PostgreSQL インサイド

PostgreSQL 15 での論理レプリケーションの改善

PostgreSQL 15 でコミットされた機能の紹介

技術者 Blog

Amit Kapila
FUJITSU Limited
Software Products Business Unit Data Management Division Senior Director
PostgreSQL Committer and Major Contributor

はじめに

PostgreSQL はバージョンアップごとに、パーティショニング、論理レプリケーション、並列クエリ、バキュームなどのさまざまな分野が改善されています。このブログでは、2022 年 10 月にリリースされた PostgreSQL 15 でユーザーが確認できる論理レプリケーションのさまざまな機能強化についてまとめます。

PostgreSQL 15 のリリースと論理レプリケーションについては、PostgreSQL オフィシャルのページを参照してください。

- Release Notes（PostgreSQL オフィシャルのページへ）
  https://www.postgresql.org/docs/release/15.0/
- Chapter 31. Logical Replication（PostgreSQL オフィシャルのページへ）
  https://www.postgresql.org/docs/devel/logical-replication.html

プリペアドトランザクションのレプリケーション

PostgreSQL 14 では、プリペアドトランザクションのロジカルデコードを可能にし 、PostgreSQL 15 では、論理レプリケーションにプリペアドトランザクションをレプリケートする機能を追加しました。以前は、プリペアドトランザクションの変更は COMMIT PREPARED コマンドの実行後にのみ送信されていました。PostgreSQL 15 では、次の構文を使用して”PREPARE”の時にレプリケーションを有効にできます。

```
CREATE PUBLICATION  mypub FOR ALL TABLES;
CREATE SUBSCRIPTION mysub CONNECTION 'dbname=postgres' PUBLICATION  mypub WITH (two_phase = true);
```

この機能の主な利点は次のとおりです。

- （a）レプリケートするタイミングを COMMIT PREPARED まで待たずに、”PREPARE”の時から行うことで、データのレプリケーションにかかる時間を短縮できます。
- （b）サブスクライバーノードでプリペアに失敗すると、パブリッシャーノードでも同様にロールバックできるため、コンフリクトのない論理レプリケーションを構築するための基盤が提供されます。

この機能の重要なポイントは次のとおりです。

- （a）すべての初期テーブル同期が完了すると、プリペアドトランザクションのレプリケーションが有効になります。
- （b）適用中のプリペアドトランザクションでのコンフリクトを避けるために、プリペア識別子を pg_gid_<subscriber-id>_<transaction-id> として使用します。
- （c）ALTER SUBSCRIPTION コマンドで two_phase オプションを変更することはできません。
- （d）サブスクリプション作成時（CREATE SUBSCRIPTION）に two_phase コミットが有効になっていると、ALTER SUBSCRIPTION REFRESH PUBLICATION は copy_data=false を指定することで実行できます。

この機能の詳細な説明については、以下のブログを参照してください。

- 論理レプリケーションにおける二相コミット

スキーマ内の全テーブルのレプリケーション

PostgreSQL 14 以前では、特定のスキーマのすべてのテーブルをパブリッシュする場合、パブリケーションの作成時に特定のスキーマのすべてのテーブルを指定する必要がありました。その後、ユーザーがそのスキーマでさらにテーブルを作成する場合は、それらもパブリケーションに個別に追加する必要がありました。これはユーザーにとって不便でした。PostgreSQL 15 では改善された機能により、ユーザーはスキーマのすべてのテーブルをパブリッシュしたい場合にスキーマ名だけの指定で済むようになり、非常に簡単になります。スキーマ名 (TABLES IN SCHEMA) を指定する構文は次のとおりです。

```
CREATE PUBLICATION mypub FOR TABLES IN SCHEMA mysch;
```

```
CREATE PUBLICATION mypub FOR TABLE mytab, TABLES IN SCHEMA mysch;
```

上の 2 つめの構文のように、デフォルトのスキーマのテーブルと別のスキーマを同時に指定することもできます。

ユーザーは、次の構文で既存のパブリケーションにスキーマを追加できます。

```
ALTER PUBLICATION mypub ADD TABLES IN SCHEMA mysch;
```

