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

SQL Generated from Includes

Single Query with Left Join(s)

Default

Query is broken up into multiple queries sent in a single command

With AsSplitQuery()

Eager Loading with Include

EF->EF6 (2008 -> today)

Include returns 100% of related objects

EF Core 1, 2 & 3 (2016-2019)

Include returns 100% of related objects

EF Core 5 (2020)

Filtered Include!

Eager Loading with Include

EF->EF6 (2008 -> today)

Include returns 100% of related objects

EF Core 1, 2 & 3 (2016-2019)

Include returns 100% of related objects

EF Core 5 (2020)

Filtered Include!

Eager Loading with Include

EF->EF6 (2008 -> today)

Include returns 100% of related objects

EF Core 1, 2 & 3 (2016-2019)

Include returns 100% of related objects

EF Core 5 (2020)

Filtered Include!

Eager Loading with Include

EF Core 5 (2020)

Filtered Include!

Eager Loading with Include

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

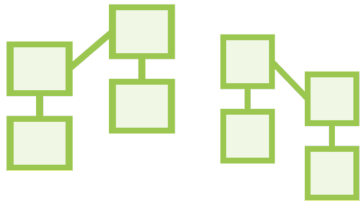
Quotes for those Samurais

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)

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

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

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

- Include child objects
- Include children & grandchildren
- Include just grandchildren
- Include different children

...Various combinations...

Projecting Related Data in Queries

EF Core Can Only Track Entities Recognized by the DbContext

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 in every entity must be virtual
Microsoft.EntityFramework.Proxies package
OnConfiguring optionsBuilder.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

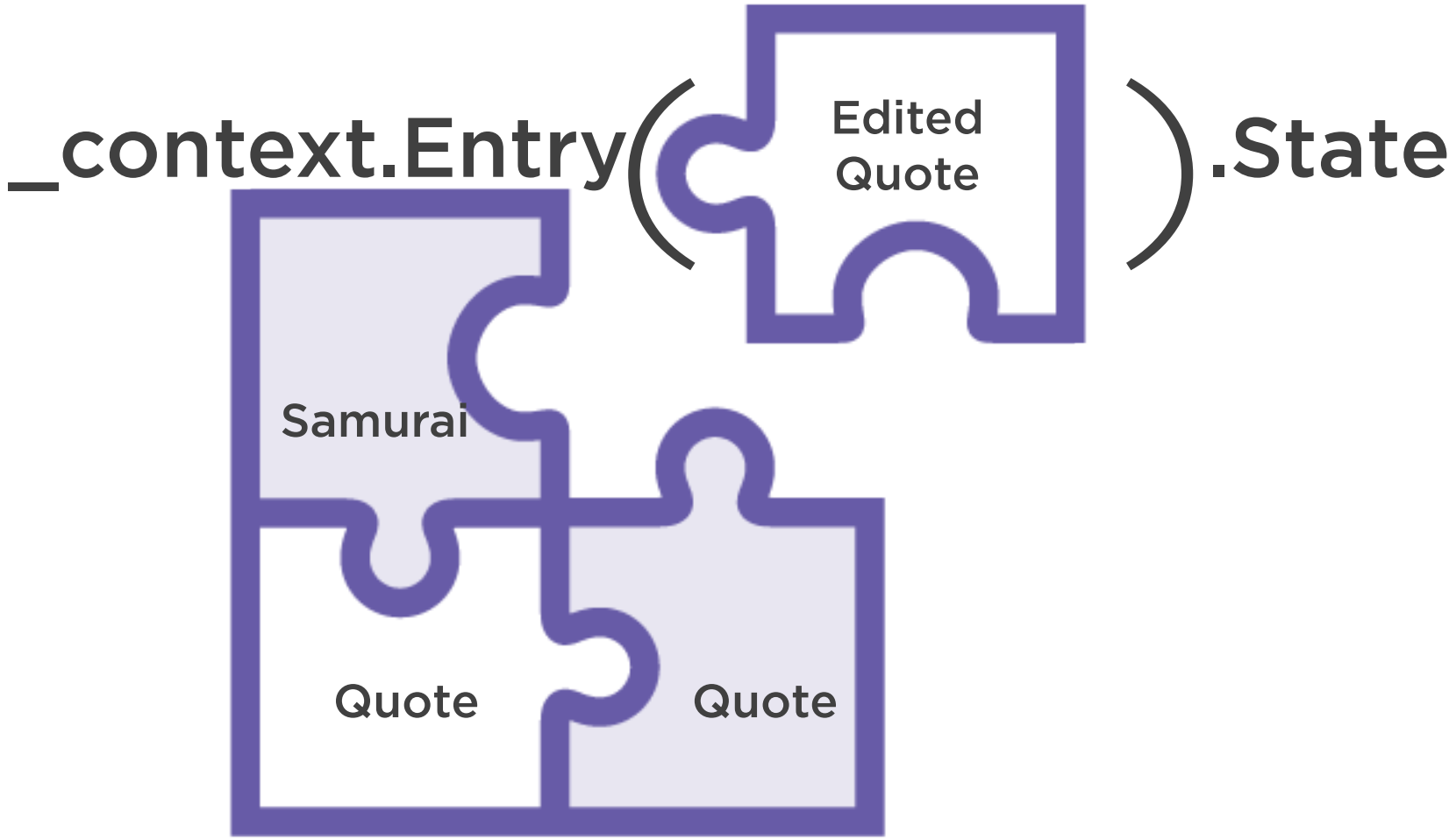
DbContext
is aware
of all changes
made to objects
that is it tracking

