Introduzione ad ADO.NET

Il presente: ADO.NET

- ADO .NET (ActiveX Data Object)
  - insieme di classi che forniscono tutte le funzionalità necessarie per l'accesso ai dati di un database e per la loro elaborazione in memoria.
  - distingue concettualmente tra l'accesso ai dati del database e la loro elaborazione.
  - il modello ad oggetti fornito è suddiviso in due parti, oggetti "connessi" ed oggetti "disconnessi", due tipologie di oggetti sono completamente distinte
- Oggetti connessi
  - vengono eseguite le operazioni di lettura e aggiornamento sul database
  - progettati per interfacciarsi con uno specifico DBMS
  - appartengono ad un determinato .NET Provider
- Oggetti disconnessi
  - consentono la memorizzazione e l'elaborazione dei dati nella memoria del programma client
  - completamente indipendenti dal database
  - non sono mai in comunicazione con esso

Modalità di accesso ai dati

Modalità di accesso ai dati

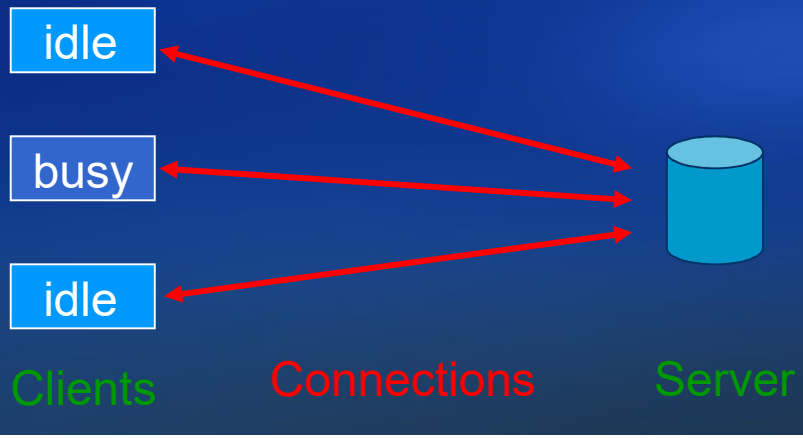
- Connesso: Forward-only, read-only
  - L'applicazione esegue una query, ottiene I risultati e li processa "immediatamente"
  - Oggetto DataReader
- Disconnesso
  - L'applicazione esegue una query, recuperà I dati e li memorizza localmente per l'elaborazione
  - Minimizza il tempo di connessione al database
  - `DataSet` object

Acceso Connesso ai Dati

- Nel modello client-server ogni cliente crea la propria connessione al DB all'avvio…

  …e la rilascia quando il programma termina.
- Il server deve mantenere una connessione attiva per ogni client…
  - …anche se la connessione è utilizzata per una piccola frazione di tempo.

Il modello connesso consuma risorse anche quando non sarebbe necessario



Acceso Connesso ai Dati

Vantaggi

- Si crea la connessione solo una volta
- Si impostano le proprietà della 'sessione‘ mediante la connessione
- Possibilità di effettuare lock e gestire la sicurezza se è previsto dal DB
- La prima riga di ogni query è 'immediatamente‘ disponibile

