Microsoft® SQL Server® 2008 Standard

# Guida per principianti

*Una introduzione a SQL Server 2008 Management Studio*

**Prof. A. Zoccheddu**

# INSTALLAZIONE DI SQL SERVER

## 1 Due parole sui RDBMS

SQL Server è un sistema di gestione di database relazionali (**RDBMS – Relational DataBase Management System**) distribuito da Microsoft ® che si posiziona come strumento potente ma a prezzi accessibili in un mercato piuttosto competitivo dove si ritrovano prodotti commerciali e Open Source tra cui dobbiamo ricordare Oracle e mySQL.

Oracle è un prestigioso RDBMS molto diffuso e assai potente e, sebbene sia un prodotto più complesso da installare e da amministrare rispetto a SQL Server, costituisce un magnifico prodotto con cui implementare database critici. Oracle è spesso scelto da organizzazioni di grandi dimensioni sia per le esigenze di ampie e potenti risorse necessarie per il suo funzionamento, sia per le sue qualità di scalabilità, di elevate prestazioni e per la notevole offerta di strumenti addizionali che si possono utilizzare. Oracle si presenta come un pacchetto macchinoso, impegnativo per i programmatori e gli amministratori, ostico da installare e non sempre intuitivo da utilizzare.

MySQL è un RDBMS composto da un client con interfaccia a riga di comando e un server, entrambi disponibili sia per sistemi Unix-like (come GNU/Linux) sia per Windows, anche se prevale un suo utilizzo in ambito Linux. Il codice di MySQL risale al 1979, ma è solo dal 1996 che viene distribuita una versione che supporta SQL. Dal 2008 l'azienda produttrice è stata acquistata da Sun Microsystems ma nel 2010 quest'ultima è stata rilevata da Oracle Corporation.

Adaptive Server Enterprise è un RDBMS che deriva da Sybase SQL Server, ma ha cambiato nome in Adaptive Server Enterprise (ASE) nel 1996 per evitare confusioni con SQL Server, il RDBMS di Microsoft. Nel 2001 è stata rilasciata la versione 12.5 che ha aggiunto alcune funzionalità come l'allocazione dinamica della memoria ed il supporto per XML ed SSL.

I sistemi RDBMS implementano linguaggi di programmazione analoghi, basati sul linguaggio SQL (Structured Query Language, Linguaggio Strutturato di Interrogazione) che vede l'ultimo standard riconosciuto col nome di ANSI-92. Tuttavia ogni RDBMS ha una sua specifica sintassi e gli RDBMS sopra menzionati, sebbene basati su SQL-92, utilizzano variazioni del linguaggio detti anche dialetti SQL. La sintassi per interrogare i dati e per svolgere le altre operazioni è molto simile in tutti i quattro sistemi, ma ciascun RDBMS ha una propria sintassi peculiare per compiere le operazioni.

## 2 Introduzione a MS SQL Server

MS SQL Server può essere una la soluzione adatta per organizzazioni di ampie dimensioni, ma anche per settori più contenuti; in generale offre soluzioni a costi minori rispetto ad Oracle, più affidabili, prestanti e flessibili rispetto sa Access, più documentate e con servizi di assistenza commerciali rispetto a prodotto Open Source. MS SQL Server è abbastanza agevole da installare e ne vengono distribuite diverse versioni che offrono diverse funzionalità in base alle esigenze delle specifiche organizzazioni che intendano utilizzarlo.

Consultando Wikipedia è possibile verificare l'elenco delle le versioni e delle distribuzioni negli anni:

- 1993 - SQL Server 4.2
- 1995 - SQL Server 6.0, nome in codice SQL95
- 1996 - SQL Server 6.5, nome in codice Hydra
- 1998 - SQL Server 7.0, nome in codice Sphinx
- 1999 - SQL Server 7.0 OLAP, nome in codice Plato
- 2000 - SQL Server 2000 32-bit, nome in codice Shiloh
- 2003 - SQL Server 2000 64-bit, nome in codice Liberty
- 2005 - SQL Server 2005 (sia 32-bit che 64-bit), nome in codice Yukon (disponibile la versione Express gratuita)
- 2008 - SQL Server 2008 (sia 32-bit che 64-bit), nome in codice Katmai
- 2010 - SQL Server 2008 R2 (sia 32-bit che 64-bit), nome in codice Kilimangiaro
- 2012 - SQL Server 2012 (sia 32-bit che 64-bit), nome in codice Denali

In questo documento si esaminerà con maggiore dettaglio la sola versione di SQL Server 2008 (edita nel 2008) che vede integrato sia lo Studio Manager (interfaccia visuale di amministrazione) sia un ampio numero di formati di dati e di servizi.

L'infrastruttura è la stessa di SQL Server 2005 ma il sistema include nuovi tipi di dati, nuove funzionalità e la possibilità di utilizzare LINQ (il Language Integrated Query); consente di operare col formato XML e prevede molte funzioni di sistema sui dati (stringhe, date, valori di sistema).

Secondo quanto riportato dalla guida in linea della Microsoft sono presenti le seguenti edizioni di SQL Server 2008 http://msdn.microsoft.com/it-it/library/ms143506%28v=sql.100%29.aspx :

- SQL Server 2008 Standard (32 bit)
- **SQL Server 2008 Standard (64 bit) x64 (versione adottata per costruire questo documento)**
- SQL Server 2008 Enterprise (32 bit)
- SQL Server 2008 Enterprise (64 bit) IA64
- SQL Server 2008 Enterprise (64 bit) x64
- cui si aggiunge SQL Server Enterprise Evaluation che è offerta per un periodo di prova di 180 giorni.

Queste edizioni si differenziano per la gamma di servizi offerti da SQL Server 2008:

- **Standard** ♦ È la versione base rivolta alla gestione dati e a piccole organizzazioni, abbastanza facile da usare e da gestire e consente di costruire applicazioni essenziali che possono essere mirati a singoli dipartimenti societari.
- **Enterprise** ♦ È la piattaforma integrale rivolta a organizzazioni ampie e articolate, in grado di offrire strumenti scalari, sicurezza, analisi e report avanzati e che consente di realizzare applicazioni data warehousing. Questa edizione consente di realizzare applicativi concorrenti e distribuiti su larga scala.

A queste edizioni si aggiungono alcune edizioni *specializzate* di SQL Server 2008:

- SQL Server 2008 Express, SQL Server 2008 Express with Tools e SQL Server 2008 Express with Advanced Services (32 bit e 64 bit)
- Workgroup (64 bit) x64 // Workgroup (32 bit)
- Developer (64 bit) IA64 Questa edizione di SQL Server 2008 non è disponibile nella versione italiana
- Developer (64 bit) x64 // Developer (32 bit)
- Web (64 bit) x64 // Web (32 bit)

In particolare le edizioni specializzate sono rivolte a:

- **Express** ♦ È l'edizione gratuita di SQL Server rivolta a chi voglia imparare a costruire applicazioni per PC domestici o per piccoli server.
- **Workgroup** (32 e 64) ♦ Piattaforma di gestione dati e creazione report che offre funzionalità di gestione e sincronizzazione remota sicure per l'esecuzione di applicazioni di filiale. Questa edizione include caratteristiche chiave del database ed è facilmente aggiornabile all'edizione Standard o Enteprise.
- **Developer** (32 e 64) ♦ Rivolto agli sviluppatori che vogliono realizzare e provare vari prototipi applicativi con SQL Server. Questa edizione è equivalente a SQL Server Enterprise, sebbene la licenza sia limitata solo allo sviluppo, al test e alle demo. Le applicazioni e i database sviluppati su questa edizione possono essere facilmente upgradati alla versione integrale Enterprise.
- **Web** (32 e 64) ♦ Appositamente progettata per gli ambienti dedicati ai servizi web ad elevata disponibilità eseguibili su Windows Server. SQL Server 2008 Web fornisce qualsiasi tipo di supporto per tutte le applicazioni web.

## 2-1 Prerequisiti di installazione

I requisiti suggeriti da Microsoft per l'istallazione di SQL Server 2008 Standard (64 bit) x64 sono i seguenti: http://msdn.microsoft.com/it-it/library/ms143506%28v=sql.100%29.aspx

| Componente | Requisito |
|---|---|
| **CPU** | Velocità minima del processore: 1,4 GHz<br>Velocità consigliata del processore: almeno 2 GHz<br>CPU Minima: AMD Opteron, AMD Athlon 64,<br>Intel Xeon con supporto Intel EM64T, Intel Pentium IV con supporto EM64T |
| **RAM** | Ampiezza minima di RAM: 512 MB<br>Ampiezza consigliata di RAM: almeno 2,048 GB |
| **Disco rigido** | Vedi sotto … |
| **Sistema operativo** | Windows XP Professional x64<br>Windows Server 2003 SP2 64 bit x64 ed. Standard1, Datacenter1, Enterprise1<br>Windows Vista Ultimate x64, Enterprise x64, Business x64<br>Windows Server 2008 x64 Web1<br>Windows Server 2008 x64 Standard, Datacenter, Enterprise,<br>Windows Server 2008 x64 versioni senza Hyper-V1<br>Windows Small Business Server 20082<br>Windows Server 2008 per Windows Essential Server Solutions2 (anche senza Hyper-V)<br>Windows 7 64 bit x64 Ultimate1.3, Enterprise1.3, Professional1.3<br>Windows 2008 R2 64 bit x64 Web1.3, Standard1.3, Enterprise1.3, Datacenter1.3<br>Windows 2008 R2 Foundation Server1.3<br>Microsoft ® avverte che SQL Server 2008 non è supportato nelle installazioni di base di Windows Server 2008 e Windows Server 2008 R2. |

**Framework** Durante l'installazione di SQL Server vengono installati i componenti software seguenti, necessari per il funzionamento del prodotto:

- .NET Framework 3.5 SP11
- SQL Server Native Client
- File di supporto dell'installazione di SQL Server

**Software aggiuntivo** Per il programma di installazione di SQL Server è necessario Microsoft Windows Installer 4.5 o versione successiva.

**Software di rete** I requisiti del software di rete per le versioni a 64 bit di SQL Server 2008 corrispondono a quelli delle versioni a 32 bit. I sistemi operativi supportati includono software di rete integrato. Le istanze autonome denominate e le istanze predefinite supportano i protocolli di rete seguenti:

- Shared Memory
- Named Pipes
- TCP/IP
- VIA

### 2-2 Requisiti di spazio su disco rigido (32 bit e 64 bit)

Durante l'installazione di SQL Server 2008 vengono creati alcuni file temporanei nell'unità di sistema; sono necessari almeno 2 GB di spazio su disco rigido per allocare temporaneamente tali file e per eseguire il programma di installazione (sia per installare sia per aggiornare SQL Server). Tale requisito deve essere soddisfatto anche se tutti i componenti di SQL Server vengono installati in un'unità di sistema non predefinita.

I requisiti effettivi di spazio su disco rigido per ospitare il programma funzionante variano in base alla configurazione del sistema e alle caratteristiche selezionate per l'installazione; nella seguente tabella sono riportate i valori di spazio su disco richiesti per i singoli componenti di SQL Server 2008:

| Funzionalità | Requisito di spazio su disco rigido |
|---|---|
| Motore di database e file di dati, replica e ricerca full-text | 280 MB |
| Analysis Services e file di dati | 90 MB |
| Reporting Services e Gestione report | 120 MB |
| Integration Services | 120 MB |
| Componenti client | 850 MB |
| Documentazione in linea di SQL Server e di SQL Server Compact Edition | 240 MB |
| **TOTALE SPAZIO DISCO PER INSTALLAZIONE COMPLETA** | **1.700 MB** |

### 2-3 Installazione di SQL Server in un controller di dominio

Per motivi di protezione, Microsoft consiglia di non installare SQL Server 2008 in un controller di dominio.

# 3 Sistemi di database

Un Data Base Management System (acronimo: DBMS) è un sistema software progettato per consentire la creazione e la gestione (da parte di un amministratore) e l'utilizzo e l'interrogazione efficace (da parte di uno o più utenti) di database (ovvero di collezioni di dati strutturati soggetti a vincoli).

Il DBMS deve essere installato su una architettura hardware che può essere un semplice computer, un server oppure un sistema integrato di server. I DBMS svolgono un ruolo primario per la gestione di numerose applicazioni informatiche, come la contabilità, la gestione delle risorse umane e la gestione di contesti specialistici e tecnici come la gestione di rete, la telefonia, la distribuzione di contenuti digitali, l'archiviazione di dati sensibili.

Un sistema di database è un pacchetto integrato di diversi componenti tra cui:

- Un insieme di programmi dedicati alla gestione dei database
- Uno o più componenti client, per consentire l'accesso degli utenti
- Uno o più motori integrati di database, per elaborare i dati e garantirne l'integrità e la consistenza
- Uno o più database, intesi come collezioni di dati legati da vicoli

Un programma dedicato alla gestione del database (Database Management System) è un software progettato ed implementato da imprese specialistiche, che prevede tutti i moduli per la gestione dei dati, dei vincoli, degli accessi e dei servizi ausiliari.

Un componente client è un software progettato anche da terze parti, con l'obiettivo di consentire agli utenti di accedere ai dati dei diversi database, con un'interfaccia agevole e semplice, e che consente di effettuare le manipolazioni dei dati ammesse (letture, scritture, modifiche e elaborazioni). L'utente può non accorgersi che i dati possono risiedere fisicamente in luoghi sparsi e che sono riaggregati per semplificarne la gestione.

I motori di database sono elementi software dedicati alla gestione efficiente dai dati contenuti in vari database fisici; il singolo client comunica con essi inviando comandi (es. query) che vengono elaborati dai motori che in seguito restituiscono il risultato al client con un protocollo analogo alla richiesta.

Un database è un insieme di dati correlati da vincoli. L'utente vede i database come collezioni di dati legati tra loro che rispondono a regole di accessibilità e di operabilità; il motore del database guarda ai dati come sequenze di byte fisici, che risiedono su archivi disposti su memorie di massa, talvolta con repliche di sicurezza, spesso in luoghi fisicamente diversi.

Ogni sistema di database deve fornire determinati meccanismi e caratteristiche fondamentali, tra cui:

1. Indipendenza fisica dei dati
2. Indipendenza logica dei dati
3. Integrità dei dati
4. Funzionalità di backup e ripristino
5. Funzionalità di sicurezza
6. Ottimizzazione delle query
7. Presidio della concorrenza
8. Interfacce utente

## 3-1 Indipendenza fisica dei dati

Il requisito di Indipendenza Fisica dei dati impone che una qualsiasi applicazione che agisce sui dati di un database non deve dipendere dalla struttura fisica dei dati memorizzati. Questa caratteristica implica che eventuali modifiche sulla struttura dei dati non richiede di dover modificare i programmi applicativi che si interfacciano con gli stessi. Per esempio se i dati memorizzati all'interno del database sono disposti secondo un certo criterio ma in seguito tale ordinamento viene variato, questa modifica fisica non influisce né sulle applicazioni esistenti, né sullo schema del database stesso.

## 3-2 Indipendenza logica dei dati

Il requisito di Indipendenza Logica dei dati significa che è possibile effettuare modifiche alla struttura logica di un database senza dover effettuare alcuna modifica ai programmi che operano sul database stesso. L'indipendenza logica può essere espressa come la possibilità di modificare lo schema logico dei dati senza modificare le viste degli utenti. Questa caratteristica permette di effettuare, in gran parte dei casi, delle modifiche nello schema logico del database senza richiedere alcun cambiamento nelle viste interessate. Altre modifiche possono essere fatte semplicemente ridefinendo le corrispondenze esistenti tra le viste e lo schema logico. In questi casi non è richiesto alcun cambiamento nei programmi applicativi. In alcuni casi il cambiamento dello schema richiede delle modifiche nelle viste utente che consistono nella cancellazione di informazioni che corrispondano ad informazioni presenti nella vista.

## 3-3 Integrità dei dati

Il requisito di integrità dei dati impone che i dati devono essere significativi sia per coerenza interna che per congruenza esterna. La coerenza interna impone che un dato deve essere significativo (per es. una distanza negativa potrebbe non avere senso) e che due dati non possano essere in contraddizione tra loro (per esempio la somma di percentuali non dovrebbe superare il 100%). Altri dati devono rispettare regole di consistenza logica (ad esempio evitare di registrare la data 30 febbraio) e altri devono rappresentare i dati della realtà modellata (il numero di biglietti venduti, gli studenti assegnati a una classe, i voti del quadrimestre).

## 3-4 Funzionalità di backup e ripristino

La funzionalità di backup deve consentire di salvare copie di sicurezza dei database; alcune tecniche consentono salvataggi periodici dell'intero database (es. su nastro), altre salvano copie dei dati frequentemente (quasi istantaneamente alle operazioni) mantenendo le copie in sincronia.

La funzionalità di ripristino (recovery) deve consentire di riportare il database in una situazione precedentemente salvata in una copia. In caso di gravi crash dei dati è possibile recuperare una situazione vecchia da una copia salvata in precedenza. In altri casi il sistema rileva in automatico anomalie e malfunzionamenti e ripristina i dati verificando la loro integrità dalle copie di sicurezza.

## 3-5 Funzionalità di sicurezza

La funzionalità di sicurezza di un database è fondata su due concetti complementari: autenticazione e autorizzazione.

L'autenticazione è il controllo di validazione dell'accesso di un cliente verificando le sue credenziali al fine di prevenire utilizzazioni illegali del sistema.

L'autorizzazione è il processo con cui si controlla su una operazione è consentita ad un cliente già autenticato e consiste generalmente in un elenco di diritti associate alle varie risorse a cui ciascun cliente è abilitato.

### 3-6 Ottimizzazione delle query

La funzionalità di ottimizzazione delle interrogazioni è la capacità di trasformare un comando generico in operazioni elementari da effettuare sui dati fisici.

Questa funzionalità è generalmente espletata da un componente chiamato ottimizzatore di interrogazioni (Query Optimizer) che analizza le diverse strategie di esecuzione per l'interrogazione dei dati e poi opta per quella più efficiente. La strategia selezionata corrisponde ad uno specifico piano di esecuzione (execution plan) della query. Si osservi che il piano di esecuzione viene spesso memorizzato per ulteriori successive esecuzioni, anche se i dati sono, nel frattempo, cambiati.

### 3-7 Presidio della concorrenza

Generalmente numerosi clienti accedono contemporaneamente agli stessi dati. Non sempre ai dati è possibile accedere parallelamente, per esempio per motivi di integrità (es. due processi, ciascuno dei quali incrementa di +1 un valore numerico).

Un componente del DBMS deve quindi garantire un accesso concorrente ai dati, preservando la loro integrità e garantendo l'efficacia delle operazioni.

### 3-8 Interfacce utente

Sebbene le interfacce non sempre sono parte integrante del sistema di database, molti database sono progettati per essere utilizzati da diversi tipi di utenti ciascuno con uno specifico livello di accesso ai dati.

Un DBMS dovrebbe offrire diverse possibilità di comunicazione dal cliente ai dati, per esempio integrando protocolli di comunicazione con cui sviluppare le varie interfacce utente rivolte ai diversi tipi di utenza. Attualmente molti di questi protocolli si stanno orientando ad applicazioni web, ma non mancano sviluppi legati a vecchi e nuovi dispositivi (interfacce visuali e console, apparecchi fissi e portatili, reti su cavo e wireless, mezzi di puntamento rapido, ecc).

# 4 Panoramica di SQL Server 2008

Rispetto alla versione 2005, l'intento di Microsoft con il rilascio di SQL Server 2008 è stato quello di semplificare e migliorare il lavoro degli amministratori (DBA) di database e dei programmatori di applicazioni che devono interfacciarsi con esso. La versione 2008 del DBMS ha aggiunto una varietà di nuove caratteristiche e funzionalità.

I DBA di SQL Server 2005 possono trovare nella versione 2008 tutti gli strumenti che già utilizzavano regolarmente nel proprio compito e noteranno che molti di essi sono stati arricchiti e migliorati. È possibile suddividere le novità principali in cinque categorie:

1. Gestione in rete
2. Performance
3. Scalabilità
4. Sicurezza
5. Sviluppo

### 4-1 Gestione in rete

Tra le novità introdotte rispetto alla versione 2005, nel 2008 si possono annoverare:

- **Policy Management**, regole per uniformare la gestione delle workstation in rete;
- **Multiple Server Interaction and Configuration Servers**, strumento per eseguire query su gruppi di server;
- **Data Collector**, per raccogliere dati di gestione da diversi server remoti.

Vediamo alcuni dettagli.

Le policy sono pacchetti di regole che possono essere imposte su un server per fare in modo che tutte le workstation collegate siano gestite allo stesso modo e presentino le stesse caratteristiche. In SQL Server questa gestione è affidata al **Policy Management**. In SQL Server 2008, con il Policy Management Framework, è molto facile creare delle policy che possano verificare regole definite a nostro piacimento. In generale conviene lasciare la regola (condizione) abilitata su tutti i database, quelli di sistema compresi.

La Policy Management permette di definire e attivare diverse regole di configurazione su uno o più server di database. Mediante tali regole l'amministratore del database può stabilire che la configurazione standard dei settaggi di ciascuna macchina venga applicata e mantenuta su ognuno dei server e dei database condizionati.

La Gestione Avanzata Criteri di gruppo di Microsoft (Advanced Group Policy Mangement, AGPM) 4.0 include inoltre nuove funzionalità sugli oggetti del tipo Criterio di Gruppo, che consentono di trovare i Criteri di gruppo, filtrare l'elenco di Criteri di gruppo visualizzati, importare ed esportare un Criterio di gruppo da e verso un'altra foresta di server e installare AGPM su computer su cui sono in esecuzione Windows® 7 e Windows Server® 2008 R2.

Per eseguire contemporaneamente un'interrogazione su dati residenti su diversi server, MS SQL server 2008 propone la **Multiple Server Interaction and Configuration Servers (MSICS)**. È possibile registrare i diversi server nel Management Studio di SQL Server ed riunirli in uno stesso gruppo. Quando in seguito è necessario applicare una policy o eseguire una query su tutti i server del gruppo basta richiedere l'operazione desiderata per l'intero gruppo. Questa funzionalità può essere configurata per restituire un insieme di risultati per ogni server oppure per unire tutti i risultati in un unico insieme.

Per consentire ad un amministratore di database di raccogliere dati di funzionamento (statistiche e prestazioni) da un diverso numero di server, MS SQL server 2008 consente di usare il **Data Collector**, la cui funzionalità dispone di un meccanismo che semplifica l'attività e facilita l'uso di componenti quali lo SQL Server Agent e lo Integration Services creando framework che raccoglie e memorizza i dati rilevanti, e consente una opportuna gestione degli errori.

## 4-2 Performance

Secondo By Rob Garrison, autore dell'articolo Performance Testing - SQL Server 2008 versus SQL Server 2005 (http://www.databasejournal.com/) i test non hanno mostrato significative differenze tra i due server, e 2008 non risulta né più lento né più veloce del 2005.

Tuttavia secondo Vincenzo Gaglio (http://www.mrwebmaster.it/sql-server/guide/guida-sql-server-2008/) SQL Server 2008 è progettato per ridurre il numero totale di blocchi generati dai vari processi e lavora su un meccanismo di partizionamento delle tabelle.

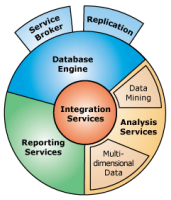
## 4-3 Scalabilità

Talvolta si dimensiona un database in funzione delle esigenze contingenti, ma che in seguito posso cambiare e richiedere un'ampiezza maggiore. SQL Server 2008 istituisce alcune nuove funzionalità che agevolano le operazioni di ridimensionamento e, in particolare, permette di usare sia la compressione di Windows che una compressione dei dati dei file dei database e dei file relativi ai log delle transazioni dove è possibile scegliere il livello di compressione a livello di riga o di pagina. Tali soluzioni evitano di installare nuovo spazio disco con una perdita di velocità contenuta.

La compressione dei dati era stata resa disponibile in SQL Server 2005 col SP2 introducendo un nuovo formato di archiviazione dei dati numerici e decimali. Questo concetto è stato esteso in SQL Server 2008 a tutti i dati di lunghezza fissa come interi, caratteri e dati in virgola mobile. La compressione riduce i costi di archiviazione e incrementa le prestazioni di interrogazione poiché riduce il carico di I/O e ottimizza la funzionalità dei buffer. SQL Server 2008 supporta sia la compressione di riga e di tabella sia per tabelle sia per gli indici. La compressione di riga ottimizza l'occupazione di memoria per valori numerici come quelli espressi dal tipo vardecimal; questa compressione riduce a zero lo spazio per valori nulli e zeri. La compressione di pagina si riferisce a insiemi di righe e sfrutta la ridondanza dei dati nelle diverse righe compresse.

In SQL 2008 è introdotto anche il gestore delle risorse (Resource Governor), che consente di gestire il carico di lavoro e l'utilizzo delle risorse di sistema di SQL Server; questa funzionalità permette di specificare un limite superiore all'uso della CPU e della memoria allocata per elaborare le applicazioni.

Con la versione 2008 SQL Server rende migliore anche la scalabilità per i Data Warehouse (DW), incontrando le esigenze dei DW per grandi organizzazioni e rendendole più agevoli. SQL Server 2008 fornisce una gamma di prodotti integrati che consentono di realizzare DW e le rispettive interrogazioni e analisi dei dati. Tra questi prodotti possiamo annoverare gli Analysis Services, gli Integration Services, e i Reporting Services. Tutte queste funzionalità contribuiscono a migliorare la scalabilità del sistema.




## 4-4 Sicurezza

Già in SQL Server 2005 si era introdotta la crittografia dei dati per migliorare la sicurezza, ma in SQL Server 2008 sono introdotte alcuni migliorie.

L'Extensible Key Management è una funzionalità che introduce ulteriori meccanismi per memorizzare in modo più sicuro le chiavi utilizzate nei procedimenti crittografici.

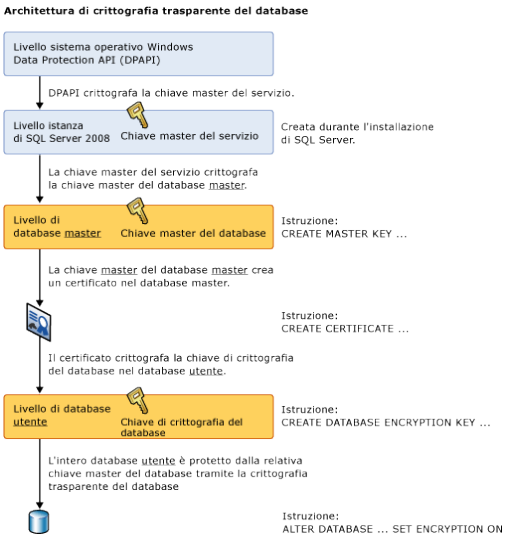
Il **Transparent Data Encryption** (TDE) è uno strumento che consente di crittografare un intero database e lo propone come metodo e non come risultato funzionale.

**Fonte: http://msdn.microsoft.com/it-it/library/bb934049.aspx**

TDE esegue la crittografia e la decrittografia I/O in tempo reale dei file di dati e di log. Per la crittografia viene utilizzata una chiave di crittografia del database (DEK), archiviata nel record di avvio del database affinché sia disponibile durante le operazioni di recupero.

La chiave di crittografia del database è una chiave simmetrica protetta con un certificato depositato nel database master del server o una chiave asimmetrica protetta da un modulo EKM.

TDE consente di proteggere i dati "non operativi", ovvero i file di dati e di log per migliorare la conformità a numerose disposizioni e normative vigenti in vari ambiti.




## 4-5 Sviluppo

SQL Server 2008 ha separato i tipi **Date** e **Time** (accorpati nel 2005) e ha introdotto i nuovi tipi di dati **Geography** e **Geometry** rivolti alla memorizzazione di dati topologici a livello di colonna. Viene aggiunto inoltre anche il tipo di dati **Filestream** destinato alla memorizzazione di BLOB (Oggetti Binari di Grandi Dimensioni) per amministrare oggetti multimediali da gestire con applicazioni di terze parti o con front-end per dispositivi remoti.

Qui inserire altri elementi di confronto tra 2005 e 2008

# 5 Installazione

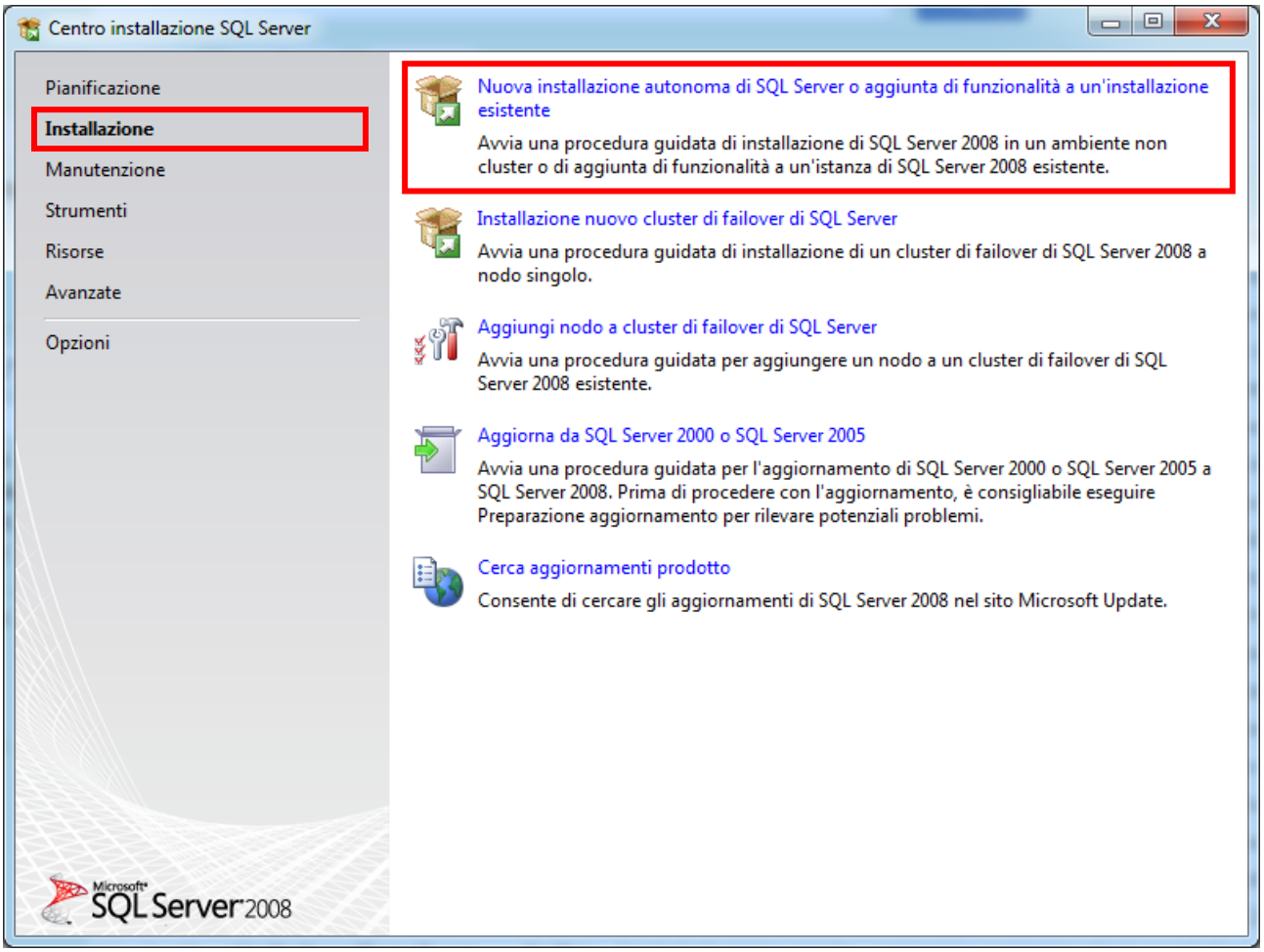
## 5-1 Wizard di Installazione di SQL Server

Se occorre installare MS SQL Server 2008 allora è possibile consultare questa sezione,altrimenti è possibile saltarla e passare all'uso di SQL Server 2008.

Per installare SQL Server 2008 è necessario essersi autenticati come utente con privilegi di amministratore sulla macchina in cui procedere con l'installazione, che provvederà alla creazione di diversi file e cartelle.

Nella prima fase dell'installazione il programma verifica se sull'elaboratore è stato già installato precedentemente il .NET Framework 3.5 altrimenti richiede la sua installazione; il Framework è utilizzato da MS SQL Server 2008 per vari scopi e consente ai programmatori di scrivere applicazioni con uno qualunque dei linguaggi del .NET (per es. Visual Studio) e di utilizzare tale codice per interagire con SQL Server.

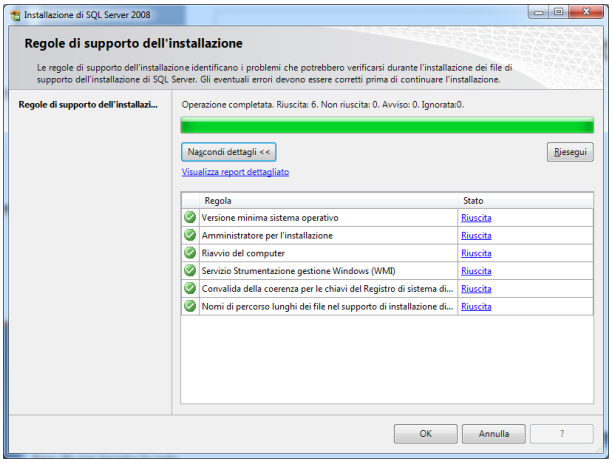
La prima finestra del wizard di installazione è la SQL *Server Installation Center*



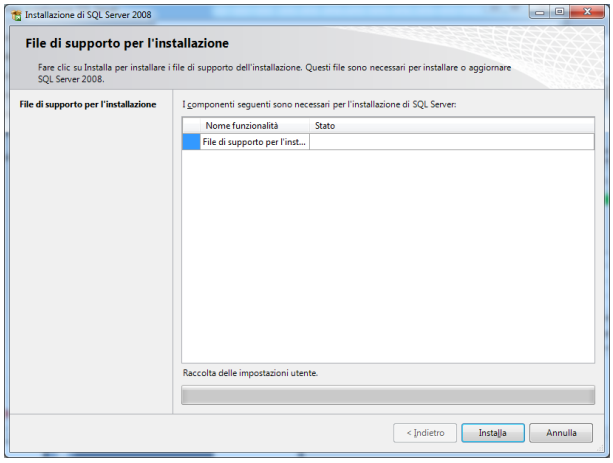

Questa finestra consente di gestire diversi aspetti di SQL Server ma ai fini di questo documento esamineremo soltanto il processo di installazione. Sul lato sinistro della finestra scegliamo il link **Installazione** e dopo selezioniamo nel riquadro a destra l'opzione **Nuova installazione autonoma di SQL Server.**

Viene effettuata una breve scansione del sistema e sono proposte alcune finestre con cui installare varie opzioni; le finestre sono riprodotte qui di seguito durante le quali sarà richiesto di inserire il codice prodotto e di accettare la licenza d'uso e infine di selezionare le caratteristiche di installazione.

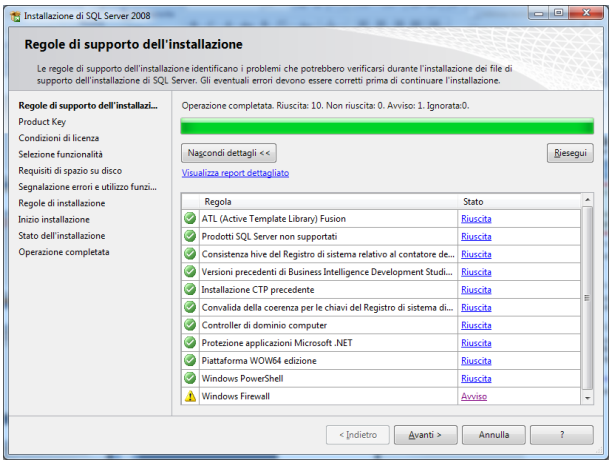
**1.** Regole di supporto di installazione (clic su OK);



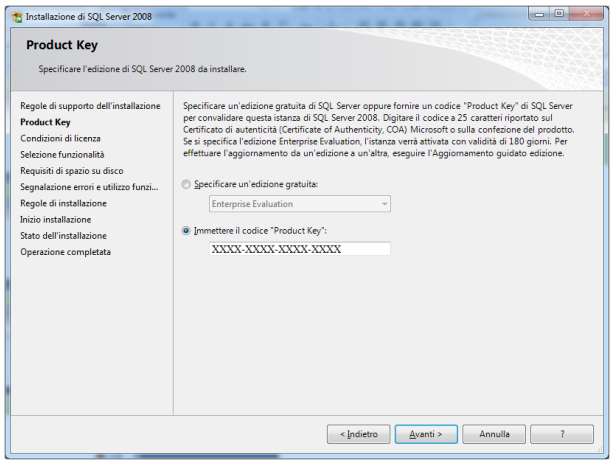

**2.** File di supporto per l'installazione (clic su Installa);



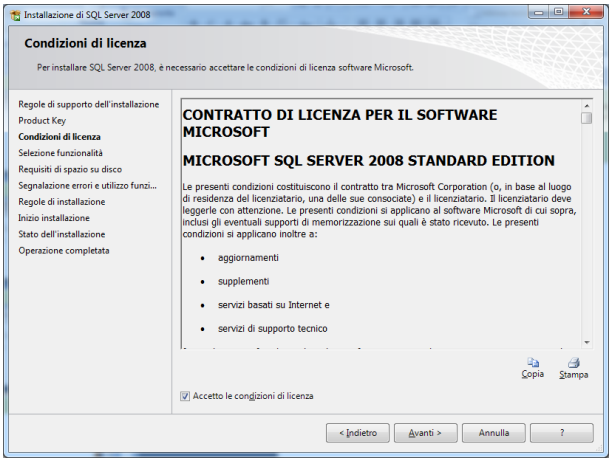

**3.** Regole di supporto per l'installazione (clic su Avanti);



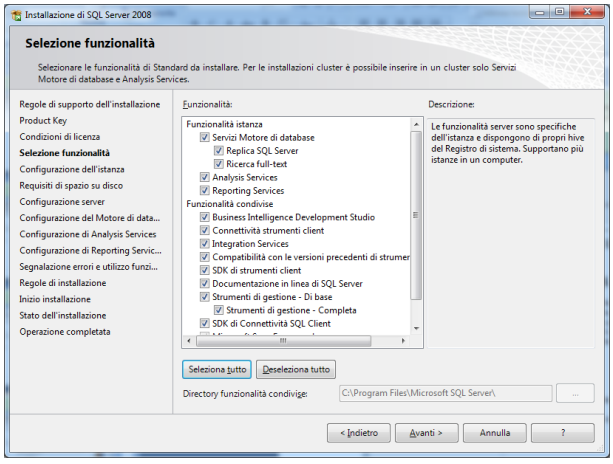

**4.** Codice Prodotto (immettere il Product Key).




**5.** Condizioni di licenza (clic su Accetto);



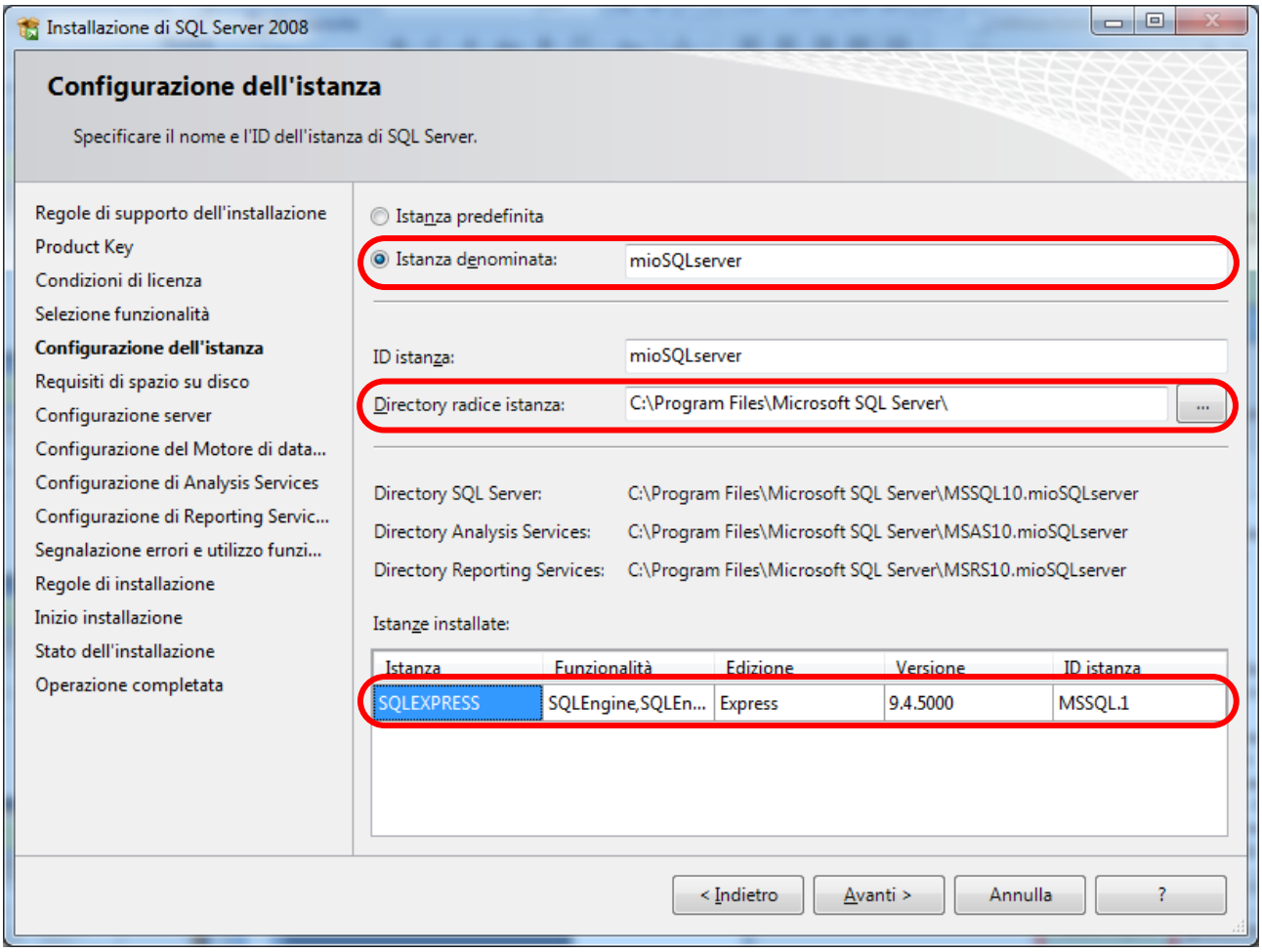

**6.** Selezione funzionalità (clic su Seleziona tutto);




L'ultima finestra riprodotta qui sopra è la Selezione dele funzionalità, in cui è necessario selezionare i componenti da installare.

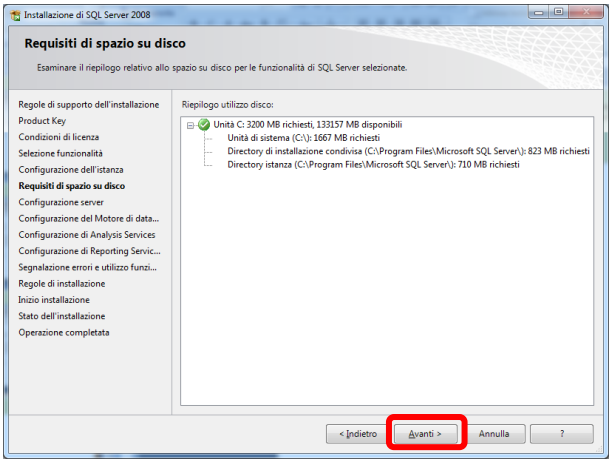
In questa installazione di prova si suggerisce di installare tutti gli strumenti disponibili (pulsante Seleziona Tutto) ma in realtà è possibile anche selezionare solo le opzioni interessanti di cui si è certi del loro successivo utilizzo e proseguire comunque nel processo di installazione; in seguito sarà comunque possibile installare successivamente le altre opzioni momentaneamente tralasciate.

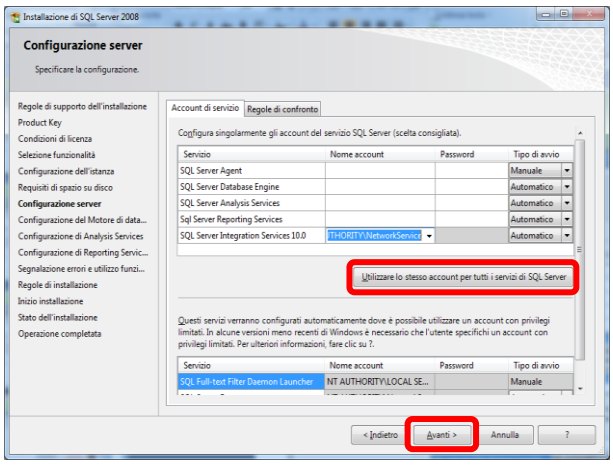
Una cosa interessante da evidenziare è che è possibile installare più volte SQL Server sulla stessa macchina e ciò consente di disporre di diversi server per database su uno stesso server fisico. Ogni installazione è denominata istanza e consente di amministrare database diversi e completamente separati.




Nell'esempio sopra riprodotto, l'istanza che si sta installando verrà denominata mioSQLserver, ed il sistema segnala che è già presente un'istanza di SQL server Express (la versione leggera gratuita).

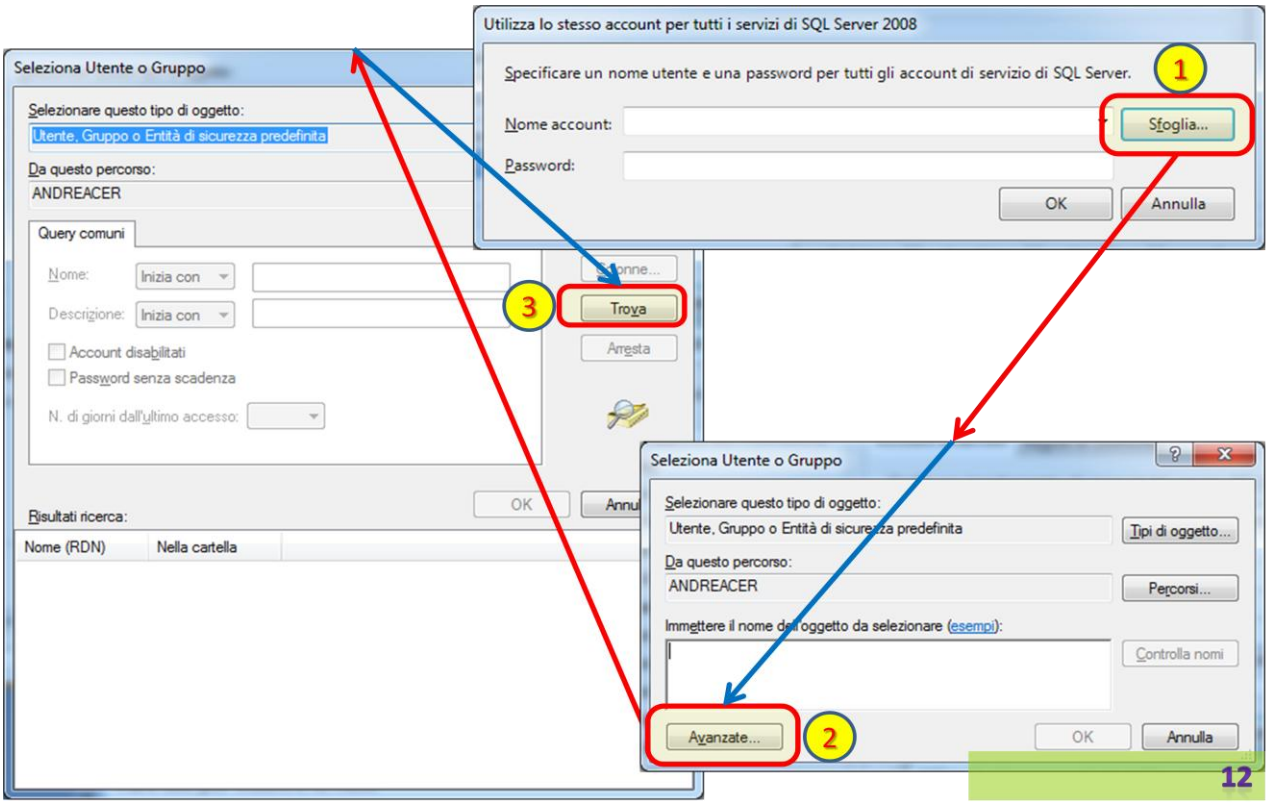
Ogni istanza deve avere un nome univoco. Spesso ad esempio nelle grandi organizzazioni si decide di creare un'istanza di SQL Server per lo sviluppo delle applicazioni concrete e un'altra per le prove.

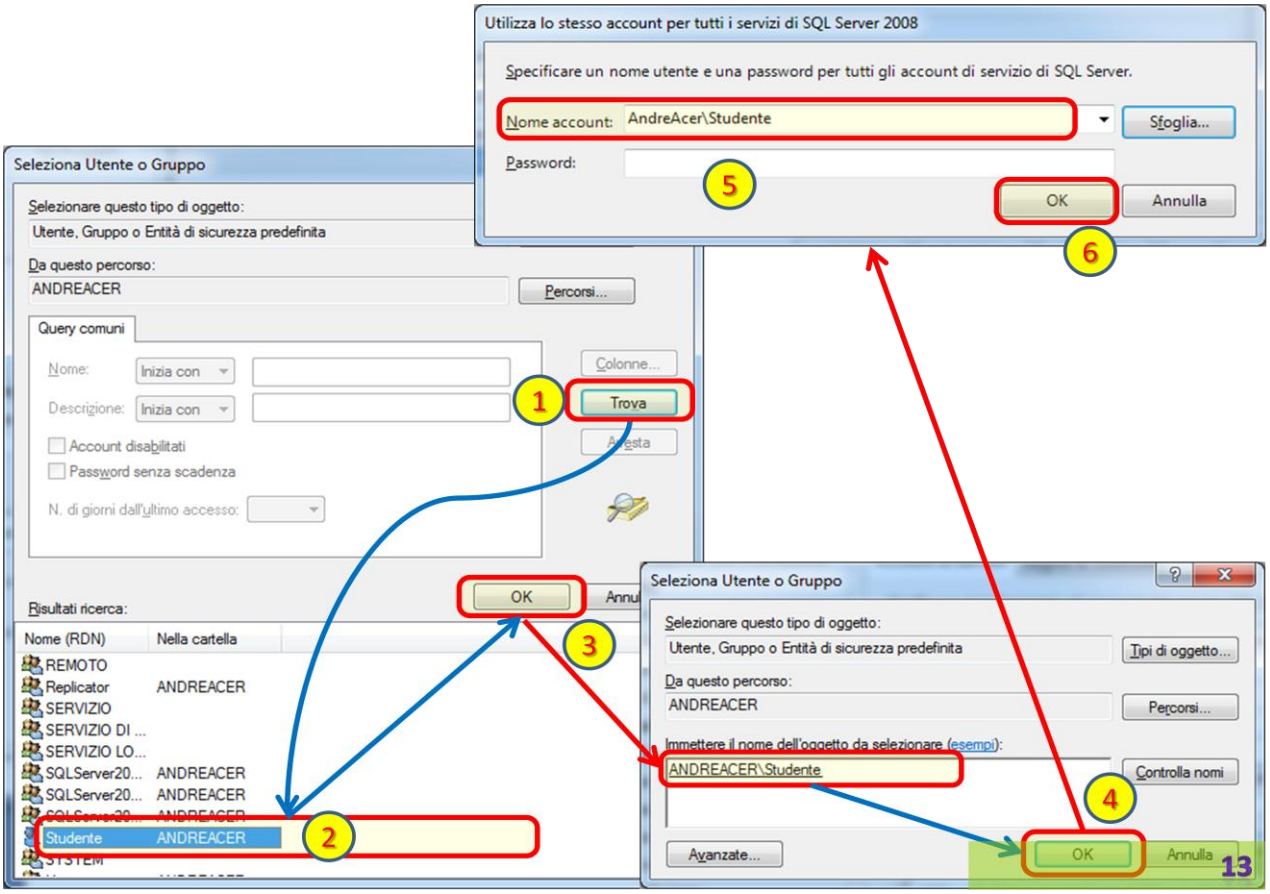



L'ultima finestra illustrata consente di scegliere un utente per i vari servizi. Nel nostro caso, per semplicità, si sceglie l'opzione di Utilizzare lo stesso account per tutti i servizi e di seguito scegliere Sfoglia per selezionare utenti già presenti nel sistema.

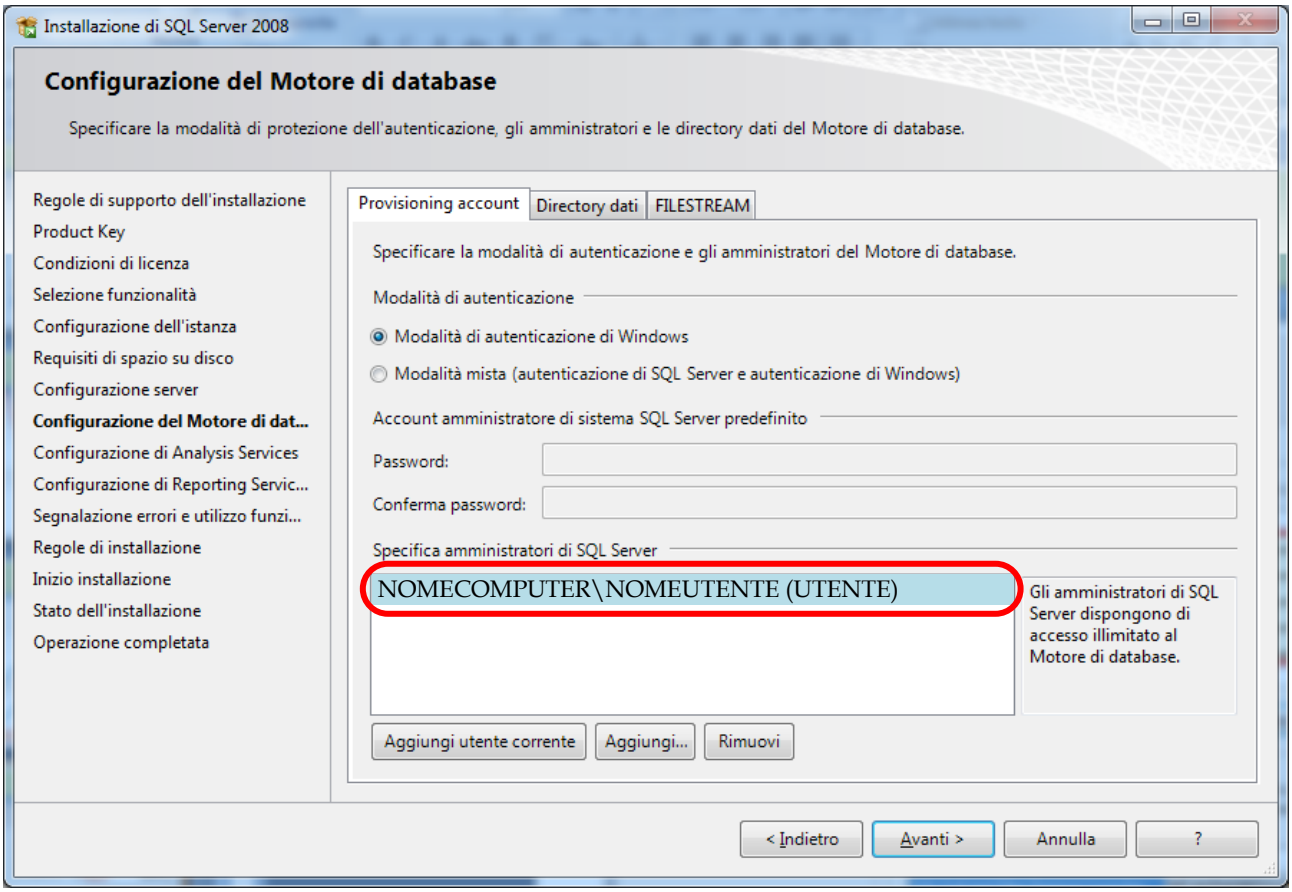
Con il pulsante Avanzate possiamo scegliere di visualizzare gli oggetti utente di Windows (sistema Widows7 nella figura) e con Trova otteniamo tutti gli utenti del sistema.



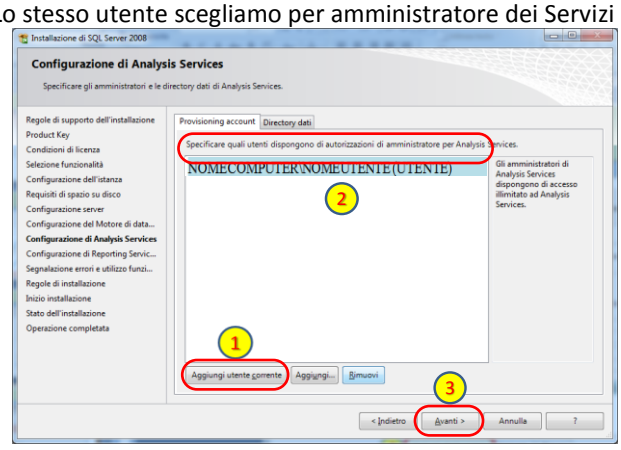



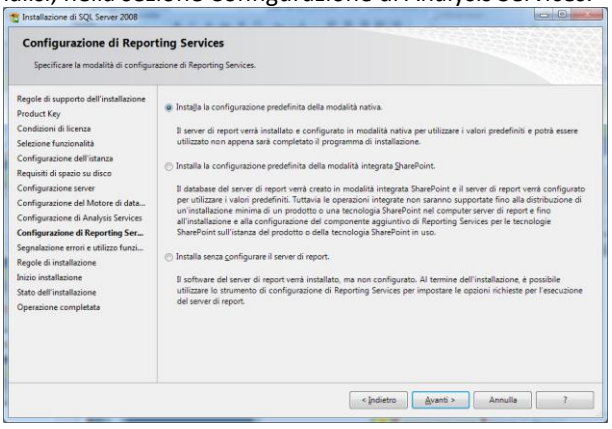

Nella installazione proposta scegliamo l'utente Studente presente sulla macchina denominata ANDREACER.

Proseguendo nell'installazione verrà proposta una finestra della sezione Configurazione del Motore di database e dobbiamo scegliere la modalità di autenticazione.




È possibile scegliere tra due modalità di autenticazione: Modalità di autenticazione di Windows e Modalità mista. La prima determina l'utilizzo dell'autenticazione di Windows per la gestione dei login a SQL Server (ed è quella che utilizzeremo nel presente documento); la seconda permette di utilizzare sia l'autenticazione di Windows sia quella di SQL Server. Sempre nella stessa schermata è necessario definire un account di amministrazione, da utilizzare in caso di emergenze. Nell'esempio è proposto un nome fittizio `NOMECOMPUTER\NOMEUTENTE (UTENTE)`. Lo stesso utente scegliamo per amministratore dei Servizi di Analisi, nella sezione Configurazione di Analysis Services.



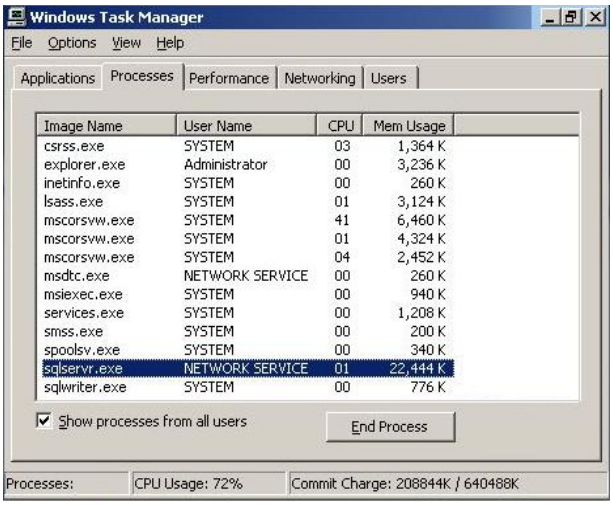

Infine saranno presentate una finestra per comunicare (opzionale) a Microsoft eventuali errori e opzioni di installazione e la validazione delle regole di installazione e in conclusione la finestra di installazione (cliccare su Installa).

# IMPLEMENTAZIONE DI UN DATABASE

# 6 Management Studio per SQL Server 2008

## 6-1 Il processo sqlservr.exe

Se l'installazione di SQL Server 2008 è andata a buon fine possiamo iniziare ad esplorare le funzionalità del prodotto e in particolare gli strumenti visuali che lo rendono facile da utilizzare e molto efficace. Il servizio principale di SQL Server consiste in un processo per Windows separato. Per visualizzarlo è sufficiente aprire il Task Manager di Windows e scorrere i processi attivi per individuare, tra gli altri, quello denominato **sqlservr.exe**. Tale processo è eseguito e monitorato da Windows come un servizio e ad esso vengono allocate le opportune risorse tra cui la necessaria quantità di memoria RAM e tempo di elaborazione CPU. Per questo le risorse allocate possono variare in funzione del carico di lavoro a cui il server SQL deve far fronte.

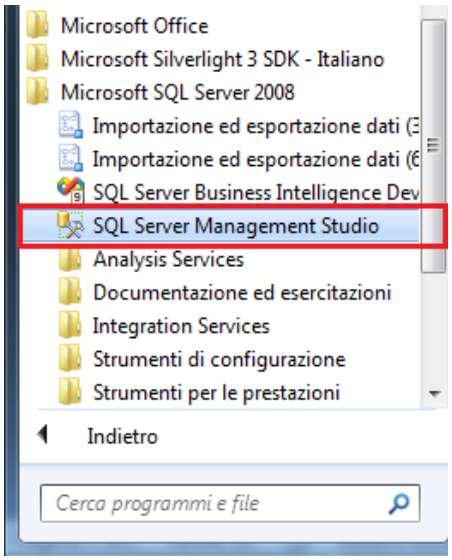



## 6-2 Avvio di SQL Server Management Studio

Siccome SQL Server viene eseguito come un servizio esso non ha interfacce collegate per l'interazione con gli utenti. Eventuali client possono comunque connettersi mediante gli appositi protocolli e comunicare con esso. Per agevolare l'amministrazione tuttavia il sistema dispone di uno strumento separato che permette di richiedere l'esecuzione di comandi e funzioni come da un utente privilegiato connesso al servizio di SQL Server; sarà poi **sqlservr.exe** ad eseguire le richieste sul database cui si è connessi. Tale strumento prende il nome di SQL Server Management Studio (SSMS).

**SQL Server Management Studio (SSMS)** si presenta come un'interfaccia visuale che fornisce molti strumenti intuitivi e utili che agevolano l'uso di SQL Server. Questo ambiente di gestione si presenta come un'interfaccia utente classica, ricca di funzionalità potenti ma semplici da utilizzare, che consente di creare, amministrare, sviluppare e gestire i database.

È possibile avviare SSMS dal menù Start, selezionando la cartella di installazione di SQL Server 2008 e cliccando su SQL Server Management Studio.

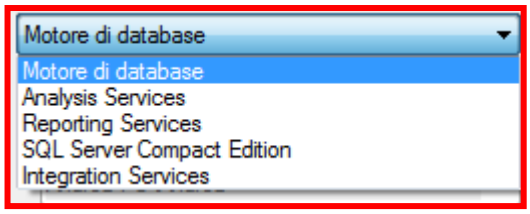



All'apertura del Management Studio (MS) compare una finestra di connessione al server simile alla seguente:


La finestra di Connessione al Server verifica l'autenticazione nella modalità decisa in fase di installazione. In tale finestra occorre specificare:

1. il tipo di server (per il momento va bene il valore Motore del Database);

Tra i tipi di server a cui ci si può connettere possiamo elencare:




- Motore di database (il server di gestione dei dati);
- Analysis Services (servizi analitici);
- Reporting Services (servizi di rendicontazione);
- SQL Server CE;
- Integration Services.

2. il nome del server (nella casella a discesa è possibile scegliere l'opzione Sfoglia dovrebbe comparire il nome della macchina su cui avete installato SQL Server, selezionate questa oppure uno degli altri server eventualmente disponibili);
3. il tipo di autenticazione, che dipende dall'opzione scelta in fase di installazione (in questa dispensa si era scelto di installare l'opzione Autenticazione di Windows, ma si potrebbe selezionare anche il tipo Autenticazione di SQL Server inserendo le opportune informazioni di login).

Col tasto Opzioni, è possibile impostare altre proprietà della connessione, tra cui per esempio la scelta del protocollo di rete e se criptare la connessione.



