

Inhaltsverzeichnis: Dokumentenorientierte DB

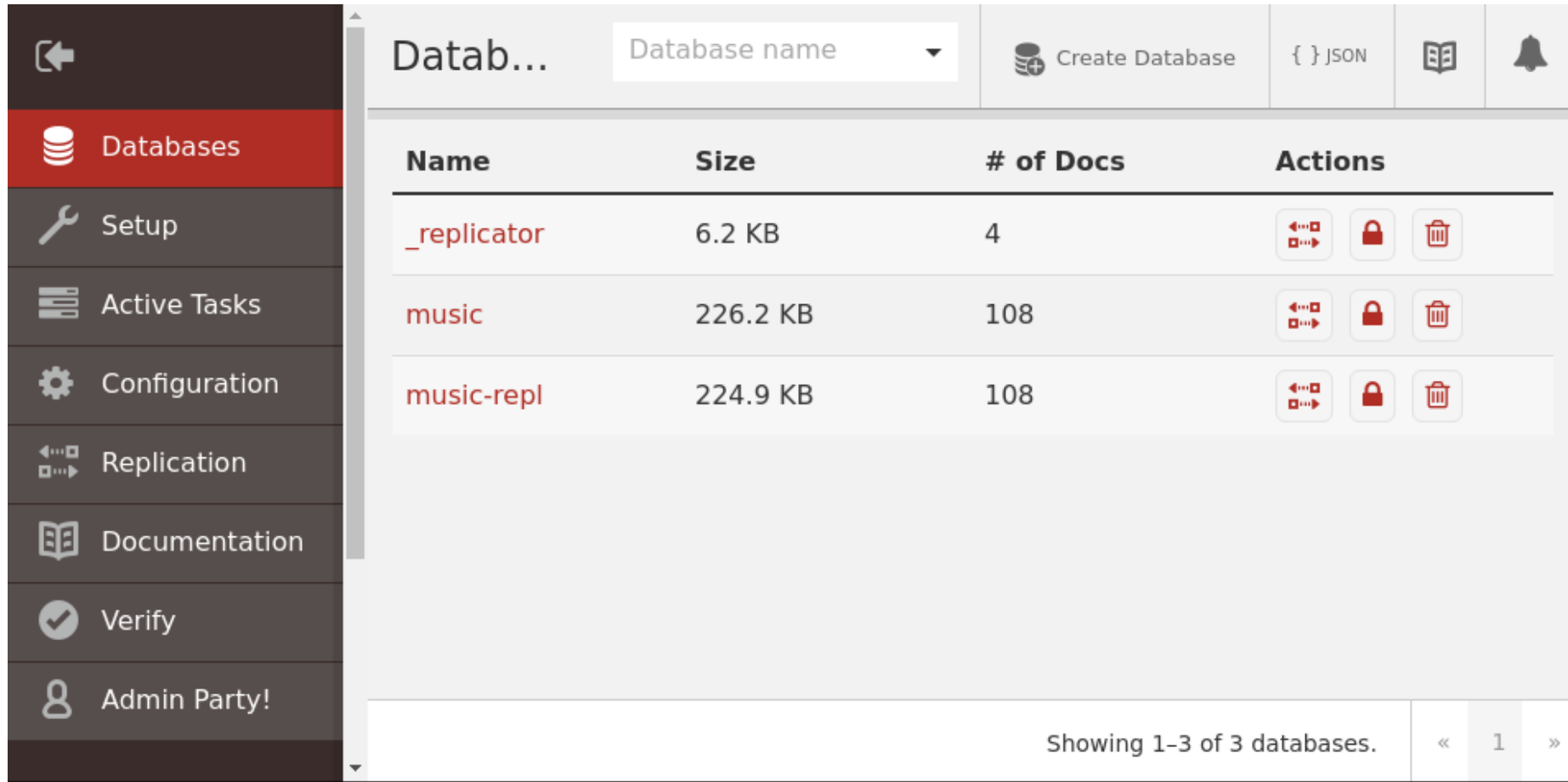
- **Einführung**
- **MongoDB**
 - Einfache Anfragen
 - Aggregation Framework und MapReduce
 - Indexierung und Anfrageoptimierung
 - Replikation und Sharding
- **CouchDB**
 - Demo: Anfragen und Views
 - Speicherstruktur: B-Bäume
 - Replikation und Konfliktlösung
- **Zusammenfassung**

- First Release: 2005
- Breites Anwendungsfeld: Von Smartphone bis Rechenzentrum
- Dokumente werden als JSON-Objekte repräsentiert
- Zugriff via RESTful HTTP-Requests (PUT, GET, ...)
- Komplexe Anfragen über JavaScript und MapReduce
- **Robust/fehlertolerant und hoch verfügbar**
 - Implementiert in Erlang
 - Append-only Speicherung über B-Bäume
 - Replikation: Multi-Master-Architektur
- Eventually consistent: Multi-Version Concurrency Control (MVCC)










Quelle [CouchDB]

CouchDB: Demo

Web interface: Fauxton



The screenshot displays the CouchDB Fauxton web interface. On the left is a dark sidebar with navigation options: Databases (highlighted), Setup, Active Tasks, Configuration, Replication, Documentation, Verify, and Admin Party!. The main content area shows a header with 'Datab...', a 'Database name' dropdown, a 'Create Database' button, and icons for JSON, documentation, and notifications. Below the header is a table with columns 'Name', 'Size', '# of Docs', and 'Actions'. The table lists three databases: '_replicator' (6.2 KB, 4 docs), 'music' (226.2 KB, 108 docs), and 'music-repl' (224.9 KB, 108 docs). Each database row has three action icons: replication, lock, and delete. At the bottom right, a pagination bar shows 'Showing 1-3 of 3 databases.' and a page number '1'.

