Interacting with Related Data

Julie Lerman
Most Trusted Authority on Entity Framework Core

@JulieLerman   www.thedatafarm.com

Module Overview

Focus on the one-to-many authors & books

Inserting related data

Eager loading queries and shaping results with projections

Loading related data for objects in memory

Filtering queries with related data

Update and delete related data

Insights into persisting untracked graphs

"Fasten your seat belts, it's gonna be a bumpy night."

Bette Davis in “All About Eve”

"Fasten your seat belts, it's gonna be ~~a bumpy night!"~~

an interesting module

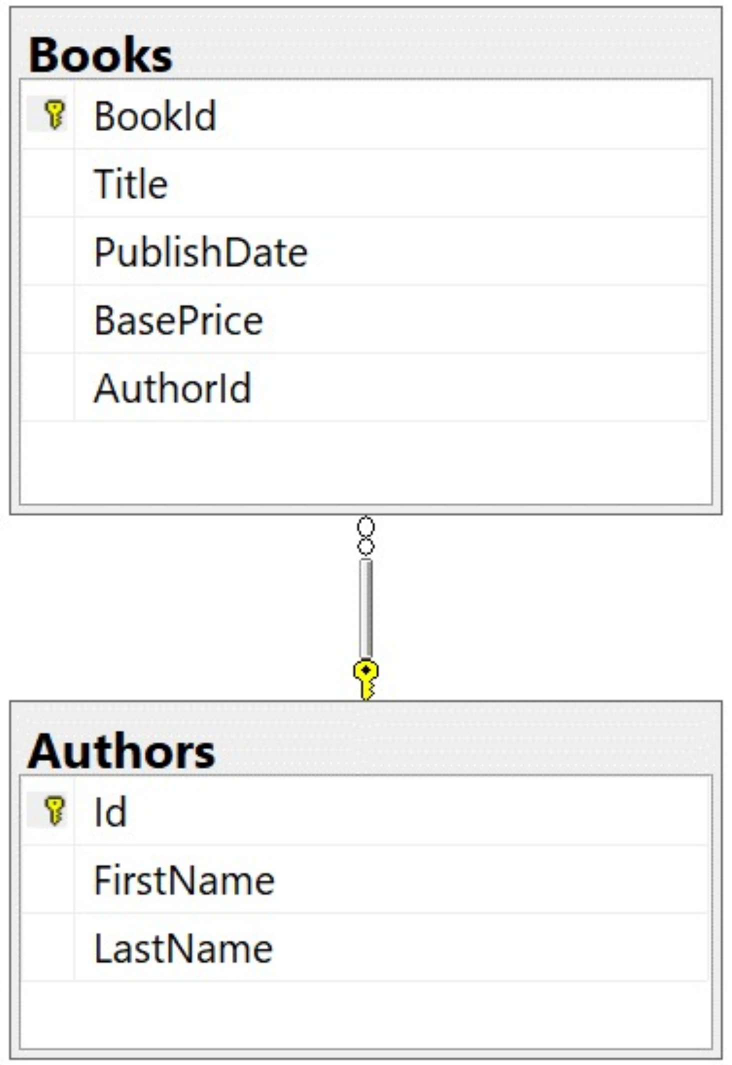
"All About Eve"

Adding Related Data

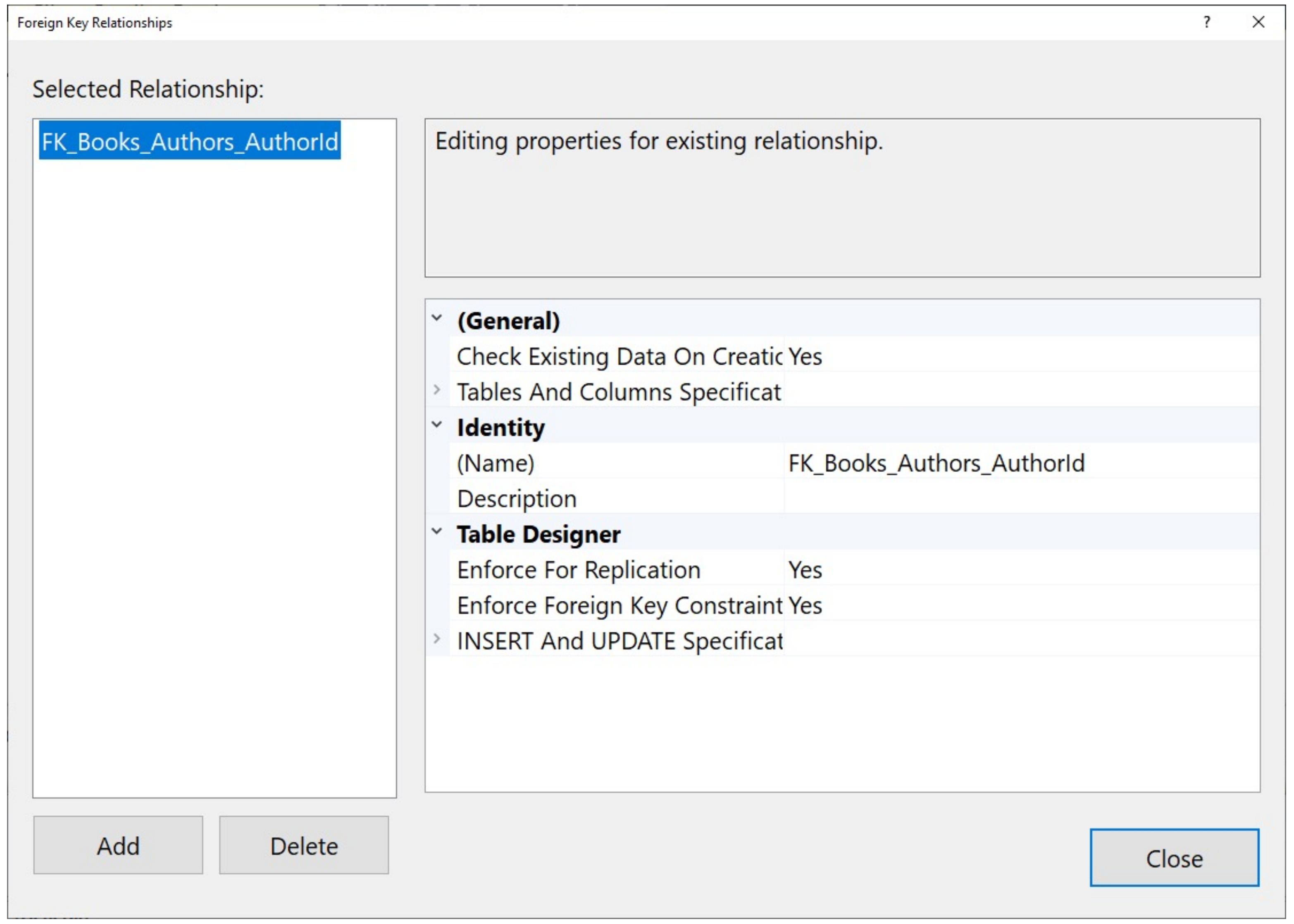
Add New Parent and New Child Together

Let’s add new authors and their new books

One-to-Many in the Database



The relationship


Foreign Key Relationships

Selected Relationship:

FK_Books_Authors_AuthorId

Editing properties for existing relationship.