Disconnected

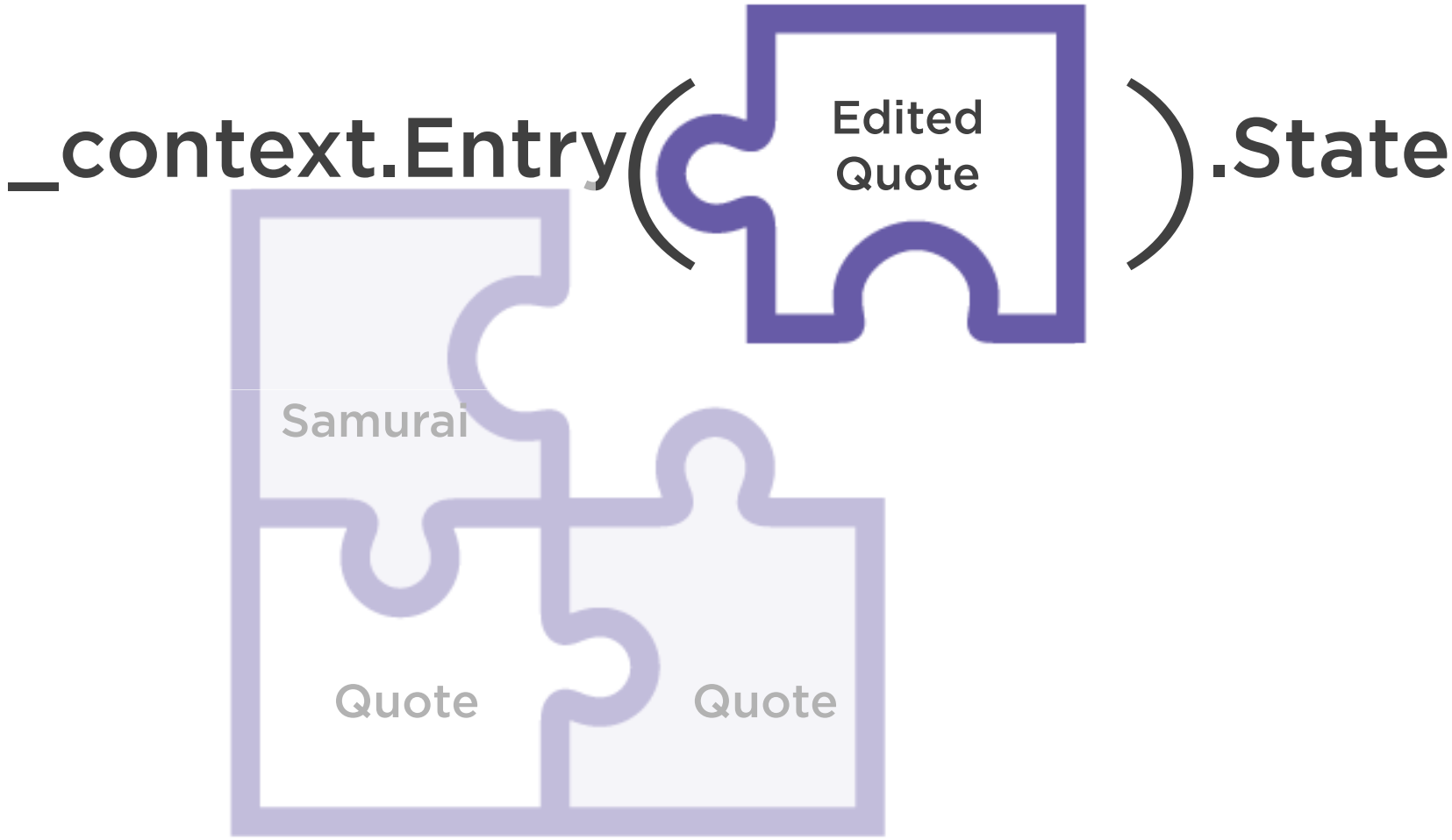
DbContext
has no clue
about
history of objects
before they are
attached

