Defining and Using Many-to-Many Relationships

Julie Lerman
Most Trusted Authority on Entity Framework Core

@JulieLerman   www.thedatafarm.com

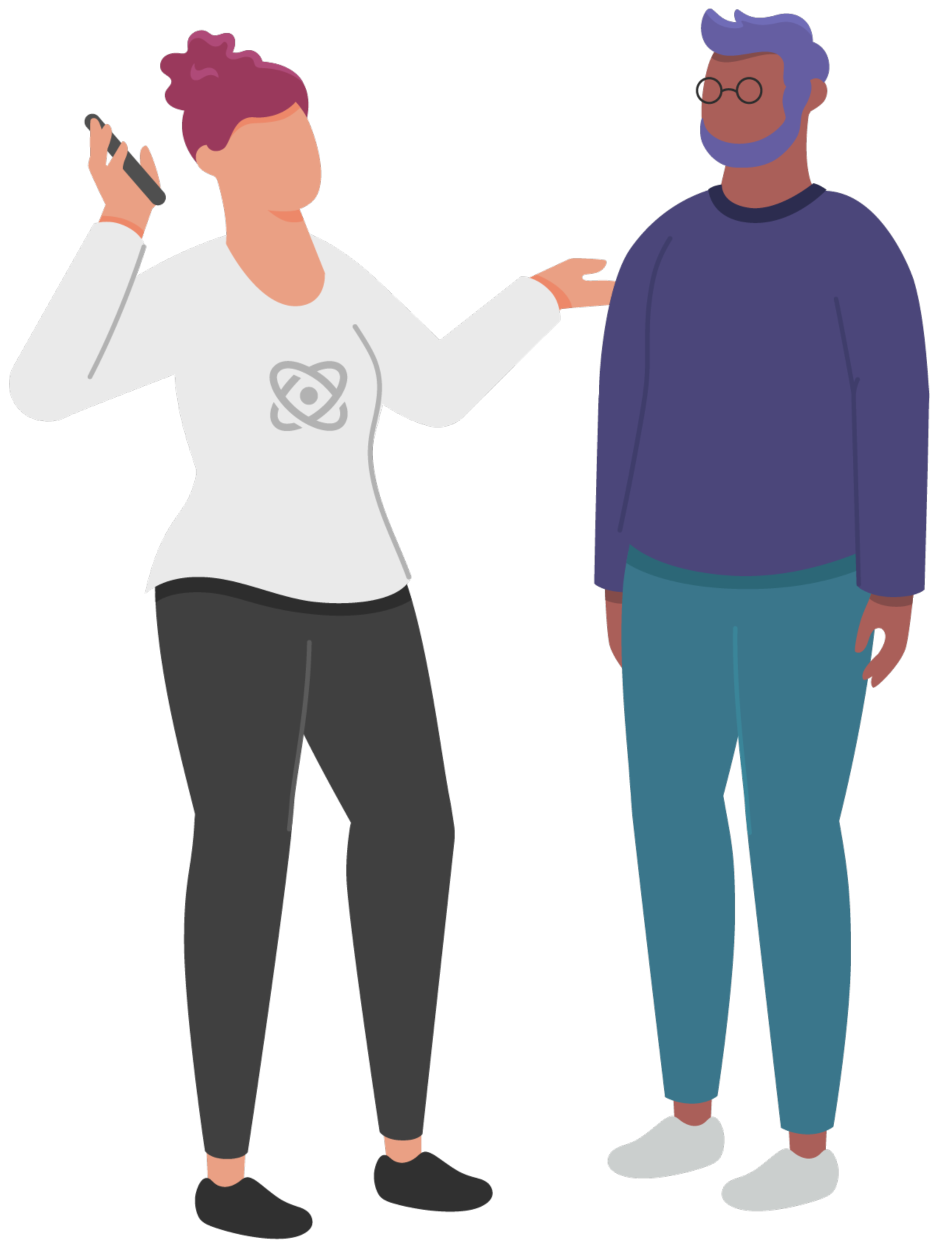
Dear Programmers,

We would like to keep track of the artists who design book covers.

Note that the artists sometimes collaborate on a cover.

Thanks a bunch!

Your friendly editors



This calls for a many-to-many relationship!

Overview

EF Core's options for many-to-many

Most common: skip navigations

Querying across many-to-many relationships

Add & remove joins between objects

Dig into circular references in graphs

Quick look at more complex M2M mappings

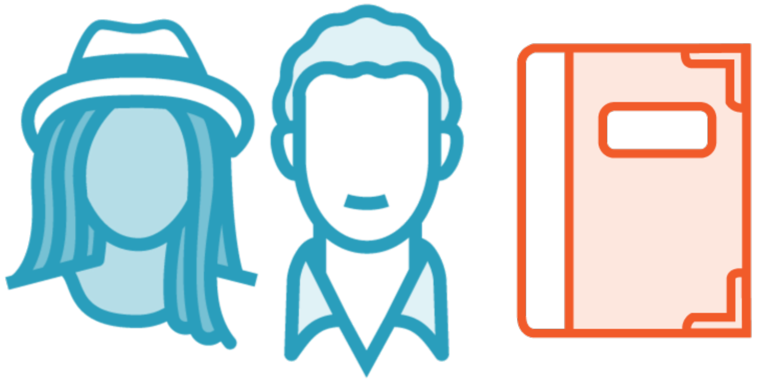
Planning the Many-to-Many Implementation

Artists from our “Pool” Design Our Book Covers

The Artist Pool

Book Covers

Artists Can Collaborate on Cover Designs



Multiple artists working on a cover

An artist can work on many covers

Coming from old EF Core or EF 6?

