Introduzione a SQL Server

Architettura

- Sistema client-server
- Server: SERVEDB,
  - indirizzo interno (dal laboratorio) 192.168.0.252
  - Indirizzo esterno (da casa con OpenVPN) 10.17.2.91
- Server: c’e’ un server su ogni macchina del laboratorio
- Client:
  - Su tutte le macchine del laboratorio
  - Visual Studio 2010

Architettura

- Ogni server contiene una o più **istanze**:
  - Una istanza corrisponde ad un processo separato sulla macchina server
  - Una istanza rimane in ascolto per le richieste dei client su una porta scelta dall'amministratore
  - Esiste una istanza di default (senza nome)
  - Le altre hanno un nome
  - Per collegarsi ad una particolare istanza su un server i client devono specificare il nome o la porta

Architettura

- Ogni istanza contiene diversi **database**
  - 4 database di sistema: master, model, msdb e tempdb
  - 0 o più database utente
- Ogni database contiene diversi oggetti: tabelle, viste, stored procedures
- Gli oggetti sono divisi in **schemi**
  - gli schemi sono posseduti dagli utenti, non gli oggetti dirrettamente
  - Gli schemi rappresentano un namespace: ogni oggetto di uno schema deve avere un nome diverso

Architettura: esempio

- Su SERVEDB c'e' una sola istanza: default (senza nome)
- Nell'istanza su SERVEDB ci sono 2 database utente di esempio:
  - AdventureWorks
  - AdventureWorksDW
- AdventureWorks ha 18 schemi, tra cui alcuni sono:
  - Person
  - Production
  - Sales

Transact-SQL: convenzioni sintattiche

- MAIUSCOLO: parola chiave Transact-SQL
- *Corsivo*: parametri forniti dall'utente
- **Grassetto**: nomi di tabelle, colonne, indici, stored procedures, utilità, tipi di dato e testo che deve essere digitato esattamente come mostrato
- Sottolineato: indica il valore di default che si applica quando la clausola che contiene il valore sottolineato è omesso dal comando
- | (barra verticale) separa oggetti sintattici all'interno di parentesi quadre o graffe. Si può scegliere solo uno degli oggetti

Transact-SQL: convenzioni sintattiche

- [ ] : racchiudono un elemento sintattico opzionale
- { } : racchiudono un elemento sintattico richiesto
- [**,**...*n*] indica che l'elemento che lo precede può essere ripetuto *n* volte. Le occorrenze sono separate da virgole.
- [...*n*] indica che l'elemento che lo precede può essere ripetuto *n* volte. Le occorrenze sono separate da spazi.
- <label> : blocco di sintassi (simbolo non terminale della grammatica)
- <label> ::= : definizione di un blocco di sintassi

Transact-SQL

- Tutti i riferimenti in Transact-SQL ad un oggetto hanno 3 forme possibili:

  database_name.[*schema_name*].*object_name*

  | *schema_name*.*object_name*

  | *object_name*
- *database_name:* Nome del database nel quale l'oggetto risiede sull'istanza locale
- *schema_name:* Nome dello schema che contiene l'oggetto
- *object_name:* nome dell'oggetto

Transact-SQL

- Il nome del database e dello schema possono essere omessi perchè ogni utente ha un database e uno schema di default
- Sono necessari se l'oggetto si trova in un database e/o in uno schema diversi da quelli di default
- Si può mettere il nome del db e omettere quello dello schema così

  database_name..*object_name*

Connessione ad una istanza

- Collegarsi al pc con username Utente
- Lanciare Visual Studio 10
- Al lancio si clicca su Server Explorer-> Data connection-> Add connection e si scrive il nome del server
  - Ad esempio: 192.168.0.252
- Connect to a database: AdventureWorks
- Si lasciano gli altri parametri invariati
  - Authentication=Windows Authentication
  - Questo assume che siamo utenti registrati nel sistema oppure siamo loggati come amministratori della macchina locale (e ci stiamo colllegando a localhost)

Visual Studio 10

- Il server explorer mostra, in maniera gerarchica, tutti gli oggetti disponibili nell'istanza e nel database a cui ci siamo collegati
- Ci possiamo collegare a più di una istanza contemporaneamente
- Le azioni disponibili per ogni oggetto si ottengono con il tasto destro del mouse
  - Possiamo vedere il contenuto di una tabella (tutte le sue righe) scegliendo "Show Table Data"

Esecuzione di una query

- Scegliere Data ->Transact SQL Editor->New Query Connection
- Appare la schermata che chiede di connettersi
- Appare la finestra del Query Editor in cui possiamo scrivere in Transact-SQL la query, ad esempio

  USE AdventureWorks

  SELECT * FROM HumanResources.Department

  (ricordate: le parole chiave SQL non sono case sensitive)
- Premendo il bottone “Execute SQL” si esegue la query e viene mostrato il risultato