The Challenge

The Challenge



The Challenge



Working with Many-to-Many Relationships

More Likely to Join Existing Objects

Existing Battle
+
Existing Samurai

Experienced samurai joins an ongoing battle

Existing Battle
+
New Samurai

Rookie Samurai joins an ongoing battle

New Battle
+
New Samurai

Rookie Samurai started a war

More Likely to Join Existing Objects

Existing Class
+
Existing student

Registered student signs up for a scheduled class

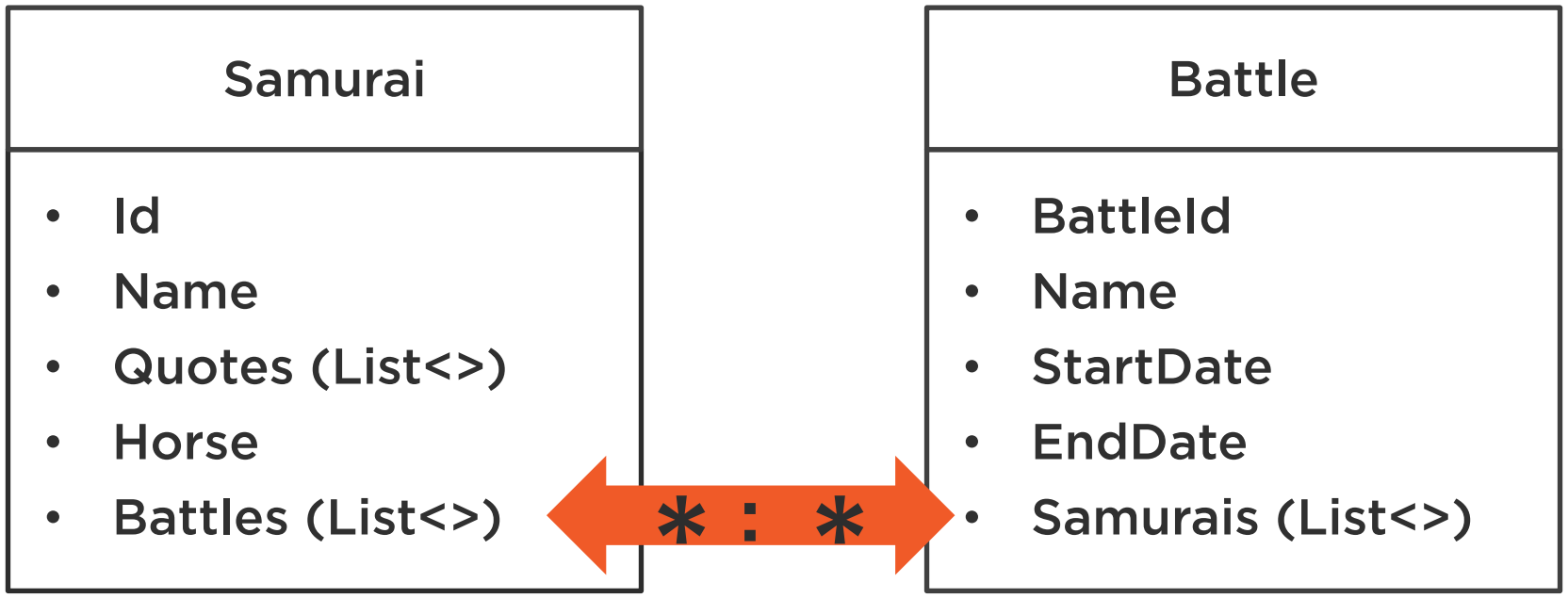
Existing class
+
New student

Unregistered student signs up for a scheduled class

New class
+
New student

Unregistered student schedules a new class

Start with Our First Many-to-Many Setup



Simplest convention
No mappings needed
Table is inferred by DbContext

One-to-Many vs. Many-to-Many

Quote
• Id • Text • SamuraiId • Samurai

Samurai
• Id • Name • Quotes (List<>) • Horse • Battles (List<>)

