PostgreSQL インサイド

論理レプリケーションの復旧方法

ストリーミングレプリケーションのフェイルオーバー併用時

技術を知る

FUJITSU

- [ ] 導入／環境設定
- [ ] 移行
- [ ] 性能
- [ ] チューニング
- [ ] バックアップ／リカバリー
- [x] 冗長化／負荷分散
- [ ] 監視
- [ ] データ連携
- [ ] 災害対策
- [ ] 豆知識

論理レプリケーション（ロジカルレプリケーション）の仕組みと設定方法については、「論理レプリケーション解説 ～ 仕組み、設定方法 ～」で説明しました。可用性が高いシステムでは、ストリーミングレプリケーションを用いてデータベースを冗長化している場合があります。その環境上で論理レプリケーションを用いる場合、ストリーミングレプリケーションがフェイルオーバーした際に、論理レプリケーションも同様に切替え、停止したサーバーを適切に復旧させる必要があります。そこで、この記事では、このような場面での手順や障害時の対応方法についてコマンド実行例や図を用いて丁寧に解説します。
なお、本記事では、PostgreSQL 17 で設定可能なパラメーターおよび機能を使用して説明しています。

ストリーミングレプリケーションと論理レプリケーションを併用する理由

ここでは、ストリーミングレプリケーションと論理レプリケーションを併用する理由から併用時の注意点について説明します。

可用性の考え方

業務システムにおいて、論理レプリケーションを同期モードで運用することで、リアルタイムにデータを複製できます。ただし、論理レプリケーションには、冗長化の範囲がテーブル内のデータのみで、可用性全般を担保する機構がありません。そのため、実際の業務においては、ストリーミングレプリケーション環境上で論理レプリケーションを使用するケースが多いと考えられます（図 1 と図 2 参照）。

