Working with Raw SQL, Views and Stored Procedures

Julie Lerman
Most Trusted Authority on Entity Framework Core

@julielerman   thedatafarm.com

EF Core Allows You To Work Directly With

Raw SQL	Stored Procedures	Views
Map to Queries in DbContext	Table Value Functions	Scalar Functions

Overview

Querying with raw SQL commands and stored procedures

Using migrations to add views and stored procedures and more to a database

Understanding and mapping keyless entities

Mapping and querying with database views

Executing non-query SQL commands on the database

Querying with Raw SQL

Use Query Commands or Call Stored Procedures

```
SELECT * FROM Authors
```

Pass in the SQL

```
EXEC MyStoredProc, p1, p1
```

Execute stored procedures and pass in any needed parameters

Two DbSet Methods to Query Using Your SQL

```
DbSet.FromSqlRaw( )
```

```
DbSet.FromSqlInterpolated( )
```

```
_context.Authors.FromSqlRaw(“some sql string”).ToList();

_context.Authors.FromSqlInterpolated($“some sql string {var}”).ToList();
```

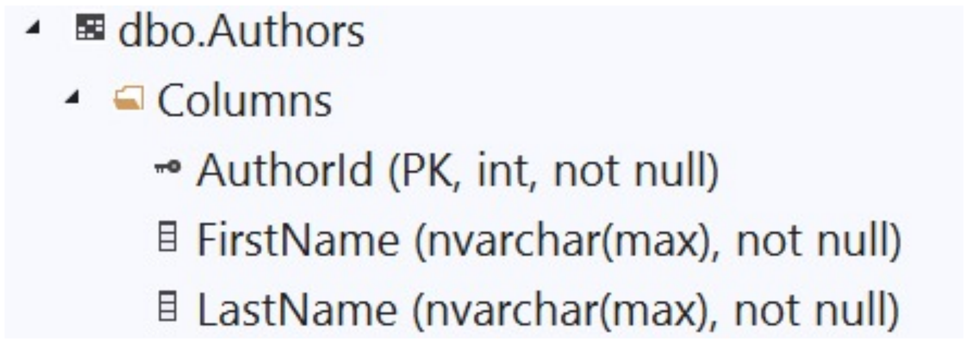
DbSet Methods to Run Raw SQL

Retrieve entities without relying on LINQ or EF Core’s generated SQL
Use parameters to avoid SQL injection
Expects results to be in the shape of the DbSet’s entity
Creates an IQueryable, so you still need an execution method

Must Align with Entity Scalars & DB Columns

FromSqlRaw(“SELECT AuthorId, FirstName, LastName FROM Authors”)

```
public class Author
{
    public Author()
    {
        Books = new List<Book>();
    }
    public int AuthorId { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public List<Book> Books { get; set; }
}
```



```
dbo.Authors
  Columns
    AuthorId (PK, int, not null)
    FirstName (nvarchar(max), not null)
    LastName (nvarchar(max), not null)
```

```
.FromSqlRaw("select * from authors").FirstOrDefault(a=>a.Id==3)

.FromSqlRaw("select * from authors").OrderBy(a=>a.LastName)

.FromSqlRaw("select * from authors").Include(a=>a.Books)
```

You Can Compose on Top of Raw SQL Queries

LINQ methods such as Where and OrderBy
LINQ execution methods like ToList and FirstOrDefault
Iqueryable methods like Include and AsNoTracking
Not DbSet methods e.g., Find!

```
_context.Authors.FromSqlRaw(“some sql string”).ToList();

_context.Authors.FromSqlInterpolated($“some sql string {var}”).ToList();
```

DbSet Methods to Run Raw SQL

Expects results to be in the shape of the DbSet’s entity
Special method for interpolated strings
Creates an IQueryable, so you still need an execution method (can be an async method)
Use parameters to avoid SQL injection

Rules and Limitations for Raw SQL Results

- ✓ Must return data for all properties of the entity type
- ✓ Column names in results match mapped column names
- ✓ Query can’t contain related data
- ✓ Only query entities and keyless entities known by DbContext

Interpolation for Parameters

Interpolated Strings

```
string name=“Josie”;
string s=$“Happy Birthday, {name}”;
```

