Working with Lazy Loading

Vladimir Khorikov

@vkhorikov www.enterprisecraftsmanship.com

Eager Loading of Relationships

Partially initialized entities anti-pattern

Eager Loading of Relationships

Partially initialized entities anti-pattern

Load all the entity's relationships

Eager Loading of Relationships

Can you only load relationships required for the particular business scenario?

```
public class Student
{
    public long Id { }
    public string Name { }
    public string Email { }
    public Course FavoriteCourse { }
    public ICollection<Enrollment> Enrollments { }
}
```

StartProject()

AddEnrollment()

Additional complexity

Lazy Loading

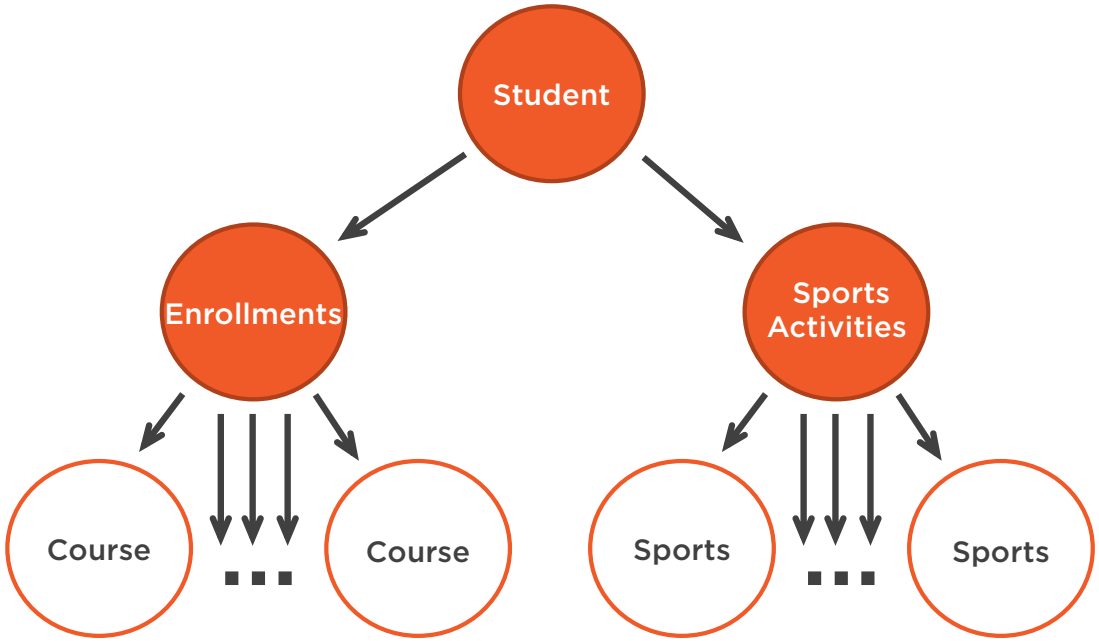
Lazy loading defers initialization of an object until the point at which this object is needed.

Lazy Loading of Relationships

Prefer lazy loading over eager loading by default

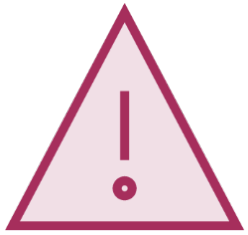
- Helps avoid partially initialized entities
- More performant in some scenarios
- Code simplicity