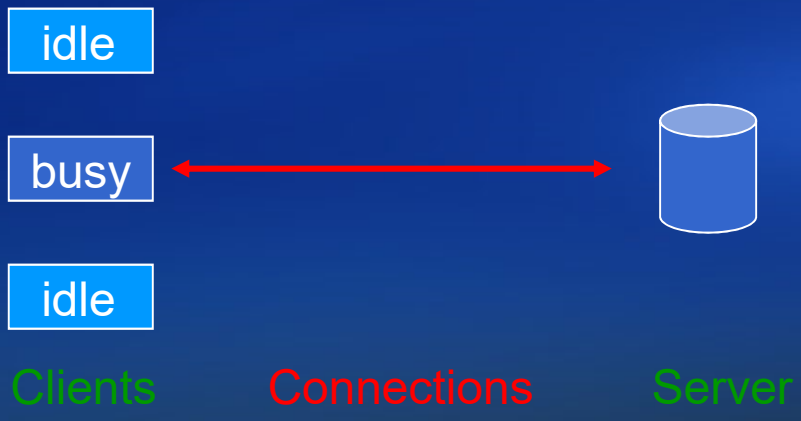
Svantaggi

- Le connessioni che tengono “aperto” il database consumano risorse

Accesso Disconnesso ai Dati

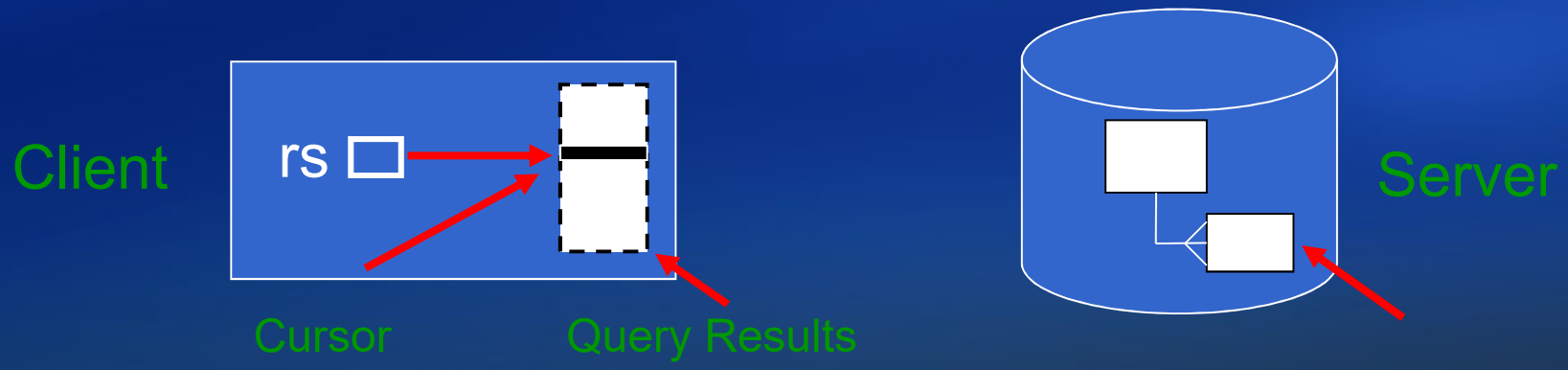
- Nel modello disconnesso, le connessioni al DB sono rilasciate appena possibile
- I Dati possono essere utilizzati anche dopo che la connessione è stata chiusa
  - La connessione viene ri-creata solo per scrivere delle modifiche nel server

Il modello disconnesso utilizza le risorse solo quando è necessario



Accesso Disconnesso ai Dati

- Tutti I dati sono inviati al client in un unica operazione
  - Il risultato delle query viene mantenuto in memoria
  - Tutte le risorse del server vengono rilasciate
- Il client manipola ed aggiorna i dati off-line
- Il client si ri-connette al database per scrivere gli aggiornamenti
  - Utilizzo della chiave primaria per iidentificare il record corretto



