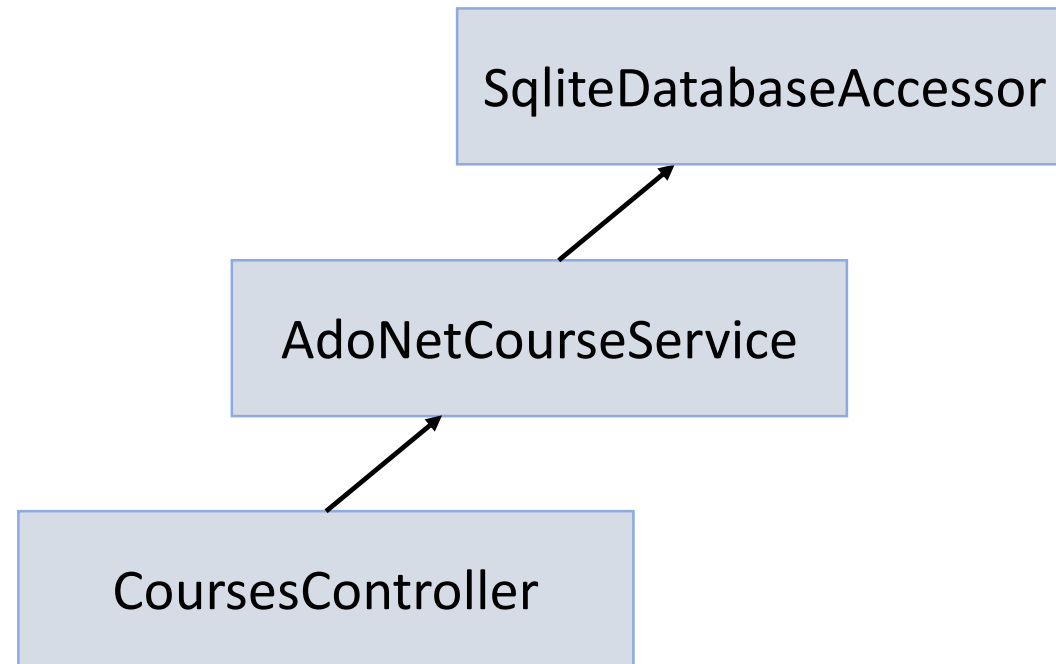


## Sezione 10

# Accedere al database con ADO.NET

# La nostra gerarchia di componenti



# Per iniziare con ADO.NET...

- ...per prima cosa procuriamoci il provider da NuGet per la tecnologia database che abbiamo scelto di usare.

Tecnologia database	Comando da lanciare per l'aggiunta del pacchetto NuGet
SQLite	<code>dotnet add package Microsoft.Data.SQLite</code>
SQL Server	<code>dotnet add package System.Data.SqlClient</code>
MySQL	<code>dotnet add package MySqlConnector</code>
Oracle	<code>dotnet add package Oracle.ManagedDataAccess.Core</code>
PostgreSQL	<code>dotnet add package Npgsql</code>

# Aggiornare un pacchetto NuGet

- Semplicemente ri-aggiungiamolo indicando la versione specifica.

```
dotnet add package Microsoft.Data.SQLite --version 2.2.2
```

# Alcune classi fornite dal provider per Sqlite

SqlConnection

Serve a stabilire una connessione al database

SqlCommand

Rappresenta una query o un comando SQL

SqlParameter

Lo usiamo per introdurre l'input dell'utente nelle nostre query o comandi (previene la SQL injection)

SqlDataReader

Serve a leggere il risultato restituito dal database

SqlTransaction

Crea un contesto in cui eseguire i comandi in isolamento e in maniera atomica (o tutto o niente).

