NOSQL MIT COUCHBASE

MODUL: WEB TECHNOLOGIES · PROF. KNEISEL
AUTOR: THOMAS REHM
DATUM: 13.03.15

ÜBERSICHT

DBMS Revolution

NoSQL Basics

Warum NoSQL?

Couchbase Server & Couchbase Mobile

Couchbase Konzepte & Techniken

Ausblick Übung

NOSQL REVOLUTION

Durch BigData, Big User & Cloud Computing neue Anforderung:

- **Skalierbarkeit & Performance**
- **Flexible Datenmodelle**
- **RDBMS schnell aufwändig & komplex**

NOSQL BASICS

- Bruch mit relationalem Ansatz
- schemaloses Datenmodel
- Andere Protokolle als SQL >> „Not only SQL“
- BASE (Basic Availability, Soft-State, Eventual Consistency) statt ACID (Atomic, Consistent, Isolated, Durable)

NOSQL ARTEN

Key-Value Datenbank

- **Einfachste Form der Datenspeicherung**
- **Zugriff nur über Key, keine Indizierung, keine Suche**
- **Extrem schnell**
- **Bsp.: Konfigurations-Dateien etc.**

(Riak, Dynamo…)

```
{key : value}
```

NOSQL ARTEN

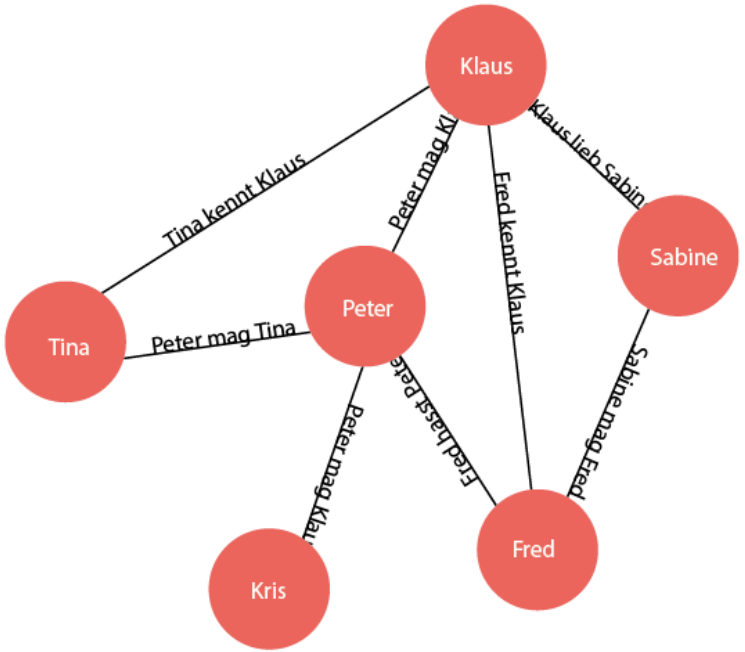
Graph Datenbank

- **Stark vernetzte Daten**
- **Graphen um Beziehungen darzustellen**

Bsp.:

```
Tina --kennt-> Klaus;
Peter --mag-> Tina;
```

(Google Knowledge Graph & Google Cayley, Neo4J…)



NOSQL ARTEN

Dokumenten-basierte Datenbank

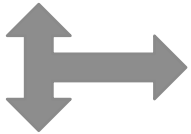
- **Erweitertes Key-Value-Model**
- **Strukturiertes Datenformat**
- **Speicherung der Daten in z.B. JSON**

(CouchDB, MongoDB, Couchbase…)

```
{
  "strings" : "simple strings",
  "numbers" : 43,
  "values" : true,
  "arrays" : ["one", "two", "three"],
  "objects" : {
          "objects" : "inside of objects",
          "work" : "like a charm"
        }
}
```

