



EF Core 3.1

Cosmos DB



Thorsten Kansy (tkansy@dotnetconsulting.eu)

Meine Person- Thorsten Kansy

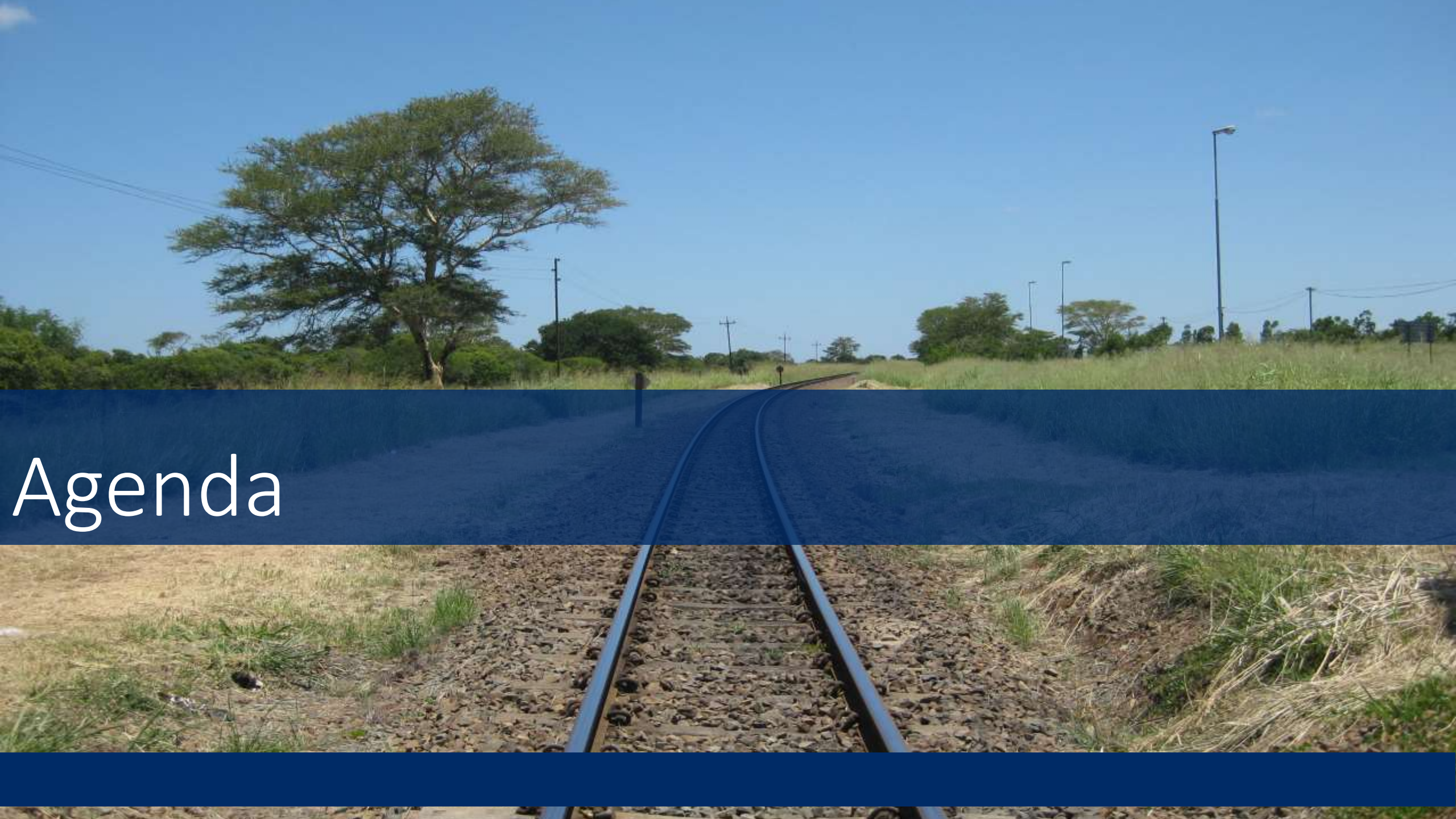
Freier Consultant, Software Architekt,
Entwickler, Trainer & Fachautor