Identificatori

- I nomi degli oggetti di un database sono chiamati **identificatori**
- Si distinguono in **regolari** e **delimitati**
- Gli identificatori regolari devono seguire le seguenti regole:
- Il primo carattere deve essere uno dei seguenti
  - Una lettera come definito dallo Standard Unicode 3.2. La definizione Unicode di lettere include i caratteri Latini dalla a alla z, dalla A alla Z e anche lettere da altri linguaggi
  - L'underscore (_), simbolo "at" (@), o simbolo di numero (#). Alcuni simboli all'inizio di un identificatore hanno un significato speciale in SQL Server. Un identificatore che comincia con @ denota una variabile locale o un parametro di una stored procedure. Un identificatore che comincia con # indica una tabella o una procedura temporanea. Un identificatore che comincia con ## denota un oggetto temporaneo globale..
    Alcune funzioni Transact-SQL hanno nomi che cominciano con @@. Per evitare confusione con queste funzioni è meglio evitare nomi che cominciano con @@

Identificatori

- I caratteri seguenti possono includere I seguenti
  - Lettere come definite nello Standard Unicode 3.2
  - Numeri decimali da Basic Latin o altri script nazionali
  - at (@), dollaro ($), numero (#) o underscore (_)
  - L'identificatore non deve essere una parola riservata Transact-SQL. SQL Server riserva sia le versioni maiuscole che minuscole delle parole riservate.
  - Spazi o caratteri speciali non sono ammessi

Identificatori delimitati

- Identificatori che non seguono le regole degli identificatori regolari
- Sono delimitati da parentesi quadre ([]) o doppie virgolette (“)

```
SELECT *
FROM [My Table]  --Identifier contains a space and uses a
    -- reserved keyword.
WHERE [order] = 10   --Identifier is a reserved keyword.
```

- Gli identificatori regolari possono o meno essere delimitati
- Tutti gli identificatori devono contenere da 1 a 128 caratteri. Le tabelle temporanee locali hanno un massimo di 116 caratteri

Identificatori delimitati

- Se voglio usare un delimitatore in un identificatore devo farlo precedere da un altro delimitatore (sequenza escape)

```
CREATE TABLE [Employee]]]
(
EmployeeID int IDENTITY (1,1) NOT NULL,
FirstName varchar(30),
LastName varchar(30)
)
```

SQL da riga di comando: sqlcmd

- Aprire un prompt dei comandi
- Lanciare sqlcmd
- Sintassi

  sqlcmd –S *server_name* [*instance_name*]
- Esempio

  sqlcmd –S 192.168.0.252

  Appare il prompt di sqlcmd: 1>
- Si possono scrivere I comandi Transact-SQL
  - I comandi possono occcupare più di una riga
- Una volta scritti i comandi si scrive GO per eseguirli

sqlcmd

- Oltre ai comandi Transact-SQL, sono disponibili i seguenti comandi:
- **GO** [*count*]: i comandi inseriti vengono messi in una cache che rappresenta un batch. Quando si scrive GO il batch viene eseguito, con GO *count* viene eseguito *count* volte
- **:QUIT**: esci da sqlcmd
- **:RESET**: cancella la cache delle istruzioni
- **:List**: mostra le istruzioni nella cache
- **:Out <*filename*>| STDERR| STDOUT**: manda l'output del batch al file <filename> o allo standard error o allo standard output
- **:Error <*filename*>| STDERR| STDOUT**: manda i messaggi di errore del batch al file <filename> o allo standard error o allo standard output
- **:!!** <*command*> : esegue il comando di sistema operativo <*command*> sulla macchina sulla quale sqlcmd sta eseguendo

sqlcmd

- **:r <*filename*>** : aggiunge i comandi SQL presenti in <*filename*> alla cache corrente, la directory corrente è quella da cui è stato lanciato sqlcmd
- **:Perftrace <*filename*>| STDERR| STDOUT** : ridirige i messaggi di controllo delle prestazioni
- **:Help** : mostra i comandi disponibili in sqlcmd
- **:Connect** *server_name*[*****instance_name*] : chiude la connessione corrente e si collega a *server_name*[*****instance_name*]
- **:ED** : lancia l'editor di testo (di default è edit) e apre il batch appena eseguito

sqlcmd

- Esempi di comandi:

  USE AdventureWorks

  GO
- Effetto

  Changed database context to 'AdventureWorks'.
- Comando

  SELECT * from HumanResources.Department

  GO
- Effetto: mostra le righe della tabella Department

sqlcmd

- Opzioni della riga di comando
- -i *input_file*[**,***input_file2*...] : Legge ed esegue i comandi presenti nei file specificati nell'ordine in cui appaiono i file. Non usare spazi tra i nomi dei file. Se uno dei file non esiste sqlcmd esce. Se il nome del file contiene spazi usare le doppie virgolette
- -o *output_file* : Scrive l'output dei comandi sul file *output_file.*
- Esempio:
  - sqlcmd –i "prova 1.sql" –o "prova 1.out"

Visual Studio

1. Server Explorer-> Data Connections -> servedb.AdventureWorks.dbo-> New Query
2. E’ possibile comporre una query (select) in maniera simile ad Access

Query Editor

1. Per usare il query editor grafico con istruzioni INSERT, DELETE e UPDATE scegliere Change Type

Soluzioni e progetti

- È possibile organizzare i propri script in SQL in soluzioni e progetti
- Un progetto è un insieme di file e scripts
- Una soluzione è un insieme di progetti
- Gli oggetti di una soluzione possono essere visualizzati utilizzando il Solution Explorer

