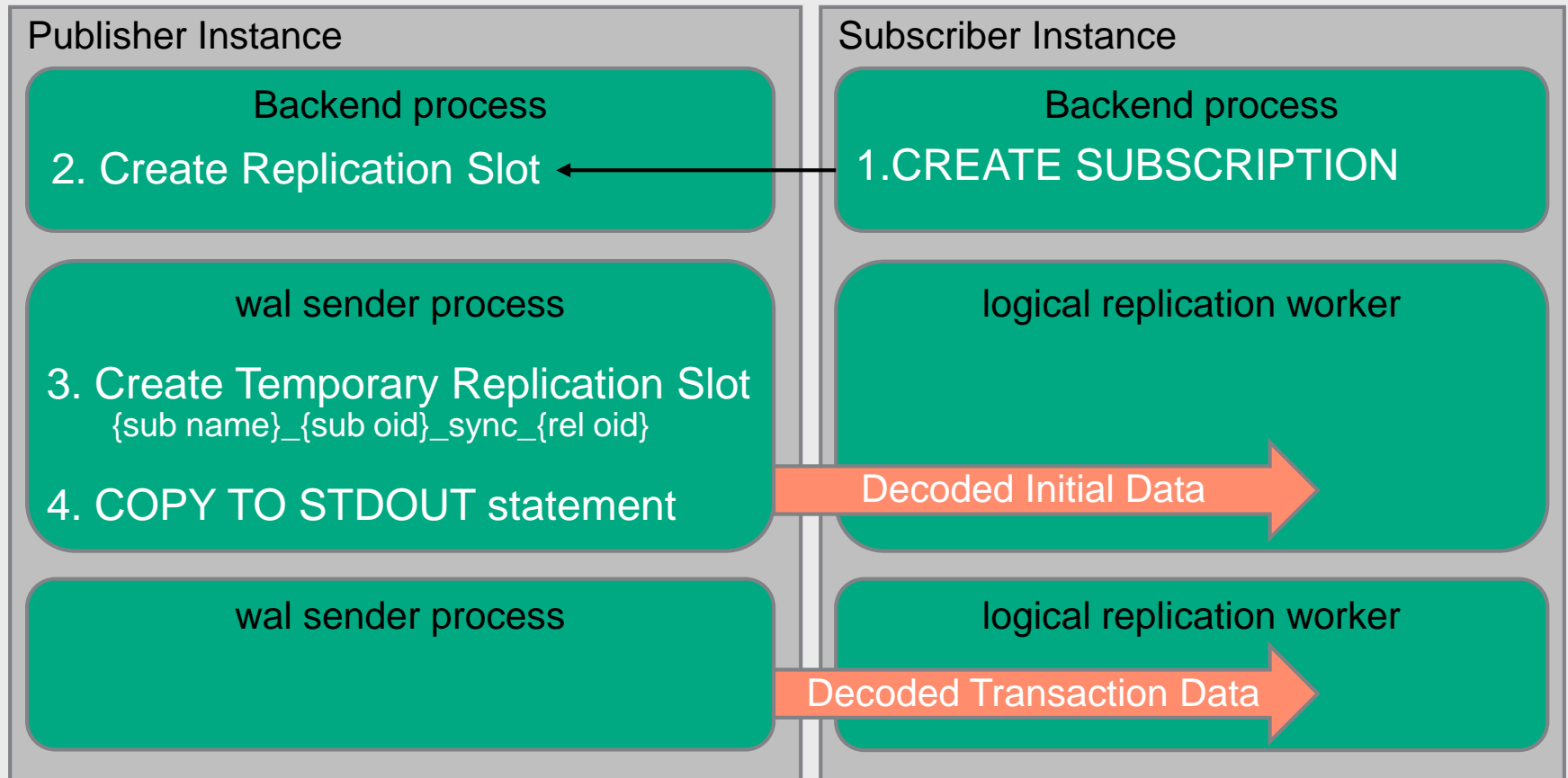
# アーキテクチャー

## SUBSCRIPTION作成時の動作

- **pg\_subscription**カタログに情報格納
  - 接続先インスタンスの情報
  - Replication Slot名
  - 接続先PUBLICATION名
  - 同期／非同期レプリケーションの情報
- PUBLICATION側インスタンスに接続
  - 接続ユーザーのREPLICATION権限のチェック
  - Logical Replication Slot作成(名前はデフォルトはSUBSCRIPTION名)
  - **PUBLICATIONが存在するかはチェックされない**
  - **pg\_subscription\_rel**カタログにレプリケーション対象テーブルを登録
- 初期データのロード
  - 非同期に実行される
  - SUBSCRIPTION側の既存データは削除されない

# アーキテクチャ

## 初期データの同期



# アーキテクチャー

## Replication Slot

### – Logical Replication Slot

- SUBSCRIPTIONと1対1で構成
- 送信済WALの管理
- プラグインの提供
- SUBSCRIPTION作成時に以下のSQL文を自動実行

```
pg_create_logical_replication_slot ( name, 'pgoutput' )
```

### – レプリケーション・スロット名

- デフォルトではSUBSCRIPTION名