Accesso Disconnesso ai Dati

Vantaggi

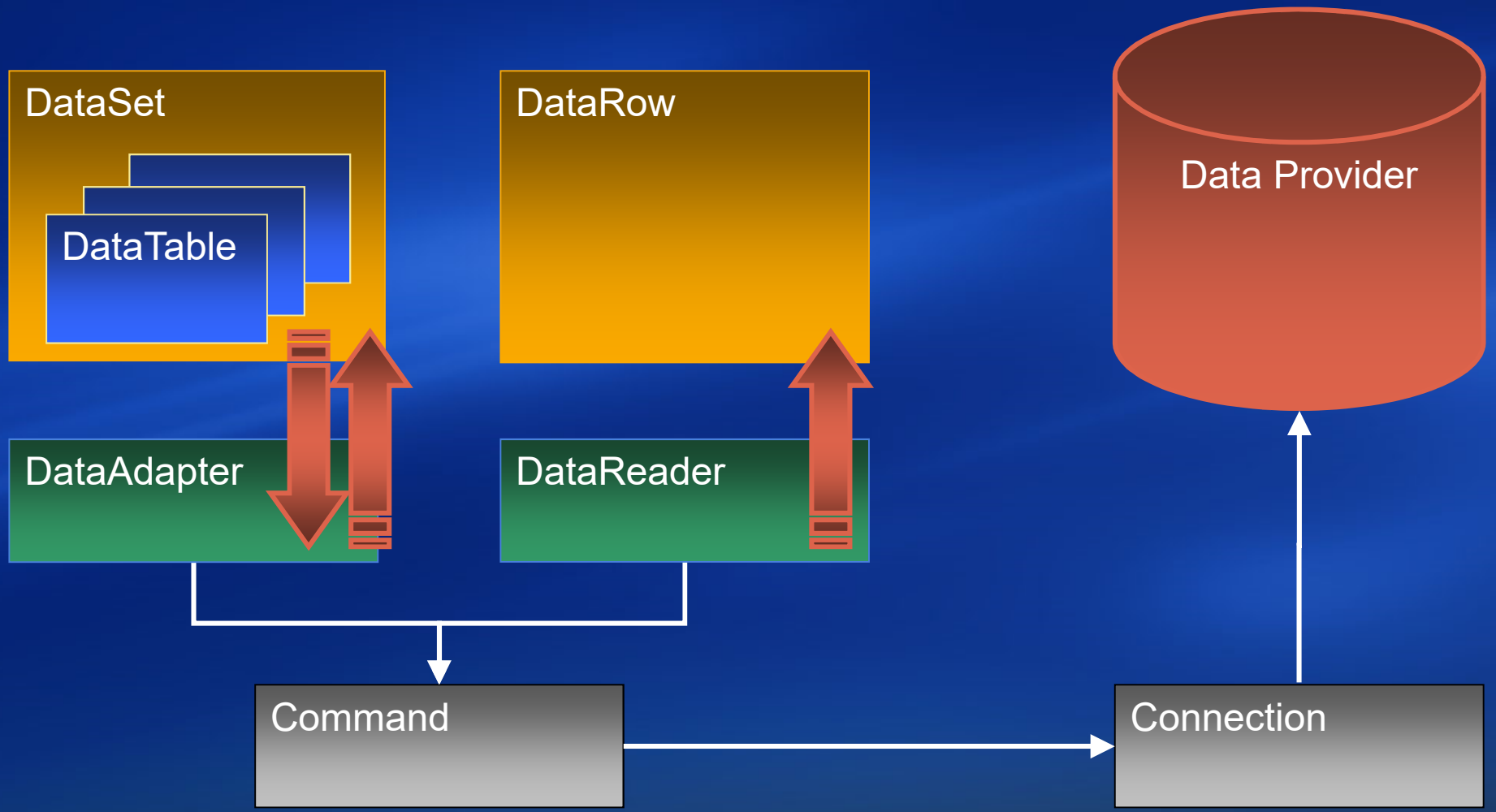
- Un singolo DB server può gestire più utenti
- La manipolazione dei dati è più veloce e flessibile
- I dati non sono "vincolati" alla connessione
  - facilità di passaggio tra I vari livelli o per scriverli su file
- Altamente indicato per app Web e n-tier

Svantaggi

- Aprire e chiudere la connessione può essere "costoso"
- Recuperare molte righe può essere molto lento
- Necessita di potenza lato client

