
Recherche

Couchbase Basics

INSY
4CHITT 2015/16

Daniel May, Martin Weber

Version 1.0

Note:

Begonnen am 25. April 2016

Betreuer: Michael Borko

Beendet am 05. Mai 2016

Inhaltsverzeichnis

1	Einführung	3
1.1	Ziele	3
1.2	Voraussetzungen	3
1.3	Aufgabenstellung	3
2	Ergebnisse	4
2.1	Installation und Inbetriebnahme	4
2.2	CLI Befehle	9
	Create/Update	12
	Read	12
	Delete	12
2.3	API Installation und Verwendung	12
	Verbindung	13
	Administratives	14
	Create	15
	Read	15
	Update	16
	Delete	16
2.4	Dokumentenstruktur	17
2.5	Indizierung & Map-Reduce	18
	MapReduce	18
	Spatial View	19
	Global Secondary Index (GSI)	20
2.6	Erstellung und Verwendung von eigenen Views	20
3	GitHub Link	23
4	Zeitmanagement	23
5	Literaturverzeichnis	25

1 Einführung

Nachdem NoSQL im Unterricht besprochen wurde, soll auch praktisch damit gearbeitet werden. Dafür wird vorerst das dokumentenorientierte, key-value basierte System Couchbase verwendet.

1.1 Ziele

Ziel ist es sich mit der Verwendung von Couchbase vertraut zu machen, als auch ein Nachschlagewerk zur Unterstützung zu schaffen.

1.2 Voraussetzungen

Vorausgesetzt werden die theoretischen Grundlagen zu NoSQL.

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“ [BOR]

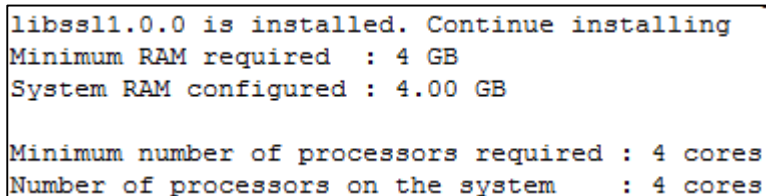
2 Ergebnisse

2.1 Installation und Inbetriebnahme

Zuerst wählt man auf der Couchbase Homepage zwischen der Enterprise Edition und der Community Edition von Couchbase Server. Da man die Enterprise Edition ohne Bezahlung verwenden kann, wird im Unterricht diese Version verwendet. Danach wählt man eine Version, üblicherweise den letzten stabilen Release, sowie das verwendete System aus und lädt die Installationsdaten herunter. Da wir Debian verwenden, erhält man ein .deb Paket. [COU.1]

Nun muss eine virtuelle Maschine erstellt werden. Für Couchbase sollte eine separate VM genutzt werden, da es relativ viele Ressourcen benötigt. Derzeit liegen die minimalen Systemanforderungen bei 4 Prozessorkernen und 4 GB RAM.

Vor der Installation sollte die aktuellste verfügbare Version von OpenSSL installiert werden. Das Debian Package heißt `libssl<version>`. Installiert wird das Paket, mit dem Befehl `dpkg -i couchbasePackage.deb`. [COU.2]



```
libssl1.0.0 is installed. Continue installing
Minimum RAM required   : 4 GB
System RAM configured  : 4.00 GB

Minimum number of processors required : 4 cores
Number of processors on the system    : 4 cores
```

Abbildung 1: Installationsausgabe

Standardmäßig befinden sich nach der Installation alle Dateien unter `/opt/couchbase`.

Zusätzlich wurde ich noch darauf hingewiesen, zwei Linux Funktionen zu deaktivieren um die Performance der Datenbank zu erhöhen.

Zuerst habe ich die Transparent Huge Pages (THP) deaktiviert. Diese dienen normalerweise der Performance, indem sie zusammengehörigen pre-allokierten Speicher erstellen. Es ist jedoch empfohlen diese Funktion für Datenbanken zu deaktivieren. Nähere Informationen sind auf der Website vorhanden, die der Installer automatisch ausgibt, sollten die THP aktiviert sein. [COU.3]

Danach wurde, wie ebenfalls im Installer darauf hingewiesen, die sogenannte Swappiness auf 0 gesetzt. Dies bewirkt, dass der Arbeitsspeicher möglichst nicht auf den Festspeicher ausgelagert wird. Dies soll ebenfalls eine Performancesteigerung bei Couchbase erwirken. [COU.4]

Der Serverprozess wird automatisch nach der Installation gestartet. Wird eine Firewall verwendet, sollten die Ports, auf die im Installer hingewiesen wird, freigeschalten werden. Nun kann man unter `http://<couchbaseaddress>:8091` das Setup aufrufen.

CONFIGURE SERVER

Step 1 of 5

Configure Disk Storage

Databases Path:

Free: 17 GB

Indexes Path:

Free: 17 GB

Hint: if you use this server in a production environment, use different file systems for databases and indexes.

Configure Server Hostname

Hostname:

Join Cluster / Start new Cluster

If you want to add this server to an existing Couchbase Cluster, select "Join a cluster now". Alternatively, you may create a new Couchbase Cluster by selecting "Start a new cluster".

If you start a new cluster the "Per Server RAM Quota" you set below will define the amount of RAM each server provides to the Couchbase Cluster. This value will be inherited by all servers subsequently joining the cluster, so please set appropriately.

☒ Start a new cluster.

RAM Available: 4092 MB

Services: ☒ Data ☒ Index ☒ Query [What's this?](#)

Data RAM Quota: MB (min 256 MB) [What's this?](#)

Index RAM Quota: MB (min 256 MB) [What's this?](#)

Total Per Server: 2711 MB (must be less than 3274 MB)

☐ Join a cluster now.

Next

Abbildung 2: Konfiguration 1

Für nicht produzierende Umgebungen können großteils die Default-Werte akzeptiert werden, trotzdem wird das Setup hier ein wenig erläutert.

