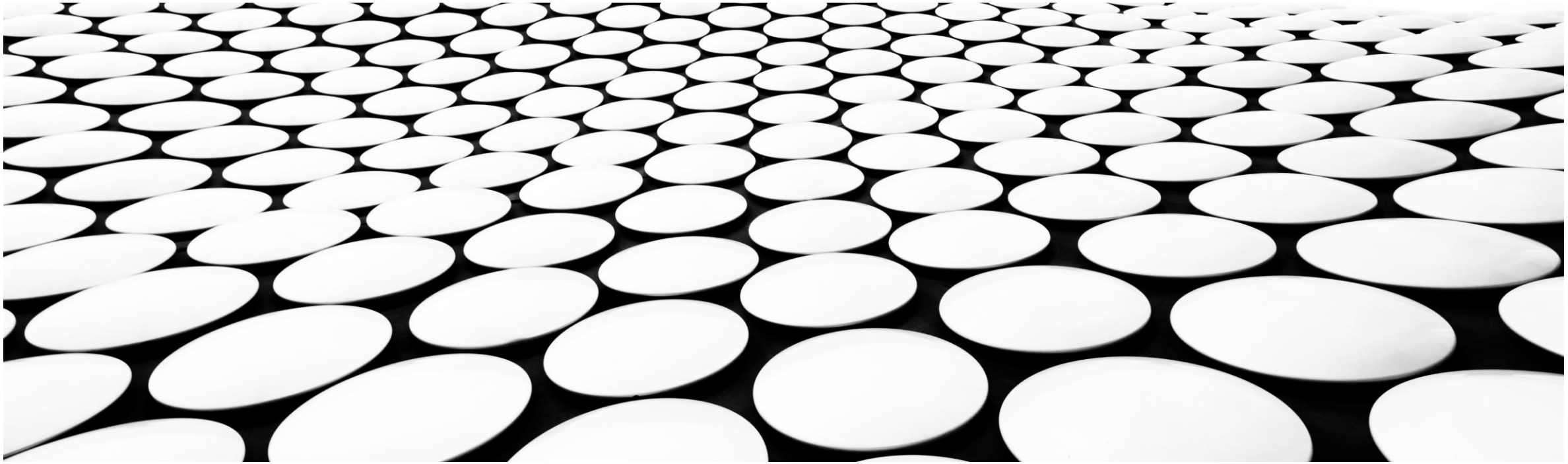
EF CORE 6: WORKING ON THE BUCKET LIST

JULIE LERMAN THEDATAFARM.COM @JULIELERMAN



EF CORE 6 FUNDAMENTALS ON PLURALSIGHT

COMING SOON!

