Big Data in Datenbanken

Realisierung des Performancemonitorings mit MySQL und MongoDB

P r a x i s a r b e i t

an der Hochschule für Angewandte Wissenschaften Hof
Fakultät Informatik
Studiengang Wirtschaftsinformatik

Vorgelegt bei
Prof. Dr. Jürgen Heym

Vorgelegt von
Andrea Reisenauer

Hof, 14.03.2019

Vorwort

Die vorliegende Praxisarbeit entstand im Rahmen meines Studiums der Wirtschaftsinformatik an der Hochschule für angewandte Wissenschaften in Hof.

Mein besonderer Dank gilt Herrn Prof. Dr. Heym für die Gelegenheit zur Anfertigung dieser Praxisarbeit sowie für seine umfassende Betreuung. Vielen Dank, dass Sie mir stets mit Rat und Tat zur Seite standen.

Darüber hinaus bedanke ich mich bei Herrn Hettmer vom Bayerischen Landesamt für Digitalisierung, Breitband und Vermessung für die zur Verfügung gestellten Geodaten und die interessanten Einblicke in die Vermessungsverwaltung.

Ebenso möchte ich meiner Familie und meinem Freund Max danken, die mich mit viel Geduld und Nachsicht durch die Zeit der Praxisarbeit begleitet haben.

Inhaltsverzeichnis

Darstellungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

LDBV ..... Landesamt für Digitalisierung, Breitband und Vermessung
NRDBMS ..... nicht-relationale Datenbankmanagement Systeme
RDBMS..... relationale Datenbankmanagement Systeme
VPN ..... Virtual Private Network
WKT ..... Well-known text

1 Einleitung

1.1 Motivation

In der Vermessungsverwaltung werden bisher relationale Datenbankmodelle zur Speicherung der Daten verwendet. Da es mittlerweile jedoch deutlich performantere Datenbanken zur Speicherung großer Datenmengen gibt, zieht das Landesamt für Digitalisierung, Breitband und Vermessung (LDBV) einen Technologiewechsel in Erwägung.

Ziel dieser Praxisarbeit ist es das in der Bachelorarbeit beschriebene Konzept des Performancemonitorings von relationalen Datenbankmanagement Systemen und (RDBMS) und nicht relationalen Datenbankmanagement Systemen (NRDBMS) zu realisieren, die Messergebnisse zu interpretieren und anschließend eine Empfehlung für das weitere Vorgehen des LDBV auszusprechen. Als Vertreter für RDBMS wurde MySQL gewählt. Die NRDBMS repräsentiert in dieser Arbeit MongoDB. Die Entscheidungsgrundlage für die Wahl der beiden Datenbankverwaltungssysteme wurde bereits in der Bachelorarbeit dargelegt.

1.2 Fragestellung

Bei der Realisierung des Performancemonitorings soll die Beantwortung folgender Fragen im Mittelpunkt stehen:

- Welche Vorbereitungen sind an den Testservern vorzunehmen um das Performancemonitoring realisieren zu können?
- Wie installiert man MySQL bzw. MongoDB auf einem Ubuntu18.04 System?
- Welche grafischen Oberflächen erleichtern die Arbeit mit MySQL bzw. MongoDB?
- Wie importiert man Daten in eine MySQL-Datenbank bzw. in eine MongoDB und was ist dabei besonders zu beachten?
- Welche Programmiersprache eignet sich für die Programmierung der Software für das Performancemonitoring und wie muss das Programm für das Performancemonitoring gestaltet sein?
- Welche Ergebnisse erzielt das Performancemonitoring und wie sind diese zu interpretieren?

1.3 Aufbau der Arbeit

Die Arbeit gliedert sich in sechs Teile. Zunächst werden im zweiten Kapitel alle Vorbereitungen und Installationen auf den Testservern aufgezeigt, die für die erfolgreiche Durchführung des Perfomancemonitorings notwendig sind. Besonderes Augenmerk wird dabei auf die Installation der MySQL-Datenbank und der MongoDB gelegt. Darüber hinaus wird für jede Datenbank eine kostenfreie grafische Benutzeroberfläche vorgestellt.

Im dritten Kapitel wird der Ablauf beim Import der Geodaten in die MySQL Datenbank mit Hilfe von LOAD DATA beschrieben. Des Weiteren behandelt dieses Kapitel den import von Geodaten in die MongoDB mittels mongoimport sowie die Herausforderungen des Datenimports in die jeweilige Datenbank.

Anschließend wird im vierten Kapitel das Programm zur Messung der Performance beschrieben. Darüber hinaus werden in diesem Kapitel auf die Probleme einer solchen Performanceanalyse sowie die Qualität der entstandenen Messergebnisse eingegangen.

Schließlich werden die Messergebnisse des Performancemonitorings der beiden Datenbanken verglichen und interpretiert sowie eine Empfehlung für das weitere Vorgehen des LDBV ausgesprochen.

Das letzte Kapitel beinhaltet die Zusammenfassung sowie einen Ausblick in die Zukunft.

2 Vorbereitung der Server für das Performancemonitoring

2.1 Aufsetzen eines Ubuntu-Servers

Um die gleichen Rahmenbedingungen für das Performancemonitoring zu schaffen, werden zwei identische Ubuntu 18.04 Server aufgesetzt. Auf dem Server t3rd-ar-3.netlab.hof-university.de wird die Zeitmessung mit MySQL erfolgen, Server t3rd-ar-4.netlab.hof-university.de wird für das Performancemonitoring mit MongoDB verwendet. Die Installation der beiden Datenbanken wird in Kapitel 2.3 und 2.4 näher erläutert.

Der Ubuntu 18.04 Server ist eine virtuelle Maschine auf einem Hochschulserver und wurde dankenswerterweise von Herrn Prof. Dr. Heym aufgesetzt und administriert. Die Grundlagen zum Thema virtuelle Maschinen sind in der Bachelorarbeit in Kapitel 5.1 beschrieben. Folgende Schritte wurden bei der Installation der Testserver mit der Software VMware Remote Console durchgeführt:

- Auswahl der bevorzugten Sprache (English wählen)
- Festlegen des Tastaturlayouts
- Konfigurieren der Netzwerkeinstellungen (siehe Abbildung 1)
- Festlegen der Proxy-Einstellungen (siehe Abbildung 2)
- Dateisystem-Setup („Use An Entire Disk“ auswählen)

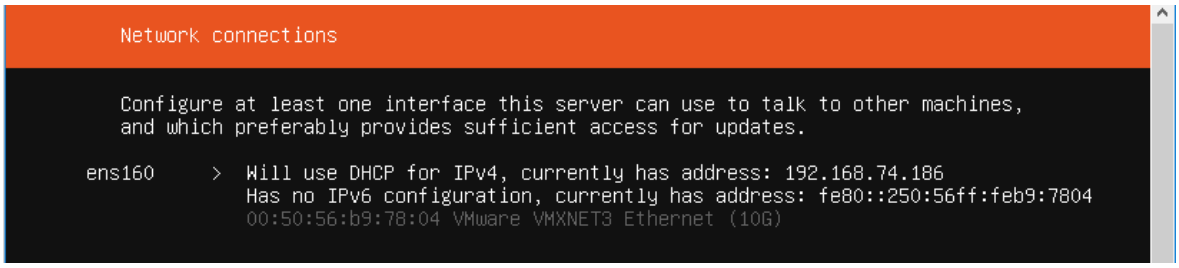


Abbildung 1: Konfiguration der Netzwerkverbindungen

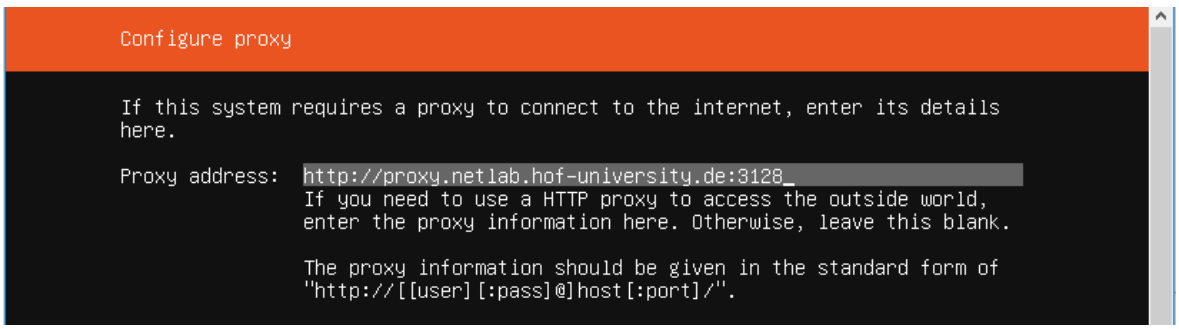


Abbildung 2: Festlegen der Proxy-Einstellungen

2.2 Performancemonitoring mit MySQL

2.2.1 Installation von MySQL

Auf dem Ubuntu 18.04 Host ist bisher keine Datenbank vorinstalliert. Im Rahmen der Bachelorarbeit wurde sich für MySQL als Vertreter der RDBMS entschieden. Um den MySQL-Server zu installieren, wird `apt-get install mysql-server` in der Kommandozeile eingegeben und ausgeführt. Um die Sicherheit zu erhöhen wird im Anschluss `mysql_secure_installation` ausgeführt. Im Rahmen dieser Installation kann man folgende Einstellungen vornehmen:

- Installation des VALIDATE PASSWORD PLUGIN zur Überprüfung der Qualität der Passwörter
- Setzen des root Passworts
- Entfernen des anonymen Benutzers. Dieser ist standardmäßig in der MySQL-Installation enthalten und ermöglicht jedem den Login in die MySQL Datenbank, ohne dass dafür vorab ein Benutzerkonto erstellt werden muss. Dieser User ist nur für Testzwecke vorgesehen, um die Installation reibungsloser zu gestalten. Er sollte vor einem Wechsel in die Produktionsumgebung unbedingt entfernt werden
- Deaktivieren des Remote-Zugriffs für den root-User
- Entfernen der Datenbank „test“ auf die standardmäßig alle Benutzer, auch anonyme Benutzer, zugreifen können. Darüber hinaus werden die Berechtigungen entfernt, die es jedem ermöglichen auf Datenbanken zuzugreifen, deren Datenbankname mit test_ beginnen.

