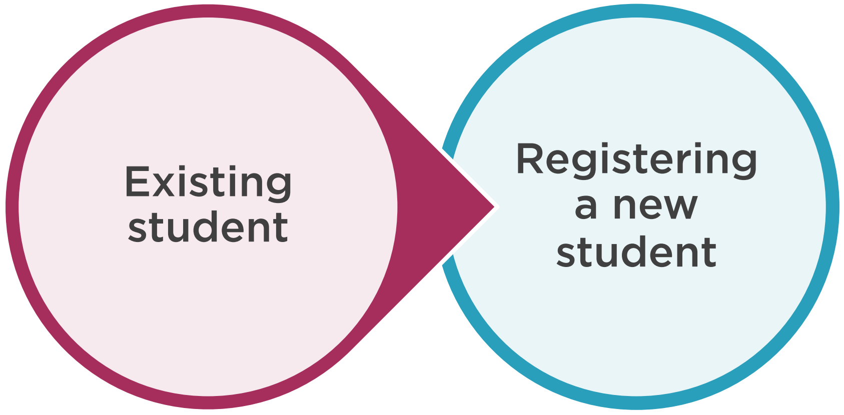
Working with Disconnected Graphs of Objects

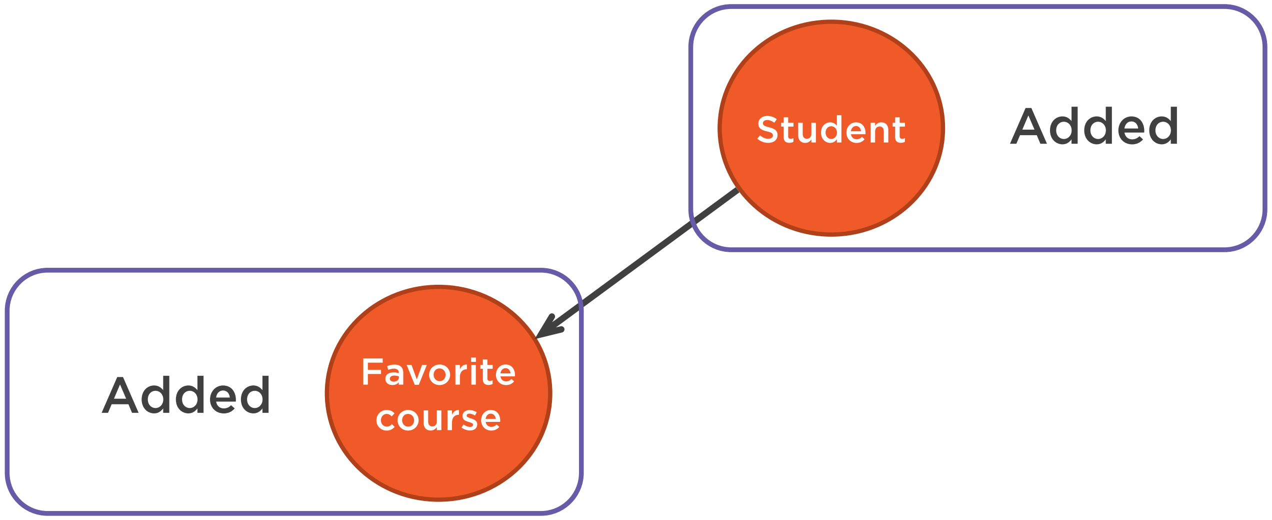
Vladimir Khorikov

@vkhorikov www.enterprisecraftsmanship.com

New Use Case: Registering a Student



Recap: Registering a Student

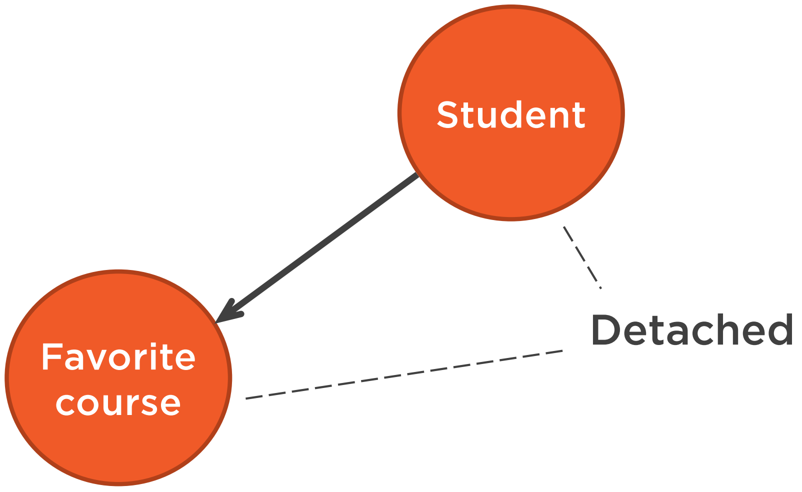


❌ DbSet.Add() marks all entities in the aggregate as Added

Recap: Registering a Student

Detached	DbContext doesn’t track the entity
Unchanged	Entity is not changed
Deleted	Entity will be deleted