Battle
• BattleId • Name • StartDate • EndDate • Samurais (List<>)

One Samurai to Many Quotes

Many Samurais to Many Quotes

Let’s focus on the SQL generated for our many-to-many relationships

EF Core supports optimistic concurrency. See docs for details on how to handle exceptions like this.

Altering or Removing Many-to-Many Joins

Battle of Nagashino

Battle of Anegawa

Changing a Join: Preferred Workflow

1. Remove the original join
2. Then create the new join

Be mindful of side effects from your business logic

Deleting a M2M Relationship is Easier With

Stored Procedure

Explicit M2M Mapping

Adding a Many-to-Many Payload to Existing Join Table & Data

Migrating to explicit mapping won't impact existing data ...

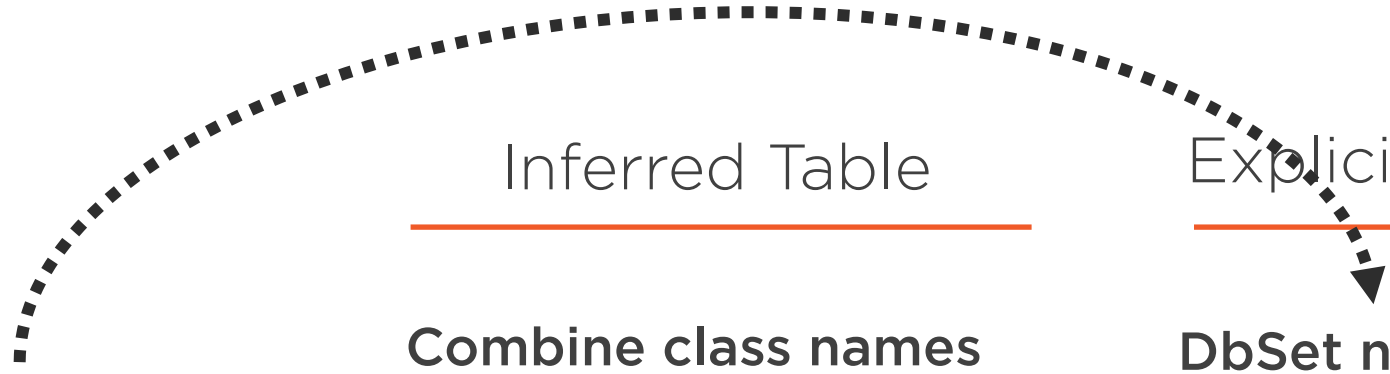
Migrating to explicit mapping won't impact existing data ...

 **UNLESS....** 

Beware of Migration Conventions Conflicts

	Inferred Table	Explicit Class & Table Mapping
Table name	**Combine class names alphabetically:** BattleSamurai	**DbSet name (doesn't exist) or Class Name (BattleSamurai)**
Column names	**Table name + PK name:** SamuraisId BattlesBattleId	**Property names from class** SamuraiId BattleId

