ENTERPRISE SOFTWARE DEVELOPMENT

V3-2024-04

The Harmonious Dance of EF Core and SQL Server

Brisbane .NET UG

Jernej Kavka (JK)

SSW Solution Architect

@Jernej_kavka

github.com/jernejk

jkdev.me

linkedin.com/in/jernejkavka/

Brisbane Full Stack User Group

Host @ Global AI The Podcast

.NET and EF Core dev

Microsoft AI MVP and AI generalist

Join the Conversation @SSW_TV | #efcore | @jernej_kavka

NDC {Oslo}

Common mistakes in EF Core

Jernej Kavka

Microsoft AI MVP, SSW Solution Architect

Recommended watch – youtu.be/gOABWmQ5zdM

Agenda

- Logs, telemetry and tagging
- The magic of SQL (querying, joining, grouping)
- Indexing

Why logging is important?

⚠️ What happened?

A web request crashed

🕵️ Why it happened?

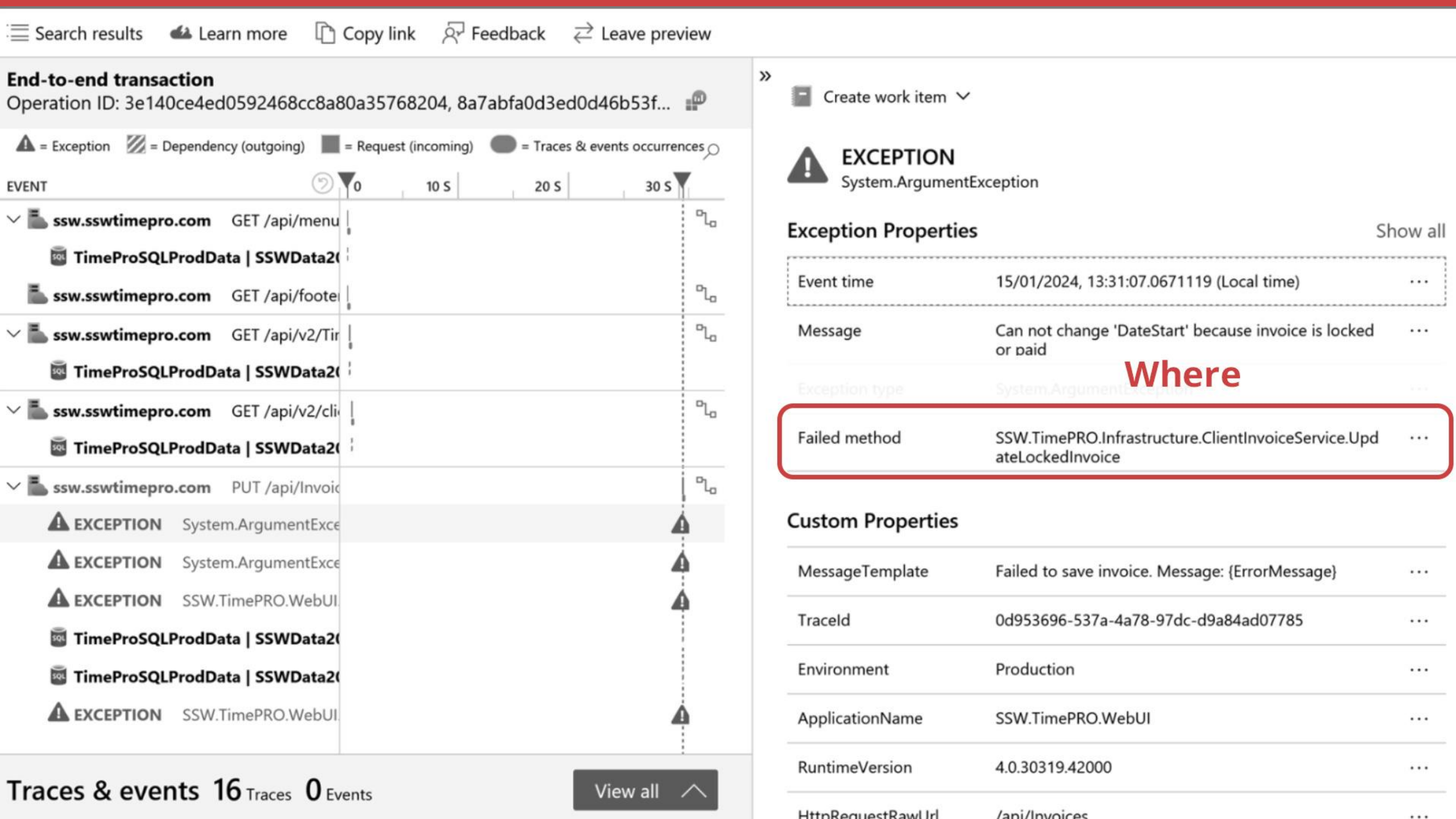
We are trying to update a locked entity

🕗 When it happened?

Tuesday morning

🗺️ Where it happened?

InvoiceService in UpdateLockedInvoice


Search results
Learn more
Copy link
Feedback
Leave preview
End-to-end transaction
Operation ID: 3e140ce4ed0592468cc8a80a35768204, 8a7abfa0d3ed0d46b53f...
= Exception
= Dependency (outgoing)
= Request (incoming)
= Traces & events occurrences
EVENT
0
10 S
20 S
30 S
ssw.sswtimepro.com GET /api/menu
TimeProSQLProdData | SSWData2(
ssw.sswtimepro.com GET /api/foote
ssw.sswtimepro.com GET /api/v2/Tir
TimeProSQLProdData | SSWData2(
ssw.sswtimepro.com GET /api/v2/cli
TimeProSQLProdData | SSWData2(
ssw.sswtimepro.com PUT /api/Invoic
EXCEPTION System.ArgumentExce
EXCEPTION System.ArgumentExce
EXCEPTION SSW.TimePRO.WebUI
TimeProSQLProdData | SSWData2(
TimeProSQLProdData | SSWData2(
EXCEPTION SSW.TimePRO.WebUI
Traces & events 16 Traces 0 Events
View all
Create work item
EXCEPTION
System.ArgumentException
Exception Properties
Show all
Event time
15/01/2024, 13:31:07.0671119 (Local time)
Message
Can not change 'DateStart' because invoice is locked or paid
Where
Failed method
SSW.TimePRO.Infrastructure.ClientInvoiceService.UpdateLockedInvoice
Custom Properties
MessageTemplate
Failed to save invoice. Message: {ErrorMessage}
TraceId
0d953696-537a-4a78-97dc-d9a84ad07785
Environment
Production
ApplicationName
SSW.TimePRO.WebUI
RuntimeVersion
4.0.30319.42000
/api/Invoices

Different stages of logging

1. Telemetry

   Application Insights

   Open Telemetry

Telemetry – Application Insights

TimeProSQLProdData | SSWData2005
SSWData2005

Dependency Properties Show all

Event time	15/01/2024, 13:31:07.0866591 (Local time)
Type	SQL
Call status	true
Duration	1.3 ms
Name	SQL: TimeProSQLProdData | SSWData2005

Command

Copy

```
TimeProSQLProdData | SSWData2005
```

ASP.NET Core | .NET | Java | Node.js | Python

Add `UseAzureMonitor()` to your application startup. Depending on your version of .NET, it is in either your `startup.cs` or `program.cs` class.

C# Copy

```
// Import the Azure.Monitor.OpenTelemetry.AspNetCore namespace.
using Azure.Monitor.OpenTelemetry.AspNetCore;

// Create a new WebApplicationBuilder instance.
var builder = WebApplication.CreateBuilder(args);

// Add the OpenTelemetry NuGet package to the application's services and conf
builder.Services.AddOpenTelemetry().UseAzureMonitor();

// Build the application.
var app = builder.Build();

// Run the application.
app.Run();
```

Copy the Connection String from your Application Insights Resource

If you don't already have one, now is a great time to **Create an Application Insights Resource**. Here's when we recommend you **create a new Application Insights**

Additional resources

Training

Module
Enable monitoring and end-to-end tracing
Training

Enable monitoring and end-to-end tracing

Documentation

Configure Azure Monitor OpenTelemetry
.NET, Java, Node.js, and Python applicatio
Azure Monitor

This article provides configuration guidance
.NET, Java, Node.js, and Python applications

Add, modify, and filter Azure Monitor
OpenTelemetry for .NET, Java, Node.js, an
Python applications - Azure Monitor

This article provides guidance on how to ac
modify, and filter OpenTelemetry for
applications using Azure Monitor.

Configuration options - Azure Monitor
Application Insights for Java - Azure Moni

configure Azu
Monitor Application Insights for Java.