Modified	Entity will be updated
Added	Entity will be inserted

Recap: Registering a Student



❌ DbSet.Add() marks all *detached* entities in the aggregate as Added

Recap: Registering a Student

How to avoid the additional DB query?

```
_context.Entry(student.FavoriteCourse).State = EntityState.Unchanged;
```

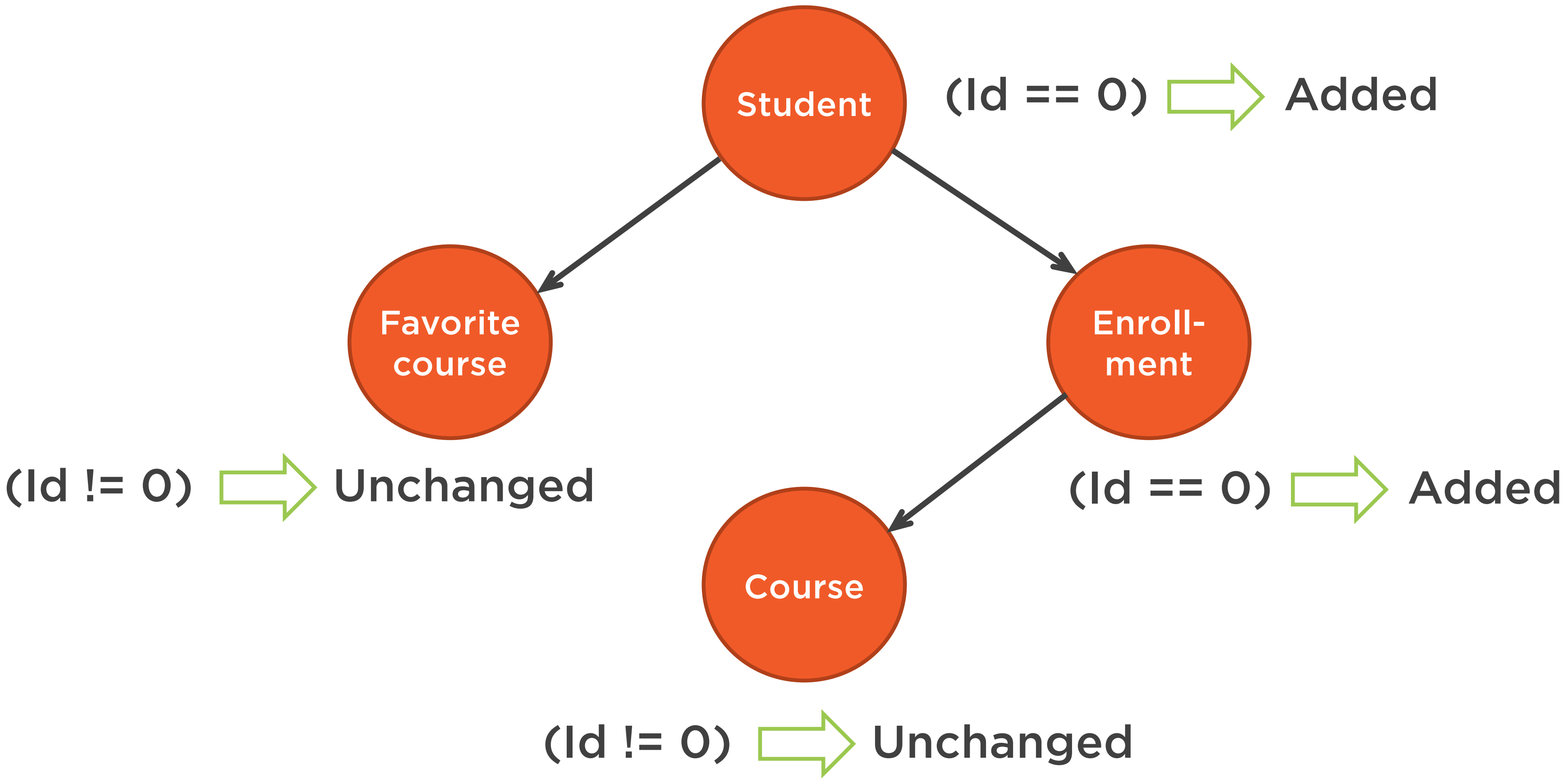
Update and Attach Methods in DbSet

```
~~_context.Entry(student.FavoriteCourse).State = EntityState.Unchanged;~~
```