Zuerst kann bestimmt werden, wo die Datenbanken und die Indizes gespeichert werden. In einer produzierenden Umgebung sollten es zwei unterschiedliche Speicherorte sein.

Danach kann der Hostname des Servers gesetzt werden.

Nun muss man sich entscheiden, ob diese Couchbase Installation einen neuen Cluster erstellt oder einem Cluster beiträgt. Hier lässt sich gut erkennen, dass Couchbase für Hochverfügbarkeit ausgelegt ist.

Wird ein neuer Cluster erstellt, müssen zuerst die gewünschten Services konfiguriert werden. Auf jeden Fall muss diese Node (Couchbase Installation) die Daten beinhalten, zusätzlich können noch die Services Index und Query gewählt werden. Die Höhe des allokierten Hauptspeichers muss ebenfalls für die Daten, sowie den Index gesetzt werden. Diese Werte gelten für alle Nodes, die in Zukunft dem Cluster beitreten.

Tritt man einem Cluster bei, so muss man nur die IP-Adresse und die Benutzerdaten des Administrators der Node eingeben. Zusätzlich können wieder die gewünschten Services gewählt werden.

Im zweiten Schritt können Beispiele installiert werden. Diese beinhalten Datenbanken (Buckets) mit Daten und Views.

CREATE DEFAULT BUCKET

Step 3 of 5

Bucket Settings

Bucket Name: **default**

Bucket Type: ☒ Couchbase
☐ Memcached

Memory Size

Per Node RAM Quota: MB

Cluster quota (2.39 GB)

Other Buckets (100 MB) This Bucket (2.29 GB) Free (0 B)

Total bucket size = 2355 MB (2355 MB x 1 node)

Cache Metadata: ☒ Value Ejection [What's this?](#)
☐ Full Ejection

Replicas

☒ Enable Number of replica (backup) copies
☐ View index replicas

Disk I/O Optimization

Set the bucket disk I/O priority: ☒ Low (default) [What's this?](#)
☐ High

Flush

☐ Enable [What's this?](#)

Back

Next

Abbildung 3: Dritter Konfigurationsschritt

Hier wird der default-Bucket konfiguriert. Dieser sollte ausschließlich für Testzwecke verwendet werden. Beim Bucket Type sollte Couchbase gewählt werden, da hier mehr Funktionen wie Persistenz, Replikation, Rebalance, XDCR und Backup verfügbar sind. Bei Memcached werden die Daten nur im RAM gespeichert, bei Couchbase zusätzlich noch persistiert. Ein Anwendungsfall für Memcached ist die Verwendung parallel zu einem RDBMS. Die häufig verwendeten Daten des RDBMS werden in einem Memcached Bucket zusätzlich

zur Verfügung gestellt, um die Performance zu erhöhen. [COU.6]

Danach kann die Höhe des allokierten Hauptspeichers für diesen Bucket gesetzt werden. Dieser Wert gilt ebenfalls auf jeder Node. Nun kann das Verhalten der gecachten Metadaten gesetzt werden. Value Ejection benötigt mehr Speicherplatz, bietet aber eine bessere Performance. Full Ejection reduziert den Speicherplatz Overhead.

Danach wird die Anzahl der Backup Kopien (replicas) gesetzt. Standardmäßig werden alle Daten (Dokumente) und Funktionen repliziert. Zusätzlich können View Index Replicas erstellt werden. Damit wird der Index auf jeder Node aus den replizierten Daten konstruiert. Dies erlaubt auch nach dem Ausfall einer Node, schnelle Queries auszuführen. [COU.7]

Für jeden Bucket kann auch die Disk I/O Priority gesetzt werden. Dementsprechend werden bei gewissen Operationen Buckets mit der Priority High bevorzugt.

Nun kann gesetzt werden ob die Daten im Bucket geflusht werden können. Das heißt ob der Bucket geleert werden kann. In einer Produktionsumgebung sollte diese Option nicht gesetzt werden.

