Mapping Complex Types & Value Objects as Owned Types

Julie Lerman
MOST TRUSTED AUTHORITY ON ENTITY FRAMEWORK
@julielerman thedatafarm.com

Module Overview

Benefits of the owned type mapping

Create and map an owned type

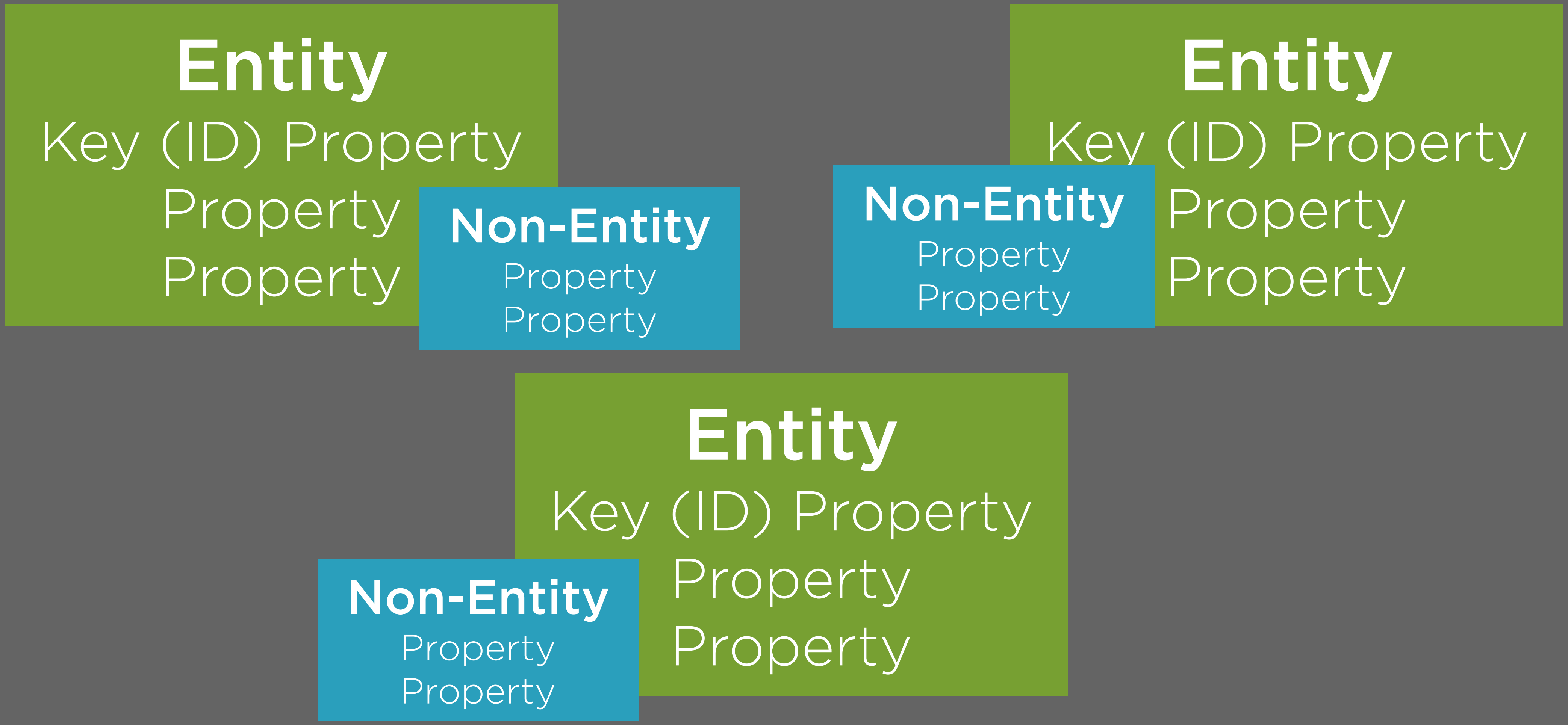
Retrieving and updating entities with owned type properties

Learn workarounds some current shortcomings

Map a DDD value object as an owned type

Why We Need the Owned Types Mapping

Types in Your Domain Model



Person Name Rules

Don't forget!