Show 5 more

Telemetry – Application Insights

TimeProSQLProdData | SSWData2005
SSWData2005

Dependency Properties Show all

Event time	15/01/2024, 13:31:07.0866591 (Local time)
Type	SQL
Call status	true
Duration	1.3 ms
Name	SQL: TimeProSQLProdData | SSWData2005

Command Copy

```
TimeProSQLProdData | SSWData2005
```

TimeProSQLProdData | SSWData2005
SSWData2005

Dependency Properties Show all

Event time	14/01/2024, 15:29:16.9326191 (Local time)
Type	SQL
Result code	547
Call status	false
Duration	20.7 ms
Name	SQL: TimeProSQLProdData | SSWData2005

Command Copy

```
SET NOCOUNT ON;
INSERT INTO [ClientTag] ([CategoryID], [ContactID], [CRMClientTagGUID],
[ClientID], [DateCreated], [DateUpdated], [EmpUpdated], [Note])
VALUES (@p0, @p1, @p2, @p3, @p4, @p5, @p6, @p7);
SELECT [rowguid]
FROM [ClientTag]
WHERE @@ROWCOUNT = 1 AND [CategoryID] = @p0 AND [ContactID] = @p1;
```

Different stages of logging

1. Telemetry

   Application Insights

   Open Telemetry

2. .NET + EF Core logging

   Logger configuration

.NET + EF Core logging

Update **appsettings.json**

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information",
      "Microsoft.EntityFrameworkCore": "Information"
    }
  },
```

```
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (301ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT COUNT(*) as Count FROM [Sales]
```

.NET + EF Core logging

Update **appsettings.json**

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information",
      "Microsoft.EntityFrameworkCore": "Information"
    }
  },
```

Do you use the best trace logging library? - ssw.com.au/rules/best-trace-logging

Steps to enable OpenTelemetry.Instrumentation.EntityFrameworkCore

Step 1: Install Package

Add a reference to the `OpenTelemetry.Instrumentation.EntityFrameworkCore` package. Also, add any other instrumentations & exporters you will need.

```
dotnet add package --prerelease OpenTelemetry.Instrumentation.EntityFrameworkCore
```

Step 2: Enable EntityFrameworkCore Instrumentation at application startup

`EntityFrameworkCore` instrumentation must be enabled at application startup.

The following example demonstrates adding `EntityFrameworkCore` instrumentation to a console application. This example also sets up the OpenTelemetry Console exporter, which requires adding the package `OpenTelemetry.Exporter.Console` to the application.

```
using OpenTelemetry;
using OpenTelemetry.Trace;
public class Program
{
    public static void Main(string[] args)
    {
        using var tracerProvider = Sdk.CreateTracerProviderBuilder()
            .AddEntityFrameworkCoreInstrumentation()
            .AddConsoleExporter()
            .Build();
    }
}
```

OpenTelemetry Nuget – EF Core extension (currently in preview)

For an ASP.NET Core application, adding instrumentation is typically done in the `ConfigureServices` of your `Startup` class. Refer to

Different stages of logging

1. Telemetry

   Application Insights

   Open Telemetry

2. .NET + EF Core logging

   Logger configuration

3. Additional context in code to SQL query

EF Core TagWith (EF Core 2.2+)

TagWith adds SQL comments to EF Core query

```
var count = _dbContext.Sales
    .TagWith("ExamplesCountController-BaseCase")
    .Count();
```

```
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (220ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      -- ExamplesCountController-BestCase

      SELECT COUNT(*)
      FROM [Sales] AS [s]
```

TagWithCallSite (EF Core 6+)

Automatically adds file path and line number

```
var count = _dbContext.Sales
    .TagWithCallSite()
    .Count();
```

```
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (279ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      -- File: D:\DataJK\git\EFCoreSamples.StabilityAndPerformance\EFCoreSamples.StabilityA
\ExamplesCountController.cs:85

      SELECT COUNT(*)
      FROM [Sales] AS [s]
```

Custom TagWith

We can create our own TagWith

Useful data: class name, function, purpose

```
var count = _dbContext.Sales
    .TagWithContext("CountingSales")
    .Count();
```

```
-- ExamplesCountController-BestCase-CountingSales

SELECT COUNT(*)
FROM [Sales] AS [s]
```

```
– references | Jernej Kavka (JK), 111 days ago | 1 author, 1 change
public static class WithTagsExtensions
{
    – references | Jernej Kavka (JK), 111 days ago | 1 author, 1 change
    public static IQueryable<T> TagWithContext<T>(this IQueryable<T> queryable, string message = "",
        [CallerFilePath] string callerFileName = "", [CallerMemberName] string callerName = "")
    {
        string logScopeName = GenerateLogScopeName(message, callerFileName, callerName);
        return queryable.TagWith(logScopeName);
    }

    – references | 0 changes | 0 authors, 0 changes
    private static string GenerateLogScopeName(string? message = null,
        string callerFileName = "",
        string callerName = "")
    {
        if (!string.IsNullOrWhiteSpace(message))
        {
            message = "-" + message;
        }

        string className = Path.GetFileNameWithoutExtension(callerFileName);
        return className + "-" + callerName + message;
    }
}
```

ssw.com.au/rules/use-tagwith

Different stages of logging

1. Telemetry

   Application Insights

   Open Telemetry

2. .NET + EF Core logging

   Logger configuration

3. Additional context in code to SQL query

Once logs work well

Filter to a specific component and call

All [Component | Call]

09:23:37.878 Trace **Severity level:** Information, **Message:** Executed DbC

```
-- TimesheetService-SaveTimesheet-EmpTimeDayAny

SELECT CASE
    WHEN EXISTS (
        SELECT 1
        FROM [EmpTimeDay] AS [e]
        WHERE ([e].[EmpID] = @__empID_0) AND ([e].
    ELSE CAST(0 AS bit)
END
```

09:23:37.878 Trace **Severity level:** Information, **Message:** Executed DbC

```
-- TimesheetService-SaveTimesheet-Client

SELECT TOP(1) [c].[ClientID]
FROM [Client] AS [c]
WHERE [c].[ClientID] = @__clientId_0
```

09:23:37.878 Trace **Severity level:** Information, **Message:** Executed DbC

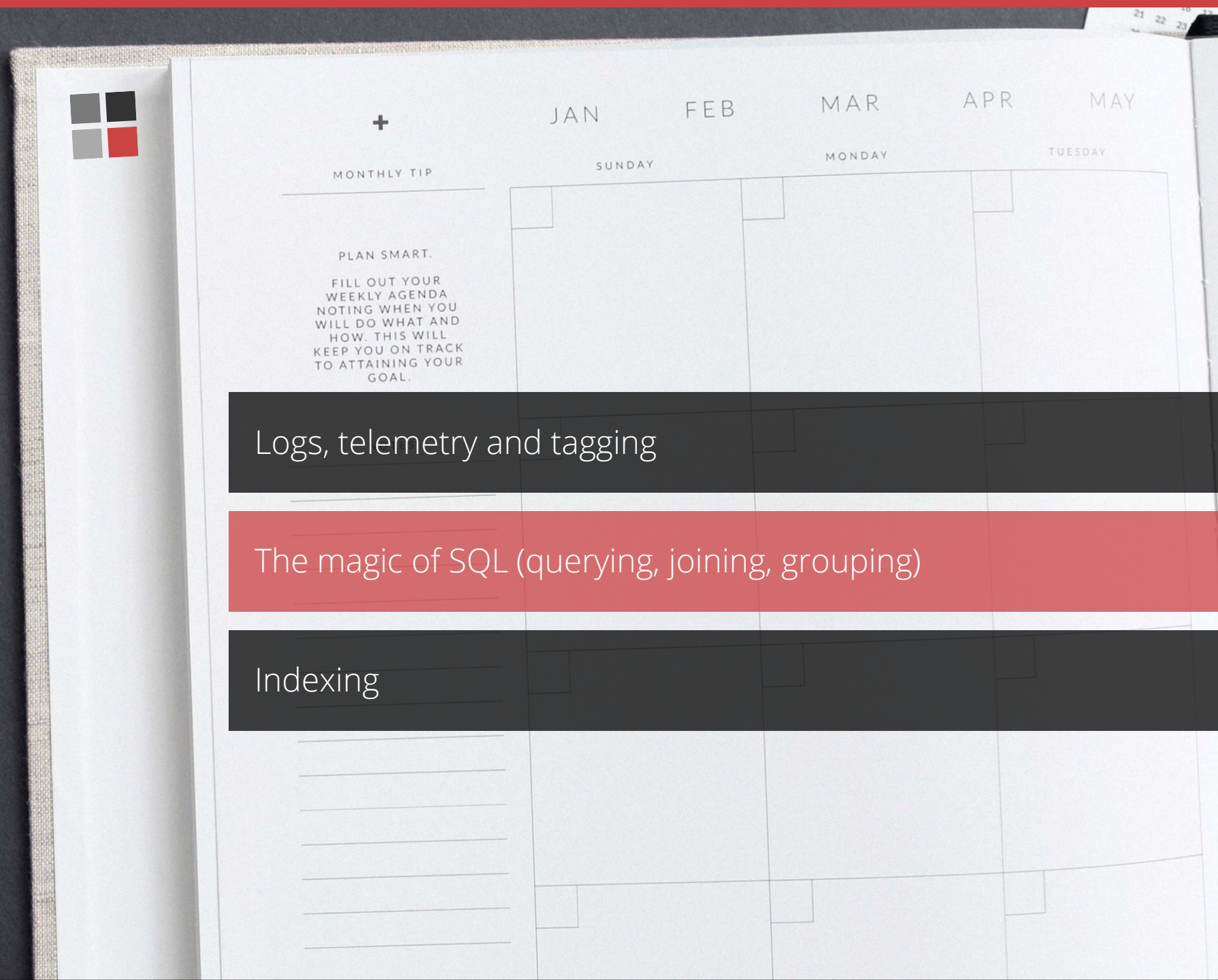
```
-- TimesheetService-SaveTimesheet-TimesheetBillabl

SELECT TOP(1) [e].[BillableID]
FROM [EmpTime] AS [e]
WHERE [e].[TimeID] = @__timeID_0
```

```
bool timesheetDayExists = await _tenantDbContext.TimesheetDays
    .TagWithContext("EmpTimeDayAny")
    .AnyAsync(x ⇒ x.EmpID == empID && x.DateCreated == date, ct
```

```
clientId = await _tenantDbContext.Clients
    .TagWithContext("Client")
    .Select(x ⇒ x.ClientID)
    .FirstOrDefaultAsync(x ⇒ x == clientId, ct);
```

```
string timesheetBillableType = await _tenantDbContext.Timesheets
    .TagWithContext("TimesheetBillableType")
    .Where(x ⇒ x.TimeID == timeID)
    .Select(x ⇒ x.BillableID)
    .FirstOrDefaultAsync(ct);
```