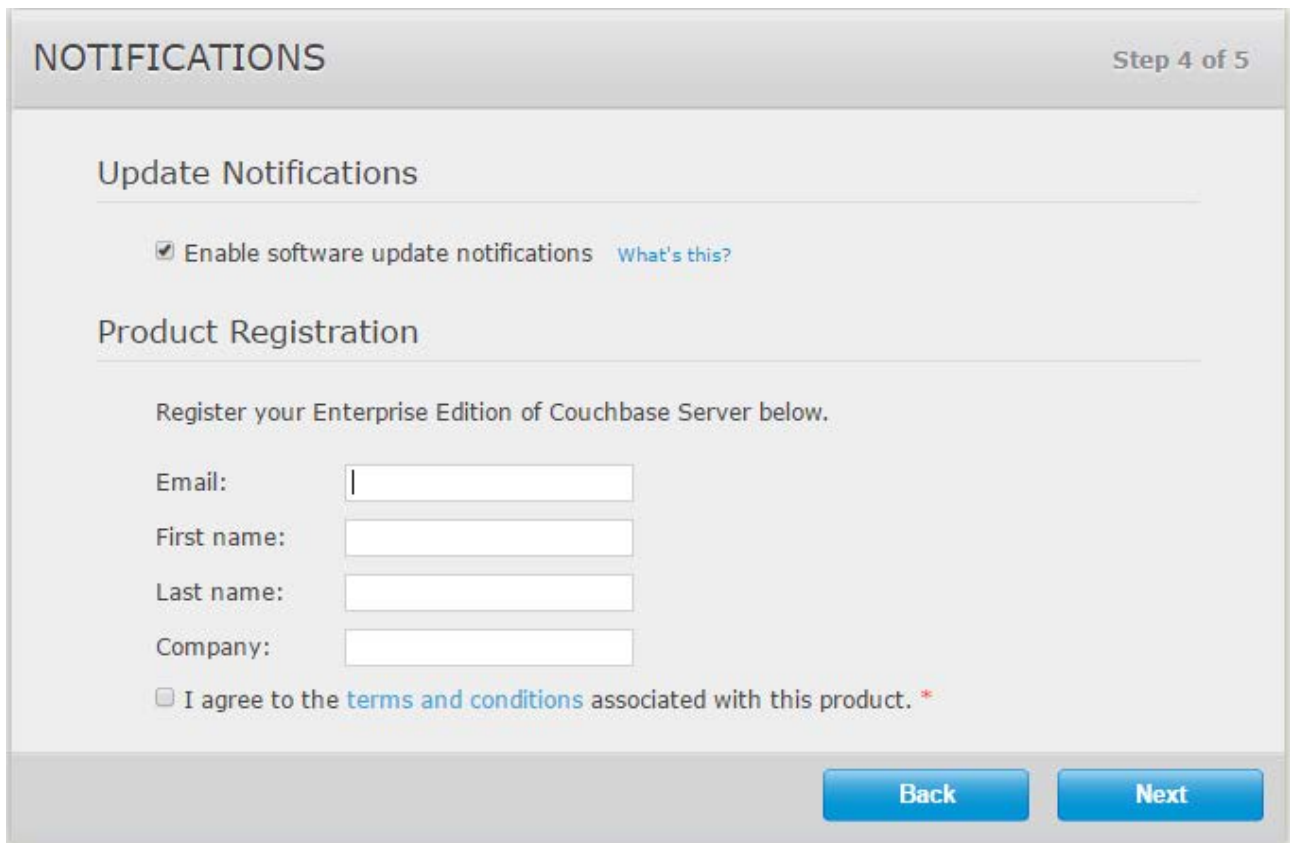


Abbildung 4: Benachrichtigungen

Nun wird das Benachrichtigungsverhalten gesetzt.

Zuletzt muss noch ein Administrator Account angelegt werden. [COU.5]

2.2 CLI Befehle

Couchbase bietet die Möglichkeit viele Funktionen auf der Konsole auszuführen. Die couchbase-cli wird dabei je nach System in einen anderen Ordner installiert.

Operating System	Directory Locations
Linux	/opt/couchbase/bin, /opt/couchbase/bin/install, and /opt/couchbase/bin/tools
Windows	C:\Program Files\couchbase\server\bin, C:\Program Files\couchbase\server\bin\install, and C:\Program Files\couchbase\server\bin\tools.
Mac OS X	/Applications/Couchbase Server.app/Contents/Resources/couchbase-core/bin /Applications/Couchbase Server.app/Contents/Resources/couchbase-core/bin/tools /Applications/Couchbase Server.app/Contents/Resources/couchbasecore/bin/install

Abbildung 5 Couchbase CLI [COU.9]

Die grundlegende Struktur eines Befehls ist Folgende:

```
couchbase-cli COMMAND CLUSTER [options]
```

couchbase-cli ist hier die sich im oben angegebenen Ordner befindende ausführbare Datei. Bei COMMAND kann ein beliebiger Befehl aus der Liste der möglichen eingegeben werden. Cluster gibt den Cluster an. Der Cluster wird in der Form

```
-c [host]:[port] bzw. --cluster=[host]:[port]
```

angegeben.

[COU.9]

Die möglichen OPTIONS sind folgende:

Option	Description
<code>-u USERNAME,</code> <code>--user=USERNAME</code>	Admin username of the cluster.
<code>-p PASSWORD,</code> <code>--password=PASSWORD</code>	Admin password of the cluster.
<code>-o KIND, --output=KIND</code>	Type of document: JSON or standard.
<code>-d, --debug</code>	Output debug information.
<code>-h, --help</code>	Help information.
<code>-s, --ssl</code>	Uses SSL for communication with secure servers.

Abbildung 6: OPTIONS bei einem CLI Command [COU.10]

Falls mehrere Befehle ausgeführt werden sollen gibt es auch die Möglichkeit Username und Passwort in Umgebungsvariablen zu speichern.

Diese lauten: `CB_REST_USERNAME` und `CB_REST_PASSWORD`.

```
export CB_REST_USERNAME=Administrator
```

In Form dieser Syntax kann dann eine Umgebungsvariable gesetzt werden.

[COU.9]

Mithilfe der couchbase-cli kann man Buckets erstellen, löschen sowie verändern. Außerdem können an Clustern Änderungen vorgenommen werden sowie das mitloggen von Informationen gestartet und gestoppt werden. Die couchbase-cli ist generell dazu da, um Einstellungen vorzunehmen.

Die Befehle die auf Buckets ausgeführt werden können lauten hier z.B.: `bucket-compact`, `bucket-create`, `bucket-delete`, `bucket-edit`, `bucket-flush`, `bucket-list`

Compact formatiert Files im Bucket um Platz zu schaffen. Create, delete und edit tun genau das, was ihr Name sagt. Bei Create werden Argumente die zum Erstellen eines Buckets benötigt werden über `--<Argument> <Wert>` übergeben. Mithilfe von Edit können diese Werte dann bearbeitet werden. Flush löscht alle Daten aus einem Bucket und List listet alle im Cluster vorhandenen Buckets auf.

Mithilfe der CLI können noch einige weitere Einstellungen getroffen werden wie Server hinzuzufügen, LDAP Einstellungen oder auch Indexierungseinstellungen und Replikationseinstellungen.

[COU.10]

Möchte man über die CLI Abfragen starten muss man dafür cpq benutzen. Dieses Tool befindet sich in Linux auch unter:

```
/opt/couchbase/bin
```

Mithilfe von ./cpq wird es für die lokale Node gestartet. Möchte man sich zu einer anderen Node verbinden kann man diese mithilfe eines Arguments angeben:

```
-engine=http://123.45.67.89:8093
```

Abfragen können hier mithilfe von N1QL Querys durchgeführt werden. Diese sind aber nicht sehr schnell, da sie vor dem Ausführen zuerst einmal indexiert werden müssen.