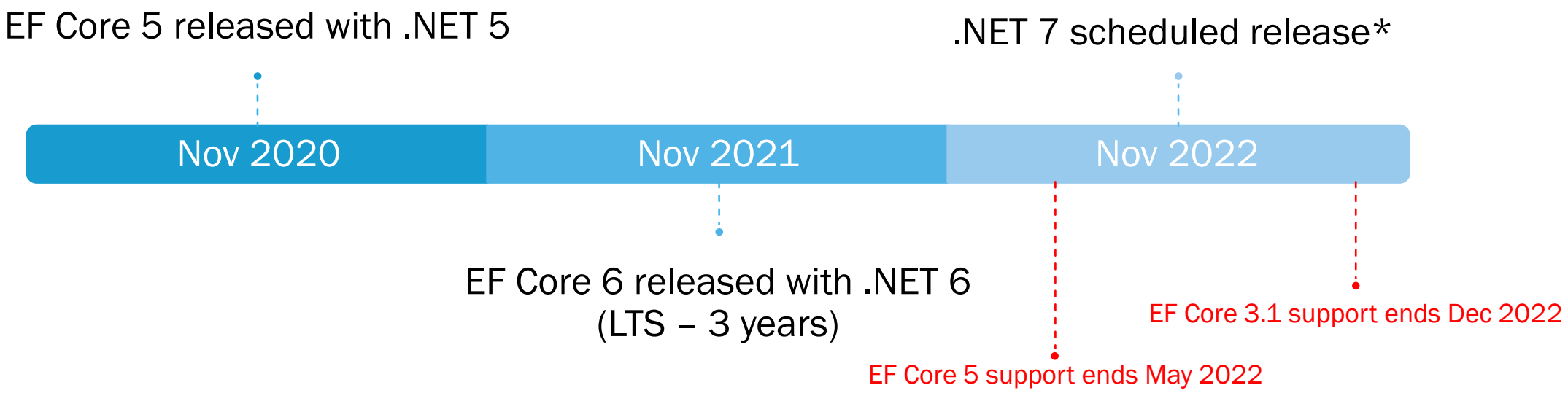
THIS SESSION

High impact changes

Highly requested

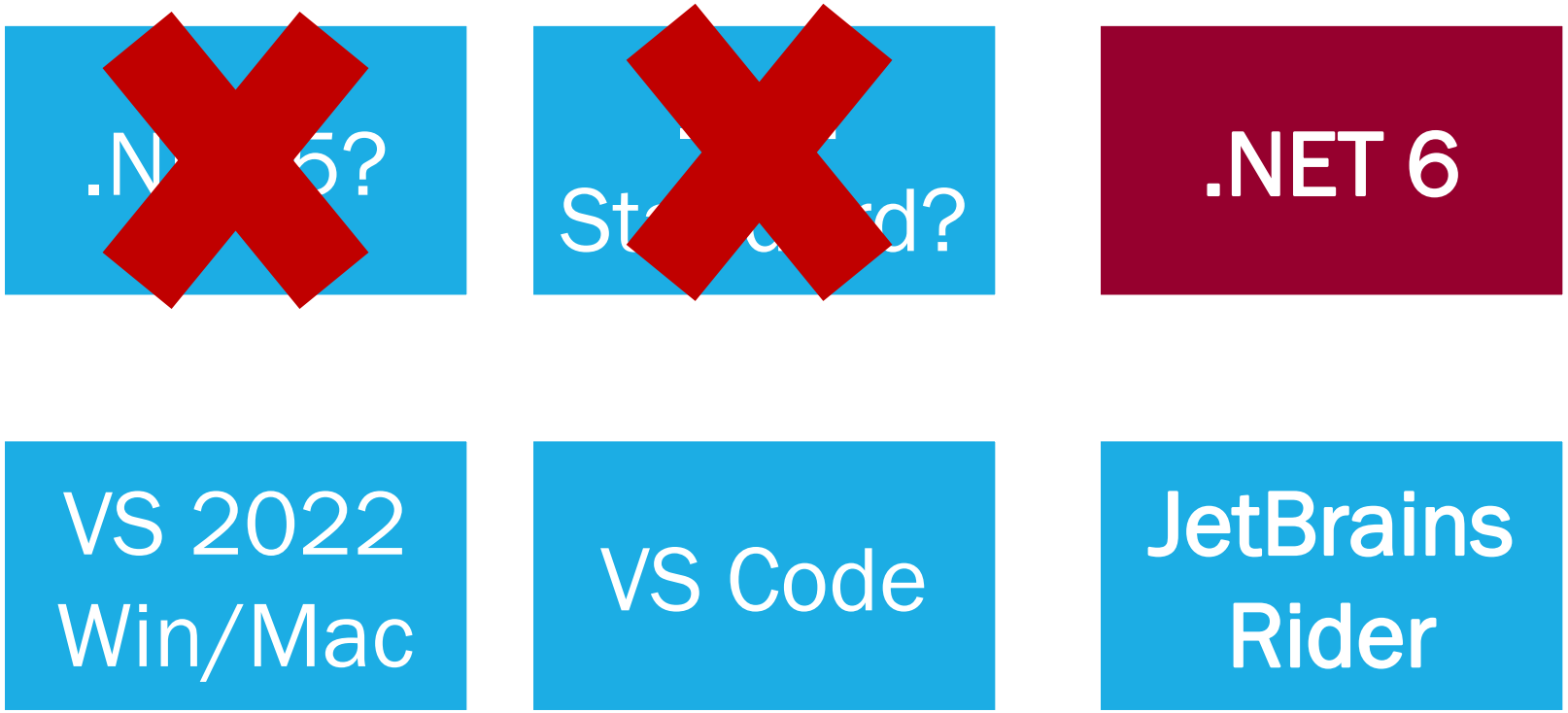
Various notable changes

EF CORE RELEASES ALIGN WITH .NET



*github.com/dotnet/core/blob/main/roadmap.md#net-release-schedule

EF CORE 6 DEPENDS ON .NET 6



QUERY PERFORMANCE

@JulieLerman

RUNTIME PERFORMANCE PUSH WITH DAPPER AS BENCHMARK

Full stack (.NET6 + ASP.NET Core 6 + EF Core 6) perf is **70% faster** on the industry-standard TechEmpower Fortunes benchmark, compared to 5.0.

"EF Core 6.0 itself is **31% faster** executing queries."

Source:EF Core team blog post at bit.ly/EFCore6Perf

EFCORE & DAPPER IN TECHEMPOWER FORTUNES TESTS

All benchmarks executed on aspnet-citrine-lin (TechEmpower hardware), 3 iterations.

Summary

Description	EF Core	Change	Dapper	Change	EF/Dapper Gap
Starting point (EF 5.0 + .NET 5.0)	147,852		230,394		55.8%
EF benchmark improvements (no product changes)	191,189	29.3%	230,394		20.5%
.NET 5 -> .NET 6	209,953	9.8%	263,314	14.3%	25.4%
EF 6.0 + .NET 6.0 (product optimizations)	251,992	31.8%	263,314		4.5%
Total change:		70.4%		14.3%	

Tweaking tests → EF benchmark improvements (no product changes)

Tweaking APIs → EF 6.0 + .NET 6.0 (product optimizations)

Fortune Benchmarks PREVIEW for Round 21:
https://www.techempower.com/benchmarks/#section=test&runid=ab73a704-fc44-49d0-a92a-bae1ba7eb82e

KEY API CHANGES FOR PERFORMANCE GAINS

Pooling & recycling (23%)

DbContext, DbConnection, DbCommand, DbDataReader, internal objects

Reduce expensive checks for logging or interception points (7%)

