**ALTER TABLE**

**Name**

ALTER TABLE -- mudar a definição de uma tabela

**Sinopse**

ALTER TABLE [ ONLY ] ***name*** [ \* ]

***action*** [, ... ]

ALTER TABLE [ ONLY ] ***name*** [ \* ]

RENAME [ COLUMN ] ***column*** TO ***new\_column***

ALTER TABLE ***name***

RENAME TO ***new\_name***

ALTER TABLE ***name***

SET SCHEMA ***new\_schema***

where ***action*** is one of:

ADD [ COLUMN ] ***column*** ***data\_type*** [ COLLATE ***collation*** ] [ ***column\_constraint*** [ ... ] ]

DROP [ COLUMN ] [ IF EXISTS ] ***column*** [ RESTRICT | CASCADE ]

ALTER [ COLUMN ] ***column*** [ SET DATA ] TYPE ***data\_type*** [ COLLATE ***collation*** ] [ USING ***expression*** ]

ALTER [ COLUMN ] ***column*** SET DEFAULT ***expression***

ALTER [ COLUMN ] ***column*** DROP DEFAULT

ALTER [ COLUMN ] ***column*** { SET | DROP } NOT NULL

ALTER [ COLUMN ] ***column*** SET STATISTICS ***integer***

ALTER [ COLUMN ] ***column*** SET ( ***attribute\_option*** = ***value*** [, ... ] )

ALTER [ COLUMN ] ***column*** RESET ( ***attribute\_option*** [, ... ] )

ALTER [ COLUMN ] ***column*** SET STORAGE { PLAIN | EXTERNAL | EXTENDED | MAIN }

ADD ***table\_constraint*** [ NOT VALID ]

ADD ***table\_constraint\_using\_index***

VALIDATE CONSTRAINT ***constraint\_name***

DROP CONSTRAINT [ IF EXISTS ] ***constraint\_name*** [ RESTRICT | CASCADE ]

DISABLE TRIGGER [ ***trigger\_name*** | ALL | USER ]

ENABLE TRIGGER [ ***trigger\_name*** | ALL | USER ]

ENABLE REPLICA TRIGGER ***trigger\_name***

ENABLE ALWAYS TRIGGER ***trigger\_name***

DISABLE RULE ***rewrite\_rule\_name***

ENABLE RULE ***rewrite\_rule\_name***

ENABLE REPLICA RULE ***rewrite\_rule\_name***

ENABLE ALWAYS RULE ***rewrite\_rule\_name***

CLUSTER ON ***index\_name***

SET WITHOUT CLUSTER

SET WITH OIDS

SET WITHOUT OIDS

SET ( ***storage\_parameter*** = ***value*** [, ... ] )

RESET ( ***storage\_parameter*** [, ... ] )

INHERIT ***parent\_table***

NO INHERIT ***parent\_table***

OF ***type\_name***

NOT OF

OWNER TO ***new\_owner***

SET TABLESPACE ***new\_tablespace***

and ***table\_constraint\_using\_index*** is:

[ CONSTRAINT ***constraint\_name*** ]

{ UNIQUE | PRIMARY KEY } USING INDEX ***index\_name***

[ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]

**Description**

ALTER TABLE altera a definição de uma tabela existente. Existem várias sub-formas:

ADD COLUMN

Esta forma adiciona uma nova coluna à tabela, usando a mesma sintaxe do

[CREATE TABLE](http://www.postgresql.org/docs/9.1/static/sql-createtable.html).

DROP COLUMN [ IF EXISTS ]

Esta forma remove uma coluna de uma tabela. Os índices e as restrições da tabela que envolvem a coluna será automaticamente caiu bem. Você vai precisar de CASCADE se algum objeto fora da tabela depender da coluna, por exemplo, referências de chaves estrangeiras ou visões. Se ele existir é especificada ea coluna não existe, nenhum erro é lançada. Neste caso, um aviso é emitido em seu lugar.

