ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

NOME CORSO

Fintech Software Developer

Unità Formativa (UF): Basi di dati SQL

Docente: Durando Giulio

Titolo argomento: Transact-SQL

T-SQL (Transact-SQL)

Transact SQL è una collezione di estensioni capaci di aumentare le potenzialità dell'ANSI-SQL 92 (il linguaggio SQL standard). Oltre a questo Transact SQL ggiunge costrutti tipici della programmazione come istruzioni per il controllo del flusso (if e while per esempio), variabili, gestione degli errori, ecc… che ci mettono in grado di fare esercizio di programmazione dei databases.

Quindi con il T-SQL possiamo creare veri e propri programmi (le stored procedure e i triggers) sfruttando da un lato la potenza nell'interrogazione di basi dati delle istruzioni SQL e dall'altro funzioni e costrutti del T-SQL per la creazione della logica del programma.

T-SQL è fondamentale per l'uso dei prodotti e dei servizi Microsoft SQL. Tutti gli strumenti e le applicazioni che comunicano con un database SQL inviano comandi T-SQL.

Specificare gli oggetti del database

In Transact SQL il riferimento ai nomi degli oggetti del database avviene attraverso una sintassi composta tipicamente da quattro elementi:

```
[server_name.[database_name].[owner_name].
        | database_name.[owner_name].
         owner_name.
        ]
]
object_name
```

dove

- server_name è il nome del SQL Server (remoto o linked)
- database_name è il nome del database all'interno del SQL Server
- owner_name è il nome del proprietario dell'oggetto
- object_name è il nome dell'oggetto a cui voglio fare riferimento

Quando facciamo un riferimento ad uno specifico oggetto di un database, abbiamo diversi modi per identificarlo in modo corretto. La stringa di identificazione completa è:

server.database.propietario.oggetto >> mioserver.pubs.dbo.authors

Il nome del server, del database e del proprietario sono comunque opzionali e possono essere vicariati dal punto (.) per indicarne la posizione.

Variabili

Una variabile Transact-SQL locale è un oggetto che può contenere un solo valore di dati di un tipo specifico. Le variabili vengono in genere utilizzate in batch e script per gli scopi seguenti:

- Contare o controllare il numero di esecuzioni di un ciclo fungendo da contatori.
- Archiviare un valore di dati che deve essere testato da un'istruzione per il controllo di flusso.
- Salvare un valore di dati che deve essere restituito dal codice restituito di una stored procedure o dal valore restituito da una funzione.

Ogni variabile utilizzata all'interno di una stored procedure (o di un batch) deve essere dichiarata con una istruzione DECLARE e successivamente è possibile assegnarle un valore attraverso una istruzione SET o SELECT. La dichiarazione di una o più variabile può essere fatta su linea singola:

```
DECLARE @Variabile_1 AS VARCHAR(50), Variabile_2 AS INTEGER
```

o su più righe

```
DECLARE
@ Variabile_1 AS INTEGER,
@ Variabile_2 AS CHAR(50)
```

Il nome delle variabili deve iniziare con il simbolo “at” (@) e seguire le regole standard per gli identificatori (sono permessi caratteri alfanumerici e simboli come _, @, $ e #), la parola chiave AS è opzionale. Al contrario di altri linguaggi come il C++ od altri nel T-SQL non è possibile dichiarare costanti.

Impostazione di un valore in una variabile Transact-SQL

Quando si dichiara una variabile, il valore corrispondente viene impostato su NULL. Per assegnare un valore alla variabile, si può utilizzare l'istruzione SET. È consigliabile adottare sempre questo metodo. Per assegnare un valore a una variabile, è inoltre possibile fare riferimento alla variabile stessa nell'elenco di selezione di un'istruzione SELECT.

```
DECLARE @Cognome AS NVARCHAR(50),
        @Nome AS NVARCHAR(50);

SET @Conome  = N'Rossi';
SET @Nome = N'Mario';
```

Quando si assegna un valore a una variabile tramite un riferimento in un elenco di selezione, è necessario assegnare un valore scalare oppure l'istruzione SELECT deve restituire una sola riga. Ad esempio:

```
DECLARE @EmpIDVariable INT;

SELECT @EmpIDVariable = MAX(EmployeeID) FROM HumanResources.Employee;
```