N+1 Problem



Bad for performance

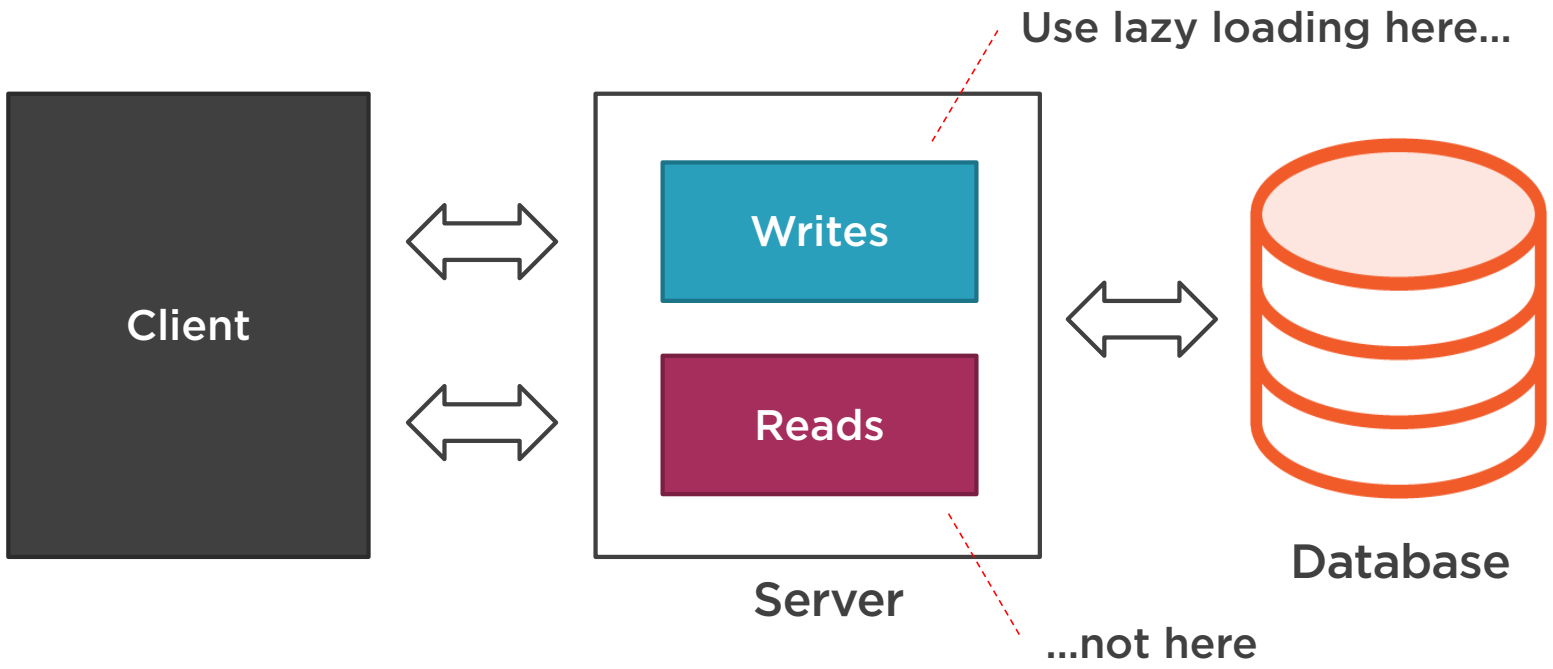
N+1 Problem



N+1 problem only takes place in reads

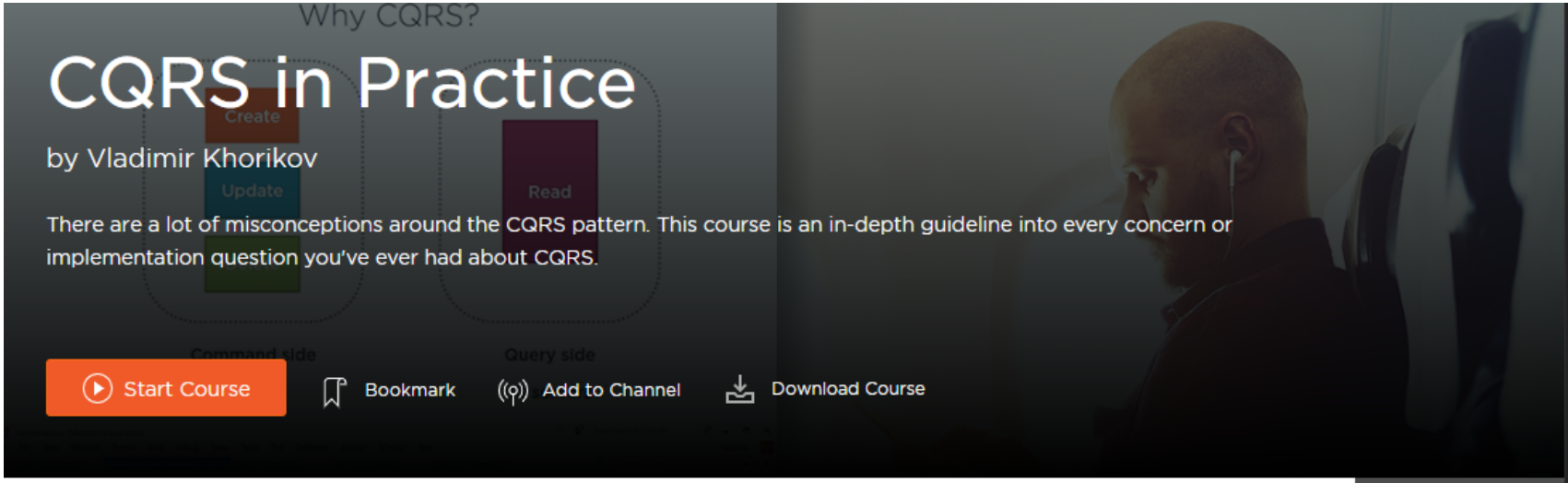
Lazy loading is beneficial in writes

Lazy Loading of Relationships



No data modifications =

- No need for encapsulation
- No need for fully-fledged ORMs



CQRS in Practice

by Vladimir Khorikov

There are a lot of misconceptions around the CQRS pattern. This course is an in-depth guideline into every concern or implementation question you've ever had about CQRS.

Start Course | Bookmark | Add to Channel | Download Course

Table of contents | Description | Transcript | Exercise files | Discussion | Learning Check | Recommended

This course is part of: Domain-Driven Design Path

Expand All

Course Overview	✓	1m 34s
Introduction	✓	15m 39s
Introducing a Sample Project		27m 41s
Refactoring Towards a Task-based Interface		32m 21s
Segregating Commands and Queries		52m 44s

Course author

Vladimir Khorikov

Vladimir Khorikov is a Microsoft MVP and has been professionally involved in software development for more than 10 years. Nowadays he specializes in rescuing legacy code bases and helping teams...

Course info

Level	Intermediate
Rating	★★★★★ (177)
My rating	★★★★★
Duration	4h 22m
Released	11 Oct 2018

Share course

Recap: Refactoring to Lazy Loading

- **Transitioned to lazy loading**
- **Prefer lazy loading over eager loading**
  - **Helps reduce code complexity**
  - **Only use eager loading when performance benefits are significant**

Recap: Refactoring to Lazy Loading

Modified the project

```
<PackageReference Include="Microsoft.EntityFrameworkCore.Proxies" />
```

```
optionsBuilder.UseLazyLoadingProxies();
```