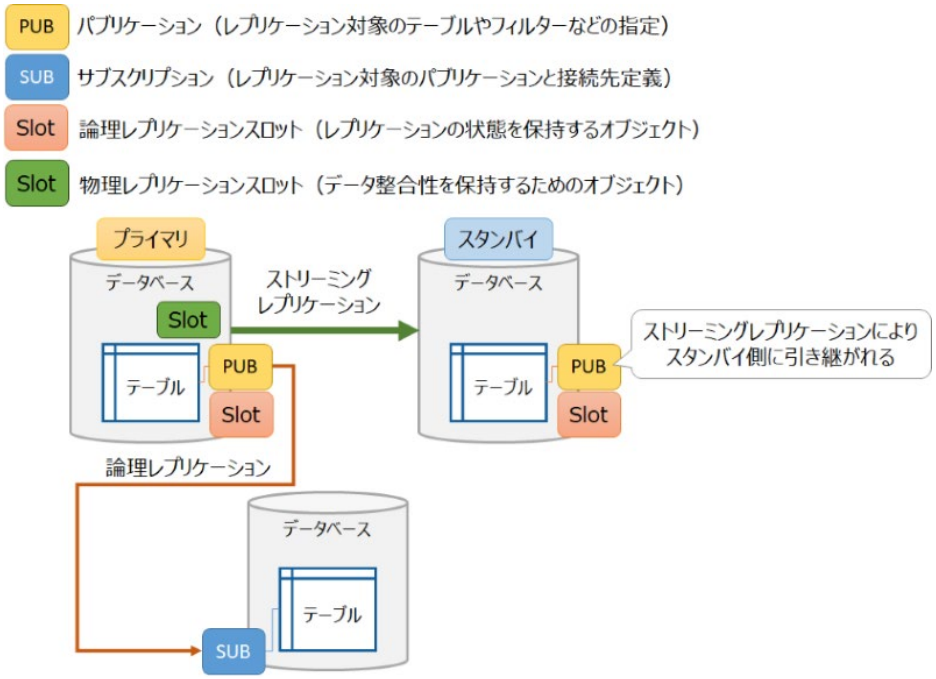

図 1：ストリーミングレプリケーションと論理レプリケーション（パブリケーション）を併用した構成例

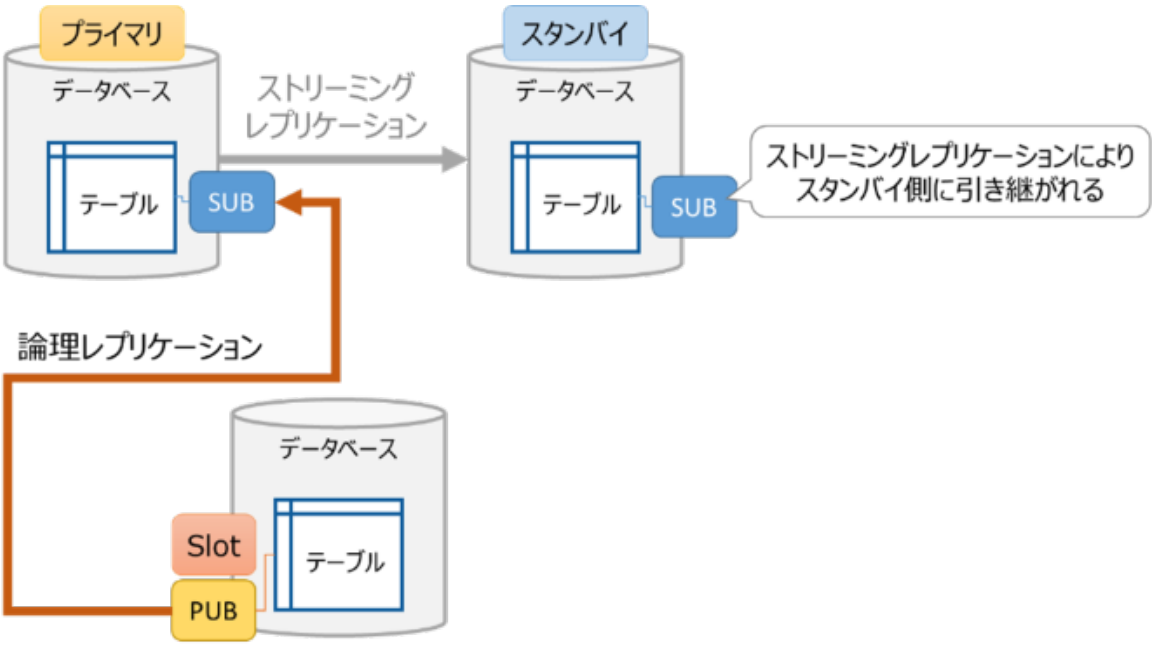

図 2：ストリーミングレプリケーションと論理レプリケーション（サブスクリプション）を併用した構成例

ストリーミングレプリケーションのフェイルオーバー時の対処方法

ストリーミングレプリケーションにてプライマリ側にて障害が発生し、スタンバイが昇格し新プライマリになるとき、論理レプリケーションはプライマリから新プライマリに接続しなおす必要があります。その際、PostgreSQL 16 以前ではデータの整合性を考えた対応が必要であり、サブスクリプションの再作成などが必要でした。
PostgreSQL 17 では、図 1 のように構築時にプライマリ上に物理レプリケーションスロットを設定することで、論理レプリケーションスロットがデータの整合性を保つ仕組みが導入されました。これにより、サブスクリプションの接続先を変更するだけで済むようになりました。
本記事では、PostgreSQL 17 で本機能を利用した場合（以降、PostgreSQL 17 の場合）の手順と、PostgreSQL 16 以前や本機能を使用していない場合（以降、PostgreSQL 16 以前の場合）の手順について解説します。

障害によるフェイルオーバー発生時の復旧の流れ

ストリーミングレプリケーションと論理レプリケーションを併用した構成において、構成サーバーで障害が発生し、ストリーミングレプリケーションでフェイルオーバーが行われた場合、ストリーミングレプリケーションと論理レプリケーションの復旧が必要です。このような障害が発生した場合、「パブリッシャー」や「サブスクライバー」が停止します。以降で、各パターンの復旧の流れを説明します。

- パブリッシャー側が停止した場合（PostgreSQL 17 の場合）
- パブリッシャー側が停止した場合（PostgreSQL 16 以前の場合）
- サブスクライバー側が停止した場合