✓

```
_context.Students.Attach(student);
```

Update and Attach Methods in DbSet



Update and Attach Methods in DbSet

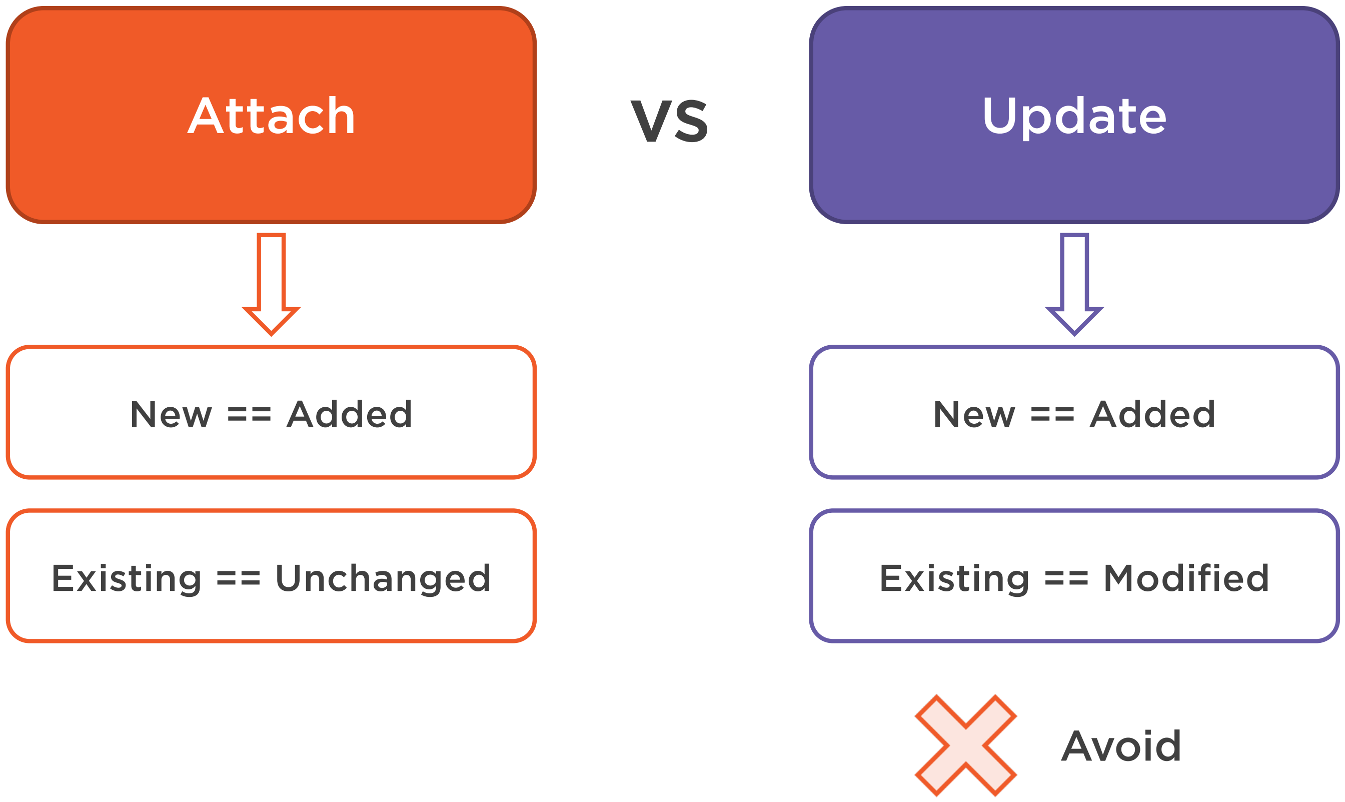
```
var student = new Student(
    name, email, favoriteCourse);
```