(General)	
Check Existing Data On Creatic	Yes
Tables And Columns Specificat	
Identity	
(Name)	FK_Books_Authors_AuthorId
Description	
Table Designer	
Enforce For Replication	Yes
Enforce Foreign Key Constraint	Yes
INSERT And UPDATE Specificat	

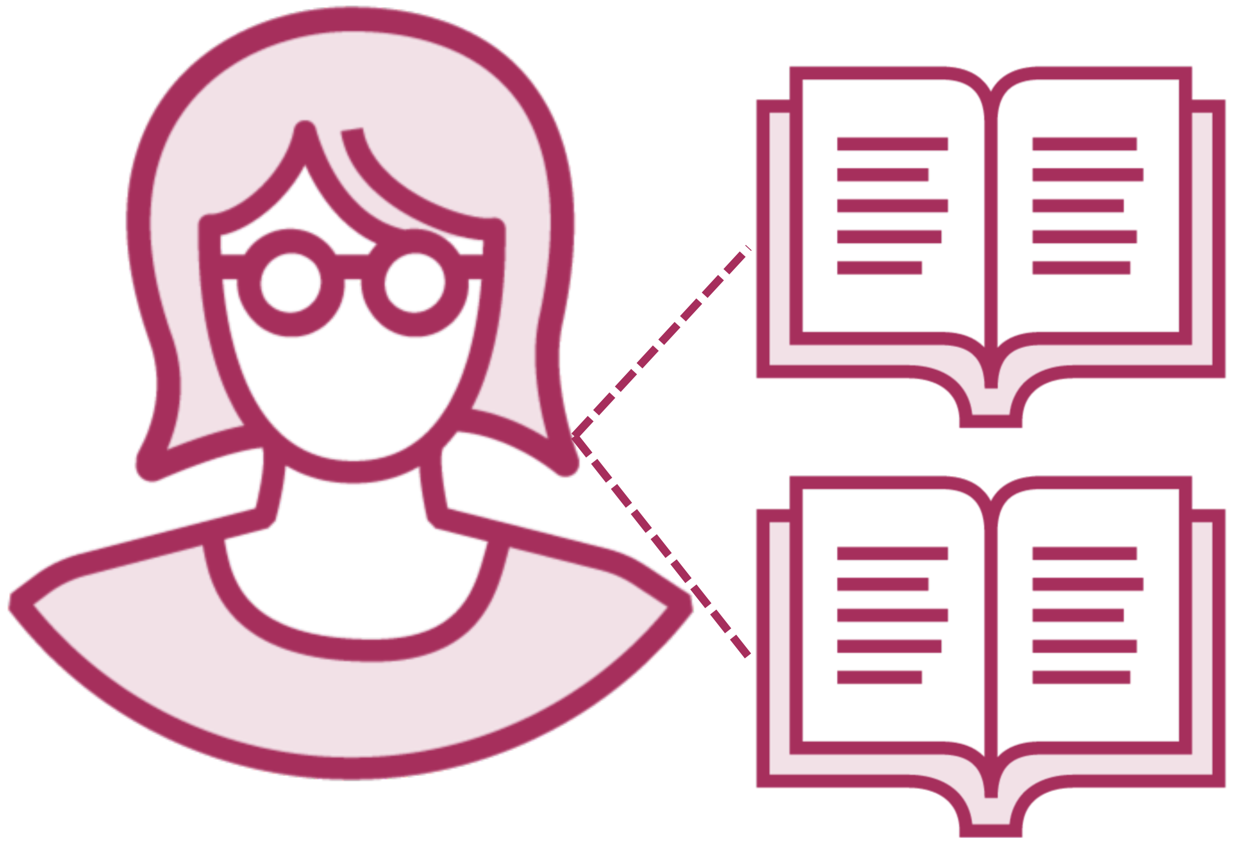
Constraints of this relationship

Views from SQL Server Management Studio

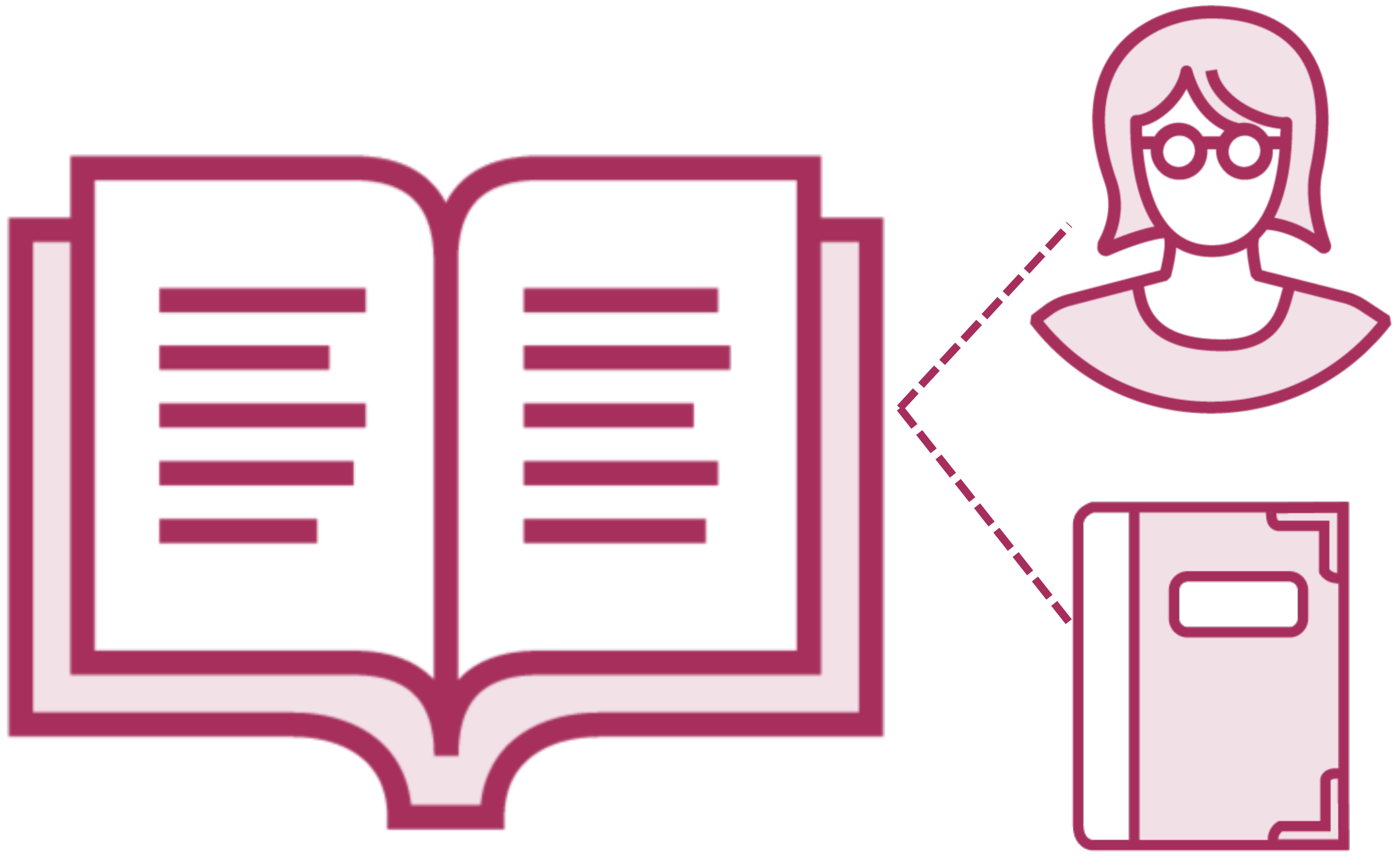
Object Graph

A connected set of related objects

Any Object Can Be the Head of a Graph



Author Graph
An author with some books in memory



Book Graph
A book with an author and its book jacket in memory

Add a New Child to an Existing Parent

Our authors keep writing new books.
Let's get those books into the database

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. Book.AuthorId) is set to parent's key

Reminder: DbContext/DbSet Tracking Methods

Add

Update

Remove

Attach

Change Tracker Response to New Child of Existing Parent

Add child to child collection of existing tracked parent.

SaveChanges

Add existing tracked parent to ref property of child.

SaveChanges

Set foreign key property in child class to parent's key value.

Add & SaveChanges

Beware accidental inserts!

Passing a pre-existing entity into its DbSet Add will cause EF Core to try to insert it into the database!

EF Core's Default Entity State of Graph Data

	Has Key Value	No Key Value
Add(graph)	Added	Added
Update(graph)		
Attach(graph)		

EF Core's Default Entity State of Graph Data

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

*Database will throw an exception unless IDENTITY INSERT is explicitly enabled and the key doesn't already exist

Understand