Sicurezza

- SQL Server offre meccanismi basati sull'autenticazione per controllare gli accessi alle risorse
- Si basa sui *principals*: individui, gruppi o processi che possono richiedere risorse di SQL Server. Ogni principal ha un unico security identifier (SID)
- **Windows-level principals**
  - Windows Domain Login
  - Windows Local Login
- **SQL Server-level principal**
  - SQL Server Login: utente registrato in SQL Sever
- **Database-level principals**
  - Database User: utente registrato in SQL Sever
  - Database Role
  - Application Role

Autenticazione

- SQL Server offre due tipi di autenticazione quando ci si connette ad una istanza:
- Windows Authentication mode: l'autenticazione avviene tramite lo username e password di Windows. Usando Visual Studio, Management Studio o sqlcmd non occorre inserire username e password, ci si collega come l'utente con il quale si è loggati in Windows
- SQL Server and Windows Authentication mode: l'autenticazione avviene tramite l'utente con il quale si è loggati in Windows oppure tramite username e password associati ad un login di SQL Server

Autenticazione

- I login definiscono chi può collegarsi a SQL Server
  - Sono caratterizzati da nome (case sensitive) e password
  - Sono definiti a livello di istanza (si trovano nella cartella Security dell'istanza)
- I database user (user per semplicità)
  - Sono caratterizzati solo dal nome
  - Sono definiti a livello di database (si trovano nella cartella Security del database)
  - I diritti sulle risorse del database sono definiti in relazione agli user
  - Ogni user è associato ad uno ed un solo login

Autenticazione

- Per consentire l'accesso ad una istanza da parte di un utente Windows occorre
  - aggiungere un login (New Login)
  - Indicare il nome dell'utente Windows (nella forma Dominio\Utente)
  - Specificare il database di default (master di default)
  - Specificare (eventualmente) i ruoli a livello di istanza
  - Specficare gli user dei vari database associati a questo login (quando si seleziona un database viene creato automaticamente uno user con lo stesso nome)
  - Specificare l'eventuale schema di default per ciascun database

Autenticazione

- Per consentire l'accesso ad una istanza da parte di un utente SQL Sever occorre
  - aggiungere un login (New Login)
  - Indicare il nome dell'utente
  - Indicare la password
  - Specificare il database di default (master di default)
  - Specificare (eventualmente) i ruoli a livello di istanza
  - Specficare gli user dei vari database associati a questo login (quando si seleziona un database viene creato automaticamente uno user con lo stesso nome)
  - Specificare l'eventuale schema di default per ciascun database

Aggiunta di un nuovo user

- Se non si specifica quando si crea un login lo user collegato al login, lo si può aggiungere dopo dal database con New User
- Occorre indicare:
  - Il login a cui quello user è associato. Il login non deve essere già associato ad un altro utente del database
  - Gli schemi di cui lo user è proprietario
  - I database role a cui lo user appartiene
  - Gli eventuali diritti sulle risorse

Diritti

- Ogni user appartiene al database role **public**
- Se non si assegnano o proibiscono diritti specifici ad uno user, su una risorsa lo user eredita i permessi assegnati a public su quella risorsa
- Ogni database include tre user predefiniti:
  - **INFORMATION_SCHEMA** e **sys** sono richiesti da SQL Server e non possono essere modificati o cancellati.
  - **guest**: non può essere cancellato ma si può impedire che si connetta (REVOKE CONNECT FROM GUEST). È l'utente con il quale un login non associato ad uno user nel database può collegarsi al database

Database roles fissi

- SQL Server ha alcuni database role fissi
- Sono predefiniti e sono presenti in tutti i database
  - db_accessadmin
  - db_backupoperator
  - db_datareader
  - db_datawriter
  - db_ddladmin
  - db_denydatareader
  - db_denydatawriter
  - db_owner
  - db_securityadmin
  - public
- Utente ha il ruolo db_datareader sui database esistenti

Server roles fissi

- Sono roles a livello di server (ovvero di istanza)
  - bulkadmin
  - dbcreator
  - diskadmin
  - processadmin
  - securityadmin
  - serveradmin
  - setupadmin
  - sysadmin
- Utente ha il ruolo dbcreator: puo' creare database

Securables

- I securables sono le risorse alle quali SQL Server regola l'accesso
- Alcuni securables possono essere contenuti in altri, creando gerarchie chiamate **scopes**
- I securable scopes sono **server**, **database**, e **schema**.
- **Securable scope: Server**
  - Endpoint
  - Login
  - Database

Securables

- Securable scope: Database
  - User, Role, Application role, Assembly, Message Type, Route, Service, Remote Service Binding, Fulltext Catalog, Certificate, Asymmetric Key, Symmetric Key, Contract
- Securable scope: Schema
  - Type, XML Schema Collection, Object
  - Object:
    - Aggregate, Constraint, Function, Procedure, Queue, Statistic, Synonym, Table, View
- Inoltre diritti possono essere assegnati a principals su colonne di tabelle

Cancellazione di un principal

- Cancellazione di un login:
  - Un login che possiede securables o oggetti server-level non può essere cancellato
  - Non cancella gli user associati al login nei vari database
  - Gli user associati al login diventano “orfani”
- Cancellazione di uno user:
  - Uno user che possiede securables non può essere cancellato
  - Occorre prima trasferire la proprietà dei securables

