Document Stores: JSON Persistenz, Queries & Offline-Sync

Willkommen zur dritten Session! Nachdem wir in Session 1 die Grenzen von Flat Files erlebt und in Session 2 Key-Value Stores als erste Lösung kennengelernt haben, machen wir heute den nächsten evolutionären Schritt: Document Stores. JSON wird nicht mehr nur als Datenformat verwendet, sondern als natives Datenmodell persistiert – mit all seinen verschachtelten Strukturen, Arrays und flexiblen Schemas.

Rückblick Sessions 1–2:

> - **Session 1:** CSV/JSON als Flat Files – flexibel, aber ohne Abfragemechanismen
> - **Session 2:** Key-Value Stores – O(1) Zugriff per Schlüssel, aber keine strukturierten Queries

Die zentrale Frage heute:

> Wie speichern und durchsuchen wir **strukturierte, verschachtelte Daten** effizient?

Document Stores sind die natürliche Evolution von Key-Value Systemen: Statt opake Werte zu speichern, versteht die Datenbank die Struktur der Dokumente. JSON-Objekte werden nicht mehr als Strings abgelegt, sondern als First-Class Citizens behandelt. Das ermöglicht Queries auf verschachtelte Felder, Indexierung von Objekt-Properties und flexible Schema-Evolution – ohne die Flexibilität von NoSQL aufzugeben.

🎯 Lernziele dieser Session

Nach dieser Session können Sie:

1. **Document Store Konzepte** erklären und von Key-Value Stores abgrenzen
2. **PouchDB im Browser** nutzen für lokale Datenpersistenz
3. **Mango-Queries** schreiben (Selektoren, logische Operatoren, verschachtelte Felder)
4. **Index-Strategien** anwenden für Performance-Optimierung
5. **Offline-First Synchronisation** verstehen und Konflikte auflösen
6. **Use Cases** bewerten: Wann Document Store, wann Alternative?

Block 1: Document Store Grundlagen

Was ist ein Document Store?

Starten wir mit den Basics: Was macht einen Document Store aus? Der Kernunterschied zu Key-Value Stores liegt nicht nur darin, dass wir JSON speichern – das könnten wir auch in Redis. Der Unterschied ist, dass die Datenbank die JSON-Struktur versteht und darauf operieren kann.

Definition:

> Ein **Document Store** ist eine NoSQL-Datenbank, die semi-strukturierte Dokumente (meist JSON/BSON) als atomare Einheiten speichert und durchsuchbar macht.

Kernmerkmale:

- **Dokument = Atomare Einheit**: Ein JSON-Objekt ist die kleinste Speichereinheit
- **Schema-optional**: Dokumente können unterschiedliche Felder haben
- **Verschachtelung nativ**: Arrays und Objekte sind First-Class Citizens
- **Sekundäre Indizes**: Abfragen auf beliebige Felder, nicht nur den Key

Document Store vs. Key-Value Store

Der entscheidende Unterschied: In einem Key-Value Store ist `{"name": "Alice", "age": 30}` nur ein String. In einem Document Store versteht das System, dass dort ein Objekt mit Feldern `name` und `age` liegt – und Sie können direkt danach suchen.

Aspekt	Key-Value Store	Document Store
Wert-Typ	Opak (String, Binary)	Strukturiert (JSON/BSON)
Abfragen	Nur per Key	Per Key **und** Felder
Indizes	Primärschlüssel	Primär + Sekundär
Schema	Keine Validierung	Optional validierbar
Typisches Beispiel	Redis, Memcached	MongoDB, CouchDB, PouchDB

Analogie:

- **KV:** Schließfach – Sie brauchen den Schlüssel, Inhalt ist egal
- **Document:** Bibliothekskatalog – Suche nach Autor, Titel, Jahr, …

Diese Flexibilität hat ihren Preis: Document Stores sind komplexer und oft langsamer als reine Key-Value Stores. Aber sie lösen ein fundamentales Problem: Wie finde ich alle Nutzer über 18? Wie finde ich alle Produkte in Kategorie „Electronics“? In KV müssten Sie alle Keys kennen oder alle Werte laden und filtern. In Document Stores nutzen Sie Queries.

Operationen in Document Stores

Document Stores bieten eine Reihe von Grundoperationen, die über die klassischen Key-Value-Funktionen hinausgehen. Im Folgenden werden die wichtigsten Operationen vorgestellt und jeweils mit einem Sprecherkommentar erläutert.

get: Ein Dokument anhand seiner ID abrufen.

Mit der get-Operation können Sie gezielt ein einzelnes Dokument aus der Datenbank laden, sofern Sie die eindeutige ID kennen. Dies entspricht dem klassischen Zugriff per Schlüssel im Key-Value Store, ist aber auf komplexe Dokumente anwendbar.

set/put: Ein neues Dokument speichern oder ein bestehendes aktualisieren.

Die put-Operation erlaubt es, ein Dokument in der Datenbank zu speichern. Existiert bereits ein Dokument mit derselben ID, wird es überschrieben. So können Sie Daten flexibel anlegen und aktualisieren.

delete: Ein Dokument anhand seiner ID löschen.

Mit delete entfernen Sie ein Dokument dauerhaft aus der Datenbank. Diese Operation ist wichtig, um veraltete oder nicht mehr benötigte Daten zu bereinigen.

exist: Prüfen, ob ein Dokument existiert (meist über get mit Fehlerbehandlung).

Die Existenzprüfung erfolgt oft indirekt, indem versucht wird, ein Dokument zu laden. Schlägt dies fehl, existiert das Dokument nicht. Manche Systeme bieten dafür eine eigene Methode.

find/query: Dokumente nach beliebigen Feldern oder Bedingungen suchen.

Die mächtigste Operation in Document Stores ist die Query-Funktion. Sie können nach beliebigen Feldwerten, Bedingungen oder Kombinationen suchen – weit über den reinen Key hinaus. Das macht Document Stores besonders flexibel für komplexe Anwendungsfälle.

bulk operations: Mehrere Dokumente gleichzeitig speichern, abrufen oder löschen.

Für Performance und Effizienz bieten Document Stores oft Bulk-Operationen, mit denen Sie viele Dokumente in einem Schritt verarbeiten können. Das ist besonders bei Migrationen oder großen Datenmengen hilfreich.

update/patch: Teilweise Aktualisierung eines Dokuments (je nach System).

Manche Document Stores erlauben es, nur bestimmte Felder eines Dokuments zu ändern, ohne das gesamte Dokument neu zu schreiben. Das spart Ressourcen und vereinfacht die Datenpflege.

allDocs/list: Alle Dokumente oder IDs auflisten.

Mit dieser Operation können Sie sich einen Überblick über alle gespeicherten Dokumente verschaffen, etwa für Analysen oder Verwaltungsaufgaben.

Historische Entwicklung der Document Stores

Document Stores sind heute ein zentraler Bestandteil moderner Datenbanklandschaften. Doch wie entstand dieses Paradigma?

Frühe Wurzeln

- Bereits in den 1970er/80er Jahren wurden semi-strukturierte Daten (z. B. SGML, später XML) genutzt, um komplexe Dokumente zu speichern.
- Hierarchische und Netzwerk-Datenbanken (z. B. IMS, CODASYL) boten erste Ansätze für flexible Strukturen.

Die Idee, Daten als „Dokumente“ mit variabler Struktur zu speichern, war eine Antwort auf die Limitierungen relationaler Modelle bei komplexen, unregelmäßigen Daten.

SGML (Standard Generalized Markup Language) ist eine Metasprache, die 1986 als ISO-Standard veröffentlicht wurde. Sie erlaubt es, die Struktur von Texten durch frei definierbare Tags zu beschreiben und ist der Vorläufer von HTML und XML. SGML trennt Inhalt und Struktur, sodass komplexe Dokumente wie technische Handbücher oder Gesetzestexte flexibel modelliert werden können. Die Dokumenttypdefinition (DTD) legt fest, welche Elemente und Verschachtelungen erlaubt sind; Parser prüfen die Einhaltung dieser Regeln.

IMS (Information Management System) ist ein hierarchisches Datenbanksystem, das von IBM bereits 1966 entwickelt wurde. IMS speichert Daten in Baumstrukturen, wobei jedes „Parent“-Element beliebig viele „Child“-Elemente haben kann. Diese Struktur eignet sich besonders für Anwendungen mit klaren Hierarchien, etwa Stücklisten oder Organisationsstrukturen, ist aber weniger flexibel für Querverbindungen.

CODASYL war ein Konsortium, das in den 1970er Jahren das Netzwerkdatenbankmodell entwickelte. Im Gegensatz zu IMS erlaubt CODASYL komplexe Beziehungen zwischen Datensätzen („Records“) über sogenannte „Sets“. Dadurch können beliebige Verknüpfungen modelliert werden, etwa viele-zu-viele-Beziehungen. Die Navigation erfolgt explizit über Pointer, was flexible, aber oft schwer wartbare Strukturen ergibt.

Web-Ära & XML/JSON

- Mit dem Siegeszug des Webs wurden XML und später JSON zu Standardformaten für Datenaustausch.
- XML-Datenbanken (z. B. eXist-db, BaseX) und erste JSON-basierte Systeme entstanden.

Die Flexibilität von XML/JSON ermöglichte es, Daten ohne starres Schema zu speichern und zu übertragen – ideal für Web-APIs und Microservices.

eXist-db ist eine Open-Source-XML-Datenbank, die speziell für die Speicherung, Suche und Abfrage von XML-Dokumenten entwickelt wurde. Sie unterstützt die Abfragesprachen XPath und XQuery, mit denen komplexe, hierarchische Datenstrukturen effizient durchsucht werden können. eXist-db bietet Features wie Volltextsuche, Indexierung, REST- und WebDAV-Schnittstellen und wird häufig für digitale Bibliotheken, wissenschaftliche Publikationen und Projekte im Bereich Digital Humanities eingesetzt.

BaseX ist ebenfalls eine leistungsfähige Open-Source-XML-Datenbank und XQuery-Prozessor. Sie ist besonders auf hohe Geschwindigkeit und effiziente Speicherverwaltung ausgelegt. BaseX unterstützt XPath, XQuery und XSLT, bietet eine grafische Benutzeroberfläche, REST-API und verschiedene Integrationsmöglichkeiten. Typische Einsatzgebiete sind die Analyse und Verwaltung großer XML-Datensätze, z. B. für wissenschaftliche Daten, technische Dokumentationen oder Web-Content-Management.

Moderne Document Stores

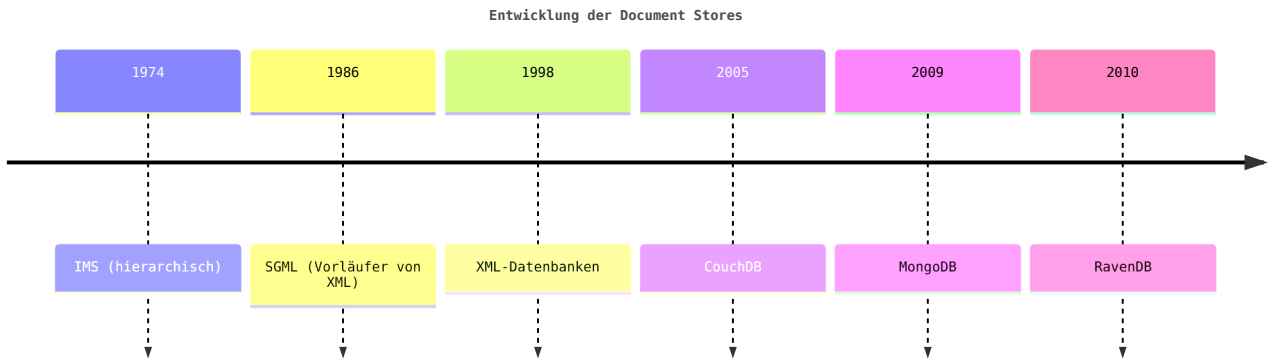
- Ab ca. 2005: CouchDB, MongoDB, RavenDB als Pioniere.
- Fokus: Skalierbarkeit, flexible Schemata, einfache Replikation und horizontale Verteilung.

Document Stores wurden zum Rückgrat für Content Management, User Profiles, Logdaten und viele Cloud-Anwendungen.

CouchDB (cluster of unreliable commodity hardware) ist eine Open-Source-Dokumentendatenbank, die 2005 von Damien Katz entwickelt wurde. CouchDB speichert Daten in JSON-Dokumenten und verwendet JavaScript für Abfragen und MapReduce-Operationen. Ein zentrales Merkmal von CouchDB ist sein Multi-Master-Replikationsmodell, das es ermöglicht, Datenbanken über verschiedene Server hinweg zu synchronisieren und offline zu arbeiten. CouchDB eignet sich besonders für verteilte Systeme, mobile Anwendungen und Szenarien, in denen Datenkonsistenz flexibel gehandhabt werden muss.

MongoDB ist eine der bekanntesten NoSQL-Datenbanken und wurde 2009 veröffentlicht. Sie speichert Daten in flexiblen, JSON-ähnlichen Dokumenten (BSON) und bietet eine leistungsfähige Abfragesprache, sekundäre Indizes und Aggregationsframeworks. MongoDB ist für seine Skalierbarkeit und einfache Handhabung bekannt und wird häufig in Webanwendungen, Big Data und Echtzeit-Analysen eingesetzt.

RavenDB ist eine dokumentenorientierte NoSQL-Datenbank, die 2010 von Oren Eini entwickelt wurde. Sie ist in C# geschrieben und bietet eine enge Integration mit dem .NET-Ökosystem. RavenDB unterstützt ACID-Transaktionen, sekundäre Indizes, Volltextsuche und Replikation. Ein besonderes Merkmal von RavenDB ist die automatische Indexerstellung, die es Entwicklern ermöglicht, Abfragen zu schreiben, ohne sich um die Indexierung kümmern zu müssen. RavenDB wird häufig in Unternehmensanwendungen eingesetzt, die hohe Anforderungen an Konsistenz und Performance stellen.



Der Zeitstrahl zeigt: Document Stores sind eine Antwort auf die wachsende Vielfalt und Komplexität von Daten im Web-Zeitalter.

> **Reflexionsfrage:** Warum sind Document Stores besonders für moderne Web- und Cloud-Anwendungen geeignet? Überlegt, welche Vorteile flexible Schemata und JSON-Dokumente bieten.

PouchDB: Document Store im Browser

Warum PouchDB für diese Vorlesung?

PouchDB wurde 2012 von Dale Harvey entwickelt, inspiriert von CouchDB und dem Ziel, eine Datenbank direkt im Browser bereitzustellen. Die Idee war, Offline-First-Apps zu ermöglichen, bei denen Daten lokal gespeichert und später mit einem Server synchronisiert werden können. PouchDB ist Open Source und hat sich schnell als Standard für Web-Apps etabliert, die Synchronisation und Flexibilität benötigen. Die Architektur orientiert sich stark am Replikationsmodell von CouchDB, wodurch eine nahtlose Integration und Synchronisation zwischen Client und Server möglich ist.

- ✅ Läuft nativ im Browser (keine Server-Installation)
- ✅ CouchDB-kompatibel (Sync zu Remote-DB möglich)
- ✅ Offline-First Design (perfekt für moderne Web-Apps)
- ✅ Mango-Query-Language (MongoDB-ähnlich)
- ✅ IndexedDB als Storage-Backend

Setup (JavaScript):

Führen Sie den folgenden Code aus und betrachten Sie die Ausgabe. Es gibt ein weiteres Feld `"_rev"` – das ist die Revisionsnummer, die PouchDB automatisch verwaltet, um Versionskonflikte zu handhaben. Die Revisionsnummer besteht aus zwei Teilen: einer Zahl, die die Version des Dokuments angibt, und einem Hash-Wert, der eine eindeutige Kennung für die spezifische Version des Dokuments darstellt. Jedes Mal, wenn ein Dokument aktualisiert wird, erhöht sich die Zahl um eins, und ein neuer Hash wird generiert. Dies ermöglicht es PouchDB, Konflikte zu erkennen und zu verwalten, wenn mehrere Clients gleichzeitig Änderungen an demselben Dokument vornehmen.

```
// Datenbank erstellen
const db = new PouchDB('my_database', {adapter: 'memory'});

// Dokument einfügen
await db.put({
  _id: 'user_alice',
  name: 'Alice',
  age: 30,
  email: 'alice@example.com'
});

let user = await db.get('user_alice');
console.log(JSON.stringify(user, null, 2));

await db.destroy(); // Datenbank löschen (optional)
```

```
{
  "name": "Alice",
  "age": 30,
  "email": "alice@example.com",
  "_id": "user_alice",
  "_rev": "1-12824c781251191b08637ef8f0e46711"
}
```

Versuchen Sie, das Dokument erneut mit `db.put()` zu speichern, ohne die `_rev`-Nummer zu aktualisieren. Sie sollten einen Fehler erhalten „Document update conflict“, da bereits eine Version dieses Dokuments existiert. Um das Dokument erfolgreich zu aktualisieren, müssen Sie die aktuelle (alte) `_rev`-Nummer angeben.
PouchDB kann Daten auch in IndexedDB – einer Browser-nativen Key-Value API. Aber PouchDB abstrahiert die Komplexität und bietet ein Document-Modell. Jedes Dokument braucht eine `_id` (ähnlich wie ein Key), kann aber beliebige weitere Felder haben. Das Präfix-Underscore (`_id`, `_rev`) markiert System-Felder.

Live-Demo: Große Schritte mit PouchDB

Führen sie den folgenden Code aus, um mehrere Dokumente auf einmal zu speichern und abzurufen. Beachten Sie, wie PouchDB automatisch die `_id`-Felder verwendet und die verschachtelten Strukturen (Arrays) direkt speichert. In diesem Beispiel verwenden wir die zuvor geladenen Produktdaten aus der CSV-Datei.

```
1  // Datenbank erstellen
2  const db = new PouchDB('lecture_demo', {adapter: 'memory'});
3
4  // Shop-Daten einfügen
5  await db.bulkDocs(window.shopData);
6
7  // Dokument abrufen
8  const doc = await db.get('P00001');
9  console.log(JSON.stringify(doc, null, 2));
10
11 // Alle Dokumente auflisten
12 const result = await db.allDocs({ include_docs: true, limit: 3 });
13 console.log(JSON.stringify(result.rows, null, 2));
```

```
{
  "name": "Laptop 14\" 2.0",
  "category": "Electronics",
  "brand": "PixelPeak",
  "price": 1399.03,
  "stock": 138,
  "rating": 3.7,
  "created_at": "2025-06-14",
  "_id": "P00001",
  "_rev": "1-9f04c28afe20e1591157eab6468badeb"
}
[
  {
    "id": "P00001",
    "key": "P00001",
    "value": {
      "rev": "1-9f04c28afe20e1591157eab6468badeb"
    },
    "doc": {
      "name": "Laptop 14\" 2.0",
      "category": "Electronics",
      "brand": "PixelPeak",
      "price": 1399.03,
      "stock": 138,
      "rating": 3.7,
      "created_at": "2025-06-14",
      "_id": "P00001",
      "_rev": "1-9f04c28afe20e1591157eab6468badeb"
    }
  },
  {
    "id": "P00002",
    "key": "P00002",
    "value": {
      "rev": "1-555949b498ecee3c9636a7864a1ed37b"
    },
    "doc": {
      "name": "Portable SSD Max M",
      "category": "Electronics",
      "brand": "ZenCore",
      "price": 805.93,
      "stock": 211,
      "rating": 3.9,
```

```
      "created_at": "2024-05-21",
      "_id": "P00002",
      "_rev": "1-555949b498ecee3c9636a7864a1ed37b"
    }
  },
  {
    "id": "P00003",
    "key": "P00003",
    "value": {
      "rev": "1-2d1821be050e0dd6b60ebf520e9edec3"
    },
    "doc": {
      "name": "4K Monitor - Green",
      "category": "Electronics",
      "brand": "Neutrino",
      "price": 522.48,
      "stock": 261,
      "rating": 5,
      "created_at": "2025-10-16",
      "_id": "P00003",
      "_rev": "1-2d1821be050e0dd6b60ebf520e9edec3"
    }
  }
]
```


Was Sie beobachten sollten:

- Jedes Dokument hat automatisch `_id` und `_rev` (Revision für Versionierung)
- Verschachtelte Strukturen (Arrays) werden direkt gespeichert
- `allDocs()` gibt Metadaten + Dokumente zurück

Jedes Dokument in PouchDB besitzt automatisch die Felder `_id` (eindeutige Kennung) und `_rev` (Versionsnummer), die für die Verwaltung und Versionierung genutzt werden. Verschachtelte Strukturen wie Arrays oder Objekte werden direkt und ohne Einschränkungen gespeichert, was flexible Datenmodelle ermöglicht. Die Methode `allDocs()` liefert nicht nur die eigentlichen Dokumente, sondern auch wichtige Metadaten wie `id`, `key` und `rev`. Der key ist dabei meist identisch mit der Dokument-ID und dient als Sortier- und Suchschlüssel im Index. Er wird genutzt, um Dokumente effizient zu finden, zu sortieren oder gezielt Teilmengen abzufragen – etwa mit `startkey` und `endkey`. So kann man performant auf die Datenbank zugreifen und die gewünschten Daten extrahieren.

Block 2: Von Scans zu strukturierten Queries

Bevor wir in die Details von Mango-Queries einsteigen, sollten wir einen Moment innehalten und uns fragen: Was haben wir eigentlich gewonnen gegenüber CSV-Dateien und Key-Value Stores? In Session 1 konnten wir CSV-Dateien abfragen – aber nur durch vollständiges Durchscannen. In Session 2 lernten wir Key-Value Stores kennen – brillant für direkte Lookups, aber hilflos bei komplexen Filtern. Document Stores sind der nächste evolutionäre Schritt: Sie verstehen die Struktur Ihrer Daten und ermöglichen strukturierte Abfragen. Aber ohne Indizes zahlen wir dafür einen hohen Preis.

Wiederholung: Das Scan-Problem

Rückblick Session (CSV):

Auch wenn wir die CSV-Daten in ein Dictionary, eine assoziative Map oder eine Hash-Tabelle umwandeln, sodass wir einzelne Werte blitzschnell – meist in $O(1)$O(1), im schlechtesten Fall in $O(\log n)$O(log n) – abrufen können, bleibt ein zentrales Problem bestehen: Sobald wir nach Produkten mit stock < 10 suchen, müssen wir trotzdem jedes einzelne Produkt prüfen. Das bedeutet: Die Suche ist $O(n)$O(n) – also linear in der Anzahl der Produkte, unabhängig davon, wie effizient der Zugriff auf einzelne Werte ist.

```
function filterProductsBy(filter) {
  const start = performance.now();
  const results = [];

  // Müssen ALLE Keys durchsuchen! (wieder O(n))
  for (const productId in window.shopData) {
    sleep(1); // Simuliere Latenz
    const product = window.shopData[productId];
    if (filter(product)) {
      results.push(product);
    }
  }

  const end = performance.now();

  if (results.length > 0) {
    console.log("✅ Produkte gefunden:", JSON.stringify(results, null
  } else {
    console.log("❌ Keine Produkte gefunden");
  }

  console.log("⏱️  Zeit:", (end - start).toFixed(2), "ms");
}

filterProductsBy(p => p.stock < 10);
```

```
✅ Produkte gefunden: [
  {
    "name": "4K Monitor",
    "category": "Electronics",
    "brand": "Neutrino",
    "price": 24.92,
    "stock": 1,
    "rating": 3.3,
    "created_at": "2025-07-13",
    "_id": "P00011"
  },
  {
    "name": "Gaming Mouse Plus S",
    "category": "Electronics",
    "brand": "BlueWave",
    "price": 562.3,
    "stock": 5,
    "rating": 4,
    "created_at": "2025-07-05",
    "_id": "P00065"
  },
  {
    "name": "Portable SSD",
    "category": "Electronics",
    "brand": "Neutrino",
    "price": 968.47,
    "stock": 4,
    "rating": 3.2,
    "created_at": "2024-06-10",
    "_id": "P00085"
  },
  {
    "name": "Tablet 10\" Pro - Black",
    "category": "Electronics",
    "brand": "Voltix",
    "price": 827.84,
    "stock": 3,
    "rating": 3.4,
    "created_at": "2025-10-10",
    "_id": "P00086"
  },
  {
    "name": "Bluetooth Speaker",
```

```
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 1493.82,
    "stock": 5,
    "rating": 4,
    "created_at": "2024-09-19",
    "_id": "P00096"
  },
  {
    "name": "Dress Shirt Plus - Graphite",
    "category": "Clothing",
    "brand": "PureThread",
    "price": 198.67,
    "stock": 3,
    "rating": 2.4,
    "created_at": "2025-02-17",
    "_id": "P00132"
  },
  {
    "name": "Dress Shirt - Blue",
    "category": "Clothing",
    "brand": "AeroLoom",
    "price": 39.97,
    "stock": 3,
    "rating": 4.9,
    "created_at": "2025-03-08",
    "_id": "P00141"
  },
  {
    "name": "Whiteboard Markers (8-pack)",
    "category": "Office",
    "brand": "InkFlow",
    "price": 415.87,
    "stock": 2,
    "rating": 3.4,
    "created_at": "2025-05-16",
    "_id": "P00393"
  },
  {
    "name": "Cutlery Set (24 pcs) - Graphite",
    "category": "Home & Kitchen",
    "brand": "Hearthstone",
    "price": 130.55,
```

```
    "stock": 7,
    "rating": 3.3,
    "created_at": "2025-10-18",
    "_id": "P00416"
  },
  {
    "name": "Toaster 2-slice Max L",
    "category": "Home & Kitchen",
    "brand": "CozyNest",
    "price": 148.36,
    "stock": 3,
    "rating": 3.9,
    "created_at": "2025-01-19",
    "_id": "P00450"
  },
  {
    "name": "Micellar Water 400ml Pro",
    "category": "Beauty",
    "brand": "AurumCare",
    "price": 100.06,
    "stock": 7,
    "rating": 4.3,
    "created_at": "2024-10-24",
    "_id": "P00749"
  },
  {
    "name": "Poetry Collection Pro - White",
    "category": "Books",
    "brand": "Silver Quill",
    "price": 27.97,
    "stock": 2,
    "rating": 3.3,
    "created_at": "2025-03-09",
    "_id": "P00804"
  },
  {
    "name": "Beginners German Max",
    "category": "Books",
    "brand": "Atlas Editions",
    "price": 51.71,
    "stock": 9,
    "rating": 4.7,
    "created_at": "2024-07-27",
```

```
    "_id": "P00809"
  },
  {
    "name": "Poetry Collection",
    "category": "Books",
    "brand": "Silver Quill",
    "price": 70.25,
    "stock": 3,
    "rating": 4,
    "created_at": "2024-05-05",
    "_id": "P00854"
  },
  {
    "name": "Cookbook - Quick Meals 2.0",
    "category": "Books",
    "brand": "Northwind Press",
    "price": 53.36,
    "stock": 3,
    "rating": 4.8,
    "created_at": "2024-12-28",
    "_id": "P00878"
  },
  {
    "name": "Bird Seed 1kg Max",
    "category": "Pets",
    "brand": "WildFeast",
    "price": 93.18,
    "stock": 1,
    "rating": 4.3,
    "created_at": "2025-01-19",
    "_id": "P00902"
  }
]
⏱️  Zeit: 1000.50 ms
```