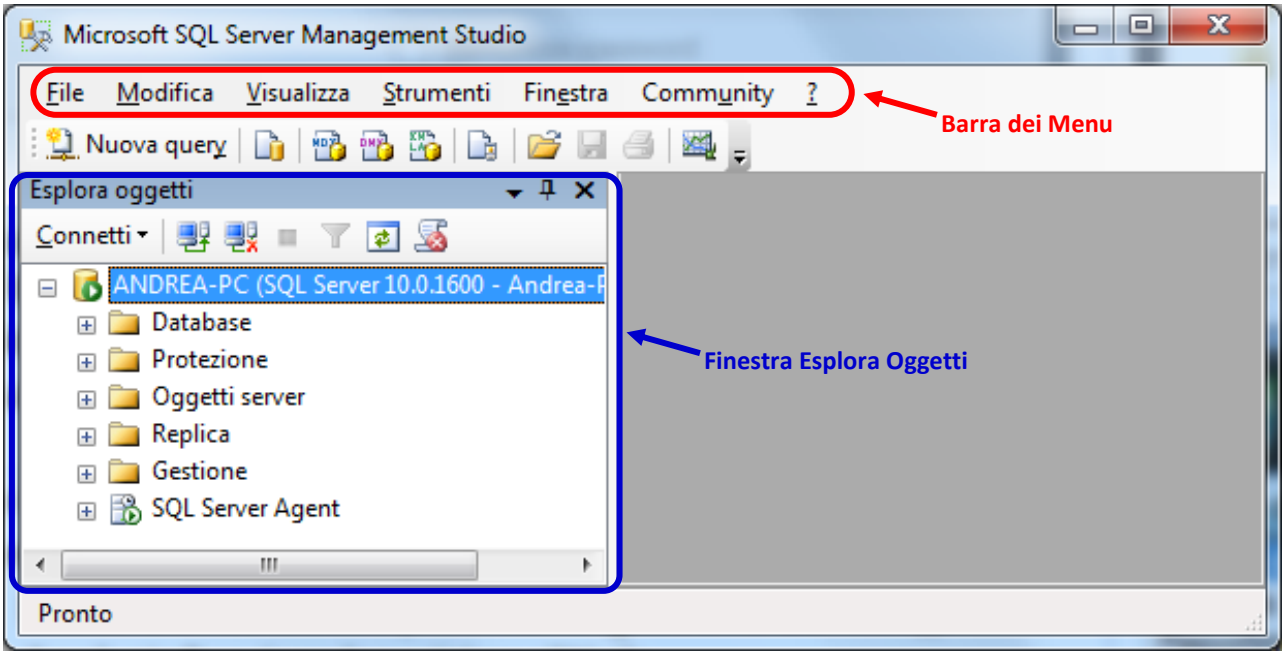
Dopo avere compiuto le opportune scelte è possibile cliccare sul pulsante Connetti per accedere alla finestra principale di SSMS.

L'interfaccia utente di SSMS è piuttosto consueta e risulta semplice da usare.

In questo esempio ci si è connessi all'istanza di SQL Server attiva sul computer locale, ma ogni studente potrebbe trovare alcune differenze poiché si connetterà alla propria.

## 6-3 Panoramica sulle sezioni di SSMS

Dopo essersi autenticati sul sistema compare la finestra principale di MSSMS che propone diverse finestre e menu.

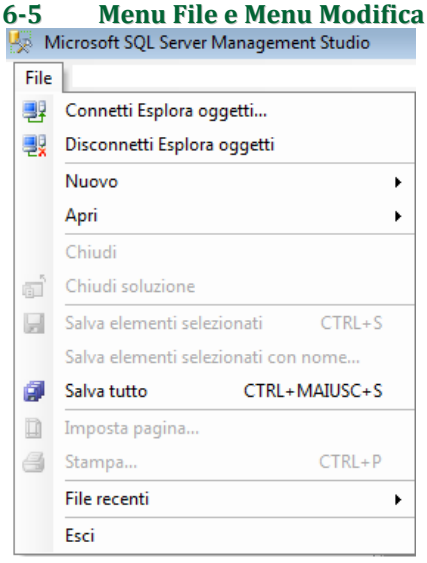



## 6-4 Barra dei menu

Nella barra dei menù di MSSMS sono presenti le seguenti voci:

- File, per gestire le operazioni principali di connessione ai database e ai server;
- Modifica, per le ordinarie operazioni di taglia, copia, incolla e analoghe
- Visualizza, che consente di mostrare le diverse aree di lavoro
- Strumenti, per
- Finestra, con cui
- Community,
- ?

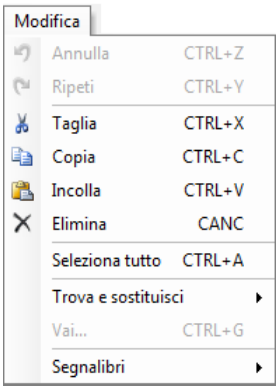
## 6-5 Menu File e Menu Modifica




Il menu **File** consente di gestire le aree relative alla connessione.

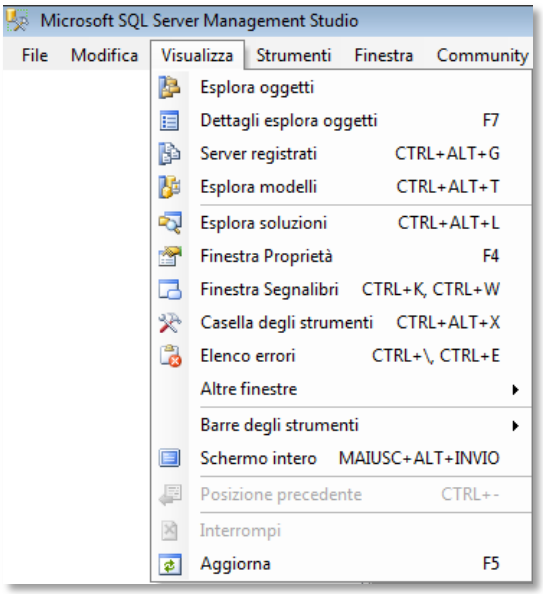
- Connetti Esplora oggetti permette di connettersi a un diverso server;
- Disconnetti Esplora oggetti effettua l'azione contraria
- Nuovo è relativo alla creazione di Script e di Query
- Apri si rivolge alla manipolazione di Script e di Query

Molte delle voci del menu File saranno comprese in seguito dopo aver analizzato le operazioni sul database.




Il menu **Modifica** comprende le consuete operazioni di modifica del testo, annullamento e ripristino di operazioni, ricerche e sostituzioni nel testo, utilizzo di segnalibri per facilitare la gestione del DBA.

### 6-6 Menu Visualizza



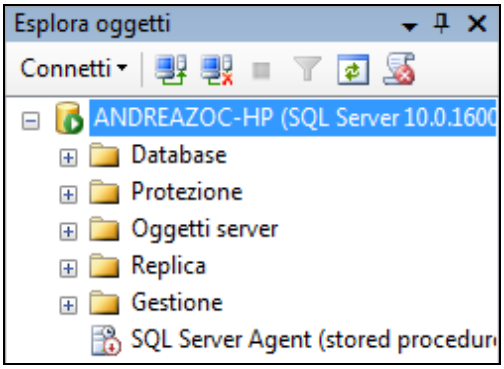

Il Menu Visualizza è utile per rivelare componenti di gestione di SSMS. Nel caso che un'area sia nascosta o chiusa è possibile renderla visibile da questo menu. Molte delle opzioni del menu mostrano finestra specifiche di lavoro:

- **Esplora Oggetti** è l'albero degli elementi del database
- **Dettagli Esplora Oggetti** è relativo al nodo selezionato
- **Esplora Modelli** è attinente a esempi di codice TSQL
- **Finestra Proprietà** mostra le proprietà di un oggetto
- **Finestra Segnalibri** per gestire i segnalibri
- **Casella degli Strumenti** per gestire le barre di sviluppo
- **Elenco Errori** che mostra gli errori del codice presente

La voce Server Registrati mostra tutti i server SQL Server che vengono registrati nel SSMS. Probabilmente si vedrà l'unico server attualmente registrato, ma è possibile registrare un ulteriore server.

### 6-7 Finestra Esplora Oggetti (Object Explorer)

Sul lato sinistro di MSSMS compare (altrimenti è possibile utilizzare il menu Visualizza per farla apparire) la finestra Esplora Oggetti che mostra un albero la cui radice coincide con l'istanza del server installata e a cui sono agganciati diversi nodi. I nodi che compaiono nell'albero sono:




- **Database** ♦ Elenca i database sia di sistema che di utente ai quale si è connessi
- **Protezione** ♦ Elenca la lista dei login degli utenti che possono connettersi a SQL Server
- **Oggetti Server** ♦ Contiene oggetti come le periferiche di backup e fornisce la lista dei linked server (server remoti connessi al nostro)
- **Replica** ♦ Mostra i dettagli relativi alla copia di dati dal database sul server ad un altro database sullo stesso o su un altro server
- **Gestione** ♦ Consente di gestire i piani di manutenzione (maintenance plans), le policy, i log e i messaggi di errore
- **SQL Server Agent** ♦ Permette di pianificare ed eseguire attività specifiche in determinati momenti, riportando i dettagli sia in caso di successo che di fallimento delle stesse

Per il momento è sufficiente questa brevissima panoramica degli oggetti che si analizzeranno in seguito dopo aver costruito un primo database di prova.

# 7 Elementi di un database e database di sistema

Questo documento non è rivolto alla progettazione di un database che richiede l'impiego di molteplici strumenti di analisi, di rappresentazione e modellazione, di scelte operative e tecniche ciascuna delle quali può costituire un testo specifico.

Molti dei concetti analizzati nel presente documento daranno per scontato il significato e l'uso degli elementi del database. Tuttavia in questo documento si farà riferimento a elementi fondamentali per la costruzione del database. Tra questi SSMS provvede alla gestione di:

- Tabelle, i costituenti fondamentali del database, contenitori dei dati soggetti ai vincoli;
- Colonne (campi o domini di una tabella) e righe (o record o tuple, i mattoni informativi)
- Procedure Memorizzate e Funzioni, routine scritte in TSQL che espletano compiti circoscritti sui dati
- Indici, archivi che velocizzano le ricerche sulle tabelle e su colonne e righe
- Viste, interrogazioni memorizzate, dotate di un nome e di diritti di accesso e utilizzo

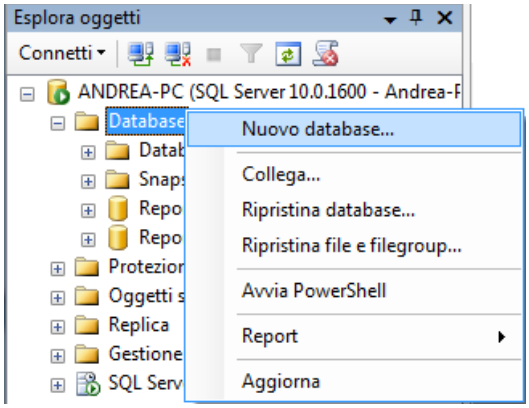
# 8 Implementazione di un database

Per poter studiare le funzionalità di SSMS è necessario creare un database di prova. Per costruire un nuovo database è possibile utilizzare due modi: mediante l'interfaccia grafica o scrivendo manualmente del codice T-SQL.

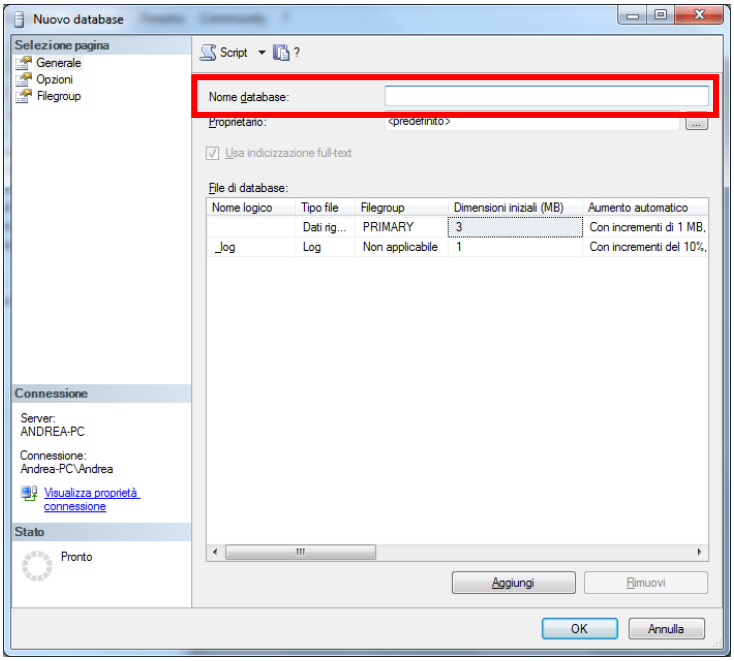
In questo documento si analizzerà principalmente il procedimento visuale, lasciando l'analisi del linguaggio TSQL a un ulteriore approfondimento.

L'uso di istruzioni T-SQL è comunque una scelta obbligata per coloro che abbiano intenzione di amministrare database anche per eventuali backup e strumenti di front-end.

In modalità visuale si selezioni il nodo Database dell'Esplora Oggetti e si proceda con un clic destro (vedi figura a lato).




Scegliendo Nuovo Database compare una finestra simile a quella qui a lato:




Nella parte superiore della finestra è richiesto un nome per il database che stiamo costruendo. Per il momento ipotizziamo che il nuovo database si chiami Scuola.

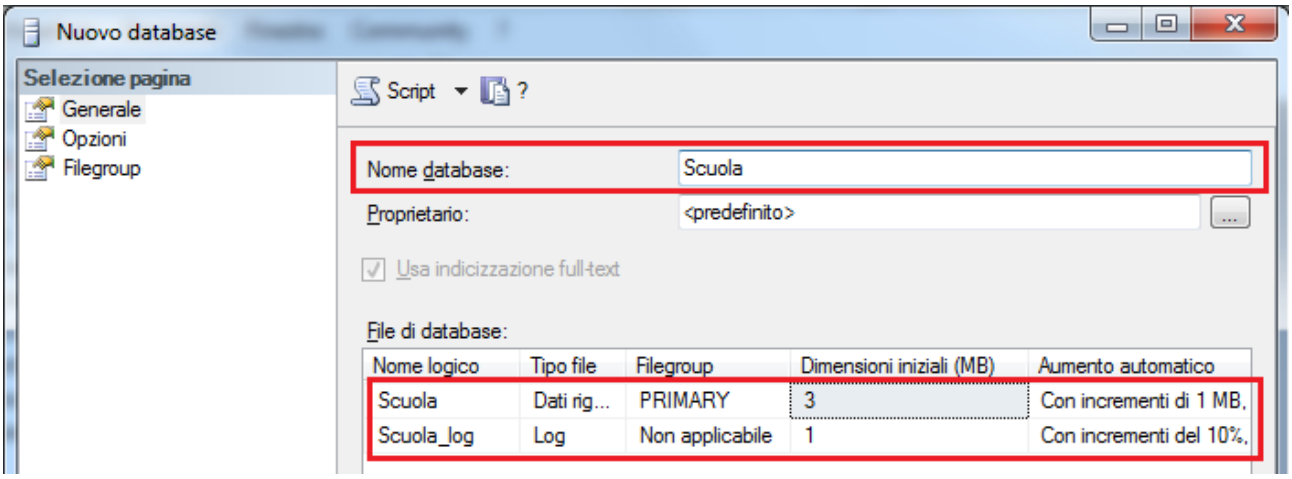
Si deve digitare il nome del database (in questo esempio Scuola) e si può osservare che i nomi dei file relativi al database si adattano automaticamente.

Sotto il nome del database è possibile scegliere un valore per il Proprietario che deve essere scelto con un utente con l'autorità di creazione di database.

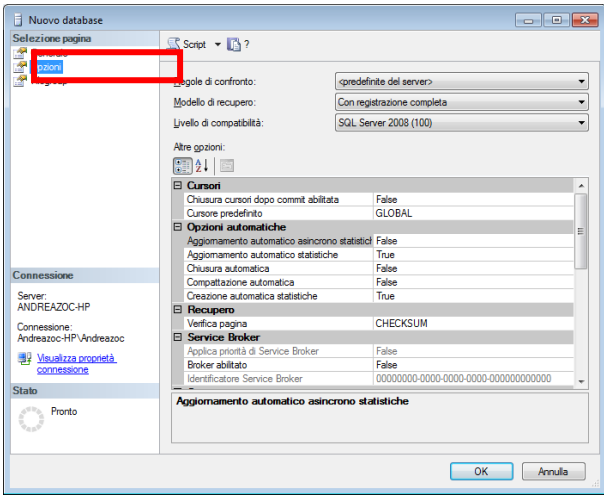
Il file di database sarà memorizzato su disco rigido con l'estensione .MDF (ad esempio Scuola.mdf) che significa Master Data File (file di dati principale) obbligatorio per ogni database.

Il database può essere corredato con un Data File Secondario che ha estensione .NDF, file utile per suddividere tabelle e indici di un database su locazioni diverse (per esempio dischi rigidi distinti), ma con un peggioramento delle prestazioni. Per creare un file NDF occorre cliccare sul tasto Aggiungi (in basso) nella finestra.

La finestra mostra anche la dimensione iniziale del file MDF, l'aumento automatico delle dimensioni e il percorso in cui sarà salvato il file.




È possibile visualizzare le opzioni del database da creare, cliccando Opzioni sul lato sinistro della finestra.




Con le opzioni è possibile definire le Regole di Confronto, ovvero il set di caratteri vigente e le direttive di ricerca e di ordinamento come il confronto tra caratteri maiuscoli e minuscoli.

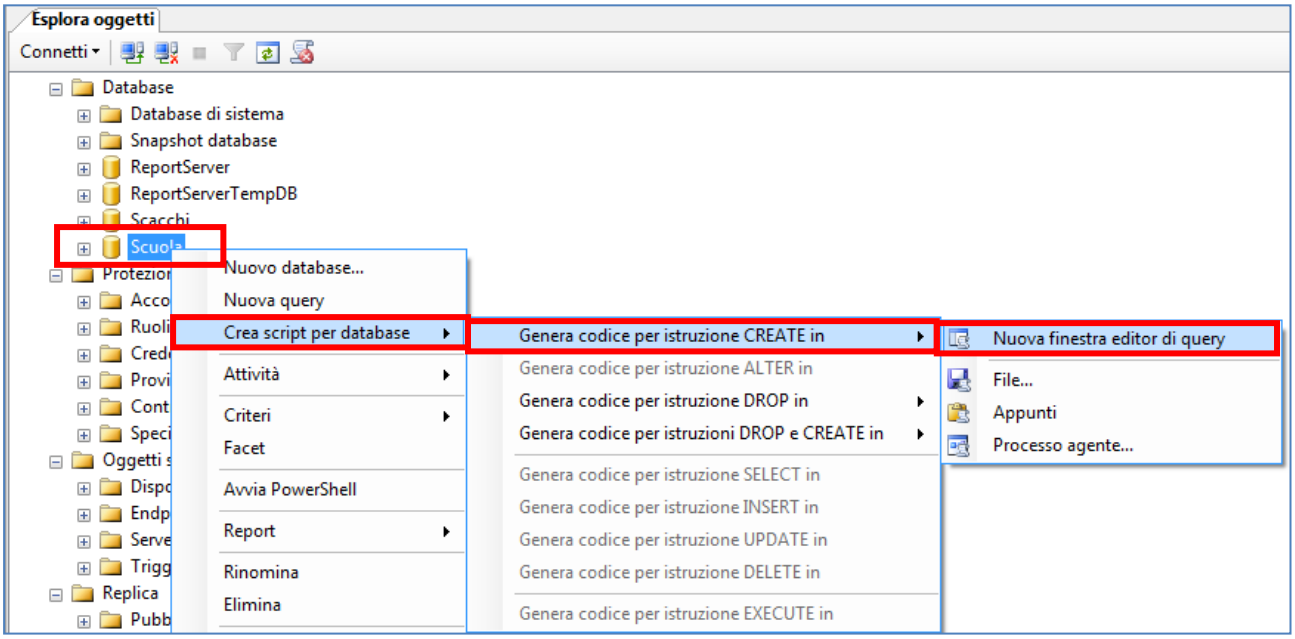
Possiamo anche stabilire quale sia il Modello di Recupero scegliendo tra tre possibili opzioni:

- con Registrazione Completa (**opzione di default**)
- con Registrazione Minima delle operazioni Bulk
- con Registrazione Minima

Infine è possibile impostare il Livello di Compatibilità verso le precedenti versioni di SQL Server, sebbene rinunciando a alcune funzionalità e prestazioni.

Conviene lasciare le opzioni di default proposte da SQL server.

Dopo la conferma della creazione, nell'Esplora Oggetti compare un nuovo nodo, col nome del database creato (Scuola) che è possibile selezionare e per il quale è possibile visualizzare le caratteristiche col tasto destro. Per esempio è possibile visualizzare il codice T-SQL che il server ha generato ed eseguito scegliendo l'opzione Crea Script per database | Genera Codice per istruzione CREATE | Nuova Finestra Editor di Query (vedi figura sotto):




Non esamineremo in questa sede il codice generato; tuttavia vale la pena di esaminare alcune delle istruzioni presenti:

Transact-SQL

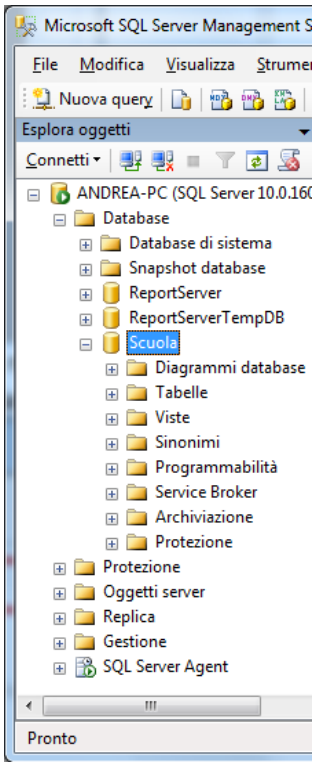
```
CREATE DATABASE [Scuola] ON  PRIMARY
( NAME = N'Scuola', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\Scuola.mdf' , SIZE = 3072KB , MAXSIZE =
UNLIMITED, FILEGROWTH = 1024KB )
```

Transact-SQL

```
LOG ON
( NAME = N'Scuola_log', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\Scuola_log.ldf' , SIZE = 1024KB , MAXSIZE
= 2048GB , FILEGROWTH = 10%)
GO
```

Il comando CREATE DATABASE è l'istruzione SQL per creare un nuovo database e prevede gli argomenti NAME (nome del database) e FILENAME (percorso di salvataggio); la clausola LOG ON del comando definisce le caratteristiche del file LDF, ovvero il file di LOG (registro delle transazioni) collegato al file principale.

### 8-1 Elementi del database

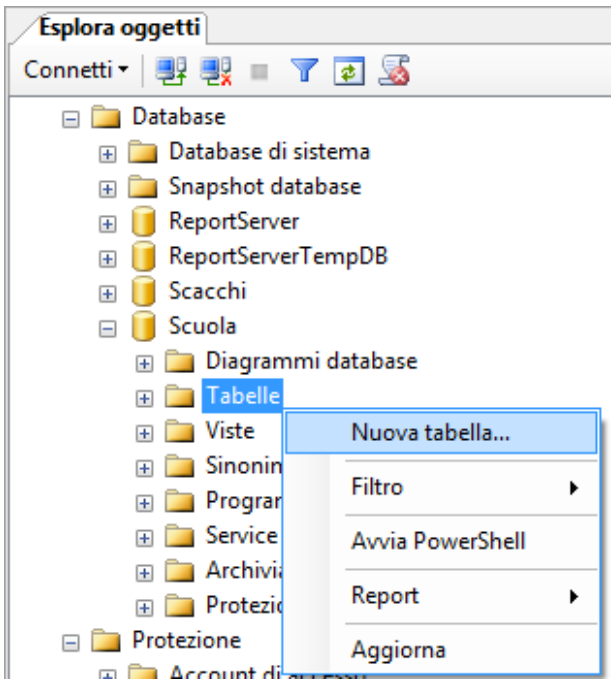



Chiudiamo la finestra e esaminiamo il nodo del database (Scuola) che può essere "espanso" e mostrare tutti i suoi elementi:

- **Diagrammi database**
  *Per impostare e gestire le associazioni tra tabelle in modo visuale*
- **Tabelle**
  *Per creare e modificare le tabelle del database e i vincoli rispettivi*
- **Viste**
  *Per creare e modificare le viste dei dati*
- **Sinonimi**
  *Per creare e modificare oggetti utili per fornire un nome alternativo a un altro oggetto di database o per realizzare un livello di astrazione per proteggere un'applicazione client dalle modifiche apportate al nome o alla posizione dell'oggetto di base*
- **Programmabilità**
  *La sezione riservata a procedure, funzioni, trigger, assembly, tipi utente, regole (asserzioni ormai deprecate), costanti, guide di piano*
- **Service Broker**
  *È una tecnologia (parte del Motore di database) introdotta in Microsoft SQL Server 2005 che consente agli sviluppatori di database di creare applicazioni protette, affidabili e scalabili. Service Broker offre servizi di accodamento e messaggistica affidabili ed è utilizzato per le applicazioni che usino sia un'unica istanza di SQL Server sia che distribuiscano il carico di lavoro tra più istanze*
- **Archiviazione**
  *La gestione del Full-Text, la archiviazione con FILESTREAM*
- **Protezione**
  *La gestione di utenti e ruoli, la crittografia e le chiavi, i certificati e i controlli.*

Come primo passaggio è opportuno soffermarsi sull'elemento Tabelle. La tabella è l'elemento centrale per un database e costituisce il mattone fondamentale su cui costruire un buon sistema di dati. La tabella è essenzialmente un contenitore di dati in cui archiviare registrazioni di ciascun elemento. Solo in seguito vedremo i diagrammi (e le chiavi esterne), le viste, i sinonimi e i servizi.

# 9 Implementazione di una tabella




Analogamente alla creazione del database, anche la creazione di una tabella può essere eseguita mediante l'interfaccia visuale o con un comando T-SQL.

In visuale si deve selezionare il nodo Tabelle, fare clic destro col mouse e scegliere la voce Nuova tabella del menu contestuale.

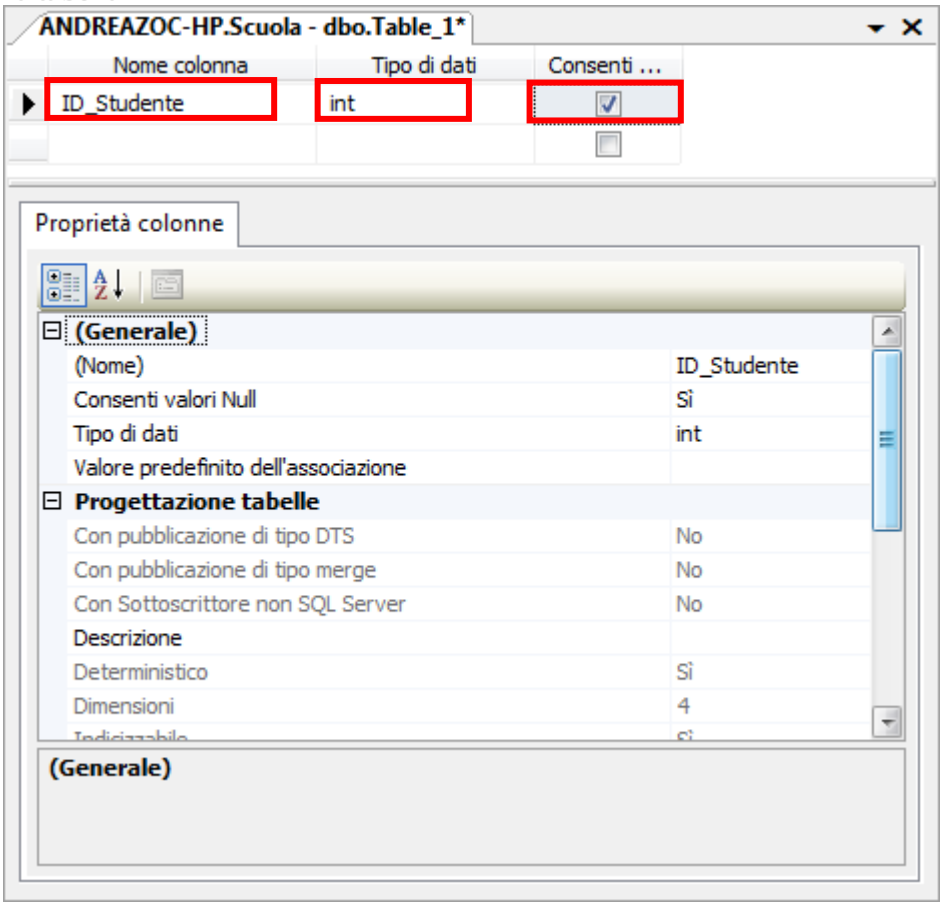
La definizione della tabella è effettuata mediante la finestra Designer di Tabella.

La tabella è un oggetto di database descritto da un certo numero di colonne ciascuna delle quali è caratterizzata da un nome, da un tipo e da diversi vincoli.

Come suggerimento operativo è consigliabile utilizzare il tipo intero vincolato come IDENTITY auto incrementante per la chiave primaria della tabella.

## 9-1 Chiave primaria di una tabella

Il nome della colonna è il primo valore richiesto e ciascuna colonna deve avere un nome univoco (non sono ammesse colonne omonime nella stessa tabella).

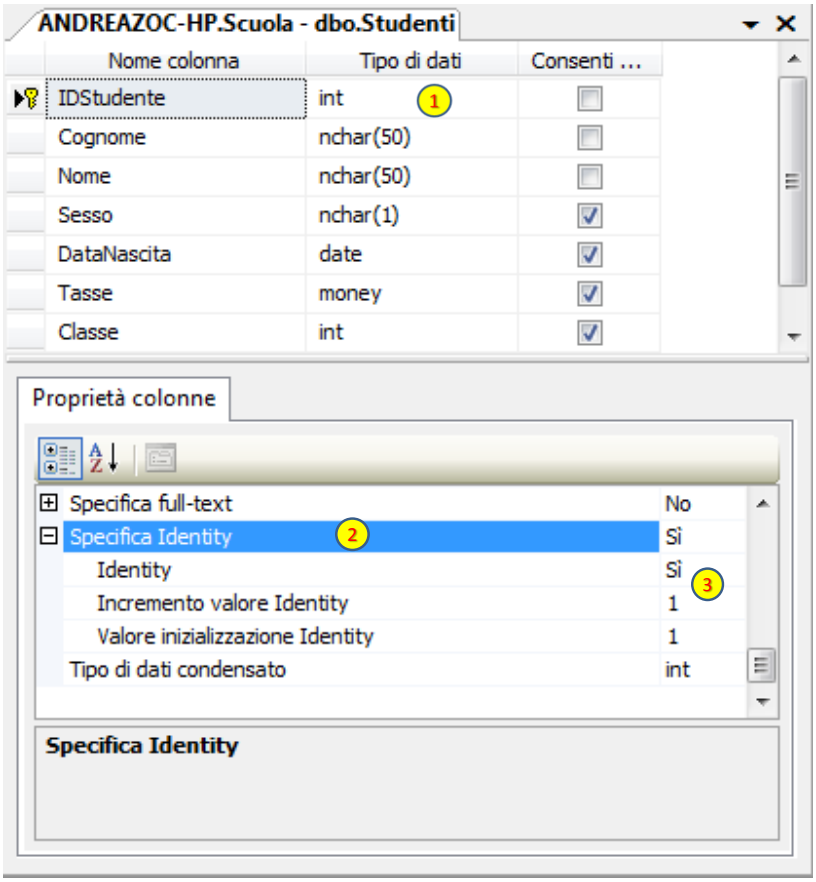



Subito dopo è richiesto il tipo di dato della colonna; la discussione dei molti tipi di dato disponibili in SQL server 2008 è presentata nella apposita appendice, in fondo a questo documento.

A fianco del tipo di dati è indicato il vincolo Consenti valore nullo, che decide se ammettere valori NULL oppure non ammetterli (nel primo caso il campo è detto facoltativo, altrimenti è detto obbligatorio).

La sezione inferiore del Designer propone le proprietà della singola colonna in via di definizione.

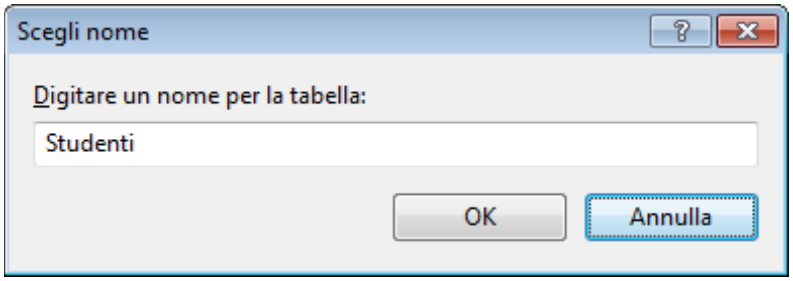
Si suggerisce di impostare la prima colonna come chiave primaria della tabella, semplicemente facendo clic sul pulsante con la chiave che compare in alto sulla sinistra. Il campo sarà contraddistinto dalla chiave dorata.




Oltre a specificare che la colonna è una chiave primaria, poiché il campo è intero si può impostare la proprietà Specifica Identity. IDENTITY è di fatto un vincolo che consente di generare automaticamente (auto incremento) un numero progressivo per quella colonna ogni volta che si inserirà una nuova riga nella tabella.

Impostare a SI il valore della proprietà Identity e lasciare i valori (1,1) di default proposti come valore iniziale e passo del contatore.

Clicchiamo sul pulsante di salvataggio e si aprirà una finestrella in cui inserire il nome della tabella.




Il nome della tabella (molto importante) sarà spesso esplicativo del significato dei dati archiviati in essa e sarà univoco nel database (non sono ammesse due tabelle omonime nello stesso database).

Per seguire l'esempio qui proposto si suggerisce di creare due tabelle: Studenti e Classi.

Ecco di seguito le tabelle da costruire:

**Studenti**

| Nome colonna | Tipo di dati | Consenti … |
|---|---|---|
| IDStudente | int | ☐ |
| Cognome | nchar(50) | ☐ |
| Nome | nchar(50) | ☐ |
| Sesso | nchar(1) | ☑ |
| DataNascita | date | ☑ |
| Tasse | money | ☑ |
| Classe | int | ☑ |

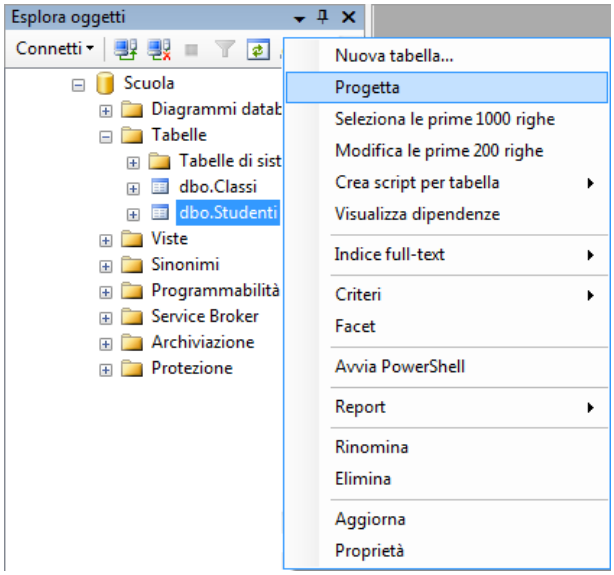
**Classi**

| Nome colonna | Tipo di dati | Consenti … |
|---|---|---|
| IDClasse | int | ☐ |
| Anno | nchar(1) | ☐ |
| Sezione | nchar(1) | ☐ |
| Indirizzo | nchar(3) | ☐ |

Le chiavi primarie sono tutte INT IDENTITY (1,1).

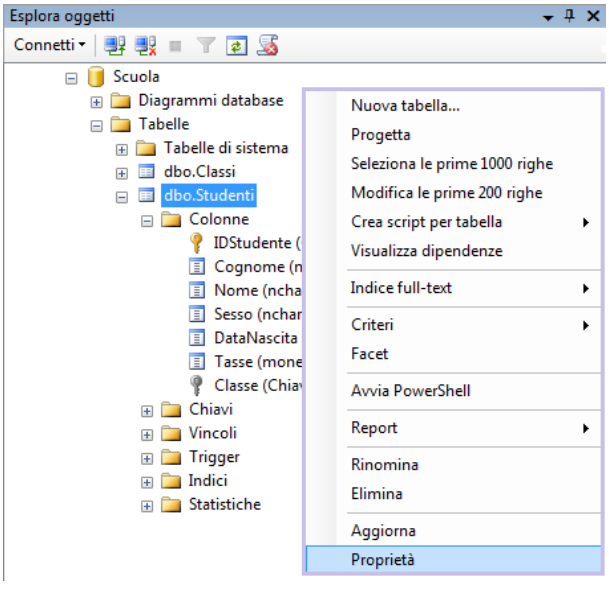
## 9-2 Modifica di una tabella

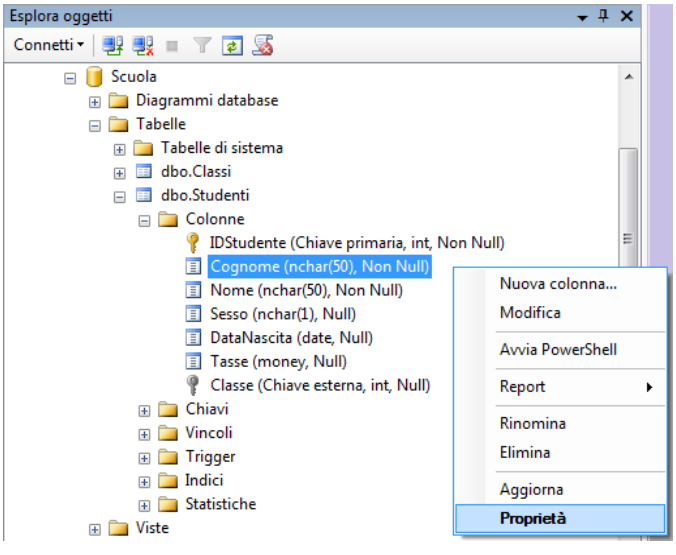
Dopo aver costruito una tabella è sempre possibile modificarla accedendo al Designer di Tabella, facendo clic sul nodo della Tabella e, col clic destro, scegliendo la voce Progetta del menu contestuale (vedi figura a lato).



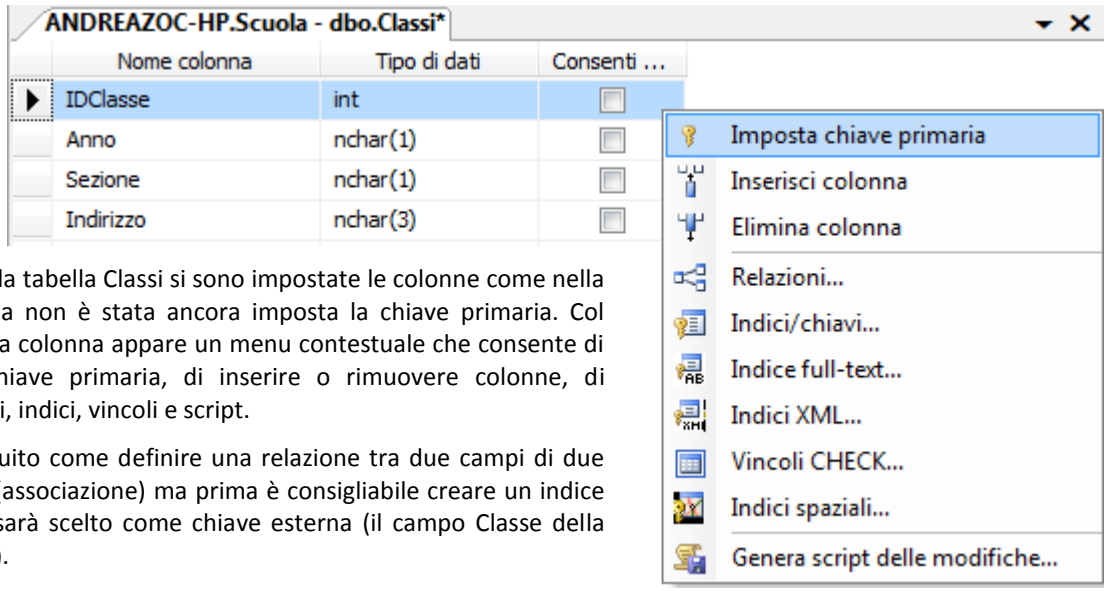

È anche possibile visualizzare le Proprietà sia della tabella che della singola colonna, sempre dal menu contestuale che appare col clic destro (figure sotto).

Il menu offre anche varie opzioni per creare nuove tabelle, nuove colonne, per visualizzare elementi interessanti e per aggiornare la visualizzazione (utile specie in caso di comandi testuali o di multiutenza).



Nella scheda Progetta Tabella è anche possibile modificare proprietà della colonna e per esempio impostare una chiave primaria ancora non definita.




Per esempio nella tabella Classi si sono impostate le colonne come nella figura a lato, ma non è stata ancora imposta la chiave primaria. Col tasto destro sulla colonna appare un menu contestuale che consente di Impostare la chiave primaria, di inserire o rimuovere colonne, di definire relazioni, indici, vincoli e script.

Vedremo in seguito come definire una relazione tra due campi di due tabelle distinte (associazione) ma prima è consigliabile creare un indice sul campo che sarà scelto come chiave esterna (il campo Classe della tabella Studenti).

# 10 Definizione di indici

## 10-1 Panoramica sugli Indici

Dopo aver creato le tabelle database è opportuno definire gli indici ad esse associati. Gli indici sono particolari archivi associati a una tabella che migliorano le prestazioni per dati che sono soggetti a frequenti ricerche, vincoli e controlli. SQL server crea automaticamente l'indice associato alla chiave primaria di una tabella, poiché essa ha vincoli di univocità (occorre fare frequenti ricerche per evitare duplicati) ed è utilizzata nella definizione di correlazioni (associazioni tra chiavi di diverse tabelle).

Anche i campi su cui si compiono frequenti ricerche per query o inserimenti (es. il nominativo di una persona) sono candidati ideali per definire degli indici. Gli indici sono particolarmente efficaci in database con un ampio numero di dati e con accessi frequenti per operazioni sui dati.

La definizione di un indice in SQL Server serve a innalzare le prestazioni. Quando si deve compiere una perlustrazione su una tabella dotata di un indice, la ricerca è effettuata solo sui dati memorizzati nell'indice, invece che sui dati della tabella, e si l'operazione diviene molto più rapida.

Un indice può essere definito su una singola colonna (indice semplice) o su un gruppo di colonne aggregate (indice composto) della stessa tabella.

Le scelte sulla definizione degli indici dipendono dall'analisi delle tabelle del database e dagli utilizzi presunti dei dati. In SQL Server, un indice composto può associare fino a sedici colonne ma in genere un DBA non dovrebbe superare le cinque; la quantità complessiva dei valori delle colonne aggregate in un indice di una riga non può superare i 900 byte.

Se i valori dell'indice sono soggetti a modifiche frequenti, le operazioni di aggiornamento possono subire dei peggioramenti, poiché oltre a modificare i dati della tabella si aggiornano anche i dati e gli ordinamenti dell'indice.

Se si definisce un campo che sarà usato come chiave esterna conviene definire un indice su di essa.

## 10-2 Indice clustered

Un indice clustered definisce l'ordine fisico dei dati di una tabella. Se abbiamo più di una colonna definita in un indice clustered i dati vengono ordinati in modo sequenziale in base a tali colonne. Può essere definito solo un indice clustered per tabella perché ovviamente sarebbe impossibile memorizzare i dati in più ordini contemporaneamente.

Quando vengono inseriti dati SQL Server inserisce i riferimenti agli stessi all'interno dell'indice e inserisce la riga nella posizione appropriata. Una buona norma è quella di non definire un indice su colonne che vengono aggiornate frequentemente poiché in questo caso SQL Server si troverà a dover continuamente modificare l'ordine fisico dei dati con un grosso dispendio di capacità di elaborazione della macchina.

## 10-3 Indice nonclustered

Un indice **nonclustered** non memorizza direttamente una parte dei dati della tabella ma memorizza riferimenti ai dati che corrispondono alle colonne su cui è definito. Gli indici **nonclustered** sono memorizzati in strutture separate rispetto alla tabella per cui è costruito.

Quando si compie una ricerca sui dati memorizzati per quali è previsto un indice nonclustered, la ricerca di SQL Server è basata sull'informazione memorizzata nell'indice e sfrutta i puntatori dell'indice per individuare le righe che corrispondono ai criteri della ricerca.

## 10-4 Indice unique

Un indice può essere unico (unique) o non unico (nonunique). Un indice unico assicura che i valori contenuti nelle colonne definite nell'indice siano univoci all'interno della tabella. Un indice non unico invece permette di inserire nella tabella più righe con gli stessi valori per le colonne definite nello stesso. Per tale motivo gli indici unici sono comunemente utilizzati per supportare costanti come le chiavi primarie mentre quelli non unici vengono comunemente utilizzati per supportare la ricerca di dati tramite colonne che non costituiscono chiavi.

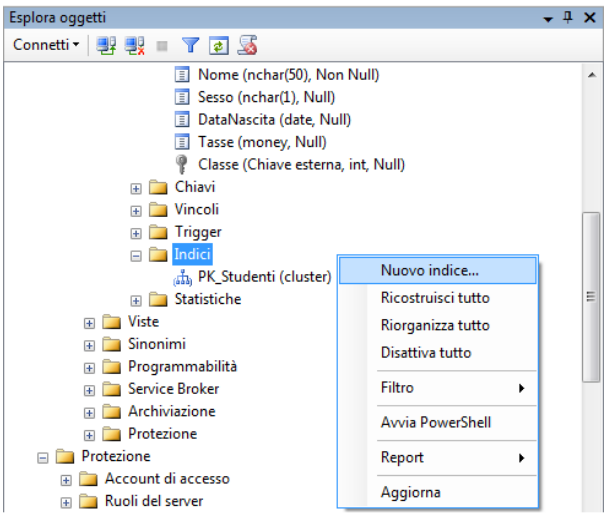
## 10-5 Indice XML primario ⓘ (da fare)

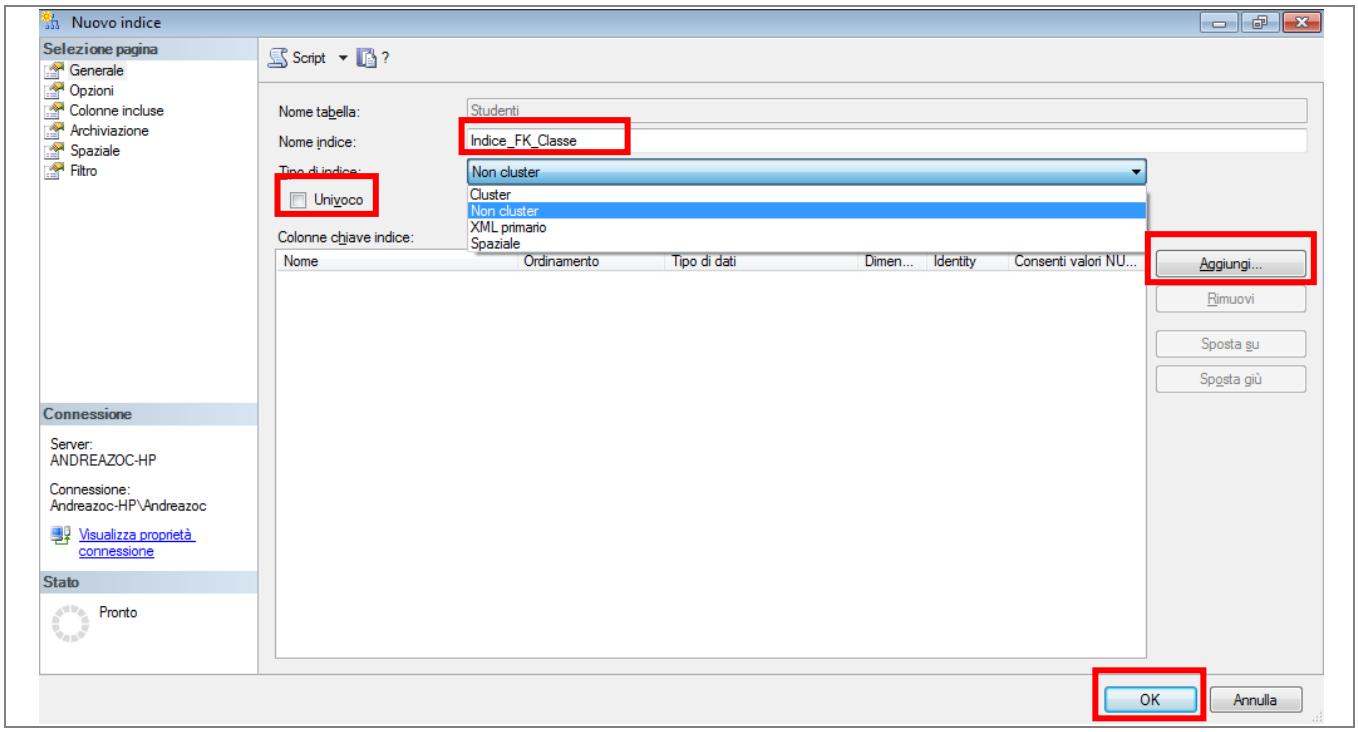
## 10-6 Indice Spaziale ⓘ (da fare)

## 10-7 Creazione di un indice

Per creare un indice si può cercare il nodo Indici relativo alla tabella scelta (nell'esempio Studenti) e la cui espansione mostra quelli esistenti (nel nostro esempio l'indice predefinito PK_Studenti relativo alla chiave primaria.

Col clic destro del mouse appare il consueto menu contestuale da cui scegliamo l'opzione Nuovo Indice. Compare una finestra di dialogo (Nuovo Indice) che richiede di digitare il nome del nuovo indice (nell'esempio si è digitato il nome Indice_FK_Classe) e da cui è possibile scegliere il tipo (cluster, non cluster, XML primario, Spaziale) e se sia univoco (non ammette duplicati).





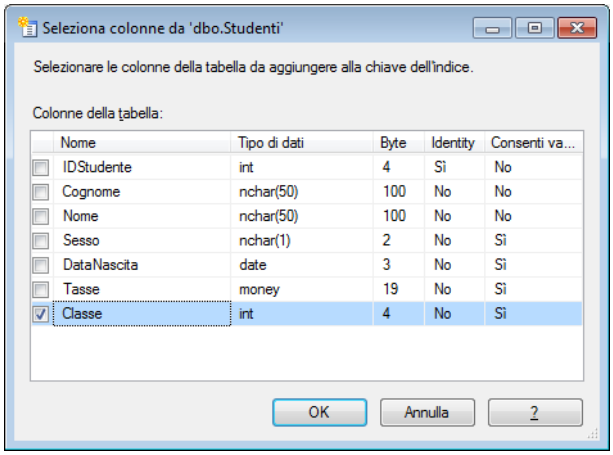

Nel nostro esempio **non** spuntiamo la casella **Univoco** poiché desideriamo che lo stesso valore possa comparire più volte; nella realtà corrisponde al fatto che la stessa classe può essere associata a più studenti (associazione 1:N).

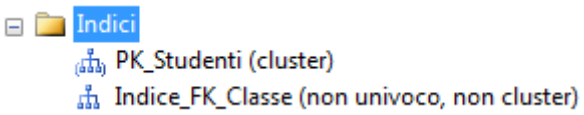
Il pulsante **Aggiungi** consente di scegliere i campi della tabella su cui definire l'indice.

Come si nota è possibile scegliere più di una colonna per costruire un **indice composto**, ma nel nostro esempio selezioniamo la sola chiave esterna Classe (che poi verrà associata alla chiave primaria della tabella Classi).

Infine scegliamo OK per confermare i campi selezionati.

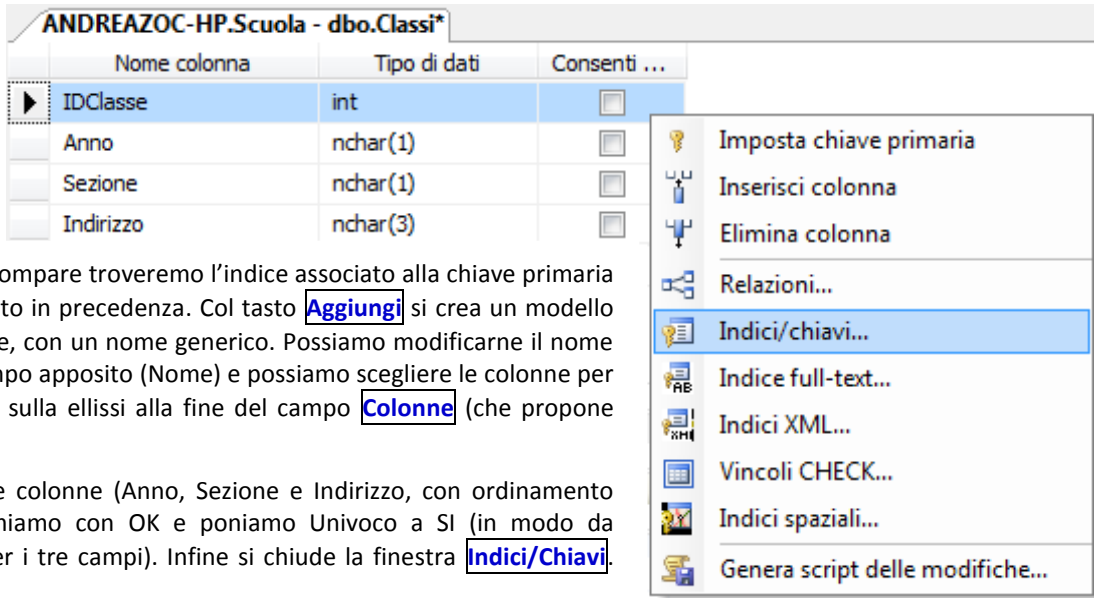
Tornati alla finestra **Nuovo Indice** confermiamo con OK. Infine la finestra si chiude e l'applicazione ritorna alla finestra principale dove possiamo vedere che il nodo Indici è stato arricchito con un nuovo nodo per l'indice appena creato.



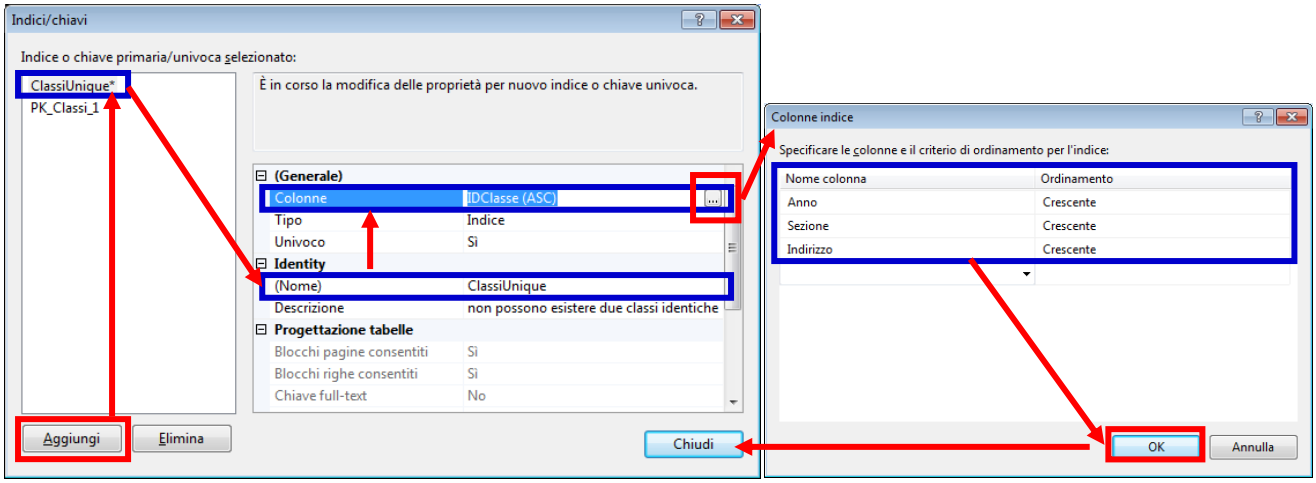
### 10-8 Creazione di un indice univoco su più campi

Supponiamo di volere creare adesso un indice clustered unique sulle tre colonne (Anno, Sezione, Indirizzo) della tabella Classi. Clicchiamo con il tasto destro del mouse sulla tabella e scegliamo l'opzione Progetta per l'apertura della finestra di progettazione. Clicchiamo sul tasto **Indici/Chiavi...** e lavoriamo nuovamente sulla finestra Indexes/Keys.




Nella finestra che compare troveremo l'indice associato alla chiave primaria che abbiamo definito in precedenza. Col tasto **Aggiungi** si crea un modello per un nuovo indice, con un nome generico. Possiamo modificarne il nome digitandolo nel campo apposito (Nome) e possiamo scegliere le colonne per l'indice con un clic sulla ellissi alla fine del campo **Colonne** (che propone IDClasse (ASC).

Selezioniamo le tre colonne (Anno, Sezione e Indirizzo, con ordinamento crescente) confermiamo con OK e poniamo Univoco a SI (in modo da evitare duplicati per i tre campi). Infine si chiude la finestra **Indici/Chiavi**.




# 11 Vincoli di integrità referenziale

In un database relazionale assume particolare importanza il vincolo che lega una chiave esterna ad una chiave primaria. Questo vincolo è spesso impropriamente denominato relazione e SSMS non fa eccezione.

Il significato e le implicazioni di una associazione tra una FK e una PK non è spiegato in questo documento, dove ci si occupa meramente di esaminare le funzionalità operative dell'interfaccia di amministrazione.

Nel nostro esempio dobbiamo associare la chiave esterna Classi (int) della tabella Studenti alla chiave primaria ID_Classe (int identity) della tabella Classi. Nella figura qui di seguito è illustrato il vincolo da costruire sulle tabelle:

**Studenti**

| Nome colonna | Tipo di dati | Consenti ... |
|---|---|---|
| IDStudente | int | ☐ |
| Cognome | nchar(50) | ☐ |
| Nome | nchar(50) | ☐ |
| Sesso | nchar(1) | ☑ |
| DataNascita | date | ☑ |
| Tasse | money | ☑ |
| Classe | int | ☑ |

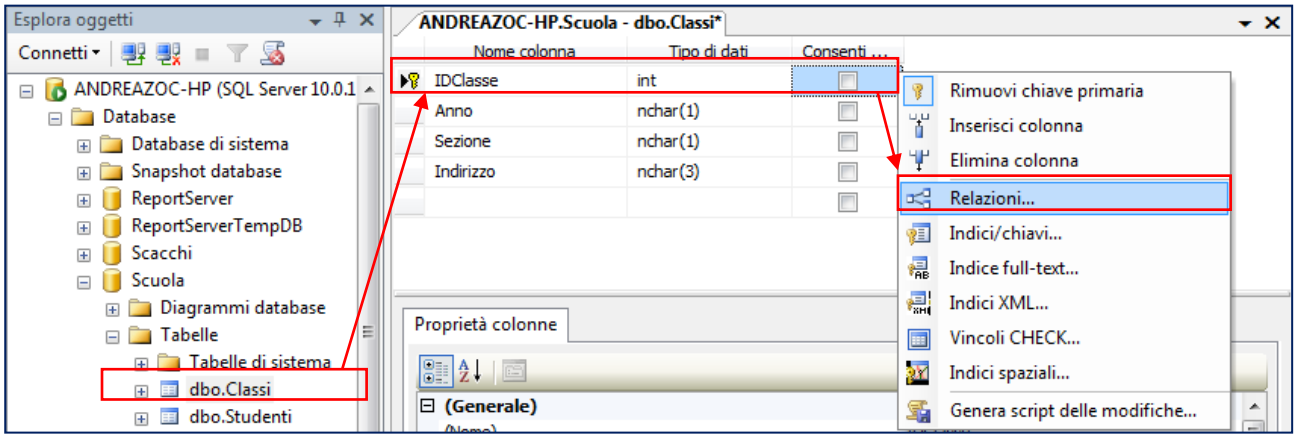
**Classi**

| Nome colonna | Tipo di dati | Consenti ... |
|---|---|---|
| IDClasse | int | ☐ |
| Anno | nchar(1) | ☐ |
| Sezione | nchar(1) | ☐ |
| Indirizzo | nchar(3) | ☐ |

Le chiavi primarie sono tutte INT IDENTITY (1,1).

### 11-1 Creazione di un vincolo di chiave esterna

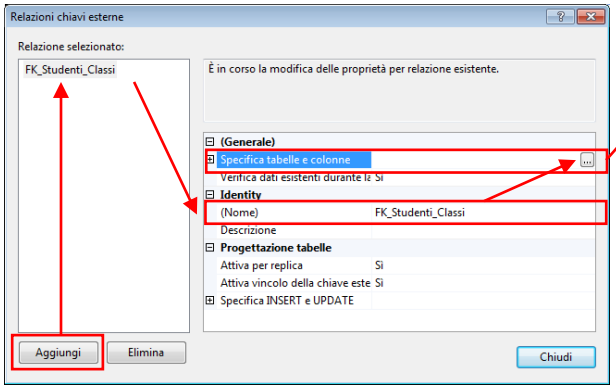
Per creare una associazione occorre aprire in modalità Progettazione la tabella contenente la chiave primaria (nel nostro esempio Classi). Aprire in modalità Progettazione la tabella Studenti e selezionare la colonna Classe (chiave esterna) e fare clic destro col mouse in modo da visualizzare il menu contestuale contenente la voce **Relazioni**.

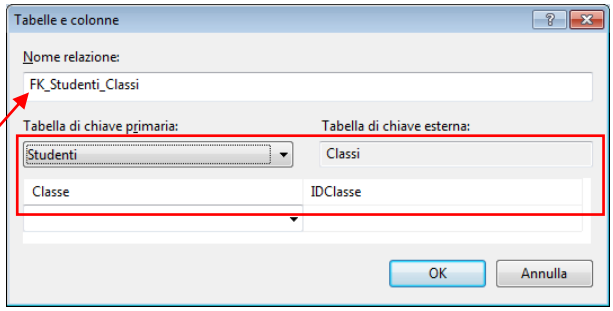



Quando appare la finestra è possibile creare una nuova relazione con il pulsante Aggiungi che genera una associazione con un nome provvisorio e delle proprietà iniziali.

Solitamente è consigliabile modificare il nome (per esempio in FK_Studenti_Classi).

È opportuno quindi modificare le colonne coinvolte nell'associazione cliccando sull'ellisse a fianco della proprietà Specifica tabelle e colonne ... (vedi figura a lato). Occorre modificare la tabella primaria in Studenti e selezionare la




colonna Classe della tabella Studenti e la colonna IDClasse della tabella Classi. Quindi confermare con OK.

# 12 Diagrammi

Lo strumento denominato Diagrammi di SQL server ha un duplice scopo:

- Permette di rappresentare graficamente lo schema del database logico, evidenziando le associazioni;
- Consente di costruire, manipolare e rimuovere relazioni, chiavi, indici e altri oggetti del database.

La prima funzionalità dei Diagrammi è quindi di tipo documentativo e illustrativo; in molti casi è necessario raffigurare le tabelle del database e le associazioni che le legano. Un primo motivo scaturisce dalla necessità di avere documentazione per la manutenzione del database, nel caso che un'organizzazione debba modificare le strutture, aggiornare il software o sviluppare front-end per gli utenti. In altri casi la documentazione è il primo passo per la realizzazione di manuali d'uso, per certificazioni e relazioni tecniche.

La documentazione di un database richiede di rappresentare le tabelle e le relative relazioni in uno schema che nei casi più faticosi deve essere tracciato manualmente con software di terze parti; se il database è di grandi dimensioni con numerose associazioni tra le tabelle il compito di disegnare lo schema può essere abbastanza impegnativo.

La seconda funzionalità dei Diagrammi è di agire sul database stesso; alcune delle precedenti funzionalità, come la definizione di indici e di associazioni tra tabelle, possono essere compiute direttamente passando dal Diagramma. Le operazioni compiute mediante il Diagramma si ripercuote automaticamente sul database.

Nella funzionalità dei Diagrammi si possono rappresentare graficamente solo tabelle e non ammette altri oggetti come schemi, viste, procedure o funzioni, utenti o gruppi, diritti e politiche.

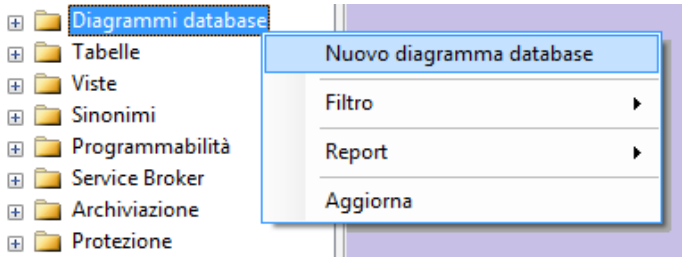
## 12-1 Installazione del supporto

Quando un database è creato non sempre è automaticamente installato il Supporto per Diagrammi. Per richiedere l'installazione è sufficiente aprire il menu contestuale sul nodo Diagrammi col tasto destro del mouse e scegliere Installa Supporto per Diagramma.

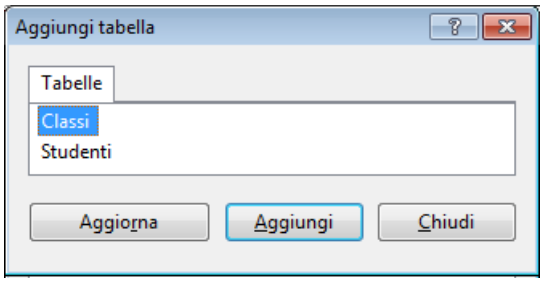
Comunque richiedendo di accedere alla funzionalità, il programma chiede di installare il supporto nel caso che non sia stato già fatto.