RELATIONAL TO JSON-DOC

ID	FIRST_NAME	LAST_NAME	EMAIL
10	Edward	Mason	edward@mason.com



ID	CONTACT_ID	STREET	ZIP	CITY
21	10	Wiesenhof 10	61169	Friedberg
22	10	Sonnenring 1	61118	Bad Vilbel

```
{
  "ID": 10,
  "FIRST_NAME": "Edward",
  "LAST_NAME": "Mason",
  "EMAIL": "edward@mason.com",
  "ADRESSES": [
  {
    "STREET": "Wiesenhof 10",
    "ZIP": 61169,
    "CITY": "Friedberg"
  },
  {
    "STREET": "Sonnenring 1",
    "ZIP": 61118,
    "CITY": "Bad Vilbel"
  }
  ]
}
```

WARUM & WOFÜR?

- Durch Skalierbarkeit rasant wachsenden Anforderungen gerecht werden
- Performance um Millionen von Anfragen zu handlen
- Durch flexiblere Datenmodele eine einfachere Datenmodellierung erreichen
- Online-Offline-Sync

Mögliche Einsatzbereiche:

- Logging
- Caching
- User Registration, Profile
- Apps mit z.B. Geodaten
- …

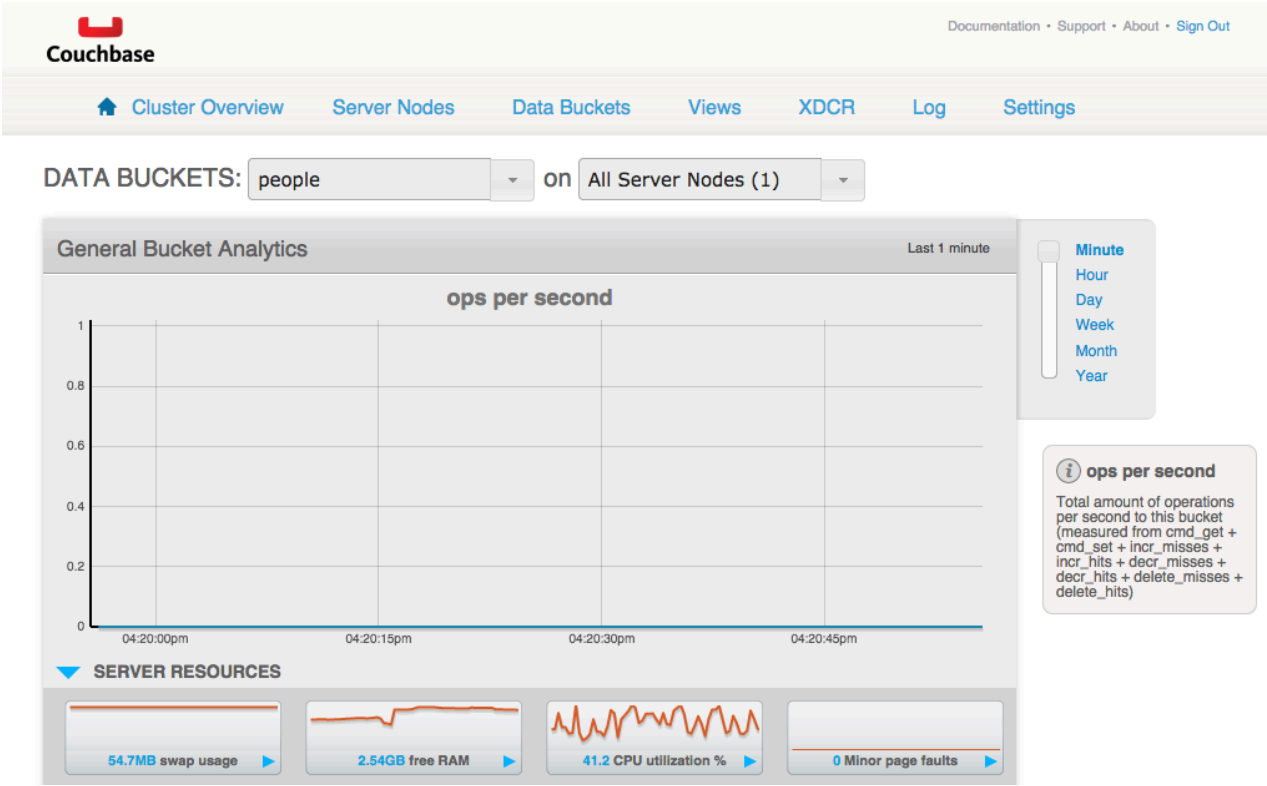
COUCHBASE!?