Die einzelnen Schritte werden im Rahmen der sicheren Installation nacheinander durchgeführt. Der User wird nach jedem Schritt gefragt, ob er diesen ausführen möchte. Die Bestätigung der Auswahl erfolgt mit y oder Y die Ablehnung erfolgt mit jeder belieben anderen Taste. Die Installation auf dem Server t3rd-ar-3 wird in Abbildung 3 gezeigt.

```
Securing the MySQL server deployment.

Connecting to MySQL using a blank password.

VALIDATE PASSWORD PLUGIN can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD plugin?

Press y|Y for Yes, any other key for No: y

There are three levels of password validation policy:

LOW    Length >= 8
MEDIUM Length >= 8, numeric, mixed case, and special characters
STRONG Length >= 8, numeric, mixed case, special characters and dictionary
            file

Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 0
Please set the password for root here.

New password:

Re-enter new password:

Estimated strength of the password: 25
Do you wish to continue with the password provided?(Press y|Y for Yes, any other
 key for No) : y
By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have
a user account created for them. This is intended only for
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.

Remove anonymous users? (Press y|Y for Yes, any other key for No) : y
Success.


Normally, root should only be allowed to connect from
'localhost'. This ensures that someone cannot guess at
the root password from the network.

Disallow root login remotely? (Press y|Y for Yes, any other key for No) : y
Success.

By default, MySQL comes with a database named 'test' that
anyone can access. This is also intended only for testing,
and should be removed before moving into a production
environment.


Remove test database and access to it? (Press y|Y for Yes, any other key for No)
 : y
 - Dropping test database...
Success.

 - Removing privileges on test database...
Success.

Reloading the privilege tables will ensure that all changes
made so far will take effect immediately.

Reload privilege tables now? (Press y|Y for Yes, any other key for No) : y
Success.

All done!
```

Abbildung 3: Verbesserung der MySQL-Sicherheit

Im Anschluss an die Installation wird die MySQL-Datenbank einem ersten kleinen Test unterzogen. Hierfür wird ein das Skript createMyDB (siehe Abbildung 4) erstellt, dass die Datenbank „Library“ mit der Tabelle „books“ erstellt und einen Datensatz einfügt. Mit dem Befehl `mysql -p -u root < createMyDB` wird das Skript ausgeführt. Um den korrekten Import der Datensätze zu überprüfen, loggt man sich mit dem Befehl `mysql -p -u root` in der Datenbank ein und wechselt anschließend mit dem Befehl `use Library` in die soeben erstellte Datenbank. Die Daten in der Tabelle books werden mit Hilfe des Befehls `SELECT * FROM books;` aus der Datenbank ausgelesen und angezeigt (siehe Abbildung 5).

```
create database Library;
use Library;
CREATE TABLE books (ISBN VARCHAR(20),
                    title VARCHAR(20),
                    author VARCHAR(20),
                    borrowing_start DATE,
                    borrowing_end DATE);
Insert into books values('978-382663022',
                         'MySQL für Dummies',
                         'Michael Rüttger',
                         '2018-12-24',
                         '2019-01-26');
```

Abbildung 4: Skript zum Testen von MySQL

```
areisenauer@t3rd-ar-3: ~/mysqlskripte/Test
mysql>
mysql> use Library
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> Select * FROM books;
+---------------+--------------------+------------------+-----------------+---------------+
| ISBN          | title              | author           | borrowing_start | borrowing_end |
+---------------+--------------------+------------------+-----------------+---------------+
| 978-382663022 | MySQL für Dummies  | Michael Rüttger  | 2018-12-24      | 2019-01-26    |
+---------------+--------------------+------------------+-----------------+---------------+
1 row in set (0.00 sec)

mysql>
```

Abbildung 5: Anzeigen aller Daten der Tabelle books

Der Zugriff von anderen IP-Adressen als dem Localhost (127.0.0.1) ist in der Standardkonfiguration von MySQL untersagt. Da das Java-Programm für das Performancemonitoring aber auf dem Server der Hochschule mit dem Hostname http://t3rd-ar-3.netlab.hof-university.de und der IP-Adresse 192.168.74.243 ausgeführt wird, muss der Zugriff von diesem Server auf die Datenbank erlaubt werden. Hierfür muss die bind-address in der Datei mysqld.cnf, die im Verzeichnis /etc/mysql/mysql.conf.d abgelegt ist, geändert werden. Dort

ändert man wie in Abbildung 6 dargestellt in der Zeile bind-address den Wert von 127.0.0.1 auf 192.168.74.243. Der Hochschulserver t3rd-ar-3 hat nun Zugriff auf die MySQL Datenbank. Sollte man die IP-Adresse des Servers, der Zugriff auf die MySQL-Datenbank erhalten soll, nicht kennen, kann man diese mit dem Befehl `ip addr` ermitteln. Alternativ könnte man als bind-address auch den Wert 0.0.0.0 eingeben. Dies bedeutet, dass von allen anderen Rechnern auf die MySQL-Datenbank zugegriffen werden kann.

Sämtliche Änderungen in der mysql.conf.d werden erst nach einem Neustart des MySQL-Servers mit dem Befehl `sudo /etc/init.d/mysql restart` wirksam.

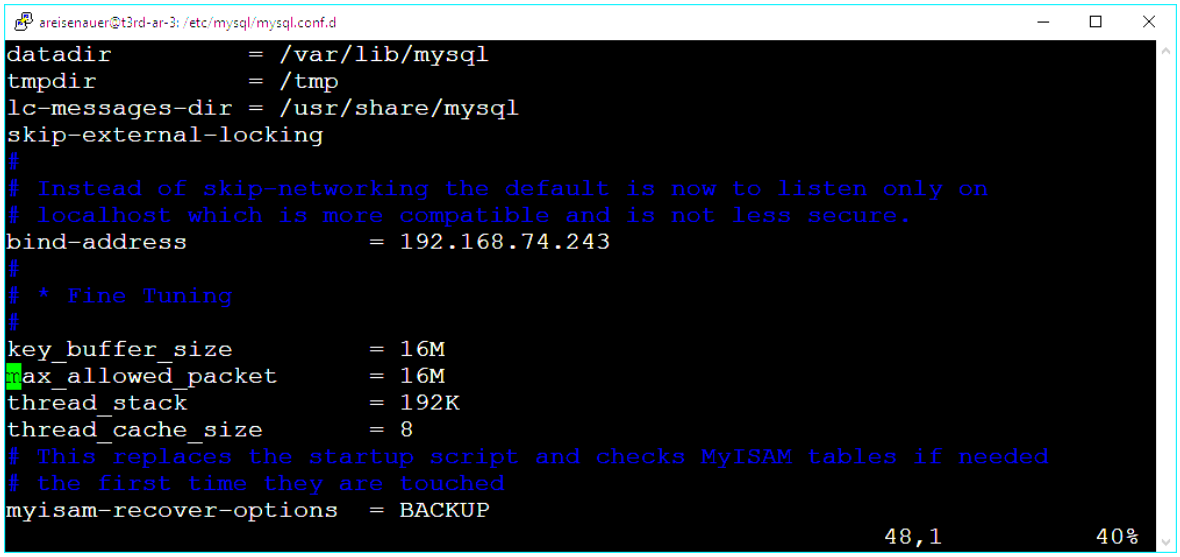


Abbildung 6: Änderung der IP-Adresse des MySQL-Servers

2.2.2 Installation von phpMyAdmin

Im täglichen Gebrauch von MySQL Datenbanken gibt es verschiedene Software-Anwendungen, die die Arbeit erleichtern können. Um nicht dauerhaft mit der Konsole arbeiten zu müssen und die Daten besser verwalten zu können, wird auf dem Server phpMyAdmin installiert. Dies ist eine kostenlose, in der Programmiersprache PHP implementierte Webanwendung zur Administration von MySQL-Datenbanken. PhpMyAdmin stellt viele Funktionen wie beispielsweise das Auflisten von Datensätzen, das Anlegen von Tabellen oder das Verwalten von Benutzern bereit, ohne dass der Anwender hierfür selbst SQL- Anweisungen schreiben muss.

Gründer des phpMyAdmin- Projekts ist der Südtiroler Thomas Ratschiller, der zusammen mit Marc Delisle im Dezember 1998 die erste mehrsprachige Version von phpMyAdmin veröffentlichte (vgl. The phpMyAdmin devel team 2018, S. 149).

Um phpMyAdmin auf dem Ubuntu-Server installieren zu können, muss man zunächst auf der phpMyAdmin Website die Linkadresse des phpMyAdmin-4.8.4-all-languages.tar.gz Pakets kopieren. Anschließend führt man, wie in Abbildung 7 dargestellt, auf dem Terminal den `wget` Befehl in Kombination mit der Linkadresse des phpMyAdmin Pakets aus, um direkt über das Terminal die phpMyAdmin Dateien vom HTTP-Server herunterzuladen.

```
areisenauer@t3rd-ar-1:~$ wget https://files.phpmyadmin.net/phpMyAdmin/4.8.4
/phpMyAdmin-4.8.4-all-languages.tar.gz
--2018-12-30 11:21:10--  https://files.phpmyadmin.net/phpMyAdmin/4.8.4/phpM
yAdmin-4.8.4-all-languages.tar.gz
Resolving proxy.netlab.hof-university.de (proxy.netlab.hof-university.de)..
. 192.168.74.123
Connecting to proxy.netlab.hof-university.de (proxy.netlab.hof-university.d
e)|192.168.74.123|:3128... connected.
Proxy request sent, awaiting response... 200 OK
Length: 9779796 (9.3M) [application/octet-stream]
Saving to: 'phpMyAdmin-4.8.4-all-languages.tar.gz'

phpMyAdmin-4.8.4-a 100%[===============>]   9.33M  8.64MB/s    in 1.1s

2018-12-30 11:21:11 (8.64 MB/s) - 'phpMyAdmin-4.8.4-all-languages.tar.gz' s
aved [9779796/9779796]
```

Abbildung 7: Herunterladen des phpMyAdmin-Pakets

Im Anschluss daran muss das mit tar und gzip doppelt komprimierte Softwarepaket entpackt werden. Da es sich um ein tar-Archiv handelt, verwendet man zum Entpacken den Befehl `tar.` Zusätzlich zu dem `tar` Befehl muss man noch die Optionen `xzf` angeben. Das x steht dabei für das extrahieren der Dateien aus dem Archiv. Die Option z führt dazu, dass das Archiv zusätzlich mit gzip dekomprimiert wird. Die Option f gibt an, aus welcher Datei die Daten gelesen und entpackt werden sollen. Der gesamte Befehl sowie das Ergebnis des entpackten Pakets sind in Abbildung 8 dargestellt.

```
areisenauer@t3rd-ar-1:~$ ls
mysqlskripte  phpMyAdmin-4.8.4-all-languages.tar.gz
areisenauer@t3rd-ar-1:~$ tar xzf phpMyAdmin-4.8.4-all-languages.tar.gz
areisenauer@t3rd-ar-1:~$ ls
mysqlskripte                  phpMyAdmin-4.8.4-all-languages.tar.gz
phpMyAdmin-4.8.4-all-languages
```

Abbildung 8: Entpacken des phpMyAdmin-Pakets

Das Entpackte phyMyAdmin-4.8.4-all-languages Paket wird nun wie in Abbildung 9 dargestellt mit dem `mv` Befehl in Paket in phpMyAdmin umbenannt.

```
areisenauer@t3rd-ar-1:~$ ls
mysqlskripte                    phpMyAdmin-4.8.4-all-languages.tar.gz
phpMyAdmin-4.8.4-all-languages
areisenauer@t3rd-ar-1:~$ mv phpMyAdmin-4.8.4-all-languages phpMyAdmin
areisenauer@t3rd-ar-1:~$ ls
mysqlskripte  phpMyAdmin  phpMyAdmin-4.8.4-all-languages.tar.gz
```

