NOME CORSO

Fintech Software Developer

Unità Formativa (UF): Basi di dati SQL

Docente: Durando Giulio

Titolo argomento: I Trigger

Trigger

I trigger di SQL Server sono stored procedure speciali che vengono eseguite automaticamente in risposta all'oggetto del database, al database e agli eventi del server. SQL Server fornisce tre tipi di trigger:

- **Trigger DDL (Data Definition Language).** Questa classe di trigger si attiva in caso di eventi che modificano la struttura (come la creazione, la modifica o l'eliminazione di una tabella) o in determinati eventi relativi al server come le modifiche alla sicurezza o gli eventi di aggiornamento delle statistiche.
- **Trigger DML (Data Modification Language).** Questa è la classe di trigger più utilizzata. In questo caso l'evento di attivazione è un'istruzione di modifica dei dati; potrebbe essere un'istruzione di inserimento, aggiornamento o eliminazione su una tabella o una vista.
- Attivazioni di accesso che si attivano in risposta agli eventi **LOGON**

Inoltre, i trigger DML hanno diversi tipi:

- **FOR o AFTER [INSERT, UPDATE, DELETE]:** questi tipi di trigger vengono eseguiti al termine dell'istruzione di attivazione (inserimento, aggiornamento o eliminazione).
- **INSTEAD OF [INSTEAD, UPDATE, DELETE]:** contrariamente al tipo FOR (AFTER), i trigger INSTEAD OF vengono eseguiti invece dell'istruzione di attivazione. In altre parole, questo tipo di trigger sostituisce l'istruzione di attivazione. Ciò è molto utile nei casi in cui è necessario disporre dell'integrità referenziale di database incrociati.

Trigger LOGON

I trigger LOGON consentono di attivare stored procedure in risposta a un evento LOGON generato quando viene stabilita una sessione utente a un'istanza di SQL Server. I trigger LOGON vengono attivati dopo il completamento della fase di autenticazione della procedura di accesso, ma prima che la sessione utente venga effettivamente stabilita. Per questo motivo, tutti i messaggi generati all'interno del trigger che verrebbero normalmente visualizzati all'utente, come i messaggi di errore e i messaggi dall'istruzione PRINT, vengono invece indirizzati al log degli errori di SQL Server. I trigger LOGON non vengono attivati in caso di esito negativo dell'autenticazione.

È possibile utilizzare i trigger LOGON per controllare e gestire le sessioni server, ad esempio tenendo traccia delle attività di accesso, limitando gli accessi a SQL Server o limitando il numero di sessioni per uno specifico account di accesso. Nel codice seguente, ad esempio, il trigger LOGON nega i tentativi di accesso a SQL Server eseguiti dall'account di accesso *login_test* se esistono già tre sessioni utente in esecuzione create da tale account di accesso.

Trigger DML

Un trigger DML è un tipo speciale di stored procedure che diventa effettiva automaticamente quando viene eseguito un evento del linguaggio DML (Data Manipulation Language) che influisce sulla vista o tabella definita nel trigger. Gli eventi DML includono istruzioni INSERT, UPDATE o DELETE. I trigger DML possono essere utilizzati per applicare regole business e l'integrità dei dati, eseguire query su altre tabelle e includere istruzioni Transact-SQL complesse. Il trigger e l'istruzione che lo attiva vengono considerati come una singola transazione, di cui è possibile eseguire il rollback dal trigger stesso. Se viene rilevato un errore grave, ad esempio un'insufficienza di spazio su disco, viene eseguito automaticamente il rollback dell'intera transazione.

Vantaggi del trigger DML

I trigger DML sono simili ai vincoli in quanto sono in grado di applicare l'integrità di entità o di dominio. L'integrità di entità dovrebbe essere sempre applicata al livello più basso utilizzando indici che fanno parte dei vincoli PRIMARY KEY e UNIQUE oppure creati indipendentemente dai vincoli. Per applicare l'integrità di dominio è consigliabile utilizzare i vincoli CHECK, mentre per applicare l'integrità referenziale (RI) è consigliabile utilizzare i vincoli FOREIGN KEY. I trigger DML sono particolarmente utili quando le caratteristiche supportate dai vincoli non sono in grado di soddisfare le esigenze funzionali dell'applicazione.

