2026

Software und Plattform Architektur

DOCUMENT DATENBANKEN

ANGELO LUCIANO & DONIKA HAJDARAJ

Inhaltsverzeichnis

Ausgangslage und Auftrag

Datenbanken bilden das Fundament moderner Anwendungen, indem sie Daten strukturiert und performant speichern, verwalten und abrufbar machen. Es existieren verschiedene Datenbanktypen, die je nach Anwendungsfall und Anforderungen ihre Stärken ausspielen.

Die vorliegende Aufgabe fordert die Studierenden dazu auf, einen bestimmten Datenbanktyp zu recherchieren, die wichtigsten Aspekte zu präsentieren und ein praktisches Beispiel eines konkreten Produkts aus diesem Typ zu demonstrieren. Ziel ist es, ein tiefgehendes Verständnis für unterschiedliche Datenbanktechnologien zu entwickeln und deren Einsatzgebiete sowie Funktionsweisen kennenzulernen.

(Quelle: Auftrag Patrick Michel)

Was sind Dokument Datenbanken?

Moderne Anwendungen müssen grosse Mengen unterschiedlicher Daten flexibel, schnell und zuverlässig verarbeiten können. Klassische relationale Datenbanken stossen dabei bei stark variierenden oder sich häufig ändernden Datenstrukturen an ihre Grenzen. Document-Datenbanken gehören zur Familie der NoSQL-Datenbanken und wurden entwickelt, um genau diese Anforderungen zu erfüllen. Sie speichern Daten nicht in Tabellen, sondern in Dokumenten, meist im JSON- oder BSON-Format. Document-Datenbanken speichern Informationen als Dokumente, die aus Schlüssel-Wert-Paaren bestehen und hierarchisch aufgebaut sein können (verschachtelte Strukturen).

Eigenschaften:

- Jedes Dokument ist in sich abgeschlossen.
- NoSQL – Not only Structured Query Language
- Keine feste Tabellenstruktur notwendig.
- Unterschiedliche Dokumente in derselben Sammlung können unterschiedliche Felder besitzen.
- Kein komplizierter Joint wie in komplexen SQL.417
- Atomaren Transaktionen

Vorteile und Nachteile

Vorteile	Nachteile
Flexibilität bei wechselnden Datenstrukturen	Weniger standardisiert als relationale Datenbanken
Gute Skalierbarkeit (horizontal), Sharding	Weniger geeignet für komplexe relationale Abfragen
Hohe Performance beim Zugriff auf komplette Dokumente	Teilweise geringere Konsistenz
Entwicklerfreundlich durch JSON-ähnliche Struktur	ACID Atomicity Consistency Isolation Durability

Tabelle 1 Vorteile und Nachteile von dokumentorientierten Datenbanken NoSQL

Einsatzgebiete/ Beispiele für Document-Datenbanken

Einsatzgebiete:

- Web- und Mobile-Anwendungen
- Content-Management-Systeme
- E-Commerce (Web-Shops / Katalogen)
- Benutzerprofile
- IoT- und Event-Daten

Beispiele für Document-Datenbanken:

- MongoDB
- CouchDB
- RavenDB

Recherche, Vergleich von zwei Produkten

Wir haben uns dazu beschlossen über MongoDB und CouchDB zu informieren.

MongoDB

Grafik 1 MongoDB - Logo

MongoDB ist eine weit verbreitete Open-Source Document-Datenbank, die Daten im BSON-Format speichert. Sie ist schemaflexibel, horizontal skalierbar und wird häufig in Web- und Cloud-Anwendungen eingesetzt. MongoDB bietet leistungsfähige Abfragefunktionen, Indizierung, Replikation und Sharding.

Wir werden eine Demo mit einem flexiblen RPG-Inventar, mit Postman für API-Anfragen und MongoDB Compass, um die Flexibilität zu zeigen.

Feature	JSON	BSON
Format Type	Text-based (human-readable)	Binary-based (machine-readable)
Readability	easily readable by humans	not human-readable
Data Types Supported	Strings, numbers, booleans, arrays, null	All JSON types + Date, Binary, ObjectId, and others
Space Efficiency	Less efficient for complex data types	More efficient due to binary encoding
Performance	Slower parsing and retrieval	Faster parsing and retrieval, especially for large data
Use Case	Data exchange between client and server	Data storage, especially in MongoDB
Compression	No built-in compression	More compact due to binary encoding
Support for Large Data	JSON may struggle with very large data	BSON supports large documents and binary data natively

Tabelle 2 JSON vs BSON - geeksforgeeks.org

CouchDB

Grafik 2 Apache CouchDB - Logo

Apache CouchDB speichert Daten als JSON-Dokumente und stellt eine REST-API bereit. Ein zentrales Merkmal von CouchDB ist die eingebaute Replikation: Daten können automatisch zwischen Servern und Clients synchronisiert werden, auch wenn diese zeitweise offline sind, was CouchDB für verteilte Systeme und mobile Apps sehr geeignet macht. Sprich, bei unstabilen Netzwerken ist CouchDB besonders empfohlen.