Must have a given name

Must have a surname

Person Name Type

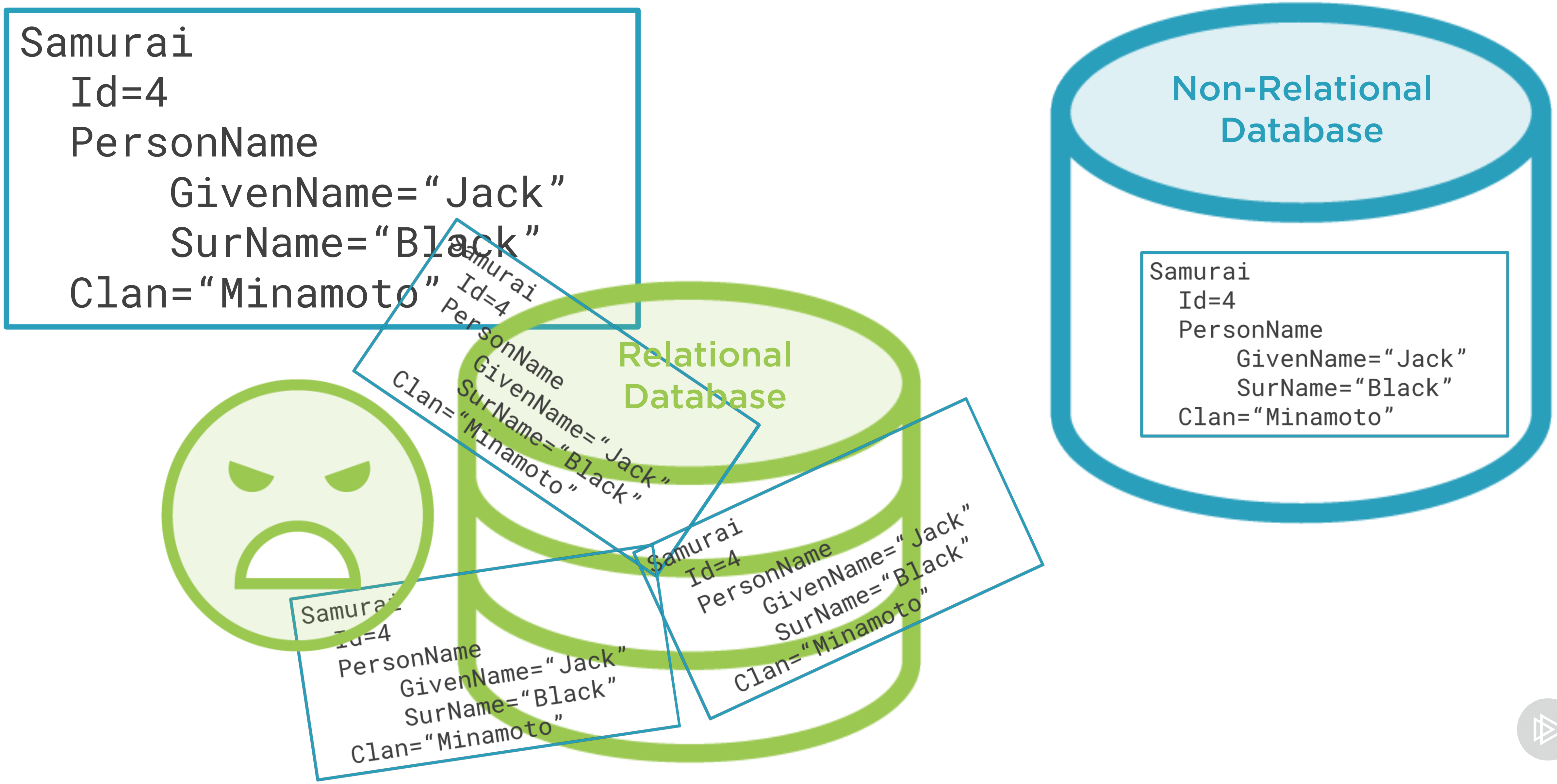
```
public class PersonName
{
   public PersonName(string givenName,string surName)
   {
     SurName = surName;
     GivenName = givenName;
   }
   public string GivenName { get;  set; }
   public string SurName { get;  set; }
   public string FullName => $"{GivenName} {SurName}";
   public string FullNameReverse => $"{SurName}, {GivenName}";
}
```

```
public class PersonName
{
   public PersonName(string givenName,string surName)
   {
     SurName = surName;
     GivenName = givenName;
   }
   public string GivenName { get;  set; }
   public string SurName { get;  set; }
   public string FullName => $"{GivenName} {SurName}";
   public string FullNameReverse => $"{SurName}, {GivenName}";
}
```

```
public class Samurai
{
  public int Id {get;set;}
  public PersonName Name {get;set;}
  ...
}
```

```
public class Contact
{
  public int Id {get;set;}
  public PersonName Name {get;set;}
  ...
}
```