L'elenco seguente confronta i trigger DML con i vincoli e identifica quando i trigger DML hanno vantaggi rispetto ai vincoli.

- I trigger DML consentono di propagare le modifiche nelle tabelle correlate del database, tuttavia è possibile eseguire le modifiche in modo più efficiente utilizzando vincoli di integrità referenziale di propagazione. I vincoli FOREIGN KEY consentono di convalidare un valore di colonna soltanto se corrisponde esattamente al valore di un'altra colonna, a meno che la clausola REFERENCES non definisca un'operazione referenziale di propagazione.
- Assicurano la protezione contro operazioni INSERT, UPDATE e DELETE dannose o non corrette e applicano altre restrizioni più complesse rispetto a quelle definite con i vincoli CHECK.

  A differenza dei vincoli CHECK, i trigger DML possono fare riferimento alle colonne di altre tabelle. Un trigger, ad esempio, può utilizzare un'istruzione SELECT di un'altra tabella per eseguire il confronto con i dati inseriti o aggiornati e per eseguire ulteriori operazioni, ad esempio la modifica di dati o la visualizzazione di un messaggio di errore definito dall'utente.
- Consentono di valutare lo stato di una tabella prima e dopo la modifica dei dati e di eseguire le operazioni appropriate sulla base delle differenze.
- Più trigger DML dello stesso tipo (INSERT, UPDATE o DELETE) in una tabella consentono di eseguire più operazioni diverse in risposta alla stessa istruzione di modifica.
- I vincoli sono in grado di segnalare errori soltanto tramite messaggi di errore standard di sistema. Se nell'applicazione è necessario o consigliabile utilizzare messaggi personalizzati e gestire gli errori in modo più complesso, è necessario utilizzare un trigger.
- I trigger DML impediscono di apportare modifiche che violano l'integrità referenziale o consentono di eseguirne il rollback, annullando in tal modo il tentativo di modifica dei dati. Un trigger di questo tipo potrebbe essere attivato quando si modifica una chiave esterna e il nuovo valore non corrisponde alla chiave primaria. A questo scopo tuttavia vengono normalmente utilizzati i vincoli FOREIGN KEY.
- Gli eventuali vincoli inclusi nella tabella di trigger vengono verificati dopo l'esecuzione del trigger INSTEAD OF e prima dell'esecuzione del trigger AFTER. In caso di violazione dei vincoli, viene eseguito il rollback delle azioni del trigger INSTEAD OF e il trigger AFTER non viene eseguito.

Creazione di un trigger DML

È possibile usare uno dei seguenti elementi:

- **SQL Server Management Studio**
- **Transact-SQL**

Creazione con SQL Server Management Studio

1. In **Esplora oggetti** connettersi a un'istanza del Motore di database , quindi espanderla.
2. Espandere **Database**, espandere **Tabelle** e quindi espandere la tabella su cui si vuole creare il trigger
3. Fare clic con il pulsante destro del mouse su **Trigger**, quindi scegliere **Nuovo trigger**.
4. Scegliere **Imposta valori per parametri modello** dal menu **Query**. In alternativa, è possibile premere (CTRL+MAIUSC+M) per aprire la finestra di dialogo **Imposta valori per parametri modello** .
5. Nella finestra di dialogo **Imposta valori per parametri modello** immettere i seguenti valori per i parametri indicati.
6. Fare clic su **OK**.
7. Nell' **Editor di query** scrivere la procedura che deve essere svolta dal trigger
8. Per verificare la validità della sintassi, scegliere **Analizza** dal menu **Query**. Se viene restituito un messaggio di errore, confrontare l'istruzione con le informazioni precedenti, apportare le modifiche necessarie e ripetere il passaggio.
9. Per creare il trigger DML, scegliere **Esegui** dal menu **Query**. Il trigger DML viene creato come un oggetto nel database.
10. Per visualizzare il trigger DML nell'elenco di Esplora oggetti, fare clic con il pulsante destro del mouse su **Trigger** e scegliere **Aggiorna**.