パブリッシャー側が停止した場合（PostgreSQL 17 の場合）

パブリッシャーを配置したプライマリ側が停止した場合の復旧の流れを解説します。

まず、ストリーミングレプリケーションのスタンバイが新プライマリへ昇格後に論理レプリケーションの状態を確認します（図 3 参照）。

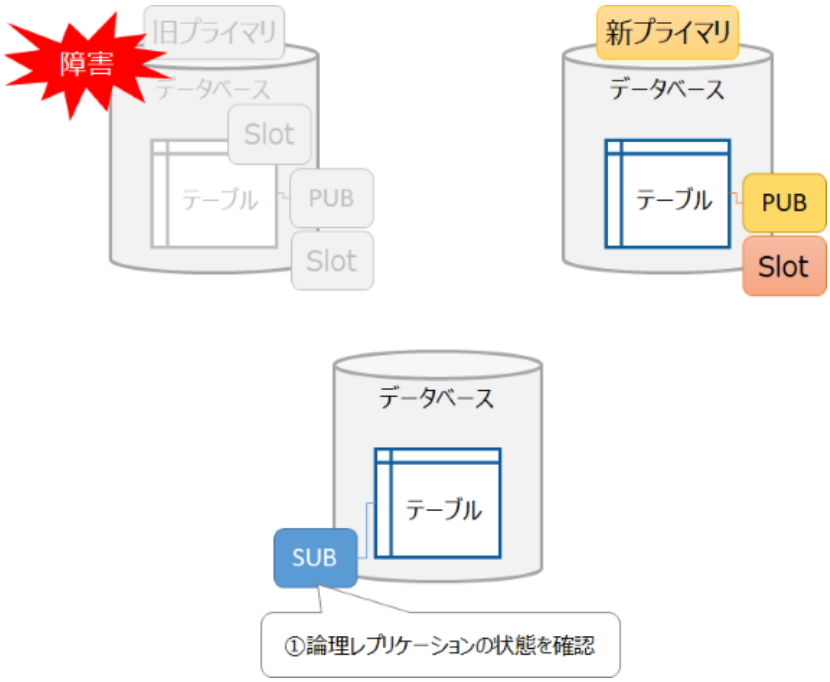

図 3：論理レプリケーションの確認

サブスクライバーの接続先を変更で対処完了します　（図 4 参照）。

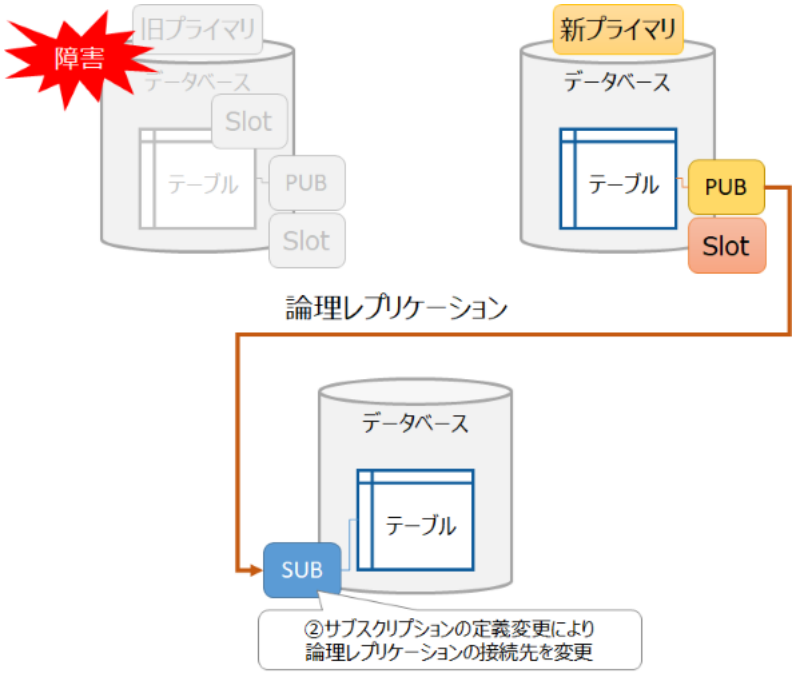

図 4：サブスクライバーの接続先を変更（PostgreSQL 17）

新スタンバイ復旧時に、新プライマリの論理レプリケーションスロットが同期されるよう再作成します（図 5 参照）。

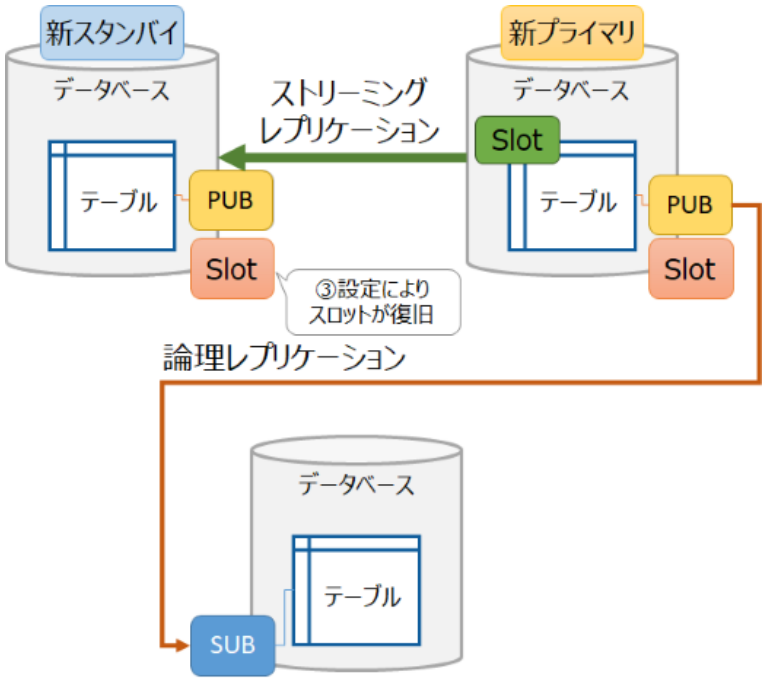

図 5：新スタンバイの復旧（PostgreSQL 17）

各流れの詳細に関しては、「検証：パブリッシャー側が停止した場合の復旧手順」で解説します。

パブリッシャー側が停止した場合（PostgreSQL 16 以前の場合）

パブリッシャーを配置したプライマリ側が停止した場合の復旧の流れを解説します。なお、復旧には「サブスクライバー側のテーブルデータを残した状態で復旧する方法」と、「サブスクライバー側のテーブルデータを空にして初期データ同期から復旧する方法」があります。表 1 を参照していずれかを選択してください。

