EXECUTE statement in SQL Server 2012

#### di [Sergio Govoni](http://mvp.microsoft.com/profiles/Sergio.Govoni) - Microsoft MVP ([blog](http://www.ugiss.org/sgovoni/))

* 1. 

*Dicembre, 2012*

Introduzione

In questo articolo parleremo del nuovo gruppo di opzioni RESULT SETS implementate per la clausola WITH del comando [EXECUTE](http://msdn.microsoft.com/en-us/library/ms188332(v=SQL.110).aspx) in SQL Server 2012.

Quante volte avete avuto la necessità di utilizzare il result-set restituito, ad esempio, da una stored procedure? E quante volte il result-set restituito era esattamente come ve lo aspettavate? Pensando alle risposte a queste domande probabilmente vi verranno in mente le situazioni in cui avete dovuto “sistemare” il result-set restituito in output dal comando EXECUTE, e uno dei workaround più tipici per risolvere il problema è quello di scaricare tale result-set in una tabella temporanea per poterlo adeguare alle vostre esigenze. L’opzione RESULT SETS garantisce metadati specifici per i result-set di una stored procedure o di un batch T-SQL implementando un “contratto” tra il comando EXECUTE (chiamante) e la relativa stored procedure o batch T-SQL (chiamato).

Le opzioni supportate da RESULT SETS

RESULT SETS supporta le seguenti opzioni:

* + UNDEFINED
  + NONE
  + (<result\_set\_definition>)

che descriviamo di seguito.

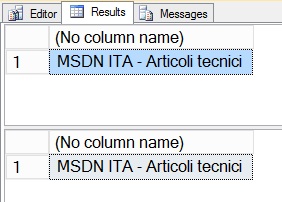
RESULT SETS UNDEFINED

L’opzione RESULT SETS UNDEFINED rappresenta il default; specificando questa opzione oppure omettendo RESULT SETS, il comando EXECUTE mantiene il comportamento che aveva nelle precedenti edizioni di SQL Server ovvero fino all’edizione 2008 R2. By default, quindi, lo statement EXECUTE non fornisce garanzie sui metadati restituiti dall’esecuzione di una stored procedure o di un batch T-SQL.

Nel seguente frammento di codice si osserva che i due statement EXECUTE restituiscono lo stesso output dimostrando che l’opzione RESULT SETS UNDEFINED rappresenta il comportamento di default.

* 1. USE [tempdb];
  2. GO
  3. -- RESULT SETS UNDEFINED (default)
  4. EXECUTE('SELECT ''MSDN ITA - Articoli tecnici''');
  5. GO
  6. EXECUTE('SELECT ''MSDN ITA - Articoli tecnici''')
  7. WITH RESULT SETS UNDEFINED;
  8. GO

L’output è illustrato in Figura 1.

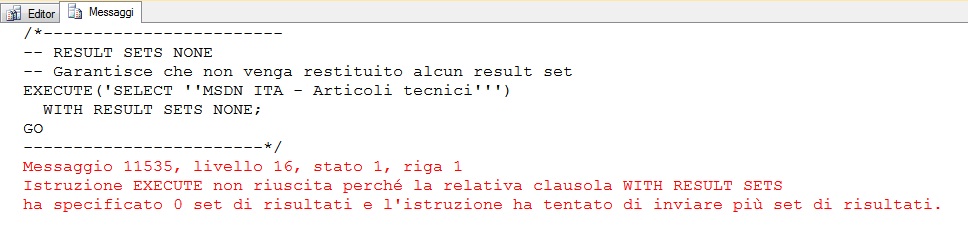
* 1. 
  2. Figura 1 – L’opzione RESULT SETS UNDEFINED rappresenta il default

RESULT SETS NONE

L’opzione RESULT SETS NONE garantisce che non vengano restituiti result-set per il comando EXECUTE; verrà generato un errore qualora la stored procedure o il batch T-SQL invocati con EXECUTE tentino la restituzione di un result-set. Nel seguente frammento di codice viene utilizzata l’opzione RESULT SETS NONE per garantire al chiamante che nessun result-set venga restituito.