Interpolated Strings in Raw SQL

```
var lastnameStart = "L";
 var ontheflySQL = _context.Authors.FromSqlInterpolated(
       $"SELECT * FROM authors WHERE lastname LIKE '{lastnameStart}%'")
       .ToList();
 var storedProc = _context.Authors.FromSqlInterpolated(
       $“EXEC FindAuthorByLastNameStart'{lastnameStart}%'")
       .ToList();
```

Keeping Your Database Safe with Parameterized Raw SQL Queries

Never use SQL with parameters embedded directly into the string

Learn about SQL Injection from Microsoft Docs

SQL Injection

docs.microsoft.com/sql/relational-databases/security/sql-injection

Combining Strings in C#

```
string name=“Josie”;
string s=“Happy Birthday,” + name;
```

◀ **Concatenated string**

Happy Birthday, Josie

```
string name=“Josie”;
int age=5;
string s= String.Format(
  “Happy {0} Birthday {1},”,
   age, name);
```

◀ **Formatted string**

Happy 5 Birthday, Josie

```
string name=“Josie”;
string s=$“Happy Birthday, {name}”;
```

◀ **Interpolated string**

Happy Birthday, Josie

The Compiler Will Catch a Few Bad Apples

FromSqlInterpolated expects one formatted string as its parameter

FromSqlInterpolated will not accept a string

Don’t depend on the compiler to protect you from unsafe raw SQL!

Review of Safe & Unsafe/UnCompilable Queries

Safe Query strings

Formatted as param of FromSqlRaw

Interpolated as param of FromSqlInterpolated

Unsafe Raw Queries

All concatenated queries are unsafe

Formatted strings in a variable

Interpolated strings in a variable

Won't even compile:

String in FromRawInterpolated

Formatted string in FromSqlInterpolated

Adding Stored Procedures and Other Database Objects Using Migrations

Embedding SQL in Your Apps

If you can keep SQL commands out of your code base, that's a bonus

Sometimes you don't have that option and EF Core is there to support your need

EF Core’s raw SQL methods support calling stored procedures

We Need a Stored Procedure in Our DB

While there are various ways to achieve this, we will use EF Core migrations to do the job.

Consistent workflows are important whether you are solo or working on a team

The Stored Procedure

Retrieve authors who published a book in a range of years

```
EXEC thesproc startyear, endyear
```

Workflow for Adding DB Objects with Migrations

Work out the query directly against the database

Build the command to create the procedure

Test the command by creating & running the stored procedure

Remove the procedure from the database

Add the create procedure command to a new migration

Why Use the Migration for This?

Other team members can easily migrate their own development databases

The migration becomes part of your source code

Useful in other environments such as CI/CD, acceptance testing and possibly production

Running Stored Procedure Queries with Raw SQL

```
FromSqlRaw with formatted string

DbSet.FromSqlRaw(“Exec MyStoredProc, {0}, {1}”, firstValue, secondValue)


FromSqlInterpolated with interpolated string

DbSet.FromSqlInterpolated($”EXEC MyStoredProc {firstValue}, {secondValue})
```

Querying via DbSets Using Stored Procedures

Include values of expected parameters
EF Core will transform this as needed by the database

Examples of Composing on Raw SQL with Sprocs

```
_context.Authors
  .FromSqlRaw("AuthorsSproc,{0}, {1}",
              2010, 2015)
  .OrderBy(a=>a.LastName)
  .ToList();

_context.Authors
  .FromSqlRaw("AuthorsSproc,{0}, {1}",
              2010, 2015)
  .Include(a=>a.Books)
  .FirstOrDefault();

_context.Authors
  .FromSqlRaw("AuthorsSproc,{0}, {1}",
              2010, 2015)
  .Include(a=>a.Books)
  .ToList();
```

- **Other methods like OrderBy or even AsNoTracking**
- **Different LINQ execution methods**
- **Eager load with Include**

Raw SQL Method Rules Apply to Sprocs, Too

SQL cannot return shaped data (use Include() to do that)

Schema of results must match the entity of the DbSet

Column names of results must match property names of entity

Remove the procedure from the database

Add the create procedure command to a new migration

Raw SQL Method Rules Apply to Sprocs, Too

- ✓ **SQL cannot return shaped data (use Include() to do that)**
- ✓ **Schema of results must match the entity of the DbSet**
- ✓ **Column names of results must match property names of entity**