EF Core 5 brought back “skip navigations” which are much smarter and more flexible than EF6

Three Ways to Define Many-to-Many

Skip Navigations
Most common
Direct refs from both ends

Skip with Payload
Allows database-generated data in extra columns

Explicit Join Class
Additional properties accessible via code

Changes Needed for Our Simple App

Create new Artist and Cover classes, update PubContext

Create a migration to reflect the changes needed in the database

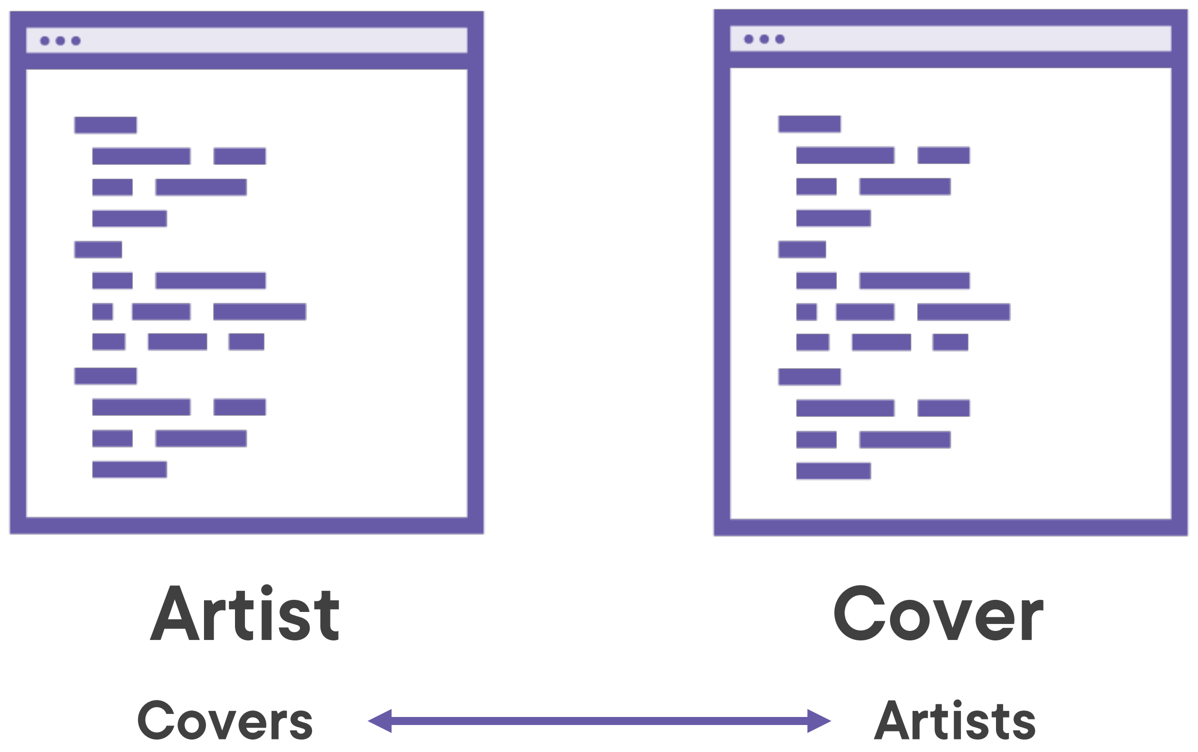
Apply the migration to the database

Write our code to manage artists and book covers

In the future (next module) we will connect books to covers

Understanding and Creating Skip Navigations

Many-to-Many with “Skip Navigations”