Abbildung 9: Umbenennen des phpMyAdmin-Pakets

Nun müssen noch die Rechte für den phpMyAdmin Ordner angepasst werden. Mit dem Befehl `sudo chown -R www-data:www-data phpMyAdmin` (siehe Abbildung 10) wird der Ordner selbst, sein Inhalt sowie alle Unterordner des Ordners phpMyAdmin dem Besitzer und der Gruppe www-data zugeordnet (Braun Heider GmbH).

```
areisenauer@t3rd-ar-1:~$ sudo chown -R www-data:www-data phpMyAdmin
[sudo] password for areisenauer:
```

Abbildung 10: Ändern der Rechte für phpMyAdmin

Um phpMyAdmin verwenden zu können, muss der Ordner in den html Ordner verschoben werden. Das Verschieben erfolgt mit dem `mv` Befehl, gefolgt von dem Dateinamen der Datei, die verschoben werden soll sowie an dritter Stelle der Zielordner (siehe Abbildung 11).

```
areisenauer@t3rd-ar-1:~$ cd /var/www/html
areisenauer@t3rd-ar-1:/var/www/html$ ls
index.html  info.php
areisenauer@t3rd-ar-1:/var/www/html$ cd /home/areisenauer/
areisenauer@t3rd-ar-1:~$ ls
mysqlskripte  phpMyAdmin  phpMyAdmin-4.8.4-all-languages.tar.gz
areisenauer@t3rd-ar-1:~$ sudo mv phpMyAdmin /var/www/html
areisenauer@t3rd-ar-1:~$ cd /var/www/html
areisenauer@t3rd-ar-1:/var/www/html$ ls
index.html  info.php  phpMyAdmin
```

phpMyAdmin ist noch nicht im html Ordner enthalten

phpMyAdmin wurde erfolgreich verschoben

Abbildung 11: Verschieben des phpMyAdmin Ordners

Das phpMyAdmin-Webinterface soll im Browser aufrufen werden können. Aus diesem Grund muss die Konfiguration des Apache2 Webservers angepasst werden. Hierzu öffnet man im Ordner /etc/apache2 die Konfigurationsdatei apache2.conf mit Hilfe eines Texteditors wie beispielsweise vim. Um Änderungen an der Datei vornehmen zu können, ist es wichtig, dass diese als Super-User geöffnet wird (siehe Abbildung 12). In der geöffneten Datei wird nun an letzter Stelle die Zeile `Include /etc/phpmyadmin/apache.conf` ergänzt (siehe Abbildung 13). Ebenso wie bei der mysql-Konfigurationsdatei werden auch in diesem Fall Änderungen in der Konfigurationsdatei erst wirksam, nachdem der Webserver

neu gestartet wurde. Der Neustart wird, wie in Abbildung 14 zu sehen, mit dem Befehl `/etc/init.d/apache2 restart` veranlasst.

```
areisenauer@t3rd-ar-1:/$ cd etc/apache2
areisenauer@t3rd-ar-1:/etc/apache2$ ls
apache2.conf       envvars         ports.conf
apache2.conf.save  magic           sites-available
conf-available     mods-available  sites-enabled
conf-enabled       mods-enabled
areisenauer@t3rd-ar-1:/etc/apache2$ sudo vim apache2.conf
[sudo] password for areisenauer:
```

Abbildung 12: Öffnen der Apache2 Konfigurationsdatei mit vim

```
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

# Include of directories ignores editors' and dpkg's backup files,
# see README.Debian for details.

# Include generic snippets of statements
IncludeOptional conf-enabled/*.conf

# Include the virtual host configurations:
IncludeOptional sites-enabled/*.conf
```

Abbildung 13: Ergänzen der Konfigurationsdatei apache2.conf

```
areisenauer@t3rd-ar-1:/etc/apache2$ sudo /etc/init.d/apache2 restart
[ ok ] Restarting apache2 (via systemctl): apache2.service.
```

Abbildung 14: Restart des Apache Webservers

Die mbstring-Funktionen von PHP bieten Unterstützung für Sprachen, die durch Multibyte Zeichensätze dargestellt werden (vgl. The phpMyAdmin devel team 2018, S. 170). Multibytezeichensätze verwenden abhängig vom zu speichernden Zeichen unterschiedlich viele Bytes (Wolf 2019). Da es sich bei dem Zeichensatz UTF-8, der in der MySQL Datenbank verwendet werden soll, um einen Multibyte Zeichensatz handelt, muss php-mbstring auf dem Server installiert werden. Hierzu wird wie in Abbildung 15 dargestellt, der Befehl `sudo apt-get install php-mbstring` ausgeführt.

```
areisenauer@t3rd-ar-1:/$ sudo apt-get install php-mbstring
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  php-mbstring
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 2,008 B of archives.
After this operation, 12.3 kB of additional disk space will be used.
```

Abbildung 15: Installation von php-mbstring

Nach erfolgreicher Ausführung der bisher genannten Schritte kann man nun versuchen sich mit Hilfe von phpMyAdmin in der MySQL Datenbank einzuloggen. Hierzu ruft im Internetbrowser die Adresse http://t3rd-ar-3.netlab.hof-university.de/phpMyAdmin/ auf. Für den Login gibt man nun die bei der Installation der MySQL Datenbank festgelegten Daten ein (User: root, Passwort: sql2018). Auf dem Bildschirm erscheint nun wie in Abbildung 16 zu sehen ist, die Fehlermeldung „mysqli_real_connect(): (HY000/1698): Access denied for user 'root'@'localhost'“.

Abbildung 16: Fehlermeldung beim Remote-Login als root User

Aus Sicherheitsgründen haben MySQL Remote-Benutzer nicht die Möglichkeit, sich als root anzumelden. Da Hacker kontinuierlich nach diesen Konten suchen, ist es auch nicht empfohlen die Einstellungen so anzupassen, dass das root-Konto über phpMyAdmin aus der Ferne erreichbar ist (Can’t log into phpMyAdmin: mysqli_real_connect: (HY000/1698): Access denied for user ‘root’@’localhost’ 2018). Der remote Zugriff auf die Datenbank ist für die Praxisarbeit jedoch zwingend erforderlich, da die Datenbank auf dem Server im Netzwerk der Hochschule Hof (Hostname t3rd-ar-3.netlab.hof-university.de) installiert ist, die

arbeiten an der Datenbank jedoch aus dem privaten Netzwerk erfolgen. Um einen neuen Benutzer anlegen zu können, loggt man sich zunächst mit dem Befehl `sudo mysql -p -u root` als root Benutzer in der Datenbank ein. Im Anschluss daran legt man mit dem Befehl `CREATE USER 'pmauser'@'%' IDENTIFIED BY 'sql2018';` den neuen Benutzer pmauser mit dem Passwort „sql2018“ an. Das % -Symbol teilt der MySQL-Datenbank mit, dass sich dieser Benutzer von überall aus anmelden kann. Zur Erhöhung der Sicherheit kann stattdessen auch eine IP-Adresse verwendet werden. Um zu überprüfen, ob der Benutzer erfolgreich angelegt wurde, kann man sich mit dem Befehl `select host, user from mysql.user;` alle Benutzer und deren Host anzeigen lassen. Um sich die Rechte des Benutzers „pmauser“ anzeigen lassen zu können, gibt man `SHOW GRANTS FOR 'pmauser'@'%'`; in der Konsole ein. Da die Standardrechte `GRANT USAGE ON *.* TO 'pmauser'@'%'` nicht für alle Datenbankoperationen ausreichen, werden dem Benutzer „pmauser“ mit dem Befehl `GRANT ALL PRIVILEGES ON *.* TO 'pmauser'@'%' WITH GRANT OPTION;` sämtliche Rechte zugewiesen. Die Befehle zur Erstellung des Benutzers sowie das zuweisen der Privilegien ist in Abbildung 17 dargestellt.

```
mysql> CREATE USER 'pmauser'@'%' IDENTIFIED BY 'sql2018';
Query OK, 0 rows affected (0.00 sec)

mysql> select host, user from mysql.user;
+-----------+------------------+
| host      | user             |
+-----------+------------------+
| %         | pmauser          |
| localhost | areisenauer      |
| localhost | debian-sys-maint |
| localhost | jheym            |
| localhost | mysql.session    |
| localhost | mysql.sys        |
| localhost | phpmyadmin       |
| localhost | root             |
+-----------+------------------+
8 rows in set (0.00 sec)

mysql> SHOW GRANTS FOR 'pmauser'@'%';
+-----------------------------------+
| Grants for pmauser@%              |
+-----------------------------------+
| GRANT USAGE ON *.* TO 'pmauser'@'%' |
+-----------------------------------+
1 row in set (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON *.* TO 'pmauser'@'%' WITH GRANT OPTION;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS FOR 'pmauser'@'%';
+-------------------------------------------------------------------+
| Grants for pmauser@%                                              |
+-------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'pmauser'@'%' WITH GRANT OPTION |
+-------------------------------------------------------------------+
```

Abbildung 17: Anlegen des Benutzers "pmauser" und zuweisen sämtlicher Privilegien

Nach erfolgreichem Login als pmauser auf der Weboberfläche erscheint die Fehlermeldung „Ab sofort muss ein geheimes Passwort zur Verschlüsselung in der Konfigurationsdatei gesetzt werden (blowfish_secret)“. Um das Passwort für die Verschlüsselung setzten zu können wechselt man zunächst in den Ordner /var/www/html/phpMyAdmin. Dort kopiert man mit dem Befehl `cp -p config.sample.inc.php config.inc.php` die Datei config.sample.inc.php in die Datei config.inc.php. Die Option -p führt dazu, dass die kopierte Datei die Zugriffsrechte und Eigentümer des Originals erhalten bleiben (vgl. F. Kalhammer 2001). Die Datei config.inc.php wird nun als Super-User geöffnet und in der Zeile `$cfg['blowfish_secret'] =` wird ein 32 Zeichen langes Passwort für die Verschlüsselung eingegeben (siehe Abbildung 18).

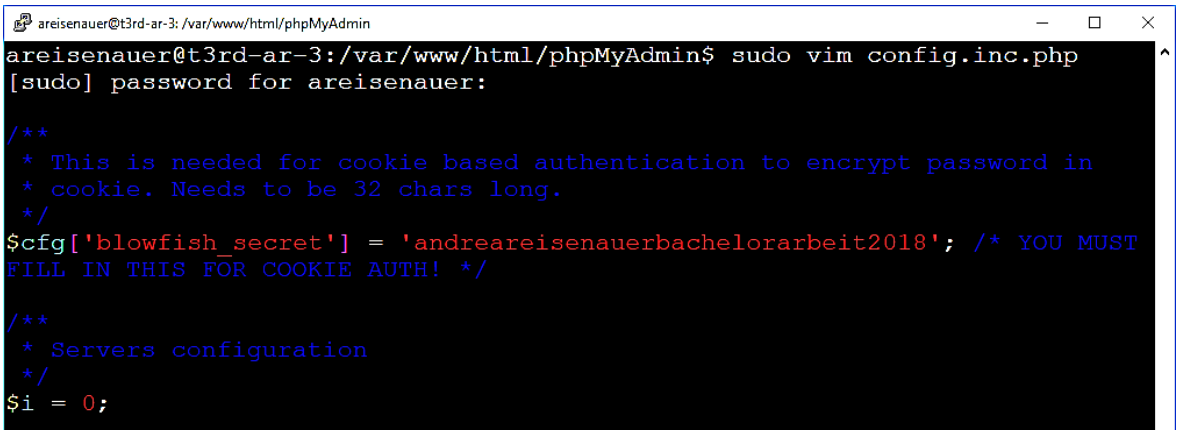


Abbildung 18: Eingabe des blowfish_secret

2.2.3 Installation von Java

Als Programmiersprache für die Software für das Performancemonitoring wurde Java gewählt. Die Vorteile von Java als Programmiersprache sind in Kapitel 4 dargelegt.

