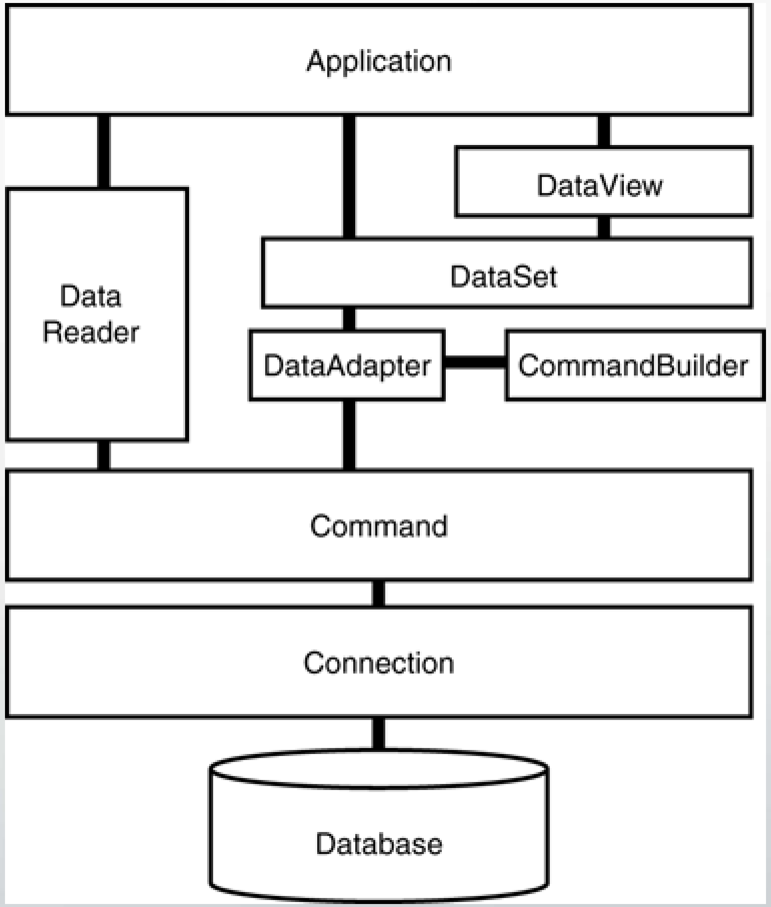
ORM

Object Relational Mapping

Omar El-Essiely

Data Access in .NET

- System.Data
  - IDbConnection
  - IDbTransaction
  - IDbCommand
  - IDataReader



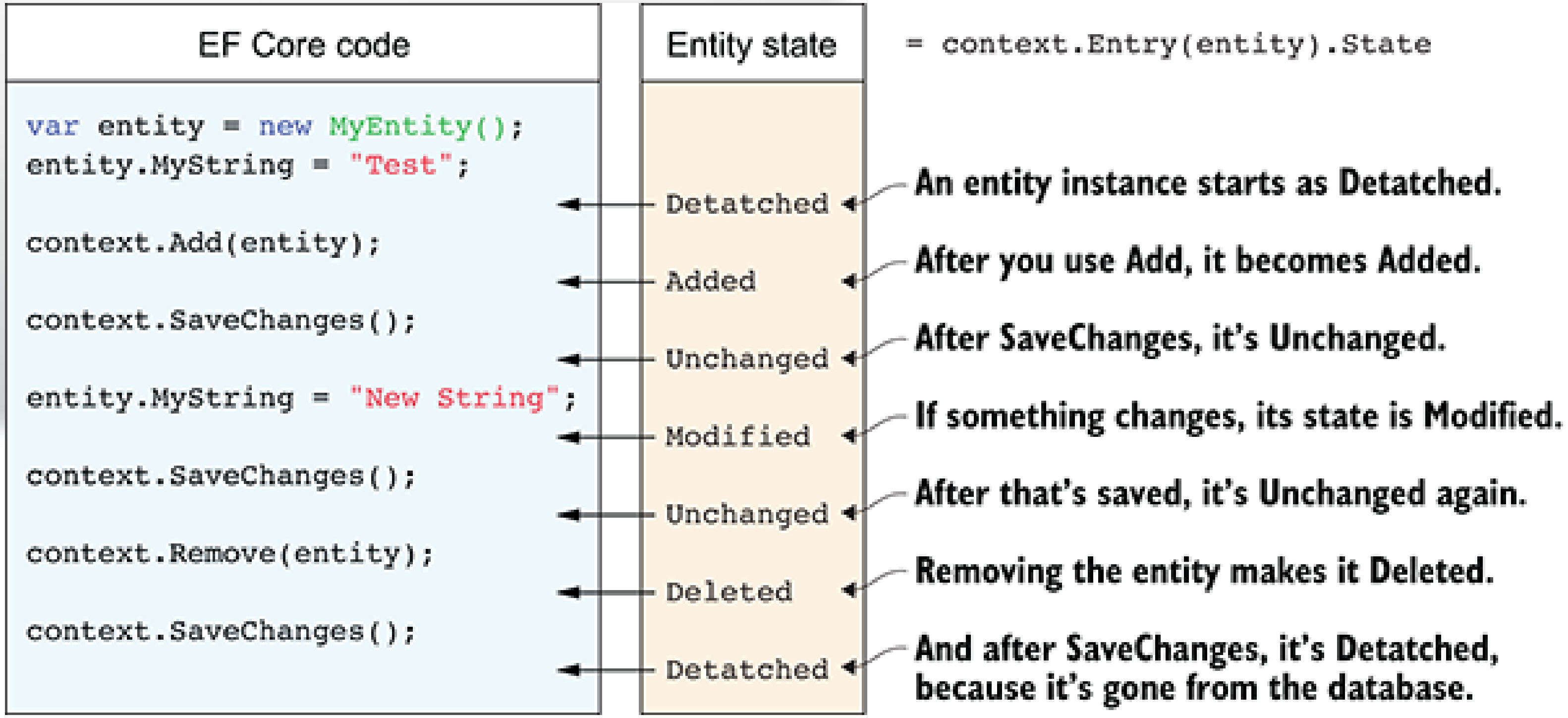
Sytem.Data - Reading Records

```
var results = new List<Employee>();
using (SqlConnection connection = new SqlConnection(Settings.ConnectionString))
{
    SqlCommand command = new SqlCommand(“SELECT * FROM Employees”, connection);
    connection.Open();
    IDataReader reader = command.ExecuteReader();
    while (reader.Read())
    {
        results.Add(ReadSingleRow(reader));
    }
    reader.Close();
}
return results;
```