Se l'istruzione SELECT restituisce più di una riga e la variabile fa riferimento a un'espressione non scalare, la variabile viene impostata sul valore restituito per l'espressione nell'ultima riga del set di risultati

Controllo del flusso nel T-SQL

Come in tutti i linguaggi di programmazione che si rispettino anche il Transact SQL possiede istruzioni per il controllo del flusso logico del programma.
Ne abbiamo di due tipi: **IF ELSE** e il loop **WHILE**, vorrei però aggiungere in questa lezione anche la funzione **CASE** molto utile nella programmazione T-SQL.
Prima di analizzare in dettaglio questi costrutti vediamo come usare l’istruzione **BEGIN END**.

BEGIN….END

BEGIN ed END sono parole chiave del linguaggio per il controllo di flusso. La coppia BEGIN…END, viene comunemente utilizzata per raggruppare più istruzioni SQL o T-SQL in un unico blocco. Dal punto di vista della logica di programmazione la coppia BEGIN…END è utile per racchiudere porzioni di codice, ad esempio, all’interno di una IF ELSE.

L’uso deve essere fatto in coppia, ad ogni BEGIN deve corrispondere una propria END, altrimenti il parser ci segnala un errore.

BEGIN END viene generalmente usata in questi casi:

- un loop WHILE che ha bisogno di includere interi blocchi di istruzioni.
- un elemento della funzione CASE che deve includere blocchi di istruzioni.
- una clausola IF o ELSE che deve includere blocchi di istruzioni.

Il costrutto IF...ELSE

Quando il programma incontra una IF valuta la condizione specificata e se è vera prende una certa strada, altrimenti se è falsa ne prende un'altra. Le condizioni che può valutare una IF ELSE in SQL Server possono essere due: TRUE o FALSE.

La sintassi dell'istruzione è

```
IF espressione_booleana
        { istruzione_SQL | blocco_istruzioni }
[ ELSE
        { istruzione_SQL | blocco_istruzioni } ]
```

Utilizzo di una query come parte di un'espressione booleana

Nell'esempio seguente viene eseguita una query come parte dell'espressione booleana. Poiché nella tabella Product sono presenti 10 biciclette che soddisfano la clausola WHERE, verrà eseguita la prima istruzione di stampa. Se ce ne fossero state un numero minore o uguale a 5 verrebbe eseguita l'istruzione relativa all'Else.

```
IF
(SELECT COUNT(*) FROM Production.Product WHERE Name LIKE 'Touring-3000%' ) > 5
        PRINT Ci sono più di 5 biciclette Touring-3000'
ELSE
        PRINT Ci sono 5 meno di 5biciclette Touring-3000' ;
```

Il costrutto WHILE

Imposta una condizione per l'esecuzione ripetuta di un'istruzione o di un blocco di istruzioni di SQL. Le istruzioni vengono eseguite ripetutamente per tutto il tempo in cui la condizione specificata risulta vera. È possibile controllare l'esecuzione di istruzioni nel ciclo WHILE dall'interno del ciclo tramite le parole chiave BREAK e CONTINUE.

La sintassi del costrutto è:

```
WHILE Boolean_expression
    { sql_statement | statement_block | BREAK | CONTINUE }
```

dove

Boolean_expression

E' una espressione che restituisce **TRUE** o **FALSE**. Se l'espressione booleana include un'istruzione SELECT, tale istruzione deve essere racchiusa tra parentesi.

{sql_statement | statement_block}

Qualsiasi istruzione o serie di istruzioni Transact-SQL valide definite con un blocco di istruzioni. Per definire un blocco di istruzioni, utilizzare le parole chiave per il controllo di flusso BEGIN ed END.

BREAK

Consente di uscire dal ciclo WHILE più interno. Vengono eseguite le istruzioni che si trovano dopo la parola chiave END, che segna la fine del ciclo. Quindi quando all'interno di un ciclo WHILE si incontra la parola chiave BREAK le istruzioni SQL fino alla parola chiave END vengono saltate.

CONTINUE

Consente il riavvio del ciclo WHILE, ignorando tutte le istruzioni che seguono la parola chiave CONTINUE.

