B1G

Ich kann ein Datenmodell für eine NoSQL Datenbank interpretieren und erläutern.

Fragenstellung und Lernziele

- Die Struktur eines Dokuments in einer dokumentenorientierten NoSQL-Datenbank verstehen.
- Eingebettete Dokumente und Arrays innerhalb eines Dokuments identifizieren und deren Zweck erläutern.
- Die Unterschiede zwischen dokumentenorientierten NoSQL-Datenbanken und relationalen Datenbanken hinsichtlich der Datenmodellierung beschreiben.
- Die Vorteile der flexiblen Schema-Struktur von NoSQL-Datenbanken für verschiedene Anwendungsfälle bewerten.

Umsetzung

Dokumentenorientierte NoSQL-Datenbanken speichern Daten in einer flexiblen, hierarchischen Struktur. Anstelle von Tabellen mit festen Spalten und Zeilen verwenden sie **Dokumente**, die meist im **JSON-Format** gespeichert werden. Diese Dokumente enthalten verschiedene **Schlüssel-Wert-Paare** und können geschachtelte Strukturen wie **eingebettete Dokumente** und **Arrays** enthalten.

Wichtige Begriffe und Strukturen

Um die Struktur einer dokumentenorientierten NoSQL-Datenbank zu verstehen, müssen einige zentrale Konzepte betrachtet werden:

- 📜 **Dokument** → Ein einzelner Datensatz, gespeichert als JSON-Objekt mit verschiedenen Attributen.
- 📂 **Collection** → Eine Sammlung von Dokumenten ähnlicher Art, vergleichbar mit einer Tabelle in relationalen Datenbanken.
- 📎 **Eingebettetes Dokument** → Ein Dokument, das innerhalb eines anderen Dokuments gespeichert ist. Dadurch können zusammengehörige Daten direkt in einem Eintrag gespeichert werden.
- 📑 **Array** → Eine Liste von Werten oder Dokumenten innerhalb eines Dokuments. Dies ermöglicht die Speicherung von mehreren Objekten innerhalb eines Feldes.
- 🔑 **Schlüssel-Wert-Paar** → Grundlegendes Element eines Dokuments, bestehend aus einem „Schlüssel“ und einem entsprechenden „Wert“.

Beispiel eines Dokuments in einer „Benutzer“-Collection

Ein praktisches Beispiel zeigt die Struktur eines Dokuments in einer NoSQL-Datenbank:

```
// Collection: Benutzer
{                                                  // Dokument für "Max Muster
  "benutzer_id": "12345",                          // Schlüssel-Wert-Paar "ben
  "name": "Max Mustermann",
  "email": "<max.mustermann@example.com>",
  "adresse": {                                     // Eingebettetes Dokument "
    "strasse": "Musterstrasse 1",
    "stadt": "Musterstadt",
    "plz": "12345"
  },
  "bestellungen": [                                // Array "bestellungen"
    {
      "bestell_id": "98765",
      "datum": "2025-03-01",
      "betrag": 99.99,
      "artikel": ["Buch", "Stift"]           // Array von Werten
    },
    {
      "bestell_id": "98766",
      "datum": "2025-03-15",
      "betrag": 49.99,
      "artikel": ["Notizbuch"]
    }
  ]
}
```

Hierbei sind mehrere NoSQL-spezifische Strukturen erkennbar:

- **Eingebettetes Dokument**: Die Adresse des Benutzers ist als Unterobjekt innerhalb des Hauptdokuments gespeichert.
- **Array**: Das Feld `bestellungen` enthält eine Liste von Bestellobjekten, was eine 1:n-Beziehung direkt im Dokument abbildet.

Diese Struktur ermöglicht eine **schnelle und effiziente Abfrage**, da alle relevanten Informationen direkt in einem Dokument gespeichert sind.

Unterschiede zwischen dokumentenorientierten NoSQL- und relationalen Datenbanken

Die wesentlichen Unterschiede zwischen NoSQL- und relationalen Datenbanken lassen sich anhand verschiedener Aspekte aufzeigen:

Merkmal	**Dokumentenorientierte NoSQL-DB**	**Relationale DB**
Datenstruktur	JSON-Dokumente, eingebettete	Tabellen mit Zeilen und

	Objekte, Arrays	Spalten
Schema	Flexibel, keine feste Struktur	Fest definiert (Spalten müssen vorab definiert sein)
Beziehungen	Daten können eingebettet sein	Normalisierte Tabellen mit Fremdschlüsseln
Skalierung	Horizontal (mehrere Server)	Meist vertikal (leistungsstärkere Hardware)
Performance für Lesezugriffe	Sehr effizient für komplexe Objekte	Häufig JOIN-Operationen notwendig

Während relationale Datenbanken für **strukturierte Daten** mit klaren Beziehungen optimiert sind, bieten dokumentenorientierte NoSQL-Datenbanken mehr **Flexibilität** und **Skalierbarkeit**.

Vorteile

Flexibilität in der Datenmodellierung

NoSQL-Datenbanken benötigen kein **festes Schema**. Dadurch können neue Felder hinzugefügt oder bestehende geändert werden, ohne dass komplexe **Schema-Migrationen** erforderlich sind. Dies ist besonders nützlich in agilen Entwicklungsprozessen, in denen sich Anforderungen häufig ändern.

Schnellere Lesezugriffe durch optimierte Abfragen

Da alle relevanten Daten in einem Dokument gespeichert sind, können Abfragen **schneller** ausgeführt werden, da keine **JOIN-Operationen** erforderlich sind. Besonders bei Anwendungen mit **hohem Leseaufkommen** (z. B. Webanwendungen, Echtzeitanalysen) ist dies ein grosser Vorteil.

Skalierbarkeit für grosse Datenmengen

NoSQL-Datenbanken unterstützen **horizontale Skalierung**, d. h. sie können einfach auf mehrere Server verteilt werden. Dies ist essenziell für Anwendungen mit **grossen Datenmengen** und **hohem Benutzeraufkommen**, z. B.:

- **Social Media Plattformen** (dynamische User-Daten, Kommentare, Likes)
- **E-Commerce Systeme** (Produktkataloge, Nutzerverhalten, Transaktionen)
- **IoT-Anwendungen** (Sensor-Daten, Echtzeitverarbeitung)

Flexibilität in der Speicherung von unstrukturierten und semi-strukturierten Daten

Im Gegensatz zu relationalen Datenbanken, die ein **striktes Schema** erfordern, können dokumentenorientierte NoSQL-Datenbanken **heterogene Daten** speichern. Dies ist ideal für:

- **Log-Daten** (verschiedene Formate, je nach Quelle)
- **Benutzergenerierte Inhalte** (Kommentare, Bewertungen, Posts)

- **Maschinengenerierte Daten** (IoT- und Sensordaten)

Fazit

Dokumentenorientierte NoSQL-Datenbanken bieten eine **hohe Flexibilität**, **effiziente Abfragen** und **gute Skalierbarkeit**, insbesondere für Anwendungen mit **dynamischen Datenmodellen** oder **grossen Datenmengen**. Sie eignen sich besonders gut für **Webanwendungen, Big Data und Echtzeitanalysen**, während relationale Datenbanken weiterhin eine gute Wahl für **transaktionsbasierte Systeme** mit **komplexen Beziehungen** sind.

Nachweis