ORMs

- Hide details of database
- Generate SQL based on object model and configuration
- Change Tracking
- Complex mapping strategies
  - Many-to-many relationships
  - Inheritance

ENTITY FRAMEWORK™



EF Core code

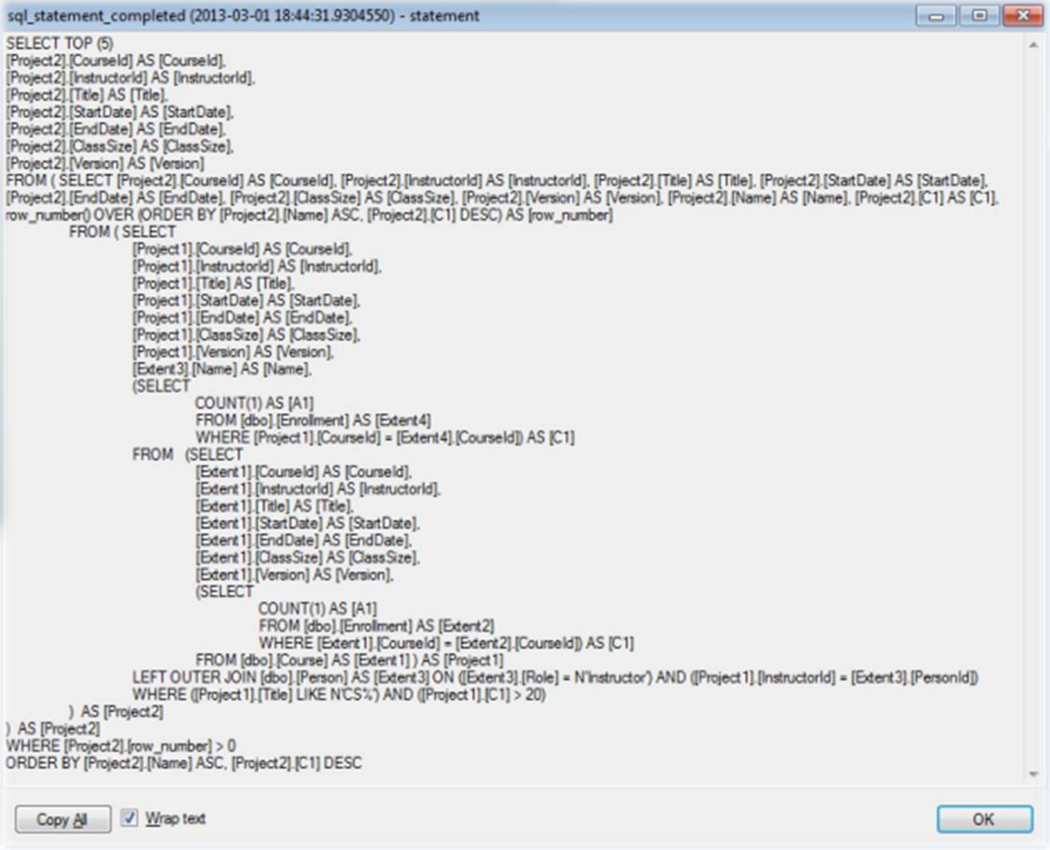
```
var entity = new MyEntity();
entity.MyString = "Test";

context.Add(entity);

context.SaveChanges();

entity.MyString = "New String";

context.SaveChanges();

context.Remove(entity);

context.SaveChanges();
```

Entity state = `context.Entry(entity).State`

Entity state	
Detatched	An entity instance starts as Detatched.
Added	After you use Add, it becomes Added.
Unchanged	After SaveChanges, it's Unchanged.
Modified	If something changes, its state is Modified.
Unchanged	After that's saved, it's Unchanged again.
Deleted	Removing the entity makes it Deleted.
Detatched	And after SaveChanges, it's Detatched, because it's gone from the database.

ORM Pain Points

- Black box code generation – What is going on?
- Performance Problems
  - Check out MiniProfiler (http://miniprofiler.com/)
- Dealing with disconnected entities (in a web context)
- Eager Loading vs Lazy Loading
- Complex Inheritance Chains


sql_statement_completed (2013-03-01 18:44:31.9304550) - statement
Copy All
☑ Wrap text
OK

```
SELECT TOP (5)
[Project2].[CourseId] AS [CourseId],
[Project2].[InstructorId] AS [InstructorId],
[Project2].[Title] AS [Title],
[Project2].[StartDate] AS [StartDate],
[Project2].[EndDate] AS [EndDate],
[Project2].[ClassSize] AS [ClassSize],
[Project2].[Version] AS [Version]
FROM ( SELECT [Project2].[CourseId] AS [CourseId], [Project2].[InstructorId] AS [InstructorId], [Project2].[Title] AS [Title], [Project2].[StartDate] AS [StartDate],
[Project2].[EndDate] AS [EndDate], [Project2].[ClassSize] AS [ClassSize], [Project2].[Version] AS [Version], [Project2].[Name] AS [Name], [Project2].[C1] AS [C1],
row_number() OVER (ORDER BY [Project2].[Name] ASC, [Project2].[C1] DESC) AS [row_number]
        FROM ( SELECT
                [Project1].[CourseId] AS [CourseId],
                [Project1].[InstructorId] AS [InstructorId],
                [Project1].[Title] AS [Title],
                [Project1].[StartDate] AS [StartDate],
                [Project1].[EndDate] AS [EndDate],
                [Project1].[ClassSize] AS [ClassSize],
                [Project1].[Version] AS [Version],
                [Extent3].[Name] AS [Name],
                (SELECT
                        COUNT(1) AS [A1]
                        FROM [dbo].[Enrollment] AS [Extent4]
                        WHERE [Project1].[CourseId] = [Extent4].[CourseId]) AS [C1]
                FROM   (SELECT
                        [Extent1].[CourseId] AS [CourseId],
                        [Extent1].[InstructorId] AS [InstructorId],
                        [Extent1].[Title] AS [Title],
                        [Extent1].[StartDate] AS [StartDate],
                        [Extent1].[EndDate] AS [EndDate],
                        [Extent1].[ClassSize] AS [ClassSize],
                        [Extent1].[Version] AS [Version],
                        (SELECT
                                COUNT(1) AS [A1]
                                FROM [dbo].[Enrollment] AS [Extent2]
                                WHERE [Extent1].[CourseId] = [Extent2].[CourseId]) AS [C1]
                        FROM [dbo].[Course] AS [Extent1] ) AS [Project1]
                LEFT OUTER JOIN [dbo].[Person] AS [Extent3] ON ([Extent3].[Role] = N'Instructor') AND ([Project1].[InstructorId] = [Extent3].[PersonId])
                WHERE ([Project1].[Title] LIKE N'CS%') AND ([Project1].[C1] > 20)
        )  AS [Project2]
)  AS [Project2]
WHERE [Project2].[row_number] > 0
ORDER BY [Project2].[Name] ASC, [Project2].[C1] DESC
```