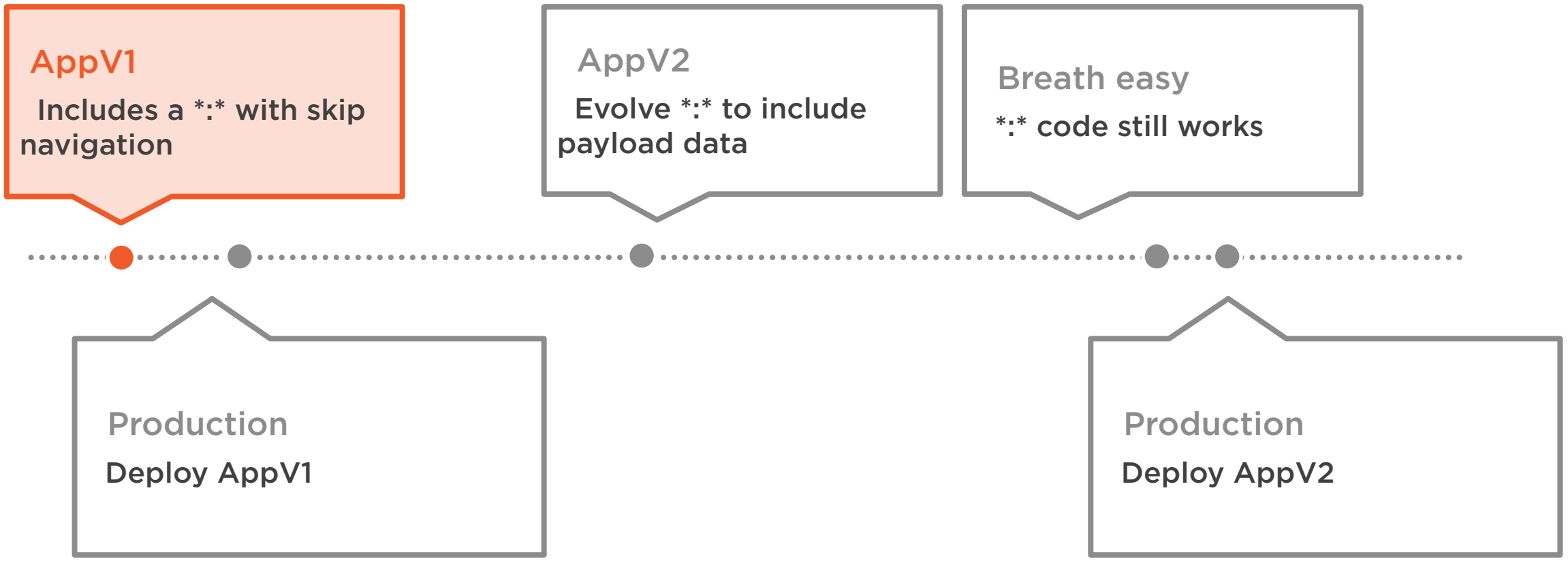
Beware of Migration Conventions Conflicts



	Inferred Table	Explicit Class & Table Mapping
Table name	**Combine class names alphabetically:** BattleSamurai	**DbSet name (doesn't exist) or Class Name (BattleSamurai)**
Column names	**Table name + PK name:** SamuraisId BattlesBattleId	**Property names from class** SamuraiId BattleId