一部のサブスクライバーによってすでにサブスクライブされているパブリケーションにスキーマを追加する場合は、サブスクライバー側で"ALTER SUBSCRIPTION ... REFRESH PUBLICATION"を実行する必要があることに注意してください。

この機能の詳細な説明については、以下のブログを参照してください。

- スキーマ内のテーブルを対象とした論理レプリケーション

論理レプリケーションの行フィルター

この機能により、パブリケーション定義の各テーブルの後に追加の WHERE 句を指定できます。この WHERE 句を満たさない行はフィルタリング（除外）されます。これにより、一連のテーブルを部分的にレプリケーションできます。行フィルターはテーブルごとに行われます。WHERE 句は括弧()で囲む必要があります。ユーザーは、次のコマンドを使用して行フィルターを定義できます。

```
CREATE PUBLICATION mypub FOR TABLE mytab1 WHERE (c1 > 10 and c2 < 20), mytab2 WHERE (c3 LIKE 'bob');
```

ユーザーは、次のコマンドを使用して、既存のパブリケーション内のテーブルに行フィルターを指定できます。

```
ALTER PUBLICATION mypub SET TABLE mytab1 WHERE (c1 > 10 and c2 < 20), mytab2 WHERE (c3 LIKE 'bob');
```

これは、ノード間でデータを分散させたり、データを選択的に送信してパフォーマンスを向上させたり、一部の機密データをレプリケーションから除外したりするのに役立ちます。

この機能の重要なポイントは次のとおりです。

- （a）UPDATE コマンドや DELETE コマンドの操作をパブリッシュする場合、行フィルターの WHERE 句には、テーブルにレプリカアイデンティティ（REPLICA IDENTITY）が設定されている列を指定する必要があります。

- （b）INSERT コマンドを発行するパブリケーションに追加されたテーブルの行フィルターの WHERE 句は、任意の列を使用できます。
- （c）TRUNCATE TABLE コマンドでは、行フィルターは無視されます。
- （d）行フィルターが NULL または"false"と評価された場合、対応する行はレプリケートされません。
- （e）WHERE 句は、ユーザー定義関数、ユーザー定義演算子、ユーザー定義型、ユーザー定義照合順序、非 immutable の組み込み関数、またはシステム列参照を持たない単純な式のみを許可します。
- （f）初期テーブル同期中に、行フィルターを満たすデータのみがサブスクライバーにレプリケートされます。
- （g）パーティション分割されたテーブルの場合、パブリケーションのパラメーター publish_via_partition_root の値によって使用される行フィルターが決まります。publish_via_partition_root が true の場合、ルートパーティション化されたテーブルの行フィルターが使用されます。publish_via_partition_root が false（デフォルト)の場合は、各パーティションの行フィルターが使用されます。

この機能の詳細な説明については、以下を参照してください。

- 31.3 Row Filters（PostgreSQL オフィシャルのページへ）
  https://www.postgresql.org/docs/devel/logical-replication-row-filter.html
- 論理レプリケーションにおけるパブリケーションの行フィルター

論理レプリケーションの列リスト

この機能により、論理レプリケーションにテーブルを追加するときに、オプションの列リストを指定できます。このリストに含まれていない列はサブスクライバーに送信されないため、サブスクライバー側のスキーマはパブリッシャー側のスキーマのサブセットにすることができます。列を選択的に送信して、パフォーマンスを向上させたり、一部の機密データをレプリケーションから除外したりするのに役立ちます。ユーザーは、次の構文で列リストを定義できます。

```
CREATE PUBLICATION mypub FOR TABLE mytab1 (c1, c2), mytab2 (c3);
```

ユーザーは、次のコマンドを使用して、既存のパブリケーション内のテーブルの列リストを指定できます。

```
ALTER PUBLICATION mypub SET TABLE mytab1 (c1, c2);
```

