CORSO DI SQL SERVER 7

Ing. Andrea Bulgarelli
0328/2698871
andrea.bulgarelli@libero.it
http://web.tiscalinet.it/andbulga

INDICE

GENERALITÀ SU SQL-SERVER

Database

Un database può essere descritto in termini semplici come un insieme di oggetti che contengono e gestiscono dati. Una tabella può ad esempio descrivere tutti i clienti o gli ordini di un'azienda.
Ogni tabella è organizzata in righe e colonne. Una riga rappresenta un record di una tabella (ad esempio, un singolo cliente, un singolo ordine). Le colonne invece descrivono le caratteristiche dei singoli record (ad esempio, un record può essere composto dal nome, cognome e indirizzo del cliente).
La seguente figura mostra due possibili tabelle di un database per la gestione di una libreria.

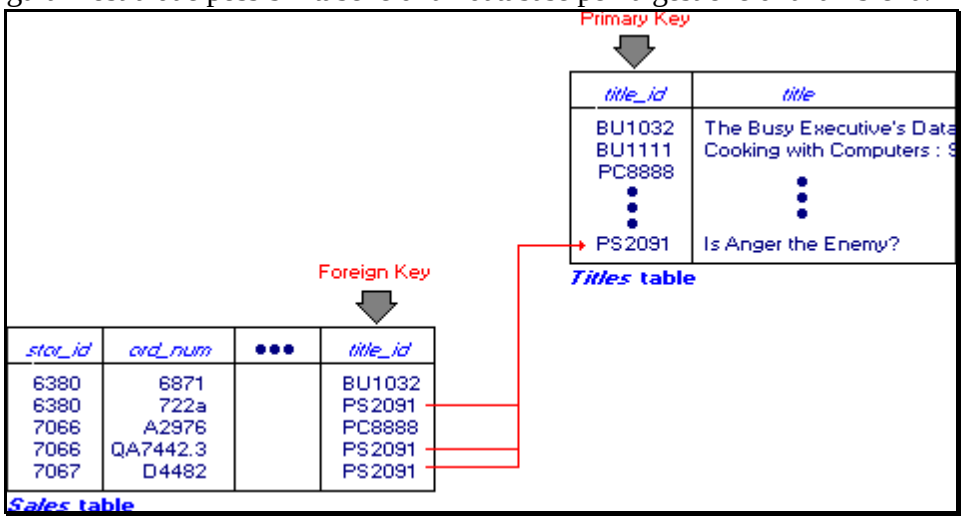

Figura 1: Tabelle, righe e colonne

DBMS

Un DBMS (Database Management System), o Sistema di Gestione di basi di dati rappresenta il risultato più rilevante dell'evoluzione tecnologica nel campo dei *sistemi informativi.* Un DBMS è un componente software fondamentale di tale sistema informativo e consente la gestione efficiente delle informazioni in forma integrata e ne garantisce la persistenza.
Prima dell'avvento dei DBMS, in grosse aziende i dati erano organizzati in *sistemi informatici settoriali*, nei quali la comunicazione di dati tra un settore e l'altro avveniva mediante copia (cartacea o elettronica). L'utilizzo dei sistemi informatici settoriali comportava numerosi problemi, tra i quali è possibile ricordare:

- Ridondanza dei dati
- Le operazioni di aggiornamento di un archivio si devono propagare in tutti gli archivi
- Non è possibile imporre tutti i vincoli necessari tra i dati
- I meccanismi di sicurezza dei dati possono variare da settore a settore

In un DBMS i dati sono rappresentati in forma integrata (logicamente, non necessariamente fisicamente). In tal modo tutta l'azienda può condividere lo stesso database, risolvendo molti dei problemi citati in precedenza (lasciando tale compito al DBMS).

Funzionalità del DBMS

Un DBMS è quindi un sistema software in grado di gestire grandi quantità di dati in modo efficiente. In SQL Server un database può avere dimensioni da 1 MB a 1 TB (1.048.576 MB).
Le caratteristiche funzionali di un DBMS possono essere così riassunte:

- Supporto per almeno un modello dei dati
- Linguaggi di alto livello per la creazione, la manipolazione (inserimento, modifica e cancellazione) e l'interrogazione dei dati e per il controllo del database. Da un punto di vista concettuale, i linguaggi di alto livello possono essere classificati in:
  - DDL → Data Definition Language: linguaggio non procedurale che permette di definire e mantenere la struttura del database. Permette inoltre di specificare la connessione tra le strutture logiche e fisiche del database
  - DML → Data Manipulation Language: o query language, consente l'interrogazione e la modifica dei dati del database
- Gestione delle transazioni
- Controllo degli accessi (o protezione):
- Resilienza (fault tolerance): SQL Server garantisce la massima disponibilità mediante le funzioni di backup e manutenzione in linea, il ripristino automatico nel caso di malfunzionamenti e la possibilità di installare SQL Server in un cluster per il supporto del failover.
- Presenza di diversi ambienti di sviluppo e utilità: interfacce di programmazione, Visual Basic, C++, ASP (per la programmazione Internet).