Rückblick Session (Key-Value):

Key-Value Stores sind fantastisch, wenn Sie den Schlüssel kennen und direkt darauf zugreifen möchten – das ist $O(1)$O(1). Aber sobald Sie nach Werten filtern müssen, sind Sie wieder bei $O(n)$O(n), weil Sie alle Einträge laden und prüfen müssen. Hier ein Beispiel mit Redis:

```
1  const redis = new Redis();
2
3  for (const product of window.shopData) {
4    await redis.set(product._id, JSON.stringify(product));
```

```
}

// Key-Value Store: Direkter Zugriff O(1)
const product = await redis.get('P00001');  // ✅ Schnell!

console.log(JSON.parse(product))
// Aber: Filter über Werte? Zurück zu O(n)!

const allKeys = await redis.keys('P*');
const results = [];
for (const key of allKeys) {
  const product = JSON.parse(await redis.get(key));
  if (product.stock < 10) {  // JEDES Dokument laden & prüfen!
    results.push(product);
  }
}

console.log(JSON.stringify(results, null, 2))
// Problem: Keine strukturierten Queries möglich
```

{"name":"Laptop 14\"
2.0","category":"Electronics","brand":"PixelPeak","price":1399.03,"stock":
06-14","_id":"P00001"}
[
  {
    "name": "4K Monitor",
    "category": "Electronics",
    "brand": "Neutrino",
    "price": 24.92,
    "stock": 1,
    "rating": 3.3,
    "created_at": "2025-07-13",
    "_id": "P00011"
  },
  {
    "name": "Gaming Mouse Plus S",
    "category": "Electronics",
    "brand": "BlueWave",
    "price": 562.3,
    "stock": 5,
    "rating": 4,
    "created_at": "2025-07-05",
    "_id": "P00065"
  },
  {
    "name": "Portable SSD",
    "category": "Electronics",
    "brand": "Neutrino",
    "price": 968.47,
    "stock": 4,
    "rating": 3.2,
    "created_at": "2024-06-10",
    "_id": "P00085"
  },
  {
    "name": "Tablet 10\" Pro - Black",
    "category": "Electronics",
    "brand": "Voltix",
    "price": 827.84,
    "stock": 3,
    "rating": 3.4,
    "created_at": "2025-10-10",
    "_id": "P00086"

```
  },
  {
    "name": "Bluetooth Speaker",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 1493.82,
    "stock": 5,
    "rating": 4,
    "created_at": "2024-09-19",
    "_id": "P00096"
  },
  {
    "name": "Dress Shirt Plus - Graphite",
    "category": "Clothing",
    "brand": "PureThread",
    "price": 198.67,
    "stock": 3,
    "rating": 2.4,
    "created_at": "2025-02-17",
    "_id": "P00132"
  },
  {
    "name": "Dress Shirt - Blue",
    "category": "Clothing",
    "brand": "AeroLoom",
    "price": 39.97,
    "stock": 3,
    "rating": 4.9,
    "created_at": "2025-03-08",
    "_id": "P00141"
  },
  {
    "name": "Whiteboard Markers (8-pack)",
    "category": "Office",
    "brand": "InkFlow",
    "price": 415.87,
    "stock": 2,
    "rating": 3.4,
    "created_at": "2025-05-16",
    "_id": "P00393"
  },
  {
    "name": "Cutlery Set (24 pcs) - Graphite",
```

```
    "category": "Home & Kitchen",
    "brand": "Hearthstone",
    "price": 130.55,
    "stock": 7,
    "rating": 3.3,
    "created_at": "2025-10-18",
    "_id": "P00416"
  },
  {
    "name": "Toaster 2-slice Max L",
    "category": "Home & Kitchen",
    "brand": "CozyNest",
    "price": 148.36,
    "stock": 3,
    "rating": 3.9,
    "created_at": "2025-01-19",
    "_id": "P00450"
  },
  {
    "name": "Micellar Water 400ml Pro",
    "category": "Beauty",
    "brand": "AurumCare",
    "price": 100.06,
    "stock": 7,
    "rating": 4.3,
    "created_at": "2024-10-24",
    "_id": "P00749"
  },
  {
    "name": "Poetry Collection Pro - White",
    "category": "Books",
    "brand": "Silver Quill",
    "price": 27.97,
    "stock": 2,
    "rating": 3.3,
    "created_at": "2025-03-09",
    "_id": "P00804"
  },
  {
    "name": "Beginners German Max",
    "category": "Books",
    "brand": "Atlas Editions",
    "price": 51.71,
```

```
    "stock": 9,
    "rating": 4.7,
    "created_at": "2024-07-27",
    "_id": "P00809"
  },
  {
    "name": "Poetry Collection",
    "category": "Books",
    "brand": "Silver Quill",
    "price": 70.25,
    "stock": 3,
    "rating": 4,
    "created_at": "2024-05-05",
    "_id": "P00854"
  },
  {
    "name": "Cookbook - Quick Meals 2.0",
    "category": "Books",
    "brand": "Northwind Press",
    "price": 53.36,
    "stock": 3,
    "rating": 4.8,
    "created_at": "2024-12-28",
    "_id": "P00878"
  },
  {
    "name": "Bird Seed 1kg Max",
    "category": "Pets",
    "brand": "WildFeast",
    "price": 93.18,
    "stock": 1,
    "rating": 4.3,
    "created_at": "2025-01-19",
    "_id": "P00902"
  }
]
```


Seit Redis die Möglichkeit bietet, Lua-Skripte direkt auf dem Server auszuführen, muss die Filterung nicht zwingend im Client erfolgen. Theoretisch könnten wir ein Lua-Skript schreiben, das die gewünschten Daten direkt auf dem Server filtert. Dennoch bleibt das fundamentale Problem bestehen: Ohne Kenntnis der Datenstruktur kann das System nicht effizient filtern – es muss weiterhin alle relevanten Schlüssel prüfen.

```
1  const redis = new Redis()
2
```

```
for (const product of window.shopData) {
  await redis.set(product._id, JSON.stringify(product));
}

// Lua script: Filtere Produkte mit stock < 10
// funktioniert leider nicht, da table.insert nicht korrekt funkionie
const script = `
  local result = {}
  local keys = redis.call('KEYS', 'P*')
  for i, key in ipairs(keys) do
    local json = redis.call('GET', key)
    if json then
      local stock = string.match(json, '"stock"%s*:%s*(%d+)')
      if stock and tonumber(stock) < 10 then
        table.insert(result, json) -- oder: table.insert(result, key)
          nur die Keys
      end
    end
  end
  return result
`;

const lowStockKeys = await redis.eval(script, 0);
console.log('Keys mit niedrigem Bestand:', lowStockKeys);
```

```
Keys mit niedrigem Bestand: undefined
```

Ein wichtiger Aspekt bei Redis ist, dass Lua-Skripte atomar ausgeführt werden. Das bedeutet: Während ein Skript läuft, werden alle anderen Client-Befehle blockiert – keine GET, SET, HSET oder andere Operationen können in dieser Zeit ausgeführt werden. Selbst PINGs oder Pub/Sub-Nachrichten werden nicht verarbeitet. Alle anderen Clients müssen warten; ihre Requests werden in die Warteschlange gelegt. Diese strikte Atomarität ist ein bewusstes Designziel von Redis, um Race Conditions und Zwischenzustände zu vermeiden. So erscheint jede Operation – auch komplexe Lua-Skripte – nach außen wie eine einzige, unteilbare Aktion.

> **Das fundamentale Problem:** Ohne Kenntnis der Datenstruktur kann das System nicht effizient filtern.

Document Stores: Struktur wird zum Vorteil

Der Durchbruch:

CSV / Key–Value Store:

Daten = Opak

P001,Laptop,999

System versteht:
❌ Keine Felder
❌ Keine Typen
❌ Keine Queries

→ Nur Scan möglich

→

Document Store:

Daten = Strukturiert

```
{
  _id: 'P001'
  name: 'Laptop
  price: 999
}
```

System versteht:
✅ Felder (price)
✅ Typen (Number)
✅ Queries möglich!

PouchDB kann jetzt:

```
1  // Datenbank erstellen
2  const db = new PouchDB('lecture_demo');
3  // Shop-Daten einfügen
4  await db.bulkDocs(window.shopData);
5
6  // Query auf Feldebene - keine manuelle Iteration nötig!
7  const result = await db.find({
8    selector: {
9      stock: { $lt: 10 }  // PouchDB versteht "stock" als Feld
10   }
11 });
12
13 console.log(result.docs);  // Nur passende Dokumente
```

```
[{"price":null,"stock":null,"rating":null,"_id":"09428382-96f4-4b04-
8090-8378ac335e8d","_rev":"1-82aa517d2825282ea32db861823aaee9"},
{"name":"4K
Monitor","category":"Electronics","brand":"Neutrino","price":24.92,"stock"
07-13","_id":"P00011","_rev":"1-5b8e2f5a4b3000038d95128d6683862e"},
{"name":"Gaming Mouse Plus
S","category":"Electronics","brand":"BlueWave","price":562.3,"stock":5,"ra
07-05","_id":"P00065","_rev":"1-24f8c99f1a2ff087f3322a017550987d"},
{"name":"Portable
SSD","category":"Electronics","brand":"Neutrino","price":968.47,"stock":4,
06-10","_id":"P00085","_rev":"1-080dc94abe3f149e27c1d8509d0c17b0"},
{"name":"Tablet 10\" Pro -
Black","category":"Electronics","brand":"Voltix","price":827.84,"stock":3,
10-10","_id":"P00086","_rev":"1-b39498e68a68e20adff0505d689b3d3b"},
{"name":"Bluetooth
Speaker","category":"Electronics","brand":"ZenCore","price":1493.82,"stock
09-19","_id":"P00096","_rev":"1-11e9abed0dc189c3d8d7abd687989228"},
{"name":"Dress Shirt Plus -
Graphite","category":"Clothing","brand":"PureThread","price":198.67,"stock
02-17","_id":"P00132","_rev":"1-56476784b6e2da0c21c2aecc79d7aff1"},
{"name":"Dress Shirt -
Blue","category":"Clothing","brand":"AeroLoom","price":39.97,"stock":3,"ra
03-08","_id":"P00141","_rev":"1-586e900bdf8c9f5de400efaa6440b434"},
{"name":"Whiteboard Markers (8-
pack)","category":"Office","brand":"InkFlow","price":415.87,"stock":2,"rat
05-16","_id":"P00393","_rev":"1-2a801a89df59fbcd5f4ef82f0966d509"},
{"name":"Cutlery Set (24 pcs) - Graphite","category":"Home &
Kitchen","brand":"Hearthstone","price":130.55,"stock":7,"rating":3.3,"crea
10-18","_id":"P00416","_rev":"1-7bb04e9dd280eeb1a171c4952d714072"},
{"name":"Toaster 2-slice Max L","category":"Home &
Kitchen","brand":"CozyNest","price":148.36,"stock":3,"rating":3.9,"created
01-19","_id":"P00450","_rev":"1-ccbdd338e62b71fd03986315b3b8aae8"},
{"name":"Micellar Water 400ml
Pro","category":"Beauty","brand":"AurumCare","price":100.06,"stock":7,"rat
10-24","_id":"P00749","_rev":"1-4400dd1de810a9ca4ee8d1ff4d0404e9"},
{"name":"Poetry Collection Pro -
White","category":"Books","brand":"Silver
Quill","price":27.97,"stock":2,"rating":3.3,"created_at":"2025-03-
09","_id":"P00804","_rev":"1-ab026fa94ad6391dd93ffd23909a40fc"},
{"name":"Beginners German Max","category":"Books","brand":"Atlas
Editions","price":51.71,"stock":9,"rating":4.7,"created_at":"2024-07-
27","_id":"P00809","_rev":"1-9be376ec2015d169e47603b2d6383c41"},
{"name":"Poetry Collection","category":"Books","brand":"Silver
```

```
Quill","price":70.25,"stock":3,"rating":4,"created_at":"2024-05-
05","_id":"P00854","_rev":"1-7e61c0f1beac24948513ef51b22aca1d"},
{"name":"Cookbook - Quick Meals
2.0","category":"Books","brand":"Northwind
Press","price":53.36,"stock":3,"rating":4.8,"created_at":"2024-12-
28","_id":"P00878","_rev":"1-ac13d605937df2cc3812def4dc77b11e"},
{"name":"Bird Seed 1kg
Max","category":"Pets","brand":"WildFeast","price":93.18,"stock":1,"rating
01-19","_id":"P00902","_rev":"1-22040e185ba1ff7e7935e7cbc3a4fc5d"}]
```


> **Wichtig:** Das funktioniert – aber **erstmal immer noch als Scan**! Der Unterschied: Das System übernimmt den Scan für Sie. Wirkliche Performance kommt erst mit Indizes.

Hier ist der entscheidende Punkt: Document Stores machen Queries bequemer und deklarativer – aber ohne Indizes sind sie nicht magisch schneller als Ihre CSV-Schleife. Der wahre Durchbruch kommt im nächsten Schritt: Wenn das System die Struktur kennt, kann es Indizes darauf bauen.

Block 3: Mango Query Language

Jetzt, da Sie verstehen, warum Document Stores strukturierte Queries ermöglichen und warum Indizes essentiell sind, schauen wir uns die Mango Query Language im Detail an. Mango ist JSON-basiert, deklarativ und MongoDB-Nutzern vertraut. Sie beschreiben, WAS Sie suchen – nicht WIE.

Historie der Mango Query Language

Die Mango Query Language wurde ursprünglich von der CouchDB-Community entwickelt, um eine einfachere und deklarative Alternative zu MapReduce-Views für Abfragen bereitzustellen. Der Name „Mango“ ist ein Wortspiel aus „MongoDB“ und „MapReduce“ – Mango orientiert sich stark an der Syntax von MongoDB, ist aber für JSON-Dokumente und die Bedürfnisse von CouchDB/PouchDB optimiert.

- **Vor Mango:** CouchDB setzte auf MapReduce-Views, die zwar mächtig, aber komplex und wenig intuitiv waren. Für einfache Filter und dynamische Queries waren sie oft überdimensioniert.
- **MongoDB-Einfluss:** Die populäre Abfragesprache von MongoDB inspirierte die Entwicklung einer JSON-basierten Query-Syntax, die leicht zu lesen und zu schreiben ist.
- **2015:** Die Mango Query API wurde als offizielles Feature in CouchDB eingeführt und kurz darauf von PouchDB übernommen. Ziel war es, flexible, dynamische und clientseitige Queries zu ermöglichen – besonders für Offline-First-Apps.
- **Heute:** Mango ist Standard für Queries in CouchDB und PouchDB und wird kontinuierlich weiterentwickelt. Sie ist ein Paradebeispiel für die Evolution von NoSQL-Abfragesprachen: weg von komplexen MapReduce-Logik, hin zu deklarativen, JSON-basierten Filtern, die auch für Web-Entwickler leicht verständlich sind.

Mango Query Language: Grundlagen

Konzept:

Das grundlegende Konzept von Mango-Queries ist denkbar einfach: Sie formulieren Ihre Suchkriterien als JSON-Objekt, das genau beschreibt, welche Dokumente Sie aus der Datenbank herausfiltern möchten. PouchDB übernimmt dann die Arbeit, durchsucht alle Dokumente und gibt Ihnen nur die passenden zurück. So wird die Abfrage nicht nur deklarativ und übersichtlich, sondern auch besonders flexibel – Sie können beliebige Felder und Bedingungen kombinieren, ohne eine eigene Abfragesprache lernen zu müssen.

> Mango-Queries sind JSON-Objekte, die Filterkriterien beschreiben. PouchDB durchsucht Dokumente und gibt nur passende zurück.

Basis-Syntax:

Der `selector` ist das Herzstück jeder Mango-Query. Hier geben Sie als JSON-Objekt an, welche Bedingungen ein Dokument erfüllen muss, damit es von der Datenbank gefunden und zurückgegeben wird. In der einfachsten Form prüfen Sie auf Gleichheit eines Feldes – zum Beispiel alle Produkte mit einer bestimmten Kategorie. Das macht Abfragen sehr intuitiv und sicher, da Sie keine komplexe Abfragesprache lernen müssen, sondern direkt mit Datenstrukturen arbeiten.

```
await db.find({
  selector: {
    field: value  // Einfache Gleichheit
  }
});
```

Beispiel:

Im folgenden greifen wir direkt auch die in indexedDB gespeicherte DB „lecture_demo“ zurück, falls es eine Fehlermeldung erscheint, gehen sie zurück zu Abschnitt Document Stores: Struktur wird zum Vorteil und führen sie das letzte Beispiel nochmal aus.

Das folgende Beispiel ist denkbar einfach, es werden Produkte aus verschiedenen Kategorien gefiltert. Versuchen Sie verschiedene Kategorien aus und experimentieren Sie mit Kombinationen von anderen Attributen wie `brand`, indem Sie die `selector`-Bedingungen anpassen.

```
1  const db = new PouchDB('lecture_demo');
2
3  // Finde alle Produkte in Kategorie "Electronics"
4  const result = await db.find({
5    selector: {
6      category: 'Electronics'
7    }
8  });
9
10 console.log(JSON.stringify(result.docs, null, 2));  // Array von pass
```

```
Dokumenten
```

```
[
  {
    "name": "Laptop 14\" 2.0",
    "category": "Electronics",
    "brand": "PixelPeak",
    "price": 1399.03,
    "stock": 138,
    "rating": 3.7,
    "created_at": "2025-06-14",
    "_id": "P00001",
    "_rev": "1-9f04c28afe20e1591157eab6468badeb"
  },
  {
    "name": "Portable SSD Max M",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 805.93,
    "stock": 211,
    "rating": 3.9,
    "created_at": "2024-05-21",
    "_id": "P00002",
    "_rev": "1-555949b498ecee3c9636a7864a1ed37b"
  },
  {
    "name": "4K Monitor - Green",
    "category": "Electronics",
    "brand": "Neutrino",
    "price": 522.48,
    "stock": 261,
    "rating": 5,
    "created_at": "2025-10-16",
    "_id": "P00003",
    "_rev": "1-2d1821be050e0dd6b60ebf520e9edec3"
  },
  {
    "name": "Tablet 10\"",
    "category": "Electronics",
    "brand": "Voltix",
    "price": 985.75,
    "stock": 54,
    "rating": 3.9,
    "created_at": "2025-04-16",
    "_id": "P00004",
```

```
    "_rev": "1-c6843bbcdfac1180e71dc3b662dab46c"
  },
  {
    "name": "Portable SSD Lite",
    "category": "Electronics",
    "brand": "OrbiTech",
    "price": 508.23,
    "stock": 20,
    "rating": 4.2,
    "created_at": "2025-03-12",
    "_id": "P00005",
    "_rev": "1-df3d70067532eb326fb84c8d6a4b4795"
  },
  {
    "name": "Soundbar - Red",
    "category": "Electronics",
    "brand": "OrbiTech",
    "price": 786.98,
    "stock": 66,
    "rating": 5,
    "created_at": "2025-09-07",
    "_id": "P00006",
    "_rev": "1-eb9be16ecf7d7ffdda04c4a18985bec1"
  },
  {
    "name": "Mechanical Keyboard - Black",
    "category": "Electronics",
    "brand": "Neutrino",
    "price": 537.59,
    "stock": 175,
    "rating": 4.4,
    "created_at": "2025-08-30",
    "_id": "P00007",
    "_rev": "1-61ec1d140622e541690233cc1490141f"
  },
  {
    "name": "Laptop 14\" Plus",
    "category": "Electronics",
    "brand": "Voltix",
    "price": 1187.46,
    "stock": 188,
    "rating": 4.6,
    "created_at": "2025-03-13",
```

```
    "_id": "P00008",
    "_rev": "1-17f15836c57e7a1bea1a65350643f6b0"
  },
  {
    "name": "Action Camera - Green",
    "category": "Electronics",
    "brand": "OrbiTech",
    "price": 502.12,
    "stock": 312,
    "rating": 5,
    "created_at": "2025-03-27",
    "_id": "P00009",
    "_rev": "1-86198d189c1a54c4dc9a6b162c2879b0"
  },
  {
    "name": "Soundbar",
    "category": "Electronics",
    "brand": "Auralite",
    "price": 1340.9,
    "stock": 171,
    "rating": 3.7,
    "created_at": "2025-02-25",
    "_id": "P00010",
    "_rev": "1-20c204a070f3727c39973e4f08548003"
  },
  {
    "name": "4K Monitor",
    "category": "Electronics",
    "brand": "Neutrino",
    "price": 24.92,
    "stock": 1,
    "rating": 3.3,
    "created_at": "2025-07-13",
    "_id": "P00011",
    "_rev": "1-5b8e2f5a4b3000038d95128d6683862e"
  },
  {
    "name": "Action Camera XL",
    "category": "Electronics",
    "brand": "Auralite",
    "price": 636.73,
    "stock": 124,
    "rating": 2.8,
```

```
    "created_at": "2025-04-14",
    "_id": "P00012",
    "_rev": "1-ce0570a88e0e2c70def7423f29530b4f"
  },
  {
    "name": "Gaming Mouse Plus M",
    "category": "Electronics",
    "brand": "Voltix",
    "price": 787.02,
    "stock": 127,
    "rating": 4.9,
    "created_at": "2024-08-18",
    "_id": "P00013",
    "_rev": "1-492a09e47772e7bce2da1027ea2389b0"
  },
  {
    "name": "Bluetooth Speaker Plus",
    "category": "Electronics",
    "brand": "Auralite",
    "price": 865.42,
    "stock": 69,
    "rating": 3.8,
    "created_at": "2025-02-23",
    "_id": "P00014",
    "_rev": "1-b8af9ee6d5b789dc92a7b6d60b4722d3"
  },
  {
    "name": "Noise-Canceling Earbuds Lite",
    "category": "Electronics",
    "brand": "Auralite",
    "price": 1339.39,
    "stock": 30,
    "rating": 4,
    "created_at": "2024-10-12",
    "_id": "P00015",
    "_rev": "1-cf519299d44a7e9353f3726a7c6cebf6"
  },
  {
    "name": "Action Camera - Blue",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 1322.9,
    "stock": 325,
```

```
    "rating": 4.8,
    "created_at": "2025-09-13",
    "_id": "P00016",
    "_rev": "1-864e301d319fdbcf80fc810caec9e7f3"
  },
  {
    "name": "Laptop 14\" Pro S",
    "category": "Electronics",
    "brand": "Auralite",
    "price": 481.53,
    "stock": 252,
    "rating": 3.5,
    "created_at": "2025-09-19",
    "_id": "P00017",
    "_rev": "1-f356f026028fe3ca97d14029c5417828"
  },
  {
    "name": "Gaming Mouse",
    "category": "Electronics",
    "brand": "Neutrino",
    "price": 1402.13,
    "stock": 15,
    "rating": 4.1,
    "created_at": "2025-06-28",
    "_id": "P00018",
    "_rev": "1-af83117f9f5fc91a71e9b8c154f8fb82"
  },
  {
    "name": "USB-C Hub - White",
    "category": "Electronics",
    "brand": "BlueWave",
    "price": 1093.57,
    "stock": 198,
    "rating": 4.6,
    "created_at": "2025-09-08",
    "_id": "P00019",
    "_rev": "1-3f579988745061cc7e37b631ac7cbead"
  },
  {
    "name": "USB-C Hub - Red",
    "category": "Electronics",
    "brand": "PixelPeak",
    "price": 317.33,
```

```
    "stock": 76,
    "rating": 4.3,
    "created_at": "2025-06-25",
    "_id": "P00020",
    "_rev": "1-3d0d828d9f1a3c77e40a62f05167e247"
  },
  {
    "name": "Wireless Headphones",
    "category": "Electronics",
    "brand": "BlueWave",
    "price": 377.2,
    "stock": 248,
    "rating": 4.3,
    "created_at": "2024-10-17",
    "_id": "P00021",
    "_rev": "1-b846b9c175fe914fdf61dba334943e6d"
  },
  {
    "name": "Laptop 14\" 2.0 - Black",
    "category": "Electronics",
    "brand": "Neutrino",
    "price": 833.21,
    "stock": 336,
    "rating": 4.4,
    "created_at": "2024-08-26",
    "_id": "P00022",
    "_rev": "1-d944003353ed16dcf8166228dec9b41e"
  },
  {
    "name": "Wireless Headphones M",
    "category": "Electronics",
    "brand": "BlueWave",
    "price": 697.05,
    "stock": 225,
    "rating": 5,
    "created_at": "2024-09-17",
    "_id": "P00023",
    "_rev": "1-dd439892547e9a222703037aef3f49d1"
  },
  {
    "name": "Tablet 10\" - Green",
    "category": "Electronics",
    "brand": "Auralite",
```

```
    "price": 667.7,
    "stock": 163,
    "rating": 4.2,
    "created_at": "2024-12-03",
    "_id": "P00024",
    "_rev": "1-ddad3dd805ccd126c1dc8299c4d9c13f"
  },
  {
    "name": "Soundbar Plus",
    "category": "Electronics",
    "brand": "PixelPeak",
    "price": 739.53,
    "stock": 60,
    "rating": 4.4,
    "created_at": "2025-08-28",
    "_id": "P00025",
    "_rev": "1-0bf0c0f9b9dd387404e3ac890f4a40c0"
  }
]
```


Diese Basis-Queries funktionieren – aber PouchDB führt einen Full-Table-Scan durch. Bei 1000 Dokumenten kein Problem, bei 100.000 katastrophal. Deshalb behandeln wir gleich Indizes. Aber zuerst: komplexere Queries.