Bei ersten Tests wurde das Java Programm auf dem lokalen Rechner ausgeführt und die Verbindung mit den auf dem Hochschulserver installierten Datenbanken erfolgte per Virtual Private Network (VPN). Bei diesem Vorgehen konnten jedoch zwei Probleme identifiziert werden. Zum einen kam es bei Störungen der Internet-Verbindung zu Unterbrechungen der VPN-Verbindung und dadurch zu einer Verfälschung der Messergebnisse oder sogar zum Abbruch des Programms. Zum anderen hängt die Zeit, die man zum Auslesen der Daten aus der Datenbank benötigt, sehr stark von der eingesetzten Netzwerktechnik und deren Datendurchsatz ab. „Der Datendurchsatz gibt die Netto-Datenmenge pro Zeit an, die über ein kabelgebundenes oder kabelloses Netz übertragen werden kann“ (Datendurchsatz 2018).

Um möglichst genaue Messergebnisse zu erhalten und den Einfluss der oben genannten Randbedingungen zu minimieren, wird das Programm für das Performancemonitoring unmittelbar auf dem Server gespeichert. Java ist standardmäßig nicht im Ubuntu Paket enthalten und muss deshalb nachträglich installiert werden.

Für die Installation von Java auf dem Server muss man lediglich den Befehl `sudo apt-get install openjdk-8-jdk` ausführen. Die erforderlichen Pakete werden anschließen vom Ubuntu Server heruntergeladen und die notwendigen Java Umgebungsvariablen werden automatisch gesetzt. Um zu kontrollieren, ob Java korrekt installiert wurde und sich die installierte Java-Version anzeigen zu lassen, führt man wie in Abbildung 19 dargestellt, den Befehl `java -version` aus.

```
areisenauer@t3rd-ar-3:~$ java -version
openjdk version "1.8.0_191"
OpenJDK Runtime Environment (build 1.8.0_191-8u191-b12-0ubuntu0.18.04.1-b12)
OpenJDK 64-Bit Server VM (build 25.191-b12, mixed mode)
```

Abbildung 19: Anzeigen der installierten Java-Version

2.3 Performancemonitoring mit MongoDB

2.3.1 Installation von MongoDB

Die Installation von MongoDB ist vergleichsweise einfach. Es muss lediglich der Befehl `sudo apt-get install mongodb` ausgeführt werden. Der Login in die Datenbank erfolgt ohne Authentifizierung mithilfe der Befehlszeile `mongo`.

Auch bei der MongoDB muss für den externen Zugriff auf die Datenbank die bind-Adresse geändert werden. Die Konfigurationsdatei der MongoDB liegt im Verzeichnis `/etc` und heißt `mongodb.conf`. Ebenso wie bei der MySQL-Datenbank werden die Änderungen erst nach einem Neustart wirksam. Der Befehl hierfür lautet `sudo service mongod restart`.

2.3.2 Verwendung von Robo3T als grafische Oberfläche für MongoDB

Ebenso wie für MySQL gibt es auch für MongoDB eine Software-Anwendung, die die Arbeit mit der Datenbank erleichtert. Bei Robo3T handelt es sich um eine kostenlose grafische Benutzeroberfläche mit eingebetteter Shell.

Nach der erfolgreichen Installation von Robo3T, die Installationsdatei findet man zum Download auf der Homepage https://robomongo.org/download, startet man die

Anwendung. Zunächst gibt man, wie in Abbildung 20 zu sehen, einen frei wählbaren Namen zur Identifikation der Verbindung sowie die IP-Adresse des MongoDB Servers ein.

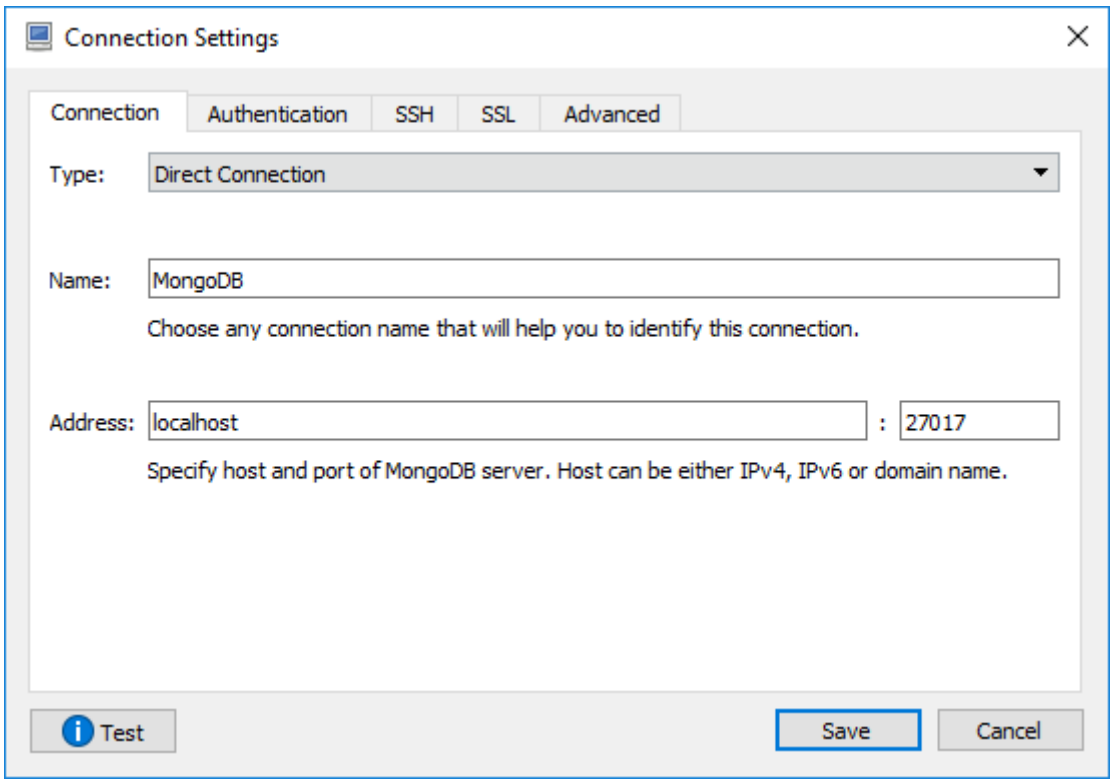


Abbildung 20: Eingeben der Verbindungsdaten zur MongoDB

Da für die Kommunikation mit dem Server das Netzwerkprotokoll SSH verwendet wird, werden die weiteren Einstellungen in der Registerkarte SSH vorgenommen. Als Autorisierungsmethode kann zwischen Private Key und Passwort wählen. Alle weiteren Details für die korrekte Befüllung der Felder kann man Abbildung 21 entnehmen.

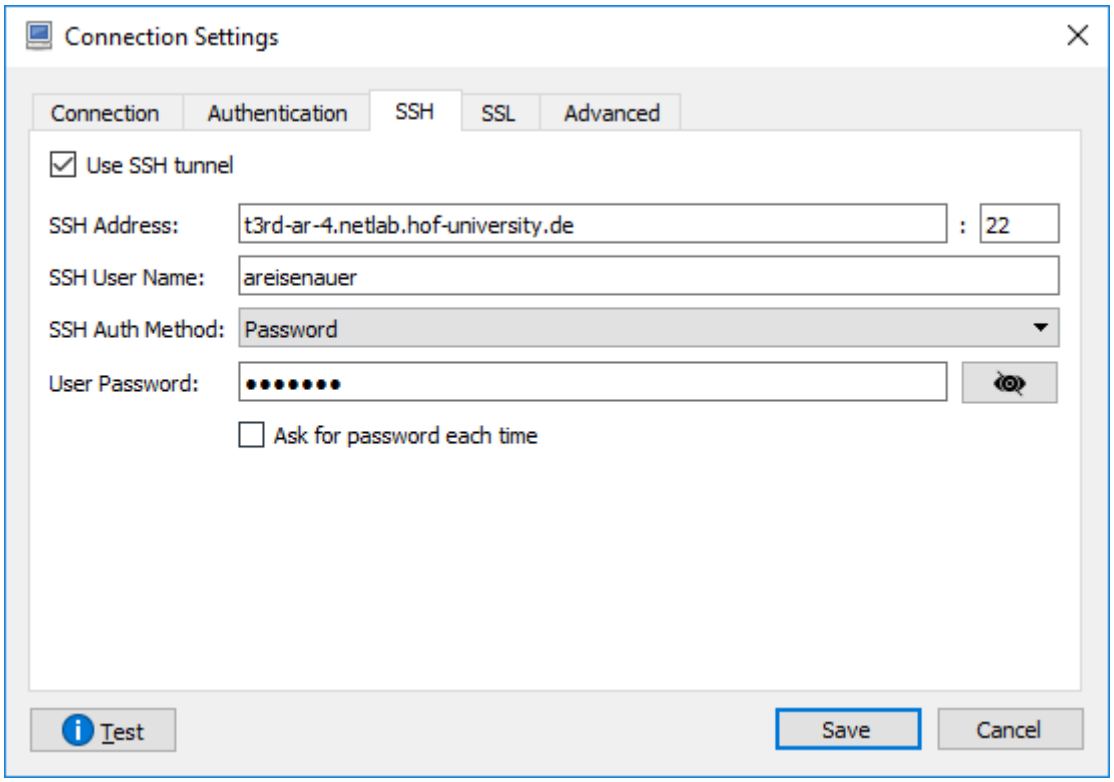


Abbildung 21: SSH Einstellungen für die Verbindung mit dem MongoDB Server

Sofern alle Einstellungen korrekt vorgenommen wurden, erhält man nach dem Klicken des Buttons „Test“ eine positive Diagnostic Rückmeldung.

