SQL Server 2005 Books Online (September 2007)

ALTER TABLE (Transact-SQL)

Updated: **15 September 2007**

Modifies a table definition by altering, adding, or dropping columns and constraints, reassigning partitions, or disabling or enabling constraints and triggers.

Transact-SQL Syntax Conventions [http://msdn2.microsoft.com/en-us/library/ms177563(printer).aspx]

Syntax

```
ALTER TABLE [ database_name . [ schema_name ] . | schema_name . ] table_name
{
    ALTER COLUMN column_name
    {
        [ type_schema_name. ] type_name [ ( { precision [ , scale ]
            | max | xml_schema_collection } ) ]
        [ COLLATE collation_name ]
        [ NULL | NOT NULL ]
    | {ADD | DROP } { ROWGUIDCOL | PERSISTED | NOT FOR REPLICATION}
    }
    | [ WITH { CHECK | NOCHECK } ] ADD
    {
        <column_definition>
      | <computed_column_definition>
      | <table_constraint>
    } [ ,...n ]
    | DROP
    {
        [ CONSTRAINT ] constraint_name
        [ WITH ( <drop_clustered_constraint_option> [ ,...n ] ) ]
        | COLUMN column_name
    } [ ,...n ]
    | [ WITH { CHECK | NOCHECK } ] { CHECK | NOCHECK } CONSTRAINT
        { ALL | constraint_name [ ,...n ] }
    | { ENABLE | DISABLE } TRIGGER
        { ALL | trigger_name [ ,...n ] }
    | SWITCH [ PARTITION source_partition_number_expression ]
        TO target_table
        [ PARTITION target_partition_number_expression ]
}
[ ; ]

<drop_clustered_constraint_option> ::=
    {
        MAXDOP = max_degree_of_parallelism
      | ONLINE = {ON | OFF }
      | MOVE TO { partition_scheme_name ( column_name ) | filegroup
          | "default"}
    }
```

Arguments

database_name
Is the name of the database in which the table was created.

schema_name
Is the name of the schema to which the table belongs.

table_name

Is the name of the table to be altered. If the table is not in the current database or is not contained by the schema owned by the current user, the database and schema must be explicitly specified.

ALTER COLUMN
Specifies that the named column is to be changed or altered. ALTER COLUMN is not allowed if the compatibility level is 65 or lower. For more information, see sp_dbcmptlevel (Transact-SQL) [http:// msdn2.microsoft.com/en-us/library/ms178653(printer).aspx] .

The modified column cannot be any one of the following:

- A column with a **timestamp** data type.
- The ROWGUIDCOL for the table.
- A computed column or used in a computed column.
- Used in an index, unless the column is a **varchar**, **nvarchar**, or **varbinary** data type, the data type is not changed, the new size is equal to or larger than the old size, and the index is not the result of a PRIMARY KEY constraint.
- Used in statistics generated by the CREATE STATISTICS statement. First, remove the statistics using the DROP STATISTICS statement. Statistics that are automatically generated by the query optimizer are automatically dropped by ALTER COLUMN.
- Used in a PRIMARY KEY or [FOREIGN KEY] REFERENCES constraint.
- Used in a CHECK or UNIQUE constraint. However, changing the length of a variable-length column used in a CHECK or UNIQUE constraint is allowed.
- Associated with a default definition. However, the length, precision, or scale of a column can be changed if the data type is not changed.
  The data type of **text**, **ntext** and **image** columns can be changed only in the following ways:
  - **text** to **varchar(max)**, **nvarchar(max)**, or **xml**
  - **ntext** to **varchar(max)**, **nvarchar(max)**, or **xml**
  - **image** to **varbinary(max)**

  Some data type changes may cause a change in the data. For example, changing an **nchar** or **nvarchar** column to **char** or **varchar** may cause the conversion of extended characters. For more information, see CAST and CONVERT (Transact-SQL) [http://msdn2.microsoft.com/en-us/library/ms187928(printer).aspx] . Reducing the precision or scale of a column may cause data truncation.
  The data type of a column of a partitioned table cannot be changed.

column_name
Is the name of the column to be altered, added, or dropped. *column_name* can be a maximum of 128 characters. For new columns, *column_name* can be omitted for columns created with a **timestamp** data type. The name **timestamp** is used if no *column_name* is specified for a **timestamp** data type column.

[*type_schema_name***.**] *type_name*
Is the new data type for the altered column, or the data type for the added column. *type_name* cannot be specified for existing columns of partitioned tables. *type_name* can be any one of the following:

- A SQL Server 2005 system data type.
- An alias data type based on a SQL Server system data type. Alias data types are created with the CREATE TYPE statement before they can be used in a table definition.
- A .NET Framework user-defined type, and the schema to which it belongs. .NET Framework user-defined types are created with the CREATE TYPE statement before they can be used in a table definition.