how your tools work!

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

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

Some of this change tracking behavior is different in disconnected scenarios.

Eager Loading Related Data in Queries

Means to Get Related Data from the Database

Eager Loading

Include related objects in query

Query Projections

Define the shape of query results

Lazy Loading

On-the-fly retrieval of data related to objects in memory

Explicit Loading

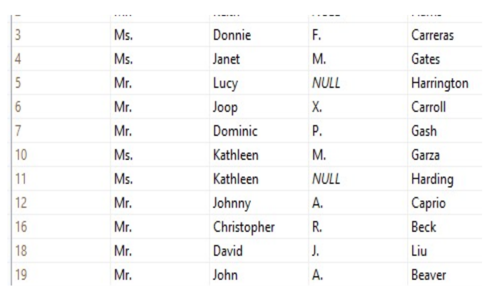
Explicitly request related data for objects in memory

*Arrived with EF Core 2.1

Query Workflow

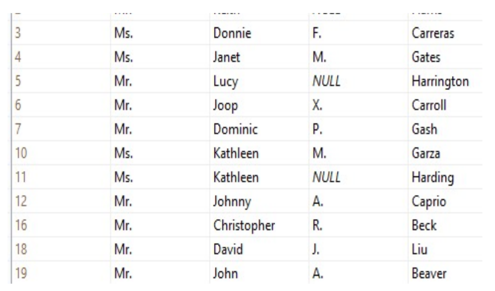
Receives tabular results

Authors



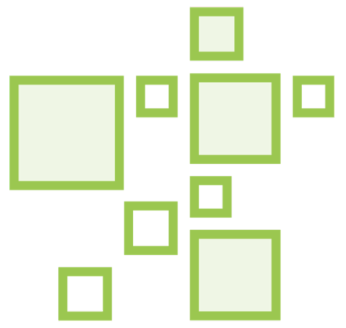
3	Ms.	Donnie	F.	Carreras
4	Ms.	Janet	M.	Gates
5	Mr.	Lucy	*NULL*	Harrington
6	Mr.	Joop	X.	Carroll
7	Mr.	Dominic	P.	Gash
10	Ms.	Kathleen	M.	Garza
11	Ms.	Kathleen	*NULL*	Harding
12	Mr.	Johnny	A.	Caprio
16	Mr.	Christopher	R.	Beck
18	Mr.	David	J.	Liu
19	Mr.	John	A.	Beaver

Books for those Authors



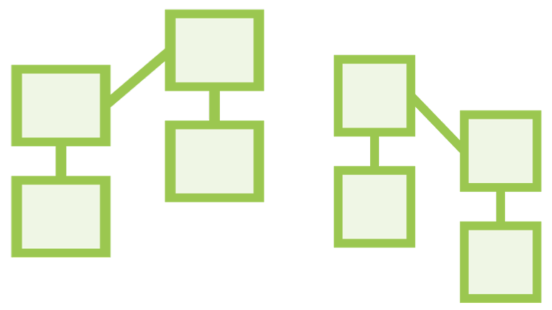
3	Ms.	Donnie	F.	Carreras
4	Ms.	Janet	M.	Gates
5	Mr.	Lucy	*NULL*	Harrington
6	Mr.	Joop	X.	Carroll
7	Mr.	Dominic	P.	Gash
10	Ms.	Kathleen	M.	Garza
11	Ms.	Kathleen	*NULL*	Harding
12	Mr.	Johnny	A.	Caprio
16	Mr.	Christopher	R.	Beck
18	Mr.	David	J.	Liu
19	Mr.	John	A.	Beaver

Materializes results as objects



Adds tracking details to DbContext instance

DbContext connects the relationships



Filtering & Sorting the Included Data

By default, the entire collection is retrieved

You can filter and sort the related data

Long requested feature that finally arrived in EF Core 5!