Prima di entrare nel dettaglio su come SQL Server fornisca le funzionalità elencate in precedenza, vediamo in dettaglio il significato di transazione.

Transazione

Una *transazione* è l'unità fondamentale di lavoro di SQL Server. Essa consiste in diversi comandi DML che leggono e aggiornano il database. Un DBMS deve gestire:

- Transazioni concorrenti: due transazioni si dicono concorrenti se avvengono nello stesso istante di tempo.
- Malfunzionamenti:
  - Transaction abort: interruzione di una transazione. Non comporta perdita dei dati. Causata da una situazione prevista dal programma
  - System abort: interruzione di transazioni attive derivanti da un'anomalia hardware o software. Si assume che il contenuto della memoria permanente sopravviva, mentre viene perduto il contenuto della memoria temporanea
  - System crash: anomalia che danneggia il contenuto della memoria permanente

L'aggiornamento non viene considerato definitivo fino a quando non viene inoltrato il COMMIT, che indica la fine della transazione. Si consideri il seguente esempio di un trasferimento bancario tra due conti, per comprendere meglio che cosa sia una transazione:

BEGIN TRANSACTION
Accredita 1 ML sul conto 2254/1
Addebita 1 ML sul conto 2378/1

COMMIT TRANSACTION

che potrebbe anche essere eseguita nel seguente ordine:

BEGIN TRANSACTION
Addebita 1 ML sul conto 2378/1
Accredita 1 ML sul conto 2254/1
COMMIT TRANSACTION

L'elaborazione delle transazioni in SQL Server (ma è lo stesso anche per gli altri DBMS) garantisce che tutte le manipolazioni dei dati siano eseguite come unità di lavoro individuale, anche in presenza di malfunzionamenti.
Per raggiungere questo obiettivo le transazioni devono rispettare le proprietà ACID:

1. **Atomicità**: una transazione è confermata o annullata senza possibilità intermedie. Se una transazione è confermata, tutti i suoi effetti sono mantenuti;
2. **Coerenza**: questa proprietà fa in modo che le transazioni non consentano al database di raggiungere uno stato logico non corretto. In questo caso la regola logica è che il denaro non può essere né creato né distrutto;
3. **Isolamento**: l'isolamento separa le transazioni concorrenti da altre transazioni concorrenti.
4. **Durabilità**: Successivamente al commit di una transazione, la durabilità garantisce che gli effetti della transazione restino validi anche in caso di errori del sistema

Durante la sua evoluzione, una transazione può passare nei seguenti stati:

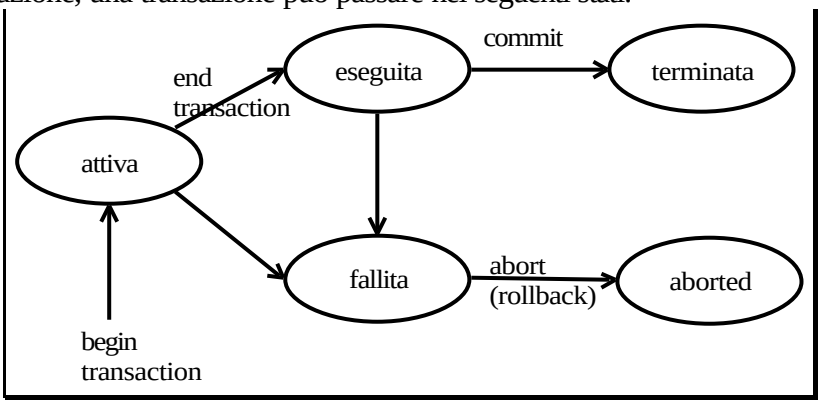

Figura 2: stati di una transazione

Problema della concorrenza

SQL Server risolve anche i problemi legati alla concorrenza delle transazioni, come quelli di seguito rappresentati.

- Lost update problem

Transaction A	**Time**	**Transaction B**
Read R	T1	
	T2	Read R
Update R- commit	T3	
	T4	Update R - commit

- Uncommitted dependency problem

Transaction A	**Time**	**Transaction B**
	T1	
	T2	Update R
Read R	T3	
	T4	Rollback

- Inconsistency analysys problem: consiste nell'eseguire analisi sui dati durante la modifica di questi (si vedano i DSS, Decision Support System).

SQL Server risolve tutti questi problemi, ma prima di entrare nel dettaglio della struttura e degli strumenti di questo DBMS, conviene comprendere meglio come progettare un database.

SCHEMA RELAZIONALE

Una **relazione**, o **tabella**, è formata dall'insieme degli attributi, detti anche **campi** o **colonne** della tabella**.** Ogni attributo è caratterizzato da un **dominio**, che rappresenta la collezione di tutti i possibili valori che la colonna può assumere, e da un **nome**.
SQL Server è un *DBMS relazionale*, poiché utilizza il modello relazionale per descrivere la struttura e l'organizzazione dei database.