The following are criteria for *type_name* of an altered column:

- The previous data type must be implicitly convertible to the new data type.
- *type_name* cannot be **timestamp**.
- ANSI_NULL defaults are always on for ALTER COLUMN; if not specified, the column is nullable.
- ANSI_PADDING padding is always ON for ALTER COLUMN.
- If the modified column is an identity column, *new_data_type* must be a data type that supports the identity property.
- The current setting for SET ARITHABORT is ignored. ALTER TABLE operates as if ARITHABORT is set to ON.

Note:

If the COLLATE clause is not specified, changing the data type of a column will cause a collation change to the default collation of the database.

precision
Is the precision for the specified data type. For more information about valid precision values, see Precision, Scale, and Length (Transact-SQL) [http://msdn2.microsoft.com/en-us/library/ms190476(printer).aspx] .

scale
Is the scale for the specified data type. For more information about valid scale values, see Precision, Scale, and Length (Transact-SQL) [http://msdn2.microsoft.com/en-us/library/ms190476(printer).aspx] .

max
Applies only to the **varchar**, **nvarchar**, and **varbinary** data types for storing 2^31-1 bytes of character, binary data, and of Unicode data.

xml_schema_collection
Applies only to the **xml** data type for associating an XML schema with the type. Before typing an **xml** column to a schema collection, the schema collection must first be created in the database by using CREATE XML SCHEMA COLLECTION [http://msdn2.microsoft.com/en-us/library/ms176009(printer).aspx] .

COLLATE < *collation_name* >
Specifies the new collation for the altered column. If not specified, the column is assigned the default collation of the database. Collation name can be either a Windows collation name or a SQL collation name. For a list and more information, see Windows Collation Name (Transact-SQL) [http://msdn2.microsoft.com/en-us/library/ms188046(printer).aspx] and SQL Collation Name (Transact-SQL) [http://msdn2.microsoft.com/en-us/library/ms180175(printer).aspx] .

The COLLATE clause can be used to change the collations only of columns of the **char**, **varchar**, **nchar**, and **nvarchar** data types. To change the collation of a user-defined alias data type column, you must execute separate ALTER TABLE statements to change the column to a SQL Server system data type and change its collation, and then change the column back to an alias data type.

ALTER COLUMN cannot have a collation change if one or more of the following conditions exist:

- If a CHECK constraint, FOREIGN KEY constraint, or computed columns reference the column changed.
- If any index, statistics, or full-text index are created on the column. Statistics created automatically on the column changed are dropped if the column collation is changed.
- If a schema-bound view or function references the column.

For more information, see COLLATE (Transact-SQL) [http://msdn2.microsoft.com/en-us/library/ms184391(printer).aspx] .

NULL | NOT NULL
Specifies whether the column can accept null values. Columns that do not allow null values can be added with ALTER TABLE only if they have a default specified or if the table is empty. NOT NULL can be specified for computed columns only if PERSISTED is also specified. If the new column allows null values and no default is specified, the new column contains a null value for each row in the table. If the new column allows null values and a default definition is added with the new column, WITH VALUES can be used to store the default value in the new column for each existing row in the table.

If the new column does not allow null values and the table is not empty, a DEFAULT definition must be added with the new column, and the new column automatically loads with the default value in the new columns in each existing row.

NULL can be specified in ALTER COLUMN to force a NOT NULL column to allow null values, except for columns in PRIMARY KEY constraints. NOT NULL can be specified in ALTER COLUMN only if the column contains no null values. The null values must be updated to some value before the ALTER COLUMN NOT NULL is allowed, for example:

```
UPDATE MyTable SET NullCol = N'some_value' WHERE NullCol IS NULL
ALTER TABLE MyTable ALTER COLUMN NullCOl NVARCHAR(20) NOT NULL
```

When you create or alter a table with the CREATE TABLE or ALTER TABLE statements, the database and session settings influence and possibly override the nullability of the data type that is used in a column definition. We recommend that you always explicitly define a column as NULL or NOT NULL for noncomputed columns or, if you use a user-defined data type, that you allow the column to use the default nullability of the data type. For more information, see CREATE TABLE (Transact-SQL) [http://msdn2.microsoft.com/en-us/library/ms174979(printer).aspx] .

Note:

If NULL or NOT NULL is specified with ALTER COLUMN, *new_data_type* [(*precision* [, *scale*])] must also be specified. If the data type, precision, and scale are not changed, specify the current column values.

[{ADD | DROP} ROWGUIDCOL]
Specifies the ROWGUIDCOL property is added to or dropped from the specified column. ROWGUIDCOL indicates that the column is a row GUID column. Only one **uniqueidentifier** column per table can be designated as the ROWGUIDCOL column, and the ROWGUIDCOL property can be assigned only to a **uniqueidentifier** column. ROWGUIDCOL cannot be assigned to a column of a user-defined data type.