Selektoren: Vergleichsoperatoren

Mango bietet reichhaltige Operatoren:

Operator	Bedeutung	Beispiel
`$eq`	Gleich	`{age: {$eq: 30}}`
`$ne`	Ungleich	`{status: {$ne: 'deleted'}}`
`$gt`	Größer als	`{price: {$gt: 100}}`
`$gte`	Größer oder gleich	`{age: {$gte: 18}}`
`$lt`	Kleiner als	`{stock: {$lt: 10}}`
`$lte`	Kleiner oder gleich	`{rating: {$lte: 3}}`
`$in`	Ist in Liste	`{category: {$in: ['A', 'B']}}`
`$nin`	Nicht in Liste	`{status: {$nin: ['draft', 'deleted']}}`

Beispiel: Produkte zwischen 100€ und 500€:

```
1  const db = new PouchDB('lecture_demo');
2
3  const products = await db.find({
4    selector: {
5      price: {
6        $gte: 100,
7        $lte: 500
8      }
9    }
10 });
11
12 console.log(JSON.stringify(products, null, 2))
```

```
{
  "docs": [
    {
      "name": "Laptop 14\" Pro S",
      "category": "Electronics",
      "brand": "Auralite",
      "price": 481.53,
      "stock": 252,
      "rating": 3.5,
      "created_at": "2025-09-19",
      "_id": "P00017",
      "_rev": "1-f356f026028fe3ca97d14029c5417828"
    },
    {
      "name": "USB-C Hub - Red",
      "category": "Electronics",
      "brand": "PixelPeak",
      "price": 317.33,
      "stock": 76,
      "rating": 4.3,
      "created_at": "2025-06-25",
      "_id": "P00020",
      "_rev": "1-3d0d828d9f1a3c77e40a62f05167e247"
    },
    {
      "name": "Wireless Headphones",
      "category": "Electronics",
      "brand": "BlueWave",
      "price": 377.2,
      "stock": 248,
      "rating": 4.3,
      "created_at": "2024-10-17",
      "_id": "P00021",
      "_rev": "1-b846b9c175fe914fdf61dba334943e6d"
    },
    {
      "name": "Tablet 10\" - Black",
      "category": "Electronics",
      "brand": "Neutrino",
      "price": 128.34,
      "stock": 35,
      "rating": 3.7,
      "created_at": "2025-06-16",
```

```
    "_id": "P00028",
    "_rev": "1-6ac1209c1dd621f76c8b108e59ecbc04"
  },
  {
    "name": "Gaming Mouse Lite S",
    "category": "Electronics",
    "brand": "Auralite",
    "price": 486.97,
    "stock": 130,
    "rating": 4.9,
    "created_at": "2025-08-18",
    "_id": "P00045",
    "_rev": "1-cef376a1ea29af452abf221057235594"
  },
  {
    "name": "4K Monitor",
    "category": "Electronics",
    "brand": "OrbiTech",
    "price": 226.26,
    "stock": 18,
    "rating": 4.5,
    "created_at": "2024-08-15",
    "_id": "P00047",
    "_rev": "1-9f665d0ad699a08899156dce14ed0377"
  },
  {
    "name": "Bluetooth Speaker XL",
    "category": "Electronics",
    "brand": "OrbiTech",
    "price": 215.56,
    "stock": 184,
    "rating": 3.8,
    "created_at": "2025-04-02",
    "_id": "P00049",
    "_rev": "1-b894a6bdacd63321add40dc5fa104dbb"
  },
  {
    "name": "Gaming Mouse Pro S",
    "category": "Electronics",
    "brand": "BlueWave",
    "price": 270.97,
    "stock": 67,
    "rating": 4.5,
```

```
    "created_at": "2024-09-21",
    "_id": "P00050",
    "_rev": "1-986fadef2c76dfe5b550020ee5e6c85d"
  },
  {
    "name": "Smartwatch Pro L - Graphite",
    "category": "Electronics",
    "brand": "Voltix",
    "price": 394.22,
    "stock": 130,
    "rating": 4,
    "created_at": "2024-10-07",
    "_id": "P00051",
    "_rev": "1-58e4653a81b393cba8ff8962094e2b06"
  },
  {
    "name": "Portable SSD Plus",
    "category": "Electronics",
    "brand": "OrbiTech",
    "price": 271.37,
    "stock": 256,
    "rating": 4.1,
    "created_at": "2025-04-13",
    "_id": "P00052",
    "_rev": "1-ee3ec16d1f234e368f5035a83bcd588f"
  },
  {
    "name": "Laptop 14\" 2.0 - White",
    "category": "Electronics",
    "brand": "Voltix",
    "price": 132.35,
    "stock": 103,
    "rating": 4.1,
    "created_at": "2025-09-23",
    "_id": "P00062",
    "_rev": "1-a64c20dc5ab6349f9c407f2e77d5b8a5"
  },
  {
    "name": "Portable SSD",
    "category": "Electronics",
    "brand": "Voltix",
    "price": 289.75,
    "stock": 137,
```

```
      "rating": 3.7,
      "created_at": "2025-04-28",
      "_id": "P00063",
      "_rev": "1-df75a25d096c75163d1e8039bef9f2c8"
    },
    {
      "name": "USB-C Hub S",
      "category": "Electronics",
      "brand": "PixelPeak",
      "price": 298.11,
      "stock": 228,
      "rating": 4.2,
      "created_at": "2024-08-15",
      "_id": "P00064",
      "_rev": "1-3fe7cc927001d11d819fe4ad4ac13b81"
    },
    {
      "name": "Soundbar Plus",
      "category": "Electronics",
      "brand": "ZenCore",
      "price": 146.53,
      "stock": 79,
      "rating": 4.3,
      "created_at": "2024-07-27",
      "_id": "P00066",
      "_rev": "1-e53f7e7adf95443056db0c91ca1a6c0b"
    },
    {
      "name": "Soundbar 2.0",
      "category": "Electronics",
      "brand": "ZenCore",
      "price": 231.87,
      "stock": 299,
      "rating": 3,
      "created_at": "2025-10-19",
      "_id": "P00072",
      "_rev": "1-87326b4d2b4a309c812133b2c635bd3b"
    },
    {
      "name": "Soundbar Max M",
      "category": "Electronics",
      "brand": "Neutrino",
      "price": 317.14,
```

```
    "stock": 169,
    "rating": 3.8,
    "created_at": "2025-02-11",
    "_id": "P00073",
    "_rev": "1-d47ea37d48d89fe290bd8b86e89a752c"
  },
  {
    "name": "Tablet 10\" S - Black",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 192.98,
    "stock": 328,
    "rating": 4.4,
    "created_at": "2025-06-14",
    "_id": "P00077",
    "_rev": "1-c141f6726f7b81cd31e4bb93579c3984"
  },
  {
    "name": "Gaming Mouse",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 317.38,
    "stock": 154,
    "rating": 3.6,
    "created_at": "2025-06-01",
    "_id": "P00082",
    "_rev": "1-b731671bb51440101ef9d8b743acd0ba"
  },
  {
    "name": "Gaming Mouse Pro",
    "category": "Electronics",
    "brand": "Auralite",
    "price": 415.47,
    "stock": 320,
    "rating": 4.8,
    "created_at": "2024-05-12",
    "_id": "P00091",
    "_rev": "1-cbce3892a774b2fe1c9cbc374a44d9fa"
  },
  {
    "name": "Action Camera - White",
    "category": "Electronics",
    "brand": "ZenCore",
```

```
    "price": 129.17,
    "stock": 48,
    "rating": 5,
    "created_at": "2025-07-25",
    "_id": "P00097",
    "_rev": "1-6774a40650a7703452a1f9d4de4af402"
  },
  {
    "name": "USB-C Hub Plus",
    "category": "Electronics",
    "brand": "Voltix",
    "price": 408.92,
    "stock": 240,
    "rating": 3.4,
    "created_at": "2024-08-13",
    "_id": "P00099",
    "_rev": "1-a9ced38bf10554e77188bac066fe684f"
  },
  {
    "name": "Leggings Max - Silver",
    "category": "Clothing",
    "brand": "AeroLoom",
    "price": 245.35,
    "stock": 92,
    "rating": 4.7,
    "created_at": "2025-06-24",
    "_id": "P00101",
    "_rev": "1-3dab46ba81e4471cb94976e1f394f8db"
  },
  {
    "name": "Jacket Max XL",
    "category": "Clothing",
    "brand": "Cotton&Co",
    "price": 110.77,
    "stock": 464,
    "rating": 4.1,
    "created_at": "2025-01-25",
    "_id": "P00102",
    "_rev": "1-2b488c1559b8f90d0b6da73fb1f0e22d"
  },
  {
    "name": "Hoodie - Black",
    "category": "Clothing",
```

```
      "brand": "UrbanTrail",
      "price": 106.25,
      "stock": 146,
      "rating": 5,
      "created_at": "2024-10-22",
      "_id": "P00104",
      "_rev": "1-f06d20a0682822baf64e09520cc7b85e"
    },
    {
      "name": "Hoodie",
      "category": "Clothing",
      "brand": "Nordline",
      "price": 184.13,
      "stock": 548,
      "rating": 4.2,
      "created_at": "2024-12-19",
      "_id": "P00106",
      "_rev": "1-117174466a94b7b57140813b0c878cf2"
    }
  ],
  "warning": "No matching index found, create an index to optimize
query time."
}
```


Diese Operatoren sollten SQL-Nutzern vertraut vorkommen – aber in JSON-Notation. Der Vorteil: Queries sind selbst Daten und können programmatisch generiert werden. Kein String-Concatenation wie bei SQL-Injection-Risiken.

Logische Operatoren: AND, OR, NOT

Ein besonderer Vorteil der Mango-Query-Operatoren wie `$and`, `$or`, `$not` und `$nor` ist ihre Flexibilität: Sie können nicht nur zwei, sondern beliebig viele Bedingungen kombinieren. Anders als in klassischen Programmiersprachen, wo `&&` und `||` meist nur zwei Operanden verknüpfen, akzeptieren Mango-Operatoren Arrays mit beliebig vielen Elementen. So können Sie zum Beispiel mit `$and` eine ganze Liste von Bedingungen formulieren, die alle erfüllt sein müssen, oder mit `$or` eine Auswahl von Alternativen angeben. Das macht komplexe Filter sehr übersichtlich und vermeidet verschachtelte Ausdrücke.

Operator	Bedeutung	Beispiel
`$and`	Alle Bedingungen müssen wahr sein	`{$and: [{a: 1}, {b: 2}, {c: 3}]}`
`$or`	Mindestens eine Bedingung wahr	`{$or: [{a: 1}, {b: 2}, {c: 3}]}`
`$not`	Negation	`{age: {$not: {$lt: 18}}}`
`$nor`	Keine Bedingung wahr	`{$nor: [{a: 1}, {b: 2}, {c: 3}]}`

Beispiel: Produkte in „Electronics“ ODER „Books“ mit Preis > 20€:

Beachten Sie die Verschachtelung: AND auf oberster Ebene, OR darunter. Diese Struktur reflektiert die logische Priorität. In SQL wäre das: WHERE (category = ‚Electronics‘ OR category = ‚Books‘) AND price > 20. Mango ist expliziter, aber auch verboseiser.

```
const db = new PouchDB('lecture_demo');

const products = await db.find({
  selector: {
    $and: [
      {
        $or: [
          { category: 'Electronics' },
          { category: 'Books' }
        ]
      },
      {
        price: { $gt: 20 }
      }
    ]
  }
});

console.log(JSON.stringify(products, null, 2))
```

```
{
  "docs": [
    {
      "name": "Laptop 14\" 2.0",
      "category": "Electronics",
      "brand": "PixelPeak",
      "price": 1399.03,
      "stock": 138,
      "rating": 3.7,
      "created_at": "2025-06-14",
      "_id": "P00001",
      "_rev": "1-9f04c28afe20e1591157eab6468badeb"
    },
    {
      "name": "Portable SSD Max M",
      "category": "Electronics",
      "brand": "ZenCore",
      "price": 805.93,
      "stock": 211,
      "rating": 3.9,
      "created_at": "2024-05-21",
      "_id": "P00002",
      "_rev": "1-555949b498ecee3c9636a7864a1ed37b"
    },
    {
      "name": "4K Monitor - Green",
      "category": "Electronics",
      "brand": "Neutrino",
      "price": 522.48,
      "stock": 261,
      "rating": 5,
      "created_at": "2025-10-16",
      "_id": "P00003",
      "_rev": "1-2d1821be050e0dd6b60ebf520e9edec3"
    },
    {
      "name": "Tablet 10\"",
      "category": "Electronics",
      "brand": "Voltix",
      "price": 985.75,
      "stock": 54,
      "rating": 3.9,
      "created_at": "2025-04-16",
```

```
    "_id": "P00004",
    "_rev": "1-c6843bbcdfac1180e71dc3b662dab46c"
  },
  {
    "name": "Portable SSD Lite",
    "category": "Electronics",
    "brand": "OrbiTech",
    "price": 508.23,
    "stock": 20,
    "rating": 4.2,
    "created_at": "2025-03-12",
    "_id": "P00005",
    "_rev": "1-df3d70067532eb326fb84c8d6a4b4795"
  },
  {
    "name": "Soundbar - Red",
    "category": "Electronics",
    "brand": "OrbiTech",
    "price": 786.98,
    "stock": 66,
    "rating": 5,
    "created_at": "2025-09-07",
    "_id": "P00006",
    "_rev": "1-eb9be16ecf7d7ffdda04c4a18985bec1"
  },
  {
    "name": "Mechanical Keyboard - Black",
    "category": "Electronics",
    "brand": "Neutrino",
    "price": 537.59,
    "stock": 175,
    "rating": 4.4,
    "created_at": "2025-08-30",
    "_id": "P00007",
    "_rev": "1-61ec1d140622e541690233cc1490141f"
  },
  {
    "name": "Laptop 14\" Plus",
    "category": "Electronics",
    "brand": "Voltix",
    "price": 1187.46,
    "stock": 188,
    "rating": 4.6,
```

```
    "created_at": "2025-03-13",
    "_id": "P00008",
    "_rev": "1-17f15836c57e7a1bea1a65350643f6b0"
  },
  {
    "name": "Action Camera - Green",
    "category": "Electronics",
    "brand": "OrbiTech",
    "price": 502.12,
    "stock": 312,
    "rating": 5,
    "created_at": "2025-03-27",
    "_id": "P00009",
    "_rev": "1-86198d189c1a54c4dc9a6b162c2879b0"
  },
  {
    "name": "Soundbar",
    "category": "Electronics",
    "brand": "Auralite",
    "price": 1340.9,
    "stock": 171,
    "rating": 3.7,
    "created_at": "2025-02-25",
    "_id": "P00010",
    "_rev": "1-20c204a070f3727c39973e4f08548003"
  },
  {
    "name": "4K Monitor",
    "category": "Electronics",
    "brand": "Neutrino",
    "price": 24.92,
    "stock": 1,
    "rating": 3.3,
    "created_at": "2025-07-13",
    "_id": "P00011",
    "_rev": "1-5b8e2f5a4b3000038d95128d6683862e"
  },
  {
    "name": "Action Camera XL",
    "category": "Electronics",
    "brand": "Auralite",
    "price": 636.73,
    "stock": 124,
```

```
    "rating": 2.8,
    "created_at": "2025-04-14",
    "_id": "P00012",
    "_rev": "1-ce0570a88e0e2c70def7423f29530b4f"
  },
  {
    "name": "Gaming Mouse Plus M",
    "category": "Electronics",
    "brand": "Voltix",
    "price": 787.02,
    "stock": 127,
    "rating": 4.9,
    "created_at": "2024-08-18",
    "_id": "P00013",
    "_rev": "1-492a09e47772e7bce2da1027ea2389b0"
  },
  {
    "name": "Bluetooth Speaker Plus",
    "category": "Electronics",
    "brand": "Auralite",
    "price": 865.42,
    "stock": 69,
    "rating": 3.8,
    "created_at": "2025-02-23",
    "_id": "P00014",
    "_rev": "1-b8af9ee6d5b789dc92a7b6d60b4722d3"
  },
  {
    "name": "Noise-Canceling Earbuds Lite",
    "category": "Electronics",
    "brand": "Auralite",
    "price": 1339.39,
    "stock": 30,
    "rating": 4,
    "created_at": "2024-10-12",
    "_id": "P00015",
    "_rev": "1-cf519299d44a7e9353f3726a7c6cebf6"
  },
  {
    "name": "Action Camera - Blue",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 1322.9,
```

```
    "stock": 325,
    "rating": 4.8,
    "created_at": "2025-09-13",
    "_id": "P00016",
    "_rev": "1-864e301d319fdbcf80fc810caec9e7f3"
  },
  {
    "name": "Laptop 14\" Pro S",
    "category": "Electronics",
    "brand": "Auralite",
    "price": 481.53,
    "stock": 252,
    "rating": 3.5,
    "created_at": "2025-09-19",
    "_id": "P00017",
    "_rev": "1-f356f026028fe3ca97d14029c5417828"
  },
  {
    "name": "Gaming Mouse",
    "category": "Electronics",
    "brand": "Neutrino",
    "price": 1402.13,
    "stock": 15,
    "rating": 4.1,
    "created_at": "2025-06-28",
    "_id": "P00018",
    "_rev": "1-af83117f9f5fc91a71e9b8c154f8fb82"
  },
  {
    "name": "USB-C Hub - White",
    "category": "Electronics",
    "brand": "BlueWave",
    "price": 1093.57,
    "stock": 198,
    "rating": 4.6,
    "created_at": "2025-09-08",
    "_id": "P00019",
    "_rev": "1-3f579988745061cc7e37b631ac7cbead"
  },
  {
    "name": "USB-C Hub - Red",
    "category": "Electronics",
    "brand": "PixelPeak",
```

```
    "price": 317.33,
    "stock": 76,
    "rating": 4.3,
    "created_at": "2025-06-25",
    "_id": "P00020",
    "_rev": "1-3d0d828d9f1a3c77e40a62f05167e247"
  },
  {
    "name": "Wireless Headphones",
    "category": "Electronics",
    "brand": "BlueWave",
    "price": 377.2,
    "stock": 248,
    "rating": 4.3,
    "created_at": "2024-10-17",
    "_id": "P00021",
    "_rev": "1-b846b9c175fe914fdf61dba334943e6d"
  },
  {
    "name": "Laptop 14\" 2.0 - Black",
    "category": "Electronics",
    "brand": "Neutrino",
    "price": 833.21,
    "stock": 336,
    "rating": 4.4,
    "created_at": "2024-08-26",
    "_id": "P00022",
    "_rev": "1-d944003353ed16dcf8166228dec9b41e"
  },
  {
    "name": "Wireless Headphones M",
    "category": "Electronics",
    "brand": "BlueWave",
    "price": 697.05,
    "stock": 225,
    "rating": 5,
    "created_at": "2024-09-17",
    "_id": "P00023",
    "_rev": "1-dd439892547e9a222703037aef3f49d1"
  },
  {
    "name": "Tablet 10\" - Green",
    "category": "Electronics",
```

```
      "brand": "Auralite",
      "price": 667.7,
      "stock": 163,
      "rating": 4.2,
      "created_at": "2024-12-03",
      "_id": "P00024",
      "_rev": "1-ddad3dd805ccd126c1dc8299c4d9c13f"
    },
    {
      "name": "Soundbar Plus",
      "category": "Electronics",
      "brand": "PixelPeak",
      "price": 739.53,
      "stock": 60,
      "rating": 4.4,
      "created_at": "2025-08-28",
      "_id": "P00025",
      "_rev": "1-0bf0c0f9b9dd387404e3ac890f4a40c0"
    }
  ],
  "warning": "No matching index found, create an index to optimize
query time."
}
```


Verschachtelte Felder & Arrays

Dot-Notation für verschachtelte Objekte:

Die Dot-Notation ist ein zentrales Werkzeug, um in Mango-Queries auf verschachtelte Felder innerhalb von Dokumenten zuzugreifen. Statt komplexer Verschachtelungen können Sie einfach mit einem Punkt getrennte Feldnamen angeben, zum Beispiel ‚address.city'. So lassen sich auch tieferliegende Werte gezielt abfragen, ohne das gesamte Objekt durchsuchen zu müssen. Das macht die Arbeit mit komplexen JSON-Strukturen besonders effizient und übersichtlich.

```
1  const db = new PouchDB('user-db', {adapter: "memory"});
2
3  const userDB = [{
4    _id: 'user_001',
5    name: 'Alice',
6    address: {
7      city: 'Berlin',
8      zip: '10115'
9    }
10 }, {
11   _id: 'user_002',
12   name: 'Bob',
13   address: {
```

```
    city: 'Freiberg',
    zip: '09599'
  }
}]

await db.bulkDocs(userDB)

// Query:
const user = await db.find({
  selector: {
    'address.city': 'Berlin'  // Achtung: String mit Punkt!
  }
});

console.log(JSON.stringify(user, null, 2))
```


```
{
  "docs": [
    {
      "name": "Alice",
      "address": {
        "city": "Berlin",
        "zip": "10115"
      },
      "_id": "user_001",
      "_rev": "1-feb4924d38a65988c3f9e7fd33ff5c4b"
    }
  ],
  "warning": "No matching index found, create an index to optimize
query time."
}
```


Array-Operatoren:

Array-Operatoren sind besonders wichtig, wenn Sie mit Listen oder mehreren Werten in Ihren Dokumenten arbeiten. Mit `$elemMatch` können Sie gezielt prüfen, ob mindestens ein Element im Array eine bestimmte Bedingung erfüllt. `$size` erlaubt es, nach Arrays mit einer bestimmten Länge zu suchen, und `$all` prüft, ob alle angegebenen Werte im Array enthalten sind.

Operator	Bedeutung	Beispiel
`$elemMatch`	Mindestens ein Array-Element erfüllt Bedingung	`{tags: {$elemMatch: {$eq: 'urgent'}}}`
`$size`	Array hat bestimmte Länge	`{tags: {$size: 3}}`
`$all`	Array enthält alle Werte	`{tags: {$all: ['red', 'blue']}}`

Beispiel: Produkte mit Tag „computer“:

Arrays können tricky sein: PouchDB prüft automatisch, ob der Wert im Array enthalten ist. `tags: 'computer'` findet Dokumente mit `tags: ['computer', 'laptop']`. Für komplexere Bedingungen – etwa „Array enthält Objekt mit property X“ – brauchen Sie `$elemMatch`.

```
await db.find({
  selector: {
    tags: 'computer'  // Vereinfachte Schreibweise
  }
});

// Oder explizit:
await db.find({
  selector: {
    tags: { $elemMatch: { $eq: 'computer' } }
  }
});
```

String-Operationen

String-Operationen in Mango-Queries

String-Operatoren erlauben es, gezielt nach Textmustern, Teilstrings oder regulären Ausdrücken in Ihren Dokumenten zu suchen. Sie sind besonders nützlich, wenn Sie Felder wie Namen, Beschreibungen oder Tags filtern möchten. Mit Operatoren wie `$regex`, `$exists` und `$type` können Sie flexible und leistungsfähige Textabfragen formulieren, die weit über einfache Gleichheit hinausgehen.