Persisting the Shaped Data



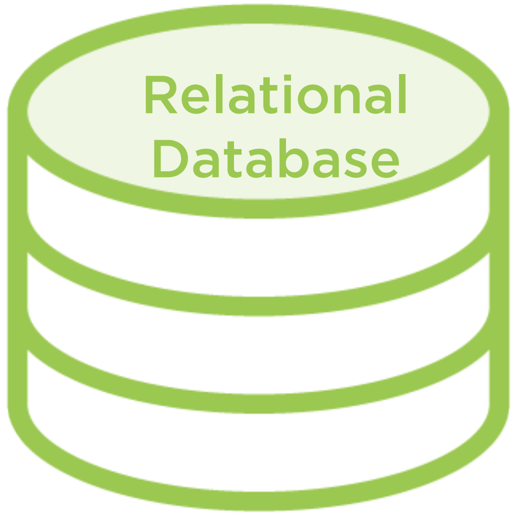
```
Samurai
  Id=4
  PersonName
       GivenName="Jack"
       SurName="Black"
  Clan="Minamoto"
```

Mapping the Shaped Data

```
Samurai
  Id=4
  PersonName
      GivenName="Jack"
      SurName="Black"
  Clan="Minamoto"
```

EF6: ComplexType Mapping

```
Samurai
  Id=4
  GivenName="Jack"
  SurName="Black"
  Clan="Minamoto"
```