SET DATA TYPE

Esta forma muda o tipo de uma coluna de uma tabela. Os índices e as restrições de tabela simples que envolvem a coluna será automaticamente convertido para usar o novo tipo de coluna por reanálise da expressão original fornecida. A cláusula COLLATE opcional especifica um agrupamento para a nova coluna, se omitido, o agrupamento é o padrão para o novo tipo de coluna. A cláusula opcional USING especifica como calcular o novo valor da coluna do antigo, se omitida, a conversão padrão é o mesmo que lançar uma atribuição do tipo de dados antigo para o novo. A cláusula USING deve ser fornecido, se não há nenhuma conversão implícita ou de atribuição do antigo para o novo tipo.

SET/DROP DEFAULT

Estas formas definem ou removem o valor padrão para uma coluna. Os valores padrão aplicam-se apenas aos comandos INSERT subseqüentes; eles não causam filas já na mesa para mudar. Padrões também podem ser criados para vistas, caso em que estão inseridos dentro do comando INSERT na visão antes do ON INSERT a regra é aplicada.

SET/DROP NOT NULL

These forms change whether a column is marked to allow null values or to reject null values. You can only use SET NOT NULL when the column contains no null values.

SET STATISTICS

Esta forma define a meta de elaboração de estatísticas por coluna para as operações subseqüentes analisar. O alvo pode ser definido na faixa de 0 a 10000; alternativamente, defina-o como -1 para voltar a usar o alvo estatísticas padrão do sistema (default\_statistics\_target). Para mais informações sobre o uso de estatísticas pelo planejador de consultas do PostgreSQL, consulte a Seção 14.2.

SET ( ***attribute\_option*** = ***value*** [, ... ] )  
RESET ( ***attribute\_option*** [, ... ] )

Esta forma define ou redefine opções por atributo. Atualmente, as opções por atributos só são definidos n\_distinct e n\_distinct\_inherited , que substituem o número de - valores distintos - estimativas feitas por operações subseqüentes analisar. n\_distinct afeta as estatísticas para a própria mesa, enquanto n\_distinct\_inherited afeta as estatísticas recolhidas para a tabela mais seus filhos à herança. Quando ajustado para um valor positivo, ANALISAR irá assumir que a coluna contém exatamente o número especificado de valores não nulos distintos. Quando ajustado para um valor negativo, o que deve ser maior do que ou igual a -1 , ANALISAR irão assumir que o número de valores não nulos distintos na coluna é linear no tamanho da tabela , a contagem exacta deve ser calculado multiplicando-se a Estima tamanho da tabela pelo valor absoluto do número dado. Por exemplo, um valor de -1 significa que todos os valores da coluna são distintas , enquanto que um valor de -0,5 implica que cada valor aparece duas vezes em média. Isto pode ser útil quando o tamanho da tabela de mudanças ao longo do tempo, já que a multiplicação do número de linhas na tabela não é executada até que o tempo de planejamento de consulta. Especifica um valor de 0 a reverter para estimar o número de valores distintos normalmente. Para mais informações sobre o uso de estatísticas pelo planejador de consultas do PostgreSQL , consulte a Seção 14.2.

SET STORAGE

Esta forma define o modo de armazenamento para uma coluna. Controla se a coluna é mantida em linha ou em uma tabela TOAST secundário, e se os dados devem ser comprimidos ou não. Deve ser utilizado PLAIN para valores de comprimento fixo, como integer e está em linha, sem compressão. MAIN é para inline, dados compressíveis. EXTERNAL é utilizado para dados externos não comprimidos, e EXTENDED é utilizado para dados externos comprimidos. EXTENDED é o padrão para a maioria dos tipos de dados que suportam armazenamento não-PLAIN. Uso de EXTERNA fará operações de substring em grande texto e valores bytea correr mais rápido, a pena de maior espaço de armazenamento. Note-se que SET STORAGE não em si muda nada na tabela, apenas define a estratégia a ser seguida durante a atualização da tabela futuras. Consulte a Seção 55.2 para obter mais informações.

ADD ***table\_constraint*** [ NOT VALID ]

Esta forma adiciona uma nova restrição à tabela utilizando a mesma sintaxe de CREATE TABLE, além da opção não é válida, que atualmente só é permitido para restrições de chave estrangeira. Se a restrição é marcada não é válida, a verificação inicial potencialmente longo para verificar se todas as linhas na tabela satisfazer a restrição é ignorada. A restrição ainda serão aplicadas contra inserções ou atualizações (isto é, eles vão falhar a menos que haja uma linha correspondente na tabela referenciada) subseqüentes. Mas o banco de dados não irá assumir que a restrição é válida para todas as linhas na tabela, até que seja validado usando a opção de validação CONSTRAINT.

