Fachhochschule Köln
Cologne University of Applied Sciences

Institut für Medien- und Phototechnik

Bachelorarbeit Medientechnik

Ein Vergleich eines relationalen mit einem non-relationalen Datenbanksystems (MySQL / MongoDB)

vorgelegt von

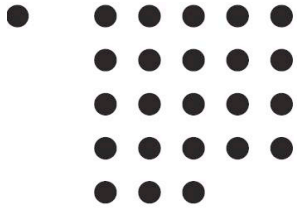
Philipp Ständer

Mat.-Nr. 11052163

Erstgutachter: Prof. Dr. phil. Gregor Büchel (Fachhochschule Köln)

Zweitgutachter: Prof. Dr. rer. nat. Stefan Grünvogel (Fachhochschule Köln)

April 2011



Fachhochschule Köln
Cologne University of Applied Sciences

Institut für Medien- und Phototechnik

Bachelor Thesis

Comparison of a relational and a non-relational database system (MySQL / MongoDB)

submitted by

Philipp Ständer

Mat.-Nr. 11052163

First Reviewer: Prof. Dr. phil. Gregor Büchel (Fachhochschule Köln)

Second Reviewer: Prof. Dr. rer. nat. Stefan Grünvogel (Fachhochschule Köln)

April 2011

Bachelorarbeit

Titel: Ein Vergleich eines relationalen mit einem non-relationalen Datenbanksystems (MySQL / MongoDB)

Gutachter:

Prof. Dr. phil. Gregor Büchel (Fachhochschule Köln)

Prof. Dr. rer. nat. Stefan Grünvogel (Fachhochschule Köln)

Zusammenfassung: Bei dem Vergleich werden die Besonderheiten, Vor- und Nachteile sowie der Anwendungszweck beider Datenbanksysteme erläutert und anhand der Wissensdatenbank Wikipedia angewendet. Der abschließende Geschwindigkeitstest soll einen praxisnahen Vergleich liefern und die unterschiedlichen Ansätze beider Datenbanksysteme verdeutlichen.

Stichwörter: Datenbank, Dokumentenorientierte Datenbank, Wikipedia, NoSQL

Datum: 14. April 2011

Bachelor Thesis

Title: Comparison of a relational and a non-relational database system (MySQL / MongoDB)

Reviewers:

Prof. Dr. phil. Gregor Büchel (Fachhochschule Köln)

Prof. Dr. rer. nat. Stefan Grünvogel (Fachhochschule Köln)

Abstract: By comparing the characteristics, advantages, disadvantages and the designed purposes, the comparable merits of both database systems can be explained. Based on information obtained from the online knowledge base 'Wikipedia' a clear comparison can be made between the two. The final benchmarktest is intended to provide a practical comparison and illustrate the different approaches of both database systems.

Keywords: database, document-oriented database, Wikipedia, NoSQL

Date: 14. April 2011

Inhaltsverzeichnis [1]

[1] Aufgrund einer Einschränkung der hier verwendeten Textverarbeitung ist es nicht möglich, den Anhang und das Quellenverzeichnis im Inhaltsverzeichnis mit Buchstaben zu nummerieren

Einleitung

Datenbanken werden im Computeralltag mittlerweile flächendeckend eingesetzt, u.a. bei:

- Desktopapplikationen [2]
- Webapplikationen
- Enterprisenanwendungen [3] / Geschäftsbetrieb
- Privatnutzer [4]

So ist es auch nicht verwunderlich, dass sich mit der gestiegenen Anzahl von Einsatzfeldern auch die Nachfrage zu neuen, flexibleren Datenbanksystemen geändert hat. Die NoSQL-Bewegung ist - grob zusammengefasst - eine von diesen. Der Name NoSQL steht für not-only-SQL und bedeutet in der Regel, dass neue Ansätze bezogen auf die API, Datenverwaltung, Verfügbarkeit, Skalierbarkeit und Geschwindigkeit der Datenbanksysteme verfolgt werden.

In der Bachelorarbeit möchte ich ein herkömmliches relationales Datenbanksystem (MySQL) mit einem neunen, nicht-relationalen Datenbanksystem (MongoDB) vergleichen. Beide Datenbanken sind über Open-Source-Lizenzen verfügbar, haben eine große Nutzerzahl (vor allem im Webapplikationsbereich), werden jeweils von einer Firma betreut/vermarktet und werden aktiv weiterentwickelt. Damit soll sichergestellt sein, dass wir hier zwei zeitgemäße Datenbanksysteme vergleichen.

Als Datenmaterial wird die Wikipedia verwendet, da sie zum einen strukturell für eine NoSQL-Datenbank prädestiniert ist und bis heute offiziell auf einem relationalen Datenbanksystem (MySQL) betrieben wird. Zudem liegen die Datenmengen mit über 2 Millionen Einträgen (entsprechend 7GB an Datenvolumen) in einer „businesstauglichen“ Größenordnung und sind frei verfügbar [5].

Als non-relationale Datenbank kommt die MongoDB zum Einsatz. Sie ist eine dokumentenbasierte Datenbank mit Eigenschaften einer herkömmlichen relationalen Datenbank und deshalb ein adäquates Gegenstück zu MySQL. Andere populäre NoSQL-Vertreter, wie z.B. CouchDB [6], sind in ihren Ansätzen und Unterschieden wesentlich radikaler. So besitzt die MongoDB ebenfalls Hauptschlüssel, Feldindexierung und bietet auch die Möglichkeit Datensätze direkt miteinander verknüpfen [7].

---

[2] z.B. http://wiki.apache.org/couchdb/CouchDB_in_the_wild

[3] z.B. http://glassfish.java.net/

[4] z.B. http://www.filemaker.de/

[5] http://de.wikipedia.org/wiki/Wikipedia:Download#Herunterladen_aller_Seiten_als_XML-Dump

[6] http://couchdb.apache.org/

[7] http://stackoverflow.com/questions/4992648/many-to-many-update-in-mongodb-without-transactions

Beim Importvorgang wird für beide Datenbanksysteme jeweils eine angepasste Struktur gewählt, um den abschließenden Geschwindigkeitsvergleich so fair wie möglich zu gestalten.

Das Ergebnis der Arbeit soll ein übersichtlicher Vergleich beider Datenbanksysteme sein. Dabei geht es nicht um das theoretische Aufzählen von Vor- und Nachteilen der jeweiligen Datenbank, sondern um einen praxisnahen Anwendungsfall, der von beiden Datenbanken bewältigt werden muss. Zudem soll gezeigt werden, dass es durchaus Sinn macht, für bestimmte Anwendungszwecke spezielle Datenbanksysteme zu verwenden.

Philipp Ständer
Berlin, den 14. April 2011

1. Das allgemeine Datenbankmodell [8]

Eine Datenbank (DB) ist die Menge aller gespeicherten Daten eines Datenbanksystems. **Ein Datenbankmanagementsystem (DBMS)** ist die Menge der Software zum Verwalten von gespeicherten Daten in einem Datenbanksystem.

1.1. Persistenz

Jedes **Datenbankmanagementsystem (DBMS)** muss eine dauerhafte und strukturtreue Speicherung von Datenbeständen gewährleisten. Dieses Vorgang, Persistenz genannt, steht im Gegensatz zur flüchtigen Speicherung, wie zum Beispiel im Arbeitsspeicher. Die Datenbestände können auch komplexe Datenstrukturen besitzen.

1.2. Sekundärspeicherverwaltung

Die Sekundärspeicherverwaltung soll die Lese-Operationen auf Sekundärspeicher (z.B. Festplatte) minimieren um Datensätze schneller auszugeben. Durch die Sekundärspeicherverwaltung werden Direktzugriffe wie binäre Suchvorgänge oder anfügen von Datensätze beschleunigt.
Objektorientiertedatenbanksysteme (ODBs) nutzen zum Beispiel Sekundärspeicher, um persistente Objekte aus der **Objektorientierten Programmierung (OOP)** schneller zu speichern [9] .

1.3. Data-Dictionary

Das Data-Dictionary (Schemakatalog) gibt Metadaten aus, welche die Struktur und Änderung der Anwendungsdaten beschreiben. Die Metadaten geben nicht den Inhalt zurück.
Änderungen an der Struktur eines Data-Dictionaries erfolgt mit Hilfe einer Datendefinitionssprache (DDL), zu denen auch SQL zählt.

1.4. Interpretation einer Anfragesprache

Um Daten aus der DB zu lesen oder zu schreiben, wird eine Data Manipulation Language (DML) zur Verfügung gestellt, um Daten zu lesen, schreiben oder löschen. Bei dem relationalen Datenmodell hat sich die Structure Query Language (SQL) etabliert. Meistens wird die Datenabfragesprache (DQL = Data Query Language) als eigene Kategorie angesehen, da sie Daten ausgibt und nicht ändert.[10]

---

[8] In den Punkten Persistenz, Sekundärspeicherverwaltung und Data-Dictionary orientiere ich mich an dem Skript „Datenbanken und Algorithmen“ von Prof. Dr. Gregor Büchel, Stand 2005/2006

[9] Ramez Elmasri, Shamkant B. Navathe „Grundlagen von Datenbanksystemen“, 3. Auflage 2005, Seite 271

[10] http://de.wikipedia.org/wiki/Data_Manipulation_Language, http://de.wikipedia.org/wiki/Data_Definition_Language

1.5. Integrität

Die Integrität verhindert Inkonsistenzen in einer DB. Damit wird gewährleistet, dass die DB nur von einem korrekten Zustand in einen neuen korrekten Zustand überführt wird. Die Definitionen (Integritätsbedingungen) werden im Data-Dictionary festgelegt. So werden zum Beispiel korrekte Verknüpfungen von Tabellen über Primär- und Fremdschlüssel (PRIMARY / FOREIGN-Key) mit dem Begriff Integrität beschreiben.

1.6. Transaktionsverwaltung und das ACID-Modell

Aufeinanderfolgende Schreibzugriffe auf die DB werden in einer Transaktionsgruppe zusammengefasst und geloggt. Wenn nicht alle Zugriffe einer Gruppe ausgeführt werden, wird keiner ausgeführt bzw. die DB wird durch das loggen in den ursprünglichen Zustand zurückgesetzt. Eine Transaktionsgruppe soll die ACID-Kriterien erfüllen.

ACID steht für: [11]

- Atomicity (unzertrennliche Vollständigkeit)
- Consistency (Gewährleistung der Integrität)
- Isolation (Kapselung paralleler Transaktionen)
- Durability (Persistenz aller Änderungen)

Die Konsistenz sielt hierbei eine zentrale Rolle. Es bildet das Kernprinzip der relationalen Datenbankstruktur. Das ACID Prinzip stellt sicher, dass bei parallelen Zugriffen auf eine Datenbank stets die Konsistenz der Datensätze gewährleistet ist.

1.7. CAP-Theorem

Da Datenbanken heutzutage meist in verteilten Systemen eingesetzt werden und einem stetigen Wachstumsprozess unterworfen sind, ergeben sich Probleme bezüglich der (horizontalen) Skalierung. Bei relationalen Datenbanken, die dem ACID-Ansatz verfolgen, ist das schwer zu vereinbaren.

Das CAP-Theorem von Eric Brewer[12] (**C**onsistency **A**vailability und **P**artition Tolerance) besagt, dass verteilte Datenbanksysteme nicht die Anforderungen an Konsistenz, Verfügbarkeit und Ausfalltoleranz gleichermaßen genügen können, genau genommen können maximal zwei dieser Größen Erreicht werden [13].

---

[11] http://www.scribd.com/doc/30637338/ACID-vs-BASE-%E2%80%94-NoSQL-erklart

[12] http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf

[13] Stefan Edlich u.a. „NOSQL - Einsteig in die Welt nichtrelationaler ... Datenbanken“, Hanser 2010, Seite 31

1.7.1. Konsistenz (Consistency)

Alle verteilte Datenbanken sollen nach Abschluss einen konsistenten Zustand haben. Das bedeutet, dass (alle verteilten Datenbanken) bei nachfolgenden Lesezugriffen unmittelbar nach dem Schreibvorgang den aktuellen Wert zurückgeben.

1.7.2. Verfügbarkeit (Availability)

Die verteilten Datenbanken sollen eine akzeptable Reaktionszeit liefern.

1.7.3. Ausfalltoleranz (Partition Tolerance)

Falls ein Knoten eines verteilten Datenbanksystems ausfällt soll das System weiterhin verfügbar sein und auf Anfragen reagieren.

2. Aufbau einer relationalen Datenbank [14]

2.1. Die Geschichte von MySQL

Als relationales Datenbanksystem wird für den Vergleich MySQL verwendet. MySQL wurde 1994 ursprünglich als mSQL-Clone entwickelt [15] . Der Fokus wurde auf Verarbeitung von großen Datenmengen und hohe Performance gelegt - weniger auf Stabilität und Verfügbarkeit [16]. Bereits die frühe Version 3.23 von 1997 verfügte über eine Storage-Engine, die dem ACID-Kriterium genügt [17]. Die nachfolgenden Versionen erhielten unter anderem verbesserte Caching Funktionen [18], Unicode-Unterstützung[19] sowie vom SQL3-Standard definierte Objekttypen [20].
Aufgrund seines Open-Source-Lizenzmodells hat sich MySQL als Webdatenbank etabliert, wird aber auch von großen Webseiten wie Wikipedia [21] und anderen bekannten Firmen kommerziell eingesetzt [22].
Seit Januar 2010 gehört MySQL dem Datenbanksoftwarehersteller Oracle [23], ist aber weiterhin offiziell unter der GPL-Lizenz verfügbar. [24]

2.2. Storage-Engine MyISAM

MyISAM steht für „My Indexed Sequential Access Method“ und besitzt eine hohe Effizienz und besitzt eine leistungsfähige Volltextsuche [25]. MyISAM wird standardmäßig für MySQL-Tabellen verwendet [26], auch für den hier durchgeführten Geschwindigkeitsvergleich.

---

[14] Als relationale Datenbank wird MySQL der Firma Oracle verwendet

15 http://dev.mysql.com/doc/refman/5.1/de/history.html

16 http://de.wikipedia.org/wiki/MySQL

17 http://de.wikipedia.org/wiki/InnoDB

18 http://dev.mysql.com/doc/refman/4.1/en/news-4-0-x.html

19 http://dev.mysql.com/doc/refman/4.1/en/news-4-1-x.html

20 http://de.wikipedia.org/wiki/MySQL

[21] http://dev.mysql.com/tech-resources/articles/mysqluc-2007.html

22 http://www.mysql.de/customers/

23 http://www.oracle.com/us/corporate/press/044428

[24] http://www.mysql.de/about/legal/licensing/index.html

25 http://de.wikipedia.org/wiki/MyISAM

26 http://dev.mysql.com/doc/refman/5.1/de/myisam-storage-engine.html

2.3. Entity-Relationship-Modell [27]

Ein einfaches Attribut eines Entitätstyps ist mit einer Wertemenge assoziiert, die dem Attribut eine festgelegte Menge an Werten zuweist. So kann zum Beispiel bei einem Datensatz eines Buches der Titel als Attribut betrachtet werden, welcher als Menge alphanumerische Zeichenketten definiert hat.
Ein Attribut A von Entitätstyp E, der die Wertemenge V besitzt, kann als Funktion E zur Potenzmenge P(V) definiert werden:

$$A:E \longrightarrow P(V)$$

Wir verwenden den Wert von Attribut *A* für die Entität *e* als *A(e)* , was für einwertige, mehrwertige Attribute und Nullwerte (leere Menge) definiert ist. Für einwertige Attribute wird *A(e)* als Singleton für jedes *e* in *E* eingeschränkt, bei mehrwertigen Attributen nicht. Ein zusammengesetztes A besteht aus der Wertemenge V definiert als kartesisches Produkt von $P(V_n)$, wobei V_n die Wertemengen der einfachen Komponentenattribute ist, die A bilden

$$V = P(V_1) \times P(V_2) \times ... \times P(V_n)$$

Um das Entity-Relationship-Modell in das Relationen-Modell zu überführen, werden folgende Abbildungen definiert [28]:

Entitätstyp ⟶ Relation

Beziehungstyp ⟶ Fremdschlüssel (Foreign-Key)

Attribut ⟶ Attribut

2.3.1. Starke Entitätstypen

werden mit der folgenden Relation erstellt

$$R=\{a_1,a_2,...,a_n\} \cup k$$

wobei k : Primärschlüssel, a_n : Attribut der Entität

2.3.2. Schwache Entitätstypen

$$R=\{a_1,a_2,...,a_n\} \cup \{k\}$$

wobei

---

[27] Alle folgende mathematische Beschreibungen und Definitionen zu diesem Unterpunkt stammen aus:
Ramez Elmasri, Shamkant B. Navathe „Grundlagen von Datenbanksystemen“, 3. Auflage 2005, Seite 66ff

[28] http://de.wikipedia.org/wiki/Entity-Relationship-Modell

k : Fremdschlüssel

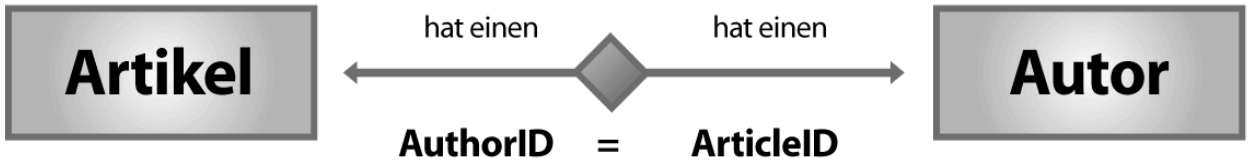
{k} ∪ {a_x}: Primärschlüssel

{a_x} identifiziert schwachen Entitätstyp, {k} identifiziert starken Entitätstyp

2.3.3. 1:1-Beziehungstypen

Ein Datensatz Typ A kann mit einem Datensatz Typ B in Relation stehen.
Wird eine der beiden Relationen um den Fremdschlüssel der jeweils anderen Relationen ergänzt.

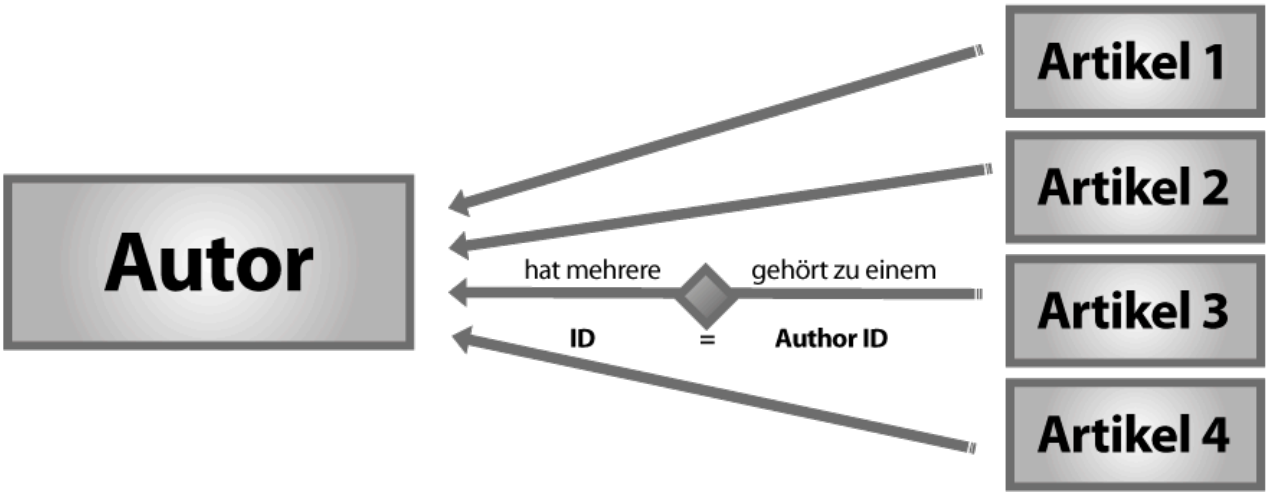
ArticleID	Title	AuthorID		AuthorID	Firstname	ArticleID
1	Lorem ipsum	*1*	↔	*1*	Alice	1
2	Dolor sit amet	*3*	↔	*3*	Chris	2
3	Consetetur	**2**	↔	**2**	Bop	*3*



2.3.4. 1:n-Beziehungstypen

Ein Datensatz Typ A kann mit mehreren Datensätzen Typ B in Relation stehen.
Wird die mit der Kardinalität N eingehende Relation T um den Fremdschlüssel S erweitert.