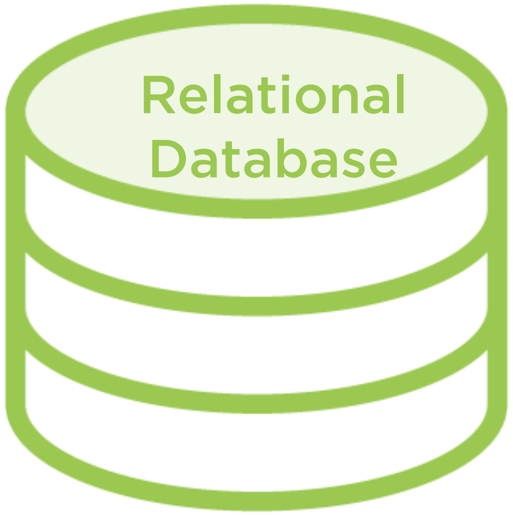


Mapping the Shaped Data

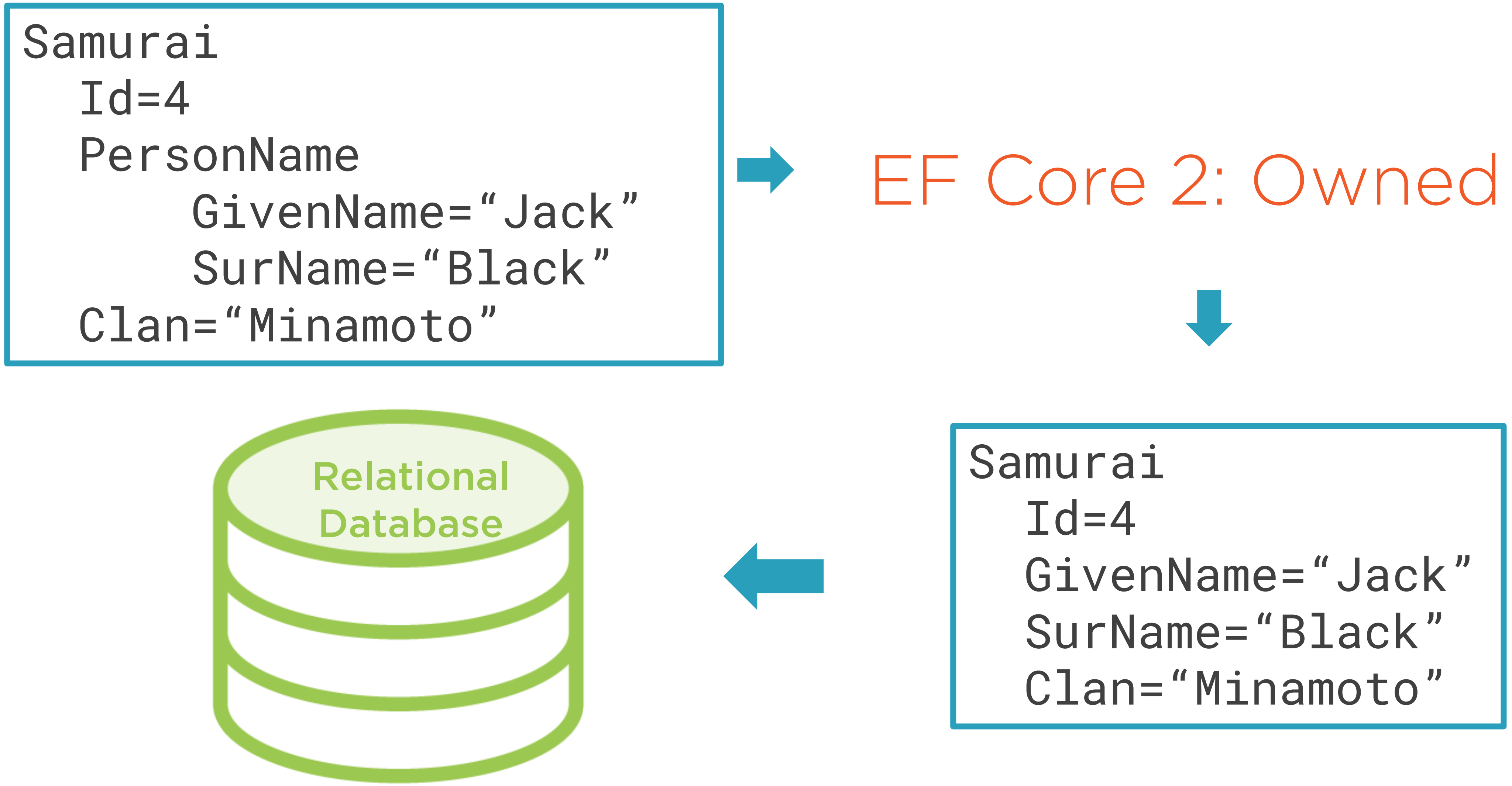
```
Samurai
  Id=4
  PersonName
      GivenName="Jack"
      SurName="Black"
  Clan="Minamoto"
```