- CREATE SUBSCRIPTION文のWITH (**slot\_name=name**) で変更可能



# アーキテクチャー

## pgoutputプラグインとメッセージ

### – pgoutputプラグイン

- Logical Replication用の標準プラグイン
- wal senderプロセスから利用
- WALからLogical Replicationメッセージを作成
  - 文字コード変換
  - バイナリーデータのテキスト変換
- pg\_logical\_slot\_get\_changes関数のテキスト出力に対応していない？

### – メッセージ

- テキスト・フォーマット
- プロトコル
  - <https://www.postgresql.org/docs/10/static/protocol-logicalrep-message-formats.html>
- UPDATE/DELET文の実行
  - 更新対象列を特定するデータと更新後のデータを送信



# アーキテクチャー

## Replica Identity

– UPDATE文 / DELETE文が伝播する条件

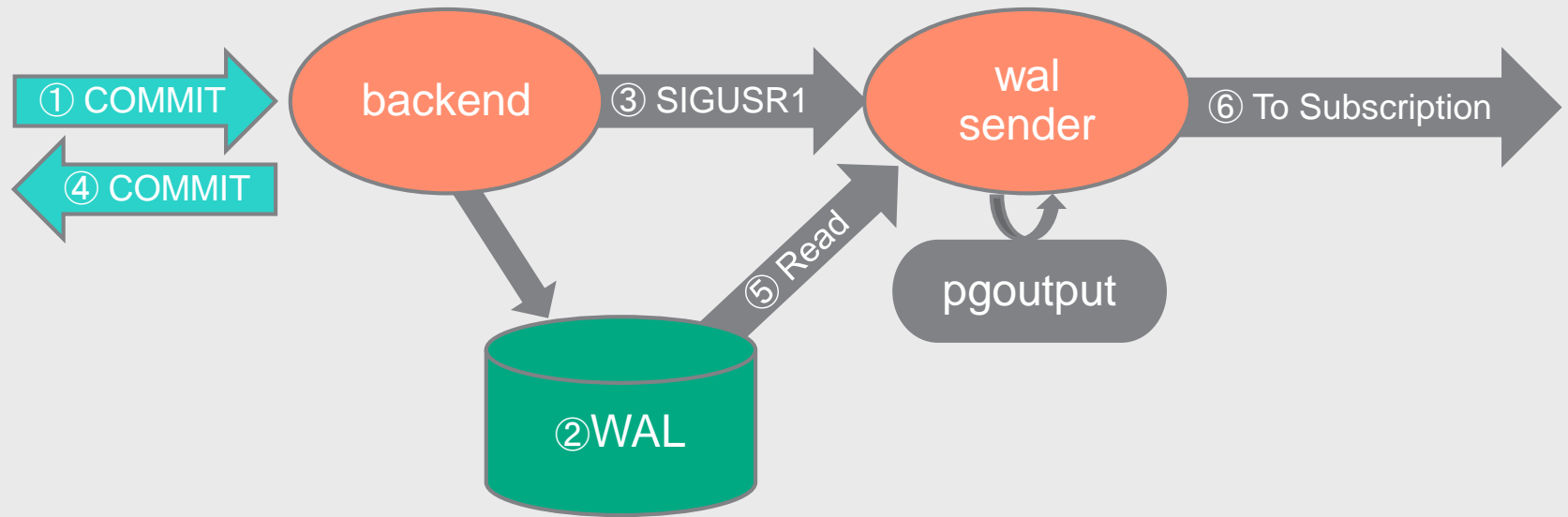
テーブル構成	PUBLICATION	SUBSCRIPTION
PRIMARY KEY	O	O
REPLICA IDENTITY FULL	O	×
REPLICA IDENTITY USING INDEX + UNIQUE index + NOT NULL	O	O