ADD ***table\_constraint\_using\_index***

This form adds a new PRIMARY KEY or UNIQUE constraint to a table based on an existing unique index. All the columns of the index will be included in the constraint.

The index cannot have expression columns nor be a partial index. Also, it must be a b-tree index with default sort ordering. These restrictions ensure that the index is equivalent to one that would be built by a regular ADD PRIMARY KEY or ADD UNIQUE command.

If PRIMARY KEY is specified, and the index's columns are not already marked NOT NULL, then this command will attempt to do ALTER COLUMN SET NOT NULL against each such column. That requires a full table scan to verify the column(s) contain no nulls. In all other cases, this is a fast operation.

If a constraint name is provided then the index will be renamed to match the constraint name. Otherwise the constraint will be named the same as the index.

After this command is executed, the index is "owned" by the constraint, in the same way as if the index had been built by a regular ADD PRIMARY KEY or ADD UNIQUEcommand. In particular, dropping the constraint will make the index disappear too.

**Note:** Adding a constraint using an existing index can be helpful in situations where a new constraint needs to be added without blocking table updates for a long time. To do that, create the index using CREATE INDEX CONCURRENTLY, and then install it as an official constraint using this syntax. See the example below.

VALIDATE CONSTRAINT

This form validates a foreign key constraint that was previously created as NOT VALID, by scanning the table to ensure there are no unmatched rows. Nothing happens if the constraint is already marked valid.

Validation can be a long process on larger tables and currently requires an ACCESS EXCLUSIVE lock. The value of separating validation from initial creation is that you can defer validation to less busy times, or can be used to give additional time to correct pre-existing errors while preventing new errors.

DROP CONSTRAINT [ IF EXISTS ]

This form drops the specified constraint on a table. If IF EXISTS is specified and the constraint does not exist, no error is thrown. In this case a notice is issued instead.

DISABLE/ENABLE [ REPLICA | ALWAYS ] TRIGGER

These forms configure the firing of trigger(s) belonging to the table. A disabled trigger is still known to the system, but is not executed when its triggering event occurs. For a deferred trigger, the enable status is checked when the event occurs, not when the trigger function is actually executed. One can disable or enable a single trigger specified by name, or all triggers on the table, or only user triggers (this option excludes internally generated constraint triggers such as those that are used to implement foreign key constraints or deferrable uniqueness and exclusion constraints). Disabling or enabling internally generated constraint triggers requires superuser privileges; it should be done with caution since of course the integrity of the constraint cannot be guaranteed if the triggers are not executed. The trigger firing mechanism is also affected by the configuration variable [session\_replication\_role](http://www.postgresql.org/docs/9.1/static/runtime-config-client.html#GUC-SESSION-REPLICATION-ROLE). Simply enabled triggers will fire when the replication role is "origin" (the default) or"local". Triggers configured as ENABLE REPLICA will only fire if the session is in "replica" mode, and triggers configured as ENABLE ALWAYS will fire regardless of the current replication mode.

DISABLE/ENABLE [ REPLICA | ALWAYS ] RULE

These forms configure the firing of rewrite rules belonging to the table. A disabled rule is still known to the system, but is not applied during query rewriting. The semantics are as for disabled/enabled triggers. This configuration is ignored for ON SELECT rules, which are always applied in order to keep views working even if the current session is in a non-default replication role.

CLUSTER

This form selects the default index for future [CLUSTER](http://www.postgresql.org/docs/9.1/static/sql-cluster.html) operations. It does not actually re-cluster the table.

SET WITHOUT CLUSTER

This form removes the most recently used [CLUSTER](http://www.postgresql.org/docs/9.1/static/sql-cluster.html) index specification from the table. This affects future cluster operations that don't specify an index.

SET WITH OIDS

This form adds an oid system column to the table (see [Section 5.4](http://www.postgresql.org/docs/9.1/static/ddl-system-columns.html)). It does nothing if the table already has OIDs.

Note that this is not equivalent to ADD COLUMN oid oid; that would add a normal column that happened to be named oid, not a system column.

SET WITHOUT OIDS

This form removes the oid system column from the table. This is exactly equivalent to DROP COLUMN oid RESTRICT, except that it will not complain if there is already no oidcolumn.