No Id yet

```
_context.Students.Add(student);
_context.SaveChanges();
```

EF Core sets the Id

Update and Attach Methods in DbSet



Update and Attach Methods in DbSet

Always prefer Attach over Update or Add

Never modify domain objects outside of a DbContext

Update and Attach Methods in DbSet

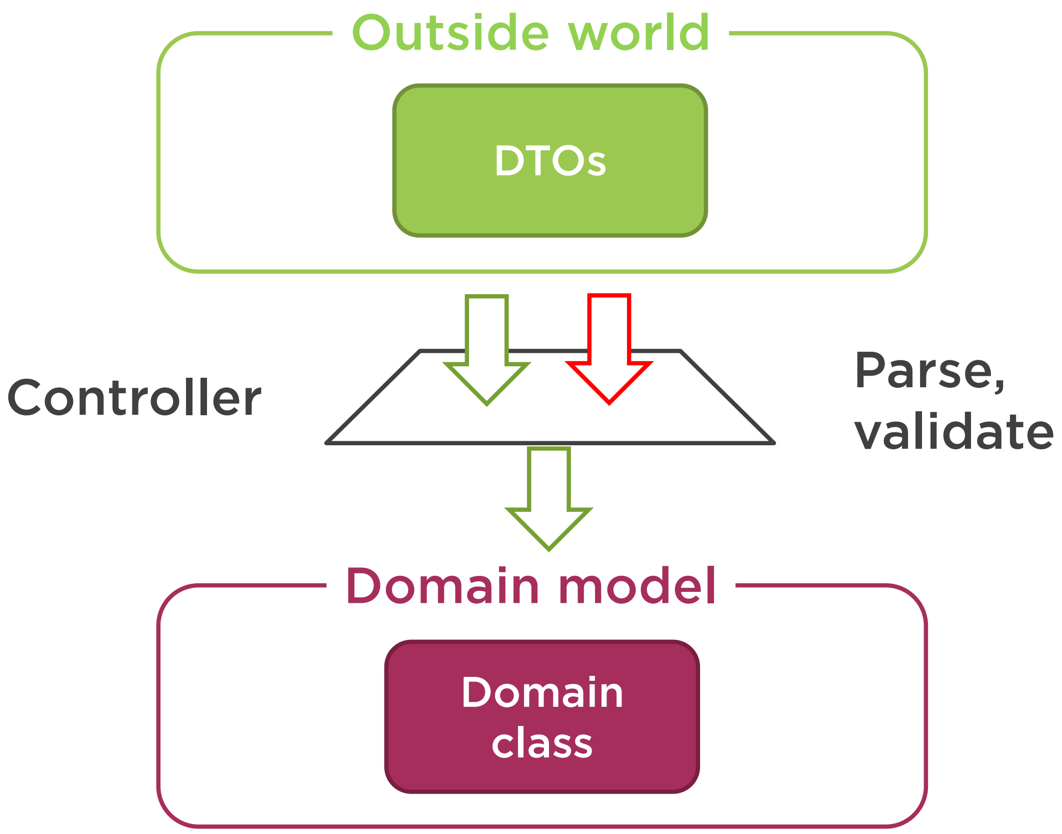
```
// Controller
public string RegisterStudent(Student student)
{
    _context.Students.Update(student);
    _context.SaveChanges();

    return "OK";
}
```

