



EDB

PG Failover Slots

リリース *1.0.0*

EnterpriseDB Corporation

2023 年 04 月 06 日

目次

第 1 章	PG Failover Slots release notes	3
1.1	Release notes for PG Failover Slots version 1.0.0	3
第 2 章	Installing PG Failover Slots	5
2.1	前提条件	5
2.2	パッケージをインストールする	5
第 3 章	Configuring PG Failover Slots	7
3.1	synchronize_slot_names	7
3.2	drop_extra_slots	8
3.3	プライマリ DSN	8
3.4	スタンバイスロット名	8
3.5	スタンバイ_スロット_最小_確認済み	9
第 4 章	Using PG Failover Slots	11

PG フェイルオーバー スロット (pg_failover_slots) は、論理レプリケーションスロットを物理フェイルオーバー全体で実際に使用できるようにする拡張機能です。

この拡張機能は次のことを行います。

- 欠落しているスロットをプライマリからスタンバイにコピーします
- プライマリで見つからないスロットをスタンバイから削除します
- プライマリに基づいて、スタンバイのスロットの位置を定期的に同期します
- 選択されたスタンバイが、論理スロット walsender のいずれかがコンシューマにデータを送信する前にデータを受信することを保証します

第1章 PG Failover Slots release notes

PG フェールオーバー スロットのドキュメントには、マイナーリリースやパッチを含む PG フェールオーバー スロットの最新バージョンが記載されています。これらのリリースノートでは、各リリースの新機能について説明します。マイナーリリースまたはパッチリリースで導入された新機能については、その機能を導入したリリースに関するコンテンツにもインジケータがあります。

Version	Release Date
<i>Release notes for PG Failover Slots version 1.0.0</i>	2023 Mar 31

1.1 Release notes for PG Failover Slots version 1.0.0

PG フェールオーバー スロットのこのリリースには、次のものが含まれます。

Type	Description
Feature	これは最初のリリースです。

第2章 Installing PG Failover Slots

PG Failover Slots は、使用している Postgres ディストリビューションと同じプラットフォームでサポートされています。PG フェールオーバー スロットのサポートは、Postgres 12 から開始されます。詳細については、以下を参照してください。

- [EDB Postgres Advanced Server Product Compatibility](#)
- [PostgreSQL Product Compatibility](#)
- [EDB Postgres Extended Server Product Compatibility](#)

2.1 前提条件

インストールプロセスを開始する前に:

- Postgres をインストールします。参照: - [Installing EDB Postgres Advanced Server](#)
- [Installing PostgreSQL](#)
- [Installing EDB Postgres Extended Server](#)
- リポジトリをセットアップします。

リポジトリの設定は1回限りのタスクです。リポジトリをすでに設定している場合は、この手順を実行する必要はありません。

リポジトリをセットアップするには、[EDB repositories](#) に移動し、そこにある指示に従います。

2.2 パッケージをインストールする

RPM パッケージ インストール コマンドの構文は次のとおりです。

```
sudo <package-manager> -y install pg-<postgres><postgres_version>-pg-failover_slots  
→<major_version>
```

Debian パッケージ インストール コマンドの構文は次のとおりです。

```
sudo <package-manager> -y install pg-<postgres><postgres_version>-pg-failover_slots-  
→<major_version>
```

どこ :

- <package-manager> は、オペレーティングシステムで使用するパッケージ マネージャーです。

Package manager	Operating system
dnf	RHEL 8 and derivatives
yum	RHEL 7 and derivatives, CentOS 7
zypper	SLES
apt-get	Debian and derivatives

- <postgres> は、使用している Postgres のディストリビューションです。

Postgres distribution	Value
PostgreSQL	pg
EDB Postgres Advanced Server	as
EDB Postgres Extended	postgresextended

- <postgres_version> は、使用している Postgres のバージョンです。
- <major_version> は、インストールする PG Failover Slots のメジャーバージョンです。

たとえば、RHEL 8 プラットフォームに EDB Postgres Advanced Server 15 用の PG Failover Slots 1.0.0 をインストールするには、次のようにします。

```
sudo dnf -y install pg-as15-pg-failover_slots
```

Debian 11 プラットフォームに EDB Postgres Advanced Server 15 用の PG Failover Slots 1.0.0 をインストールするには:

```
sudo apt-get -y install pg-as15-pg-failover_slots-1
```

第3章 Configuring PG Failover Slots

プライマリ インスタンスと、高可用性 (フェールオーバーまたはスイッチオーバー) の目的で使用するスタンバイの両方で、`shared_preload_libraries` に拡張機能を追加する必要があります。

`postgresql.conf` で次の構成オプションを使用して、PG フェールオーバー Slots の動作を構成できます。構成オプションには、`pg_failover_slots.synchronize_slot_names` のように接頭辞 `pg_failover_slots.` が付きます。

3.1 synchronize_slot_names

このフィジカルスタンバイと同期するように論理 Slots を設定できるオプション。

Slots フィルターを使用して、同期する Slots を選択します。これらの Slots フィルターは、具体的に名前が付けられた Slots、SQL LIKE 式に一致する名前を持つ Slots、または Slots が使用しているプラグインの名前と一致させることができます。Slots フィルターを **key:value** ペアとして定義します。キーは次のいずれかになります。

- **name** — 正確な Slots 名 (`name:my_slot` など) に一致します。
- **name_like** — `name_like:change_%` など、SQL LIKE 式に一致する任意の Slots 名に一致します。
- **plugin** — `plugin:test_decoding` などの名前付きプラグインを使用して任意の Slots に一致します。

