

論理レプリケーションにおける列リスト

PostgreSQL 15 でコミットされた機能の紹介

技術者 Blog

Vigneshwaran C

FUJITSU Limited

Software Open Innovation Business Unit Data Management Division Software Lead Developer

PostgreSQL Contributor

はじめに

通常、論理レプリケーションのパブリッシュとサブスクライブモデルを使用する場合、パブリッシュされたテーブルのすべてのデータ変更が適切なサブスクライバーにレプリケートされます。新しい列リスト機能を利用すると、指定した列のデータのみがレプリケートされるように制限できます。

機能の概要

列リストは、CREATE PUBLICATION 構文において、テーブルごとにオプションで指定します。列リストはテーブル名の後に、括弧 () で囲んで指定します。各パブリケーションには、0 個以上の列リストを持つことができます。

列リストが指定されると、指定した列のデータのみがレプリケートされます。列リストに書く順序は関係ありません。

列リストが指定されない場合、後で追加された列も含めて、テーブルのすべての列は、このパブリケーションを通じてレプリケートされます。これは、すべての列を指定する列リストが、列リストが無い場合と全く同じではないことを意味します。例えば、テーブルに列が追加された場合、(ALTER SUBSCRIPTION...REFRESH PUBLICATION を実行した後) に列リストがあれば、その列リストに書かれた名前の列だけがレプリケートされ続けます。

なお、列リストは、TRUNCATE コマンドには影響しません。

CREATE / ALTER PUBLICATION 構文の改良

次に、簡略化した CREATE / ALTER PUBLICATION 構文を示します。PostgreSQL 15 で追加された新しい列リスト句を強調表示しています。

```
CREATE PUBLICATION <pub-name> FOR TABLE <table-name>[(column-list)]
ALTER PUBLICATION <pub-name> ADD TABLE <table-name>[(column-list)]
ALTER PUBLICATION <pub-name> SET TABLE <table-name>[(column-list)]

<column-list> ::= <column-list-item> [, <column-list>]
<column-list-item> ::= <column-name>
```

列リスト句

この句では、列リストを指定できます。

例 1

```
CREATE PUBLICATION salary_information FOR TABLE employee (user_id, name, salary);
```

UPDATE または DELETE 操作をパブリッシュするには、列リストに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。また、テーブルに REPLICA IDENTITY FULL が設定されている場合、列リストは指定できません (UPDATE または DELETE

の操作でパブリケーションのエラーが発生します)。パブリケーションが INSERT の操作のみをパブリッシュする場合、列リストにはテーブル内の任意の列を指定できます。

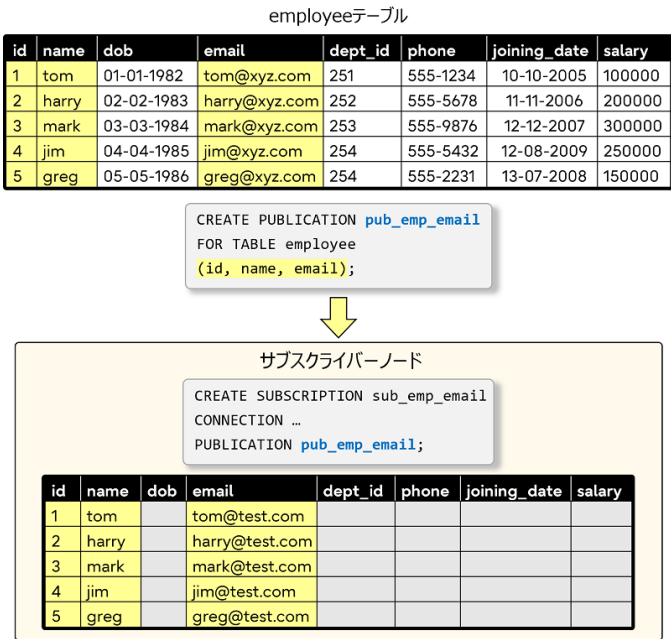
列リストの適用

列リストは、データの変更をパブリッシュするかどうかを決定する前に適用されます。パブリッシャーは列リストで指定されていない列のレプリケートをスキップします。

注意 サブスクリプションは、同じテーブルが異なる列リストでパブリッシュされた複数のパブリケーションにサブスクライブできません。例えば、employee テーブルの id、name、salary カラムをパブリッシュするパブリケーション：pub1 と、employee テーブルの id および email カラムをパブリッシュする別のパブリケーション：pub2 がある場合、列リストが異なるため、pub1 および pub2 のパブリケーションをサブスクライブするサブスクリプションを作成できません。