Agenda



- Logs, telemetry and tagging
- The magic of SQL (querying, joining, grouping)
- Indexing

Data representation

SQL Server

```
SELECT * FROM Author WHERE ID = 1
```

Id	Name	Country
1	William Shakespeare	UK

EF Core

```
var simple = dbContext.Author
    .TagWith("SimpleAuthorQuery")
    .Where(a => a.Id == selectedAuthorId)
    .FirstOrDefault();
```

```
SELECT TOP(1) [a].[Id], [a].[Country], [a].[Name]
FROM [Author] AS [a]
WHERE [a].[Id] = @__selectedAuthorId_0
```

SQL: Joining table #1

```
SELECT a.Id as [AuthorId], a.Name, p.Id as [PostId], p.Title, p.Content
    FROM Author a
    LEFT JOIN Post p ON a.Id = p.AuthorId
```

lts Messages

AuthorId	Name	PostId	Title	Content
1	William Shakespeare	1	The futility of love	In the tragic play of Romeo …
1	William Shakespeare	2	Meaning of Romeo's love	What is the true meaning of …

SQL: Joining table #2

```
SELECT a.Id as [AuthorId], a.Name, p.Id as [PostId], p.Title, p.Content, t.Name
    FROM Author a
    LEFT JOIN Post p ON a.Id = p.AuthorId
    LEFT JOIN PostTag pt ON pt.PostsId = p.Id
    LEFT JOIN Tag t ON t.Id = pt.TagsId
```

lts Messages

AuthorId	Name	PostId	Title	Content	Name
1	William Shakespeare	1	The futility of love	In the tragic play of Romeo …	love
1	William Shakespeare	1	The futility of love	In the tragic play of Romeo …	futility
1	William Shakespeare	1	The futility of love	In the tragic play of Romeo …	tragedy
1	William Shakespeare	2	Meaning of Romeo's love	What is the true meaning of …	character
1	William Shakespeare	2	Meaning of Romeo's love	What is the true meaning of …	analysis

⚠️ Author and blog posts are duplicated!

EF Core: Joining table #1

```
var simple = dbContext.Author
    .TagWith("SimpleAuthorQuery")
    .Where(a ⇒ a.Id == selectedAuthorId)
    .FirstOrDefault();
```

```
-- SimpleAuthorQuery

SELECT TOP(1) [a].[Id], [a].[Country], [a].[Name]
FROM [Author] AS [a]
WHERE [a].[Id] = @__selectedAuthorId_0
```

EF Core: Joining table #2

```
var authorWithPosts = dbContext.Author
    .TagWith("AuthorWithPostsQuery")
    .Where(a => a.Id == selectedAuthorId)
    .Select(a => new
    {
        a.Id,
        a.Name,
        Posts = a.Posts.Select(p => new
        {
            p.Id,
            p.Title
        })
    })
    .FirstOrDefault();
```

```
-- AuthorWithPostsQuery

SELECT [t].[Id], [t].[Name], [p].[Id], [p].[Title]
FROM (
    SELECT TOP(1) [a].[Id], [a].[Name]
    FROM [Author] AS [a]
    WHERE [a].[Id] = @__selectedAuthorId_0
) AS [t]
LEFT JOIN [Post] AS [p] ON [t].[Id] = [p].[AuthorId]
ORDER BY [t].[Id]
```

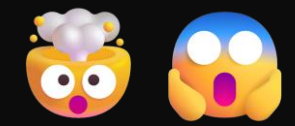
⚡ Prefilters tables before joining!

EF Core: Joining table #3

```
var authorWithAllData = dbContext.Author
    .TagWith("FetchWillsPostsAndTags")
    .Where(a => a.Id == selectedAuthorId)
    .Select(a => new
    {
        a.Id,
        a.Name,
        Posts = a.Posts.Select(p => new
        {
            p.Id,
            p.Title,
            Tags = p.Tags.Select(t => new
            {
                t.Name
            })
        })

    })
    .FirstOrDefault();
```

EF Core: Joining table #4

```
-- FetchWillsPostsAndTags

SELECT [t].[Id], [t].[Name], [t1].[Id], [t1].[Title], [t1].[Name], [t1].[PostsId], [t1].[TagsId], [t1].[Id0]
FROM (
    SELECT TOP(1) [a].[Id], [a].[Name]
    FROM [Author] AS [a]
    WHERE [a].[Id] = @__selectedAuthorId_0
) AS [t]
LEFT JOIN (
    SELECT [p].[Id], [p].[Title], [t0].[Name], [t0].[PostsId], [t0].[TagsId], [t0].[Id] AS [Id0], [p].[AuthorId]
    FROM [Post] AS [p]
    LEFT JOIN (
        SELECT [t2].[Name], [p0].[PostsId], [p0].[TagsId], [t2].[Id]
        FROM [PostTag] AS [p0]
        INNER JOIN [Tag] AS [t2] ON [p0].[TagsId] = [t2].[Id]
    ) AS [t0] ON [p].[Id] = [t0].[PostsId]
) AS [t1] ON [t].[Id] = [t1].[AuthorId]
ORDER BY [t].[Id], [t1].[Id], [t1].[PostsId], [t1].[TagsId]
```



Post tags are linked by a link table 💡

EF Core: Joining table #5

```
var authorWithAllDataInclude = dbContext.Author
    .TagWith("FetchWillsPostsAndTagsInclude")
    .Include(x => x.Posts)
        .ThenInclude(x => x.Tags)
    .Where(a => a.Id == selectedAuthorId)
    .Select(a => new
    {
        a.Id,
        a.Name,
        Posts = a.Posts.Select(p => new
        {
            p.Id,
            p.Title,
            Tags = p.Tags.Select(t => new
            {
                t.Name
            })
        })

    })
    .FirstOrDefault();
```