AuthorID	Firstname		ArticleID	Title	AuthorID
1	Alice	←	1	Lorem ipsum	*1*
2	Bob	↖	2	Dolor sit amet	*1*
3	Chris	↖	3	Consetetur	***2***



2.3.5. Foreign-Key-Integrität und Normalisierung (n:m Beziehungstyp)

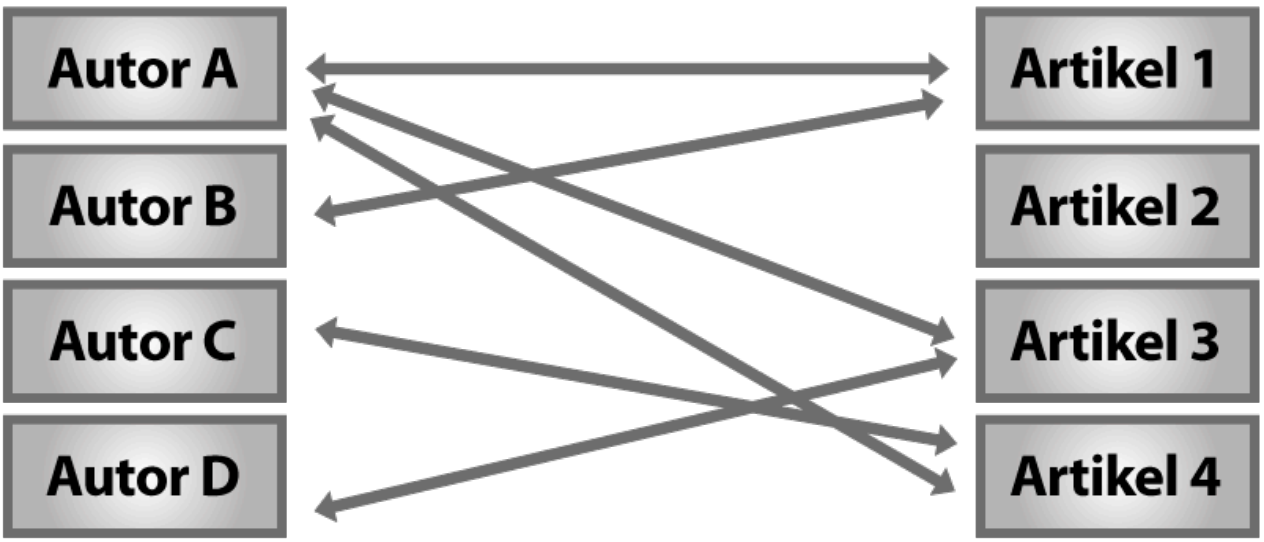
Für den **n:m Beziehungstyp** (mehrere Datensätze von Typ A können mit mehreren Datensätzen von Typ B in Relation stehen) wird eine neue Relation *R* mit den Attributen a_n erstellt:

$$R=\{a_1,a_2,...,a_n\} \cup \{k_T\} \cup \{k_S\}$$

wobei

$\{a_1,a_2,...,a_n\}$ *: Attribute der Beziehung sowie* k_T *und* k_S *für Primärschlüssel der jeweiligen Relation*

Im praktischen Einsatz von relationalen Datenbanken bedeutet das, dass eine Normalisierungstabelle zusätzlich zu den vorhanden Tabellen erstellt werden muss.



ID	Firstname
1	Alice

ID	Artikel
1	Lorem ipsum

ID	Artikel_ID	Link_ID
1	2	1

ID	Firstname
2	Bop
3	Chris

ID	Artikel
2	Dolor sit amet
3	Consetetur

ID	Artikel_ID	Link_ID
2	2	2
3	3	1

2.4. Aufbau der Abfragesprache SQL

Die **S**tructured **Q**uery **L**anguage ist eine standardisierte Sprache zur Abfrage, Definition und Veränderung von Datenbeständen in einer relationalen Datenbank. Sie relativ einfach zu verstehen und semantisch an die englische Sprache angelehnt. Auch die hier verwendetet Datenbank MySQL verwendet SQL als Abfragesprache [29] .

Beispiele für DML-Befehle in SQL: [30]

```
INSERT INTO Relation [( Attribut+ )] VALUES ( ( Konstante+ ) )+
INSERT INTO Relation [( Attribut+ )] SFW-Block[31]
UPDATE Relation SET (Attribut=Ausdruck)+ [WHERE Where-Klausel]
MERGE INTO Relation USING Quelle ON Join-Klausel
    WHEN MATCHED UPDATE SET (Attribut=Ausdruck)+
    WHEN NOT MATCHED [BY TARGET] INSERT (Attributliste) VALUES
(Ausdruckliste)
    [WHEN NOT MATCHED BY SOURCE DELETE]
DELETE FROM Relation [WHERE Where-Klausel]
TRUNCATE Relation
```

Beispiel eines ausführlicheren DQL-Befehls in SQL:

```
SELECT Auswahlliste AS Spaltenalias
FROM Quelltabelle AS Tabellenalias
WHERE Where-Bedingung
GROUP BY Group-by-Attribut
HAVING Having-Klausel
ORDER BY Sortierungsattribut ASC | DESC;
```

[29] http://dev.mysql.com/doc/refman/5.1/de/sql-syntax.html

[30] http://de.wikipedia.org/wiki/Data_Manipulation_Language#SQL

[31] SWF-Block, Abkürzung für **S**ELECT-**F**ROM-**W**HERE Block

2.5. Reguläre Ausdrücke

Ein regulärer Ausdruck ist eine Zeichenkette, die mit syntaktischen Regeln Mengen und Untermengen von anderen Zeichenketten beschreibt. In der Praxis wird dies z.B. zur Überprüfung von bestimmten Zeichenmustern verwendet.

So überprüft als Beispiel der reguläre Ausdruck

^([a-zA-Z0-9_+\.\-]+)@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.)|(([a-zA-Z0-9\-]+\.)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)$

die Syntax einer Zeichenkette auf eine gültige eMail-Adresse. In MySQL und MongoDB können Abfragen auch mit regulären Ausdrücken definiert werden: [32]

```
mysql> SELECT Email
            FROM User
            WHERE Email RLIKE
            "^([a-zA-Z0-9_+\.\-]+)@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]
{1,3}\.)|(([a-zA-Z0-9\-]+\.)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)$"
            LIMIT 5;
+---------------------------+
| Email                     |
+---------------------------+
| 1.netzwerk@gmx.de         |
| 11546@portrait.de         |
| 17605@portrait.de         |
| alexxandra17@aol.com      |
| amica.ambiente@t-online.de |
+---------------------------+
```

2.6. Einsatzbereich von MySQL

Aufgrund des Open-Source-Hintergrunds und der dahinterstehenden Firma MySQL (später SUN[33], heute ORACLE) bot MySQL den Nutzern von Anfang an professionellen Support und freie Softwarelizenzierung. Dadurch wurde es zum Quasistandard für Webentwicklung, in der sogenannten LAMP (Linux, Apache, MySQL und PHP) Umgebung. MySQL wird aber auch als Hochverfügbarkeitsdatenbank in Firmen und Institutionen eingesetzt [34] [35].

---

[32] http://dev.mysql.com/doc/refman/5.5/en/regexp.html

[33] http://www.heise.de/newsticker/meldung/Sun-kauft-MySQL-AB-fuer-eine-Milliarde-US-Dollar-Update-179176.html

[34] http://www.mysql.com/customers/

[35] http://www.mysql.de/why-mysql/case-studies/de/neckermann.php

2.7. Performanceoptimierung

MySQL bietet mehre Möglichkeiten von Performanceoptimierung. Durch Pufferoptimierung und RAID-Konfigurationen, können Lese- und Schreibgeschwindigkeiten verbessert werden [36]. Zusätzlich trägt gutes Datenbankdesign und Query-Optimierung positiv zur Performance bei. Eine wichtige Optimierungsmöglichkeit ist das Setzen von Indizes auf bestimmte Spalten:

```
mysql> CREATE INDEX indexierte_spalte ON spalte(zu_indexierende_spalte)
```

2.8. Horizontale Skalierbarkeit

Statt die Hardware bei Auslastung in einem Server auszubauen (vertikale Skalierung, scale up) werden zusätzliche Server in das System eingefügt (horizontale Skalierung, scale out). Dadurch wird gewährleistet, dass ein Serverausfall kein Systemversagen verursacht[37] . Zudem ist eine Aufrüstung von neuen Servereinheiten wesentlich einfacher. Bei horizontaler Skalierung wird zeilenweise aufgeteilt. Bei Normalisierung spaltenweise. MySQL bietet mehre Möglichkeiten zur Replikation. [38]

2.9. Anweisungsbasierte und datensatzbasierte Replikation

Seit MySQL 5.1.5 steht neben der anweisungsbasierten Replikation (auch horizontale anweisungsbasierte Replikation genannt) auch eine datensatzbasierte Replikation (auch vertikale anweisungsbasierte Replikation genannt) zur Verfügung. Die anweisungsbasierte Replikation ist schnell und erzeugt kleinere Logdateien aber nichtdeterministische Anweisungen können nicht korrekt repliziert werden. Bei datensatzbasierter Replikation können alle Daten korrekt Repliziert werden, auf Kosten von Geschwindigkeit und größeren Logdateien. [39]

---

[36] Mark Matthews Jim Cole Joseph D. Gradecki, MySQL and Java Developer's Guide, 2003, Wiley Publishing, S. 310ff

[37] http://www.oser.org/~hp/bsyII/node6.html

[38] http://dev.mysql.com/tech-resources/articles/application_partitioning_wp.pdf

[39] http://dev.mysql.com/doc/refman/5.1/de/replication-row-based.html

3. Aufbau einer non-relationalen Datenbank [40]

3.1. Begriffserklärung NoSQL

Der Begriff NoSQL (steht für **n**ot **o**nly **SQL**) ist zu einem Schlagwort nicht-relationaler Datenbanksysteme geworden. Dabei ist nicht eine grundsätzliche Ablehnung von SQL als Abfragesprache gemeint, sondern neue Ansätze und Techniken um Datenbanken flexibler bezüglich Skalierbarkeit, Verfügbarkeit und Migrationen gegenüber herkömmlichen relationalen Datenbanksysteme zu gestalten.

Besonders im Bereich der Webapplikationen brauchen Datenbanken die Eigenschaften:

- hohe Flexibilität (eine Webseite ändert sich ständig, dadurch muss auch die Datenstruktur ständig neu angepasst werden)
- hohe Verfügbarkeit (eine Webseite die nicht erreichbar ist, wird kein Erfolg haben)
- hohe Skalierbarkeit (anfangs kleine Webseiten können sehr schnell sehr groß werden)
- eventuelle Konsistenz (oft können es Webseiten verschmerzen, für kurze Zeit veraltete/nicht konsistente Datensätze auszugeben)

3.2. Geschichte von MongoDB

MongoDB gehört zu den dokumentenbasierten Datenbanken und zählt zusammen mit CouchDB zu den bekanntesten Vertretern in der Open-Source-Liga. Die Firma 10gen der MongoDB hat bekannte Gesichter aus dem Internetbereich in den Führungspositionen.
Die Datenbank ist in C++ geschrieben und auf hohe Leistung und große Datenmengen ausgelegt.

3.3. Document-Stores-Datenbank

Dokumentenbasierte Datenbanken sind auf eine schemafreie Struktur ausgelegt. Es gibt also kein festes Tabellenschema und dadurch z.B. keine zwingenden Relationstabellen. Stattdessen können Relationen entweder direkt im Datensatz gespeichert oder bei Bedarf individuell erstellt werden. Dennoch ähnelt MongoDB in vielen Punkten relationalen Datenbanken:

- es wird eine eindeutig ID pro Datensatz erstellt
- es wird in Collections gearbeitet, gleichbedeutend zu Tabellen bei relationalen Datenbanken
- es können Select-Relevante Spalten indiziert werden, um die Performance bei der Abfrage dieser Felder zu erhöhen

3.4. Kollektionen und Dokumentenstruktur

Statt von Tabellen wird von Kollektionen gesprochen, da kein festes Schema vorgegeben ist. Jede Kollektion kann Dokumente beinhalten, analog zu Zeilen / Datensätze in einer Tabelle.

---

[40] Als non-relationales Datenbanksystem wird die MongoDB der Firma 10gen verwendet

Jedes Dokument einer Kollektion hat einen Tupelaufbau und ist durch einen Key-Value-Eintrag gegeben:

$$E = ((k_1, v_1), (k_2, v_2) \dots , (k_n, v_n))$$

k : key, v : value
wobei $1 \leq i \leq n,\ n \in N$

Wie schon erwähnt geht MongoDB über die übliche Key-Value-Speicherung hinaus. Beliebige Felder können indexiert werden und es kann das standardmäßig gesetzte id-Feld (siehe ObjectID) als Primärschlüssel verwendet werden.

3.5. BSON Format

Alle Dokumente werden in der MongoDB im BSON-Format [41] (**B**inary **JSON** [**J**ava**S**cript **O**bject **N**otation]) gespeichert und ausgegeben. Das BSON-Format stammt von dem kompakten JSON-Format [42] ab und ist, wie das Präfix Binary andeutet, für eine overhead-arme Darstellung von binären Datenobjekten ausgelegt.

Beispiel:

```
{
  "Benutzername": "m.mustermann",
  "Name": {
    "Vorname": "Max",
    "Nachname": "Mustermnn",
  },
  "ÜbergeordneterChef": {},
  "Kollegen": [ "a.anderson", "b.benetton" ],
  "IstAbteilungsleiter": true,
  "Abteilungen": null,
}
```

[41] http://www.bsonspec.org

[42] http://en.wikipedia.org/wiki/JSON

3.6. CRUD

3.6.1. Create [43]

Um Datensätze einzufügen werden in der jeweiligen Programmiersprache Objekte erzeugt und diese dann der Kollektion hinzugefügt.

mycollection = *Beispielkollektion/Tabelle*
key = *Schlüssel / Feldname*
value = *Schlüsselwert / Wert des Eintrags*

```
> db.mycollection.insert({key:"value"});
```

3.6.2. Read [44]

Abfragen werden ebenfalls mit Objekten definiert und werden mit Hilfe von Cursorn ressourcenschonend abgearbeitet. Es werden immer nur die Felder ausgelesen, die explizit abgefragt werden.

```
> db.mycollection.findOne({key:"value"});
{ "_id" : ObjectId("4d8fa35d2e122c3c62885d2f"), "key" : "value" }
```

3.6.3. Update [45]

Wenn ein Dokumentenobjekt aus der Datenbank geladen ist, kann es verändert und wieder gespeichert werden.

```
db.mycollection.update(
  {
    Bedingungsarray
  },
    Wertepaare für neuen Attributen,
    [Dokument Erstellen wenn nicht gefunden (true/false) ]
  );
```

[43] http://www.mongodb.org/display/DOCS/Inserting

[44] http://www.mongodb.org/display/DOCS/Advanced+Queries

[45] http://www.mongodb.org/display/DOCS/Updating

```
> myValue = db.mycollection.findOne({key:"value"});
> db.mycollection.update( Bedingung, Objekt,
ErstellenWennNichtGefunden, WennAlleKriterienErfülltSind );
> db.mycollection.update({ _id : myValue._id }, { key : null },
false);
> db.mycollection.findOne({ _id : myValue._id });
{ "_id" : ObjectId("4d8fa35d2e122c3c62885d2f"), "value" : null }
```

myValue = *Referenz auf das gefundene Dokument*

3.6.4. Delete [46]

Jedes Dokumentenobjekt kann über eine Methode gelöscht werden.

```
> db.mycollection.remove({ _id : myValue._id });
```

3.7. ACID in MongoDB

Es gibt die Möglichkeit Primärschlüssel zu definieren, indexieren und mit einmalig-vorhandenen Werten zu versehen. Standardmäßig erledigt das die ObjectID, die jedem Datensatz automatisch zugewiesen wird. Auf Wunsch können auch andere Felder die Funktion des Primärschlüssels übernehmen, hier z.B. das Feld „username“:

```
> db.mycollection.ensureIndex({"username" : 1}, {"unique" : true} )
```

Um eine Stapelverarbeitung von bestimmten Suchkriterien durchzuführen, kann mit Hilfe des Cursors eine große Zahl von Dokumenten abgearbeitet werden.

Um konsistente Resultate zu bekommen, d.h. dass die Veränderung durch die Stapelverarbeitung keinen Einfluss auf das abzuarbeitende Resultat hat, kann die „Snapshot“-Funktion verwendet werden [47]:

```
cursor = db.mycollection.find().snapshot();
```

[46] http://www.mongodb.org/display/DOCS/Removing

[47] Beispiel aus: Kristina Chodorow & Michael Dirolf „MongoDB: The Definitive Guide“, 2010 O’Reilly Media, Seite 62f

```
while (cursor.hasNext()) {
  var doc = cursor.next();
  doc = process(doc);
  db.mycollection.save(doc);
}
```

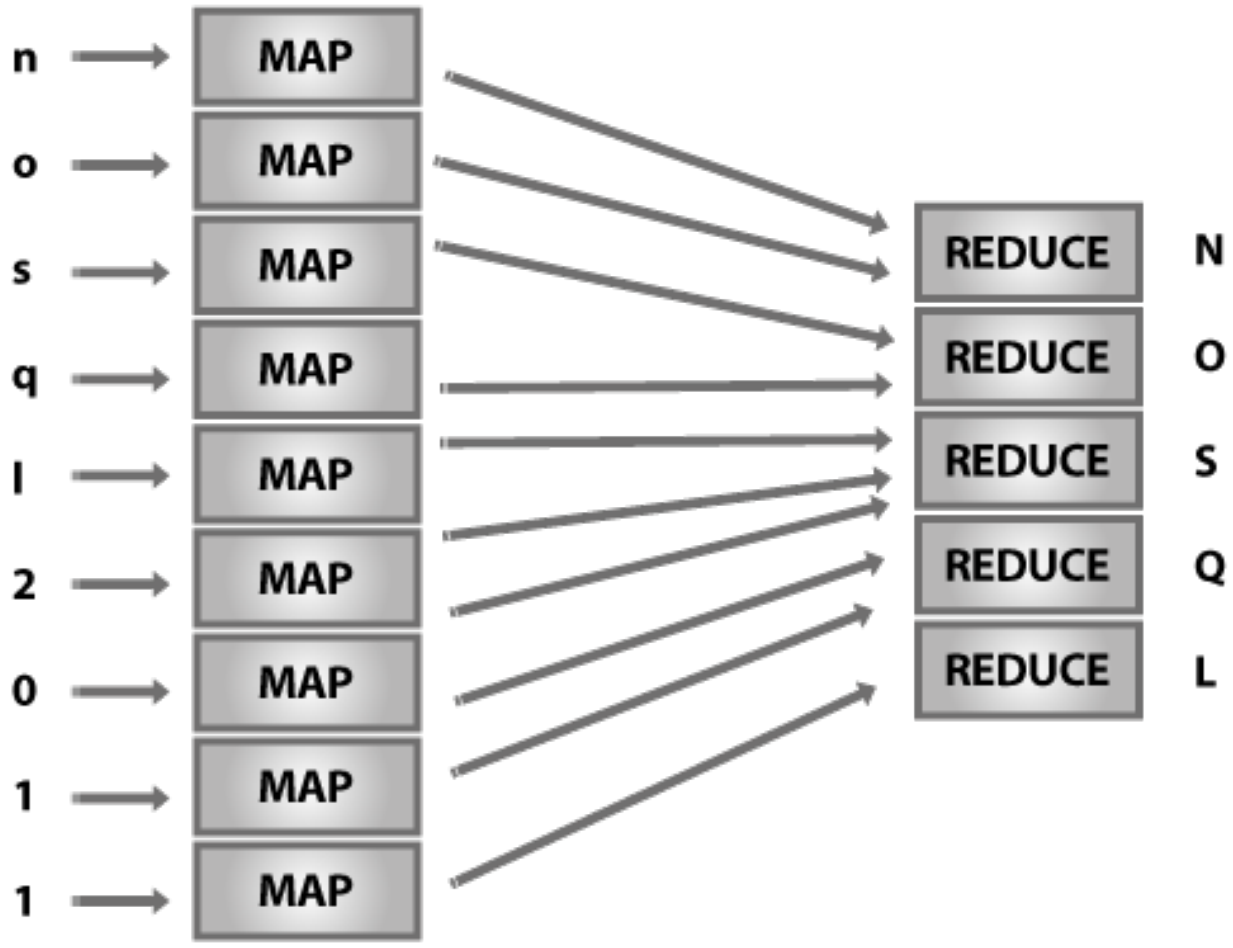
process() = eigens definierte Funktion zur Bearbeitung des Dokuments

3.8. Map-Reduce-Verfahren

Das Map-Reduce-Framework wurde von Google entwickelt, um die parallele Verarbeitung und Ausgabe von großen Datenmengen zu ermöglichen. Die Funktionen map und reduce haben in abgewandelter Form ihren Ursprung in der funktionalen Programmierung [48].

Beispiel eine map-reduce-Funktion:

Die Funktion wandelt eine beliebige Zeichenkette in ausschliesslich alphabetische Großbuchstaben um.



[48] http://labs.google.com/papers/mapreduce.html

Die MongoDB kann map-reduce-Fuktionen verwenden um komplexe Datenbankabfragen und Rechenoperationen auf mehrere Server zu verteilen [49].

3.9. Datenabfrage mittels MongoDB-API

Statt einer Abfragesprache verwendet MongoDB objektspezifische Methoden der jeweiligen Programmiersprache. Es werden für Abfragen und Änderungen keine SQL-Zeichenketten zusammengefügt. Stattdessen werden Abfragen und Änderungen durch zustandsspezifische Objektmethoden erzeugt.

3.9.1. Erzeugen einer Datenbankinstanz und Verwendung von Kollektionen

Wie bereits erwähnt, wird bei MongoDB statt von Tabellen von Kollektionen (collections) gesprochen, und anstatt von Datensätzen von Dokumenten (documents). Wenn eine Datenbank und eine Kollektion abgefragt, wird sie bei nicht Vorhandensein automatisch erstellt.

3.9.2. Aufbau eines Dokuments

Jedes Dokument kann eine beliebige (unabhängige) Anzahl von Feldern besitzen - auch eine verschachtelte Array-Struktur ist möglich. Dokumente dürfen auch innerhalb eines Dokuments gespeichert werden.

3.9.3. ObjectID

Jedes Dokument erhält einen indexierten, einmaligen ID-Wert (ähnlich eines fortlaufenden Primärschlüssels bei MySQL).
Der Schlüssel ist 12-Byte groß und wird aus dem aktuellen Zeitwert, ComputerID, vorangegangener ID und eines fortlaufenden Zählers generiert [50]. Er wird automatisch jedem neuen Dokument zugewiesen, außer man legt den ID-Wert manuell fest. Der Wert muss einzigartig sein, darf also demnach nicht mehrmals vorkommen.