Vediamo un esempio

```
DECLARE @contatore AS INTEGER
SET @contatore = 0
WHILE @contatore < 100
BEGIN
IF @contatore = 30
                BREAK
      SET @contatore = @contatore + 1
END
PRINT @contatore
```

CASE

Valuta un elenco di condizioni e restituisce una tra più espressioni di risultato possibili.

L'espressione CASE ha due formati:

- L'espressione CASE semplice confronta un'espressione con un set di espressioni semplici per determinare il risultato.
- L'espressione CASE avanzata valuta un set di espressioni booleane per determinare il risultato.

Entrambi i formati supportano un argomento facoltativo ELSE.

Istruzione SELECT con espressione CASE semplice

In un'istruzione SELECT un'espressione CASE semplice consente di eseguire solo un controllo di uguaglianza senza ulteriori confronti. Nell'esempio seguente viene utilizzata l'espressione CASE per modificare la visualizzazione delle categorie delle linee di prodotti in modo da renderle più intuitive.

```
SELECT   ProductNumber, Category =
    CASE ProductLine
      WHEN 'R' THEN 'Road'
      WHEN 'M' THEN 'Mountain'
      WHEN 'T' THEN 'Touring'
      WHEN 'S' THEN 'Other sale items'
      ELSE 'Not for sale'
    END,
  Name
FROM Production.Product
ORDER BY ProductNumber;
GO
```

Istruzione SELECT con un'espressione CASE avanzata

In un'istruzione SELECT l'espressione CASE avanzata consente di sostituire valori nel set di risultati in base ai valori di confronto. Nell'esempio seguente viene visualizzato il prezzo di listino come commento di testo in base alla fascia di prezzi per un prodotto.

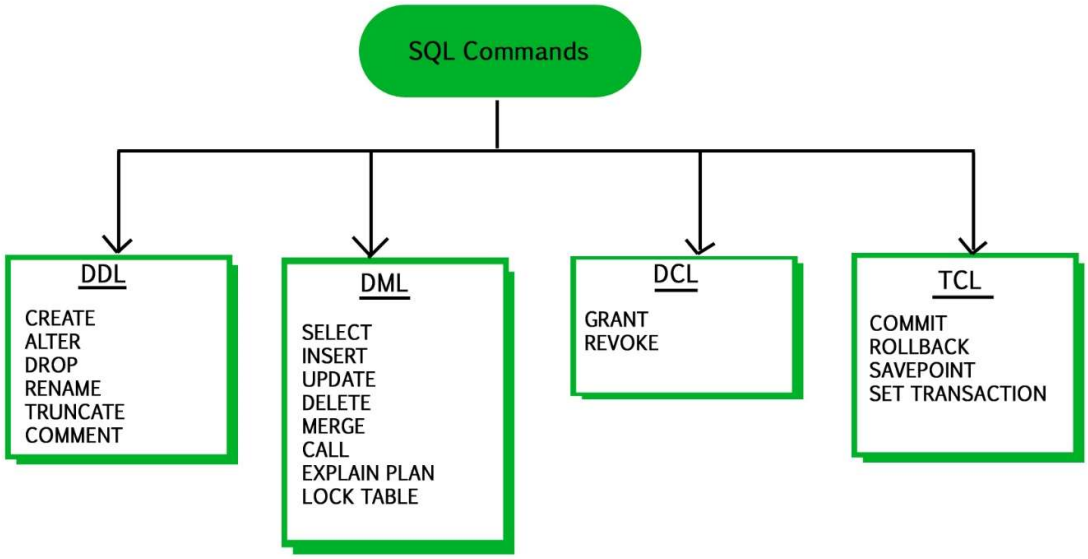
```
SELECT   ProductNumber, Name, "Price Range" =
    CASE
      WHEN ListPrice =  0 THEN 'Mfg item - not for resale'
      WHEN ListPrice < 50 THEN 'Under $50'
      WHEN ListPrice >= 50 and ListPrice < 250 THEN 'Under $250'
      WHEN ListPrice >= 250 and ListPrice < 1000 THEN 'Under $1000'
      ELSE 'Over $1000'
    END
FROM Production.Product
ORDER BY ProductNumber ;
GO
```