Esempio

Vediamo ora la traduzione in schema relazionale dell'esempio presente in **Errore. L'origine riferimento non è stata trovata.**.

```
CLIENTE(CF, Nome, Cognome, Indirizzo)

ORDINE(Numero, Anno, Data, CF, id)
FK: CF REFERENCES CLIENTE
AK: id

PRODOTTO(Codice, Descrizione, Quantita_Magazzino)
```

Per porre in relazione le tabelle precedente, è necessario creare anche la seguente tabella:

```
PRODOTTO_ORDINE(Codice_prodotto, Numero_Ordine, Anno, Quantita)
FK: Numero_Ordine, Anno REFERENCES ORDINE
FK: Codice_prodotto REFERENCES PRODOTTO
```

Il modello relazionale, i DBMS e l'integrità dei dati (Data Integrity)

Uno dei punti di forza del modello relazionale è quello di disporre di numerosi strumenti per assicurare l'integrità dei dati, imponendo i cosiddetti vincoli d'integrità. Per integrità dei dati si indica solitamente la caratteristica dei dati di essere accurati e attendibili in accordo con il modello logico. Uno dei compiti più importanti dei DBMS è quello di garantire questa integrità, in modo tale che i dati siano sempre consistenti e corretti.
Il rafforzamento dell'integrità dei dati consente di avere dati di qualità nel database. Tipico esempio è il vincolo che impedisce di memorizzare due volte lo stesso codice fiscale nella tabella dei clienti, perché questo comporterebbe una doppia registrazione dello stesso cliente con tutti i problemi d'identificazione correlati. Un altro vincolo potrebbe essere quello di impedire di effettuare ordini con quantità negative.

I tipi di integrità

- **Entity integrity (o primo vincolo di integrità)**: ogni riga deve essere unica all'interno della tabella. Questo è realizzato mediante la selezione di un identificatore (costituito da una o più colonne della tabella) che deve assumere valori univoci in tutte le righe della stessa tabella.
- **Domain integrity** indica la validità di un dato di una determinata colonna. In sostanza, con una serie di costrutti come i tipi di dati, le regole di CHECK oppure le rules, i DEFAULT e i NOT NULL è possibile definire in modo preciso il dominio (l'insieme dei valori) che una determinata colonna può assumere. Anche con i vincoli di foreign key è possibile restringere il dominio della colonna, ma mediante tale costrutto si realizza anche l'integrità di seguito descritta.

- **Referential integrity (o secondo vincolo di integrità)** consente di conservare le relazioni definite tra le tabelle quando i record sono aggiunti o eliminati. Per la definizione di questo vincolo si stabilisce la relazione tra uno o più attributi di una tabella (chiamata tabella correlata) e la chiave primaria (o alternata) di un'altra tabella (chiamata tabella primaria). Lo scopo è quello di assicurare che i valori delle chiavi siano consistenti attraverso le tabelle. Nell'esempio considerato in precedenza la relazione *Ordini* deve contenere il codice fiscale del cliente che effettua l'ordine; con questo tipo di integrità è possibile assicurare che tale codice fiscale sia compreso tra quello presenti nella tabella *Clienti*. In sostanza, è impossibile fare riferimento ad una riga di un'altra (identificata dalla sua chiave) se questa non esiste. Inoltre, le modifiche ad una tabella devono ripercuotersi anche nel resto del database. SQL Server impedisce di
  - Aggiungere record in una tabella correlata se non è presente il record correlato nella tabella primaria
  - Cambiare valore della chiave nella tabella primaria se tale valore chiave è referenziato da un record nella tabella correlata
  - Eliminare il record di una tabella primaria se questo è referenziato da una tabella correlata

  Per permettere alcune delle operazioni precedenti (ad esempio la cancellazione) si ricorre alle operazioni referenziali.
- **User-defined integrity** permette di definire le business rules che non è possibile imporre con i tre tipi di integrità descritti in precedenza.

Con lo schema relazionale presentato nell'esempio precedente sono stati imposti il primo e il secondo vincolo di integrità. Per gli altri tipi di vincoli è necessario ricorrere ai costrutti di seguito presentati.

Gli strumenti forniti dai DBMS per imporre i vincoli di integrità

Ci sono diversi modi per rafforzare i vincoli di integrità.

Integrity type	Opzione raccomandata
Entity	PRIMARY KEY constraint UNIQUE constraint IDENTITY property
Domain	DEFAULT definition FOREIGN KEY constraint CHECK constraint NOT NULL
Referential	FOREIGN KEY constraint CHECK constraint
User-defined	All column- and table-level constraints in CREATE TABLE Stored Procedures Triggers

Imporre i vincoli di integrità

Tipi di dati

I tipi di dati sono il modo più semplice per imporre i vincoli di dominio. Il dominio selezionato in base al tipo di dato potrà essere poi ristretto con i vincoli di check descritti nel paragrafo successivo.