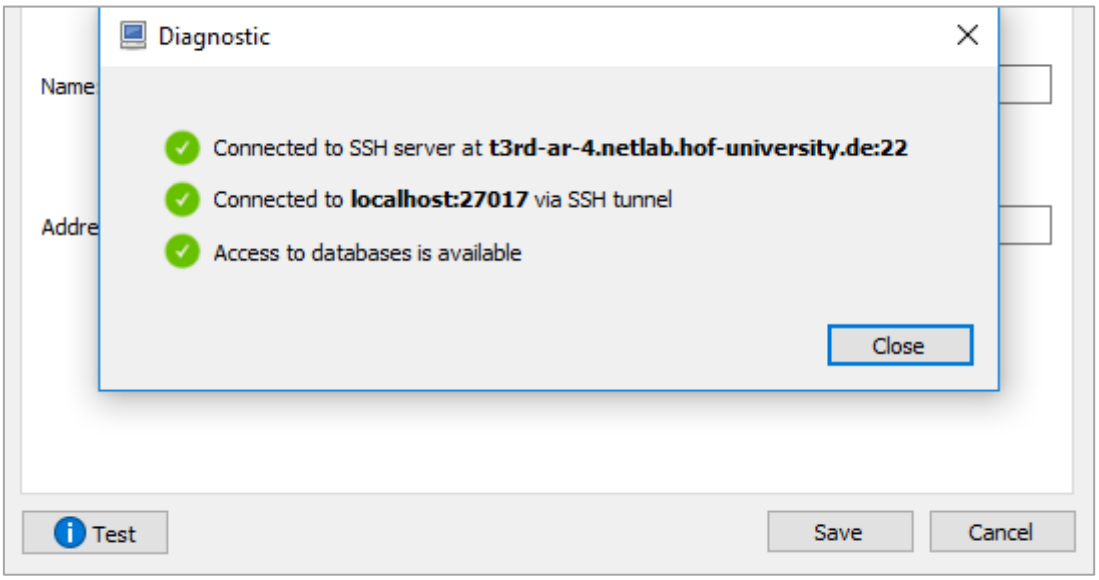


Abbildung 22: Erfolgreiche Verbindung zum MongoDB Server

3 Import der Geodaten

Die für diese Praxisarbeit verwendeten Geodaten sowie die Schemata zum Anlegen der Datenbanktabellen in der MySQL-Datenbank werden dankenswerterweise vom Landesamt für Digitalisierung, Breitband und Vermessung (LDBV) zur Verfügung gestellt. Das LDBV verwendet standardmäßig Oracle-Datenbanken.

Zur Speicherung von Geodaten wird in Oracle der herstellerspezifische Datentyp SDO_GEOMETRY verwendet. Die unterschiedlichen Datentypen zur Speicherung von Geodaten in Datenbanken sind in der Bachelorarbeit detailliert erläutert. Da MySQL und MongoDB mit dem Datentyp SDO_GEOMETRY nicht umgehen können, wurden die Geodaten vom LDBV beim Exportieren der Daten als csv-Datei mit Hilfe der Funktion SDO_UTIL.TO_WKTGEOMETRY(geom) in das für MySQL und MongoDB geeignete WKT-Format umgewandelt. WKT ist ein allgemeingültiger Standard zur Speicherung von Geodaten.

Alle im folgenden Teil beschriebenen Schritte werden für die MySQL-Datenbank und die MongoDB jeweils identisch durchgeführt.

Um die csv-Dateien in die beiden Datenbanken importieren zu können, müssen diese zunächst vom USB-Stick auf den jeweiligen Server verschoben werden. Diese Datenmigration wurde mittels des Programms WinSCP durchgeführt, da WinSCP es ermöglicht, sich mit dem entfernten Server für die MySQL-Datenbank (http://t3rd-ar-3.netlab.hof-university.de) bzw. mit dem Server für die MongoDB (http://t3rd-ar-4.netlab.hof-university.de) über das SSH Netzwerkprotokoll zu verbinden (Martin 2014). Das Hochladen der Dateien erfolgt per Drag & Drop und ist in Abbildung 23 exemplarisch für die MySQL-Datenbank dargestellt.

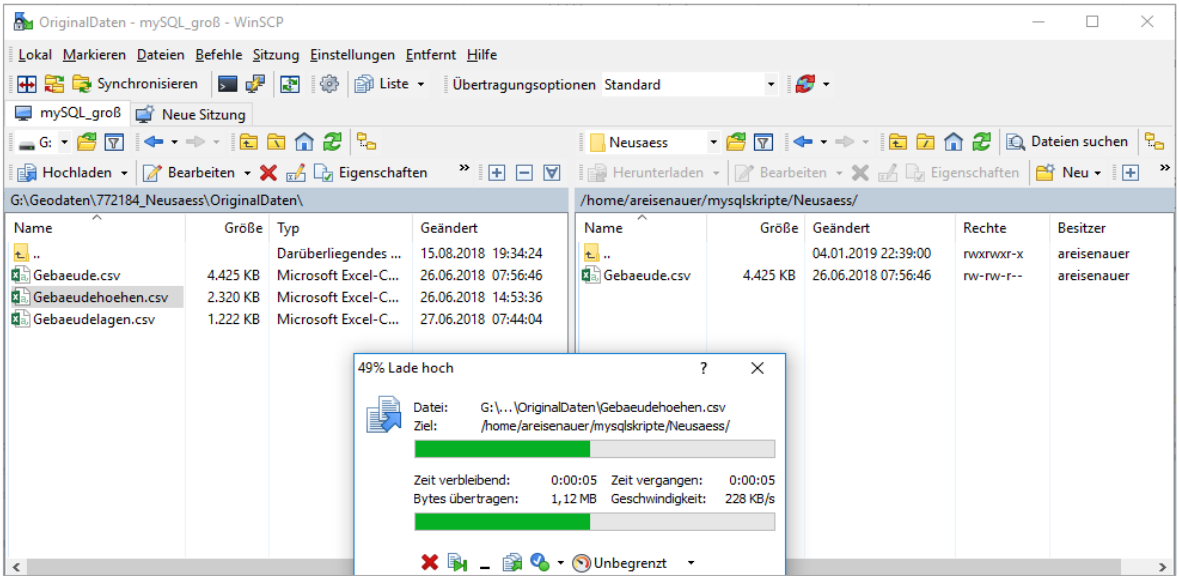


Abbildung 23: Datenmigration vom USB-Stick auf den Server

Alle weiteren Schritte werden nun wieder auf der Konsole ausgeführt. Da die Datenbanken im UTF-8 Format kodiert werden, müssen auch die zu importierenden Daten im UTF-8 Format kodiert vorliegen, da ansonsten Probleme beim Import oder beim Anzeigen der Daten auftreten können. Um herauszufinden in welchem Zeichenformat die Dateien vorliegen, führt man auf der Konsole den Befehl file +[Dateiname] aus. Daraus ergibt sich, dass die Dateien im ISO-8859 Format vorliegen und mit dem Befehl iconv konvertiert werden müs sen. Dem Befehl `iconv` folgt die Option `-f`, die den Zeichensatz angibt, aus dem konvertiert werden soll. Der Zeichensatz, in den Konvertiert werden soll wird mit der Option `-t` angegeben. Im Anschluss daran folgt der Name der Datei, die konvertiert werden soll gefolgt von der Option `-o` und dem Namen der neuen Datei. Der gesamte Befehl ist in Abbildung 24 aufgeführt.

```
areisenauer@t3rd-ar-3:~/mysqlskripte/Neusaess$ file Gebaeude.csv
Gebaeude.csv: ISO-8859 text, with very long lines, with CRLF line terminators
areisenauer@t3rd-ar-3:~/mysqlskripte/Neusaess$ iconv -f ISO-8859-1 -t UTF-8 Gebaeude.csv  -o GebaeudeUTF.csv
areisenauer@t3rd-ar-3:~/mysqlskripte/Neusaess$ file GebaeudeUTF.csv
GebaeudeUTF.csv: UTF-8 Unicode text, with very long lines, with CRLF line terminators
```

Abbildung 24: Zeichensatzkonvertierung von ISO-8859 zu UTF-8

Nicht vorhandene Werte, also NULL-Werte, sind in den ursprünglichen csv-Dateien des LDBV als (null) dargestellt. Damit die Felder in der MySQL-Datenbank und der MongoDB auch wirklich nicht vorhanden sind, müssen diese Strings wahlweise durch „\N“ oder kein Zeichen ersetzt werden. Mit dem Linux-Befehl `sed s/\(null\)//g GebaeudeUTF.csv > GebaeudeUTF_ohneNULL.csv` wird die Zeichenkette (null) in der Datei GebaeudeUTF.csv durch kein Zeichen ersetzt und das Ergebnis wird in die neue Datei GebaeudeUTF_ohneNULL.csv geschrieben.

Die so aufbereiteten csv-Daten können nun vom Server in die jeweilige Datenbank importiert werden.

3.1 Import der Geodaten in die MySQL Datenbank

Oracle verwendet einige herstellerspezifische Standards, weshalb bei der Datenmigration von einer Oracle SQL-Datenbank in eine MySQL-Datenbank besondere Vorsicht geboten ist.

Sowohl bei der Bezeichnung der Datentypen als auch bei den zur Speicherung von Geodaten zur Verfügung stehenden Datentypen gibt es erhebliche Unterschiede zwischen Oracle und MySQL. So heißt beispielsweise der Datentyp zur Speicherung von Zeichenketten variabler Länge in Oracle VARCHAR2 in MySQL dagegen VARCHAR. Auch bei der Speicherung von Zahlen gibt es Unterschiede. Für ganzzahlige Werte wird in Oracle der

Datentyp NUMBER verwendet, in MySQL dagegen INTEGER. Eine Gegenüberstellung der wichtigsten Datentypen ist in Tabelle 1 beschrieben und stammt aus der Oracle Dokumentation.

MySQL Datentyp	Oracle Datentyp
CHAR	CHAR
DATE	DATE
DATETIME	DATE
DOUBLE	FLOAT
INTEGER	NUMBER
CHAR	CHAR
VARCHAR	VARCHAR2
MULTIPOLYGON	SDO_Geometry

Tabelle 1: Gegenüberstellung der MySQL und Oracle Datentypen

Aus diesem Grund müssen die vom LDBV gelieferten sql-Dateien zum Anlegen der Datenbanktabelle an die von MySQL implementierten Datentypen angepasst werden. Der Original SQL-Befehl zum Anlegen der Datenbanktabelle Gebaeude in der Datenbank augsburg ist in Abbildung 25 und der für die MySQL-Datenbank aufbereitete SQL-Befehl in Abbildung 26 dargestellt.

```
CREATE TABLE augsburg.GEBAEUDE  (
    ID              NUMBER(10) NOT NULL,
    OBNR            CHAR(16) NULL,
    GEMEINDE        NUMBER(6) NULL,
    STRASSE         VARCHAR2(110) NULL,
    HAUSNR          NUMBER(5) NULL,
    ZUSATZ          VARCHAR2(500) NULL,
    FUNKTION        NUMBER(4) NULL,
    GEBHOEHE        NUMBER(7,3) NULL,
    SHAPE           SDO_GEOMETRY NULL,
    ERFASSER        VARCHAR2(30) NULL,
    ERFASSUNGSZEIT  TIMESTAMP(0) NULL,
    AENDERER        VARCHAR2(30) NULL,
    AENDERUNGSZEIT  TIMESTAMP(0) NULL,
    CONSTRAINT GEB_PK PRIMARY KEY(ID)
);
```

Abbildung 25: Original SQL-Befehl zum Anlegen der Tabelle Gebaeude

```
CREATE TABLE augsburg.GEBAEUDE  (
    ID              INTEGER(10) NOT NULL,
    OBNR            CHAR(16) NULL,
    GEMEINDE        INTEGER(6) NULL,
    STRASSE         VARCHAR(110) NULL,
    HAUSNR          FLOAT(5) NULL,
    ZUSATZ          VARCHAR(500) NULL,
    FUNKTION        FLOAT(4) NULL,
    GEBHOEHE        FLOAT(7,3) NULL,
    MULTIPOLYGON    GEOMETRY NULL,
    ERFASSER        VARCHAR(30) NULL,
    ERFASSUNGSZEIT  TIMESTAMP(0) NULL,
    AENDERER        VARCHAR(30) NULL,
    AENDERUNGSZEIT  TIMESTAMP(0) NULL,
    CONSTRAINT GEB_PK PRIMARY KEY(ID)
);
```

Abbildung 26: Modifizierter SQL-Befehl zum Anlegen der Tabelle Gebaeude

Zum Importieren der Daten eignet sich die Anweisung LOAD DATA besonders gut, da sie die Zeilen einer Textdatei mit sehr hoher Geschwindigkeit in eine Datenbanktabelle schreibt (vgl. Oracle). Die Datei loadGebaeude.sql (siehe Abbildung 27) zum Laden der Gebäudedaten in die Datenbanktabelle Gebäude wird im Folgenden näher erläutert.