One check, then delay any enabled logging/interception by 1 second

Avoids interfering with query performance

Opt out of thread safety checks (6.7%)

Expensive for async queries

Opt-in for debugging/testing, opt-out for production

Full details for EF Core API improvements at github.com/dotnet/efcore/issues/23611#issuecomment-778191277

EF Core 6 focus for perf was on NON-tracking queries

Team anticipates making similar effort for tracked queries in EF Core 7

RESULTS OF MY “BACKYARD” BENCHMARKDOTNET TESTS

EF Core 5
on .NET 6

```
NET SDK=6.0.100
 [Host]     : .NET 6.0.0 (6.0.21.52210), X64 RyuJIT
 DefaultJob : .NET 6.0.0 (6.0.21.52210), X64 RyuJIT


                   Method |       Mean |    Error |    StdDev |
------------------------- |----------- |---------:|----------:|
 EFC5_1KRows_withPooling | 3,075.9 us | 78.34 us | 227.28 us |
   EFC5_1Row_WithPooling |   487.2 us |  9.74 us |  14.57 us |
```

EF Core 6
on .NET 6

```
.NET SDK=6.0.100
  [Host]     : .NET 6.0.0 (6.0.21.52210), X64 RyuJIT
  DefaultJob : .NET 6.0.0 (6.0.21.52210), X64 RyuJIT


|                  Method |       Mean |    Error |   StdDev |
|------------------------- |----------- |---------:|---------:|
| EFC6_1KRows_withPooling | 1,466.1 us | 24.80 us | 23.20 us |
|   EFC6_1Row_WithPooling |   483.3 us |  5.97 us |  5.59 us |
```


STARTUP
PERFORMANCE
“COMPILED MODELS”
@JulieLerman
flickr.com/photos/53370644@N06/4976491750

WHEN DOES EF CORE BUILD ITS IN-MEMORY DATA MODEL?

Yep!

Nope

Instantiate DbContext

First DB Interaction in App Instance
(e.g., execute query or call SaveChanges)

Nope

Subsequent DB Interaction

ARTHUR VICKER'S DEMO WITH COMPLEX MODEL USING SQLITE

Model: 449 entity types 6390 properties 720 relationships

On-the-fly runtime model

```
AMD Ryzen 5 2600, 1 CPU, 12 logical and 6 physical cores
.NET SDK=6.0.100-rc.1.21463.6
  [Host]     : .NET 6.0.0 (6.0.21.45113), X64 RyuJIT DEBUG
  Job-SNYBMI : .NET 6.0.0 (6.0.21.45113), X64 RyuJIT

InvocationCount=1  IterationCount=20  UnrollFactor=1

|           Method |     Mean |     Error |   StdDev |
|----------------- |--------:|---------:|---------:|
| TimeToFirstQuery | 1.799 s | 0.0175 s | 0.0202 s |
```

Using pre-compiled model

```
AMD Ryzen 5 2600, 1 CPU, 12 logical and 6 physical cores
.NET SDK=6.0.100-rc.1.21463.6
  [Host]     : .NET 6.0.0 (6.0.21.45113), X64 RyuJIT DEBUG
  Job-YOVZPM : .NET 6.0.0 (6.0.21.45113), X64 RyuJIT

InvocationCount=1  IterationCount=20  UnrollFactor=1

|           Method |     Mean |    Error |   StdDev |
|----------------- |---------:|--------:|--------:|
| TimeToFirstQuery | 157.2 ms | 6.52 ms | 7.51 ms |
```

Approximately 11X faster

*Using SQLite
Caching is disabled for testing purposes

github.com/ajcvickers/CompiledModelsDemo

IMPLEMENTING COMPILED MODELS

CLI: dotnet ef dbcontext optimize

PS: Optimize-DbContext

Creates a folder with optimized model files

Tell DbContext to use the version from those optimized files:

```
protected override void OnConfiguring (DbContextOptionsBuilder optionsBuilder) {
     optionsBuilder.UseSqlite("Data Source=MyDatabase.db");
      .UseModel(MyDbContextModel.Instance);
 }
```

“DEVOPS FRIENDLY” MIGRATIONS

“MIGRATION BUNDLES”

@JulieLerman

APPLYING MIGRATIONS IN WEB BASED APPS IS TRICKY

Migrations in app startup? NOPE!

Generate SQL script and feed to your favorite tool e.g., FlywayDB

New to EF Core 6: Bundle migrations and execute

IMPLEMENTING MIGRATION BUNDLES

CLI: dotnet ef migrations bundle

PS: Bundle-Migration

Key Options - - output, - - self-contained, - - force

.\efbundle (efbundle.exe)

Can be run with - - connection
Creates idempotent logic

TEMPORAL TABLE SUPPORT

@JulieLerman

WHAT ARE TEMPORAL TABLES?

