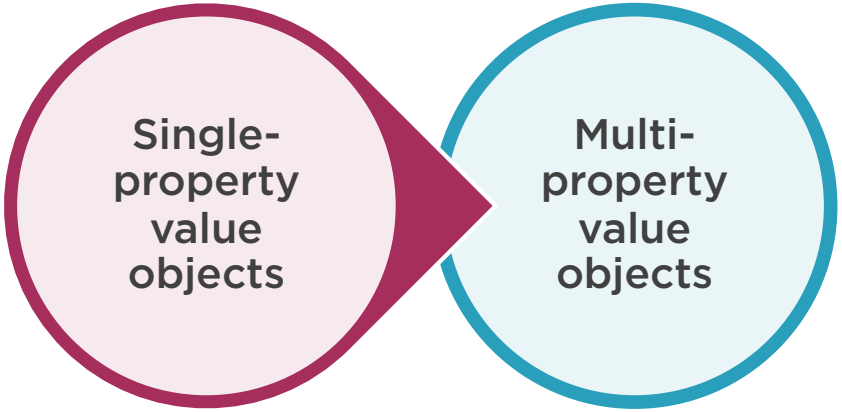
Mapping Value Objects

Vladimir Khorikov

@vkhorikov www.enterprisecraftsmanship.com

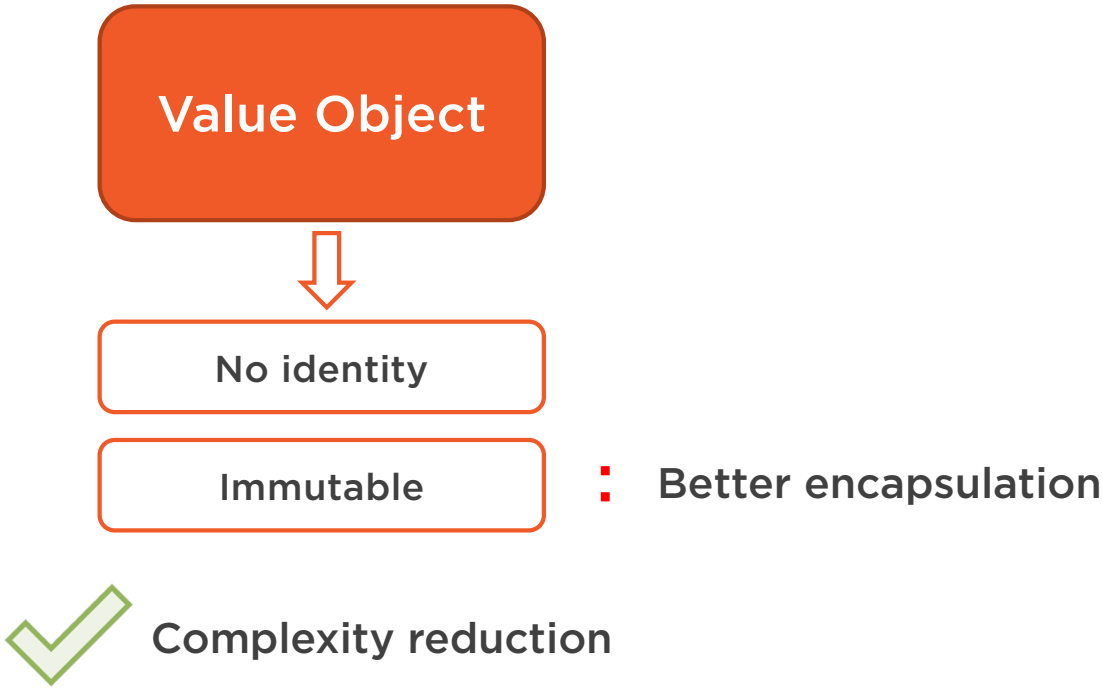
Mapping Value Objects



Value Objects

Domain-Driven Design in Practice

Applying Functional Principles in C#



Value Objects

```
public class Student : Entity
{
    public string Name { get; set; }

            Email
    public ~~string~~ Email { get; set; }
}
```