SET ( ***storage\_parameter*** = ***value*** [, ... ] )

This form changes one or more storage parameters for the table. See [Storage Parameters](http://www.postgresql.org/docs/9.1/static/sql-createtable.html#SQL-CREATETABLE-STORAGE-PARAMETERS) for details on the available parameters. Note that the table contents will not be modified immediately by this command; depending on the parameter you might need to rewrite the table to get the desired effects. That can be done with [VACUUM FULL](http://www.postgresql.org/docs/9.1/static/sql-vacuum.html),[CLUSTER](http://www.postgresql.org/docs/9.1/static/sql-cluster.html) or one of the forms of ALTER TABLE that forces a table rewrite.

**Note:** While CREATE TABLE allows OIDS to be specified in the WITH (***storage\_parameter***)syntax, ALTER TABLE does not treat OIDS as a storage parameter. Instead use the SET WITH OIDS and SET WITHOUT OIDS forms to change OID status.

RESET ( ***storage\_parameter*** [, ... ] )

This form resets one or more storage parameters to their defaults. As with SET, a table rewrite might be needed to update the table entirely.

INHERIT ***parent\_table***

This form adds the target table as a new child of the specified parent table. Subsequently, queries against the parent will include records of the target table. To be added as a child, the target table must already contain all the same columns as the parent (it could have additional columns, too). The columns must have matching data types, and if they have NOT NULL constraints in the parent then they must also have NOT NULL constraints in the child.

There must also be matching child-table constraints for all CHECK constraints of the parent. Currently UNIQUE, PRIMARY KEY, and FOREIGN KEY constraints are not considered, but this might change in the future.

NO INHERIT ***parent\_table***

This form removes the target table from the list of children of the specified parent table. Queries against the parent table will no longer include records drawn from the target table.

OF ***type\_name***

This form links the table to a composite type as though CREATE TABLE OF had formed it. The table's list of column names and types must precisely match that of the composite type; the presence of an oid system column is permitted to differ. The table must not inherit from any other table. These restrictions ensure that CREATE TABLE OF would permit an equivalent table definition.

NOT OF

This form dissociates a typed table from its type.

OWNER

This form changes the owner of the table, sequence, or view to the specified user.

SET TABLESPACE

This form changes the table's tablespace to the specified tablespace and moves the data file(s) associated with the table to the new tablespace. Indexes on the table, if any, are not moved; but they can be moved separately with additional SET TABLESPACE commands. See also [CREATE TABLESPACE](http://www.postgresql.org/docs/9.1/static/sql-createtablespace.html).

RENAME

The RENAME forms change the name of a table (or an index, sequence, or view) or the name of an individual column in a table. There is no effect on the stored data.

SET SCHEMA

This form moves the table into another schema. Associated indexes, constraints, and sequences owned by table columns are moved as well.

All the actions except RENAME and SET SCHEMA can be combined into a list of multiple alterations to apply in parallel. For example, it is possible to add several columns and/or alter the type of several columns in a single command. This is particularly useful with large tables, since only one pass over the table need be made.

You must own the table to use ALTER TABLE. To change the schema of a table, you must also have CREATE privilege on the new schema. To add the table as a new child of a parent table, you must own the parent table as well. To alter the owner, you must also be a direct or indirect member of the new owning role, and that role must have CREATEprivilege on the table's schema. (These restrictions enforce that altering the owner doesn't do anything you couldn't do by dropping and recreating the table. However, a superuser can alter ownership of any table anyway.)

**Parameters**

***name***

The name (optionally schema-qualified) of an existing table to alter. If ONLY is specified before the table name, only that table is altered. If ONLY is not specified, the table and all its descendant tables (if any) are altered. Optionally, \* can be specified after the table name to explicitly indicate that descendant tables are included.

***column***

Name of a new or existing column.

***new\_column***

New name for an existing column.

***new\_name***

New name for the table.

***type***

Data type of the new column, or new data type for an existing column.

***table\_constraint***

New table constraint for the table.

***constraint\_name***

Name of an existing constraint to drop.

CASCADE

Automatically drop objects that depend on the dropped column or constraint (for example, views referencing the column).

RESTRICT

Refuse to drop the column or constraint if there are any dependent objects. This is the default behavior.