[COU.11]

Möchte man Couchbase von der CLI effektiver nutzen gibt es cbc. Cbc ist ein Teil der C SDK. Um cbc nutzen zu können sind für die Installation ein paar Schritte notwendig. Die auf der Website angegebenen Wege über ein perl skript sowie über dpkg -i haben nicht funktioniert. Dafür mussten die apt Repositories manuell konfiguriert werden. Zuerst musste der Couchbase GPG Key heruntergeladen und hinzugefügt werden:

```
wget http://packages.couchbase.com/ubuntu/couchbase.key
```

```
apt-key add couchbase.key
```

Als nächstes muss in /etc/apt/sources.list.d ein couchbase.list File erstellt werden. Diesem fügt man dann folgende Zeile hinzu:

```
deb http://packages.couchbase.com/ubuntu wheezy wheezy/main
```

[COU.12]

deb steht hier für binäre Pakete. Eine andere Möglichkeit ist deb-src. Das steht für den Quellcode der Pakete, falls man diese nachträglich kompilieren möchte. Als nächstes wird der Ort angegeben von dem das Paket installiert wird. Hier kann außer einem Server auch eine lokale Adresse oder eine CD angegeben werden. Als letztes wird die Distribution angegeben. [WDE]

Dann kann das benötigte Package mittels:

```
apt-get install libcouchbase2-bin
```

installiert werden.

Mithilfe von cbc kann man Buckets erstellen, löschen, CRUD Operationen durchführen und auch Administrative Aufgaben erledigen.

Der grundsätzliche Aufbau eines cbc Befehls sieht so aus:

```
cbc <command> -U couchbase://ip-adresse
```

Create/Update

Mit cbc-create kann ein Json Dokument erstellt werden

```
cbc-create weberbrew -V '{"json":"test"}' -M upsert -U  
couchbase://192.168.187.137/beer-sample
```

Mit -V wird der zu benutzende Wert angegeben. -M ist die Art wie es gespeichert werden soll. <upsert|insert|replace>. Insert würde einen Fehler liefern wenn das Dokument schon vorhanden ist. -U gibt die credentials an.

Read

Mit cbc-cat können JSON Dokumente auch ausgelesen werden.

```
cbc-cat weberbrew -U couchbase://192.168.187.137/beer-sample  
  
martin@martin:~$ cbc-cat weberbrew -U couchbase://192.168.187.137/beer-sample  
weberbrew          CAS=0x7b6ceaac4b14, Flags=0x0. Size=15  
{ "json": "test" }  
martin@martin:~$ █
```

Delete

Um Dokumente wieder zu entfernen kann cbc-rm benutzt werden

```
cbc-rm weberbrew -U couchbase://192.168.187.138/beer-sample
```

2.3 API Installation und Verwendung

Da wir mit Java die meiste Erfahrung sammeln konnten, wird die Installation und Verwendung der Java SDK beschrieben.

Zuerst sollte man sich unter dem Punkt „Overview“ die passende SDK Version zur Couchbase Version suchen.

Unter „Download and API reference“ findet man die gewünschte SDK Version als Download, sowie die zugehörige API. Zur Installation kann man entweder ein Maven Projekt verwenden und die Dependencies wie beschrieben eintragen oder man lädt ein Archiv herunter und bindet die JAR Files in den Build Path ein. [COU.8]

Verbindung

```
Cluster cluster = CouchbaseCluster.create("192.168.38.140");  
Bucket bucket = cluster.openBucket();
```

Zuerst muss man sich mit dem Cluster verbinden und danach kann man einen Bucket öffnen. Gibt man keine Argumente bei dem Cluster an, so wird eine Verbindung zu localhost aufgebaut. In einer produzierenden Umgebung sollte man mindestens zwei IP Adressen angeben, damit im Fehlerfall auf einer anderen Node weitergearbeitet werden kann. Dies geschieht automatisch mit den mitgelieferten Libraries, sodass sich der Programmierer nicht darum kümmern muss. Mehrerer IP-Adressen werden entweder als Liste oder als mehrere Parameter angegeben. [COU.8]

Da bei den IP Adressen ein Reverse DNS Lookup durchgeführt wird, sollte man einen Hosts Eintrag am ausführenden Betriebssystem anlegen. Ansonsten dauert das Lookup, vor allem auf Windows Clients, länger als 5 Sekunden, was der Standard-Timeout Wert ist. Deshalb schlägt die Verbindung fehl. Möglich wäre es auch das Timeout zu verändern, jedoch ist der Eintrag im Hosts File praktischer, da die Verbindung auch schneller aufgebaut wird. [HUB]

Danach lässt sich ein Bucket öffnen. Ohne angegebene Argumente öffnet man den Default Bucket. Man kann den zu öffnenden Bucket auch auswählen, sowie ein eventuelles Passwort dafür spezifizieren. [COU.8]

Von CouchbaseCluster, Bucket und ClusterEnvironment sollten jeweils nur eine Instanz erstellt werden.

Das ganze lässt sich natürlich auch asynchron durchführen. Grundsätzlich ist die gesamte API asynchron umgesetzt, die synchronen Methoden sind lediglich Wrapper um die asynchronen Methoden.

```
AsyncCluster cluster = CouchbaseAsyncCluster.create();  
Observable<AsyncBucket> bucketObservable = cluster.openBucket();
```

Abbildung 7: Asynchrone Verbindung [COU.8]

Beim Verbindungsabbau kann entweder der ganze Cluster geschlossen werden, was gleichzeitig alle geöffneten Buckets schließt, oder es werden die Buckets einzeln geschlossen, damit mit dem Cluster noch weitergearbeitet werden kann. [COU.8]

```
System.out.println("disconnect");  
cluster.disconnect();
```

Administratives

Mit einem ClusterManager lassen sich administrative Aufgaben auch über die Java SDK ausüben.

```
ClusterManager clusterManager = cluster.clusterManager("Administrator",
"secretpassword");
    for (BucketSettings bs : clusterManager.getBuckets())
        System.out.println(bs);
```

