Tracking and Saving Data with EF Core

Julie Lerman

EF Core Expert and Software Coach

@julielerman | thedatafarm.com

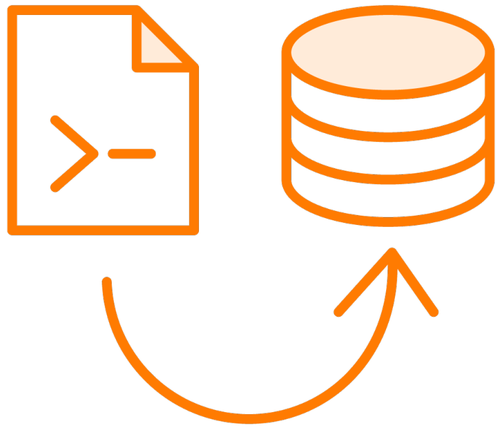
Module Overview

- DbContext and entity roles in tracking
- Tracking and saving workflow
- Inserting, updating, and deleting objects
- Tracking and saving multiple objects
- Bulk operation support
- How persistence differs in disconnected apps

Gaining a Better Understanding of DbContext and Entity

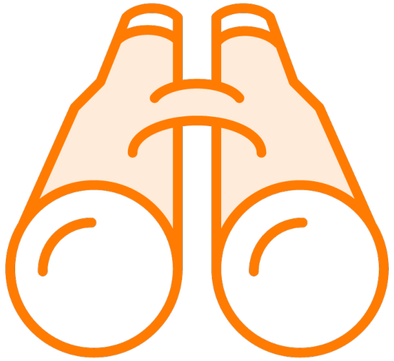
A fundamental understanding of how things work will pay off in productivity.

Tracking Components



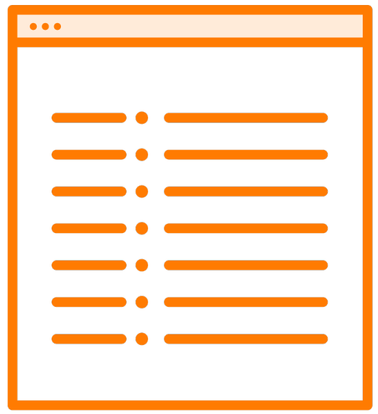
DbContext

Represents a session with the database



DbContext.ChangeTracker

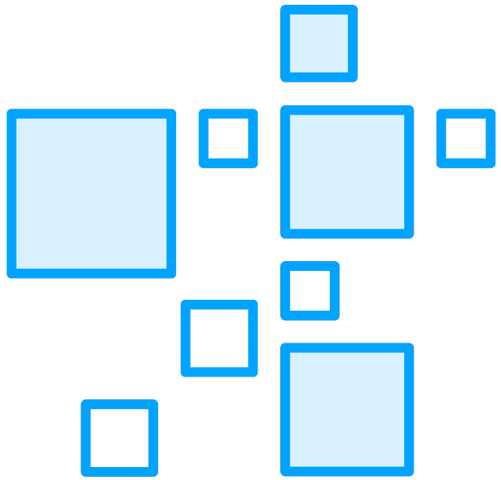
Manages a collection of EntityEntry objects



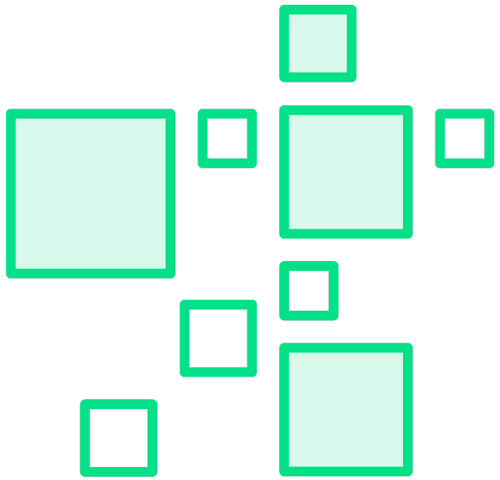
EntityEntry

State info for each entity: CurrentValues, OriginalValues, State enum, Entity & more

