## NAME

ALTER_VIEW − change the definition of a view

## SYNOPSIS

ALTER VIEW [ IF EXISTS ] *name* ALTER [ COLUMN ] *column_name* SET DEFAULT *expression*
ALTER VIEW [ IF EXISTS ] *name* ALTER [ COLUMN ] *column_name* DROP DEFAULT
ALTER VIEW [ IF EXISTS ] *name* OWNER TO { *new_owner* | CURRENT_ROLE | CURRENT_USER | SESSION_USER
ALTER VIEW [ IF EXISTS ] *name* RENAME [ COLUMN ] *column_name* TO *new_column_name*
ALTER VIEW [ IF EXISTS ] *name* RENAME TO *new_name*
ALTER VIEW [ IF EXISTS ] *name* SET SCHEMA *new_schema*
ALTER VIEW [ IF EXISTS ] *name* SET ( *view_option_name* [= *view_option_value*] [, ... ] )
ALTER VIEW [ IF EXISTS ] *name* RESET ( *view_option_name* [, ... ] )

## DESCRIPTION

**ALTER VIEW** changes various auxiliary properties of a view. (If you want to modify the view's defining query, use **CREATE OR REPLACE VIEW**.)

You must own the view to use **ALTER VIEW**. To change a view's schema, you must also have CREATE privilege on the new schema. To alter the owner, you must be able to SET ROLE to the new owning role, and that role must have CREATE privilege on the view's schema. (These restrictions enforce that altering the owner doesn't do anything you couldn't do by dropping and recreating the view. However, a superuser can alter ownership of any view anyway.)

## PARAMETERS

*name*
The name (optionally schema−qualified) of an existing view.

*column_name*
Name of an existing column.

*new_column_name*
New name for an existing column.

IF EXISTS
Do not throw an error if the view does not exist. A notice is issued in this case.

SET/DROP DEFAULT
These forms set or remove the default value for a column. A view column's default value is substituted into any **INSERT** or **UPDATE** command whose target is the view, before applying any rules or triggers for the view. The view's default will therefore take precedence over any default values from underlying relations.

*new_owner*
The user name of the new owner of the view.

*new_name*
The new name for the view.

*new_schema*
The new schema for the view.

SET ( *view_option_name* [= *view_option_value*] [, ... ] )
RESET ( *view_option_name* [, ... ] )
Sets or resets a view option. Currently supported options are:

check_option (enum)
Changes the check option of the view. The value must be local or cascaded.

security_barrier (boolean)
Changes the security−barrier property of the view. The value must be a Boolean value, such as true or false.

security_invoker (boolean)

Changes the security−invoker property of the view. The value must be a Boolean value, such as true or false.

## NOTES

For historical reasons, **ALTER TABLE** can be used with views too; but the only variants of **ALTER TABLE** that are allowed with views are equivalent to the ones shown above.

## EXAMPLES

To rename the view foo to bar:

```
ALTER VIEW foo RENAME TO bar;
```

To attach a default column value to an updatable view:

```
CREATE TABLE base_table (id int, ts timestamptz);
CREATE VIEW a_view AS SELECT * FROM base_table;
ALTER VIEW a_view ALTER COLUMN ts SET DEFAULT now();
INSERT INTO base_table(id) VALUES(1);  −− ts will receive a NULL
INSERT INTO a_view(id) VALUES(2);  −− ts will receive the current time
```

## COMPATIBILITY

**ALTER VIEW** is a PostgreSQL extension of the SQL standard.

## SEE ALSO

CREATE VIEW (**CREATE_VIEW**(7)), DROP VIEW (**DROP_VIEW**(7))