まず、ストリーミングレプリケーションのスタンバイが新プライマリへ昇格後に論理レプリケーションが実施されていないことを確認します（図 3 参照）。

続いて、表 1 の復旧方法のいずれかにより、サブスクライバーの再作成により、論理レプリケーション接続先を変更します（図 6 参照）。「サブスクライバー側のテーブルデータを空にして初期データ同期から復旧」を選択した場合、初期データ同期と WAL 適用を実施します。一方の「サブスクライバー側のテーブルデータを残した状態で復旧」を選択した場合、WAL 適用のみ実施されます。そのため、表 1 に示したメリットとデメリットが挙げられます。初期データ同期や WAL 適用の流れに関しては、以下の記事で解説しています。

- 論理レプリケーション解説 ～ 仕組み、設定方法 ～

表 1 サブスクライバーの復旧方法

方法	メリット	デメリット
サブスクライバー側のテーブルデータを残した状態で復旧	即座に論理レプリケーションの状態に復旧できる	データの欠落の可能性がある（論理レプリケーション停止中の更新が反映されない）
サブスクライバー側のテーブルデータを空にして初期データ同期から復旧	新プライマリからテーブル内のデータをすべて取得し直すため、データの欠落が発生しない	データ量が多い場合には、初期データ同期に時間がかかる（早期復旧ができない）

