JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Protokoll

COUCHBASE

INSY/SEW
4AHITM 2015/16

Matthias Mischek
Benedikt Berger

Version 1.0

Note:

Betreuer:

Begonnen am 1. Mai 2016

Beendet am 7. Mai 2016

Inhaltsverzeichnis

1. Einführung

Um uns mit Couchbase mehr vertraut zu machen, wurde uns die folgende Aufgabe gegeben. Mit dieser Übung soll ein umfangreiches Protokoll für weitere Aufgaben ausgearbeitet werden.

[BOR16]

1.1. Ziele

Das Ziel dieser Aufgabe besteht darin, ein Protokoll zu verfassen, welches das Arbeiten bei weiteren Projekten helfen soll.

1.2. Voraussetzungen

- Postgresql Kenntnisse
- JavaFx Kenntnisse
- Java Kenntnisse
- Linux Kenntnisse

1.3. Aufgabenstellung

Protokollieren Sie die einzelnen Schritte zur Installation und Inbetriebnahme sowie die Verwendung einer API (Java, PHP, Python, Node.js oder C) mit Couchbase. Gehen Sie dabei näher auf das Dokumentenformat und die Abfrage (Views) der Daten ein. Verwenden Sie dabei auch die CLI um auch in der Konsole mit Couchbase arbeiten zu können.

Abzugeben ist ein detailliertes Protokoll als Teamarbeit (2er Gruppen) zur Unterstützung und Nachschlagewerk für die zukünftige Verwendung von Couchbase. Vergessen Sie nicht die wichtigen CRUD Befehle ins Protokoll aufzunehmen (SDK und CLI, Tipp: cbtransfer).

Folgende Eckpunkte sollen enthalten sein:

- Installation
- CLI Befehle
- API Installation und Verwendung
- Dokumentenstruktur (JSON, GSON)
- Erläutern der Indizierung und des Map/Reduce Vorgangs
- Erstellung und Verwendung von Views

[ELE01]

2. Vorbereitungen

2.1. Couchbase installieren

Es wurde Couchbase Server Enterprise Edition für Debain 8 verwendet.

Das Paket kann hier heruntergeladen werden: http://www.couchbase.com/nosql-databases/downloads

Um das Paket zu installieren wird folgender Befehl verwendet:

```
dpkg -i 'pfad/couchbase_paket.de'
```

Debian-Paketverwaltungswerkzeuge: https://www.debian.org/doc/manuals/debian-faq/ch-pkgtools.de.html

2.2. Couchbase konfigurieren

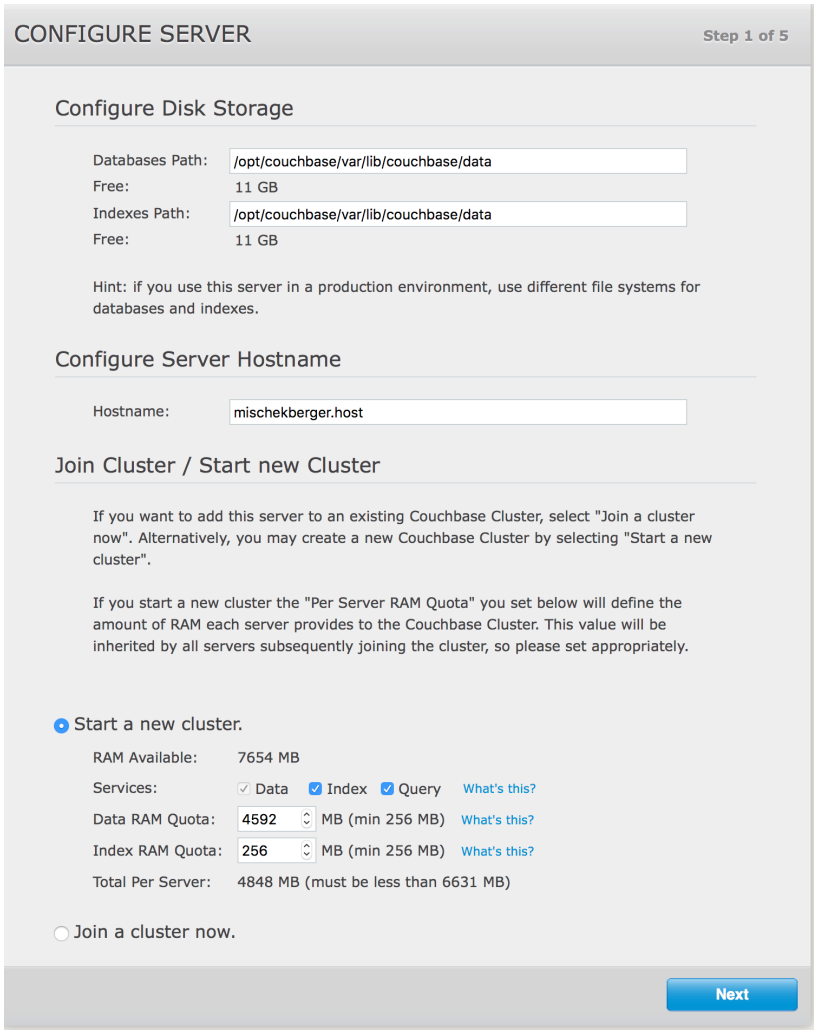
CouchBase kann im Browser über der IP-Adresse des Servers und dem Port 8091 erreicht werden.

