Querying and Saving Related Data

Julie Lerman
MOST TRUSTED AUTHORITY ON ENTITY FRAMEWORK

@julielerman   thedatafarm.com

Module Overview

Inserting, update & deleting related data

Saving related data that wasn’t tracked

Eager loading queries and shaping results with projections

Loading related data for objects in memory

Filtering queries with related data

Querying and persisting across many-to-many relationships

Querying and persisting across one-to-one relationships

Inserting Related Data

Change Tracker Response to New Child of Existing Parent

As child’s key value is not set, state will automatically be “Added”

Child’s FK value to parent (e.g. Quote.SamuraiId) is set to parent’s key

DbContext/DbSet Tracking Methods

- ✓ Add
- ✓ Update
- ✓ Remove
- ✓ Attach

EF Core's Default Entity State of Graph Data

	Has Key Value	No Key Value
Add(graph)	Added*	Added
Update(graph)	Modified	Added
Attach(graph)	Unchanged	Added

*Database will throw an exception if IDENTITY INSERT is illegal (default)

“Foreign keys? NEVER!
They will make my classes dirty!”

“Foreign keys in my classes
make my life so much simpler!”

Eager Loading Related Data

Methods to Load Related Data

Eager Loading

Include related objects in query

Query Projections

Define the shape of query results

Explicit Loading

Request related data of objects in memory

Lazy Loading*

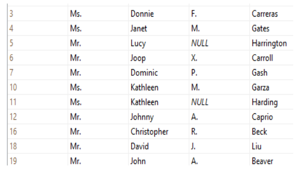
On-the-fly retrieval of related data

*Arrived with EF Core 2.1

Query Workflow

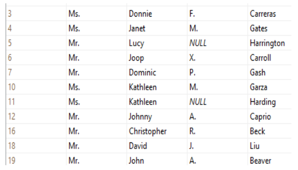
Receives tabular results

Samurais



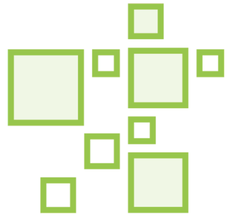
3	Ms.	Donnie	F.	Carreras
4	Ms.	Janet	M.	Gates
5	Mr.	Lucy	*NULL*	Harrington
6	Mr.	Joop	X.	Carroll
7	Mr.	Dominic	P.	Gash
10	Ms.	Kathleen	M.	Garza
11	Ms.	Kathleen	*NULL*	Harding
12	Mr.	Johnny	A.	Caprio
16	Mr.	Christopher	R.	Beck
18	Mr.	David	J.	Liu
19	Mr.	John	A.	Beaver

Quotes for those Samurais



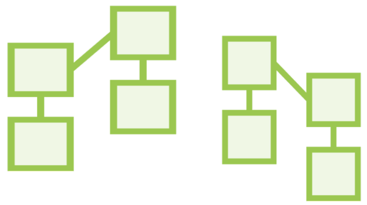
3	Ms.	Donnie	F.	Carreras
4	Ms.	Janet	M.	Gates
5	Mr.	Lucy	*NULL*	Harrington
6	Mr.	Joop	X.	Carroll
7	Mr.	Dominic	P.	Gash
10	Ms.	Kathleen	M.	Garza
11	Ms.	Kathleen	*NULL*	Harding
12	Mr.	Johnny	A.	Caprio
16	Mr.	Christopher	R.	Beck
18	Mr.	David	J.	Liu
19	Mr.	John	A.	Beaver

Materializes results as objects



Adds tracking details to DbContext instance

DbContext connects the relationships



More Ways to Use Include

```
_context.Samurais
.Include(s => s.Quotes)
.ThenInclude(q=>q.Translations)
.FirstOrDefault();
```

- Get quotes for the samurai
- Then get the translations for those quotes

```
_context.Samurais
.Include(s => s.Quotes)
.Include(s=>s.Clan)
.FirstOrDefault();
```

- Get quotes for samurais
- Also get the clan for samurais

```
_context.Samurais
  .Include(s=>s.Quotes)
```

◂ Include child objects

```
_context.Samurais
  .Include(s=>s.Quotes)
  .ThenInclude(q=>q.Translations)
```

◂ Include children & grandchildren

```
_context.Samurais
 .Include(s=>s.Quotes.Translations)
```

◂ Include just grandchildren

```
 _context.Samurais
  .Include(s=>s.Quotes)
  .Include(s=>s.Clan)
```

◂ Include different children

…Various combinations…

`Include` always loads the entire set of related objects.

