Lo Statement MERGE e i Trigger

#### di [Sergio Govoni](http://mvp.microsoft.com/en-us/mvp/Sergio%20Govoni-4029181) - Microsoft MVP

Blog: <http://www.ugiss.org/sgovoni/>

Twitter: <https://twitter.com/segovoni>

* 1. 

*Aprile, 2013*

Introduzione

Abbiamo parlato dello statement MERGE nel precedente articolo [Come usare lo Statement MERGE (T-SQL)](http://msdn.microsoft.com/it-it/library/jj973188), ora tratteremo un approfondimento che ci permetterà di comprendere il comportamento di eventuali Trigger attivi sulla tabella (destinazione) oggetto delle operazioni DML specificate nelle clausole del comando MERGE. In particolare, risponderemo alle seguenti domande: Come viene regolata l’attivazione dei Trigger definiti sulla tabella di destinazione? L’esecuzione delle azioni specificate nelle clausole del comando MERGE scatena l’attivazione di tali Trigger come una qualsiasi operazione DML?

MERGE Trigger

SQL Server non supporta i MERGE Trigger, tuttavia qualora esistano oggetti Trigger definiti sulla tabella destinazione di un comando MERGE, l’esecuzione delle azioni specificate nelle clausole del comando scatena l’attivazione di tali Trigger come una qualsiasi operazione DML.

Esiste però una particolarità interessante: verrà scatenata una sola occorrenza di ogni Trigger, in risposta al comando MERGE, anche in presenza di esecuzioni multiple della stessa azione da parte di quest’ultimo. Come sempre, i Trigger avranno accesso alle tabelle virtuali Inserted e Deleted che conterranno tutte le righe modificate dallo statement. Dimostreremo questa particolarità del comando MERGE utilizzando lo scenario di esempio già precedentemente descritto nell’articolo [Come usare lo Statement MERGE (T-SQL)](http://msdn.microsoft.com/it-it/library/jj973188).

Scenario OLTP: Inventario di Magazzino

Ipotizziamo di eseguire l’inventario a rotazione, dei prodotti giacenti a magazzino e di dover aggiornare le giacenze logiche, memorizzate nella tabella dbo.ProductInventory del database AdventureWorks2012, con le giacenze fisiche rilevate dagli operatori durante l’inventario. Il comando MERGE sarà di grande aiuto per aggiornare i dati contenuti nella tabella dbo.ProductInventory (destinazione). Si dovrà quindi prevedere di:

* + Aggiornare le giacenze e Inserire, nella tabella dbo.ProductInventory, eventuali nuovi prodotti rilevati a magazzino (e non presenti nella destinazione)
  + Eliminare i prodotti non giacenti (giacenza uguale a zero)
  + Aggiornare la colonna Status per i prodotti non rilevati (prodotti nella tabella di destinazione che non hanno corrispondenza nella sorgente)

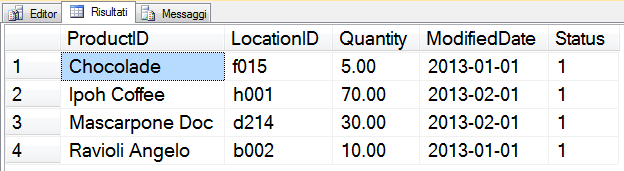
Procediamo con la creazione della tabella dbo.ProductInventory sul database AdventureWorks2012 (scaricabile da [CodePlex](http://msftdbprodsamples.codeplex.com/releases/view/55330)) e inseriamo alcuni dati di prova.

* 1. USE [AdventureWorks2012];
  2. GO
  3. ------------------------------------------------------------------------
  4. -- Creazione tabella dbo.ProductInventory e inserimento dati
  5. -- di prova
  6. ------------------------------------------------------------------------
  7. -- dbo.ProductInventory
  8. IF OBJECT\_ID('dbo.ProductInventory', 'U') IS NOT NULL
  9. DROP TABLE dbo.ProductInventory;
  10. GO
  11. CREATE TABLE dbo.ProductInventory
  12. (
  13. ProductID VARCHAR(25) NOT NULL
  14. ,LocationID VARCHAR(20) NOT NULL
  15. ,Quantity DECIMAL(9, 2) NOT NULL
  16. ,ModifiedDate DATE NOT NULL DEFAULT GETDATE()
  17. ,Status BIT NOT NULL DEFAULT(1)
  18. ,constraint PK\_Product\_ProductID\_LocationID PRIMARY KEY(ProductID, LocationID)
  19. );
  20. INSERT INTO dbo.ProductInventory
  21. (
  22. ProductID
  23. ,LocationID
  24. ,Quantity
  25. ,ModifiedDate
  26. )
  27. VALUES
  28. ('Ravioli Angelo', 'b002', 10, '20130101'),
  29. ('Chocolade', 'f015', 5, '20130101'),
  30. ('Ipoh Coffee', 'h001', 70, '20130201'),
  31. ('Mascarpone Doc', 'd214', 30, '20130201');
  32. GO

Consultiamo i dati inseriti, essi rappresentano le giacenze logiche dei prodotti in magazzino:

* 1. USE [AdventureWorks2012];
  2. GO
  3. SELECT \* FROM dbo.ProductInventory;
  4. GO

L’output è illustrato in figura 1.