ROWGUIDCOL does not enforce uniqueness of the values that are stored in the column and does not automatically generate values for new rows that are inserted into the table. To generate unique values for each column, either use the NEWID function on INSERT statements or specify the NEWID function as the default for the column.

[{ADD | DROP} PERSISTED]
Specifies that the PERSISTED property is added to or dropped from the specified column. The column must be a computed column that is defined with a deterministic expression. For columns specified as PERSISTED, the SQL Server 2005 Database Engine physically stores the computed values in the table and updates the values when any other columns on which the computed column depends are updated. By marking a computed column as PERSISTED, you can create indexes on computed columns defined on expressions that are deterministic, but not precise. For more information, see Creating Indexes on Computed Columns [http://msdn2.microsoft.com/en-us/library/ms189292(printer).aspx] .

Any computed column that is used as a partitioning column of a partitioned table must be explicitly marked PERSISTED.

NOT FOR REPLICATION
Specifies that values are not incremented in identity columns when replication agents perform insert operations. This clause can be specified only if *column_name* is an identity column. For more information, see Controlling Constraints, Identities, and Triggers with NOT FOR REPLICATION [http://msdn2.microsoft.com/en-us/library/ms152529(printer).aspx] .

WITH CHECK | WITH NOCHECK
Specifies whether the data in the table is or is not validated against a newly added or re-enabled FOREIGN KEY or CHECK constraint. If not specified, WITH CHECK is assumed for new constraints, and WITH NOCHECK is assumed for re-enabled constraints.

If you do not want to verify new CHECK or FOREIGN KEY constraints against existing data, use WITH NOCHECK. We do not recommend doing this, except in rare cases. The new constraint will be evaluated in all later data updates. Any constraint violations that are suppressed by WITH NOCHECK when the constraint is added may cause future updates to fail if they update rows with data that does not comply with the constraint.

The query optimizer does not consider constraints that are defined WITH NOCHECK. Such constraints are ignored until they are re-enabled by using ALTER TABLE *table* CHECK CONSTRAINT ALL.

ADD
Specifies that one or more column definitions, computed column definitions, or table constraints are added.

DROP { [CONSTRAINT] *constraint_name* | COLUMN *column_name* }
Specifies that *constraint_name* or *column_name* is removed from the table. Multiple columns and constraints can be listed. DROP COLUMN is not allowed if the compatibility level is 65 or earlier. For more information, see sp_dbcmptlevel (Transact-SQL) [http://msdn2.microsoft.com/en-us/library/ms178653(printer).aspx] .

The user-defined or system-supplied name of the constraint can be determined by querying the **sys.check_constraint**, **sys.default_constraints**, **sys.key_constraints**, and **sys.foreign_keys** catalog views.

A PRIMARY KEY constraint cannot be dropped if an XML index exists on the table.

A column cannot be dropped when it is:

- Used in an index.
- Used in a CHECK, FOREIGN KEY, UNIQUE, or PRIMARY KEY constraint.
- Associated with a default that is defined with the DEFAULT keyword, or bound to a default object.
- Bound to a rule.

> **Note:**
>
> Dropping a column does not reclaim the disk space of the column. You may have to reclaim the disk space of a dropped column when the row size of a table is near, or has exceeded, its limit. Reclaim space by creating a clustered index on the table or rebuilding an existing clustered index by using ALTER INDEX [http://msdn2.microsoft.com/en-us/library/ms188388(printer).aspx] .

WITH <drop_clustered_constraint_option>
Specifies that one or more drop clustered constraint options are set.

MAXDOP = *max_degree_of_parallelism*
Overrides the **max degree of parallelism** configuration option only for the duration of the operation. For more information, see max degree of parallelism Option [http://msdn2.microsoft.com/en-us/library/ms181007(printer).aspx] .

Use the MAXDOP option to limit the number of processors used in parallel plan execution. The maximum is 64 processors.

max_degree_of_parallelism can be one of the following values:

1
Suppresses parallel plan generation.

>1
Restricts the maximum number of processors used in a parallel index operation to the specified number.

0 (default)
Uses the actual number of processors or fewer based on the current system workload.

For more information, see Configuring Parallel Index Operations [http://msdn2.microsoft.com/en-us/library/ms189329(printer).aspx] .

> **Note:**
>
> Parallel index operations are available only in SQL Server 2005 Enterprise Edition.

ONLINE **=** { ON | OFF }
Specifies whether underlying tables and associated indexes are available for queries and data modification during the index operation. The default is OFF.