EF Core: Joining table #6

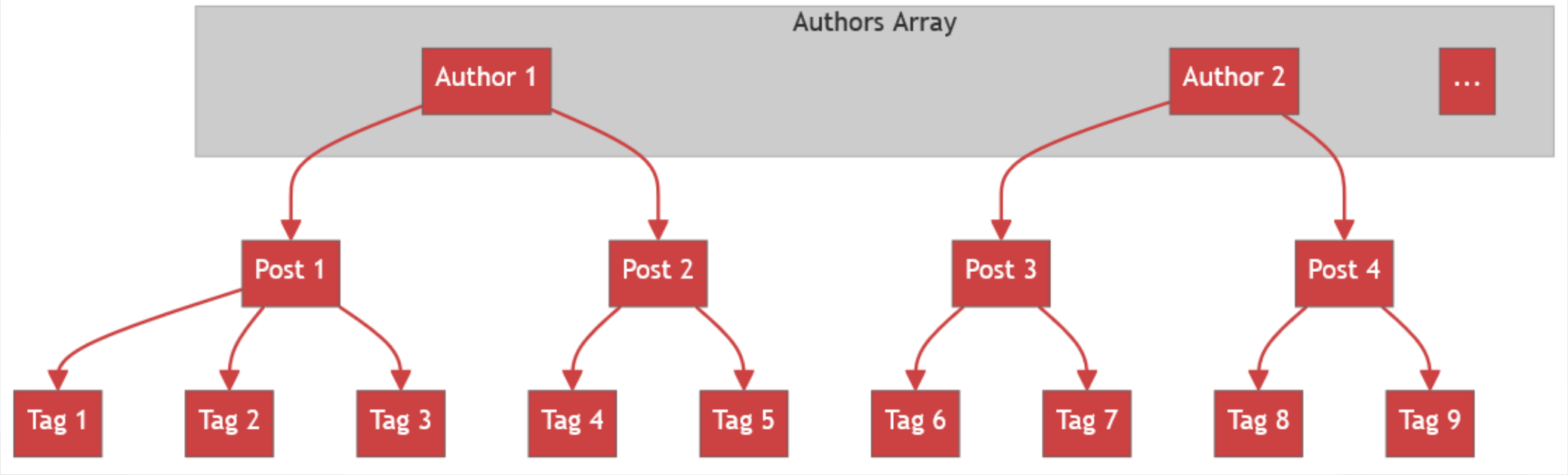
```
-- FetchWillsPostsAndTagsInclude

SELECT [t].[Id], [t].[Name], [t1].[Id], [t1].[Title], [t1].[Name], [t1].[PostsId], [t1].[TagsId], [t1].[Id0]
FROM (
    SELECT TOP(1) [a].[Id], [a].[Name]
    FROM [Author] AS [a]
    WHERE [a].[Id] = @__selectedAuthorId_0
) AS [t]
LEFT JOIN (
    SELECT [p].[Id], [p].[Title], [t0].[Name], [t0].[PostsId], [t0].[TagsId], [t0].[Id] AS [Id0], [p].[AuthorId]
    FROM [Post] AS [p]
    LEFT JOIN (
        SELECT [t2].[Name], [p0].[PostsId], [p0].[TagsId], [t2].[Id]
        FROM [PostTag] AS [p0]
        INNER JOIN [Tag] AS [t2] ON [p0].[TagsId] = [t2].[Id]
    ) AS [t0] ON [p].[Id] = [t0].[PostsId]
) AS [t1] ON [t].[Id] = [t1].[AuthorId]
ORDER BY [t].[Id], [t1].[Id], [t1].[PostsId], [t1].[TagsId]
```

It's identical! 

EF Core: Joining table #7

Maps flat table into

AuthorId	Name	PostId	Title	Content	Name
1	William Shakespeare	1	The futility of love	In the tragic play of Romeo …	love
1	William Shakespeare	1	The futility of love	In the tragic play of Romeo …	futility
1	William Shakespeare	1	The futility of love	In the tragic play of Romeo …	tragedy
1	William Shakespeare	2	Meaning of Romeo's love	What is the true meaning of …	character
1	William Shakespeare	2	Meaning of Romeo's love	What is the true meaning of …	analysis



Lists of graph structures

⚠️ What if this is 1MB of content?

What about data redundancy?

🤔

AuthorId	Name	PostId	Title	Content	Name
1	William Shakespeare	1	The futility of love	In the tragic play of Romeo …	love
1	William Shakespeare	1	The futility of love	In the tragic play of Romeo …	futility
1	William Shakespeare	1	The futility of love	In the tragic play of Romeo …	tragedy
1	William Shakespeare	2	Meaning of Romeo's love	What is the true meaning of …	character
1	William Shakespeare	2	Meaning of Romeo's love	What is the true meaning of …	analysis

SQL: Query Splitting #1

```
SELECT [Id], [Name]
  FROM [BlogTestDB].[dbo].[Author]
  where [Id] = 1

SELECT [Id], [Content]
      FROM Post
      WHERE [AuthorId] = 1

SELECT [p].[Id], [t].[Id], [t].[Name]
      FROM Post AS [p]
      INNER JOIN PostTag AS [pt] ON [p].[Id] = [pt].[PostsId]
      INNER JOIN Tag AS [t] ON [pt].[TagsId] = [pt].[TagsId]
```

SQL: Query Splitting #2

✅ Usually easy

✅ No data redundancy

❌ Needs manual mapping on .NET end

❌ May not be faster than single query! ⚠️

EF Core: Query Splitting #1

```
var authorWithAllDataSplit = dbContext.Author
    .TagWith("FetchWillsPostsAndTags-Split")
    .AsSplitQuery()
    .Where(a => a.Id == selectedAuthorId)
    .Select(a => new
    {
        AuthorId = a.Id,
        a.Name,
        Posts = a.Posts.Select(p => new
        {
            PostId = p.Id,
            p.Title,
            Tags = p.Tags.Select(t => new
            {
                t.Name
            })
        })

    })
    .FirstOrDefault();
```

EF Core: Query Splitting #2

```
-- FetchWillsPostsAndTags-Split

SELECT TOP(1) [a].[Id], [a].[Name]
FROM [Author] AS [a]
WHERE [a].[Id] = @__selectedAuthorId_0
ORDER BY [a].[Id]
```

Author

```
-- FetchWillsPostsAndTags-Split

SELECT [p].[Id], [p].[Title], [t].[Id]
FROM (
    SELECT TOP(1) [a].[Id]
    FROM [Author] AS [a]
    WHERE [a].[Id] = @__selectedAuthorId_0
) AS [t]
INNER JOIN [Post] AS [p] ON [t].[Id] = [p].[AuthorId]
ORDER BY [t].[Id], [p].[Id]
```

Posts

```
-- FetchWillsPostsAndTags-Split

SELECT [t0].[Name], [t].[Id], [p].[Id]
FROM (
    SELECT TOP(1) [a].[Id]
    FROM [Author] AS [a]
    WHERE [a].[Id] = @__selectedAuthorId_0
) AS [t]
INNER JOIN [Post] AS [p] ON [t].[Id] = [p].[AuthorId]
INNER JOIN (
    SELECT [t1].[Name], [p0].[PostsId]
    FROM [PostTag] AS [p0]
    INNER JOIN [Tag] AS [t1] ON [p0].[TagsId] = [t1].[Id]
) AS [t0] ON [p].[Id] = [t0].[PostsId]
ORDER BY [t].[Id], [p].[Id]
```

Post tags

EF Core: Query Splitting #3

- ✅ 1 line code change
- ✅ Can drastically improve performance
- ✅ Does a good job in splitting queries
- ❌ Performance depends on many factors!
- ❌ In most cases, a bit worse than non-split query

SQL GROUP BY vs EF Core GroupBy

- Flat table
- Summarizes each group

- Complex structure
- Groups result

```
SELECT Country, COUNT(*)
    FROM Author
    GROUP BY Country
```

```
var authorsByCountry = dbContext.Author
    .GroupBy(x ⇒ x.Country);
```

```
SELECT [a].[Country], [a].[Id], [a].[Name]
FROM [Author] AS [a]
ORDER BY [a].[Country]
```

	Country	(No column name)
1	UK	1

EF Core: GroupBy

```
var authorsByCountry = dbContext.Author
    .GroupBy(x ⇒ x.Country);

foreach (var country in authorsByCountry)
{
    Console.WriteLine($"Country: {country.Key}");

    foreach (var author in country)
    {
        Console.WriteLine($"Author: {author.Name}");
    }
}
```

```
SELECT [a].[Country], [a].[Id], [a].[Name]
FROM [Author] AS [a]
ORDER BY [a].[Country]
```

```
Country: UK
Author: William Shakespeare
```

EF Core: GROUP BY like SQL

```
var authorCountByCountry = dbContext.Author
    .TagWith("GroupByLikeSql")
    .GroupBy(x ⇒ x.Country)
    .Select(x ⇒ new
    {
        Country = x.Key,
        Count = x.Count()
    })
    .ToList();
```

```
-- GroupByLikeSql

SELECT [a].[Country], COUNT(*) AS [Count]
FROM [Author] AS [a]
GROUP BY [a].[Country]
```

Teaser: Different ways of querying #1

How would you get

- TOP 5 expensive products
- TOP 5 cheapest products

as 1 query?

Teaser: Different ways of querying #2

```sql
SELECT [ProductID],[Name],[Price], 1 AS [Expensive]
FROM (
    SELECT TOP (5) [ProductID],[Name] ,[Price]
    FROM [SalesDB].[dbo].[Products]
    ORDER BY Price desc
) AS expensive
UNION ALL
SELECT [ProductID],[Name],[Price], 0 AS [Expensive]
FROM (
    SELECT TOP (5) [ProductID],[Name] ,[Price]
    FROM [SalesDB].[dbo].[Products]
    WHERE Price > 0
    ORDER BY Price
) AS cheap
ORDER BY Price DESC
```

Teaser: Different ways of querying #3

```
var topProducts = await _dbContext.Products
    .OrderByDescending(x => x.Price)
    .Select(x => new { x.Name, x.Price, Expensive = true })
    .Take(5)
    .Concat(_dbContext.Products
                .Where(x => x.Price > 0)
                .Select(x => new { x.Name, x.Price, Expensive = false })
                .OrderBy(x => x.Price)
                .Take(5))
    .OrderBy(x => x.Price)
    .ToListAsync(ct);
```

Teaser: Different ways of querying #4

How would you get title of books and posts?