- dbo.People (System-Versioned)
  - dbo.PeopleHistory (History)
    - Columns
      - Id (PK, uniqueidentifier, not null)
      - FirstName (nvarchar(max), null)
      - LastName (nvarchar(max), null)
      - MiddleName (nvarchar(max), null)
      - PeriodEnd (datetime2(7), not null)
      - PeriodStart (datetime2(7), not null)

→

- dbo.PeopleHistory (History)
  - Columns
    - Id (uniqueidentifier, not null)
    - FirstName (nvarchar(max), null)
    - LastName (nvarchar(max), null)
    - MiddleName (nvarchar(max), null)
    - PeriodEnd (datetime2(7), not null)
    - PeriodStart (datetime2(7), not null)

Table view from SSMS Object Explorer

EF CORE 6 MAPPINGS, MIGRATIONS & SCHEMA

Map with IsTemporal()

```
modelBuilder.Entity<entity>()
    .ToTable(tb => tb.IsTemporal());
```

Migration adds

```
PeriodStart datetime + temporal annotations
PeriodEnd datetime + temporal annotations
Temporal annotations for table
```

Generated SQL

```
DECLARE @historyTableSchema sysname = SCHEMA_NAME()
EXEC(N'CREATE TABLE [People] (
    [columns from properties]
    [PeriodEnd] datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
    [PeriodStart] datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
    CONSTRAINT [PK_People] PRIMARY KEY ([Id]),
    PERIOD FOR SYSTEM_TIME([PeriodStart], [PeriodEnd])
) WITH (SYSTEM_VERSIONING = ON
    (HISTORY_TABLE = [' + @historyTableSchema + N'].[PersonHistory]))');
```

LINQ METHODS FOR QUERYING TEMPORAL TABLES

TemporalAsOf

TemporalAll

TemporalBetween

TemporalFromTo

TemporalContainedIn

BULK CONVENTIONS & CONVERSIONS

@JulieLerman

SIMPLER BULK CONFIGURATIONS

OLD

```
foreach (var entity in modelBuilder.Model.GetEntityTypes ())
    {
      foreach (var prop in entity.GetProperties ()
               .Where (p => p.ClrType == typeof (string)))
      {
        prop.SetColumnType (“nvarchar(100)");
      }
    }
```

NEW

```
configurationBuilder.Properties<string>()
    .HaveColumnType ("nvarchar(100)");
```

NEW DBCONTEXT METHODS, TYPES & "HAVE" MAPPINGS FOR BULK MAPPINGS

DDD

```
protected override void ConfigureConventions
 (ModelConfigurationBuilder configurationBuilder)
{
  configurationBuilder.Properties<string>()
    .HaveColumnType ("nvarchar(100)");

  configurationBuilder.Properties<AddressTypeEnum> ()
    .HaveConversion<string> ();

  configurationBuilder.Properties<Color>()
    .HaveConversion<MyCustomColorToStringConverter>();
}
```

NOTE: LESS OPTIONS FOR BULK CUSTOM CONVERSIONS

Single property: specify write & read functions on-the-fly, in a method or custom class

```
modelBuilder.Entity<Address>()
    .Property(ad=>ad.StructureColor)
      .HasConversion(c=>c.ToString(),s=>Color.FromName(s));
```

Write to DB

Read from DB

Bulk conversion: custom converter class only. Why? Can't pre-compile the model!

```
public class ColorToStringConverter : ValueConverter<Color, string> {

   public ColorToStringConverter () : base (ColorString, ColorStruct) { }


  private static Expression<Func<Color, string>> ColorString = v => new String (v.Name);
  private static Expression<Func<string, Color>> ColorStruct = v => Color.FromName (v);
}
```

Write to DB

Read from DB

DOCUMENT DB (*COSMOS DB*) IMPROVEMENTS

@JulieLerman

DOCUMENT DATABASE: IMPLICIT OWNERSHIP

No need to specify OwnsOne/OwnsMany mapping

```
{
  "Id": "18369f48-c9c9-41e0-a6c1-427dcca4816b",
  "Discriminator": "Person",
  "id": "Person|
         18369f48-c9c9-41e0-a6c1-427dcca4816b",
  "Name": {
    "FirstName": "Andriy",
    "LastName": "Svyryd"
  }
}
```

Related objects with no DbSet

```
{
  "Id": "18369f48-c9c9-41e0-a6c1-427dcca4816b",
  "Discriminator": "Person",
  "id": "Person|
         18369f48-c9c9-41e0-a6c1-427dcca4816b",
   "Addresses": [
    {
      "Id": "16b8dda6-f3d5-43d0-a58b-ced48724666c",
      "AddressType": 2,
      "PostalCode": "12345",
      "Street": "Two Main",
      "StreetLine2": null
    }
  ],
}
```

Also will nest related objects with no DbSets

DOCUMENT DATABASE: RICHER LOGGING DETAILS

```
info: 8/30/2021 14:41:13.651
CosmosEventId.ExecutedReadNext[30102]
(Microsoft.EntityFrameworkCore.Database.Command) Executed ReadNext
(169.6126 ms, 2.93 RU) ActivityId='4e465fae-3d49-4c1f-bd04-
142bc5d0b0a1', Container='Shapes', Partition='(null)', Parameters=[]
SELECT c FROM root c WHERE ((c["Discriminator"] = "Triangle") AND
(c["id"] = "Equilateral")) OFFSET 0 LIMIT 2
```