キーを指定しない場合、システムはデフォルトで **name** として解釈します。したがって、`my_slot` は `name:my_slot` と同等です。

コンマ区切りの文字列値で Slots フィルターを組み合わせることができます。`my_slot`, `name_like:change_%,plugin:test_decoding` を設定すると、`my_slot` という名前の Slots、名前が `change_` で始まるすべての Slots、および `test_decoding` プラグインを使用したすべての Slots が同期されます。

`pg_failover_slots.synchronize_slot_names` が空の文字列に設定されている場合、Slots は同期されません。

`pg_failover_slots.synchronize_slot_names` のデフォルト値は `name_like:%` で、すべての論理レプリケーション Slots に一致し、すべての Slots を同期します。

3.2 drop_extra_slots

`pg_failover_slots.synchronize_slot_names` フィルターを使用して、プライマリで見つからないスタンバイの余分なスロットに何が起こるかを制御します。`true` (デフォルト) に設定すると、それらは削除されます。それ以外の場合は保持されます。

3.3 プライマリ DSN

スロット情報をフェッチするときにプライマリに接続するために使用する接続文字列を指定します。

空の場合 (デフォルト)、`primary_conninfo` と同じ接続文字列を使用します。

!!!注: 接続文字列にパスワードフィールドがある場合、`primary_conninfo` は使用できません。パスワードは PostgreSQL によって難読化され、PG フェールオーバースロットは実際にはパスワードを見ることができません。この場合、`pg_failover_slots.primary_dsn` を構成する必要があります。

3.4 スタンバイスロット名

フェイルオーバー候補のストリーミング物理レプリカが、サブスクライバーに表示される前にすべての変更を受信してフラッシュしたことを確認します。このオプションは、論理スロットのコンシューマーのスタンバイへのフェイルオーバー時にコミットが消失しないことを保証します。

コンマ区切りの `pg_failover_slots.standby_slot_names` リストに名前が含まれているレプリケーションスロットは、プライマリの `walsender` によって特別に扱われます。

論理レプリケーション `walsender` は、`walsender` が論理レプリケーションスロットの変更を送信する前に、すべてのローカル変更が送信され、`pg_failover_slots.standby_slot_names` のレプリケーションスロットにフラッシュされることを保証します。事実上、スロットの名前付きリストと `walsender` からの論理的にデコードされたストリームのすべてのコンシューマとの間に同期レプリケーションバリアを提供します。

`pg_failover_slots.standby_slot_names` で任意のレプリケーションスロットを一覧表示できます。論理スロットと物理スロットの両方が機能しますが、通常は物理スロットに使用されます。

このセーフガードがないと、次の 2 つの異常が発生する可能性があります。コミットがサブスクライバーによって受信され、フェールオーバー候補がまだ受信されていないため、フェールオーバー時にプロバイダーから消失する可能性があります。

- 1 つまたは複数のサブスクライバーの場合、サブスクライバーは変更を適用した可能性があります、新しいプロバイダーは、受信した変更と競合する新しいトランザクションを実行する可能性があります。これは、プロバイダーに関する限り発生したことがないためです。
- 2 つ以上のサブスクライバーの場合、フェールオーバー時に、すべてのサブスクライバーが変更を適用するわけではありませんでした。コミットを受信しなかったサブスクライバーは現在コミットを取得する方法がないため、サブスクライバーは一貫性のない、和解できない状態になっています。

`pg_failover_slots.standby_slot_names` を設定すると、設計上、プロバイダーのフェイルオーバー候補のレプリカが追いついていない場合、サブスクライバーはプロバイダーに遅れをとります。したがって、監視は不可欠です。

3.5 スタンバイ_スロット_最小_確認済み

データが論理レプリケーションスロットを介して送信される前に確認する必要がある `pg_failover_slots.standby_slot_names` の数を制御します。 `-1` (デフォルト) を設定すると、`pg_failover_slots.standby_slot_names` のすべてのエントリを待機するように指定されます。

第4章 Using PG Failover Slots

フェールオーバー中、一貫性の理由により、スロットはすぐにはスタンバイに同期されません。`pg_failover_slots` 拡張機能がアクティブ化されている場合、スタンバイは論理スロットよりも遅れているか、プライマリの論理スロットよりも進んでいる可能性があります。拡張機能は検証を行い、実際に安全な場合にのみスロットを同期します。

ただし、これにより、スロットが同期されていること、およびスタンバイが実際にすべてのスロットの一貫した論理デコードを使用してフェイルオーバーターゲットになる準備ができていることを確認する必要があります。これは最初だけ行う必要があります。スロットが初めて同期されると、クラスター内で拡張機能がアクティブである限り、スロットは常に一貫しています。

スロットがプライマリと完全に同期しているかどうかのチェックは比較的簡単です。スロットは、スタンバイの `pg_replication_slots` ビューに存在し、アクティブ状態 `false` を持っている必要があります。アクティブ状態の `true` は、スロットが現在初期化されていることを意味します。

たとえば、次の `psql` セッションについて考えてみます。

```
# SELECT slot_name, active FROM pg_replication_slots WHERE slot_type = logical;
```

__OUTPUT__

slot_name	active
regression_slot1	f
regression_slot2	f
regression_slot3	t

この例は、スロット `regression_slot1` と `regression_slot2` がプライマリからスタンバイに同期され、`regression_slot3` がまだ同期されていることを示しています。この段階でフェイルオーバーが発生すると、`regression_slot3` が失われます。

少し待ってから、再度クエリを実行できます。

```
```sql # SELECT slot_name, active FROM pg_replication_slots WHERE slot_type = logical;
```

```
OUTPUT slot_name | active - - - - - + - - - regression_slot1 | f regression_slot2 | f regression_slot3 | f ```
```

これで、3つのスロットすべてが同期されました。それらの論理デコード状態を失うことなく、スタンバイをフェイルオーバーに使用できます。