3.9.4. Datentypen

Ein MongoDB-Dokument wird im BSON-Format verwaltet (siehe Kapitel BSON-Format) und kennt u.a. folgende Datentypen [51]:

- null (i.d.R. für einen nicht gesetzten Wert)
- undefined (JavaScript unterscheidet zwischen null und undefined)
- boolean
- integer (32 und 64-Bit)
- float
- string

---

[49] Kristina Chodorow & Michael Dirolf „MongoDB: The Definitive Guide“, 2010 O’Reilly Media, Seite 86ff

[50] Kristina Chodorow & Michael Dirolf „MongoDB: The Definitive Guide“, 2010 O’Reilly Media, Seite 21

[51] Kristina Chodorow & Michael Dirolf „MongoDB: The Definitive Guide“, 2010 O’Reilly Media, S.15ff

• symbol (Objektbezogen)
• object id (siehe oben)
• reguläre Ausdrücke
• array
• code (JavaScript-Code)
• binary data
• embedded document (Dokument eingebettet in eine anderes Dokument)

Datentypen können auch als Suchkriterium verwendet werden.[52]

3.9.5. Datensätze mit Java erstellen, einfügen, finden, ändern und löschen [53]

Im folgenden Beispiel wird die Java-API [54] ausführlich demonstriert:

```
import com.mongodb.*;
import java.net.UnknownHostException;
import java.util.Date;
import java.util.List;
import java.util.regex.Pattern;
import org.bson.BSONObject;


public class Main {

    public static void main(String[] args) throws UnknownHostException {
```

Es wird eine MongoDB-Verbindung aufgebaut und auf ***m*** gelegt. Anschließend werden zur Übersicht alle vorhandenen Datenbanken aufgelistet.

```
        Mongo m = new Mongo("localhost");

        Main.echo("Liste alle Datenbanken auf");
        for (String o : m.getDatabaseNames()) {
            System.out.println(o);
        }
```

Auf ***db*** wird eine Datenbankinstanz auf die Datenbank **contacts** gelegt - wenn die angeforderte Datenbank nicht existiert, wird sie automatisch von MongoDB angelegt. Anschließend wird die Kollektion **clients** auf das DBCollection-Objekt ***clients*** instanziert. Auch hier gilt: bei Nichtvorhandensein wird die Kollektion automatisch angelegt.

```
        //Verwende die Datenbank "contacts"
```

[52] http://www.mongodb.org/display/DOCS/Advanced+Queries#AdvancedQueries-%24type

[53] http://api.mongodb.org/java/2.6-pre-/index.html

[54] http://www.mongodb.org/display/DOCS/Java+Tutorial

```
DB db = m.getDB("contacts");
//Verwende Kollektion "clients"
DBCollection clients = db.getCollection("clients");
//Lösche die Kollektion "clients"
clients.drop();
```

Es können bestimmte Felder indexiert werden - theoretisch für jedes Dokument unterschiedliche.

```
//Indexiere alle name.first-Felder
clients.ensureIndex("name.first");
clients.ensureIndex("name.sur");

//Gebe die Metadaten bezogen auf die Indexierung aus
List<DBObject> list = clients.getIndexInfo();
for (DBObject o : list) {
    System.out.println(o);
}

//Erstelle benötigte Datenbankobjekte
BasicDBObject alice = new BasicDBObject();
BasicDBObject name = new BasicDBObject();
//erstelle Person Alice
name.put("first","Alice");
name.put("sur","Aniston");
alice.put("name",name);
alice.put("zipcode", (int) 50123);
alice.put("city", "New York");
//Erzeuge das Geburtstdatum 31.12.1980 06:00:00
alice.put("birthday", new Date(80,11,31,6,0,0));
//füge das Dokument zur Kollektion „clients“ hinzu
clients.insert(alice);
//Erstelle Person Bop
name = new BasicDBObject();
name.put("first","Bop");
name.put("sur","Bennetton");
BasicDBObject bob = new BasicDBObject();
bob.put("name", name);
bob.put("zipcode",(int) 12345);
bob.put("city", "New York");
//Erstelle Kopie von Alice bei Bop
bob.put("supervisor_copy",alice);
clients.insert(bob);

//Bearbeite Alice
alice.put("is_supervisor",(boolean) true);
clients.save(alice);
DBCursor results = clients.find();
//Gebe alle Personen aus
Main.dumpResults(results);

//Bearbeite Bob, erstelle Relation mit Alice
DBRef supervisor = new DBRef(db,"related_supervisor",alice.get("_id"));
```

```
        bob.remove("supervisor_copy");
        bob.put("supervisor",supervisor);
        clients.save(bob);
        results = clients.find();
        Main.dumpResults(results);

        //Erstelle Kopie von Bob und nenne sie Chris
        name.put("first","Chris");
        name.put("sur", "Cartner");
        bob.put("name", name);
        //ID wird zurückgesetzt, damit ein neuer Datensatz erstellt wird
        bob.put("_id",null);
        clients.save(bob);

        BasicDBObject condition = new BasicDBObject();

        //Gebe alle Personen aus, deren Nachnamen auf 'ton' endet (reg. Ausdruck)
        condition.put("name.sur", Pattern.compile("ton$",Pattern.CASE_INSENSITIVE));
        results = clients.find(condition);
        Main.dumpResultsShort(results);

        //Gebe alle Personen aus, wo 10000 < PLZ <= 20000
        condition = new BasicDBObject();
        condition.put("zipcode", new BasicDBObject("$gt", 10000).append("$lte",20000));
        results = clients.find(condition);
        Main.dumpResultsShort(results);

        //Gebe alle Personen aus, sortiere nach Nachnamen absteigend
        results = clients.find();
        BasicDBObject sort = new BasicDBObject("name.sur",0);
        results.sort(sort);
        Main.dumpResultsShort(results);

        //Gebe alle Personen alphab. (+aufsteigend) aus, beginne beim 2ten, max. 2
        //analog zu LIMIT 1,2
        results = clients.find();
        sort.put("name.sur", 1);
        Main.dumpResultsShort(results.sort(sort).skip(1).limit(2));

    }

    //Methode zur Textausgabe auf der Konsole
    private static void echo(String message) {
        System.out.println(message);
    }

    //Gibt alle Datensätze (komplett) auf der Konsole aus
    private static void dumpResults(DBCursor results) {
        while (results.hasNext()) {
            System.out.println(results.next());
        }
    }

    //Gibt alle Datensätze übersichtlich auf der Konsole aus
```

```
        //mit den Feldern ID,Vorname,Nachname,PLZ,Stadt
        private static void dumpResultsShort(DBCursor results) {
            while (results.hasNext()) {
                DBObject record = results.next();
                BSONObject name = (BSONObject) record.get("name");
                System.out.println("#"+record.get("_id")+": "+name.get("first")+" "+
                   name.get("sur")+",
               "+record.get("zipcode")+" "+record.get("city"));
            }
        }


    }
```

3.10. Anwendung der MongoDB Konsole

Über eine interaktive Konsole [55] kann ähnlich wie bei einer SQL-Server-Konsole mittels einer Java-Script API [56] direkt auf der Datenbank gearbeitet werden.

Liste alle Datenbanken auf:

```
> show dbs;
admin
local
wikipedia
```

Verwenden die Datenbank wikipedia

```
> use wikipedia;
switched to db wikipedia
```

Liste alle Kollektionen / Tabellen auf:

```
> show collections;
articles
system.indexes
textindex
```

[55] http://www.mongodb.org/display/DOCS/mongo+-+The+Interactive+Shell

[56] http://www.mongodb.org/display/DOCS/Server-side+Code+Execution

Suche alle Datensätze aus der Kollektion **articles**, deren Titel mit SQL beginnt:

```
> db.articles.find({title : /^SQL/ },{title:true});
{ "_id" : "47699eee6169d5b455486e7fca5c0df77eb79f5b", "title" : "SQL" }
{ "_id" : "7afcd2b29d4b54593d2def079419a8ea2b521e4c", "title" : "SQL Injection" }
{ "_id" : "3dba7003c5e8c3531f9e7614b58c389f25657c96", "title" : "SQL Slammer" }
{ "_id" : "3973b00f5368050bd2d2e8ea48e12300c576ba34", "title" : "SQLJ" }
{ "_id" : "edf684fb5bd8b7c55c1ef24ed523ec03f0464b1b", "title" : "SQLite" }
...
```

Füge zu der Kollektion **foo** einen neuen Datensatz mit den Werten:
„name.first“ => „Alice“ und „random_key“ => Zufallszahl zwischen 0 und 1000

```
> db.foo.insert({
  "name" : {
    "first" : "Alice"
  },
  "random_key" : Math.round(Math.random()*1000)
})
```

Finde den ersten Datensatz

`SELECT * FROM `foo` **WHERE 1 LIMIT 1** (analog zu SQL)`

```
> db.foo.findOne()
{
        "_id" : ObjectId("4d8e1ce9c299ea272f9783fd"),
        "name" : {
                "first" : "Alice"
        },
        "random_key" : 499
}
```

Suche alle Datensätze raus und zeige die Felder „name.first“ und „random_key“ an

**`SELECT `FirstName`, `RandomKey`** FROM `foo` WHERE 1`

```
> db.foo.find({},{"name.first" : 1, "random_key" : 1});
{ "_id" : ObjectId("4d8e2fe5c299ea272f978401"), "name" : { "first" : "Alice" }, "random_key" :
499 }
{ "_id" : ObjectId("4d8e3003c299ea272f978402"), "name" : { "first" : "Bop" }, "random_key" :
148 }
```

Zeige alle, beginne bei 50 und zeige max.100

```
SELECT * FROM `foo` WHERE 1 LIMIT 50,100
```

```
> db.foo.find().limit(100).skip(50)
```

Zeige alle, wo Feld x und Feld y zusammen 10 ergeben

```
SELECT * FROM `foo` WHERE SUM(X,Y)=10
```

```
> db.foo.find({"$where" : "this.x + this.y == 10"})
```

Eine eigene Funktion als Bedingung

```
DELIMITER //
CREATE FUNCTION sumIsTen (n1 INT, n2 INT) RETURNS boolean
DETERMINISTIC
BEGIN
  DECLARE sum int;
 SET sum = (n1+n2);
 RETURN IF(sum=10,true,false);
END
//
```

```
> db.foo.find({"$where" : "function() { return this.x + this.y == 10; }"})
```

Durchsuchen von Arrays:

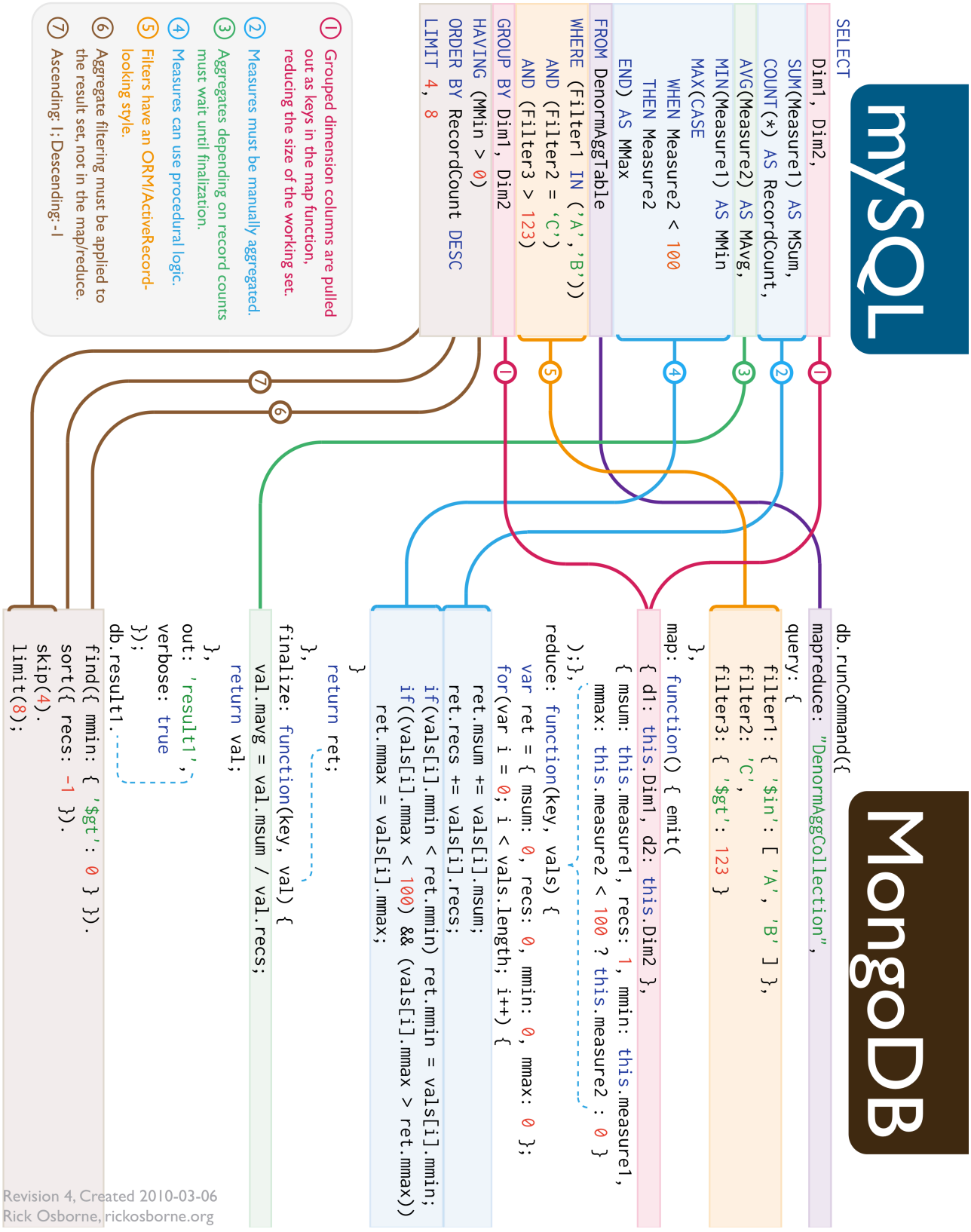
```
> db.foo.find({"name.first" : "Alan", "name.last" : "Smithee"})
```

Suchen nach Array-Schlüsseln:

```
> db.foo.find({"links" : {"$elemMatch" : {"sql" : true, "related_links_count" :
{"$gte" : 5}}}})
```

Abarbeitung von Suchresultaten:

```
> cursor = db.foo.find();
> while (cursor.hasNext()) {
  var doc = cursor.next();
  doc = process(doc);
  db.foo.save(doc);
}
```



Gegenüberstellung der MongoDB-API mit der MySQL-Abfragespache [57]

Zu 1-4: In MongoDB wird das Map-Reduce-Verfahren eingesetzt, um:

[57] http://rickosborne.org/download/SQL-to-MongoDB.pdf

- Resultate auszuwählen und zu gruppieren (Map)
- Nach bestimmten Kriterien neuzuordnen und mit Funktionen einzuschränken (Reduce)
- Resultate zusammenzufassen, die von der Anzahl der gefundenen Dokumente abhängt (Finalize, wird nach Map-Reduce ausgeführt)

Zu 5: Suchfilter werden in Form von Objekten und Arrays definiert
Zu 6: Um Resultate weiter zusammenzufassen, muss man das Resultat der Map-Reduce-Finalize-Funktionen abwarten
Zu 7: Sortierung: ASC = 1, DESC = -1

3.11. Horizontale Skalierbarkeit

MongoDB ist auf horizontale Skalierbarkeit ausgelegt und verfügt mit Hilfe des „Shardings“ [58] (eine Methode der horizontalen Skalierung) eine mächtige Funktion zum Aufteilen der Datenbank auf mehrere Server [59] .

Beispiel für das Einrichten eines Sharding-Servers: [60]

Es muss ein Verzeichnis für die Konfiguration auf dem Sharding-Server angelegt werden:

```
$ mkdir -p ~/dbs/config
```

Anschließend wird der MongoDB-Dienst auf dem Sharding-Server gestartet:

```
$ ./mongod --dbpath ~/dbs/config --port 20000
```

Auf dem Hauptserver (mongos) wird der Sharding-Server dann über folgenden Befehl eingebunden:

```
$ ./mongos --port 30000 --configdb localhost:20000
```

Über jede beliebige Konsole kann eine Verbindung zum Hauptserver aufgebaut werden

```
$ ./mongo localhost:30000/admin
```

[58] http://en.wikipedia.org/wiki/Shard_(database_architecture)

[59] Kristina Chodorow & Michael Dirolf „MongoDB: The Definitive Guide“, 2010 O’Reilly Media, Seite 143ff

[60] Befehle entnommen aus:
Kristina Chodorow & Michael Dirolf „MongoDB: The Definitive Guide“, 2010 O’Reilly Media, Seite 147f

```
MongoDB shell version: 1.6.0
url: localhost:30000/admin
connecting to localhost:30000/admin
>
```

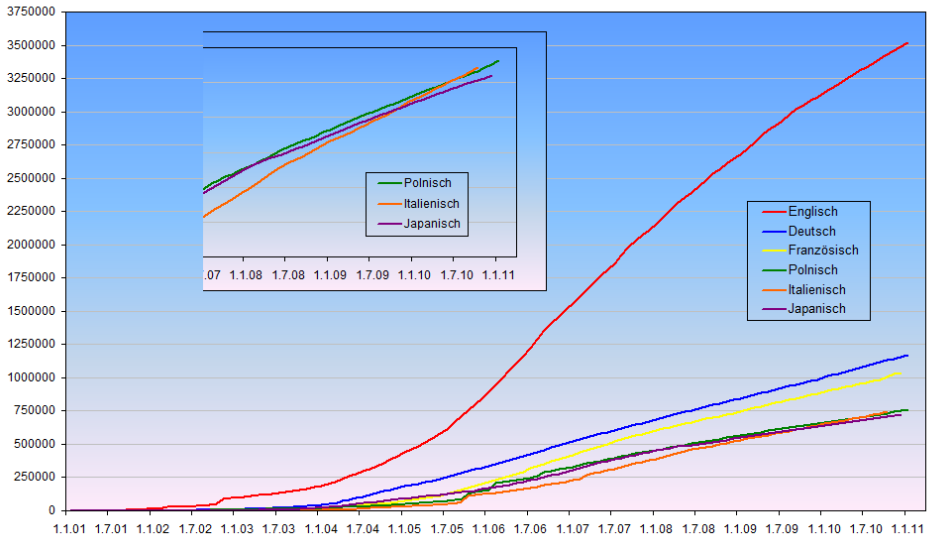
Zum überprüfen, ob der Sharding-Server eingebunden ist, reicht der Befehl:

```
> db.runCommand({addshard : "localhost:10000", allowLocal : true})
{
        "added" : "localhost:10000",
        "ok" : true
}
```

4. Migration SQL zu NoSQL

4.1. Geschichte und Probleme der Datenbankarchitektur der Wikipedia

Die Wikipedia zählt zu den meist besuchten Webangeboten mit über 10 Mio. Besuchern pro Tag und verfügt mittlerweile über 17 Mio. Artikel in insgesamt 270 verschiedenen Sprachen [61] [62] [63] .



Entwicklung der Artikelanzahl der 6 größten Wikipedias [64]

Aufgrund ihrer „älteren“ Geschichte wurde die Webapplikation Wikimedia (Frontend- und Backend für die meisten Wikis, auch Wikipedia) für die relationale MySQL Datenbank entwickelt und bis heute verwendet.

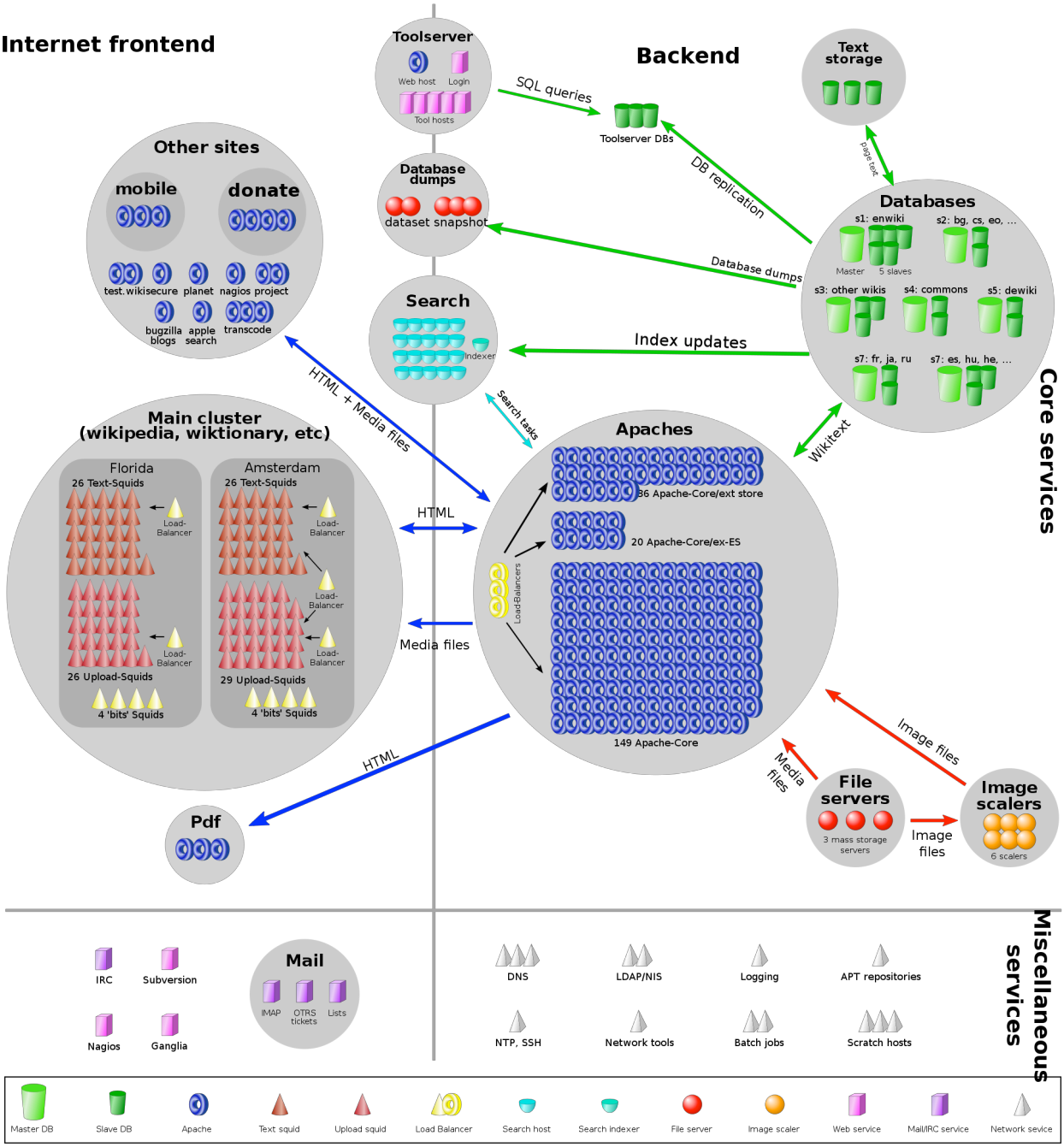
---

[61] http://en.wikipedia.org/wiki/Wikipedia:About

[62] http://de.wikipedia.org/wiki/Wikipedia:Gr%C3%B6%C3%9Fenvergleich

[63] http://en.wikipedia.org/wiki/Wikipedia:Modelling_Wikipedia's_growth

[64] http://upload.wikimedia.org/wikipedia/commons/b/b5/Entwicklung_der_Artikelanzahlen_der_f%C3%BCnf_gr%C3%B6%C3%9Ften_Wikipedias.png



Netzwerkstruktur der Wikipedia [65]

Die englische Wikipedia (als Beispiel) wird mit einem Master und 5 Slaves betrieben. Alle Datenbanken werden zu der Toolserver-DB-Einheit repliziert.