Projecting Related Data in Queries

EF Core can only track entities recognized by the DbContext model.

Anonymous types
are not tracked

Entities that are properties of an anonymous type
are tracked

Loading Related Data for Objects Already in Memory

Methods to Load Related Data

Eager Loading

Include related objects in query

Query Projections

Define the shape of query results

Explicit Loading

Request related data of objects in memory

Lazy Loading*

On-the-fly retrieval of related data

*Arrived with EF Core 2.1

```
With samurai object already in memory

_context.Entry(samurai).Collection(s => s.Quotes).Load();

_context.Entry(samurai).Reference(s => s.Horse).Load();
```

Explicit Loading

Explicitly retrieve related data for objects already in memory

DbContext.Entry().Collection().Load()

DbContext.Entry().Reference().Load

More on Explicit Loading

You can only load from a single object

- Profile to determine if LINQ query would be better performance

Filter loaded data using the Query method

```
var happyQuotes = context.Entry(samurai)
      .Collection(b => b.Quotes)
      .Query()
      .Where(q => q.Quote.Contains(“Happy”)
      .ToList();
```

Lazy Loading

is

OFF

by default

Lazy Loading

Happens implicitly by mention of the navigation

Enable with these requirements:

Every navigation property must be virtual

Microsoft.EntityFramework.Proxies package

ModelBuilder.UseLazyLoadingProxies()

Many “gotchas” to be wary of

```
foreach(var q in samurai.Quotes)
{
  Console.WriteLine(q.Text);
}
```

This will send one command to retrieve all of the Quotes for that samurai, then iterate through them

```
var qCount=samurai.Quotes.Count();
```

This will retrieve all of the quote objects from the database and materialize them and then give you the count.

Data bind a grid to lazy-loaded data

This happened a lot in ASP.NET pages. The grid populate one row at a time and lazy loads the related data for that row, then the next, then the next. N+1 commands sent to the database!