Creazione con Transact-SQL

1. In **Esplora oggetti** connettersi a un'istanza del Motore di database , quindi espanderla.
2. Nel menu **File** , fare clic su **Nuova query**.
3. Scrivere in linguaggio Transact-SQL la procedura relativa al trigger
4. Per verificare la validità della sintassi, scegliere **Analizza** dal menu **Query**. Se viene restituito un messaggio di errore, confrontare l'istruzione con le informazioni precedenti, apportare le modifiche necessarie e ripetere il passaggio.
5. Per creare il trigger DML, scegliere **Esegui** dal menu **Query**. Il trigger DML viene creato come un oggetto nel database.
6. Per visualizzare il trigger DML nell'elenco di Esplora oggetti, fare clic con il pulsante destro del mouse su **Trigger** e scegliere **Aggiorna**.

Modifica di un trigger DML

Quando si rinomina un trigger, il trigger deve trovarsi nel database corrente e il nuovo nome deve seguire le regole per gli identificatori.

Se si modifica il nome di un oggetto a cui fa riferimento un trigger DML, è necessario modificare il trigger in modo che il testo corrisponda al nuovo nome. Pertanto, prima di rinominare un oggetto, visualizzare le dipendenze dell'oggetto per determinare se la modifica proposta interessa i trigger.

Modifica di un trigger con SQL Server Management Studio

- **Per modificare un trigger DML**
  1. In **Esplora oggetti** connettersi a un'istanza del Motore di database , quindi espanderla.
  2. Espandere il database desiderato, espandere **Tabelle**, quindi espandere la tabella che contiene il trigger da modificare.
  3. Espandere **Trigger**, fare clic con il pulsante destro del mouse sul trigger da modificare, quindi scegliere **Modifica**.
  4. Modificare il trigger e fare clic su **Esegui**

- **Per rinominare un trigger DML**
  1. Eliminare il trigger che si desidera rinominare.
  2. Ricreare il trigger, specificando il nuovo nome.

Modifica di un trigger con Transact-SQL

1. Connettersi al Motore di database.
2. Dalla barra Standard fare clic su **Nuova query**.
3. Scrivere in linguaggio Transact-SQL la procedura relativa al trigger usando l'istruzione **ALTER TRIGGER**

Sintassi del trigger DML di SQL Server

La sintassi di base CREATE TRIGGER è la seguente

```
CREATE TRIGGER nome_trigger
ON { nome_tabella o nome_vista }
[ WITH <Options> ]
{ FOR | AFTER | INSTEAD OF }
{ [INSERT], [UPDATE] , [DELETE] }
```

La tabella successiva descrive ciascuno degli argomenti della sintassi CREATE TRIGGER.

Argomento	Descrizione
WITH <Options>	In questo argomento è possibile specificare opzioni aggiuntive per la creazione del trigger
FOR | AFTER | INSTEAD OF	Indica quando il trigger deve essere attivato quando si verifica un determinato evento, come un evento di inserimento, aggiornamento o eliminazione
[INSERT], [UPDATE] , [DELETE]	L'evento DML (o l'elenco di eventi) che provocherà l'attivazione del trigger.

Opzioni WITH	Descrizione	Osservazioni
ENCRYPTION	Crittografa il codice del Trigger.	Non funziona con le tabelle con ottimizzazione per la memoria
EXECUTE AS	Modifica il contesto di sicurezza su cui verrà eseguito il trigger	Necessario per i trigger su tabelle con ottimizzazione per la memoria.
NATIVE_COMPILATION	Compila il codice trigger in un binario per farlo funzionare in modo nativo.	Necessario per i trigger su tabelle con ottimizzazione per la memoria
SCHEMABINDING	Garantisce che le tabelle a cui fa riferimento un trigger non possano essere eliminate o modificate.	Necessario per i trigger su tabelle con ottimizzazione per la memoria.

Trigger AFTER

L'istruzione CREATE TRIGGER consente di creare un nuovo trigger che viene attivato automaticamente ogni volta che si verifica un evento come INSERT, DELETE o UPDATE in una tabella.

```
CREATE TRIGGER [schema_name.]trigger_name
ON table_name
AFTER {[INSERT],[UPDATE],[DELETE]}
[NOT FOR REPLICATION]
AS
{sql_statements}
```

dove:

- **schema_name** è il nome dello schema a cui appartiene il nuovo trigger. Il nome dello schema è facoltativo.