Panoramica su ADO.NET

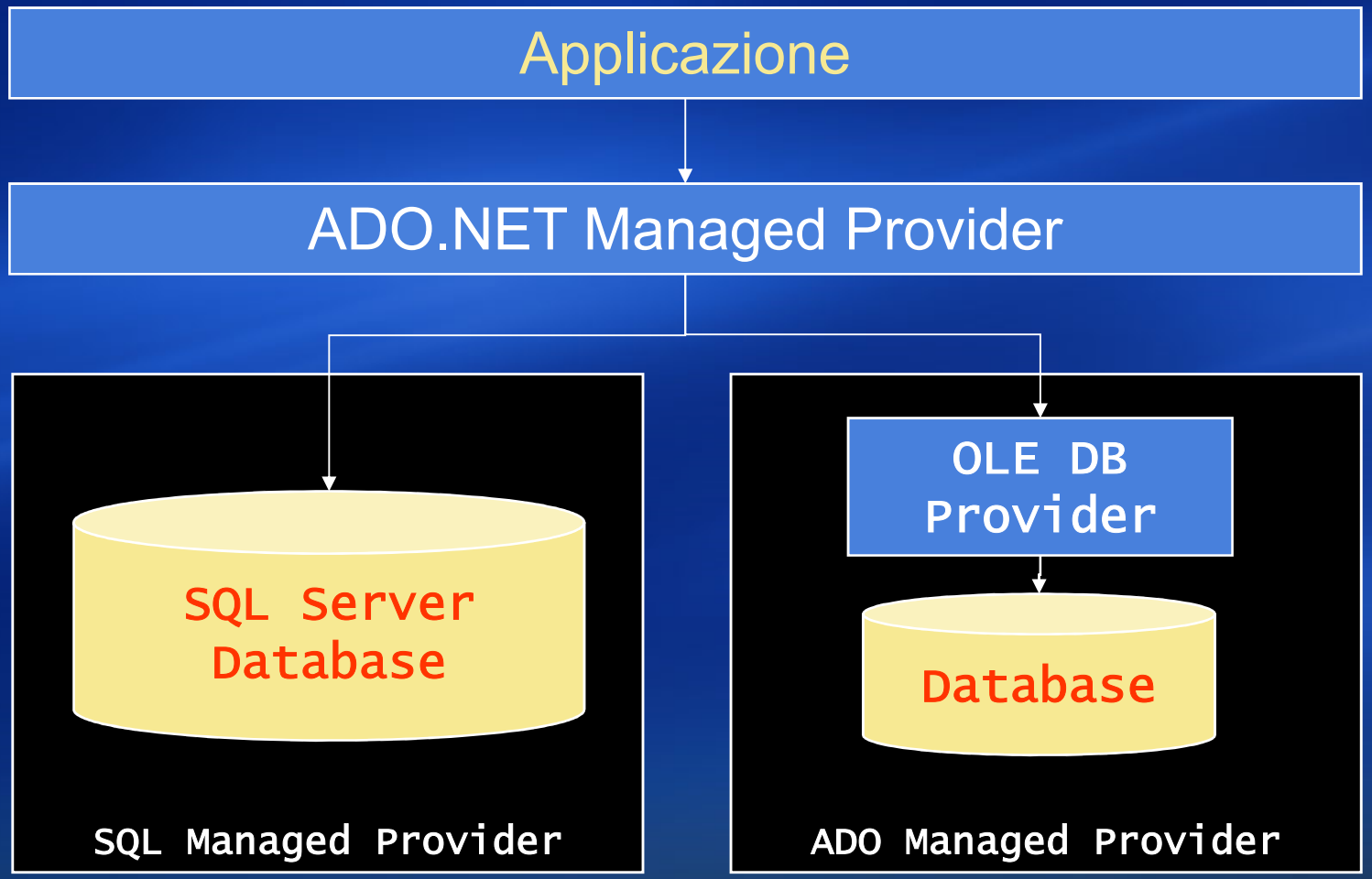
Panoramica su ADO .NET



Organizzazione di ADO.NET

- Due grandi aree:
  - .NET Data Providers
    - Un provider per ogni sorgente dati
    - Ogni provider ha il proprio namespace
    - Supporti per la programmazione “connessa”
    - Le Forms (WEB/WIN) fanno tra tramite tra DataSet e data source
  - System.Data namespace
    - Fornisce il modello ad oggetti del DataSet
    - Supporto alla programmazione disconnessa
    - Fornisce un modello di programmazione basato su interfacce per l’accesso “generico” ai managed providers

Managed Providers

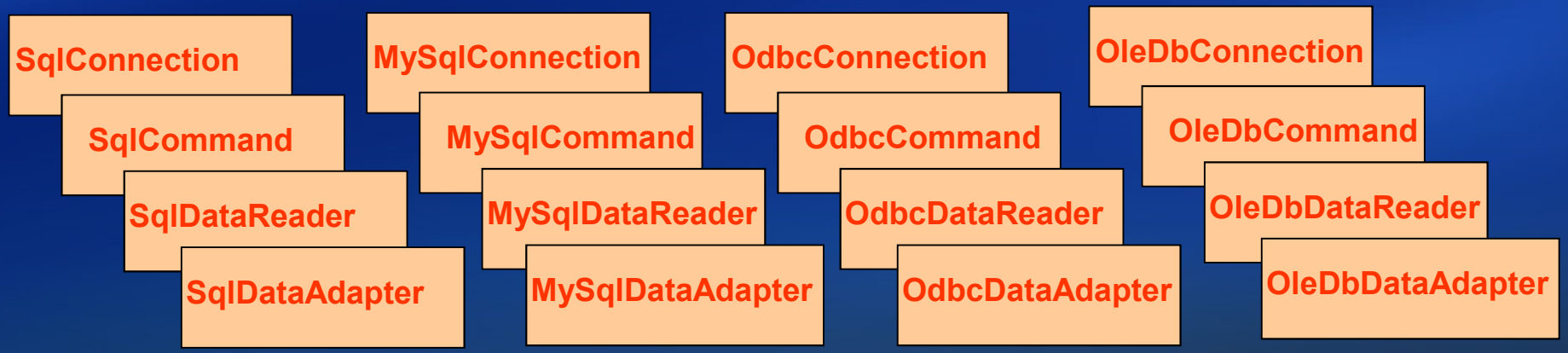


Modelli di programmazione

- ADO.NET fornisce il supporto per i due distinti modelli di programmazione
  - Connesso
    - Usa gli oggetti connessi Command e DataReader
    - Il DataReader è di tipo forward only e read-only
    - Può eseguire aggiornamenti mediante l'oggetto Command
  - Disconnesso
    - Utilizza i DataSets
    - I DataAdapters riempiono i DataSets con i dati ed eseguono le modifiche nel server
    - I DataSets sono indipendenti dal Provider
    - I DataSets sono memorizzati e serializzati come XML