Der Textinhalt inklusive alle Absätze (mit Bild-, Referenzverweise und Erläuterungen) wird in einer einer Tabellenspalte gespeichert. Das ist weder performant noch strukturtechnisch sinnvoll sondern eine Einschränkung von schemafesten Datenbanken, zu denen MySQL gehört.

[65] http://en.wikipedia.org/wiki/File:Wikimedia-servers-2010-12-28.svg
http://upload.wikimedia.org/wikipedia/commons/d/d8/Wikimedia-servers-2010-12-28.svg

4.2. Vorteile vom Einsatz einer NoSQL Datenbank

Aufgrund der Wikipedia-Artikelstruktur, die sich von Beitrag zu Beitrag unterscheidet, ist eine schemafreie Datenbank an sich sinnvoller. Eine solch hochfrequentierte Webseite muss zudem auf Verteilung und Redundanz ausgelegt ist und kann in dem Fall Wikipedia auch problemlos „Eventual Consitent“ betrieben werden. Für diese Voraussetzungen ist, wie bereits erwähnt, die hier verwendete MongoDB konzipiert.

4.3. Relationale Datenbankstruktur des Wikipedia-Benchmark-Datensatzes in MySQL

Tabellenstruktur [66]:

```
CREATE TABLE `articles` (
       `ID` int(11) NOT NULL AUTO_INCREMENT,
       `MongoID` varchar(255) NOT NULL,
       `Title` varchar(255) NOT NULL,
       `Redirect` varchar(255) NOT NULL,
       `Comment` text NOT NULL,
       `Content` longtext NOT NULL,
       `Links` text NOT NULL,
       PRIMARY KEY (`ID`),
       KEY `ArticleTitle` (`Title`)
       ) ENGINE=MyISAM  DEFAULT CHARSET=utf8;
```

- **ID (**Primärschlüssel, wird mit jedem Datensatz raufgesetzt und ist ist einmalig)
- **MongoID** (entspricht der ObjectID des MongoDB-Dokuments)
- **Title** (ist indexiert, zur schnellen Suche)
- **Redirect** [67] (enthält ggf. den Artikeltitel, auf den verwiesen wird)
- **Comment** (Kommentar)
- **Content** (Kompletter Artikelinhalt)
- **Links** (Alle verlinkten Artikel in kommaseparierter Auflistung)

Abfrage eines Artikels: [68]

[66] Entspricht der hier verwendeten Tabellenstruktur für den Benchmark

[67] Eigentlich wird eine Verweisartikel in der „Content“-Spalte mit „Weiterleitung“ definiert, für bessere Performance auf beiden Systemen wird dieses Feld extra gespeichert

[68] Aufgrund von Platzmangel mit eingeschränkter Spaltenauswahl

```
mysql> SELECT `ID`, `MongoID`, `Title`, `Links` FROM articles WHERE
Title = 'Aussagenlogik' LIMIT 1;
+----+------------------------------------+--------------+-------------------+
| ID | MongoID                            | Title        | Links             |
+----+------------------------------------+--------------+-------------------+
|  6 | 4ab08a644c8c7f8daa93b9ba8317bdeb0757af25 | Aussagenlogik | Logik,Junktor, ... |
+----+------------------------------------+--------------+-------------------+
1 row in set (0.00 sec)
```

4.4. Non-Relationale Struktur eines Bechmark-Datensatzes in MongoDB

Da wir von der Struktur grundsätzlich frei sind, unterteilen wir den Artikel auf erster Ebene in Unterabschnitte [69].

Als Muster ergibt sich demnach analog zur MySQL-Tabellenstruktur:

- **_id** (wird automatisch von MongoDB zugewiesen, einmalig)
- **title** (Artikeltitel, indexiert)
- **comment** (Kommentar)
- **redirect** (optional, enthält ggf. den Artikeltites, auf den verwiesen wird)
- **sections[]** (Jeder Artikel kann *n* Abschnitte haben)
  - sections[*n*]
    - **subtitle** (Überschrift eines Unterabschnitts, indexiert)
    - **content** (Inhalt eines Unterabschnitts)
- **links[]** (Jeder Artikel kann *m* Links haben, indexiert [70])
  - links[*m*] (Titel des verlinkten Artikels)

wobei n,m ∈ N einschl. 0

Struktur eines kompletten Artikels:

```
> db.articles.findOne({title:"Aussagenlogik"});
{
        "_id" : "4ab08a644c8c7f8daa93b9ba8317bdeb0757af25",
        "title" : "Aussagenlogik",
        "comment" : "kram",
        "sections" : [
                {
```

[69] theoretisch könnte die Verschachtlung tiefer gehen, der Arbeitsaufwand würde aber den Nutzen in unserem Falle übersteigen

[70] von der Indexierung wird in dem Benchmarktest kein Gebrauch gemacht - es dient lediglich zu anderwertigen Nutzung einer „Verwandte Artikel“-Suche

```
                        "subtitle" : "Aussagenlogik",
                        "content" : "Die '''Aussagenlogik''' ist ein
    Teilgebiet der [[Logik]], das sich mit Aussagen und deren Verknüpfung durch
    [[Junktor]]en befasst, ausgehend von strukturlosen [[Elementaraussage]]n
    (Atomen), denen ein [[Wahrheitswert]] zugeordnet wird. In der ''klassischen
    Aussagenlogik'' wird jeder Aussage genau einer der zwei Wahrheitswerte „wahr“
    und „falsch“ zugeordnet. Der Wahrheitswert einer zusammengesetzten Aussage
    lässt sich ohne zusätzliche Informationen aus den Wahrheitswerten ihrer
    Teilaussagen bestimmen."
                },
                ...
                {
                        "subtitle" : "Literatur",
                        "content" : "* Jon Barwise, John Etchemendy: ''The
    Language of First Order Logic'', Stanford: CSLI 1991 (=CSLI Lecture Notes; 23)
    ISBN 0-937073-74-1 \n* Ansgar Beckermann: ''Einführung in die Logik'' ISBN
    3-11-017965-2\n* Wolfgang Detel, ''Grundkurs Philosophie, Band 1. Logik'',
    Stuttgart: Reclam 2007, ISBN 978-3-15-018468-4 (Universal-Bibliothek, Nr.
    18468).\n* Wilfrid Hodges: ''Logic'', London: Penguin 1977, 2. Aufl. 2001 ISBN
    0-14-100314-6 \n* E. J. Lemmon: ''Beginning Logic'', London: Chapman and Hall
    London 1965, 2. Aufl. 1987 ISBN 0-412-38090-0 \n* Wesley C. Salmon: ''Logik'',
    Stuttgart: Reclam 1983 (=Universal-Bibliothek) ISBN 3-15-007996-9\n* [[Karel
    Berka]], Lothar Kreiser: ''Logik-Texte. Kommentierte Auswahl zur Geschichte
    der modernen Logik'', Berlin: Akademie 4. Aufl. 1986 ISBN 3-05-001500-4\n*
    Rüdiger Inhetveen: ''Logik. Eine dialog-orientierte Einführung.'' Leipzig 2003
    ISBN 3-937219-02-1"
                }
        ],
        "links" : [
                "Logik",
                        ...
                "Karel Berka"
        ]
}
```

4.5. Import des Wikipedia-SQL-Dumps

Alle aktuellen deutschsprachigen Artikel stehen in Form eines XML-Dumps zur Verfügung[71] . Mit Hilfe des mwdumpers[72] können die xml-Dateien in einen SQL-Dump konvertiert werden.

Um den XML-Dump sowohl in die MySQL als auch in die MongoDB zu importieren, muss der mwdumper entsprechend modifiziert werden.

---

[71] http://de.wikipedia.org/wiki/Wikipedia:Download, http://dumps.wikimedia.org/backup-index.html

[72] http://www.mediawiki.org/wiki/Manual:MWDumper

4.6. Starten und Dauer des Importvorgangs

Über den Befehl

```
$> java -Xms1024m -Xmx2048m -XX:NewSize=256m -XX:MaxNewSize=256m -XX:
+UseConcMarkSweepGC -XX:+UseParNewGC -XX:PermSize=128m -XX:MaxPermSize=128m
-jar ~/NoSQLDumper.jar --format=sql:1.5 ~/dewiki-latest-pages-articles.xml
```

wird der Importvorgang mit dem modifizierten mwdumper gestartet [73].
Der Importvorgang dauerte ca. 4,5 Stunden [74], wobei gleichzeitig in MySQL und MongoDB importiert wurde. Insgesamt wurde über 2,2 Mio. Datensätze importiert [75].

[73] Aufgrund von Memory-Heap-Problemen, empfiehlt es sich der Java-Umgebung mehr Arbeitsspeicher zuzuteilen

[74] **Verwendetes Betriebssystem und Hardware für alle Angaben:**
Mac OS X 10.6.7 mit Java-Runtime 1.6.0_24 (beide 64-Bit)
MySQL 5.5.9 (64-Bit)
MongoDB 1.6.6-pre (64-Bit) für Import
MongoDB 1.8.0 (64-Bit) für Benchmarktests
Dateisystem: Journaled HFS+
MacBookPro (Modell Herbst 2010)
2.4 GHz Intel Core 2 Duo, 8 GB 1067 MHz RAM, SAMSUNG HM500JI 500GB mit 5000 Umdr./s

[75] Stand des XML-Dumps: Juli 2010

5. Benchmarktest

5.1. Suchkriterien

Es wird getestet, wie schnell beide Datenbanksysteme Artikel nach bestimmten Kriterien raussuchen - es werden in diesem Benchmark also lediglich Lesevorgänge verwendet.

Dabei werden drei verschiedene Felder durchsucht

• Titel (Überschrift des Artikels)
• Unterüberschrift (Überschrift eines Unterbereichs)
• Inhalt (der komplette Textinhalt eines Artikels)

Die Suchkriterien:

Exakte Wortgruppe

```
Feld = 'Meine Suchbegriffe' (MySQL)
feld : 'Meine Suchbegriffe' (MongoDB)
```

Enthält Wortgruppe

```
Feld LIKE '%Meine Suchbegriffe%' (MySQL)
feld   :  /Meine Suchbegriffe/i  (reguläre Ausdruck für MongoDB)
```

5.2. Einschränkungen

Selbstverständlich sind beide Datenbanksysteme in vielen Punkten zu unterschiedlich, um einen „fairen“ Vergleich zu gewährleisten.

Da sind u.a. die unterschiedlichen Programmier-API‘s, Indexierungsfunktionen, Wildcardsuchfunktionen und Skalierungsmöglichkeiten. Beide Datenbanksysteme haben jeweils unterschiedliche Stärken und Schwächen in den aufgezählten Bereichen. Trotzdem wird sich erkennen lassen, wo sich beide Systeme in Sachen Geschwindigkeiten bewegen.

Reguläre Ausdrücke in MySQL habe ich nicht verwendet, da es dort zu erheblichen Performanceeinbußen gekommen ist. Stattdessen wurde ein LIKE-Statement verwendet. Zwar hat dadurch MySQL einen Geschwindigkeitsvorteil, aber für unsere Suche reicht das LIKE-Statement aus und ist deshalb kein unfairer Vorteil.

5.3. Zeitmessungen

Alle Zeitangaben sind in Sekunden [s] und auf 2 Nachkommastellen gerundet.

5.3.1. Suchbegriff als Titel des Artikels

Suche Titel mit **exaktem** Vorkommen des Suchbegriffs. Zusätzlich werden 10 weitere Artikel rekursiv pro Durchlauf rausgesucht (Limit 10). Aufgrund dessen wird der Mittelwert zum Schluss auf ein zehntel reduziert.

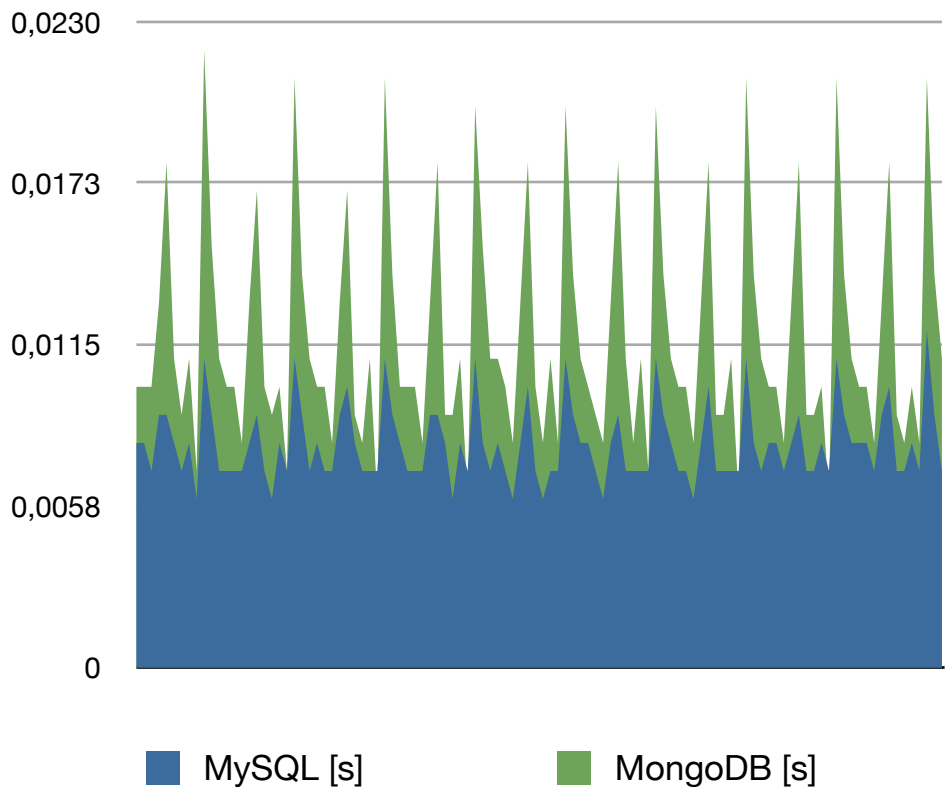
```
    SELECT * FROM `article` WHERE `Title` = 'Suchbegriff‘ LIMIT 1 (MySQL)
```

```
    db.articles.find( { title : 'Suchbegriff‘ } ) (MongoDB)
```

Titel	MySQL [s]	MongoDB [s]
Psychologie	0,0080	0,0100
Sommerzeit	0,0080	0,0100
Waschmaschine	0,0070	0,0100
Kaffee	0,0090	0,0130
Flugzeug	0,0090	0,0180
Bill_Clinton	0,0080	0,0110
SQL	0,0070	0,0090
Klavier	0,0080	0,0110
Sofa	0,0060	0,0070
Berlin	0,0110	0,0220
Sonne	0,0090	0,0150
Radioaktivität	0,0070	0,0110
Psychologie	0,0070	0,0100
Sommerzeit	0,0070	0,0100
Waschmaschine	0,0070	0,0080
Kaffee	0,0080	0,0130
Flugzeug	0,0090	0,0170
Bill_Clinton	0,0070	0,0100
SQL	0,0060	0,0090
Klavier	0,0080	0,0100
Sofa	0,0070	0,0070
Berlin	0,0110	0,0210
Sonne	0,0090	0,0140
Radioaktivität	0,0070	0,0110
Psychologie	0,0080	0,0100
Sommerzeit	0,0070	0,0100
Waschmaschine	0,0070	0,0080
Kaffee	0,0090	0,0130
Flugzeug	0,0100	0,0170
Bill_Clinton	0,0080	0,0090
SQL	0,0070	0,0080
Klavier	0,0070	0,0110
Sofa	0,0070	0,0060
Berlin	0,0110	0,0210
Sonne	0,0090	0,0140
Radioaktivität	0,0080	0,0100
Psychologie	0,0070	0,0100

Titel	MySQL [s]	MongoDB [s]
Sommerzeit	0,0070	0,0100
Waschmaschine	0,0070	0,0080
Kaffee	0,0090	0,0130
Flugzeug	0,0090	0,0180
Bill_Clinton	0,0080	0,0090
SQL	0,0060	0,0090
Klavier	0,0080	0,0110
Sofa	0,0070	0,0060
Berlin	0,0110	0,0200
Sonne	0,0080	0,0150
Radioaktivität	0,0070	0,0110
Psychologie	0,0080	0,0110
Sommerzeit	0,0070	0,0100
Waschmaschine	0,0060	0,0080
Kaffee	0,0080	0,0130
Flugzeug	0,0100	0,0180
Bill_Clinton	0,0070	0,0100
SQL	0,0060	0,0080
Klavier	0,0070	0,0110
Sofa	0,0070	0,0080
Berlin	0,0110	0,0200
Sonne	0,0090	0,0140
Radioaktivität	0,0080	0,0110
Psychologie	0,0080	0,0100
Sommerzeit	0,0070	0,0090
Waschmaschine	0,0060	0,0080
Kaffee	0,0080	0,0130
Flugzeug	0,0090	0,0180
Bill_Clinton	0,0070	0,0110
SQL	0,0070	0,0080
Klavier	0,0070	0,0110
Sofa	0,0070	0,0070
Berlin	0,0110	0,0200
Sonne	0,0090	0,0140
Radioaktivität	0,0080	0,0110
Psychologie	0,0070	0,0100
Sommerzeit	0,0070	0,0100
Waschmaschine	0,0060	0,0080
Kaffee	0,0080	0,0130
Flugzeug	0,0100	0,0180
Bill_Clinton	0,0070	0,0090
SQL	0,0070	0,0090
Klavier	0,0070	0,0110
Sofa	0,0070	0,0060
Berlin	0,0110	0,0210
Sonne	0,0080	0,0140
Radioaktivität	0,0070	0,0110
Psychologie	0,0080	0,0100
Sommerzeit	0,0080	0,0100
Waschmaschine	0,0070	0,0080
Kaffee	0,0080	0,0130
Flugzeug	0,0090	0,0180
Bill_Clinton	0,0070	0,0090

Titel	MySQL [s]	MongoDB [s]
SQL	0,0070	0,0090
Klavier	0,0080	0,0100
Sofa	0,0070	0,0060
Berlin	0,0110	0,0210
Sonne	0,0090	0,0140
Radioaktivität	0,0080	0,0110
Psychologie	0,0080	0,0100
Sommerzeit	0,0080	0,0100
Waschmaschine	0,0070	0,0080
Kaffee	0,0090	0,0130
Flugzeug	0,0100	0,0180
Bill_Clinton	0,0070	0,0090
SQL	0,0070	0,0080
Klavier	0,0080	0,0100
Sofa	0,0070	0,0080
Berlin	0,0120	0,0210
Sonne	0,0090	0,0140
Radioaktivität	0,0070	0,0110
Arithm. Mittel	**0,0079**	**0,0117**



Man erkennt, dass MySQL im Gegensatz zur MongoDB die Artikel im Schnitt nahezu doppelt so schnell raussucht. Die Indexierung des Titelfeldes bewirkte bei beiden Datenbanksystemen eine enorme Geschwindigkeitssteigerung.

Suche nach Vorkommen des Titels. Groß- und Kleinschreibung sowie die Position des Suchbegriffs sind variabel. Zusätzlich werden 10 weitere Artikel rekursiv pro Durchlauf rausgesucht.