What a MicroORM is?

Just do one simple thing, take data coming from a database query an use it to populate pre-existing or dynamic objects

- Nothing more and nothing less
- No frills approach: Not identity mapping, no lazy load
- SQL *MUST* be written manually (no LINQ or other intermediate language like HQL)
- Tries not to introduce friction when accessing and operating on data

One of the most used, proven and well-known is Dapper .NET

https://blogs.msdn.microsoft.com/dotnet/2016/11/09/net-core-data-access/

When to use a Micro-ORM

- Speed & Efficiency are extremely important
- You don’t mind (or prefer) writing your own SQL
- Simple object graphs
- Read models / Reports

Performance of SELECT mapping over 500 iterations - POCO serialization

Method	Duration
Hand coded (using a SqlDataReader)	47ms
Dapper ExecuteMapperQuery	49ms
ServiceStack.OrmLite (QueryById)	50ms
PetaPoco	52ms
BLToolkit	80ms
SubSonic CodingHorror	107ms
NHibernate SQL	104ms
Linq 2 SQL ExecuteQuery	181ms
Entity framework ExecuteStoreQuery	631ms

When to use an ORM

- You like the convenience of change tracking
- Your application logic requires a complex object graph
- You prefer writing queries using LINQ

Writing your own Data Access Layer / ORM

Optimizing your ORM experience

- No Tracking
- SQL Projections
- Avoid Lazy Loading
- Avoid the God Context
- Write SQL!

No Tracking + No Lazy Loading + Projections

```
return await _context.Tasks.AsNoTracking()
               .Include(t => t.Event).ThenInclude(a => a.Campaign)
               .Include(t => t.RequiredSkills).ThenInclude(ts => ts.Skill)
               .Select(task => new EditViewModel
               {
                   Id = task.Id,
                   Name = task.Name,
                   Description = task.Description
             //etc…
            }).SingleAsync(t => t.Id == taskId);
```

Surprisingly Efficient

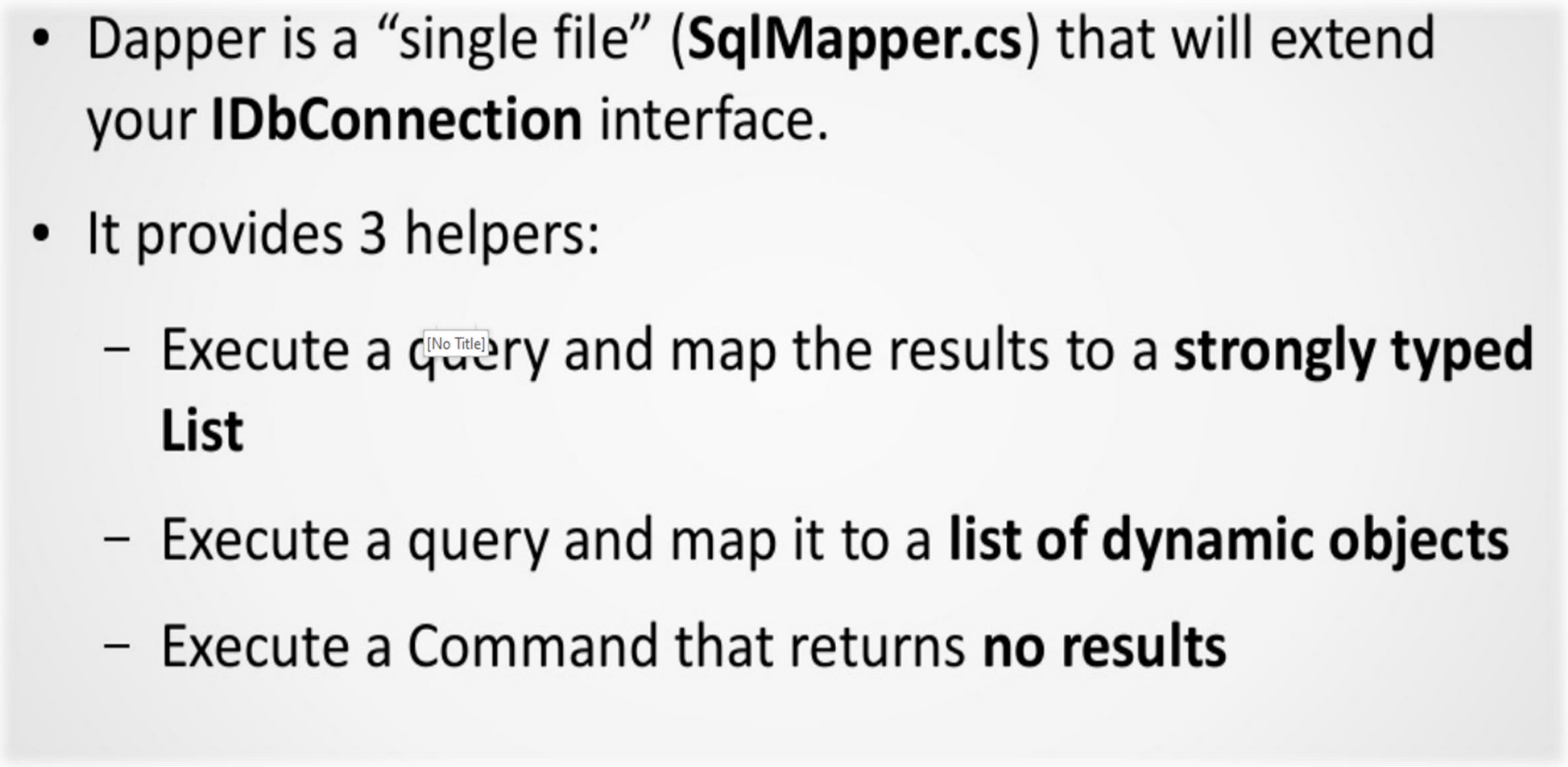
```sql
SELECT TOP(2) [task].[Id], [task].[Name], [task].[Description]
          --, More columns
FROM [AllReadyTask] AS [task]
INNER JOIN [Event] AS [task.Event] ON [task].[EventId] = [task.Event].[Id]
INNER JOIN [Campaign] AS [task.Event.Campaign] ON [task.Event].[CampaignId] =
[task.Event.Campaign].[Id]
WHERE [task].[Id] = @__message_TaskId_0
```

