Remote on-demand Support



Was EF Core Entwickler über SQL Server-Datenbanken wissen sollten

Thorsten Kansy (tkansy@dotnetconsulting.eu)

Meine Person- Thorsten Kansy

Freier Consultant, Software Architekt,
Entwickler, Trainer & Fachautor

Microsoft SQL Server

.NET Core

ASP.NET Core

Azure Cosmos DB

DOTNETCONSULTING

Mein Service- Ihr Benefit

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



DOTNETCONSULTING

Agenda

- Allgemeines
- Der große Abgrund
- Datentypen
- Collation
- Wie Daten organisiert werden
- Sichten, Prozeduren & Co
- Indizes-Mythen

DOTNET CONSULTING

Agenda II

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

DOTNET CONSULTING

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

DOTNETCONSULTING

Der große Abgrund

Der große Abgrund

Zugriffe auf den Server sind langsam!

1.000 Milisekunden (ms)

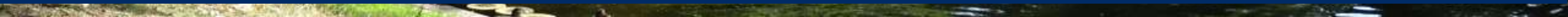
1.000.000 Mikrosekunden (µs)

1.000.000.000 Nanosekunden (ns)

RAM (ns) > SSD (µs) > HDD(ms) > Server (ms)

DOTNETCONSULTING

Temporal Tables



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'];
```

DOTNET CONSULTING

.

IsTemporal (Fluent API)

6.0

- Temportal 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.

JSON (Owned Types)

Owned Types

Aka Complex Types (Nested Documents bei NoSQL)

Owned-Attribute

C# Copy

```
[Owned]
public class StreetAddress
{
    public string Street { get; set; }
    public string City { get; set; }
}

public class Order
{
    public int Id { get; set; }
    public StreetAddress ShippingAddress { get; set; }
}
```

Owned Types (n-Elemente)

```
modelBuilder.Entity<Book>()
    .OwnsMany<Chapter>("Chapters")
    .WithOwner();

modelBuilder.Entity<Book>()
    .OwnsOne<Author>("Author")
    .WithOwner();
```


```
{
    "id": "f7949954-f258-44b1-b9fe-1e1fc2c85d17",
    "Discriminator": "Book",
    "Title": "How to EF Core",
    "Author": {
        "Name": "Thorsten Kansy",
        "id": "00000000-0000-0000-0000-000000000000"
    },
    "Chapters": [
        {
            "id": "1ffbe825-e4fd-4d81-ba1f-e733d31e9fe1",
            "Content": "...",
            "Index": 1,
            "Title": "Chapter #1"
        },
        {
            "id": "563e9c0a-43cc-4697-8714-136317913980",
            "Content": "...",
            "Index": 2,
            "Title": "Chapter #2"
        }
```


JSON Column

EF 7.0

Map von JSON Columns auf Atrribute

```
modelBuilder.Entity<Post>().OwnsOne(
                post => post.Metadata, ownedNavigationBuilder =>
                {
                    ownedNavigationBuilder.ToJson();
                    ownedNavigationBuilder.OwnsMany(metadata => metadata.TopSearches);
                    ownedNavigationBuilder.OwnsMany(metadata => metadata.TopGeographies);
                    ownedNavigationBuilder.OwnsMany(
                        metadata => metadata.Updates,
                        ownedOwnedNavigationBuilder =>
                           ownedOwnedNavigationBuilder.OwnsMany(update => update.Commits));
                });
```

Sichten, Prozeduren & Co.

Sichten, Prozeduren & Co.

SQL Server kann viel mehr als nur Tabellen

- Sichten
- Stored Procedures
- Funktionen
- Trigger
- Volltextsuche
- …

DOTNET CONSULTING

DML mit Stored Procedures

DML mit Stored Procedures

EF 7.0

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 =>
```

