

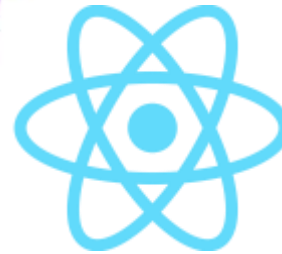
# Fehler, die EF Core-Entwickler mit SQL Server häufig machen



Thorsten Kansy (tkansy@dotnetconsulting.eu)

# Meine Person- Thorsten Kansy

Freier Consultant, Software Architekt,  
Entwickler, Trainer & Fachautor



Azure Cosmos DB



# Mein Service- Ihr Benefit

- Individuelle Trainings
- Projektbegleitung
- Beratung
  - Problemanalyse und Lösungen
  - Technologieentscheidungen



# Agenda

- Allgemeines
- Der große Abgrund
- Nur und ausschließlich LINQ verwenden
- Nur Migration verwenden und die Features der Datenbank ignorieren
- Arbeit mit DbContext
- Keine Echt-Zeit-Abbildung der Serverdaten
- Exceptions beim Abbruch asynchroner Anfragen nicht behandeln
- Parallele Abfragen mit einer einzelnen DbContext-Instanz versuchen



# Drama, Drama, Drama

- Das Multi-User-Dilemma
- Das Abfrage-Dilemma
- Das Index-Dilemma





# Allgemeines

# Allgemeines

Server und Client haben nur sporadische Verbindung.

Große Datenbanken sind nicht automatisch langsam,  
kleine nicht schnell.

Bessere Hardware bedeutet nicht automatisch bessere Performance.

Optimierung ist nicht unendlich möglich.

Der (SQL) Server ist sprach-neutral.

Und nein, an schlechter Performance ist *nicht* immer der SQL Server schuld.





# Der große Abgrund



# Der große Abgrund

Zugriffe auf den Server sind langsam!

1.000 Milisekunden (ms)

1.000.000 Mikrosekunden ( $\mu$ s)

1.000.000.000 Nanosekunden (ns)

RAM (ns) > SSD ( $\mu$ s) > HDD(ms) > Server (ms)



A photograph of a traditional Chinese building with a tiled roof and a stone courtyard. The building has a central entrance with a prominent roof. The courtyard is paved with large, irregular stones. The text "Nur LINQ verwenden" is overlaid on the image in a white font on a dark blue background.

Nur LINQ verwenden

# Nur und ausschließlich LINQ verwenden

LINQ ist genial

JOIN => Probleme mit dem DbContext?

LINQ ist im Vergleich zu T-SQL eingeschränkt

...



# Sichten, Prozeduren & Co.

SQL Server kann viel mehr als nur Tabellen

- Sichten
- Stored Procedures
- Funktionen
- Trigger
- Volltextsuche
- ...



# DML mit Stored Procedures

## Insert/ Delete/ Update mit Stored Procedures

```
modelBuilder.Entity<Document>(  
    entityTypeBuilder =>  
    {  
        if (UseStoredProcedures)  
        {  
            entityTypeBuilder.InsertUsingStoredProcedure("dbo.usp_InsertDocument",  
                storedProcedureBuilder =>  
                {  
                    storedProcedureBuilder.HasParameter("Discriminator");  
                    storedProcedureBuilder.HasParameter(document => document.Title);  
                    storedProcedureBuilder.HasParameter(document => document.NumberOfPages);  
                    storedProcedureBuilder.HasParameter(document => document.PublicationDate);  
                    storedProcedureBuilder.HasParameter(document => document.CoverArt);  
                    storedProcedureBuilder.HasResultColumn(document => document.Id);  
                    storedProcedureBuilder.HasParameter((Book document) => document.Isbn);  
                    storedProcedureBuilder.HasParameter((Magazine document) => document.CoverPrice);  
                    storedProcedureBuilder.HasParameter((Magazine document) => document.IssueNumber);  
                    storedProcedureBuilder.HasParameter("EditorId");  
                    storedProcedureBuilder.HasResultColumn(document => document.FirstRecordedOn);  
                    storedProcedureBuilder.HasResultColumn(document => document.RetrievedOn);  
                    storedProcedureBuilder.HasResultColumn(document => document.RowVersion);  
                })  
            .UpdateUsingStoredProcedure("dbo.usp_UpdateDocument",  
                storedProcedureBuilder =>
```