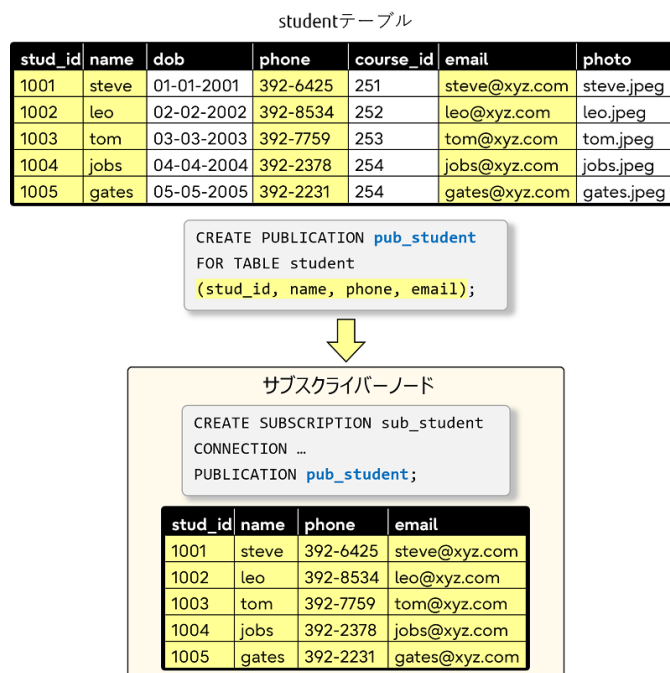
例 2：パブリケーション側と同一定義のテーブルにレプリケートする

この例では、サブスクライバーでテーブルを作成する際、レプリケートしない列（パブリッシャーの列リストに含まれない列）は NULL が設定されるため、NULL 値を許容する必要があります。



上記から、dob, dept_id, phone, joining_date, salary の列がフィルタリング（除外）されていることがわかります。したがって、上記のパブリケーションにより、これらの列のデータはレプリケートされません。

例 3 : パブリケーションに指定された列リストと同一の列のみを持つテーブルにレプリケートする



サブスクライバーノードで、パブリッシャーノードの student テーブルにあった列のサブセットのみ (stud_id、name、phone、email) を必要とする student テーブルを作成します。サブスクライバーノードの student テーブルには、dob、course_id、photo の列が無いことがわかります。

テーブルの初期同期

ユーザーが CREATE SUBSCRIPTION 構文でパラメーター「copy_data = true (デフォルト)」を指定すると、列リストに含まれる列のみがサブスクライバーにレプリケートされます。

psql コマンドの拡張

この新機能の一環として、PostgreSQL 15 の psql コマンドについても列リストに関する有用な情報を提供するように拡張されました。

テーブルを表示するメタコマンド (\d) では、そのテーブルが所属するパブリケーションに指定された列リストが表示されるようになりました。パブリケーションを一覧表示するメタコマンド (\dRp+) では、そのテーブルに指定された列リストが表示されるようになりました。

例 4

```
postgres=# CREATE TABLE employee (emp_no INTEGER NOT NULL, name VARCHAR, phone VARCHAR, email VARCHAR, address VARCHAR, dept_no
INTEGER, birthday DATE, salary INTEGER);
CREATE TABLE
postgres=# CREATE TABLE department (dept_no INTEGER NOT NULL, dept_name VARCHAR, mgr_no INT, location VARCHAR);
CREATE TABLE
postgres=# CREATE PUBLICATION pub_emp FOR TABLE employee;
CREATE PUBLICATION
-- Publication can be altered to include another table with specified -- columns.
postgres=# ALTER PUBLICATION pub_emp ADD TABLE department(dept_no, dept_name);
```