Beispielsweise kann man die Einstellungen eines Buckets auslesen oder verändern. Dieser Code erzeugt zum Beispiel folgenden Output:

```
DefaultClusterBucketSettings{name='beer-sample', type=COUCHBASE, quota=100, port=0, password='', replicas=1, indexReplicas=false, enableFlush=false}
DefaultClusterBucketSettings{name='default', type=COUCHBASE, quota=2355, port=0, password='', replicas=1, indexReplicas=false, enableFlush=false}
```

Abbildung 8: BucketSettings

Folgende Möglichkeiten bestehen mittels ClusterManager:

Method	Description
<code>info</code>	Provides cluster information
<code>getBuckets</code>	Lists all buckets with their settings from the cluster
<code>getBucket</code>	Gets a bucket with its settings from the cluster
<code>hasBucket</code>	Checks if a bucket exists or not on the cluster
<code>insertBucket</code>	Creates a new bucket on the cluster
<code>updateBucket</code>	Updates a bucket on the cluster
<code>removeBucket</code>	Removes a bucket from the cluster

Abbildung 9: Methoden des ClusterManagers

Sollten Standardwerte bei Timeouts und ähnlichen Konfigurationen nicht ausreichen, benötigt man eine CouchbaseEnvironment.

```
CouchbaseEnvironment env =
DefaultCouchbaseEnvironment.builder().connectTimeout(10000).build();
Cluster cluster = CouchbaseCluster.create(env, "192.168.38.140");
```

Soll zum Beispiel die Zeitdauer des Connection Timeouts statt 5 Sekunden 10 sein, konfiguriert man die Umgebung wie oben beschrieben und gibt die Umgebung dann an die create Methode weiter. [COU.8]

Create

Mit der SDK können Buckets und Dokumente erzeugt werden.

Für einen Bucket muss man zuerst alle Einstellungen definieren und dann kann man ihn erstellen.

```
BucketSettings bucketSettings = new
DefaultBucketSettings.Builder().type(BucketType.COUCBASE)
.name("famous-persons").password("")
.quota(100).replicas(1).indexReplicas(false).enableFlush(false).build();
clusterManager.insertBucket(bucketSettings);
```

Anschließend verbindet man sich zu diesem Bucket und kann auch schon Dokumente hochladen.

```
JsonObject user = JsonObject.empty().put("firstname", "Walter").put("lastname",
"White")
                .put("job", "chemistry teacher").put("age", 50);
JsonDocument doc = JsonDocument.create("walter", user);
JsonDocument response = bucket.upsert(doc);
```

Zuerst erstellt man das gewünschte JSON Objekt oder Array. Danach erstellt man daraus ein JSON Dokument mit einer ID und dem Inhalt. Mit insert oder upsert lassen sich die Daten hochladen. Insert liefert jedoch einen Fehler, wenn das Dokument bereits vorhanden ist. [COU.8]

Read

Wie bereits vorhin erwähnt lassen sich Informationen wie die verfügbaren Buckets usw. über einen ClusterManager auslesen.

Die erste Variante des Lesens ist, ein ganzes Dokument herunterzuladen.

```
JsonDocument loadedFromId = bucket.get("batman");
System.out.println(loadedFromId.content());
```

Dabei spezifiziert man die ID des Dokuments und danach erhält man ein JSON Dokument.

Die nächste Variante ist, eine View zu verwenden.

```
ViewResult result = bucket.query(ViewQuery.from("dev_firstnames",
"firstnames"));
for (ViewRow row : result)
    System.out.println(row);
```

In einer Java Client Applikation wird dies die häufigste Variante sein, da sie die schnellen, im Vorhinein erstellten, Views verwendet.

Danach gibt es noch die Möglichkeit einer N1QL Query. Diese ist wie SQL aufgebaut, jedoch ist es eine langsame Variante Daten abzufragen. Bevor eine solche Query überhaupt abgesetzt werden kann muss ein Index erstellt werden.

```
bucket.query(N1qlQuery.simple("CREATE PRIMARY INDEX ON `famous-persons` USING GSI"));
```

Danach kann eine Query abgesetzt werden. Diese kann entweder als reiner Text oder mittels eigenen Methoden erstellt werden. Mit Text:

```
N1qlQueryResult queryResult = bucket.query(N1qlQuery.simple("SELECT job FROM `famous-persons`"));
for (N1qlQueryRow row : queryResult)
    System.out.println(row);
```

Oder mit Hilfsmethoden:

```
N1qlQueryResult queryResult2 = bucket.query(select("age").from("`famous-persons`"));
for (N1qlQueryRow row : queryResult2)
    System.out.println(row);
```

Dazu wird der statische Import

(com.couchbase.client.java.query.Select.select) benötigt. [COU.8]

Update

Wie bereits vorhin erwähnt lassen sich Einstellungen von Buckets über einen ClusterManager verändern.

Bereits erwähnt wurde ebenfalls die `upsert` Methode, die ein Dokument auf dem Server überschreibt oder auch neu hinzufügt. Zusätzlich gibt es noch die `replace` Methode, die einen Fehler wirft, sollte das Dokument noch nicht vorhanden sein. [COU.8]

Delete

Möglich ist das Löschen einzelner Dokumente:

```
JsonDocument doc = bucket.remove("batman");
```

Dazu gibt man lediglich eine ID an. Außerdem lassen sich mit dem ClusterManager ganze Buckets löschen.

```
clusterManager.removeBucket("famous-persons");
```

Dazu muss man auch nur den Namen des Buckets angeben. [COU.8]

2.4 Dokumentenstruktur

In Couchbase gibt es Buckets, diese entsprechen Datenbanken in RDBMS. In diesen Buckets befinden sich Dokumente, die je nach Modellierung Tabellen oder Reihen darstellen können. Diese Dokumente werden in JSON (JavaScript Object Notation) gespeichert.

JSON kennt zwei verschiedene Strukturen:

- Objekte
- Arrays

Objekte werden durch { } gekennzeichnet und sind eine Menge aus Name/Werte Paaren (key-value). Name und Wert werden durch einen Doppelpunkt getrennt, sowie die Paare voneinander mit Beistrichen getrennt werden.