## 12-2 Utilizzo di Diagrammi

È possibile creare più di un diagramma per un database.
Per creare un nuovo diagramma per il database di esempio è sufficiente aprire il menu contestuale sul nodo Diagrammi col tasto destro del mouse e scegliere Nuovo Diagramma Database (vedi figura a lato).



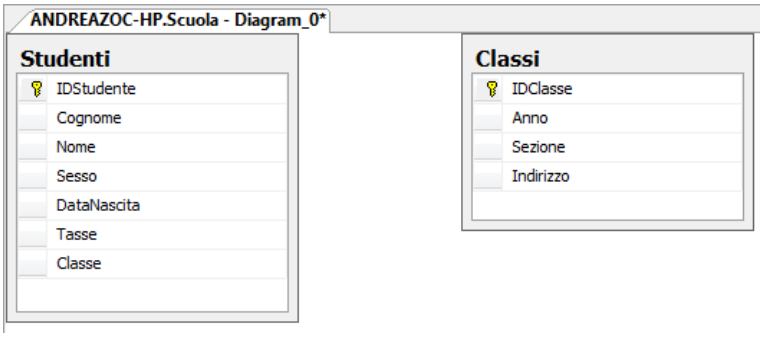

L'applicazione richiede di scegliere le tabelle da visualizzare nel diagramma. Ciascuna tabella può essere inserita solo una volta nello schema e l'inserimento rimuove il nome dall'elenco.

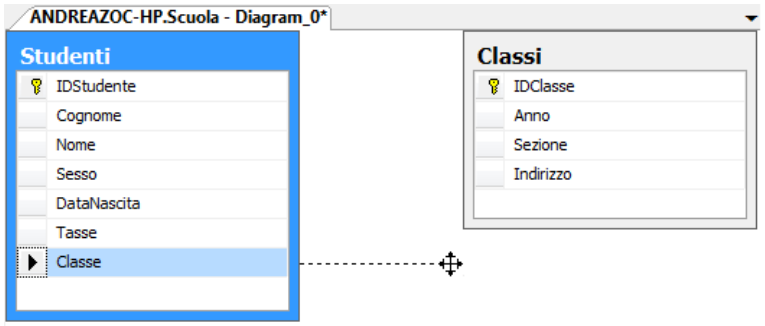



Una volta inserite le tabelle desiderate (nel nostro esempio le uniche due esistenti) si può chiudere la finestra di dialogo ed esaminare lo schema proposto.

Il diagramma proposto dal sistema come base di lavoro illustra le tabelle inserite ed elenca tutti i campi di entrambe.

Sul lato sinistro di ciascun campo appaiono eventuali vincoli come la chiave dorata che raffigura la chiave primaria della tabella.




Per creare un'associazione tra le tabelle si può selezionare la chiave esterna (Classe della tabella Studenti) e trascinarla (drag and drop) verso la chiave primaria della tabella Classi.

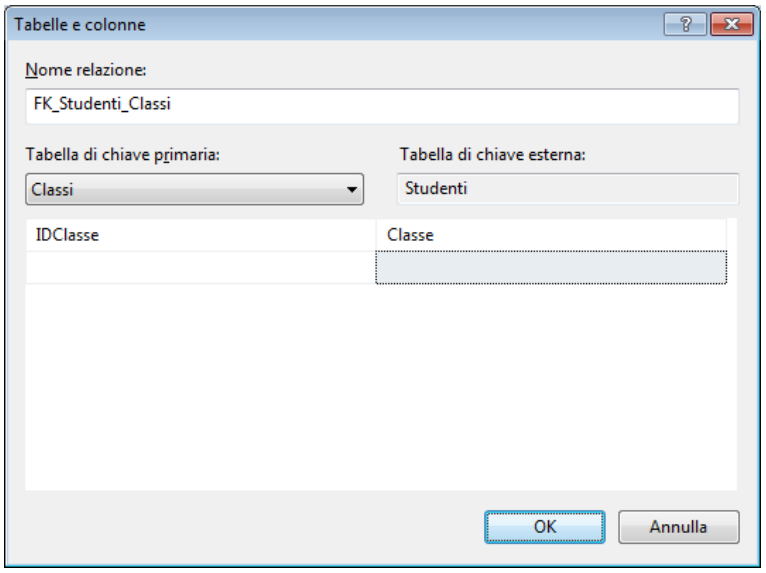
Alla conclusione del trascinamento (si rilascia il tasto del mouse) SSMS propone una finestra di composizione.

La finestra consente di selezionare i campi delle due tabelle e di vincolarli.

La finestra è analoga a quella vista nella discussione sulle associazioni e non richiede ulteriori discussioni.

Nel nostro esempio è possibile creare la associazione (relazione nella terminologia di SQL server) tra Classi e Studenti col pulsante OK.

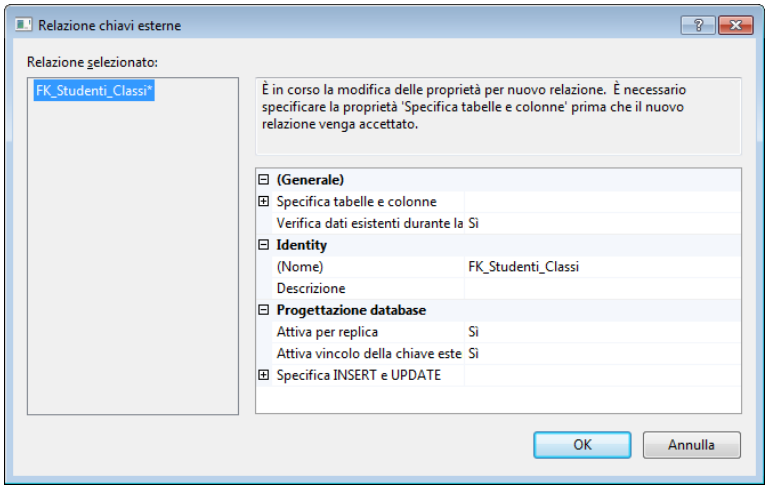
Ovviamente non è opportuno costruire due associazioni identiche tra i campi delle tabelle e se si è già definita la relazione in precedenza conviene annullare la creazione.




Quando si conferma con OK la finestra di dialogo si chiude e il sistema propone la finestra di proprietà della relazione di chiave esterna.

Nella finestra della relazione è possibile modificare il nome (FK_Studenti_Classi) e modificare le altre proprietà come visto in precedenza.

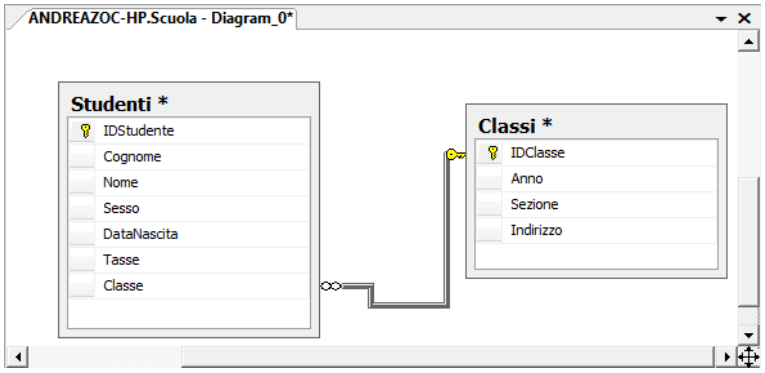
È possibile indicare una descrizione della relazione (es. vincolo di integrità referenziale tra Studenti e Classi) soprattutto se le due tabelle sono legate da molteplici associazioni distinte.




Infine si conclude la creazione della relazione col pulsante OK e si ritorna alla finestra principale di diagramma.

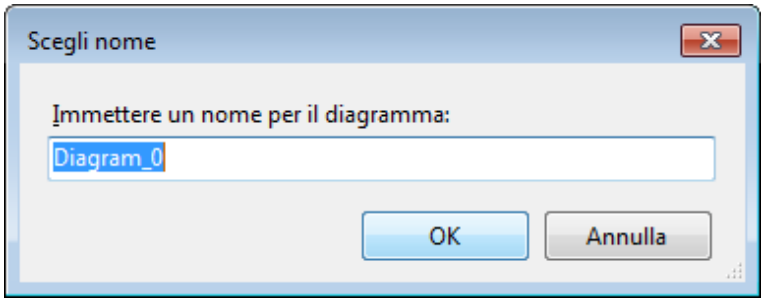
Lo schema del diagramma viene rappresentato graficamente in modo simile alla figura a lato, dove però si è provveduto a spostare (trascinando) le tabelle in posizione diversa dall'originale; poi si è anche variato l'aspetto della congiunzione tra le tabelle, trascinando uno alla volta ciascuno degli estremi del segmento spezzato che raffigura l'associazione.

L'estremità con la chiave è legata alla tabella primaria (dove si aggancia alla PK) mentre l'altro capo presenta una chiusura ad otto.



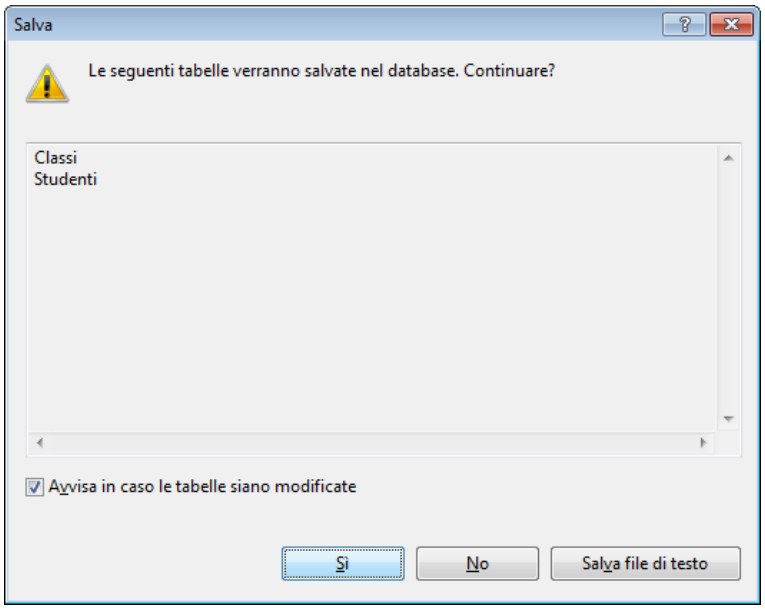

Col pulsante di salvataggio si conferma la creazione del diagramma così modificato e le conseguenze da esso derivanti.

SSMS chiede un nome per salvare l'oggetto diagramma in modo che sia identificato univocamente nel sistema.



Infine SSMS chiede anche una conferma del salvataggio delle modifiche apportate alle tabelle; difatti l'associazione è un vincolo associato ad una tabella (quella con chiave esterna) come si vedrà quando analizzeremo nei dettagli le sintassi SQL.

L'opzione Salva file di testo non corrisponde a una attuazione delle modifiche, ma solo al salvataggio su un file testuale con estensione txt col codice TSQL corrispondente (il sistema chiederà ancora il salvataggio delle modifiche alle tabelle).




## 12-3 Pulsanti della barra dei Diagrammi

Quando si passa alla progettazione di un diagramma compare anche una barra aggiuntiva che presenta diversi pulsanti.


Nuova Tabella, per creare ulteriori tabelle usando lo strumento Diagrammi;

Aggiungi Tabella, per inserire una tabella già esistente nel Diagramma in via di costruzione;

Aggiungi tabelle correlate, permette di inserire nel diagramma tutte le tabelle correlate alla selezionata;

Elimina tabelle da database, consente di distruggere una tabella del database;

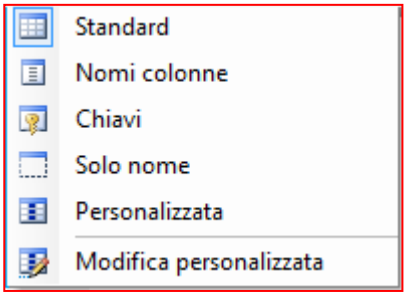
Rimuovi da Diagramma, toglie dal diagramma (ma non dal database) una tabella;

Genera Script delle modifiche, per visualizzare il codice TSQL dell'azione appena invocata;

Imposta chiave primaria, per settare una colonna come PK in una tabella in via di progettazione;

Nuova annotazione di testo,

Vista tabella — Visualizzazione tabella, per modificare gli elementi visualizzati, secondo una delle modalità riportate qui a lato;




Mostra etichette relazioni,

Visualizza interruzioni di pagina, per eventuali stampe

Ricalcola interruzioni di pagina, per eventuali stampe

Ridimensiona automaticamente tabelle selezionate

Disposi selezione e disponi tabelle, per collocarle in automatico

100% — Zoom, per visualizzare il diagramma a video

Relazioni, come la funzionalità già analizzata in precedenza

Gestione Indici e chiavi, come la funzionalità già analizzata in precedenza

Gestione indice full-text, come la funzionalità full-text che vedremo in seguito

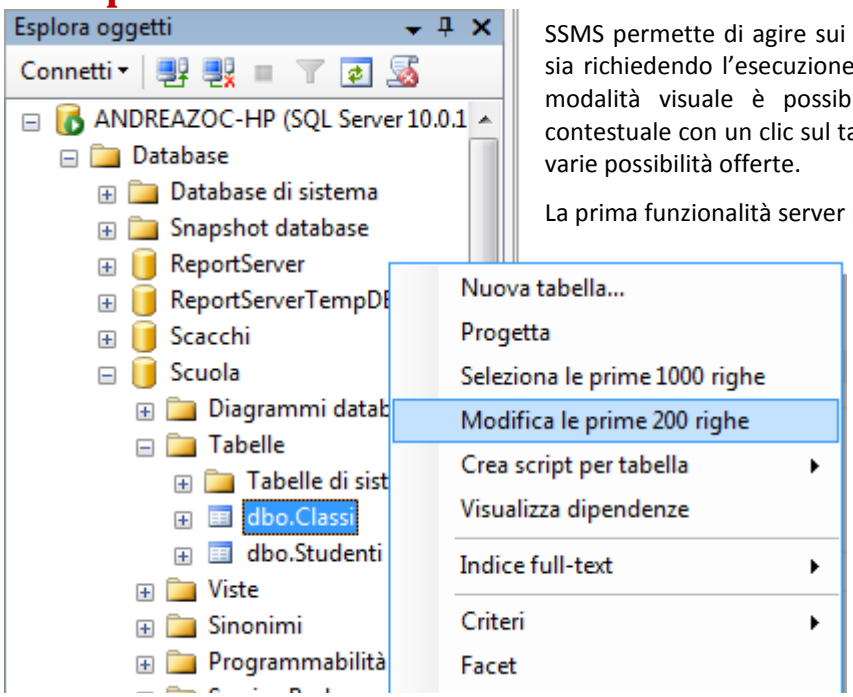
Gestione indice XML, come la funzionalità XML che vedremo in seguito

Gestione vincoli CHECK, che vedremo in seguito

Gestione indici spaziali, che vedremo in seguito

# SQL E DATA MANIPULATION

## 13 Operazioni sui dati in visuale




SSMS permette di agire sui dati sia sfruttando la modalità visuale sia richiedendo l'esecuzione di comandi SQL. Per agire sui dati in modalità visuale è possibile innanzitutto richiamare il menu contestuale con un clic sul tasto destro del mouse e scegliere tra le varie possibilità offerte.

La prima funzionalità server per modificare i dati in forma tabellare

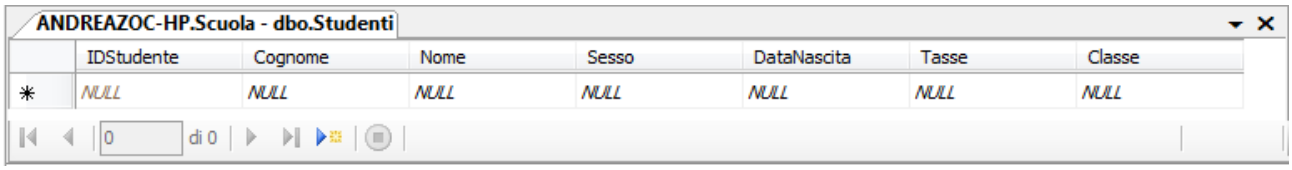
Un momento importante per una base di dati è, ovviamente, quello dell'inserimento dei dati.

Conseguentemente saranno cruciali le operazioni di modifica, cancellazione e visualizzazione degli stessi.

Una prima possibilità è fornita dalla voce **Modifica delle prime 200 righe**.

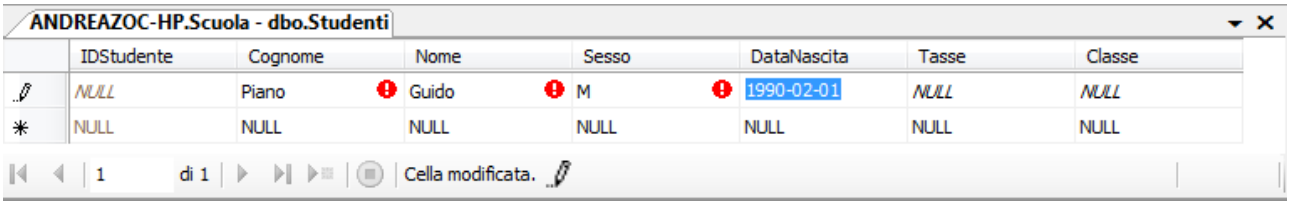
Scegliendo l'azione appare subito il risultato in forma tabellare, come appare nella figura sottostante.

Il risultato della query appare in modalità tabella come nella figura sottostante:

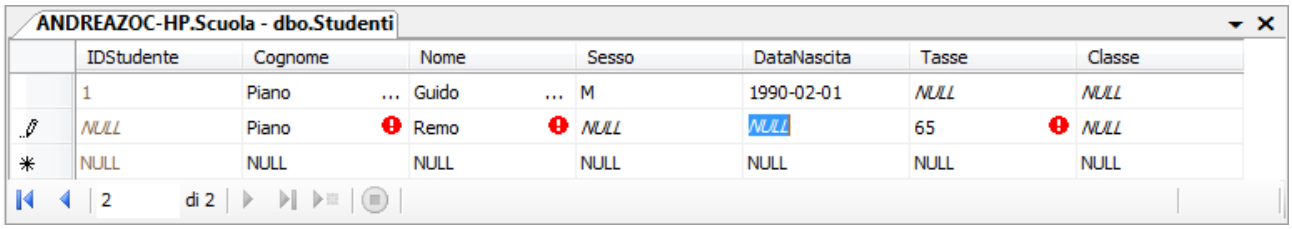



In basso appare la barra di stato con alcune funzionalità. La barra riporta il numero di posizione corrente e il numero di registrazioni totali; nella barra sono disponibili anche i pulsanti per navigare tra i record (primo, precedente, successivo, ultimo) e i pulsanti per inserire un nuovo record.

Per inserire un nuovo record è sufficiente posizionarsi sulla riga con tutti valori nulli e etichettata a sinistra con un asterisco (nuova riga) e digitare i dati da registrare. Le celle col simbolo di elaborazione (❗) indica che il sistema ancora deve valutare la soddisfazione dei vincoli e la ricostruzione degli indici.

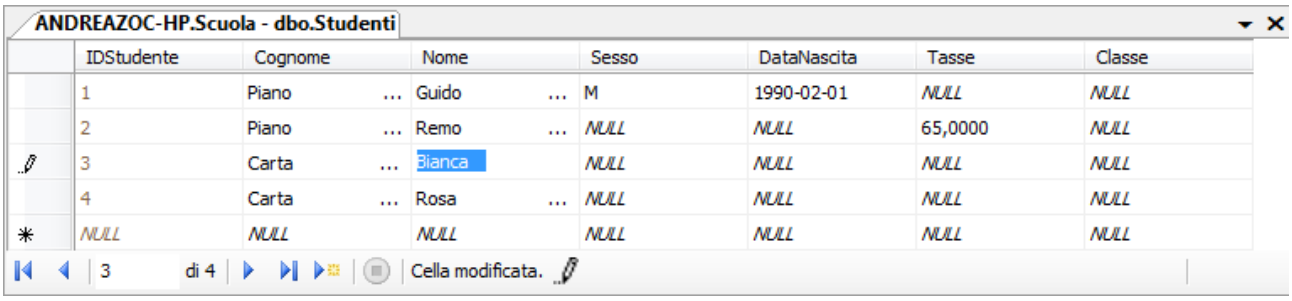



Per confermare la registrazione è sufficiente spostarsi con le frecce nel record sottostante. Il simbolo di elaborazione sparisce dopo aver confermato tutte le validazioni.

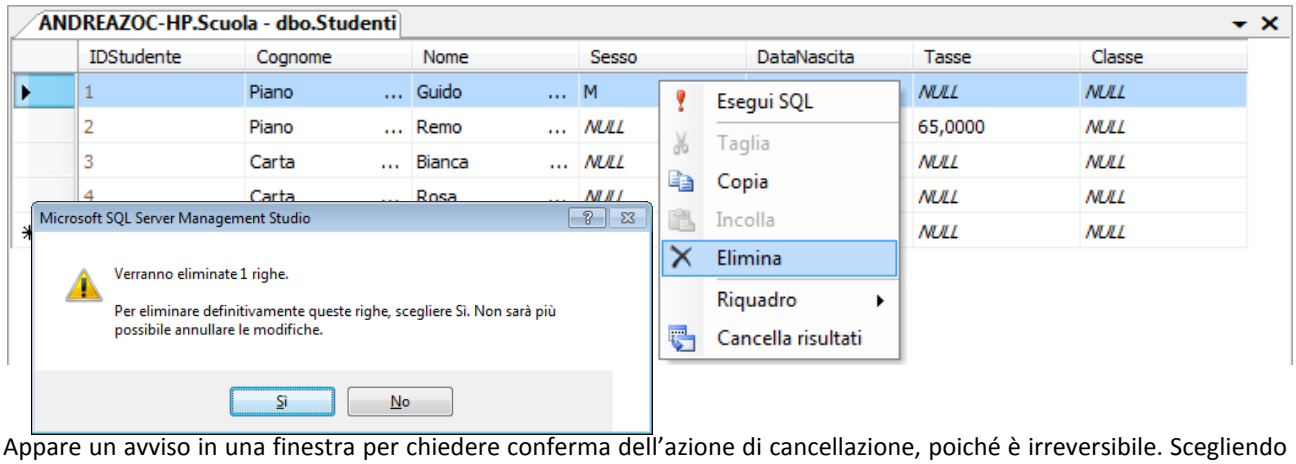



Per le celle vincolate Nel caso di modifica di valori vincolati appare l'avviso «cella in sola lettura» in basso vicino alla barra di stato.

Questa modalità consente di modificare i dati, se possibile; è sufficiente posizionarsi sulla riga della tabella e digitare le modifiche desiderate. La barra di stato riporta fedelmente la situazione corrente.



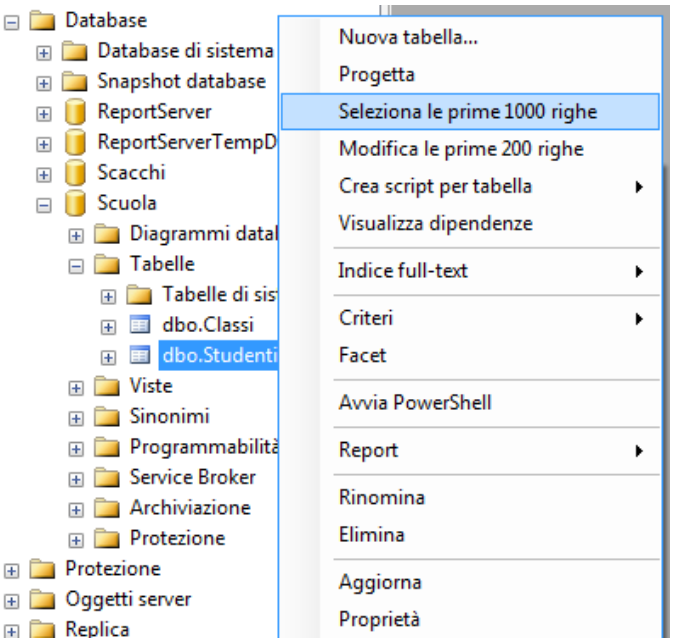

Per eliminare una riga occorre selezionare il record, con un clic sulla sinistra, sopra l'intestazione di riga, evocare il menu contestuale con un clic destro su un qualsiasi punto della riga, come nella figura seguente e scegliere **Elimina**:




Appare un avviso in una finestra per chiedere conferma dell'azione di cancellazione, poiché è irreversibile. Scegliendo Sì, la registrazione viene eliminata dalla tabella.

## 13-1 Visualizzazione dei dati

Oltre alla modifica è possibile visualizzare le prime 1000 righe di una tabella scegliendo la voce Visualizza opportuna dal menu contestuale della tabella desiderata.




La figura a lato mostra la voce del menu corrispondente alla richiesta.

Nella figura successiva illustra il risultato dell'esecuzione della query.

La sezione è divisa in riquadri ciascuno adibito a una specifica funzione.

Il riquadro in alto mostra la query SQL corrispondente e il numero 1000 indica il massimo numero da visualizzare.

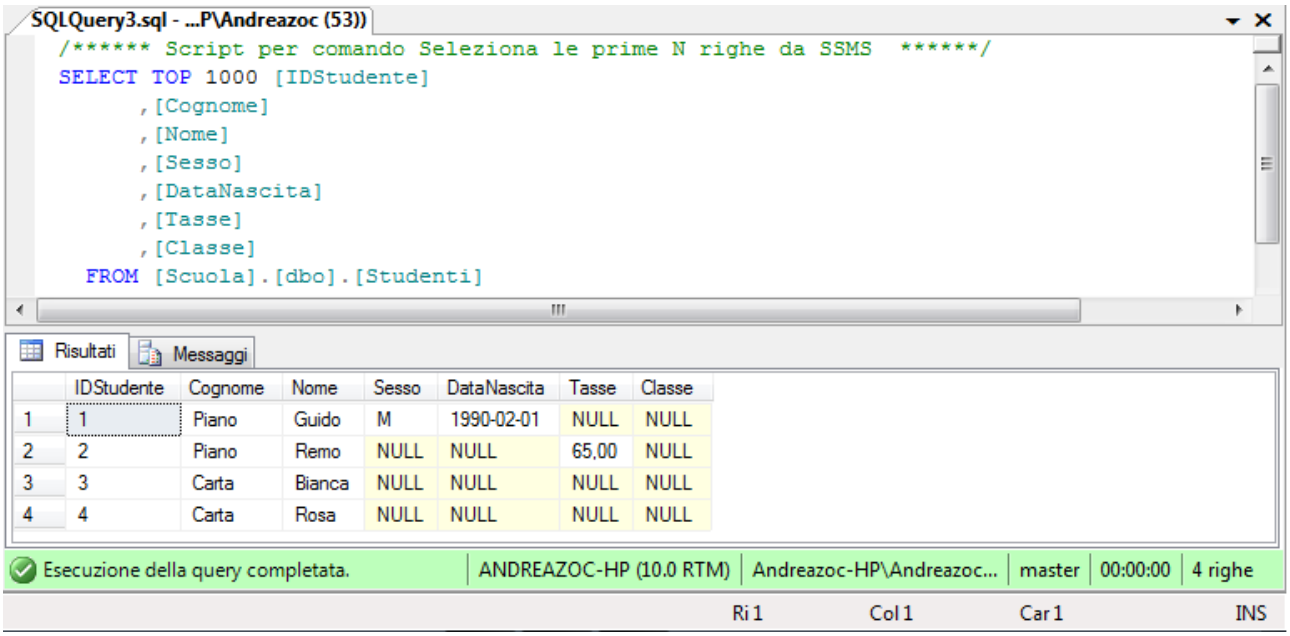
Se lo si desidera è possibile modificare la query e rieseguirla.

Il secondo riquadro è diviso in linguette (Risultati e Messaggi).

Infine l'ultimo riquadro è rappresentato dalla sola barra di stato, con informative sugli esiti.

Questa forma tabellare non consente operazioni come inserimenti, modifiche o cancellazioni.

Tuttavia l'Editor delle Query consente di modificare l'istruzione SQL che la genera e quindi permette di decidere quali dati visualizzare (filtri) e quali colonne mostrare (proiezioni).




La scrittura corretta della query non rigurda prettamente questo manuale, sebbene si discuterà della sintassi SQL nella specifica sezione.

La query SELECT consente di utilizzare:

- Filtri, mediante la clausola WHERE, utilizzando operatori come quelli di confronto (<, >, =) di string pattern (LIKE e i metacaratteri) e logici (AND, OR, NOT).
- Funzioni di aggregazione standard che sono AVG, COUNT, MIN, MAX, SUM.
- Raggruppamenti, mediante la clausola GROUP BY e indicando le colonne in base a cui raggruppare. Inoltre la clausola HAVING permette di filtrare su elementi di raggruppamento incluse le funzioni di aggregazione.
- Annidamenti (inserimenti di SELECT dentro altre SELECT) in diverse posizioni (es. in clausole FROM e WHERE).

I risultati di una query costituiscono tabelle dinamiche, ovvero collezioni di dati calcolate «al volo» in base ai dati archiviati. Queste tabelle dinamiche sono denominate spesso DYNASET o anche RESULT-SET o RECORDSET.

Transact-SQL

```
SELECT TOP 50 IDStudente, Cognome, Nome
FROM Studenti
WHERE Classe = '5';
```

## 13-2 Istruzioni elementari DML di SQL

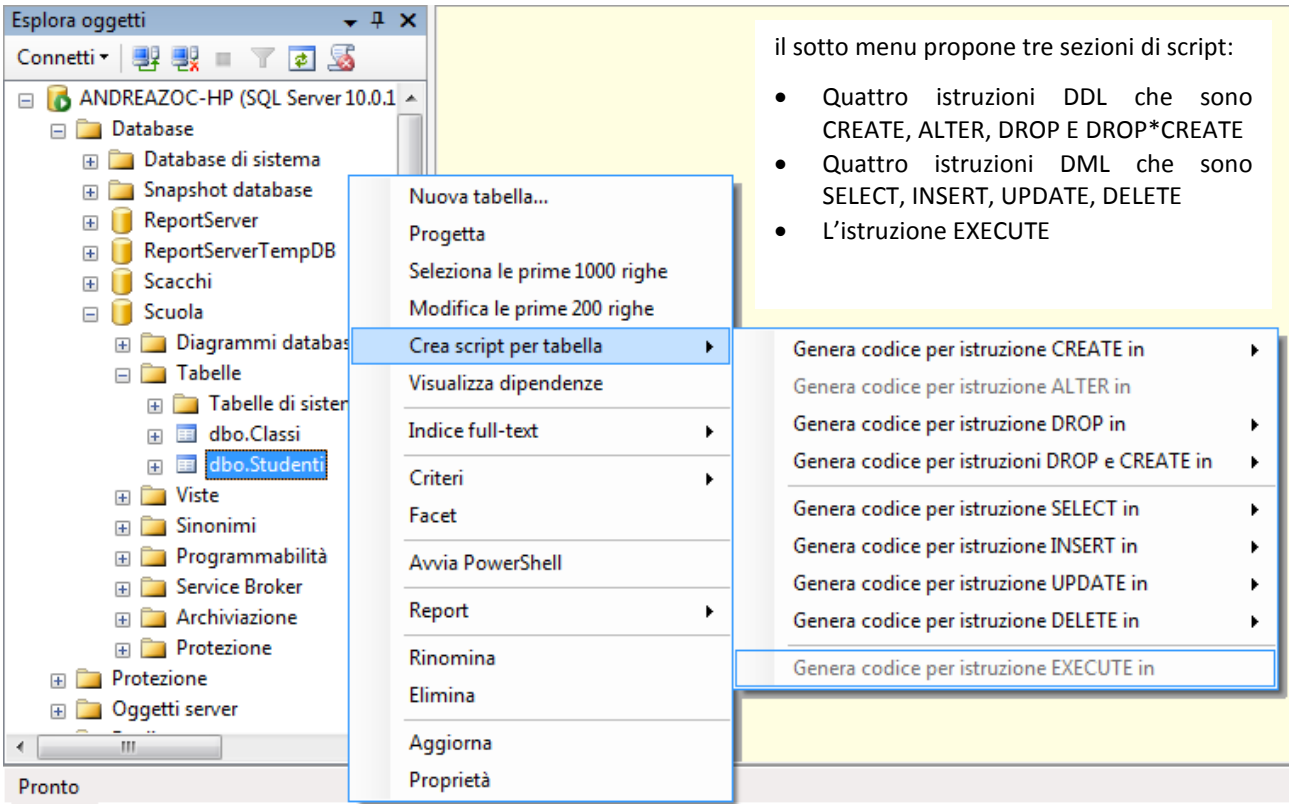
Oltre a sfruttare le possibilità visuali di SSMS, è possibile scrivere ed eseguire operazioni sui dati col linguaggio SQL. Tuttavia occorre soffermarsi a riflettere che spesso dette operazioni non sono espletate direttamente con semplici comandi SQL, ma vengono incorporate in procedure e funzioni, e invocate da utente (front-end) che aiutano l'utente a compilare correttamente di dati senza preoccuparsi della sintassi del linguaggio.

In ogni caso, le istruzioni di **data manipulation**, come definite nello standard ANSI-92, sono le seguenti:

- INSERT operazione di inserimento
- DELETE operazione di inserimento
- UPDATE operazione di inserimento
- SELECT operazione di inserimento

Come detto in altri punti, questo manuale non intende esaminare tutti i risvolti della sintassi SQL né l'opportunità di eseguire le possibili varianti di un comando. In questa sede ci si limiterà a esaminare le funzionalità di SSMS.

Come premessa osserviamo che per visualizzare la progettazione di comandi SQL conviene sfruttare il menu contestuale che appare a partire da una tabella qualsiasi; per esempio è possibile fare clic destro sulla tabella Studenti e scegliere la voce del menu **Crea script per tabella**;

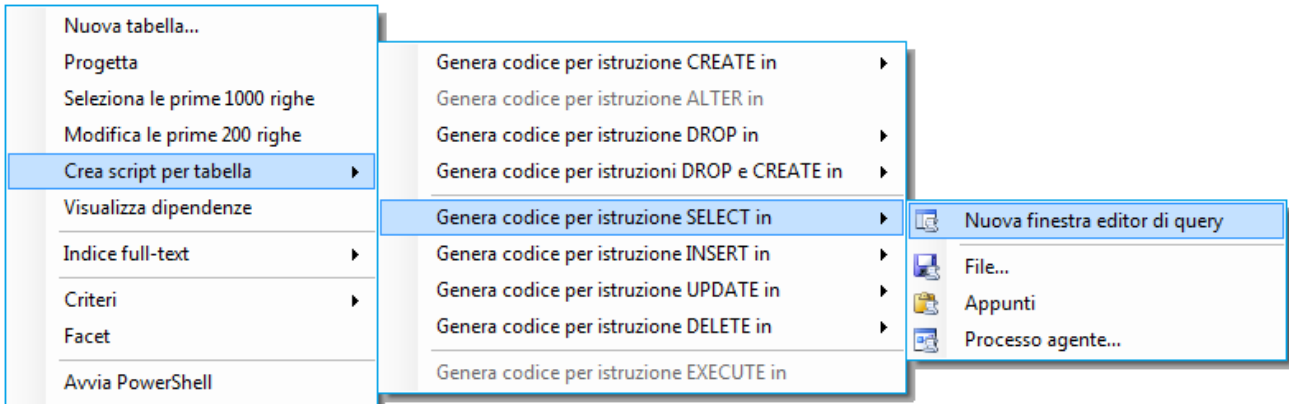



il sotto menu propone tre sezioni di script:

- Quattro istruzioni DDL che sono CREATE, ALTER, DROP E DROP*CREATE
- Quattro istruzioni DML che sono SELECT, INSERT, UPDATE, DELETE
- L'istruzione EXECUTE

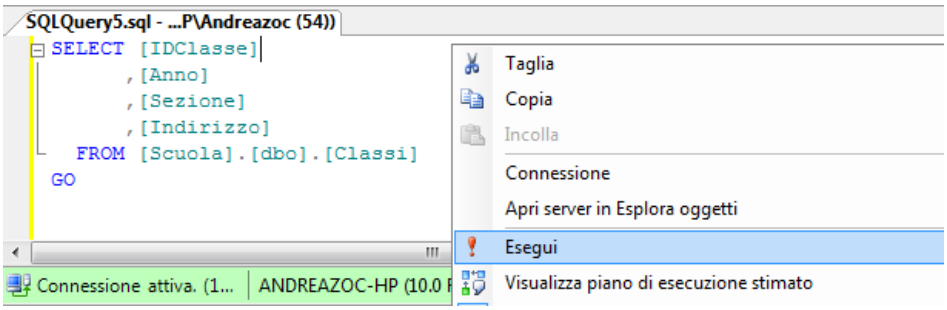
# 14 Istruzione SELECT

Oltre alla modalità vista prima, possiamo agire come si è agito per l'istruzione INSERT. Selezioniamo la tabella Classi e dal menu contestuale appena descritto, scegliamo la voce **Genera codice per istruzione SELECT** e da qui la voce **Nuova finestra Editor di Query**. La selezione della voce del menu mostra una finestra SQL.

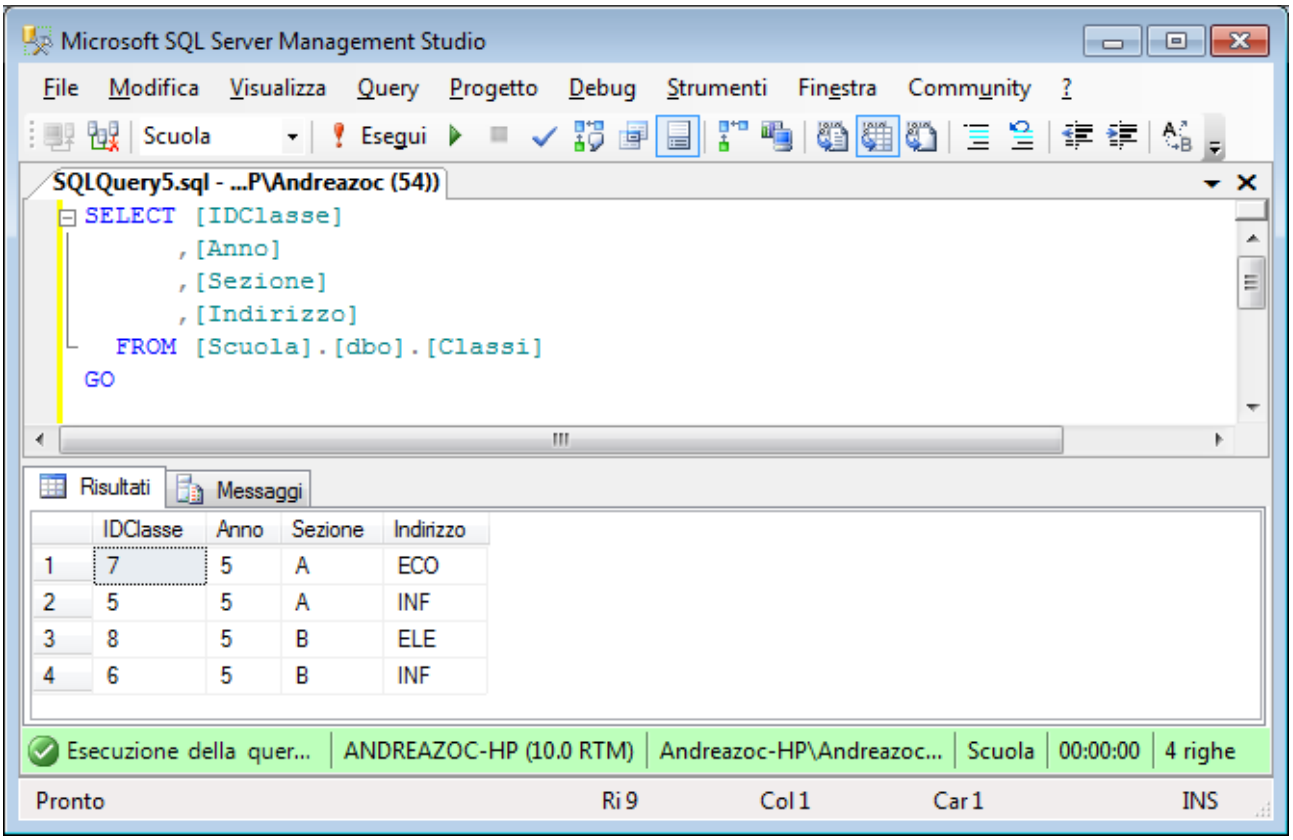



Avendo scelto l'opzione del menu compare la finestra della Query SQL, simile alla seguente:


È possibile eseguire la Query scegliendo la voce Esegui dal menu contestuale:




L'esecuzione mostra la finestra seguente:




La finestra riprodotta sopra è divisa in tre riquadri:

- Il riquadro in alto mostra la query SQL ; se lo si desidera, è possibile modificare la query e rieseguirla.
- Il secondo riquadro è diviso in linguette (Risultati e Messaggi).
- Infine l'ultimo riquadro è rappresentato dalla sola barra di stato, con informative sugli esiti.

Questa forma tabellare non consente operazioni come inserimenti, modifiche o cancellazioni.

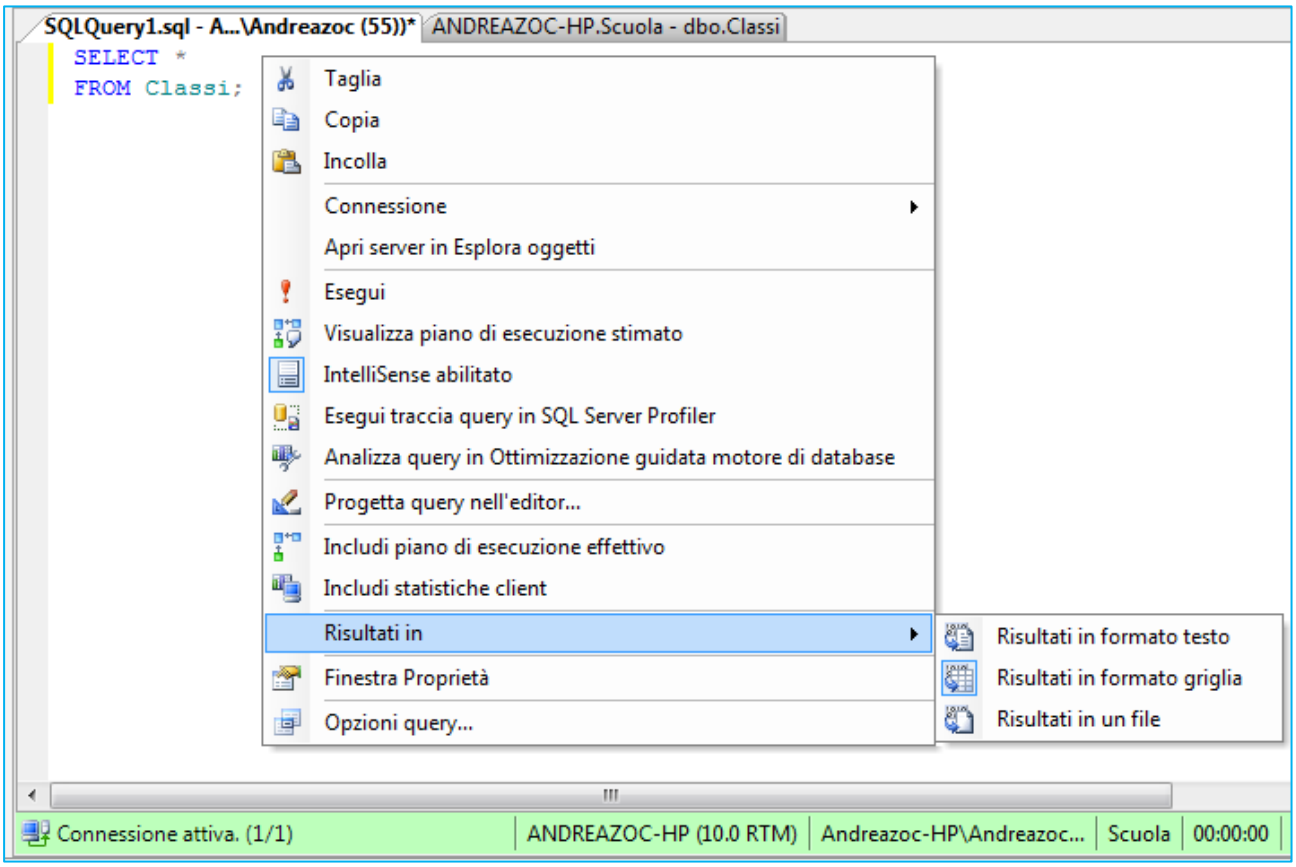
In generale la sintassi del comando SELECT è quella mostrata qui a lato.

```
SELECT [ ALL | DISTINCT ]
[ TOP expression [ PERCENT ] [ WITH TIES ] ]
    {
              *
      | { table_name | view_name | alias_name }.*
      | { column_name | [ ] expression | $IDENTITY | $ROWGUID }
      [ [ AS ] column_alias ]
      | column_alias = expression
    } [ ,...n ]
FROM table_name | view_name alias_name
WHERE filter_criteria
ORDER BY ordering_criteria
```

È in generale conveniente esplicitare le colonne che si desidera visualizzare, ed omettere quelle non rilevanti. Sebbene possa sembrare una pratica seccante da compiersi, essa migliora la velocità delle query nel caso di elaborazioni articolate che coinvolgono molteplici tabelle.

## 14-1 Modalità dei risultati delle query

SSMS propone diverse modalità per visualizzare i risultati di una interrogazione: modalità griglia, modalità testuale e modalità file.

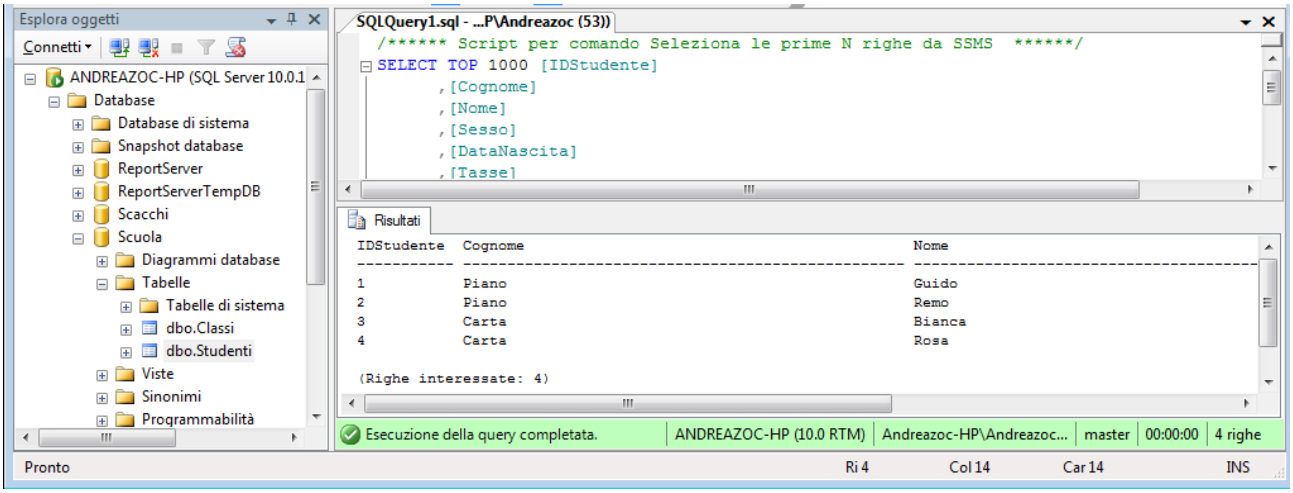



Per scegliere quella desiderata è sufficiente visualizzare il menu contestuale sulla finestra Editor di Query (dove si è digitata la query da eseguire) e selezionare l'opzione **Risultati in** che ci permette di scegliere fra le tre modalità:

- In formato testo, il risultato appare con il classico aspetto console
- In formato griglia, il risultato appare con l'aspetto visuale tabellare
- In un file, per salvare il risultato in un file

Dopo aver scelto il formato occorre eseguire la query (se già eseguita non si aggiorna in automatico).

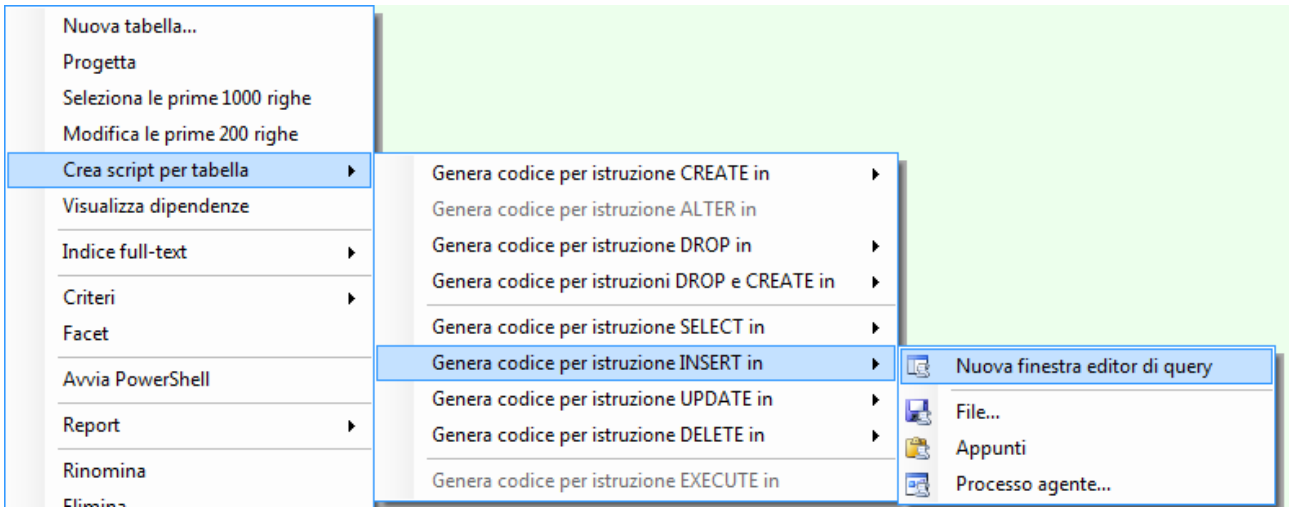
Ad esempio la visualizzazione in formato testo appare come nella figura seguente:



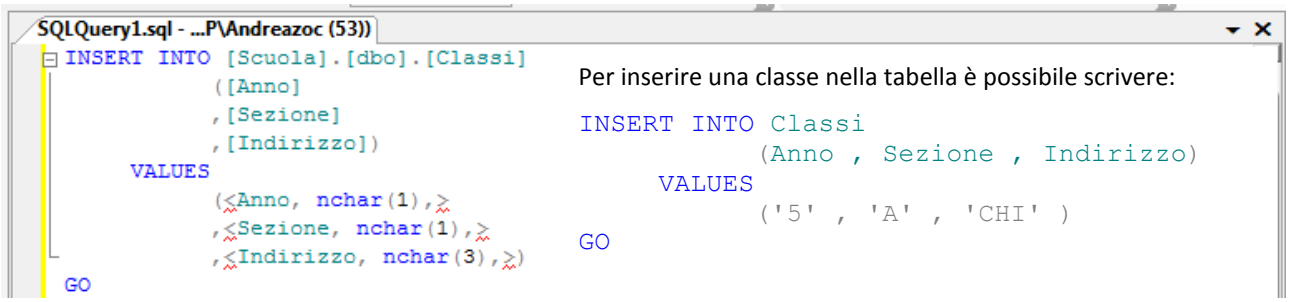
# 15 Istruzione INSERT

Per poter lavorare con dei dati di esempio dobbiamo anzi risultato appare con il classico aspetto console tutto inserire dei record nelle tabelle del database. In SQL Server l'inserimento si può effettuare mediante istruzioni SQL nella finestra Query Editor o direttamente con SSMS.

Selezioniamo la tabella Classi e dal menu contestuale appena descritto, scegliamo la voce **Genera codice per istruzione INSERT** e da qui la voce **Nuova finestra Editor di Query**.




Appare una finestra per editare la query.



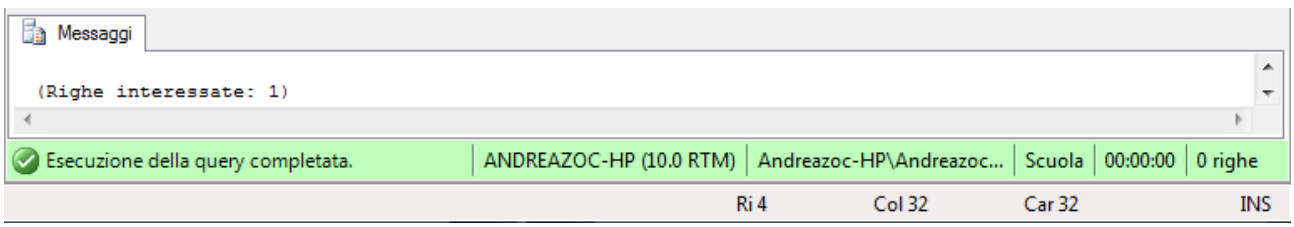

Per inserire una classe nella tabella è possibile scrivere:

```
INSERT INTO Classi
            (Anno , Sezione , Indirizzo)
      VALUES
            ('5' , 'A' , 'CHI' )
GO
```

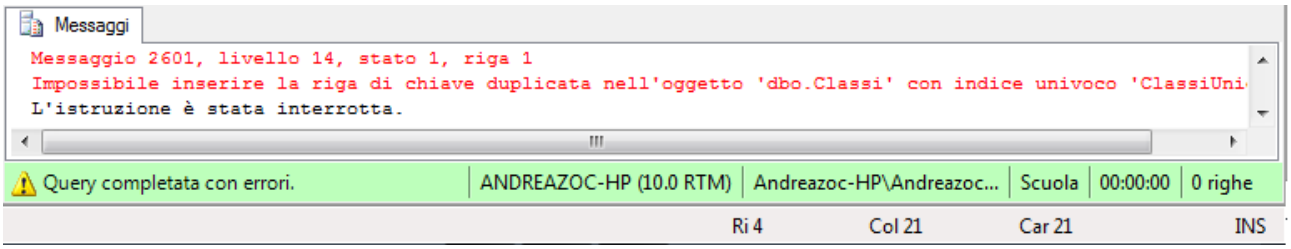
Si noti che nella sintassi rivisitata sono state eliminate alcune parti: il percorso della tabella, le parentesi quadre e le parentesi angolate. Per semplificare il comando si sono posti i campi (le colonne) sulla stessa riga e di seguito analogamente sono stati elencati i valori da inserire. Nell'esempio proposto si vuole inserire la quinta A chimica del nostro istituto. Si osservi anche l'uso dei singoli apici per delimitare i letterali stringa che si vuole inserire nei campi. Per completezza rispetto all'esempio proposto si suggerisce di inserire i seguenti record nella tabella:

| Classi | Anno | Sezione | Indirizzo |
|---|---|---|---|
| | 5 | A | CHI |
| | 5 | B | CHI |
| | 5 | A | ELE |
| | 5 | B | ELE |
| | 5 | A | INF |
| | 5 | B | INF |

Ogni inserimento va confermato, premendo il pulsante **Esegui** nella barra degli strumenti o il tasto F5 della tastiera. Se lo script non contiene errori dovremmo vedere il seguente risultato

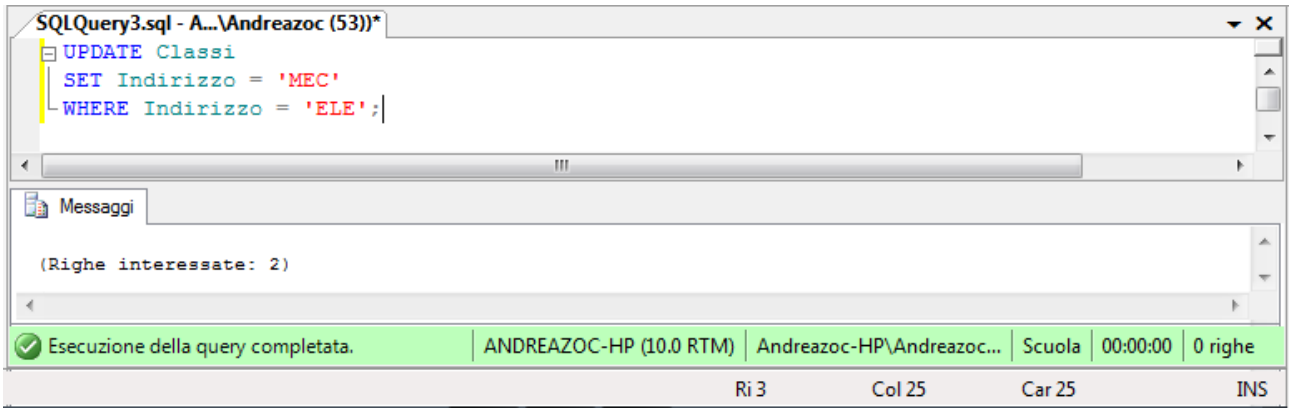


che indica la corretta elaborazione del comando SQL. Se il comando solleva errori la sua esecuzione sarà abortita e comparirà un messaggio simile al seguente:




# 16 Istruzione UPDATE

Se si desidera modificare dei dati già presenti in tabella si può procedere come già visto in precedenza. Visualizziamo il menu contestuale sulla tabella Classi e scegliamo la voce **Genera codice per istruzione UPDATE** e da qui la voce **Nuova finestra Editor di Query**.




In generale la sintassi del comando UPDATE è la seguente:

```
UPDATE
    [ TOP ( expression ) [ PERCENT ] ]
    [[ server_name . database_name . schema_name .
    | database_name .[ schema_name ] .
    | schema_name .]
table_or_viewname
    SET
        { column_name = { expression | DEFAULT | NULL }
          | column_name { .WRITE ( expression , @Offset , @Length ) }
          | @variable = expression
          | @variable = column = expression [ ,...n ]
        } [ ,...n ]
    [FROM { <table_source> } [ ,...n ] ]
    [ WHERE { <search_condition>]
```

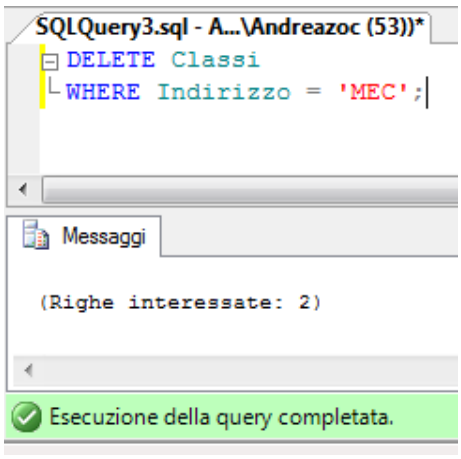
# 17 Istruzione DELETE

Se si desidera eliminare dati dalle tabelle si può procedere come già visto in precedenza. Visualizziamo il menu contestuale sulla tabella Classi e scegliamo la voce **Genera codice per istruzione DELETE** e da qui la voce **Nuova finestra Editor di Query**.

In generale l'istruzione SQL è la seguente:

```
DELETE tablename
WHERE where_condition
```

Come ultima annotazione è il caso di osservare che SSMS propone una barra per le query SQL ma la sua discussione è rimandata all'apposita Appendice alla fine del documento.



# SQL e Data Definition

# 18 Istruzioni elementari DDL di SQL

## 18-1 Cosa comprende il DDL

Il DDL è il sottoinsieme dei comandi SQL che permettono di modificare le strutture degli oggetti del database. Tipicamente le operazioni DDL sono quelle di definizione dei dati e quelle di programmabilità.

I comandi frequenti di **definizione** dei dati sono:

- CREATE DATABASE La creazione del database
- CREATE TABLE la creazione di una tabella
- CREATE VIEW la creazione di una vista

I comandi frequenti di **programmabilità** sono:

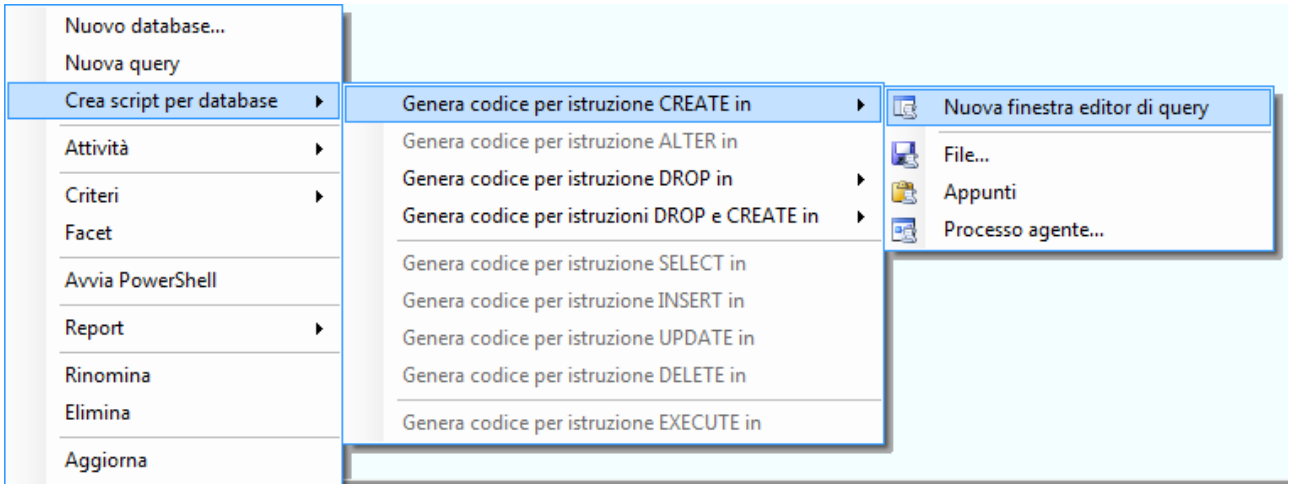
- CREATE PROCEDURE La creazione di una procedura memorizzata
- CREATE FUNCTION la creazione di una funzione
- CREATE TRIGGER la creazione di un trigger, una procedura automatica legata a un evento
- CREATE RULE la creazione di una regola, o asserzione, ormai deprecata e caduta in disuso

Usando ALTER al posto di CREATE si modifica un oggetto esistente.

Usando DROP al posto di CREATE si elimina un oggetto esistente.

## 18-2 Comando CREATE DATABASE

La creazione del database è stata già discussa in apertura di questo documento. È il caso però di ricordare che è sempre possibile creare un database con la sintassi SQL completa. Per visualizzare il codice SQL che ha generato un database è sufficiente selezionare il database appena creato e col tasto destro visualizzare il menu contestuale, da cui scegliere la voce **Genera codice per istruzione CREATE** e quindi la voce **Nuova finestra editor di query**:




**La sintassi completa è possibile analizzarla qui:** http://msdn.microsoft.com/en-us/library/ms176061.aspx

Una sintassi rapida per CREATE DATABASE è la seguente:

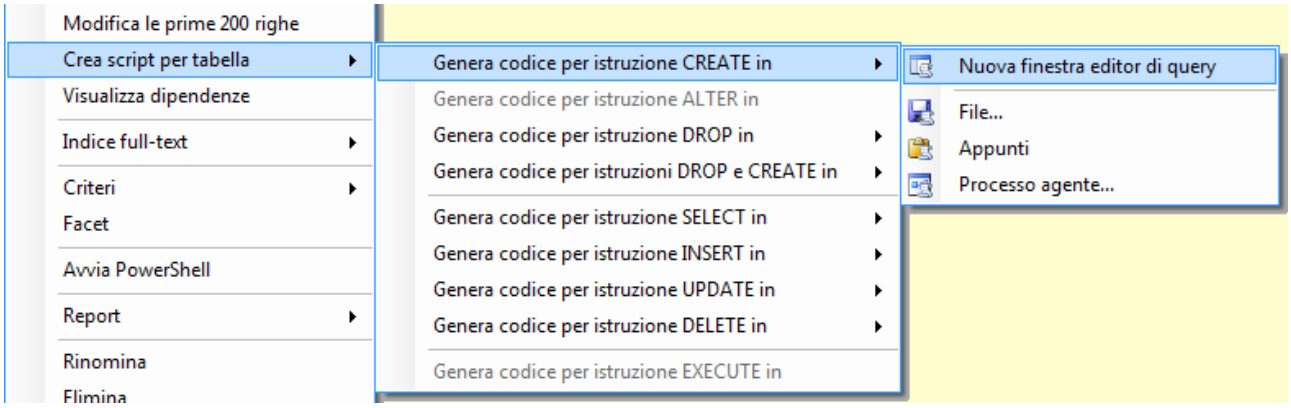
Transact-SQL

```
CREATE DATABASE NomeDatabase
ON PRIMARY
(  NAME   =    'NomeDatabase',
   FILENAME  =  'C:\PercorsoSalvataggioFileMDF\NomeDatabase.mdf' ,
   SIZE = 3072KB,
   MAXSIZE = UNLIMITED,
   FILEGROWTH = 1024KB
)
```

### 18-3 Comando CREATE TABLE

La creazione della tabella è stata già discussa subito dopo la creazione del database.

Per visualizzare il codice SQL che ha generato **una tabella** è sufficiente selezionare la tabella appena creata e col tasto destro visualizzare il menu contestuale, da cui scegliere la voce **Crea script per tabella** → **Genera codice per istruzione CREATE** e quindi la voce **Nuova finestra editor di query**:




Anche in questo caso però occorre ricordare che è sempre possibile creare una tabella con la sintassi SQL che definisce anche eventuali vincoli.

