

## Re: ORDER BY and IDENTITY

**Source:**

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.programming/2004-12/4835.html>

---

**From:** Itzik Ben-Gan (*itzik\_at\_REMOVETHIS.SolidQualityLearning.com*)

**Date:** 12/21/04

Date: Tue, 21 Dec 2004 13:53:04 +0200

Hi,

I guess I'm the MVP you're referring to here, and that your question relates to my comments on the subject in the following article:

[http://www.windowsitpro.com/SQLServer/Article/ArticleID/44138/SQLServer\\_44138.html](http://www.windowsitpro.com/SQLServer/Article/ArticleID/44138/SQLServer_44138.html)

I see that you already got the pointers to the following articles discussing the SELECT INTO part of the problem:

<http://support.microsoft.com/default.aspx?scid=kb:en-us:273586>

<http://www.windowsitpro.com/SQLServer/Article/ArticleID/43553/43553.html>

The situation is somewhat similar to mathematics, where someone comes up with an assumption/conjecture based on observations, e.g., Fermat's last theorem which was proven right, and Goldbach's conjecture which is still unsolved. In Mathematics, suffice to come up with one example that contradicts the assumption to prove it wrong. With a T-SQL behavior, I'd say that it's nice to have an example, but suffice that Microsoft says that something works in a certain way to prove it so. I understand that you suspect that the SELECT INTO + IDENTITY + ORDER BY (call it Technique 1) variation might produce identity values that do not reflect the sort, but think it's wrong of INSERT INTO + ORDER BY into table with IDENTITY (call it Technique 2) to produce identity values that don't reflect the sort. While we're at it, many programmers also believe that SELECT INTO + IDENTITY FROM (SELECT + TOP 100 PERCENT + ORDER BY) (call it Technique 3) generates identity values that reflect the desired sort.

I just generated a contradicting (on my machine) example for the SELECT INTO statement:

```
set nocount on
use tempdb
go
drop table ta, tb, #t
go
create table ta(a int not null)
insert into ta values(3)
insert into ta values(1)
```

```
insert into ta values(2)
```

```
create table tb(b int not null primary key)
```

```
insert into tb values(4)
```

```
insert into tb values(1)
```

```
insert into tb values(3)
```

```
insert into tb values(2)
```

```
select b, identity(int, 1, 1) as rn
```

```
into #t
```

```
from ta join tb on a = b
```

```
order by b
```

```
select * from #t order by b
```

```
-- Output:
```

```
b rn
```

```
-----
```

```
1 2
```

```
2 3
```

```
3 1
```

```
-- Plan:
```

```
|--Table Insert(OBJECT:([#t]), SET:([#t].[rn]=[Expr1005],
```

```
#[t].[b]=[tb].[b]))
```

```
|--Top(ROWCOUNT est 0)
```

```
|--Compute Scalar(DEFINE:([Expr1005]=setidentity([Expr1004], -7, 0, '#t')))
```

```
|--Sort(ORDER BY:([tb].[b] ASC))
```

```
|--Compute Scalar(DEFINE:([Expr1004]=getidentity(-7, 0, '#t')))
```

```
|--Nested Loops(Inner Join, OUTER REFERENCES:([ta].[a]))
```

```
|--Table Scan(OBJECT:([tempdb].[dbo].[ta]))
```

```
|--Clustered Index
```

```
Seek(OBJECT:([tempdb].[dbo].[tb].[PK__tb__522A84E8]),
```

```
SEEK:([tb].[b]=[ta].[a]) ORDERED FORWARD)
```

You can see in the plan that the `getidentity()` function is invoked prior to the sort. That's the case on my machine (SQL2K Dev / SP3, single CPU) with my data. I don't know how this code would behave on other machines and with other data. But that was exactly my point.

But you mentioned that you can accept that Technique 1 is not guaranteed, but not Technique 2. Why? Because you assume that the `SELECT` must be first fully processed and then loaded into the table? There's nothing in the relational model or ANSI SQL to support this because such an `INSERT` is non-relational and non-standard. So you expect an undocumented proprietary feature to behave in a certain way because that's what seems right to you. You can clearly see in the plan that the `Compute Scalar` operator with the `getidentity()` function is invoked before the data is actually loaded, so why doesn't it make sense that it might be invoked before the sort?

The way I see it, if Technique 1 doesn't guarantee anything, I'd be very

reluctant to assume Technique 2 does without Microsoft explicitly guaranteeing it.

I'm afraid I currently don't have contradicting examples for Techniques 2 and 3, and since I'm at home, I don't have access to machines with multiple CPUs.

I'll keep looking for ones, and if someone has contradicting examples, I'd be glad to hear about those.

I posted a request in the private MVP forum to get Microsoft's official stand on the subject for all variations of the techniques. I'll post any info that can be shared publicly.

Cheers,

```
--
BG, SQL Server MVP
www.SolidQualityLearning.com
"Questar" <Questar@discussions.microsoft.com> wrote in message
news:7A556581-9D66-4E2E-8634-CA4E1342AE44@microsoft.com...
> It has come to our attention that the expected (and observed) behavior of
> a
> specific T-SQL statement is not guaranteed. You have probably encountered
> the issue before, but I will provide two examples here.
>
> Syntax Example 1:
>
> CREATE TABLE #T1 (T1 int, Z1 varchar(20))
>
> INSERT #T1 VALUES (1,'X')
> INSERT #T1 VALUES (2,'Y')
> INSERT #T1 VALUES (3,'Z')
>
> SELECT T1 AS T2, Z1 AS Z2, IDENTITY(int,1,1) AS I2 INTO #T2 FROM #T1 ORDER
> BY T1 DESC
>
> DROP TABLE #T1
> DROP TABLE #T2
>
> Syntax Example 2:
>
> CREATE TABLE #T1 (T1 int, Z1 varchar(20))
> CREATE TABLE #T2 (T2 int, Z2 varchar(20), I2 int IDENTITY(1,1))
>
> INSERT #T1 VALUES (1,'X')
> INSERT #T1 VALUES (2,'Y')
> INSERT #T1 VALUES (3,'Z')
>
> INSERT #T2 (T2,Z2) SELECT T1,Z1 FROM #T1 ORDER BY T1 DESC
>
> DROP TABLE #T1
> DROP TABLE #T2
>
> According to a SQL Server MVP, the SQL Server 2000 execution plans for
> these
> two examples are not guaranteed to have the sort occur before the
> assignment
> of IDENTITY values. The MVP said the expected behavior was guaranteed in
> SQL
```

## microsoft.public.sqlserver.programming: Re: ORDER BY and IDENTITY

> Server 2005. These two examples produce very similar, but not identical,  
> execution plans. Looking at the statements logically I could understand  
> if  
> the syntax of Example 1 resulted in the Compute Scalar (IDENTITY)  
> operation  
> happening before the Sort operation, but that behavior would seem very  
> wrong  
> for Example 2. I believe there's a significant amount of T-SQL code being  
> used (including some of ours) that depends upon the syntax of Example 2  
> resulting in the Sort operation happening before the Compute Scalar  
> (IDENTITY) operation.  
>  
> Does anybody have any additional thoughts or information on this matter?  
>