PostgreSQL インサイド

PostgreSQL 15 での論理レプリケーションのパーミッション PostgreSQL 15 でコミットされた機能の紹介

技術者 Blog

Ajin Cherian
Fujitsu Australia Limited
Senior Software Development Engineer

はじめに

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権限で実行されるようになりました。
本記事では、PostgreSQL 15 で導入された論理レプリケーションにおける権限チェックに関する重要な変更について、これがどのように機能するかを紹介します

背景

現在、サブスクリプションを作成できるのはスーパーユーザーのみです。
PostgreSQL 14 以前では、サブスクリプションがスーパーユーザーによって作成され、そのユーザーが後に非スーパーユーザーに変更されても、サブスクリプションの適用ワーカおよびテーブル同期ワーカは、引き続き論理レプリケーションの変更をスーパーユーザー権限で適用し続けていました。しかし、スーパーユーザー権限を前提として変更を適用し続けることは、セキュリティ違反になります。

機能概要

PostgreSQL 15 では、サブスクライバー上の論理レプリケーションの一環として、サブスクリプションの所有者が必要な操作を実行できる権限（適用ワーカやテーブル同期ワーカが INSERT、UPDATE、DELETE、TRUNCATE または COPY コマンドを実行できる権限）を持っているかを確認します。サブスクリプションの所有者が必要な権限を持っていない場合、操作はエラーで失敗します。

この権限チェックは、各トランザクションが適用されるときに実行されます。ワーカがトランザクションの変更を適用中に、同時実行中のトランザクションによってサブスクリプションの所有者が変更された場合、現在のトランザクションの適用は変更前の所有者の権限で続行されます。

テーブルには、通常の問い合わせで取得できる行、またはデータ変更コマンドで挿入、更新または削除できる行をユーザーごとに制限する、行単位セキュリティ （RLS：row-level-security） ポリシーを設定できます。PostgreSQL 14 以前では、サブスクリプション所有者は、サブスクリプション所有者が従う RLS ポリシーを持つテーブルにレプリケーションすることで RLS を迂回できていましたが、PostgreSQL 15 以降は迂回できないように変更されました。このような RLS ポリシーを持つテーブルへのレプリケーションの制限は、レプリケートされる INSERT、UPDATE、DELETE または TRUNCATE をポリシーが通常、許可するかどうかに関係なく発生します。どの特定の操作（例えば UPDATE のみ）が RLS ポリシーによって制限されているかに関わらず、サブスクリプションの所有者に RLS ポリシーが設定されているテーブルへのすべてのレプリケーションは禁止されます。なお、スーパーユーザー、「BYPASSRLS」属性を持つロール、および対象となるテーブルの所有者は、RLS ポリシーに関係なくレプリケートできます。

アクション	PostgreSQL 14 以前	PostgreSQL 15
スーパーユーザー以外のロールによるサブスクリプションの作成	許可されない	許可されない
サブスクリプション所有者であるロールを非スーパーユーザーレベルに変更	許可	許可

アクション	PostgreSQL 14 以前	PostgreSQL 15
サブスクリプション（現在は非スーパーユーザーが所有）が、スーパーユーザーの権限で論理レプリケーションの変更を適用し続ける	許可（非推奨）	許可されない
スーパーユーザーであるサブスクリプション所有者に適用される RLS ポリシーを持つテーブルへの論理レプリケーション	許可	許可
スーパーユーザーではなく、BYPASSRLS 属性を持つサブスクリプション所有者に適用される RLS ポリシーを持つテーブルへの論理レプリケーション	許可	許可
スーパーユーザーではないサブスクリプション所有者に適用される RLS ポリシーを持つテーブルへの論理レプリケーション	許可（非推奨）	許可されない

例

次に、サブスクリプションを作成したスーパーユーザーが、後で非スーパーユーザーに降格された場合に、変更の適用に失敗するサブスクリプションの例を示します。

サブスクライバーとパブリッシャーの両方にスーパーユーザーとしてロール 「alice」 と 「bob」 を作成し、これらのロールを使用していくつかのテーブルを作成します。