EF Core 1: No Solution

```
Samurai
  Id=4
  GivenName="Jack"
  SurName="Black"
  Clan="Minamoto"
```



Mapping the Shaped Data



```
Samurai
  Id=4
  PersonName
      GivenName="Jack"
      SurName="Black"
  Clan="Minamoto"
```

EF Core 2: Owned Type

```
Samurai
  Id=4
  GivenName="Jack"
  SurName="Black"
  Clan="Minamoto"
```

Identifying Non-entity Types in Your Model

Mapping Complex Objects as Owned Types

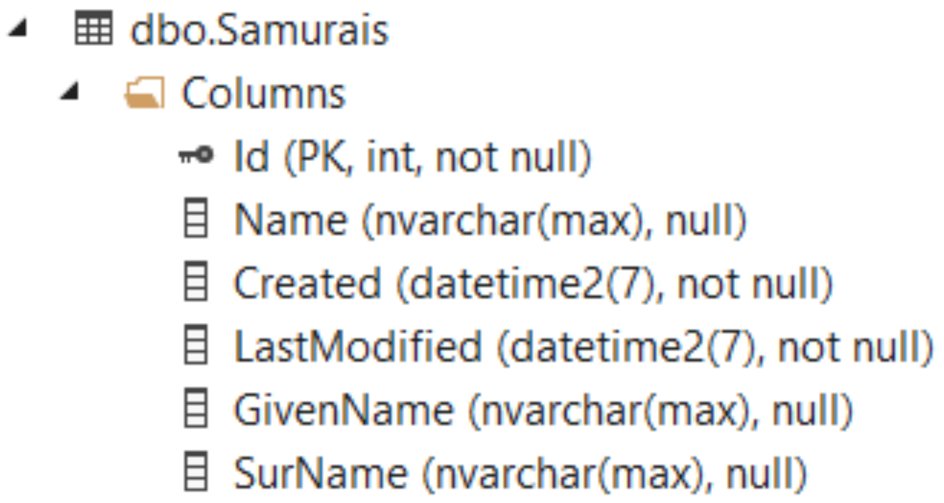
You have to explicitly map owned types

The owned type mapping uses shadow properties to do it’s job

Owned Entity Properties in Your DB

Convention

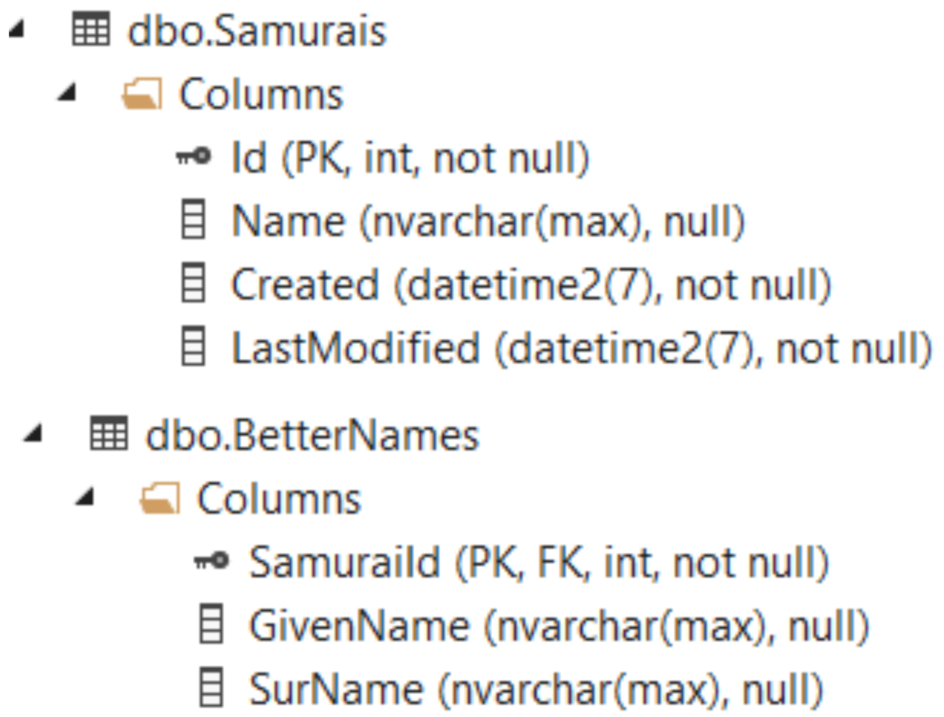
Columns go in same table as entity



```
dbo.Samurais
  Columns
    Id (PK, int, not null)
    Name (nvarchar(max), null)
    Created (datetime2(7), not null)
    LastModified (datetime2(7), not null)
    GivenName (nvarchar(max), null)
    SurName (nvarchar(max), null)
```