EF Core cannot capture random data, e.g., anonymous types, from raw SQL

Compiler Cannot Detect Bad SQL

SQL errors will only be caught at runtime by the database which will return an error

Using Keyless Entities to Map to Views

Historically, EF and EF Core only understood entities with keys

But now you can use keyless entities

Keyless Entities != Non-Tracking Queries

Entity has a key prop

No-tracking is optional

Maps to tables with PK

Non-Tracking Query

Entity has a key prop

Maps to view and table

Query from the view, update to the table

Mixed Use

Keyless entities are always read-only

Dear Programmers,

We need to see for which authors the artists are working on book covers.

Thanks!

Your friendly editors

This calls for a database view.

Hey, DB guru, we have a favor to ask ....

Mapping Keyless Entities & DB Views

✓ Create DB View

✓ Define Keyless Entity

✓ Mapping the Keyless Entity

✓ Mapping the database view

EF Core maps entities to tables by default

It will expect (& ask migrations to create) a table named AuthorsByArtist!

We want it to use our view.

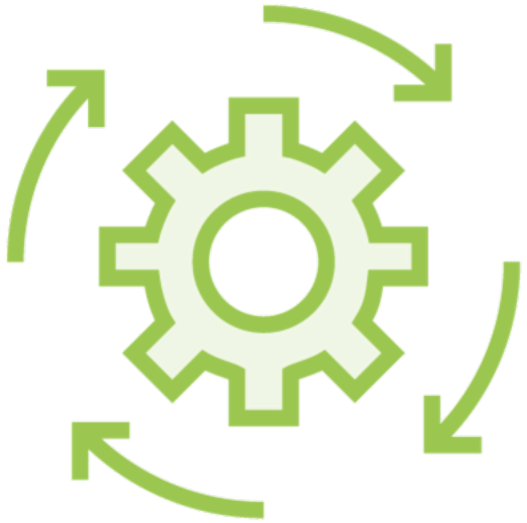
Migrations will not attempt to create a database view that's mapped in a DbContext

Keyless entities will never be tracked.

Full stop.

Querying the Database Views

Not All DbSet Methods Work with Keyless Entities



Find() will compile

But Find() will fail at runtime!

Executing Non-Query Raw SQL Commands

Commands to Execute Raw SQL and Stored Procs

ExecuteSqlRaw

ExecuteSqlInterpolated

Commands to Execute Raw SQL and Stored Procs

DbContext.Database .ExecuteSqlRaw

DbContext.Database .ExecuteSqlInterpolated

```
_context.Database.ExecuteSQLRaw(“some SQL string”);

_context.Database.ExecuteSQLRawAsync(“some SQL string”);

_context.Database.ExecuteSQLInterpolated($“some SQL string {variable}”);

_context.Database.ExecuteSQLInterpolatedAsync($“some SQL string {var}”);
```

Run Raw SQL for Non-Query Commands from the Database property

Only result is number of rows affected
On-the-fly SQL or Stored Procedures

Finally!

We’ll use a stored procedure to delete data

Review

More querying using raw SQL and stored procedures including interpolated strings

Used customized migrations to add views, stored procedures and more to DB

Learned about keyless entities and mapped them to a database view

Used DbSet to query that database view

Execute non-query commands from the DbContext.Database property

...and a handy way to delete entities

Up Next:
Using EF Core in ASP.NET Core and Blazor Apps

Resources

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

EF Core Documentation: docs.microsoft.com/ef

EF Core Query Types Consolidated with Entity Types: docs.microsoft.com/en-us/ef/core/what-is-new/ef-core-3.0/breaking-changes#qt

About SQL Injection: docs.microsoft.com/sql/relational-databases/security/sql-injection

Resources, Continued

Composing Raw SQL Queries with LINQ:
docs.microsoft.com/en-us/ef/core/querying/raw-sql#composing-with-linq

"Mapping Database Scalar Functions" in EF Core 2: Mappings Course:
bit.ly/2LppcMj

Database functions in EF Core Docs: https://docs.microsoft.com/en-us/ef/core/querying/database-functions