***trigger\_name***

Name of a single trigger to disable or enable.

ALL

Disable or enable all triggers belonging to the table. (This requires superuser privilege if any of the triggers are internally generated constraint triggers such as those that are used to implement foreign key constraints or deferrable uniqueness and exclusion constraints.)

USER

Disable or enable all triggers belonging to the table except for internally generated constraint triggers such as those that are used to implement foreign key constraints or deferrable uniqueness and exclusion constraints.

***index\_name***

The index name on which the table should be marked for clustering.

***storage\_parameter***

The name of a table storage parameter.

***value***

The new value for a table storage parameter. This might be a number or a word depending on the parameter.

***parent\_table***

A parent table to associate or de-associate with this table.

***new\_owner***

The user name of the new owner of the table.

***new\_tablespace***

The name of the tablespace to which the table will be moved.

***new\_schema***

The name of the schema to which the table will be moved.

**Notes**

The key word COLUMN is noise and can be omitted.

When a column is added with ADD COLUMN, all existing rows in the table are initialized with the column's default value (NULL if no DEFAULT clause is specified).

Adding a column with a non-null default or changing the type of an existing column will require the entire table and indexes to be rewritten. As an exception, if the USING clause does not change the column contents and the old type is either binary coercible to the new type or an unconstrained domain over the new type, a table rewrite is not needed, but any indexes on the affected columns must still be rebuilt. Adding or removing a system oid column also requires rewriting the entire table. Table and/or index rebuilds may take a significant amount of time for a large table; and will temporarily require as much as double the disk space.

Adding a CHECK or NOT NULL constraint requires scanning the table to verify that existing rows meet the constraint.

The main reason for providing the option to specify multiple changes in a single ALTER TABLE is that multiple table scans or rewrites can thereby be combined into a single pass over the table.

The DROP COLUMN form does not physically remove the column, but simply makes it invisible to SQL operations. Subsequent insert and update operations in the table will store a null value for the column. Thus, dropping a column is quick but it will not immediately reduce the on-disk size of your table, as the space occupied by the dropped column is not reclaimed. The space will be reclaimed over time as existing rows are updated. (These statements do not apply when dropping the system oid column; that is done with an immediate rewrite.)

To force an immediate rewrite of the table, you can use [VACUUM FULL](http://www.postgresql.org/docs/9.1/static/sql-vacuum.html), [CLUSTER](http://www.postgresql.org/docs/9.1/static/sql-cluster.html) or one of the forms of ALTER TABLE that forces a rewrite. This results in no semantically-visible change in the table, but gets rid of no-longer-useful data.

The USING option of SET DATA TYPE can actually specify any expression involving the old values of the row; that is, it can refer to other columns as well as the one being converted. This allows very general conversions to be done with the SET DATA TYPE syntax. Because of this flexibility, the USING expression is not applied to the column's default value (if any); the result might not be a constant expression as required for a default. This means that when there is no implicit or assignment cast from old to new type, SET DATA TYPE might fail to convert the default even though a USING clause is supplied. In such cases, drop the default with DROP DEFAULT, perform the ALTER TYPE, and then use SET DEFAULT to add a suitable new default. Similar considerations apply to indexes and constraints involving the column.

If a table has any descendant tables, it is not permitted to add, rename, or change the type of a column in the parent table without doing the same to the descendants. That is,ALTER TABLE ONLY will be rejected. This ensures that the descendants always have columns matching the parent.

A recursive DROP COLUMN operation will remove a descendant table's column only if the descendant does not inherit that column from any other parents and never had an independent definition of the column. A nonrecursive DROP COLUMN (i.e., ALTER TABLE ONLY ... DROP COLUMN) never removes any descendant columns, but instead marks them as independently defined rather than inherited.

The TRIGGER, CLUSTER, OWNER, and TABLESPACE actions never recurse to descendant tables; that is, they always act as though ONLY were specified. Adding a constraint can recurse only for CHECK constraints, and is required to do so for such constraints.

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

Refer to [CREATE TABLE](http://www.postgresql.org/docs/9.1/static/sql-createtable.html) for a further description of valid parameters. [Chapter 5](http://www.postgresql.org/docs/9.1/static/ddl.html) has further information on inheritance.