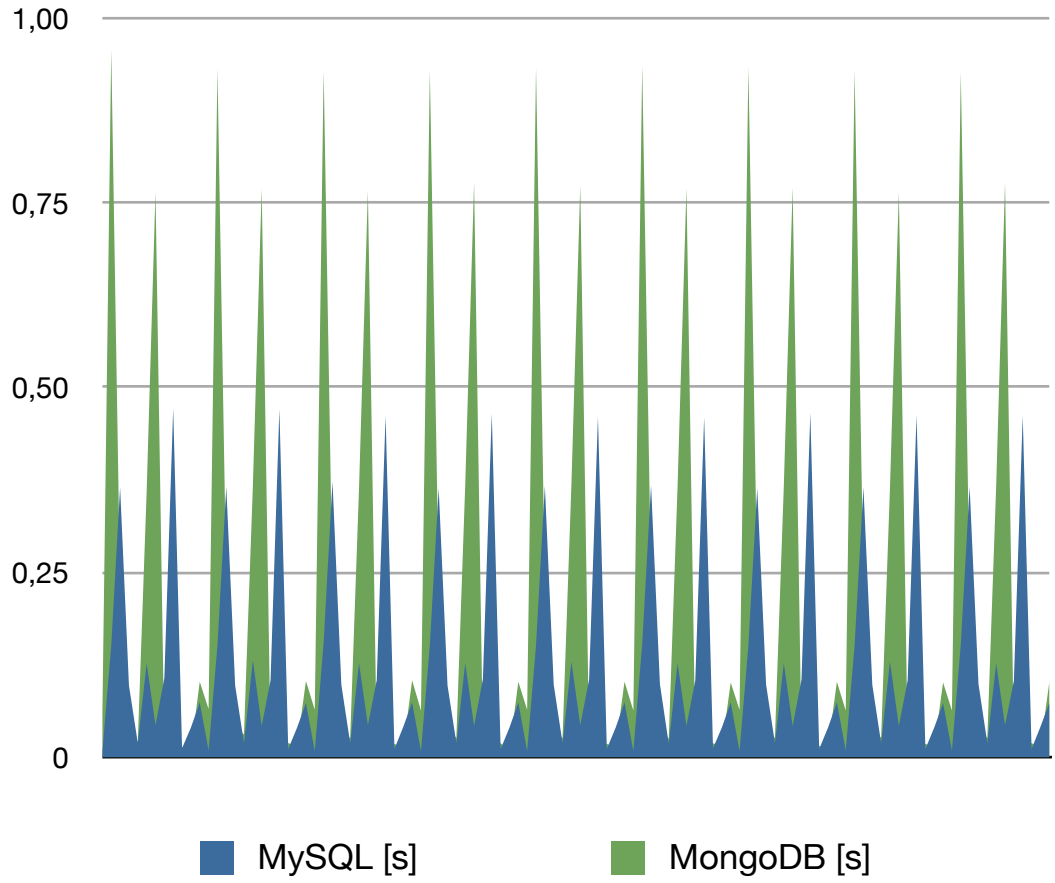
```
SELECT * FROM `article` WHERE `Title` LIKE '%Suchbegriff%‘ LIMIT 1
```

```
db.articles.find( { title : /Suchbegriff/i } )
```

Titel	MySQL [s]	MongoDB [s]
Psychologie	0,01	0,06
Sommerzeit	0,15	0,96
Waschmaschine	0,37	0,23
Kaffee	0,10	0,03
Flugzeug	0,02	0,02
Bill_Clinton	0,13	0,36
SQL	0,04	0,77
Klavier	0,11	0,02
Sofa	0,47	0,01
Berlin	0,01	0,01
Sonne	0,04	0,01
Radioaktivität	0,07	0,10
Psychologie	0,01	0,07
Sommerzeit	0,15	0,93
Waschmaschine	0,37	0,24
Kaffee	0,10	0,04
Flugzeug	0,02	0,03
Bill_Clinton	0,13	0,37
SQL	0,04	0,77
Klavier	0,10	0,02
Sofa	0,47	0,01
Berlin	0,01	0,02
Sonne	0,04	0,02
Radioaktivität	0,07	0,10
Psychologie	0,01	0,06
Sommerzeit	0,16	0,93
Waschmaschine	0,37	0,24
Kaffee	0,10	0,05
Flugzeug	0,02	0,03
Bill_Clinton	0,13	0,36
SQL	0,04	0,77
Klavier	0,10	0,02
Sofa	0,46	0,01
Berlin	0,01	0,02
Sonne	0,04	0,01
Radioaktivität	0,07	0,10
Psychologie	0,01	0,06
Sommerzeit	0,15	0,93
Waschmaschine	0,36	0,24
Kaffee	0,10	0,04

Titel	MySQL [s]	MongoDB [s]
Flugzeug	0,02	0,03
Bill_Clinton	0,13	0,37
SQL	0,04	0,78
Klavier	0,11	0,02
Sofa	0,47	0,01
Berlin	0,01	0,02
Sonne	0,04	0,01
Radioaktivität	0,07	0,10
Psychologie	0,01	0,06
Sommerzeit	0,15	0,93
Waschmaschine	0,37	0,24
Kaffee	0,10	0,04
Flugzeug	0,02	0,03
Bill_Clinton	0,13	0,37
SQL	0,04	0,77
Klavier	0,11	0,02
Sofa	0,46	0,01
Berlin	0,01	0,02
Sonne	0,04	0,01
Radioaktivität	0,07	0,10
Psychologie	0,01	0,06
Sommerzeit	0,15	0,94
Waschmaschine	0,37	0,24
Kaffee	0,10	0,04
Flugzeug	0,02	0,03
Bill_Clinton	0,13	0,37
SQL	0,04	0,77
Klavier	0,10	0,02
Sofa	0,46	0,01
Berlin	0,01	0,02
Sonne	0,04	0,01
Radioaktivität	0,07	0,10
Psychologie	0,01	0,06
Sommerzeit	0,15	0,93
Waschmaschine	0,36	0,24
Kaffee	0,10	0,04
Flugzeug	0,02	0,03
Bill_Clinton	0,13	0,37
SQL	0,04	0,77
Klavier	0,11	0,02
Sofa	0,47	0,01
Berlin	0,01	0,02
Sonne	0,04	0,01
Radioaktivität	0,07	0,10
Psychologie	0,01	0,06
Sommerzeit	0,15	0,93
Waschmaschine	0,37	0,24
Kaffee	0,10	0,04
Flugzeug	0,02	0,03
Bill_Clinton	0,13	0,37
SQL	0,04	0,77
Klavier	0,10	0,02
Sofa	0,46	0,01

Titel	MySQL [s]	MongoDB [s]
Berlin	0,01	0,02
Sonne	0,04	0,01
Radioaktivität	0,07	0,10
Psychologie	0,01	0,06
Sommerzeit	0,15	0,93
Waschmaschine	0,37	0,23
Kaffee	0,10	0,04
Flugzeug	0,02	0,03
Bill_Clinton	0,13	0,37
SQL	0,04	0,78
Klavier	0,10	0,02
Sofa	0,46	0,01
Berlin	0,01	0,02
Sonne	0,04	0,01
Radioaktivität	0,07	0,10
Airthm. Mittel	**0,13**	**0,22**



Auch bei der toleranten Suche ist MySQL nahezu unverändert schneller. Allerdings sei angemerkt, dass, wie bereits im vorherigen Abschnitt erwähnt, die MongoDB mit regulären Ausdrücken sucht. Zwar bieten sich durch reguläre Ausdrücke wesentlich flexiblere Abfragemöglichkeiten, aber auf Kosten der Performance.

5.3.2. Nach Untertitel suchen [76]

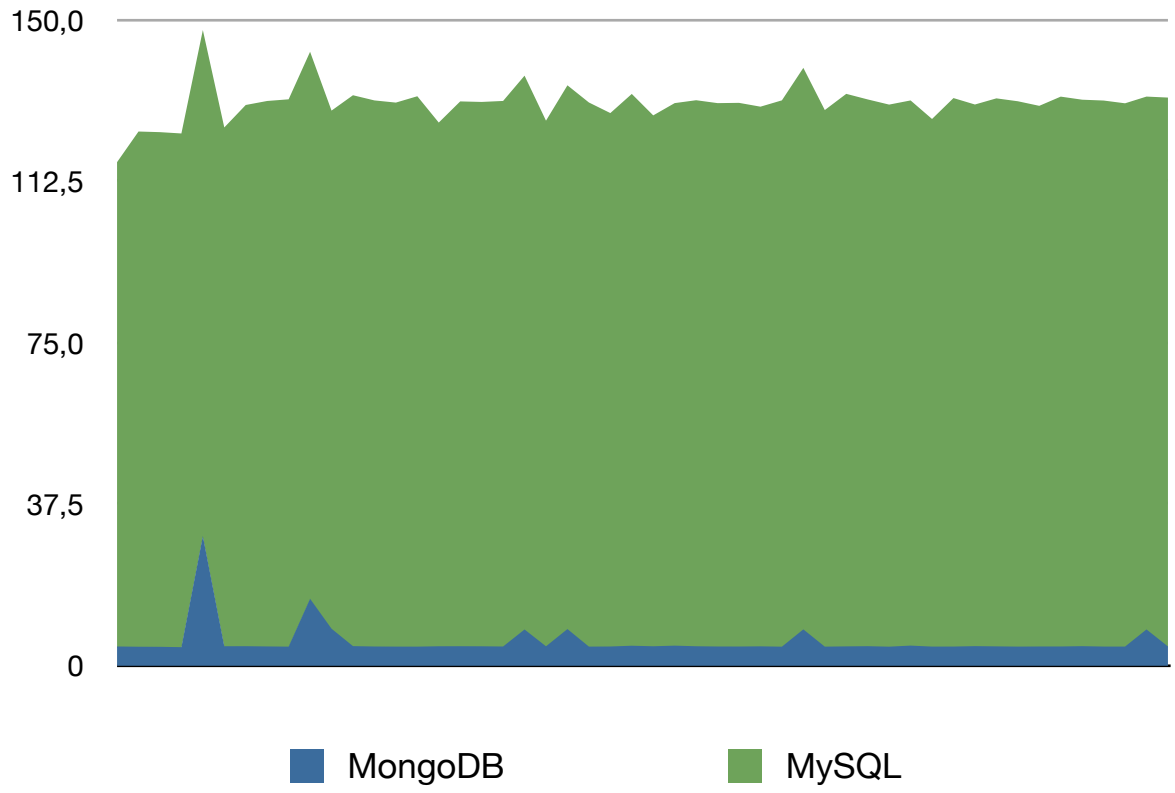
```
SELECT COUNT(*) FROM articles WHERE Content LIKE "%== Suchbegriff ==%"
```

```
db.articles.find({"sections.subtitle":"Suchbegriff"}).count()
```

	MySQL [s]	**gefunden**	**MongoDB [s]**	**gefunden**
Geschichte	112,54	141340	4,486	138190
	119,72		4,425	
	119,59		4,407	
	119,37		4,322	
	117,56		30,159	
Referenzen	120,55	4115	4,539	3993
	125,76		4,55	
	126,75		4,492	
	127,18		4,447	
	127,11		15,562	
Einflüsse	120,39	154	8,613	114
	128,01		4,565	
	126,89		4,48	
	126,41		4,456	
	127,89		4,45	
Kritik	121,69	6717	4,533	6207
	126,60		4,541	
	126,49		4,528	
	126,77		4,478	
	128,69		8,431	
Stil	122,14	1744	4,524	1666
	126,35		8,53	
	126,44		4,454	
	123,95		4,48	
	128,22		4,658	
Diskografie	123,36	12979	4,533	12886
	126,07		4,675	
	126,88		4,537	
	126,26		4,481	
	126,33		4,475	
Stil	125,40		4,517	
	126,95		4,438	
	130,48		8,454	
	124,69		4,443	

[76] Obwohl es einen geringfügigen Unterschied bei der Gesamtanzahl aller Datensätze für den hier ausgeführten Benchmarktest gibt (2287448 bei MySQL und 2288496 bei MongoDB), ist der merkliche Unterschied der gefundenen Artikel auf die Auswahlkriterien der Datenbankabfragen zurückzuführen. Bei einem %= = Titel = =% - Statement werden auch Untertitel niedrigeren Grades (bei Wikipedia = = = Titel = = = und = = = = Unteruntertitel = = = = =) gezählt, ist aber aufgrund des eingeschränkten LIKE-Statements nicht anders machbar.

	128,39		4,507	
Werke	127,05	32492	4,551	32256
	125,96		4,44	
	126,68		4,687	
	122,60		4,46	
	127,45		4,467	
Vorgeschichte	125,86	3395	4,562	2577
	127,34		4,509	
	126,72		4,456	
	125,63		4,482	
	127,77		4,483	
Microsoft	126,98	7	4,558	5
	126,88		4,464	
	126,24		4,461	
	123,82		8,444	
	127,54		4,492	
Arithm. Mittel	**125,25**		**5,63372**	



In diesem Geschwindigkeitstest macht sich die Auswirkung der strukturellen Flexibilität der Dokumentendatenbank bemerkbar. Da auch die Untertitel in der MongoDB als Schlüssel festgelegt und indexiert wurden, sind im Gegensatz zur eine Suche in einem großen Textfeld schnellere Suchergebnisse die Folge.

5.4. Auswertung und Interpretation der Messungen

Die Geschwindigkeitstest variieren teilweise sehr stark, was einen Vergleich schwierig macht. Obwohl zu jedem Test über zehn Durchläufe pro Suchbegriff gemessen wurden um konstantere

Resultate zu erhalten, kann man zwischen den beiden Datenbanksystemen in Sachen Geschwindigkeit keinen klaren Gewinner festlegen. Denn zum einen verwenden beide Systeme völlig unterschiedliche Treiber, Abfragesprachen und API‘s - zum anderen spielen beide Systeme ihre Stärken und Schwächen erst bei hoher Nutzlast und verteilten Systemen aus; ein Testszenario, dass den Umfang der Bachelorarbeit gesprengt hätte.

Trotzdem lassen sich meiner Meinung nach die Vorteile einer dokumentenbasierten Datenbank wie im Falle der Wikipedia erkennen. Durch die flexible Datenstruktur kann der Fokus besser auf wichtige Felder gelegt und somit eine bessere Performance und Organisation gewährleistet werden. Das eindeutigste Beispiel der Resultate - die Geschwindigkeitsmessung der Untertitelsuche - kann als Beleg dafür betrachtet werden.

Ein weiterer Punkt sind die jeweiligen Entwicklungsstände beider Datenbanksysteme. So hatte ich Anfangs sowohl eine frühere MySQL Version (5.1) als auch eine frühere MongoDB Version (1.6) verwendet. Die Ergebnisse fielen dort in nahezu jeder Hinsicht schlechter für MySQL aus - eine verbesserte Multi-Core-Unterstützung als auch überarbeitete Storage-Engines könnten u.a. die Ursachen für den Performanceschub sein [77].

Doch auch bei der MongoDB wirkt sich, ähnlich wie bei einer relationalen Datenbank, eine Aufteilung des Dokuments in mehrere Datensätze positiv auf die Performance aus. So hatte ich während der Importphase verschiedene Ansätze getestet. Ein Szenario war die Aufteilung der Absätze in einzelne Datensätze (hier Dokumente genannt) - mit dem Ergebnis, dass die (nicht indexierten) Textfelder wesentlich schneller durchsucht wurden als ein Dokument, dass alle Absätze enthält.

[77] http://ronaldbradford.com/blog/five-reasons-to-upgrade-to-mysql-5-5-2010-12-15/

6. Aufbau der entwickelten Programme

6.1. Verwendete Programme

Insgesamt wurden zwei in JAVA geschriebene Programme verwendet: Der modifizierte mwdumper und der Benchmarktest.

Der mwdumper liest den offiziellen XML-Dump der Wikipedia ein und schreibt ihn sowohl in die MySQL-Datenbank als auch in die MongoDB.

6.2. wikipedia2nosql

Der mwdumper verarbeitet den XML-Dump mit Hilfe des SAX-Parsers. Normalerweise wird über den Stdout [78] ein MySQL-Import ausgegeben, der dann üblicherweise in eine .sql-Datei geschrieben wird.

Im folgenden wird der Vorgang zu den wichtigsten Punkten erläutert. Der vollständige Quellcode mit Kommentaren zu den modifizierten Klassen befindet sich im Anhang.

Als empfohlener Kommanozeilenaufruf wird empfohlen: [79]

```
java -Dfile.encoding=U8 -Xms1024m -Xmx2048m -XX:NewSize=256m -
XX:MaxNewSize=256m -XX:+UseConcMarkSweepGC -XX:+UseParNewGC -
XX:PermSize=128m -XX:MaxPermSize=128m -jar ./NoSQLDumper.jar --format=sql:
1.5 ~/dewiki-latest-pages-articles.xml
```

Wobei **~/dewiki-latest-pages-articles.xml** für den Pfad zum XML-Dump steht.

Das Importprogramm liest die XML stückchenweise mit Hilfe des SAX-Parsers ein und trennt jeden Artikel nach seinen Hauptfeldern (Titel, Kommentar, Inhalt und - falls vorhanden - Titel des Artikels, auf den referenziert wird). Zudem wird der Inhalt für die MongoDB zusätzlich in Abschnitte erster Ebene unterteilt. Das heißt, alle Kapitel eines Artikel werden in einer Arraystruktur gespeichert:

```
{ "title" : "Titel des Artikels",
       "content" : {
            "section" : {
                 "subtitle" : "Geschichte",
                 "content" : "lorem ipsum...",
```

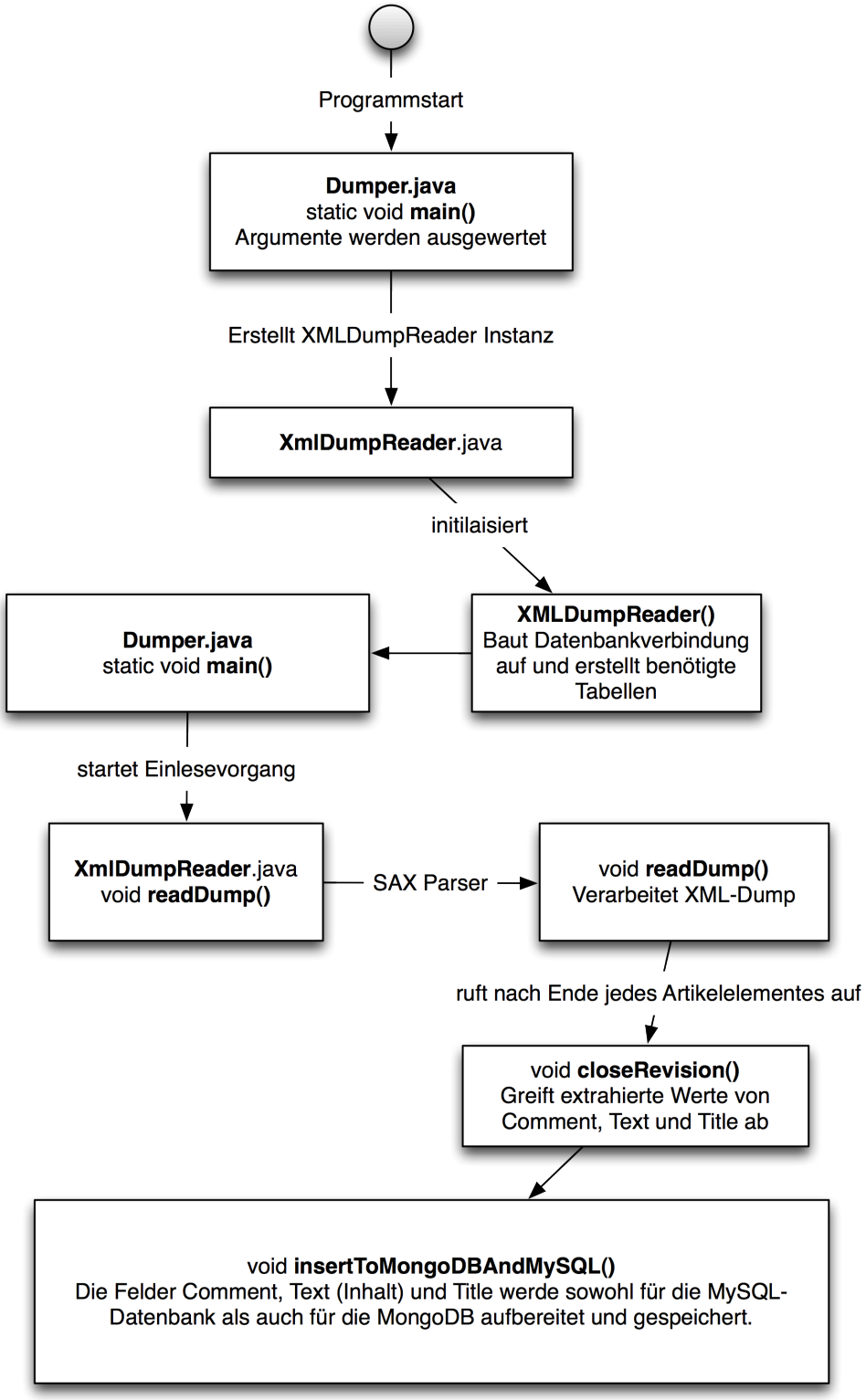
[78] Standard Ausgabestream (hier: auf der Konsole)

[79] Obwohl die ausführbare JAR-Datei hier NoSQLDumper heißt, handelt es sich um das Importprogramm

```
                },
                "section" : {
                      "subtitle" : "Literatur",
                      "content" : "lorem ipsum...",
                },
                ...
          }
    }
```

Anschließend wird jeder Artikel sowohl in die MySQL als auch in die MongoDB gespeichert. Dies erfolgt jeweils über die MySQL-JDBC-Treiber bzw. über die MongoDB-Treiber.

6.2.1. Vereinfachter Programmablaufsplan des Wikipedia-Imports [80]



[80] Modifiziertes UML-Aktionsdiagramm zur Beschreibung des Programmablaufs

6.3. Benchmarktest

Die Benchmarkapplikation läuft auf der Textkonsole und führt automatisierte Geschwindigkeitstests sowohl mit der MySQL-Datenbank als auch mit der MongoDB durch. Die Ergebnisse können in einem CSV-kompatiblen Format zurückgegeben werden.