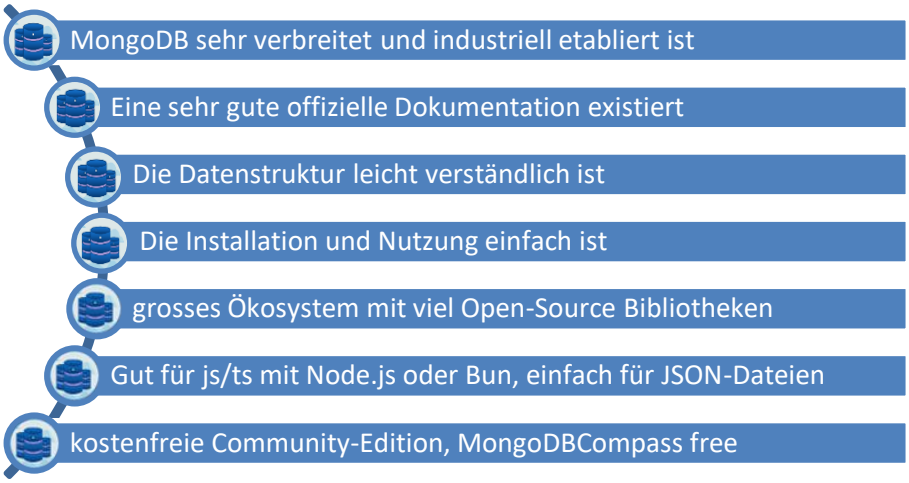
MongoDB und CouchDB sind beide dokumentenorientierte Datenbanken und speichern Daten im JSON-Format, unterscheiden sich jedoch deutlich in ihrer Ausrichtung.

MongoDB ist auf hohe Performance, Skalierbarkeit und komplexe Abfragen optimiert. Es bietet eine leistungsfähige Query-Sprache, Indizes und Aggregationen und wird häufig für Web-APIs, Microservices, E-Commerce-Plattformen und datenintensive Anwendungen eingesetzt.

CouchDB legt den Fokus auf Datenreplikation und Offline-Fähigkeit.
Die Kommunikation erfolgt ausschliesslich über HTTP/JSON, was die Integration vereinfacht. CouchDB eignet sich besonders für Offline-First-Anwendungen, Mobile Apps und Systeme, bei denen Daten zwischen mehreren Geräten synchron gehalten werden müssen.

Begründung der Produktauswahl für die Demo

Für die Demo wurde MongoDB ausgewählt, da:



Warum spricht dieser Code für MongoDB?

```
const customer = {
  name: "Anna Müller",
  email: "anna@example.com",
  address: {
    street: "Hauptstrasse 5",
    city: "Berlin"
  },
  orders: [
    { product: "Smartphone", price: 699 },
    { product: "Kopfhörer", price: 199 }
  ]
};

// direkt als Dokument speichern
db.customers.insertOne(customer);
```

Grafik 3 JSON-Code-Abschnitt - Google Image Search

- Das Objekt im Code ist fast identisch mit dem Dokument in der Datenbank
- Keine Umwandlung in Tabellen notwendig
- Verschachtelte Daten (Adresse, Bestellungen) sind direkt enthalten
- Sehr wenig Code, leicht verständlich

Demo – Kurzbeschreibung

- Code Struktur Idee mittels mehrere Google Recherchen, YouTube-Tutorials und KI gefunden und initialisiert.
- JS-Code Struktur auf eigene Bedürfnisse und Zwecke beschrieben und angepasst.
- Für RPG-Inventar – Spieler Initialisierung. Nur trinkbare Items für HP dürfen gebraucht werden.
- Erste Tests.
- Anpassung Code und Erweiterung mit neuen Ideen → Alle Objekten nutzbar über die Objekt-ID.
- Tests und Troubleshooting → Test erfolgslos. Neue Funktionen nur halb funktionsfähig. Items sollen je nach Typ anders abgezogen bzw. updatet werden. Bei (item == 0) hat die If-Iteration Item nicht aus Inventar gelöscht, da 0 in JS als false – typ boolean angesehen wird.
- Zweiter Test, statt Item löschen, haben wir Spieler und alles gelöscht. Lösung → statt Players.deleteOne(),
  Players.updateOne() mit $pull {} eingesetzt
- Anpassung: Löschung läuft sauber → Neue Idee, spezielle Items, Code soll anders reagieren. Zweite Idee, alle CRUD-Operationen integrieren: delete um Items zu löschen und put um Items hinzufügen. Tests halbwegs funktionsfähig. **Troubleshooting**: ebenfalls Players.updateOne-Funktion statt Players.deleteOne-Funktion. Für put Anfrage direkt auf `$push: {inventory: newItem}`, um Items einfach mit den verschiedenen Eigenschaften hinzufügen, ohne Klammern.
- Readme – commitet und gepusht
- Bun Installation und Test mit bun → Code unverändert. Erfolgreich!
- Faker Installation und Import, um zufällig Player-Erzeugung mit gefülltem Inventar (random items) → pendent: nice to have
- Einfügen von docker compose (yml) für One-Command-Start→ docker compose up. Erfolgslos (Code startet nicht). Troubleshooting noch pendent

Präsentation mittels PowerPoint.
Demo wird mittels Docker Desktop, Postman für API-http-Abfragen und MongoDB Compass ausgeführt und gezeigt.

Grafik 6 Logo Docker Desktop

Grafik 5 Logo Postman API

Grafik 4 Logo MongoDB Compass

Quellenverzeichnis

- MongoDB Dokumentation: https://www.mongodb.com/docs/
- Tabelle:https://www.geeksforgeeks.org/javascript/difference-between-json-and-bson/
- Apache CouchDB Dokumentation: https://docs.couchdb.org/
- IBM Document Databases: https://www.ibm.com/think/topics/nosql-databases
- AWS NoSQL Document Databases: https://aws.amazon.com/nosql/document/
- DB-Ranking: https://db-engines.com/en/ranking/document+store

Abbildungsverzeichnis

Tabellenverzeichnis