Name	Size	# of Docs	Actions
_replicator	6.2 KB	4	  
music	226.2 KB	108	  
music-repl	224.9 KB	108	  

CouchDB: Anfragen

- Create:

```
curl -i -X POST "http://localhost:5984/music/" \  
-H "Content-Type: application/json" \  
-d '{"_id": "wings", "name": "Wings" }'
```
- Retrieve:

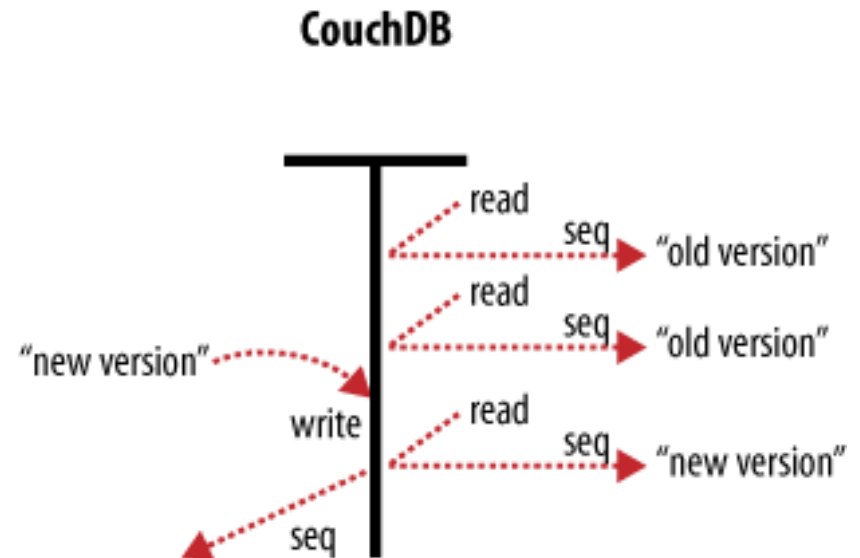
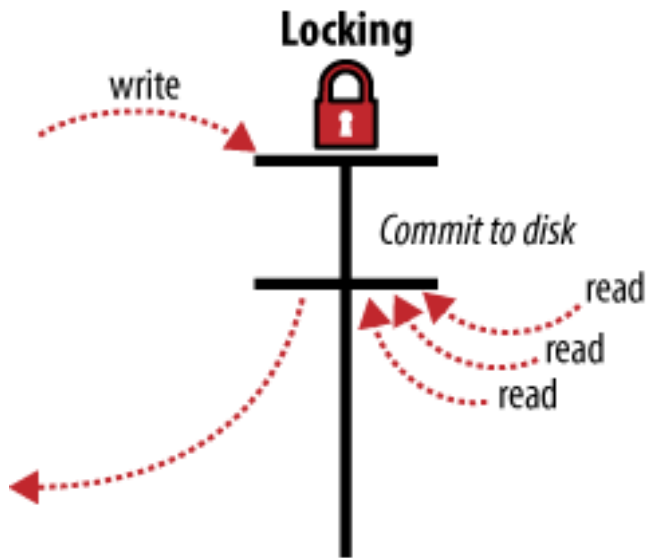
```
curl http://localhost:5984/music/wings
```
- Update:

```
curl -i -X PUT \  
"http://localhost:5984/music/wings" \  
-H "Content-Type: application/json" \  
-d '{"_id": "wings",  
      "_rev": "1-2fe1dd1911153eb9d",  
      "name": "Wings",  
      "albums": ["Wild Life", "Band on the Run"]  
}'
```
- Delete:

```
curl -i -X DELETE \  
"http://localhost:5984/music/wings" \  
-H "If-Match: 2-17e4ce41cd33d"
```

Multi-Version Concurrency Control

- Locking
 - Erst nach abgeschlossenem Schreiben können Clients lesen
 - Strikte Konsistenz
- MVCC
 - Schreibzugriffe blockieren Lesezugriffe nicht
 - Eventual Consistency



Quelle: <http://guide.couchdb.org/editions/1/en/consistency.html>

CouchDB: Views

- Allgemeine Form des Datenzugriffs und -indexierung
- Realisiert über MapReduce: Geordnete Liste von Schlüssel-Wert-Paaren

- Map-Funktion (via Fauxton):

- JavaScript-Funktion
- emit()

```
function(doc) {  
  if ('name' in doc) {  
    emit(doc.name, doc._id);  
  }  
}
```

- Abfrage des View

```
curl http://localhost:5984/music/_design/artists/_view/by_name
```

- URL-Parameters: key, limit, startkey, endkey, descending

```
curl 'http://localhost:5984/music/_design/artists/_view/by_name?key="Bleacher"'
```

```
curl 'http://localhost:5984/music/_design/artists/_view/by_name?limit=5'
```

```
curl 'http://localhost:5984/music/_design/artists/_view/by_name?startkey="C"'
```

CouchDB: Views

- Reduce-Funktion:

```
function (keys, values, rereduce){  
    return sum(values); }  

```

- *Keys*: Array aus Schlüssel-ID-Paaren (Schlüssel ist konstant)

- *Values*: Array mit zugehörigen Werten

- *Rereduce* = *false*: normale Reduce-Ausführung

- *Rereduce* = *true*: Erneuter rekursiver Aufruf der Reduce-Funktion

- *Keys*: Leeres Array
- *Values*: Ergebnisse einer anderen Reduce-Ausführung
- Benötigt z.B. bei Berechnung der Anzahl der Werte

```
function(keys, values, rereduce) {  
    if (rereduce) {  
        return sum(values);  
    } else {  
        return values.length;  
    }  
}
```