Composing Include with Other LINQ Methods

👍

```
_context.Authors.Where(a=>a.LastName.StartsWith("L"))
       .Include(a=>a.Books).ToList()
```

👍

```
_context.Authors.Where(a => a.LastName == "Lerman")
       .Include(a => a.Books).FirstOrDefault()
```

👎

```
_context.Authors.Where(a => a.LastName == "Lerman")
.FirstOrDefault().Include(a => a.Books)
```

👎

```
_context.Authors.Find(1).Include(a=>a.Books)
_context.Authors.Include(a=>a.Books).Find(1)
```

Remember, Find is not a LINQ method

Using Include for Multiple Layers of Relationships

```
_context.Authors
.Include(a => a.Books)
.ThenInclude(b=>b.BookJackets)
.ToList();
```

◀ **Get books for each author**

◀ **Then get the jackets for each book**

```
_context.Authors
.Include(a => a.Books)
.Include(a=>a.ContactInfo)
.ToList();
```

◀ **Get books for each author**

◀ **Also get the contact info each author**

```
_context.Authors
 .Include(a=>a.Books.BookJackets)
 .ToList();
```

◀ **Get the jackets for each author's books**

(But don't get the books)

Performance Considerations with Include

Composing many Includes in one query *could* create performance issues. Monitor your queries!

Include defaults to a single SQL command. Use AsSplitQuery() to send multiple SQL commands instead.

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()

Projecting Related Data in Queries

Means to Get Related Data from the Database

Eager Loading

Include related objects in query

Query Projections

Define the shape of query results

Lazy Loading

On-the-fly retrieval of data related to objects in memory

Explicit Loading

Explicitly request related data for objects in memory

```
var someType=_context.Authors
    .Select(properties into a new type)
    .ToList()

var someType=_context.Authors
    .Select(a=>new {a.FirstName, a.LastName, a.Books.Count() }
    .ToList()

someType structure:
    FirstName
    LastName
    Books
```

Projecting into Undefined ("Anonymous") Types

Use LINQ's Select method
Use a lambda expression to specify properties to retrieve
Instantiate a type to capture the resulting structure
Anonymous types are not available outside of the method

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

Means to Get Related Data from the Database

Eager Loading

Include related objects in query

Query Projections

Define the shape of query results

Loading related data for objects already in memory

Means to Get Related Data from the Database

Eager Loading

Include related objects in query

Query Projections

Define the shape of query results

Explicit Loading

Explicitly request related data for objects in memory

Lazy Loading

On-the-fly retrieval of data related to objects in memory

```
With author object already in memory, load a collection

_context.Entry(author).Collection(a => a.Books).Load();


With book object already in memory, load a reference (e.g., parent or 1:1)

_context.Entry(book).Reference(b => b.Author).Load();
```

Explicit Loading

Explicitly retrieve related data for objects already in memory
DbContext.Entry(object).Collection().Load()
DbContext.Entry(object).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();
```

More on Explicit Loading

You can only load from a single object

Profile to determine if LINQ query would be better performance

Filter on loading using Query() method

```
var newfBooks =
  context.Entry(author)
    .Collection(a => a.Books)
    .Query().Where(b =>
       b.Title.Contains(“Newf”)
    .ToList();
```

Using Lazy Loading to Retrieve Related Data

Means to Get Related Data from the Database

Eager Loading

Include related objects in query

Query Projections

Define the shape of query results

Explicit Loading

Explicitly request related data for objects in memory

Lazy Loading

On-the-fly retrieval of data related to objects in memory

Lazy loading is easy to misuse!

I recommend some advanced learning before using it.

Lazy Loading is OFF by default

Enabling Lazy Loading

Every navigation property in every entity must be virtual

```
e.g., public virtual List<Book> Books { get; set; }
```

Reference the Microsoft.EntityFramework.Proxies package

Use the proxy logic provided by that package

```
optionsBuilder.UseLazyLoadingProxies()
```

Some Good and Not So Good Ways to Use Lazy Loading

Good Behavior

```
foreach(var b in author.Books)
{
  Console.WriteLine(b.Title);
}
```

◀ **One command to the db to get the books for this one author**

Behavior to Avoid

```
var bookCount= author.Books.Count();
```

◀ **Retrieves all the Book objects from the database and materialize them and *then* give you the count.**

```
Data bind a grid to lazy-loaded data
```

◀ **Sends N+1 commands to the database as each author's book is loaded into a grid row**

