Steve Stedman
Freelance SQL Server Consultant
http://stevestedman.com

Are you wasting half the capacity of IDENTITY?

When creating a table you can specify IDENTITY with the following syntax:

For example

The IDENTITY property is valid for data types of the integer data type category, except for the bit data type, or decimal data type. So how many rows can you insert into that table? It depends on the data type used.

- BIGINT maximum value is 9,223,372,036,854,775,807
- INT maximum value is 2,147,483,647
- SMALLINT maximum value is 32,767
- TINYINT maximum value is 255

What happens when I run out of numbers? Let’s take a look at an example using SMALLINT.

At this point we have 1 row in the table. Now lets fill almost all the way full.

Keep in mind that specifying a number after the GO statement specifies that we want the previous statement run that many times. In this case run the insert statement 32756 times plus the one previous should leave us with 32766 rows in the table as shown here.

Now lets take a quick look at this table using the Identity Column Usage Report from Database Health Reports project.

Now the big question, there are 32766 items in the table, we know that it will hold 32767, what happens when we do another insert or 2?

This one works fine and puts our count at 32767, what next. Insert one more row...

This message is SQL Servers way of saying that the identity column is full.

```
Msg 8115, Level 16, State 1, Line 1
Arith
metic over
flow error convertin
g IDENTITY to data type smallint.  Arithmetic overflow occurred.
```

How do I fix it?

This is where it gets ugly. You can't change the datatype on a IDENTITY value on an existing table with dropping it and recreating it, which would lead to data loss.

If there are no tables with foreign key relationships, you could just do the following:

1. Create another table with a larger integer type for the IDENTITY column, in this case BIGINT or INT.
2. Then select everything out of this table into the new table, excluding the primary key.
3. Then drop the original table.
4. Then rename the new table to the original name.

The more complex solution is if there are foreign keys, you need to also drop the foreign key relationship, turn on INDENTITY_INSERT do the insert, and turn off IDENTITY INSERT, then re-enable the foreign keys. Here are the steps.

1. Create another table with a larger integer type for the IDENTITY column, in this case BIGINT or INT.
2. Disable all the foreign keys associated with the original table.
3. Turn on identity_insert for the new table.
4. Then select everything out of the original table into the new table, including the identity column.
5. Then drop the original table.
6. Then rename the new table to the original name.
7. Turn off IDENTITY_INSERT for the new table.
8. Re-enable all the foreign keys.

Oh, and did I mention do this at a time when nobody is inserting anything new into the original table.

How to prevent it?

2 options when you originally create a table.

- Use a larger integer data type.
- Seed the identity at the smallest value available for that data type.

- On SQL Server 2012 you could consider the SEQUENCE object instead of an IDENTITY.

The first option of a larger integer data type is more expensive on the database, for instance if you just decided to set every identity to be a BIGINT, you would have a huge amount of overhead and waste on many of your tables.

The second option of seeding the identity to the smallest value available for the sequence.

- BIGINT minimum value is -9,223,372,036,854,775,808
- INT minimum value is -2,147,483,648
- SMALLINT minimum value is -32,768
- TINYINT minimum value is 0

There really isn't anything to be gained with the TINYINT since its minimum value is 0, but with the 3 other INT data types, you double the capacity of your identity values by starting them at their minimum value instead of at 1.

Take a look at the following example, seeding the value with -32768, instead of 1.

After inserting 1 row, we see the following output.

Where you can see that the first value inserted has an identity value of -32768. Now what happens when we insert 37268 values.

Which produces the following output.

Where you can see that after inserting 32769 rows were are now at an IDENTITY value of 0, rather than overflowing.

Now lets take a look at the same table using the Identity Column Usage Report from Database Health Reports project.

Summary

From what we have observed, I ponder the question of why would you ever not start an identity at its minimum value in order to take full advantage the capacity of the datatype you are using.

-Steve Stedman

Bellingham WA