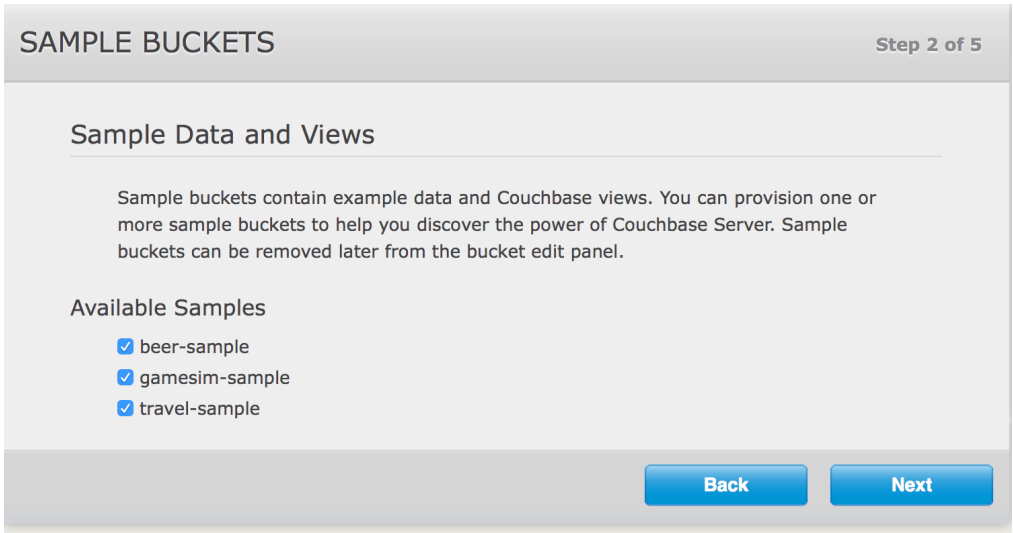
Im nächsten Fenster wird der Server konfiguriert. Um einen beliebigen Hostname eintragen zu können, muss dieser erst in der Host-Datei am Server eingetragen werden, welche sich in dem folgenden Verzeichnis befindet:

/etc/hosts

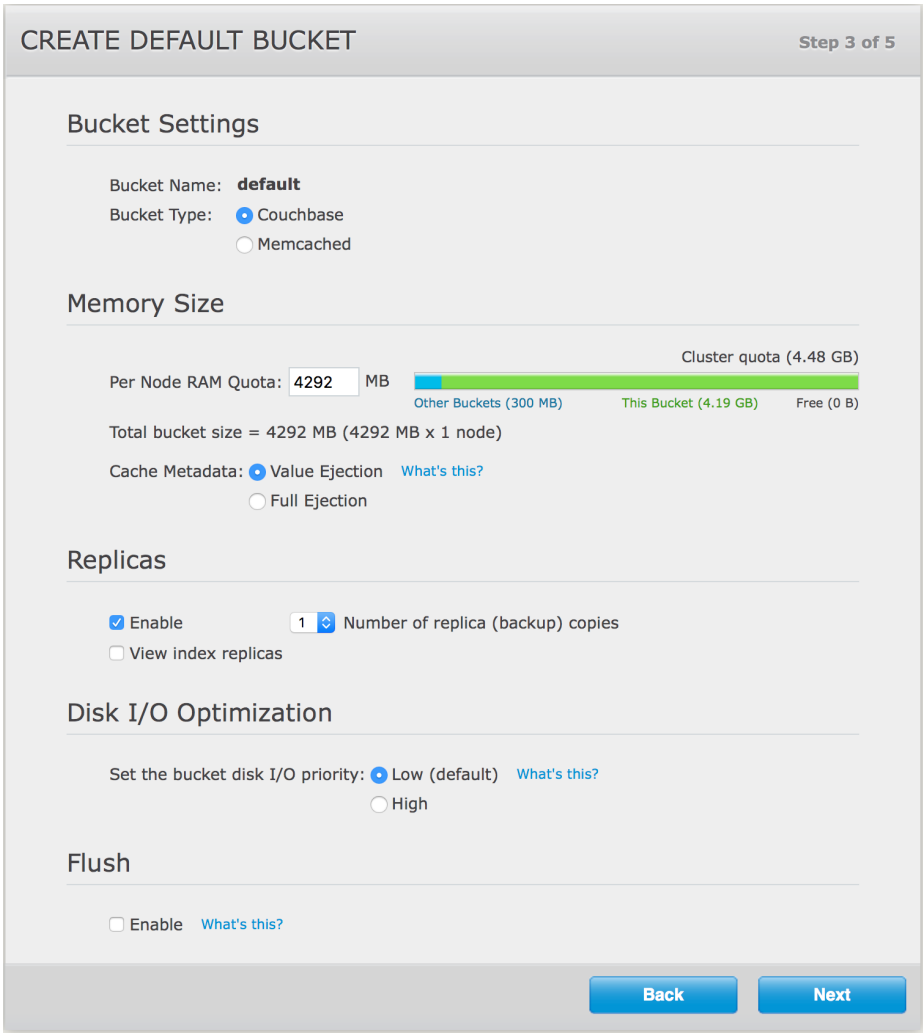
Des Weiteren wird ein neuer Cluster erstellt, wobei der zugewiesene RAM eingetragen werden kann.