```
Lazy loading when no context in scope
```

◀ **No data is retrieved**

Using Related Data to Filter Objects

Modifying Related Data

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)		

Connected

DbContext is aware
of all changes made to objects
that it is tracking
(when DetectChanges is called)

Disconnected

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

Reminder: DbContext/DbSet Tracking Methods

Add

Update

Remove

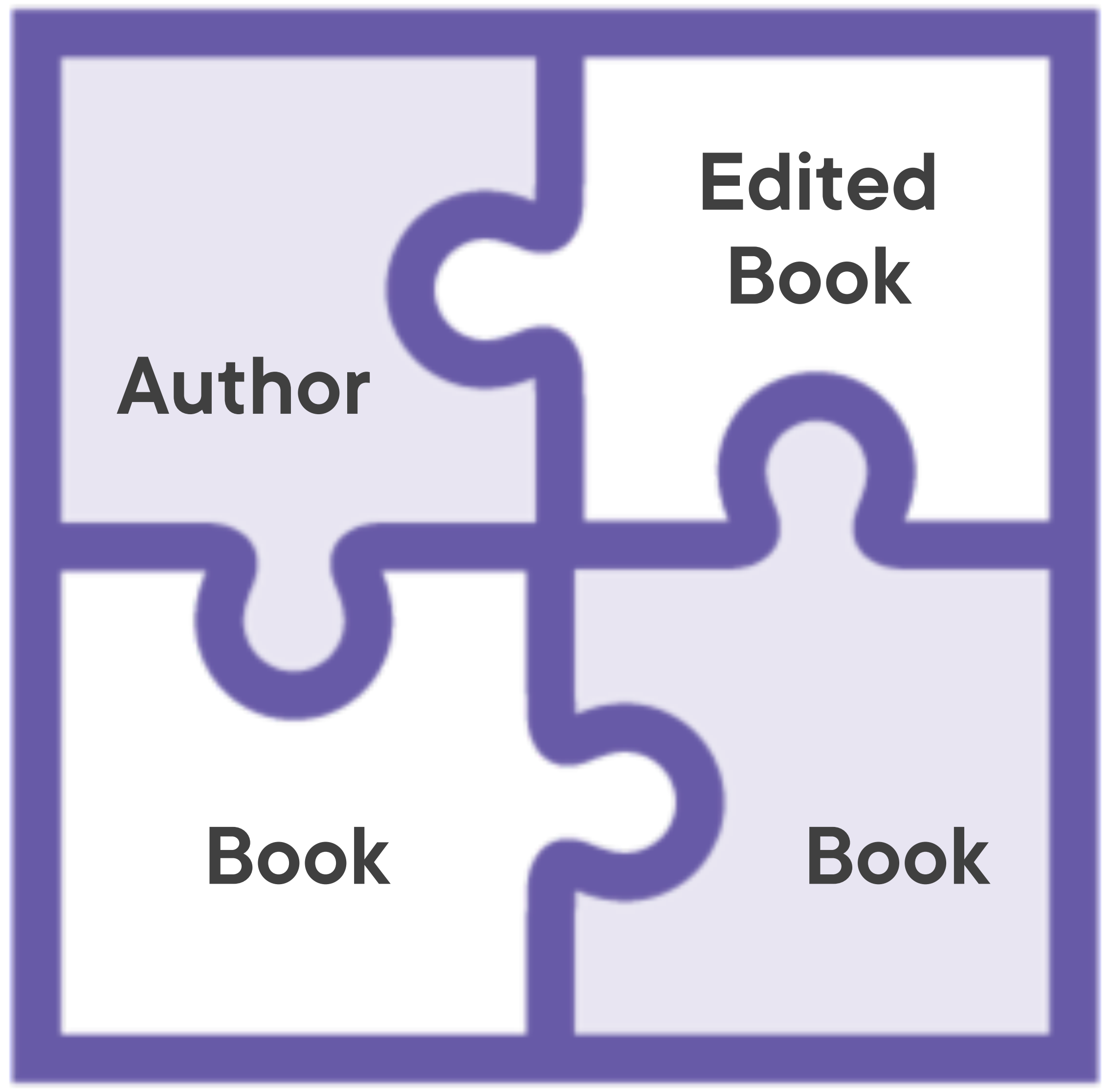
Attach

Attach starts tracking with state set to Unchanged