- Vorteile

- Aufteilung der Mapper-Ausgaben mit gleichen Schlüssel auf mehrere Reducer
- (Speicher-)Effiziente Wiederverwendung von vorberechneten Reduce-Ergebnissen

Inhaltsverzeichnis: Dokumentenorientierte DB

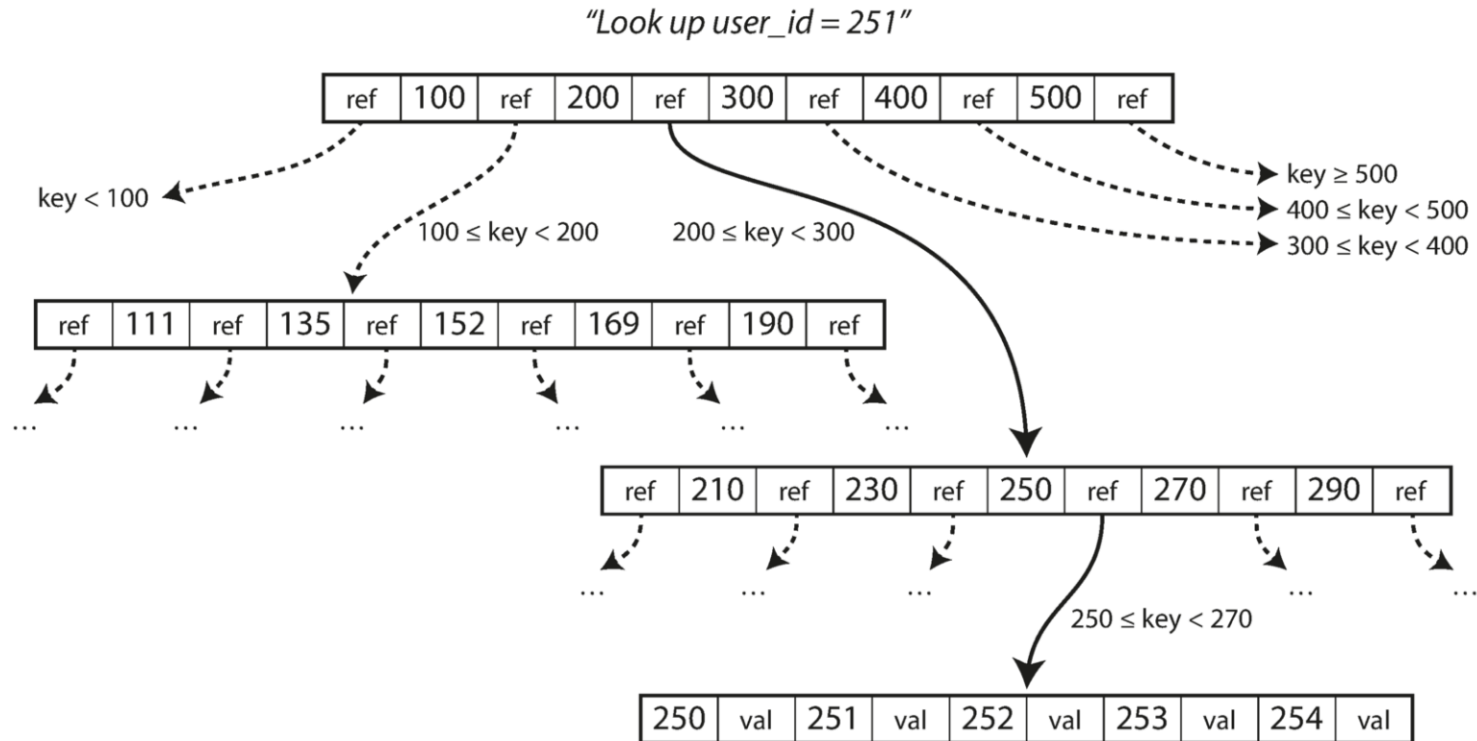
- **Einführung**
- **MongoDB**
 - Einfache Anfragen
 - Aggregation Framework und MapReduce
 - Indexierung und Anfrageoptimierung
 - Replikation und Sharding
- **CouchDB**
 - Demo: Anfragen und Views
 - **Speicherstruktur: B-Bäume**
 - Replikation und Konfliktlösung
- **Zusammenfassung**

CouchDB: Speicherstruktur

- Änderungsoperationen sind **logische Einfügeoperationen**, d.h. Daten werden nicht überschrieben
- Jedes Dokument hat statische ID und dynamische Revisions-ID
 - Änderung: neues Dokument mit alter ID und neuer Revision-ID
 - Löschen: neues Dokument mit alter ID, neuer Revision-ID und Flag „Deleted“
- Vorteile:
 - Geringerer Einfluss technischer Fehler auf Datensicherheit
 - Weniger Festplattenzugriffe beim Schreiben
- Einfügen stets durch Anfügen von Daten in zwei Dateien
 - Datenbank-File = Dokumente in Einfügereihenfolge
 - Index-File = B-Baum zur effizienten Suche
- CouchDB verwendet einen „B-Tree Storage Engine“ für alle Anfragen
 - Anfragen über Dokumenten-Schlüssel
 - Komplexe Anfragen (=Views) über MapReduce