* 1. USE [tempdb];
  2. GO
  3. -- RESULT SETS NONE
  4. -- Garantisce che non venga restituito alcun result-set
  5. EXECUTE('SELECT ''MSDN ITA - Articoli tecnici''')
  6. WITH RESULT SETS NONE;
  7. GO

L’output è illustrato in Figura 2, il batch T-SQL tenta di restituire al chiamante una colonna con il testo “MSDN ITA - Articoli tecnici”, ma vista la presenza dell’opzione RESULT SETS NONE, che garantisce non vengano restituiti result-set, l’esecuzione verrà respinta con un errore.

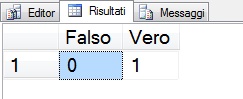
* 1. 
  2. Figura 2 – L’opzione RESULT SETS NONE garantisce che non vengano restituiti result-set

RESULT SETS ({…})

L’opzione RESULT SET (<result\_set\_definition>) permette di definire esattamente i metadati attesi per tutti i result-set restituiti dal comando EXECUTE; garantisce che i metadati restituiti siano in accordo, per numero e tipo, con quelli definiti. Nel seguente frammento di codice, vengono definiti metadati specifici per l’unico result-set restituito dallo statement EXECUTE.

* 1. USE [tempdb];
  2. GO
  3. -- RESULT SETS ({...})
  4. -- Definisce i metadati per uno o più result-set
  5. -- Garantisce che i metadati restituiti siano in accordo con quelli attesi
  6. EXECUTE('SELECT 0 AS False, 1 AS True')
  7. WITH RESULT SETS
  8. (
  9. (
  10. [Falso] SMALLINT
  11. ,[Vero] SMALLINT
  12. )
  13. );
  14. GO

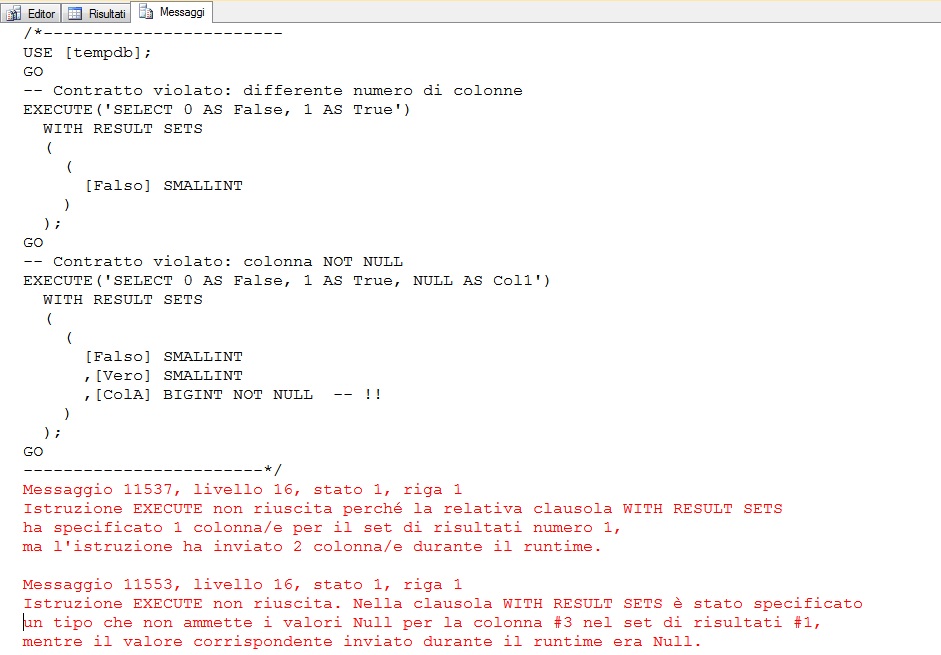
L’output è illustrato in figura 3.

* 1. 
  2. Figura 3 – L’opzione RESULT SET ({…}) definisce esattamente i metadati per il result-set

Nel prossimo esempio vedremo cosa accade quando si tenta di violare il “contratto”, stabilito con l’opzione RESULT SET, tra chiamate e chiamato che individuiamo rispettivamente nel comando EXECUTE e nell’oggetto eseguito, sia esso una stored procedure o un batch T-SQL.