EF Core Can Interpret Skip Navigations

Join the Ends with Class Properties

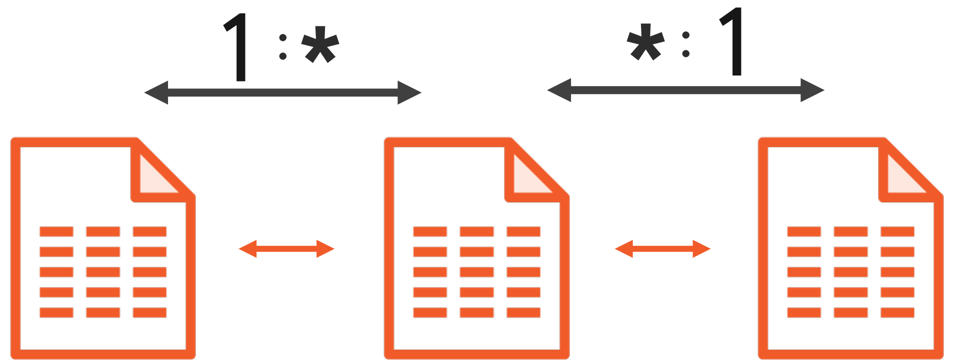
Artist

List<Cover>

Cover

List<Artist>

Relational Database: Join Table



Artists

ArtistId

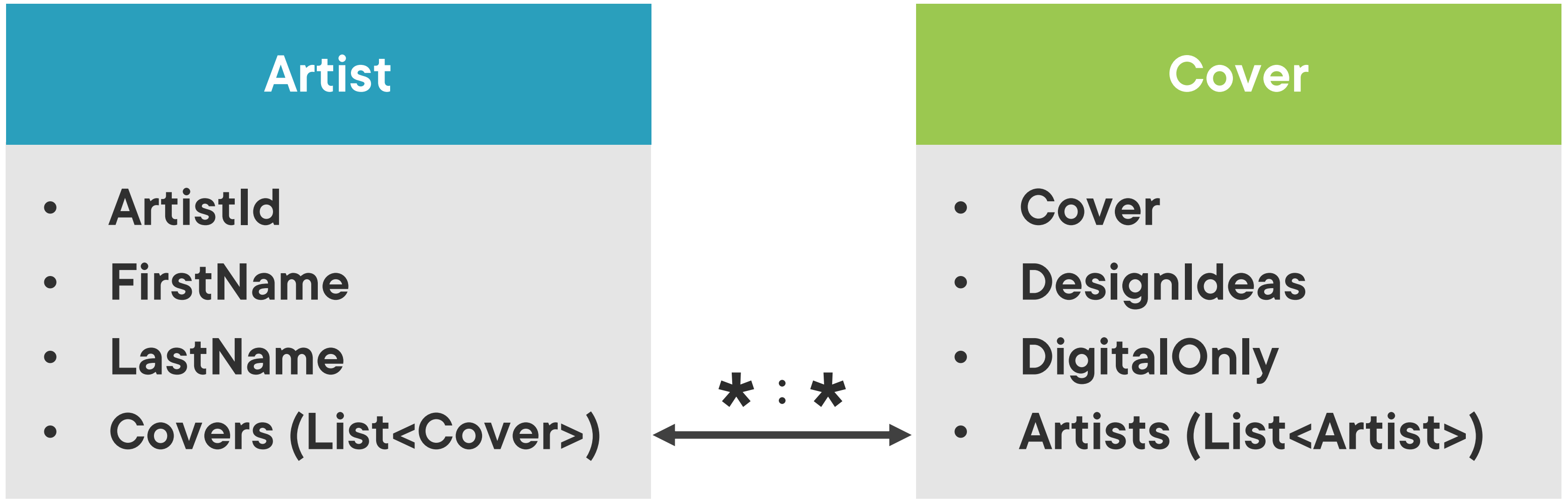
CoverArtist

ArtistId
CoverId

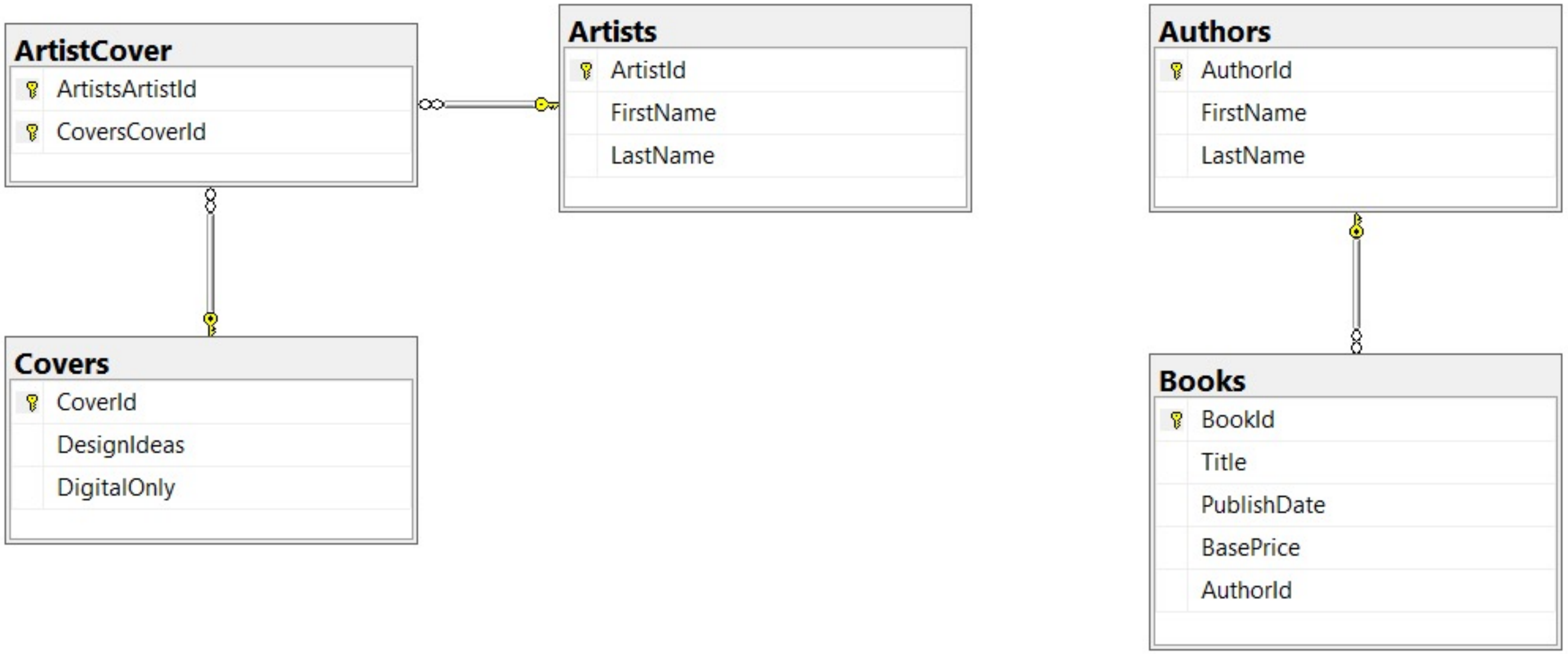
Covers

CoverId

Easy to Code Against Skip Navigations



SSMS Diagram of PubDatabase Tables



Joining Objects in New Many-to-Many Relationships

Joining Covers and Artists of Differing States

Existing Cover
+
Existing Artist

Existing artist is assigned to a pre-defined book Cover

New Cover
+