– 主キーまたはREPLICA IDENTITY設定が無いテーブルに対するUPDATE / DELETEはSQL実行エラー

```
pubdb=> UPDATE data1 SET c2='update' WHERE c1=100 ;  
ERROR:  cannot update table "data1" because it does not have  
replica identity and publishes updates  
HINT:  To enable updating the table, set REPLICA IDENTITY using  
ALTER TABLE.
```

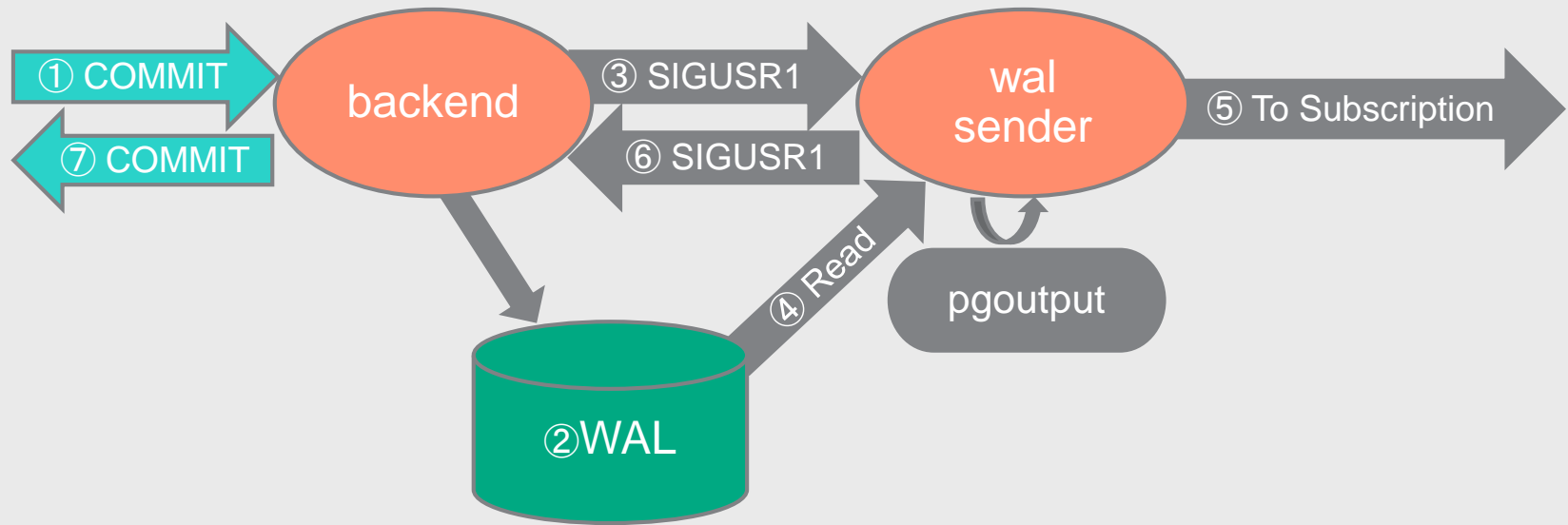
# アーキテクチャ

## DML実行時の動作(非同期レプリケーション)



# アーキテクチャ

## DML実行時の動作(同期レプリケーション)



# アーキテクチャー

## ストレージ

### – 論理ログ

- wal senderプロセスがwal writerプロセスからSIGUSR1シグナルを受信すると作成
- WALファイルの読み込みを行いデータを作成
- ファイル名

```
$ {PGDATA} /pg_logical/snapshots/ {LSN上位} - {LSN下位} . snap
```