DOCUMENT DATABASE:
STORE COLLECTIONS & DICTIONARIES OF PRIMITIVES

DDD

```
person.Nicknames = new List<string>
 { "Shay", "Roji", "Postgres Guy" };




person.DailyChocolate =
 new Dictionary<string, string>
 { { "Monday", "Dark Chocolate" },
   { "Tuesday", "Salted Milk Chocolate" }
 };
```

```
"Nicknames": [
    "Shay",
    "Roji",
    "Postgres Guy"
  ],

"DailyChocolate": {
    "Monday": "Dark Chocolate",
    "Tuesday": "Salted Milk Chocolate"
  }
```

MORE EF6 PARITY FOR GROUPBY

@JulieLerman

ACCESS PROPERTIES OF NAVIGATIONS AFTER GROUPBY

```
_context.People.Include(p=>p.Addresses)
   .GroupBy(p=>p.LastName)
   .Select(g=>new
   { g.Key,Count=g.Count(),
     AddressCount=g.Sum(p=>
        p.Addresses.Count(a=>a.Street.Contains("Main")))
   }).ToList();
```

SERVER SIDE EVAL CAN RETURN ENTITIES WITH GROUPBY

EF Core 5: Aggregates like Count are supported

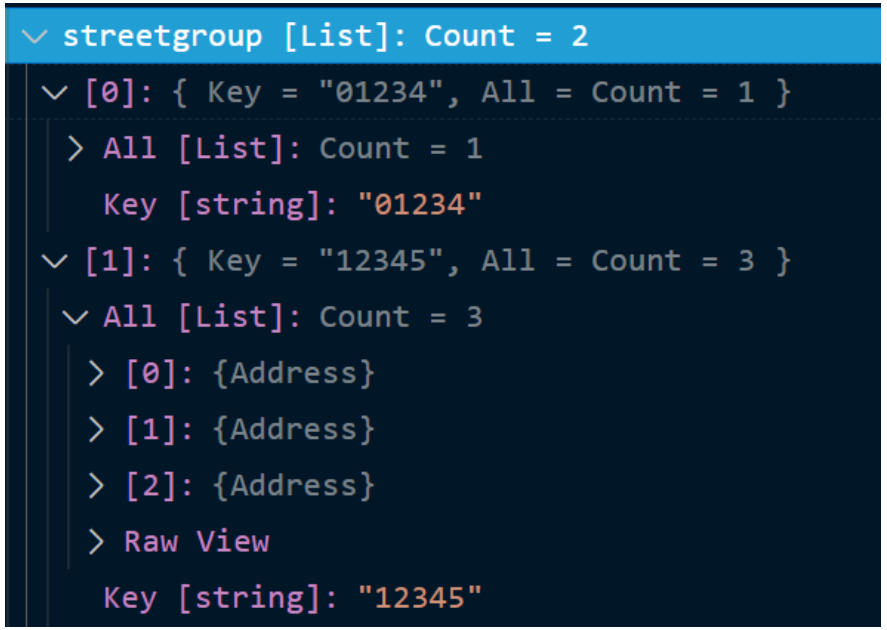
```
_context.Set<Address>()
.AsEnumerable()
.GroupBy(g=>new{g.Key,g.Count)
.ToList();
```

EF Core 5: AsEnumerable required for entity details

```
_context.Set<Address>()
.AsEnumerable()
.GroupBy(ad=>ad.PostalCode)
.ToList();
```

EF Core 6: Entities can be selected

```
_context.Addresses
  .GroupBy(ad=>ad.PostalCode)
  .Select(g=>new{g.Key,All=g.ToList()})
  .ToList();
```



THIS ALLOWS US TO USE FIRST & FIRSTORDEFAULT …

FirstOrDefault (in each group)

```
_context.Addresses
.GroupBy(ad=>ad.PostalCode)
        .Select(g=>new{g.Key,
                First=g.FirstOrDefault()} )
.ToList();
```

```
streetgroup [List]: Count = 2
[0]: { Key = "01234", First = {Address} }
  First: {Address}
  Key [string]: "01234"
[1]: { Key = "12345", First = {Address} }
  First: {Address}
  Key [string]: "12345"
```

AND … TOP N WORK!

Top N via Take (in each group)

```
_context.Addresses
.GroupBy(ad=>ad.PostalCode)
.Select(g=>new{g.Key,Top2=g.Take(2)})
.ToList();
```

```
streetgroup [List]: Count = 2
  [0]: { Key = "01234", Top2 = Count = 1 }
    Key [string]: "01234"
    Top2 [IEnumerable]: Count = 1
      [0]: {Address}
      Raw View
  [1]: { Key = "12345", Top2 = Count = 2 }
    Key [string]: "12345"
    Top2 [IEnumerable]: Count = 2
      [0]: {Address}
      [1]: {Address}
```

Top N via Take with Projection (in each group)

```
_context.Addresses
.GroupBy(ad=>ad.PostalCode)
.Select(g=>g.Take(2).Select(ad=>ad.Street))
.ToList();
```

```
streetgroup [List]: Count = 2
  [0] [IEnumerable]: Count = 1
    [0] [string]: "1 Cross"
  [1] [IEnumerable]: Count = 2
    [0] [string]: "1 Main"
    [1] [string]: "2 Main"
```

