ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

NOME CORSO

Fintech Software Developer

Unità Formativa (UF): Basi di dati SQL

Docente: Durando Giulio

Titolo argomento: Tipi di dati di SQL SERVER

Tipi di dati in SQL SERVER

In SQL Server a ogni colonna, variabile locale, espressione e parametro è associato un tipo di dati. Un tipo di dati è un attributo che specifica il tipo di dati che l'oggetto può contenere, ovvero numeri interi, caratteri, valute, date e ore, stringhe binarie e così via.

In SQL Server è disponibile un set di tipi di dati di sistema che definisce tutti i tipi di dati utilizzabili con SQL Server. È anche possibile definire tipi di dati personalizzati in Transact-SQL o Microsoft .NET Framework. I tipi di dati alias sono basati sui tipi di dati di sistema. I tipi definiti dall'utente ottengono le caratteristiche dai metodi e dagli operatori di una classe creata usando uno dei linguaggi di programmazione supportati da .NET Framework.

Quando due espressioni con tipi di dati, regole di confronto, precisione, scala o lunghezza diversi vengono combinati mediante un operatore, le caratteristiche del risultato vengono determinate come descritto di seguito.

- Il tipo di dati del risultato viene determinato applicando le regole sulla precedenza dei tipi di dati ai tipi di dati delle espressioni di input.
- Le regole di confronto del risultato sono determinate dalle regole di precedenza delle regole di confronto quando il tipo di dati del risultato è **char**, **varchar**, **text**, **nchar**, **nvarchar** o **ntext**.
- La precisione, la scala e la lunghezza del risultato dipendono dalla precisione, dalla scala e dalla lunghezza delle espressioni di input.

In SQL Server sono disponibili sinonimi dei tipi di dati per la compatibilità con ISO.

Categorie dei tipi di dati

I tipi di dati di SQL Server sono organizzati nelle categorie seguenti:

- **Dati numerici esatti**
- **Stringhe di testo Unicode**
- **Numerici approssimati**
- **Stringhe binarie**
- **Date e Time**
- **Altri tipi di dati**
- **Stringhe di caratteri**

In SQL Server, a seconda delle caratteristiche relative all'archiviazione, alcuni tipi di dati appartengono ai gruppi seguenti:

- Tipi di dati per valori di grandi dimensioni: **varchar(max)** e **nvarchar(max)**
- Tipi di dati per oggetti di grandi dimensioni: **text**, **ntext**, **image**, **varbinary(max)** e **xml**

Dati numerici esatti

Tipi di dati numerici esatti che utilizzano dati interi. Per risparmiare spazio nel database, conviene usare il tipo di dati più piccolo che può contenere in modo affidabile tutti i valori possibili. Ad esempio, tinyint sarebbe sufficiente per l'età di una persona perché nessuno raggiunge un'età maggiore di 255 anni. Ma tinyint non sarebbe sufficiente per la durata di un edificio perché un edificio può avere più di 255 anni.

bigint	numeri interi compresi tra	-2^{63} e $2^{63}-1$	occupa 8 byte
int	numeri interi compresi tra	-2^{31} e $2^{31}-1$	occupa 4 byte
smallint	numeri interi compresi tra	-2^{15} e $2^{15}-1$	occupa 2 byte
tinyint	numeri interi compresi tra	0 e 255	occupa 1 byte

int è il tipo di dati integer primario in SQL Server.

bigint è stato progettato per essere utilizzato quando i valori interi potrebbero non rientrare nell'intervallo supportato dal tipo di dati **int**.

Nell'ordine di precedenza dei tipi di dati **bigint** è compreso tra i tipi di dati **smallmoney** e **int**.

Le funzioni restituiscono **bigint** solo se l'espressione per il parametro è un tipo di dati **bigint**. SQL Server non promuove automaticamente altri tipi di dati Integer (**tinyint**, **smallint** e **int**) to **bigint**.