Existing Artist

New artist is hired to work on a pre-defined book Cover

New Cover
+
New Artist

New artist is hired and declares a new book Cover

Skip Navigations Require Objects

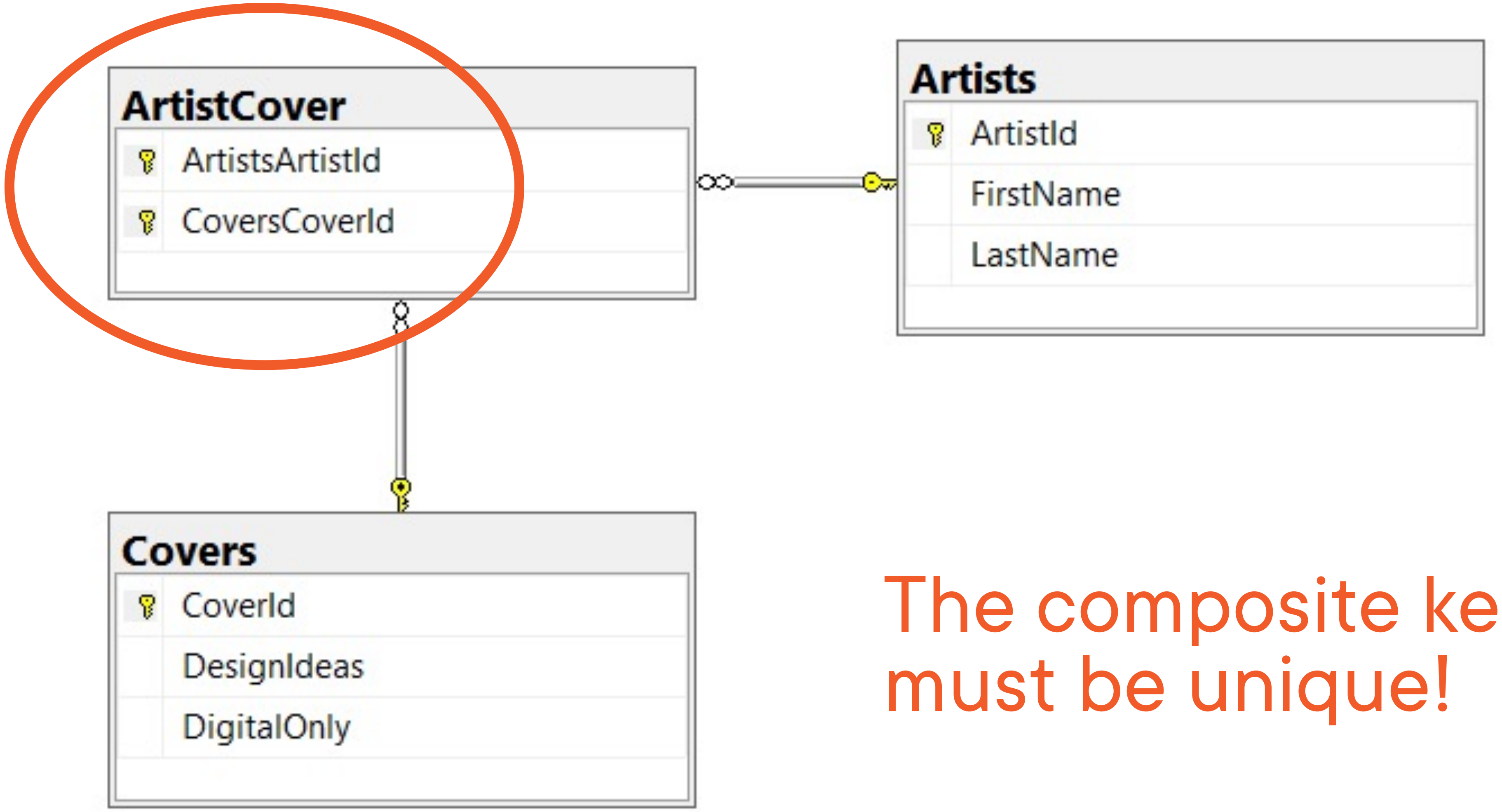
```
public class Book
{
  ...other properties
  public int AuthorId {get;set;}
}
```

One-to-Many with FK Property

```
public class Artist
{  ...other properties
  public List<Cover> Covers {get;set;}
}
public class Cover
{  ...other properties
  public List<Artist> Artists {get;set;}
}
```

Many-to-Many with Skip Navigations

Join Table Has a “Composite” Primary Key



The composite key must be unique!