# Alcune classi fornite dal provider per SQLServer

SqlConnection

SqlCommand

SqlParameter

SqlDataReader

SqlTransaction

# Alcune classi fornite dal provider per MySql

MySqlConnection

MySqlCommand

MySqlParameter

MySqlDataReader

MySqlTransaction

# Alcune classi fornite dal provider per Oracle

OracleConnection

OracleCommand

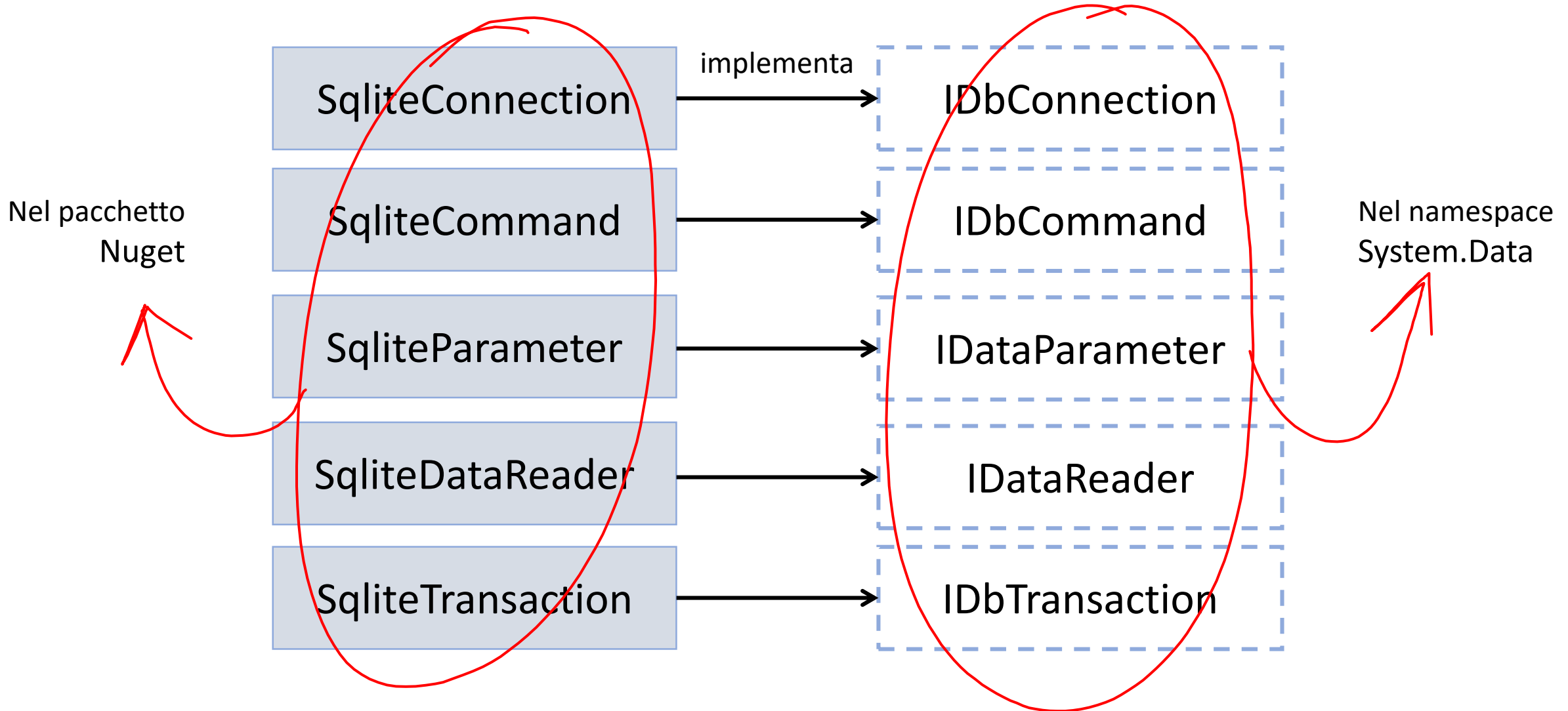
OracleParameter

OracleDataReader

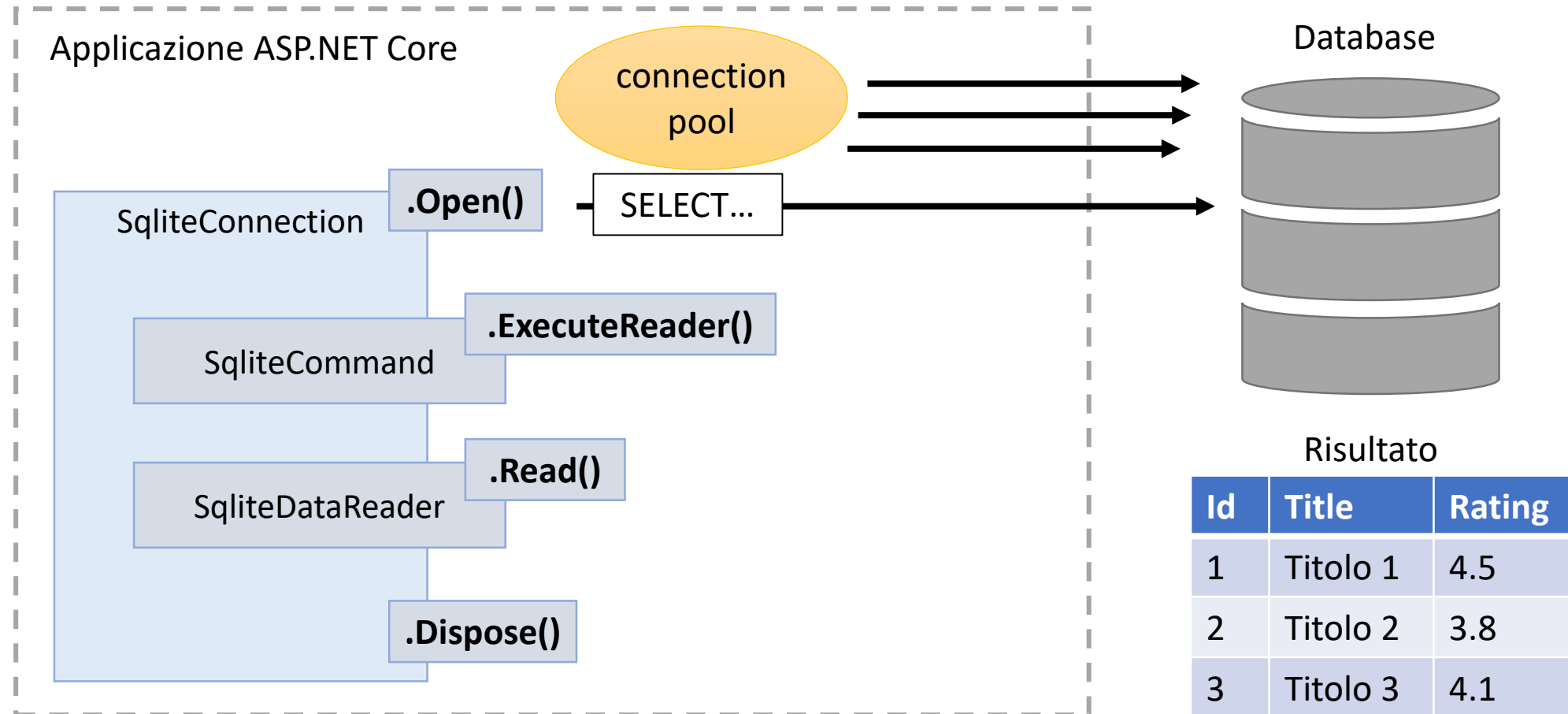
OracleTransaction



# Accoppiamento debole con le classi di ADO.NET



# Panoramica delle classi di ADO.NET



# Classi "connesse" e "disconnesse"

Classi  