Wiederholung: B-Baum

- Index mit sortierten Schlüsseln und Baumstruktur



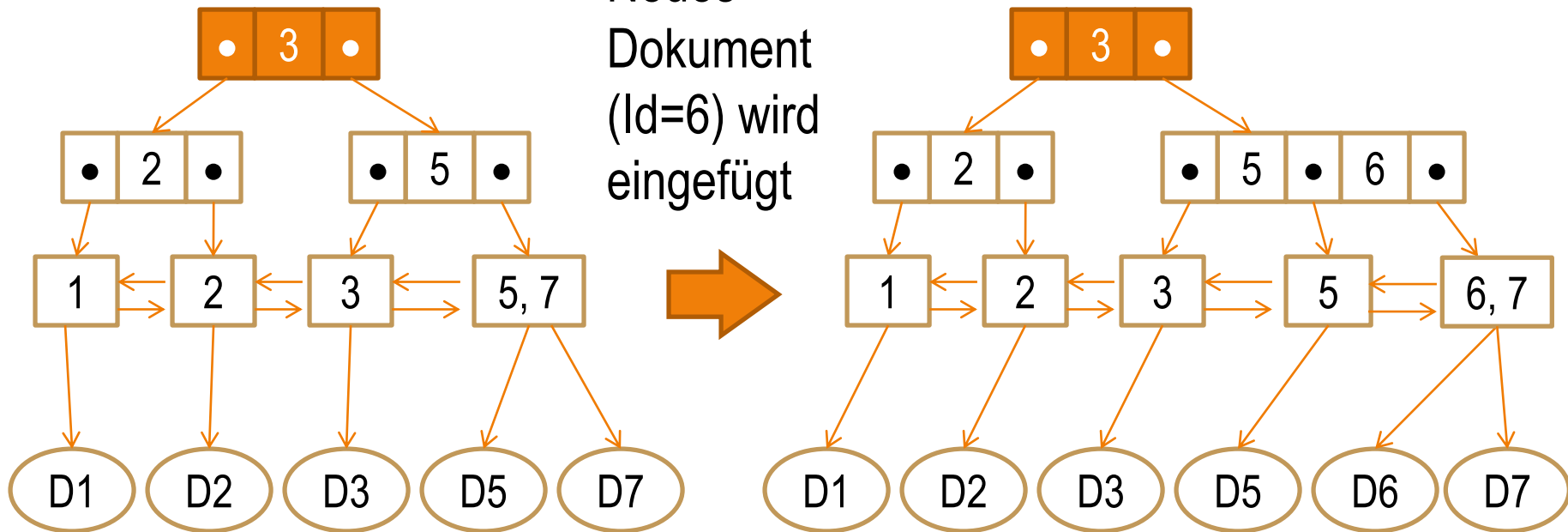
Quelle: <https://medium.com/@shashankbaravani/database-storage-engines-under-the-hood-705418dc0e35>

- Ausbalanciert bezüglich der Höhe
- Einfügen, Suchen (auch Bereiche) und Löschen in logarithmischer Zeit
- Aktualisierungen erfordern Laden und Anpassen des gesamten Blocks

CouchDB: Speicherstruktur

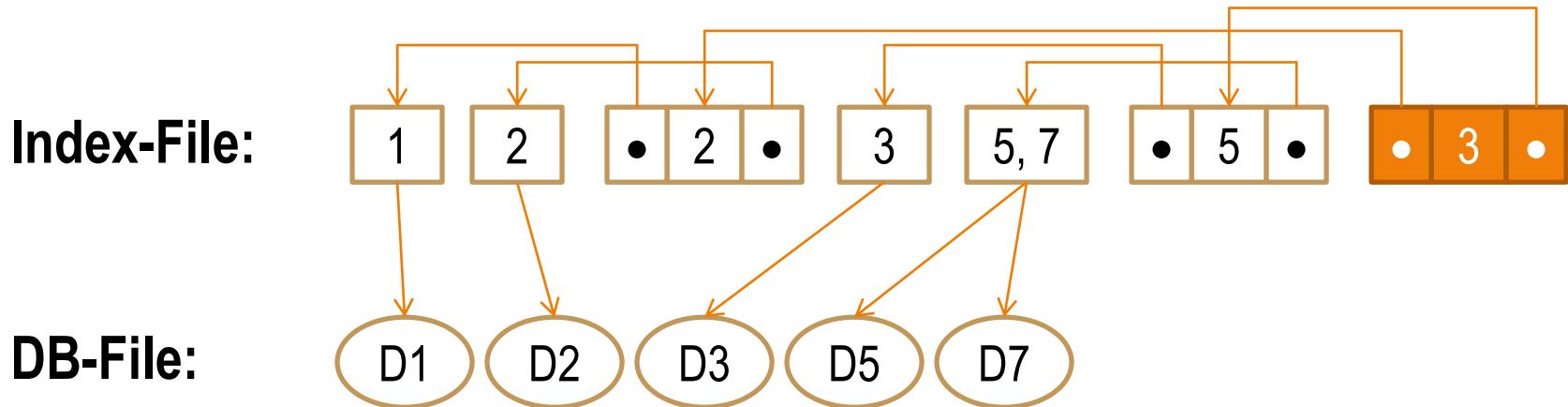
- Zugriff der Dokumente über serialisierten B-Baum
- Blattknoten: Doppel verlinkte Liste und Verweise auf Daten
- Beispiel: B-Baum der Größe 2
 - Alle Knoten haben max. 2 Einträge
 - Zwischenknoten somit max. 3 Kinder