An aerial photograph of a city skyline at dusk. The sky is a mix of blue and orange, with some clouds. In the foreground, there's a large, modern building with a curved facade and many windows, some of which are lit up. The building has a sign that says "QSNCC". To the left, there's another building with a sign in Thai script. The overall scene is a cityscape at twilight.

Was kann das Datenbanksystem?

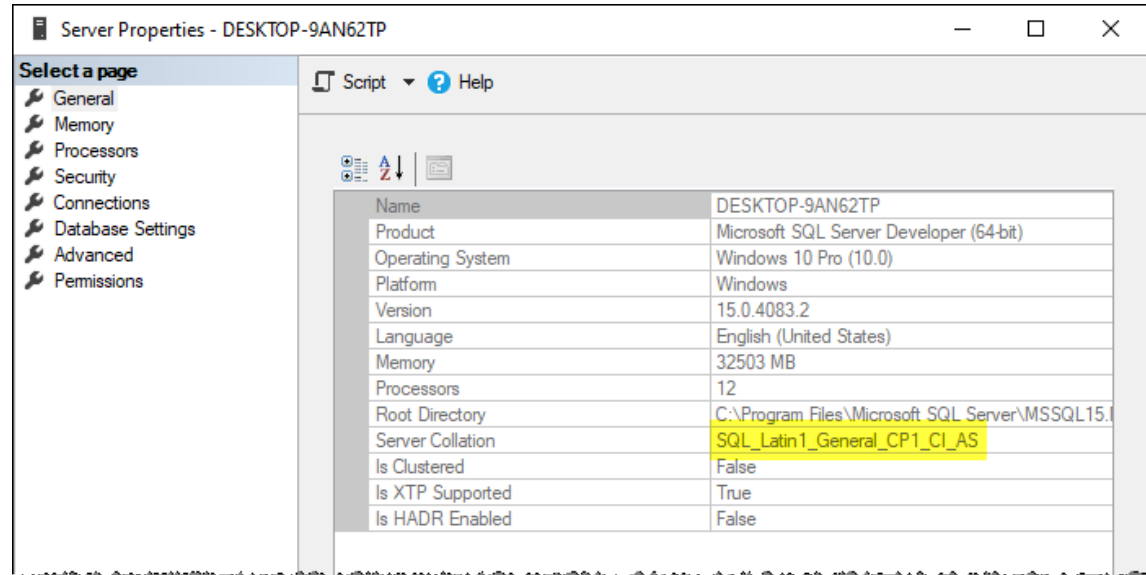
# Was kann das Datenbanksystem?

- Unicode oder kein Unicode?
  - Kostet sonst unnötig RAM und I/O
- Abfragetechniken > LINQ
- Collation
  - Groß-/ Kleinschreibung
- Volltextsuche
- Temporal Tables (aka. system-versioned tables)
- ...



# Collation

Groß-/ Kleinschreibung  
Sortierung



Server Properties - DESKTOP-9AN62TP

Select a page

- General
- Memory
- Processors
- Security
- Connections
- Database Settings
- Advanced
- Permissions

Script Help

Name	Value
Name	DESKTOP-9AN62TP
Product	Microsoft SQL Server Developer (64-bit)
Operating System	Windows 10 Pro (10.0)
Platform	Windows
Version	15.0.4083.2
Language	English (United States)
Memory	32503 MB
Processors	12
Root Directory	C:\Program Files\Microsoft SQL Server\MSSQL15.I
Server Collation	SQL_Latin1_General_CP1_CI_AS
Is Clustered	False
Is XTP Supported	True
Is HADR Enabled	False