MINIMAL API SUPPORT IN ASP.NET CORE APPS

```
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddSqlite<MyDbContext>("Data Source=mydatabase.db");
```

CONTROL THE ORDER COLUMNS FOR DB MIGRATIONS

```
modelBuilder.Entity<Address>()
   .Property(ad=>ad.Street).HasColumnOrder(1);
```

```
Column(Order=1)]
public string PostalCode { get; set; }
```

MORE FLEXIBLE SQL SERVER FREE-TEXT SEARCH

DDD

Name type property

is mapped to store as JSON in nvarvar(max)

(this is not new!)

```
public class Person
{
  public int Id { get; set; }
  public Name Name{ get; set; }
}
public class Name
{
  public string First { get; set; }
  public string Last { get; set; }
}
```

```
modelBuilder.Entity<Person>() .Property(e => e.Name).HasConversion(
   v => JsonSerializer.Serialize (v, (JsonSerializerOptions)null),
   v => JsonSerializer.Deserialize<PersonName>(v, (JsonSerializerOptions)null));
```

Results Messages

	Id	Name
1	fb36934b-931d-4843-d029-08d9…	{"FirstName":"Max","LastName":"Arshinov","FullName":"Max Arshinov"}

...AND NOW WE CAN QUERY THAT DATA

DDD

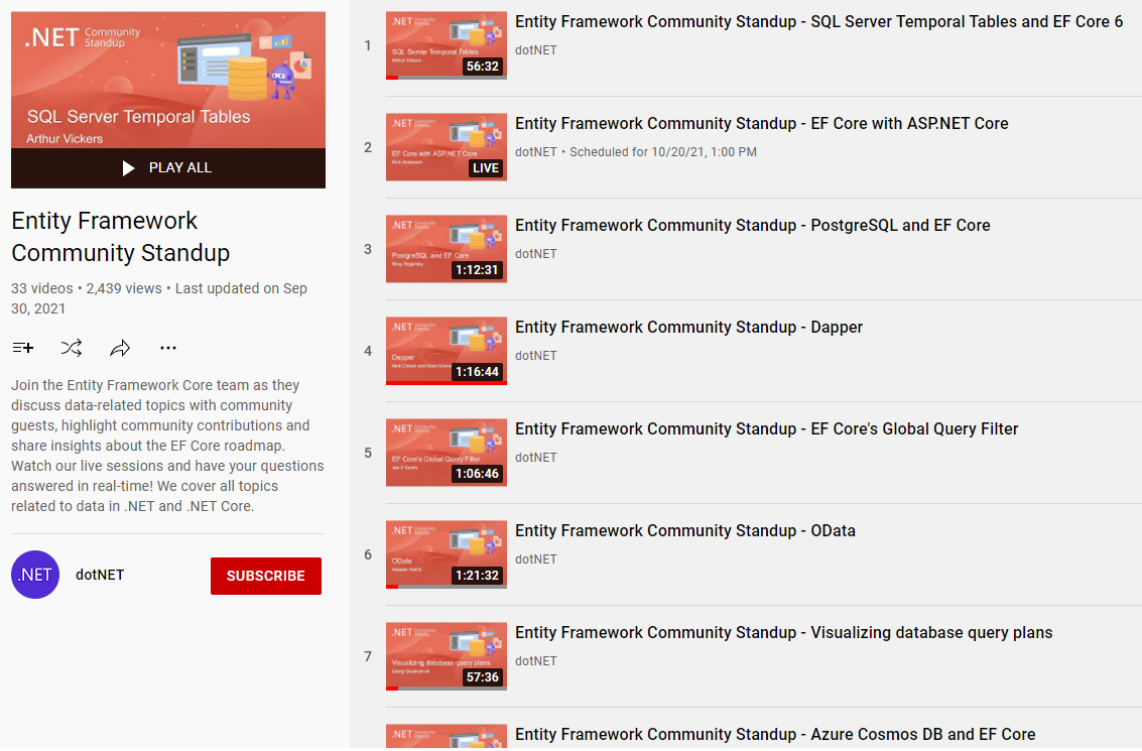
In EF Core 5, this query to search the NAME property will fail because Contains & FullText functions only accept strings

```
var result = context.Customers
 .Where(e => EF.Functions.Contains(e.Name, "Max"))
 .ToList();
```