"Connesse"  
Fornite dal provider

SqlConnection

SqlCommand

SqlParameter

SqlDataReader

SqlTransaction

DataTable

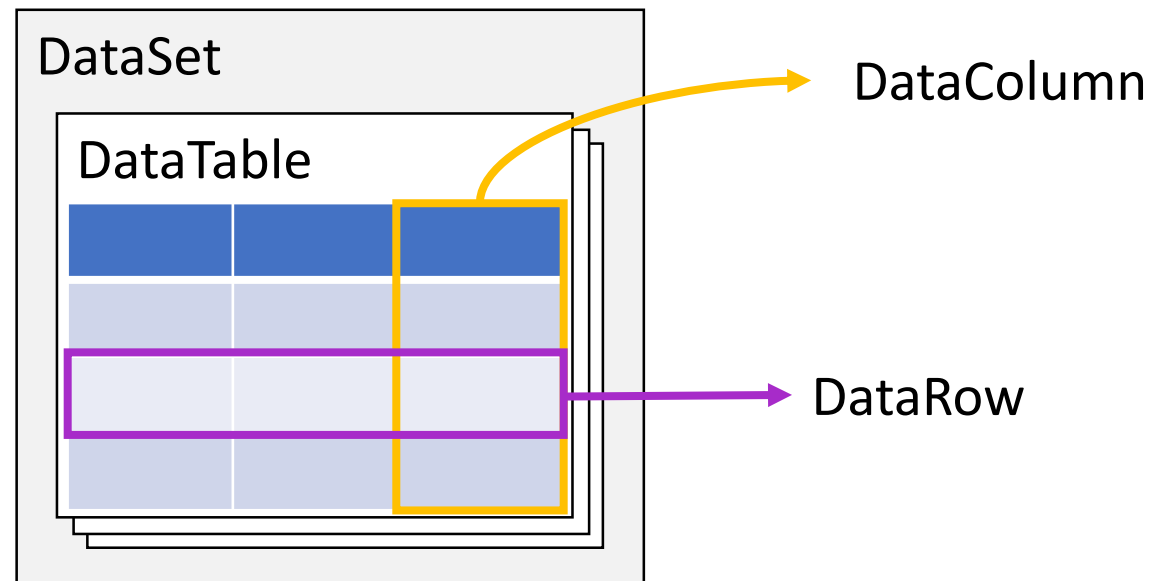
DataSet

Classi  
"Disconnesse"  
fornite da .NET Core

# DataSet e DataTable

Un **DataTable** è un oggetto che imita la tabella di un database e serve contenere dei risultati in memoria.

Un **DataSet** è una collezione di DataTable.



# Assicurarsi che le connessioni vengano chiuse

**No**

```
var conn = new SqlConnection("...");  
//...  
conn.Dispose();
```

**Sì**

```
using(var conn = new SqlConnection("..."))  
{  
    //...  
}
```

# Il blocco using

Il blocco **using** è necessario quando abbiamo oggetti che implementano `IDisposable` e che quindi possiedono il metodo `Dispose()`.

Solo in questo modo ci assicuriamo che questi oggetti verranno distrutti correttamente, anche quando dovesse verificarsi un'eccezione.

```
using(var cmd = new SqlCommand("...", conn))  
{  
    // ...  
}
```

Nota: quasi tutte le classi connesse di ADO.NET implementano `IDisposable`.