Teaser: Different ways of querying #5

```
SELECT [b].[Name] AS [Title], 'Book' AS [Type]
    FROM Books AS [b]
    WHERE AuthorId = 1
UNION ALL
SELECT [p].[Title], 'Post' AS [Type]
    FROM Post AS [p]
    WHERE AuthorId = 1
```

Teaser: Different ways of querying #6

```
var items = dbContext.Post
    .Where(x => x.AuthorId == selectedAuthorId)
    .Select(x => new { Name = x.Title, Type = "Post" })
    .Concat(dbContext.Books
        .Where(x => x.AuthorId == selectedAuthorId)
        .Select(x => new { x.Name, Type = "Book" }))
    .ToList();
```

EF Core LINQ to SQL

`Join` -> `INNER JOIN`

`Union` -> `UNION`

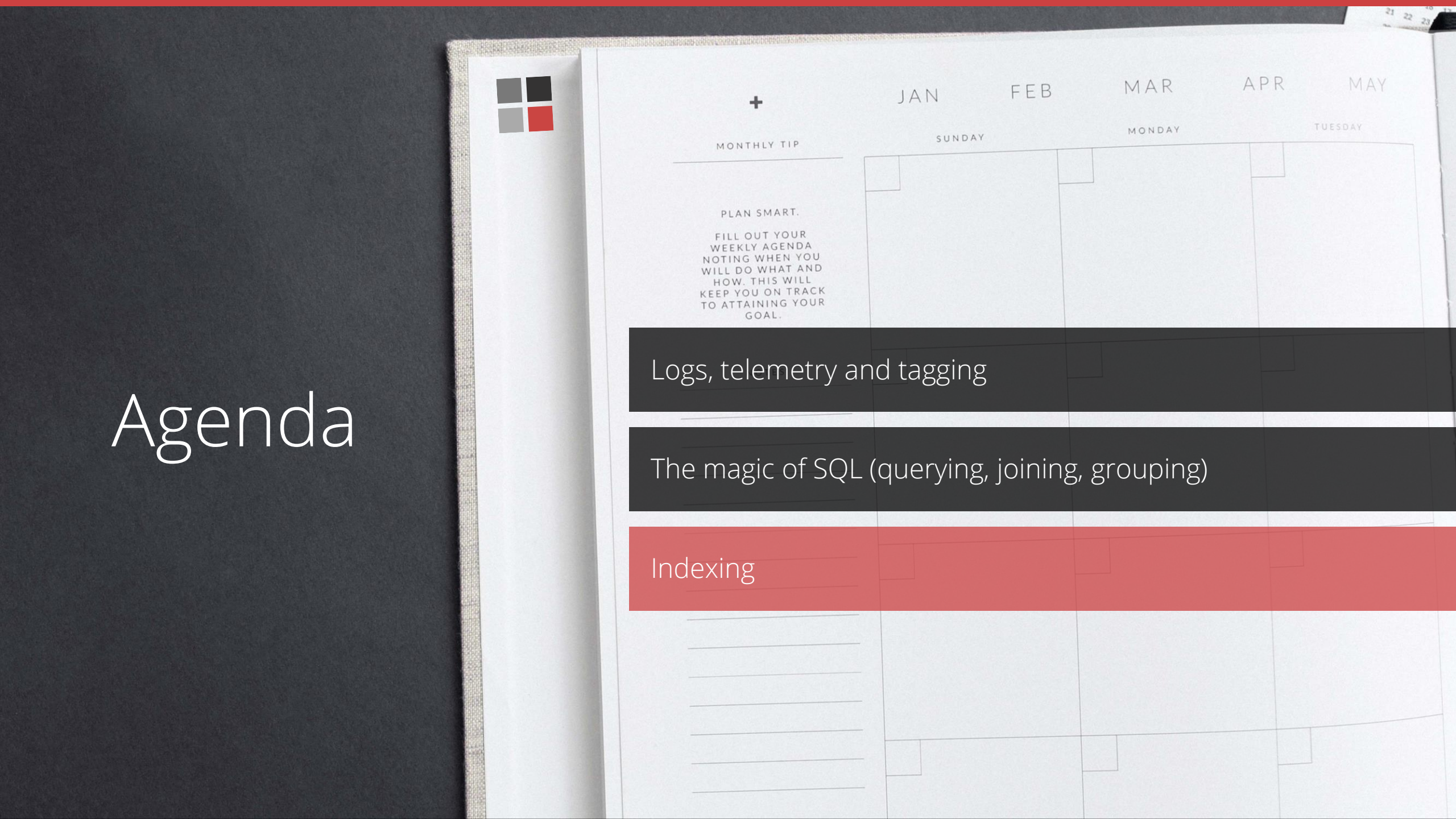
`Concat` -> `UNION ALL`

`Intersect` -> `INTERSECT`

`Except` -> `EXCEPT`

`GroupJoin` -> in-line `SELECT` or `LEFT JOIN` or `OUTER APPLY`

Agenda



Logs, telemetry and tagging

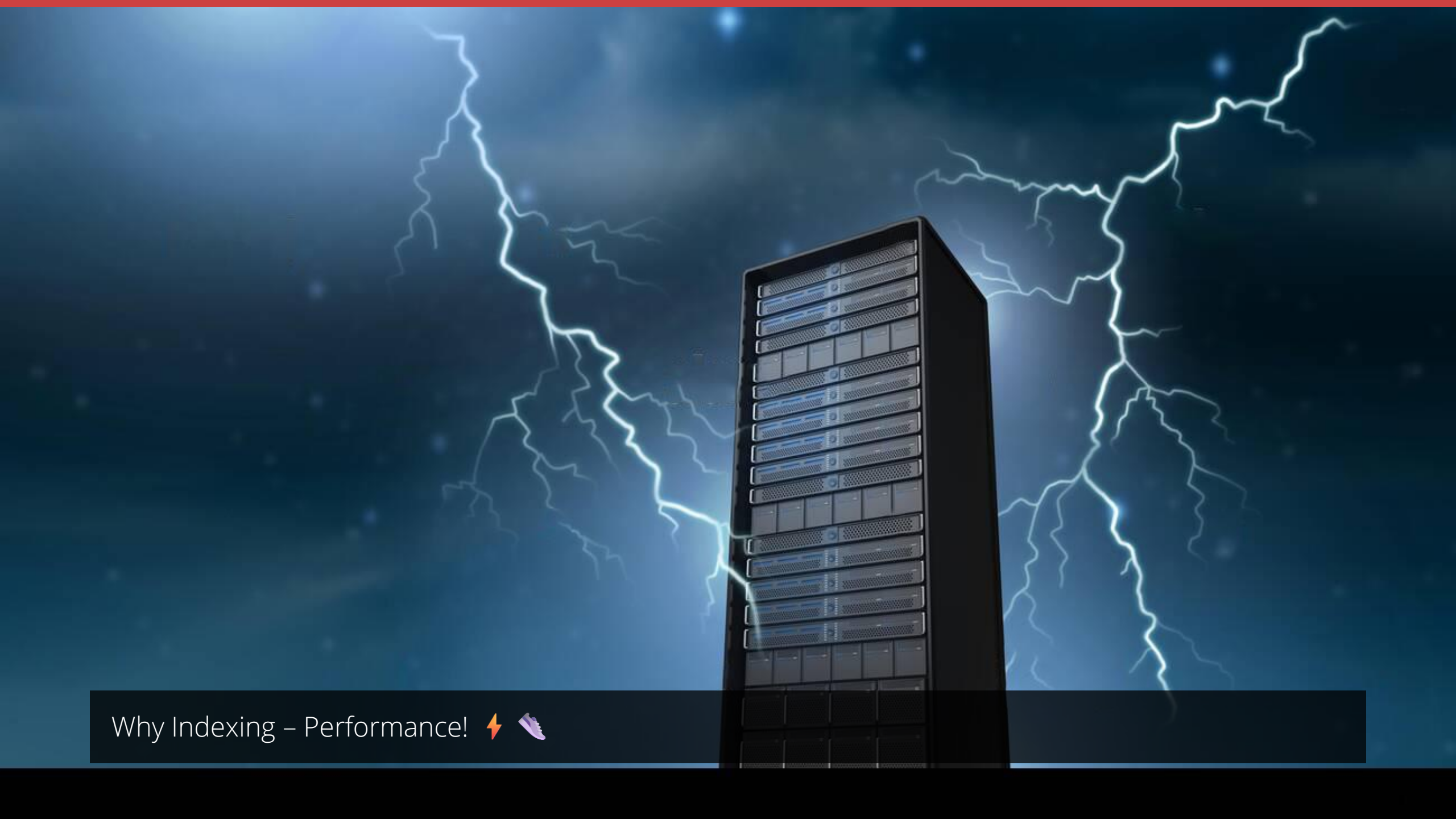
The magic of SQL (querying, joining, grouping)

Indexing

HOW TO PLAN FOR DATABASE PERFORMANCE SUCCESS

with Bryden Oliver

Recommended watch – How to Plan for Database Performance | Bryden Oliver
http://bit.ly/4b2Ng4C


Why Indexing – Performance!