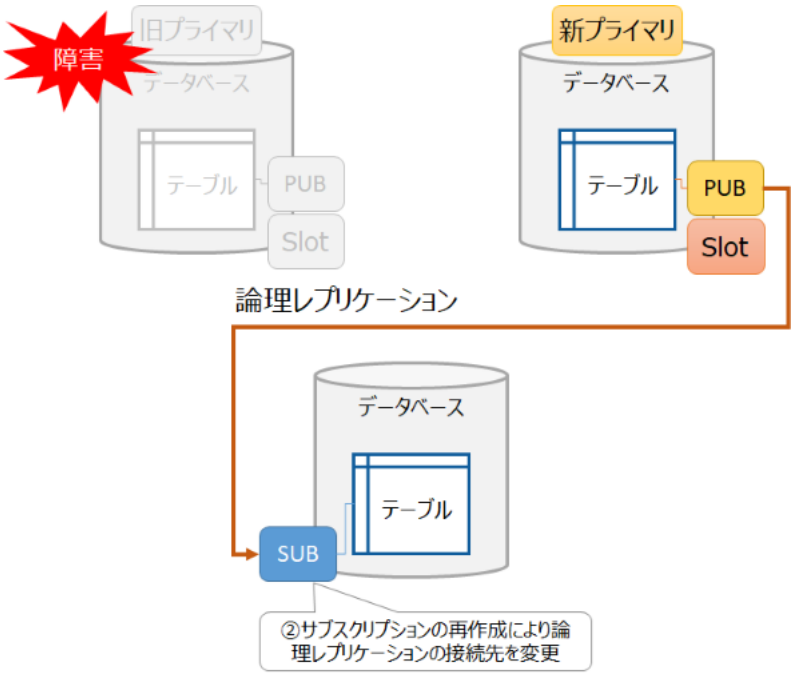

図 6：サブスクライバーの接続先を変更（PostgreSQL 16 以前）

最後に、新スタンバイ復旧後にスロットを削除します（図 7 参照）。

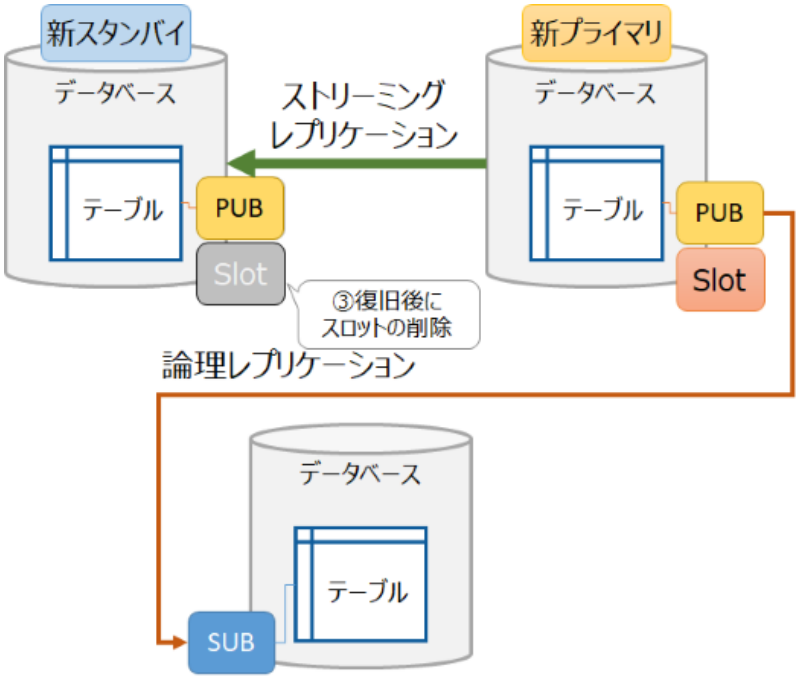

図 7：新スタンバイの復旧（PostgreSQL 16 以前）

各流れの詳細に関しては、「検証：パブリッシャー側が停止した場合の復旧手順」で解説します。

サブスクライバー側が停止した場合

サブスクライバーを配置したプライマリ側が停止しスタンバイ側にフェイルオーバーした場合、スロットやサブスクリプションの接続先は変わりません。また、サブスクリプション側の設定がそのままスタンバイにも引き継がれます。これにより論理レプ

