NAME

ALTER_SUBSCRIPTION − change the definition of a subscription

SYNOPSIS

ALTER SUBSCRIPTION *name* CONNECTION '*conninfo*'
ALTER SUBSCRIPTION *name* SET PUBLICATION *publication_name* [, ...] [WITH (*publication_option* [= *value*] [, ...])
ALTER SUBSCRIPTION *name* ADD PUBLICATION *publication_name* [, ...] [WITH (*publication_option* [= *value*] [, ...])
ALTER SUBSCRIPTION *name* DROP PUBLICATION *publication_name* [, ...] [WITH (*publication_option* [= *value*] [, ...]
ALTER SUBSCRIPTION *name* REFRESH PUBLICATION [WITH (*refresh_option* [= *value*] [, ...])]
ALTER SUBSCRIPTION *name* ENABLE
ALTER SUBSCRIPTION *name* DISABLE
ALTER SUBSCRIPTION *name* SET (*subscription_parameter* [= *value*] [, ...])
ALTER SUBSCRIPTION *name* SKIP (*skip_option* = *value*)
ALTER SUBSCRIPTION *name* OWNER TO { *new_owner* | CURRENT_ROLE | CURRENT_USER | SESSION_USER }
ALTER SUBSCRIPTION *name* RENAME TO *new_name*

DESCRIPTION

ALTER SUBSCRIPTION can change most of the subscription properties that can be specified in CREATE SUBSCRIPTION (**CREATE_SUBSCRIPTION**(7)).

You must own the subscription to use **ALTER SUBSCRIPTION**. To rename a subscription or alter the owner, you must have CREATE permission on the database. In addition, to alter the owner, you must be able to SET ROLE to the new owning role. If the subscription has password_required=false, only superusers can modify it.

When refreshing a publication we remove the relations that are no longer part of the publication and we also remove the table synchronization slots if there are any. It is necessary to remove these slots so that the resources allocated for the subscription on the remote host are released. If due to network breakdown or some other error, PostgreSQL is unable to remove the slots, an error will be reported. To proceed in this situation, the user either needs to retry the operation or disassociate the slot from the subscription and drop the subscription as explained in DROP SUBSCRIPTION (**DROP_SUBSCRIPTION**(7)).

Commands **ALTER SUBSCRIPTION ... REFRESH PUBLICATION**, **ALTER SUBSCRIPTION ... {SET|ADD|DROP} PUBLICATION ...** with refresh option as true, **ALTER SUBSCRIPTION ... SET (failover = true|false)** and **ALTER SUBSCRIPTION ... SET (two_phase = false)** cannot be executed inside a transaction block.

Commands **ALTER SUBSCRIPTION ... REFRESH PUBLICATION** and **ALTER SUBSCRIPTION ... {SET|ADD|DROP} PUBLICATION ...** with refresh option as true also cannot be executed when the subscription has two_phase commit enabled, unless copy_data is false. See column subtwophasestate of pg_subscription to know the actual two−phase state.

PARAMETERS

name
The name of a subscription whose properties are to be altered.

CONNECTION '*conninfo*'
This clause replaces the connection string originally set by CREATE SUBSCRIPTION (**CREATE_SUBSCRIPTION**(7)). See there for more information.

SET PUBLICATION *publication_name*
ADD PUBLICATION *publication_name*
DROP PUBLICATION *publication_name*
These forms change the list of subscribed publications. SET replaces the entire list of publications with a new list, ADD adds additional publications to the list of publications, and DROP removes the publications from the list of publications. We allow non−existent publications to be specified in ADD and SET variants so that users can add those later. See CREATE SUBSCRIPTION (**CREATE_SUBSCRIPTION**(7)) for more information. By default, this command will also act like REFRESH PUBLICATION.

publication_option specifies additional options for this operation. The supported options are:

refresh (boolean)
When false, the command will not try to refresh table information. REFRESH PUBLICATION should then be executed separately. The default is true.

Additionally, the options described under REFRESH PUBLICATION may be specified, to control the implicit refresh operation.