Tipo di dati	Descrizione
binary	Consente di memorizzare un massimo di 255 byte di dati binari a lunghezza fissa. La lunghezza predefinita è di 10 byte. L'accesso alle colonne con tipo di dati binary risulta più veloce rispetto all'acceso alle colonne di tipo varbinary in quanto viene utilizzata una lunghezza di memorizzazione fissa. Scegliere questo tipo di dati per colonne in cui vengono immessi valori con lunghezza simile. È possibile creare relazioni tra colonne binary e colonne varbinary.
bit	Consente di memorizzare il valore 1 o 0. Sono ammessi anche valori interi diversi da 1 e 0, che vengono tuttavia interpretati sempre come 1. Le dimensioni di memorizzazione sono di 1 byte. In una tabella è possibile raggruppare più tipi di bit in byte. Le colonne a 7 bit, ad esempio, occupano 1 byte, mentre le colonne a 9 bit occupano 2 byte. Nelle colonne con tipo di dati bit non sono ammessi valori Null o indici Scegliere questo tipo di dati per colonne in cui vengono immessi dati di tipo vero/falso oppure sì/no.
char	Consente di memorizzare un massimo di 255 caratteri. La lunghezza predefinita è di 10 caratteri. L'accesso alle colonne con tipo di dati char risulta più veloce rispetto all'acceso alle colonne di tipo varchar in quanto viene utilizzata una lunghezza di memorizzazione fissa. Scegliere questo tipo di dati per colonne in cui vengono immessi valori con lunghezza simile. È possibile creare relazioni tra colonne char e colonne varchar.
datetime	Consente di memorizzare valori in due interi da 4 byte, ovvero 4 byte per il numero di giorni precedenti o successivi alla data di base 1 gennaio 1900 e 4 byte per il numero di millisecondi dopo la mezzanotte. I segmenti di data dei valori datetime che rappresentano date precedenti alla data di base vengono memorizzati come valori negativi. I valori datetime sono compresi tra l'1 gennaio 1753 e il 31 dicembre 9999, con precisione a 1 trecentesimo di secondo, ovvero 3,33 millisecondi. In Microsoft SQL Server i valori che non vengono riconosciuti come date comprese tra il 1753 e il 9999 vengono rifiutati. È possibile omettere la parte della data o quella dell'ora. Se si omettono entrambe, la data predefinita è la mezzanotte dell'1 gennaio 1990 (January 1, 1900, 12:00:00:000AM). Se si omette l'ora, il valore predefinito è la mezzanotte (12:00:00:000AM), mentre se si omette la data, il valore predefinito è l'1 gennaio 1990 (Jan 1 1990).
decimal(p, s)	Tipo di dati numerico esatto che consente di memorizzare valori interi o frazionari compresi tra $10^{38} - 1$ e -10^{38}. La proprietà Precisione **p** specifica il numero totale massimo di cifre decimali che è possibile memorizzare, sia a sinistra che a destra del separatore decimale, mentre la proprietà Scala **s** specifica il numero massimo di cifre decimali che è possibile memorizzare a destra del separatore decimale; tale valore deve essere minore o uguale alla precisione. Le dimensioni dello spazio di memorizzazione variano a seconda della precisione specificata e sono comprese tra 2 e 17 byte. La regola generale per calcolare il numero di byte di memorizzazione necessari consiste nel dividere la precisione per 2. Per una precisione compresa tra 17 e 19, ad esempio, sono necessari 9 byte di spazio di memorizzazione. Per ulteriori informazioni, consultare la documentazione del database.
float	Consente di memorizzare numeri a virgola mobile positivi o negativi. Per impostazione predefinita, la precisione di questo tipo di dati è di 15 cifre. I valori positivi sono compresi approssimativamente tra 2,23E - 308 e 1,79E 308, mentre quelli negativi sono compresi approssimativamente tra –2,23E - 308 e –1,79E 308. È inoltre possibile memorizzare lo zero. Le dimensioni dello spazio di memorizzazione sono di 8 byte. Con il tipo di dati float è possibile eseguire tutte le operazioni aritmetiche, ad eccezione del modulo. Nel caso di dati float con componente esponenziale, immettere un numero con o senza separatore decimale e segno positivo o negativo, seguito da e o E e quindi da un intero. Il valore rappresentato da un tipo di dati float è uguale a un numero moltiplicato per 10 ed elevato quindi alla potenza di un secondo numero.
image	Tipo di dati a lunghezza variabile che consente di memorizzare un massimo di byte di dati