Execution Time: ~35ms

Dapper

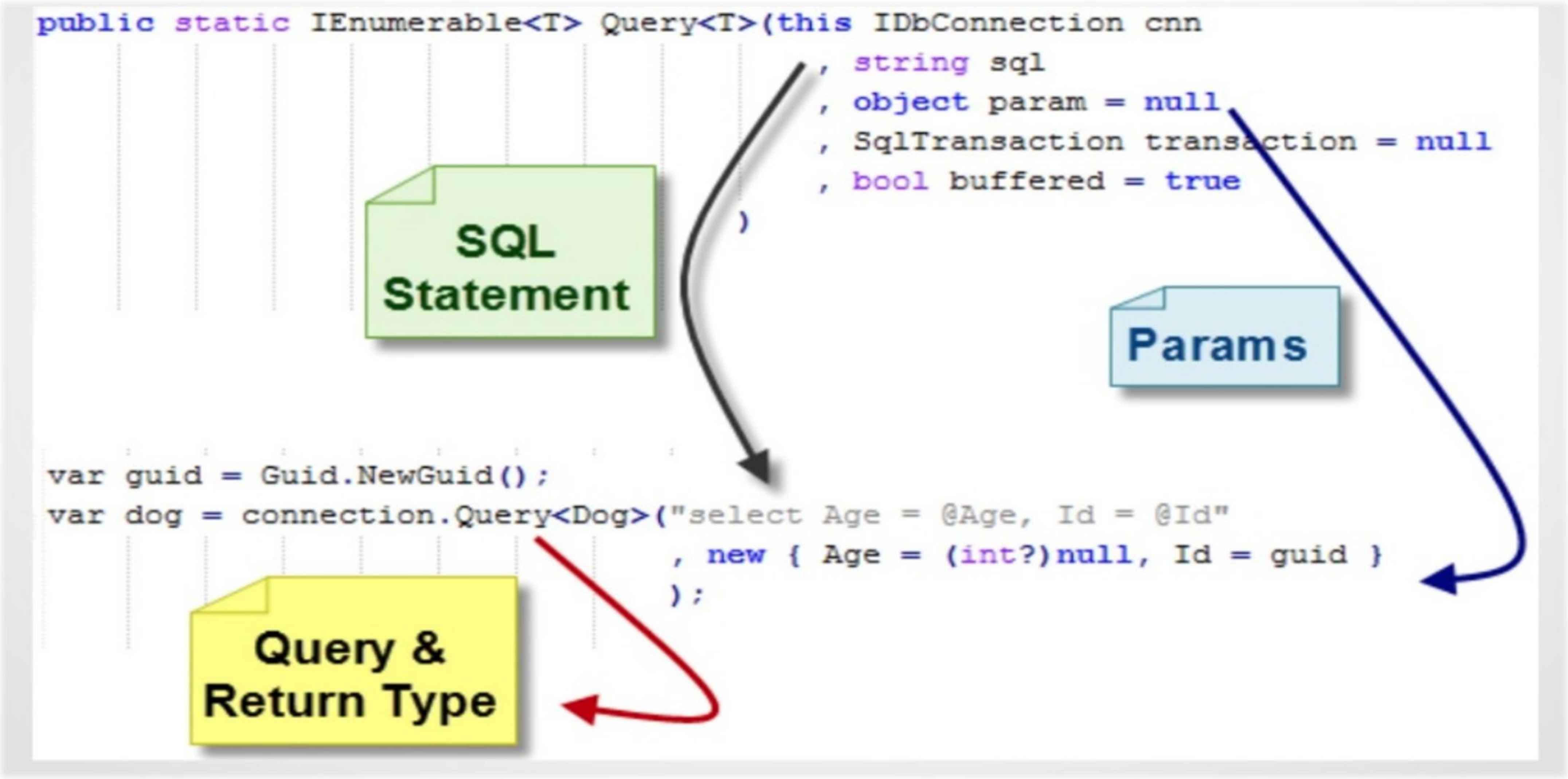
(A micro ORM)

Dapper In Action



- Dapper is a “single file” (**SqlMapper.cs**) that will extend your **IDbConnection** interface.
- It provides 3 helpers:
    - Execute a query and map the results to a **strongly typed List**
    - Execute a query and map it to a **list of dynamic objects**
    - Execute a Command that returns **no results**