- ✗ **Student may contain sensitive properties**
- ✗ **Hinders backward compatibility**

Refactoring from Anemic Domain Model Towards a Rich One

Update and Attach Methods in DbSet



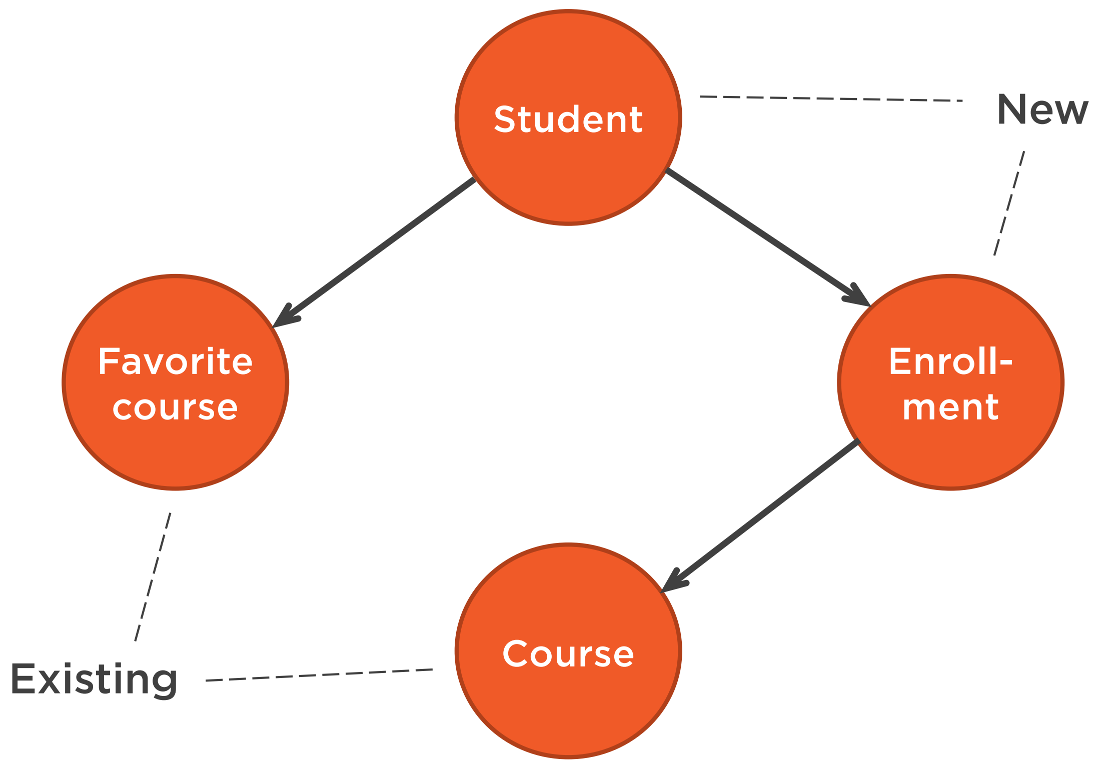
✓ Cache of domain objects

✓ Desktop applications

Recap: Add vs. Update vs. Attach Methods in DbSet

✓ **Student registration**

Recap: Add vs. Update vs. Attach Methods in DbSet



✗ Add(student)