.NET Data Providers

- I .NET Data Providers sono utilizzati per stabilire le connessioni con i DB
- Ecco alcuni Data Providers del .NET
  - SQL Data Provider (System.Data.SqlClient)
  - ODBC Data Provider (System.Data.Odbc)
  - OLEDB Data Provider (System.Data.OleDb)
  - MySql Data Provider (MySql.Data.MySqlClient)



Classi del .NET Data Provider

- Ogni Data Provider del .NET ha 4 classi

Classe	Descrizione
Connection	Stabilisce una connessione ad una specifica sorgente dati
Command	Esegue un comando verso la sorgente dati ha dei **Parameters** e può utilizzare una **Transaction** per una certa **Connection**
DataReader	Legge flussi di dati in modalità forward-only, read-only da una sorgente dati
DataAdapter	Popola un **DataSets** e risolve gli aggiornamenti con la sorgente dati

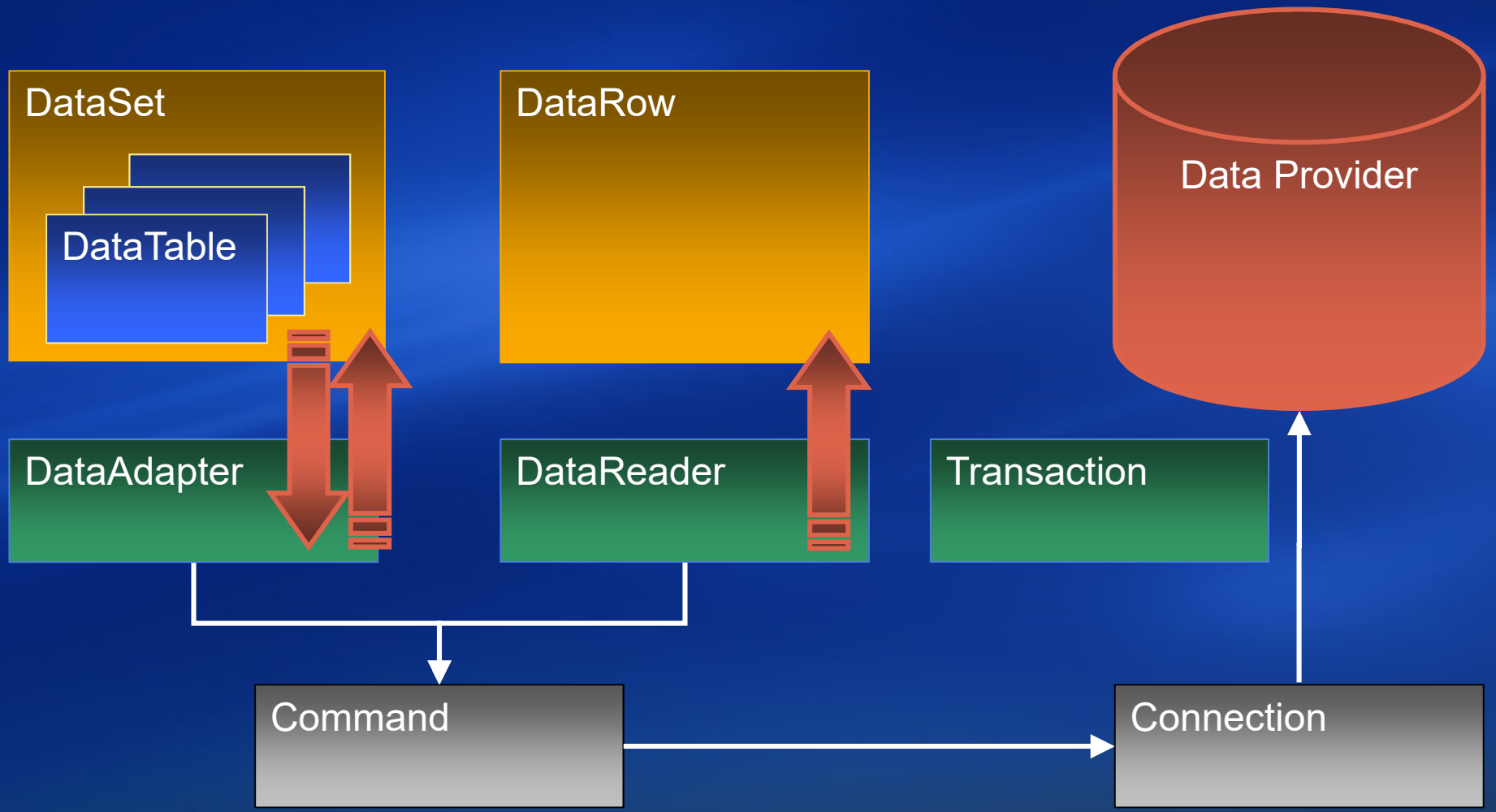
- Per esempio
  - Il provider per SQL Server implementa SqlConnection
  - The OLE DB data provider implements OleDbConnection

Riassumendo….