Mein Service

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





Agenda

Agenda

- Grundlagen
 - Vergleich zu RDBMS
 - Aufbau
- DbContext
- Entity Types
- Owned Types
- Shadow States

Grundlagen



RDBMS vs. NoSQL



SQL Server

Festes Schema

Wenig Redundanzen

Vertikal skaliert



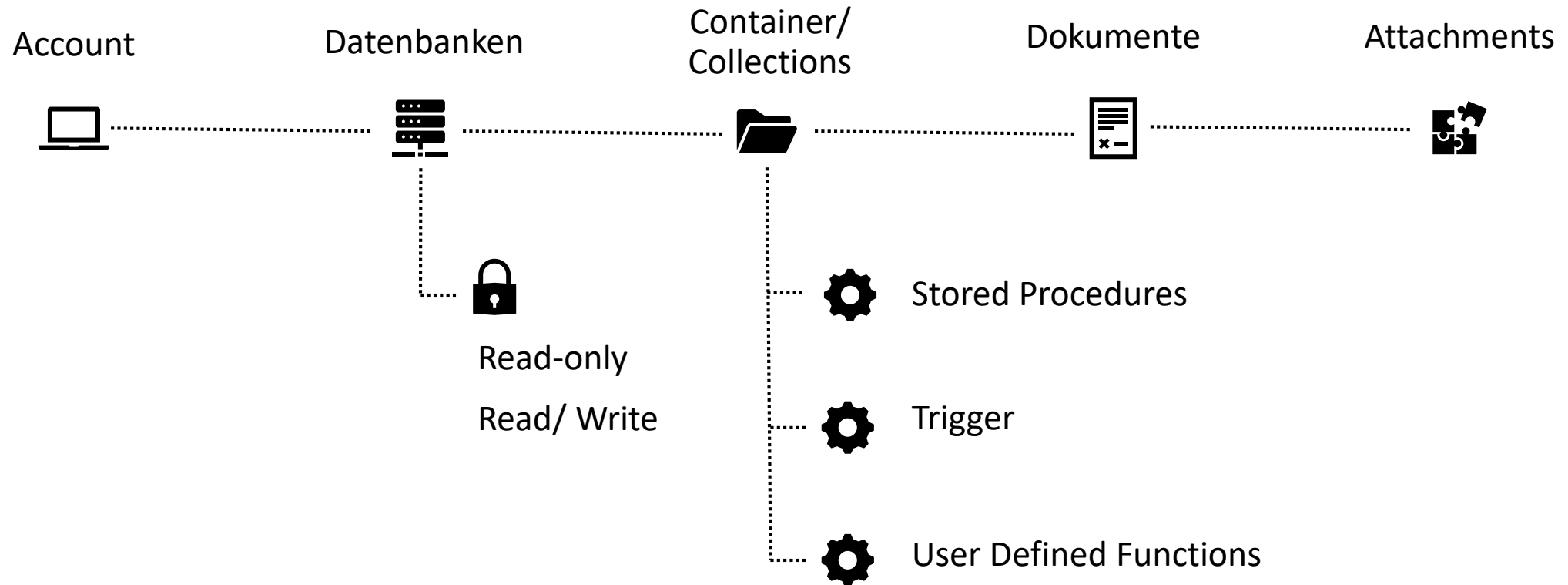
Azure Cosmos DB

Schema frei (NoSQL)