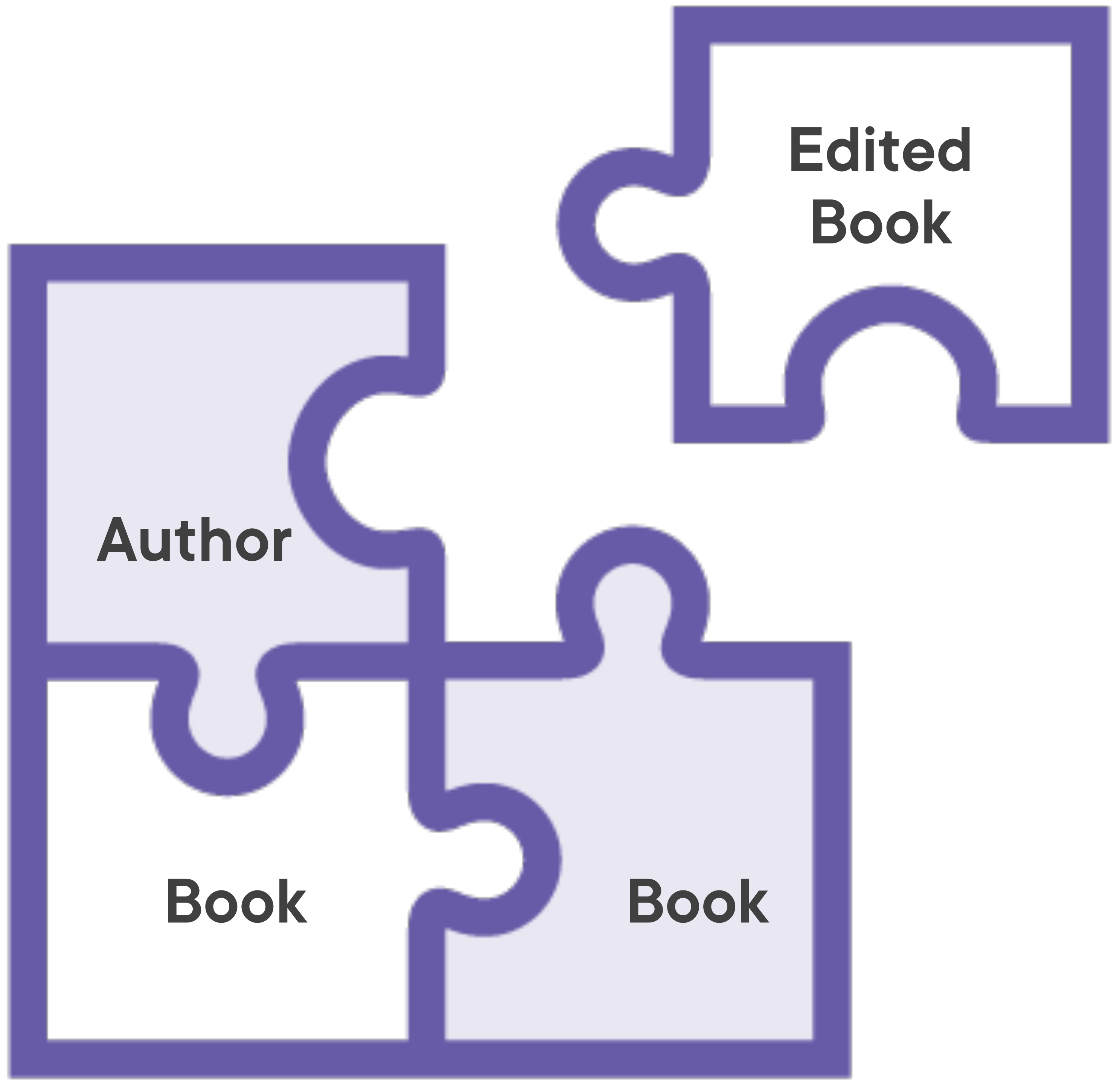
EF Core's Default Entity State of Graph Data

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

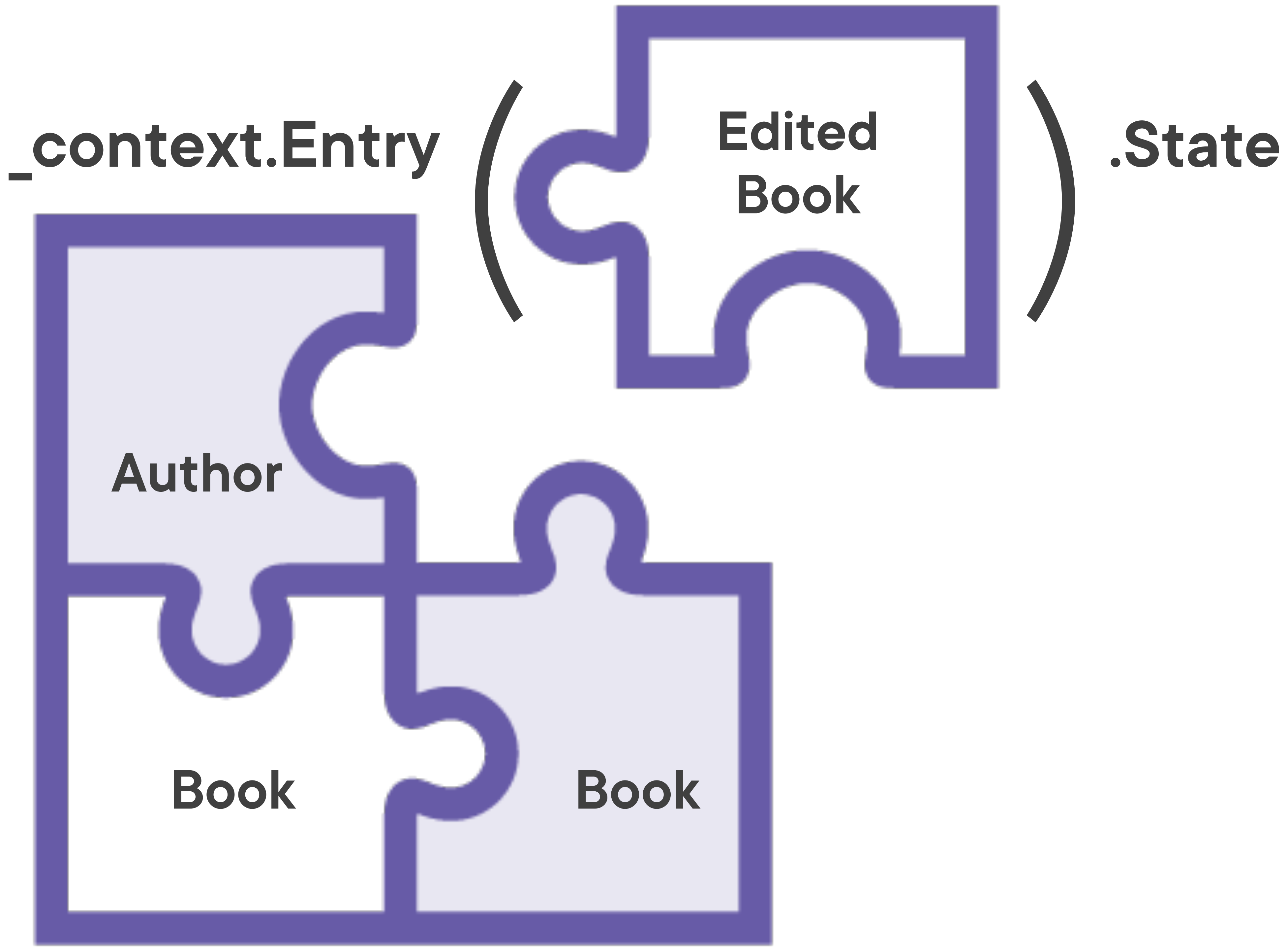
The Challenge



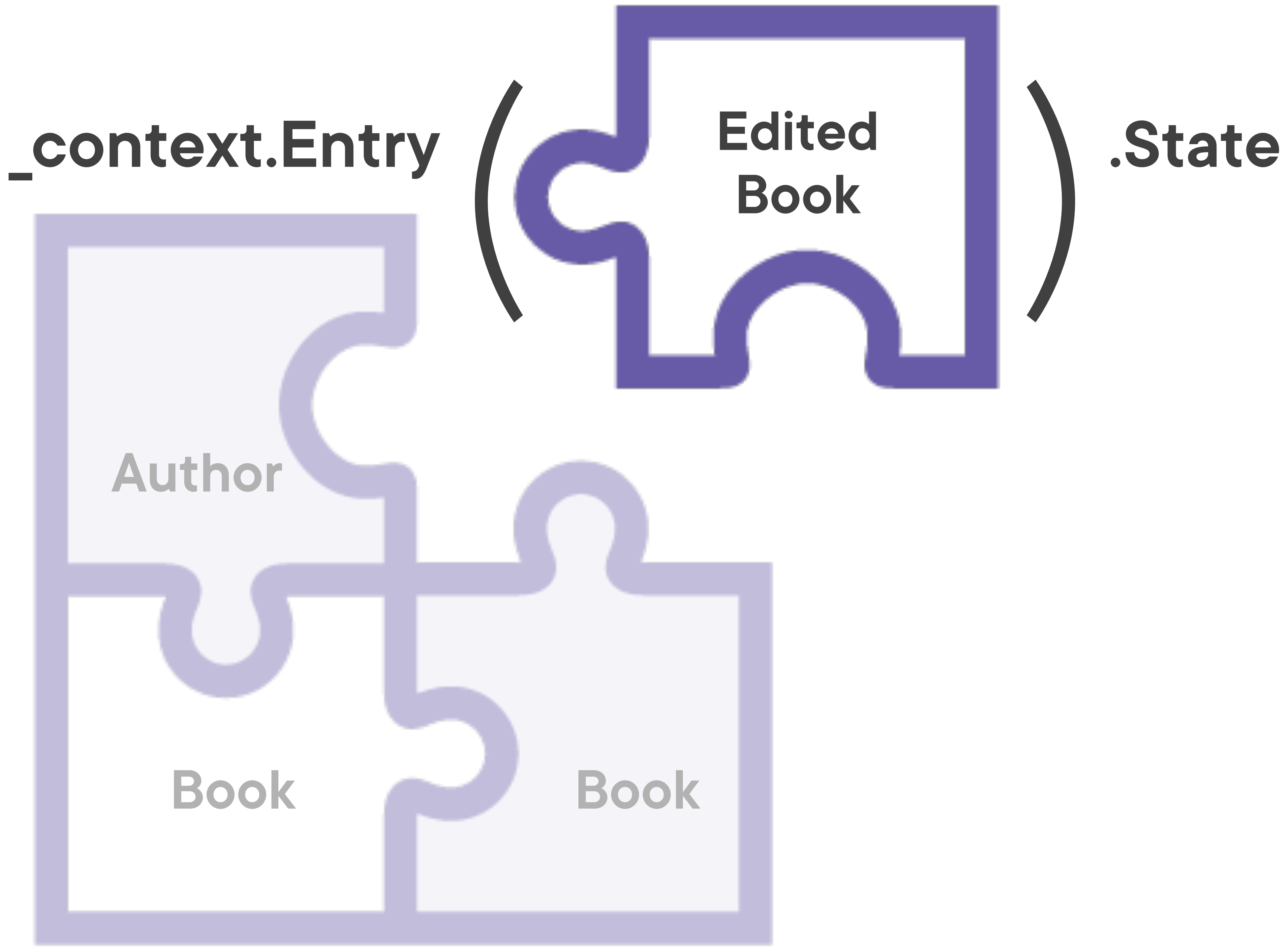
The Challenge



The Challenge



The Challenge



DbContext.Entry gives you a lot of fine-grained control over the change tracker.

Understanding Deleting Within Graphs

Multiple Meanings of Remove/Delete

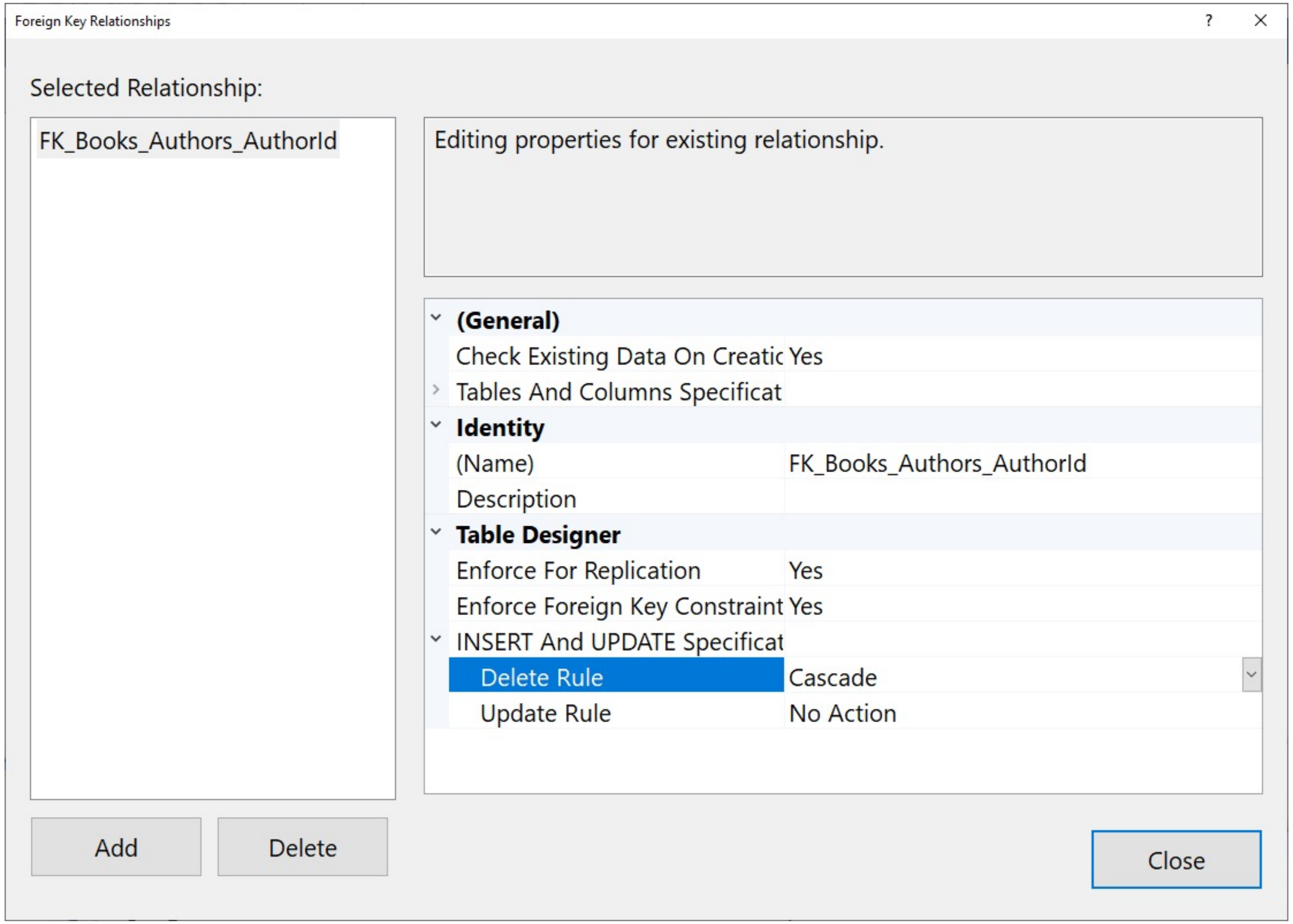
Remove from an in-memory collection

Set State to Deleted in Change Tracker

Delete from database