- Il namespace System.Data.SqlClient:
  - Classi per MS SQL Server 7 (e superiori)
  - SqlConnection, SqlAdapter, ...
- Il namespace System.Data.OleDb
  - Classi per il provider OLEDB
  - OleDbConnection, OleDbAdapter, …

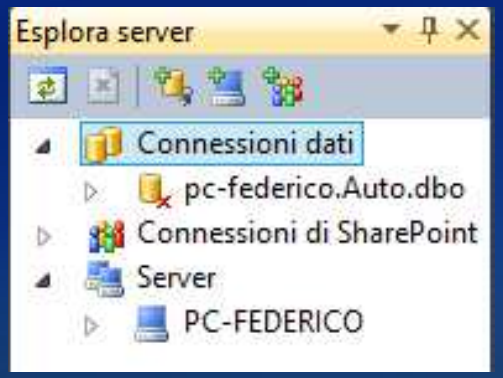
Accesso connesso

Effettuare una Connessione



ADO.NET

- - Attivazione server (Gestione configurazione sql server)
- -Inserimento del database (SQL Managment Studio)
- IDE Visual studio (visualizza: esplora server)



ADO SQLCONNECTION

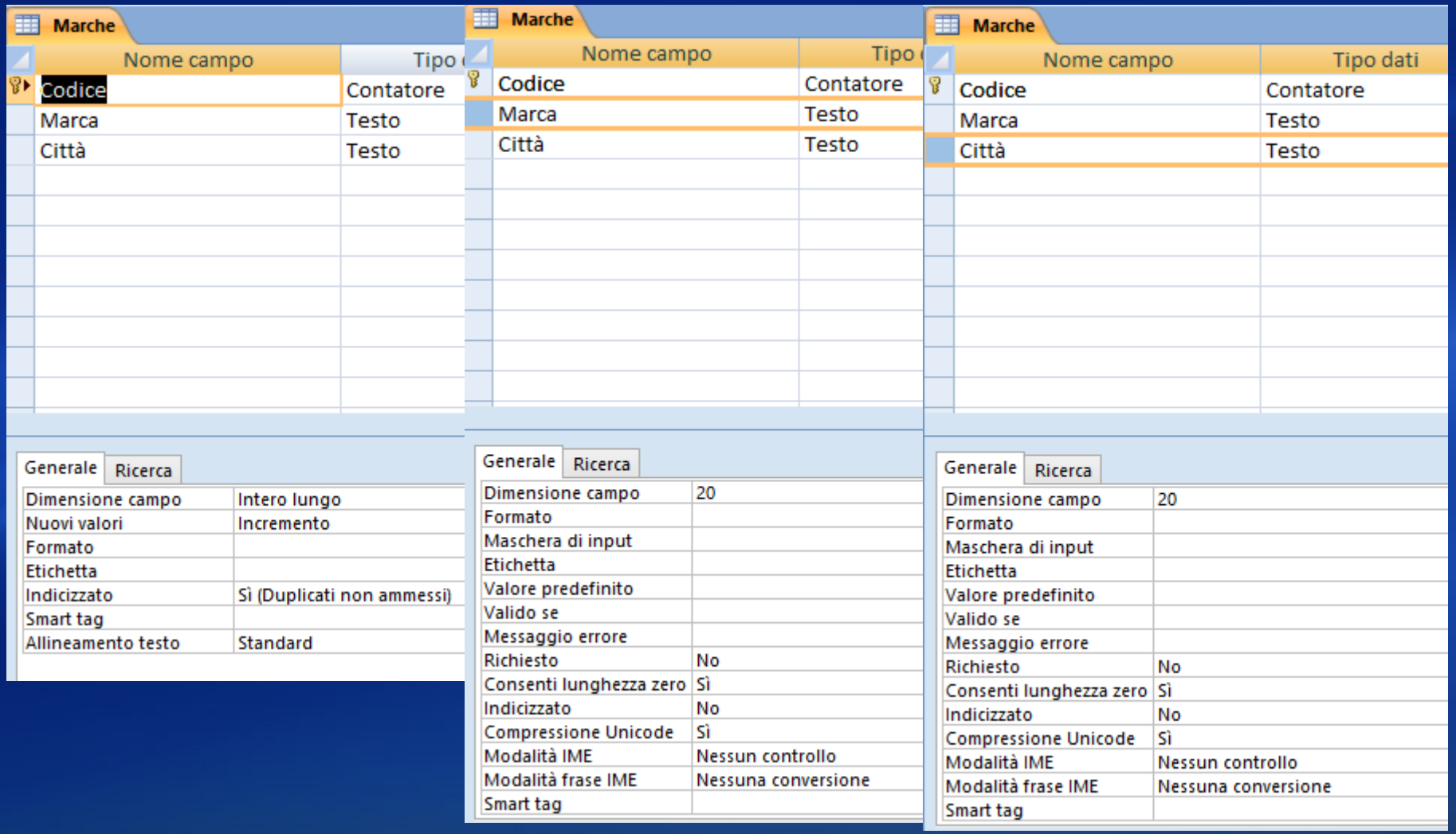
```
//crea l'oggetto Connection
        SqlConnection DBConn = new SqlConnection();

        //crea la stringa di connessione
        DBConn.ConnectionString = "Data Source =.\\SQLEXPRESS;
        Persist Security Info=false;  Elimina ID Utente e password dopo
    la connessione
        Trusted_Connection=Yes; Utilizzo sicurezza integrata
         DATABASE=auto";

        //apri la connessione
        DBConn.Open();


        //chiudi la connessione
        DBConn.Close();
```