* 1. 
  2. Figura 1 – Giacenze logiche dei prodotti in magazzino

I Trigger definiti nella tabella di destinazione

Per analizzare il comportamento dei Trigger in presenza del comando MERGE, definiamo sulla tabella di destinazione dbo.ProductInventory i seguenti Trigger rispettivamente per le azioni INSERT, UPDATE e DELETE.

* 1. USE [AdventureWorks2012];
  2. GO
  3. ------------------------------------------------------------------------
  4. -- Creazione trigger sulla tabella dbo.ProductInventory
  5. ------------------------------------------------------------------------
  6. CREATE TRIGGER dbo.Tr\_ProductInventory\_INSERT ON dbo.ProductInventory
  7. AFTER INSERT
  8. AS
  9. PRINT 'INSERT detected on dbo.ProductInventory';
  10. GO
  11. CREATE TRIGGER dbo.Tr\_ProductInventory\_UPDATE ON dbo.ProductInventory
  12. AFTER UPDATE
  13. AS
  14. PRINT 'UPDATE detected on dbo.ProductInventory';
  15. GO
  16. CREATE TRIGGER dbo.Tr\_ProductInventory\_DELETE ON dbo.ProductInventory
  17. AFTER DELETE
  18. AS
  19. PRINT 'DELETE detected on dbo.ProductInventory';
  20. GO

Questi Trigger di esempio eseguiranno semplicemente il comando PRINT, in cui viene riportato il nome del Trigger e l’azione (INSERT, UPDATE o DELETE) rilevata.

Le rilevazioni inventariali

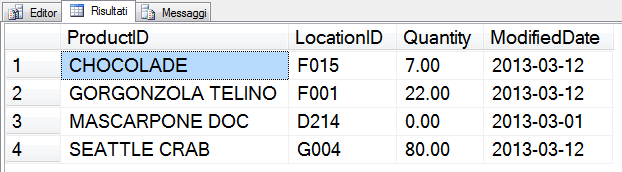
Ipotizziamo di raccogliere i dati inventariali nella tabella dbo.FrequentInventory, che possiamo creare e popolare con alcuni dati di prova (le rilevazioni inventariali) attraverso il seguente frammento di codice T-SQL.

* 1. USE [AdventureWorks2012];
  2. GO
  3. ------------------------------------------------------------------------
  4. -- Creazione tabella dbo.FrequentInventory e inserimento dei dati
  5. -- di prova che rappresentano le rilevazioni inventariali
  6. ------------------------------------------------------------------------
  7. IF OBJECT\_ID('dbo.FrequentInventory', 'U') IS NOT NULL
  8. DROP TABLE dbo.FrequentInventory;
  9. GO
  10. CREATE TABLE dbo.FrequentInventory
  11. (
  12. ProductID VARCHAR(25) NOT NULL
  13. ,LocationID VARCHAR(20) NOT NULL
  14. ,Quantity DECIMAL(9, 2) NOT NULL
  15. ,ModifiedDate DATE NOT NULL DEFAULT GETDATE()
  16. ,constraint PK\_FrequentInventory\_ProductID\_LocationID PRIMARY KEY(ProductID, LocationID)
  17. );
  18. go
  19. INSERT INTO dbo.FrequentInventory
  20. (
  21. ProductID
  22. ,LocationID
  23. ,Quantity
  24. ,ModifiedDate
  25. )
  26. VALUES
  27. ('CHOCOLADE', 'F015', 7, '20130312'),
  28. ('GORGONZOLA TELINO', 'F001', 22, '20130312'),
  29. ('SEATTLE CRAB', 'G004', 80, '20130312'),
  30. ('MASCARPONE DOC', 'D214', 0, '20130301');
  31. GO

Consultiamo le rilevazioni inventariali memorizzate nella tabella dbo.FrequentInventory.

* 1. USE [AdventureWorks2012];
  2. GO
  3. SELECT \* FROM dbo.FrequentInventory;
  4. GO

L’output è illustrato in figura 2.

