



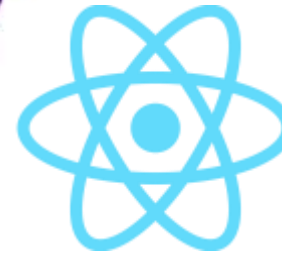
Warum deine Anwendung den SQL Server langsam macht (und nicht umgekehrt)

Thorsten Kansy



Meine Person- Thorsten Kansy

Freier Consultant, Software Architekt,
Entwickler, Trainer & Fachautor



Azure Cosmos DB



Mein Service- Ihr Benefit

- Individuelle Inhouse Trainings
- (Online on-demand) Projektbegleitung
- Beratung
 - Problemanalyse und Lösungen
 - Technologieentscheidungen



Agenda- Auswahl für Entwickler

- Der SQL Server ist langsam!
- Erst messen, dann optimieren
- Zu viele Datenbankzugriffe
- Zu viele Daten laden
- Harmloses LINQ, problematisches SQL
- Verbindungen, Transaktionen und Sperren
- Parallelität, Async und selbst erzeugte Überlastung
- Die wichtigsten Regeln



Der SQL Server ist langsam!

Der SQL Server ist langsam

- Anwendung braucht plötzlich mehrere Sekunden
- DBA sieht dagegen normale Serverauslastung
- Entwickler verdächtigen alle anderen oder fehlende Hardware

Leitfrage: Wo entsteht die Wartezeit wirklich?

- Abgrenzung zwischen:
 - Datenbank
 - Abfragen
 - Datenmengen
 - Netzwerk
 - Sonstige Komponenten

Bevor wir dem SQL Server mehr Hardware geben, sollten wir prüfen, was unsere Anwendung eigentlich mit ihm macht.

A photograph of a wooden staircase with a metal railing in a forest. The railing is made of dark wood and yellow-painted metal poles. The stairs are made of wooden planks. The background is a dense forest with many trees and green foliage. A dark blue banner is overlaid across the middle of the image, containing the text "Erst messen, dann optimieren" in white.

Erst messen, dann optimieren

Gesamtlaufzeit einer Operation zerlegen

- Verbindungsaufbau
- Übertragung zum Server
- Ausführung der Abfrage
- Übertragung der Ergebnisse
- Materialisierung durch EF Core
- Verarbeitung und Serialisierung in der Anwendung

Gesamtlaufzeit einer Operation zerlegen

- Anwendungslogging mit Zeitmessung
- SQL-Ausgaben von EF Core sichtbar machen
- SQL Server Query Store
- Ausführungspläne
- `SET STATISTICS IO, TIME ON`
- Korrelation zwischen HTTP-Request und SQL-Abfrage
- Durchschnittswerte versus Ausreißer
- Lokale Entwicklung versus produktive Datenmengen

Eine Operation, die fünf Sekunden dauert,
muss nicht fünf Sekunden auf dem SQL
Server verbringen.



Zu viele Datenbankzugriffe

Zu viele Datenbankzugriffe

- Lazy Loading als unsichtbares Problem (N+1-Abfragen)
- Datenbankzugriffe innerhalb von Schleifen
- Für jeden Datensatz eine weitere Abfrage
- Mehrere kleine Abfragen statt einer gezielten Abfrage
- Datenbankzugriffe über mehrere Anwendungsschichten verteilt
- Netzwerk-Roundtrips als eigentliche Ursache
- Lokale Datenbank kaschiert das Problem
- Cloud-Datenbank macht Latenz sichtbar

A photograph of a hippopotamus in a grassy field. The hippo is dark grey and is partially obscured by a semi-transparent blue horizontal band that spans the width of the image. The text "Zu viele Daten laden" is written in white on this band. The background consists of green grass and some dry stalks.