Viele Redundanzen

Horizontal skaliert

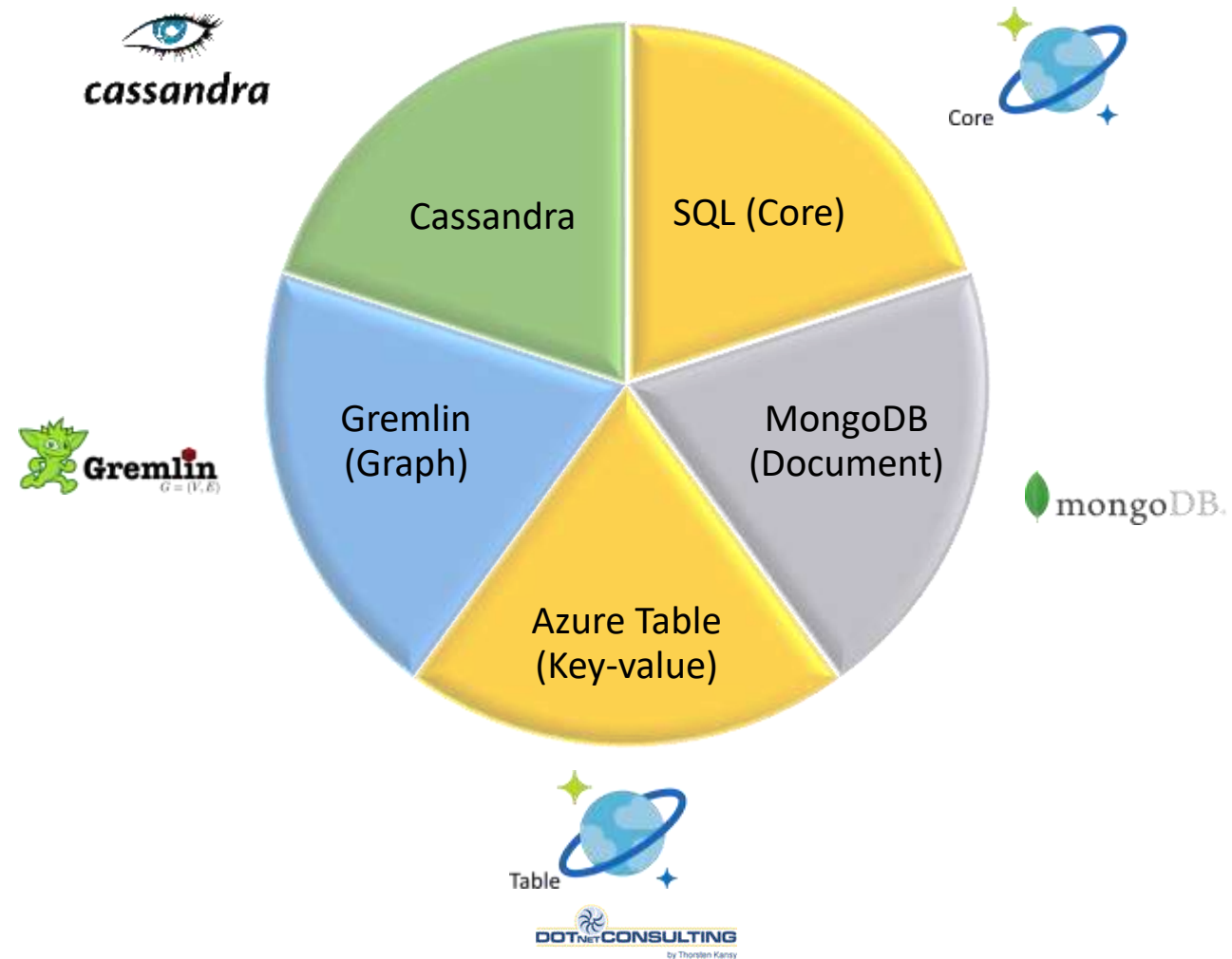
Grundaufbau



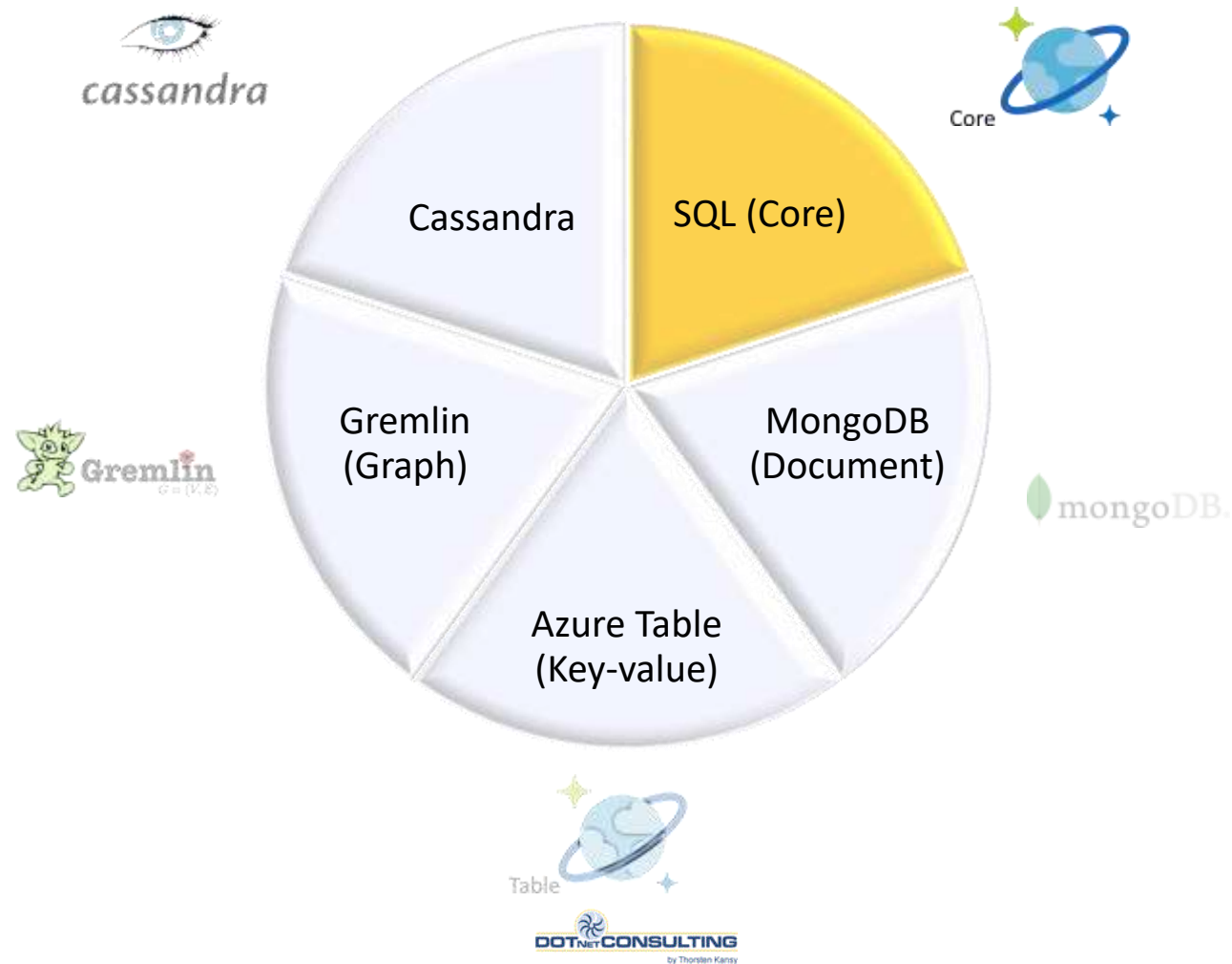
Grundaufbau

- Datenbank
 - Umfaßt 1..n Container/ Collections
- Container/ Collection
 - Umfaßt n Dokumente
 - Einstellungen
 - Partition Key
 - Index Policies
 - Unique Key
 - Stored Procedures
 - User Defined Functions
 - Triggers