SQL Server indexing

🔑 Clustered index

Only 1 per table

Defines how data is **physically stored on disk**

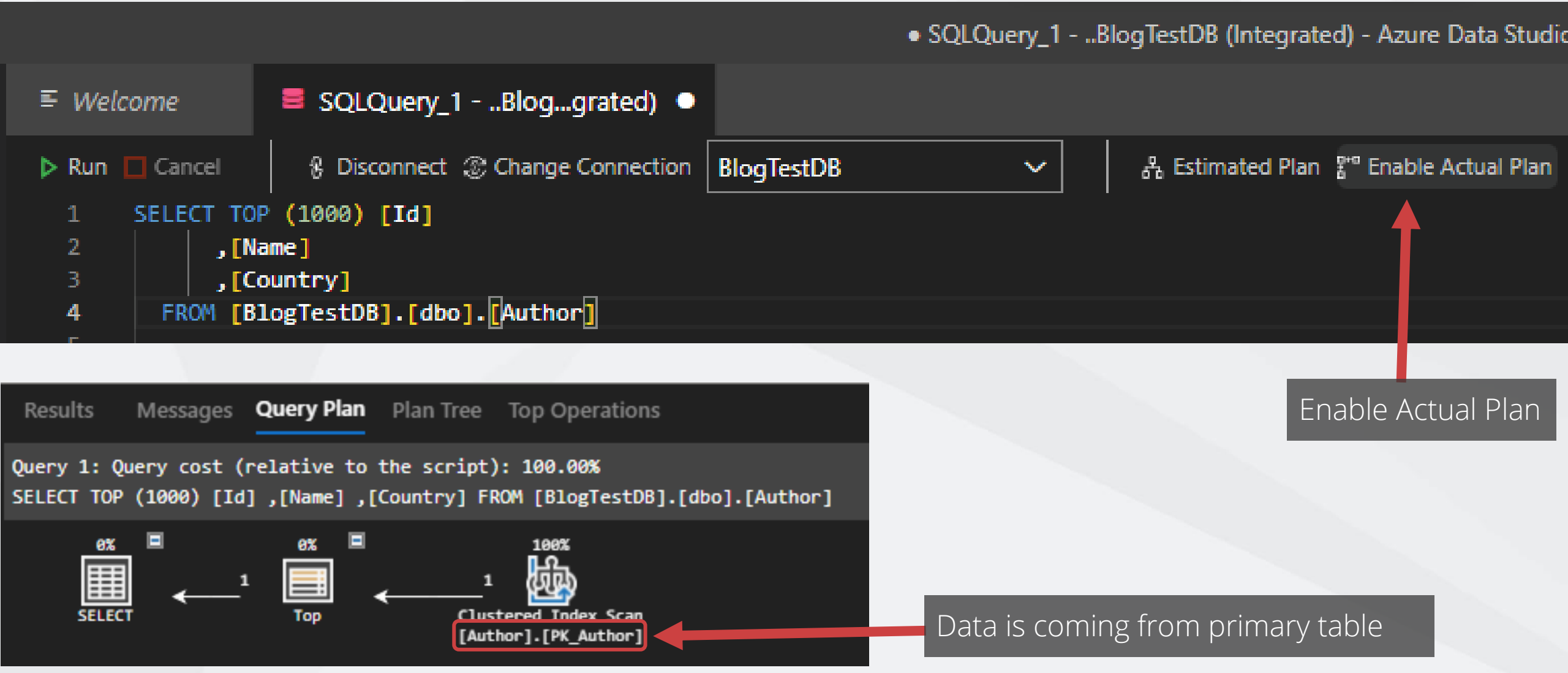
EF Core uses primary key by default

🗂️ Unclustered index

Multiples per table

Additional table per unclustered index (sorted by clustered index)

How to see what index is used?



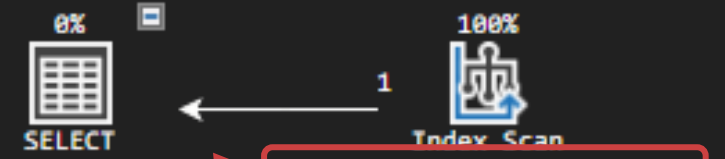
Index: Add simple index

```
public class BlogDbContext : DbContext
{
    – references
    public BlogDbContext(DbContextOptions<BlogDbContext> options)
        : base(options) { }

    – references
    public DbSet<Author> Author { get; set; }
Other DbSets

    – references
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        modelBuilder.Entity<Author>()
            .HasIndex(b => b.Country);
    }
}
```

```
1   select Country from Author
```

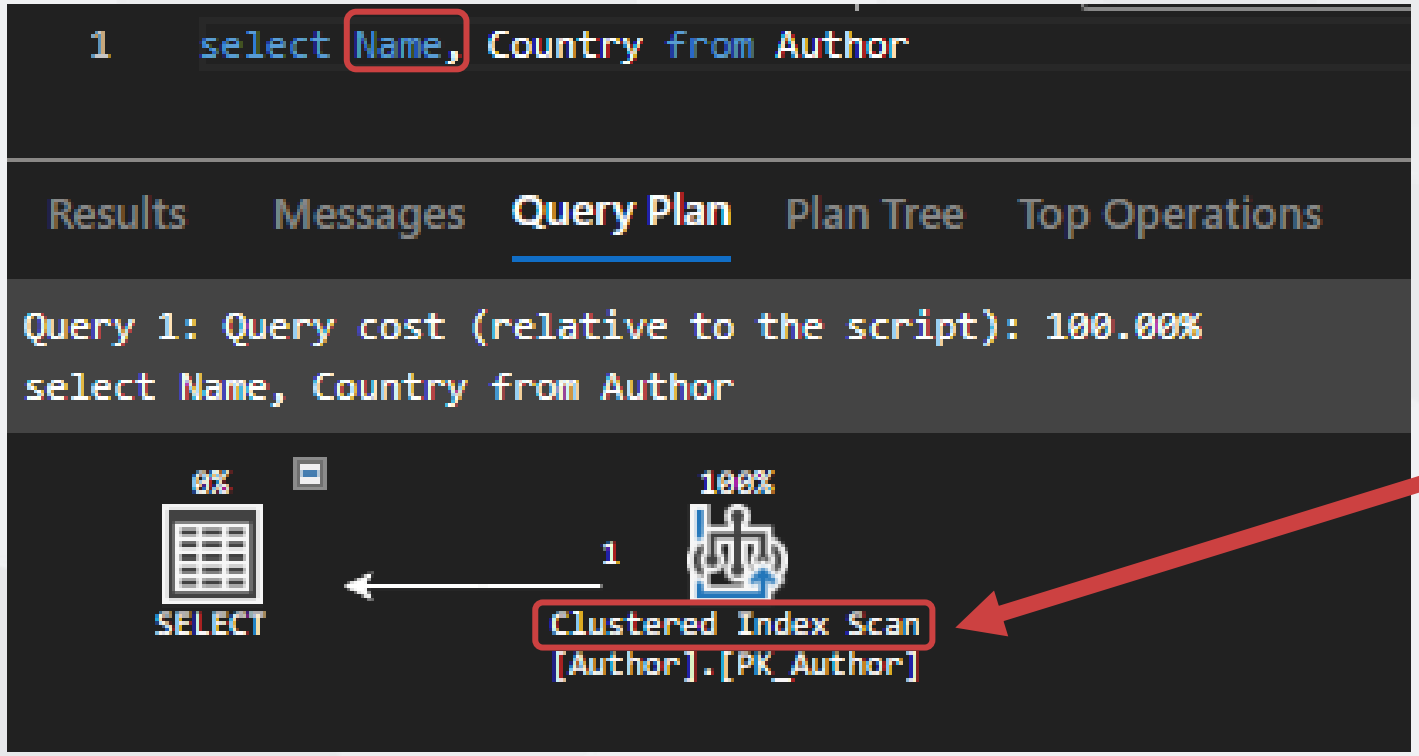
Results | Messages | **Query Plan** | Plan Tree | Top Operations