	binari pari a 2E31 - 1 (2.147.483.647). La lunghezza predefinita è di 16 byte. Questo tipo di dati non può essere utilizzato per le variabili e i parametri di stored procedure. Quando si immettono dati di tipo image con un numero di byte dispari e minore di 255, vengono inseriti automaticamente zero di riempimento. Non è possibile inserire valori di tipo image con un numero di caratteri dispari maggiore di 255 byte.
int	Consente di memorizzare numeri interi compresi tra -2E31 (-2.147.483.648) e 2E31 - 1 (2.147.483.647). Le dimensioni dello spazio di memorizzazione sono di 4 byte.
money	Consente di memorizzare valori monetari compresi tra –922.337.203.685.477,5808 e +922.337.203.685.477,5807 con precisione pari a dieci millesimi di una unità monetaria. I valori di tipo money sono rappresentati come interi a precisione doppia. Le dimensioni dello spazio di memorizzazione sono di 8 byte.
numeric	Identico al tipo di dati decimal in Microsoft SQL Server 6.5. Vengono forniti entrambi i tipi di dati per compatibilità con ANSI. Vedere il tipo di dati decimal. **Attenzione** Per dati numerici esatti di un database è consigliabile utilizzare il tipo di dati decimal o numeric in modo da evitare la conversione tra questi due tipi di dati e per consentire la creazione di relazioni tra colonne con dati numerici esatti.
real	Tipo di dati in virgola mobile simile a float con precisione di 7 cifre. L'intervallo di valori positivi è compreso approssimativamente tra 1,18E - 38 e 3,40E 38, mentre quelli negativi sono compresi approssimativamente tra –1,18E - 38 e –3,4E 38. È inoltre possibile memorizzare lo zero. Le dimensioni dello spazio di memorizzazione sono di 4 byte.
smalldatetime	Tipo di dati per la data e l'ora meno preciso del tipo di dati datetime. Le dimensioni dello spazio di memorizzazione sono di 4 byte, composti da un intero breve per il numero di giorni successivi all'1 gennaio 1900 e da un intero breve per il numero di minuti successivi alla mezzanotte. L'intervallo dei valori smalldatetime è compreso tra l'1 gennaio 1900 e il 6 giugno 2079, con precisione di un minuto.
smallint	Consente di memorizzare numeri interi compresi tra -2E15 (-32.768) e 2E15 - 1 (32.767). Le dimensioni dello spazio di memorizzazione sono di 2 byte.
smallmoney	Consente di memorizzare valori monetari compresi tra –214.748,3648 e +214.748,3647, con precisione pari a 10 millesimi di una unità monetaria. I valori di questo tipo i dati vengono visualizzati con un arrotondamento a due cifre.
sysname	Tipo di dati equivalente a varchar(30) che dovrebbe essere assegnato soltanto a colonne che fanno riferimento al nome di un oggetto di database delle tabelle di sistema di Microsoft SQL Server, ad esempio `sysbobjects`.
text	Tipo di dati a lunghezza variabile che consente di memorizzare un massimo di 2E31 - 1 (2.147.483.647) caratteri. La lunghezza predefinita è di 16 caratteri.
timestamp	Viene aggiornato automaticamente a ogni inserimento e aggiornamento di una riga contenente una colonna timestamp. I valori di queste colonne non sono dati di tipo datetime, ma di tipo binary(8) o varbinary(8) che indicano la sequenza dell'attività di Microsoft SQL Server nella riga. Una tabella può includere una sola colonna timestamp. Il tipo di dati timestamp non è correlato in alcun modo all'ora di sistema. Si tratta semplicemente di un contatore a incremento monotonico i cui valori sono sempre univoci all'interno del database.
tinyint	Consente di memorizzare numeri interi compresi tra 0 e 255. Le dimensioni dell'area di memorizzazione sono di 1 byte.
varbinary	Tipo di dati binary a lunghezza variabile che consente di memorizzare un massimo di 255 byte di dati binari a lunghezza variabile. La lunghezza predefinita è di 50 byte. Le dimensioni dell'area di memorizzazione sono pari alla lunghezza effettiva dei dati immessi. Scegliere il tipo di dati varbinary quando sono previsti valori Null o dati di dimensioni

	variabili. È possibile creare relazioni tra colonne binary e colonne varbinary.
varchar	Consente di memorizzare un massimo di 8000 caratteri. La lunghezza predefinita è di 50 caratteri. Le dimensioni dell'area di memorizzazione non sono fisse, ma dipendono dalle dimensioni effettive dei dati immessi (gli spazi finali vengono ignorati). Scegliere il tipo di dati varchar quando sono previsti valori Null o dati di dimensioni molto variabili. È possibile creare relazioni tra colonne char e colonne varchar.

Altri tipi di dati:

1. Stringhe di caratteri Unicode:
   a. nchar → fino a 4000 caratteri
   b. nvarchar → fino a 4000 caratteri
   c. ntext → fino a 2E30-1 caratteri
2. cursor: riferimento ad un cursore. Può essere utilizzato solamente come variabile e non come colonna di tabella
3. uniqueidentifier: identificatore univoco globale di 16 byte.

Vincoli di CHECK

Specifica i valori di dati accettabili in una colonna. È possibile applicare un singolo vincolo CHECK a più colonne oppure più vincoli a una singola colonna. Con l'eliminazione di una tabella, vengono eliminati anche tutti i vincoli CHECK.
I vincoli di CHECK sono espressioni SQL.
Nella casella **Espressione di vincolo** nella scheda **Tabelle** delle pagine delle proprietà digitare l'espressione in base alla seguente sintassi:

{*costante* | *nome_colonna* | *funzione* | (*sottoquery*)}
[{*operatore* | AND | OR | NOT}
{*costante* | *nome_colonna* | *funzione* | (*sottoquery*)}…]

La sintassi SQL è composta dai seguenti parametri:

Parametro	**Descrizione**
costante	Valore letterale, ad esempio dati numerici o caratteri. I caratteri devono essere racchiusi tra virgolette singole (').
nome_colonna	Specifica una colonna.
funzione	Funzione predefinita. Per informazioni dettagliate sulle funzioni, vedere "Functions" in *SQL Server Books Online* o in *Transact-SQL Reference* (informazioni in lingua inglese).
operatore	Operatore aritmetico, bit per bit, di confronto o stringa. Per informazioni dettagliate sugli operatori, vedere "Operators" in *SQL Server Books Online* o in *Transact-SQL Reference*.
AND	Operatore utilizzato in espressioni booleane per collegare due espressioni. I risultati vengono restituiti quando entrambe le espressioni risultano vere. Quando un'istruzione include sia AND che OR, viene elaborato per primo l'operatore AND. È tuttavia possibile modificare l'ordine di esecuzione utilizzando le parentesi.
OR	Operatore utilizzato in espressioni booleane per collegare due o più condizioni. I risultati vengono restituiti quando una delle espressioni risulta vera. Quando un'istruzione include sia AND che OR, l'operatore OR viene elaborato dopo AND. È tuttavia possibile modificare l'ordine di esecuzione utilizzando le parentesi.
NOT	Consente di negare un'espressione booleana, che può includere parole chiave quali LIKE, NULL, BETWEEN, IN e EXISTS.

	Quando un'istruzione include più operatori logici, viene elaborato per primo l'operatore NOT. È tuttavia possibile modificare l'ordine di esecuzione utilizzando le parentesi.

Esempi

Espressione di vincolo con cui vengono accettati esclusivamente numeri a cinque cifre:

```
zip LIKE '[0-9][0-9][0-9][0-9][0-9]'
```

Vengono accettati esclusivamente valori positivi:

```
qty > 0
```

Vengono accettati solo gli ordini eseguiti con Visa, MasterCard o American Express:

```
NOT (payment_method = 'credit card') OR
   (card_type IN ('VISA', 'MASTERCARD', 'AMERICAN EXPRESS'))
```

Rules

Le rules sono una caratteristica di SQL Server mantenuta per compatibilità con le versioni precedenti e hanno la stessa funzione dei vincoli di CHECK, che comunque sono da preferire alle rules. E' possibile associare una sola rule per ogni colonna mentre questa limitazione non è presente per i CHECK. Inoltre, mentre i CHECK constraint sono specificati in fase di creazione della tabella, le rules devono essere specificate come oggetti separati.
Esempio: @field LIKE '[0-9][0-9][0-9][0-9][0-9]'

Vincoli PRIMARY KEY

Un vincolo PRIMARY KEY (chiave primaria) consente di impedire l'immissione di valori duplicati o NULL in colonne specifiche. Con questo tipo di vincolo è possibile impostare sia l'univocità che l'integrità referenziale. La colonna `au_id`, ad esempio, identifica in modo univoco ciascun autore memorizzato nella tabella `authors`.
I vincoli PRIMARY KEY vengono creati direttamente nei diagrammi di database.

Vincoli UNIQUE

Un vincolo UNIQUE consente di impedire l'immissione di valori duplicati in colonne specifiche non incluse nella chiave primaria della tabella. Ad esempio, nella tabella `employee` in cui la colonna `emp_id` è la chiave primaria è possibile definire un vincolo UNIQUE in base al quale tutte le voci del numero di codice fiscale nella colonna `ssn` devono essere univoche all'interno della tabella.
Nei diagrammi di database è possibile creare, modificare ed eliminare vincoli UNIQUE mediante la pagina delle proprietà Indici/chiavi.

DEFAULT Definitions

Ogni colonna di una tabella deve contenere un valore, anche se è possibile inserire il NULL poiché in questo contesto è considerato un valore. Con l'opzione DEFAULT è possibile specificare un valore iniziale per la colonna utilizzabile nel caso in cui non sia specificato esplicitamente nessun valore per la stessa.

IDENTITY Property

La proprietà IDENTITY può essere utilizzata per costruire degli identificatori in cui il valore della chiave è stabilito in modo automatico. Tali valori sono numerici (tipicamente int) ed è necessario specificare l'**Identity Seed**, cioè il valore iniziale della chiave (da assegnare al primo record che viene inserito nella tabella) e l'**Identity Increment**, per specificare l'incremento di valore per ogni nuovo record che viene inserito nella tabella.

Per l'utilizzo dell'IDENTITY si tenga in considerazione che è possibile utilizzare i tipi di dati **decimal**, **int**, **numeric**, **smallint** o **tinyint** e che per una colonna IDENTITY non è possibile specificare un valore di DEFAULT e non può contenere valori NULL.

Finestra Visualizzazione Dati

La finestra di visualizzazione dati è presenti in SQL Server Enterprise Manager oppure nei Visual Database Tool di Visual Studio e consente di definire le tabelle e tutti gli oggetti correlati. Di seguito sono elencati i campi principali.