Die Anweisung beginnt mit LOAD DATA gefolgt von dem optionalen Parameter LOCAL. Das Wort LOCAL beeinflusst, den Speicherort an dem erwartet wird, die Datei zu finden. Wenn LOCAL angegeben ist, wird die Datei vom Clientprogramm auf dem Clienthost gelesen und an den Server gesendet. Hierfür wird vorab eine Kopie der Datei in dem Verzeichnis erstellt, in dem der MySQL Server temporäre Dateien speichert. Wenn nicht genügend Speicherplatz für die Kopie in diesem Verzeichnis vorhanden ist, kann die Anweisung LOAD DATA LOCAL fehlschlagen. Ist die Option LOCAL nicht angegeben, muss sich die Datei auf dem Serverhost befinden und wird direkt vom Server gelesen. (vgl. Oracle)

Im Anschluss an LOAD DATA LOCAL INFILE wird der Name der Datei, deren Daten in die in die Datenbank importiert werden sollen ein einfachen Hochkommata angegeben. In unserem Beispiel soll die Datei GebaueudeUTF.csv in die Datenbank importiert werden. Da sich sowohl die Datei loadGebaeude.sql als auch die Datei GebaueudeUTF.csv im gleichen Ordner befinden muss Speicherort nicht weiter spezifiziert werden.

Nach dem Stichwort INTO TABLE wird die Datenbanktabelle angegeben, in der die Daten aus der Datei GebaueudeUTF.csv gespeichert werden sollen.

Zur Kennzeichnung der einzelnen Datenfelder wird bei der hier verwendeten csv-Datei das Semikolon als Trennzeichen verwendet. Sollten andere Trennzeichen verwendet werden, kann man diese jederzeit mit Hilfe von FIELDS TERMINATED BY spezifizieren.

Die Option \n (= newline) gibt an, dass das Ende eines Datensatzes mit einem Zeilenumbruch markiert wird und in der nächsten Zeile ein neuer Datensatz beginnt.

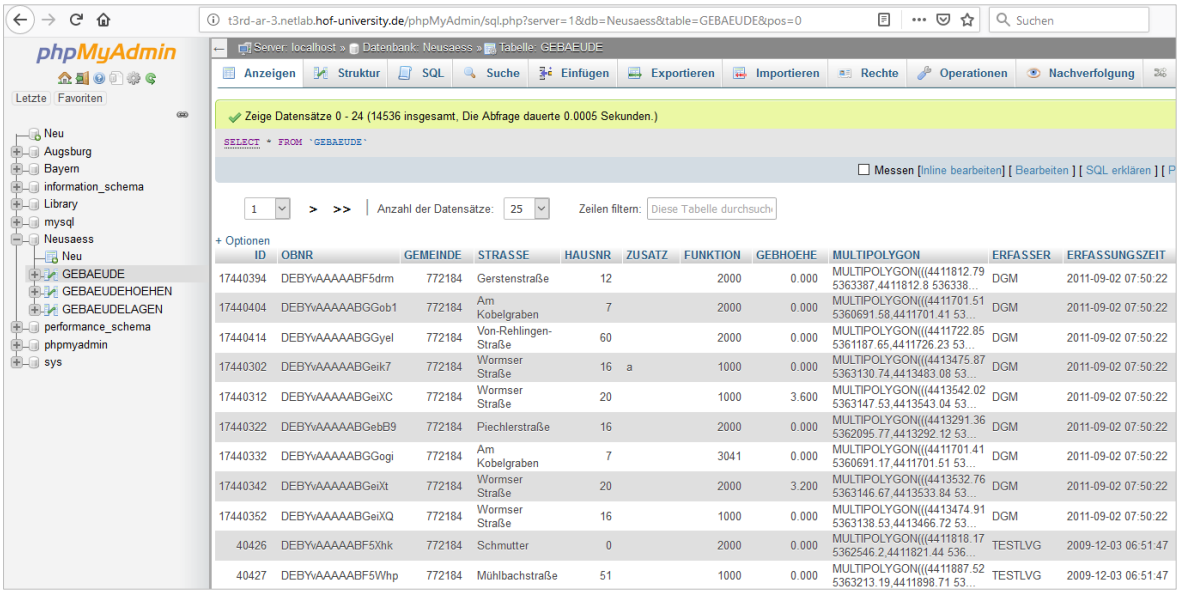
Damit die erste Zeile, die lediglich die Spaltennamen angibt, nicht importiert wird. wird IGNORE 1 LINES angegeben.

Um die Geodaten auch wirklich als solche speichern zu können, werden die WKT-Daten der Spalte Multipolygon in den Geodatentyp Multipolygon der MySQL-Datenbank transformiert. Diese Umwandlung erfolgt mit der Funktion ST_GeomFromText() und ist in der letzten Zeile der Datei loadGebaeude.sql zu sehen. (MySQL 5.7 Reference Manual)

```
LOAD DATA LOCAL INFILE './GebaeudeUTF.csv'
    INTO TABLE GEBAEUDE
    FIELDS
        TERMINATED BY ';'
        OPTIONALLY ENCLOSED BY '\''
    LINES
        TERMINATED BY '\n'
    IGNORE 1 LINES
        (ID, OBNR, GEMEINDE, STRASSE, HAUSNR, ZUSATZ,
         FUNKTION, GEBHOEHE, @MULTIPOLYGON , ERFASSER,
         ERFASSUNGSZEIT, AENDERER, AENDERUNGSZEIT)
        SET MULTIPOLYGON=ST_GeomFromText(@MULTIPOLYGON);
```

Abbildung 27: Abbildung der Datei loadGebaeude.sql

Um nun den Datenimport zu starten, wechselt man zunächst in den Ordner, in dem sowohl die csv-Dateien für den Import als auch die loadGebaeude.sql abgelegt sind. Die Dateien für die Datenbank neusaess sind unter dem folgenden Dateipfad zu finden: /home/areisenauer/mysqlskripte/772184_Neusaess. Der Import der Daten erfolgt nun mit der Kommadozeileneingabe `mysql -u pmauser -p neusaess < loadGebaeude.sql`. Der Angabe der Datenbank, also in diesem Beispiel neusaess, ist dabei obligatorisch. Der erfolgreiche Import der Daten aus der Datei GebaueudeUTF.csv in die Tabelle Gebaeude der Datenbank neusaess ist in Abbildung 28 abgebildet.



ID	OBNR	GEMEINDE	STRASSE	HAUSNR	ZUSATZ	FUNKTION	GEBHOEHE	MULTIPOLYGON	ERFASSER	ERFASSUNGSZEIT
17440394	DEBYvAAAAABF5drm	772184	Gerstenstraße	12		2000	0.000	MULTIPOLYGON(((4411812.79 5363387,4411812.8 536338…	DGM	2011-09-02 07:50:22
17440404	DEBYvAAAAABGGob1	772184	Am Kobelgraben	7		2000	0.000	MULTIPOLYGON(((4411701.51 5360691.58,4411701.41 53…	DGM	2011-09-02 07:50:22
17440414	DEBYvAAAAABGGyel	772184	Von-Rehlingen-Straße	60		2000	0.000	MULTIPOLYGON(((4411722.85 5361187.65,4411726.23 53…	DGM	2011-09-02 07:50:22
17440302	DEBYvAAAAABGeik7	772184	Wormser Straße	16	a	1000	0.000	MULTIPOLYGON(((4413475.87 5363130.74,4413483.08 53…	DGM	2011-09-02 07:50:22
17440312	DEBYvAAAAABGeiXC	772184	Wormser Straße	20		1000	3.600	MULTIPOLYGON(((4413542.02 5363147.53,4413543.04 53…	DGM	2011-09-02 07:50:22
17440322	DEBYvAAAAABGebB9	772184	Piechlerstraße	16		2000	0.000	MULTIPOLYGON(((4413291.36 5362095.77,4413292.12 53…	DGM	2011-09-02 07:50:22
17440332	DEBYvAAAAABGGogi	772184	Am Kobelgraben	7		3041	0.000	MULTIPOLYGON(((4411701.41 5360691.17,4411701.51 53…	DGM	2011-09-02 07:50:22
17440342	DEBYvAAAAABGeiXt	772184	Wormser Straße	20		2000	3.200	MULTIPOLYGON(((4413532.76 5363146.67,4413533.84 53…	DGM	2011-09-02 07:50:22
17440352	DEBYvAAAAABGeiXQ	772184	Wormser Straße	16		1000	0.000	MULTIPOLYGON(((4413474.91 5363138.53,4413466.72 53…	DGM	2011-09-02 07:50:22
40426	DEBYvAAAAABF5Xhk	772184	Schmutter	0		2000	0.000	MULTIPOLYGON(((4411818.17 5362546.2,4411821.44 536…	TESTLVG	2009-12-03 06:51:47
40427	DEBYvAAAAABF5Whp	772184	Mühlbachstraße	51		1000	0.000	MULTIPOLYGON(((4411887.52 5363213.19,4411898.71 53…	TESTLVG	2009-12-03 06:51:47

Abbildung 28: Anzeigen der importierten Daten mit phpMyAdmin

3.2 Import der Geodaten in die MongoDB

Bei der MongoDB handelt es sich um eine schemafreie Datenbank. Deswegen ist es im Gegensatz zu SQL-Datenbanken, bei denen vor dem Einfügen von Daten das Schema einer Tabelle deklariert werden muss, bei MongoDB nicht notwendig, dass die Dokumente dasselbe Schema haben. Die Dokumente in einer Collection (= Sammlung) müssen nicht dieselbe Anzahl von Feldern haben und der Datentyp für ein Feld kann bei Dokumenten innerhalb einer Sammlung unterschiedlich sein. Die Dokumente in einer Sammlung weisen in der Praxis jedoch meist eine ähnliche Struktur auf. Regeln für die Validierung von Dokumenten vor dem Aktualisieren oder Einfügen in eine Sammlung können optional hinzugefügt werden. (MongoDB Manual 2019a)

Die vom Vermessungsamt zur Verfügung gestellten Daten stammen ursprünglich aus einer relationalen Datenbank und weisen deshalb alle die gleiche Struktur aus. Da, abgesehen von den zur Verfügung gestellten Daten, keine neuen Daten in die MongoDB importiert werden sollen und es nicht dem MongoDB Standard entspricht, wird im Rahmen dieser Praxisarbeit auf das Erstellen von Regeln für die Dokumentvalidierung verzichtet.