この機能の重要なポイントは次のとおりです。

- （a）パブリケーションが UPDATE コマンドや DELETE コマンドをパブリッシュする場合、列リストにテーブルのレプリカアイデンティティ（REPLICA IDENTITY）列を含める必要があります。
- （b）パブリケーションが INSERT コマンドのみをパブリッシュする場合、列リストはレプリカアイデンティティ列を省略できます。
- （c）TRUNCATE TABLE コマンドの列リストは無視されます。
- （d）列リストには、単純な列参照のみを含めることができます。
- （e）パブリケーションが FOR TABLES IN SCHEMA もパブリッシュしている場合、列リストは指定できません。
- （f）初期データ同期中は、パブリッシュされた列のみがレプリケートされます。
- （g）パーティション分割されたテーブルの場合、パブリケーション パラメーター publish_via_partition_root によって、使用される列リストが決まります。publish_via_partition_root が true の場合、ルートパーティションテーブルの列リストが使用されます。それ以外の場合、publish_via_partition_root が false (デフォルト) の場合、各パーティションの列リストが使用されます。

この機能の詳細な説明については、以下を参照してください。

- 31.4 Column Lists（PostgreSQL オフィシャルのページへ）
  https://www.postgresql.org/docs/devel/logical-replication-col-lists.html
- 論理レプリケーションにおける列リスト

サブスクリプションの所有者権限

PostgreSQL 14 以前は、サブスクリプションの適用プロセスはスーパーユーザーの権限で実行されていましたが、PostgreSQL 15 ではサブスクリプションの所有者の権限で実行されます。そのため、サブスクリプションの所有者に権限がない限り、論理レプリケーションワーカーはテーブルに対して INSERT、UPDATE、DELETE、TRUNCATE、COPY コマンドを実行できなくなります。行レベルのセキュリティポリシーを使用してテーブルにレプリケートできるのは、スーパーユーザー、BYPASSRL 属性（注）を持つロール、およびテーブルの所有者のみです。

- 注）CREATE ROLE で付与する属性の 1 つです。

この対応の目的は、スーパーユーザー以外でもサブスクリプションを管理できるようにし、パブリッシャー側での悪意のある行動からサブスクリプションを持つサーバーを保護することです。

コンフリクトの解決

コンフリクトは、サブスクライバーでのトランザクションの適用中に、主キー違反、スキーマの違いなど、さまざまな理由で発生する可能性があります。デフォルトでは、PostgreSQL はエラー時に操作を再試行し続けます。PostgreSQL 14 以前では、ユーザーは次の選択肢があります。

- （a）競合するデータを手動で削除して、レプリケーションを続行できるようにする。
- （b）pg_replication_origin_advance()関数を使用して、失敗したトランザクションより先に LSN（Log Sequence Number）を進め、再起動時に競合するトランザクションの後の時点からレプリケーションが開始されるようにする。

（a）の場合、パブリッシャーからの対応するデータを無視したいときでも、ユーザーはサブスクライバーのデータを削除または変更する必要があるため、ユーザーがこれらの方法を使用するのは非常に不便です。
（b）の場合、パブリッシャー側で pg_waldump コマンドまたはその他のツールを使用して、ユーザーは失敗したトランザクションの LSN を見つける必要があります。また、元の情報はレプリケーションのために内部的に生成されたものであるため、明らかではありません。pg_replication_origin_advance()関数を使用しているときに、ユーザーが間違った LSN(この先のコミットまたはトランザクション間の何らかの操作のいずれか)を設定したとき、システムは想定外のデータを省略し、一貫性のない複製を生成してしまう可能性があります。

もう 1 つの問題は、トランザクションが成功するまで、人手で止めない限り、トランザクションを適用し続けることです。ユーザーが ALTER SUBSCRIPTION mysub DISABLE を使用して手動でサブスクリプションを無効にするしか、トランザクションを停止する方法がありません。

PostgreSQL 15 では、必要な情報をサーバーログで提供することと、より堅牢な手段を提供することで、pg_replication_origin_advance()関数をより簡単に使用できるようにしました。その他にも、エラー発生時に自動でサブスクリプションを無効にする機能を提供しました。
新しくサブスクリプションのオプション”disable_on_error”が導入され、パブリッシャーからのデータレプリケーション中にサブスクリプションワーカーが何らかのエラーを検出した場合、サブスクリプションを自動的に無効化できます。このオプションは CREATE SUBSCRIPTION または ALTER SUBSCRIPTION コマンドで指定できます。

```
CREATE SUBSCRIPTION mysub CONNECTION '…' PUBLICATION mypub WITH (disable_on_error = true);
```



```
ALTER SUBSCRIPTION mysub SET (disable_on_error = true);
```