Diesem sollte optimaler weise mehr als 4 GB zugewiesen werden, da die Datensätze im RAM gespeichert werden.



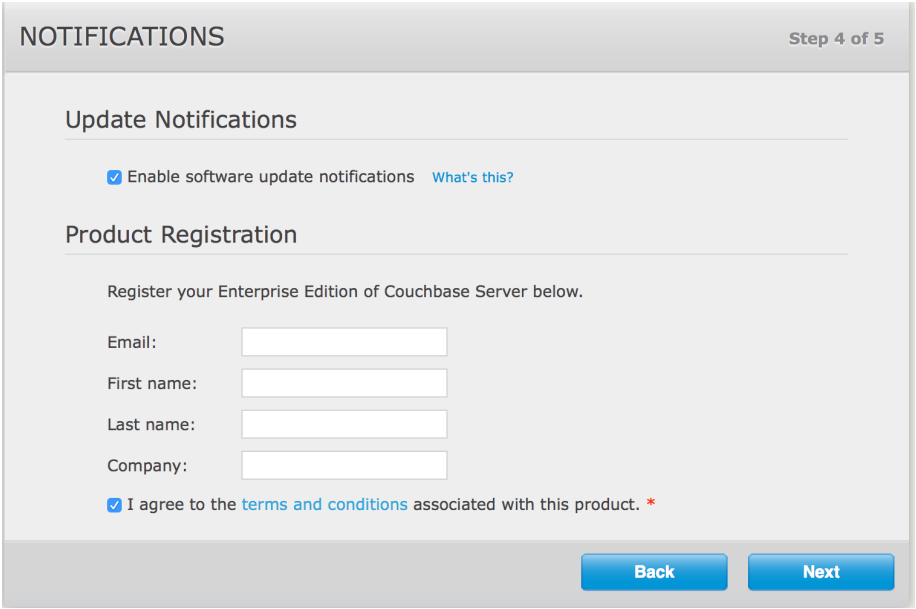


Im nächsten Schritt hat man die Möglichkeit, Sample Buckets auszuwählen, welche Beispieldaten in die Datenbank spielen.

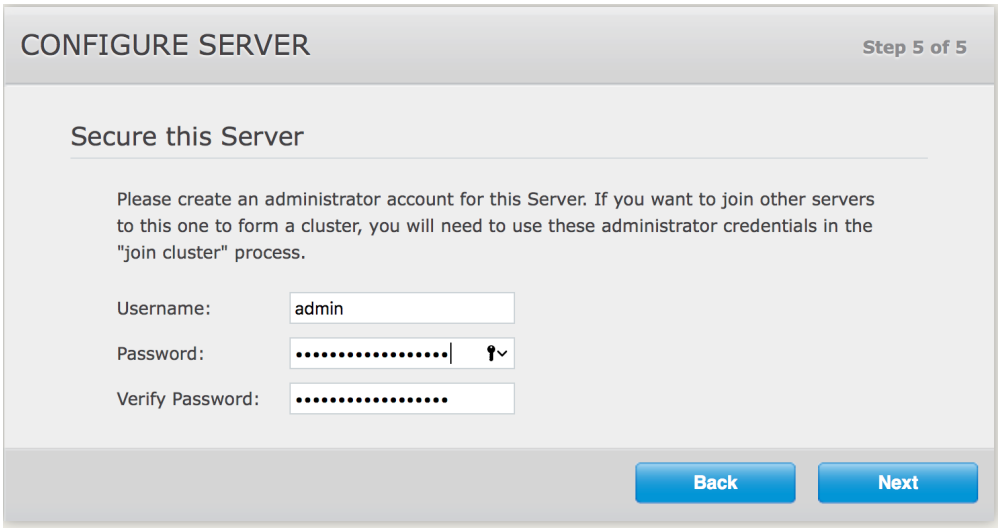


Durch klicken auf "Next" kommt man zum "Create default bucket" Fenster, wobei die Einstellungen beibehalten werden und einfach fortgefahren wird.