Vor dem Import der Geodaten muss zunächst die Datenbank und die Collection angelegt werden, in die die Daten importiert werden sollen. Auf der Konsole erfolgt der Login in die MongoDB mit dem Befehl `mongo`. Um in die gewünschte Datenbank zu wechseln oder, falls diese noch nicht existiert, die Datenbank anzulegen führt man den Befehl `use`

`Datenbankname` aus. Die drei Datenbanken Neusäß, Augsburg und Bayern werden demnach mit den folgenden Befehlen angelegt:

```
use neusaess

use augsburg

use bayern
```

Um zu überprüfen, ob die Datenbanken erfolgreich angelegt wurden, gibt man auf der Konsole `show databases` ein. Alle vorhandenen Datenbanken werden nun aufgelistet.

Im Anschluss an das Erstellen der Datenbanken wird für jede Datenbank eine Collection angelegt, in der die Gebäudedaten gespeichert werden. Der Befehl zum Erstellen der Collection gebaeude lautet für jede Datenbank `db.createCollection("gebaeude")`. Abbildung 29 zeigt die Robo3T-Ansicht nach dem Anlegen der Datenbanken und Erstellen der Collections.

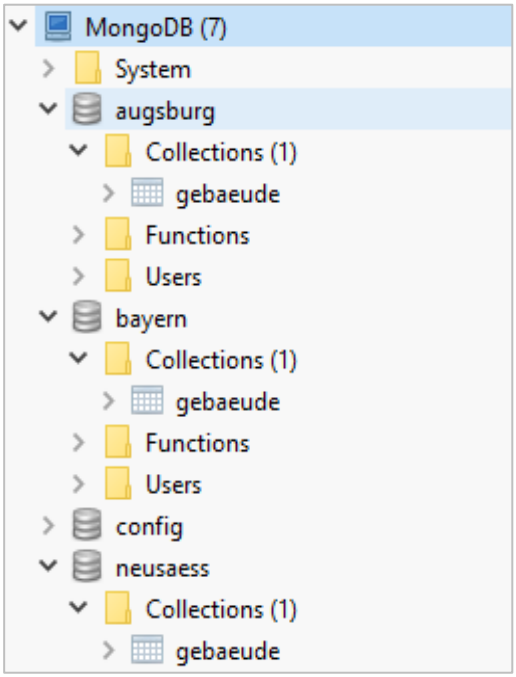


Abbildung 29: Auflisten der vorhandenen Datenbanken und ihrer Collections

Im Gegensatz zu LOAD DATA bei MySQL kann man bei mongoimport das Trennzeichen der csv-Datei nicht genauer spezifizieren. Mongoimport akzeptiert lediglich Tabs oder Kommata als Trennzeichen zwischen einzelnen Elementen eines Datensatzes (vgl. Alex Howansky 2016). Da das Komma zur Trennung der einzelnen Koordinaten verwendet wird, kann es nicht als Trennzeichen für die einzelnen Elemente eines Datensatzes verwendet

werden. Deshalb werden alle Semikolons in den csv- Dateien durch Tabulatoren ersetzt und anschließend der Import in die Datenbank durchgeführt. Dies geschieht mit dem Befehl `sed -e 's/;/\t/g' GebaeudeUTF_ohneNull.csv | mongoimport -d neusaess -c gebaeude --type tsv --headerline.` Nach der Option `-d` wird der Name der Datenbank angegeben, in die die Daten importiert werden sollen. Die Collection der Datenbank wird nach der Option `-c` angegeben. Nach `--type` gibt man den Dateityp der Datei an, aus der die Daten in die Datenbank übertragen werden sollen. Bei dem Dateityp tsv handelt es sich um ein textbasiertes Datenformat, das aus durch Tabulatoren getrennten Werten besteht. Die Option `--headerline` führt dazu, dass die erste Zeile einer csv oder tsv Datei als Feldname verwendet wird. Vergisst man diese Option importiert mongoimport die erste Zeile als separates Dokument. (vgl. MongoDB Manual 2019b)

Der erfolgreiche Import der Daten aus der Datei GebaeudeUTF_ohneNull.csv in die Tabelle gebaeude der Datenbank augsburg ist in Abbildung 30 abgebildet. Die Spalte _id wurde von MongoDB automatisch hinzugefügt und dient der eindeutigen Identifikation der Dokumente in der Datenbank, da, wie in Kapitel 3.3.3 der Bachelorarbeit beschrieben, die Speicherung in der MongoDB mit Hilfe von Schlüssel-Wert-Paaren erfolgt.

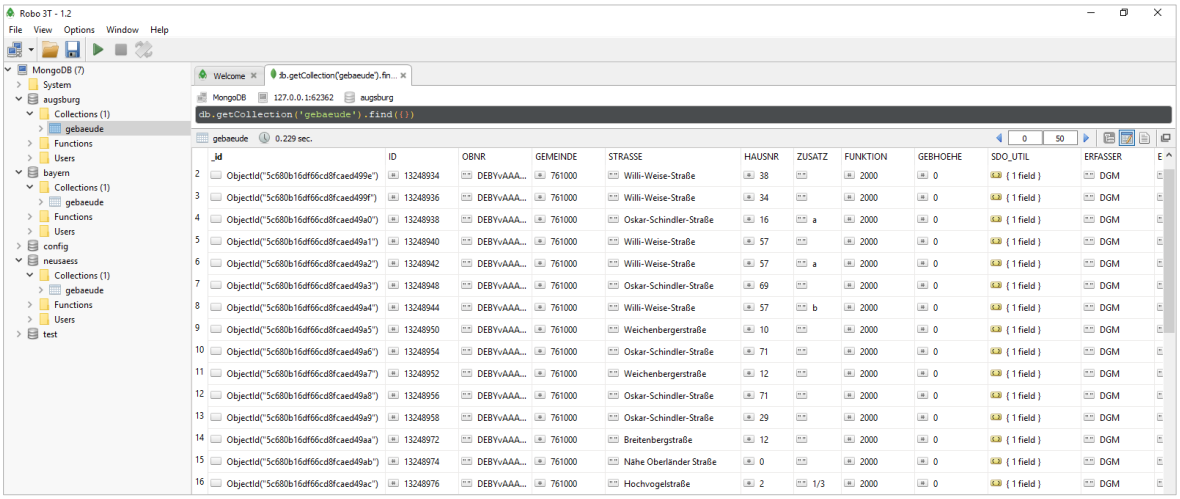


Abbildung 30: Anzeigen der importierten Daten mit Robo3T

Da die Geodaten in MongoDB im GeoJSON-Format abgelegt werden, entstehen beim import der Daten zwei Dokumente. Das Hauptdokument enthält Daten wie ID, Adresse, Hausnummer etc. und das Unterdokument enthält die Geodaten. Ein solches Dokument ist in Abbildung 31: Darstellung eines Dokuments in der MongoDB abgebildet.

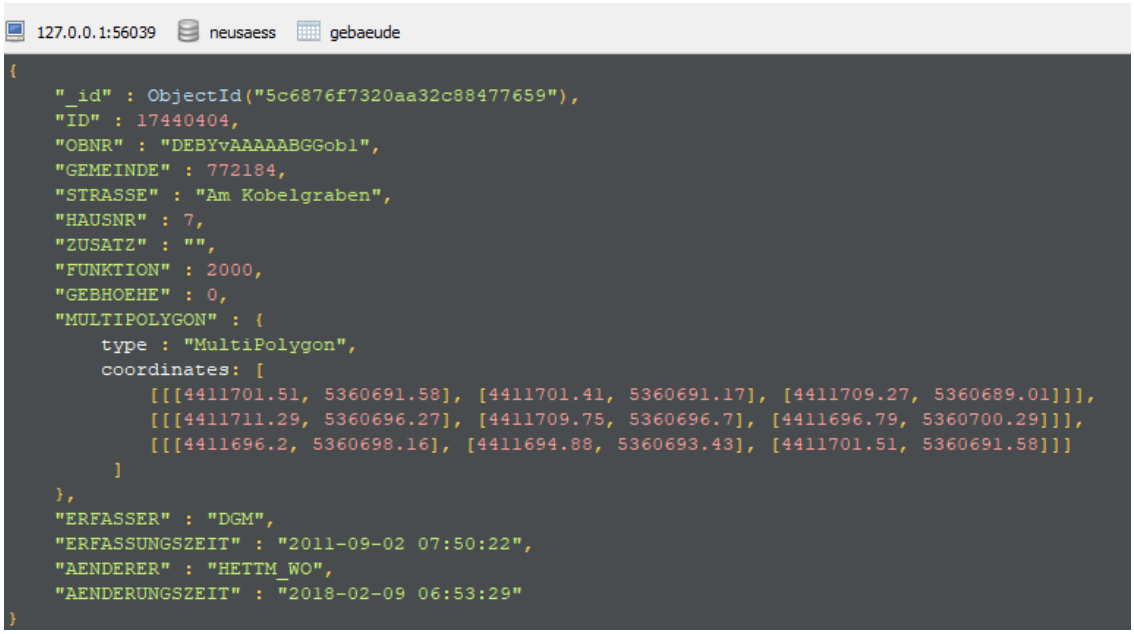


Abbildung 31: Darstellung eines Dokuments in der MongoDB

Nachdem nun alle Daten in die beiden Datenbanken importiert wurden, kann mit Erstellen des Programms für das Performancemonitoring begonnen werden.

4 Durchführung des Performancemonitorings

Der Ablauf der Performancemonitorings wird im Kapitel 5.3.2 der Bachelorarbeit beschrieben. Um sich den Ablauf des Performancemonitorings nochmal ins Gedächtnis rufen zu können, ist der Programmablaufplan in Abbildung 32 dargestellt.