DATABASE ACCESS

Marche

Codice	Marca	Città	*Aggiungi nuovo campo*
1	Lamborghini	Bologna	
2	Ferrari	Modena	
3	Fiat	Torino	
4	Lancia	Torino	
5	Alfa Romeo	Melfi	
(Nuovo)			

DATABASE ACCESS



ADO OLEDBCONNECTION

using System.Data.OleDb; //NET PROVIDER

Stringa di connessione

string cnstr=@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Database\"+"AccessAuto.accdb";

Conn.ConnectionString = @"File Name=C:\Users\informatica 201\Desktop\melon\Auto.udl";

Creazione della connessione (Esempio: Gestione Database Access)

cn = new OleDbConnection(cnstr);// istanza della connessione

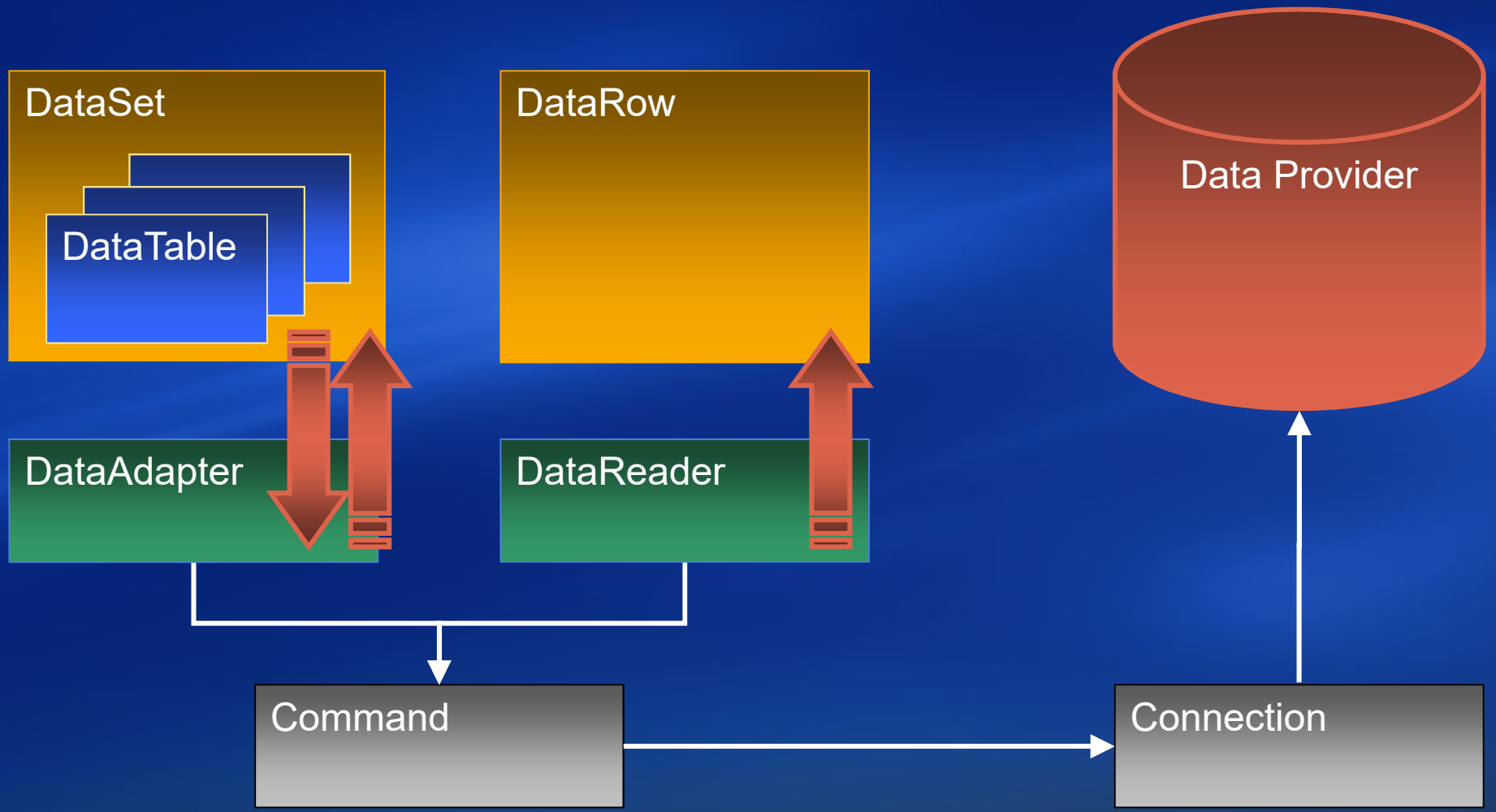
Creazione del commmand

string strSql = " Select count(*) from Negozi";

OleDbCommand cm;

cm = new OleDbCommand(strSql, cn);

Lettura dei dati



L’oggetto Command

- Una volta che si è creata una connesione si utilizza l’oggetto Command per eseguire istruzioni SQL o Stored Procedures

Proprietà e metodi

- CommandType
  - Text, StoredProcedure e TableDirect
- CommandText
  - query, nome di una tabella o di una stored procedure
- Parameters
  - Collezione di Parametri
- ExecuteReader()
  - Ritorna a DataReader
- ExecuteScalar()
  - Ritorna un singolo valore scalare
- ExecuteNonQuery()
  - Returna il numero di righe interessate
  - Utilizzarlo per operazioni di update e delete

Esempio

```
string cnStr =@"Provider=Microsoft.Jet.OLEDB.4.0;Data
    Source=C:\AmiciAGG.mdb;Persist Security Info=False";
OleDbConnection cn = new OleDbConnection(cnStr);
OleDbCommand cm = new OleDbCommand();