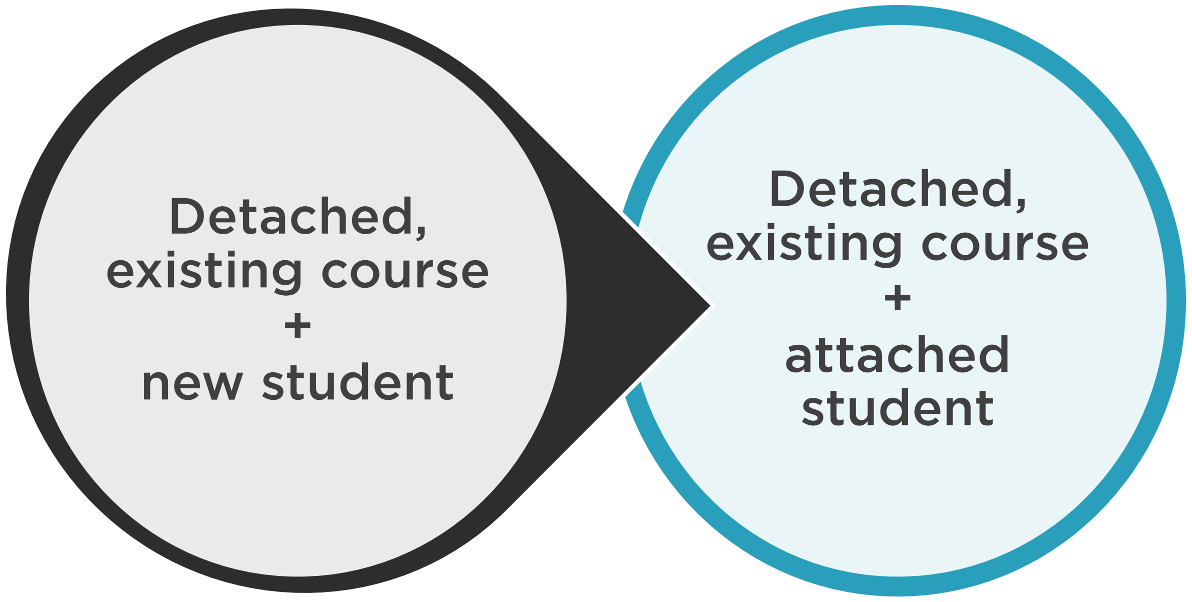
✓ Attach(student)

✗ Update(student)

⚠ Anti-pattern

✓ Used repository to attach student

Assigning a Disconnected Entity to a Connected One


Detached, existing course + new student
Detached, existing course + attached student

Recap: Assigning a Disconnected Entity to a Connected One

Detached, existing course + **Attached student**

✗ **EF Core marks course as Modified**

✓
```
.Metadata.SetAfterSaveBehavior(
    PropertySaveBehavior.Ignore);
```

✓ **Override SaveChanges()**

Summary

Working with disconnected (detached) objects

New use case: student registration

- Detached, existing course + detached, new student
- Add() marks the whole aggregate as Added
- Used Attach()
- The use of Update() is an anti-pattern (except for desktop applications)

New use case: editing student personal information

- Detached, existing course + attached student
- EF Core marks the course as Modified
- Used Metadata.SetAfterSaveBehavior()
- Overrode SaveChanges()

In the Next Module

Mapping Value Objects