Folgende Parameter können festgelegt werden: [81]

- **-h** für Hilfe
- **-v** für ausführliches Loggen (z.B. Datenbankbefehle ausgeben, Anzahl gefundener Artikel etc.)
- **--mongodb** es soll die MongoDB durchsucht werden
- **--mysql** es soll die MySQL-Datenbank durchsucht werden
- **--regex** Suche mit regulären Ausdrücken (MongoDB) bzw. LIKE-Statements (MySQL)
- **--exact** Such nach genauem Vorkommen (Pendant zu --regex)
- **-l{Zahl}** Gibt die Tiefe an, mit der rekursiv von dem Startdokument weitergesucht wird
- **-t{Titel}** Durchsucht das Feld Titel [82]
- **-s{Titel}** Durchsucht das Feld Untertitel, bzw. Überschrift der Absätze
- **-c{Titel}** Durchsucht das Feld Inhalt, also den kompletten Artikel
- **--silent** Sehr sparsame Textausgabe
- **--formatCSV** optionales Textformat für Ausgabe

Beispiel Konsolenparameter für einen ausführlichen Benchmarktest:

```
java –jar ./DatabasePerformance.jar –l100 –tAlfred_Hitchcock ––mongodb ––
mysql ––exact –v
```

Beispiel Konsolenparameter für einen Benchmarktest zur späteren Auswertung:

```
java –jar ./DatabasePerformance.jar –l100 –tAlfred_Hitchcock ––mongodb ––
mysql ––exact ––silent ––formatCSV
```

Mit Hilfe eines einfachen Skriptes (hier in der Skriptsprache Ruby) lässt sich auch dieser Vorgang nochmals stärker automatisieren:

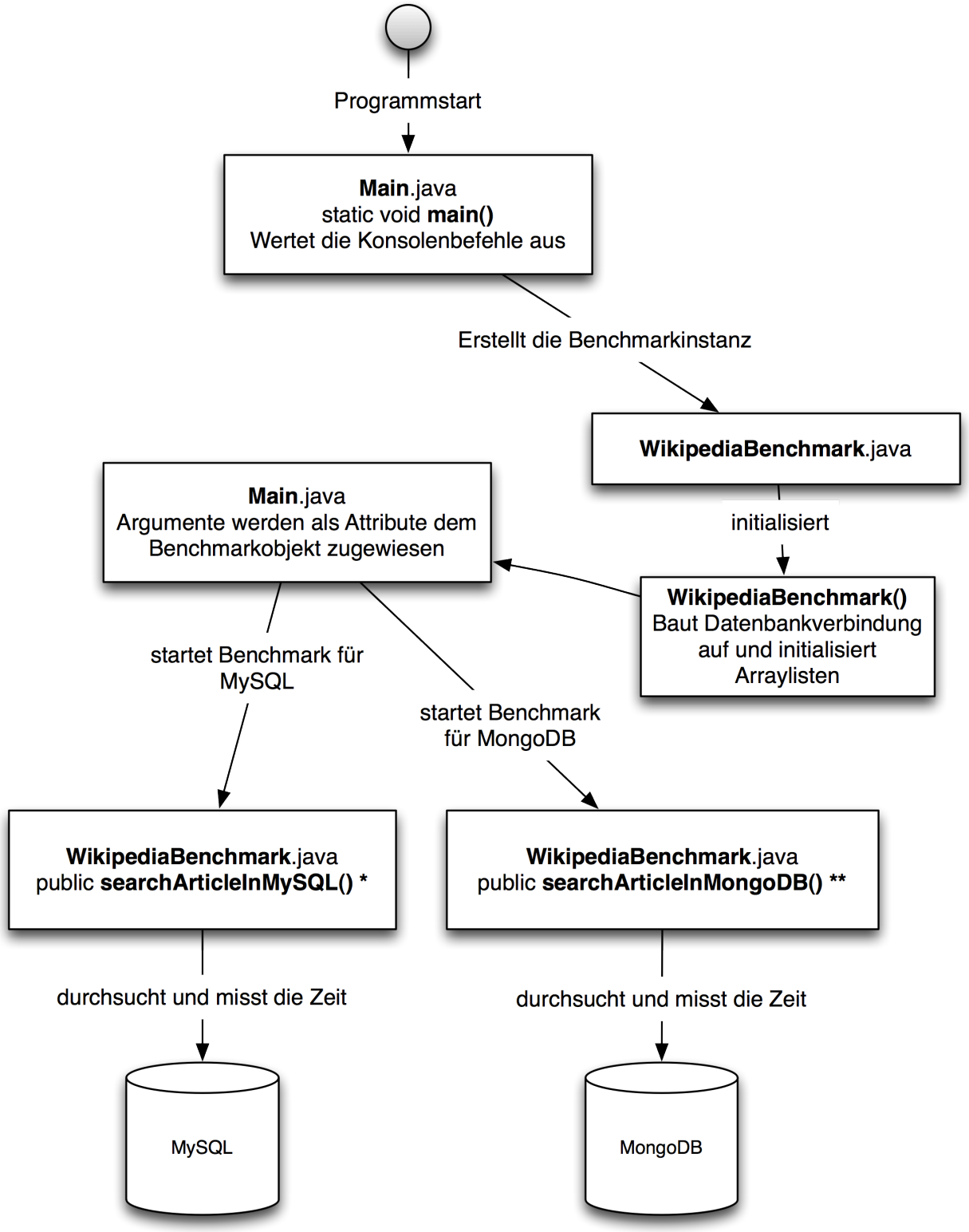
```
#!/usr/bin/env ruby
begin
  articles = "Psychologie, Sommerzeit, Waschmaschine, Kaffee, Flugzeug, Bill Clinton,
SQL, Klavier, Sofa, Berlin, Sonne, Radioaktivität"
```

[81] Beim aufrufen ohne Parameter werden mit Hilfe einer interaktiven Texteingabe alle benötigten Werte fetsgelegt

[82] Leerzeichen werden durch _ ersetzt; Beispiel: Coca_Cola

```
  articles = articles.split(/,\s*/)
  regularExpression = false;
  x = 10
  searchedField = "s";
  comment = "";
  limit = 1
  parameter = " --exact ";
  print "\nTitel;Befehl;Parameter;Limit;MySQL;NoSQL;"
  #wiederhole den test x-mal
  for i in 1..x do
    #arbeite jeden begriff einzeln ab
    articles.each{ |title|
      title = title.gsub(" ","_")
      #aufruf des java-programms
      command = "java -jar ~/NetBeansProjects/DatabasePerformance/dist/
DatabasePerformance.jar -l#{limit} -#{searchedField}#{title} --mongodb --mysql #
{parameter} --silent --formatCSV"
      #textausgabe, hier für csv
      print "\n#{title};-#{searchedField}#{title};#{parameter}#{comment};#{limit};"
      print `#{command}`
    }
  end
end
```

6.3.1. Vereinfachter Programmablauf der Benchmarkapplikation [83]



* startet den Timer [this.startTimer()], öffnet die MySQL-Datenbank [this.openConnection()], sucht die Artikel raus [this.findArticleInSQL()] und loggt die Zeitmessung

** startet den Timer [this.startTimer()], öffnet die MongoDB-Datenbank [Mongo.getDB()], sucht die Artikel raus [this.findArticleInNoSQL()] und loggt die Zeitmessung

[83] Modifiziertes UML-Aktionsdiagramm zur Beschreibung des Programmablaufs

7. Zusammenfassung

Die Arbeit mit der MongoDB hat mir durchweg gefallen. Anfangs ist es ungewohnt ohne SQL-Statements zu arbeiten, und nicht die gewohnte Tabellenansicht zu haben. Da man aber auch ohne ein ORM-Framework problemlos mit der MongoDB-API arbeiten kann, braucht man sich weniger Gedanken um datenbankspezifische Funktionen zu machen und kann sich mehr auf die Programmierung konzentrieren.

Der Performanceeindruck war bei beiden Datenbanksystemen sehr gut - obwohl bei beiden Systemen nicht alle Möglichkeiten der Geschwindigkeitsoptimierung ausgeschöpft wurden.

Aufgrund des schemafreien Arbeitens in der MongoDB muss man bei der Programmierung mehr auf die Einhaltung der Struktur achten und trägt mehr Verantwortung auf das Gewährleisten intakter Relationen. Ein Framework ist hierbei, wie allerdings auch beim Arbeiten mit MySQL, dringend empfohlen.

Insgesamt hat mich das MongoDB-Datenbankkonzept überzeugt - wenn auch nur bezogen auf bestimmte Anwendungsbereiche. Viele überflüssige Altlasten von relationalen Datenbanken wurden weggelassen und durch einen schlanken, spezialisierten Funktionsumfang ersetzt.

A. Anhang

A.A. Vollständiger Quellcode

- Original mwdumper: http://svn.wikimedia.org/svnroot/mediawiki/trunk/mwdumper/
- Modifizierter mwdumper: https://github.com/philipp1982/wikipedia2mongodbd
- Benchmarktest: https://github.com/philipp1982/Database-Benchmark

A.B. Der Modifizierte mwdumper [84]

wikipedia2nosql/Dumper.java

```
package org.mediawiki.dumper;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import java.sql.Connection;
import java.sql.DriverManager;
import java.util.zip.GZIPOutputStream;
import java.lang.ClassNotFoundException;

import java.text.ParseException;

import org.mediawiki.importer.*;


class Dumper {

public static void main(String[] args) throws IOException, ParseException {
                  InputStream input = null;
                  OutputWrapper output = null;
                  DumpWriter sink = null;
                  MultiWriter writers = new MultiWriter();
                  int progressInterval = 1000;

                  for (int i = 0; i < args.length; i++) {
                          String arg = args[i];
                          String[] bits = splitArg(arg);
                          if (bits != null) {
                                  String opt = bits[0], val = bits[1], param = bits[2];
                                  if (opt.equals("output")) {
                                          if (output != null) {
                                                  // Finish constructing the previous output...
                                                  if (sink == null)
                                                          sink = new XmlDumpWriter
(output.getFileStream());

                                                  writers.add(sink);
                                                  sink = null;
                                          }
```

[84] Alle von mir hinzugefügten / modifizierten Passagen sind fett-markiert

```
                                        output = openOutputFile(val, param);
                                } else if (opt.equals("format")) {
                                        if (output == null)
                                                output = new OutputWrapper
(Tools.openStandardOutput());
                                        if (sink != null)
                                                throw new IllegalArgumentException("Only one
format per output allowed.");
                                        sink = openOutputSink(output, val, param);
                                } else if (opt.equals("filter")) {
                                        if (sink == null) {
                                                if (output == null)
                                                        output = new OutputWrapper
(Tools.openStandardOutput());
                                                sink = new XmlDumpWriter(output.getFileStream
());
                                        }
                                        sink = addFilter(sink, val, param);
                                } else if (opt.equals("progress")) {
                                        progressInterval = Integer.parseInt(val);
                                } else if (opt.equals("quiet")) {
                                        progressInterval = 0;
                                } else {
                                        throw new IllegalArgumentException("Unrecognized option "
+ opt);
                                }
                        } else if (arg.equals("-")) {
                                if (input != null)
                                        throw new IllegalArgumentException("Input already set;
can't set to stdin");
                                input = Tools.openStandardInput();
                        } else {
                                if (input != null)
                                        throw new IllegalArgumentException("Input already set;
can't set to " + arg);
                                input = Tools.openInputFile(arg);
                        }
                }

                if (input == null)
                        input = Tools.openStandardInput();
                if (output == null)
                        output = new OutputWrapper(Tools.openStandardOutput());
                // Finish stacking the last output sink
                if (sink == null)
                        sink = new XmlDumpWriter(output.getFileStream());
                writers.add(sink);

                DumpWriter outputSink = (progressInterval > 0)
                                ? (DumpWriter)new ProgressFilter(writers, progressInterval)
                                : (DumpWriter)writers;

                XmlDumpReader reader = new XmlDumpReader(input, outputSink);
                reader.readDump();
        }
}
```