Operator	Bedeutung	Beispiel
`$regex`	Regulärer Ausdruck	`{name: {$regex: '^A.*'}}`
`$exists`	Feld existiert oder nicht	`{description: {$exists: true}}`
`$type`	Feldtyp prüfen	`{name: {$type: 'string'}}`

Beispiel: Produkte mit Namen, die mit „A“ beginnen

Mit dem `$regex`-Operator können Sie nach Mustern im Text suchen. Im folgenden Beispiel werden alle Produkte gefunden, deren Name mit dem Buchstaben „A“ beginnt. Sie können den regulären Ausdruck beliebig anpassen, um andere Muster zu finden.

```
1  const db = new PouchDB('lecture_demo');
2
3  const products = await db.find({
4    selector: {
5      name: { $regex: '^A.*' }
6    }
7  });
8
9  console.log(JSON.stringify(products.docs, null, 2))
```

```
[
  {
    "name": "Action Camera - Green",
    "category": "Electronics",
    "brand": "OrbiTech",
    "price": 502.12,
    "stock": 312,
    "rating": 5,
    "created_at": "2025-03-27",
    "_id": "P00009",
    "_rev": "1-86198d189c1a54c4dc9a6b162c2879b0"
  },
  {
    "name": "Action Camera XL",
    "category": "Electronics",
    "brand": "Auralite",
    "price": 636.73,
    "stock": 124,
    "rating": 2.8,
    "created_at": "2025-04-14",
    "_id": "P00012",
    "_rev": "1-ce0570a88e0e2c70def7423f29530b4f"
  },
  {
    "name": "Action Camera - Blue",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 1322.9,
    "stock": 325,
    "rating": 4.8,
    "created_at": "2025-09-13",
    "_id": "P00016",
    "_rev": "1-864e301d319fdbcf80fc810caec9e7f3"
  },
  {
    "name": "Action Camera",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 1027.7,
    "stock": 122,
    "rating": 3.3,
    "created_at": "2024-09-05",
    "_id": "P00030",
```

```
    "_rev": "1-b5659b14728bbf6c5acfd8da6946ed0d"
  },
  {
    "name": "Action Camera Plus",
    "category": "Electronics",
    "brand": "Auralite",
    "price": 1376.9,
    "stock": 81,
    "rating": 4.2,
    "created_at": "2025-07-11",
    "_id": "P00037",
    "_rev": "1-862a6f7dfe35a2bdf81b80d748bd6ab9"
  },
  {
    "name": "Action Camera 2.0",
    "category": "Electronics",
    "brand": "Voltix",
    "price": 885.27,
    "stock": 173,
    "rating": 3.9,
    "created_at": "2024-08-24",
    "_id": "P00059",
    "_rev": "1-a34b9bf60b814c2eb43c0af09a7a977c"
  },
  {
    "name": "Action Camera",
    "category": "Electronics",
    "brand": "OrbiTech",
    "price": 1259.3,
    "stock": 115,
    "rating": 4,
    "created_at": "2024-11-01",
    "_id": "P00061",
    "_rev": "1-b87e3b0fe6798cb97f08eace5ec3708a"
  },
  {
    "name": "Action Camera - White",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 129.17,
    "stock": 48,
    "rating": 5,
    "created_at": "2025-07-25",
```

```
    "_id": "P00097",
    "_rev": "1-6774a40650a7703452a1f9d4de4af402"
  },
  {
    "name": "Arabica Coffee Beans 1kg",
    "category": "Groceries",
    "brand": "FreshField",
    "price": 0.9,
    "stock": 1803,
    "rating": 5,
    "created_at": "2025-10-11",
    "_id": "P00207",
    "_rev": "1-e167ff850147ce40dcd82ed497acc5c8"
  },
  {
    "name": "Arabica Coffee Beans 1kg Plus",
    "category": "Groceries",
    "brand": "FreshField",
    "price": 19.88,
    "stock": 1804,
    "rating": 3.8,
    "created_at": "2025-01-09",
    "_id": "P00211",
    "_rev": "1-90904176b3c33cc45f063a24b6674e19"
  },
  {
    "name": "Arabica Coffee Beans 1kg",
    "category": "Groceries",
    "brand": "SunHarvest",
    "price": 37.91,
    "stock": 1172,
    "rating": 4.1,
    "created_at": "2025-07-25",
    "_id": "P00214",
    "_rev": "1-7b30e4a74ff969429b7e6aaeab7d7cc6"
  },
  {
    "name": "Arabica Coffee Beans 1kg Pro XL - Blue",
    "category": "Groceries",
    "brand": "DailyChoice",
    "price": 33.57,
    "stock": 596,
    "rating": 4.7,
```

```
    "created_at": "2025-03-26",
    "_id": "P00236",
    "_rev": "1-80789be3396510967dd4570efcf7e1fc"
  },
  {
    "name": "Arabica Coffee Beans 1kg Lite",
    "category": "Groceries",
    "brand": "Pure&Simple",
    "price": 21.66,
    "stock": 918,
    "rating": 3.2,
    "created_at": "2025-04-01",
    "_id": "P00239",
    "_rev": "1-944caf4ee12b7b4f9b0f6d749611ff6b"
  },
  {
    "name": "Almonds 200g Lite XL",
    "category": "Groceries",
    "brand": "Everfarm",
    "price": 2.48,
    "stock": 651,
    "rating": 3.5,
    "created_at": "2024-09-10",
    "_id": "P00242",
    "_rev": "1-576a167b51440b5b291542311a457136"
  },
  {
    "name": "Almonds 200g - Green",
    "category": "Groceries",
    "brand": "FreshField",
    "price": 31.09,
    "stock": 1468,
    "rating": 4.6,
    "created_at": "2025-06-12",
    "_id": "P00255",
    "_rev": "1-a4619665f118a05786f17062b3d7abc7"
  },
  {
    "name": "Arabica Coffee Beans 1kg - Green",
    "category": "Groceries",
    "brand": "FreshField",
    "price": 18.68,
    "stock": 832,
```

```
    "rating": 4,
    "created_at": "2024-06-16",
    "_id": "P00260",
    "_rev": "1-58a8f1956caaa91130cb422d77a09de5"
  },
  {
    "name": "Almonds 200g",
    "category": "Groceries",
    "brand": "DailyChoice",
    "price": 3.59,
    "stock": 300,
    "rating": 4.1,
    "created_at": "2025-04-08",
    "_id": "P00264",
    "_rev": "1-a1c38efd5d46e9af92fab8a144731e62"
  },
  {
    "name": "Almonds 200g Max",
    "category": "Groceries",
    "brand": "Everfarm",
    "price": 26.28,
    "stock": 1890,
    "rating": 3.1,
    "created_at": "2025-10-08",
    "_id": "P00271",
    "_rev": "1-9c038c0ec56a40565450ae1077d15975"
  },
  {
    "name": "Almonds 200g - Graphite",
    "category": "Groceries",
    "brand": "DailyChoice",
    "price": 28.17,
    "stock": 1159,
    "rating": 3.5,
    "created_at": "2025-02-14",
    "_id": "P00274",
    "_rev": "1-428a2a355934605277b51583f86b0d86"
  },
  {
    "name": "Almonds 200g Max - White",
    "category": "Groceries",
    "brand": "Everfarm",
    "price": 7.2,
```

```
    "stock": 388,
    "rating": 3.7,
    "created_at": "2025-10-06",
    "_id": "P00277",
    "_rev": "1-d7bf00037c27a0f9acc82e3b4efa412d"
  },
  {
    "name": "Arabica Coffee Beans 1kg",
    "category": "Groceries",
    "brand": "SunHarvest",
    "price": 3.09,
    "stock": 968,
    "rating": 3.6,
    "created_at": "2024-08-21",
    "_id": "P00280",
    "_rev": "1-52119dec73f9d245ca60c2dcc2a871e3"
  },
  {
    "name": "Arabica Coffee Beans 1kg",
    "category": "Groceries",
    "brand": "SunHarvest",
    "price": 36.58,
    "stock": 994,
    "rating": 4.6,
    "created_at": "2024-11-05",
    "_id": "P00282",
    "_rev": "1-6d1dc842bb12358707c1c45988aeb3cd"
  },
  {
    "name": "Arabica Coffee Beans 1kg Plus - Blue",
    "category": "Groceries",
    "brand": "DailyChoice",
    "price": 17.32,
    "stock": 1462,
    "rating": 3.2,
    "created_at": "2025-05-27",
    "_id": "P00297",
    "_rev": "1-f6dcc70fbdd38dc3b411150551d77754"
  },
  {
    "name": "Arabica Coffee Beans 1kg Max XL - Silver",
    "category": "Groceries",
    "brand": "DailyChoice",
```

```
    "price": 22.38,
    "stock": 1517,
    "rating": 4.6,
    "created_at": "2024-07-13",
    "_id": "P00299",
    "_rev": "1-f22e317fd889d7cea82a506da4bc452d"
  },
  {
    "name": "A4 Copy Paper 500 sheets - Silver",
    "category": "Office",
    "brand": "ErgoCraft",
    "price": 417.1,
    "stock": 725,
    "rating": 4.5,
    "created_at": "2024-10-29",
    "_id": "P00306",
    "_rev": "1-be0439a10bd48ca53eb0e0d167fd56d2"
  }
]
```


Sortierung, Limitierung & Pagination

Sortierung, Limitierung und Pagination sind essenzielle Werkzeuge, um große Datenmengen effizient zu durchsuchen und übersichtlich darzustellen. Mit diesen Operatoren können Sie die Reihenfolge der Ergebnisse steuern, die Anzahl der zurückgegebenen Dokumente begrenzen und gezielt Teilmengen abfragen. Besonders bei Listen, Suchergebnissen oder Reports sind diese Funktionen unverzichtbar.

Operator	Bedeutung	Beispiel
`sort`	Ergebnisse sortieren	`sort: [{ price: 'asc' }]`
`limit`	Maximale Anzahl Ergebnisse	`limit: 10`
`skip`	Ergebnisse überspringen (Pagination)	`skip: 20`
`fields`	Nur bestimmte Felder laden	`fields: ['name', 'price']`

Beispiel: Top 5 teuerste Produkte

Im folgenden Beispiel werden die fünf teuersten Produkte aus der Datenbank abgefragt. Die Ergebnisse werden nach dem Preis absteigend sortiert, und nur die obersten fünf Dokumente werden zurückgegeben. Sie können die Operatoren beliebig kombinieren, um die Ausgabe weiter zu steuern.

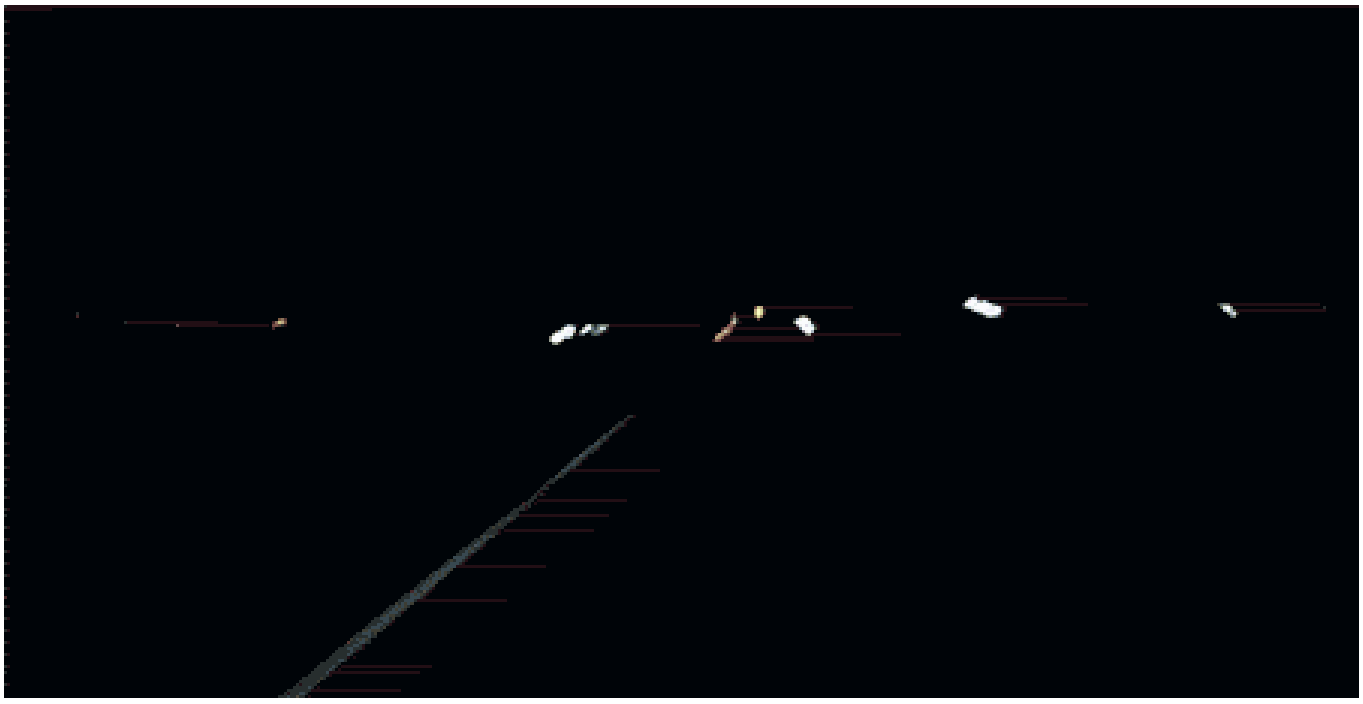
```
const db = new PouchDB('lecture_demo');

const result = await db.find({
  selector: {
    price: { $exists: true }
  },
  sort: [{ price: 'desc' }],
  limit: 5
});

console.log(JSON.stringify(result.docs, null, 2))
```

```
error Cannot sort on field(s) "price" when using the default index
```

Ups, wenn hier eine Fehlermeldung erscheint, dann liegt das daran, dass noch kein Index für das Feld `price` existiert. Aus Performance-Gründen muss PouchDB Indizes für sortierte Abfragen anlegen. Wir werden das im nächsten Abschnitt behandeln!

Block 4: Index-Strategien für Performance

Bisher konnten wir alle Abfragen direkt an PouchDB weitergeben und die mächtigen Query-Operatoren nutzen. Allerdings ist die Performance noch nicht optimal: Jede Suche, egal wie komplex, führt intern zu einem vollständigen Scan aller Dokumente. Das ist bei kleinen Datenmengen kein Problem, wird aber mit wachsender Datenbank schnell zum Flaschenhals. Im folgenden Abschnitt lernen Sie, wie Indizes die Suche dramatisch beschleunigen und aus dem Full-Scan einen gezielten Lookup machen.



Das Index-Konzept: Von $O(n)$O(n) zu $O(\log n)$O(log n)

Was ist ein Index?

Ein Index (Sekundärindex) ist eine spezielle Datenstruktur, die es ermöglicht, Daten schnell zu finden, ohne alle Einträge durchsuchen zu müssen. Indizes werden auf bestimmten Feldern erstellt und erlauben es, Abfragen auf diesen Feldern effizient auszuführen. Statt jedes Dokument zu prüfen, kann das System den Index nutzen, um direkt zu den relevanten Dokumenten zu springen. Wie man am Beispiel sieht, erstellen wir Indizes auf den Feldern `brand`, `category` und `stock`, um schnelle Lookups zu ermöglichen. Und weiterhin sieht man das die Nutzung von Indizes die Suchzeit drastisch reduziert aber auch den Code etwas komplexer macht.

Index.js

```
const Indexes = {
  product: new Map(),
  brand: new Map(),
  category: new Map(),
  stock: new Map(),
  price: new Map(),
  rating: new Map(),
  created_at: new Map()
};


window.shopData.forEach((row, index) => {
  Indexes.product.set(row._id, index);
  Indexes.brand.set(row.brand, (Indexes.brand.get(row.brand) || []).c
    (index));
  Indexes.category.set(row.category, (Indexes.category.get(row.catego
    []).concat(index));
  Indexes.stock.set(row.stock, (Indexes.stock.get(row.stock) || []).c
    (index));
});

// console.log("Produktindex:", [...Indexes.product]);
console.log("Markenindex:", [...Indexes.brand]);
console.log("Kategorieindex:", [...Indexes.category]);
console.log("Bestandsindex:", [...Indexes.stock]);

```

Search.js

```
function filterIndexBy(indexName, keyFilter) {
  const start = performance.now();
  const index = Indexes[indexName];

  if (!index) {
    console.log(`Index "${indexName}" nicht gefunden`);
    return;
  }

  const results = [];

  // Über alle Keys im Index iterieren
  for (const [key, rowIndexes] of index) {
    // Filter auf den Key anwenden
    sleep(1) // Simuliere Latenz
    if (keyFilter(key)) {
      // Bei Single-Value Index (z.B. product)
      if (typeof rowIndexes === 'number') {
        results.push(window.shopData[rowIndexes]);
      }
      // Bei Multi-Value Index (z.B. brand, category)
      else if (Array.isArray(rowIndexes)) {
        rowIndexes.forEach(i => results.push(window.shopData[i]));
      }
    }
  }

  const end = performance.now();

  if (results.length > 0) {
    console.log(`Gefunden: ${results.length} Produkte`);
    console.log(JSON.stringify(results, null, 2));
  } else {
    console.log("Keine Produkte gefunden");
  }

  console.debug("Zeit:", (end - start));
}

filterIndexBy("brand", key => key === "ZenCore")
```

```
Markenindex: [["PixelPeak",[0,19,24,26,30,42,56,63,87]],["ZenCore",
[1,15,29,32,35,65,71,73,76,81,83,91,92,94,95,96]],["Neutrino",
[2,6,10,17,21,27,33,34,57,72,82,84,89,97]],["Voltix",
[3,7,12,28,43,47,50,52,58,59,61,62,67,69,79,80,85,88,98,99]],
["OrbiTech",[4,5,8,37,38,40,41,46,48,51,60,70,74,77,86]],["Auralite",
[9,11,13,14,16,23,25,31,36,39,44,53,54,55,78,90,93]],["BlueWave",
[18,20,22,45,49,64,66,68,75]],["AeroLoom",
[100,104,108,111,112,113,115,122,130,140,149,158,162,164,165,175,176,177,1
["Cotton&Co",[101,121,136,137,146,147,155,156,161,167,179]],
["Nordline",
[102,105,110,125,128,139,143,144,145,151,152,157,163,170,171,174,182,183,1
["UrbanTrail",[103,106,107,109,134,154,169,187,189,192,194]],
["PureThread",
[114,117,119,120,123,129,131,141,150,159,166,168,172,173,180,198]],
["Seaside",[116,126,127,132,135,138,148,181,185,191]],["Meridian Wear",
[118,124,133,142,153,160,186,188,193,199]],["GreenVale",
[200,208,214,218,234,237,244,253,261,268,272,291,293]],["Everfarm",
[201,202,207,209,211,217,222,229,231,233,240,241,250,255,256,258,270,276,2
["DailyChoice",
[203,215,221,223,224,225,235,236,246,247,260,263,264,267,271,273,278,280,2
["FreshField",
[204,206,210,216,219,227,239,242,245,248,251,254,259,282,283,295]],
["SunHarvest",
[205,213,220,228,230,249,266,269,274,279,281,285,288,290,299]],
["Pure&Simple",[212,226,232,238,243,252,257,262,265,275,277,284,287]],
["DeskPro",
[300,301,302,309,318,322,337,338,353,358,361,363,366,367,369,373,381,391,3
["InkFlow",
[303,308,315,316,321,323,329,333,335,342,380,382,386,392,396,397]],
["WriteRight",
[304,310,313,314,317,324,330,331,336,339,341,344,346,348,350,355,362,372,3
["ErgoCraft",[305,319,340,345,351,368,370,375,383,384,393]],
["PaperMint",
[306,307,311,312,326,328,332,343,347,354,356,360,364,371,377,385,390]],
["StapleStar",
[320,325,327,334,349,352,357,359,365,374,376,378,388,389,395,399]],
["CozyNest",[400,402,410,413,414,429,434,435,437,448,449,458,467,476]],
["CasaNova",[401,421,426,428,439,456,466,477,487,499]],["Hearthstone",
[403,407,408,415,422,424,425,427,436,442,451,455,470,479,482,483,485,488,4
["KitchenCrafter",
[404,405,409,411,412,423,430,438,440,441,443,452,457,459,461,465,468,469,4
["BrightGlow",
[406,417,419,433,445,446,447,453,454,460,462,464,472,473,489,491,498]],
```

```
["AquaPure",[416,418,420,431,432,444,450,463,474,480,484,486,492,496]],
["RoboPal",
[500,502,504,510,513,515,516,521,526,543,554,555,571,575,578,588,590,599]]
["KiddyLab",[501,505,511,522,523,536,540,550,553,559,567,596]],
["SunnyPlay",
[503,506,507,508,512,518,525,527,548,560,562,565,566,569,574,581,582,584,5
["PlayVerse",
[509,524,533,534,535,541,546,549,558,564,570,572,591,595,598]],
["HappyBlocks",
[514,519,528,531,532,537,538,539,542,561,579,580,583,586,589,594]],
["WonderWorks",
[517,520,529,530,544,545,547,551,552,556,557,563,568,573,576,577,585,592]]
["TrailRunner",
[600,601,611,612,628,633,634,637,643,648,650,653,666,668,676,678,680,692,6
["AeroFlex",[602,609,614,618,632,635,640,645,656,660,661,675,690]],
["WaveRider",
[603,605,613,616,621,622,624,626,636,654,659,662,665,670,677,684,686]],
["PeakMotion",
[604,608,617,625,629,644,647,651,655,664,669,671,673,674,685,691]],
["CoreAthlete",[606,615,619,620,623,627,638,649,652,663,667,672,679]],
["FitFoundry",
[607,610,630,631,639,641,642,646,657,658,681,682,683,687,688,689,697,699]]
["OceanMist",[700,701,704,717,731,747,749,756,762,772,789,793]],
["SilkAura",
[702,705,711,718,722,724,725,728,732,736,744,745,746,755,757,760,761,770,7
["VelvetSkin",
[703,712,721,723,727,729,735,740,751,754,758,763,769,771,775,784,795,796,7
["PureGlow",
[706,709,714,715,716,720,730,738,741,742,765,768,778,786,791]],
["AurumCare",
[707,708,713,719,733,739,743,748,753,759,764,767,777,779,783,790]],
["LunaLeaf",
[710,726,734,737,750,752,766,773,774,781,785,787,788,792,794,799]],
["Atlas Editions",
[800,806,807,808,821,824,826,828,842,845,847,852,855,857,862,863,876,878,8
["Orion Leaf",
[801,802,829,844,846,849,851,861,865,868,870,875,883,887,889,893]],
["Silver Quill",
[803,804,814,830,831,836,837,838,840,843,850,853,859,866,867,869,874,881,8
["Northwind Press",
[805,810,812,815,818,820,822,823,834,835,872,877,880,895,896,897]],
["OpenPage",
[809,817,819,825,827,832,833,839,858,860,864,873,879,885,886]],["Sunset
```

```
House",[811,813,816,841,848,854,856,871,882,890]],["WildFeast",
[900,901,913,921,926,939,944,945,953,960,961,962,967,968,970,973,982,990,9
["Paw&Co",
[902,905,910,919,923,932,937,946,949,954,955,956,981,983,989,999]],
["PetNest",
[903,904,907,912,918,920,922,934,936,941,943,947,951,963,964,971,972,976,9
["FurryFriends",[906,914,916,917,931,933,935,940,969,975,991,992,998]],
["AquaPets",[908,924,925,928,929,930,938,948,959,966,979,984,986,996]],
["HappyTail",
[909,911,915,927,942,950,952,957,958,965,974,977,978,987,988,994,997]],
[null,[1000]]]
Kategorieindex: [["Electronics",
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27
["Clothing",
[100,101,102,103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,1
["Groceries",
[200,201,202,203,204,205,206,207,208,209,210,211,212,213,214,215,216,217,2
["Office",
[300,301,302,303,304,305,306,307,308,309,310,311,312,313,314,315,316,317,3
["Home & Kitchen",
[400,401,402,403,404,405,406,407,408,409,410,411,412,413,414,415,416,417,4
["Toys",
[500,501,502,503,504,505,506,507,508,509,510,511,512,513,514,515,516,517,5
["Sports",
[600,601,602,603,604,605,606,607,608,609,610,611,612,613,614,615,616,617,6
["Beauty",
[700,701,702,703,704,705,706,707,708,709,710,711,712,713,714,715,716,717,7
["Books",
[800,801,802,803,804,805,806,807,808,809,810,811,812,813,814,815,816,817,8
["Pets",
[900,901,902,903,904,905,906,907,908,909,910,911,912,913,914,915,916,917,9
[null,[1000]]]
Bestandsindex: [[138,[0]],[211,[1,285,478,682,719]],[261,[2,829]],[54,
[3,774,986]],[20,[4,335,385,989]],[66,[5,549,959,968]],[175,[6]],[188,
[7]],[312,[8,480,806,931]],[171,[9]],[1,[10,901]],[124,[11,994]],[127,
[12,360,750]],[69,[13,513,953]],[30,[14,57,640]],[325,[15]],[252,
[16,578,933]],[15,[17,77,964]],[198,[18]],[76,[19,527]],[248,
[20,655,885]],[336,[21,444]],[225,[22]],[163,[23,922]],[60,
[24,419,882]],[168,[25,92]],[283,[26,731,963]],[35,[27,187]],[228,
[28,63,132,697,841,995]],[122,[29,223,956]],[63,[30,501]],[141,[31]],
[203,[32,747,775]],[68,[33,159,169,504]],[219,[34,563,874]],[143,
[35,756,810,812]],[81,[36]],[199,[37,165,450]],[347,[38,467,862]],[13,
[39,300]],[152,[40,739]],[151,[41]],[64,[42]],[293,[43,344]],[130,
```

[44,50,873]],[24,[45,930]],[18,[46]],[53,[47,55]],[184,[48]],[67,
[49,540,886]],[256,[51,218,487]],[23,[52]],[218,[53,69,801,991]],[301,
[54,194]],[39,[56,146,490]],[173,[58,67,365,455]],[337,
[59,78,543,594]],[115,[60,637,654]],[103,[61,256]],[137,[62,508]],[5,
[64,95]],[79,[65,658,950]],[279,[66,485]],[272,[68]],[49,
[70,605,746,859]],[299,[71]],[169,[72]],[118,[73,843]],[238,[74,915]],
[222,[75]],[328,[76,657]],[229,[79]],[47,[80,461]],[154,[81,635]],[239,
[82]],[253,[83]],[4,[84]],[3,[85,131,140,449,853,877]],[46,[86,889]],
[285,[87,321,507]],[335,[88,97,443,776]],[37,[89,838]],[320,[90]],[202,
[91]],[161,[93]],[147,[94]],[48,[96,577,667]],[240,[98,386]],[207,
[99,482]],[92,[100,168,600]],[464,[101]],[295,[102,421]],[146,
[103,769,992]],[702,[104]],[548,[105]],[86,[106,966]],[351,[107,582]],
[473,[108]],[396,[109]],[425,[110,888]],[234,[111,325]],[647,[112]],
[761,[113,562]],[318,[114,529,536]],[148,[115,144]],[45,[116,414]],[88,
[117,446,934]],[331,[118,591]],[520,[119]],[701,[120]],[525,[121]],
[799,[122,523,648,788]],[730,[123,212,368]],[524,[124]],[354,
[125,483]],[150,[126,830]],[696,[127,541]],[580,[128,191,406,639]],
[393,[129,199]],[681,[130]],[303,[133,183,909]],[373,[134]],[107,
[135,945]],[44,[136,902]],[204,[137,587,875]],[748,[138]],[775,
[139,672]],[608,[141]],[70,[142]],[400,[143,151,910]],[536,[145,380]],
[476,[147]],[426,[148,973]],[434,[149,289,857]],[729,[150]],[440,
[152,437]],[785,[153,247]],[206,[154,827]],[311,[155,818]],[683,[156]],
[114,[157,495]],[40,[158,503]],[280,[160]],[587,[161,629]],[192,
[162,545,723]],[568,[163]],[617,[164]],[671,[166]],[714,[167]],[58,
[170,498]],[597,[171]],[642,[172]],[546,[173,624]],[687,[174,685]],[56,
[175,675]],[778,[176,179]],[477,[177,436,990]],[136,[178]],[783,
[180,343,716]],[410,[181,880]],[360,[182,207,439,783]],[534,[184,497]],
[661,[185]],[461,[186,331]],[772,[188,356]],[391,[189,690,876]],[732,
[190,674]],[287,[192,410,765]],[690,[193]],[465,[195]],[516,[196]],
[377,[197]],[713,[198,627]],[766,[200]],[1843,[201]],[1030,[202]],
[1126,[203,768]],[186,[204,555,971]],[1024,[205]],[1803,[206]],[1192,
[208]],[1054,[209,700]],[1804,[210]],[1990,[211]],[1172,[213]],[922,
[214,770]],[1407,[215]],[1946,[216]],[902,[217]],[1555,[219]],[615,
[220]],[421,[221,379]],[205,[222,546,918]],[863,[224,680]],[1088,
[225]],[959,[226]],[1552,[227]],[1111,[228]],[802,[229]],[1368,[230]],
[686,[231,398,938]],[57,[232]],[87,[233,476]],[1753,[234]],[596,
[235,391,911]],[1185,[236]],[213,[237,494]],[918,[238]],[1943,[239]],
[1960,[240]],[651,[241,583,952]],[420,[242]],[905,[243]],[737,[244]],
[1284,[245]],[1173,[246]],[1032,[248]],[540,[249]],[159,[250]],[1975,
[251]],[1101,[252]],[1286,[253]],[1468,[254]],[982,[255]],[1367,[257]],
[436,[258]],[832,[259]],[1441,[260]],[1889,[261]],[1775,[262]],[300,
[263]],[1109,[264]],[583,[265,295,312,733]],[774,[266]],[495,
[267,832]],[555,[268]],[1700,[269]],[1890,[270]],[1364,[271]],[1406,