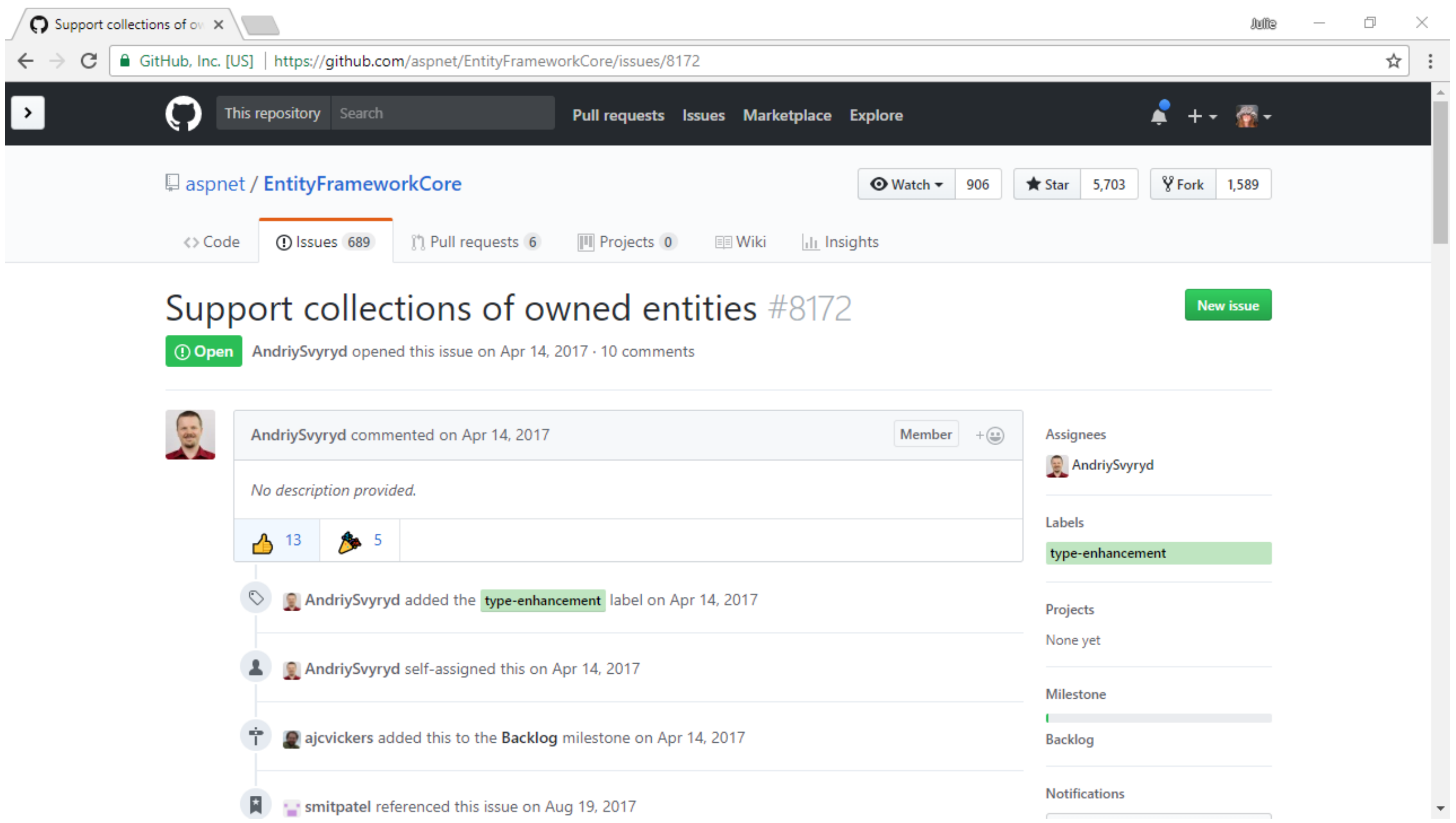
ToTable() Mapping

Split owned type into its own table



```
dbo.Samurais
  Columns
    Id (PK, int, not null)
    Name (nvarchar(max), null)
    Created (datetime2(7), not null)
    LastModified (datetime2(7), not null)
dbo.BetterNames
  Columns
    SamuraiId (PK, FK, int, not null)
    GivenName (nvarchar(max), null)
    SurName (nvarchar(max), null)
```

Support for collections of owned types is coming soon to EF Core


Support collections of ow

GitHub, Inc. [US] | https://github.com/aspnet/EntityFrameworkCore/issues/8172

aspnet / EntityFrameworkCore

Watch 906 Star 5,703 Fork 1,589

Code Issues 689 Pull requests 6 Projects 0 Wiki Insights

Support collections of owned entities #8172

New issue

Open AndriySvyryd opened this issue on Apr 14, 2017 · 10 comments

AndriySvyryd commented on Apr 14, 2017 Member

No description provided.

👍 13 🎉 5

AndriySvyryd added the type-enhancement label on Apr 14, 2017

AndriySvyryd self-assigned this on Apr 14, 2017

ajcvickers added this to the **Backlog** milestone on Apr 14, 2017

smitpatel referenced this issue on Aug 19, 2017

Assignees: AndriySvyryd

Labels: type-enhancement

Projects: None yet

Milestone: Backlog

Notifications

Working with Owned Types

Change Tracker & ModelBuilder Treat Owned Types Differently

Problem

ModelBuilder understands that owned types are not entities

Change Tracker does not understand this!

.

Solution

Make ChangeTracker aware of owned types.

`EntityEntry.Metadata.IsOwned()`

Inserting Entities with Null Owned Type Properties

Specific to EF Core 2.0 & 2.1*

*As per github.com/entityframeworkcore, this will continue in 2.1

This behavior will change (for the better) with a future version of EF Core

The EF Core 2 Gotchas

You must instantiate Samurai.BetterName

Owned type properties cannot be null

Setting Samurai.BetterName on an existing Samurai will try to add a second BetterName

You’ll need to help EF Core understand owned type replacements

EF Core Logic for Building Insert Commands

```
Samurai
    Name
    Date of Birth
    County of Origin

    BetterName
           GivenName
           SurName
```

- Read scalar values of entity object
- Read scalar values from the object that is the entity's owned type property

No conditional logic if the object does not exist!