- OpenSource Projekt supported by Couchbase Inc.
- Geht aus CouchDB & Project Membase hervor
- Dokumenten-basierte NoSQL Datenbank
- Community & Enterprise Editions
- Couchbase Server
- Couchbase Mobile

COUCHBASE SERVER

- REST API oder Admin-Interface
- Nutzt HTTP als Kommunikations-Protokoll
- Mehrere CLI Tools für Managing und Monitoring des Clusters/Servers/Buckets
- Interface stellt vielfältige Statistiken bereit
- Am einfachsten erreichbar über Port 8091 (administrative Aufgaben) und Port 8092 (Views, Queries, etc.)
- SDKs für Java, PHP, C, Python, Ruby, .NET, Node.js, Go …
- Integrierte Techniken für Node Failovers

SERVER INTERFACE



COUCHBASE SERVER DATA STORAGE

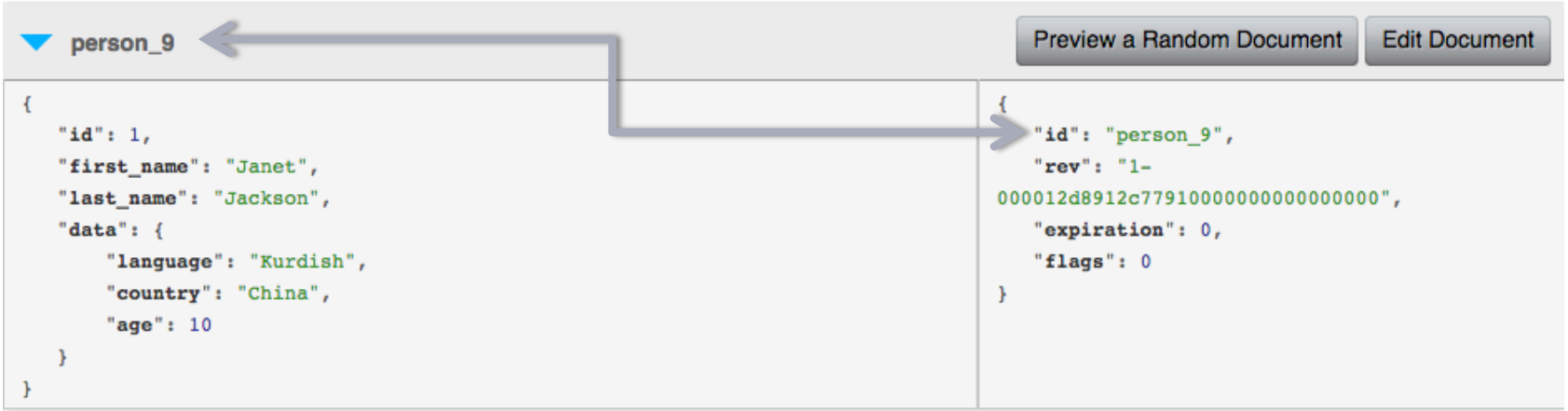
- Read/Write Operationen laufen im RAM
- Hohe Geschwindigkeit & Performance durch halten der Daten im RAM
- Persistieren der Daten auf die Platte
- Working Set im RAM für Daten, die oft genutzt werden
- Documents werden im JSON-Format gespeichert
- Zahlen, Strings, Booleans, Arrays & Objekte werden unterstützt