### – SUBSCRIPTION遅延時の論理ログ

- SUBSCRIPTIONインスタンスが停止し、wal senderプロセスが再起動された場合に作成
- 転送が完了すると削除される
- ファイル名

```
$ {PGDATA} /pg_replslot/ {SLOT_NAME} /xid- {XID} -lsn- {LSN上位} - {LSN下位} . snap
```







# 制約



---

# 制約

## レプリケーションできないSQL

- TRUNCATE文
- ALTER TABLE文
- CREATE TABLE文
  - CREATE REPLICATION FOR ALL TABLES文実行時



# 制約

## レプリケーションできないオブジェクト

- PUBLICATIONに追加できるのはテーブルのみ
  - `pg_class.relkind='r'`
- レプリケーション対象外
  - MATERIALIZED VIEW
  - INDEX
  - SEQUENCE
  - FOREIGN TABLE
  - UNLOGGED TABLE
  - INHERIT TABLE = OK (ONLY句)
  - Partition parent table
  - Large Object



---

# 制約

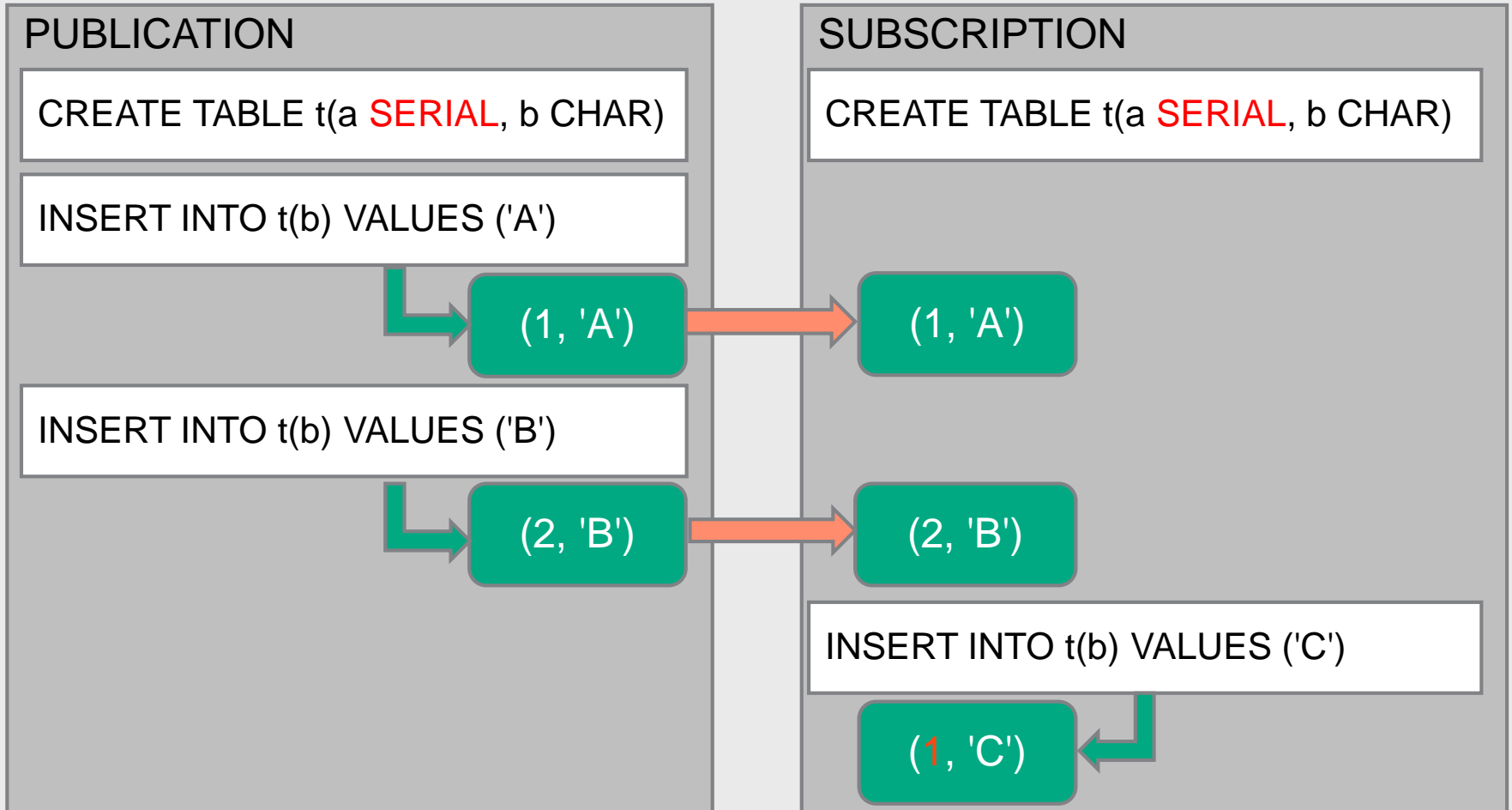