* 1. USE [tempdb];
  2. GO
  3. -- Contratto violato: differente numero di colonne
  4. EXECUTE('SELECT 0 AS False, 1 AS True')
  5. WITH RESULT SETS
  6. (
  7. (
  8. [Falso] SMALLINT
  9. )
  10. );
  11. GO
  12. -- Contratto violato: colonna NOT NULL
  13. EXECUTE('SELECT 0 AS False, 1 AS True, NULL AS Col1')
  14. WITH RESULT SETS
  15. (
  16. (
  17. [Falso] SMALLINT
  18. ,[Vero] SMALLINT
  19. ,[ColA] BIGINT NOT NULL -- !!
  20. )
  21. );
  22. GO

L’output è illustrato in Figura 4.

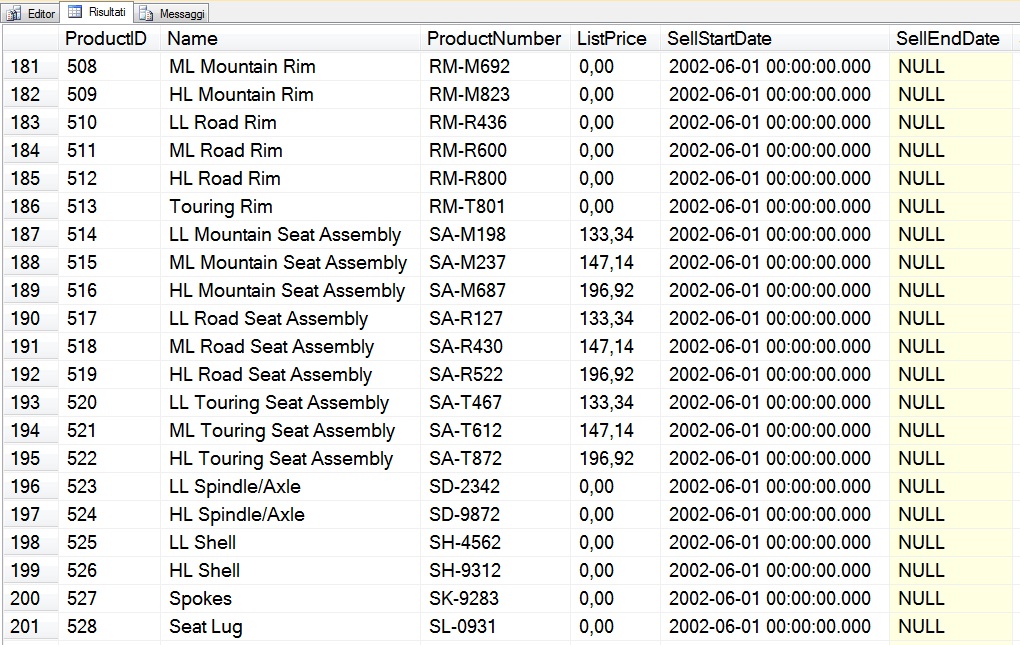
* 1. 
  2. Figura 4 – Violazione del “contratto” definito tra chiamante e chiamato

Entrambi gli statement in Figura 4, sono stati respinti; il primo statement ha restituito un result-set composto da due colonne, la colonna “False” e la colonna “True”, attraverso l’opzione RESULT SETS, la prima colonna è stata rinominata in “Falso” e il relativo tipo di dato convertito in SMALLINT, della seconda colonna però non vi è traccia nella definizione del risultato, da qui la violazione del “contratto” e la generazione dell’errore. Nel secondo statement, invece, il numero delle colonne attese è uguale al numero delle colonne restituite, una di essere però, la colonna “ColA” è definita NOT NULL mente il batch eseguito tenta proprio la restituzione di un NULL, anche in questo caso il “contratto” è stato violato.