Shortcomings of EF Core Value Conversions

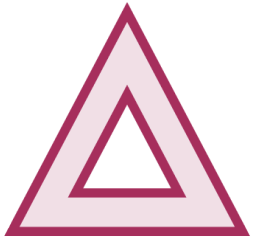
Created an Email value object

Single-property value object

Used EF Core Value Conversions

```
x.Property(p => p.Email)
    .HasConversion(
        p => p.Value,
        p => Email.Create(p).Value);
```

Shortcomings of EF Core Value Conversions



EF Core doesn't pass nulls to the factory method

```
x.Property(p => p.Email)
    .HasConversion(
        p => p.Value,
        p => Email.Create(p).Value
    );
```

Method not called for nulls

Shortcomings of EF Core Value Conversions

Allow HasConversion/ValueConverters to convert nulls #13850

Open MarkGodwin opened this issue on Nov 1, 2018 · 9 comments

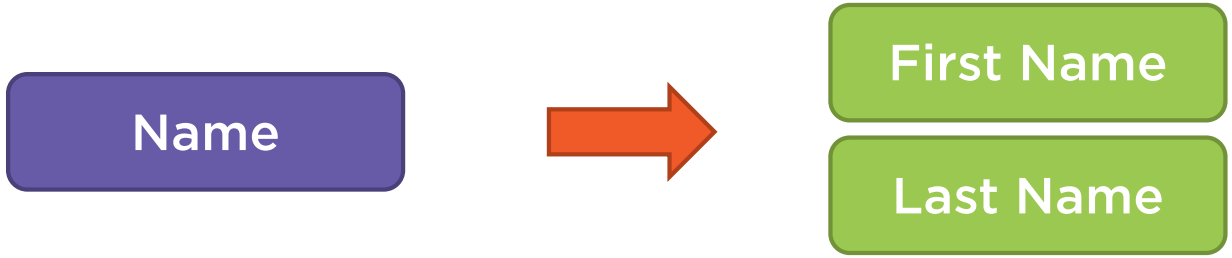
https://github.com/dotnet/efcore/issues/13850

Implement value conversions that spread out over multiple columns #13947

Open eveneveneven opened this issue on Nov 13, 2018 · 6 comments

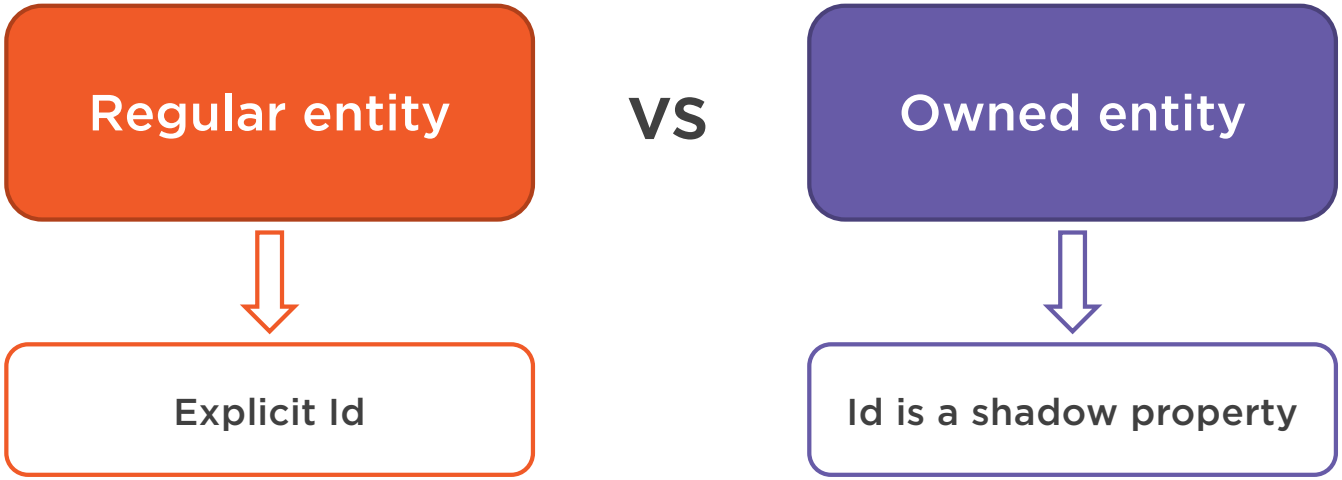
https://github.com/dotnet/efcore/issues/13947

