NAME

ALTER_SEQUENCE − change the definition of a sequence generator

SYNOPSIS

ALTER SEQUENCE [IF EXISTS] *name*
[AS *data_type*]
[INCREMENT [BY] *increment*]
[MINVALUE *minvalue* | NO MINVALUE] [MAXVALUE *maxvalue* | NO MAXVALUE]
[[NO] CYCLE]
[START [WITH] *start*]
[RESTART [[WITH] *restart*]]
[CACHE *cache*]
[OWNED BY { *table_name*.*column_name* | NONE }]
ALTER SEQUENCE [IF EXISTS] *name* SET { LOGGED | UNLOGGED }
ALTER SEQUENCE [IF EXISTS] *name* OWNER TO { *new_owner* | CURRENT_ROLE | CURRENT_USER | SESSION_U
ALTER SEQUENCE [IF EXISTS] *name* RENAME TO *new_name*
ALTER SEQUENCE [IF EXISTS] *name* SET SCHEMA *new_schema*

DESCRIPTION

ALTER SEQUENCE changes the parameters of an existing sequence generator. Any parameters not specifically set in the **ALTER SEQUENCE** command retain their prior settings.

You must own the sequence to use **ALTER SEQUENCE**. To change a sequence'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 sequence's schema. (These restrictions enforce that altering the owner doesn't do anything you couldn't do by dropping and recreating the sequence. However, a superuser can alter ownership of any sequence anyway.)

PARAMETERS

name
The name (optionally schema−qualified) of a sequence to be altered.

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

data_type
The optional clause AS *data_type* changes the data type of the sequence. Valid types are smallint, integer, and bigint.

Changing the data type automatically changes the minimum and maximum values of the sequence if and only if the previous minimum and maximum values were the minimum or maximum value of the old data type (in other words, if the sequence had been created using NO MINVALUE or NO MAXVALUE, implicitly or explicitly). Otherwise, the minimum and maximum values are preserved, unless new values are given as part of the same command. If the minimum and maximum values do not fit into the new data type, an error will be generated.

increment
The clause INCREMENT BY *increment* is optional. A positive value will make an ascending sequence, a negative one a descending sequence. If unspecified, the old increment value will be maintained.

minvalue
NO MINVALUE
The optional clause MINVALUE *minvalue* determines the minimum value a sequence can generate. If NO MINVALUE is specified, the defaults of 1 and the minimum value of the data type for ascending and descending sequences, respectively, will be used. If neither option is specified, the current minimum value will be maintained.

maxvalue
NO MAXVALUE

The optional clause MAXVALUE *maxvalue* determines the maximum value for the sequence. If NO MAXVALUE is specified, the defaults of the maximum value of the data type and −1 for ascending and descending sequences, respectively, will be used. If neither option is specified, the current maximum value will be maintained.

CYCLE
: The optional CYCLE key word can be used to enable the sequence to wrap around when the *maxvalue* or *minvalue* has been reached by an ascending or descending sequence respectively. If the limit is reached, the next number generated will be the *minvalue* or *maxvalue*, respectively.

NO CYCLE
: If the optional NO CYCLE key word is specified, any calls to **nextval** after the sequence has reached its maximum value will return an error. If neither CYCLE or NO CYCLE are specified, the old cycle behavior will be maintained.

start
: The optional clause START WITH *start* changes the recorded start value of the sequence. This has no effect on the *current* sequence value; it simply sets the value that future **ALTER SEQUENCE RESTART** commands will use.

restart
: The optional clause RESTART [WITH *restart*] changes the current value of the sequence. This is similar to calling the **setval** function with is_called = false: the specified value will be returned by the *next* call of **nextval**. Writing RESTART with no *restart* value is equivalent to supplying the start value that was recorded by **CREATE SEQUENCE** or last set by **ALTER SEQUENCE START WITH**.

  In contrast to a **setval** call, a RESTART operation on a sequence is transactional and blocks concurrent transactions from obtaining numbers from the same sequence. If that's not the desired mode of operation, **setval** should be used.

cache
: The clause CACHE *cache* enables sequence numbers to be preallocated and stored in memory for faster access. The minimum value is 1 (only one value can be generated at a time, i.e., no cache). If unspecified, the old cache value will be maintained.

SET { LOGGED | UNLOGGED }
: This form changes the sequence from unlogged to logged or vice−versa (see CREATE SEQUENCE (**CREATE_SEQUENCE**(7))). It cannot be applied to a temporary sequence.

OWNED BY *table_name.column_name*
OWNED BY NONE
: The OWNED BY option causes the sequence to be associated with a specific table column, such that if that column (or its whole table) is dropped, the sequence will be automatically dropped as well. If specified, this association replaces any previously specified association for the sequence. The specified table must have the same owner and be in the same schema as the sequence. Specifying OWNED BY NONE removes any existing association, making the sequence “free−standing”.

new_owner
: The user name of the new owner of the sequence.

new_name
: The new name for the sequence.

new_schema
: The new schema for the sequence.

NOTES

ALTER SEQUENCE will not immediately affect **nextval** results in backends, other than the current one, that have preallocated (cached) sequence values. They will use up all cached values prior to noticing the changed sequence generation parameters. The current backend will be affected immediately.

ALTER SEQUENCE does not affect the **currval** status for the sequence. (Before PostgreSQL 8.3, it

sometimes did.)

ALTER SEQUENCE blocks concurrent **nextval**, **currval**, **lastval**, and **setval** calls.

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

EXAMPLES

Restart a sequence called serial, at 105:

```
ALTER SEQUENCE serial RESTART WITH 105;
```

COMPATIBILITY

ALTER SEQUENCE conforms to the SQL standard, except for the AS, START WITH, OWNED BY, OWNER TO, RENAME TO, and SET SCHEMA clauses, which are PostgreSQL extensions.

SEE ALSO

CREATE SEQUENCE (**CREATE_SEQUENCE**(7)), DROP SEQUENCE (**DROP_SEQUENCE**(7))