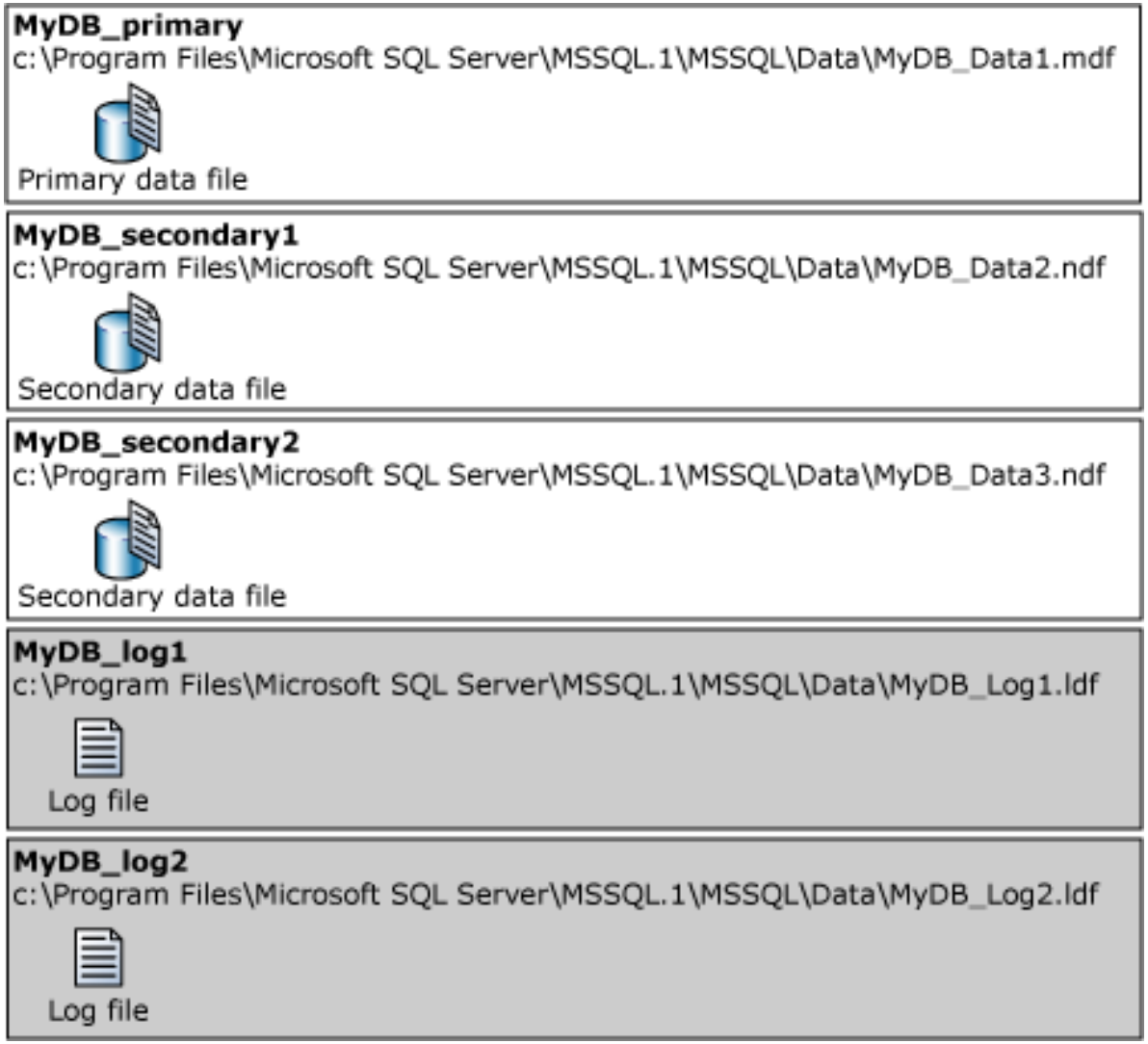
Organizzazione fisica

- SQL Server memorizza i dati in files del sistema operativo
- Usa tre tipi di file
  - Primary data files: ogni database ha un solo primary data file. È il punto di inizio del database e punta ai secondary data files. L'estensione di file raccomandata è .mdf
  - Secondary data files: contengono i dati non contenuti nel primary data file. Un database può avere zero o più secondary data files. L'estensione di file raccomandata è .ndf.
  - Log files: contengono il log del database. Ci deve essere almeno un file di log per database, anche se ce ne possono essere più di uno. L'estensione di file raccomandata è .ldf

Nomi logici e fisici

- I file di SQL Server 2005 hanno due nomi:
- **logical_file_name**: il nome usato per riferirsi al file fisico in tutti i comandi Transact-SQL. Il nome logico deve rispettare le regole per gli identificatori di SQL Server e deve essere unico tra i nomi logici di file nel database
- **os_file_name**: il nome del file fisico incluso il path assoluto. Deve seguire le regole del sistema operativo per i nomi di file.

