Transaktionskonzepte in No-SQL Datenbanken

Digitales Handout
https://github.com/phd4S/AKAD

No-SQL Datenbanken

- Not Only SQL (Strukturierte Datenspeicher)

  Zugriff über REST API oder
  einfacher bzw. SQL ähnlicher Sprache

- Kein relationales Datenmodell
- Keine Normalisierung
- Verteiltes, auf horizontale Skalierbarkeit ausgelegtes, System

No-SQL Datenbanken

402 Systeme im Ranking, Januar 2023

Rang Jan 2023	Rang Dez 2022	Rang Jan 2022	DBMS	Datenbankmodell	Punkte Jan 2023	Punkte Dez 2022	Punkte Jan 2022
1.	1.	1.	Oracle	Relational, Multi-Model	1245,17	-5,14	-21,72
2.	2.	2.	MySQL	Relational, Multi-Model	1211,96	+12,56	+5,91
3.	3.	3.	Microsoft SQL Server	Relational, Multi-Model	919,39	-4,96	-25,43
4.	4.	4.	PostgreSQL	Relational, Multi-Model	614,85	-3,13	+8,29
5.	5.	5.	MongoDB	Document, Multi-Model	455,18	-14,15	-33,38
6.	6.	6.	Redis	Key-value, Multi-Model	177,56	-5,01	-0,43
7.	7.	7.	IBM Db2	Relational, Multi-Model	143,57	-3,05	-20,63
8.	8.	8.	Elasticsearch	Suchmaschine, Multi-Model	141,16	-3,76	-19,59
9.	9.	9.	Microsoft Access	Relational	133,36	-0,47	+4,41
10.	10.	10.	SQLite	Relational	131,49	-0,94	+4,06

https://db-engines.com/de/ranking

Transaktionen in **SQL** Datenbanken

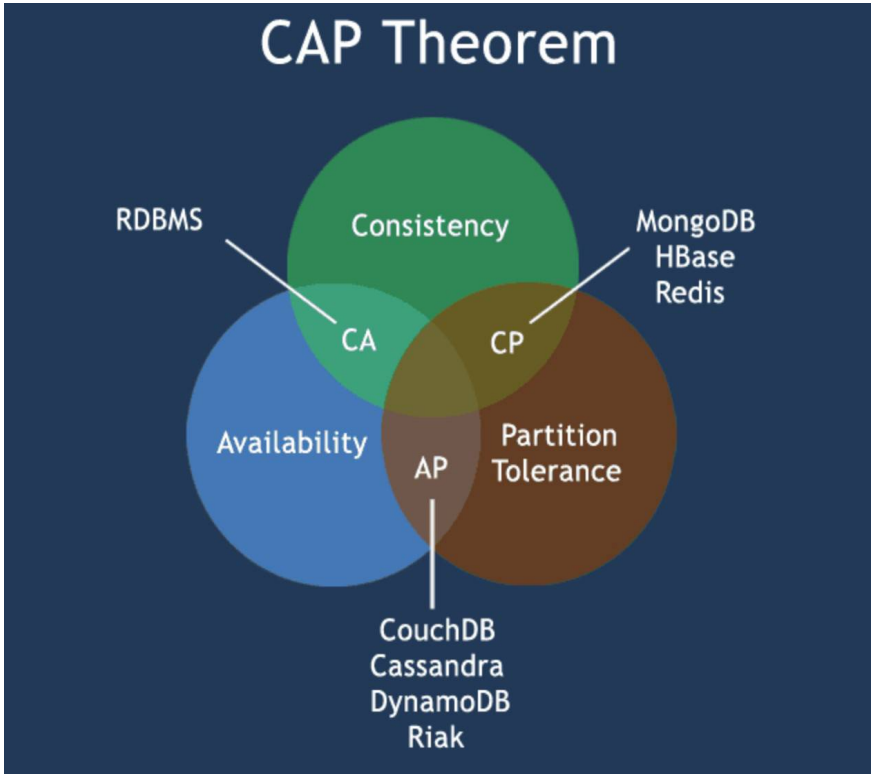
ACID-Konzept

- Atomicy — Nicht unterbrechbar
- Consistency — Daten bleiben konsistent
- Isolation — Isoliert von anderen Transaktionen
- Durability — Dauerhaft gespeichert

No-SQL hat schwächere Garantien

BASE

- Basically Available
- Soft state
- Eventual consistency



https://www.w3resource.com/mongodb/nosql.php

Unterstützt No-SQL überhaupt Transaktionen?

MongoDB

- Dokumentenspeicher
- https://www.mongodb.com/docs/manual/core/transactions/
  - *An operation on a single document is atomic*
  - *MongoDB supports multi-document transactions*
  - *With distributed transactions, transactions can be used across multiple operations, collections, databases, documents, and shards*

Transaktionen in MongoDB

```
// Traditioneller NoSQL Ansatz (Denormalisiert)
db.myCollection.insertMany( [
    { "name": "Anton", "abteilung": "Einkauf" },
    { "name": "Berta", "abteilung": "Verkauf" } ] )

// Traditioneller SQL Ansatz (Normalisiert)
abt_einkauf = db.abteilung.insertOne( { "bezeichnung": "Einkauf" } )
db.mitarbeiter.insertOne( { "name": "Anton", "abt_id": abt_einkauf.insertedId } )

abt_verkauf = db.abteilung.insertOne( { "bezeichnung": "Verkauf" } )
db.mitarbeiter.insertOne( { "name": "Berta", "abt_id": abt_verkauf.insertedId } )
```

Transaktionen in MongoDB

```
// Start a session.
session = db.getMongo().startSession( { readPreference: { mode: "primary" } } );
mitarbeiter = session.getDatabase("db").mitarbeiter;
abteilung = session.getDatabase("db").abteilung;

// Start a transaction
session.startTransaction( { readConcern: { level: "local" }, writeConcern: { w: "majority" } } );

// Operations inside the transaction
try {
   abt_marketing = abteilung.insertOne( { bezeichnung: "Marketing" } );
   mitarbeiter.insertOne( { name: "Carla", "abt_id": abt_marketing.insertedId } );
} catch (error) {
   // Abort transaction on error
   session.abortTransaction();
   throw error;
}

// Commit the transaction using write concern set at transaction start
session.commitTransaction();
session.endSession();
```