Unterstützte APIs

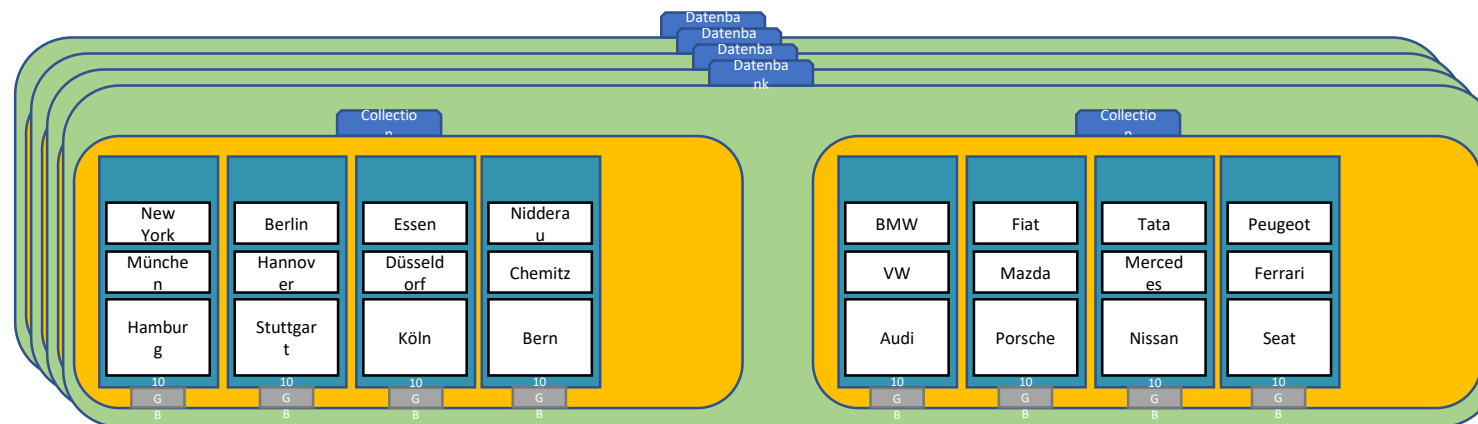


Von EF Core unterstützte API



Physikalischer Ausbau

- Mindestens eine Region pro Datenbank
- Min 4 Instanzen pro Datenbank & Region
 - Lastverteilung
- Automatische Replikation
 - Intra- und Interregional



Anforderungen an das JSON-Dokument

~~Eindeutige id-Eigenschaft (Key)~~

ODER

Eindeutige id-Eigenschaft + PartitionKey

Einige Bezeichner sind verboten

id

Kleingeschrieben(!)

GUID (optional)

PartitionKey

Beliebige Eigenschaft

Performance
relevant

PartitionKey: 1	PartitionKey: 2	PartitionKey: 3
Key:1	Key:1	Key:1
Key: 2	Key: 20	Key: 30
Key: 3	Key: 21	Key: 42
Key: 4	Key: 22	Key: 344

Request Units (RUs)

- Request Unit (RU) $\neq$ Request
- RU
 - Maß für Berechnungsaufwand („Comos DB-Währung“)
 - CPU • Speicher • Disk I/O • Netzwerk I/O
 - Deterministisch
- Reservierte RU/s
 - 400..10.000.. ∞ (je mehr, desto teurer)
 - Kostenberechnung nach RU/s je Container/ Collection oder Datenbank
 - Bei Überschreitung folgt HTTP 429



<https://www.documentdb.com/capacityplanner>

<https://azure.microsoft.com/en-us/pricing/details/cosmos-db/>

<https://httpstatusdogs.com/>

A scenic landscape photograph featuring a body of water in the foreground, a dense forested hill in the middle ground, and bare tree branches in the upper left corner. Two tall palm trees stand out against the sky. A dark blue horizontal band is overlaid across the middle of the image, containing white text.

SQL für Azure Cosmos DB



SQL für Azure Cosmos DB

- Nur lesende Abfragen (SELECT)
- Pflicht zur Qualifizierung
- Case-sensitive (Eigenschaften und Werte)
- Operatoren & eingebaute Funktionen
- Benutzerdefinierte Funktionen



<https://docs.microsoft.com/en-us/azure/cosmos-db/sql-api-query-reference>

Sprache Fragmente

- SELECT
 - Rückgabe von Eigenschaften und Berechnungen (inkl. AS)
- WHERE
 - Filter
- ORDER BY
 - (Einfache) Sortierung
- JOIN IN
 - Inner Join mit Array als Sub-Property

Eingebaute Funktionen

Arithmetic	ABS, CEILING, EXP, FLOOR, LOG, LOG10, POWER, ROUND, SIGN, SQRT, SQUARE, TRUNC, ACOS, ASIN, ATAN, ATN2, COS, COT, DEGREES, PI, RADIANS, SIN, TAN
Type	IS_ARRAY, IS_BOOL, IS_NULL, IS_NUMBER, IS_OBJECT, IS_STRING, IS_DEFINED, IS_PRIMITIVE
String	CONCAT, CONTAINS, ENDWITH, INDEX_OF, LEFT, LENGTH, LOWER, LTRIM, REPLACE, REPLICATE, REVERSE, RIGHT, RTRIM, TOSTRING, STARTSWITH, SUBSTRING, UPPER
Array	ARRAY_CONCAT, ARRAY_CONTAINS, ARRAY_LENGTH, ARRAY_SLICE
Aggregate	COUNT, SUM, MIN, MAX, AVG
Spatial	ST_DISTANCE, ST_WITHIN, ST_INTERSECTS, ST_ISVALID, ST_ISVALIDDETAILED