- **trigger_name** è il nome definito dall'utente per il nuovo trigger.
- **table_name** è la tabella a cui si applica il trigger.
- L'evento è elencato nella clausola AFTER. L'evento puòe essere INSERT, UPDATE, o DELETE. Un singolo trigger può essere attivato in risposta a una o più azioni contro sulla tabella.
- **NOT FOR REPLICATION** l'opzione indica a SQL Server di non attivare il trigger quando viene apportata una modifica ai dati come parte di un processo di replica.
- **sql_statements** tratta di una o più Transact-SQL utilizzate per eseguire azioni una volta che si verifica un evento.

Tabelle virtuali inserted e deleted

I trigger DML utilizzano due tabelle virtuali disponibili specificamente per i trigger chiamate **inserted** e **deleted**. Da un punto di vista strutturale queste tabelle sono simili alla tabella in cui viene definito il trigger, ovvero la tabella in cui si prova a eseguire l'azione utente. SQL Server utilizza queste tabelle per acquisire i dati della riga modificata prima e dopo il verificarsi dell'evento.

La tabella seguente mostra il contenuto delle tabelle **inserted** e **deleted** prima e dopo ogni evento:

Evento DML	La tabella inserted contiene	La tabella deleted contiene
INSERT	Le righe da inserire	vuota
UPDATE	nuove righe modificate dall'aggiornamento	righe esistenti modificate dall'aggiornamento
DELETED	vuota	righe da eliminare

Trigger INSTEAD OF

Un trigger INSTEAD OF è un trigger che consente di saltare un'istruzione INSERT, DELETE o UPDATE in una tabella o una vista ed eseguire invece altre istruzioni definite nel trigger. L'effettiva operazione di inserimento, eliminazione o aggiornamento non si verifica affatto. In altre parole, un trigger INSTEAD OF salta un'istruzione DML ed esegue altre istruzioni.

Sintassi del trigger di SQL Server INSTEAD OF

```
CREATE TRIGGER [nome.schema].nome_trigger
ON {nome_tabella_name | nome_vista }
INSTEAD OF {[INSERT] [,] [UPDATE] [,] [DELETE] }
AS
{sql_statements}
```

Dove

- **nome_schema** è il nome dello schema a cui appartiene il nuovo trigger. Il nome dello schema è facoltativo.
- **nome_trigger** è il nome definito dall'utente per il nuovo trigger.
- **nome_tabella** è la tabella a cui si applica il trigger.
- nella clausola **INSTEAD OF** specificare un evento INSERT, UPDATE O DELETE che verrà attivato dal trigger. Il trigger può essere chiamato per rispondere a uno o più eventi.
- **sql_statements** Il corpo di un trigger può essere costituito da una o più istruzioni Transact-SQL.

Esempio di trigger di SQL Server

Supponiamo che un'applicazione debba inserire nuovi marchi nella tabella **production_brands**. Tuttavia, i nuovi marchi dovrebbero essere archiviati in un'altra tabella chiamata **production.brand_approvals** per approvazione prima di inserirli nella tabella **production.brands**.

Per fare ciò, nell'applicazione creiamo una vista chiamata production.vw_brands per inserire nuovi marchi. Se i marchi vengono inseriti nella vista, verrà attivato un trigger INSTEAD OF per inserire i marchi nella tabella production.brand_approvals

Il codice è il seguente
La seguente istruzione crea una nuova tabella denominata **production.brand_approvals** per l'archiviazione dei marchi in attesa di approvazione:

```
CREATE TABLE production.brand_approvals(
    brand_id INT IDENTITY PRIMARY KEY,
    brand_name VARCHAR(255) NOT NULL
);
```

L'istruzione seguente crea una nuova vista denominata **production.vw_brands** riferita alle tabelle **production.brands** e **production.brand_approvals** con due campi (codice marchio e "Approved" per la tabella **production.brands** e (codice marchio e "Pending Approval" per la tabella **production.brand_approvals**

```
CREATE VIEW production.vw_brands
AS
SELECT
    brand_name,
    'Approved' approval_status              -- Stampa Approved nella Colonna approval status
FROM
    production.brands
UNION
SELECT
    brand_name,
    'Pending Approval' approval_status  -- Stampa Pending Approved nella Colonna approval status
FROM
    production.brand_approvals;
```

Quando inseriamo una riga nella vista production.vw_brands, questa riga va inserita nella tabella production.brand_approvals tramite il seguente trigger INSTEAD OF:

```
CREATE TRIGGER production.trg_vw_brands
ON production.vw_brands
INSTEAD OF INSERT
AS
BEGIN
    SET NOCOUNT ON;
    INSERT INTO production.brand_approvals (
        brand_name
    )
    SELECT
        i.brand_name
    FROM
```

```
        inserted i
    WHERE
        i.brand_name NOT IN (
            SELECT
                brand_name
            FROM
                production.brands
        );
END
```

Il trigger inserisce il nuovo marchio in production.brand_approvals se il marchio non esiste in production.brands.