Cascade Delete When Dependents Can’t Be “Orphaned”

Foreign Key Relationship Constraint in SQL Server



Foreign Key Relationships

Selected Relationship:

FK_Books_Authors_AuthorId

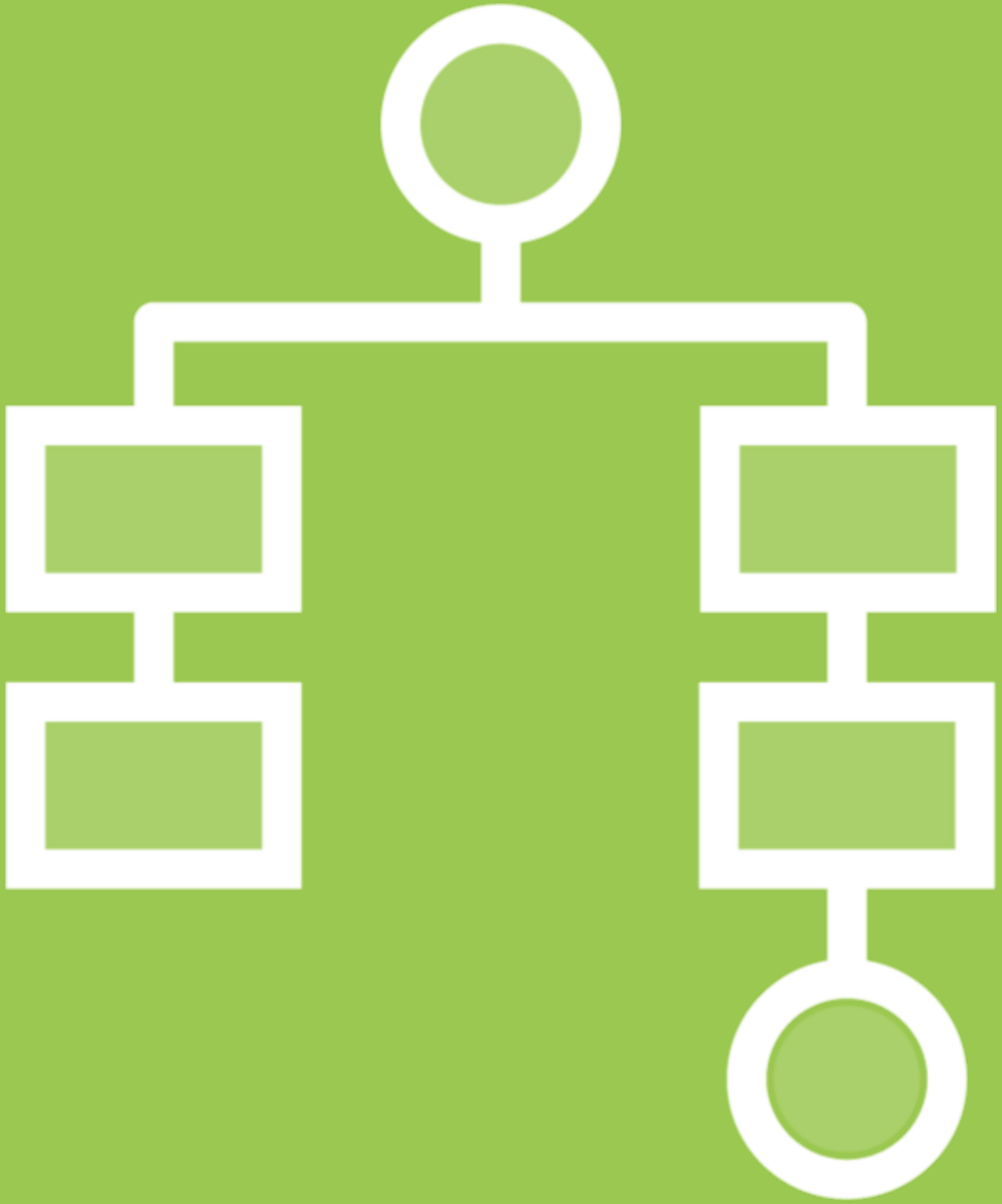
Editing properties for existing relationship.

(General)	
Check Existing Data On Creatic	Yes
Tables And Columns Specificat	
Identity	
(Name)	FK_Books_Authors_AuthorId
Description	
Table Designer	
Enforce For Replication	Yes
Enforce Foreign Key Constraint	Yes
INSERT And UPDATE Specificat	
Delete Rule	Cascade
Update Rule	No Action

