alter_index

- Description
- Parameters
- Notes
- Examples
- Compatibility
- See Also

ALTER INDEX

ALTER INDEX

7

SQL - Language Statements

ALTER INDEX

change the definition of an index

ALTER INDEX [IF EXISTS]

name

RENAME TO

new_name

ALTER INDEX [IF EXISTS]

name

SET TABLESPACE

tablespace_name

ALTER INDEX

name

ATTACH PARTITION

index_name

ALTER INDEX

name

[NO] DEPENDS ON EXTENSION

extension_name

ALTER INDEX [IF EXISTS]

name

SET (

storage_parameter

[=

value

] [, ...]) ALTER INDEX [IF EXISTS]

name

RESET (

storage_parameter

[, ...]) ALTER INDEX [IF EXISTS]

name

ALTER [COLUMN]

column_number

SET STATISTICS

integer

ALTER INDEX ALL IN TABLESPACE

name

[OWNED BY

role_name

[, ...]] SET TABLESPACE

new_tablespace

[NOWAIT]

Description

`ALTER INDEX` changes the definition of an existing index. There are several subforms described below. Note that the lock level required may differ for each subform. An `ACCESS EXCLUSIVE` lock is held unless explicitly noted. When multiple subcommands are listed, the lock held will be the strictest one required from any subcommand.

`RENAME`
The `RENAME` form changes the name of the index. If the index is associated with a table constraint (either `UNIQUE`, `PRIMARY KEY`, or `EXCLUDE`), the constraint is renamed as well. There is no effect on the stored data.

Renaming an index acquires a `SHARE UPDATE EXCLUSIVE` lock.

`SET TABLESPACE`
This form changes the index's tablespace to the specified tablespace and moves the data file(s) associated with the index to the new tablespace. To change the tablespace of an index, you must own the index and have `CREATE` privilege on the new tablespace. All indexes in the current database in a tablespace can be moved by using the `ALL IN TABLESPACE` form, which will lock all indexes to be moved and then move each one. This form also supports `OWNED BY`, which will only move indexes owned by the roles specified. If the `NOWAIT` option is specified then the command will fail if it is unable to acquire all of the locks required immediately. Note that system catalogs will not be moved by this command, use `ALTER DATABASE` or explicit `ALTER INDEX` invocations instead if desired. See also `CREATE TABLESPACE`.

`ATTACH PARTITION index_name`
Causes the named index (possibly schema-qualified) to become attached to the altered index. The named index must be on a partition of the table containing the index being altered, and have an equivalent definition. An attached index cannot be dropped by itself, and will automatically be dropped if its parent index is dropped.

If the named index is already attached to the altered index, the command will attempt to validate the parent index if the parent is currently invalid.

`DEPENDS ON EXTENSION extension_name`; `NO DEPENDS ON EXTENSION extension_name`
This form marks the index as dependent on the extension, or no longer dependent on that extension if `NO` is specified. An index that's marked as dependent on an extension is automatically dropped when the extension is dropped.

`SET ( storage_parameter [= value] [, ... ] )`
This form changes one or more index-method-specific storage parameters for the index. See `CREATE INDEX` for details on the available parameters. Note that the index contents will not be modified immediately by this command; depending on the parameter you might need to rebuild the index with `REINDEX` to get the desired effects.

`RESET ( storage_parameter [, ... ] )`
This form resets one or more index-method-specific storage parameters to their defaults. As with `SET`, a `REINDEX` might be needed to update the index entirely.

`ALTER [ COLUMN ] column_number SET STATISTICS integer`
This form sets the per-column statistics-gathering target for subsequent `ANALYZE` operations, though can be used only on index columns that are defined as an expression. Since expressions lack a unique name, we refer to them using the ordinal number of the index column. The target can be set in the range 0 to 10000; alternatively, set it to -1 to revert to using the system default statistics target (???). For more information on the use of statistics by the PostgreSQL query planner, refer to ???.

Parameters

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

<column_number>
The ordinal number refers to the ordinal (left-to-right) position of the index column.

<name>
The name (possibly schema-qualified) of an existing index to alter.

<new_name>
The new name for the index.

<tablespace_name>
The tablespace to which the index will be moved.

<extension_name>
The name of the extension that the index is to depend on.

<storage_parameter>
The name of an index-method-specific storage parameter.

<value>
The new value for an index-method-specific storage parameter. This might be a number or a word depending on the parameter.

Notes

These operations are also possible using `ALTER TABLE`. `ALTER INDEX` is in fact just an alias for the forms of `ALTER TABLE` that apply to indexes.

There was formerly an `ALTER INDEX OWNER` variant, but this is now ignored (with a warning). An index cannot have an owner different from its table's owner. Changing the table's owner automatically changes the index as well.

Changing any part of a system catalog index is not permitted.

Examples

To rename an existing index:

```
ALTER INDEX distributors RENAME TO suppliers;
```

To move an index to a different tablespace:

```
ALTER INDEX distributors SET TABLESPACE fasttablespace;
```

To change an index's fill factor (assuming that the index method supports it):

```
ALTER INDEX distributors SET (fillfactor = 75);
REINDEX INDEX distributors;
```

Set the statistics-gathering target for an expression index:

```
CREATE INDEX coord_idx ON measured (x, y, (z + t));
ALTER INDEX coord_idx ALTER COLUMN 3 SET STATISTICS 1000;
```

Compatibility

`ALTER INDEX` is a PostgreSQL extension.

See Also

□□□ #2
□ □□□ □□□ 2026-10-31 14:07:19 UTC
□ □□□ □□□ 2026-10-31 14:07:20 UTC