Lazy loading when no context in place

No data is retrieved

Using Related Data to Filter Objects

Modifying Related Data

Connected

Disconnected

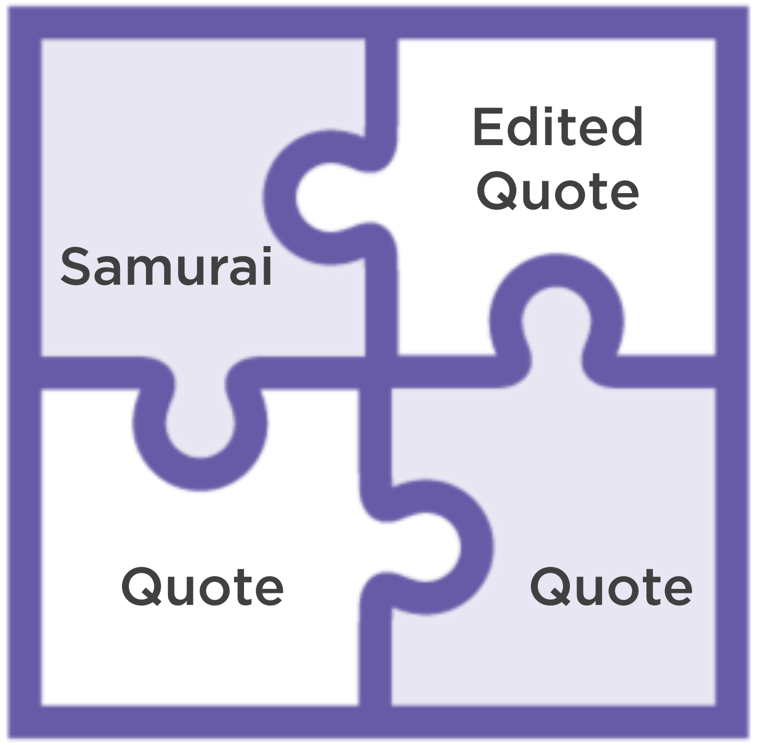
Connected

DbContext
is aware