Entity and the DbContext

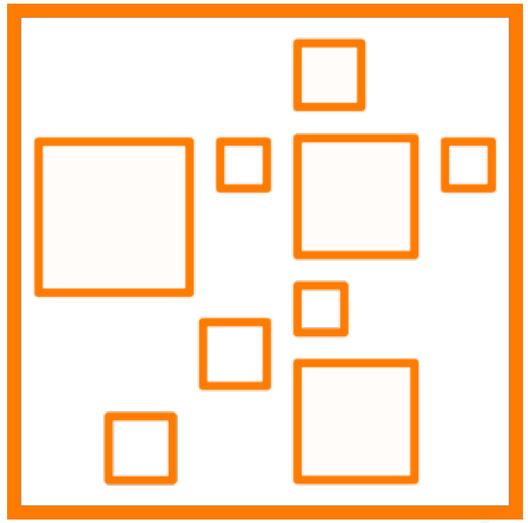


In-Memory Objects



Entities

In-memory objects with key (identity) properties that the DbContext is aware of

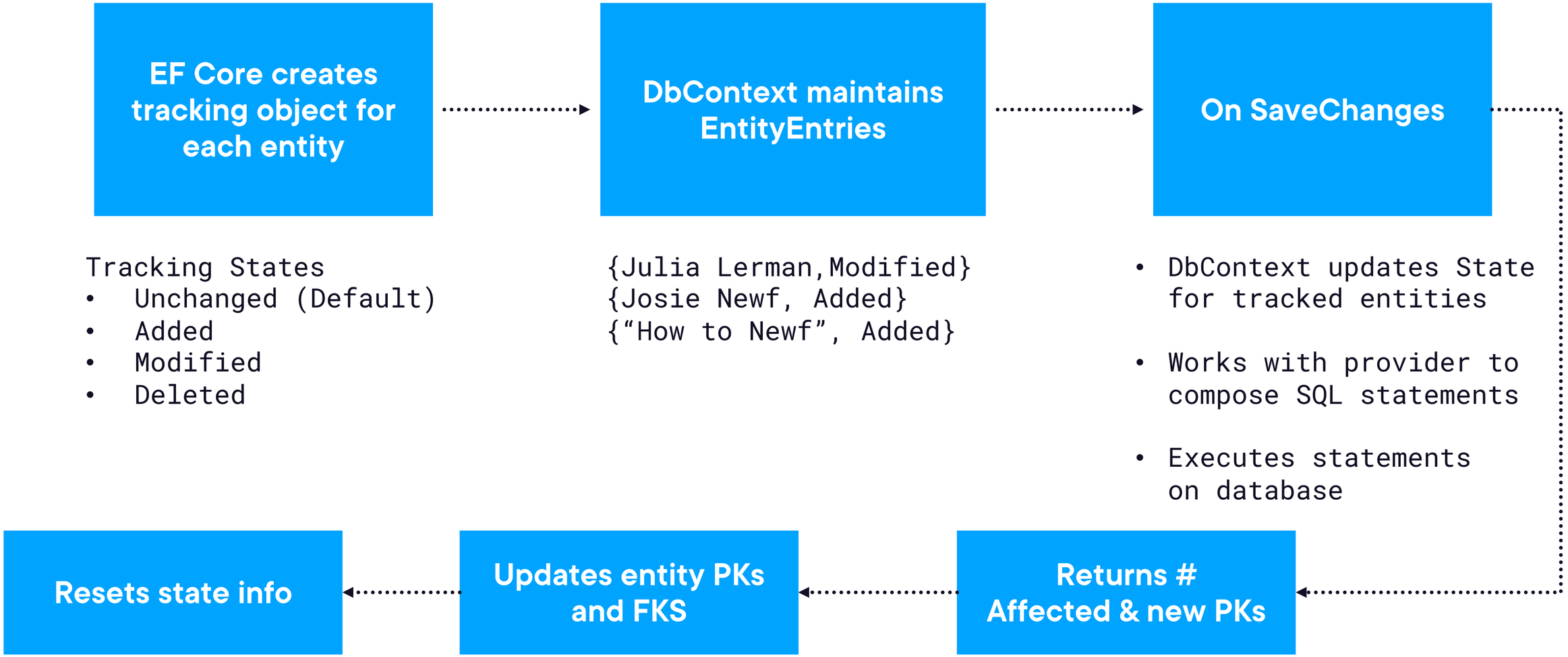


DbContext

Contains EntityEntry objects with reference pointers to in-memory objects

Understanding Tracking and Saving Workflow

Tracking and Saving Workflow



Inserting Simple Objects

Add from DbSet or DbContext

```
context.Authors.Add(…)
```

Track via DbSet

DbSet knows the type
DbContext knows that it's *Added*

```
context.Add(…)
```

Track via DbContext

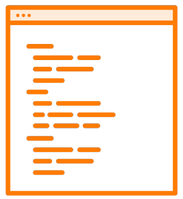
DbContext will discover type
Knows it's *Added*

Updating Simple Objects

How DbContext Discovers About Changes

DbContext.ChangeTracker.DetectChanges()
Reads each object being tracked and updates state details in EntityEntry object

DbContext.SaveChanges()
Always calls DetectChanges for you



You can call DetectChanges() in your code if needed

The entities are just plain objects and don't communicate their changes to the DbContext.