CI Case insensitive / CS Case sensitive

AI Accent insensitive / AS Accent sensitive







# Arbeit mit dem DbContext

# Arbeit mit dem DbContext

- Falsche Lebenszeit für DbContext
  - So lang wie nötig, so kurz wie möglich
- Zu umfangreicher DbContext
  - Kein Undo/ Nicht speichern durch den Anwender durch „Disposen“ der DbContext
  - Probleme mit Optimistic Concurrency beim Speichern



The image shows the ruins of an ancient building, possibly a temple or public structure, with a central courtyard. The architecture features stone walls, a central column, and a rectangular platform with a large terracotta pot in the center. A blue semi-transparent banner is overlaid across the middle of the image, containing the text "Asynchrone Exceptions".

# Asynchrone Exceptions

# Exceptions beim Abbruch asynchroner Abfragen

```
public async Task<IEnumerable<UserDto>> FetchAllUsersAsync(CancellationTokentoken cancellationToken)
{
    logger.LogInformation("FetchAllUsersAsync()");

    try
    {
        var rawResult = await dbContext.Users.Include(i => i.Roles)
                                             .Include(i => i.UserPlantCodes)
                                             .AsNoTracking()
                                             .OrderBy(o => o.Name)
                                             .ToListAsync(cancellationToken);

        // ...
    }
    catch (OperationCanceledException)
    {
        // Nur Abbruch des Tasks
        return null!;
    }
    catch (Exception ex)
    {
        logger.LogError(ex, "Can't fetch system permissions");
        throw;
    }
}
```





# Parallele Abfragen

# Parallele Abfragen

Pro DbContext-Instanz eine Verbindung zum SQL Server

Eine Verbindung, ein Zugriff

Ergo: mehrere parallele Abfragen, mehrere DbContext-Instanzen





# Indizes-Mythen

# Indizes-Mythen

- Indizes sind optional
- Je mehr Indizes, desto besser (“Viel hilft viel”)
- Indizes sind nicht statisch- einmal konzipieren reicht nicht!
- ...





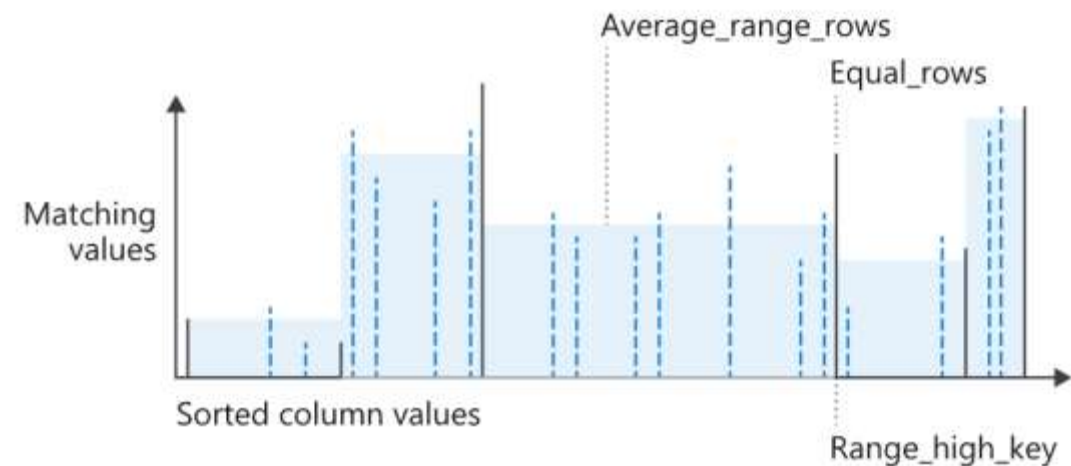


# Das Index-Dilemma

# Das Index-Dilemma

Der SQL Server muss möglichst schnell entscheiden, ob und welchen Index er verwendet und das kostet Zeit.