of all changes
made to objects
that is it tracking

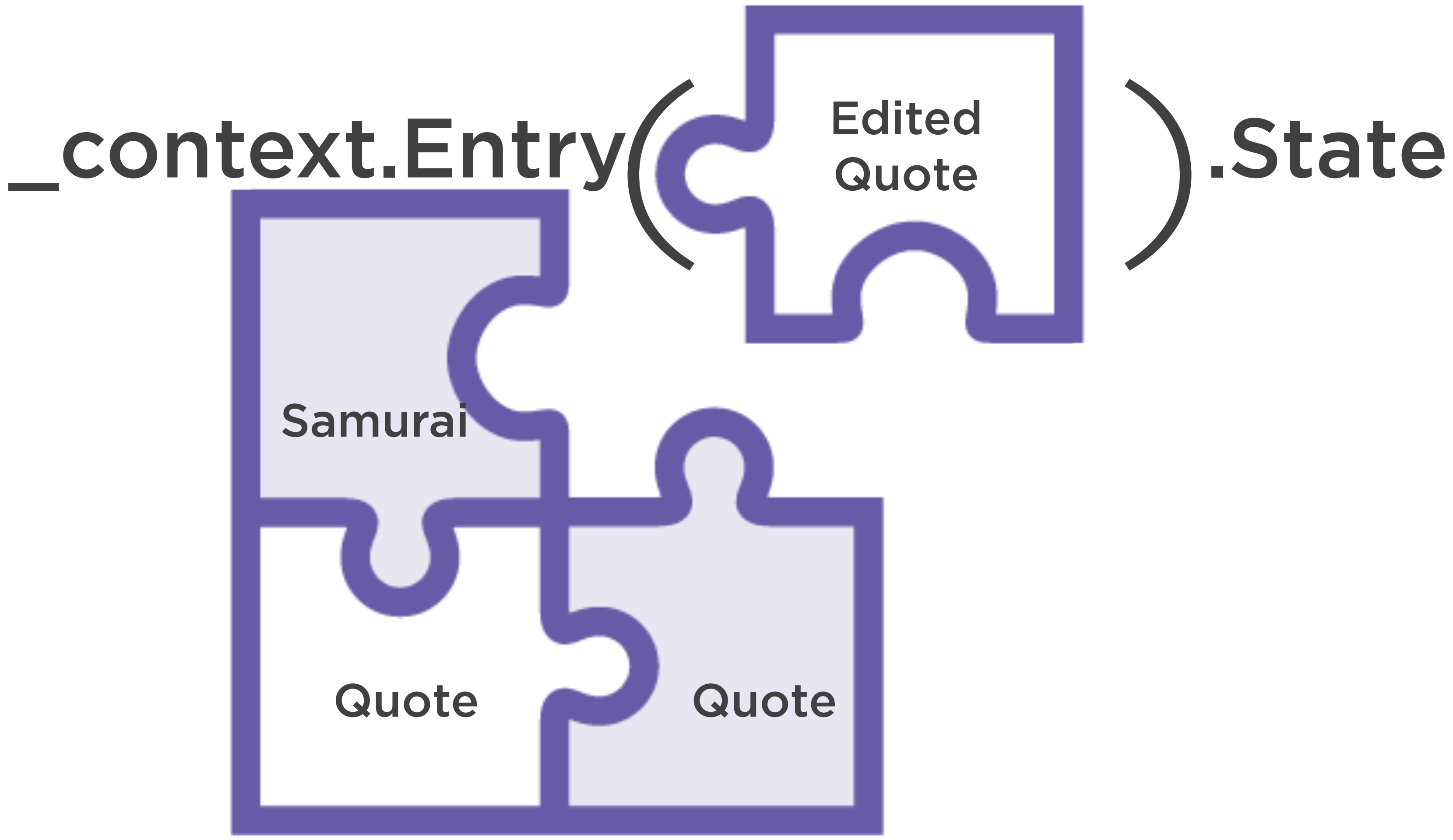
Disconnected

DbContext
has no clue
about
history of objects
before they are
attached

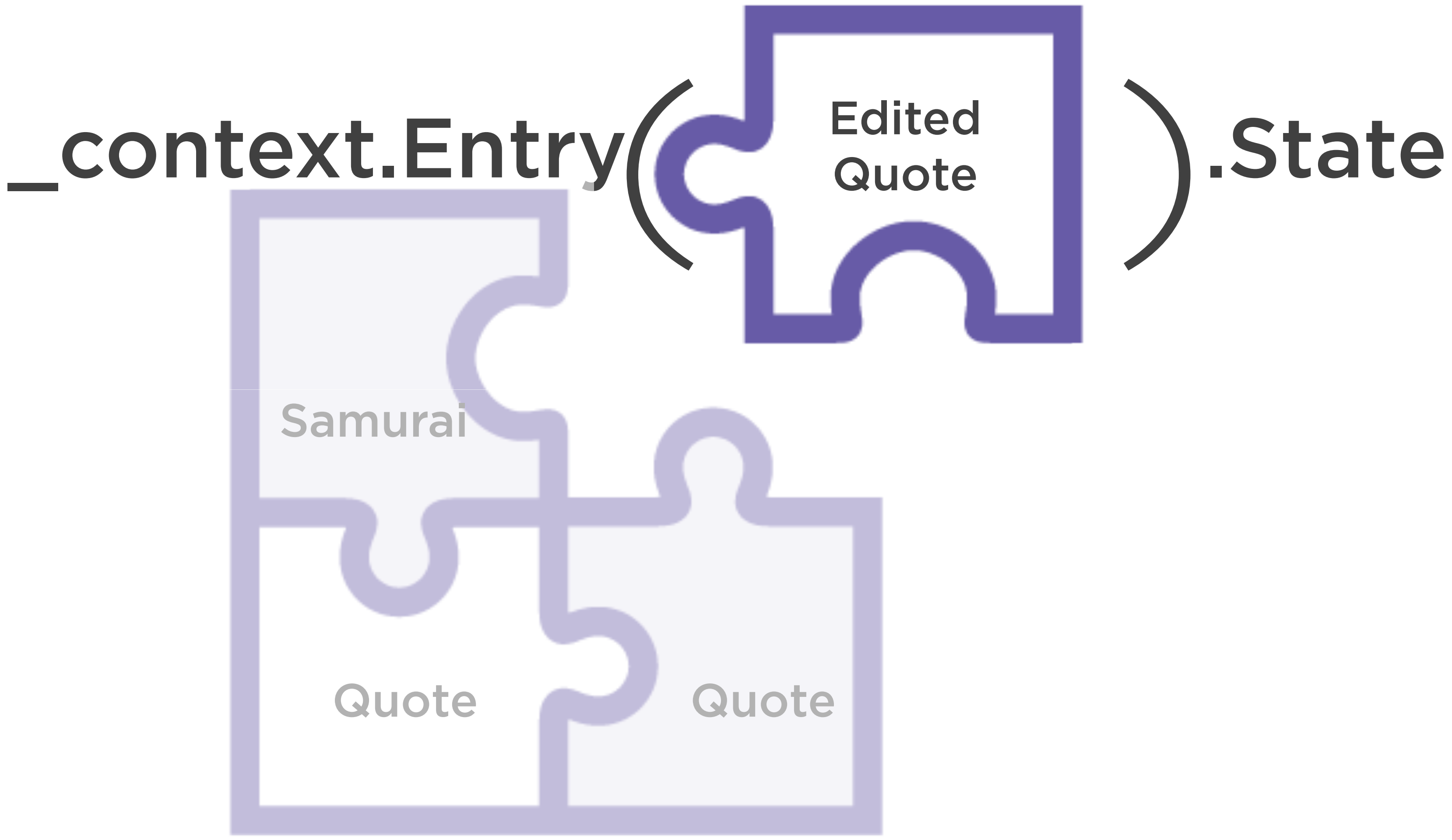