How I feel about putting persistence rules into my business logic

Workaround for Non-Null Owned Type Rule

Mods to Owned Type Class

Factory methods: Create() & Empty()

Private Constructor

IsEmpty Property

Mods to SaveChanges

Replace null property with Empty() object

Replacing Owned Type Properties

EF Core 2 does not understand replacing owned type properties

```
var samurai=_context.Samurais
               .FirstOrDefault();
```

◂ Retrieve a samurai from the db

(BetterName property will exist and be populated)

```
samurai.BetterName =
 PersonFullName
 .Create(“Shohreh”,”Aghdashloo”);
```

◂ Set BetterName to another name

```
_context.SaveChanges();
```

◂ ChangeTracker will try to Add the new PersonFullName object to samurai

Workarounds for Weird Owned Type Rules

Mods to Owned Type Class	Mods to SaveChanges
For non-null owned type properties	
Factory methods: Create() & Empty() Private Constructor IsEmpty Property	Replace null property with Empty() object
To replace owned type properties	
No changes needed	Set state of owned type to the same state as its owner. If owner is modified, make owned type modified to update values.

Pattern for Leveraging the Replacement Fix

Replacing property when untracked

Connect to change tracker as an update

```
_context.Samurais.Update(samurai)
```

Replacing property when tracked

Detach original property’s entity

```
_context.Entry(samurai)
   .Reference(s=>s.BetterName)
   .TargetEntry.State=EntityState.Detached;
```

Set the new property

```
samurai.BetterName=PersonFullName.Create(“A",“B");
```

DbSet.Update

```
_context.Samurais.Update(samurai)
```

Mapping Value Objects as Owned Types

Value Objects

Objects that have **no identity key**, are used as **properties of other types** and are **identified by the composition of all of their property values**.

Value Object Rules

	EF Core Can Map as Owned Type
No identity key	✓
Exists only as a property	✓

Value Object Rules

	EF Core Can Map as Owned Type
No identity key	✓
Exists only as a property	✓
Immutable	
Equals compares all properties	
GetHashCode for all properties	

Value Object Rules

	EF Core Can Map as Owned Type
No identity key	✓
Exists only as a property	✓
Immutable	Impact is on in-memory objects, not data persistence
Equals compares all properties	
GetHashCode for all properties	

Value Object Rules

	PersonFullName
No identity key	✓
Exists only as a property	✓
Immutable	
Equals compares all properties	
GetHashKey for all properties	

You can use EF Core **owned types** to map **value objects** to a relational database

Another Example of a Value Object

```
public class MonetaryValue
{

  public int NumberOfUnits{..}

  public CurrencyEnum Currency{..}

}
```

```
public class MonetaryValue
{

  public int NumberOfUnits{..}

  public CurrencyEnum Currency{..}

  public DateTime Moment { ..}

}
```

NumberOfUnits=100
Currency=US Dollar

NumberOfUnits=100
Currency=Bitcoin

NumberOfUnits=100
Currency=Bitcoin

Value Object Rules

	PersonFullName
No identity key	✓
Exists only as a property	✓
Immutable	✓
Equals compares all properties	✓
GetHashKey for all properties	✓

Review

Why we have the owned type mapping

Not conventional, you must map it

No identity key & used as a property

You can also map value objects

Temporary workaround for null owned type properties

Temporary workaround to replace owned type properties

Support for collections of owned types coming

Resources

Entity Framework Core on GitHub github.com/aspnet/entityframework

EF Core Roadmap bit.ly/efcoreroadmap

EF Core Documentation docs.microsoft.com/ef

[Pluralsight] Entity Framework Core 2: Getting Started bit.ly/PS_EFCore2
[Pluralsight] Domain-Driven Design Fundamentals bit.ly/PS-DDD

Data Points - EF Core 2 Owned Entities and Temporary Work-Arounds
msdn.microsoft.com/magazine/mt846463

Follow status owned types on GitHub
Collection support: github.com/aspnet/EntityFrameworkCore/issues/8172
Optional (null support): github.com/aspnet/EntityFrameworkCore/issues/9005
Replacement: https://github.com/aspnet/EntityFrameworkCore/issues/9803

Mapping Value Objects and Complex Types as EF Core Owned Type

Julie Lerman
MOST TRUSTED AUTHORITY ON ENTITY FRAMEWORK

@julielerman   thedatafarm.com