Nomi logici e fisici



Posizione dei file

- Quando istanze multiple di SQL Server sono eseguite su un singolo computer, ogni istanza riceve una differente directory di default dove mettere i file per i database creati nell'istanza.
- I file comuni a tutte le istanze sono messi in C:\Programmi\Microsoft SQL Server\90\
- Durante l'installazione di una nuova istanza, viene generato un instance ID che la identifica
- Gli instance ID sono della forma MSSQL.n, dove n è un numero ordinale dell'istanza installata

Posizione dei file

- La prima istanza ha instance ID MSSQL.1, le altre MSSQL.2, MSSQL.3, …
- L'istanza di default viene messa in MSSQL.1
- La directory di default di una istanza di ID MSSQL.n è C:\Programmi\SQL Server\MSSQL.n
- I file dell'istanza sono poi messi in una sottocartella che prende il nome dal tipo di istanza
  - MSSQL per Database Engine
  - OLAP per Analysis Services
  - RS per Reporting Services

Posizione dei file

- Ad esempio, una installazione tipica comprende tre istanze
- C:\Programmi\Microsoft SQL Server\MSSQL.1\MSSQL\
- C:\Programmi\Microsoft SQL Server\MSSQL.2\OLAP\
- C:\Programmi\Microsoft SQL Server\MSSQL.3\RS\

Posizione dei file

- Di default i file di dati di una istanza di Database Engine sono messi in
- C:\Programmi\Microsoft SQL Server\MSSQL.*n* \MSSQL\Data\
- Alla creazione di un database il numero e la posizione dei file di dati può essere specificata dall'utente
- Ad esempio, si possono mettere i file di log su un disco diverso da quelli di dati, in modo che se il disco che contiene i dati ha un guasto il file di log si salvi

Filegroup

- I file e gli oggetti di database possono essere raggruppati in filegroups per ragioni di allocazione e amministrazione
- Ci sono due tipi di filegroups
  - Primary: coniene il data file primario e tutti gli altri file non specificamente assegnati ad altri filegroup. Tutte le pagine per le tabelle di sistema sono allocate nel filegroup primario
  - User-defined
- I file di log non appartengono ai filegroup

Filegroup

- Tabelle, indici e oggetti grandi possono essere associati ad un filegroup specifico.
- In tal caso tutte le loro pagine saranno allocate in quel filegroup
- Altrimenti le tabelle e gli indici possono essere partizionati: sono divisi in unità che possono essere messi in filegroups separati
- Un filegroup in ogni database è specificato come il filegroup di default.
- Quando una tabella o un indice sono creati senza specificare un filegroup, tutte le pagine sono messe nel filegroup di default
- Membri del gruppo **db_owner** possono cambiare il filegroup di default

Dimensioni dei file

- Alla creazione, i file hanno una dimensione
- Possono aumentare la loro dimensione automaticamnete
- Alla creazione di un file, si può specificare l'incremento di dimensione
- Ogni volta che tutti i files di un filegroup sono riempiti, la dimensione viene aumentata dell'incremento ad uno dei file del filegroup a turno
- Ogni file può avere anche una dimensione massima
- Se non è specificata, il file può continuare a crescere finchè non ha utilizzato tutto lo spazio su disco

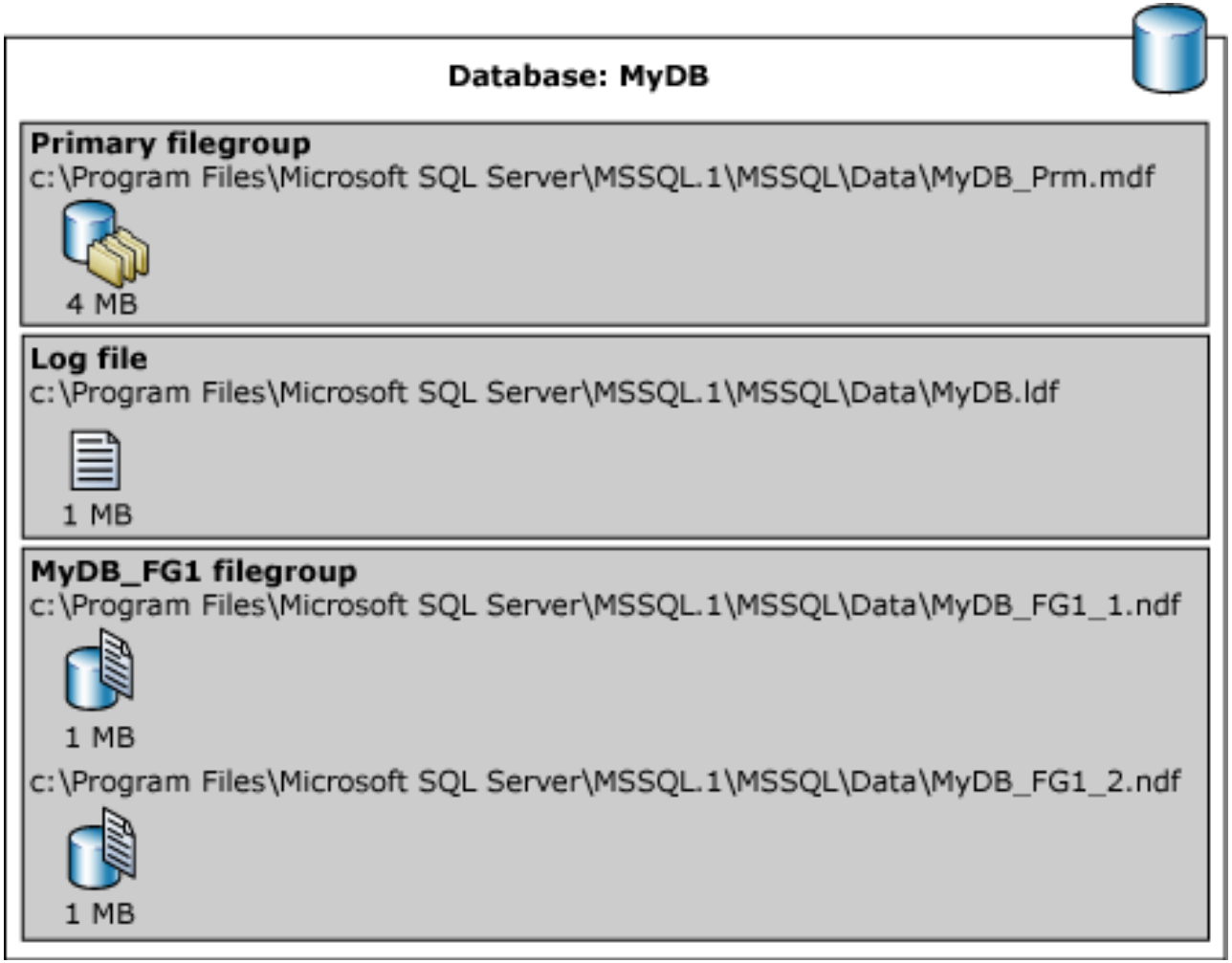
Esempio

```
USE master;
GO
-- Create the database with the default data filegroup and a log file. Specify the
-- growth increment and the max size for the primary data file.
CREATE DATABASE MyDB
ON PRIMARY
  ( NAME='MyDB_Primary',
    FILENAME=
       'c:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\data\MyDB_Prm.mdf',
    SIZE=4MB,
    MAXSIZE=10MB,
    FILEGROWTH=1MB),
FILEGROUP MyDB_FG1
  ( NAME = 'MyDB_FG1_Dat1',
    FILENAME =
       'c:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\data\MyDB_FG1_1.ndf',
    SIZE = 1MB,
    MAXSIZE=10MB,
    FILEGROWTH=1MB),
```