Tipo di dati	Descrizione	Valore predefinito
Nome colonna	Nome di una colonna inclusa in una tabella. Per i nomi di colonna, che devono essere univoci nella tabella, è necessario seguire le convenzioni di denominazione per gli identificatori del database in uso.	Nessun valore
Tipo di dati	Tipo di dati della colonna. Sono ammessi tipi definiti dall'utente o dal sistema.	Carattere (char)
Lunghezza	Numero massimo di caratteri o cifre (per i tipi di dati numerici) per i valori della colonna.	10
Precisione	Numero totale massimo di cifre decimali che è possibile memorizzare, sia a sinistra che a destra del separatore decimale.	0
Scala	Numero massimo di cifre decimali che è possibile memorizzare a destra del separatore decimale. Questo valore deve essere minore o uguale alla precisione.	0
Ammetti Null	Indica se nella colonna sono ammessi o meno valori Null.	Sì (casella di controllo selezionata)
Valore predefinito	Valore che viene inserito automaticamente nella colonna quando l'utente non immette alcun valore. I valori predefiniti di colonne con tipo di dati timestamp vengono ignorati. Nelle colonne che ammettono valori Null e per le quali non è stato impostato un valore predefinito viene inserito un valore Null.	Nessun valore
Identità	Indica se per le nuove righe della colonna vengono generati o meno valori incrementali in base alle impostazioni di **Inizio identità** e **Incremento attività**.	No (casella di controllo deselezionata)
Inizio identità	Valore assegnato alla prima riga della tabella. Se non si specifica alcun valore, viene impostato automaticamente 1.	Nessun valore
Incremento attività	Valore aggiunto in **Inizio identità** e assegnato alla seconda riga della tabella. Tale valore viene quindi aggiunto al valore di ciascuna riga successiva. Se non si specifica alcun valore, viene impostato automaticamente 1.	Nessun valore

SQL

Select

```
SELECT [DISTINCT][TOP n] <columns to be chosen, optionally
                         eliminating duplicate rows from result set or
                         limiting number of rows to be returned>
[FROM] <table names>
[WHERE] <criteria that must be true for a row to be chosen>
[GROUP BY] <columns for grouping aggregate functions>
[HAVING] <criteria that must be met for aggregate functions>
[ORDER BY] <optional specification of how the results should be
           sorted>
```

Join:

```
SELECT
'Author'=RTRIM(au_lname) + ', ' + au_fname,
'Title'=title
FROM authors AS A JOIN titleauthor AS TA
    ON A.au_id=TA.au_id          -- JOIN CONDITION
WHERE A.state <> 'CA'
ORDER BY 1
```

Gestione dei NULL: esiste IS NULL

Insert

```
INSERT [INTO] {table_name|view_name} [(column_list)]
VALUES value_list

INSERT author_sales
    SELECT 'SELECT', authors.au_id, authors.au_lname,
        SUM(titles.price * sales.qty)
    FROM authors INNER JOIN titleauthor
        ON authors.au_id = titleauthor.au_id INNER JOIN titles
        ON titleauthor.title_id = titles.title_id INNER JOIN sales
        ON titles.title_id = sales.title_id
    WHERE authors.au_id LIKE '8%'
    GROUP BY authors.au_id, authors.au_lname
```

Update

```
UPDATE {table_name|view_name}
SET column_name1 = {expression1|NULL|DEFAULT|(SELECT)}
    [, column_name2 = {expression2|NULL|DEFAULT|(SELECT)}
WHERE {search_conditions}

UPDATE titles
SET price = price * 2

UPDATE authors
    SET state = 'PC', city = 'Bay City'
        WHERE state = 'CA' AND city = 'Oakland'

UPDATE titles
```

```
    SET ytd_sales = titles.ytd_sales + sales.qty
        FROM titles, sales
            WHERE titles.title_id = sales.title_id
            AND sales.ord_date = (SELECT MAX(sales.ord_date) FROM sales)
```

Delete

```
DELETE [FROM] {table_name | view_name} [WHERE clause]
```

```
DELETE authors
```

```
DELETE FROM authors
WHERE au_lname = 'McBadden'
```

T-SQL

T-SQL è una estensione dello standard SQL e consente di facilitare lo sviluppo di applicazioni e riduce la comunicazione tra applicazioni client e server. Rispetto allo standard SQL, T-SQL dispone di controllo di flusso, cicli, gestione degli errori.

If

```
IF Boolean_expression
    {sql_statement | statement_block}
[ELSE
    {sql_statement | statement_block}]
```

While

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

BATCH, STORED PROCEDURE E TRIGGER

Bacth

Un bacth è costituito da uno o più comandi SQL inviati ed eseguiti insieme. Il batch è essenzialmente un modo per raggruppare ed inviare contemporaneamente più istruzione dal lato client verso il lato server, e questo invio può risultare più efficace che non l'invio separato di singoli comandi. Se si usa Query Analyzer, tutti i comandi contenuti nella finestra sono un batch e sono inviati assieme. L'esecuzione completa o parziale del batch dipende da numerosi fattori:

- Se è presente un errore di sintassi, tutto il batch è annullato
- Se è presente un nome di tabella errata, il batch viene eseguito solo fino al punto dell'errore: questo perché SQL Server non risolve i nomi delle tabelle prima della fase di esecuzione