**La sintassi completa è possibile analizzarla qui:** http://msdn.microsoft.com/en-us/library/ms174979.aspx

# 19 Introduzione alle viste

### 19-1 Concetto di Vista

Una vista è una sorta di tabella virtuale che non contiene fisicamente dei dati ma consiste di una query che il programmatore deve definire insieme al momento della sua creazione. Quando un codice fa riferimento ad (utilizza una) vista, in server elabora al momento i risultati corrispondenti e li restituisce, come se fosse una tabella vera e propria. La definizione della query richiede di definire anche le colonne corrispondenti, sebbene i tipi siano derivati dalle tabelle originali.

Le viste sono utilizzate per diversi motivi: il primo riguarda la sicurezza, poiché è possibile nascondere i dati effettivi all'utilizzatore finale. Per esempio la query contenuta nella vista può elaborare più tabelle (spesso giunte da join) che appariranno all'utilizzatore come un'unica tabella. Inoltre è possibile nascondere alcune colonne delle tabelle originali e mostrare solo i dati desiderati (per es. mostrare l'elenco dei dipendenti senza mostrare il loro salario oppure mostrare l'elenco degli studenti di una classe, ma senza indicazioni personali come situazioni di disabilità). Infine è possibile restituire solo dati raggruppati e aggregati ma non i dati originari (quindi medie e calcoli).

A differenza di procedure e funzioni, una vista TSQL non ammette variabili, istruzioni procedurali (come IF o WHILE) e opzioni; semplicemente può elaborare una semplice query. È possibile che una vista utilizzi al suo interno un'altra vista, per realizzare calcoli di maggiore aggregazione.

Sempre per incrementare la sicurezza del database SQL server ammette la possibilità di crittografare le viste e rendere invisibili le query racchiuse al suo interno.

### 19-2 Viste aggiornabili

Se la vista risponde a alcuni requisiti può permettere l'inserimento, la modifica o la cancellazione di dati; per esempio è possibile eliminare un dipendente dal database, pur agendo sulla sola vista e non potendo accedere ai suoi dati personali. Una vista di questo tipo è detta Vista Aggiornabile.

Una Vista aggiornabile, permette di eseguire su di essa comandi DML come INSERT, UPDATE e DELETE. Poiché una vista non è altro che una connessione verso una o più tabelle, i comandi andranno a modificare le tabelle correlate.

Non tutte le viste sono aggiornabili e sono scrivibili. Si parla di viste "updatable" quando il DBMS è in grado di stabilire una mappatura inversa tra i record presenti nella vista e quelli nelle tabelle; su queste viste si possono eseguire comandi UPDATE e DELETE. Una vista è "insertable" quando il DBMS è in grado di inserire il record nella tabella corretta e quindi ammette comandi INSERT.

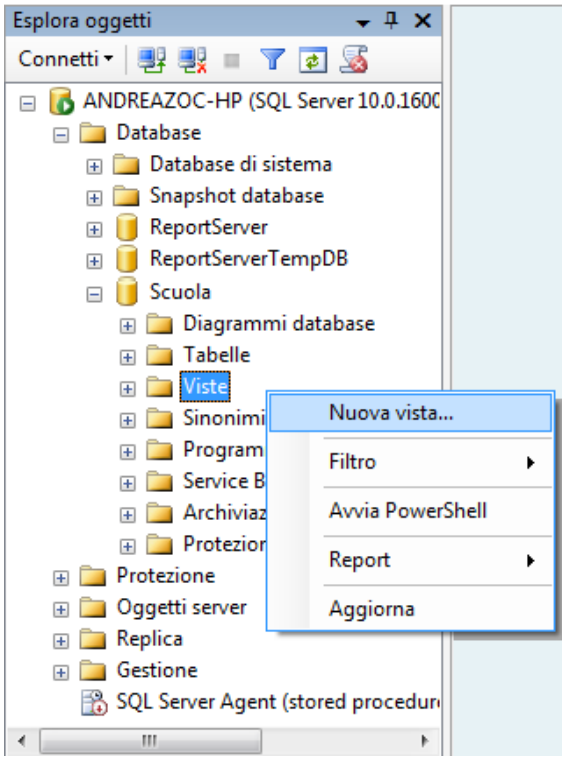
Una tabella che restituisce dati aggregati (es. con funzioni di aggregazione) non è aggiornabile né scrivibile.

Generalmente non è modificabile una vista basata su una UNION.

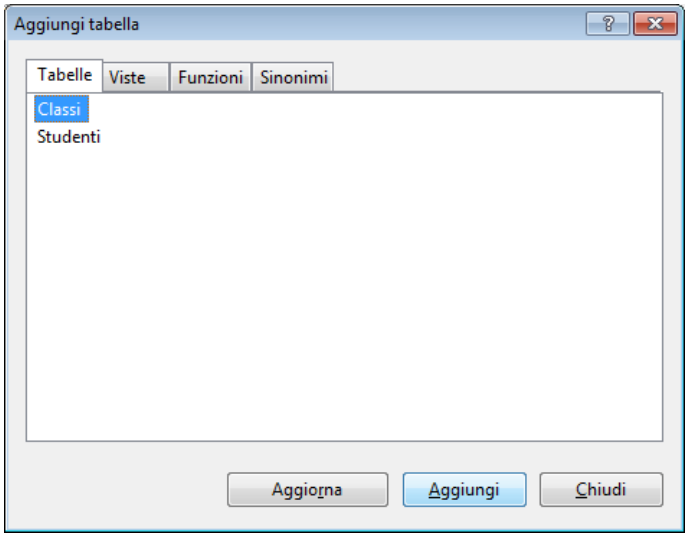
## 19-3 Creazione di una Vista

Come già appreso per le tabelle e per gli indici SSMS consente di creare anche le viste con pochi clic del mouse utilizzando quasi solo l'interfaccia visuale e pochissimo codice. Il punto di partenza è, come di consueto, l'**Esplora Oggetti**, dove dobbiamo individuare e selezionare il nodo delle viste.

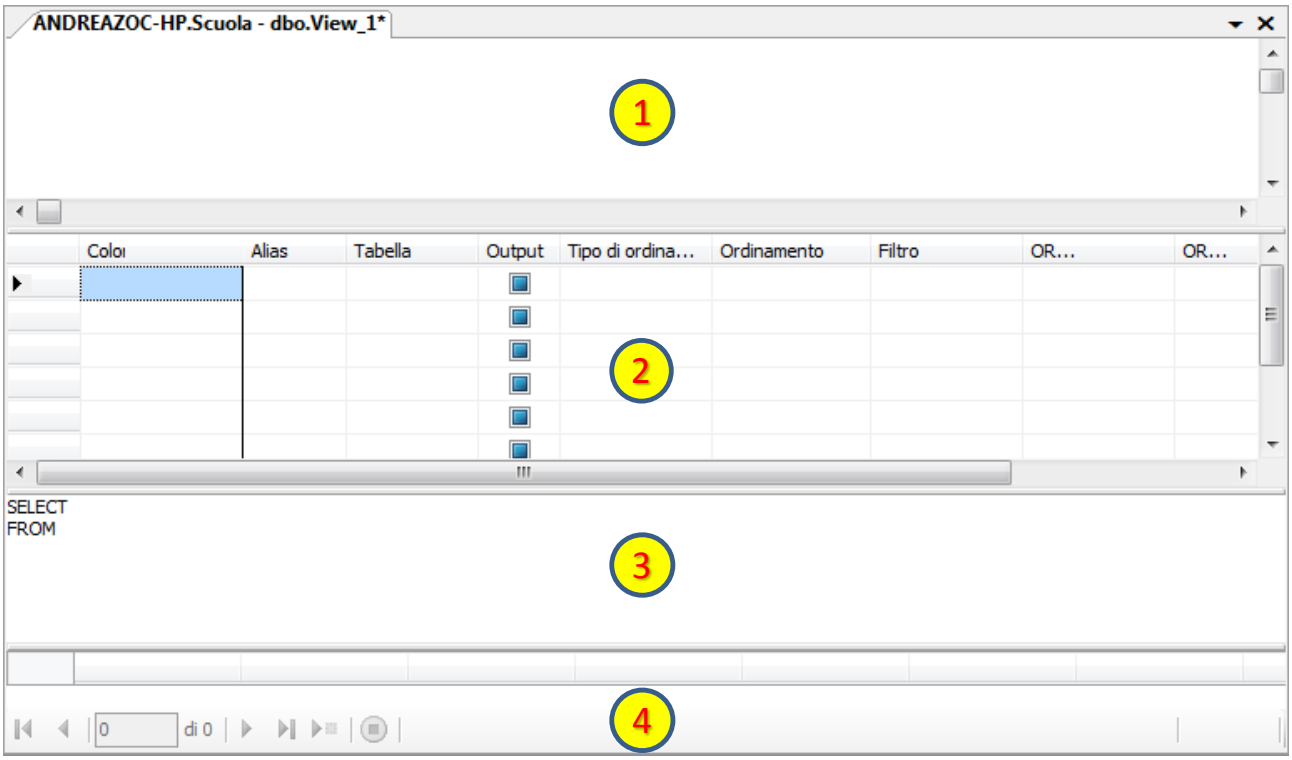
Le viste sono il terzo nodo del database e, col consueto menu contestuale, è possibile scegliere **Nuova vista ...**




L'interfaccia chiede di inserire tabelle ma possiamo anche non sceglierne alcuna e costruire la vista direttamente dall'editor.

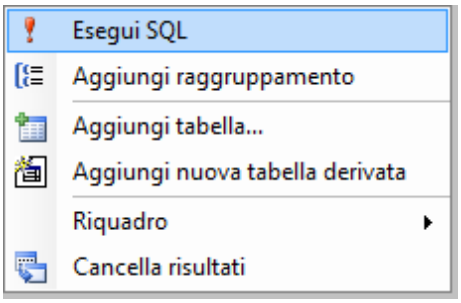



L'interfaccia della vista presenta quattro zone distinte, ciascuna con un compito specifico:




- La prima sezione (area Diagramma) serve per la rappresentazione grafica delle tabelle su cui la vista lavora.
- La seconda sezione (area Criteri) serve per la composizione visuale della query (Query By Example).
- La terza sezione (area SQL) è riservata alla composizione testuale della query.
- La quarta sezione (area Risultati) è la visualizzazione dei risultati.

Con un clic destro sulla prima sezione (Diagramma) si ottiene il menu contestuale.

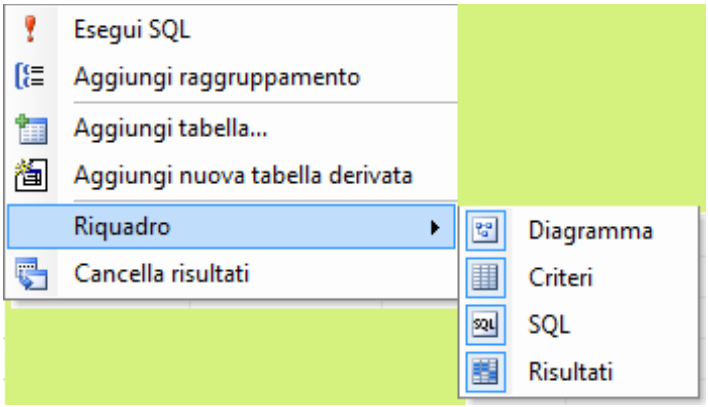



La voce **Esegui SQL** serve per lanciare l'esecuzione della query corrispondente al corpo della vista; attualmente è vuota e non avrebbe effetti.

La voce **Aggiungi raggruppamento** permette di inserire o modificare la clausola GROUP BY della query posta nel corpo della vista.

La voce **Aggiungi tabella** consente di aggiungere tabelle alla query, in modo analogo alla finestra introduttiva della creazione della vista.

La voce **Aggiungi nuova tabella derivata** permette di inserire delle query nidificate nella clausola FROM della query principale della vista.



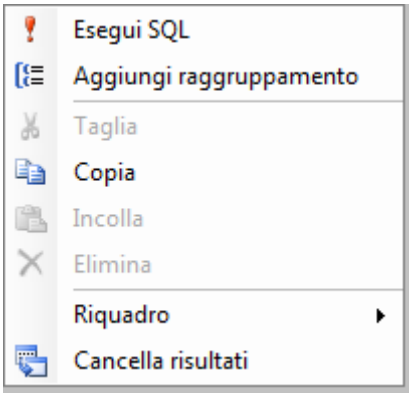

La voce **Riquadro** visualizza le sezioni che si sta descrivendo proprio adesso:

- Diagramma (sezione 1)
- Criteri (sezione 2)
- SQL (sezione 3)
- Risultati (sezione 4)

La voce Cancella Risultati ripulisce la quarta sezione.

Con un clic destro sulla seconda sezione (Criteri) si ottiene un diverso menu contestuale, simile al precedente ma alcune voci sono scomparse e invece propone le azioni Taglia Copia Incolla Elimina.

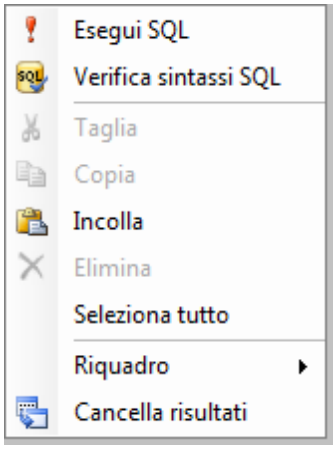
Per il momento non esamineremo questa sezione per concentrarci sulla terza sezione, quella successiva.




La terza sezione è quella riservata al codice SQL. È la sezione che richiede maggiore esperienza e attenzione e la conoscenza della sintassi SQL.

Con un clic destro sulla terza sezione (SQL) si ottiene un altro menu contestuale, simile all'altro ma con la voce **Verifica sintassi SQL**. Questa opzione permette di analizzare quanto scritto senza eseguire la query e quindi senza il rischio di modificare il database.

La voce **Seleziona tutto** è quella consueta degli editor testuali.



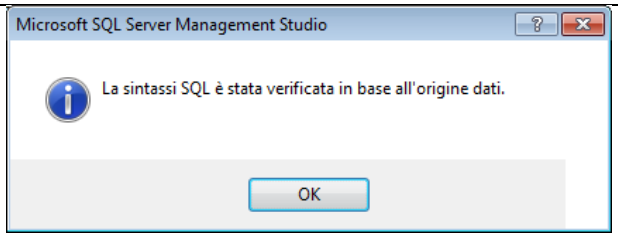

## 19-4 Dichiarazione di una Vista

Dopo aver creato una nuova vista è possibile procedere in molto modi diversi. Proviamo però a costruire la vista scrivendo direttamente il codice della query associata. Per fare questo, si può scrivere nella terza sezione il codice seguente:

Transact-SQL

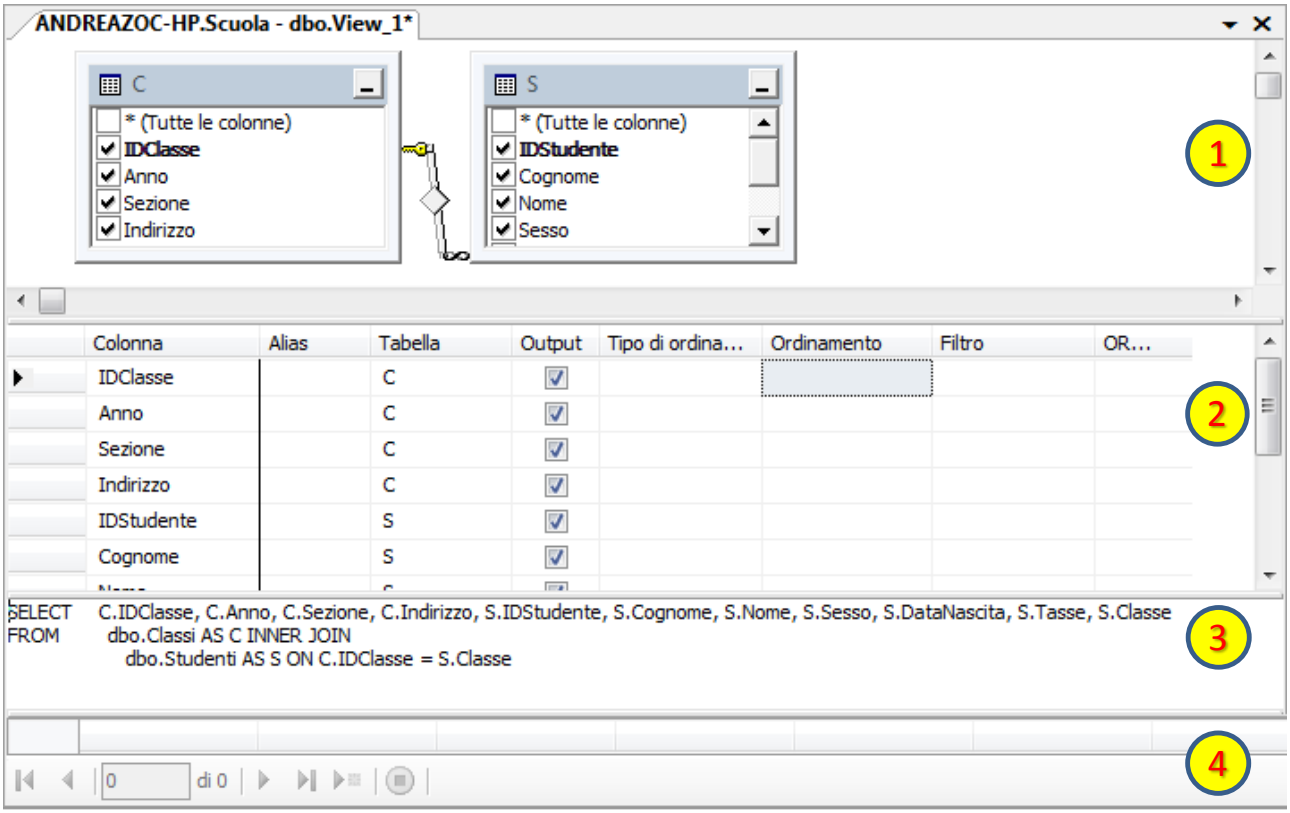
```
SELECT C.*, S.*
FROM Classi AS C, Studenti AS S
WHERE  C.IDClasse = S.Classe;
GO
```

Adesso dal menu contestuale controlliamo la sintassi SQL




Se il codice è convalidato apparirà una finestra simile alla figura qui a lato.

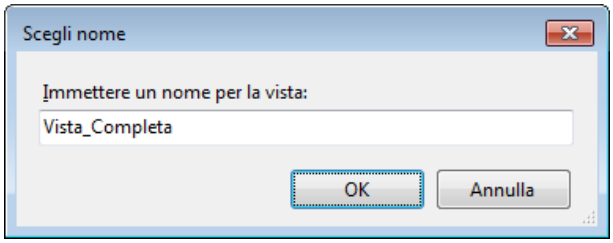
Appena si conferma con OK, il sistema si adegua e modifica le varie sezioni per adattarsi alla query appena scritta; la figura sottostante mostra un esempio.



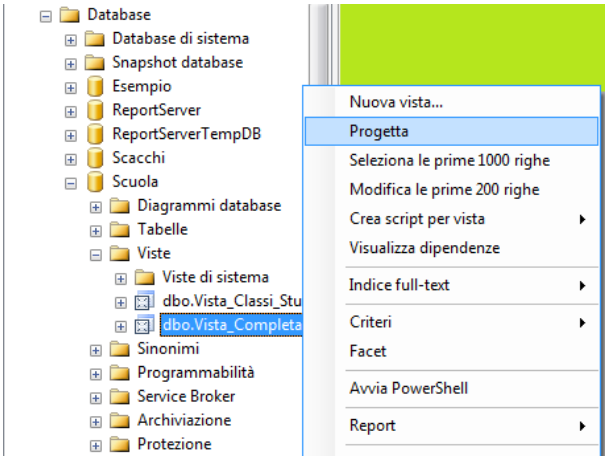

Osserviamo le sezioni modificate in automatico:

(1) Il diagramma della Vista mostra le due tabelle coinvolte e le lega con l'associazione definita nella giunzione (chiave primaria ← → chiave esterna);
(2) Nella sezione dei criteri della Vista sono visualizzate le colonne coinvolte nella vista (che si potrebbero ordinare ad esempio);
(3) La sezione SQL si è modificata per utilizzare la sintassi delle INNER JOIN come in SQL standard; anche i campi della clausola SELECT sono stati modificati per evitare il simbolo * e elencare i campi visualizzati.
(4) Nessun risultato appare nella sezione perché la query non è stata eseguita ma solo validata; d'altronde si osservi che finora non vi sono dati nel database.

Infine salviamo la vista con un nome opportuno:



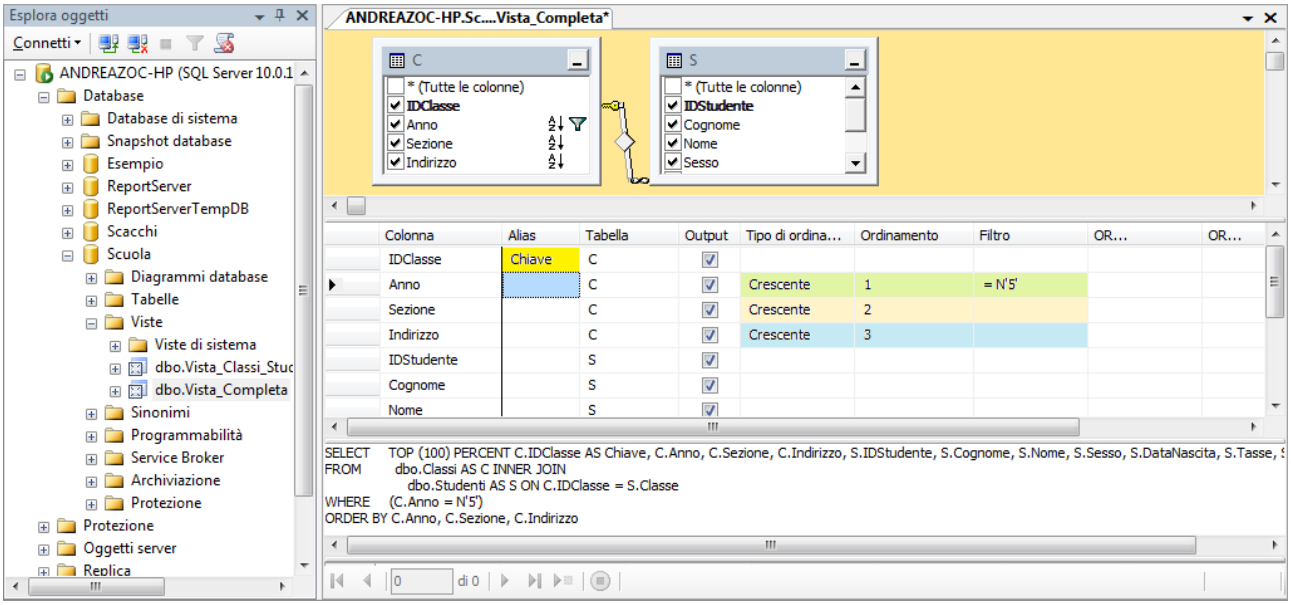
È possibile modificare il corpo della vista e le sue proprietà mediante il menu contestuale e la voce Progetta, che mostra la finestra di progettazione identica a quella della sua costruzione.



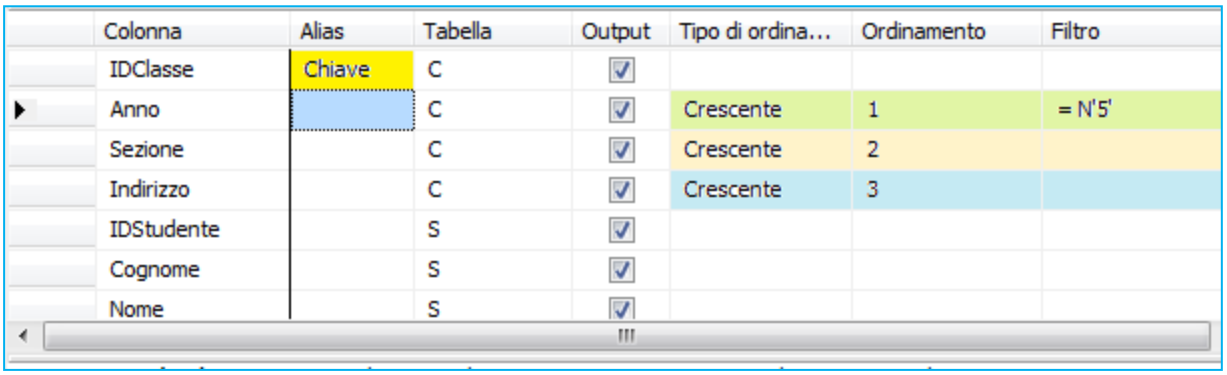

Agendo sul riquadro visuale oppure sul codice SQL possiamo modificare la vista.

Digitando nella casella **Alias** possiamo rinominare la colonna come apparirà all'utilizzatore della vista finale.

Digitando nella casella **Tipo di Ordinamento** possiamo decidere se crescente o decrescente (o non presente) e nella colonna a fianco la priorità di ordinamento tra le colonne.




Digitando nella casella Filtro è possibile decidere quali record restituire nella tabella calcolata dalla vista.



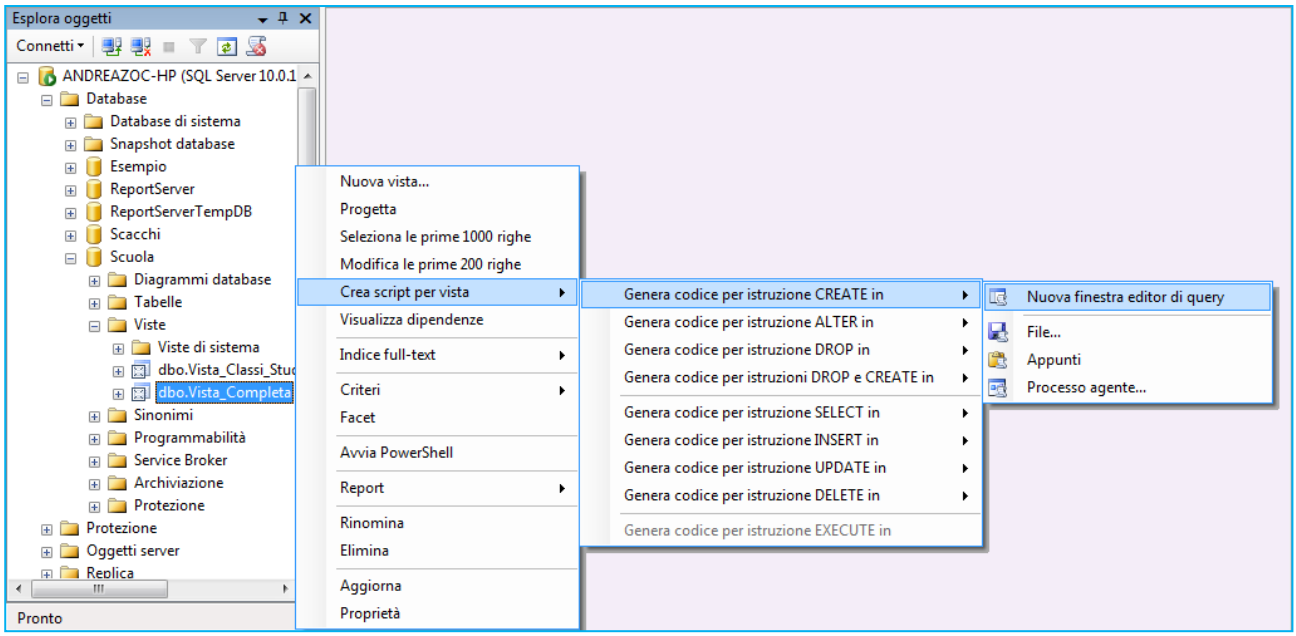

Modificando in visuale in secondo riquadro, il sistema adatta il codice TSQL sottostate nel terzo riquadro; e viceversa.

## 19-5 Pulsanti della barra delle Viste

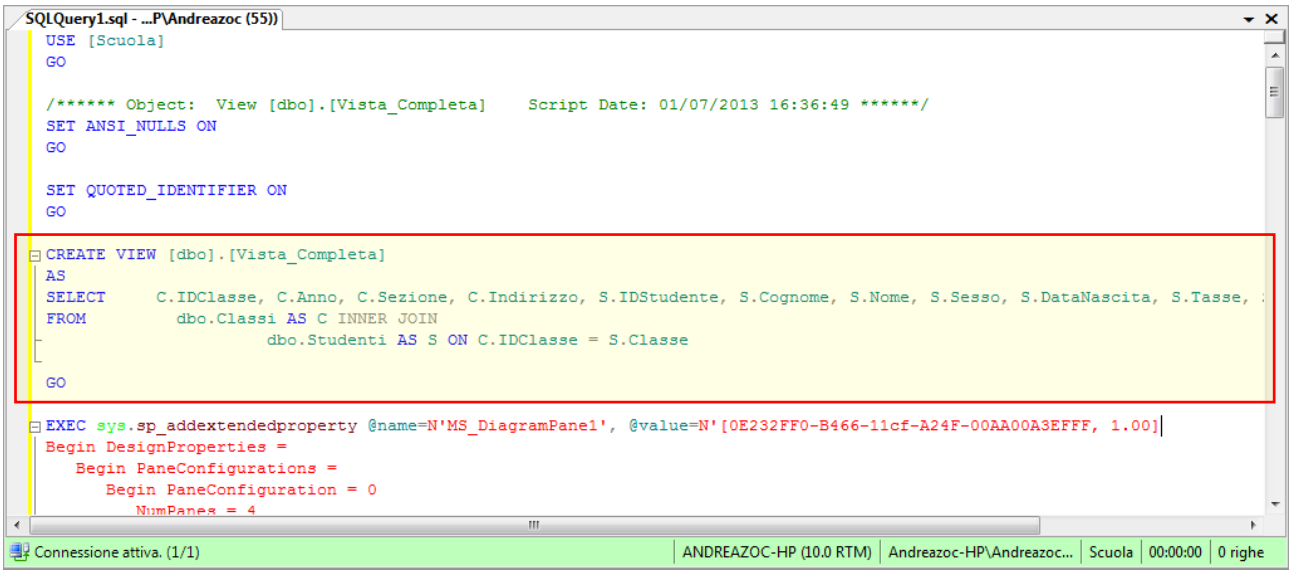
La Barra Progettazione Viste è descritta nella opportuna Appendice.

## 19-6 Comando CREATE VIEW

La sintassi della creazione di un vista può essere facilmente richiesta al sistema, semplicemente evocando il menu contestuale sulla vista desiderata e scegliendo la voce **Crea script per vista** → **Genera codice per istruzione CREATE** e quindi la voce **Nuova finestra editor di query** come nella figura riprodotta qui sotto:




Dopo aver lanciato il comando compare la finestra di Progettazione della Vista, con il comando CREATE VIEW di SQL:




Come per i qualsiasi altra struttura dati è possibile creare una vista con la sintassi SQL che può definire anche gli accessi e gli usi.

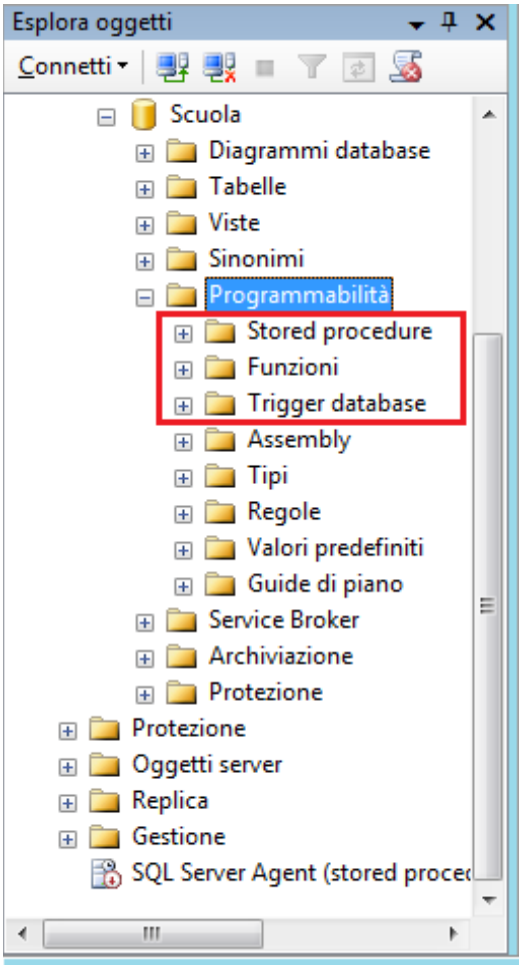
**La sintassi completa è possibile analizzarla qui:** http://msdn.microsoft.com/en-us/library/ms187956.aspx

```
CREATE VIEW [ schema_name . ] view_name [ (column [ ,...n ] ) ]
[ WITH <view_attribute> [ ,...n ] ]
AS select_statement
[ WITH CHECK OPTION ] [ ; ]

<view_attribute> ::=
{
    [ ENCRYPTION ]
    [ SCHEMABINDING ]
    [ VIEW_METADATA ]     }
```

# PROGRAMMABILITÀ DI SQL SERVER

# 20 Routine definite dall'utente

## 20-1 Routine precompilate di SQL




Le procedure definite dall'utente sono formalmente un aspetto del DDL (**Data Description Language**) poiché sono considerate alla stregua delle strutture dei dati, sebbene in realtà presidino aspetti processuali piuttosto che descrittivi.

SQL prevede tre tipi di routine differenti per tipologia di esecuzione e di risultato:

- Le **stored procedure** o procedure memorizzate, routine che generalmente non restituiscono valori ma accettano parametri di ingresso e/o uscita;
- Le **funzioni** definite dall'utente che, oltre ad accettare parametri di ingresso e/o uscita, restituiscono anche un singolo valore (risultato);
- I **trigger**, che sono sequenze di comandi attivate da eventi, rivolte alla gestione della congruenza del database.

Queste routine sono dichiarate con un identificatore (un nome) e un corpo scritto con un linguaggio costituito da istruzioni SQL e istruzioni procedurali, utilizzate soprattutto per la gestione delle variabili e per il controllo del flusso. Queste routine costituiscono un blocco unitario, salvato su server e invocabili quando necessario.

Entrambe queste routine sono precompilate e restano pronte per una successiva invocazione.

SQL Server presume che ognuno di questi oggetti sarà eseguito più volte e per questo motivo all'atto della compilazione viene creato un piano di esecuzione ottimizzato della query, corredato dai dettagli su come eseguire al meglio le istruzioni. Come per gli altri oggetti di un database è possibile assegnare un determinato livello di sicurezza a stored procedure e funzioni, in modo che soltanto certi utenti possano eseguirle.

## 20-2 Vantaggi e svantaggi

Il principale motivo delle routine archiviate è che non è sempre possibile salvare una query nella memoria del server (è possibile salvarne il codice su un file separato e indipendente, da cui copiarlo e eseguirlo in seguito). Ma se è richiesto di eseguire con frequenza una sequenza di comandi conviene memorizzare tale sequenza sul server. Questo è reso possibile mediante stored procedure e funzioni.

Uno dei vantaggi delle routine memorizzate è che offre la possibilità di richiamare una procedura complicata e articolata già archiviata all'interno del database, sollevando il client dal compito di scriverle; in questo senso la sequenza di comandi dovrebbe essere più sicura. In questo modo si rende standardizzata una sequenza di comandi a tutti i client.

Un altro vantaggio è che si assicura l'uso pertinente del linguaggio, anche avvantaggiandosi dei comandi specifici del sistema di database specifico. La possibilità di scrivere elaborazioni complesse non altrimenti realizzabili usando unicamente comandi SQL, consente di sfruttare le istruzioni dei linguaggi strutturati. Inoltre La compilazione è elaborata una volta sola, e il codice compilato sul server è generalmente ottimizzato e sfrutta le opzioni del server. Ogni volta che la procedura viene richiamata viene semplicemente eseguita e questo aumenta significativamente le prestazioni.

Generalmente le prestazioni medie sono migliori, poiché le routine sono eseguite dal lato server (server-side) e il numero di dati scambiati tra client e server sarà minore.

È possibile criptare le procedure, in modo che il client non possa accedere al codice da eseguire e non sappia come e da dove siano ottenuti i risultati resi. È utile anche per questioni di copyright del codice.

L'utilizzo di stored procedure permette di mantenere librerie di funzioni da utilizzare all'interno del database stesso. Potenzialmente si potrebbe eseguire ogni operazione richiamando una diversa procedura, senza conoscere la struttura di un database magari complesso, o avendone una conoscenza limitata.

Proprio per questo motivo, l'amministratore di database può a volte evitare di concedere i permessi di modifica o di lettura su molte tabelle a determinati utenti, concedendo semplicemente il permesso di eseguire le stored procedure.

Gli svantaggi principali sono invece i seguenti:

- Le routine server side aumentano il carico di lavoro per il server.
- Il server condiziona la scelta del linguaggio da usare, talvolta fuori standard, costringendo a un impegno specifico.
- Rendono ardua la pratica dei modelli three-tier (logica a tre livelli) dove si distribuisce il carico di lavoro su tre elementi (client, server e software intermedio di gestione).

# 21 Procedure memorizzate

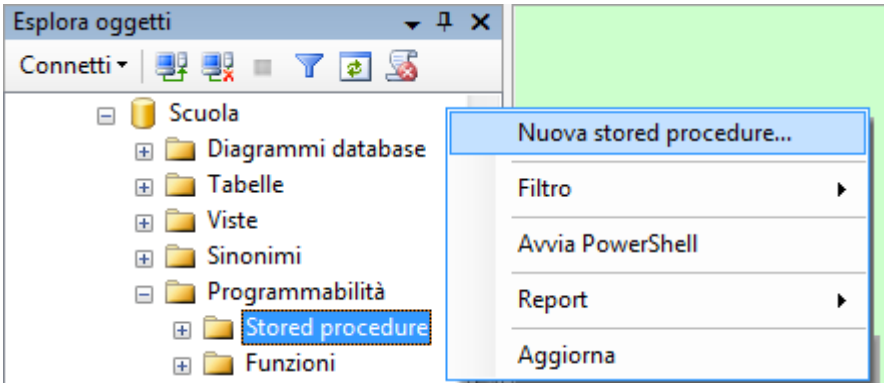
## 21-1 Utilizzo di Esplora Oggetti per creare procedure

Una stored procedure è un insieme di comandi (corpo) T-SQL compilati, che si può invocare specificandone il nome. Il corpo della procedura è eseguita come un blocco unitario di lavoro sul server (server side) e ammette l'uso sia di comandi SQL (SELECT, DELETE, INSERT, UPDATE e altri) e comandi procedurali (IF, WHILE, SET e altri).

Le stored procedure sono simili alle procedure di altri linguaggi di programmazione in quanto sono in grado di:

- Accettare parametri di input e restituire più valori sotto forma di parametri di output alla procedura o al batch che esegue la chiamata.
- Includere istruzioni di programmazione che eseguono le operazioni nel database, tra cui la chiamata di altre procedure.
- Restituire un valore di stato a una procedura o a un batch che esegue la chiamata per indicare l'esito positivo o negativo (e il motivo dell'esito negativo).

Con Microsoft SQL Server è possibile utilizzare Esplora Oggetti per creare funzioni, procedure e trigger.

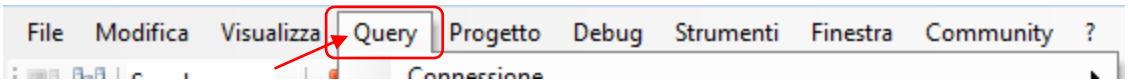



Nella guida in linea di Microsoft sono descritte le procedure per la creazione di una stored procedure utilizzando sia SQL Server Management Studio sia l'istruzione T-SQL CREATE PROCEDURE.

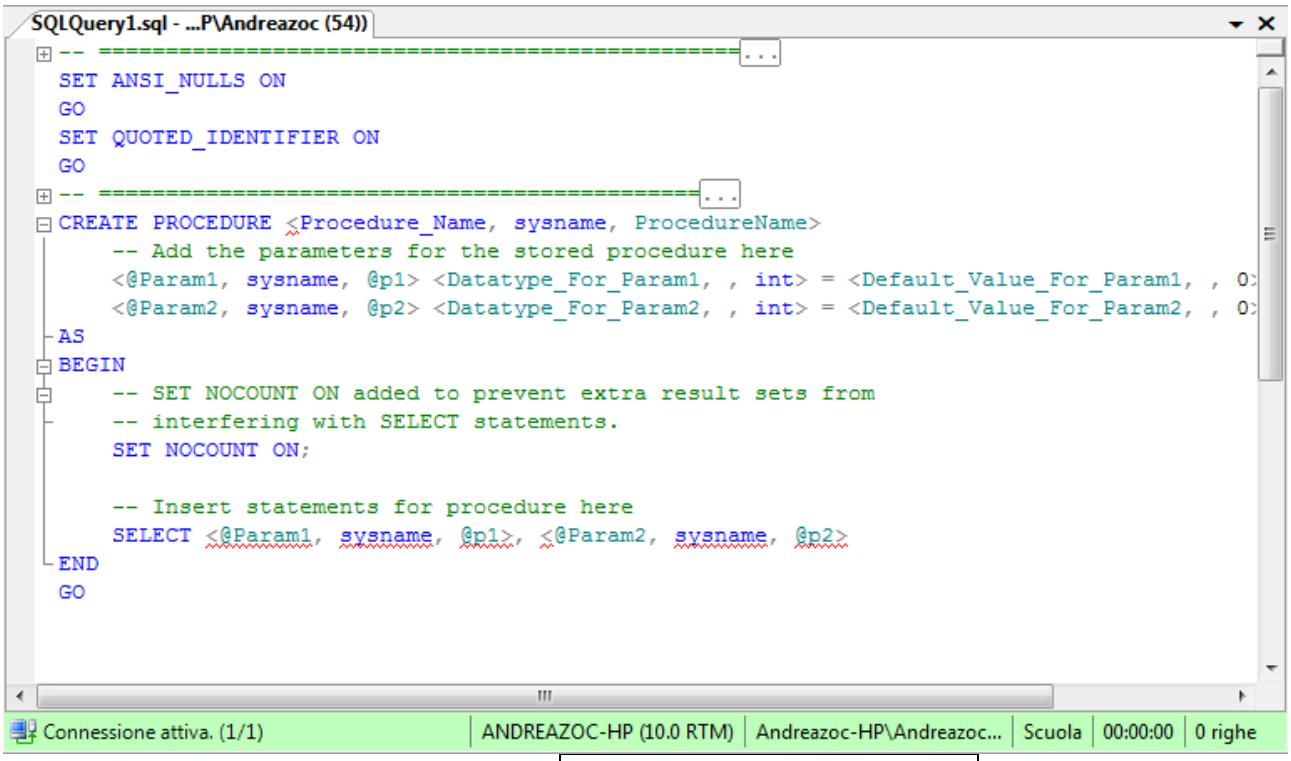
Per creare una procedura è necessario disporre della adeguata autorizzazione per il database e/o dell'autorizzazione ALTER per lo schema in cui la stored procedure viene creata.

Per creare una stored procedure in **Esplora oggetti** si deve evocare il menu contestuale con il pulsante destro del mouse su Stored procedure e quindi scegliere **Nuova stored procedure**.

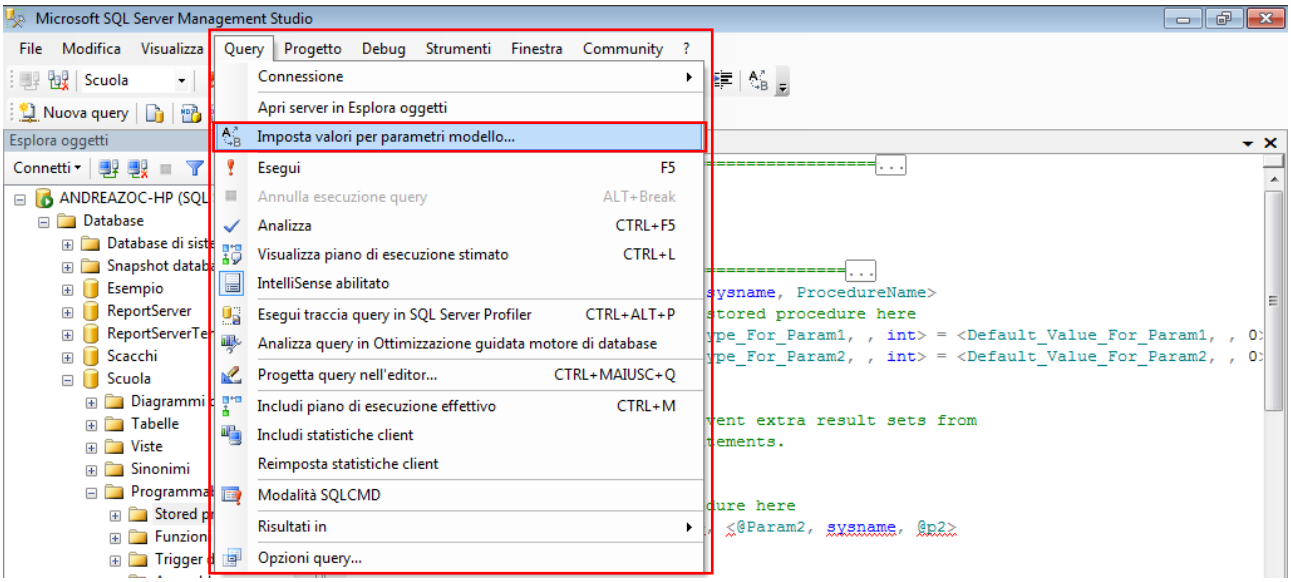
Il menu principale (in alto) si adegua mostrando la voce Query.



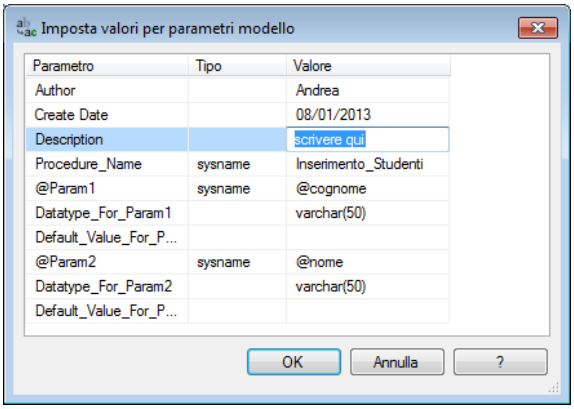

Il riquadro della finestra di progettazione della procedura mostra un codice modello.




Dal menu Query (vedi figura sotto) si può scegliere **Imposta valori per parametri modello** dal menu Query.




Appare una finestra di dialogo Imposta valori per parametri modello immettere i valori seguenti per i parametri indicati.



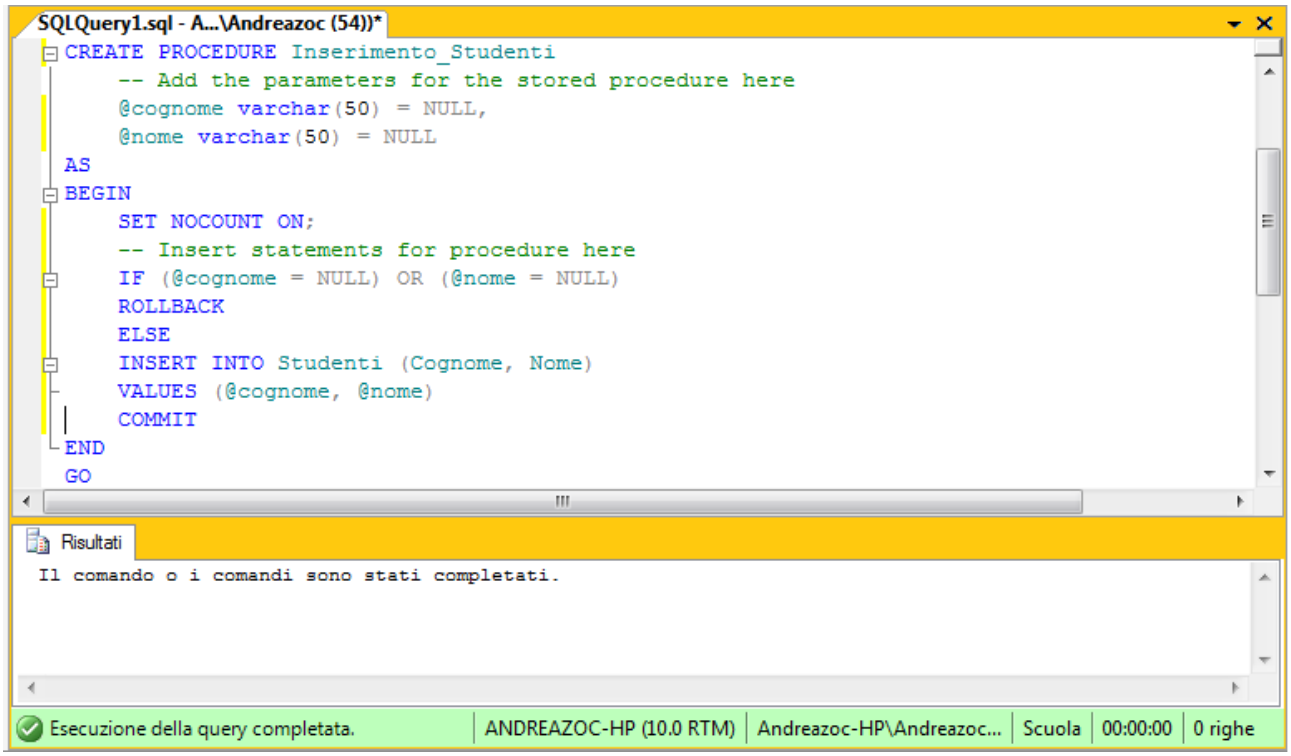

L'autore della query è opzionale, così come la data di creazione e la descrizione della query (sebbene in software house siano richieste per la documentazione e la validazione del codice).

Il nome della procedura è obbligatorio.

I parametri, se necessari, richiedono almeno il nome (che deve sempre iniziare con la chiocciola @) ed il tipo; invece il valore di default non è indispensabile.

Con un clic su OK confermiamo la modifica della query che si adatta nell'apposita finestra. Nell'editor della query è possibile digitare le istruzioni necessarie.

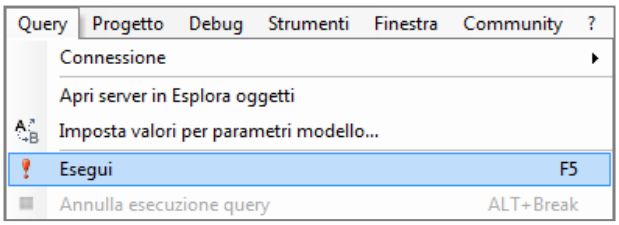
Per controllare la sintassi, scegliere **Analizza** dal menu **Query**. Nel riquadro in basso possono comparire i messaggi di errore, ed occorre saper utilizzare le istruzioni ammesse da T-SQL ed apportare le modifiche necessarie.




Nel nostro esempio scriviamo una procedura come la seguente:

Transact-SQL

```
CREATE PROCEDURE Inserimento_Studenti
   -- Inserisci qui i parametri
   @cognome varchar(50) = NULL,
   @nome varchar(50) = NULL
AS
BEGIN
   SET NOCOUNT ON;
   -- Inserisci qui le istruzioni
   IF (@cognome = NULL) OR (@nome = NULL)
   ROLLBACK
   ELSE
   INSERT INTO Studenti (Cognome, Nome)
   VALUES (@cognome, @nome)
   COMMIT
END
GO
```




Quando si è ben definita la procedura, con un corpo funzionante, è possibile salvare la query creandola definitivamente nel database utilizzando la voce **Esegui** del menu Query. La stored procedure viene creata come un oggetto nel database. La sintassi completa dei comandi utilizzabili nella procedura è rinviata a un capitolo successivo. Per eseguire una stored procedure è possibile scrivere in una finestra di query **uno** dei comandi seguenti:

Transact-SQL

```
EXECUTE Scuola.Inserimento_Studenti @Cognome = N'Carri', @Nome = N'Guido';
EXECUTE Scuola.Inserimento_Studenti N'Carri', N'Guido';
EXEC Scuola.Inserimento_Studenti N'Carri', N'Guido';
```

poi seguito dal comando GO che serve per imporre le eventuali modifiche sul database causate dalla procedura:

Transact-SQL

```
GO
```

## 21-2 Comando CREATE PROCEDURE

È possibile anche creare una stored procedure nell'editor di query, creando una nuova query vuota e scrivendovi il codice manualmente. Nel nostro esempio è possibile scrivere una procedura come la seguente:

Transact-SQL

```
CREATE PROCEDURE Inserimento_Studenti_2
   @cognome varchar(50) = NULL,
   @nome varchar(50) = NULL
AS
BEGIN
   SET NOCOUNT ON;
   IF (@cognome = NULL) OR (@nome = NULL)
   ROLLBACK
   ELSE
   INSERT INTO Studenti (Cognome, Nome)
   VALUES (@cognome, @nome)
   COMMIT
END
GO
```

Ed infine lanciare l'esecuzione con il pulsante Esegui.

## 21-3 Sintassi della CREATE PROCEDURE

**La sintassi si può reperire qui: http://msdn.microsoft.com/en-us/library/ms187926.aspx**

```
--Transact-SQL Stored Procedure Syntax
CREATE { PROC | PROCEDURE } [schema_name.] procedure_name [ ; number ]
    [ { @parameter [ type_schema_name. ] data_type }
        [ VARYING ] [ = default ] [ OUT | OUTPUT ] [READONLY]
    ] [ ,...n ]
[ WITH <procedure_option> [ ,...n ] ]
[ FOR REPLICATION ]
AS { [ BEGIN ] sql_statement [;] [ ...n ] [ END ] }
[;]

<procedure_option> ::=
    [ ENCRYPTION ]
    [ RECOMPILE ]
    [ EXECUTE AS Clause ]
```

**Fonte: http://msdn.microsoft.com/it-it/library/ms187926.aspx**

**schema_name** — Nome dello schema a cui appartiene la procedura. Le procedure sono associate a schema. Se durante la creazione della procedura non viene specificato un nome dello schema, viene assegnato automaticamente lo schema predefinito dell'utente che sta creando la procedura.

**procedure_name** — Nome della procedura. I nomi di procedura devono essere conformi alle regole per gli identificatori e devono essere univoci all'interno dello schema.

Evitare l'utilizzo del prefisso sp_ per la denominazione delle procedure. Questo prefisso viene utilizzato da SQL Server per definire le procedure di sistema. L'utilizzo del prefisso può comportare l'interruzione del codice dell'applicazione, se è presente una procedura di sistema con lo stesso nome.

Le procedure temporanee locali o globali possono essere create utilizzando un simbolo di cancelletto (#) prima del procedure_name (#procedure_name) per le procedure temporanee locali e due simboli di cancelletto per le procedure temporanee globali (##procedure_name). Una procedura temporanea locale è visibile solo alla connessione da cui è stata creata e, alla chiusura di quest'ultima, viene eliminata. Una procedura temporanea globale è disponibile per tutte le connessioni e viene eliminata al termine dell'ultima sessione tramite la procedura. Non è possibile specificare nomi temporanei per le procedure CLR.

Il nome completo di una procedura o di una procedura temporanea globale, inclusi i simboli ##, non deve superare i 128 caratteri. Il nome completo di una procedura temporanea locale, incluso il simbolo #, non deve superare i 116 caratteri.

**; number** Integer facoltativo utilizzato per raggruppare le procedure con lo stesso nome. Tali procedure possono essere eliminate contemporaneamente tramite un'istruzione DROP PROCEDURE.

**@ parameter** Parametro dichiarato nella procedura. Specificare un nome di parametro utilizzando la chiocciola (@) come primo carattere. Il nome del parametro deve essere conforme alle regole per gli identificatori. Poiché i parametri sono locali rispetto alla procedura, è possibile utilizzare gli stessi nomi di parametro in altre procedure.

È possibile specificare uno o più parametri con un limite massimo di 2.100. Il valore di ogni parametro dichiarato deve essere specificato dall'utente quando viene chiamata la procedura, a meno che non venga indicato un valore predefinito per il parametro oppure il valore venga impostato in modo da corrispondere a quello di un altro parametro. Se una procedura contiene parametri con valori di tabella e nella chiamata il parametro non è presente, viene passata una tabella vuota. I parametri possono rappresentare solo espressioni costanti, non nomi di tabella, nomi di colonna o nomi di altri oggetti di database. Per ulteriori informazioni, vedere EXECUTE (Transact-SQL).

Se viene specificata l'opzione FOR REPLICATION, non è possibile dichiarare alcun parametro.

**data_type** Tipo di dati del parametro e schema a cui appartiene il tipo di dati.

Tutti i tipi di dati Transact-SQL possono essere utilizzati come parametri.

Per creare parametri con valori di tabella è possibile utilizzare il tipo di tabella definito dall'utente. I parametri con valori di tabella possono essere solo parametri di input e insieme a essi deve essere associata la parola chiave READONLY. Per ulteriori informazioni, vedere Utilizzare parametri con valori di tabella (Motore di database)

I tipi di parametri cursor possono essere solo parametri di output e insieme a essi deve essere associata la parola chiave VARYING.

## 21-4 Suggerimenti sulle Stored Procedure

Microsoft propone dei suggerimenti che possono migliorare le prestazioni della procedura.

L'uso dell'istruzione SET NOCOUNT ON come prima istruzione nel corpo della procedura (subito sotto subito dopo la parola chiave AS)permette di disabilitare i messaggi restituiti al client da SQL Server dopo l'esecuzione delle istruzioni SELECT, INSERT, UPDATE, MERGE e DELETE. Questo migliora anche le prestazioni generali del database e di sovraccarico di rete (overhead di rete).

Utilizzare i nomi degli schemi per gli oggetti (anche i nuovi tipi di dato); questo richiede un minor tempo di elaborazione se si lavora su un unico schema. Inoltre permette di evitare problemi di autorizzazioni e di accesso allo schema.

Evitare l'utilizzo di funzioni scalari nelle istruzioni SELECT che restituiscono molte righe di dati.

Evitare l'utilizzo di SELECT * e specificare i nomi delle colonne. In questo modo è possibile evitare alcuni errori del Motore di database che possono causare l'arresto dell'esecuzione della procedura.

Evitare l'elaborazione o la restituzione di troppi dati. Se possibile, restringere i risultati (anche intermedi) in modo che le operazioni effettuate dalla procedura siano eseguite col set di dati più piccolo possibile.

Inviare al client solo i dati essenziali. L'operazione migliora le prestazioni di rete, nonché l'elaborazione dal lato del client.

Utilizzare le transazioni esplicite racchiuse nei comandi BEGIN/END TRANSACTION nel modo più breve possibile. Se si usano transazioni lunghe si causa un ampio blocco dei record e un rischio maggiore di stallo di concorrenza.

Utilizzare la funzionalità TRY…CATCH per la gestione degli errori in una procedura che consente di rendere più snelle le istruzioni del blocco. In questo modo si garantisce un minor sovraccarico di elaborazione e una segnalazione errori più precisa.

Utilizzare la parola chiave DEFAULT in tutte le colonne della tabella a cui viene fatto riferimento dalle istruzioni CREATE TABLE o ALTER TABLE presenti nel corpo della procedura, per evitare di passare NULL a campi obbligatori causando un errore.

Utilizzare istruzioni di UPDATE per convertire valori Null ed eliminare le righe con valori Null dalle query.

### 21-5 Osservazioni generali sulle procedure

Non è prevista una dimensione massima predefinita per una procedura.

Alla prima esecuzione, la procedura viene compilata in modo da determinare un piano di accesso ottimale per il recupero dei dati. Se il piano generato rimane archiviato nell'apposita cache del Motore di database, può essere riutilizzato nelle successive esecuzioni della procedura.

È possibile far eseguire automaticamente una o più procedure all'avvio di SQL Server. Le procedure devono essere create dall'amministratore del sistema nel database master ed eseguite dal ruolo predefinito del server (**sysadmin**) come processo in background. In queste procedure non è possibile utilizzare parametri di input o output.

Le procedure si ritengono nidificate quando una procedura sfrutta la chiamata di un'altra procedura o causa l'esecuzione di codice. È possibile nidificare fino a 32 livelli di procedure e riferimenti a codice gestito. Il tentativo di superare il livello di nidificazione massimo causa l'esito negativo dell'intera catena di chiamata. Per restituire il livello di nidificazione dell'esecuzione della stored procedure corrente è possibile utilizzare la funzione @@NESTLEVEL.

La creazione di una procedura può utilizzare tabelle che non esistono ancora. In fase di creazione si compie solo un controllo di sintassi ma la procedura non sarà compilata fino alla prima esecuzione. Alla prima esecuzione la procedura sarà compilata e si devono risolvere tutti i riferimenti agli oggetti. Se in fase di esecuzione le tabelle a cui si fa riferimento non esistono ancora, la procedura si arresterà con esito negativo.

Non è possibile specificare un nome di funzione come valore predefinito di un parametro o come valore passato a un parametro durante l'esecuzione di una procedura. Tuttavia, è possibile passare una funzione come variabile.

# 22 Funzioni definite dall'utente

### 22-1 Utilizzo di Esplora Oggetti per creare funzioni

Una funzione definita dall'utente è una routine Transact-SQL o Common Language Runtime (CLR) tramite cui vengono accettati parametri, viene effettuata un'azione, ad esempio un calcolo complesso, e viene restituito il risultato di tale azione sotto forma di valore. Il valore restituito può essere un valore scalare (singolo) o una tabella. A seconda del tipo restituito le funzioni possono essere di due tipi: scalari o tabellari.

È opportuno osservare che anche le funzioni possono prevedere parametri, analogamente alle procedure, e che questi sono modificabili nel corpo della funzione salvo se nella loro dichiarazione, subito dopo il tipo di dati, non sia specificato il vincolo READONLY.

A differenza delle procedure, prima del corpo è prevista l'istruzione RETURNS il cui scopo è definire il tipo restituito dalla funzione (un tipo di dati oppure TABEL). L'istruzione RETURNS impone i valori che il comando RETURN può rendere. Il comando RETURN sarà discusso in seguito.

Le funzioni scalari restituiscono un singolo valore e la relativa sintassi è la seguente:

Transact-SQL

```
--Transact-SQL Scalar Function Syntax
CREATE FUNCTION [ schema_name. ] function_name
( [ { @parameter_name [ AS ][ type_schema_name. ] parameter_data_type
    [ = default ] [ READONLY ] }
    [ ,...n ]
  ]
)
RETURNS return_data_type
    [ WITH <function_option> [ ,...n ] ]
    [ AS ]
    BEGIN
        function_body
        RETURN scalar_expression
    END
[ ; ]
```

Le funzioni tabellari restituiscono una tabella e la relativa sintassi è la seguente:

Transact-SQL

```
--Transact-SQL Inline Table-Valued Function Syntax
CREATE FUNCTION [ schema_name. ] function_name
( [ { @parameter_name [ AS ] [ type_schema_name. ] parameter_data_type
    [ = default ] [ READONLY ] }
    [ ,...n ]
  ]
)
RETURNS TABLE
    [ WITH <function_option> [ ,...n ] ]
    [ AS ]
    RETURN [ ( ] select_stmt [ ) ]
[ ; ]
```

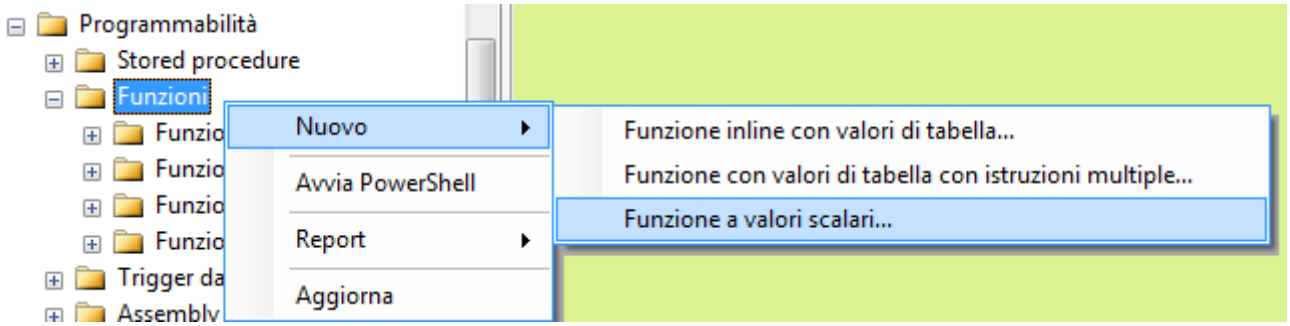
Una funzione somiglia molto ad una procedura ed anch'essa è un insieme di comandi (corpo) T-SQL compilati, che si può invocare specificandone il nome. Il corpo della funzione è eseguita come un blocco unitario di lavoro sul server (server side) e ammette l'uso sia di comandi SQL (SELECT, DELETE, INSERT, UPDATE e altri) e comandi procedurali (IF, WHILE, SET e altri).

Le funzioni sono in grado di:

- Accettare parametri di input e restituire più valori sotto forma di parametri di output alla procedura o al batch che esegue la chiamata.
- Includere istruzioni di programmazione che eseguono le operazioni nel database, tra cui la chiamata di altre procedure.
- Restituire un valore tipizzato o tabellare al chiamante (eventualmente batch) oltre a restituire un valore scalare per l'esito di successo o di fallimento (e il motivo del fallimento).