Querying Across Many-to-Many Relationships

Means to Get Related Data from the Database
Same patterns for many-to-many

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

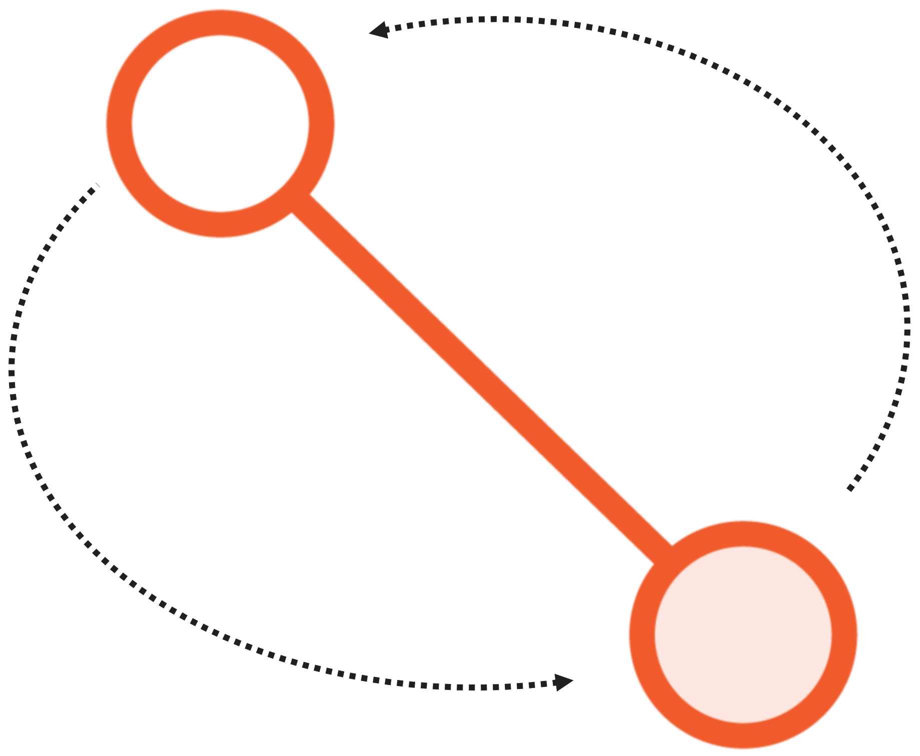
Lazy Loading

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

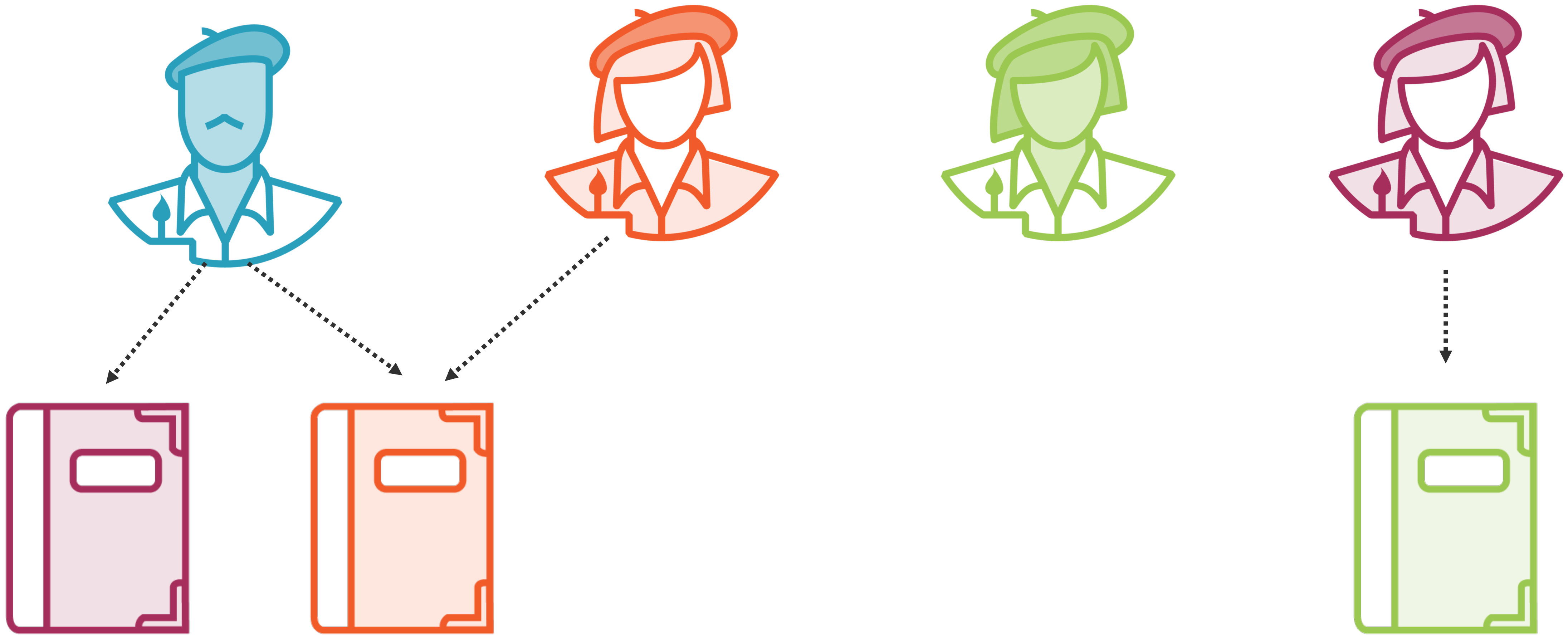
Understanding and Benefiting From Circular References in Graphs

This happens with all relationships,
not just many-to-many
not just EF Core
not just .NET

We Keep Pointing to the Same Objects



We Still Only Have 4 Artists Working on 3 Covers

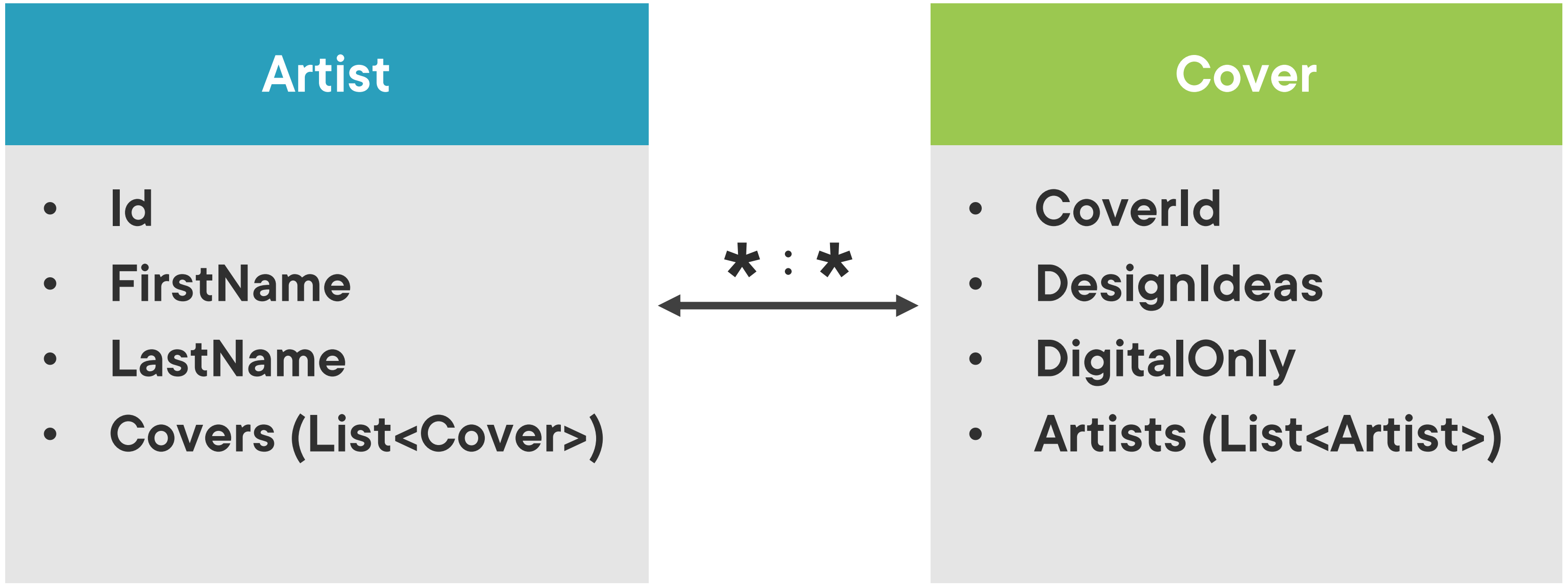


Serializers* can’t handle recursive graphs!