Query 1: Query cost (relative to the script): 100.00%
select Country from Author



Index: Querying on SQL Server

SQL Server chooses the index



SQL Server decided to ignore Country Index

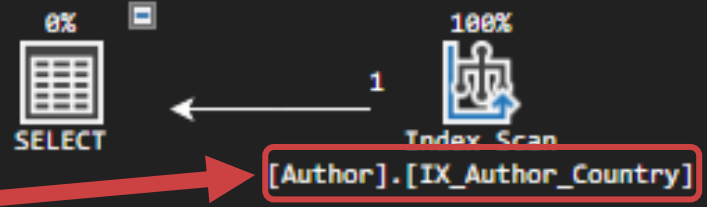
Index: Add non-Indexed column

```
public class BlogDbContext : DbContext
{
    – references
    public BlogDbContext(DbContextOptions<BlogDbContext> options)
        : base(options) { }

    – references
    public DbSet<Author> Author { get; set; }
Other DbSets

    – references
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        modelBuilder.Entity<Author>()
            .HasIndex(b ⇒ b.Country)
            .IncludeProperties(p ⇒ p.Name);
    }
}
```

```
1   select Name, Country from Author
```

Results | Messages | **Query Plan** | Plan Tree | Top Operations

Query 1: Query cost (relative to the script): 100.00%
select Name, Country from Author



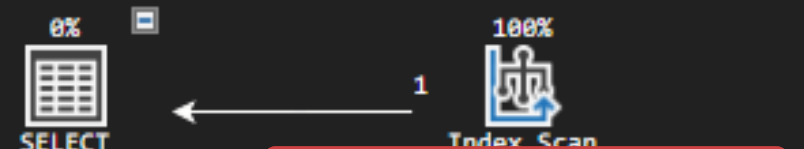
Index: Add Composite Index

```
public class BlogDbContext : DbContext
{
    – references
    public BlogDbContext(DbContextOptions<BlogDbContext> options)
        : base(options) { }

    – references
    public DbSet<Author> Author { get; set; }
Other DbSets

    – references
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);
        modelBuilder.Entity<Author>()
            .HasIndex(b => new { b.Country, b.Name });
    }
}
```

```
1   select Name, Country from Author
2
```

Results | Messages | **Query Plan** | Plan Tree | Top Operations

Query 1: Query cost (relative to the script): 100.00%
select Name, Country from Author



Microsoft Azure

Search resources, services, and docs (G+/)

Dashboard > mySampleDatabase (mynewserver20190926/mySampleDatabase)

mySampleDatabase (mynewserver20190926/mySampleDatabase) - Performance recommendations

SQL database

Search (Ctrl+/)

- Locks
- Export template

Integrations

- Stream analytics (preview)

Security

- Advanced data security
- Auditing
- Dynamic Data Masking
- Transparent data encryption

Intelligent Performance

- Performance overview
- Performance recommendati...
- Query Performance Insight
- Automatic tuning

Monitoring

- Alerts

Automate | View discarded | Getting started | Feedback

Click here to see recommended actions for parent server

Recommendations

Action	Recommendation description
Parameterize queries	Scope: Entire database Reason: Non-parameterized queries are causing performance issues

Tuning history

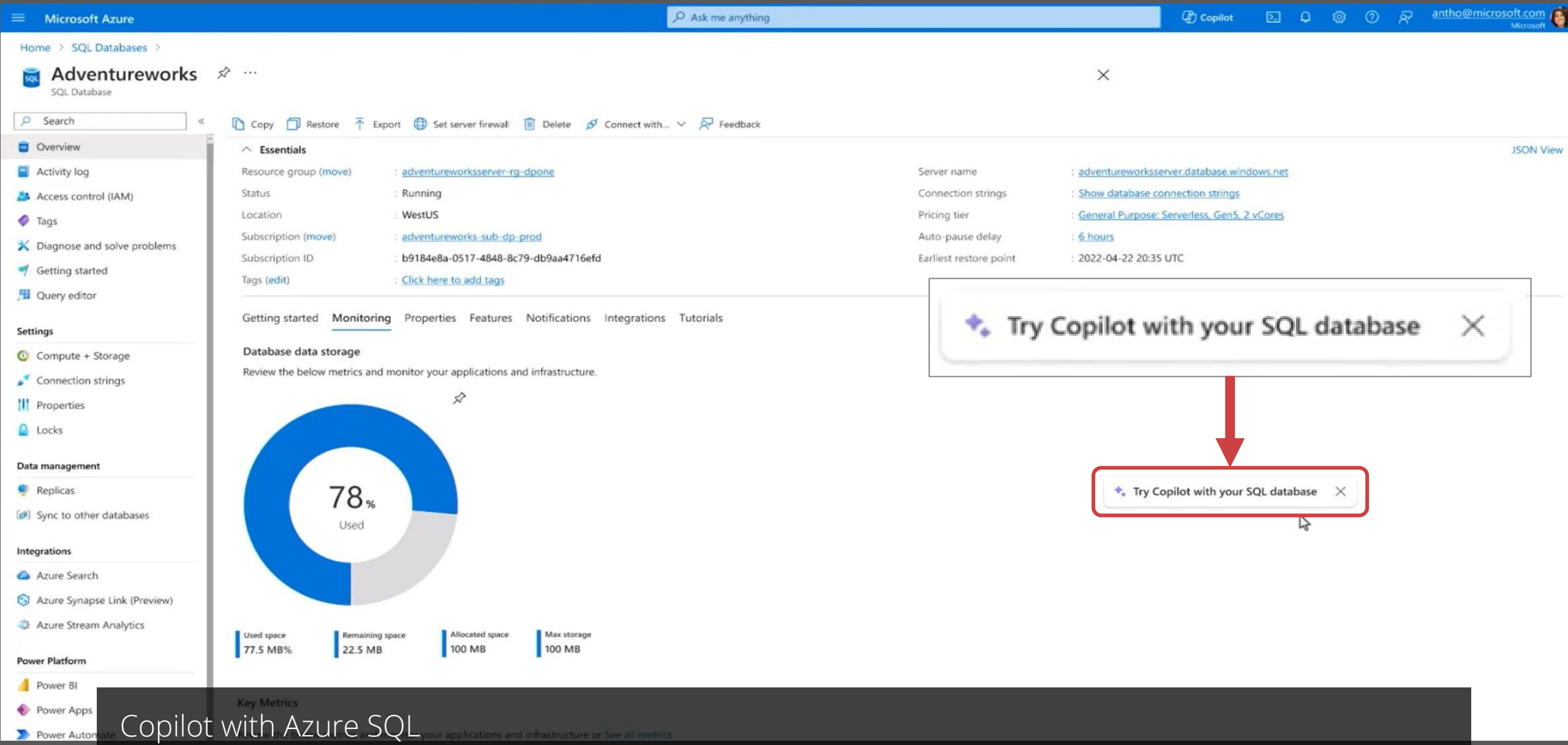
Action	Recommendation description	Status
Create index Initiated by: User	Table: [DataPoints] Indexed columns:[Name],[Money]	Succ
Drop index Initiated by: User	Index name: MyIndex321 Reason: Duplicate index	Succ
Drop index Initiated by: System	Index name: IX_FF Reason: Duplicate index	Succ

Performance Recommendation – Azure SQL Server is a great source of performance tips