## 22-2 Utilizzo di Esplora Oggetti per creare funzioni

Con SSMS è possibile utilizzare Esplora Oggetti per creare le funzioni.

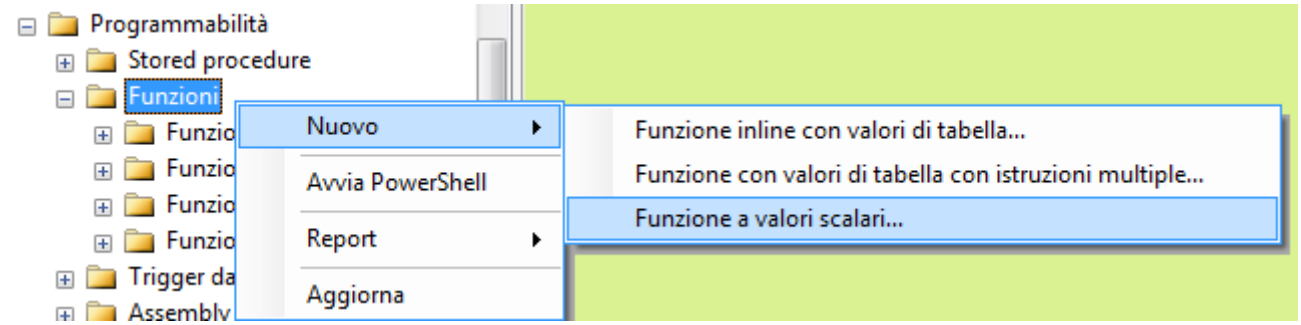



Le funzioni definite dall'utente restituiscono sempre un valore. A differenza delle procedure, il menu contestuale per le funzioni propone diverse funzioni alternative distinte in base al tipo di valore restituito, e ogni funzione deve rientrare in una delle tre seguenti categorie:

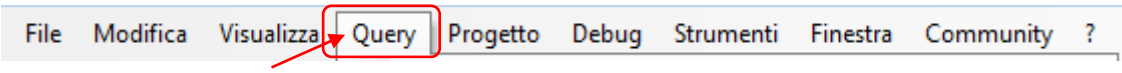
- **Funzione inline** – È una funzione che restituisce un dynaset (una tabella dinamica). Questo tipo di funzioni contiene una sola istruzione SELECT e la tabella resa è aggiornabile ovvero potrà essere aggiornato anche il risultato in formato tabellare restituito dalla funzione. Tali funzioni sono chiamate funzioni inline. È possibile utilizzare una funzione inline nella clausola FROM di un'altra query.
- **Funzione con valori di tabella con istruzioni multiple** - È una funzione che restituisce un dynaset (una tabella dinamica), calcolata mediante più di una query o usa query non aggiornabili. In questo caso non sarà possibile aggiornare il risultato in formato tabella restituito dalla funzione. Anche questa funzione può essere utilizzata nella clausola FROM di un'altra query.
- **Funzione con valori scalari –** È un funzione definita dall'utente che rende un valore scalare (un valore elementare come un numero, una stringa, ecc). Se una funzione restituisce un valore scalare è possibile utilizzarla in una query in qualsiasi punto si desideri utilizzare un nome di colonna.

## 22-3 Funzione con valori scalari

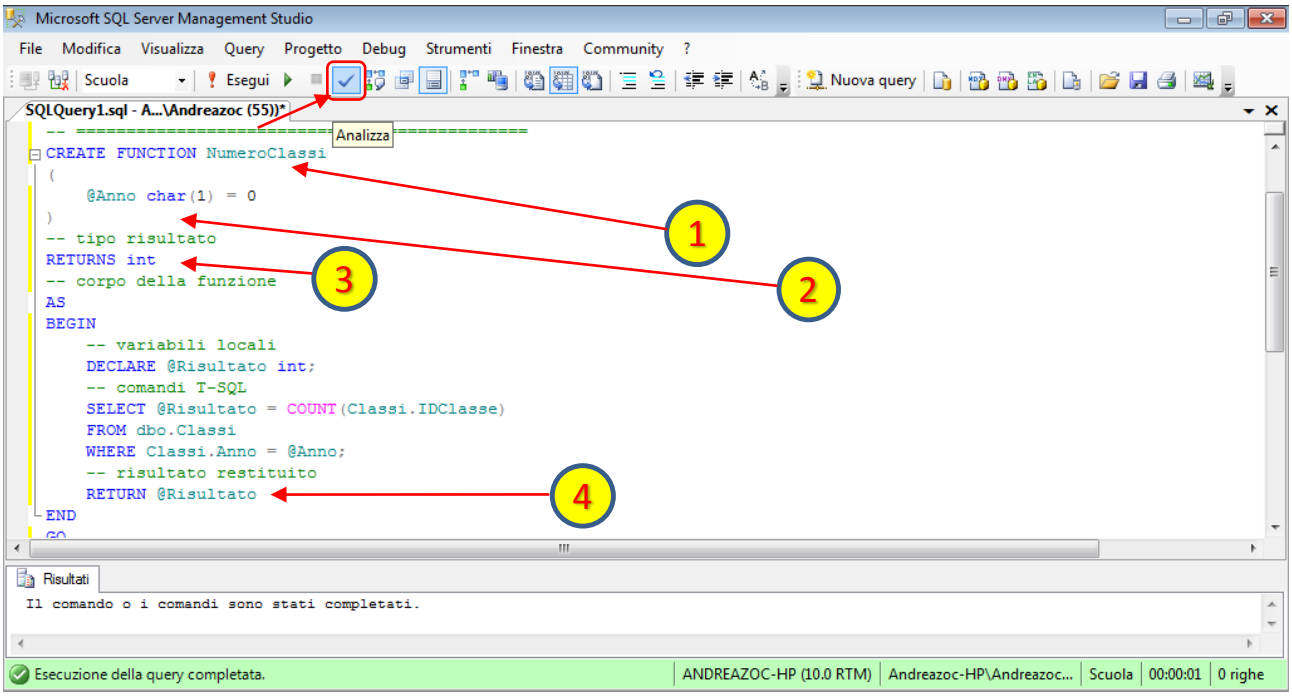
Per creare una funzione con valori scalari in **Esplora oggetti** si deve evocare il menu contestuale con il pulsante destro del mouse su Funzioni e quindi scegliere **Nuova funzione**.




Il menu principale (in alto) si adegua mostrando la voce Query.

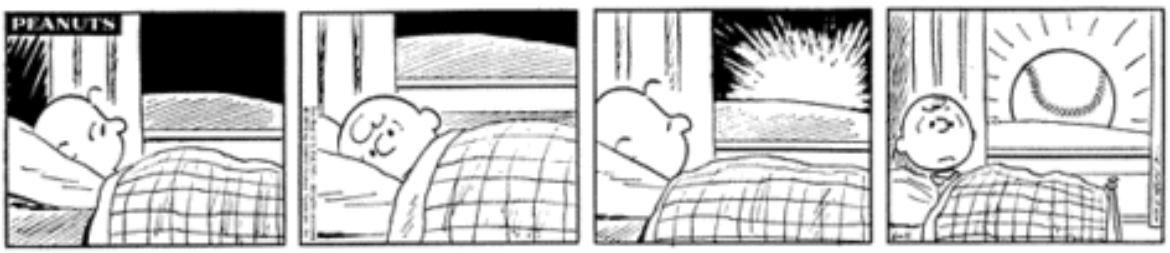



Il riquadro della finestra di progettazione della procedura mostra un codice modello.



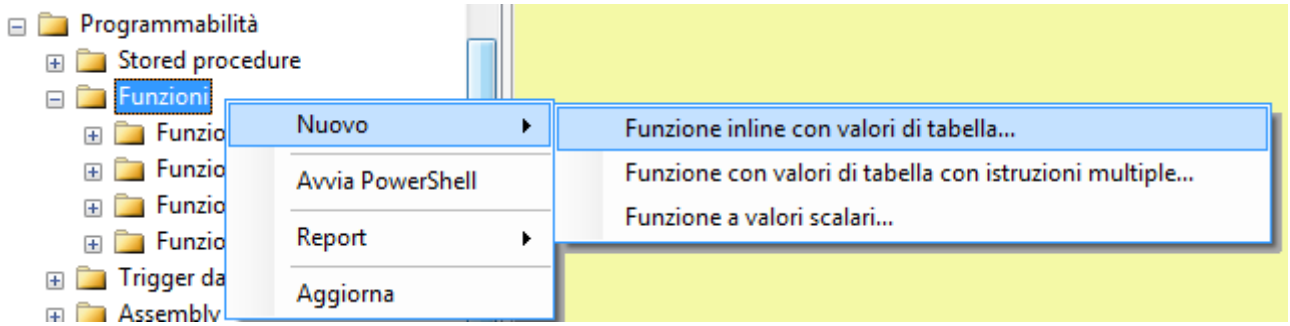

Analogamente alle funzioni è possibile richiedere di analizzare il codice sia col menu contestuale sula query che dalla barra Editor SQL (nella figura è evidenziato il pulsante Analizza)

Le funzioni scalari definite dall'utente restituiscono un singolo valore di dati (ad esempio un valore string, integer o bit) del tipo dichiarato nella clausola RETURNS. Per una funzione scalare inline, non è disponibile alcun corpo della funzione. Il valore scalare corrisponde al risultato di una singola istruzione. Per una funzione scalare con istruzioni multiple, il corpo della funzione, definito in un blocco BEGIN...END, include una serie di istruzioni Transact-SQL che restituiscono un solo valore. Il tipo restituito può essere qualsiasi tipo di dati ad eccezione di text, ntext, image, cursor e timestamp.

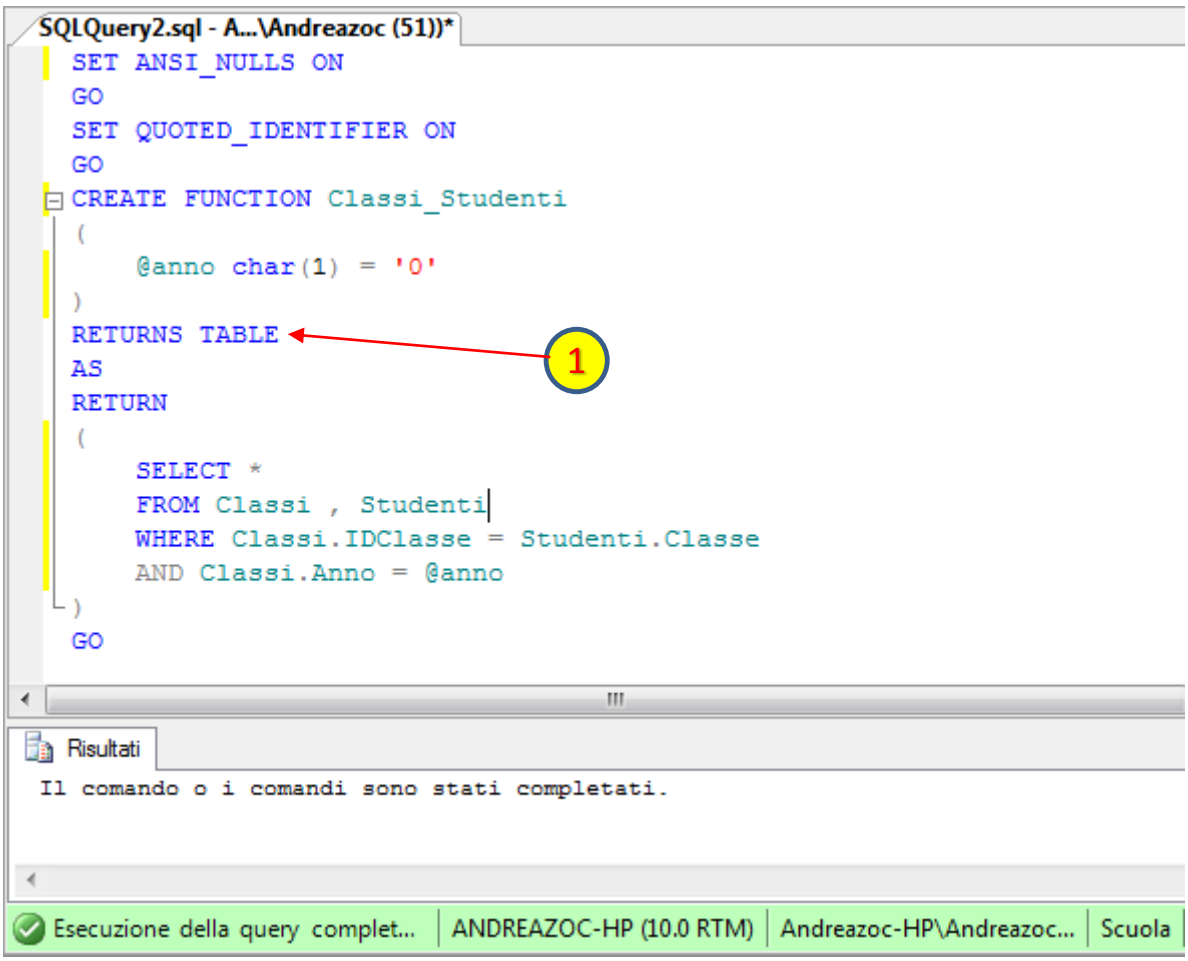


## 22-4 Funzioni con valori di tabella

Le funzioni con valori di tabella definite dall'utente restituiscono un tipo di dati **table**. Per una funzione inline con valori di tabella non è disponibile alcun corpo della funzione. La tabella corrisponde al set di risultati di una singola istruzione SELECT.



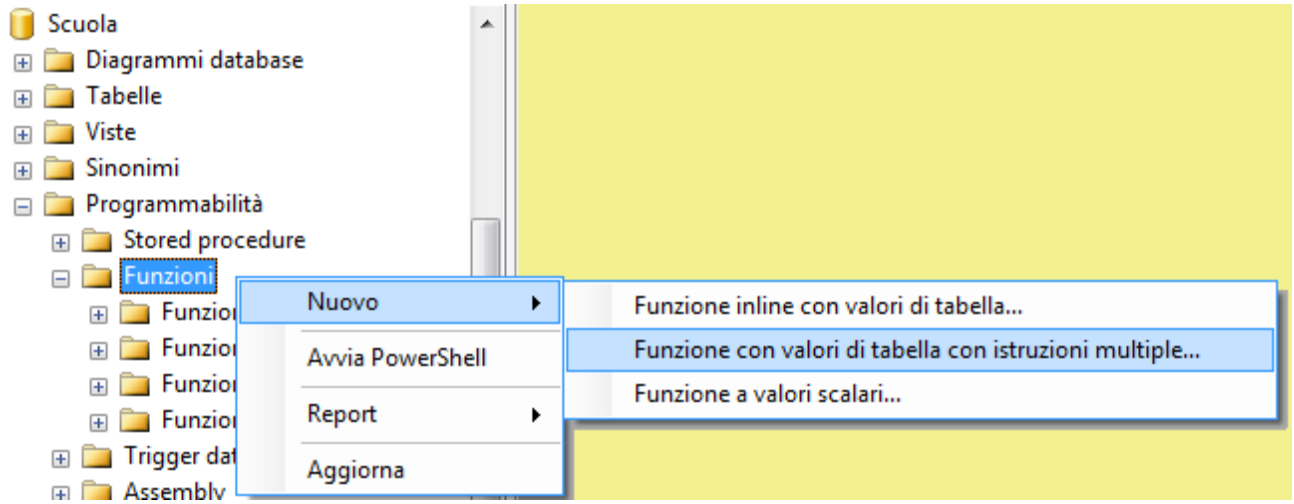

Per creare una funzione con valori di tabella in **Esplora oggetti** si deve evocare il menu contestuale con il pulsante destro del mouse su Funzioni, scegliere **Nuovo** e scegliere la voce **Funzione inline con valori di tabella**.




Nell'esempio si è costruita una funzione che restituisce una tabella congiungendo le tabelle Classi e Studenti, con una INNERJOIN e filtrando i record in base al parametro passato alla funzione. Il valore zero assegnato di default al parametro comporta che nessun record sarà restituito se si passa un valore nullo alla funzione.

## 22-5 Funzioni con valori di tabella con istruzioni multiple

Per una funzione con valori di tabella con istruzioni multiple, il corpo della funzione, definito in un blocco BEGIN...END, include una serie di istruzioni Transact-SQL che compilano e inseriscono righe nella tabella che verrà restituita. La creazione della funzione è del tutto analoga alle precedenti, come illustrato nella figura sottostante.




## 22-6 Comando CREATE FUNCTION

Il comando CREATE FUNZION consente di creare una funzione definita dall'utente. Una funzione è una routine T-SQL con cui (anche con parametri), viene compiuta un'attività (spesso un calcolo articolato), e che rende il risultato dell'attività mediante un valore. Il valore restituito può essere un valore scalare (singolo) o una tabella. Si usa per creare una funzione riutilizzabile. La funzione si può utilizzare per:

- Calcolare un valore da utilizzare in una query, ad esempio in una SELECT.
- In frammenti di codice T-SQL che invocano la funzione.
- Nel corpo di un'altra funzione definita dall'utente.
- Parametrizzare una vista o per strutturare meglio una vista indicizzata.
- In una CREATE TABLE (o ALTER TABLE) per definire una colonna di una tabella.
- Definire un vincolo CHECK su una colonna.
- Sostituire una stored procedure.

## 22-7 Esempio di funzione

È possibile anche creare una funzione nell'editor di query, creando una nuova query vuota e scrivendovi il codice manualmente. Nel nostro esempio è possibile scrivere una procedura come la seguente:

Transact-SQL

```
CREATE FUNCTION Numero_Studenti_Maggiorenni (
   @anno char(1) = '0',
)
RETURNS int
AS
BEGIN
   SET NOCOUNT ON;
   DECLARE @ris int;
   SELECT @ris = COUNT(*)
   FROM Studenti , Classi
   WHERE Studenti.Classe = Classi.IDClasse
      AND = Classi.Anno = @anno;
   RETURN @ris;
END
GO
```

Infine lanciare l'esecuzione con il pulsante Esegui.

### 22-8 Richiamare una funzione

Per chiamare tale procedura tramite il Query Editor basta scrivere il comando EXEC seguito dal nome della funzione e dagli eventuali parametri:

Transact-SQL
```
EXECUTE Classi_Studenti
   @anno = N'5'
GO
```

Transact-SQL
```
EXEC Classi_Studenti
   @anno = N'5'
GO
```

### 22-9 Commenti sulle funzioni

Una SP non può essere non inclusa in un'istruzione di SELECT, essa semplicemente completa il proprio lavoro e termina la propria esecuzione. Questo è proprio uno scenario in cui le funzioni sono molto utili poiché esse sono simili alle SP ma è possibile utilizzarle all'interno di una query per sfruttare i dati che esse restituiscono.

Notate che anche le funzioni possono richiedere parametri, modificabili all'interno delle stesse se nella loro definizione dopo il tipo di dati non viene messo il valore READONLY. Di diverso rispetto alle SP c'è l'istruzione RETURNS che serve a specificare il tipo di dati del valore scalare che la funzione restituirà e l'istruzione RETURN che fa sì che la funzione restituisca tale valore.

# 23 Trigger

### 23-1 Descrizione del Trigger

Può accadere che, specie nei database di grandi dimensioni, la modifica di alcuni elementi (strutture o dati) debba implicare un'azione automatica su altri dati. I trigger sono gli strumenti più opportuni per realizzare queste azioni. Un trigger può essere definito come una procedura automatica (un insieme di comandi T-SQL) che risponde a particolari requisiti:

- Il trigger ha lo scopo di verificare il rispetto di regole di integrità dei dati, per conservare la loro congruenza;
- Il trigger può anche agire sui dati, inserendo , modificando e eliminando dati per conservare la loro congruenza;
- Un trigger viene innescato automaticamente e non viene invocato come accade per procedure e funzioni.

Un trigger in SQL server può essere di due tipi a seconda dell'ambito di innesco:

- Un TRIGGER DML (Data Modification Language) è eseguito in caso di modifica dei dati contenuti nelle strutture;
- Un TRIGGER DDL (Data Definition Language) è eseguito in caso di modifica delle strutture;

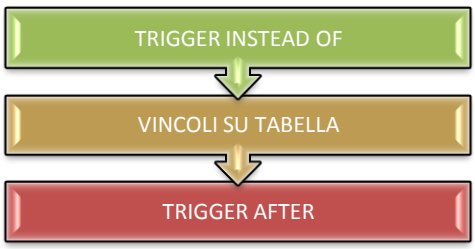
Un trigger DML è innescato quando si verifica un'operazione tipo DELETE, INSERT, UPDATE; un trigger DDL è innescato quando si esegue un'azione tipo CREATE oppure ALTER (TABLE, INDEX, VIEW, PROCEDURE, FUNCTION).

### 23-2 Trigger DML

I trigger DML possono essere utilizzati per diversi scopi. Un impiego alquanto consueto è riferito a controlli di validazione sui dati per garantire la congruenza e la loro integrità. Se il controllo è riferito a una sola colonna allora conviene usare un vincolo CHECK che risulta più compatto e rapido. Inoltre altri tipi di controllo, come i vincoli di integrità referenziale dichiarativa, sono realizzati direttamente nei comandi ALTER TABLE e CREATE TABLE.

I trigger invece costituiscono la scelta migliore quando le validazioni da verificare sono articolate e su più dati sparsi nel database. I trigger DML sono utilizzati di frequente per applicare regole specifiche dell'organizzazione e di integrità dei dati.

Se ci sono vincoli (come CHECK o FOREIGN KEY) nella tabella allora i vincoli sono controllati 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 attivato.



I trigger DML possono essere di nove tipi, a seconda della operazione DML a cui sono associati e del momento in cui si innescano:

| TRIGGER | DELETE | INSERT | UPDATE |
|---|---|---|---|
| FOR | 1 | 2 | 3 |
| AFTER | 4 | 5 | 6 |
| INSTEAD OF | 7 | 8 | 9 |

E' possibile però definire un trigger associato a più di un'operazione DML (combinazione).

## 23-3 Trigger DDL

I trigger DDL sono usati invece per agire in reazione a modifiche sulle strutture dei dati. Per esempio modifiche su colonne di una tabella o di una vista, modifiche su indici, funzioni o procedure possono innescare controlli di validazione relativi all'intero database.

T-SQL prevede anche dei trigger di tipo LOGON che sono associati all'evento di connessione di un cliente (LOGON) che si solleva quando un utente stabilisce una nuova sessione.

## 23-4 Sintassi del trigger

Trigger on an INSERT, UPDATE, or DELETE statement to a table or view (DML Trigger)

Transact-SQL

```
CREATE TRIGGER [ schema_name . ]trigger_name
ON { table | view }
[ WITH <dml_trigger_option> [ ,...n ] ]
{ FOR | AFTER | INSTEAD OF }
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
[ NOT FOR REPLICATION ]
AS { sql_statement  [ ; ] [ ,...n ] | EXTERNAL NAME <method specifier [ ; ] > }
```

A parte gli elementi che è facile comprendere vorrei soffermarmi su alcuni elementi di tale sintassi.

Il nome del trigger e la struttura a cui è associata è definita nelle prime righe della CREATE TRIGGER; occorre osservare che si può agganciare un trigger ad una tabella ma anche a una vista aggiornabile.

Transact-SQL

```
CREATE TRIGGER [ schema_name . ]trigger_name
ON { table | view }
```

Transact-SQL

```
CREATE TRIGGER Scuola.VerificaDati
ON miaVista
```

Il momento in cui dover eseguire il trigger ed il comando DML sono definiti nella seconda sezione sintattica:

Transact-SQL

```
{ FOR | AFTER | INSTEAD OF }
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
```

Transact-SQL

```
FOR
INSERT , UPDATE
```

Le tre opzioni FOR, AFTER e INSTEAD OF sono le tre possibilità che specificano quando il trigger deve scattare:

- L'opzione FOR indica un'azione che deve essere contemporanea all'aggiornamento dei dati della tabella;
- L'opzione AFTER indica un'azione immediatamente successiva all'aggiornamento dei dati della tabella;
- l'opzione INSTEAD OF è la più complessa da comprendere e indica che deve essere eseguita l'istruzione del trigger al posto dell'istruzione T-SQL predefinita specificata dopo (INSERT, DELETE o UPDATE).

SQL Server utilizza due tabelle logiche di sistema, DELETED e INSERTED, per rappresentare una tabella in via di modifica.

Quando si inserisce un record in una tabella, una copia della riga è anche inserita nella tabella INSERTED.

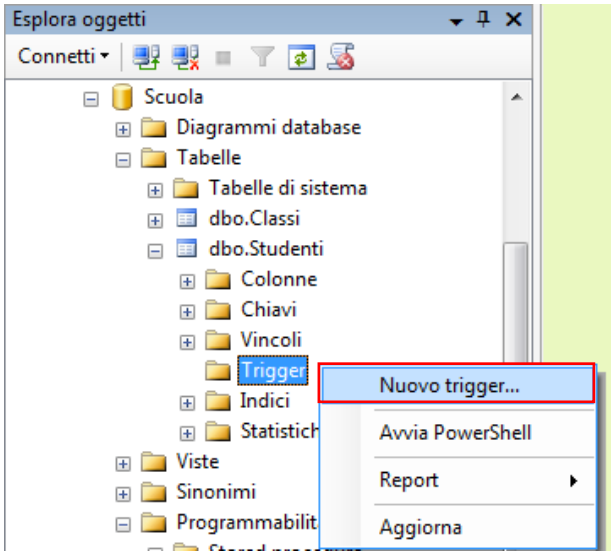
Quando si cancella una riga da una tabella, una copia della riga è anche inserita nella tabella DELETED.

Quando si modifica una riga esistente, una copia della vecchia riga è inserita in DELETED , e una copia della riga è inserita in INSERTED .

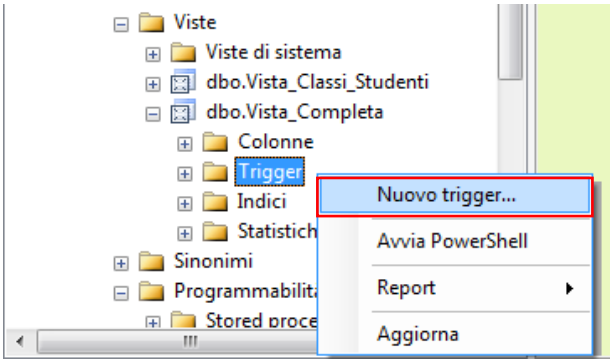
Queste tabelle sono utili per effettuare controlli di validazione dei dati per poter gestire correttamente le istruzioni del trigger.

## 23-5 Utilizzo di Esplora Oggetti per creare Trigger

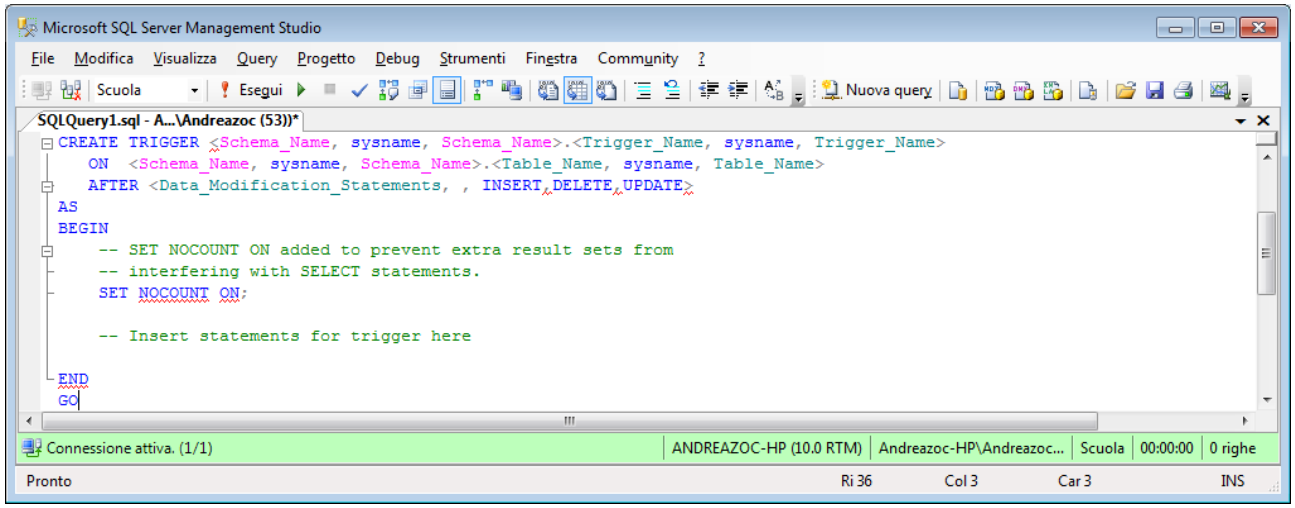
Per creare un trigger in modalità visuale con SSMS si deve espandere il nodo della tabella a cui associare il trigger, selezionare il nodo Trigger ed evocare il menu contestuale con il tasto destro del mouse e infine scegliere l'opzione **Nuovo Trigger ...** .



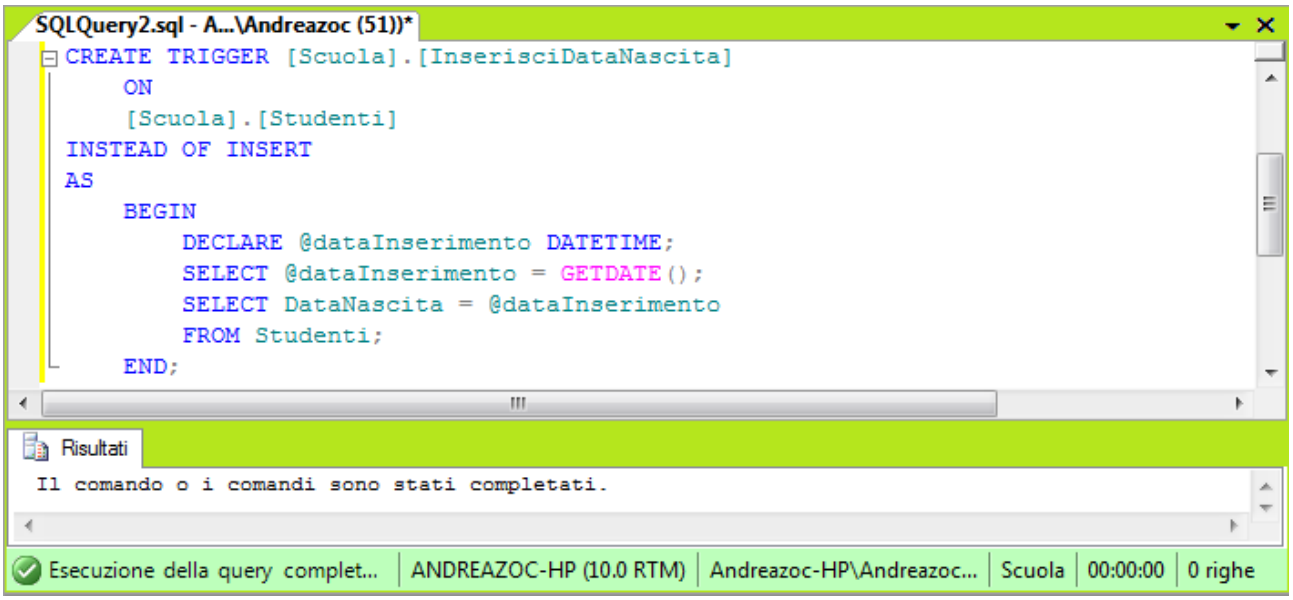

Analogamente è possibile creare un trigger per una vista espandendo il nodo della vista prescelta, selezionando il nodo Trigger, evocare il menu contestuale col tasto destro del mouse ed infine scegliere l'opzione **Nuovo Trigger ...** .




Il sistema genera un modello di codice T-SQL per il trigger che dovrà essere modificato per i propri scopi. Un esempio di template è proposto nella figura proposta qui di seguito:




Quando si è modificato il codice TSQL è possibile verificare la sintassi con Analizza (come per le query):




Per creare definitivamente il trigger fare clic su Esegui dalla barra SQL oppure da menu contestuale:


## 23-6 Esempi di Trigger

Alcuni esempi di trigger possono essere i seguenti:

Transact-SQL

```
CREATE TRIGGER Scuola.[VerificaDataNascita]
    ON
    Scuola.Studenti
INSTEAD OF INSERT
AS
    BEGIN
        DECLARE @dataNascita DATETIME;
        -- query
        SELECT @dataNascita = S.DataNascita
        FROM Scuola.Studenti AS S;
        -- controllo
        IF @dataNascita >= GETDATE()
            RAISERROR('Errore nella data di nascita.', 10, 1);
            ROLLBACK;
    END
```

Transact-SQL

```
CREATE TRIGGER [Scuola].[InserisciDataNascita]
    ON
    [Scuola].[Studenti]
INSTEAD OF INSERT
AS
    BEGIN
        DECLARE @dataInserimento DATETIME;
        SELECT @dataInserimento = GETDATE();
        SELECT DataNascita = @dataInserimento

    END;
```

Transact-SQL

```
CREATE TRIGGER TassaBase
    ON
    [dbo].[Studenti]
AFTER INSERT
```

```
AS
    BEGIN
        DECLARE @TassaBase MONEY;
        SET @TassaBase = 50.00;
        SELECT Tasse = @TassaBase
        FROM INSERTED;
    END;
```

Transact-SQL

```
CODICE DI UN TRIGGER DA DEFINIRE
```

# 24 Tipi personalizzati

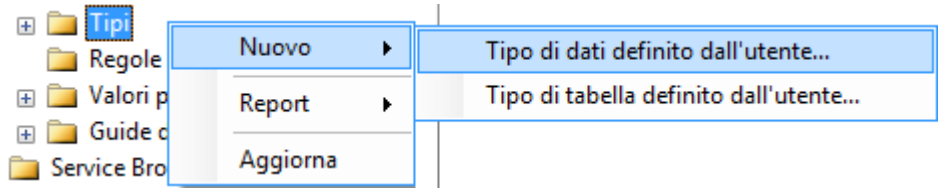
## 24-1 Definizione di tipi da Esplora Oggetti

Sebbene SQL server metta a disposizione del programmatore un ampio numero di tipi di dato, è possibile che il programmatore desideri definire un nuovo tipo di dato, specifico per l'organizzazione che si intende gestire. In alcuni casi è preferibile dichiarare dei vincoli su una colonna piuttosto che creare un tipo di dati nuovo. Comunque la definizione di tipi da parte dell'utente è già presente nello standard ANSI-92.

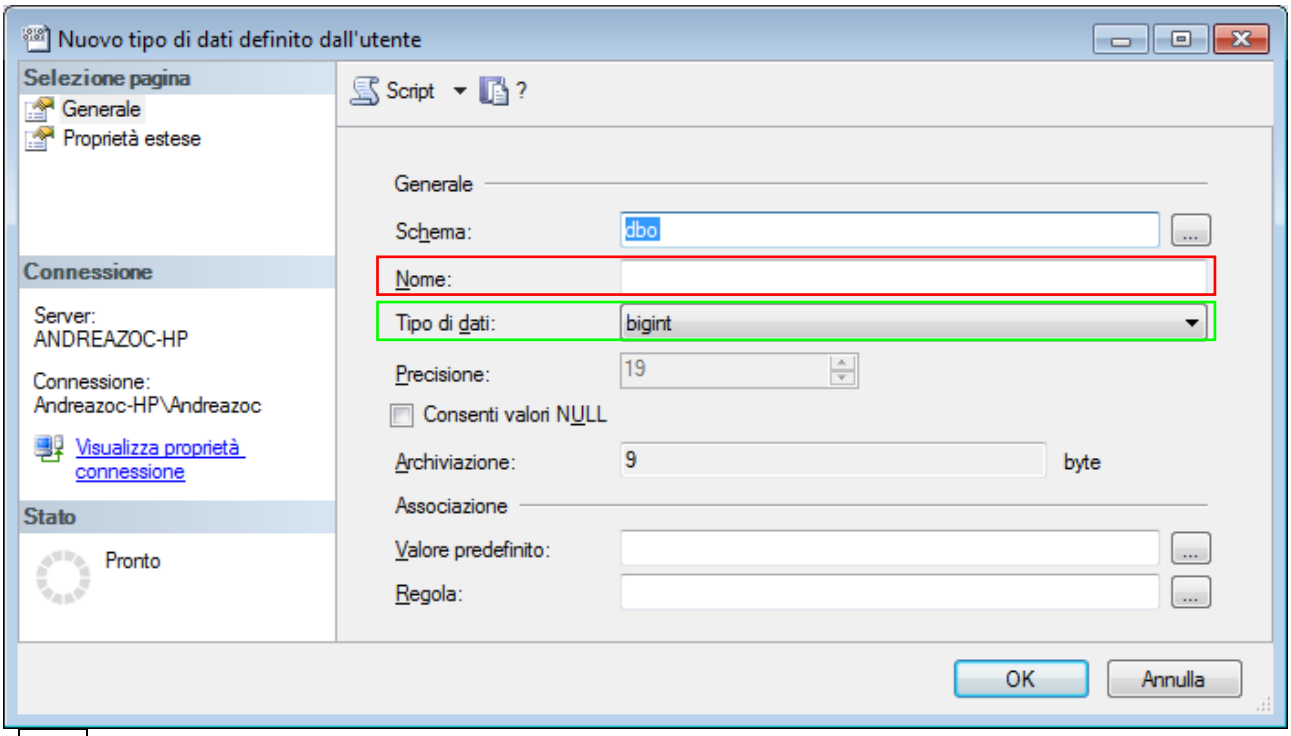
SQL server propone due diverse categorie di tipi personalizzati:

- Tipo di dati scalare
- Tipo di dati tabellare

Per definire un nuovo tipo di dati con l'interfaccia visuale si deve espandere l'albero dell'Esplora Oggetti e selezionare il nodo Tipi; col clic destro si evoca il menu contestuale, scegliere **Nuovo** e selezionare la voce desiderata.




La scelta fa comparire una finestra di dialogo che chiede le informazioni di base del tipo da costruire:




Il **Nome** serve per l'individuazione del nuovo tipo di dato.

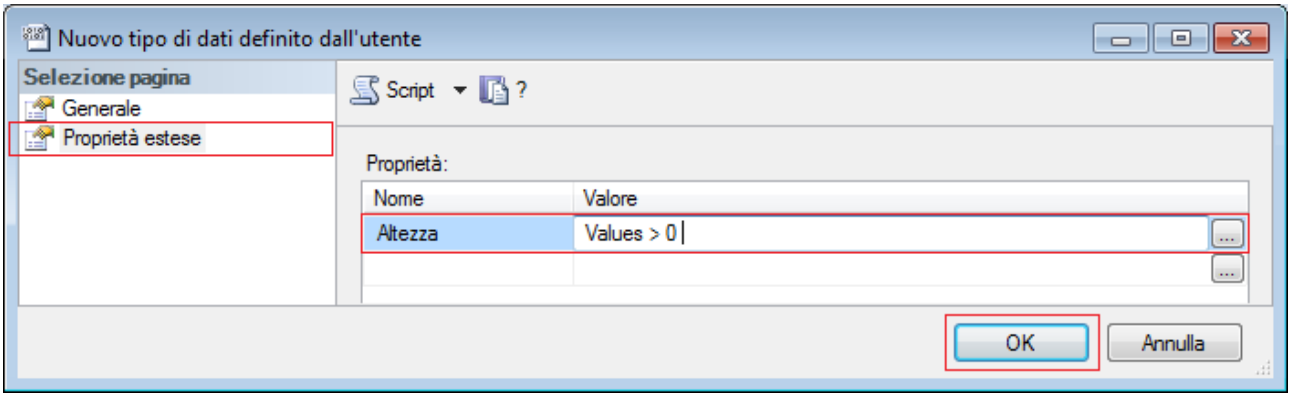
Il **Tipo di Dati** indica il tipo base su cui costruire il nuovo tipo di dato.

La casella di spunta **Consenti valori nulli** determina se il nuovo tipo ammette o meno valori NULL.

Come caso di esempio, si definirà un nuovo tipo con Nome Altezza (digitare Altezza nella casella del Nome), il cui tipo base sarà smallint (precisione 5 byte).

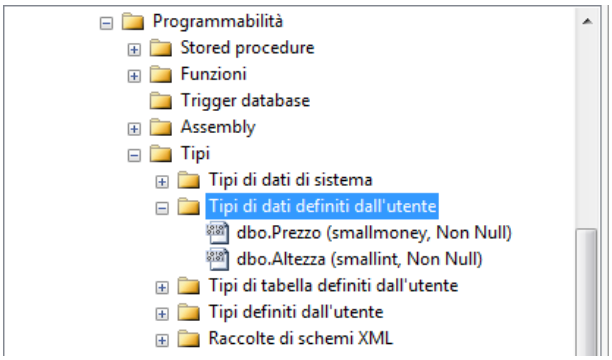
Nell'area Associazione è possibile compilare le caselle **Valore predefinito** o **Regola** per associare un valore predefinito o una regola al nuovo tipo di dati. In SQL Server Management Studio non è possibile creare valori predefiniti e regole.

L'opzione Proprietà Estese, che compare sulla sinistra della finestra, consente di accedere a validazioni del tipo.




Con OK si conferma la creazione del tipo.

È possibile verificare la presenza dei nuovi tipi espandendo il nodo opportuno, come illustrato nella figura seguente:




## 24-2 Considerazioni sui tipi personalizzati

È possibile consultare la guida in linea di MS sui tipi personalizzati al seguente indirizzo:

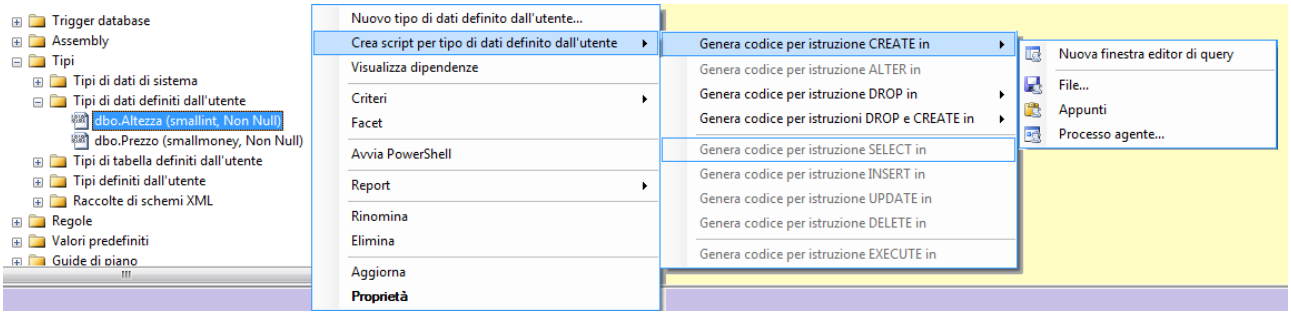
http://technet.microsoft.com/it-it/library/ms190232(v=sql.100).aspx

Anche per i tipi di dato il procedimento visuale corrisponde in realtà a un comando DDL: CREATE TYPE

Transact-SQL

```
CREATE TYPE [ schema_name. ] type_name
{
    FROM base_type
    [ ( precision [ , scale ] ) ]
    [ NULL | NOT NULL ]
  | EXTERNAL NAME assembly_name [ .class_name ]
  | AS TABLE ( { <column_definition> | <computed_column_definition> }
        [ <table_constraint> ] [ ,...n ] )
} [ ; ]
```

Il comando CREATE TYPE crea un tipo di dati alias o un tipo definito dall'utente nel database corrente. È possibile anche richiedere la composizione del comando di creazione del Tipo selezionando il tipo costruito e evocando il menu contestuale **Crea script** e **Genera codice per istruzione CREATE in** e infine **Nuova finestra editor di query**.




## 24-3 DROP TYPE (Transact-SQL)

Per eliminare un tipo si può usare il comando DROP TYPE, ottenibile anche dal menu contestuale appena visto sopra.

Non è possibile eliminare un tipo definito dall'utente finché non sono stati eliminati tutti i riferimenti a esso.

Per trovare le colonne che dipendono da un tipo definito dall'utente, si può accedere alla tabella di sistema **sys.columns**, come segue:

Transact-SQL

```
SELECT * FROM sys.columns
WHERE user_type_id = TYPE_ID('ComplexNumber');
```

per trovare le colonne calcolate e i vincoli CHECK le cui espressioni fanno riferimento al tipo si può accedere alla tabella di sistema **sys.column_type_usages**, come segue:

Transact-SQL

```
SELECT * FROM sys.column_type_usages
WHERE user_type_id = TYPE_ID('ComplexNumber');
```

Per trovare espressioni di colonne calcolate, espressioni di vincolo CHECK ed espressioni di moduli che dipendono da un tipo definito dall'utente si può accedere alle due tabelle di sistema **sys.sql_expression_dependencies** e **sys.objects**; si consulti il link seguente: http://msdn.microsoft.com/it-it/library/ms179298(v=sql.105).aspx

# PROGRAMMABILITÀ IN T-SQL

## 25 Linguaggio Transact-SQL

Il linguaggio T-SQL è un dialetto ottenuto da SQL-92 detenuto dalla Microsoft. Come linguaggio per database esso include elementi dei linguaggi per database e dei linguaggi procedurali.

Tra gli elementi che si suggerisce di esaminare in guide specialistiche possiamo annoverare:

1. **Commenti**
2. **Parole riservate**
3. **Costanti**
4. **Variabili e tipi**
5. **Procedure e Funzioni**
6. **Parametri**
7. **Istruzioni procedurali**
8. **Comandi SQL e istruzioni procedurali**

### 25-1 Commenti

I commenti possono essere inseriti su una riga distinta oppure nidificati alla fine di una riga di comando di T-SQL o in un'istruzione T-SQL. T-SQL prevede due modi per scrivere un commento in mezzo al codice.

I commenti inseriti con -- sono terminati dal carattere di nuova riga. I commenti possono essere di qualsiasi lunghezza.

Transact-SQL

```
-- dopo le due linee si può scrivere un commento

```

I commenti su più righe devono essere racchiusi dalle sequenze di simboli /* e */. Una convenzione utilizzata spesso per i commenti su più righe consiste nell'iniziare la prima riga di commento con /* e le righe successive con ** e nel terminare quindi il commento con */.

Transact-SQL

```
/*
** i due asterischi qui non sono obbligatori
** ma vige la convenzione di scriverli per commenti
** racchiusi tra i simboli detti
*/
```

### 25-2 Costanti

Una costante (detta spesso valore letterale o scalare), è un simbolo che rappresenta un valore di dati specifico. Il formato di una costante dipende dal tipo di dati del valore che essa rappresenta.

Le costanti di stringhe di caratteri sono racchiuse tra apici singoli e possono includere caratteri alfanumerici ordinari (a-z, A-Z e 0-9) e speciali come punto esclamativo (!),simbolo di chiocciola (@)simbolo di cancelletto (#) e altri. A queste costanti possono essere applicate le consuete regole di confronto previste per il database corrente, salvo che non siano state definite regole di confronto specifiche mediante la clausola COLLATE. Le stringhe vuote vengono rappresentate da due virgolette singole che non racchiudono alcun contenuto.

Transact-SQL

```
'Sassari Città Regia'
'La CPU lavora al 50% del suo potenziale!'
'La temperatura deve essere compresa tra %t1 e %t2.'
'L''aeroplano'
"L'aeroplano"
'' -- la stringa vuota sono due singoli apici consecutivi, senza spazi

```

Le costanti binarie (che in realtà sono sequenze di bit) hanno il prefisso 0x e rappresentano stringhe di numeri esadecimali non racchiuse tra virgolette.

Transact-SQL
```
0xAE    --vale 10101110
0x12Ef --vale 1001011101111
0x1234567890abcd    --vale 10010001101000101011001111000100100001010101111001101
0x      --vale una stringa binaria vuota (lunghezza zero)
```

Le costanti di tipo bit sono rappresentate dai soli numeri 0 oppure 1 e non vengono racchiuse tra virgolette. I numeri maggiori di uno vengono convertiti nel numero uno.

Transact-SQL
```
0    --vale il bit zero
1    --vale il bit uno
```

Le costanti datetime sono rappresentate tramite valori di data di tipo carattere in formati specifici e sono racchiuse tra **virgolette singole**. Occorre fare attenzione se il server è installato in notazione italiana o anglosassone.

Transact-SQL
```
'Dicembre 25, 2012'
'25 Dicembre, 2012'
'121225'
'12/25/12'     --notazione anglosassone
'25/12/12'     --notazione italiana
'14:30:24'     --vale ore 14 minuti 30 secondi 24
'04:24 PM'     --vale 4:24 del pomeriggio
```

Le costanti intere sono rappresentate da una sequenza di numeri non racchiusa tra virgolette e priva di separatori decimali. Le costanti intere devono essere numeri interi e non includere decimali.

Transact-SQL
```
98765
10101          --è un numero intero diecimila cento uno
+145345234     --è un numero intero positivo con segno
-2147483648    --è un numero intero negativo
```

Le costanti decimali vengono rappresentate da una sequenza di numeri non racchiusa tra virgolette e con un separatore decimale (il punto).

Transact-SQL
```
1894.1204   --è un numero decimale perché c'è la virgola (il dot)
2.000
0.314
3.1415
+145345234.2234
-2147483648.10
```

Le costanti float e real vengono rappresentate con la notazione scientifica.

Transact-SQL
```
101.5E5
0.5E-2
```

Le costanti money vengono rappresentate come stringhe di numeri con separatore decimale facoltativo e un simbolo di valuta come prefisso facoltativo. Le costanti money non vengono racchiuse tra virgolette.

Transact-SQL
```
$12          --è una valuta perché c'è il simbolo dollaro
$542023.14   --è una valuta perché c'è il simbolo dollaro
-$45.56
+$423456.99
```

### 25-3 Identificatori

L'identificatore di un oggetto di database coincide col suo nome. È possibile nominare qualsiasi elemento di un database con un identificatore, tra cui lo stesso database e i suoi oggetti (tipi, tabelle, colonne, indici, viste, procedure, funzioni, trigger, vincoli, regole e altri ancora). Gli identificatori sono obbligatori per la maggior parte degli oggetti. Solo rari oggetti possono essere definiti senza identificatore. Un identificatore è costituito da una sequenza da 0 a 128 caratteri.

Talvolta è opportuno delimitare un identificatore tra parentesi quadre (identificatori delimitati) oppure tra virgolette doppie :

Transact-SQL

```
SELECT *
FROM [miaTabella]       -- miaTabella è un identificatore delimitato
WHERE [miaColonna] > 10  -- miaColonna è un identificatore delimitato
```

### 25-4 Parole riservate

Come ogni linguaggio di programmazione anche T-SQL prevede un elenco di parole chiave riservate ai comandi e alla sintassi operativa. Queste parole non possono essere utilizzate come identificatori come per es. CREATE, SELECT, ecc. Un elenco delle parole chiave si può trovare qui: http://msdn.microsoft.com/it-it/library/ms189822.aspx

# 26 Tipi di dato in SQL Server 2008

I tipi di dato in T-SQL possono essere impiegati per:

- Definizione di nuovi tipi
- Definizione di colonne (per tabelle e viste)
- Dichiarazioni di variabili
- Dichiarazioni di parametri
- Risultati di funzione

T-SQL annovera un notevole ammontare di differenti tipi di dato, classificabili per ambito di riferimento:

- Numerici (interi e con virgola)
- Stringhe (sequenze di caratteri)
- Temporali (date e orari)
- Altri (moneta e bit)

In questa sede faremo un semplice elenco dei tipi e del loro scopo. I dettagli sono reperibili nella guida in linea.

### 26-1 Numerici

**int** Intero tra -2147483648 e 2147483647 ( 4 byte )

**smallint** Intero tra -32768 e 32,767 ( 2 byte )

**tinyint** Intero tra 0 e 255 ( 1 byte )

**decimal[(p[, s])] e numeric[(p[, s])]** Decimal e Numeric sono sinonimi, e servono per numeri con la virgola; il valore **p** è la precisione, che rappresenta il numero massimo di cifre decimali complessive (da entrambe le parti della virgola). Invece **s** è la scala, ovvero il numero di massimo di cifre decimali dopo la virgola e deve essere minore od uguale a **p**.

**float[(n)]** Contiene numeri a virgola mobile positivi e negativi, compresi tra 2.23E-308 e 1.79E308 per i valori positivi e tra -2.23E-308 e -1.79E308 per i valori negativi, occupa 8 bytes di memoria ed ha una precisione di 15 cifre

**real** Contiene numeri a virgola mobile positivi e negativi comprese tra 1.18E-38 e 3.40E38 per i valori positivi e tra -1.18E-38 e -3.40E38 per i valori negativi, occupa 4 bytes di memoria ed ha una precisione di 7 cifre

### 26-2 Stringhe

Possono essere stringhe di singoli byte di caratteri oppure stringhe di caratteri Unicode:

**char[(n)]** Ha una lunghezza fissa e può contenere fino ad 8000 caratteri ANSI (cioè 8000 bytes)

**varchar[(n)]** Ha una lunghezza variabile e può contenere fino ad 8000 caratteri ANSI (cioè 8000 bytes)

**nchar[(n)]** Ha una lunghezza fissa e può contenere fino a 4000 caratteri UNICODE (cioè 8000 bytes, ricordiamo che per i caratteri UNICODE servono 2 bytes per memorizzare un carattere)

**nvarchar[(n)]** ha una lunghezza variabile e può contenere fino a 4000 caratteri UNICODE (cioè 8000 bytes, ricordiamo che per i caratteri UNICODE servono 2 bytes per memorizzare un carattere)

### 26-3 Temporali

Alcuni tipi di dati temporali sono:

**datetime** Ammette valori compresi dal 1 gennaio 1753 al 31 dicembre 9999 (precisione al trecentesimo di secondo), occupa uno spazio di 8 byte

**smalldatetime** Meno preciso del precedente (precisione al minuto), occupa uno spazio di 4 byte

**date** Occupa uno spazio di 3 byte per memorizzare solo date

**time** Occupa uno spazio da 3 a 5 byte per memorizzare solo orari

### 26-4 Altri

**bit** Tipicamente è usato per rappresentare i flag, vero/false o true/false o si/no, perché può accettare solo due valori 0 o 1. Occupa un bit ovviamente. Le colonne che hanno un tipo dati bit non possono avere valori nulli e non possono avere indici.

**timestamp** Occupa 8 bytes ed è un contatore incrementale per colonna assegnato automaticamente da SQL Server 7.

**money** Contiene valori monetari da -922337203685477.5808 a 922337203685477.5807 con una precisione al decimillesimo di unità monetaria, occupa 8 bytes di memoria

**smallmoney** Contiene valori monetari da - 214748.3648 a 214748.3647 con una precisione al decimillesimo di unità monetaria, occupa 4 bytes di memoria.

**binary[(n)]** Ha una lunghezza fissa e può contenere fino ad 8000 bytes di dati binari

**varbinary[(n)]** Ha una lunghezza variabile e può contenere fino ad 8000 bytes di dati binari

Esistono anche altri tipi di dati rivolti alla gestione di oggetti multimediali, geografici, oggetti e file esterni, per i quali si rinvia alla documentazione ufficiale di T-SQL.

### 26-5 Valori NULL

**NULL** è un particolare valore SQL che può essere assegnato a una locazione di qualsiasi tipo (quindi è compatibile con tutti) ma, se confrontato con altri tipi, risulta sempre diverso. Rappresenta un dato sconosciuto o inapplicabile. Ogni espressione aritmetica o booleana che coinvolge valori NULL restituisce un valore NULL. Nelle funzioni di aggregazione (es. AVG) i valori NULL vengono eliminati. Nei raggruppamenti non costituisce categoria.

# 27 Variabili

### 27-1 Variabili locali

**Fonte:** http://msdn.microsoft.com/it-it/library/ms187953(v=sql.100).aspx

Le variabili sono locazioni per i dati, contraddistinte da un identificatore, che possono essere dichiarate localmente nel corpo di una routine o di un codice batch. In T-SQL la dichiarazione avviene con l'istruzione DECLARE mentre e i valori possono essere assegnati con le istruzioni SET e SELECT.

Il nome di un variabile T-SQL deve sempre iniziare con il carattere chiocciola (**@**).

L'istruzione **DECLARE** definisce una variabile Transact-SQL. Per le variabili numeriche sono definite precisione e scala. Dopo la dichiarazione, tutte le variabili sono inizializzate col valore NULL, salvo sia specificato diversamente.

Transact-SQL

```
DECLARE @cerca varchar(30);
DECLARE @frase varchar(30) = 'Zoc%';
SET @cerca = 'Zoc%';
DECLARE @nome nvarchar(50), @Prezzo money, @età int, @flag bit;
```

Le variabili possono essere utilizzate per compiti quali conteggi, accumulatori, insieme alle istruzioni procedurali come WHILE ma anche in clausole come WHERE.

L'ambito di una variabile locale è il blocco in cui è dichiarata (per es. corpo di procedura, o blocco IF).

Nella dichiarazione di una variabile si deve sempre specificare il tipo di appartenenza.

### 27-2 Variabili globali (vedi funzioni di sistema)

**Fonte:** http://msdn.microsoft.com/it-it/library/ms187953(v=sql.100).aspx

Il nome di alcune funzioni T-SQL di sistema inizia con due simboli di chiocciola (**@@**). Benché nelle versioni precedenti di SQL Server tali funzioni siano denominate **variabili globali**, non si tratta in realtà di variabili e non funzionano come le variabili. Sono a tutti gli effetti **funzioni di sistema** e la loro relativa sintassi segue le regole delle funzioni. Alcune funzioni di sistema sono:

- @@CONNECTIONS – Restituisce il numero di tentativi di login effettuati da quando il sistema è stato avviato
- @@ERROR – Restituisce informazioni sul valore ritornato dall'ultima istruzione eseguita
- @@LANGID – Restituisce l'identificatore del linguaggio utilizzato correntemente nel database
- @@ROWCOUNT – Restituisce in numero di righe coinvolte dall'ultima istruzione
- @@SERVERNAME – Restituisce informazioni riguardanti il server

**L'elenco delle funzioni di sistema è qui:** http://msdn.microsoft.com/it-it/library/ms187786(v=sql.100).aspx

# 28 Operatori e Funzioni

### 28-1 Operatori sui dati

Sui dati sono ammesse le operazioni canoniche dei linguaggi di programmazione.

Sui dati numerici sono ammesse le operazioni algebriche (+, -, *. /, %).

Sui dati alfanumerici è ammessa l'operazione di concatenazione rappresentata dal simbolo +. Nelle clausole WHERE e HAVING si può usare l'operazione LIKE e i metacaratteri di confronto.

Sui dati booleani (logici) sono ammessa le operazioni di negazione (NOT), di congiunzione (AND) e disgiunzione (OR) scritte per esteso. Questi operatori sono spesso utilizzati nelle clausole WHERE ed HAVING delle query.

### 28-2 Funzioni in generale

Il linguaggio T-SQL mette a disposizione del programmatore molte funzioni predefinite. Le funzioni possono essere suddivise in due categorie:

- Funzioni aggregate sono le funzioni standard di SQL ANSI-92 ovvero AVG, COUNT, MAX, MIN, SUM;
- Funzioni scalari sono funzioni fuori standard che restituiscono valori sui tipi del linguaggio.

In questo documento non saranno analizzate nello specifico le funzioni T-SQL, ma è offerta una panoramica che possa aiutare il programmatore a orientarsi nello stile di costruzione delle applicazioni.

### 28-3 Funzioni aggregate

Le funzioni aggregate (o funzioni di aggregazione) sono applicabili ad un insieme di dati riferiti ad una colonna (o a una espressione tra colonne) e restituiscono sempre un singolo valore. Le funzioni aggregate sono:

**AVG** – Calcola la media aritmetica dei valori di una colonna (su valori numerici);

**MAX** – Calcola il valore massimo dei dati di una colonna (su valori numerici, stringhe e temporali);

**MIN** – Calcola il valore minimo dei dati di una colonna su valori numerici, stringhe e temporali);

**SUM** – Calcola la somma dei valori di una colonna (su valori numerici);

**COUNT** – Rende il conteggio del numero di valori non nulli in una colonna. Se usata nella forma COUNT(*) rende il numero delle righe di una tabella.

### 28-4 Funzioni scalari

Le funzioni scalari sono funzioni che permettono il calcolo di valori su specifici tipi di dato riferiti al linguaggio T-SQL. A seconda dell'ambito in cui operano possono essere suddivise nelle seguenti categorie:

- Funzioni numeriche rendono valori trigonometrici, esponenziali, costanti matematiche, ecc.
- Funzioni temporali rendono dati di sistema e calcolano operazioni su tipi temporali;
- Funzioni testuali compiono ricerche e operazioni sulle stringhe di testo;
- Funzioni di sistema rendono informazioni sul sistema operativo e sul server.

**Le funzioni scalari sono elencate nell'Appendice B.**

# 29 Istruzioni procedurali

## 29-1 Dichiarazione di variabili

L'istruzione **DECLARE** definisce una variabile Transact-SQL. Per le variabili numeriche sono definite precisione e scala. Dopo la dichiarazione, tutte le variabili sono inizializzate col valore NULL, salvo sia specificato diversamente.

Transact-SQL

```
DECLARE @cerca varchar(30);
DECLARE @frase varchar(30) = 'Zoc%';
DECLARE @nome nvarchar(50), @Prezzo money, @età int, @flag bit;
```

## 29-2 Assegnazione alle variabili

Ci sono diversi modi per assegnare un valore ad una variabile.

Un primo modo è assegnarlo direttamente al momento della sua dichiarazione (inizializzazione):

Transact-SQL

```
DECLARE @intero int = 123;
DECLARE @frase varchar(30) = 'Zoc%';
DECLARE @prezzo money = $10.75;
DECLARE @flag bit = 1;
```

Per assegnare un valore alla variabile, è possibile anche utilizzare l'istruzione SET. Per usare SET la variabile deve essere già dichiarata: è consigliabile adottare sempre questo metodo. Se si desidera utilizzare l'istruzione SET per assegnare un valore a una variabile, specificare il nome della variabile e il valore da assegnarvi e il simbolo =.

Transact-SQL

```
SET @intero = 123;
SET @intero = @intero + 54;
SET @attore  = 'Dino';
SET @attore  = @attore + 'sauro';
```

Per assegnare un valore a una variabile, è inoltre possibile fare riferimento alla variabile stessa nell'elenco di selezione di un'istruzione SELECT.

Transact-SQL

```
DECLARE @TassaMassima money;
DECLARE @DataRecente date;
/* assegnazione nella clausola SELECT */
SELECT @TassaMassima = MAX(Tasse)
FROM Studenti;
/* assegnazione nella clausola SELECT */
SELECT @DataRecente = MAX(Data)
FROM Prestiti;
```

Occorre fare attenzione ad usare questo metodo, specie con assegnazioni multiple correlate: se in una singola istruzione SELECT sono presenti più clausole di assegnazione, l'ordine di valutazione delle espressioni non viene garantito.

## 29-3 Utilizzo delle variabili

**Fonte:** http://msdn.microsoft.com/it-it/library/ms174290.aspx

Oltre all'accesso in scrittura (assegnazione) è possibile utilizzare la variabile con accesso in lettura (valore). Un uso ordinario è nella clausola WHERE di un comando SELECT, per filtrare i record di una query.

Transact-SQL
```
-- Dichiarazione di due variabili
DECLARE @Marca nvarchar(10), @Targa nvarchar(15);

-- Inizializzazione
SET @Marca = N'Feat';
SET @Targa = N'AY 313 HX';

-- Uso nella clausola WHERE di un comando SELECT
SELECT *
FROM Veicoli
WHERE Marca = @Marca
   OR Targa = @Targa;
GO
```
La vita di una variabile è limitata al blocco in cui è stata dichiarata (di solito una routine, come una procedura, una funzione o un trigger, ma anche un blocco di un'istruzione come IF o WHILE).

## 29-4 Elementi del linguaggio per il controllo di flusso

Le parole chiave del linguaggio per il controllo di flusso T-SQL sono le seguenti:

- BEGIN...END — blocco (sequenza) di istruzioni
- IF...ELSE — istruzione decisionale condizionata
- WHILE — istruzione di ciclo (loop) con guardia logica booleana
- BREAK — interruzione di un ciclo o di una istruzione
- CONTINUE — prosecuzione di un comando
- GOTO label — salto di flusso a una etichetta
- RETURN — uscita dalla procedura con restituzione di valore
- THROW — blocco con protezione da errori e eccezioni
- TRY...CATCH — blocco con protezione da errori e eccezioni
- WAITFOR — attesa

## 29-5 Blocco / Sequenza

**BEGIN...END**

Permette di raccogliere una sequenza di istruzioni T-SQL per consentirne l'esecuzione consecutiva.

Transact-SQL
```
BEGIN
  Istruzione_1;
  Istruzione_2;
  . . .
  Istruzione_K;
END
```
Con Istruzione_X si intende qualsiasi istruzione o altro blocco di istruzioni T-SQL valide. Le istruzioni sono eseguite in sequenza dal sopra verso sotto.

Transact-SQL
```
DECLARE @PrezzoStandard money , @Sconto money ;
IF ( EXISTS (SELECT IDProdotto FROM Prodotti WHERE Prezzo > $1000) )
   BEGIN
      @PrezzoStandard = $1000;
      @Sconto = $100;
   END
ELSE
      @Sconto = $10;
```

## 29-6 Istruzione decisionale

**Fonte:**

### IF...ELSE

Verifica se eseguire altre istruzioni o alternative. Valuta le condizioni per l'esecuzione di un'istruzione T-SQL.