Working with the Many-to-Many Payload Data

Skip Navigations Can Evolve Safely



Timeline of Events with Fade Transition

AppV1

Includes a *:* with skip navigation

AppV2

Evolve *:* to include payload data

Breath easy

: code still works

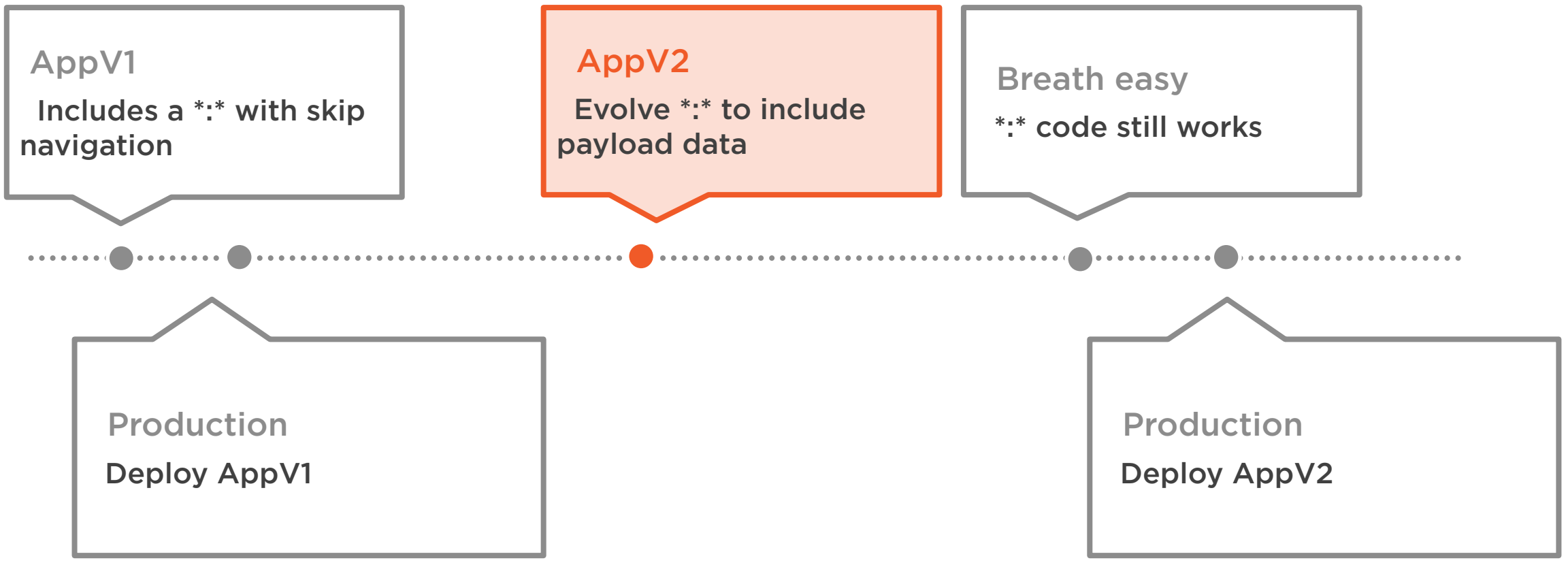
Production

Deploy AppV1

Production

Deploy AppV2

Timeline of Events with Fade Transition



Timeline of Events with Fade Transition

AppV1

Includes a *:* with skip navigation

AppV2

Evolve *:* to include payload data

Breath easy

: code still works

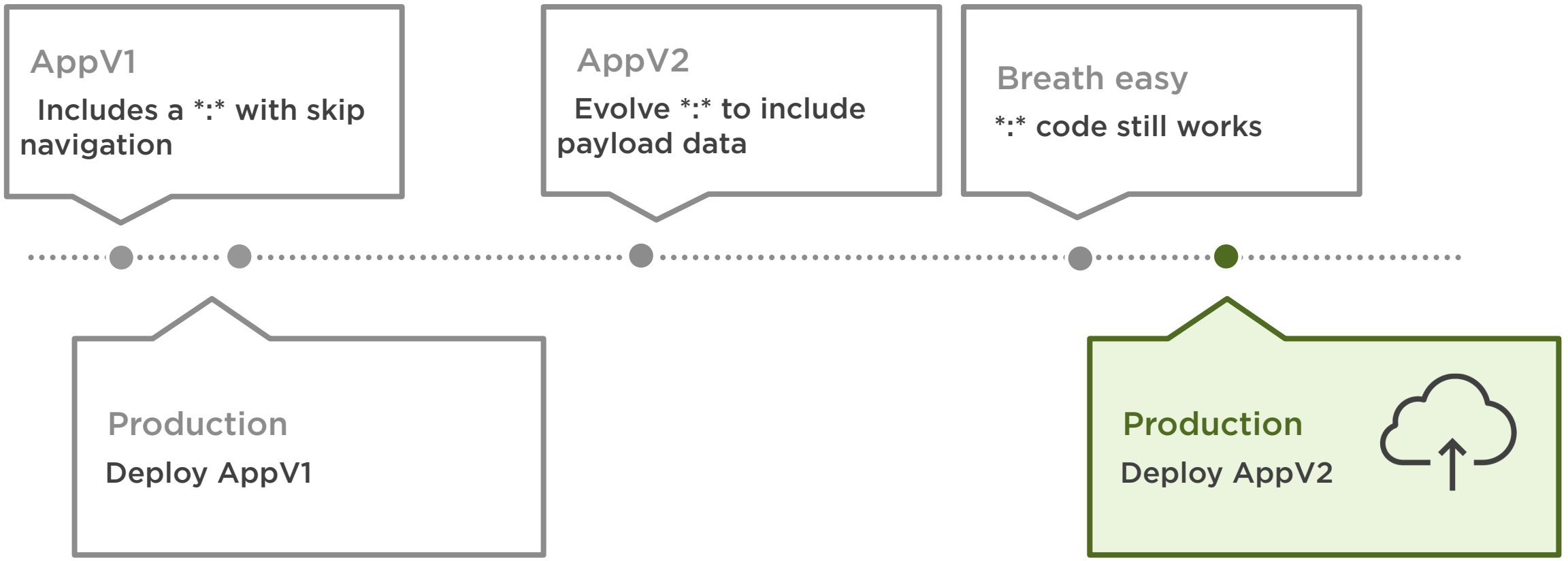
Production

Deploy AppV1

Production

Deploy AppV2

Timeline of Events with Fade Transition



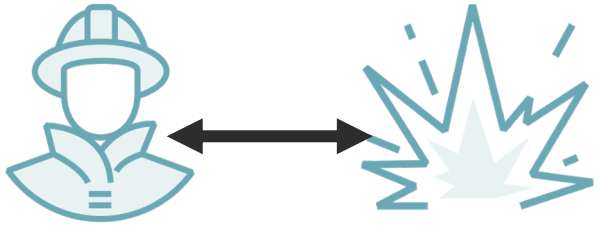
Adding payload mapping to existing skip navigation doesn't break existing code

Explicit Many-to-Many Skip Navigation

```
public class BattleSamurai
{
    public int SamuraiId { get; set; }
    public int BattleId { get; set; }
    public DateTime DateJoined { get; set; }
}
```

```
public class BattleSamurai
{
    public int SamuraiId { get; set; }
    public int BattleId { get; set; }
}
```

```
protected override void OnModelCreating(ModelBuil
{
    modelBuilder.Entity<Samurai>()
     .HasMany(s => s.Battles)
     .WithMany(b => b.Samurais)
     .UsingEntity<BattleSamurai>
      (bs => bs.HasOne<Battle>().WithMany(),
       bs => bs.HasOne<Samurai>().WithMany())
     .Property(bs => bs.DateJoined)
     .HasDefaultValueSql("getdate()");
```