ON
Long-term table locks are not held for the duration of the index operation. During the main phase of the index operation, only an Intent Share (IS) lock is held on the source table. This enables queries or updates to the underlying table and indexes to continue. At the start of the operation, a Shared (S) lock is held on the source object for a very short time. At the end of the operation, for a short time, an S (Shared) lock is acquired on the source if a nonclustered index is being created; or an SCH-M (Schema Modification) lock is acquired when a clustered index is created or dropped online and when a clustered or nonclustered index is being rebuilt. ONLINE cannot be set to ON when an index is being created on a local temporary table.

OFF
Table locks are applied for the duration of the index operation. An offline index operation that creates, rebuilds, or drops a clustered index, or rebuilds or drops a nonclustered index, acquires a Schema modification (Sch-M) lock on the table. This prevents all user access to the underlying table for the duration of the operation. An offline index operation that creates a nonclustered index acquires a Shared (S) lock on the table. This prevents updates to the underlying table but allows read operations, such as SELECT statements.

For more information, see How Online Index Operations Work [http://msdn2.microsoft.com/en-us/library/ms191261(printer).aspx] . For more information about locks, see Lock Modes [http://msdn2.microsoft.com/en-us/library/ms175519(printer).aspx] .

Note:

Online index operations are available only in SQL Server 2005 Enterprise Edition.

MOVE TO (*partition_scheme_name* (*column_name* [1**,** ... *n*]) | *filegroup* | **"**default**"**}
Specifies a location to move the data rows currently in the leaf level of the clustered index. The table is moved to the new location.

Note:

In this context, default is not a keyword. It is an identifier for the default filegroup and must be delimited, as in MOVE TO **"**default**"** or MOVE TO **[**default**]**. If **"**default**"** is specified, the QUOTED_IDENTIFIER option must be ON for the current session. This is the default setting. For more information, see SET QUOTED_IDENTIFIER (Transact-SQL) [http://msdn2.microsoft.com/en-us/library/ms174393(printer).aspx] .

{ CHECK | NOCHECK } CONSTRAINT
Specifies that *constraint_name* is enabled or disabled. This option can only be used with FOREIGN KEY and CHECK constraints. When NOCHECK is specified, the constraint is disabled and future inserts or updates to the column are not validated against the constraint conditions. DEFAULT, PRIMARY KEY, and UNIQUE constraints cannot be disabled.

ALL
Specifies that all constraints are either disabled with the NOCHECK option or enabled with the CHECK option.

{ ENABLE | DISABLE } TRIGGER
Specifies that *trigger_name* is enabled or disabled. When a trigger is disabled it is still defined for the table; however, when INSERT, UPDATE, or DELETE statements are executed against the table, the actions in the trigger are not performed until the trigger is re-enabled.

ALL
Specifies that all triggers in the table are enabled or disabled.

trigger_name
Specifies the name of the trigger to disable or enable.

SWITCH [PARTITION *source_partition_number_expression*] TO *target_table* [PARTITION target_ *partition_number_expression*]
Switches a block of data in one of the following ways:

- Reassigns all data of a table as a partition to an already-existing partitioned table.
- Switches a partition from one partitioned table to another.
- Reassigns all data in one partition of a partitioned table to an existing non-partitioned table.

If *table* is a partitioned table, *source_partition_number_expression* must be specified. If *target_table* is partitioned, *target_partition_number_expression* must be specified. If reassigning a table's data as a partition to an already-existing partitioned table, or switching a partition from one partitioned table to another, the target partition must exist and it must be empty.

If reassigning one partition's data to form a single table, the target table must already be created and it must be empty. Both the source table or partition, and the target table or partition, must reside in the same filegroup. The corresponding indexes, or index partitions, must also reside in the same filegroup. Many additional restrictions apply to switching partitions. For more information, see Transferring Data Efficiently by Using Partition Switching [http://msdn2.microsoft.com/en-us/library/ms191160(printer).aspx] . *table* and *target_table* cannot be the same. *target_table* can be a multi-part identifier.

source_partition_number_expression and *target_partition_number_expression* are constant expressions that can reference variables and functions. These include user-defined type variables and user-defined functions. They cannot reference Transact-SQL expressions.

Note:

You cannot use the SWITCH statement on replicated tables.

Remarks

To add new rows of data, use INSERT [http://msdn2.microsoft.com/en-us/library/ms174335(printer).aspx] . To remove rows of data, use DELETE [http://msdn2.microsoft.com/en-us/library/ms189835(printer).aspx] or TRUNCATE TABLE [http://msdn2.microsoft.com/en-us/library/ms177570(printer).aspx] . To change the values in existing rows, use UPDATE [http://msdn2.microsoft.com/en-us/library/ms177523(printer).aspx] .

If there are any execution plans in the procedure cache that reference the table, ALTER TABLE marks them to be recompiled on their next execution.