EXECUTE … WITH RESULT SETS in azione

Nel prossimo esempio, dopo averla creata, utilizzeremo la stored procedure Production.uspGetProductForSell che interroga la tabella Production.Product del database di esempio [AdventureWorks2012](http://msftdbprodsamples.codeplex.com/releases/view/55330). La stored procedure può restituire in output due result-set; il primo result-set, sempre presente, è rappresentato dai prodotti vendibili, dall’azienda, alla data passata nel parametro @vDate. Il secondo result-set, la cui presenza è condizionata da valori maggiori di zero nel parametro @vPrice, restituisce i prodotti vendibili, dall’azienda, alla data passata nel parametro @vDate e che hanno Prezzo di Listino minore o uguale al valore passato in @vPrice.

Prendiamo confidenza con i dati memorizzati nella tabella Production.Product del database AdventureWorks2012, la Figura 5 illustra alcuni dei record presenti in tabella.

* 1. 
  2. Figura 5 – Dati contenuti nella tabella Production.Product

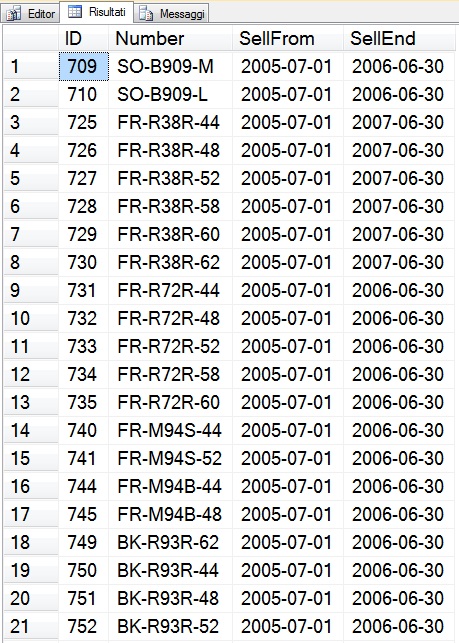
Definiamo ora la stored procedure Production.uspGetProductForSell utilizzata dagli operatori commerciali per conoscere i prodotti vendibili in un determinato momento. Come accennato precedentemente, la stored procedure accetta in input due parametri: @vDate rappresenta il giorno per il quale si desidera conoscere i prodotti vendibili e @vPrice rappresenta il parametro di filtro sulla colonna ListPrice. Il seguente frammento di codice implementa la CREATE della stored procedure nel database di esempio AdventureWorks2012.

* 1. USE [AdventureWorks2012];
  2. GO
  3. CREATE PROCEDURE Production.uspGetProductForSell
  4. (
  5. @vDate AS DATE
  6. ,@vPrice AS DECIMAL(8, 2) = 0
  7. )
  8. AS
  9. BEGIN
  10. /\*
  11. Stored procedure di esempio
  12. Restituisce i prodotti vendibili alla data passata
  13. come parametro
  14. \*/
  15. IF (ISNULL(@vDate, '') = '')
  16. BEGIN
  17. PRINT ('Errore: Valore del parametro @vDate non valido')
  18. RETURN
  19. END
  20. -- Primo result-set
  21. -- Prodotti vendibili alla data passata come parametro
  22. SELECT
  23. ProductID
  24. ,ProductNumber
  25. ,SellStartDate
  26. ,SellEndDate
  27. FROM
  28. Production.Product
  29. WHERE
  30. (@vDate BETWEEN SellStartDate AND SellEndDate)
  31. -- Secondo result-set
  32. -- Prodotti vendibili alla data passata come parametro
  33. -- con prezzo di listino minore o uguale al valore
  34. -- passato come parametro
  35. IF (ISNULL(@vPrice, 0) > 0)
  36. BEGIN
  37. SELECT
  38. ProductID
  39. ,ListPrice
  40. FROM
  41. Production.Product
  42. WHERE
  43. (ListPrice <= @vPrice)
  44. AND (@vDate BETWEEN SellStartDate AND SellEndDate)
  45. END
  46. END;

Eseguiamo ora la stored procedure Production.uspGetProductForSell per ottenere in output gli articoli vendibili al 31/12/2005, utilizzeremo il comando EXECUTE specificando la clausola RESULT SETS per assegnare un nome ed un tipo di dato specifico ad ogni colonna restituita.