Transact-SQL

```
IF Condizione
     Istruzione_V
[ ELSE
     Istruzione_F
]
```

**Condizione** è una espressione logica che restituisce TRUE o FALSE. La condizione può includere operatori di confronto (come maggiore) e comandi SELECT. Se la condizione include un'istruzione SELECT, essa deve essere racchiusa tra parentesi tonde.

**Istruzione_X** è una qualsiasi istruzione o blocco di istruzioni T-SQL valido. Nel ramo dopo IF si ammette una sola istruzione, salvo l'uso di un blocco BEGIN END. L'istruzione T-SQL che segue una parola chiave IF e le relative condizioni viene eseguita se la condizione rende TRUE.

**ELSE** è una parola chiave facoltativa che introduce un'altra istruzione T-SQL; questa è eseguita se e solo se la condizione restituisce FALSE.

Transact-SQL

```
IF ( SELECT MAX(Prezzo) FROM Prodotti) > $500 )
   BREAK
ELSE
   CONTINUE ;
```

## 29-7 Ciclo while

### 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.

Transact-SQL

```
WHILE Condizione
     Istruzione_X;
```

**Condizione** è una espressione logica che restituisce TRUE o FALSE. La condizione può includere operatori di confronto (come maggiore) e comandi SELECT. Se la condizione include un'istruzione SELECT, essa deve essere racchiusa tra parentesi tonde.

**Istruzione_X** è una qualsiasi istruzione o blocco di istruzioni T-SQL valido. Nel ramo dopo IF si ammette una sola istruzione, salvo l'uso di un blocco BEGIN END. L'istruzione T-SQL che segue una parola chiave IF e le relative condizioni viene eseguita se la condizione rende TRUE.

Transact-SQL

```
WHILE (SELECT AVG(Prezzo) FROM Prodotti) < $300
   BEGIN
      UPDATE Prodotti
           SET Prezzo = Prezzo * 2 ;
      IF ( (SELECT MAX(Prezzo) FROM Prodotti) > $500 )
           BREAK
      ELSE
         CONTINUE
   END
PRINT 'Prezzi pazzi per pizzi e pizze';
```

## 29-8 Uscita / Interruzione

**BREAK**

Causa l'uscita dal ciclo più interno di un'istruzione WHILE o IF…ELSE all'interno di un ciclo WHILE. Il flusso di esecuzione salta alla prima istruzione che si trova dopo la parola chiave END, che contraddistingue la fine del ciclo. Conviene che l'istruzione BREAK sia condizionata mediante un controllo IF.

## 29-9 Continua / Prosecuzione

**CONTINUE**

Forza il ricominciare di un ciclo WHILE. Tutte le istruzioni che seguono la parola chiave CONTINUE vengono ignorate. L'istruzione CONTINUE è spesso attivata, anche se non sempre, da un test IF.

Transact-SQL

```
WHILE (SELECT AVG(Prezzo) FROM Prodotti) < $300
   BEGIN
      UPDATE Prodotti
          SET Prezzo = Prezzo * 2 ;
      IF ( (SELECT MAX(Prezzo) FROM Prodotti) > $500 )
          BREAK
      ELSE
         CONTINUE
   END
PRINT 'Prezzi pazzi per pizzi e pizze';
```

## 29-10 Salto

**GOTO**

Forza il flusso di esecuzione a proseguire in un altro punto, individuato da un'etichetta. L'istruzione o le istruzioni T-SQL successive al GOTO saranno ignorate poiché l'elaborazione sarà ripresa in corrispondenza dell'etichetta. È possibile utilizzare istruzioni ed etichette GOTO in qualsiasi punto di una procedura, un batch o un blocco di istruzioni. Le istruzioni GOTO possono essere nidificate.

Transact-SQL

```
-- Dichiarazione di una etichetta
miaEtichetta:
```

Transact-SQL

```
-- Istruzione di salto alla etichetta
GOTO label ;
```

Esempio:

Transact-SQL

```
DECLARE @Cont int;
SET @Cont = 1;
WHILE @Cont < 10
BEGIN
    SELECT @Cont ;
    SET @Cont = @Cont + 1;
    IF @Cont = 4 GOTO Salto_001 ; -- Salta alla prima etichetta
    IF @Cont = 5 GOTO Salto_002 ; -- Istruzione mai eseguita
END
Salto_001:
    PRINT Salto effettuato in 001'
    GOTO Salto_003; -- Evita di eseguire 002 e salta a 003
Salto_002:
```

```
    PRINT Salto effettuato in 002'
Salto_003:
    PRINT Salto effettuato in 003'
```

## 29-11 Gestione degli errori

### TRY...CATCH

**Fonte:** http://msdn.microsoft.com/it-it/library/ms175976.aspx

L'istruzione TRY...CATCH permette la gestione degli errori per T-SQL, in modo simile alla gestione delle eccezioni dei linguaggi MS Visual C#. Una sequenza di istruzioni T-SQL può essere racchiusa in un blocco TRY. Se accade un errore nel blocco TRY, il flusso è trasferito a un'altra sequenza di istruzioni racchiuso in un blocco CATCH.

Transact-SQL

```
BEGIN TRY
     Istruzione_X;
END TRY
BEGIN CATCH
     Istruzione_X;
END CATCH
```

Un costrutto TRY…CATCH intercetta tutti gli errori di esecuzione con livello di gravità superiore a 10 che non determinano la chiusura della connessione al database.

Un blocco TRY deve essere immediatamente seguito da un blocco CATCH associato. L'inclusione di qualsiasi altra istruzione tra le istruzioni END TRY e BEGIN CATCH genera un errore di sintassi.

Se non si verificano errori nel codice racchiuso in un blocco TRY, al termine dell'esecuzione dell'ultima istruzione nel blocco TRY il controllo passa all'istruzione immediatamente successiva all'istruzione END CATCH associata. Se si verifica un errore nel codice racchiuso in un blocco TRY, il controllo passa alla prima istruzione nel blocco CATCH associato. Se l'istruzione END CATCH è l'ultima istruzione in una stored procedure o un trigger, il controllo viene restituito all'istruzione che ha chiamato la stored procedure o attivato il trigger.

Al termine del codice nel blocco CATCH, il controllo passa all'istruzione immediatamente successiva all'istruzione END CATCH. Gli errori intercettati da un blocco CATCH non vengono restituiti all'applicazione chiamante. Se qualsiasi parte delle informazioni sugli errori deve essere restituita all'applicazione, il codice nel blocco CATCH deve utilizzare a tale scopo meccanismi come i set di risultati SELECT o le istruzioni RAISERROR e PRINT.

### THROW

**Fonte:** http://msdn.microsoft.com/it-it/library/ee677615.aspx

L'istruzione consente di generare forzatamente un'eccezione e di trasferire l'esecuzione a un blocco CATCH associato ad un costrutto TRY…CATCH.

Transact-SQL

```
THROW [ { error_number | @local_variable },
        { message | @local_variable },
    { state | @local_variable }
] [ ; ]
```

| | |
|---|---|
| **error_number** | valore intero (costante o variabile) che rappresenta l'eccezione; il numero errore deve essere maggiore o uguale a 50000 e minore o uguale a 2147483647. |
| **message** | valore stringa (costante o variabile) per descrivere l'eccezione; è di tipo nvarchar(2048). |
| **state** | valore tinyint (costante o variabile) tra 0 e 255 con cui indicare uno stato associato al messaggio. |

Esempio:

Transact-SQL
```
DECLARE @Msg NVARCHAR(2048);
SELECT @Msg = FORMATMESSAGE(1127);
THROW 50001, @Msg, 1;
```

Transact-SQL
```
THROW 51000, 'Operazione non ammessa.' , 1 ;
```

Transact-SQL
```
BEGIN TRY
   INSERT dbo.miaTabella(ID) VALUES (1);
   --  Causa un errore 2627, violazione del vincolo di PRIMARY KEY
   INSERT dbo.miaTabella(ID) VALUES (1);
END TRY
BEGIN CATCH
    PRINT 'Sei giunto nel blocco CATCH!';
    THROW;
END CATCH;
```

## 29-12 Altre istruzioni

### WAITFOR

**Fonte:** http://msdn.microsoft.com/it-it/library/ms187331.aspx

Impedisce l'esecuzione di una routine o una transazione fino ad un'ora specifica oppure finché sia trascorso un intervallo di tempo specifico oppure finché un'istruzione specifica modifica o restituisce almeno una riga.

Transact-SQL
```
WAITFOR
{
    DELAY 'time_to_pass'
  | TIME 'time_to_execute'
  | [ ( receive_statement ) | ( get_conversation_group_statement ) ]
    [ , TIMEOUT timeout ]
}
```

**DELAY** questa clausola è obbligatoria e serve per indicare l'ammontare di tempo che deve trascorrere (massimo 24 ore) prima che l'esecuzione di una routine possa continuare.

**'time_to_pass'** questo valore (di tipo datetime) indica il periodo di tempo di attesa. Può essere una costante oppure una variabile locale. Non è possibile indicare una data e pertanto la parte relativa alla data (prevista per un tipo datetime) non è consentita.

**TIME** questa clausola è facoltativa e serve per indicare l'ora precisa di esecuzione della routine o della transazione.

**'time_to_execute'** questo valore (di tipo datetime) indica l'ora in cui l'istruzione WAITFOR viene interrotta. Il valore può essere una costante oppure una variabile locale. Non è possibile specificare date come sopra.

**receive_statement** Una istruzione RECEIVE valida.

## 29-13 Istruzione RETURN

### RETURN

**Fonte:** http://msdn.microsoft.com/it-it/library/ms187331.aspx

L'istruzione RETURN definisce il risultato reso da una funzione. Il suo effetto è il seguente: esce immediatamente da una query o da una procedura e eventualmente specifica un valore da restituire. RETURN è un'istruzione immediata e

completa e può essere utilizzata in qualsiasi momento per uscire da una procedura, da una funzione, da un trigger, da un batch o da un blocco di istruzioni (BEGIN END). Le istruzioni che seguono RETURN non saranno eseguite.

Transact-SQL

```
RETURN [ intero_restituito ]
```

Il valore (intero) restituito è opzionale. il valore restituito Valore intero restituito.

Se non indicato diversamente, tutte le stored procedure restituiscono il valore 0 ad indicare l'esito positivo e un valore diverso da zero per indicare l'errore.

# 30 Linguaggio T-SQL e routine

## 30-1 Forme di routine del linguaggio

Si sono già introdotte le routine del linguaggio nel capitolo sulla programmabilità. Si è visto che le routine rientrano nel DDL (Data Description Language) poiché sono considerate alla stregua delle strutture dei dati. Si è anche chiarito che SQL prevede tre tipi di routine differenti per tipologia di esecuzione e di risultato:

- Le stored procedure o procedure memorizzate, routine che generalmente non restituiscono valori ma accettano parametri di ingresso e/o uscita;
- Le funzioni definite dall'utente che, oltre ad accettare parametri di ingresso e/o uscita, restituiscono anche un singolo valore (risultato);
- I trigger, che sono sequenze di comandi attivate da eventi, rivolte alla gestione della congruenza del database.

Vediamo adesso alcuni aspetti specifici della programmazione delle routine.

**Fonte:** http://msdn.microsoft.com/it-it/library/ms190669(v=sql.105).aspx

## 30-2 Discussione sui parametri

I parametri sono il principale strumento di comunicazione tra una procedura e il programma chiamante. Generalmente i parametri servono per:

- Inviare dei dati dal chiamante alla procedura affinché essa possa processare correttamente i dati;
- Restituire dei dati dalla procedura al chiamante, perché possa ricevere i risultati elaborati.

Quando il chiamante invoca una routine, può passare dei valori ad essa tramite i parametri che sono previsti; tali valori sono utilizzati in modo analogo alle variabili standard di T-SQL. La routine può inoltre restituire valori al programma chiamante attraverso i parametri in uscita. In T-SQL una routine può includere fino a 2100 parametri e per ogni parametro occorre stabilire un nome, un tipo di dati, una direzione e un valore predefinito.

## 30-3 Impostazione di un nome di parametro

In T-SQL ogni parametro deve essere un nome univoco nell'ambito della routine; i nomi di parametro devono iniziare col carattere @ (come una normale variabile) e rispettare le regole previste per gli identificatori. Il nome di parametro è utilizzato per leggere e modificare il valore del parametro stesso.

Nell'invocazione di una routine, è possibile passare i valori ai parametri sia denominando esplicitamente i parametri (seguiti da una assegnazione del valore di chiamata) oppure specificando i soli valori dei parametri senza denominarli e assumendo un passaggio posizionale. La denominazione dei parametri durante l'esecuzione della stored procedure consente di specificare i parametri in qualsiasi ordine. Se i parametri non sono denominati, è necessario specificarli nello stesso ordine (da sinistra a destra) definito nella stored procedure, inclusi quelli facoltativi o che prevedono valori predefiniti.

**Esempio**: la seguente la stored procedure miaProc prevede tre parametri denominati @uno, @due e @tre

Transact-SQL

```
CREATE PROCEDURE miaProc
   @uno bit ,
   @due varchar(3) ,
   @tre int
AS
    SET NOCOUNT ON;
    SELECT @uno, @due, @tre
```

I valori passati alla stored procedure possono essere assegnati ai nomi di parametro, ad esempio:

Transact-SQL
```
EXECUTE miaProc @due = '2', @uno = 1, @tre = 3;
```

Ma è possibile anche passare I valori in base alla posizione senza denominarli:

Transact-SQL
```
EXECUTE miaProc 1 , '2', 3;
```

### 30-4 Specifica di un tipo di dati per i parametri

Ogni parametro di una stored procedure è definito con un tipo di dati, analogamente a una variabile; un parametro di T-SQL ammette qualsiasi tipo di dati del linguaggio (inclusi quelli definiti dall'utente) ad eccezione dei tipi table. Il tipo deve seguire immediatamente il nome del parametro.

Il tipo di dati di un parametro determina la forma e l'intervallo dei valori ammessi. Per esempio, un parametro di tipo **tinyint**, ammette solo valori interi compresi tra 0 e 255 e se una invocazione passa un valore incompatibile o fuori dall'intervallo si solleva un errore.

### 30-5 Impostazione della direzione di un parametro

T-SQL ammette una direzione per ogni parametro. Le direzioni possono essere:

- **IN** ovvero il valore è passato dal chiamante al parametro (input) della stored procedure (forma predefinita)
- **OUT** ovvero la stored procedure rende un valore al chiamante depositandolo in un parametro (output)

Per specificare un parametro output è necessario includere la parola chiave OUTPUT nella definizione del parametro nella stored procedure. Il programma chiamante deve anche specificare la parola chiave OUTPUT quando chiama la stored procedure insieme all'argomento utilizzato.

### 30-6 Specifica di un valore di parametro predefinito

È possibile definire un parametro facoltativo specificando per esso un valore predefinito. Quando si esegue la stored procedure, userà il valore predefinito nel caso che non sia stato passato nessun altro valore. Senza un valore predefinito il parametro è obbligatorio. Se non è previsto un valore predefinito e il chiamante non specifica un valore per il parametro, verrà sollevato un errore.

Se non è possibile indicare un migliore valore predefinito, si può sempre specificare il valore NULL e pilotare un messaggio personalizzato se il chiamante scorda di indicare un valore diverso.

# 31 Linguaggio T-SQL ed esempi di routine

Di seguito sono proposti esempi di procedure con parametri di vario tipo e forma:

### 31-1 Procedura semplice

Transact-SQL
```
CREATE PROCEDURE RendiStudenti_A
AS
    SET NOCOUNT ON;
    SELECT Cognome, Nome, Tasse
    FROM Studenti;
```

### 31-2 Procedura semplice con parametri

Transact-SQL
```
CREATE PROCEDURE RendiStudenti_B
    @Cognome nvarchar(50),
    @Nome nvarchar(50)
AS
    SET NOCOUNT ON;
    SELECT Cognome, Nome, Tasse
    FROM Studenti
    WHERE Cognome = @Cognome OR Nome = @Nome;
```

### 31-3 Procedura semplice con parametri jolly

Transact-SQL

```
CREATE PROCEDURE RendiStudenti_C
    @Cognome nvarchar(50) = N'D%',
    @Nome nvarchar(50) = N'%'
AS
    SET NOCOUNT ON;
    SELECT Cognome, Nome, Tasse
    FROM Studenti
    WHERE Cognome LIKE @Cognome OR Nome LIKE @Nome;
```

### 31-4 Procedura con parametri OUTPUT

Transact-SQL

```
CREATE PROCEDURE TrovaStatistiche
       @MinPrezzo money = 0 OUT
    ,  @MaxPrezzo money = 0 OUT
    ,  @MediaPrezzo money = 0 OUT
AS
     SET NOCOUNT ON;
     SELECT Prodotto.Nome, Prodotto.Prezzo
     FROM Prodotti AS P
     WHERE P.Prezzo > @MinPrezzo;
     -- Definizione dei parametri output @MaxPrezzo e @MediaPrezzo
     SELECT @MaxPrezzo = MAX(Prezzo), @MediaPrezzo = AVG(Prezzo)
     FROM Prodotti
     WHERE P.Prezzo > @MinPrezzo;
     -- Definizione del parametro output @MinPrezzo (potrebbe cambiare)
     SELECT @MinPrezzo = MIN(Prezzo)
     FROM Prodotti
     WHERE P.Prezzo > @MinPrezzo;
```

# 32 Invocazioni di routine

### 32-1 Invocazione di stored procedure senza parametri

**Fonte:** http://msdn.microsoft.com/it-it/library/ms378917(v=sql.100).aspx

La forma più semplice di stored procedure in T-SQL è senza parametri e restituisce un solo insieme di risultati.

Per invocare le procedure sfruttando i driver JDBC per MS SQL Server, si fa riferimento alla classe **SQLServerStatement** che consente di chiamare la stored procedure ed elaborare i dati restituiti. L'uso del driver prevede di utilizzare la sequenza **SQL call** la cui sintassi (semplicissima) è la seguente:

Visual C#

```
{call procedure-name}
```

Per esempio, consideriamo la seguente procedura posta in un database di esempio Scuola:

Transact-SQL

```
CREATE PROCEDURE ElencaMaggiorenni
AS
BEGIN
   SELECT   S.Nome + ' ' + S.Cognome AS Nominativo,
            C.Anno + ' ' + C.Sezione + ' ' + C.Indirizzo AS ClasseFrequentata
   FROM Studenti AS S, Classi AS C
   WHERE S.Classe = C.IDClasse
     AND Età >= 18;
END
```

Questa stored procedure restituisce una sola tabella con una sola colonna con dati presi dalla tabella Studenti.

Visual C#
```
public static void invocaProcSenzaParam (Connection x) {
   try {
      Statement stmt = x.createStatement();
      ResultSet rs = stmt.executeQuery("{call dbo.ElencaMaggiorenni}");

      while (rs.next()) {
          string s = "Studente:" + rs.getString("Nominativo");
          s += rs.getString("ClasseFrequentata");
          System.out.println(s); /*dipende dall'uso*/
      }
      rs.close();
      stmt.close();
   }
   catch (Exception e) {
      e.printStackTrace();
   }
}
```

Nell'esempio illustrato qui sopra la funzione usa una connessione già aperta (il parametro) e mediante il metodo **executeQuery** può invocare la stored procedure **ElencaMaggiorenni**.

## 32-2 Utilizzo di una stored procedure con parametri di input

Se si invoca una stored procedure utilizzando il driver JDBC con parametri IN, si usa la sequenza **SQL call** insieme al metodo **prepareCall** della classe **SQLServerConnection**. La loro sintassi è la seguente:

Visual C#
```
{call procedure-name[([parameter][,[parameter]]...)]}
```

**Fonte:** http://msdn.microsoft.com/it-it/library/ms378675(v=sql.100).aspx

Per esempio, consideriamo la seguente procedura posta in un database di esempio Scuola:

Transact-SQL
```
CREATE PROCEDURE ElencaOltreEtà
   @EtàSoglia int = 18
AS
BEGIN
   SELECT   S.Nome + ' ' + S.Cognome AS Nominativo,
            C.Anno + ' ' + C.Sezione + ' ' + C.Indirizzo AS ClasseFrequentata
   FROM Studenti AS S, Classi AS C
   WHERE S.Classe = C.IDClasse
     AND Età >= @EtàSoglia;
END
```

Questa stored procedure è analoga alla precedente, ma usa un parametro INPUT.

Per invocarla potremmo eseguire:

Visual C#
```
public static void invocaProcConParam (Connection x) {
   try {
      PreparedStatement pstmt = x.prepareStatement(
                                        "{call dbo. ElencaOltreEtà (?)}"  );
      pstmt.setInt(1, 17);
      ResultSet rs = pstmt.executeQuery();
      while (rs.next()) {
          string s = "Studente:" + rs.getString("Nominativo");
          s += rs.getString("ClasseFrequentata");
```

```
            System.out.println(s); /*dipende dall'uso*/
        }
        rs.close();
        pstmt.close();
    }
    catch (Exception e) {
        e.printStackTrace();
     }
}
```

## 32-3 Utilizzo di una stored procedure con parametri di output

**Fonte:** http://msdn.microsoft.com/it-it/library/ms378108(v=sql.100).aspx

Se una stored procedure restituisce dati in uno o più parametri OUT, è possibile utilizzare il consueto driver JDBC per chiamare questo tipo di stored procedure ed elaborare i dati restituiti, utilizzando la classe **SQLServerCallableStatement**.

Quando si chiama questa forma di stored procedure col driver JDBC, si può utilizzare la chiamata **call** insieme al metodo **prepareCall** della classe **SQLServerConnection**.

La sintassi della sequenza di escape call con parametri OUT è la seguente:

Visual C#

```
{call procedure-name[([parameter][,[parameter]]...)]}
```

Per esempio, consideriamo la seguente procedura posta in un database di esempio Scuola:

Transact-SQL

```
CREATE PROCEDURE QuantiStudenti
   @Femmine int = 0 OUT ,
   @Maschi int = 0 OUT
AS
BEGIN
    SET @Femmine =  SELECT COUNT(*)
                    FROM Studenti AS S
                    WHERE S.Sesso = 'F';
    SET @Maschi =   SELECT COUNT(*)
                    FROM Studenti AS S
                    WHERE S.Sesso = 'M';
END
```

Per invocarla potremmo eseguire:

Visual C#

```
public static void invocaProcConParam (Connection x) {
    try {
        CallableStatement cstmt =
              x.prepareCall("{call dbo.QuantiStudenti (?, ?)}");
        cstmt.setInt(0, 0);
        cstmt.registerOutParameter (1, java.sql.Types.INTEGER);
        cstmt.registerOutParameter (2, java.sql.Types.INTEGER);
        cstmt.execute();
        System.out.println("Femmine: " + cstmt.getInt(1));
        System.out.println("Maschi : " + cstmt.getInt(2));
        }
    catch (Exception e) {
        e.printStackTrace();
     }
}
```

### 32-4 Utilizzo di una stored procedure con stato restituito

**Fonte:** http://msdn.microsoft.com/it-it/library/ms378371(v=sql.100).aspx

Non approfondito.

### 32-5 Utilizzo di una stored procedure con i conteggi di aggiornamento

**Fonte:** http://msdn.microsoft.com/it-it/library/ms378487(v=sql.100).aspx

Non approfondito.

# 33 Transazioni

### 33-1 Concetto di concorrenza

Per concorrenza si intende l'esecuzione contemporanea di due o più processi che possono condividere una stessa risorsa. In una base di dati la concorrenza si realizza quando più processi (attività di clienti o batch) elaborano i dati ed accedono quasi in parallelo ai dati archiviati. Il processo di concorrenza può causare problemi derivati dall'accesso contemporaneo alla risorsa, mettendo a rischio la congruenza e l'integrità delle informazioni.

Consideriamo un caso teorico a titolo di esempio con due processi che accedono a una stessa registrazione; per semplicità denominiamo R la locazione che contiene un dato, in particolare possiamo assumere che sia un valore intero e chiamiamo A e B i due processi.

Supponiamo che inizialmente la risorsa R contenga il valore 1.

Supponiamo anche che il codice dei processi sia il seguente:

| PROCESSO A | PROCESSO B |
|---|---|
| BEGIN<br>DECLARE X:<br>SET X = R;<br>SET X = X + 1;<br>SET R = X;<br>END; | BEGIN<br>DECLARE Y:<br>SET Y = R;<br>SET Y = Y + 1;<br>SET R = Y;<br>END; |

COME SI può notare i processi hanno un comportamento analogo che elabora le seguenti operazioni in sequenza:

1) lettura del valore di R e copia del valore in una variabile locale;

2) incremento della variabile locale;

3) scrittura del nuovo valore nella locazione R

In teoria ci si aspetta che se i due processi A e B vengono eseguiti contemporaneamente, ciascuno incrementi di +1 il valore di R e quindi, al termine di entrambi, R risulterà incrementato di +2. Se R inizialmente contiene il valore 1, al termine dell'esecuzione dei processi A e B dovrebbe contenere 3.

Tuttavia questa aspettativa non è sempre confermata e dipende da come sono gestiti i modi e i tempi di accesso alla risorsa R. Per comprendere il meccanismo si consideri questa sequenza temporale degli eventi:

1) il processo A legge il valore di R (1) X=1;

2) il processo B legge il valore di R (1) Y=1;

3) il processo A incrementa il valore di X X=2;

4) il processo B incrementa il valore di Y Y=2;

5) il processo A scrive in R il valore di X R=2;

6) il processo B scrive in R il valore di X R=2;

Come si nota, al termine dei due processi A e B, il valore della locazione R è stato incrementato solo di +1 e non di +2; in pratica l'elaborazione di un processo interferisce con quella dell'altro causando un errore nei risultati finali.

Il problema è causato dalla possibilità di accedere in modo concorrente alla locazione condivisa (generalmente un record, una tabella o un oggetto del database) senza una garanzia sul suo valore. Per risolvere questo problema, una delle strategie possibili è quella del blocco della risorsa (non entreremo nei dettagli di cosa significhi e come sia

implementato il blocco di risorse nei database e in SQL Server, ma è importante intuire i pericoli del problema e le strategie con cui si affronta la sua soluzione).

Se l'esecuzione di una certa attività può bloccare la risorsa, allora si costringe l'esecuzione concorrente dei processi ad essere parzialmente serializzata. Sempre a titolo di esempio supponiamo che esista una istruzione BLOCK K che blocca la risorsa K (sia in lettura che in scrittura). L'istruzione UNBLOCK K la libera dal blocco.

Sfruttando questa istruzione sarebbe possibile riscrivere il codice dei processi come segue:

COME SI può notare i processi hanno un comportamento analogo che elabora le seguenti operazioni in sequenza:

| PROCESSO A | PROCESSO B |
|---|---|
| BEGIN<br>DECLARE X;<br>BLOCK R;<br>SET X = R;<br>SET X = X + 1;<br>SET R = X;<br>UNBLOCK R;<br>END; | BEGIN<br>DECLARE Y;<br>BLOCK R;<br>SET Y = R;<br>SET Y = Y + 1;<br>SET R = Y;<br>UNBLOCK R;<br>END; |

Consideriamo adesso un altro problema. Supponiamo che i due processi intendano accedere entrambi a due risorse distinte, R ed S. Supponiamo che il processo A richieda prima il blocco della risorsa R e dopo della risorsa S; e supponiamo, invece, che il processo B richieda prima il blocco della risorsa S e dopo della risorsa R.

Il problema che si può generare in questa situazione è che il processo A ottenga e blocchi la risorsa R e il processo B ottenga e blocchi la risorsa S. A questo punto quando il processo A tenta di accedere alla risorsa S deve porsi in attesa (WAIT) ma lo stesso farà il processo B quando tenterà di accedere alla risorsa R. In questa situazione i processi A e B sono reciprocamente in attesa di risorse , vicendevolmente bloccate, che non saranno mai liberate, salvo che uno dei due sia forzato a rilasciarla ( e ricominciare da capo).

Quando due o più processi sono in attesa di eventi reciproci che ne impediscono la prosecuzione si parla di **STALLO** o **DEADLOCK**.

## 33-2 Concetto di transazione

Si è già discusso del fatto che un RDBMS deve ammettere l'accesso concorrente di più clienti alla base di dati. Si è anche discusso del fatto che le richieste possano inserire, modificare e cancellare dati nelle tabelle. Nel paragrafo precedente però si è accennato a due importanti problemi che si possono verificare nel caso di accessi concorrenti alle risorse: il rischio di esecuzioni errate a causa della concorrenza e il rischio di stallo per l'accesso concorrente alle risorse.

Come si intuisce è fondamentale che la gestione della base di dati deve garantire che l'esecuzione di sequenze di comandi contenenti tali operazioni deve sempre portare a situazioni congruenti (nessun errore a causa della concorrenza) e che siano eseguite tutte le operazioni relative alla sequenza oppure a non eseguirne alcuna.

Il concetto di transazione nasce proprio dall'esigenza di fornire una tecnica risolutiva per l'accesso concorrente alle risorse. Una transazione è uno strumento con cui i programmatori possono definire un'unità di lavoro (composta da una o più istruzioni) che deve essere completata nella sua interezza per essere considerata valida.

Nella letteratura canonica si annoverano quattro concetti fondamentali che definiscono una transazione:

**ATOMICITÀ** (***la transazione è indivisibile***) Tutte le modifiche sui dati causate all'interno di una transazione devono essere valide e trasmesse correttamente al database. Se anche una soltanto delle operazioni non giunge al successo allora tutte le altre devono essere annullate (il cosiddetto **ROLLBACK** della transazione).

**CONSISTENZA** (***il database deve restare consistente***) Finché i dati non sono trasmessi al database, il database deve rimanere in uno stato di consistenza e mantenere l'integrità su tutti i dati.

**ISOLAMENTO** (***le transazioni non interferiscono***) Qualsiasi modifica compiuta all'interno di una transazione deve restare isolata da qualsiasi altra modifica apportate da qualsiasi altra transazione concorrente.

**PERMANENZA** (***il database conserva gli effetti***) Se la transazione va a buon fine le modifiche compiute da essa saranno trasmesse al database che deve garantire che qualsiasi malfunzionamento di sistema (hardware o software) non possa causare la perdita di tali modifiche.

La transazione nasce per gestire l'accesso concorrente alle risorse. Diversamente dalla soluzione di blocco vista a titolo di esempio nell'introduzione del paragrafo, il programmatore non deve esplicitare se e quali risorse il processo deve bloccare: è il sistema che si fa carico di individuare le risorse coinvolte e di presidiare la politica dei blocchi.

Una transazione può essere utilizzata dal programmatore per gestire qualsiasi manipolazione sui dati che coinvolga istruzioni UPDATE, DELETE e INSERT. il programmatore dovrebbe evitare transazioni con sole SELECT sebbene il sistema provveda a evitare di eseguire query che facciano riferimento a risorse utilizzate in altre transazioni. Poiché la transazione è essenzialmente uno strumento di blocco delle risorse, assume importanza il concetto di stallo (deadlock) introdotto all'inizio del capitolo.

In effetti l'esecuzione concorrente di transazioni può causare situazioni in cui due differenti azioni sui dati pongano in reciproca attesa i processi coinvolti. Se ogni transazione attende che l'altra rilasci le sue risorse si può creare uno stallo (il deadlock o abbraccio mortale) che coinvolge le righe usate o le tabelle utilizzate ma anche, nei casi peggiori, l'intero database.

## 33-3 Blocco e completamento di una transazione

In T-SQL una transazione è definita con un comando di inizio **BEGIN TRAN** e con un comando conclusivo. Il comando conclusivo deve essere **COMMIT TRAN** oppure **ROLLBACK TRAN**. Le istruzioni devono essere racchiuse tra questi due comandi.

L'inizio della transazione impone che il sistema assicuri al processo le risorse (tabelle) necessarie e che le blocchi per essa. Durante l'elaborazione della transazione le risorse sono affidate al processo che esegue la transazione e infine la sequenza di istruzioni può terminare con successo o fallimento; se si intende far terminare la transazione con successo allora si deve eseguire una istruzione COMMIT TRAN che assicura le proprietà di permanenza dei dati. Se invece l'esecuzione del programma deve terminare con fallimento, allora occorre eseguire una istruzione ROLLBACK TRAN che assicura l'annullamento di tutte le operazioni compiute dalla transazione e ripristina lo stato iniziale del database garantendo la sua integrità.

Il comando COMMIT TRAN rende le modifiche effettuate sui dati definitive sul database e una volta eseguito non consente di ritornare alla situazione precedente alla transazione. Se invece si vogliono annullare tutte le modifiche fatte sui dati a partire dall'inizio di una transazione occorre utilizzare il comando ROLLBACK TRAN. Per riflettere sul significato dei comandi relativi alle transazioni, si esamini il seguente codice T-SQL:

Transact-SQL

```
SELECT 'Prima',IDStudente,Cognome,Nome
   FROM dbo.Studenti
   WHERE IDStudente = 3;
BEGIN TRAN CambiaNome
   UPDATE Studenti
      SET Nome = 'Viola'
      WHERE IDStudente = 3
COMMIT TRAN
SELECT 'Dopo', IDStudente,Cognome,Nome
   FROM dbo.Studenti
   WHERE IDStudente = 3;
```

I due comandi SELECT restano esterni alla transazione, e servono solo per visualizzare i dati prima e dopo della transazione. La transazione manipola i dati della tabella Studenti, per cui la transazione presumibilmente bloccherà questa tabella. Il comando COMMIT TRAN determina la scrittura sul database degli effetti del comando UPDATE confermando la sua esecuzione.

## 33-4 Annullamento di una transazione

Per riflettere sul significato del comando ROLLBACK TRAN, si esamini il seguente codice T-SQL:

Transact-SQL

```
SELECT 'Inizio',IDStudente,Cognome,Nome
   FROM dbo.Studenti
   WHERE IDStudente = 3;
BEGIN TRAN CambiaNome
   UPDATE Studenti
      SET Nome = 'Viola'
```

```
      WHERE IDStudente = 3
   SELECT 'Transitorio',IDStudente,Cognome,Nome
      FROM dbo.Studenti
      WHERE IDStudente = 3;
ROLLBACK TRAN
SELECT 'Conclusione', IDStudente,Cognome,Nome
   FROM dbo.Studenti
   WHERE IDStudente = 3;
```

Il risultato mostrerà che la SELECT all'interno della transazione mostra l'aggiornamento dei dati richiesto dal comando UPDATE ma la SELECT eseguita dopo il comando ROLLBACK rivela che la situazione è stata riportata allo stato originale. Chiaramente non ha senso utilizzare una transazione per un solo UPDATE ma questo esempio è molto utile per capire la modalità di funzionamento di tale meccanismo.

# PROGETTAZIONE DI DATABASE IN SQL SERVER 2008

# 34 Linee guida per la progettazione di un database

Oltre agli strumenti analitici concettuali e logici utilizzati nella progettazione di un database, assume particolare importanza uno studio della sua progettazione fisica, o implementazione e quindi stabilire delle scelte cruciali quali:

- la definizione della tipologia del database: transazionale o analitico;
- la determinazione delle misure di sicurezza e di accesso controllato dei clienti;
- la predisposizione di misure di replica e ripristino in caso di guasto.

## 34-1 Sistemi OLTP e OLAP

Generalmente la funzione principale è quella di memorizzare dati operazionali ovvero quei dati derivati da operazioni di carattere amministrativo (anagrafiche, registrazioni di eventi, elenchi e listini di prodotti e di servizi) e gestionale (dati di orari, verifiche, risultati, prestiti, prenotazioni, relazioni tra persone e oggetti).

Alcuni ambiti delle organizzazioni, tuttavia, hanno bisogno di esaminare masse di dati di lungo periodo, per verificare andamenti, statistiche, tendenze, anomalie, ciclicità, ecc. come, per esempio, lo studio dell'andamento delle vendite di un prodotto, l'analisi della correlazione e della distribuzione tra periodi cronologici ed eventi e simili indagini. Sono compiti tipici del marketing strategico, del management amministrativo, della pianificazione delle attività, ecc.

I **database operazionali** sono orientati a gestire il traffico di dati e delle rispettive operazioni (inserimenti, modifiche e cancellazione dei dati in tabella, per lo più). Tuttavia i rapporti sopra citati non sono agevolmente elaborati da server di questo tipo, quanto piuttosto da server dedicati preposti alla verifica dei dati di periodo. I dati storici saranno così confrontati e stimati in relazione ai dati opportunamente strutturati in altri database, detti data-warehouse.

I **data-warehouse** sono orientati all'elaborazione di dati di sintesi, che consentono di stimare le tendenze dei fenomeni di periodo, conoscenza utile alla scelta delle decisioni manageriali.

Si delineano così due modelli diversi di database:

- un modello operazionale, rivolto ad un aggiornamento sistematico, repentino e costante che le transazioni impongono;
- un modello analitico, orientato all'analisi dei dati aggregati e di sintesi, utile alle decisioni manageriali.

I due modelli sono talvolta contrapposti e vedono quindi il definirsi di due forme di database, talvolta entrambi impiegati per la stessa organizzazione.

il primo modello è quello dei sistemi OLTP (**O**n **L**ine **T**ransaction **P**rocessing); il secondo modello invece è quello dei sistemi OLAP (**O**n **L**ine **A**nalytical **P**rocessing).

## 34-2 OLTP

L'acronimo OLAP è stato coniato da E.F. Codd in un articolo del '94 quando individua una classe di prodotti informatici rivolti alla navigazione tra i dati (data surfing) per fornire supporto all'analisi dei dati anche con tecniche complesse e articolate.
OLTP (**Online Transaction Processing**) è una classe di programmi finalizzati a semplificare e gestire le transazioni elaborate da un server, in genere le immissioni dei dati e le operazioni di elaborazioni; per esempio il recupero delle informazioni in database. I server OLTP sono particolarmente impegnati in alcuni settori, tra cui quello bancario, del trasporto aereo, del commercio a distanza e di della grande distribuzione, e dei grossi gruppi di produzione industriale.

Nei sistemi odierni l'elaborazione di transazioni online coinvolge in maniera sempre più ampia il supporto per le transazioni di rete. I sistemi OLTP quindi si soffermano su caratteristiche come la gestione della concorrenza, la gestione del traffico dati in rete, la sincronizzazione tra i dati di organizzazioni collegate.

I sistemi OLTP richiedono che siano utilizzati servizi di database che assicurino elevate prestazioni ai client e eccellenti misure di affidabilità e integrità dei dati. Poiché i sistemi OLTP sono sottoposti a carichi di lavoro frequenti e concorrenti si deve prevedere un sistema di server fisici che possa migliorare replicazioni, ripristini e aggiornamenti repentini.

TERMINI COLLEGATI: Accesso dotazione Resource Control (RACF) , processore a 64 bit , Teraplex (Centro Integrazione Teraplex) , P/390 , VSAM (Virtual Storage Access Method) , in tempo reale, il sistema operativo (RTOS) , SuperZap , ISPF (Interactive System Facility Produttività) , la scalabilità , Remote Job Entry (RJE)

### 34-3 OLAP

OLAP (Online Analytical Processing) è elaborazione dei dati aggregati e di lungo periodo. I dati OLAP possono essere distinti e derivati dai dati OLTP della medesima organizzazione. Per evitare di gravare sul sistema OLTP, i sistemi OLAP possono essere separati fisicamente, come hardware e come database.

Un sistema OLAP viene utilizzato da clienti specifici, si prefigge l'elaborazione di dati statistici e mira all'estrazione di indici di correlazione e di tendenza con cui stilare relazioni di andamento e di previsione.

Un database OLAP è generalmente di dimensioni più contenute rispetto ad un OLTP (data-warehouse) della medesima azienda. I dati in ingresso possono essere ciclicamente importati direttamente da sistemi OLTP.

TERMINI COLLEGATI GLOSSARIO: colonna di database , NoSQL (Not Only SQL) , scarsità e la densità , database in memoria , denormalizzazione , cubo virtuale , Cassandra (Apache Cassandra) , classificazione dei dati

# 35 Sicurezza

Per sicurezza di un database si intende generalmente la protezione del sistema da accessi e utilizzi indesiderati o insicuri; l'elemento di affidabilità invece è mantenuto distinto dalla sicurezza e attiene l'ambito del recupero dei dati in caso di guasto.

La sicurezza di un database assume rilievo soprattutto in materia di custodia di dati sensibili (privatezza delle informazioni), di dati critici di missione (elementi centrali per l'organizzazione di riferimento) di protezione dei dati rispetto ai diritti di accesso. Un database privo di misure di sicurezza, si espone a situazioni dove i dati riservati possono essere accessibili da chiunque, a corruzione dei dati anche per imperizia del cliente, a situazioni di incoerenza dei dati rispetto alla realtà modellata.

SQL Server propone diverse misure orientate alla gestione della sicurezza dei dati:

- Strumenti di autenticazione: gestione degli accessi mediante **autenticazioni** anche sfruttando il sistema operativo Windows;
- Associazione di diritti d'uso: possibilità di associare capability (diritti) di utilizzo dei dati a utenze e gruppi; possibilità di creare **utenze, gruppi, login e ruoli** che abbiano espliciti livelli di accesso
- Funzionalità delle viste: illustrazione dei dati senza accesso diretto alla tabella, ma filtrati utilizzando viste che filtrano i dati per colonna e per criteri di valore;
- Applicazione di tecniche di **crittografia** sia del database che dei singoli oggetti contenuti, dei log e dei file.

Si è discusso della autenticazione al database già in fase di installazione e di utilizzo di SSMS; si è anche visto che esiste la possibilità di fare riferimento agli utenti custoditi dal sistema operativo Windows.

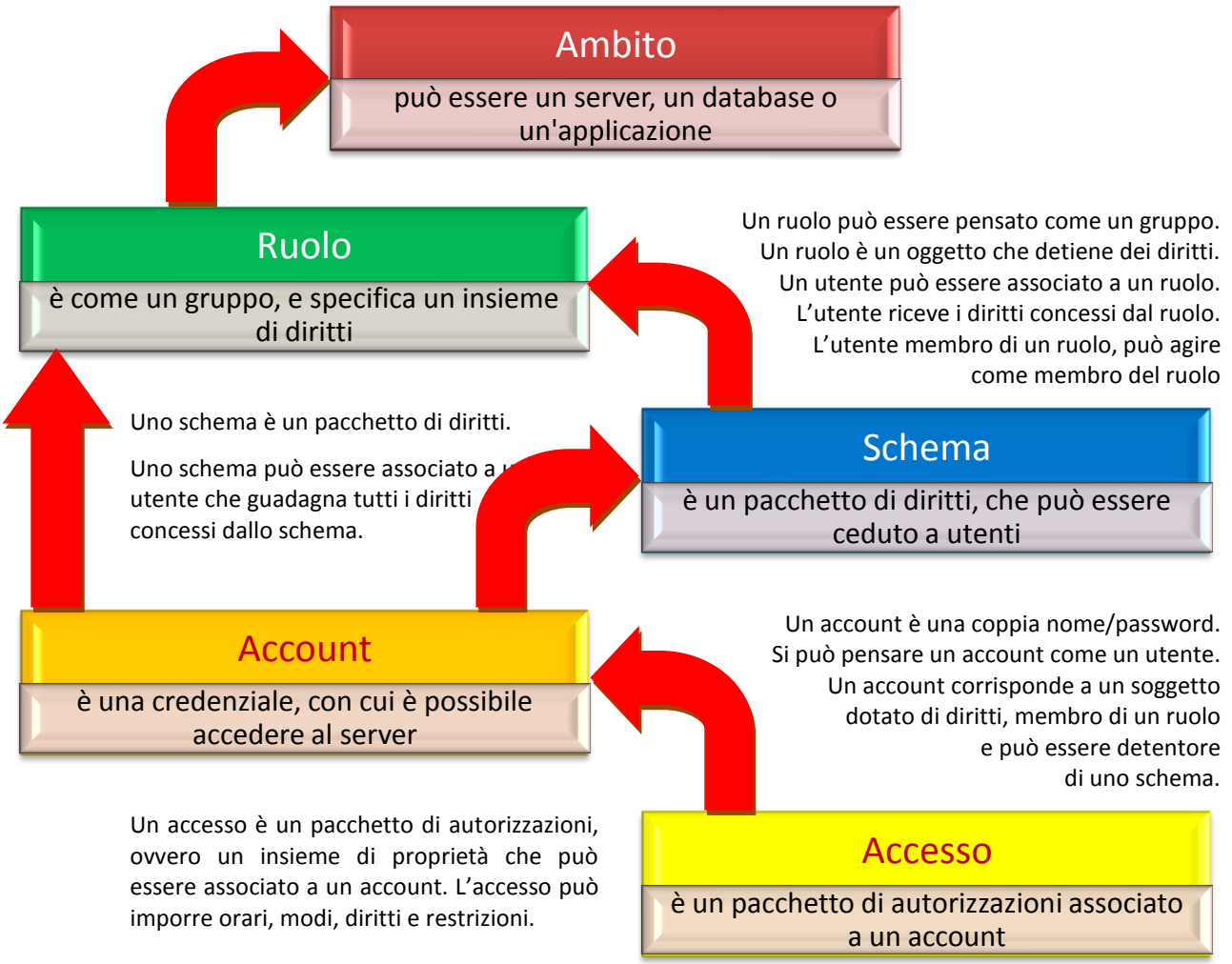
In questa sezione si intende affrontare la questione dei Login, dei Ruoli e degli Schemi.

Innanzitutto si deve precisare che l'autenticazione e la gestione delle credenziali è stata modificata da SS 2005 a S S 2008. Il sistema delle autenticazioni prevede diversi elementi coinvolti nella detenzione di diritti e autorizzazioni.

### 35-1 Elementi della autenticazione in SQL server

| | |
|---|---|
| Ambito | Per ambito s'intende l'area logica a cui si estendono le autorizzazioni considerate. In SS2008 sono:<br>▪ L'intera istanza del server, ambito denominato livello di server<br>▪ Un singolo database, ambito denominato livello del database<br>▪ Una specifica applicazione interna a un database, ambito denominato livello applicazione |
| Ruolo | Per ruolo s'intende un oggetto detentore di diritti, simile a un gruppo. Al ruolo possono essere iscritti diversi utenti, detti membri del ruolo. |
| Schema | Per schema s'intende un oggetto che impacchetta diritti, ma non è un ruolo. Un ruolo può agganciare schemi per ampliare i suoi diritti. Un account può essere associato a uno schema e riceve i suoi diritti. |
| Account | Un account è un'autorizzazione per connettersi al server. Un account può essere pensato come un utente. Un account (utente) può essere associato a schemi e/o diventare membro di ruoli. |
| Accesso | Un accesso è un pacchetto di diritti e restrizioni che corrisponde a un account. Un accesso può prevedere orari, modi, restrizioni che descrivono le capacità di accesso di un account. |

Gli elementi della sicurezza sono illustrati nel seguente prospetto:




Per iniziare a discutere gli elementi dell'autenticazione, conviene iniziare dagli Account, che possono essere immaginati come degli utenti. In effetti un account spesso coincide con un utente.

## 35-2 Introduzione agli Account

In SQL Server l'unico modo di connettersi è usando una credenziale per il Login. All'inizio, quando un database è appena creato, solo il creatore è proprietario e ha i diritti per accedere e compiere qualsiasi operazione su di esso.

Una pratica consueta è quella fare un login di amministratore su Windows e nel sistema operativo creare un gruppo Windows in cui agganciare gli account Windows che saranno poi ammessi a SQL Server.

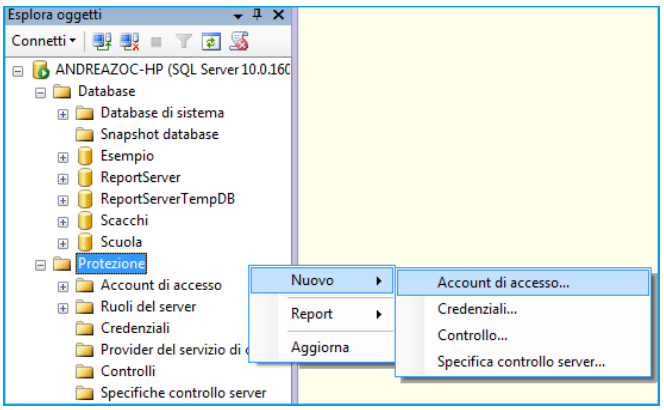
In questo modo è possibile, al completamento del database, agganciare il gruppo alla lista dei diritti di accesso di SQL Server, e gli utenti si autenticheranno in questo modo.

## 35-3 Creazione di un account

Per la connessione a SQL Server, un utente deve avere a disposizione un account di accesso di SQL Server. Per creare un account di accesso esistono diversi modi:

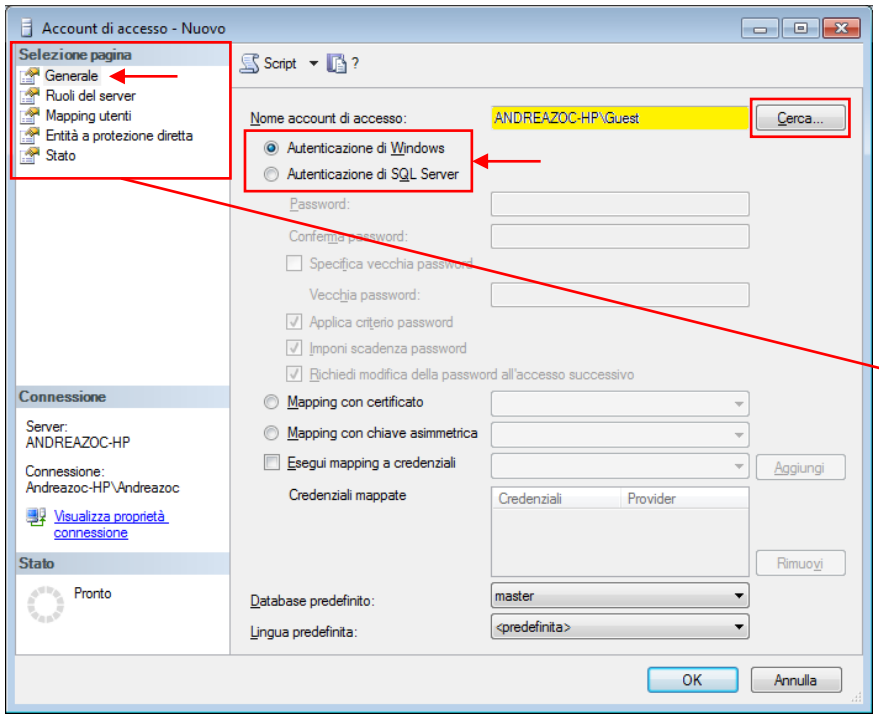
- In visuale, nel caso che SQL server utilizzi l'autenticazione di **Windows** (usando SSMS);
- In visuale, nel caso che SQL server utilizzi l'autenticazione di **SQL Server** (usando SSMS);
- Con comandi T-SQL.

Per creare un account in visuale si usa il menu contestuale del nodo Protezione (figura a lato).



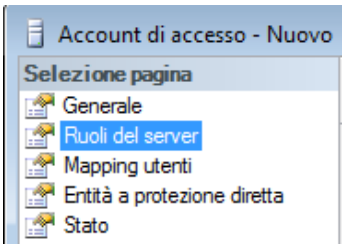
### *Per creare un account di accesso di SQL Server che utilizza l'autenticazione di Windows (SSMS)*

1. Aprire **Esplora oggetti** ed espandere il nodo dell'istanza del server in cui si desidera creare il nuovo account.
2. Col pulsante destro del mouse sulla cartella **Protezione**, scegliere **Nuovo** e quindi selezionare **Account di accesso**.
3. Nella pagina **Generale** immettere il nome di un utente di Windows nella casella **Nome account di accesso**.
4. Selezionare **Autenticazione di Windows**.
5. Confermare la creazione con clic sul pulsante **OK**.

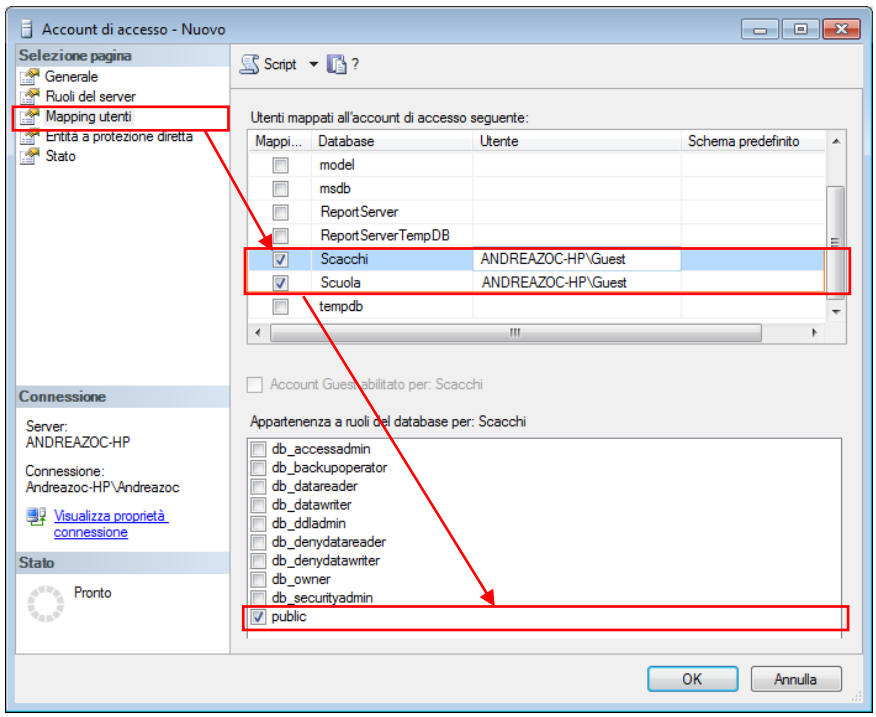



Si può osservare che il pulsante **Cerca** che compare sulla destra della finestra di dialogo consente di selezionare un utente registrato nel sistema operativo e immetterlo senza digitarne l'identificativo.

Sulla sinistra della finestra compare un elenco di opzioni:




I Ruoli del server saranno oggetto di studio successivo.




Per stabilire quali database siano accessibili all'utente che si sta definendo è possibile selezionare l'opzione **Mapping utenti** e aprendo la schermata riprodotta qui a sinistra.

Il primo riquadro della finestra permette di selezionare i DB a cui l'account ha accesso; nella figura l'account Guest viene abilitato all'accesso dei database Scacchi e Scuola.

Il secondo riquadro della finestra indica di quali ruoli è membro l'account corrente; nell'esempio l'account Guest è membro del ruolo (di database) public. Si osservi che quelli elencati sono ruoli per il database (nei prossimi paragrafi sarà esaminato questo concetto.

### *Per creare un account di accesso di SQL Server che utilizza l'autenticazione di SQL Server (SSMS)*

Il procedimento è identico al precedente, ma invece di selezionare l'opzione Autenticazione di Windows occorre selezionare invece Autenticazione di SQL Server:

1. Evocare il menu contestuale dela cartella **Protezione**, scegliere **Nuovo** e quindi selezionare **Account di accesso**.
2. Selezionare **Autenticazione di SQL Server**.

3. Nella pagina Generale immettere un nome per il nuovo account di accesso nella casella Nome account di accesso.
4. Immettere una password per l'account di accesso.
5. Selezionare le opzioni relative ai criteri password da applicare al nuovo account di accesso. In genere, è possibile ottenere una protezione maggiore selezionando l'opzione Applica criterio password.
6. Fare clic su OK.

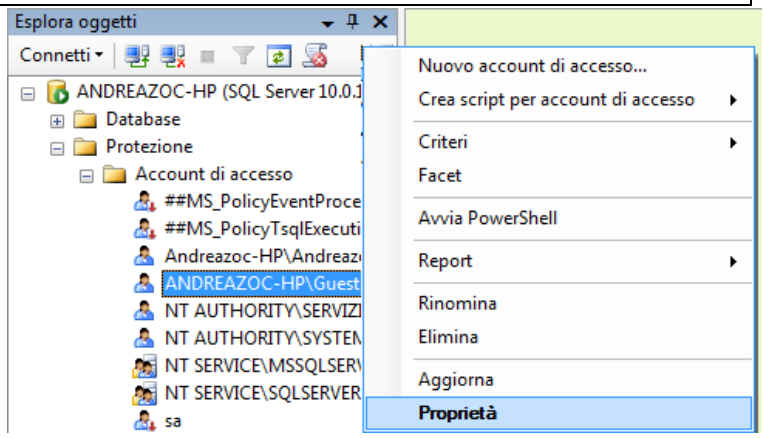
**Nota**

È importante osservare che la stessa guida di SQL server suggerisce di utilizzare l'autenticazione del sistema operativo Windows poiché offre maggiore protezione rispetto all'autenticazione di MSSS.

Se possibile, quindi, si consiglia di utilizzare l'autenticazione di Windows. In questa modalità viene utilizzata una serie di messaggi crittografati per autenticare gli utenti in SQL Server. Quando vengono utilizzati gli account di accesso di SQL Server, i nomi e le password di SQL Server vengono passati attraverso la rete, riducendone la sicurezza.

Se si desidera modificare un Account è possibile farlo selezionando il suo nodo e dal menu contestuale scegliere la voce Proprietà, come nella figura a lato.

La finestra che compare è del tutto analoga a quella della creazione di un Account.



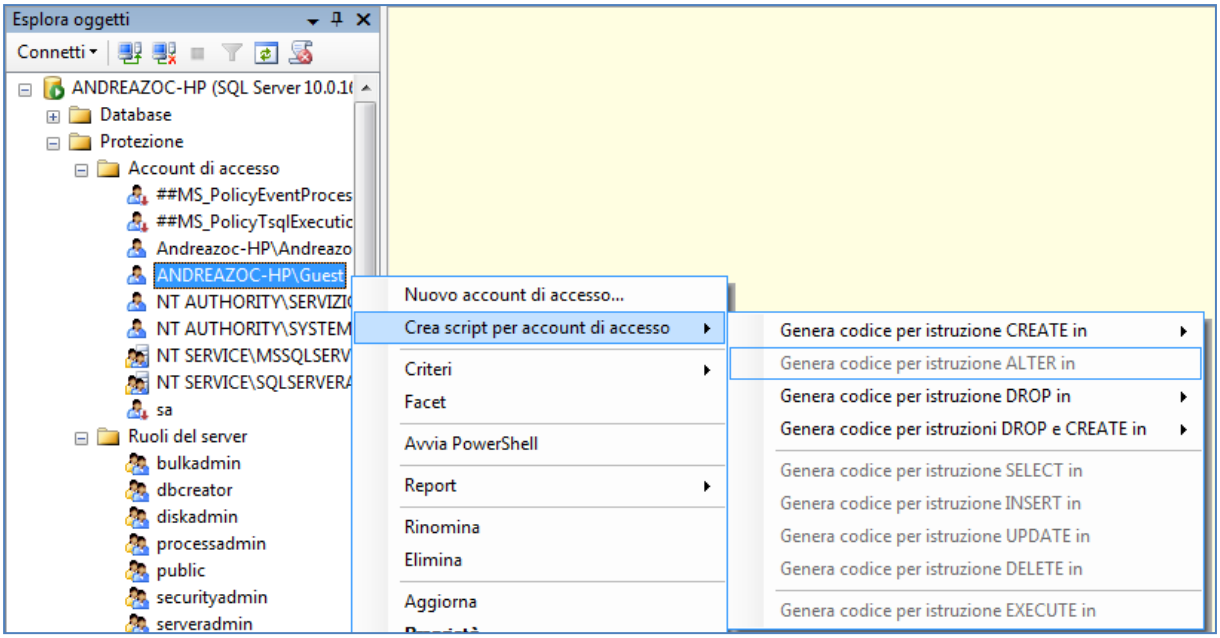

### *Comando T-SQL per creare un account di accesso che utilizza l'autenticazione di Windows*

Per creare un account di accesso, è anche possibile usare un comando T-SQL in un qualsiasi editor di Query:

**Transact-SQL**

```
CREATE LOGIN  nomeAccount  FROM WINDOWS;
GO
```

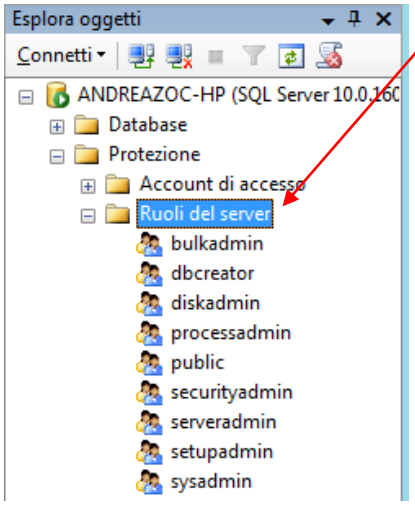
Per ottenere una sintassi estesa del comando è anche possibile richiedere a SSMS di visualizzare lo script corrispondente come di consueto dal menu contestuale, scegliendo Crea script e poi Genera codice per CREATE:



## 35-4 I Ruoli in generale

Come abbiamo visto all'inizio del capitolo, un account può essere membro di uno o più **ruoli**. Per comprendere intuitivamente il significato dei ruoli all'interno di SQL server possiamo pensarli come i **gruppi** analoghi a quelli previsti nel sistema operativo Microsoft Windows, cioè entità a cui un membro può essere associato. I ruoli sono un sistema costruito per semplificare la gestione delle autorizzazioni dei database. SQL Server fornisce diversi ruoli che sono delle entità di sicurezza al cui interno sono raggruppate altre entità. Un utente può essere associato a più di un ruolo. Ogni ruolo concede specifici diritti di diverso livello. All'interno di SQL Server esistono tre differenti tipi di ruoli:

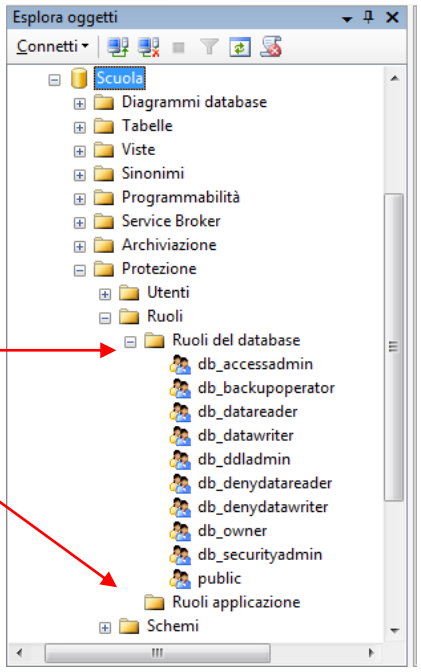
- RUOLI A LIVELLO SERVER, diritti a livello dell'intera istanza del server;
- RUOLI A LIVELLO DATABASE diritti a livello di un intero database creato dentro un'istanza del server;
- RUOLI A LIVELLO APPLICAZIONE diritti a livello di una specifica entità creata dentro un database.




I ruoli a livello di server sono visibili in Esplora oggetti espandendo il nodo Protezione e il nodo Ruoli del server.

I ruoli a livello di database e a livello di applicazione, invece, si possono rendere visibili (se esistono) selezionando un singolo database (es. **Scuola** del nostro esempio), il nodo Protezione e infine uno dei nodi

- Ruoli del database
- Ruoli applicazione




## 35-5 Ruoli a livello di server

I ruoli di livello server consentono ai rispettivi membri di riscuotere autorizzazioni estese a tutta l'istanza del server, quindi molto ampi. I ruoli predefiniti di server sono forniti soprattutto per motivi di compatibilità con le versioni precedenti di SQL server. Se possibile però, l'amministratore dovrebbe assegnare autorizzazioni più specifiche.

SQL Server annovera **nove ruoli predefiniti** del server. Le autorizzazioni concesse ai ruoli predefiniti del server non possono essere modificate. Solo da SQL Server 2012 sarà possibile creare ruoli del server definiti dall'utente e aggiungere autorizzazioni a livello di server a tali ruoli; in 2008 questo non è previsto.

Nei ruoli a livello di server è possibile aggiungere (includere come membri) entità a livello di server, ad esempio account di accesso di SQL Server, account di Windows e gruppi di Windows. Tutti i membri di un ruolo predefinito del server possono, a loro volta, aggiungere altri account di accesso allo stesso ruolo. In SQL server 2012 i membri dei ruoli del server definiti dall'utente non potranno però aggiungere altre entità del server al ruolo.

Nella tabella seguente sono elencati i ruoli predefiniti a livello di server e una descrizione sintetica.

| Ruolo | Descrizione |
|---|---|
| **bulkadmin** | I membri bulkadmin possono eseguire l'istruzione BULK INSERT. |
| **dbcreator** | I membri dbcreator possono creare, modificare, eliminare e ripristinare qualsiasi database. |
| **diskadmin** | Il ruolo diskadmin consente di gestire file su disco. |
| **processadmin** | I membri processadmin possono uccidere i processi in esecuzione in un'istanza di SQL Server. |
| **public** | Ogni Account di SQL Server appartiene al ruolo del server **public**. Se per un'entità del server non sono concesse o negate autorizzazioni specifiche per un oggetto a protezione diretta, l'account eredita le autorizzazioni concesse a public su tale oggetto. Non è possibile modificare l'appartenenza in public. Concedere autorizzazioni su un oggetto al ruolo public implica che l'oggetto sia disponibile per tutti gli account. |

| | |
|---|---|
| **securityadmin** | I membri securityadmin gestiscono gli account di accesso e le relative proprietà. I membri possono concedere, negare e revocare le autorizzazioni a livello di server e a livello di database se hanno accesso a un database. Questi membri possono anche reimpostare le password per gli account di accesso di SQL Server. |
| **serveradmin** | I membri serveradmin possono modificare la configurazione del server e arrestare il server. |
| **setupadmin** | I membri setupadmin possono aggiungere e rimuovere server collegati. |
| **sysadmin** | I membri sysadmin possono eseguire qualsiasi attività nel server. |

**Nota**

Nota sulla sicurezza

La possibilità di concedere l'accesso al Motore di database e di configurare autorizzazioni utente consente all'amministratore responsabile della sicurezza di assegnare la maggior parte delle autorizzazioni server.