Various Paths for Tracking to Detect Edits

If DbContext is aware of object, it will detect changes on SaveChanges

DbSet and DbContext also have an Update() method

More advanced: modify state directly

Updating Untracked Objects

Begin Tracking and Set State to Modified

```
context.Authors.Update(…)
```

Track via DbSet

DbSet knows the type
Knows right away that it's Modified

```
context.Update(…)
```

Track via DbContext

DbContext will discover the type
Knows right away that it's *Modified*

More Pointers for Disconnected Data

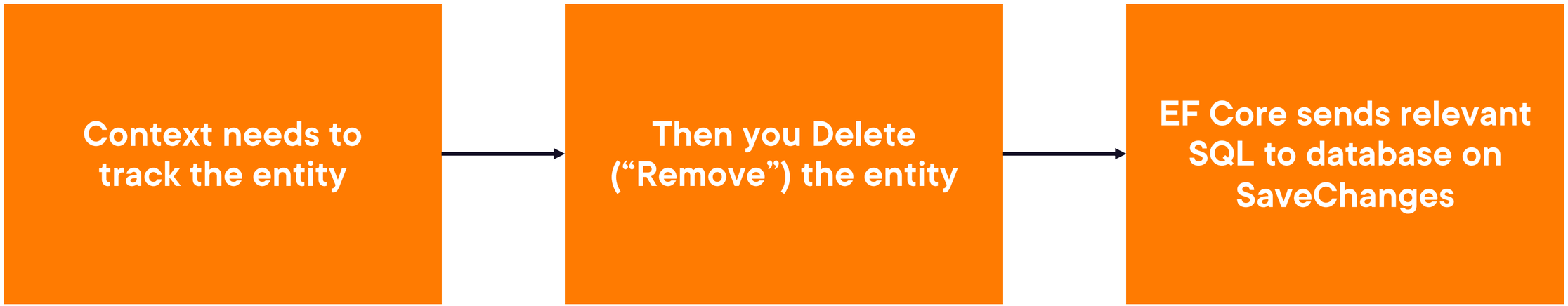
Update is not typically needed when object is tracked

Update and Add and Remove are important for untracked objects

Your app logic will need to know when to call Update, Add or Remove

Deleting Simple Objects

Deleting May Seem a Little Weird

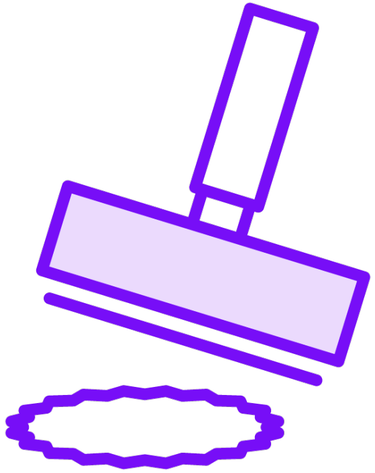


There is no Delete(id) similar to Find(id)

Deleting without Querying

Fake object with key property filled
Watch out for possible side effects

Stored procedure via raw SQL
for untracked data
Further on in course



Soft delete via Global Query Filters
Link in resources

Bulk Delete

ExecuteDelete()
for untracked data

Tracking Multiple Objects and Batch Support

Inserting Multiple Objects

The Tedious Way

```
context.Authors.Add( new Author { FirstName = "Ruth", LastName = "Ozeki" });
context.Authors.Add( new Author { FirstName = "Sofia", LastName = "Segovia" });
context.Authors.Add( new Author { FirstName = "Ursula K.", LastName = "LeGuin" });
context.Authors.Add( new Author { FirstName = "Hugh", LastName = "Howey" });
context.Authors.Add( new Author { FirstName = "Isabelle", LastName = "Allende" });
```

Or Build a List<Author> and...

```
context.Authors.AddRange(authorList);
```

Tracking Multiple Entities

```
context.Authors.AddRange(…)
context.Authors.UpdateRange(…)
context.Authors.RemoveRange(…)
```

Track via DbSet

DbSet indicates type

```
context.AddRange(…)
context.UpdateRange(…)
context.RemoveRange(…)
```

Track via DbContext

Context will discover type(s)