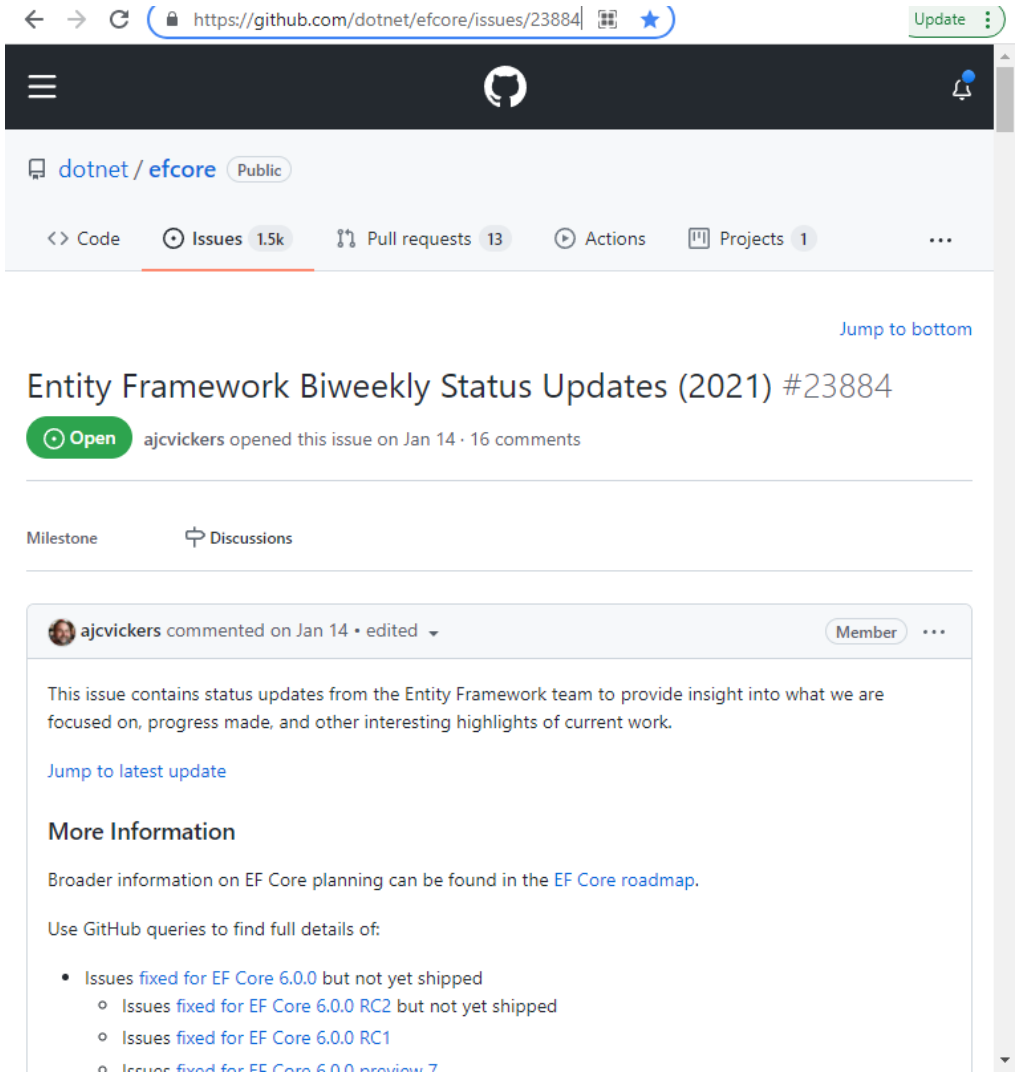
EF Core 6 relaxes the parameter requirements. Here is the SQL searching the nvchar(max) field

```
SELECT [c].[Id], [c].[Name]
FROM [Customers] AS [c]
WHERE CONTAINS([c].[Name], N'Max')
```

COMMUNITY, TRANSPARENCY, OPEN SOURCE



bit.ly/EFCoreStandup



github.com/dotnet/efcore/issues/23884

PUTTING A FACE TO THE TEAM … AND THE COMMUNITY!

https://bit.ly/3BJf7EK

@JulieLerman

Thank you from the team

A big thank you from the EF team to everyone who has used EF over the years!


Arthur Vickers


Andriy Svyryd


Brice Lambson


Jeremy Likness


Maurycy Markowski


Shay Rojansky


Smit Patel

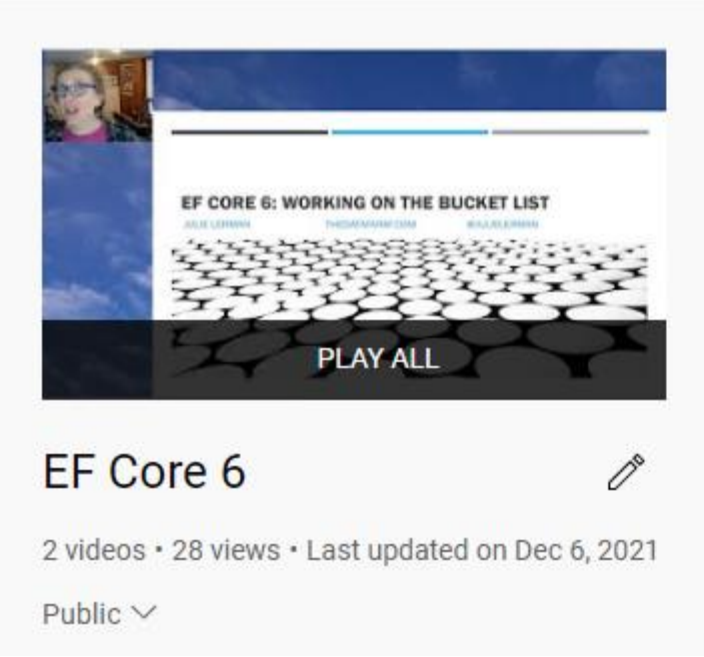
Thank you to our contributors!