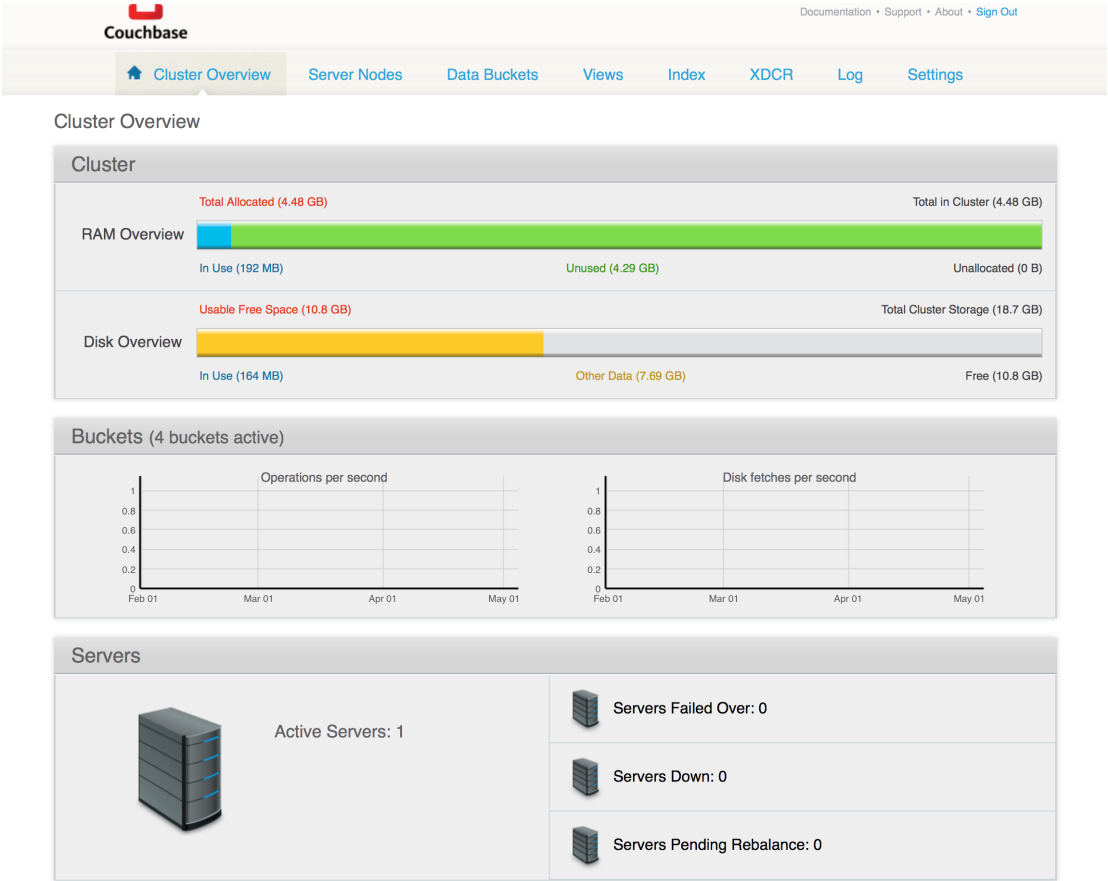
Beim Fenster "Notifications" werden die Benutzungsvereinbarungen einfach nur bestätigt.



Jetzt müssen die Zugangsdaten des Administrator Accounts eingegeben werden.