[272]],[1159,[273]],[1132,[274]],[691,[275,512,919]],[388,[276]],[1139,
[277]],[353,[278,872,940]],[968,[279]],[878,[280]],[994,[281]],[1732,
[282]],[1974,[283]],[1313,[284]],[1547,[286]],[1711,[287]],[1086,
[288]],[1568,[290]],[1397,[291]],[1238,[292]],[1921,[293]],[689,
[294,363,677]],[1462,[296]],[1424,[297]],[1517,[298]],[1385,[299]],
[613,[301,997]],[52,[302]],[757,[303,722]],[220,[304,870]],[725,
[305,709]],[155,[306,351]],[510,[307,442]],[352,[308,914]],[372,
[309,405,424,824]],[743,[310]],[532,[311,789]],[896,[313,729]],[386,
[314,802]],[366,[315,535,805]],[796,[316,564]],[574,[317,999]],[359,
[318,404,492,604]],[716,[319]],[552,[320,452]],[189,[322]],[856,
[323,567,626,678]],[852,[324]],[375,[326]],[851,[327]],[341,[328]],
[604,[329]],[116,[330,961]],[445,[332,647]],[514,[333]],[41,
[334,338,345,441,489]],[403,[336]],[488,[337,539]],[406,[339]],[658,
[340]],[598,[341]],[805,[342]],[890,[346]],[160,[347,743,846]],[512,
[348]],[625,[349]],[482,[350]],[314,[352,460,506]],[348,[353,856]],
[820,[354]],[710,[355,652]],[442,[357,960]],[644,[358]],[843,
[359,593]],[112,[361]],[864,[362]],[588,[364,651]],[167,[366]],[827,
[367,712]],[125,[369,725,976]],[504,[370,496]],[868,[371]],[259,
[372,962]],[16,[373]],[32,[374,445]],[14,[375]],[417,[376,804]],[873,
[377,616]],[368,[378,407,935]],[637,[381]],[84,[382]],[273,[383]],[323,
[384]],[288,[387,860]],[340,[388,484,554,893]],[304,[389]],[571,[390]],
[2,[392,803]],[258,[393]],[821,[394,759]],[83,[395,796]],[182,
[396,551]],[404,[397]],[808,[399]],[485,[400]],[236,[401]],[521,[402]],
[100,[403,448,764]],[500,[408,977]],[117,[409,894]],[242,[411,650]],
[496,[412]],[149,[413,663]],[7,[415,748]],[109,[416,828,845]],[200,
[417,499]],[343,[418,532]],[367,[420,520]],[494,[422,619]],[108,
[423,792]],[223,[425]],[22,[426]],[582,[427]],[120,[428]],[478,
[429,799]],[467,[430,833]],[535,[431]],[291,[432,440,573]],[95,[433]],
[251,[434]],[544,[435,744]],[176,[438]],[586,[447]],[402,
[451,737,879]],[452,[453,851]],[309,[454,822]],[74,[456]],[530,
[457,592]],[289,[458]],[489,[459]],[537,[462]],[433,[463]],[85,
[464,470]],[385,[465,609]],[508,[466,479,525]],[594,[468]],[505,
[469,601,793]],[276,[471,608]],[153,[472]],[589,[473,988]],[201,
[474,865]],[358,[475,786]],[463,[477]],[316,[481,867]],[43,[486]],[578,
[488,662]],[174,[491,844,912]],[230,[493]],[268,[500,993]],[648,[502]],
[349,[505]],[319,[509]],[673,[510]],[170,[511]],[226,[514,569,826]],
[741,[515]],[113,[516,631]],[569,[517]],[460,[518]],[815,[519]],[34,
[521]],[638,[522,735]],[423,[524,836,847]],[736,[526]],[694,[528]],
[752,[530]],[451,[531,653,863]],[874,[533,618]],[179,[534]],[62,[537]],
[634,[538]],[703,[542]],[871,[544]],[232,[547,979]],[443,
[548,921,937]],[382,[550]],[866,[552]],[831,[553]],[409,[556,586,691]],
[492,[557]],[26,[558]],[458,[559]],[782,[560]],[378,[561]],[379,
[565,834]],[257,[566]],[511,[568]],[475,[570]],[131,[571,929]],[413,

```
[572]],[556,[574,694]],[646,[575]],[560,[576]],[834,[579]],[162,
[580,681]],[704,[581]],[25,[584,813,861]],[708,[585]],[479,[588]],[861,
[589]],[164,[590]],[416,[595,850]],[682,[596]],[636,[597]],[656,[598]],
[663,[599,715]],[828,[602]],[572,[603]],[695,[606]],[823,[607]],[453,
[610]],[466,[611]],[727,[612]],[657,[613]],[526,[614]],[794,[615]],[94,
[617]],[65,[620]],[50,[621]],[355,[622]],[891,[623,686]],[860,[625]],
[457,[628,749,975]],[422,[630]],[209,[632]],[797,[633]],[292,[634]],
[818,[636]],[503,[638]],[384,[641,871]],[829,[642,726]],[779,[643]],
[567,[644]],[881,[645]],[626,[646]],[392,[649,816,890]],[342,
[656,701]],[865,[659]],[575,[660]],[119,[661]],[557,[664]],[618,[665]],
[621,[666]],[787,[668]],[700,[669,903]],[305,[670]],[770,[671]],[448,
[673,926]],[698,[676,797]],[825,[679]],[853,[683,724]],[481,[684]],
[390,[687,855]],[759,[688]],[746,[689]],[753,[692]],[870,[693]],[183,
[695,928]],[502,[696]],[771,[698]],[75,[699,807]],[1117,[702]],[679,
[703]],[1158,[704]],[949,[705]],[948,[706]],[487,[707,840]],[290,
[708]],[711,[710]],[1151,[711]],[915,[713]],[889,[714]],[846,[717]],
[879,[718]],[407,[720]],[19,[721]],[652,[727]],[315,[728,946,949]],
[1180,[730]],[830,[732]],[165,[734]],[255,[736]],[728,[738]],[449,
[740]],[603,[741]],[177,[742]],[819,[745]],[428,[751,969]],[601,[752]],
[962,[753]],[584,[754,987]],[427,[755]],[966,[757]],[21,[758,762]],
[1029,[760]],[339,[761]],[227,[763,821]],[833,[766]],[850,[767]],[899,
[771]],[1010,[772]],[1160,[773]],[984,[777]],[1022,[778]],[1056,[779]],
[744,[780]],[437,[781]],[1136,[782]],[344,[784]],[909,[785]],[140,
[787]],[1040,[790]],[692,[791]],[1150,[794]],[786,[795]],[357,[798]],
[246,[800,967]],[9,[808]],[438,[809,936]],[178,[811]],[414,[814]],[450,
[815,892]],[145,[817,864]],[98,[819]],[324,[820]],[91,[823]],[241,
[825]],[104,[831,848]],[121,[835]],[362,[837]],[106,[839]],[221,
[842,981]],[243,[849]],[196,[852]],[216,[854]],[306,[858]],[55,[866]],
[274,[868,954]],[262,[869]],[102,[878,942]],[133,[881]],[317,[883]],
[157,[884]],[310,[887]],[419,[891]],[491,[895,908]],[462,[896]],[187,
[897]],[493,[898]],[361,[899]],[244,[900]],[547,[904]],[89,[905]],[688,
[906]],[271,[907]],[616,[913]],[156,[916]],[513,[917]],[284,[920]],
[675,[923]],[249,[924]],[484,[925]],[275,[927]],[576,[932]],[630,
[939]],[110,[941,983]],[381,[943]],[158,[944]],[61,[947]],[591,[948]],
[267,[951]],[541,[955]],[369,[957]],[518,[958]],[389,[965]],[264,
[970]],[17,[972]],[645,[974]],[610,[978]],[126,[980]],[602,[982]],[660,
[984]],[680,[985]],[101,[996]],[321,[998]],[null,[1000]]]
Gefunden: 16 Produkte
[
  {
    "name": "Portable SSD Max M",
    "category": "Electronics",
    "brand": "ZenCore",
```

```
    "price": 805.93,
    "stock": 211,
    "rating": 3.9,
    "created_at": "2024-05-21",
    "_id": "P00002"
  },
  {
    "name": "Action Camera - Blue",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 1322.9,
    "stock": 325,
    "rating": 4.8,
    "created_at": "2025-09-13",
    "_id": "P00016"
  },
  {
    "name": "Action Camera",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 1027.7,
    "stock": 122,
    "rating": 3.3,
    "created_at": "2024-09-05",
    "_id": "P00030"
  },
  {
    "name": "Smartwatch - White",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 61.61,
    "stock": 203,
    "rating": 2.9,
    "created_at": "2025-02-05",
    "_id": "P00033"
  },
  {
    "name": "Tablet 10\"",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 852.45,
    "stock": 143,
    "rating": 4.7,
```

```
    "created_at": "2024-05-23",
    "_id": "P00036"
  },
  {
    "name": "Soundbar Plus",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 146.53,
    "stock": 79,
    "rating": 4.3,
    "created_at": "2024-07-27",
    "_id": "P00066"
  },
  {
    "name": "Soundbar 2.0",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 231.87,
    "stock": 299,
    "rating": 3,
    "created_at": "2025-10-19",
    "_id": "P00072"
  },
  {
    "name": "USB-C Hub - Black",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 1337.23,
    "stock": 118,
    "rating": 4.6,
    "created_at": "2024-07-22",
    "_id": "P00074"
  },
  {
    "name": "Tablet 10\" S - Black",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 192.98,
    "stock": 328,
    "rating": 4.4,
    "created_at": "2025-06-14",
    "_id": "P00077"
  },
```

```
  {
    "name": "Gaming Mouse",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 317.38,
    "stock": 154,
    "rating": 3.6,
    "created_at": "2025-06-01",
    "_id": "P00082"
  },
  {
    "name": "4K Monitor XL",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 1389.73,
    "stock": 253,
    "rating": 4.7,
    "created_at": "2024-12-14",
    "_id": "P00084"
  },
  {
    "name": "Smartwatch",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 1145.45,
    "stock": 202,
    "rating": 4.1,
    "created_at": "2025-01-03",
    "_id": "P00092"
  },
  {
    "name": "Bluetooth Speaker - Blue",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 636.58,
    "stock": 168,
    "rating": 2.9,
    "created_at": "2024-05-03",
    "_id": "P00093"
  },
  {
    "name": "Gaming Mouse",
    "category": "Electronics",
```

```
    "brand": "ZenCore",
    "price": 92.53,
    "stock": 147,
    "rating": 5,
    "created_at": "2025-07-23",
    "_id": "P00095"
  },
  {
    "name": "Bluetooth Speaker",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 1493.82,
    "stock": 5,
    "rating": 4,
    "created_at": "2024-09-19",
    "_id": "P00096"
  },
  {
    "name": "Action Camera - White",
    "category": "Electronics",
    "brand": "ZenCore",
    "price": 129.17,
    "stock": 48,
    "rating": 5,
    "created_at": "2025-07-25",
    "_id": "P00097"
  }
]
Zeit: 63.29999999998836
```


Problem ohne Index:

Das durchsuchen aller Dokumente ist jedoch immernoch teuer. Schauen wir uns an, wie eine Query in PouchDB ohne Index intern abläuft. Lade alle Dokumente aus IndexedDB, parse jedes JSON-Dokument und prüfe, ob das Feld `category` den Wert `'Electronics'` hat. Das bedeutet: Wir müssen jedes einzelne Dokument prüfen – also $O(n)$O(n).

```
1  const db = new PouchDB('lecture_demo');
2
3  const start = performance.now();
4
5  // PouchDB OHNE Index
6  const products = await db.find({
7    selector: { category: 'Electronics' }
8  });
9
```

```
const end = performance.now();
console.log("⏱️  Zeit ohne Index:", (end - start).toFixed(2), "ms");

// Interner Ablauf:
// 1. Lade ALLE Dokumente aus IndexedDB
// 2. Für jedes Dokument:
//    - Parse JSON
//    - Prüfe: doc.category === 'Electronics'
// 3. Sammle Treffer
// Komplexität: O(n)
// Bei 10.000 Docs: ~10.000 Operationen

console.log(products.docs);
```

⏱️ Zeit ohne Index: 48.10 ms
[{"name":"Laptop 14\"
2.0","category":"Electronics","brand":"PixelPeak","price":1399.03,"stock":
06-14","_id":"P00001","_rev":"1-9f04c28afe20e1591157eab6468badeb"},
{"name":"Portable SSD Max
M","category":"Electronics","brand":"ZenCore","price":805.93,"stock":211,"
05-21","_id":"P00002","_rev":"1-555949b498ecee3c9636a7864a1ed37b"},
{"name":"4K Monitor -
Green","category":"Electronics","brand":"Neutrino","price":522.48,"stock":
10-16","_id":"P00003","_rev":"1-2d1821be050e0dd6b60ebf520e9edec3"},
{"name":"Tablet
10\"","category":"Electronics","brand":"Voltix","price":985.75,"stock":54,
04-16","_id":"P00004","_rev":"1-c6843bbcdfac1180e71dc3b662dab46c"},
{"name":"Portable SSD
Lite","category":"Electronics","brand":"OrbiTech","price":508.23,"stock":2
03-12","_id":"P00005","_rev":"1-df3d70067532eb326fb84c8d6a4b4795"},
{"name":"Soundbar -
Red","category":"Electronics","brand":"OrbiTech","price":786.98,"stock":66
09-07","_id":"P00006","_rev":"1-eb9be16ecf7d7ffdda04c4a18985bec1"},
{"name":"Mechanical Keyboard -
Black","category":"Electronics","brand":"Neutrino","price":537.59,"stock":
08-30","_id":"P00007","_rev":"1-61ec1d140622e541690233cc1490141f"},
{"name":"Laptop 14\"
Plus","category":"Electronics","brand":"Voltix","price":1187.46,"stock":18
03-13","_id":"P00008","_rev":"1-17f15836c57e7a1bea1a65350643f6b0"},
{"name":"Action Camera -
Green","category":"Electronics","brand":"OrbiTech","price":502.12,"stock":
03-27","_id":"P00009","_rev":"1-86198d189c1a54c4dc9a6b162c2879b0"},
{"name":"Soundbar","category":"Electronics","brand":"Auralite","price":134
02-25","_id":"P00010","_rev":"1-20c204a070f3727c39973e4f08548003"},
{"name":"4K
Monitor","category":"Electronics","brand":"Neutrino","price":24.92,"stock"
07-13","_id":"P00011","_rev":"1-5b8e2f5a4b3000038d95128d6683862e"},
{"name":"Action Camera
XL","category":"Electronics","brand":"Auralite","price":636.73,"stock":124
04-14","_id":"P00012","_rev":"1-ce0570a88e0e2c70def7423f29530b4f"},
{"name":"Gaming Mouse Plus
M","category":"Electronics","brand":"Voltix","price":787.02,"stock":127,"r
08-18","_id":"P00013","_rev":"1-492a09e47772e7bce2da1027ea2389b0"},
{"name":"Bluetooth Speaker
Plus","category":"Electronics","brand":"Auralite","price":865.42,"stock":6
02-23","_id":"P00014","_rev":"1-b8af9ee6d5b789dc92a7b6d60b4722d3"},
{"name":"Noise-Canceling Earbuds

```
Lite","category":"Electronics","brand":"Auralite","price":1339.39,"stock":
10-12","_id":"P00015","_rev":"1-cf519299d44a7e9353f3726a7c6cebf6"},
{"name":"Action Camera -
Blue","category":"Electronics","brand":"ZenCore","price":1322.9,"stock":32
09-13","_id":"P00016","_rev":"1-864e301d319fdbcf80fc810caec9e7f3"},
{"name":"Laptop 14\" Pro
S","category":"Electronics","brand":"Auralite","price":481.53,"stock":252,
09-19","_id":"P00017","_rev":"1-f356f026028fe3ca97d14029c5417828"},
{"name":"Gaming
Mouse","category":"Electronics","brand":"Neutrino","price":1402.13,"stock"
06-28","_id":"P00018","_rev":"1-af83117f9f5fc91a71e9b8c154f8fb82"},
{"name":"USB-C Hub -
White","category":"Electronics","brand":"BlueWave","price":1093.57,"stock"
09-08","_id":"P00019","_rev":"1-3f579988745061cc7e37b631ac7cbead"},
{"name":"USB-C Hub -
Red","category":"Electronics","brand":"PixelPeak","price":317.33,"stock":7
06-25","_id":"P00020","_rev":"1-3d0d828d9f1a3c77e40a62f05167e247"},
{"name":"Wireless
Headphones","category":"Electronics","brand":"BlueWave","price":377.2,"sto
10-17","_id":"P00021","_rev":"1-b846b9c175fe914fdf61dba334943e6d"},
{"name":"Laptop 14\" 2.0 -
Black","category":"Electronics","brand":"Neutrino","price":833.21,"stock":
08-26","_id":"P00022","_rev":"1-d944003353ed16dcf8166228dec9b41e"},
{"name":"Wireless Headphones
M","category":"Electronics","brand":"BlueWave","price":697.05,"stock":225,
09-17","_id":"P00023","_rev":"1-dd439892547e9a222703037aef3f49d1"},
{"name":"Tablet 10\" -
Green","category":"Electronics","brand":"Auralite","price":667.7,"stock":1
12-03","_id":"P00024","_rev":"1-ddad3dd805ccd126c1dc8299c4d9c13f"},
{"name":"Soundbar
Plus","category":"Electronics","brand":"PixelPeak","price":739.53,"stock":
08-28","_id":"P00025","_rev":"1-0bf0c0f9b9dd387404e3ac890f4a40c0"}]
```


Lösung mit Index:

Jetzt erstellen wir einen Index auf dem Feld `category` und führen die gleiche Query erneut aus. PouchDB nutzt den Index, um direkt die Dokumente mit `category='Electronics'` zu finden, ohne alle Dokumente zu prüfen. Das reduziert die Anzahl der Operationen drastisch auf $O(\log n) + O(k)$ O(log n) + O(k), wobei kk die Anzahl der Treffer ist. Warum

$$O(\log n)$$

O(log n)? Weil der Index meist als B-Tree implementiert ist, der logarithmische Suchzeiten ermöglicht.

```
1  const db = new PouchDB('lecture_demo', {adapter: 'memory'});
2  await db.bulkDocs(window.shopData);
3
```

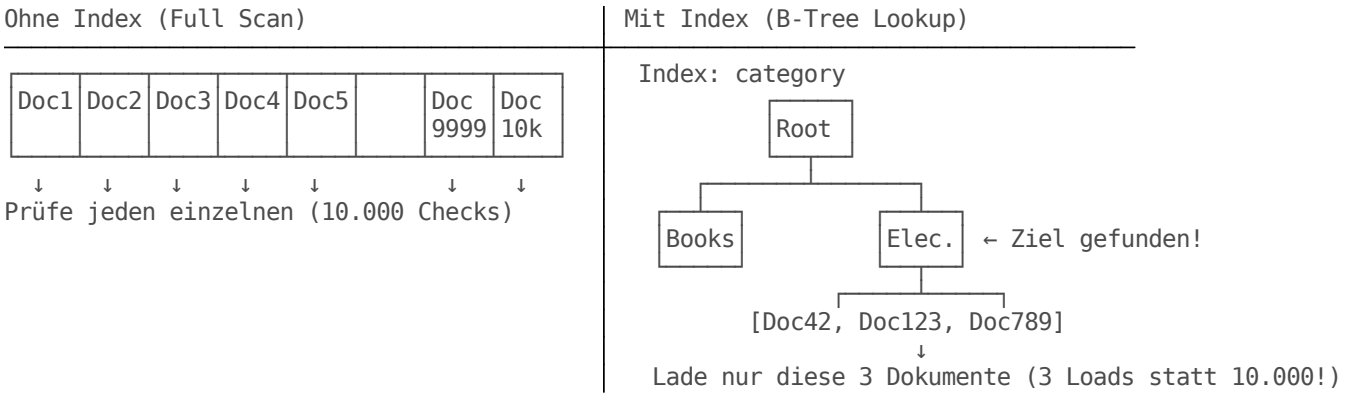
```
// Index erstellen (einmalig)
await db.createIndex({
  index: { fields: ['category'] }
});

const start = performance.now();
// Gleiche Query - jetzt mit Index
const products = await db.find({
  selector: { category: 'Electronics' }
});

const end = performance.now();
console.log("⏱️  Zeit mit Index:", (end - start).toFixed(2), "ms");
console.log(products.docs);

// Interner Ablauf:
// 1. Lookup in Index: category='Electronics' → [Doc-IDs]
// 2. Lade nur diese Dokumente
// Komplexität: O(log n) + O(k)  // k = Anzahl Treffer
// Bei 10.000 Docs, 100 Treffer: ~14 + 100 Operationen
```

```
⏱️  Zeit mit Index: 17.90 ms
[{"name":"Laptop 14\"
2.0","category":"Electronics","brand":"PixelPeak","price":1399.03,"stock":
06-14","_id":"P00001","_rev":"1-9f04c28afe20e1591157eab6468badeb"},
{"name":"Portable SSD Max
M","category":"Electronics","brand":"ZenCore","price":805.93,"stock":211,"
05-21","_id":"P00002","_rev":"1-555949b498ecee3c9636a7864a1ed37b"},
{"name":"4K Monitor -
Green","category":"Electronics","brand":"Neutrino","price":522.48,"stock":
10-16","_id":"P00003","_rev":"1-2d1821be050e0dd6b60ebf520e9edec3"},
{"name":"Tablet
10\"","category":"Electronics","brand":"Voltix","price":985.75,"stock":54,
04-16","_id":"P00004","_rev":"1-c6843bbcdfac1180e71dc3b662dab46c"},
{"name":"Portable SSD
Lite","category":"Electronics","brand":"OrbiTech","price":508.23,"stock":2
03-12","_id":"P00005","_rev":"1-df3d70067532eb326fb84c8d6a4b4795"},
{"name":"Soundbar -
Red","category":"Electronics","brand":"OrbiTech","price":786.98,"stock":66
09-07","_id":"P00006","_rev":"1-eb9be16ecf7d7ffdda04c4a18985bec1"},
{"name":"Mechanical Keyboard -
Black","category":"Electronics","brand":"Neutrino","price":537.59,"stock":
08-30","_id":"P00007","_rev":"1-61ec1d140622e541690233cc1490141f"},
{"name":"Laptop 14\"
Plus","category":"Electronics","brand":"Voltix","price":1187.46,"stock":18
03-13","_id":"P00008","_rev":"1-17f15836c57e7a1bea1a65350643f6b0"},
{"name":"Action Camera -
Green","category":"Electronics","brand":"OrbiTech","price":502.12,"stock":
03-27","_id":"P00009","_rev":"1-86198d189c1a54c4dc9a6b162c2879b0"},
{"name":"Soundbar","category":"Electronics","brand":"Auralite","price":134
02-25","_id":"P00010","_rev":"1-20c204a070f3727c39973e4f08548003"},
{"name":"4K
Monitor","category":"Electronics","brand":"Neutrino","price":24.92,"stock"
07-13","_id":"P00011","_rev":"1-5b8e2f5a4b3000038d95128d6683862e"},
{"name":"Action Camera
XL","category":"Electronics","brand":"Auralite","price":636.73,"stock":124
04-14","_id":"P00012","_rev":"1-ce0570a88e0e2c70def7423f29530b4f"},
{"name":"Gaming Mouse Plus
M","category":"Electronics","brand":"Voltix","price":787.02,"stock":127,"r
08-18","_id":"P00013","_rev":"1-492a09e47772e7bce2da1027ea2389b0"},
{"name":"Bluetooth Speaker
Plus","category":"Electronics","brand":"Auralite","price":865.42,"stock":6
02-23","_id":"P00014","_rev":"1-b8af9ee6d5b789dc92a7b6d60b4722d3"},
{"name":"Noise-Canceling Earbuds
```

```
Lite","category":"Electronics","brand":"Auralite","price":1339.39,"stock":
10-12","_id":"P00015","_rev":"1-cf519299d44a7e9353f3726a7c6cebf6"},
{"name":"Action Camera -
Blue","category":"Electronics","brand":"ZenCore","price":1322.9,"stock":32
09-13","_id":"P00016","_rev":"1-864e301d319fdbcf80fc810caec9e7f3"},
{"name":"Laptop 14\" Pro
S","category":"Electronics","brand":"Auralite","price":481.53,"stock":252,
09-19","_id":"P00017","_rev":"1-f356f026028fe3ca97d14029c5417828"},
{"name":"Gaming
Mouse","category":"Electronics","brand":"Neutrino","price":1402.13,"stock"
06-28","_id":"P00018","_rev":"1-af83117f9f5fc91a71e9b8c154f8fb82"},
{"name":"USB-C Hub -
White","category":"Electronics","brand":"BlueWave","price":1093.57,"stock"
09-08","_id":"P00019","_rev":"1-3f579988745061cc7e37b631ac7cbead"},
{"name":"USB-C Hub -
Red","category":"Electronics","brand":"PixelPeak","price":317.33,"stock":7
06-25","_id":"P00020","_rev":"1-3d0d828d9f1a3c77e40a62f05167e247"},
{"name":"Wireless
Headphones","category":"Electronics","brand":"BlueWave","price":377.2,"sto
10-17","_id":"P00021","_rev":"1-b846b9c175fe914fdf61dba334943e6d"},
{"name":"Laptop 14\" 2.0 -
Black","category":"Electronics","brand":"Neutrino","price":833.21,"stock":
08-26","_id":"P00022","_rev":"1-d944003353ed16dcf8166228dec9b41e"},
{"name":"Wireless Headphones
M","category":"Electronics","brand":"BlueWave","price":697.05,"stock":225,
09-17","_id":"P00023","_rev":"1-dd439892547e9a222703037aef3f49d1"},
{"name":"Tablet 10\" -
Green","category":"Electronics","brand":"Auralite","price":667.7,"stock":1
12-03","_id":"P00024","_rev":"1-ddad3dd805ccd126c1dc8299c4d9c13f"},
{"name":"Soundbar
Plus","category":"Electronics","brand":"PixelPeak","price":739.53,"stock":
08-28","_id":"P00025","_rev":"1-0bf0c0f9b9dd387404e3ac890f4a40c0"}]
```


Visualisierung:

Das ist das Kernprinzip von Indizes: Statt alle Dokumente zu durchsuchen, bauen wir eine separate Datenstruktur – meist einen B-Tree –, die von Feldwerten zu Dokument-IDs verweist. Das kostet Speicherplatz und macht Writes langsamer, aber beschleunigt Reads dramatisch. Dieser Trade-off ist fundamental für alle Datenbanksysteme.