Script T-SQL

Gli script T-SQL si possono classificare in quattro gruppi:

- **DDL** (*Data Definition Language*): linguaggio per la descrizione dei dati, delle tabelle e delle viste (strumento con il quale si crea la struttura fisica del database, facendo riferimento allo schema logico).
- **DML** (*Data Manipulation Language*): linguaggio per il trattamento (o manipolazione) dei dati contenuti nel database (inserimenti, modifiche o cancellazioni).
- **TCL** *(Transaction Control Language)*: linguaggio per gestire il controllo delle transazioni
- **DCL** *(Data Control Language)*: utilizzato per controllare l'accesso ai dati archiviati in un database

Nella figura sotto sono elencate le istruzioni SQL suddivise a seconda del gruppo



SET ANSI_NULLS

Specifica il comportamento conforme allo standard ISO degli operatori di confronto Uguale a (=) e Diverso da (<>) quando vengono usati con valori Null in SQL Server.

SET ANSI_NULLS { ON | OFF }

Quando l'opzione ANSI_NULLS è impostata su ON, un'istruzione SELECT che usa WHERE *column_name* = **NULL** non restituisce alcuna riga anche se *column_name* include valori Null. Un'istruzione SELECT che usa WHERE *column_name* <>**NULL** non restituisce alcuna riga anche se *column_name* include valori non Null.

Quando l'opzione ANSI_NULLS è impostata su OFF, gli operatori uguale a (=) e diverso da (<>) non seguono lo standard ISO. Un'istruzione SELECT che usa WHERE *column_name* = **Null** restituisce le righe con valori Null in *column_name*. Un'istruzione SELECT che usa WHERE *column_name* <> **Null** restituisce le righe con valori non Null nella colonna. Inoltre un'istruzione SELECT che usa WHERE *column_name* <> *XYZ_value* restituisce tutte le righe che non sono *XYZ_value* e non sono NULL.

Quando l'opzione ANSI_NULLS è impostata su ON, tutti i confronti con un valore Null restituiscono UNKNOWN. Quando l'opzione SET ANSI_NULLS è impostata su OFF, i confronti di tutti i dati con un valore Null restituiscono TRUE se il valore dei dati è NULL. In caso di omissione di SET ANSI_NULLS, viene applicata l'impostazione dell'opzione ANSI_NULLS del database corrente.

SET QUOTED_IDENTIFIER

Impone in SQL Server la conformità alle regole ISO relative all'utilizzo delle virgolette per delimitare identificatori e stringhe letterali. Gli identificatori delimitati da virgolette doppie possono essere parole chiave riservate Transact-SQL o possono includere caratteri normalmente non consentiti dalle regole della sintassi Transact-SQL per gli identificatori.

SET QUOTED_IDENTIFIER { ON | OFF }

Quando SET QUOTED_IDENTIFIER è ON (impostazione predefinita), gli identificatori possono essere racchiusi tra virgolette doppie (" "), mentre i valori letterali devono essere racchiusi tra virgolette singole (' '). Tutte le stringhe delimitate da virgolette doppie vengono interpretate come identificatori di oggetto. Gli identificatori delimitati pertanto non devono necessariamente essere conformi alle regole per gli identificatori di Transact-SQL. Possono essere parole chiave riservate e includere caratteri normalmente non consentiti negli identificatori Transact-SQL. Non è possibile utilizzare le virgolette doppie per delimitare espressioni di stringhe letterali. Inoltre le stringhe letterali devono essere racchiuse tra virgolette singole. Se la stringa letterale contiene una virgoletta singola ('), questa può essere rappresentata da due virgolette singole (''). Quando si usano parole chiave riservate per nomi di oggetti nel database, è necessario che l'opzione SET QUOTED_IDENTIFIER sia impostata su ON.

Quando l'opzione SET QUOTED_IDENTIFIER è impostata su OFF, non è possibile racchiudere tra virgolette gli identificatori, i quali devono essere conformi a tutte le regole per gli identificatori di Transact-SQL. Per altre informazioni, vedere Identificatori del database. È possibile delimitare i valori letterali con virgolette singole o doppie. Se una stringa letterale è delimitata da virgolette doppie, la stringa può includere virgolette singole, ad esempio gli apostrofi.