Changing the Size of a Column

You can change the length, precision, or scale of a column by specifying a new size for the column data type in the ALTER COLUMN clause. If data exists in the column, the new size cannot be smaller than the maximum length of the data. Also, the column cannot be defined in an index, unless the column is a **varchar**, **nvarchar**, or **varbinary** data type and the index is not the result of a PRIMARY KEY constraint. See Example P.

Locks and ALTER TABLE

The changes specified in ALTER TABLE are implemented immediately. If the changes require modifications of the rows in the table, ALTER TABLE updates the rows. ALTER TABLE acquires a schema modify lock on the table to make sure that no other connections reference even the metadata for the table during the change, except online index operations that require a very short SCH-M lock at the end. In an ALTER TABLE…SWITCH operation, the lock is acquired on both the source and target tables. The modifications made to the table are logged and fully recoverable. Changes that affect all the rows in very large tables, such as dropping a column or adding a NOT NULL column with a default, can take a long time to complete and generate many log records. These ALTER TABLE statements should be executed with the same care as any INSERT, UPDATE, or DELETE statement that affects many rows.

Parallel Plan Execution

In SQL Server 2005 Enterprise Edition, the number of processors employed to run a single ALTER TABLE ADD (index based) CONSTRAINT or DROP (clustered index) CONSTRAINT statement is determined by the **max degree of parallelism** configuration option and the current workload. If the Database Engine detects that the system is busy, the degree of parallelism of the operation is automatically reduced before statement execution starts. You can manually configure the number of processors that are used to run the statement by specifying the MAXDOP option.

Partitioned Tables

In addition to performing SWITCH operations that involve partitioned tables, ALTER TABLE can be used to change the state of the columns, constraints, and triggers of a partitioned table just like it is used for nonpartitioned tables. However, this statement cannot be used to change the way the table itself is partitioned. To repartition a partitioned table, use ALTER PARTITION SCHEME [http://msdn2.microsoft.com/en-us/library/ms190347(printer).aspx] and ALTER PARTITION FUNCTION [http://msdn2.microsoft.com/en-us/library/ms186307(printer).aspx] . Additionally, you cannot change the data type of a column of a partitioned table.

Restrictions on Tables with Schema-Bound Views

The restrictions that apply to ALTER TABLE statements on tables with schema-bound views are the same as the restrictions currently applied when modifying tables with a simple index. Adding a column is allowed. However, removing or changing a column that participates in any schema-bound view is not allowed. If the ALTER TABLE statement requires changing a column used in a schema-bound view, ALTER TABLE fails and the Database Engine raises an error message. For more information about schema binding and indexed views, see CREATE VIEW (Transact-SQL) [http://msdn2.microsoft.com/en-us/library/ms187956(printer).aspx] .

Adding or removing triggers on base tables is not affected by creating a schema-bound view that references the tables.

Indexes and ALTER TABLE

Indexes created as part of a constraint are dropped when the constraint is dropped. Indexes that were created with CREATE INDEX must be dropped with DROP INDEX. The ALTER INDEX statement can be used to rebuild an index part of a constraint definition; the constraint does not have to be dropped and added again with ALTER TABLE.

All indexes and constraints based on a column must be removed before the column can be removed.

When a constraint that created a clustered index is deleted, the data rows that were stored in the leaf level of the clustered index are stored in a nonclustered table. In SQL Server 2005, you can drop the clustered index and move the resulting table to another filegroup or partition scheme in a single transaction by specifying the MOVE TO option. The MOVE TO option has the following restrictions:

- MOVE TO is not valid for indexed views or nonclustered indexes.
- The partition scheme or filegroup must already exist.
- If MOVE TO is not specified, the table will be located in the same partition scheme or filegroup as was defined for the clustered index.

When you drop a clustered index, you can specify ONLINE = ON option so the DROP INDEX transaction does not block queries and modifications to the underlying data and associated nonclustered indexes.

ONLINE = ON has the following restrictions:

- ONLINE = ON is not valid for clustered indexes that are also disabled. Disabled indexes must be dropped by using ONLINE = OFF.
- Only one index at a time can be dropped.
- ONLINE = ON is not valid for indexed views, nonclustered indexes or indexes on local temp tables.

Temporary disk space equal to the size of the existing clustered index is required to drop a clustered index. This additional space is released as soon as the operation is completed.

Note:

The options listed under <*drop_clustered_constraint_option*> apply to clustered indexes on tables and cannot be applied to clustered indexes on views or nonclustered indexes.

Replicating Schema Changes

By default, when you run ALTER TABLE on a published table at a SQL Server Publisher, that change is propagated to all SQL Server Subscribers. This functionality has some restrictions and can be disabled. For more information, see Making Schema Changes on Publication Databases [http://msdn2.microsoft.com/en-us/library/ms151870(printer).aspx] .

Permissions

Requires ALTER permission on the table.