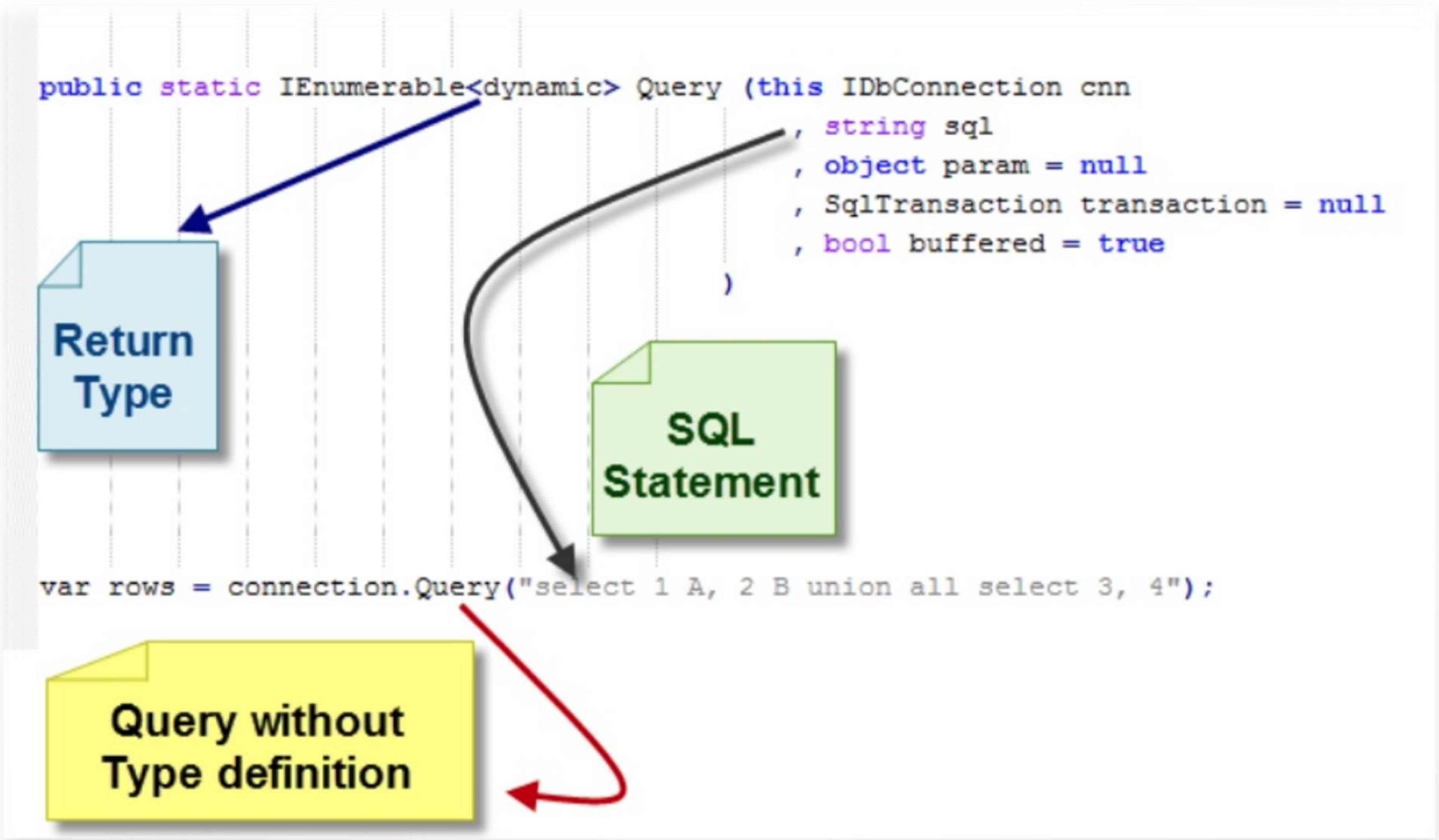
1- Query with Strongly Typed Result



```
public static IEnumerable<T> Query<T>(this IDbConnection cnn
                                      , string sql
                                      , object param = null
                                      , SqlTransaction transaction = null
                                      , bool buffered = true
                                  )
```

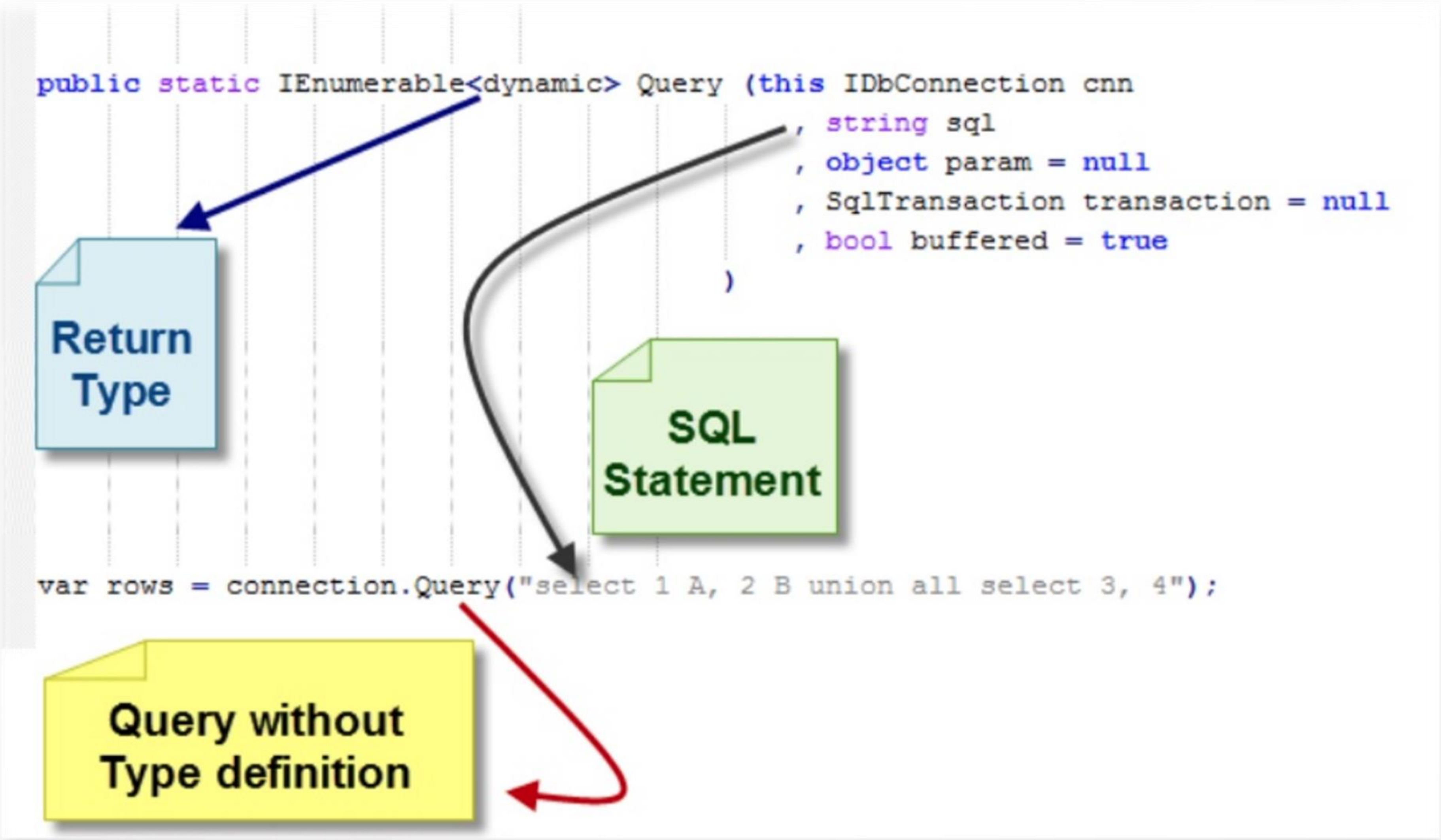
```
var guid = Guid.NewGuid();
var dog = connection.Query<Dog>("select Age = @Age, Id = @Id"
                               , new { Age = (int?)null, Id = guid }
                               );
```