Zu viele Daten laden

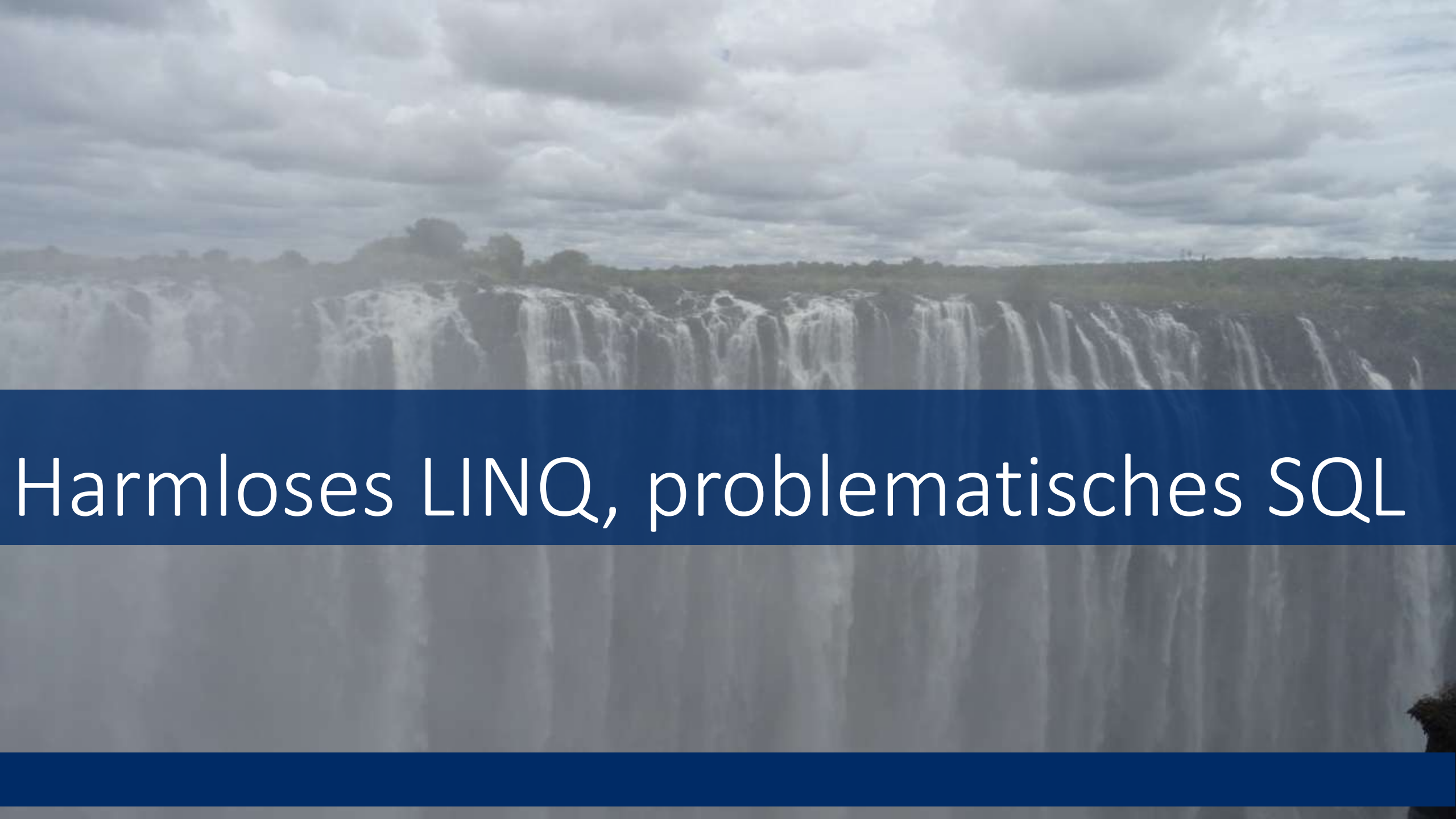
Zu viele Daten laden

- `SELECT *` beziehungsweise vollständige Entitäten
- Unmengen Spalten laden, obwohl nur wenige angezeigt werden
- Große Text-, JSON- oder Binärspalten unnötig übertragen
- Ganze Objektgraphen mit `Include()` laden
- Mehrere Collections gleichzeitig einbeziehen
- Kartesische Vervielfachung der Ergebnismenge
- Unmengen Datensätze laden und anschließend im Code filtern

Besser (wenn möglich)

- Projektionen auf DTOs
- `Select()` gezielt einsetzen
- Filterung, Aggregationen, Sortierung auf der Datenbank ausführen

Nicht die Anzahl der zurückgegebenen Objekte ist entscheidend, sondern die Datenmenge, die SQL Server erzeugen und übertragen muss.