cm = new OleDbCommand("select nome from ragazzi where
città = ?",cn);
cm.Parameters.Add("@Raga", citta);
cn.Open();

    // fai qualcosa

cn.Close();
```

L’oggetto DataReader

- Si utilizza il DataReader per leggere delle DataRows
  - Cursore Forward only (Firehose)

Proprietà e Metodi

- Item
  - Collection of fields
- Read()
  - Reads the next DataRow, returns false when no rows are left
- Accesso mediante indicizzatori, e Metodi
  - [“Cognome”]
  - [0]
  - GetString()
  - GetDecimal()
  - GetBoolean()

Lavorare con i DataReaders

- Il DataReader fornisce solamente un cursore foreward-only, read-only
  - Utilizzarlo come fosse il classico Recordset ADO
  - Non lasciare aperto il DataReaders più del necessario
  - Usato correttamente i DataReaders sono efficienti
  - Eseguire gli aggioramenti usando oggetti Command con il metodo ExecuteNonQuery()
- Per una gestione flessibile degli aggiornamenti lato client, utilizzare …
  - DataSets e DataAdapters

Esempio

```
string cnStr =@"Provider=Microsoft.Jet.OLEDB.4.0;Data
    Source=C:\AmiciAGG.mdb;Persist Security Info=False";
OleDbConnection cn = new OleDbConnection(cnStr);
OleDbCommand cm = new OleDbCommand();
cm = new OleDbCommand("select nome from ragazzi where città =
    ?",cn);
cm.Parameters.Add("@LR", citta);
cn.Open();

OleDbDataReader dr = cm.ExecuteReader();

while (dr.Read() == true) // dr.Read() torna false se non ci sono
                          // più righe
{
    Console.WriteLine("Nome = ",dr["Nome"].ToString());
}
cn.Close();
```