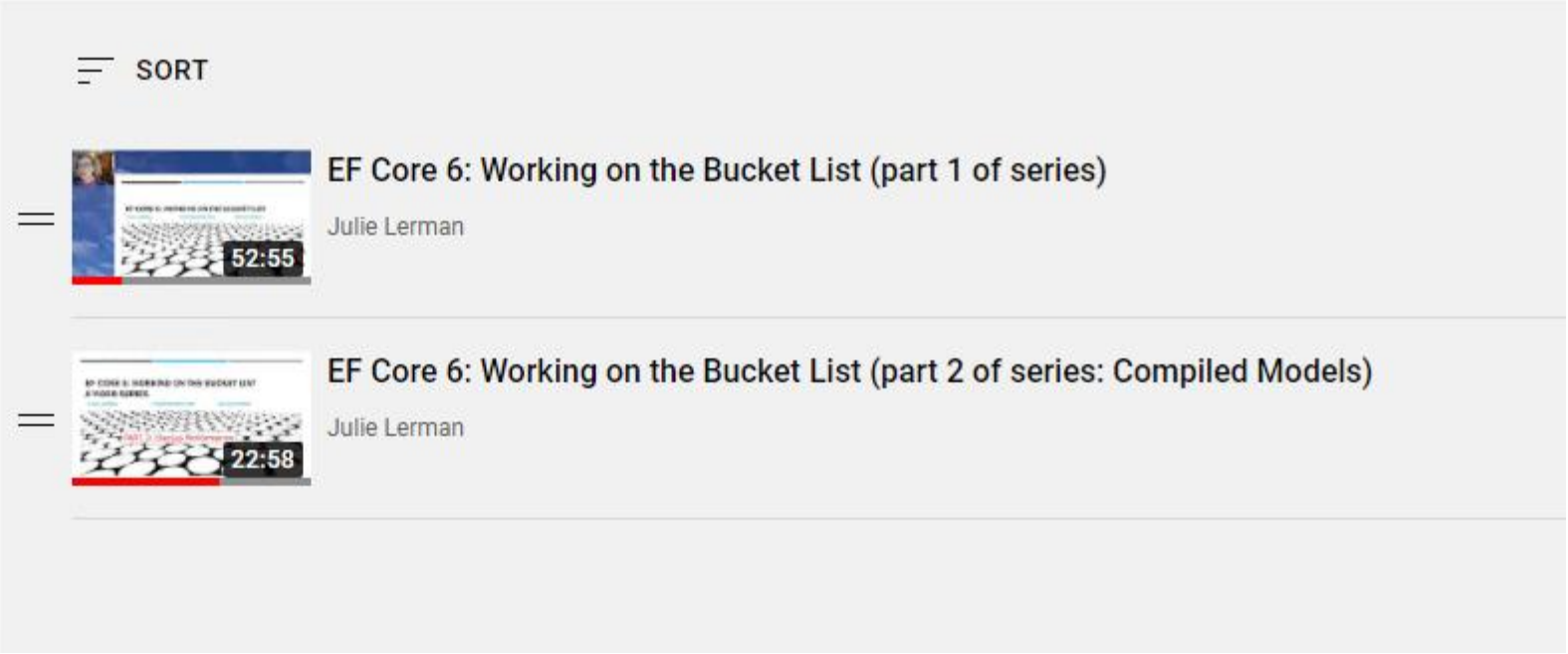
We are grateful to our amazing community of contributors. Our success is founded upon the shoulders of your efforts and feedback. If you are interested in contributing but not sure how or would like help, please reach out to us! We want to help you succeed. We would like to publicly acknowledge and thank these contributors for investing in the success of EF Core 6.0.

AkinSabriCam	alexernest	alexpotter10	Ali-YousefiTelori
#1	#1	#1	#1, #2
alireza-rezaee	andrejs86	AndrewKitu	ardalis
#1	#1	#1	#1
CaringDev	carlreid	carlreinke	cgrevil
#1, #2	#1, #2	#1, #2	#1
cgrimes01	cincuranet	dan-giddins	dennisseders

WORKING (SLOWLY) ON A VERY EXPANDED VERSION OF THIS TALK

bit.ly/efc6youtube





PLURALSIGHT COURSE COMING IN LATE WINTER 2022

Entity Framework Core 6 Fundamentals

Julie Lerman
Most Trusted Authority on Entity Framework

@julielerman thedatafarm.com

IMPORTANT LINKS

- Julie Lerman website: thedatafarm.com
- Demos from this session: github.com/Julielerman/EFCore6Demos
- My Pluralsight Author Page: bit.ly/PSJulie

- MS Docs: What's New in EF Core 6 bit.ly/EFCore6New
- MS Docs: Breaking Changes in EF Core 6: bit.ly/EFCore6Breaking
- Announcing EF Core 6.0 Preview 4: Performance Edition: bit.ly/EFCore6Perf
- EF Core Repository: github.com/dotnet/efcore
- Entity Framework Team Community Standups on YouTube: bit.ly/EFCoreStandup

JULIE LERMAN CONTACT INFO

TheDataFarm.com

Twitter: @JulieLerman

pluralsight.com/authors/julie-lerman

github.com/JulieLerman