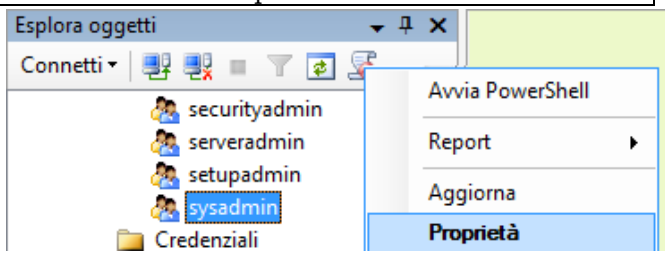
Il ruolo securityadmin deve essere considerato equivalente al ruolo sysadmin.

**Nota**

Il ruolo public viene implementato in modo diverso dagli altri ruoli.

È possibile, tuttavia, concedere, negare o revocare autorizzazioni da public.

Per accedere alle caratteristiche dello specifico ruolo è possibile evocare il menu contestuale sul rispettivo nodo e scegliere la voce Proprietà.




## 35-6 Ruoli a livello di database

L'ambito delle autorizzazioni dei ruoli a livello di database è l'intero database, quindi più specifico rispetto al server.

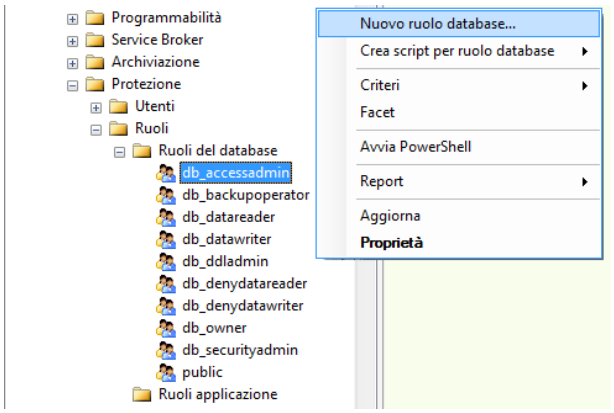
In SQL Server sono disponibili **due** tipi di ruoli a livello di database, che sono:

- i ruoli predefiniti di database (ruoli irremovibili e non creabili dagli utenti)
- i ruoli flessibili di database (ruoli che possono essere creati dagli utenti)

I ruoli predefiniti di database sono **due** e sono presenti in ogni database.

| Ruolo | Descrizione |
|---|---|
| **db_owner** | I membri db_owner possono gestire l'appartenenza al ruolo predefinito di database e possono anche aggiungere membri al ruolo predefinito di database db_owner |
| **db_securityadmin** | I membri db_securityadmin possono gestire l'appartenenza al ruolo predefinito di database. |

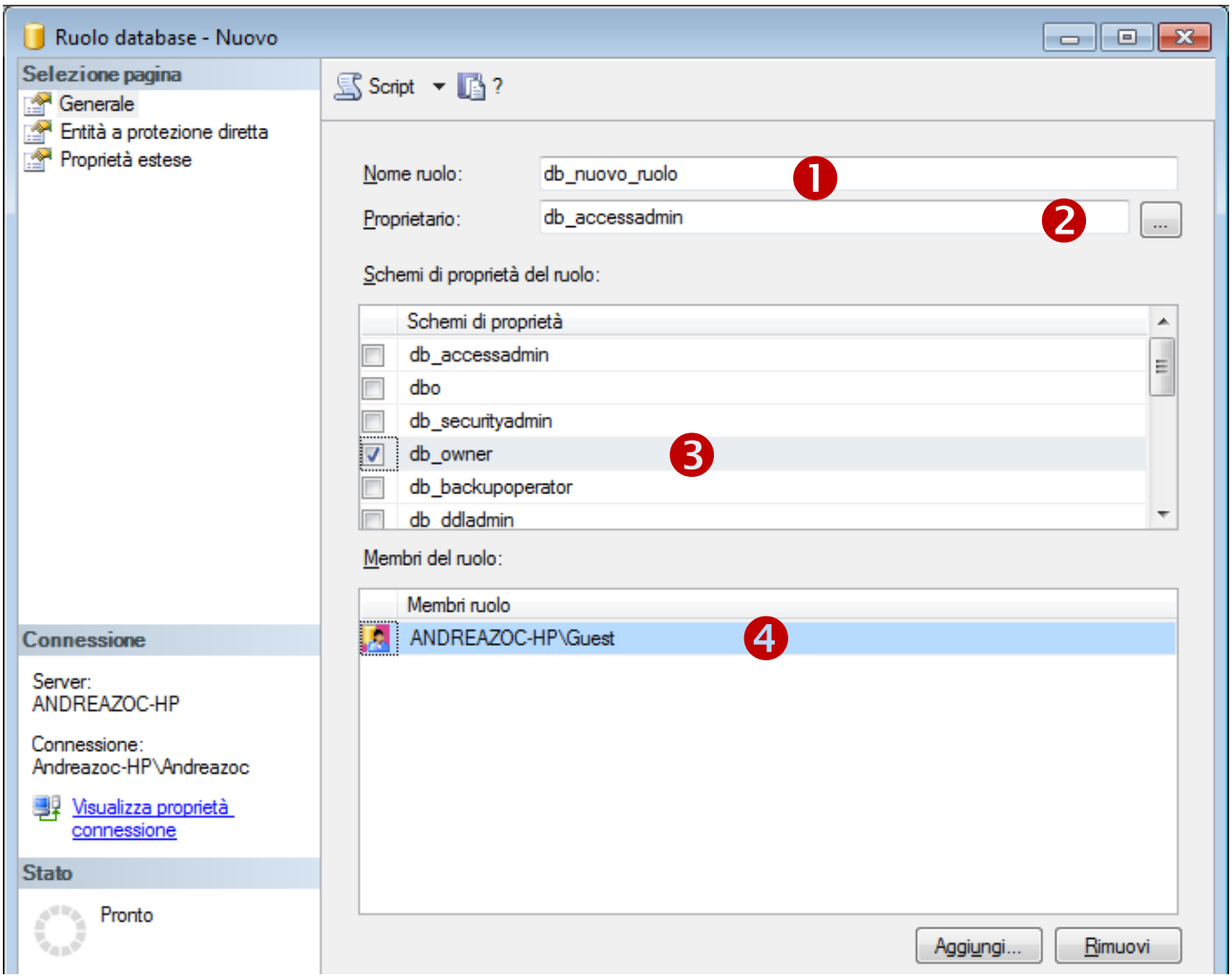
È possibile aggiungere qualsiasi account del database e altri ruoli SQL Server ai ruoli a livello di database. Tutti i membri di un ruolo predefinito del database possono aggiungere altri account di accesso allo stesso ruolo.




Per visualizzare e modificare le proprietà di un ruolo di database è sufficiente evocare il menu contestuale sul ruolo selezionato e scegliere **Proprietà**.

Per creare un nuovo ruolo, di livello database, scegliere invece la voce **Nuovo ruolo di database ...** . In questo caso appare una finestra delle proprietà in cui occorre specificare:

- Il nome del nuovo ruolo
- Il suo proprietario
- Quali schemi di proprietà gli vengono concessi
- Quali debbano essere i suoi membri




| | |
|---|---|
| ❶ | Il nome che individua il nuovo ruolo. Mediante il nome è possibile assegnarvi utenti e diritti. |
| ❷ | Il proprietario individua il soggetto che lo ha creato o che ne detiene le autorizzazioni di amministrazione. |
| ❸ | Ogni schema è un pacchetto di autorizzazioni cedibili a un utente o raggruppabili per un ruolo. |
| ❹ | Elenca i membri del ruolo; questa sezione consente di aggiungerne o rimuoverne. |

## 35-7 Gli schemi in SQL server

Gli schemi sono pacchetti di autorizzazioni indipendenti dagli Account e dai Ruoli.

È possibile associare uno schema ad un ruolo per conferire a quel ruolo le autorizzazioni specifiche dello schema.

È possibile anche associare uno schema ad un Account per conferirgli le autorizzazioni specifiche dello schema.

Un Account associato a uno Schema e membro di un Ruolo gode quindi delle autorizzazioni concesse allo Schema unite a quelle ereditate dal Ruolo; se inoltre il Ruolo è associato a uno Schema, ogni suo membro gode anche delle autorizzazioni di detto Schema.

Rispetto al comportamento degli schemi in SQL Server 2005 occorre evidenziare che c'è un notevole cambiamento. Gli schemi non equivalgono più agli utenti del database (cosa vera nel 2005); in SQL server 2008 ogni schema denota uno spazio di nomi distinto e indipendente dal database dell'utente creatore. In altre parole, uno schema è semplicemente un contenitore di oggetti. Lo schema può essere di proprietà di qualsiasi utente e la sua proprietà è trasferibile.

**Accesso** ← **Utente** ← **Schema** ← **Ruolo** ← **Entità**

**Accesso**
Possibilità di accedere al sistema, con specifiche su detto accesso

**Utente**
Account (password) che può diventare membro di un ruolo e associato a uno schema

**Schema**
Pacchetto di autorizzazioni integrate (es. modifica di una proprietà)

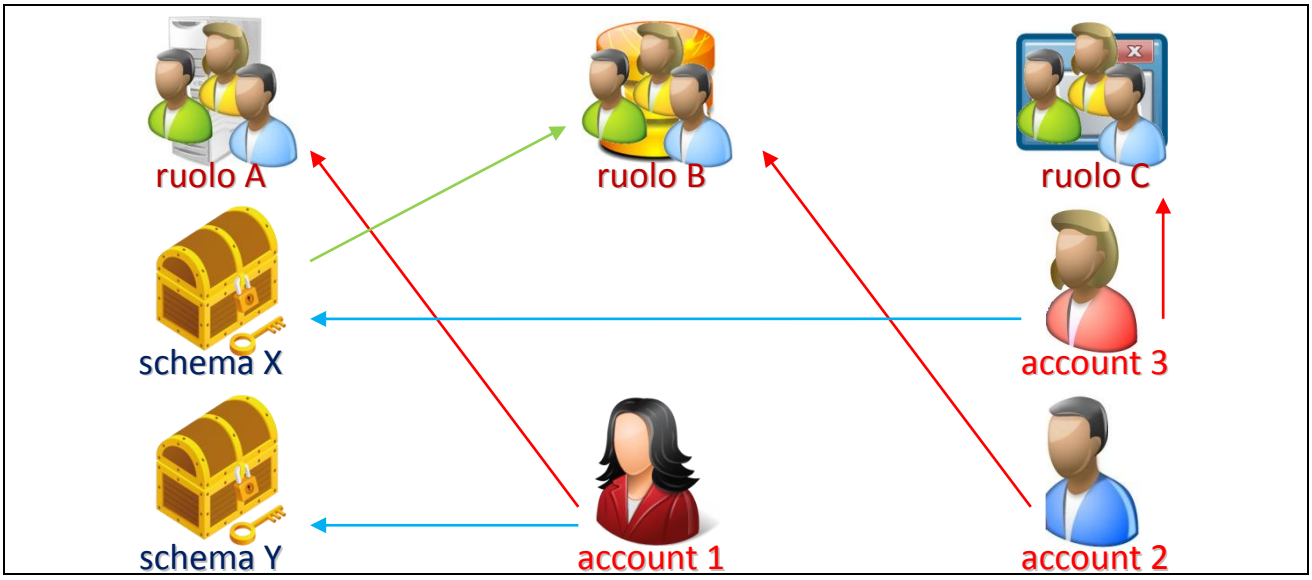
**Ruolo**
Uno dei ruoli (gruppi) dell'entità

**Entità**
Può essere:

- Server
- Database
- Applicazione

### 35-8 La sicurezza in SQL server

Uno dei concetti cruciali della sicurezza in SQL Server è che i proprietari di oggetti godono di autorizzazioni irrevocabili per amministrarli. Non è possibile rimuovere privilegi dal proprietario di un oggetto, né eliminare utenti da un database che contiene oggetti di cui sono proprietari.




Commento all'illustrazione soprastante:

1. L'account 1 riceve le autorizzazione del ruolo A e dello schema Y;
2. L'account 2 riceve le autorizzazione del ruolo B ma anche dello schema X, poiché X è associato a B;
3. L'account 3 riceve le autorizzazione del ruolo C e dello schema X;

La separazione della proprietà dagli schemi ha implicazioni importanti:

- La proprietà degli schemi e delle entità a protezione diretta definite a livello di ambito dello schema è trasferibile.
- È possibile spostare gli oggetti tra gli schemi.
- Un singolo schema può contenere oggetti di proprietà di più utenti del database.
- Più utenti del database possono condividere un singolo schema predefinito.
- Le autorizzazioni per schemi ed entità a protezione diretta contenute negli schemi possono essere gestite con una maggiore precisione rispetto alle versioni precedenti. Per ulteriori informazioni, vedere i GRANT.
- Uno schema può essere di proprietà di qualsiasi entità di database, inclusi i ruoli e i ruoli applicazione.
- È possibile eliminare un utente del database senza rimuovere gli oggetti in uno schema corrispondente.
- Il codice scritto per le versioni precedenti a SQL Server 2008 può restituire risultati non corretti, se in esso si presuppone che gli schemi siano equivalenti agli utenti del database.
- Le viste del catalogo progettate per versioni precedenti di SQL Server 2008, inclusa la vista sysobjects, possono restituire risultati non corretti.

# 36 Autorizzazioni in SQL Server

**Fonte:** http://msdn.microsoft.com/it-it/library/bb669084.aspx

Quando si creano oggetti di database, è necessario concedere in modo esplicito le autorizzazioni per renderli accessibili agli utenti. Ogni oggetto a protezione diretta gode di autorizzazioni che possono essere concesse a un'entità di sicurezza utilizzando istruzioni di autorizzazione.

### 36-1 Principio dei privilegi minimi

Lo sviluppo di un'applicazione basata su un approccio orientato a utenti con privilegi minimi rappresenta una parte importante di una strategia protettiva utile per fronteggiare i rischi per la sicurezza. Tale approccio assicura, infatti, che gli utenti seguano il principio dei privilegi minimi ed effettuino sempre l'accesso con account utente limitati. Le attività amministrative sono suddivise utilizzando i ruoli predefiniti del server, tuttavia l'utilizzo del ruolo predefinito del server **sysadmin** presenta particolari restrizioni.

È consigliabile attenersi sempre al principio dei privilegi minimi quando si concedono autorizzazioni agli utenti del database, ma anche a schemi e ruoli. Le concessioni spesso attengono a capacità di perseguire determinate attività.

Lo sviluppo e il test di un'applicazione per la quale è stato utilizzato l'approccio basato su account utente con privilegi minimi comporta maggiori difficoltà durante il processo di sviluppo. Per esempio, la scrittura di codice e la creazione di oggetti, risulta più agevole quando si è connessi come amministratore di sistema o proprietario del database rispetto a quando si utilizza un account con privilegi minimi. Lo sviluppo di applicazioni con un account dotato di privilegi maggiori può rendere, tuttavia, meno evidente all'amministratore quanto impatti il disporre di funzionalità ridotte eseguite da utenti con privilegi minimi che tentano di eseguire applicazioni che a volte richiedano autorizzazioni elevate. La concessione agli utenti di autorizzazioni eccessive per consentire loro di disporre di funzionalità elevate può però esporre il sistema a potenziali attacchi.

La progettazione, lo sviluppo e il test dell'applicazione per il cui accesso viene utilizzato un account con privilegi minimi consentono invece di applicare un approccio disciplinato alla pianificazione della sicurezza, eliminando sorprese sgradite ed impedendo di cedere alla tentazione di concedere privilegi elevati per risolvere rapidamente il problema.

Per il test è possibile utilizzare un account di accesso SQL Server anche se per la distribuzione dell'applicazione si intende utilizzare l'autenticazione di Windows.

### 36-2 Autorizzazioni basate sui ruoli

La concessione delle autorizzazioni ai ruoli anziché agli utenti consente di semplificare l'amministrazione della sicurezza. I set di autorizzazioni assegnati ai ruoli vengono ereditati da tutti i membri del ruolo. È più semplice aggiungere o rimuovere utenti da un ruolo anziché ricreare set di autorizzazioni distinti per i singoli utenti. I ruoli possono essere annidati, tuttavia un numero eccessivo di livelli di annidamento può comportare problemi di prestazioni. Per semplificare l'assegnazione delle autorizzazioni, è inoltre possibile aggiungere utenti a ruoli predefiniti del database.

È possibile concedere autorizzazioni a livello di schema. Gli utenti ereditano automaticamente autorizzazioni su tutti i nuovi oggetti creati nello schema, pertanto non è necessario concedere autorizzazioni quando vengono creati nuovi oggetti.

### 36-3 Autorizzazioni tramite codice procedurale

L'incapsulamento dell'accesso ai dati tramite moduli, ad esempio stored procedure e funzioni definite dall'utente, consente di definire un ulteriore livello di protezione per l'applicazione. È possibile impedire agli utenti di interagire direttamente con oggetti di database concedendo autorizzazioni solo a stored procedure o funzioni e negandole agli oggetti sottostanti, ad esempio le tabelle. In SQL Server è possibile eseguire questa operazione tramite il concatenamento della proprietà.

# 37 Gestione di progetti

Microsoft SS Management Studio offre due contenitori per la gestione di progetti di database quali script, query, connessioni di dati e file. Le due soluzioni sono denominate rispettivamente soluzioni e progetti. Gli oggetti raggruppati in uno di codesti contenitori sono denominati elementi.

Le soluzioni e i progetti contengono elementi che rappresentano gli script, le query, le informazioni di connessione e i file necessari per creare la soluzione di database. Questi contenitori possono essere utilizzati per:

- Implementare il controllo del codice sorgente nelle query e negli script.
- Gestire le impostazioni della soluzione nel suo complesso o per singoli progetti.
- Utilizzare Esplora soluzioni per una gestione dettagliata dei file mentre vengono considerati particolari elementi che compongono la soluzione di database.
- Aggiungere elementi utili a più progetti della soluzione o alla soluzione senza fare riferimento all'elemento in ogni progetto.
- Utilizzare file esterni indipendenti da soluzioni o progetti.

**Nota**

**Importante**

Questa caratteristica verrà rimossa a partire da una delle prossime versioni di Microsoft SQL Server.

Evitare di utilizzare questa caratteristica in un nuovo progetto di sviluppo e prevedere interventi di modifica nelle applicazioni in cui è attualmente implementata.

## 37-1 Progetti di SSMS

Un progetto è composto da un set di file e da metadati correlati, ad esempio le informazioni di connessione. I file di un progetto dipendono dal componente Microsoft SQL Server a cui il progetto fa riferimento. Un progetto SQL Server, ad esempio, può contenere comandi DDL (Data Definition Language) che dichiarano (creano) gli oggetti del database.

Una soluzione include uno o più progetti, oltre ai file e ai metadati che contribuiscono a definire la soluzione nel suo complesso.

Gli elementi contenuti nei progetti dipendono dal tipo di progetto e dall'eventuale utilizzo di SQL Server Management Studio. In questa sezione sono disponibili gli argomenti seguenti.

**Nota**

SQL Server Management Studio non supporta soluzioni o progetti di Microsoft Visual Studio o Microsoft Business Intelligence Development Studio.

# BACKUP & RECOVERY

# 38 Panoramica del backup (SQL Server)

## 38-1 Tipi e modi di backup

SQL server consente di eseguire dei backup periodici sia di un intero database di SQL Server sia di una sola parte di esso oppure anche di singoli gruppi di file (filegroup) di un database. Non consente tuttavia di creare un backup a livello di tabella.

**Nota**

Il backup e il ripristino di SQL Server possono essere utilizzati in tutti i sistemi operativi supportati, sia a 64 bit che a 32 bit. Per ottenere informazioni sui sistemi operativi supportati, consultare il sito Microsoft.

Per ogni categoria di backup sopracitata, SQL Server supporta due diversi tipi di backup:

- Backup completo
- Backup differenziale

Quindi è possibile eseguire almeno sei diversi ambiti di backup come mostrato nella seguente tabella:

| BACKUP | Database completo | Database parziale | Set di file |
|---|---|---|---|
| C Backup Completo | ❶ | ❸ | ❺ |
| D Backup Differenziale | ❷ | ❹ | ❻ |

C Un ***backup completo*** include tutti i dati di un ambito (**database completo, parziale o filegroup**), e include anche una parte di log sufficiente a consentire il recupero di tali dati.

D Un ***backup differenziale***, invece di includere tutti i dati, salva solo le modifiche dei dati più recenti. Un backup differenziale è basato sull'ultimo backup dei dati, detto ***base*** del backup differenziale (o anche base differenziale). Una base differenziale è l'ultimo backup completo di dati. Un backup differenziale include solo i dati che hanno subito modifiche dopo la base differenziale. In genere, i backup differenziali eseguiti entro un breve periodo di tempo in seguito ad un backup completo, sono più piccoli e più rapidi da eseguire rispetto alla creazione di un backup completo.

Poiché sono backup più rapidi e più piccoli, i backup differenziali sono consigliati se si possono svolgere frequentemente, riducendo così il rischio della perdita dei dati.

Se intercorre abbastanza tempo tra la base differenziale e l'esecuzione del backup differenziale, la quantità di dati da includere nel backup differenziale aumenta e questo rallenta sia la creazione che il ripristino del backup. Per questo, ogni periodo lungo è consigliabile eseguire un backup completo. Il nuovo backup completo sarà anche la nuova base differenziale.

Ogni backup dei dati include una parte del log delle transazioni sufficiente a consentire il recupero del backup fino alla fine dello stesso.

## 38-2 Backup parziali

In SQL Server 2005 sono stati introdotti i backup parziali (e i backup parziali differenziali). Un backup parziale salva tutti i dati specificati facoltativamente. I dati possono essere quelli individuati nel filegroup primario, in ogni filegroup di lettura/scrittura e in eventuali file o filegroup di sola lettura. Il backup parziale di un database di sola lettura contiene esclusivamente il filegroup primario.

## 38-3 Dispositivi di backup

I backup di SQL Server vengono creati nei dispositivi di backup, quali file su disco o supporti a nastro. È possibile aggiungere nuovi backup a tutti i backup esistenti in un dispositivo oppure sovrascrivere i backup esistenti.

## 38-4 Pianificazione dei backup

In SQL server è possibile pianificare l'esecuzione automatica dei backup a intervalli predefiniti. Con SSMS è possibile procedere alla creazione guidata di un piano di manutenzione.

La guida il linea informa che un'operazione di backup ha un effetto trascurabile sulle transazioni in esecuzione, quindi è possibile eseguire operazioni di backup durante il normale funzionamento del sistema. Durante un backup, tuttavia, possono coesistere delle restrizioni sulle possibili transazioni concorrenti.

Durante un'operazione di backup, SQL Server copia i dati direttamente dai file del database sui dispositivi di backup. I dati copiati non sono modificati e le transazioni in esecuzione durante il backup non vengono mai posticipate. Per tale motivo è possibile eseguire un backup di SQL Server con effetti minimi sui carichi di lavoro di produzione.

Nel corso dell'operazione di backup, infatti, è possibile eseguire la maggior parte delle operazioni, ad esempio istruzioni INSERT, UPDATE o DELETE; ma se si tenta di avviare un'operazione di backup durante la creazione o l'eliminazione di un file di database, l'operazione verrà rimandata fino al completamento dell'operazione di creazione o di eliminazione, oppure verrà annullata a causa di un timeout.

Le operazioni che non possono essere eseguite durante un backup del database o del log delle transazioni sono le seguenti:

- Operazioni di gestione dei file, come l'istruzione ALTER DATABASE con l'opzione ADD FILE o REMOVE FILE.
- Operazioni di compattazione di database (o di file), incluse le operazioni di compattazione automatica.
- Operazioni di creazione e di eliminazione di file di database (con relativo fallimento).

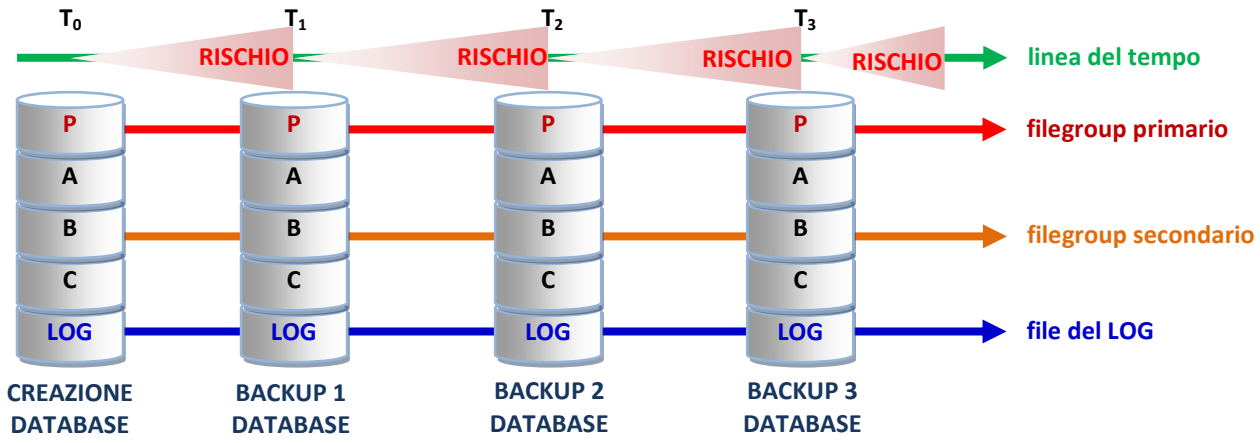
# 39 Backup completo del database

Un backup completo del database è un salvataggio dell'intero database. Nel salvataggio è inclusa una parte del log delle transazioni sufficiente a consentire un recupero (recovery) del database. I backup completi del database rappresentano il database al momento del completamento del backup.

Se si utilizza il modello di recupero con registrazione minima, dopo ogni backup, il database è esposto al rischio di perdita di dati nel caso si verifichi un'emergenza. Questo rischio aumenta a ogni aggiornamento fino al backup completo successivo, quando il rischio torna a essere zero e inizia un nuovo ciclo.

Con il modello di recupero con registrazione minima, il rischio di perdita dei dati cresce nel tempo che intercorre tra un backup e l'altro. Nella figura seguente viene illustrato il rischio di perdita di dati per una strategia di backup che utilizza solo backup completi di database.

Nella figura seguente è illustrato uno schema con tre backup effettuati in tre momenti diversi della vita del database:




Inizialmente il database è creato al tempo $T_0$; mentre è utilizzato il rischio di perdita dei dati per un crash del sistema aumenta (rappresentato da un cono rosso che aumenta di volume). Al tempo $T_1$ il database viene salvato con un backup completo, e il rischio è azzerato. Il database, quindi, è ancora usato e i dati si modificano, con il conseguente aumento del rischio. Il procedimento continua in questo modo, con salvataggi periodici di backup. È importante osservare che un database è formato da un gruppo di file primari, un file di log e diversi file secondari.

## 39-1 Comando BACKUP DATABASE

Il backup può essere eseguito da riga di comando. Questa modalità consente di scrivere procedure che poi verranno attivate a intervalli predefiniti dal sistema, oppure con invocazioni ad hoc dal DBA.

Transact-SQL

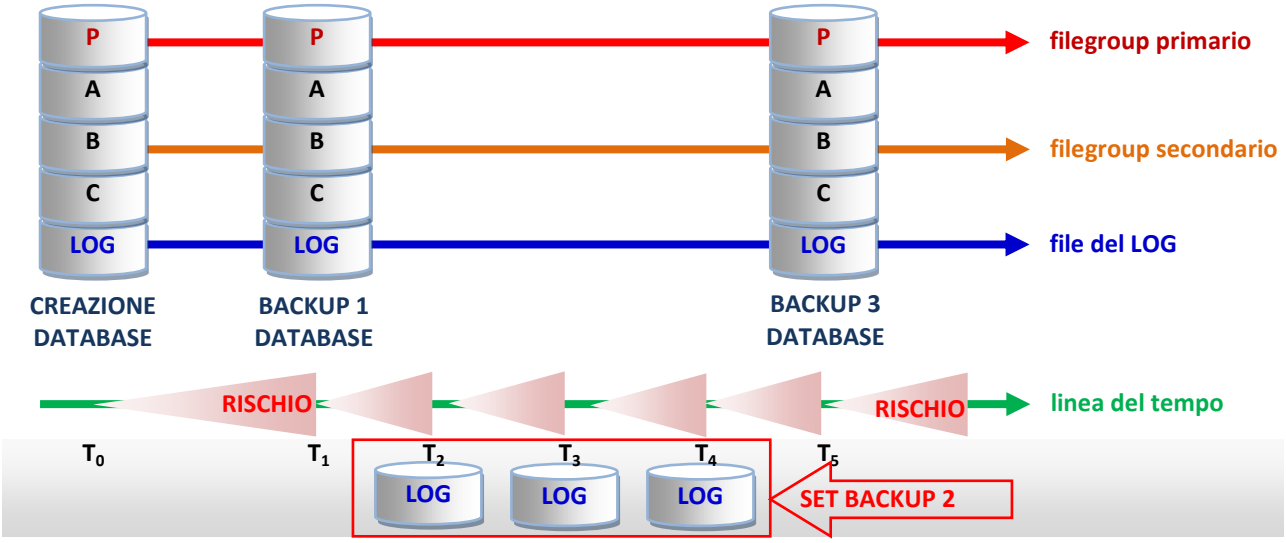
```
-- Back up del database Scuola
BACKUP DATABASE Scuola
     TO DISK = 'Z:\SS2008bk\Scuola_2012_04_27_ordinario.bak'
     WITH FORMAT;
GO
```

L'opzione **TO DISK** serve per specificare il nome completo (con percorso) del file fisico che il sistema usa per creare il nuovo backup.

L'opzione **WITH FORMAT** serve per specificare che il sistema deve sovrascrivere eventuali backup esistenti e creare un nuovo set di supporti.

## 39-2 Modello di recupero con registrazione completa

Nella figura seguente viene illustrata la strategia di backup più semplice possibile quando si utilizza il modello di recupero con registrazione completa.




Il seguente codice mostra un modo per richiedere l'esecuzione di un backup del log delle transazioni.

Transact-SQL

```
USE master;
-- Impostazione modello di recupero con registrazione completa per DB Scuola
ALTER DATABASE Scuola SET RECOVERY FULL;
GO
-- Back up del database Scuola (backup 1)
BACKUP DATABASE Scuola
     TO DISK = 'Z:\SS2008bk\Scuola_2012_04_27_full.bak'
     WITH FORMAT;
GO
-- Backup del log del database Scuola (backup 2).
BACKUP LOG Scuola
     TO DISK = 'Z:\SS2008bk\Scuola_2012_04_27_full.bak'
GO
```

In una situazione reale sarebbe necessario eseguire una sequenza regolare dei backup del log. In un simile caso il database di esempio Scuola deve essere impostato per l'utilizzo del modello di recupero con registrazione completa.

# 40 Backup parziali

I backup parziali non sono supportati in SQL Server Management Studio né nella Creazione guidata piano di manutenzione. Quindi non è possibile illustrare un procedimento visuale per questa modalità di backup.

Un backup parziale è simile a un backup completo del database, ma non contiene tutti i filegroup. Un backup parziale contiene invece tutti i dati del filegroup primario, tutti i filegroup di lettura/scrittura e i file di sola lettura specificati facoltativamente. I backup parziali sono utili quando si desidera escludere i filegroup di sola lettura. Il backup parziale di un database di sola lettura contiene esclusivamente il filegroup primario.

Il comando per richiedere l'esecuzione di un BACKUP parziale è il seguente:

Transact-SQL

```
BACKUP DATABASE database_name READ_WRITE_FILEGROUPS
[ , FILEGROUP = { logical_filegroup_name | @logical_filegroup_name_var }
     [ ,...n ]
]
TO   <dispositivo_backup> ;
```

ad esempio:

Transact-SQL

```
BACKUP DATABASE Scuola READ_WRITE_FILEGROUPS
     FILEGROUP = 'gruppo_file_012'
     TO   'Z:\SS2008bk\Scuola_2012_04_27_filegroup.bak'   ;
```

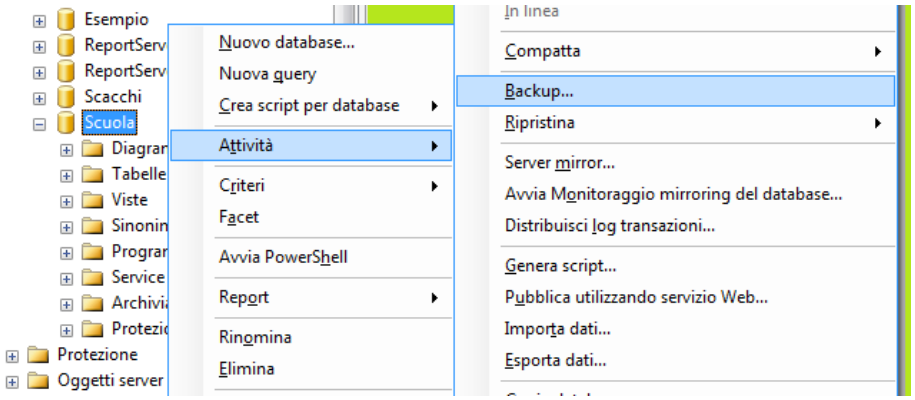
# 41 Backup di un database con Management Studio

In SSMS è prevista una modalità visuale per effettuare un backup completo di un database. La procedura è descritta nei seguenti passi per il database di esempio denominato Scuola:

Espandere il nodo **Database** e selezionare il database per cui si vuole compiere il backup.

Col pulsante destro del mouse sul database si deve evocare il menu contestuale da cui scegliere la voce Attività.

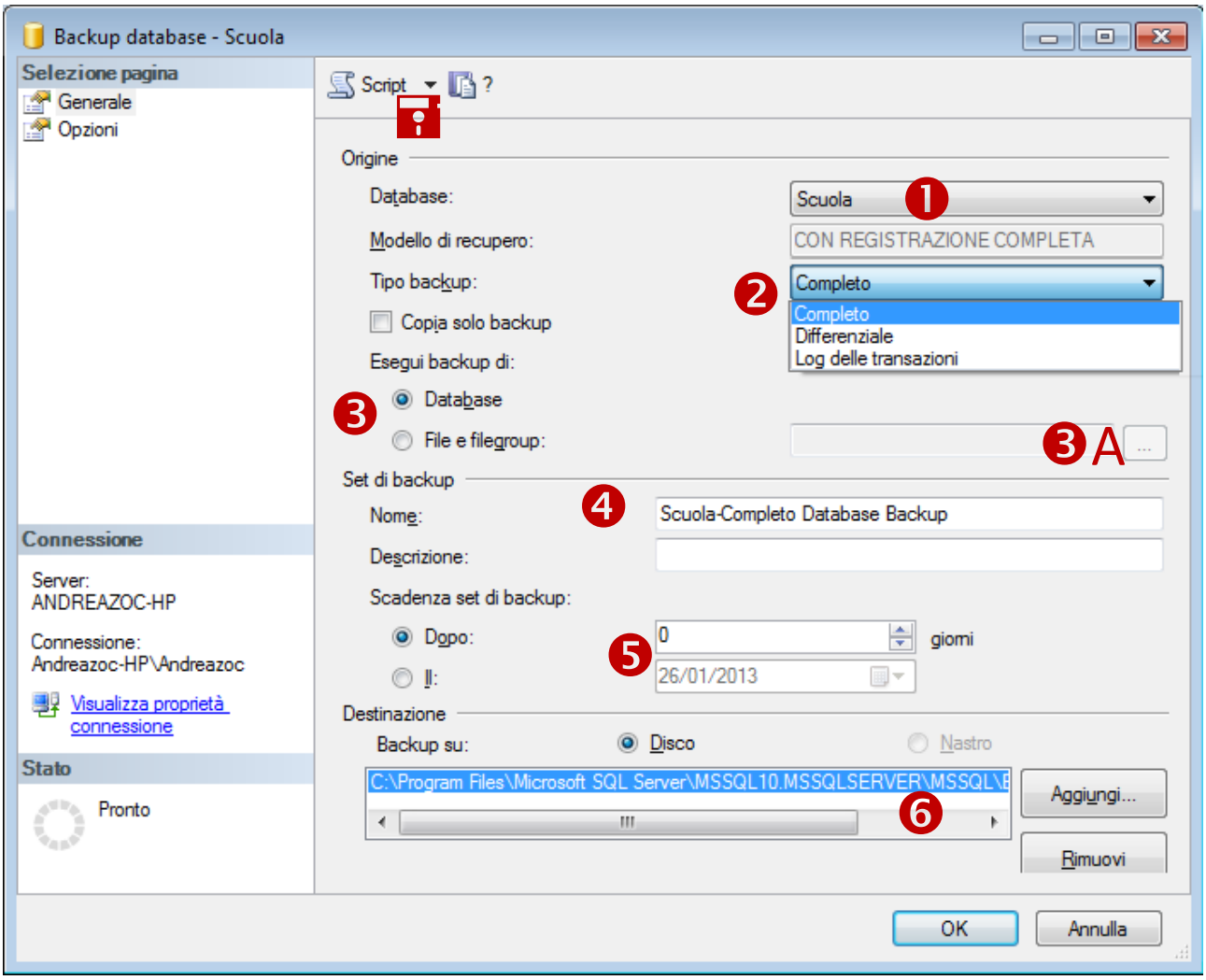
Dal sottomenu scegliere la voce Backup.




SSMS visualizza la finestra di dialogo Backup database, illustrata nella figura riportata di seguito. Nella finestra ci sono numerose opzioni che è possibile commentare:

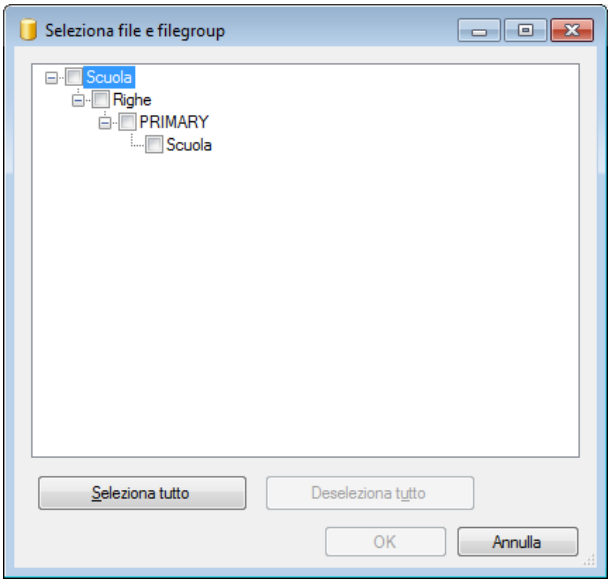
1. È indicato il database per cui si intende procedere col backup.
   Nell'esempio di riferimento è Scuola.
2. La casella indica il la modalità di backup: sono presenti le opzioni Completo, Differenziale e Log. Le opzioni corrispondono a quanto descritto nei paragrafi precedenti.
3. L'opzione consente di scegliere tra un backup del database oppure di file e filegroup.
   Scegliendo la seconda è possibile scegliere i file con il pulsante posto di fianco.
4. Nella casella si scrive il nome del backup (corrisponde alla clausola TO DISK del comando T-SQL.
   Spesso il backup è contraddistinto da un numero progressivo della forma AAAA_MM_GG_HH
5. Questa scelta indica il periodo di tempo per cui si considera valido il backup, trascorso il quale si suppone che l'ammontare di transazioni compiute renderanno eccessivo il rischio del backup.
6. Questa casella riporta il nome completo (con percorso) del file di backup, ovvero dove verrà salvato. Le opzioni appena sopra consentono, se possibile, di scegliere un nastro o un disco.

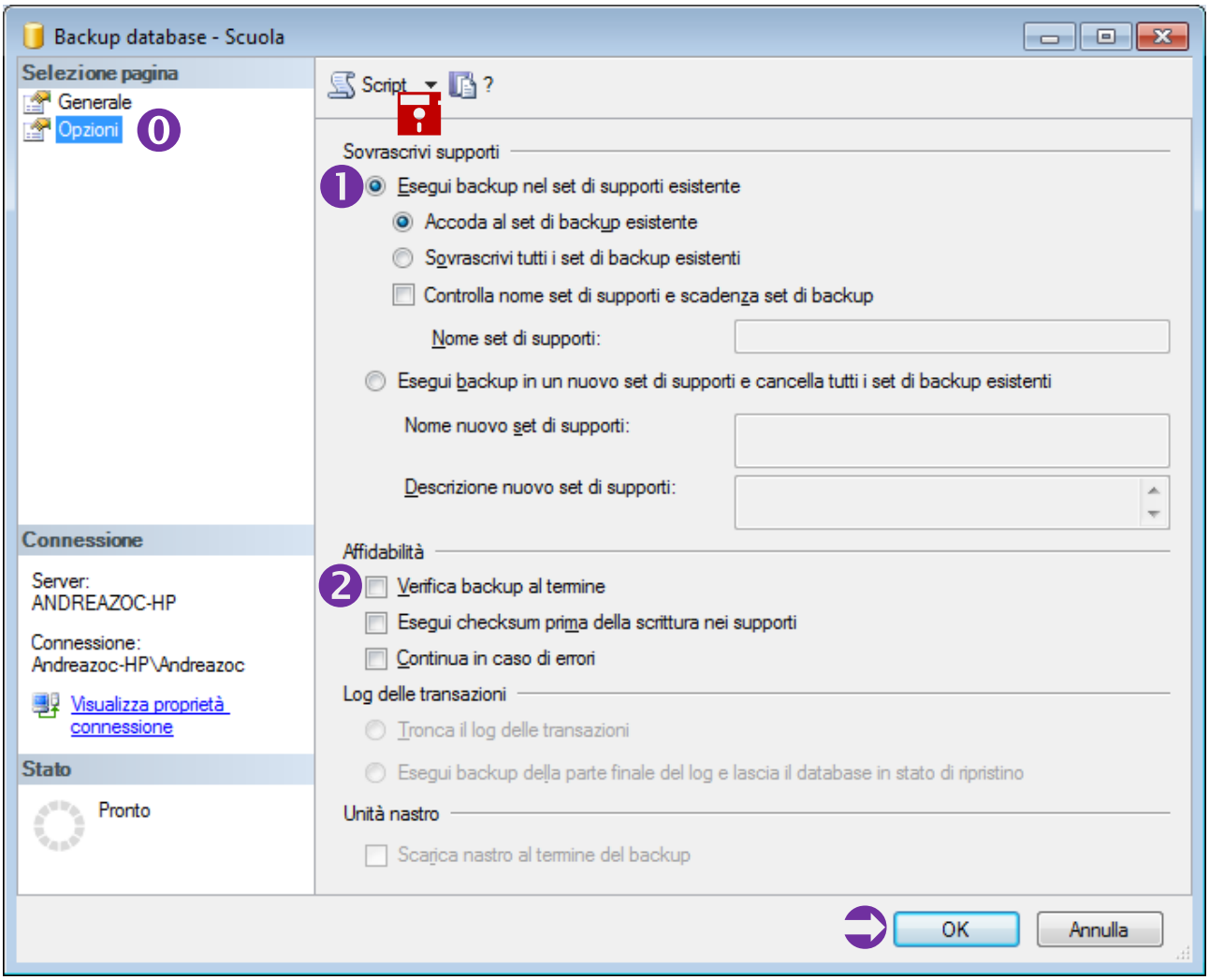
➔ Con OK si avvia l'operazione di backup.




L'opzione File e filegroup, consente di scegliere quali file si desidera coinvolgere nel backup.

Il pulsante con l'ellissi fa comparire una finestra (illustrata nella figura qui a lato) da cui si possono scegliere gli elementi oggetto del backup.






Sulla sinistra della finestra è possibile scegliere le opzioni e visualizzare la finestra illustrata nella figura proposta di seguito.

0 Voce delle opzioni, consente di accedere alla seconda sezione della finestra del backup.
Appare la finestra illustrata di seguito.

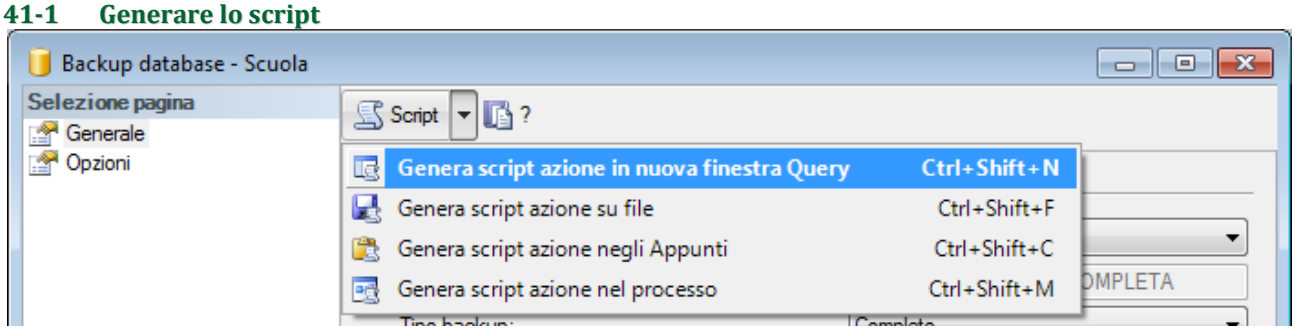
1 Scelta dei supporti. Accodare al backup esistente consente di mantenere il precedente backup. Sovrascrivi corrisponde alla clausola WITH FORMAT del comando CREATE BACKUP.

2 Affidabilità dell'operazione. Consente di attivare una verifica del backup (confronta la corrispondenza tra database e backup salvato), checksum (controlla i byte di parità del salvataggio), continua in caso di errori (forza il backup anche se si sollevano errori).

Con OK si avvia l'operazione di backup.

È possibile generare lo script T-SQL per il backup corrispondente facendo clic sul pulsante Script e selezionando una destinazione per lo script.

## 41-1 Generare lo script




Dal casella Script è possibile generare il codice T-SQL corrispondente alle opzioni contrassegnate. Lo script può essere scritto in una query, in un file, nella clipboard o nel processo del server.

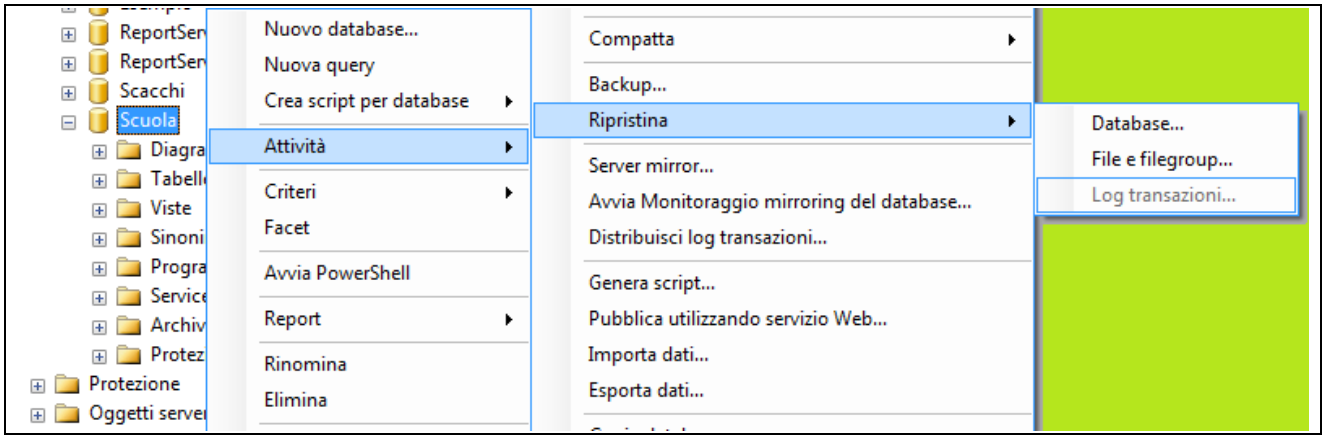
## 41-2 Back up differenziale con SSMS

È opportuno fare una riflessione sui backup differenziali. Ovviamente non è possibile eseguire alcun backup differenziale se non si è compiuto almeno un backup completo, da usare come base differenziale.

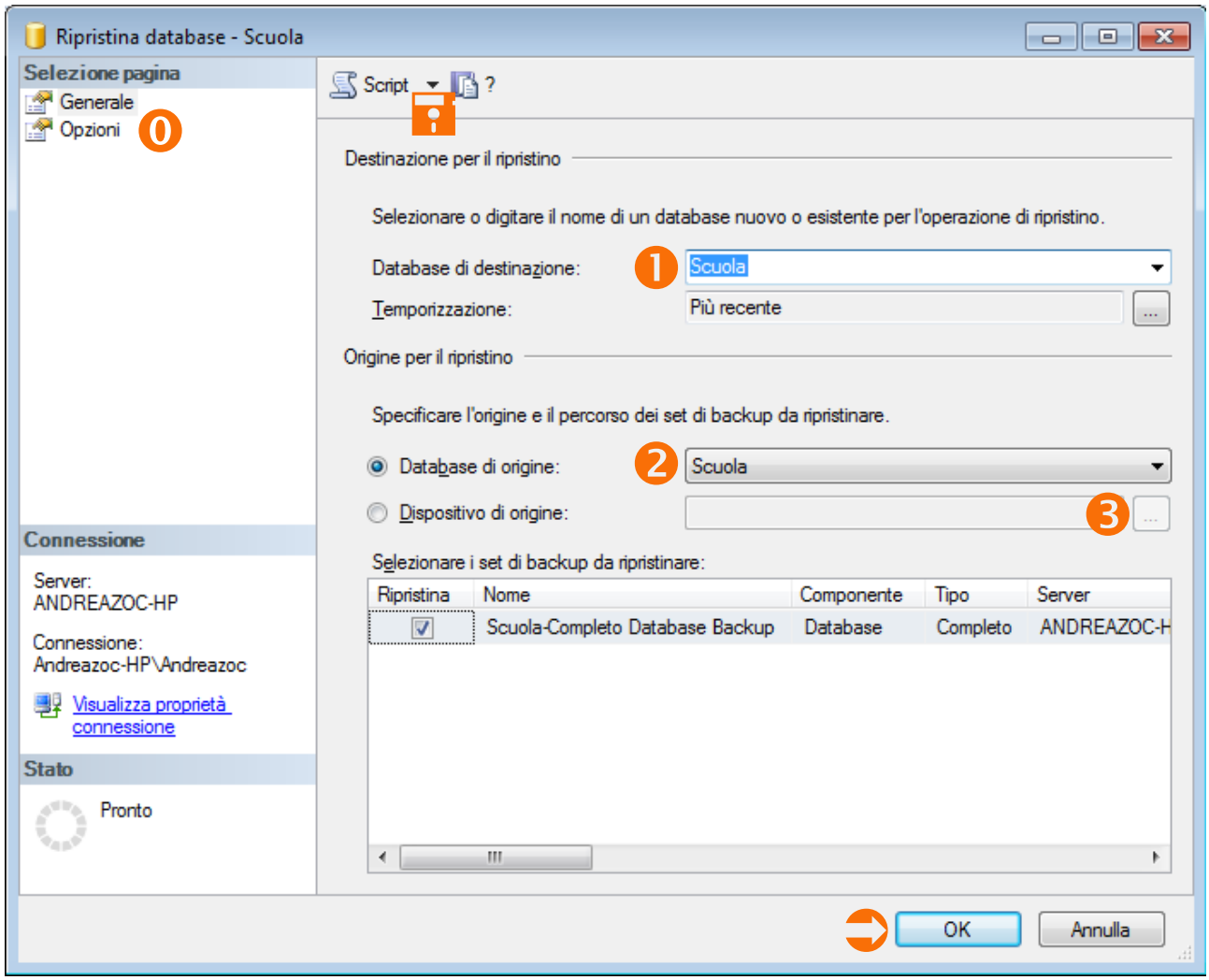
## 41-3 Ripristino (RECOVERY)

Analogamente al backup, SSMS prevede la modalità visuale per effettuare un ripristino di un database. La procedura è descritta nei seguenti passi per il database di esempio denominato Scuola.

1. Si deve espandere il nodo Database e selezionare il database per cui si vuole compiere il ripristino.
2. Col pulsante destro del mouse sul database si deve evocare il menu contestuale da cui scegliere la voce Attività.
3. Dal sottomenu scegliere la voce Ripristina e quindi una delle voci Database | Filegroup | Log.




Nell'esempio proposto si sceglie l'intero database Scuola.



0 Voce delle opzioni, consente di accedere alla seconda sezione della finestra del backup.

1 Scelta del database da ripristinare. In questo caso si desidera ripristinare il database scuola.

2 Scelta del backup cui si fa riferimento. Con la scelta Dispositivo di origine compare una finestra illustrata di seguito.

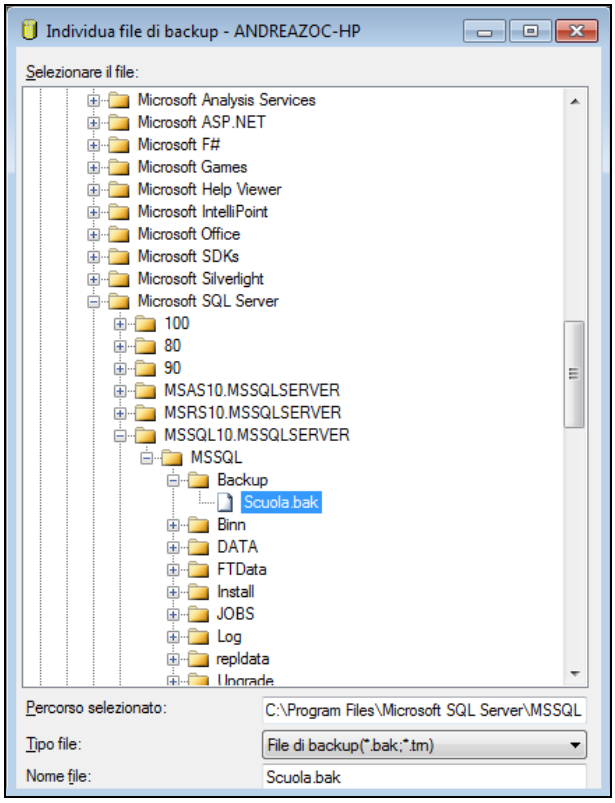
3 Individuazione del file di backup da cui eseguire il ripristino.
Si apre una finestra di dialogo e in seguito una finestra da cui scegliere il file (vedi sotto).

Con OK si avvia l'operazione di ripristino.

È possibile generare lo script T-SQL per il ripristino corrispondente facendo clic sul pulsante Script e selezionando una destinazione per lo script.

Se si sceglie l'opzione 3 compare una finestra di dialogo da cui successivamente è possibile aprirne una seconda come mostrato nella figura qui a lato.

La finestra consente di visualizzare i file di backup.




## 41-4 Dettagli del ripristino

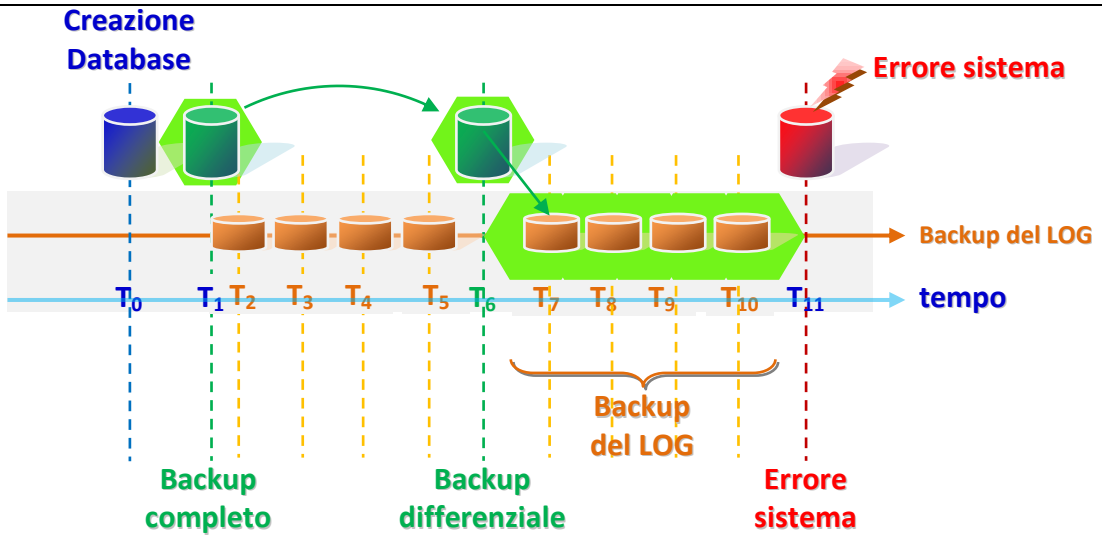
L'obiettivo di un ripristino completo del database è il ripristino dell'intero database. L'intero database non è accessibile agli utenti in linea per la tutta la durata del ripristino.

Se si utilizza il modello di recupero con registrazione completa, il database può essere ripristinato fino a uno specifico punto nel tempo. Il punto nel tempo può essere rappresentato dal backup più recente disponibile, da una data e un'ora specifiche o da una transazione contrassegnata.

Analogamente al backup è possibile chiedere lo script del RECOVERY dalla finestra visuale.

## 41-5 Recupero del database in seguito a un crash di sistema

In genere un recupero del database in seguito a un crash di sistema si esegue con una sequenza di comandi di RESTORE.



| | |
|---|---|
| **$T_0$** | Creazione del database iniziale. Da $T_0$ a $T_1$ il database viene usato. |
| **$T_1$** | Primo backup completo del database. Il rischio è azzerato. |
| **$T_2 – T_5$** | Backup del log, consente un ripristino in caso di errore del sistema. |
| **$T_6$** | Backup differenziale. Usa i backup del log. Di conseguenza i backup da $T_2$ a $T_5$ sono obsoleti. |
| **$T_7 – T_{10}$** | Backup del log, consente un ripristino in caso di errore del sistema. |
| **$T_{11}$** | Errore di sistema. Si deve procede al ripristino del database.<br>Si perdono solo le transazioni che intercorrono tra $T_{10}$ e $T_{11}$.<br>Per il ripristino si usano i backup effettuati nei tempi $T_{10}$, $T_9$, $T_8$, $T_7$, $T_6$, $T_1$.<br>Per il ripristino si usano cioè gli ultimi **backup del log**, il **backup differenziale**, il **backup completo**. |

Nella figura precedente è riportato un esempio di eventi relativi al backup del database.

A partire a ritroso dal crash del sistema, risultano:

- una sequenza di backup del log (T7 – T10);
- Un backup differenziale del database (T6);
- Un backup completo del database (T1).

Su questa situazione si esegue una sequenza di comandi volta al recupero del database che è riportata di seguito:

Transact-SQL

```
/* 1.  Ripristino da backup completo più recente
** senza recuperare il database */
RESTORE DATABASE Nome_database
   WITH NORECOVERY
/* 2.  Se sono presenti backup differenziali,
** ripristinare il più recente senza recuperare il database */
RESTORE DATABASE Nome_database
   FROM dispositivo_backup_differenziale
   WITH NORECOVERY
/* 3.  Ripristinare i log in sequenza con NORECOVERY a partire dal 1° backup
** del log delle transazioni creato dopo il backup appena ripristinato */
RESTORE LOG Nome_database
   FROM backup_log
   WITH NORECOVERY;
/* 4. Recuperare il database. */
RESTORE DATABASE Nome_database
   WITH RECOVERY
```

Per esempio:

Transact-SQL

```
USE master;
-- Creazione di un backup "coda_dei_log" --
BACKUP LOG Scuola
   TO DISK = 'Z:\SS_backup\Scuola_FullRM.bak'
   WITH NORECOVERY;
GO
-- Ripristino dell'intero database (dal backup 1) --
RESTORE DATABASE Scuola
  FROM DISK = ' Z:\SS_backup\Scuola_FullRM.bak'
  WITH FILE=1,
    NORECOVERY;

-- Ripristino del log regolare (dalla backup 2).
```

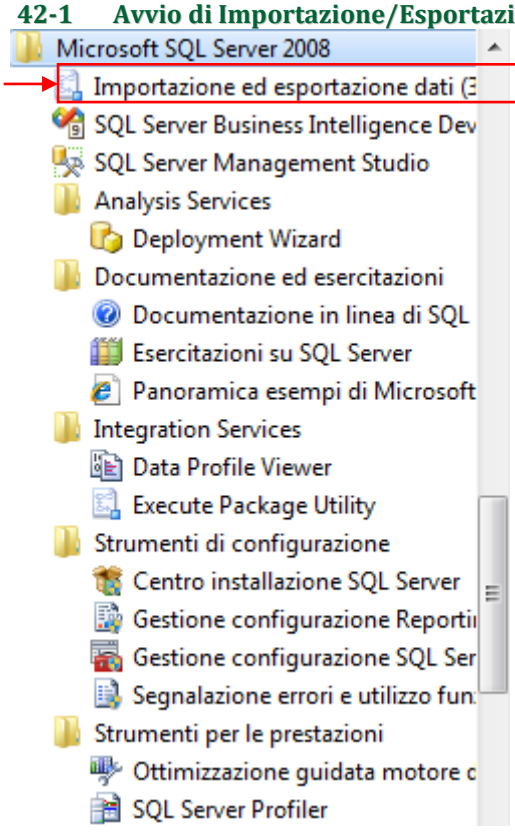
```
RESTORE LOG Scuola
  FROM DISK = 'Z:\SS_backup\Scuola_FullRM.bak'
  WITH FILE=2,
    NORECOVERY;

-- Ripristino dal backup "coda_dei_log" (dal backup 3).
RESTORE LOG Scuola
  FROM DISK = 'Z:\SS_backup\Scuola_FullRM.bak'
  WITH FILE=3,
    NORECOVERY;
GO

-- Ripristino del database
RESTORE DATABASE Scuola
 WITH RECOVERY;
GO
```

# 42 Importazione/Esportazione guidata SQL Server

Importazione/Esportazione guidata SQL Server costituisce il metodo più semplice per la copia di dati tra origini dati e per la costruzione di pacchetti di base.

In **SQL Server Express** non è disponibile la possibilità di salvare il pacchetto creato con la procedura guidata.

## 42-1 Avvio di Importazione/Esportazione di dati




Per avviare Importazione/Esportazione guidata SQL Server

- Fare clic sul menu **Start**, scegliere **Tutti i programmi**, **Microsoft SQL Server**, quindi **Importazione ed esportazione dati**.

— oppure —

- In SQL Server Data Tools (SSDT) fare clic con il pulsante destro del mouse sulla cartella **Pacchetti SSIS** e quindi fare clic su **Importazione/Esportazione guidata SSIS**.

— oppure —

- In SQL Server Data Tools (SSDT) scegliere **Importazione/Esportazione guidata SSIS** dal menu **Progetto**.

— oppure —

- In SQL Server Management Studio connettersi al tipo di server Motore di database, espandere Database, fare clic su un database con il pulsante destro del mouse, scegliere **Attività**, quindi **Importa dati** o **Esporta dati**.

— oppure —

- In una finestra del prompt dei comandi eseguire DTSWizard.exe, disponibile in C:\Programmi\Microsoft SQL Server\100\DTS\Binn.

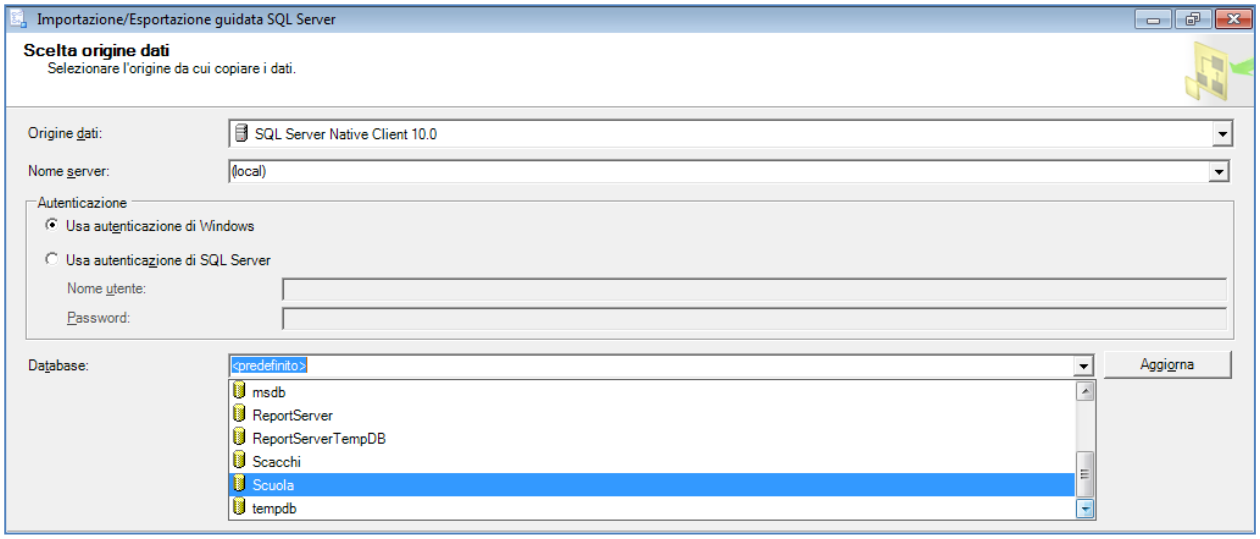
Microsoft propone anche un video dimostrativo sull'utilizzo dell'Importazione/Esportazione SQL Server che crea un pacchetto per esportare dati da un database di SQL Server verso un foglio di calcolo di Microsoft Excel, nel seguente link: http://go.microsoft.com/fwlink/?LinkId=131024 (esportazione di dati di SQL Server in Excel, video di SQL Server).