Neues
Dokument
(Id=6) wird
eingefügt



CouchDB: Speicherstruktur

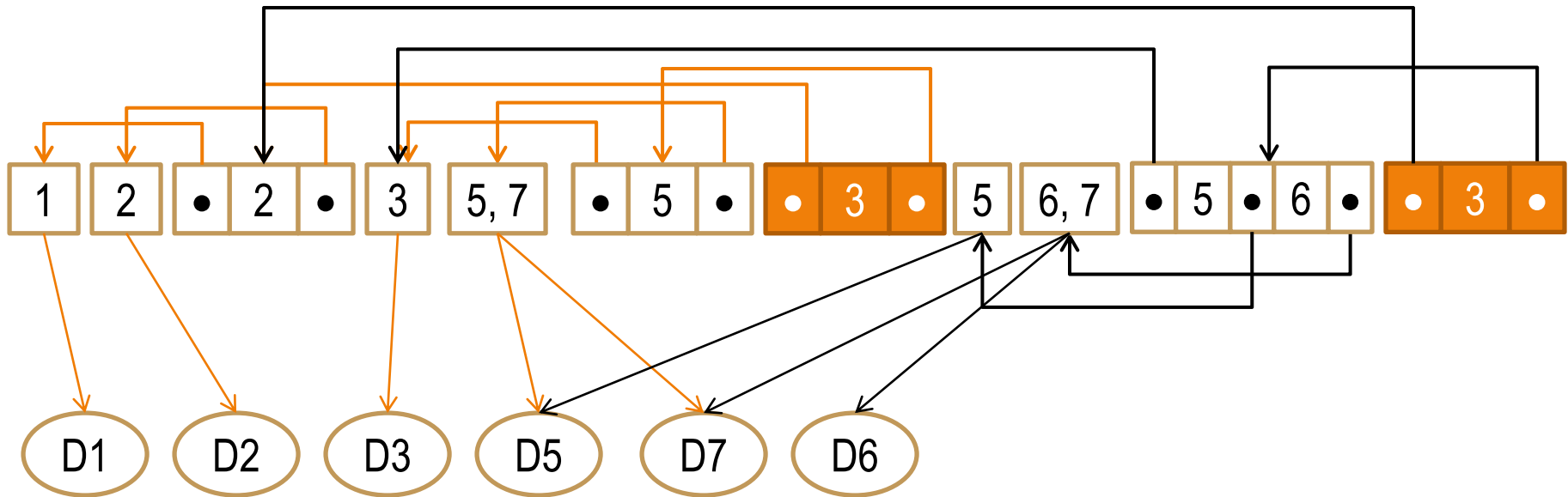
Serialisierung in zwei Dateien:



- Lesen des Index-File „von hinten“
- Beginne Suche ab erstem gefundenen Wurzelknoten