Arrays sind Listen von Werten. Das heißt es sind keine Namen enthalten und die Werte werden nur durch Beistriche getrennt.

Ein Wert selbst kann ein Objekt ein Array, ein String, eine Zahl, true, false oder null sein.

Ein String besteht aus keinen oder mehreren Unicode Zeichen und zeichnet sich durch doppelte Anführungszeichen aus.

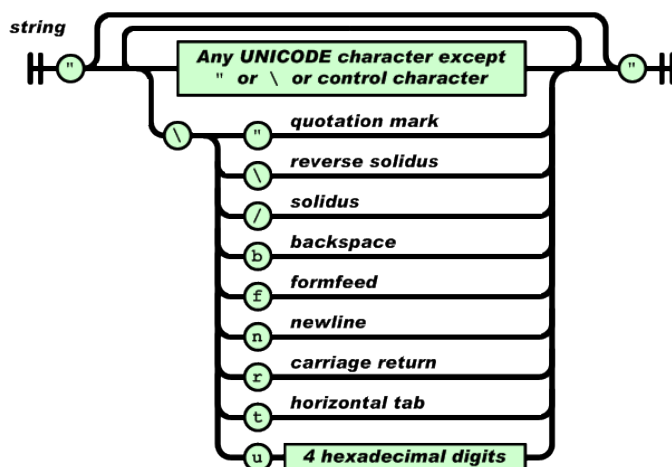


Abbildung 10: Sonderzeichen [JSO]

Zahlen können mit negativen Vorzeichen, oder ohne, mit einem Punkt als Dezimalzeichen und mit e (oder E) [Darstellung als 10-er Potenz] dargestellt werden. [JSO]

Ein gültiges JSON Dokument sieht zum Beispiel so aus:

```
{
  "firstname": "Bruce",
  "job": "batman",
  "age": 42,
  "lastname": "Wayne"
}
```

Abbildung 11: JSON Objekt mit 4 Schlüssel/Werte Paaren

Couchbase kann viele Arten von Dokumenten speichern, es ist sogar möglich eine eigene Implementierung eines Dokuments zu erstellen. Trotz all dem sollte als Speicherart ein JSON Dokument, mit einem Objekt als „root“ Eintrag, verwendet werden. [COU.8]

Jedes JSON Dokument hat beim Erstellen mittels SDK mehrere Eigenschaften:

- id: eindeutige ID pro Bucket
- content: eigentlicher Inhalt des Dokuments
- cas: Compare And Swap Wert des Dokuments
- expiry: Auslaufdatum des Dokuments
- mutationToken: optionaler Token nach einer Mutation

Unter dem Punkt Working with Documents → Document basics [COU.8] sind die unterstützten JSON Dokumente aufgelistet. Außerdem unterstützt Couchbase Dokumente wie BinaryDocument, welches binäre Daten speichern kann. Auch Java Objekte, die serialisierbar sind, können mittels SerializableDocument gespeichert werden. [COU.8]

2.5 Indizierung & Map-Reduce

In Couchbase gibt es grundsätzlich 3 Typen von Indexen.

- MapReduce Views
- Spatial Views
- Global Secondary Index (GSI)

MapReduce

MapReduce views werden oft auch einfach nur views genannt. Sie sind JavaScript Funktionen die für Dokumente in einem Bucket definiert werden. Diese views werden inkrementell aufgebaut. Das heißt kommt ein neues Dokument hinzu oder wird ein Dokument verändert werden nur die neuen/veränderten Dokumente reindexiert. Der Couchbase Server berechnet die Ergebnisse einer view im Vorhinein und speichert diese ab um bei einer Clientanfrage die Ergebnisse direkt zur Verfügung stellen zu können. Jede Node in einem Cluster besitzt einen eigenen partiellen Index für diese View. Wird

eine Query an eine View geschickt, schicken die einzelnen Nodes ihre Daten. Diese werden dann zu einem gesammelten Set zusammengesetzt.

Bei MapReduce views gibt es 2 Funktionen. Map und Reduce

```
Map
1 function(doc, meta) {
2   switch(doc.type) {
3     case "brewery":
4       emit([meta.id]);
5       break;
6     case "beer":
7       if (doc.brewery_id) {
8         emit([doc.brewery_id, meta.id]);
9       }
10      break;
11    }
12  }
13 }
```

Abbildung: 12 Map Funktion

Zuerst wird die Gesamtmenge auf eine Untermenge eingeschränkt durch die Map Funktion. Diese Untermenge besteht aus key-value Paaren. Hier wird zum Beispiel der Bucket beer-sample Durchsucht. Ist Das Dokument vom typ brewery wird die meta-ID ausgegeben. Ist es ein „beer“ wird die brewery_id vom Dokument und die meta-ID ausgegeben.

```
Reduce
1 _count
```

Abbildung 13: Reduce

In Reduce wird dann über diese Untermenge noch eine Funktion ausgeführt. Couchbase bietet hier ein paar built-in Funktionen (`_sum`, `_count`, `_stats`). Man kann jedoch auch seine persönliche reduce Funktion schreiben.

[COU.15]

Spatial View

Die Spatial view hat nur eine einzelne Funktion, die spatial function. Diese sind laut couchbase geeignet für multidimensionale Analysen sowie geographische Daten oder eine Kombination von beiden. Das GeoJSON Format (Format für geographische Datenstrukturen) wird unterstützt. [COU.16]

Global Secondary Index (GSI)

Diese Indexierungsart wird für N1QL benötigt. Dabei wird der Befehl CREATE INDEX benutzt.

Der GSI erstellt B+ Bäume um ein schnelles Suchen nach dem index key zu ermöglichen. [COU.17]

Unter diesem Link findet man noch eine Tabelle mit Unterschieden zwischen Views und dem GSI:

<http://developer.couchbase.com/documentation/server/4.1/architecture/gsi-versus-views.html>

2.6 Erstellung und Verwendung von eigenen Views

Views können ganz einfach erstellt werden in dem man eine JavaScript Funktion definiert. Ist man im Webinterface kann man unter dem Punkt Views vorhandene Views ansehen und bearbeiten und neue Views erstellen.