ALTER TABLE permissions apply to both tables involved in an ALTER TABLE SWITCH statement. Any data that is switched inherits the security of the target table.

If any columns in the ALTER TABLE statement are defined to be of a common language runtime (CLR) user-defined type or alias data type, REFERENCES permission on the type is required.

Examples

A. Adding a new column

The following example adds a column that allows null values and has no values provided through a DEFAULT definition. In the new column, each row will have `NULL`.

```
CREATE TABLE doc_exa ( column_a INT) ;
GO
ALTER TABLE doc_exa ADD column_b VARCHAR(20) NULL ;
GO
EXEC sp_help doc_exa ;
GO
DROP TABLE doc_exa ;
GO
```

B. Dropping a column

The following example modifies a table to remove a column.

```
CREATE TABLE doc_exb ( column_a INT, column_b VARCHAR(20) NULL) ;
GO
ALTER TABLE doc_exb DROP COLUMN column_b ;
GO
EXEC sp_help doc_exb ;
GO
DROP TABLE doc_exb ;
GO
```

C. Changing the data type of a column

The following example changes a column of a table from `INT` to `DECIMAL`.

```
CREATE TABLE doc_exy ( column_a INT ) ;
GO
INSERT INTO doc_exy (column_a)
VALUES (10) ;
GO
ALTER TABLE doc_exy ALTER COLUMN column_a DECIMAL (5, 2) ;
GO
DROP TABLE doc_exy ;
GO
```

D. Adding a column with a constraint

The following example adds a new column with a `UNIQUE` constraint.

```
CREATE TABLE doc_exc ( column_a INT) ;
GO
ALTER TABLE doc_exc ADD column_b VARCHAR(20) NULL
    CONSTRAINT exb_unique UNIQUE ;
GO
EXEC sp_help doc_exc ;
GO
DROP TABLE doc_exc ;
GO
```

E. Adding an unverified CHECK constraint to an existing column

The following example adds a constraint to an existing column in the table. The column has a value that violates the constraint. Therefore, `WITH NOCHECK` is used to prevent the constraint from being validated against existing rows, and to allow for the constraint to be added.

```
CREATE TABLE doc_exd ( column_a INT) ;
GO
INSERT INTO doc_exd VALUES (-1) ;
GO
ALTER TABLE doc_exd WITH NOCHECK
ADD CONSTRAINT exd_check CHECK (column_a > 1) ;
GO
EXEC sp_help doc_exd ;
GO
DROP TABLE doc_exd ;
GO
```

F. Adding a DEFAULT constraint to an existing column

The following example creates a table with two columns and inserts a value into the first column, and the other column remains NULL. A `DEFAULT` constraint is then added to the second column. To verify that the default is applied, another value is inserted into the first column, and the table is queried.

```
CREATE TABLE doc_exz ( column_a INT, column_b INT) ;
GO
INSERT INTO doc_exz (column_a)
VALUES ( 7 ) ;
GO
ALTER TABLE doc_exz
ADD CONSTRAINT col_b_def
DEFAULT 50 FOR column_b ;
GO
INSERT INTO doc_exz (column_a)
VALUES ( 10 ) ;
GO
SELECT * FROM doc_exz ;
GO
DROP TABLE doc_exz ;
GO
```

G. Adding several columns with constraints

The following example adds several columns with constraints defined with the new column. The first new column has an `IDENTITY` property. Each row in the table has new incremental values in the identity column.

```
CREATE TABLE doc_exe ( column_a INT CONSTRAINT column_a_un UNIQUE) ;
GO
ALTER TABLE doc_exe ADD

-- Add a PRIMARY KEY identity column.
column_b INT IDENTITY
CONSTRAINT column_b_pk PRIMARY KEY,

-- Add a column that references another column in the same table.
column_c INT NULL
CONSTRAINT column_c_fk
REFERENCES doc_exe(column_a),

-- Add a column with a constraint to enforce that
-- nonnull data is in a valid telephone number format.
column_d VARCHAR(16) NULL
CONSTRAINT column_d_chk
CHECK
(column_d LIKE '[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]' OR
column_d LIKE
'([0-9][0-9][0-9]) [0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]'),

-- Add a nonnull column with a default.
column_e DECIMAL(3,3)
CONSTRAINT column_e_default
DEFAULT .081 ;
GO
EXEC sp_help doc_exe ;
GO
DROP TABLE doc_exe ;
GO
```

H. Adding a nullable column with default values

The following example adds a nullable column with a `DEFAULT` definition, and uses `WITH VALUES` to provide values for each existing row in the table. If WITH VALUES is not used, each row has the value NULL in the new column.

```
Use AdventureWorks ;
GO
CREATE TABLE doc_exf ( column_a INT) ;
GO
INSERT INTO doc_exf
VALUES (1) ;
GO
ALTER TABLE doc_exf
ADD AddDate smalldatetime NULL
CONSTRAINT AddDateDflt
DEFAULT GETDATE() WITH VALUES ;
GO
DROP TABLE doc_exf ;
GO
```