## SERIAL列 / GENERATED AS IDENTITY列

- SERIAL列やGENERATED AS IDENTITY列は内部的にSEQUENCEを使用
- SUBSCRIPTIONへシーケンス値が転送されるが、シーケンスの操作は行われない



# 制約

## SERIAL列 / GENERATED AS IDENTITY列



# 制約

## トリガー

- 一部だけ実行される
  - ROW TRIGGERのみ実行
  - UPDATE **OF**トリガーは発行されない
  - STATEMENT TRIGGERは初期データ転送時のみ実行
- SUBSCRIPTION側
  - ALTER TABLE ENABLE ALWAYS or REPLICA TRIGGER文が必要
  - バグ: 10.0でBEFORE ROW DELETEトリガーが発行されない
    - <https://git.postgresql.org/gitweb/?p=postgresql.git;a=commitdiff;h=360fd1a7b2fe779cc9e696b813b12f6a8e83b558>
    - 10.1でFIX



---

# 制約

## パラメーター log\_statement

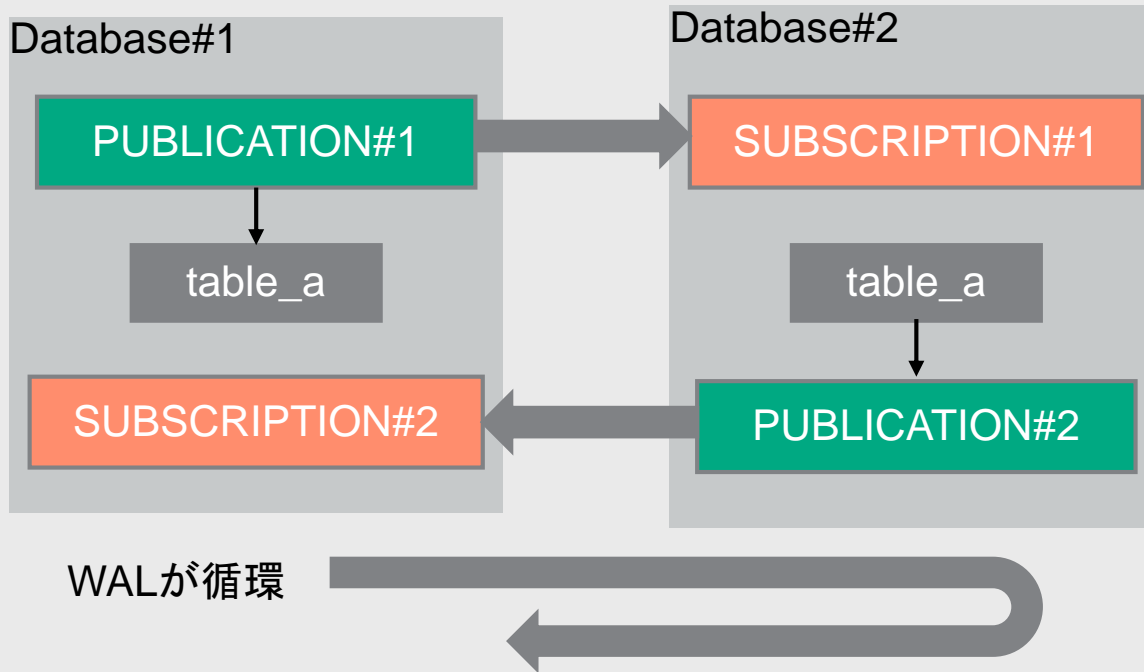
- パラメーター `log_statement = 'all'` に設定してもレプリケーションによる更新ログは出力されない