Abbildung 14 Views

Soll nun eine View erstellt werden ist sie zuerst eine Development View. Wie schon vorher erklärt kann man hier 2 Arten sehen. View (MapReduce) und Spatial View. Der Unterschied zwischen einer Development View und einer Production View ist folgender: Die Production View führt eine Indexierung für den kompletten Bucket durch, wobei die Development View nur zum Testen da ist und nur auf ein Subset der Gesamtmenge ausgeführt wird.

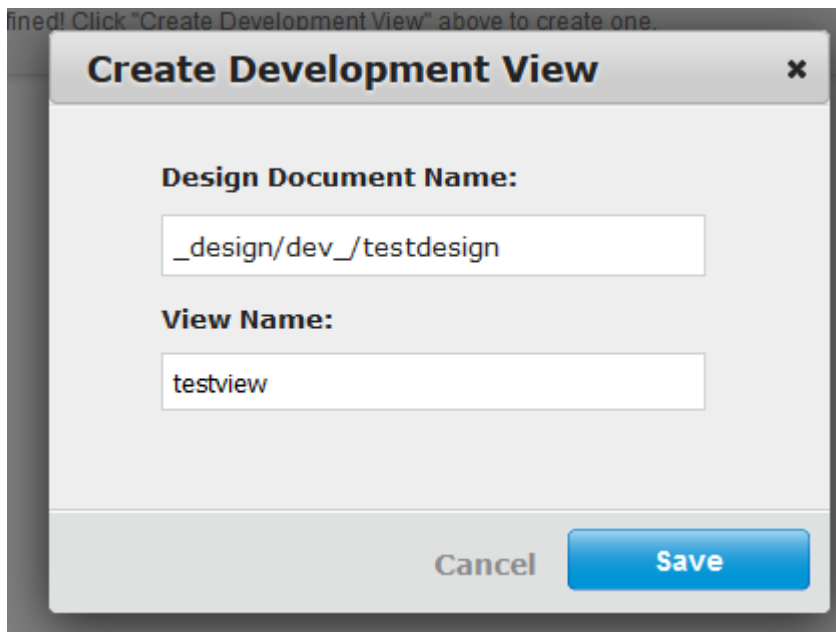


Abbildung 15: Development View erstellen

Erstellt man nun eine View muss man den Design Document Name und den Namen der View angeben.

Dann kann man die View auch schon bearbeiten.

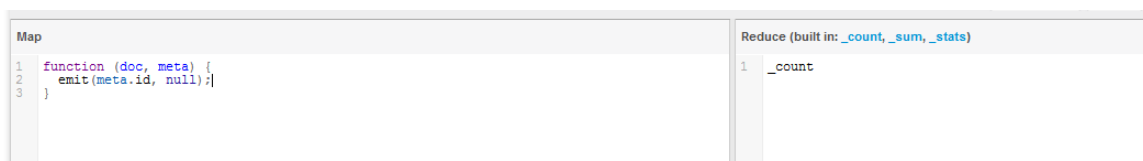


Abbildung 16: eine Testview

Diese View würde nur die Metainformationen zu allen Dokumenten im beer-sample Bucket ausgeben wenn die Reduce Funktion leer wäre. Mit der Reduce Funktion erhält man aber die Anzahl der Dokumente im Bucket.

Development Time Subset Full Cluster Data Set	
Key	Value
"21st_amendment_brewery_cafe" 21st amendment brewery cafe	null
"21st_amendment_brewery_cafe-21a_ipa" 21st amendment brewery cafe-21a ipa	null
"21st_amendment_brewery_cafe-563_stout" 21st amendment brewery cafe-563 stout	null
"21st_amendment_brewery_cafe-amendment_pale_ale" 21st amendment brewery cafe-amendment pale ale	null
"21st_amendment_brewery_cafe-bitter_american" 21st amendment brewery cafe-bitter american	null
"21st_amendment_brewery_cafe-double_trouble_ipa" 21st amendment brewery cafe-double trouble ipa	null
"21st_amendment_brewery_cafe-general_pippo_s_porter" 21st amendment brewery cafe-general pippo s porter	null
"21st_amendment_brewery_cafe-north_star_red" 21st amendment brewery cafe-north star red	null
"21st_amendment_brewery_cafe-oyster_point_oyster_stout" 21st amendment brewery cafe-oyster point oyster stout	null
"21st_amendment_brewery_cafe-potrero_esb" 21st amendment brewery cafe-potrero esb	null

Abbildung 17: View mit leerer Reduce Funktion

Im key stehen hier die Informationen zum Dokument und in Value steht null sowie in der Funktion definiert. Außerdem kann man hier noch einmal sehen, dass die Query nur auf ein Subset durchgeführt wurde, aber auch auf das komplette Datenset durchgeführt werden kann.

Key	Value
null undefined	7303

Abbildung 18: Ergebnis der View

Hier sieht man das Ergebnis mit Reduce Funktion. Value ist hier die Anzahl der Dokumente.

3 GitHub Link

<https://github.com/dmay-tgm/Couchbase-Basics>

4 Zeitmanagement

Daniel May übernimmt die Installation, API Installation inkl. Verwendung und die Dokumentenstruktur. Martin Weber übernimmt die CLI Befehle, Indizierung und Map/Reduce, sowie die Erstellung und Verwendung von Views. Korrekturen werden jeweils auch am anderen Teil durchgeführt.