Trigger DDL di SQL Server

I trigger DDL di SQL Server rispondono agli eventi del server o del database anziché alle modifiche ai dati della tabella. Questi eventi creati dall'istruzione Transact-SQL che in genere inizia con una delle seguenti parole chiave **CREATE**, **ALTER**, **DROP**, **GRANT**, **DENY**, **REVOKE** o **UPDATE STATISTICS.**

Ad esempio, è possibile scrivere un trigger DDL per registrare ogni volta che un utente esegue un'istruzione CREATE TABLE o ALTER TABLE.

I trigger DDL sono utili nei seguenti casi:

- quando si devono registrare le modifiche nello schema del database.
- Si vuole impedire alcune modifiche specifiche allo schema del database.
- Si deve rispondere a una modifica nello schema del database

La sintassi della creazione di un trigger DDL è la seguente:

```
CREATE TRIGGER nome_trigger
ON { DATABASE | ALL SERVER}
[WITH opzione_ddl_trigger]
FOR {event_type | event_group }
AS {sql_statement}
```

Dove:

- **nome_trigger** è il nome del trigger che si vuole creare. Non è necessario specificare uno schema per un trigger DDL perché non è correlato a una tabella o vista del database effettiva.
- **DATABASE | ALL SERVER si deve** utilizzare DATABASE se il trigger risponde a eventi nell'ambito del database o ALL SERVERse se il trigger risponde a eventi nell'ambito del server.
- **opzione_ddl_trigger** l'opzione può valere **ENCRYPTION** e/o la clausola **EXECUTE AS**. ENCRYPTION crittografa la definizione del trigger. EXECUTE AS definisce il contesto di sicurezza in cui viene eseguito il trigger.
- **event_type** Indica il tipo di evento DDL che provoca l'attivazione del trigger, ad esempio CREATE_TABLE, ALTER_TABLE, ALTER_TABLE, ecc.
- **event_group** è un gruppo di eventi **event_type** come DDL_TABLE_EVENTS.

Esempio di trigger DDL di SQL Server

Supponiamo di voler acquisire tutte le modifiche apportate all'indice del database in modo da poter monitorare meglio le prestazioni del server del database relative a queste modifiche all'indice.

Innanzitutto creiamo una nuova tabella di nome **index_logs** per registrare le modifiche all'indice:

```
CREATE TABLE index_logs (
   log_id INT IDENTITY PRIMARY KEY,
   event_data XML NOT NULL,
   changed_by SYSNAME NOT NULL
);
GO
```

Quindi, creiamo un trigger DDL per tenere traccia delle modifiche all'indice e inserire i dati degli eventi nella tabella index_logs:

```
CREATE TRIGGER trg_index_changes
ON DATABASE
FOR
   CREATE_INDEX,
```

```
   ALTER_INDEX,
   DROP_INDEX
AS
BEGIN
   SET NOCOUNT ON;

   INSERT INTO index_logs (
      event_data,
      changed_by
   )
   VALUES (
      EVENTDATA(),
      USER
   );
END;
GO
```

Nel corpo del trigger è stata abbiamo utilizzato la funzione **EVENTDATA()** che restituisce le informazioni sugli eventi del server o del database. La funzione è disponibile solo all'interno di DDL o trigger di accesso.

Se creiamo gli indici per le colonne **first_name** e **last_name** della tabella **sales.customers** con il seguente codice:

```
CREATE NONCLUSTERED INDEX nidx_fname
ON sales.customers(first_name);
GO

CREATE NONCLUSTERED INDEX nidx_lname
ON sales.customers(last_name);
GO
```

Successivamente, se eseguiamo una query sui dati dalla tabella index_changes potremmo vedere i dati che sono stai inseriti dal trigger