Messungen mit mehr Indizes

Führen Sie dieses Experiment aus:

Versuchen sie den folgenden Code, der den Unterschied in der Performance bei einer Query mit und ohne Index zeigt. Starten Sie zunächst mit 200 Produkte mit zufälligen Kategorien und Beständen, führen eine Query durch, messen die Zeit und erstellen dann einen Index auf den Feldern `category` und `stock`. Anschließend führen wir die gleiche Query aber mit mehr Produkten erneut aus, indem sie `NUMBER_OF_PRODUCTS` erhöhen, und messen die Zeit erneut.

```
// Setup: 200 Produkte generieren
const NUMBER_OF_PRODUCTS = 200;

const db = new PouchDB('performance_test', {adapter: 'memory'});

const products = [];
for (let i = 0; i < NUMBER_OF_PRODUCTS; i++) {
  products.push({
    _id: `product_${String(i).padStart(4, '0')}`,
    name: `Product ${i}`,
    category: ['Electronics', 'Books', 'Clothing'][i % 3],
    price: Math.floor(Math.random() * 500) + 10,
    stock: Math.floor(Math.random() * 100)
  });
}

await db.bulkDocs(products);
console.log('✅', NUMBER_OF_PRODUCTS, 'Produkte eingefügt');

// Test 1: Query OHNE Index
let t0 = performance.now();
const result1 = await db.find({
  selector: {
    category: 'Electronics',
    stock: { $lt: 10 }
  }
});
let t1 = performance.now();
console.log(`⏱️ Ohne Index: ${(t1 - t0).toFixed(2)} ms`);
```

```
30 console.log(`   Gefunden: ${result1.docs.length} Produkte`);
31
32 // Index erstellen
33 await db.createIndex({
34   index: { fields: ['category', 'stock'] }
35 });
36 console.log('✅ Index auf [category, stock] erstellt');
37
38 // Test 2: Query MIT Index
39 t0 = performance.now();
40 const result2 = await db.find({
41   selector: {
42     category: 'Electronics',
43     stock: { $lt: 10 }
44   }
45 });
46 t1 = performance.now();
47 console.log(`⏱️ Mit Index: ${(t1 - t0).toFixed(2)} ms`);
48 console.log(`   Gefunden: ${result2.docs.length} Produkte`);
49
50 // Cleanup
51 await db.destroy();
```

```
✅ 200 Produkte eingefügt
⏱️ Ohne Index: 11.20 ms
   Gefunden: 5 Produkte
✅ Index auf [category, stock] erstellt
⏱️ Mit Index: 23.40 ms
   Gefunden: 5 Produkte
```

Erwartetes Ergebnis bei 20000 Produkten:

Bei 20.000 Produkten sollten Sie eine dramatische Verbesserung der Suchzeit sehen. Ohne Index muss PouchDB alle 20.000 Dokumente prüfen, was viel Zeit in Anspruch nimmt. Mit dem Index kann PouchDB direkt zu den relevanten Dokumenten springen, was die Suchzeit erheblich reduziert.

```
✅ 20000 Produkte eingefügt
⏱️ Ohne Index: 283.00 ms
   Gefunden: 25 Produkte
✅ Index auf [category, stock] erstellt
⏱️ Mit Index: 48.00 ms
   Gefunden: 25 Produkte
```

Beobachtung:

Der Performance-Gewinn durch Indizes wird mit zunehmender Datenmenge immer deutlicher. Hier sind einige typische Ergebnisse, die Sie erwarten können.

- Speedup: kein gain bei 1.000 Dokumenten
- Bei 10.000 Docs: Speedup **~3x**
- Bei 100.000 Docs: Speedup **~200x**

> **Die Moral:** Indizes sind nicht optional für Production-Datenbanken – sie sind essentiell!

Diese Live-Demo sollte den Unterschied greifbar machen. Bei kleinen Datenmengen wirken Indizes wie Overhead – aber sie skalieren logarithmisch ($O(\log n)$O(log n)), während Scans linear wachsen ($O(n)$O(n)).

Das ist der Unterschied zwischen einer App, die bei 10.000 Nutzern zusammenbricht, und einer, die auf 10 Millionen skaliert.

Sekundärindizes: Das neue Werkzeug

Was sind Sekundärindizes?

Sekundärindizes sind zusätzliche Suchstrukturen in einer Datenbank, die es ermöglichen, gezielt nach beliebigen Feldwerten zu suchen – zum Beispiel nach Kategorie, Preis oder Status. Während der Primärindex (meist das Feld „_id“) schnellen Zugriff auf einzelne Dokumente bietet, verknüpft ein Sekundärindex einen Feldwert wie „category“ mit einer Liste von passenden Dokument-IDs. So können Sie blitzschnell alle Produkte einer bestimmten Kategorie finden, ohne die gesamte Datenbank durchsuchen zu müssen. Sekundärindizes sind damit das zentrale Werkzeug für effiziente Filter, Sortierungen und komplexe Abfragen in Document Stores.

```
Primärschlüssel (_id):
_____________________

_id → Dokument

'product_001' → {
  name: 'Laptop',
  category: 'Electronics'
}

'product_002' → {
  name: 'Novel',
  category: 'Books'
}
```

```
Sekundärindex (category):
_________________________

Feldwert → [_ids]

'Electronics' → [
  'product_001',
  'product_003',
  'product_042'
]

'Books' → [
  'product_002',
  'product_017'
]
```

Arten von Indizes

Index-Typ	Beschreibung	Beispiel
Single-Field	Index auf einem Feld	`['category']`
Composite	Index auf mehreren Feldern	`['category', 'price']`
Sparse	Nur Dokumente mit dem Feld	Automatisch in PouchDB
Unique	Werte müssen eindeutig sein	Nicht nativ in PouchDB

Ein **Single-Field-Index** ist die einfachste Form eines Indexes: Er wird auf genau ein Feld gelegt, zum Beispiel auf „category“. Damit können Sie sehr schnell nach allen Dokumenten suchen, die einen bestimmten Wert in diesem Feld haben. Ideal für häufig genutzte Filter wie Produkttypen oder Status.

Ein **Composite-Index** kombiniert mehrere Felder, etwa „category“ und „price“. Damit werden komplexe Abfragen beschleunigt, die beide Felder gleichzeitig nutzen – zum Beispiel alle Elektronikprodukte mit einem Preis über 100 Euro. Die Reihenfolge der Felder ist dabei entscheidend für die Performance.

Ein **Sparse-Index** enthält nur Dokumente, die das betreffende Feld tatsächlich besitzen. Das ist besonders nützlich, wenn nicht alle Dokumente gleich aufgebaut sind. In PouchDB werden Sparse-Indizes automatisch erzeugt, wenn Sie auf optionale Felder indexieren.

Ein **Unique-Index** stellt sicher, dass jeder Wert im indexierten Feld nur einmal vorkommt – zum Beispiel bei einer eindeutigen Produktnummer. In klassischen Datenbanken ist das Standard, in PouchDB gibt es Unique-Indizes allerdings nicht nativ. Sie müssen Eindeutigkeit selbst sicherstellen.

Wichtig:

Beim Indexieren gilt: Sie können grundsätzlich jedes Feld in Ihren Dokumenten indexieren, mit Ausnahme von Arrays und verschachtelten Objekten, die gewisse Einschränkungen haben. Es ist möglich, mehrere Indizes pro Datenbank anzulegen, um verschiedene Abfragewege zu beschleunigen. Bedenken Sie aber: Jeder zusätzliche Index verbraucht Speicher und verlangsamt Schreibvorgänge, weil bei jedem Insert oder Update alle Indizes aktualisiert werden müssen. Zu viele Indizes führen sogar zu Performance-Problemen – setzen Sie sie also gezielt und nur dort ein, wo sie wirklich gebraucht werden.

- ✅ Jedes Feld kann indexiert werden (außer Arrays, Nested Objects haben Einschränkungen)
- ✅ Mehrere Indizes pro Datenbank möglich
- ⚠️ Jeder Index kostet Speicher + verlangsamt Writes
- ⚠️ Zu viele Indizes = Performance-Regression!

Index-Planung: Welche Felder indexieren?

Entscheidungskriterien:

Die Wahl, welche Felder Sie indexieren, ist ein Balanceakt zwischen Performance und Ressourcenverbrauch. Indexieren Sie vor allem Felder, die häufig in Abfragen oder Sortierungen verwendet werden und eine hohe Selektivität besitzen – also viele unterschiedliche Werte. So profitieren Sie maximal von schnellen Lookups. Bedenken Sie aber: Jeder Index kostet Speicher und verlangsamt Schreibvorgänge. Setzen Sie Indizes gezielt und nur dort ein, wo sie wirklich gebraucht werden – alles andere ist teurer Overhead.

1. **Häufigkeit:** Wird das Feld oft in Queries verwendet?
2. **Selektivität:** Hat das Feld viele unterschiedliche Werte? (Hoch = gut für Index)
3. **Sortierung:** Wird nach dem Feld sortiert?
4. **Kosten:** Indizes verlangsamen Writes und verbrauchen Speicher

Beispiel-Bewertung (Produktkatalog):

Diese Tabelle zeigt die Felder unseres Produktkatalogs und wie man sie aus Index-Sicht bewertet. Felder mit hoher Abfragehäufigkeit und Selektivität – wie `product_id`, `category`, `brand`, `price` und `stock` – sind ideale Kandidaten für einen Index. Felder wie `name` oder `rating` werden eventuell selten gefiltert und profitieren daher kaum von einem Index. Die Entscheidung basiert immer auf echten Query-Patterns und nicht auf Bauchgefühl.

Feld	Häufigkeit	Selektivität	Index?	Begründung
`product_id`	Hoch	Sehr hoch (unique)	✅ Ja	Eindeutige Identifikation, Primärschlüssel
`name`	Mittel	Hoch	❌ Nein	Selten Filter, oft Fulltext-Suche nötig
`category`	Hoch	Mittel (5-10 Werte)	✅ Ja	Häufige Filterung nach Produkttyp
`brand`	Hoch	Mittel (10-20 Werte)	✅ Ja	Filterung nach Hersteller/Marke
`price`	Mittel	Hoch (viele Preise)	✅ Ja	Sortierung und Prei Range-Queries
`stock`	Mittel	Hoch	✅ Ja	Filterung nach Verfügbarkeit/Besta
`rating`	Niedrig	Mittel (1-5 Werte)	❌ Nein	Selten Query, meist für Sortierung
`created_at`	Niedrig	Hoch	❌ Nein	Meist Sortierung, selten Filter

Faustregel:

Zu viele Indizes sind schädlich: Jeder Write muss alle Indizes aktualisieren. Ein häufiger Fehler: „Ich erstelle Indizes auf alle Felder, dann bin ich safe.“ Falsch! Sie zahlen mit Schreib-Performance und Speicher für Indizes, die nie genutzt werden. Indexieren Sie gezielt basierend auf realen Query-Patterns.

> Indexiere Felder, die in `selector` und `sort` auftauchen – aber nicht alle!

GeoJSON - Spezialfälle für Geodaten

Was ist GeoJSON?

GeoJSON ist ein offener Standard zur Beschreibung geografischer Daten im JSON-Format. Es ermöglicht die einfache Speicherung, Übertragung und Visualisierung von Geodaten in Document Stores wie MongoDB oder CouchDB. GeoJSON ist besonders für Webanwendungen geeignet, da es auf dem weit verbreiteten JSON-Format basiert und von vielen Tools direkt unterstützt wird.

- **GeoJSON** ist ein JSON-basiertes Format zur Darstellung von Geometrien und geografischen Features.
- Es beschreibt Punkte, Linien, Flächen und deren Eigenschaften als strukturierte Objekte.

Haupttypen in GeoJSON

GeoJSON definiert mehrere Geometrietypen, die in Dokumenten verwendet werden können. Mit diesen Typen lassen sich Standorte, Punkte, Wege und Flächen, Sehenswürdigkeiten präzise und strukturiert in JSON-Dokumenten abbilden.

- **Point:** Ein einzelner geografischer Punkt, definiert durch Koordinaten (Länge, Breite).

```
{
  "type": "Point",
  "coordinates": [13.4050, 52.5200] // Berlin
}
```

- **LineString:** Eine Linie, die aus einer Reihe von Punkten besteht.

```
{
  "type": "LineString",
  "coordinates": [
    [13.3426, 50.9106], // Freiberg
    [13.4050, 52.5200], // Berlin
    [11.5761, 48.1374]  // München
  ]
}
```

- **Polygon:** Eine Fläche, die durch eine geschlossene Linie definiert ist.

```
{
  "type": "Polygon",
  "coordinates": [[
    [13.0, 50.5], [13.0, 51.5],
    [14.0, 51.5], [14.0, 50.5],
    [13.0, 50.5]
  ]]
}
```

- **MultiPoint, MultiLineString, MultiPolygon:** Sammlungen der jeweiligen Geometrietypen.

```
{
  "type": "MultiPoint",
  "coordinates": [
    [13.4050, 52.5200], // Berlin
    [11.5761, 48.1374]  // München
  ]
}
```

- **Feature:** Ein einzelnes geografisches Objekt mit Geometrie und Eigenschaften.

```
{
  "type": "Feature",
  "properties": { "name": "Berlin" },
  "geometry": {
    "type": "Point",
    "coordinates": [13.4050, 52.5200]
  }
```

```
}
```

- **FeatureCollection:** Eine Sammlung von Features.

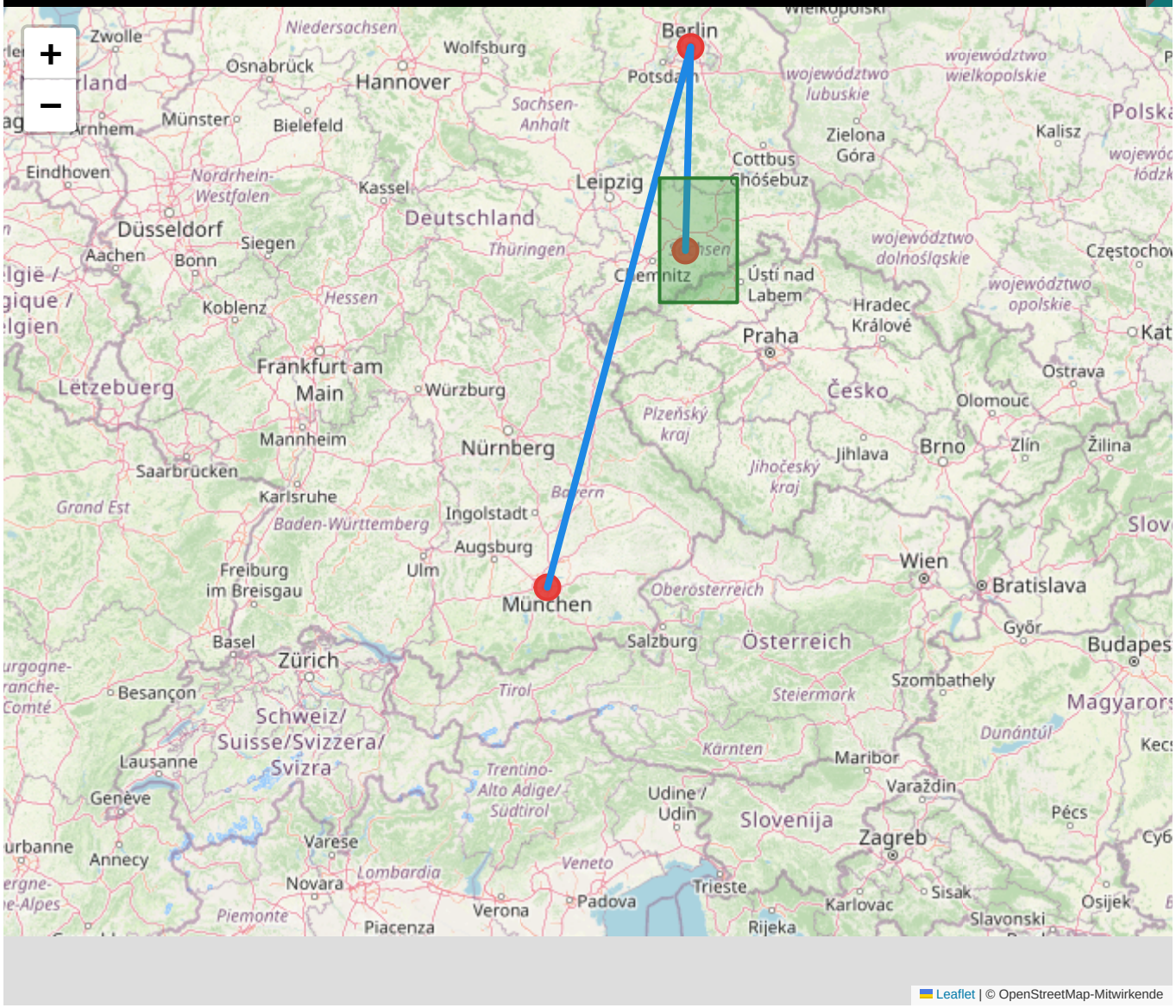
```
{
  "type": "FeatureCollection",
  "features": [
    // Array von Feature-Objekten
  ]
}
```

Wenn Sie GeoJSON-Daten visualisieren möchten, können Sie Bibliotheken wie Leaflet verwenden. Hier ein einfaches Beispiel, das Punkte, Linien und Flächen auf einer Karte darstellt.

```
{
  "type": "FeatureCollection",
  "features": [
    // Punkt(e)
    {
      "type": "Feature",
      "properties": { "name": "Freiberg" },
      "geometry": { "type": "Point", "coordinates": [13.3426, 50.9106
    },
    {
      "type": "Feature",
      "properties": { "name": "Berlin" },
      "geometry": { "type": "Point", "coordinates": [13.4050, 52.5200
    },
    {
      "type": "Feature",
      "properties": { "name": "München" },
      "geometry": { "type": "Point", "coordinates": [11.5761, 48.1374
    },
    // Linie
    {
      "type": "Feature",
      "properties": { "name": "Beispiel-Linie" },
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [13.3426, 50.9106], // Freiberg
          [13.4050, 52.5200], // Berlin
          [11.5761, 48.1374]  // München
        ]
      }
    },
    // Fläche
    {
      "type": "Feature",
      "properties": { "name": "Beispiel-Polygon" },
      "geometry": {
```

```
            "type": "Polygon",
            "coordinates": [[
              [13.0, 50.5],
              [13.0, 51.5],
              [14.0, 51.5],
              [14.0, 50.5],
              [13.0, 50.5]
            ]]
          }
        }
      ]
    }
```

[object Object]



R-Tree Index für GeoJSON

R-Trees sind spezielle Datenstrukturen, die für die effiziente Verwaltung und Suche von räumlichen Objekten wie Punkten, Linien und Flächen entwickelt wurden. Sie werden häufig als Index für GeoJSON-Daten in Datenbanken und GIS-Systemen eingesetzt.

Funktionsweise:

- Ein R-Tree organisiert räumliche Objekte in hierarchischen, überlappenden Rechtecken („Bounding Boxes“).
- Jeder Knoten im Baum fasst nahe beieinander liegende Objekte zusammen.
- Beim Suchen prüft das System zuerst die Bounding Boxes und kann große Bereiche schnell ausschließen, bevor es die eigentlichen Geometrien vergleicht.

Typische Abfragen:

- „Finde alle Punkte im Umkreis von X“ (Umkreissuche)
- „Welche Flächen schneiden dieses Gebiet?“ (Schnittmengen)
- „Welche Objekte liegen in einem bestimmten Rechteck?“ (Bereichssuche)
- Nachbarschaftsabfragen und räumliche Filter

R-Trees machen räumliche Suchen schnell und skalierbar – ein zentrales Werkzeug für Geo-Datenbanken und Kartenanwendungen.

Features.json


```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": { "name": "Freiberg Area" },
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [12.542599999999998, 50.110600000000005],
            [12.542599999999998, 51.7106],
            [14.1426, 51.7106],
            [14.1426, 50.110600000000005],
            [12.542599999999998, 50.110600000000005]
          ]
        ]
      }
    },
    {
      "type": "Feature",
      "properties": { "name": "Freiberg L1 P1" },
      "geometry": {
        "type": "Point",
        "coordinates": [13.3426, 50.9106]
      }
    },
    {
      "type": "Feature",
      "properties": { "name": "Freiberg L1 P2" },
      "geometry": {
        "type": "Point",
        "coordinates": [13.5926, 50.9106]
      }
    },
    {
      "type": "Feature",
      "properties": { "name": "Freiberg L1 P3" },
      "geometry": {
        "type": "Point",
        "coordinates": [13.0926, 50.9106]
      }
    },
    {
      "type": "Feature",
      "properties": { "name": "Freiberg L1 P4" },
      "geometry": {
        "type": "Point",
```

```
      "coordinates": [13.3426, 51.1606]
    }
  },
  {
    "type": "Feature",
    "properties": { "name": "Freiberg L1 P5" },
    "geometry": {
      "type": "Point",
      "coordinates": [13.3426, 50.6606]
    }
  },
  {
    "type": "Feature",
    "properties": { "name": "Freiberg L1 P6" },
    "geometry": {
      "type": "Point",
      "coordinates": [13.5926, 51.1606]
    }
  },
  {
    "type": "Feature",
    "properties": { "name": "Freiberg L1 P7" },
    "geometry": {
      "type": "Point",
      "coordinates": [13.5926, 50.6606]
    }
  },
  {
    "type": "Feature",
    "properties": { "name": "Freiberg L1 P8" },
    "geometry": {
      "type": "Point",
      "coordinates": [13.0926, 51.1606]
    }
  },
  {
    "type": "Feature",
    "properties": { "name": "Freiberg L1 P9" },
    "geometry": {
      "type": "Point",
      "coordinates": [13.0926, 50.6606]
    }
  },
  {
    "type": "Feature",
    "properties": { "name": "Freiberg L2 Hull E" },
    "geometry": {
      "type": "Polygon",
      "coordinates": [
        [
          [13.5026, 50.8206]
```

```
            [13.5026, 51.0006],
            [13.6826, 51.0006],
            [13.6826, 50.8206],
            [13.5026, 50.8206]
          ]
        ]
      }
    },
    {
      "type": "Feature",
      "properties": { "name": "Freiberg L2 Hull W" },
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [12.912599999999999, 50.8206],
            [12.912599999999999, 51.0006],
            [13.0926, 51.0006],
            [13.0926, 50.8206],
            [12.912599999999999, 50.8206]
          ]
        ]
      }
    },
    {
      "type": "Feature",
      "properties": { "name": "Freiberg L2 Hull N" },
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [13.2526, 51.0706],
            [13.2526, 51.2506],
            [13.4326, 51.2506],
            [13.4326, 51.0706],
            [13.2526, 51.0706]
          ]
        ]
      }
    },
    {
      "type": "Feature",
      "properties": { "name": "Freiberg L2 Hull S" },
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [13.2526, 50.5706],
            [13.2526, 50.7506],
            [13.4326, 50.7506],
```

```
            [13.4326, 50.5706],
            [13.2526, 50.5706]
          ]
        ]
      }
    },
    {
      "type": "Feature",
      "properties": { "name": "Freiberg Spine" },
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [13.0926, 50.9106],
          [13.3426, 51.1606],
          [13.5926, 50.9106]
        ]
      }
    },
    {
      "type": "Feature",
      "properties": { "name": "Freiberg Diagonal" },
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [12.542599999999998, 50.110600000000005],
          [14.1426, 51.7106]
        ]
      }
    },

    {
      "type": "Feature",
      "properties": { "name": "Berlin Area" },
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [12.605, 51.72],
            [12.605, 53.32],
            [14.205, 53.32],
            [14.205, 51.72],
            [12.605, 51.72]
          ]
        ]
      }
    },
    {
      "type": "Feature",
      "properties": { "name": "Berlin L1 P1" },
      "geometry": {
        "type": "Point",
```

```
      "coordinates": [13.405, 52.52]
    }
  },
  {
    "type": "Feature",
    "properties": { "name": "Berlin L1 P2" },
    "geometry": {
      "type": "Point",
      "coordinates": [13.655, 52.52]
    }
  },
  {
    "type": "Feature",
    "properties": { "name": "Berlin L1 P3" },
    "geometry": {
      "type": "Point",
      "coordinates": [13.155, 52.52]
    }
  },
  {
    "type": "Feature",
    "properties": { "name": "Berlin L1 P4" },
    "geometry": {
      "type": "Point",
      "coordinates": [13.405, 52.77]
    }
  },
  {
    "type": "Feature",
    "properties": { "name": "Berlin L1 P5" },
    "geometry": {
      "type": "Point",
      "coordinates": [13.405, 52.27]
    }
  },
  {
    "type": "Feature",
    "properties": { "name": "Berlin L1 P6" },
    "geometry": {
      "type": "Point",
      "coordinates": [13.655, 52.77]
    }
  },
  {
    "type": "Feature",
    "properties": { "name": "Berlin L1 P7" },
    "geometry": {
      "type": "Point",
      "coordinates": [13.655, 52.27]
    }
  },
```

```
    {
      "type": "Feature",
      "properties": { "name": "Berlin L1 P8" },
      "geometry": {
        "type": "Point",
        "coordinates": [13.155, 52.77]
      }
    },
    {
      "type": "Feature",
      "properties": { "name": "Berlin L1 P9" },
      "geometry": {
        "type": "Point",
        "coordinates": [13.155, 52.27]
      }
    },
    {
      "type": "Feature",
      "properties": { "name": "Berlin L2 Hull E" },
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [13.565, 52.43],
            [13.565, 52.61],
            [13.745000000000001, 52.61],
            [13.745000000000001, 52.43],
            [13.565, 52.43]
          ]
        ]
      }
    },
    {
      "type": "Feature",
      "properties": { "name": "Berlin L2 Hull W" },
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [12.975, 52.43],
            [12.975, 52.61],
            [13.155, 52.61],
            [13.155, 52.43],
            [12.975, 52.43]
          ]
        ]
      }
    },
    {
      "type": "Feature",
```

```
    "properties": { "name": "Berlin L2 Hull N" },
    "geometry": {
      "type": "Polygon",
      "coordinates": [
        [
          [13.315, 52.88],
          [13.315, 53.06],
          [13.495, 53.06],
          [13.495, 52.88],
          [13.315, 52.88]
        ]
      ]
    }
  },
  {
    "type": "Feature",
    "properties": { "name": "Berlin L2 Hull S" },
    "geometry": {
      "type": "Polygon",
      "coordinates": [
        [
          [13.315, 52.18],
          [13.315, 52.36],
          [13.495, 52.36],
          [13.495, 52.18],
          [13.315, 52.18]
        ]
      ]
    }
  },
  {
    "type": "Feature",
    "properties": { "name": "Berlin Spine" },
    "geometry": {
      "type": "LineString",
      "coordinates": [
        [13.155, 52.52],
        [13.405, 52.77],
        [13.655, 52.52]
      ]
    }
  },
  {
    "type": "Feature",
    "properties": { "name": "Berlin Diagonal" },
    "geometry": {
      "type": "LineString",
      "coordinates": [
        [12.605, 51.72],
        [14.205, 53.32]
      ]
```