Modified the domain model

Non-sealed classes

Protected parameter-less constructors

Virtual navigation properties

Recap: Refactoring to Lazy Loading

Beware of the N+1 problem

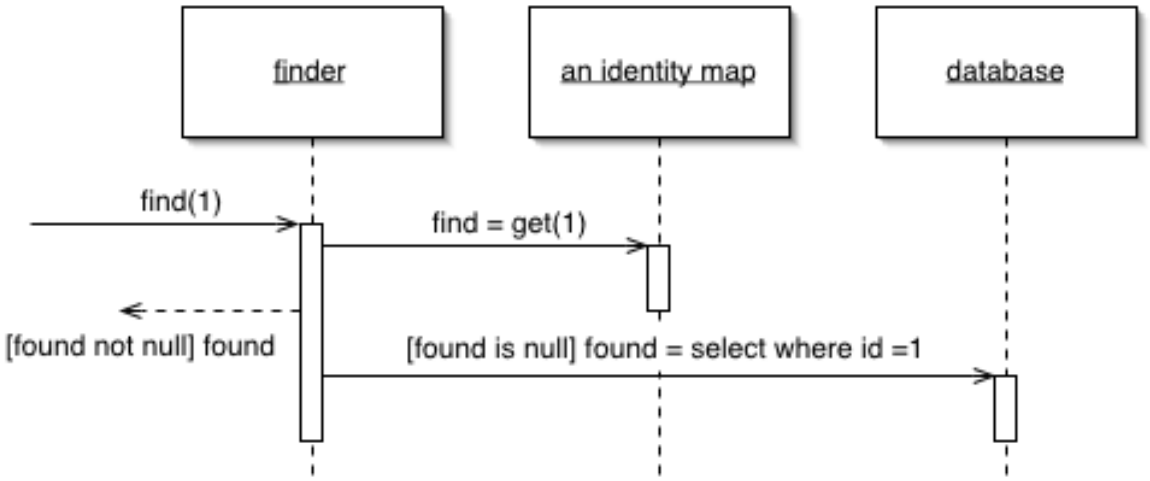
- Apply the CQRS pattern
- Don’t use EF Core or a domain model in reads
- Check out CQRS in Practice

Recap: Refactoring to Lazy Loading

```
public class Student
{
    private ILazyLoader _lazyLoader;

    public long Id { get; private set; }
    public string Name { get; }
    public string Email { get; }

    private Course _favoriteCourse;
    public Course FavoriteCourse
    {
        get => _lazyLoader.Load(this, ref _favoriteCourse);
    }

    private Student(ILazyLoader lazyLoader)
    {
        _lazyLoader = lazyLoader;
    }
}
```

Violation of the SoC principle

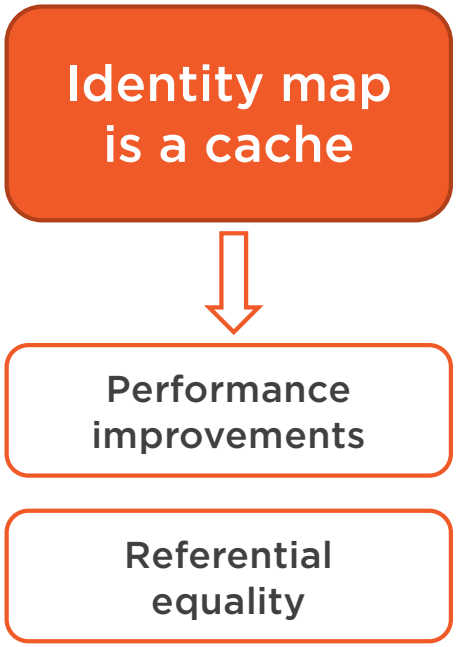
The Identity Map Pattern



Source:
https://www.martinfowler.com/eaaCatalog/identityMap.html

- Performance
- Referential equality

The Identity Map Pattern: Referential Equality



Prefer Find() over Single() or First()

Encapsulating Equality Comparison

Equality comparison

Checking if two objects are equal

~~`object.ReferenceEquals(student1, student2);`~~

`student1 == student2`

~~`student1.Equals(student2)`~~

~~`student1.Id == student2.Id`~~

Violation of SoC

Recap: Introducing a Base Entity Class

- Equality comparison should be encapsulated from the entity's clients
- Entity base class contains equality comparison logic reused by all entities
- Entity base class can work with EF's runtime proxies

Had to introduce another violation of SoC

Summary

Partially initialized entities anti-pattern leads to:

- Code complexity
- Invariant violations

Use eager or lazy loading to avoid the anti-pattern

- Prefer lazy loading by default
- Use eager loading for performance benefits

N+1 problem

- Only valid for reads
- Adhere to the CQRS pattern
- Don't use EF Core in reads

Don't use ILazyLoader

DbContext implements the Identity Map pattern

- Use Find(), not Single() or First()
- Referential equality

Encapsulate equality comparison

In the Next Module

Mapping Backing Fields