```
void DeleteAnAuthor()
{
    var author = _context.Authors.Find(2);
    _context.Authors.Remove(author);
    _context.SaveChanges();
}
```

Database Enforces Cascade Delete

Only author is in memory, tracked by EF Core & marked “Deleted”
Database’s cascade delete will take care of books when author is deleted



EF Core Enforces Cascade Deletes

Any related data that is also tracked will be marked as Deleted along with the principal object

It’s running the
DELETE FROM Books
while the books are still in the
database.

```
var author=_context.Authors
 .Include(a=>a.Books
             .Where(b=>b.Title="XYZ")
 .FirstOrDefault();
```

◂ Retrieve an author with only *some* of their books

```
_context.Authors.Remove(author);
```

◂ Mark the author as deleted

```
//Entry(author).State=Deleted
//Entry(thatbook).State=Deleted
```

◂ Change tracker will also mark that book as deleted

```
DELETE thatbook FROM BOOKS

DELETE thatauthor FROM AUTHORS
```

◂ SaveChanges sends DELETE for that book and DELETE for the author to database

```
//database cascade delete any other
books
```

◂ The database will delete any remaining books in that database for the author

A Few Last Questions about Cascade Delete

Question	Answer
Will Remove() remove everything in a graph, just like Update?	No. Remove will only remove the specific object.
What about deleting a dependent that's not in a graph?	You've actually done this! Just call it's DbSet Remove method.

Some More Points About Removing Dependents

What about ....

How to move a child from one parent to another (when tracked)?

Examples

- `book.AuthorId=3`
- `newAuthor.Books.Add(book)`
- `book.Author=newAuthor`

What about optional relationships?

- `book.AuthorId=null`
- `author.Books.Remove(book)`

What about deleting relationships in disconnected scenarios?

This is more advanced, but you will see some of it in the web app module.

Review

You can eager load related data with Include() or projections.

Lazy and explicit loading let you load after the fact.

Pay attention to lazy loading behavior.

Filter the related data or use it to filter the base data.

Adding, modifying or deleting data in graphs has varying impacts on the related objects.

DbContext.Entry() isolates and affects only the object you pass in.

Up Next:
Defining and Using Many-to-Many Relationships

Resources

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

EF Core Documentation docs.microsoft.com/ef

Lazy Loading With and Without Proxies (Module from EF Core 2.1 What's New) bit.ly/EFCoreLazy