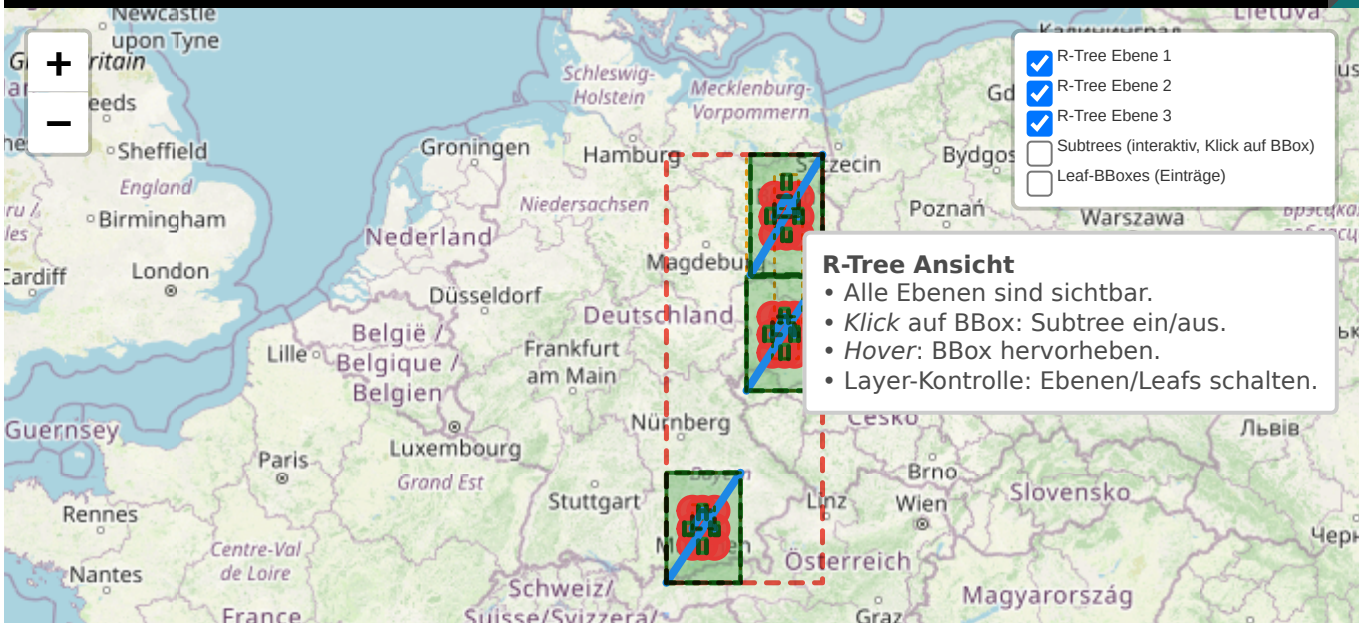
```
    }
  },

  {
    "type": "Feature",
    "properties": { "name": "München Area" },
    "geometry": {
      "type": "Polygon",
      "coordinates": [
        [
          [10.7761, 47.3374],
          [10.7761, 48.9374],
          [12.376100000000001, 48.9374],
          [12.376100000000001, 47.3374],
          [10.7761, 47.3374]
        ]
      ]
    }
  },
  {
    "type": "Feature",
    "properties": { "name": "München L1 P1" },
    "geometry": {
      "type": "Point",
      "coordinates": [11.5761, 48.1374]
    }
  },
  {
    "type": "Feature",
    "properties": { "name": "München L1 P2" },
    "geometry": {
      "type": "Point",
      "coordinates": [11.8261, 48.1374]
    }
  },
  {
    "type": "Feature",
    "properties": { "name": "München L1 P3" },
    "geometry": {
      "type": "Point",
      "coordinates": [11.326100000000001, 48.1374]
    }
  },
  {
    "type": "Feature",
    "properties": { "name": "München L1 P4" },
    "geometry": {
      "type": "Point",
      "coordinates": [11.5761, 48.3874]
    }
  },
```

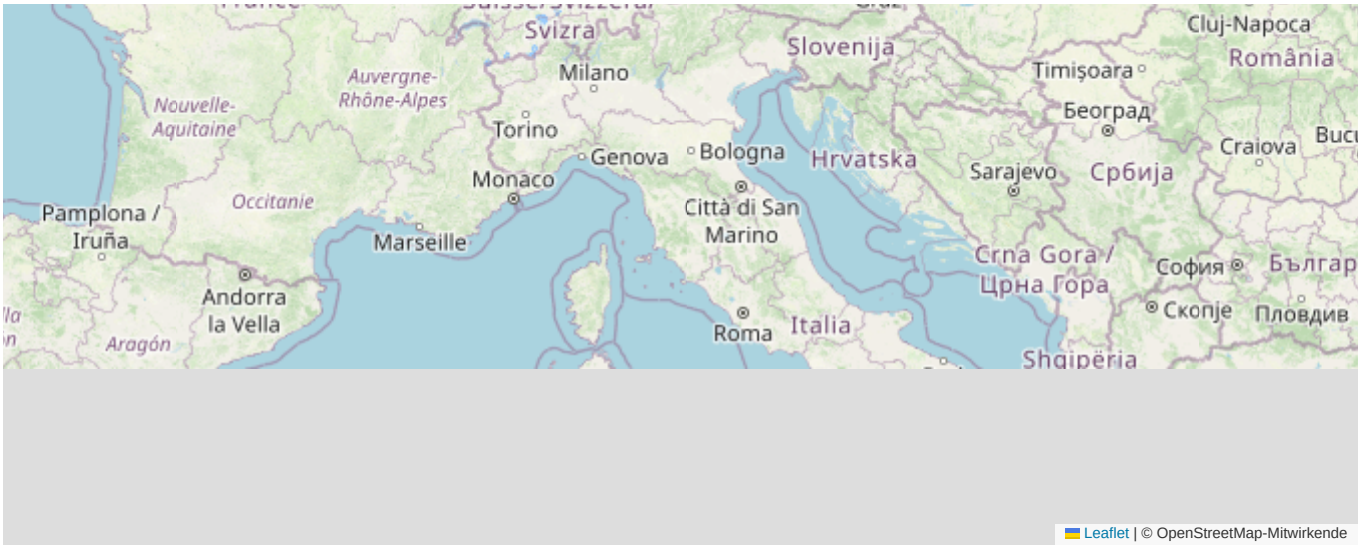
```
    {
      "type": "Feature",
      "properties": { "name": "München L1 P5" },
      "geometry": {
        "type": "Point",
        "coordinates": [11.5761, 47.887399999999996]
      }
    },
    {
      "type": "Feature",
      "properties": { "name": "München L1 P6" },
      "geometry": {
        "type": "Point",
        "coordinates": [11.8261, 48.3874]
      }
    },
    {
      "type": "Feature",
      "properties": { "name": "München L1 P7" },
      "geometry": {
        "type": "Point",
        "coordinates": [11.8261, 47.887399999999996]
      }
    },
    {
      "type": "Feature",
      "properties": { "name": "München L1 P8" },
      "geometry": {
        "type": "Point",
        "coordinates": [11.326100000000001, 48.3874]
      }
    },
    {
      "type": "Feature",
      "properties": { "name": "München L1 P9" },
      "geometry": {
        "type": "Point",
        "coordinates": [11.326100000000001, 47.887399999999996]
      }
    },
    {
      "type": "Feature",
      "properties": { "name": "München L2 Hull E" },
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [11.7361, 48.047399999999996],
            [11.7361, 48.227399999999996],
            [11.916100000000001, 48.227399999999996],
```

```
            [11.916100000000001, 48.047399999999996],
            [11.7361, 48.047399999999996]
          ]
        ]
      }
    },
    {
      "type": "Feature",
      "properties": { "name": "München L2 Hull W" },
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [11.146100000000001, 48.047399999999996],
            [11.146100000000001, 48.227399999999996],
            [11.326100000000001, 48.227399999999996],
            [11.326100000000001, 48.047399999999996],
            [11.146100000000001, 48.047399999999996]
          ]
        ]
      }
    },
    {
      "type": "Feature",
      "properties": { "name": "München L2 Hull N" },
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [11.4861, 48.2974],
            [11.4861, 48.4774],
            [11.6661, 48.4774],
            [11.6661, 48.2974],
            [11.4861, 48.2974]
          ]
        ]
      }
    },
    {
      "type": "Feature",
      "properties": { "name": "München L2 Hull S" },
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [11.4861, 47.797399999999996],
            [11.4861, 47.977399999999996],
            [11.6661, 47.977399999999996],
            [11.6661, 47.797399999999996],
            [11.4861, 47.797399999999996]
          ]
```

```
      ]
    }
  },
  {
    "type": "Feature",
    "properties": { "name": "München Spine" },
    "geometry": {
      "type": "LineString",
      "coordinates": [
        [11.326100000000001, 48.1374],
        [11.5761, 48.3874],
        [11.8261, 48.1374]
      ]
    }
  },
  {
    "type": "Feature",
    "properties": { "name": "München Diagonal" },
    "geometry": {
      "type": "LineString",
      "coordinates": [
        [10.7761, 47.3374],
        [12.376100000000001, 48.9374]
      ]
    }
  }
]
}
```


```
Einträge: 48
Höhe: 2
Ebenen-Stats (Tiefe -> {nodes, leaves}): [null,{"nodes":1,"leaves":0},
{"nodes":6,"leaves":48},{"nodes":48,"leaves":0}]
```





Block 5: Schema-Evolution & Versionierung

Ein großer Vorteil von Document Stores: Schema-Flexibilität. Aber Flexibilität bedeutet nicht Chaos. Wie gehen Sie mit Dokumenten um, die unterschiedliche Strukturen haben? Wie migrieren Sie von Version 1 zu Version 2?

Schema-Evolution: Das Problem

Szenario:

In langlebigen Anwendungen entwickeln sich die Datenstrukturen zwangsläufig weiter. Neue Features, geänderte Anforderungen oder Refactoring führen dazu, dass Dokumente unterschiedliche Versionen und Felder besitzen. Das folgende Beispiel zeigt, wie ein User-Profil von Version 1 zu Version 2 wächst.

Ihre App speichert User-Profile:

```
// Version 1 (Launch):
{
  _id: 'user_001',
  name: 'Alice',
  email: 'alice@example.com'
}

// Version 2 (nach 6 Monaten - neue Features):
{
  _id: 'user_042',
  name: 'Bob',
  email: 'bob@example.com',
  preferences: {
    theme: 'dark',
    language: 'de'
  },
  subscriptionTier: 'premium'
}
```

Problem:

Die Herausforderung: Alte Dokumente haben keine neuen Felder wie preferences oder subscriptionTier. Der Code muss mit beiden Versionen umgehen können, und Queries müssen unterschiedliche Strukturen berücksichtigen. Ohne eine Strategie entstehen Inkonsistenzen und schwer wartbarer Code.

- Alte Dokumente haben keine `preferences` oder `subscriptionTier`
- Code muss mit beiden Versionen umgehen
- Queries müssen unterschiedliche Strukturen berücksichtigen

Ohne Strategie:

Ohne ein klares Migrationskonzept wird der Code schnell unübersichtlich: Viele `if (doc.preferences)`-Checks, inkonsistente Datenqualität und schwierige Analysen sind die Folge. Die Wartbarkeit leidet und Fehler schleichen sich ein.

- ❌ Code voller `if (doc.preferences)` Checks
- ❌ Inkonsistente Datenqualität
- ❌ Schwierige Analyse (manche Felder fehlen)

Schema-Evolution ist unvermeidlich in langlebigen Systemen. Die Frage ist nicht ob, sondern wie Sie damit umgehen. Document Stores erlauben Evolution, aber erzwingen sie nicht – Sie müssen aktiv managen.

Migration-Patterns

Pattern 1: Lazy Migration (on-read)

Das Lazy-Migration-Pattern ist eine elegante Lösung für die Evolution von Dokumentstrukturen in langlebigen Anwendungen. Anstatt alle Daten auf einmal zu migrieren, werden Dokumente erst dann angepasst, wenn sie tatsächlich gelesen werden. So bleibt das System jederzeit verfügbar und es entsteht keine Downtime.

Im Beispiel sehen Sie, wie beim Lesen eines User-Dokuments geprüft wird, ob die Struktur veraltet ist. Fehlen neue Felder wie `preferences` oder `subscriptionTier`, werden sie direkt ergänzt und die Version hochgesetzt. Die Migration erfolgt also „on demand“ – nur für die Dokumente, die wirklich gebraucht werden.

```
async function getUser(id) {
  const doc = await db.get(id);

  // Migriere, falls alte Version
  if (!doc.schemaVersion || doc.schemaVersion < 2) {
    doc.preferences = doc.preferences || { theme: 'light', language: 'en' }
    doc.subscriptionTier = doc.subscriptionTier || 'free';
    doc.schemaVersion = 2;

    // Speichere migrierte Version
    await db.put(doc);
  }
```

```
  }

  return doc;
}
```

Vorteile:

Die Vorteile liegen auf der Hand: Es gibt keine Unterbrechung des Betriebs, da keine große Batch-Migration nötig ist. Die Daten werden schrittweise aktualisiert, sobald sie im Alltag verwendet werden. Das ist besonders praktisch für Systeme mit vielen selten genutzten Dokumenten.

- ✅ Keine Downtime (keine Batch-Migration)
- ✅ Dokumente werden nur bei Bedarf migriert

Nachteile:

Allerdings hat das Pattern auch Nachteile: Der Code muss weiterhin mit alten und neuen Versionen umgehen können, was die Komplexität erhöht. Außerdem existieren nach wie vor gemischte Strukturen in der Datenbank, bis alle Dokumente einmal gelesen und migriert wurden. Queries müssen also beide Varianten berücksichtigen.

- ❌ Code muss weiterhin alte Versionen verstehen
- ❌ Queries sehen gemischte Strukturen

Pattern 2: Eager Migration (batch)

Das Eager-Migration-Pattern verfolgt einen anderen Ansatz als die Lazy Migration: Hier werden alle Dokumente in einem Rutsch auf die neue Struktur gebracht. Die Migration erfolgt als Batch-Prozess – typischerweise vor einem Major-Release oder bei grundlegenden Schema-Änderungen.

Im Beispiel sehen Sie, wie alle User-Dokumente aus der Datenbank geladen und geprüft werden. Fehlen neue Felder oder ist die Version veraltet, werden die Dokumente direkt angepasst und gesammelt. Anschließend werden alle migrierten Dokumente auf einmal gespeichert. So entsteht eine konsistente Datenbasis.

```
async function migrateAllUsers() {
  const result = await db.allDocs({ include_docs: true });

  const migrations = result.rows
    .filter(row => !row.doc.schemaVersion || row.doc.schemaVersion < 2)
    .map(row => {
      const doc = row.doc;
      doc.preferences = doc.preferences || { theme: 'light', language: 'en'
      doc.subscriptionTier = doc.subscriptionTier || 'free';
      doc.schemaVersion = 2;
      return doc;
    });

  await db.bulkDocs(migrations);
  console.log(`Migrated ${migrations.length} documents`);
}
```

```
}
```

Vorteile:

Die Vorteile sind klar: Nach der Migration sind alle Dokumente auf dem neuesten Stand, und der Code muss sich nicht mehr um alte Versionen kümmern. Das erleichtert die Wartung und sorgt für einheitliche Datenstrukturen.

- ✅ Konsistente Datenstruktur nach Migration
- ✅ Code kann alte Versionen vergessen

Nachteile:

Allerdings hat die Eager Migration auch Nachteile: Bei großen Datenmengen ist ein Wartungsfenster nötig, da alle Dokumente geändert werden. Die Revision-History wächst, und während der Migration kann das System kurzzeitig nicht verfügbar sein. Für kritische Systeme ist daher eine sorgfältige Planung erforderlich.

- ❌ Erfordert Wartungsfenster bei großen Datenmengen
- ❌ Alle Dokumente werden geändert (Revision-History wächst)

In der Praxis kombinieren Sie oft beide: Lazy Migration für graduelle Änderungen, Eager Migration für Breaking Changes vor Major-Releases. Wichtig: Versionsnummern im Dokument (`schemaVersion` Feld) machen Migrationen nachvollziehbar.

Schema-Validierung (optional)

PouchDB unterstützt keine native Validierung – aber Sie können es in der App-Schicht implementieren:

Im Gegensatz zu MongoDB bietet PouchDB keine eingebaute Schema-Validierung. Das bedeutet: Sie müssen die Datenintegrität selbst sicherstellen – typischerweise in der App-Schicht, bevor Sie ein Dokument speichern.

Im Beispiel sehen Sie, wie mit der Bibliothek Zod ein Schema für Produktdokumente definiert wird. Das Schema legt fest, welche Felder vorhanden sein müssen und welche Typen sie haben. So können Sie schon vor dem Speichern prüfen, ob die Daten den Anforderungen entsprechen.

Die Funktion `saveProduct` validiert das Produkt gegen das Schema. Schlägt die Validierung fehl, wird das Dokument nicht gespeichert und ein Fehler ausgegeben. So vermeiden Sie fehlerhafte oder inkonsistente Daten in Ihrer Datenbank.

```
1  const db = new PouchDB('lecture_demo');
2
3  // Zod-Schema für Produkt
4  const productSchema = z.object({
5    _id: z.string().min(1),
6    name: z.string().min(1),
7    category: z.string().min(1),
8    brand: z.string().min(1),
9    price: z.number().min(0),
10   stock: z.number().int().min(0),
11   rating: z.number().int().min(1).max(5).optional()
```

```
  rating: z.number().int().min(1).max(5).optional(),
  created_at: z.string().optional() // ISO-Datum als String
});

// Validierung vor Save
async function saveProduct(product) {
  const result = productSchema.safeParse(product);
  if (!result.success) {
    console.error(`Validation failed: ${JSON.stringify(result.error.i
      null, 2)}`);
    return; // Stop saving if validation fails
  }
  await db.put(product);
  console.log('Product saved:', product._id);
}

saveProduct({
  _id: 'P02001',
  name: 'New Gadget',
  category: 'Electronics',
  brand: 'TechCorp',
  stock: 50
});
```

```
Validation failed: [
  {
    "code": "invalid_type",
    "expected": "number",
    "received": "undefined",
    "path": [
      "price"
    ],
    "message": "Required"
  }
]
```

Trade-off:

Die Validierung mit einem expliziten Schema bringt klare Vorteile: Sie definieren einen Vertrag für Ihre Daten und erkennen Fehler frühzeitig – noch bevor sie in die Datenbank gelangen. Allerdings entsteht zusätzlicher Aufwand: Sie müssen das Schema pflegen, und bei jeder Änderung der Datenstruktur ist ein Update im Code nötig. Das ist der Preis für Datenqualität und Wartbarkeit.

- ✅ Explizite Kontrakte
- ✅ Frühe Fehlerkennung
- ❌ Mehr Boilerplate
- ❌ Schema-Evolution erfordert Code-Updates

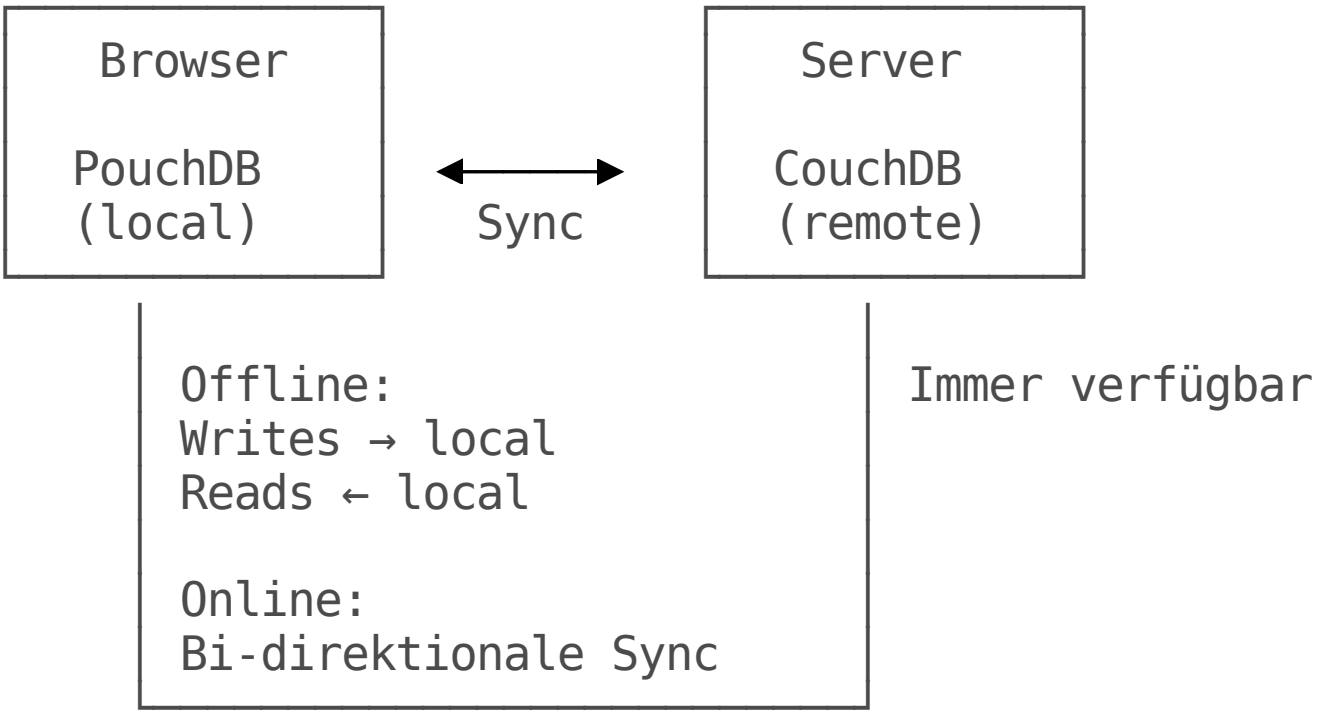
Block 6: Offline-First & Synchronisation

Jetzt kommen wir zu einem der mächtigsten Features von PouchDB: Offline-First Architektur. Ihre App funktioniert ohne Netzwerk, Daten werden lokal gespeichert, und sobald Verbindung besteht, synchronisiert alles automatisch. Klingt magisch – aber es gibt Tücken.

Offline-First Architektur

Konzept:

Das Offline-First-Konzept ist ein Paradigmenwechsel in der Webentwicklung: Ihre Anwendung funktioniert auch ohne Internetverbindung, weil alle Daten lokal im Browser gespeichert werden. Sobald wieder eine Verbindung besteht, synchronisiert PouchDB die Änderungen automatisch mit dem Server.



Die ASCII-Grafik zeigt das Prinzip: Im Offline-Modus werden alle Lese- und Schreiboperationen lokal ausgeführt. Im Online-Modus erfolgt eine bidirektionale Synchronisation – Änderungen werden hochgeladen und neue Daten vom Server übernommen. So bleibt die App jederzeit nutzbar und konsistent.

Vorteile:

Die Vorteile sind enorm: Nutzer können überall arbeiten, ohne auf Netzabdeckung zu achten. Schreibvorgänge sind sofort verfügbar, und die Synchronisation läuft im Hintergrund. Typische Einsatzszenarien sind mobile Apps mit instabiler Verbindung, kollaborative Tools wie Google Docs und Anwendungen für den Außendienst.