Transazioni

Anche una transazione, come un batch, può includere diversi comandi SQL. Tuttavia, la differenza fondamentale tra batch e transazioni è che mentre un batch è un concetto essenzialmente legato al lato client, una transazione è invece un concetto legato al lato server. La differenza sostanziale è che le modifiche apportate da una transazione sono rese permanenti solamente al raggiungendo della fine della transazione con successo (COMMIT), altrimenti se si verifica un errore tutte le modifiche sono annullate mediante un ROLLBACK e si ritorna allo stato precedente all'avvio dell'esecuzione della transazione.
I comandi per definire una transazione sono:
BEGIN TRANSACTION
ROLLBACK TRANSACTION
COMMIT TRANSACTION

Stored procedure

Le stored procedure sono delle procedure SQL memorizzate nella cache di SQL Server. La prima volta che vengono eseguite sono ottimizzate e il piano di ottimizzazione è memorizzato. E' possibile utilizzare il T-SQL per la loro definizione e sono in grado di accettare parametri.

```
CREATE PROCEDURE] procedure_name [;number]
    [
        {@parameter data_type} [VARYING] [= default] [OUTPUT]
    ]
    [,...n]
[WITH
    {
        RECOMPILE
        | ENCRYPTION
        | RECOMPILE, ENCRYPTION
    }
]
[FOR REPLICATION]
AS
    sql_statement [...n]
```

I trigger

I trigger sono tipi speciali di stored procedure che vengono avviati in base ad un evento INSERT, UPDATE o DELETE che avviene sulla tabella. Non è prevista l'attivazione diretta dei trigger. I trigger sono usati per:

- Applicare regole di integrità dei dati
- Mantenere aggiornati i totali e le colonne calcolate
- Implementare un'operazione referenziale (come la cancellazione a catena)
- Conservare un record di controllo delle modifiche
- Richiamare operazioni esterne

Un trigger viene eseguito una sola volta per ogni istruzione INSERT, UPDATE o DELETE, indipendentemente dal numero di righe interessate (è possibile controllare il numero di righe con @@ROWCOUNT).
Il trigger viene attivato dopo l'esecuzione dell'istruzione di modifica dei dati ma prima che venga eseguito il commit delle modifiche nel database. L'istruzione e le modifiche apportate nel trigger costituiscono una transazione implicita, pertanto il trigger può annullare le modifiche dei dati che lo hanno attivato.
Il trigger può accedere ad un'immagine dei dati prima e dopo le modifiche mediante le tabelle *inserted* e *deleted* (che possono essere solo lette, ma non modificate direttamente e contengono tante righe quante sono quelle modificate dalla transazione in atto). In tal modo è possibile controllare i valori prima e dopo la modifiche ed intraprendere le necessarie azioni.
E' importante tenere presente che se un trigger modifica i dati della tabella in cui si trova e se per la modifica si utilizza la stessa operazione che ha fatto scattare il trigger (INSERT, UPDATE o DELETE) il trigger non scatta nuovamente. E' possibile modificare questo comportamento permettendo l'uso di trigger ricorsivi (comunque non più di 32 livelli di nidificazione).
Una delle operazioni possibili è proprio quella dell'annullamento del trigger, che comporta l'annullamento dell'intero batch. Per far questo si utilizza l'istruzione ROLLBACK TRANSACTION.
I trigger possono anche modificare altre tabelle, attivando i relativi trigger (fino ad un massimo di 32 livelli).
Per generare dei messaggi d'errore è possibile utilizzare la funzione RAISERROR:

```
RAISERROR messaggio, gravità, stato
```

dove per gravità è possibile utilizzare numeri da 0 a 18, ma il 14 si utilizza per messaggi informativi, il 15 per gli avvisi e il 16 per gli errori.

```
CREATE TRIGGER trigger_name
ON table
[WITH ENCRYPTION]
{
    {FOR { [DELETE] [,] [INSERT] [,] [UPDATE] }
        [WITH APPEND]
        [NOT FOR REPLICATION]
        AS
            sql_statement [...n]
    }
    |
```

```
    {FOR { [INSERT] [,] [UPDATE] }
        [WITH APPEND]
        [NOT FOR REPLICATION]
        AS
        {    IF UPDATE (column)
            [{AND | OR} UPDATE (column)]
                [...n]
            | IF (COLUMNS_UPDATED() {bitwise_operator} updated_bitmask)
                { comparison_operator} column_bitmask [...n]
        }
            sql_statement [ ...n]
    }
}
```

GLOSSARIO

DDL: Data Definition Language, linguaggio per la definizione degli oggetti del database.
DML: Data Manipulation Language o query language, consente l'interrogazione e la modifica dei dati del database.
Transazione: unità fondamentale di lavoro dei DBMS composta di diversi comandi DML.
Transazioni concorrenti: sono due transazioni che avvengono nello stesso istante di tempo.
Stato consistente: stato che soddisfa tutti i vincoli di integrità.
Stato corretto: stato consistente e coerente con una situazione ammissibile della realtà. Se ad esempio sono presenti dei vincoli di integrità che non modellano bene la realtà, è possibile essere in uno stato consistente ma non corretto.