Demo



Transaktionen

Transaktionen

- ACID (Atomicity, Consistency, Isolation, Durability)
 - Snapshot Isolation
 - Exception => Rollback & Abbruch
- Transaktionsschutz pro Partition
 - Prozeduren
 - Trigger
 - Funktionen werden in einer Transaktion ausgeführt



<https://docs.microsoft.com/en-us/azure/cosmos-db/database-transactions-optimistic-concurrency>

A photograph of an elephant in a savanna landscape. The elephant is standing in a field of tall, dry grass, facing left. In the background, there are green bushes and trees. A dark blue horizontal band is overlaid across the middle of the image, containing white text.

DbProvider in Stellung bringen

DbProvider in Stellung bringen

Nuget, what else?

```
Install-package Microsoft.EntityFrameworkCore.Cosmos
```

```
<ItemGroup>  
  <PackageReference Include="Microsoft.EntityFrameworkCore.Cosmos" Version="3.1.1" />  
  <PackageReference Include="Microsoft.EntityFrameworkCore.Relational" Version="3.1.1" />  
  <PackageReference Include="Microsoft.Extensions.Logging.Console" Version="3.1.1" />  
  <PackageReference Include="Microsoft.Extensions.Logging.Debug" Version="3.1.1" />  
  <PackageReference Include="Microsoft.Extensions.DependencyInjection" Version="3.1.1" />  
  <PackageReference Include="Microsoft.Extensions.Configuration.Json" Version="3.1.1" />  
</ItemGroup>
```



DbContext

DbContext

- Zentrales Element
- Ausführen von Abfragen
- Hinzufügen/ Entfernen von Entitäten
- Registrierung von Änderungen an Entitäten

Konfiguration

- Verhalten des Kontextes
- Datenbank Provider festgelegt
- Logging

Konfiguration für Azure Cosmos DB

- Angabe von EndPoint, Key & Database

```
optionsBuilder.UseCosmos("https://localhost:8081",  
                        "C2y6yDjf5/R...lZnqyMsEcaGQy67XIw/Jw==",  
                        "EFCosmosDb");
```

Demo



Entity Types (Model für Cosmos DB)

HasDefaultContainer (Fluent API)

- Legt den Standardcontainer für alle Objekte fest
- Standard: -

```
modelBuilder.HasDefaultContainer("Books");
```

ToContainer (Fluent API)

- Legt den Container für eine Entität fest
- Standard: Name des Kontextes

```
modelBuilder.Entity<Session>()  
    .ToContainer("Sessions");
```

ToJsonProperty (Fluent API)

- Legt den Namen einer Entität im Json-Dokument fest
- Standard: Name der Eigenschaft

```
modelBuilder.Entity<Session>()  
    .JsonProperty("TechSession");
```


HasPartitionKey (Fluent API)

- Legt den Namen des PartionKeys fest
- Standard: - (__partitionKey)

```
modelBuilder.Entity<Session>()  
    .HasPartitionKey(p => p.Difficulty); // Scalar-Eigenschaft
```

Demo

A close-up photograph of a lionfish swimming in clear blue water. The fish has prominent red and white vertical stripes and long, thin, translucent spines with dark bands. The text 'Owned Types' is overlaid on the left side of the image in a white, sans-serif font.

Owned Types

Owned Types (1:0-Elemente)

Aka Complex Types (Nested Documents bei NoSQL)

Owned-Attribute 2.1

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"  
    }  
  ]  
}
```

Demo



Shadow Properties



Shadow Properties

- Eigenschaften, die nicht in der Entität angelegt sind
 - Domain-Driven Design Pattern

```
modelBuilder.Entity<TechEvent>()  
    .Property<String>("Code")  
    .ToJsonProperty("SecretCode");
```

```
Entry(techEvent1).Property("Code").CurrentValue
```

```
EF.Property<string>(w, "Code")
```

Demo

Fragen?

Links



<http://dotnetconsulting.eu/blog/>



[@Tkansy](https://twitter.com/Tkansy)



tkansy@dotnetconsulting.eu



www.dotnetconsulting.eu