Quando si utilizzano gli operatori aritmetici +, -, *, / o % per eseguire conversioni implicite o esplicite di costanti di tipo **int**, **smallint**, **tinyint**, o **bigint** per tipi di dati **float**, **real**, **decimal** o **numeric**, le regole applicate da SQL Server per calcolare il tipo di dati e la precisione dei risultati delle espressioni variano a seconda del fatto che per la query sia attivata o meno la parametrizzazione automatica.

Per questo motivo, in alcuni casi espressioni simili nelle query possono dare risultati diversi. Se per una query non è attivata la parametrizzazione automatica, il valore costante viene prima convertito nel tipo **numerico**, la cui precisione è sufficiente a contenere il valore della costante, quindi viene convertito nel tipo di dati specificato. Il valore costante 1, ad esempio, viene convertito in **numerico (1, 0)** e il valore costante 250 viene convertito in **numerico (3, 0)**.

Se per una query è attivata la parametrizzazione automatica, il valore costante viene sempre convertito in **numerico (10, 0)** prima di essere convertito nel tipo di dati finale. Se viene utilizzato l'operatore /, sia la precisione del tipo di risultati che il valore del risultato di query simili possono essere diversi. Ad esempio, il valore del risultato di una query con parametrizzazione automatica che include l'espressione SELECT CAST (1.0 / 7 AS float) è diverso dal valore del risultato della stessa query ma senza parametrizzazione automatica, perché i risultati della query con parametrizzazione automatica vengono troncati per rientrare nel tipo di dati **numerico (10, 0)** .

Tipi di dati numerici con precisione e scala fisse.

decimal

numeric

Decimal e Numeric sono sinonimi e possono essere usati in modo intercambiabile.

decimal[**(** *p*[**,** *s*] **)**] e **numeric**[**(** *p*[**,** *s*] **)**]

Numeri con precisione e scala fisse. Se viene utilizzata la precisione massima, i valori validi sono compresi nell'intervallo da - 10^38 +1 a 10^38 - 1. Dal punto di vista funzionale, **numeric** è identico a **decimal**.

Argomenti

- p (precisione): numero massimo totale di cifre decimali da archiviare. Include le cifre sia a destra sia a sinistra del separatore decimale. La precisione deve essere un valore compreso tra 1 e la precisione massima di 38. La precisione predefinita è 18.
- *s* (scala): numero massimo di cifre decimali che da archiviare a destra del separatore decimale. Questo numero viene sottratto da *p* per determinare il numero massimo di cifre a sinistra del separatore decimale. La scala deve essere un valore compreso tra 0 e *p* e può essere specificato solo se viene specificata la precisione. La scala predefinita è 0. Di conseguenza, 0 <= *s* <= *p*.

Le dimensioni massime di archiviazione variano a seconda della precisione.

Precisione	Byte per l'archiviazione
1 - 9	5
10-19	9
20-28	13
29-38	17

Per i tipi di dati **decimal** e **numeric**, SQL Server considera ogni combinazione di precisione e scala un tipo di dati diverso. **decimal(5,5)** e **decimal(5,0)** sono ad esempio considerati tipi di dati diversi.

Nelle istruzioni Transact-SQL una costante con separatore decimale viene convertita automaticamente in un valore di tipo **numeric** in base ai valori di precisione e scala minimi necessari. Ad esempio, la costante 12.345 viene convertita in un valore **numeric** con precisione 5 e scala 3.

La conversione da **decimal** o **numeric** in **float** o **real** può determinare una perdita di precisione. La conversione da **int**, **smallint**, **tinyint**, **float**, **real**, **money** o **smallmoney** in **decimal** o **numeric** può determinare un overflow.

Per impostazione predefinita, quando si converte un numero in un valore **decimal** o **numeric** con precisione e scala inferiori, in SQL Server viene applicato l'arrotondamento.

bit

tipo di dati integer che può accettare un valore di 1, 0 o NULL.

Il motore di database di SQL Server ottimizza l'archiviazione delle colonne di tipo **bit**. Se una tabella contiene al massimo 8 colonne di tipo **bit**, le colonne vengono archiviate come singolo byte. Se la tabella contiene da 9 a 16 colonne di tipo **bit**, le colonne vengono archiviate in 2 byte e così via.