Esempio

```
( NAME = 'MyDB_FG1_Dat2',
    FILENAME =
      'c:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\data\MyDB_FG1_2.ndf',
    SIZE = 1MB,
    MAXSIZE=10MB,
    FILEGROWTH=1MB)
LOG ON
  ( NAME='MyDB_log',
    FILENAME =
      'c:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\data\MyDB.ldf',
    SIZE=1MB,
    MAXSIZE=10MB,
    FILEGROWTH=1MB);
GO
ALTER DATABASE MyDB
  MODIFY FILEGROUP MyDB_FG1 DEFAULT;
GO
-- Create a table in the user-defined filegroup.
USE MyDB;
CREATE TABLE MyTable
  ( cola int PRIMARY KEY,
    colb char(8) )
ON MyDB_FG1;
GO
```

Esempio


Database: MyDB
Primary filegroup
c:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\MyDB_Prm.mdf
4 MB
Log file
c:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\MyDB.ldf
1 MB
MyDB_FG1 filegroup
c:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\MyDB_FG1_1.ndf
1 MB
c:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\MyDB_FG1_2.ndf
1 MB

Esempio: Creazione di un proprio DB

- Creare un db su SERVEDB con il proprio numero di matricola come nome:
- Data-> New Query

```
USE [master]
GO
CREATE DATABASE [matr]
GO
```

- Creare una connessione: tasto dx Data Connections-> Add connection..

Tipi di dato

- Ogni colonna, variabile locale, espressione e parametro ha un tipo di dato
- Un tipo di dato può essere di sistema o definito dall'utente in Transact-SQL o nel .NET Framework (questi ultimi si chiamano CLR User-defined Types).
- I CLR User-defined Types ottengono le loro caratteristiche dai metodi e dagli operatori della classe che si crea usando uno dei linguaggi supportati dal framework.

Categorie di tipi di dato di sistema

- Exact numerics
- Approximate numerics
- Character strings
- Unicode character strings
- Binary strings
- Date and time
- Altri tipi di dato

Exact numerics

Tipo	Range	Dimensione (bytes)
bigint	da -2^63 a 2^63-1	8
int	da -2^31 a 2^31-1	4
smallint	da -2^15 a 2^15-1	2
tinyint	da 0 a 255	1

Exact numerics

- Tipi di dato con precisione e scala fissi:
  - **decimal**[**(***p*[**,** *s*] **)**] e **numeric**[**(***p*[**,** *s*] **)**]
  - I tipi SQL-92 **dec** e **dec(***p*, *s***)** corrispondono a **decimal**.
  - **numeric** è funzionalmente equivalente a **decimal**.
- *p* (precisione): il numero massimo di cifre decimali che possono essere memorizzate, sia a sinistra che a destra della virgola. Può andare da 1 a 38. Il default è 18. Il range in caso di massima precisione è da -10^38 +1 a 10^38 - 1
- *s* (scala): il massimo numero di cifre decimali che possono essere memorizzate a destra del punto decimale. *s* deve andare da 0 a *p*. La scala di defult è 0

Exact numerics

- Occupazione di memoria di **decimal** e **numeric**

Precisione	Dimensione (bytes)
1-9	5
10-19	9
20-28	13
29-38	17

Exact numerics

- **money** e **smallmoney**: rappresentano importi di denaro. Precisione: 10^{-4}

Tipo di dato	Range	Dimensione (bytes)
money	da -922.337.203.685.477,5808 a 922.337.203.685.477,5807	8
smallmoney	da -214.748,3648 a 214.748,3647	4