2- Query with Dynamic Object Result



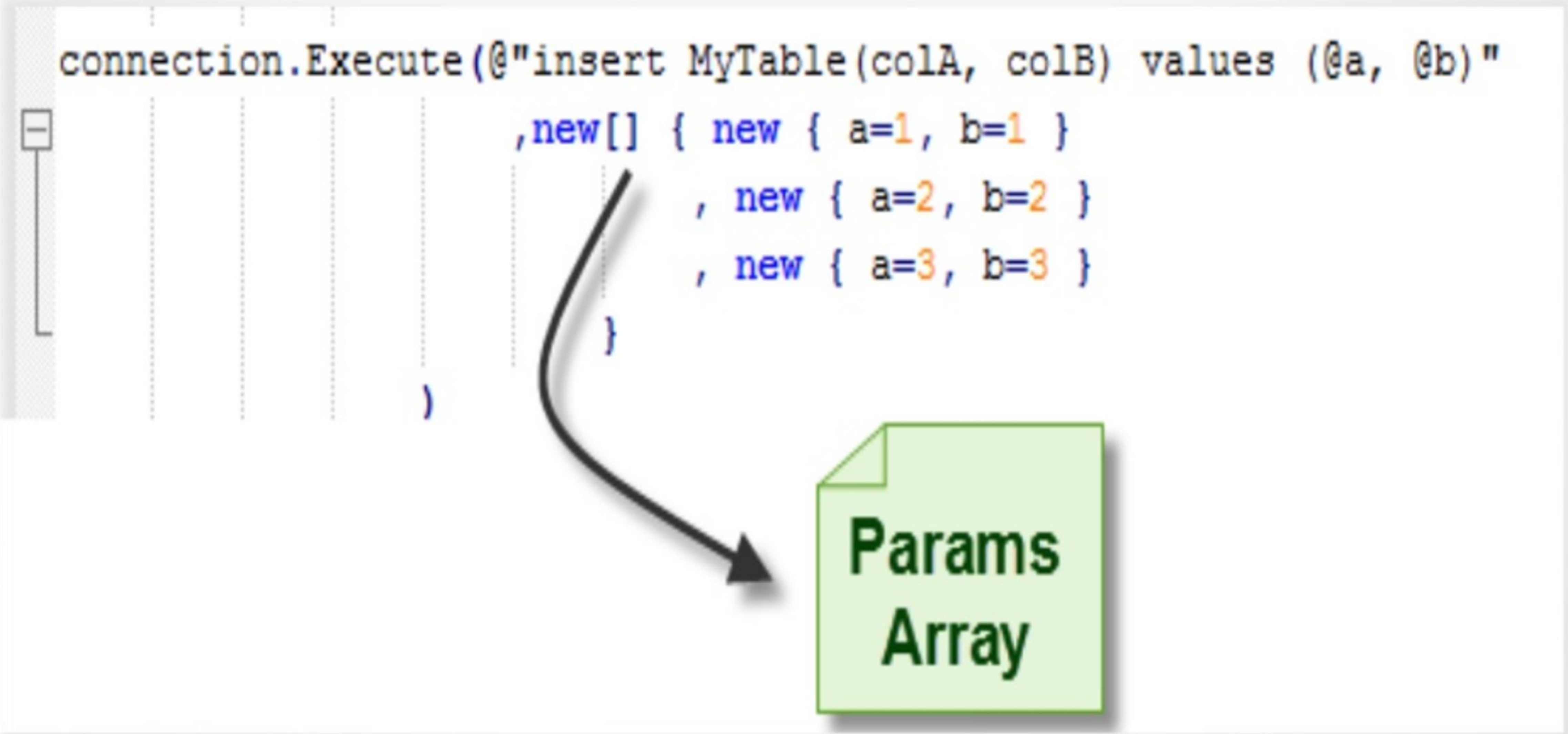
```
public static IEnumerable<dynamic> Query (this IDbConnection cnn
                                         , string sql
                                         , object param = null
                                         , SqlTransaction transaction = null
                                         , bool buffered = true
                                   )

var rows = connection.Query("select 1 A, 2 B union all select 3, 4");
```