The Challenge



The Challenge

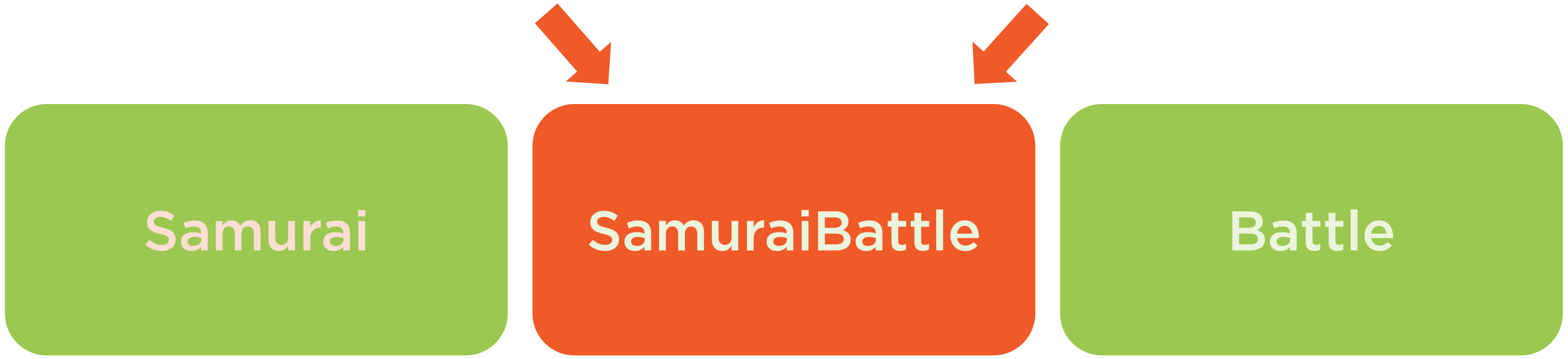


The Challenge



Creating and Changing Many-to-Many Relationships

Modifying or Deleting the Join, Not the Ends



Modifying the Join

Samurai