Histogramme geben Auskunft über die Verteilung der Daten



A landscape photograph showing a rocky terrain under a bright blue sky with scattered white clouds. In the middle ground, there are several stacks of stones, some of which are adorned with colorful prayer flags (blue, white, and red). The foreground consists of large, layered rock formations. A dark blue horizontal band is overlaid across the middle of the image, containing the title text.

# Das Multi-User-Dilemma

# Das Multi-User-Dilemma

Viele “chaotische” Zugriffe geschehen parallel und müssen vor anderen Zugriffen abgegrenzt (“geschützt”) werden. Dies erfordert Sperren und verursacht längere Zugriffszeiten.

## SQL Server

- blockt Ressourcen während der Zugriffe
- Ressourcen (Row, Pages, Extends, Tabellen, Datenbanken, ...)
- Zugriffe (Exklusiv, Shared, ...)
- Zugriffe können koexistieren oder auch eben nicht (=> Locks)



# Optimistic Concurrency

Es gibt keinen Mechanismus, um Daten zu “sperrern” in Form einer Zuweisung an einen User (wie z.B. das Auschecken in SharePoint)

Der Client erkennt nie mit 100%tiger Sicherheit, wie die Daten auf dem SQL Server aussehen.

Und Daten können “gleichzeitig” verändert werden.





Bonus

# Temporal Tables- Automatische Historisierung

```
CREATE TABLE dbo.Werte
(
    ID INT IDENTITY(1,1) NOT NULL,
    Wert1 NVARCHAR(10) NULL,
    Wert2 NVARCHAR(10) NULL,
    StartTime datetime2(7) GENERATED ALWAYS AS ROW START HIDDEN NOT NULL,
    EndTime datetime2(7) GENERATED ALWAYS AS ROW END HIDDEN NOT NULL,
    PERIOD FOR SYSTEM_TIME(StartTime, EndTime),
    CONSTRAINT Werte_PK PRIMARY KEY (ID)
)
WITH
(SYSTEM_VERSIONING = ON (HISTORY_TABLE = dbo.Werte_History));

SELECT * FROM dbo.Werte
[FOR SYSTEM_TIME AS OF '2016-01-25 18:21:24.0738473'];
```



# IsTemporal (Fluent API)

- Temporal table (SQL Server)
- Standard: -

```
modelBuilder.Entity<MyTemporalTable>()  
    .ToTable(t => t.IsTemporal(hist =>  
        {  
            hist.UseHistoryTable("SessionHistory");  
        }  
    );
```

```
dbContext.MyTemporalTable  
    .TemporalAsOf(<Date>).Where(w => ...);
```





# Temporal AsOf & Co

TemporalAll()	Alle Daten aus der Tabelle und der historischen Tabelle.
TemporalAsOf(...)	Daten zu einem bestimmten Zeitpunkt.
TemporalFromTo(...)	Daten aus der Tabelle plus der historischen Tabelle exk. Einschluss der oberen Grenze.
TemporalBetween(...)	Daten aus der Tabelle plus der historischen Tabelle inkl. Einschluss der oberen Grenze.
TemporalContainedIn(...)	Nur Daten aus den der historischen Tabelle aus dem Zeitraum.




# Fragen? Jetzt oder später!



## Kontakt

 E-Mail  
tkansy@dotnetconsulting.eu

 Telefon  
+49 (0) 6187 / 2009090

 Microsoft Teams  
Meet now

 LinkedIn  
Link me

 XING  
Xing me

 X (Twitter)  
@tkansy



# www.dotnetconsulting.eu

SQL Server meets .NET (Core)- professionally!



Ich berate, coache und trainiere im Bereich Entwicklung von .NET (Core) Anwendungen mit Microsoft SQL Server- mit Allem, was dazu gehört- und was man vielleicht weglassen sollte.