# Sql injection

```
var query = "SELECT * FROM Courses WHERE Id=" + id;  
var cmd = new SqlCommand(query, cmd);
```

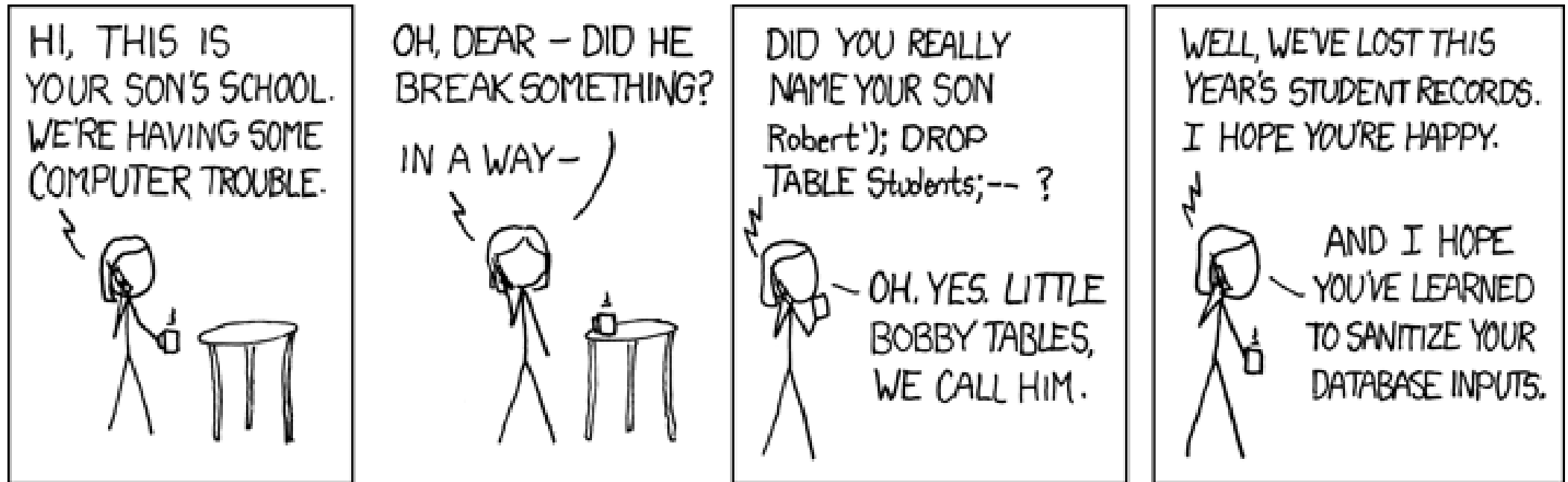


```
var query = "SELECT * FROM Courses WHERE Id=5";  
var cmd = new SqlCommand(query, cmd);
```

```
var query = "SELECT * FROM Courses WHERE Id=5; DROP TABLE Courses";  
var cmd = new SqlCommand(query, cmd);
```



# Sql injection





# Sql injection: preveniamola con i SqlParameter

**No**

```
var query = "SELECT * FROM Courses WHERE Id=" + id;  
var cmd = new SqlCommand(query, cmd);
```

**Sì**

```
var query = "SELECT * FROM Courses WHERE Id=@id";  
var cmd = new SqlCommand(query, conn);  
var parameter = new SqlParameter("id", id);  
cmd.Parameters.Add(parameter);
```

# Progresso nella specifica

- Punto 2: pagina di elenco dei corsi;
- Punto 5: pagina di dettaglio del corso.



View

Controller

Servizio applicativo

Servizio infrastrutturale

Database

Requisiti funzionali

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23		

+3000XP

Requisiti non funzionali

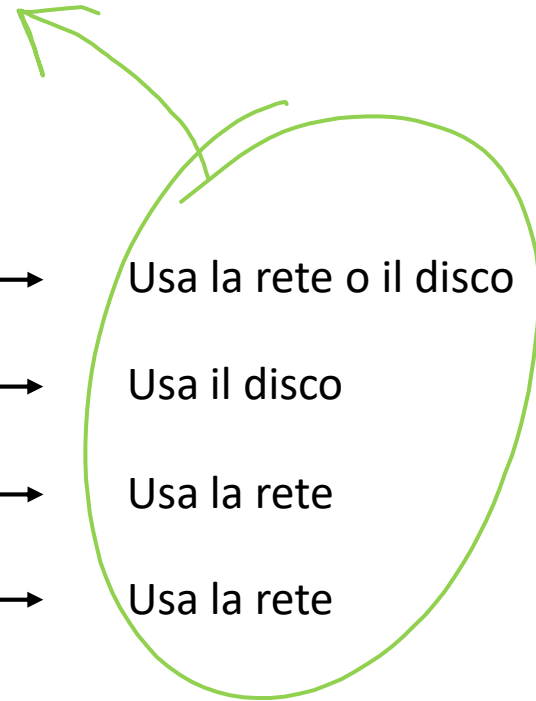
a	b	c	d
---	---	---	---

# Quando eseguire operazioni in maniera asincrona?

Quando dobbiamo ricorrere a periferiche di I/O.

Operazioni che ha senso eseguire in maniera asincrona

- Collegarsi e inviare query a un database → Usa la rete o il disco
- Leggere un file di testo → Usa il disco
- Inviare una richiesta a un webservice → Usa la rete
- Inviare un'e-mail → Usa la rete



# Quando NON eseguire operazioni asincrone?

Quando l'operazione fa uso della CPU.

