NAME

CREATE_PUBLICATION − define a new publication

SYNOPSIS

```
CREATE PUBLICATION name
  [ FOR ALL TABLES
    | FOR publication_object [, ... ] ]
  [ WITH ( publication_parameter [= value] [, ... ] ) ]
```

where *publication_object* is one of:

```
  TABLE table_and_columns [, ... ]
  TABLES IN SCHEMA { schema_name | CURRENT_SCHEMA } [, ... ]
```

and *table_and_columns* is:

```
  [ ONLY ] table_name [ * ] [ ( column_name [, ... ] ) ] [ WHERE ( expression ) ]
```

DESCRIPTION

CREATE PUBLICATION adds a new publication into the current database. The publication name must be distinct from the name of any existing publication in the current database.

A publication is essentially a group of tables whose data changes are intended to be replicated through logical replication. See Section 29.1 for details about how publications fit into the logical replication setup.

PARAMETERS

name

The name of the new publication.

FOR TABLE

Specifies a list of tables to add to the publication. If ONLY is specified before the table name, only that table is added to the publication. If ONLY is not specified, the table and all its descendant tables (if any) are added. Optionally, * can be specified after the table name to explicitly indicate that descendant tables are included. This does not apply to a partitioned table, however. The partitions of a partitioned table are always implicitly considered part of the publication, so they are never explicitly added to the publication.

If the optional WHERE clause is specified, it defines a row filter expression. Rows for which the *expression* evaluates to false or null will not be published. Note that parentheses are required around the expression. It has no effect on TRUNCATE commands.

When a column list is specified, only the named columns are replicated. The column list can contain stored generated columns as well. If the column list is omitted, the publication will replicate all non−generated columns (including any added in the future) by default. Stored generated columns can also be replicated if publish_generated_columns is set to stored. Specifying a column list has no effect on TRUNCATE commands. See Section 29.5 for details about column lists.

Only persistent base tables and partitioned tables can be part of a publication. Temporary tables, unlogged tables, foreign tables, materialized views, and regular views cannot be part of a publication.

Specifying a column list when the publication also publishes FOR TABLES IN SCHEMA is not supported.

When a partitioned table is added to a publication, all of its existing and future partitions are implicitly considered to be part of the publication. So, even operations that are performed directly on a partition are also published via publications that its ancestors are part of.

FOR ALL TABLES

Marks the publication as one that replicates changes for all tables in the database, including tables created in the future.

FOR TABLES IN SCHEMA

Marks the publication as one that replicates changes for all tables in the specified list of schemas, including tables created in the future.

Specifying a schema when the publication also publishes a table with a column list is not supported.

Only persistent base tables and partitioned tables present in the schema will be included as part of the publication. Temporary tables, unlogged tables, foreign tables, materialized views, and regular views from the schema will not be part of the publication.

When a partitioned table is published via a schema−level publication, all of its existing and future partitions are implicitly considered to be part of the publication, regardless of whether they are from the publication schema or not. So, even operations that are performed directly on a partition are also published via publications that its ancestors are part of.

WITH (*publication_parameter* [= *value*] [, ...])

This clause specifies optional parameters for a publication. The following parameters are supported:

publish (string)

This parameter determines which DML operations will be published by the new publication to the subscribers. The value is a comma−separated list of operations. The allowed operations are insert, update, delete, and truncate. The default is to publish all actions, and so the default value for this option is 'insert, update, delete, truncate'.

This parameter only affects DML operations. In particular, the initial data synchronization (see Section 29.9.1) for logical replication does not take this parameter into account when copying existing table data.

publish_generated_columns (enum)

Specifies whether the generated columns present in the tables associated with the publication should be replicated. Possible values are none and stored.

The default is none meaning the generated columns present in the tables associated with publication will not be replicated.

If set to stored, the stored generated columns present in the tables associated with publication will be replicated.

Note

If the subscriber is from a release prior to 18, then initial table synchronization won't copy generated columns even if the parameter publish_generated_columns is stored in the publisher.

See Section 29.6 for more details about logical replication of generated columns.

publish_via_partition_root (boolean)

This parameter controls how changes to a partitioned table (or any of its partitions) are published. When set to true, changes are published using the identity and schema of the root partitioned table. When set to false (the default), changes are published using the identity and schema of the individual partitions where the changes actually occurred. Enabling this option allows the changes to be replicated into a non−partitioned table or into a partitioned table whose partition structure differs from that of the publisher.