3- Command with No Result



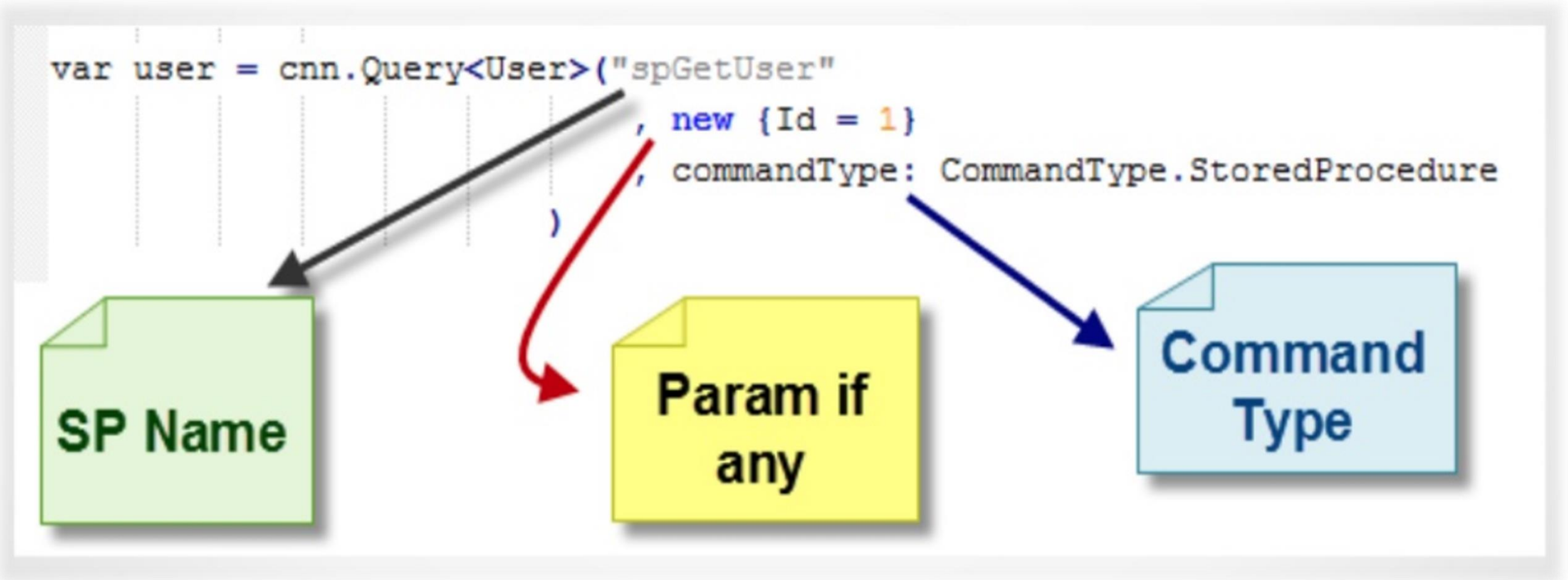
```
public static IEnumerable<dynamic> Query (this IDbConnection cnn
                                            , string sql
                                            , object param = null
                                            , SqlTransaction transaction = null
                                            , bool buffered = true
                                        )

var rows = connection.Query("select 1 A, 2 B union all select 3, 4");
```

4- Execute Command multiple times



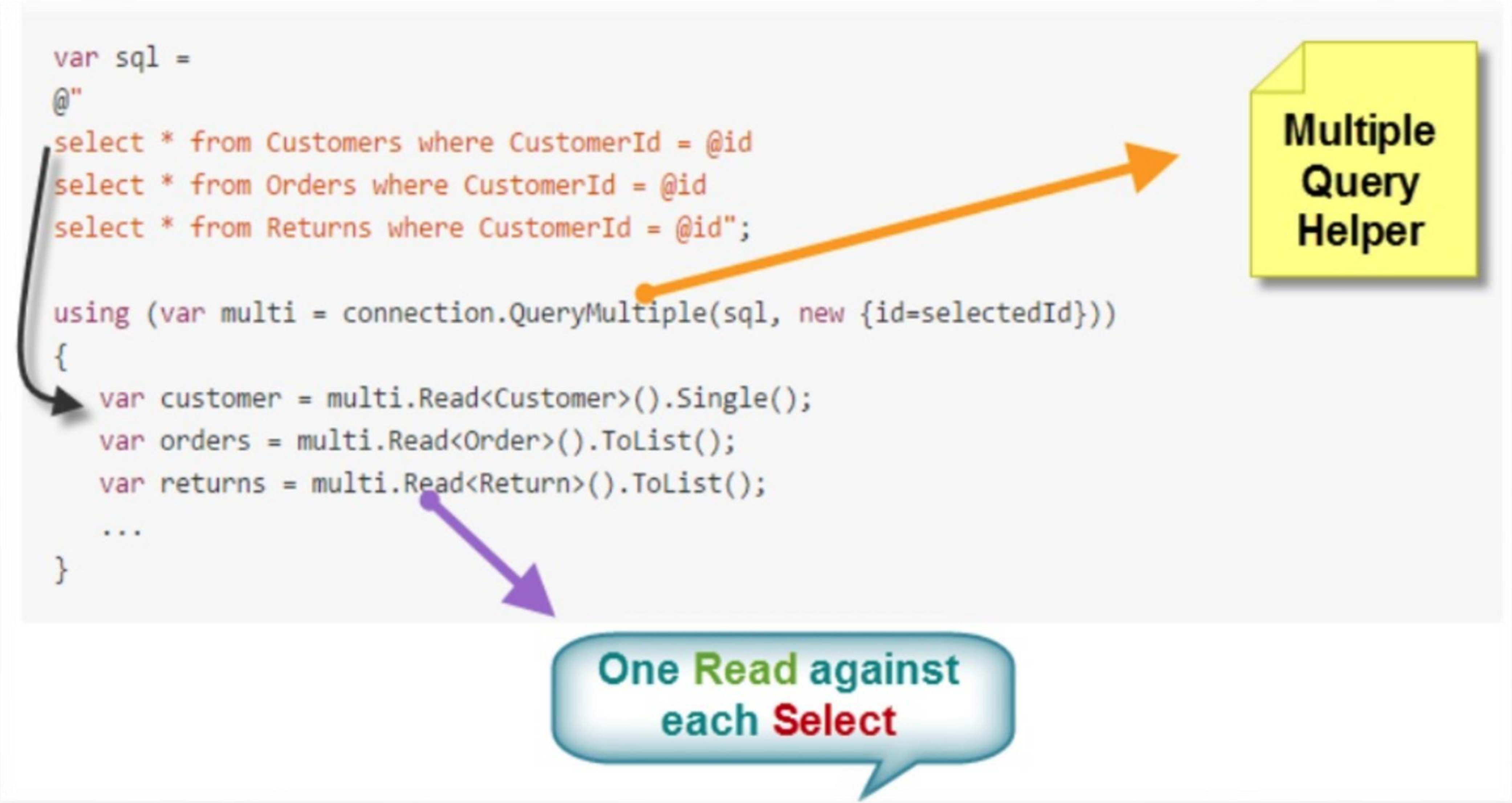
```
connection.Execute(@"insert MyTable(colA, colB) values (@a, @b)"
                    ,new[] { new { a=1, b=1 }
                            , new { a=2, b=2 }
                            , new { a=3, b=3 }
                        }
                )
```