COUCHBASE SERVER BUCKETS

- **Data Management mittels *Buckets* (vgl. Datenbank)**
- **Buckets sind virtuelle Container**
- **2 Arten: memcached (eingeschränkter) & Couchbase Buckets**
- **Couchbase Buckets unterstützen**
  - Caching der Daten im RAM
  - Asynchrone Persistenz auf Platte
  - Replikation der Buckets auf mehrere Nodes
  - Rebalancing (dynamische Last-Verteilung)
- **Authentifizierung mittels SASL (Name & Passwort) möglich**

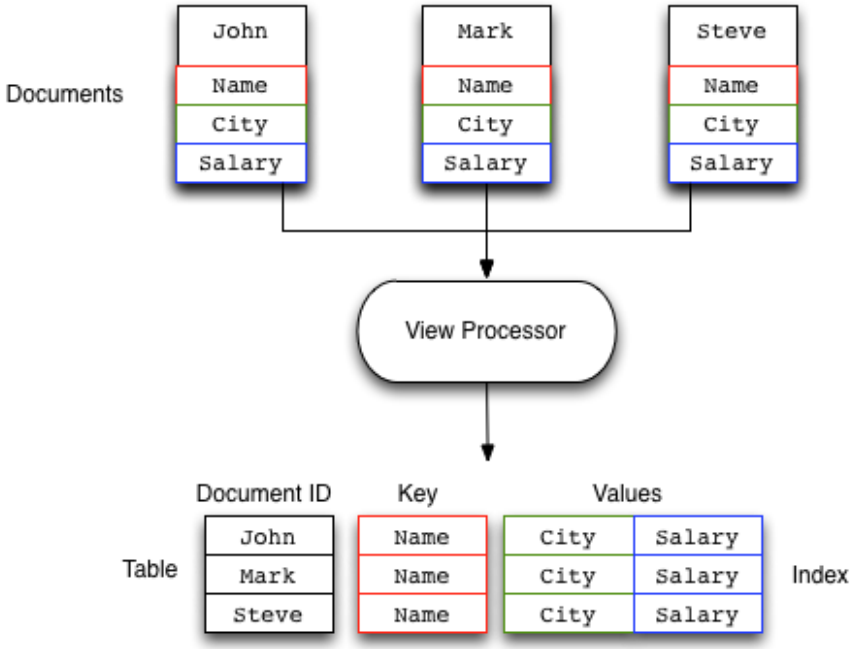
COUCHBASE SERVER DOCUMENTS & METADATA

- Metadata-Objekt zu jedem Document
- Löschung durch Delete-Flag & Tombstones



COUCHBASE SERVER VIEWS

- Indexierung & Abfragen der Daten
- Auswahl spezifischer Felder und bilden eines Index der Daten
- Mehrere unterschiedliche Views möglich
- Daten mit Ablaufdatum werden automatisch entfernt
- Map/reduce-



COUCHBASE SERVER VIEWS

- map()-Funktion in JavaScript
- Parameter der map()-Funktion: doc (Document, Pflicht) & meta (Metadata, Optional)
- emit()-Funktion erzeugt das Ergebnis
- emit() akzeptiert zwei Parameter/ Arrays für Key und Value
- Key wird genutzt um Output zu spezifizieren
- Felder die nicht existieren werden einfach als null ausgegeben

```
function (doc, meta){
  emit(meta.id, doc);
}
```

COUCHBASE SERVER VIEWS & QUERYING

Das Ergebnis eines Views kann per HTTP GET nach folgendem Schema ausgelesen werden:

```
http://[localhost]:8092/[bucket-name]/_design/[ddoc-name]/_view/[view-name]
```

Ergebnis einschränken durch anhängen von Parametern:

```
…/_view/[view-name]?key="…"&limit=00&…
```

COUCHBASE SERVER VIEW SAMPLE

Map-Function:

```
function (doc, meta) {
          emit(meta.id, ["Last name: " + doc.last_name, "First name: “ + doc.first_name]);
}
```