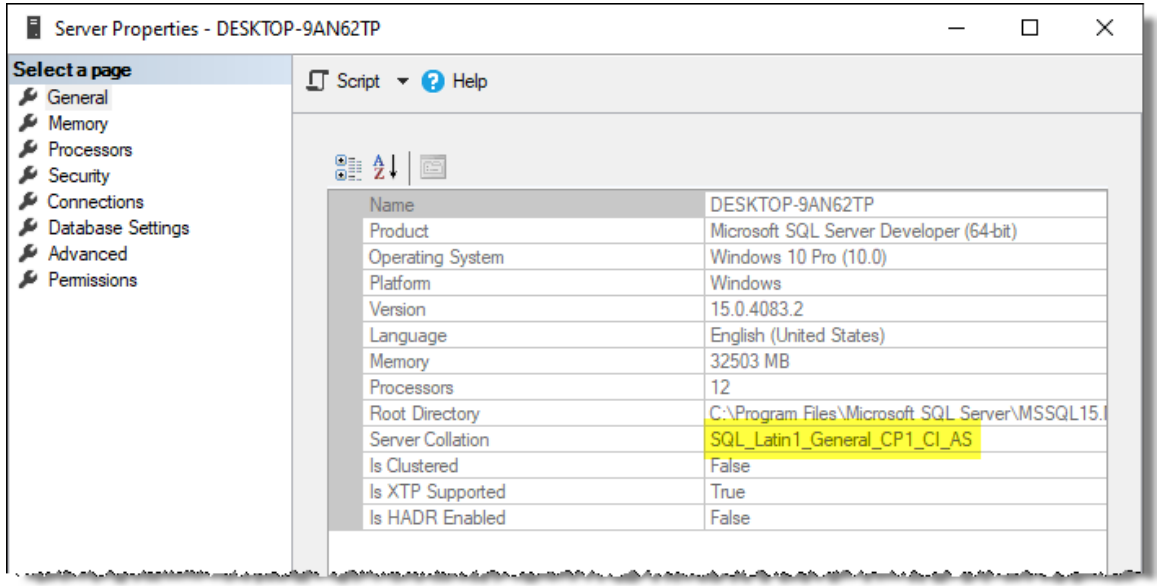

CLEMENS XII · PONT · MAX ·
AQVAM VIRGINEM
COPIA ET SALVBRITATE COMMENDATAM
CVLTV MAGNIFICO ORNAVIT
ANNO DOMINI MDCCXXXV PONTIF VI

Collation

Collation

Groß-/ Kleinschreibung

Sortierung



CI Case insensitive / CS Case sensitive

AI Accent insensitive / AS Accent sensitive

DOTNETCONSULTING

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)

DOTNETCONSULTING

Optimistic Concurrency

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

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

Und Daten können “gleichzeitig” verändert warden.

DOTNET CONSULTING

Indizes-Mythen

Indizes-Mythen

Indizes sind optional.

Je mehr Indizes, desto bester (“Viel hilft viel”)

Indizes sind nicht statisch- einmal konzipieren reicht nicht!

…

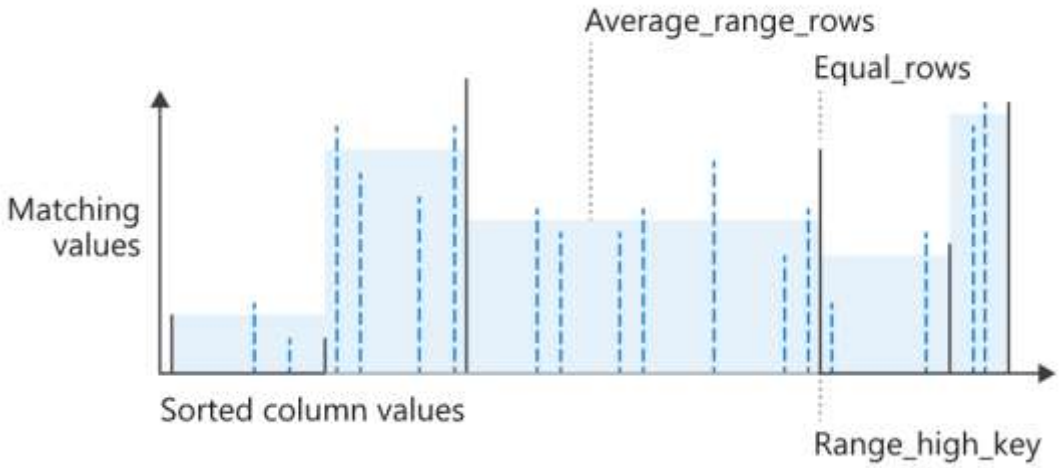
DOTNETCONSULTING

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



DOTNETCONSULTING

Fragen? Jetzt oder später!

www.dotnetconsulting.eu

DOTNETCONSULTING

Links

www.dotnetconsulting.eu

@Tkansy

tkansy@dotnetconsulting.eu

DOTNETCONSULTING