I. Disabling and re-enabling a constraint

The following example disables a constraint that limits the salaries accepted in the data. `NOCHECK CONSTRAINT` is used with `ALTER TABLE` to disable the constraint and allow for an insert that would typically violate the constraint. `CHECK CONSTRAINT` re-enables the constraint.

```
CREATE TABLE cnst_example
(id INT NOT NULL,
    name VARCHAR(10) NOT NULL,
    salary MONEY NOT NULL
    CONSTRAINT salary_cap CHECK (salary < 100000)
)

-- Valid inserts
INSERT INTO cnst_example VALUES (1,'Joe Brown',65000)
INSERT INTO cnst_example VALUES (2,'Mary Smith',75000)

-- This insert violates the constraint.
INSERT INTO cnst_example VALUES (3,'Pat Jones',105000)

-- Disable the constraint and try again.
ALTER TABLE cnst_example NOCHECK CONSTRAINT salary_cap
INSERT INTO cnst_example VALUES (3,'Pat Jones',105000)

-- Re-enable the constraint and try another insert; this will fail.
ALTER TABLE cnst_example CHECK CONSTRAINT salary_cap
INSERT INTO cnst_example VALUES (4,'Eric James',110000) ;
```

J. Dropping a constraint

The following example removes a `UNIQUE` constraint from a table.

```
CREATE TABLE doc_exc ( column_a INT
CONSTRAINT my_constraint UNIQUE) ;
GO
ALTER TABLE doc_exc DROP CONSTRAINT my_constraint ;
GO
DROP TABLE doc_exc ;
GO
```

K. Switching partitions between tables

The following example creates a partitioned table, assuming that partition scheme myRangePS1 is already created in the database. Next, a nonpartitioned table is created with the same structure as the partitioned table and on the same filegroup as PARTITION 2 of table PartitionTable. The data of PARTITION 2 of table PartitionTable is then switched into table NonPartitionTable.

```
CREATE TABLE PartitionTable (col1 int, col2 char(10))
ON myRangePS1 (col1) ;
GO
CREATE TABLE NonPartitionTable (col1 int, col2 char(10))
ON test2fg ;
GO
ALTER TABLE PartitionTable SWITCH PARTITION 2 TO NonPartitionTable ;
GO
```

L. Disabling and re-enabling a trigger

The following example uses the DISABLE TRIGGER option of ALTER TABLE to disable the trigger and allow for an insert that would typically violate the trigger. ENABLE TRIGGER is then used to re-enable the trigger.

```
CREATE TABLE trig_example
(id INT,
name VARCHAR(12),
salary MONEY) ;
GO
-- Create the trigger.
CREATE TRIGGER trig1 ON trig_example FOR INSERT
AS
IF (SELECT COUNT(*) FROM INSERTED
WHERE salary > 100000) > 0
BEGIN
    print 'TRIG1 Error: you attempted to insert a salary > $100,000'
    ROLLBACK TRANSACTION
END ;
GO
-- Try an insert that violates the trigger.
INSERT INTO trig_example VALUES (1,'Pat Smith',100001) ;
GO
-- Disable the trigger.
ALTER TABLE trig_example DISABLE TRIGGER trig1 ;
GO
-- Try an insert that would typically violate the trigger.
INSERT INTO trig_example VALUES (2,'Chuck Jones',100001) ;
GO
-- Re-enable the trigger.
ALTER TABLE trig_example ENABLE TRIGGER trig1 ;
GO
-- Try an insert that violates the trigger.
INSERT INTO trig_example VALUES (3,'Mary Booth',100001) ;
GO
```

M. Creating a PRIMARY KEY constraint with index options

The following example creates the PRIMARY KEY constraint PK_TransactionHistoryArchive_TransactionID and sets the options FILLFACTOR, ONLINE, and PAD_INDEX. The resulting clustered index will have the same name as the constraint.

```
USE AdventureWorks;
GO
ALTER TABLE Production.TransactionHistoryArchive WITH NOCHECK
ADD CONSTRAINT PK_TransactionHistoryArchive_TransactionID PRIMARY KEY CLUSTERED
(TransactionID)
```

```
WITH (FILLFACTOR = 75, ONLINE = ON, PAD_INDEX = ON)
GO
```

N. Dropping a PRIMARY KEY constraint in the ONLINE mode

The following example deletes a PRIMARY KEY constraint with the `ONLINE` option set to `ON`.

```
USE AdventureWorks;
GO
ALTER TABLE Production.TransactionHistoryArchive
DROP CONSTRAINT PK_TransactionHistoryArchive_TransactionID
WITH (ONLINE = ON);
GO
```

