

## Re: Overcoming Transact-SQL ORDER BY limitations

**Source:**

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.programming/2004-03/6064.html>

---

**From:** Kenneth Birecki (*kbirecki\_at\_hotmail.com*)

**Date:** 03/24/04

Date: Tue, 23 Mar 2004 22:43:00 -0500

Quentin,

That's a good point, dealing with NULL's. There is an article in MS' web site (<http://support.microsoft.com/default.aspx?scid=kb%3BEN-US%3Bq294942>) that suggests not using this suggestion, but to use ISNULL instead. I need to do some performance testing tomorrow on this concept since I just finished a functional stored procedure at the end of the day. I'll include a variation to examine the impact of dealing with NULL's. Thanks for the feedback!

Ken

"Quentin Ran" <ab@who.com> wrote in message  
news:uG%23f6kSEEHA.628@TK2MSFTNGP10.phx.gbl...

> *Nice plot, Ken.*

>

> *Remember to add*

>

> *set CONCAT\_NULL\_YIELDS\_NULL OFF*

>

> *before the select statement if you allow null for the candidate order by  
> columns.*

>

> *Quentin*

>

>

> *"news.microsoft.com" <kbirecki@applied-handling.com> wrote in message  
> news:OYcUaVSEEHA.576@TK2MSFTNGP11.phx.gbl...*

>> *This is FYI for anyone interested. I was looking for a way to pass*

>> *parameters to a stored procedure to determine the order of the resulting*

>> *table. I wanted to put parameters in the ORDER BY clause, such as:*

>>

>> *SELECT \* FROM tblTable ORDER BY @p\_OrderField*

>>

>> *This gives an error such as "The SELECT item identified by the ORDER BY*

>> *number 1 contains a variable*

>> *as part of the expression identifying a column position.*

> > Variables are only allowed when ordering by an expression referencing a  
> > column name."  
> >  
> > No variation of this seems to work and has been discussed by many people  
> > on  
> > the Internet. One variation that was suggested was to use a CASE  
> > statement  
> > in the ORDER BY clause, such as:  
> >  
> > SELECT \* FROM tblTable ORDER BY CASE @p\_OrderField WHEN 'Field1' THEN  
> > Field1  
> > WHEN 'Field2' THEN Field2 END  
> >  
> > The problem I had was that I wanted to sort on multiple fields,  
> > @p\_OrderField1 and @p\_OrderField2. Using the CASE concept above, it  
would  
> > look like this:  
> >  
> > SELECT \* FROM tblTable ORDER BY CASE @p\_OrderField1 WHEN 'Field1' THEN  
> > Field1 WHEN 'Field2' THEN Field2 END, CASE @p\_OrderField2 WHEN 'Field1'  
> > THEN  
> > Field1 WHEN 'Field2' THEN Field2 END  
> >  
> > But that did not work either. This caused an error saying that any  
field  
> > can only be referenced one time. Since there were now two ORDER BY  
fields  
> > using the same table fields more than once, that makes sense. So that  
way  
> > was not going to work either.  
> >  
> > And then the brainstorm hit. I could use a calculated field and sort on  
> > that field. And that would allow me to include as many fields in the  
sort  
> > process as I wanted. Here is the stored procedure that worked for me on  
> > my  
> > table.  
> >  
> > -----  
> > CREATE PROCEDURE qq4  
> > (  
> > @p\_Sort1 char(10),  
> > @p\_Sort2 char(10)  
> > )  
> >  
> > AS  
> >  
> > --Uncomment the following for debugging purposes  
> > /\*  
> > Declare @v\_SortResult char(100)  
> >

```
>> SELECT @v_SortResult =
>> CASE @p_Sort1 WHEN 'EUCustomer' THEN 'EUCustomer'
>> WHEN 'Seller' THEN 'Seller'
>> WHEN 'JobNo' THEN Cast(34567 as char(6))
>> WHEN 'JobDesc' THEN 'JobDesc'
>> END +
>> CASE @p_Sort2 WHEN 'EUCustomer' THEN 'EUCustomer'
>> WHEN 'Seller' THEN 'Seller'
>> WHEN 'JobNo' THEN Cast(34567 as char(6))
>> WHEN 'JobDesc' THEN 'JobDesc'
>> END
>>
>> Print @v_SortResult
>> */
>>
>> SELECT JobNo,
>> Seller,
>> JobDesc,
>> EUCustomer ,
>> CASE @p_Sort1 WHEN 'EUCustomer' THEN EUCustomer
>> WHEN 'Seller' THEN Seller
>> WHEN 'JobNo' THEN Cast(JobNo as varchar(6))
>> WHEN 'JobDesc' THEN JobDesc
>> END +
>> CASE @p_Sort2 WHEN 'EUCustomer' THEN EUCustomer
>> WHEN 'Seller' THEN Seller
>> WHEN 'JobNo' THEN Cast(JobNo as varchar(6))
>> WHEN 'JobDesc' THEN JobDesc
>> END AS
>> SortOrder
>> FROM tblJob
>> ORDER BY
>> SortOrder
>>
>>
>> RETURN
>> GO
>> -----
>>
>> In my table, JobNo is a bigint, and the other fields are string fields.
>> The
>> brainstorm I had was to use CASE statements to concatenated fields based
>> on
>> the parameters passed, and that results in a single (long) string, which
>> can
>> be the single field referred to in the ORDER BY clause.
>>
>> Hope this helps someone.
>> Ken
>>
>>
```

microsoft.public.sqlserver.programming: Re: Overcoming Transact-SQL ORDER BY limitations

>  
>