`Id=1`

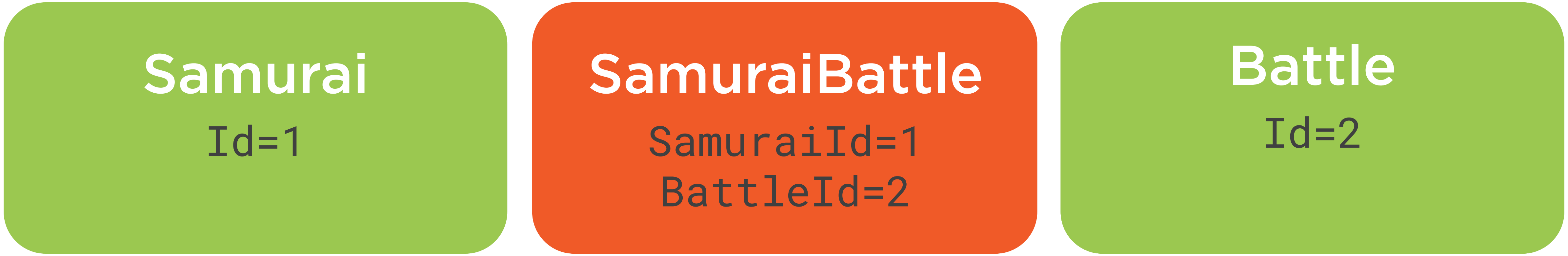
SamuraiBattle

`SamuraiId=1`
`BattleId=`**`2`**

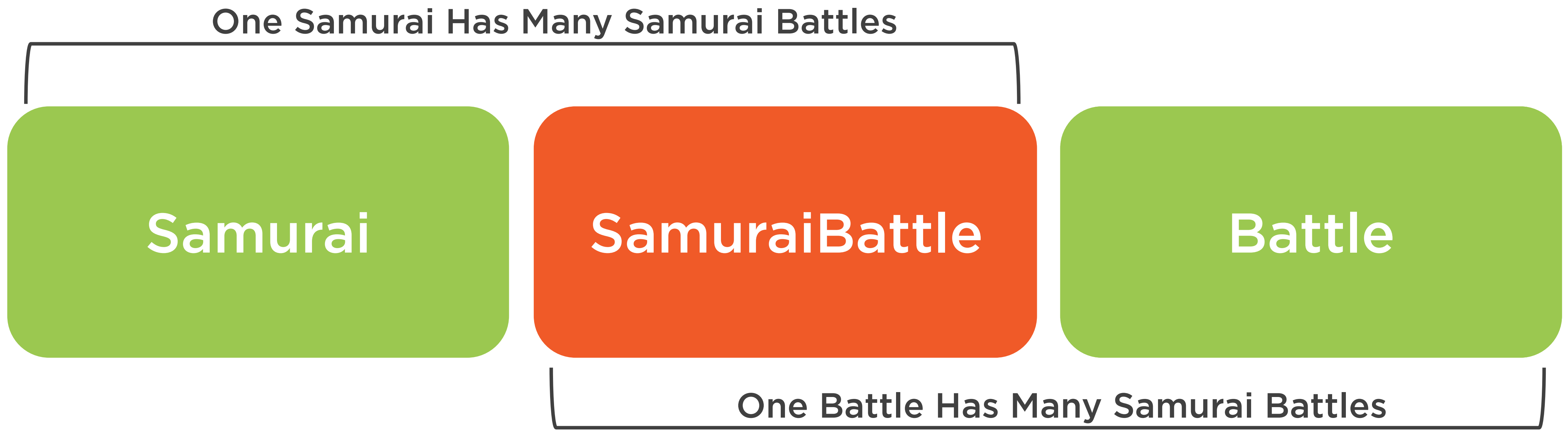
Battle

`Id=2`

Replacing the Join
aka Delete a Join, Then Add a Join