There can be a case where a subscription combines multiple publications. If a partitioned table is published by any subscribed publications which set publish_via_partition_root = true, changes on this partitioned table (or on its partitions) will be published using the identity and schema of this

partitioned table rather than that of the individual partitions.

This parameter also affects how row filters and column lists are chosen for partitions; see below for details.

If this is enabled, TRUNCATE operations performed directly on partitions are not replicated.

When specifying a parameter of type boolean, the = *value* part can be omitted, which is equivalent to specifying TRUE.

NOTES

If FOR TABLE, FOR ALL TABLES or FOR TABLES IN SCHEMA are not specified, then the publication starts out with an empty set of tables. That is useful if tables or schemas are to be added later.

The creation of a publication does not start replication. It only defines a grouping and filtering logic for future subscribers.

To create a publication, the invoking user must have the CREATE privilege for the current database. (Of course, superusers bypass this check.)

To add a table to a publication, the invoking user must have ownership rights on the table. The **FOR ALL TABLES** and **FOR TABLES IN SCHEMA** clauses require the invoking user to be a superuser.

The tables added to a publication that publishes **UPDATE** and/or **DELETE** operations must have REPLICA IDENTITY defined. Otherwise those operations will be disallowed on those tables.

Any column list must include the REPLICA IDENTITY columns in order for **UPDATE** or **DELETE** operations to be published. There are no column list restrictions if the publication publishes only **INSERT** operations.

A row filter expression (i.e., the WHERE clause) must contain only columns that are covered by the REPLICA IDENTITY, in order for **UPDATE** and **DELETE** operations to be published. For publication of **INSERT** operations, any column may be used in the WHERE expression. The row filter allows simple expressions that don't have user−defined functions, user−defined operators, user−defined types, user−defined collations, non−immutable built−in functions, or references to system columns.

The generated columns that are part of REPLICA IDENTITY must be published explicitly either by listing them in the column list or by enabling the publish_generated_columns option, in order for **UPDATE** and **DELETE** operations to be published.

The row filter on a table becomes redundant if FOR TABLES IN SCHEMA is specified and the table belongs to the referred schema.

For published partitioned tables, the row filter for each partition is taken from the published partitioned table if the publication parameter publish_via_partition_root is true, or from the partition itself if it is false (the default). See Section 29.4 for details about row filters. Similarly, for published partitioned tables, the column list for each partition is taken from the published partitioned table if the publication parameter publish_via_partition_root is true, or from the partition itself if it is false.

For an **INSERT ... ON CONFLICT** command, the publication will publish the operation that results from the command. Depending on the outcome, it may be published as either **INSERT** or **UPDATE**, or it may not be published at all.

For a **MERGE** command, the publication will publish an **INSERT**, **UPDATE**, or **DELETE** for each row inserted, updated, or deleted.

ATTACHing a table into a partition tree whose root is published using a publication with publish_via_partition_root set to true does not result in the table's existing contents being replicated.

COPY ... FROM commands are published as **INSERT** operations.

DDL operations are not published.

The WHERE clause expression is executed with the role used for the replication connection.

EXAMPLES

Create a publication that publishes all changes in two tables:

```
CREATE PUBLICATION mypublication FOR TABLE users, departments;
```

Create a publication that publishes all changes from active departments:

```
CREATE PUBLICATION active_departments FOR TABLE departments WHERE (active IS TRUE);
```

Create a publication that publishes all changes in all tables:

```
CREATE PUBLICATION alltables FOR ALL TABLES;
```

Create a publication that only publishes **INSERT** operations in one table:

```
CREATE PUBLICATION insert_only FOR TABLE mydata
  WITH (publish = 'insert');
```

Create a publication that publishes all changes for tables users, departments and all changes for all the tables present in the schema production:

```
CREATE PUBLICATION production_publication FOR TABLE users, departments, TABLES IN SCHEMA production;
```

Create a publication that publishes all changes for all the tables present in the schemas marketing and sales:

```
CREATE PUBLICATION sales_publication FOR TABLES IN SCHEMA marketing, sales;
```

Create a publication that publishes all changes for table users, but replicates only columns user_id and firstname:

```
CREATE PUBLICATION users_filtered FOR TABLE users (user_id, firstname);
```

COMPATIBILITY

CREATE PUBLICATION is a PostgreSQL extension.

SEE ALSO

ALTER PUBLICATION (**ALTER_PUBLICATION**(7)), DROP PUBLICATION (**DROP_PUBLICATION**(7)), CREATE SUBSCRIPTION (**CREATE_SUBSCRIPTION**(7)), ALTER SUBSCRIPTION (**ALTER_SUBSCRIPTION**(7))