# 制約

## 双方向レプリケーション

- 双方向レプリケーション不可
  - 設定はできるがWALが循環してエラーになる
  - テーブルが別々であればデータベース間の相互レプリケーションは可能





# 制約


## インスタンス内レプリケーション

- インスタンス内レプリケーションには注意が必要
  - 単純に構成しようとする、CREATE SUBSCRIPTION文がハング
  - SUBSCRIPTIONとReplication Slotを別々に作成する必要がある
- Logical Replication Slotの作成

```
pubdb=# SELECT  
pg_create_logical_replication_slot ('sub1', 'pgoutput') ;
```

- SUBSCRIPTIONの作成

```
subdb=# CREATE SUBSCRIPTION sub1 CONNECTION 'dbname=pubdb' WITH  
(CONNECT=off) ;
```




---

# 制約

## Streaming Replicationとの組み合わせ

- Streaming Replication環境との混在可能
- スレーブ・インスタンスからのLogical Replication不可
  - Logical Replication Slotはスタンバイ・インスタンスでは作成不可
  - Logical Decodingはスタンバイ・インスタンスでは実行不可





# トラブルシューティング

# トラブルシューティング

## リソース不足のログ

- max\_replication\_slots不足 (PUBLICATION)

```
ERROR: could not create replication slot "sub1": ERROR: all
replication slots are in use
```

- max\_wal\_senders不足 (PUBLICATION)

```
FATAL: number of requested standby connections exceeds
max_wal_senders (currently 10)
```

- max\_logical\_replication\_workers不足 (SUBSCRIPTION)

```
WARNING: out of logical replication worker slots
HINT: You might need to increase max_logical_replication_workers.
```

- max\_worker\_processes不足 (SUBSCRIPTION)

ログ無し



# トラブルシューティング

## その他のログ

### – 初期データコピー時の権限不足(PUBLICATION)

```
ERROR:  could not start initial contents copy for table  
"public.data1": ERROR:  permission denied for relation data1
```

### – DROP SUBSCRIPTION文の実行(正常)

```
FATAL:  terminating logical replication worker due to administrator  
command  
LOG:  worker process: logical replication worker for subscription  
16408 (PID 77332) exited with exit code 1
```



# トラブルシューティング

## 競合パターンと動作

– SUBSCRIPTION側で発生する競合と動作

競合パターン	レプリケーション動作	ログ出力
主キー違反／一意キー違反	停止	あり
CHECK制約違反	停止	あり
更新データが存在しない	継続	なし
削除データが存在しない	継続	なし
テーブルが存在しない	停止	あり
一部の列が存在しない	停止	あり
データ型の変換エラー	停止	あり
テーブル・ロック	待機	なし
更新対象レコード・ロック	待機	なし