さらに、サブスクリプションのワーカーエラーのエラーコンテキストを拡張し、finish LSN を追加しました。これは、コミット済のトランザクションのための commit_lsn と、プリペアドトランザクションのための prepare_lsn を示します。次に、レプリケーション起点名です。これには、レプリケーションの進行状況を追跡し、サブスクリプション定義で自動的に作成されるレプリケーション元の名前が含まれます。

拡張されたサーバーログのエラーコンテキストにより、ユーザーは pg_replication_origin_advance()関数をより簡単に使用できるようになります。そして、より堅牢な方法として、ALTER SUBSCRIPTION mysub SKIP (lsn = '0/1566D10')コマンドを使用して競合するトランザクションをスキップする方法も導入しました。これは指定された LSN に対してチェックを行うことにより、ユーザーが間違った LSN を設定するのを防ぐための、より堅牢な方法です。指定された LSN は、再起動後にパブリッシャーから送信される最初のトランザクションの finish LSN と同じでなければなりません。サブスクライバーに正常に適用された最初のトランザクションは、指定された LSN をクリアします。また、指定された LSN は、起点 LSN よりも大きくなければなりません。

この機能の詳細な説明については、以下のブログを参照してください。

- 論理レプリケーションにおけるコンフリクトの対処方法
- 論理レプリケーションにおけるコンフリクトの対処方法（ALTER SUBSCRIPTION …SKIP コマンド）

システムビューpg_stat_subscription_stats の提供

これは、論理レプリケーションの変更適用中または初期テーブル同期中に発生したエラーに関する統計を表示する新しいシステムビューです。

この機能の詳細については、以下を参照してください。

- 28.2.9 pg_stats_subscription_stats（PostgreSQL オフィシャルのページへ）
  https://www.postgresql.org/docs/devel/monitoring-stats.html#MONITORING-PG-STAT-SUBSCRIPTION-STATS

パブリッシャーとサブスクライバー間の通信の改善

PostgreSQL 15 では、次のような改善を行いました。

(a) すべてのトランザクションデータがフィルタリング（除外)されるようなトランザクションの BEGIN/END メッセージの送信を防ぎます。

(b) ほとんど、またはすべてのデータがフィルタリング（除外）されるような大規模なトランザクションの処理中のタイムアウトによって、レプリケーションが再開されるのを防ぎます。

PostgreSQL 14 以前では、空のトランザクション (すべての変更がスキップ/フィルタリングされる) に対して、BEGIN/END メッセージを送信していましたが、このようなメッセージを作成して送信するために多くの CPU サイクルとネットワーク帯域幅を浪費していました。PostgreSQL 15 では、空のトランザクションに対するメッセージの送信を避けるために、パブリッシャーからサブスクライバーに送信される最初の変更に対してのみ BEGIN メッセージの送信を開始します。BEGIN が送信されたときにのみ END (COMMIT) メッセージの送信を許可します。同期レプリケーションの遅延を回避するために、空のトランザクションをスキップした後にキープアライブメッセージを送信し、そのフィードバックを処理します。

長いトランザクションの処理中に、変更のほとんどがフィルタリング（除外）されている、いわば、特定の操作がパブリッシュされない場合、パブリッシャーはサブスクライバーに通信を送信しません。これは、通信しない時間が一定の閾値を超えると、レプリケーションの再起動につながってしまうタイムアウトが発生するためです。これを解決するために、私たちはこのような場合にキープアライブメッセージを定期的に送信するようにしました。

この機能の詳細については、以下のブログを参照してください。

- 論理レプリケーションにおける通信の改善

この記事は PostgreSQL 15 における様々な論理レプリケーションの改善点を説明しました。読者の皆さんのお役に立てば幸いです。

参考

PostgreSQL 14 での論理レプリケーションの機能強化については、以下のブログで紹介しています。

- PostgreSQL　14 での論理レプリケーションの改善

2022 年 12 月 26 日