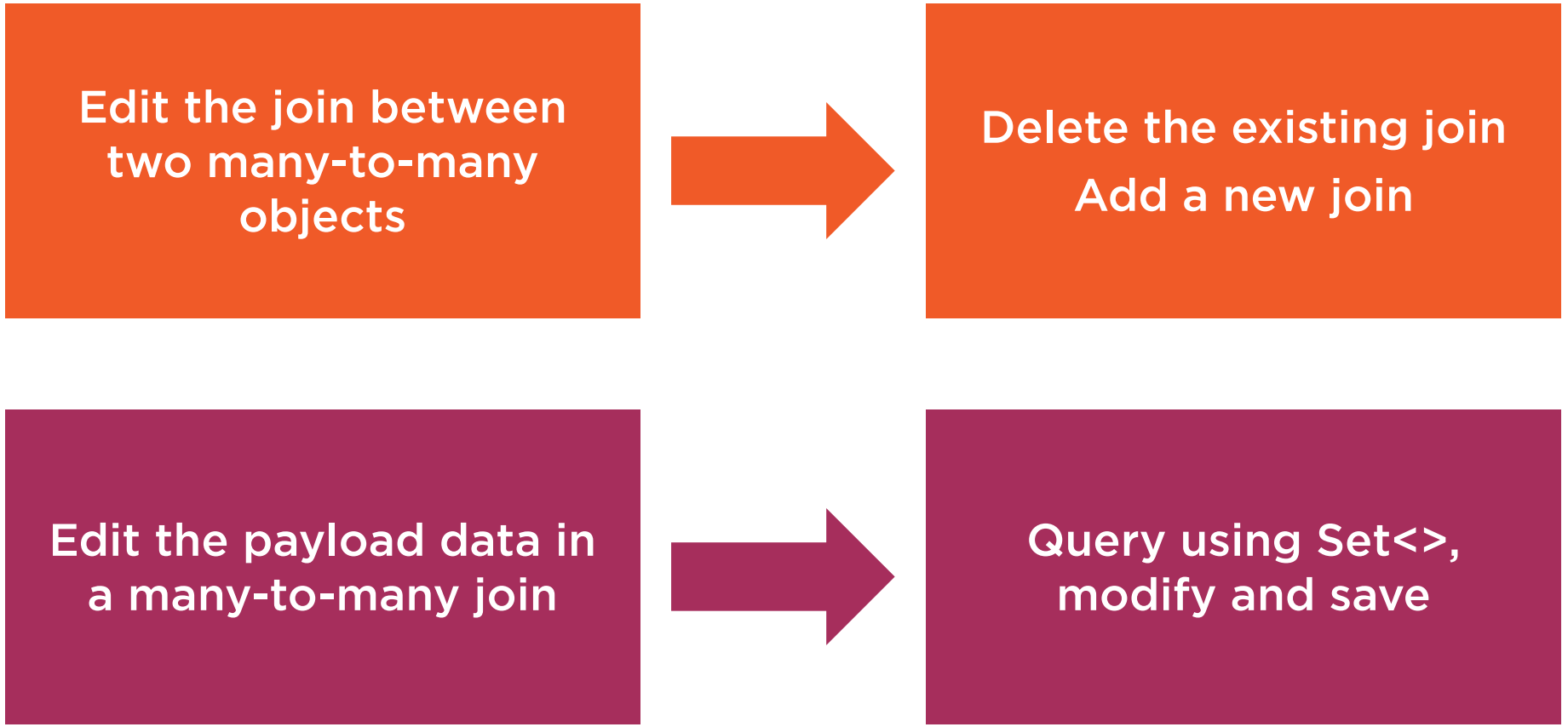


Join Entity w Optional Payload

Join Mapping

Skip Navigation

Guidance for Editing Many-to-Many



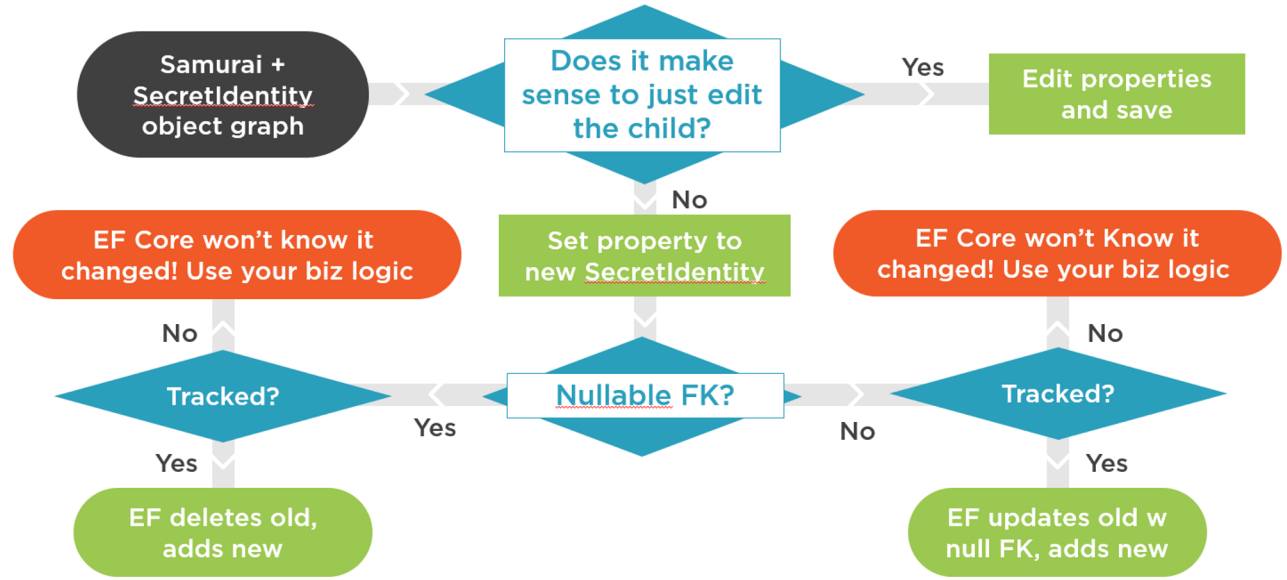
Querying Across Many-to-Many Relationships

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



Querying One-to-One Relationships

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

Review

You can eager load related data with Include or load after the fact

Pay attention to lazy loading behavior

EF Core 5 finally gives us filtered include

Understand behavior of untracked graphs

DbContext.Entry() isolates the object you care about

Intelligent SQL for *:* skip navigations

: payload data won't harm your code

Lots of rules to understand for *:* and 1:1

Up Next!

Raw SQL,
Stored Procedures &
Database Views

Resources

Entity Framework Core on GitHub github.com/dotnet/efcore

EF Core Documentation docs.microsoft.com/ef

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

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

Handling Concurrency Conflicts in EF Core docs.microsoft.com/en-us/ef/core/saving/concurrency

Arthur Vickers' (EF team) Many-to-Many examples: github.com/ajcvickers/ManyToManySamples

Querying and Saving Related Data

Julie Lerman
MOST TRUSTED AUTHORITY ON ENTITY FRAMEWORK

@julielerman thedatafarm.com