5- Execute a Stored Procedure



```
var user = cnn.Query<User>("spGetUser"
                           , new {Id = 1}
                           , commandType: CommandType.StoredProcedure
                      )
```

6- Multiple Results in Single Query

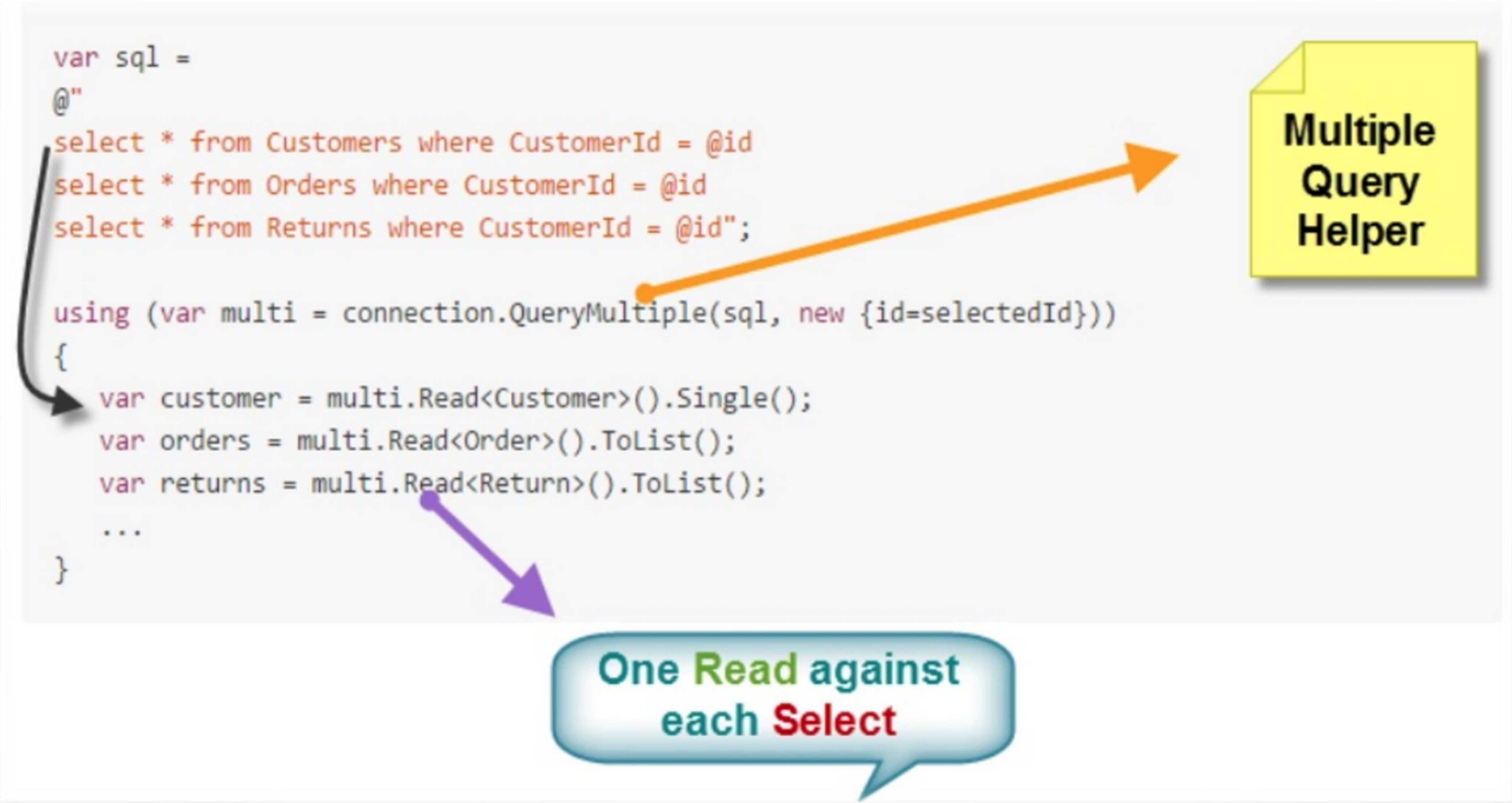


```
var sql =
@"
select * from Customers where CustomerId = @id
select * from Orders where CustomerId = @id
select * from Returns where CustomerId = @id";

using (var multi = connection.QueryMultiple(sql, new {id=selectedId}))
{
   var customer = multi.Read<Customer>().Single();
   var orders = multi.Read<Order>().ToList();
   var returns = multi.Read<Return>().ToList();
   ...
}
```

Multiple Query Helper

One Read against each Select

6- Multiple Results in Single Query



```
var sql =
@"
select * from Customers where CustomerId = @id
select * from Orders where CustomerId = @id
select * from Returns where CustomerId = @id";

using (var multi = connection.QueryMultiple(sql, new {id=selectedId}))
{
   var customer = multi.Read<Customer>().Single();
   var orders = multi.Read<Order>().ToList();
   var returns = multi.Read<Return>().ToList();
   ...
}
```

LIVE DEMO

Any questions?