A wide, powerful waterfall cascading over a rocky ledge under a cloudy sky. The water is white and frothy as it falls, creating a misty spray at the base. The sky is filled with large, grey clouds, and the horizon shows a line of green trees and vegetation.

Harmloses LINQ, problematisches SQL

Harmloses LINQ, problematisches SQL

- LINQ zeigt nicht unmittelbar, welche SQL-Abfrage entsteht
 - Komplexe Navigationen
 - Verschachtelte Unterabfragen
 - Unnötige Joins
- Generiertes SQL immer wieder kontrollieren
- `Contains()` mit sehr großen Wertelisten
- Sortierungen ohne fachliche Notwendigkeit
- Funktionen auf gefilterten Spalten
- Datentypkonvertierungen

Harmloses LINQ, problematisches SQL

- Unterschied zwischen:
 - LINQ to Objects
 - LINQ to Entities
- Zu früher Wechsel zu `AsEnumerable()`
 - Materialisierung durch `ToList()` an der falschen Stelle
- Filter erst nach der Materialisierung
- Wiederholte Ausführung derselben Abfrage



Verbindungen, Transaktionen und Locks



Verbindungen, Transaktionen und Locks

- Verbindungspooling nicht mit dauerhaft offenen Verbindungen verwechseln
- DbContext lebt zu lange Nicht freigegebene Connections
- Connection-Pool-Erschöpfung
- Transaktion beginnt zu früh
- Zu umfangreiche Transaktionen
- Benutzerinteraktion innerhalb einer Transaktion

Verbindungen, Transaktionen und Locks

- Transaktion beginnt deutlich zu früh
- Externe API-Aufrufe innerhalb einer Datenbanktransaktion
- Lange Sperren durch langsamen Anwendungscode
- Viele Änderungen einzeln speichern
- SaveChanges() innerhalb einer Schleife
- Updates bündeln
- Transaktionsgrenzen bewusst festlegen

A photograph of a train station platform. In the foreground, a person stands on the left, looking towards the tracks. The platform is covered by a large, dark, corrugated metal roof. To the right, there are several buildings, including a white one with a blue sign. In the background, a bridge with a brick structure spans over the tracks. The tracks themselves are made of steel rails and wooden sleepers, receding into the distance. The sky is overcast and hazy.

Parallelität, Async und selbst erzeugte Überlastung

Parallelität, Async und selbst erzeugte Überlastung

- `async` macht eine Abfrage nicht automatisch schneller
- Parallele Abfragen gegen dieselbe Datenbank
- Unbegrenztes `Task.WhenAll()`
- Retry-Stürme bei kurzen Störungen (Polly?)
- Hintergrundjobs konkurrieren mit Benutzeranfragen
- Abgebrochene Requests und weiterlaufende Arbeit
 - `CancellationToken` konsequent durchreichen
- Lasttests mit realistischen Datenbankzugriffen

Die wichtigsten Regeln



Die wichtigsten Regeln

1. Anzahl der SQL-Kommandos messen
2. Das tatsächlich erzeugte SQL ansehen
3. Nur benötigte Spalten laden
4. Filterung und Aggregation auf dem Server ausführen
5. N+1-Abfragen vermeiden
6. DbContext und Transaktionen kurzlebig halten
7. `SaveChanges()` nicht unnötig oft aufrufen
8. Parallelität begrenzen
9. Timeouts und Retries bewusst konfigurieren

Erst die Ursache finden, dann Hardware
kaufen

Fragen? Jetzt oder später!



Kontakt

 **E-Mail**
tkansy@dotnetconsulting.eu

 **Telefon**
[+49 \(0\) 6187 / 2009090](tel:+49(0)61872009090)

 **Microsoft Teams**
[Meet now](#)

 **LinkedIn**
[Link me](#)

 **XING**
[Xing me](#)

 **X (Twitter)**
[@tkansy](#)



Wir bitten um dein Feedback

Jetzt bewerten!



entwickler.de

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.