Query:

```
http://127.0.0.1:8092/people/_design/dev_people/_view/allpeople?limit=3
```

Result:


```
{"total_rows":20,"rows":[
{"id":"person_1","key":"person_1","value":["Last name: Carter","First name: Frank"]},
{"id":"person_10","key":"person_10","value":["Last name: Walker","First name: Joan"]},
{"id":"person_11","key":"person_11","value":["Last name: Reid","First name: Virginia"]}
]
}
```

COUCHBASE MOBILE

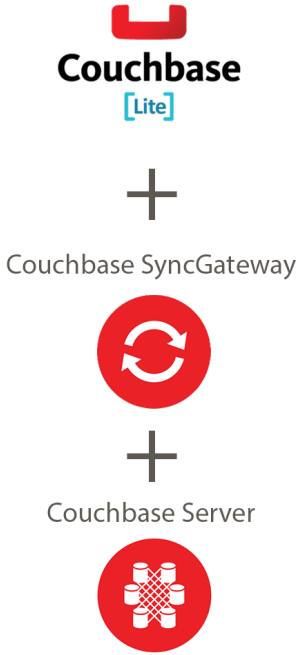
Couchbase Lite

- **Embedded JSON Datenbank**
- **Standalone und/oder lokaler Server „Außenpunkt“**
- **REST API**
- **Native APIs für iOS & Android**

Couchbase SyncGateway

- **Sync Server zum replizieren der lokalen Datenbank auf einen Couchbase Server**

Couchbase Server zur Speicherung der Daten



COUCHBASE LITE

- **Lokale JSON-Datenbank**
- **Persistence Layer mit SQLite**
- **Ähnliche Konzepte wie Couchbase Server**
  - Buckets bzw. Database
  - Views & Queries
  - Lokaler Manager wird benötigt
  - Datenbank ist immer an Manager gebunden
  - Views sind an Datenbank gebunden
  - Attachements möglich (z.B. Bilder)

