http://www.sqlservercentral.com/articles/FMTONLY/64130/
Printed 2008/11/03 02:30PM

How SSIS and Other Tools Obtain Metadata

By Renato Buda, 2008/08/19

Introduction

Tools such as SSIS, Reporting Services and Crystal Reports need access to the column metadata of a resultset. As a designer, when you provide a data source (query or stored procedure) these tools show you the names and data types of the columns of the resultset. This allows you (the designer) to use the columns in the data flow or report you are creating. Have you noticed how quickly these tools can show the metadata of your query resultset? The metadata is usually displayed in less than a second even when the query can take minutes or even hours..

This article explains how SSIS, Reporting Services and other database access tools get quick access to metadata of your query resultsets. It also describes a problem that occurs if you use stored procedures with temporary tables in conjunction with these tools, and a way I devised to work around this problem.

The Secret of Fast Metadata

It might be know to some, but probably not widely known. When you are creating a Data Source in a Data Flow task in SSIS, you supply some SQL. SSIS executes the supplied SQL with "SET FMTONLY ON". This causes an empty (zero-row) resultset to be produced without actually executing the full query. It usually takes milliseconds. SSIS analyses the resultset and sets up the available columns for the Data Flow. The same technique is used by report-writing tools such as Crystal Reports and Cognos.

Suppose you have written a stored procedure to feed data to a data flow called ExportGraphicIndex. You create a Data Flow task in SSIS and add an OleDB Source component. You supply the SqlCommand property as "EXEC dbo.ExportGraphicIndex". Pretty quickly SSIS knows which columns are returned by this procedure. If you run a trace you will see something like the following. (Actually what you see is more complex because it uses undocumented procedures to prepare a statement handle and then execute the prepared statement, but the result is the same).

```
SET FMTONLY ON
EXEC dbo.ExportGraphicIndex;
SET FMTONLY OFF
```

The Problem

The use of the FMTONLY option can be a problem when the you want to use a stored procedure that creates and uses temp tables. With FMTONLY on, the temp table is not created and an error occurs when it is subsequently referenced. In SSIS you don't see the error - you just get no columns to work with in the designer.

The Usual Workarounds

Now, there are several ways around this:

1. Use table variables instead of temp tables
2. Use permanent tables instead of temp tables
3. SET FMTONLY OFF and live with the longer execution time

Discussion of the above solutions can be found in
http://www.sqlservercentral.com/articles/Integration+Services/61824/ and the related forum topic
http://www.sqlservercentral.com/Forums/Topic511676-148-1.aspx#bm549651

I think its best to use table variables if possible.

What if the Workarounds don't Work?

Sometime none of these solutions are desirable because

1. You want to index the temp table, but can't use a primary key
2. You don't want to pollute the database with permanent tables that are only used by one procedure
3. The procedure is too slow to execute at design time

I have come across this situation several times, once with a Crystal Report and twice with a SSIS. Only recently I tested the following solution and it worked well.

The temp tables can be created by testing for FMTONLY and creating the tables even when it is ON. The following code outlines how this is done.

```
CREATE PROCEDURE dbo.ExportGraphicIndex
AS
BEGIN

DECLARE @FmtOnlyTest int; -- If this is null then FMTONLY was ON
  SELECT @FmtOnlyTest = count(*) FROM sys.filegroups;
  IF @FmtOnlyTest IS NULL -- FMTONLY was ON
    SET FMTONLY OFF;

  -- Create your indexed temp tables here
  -- Use a temp table because it needs to be indexed (cant use PK)
  IF object_id('tempdb.dbo.#GraphicIndex1') IS NOT NULL
    DROP TABLE #GraphicIndex1;

  CREATE TABLE #GraphicIndex1(
    CatalogueID int NOT NULL,
    IndexLevel int NOT NULL,
    SectionCode varchar(10) NULL,
    Description varchar(max) NULL,
    BeginDate smalldatetime NULL,
    EndDate smalldatetime NULL,
  );
  CREATE CLUSTERED INDEX IX_GraphicIndex1 ON #GraphicIndex1
    (IndexLevel, CatalogueID, SectionCode);

  IF @FmtOnlyTest IS NULL -- FMTONLY was ON
```

```
    SET FMTONLY ON;
   -- Do the main work here

   INSERT INTO #GraphicIndex1(...)
   SELECT ....
   ...
END
```

You should also clean up the temp tables but I did not illustrate this for brevity.

Conclusion

The FMTONLY option is not often used in normal DBA and programming work, but it is used behind the scenes by many tools when they obtain metadata. The knowledge presented in this article may help in troubleshooting and overcoming some problems that you encounter.