リケーションへの影響はなく、スロットの再割り当てや接続情報を変更する必要がないため、論理レプリケーションをそのまま継続できます。

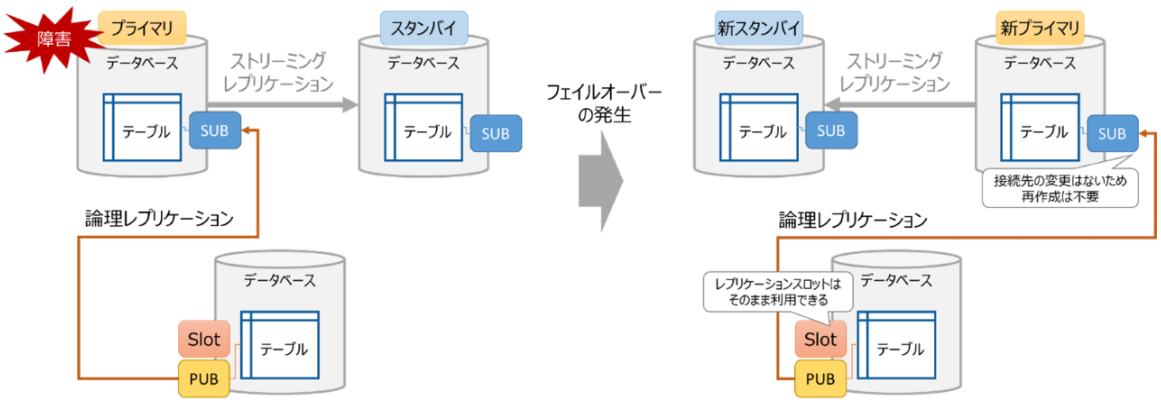

図 8：復旧前後の構成（サブスクライバー側のフェイルオーバー）

検証：パブリッシャー側が停止した場合の復旧手順

図 9 の構成で、パブリッシャー側で障害が発生した際の復旧手順および検証手順を、実行するべきコマンドを示しながら解説します。

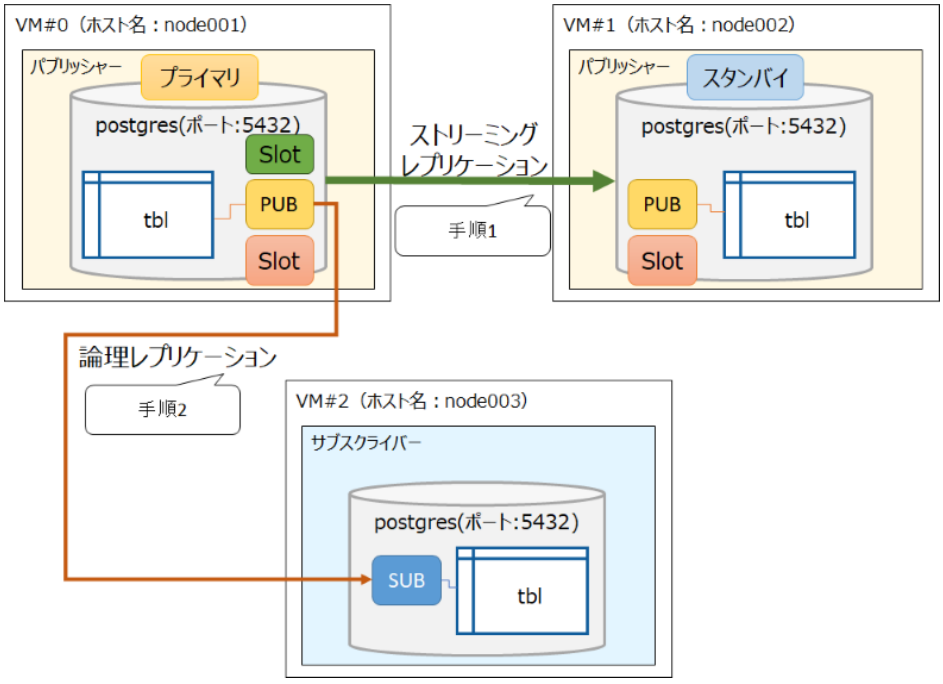

図 9：検証環境（フェイルオーバー前）

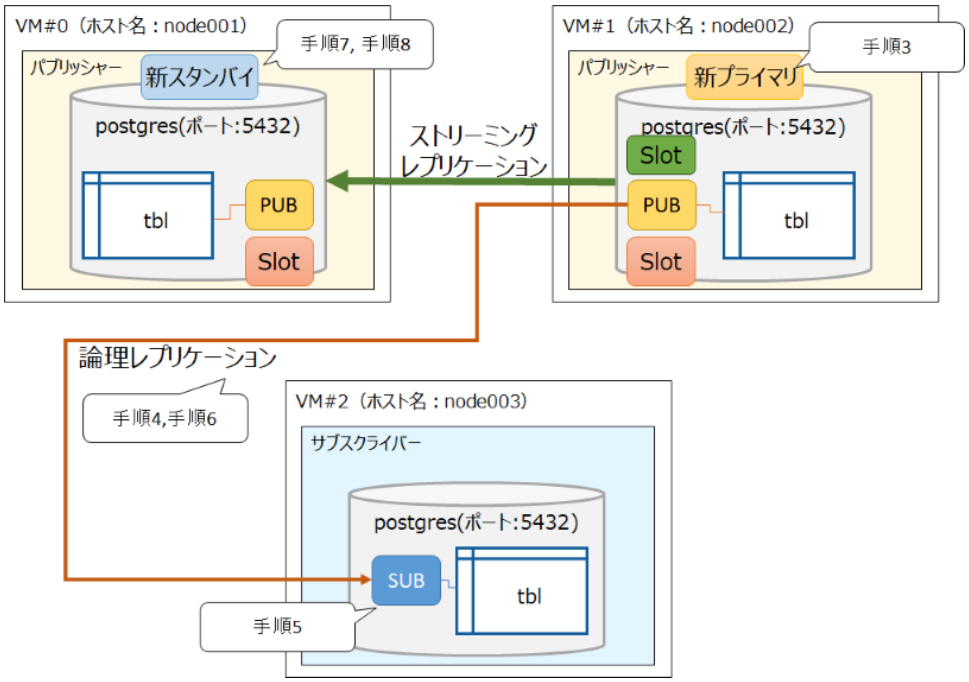

図 10：検証環境（フェイルオーバー後）

手順 1：ストリーミングレプリケーションの構成を作成

ストリーミングレプリケーションの機能概要と設定については、以下の記事で解説しています。設定例が記載されていますので、ぜひお読みください。

- 27.2.5. ストリーミングレプリケーション（PostgreSQL 日本語ドキュメント）
  https://www.postgresql.jp/document/16/html/warm-standby.html#STREAMING-REPLICATION

PostgreSQL 17 の場合、下記の設定および手順を追加します。

プライマリ側の設定

1. postgresql.conf ファイルの設定で synchronized_standby_slots を設定します。このパラメーターの設定値は次の手順にある pg_create_physical_replication_slot にて指定する名前を設定します。

```
synchronized_standby_slots = pslot
```

2. 物理レプリケーションスロットを作成します。

```
postgres=# SELECT pg_create_physical_replication_slot('pslot');
```

スタンバイ側の設定

1. スタンバイサーバを作成します。dbname オプションの dbname パラメーターに追加指定します。このパラメーターは適切なデータベース名を指定する必要があります。

```
pg_basebackup -h node1 -p 5432 -R --dbname='application_name=stb dbname=postgres'
```

2. postgresql.conf ファイルの設定を下記のように追加します。primary_slot_name は、プライマリ側で作成した物理レプリケーションスロット名を指定します。


```
primary_slot_name= 'pslot'
sync_replication_slots=on
hot_standby_feedback=on
```

なお、昇格直後の新スタンバイがない際に synchronized_standby_slots パラメーターが有効の場合、以下のメッセージが出力されるため事前にコメントアウトしておきます。

```
詳細:  Logical replication is waiting on the standby associated with replication slot "pslot".
ヒント:  Create the replication slot "pslot" or amend parameter "synchronized_standby_slots".
```

3. postgresql.auto.conf ファイルの設定にある primary_conninfo にスタンバイ側の設定手順の 1.で指定した内容があることを確認します。

```
primary_conninfo = 'user=(中略) application_name=stb (中略) dbname=postgres'
```