REFRESH PUBLICATION
Fetch missing table information from publisher. This will start replication of tables that were added to the subscribed−to publications since **CREATE SUBSCRIPTION** or the last invocation of **REFRESH PUBLICATION**.

refresh_option specifies additional options for the refresh operation. The supported options are:

copy_data (boolean)
Specifies whether to copy pre−existing data in the publications that are being subscribed to when the replication starts. The default is true.

Previously subscribed tables are not copied, even if a table's row filter WHERE clause has since been modified.

See Notes for details of how copy_data = true can interact with the origin parameter.

See the binary parameter of **CREATE SUBSCRIPTION** for details about copying pre−existing data in binary format.

ENABLE
Enables a previously disabled subscription, starting the logical replication worker at the end of the transaction.

DISABLE
Disables a running subscription, stopping the logical replication worker at the end of the transaction.

SET (*subscription_parameter* [= *value*] [, ...])
This clause alters parameters originally set by CREATE SUBSCRIPTION (**CREATE_SUBSCRIPTION**(7)). See there for more information. The parameters that can be altered are slot_name, synchronous_commit, binary, streaming, disable_on_error, password_required, run_as_owner, origin, failover, and two_phase. Only a superuser can set password_required = false.

When altering the slot_name, the failover and two_phase property values of the named slot may differ from the counterpart failover and two_phase parameters specified in the subscription. When creating the slot, ensure the slot properties failover and two_phase match their counterpart parameters of the subscription. Otherwise, the slot on the publisher may behave differently from what these subscription options say: for example, the slot on the publisher could either be synced to the standbys even when the subscription's failover option is disabled or could be disabled for sync even when the subscription's failover option is enabled.

The failover and two_phase parameters can only be altered when the subscription is disabled.

When altering two_phase from true to false, the backend process reports an error if any prepared transactions done by the logical replication worker (from when two_phase parameter was still true) are found. You can resolve prepared transactions on the publisher node, or manually roll them back on the subscriber, and then try again. The transactions prepared by logical replication worker corresponding to a particular subscription have the following pattern: “pg_gid_%u_%u” (parameters: subscription *oid*, remote transaction id *xid*). To resolve such transactions manually, you need to roll back all the prepared transactions with corresponding subscription IDs in their names. Applications can check

pg_prepared_xacts to find the required prepared transactions. After the two_phase option is changed from true to false, the publisher will replicate the transactions again when they are committed.

SKIP (*skip_option* = *value*)
Skips applying all changes of the remote transaction. If incoming data violates any constraints, logical replication will stop until it is resolved. By using the **ALTER SUBSCRIPTION ... SKIP** command, the logical replication worker skips all data modification changes within the transaction. This option has no effect on the transactions that are already prepared by enabling two_phase on the subscriber. After the logical replication worker successfully skips the transaction or finishes a transaction, the LSN (stored in pg_subscription.subskiplsn) is cleared. See Section 29.7 for the details of logical replication conflicts.

skip_option specifies options for this operation. The supported option is:

lsn (pg_lsn)
Specifies the finish LSN of the remote transaction whose changes are to be skipped by the logical replication worker. The finish LSN is the LSN at which the transaction is either committed or prepared. Skipping individual subtransactions is not supported. Setting NONE resets the LSN.

new_owner
The user name of the new owner of the subscription.

new_name
The new name for the subscription.

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

EXAMPLES

Change the publication subscribed by a subscription to insert_only:

```
ALTER SUBSCRIPTION mysub SET PUBLICATION insert_only;
```

Disable (stop) the subscription:

```
ALTER SUBSCRIPTION mysub DISABLE;
```

COMPATIBILITY

ALTER SUBSCRIPTION is a PostgreSQL extension.

SEE ALSO

CREATE SUBSCRIPTION (**CREATE_SUBSCRIPTION**(7)), DROP SUBSCRIPTION (**DROP_SUBSCRIPTION**(7)), CREATE PUBLICATION (**CREATE_PUBLICATION**(7)), ALTER PUBLICATION (**ALTER_PUBLICATION**(7))