Exact numerics

- **bit**: può assumere i valori 0, 1 o NULL
- SQL Server ottimizza la memorizzazione: se ci sono 8 o meno campi **bit** usa un byte, se ce ne sono da 9 a 16 usa due bytes e così via
- Le stringhe TRUE e FALSE possono essere convertite in bit: TRUE a 1 e FALSE a 0

Approximate numerics

- Rappresentano numeri floating point
- **float** [**(** *n* **)**] : *n* è il numero di bit da usare per memorizzare la mantissa. Può andare da 1 a 53. Il valore di default è 53
- **real** è sinomimo di **float(24)**

n	Precisione (cifre)	Dimensione (bytes)
1-24	7	4
25-53	15	8

- Il tipo SQL-92 **double precision** corrisponde a **float(53)**
- SQL Server tratta *n* in questo modo: se 1<=n<=24, *n* è trattato come 24, se 25<=*n*<=53, *n* è trattato come 53. Aderisce allo standard SQL-92

Approximate numerics

Tipo	Range	Dimensione (bytes)
float	da -1,79E+308 a -2,23E-308, 0 e da 2,23E-308 a 1,79E+308	4 o 8, dipende da n
real	da -3,40E+38 a -1,18E - 38, 0 e da 1,18E-38 a 3,40E+38	4

Date and time

- **datetime** e **smalldatetime**
  - Rappresentano la data e l'ora del giorno

Tipo	Range	Accuratezza
datetime	Dall'1 gennaio 1753, al 31 dicembre 9999	3,33 millisecondi
smalldatetime	Dall'1 gennaio 1900, al 6 giugno 2079	1 minuto

Date and time

- **datetime**: rappresentato come due interi di 4 byte
  - Il primo intero memorizza il numero di giorni prima o dopo la data di base: 1 gennaio 1900 (la data di base è la data di riferimento di sistema)
  - Il secondo intero memorizza il numero di millisecondi dalla mezzanotte
- **smalldatetime**: rappresentato come due interi di 2 byte
  - Il primo intero memorizza il numero di giorni dopo l'1 gennaio 1900
  - Il secondo intero memorizza il numero di minuti dalla mezzanotte

Character strings

- **char** [(*n*)] : stringa di lunghezza fissa, non-Unicode, di lunghezza *n* bytes. *n* può andare da 1 a 8.000. L'occupazione di memoria è di *n* bytes.
- **varchar** [(*n* | **max**)] : stringa di lunghezza variabile, non-Unicode. *n* può andare da 1 a 8.000. **max** indica che la massima occupazione di memoria è 2^{31}-1 bytes. L'occupazione di memoria è pari alla lunghezza della stringa + 2 bytes. La stringa memorizzata può essere lunga 0
- I tipi SQL-2003 **char varying** o **character varying** corrispondono a **varchar**
- *n* di default vale 1

Character Strings

- **text**: stringhe non-Unicode di lunghezza variabile con una lunghezza massima di 2^{31}-1 bytes
- Deprecato, sarà rimosso in future versioni di SQL Server
- Usare **varchar(max)** al suo posto

Character strings

- Usare **char** quando la dimensione dei dati nella colonna è più o meno sempre la stessa
- Usare **varchar** quando la dimensione dei dati nella colonna varia molto
- Usare **varchar(max)** when quando la dimensione dei dati nella colonna varia molto e può eccedere gli 8.000 bytes

Unicode character strings

- Stringhe che usano il set di caratteri UNICODE UCS-2
- **nchar** [(*n*)] : stringa di lunghezza fissa, Unicode, di lunghezza *n*. *n* può andare da 1 a 4.000. L'occupazione di memoria è di 2*n* bytes. I tipi SQL-2003 **national char** and **national character** corrispondono a **nchar**
- **nvarchar** [(*n* | **max**)] : stringa di lunghezza variabile, Unicode. *n* può andare da 1 a 4.000. **max** indica che la massima occupazione di memoria è 2^{31}-1 bytes. L'occupazione di memoria è pari alla lunghezza della stringa per 2 + 2 bytes. La stringa memorizzata può essere lunga 0. I tipi SQL-2003 **national char varying** e **national character varying** corrispondono a **nvarchar**
- Il valore di default di *n* è 1

Unicode character strings

- **ntext**: stringhe Unicode di lunghezza variabile con una lunghezza massima di 2^{30}-1 caratteri.
- L'occupazione di memoria in bytes è 2 per il numero di caratteri
- Il tipo SQL-2003 **national text** corrisponde a **ntext**.
- Deprecato, sarà rimosso in future versioni di SQL Server
- Usare **nvarchar(max)** al suo posto

Binary strings

- **binary** [(*n*)] : dati binari di lunghezza fissa con una lunghezza di *n* bytes, dove *n* va da 1 a 8.000. L'occupazione di memoria è *n* bytes
- **varbinary** [(*n* | **max**)] : dati binari di lunghezza variabile. *n* va da 1 a 8.000. **max** indica che la massima occupazione di memoria è di 2^{31}-1 bytes. L'occupazione di memoria è la lunghezza dei dati inseriti + 2 bytes. I dati inseriti possono essere lunghi 0. Il tipo SQL-2003 **binary varying** corrisponde a **varbinary**
- Il valore di default di *n* è 1