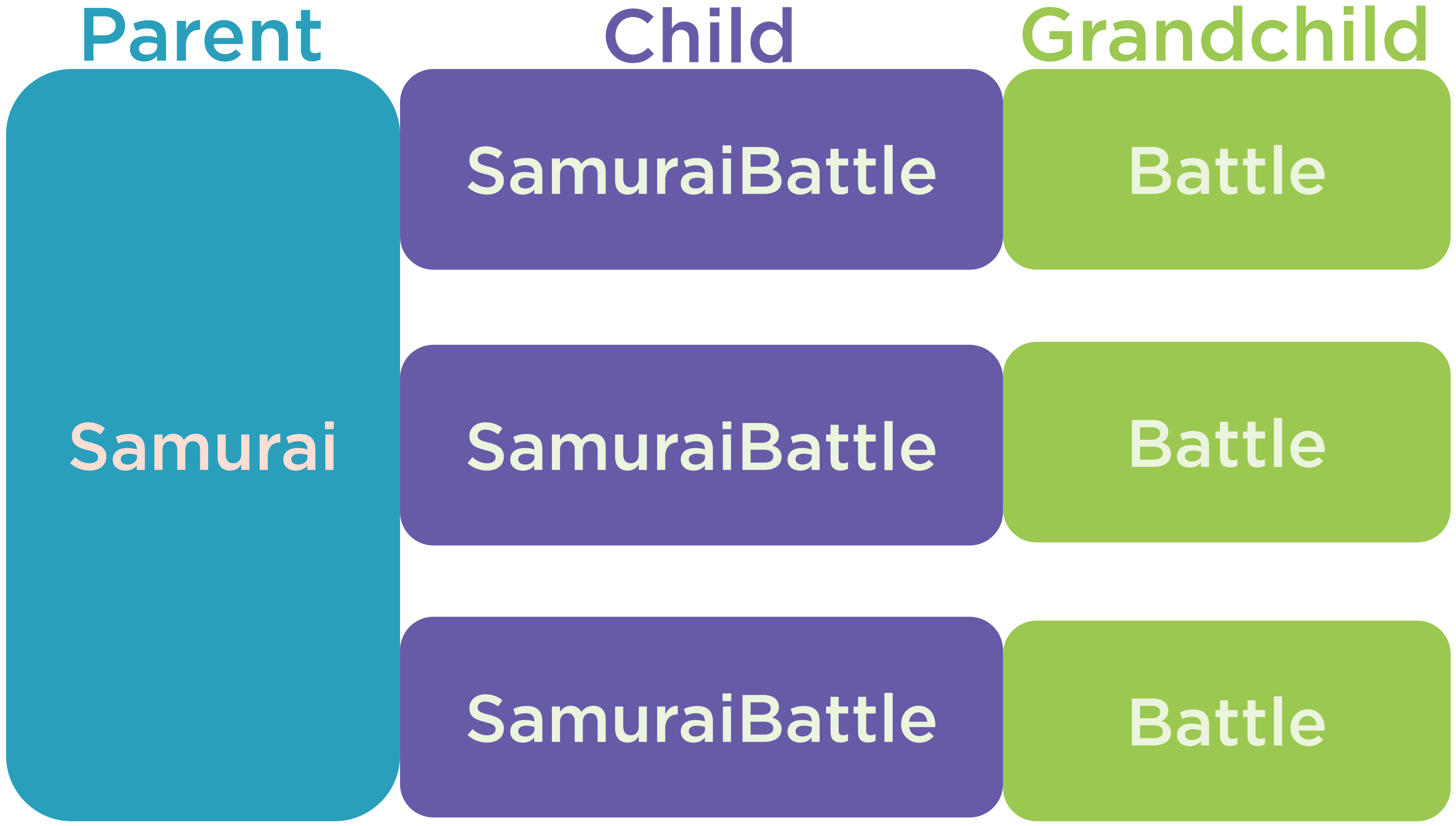


Many-to-Many Is a Pair of One-to-Many

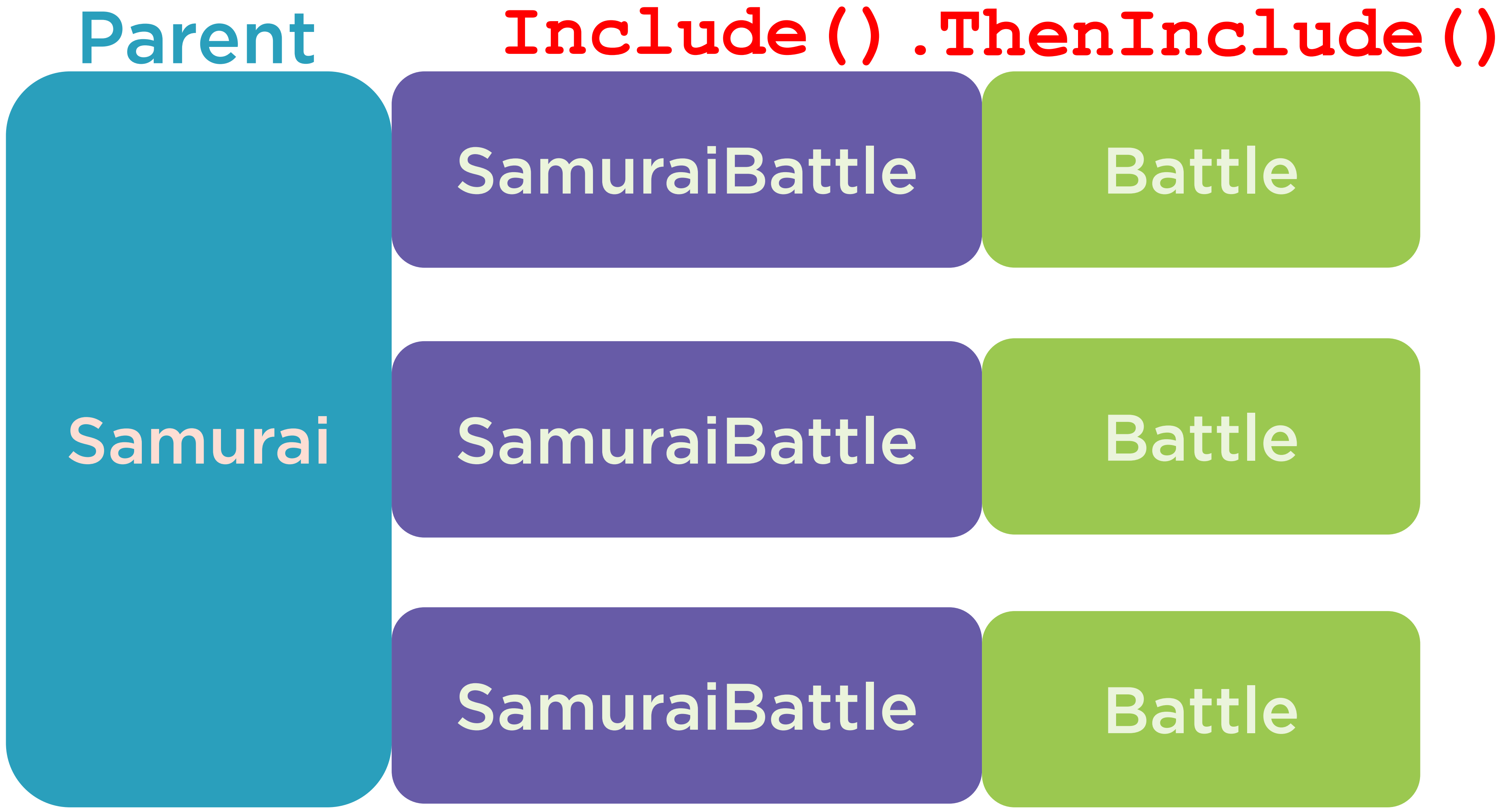

One Samurai Has Many Samurai Battles
Samurai
SamuraiBattle
Battle
One Battle Has Many Samurai Battles