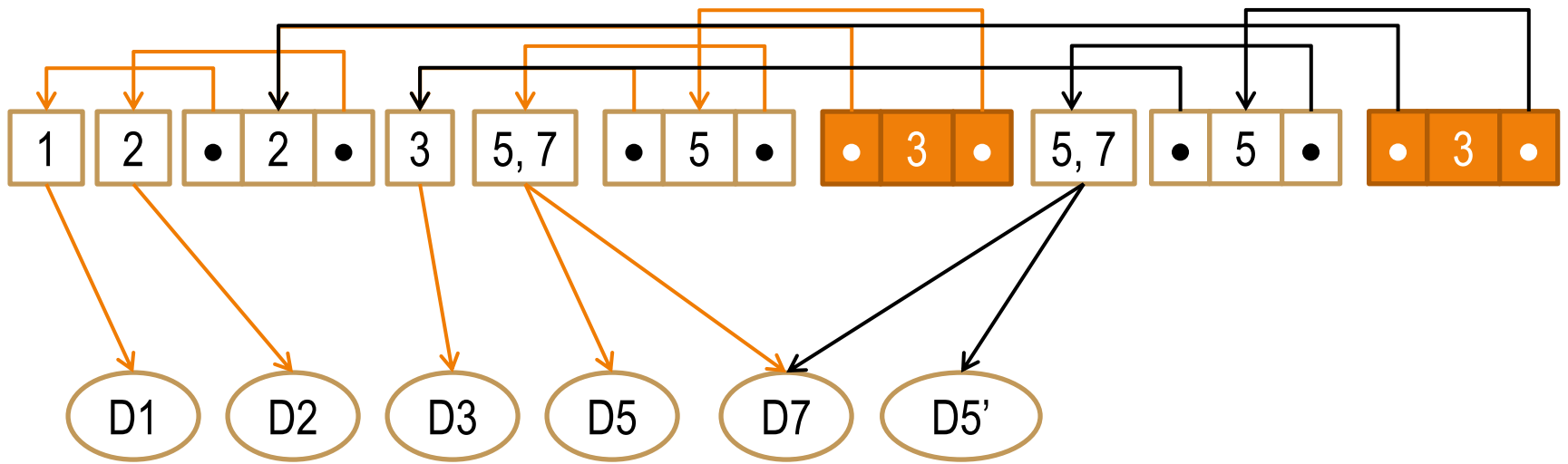
CouchDB: Einfügen von Dokumenten

Ausschließlich durch Anfügen



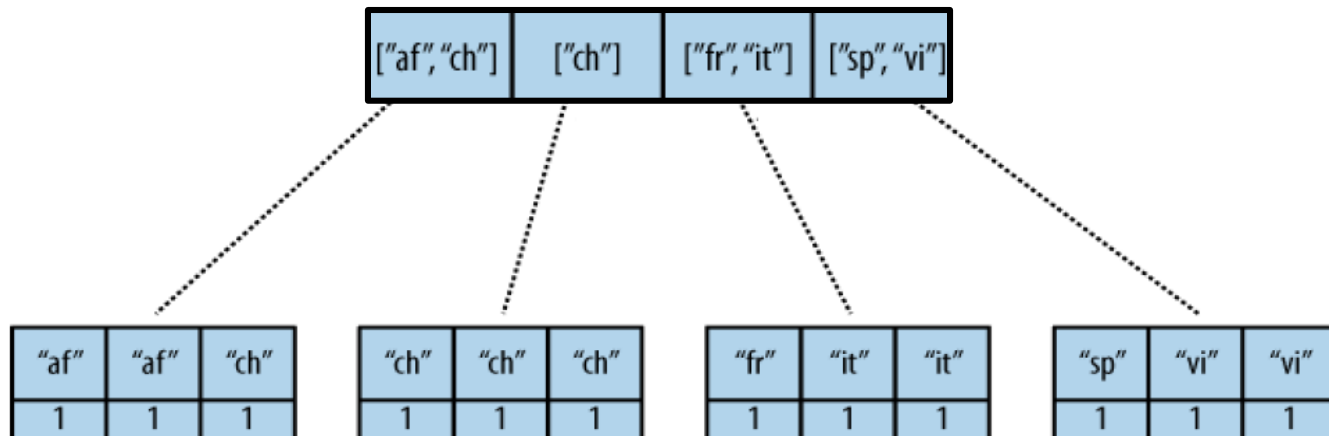
CouchDB: Aktualisierung von Dokumenten

- Keine direkte Aktualisierung des B-Baums
 - Einfügen einer neuen Dokumentenversion
 - Zeiger vom Blattknoten wird auf neue Dokumentenversion “umgebogen”
 - Einfügen eines leeren Dokuments beim Löschen
- **Beispiel:** Aktualisiere Dokument 5



CouchDB: Views

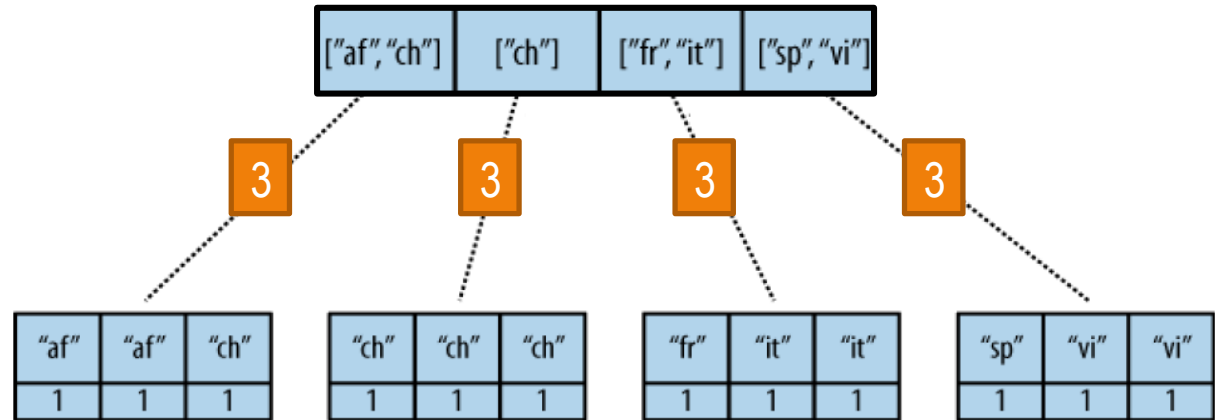
- Erstellung von Views über weiteren B-Baum
 - Anwenden der Map-Funktion auf alle Dokumente
 - Blatt-Knoten [Map-Output-Key, Map-Output-Value]
 - Sortiert nach Map-Output-Key
 - Innere Knoten mit Bereichen der Map-Output-Keys
- Aktualisierung des B-Baums bei Einfügen/Löschen/Ändern
 - Anwenden der Map-Funktion auf neue/geänderte Dokumente
 - Reorganisation des B-Baums



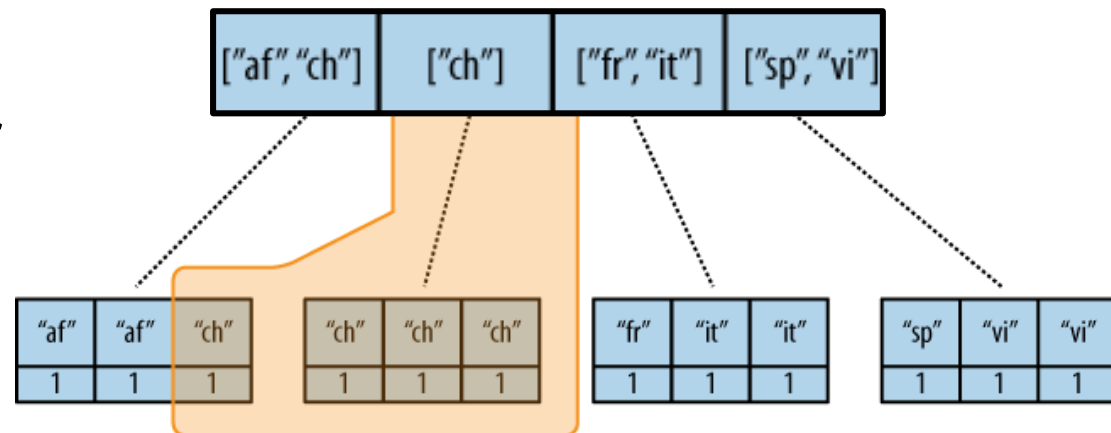
CouchDB: Reduce

```
function (keys, values, rereduce){  
  return sum(values);  
}
```

- Voraggregation
(*rereduce = false*):
Speicherung des
Reduce-Ergebnisses
pro Blattknoten



- Erst bei Bereichsabfragen
(z.B. nach „ch“), rekursive
Bottom-up-Vereinigung der
„Aggregate“ (über
rereduce = true)