* 1. EXECUTE Production.uspGetProductForSell
  2. @vDate = '20051231'
  3. WITH RESULT SETS
  4. (
  5. -- Primo result-set
  6. -- Definizione dei metadati
  7. (
  8. ID VARCHAR(20)
  9. ,Number VARCHAR(40)
  10. ,SellFrom DATE
  11. ,SellEnd DATE
  12. )
  13. );

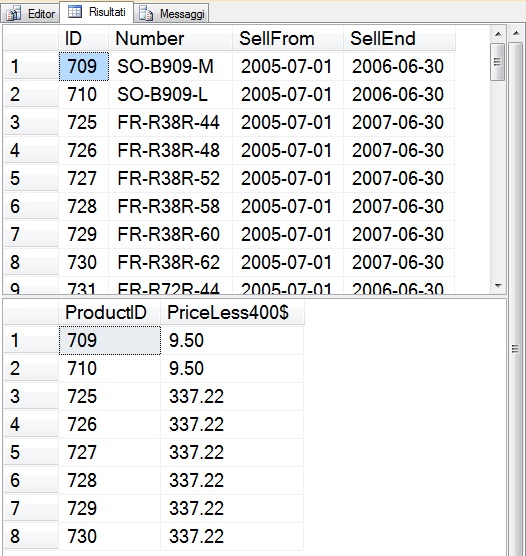
L’output ottenuto è illustrato in Figura 6 dove si può osservare come le colonne della query eseguita all’interno della stored procedure siano state rinominate in accordo con quanto specificato nel comando EXECUTE.

* 1. 
  2. Figura 6 – Prodotti vendibili al 31/12/2005, metadati specifici per il risultato

Oltre agli articoli vendibili ad una certa data, nel prossimo esempio desideriamo ricevere in output anche il secondo un result-set, che deve contenere, come nel caso del primo, gli articoli vendibili alla data passata in @vDate, ma che hanno Prezzo di Listino minore o uguale al valore passato nel parametro @vPrice. Nel prossimo esempio, illustriamo come sia possibile definire nome delle colonne e tipi di dato, anche per il secondo result-set (con la stessa logica mutuata per il primo).

* 1. EXECUTE Production.uspGetProductForSell
  2. @vDate = '20051231'
  3. ,@vPrice = 400.00
  4. WITH RESULT SETS
  5. (
  6. -- Definizione dei metadati
  7. -- Primo result-set
  8. (
  9. ID VARCHAR(20)
  10. ,Number VARCHAR(40)
  11. ,SellFrom DATE
  12. ,SellEnd DATE
  13. ),
  14. -- Secondo result-set
  15. (
  16. ProductID VARCHAR(20)
  17. ,PriceLess400$ DECIMAL(8, 2)
  18. )
  19. );

L’output ottenuto è illustrato in Figura 7.

* 1. 
  2. Figura 7 – Prodotti vendibili al 31/12/2005 con prezzo di listino minore di 400,00$, metadati specifici

Pulizia database

* 1. USE [AdventureWorks2012];
  2. GO
  3. IF OBJECT\_ID('Production.uspGetProductForSell', 'P') IS NOT NULL
  4. DROP PROCEDURE Production.uspGetProductForSell;
  5. GO

Conclusioni

L’opzione RESULT SETS implementata per il comando EXECUTE in SQL Server 2012, permette di definire metadati specifici per tutti i result-set restituiti; implementa un “contratto” tra il comando EXECUTE (chiamante) e i risultati restituiti da stored procedure o batch T-SQL (chiamato); l’Engine di SQL Server respingerà l’esecuzione dello statement qualora il “contratto” venga violato.

#### di [Sergio Govoni](http://mvp.microsoft.com/profiles/Sergio.Govoni) - Microsoft MVP ([blog](http://www.ugiss.org/sgovoni/))

* 1. [*Altri articoli di Sergio Govoni nella Libr*](http://sxp.microsoft.com/feeds/3.0/msdntn/TA_MSDN_ITA?contenttype=Article&author=Sergio%20Govoni)*ary* [](http://sxp.microsoft.com/feeds/3.0/msdntn/TA_MSDN_ITA?contenttype=Article&author=Sergio%20Govoni)