Der Aufwand wird wie folgt geschätzt:

Arbeitsteil	Durchführender	geschätzter Aufwand
Installation	Daniel May	0.5 h
CLI Befehle	Martin Weber	1.5 h
API Installation und Verwendung	Daniel May	1.5 h
Dokumentenstruktur (JSON, GSON)	Daniel May	0.5 h
Indizierung & Map/Reduce	Martin Weber	0.5 h
Erstellung und Verwendung von Views	Martin Weber	1 h
Korrekturarbeiten	D. May & M. Weber	0.5 h/Person
Gesamt	D. May & M. Weber	6.5 h

Der tatsächliche Aufwand ist wie folgt aufgeschlüsselt:

Daniel May:

Durchgeführte Arbeit	Datum	tatsächlicher Aufwand
Installation	25.04.2016	1.25 h
API Installation und Verwendung	01.05.2016	3 h
Dokumentenstruktur (JSON, GSON)	01.05.2016	0.5 h
Korrekturen	01. & 05.05.2016	0.5 h
Gesamt	25.04. – 05.05.2016	5.25 h

Martin Weber:

Durchgeführte Arbeit	Datum	tatsächlicher Aufwand
CLI Befehle	25.04. – 05.05.2016	3h
Indizierung & Map/Reduce	25.04. – 05.05.2016	1.5h
Erstellung und Verwendung von Views	25.04. – 05.05.2016	1.5h
Korrekturen	25.04. – 05.05.2016	0.1h
Gesamt	25.04. – 05.05.2016	6.1h

Gesamt:

Durchführender	tatsächlicher Aufwand
Daniel May	5.25 h
Martin Weber	6.1h
Gesamt	11.35 h

5 Literaturverzeichnis

- [BOR] Michael Borko.
Couchbase [Online]. Available at:
<https://elearning.tgm.ac.at/mod/assign/view.php?id=40870>
[abgerufen am 05.05.2016]
- [COU.1] Couchbase.
Downloads [Online]. Available at:
<http://www.couchbase.com/nosql-databases/downloads>
[abgerufen am 05.05.2016]
- [COU.2] Couchbase (Version 4.1).
Quick installation and setup [Online]. Available at:
<http://developer.couchbase.com/documentation/server/4.1/getting-started/installing.html>
[abgerufen am 05.05.2016]
- [COU.3] Couchbase (Version 4.1).
Transparent Huge Pages (THP) [Online]. Available at:
<http://developer.couchbase.com/documentation/server/4.1/install/thp-disable.html>
[abgerufen am 05.05.2016]
- [COU.4] Couchbase (Version 4.1).
Swap space and kernel swappiness [Online]. Available at:
<http://developer.couchbase.com/documentation/server/4.1/install/install-swap-space.html>
[abgerufen am 05.05.2016]
- [COU.5] Couchbase (Version 4.1).
Initial server setup using UI [Online]. Available at:
<http://developer.couchbase.com/documentation/server/4.1/install/init-setup.html#topic12527>
[abgerufen am 05.05.2016]
- [COU.6] Couchbase (Version 4.1).
Buckets [Online]. Available at:
<http://developer.couchbase.com/documentation/server/4.1/architecture/core-data-access-buckets.html>
[abgerufen am 05.05.2016]

- [COU.7] Couchbase (Version 4.1).
View replication [Online]. Available at:
<http://developer.couchbase.com/documentation/server/4.1/indexes/mapreduce-view-replication.html>
[abgerufen am 05.05.2016]
- [COU.8] Couchbase (Version 4.1).
Java SDK 2.2 (inkl. Unterpunkte) [Online]. Available at:
<http://developer.couchbase.com/documentation/server/4.1/sdks/java-2.2/java-intro.html>
[abgerufen am 05.05.2016]
- [COU.9] Couchbase (Version 4.1).
CLI reference [Online] Available at:
<http://developer.couchbase.com/documentation/server/4.1/cli/cli-intro.html>
[abgerufen am 05.05.2016]
- [COU.10] Couchbase (Version 4.1).
couchbase-cli [Online] Available at:
<http://developer.couchbase.com/documentation/server/4.1/cli/cbcli-intro.html>
[abgerufen am 05.05.2016]
- [COU.11] Couchbase (Version 4.1).
cbq [Online] Available at:
<http://developer.couchbase.com/documentation/server/4.1/cli/cbq-tool.html>
[abgerufen am 05.05.2016]
- [COU.12] Couchbase (Version 4.1).
Installation and API Reference [Online] Available at:
<http://developer.couchbase.com/documentation/server/current/sdks/c-2.4/download-install.html>
[abgerufen am 05.05.2016]

[COU.13] Couchbase (Version 4.1).

Accessing data from a command line [Online] Available at:

<http://developer.couchbase.com/documentation/server/current/developer-guide/cli-overview.html>

[abgerufen am 05.05.2016]

[COU.14] Couchbase C Client (Version 2.4.0).

Couchbase Client Commandline Utility [Online] Available at:

http://docs.couchbase.com/sdk-api/couchbase-c-client-2.4.0/md_cbc.html

[abgerufen am 05.05.2016]

[COU.15] Couchbase MapReduce (Version 2.4.0).

Querying using MapReduce views [Online] Available at:

<http://developer.couchbase.com/documentation/server/4.1/indexes/querying-using-map-reduce-views.html>

[abgerufen am 05.05.2016]

[COU.16] Couchbase Spatial View (Version 2.4.0).

Querying using spatial views [Online] Available at:

<http://developer.couchbase.com/documentation/server/current/indexes/querying-using-spatial-views.html>

[abgerufen am 05.05.2016]

[COU.17] Couchbase GSI (Version 2.4.0).

Global Secondary Indexes (GSIs) [Online] Available at:

<http://developer.couchbase.com/documentation/server/4.1/architecture/global-secondary-indexes.html>

[abgerufen am 05.05.2016]

[WDE] SourcesList [Online] Available at:

<https://wiki.debian.org/SourcesList>

[abgerufen am 05.05.2016]

[HUB] hubo3085632 (September 2015).

JavaClient 2.x connection`s warning [Online]. Available at:

<https://forums.couchbase.com/t/javaclient-2-x-connection-s-warning/5227>

[abgerufen am 05.05.2016]

[JSO] JSON.
Introducing JSON [Online].
Available at: <http://www.json.org/>
[abgerufen am 05.05.2016]