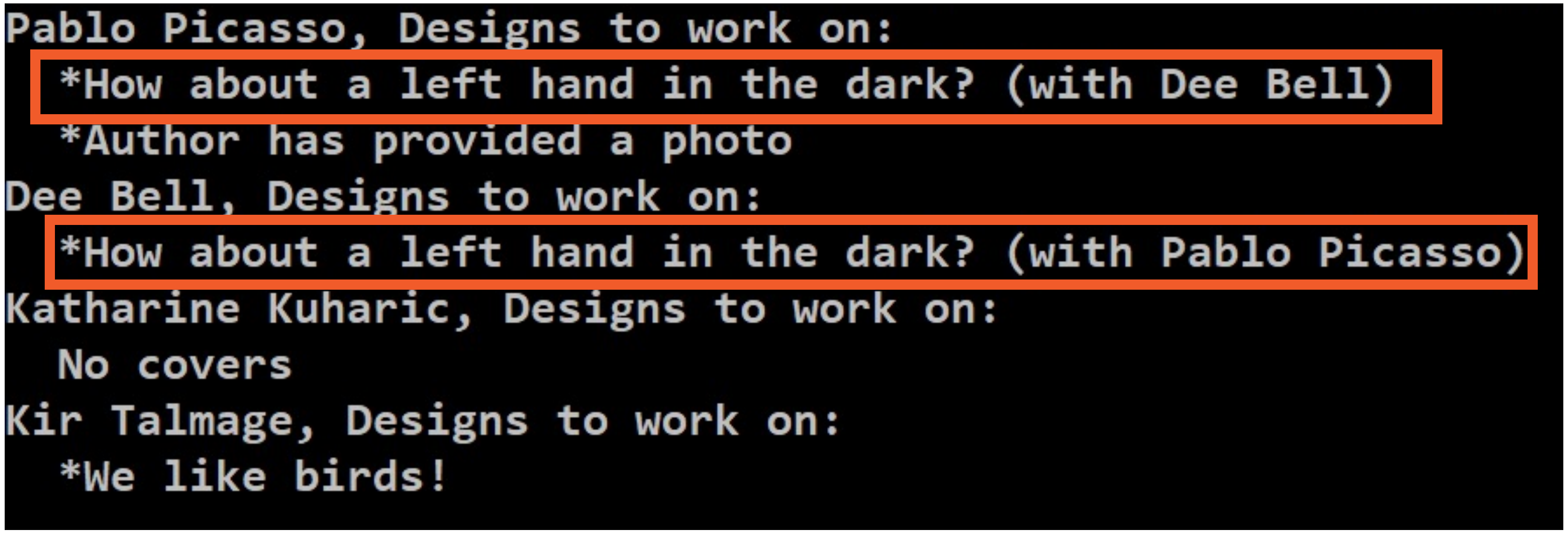
**New term? We’ll look at this later.*

Removing Joins in Many-to-Many Relationships

We Don’t Have Direct Access to the Join



The “Left Hand” Cover Has Two Artists Assigned



```
Pablo Picasso, Designs to work on:
  *How about a left hand in the dark? (with Dee Bell)
  *Author has provided a photo
Dee Bell, Designs to work on:
  *How about a left hand in the dark? (with Pablo Picasso)
Katharine Kuharic, Designs to work on:
  No covers
Kir Talmage, Designs to work on:
  *We like birds!
```

```
void DeleteAnObjectThatsInARelationship()
{
    var cover = _context.Covers.Find(4);
    _context.Covers.Remove(cover);
    _context.SaveChanges();
}
```

What If … You Deleted an Object That Is Joined to Another?

Cascade delete to the rescue!
If the join is being tracked, EF Core will cascade delete the join.
If the relationship is not being tracked, database cascade delete will remove the join.

Deleting a Many-to-Many relationship is easier with stored procedures!