Introducing a Multi-property Value Object

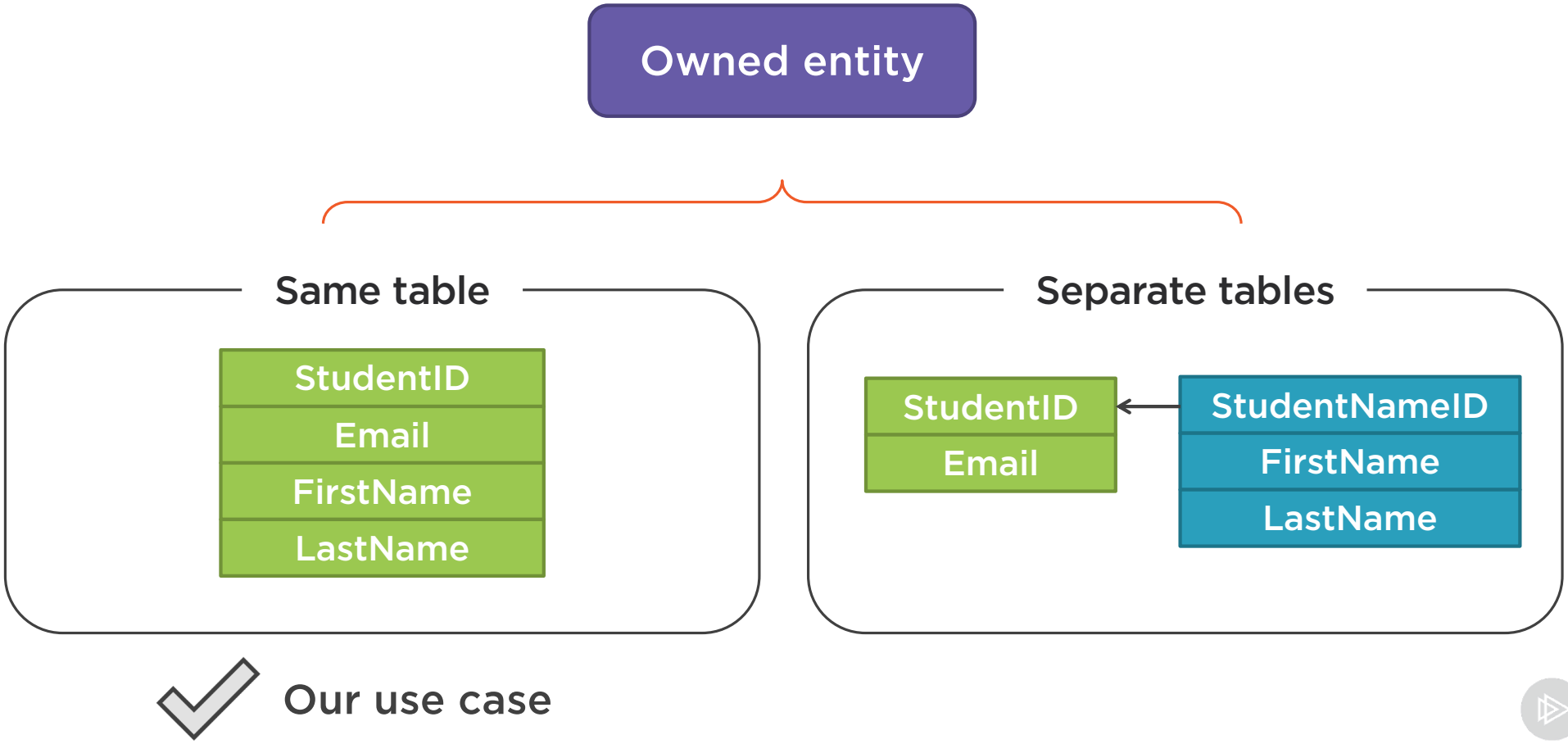


Multi-property value object

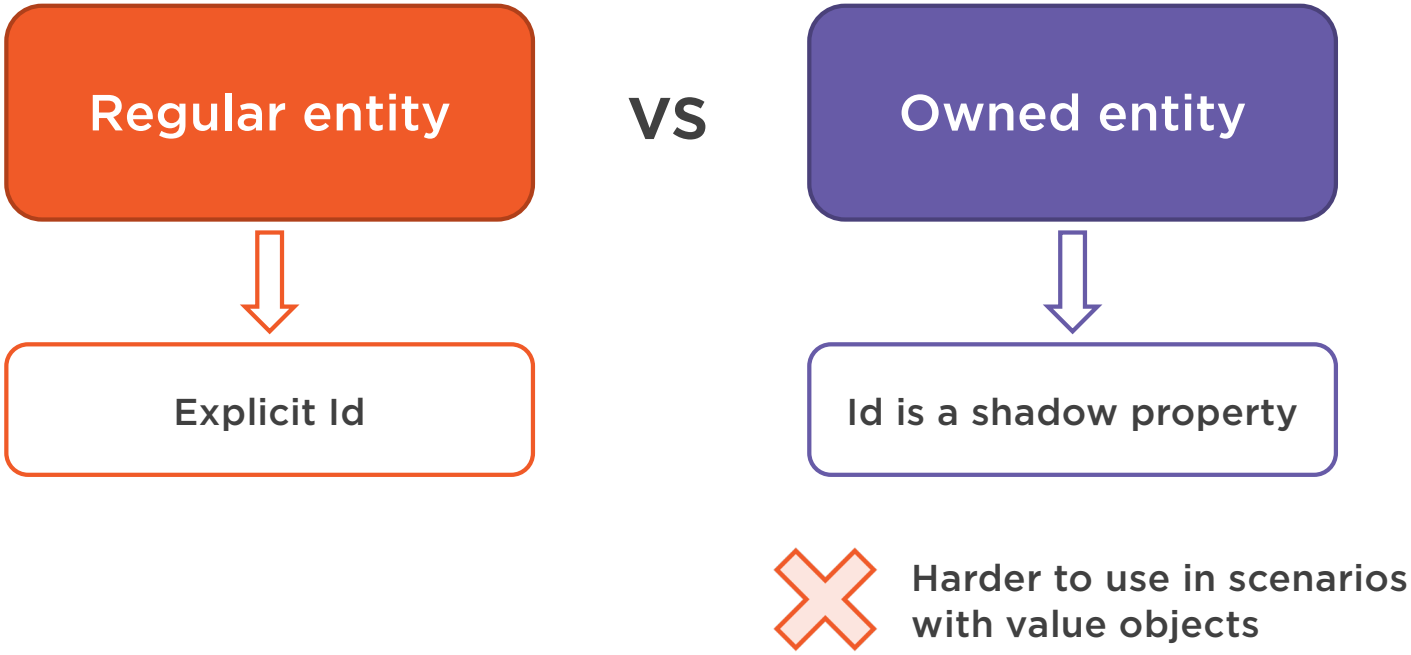
Owned Entity Types Behind the Scenes



Owned Entity Types Behind the Scenes



Owned Entity Types Behind the Scenes



Owned Entity Types Behind the Scenes

Shortcomings of Owned Entities

```
public class Student : Entity
{
    public virtual Name Name { }
    public Email Email { }
    public virtual Course FavoriteCourse { }
}
```