手順 2：論理レプリケーションの構成を作成

論理レプリケーションの機能概要と設定方法については、以下の記事で解説しています。図やコマンド例を用いてわかりやすく説明していますので、ぜひお読みください。

- 論理レプリケーション解説 ～ 仕組み、設定方法 ～

PostgreSQL 17 の場合、下記の設定および手順を追加します。

サブスクライバー側の設定

1. サブスクリプションを作成時、failover オプションを指定します。

```
postgres=# CREATE SUBSCRIPTION sub CONNECTION 'host=node001, port=5432, user=logi_repl_user, dbname=postgres'
PUBLICATION pub WITH (failover=true);
```

論理レプリケーション開始後、スタンバイ側に下記のようなメッセージが出力されレプリケーションスロットがスタンバイ側に同期されます。

```
LOG:  newly created replication slot "sub" is sync-ready now
```

スタンバイ側で以下のコマンドを実行し、failover と synced が t であることを確認します。この設定は、フェイルオーバー発生時に論理レプリケーションの再接続にて再開できる設定になっていることを意味します。

```
postgres=#  SELECT  slot_name, plugin, slot_type, failover, synced
              FROM  pg_replication_slots  WHERE slot_type='logical';
-[ RECORD 1 ]-------
slot_name | sub
plugin    | pgoutput
slot_type | logical
failover  | t
synced    | t
```


手順 3：ストリーミングレプリケーションのプライマリに障害が発生しスタンバイを昇格

1. プライマリの疑似的な故障を発生させるために、プライマリ上（VM#0）で以下のコマンドを実行してください。

```
$ pg_ctl stop -m immediate -w
```

2. スタンバイをプライマリに昇格させるために、スタンバイ上（VM#1）で以下のコマンドを実行します。

```
$ pg_ctl promote
```

コマンド実行後、以下のようなメッセージが出力された場合、スタンバイをプライマリに昇格できたと判断できます。

```
waiting for server to promote.... done
server promoted
```

手順 4：論理レプリケーションの確認

論理レプリケーションが行われていないことを確認するために、サブスクライバー側（VM#2）で以下のコマンドを実行してください。システムカタログ pg_stat_subscription の pid が空、received_lsn が空である場合、論理レプリケーションが行われていない状態であることを把握できます。

```
postgres=# SELECT subname, pid, received_lsn FROM pg_stat_subscription WHERE subname = 'sub';
-[ RECORD 1 ]---------+------
subname               | sub
pid                   |
received_lsn          |
```

手順 5：サブスクライバーの復旧

サブスクライバーの復旧は、PostgreSQL 17 と PostgreSQL 16 以前とで異なります。また、PostreSQL16 以前では 2 つの復旧方法があります。

- サブスクライバーのフェイルオーバー機能で復旧（PostgreSQL 17 の場合）
- サブスクライバー側のテーブルデータを残した状態で復旧（PostgreSQL 16 以前の場合）
- サブスクライバー側のテーブルデータを空にして初期データから復旧（PostgreSQL 16 以前の場合）

手順 5-1：サブスクライバーのフェイルオーバー機能で復旧（PostgreSQL 17 の場合）

本手順のコマンドは、サブスクライバー側（VM#2）で実行してください。

1. サブスクリプションをフェイルオーバー先に接続させるために、以下のコマンドを実行します。以下のコマンド実行直後から論理レプリケーションが再開されます。

```
postgres=# ALTER SUBSCRIPTION sub CONNECTION 'host=node002 port=5432 user=logi_repl_user dbname=postgres'
```

手順 5-2：サブスクライバー側のテーブルデータを残した状態で復旧（PostgreSQL 16 以前の場合）

本手順のコマンドは、サブスクライバー側（VM#2）で実行してください。

1. サブスクリプションを停止させるために、以下のコマンドを実行します。


```
postgres=# ALTER SUBSCRIPTION sub DISABLE;
```

サブスクリプションが停止したことを確認するために、以下のコマンドを実行してください。システムカタログ pg_subscription の subenabled が f である場合、サブスクリプションが停止状態であることを把握できます。

```
postgres=# SELECT subenabled FROM pg_subscription WHERE subname = 'sub';
-[ RECORD 1 ]----+-----------------------------------------------------------
subenabled       | f
```

2. レプリケーションスロットとの対応を無効化するために、以下のコマンドを実行します。

```
postgres=# ALTER SUBSCRIPTION sub SET (slot_name = NONE);
```

3. サブスクリプションを削除するために、以下のコマンドを実行します。

```
postgres=# DROP SUBSCRIPTION sub;
```

サブスクリプションが削除されたことを確認するために、以下のコマンドを実行してください。データが何も出力されなかった場合、サブスクリプションを削除できたことを把握できます。

```
postgres=# SELECT * FROM pg_subscription WHERE subname = 'sub';
```