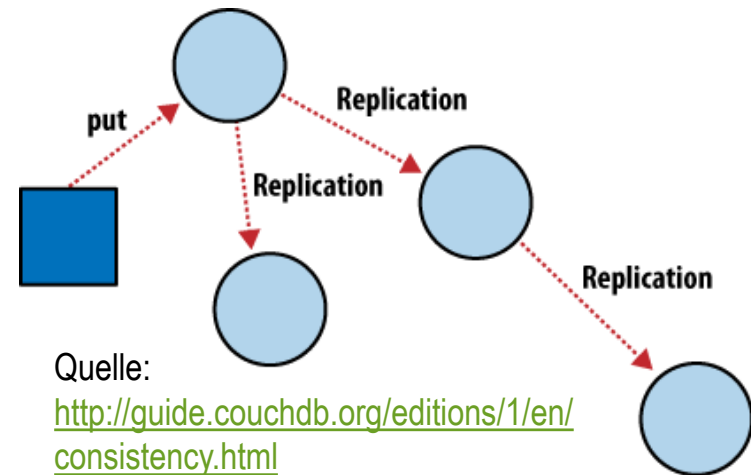


Inhaltsverzeichnis: Dokumentenorientierte DB

- **Einführung**
- **MongoDB**
 - Einfache Anfragen
 - Aggregation Framework und MapReduce
 - Indexierung und Anfrageoptimierung
 - Replikation und Sharding
- **CouchDB**
 - Demo: Anfragen und Views
 - Speicherstruktur: B-Bäume
 - **Replikation und Konfliktlösung**
- **Zusammenfassung**

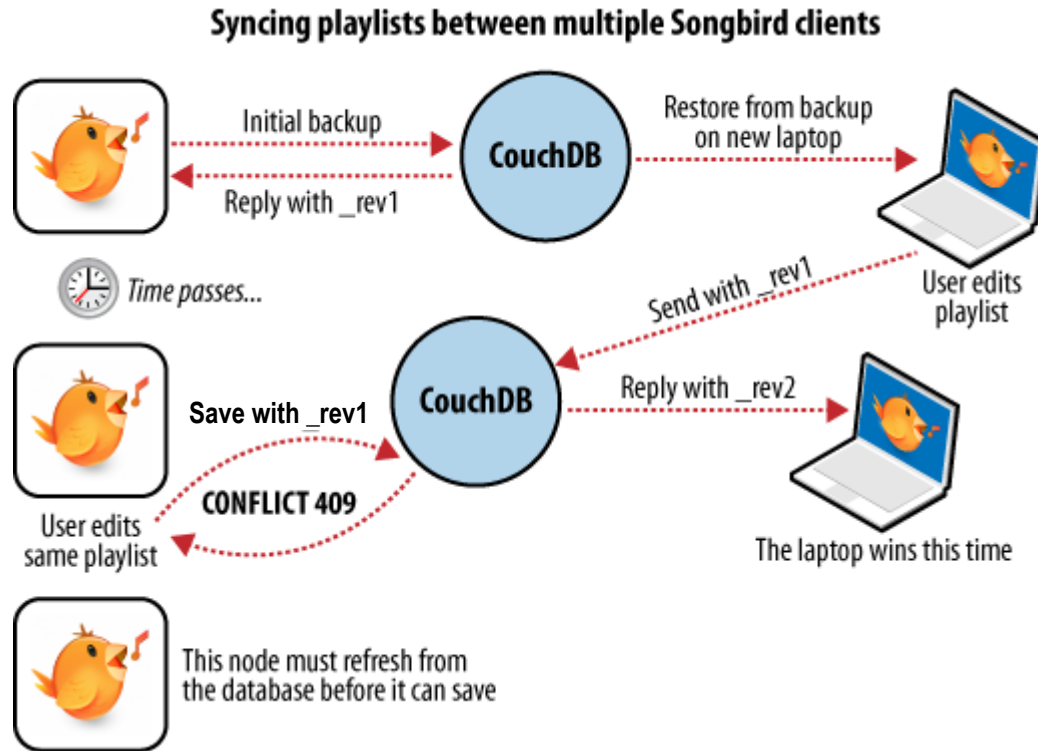
CouchDB: Replikation

- Multi-Master Replikation
- Keine ständige Verbindung zwischen den Servern erforderlich
- Inkrementelle Synchronisation des Datenbestandes zwischen Knoten
 - Manuell ausgelöst oder kontinuierlich
 - Revision-ID wird genutzt, um neue/geänderte Dokumente zu erkennen
- Durchführung
 - Index- und DB-File werden “von hinten abgearbeitet” und entsprechende Änderungsoperationen an anderen Knoten geschickt
 - Konflikt bei parallelen Änderungen am gleichen Dokument
 - Wahl der “winning version” durch deterministisches Verfahren
 - keine “Abstimmung” zwischen Knoten
 - “losing version” wird gespeichert
- Konfliktlösung durch Anwendung



