

Re: Order of execution in logical expressions

Source:

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.tools/2007-06/msg00020.html>

- *From:* "Russell Fields" <russellfields@xxxxxxxxxx>
 - *Date:* Mon, 4 Jun 2007 13:30:31 -0400
-

Wayne,

The answer is: The optimizer decides and it can change its mind at a later time, depending on statistics, indexes, et cetera, when to evaluate a clause in T-SQL. If you hide things in scalar or multi-statement table-valued functions you can also force evaluation order. (But not for in-line table valued functions.)

Your TOP 1 fix could have also been fixed by MIN or MAX, for the same reasons. It tells the optimizer that only one value will return.

RLF

"Wayne Erfling" <wayne_erfling@xxxxxxxxxxxxxxxx> wrote in message news:eNEgrGspHHA.3524@xxxxxxxxxxxxxxxxxxxxxxxxxxxx
GlacierI'm a little confused about order of execution in logical expressions.

In a statement like this:

```
SELECT * FROM myTable
WHERE PK_ID > 0 OR
(PK_ID <=0 AND 1/0 > 5)
```

apparently statement compilation jumps in and detects the divide by zero error, even though that clause would never be executed, because all PK_IDs in myTable are in fact > 0.

On the other hand, if I "hide" the divide by zero, then Transact-SQL seems to use what the .NET world calls "partial statement evaluation" (we called it something else in the old COBOL through Pascal days):

```
SELECT * FROM myTable
WHERE PK_ID > 0 OR
(PK_ID <=0 AND 1/myFunction(5) > 5)
```

In this case, myFunction(5) would return zero if called, but as this is apparently out of reach of the statement compiler, the query runs fine and

Re: Order of execution in logical expressions

returns all rows.

This question arose because of a sub-query in an UPDATE TRIGGER:

....

```
IF NOT UPDATE(fieldname) OR  
(SELECT DISTINCT fieldname FROM INSERTED) IS NULL)
```

For a multiple-row update, at execution time this statement returned an error about multiple rows being returned from a sub-query.

However, if the statement is being executed using "partial statement evaluation", then the only time the second clause should be executed is if UPDATE(fieldname) is TRUE.

When UPDATE(fieldname) is TRUE there can only be one value for fieldname, the value supplied in the UPDATE statement.

Therefore the sub-query error is inappropriate.

(Incidentally, I "fixed" it by replacing DISTINCT with TOP 1, even though DISTINCT could not have returned more than one value in this case).

I'm wondering if somebody can help me understand better when Transact-SQL will and will not execute logical clauses using AND and OR, as it seems unpredictable to me, and in the context of a WHERE clause, I don't have the luxury of using nested IF statements as one has to do in at least some members of the VB family (VBScript, VBA, VB, VB.Net):

Thanks!

---Wayne

.