Nun ist Couchase fertig konfiguriert und einsatzbereit:



2.3. CLI Befehle

Command	Description
bucket-compact	Compact database and index data.
bucket-create	Add a new bucket to the cluster.
bucket-delete	Delete an existing bucket.
bucket-edit	Modify an existing bucket.
bucket-flush	Flush all data from disk for a given bucket.
bucket-list	List all buckets in a cluster.
cluster-edit	Modify cluster settings.
cluster-init	Set the username, password and port of the cluster.
failover	Fail over one or more servers. Default: Graceful failover Hard failover is implemented with the `--force` option.
group-manage	Manages server groups (Enterprise Edition only).
help show longer	Syntax, usage, and examples.
node-init	Set node specific parameters.
rebalance	Start a cluster rebalancing.
rebalance-stop	Stop current cluster rebalancing.
rebalance-status	Show status of current cluster rebalancing.
server-add	Add one or more servers to the cluster.
server-info	Show details on one server.
server-list	List all servers in a cluster.
server-readd	Readds a server that was failed over.
setting-alert	Email alert settings.
setting-autofailover	Set auto failover settings.

setting-cluster	Set various options for a cluster.
setting-compaction	Set auto compaction settings.
setting-notification	Set notifications.
setting-xdcr	Set XDCR-related configuration which affect behavior.
ssl-manage	Manage cluster SSL certificate.
xdcr-replicate	Create and run replication via XDCR.
xdcr-setup	Set up XDCR replication.

[CBD01]

3. CRUD

Die Verbindung zur Datenbank wird aufgebaut:

```
Cluster cluster = CouchbaseCluster.create("192.168.74.135");
Bucket bucket = cluster.openBucket("default", 60, TimeUnit.SECONDS);
```

Ein neues JSON-Objekt kann wie folgt erstellt werden. Mit *put()* können neue Attribute hinzugefügt werden mit den beiden Parametern (1. -> key, 2. -> value):

```
JsonObject person1 = JsonObject.empty()
      .put("firstname", "Benedikt")
      .put("lastname", "Berger")
      .put("age", 16)
      .put("city", "Wien");
```

Ein neues JSON-Dokument wird mit den eben erstellten Attributen erstellt. Hinzufügen des JSON-Dokuments in den Bucket.

```
JsonDocument doc1 = JsonDocument.create("bberger", person1);
JsonDocument response1 = bucket.upsert(doc1);
```

Ein JsonDocument mit einem bestimmten Namen kann mit der folgenden Methode erhalten werden:

```
JsonDocument bberger = bucket.get(„bberger");
```

Wenn das JsonDocument beispielsweise in der Variable bberger gespeichert wird, kann jetzt auf den Content wie folgt zugegriffen werden:

```
System.out.println("Last name: " + bberger.content().getString("lastname"));
```

Der vollständige Code ist im Github Repository einsichtig.

4. Datenstruktur

4.1. JSON

JSON (JavaScript Object Notation) ist ein kompaktes Datenaustauschformat, das für Menschen einfach zu lesen und für Maschinen einfach zu parsen und zu generieren ist.

JSON-Dokumente sind (mit kleinen Ausnahmen) valider JavaScript-Code, welcher 1:1 in JavaScript-Objekte umgewandelt werden kann.
Zu den Hauptanwendungsgebieten zählen JavaScript-Applikationen, Ajax, Webapplikationen und Webservices.

JSON kennt folgende Datentypen:

- Strings, Boolean, Zahlen
- null
- Arrays (in eckigen Klammern [])
- Objekte (in geschwungenen Klammern {})

Code:

```
{
   "array": ["eins","zwei","drei"],
   "boolean": true,
   "null": null,
   "number": 123,
   "object" : {
      "Mischek"   : "https://www.mmischek.at",
      "Berger"  : „https://www.bberger.com"
   },
   "string": „Das ist ein String!“
}
```

Objekte beinhalten eine (ungeordnete) Liste von Eigenschaften, diese bestehen aus einem Schlüssel (Zeichenkette) und einem Wert. Ferner können Hierarchien dargestellt werden.

[JSO01]

4.2. GSON

GSON ist eine open Source Java Library um zwischen Java Objekten und JSON Objekten zu konvertieren.
Um Objekte zu konvertieren werden einfach die Methoden toJson() und fromJson() verwendet.

Einfaches Beispiel:

```
package com.javacreed.examples.gson.part1;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

public class SimpleExample1 {
    public static void main(String[] args) {
        Gson gson = new GsonBuilder().create();
        gson.toJson("Hello", System.out);
        gson.toJson(123, System.out);
    }
}
```

In diesem Beispiel wird eine Instanz von GSON erstellt und der Java String und int wird in JSON Objekte konvertiert.