* 1. 
  2. Figura 2 – Rilevazioni inventariali nella tabella dbo.FrequentInventory

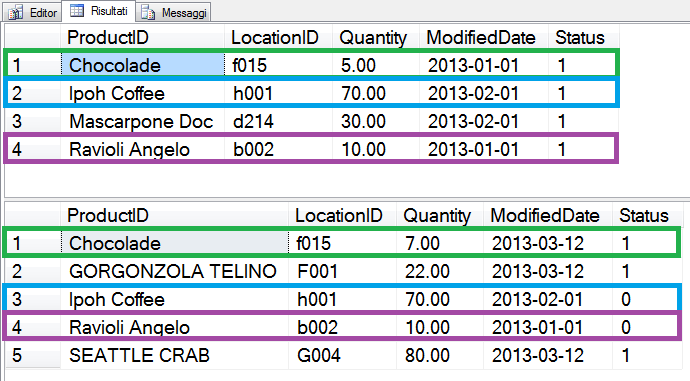
Il comportamento dei Trigger scatenati dalle azioni del comando MERGE

Applichiamo ora l’inventario attraverso l’esecuzione del seguente comando MERGE in cui sono state specificate tutte le clausole, compresa la clausola WHEN NOT MATCHED BY SOURCE proprietaria del linguaggio T-SQL.

Si noti che l’operazione di UPDATE sulla tabella di destinazione dbo.ProductInventory verrà eseguita sia dalla clausola WHEN MATCHED (per il prodotto “Chocolade” la cui giacenza viene aggiornata da 5 unità a 7 unità) che dalla clausola WHEN NOT MATCHED BY SOURCE (per i prodotti “Ipoh Coffee” e “Ravioli Angelo” che non sono stati rilevati durante l’inventario).

* 1. USE [AdventureWorks2012];
  2. GO
  3. ------------------------------------------------------------------------
  4. -- Applichiamo l'inventario utilizzando il comando MERGE e analizziamo
  5. -- il comportamento dei Trigger attivi sulla tabella di destinazione
  6. -- dbo.ProductInventory
  7. ------------------------------------------------------------------------
  8. BEGIN TRANSACTION;
  9. GO
  10. SELECT \* FROM dbo.ProductInventory;
  11. GO
  12. MERGE INTO
  13. dbo.ProductInventory AS itarget
  14. USING
  15. dbo.FrequentInventory AS isource
  16. ON ((itarget.ProductId = isource.ProductId)
  17. AND (itarget.LocationId = isource.LocationId))
  18. WHEN MATCHED AND
  19. (isource.Quantity <> 0)
  20. AND ((itarget.Quantity <> isource.Quantity)
  21. OR (itarget.ModifiedDate <> isource.ModifiedDate)) THEN
  22. UPDATE SET
  23. itarget.Quantity = isource.Quantity
  24. ,itarget.ModifiedDate = isource.ModifiedDate
  25. WHEN MATCHED AND
  26. (isource.Quantity = 0) THEN
  27. DELETE
  28. WHEN NOT MATCHED THEN
  29. INSERT
  30. (
  31. ProductId
  32. ,LocationId
  33. ,Quantity
  34. ,ModifiedDate
  35. )
  36. VALUES
  37. (
  38. isource.ProductId
  39. ,isource.LocationId
  40. ,isource.Quantity
  41. ,isource.ModifiedDate
  42. )
  43. WHEN NOT MATCHED BY SOURCE THEN
  44. UPDATE SET
  45. status = 0;
  46. GO
  47. SELECT \* FROM dbo.ProductInventory;
  48. GO
  49. ROLLBACK TRANSACTION;
  50. GO

L’output è illustrato in Figura 3.

* 1. 
  2. Figura 3 – Giacenze dei prodotti in magazzino prima e dopo l’inventario

Oltre all’aggiornamento dei dati, per effetto dei Trigger si ottiene anche il seguente output:

* 1. INSERT detected on dbo.ProductInventory
  2. UPDATE detected on dbo.ProductInventory
  3. DELETE detected on dbo.ProductInventory

Conclusioni

Il comando UPDATE è stato attivato ed eseguito in **due differenti clausole** del precedente comando MERGE, ma **ha provocato solo una attivazione del Trigger** dbo.Tr\_ProductInventory\_UPDATE per tutte le righe modificate.

#### di [Sergio Govoni](http://mvp.microsoft.com/en-us/mvp/Sergio%20Govoni-4029181) - Microsoft MVP

Blog: <http://www.ugiss.org/sgovoni/>

Twitter: <https://twitter.com/segovoni>