# トラブルシューティング

## 競合パターンと動作

- 競合発生時のアプリケーションへの影響
  - 競合が発生してもテーブルに対するSQLはブロックされない
- 競合検知の方法
  - ログファイルから検知する
  - `pg_stat_replication.flush_lag / write_lag`
  - `pg_replication_slots.confirmed_flush_lsn ≠ pg_current_wal_lsn()`
- SUBSCRIPTION側の競合発生時の動作
  - 制約違反を検知すると、logical replication workerプロセス停止
  - 5秒後に再起動し、ログ適用を再開
  - 制約違反が解消するまで上記を繰り返し



# トラブルシューティング

## エラー発生時のログ

### – SUBSCRIPTION側で主キー違反のログ

```
ERROR: duplicate key value violates unique constraint "pk_data1"  
DETAIL: Key (c1)=(500) already exists.  
LOG: worker process: logical replication worker for subscription  
16414 (PID 9644) exited with exit code 1
```

### – PUBLICATION側でwal senderのタイムアウトのログ

```
LOG: terminating walsender process due to replication timeout  
LOG: starting logical decoding for slot "sub1"  
DETAIL: streaming transactions committing after 0/5600ED48, reading  
WAL from 0/5600ED10
```





# トラブルシューティング

## エラー発生時のログ

- WALデコード時のメモリー確保エラー
- bytea型の転送時に発生
  - パラメーターbytea\_outputに応じてテキスト変換
  - デフォルトでは「レコード・サイズ x 2 + 1」バイトのメモリーを確保
- PUBLICATIONログ

```
ERROR:  invalid memory alloc request size 258291203
CONTEXT:  slot "sub1", output plugin "pgoutput", in the change
callback, associated LSN 0/2B2543E8LOG:  could not send data to
client: Broken pipe FATAL:  connection to client lost
```

- SUBSCRIPTIONログ

```
ERROR:  could not receive data from WAL stream:
ERROR:  invalid memory alloc request size 258291203
CONTEXT:  slot "sub1", output plugin "pgoutput", in the change
callback, associated LSN 0/2B2543E8
```



# トラブルシューティング

## 競合解消方法

- 自動的に競合は解消しない
- 解消方法は以下の2つ (SUBSCRIPTION側で行う)
  - 競合が発生したレコードを削除する (制約違反の解消)
  - 競合が発生したWALをスキップする (制約違反の解消 / メモリー不足の解消)



# トラブルシューティング

## 競合解消方法

- 競合が発生したWALをスキップする
  - WALの適用を開始するLSNを指定する
- PUBLICATION側で現在のLSNを確認

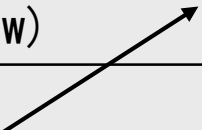
```
postgres=# SELECT pg_current_wal_lsn() ;
pg_current_wal_lsn
-----
0/7200B4F0
(1 row)
```

# トラブルシューティング

## 競合解消方法

- SUBSCRIPTION側でexternal\_idを確認

```
subdb=# SELECT * FROM pg_replication_origin_status ;
local_id | external_id | remote_lsn | local_lsn
-----+-----+-----+-----
      1 | pg_16425    | 0/7200B068 | 0/DA0078E8
(1 row)
```



- external\_id列にはpg\_{pg\_subscription.oid} が出力される

```
subdb=# SELECT oid, subname FROM pg_subscription ;
oid | subname
-----+-----
16425 | sub1
```

# トラブルシューティング

## 競合解消方法

- SUBSCRIPTION側で適用開始LSNを指定

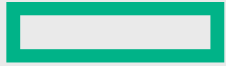
```
subdb=# SELECT
      pg_replication_origin_advance (' pg_16425', ' 0/7200B4F0' ) ;

 pg_replication_origin_advance
-----
(1 row)
```

- エラーになることがある

```
subdb=# SELECT pg_replication_origin_advance(' pg_16399',
' 0/82708760' ) ;
ERROR:  replication origin with OID 1 is already active for PID 5566
```





**Hewlett Packard**  
Enterprise

**Thank you**

noriyoshi.shinoda@hpe.com