- ✅ App funktioniert ohne Internet
- ✅ Instant Writes (keine Latenz)
- ✅ Automatische Sync bei Reconnect

Use Cases:

Offline-First-Datenbanken wie PouchDB sind besonders wertvoll in Szenarien, in denen eine stabile Internetverbindung nicht garantiert ist. Mobile Apps profitieren von lokaler Speicherung und Synchronisation, kollaborative Tools ermöglichen gemeinsames Arbeiten auch bei Verbindungsabbrüchen, und Außendienst-Anwendungen erlauben Technikern, Daten direkt vor Ort zu erfassen – unabhängig vom Netz.

- Mobile Apps (flaky connections)
- Collaborative Tools (Google Docs-Style)
- Field Service Apps (Techniker ohne Netz)

Offline-First ist kein Nischen-Feature mehr – es ist Best Practice für moderne Web-Apps. Nutzer erwarten, dass Apps funktionieren, egal ob im Flugzeug oder im Funkloch. PouchDB macht das trivial – zumindest auf der Happy-Path.

Sync Setup: PouchDB ↔ CouchDB

Einmalige Sync (Pull):

Der einmalige Pull-Sync ist der einfachste Weg, um Daten vom Server in die lokale Datenbank zu holen. In diesem Fall simulieren wir den Server durch unsere lokale PouchDB-Instanz, es könnte aber auch eine URL zum remote Server angegeben werden, mit dem synchronisiert wird. Nach der Replikation stehen alle Dokumente sofort „offline“ zur Verfügung. Das ist ideal für den initialen Datenabgleich oder das Auffrischen der lokalen Kopie.

```
1  const remoteDB = new PouchDB('lecture_demo'); // "http://server-url
     /lecture_demo"
2  const localDB = new PouchDB('my_local_db');
3
4  // Daten vom Server holen
5  await localDB.replicate.from(remoteDB);
6
7  console.log(JSON.stringify(await localDB.get("P00001") , null, 2));
8
9  console.debug("Daten vom Server geladen");
```

```
{
  "name": "Laptop 14\" 2.0",
  "category": "Electronics",
  "brand": "PixelPeak",
  "price": 1399.03,
  "stock": 138,
  "rating": 3.7,
  "created_at": "2025-06-14",
  "_id": "P00001",
  "_rev": "1-9f04c28afe20e1591157eab6468badeb"
}
Daten vom Server geladen
```

Einmalige Sync (Push):

Im Beispiel für den einmaligen Sync (Push) wird die Synchronisation mit einer Remote-Datenbank nur simuliert – tatsächlich läuft alles lokal im Browser. Das Remote-Objekt ist eine weitere PouchDB-Instanz, die wie ein Server agiert. Sie können die Änderungen direkt in der Browser-Konsole unter „Application → IndexedDB“ beobachten: Nach dem Push sind die aktualisierten Daten sowohl in der lokalen als auch in der simulierten Remote-Datenbank sichtbar. So lässt sich das Sync-Verhalten nachvollziehen, ohne einen echten Server zu benötigen.

```
const remoteDB = new PouchDB('lecture_demo');
const localDB = new PouchDB('my_local_db');

const product = await localDB.get('P00001');

console.log('Lokales Dokument:', product);

await localDB.put({
  ...product,
  stock: product.stock - 1  // Simuliere Update
});

await localDB.replicate.to(remoteDB);

console.debug("Lokale Änderungen zum Server gepusht");
```

```
Lokales Dokument: {"name":"Laptop 14\"
2.0","category":"Electronics","brand":"PixelPeak","price":1399.03,"stock":
06-14","_id":"P00001","_rev":"1-9f04c28afe20e1591157eab6468badeb"}
Lokale Änderungen zum Server gepusht
```

Bidirektionale Live-Sync:

Der bidirektionale Live-Sync ist das Herzstück der Offline-First-Architektur: Änderungen werden in Echtzeit zwischen lokaler und „Remote"-Datenbank ausgetauscht. Im Beispiel wird der Server nur simuliert – tatsächlich laufen beide Datenbanken im Browser. Sie können die Synchronisation und alle Updates direkt in der Browser-Konsole unter „Application → IndexedDB" beobachten. Die Events zeigen, wann Daten synchronisiert, pausiert oder Fehler aufgetreten sind. So lässt sich das Verhalten einer echten Client-Server-Synchronisation nachvollziehen, ohne einen externen Server zu benötigen.

```
const remoteDB = new PouchDB('lecture_demo');
const localDB = new PouchDB('my_local_db');

// Kontinuierliche Synchronisation in beide Richtungen
const sync = localDB.sync(remoteDB, {
  live: true,           // Bleibt offen, hört auf Änderungen
  retry: true           // Reconnect bei Verbindungsabbruch
});

// Events
sync.on('change', info => {
  console.log('Sync change:', info);
});

sync.on('error', err => {
  console.error('Sync error:', err);
});

sync.on('active', () => {
```

```
  console.log('Sync resumed');
});

sync.on('paused', err => {
  console.log('Sync paused', err);
});

const product = await localDB.get('P00001');
console.log('Lokales Dokument:', product);
await localDB.put({
  ...product,
  stock: product.stock + 11  // Simuliere Update
});

```

```
Lokales Dokument: {"name":"Laptop 14\"
2.0","category":"Electronics","brand":"PixelPeak","price":1399.03,"stock":
06-14","_id":"P00001","_rev":"2-ac74dfd5ff26fa2381e3242ee51d6175"}
Sync resumed
Sync change: {"direction":"pull","change":
{"ok":true,"start_time":"2026-02-
13T10:14:26.871Z","docs_read":2,"docs_written":2,"doc_write_failures":0,"e
[],"last_seq":1004,"docs":[{"name":"Laptop 14\"
2.0","category":"Electronics","brand":"PixelPeak","price":1399.03,"stock":
06-14","_id":"P00001","_rev":"3-
57053fd7b73c1e732cf26308a23a1fcf","_revisions":{"start":3,"ids":
["57053fd7b73c1e732cf26308a23a1fcf","ac74dfd5ff26fa2381e3242ee51d6175","9f
{"name":"Laptop 14\"
2.0","category":"Electronics","brand":"PixelPeak","price":1399.03,"stock":
06-14","_id":"P00001","_rev":"2-
04cbd07b5dbddfca8049f63ffdd6c7ad","_revisions":{"start":2,"ids":
["04cbd07b5dbddfca8049f63ffdd6c7ad","9f04c28afe20e1591157eab6468badeb"]}}]
Sync change: {"direction":"push","change":
{"ok":true,"start_time":"2026-02-
13T10:14:26.871Z","docs_read":1,"docs_written":1,"doc_write_failures":0,"e
[],"last_seq":1003,"docs":[{"name":"Laptop 14\"
2.0","category":"Electronics","brand":"PixelPeak","price":1399.03,"stock":
06-14","_id":"P00001","_rev":"3-
bcba59458ab5a8a45a5eb8b4fee3fd0f","_revisions":{"start":3,"ids":
["bcba59458ab5a8a45a5eb8b4fee3fd0f","ac74dfd5ff26fa2381e3242ee51d6175","9f
Sync paused undefined
```

Live-Sync ist der Kern von Offline-First: Lokale Änderungen werden automatisch hochgeladen, Remote-Änderungen heruntergeladen. Das funktioniert asynchron – Ihre App blockiert nie. Aber: Was passiert bei Konflikten?

Konfliktauflösung: Das Problem

Szenario:

Alice und Bob editieren dasselbe Dokument offline:

```
const remoteDB = new PouchDB('lecture_demo'); // "http://server-url/lecture_demo"
const Alice = new PouchDB('local_db_1', {adapter: 'memory'});
const Bob = new PouchDB('local_db_2', {adapter: 'memory'});

// Daten vom Server holen
await Alice.replicate.from(remoteDB);
await Bob.replicate.from(remoteDB);

const p1 = await Alice.get('P00001');
const p2 = await Bob.get('P00001');

await Alice.put({
  ...p1,
  stock: p1.stock - 5  // Alice verkauft 5 Einheiten
});

await Bob.put({
  ...p2,
  stock: p2.stock - 3  // Bob verkauft 3 Einheiten
});

await Alice.replicate.to(remoteDB);
await Bob.replicate.to(remoteDB);

// Beide syncen → KONFLIKT!
console.log(JSON.stringify(await remoteDB.get("P00001"), null, 2))
```


```
{
  "name": "Laptop 14\" 2.0",
  "category": "Electronics",
  "brand": "PixelPeak",
  "price": 1399.03,
  "stock": 134,
  "rating": 3.7,
  "created_at": "2025-06-14",
  "_id": "P00001",
  "_rev": "3-57053fd7b73c1e732cf26308a23a1fcf"
}
```


Wie entscheidet das System?

- ❌ **Nicht möglich:** Beide Versionen gleichzeitig richtig
- ⚠️ **Last-Write-Wins:** Einfach, aber Daten gehen verloren
- ✅ **CouchDB-Ansatz:** Deterministischer Gewinner + Conflict-Flag

CouchDB/PouchDB nutzen einen cleveren Ansatz: Das System wählt deterministisch einen „Gewinner“ (basierend auf Revisions-IDs), behält aber BEIDE Versionen. Ihre App sieht standardmäßig den Gewinner, kann aber Konflikte erkennen und manuell auflösen.

Konfliktauflösung: Implementierung

Konflikte erkennen:

Mit diesem Code können Sie gezielt prüfen, ob ein Dokument Konflikte besitzt. Die Option `{ conflicts: true }` sorgt dafür, dass PouchDB beim Abrufen des Dokuments auch die Liste der konkurrierenden Revisionen (`_conflicts`) zurückliefert. Im Beispiel werden die IDs der Konflikt-Revisionen ausgegeben – das sind die alternativen Versionen, die beim Sync entstanden sind. Sie können diese Revisionen einzeln abrufen und analysieren, bevor Sie eine Strategie zur Auflösung wählen. Besonders hilfreich: In der Browser-Konsole unter „Application → IndexedDB“ sehen Sie alle Revisionen und können den Zustand direkt nachvollziehen.

```
1  const db = new PouchDB('lecture_demo');
2
3  const doc = await db.get('P00001', { conflicts: true });
4
5  if (doc._conflicts) {
6    console.log('Konflikt erkannt!', doc._conflicts);
7    // doc._conflicts = ['2-bob', '2-alice']  // Verlierer-Revisions
8  }
```

```
Konflikt erkannt! ["3-57053fd7b73c1e732cf26308a23a1fcf","2-
04cbd07b5dbddfca8049f63ffdd6c7ad"]
```

Manuelle Auflösung:

In dieser Funktion zur manuellen Konfliktauflösung für Produktdokumente wird zunächst geprüft, ob Konflikte vorliegen. Falls ja, werden alle konkurrierenden Revisionen geladen und miteinander verglichen. Die Merge-Strategie im Beispiel wählt die Revision mit dem höchsten stock-Wert als Gewinner. Nach dem Zusammenführen wird das Ergebnis gespeichert und die unterlegenen Revisionen entfernt. So bleibt nur die konsolidierte Version erhalten und die Datenbank ist wieder konsistent. Sie können die Merge-Logik flexibel anpassen, etwa nach Zeitstempel, Nutzer oder anderen Feldern.

```
async function resolveProductConflict(id) {
  // Lade Produkt mit Konflikten
  const doc = await db.get(id, { conflicts: true });

  if (!doc._conflicts) {
    console.log("Kein Konflikt vorhanden.");
```

```
    return;
  }

  // Lade alle konfliktierenden Revisionen
  const conflicts = await Promise.all(
    doc._conflicts.map(rev => db.get(id, { rev }))
  );

  // Beispiel-Merge: Nutze die Revision mit dem höchsten 'stock'-Wert
  const merged = conflicts.reduce((best, current) => {
    return (current.stock > best.stock) ? current : best;
  }, doc);

  // Aktualisiere das Merge-Ergebnis (z.B. Zeitstempel)
  merged.lastModified = new Date().toISOString();
  delete merged._conflicts;

  // Speichere das gemergte Produkt
  await db.put(merged);

  // Lösche verlierende Revisionen
  await Promise.all(
    doc._conflicts.map(rev => db.remove(id, rev))
  );

  console.log(`Konflikt für Produkt ${id} gelöst. Gemergte Revision gespeic
    );
}
```

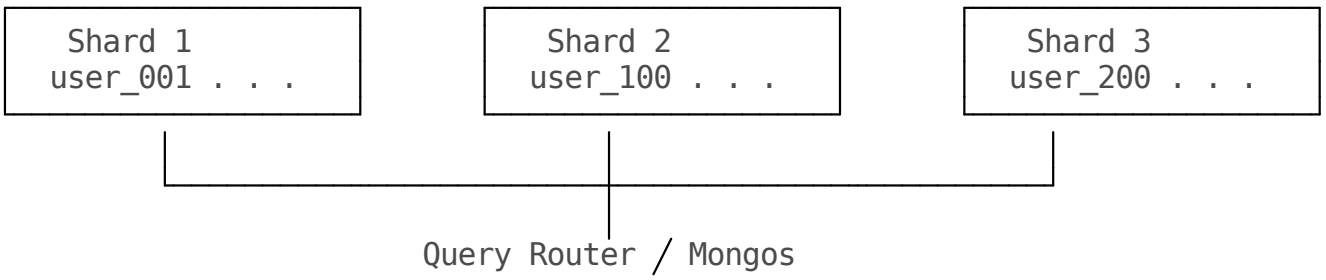
Konfliktauflösung ist komplex – es gibt keine universelle Lösung. Last-Write-Wins ist simpel, aber naiv. Operational Transformation (wie in Google Docs) ist mächtig, aber extrem komplex. CRDTs (Conflict-free Replicated Data Types) sind elegant, aber spezialisiert. Für die meisten Apps reicht: Konflikte erkennen, UI zeigen („Alice und Bob haben gleichzeitig editiert“), Nutzer entscheiden lassen.

Sharding & Skalierung in Document Stores

Document Stores wie MongoDB und CouchDB sind für horizontale Skalierung ausgelegt. Sharding bedeutet, dass große Datenmengen auf mehrere Server (Shards) verteilt werden. Jeder Shard speichert einen Teil der Dokumente – so können Reads und Writes parallelisiert und die Kapazität beliebig erhöht werden.

Was ist Sharding?

Wir erinnern uns: Sharding ist das Aufteilen einer großen Datenbank in kleinere, unabhängige Teile (Shards). Jeder Shard ist für einen bestimmten Bereich der Daten verantwortlich. Das ermöglicht es, die Last auf viele Server zu verteilen und große Datenmengen effizient zu verwalten.



Sharding-Key: Das Herzstück

Der Sharding-Key bestimmt, wie Dokumente auf die Shards verteilt werden. Typische Sharding-Keys sind User-IDs, Produkt-IDs oder Zeitstempel. Die Wahl des richtigen Keys ist entscheidend: Ein schlechter Sharding-Key kann zu „Hotspots“ führen, bei denen ein Shard überlastet ist, während andere kaum genutzt werden.

- **Gute Sharding-Keys:** Gleichmäßig verteilte Werte (z.B. zufällige UUIDs, Hashes)
- **Schlechte Sharding-Keys:** Monoton wachsende Werte (z.B. Zeitstempel, fortlaufende IDs)

```
// Beispiel: MongoDB Sharding-Key
{
  _id: "user_001",
  region: "EU",
  created_at: "2025-01-01"
}
// Sharding nach "region" oder Hash(_id)
```

Sharding-Strategien

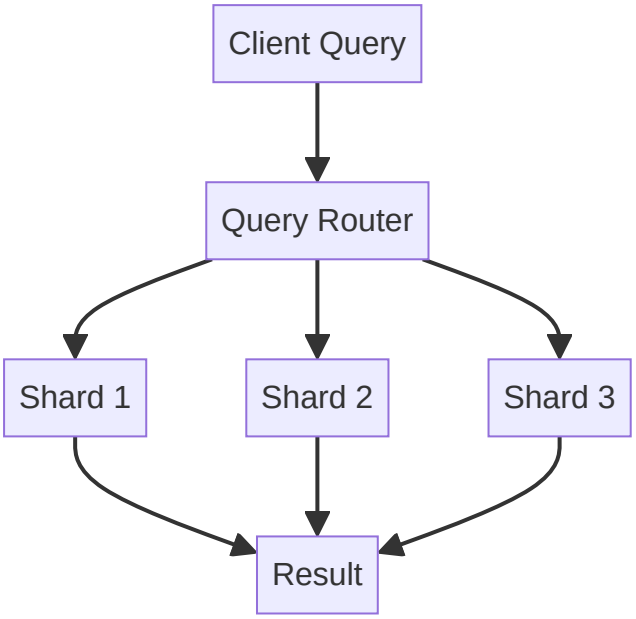
Document Stores bieten verschiedene Sharding-Strategien:

- **Range-Sharding:** Dokumente werden nach einem Bereich (z.B. alphabetisch, Zeit) verteilt.
- **Hash-Sharding:** Ein Hash des Sharding-Keys bestimmt den Shard – sorgt für gleichmäßige Verteilung.
- **Tag/Zone-Sharding:** Dokumente werden nach geografischen oder logischen Zonen verteilt (z.B. EU, US, ASIA).

Strategie	Vorteil	Nachteil
Range	Schnelle Bereichs-Queries	Gefahr von Hotspots
Hash	Gleichmäßige Verteilung	Bereichs-Queries langsam
Tag/Zone	Geo-nahe Datenhaltung	Komplexes Routing

Skalierung & Rebalancing

Wächst die Datenbank, können neue Shards hinzugefügt werden. Das System sollte die Daten automatisch neu verteilen (Rebalancing). Wichtig: Rebalancing kann zu temporären Performance-Einbußen führen. Monitoring und Planung sind essenziell.



Best Practices für Sharding-Keys

- Wähle einen Sharding-Key, der die Last gleichmäßig verteilt.
- Vermeide monoton wachsende Keys (z.B. Zeitstempel).
- Plane für Wachstum: Sharding-Keys sollten auch bei Millionen Dokumenten skalieren.
- Überwache regelmäßig die Verteilung und passe den Sharding-Key ggf. an.

Sharding ist mächtig, aber nicht trivial: Die Wahl des Sharding-Keys entscheidet über Performance und Skalierbarkeit. Document Stores bieten flexible Strategien – nutze sie bewusst!

Block 7: Use Cases & Abgrenzung

Abschließend: Wann sollten Sie Document Stores einsetzen – und wann nicht? Lassen Sie uns typische Szenarien durchgehen und Entscheidungskriterien entwickeln.

Typische Use Cases

Perfekt für Document Stores:

1. **Content Management Systeme**
    - Flexible Dokument-Strukturen (Posts, Pages, Comments)
    - Schema entwickelt sich mit Features
    - Beispiel: Blog-Plattformen, Wikis
2. **User Profiles & Session Data**
    - Heterogene Nutzer-Daten (nicht alle haben gleiche Felder)
    - Schnelle Reads/Writes
    - Beispiel: E-Commerce User Profiles
3. **Mobile Apps (Offline-First)**
    - Lokale Persistenz + Remote Sync
    - Flaky Connections
    - Beispiel: Notiz-Apps, Field Service Tools

Der gemeinsame Nenner: Flexible Schemas, hierarchische Daten, und Lesefokus mit gelegentlichen Writes. Document Stores glänzen, wenn Ihre Daten natürlich als JSON modelliert werden – nicht als Tabellen.

Wann NICHT Document Stores?

Besser Relational:

1. **Komplexe Transaktionen**
    - Bankgeschäfte, Bestellungen mit Inventory-Updates
    - Brauchen ACID-Garantien über mehrere Entities
    - Document Stores: Transaktionen nur pro Dokument
2. **Viele Joins über Entities**
    - Relationale Queries mit 5+ Tables
    - Document Stores: Joins sind teuer/unmöglich
    - Normalisierung ist besser
3. **Strikte Schema-Validierung**
    - Regulatorische Anforderungen (Finance)
    - Explizite Kontrakte über Teams
    - Document Stores: Validierung ist optional

Besser Key-Value:

1. **Pure Caching**
    - Session-Speicher, Rate-Limiting
    - Nur Key-Lookups, keine Queries
    - Document Stores sind Overkill

Besser Zeitreihen DB:

1. **Metriken & Logs**
    - Hohe Write-Throughput
    - Zeitbasierte Aggregationen
    - Document Stores: Keine optimierten Time-Series Queries

Besser Graph:

1. **Beziehungsintensive Queries**
    - Social Networks, Recommendations
    - Traversals („Freunde von Freunden“)
    - Document Stores: Graph-Queries ineffizient

Die Frage ist nie „Ist MongoDB besser als PostgreSQL?“, sondern „Welche Probleme habe ich, und welches Tool passt?“ Document Stores sind mächtig für flexible, hierarchische Daten – aber keine Allzweckwaffe.

Zusammenfassung & Reflexion

Fassen wir zusammen: Document Stores sind die natürliche Evolution von Key-Value Systemen für strukturierte Daten. Sie bieten Flexibilität ohne Chaos – wenn Sie bewusst mit Schemas, Indizes und Sync umgehen.

Kernerkenntnisse

1. **Document Stores = Strukturierte NoSQL**
    - JSON als First-Class Citizen
    - Sekundäre Indizes für flexible Queries
    - Schema-optional, aber kontrollierbar
2. **Mango-Queries = Deklarative Suche**
    - Operatoren: `$eq`, `$gt`, `$in`, `$and`, `$or`, …
    - Verschachtelte Felder & Arrays unterstützt
    - Indizes sind essentiell für Performance
3. **Offline-First = Killer-Feature**
    - PouchDB + CouchDB = Seamless Sync
    - Konflikte unvermeidlich bei Kollaboration
    - Manuelle Auflösung nötig für kritische Fälle
4. **Use Case abhängig:**
    - ✅ Flexible Schemas, hierarchische Daten, Offline-Apps
    - ❌ Komplexe Joins, strikte Transaktionen

Document Stores füllen eine wichtige Lücke zwischen simplen Key-Value Stores und rigiden relationalen Datenbanken. Sie sind nicht „besser“ oder „schlechter“ – sie lösen andere Probleme. Ihre Aufgabe als Entwickler: Erkennen Sie, welche Probleme Sie haben.

Ausblick: Session 4 – (Wide) Column Stores

In der nächsten Session wechseln wir die Perspektive: Bisher fokussierten wir auf Transaktionen und Reads einzelner Dokumente. Column Stores optimieren für eine völlig andere Workload: Aggregationen über Millionen Zeilen. Statt „Finde User 42“ fragen wir: „Was ist der Durchschnittspreis aller Produkte in Kategorie X?“

Session 4 Preview: Column Stores

Fokus:

- **Spaltenorientierte Speicherung** (vs. Zeilen in KV/Document)
- **Kompression** (warum Column Stores 10x weniger Platz brauchen)
- **Analytics-Queries** (SUM, AVG, GROUP BY über große Datasets)
- **OLAP vs. OLTP** (wann welches Paradigma?)

Demo-System:

- DuckDB (SQL-basiert, läuft im Browser via WebAssembly)
- Parquet-Format (Column-Storage)

Lernziel:

> Verstehen Sie den Trade-off: Column Stores sind brillant für Analytics, aber langsam für Punkt-Abfragen

Referenzen & Vertiefung

Für alle, die tiefer einsteigen möchten – hier die wichtigsten Ressourcen zu Document Stores, PouchDB und verwandten Themen.

Offizielle Dokumentation

- **PouchDB Guides:** https://pouchdb.com/guides/
- **PouchDB Find Plugin (Mango):** https://pouchdb.com/guides/mango-queries.html
- **CouchDB Mango Query Reference:** https://docs.couchdb.org/en/stable/api/database/find.html
- **CouchDB Replication Protocol:** https://docs.couchdb.org/en/stable/replication/protocol.html

Konzeptuelle Ressourcen

- **JSON Schema Specification:** https://json-schema.org/
- **IndexedDB API (Browser):** https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API
- **MongoDB Query Language** (Vergleich): https://www.mongodb.com/docs/manual/tutorial/query-documents/
- **Offline-First Principles:** https://offlinefirst.org/

Bücher

- **„CouchDB: The Definitive Guide“** (Anderson, Lehnardt, Slater)
- **„Designing Data-Intensive Applications“** (Kleppmann) – Kapitel 5: Replication

Verwandte Technologien

- **MongoDB:** Document Store mit großem Ecosystem
- **Firebase/Firestore:** Google's Document Store (Cloud-hosted)
- **RxDB:** Reactive Database auf PouchDB-Basis

Anhang: Praktische Code-Snippets

Zum Abschluss: Eine Sammlung wiederverwendbarer Snippets für Ihre eigenen Projekte.

Setup-Boilerplate

```
// PouchDB mit allen Plugins
import PouchDB from 'pouchdb';
import PouchDBFind from 'pouchdb-find';

PouchDB.plugin(PouchDBFind);

const db = new PouchDB('my_app_db');

// Remote-Sync konfigurieren
const remoteDB = new PouchDB('https://myserver.com/db', {
  auth: {
    username: 'user',
    password: 'pass'
  }
});

// Bidirektionale Live-Sync
const sync = db.sync(remoteDB, {
  live: true,
  retry: true
});

sync.on('change', info => console.log('Synced:', info));
sync.on('error', err => console.error('Sync error:', err));
```

CRUD-Operationen

```
// Create
await db.put({
  _id: 'user_001',
  name: 'Alice',
  email: 'alice@example.com'
});

// Read
const user = await db.get('user_001');

// Update (benötigt _rev!)
user.email = 'newemail@example.com';
await db.put(user);

// Delete
await db.remove(user);

// Bulk Insert
await db.bulkDocs([
  { _id: 'doc1', data: 'a' },
  {  id: 'doc2', data: 'b' }
```

```
]);
```

Query-Pattern

```
// Index erstellen (nur einmal nötig)
await db.createIndex({
  index: { fields: ['category', 'price'] }
});

// Query ausführen
const result = await db.find({
  selector: {
    category: 'Electronics',
    price: { $gt: 100, $lt: 500 }
  },
  sort: [{ price: 'asc' }],
  limit: 10
});

console.log(result.docs);
```

Error-Handling

```
try {
  await db.put(doc);
} catch (err) {
  if (err.status === 409) {
    // Conflict (Document wurde zwischenzeitlich geändert)
    const latest = await db.get(doc._id);
    // Merge und retry
  } else if (err.status === 404) {
    // Document nicht gefunden
  } else {
    // Anderer Fehler
    throw err;
  }
}
```