**CREATE TABLE (Transact-SQL) IDENTITY (Property)**

Syntax:

IDENTITY [ (seed , increment) ]

## Arguments

seed  
Is the value that is used for the very first row loaded into the table.

increment  
Is the incremental value that is added to the identity value of the previous row that was loaded.

You must specify both the seed and increment or neither. If neither is specified, the default is (1,1).

## Remarks

Identity columns can be used for generating key values. The identity property on a column guarantees the following:

* Each new value is generated based on the current seed & increment.
* Each new value for a particular transaction is different from other concurrent transactions on the table.

The identity property on a column does not guarantee the following:

* **Uniqueness of the value** - Uniqueness must be enforced by using a **PRIMARY KEY** or **UNIQUE** constraint or **UNIQUE** index.
* **Consecutive values within a transaction** - A transaction inserting multiple rows is not guaranteed to get consecutive values for the rows because other concurrent inserts might occur on the table. If values must be consecutive then the transaction should use an exclusive lock on the table or use the **SERIALIZABLE** isolation level.
* **Consecutive values after server restart or other failures** -SQL Server might cache identity values for performance reasons and some of the assigned values can be lost during a database failure or server restart. This can result in gaps in the identity value upon insert. If gaps are not acceptable then the application should use its own mechanism to generate key values. Using a sequence generator with the **NOCACHE** option can limit the gaps to transactions that are never committed.
* **Reuse of values** - For a given identity property with specific seed/increment, the identity values are not reused by the engine. If a particular insert statement fails or if the insert statement is rolled back then the consumed identity values are lost and will not be generated again. This can result in gaps when the subsequent identity values are generated.

These restrictions are part of the design in order to improve performance, and because they are acceptable in many common situations. If you cannot use identity values because of these restrictions, create a separate table holding a current value and manage access to the table and number assignment with your application.

If a table with an identity column is published for replication, the identity column must be managed in a way that is appropriate for the type of replication used. For more information, see [Replicate Identity Columns](https://docs.microsoft.com/en-us/sql/relational-databases/replication/publish/replicate-identity-columns?view=sql-server-ver15).

Only one identity column can be created per table.

## Example:

USE AdventureWorks2012;

IF OBJECT\_ID ('dbo.new\_employees', 'U') IS NOT NULL

DROP TABLE new\_employees;

GO

CREATE TABLE new\_employees

(

id\_num int IDENTITY(1,1),

fname varchar (20),

minit char(1),

lname varchar(30)

);

INSERT new\_employees

(fname, minit, lname)

VALUES

('Karin', 'F', 'Josephs');

INSERT new\_employees

(fname, minit, lname)

VALUES

('Pirkko', 'O', 'Koskitalo');