

# Silent performance killer with promoted column

---

*Source:*

<http://www.tech-archive.net/Archive/SOL-Server/microsoft.public.sqlserver.xml/2008-03/msg00004.html>

---

- *From:* "Marc Gravell" <[marc.gravell@xxxxxxxx](mailto:marc.gravell@xxxxxxxx)>
  - *Date:* Thu, 6 Mar 2008 09:02:41 -0000
- 

Half open question, half dire warning ;-p

[I'm kinda hoping Mr. Rys might perhaps spot this one?]

I got stung yesterday by a curious performance issue on 2005 (I haven't tried it on 2008) – basically, it involves a promoted, persisted and indexed field from an xml column. When the SET statements are all correct, it works great. But get any wrong, and it just grinds, doing a table-scan and re-evaluating the expression instead of using the index.

I can broadly understand the "why" of this, but it seems curious that most inappropriate SET statements (with xml) raise an error (which I consider a /good/ thing), where-as this one silently degrades to a crawl.

Easy to avoid if you know about it (just make sure your SET statements are appropriate) – but very hard to spot if you don't know what to look for...

Any thoughts? Full script to reproduce is below.

Marc

/\*

First part of script creates table and data

Second part of script is performance tests

Typical output:

```
ANSI_NULLS ON, QUOTED_IDENTIFIER ON
Table 'FOO'. Scan count 1, logical reads 6 [snip]
ANSI_NULLS ON, QUOTED_IDENTIFIER OFF
Table 'FOO'. Scan count 1, logical reads 3527 [snip]
ANSI_NULLS OFF, QUOTED_IDENTIFIER ON
Table 'FOO'. Scan count 1, logical reads 3527 [snip]
ANSI_NULLS OFF, QUOTED_IDENTIFIER OFF
Table 'FOO'. Scan count 1, logical reads 3527 [snip]
ANSI_NULLS ON, QUOTED_IDENTIFIER ON
Table 'FOO'. Scan count 1, logical reads 6 [snip]
*/
```

## Silent performance killer with promoted column

```
__ ***** PART 1 *****
SET STATISTICS IO OFF
SET ANSI_NULLS ON
SET QUOTED_IDENTIFIER ON
SET NOCOUNT ON

IF OBJECT_ID('FOO') IS NOT NULL
DROP TABLE FOO
IF OBJECT_ID('Bar') IS NOT NULL
DROP FUNCTION Bar
GO
CREATE FUNCTION Bar (@xml xml) RETURNS varchar(50) WITH SCHEMABINDING
BEGIN
RETURN @xml.value('/*/@name[1]', 'varchar(50)')
END
GO
CREATE TABLE FOO(
[ID] int NOT NULL IDENTITY(1,1),
[Content] xml NOT NULL,
CONSTRAINT [PK_FOO] PRIMARY KEY CLUSTERED ([ID] ASC))

CREATE PRIMARY XML INDEX FOO_Content
ON FOO ([Content])

ALTER TABLE FOO ADD [Name] AS [dbo].Bar([Content]) PERSISTED

CREATE NONCLUSTERED INDEX FOO_Name ON FOO ([Name] ASC)

DECLARE @tmp TABLE ([ID] int IDENTITY(1,1) NOT NULL, [Name] varchar(50) NOT
NULL)
INSERT @tmp ([Name]) VALUES ('Fred')
INSERT @tmp ([Name]) VALUES ('Barney')
INSERT @tmp ([Name]) VALUES ('Wilma')
INSERT @tmp ([Name]) VALUES ('Betty')

INSERT @tmp ([Name])
SELECT t1.[Name] FROM @tmp t1 CROSS JOIN @tmp t2 CROSS JOIN @tmp t3
INSERT @tmp ([Name])
SELECT t1.[Name] FROM @tmp t1 CROSS JOIN @tmp t2 CROSS JOIN @tmp t3

INSERT FOO ([Content])
SELECT '<xml name="" + [Name] + CONVERT(varchar(10), [ID]) + ""/>' FROM @tmp

SELECT COUNT(1) AS [Rows] FROM FOO
GO
__ ***** PART 2 *****
SET NOCOUNT ON SET STATISTICS IO ON
SET ANSI_NULLS ON SET QUOTED_IDENTIFIER ON
PRINT 'ANSI_NULLS ON, QUOTED_IDENTIFIER ON'
DECLARE @Name varchar(50) SET @Name = 'Betty60000'
SELECT * FROM FOO WHERE [Name] = @Name
```

## Silent performance killer with promoted column

```
GO
SET NOCOUNT ON SET STATISTICS IO ON
SET ANSI_NULLS ON SET QUOTED_IDENTIFIER OFF
PRINT 'ANSI_NULLS ON, QUOTED_IDENTIFIER OFF'
DECLARE @Name varchar(50) SET @Name = 'Betty60000'
SELECT * FROM FOO WHERE [Name] = @Name
GO
SET NOCOUNT ON SET STATISTICS IO ON
SET ANSI_NULLS OFF SET QUOTED_IDENTIFIER ON
PRINT 'ANSI_NULLS OFF, QUOTED_IDENTIFIER ON'
DECLARE @Name varchar(50) SET @Name = 'Betty60000'
SELECT * FROM FOO WHERE [Name] = @Name
GO
SET NOCOUNT ON SET STATISTICS IO ON
SET ANSI_NULLS OFF SET QUOTED_IDENTIFIER OFF
PRINT 'ANSI_NULLS OFF, QUOTED_IDENTIFIER OFF'
DECLARE @Name varchar(50) SET @Name = 'Betty60000'
SELECT * FROM FOO WHERE [Name] = @Name
GO
-- repeat for good measure
SET NOCOUNT ON SET STATISTICS IO ON
SET ANSI_NULLS ON SET QUOTED_IDENTIFIER ON
PRINT 'ANSI_NULLS ON, QUOTED_IDENTIFIER ON'
DECLARE @Name varchar(50) SET @Name = 'Betty60000'
SELECT * FROM FOO WHERE [Name] = @Name
GO
```