Changing Joins in Many-to-Many Relationships

Reassigning a Cover to a Different Artist

Cover art originally assigned to Kir

Reassigned to Katharine

Changing a Join: Preferred Workflow

Remove the original join

Then create the new join

Be mindful of side effects from your business logic

Reassigning Relationships

In One-to-Many, EF Core knows the dependent can have only one principal

In Many-to-Many, an object can be joined to unlimited partner ends

Introducing More Complex Many-to-Many Relationships

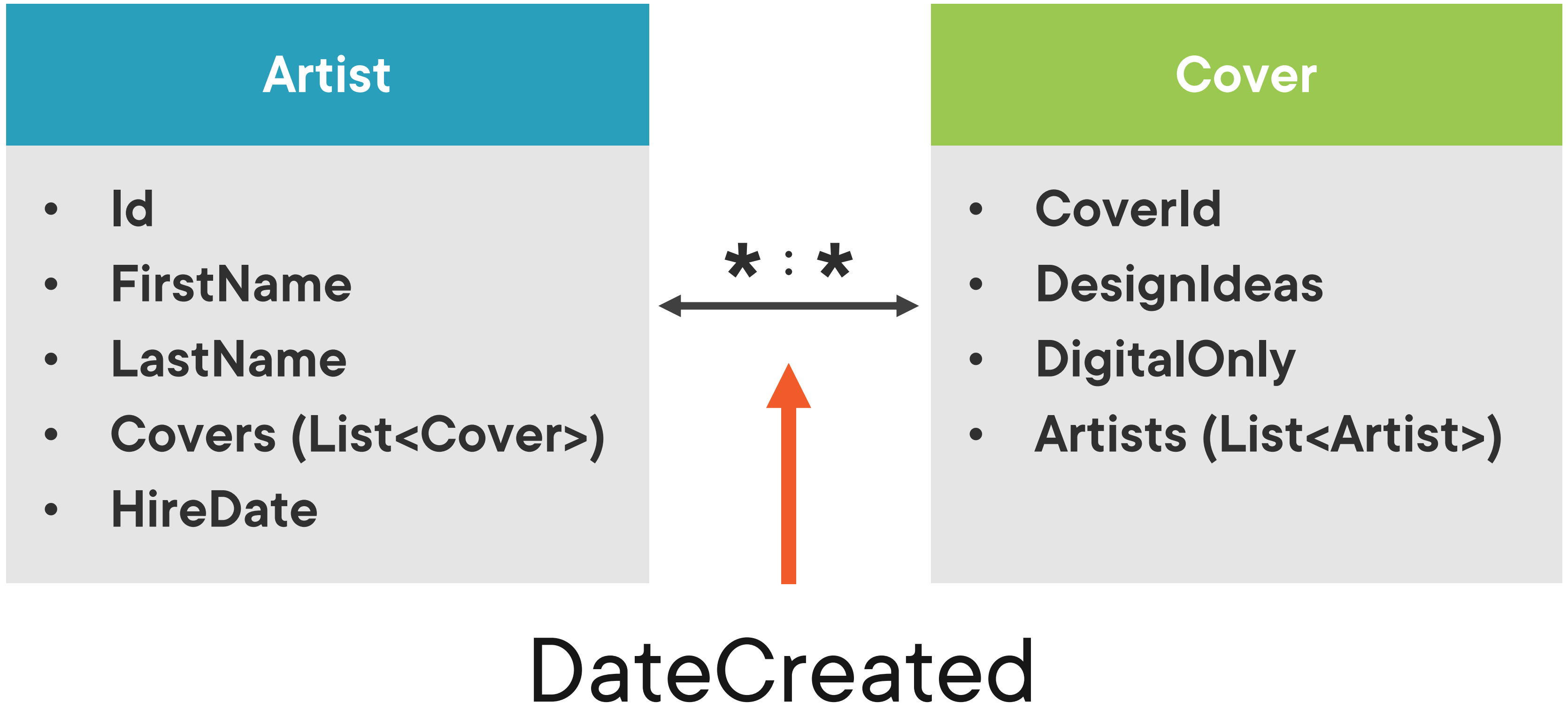
Three Ways to Define Many-to-Many

Skip Navigations
Most common
Direct refs from both ends

Skip with Payload
Allows database-generated data in extra columns

Explicit Join Class
Additional properties accessible via code

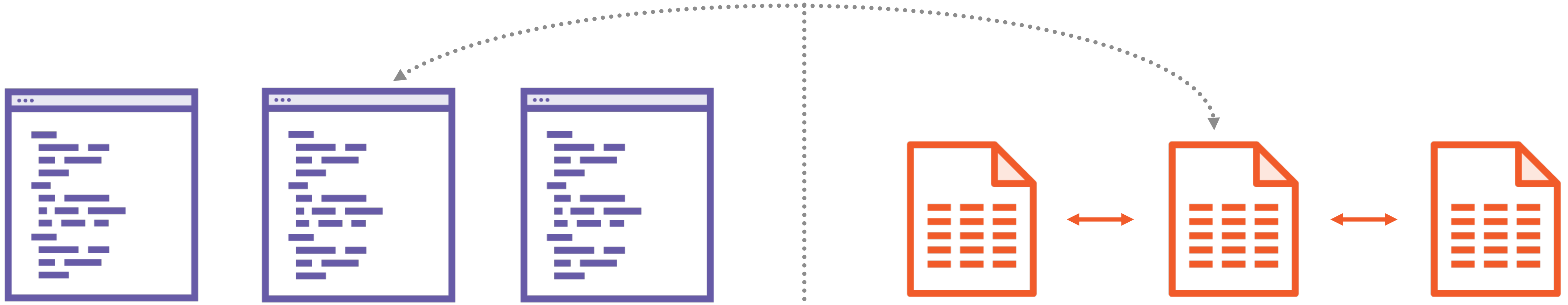
Skip Navigation with a Bit More Data!



Skip Navigation with DB Generated “Payload”

“Secret” Join Entity with Payload

Relational Database: Join Table



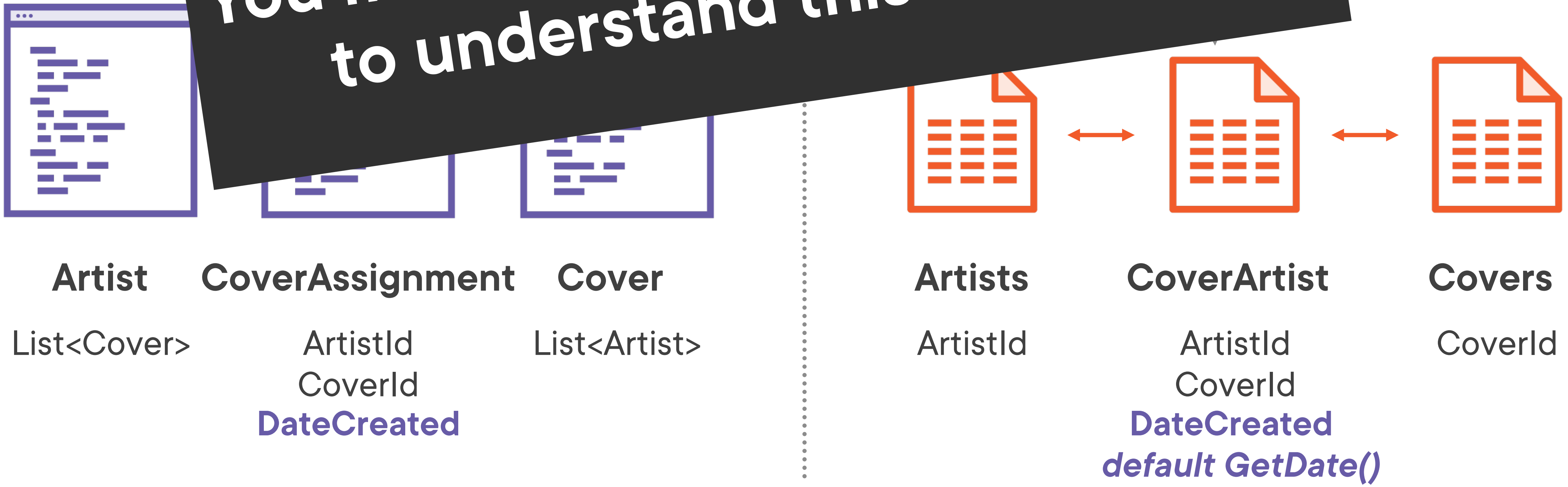
Artist	**CoverAssignment**	**Cover**	**Artists**	**CoverArtist**	**Covers**
List<Cover>	ArtistId CoverId **DateCreated**	List<Artist>	ArtistId	ArtistId CoverId **DateCreated** ***default GetDate()***	CoverId