I valori di stringa TRUE e FALSE possono essere convertiti in valori di tipo **bit**: TRUE viene convertito in 1 e FALSE viene convertito in 0.

money valori tra -922.337.203.685.477,58 a 922.337.203.685.477,580 occupa 8 byte

smallmoney valori tra -214.748,36 e 214.748,36 occupa 4 byte

I tipi di dati **money** e **smallmoney** sono caratterizzati da una precisione pari a dieci millesimi delle unità monetarie rappresentate. Per Informatica i tipi di dati **money** e **smallmoney** sono caratterizzati da una precisione pari a un centesimo delle unità monetarie rappresentate.

Per separare le unità di valuta parziali, ad esempio i centesimi, da quelle intere, utilizzare il punto. Ad esempio, 2.15 indica 2 dollari e 15 centesimi.

Non è necessario racchiudere i dati di tipo valuta tra virgolette singole ('). È importante tenere presente che anche se è possibile specificare un simbolo di valuta che precede i valori di valuta, in SQL Server le informazioni relative alla valuta archiviate non sono associate al simbolo, ma è presente solo il valore numerico.

Nella conversione dal tipo di dati Integer a **money** le unità vengono interpretate come unità di valuta. Ad esempio, il valore Integer 4 viene convertito nell'equivalente di 4 unità di valuta per il tipo **money**.

Numerici approssimati

I tipi di dati numerici approssimati devono essere utilizzati con dati numerici a virgola mobile. I dati a virgola mobile sono approssimati. Pertanto, non tutti i valori nell'intervallo del tipo di dati possono essere rappresentati in modo esatto..

float	Valori compresi tra $-1{,}79E^{+308}$ e $-2{,}23E^{-308}$, 0 e tra $2{,}23^{E-308}$ e $1{,}79^{E+308}$	Dipende dal valore di n
real	Da - 3,40E + 38 a -1,18E - 38, 0 e da 1,18E - 38 a 3,40E + 38	4 byte

I valori di tipo **float** vengono troncati in fase di conversione in un tipo di dati Integer.

Durante la conversione dal tipo di dati **float** o **real** a dati carattere, l'uso della funzione stringa STR è in genere più pratico rispetto all'uso di CAST( ). STR infatti garantisce un maggiore controllo sul formato

Date e Time

Definisce una data in SQL Server

date
il formato predefinito dei valori letterali stringa è 'YYYY-MM-DD' occupa 3 byte

datetime
Definisce una data costituita dalla combinazione di un'ora del giorno e di secondi frazionari ed espressa nel formato 24 ore.

datetime2
Definisce una data costituita dalla combinazione di un'ora del giorno espressa nel formato 24 ore. **datetime2** può essere considerato un'estensione del tipo **datetime** esistente con un più ampio intervallo di date, una maggiore precisione frazionaria predefinita e una precisione specificata dall'utente facoltativa.

Il formato predefinito dei valori letterali stringa è '**YYYY-MM-DD' hh:mm:ss[.secondi frazionari]**

datetimeoffset
Definisce una data combinata con un'ora di un giorno in base a un orologio di 24 ore, ad esempio datetime2,e aggiunge la consapevolezza del fuso orario in base all'ora UTC (Universal Time Coordinate o Greenwich Mean Time). La differenza di fuso orario specifica la differenza di fuso orario rispetto all'ora UTC per un valore **time** o **datetime**. La differenza di fuso orario può essere rappresentata nel formato [+|-] hh:mm:

Il formato predefinito è **YYYY-MM-DD hh:mm:ss[.nnnnnnn] [+|-]hh:mm** (2012-10-20 12:32:00 +01:00)

smalldatetime
Definisce una data combinata con un'ora del giorno. L'ora si basa su un formato di 24 ore, con secondi sempre a zero (: 00) e senza secondi frazionari.

time
Definisce l'ora di un giorno. Il fuso orario non viene preso in considerazione e il formato è basato sulle 24 ore.