Join the Conversation @SSW_TV | #efcore | @jernej_kavka

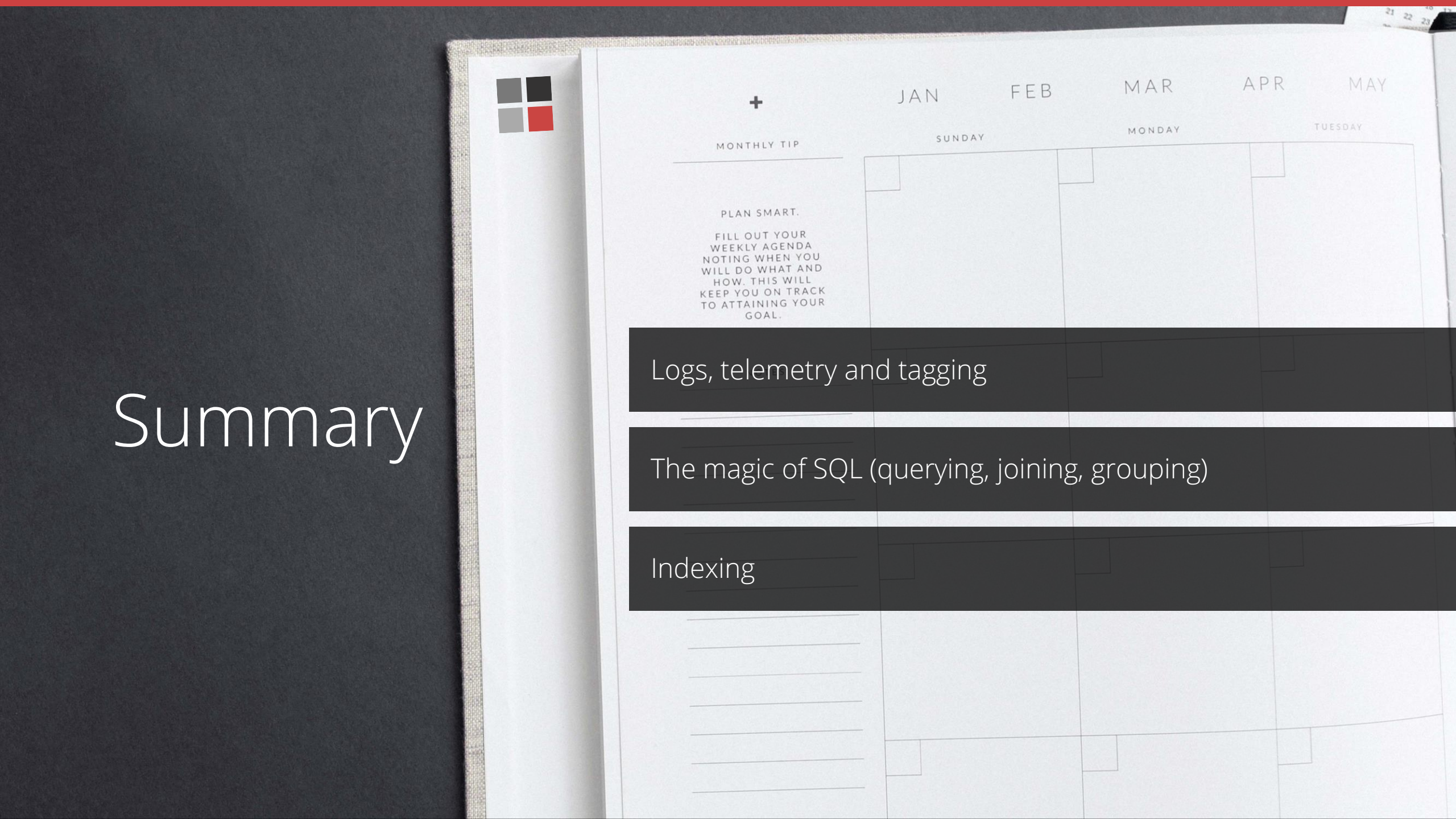
```sql
SELECT
    qs_cpu.total_worker_time / 1000 AS total_cpu_time_ms,
    q.[text],
    p.query_plan,
    qs_cpu.execution_count,
    q.dbid,
    q.objectid,
    q.encrypted AS text_encrypted
FROM
    (SELECT TOP 50 qs.plan_handle,
     qs.total_worker_time,
     qs.execution_count FROM sys.dm_exec_query_stats qs ORDER BY qs.total_worker_time DESC) AS qs_cpu
CROSS APPLY sys.dm_exec_sql_text(plan_handle) AS q
CROSS APPLY sys.dm_exec_query_plan(plan_handle) p
WHERE p.query_plan.exist('declare namespace
       qplan = "http://schemas.microsoft.com/sqlserver/2004/07/showplan";
       //qplan:MissingIndexes')=1
```

SSW Rules - What to do about SQL Server CPU Pressure?
ssw.com.au/rules/rules-to-better-sql-databases-performance/#identify-missing-indexes



Copilot with Azure SQL

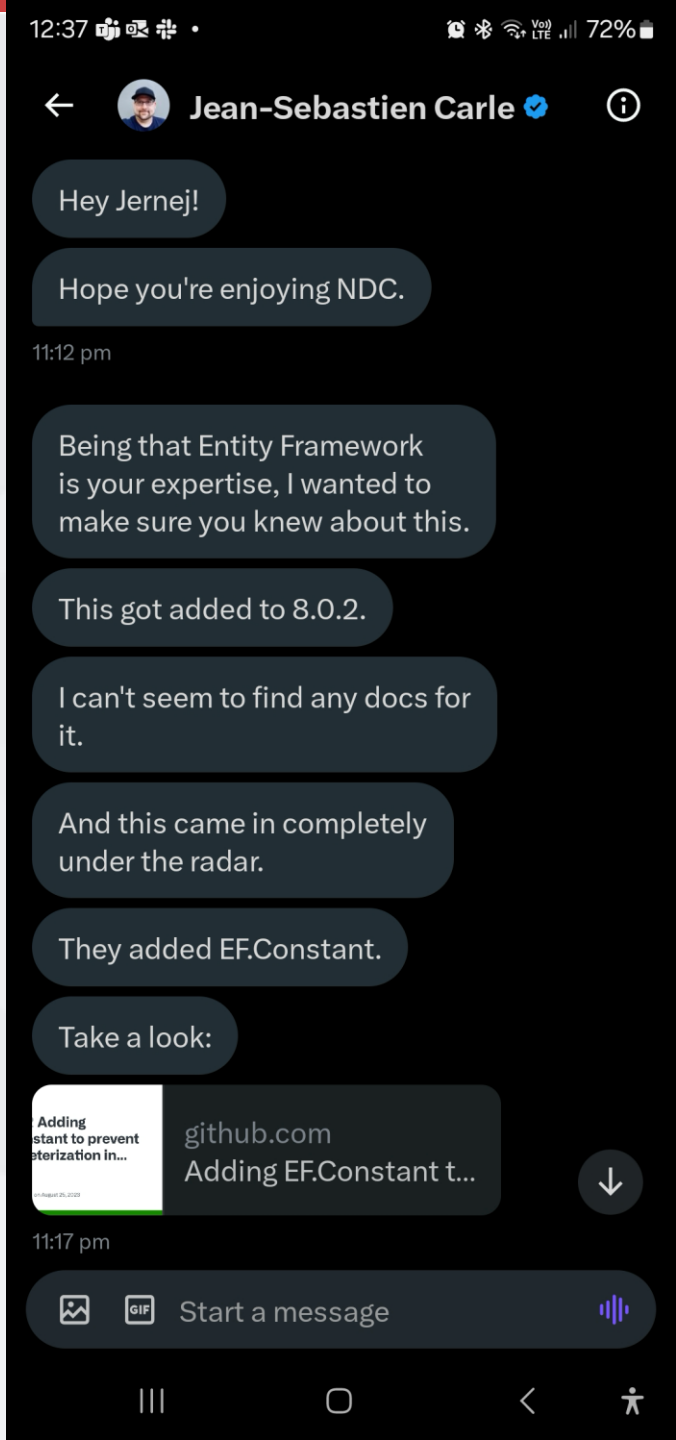
Summary



- Logs, telemetry and tagging
- The magic of SQL (querying, joining, grouping)
- Indexing

Bonus – EF.Constants #1

"Query is fast in SSMS but slow in production"



What problem are you trying to solve?

Feature coming in EF Core 8.0.2! Let SQL Server optimise for runtime constants!
github.com/dotnet/efcore/issues/31552

"Query is fast in SSMS but slow in production" kind of performance issues, where query is parametrized by EF and SQL Server chooses query execution plan suboptimal for concrete parameter value.

For example:

```
SELECT TOP(1000) * FROM dbo.MyView  -- fast
```

```
int pageSize = 60
if (loadMore) pageSize = 1000;
ctx.MyView.Take(pageSize).ToArray();  //slow when pageSize is 1000
```

Describe the solution you'd like

Introduce an EF.Constant() mechanism, which is identified by EF Core and prevents parameterization.
This goes hand in hand with **#28151**

```
ctx.MyView.Take(() => EF.Contant(pageSize)).ToArray();
ctx.MyView.Where(b => b.Name == EF.Constant("bar"))
```

Alternatively, add support for query hints like RECOMPILE or OPTIMIZE FOR

Resources

Rules to Better SQL Databases - Performance

ssw.com.au/rules/rules-to-better-sql-databases-performance

Rules to Better Entity Framework

ssw.com.au/rules/rules-to-better-entity-framework

Common Mistakes in EF Core | Jernej Kavka (JK)

youtube.com/watch?v=dDANjr5MCew

How to Plan for Database Performance | Bryden Oliver

youtube.com/watch?v=l18ltcOVN4I

Enable Azure Monitor OpenTelemetry

learn.microsoft.com/en-us/azure/azure-monitor/app/opentelemetry-enable?tabs=aspnetcore

Microsoft Learn | EF Core | Indexes

learn.microsoft.com/en-us/ef/core/modeling/indexes

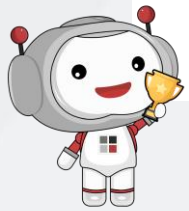
Gist | TagWithContext

gist.github.com/jernejk/7a47bc95b961777b3c8fe695675b40d8

Time to earn some swag!

Earn 800 SSW points ★ by scanning the QR Code.

Thank you!



info@ssw.com.au

www.ssw.com.au

Sydney | Melbourne | Brisbane | Newcastle | China | France | USA