O. Adding and dropping a FOREIGN KEY constraint

The following example creates the table `ContactBackup`, and then alters the table, first by adding a `FOREIGN KEY` constraint that references the table `Contact`, then by dropping the `FOREIGN KEY` constraint.

```
USE AdventureWorks ;
GO
CREATE TABLE Person.ContactBackup
(ContactID int) ;
GO
ALTER TABLE Person.ContactBackup
ADD CONSTRAINT FK_ContactBacup_Contact FOREIGN KEY (ContactID)
    REFERENCES Person.Contact (ContactID) ;
ALTER TABLE Person.ContactBackup
DROP CONSTRAINT FK_ContactBacup_Contact ;
GO
DROP TABLE Person.ContactBackup ;
```

P. Changing the size of a column

The following example increases the size of a **varchar** column and the precision and scale of a **decimal** column. Because the columns contain data, the column size can only be increased. Also notice that `col_a` is defined in a unique index. The size of `col_a` can still be increased because the data type is a **varchar** and the index is not the result of a PRIMARY KEY constraint.

```
IF OBJECT_ID ( 'dbo.doc_exy', 'U' ) IS NOT NULL
    DROP TABLE dbo.doc_exy;
GO
-- Create a two-column table with a unique index on the varchar column.
CREATE TABLE dbo.doc_exy ( col_a varchar(5) UNIQUE NOT NULL, col_b decimal (4,2));
GO
INSERT INTO dbo.doc_exy VALUES ('Test', 99.99);
GO
-- Verify the current column size.
SELECT name, TYPE_NAME(system_type_id), max_length, precision, scale
FROM sys.columns WHERE object_id = OBJECT_ID(N'dbo.doc_exy');
GO
-- Increase the size of the varchar column.
ALTER TABLE dbo.doc_exy ALTER COLUMN col_a varchar(25);
GO
-- Increase the scale and precision of the decimal column.
ALTER TABLE dbo.doc_exy ALTER COLUMN col_b decimal (10,4);
GO
-- Insert a new row.
INSERT INTO dbo.doc_exy VALUES ('MyNewColumnSize', 99999.9999) ;
GO
-- Verify the current column size.
```

```
SELECT name, TYPE_NAME(system_type_id), max_length, precision, scale
FROM sys.columns WHERE object_id = OBJECT_ID(N'dbo.doc_exy');
```

See Also

Reference

sp_rename (Transact-SQL) [http://msdn2.microsoft.com/en-us/library/ms188351(printer).aspx]
CREATE TABLE (Transact-SQL) [http://msdn2.microsoft.com/en-us/library/ms174979(printer).aspx]
DROP TABLE (Transact-SQL) [http://msdn2.microsoft.com/en-us/library/ms173790(printer).aspx]
sp_help (Transact-SQL) [http://msdn2.microsoft.com/en-us/library/ms187335(printer).aspx]
ALTER PARTITION SCHEME (Transact-SQL) [http://msdn2.microsoft.com/en-us/library/ms190347(printer).aspx]
ALTER PARTITION FUNCTION (Transact-SQL) [http://msdn2.microsoft.com/en-us/library/ms186307(printer).aspx]
EVENTDATA (Transact-SQL) [http://msdn2.microsoft.com/en-us/library/ms173781(printer).aspx]

Other Resources

Creating and Modifying Tables [http://msdn2.microsoft.com/en-us/library/ms189614(printer).aspx]
Making Schema Changes on Publication Databases [http://msdn2.microsoft.com/en-us/library/ms151870(printer).aspx]

Help and Information

Getting SQL Server 2005 Assistance [http://msdn2.microsoft.com/en-us/library/ms166016(printer).aspx]

Change History

Release	History
15 September 2007	**Changed content:** • Added the section, 'Changing the Size of a Column,' and Example P.
1 February 2007	**Changed content:** • Clarified the position and meaning of the NOT FOR REPLICATION clause in the syntax and arguments sections. • Clarified that the target table of a SWITCH clause can be expressed as a multipart identifier.
14 April 2006	**New content:** • Documented that you cannot use the SWITCH statement on replicated tables.
5 December 2005	**New content:** • Added DROP NOT FOR REPLICATION clause to syntax diagram and the Arguments definition list. **Changed content:** • Moved COLLATE clause to the correct position in the syntax diagram. • Fixed examples M and N.

Community Content

```
ERROR: syntaxerror
OFFENDING COMMAND: --nostringval--

STACK:

/Title
()
/Subject
(D:20080315095052+01’00’)
/ModDate
()
/Keywords
(PDFCreator Version 0.9.5)
/Creator
(D:20080315095052+01’00’)
/CreationDate
(Administrator)
/Author
-mark-
```