Binary strings

- **image**: dati binari di lunghezza variabile da 0 a 2^{31}-1 bytes.
- Deprecato, sarà rimosso in future versioni di SQL Server
- Usare **varbinary(max)** al suo posto

Altri tipi di dato

- **timestamp**: serve a contenere numeri binari unici generati automaticamente. Sono generalmente usati per assegnare una versione alle righe di una tabella. L'occupazione di memoria è di 8 bytes.
- Ogni database ha un contatore che è incrementato per ogni insert o update su una tabella del database che contiene una colonna timestamp. Questo contatore ha livello di database e tiene traccia di un tempo relativo nel database.
- Una tabella può avere solo una colonna timestamp. Ogni volta che una riga con una colonna timestamp viene modificata o inserita, il timestamp di database incrementato viene inserito nella colonna timestamp
- Per questa ragione una colonna timestamp non va bene come chiave primaria

Altri tipi di dato

- **timestamp**: si può usare una colonna timestamp di una riga per determinare se la riga è stata modificata dall'ultima volta che è stata letta. Infatti se è stata modificata, il valore del timestamp sarà diverso da quello che aveva quando è stata letta
- **timestamp** di Transact-SQL è diverso da timestamp di SQL-2003. **timestamp** di SQL-2003 corrisponde a **datetime**.

Altri tipi di dato

- **sql_variant**: memorizza valori di alcuni altri tipi. È un tipo generale
- Può essere usato in colonne, parametri, variabili e valori di ritorno di funzioni definite dall'utente
- Una colonna di tipo **sql_variant** può contenere valori di altri tipi di dato. Ad esempio, una colonna **sql_variant** può memorizzare valori **int**, **binary** e **char**.
- **sql_variant** non può memorizzare dati di tipo **varchar(max)**, **varbinary(max)**, **nvarchar(max)**, **xml**, **timestamp**, **sql_variant,** tipi definiti dall'utente

Altri tipi di dato

- **sql_variant** può avere una lunghezza massima di 8016 bytes. Questo include sia le informazioni sul tipo base che sul valore. La lunghezza massima del valore è di 8.000 bytes.
- Un dato **sql_variant** deve prima essere convertito nel suo tipo base prima di partecipare ad operazioni come addizioni e sottrazioni
- A una colonna **sql_variant** può essere assegnato un valore di default.
- **sql_variant** può avere NULL come valore, ma in questo caso non ha associato un tipo base
  Una colonna sql_variant può essere una chiave primaria o straniera o un campo unique, ma la lunghezza dei valori di chiave non deve essere superiore a 900 bytes, che è la dimensione massima di una chiave in un indice

Altri tipi di dato

- **cursor**: tipo di dato per variabili o parametri di uscita di stored procedures che contiene un riferimento ad un cursore
- Le variabili di tipo **cursor** possono contenere il valore NULL

Altri tipi di dato

- **uniqueidentifier**: un identificatore di 16 byte, un Globally Unique Identifier (GUID).
- Una colonna o variabile local di tipo **uniqueidentifier** può essere inizializzata nei modi seguenti:
  - Usando la funzione NEWID.
  - Con la conversione da una costante stringa della forma xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx dove ogni x è una cifra esadecimale. Ad esempio, 6F9619FF-8B86-D011-B42D-00C04FC964FF è un valore valido per **uniqueidentifier**.

Altri tipi di dato

- **table**: tipo usato per variabili locali e valori di ritorno di funzioni. Le variabili locali di tipo table servono a contenere un result set per processarlo successivamente. Serve generalmente per contenere temporaneamente le righe restituite da una funzione che ritorna una tabella.
- Quando si usa il tipo table occorre fornire tutta la definizione della tabella come in una CREATE TABLE
- Una variable di tipo table può essere usata come una tabella regolare, ad esempio in SELECT, INSERT, UPDATE e DELETE

Altri tipi di dato

- **xml**: memorizza dati in XML. Si possono memorizzare istanze XML in colonne o variabili XML
- Sintassi

xml [**(** [CONTENT | DOCUMENT] *xml_schema_collection* **)**]

- CONTENT:l'istanza XML deve essere un frammento XML ben formato. Ci possono essere zero o più elementi al livello radice. Nodi testo sono ammessi al livello radice. Comportamento di default
- DOCUMENT: l'istanza XML deve essere un frammento XML ben formato. Deve avere uno ed un solo elemento radice. Nodi testo non sono ammessi al livello radice
- *xml_schema_collection:* nome di una XML schema collection. Serve a creare XML tipato

Classe

- *Large object data types (LOB)*: **text**, **ntext**, **image**, **varchar(max)**, **nvarchar(max)**, **varbinary(max)**, e **xml**

Creare una tabella nel proprio DB

- Scegliere Data->New query e copiare il testo sottostante

```
USE [matr]
GO
CREATE TABLE Department(
    DepartmentID smallint PRIMARY KEY,
    Name nvarchar(50) NOT NULL
    )
```

Creare una tabella nel proprio DB

- Scegliere Add New Table dal menu contestuale su Tables

```
Employee(
    EmployeeID int PRIMARY KEY,
      Name nvarchar(50) NOT NULL,
      Surname nvarchar(50) NOT NULL,
      DepartmentID smallint REFERENCES
    Department(DepartmentID)
)
```

Diagramma di database

- Aggiungere un nuovo database diagram con le tabelle appena create