**Examples**

To add a column of type varchar to a table:

ALTER TABLE distributors ADD COLUMN address varchar(30);

To drop a column from a table:

ALTER TABLE distributors DROP COLUMN address RESTRICT;

To change the types of two existing columns in one operation:

ALTER TABLE distributors

ALTER COLUMN address TYPE varchar(80),

ALTER COLUMN name TYPE varchar(100);

To change an integer column containing UNIX timestamps to timestamp with time zone via a USING clause:

ALTER TABLE foo

ALTER COLUMN foo\_timestamp SET DATA TYPE timestamp with time zone

USING

timestamp with time zone 'epoch' + foo\_timestamp \* interval '1 second';

The same, when the column has a default expression that won't automatically cast to the new data type:

ALTER TABLE foo

ALTER COLUMN foo\_timestamp DROP DEFAULT,

ALTER COLUMN foo\_timestamp TYPE timestamp with time zone

USING

timestamp with time zone 'epoch' + foo\_timestamp \* interval '1 second',

ALTER COLUMN foo\_timestamp SET DEFAULT now();

To rename an existing column:

ALTER TABLE distributors RENAME COLUMN address TO city;

To rename an existing table:

ALTER TABLE distributors RENAME TO suppliers;

To add a not-null constraint to a column:

ALTER TABLE distributors ALTER COLUMN street SET NOT NULL;

To remove a not-null constraint from a column:

ALTER TABLE distributors ALTER COLUMN street DROP NOT NULL;

To add a check constraint to a table and all its children:

ALTER TABLE distributors ADD CONSTRAINT zipchk CHECK (char\_length(zipcode) = 5);

To remove a check constraint from a table and all its children:

ALTER TABLE distributors DROP CONSTRAINT zipchk;

To remove a check constraint from one table only:

ALTER TABLE ONLY distributors DROP CONSTRAINT zipchk;

(The check constraint remains in place for any child tables.)

To add a foreign key constraint to a table:

ALTER TABLE distributors ADD CONSTRAINT distfk FOREIGN KEY (address) REFERENCES addresses (address) MATCH FULL;

To add a (multicolumn) unique constraint to a table:

ALTER TABLE distributors ADD CONSTRAINT dist\_id\_zipcode\_key UNIQUE (dist\_id, zipcode);

To add an automatically named primary key constraint to a table, noting that a table can only ever have one primary key:

ALTER TABLE distributors ADD PRIMARY KEY (dist\_id);

To move a table to a different tablespace:

ALTER TABLE distributors SET TABLESPACE fasttablespace;

To move a table to a different schema:

ALTER TABLE myschema.distributors SET SCHEMA yourschema;

To recreate a primary key constraint, without blocking updates while the index is rebuilt:

CREATE UNIQUE INDEX CONCURRENTLY dist\_id\_temp\_idx ON distributors (dist\_id);

ALTER TABLE distributors DROP CONSTRAINT distributors\_pkey,

ADD CONSTRAINT distributors\_pkey PRIMARY KEY USING INDEX dist\_id\_temp\_idx;

**Compatibility**

The forms ADD (without USING INDEX), DROP, SET DEFAULT, and SET DATA TYPE (without USING) conform with the SQL standard. The other forms are PostgreSQL extensions of the SQL standard. Also, the ability to specify more than one manipulation in a single ALTER TABLE command is an extension.

ALTER TABLE DROP COLUMN can be used to drop the only column of a table, leaving a zero-column table. This is an extension of SQL, which disallows zero-column tables.

**See Also**

[CREATE TABLE](http://www.postgresql.org/docs/9.1/static/sql-createtable.html)

|  |  |  |
| --- | --- | --- |
| [Prev](http://www.postgresql.org/docs/9.1/static/sql-alterserver.html) | [Home](http://www.postgresql.org/docs/9.1/static/index.html) | [Next](http://www.postgresql.org/docs/9.1/static/sql-altertablespace.html) |
| ALTER SERVER | [Up](http://www.postgresql.org/docs/9.1/static/sql-commands.html) | ALTER TABLESPACE |

[Privacy Policy](http://www.postgresql.org/about/privacypolicy) | [About PostgreSQL](http://www.postgresql.org/about/)  
Copyright © 1996-2013 The PostgreSQL Global Development Group