Stringhe di caratteri

Sono tipi di dati character a dimensione fissa **char** o a dimensione variabile **varchar**. A partire da SQL Server 2019 (15.x), quando si usano regole di confronto che supportano UTF-8, questi tipi di dati archiviano l'intera gamma dei dati di tipo carattere Unicode e usano la codifica dei caratteri UTF-8 . Se si specificano regole di confronto non UTF-8, questi tipi di dati archiviano solo un subset dei caratteri supportati dalla tabella codici corrispondente di tali regole di confronto. Se non si specifica *n* in un'istruzione di definizione dei dati o di dichiarazione di variabili, la lunghezza predefinita è 1

char [(n)]

Dati stringa a dimensione fissa. *n* definisce le dimensioni della stringa in byte e deve essere un valore compreso tra 1 e 8.000. Per i set di caratteri con codifica a byte singolo, ad esempio *Latin*, le dimensioni di archiviazione sono pari a *n* byte e anche il numero di caratteri che possono essere

archiviati è *n*. Per i set di caratteri con codifica multibyte, le dimensioni di archiviazione sono di nuovo *n* byte, ma il numero di caratteri che possono essere archiviati può essere inferiore a *n*. Il sinonimo ISO per **char** è **character**.

varchar [(n | max)]
Dati stringa a dimensione variabile. *n* serve a definire la dimensione della stringa in byte, che può essere un valore compreso tra 1 e 8000, si deve usare **max** per indicare la dimensione massima di archiviazione di un vincolo di colonna pari a 2^31-1 byte (2 GB). Per i set di caratteri con codifica a byte singolo, ad esempio *Latin*, le dimensioni di archiviazione sono pari a *n* byte + 2 byte e anche il numero di caratteri che possono essere archiviati è *n*. Per i set di caratteri con codifica multibyte, le dimensioni di archiviazione sono di nuovo *n* byte + 2 byte, ma il numero di caratteri che possono essere archiviati può essere inferiore a *n*. I sinonimi ISO per **varchar** sono **charvarying** o **charactervarying**.

Stringhe di testo Unicode

nchar [(n)]
Dati stringa a dimensione fissa. *n* definisce le dimensioni della stringa in coppie di byte e deve essere un valore compreso tra 1 e 4.000. Le dimensioni di archiviazione, espresse in byte, sono pari al doppio di *n*. Per la codifica UCS-2, le dimensioni di archiviazione sono pari al doppio di *n* byte e anche il numero di caratteri che possono essere archiviati è *n*. Per la codifica UTF-16, le dimensioni di archiviazione sono ancora pari al doppio di *n* byte, ma il numero di caratteri che possono essere archiviati può essere inferiore a *n* perché i caratteri supplementari usano due coppie di byte, dette anche coppie di surrogati. I sinonimi ISO per **nchar** sono **national char** e **national character**.

nvarchar [(n | max)]
Dati stringa a dimensione variabile. *n* definisce le dimensioni della stringa in coppie di byte e può essere un valore compreso tra 1 e 4.000. **max** indica che le dimensioni di archiviazione massime sono pari a 2^30-1 caratteri (2 GB). Le dimensioni di archiviazione, espresse in byte, sono pari al doppio di *n* byte + 2 byte. I sinonimi ISO per **nvarchar** sono **national char varying** e **national character varying**.

ntext
Dati Unicode a lunghezza variabile con lunghezza massima della stringa di 2^30 - 1 (1.073.741.823) byte. Le dimensioni dello spazio di archiviazione, espresse in byte, sono pari al doppio della lunghezza della stringa immessa. Il sinonimo ISO per **ntext** è **national text**.

text
Dati non Unicode a lunghezza variabile nella tabella codici del server con lunghezza massima della stringa di 2^31-1 (2.147.483.647). Quando nella tabella codici del server vengono utilizzati caratteri DBCS, lo spazio di archiviazione è sempre pari a 2.147.483.647 byte. In base alla stringa di caratteri, le dimensioni dello spazio di archiviazione possono essere minori di 2.147.483.647 byte.

image
Dati binari a lunghezza variabile da 0 a 2^31-1 (2.147.483.647) byte.