Skip Navigation with DB Generated “Payload”

“Secret” Join

Join Table

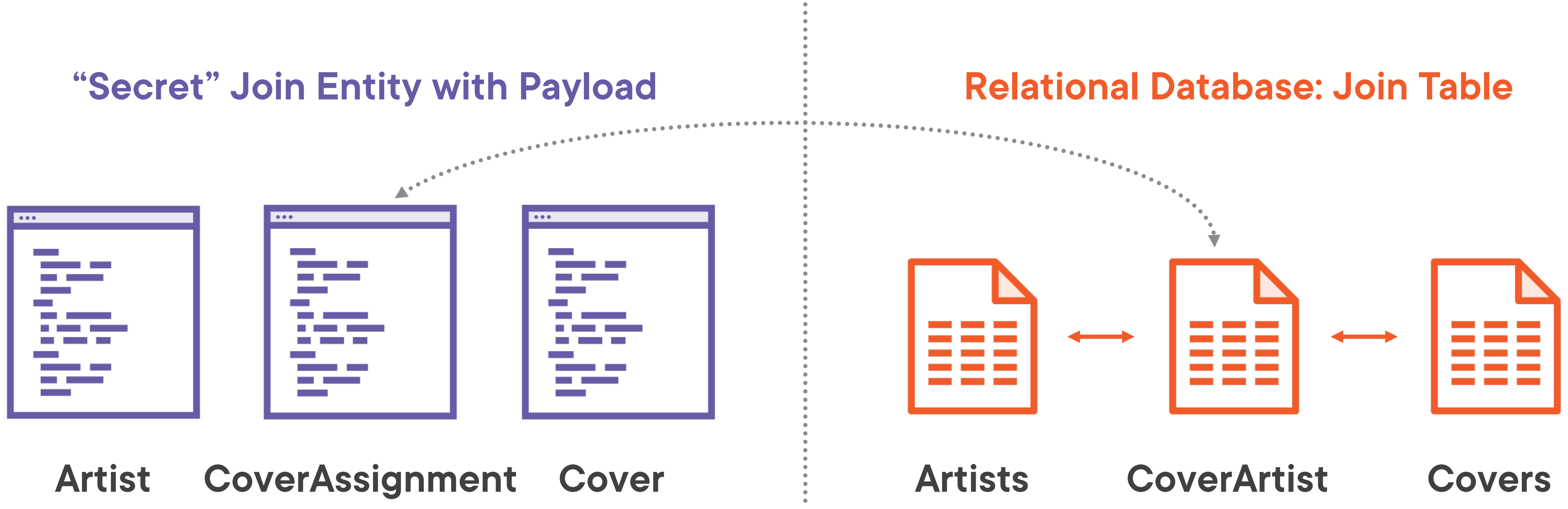
You must configure the DbContext to understand this mapping



Artist	CoverAssignment	Cover	Artists	CoverArtist	Covers
List<Cover>	ArtistId CoverId DateCreated	List<Artist>	ArtistId	ArtistId CoverId DateCreated *default GetDate()*	CoverId

Skip Navigation with DB Generated “Payload”

You must configure the DbContext to understand this mapping


“Secret” Join Entity with Payload
Relational Database: Join Table
Artist
CoverAssignment
Cover
Artists
CoverArtist
Covers

Your Code Can Ignore the Join Entity

Migration would update the schema of ArtistCover table

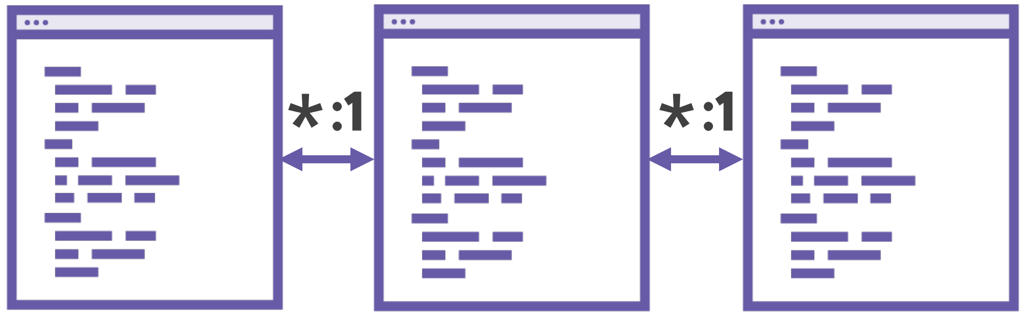
Just keep using the skip navigations in code

Data model is aware of the join entity

The database will take care of updating DateCreated

Many-to-Many Relationship with Join Entity

Explicit Join Entity
[A pair of one-to-many relationships!]



Artist List<Commissions>

CoverAssignment ArtistId CoverId DateContracted Payment

Cover List<Commissions>

Relational Database: Join Table

Artists ArtistId

CoverAssignment ArtistId CoverId RequestDate ContractDate Payment

Covers CoverId

Dual One-To-Many Is Harder

Artist

- ArtistId
- FirstName
- LastName
- Assignments(List<>)

CoverAssignment

- ArtistId
- CoverId
- Artists (List<>)
- Covers (List<>)
- DateContracted
- Payment

Cover

- CoverId
- DesignIdeas
- Completed
- Assignments (List<>)

One Artist to
Many Assignments

Many Commissions
to One Cover

Review

Skip navigations represent how we often think of many-to-many

They are the easiest to configure and query

Removing joins between skip navigations is simplest with stored procedures

Skip navigations with payload lets you store more data

Explicit join entities give you more control but can be more complicated to interact with

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

Resources

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

EF Core Many-to-Many Documentation: bit.ly/M2MDocs

EF Core 2: Mappings (Pluralsight course includes M2M with explicit join): bit.ly/2LppcMj

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

Resources, Continued

Follow FK Indexing Issue on GitHub:
https://github.com/dotnet/efcore/issues/27445

Foreign Key Indexing Guidance
sqlskills.com/blogs/kimberly/sqlskills-sql101-indexes-foreign-keys