パブリッシャーおよびサブスクライバーでの実行例：

```
postgres=#   CREATE ROLE alice SUPERUSER LOGIN;
CREATE ROLE
postgres=#   CREATE ROLE bob SUPERUSER LOGIN;
CREATE ROLE
postgres=#   GRANT CREATE ON DATABASE postgres TO alice;
GRANT
postgres=#   GRANT CREATE ON DATABASE postgres TO bob;
GRANT
postgres=#   SET SESSION AUTHORIZATION alice;
SET
postgres=#   CREATE TABLE alice_table (i INTEGER);
CREATE TABLE
postgres=#   ALTER TABLE alice_table REPLICA IDENTITY FULL;
ALTER TABLE
postgres=#   GRANT SELECT ON TABLE alice_table TO alice;
GRANT
postgres=#   SET SESSION AUTHORIZATION bob;
SET
postgres=#   CREATE TABLE bob_table (i INTEGER);
CREATE TABLE
postgres=#   ALTER TABLE bob_table REPLICA IDENTITY FULL;
```



```
ALTER TABLE
postgres=#   GRANT SELECT ON TABLE bob_table TO bob;
GRANT
```

ロール「alice」を使用してパブリッシャーに FOR ALL TABLES パラメーターを指定してパブリケーション（alice_pub）を作成します。

パブリッシャーでの実行例：

```
postgres=# SET SESSION AUTHORIZATION alice;
SET
postgres=# CREATE PUBLICATION alice_pub FOR ALL TABLES;
CREATE PUBLICATION
```

ロール「alice」を使用してサブスクライバーにサブスクリプション（alice_sub）を作成します。

サブスクライバーでの実行例：

```
postgres=# SET SESSION AUTHORIZATION alice;
SET
postgres=# CREATE SUBSCRIPTION alice_sub CONNECTION 'dbname=postgres host=localhost port=6972' PUBLICATION alice_pub;
NOTICE:  created replication slot "alice_sub" on publisher
CREATE SUBSCRIPTION
```

パブリッシャーのテーブル（alice_table と bob_table）にデータを挿入し、サブスクライバーにレプリケートされていることを確認します。

パブリッシャーでの実行例：

```
postgres=# INSERT INTO alice_table VALUES (1);
INSERT 0 1
postgres=# SELECT * FROM alice_table;
 i
---
 1
(1 row)
postgres=# INSERT INTO bob_table VALUES (1);
INSERT 0 1
postgres=# SELECT * FROM bob_table;
 i
---
 1
(1 row)
```

サブスクライバーでの実行例：

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



```
 i
---
 1
(1 row)
postgres=# SELECT * FROM bob_table;
 i
---
 1
(1 row)
```

次に、サブスクリプションの所有者である「alice」のスーパーユーザー権限をサブスクライバーから削除します。

サブスクライバーでの実行例：

```
postgres=# ALTER ROLE alice NOSUPERUSER;
ALTER ROLE
```

パブリッシャーのテーブル（bob_table）にデータを挿入します。

パブリッシャーでの実行例：

```
postgres=# INSERT INTO bob_table VALUES (2);
INSERT 0 1
postgres=# SELECT * FROM bob_table;
 i
---
 1
 2
(2 rows)
```

データがサブスクライバーに反映されていないことを確認し、ログに permission denied というエラーが出力されていることを確認します。

サブスクライバーでの実行例：

```
postgres=# SELECT * FROM bob_table;
 i
---
 1
(1 row)
```


サーバーログの例：

```
2023-03-27 06:21:43.759 EDT [4746] ERROR:  permission denied for table bob_table
2023-03-27 06:21:43.759 EDT [4746] CONTEXT:  processing remote data for replication origin "pg_16408" during message type
"INSERT" for replication target relation "public.bob_table" in transaction 759, finished at 0/1581580
2023-03-27 06:21:43.760 EDT [6690] LOG:  background worker "logical replication worker" (PID 4746) exited with exit code 1
```

このエラーは、ロール 「alice」（サブスクリプションの所有者）にスーパーユーザー権限がなくなり、bob_table テーブルへの書き込み権限を持たなくなったために発生します。

今後に向けて

今後の計画では、非スーパーユーザーのロールによってサブスクリプションを作成し、そのロールの権限内で論理レプリケーションを実行できるようにします。これにより、PostgreSQL ユーザーはスーパーユーザーのロールを避けられるようになり、より柔軟に対応できます。すべての論理レプリケーションがスーパーユーザーによって実行される場合、不正なユーザーがそのテーブルの 1 つにトリガー関数を作成し、スーパーユーザーとして不正なコードを実行する可能性があります。

詳細情報

本ブログで紹介した本機能についての詳細は、以下のコミット情報をご覧ください。

- Respect permissions within logical replication.（PostgreSQL GIT repository のページへ）
  https://git.postgresql.org/gitweb/?p=postgresql.git;a=commitdiff;h=a2ab9c06ea15fbcb2bfde570986a06b37f52bcca

2023 年 5 月 26 日