Value Objects

Entity

EF Core treats Name as an entity

Shortcomings of Owned Entities

```sql
SELECT TOP(1)
    [s].[StudentID], [s].[Email], [s].[FavoriteCourseId],
    [t].[StudentID], [t].[FirstName], [t].[LastName]
FROM [Student] AS [s]
LEFT JOIN (
    SELECT [s0].[StudentID], [s0].[FirstName], [s0].[LastName]
    FROM [Student] AS [s0]
    INNER JOIN [Student] AS [s1] ON [s0].[StudentID] = [s1].[StudentID]
    WHERE [s0].[LastName] IS NOT NULL OR [s0].[FirstName] IS NOT NULL
) AS [t] ON [s].[StudentID] = [t].[StudentID]
WHERE [s].[StudentID] = @StudentID
```

EF Core treats Name as if it resides in a separate table

```sql
SELECT
    [s].[StudentID], [s].[Email], [s].[FavoriteCourseId],
    [s].[FirstName], [s].[LastName]
FROM [Student] AS [s]
WHERE [s].[StudentID] = @StudentID
```

Domain-Driven Design in Practice

by Vladimir Khorikov

A descriptive, in-depth walk-through for applying Domain-Driven Design principles in practice.

Resume Course | Bookmark | Add to Channel | Download Course

Table of contents | Description | Exercise files | Discussion | Learning Check | Related Courses

This course is part of: Domain-Driven Design Path

Expand All

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	★★★★½ (483)
My rating	★★★★★
Duration	4h 19m
Updated	16 Sep 2019

Share course

Recap: Owned Entity Types Behind the Scenes

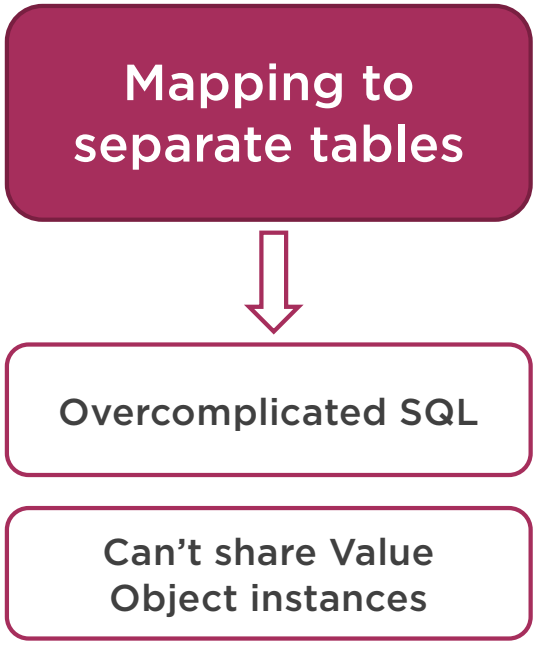
Regular entities with a hidden Id

Mapping to separate tables

Harder to use as value objects

Value Objects shouldn't be mapped to separate tables

Recap: Owned Entity Types Behind the Scenes



Recap: Owned Entity Types Behind the Scenes

Had to use the Null Object pattern instead of assigning a null

```
                    new Name(null, null);
student.Name = null;
```

Breach in encapsulation

Fixed in EF Core 3.1

Alternative Owned Entities Implementation

Drop the support for the separate tables use case

```
public class SlotMap : ClassMap<Slot> {
    public SlotMap() {
        Id(x => x.Id);

        Component(x => x.SnackPile, y => {
            y.Map(x => x.Quantity);
            y.Map(x => x.Price);
            y.References(x => x.Snack);
        });
    }
}
```

No Id

All changes are attributed to the entity

Perfect fit for Value Objects

Demo

Add a navigation property to an owned entity

Summary

Mapping value objects

Used value conversions to map a single-property value object: Email

- Cannot represent a nullable value object with a non-nullable property type

Used owned entity types to map a multi-property value object: Name

Owned entities support mapping to both same and separate tables

- Overcomplicated SQL
- Cannot share value objects between entities

It's better to drop the support for the separate tables use case

Adding a navigation property to a multi-property value object

In the Next Module

Implementing a Domain Event Dispatcher