## 42-2 Procedura visuale di Importazione/Esportazione di dati

Per importare o esportare i dati utilizzando l'Importazione/Esportazione guidata SQL Server

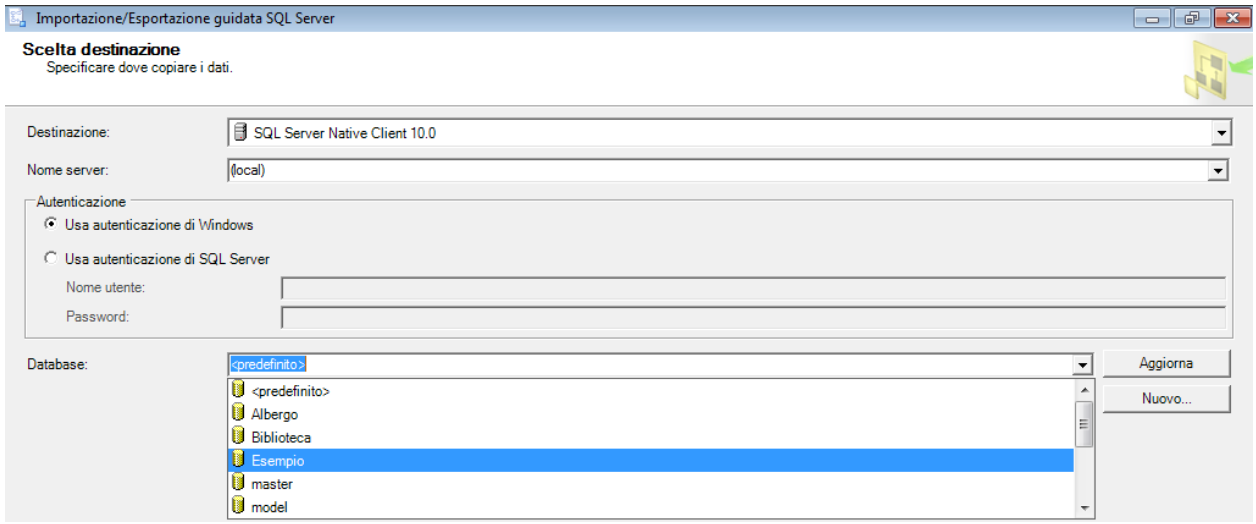
1. **Avviare l'Importazione/Esportazione guidata SQL Server.**


2. **La prima pagina della procedura guidata chiede di selezionare un'origine dei dati.**




Tra i database che possiamo scegliere troviamo anche il database di esempio denominato Scuola, che è utilizzato anche per questo caso.

3. **La seconda pagina della procedura guidata chiede di selezionare una destinazione dei dati.**

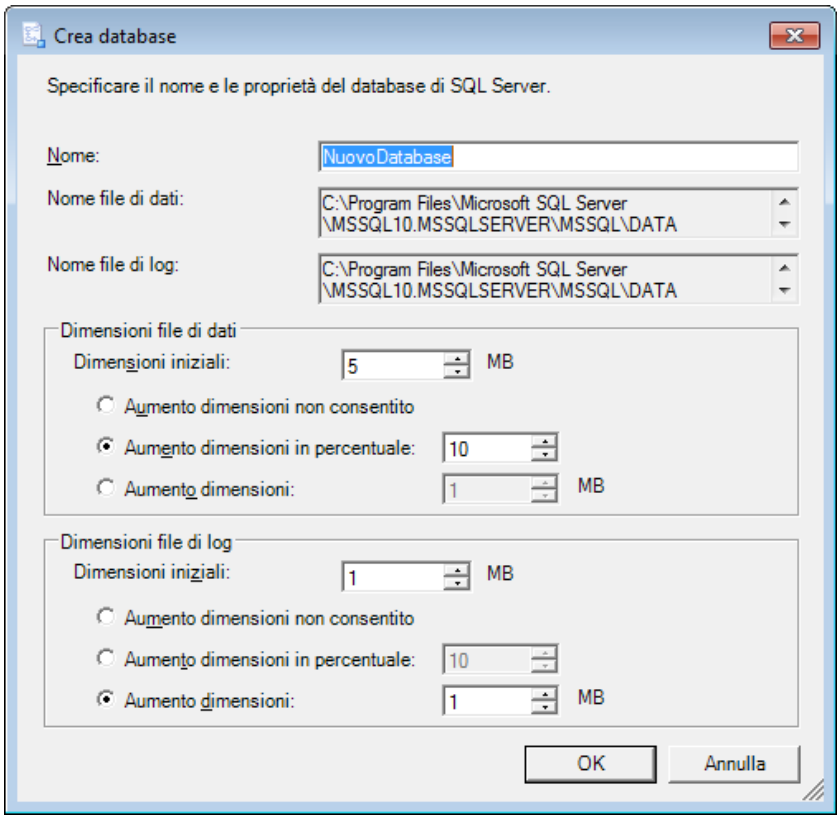



Le origini dati disponibili includono tutti i provider di dati .NET Framework, i provider OLE DB, i provider SQL Server Native Client, i provider ADO.NET, Microsoft Office Excel, Microsoft Office Access e l'origine file flat. A seconda dell'origine, è necessario impostare opzioni quali la modalità di autenticazione, il nome del server, il nome del database e il formato del file.

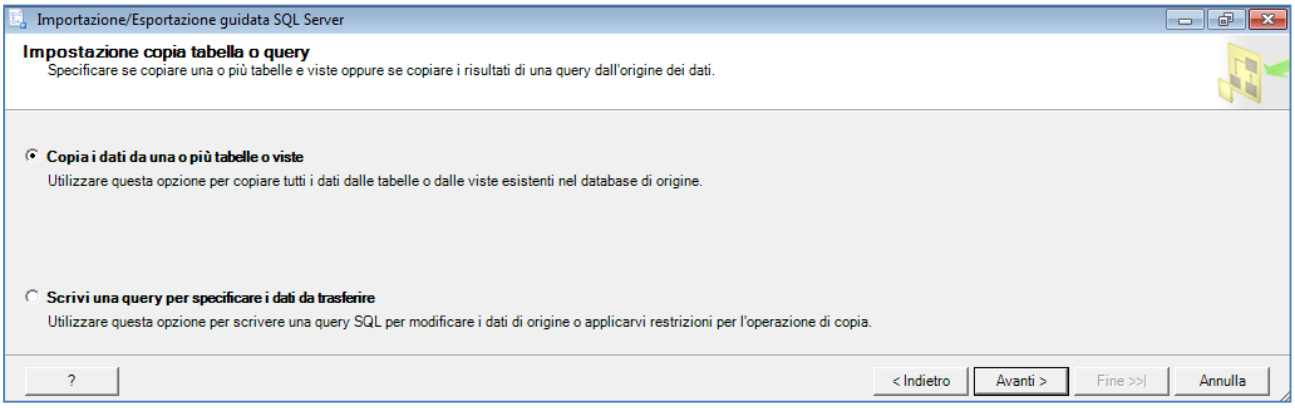
Le destinazioni dei dati disponibili includono i provider di dati .NET Framework, i provider OLE DB, SQL Server Native Client, Excel, Access e la destinazione file

Tra le possibili destinazioni dei dati è possibile scegliere un nuovo database, con un clic sul pulsante Nuovo della finestra precedente.

Appare una finestra simile a quella illustrata qui a lato che richiede un nome per il nuovo database e le opzioni di costruzione.




4. **Impostare le opzioni per il tipo di destinazione selezionato.**



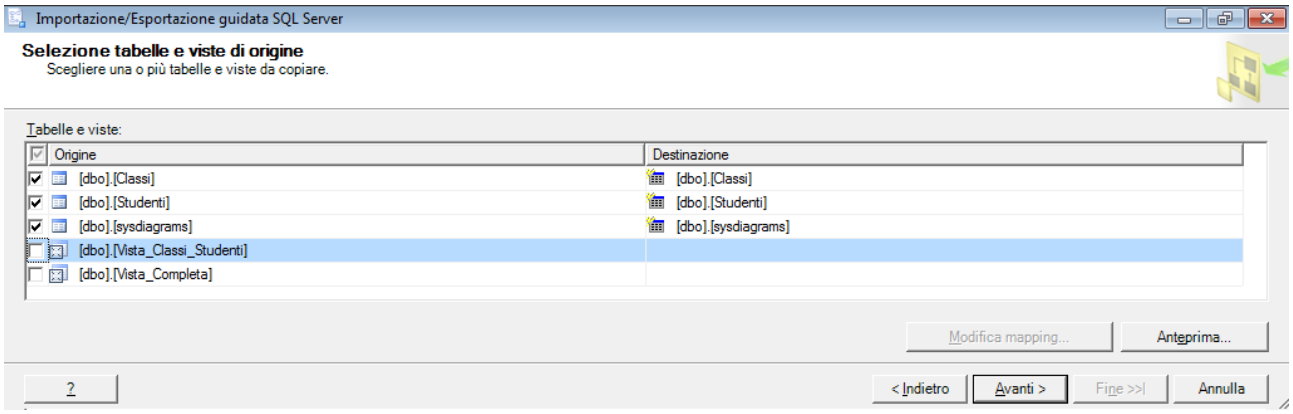

Se la destinazione è un database di SQL Server, sarà possibile:

- Indicare se creare un nuovo database e impostarne le proprietà.
- Selezionare se copiare i dati da tabelle o viste oppure copiare risultati di query.

La prima opzione consente di copiare i dati dalle tabelle o dalle viste; è una buona scelta per trasferire tutti i dati di un database o di parte di un database.

La seconda opzione consente di specificare con una nuova query i dati da trasferire; è una scelta opportuna per costruire dei report o comunicare informazioni per terze parti.

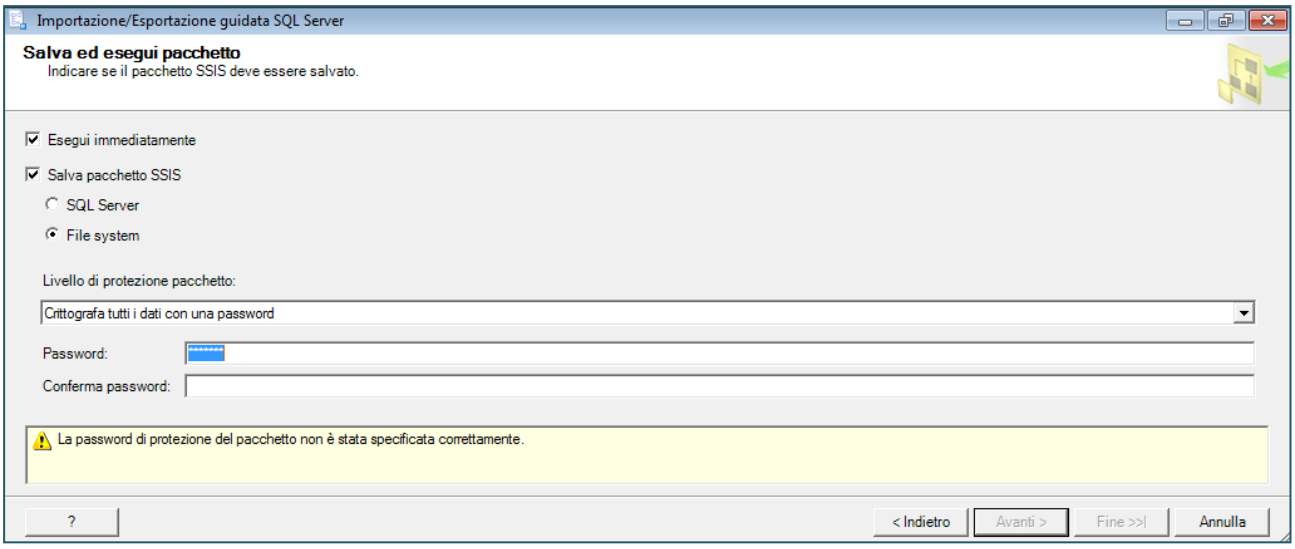
Se si desidera eseguire una query sull'origine dei dati e copiare i risultati, sarà possibile creare una query T-SQL. È possibile immettere la query T-SQL manualmente o utilizzare una query salvata in un file. Per individuare il file è possibile utilizzare la caratteristica di esplorazione disponibile nella procedura guidata, la quale apre automaticamente il file e ne incolla il contenuto nella pagina da cui è stato selezionato il file.



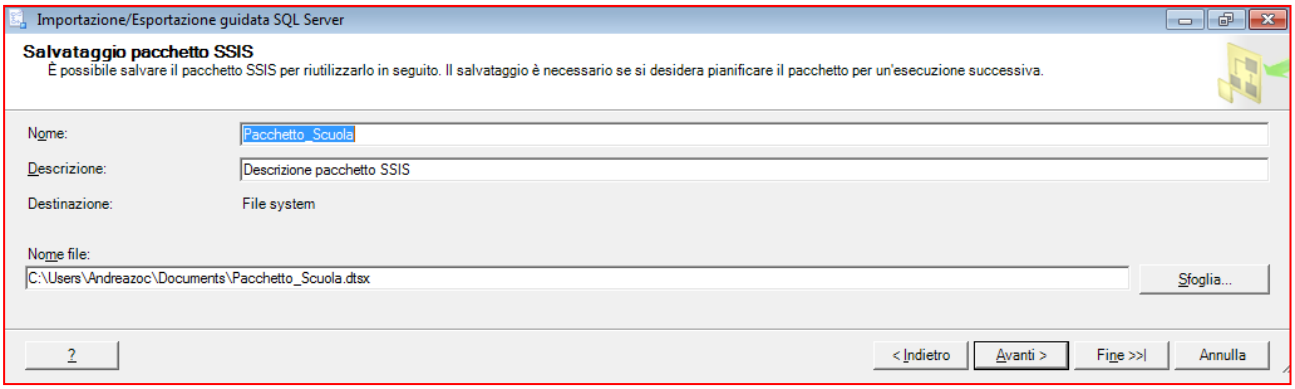

L'utente può scegliere le tabelle e le viste con un clic del mouse.

5. **Salvare ed eseguire un pacchetto.**

   - Se la procedura guidata viene avviata da SSMS o dal prompt dei comandi, il pacchetto potrà essere eseguito immediatamente. Se lo si desidera, è possibile salvare il pacchetto nel file system o nel database **msdb** di SQL Server. Se si salva il pacchetto, è possibile impostarne il livello di protezione e, se per quest'ultimo si intende usare una password, si deve digitarla due volte nelle apposite caselle di testo.




   - La procedura di salvataggio chiede un nome per il pacchetto:



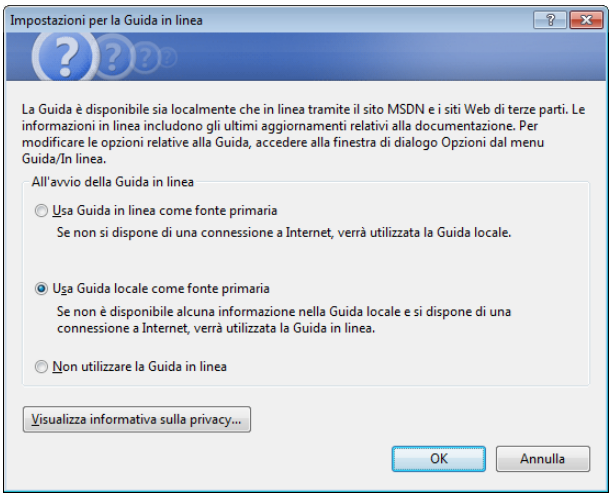

Se la procedura guidata è avviata da un progetto di Integration Services in SQL Server Data Tools (SSDT), non sarà possibile eseguire il pacchetto dalla procedura guidata. Il pacchetto sarà invece aggiunto al progetto di Integration Services da cui è stata avviata la procedura guidata. Potrà essere eseguito in seguito con SQL Server Data Tools (SSDT).

**LA GUIDA PER PRINCIPIANTI TERMINA QUI. DI SEGUITO SONO RIPORTATE LE APPENDICI PER IL PROGRAMMATORE.**

# APPENDICE A. – GUIDA DI SISTEMA (F1)

Dopo essersi autenticati sul (connessi al) server è possibile invocare la guida sia su particolari schermate che su parole chiave.

Con F1 (da tastiera) è possibile evocare la guida ipertestuale di SSMS.



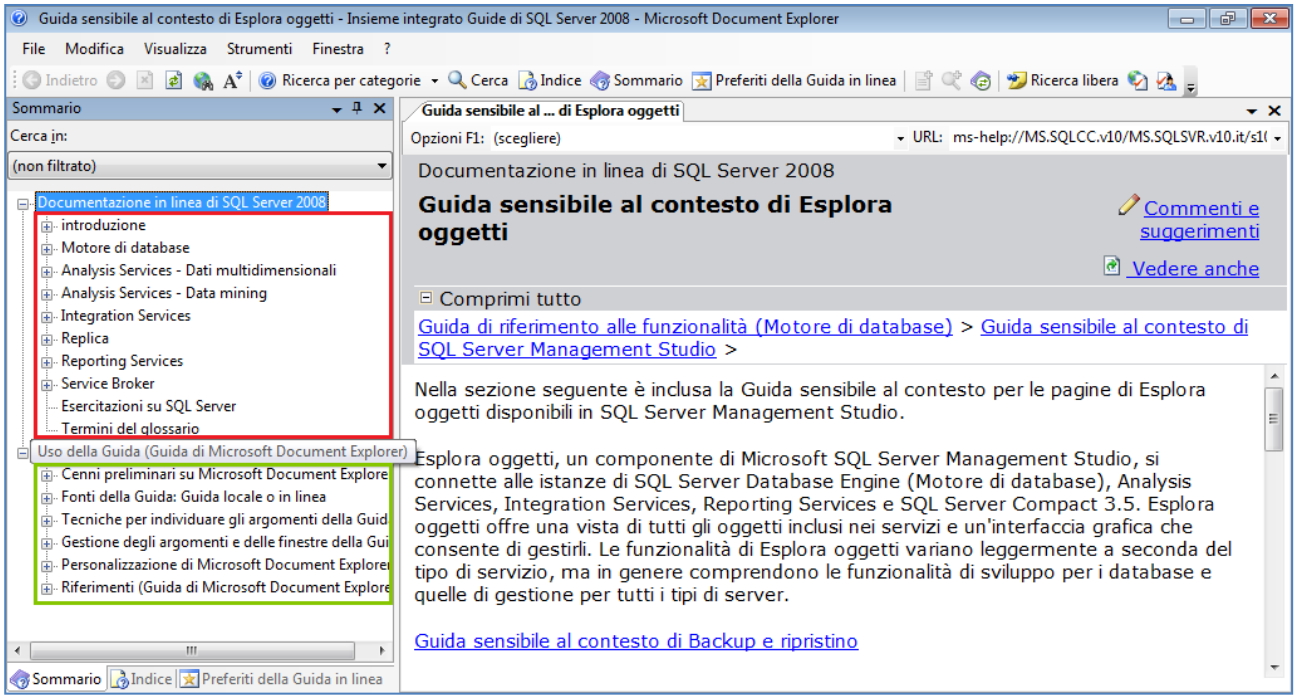

Al primo utilizzo il sistema chiede se si vuole usare la guida on Line (internet) come fonte primaria, oppure la guida locale (su HD della macchina del server) oppure se si vuole escludere la guida in linea (internet)

La prima scelta è consigliabile se si dispone di un server permanentemente connesso a internet.

La seconda scelta permette di alleggerire il carico di informazioni.

La terza scelta è utile solo per server off line, quasi mai connessi in internet, utilizzati come server di prova e di testing oppure come massima sicurezza.

Compare, quindi, la finestra illustrata qui sotto:




Il riquadro a sinistra propone un albero di scelte, in cui selezionare l'argomento desiderato.

Il riquadro a destra illustra l'argomento corrente della guida, con collegamenti ipertestuali per muoversi nel grafo delle informazioni di Management Studio.

# APPENDICE B. – TIPI DI DATO

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

SQL Server mette a disposizione un insieme di tipi di dato di sistema che possono essere usati direttamente in ogni porzione di codice che lo richieda. Inoltre, rispettando lo standard ANSI-92, mette a disposizione del programmatore la possibilità di definire nuovi tipi di dato personalizzati utilizzando dichiarazioni del linguaggio T-SQL e persino MS .NET Framework.

I tipi di dati alias sono nomi corrispondenti a tipi di dati di sistema predefiniti.

Per creare nuovi tipi di dati consultare l'istruzione CREATE TYPE (T-SQL), corrispondente all'istruzione CREATE DOMAIN dello standard ANSI-92.

SQL Server e .NET Framework sono basati su sistemi di tipi diversi.

Ad esempio il tipo **Decimal** .NET Framework dispone ad esempio di una scala massima di 28, mentre i tipi di dati decimali e numerici di SQL Server dispongono di una scala massima di 38.

Esiste un supporto che garantisce l'integrità dei dati in caso di lettura e scrittura dei dati tra applicazioni .NET e strutture SQL server. In questa guida non sarà analizzato nel dettaglio questo aspetto relativo ai tipi di dato.

Di seguito sono elencati i tipi predefiniti in SQL server, organizzati per gruppi affini.

## 43 Numerici esatti

### 43-1 Tipi di dati numerici esatti che utilizzano dati interi

| Tipo | Descrizione | Archiviazione |
|---|---|---|
| **bigint** | da -2^63 (-9,223,372,036,854,775,808) a 2^63-1 (9,223,372,036,854,775,807) | 8 byte |
| **int** | da -2^31 (-2.147.483.648) a 2^31-1 (2.147.483.647) | 4 byte |
| **smallint** | da -2^15 (-32,768) a 2^15-1 (32,767) | 2 byte |
| **tinyint** | da 0 a 255 | 1 byte |

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

### 43-2 Tipo di dato bit

| Tipo | Descrizione | Archiviazione |
|---|---|---|
| **Bit** | Tipo di dati integer che può accettare un valore di 1, 0 o NULL. | Vedi sotto. |

In Motore di database di SQL Server l'archiviazione utilizzata per le colonne bit viene ottimizzata. Se una tabella contiene 8 colonne di tipo bit o un numero inferiore, 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. Nelle conversioni nel tipo di dati bit i valori diversi da zero vengono promossi a 1.

### 43-3 Tipi di dati numerici con precisione e scala fisse.

| Tipo | Descrizione | Archiviazione |
|---|---|---|
| **decimal (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. I sinonimi ISO per il tipo di dati decimal sono dec e dec(p, s). Dal punto di vista funzionale numeric è equivalente a decimal.<br>p (precisione)<br>Numero massimo totale di cifre decimali che è possibile archiviare, sia a | Byte per archiviazione<br>1 - 9 : 5<br>10-19 : 9 |

| | | |
|---|---|---|
| | destra che 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.<br>s (scala)<br>Numero massimo di cifre decimali che è possibile archiviare a destra del separatore decimale. La scala deve essere un valore compreso tra 0 e p. È possibile specificare la scala 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. | 20-28: 13<br>29-38: 17 |
| **numeric (p,s)** | Identico a decimal (p,s) | |

### 43-4 Tipi valuta money e smallmoney

| Tipo | Descrizione | Archiviazione |
|---|---|---|
| **money** | Da -922.337.203.685.477,5808 a 922.337.203.685.477,5807 | 8 byte |
| **smallmoney** | Da -214.748,3648 a 214.748,3647 | 4 byte |

Tipi di dati che rappresentano valori monetari o valutari.

I tipi di dati money e smallmoney sono caratterizzati da una precisione pari a dieci millesimi 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. Esistono decine di valute predefinite in SS.

**Fonte:** http://msdn.microsoft.com/it-it/library/ms179882.aspx

# 44 Numerici approssimati

### 44-1 Tipi float e real, i numeri con la virgola

| Tipo | Descrizione | Archiviazione |
|---|---|---|
| **float (n)** | Da - 1,79E+308 a -2,23E-308, 0 e da 2,23E-308 a 1,79E+308<br>Dove n è il numero di bit utilizzato per archiviare la mantissa del numero float nella notazione scientifica e che pertanto determina la precisione e le dimensioni di archiviazione. Se si specifica n, il valore deve essere compreso nell'intervallo da 1 a 53. Il valore predefinito di n è 53.<br>N: 1-24 — Precisione: 7 cifre — Dimensioni dello spazio di archiviazione: 4 byte<br>N: 25-53 — Precisione: 15 cifre — Dimensioni dello spazio di archiviazione: 8 byte | Dipende da N |
| **real** | Da - 3,40E + 38 a -1,18E - 38, 0 e da 1,18E - 38 a 3,40E + 38 | 4 byte |

Il tipo di dati SQL Server **float(n)** è conforme allo standard ISO per tutti i valori di n da 1 a 53.

Il sinonimo di ISO per **real** è float(24). Il sinonimo per **double precision** è float(53).

**Fonte:** http://msdn.microsoft.com/it-it/library/ms173773.aspx

# 45 Date e tempo

### 45-1 I tipi Data

| Tipo | Descrizione | Archiviazione |
|---|---|---|
| **date** | Definisce una data. 10 caratteri, archiviato come numero precisione (10,0).<br>Struttura di archiviazione: Valore integer di 1, 3 byte archivia la data.<br>Accuratezza: Un giorno<br>Valore predefinito: 1900-01-01 | 3 byte, fissa |
| **datetime2** | Definisce una data costituita dalla combinazione con un'ora del giorno espressa nel formato 24 ore. datetime2 può essere considerato un'estensione del tipo | Da 6 a 8 byte |

| | | |
|---|---|---|
| | datetime esistente con un più ampio intervallo di date, una maggiore precisione frazionaria predefinita e una precisione specificata dall'utente facoltativa.<br>Struttura di archiviazione: 6 byte, o 7 byte o 8 byte.<br>Accuratezza: 100 nanosecondi<br>Valore predefinito: 1900-01-01 00:00:00 | |
| **datetime** | Tipo superato. Utilizzare i tipi di dati time, date, datetime2 e datetimeoffset per un nuovo lavoro. Definisce una data costituita dalla combinazione di un'ora del giorno e di secondi frazionari ed espressa nel formato 24 ore.<br>Accuratezza: Arrotondato a incrementi di 0,000, 0,003 o 0,007 secondi<br>Valore predefinito: 1900-01-01 00:00:00 | 8 byte fisso |
| **datetimeoffset** | Definisce una data in combinazione con un'ora del giorno con considerazione del fuso orario ed espressa nel formato 24 ore.<br>Accuratezza: 100 nanosecondi<br>Valore predefinito: 1900-01-01 00:00:00 00:00 | 10 byte, fissa |
| **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.<br>Accuratezza: Un minuto<br>Valore predefinito: 1900-01-01 00:00 | 4 byte, fissa |
| **time** | Definisce l'ora di un giorno. Il fuso orario non viene preso in considerazione e il formato è basato sulle 24 ore.<br>Accuratezza: 100 nanosecondi<br>Valore predefinito: 00:00:00 | 5 byte, fissa |

**Fonte:** http://msdn.microsoft.com/it-it/library/ms187752.aspx

# 46 Stringhe di caratteri

## 46-1 I tipi stringa non UNICODE

| Tipo | Descrizione | Archiviazione |
|---|---|---|
| **char (n)** | Dati di tipo string a lunghezza fissa non Unicode. Il parametro n definisce la lunghezza della stringa ed àè compreso tra 1 e 8.000.<br>Il sinonimo di ISO per char è **character**. | n byte |
| **text** | **Da evitare.** Dati non Unicode a lunghezza variabile. | Fino a 2E9 byte |
| **varchar (n)** | Dati di tipo string a lunghezza variabile non Unicode. Il parametro n definisce la lunghezza della stringa ed è tra 1 e 8.000. | Dati effettivi + 2byte |

I tipi di dati ntext, text e image verranno rimossi a partire da SQL Server 2012. Evitare di utilizzare questi tipi di dati.

**Fonte:** http://msdn.microsoft.com/it-it/library/ms176089.aspx

## 46-2 I tipi stringa UNICODE

| Tipo | Descrizione | Archiviazione |
|---|---|---|
| **nchar (n)** | Dati di tipo string a lunghezza fissa Unicode. Il parametro n (tra 1 e 4000) indica la lunghezza della stringa.<br>I suoi sinonimi ISO sono **national char** e **national character**. | Doppio di n |
| **ntext** | **Da evitare.** Dati Unicode a lunghezza variabile entro $2^{30}$ - 1 (1.073.741.823) byte. | Doppio della stringa immessa. |
| **nvarchar (n)** | Dati di tipo string a lunghezza variabile Unicode. Il parametro n (tra 1 e 4000) indica la lunghezza della stringa.<br>I suoi sinonimi ISO sono **national char varying** e **national character varying**. | Doppio della stringa immessa +2byte |

I tipi di dati ntext, text e image verranno rimossi a partire da SQL Server 2012. Evitare di utilizzare questi tipi di dati.

**Fonte:** http://msdn.microsoft.com/it-it/library/ms186939.aspx

# 47 Stringhe binarie

## 47-1 I tipi stringa binarie

| Tipo | Descrizione | Archiviazione |
|---|---|---|
| **binary (n)** | Dati binari a lunghezza fissa con lunghezza di n byte, con n tra 1 e 8.000. Se n non è specificato tramite la funzione CAST, la lunghezza predefinita è 30. Utilizzare **binary** se le dimensioni dei dati delle colonne sono consistenti. | n byte |
| **varbinary (n)** | Dati binary a lunghezza variabile, con n tra 1 e 8.000. il sinonimo per varbinary è **binary varying**. | Dati immessi +2 byte |
| **image** | **Da evitare.** Dati binari a lunghezza variabile da 0 a 2^31-1 (2.147.483.647) byte. | |

**Fonte:** http://msdn.microsoft.com/it-it/library/ms188362.aspx

# 48 Altri tipi di dati

## 48-1 I tipi stringa UNICODE

| Tipo | Descrizione | Archiviazione |
|---|---|---|
| **cursor** | **Fonte:** http://msdn.microsoft.com/it-it/library/bb677290.aspx<br>Serve per esplorare tabelle in modo procedurale. Le variabili di tipo cursor ammettono valori Null. | |
| **hierarchyid** | **Fonte:** http://msdn.microsoft.com/it-it/library/bb677290.aspx<br>Serve per rappresentare una posizione in un albero gerarchico. | |
| **sql_variant** | **Fonte:** http://msdn.microsoft.com/it-it/library/ms173829.aspx<br>Serve per l'archiviazione di valori di vari tipi supportati da SQL Server. | |
| **table** | **Fonte:** http://msdn.microsoft.com/it-it/library/ms175010.aspx<br>Tipo di dati speciale che può essere utilizzato per archiviare un set di risultati per l'elaborazione in un secondo momento. TABLE è principalmente usato per funzioni che rendono tabelle e per l'archiviazione temporanea di un set di righe rese da una funzione. È possibile dichiarare funzioni e variabili di tipo TABLE. | |
| **timestamp**<br>**rowversion** | **Fonte:** http://msdn.microsoft.com/it-it/library/ms182776.aspx<br>timestamp è sinonimo del tipo di dati **rowversion**: utilizzare rowversion anziché timestamp, se possibile. La sintassi timestamp è deprecata. La keyword sarà rimossa da MS SQL Server 2012.<br>Il tipo di dati T-SQL timestamp è diverso dal tipo di dati timestamp definito nello standard ISO, e non è un orario preciso.<br>È usato per numeri binari univoci generati automaticamente all'interno di un database. È spesso usato come meccanismo d'individuazione delle tuple con il numero di versione. | 8 byte |
| **xml** | **Fonte:** http://msdn.microsoft.com/it-it/library/ms187339.aspx<br>Tipo di dati in cui sono archiviati dati XML. | |
| **uniqueidentifier** | **Fonte:** http://msdn.microsoft.com/it-it/library/ms187942.aspx<br>GUID a 16 byte. | 16 byte |

# APPENDICE C. – FUNZIONI SCALARI

**Fonte:** http://msdn.microsoft.com/it-it/library/ms174318(v=sql.100).aspx

Le principali funzioni scalari messe a disposizione da SQL Server 2008 sono le seguenti:

## 49 Funzioni di data e ora

**Fonte:** http://msdn.microsoft.com/it-it/library/ms186724(v=sql.100).aspx

| 49-1 Elenco funzioni | Descrizione |
|---|---|
| SYSDATETIME () | Rende la data e l'ora del computer in cui è attivo SQL Server. **datetime2** |
| CURRENT_TIMESTAMP () | Rende la data e l'ora del computer in cui è attivo SQL Server. **datetime** |
| GETDATE ( ) | Rende la data e l'ora del computer in cui è attivo SQL Server. **Datetime** |
| DATENAME ( datepart , date ) | Rende una stringa di caratteri che rappresenta la datepart della data specificata. |
| DATENAME (datepart, date) | Rende un integer che rappresenta il datepart dell'argomento date specificato. |
| DAY ( date ) | Rende un integer che rappresenta la parte del giorno della date specificata. |
| MONTH ( date ) | Rende un integer che rappresenta la parte mese di un date specificato. |
| YEAR ( date ) | Rende un integer che rappresenta la parte dell'anno di date specificata. |
| DATEDIFF (dp, date1, date2) | Rende la differenza tra le due date date1 e date2; dp è la datepart. |
| DATEADD (dp, number, date) | Rende una data ottenuta sommando un intervallo alla data di partenza. |
| ISDATE ( expression ) | Determina se l'espressione è un tipo di data o di ora valido. |

## 50 Funzioni matematiche

**Fonte:** http://msdn.microsoft.com/it-it/library/ms177516(v=sql.100).aspx

| 50-1 Elenco funzioni | Descrizione |
|---|---|
| ABS (num_expr) | Valore assoluto di num_expr |
| ACOS ( num_expr) | Arcoseno di num_expr |
| ASIN ( num_expr) | Arcoseno di num_expr |
| ATAN ( num_expr) | Arcotangente di num_expr |
| COS (num_expr) | Coseno di num_expr |
| COT (num_expr) | Cotangente di num_expr |
| EXP (num_expr) | Restituisce il valore esponenziale dell'espressione float specificata |
| ISNUMERIC (espressione) | Valuta se l'espressione è di tipo NUMERIC |
| LOG (num_expr) | Logaritmo naturale di num_expr |
| LOG10 (num_expr) | Logaritmo in base 10 di num_expr |
| SQRT ( float_expression ) | Restituisce la radice quadrata del valore float specificato |
| SQUARE ( float_expression ) | Restituisce il quadrato del valore float specificato |
| CEILING (num_expr) | Rende il più piccolo valore integer maggiore o uguale a num_expr |
| FLOOR (num_expr) | Rende il valore integer maggiore che risulta minore o uguale a num_expr |

## 51 Funzioni varie

| 51-1 Elenco funzioni | Descrizione |
|---|---|
| ASCII (carattere) | Converte un carattere nel suo codice ASCII |
| CAST (expr AS Tipo) | Converte una expr SQL in un determinato Tipo di dato |
| CHAR (intero) | Converte in ASCII il carattere corrispondente all'intero |
| CONVERT (tipodata (lung), exp) | Converte dati da un tipo ad un altro |
| ISNULL (exp, alternativa) | Se exp è un NULL la sostituisce con il secondo valore |
| LEFT (exp, numero) | La parte di stringa a partire dal numero |
| LEN (exp_string) | La lunghezza della stringa |
| LOWER (exp_char) | Converte in minuscolo |
| LTRIM (exp_char) | Toglie gli spazi a sinistra |
| NULLIF (exp, exp) | Restituisce NULL se espressioni sono equivalenti |

# APPENDICE D. – BARRE DEGLI STRUMENTI

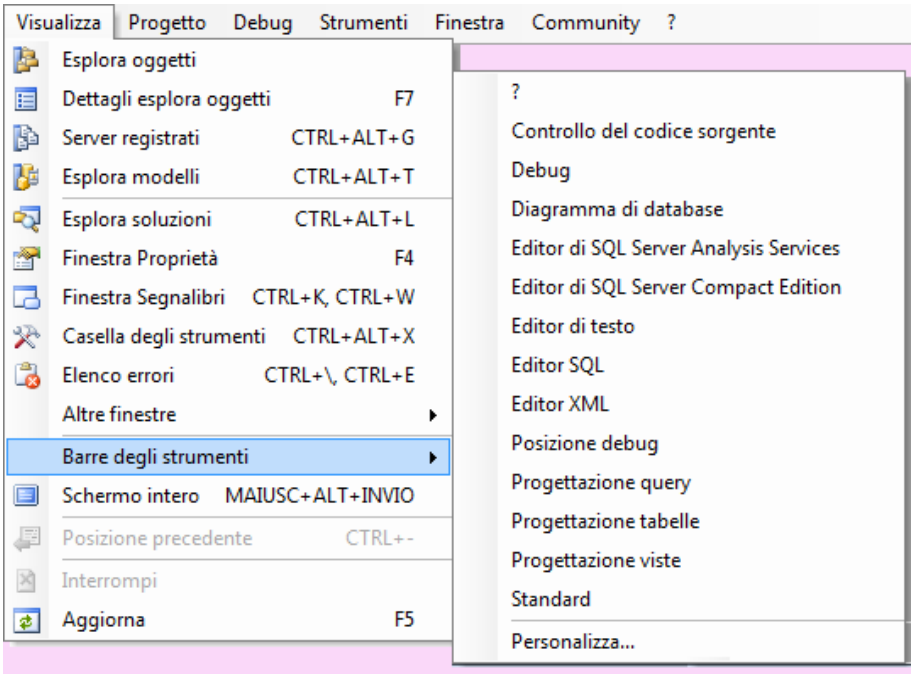
## 52 Visualizzare le barre

In questo documento è stata già introdotta la Visualizzazione dei vari elementi di SSMS.

In questa sezione saranno analizzate le diverse barre degli strumenti e i menu disponibili nell'interfaccia visuale.

Oltre a proporre numerose barre predefinite, occorre osservare che SSMS consente di creare menu e barre personalizzate, come già accadeva in alcuni prodotti Microsoft precedenti.

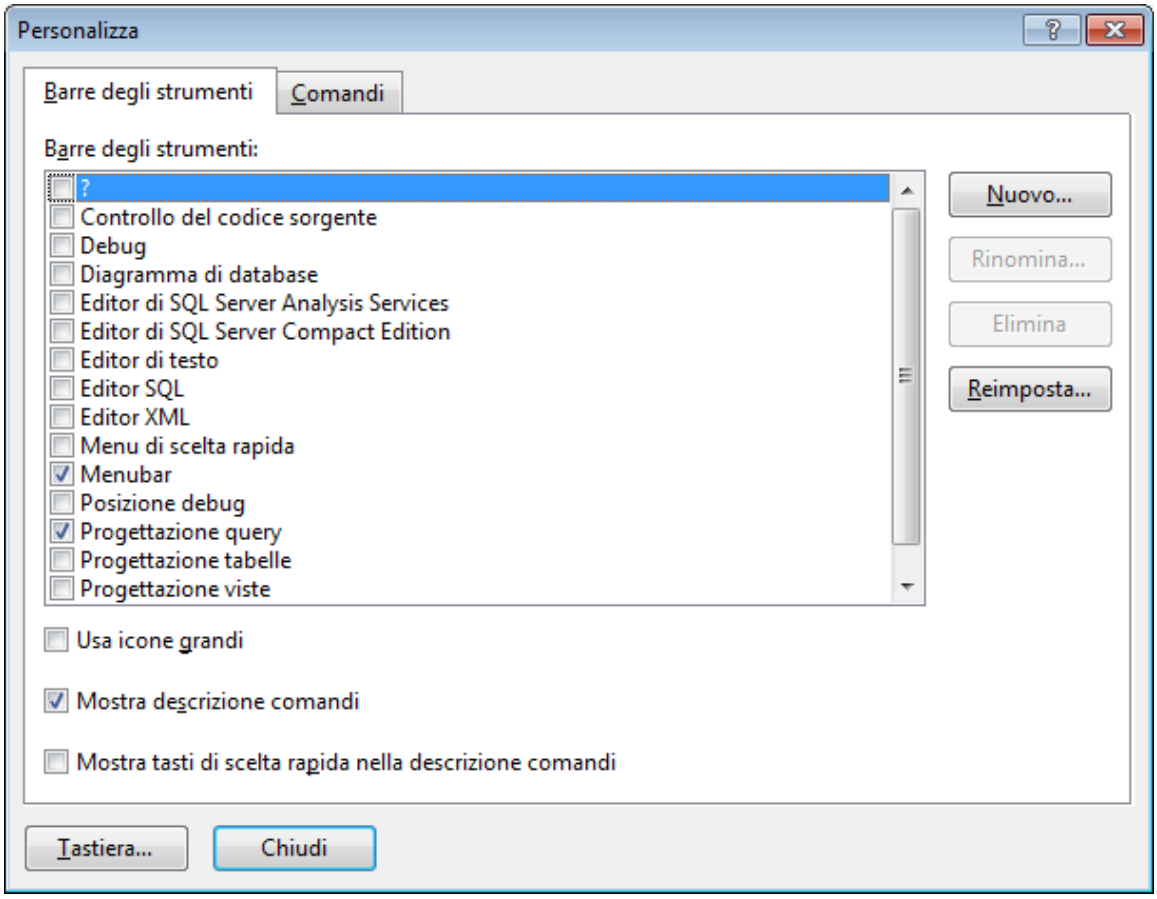
Dal menu Visualizza è possibile elencare gli elementi da mostrare e in particolare le barre elencate dalla voce **Barre degli strumenti**.




## 53 Personalizzare le barre

**Fonte:** http://technet.microsoft.com/it-it/library/ms183484.aspx

Un altro modo per visualizzare le barre consiste nel scegliere dal menu la voce **Personalizza** (l'ultima del menu).

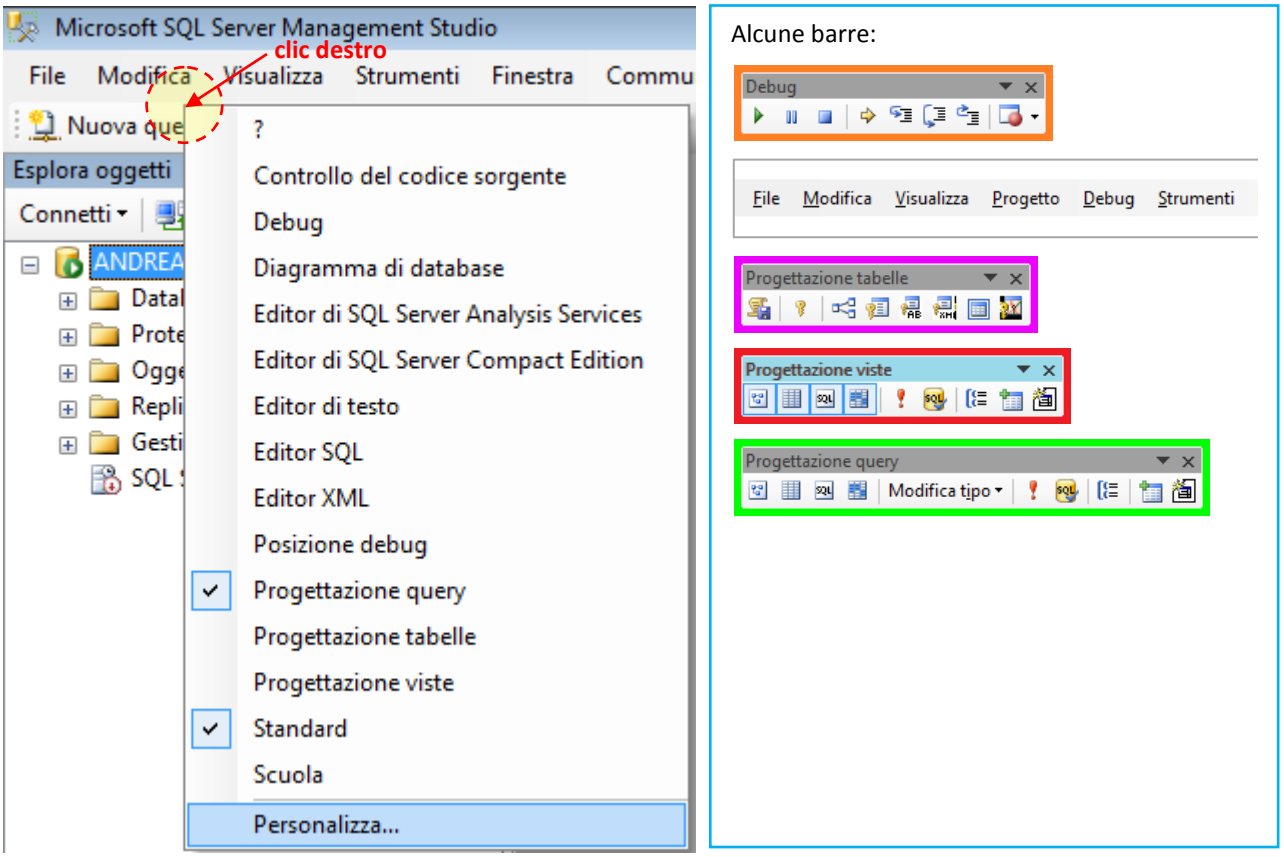



Compare la finestra qui riprodotta che consente di visualizzare menu e barre semplicemente spuntando le caselle.

Se si spunta una casella appare subito la barra, se si de-spunta la si nasconde.

Nei seguenti paragrafi si analizzeranno le diverse barre tra cui quelle dei menu. È importante anche osservare che è possibile spostare le barre e ancorarle alle barre dei menu in alto, oppure renderle mobili semplicemente agganciandole e trascinandole.

Un modo rapido per far apparire e nascondere le barre degli strumenti è un clic destro su una parte vuota dello spazio riservato ai menu e alle barre, come nella seguente figura:



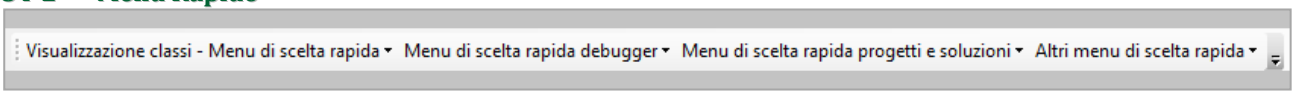

# 54 Menu

## 54-1 Menu Bar (principale)


La barra dei Menu si adatta automaticamente alle esigenze di lavoro. Per esempio i menu Progetto e Debug potrebbero non essere visibili inizialmente, ma è sufficiente aprire una tabella in modalità progettazione per visualizzarli.

La barra dei Menu contiene le voci principali di attività. Le voci File, Modifica e Visualizza sono auto esplicative. La voce Progetto riguarda costruzione di pacchetti applicativi, migrazioni, cartelle, nuovi oggetti (tabelle, query, viste per esempio) e altri elementi analoghi. La voce Debug serve per eseguire un codice, anche in modalità controllata. Il menu Strumenti è riservata alla gestione del motore del database e all'uso di strumenti esterni. Il menu Finestra serve per disporre e nascondere le finestre dell'interfaccia, mentre il menu Community per documentarsi in collaborazione. Infine il punto di domanda rappresenta l'Aiuto del sistema.

## 54-2 Menu Rapido

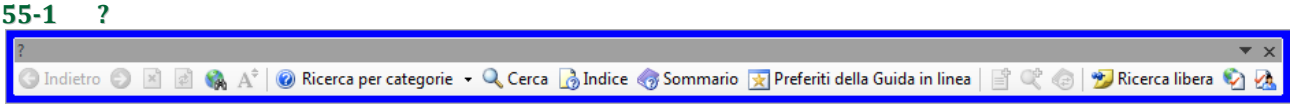



È un aiuto per gli amministratori del database e gli sviluppatori di applicazioni. Consente di accedere a voci usate frequentemente con rapidità. In realtà coloro che gestiscono o sviluppano preferiranno costruirsi i propri menu e le proprie barre personali che consentono una ancora migliore celerità di accesso alle opzioni più frequenti.

# 55 Barre

Le barre dell'interfaccia appaiono e scompaiono in funzione del lavoro in corso. Per esempio se si progetta una vista appare la Barra rispettiva, che mostra le opzioni più utili.

## 55-1 ?

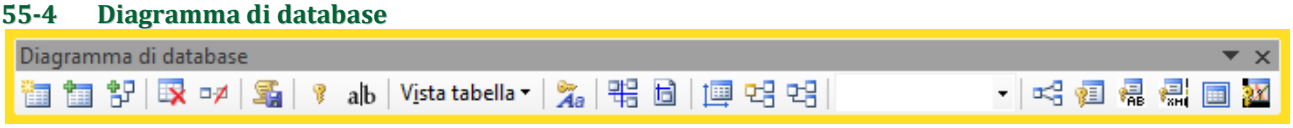



La barra di aiuto consente ricerche di documentazione nei materiali on-line e off-line. In questa sede la tralasciamo.

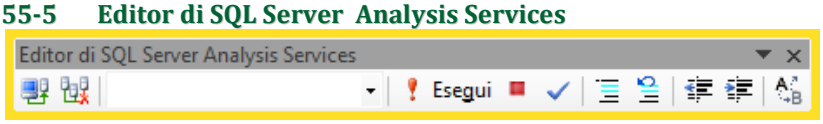
## 55-2 Controllo del codice sorgente


## 55-3 Debug

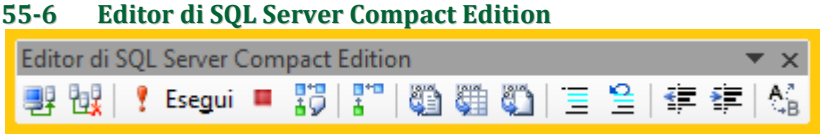

## 55-4 Diagramma di database




## 55-5 Editor di SQL Server Analysis Services




## 55-6 Editor di SQL Server Compact Edition

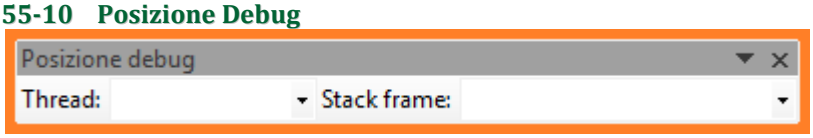



## 55-7 Editor di testo

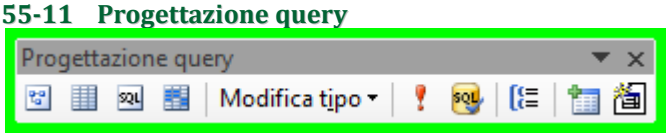

## 55-8 Editor SQL


## 55-9 Editor XML

## 55-10 Posizione Debug




## 55-11 Progettazione query




## 55-12 Progettazione tabelle

La barra Progettazione Tabelle, serve per attivare le principali funzionalità relative alle tabelle.


Genera script delle modifiche,

Imposta chiave primaria,

Relazioni,

Gestione Indici/Chiavi

Gestione Indice Full-Text

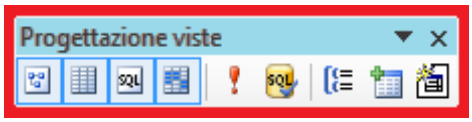
Gestione Indici XML,

Gestione Vincoli CHECK

Gestione Indici spaziali

## 55-13 Progettazione viste

La barra Progettazione Viste, serve per attivare le principali funzionalità relative alle viste.




Mostra Riquadro Diagramma,

Mostra Riquadro criteri,

Mostra Riquadro SQL,

Mostra Riquadro risultati,

Esegui SQL,

Verifica Sintassi SQL,

Aggiungi Raggruppamento,

Aggiungi Tabella,

Aggiungi Nuova Tabella derivata,

## 55-14 Standard

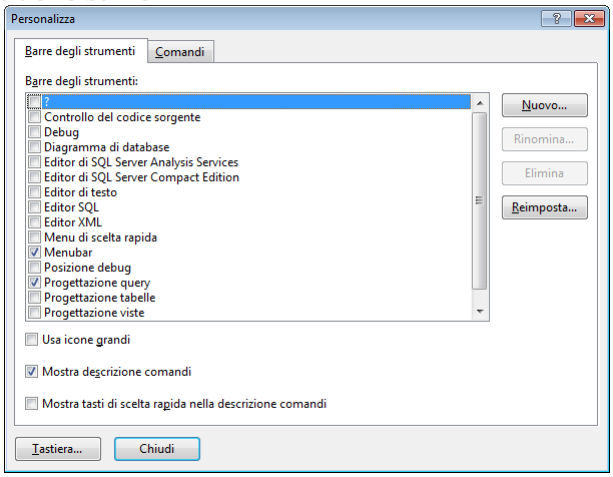

Query MDX di Analysis Services

Nuova Query

Query del motore di database

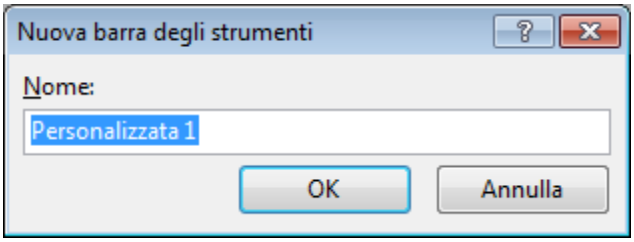
## 55-15 PARAGRAFO DA MODIFICARE personalizzazione delle barre

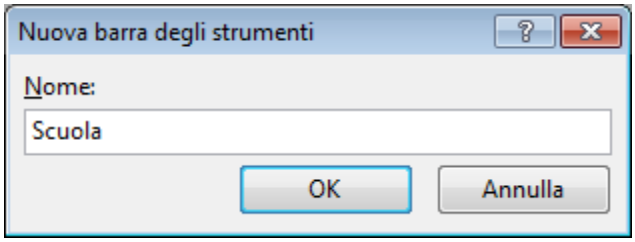
SQL SSMS 2008 offre la possibilità di personalizzare le barre esistenti e persino creare delle barre personalizzate dall'utente. Dal menu Visualizza o dal menu contestuale è possibile scegliere la voce Personalizza che presenta una finestra simile alla figura qui di lato.




Il pulsante **Nuovo ...** consente di creare una nuova barra, mentre visualizzando la paletta **Comandi** è possibile trascinare (drag & drop) i comandi (pulsanti) nelle barre desiderate; è anche possibile eliminare pulsanti dalle barre, e modificarne alcune caratteristiche.

Se si sceglie **Nuovo** compare la finestra qui sotto, in cui è possibile assegnare un nome alla barra (nel nostro esempio la barra Scuola).

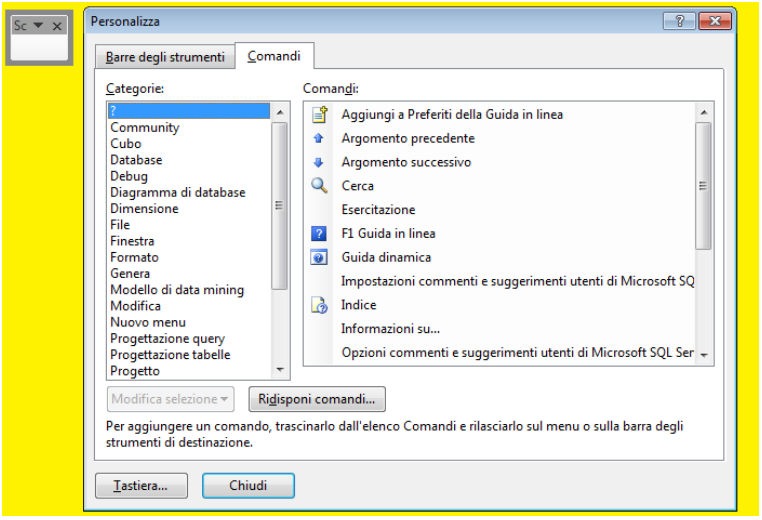



La paletta Comandi elenca i comandi disponibili per il SSMS, raggruppati per categoria.

Se è stata richiesta una nuova barra, essa compare, vuota, vicino alla finestra Personalizza.

Qui a lato è mostrato una raffigurazione della finestra e della barra nuova Scuola.




Qui sotto la barra di esempio completata:

# APPENDICE E. – SINTASSI SEMPLIFICATA

## 56 Database

### 56-1 DDL: CREATE DATABASE

Transact-SQL

```
CREATE DATABASE nomeDB ON  PRIMARY
 (NAME = N'nomeDB',
  FILENAME = N'C:\Cartella\nomeFile.mdf' ,
  SIZE = 3072KB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB
 )
 LOG ON
 (NAME = N' nomeDB_log',
  FILENAME = N'C:\Cartella\ nomeFile_log.ldf' ,
  SIZE = 1024KB , MAXSIZE = 2048GB ,  FILEGROWTH = 10%
 )
```

### 56-2 Esempio:

Transact-SQL

```
CREATE DATABASE [Biblioteca] ON  PRIMARY
 (NAME = N'Biblioteca',
  FILENAME = N'C:\Cartella\Biblioteca.mdf' ,
  SIZE = 3072KB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB
 )
 LOG ON
 (NAME = N'Biblioteca_log',
  FILENAME = N'C:\Cartella\Biblioteca_log.ldf' ,
  SIZE = 1024KB , MAXSIZE = 2048GB ,  FILEGROWTH = 10%
 )
```

## 57 Tabelle

### 57-1 DDL: CREATE TABLE

Transact-SQL

```
CREATE TABLE nomeTabella
    (
  nomeCampo Tipo [ NULL | NOT NULL ] ::vincolo ,
  nomeCampo Tipo [ NULL | NOT NULL ] ::vincolo ,
   . . .
  nomeCampo Tipo [ NULL | NOT NULL ] ::vincolo = 0 ,
 CONSTRAINT nomeVincolo
    | PRIMARY KEY | UNIQUE | CLUSTERED | NONCLUSTERED (campo [ ASC | DESC ] )
          ON (partition_column_name)
    | FOREIGN KEY ( campo [ ,...n ] )
          REFERENCES tabellaCollegata [ (campiEsterni ) ]
             [ ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
             [ ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
    | CHECK ( espressioneBooleana )
   )
//--- --- --- --- --- --- --- --- --- --- --- ---
::vincolo -->
 - [ DEFAULT valoreIniziale ]
 - [ CONSTRAINT nomeVincolo ]
 - [ IDENTITY ( base, incremento ) ]
 - [ PRIMARY KEY | UNIQUE ]
 - [ CLUSTERED | NONCLUSTERED ]
 - [ FOREIGN KEY ]
       REFERENCES tabellaCollegata ( colonnaCollegata )
       [ ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT ]
       [ ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT ]
 - CHECK ( espressioneBooleana )
```

### 57-2 Esempio:

Transact-SQL

```
CREATE TABLE Prestiti (
   IDPrestito int IDENTITY (1,1) NOT NULL PRIMARY KEY,
   Data date NOT NULL ,
   DataScadenza date NOT NULL ,
   DataRestituzione date NULL ,
   IDRichiedente int NOT NULL FOREIGN KEY REFERENCES Persone (IDPersona)
        ON DELETE NO ACTION
        ON UPDATE CASCADE ,
   IDCopiaLibro int NOT NULL FOREIGN KEY REFERENCES CopieLibri (IDCopiaLibro)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION ,
  CONSTRAINT ControlloDate
        CHECK ( ( Data < DataScadenza ) AND ( Data < DataRestituzione ) )
)
```

# 58 Indici

### 58-1 DDL: CREATE INDEX

Transact-SQL

```
CREATE [UNIQUE] [CLUSTERED | NONCLUSTERED]
INDEX nomeNDX
ON nomeTabella
(
   Colonna [ASC | DESC ],
   Colonna [ASC | DESC ],
   . . .
   Colonna [ASC | DESC ],
)
```

### 58-2 Esempio

Transact-SQL

```
CREATE UNIQUE NONCLUSTERED
INDEX EvitaDuplicati
ON Persone
(
   Cognome ASC,
   Nome ASC,
   Sesso ASC,
   DataNascita ASC
)
```

È ammesso un solo indice CLUSTERED per una tabella; solitamente si ha un indice CLUSTERED sulla PK, tutti gli altri sono UNCLUSTERED

L'indice ha un nome che lo identifica e si riferisce a una (sola) tabella; tra le parentesi si elencano le colonne e (opzionalmente) se vige ordinamento ascendente o discendente.

# 59 Viste

### 59-1 DDL: CREATE VIEW

Transact-SQL

```
CREATE VIEW nomeVista
AS
Comando:SELECT_TSQL
```

### 59-2 Esempio

Transact-SQL

```
CREATE VIEW Vista_Completa
AS
SELECT    C.IDClasse AS Chiave, C.Anno, C.Sezione, C.Indirizzo, S.IDStudente,
          S.Cognome, S.Nome, S.Sesso, S.DataNascita, S.Tasse, S.Classe
FROM      Classi AS C INNER JOIN
          dbo.Studenti AS S ON C.IDClasse = S.Classe
ORDER BY  C.Anno, C.Sezione, C.Indirizzo
```

# 60 Stored Procedure

## 60-1 DDL: CREATE PROCEDURE

Transact-SQL

```
CREATE PROCEDURE nomeProcedura
   -- ELENCO PARAMETRI OPZIONALE
   @nomeParametro tipo vincolo = valore_default,
   @nomeParametro tipo vincolo = valore_default,
   . . .
   @nomeParametro tipo vincolo = valore_default,
AS
BEGIN
   Comandi T-SQL
   -- l'uso di RETURN forza l'uscita
END
```

## 60-2 Esempio

Transact-SQL

```
CREATE PROCEDURE Inserimento_Studenti
   -- ELENCO PARAMETRI
   @cognome varchar(50) = NULL,
   @nome varchar(50) = NULL
AS
BEGIN
   SET NOCOUNT ON;
     -- comandi della procedura
   IF (@cognome = NULL) OR (@nome = NULL)
   ROLLBACK
   ELSE
   INSERT INTO Studenti (Cognome, Nome)
   VALUES (@cognome, @nome)
   COMMIT
END
```

# 61 Trigger

## 61-1 DML Trigger su una tabella o una vista

Transact-SQL

```
CREATE TRIGGER nomeTrigger
ON { nomeTabella | nomeVista } [ WITH opzioniDelTrigger ]
   { FOR | AFTER | INSTEAD OF }
   { [ INSERT ] [ , ]      [ UPDATE ] [ , ]      [ DELETE ] }
AS
   sequenzaComandiSQL [ ; ]
```

## 61-2 Esempio

Transact-SQL

```
CREATE TRIGGER Max30studenti
   ON  Studenti
   AFTER INSERT
AS
BEGIN
   SET NOCOUNT ON;
   DECLARE @quanti int = 0;
   SELECT @quanti = COUNT(*)
   FROM Studenti , Classi
   WHERE Studenti.Classe = Classi.IDClasse
   GROUP BY Classi.IDClasse
   HAVING COUNT(*) > 30;
   IF (@quanti > 30)       ROLLBACK;
   ELSE                COMMIT;
END
```

# APPENDICE F. – SQL SERVER SUL WEB

## 62 Documenti consultati

I documenti da cui sono stati tratti i principali spunti di lavoro per il presente documento sono i seguenti:

| | |
|---|---|
| **Guida di SQL Server** | Fonte: http://www.mrwebmaster.it/sql-server/guide/guida-sql-server-2008/ |
| **Requisiti hw e sw** | Fonte: http://msdn.microsoft.com/it-it/library/ms143506%28v=sql.100%29.aspx |
| **Edizioni del server** | Fonte: http://msdn.microsoft.com/it-it/library/ms143506%28v=sql.100%29.aspx |
| **Data Warehouse Scalability** | Fonte: http://technet.microsoft.com/library/cc278097%28SQL.100%29.aspx |
| **SQL Server 2008 vs 2005** | Fonte: http://www.databasejournal.com/features/mssql/article.php/3792476/Performance-Testing-150-SQL-Server-2008-versus-SQL-Server-2005.htm |
| **Sinonimi di SQL server** | Fonte: http://technet.microsoft.com/it-it/library/ms187552.aspx |
| **Caratteristiche del database** | Fonte: http://technet.microsoft.com/it-it/library/hh230827.aspx |
| **Introduzione a SQL server** | Fonte: http://msdn.microsoft.com/it-it/library/cc185016.aspx |

## 63 Service Pack

In questo articolo vengono elencati i problemi di SQL Server 2008 che sono stati corretti da Microsoft SQL Server 2008 Service Pack 3 (SP3): http://support.microsoft.com/kb/2546951/it

Microsoft distribuisce le correzioni di Microsoft SQL Server 2008 come un unico file scaricabile. Poiché le correzioni sono cumulative, ogni nuova versione contiene tutte le correzioni rapide e rilascio delle correzioni per tutte le correzioni fornite con il precedente 2008 di SQL Server. In genere, è necessario scaricare il service pack più recente.

### 63-1 SP3 per SQL server 2008

http://www.microsoft.com/en-us/download/details.aspx?id=27594

# INDICE