CouchDB: Konfliktlösung

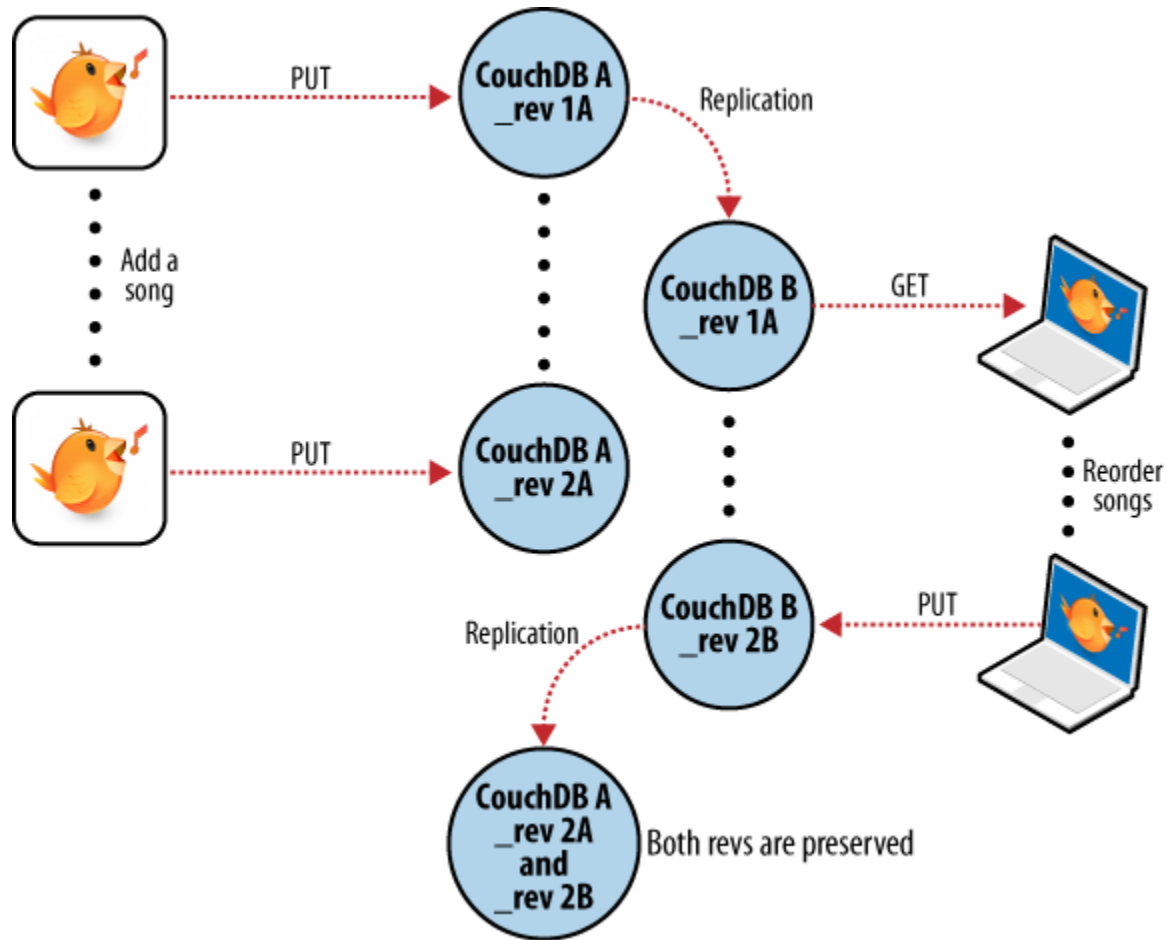
Ohne Replikation: Ablehnung von Aktualisierung mit alter Revisions-ID



Quelle: <http://guide.couchdb.org/editions/1/en/consistency.html>

CouchDB: Konfliktlösung

Konflikte bei Replikation



Quelle: <http://guide.couchdb.org/editions/1/en/consistency.html>

Inhaltsverzeichnis: Dokumentenorientierte DB

- **Einführung**
- **MongoDB**
 - Einfache Anfragen
 - Aggregation Framework und MapReduce
 - Indexierung und Anfrageoptimierung
 - Replikation und Sharding
- **CouchDB**
 - Demo: Anfragen und Views
 - Speicherstruktur: B-Bäume
 - Replikation und Konfliktlösung
- **Zusammenfassung**

Zusammenfassung: Dokumentenbasierte DB

- Key-Value Stores mit Dokumenten (JSON) als Werte
 - Effiziente Verarbeitung beliebig verschachtelter (semi-strukturierter) Dokumente
 - Einfache bis komplexe Anfragen über HTTP Rest oder spezielle Anfragesprachen
 - Schemafrei und Denormalisierung für schnellere Anfragebearbeitung
 - Inkonsistenzen müssen evtl. über Anwendung gelöst werden

	MongoDB	CouchDB
Stärken	<ul style="list-style-type: none">- Umfangreiche Daten und Anfragen- Relativ einfache Bedienbarkeit- Ähnlichkeit zwischen Anfragesprache und SQL- Umfangreiche Features	<ul style="list-style-type: none">- Robust/fehlertolerant- Hoch verfügbar- Web-kompatibel: HTTP und JSON- Einsatz: Smartphone bis Rechenzentrum
Schwächen	<ul style="list-style-type: none">- Anregung zur Denormalisierung- Aufwendiges Design und Administration- Einsatz: Rechenzentrum	<ul style="list-style-type: none">- Keine Ad-hoc Anfragen (Datenzugriff über Views und MapReduce)- Konfliktlösung durch Anwender

Literatur

- [MongoDB] The MongoDB Manual: <https://docs.mongodb.com/manual/>
- [CouchDB] CouchDB Documentation: <http://docs.couchdb.org/en/stable/>

- [Kval14] Christian Kvalheim: The Little Mongo DB Schema Design Book, 2014-2015
- [Cop13] Rick Copeland: MongoDB Applied Design Patterns. Practical Use Cases with the Leading NoSQL Database. O'Reilly Media, 2013
- [Har18] Guy Harrison: Studies in MongoDB Schema Design.
<https://medium.com/dbkoda/studies-in-mongodb-schema-design-pt-1-9ebef72ae7b9>
- [Hou14] Rick Houlihan: The Aggregation Framework,
<https://www.mongodb.com/presentations/aggregation-framework-0>
- [MapReduce] Dean & Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. <https://research.google.com/archive/mapreduce.html>