Operazioni che non danno vantaggi nell'essere eseguite in maniera asincrona

- Calcolare le cifre del PI greco → Usa la CPU
- Ordinare gli oggetti in una List<T> → Usa la CPU
- Ridimensionare un'immagine → Usa la CPU

In questi casi il thread è sempre  
attivamente impegnato

# Usare le operazioni asincrone in tre passi

```
public DataSet Query(FormattableString query)
{
    using(var conn = new SqlConnection("..."))
    {
        conn.OpenAsync();
        // ...
    }
}
```

1

Usare **await** in corrispondenza del metodo asincrono

# Usare le operazioni asincrone in tre passi

```
public DataSet Query(FormattableString query)
{
    using(var conn = new SqlConnection("..."))
    {
        await conn.OpenAsync();
        // ...
    }
}
```

1

Usare **await** in corrispondenza del metodo asincrono

2

Aggiungere **async** alla firma del metodo

# Usare le operazioni asincrone in tre passi

```
public async DataSet Query(FormattableString query)
{
    using(var conn = new SqlConnection("..."))
    {
        await conn.OpenAsync();
        // ...
    }
}
```

- 1 Usare **await** in corrispondenza del metodo asincrono
- 2 Aggiungere **async** alla firma del metodo
- 3 Restituire un **Task<T>** o **Task**

# Usare le operazioni asincrone in tre passi

```
public async Task<DataSet> Query(FormattableString query)
{
    using(var conn = new SqlConnection("..."))
    {
        await conn.OpenAsync();
        // ...
    }
}
```

- 1 Usare **await** in corrispondenza del metodo asincrono
- 2 Aggiungere **async** alla firma del metodo
- 3 Restituire un **Task<T>** o **Task**



# Usare le operazioni asincrone in tre passi

```
public async Task<DataSet> QueryAsync(FormattableString query)
{
    using(var conn = new SqlConnection("..."))
    {
        await conn.OpenAsync();
        // ...
    }
}
```

- 1 Usare **await** in corrispondenza del metodo asincrono
- 2 Aggiungere **async** alla firma del metodo
- 3 Restituire un **Task<T>** o **Task**

```

1 function hell(win) {
2   // for listener purpose
3   return function() {
4     loadLink(win, REMOTE_SRC+'/assets/css/style.css', function() {
5       loadLink(win, REMOTE_SRC+'/lib/async.js', function() {
6         loadLink(win, REMOTE_SRC+'/lib/easyXDM.js', function() {
7           loadLink(win, REMOTE_SRC+'/lib/json2.js', function() {
8             loadLink(win, REMOTE_SRC+'/lib/underscore.min.js', function() {
9               loadLink(win, REMOTE_SRC+'/lib/backbone.min.js', function() {
10                loadLink(win, REMOTE_SRC+'/dev/base_dev.js', function() {
11                  loadLink(win, REMOTE_SRC+'/assets/js/deps.js', function() {
12                    loadLink(win, REMOTE_SRC+'/src/' + win.loader_path + '/loader.js', function() {
13                      async.eachSeries(SRIPTS, function(src, callback) {
14                        loadScript(win, BASE_URL+src, callback);
15                      });
16                    });
17                  });
18                });
19              });
20            });
21          });
22        });
23      });
24    });
25  };
26 }

```



Se non usiamo async/await: callback hell

<https://medium.com/ninjadevs/node-7-6-koa-2-asynchronous-flow-control-made-right-b0d41c6ba570>

# Promemoria dei metodi più usati di ADO.NET

Oggetto	Metodo	Cosa fa?
SqlConnection	OpenAsync()	Ottiene una connessione dal connection pool
SqlCommand	ExecuteReaderAsync()	Invia una query (SELECT) e restituisce un oggetto SqlDataReader usato per leggere i risultati.
SqlCommand	ExecuteNonQueryAsync()	Invia un comando al database (es. UPDATE) e restituisce il numero di righe interessate.
SqlCommand	ExecuteScalarAsync()	Invia una query (SELECT) e restituisce il valore trovato nella prima colonna della prima riga.
SqlDataReader	ReadAsync()	Indica al database che deve restituire la prossima riga di risultati. Restituisce true se la riga esiste.