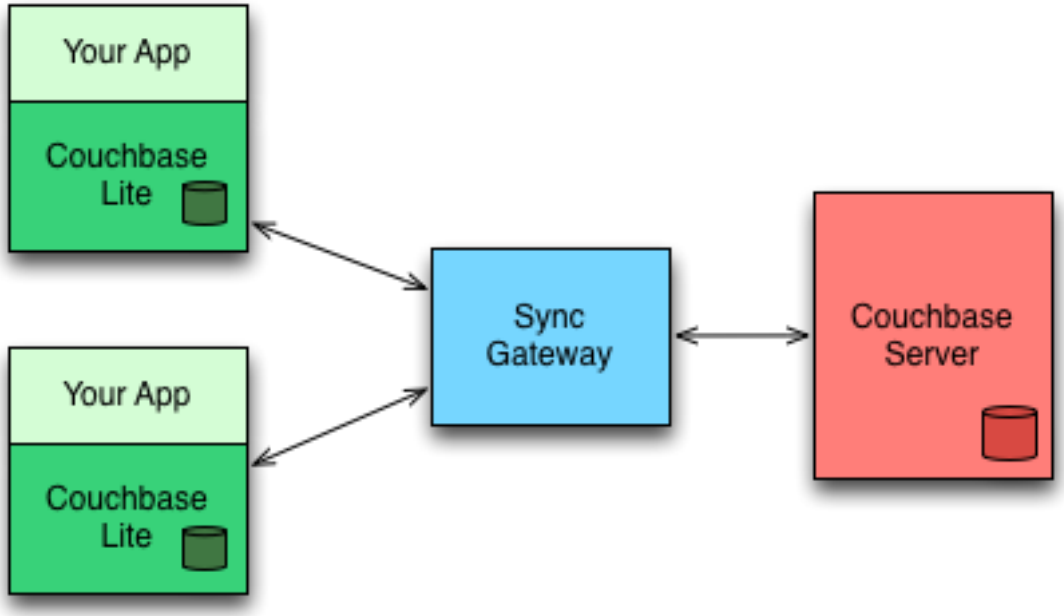
COUCHBASE LITE

Replicator

- **Push/Pull Replication ist vorbereitet**
- **Asynchrone Replication in einem Background Thread**
- **Replicator muss nur die URL des Server kennen und kümmert sich dann um den Sync**
- **Replicator synchronisiert bei vorhandener Verbindung kontinuierlich**
- **Filtered Replication & Authentication möglich**
- **Peer-to-peer Replication möglich (anderes Gerät mit Couchbase Lite + HTTP)**

COUCHBASE SYNCGATEWAY

- SyncGateway stellt Verbindung zum Server her
- Zwischenschicht zwischen App und Server



COUCHBASE SYNCGATEWAY

- sync_gateway Binary mit Config-Datei ausführen
- Admin REST API (Accounts, Roles | Port 4985)
- Sync REST API (Client Replication | Port 4984)

```
{  "interface": ":4984",
  "adminInterface": ":4985",
  "log": ["CRUD", "HTTP+", "REST"],
  "databases": {
    "bucketname": {
      "server": "http://127.0.0.1:8091",
      "username": "bucketname",
      "password": "bucketpassword",
      "sync":
        `function(doc){
          channel(doc.channels);
        }`,
      "users": {
        "testuser": {
          "name": "testuser”,
          "password": "1234"
        },
        "GUEST": {
          "disabled": false
         }      }      }  }  }
```

COUCHBASE SYNCGATEWAY

Weitere Möglichkeiten des SyncGateway:

- **Authentication-Methods (Facebook, Persona, …)**
- **Sync-Function zur Validierung der Documents und des Users, Zuweisung zu Channels**
- **Steuerung der Zugriffe für User mittels Channels und Roles**

AUSBLICK ÜBUNG

Teil 1:

Couchbase Server ausprobieren

Teil 2:

Kleine Quick’n’Dirty-Android-App mit AndroidStudio, Couchbase Lite & Couchbase SyncGateway

Link zum GithubRepo inkl. Übung & Folien:

github.com/thomasrehm/CouchbaseSample

THAT‘S IT.

FRAGEN?

VIELEN DANK FÜR DIE AUFMERKSAMKEIT & VIEL SPASS IN DER ÜBUNG!

ENDE

QUELLEN

Couchbase, Couchbase Mobile.
Available at: http://www.couchbase.com/nosql-databases/couchbase-mobile [Accessed February 26, 2015a].

Couchbase, Couchbase Open Source.
Available at: http://www.couchbase.com/open-source [Accessed February 26, 2015b].

Dj Walker-Morgan, 2010. NoSQL im Überblick. heise Open Source.
Available at: http://www.heise.de/open/artikel/NoSQL-im-Ueberblick-1012483.html [Accessed February 22, 2015].

Hoff, T., 2011. 35+ Use Cases for Choosing Your Next NoSQL Database - High Scalability -. Available at: http://highscalability.com/blog/2011/6/20/35-use-cases-for-choosing-your-next-nosql-database.html [Accessed February 26, 2015].

Hoff, T., 2010. What the heck are you actually using NoSQL for? - High Scalability -.
Available at: http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html [Accessed February 26, 2015].

Marcus, A., The Architecture of Open Source Applications: The NoSQL Ecosystem.
Available at: http://www.aosabook.org/en/nosql.html [Accessed February 26, 2015].

Couchbase, Why NoSQL?
Available at: http://www.couchbase.com/de/why-nosql/nosql-database

QUELLEN

PDF:

Couchbase, Couchbase Mobile

Couchbase, Couchbase Server 3.0 Documentation

Couchbase, The NoSQL Technical Comparison Report

MongoDB, 2015. Top 5 Considerations When Evaluating NoSQL Databases