GitHub - GSON von Google: https://github.com/google/gson
GSON API: http://www.javadoc.io/doc/com.google.code.gson/gson/2.6.2

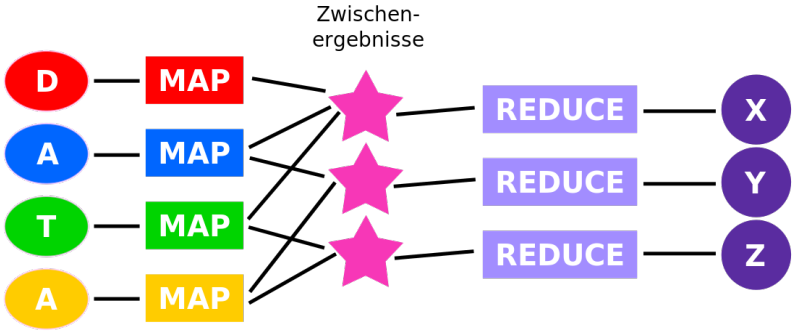
[JCR01]

5. Map/Reduce

Map/Reduce ist ein Programmiermodell zur Verarbeitung von großen unstrukturierten oder semi-strukturierten Datensätzen auf Computerclustern. Das Programmiermodell nutzt verteilte Speicherung der Daten in Blöcken. Ferner sorgt es für eine Aufteilung der Berechnungen auf mehreren Recheneinheiten, somit lassen sich die Berechnungen parallelisieren.
Beim MapReduce-Verfahren werden die Daten in drei Phasen verarbeitet (Map, Shuffle, Reduce) von denen zwei durch den Anwender spezifiziert werden (Map und Reduce).

Nähere praktische Anwendungen in Kapitel 6.

5.1. Arbeitsweise



Die Eingabedaten (D, A, T, A) werden auf eine Reihe von Map-Prozessen verteilt (bunte Rechtecke), welche jeweils die vom Nutzer bereitgestellte Map-Funktion berechnen.
Die Map-Prozesse werden idealerweise parallel ausgeführt.
Jede dieser Map-Instanzen legt Zwischenergebnisse ab (dargestellt durch die pink-farbigen Sterne).
Von jeder Map-Instanz fließen Daten in eventuell verschiedene Zwischenergebnisspeicher.
Die Verwaltung der Zwischenergebnisse wird auch als Shuffle-Phase bezeichnet, da hier in der Regel die Daten zwischen vielen Computersystemen ausgetauscht werden müssen.
Sind alle Zwischenergebnisse berechnet, ist diese sogenannte Map-Phase beendet und die Reduce-Phase beginnt.
Für jeden Satz an Zwischenergebnissen berechnet jeweils genau ein Reduce-Prozess (violette Rechtecke) die vom Nutzer bereitgestellte Reduce-Funktion und damit die Ausgabedaten (violette Kreise X, Y und Z).
Die Reduce-Prozesse werden idealerweise auch parallel ausgeführt.

[WIK01, RZW01, ULD01]

6. Views

In Couchbase ermöglicht eine View das Indizieren und Abfragen von Daten.
Eine View erzeugt einen Index für die Daten gemäß dem festgelegten Format und Struktur.
Views sind letztendlich konsistent, im Vergleich zu den zugrunde liegenden gespeicherten Dokumente.

Views werden verwendet für:

- Indizierung und Abfrage von Daten aus gespeicherten Objekten
- Erzeugen von Tabellen
- Extrahieren oder Herausfiltern von Informationen aus der Datenbank

Der View Builder bietet eine Schnittstelle für das Erstellen von Views über die Web-Konsole.

6.1. Vorbereitung für Kommunikation zwischen Java und CouchBase

Durch das importieren der erforderlichen Libraries mithilfe einer Entwicklungsumgebung wie "Eclipse", kann mit der Datenbank kommuniziert werden.
Die folgenden Imports sind dazu notwendig:

```
import com.couchbase.client.java.Bucket;
import com.couchbase.client.java.Cluster;
import com.couchbase.client.java.bucket.BucketManager;
import
com.couchbase.client.java.error.DesignDocumentAlreadyExistsException;
import com.couchbase.client.java.view.DefaultView;
import com.couchbase.client.java.CouchbaseCluster;
import com.couchbase.client.java.view.DesignDocument;
```

6.2. Verbindung zu Datenbank aufbauen

Zu aller erst muss eine Verbindung zur Datenbank aufgebaut werden, um mit ihr kommunizieren zu können. Das wird in Java wie folgt umgesetzt:

```
public void connect(){
      cluster = CouchbaseCluster.create("192.168.74.135");
      bucket = cluster.openBucket("beer-sample", 60, TimeUnit.SECONDS);
      bucketManager = bucket.bucketManager();
}
```

In der 1. Zeile der Methode muss ein String mit der IP-Adresse des Servers übergeben werden. Als nächstes wird der Bucket in einer Variable gespeichert, wo der Name des Buckets, das Timeout und die Zeiteinheit übergeben werden. Mithilfe des BucketManagers können erstellte Design Documents auch in die Datenbank eingefügt werden.