Querying Across Many-to-Many Relationships

Many-to-Many: Just Children & Grandchildren



Many-to-Many: Just Children & Grandchildren

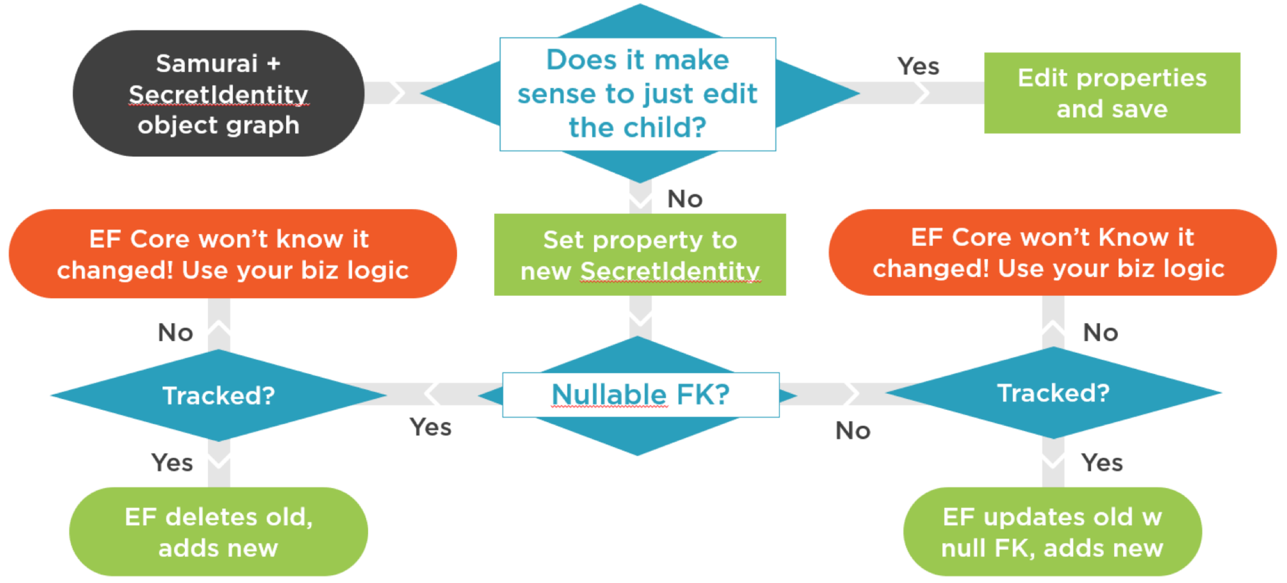


Persisting Data in One-to-One Relationships

Changing the Child of an Existing Parent

- ✓ Is foreign key nullable?
- ✓ Is the child object in memory?
- ✓ Are the objects being tracked?

In EF Core 2: Mappings Course


Samurai + SecretIdentity object graph
Does it make sense to just edit the child?
Yes
Edit properties and save
No
Set property to new SecretIdentity
Nullable FK?
Yes
Tracked?
No
EF Core won't know it changed! Use your biz logic
Yes
EF deletes old, adds new
No
Tracked?
No
EF Core won't Know it changed! Use your biz logic
Yes
EF updates old w null FK, adds new

Querying One-to-One Relationships

Working with a Relationship that has Minimal Properties

"Clean" entities may be more difficult to work with, requiring more advanced skills with EF Core

Review

You can eager load related data or load after the fact

Be sure to understand Lazy Loading before using it!

Filter/sort related data with projections and Load

Important to understand how EF Core treats graphs that were not being tracked

DbContext.Entry() isolates the object you care about

m:m and 1:1 can get complicated! Learn more in the EF Core 2: Mappings course

Up Next!

Raw SQL,
Stored Procedures &
Database Views

Resources

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

EF Core Documentation docs.microsoft.com/ef

EF Core Power Tools on GitHub github.com/ErikEJ/EFCorePowerTools/wiki

Entity Framework Core 3.0: A Foundation for the Future
codemag.com/Article/1911062/Entity-Framework-Core-3.0-A-Foundation-for-the-Future

EF Core 2: Mappings (Pluralsight course) bit.ly/2LppcMj

Querying and Saving Related Data

Julie Lerman
MOST TRUSTED AUTHORITY ON ENTITY FRAMEWORK

@julielerman thedatafarm.com