4. サブスクリプションを再作成するために、以下のコマンドを実行します。copy_data = false を指定することで、サブスクリプション作成後の初期データ同期を無効化できます。以下のコマンド実行直後から論理レプリケーションが再開されます。

```
postgres=# CREATE SUBSCRIPTION sub CONNECTION 'host=node002 port=5432 user=logi_repl_user dbname=postgres' PUBLICATION
pub WITH (copy_data = false);
```

手順 5-3：サブスクライバー側のテーブルデータを空にして初期データ同期から復旧（PostgreSQL 16 以前の場合）

本手順のコマンドは、サブスクライバー側（VM#2）で実行してください。

1. サブスクリプションを停止するために、以下のコマンドを実行します。

```
postgres=# ALTER SUBSCRIPTION sub DISABLE;
```

サブスクリプションが停止したことを確認するために、以下のコマンドを実行してください。システムカタログ pg_subscription の subenabled が f である場合、サブスクリプションが停止状態であることを把握できます。

```
postgres=# SELECT subenabled FROM pg_subscription WHERE subname = 'sub';
-[ RECORD 1 ]----+-----------------------------------------------------------
subenabled       | f
```


2. レプリケーションスロットとの対応を無効化するために、以下のコマンドを実行します。

```
postgres=# ALTER SUBSCRIPTION sub SET (slot_name = NONE);
```

3. サブスクリプションを削除するために、以下のコマンドを実行しましょう。

```
postgres=# DROP SUBSCRIPTION sub;
```

サブスクリプションが削除されたことを確認するために、以下のコマンドを実行してください。データが何も出力されなかった場合、サブスクリプションを削除できたことを把握できます。

```
postgres=# SELECT * FROM pg_subscription WHERE subname = 'sub';
```

4. 論理レプリケーションの対象テーブルのデータを削除するために、以下のコマンドを実行してください。

```
postgres=# TRUNCATE TABLE tbl;
```

対象テーブルのデータが削除されたことを確認するために、以下のコマンドを実行しましょう。データが格納されていない場合、対象テーブルのデータが削除された状態であることを把握できます。

```
postgres=# SELECT * FROM tbl;
```

5. サブスクリプションを再作成するために、以下のコマンドを実行します。以下のコマンド実行直後から論理レプリケーションが再開されます。

```
postgres=# CREATE SUBSCRIPTION sub CONNECTION 'host=node002 port=5432 user=logi_repl_user dbname=postgres' PUBLICATION pub;
```

手順 6：論理レプリケーションの復旧確認

論理レプリケーションが復旧したことを確認するために、サブスクライバー側（VM#2）で以下のコマンドを実行します。システムカタログ pg_subscription の subenabled が t である場合、論理レプリケーションが実施状態であることを把握できます。

```
postgres=# SELECT subenabled FROM pg_subscription WHERE subname = 'sub';
-[ RECORD 1 ]----+-----------------------------------------------------------
subenabled       | t
```

手順 7：ストリーミングレプリケーションの復旧

ストリーミングレプリケーションの復旧手順については、以下の記事で解説しています。図を用いてわかりやすく説明していますので、ぜひお読みください。

- ストリーミングレプリケーション ～ 運用のための機能 ～

PostgreSQL 17 の場合、手順 1 を実施してください。詳細は「手順 1：ストリーミングレプリケーションの構成を作成」を確認ください。

手順 8：不要なレプリケーションスロットの削除（PostgreSQL 16 以前の場合）

新スタンバイ側で論理レプリケーションのために使用したレプリケーションスロットを削除するために、新スタンバイ側（VM#0）で以下のコマンドを実行します。

```
postgres=# SELECT pg_drop_replication_slot('sub');
```

以上の手順により、ストリーミングレプリケーション環境上の論理レプリケーションを復旧できます。

ストリーミングレプリケーションがフェイルオーバーした際に論理レプリケーションを復旧する方法について解説しました。ストリーミングレプリケーション機能と併用し、データベースの高可用化にお役立てください。

可用性をさらに高めたい場合、Fujitsu Enterprise Postgres を利用する方法があります。Fujitsu Enterprise Postgres は、PostgreSQL の機能をエンタープライズ向けに強化した製品であり、異常検知や自動フェイルオーバーの機能として「データベース多重化機能」、データベースへの接続先を切替える機能として「Connection Manager 機能」があります。
本機能の詳細に関しては、以下をご覧ください。

- 業務停止はさせない!トラブル時は自動切替えで PostgreSQL の運用を継続 – 富士通の技術者に聞く！PostgreSQL の技術 –
- PostgreSQL の高可用性を追求！Connection Manager でデータベースへの接続を瞬時に切り替える – 富士通の技術者に聞く！PostgreSQL の技術 –

2025 年 3 月 11 日