6.3. View erstellen

Das erstellen von Views in Java wird wie folgt umgesetzt:

```
public void create(){
      try {
            DesignDocument designDoc = DesignDocument.create("name",
                  Arrays.asList(
                        DefaultView.create("bewery_name",
                              "function (doc, meta) { "
                                    + "switch(doc.type) { "
                                    + "case \"brewery\":"
                                    + "emit(meta.id, [doc.name, doc.city]);"
                                    + "}"
                                    + "}")
            ));
            bucketManager.insertDesignDocument(designDoc, true);
      } catch (DesignDocumentAlreadyExistsException e) {
            System.err.print("design document already exists!");
      }
}
```

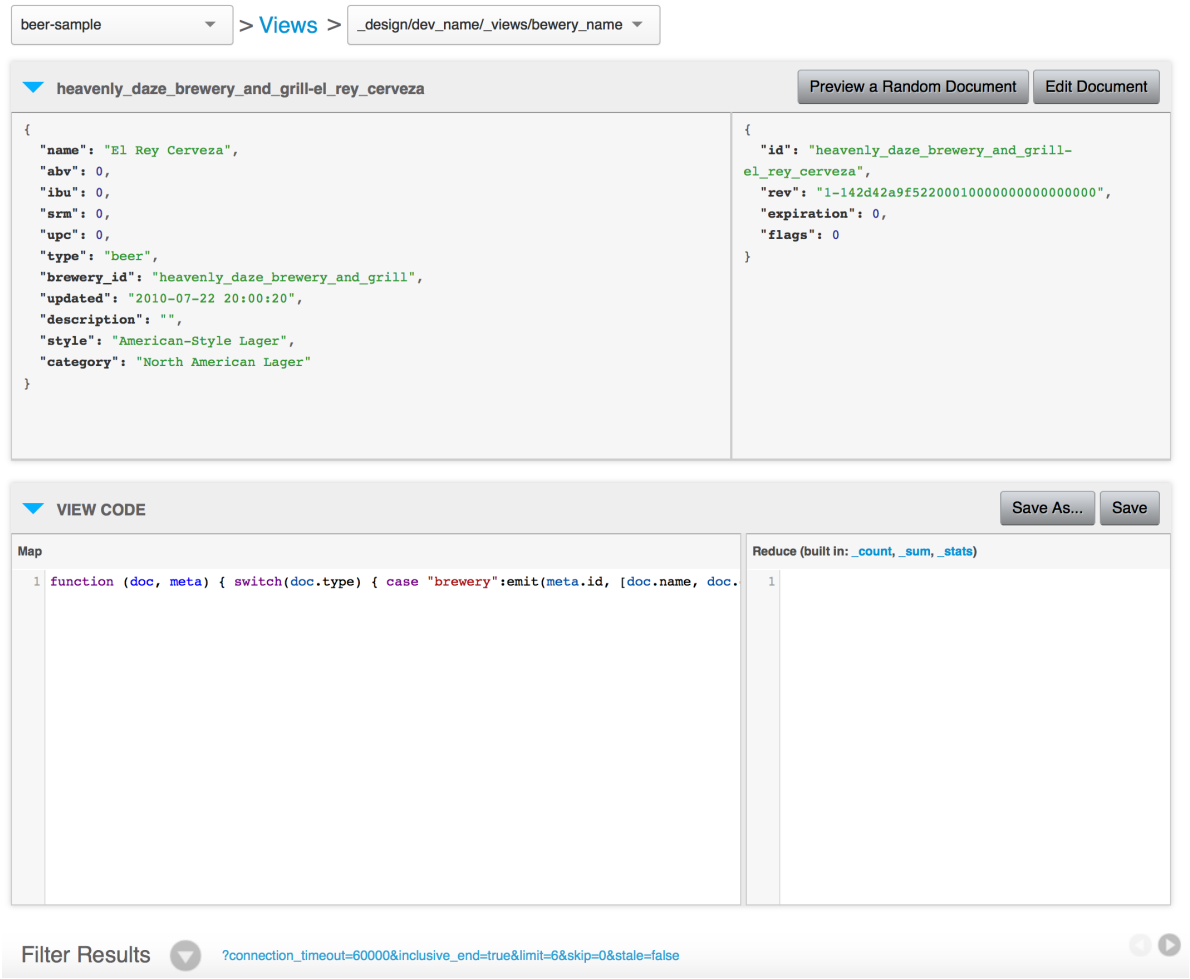
Das try-catch ist notwendig, da wenn dieses design document bereits existiert, ein Fehler ausgeworfen wird, welcher mit dieser Maßnahme aufgefangen wird. In der 1. Zeile innerhalb des try wird ein neues Design Document erstellt, wo zuerst als Parameter der name übergeben werden muss und des weiteren ein Array mit den Views, welche in diesem Fall "bewery_name" heißt. Als zweiten Parameter der *DefaultView.create* Methode wird der tatsächliche View Code angegeben.