wikipedia2nosql/XMLDumpReader.java

```
package org.mediawiki.importer;

import java.io.IOException;
import java.io.InputStream;
import java.security.NoSuchAlgorithmException;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.HashMap;
import java.util.Map;
import java.util.TimeZone;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import com.mysql.jdbc.Connection;
import com.mysql.jdbc.Statement;

import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

//added by philipp.staender@gmail.com
//wird benoetigt fuer monogdb+mysql zugriff
import com.mongodb.BasicDBObject;
import com.mongodb.Mongo;
import com.mongodb.DB;
import com.mongodb.DBCollection;
import com.mongodb.DBCursor;
import com.mongodb.DBObject;

import java.security.MessageDigest;
import java.util.ArrayList;
import java.util.List;
import java.util.Set;

import java.util.regex.Matcher;
import java.util.regex.Pattern;
import org.bson.types.ObjectId;

public class XmlDumpReader  extends DefaultHandler {
          InputStream input;
          DumpWriter writer;

          private char[] buffer;
          private int len;
          private boolean hasContent = false;
          private boolean deleted = false;

          Siteinfo siteinfo;
          Page page;
          boolean pageSent;
          Contributor contrib;
          Revision rev;
          int nskey;
```

```
            boolean abortFlag;

        private DBCollection mongodbArticles = null;
        private DBCollection mongodbTextindexes = null;
        private Connection mysqlConnection = null;
        private String mysqlUsername = "root";
        private String mysqlPassword = "root";
        private String mysqlHost = "localhost";
        private int mysqlPort = 3306;
        private String mysqlDatabase = "nosql";
        private String mongodbDatabasename = "wikipedia";
        private int lastInsertedArticleID = 0;
        private static boolean generateTextIndizes = false;

          /**
           * Initialize a processor for a MediaWiki XML dump stream.
           * Events are sent to a single DumpWriter output sink, but you
           * can chain multiple output processors with a MultiWriter.
           * @param inputStream Stream to read XML from.
           * @param writer Output sink to send processed events to.
           */
          public XmlDumpReader(InputStream inputStream, DumpWriter writer) {
                  input = inputStream;
                  this.writer = writer;
                  buffer = new char[4096];
                  len = 0;
                  hasContent = false;

                try {
                    //open connection to monogdb an get collections
                    Mongo m = new Mongo();
                    System.out.println("Initialisiere mongodb Datenbank '"+this.mongodbDatabasename
+"'");
                    m.dropDatabase(this.mongodbDatabasename);
                    DB db = m.getDB(this.mongodbDatabasename);
                    System.out.println("Setze Indizes in mongodb collections...");
                    this.mongodbArticles = db.getCollection("articles");
                    this.mongodbArticles.ensureIndex("title");
                    if (XmlDumpReader.generateTextIndizes) {
                        this.mongodbTextindexes = db.getCollection("textindex");
                        this.mongodbTextindexes.ensureIndex("title");
                        this.mongodbTextindexes.ensureIndex("article");
                        this.mongodbTextindexes.ensureIndex("links");
                        this.mongodbTextindexes.ensureIndex("sections.subtitle");
                    }
                } catch (Exception e) {
                    System.err.println("Fehler beim Initialisieren  der mongodb: "+e.getMessage());
                    System.err.println("Applikation wird beendet");
                    System.exit(0);
                }
                //open connection to mysql
                String dbURL = "jdbc:mysql://"+this.mysqlHost+":"+this.mysqlPort
+"/"+this.mysqlDatabase;
             try {
                this.mysqlConnection = (Connection) DriverManager.getConnection
(dbURL,this.mysqlUsername,this.mysqlPassword);
                //Prepare tables for database
                System.out.println("Initialisiere notwendige Tabellen in mysql Datenbank
'"+this.mysqlDatabase+"'");
                Statement stmt = (Statement) this.mysqlConnection.createStatement();
                String sql = "DROP TABLE IF EXISTS `articles`;";
```

```
                stmt.executeUpdate(sql);
                sql = "CREATE TABLE IF NOT EXISTS `articles` (  `ID` int(11) NOT NULL
AUTO_INCREMENT,  `MongoID` varchar(255) NOT NULL,  `Title` varchar(255) NOT NULL,  `Redirect`
varchar(255) NOT NULL,  `Comment` text NOT NULL,  `Content` longtext NOT NULL, `Links` text NOT
NULL, PRIMARY KEY (`ID`),  KEY `ArticleTitle` (`Title`)) ENGINE=MyISAM  DEFAULT CHARSET=utf8;";
                stmt.executeUpdate(sql);
                if (XmlDumpReader.generateTextIndizes) {
                    sql = "DROP TABLE IF EXISTS `textindex`;";
                    stmt.executeUpdate(sql);
                    sql = "CREATE TABLE IF NOT EXISTS `textindex` (  `ID` int(11) NOT NULL
AUTO_INCREMENT,  `ArticleID` int(11) NOT NULL,  `MongoID` varchar(255) DEFAULT NULL,  `Sort` int(11)
NOT NULL, `Title` varchar(255) DEFAULT NULL,  `Text` longtext,  `Links` text NOT NULL,  PRIMARY KEY
(`ID`),  KEY `ArticleID` (`ArticleID`)) ENGINE=MyISAM  DEFAULT CHARSET=utf8;";
                    stmt.executeUpdate(sql);
                }
              } catch (SQLException ex) {
                // handle any errors
                System.err.println("Fehler beim Initialisieren der mysql Datenbank...");
                System.err.println("SQLException: " + ex.getMessage());
                System.err.println("SQLState: " + ex.getSQLState());
                System.err.println("VendorError: " + ex.getErrorCode());
                System.err.println("Applikation wird beendet");
                System.exit(0);
              }
            }

            /**
             * Reads through the entire XML dump on the input stream, sending
             * events to the DumpWriter as it goes. May throw exceptions on
             * invalid input or due to problems with the output.
             * @throws IOException
             */
            public void readDump() throws IOException {
                    try {
                            SAXParserFactory factory = SAXParserFactory.newInstance();
                            SAXParser parser = factory.newSAXParser();

                            parser.parse(input, this);
                    } catch (ParserConfigurationException e) {
                            throw (IOException)new IOException(e.getMessage()).initCause(e);
                    } catch (SAXException e) {
                            throw (IOException)new IOException(e.getMessage()).initCause(e);
                    }
                    writer.close();
            }

            /**
             * Request that the dump processing be aborted.
             * At the next element, an exception will be thrown to stop the XML parser.
             * @fixme Is setting a bool thread-safe? It should be atomic...
             */
            public void abort() {
                    abortFlag = true;
            }

            // --------------------------
            // SAX handler interface methods:

            private static final Map startElements = new HashMap(64);
            private static final Map endElements = new HashMap(64);
            static {
```

```
                startElements.put("revision","revision");
                startElements.put("contributor","contributor");
                startElements.put("page","page");
                startElements.put("mediawiki", "mediawiki");
                startElements.put("siteinfo","siteinfo");
                startElements.put("namespaces","namespaces");
                startElements.put("namespace","namespace");

                endElements.put("ThreadSubject","ThreadSubject");
                endElements.put("ThreadParent","ThreadParent");
                endElements.put("ThreadAncestor","ThreadAncestor");
                endElements.put("ThreadPage","ThreadPage");
                endElements.put("ThreadID","ThreadID");
                endElements.put("ThreadSummaryPage","ThreadSummaryPage");
                endElements.put("ThreadAuthor","ThreadAuthor");
                endElements.put("ThreadEditStatus","ThreadEditStatus");
                endElements.put("ThreadType","ThreadType");
                endElements.put("base","base");
                endElements.put("case","case");
                endElements.put("comment","comment");
                endElements.put("contributor","contributor");
                endElements.put("generator","generator");
                endElements.put("id","id");
                endElements.put("ip","ip");
                endElements.put("mediawiki", "mediawiki");
                endElements.put("minor","minor");
                endElements.put("namespaces","namespaces");
                endElements.put("namespace","namespace");
                endElements.put("page","page");
                endElements.put("restrictions","restrictions");
                endElements.put("revision","revision");
                endElements.put("siteinfo","siteinfo");
                endElements.put("sitename","sitename");
                endElements.put("text","text");
                endElements.put("timestamp","timestamp");
                endElements.put("title","title");
                endElements.put("username","username");
        }

        public void startElement(String uri, String localname, String qName, Attributes
attributes) throws SAXException {
                // Clear the buffer for character data; we'll initialize it
                // if and when character data arrives -- at that point we
                // have a length.
                len = 0;
                hasContent = false;

                if (abortFlag)
                        throw new SAXException("XmlDumpReader set abort flag.");

                // check for deleted="deleted", and set deleted flag for the current element.
                String d = attributes.getValue("deleted");
                deleted = (d!=null && d.equals("deleted"));

                try {
                        qName = (String)startElements.get(qName);
                        if (qName == null)
                                return;
                        // frequent tags:
                        if (qName == "revision") openRevision();
                        else if (qName == "contributor") openContributor();
```

```
                        else if (qName == "page") openPage();
                        // rare tags:
                        else if (qName == "mediawiki") openMediaWiki();
                        else if (qName == "siteinfo") openSiteinfo();
                        else if (qName == "namespaces") openNamespaces();
                        else if (qName == "namespace") openNamespace(attributes);
                } catch (IOException e) {
                        throw new SAXException(e);
                }
        }

        public void characters(char[] ch, int start, int length) {
                if (buffer.length < len + length) {
                        int maxlen = buffer.length * 2;
                        if (maxlen < len + length)
                                maxlen = len + length;
                        char[] tmp = new char[maxlen];
                        System.arraycopy(buffer, 0, tmp, 0, len);
                        buffer = tmp;
                }
                System.arraycopy(ch, start, buffer, len, length);
                len += length;
                hasContent = true;
        }

        public void endElement(String uri, String localname, String qName) throws SAXException {
                try {
                        qName = (String)endElements.get(qName);
                        if (qName == null)
                                return;
                        // frequent tags:
                        if (qName == "id") readId();
                        else if (qName == "revision") closeRevision();
                        else if (qName == "timestamp") readTimestamp();
                        else if (qName == "text") readText();
                        else if (qName == "contributor") closeContributor();
                        else if (qName == "username") readUsername();
                        else if (qName == "ip") readIp();
                        else if (qName == "comment") readComment();
                        else if (qName == "minor") readMinor();
                        else if (qName == "page") closePage();
                        else if (qName == "title") readTitle();
                        else if (qName == "restrictions") readRestrictions();
                        // rare tags:
                        else if (qName.startsWith("Thread")) threadAttribute(qName);
                        else if (qName == "mediawiki") closeMediaWiki();
                        else if (qName == "siteinfo") closeSiteinfo();
                        else if (qName == "sitename") readSitename();
                        else if (qName == "base") readBase();
                        else if (qName == "generator") readGenerator();
                        else if (qName == "case") readCase();
                        else if (qName == "namespaces") closeNamespaces();
                        else if (qName == "namespace") closeNamespace();
//                      else throw(SAXException)new SAXException("Unrecognised "+qName
+"(substring "+qName.length()+qName.substring(0,6)+")");
                } catch (IOException e) {
                        throw (SAXException)new SAXException(e.getMessage()).initCause(e);
                }
        }

        // ----------
```

```
		void threadAttribute(String attrib) throws IOException {
			if(attrib.equals("ThreadPage")) // parse title
				page.DiscussionThreadingInfo.put(attrib, new Title(bufferContents(),
	siteinfo.Namespaces));
			else
				page.DiscussionThreadingInfo.put(attrib, bufferContents());
		}

		void openMediaWiki() throws IOException {
			siteinfo = null;
			writer.writeStartWiki();
		}

		void closeMediaWiki() throws IOException {
			writer.writeEndWiki();
			siteinfo = null;
		}

		// ------------------

		void openSiteinfo() {
			siteinfo = new Siteinfo();
		}

		void closeSiteinfo() throws IOException {
			writer.writeSiteinfo(siteinfo);
		}

		private String bufferContentsOrNull() {
			if (!hasContent) return null;
			else return bufferContents();
		}

		private String bufferContents() {
			return len == 0 ? "" : new String(buffer, 0, len);
		}

		void readSitename() {
			siteinfo.Sitename = bufferContents();
		}

		void readBase() {
			siteinfo.Base = bufferContents();
		}

		void readGenerator() {
			siteinfo.Generator = bufferContents();
		}

		void readCase() {
			siteinfo.Case = bufferContents();
		}

		void openNamespaces() {
			siteinfo.Namespaces = new NamespaceSet();
		}

		void openNamespace(Attributes attribs) {
			nskey = Integer.parseInt(attribs.getValue("key"));
		}
```

```
            void closeNamespace() {
                    siteinfo.Namespaces.add(nskey, bufferContents());
            }

            void closeNamespaces() {
                    // NOP
            }

            // ------------

            void openPage() {
                    page = new Page();
                    pageSent = false;
            }

            void closePage() throws IOException {
                    if (pageSent)
                            writer.writeEndPage();
                    page = null;
            }

            void readTitle() {
                    page.Title = new Title(bufferContents(), siteinfo.Namespaces);
            }

            void readId() {
                    int id = Integer.parseInt(bufferContents());
                    if (contrib != null)
                            contrib.Id = id;
                    else if (rev != null)
                            rev.Id = id;
                    else if (page != null)
                            page.Id = id;
                    else
                            throw new IllegalArgumentException("Unexpected <id> outside a <page>,
    <revision>, or <contributor>");
            }

            void readRestrictions() {
                    page.Restrictions = bufferContents();
            }

            // ------

            void openRevision() throws IOException {
                    if (!pageSent) {
                            writer.writeStartPage(page);
                            pageSent = true;
                    }

                    rev = new Revision();
            }

        void closeRevision() throws IOException {
            //greife des text ab, und mache ein eigens (einfacheres insert)
            String comment = "'";
            if (rev.Comment!=null) comment = sqlEscape(rev.Comment);
            String text = "'";
            if (rev.Text!=null) text = sqlEscape(rev.Text);
            String title = "'";
```

```
        if (page.Title!=null) title = sqlEscape(page.Title.toString());
        insertToMongoDBAndMySQL();
        rev = null;
      }

    static String generateHashForID(String text) {
        //erstelle hash
        try {
          MessageDigest sha = MessageDigest.getInstance("SHA1");
          byte[] hash = sha.digest(text.getBytes());

          StringBuilder sb = new StringBuilder();
          for (int i = 0; i < hash.length; ++i) {
             sb.append(
                 Integer.toHexString(
                     (hash[i] & 0xFF) | 0x100
                 ).toLowerCase().substring(1,3)
             );
          }
          return sb.toString();
        } catch (Exception e) {
          System.out.println("Error:"+e.getMessage());
          return "";
        }
    }

    void insertToMongoDBAndMySQL() {
      String comment = "";
      if (rev.Comment!=null) comment = rev.Comment;
      String title = "";
      if (page.Title!=null) title = page.Title.toString();

      String text = "";
      if (rev.Text!=null) text = rev.Text;
      text = text.replaceAll("(?s)<!--.*?-->", "");

      //Der erste Absatz erhält einfach nochmal den Artikeltitel
      text = "\n== "+title+" ==\n \n"+text;

      //unterteile text in unterüberschriften
      String expression = "\\s+\\=\\=\\s+.+\\s+\\=\\=\\s+";
      Matcher match = Pattern.compile(expression).matcher(text);
      String[] splittedText = text.split(expression);
      ArrayList<String> subtitles = new ArrayList<String>();
      String subtitle = "";
      while (match.find()) {
        subtitle = match.group().trim();
        //entferne == ... == vom Titel
        subtitles.add(subtitle.substring(3,subtitle.length()-3));
      }

      try {

          DBCollection article;
          article = this.mongodbArticles;
          DBCollection textindex;
          textindex = this.mongodbTextindexes;

          BasicDBObject doc = new BasicDBObject();
          doc.put("title", title);
          doc.put("comment", comment);
```

```
            String mongoid = XmlDumpReader.generateHashForID(text);
            doc.put("_id",mongoid);
            BasicDBObject textindizies = new BasicDBObject();
            BasicDBObject content = new BasicDBObject();
            BasicDBObject link = new BasicDBObject();
            ArrayList<Object> paragraphs = new ArrayList<Object>();
            ArrayList<String> links = new ArrayList<String>();

            int sectionCount = -2;
            long textindexCount = 0;
            String sqlLinkValue = "";
            Connection mysqlConnection = null;
            String redirect = "";

            //ist artikel ein redirect?
            if (rev.Text.trim().toLowerCase().matches("\\A\\#(redirect|weiterleitung)\\s.*")) {
                String[] splittedRedirect = rev.Text.split("\\#(redirect|weiterleitung|Weiterleitung|
    WEITERLEITUNG|Redirect|REDIRECT)\\s+\\[\\[");
                redirect = splittedRedirect[1].trim().substring(0, splittedRedirect[1].length()-2);
            }

            //jedes einzelene unterkapitel wird zusaetzlich seperat gespeichert
            for (String string : splittedText) {
                sectionCount++;
                try {
                  subtitle = subtitles.get(sectionCount);
                } catch (Exception e) {
                  subtitle = "";
                }
                //nur hinzufügen, wenn text vorhanden ist
                if (string.trim().length()>0) {
                    BasicDBObject paragraph = new BasicDBObject();
                    textindexCount++;
                    if (redirect.length()>0) {
                        doc.put("redirect", redirect);
                        System.out.println(title+" => "+redirect);
                    } else {


                        String textindexMongoID=XmlDumpReader.generateHashForID(text
    +"fortextsearch"+String.valueOf(sectionCount));
                        //erstelle unterteilungen, hier kapitel genannt
                        //jedes kapitel wird nochmal in eine eigene collection gesetzt, für
    Volltextsuche
                        if (XmlDumpReader.generateTextIndizes) {
                            textindizies.put("article",title);
                            textindizies.put("order", (int) textindexCount);
                            textindizies.put("title", subtitle);
                            textindizies.put("text",string);
                            textindizies.put("_id",textindexMongoID);
                        }

                        paragraph.put("subtitle",subtitle);
                        paragraph.put("content", string);
                        paragraphs.add(paragraph);


                        //füge Links hinzu + notiere alle verlinkungen
                        String linkExpression = "\\[\\[[0-9\\s\\'\\\"\\.\\-\\_\\p{L}]+\\]\\]";
                        Matcher matchLinks = Pattern.compile(linkExpression).matcher(string);
                        String[] splittedLinks = text.split(linkExpression);
```

```
                    ArrayList<String> linksParagraph = new ArrayList<String>();
                    String linkText = "";
                    int linksCount = -1;

                    while (matchLinks.find()) {
                        linksCount++;
                        linkText = matchLinks.group().trim();
                        //entferne [[ ... ]] vom Titel
                        linkText = linkText.substring(2,linkText.length()-2);
                        links.add(linkText);
                        linksParagraph.add(linkText);
                        sqlLinkValue=sqlLinkValue+","+linkText;
                    }

                    //mysql + mongodb insert für unterkapitel für textindexierung
                    if (XmlDumpReader.generateTextIndizes) {
                        try {
                            mysqlConnection = this.mysqlConnection;
                            Statement stmt = (Statement) mysqlConnection.createStatement();
                            String sql = "INSERT INTO  `textindex` (`ID` , `ArticleID`, `MongoID` ,
`Sort`, `Title` , `Text` , `Links`)"
                                    + "VALUES ("
                                    + "NULL ,  '"+lastInsertedArticleID+"', \""+textindexMongoID
+"\", "+(int)+textindexCount+", "+XmlDumpReader.sqlEscape(subtitle)+", "+XmlDumpReader.sqlEscape
(string)+",  "+XmlDumpReader.sqlEscape(sqlLinkValue)+" "
                                    + ");";
                            stmt.executeUpdate(sql);
                        } catch(SQLException e) {
                            System.err.println("Fehler beim mysql insert von textindex:
"+e.getMessage());
                        }
                        textindizies.put("link",links);
                        textindex.insert(textindizies);
                    }









                }
            }
        }

            //mysql insert kompletter artikel
            try {
                mysqlConnection = this.mysqlConnection;
                Statement stmt = (Statement) mysqlConnection.createStatement();
                if (sqlLinkValue.length()>0) sqlLinkValue=sqlLinkValue.substring(1);
                String sql = "INSERT INTO  `articles` (`ID` , `MongoID` , `Title` , `Redirect` ,
`Comment` , `Content`, `Links` )"
                        + "VALUES ("
                        + "NULL ,  \""+mongoid+"\",  "+XmlDumpReader.sqlEscape(title)+",
'"+redirect+"',  "+XmlDumpReader.sqlEscape(comment)+",  "+XmlDumpReader.sqlEscape(rev.Text)+",
"+XmlDumpReader.sqlEscape(sqlLinkValue)+" "
                        + ");";
                stmt.executeUpdate(sql);
                this.lastInsertedArticleID++;
                //frage eingefuegte ID ab, da sie als Relation fuer die Absatze gebraucht wird
                //deaktiviert aus Performancegruenden
                /*sql = "SELECT `ID` FROM `articles` WHERE 1 ORDER BY `ID` DESC LIMIT 1;";
```

```
                ResultSet lastArticle = stmt.executeQuery(sql);
                while (lastArticle.next()) {
                    lastInsertedArticleID = lastArticle.getInt("ID");
                }*/

            } catch(SQLException e) {
                System.err.println("Fehler beim mysql insert: "+e.getMessage());
            }

            //artikel in mongodb einfuegen

            doc.put("sections",paragraphs);
            doc.put("links", links);
            article.insert(doc);

            //Speicher freigeben
            article = null;
            mysqlConnection = null;

            System.out.println("'"+title+"' ... ok\n");
        } catch (Exception e) {
          System.err.println("Fehler beim mongodb insert von '"+title+"': "+e.getMessage());
        }
    }

    //neu hinzugefügt, genommen aus SQLWriter.java
    protected static String sqlEscape(String str) {
                    if (str.length() == 0)
                            return "''"; //TODO "NULL",too ?
                    final int len = str.length();
                    StringBuffer sql = new StringBuffer(len * 2);
                    synchronized (sql) { //only for StringBuffer
                    sql.append('\'');
                    for (int i = 0; i < len; i++) {
                            char c = str.charAt(i);
                            switch (c) {
                            case '\u0000':
                                    sql.append('\\').append('0');
                                    break;
                            case '\n':
                                    sql.append('\\').append('n');
                                    break;
                            case '\r':
                                    sql.append('\\').append('r');
                                    break;
                            case '\u001a':
                                    sql.append('\\').append('Z');
                                    break;
                            case '"':
                            case '\'':
                            case '\\':
                                    sql.append('\\');
                                    // fall through
                            default:
                                    sql.append(c);
                                    break;
                            }
                    }
                    sql.append('\'');
                    return sql.toString();
                    }
```

```
		}

		void readTimestamp() {
			rev.Timestamp = parseUTCTimestamp(bufferContents());
		}

		void readComment() {
			rev.Comment = bufferContentsOrNull();
			if (rev.Comment==null && !deleted) rev.Comment = ""; //NOTE: null means deleted/
supressed
		}

		void readMinor() {
			rev.Minor = true;
		}

		void readText() {
			rev.Text = bufferContentsOrNull();
			if (rev.Text==null && !deleted) rev.Text = ""; //NOTE: null means deleted/
supressed
		}

		// -----------
		void openContributor() {
			//XXX: record deleted flag?! as it is, any empty <contributor> tag counts as
"deleted"
			contrib =  new Contributor();
		}

		void closeContributor() {
			//NOTE: if the contributor was supressed, nither username nor id have been set in
the Contributor object
			rev.Contributor = contrib;
			contrib = null;
		}


		void readUsername() {
			contrib.Username = bufferContentsOrNull();
		}

		void readIp() {
			contrib.Username = bufferContents();
			contrib.isIP = true;
		}

		private static final TimeZone utc = TimeZone.getTimeZone("UTC");
		private static Calendar parseUTCTimestamp(String text) {
			// 2003-10-26T04:50:47Z
			// We're doing this manually for now, though DateFormatter might work...
			String trimmed = text.trim();
			GregorianCalendar ts = new GregorianCalendar(utc);
			ts.set(
				Integer.parseInt(trimmed.substring(0,0+4)),     // year
				Integer.parseInt(trimmed.substring(5,5+2)) - 1, // month is 0-based!
				Integer.parseInt(trimmed.substring(8,8+2)),     // day
				Integer.parseInt(trimmed.substring(11,11+2)),   // hour
				Integer.parseInt(trimmed.substring(14,14+2)),   // minute
				Integer.parseInt(trimmed.substring(17,17+2)));  // second
			return ts;
		}
```

```
}
```

A.C. Programme für den Benchmarktest

benchmark.rb

```
#!/usr/bin/env ruby
begin
  articles = "Wetter, Gravitation ... "
  articles = articles.split(/,\s*/)
  regularExpression = false;
  searchedField = "t";
  limit = 10
  print "\nTitel;Parameter;Limit;MySQL;NoSQL;"
  articles.each{ |title|
    title = title.gsub(" ","_")
    parameter = " --exact ";

    print "\n#{title};#{parameter};#{limit};"
    print `java -jar ~/NetBeansProjects/DatabasePerformance/dist/DatabasePerformance.jar -l#{limit}
-#{searchedField}#{title} --mongodb --mysql #{parameter} --silent --formatCSV`
  }
end
```

databaseperformance/Main.java

```
package databaseperformance;

import java.io.*;
import java.util.Arrays;
import java.util.List;
import java.util.regex.Pattern;
import java.util.ListIterator;

/**
 *
 * @author Philipp Ständer
 */
public class Main {


    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        //werte die argumente der konsole aus
        List<String> arguments = Arrays.asList(args);
        if (!(arguments.contains("--silent"))) {
            System.out.println("****** Wikipedia MySQL/MongoDB Benchmarktest *****");
```

```
            System.out.println("(c) 2010-2011 philipp staender");
            System.out.println("-h            Hilfe (fuer alle Parameter auflisten)\n");
        }
        if ( (arguments.contains("-h")) ) {
            System.out.println("Startparameter:\n===============");
            System.out.println("-v            Ausfuehrliches Logging, Performance beachten\n");
            System.out.println("--mongodb     Durchsuche die MongoDB");
            System.out.println("--mysql       Durchsuche die MySQL DB");
            System.out.println("--regex       Durchsuche mit reg. Ausdruecken / LIKE %% Statemnts");
            System.out.println("--exact       Durchsuche nach exakten Treffern");
            System.out.println("-l100         Limit setzen, hier z.B. 100");
            System.out.println("-tCoca_Cola   Nach Ueberschrift suchen, hier z.B. Coca Cola");
            System.out.println("-sGeschichte  Nach Unterueberschrift suchen, hier z.B. Geschichte");
            System.out.println("-cTo_be_or    Inhalt durchsuchen");
            System.out.println("--silent      Keine Erklaerungen");
            System.out.println("--formatCSV   Ausgabeformat (Optionen bis jetz: CSV)");
            System.out.println("");
            System.exit(0);
        }
        try {

            ListIterator<String> li = arguments.listIterator();

            int limit = 0;
            String databaseType = "";
            String useRegularExpression = "";
            String searchInSubtitles = "";
            String searchtext = "";
            String outputFormat = "";
            boolean searchInContent = false;

            BufferedReader lineOfText = new BufferedReader(new InputStreamReader(System.in));

            while(li.hasNext()) {
                int i = li.nextIndex();
                String next = li.next();
                if(Pattern.matches("^-l.+", next)) {
                  limit = Integer.parseInt(next.substring(2));
                }
                if(Pattern.matches("^-t.+", next)) {
                  searchtext = next.substring(2).replace("_", " ");
                  searchInSubtitles = "h";
                }
                if(Pattern.matches("^-s.+", next)) {
                  searchtext = next.substring(2).replace("_", " ");
                  searchInSubtitles = "u";
                }
                if(Pattern.matches("^-c.+", next)) {
                  searchtext = next.substring(2).replace("_", " ");
                  searchInContent = true;
                }
                if(Pattern.matches("^--mongodb$", next)) {
                  databaseType = (databaseType.equals("s")) ? "a" : "n";
                }
                if(Pattern.matches("^--mysql$", next)) {
                  databaseType = (databaseType.equals("n")) ? "a" : "s";
                }
                if(Pattern.matches("^--regex$", next)) {
                  useRegularExpression = "j";
                }
                if(Pattern.matches("^--exact$", next)) {
```

```
                    useRegularExpression = "n";
                }
                if(Pattern.matches("^--format.+", next)) {
                    outputFormat = next.substring(8).toLowerCase();
                }
            }

            while (!(searchtext.trim().length()>0)) {
                System.out.print("Artikel (z.B. Alfred Hitchcock): ");
                lineOfText = new BufferedReader(new InputStreamReader(System.in,"UTF-8"));
                searchtext = lineOfText.readLine();
                if (searchtext.trim().length()==0) searchtext = "Alfred Hitchcock";
                else System.out.println("Gewaehlter Artikel: '"+searchtext+"'");
            }

            while (!(limit>=1)) {
                System.out.print("Max. Abfragen [100]: ");
                lineOfText = new BufferedReader(new InputStreamReader(System.in,"UTF-8"));
                String readedLine = lineOfText.readLine().trim();
                if (readedLine.equals("")) limit = 100;
                else limit = Integer.parseInt(readedLine);
                System.out.println("Gewaehltes Limit: "+limit+"");
            }

            while (!((databaseType.equals("s")) || (databaseType.equals("n")) ||
(databaseType.equals("a")) )) {
                System.out.print("Welche Datenbanktypen [s]ql, [n]osql, [a]lle: [a] ");
                lineOfText = new BufferedReader(new InputStreamReader(System.in,"UTF-8"));
                databaseType = lineOfText.readLine();
                if (databaseType.trim().equals("")) databaseType="a";
                System.out.println("Gewaehlte Option: ["+databaseType+"]");
            }

            while (!((useRegularExpression.equals("j")) || (useRegularExpression.equals("n")))) {
                System.out.print("Suche mit regulaeren Ausdruecken / `foo LIKE '%bar%'`-Statements
[j]/[n] : [n] ");
                lineOfText = new BufferedReader(new InputStreamReader(System.in,"UTF-8"));
                useRegularExpression = lineOfText.readLine();
                if ((useRegularExpression.trim().equals(""))) useRegularExpression="n";
                System.out.println("Gewaehlte Option: ["+useRegularExpression+"]");
            }

            while (!((searchInSubtitles.equals("h")) || (searchInSubtitles.equals("u")) ||
(searchInContent))) {
                System.out.print("Duchsuche [h]aupt-,[u]nterueberschriften oder [i]nhalt: [h] ");
                lineOfText = new BufferedReader(new InputStreamReader(System.in,"UTF-8"));
                searchInSubtitles = lineOfText.readLine();
                if ((searchInSubtitles.trim().equals(""))) searchInSubtitles="h";
                if (searchInSubtitles.equals("i")) searchInContent=true;
                System.out.println("Gewaehlte Option: ["+searchInSubtitles+"]");
            }


            //Erstelle Benchmarktest mit den angeforderten Parametern
            WikipediaBenchmark bench = new WikipediaBenchmark();
            bench.startTimer();
            //zugangsdaten fuer mysql, fuer mongodb wird keine benutzer-authent. erwartet
            bench.mysqlUsername="root";
            bench.mysqlPassword="root";
            //setze die tabellen/collection und datenbank namen fest
            bench.mysqlTable="articles";
```

```
                bench.sqlDatabase="nosql";
                bench.nosqlCollection="articles";
                bench.nosqlDatabase="wikipedia";
                //logging level festlegen
                bench.logNoSQL=(arguments.contains("-v"));
                bench.logSQL=(arguments.contains("-v"));
                bench.logVerbose=(arguments.contains("-v"));
                //suchgenauigkeit
                bench.useRegularExpressions=(useRegularExpression.equals("j"));
                //welches feld soll durchsucht werden
                bench.searchInSubtitles=(searchInSubtitles.equals("u"));
                bench.searchInContent=searchInContent;
                bench.outputFormat=outputFormat;

                //wenn inhalt dursucht wird, werden auto. reguläre ausdrücke verwendet
                if (bench.searchInContent) bench.useRegularExpressions=true;
                //silentmodus ja/nein
                if (!(arguments.contains("--silent"))) {
                    System.out.println("Suchbegriff: " + searchtext);
                    System.out.println("Maximal Abfragen: " + limit);
                    bench.log = true;
                } else {
                    bench.log = false;
                    bench.logVerbose = false;
                }

                //durchsuche mysql
                if ((databaseType.equals("s")) || (databaseType.equals("a"))) bench.searchArticleInMySQL
    (searchtext,limit);
                //durchsuche mongodb
                if ((databaseType.equals("n")) || (databaseType.equals("a")))
    bench.searchArticleInMongoDB(searchtext,limit);

                if (!(arguments.contains("--silent"))) {
                    System.out.println("fertig :)");
                }

            } catch (Exception e) {
                e.printStackTrace();
            }
        }

    }
```