Batched Add/Update/Delete

Min and Max driven by provider

SQL Server minimum is 2, and maximum is 42

Inserts return new key in OUTPUT

Update & Delete return 1 in OUTPUT

Batching can combine types and operations

```
public class SqlServerModificationCommandBatch :
 AffectedCountModificationCommandBatch
{
     // https://docs.microsoft.com/sql/sql-server/maximum-capacity-specifications-for-sql-server
    private const int DefaultNetworkPacketSizeBytes = 4096;
    private const int MaxScriptLength = 65536 * DefaultNetworkPacketSizeBytes / 2;

    /// <summary>
    ///     The SQL Server limit on parameters, including two extra parameters to
     ///        sp_executesql (@stmt and @params).
    /// </summary>
    private const int MaxParameterCount = 2100 - 2;
```

Additional Batch Constraints of SQL Server Provider

Note that min, max, etc. are configurable

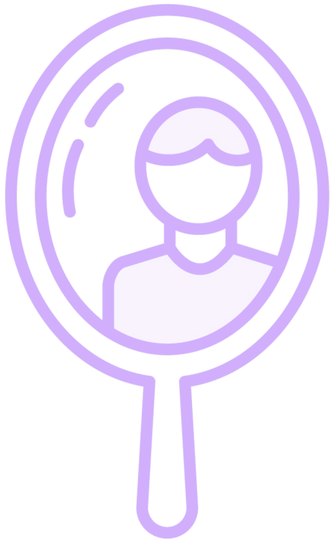
Updating and Deleting Multiple Untracked Objects

The Methods

[Query].ExecuteDelete()

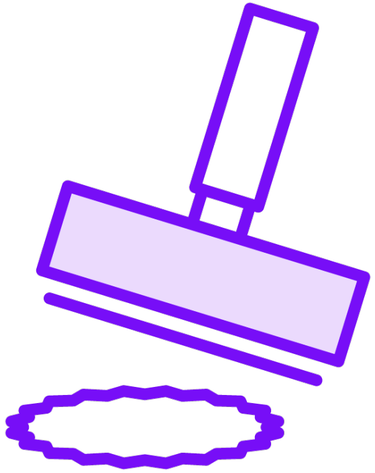
[Query].ExecuteUpdate()

Deleting without Querying



Fake object with key property filled
Watch out for possible side effects

Stored procedure via raw SQL
for untracked data
Further on in course

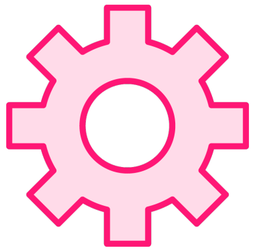


Soft delete via Global Query Filters
Link in resources

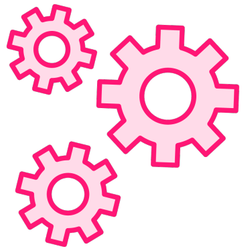
Bulk Delete

ExecuteDelete()
for untracked data

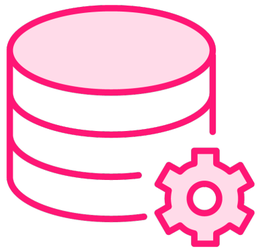
Executing Directly in the Database on Untracked Data



Delete (or update) a row when you only know its' ID



Delete (or update) rows matching a certain criteria



Avoid wasting time and resources retrieving unneeded objects

Execute Means “Do It Now”

Executes right away

Does not interact with ChangeTracker

Returns # of rows affected

ExecuteUpdate

Set a simple value

```
[Query].ExecuteUpdate(
   setters => setters.SetProperty(b => b.PropertyToChange, newvalue)
   );
```

Set a value based on a property

```
[Query].ExecuteUpdate(
   setters => setters.SetProperty(b => b.PropertyToChange,b=>Transform(b.Property))
   );
```

Do not mix tracking and these Execute methods!

Review

ChangeTracker and SaveChanges provides a lot of benefits

An entity is unaware of the change tracker

Database provider works with EF Core to send correct and efficient commands

Batches insert, update, and delete commands to reduce roundtrips

Determines if an explicit transaction is needed or not

DB generated values returned efficiently and propagated back into entities

Up Next:

Controlling Database Creation and Schema Changes with Migrations

Resources

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

EF Core Documentation: learn.microsoft.com/ef/core

Article about SQL Server Merge Joins:
brentozar.com/archive/2017/05/case-entity-framework-cores-odd-sql/

Soft Delete with Global Query Filters in EF Core Docs:
learn.microsoft.com/ef/core/querying/filters