6.4. View Code erstellen

Der View Code kann am einfachsten in der CouchBase Oberfläche des Servers getestet und erstellt werden. Hier kann im Reiter *Indexes* > *Views* und anschließend durch Auswählen aus der CombiBox der gewünschte Bucket ausgewählt werden, um anschließend Views erstellen zu können. Durch klicken auf "*Create Development View*" kann ein neuer View erstellt werden.

Im Textfeld mit der Überschrift "View Code" kann der gewünschte Code eingefügt werden und mit "Save" gespeichert werden. Falls Syntax Fehler auftreten sollten, werden diese ebenfalls angezeigt. Wenn keine Fehler auftreten, kann gleich darunter auf den Links rechts neben "Filter Results" geklickt werden, welcher dann in einem neuen Fenster die Ergebnisse anzeigt.

6.5. View auslesen



Eine View kann in Java wie folgt ausgelesen werden:

```
public void read(){
      DesignDocument designDoc = bucketManager.getDesignDocument("name", true);
      System.out.println(designDoc.name() + " contains " + designDoc.views().size(),
+ " view(s)\n");
      List<DesignDocument> designDocs = bucketManager.getDesignDocuments(true);
      System.out.println("the selected bucket '" + bucket.name() + "' contains of "
+ designDocs.size() + " the following design document(s):");
      for (DesignDocument doc : designDocs) {
                  System.out.println("\t" + "the design document " + doc.name() + "
has " + doc.views().size() + " view(s)");
            }
      }
```

In der 1. Zeile des oberen Codes stellt der 1. Parameter der *getDesignDocument* Methode den Namen des Views dar. Die folgenden nächsten Zeilen geben die einzelnen Design Documents und die Anzahl der enthaltenen Views aus.

Der vollständige Code ist ebenfalls auf GitHub verfügbar:

https://github.com/mmischek/Couchbase.git

[CBD02]

7. Management

7.1. Zeitmanagment Mischek

Arbeitsbereich	Zeitlicher Aufwand (in h)
Vorbreitung	3
Datenstruktur	2
Views	1
Map/Reduce	1,5
Gesamtanzahl	**7,5**

7.2. Zeitmanagment Berger

Arbeitsbereich	Zeitlicher Aufwand (in h)
Vorbreitung	2
Views	3
CRUD	2,5
Gesamtanzahl	**7,5**

7.3. Versionierung

Wir haben github.com für die Versionierung dieser Aufgabe verwendet.

Link zu unserem Repository: https://github.com/mmischek/Couchbase.git

8. Quellen

[ELE01] COUCHBASE AUFGABE. MOODLE [ONLINE] AVAILABLE AT:
HTTPS://ELEARNING.TGM.AC.AT/MOD/ASSIGN/VIEW.PHP?ID=40870
[ABGERUFEN AM 18. APRIL 2016]

[JSO01] JSON [ONLINE] AVAILABLE AT:
HTTP://WWW.JSON.ORG/JSON-DE.HTML
[ABGERUFEN AM 18. APRIL 2016]

[JCR01] SIMPLE GSON EXAMPLE [ONLINE] AVAILABLE AT:
HTTP://WWW.JAVACREED.COM/SIMPLE-GSON-EXAMPLE/
[ABGERUFEN AM 18. APRIL 2016]

[CBD01] COUCHBASE-CLI COMMANDS [ONLINE] AVAILABLE AT:
HTTP://DOCS.COUCHBASE.COM/ADMIN/ADMIN/CLI/CBCLI/CBCLI-COMMANDS.HTML
[ABGERUFEN AM 18. APRIL 2016]

[WIK01] MAPREDUCE [ONLINE] AVAILABLE AT:
HTTPS://DE.WIKIPEDIA.ORG/WIKI/MAPREDUCE
[ABGERUFEN AM 18. APRIL 2016]

[RZW01] MAPREDUCE [ONLINE] AVAILABLE AT:
HTTPS://WIKI.RAUMZEITLABOR.DE/IMAGES/0/0A/MAPREDUCE.PDF
[ABGERUFEN AM 18. APRIL 2016]

[ULD01] MAPREDUCE - KONZEPT [ONLINE] AVAILABLE AT:
HTTP://DBS.UNI-LEIPZIG.DE/FILE/SEMINAR_0910_FINDLING_KÖNIG.PDF
[ABGERUFEN AM 18. APRIL 2016]

[CBD02] JAVA SDK 2.2 - COUCHBASE [ONLINE] AVAILABLE AT:
HTTP://DEVELOPER.COUCHBASE.COM/DOCUMENTATION/SERVER/4.1/SDKS/JAVA-2.2/JAVA-INTRO.HTML
[ABGERUFEN AM 18. APRIL 2016]