databaseperformance/WikipediaBenchmark.java

```
    package databaseperformance;

    import com.mysql.jdbc.Connection;
    import com.mysql.jdbc.Statement;
    import java.sql.DriverManager;
    import java.sql.ResultSet;
    import java.sql.SQLException;
    import java.util.*;

    import java.util.regex.Pattern;
```

```
import com.mongodb.Mongo;
import com.mongodb.DB;
import com.mongodb.DBCollection;
import com.mongodb.BasicDBObject;
import com.mongodb.DBObject;
import com.mongodb.DBCursor;

/**
 *
 * @author philipp staender
 */
public class WikipediaBenchmark {

    public String type = "";
    public String sqlDatabase = "";
    public String nosqlDatabase = "";
    public String nosqlCollection = "";
    public String mysqlTable = "";
    public String mysqlUsername = "root";
    public String mysqlPassword = "root";
    public String mysqlHost = "localhost";
    public int mysqlPort = 3306;
    public boolean searchInSubtitles = false;
    public boolean searchInContent = false;
    public boolean useRegularExpressions = false;
    public boolean logSQL = false;
    public boolean logVerbose = false;
    public boolean logNoSQL = false;
    public boolean log = true;
    public String outputFormat = "";

    public int limit = 100;
    private int articleCount = 0;

    private long startTime = 0;
    private long stopTime = 0;
    private ArrayList<Number> timeStack;
    private ArrayList<String> checkedArticle;
    private ArrayList<String> finalLogs;

    /**
     * Konstruktor
     * Öffnet die mySQL Treiber und initalisiert Arraylisten
     */
    public WikipediaBenchmark() {
        try {
            Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        } catch (Exception e) {
            System.err.println("Fehler beim laden der MySQL Treiber: "+e.getMessage());
            System.exit(-1);
        }
        //initialisiere array
        this.checkedArticle = new ArrayList<String>();
        this.finalLogs = new ArrayList<String>();
    }

    /**
     * Gibt zurück, ob das Limit erreicht ist
     * @return true if more articles found then set
     */
```

```
    public boolean limitReached() {
        return (this.articleCount>=this.limit);
    }

    /**
     * Methode durchsucht die NoSQL Datenbank nach dem Suchbegriff
     * Sie wird rekursiv aufgerufen, bis das Limit erreicht ist
     * @param title Suchbegriff
     * @param i Zähler für das Limit
     * @param db Datenbankinstanz
     * @return -1 Wenn Limit erreicht ist
     */
    private int findArticleInNoSQL(String title, int i, DB db) {
        if (this.limitReached()) return -1;
        i++;
        DBCollection articles = db.getCollection(this.nosqlCollection);
        String searchField = "title";
        if (this.searchInSubtitles) searchField = "sections.subtitle";
        if (this.searchInContent) searchField = "section.content";
        BasicDBObject query = new BasicDBObject(
                searchField,
                ((this.useRegularExpressions) ? Pattern.compile(title, Pattern.CASE_INSENSITIVE) :
title)
                );


        if (this.useRegularExpressions) this.logNoSQL("db.articles.find({\""+searchField
+"\":/"+title+"/});");
        else this.logNoSQL("db.articles.findOne({\""+searchField+"\":'"+title+"'});");

        DBObject article = articles.findOne(query);

        if (article != null) this.articleCount++;
        else return -1;

        String nextArticleTitle = "";

        List<Object> links = (List<Object>) article.get("links");
        this.addLog(links.size()+" Verlinkungen im Artikel '"+title+"' gefunden");
        for (Object link : links) {
            if (this.checkedArticle.contains((String)link)) {
                //Artikel wurde bereits aufgerufen
            } else {
                //Artikel wurde noch nicht aufgerufen, also weiter
                nextArticleTitle = link.toString();
                this.checkedArticle.add(nextArticleTitle);
                //abbruch, falls limit erreicht ist
                if (i>this.limit) return -1;
                else {
                    return this.findArticleInNoSQL(nextArticleTitle, i, db);
                }
            }
        }
        return -1;
    }

    /**
     * Methode durchsucht die SQL Datenbank nach dem Suchbegriff
     * Sie wird rekursiv aufgerufen, bis das Limit erreicht ist
     * @param title Suchbegriff
     * @param i Zähler für das Limit
```

```
         * @param conn Datenbankinstanz
         * @return -1 Wenn Limit erreicht ist
         * @throws SQLException
         */

        private int findArticleInSQL(String title, int i, Connection conn) throws SQLException {
            if (this.limitReached()) return -1;
            i++;
            Statement stmt = (Statement) conn.createStatement();
            String whereCondition = "Title";
            String sqlTitle = title;
            if (this.searchInSubtitles) {
                whereCondition = "Content";
                if (this.useRegularExpressions) title="%"+title+"%";
                sqlTitle = "== "+title+" ==";
            }
            if (this.searchInContent) {
                whereCondition = "Content";
            }
            if ((this.useRegularExpressions) || (this.searchInSubtitles)) whereCondition += " LIKE
    \"%**title**%\"";
            else whereCondition += " = \"**title**\"";
            whereCondition = whereCondition.replace("**title**",WikipediaBenchmark.sqlEscape(sqlTitle));
            String sqlCommand = "SELECT * FROM "+this.mysqlTable+" "+
                            "WHERE "+whereCondition+" LIMIT 1;\n";
            this.logSQL(sqlCommand);
            ResultSet articles,texts;
            articles = stmt.executeQuery(sqlCommand);
            if (articles.next()) {
                String articleID = articles.getString("ID");
                this.articleCount++;
                //Links suchen
                this.logSQL(sqlCommand);
                String[] links = articles.getString("Links").split(",");
                this.addLog(links.length+" Verlinkungen im Artikel '"+title+"' gefunden");

                int linksCount=0;
                String nextArticleTitle;
                int found;
                for (String link : links) {

                    if (this.checkedArticle.contains(link)) {
                        //Artikel wurde bereits aufgerufen
                    } else {
                        //Artikel wurde noch nicht aufgerufen, also weiter
                        nextArticleTitle = link;
                        this.checkedArticle.add(nextArticleTitle);
                        //abbruch, falls limit erreicht ist
                        if (i>this.limit) return -1;
                        else {
                            return this.findArticleInSQL(nextArticleTitle, i, conn);
                        }
                    }
                    linksCount++;
                }
            } else {
                return 0;
            }
            return -1;
        }
```

```
        /**
         * Hauptmethode, welche den Geschwindigkeitstest in der MySQL startet und die Zeit misst
         * @param text Suchbegriff
         * @param limit Maximales Limit
         */
        public void searchArticleInMySQL(String text, int limit) {
            this.startTimer();
            this.limit=limit;
            try {
                //mysql
                this.type="mysql";
                this.addLog("Oeffne MySQL Verbindung");
                Connection conn = this.openConnection();
                this.checkedArticle = new ArrayList<String>();
                this.articleCount = 0;
                //Artikel suchen
                this.articleCount = 0;
                this.addLog("Start mysql Bechnmark...");
                this.resetTimer();
                while (!this.limitReached()) {
                    this.findArticleInSQL(text, this.articleCount, conn);
                }
                if (this.outputIsCSV()) {
                    this.addFinalLog(this.logTimer());
                } else {
                    this.addFinalLog(this.logTimer()+"[s]  \tMySQL");
                }
                this.addLog("Es wurden insg. "+this.articleCount+" Artikel via MySQL rausgesucht");
            } catch (Exception e) {
                System.err.println("Fehler bei der Abfrage der MySQL-DB: "+e.getMessage());
                System.exit(-1);
            }
        }

        /**
         * Hauptmethode, welche den Geschwindigkeitstest in der MongoDB startet und die Zeit misst
         * @param text Suchbegriff
         * @param limit Maximales Limit
         */
        public void searchArticleInMongoDB(String text, int limit) {
            this.startTimer();
            this.limit=limit;
            try {
                //nosql
                this.type="mongodb";
                this.addLog("Oeffne mongodb Verbindung");
                Mongo m = new Mongo();
                DB db = m.getDB( this.nosqlDatabase );
                this.checkedArticle = new ArrayList<String>();
                this.articleCount = 0;
                this.addLog("Start mongodb Bechnmark...");
                this.resetTimer();
                while (!this.limitReached()) {
                    this.findArticleInNoSQL(text, this.articleCount, db);
                }
                if (this.outputIsCSV()) {
                    this.addFinalLog(this.logTimer());
                } else {
                    this.addFinalLog(this.logTimer()+"[s]  \tMongoDB");
                }
                this.addLog("Es wurden insg. "+this.articleCount+" Artikel via mongodb rausgesucht");
```

```
            if (this.outputFormat.equals("csv")) {
                for (Object line : this.finalLogs) {
                    System.out.print(line+";");
                }
            } else {
                System.out.println("========");
                System.out.println(this.getFinalLogMessage());
                System.out.println("========");
            }


        } catch (Exception e) {
            System.err.println("Fehler bei der Abfrage der MongoDB: "+e.getMessage());
            System.exit(-1);
        }
    }

    /**
     * Fügt einen Text zum Log hinzu
     * @param message
     * @return Hinzugefügte Nachricht
     */
    private String addLog(String message) {
        if (this.log) System.out.println(message);
        return message;
    }

    /**
     * Fügt einen optionalen Text zum Log hinzu
     * @param message
     * @return Hinzugefügte Nachricht
     */
    private String addOptionalLog(String message) {
        if (this.logVerbose) System.out.println(message);
        return message;
    }

    /**
     * Fügt einen abschließenden Text zum Log hinzu
     * Meist die Ergebnisse der Geschwindigkeitsmessungen
     * @param message
     * @return Hinzugefügte Nachricht
     */
    public String addFinalLog(String message) {
        this.finalLogs.add(message);
        return message;
    }

    /**
     * Gibt das abschließende Log "formatiert" zurück
     * @return Abschließenede Nachricht
     */
    public String getFinalLogMessage() {
        String message = "";
        for (Object line : this.finalLogs) {
            message += line+"\n";
        }
        return message.trim();
    }

    /**
```

```
     * Ist momentane Datenbank vom Typ SQL?
     * @return true wenn ja, false wenn nicht
     */
    public Boolean isSQL() {
        return (this.type.toLowerCase()=="mysql") ? true : false;
    }

    /**
     * Ist momentane Datenbank nicht vom Typ SQL?
     * @return true wenn nicht, false wenn ja
     */
    public Boolean isNoSQL() {
        return !(this.isSQL());
    }

    /**
     * Ist als Output CSV gewählt?
     * @return true wenn ja, false wenn nicht
     */
    public Boolean outputIsCSV() {
        return (this.outputFormat.toLowerCase().equals("csv"));
    }

    /**
     * Starte den Zeitmessvorgange
     */
    public void startTimer() {
        this.startTime=System.currentTimeMillis();
    }

    /**
     * Starte den Zeitmessvorgang erneut / setze in zurück
     */
    public void resetTimer() {
        this.startTimer();
    }

    /**
     * Gibt die gemessene Zeit vom Start bis jetzt zurück
     * @return
     */
    public long measureTimer() {
        this.stopTime=System.currentTimeMillis();
        long difference = this.stopTime-this.startTime;
        difference=Math.round(difference);
        return difference;
    }

    /**
     * Gib die gemessene Zeit als Text zurück
     * @return gemessene Zeit
     */
    public String logTimer() {
        return logTimer("");
    }

    /**
     * Gib die gemessene Zeit als Text mit Prefix zurück
     * @param message
     * @return gemessene Zeit+Prefix
     */
```

```
    public String logTimer(String message) {
        double time = this.measureTimer();
        time = Math.round(time);
        time = time/1000;
        message = time+message;
        this.addLog(message);
        return message;
    }

    /**
     * Logge SQL Befehl
     * @param sql
     */
    public void logSQL(String sql) {
        if (this.logSQL) this.addLog("mysql> "+sql);
    }

    /**
     * Logge NoSQL Befehl
     * @param command
     */
    public void logNoSQL(String command) {
        if (this.logNoSQL) this.addLog("mongodb> "+command);
    }
    /**
     * Öffne MySQL Verbindung
     * @return MySQL-Verbindung
     */
    private Connection openConnection() {
        if (this.isSQL()) {
            String dbURL = "jdbc:mysql://"+this.mysqlHost+":"+this.mysqlPort+"/"+this.sqlDatabase;
            try {
                Connection conn = (Connection) DriverManager.getConnection
(dbURL,this.mysqlUsername,this.mysqlPassword);
                return conn;
                } catch (SQLException ex) {
                // handle any errors
                System.out.println("SQLException: " + ex.getMessage());
                System.out.println("SQLState: " + ex.getSQLState());
                System.out.println("VendorError: " + ex.getErrorCode());
                }
        }
        return null;
    }

    /**
     * Konvertiere String zu einem SQL-sicheren String
     * @param str
     * @return SQL-sicherer String
     */
    public static String sqlEscape(String str) {
                if (str.length() == 0)
                        return ""; //TODO "NULL",too ?
                final int len = str.length();
                StringBuffer sql = new StringBuffer(len * 2);
                synchronized (sql) { //only for StringBuffer
                for (int i = 0; i < len; i++) {
                        char c = str.charAt(i);
                        switch (c) {
                        case '\u0000':
                                sql.append('\\').append('0');
```

```
                                        break;
                                case '\n':
                                        sql.append('\\').append('n');
                                        break;
                                case '\r':
                                        sql.append('\\').append('r');
                                        break;
                                case '\u001a':
                                        sql.append('\\').append('Z');
                                        break;
                                case '"':
                                case '\'':
                                case '\\':
                                        sql.append('\\');
                                        // fall through
                                default:
                                        sql.append(c);
                                        break;
                                }
                        }
                        return sql.toString();
                        }
                }
        }
```

B.Quellenverzeichnis

B.A. Gedruckte Quellen

Prof. Dr. phil. Gregor Büchel
Datenbanken und Algorithmen
Vorlesungsskript, Fachhochschule Köln, 2005/2006

Kristina Chodorow & Michael Dirolf
MongoDB: The Definitive Guide
2010, O'Reilly Media, Sebastopol, 1. Auflage

Stefan Edlich, Achim Friedland, Jens Hampe, Benjamin Brauer
NoSQL
Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken
2010, Hanser Fachbuchverlag, München, 1. Auflage

Ramez Elmasri, Shamkant B. Navathe
Grundlagen von Datenbanksystemen
2005, Pearson Studium, Auflage: 3., überarb. Auflage

Mark Matthews Jim Cole Joseph D. Gradecki
MySQL and Java Developer's Guide
2003, Wiley Publishing, Indianapolis

B.B. Onlinequellen

Data Definition Language *
http://de.wikipedia.org/wiki/Data_Definition_Language

ACID vs. BASE — NoSQL erklärt
Hinnerk Haardt
http://www.scribd.com/doc/30637338/ACID-vs-BASE-—-NoSQL-erklart

Towards Robust Distributed Systems
Dr. Eric A. Brewer
http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf

Geschichte von MySQL
http://dev.mysql.com/doc/refman/5.1/de/history.html
Copyright © 1997, 2010, Oracle and/or its affiliates

MySQL *
http://de.wikipedia.org/wiki/MySQL

InnoDB *
http://de.wikipedia.org/wiki/InnoDB

Changes in Release 4.0.x
http://dev.mysql.com/doc/refman/4.1/en/news-4-0-x.html
Copyright © 1997, 2011, Oracle and/or its affiliates

Changes in Release 4.1.x
http://dev.mysql.com/doc/refman/4.1/en/news-4-1-x.html
Copyright © 1997, 2011, Oracle and/or its affiliates

Oracle Completes Acquisition of Sun
Karen Tillman
http://www.oracle.com/us/corporate/press/044428

YouTube, Flickr, and Wikipedia to Share their Secrets of Success at the 2007 MySQL Conference & Expo
© 2010, Oracle Corporation and/or its affiliates
http://dev.mysql.com/tech-resources/articles/mysqluc-2007.html

MySQL Kunden
© 2010, Oracle Corporation and/or its affiliates
http://www.mysql.de/customers/

MyISAM *
http://de.wikipedia.org/wiki/MyISAM

Die MyISAM-Speicher-Engine
Copyright © 1997, 2010, Oracle and/or its affiliates.
http://dev.mysql.com/doc/refman/5.1/de/myisam-storage-engine.html

Entity-Relationship-Modell *
http://de.wikipedia.org/wiki/Entity-Relationship-Modell

SQL-Anweisungssyntax
Copyright © 1997, 2010, Oracle and/or its affiliates.
http://dev.mysql.com/doc/refman/5.1/de/sql-syntax.html

Data Manipulation Language, SQL *
http://de.wikipedia.org/wiki/Data_Manipulation_Language

Regular Expressions
Copyright © 1997, 2010, Oracle and/or its affiliates.
http://dev.mysql.com/doc/refman/5.5/en/regexp.html

Sun kauft MySQL AB für eine Milliarde US-Dollar
Heise Verlag, odi@heise.de
http://www.heise.de/newsticker/meldung/Sun-kauft-MySQL-AB-fuer-eine-Milliarde-US-Dollar-Update-179176.html

MySQL-Lizenzpolitik
© 2010, Oracle Corporation and/or its affiliates
http://www.mysql.de/about/legal/licensing/index.html

neckermann.de erreicht kostengünstige Datenbank-Skalierung mit MySQL Cluster
© 2010, Oracle Corporation and/or its affiliates
http://www.mysql.de/why-mysql/case-studies/de/neckermann.php

Skalierbarkeit - Welches der nachfolgenden Systeme skaliert gut?
Hans-Peter Oser
http://www.oser.org/~hp/bsyII/node6.html

MySQL Scale-Out by application partitioning
Oli Sennhauser
http://dev.mysql.com/tech-resources/articles/application_partitioning_wp.pdf

Zeilenbasierte Replikation
Copyright © 1997, 2010, Oracle and/or its affiliates.
http://dev.mysql.com/doc/refman/5.1/de/replication-row-based.html

BSON - Binary JSON
creative commons
http://bsonspec.org/

JSON *
http://en.wikipedia.org/wiki/JSON

Inserting
Kristina Chodorow, Dwight Merriman
http://www.mongodb.org/display/DOCS/Inserting

Advanced Queries
Rian Murphy, Scott Hernandez
http://www.mongodb.org/display/DOCS/Advanced+Queries

Updating
Kristina Chodorow, Dan Pasette
http://www.mongodb.org/display/DOCS/Updating

Removing
Kristina Chodorow, Dwight Merriman
http://www.mongodb.org/display/DOCS/Removing

Advanced Queries
Rian Murphy, Scott Hernandez
http://www.mongodb.org/display/DOCS/Advanced+Queries#AdvancedQueries-%24type

Java Tutorial
Rian Murphy, Scott Hernandez
http://www.mongodb.org/display/DOCS/Java+Tutorial

MapReduce: Simplified Data Processing on Large Clusters
Jeffrey Dean and Sanjay Ghemawat
http://labs.google.com/papers/mapreduce.html

mongo - The Interactive Shell
Rian Murphy, Scott Hernandez
http://www.mongodb.org/display/DOCS/mongo+-+The+Interactive+Shell

Server-side Code Execution
Rian Murphy, Eliot Horowitz
http://www.mongodb.org/display/DOCS/Server-side+Code+Execution

SQL to MongoDB
Rick Osborne
http://rickosborne.org/download/SQL-to-MongoDB.pdf

Five reasons to upgrade to MySQL 5.5
Ronald Bradford
http://ronaldbradford.com/blog/five-reasons-to-upgrade-to-mysql-5-5-2010-12-15/

Shard (database architecture) *
http://en.wikipedia.org/wiki/Shard_(database_architecture)

Objektrelationale Abbildung *
http://de.wikipedia.org/wiki/Objektrelationale_Abbildung

About Wikipedia *
http://en.wikipedia.org/wiki/Wikipedia:About

Größenvergleich der Wikipedia *
http://de.wikipedia.org/wiki/Wikipedia:Gr%C3%B6%C3%9Fenvergleich

Modelling Wikipedia's growth *
http://en.wikipedia.org/wiki/Wikipedia:Modelling_Wikipedia's_growth

CouchDB In The Wild - A list of organizations using CouchDB to create software and websites in the wild
Stephen Ausman
http://wiki.apache.org/couchdb/CouchDB_in_the_wild

GlassFish Server
http://glassfish.java.net/

Datenbankanwendung für Windows- sowie Mac OS-Betriebssysteme und das Web
Copyright © 1994-2011, FileMaker, Inc.
http://www.filemaker.de/

Download der Wikipedia*
http://de.wikipedia.org/wiki/Wikipedia:Download#Herunterladen_aller_Seiten_als_XML-Dump

The Apache CouchDB Project
Copyright © 2008–2011 The Apache Software Foundation
http://couchdb.apache.org/

Many to many update in MongoDB without transactions
Gareth Elms
http://stackoverflow.com/questions/4992648/many-to-many-update-in-mongodb-without-transactions

* Dieser Text stammt aus der Wikipedia. Diese Quelle hat keinen eindeutigen Autor und ist unter der „Creative Commons Attribution/Share Alike“-Lizenz verfügbar

C. Eidesstattliche Erklärung

Ich versichere hiermit, die vorgelegte Arbeit in dem gemeldeten Zeitraum ohne fremde Hilfe verfasst und mich keiner anderen als der angegebenen Hilfsmittel und Quellen bedient zu haben.

Berlin, den 14.April 2011

Philipp Ständer