ALTER PUBLICATION

postgres=# CREATE PUBLICATION pub_emp_birthday FOR TABLE employee(emp_no, birthday);

CREATE PUBLICATION

postgres=# CREATE PUBLICATION pub_emp_email FOR TABLE employee(emp_no, email);

CREATE PUBLICATION

postgres=# \d employee

Table "public.employee"

Column	Type	Collation	Nullable	Default
emp_no	integer		not null	
name	character varying			
phone	character varying			
email	character varying			
address	character varying			
dept_no	integer			
birthday	date			
salary	integer			

Publications:

"pub_emp"

"pub_emp_birthday" (emp_no, birthday)

"pub_emp_email" (emp_no, email)

postgres=# \dRp+

Publication pub_emp

Owner	All tables	Inserts	Updates	Deletes	Truncates	Via root
vignesh	f	t	t	t	t	f

Tables:

"public.department" (dept_no, dept_name)

"public.employee"

Publication pub_emp_birthday

Owner	All tables	Inserts	Updates	Deletes	Truncates	Via root
vignesh	f	t	t	t	t	f

Tables:

"public.employee" (emp_no, birthday)

Publication pub_emp_email

Owner	All tables	Inserts	Updates	Deletes	Truncates	Via root
vignesh	f	t	t	t	t	f

Tables:

"public.employee" (emp_no, email)

postgres=#

効果

論理レプリケーションの列リストを定義することで、以下のような効果があります。

例 5 : パフォーマンス向上

大規模なデータテーブルの小さなサブセットのみをレプリケート（イメージをレプリケートせず）することで、ネットワークトラフィックを削減し、パフォーマンスを向上させる

```
CREATE PUBLICATION pub_data_staff_email
FOR TABLE staff(name, email, dept_id, salary);
```

例 6 : 不要データの削減

サブスクライバーノードに関連するデータのみを提供する

```
CREATE PUBLICATION pub_data_city_population
FOR TABLE city(name, population);
```

例 7 : 情報漏洩対策

クレジットカード番号などの機密情報を非表示（レプリケートしない）にすることで、セキュリティの一形態として動作させる

```
CREATE PUBLICATION pub_data_customer_investment FOR TABLE customer_account(name, phone, investment);
```

今後に向けて

列リストの追加は、PostgreSQL 15 の論理レプリケーションの主要な機能です。この機能により、ユーザーのツールボックスにまた 1 つ便利な機能が増え、より洗練され、カスタマイズされた論理レプリケーション・ソリューションを作成できるようになります。今後も、富士通の OSS チームは、PostgreSQL の論理レプリケーションの機能強化と機能追加を支援していきます。

詳細情報

本ブログで解説した「列リスト」についての詳細は、PostgreSQL 文書または GitHub に投稿したコミット情報をご覧ください。

- 列リストを記述するための CREATE PUBLICATION について
 - PostgreSQL: Documentation: 15: CREATE PUBLICATION（PostgreSQL オフィシャルのページへ）
<https://www.postgresql.org/docs/15/sql-createpublication.html>
- 「レプリカアイデンティティ」について
 - PostgreSQL: Documentation: 15: 31.1. Publication（PostgreSQL オフィシャルのページへ）
<https://www.postgresql.org/docs/15/logical-replication-publication.html>
 - PostgreSQL: Documentation: 15: ALTER TABLE（PostgreSQL オフィシャルのページへ）
<https://www.postgresql.org/docs/15/sql-altertable.html#SQL-ALTERTABLE-REPLICA-IDENTITY>
- 新しい列リスト機能を追加するための GitHub ソースコードについて
 - Allow specifying column lists for logical replication（GitHub のページへ）
<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commitdiff;h=923def9a533a7d986acfb524139d8b9e5466d0a5>

2022 年 11 月 4 日