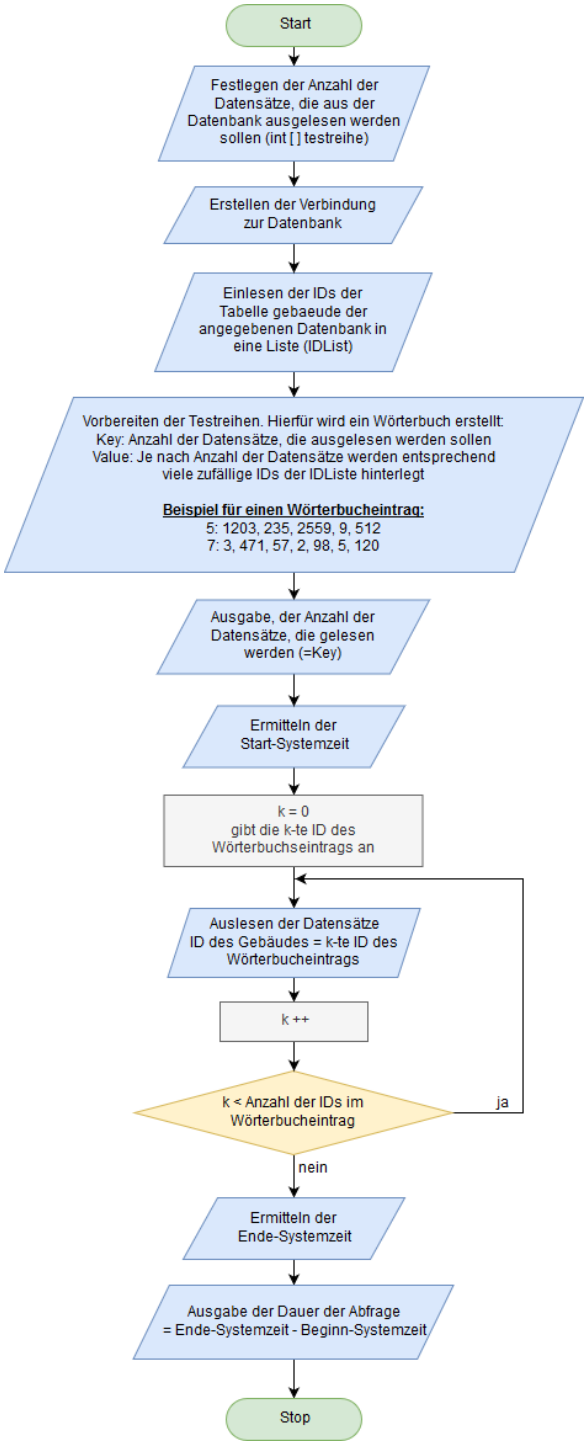


Abbildung 32: Programmablaufplan

Das Programm für das Performancemonitoring wird in Java geschrieben. Ein wichtiger Vorteil von Java ist seine Plattformunabhängigkeit bzw. Betriebssystemunabhängigkeit. Java bindet sich nicht an einen bestimmten Prozessor oder eine bestimmte Systemarchitektur, stattdessen generiert der Compiler Bytecode, den eine Laufzeitumgebung dann abarbeitet. (vgl. Ullenboom 2018)

Darüber hinaus stellt Java zahlreiche Bibliotheken und Schnittstellen zur Verfügung. Datenbankverbindungen zu MySQL und MongoDB lassen sich dank der JDBC-Schnittstelle mit Java besonders leicht realisieren. Außerdem wird Java an der Hochschule Hof bereits ab dem ersten Semester erlernt und im Verlauf des Studiums in zahlreichen weiteren Vorlesungen und Projektarbeiten vertieft.

4.1 Realisierung des Performancemonitorings

In diesem Kapitel wird der Quellcode des Performancemonitoring-Programms kurz beschrieben und gezeigt. Außerdem wird erläutert wie man das Java-Programm für das Single-User- bzw. das Multi-User-Monitoring startet. Die beschriebenen Schritte sind, abgesehen von den unterschiedlichen Dateinamen, für MySQL und MongoDB identisch.

Das Java-Programm ist in zwei Klassen unterteilt. Die Klasse Utils enthält sämtliche Vorbereitungen wie beispielsweise das Festlegen der Testreihen, den Aufbau des Wörterbuchs oder das Logging-Konzept. Die Klasse PerformanceSQL bzw. PerformanceNoSQL führt dann das datenbankspezifische Performancemonitoring durch.

Vor dem Beginn des Performancemonitorings müssen die Testdaten vorbereitet werden. In einem ersten Schritt werden die Testreihen festgelegt. Diese werden in einem Array vom Typ int gespeichert. Der zugehörige Codeabschnitt sieht wie folgt aus:

```
// Gibt die Anzahl der Datensätze an, die aus der Datenbank
// ausgelesen werden sollen
int []testreihe = { 10, 25, 50, 75,
                    100, 250, 500, 750,
                    1000, 2500, 5000, 7500,
                    10000, 12500, 15000, 17500,
                    20000, 22500, 25000, 27500, 30000 };
```

Im Anschluss daran wird für das Auslesen aller IDs der Tabelle Gebäude die Methode `idsAusDbLaden()` aufgerufen. Diese Methode erstellt zunächst eine Verbindung zu MySQL bzw. MongoDB und liest anschließend alle IDs der Tabelle gebauede aus. Die ausgelesenen IDs werden in einer IDListe gespeichert, die später die Basis für das Erstellen des Wörterbuchs darstellt

```
	/**
	 * Die Gebaeude in der Datenbanktabelle Gebaeude sind eindeutig
	 * über IDs idenzifiziert.
	 *
	 * Diese IDs werden zunächst in eine Liste geladen.
	 * Für das Auslesen der Daten beim Performancemoitoring werden
	 * zufällige IDs aus dieser Liste verwendet.
	 *
	 * @throws SQLException
	 */
public void idsAusDbLaden() throws SQLException {

	// Für MySQL:
	if (datenbanktyp == "SQL") {
		// Verbindung zur MySQL Datenbank wird hergestellt
		String database = "jdbc:mysql://192.168.74.243:3306/" + datenbank;
		String user = "pmauser";
		String pw = "sql2018";
		con = DriverManager.getConnection(database, user, pw);
		Statement stmt = con.createStatement();

		// Alle IDs der Tabelle Gebaeude der angegebenen Datenbank
		// (neusaess, augsburg, bayern)
		// werden ausgelesen
		ResultSet rs = stmt.executeQuery("Select ID From GEBAEUDE;");

		// Die ausgelesenen IDs werden zu einer LinkedList hinzugefügt
		while (rs.next()) {
			IDList.add(rs.getInt(1));
		}
		stmt.close();
	}

	// Für MongoDB:
	else if(datenbanktyp == "NoSQL") {

		// Verbindung zur MongoDB Datenbank wird hergestellt
		MongoClientURI uri = new MongoClientURI(
			"mongodb://192.168.74.244:27017/"+datenbank);
		client = new MongoClient(uri);
		MongoDatabase db = client.getDatabase(uri.getDatabase());
		MongoCollection<Document> gebaeude = db.getCollection("gebaeude");

		// Alle IDs der Tabelle Gebaeude der angegebenen Datenbank
		// (neusaess, augsburg, bayern)
		// werden ausgelesen
		cursor = gebaeude.find(new BasicDBObject()).projection(
				Projections.fields(Projections.include("ID"),
				Projections.excludeId())).iterator();


		// Die ausgelesenen IDs werden zu einer LinkedList hinzugefügt
		while (cursor.hasNext()) {
			Document d = cursor.next();
			IDList.add((Integer) d.get("ID"));
		}
	}
}
```

Für die Erstellung des Wörterbuchs werden jeder Testreihe zufällige IDs zugeordnet. Zufallszahlen können mit der Methode `Math.random()` erzeugt werden.

```
/**
 * Je nach Länge der Testreihe, werden zufällig IDs aus der
 * ID-Liste in das Wörterbuch geladen
 */
public void testreihenVorbereiten() {
      for (int i = 0; i < testreihe.length; i++) {
            int[] ds;
            ds = new int[testreihe[i]];
            for (int j = 0; j < ds.length; j++) {
                  ds[j] = IDList.get((int) (Math.random() * IDList.size()));
            }
            dictionary.put(testreihe[i], ds);
      }
}
```

Nach der erfolgreichen Festlegung der Testdaten, kann mit dem Performancemonitoring begonnen werden.

```
/**
 * Perfomancemonitoring MySQL
 */
Util dictionary = new Util(datenbank, username, "SQL");
dictionary.setupTest();
printer.println("Performancemonitoring MySQL wird begonnen " + format.for-
mat(Calendar.getInstance().getTime()));
for (Map.Entry<Integer, int[]> entry : dictionary.dictionary.entrySet()) {
      printer.print(entry.getKey());
      aktuelleTestreihe = entry.getKey();
      for (int l = 0; l < 10; l++) {
            int timeStart = (int) System.currentTimeMillis();
            for (int k = 0; k < entry.getValue().length; k++) {
                  ResultSet result = stmt.executeQuery("Select * From GEBAEUDE
                        WHERE ID = " + entry.getValue()[k]);
            }
            int timeEnd = (int) System.currentTimeMillis();
            int time = timeEnd - timeStart;
            printer.print(";");
            printer.print(time);

      }
      printer.println();
}
```

Abgesehen vom SELECT Statement ist der Ablauf für MySQL und MongoDB identisch. Das Statement zum Auslesen der Daten aus der MongoDB lautet:

```
Document findQuery = new Document("ID", new Document(
                                                "$eq",entry.getValue()[k]));
Document result = gebaeude.find(findQuery).first();
```

Den entstandenen Java-Quellcode exportiert man mit Hilfe der Entwicklungsumgebung eclipse als ausführbare jar-Datei. Das so entstandene Programm wird mit WinSCP vom lokalen Computer auf die virtuellen Maschinen kopiert. Das Java-Programm wird mit dem Befehl `java -jar PerformanceSQL.jar neusaess user1` in der Konsole gestartet. Wie in der Bachelorarbeit beschrieben gibt dabei der erste Parameter die Datenbank an, aus der die Daten gelesen werden sollen. Der zweite Parameter dient der Identifikation der Messdateien und ist frei wählbar.

Für das Multiuser Monitoring müssen mehrere Benutzer angelegt werden. Das Erstellen eines Benutzers erfolgt mit dem Befehl `adduser + username`. Anschließend wird in jedem Benutzerverzeichnis die Performancemonitoring.jar abgelegt. Um sicherzustellen, dass das Programm für jeden User zur gleichen Zeit begonnen wird, verwendet man das Kommando `at + Zeitangabe`. Dabei ist es wichtig, dass man sich vorher als root eingeloggt hat, da nur root ausreichende Rechte besitzt, um für jeden User das Programm starten zu können. `Atq` zeigt die Jobnummer und die Ausführungszeit aller laufenden Anweisungen. Wichtig hierbei ist, dass man nicht erkennen kann welche Kommandos der jeweilige Job ausführt. Abbildung 33 zeigt das gleichzeitige Starten des Performancemonitorings für drei Benutzer.

```
root@t3rd-ar-3:/home/areisenauer# at 15:30
warning: commands will be executed using /bin/sh
at> java -jar /home/user1/PerformanceSQL.jar neusaess user1<EOT>
job 25 at Mon Feb 25 15:30:00 2019
root@t3rd-ar-3:/home/areisenauer# at 15:30
warning: commands will be executed using /bin/sh
at> java -jar /home/user2/PerformanceSQL.jar neusaess user2<EOT>
job 26 at Mon Feb 25 15:30:00 2019
root@t3rd-ar-3:/home/areisenauer# at 15:30
warning: commands will be executed using /bin/sh
at> java -jar /home/user3/PerformanceSQL.jar neusaess user3<EOT>
job 27 at Mon Feb 25 15:30:00 2019
root@t3rd-ar-3:/home/areisenauer# atq
26      Mon Feb 25 15:30:00 2019 a root
27      Mon Feb 25 15:30:00 2019 a root
25      Mon Feb 25 15:30:00 2019 a root
root@t3rd-ar-3:/home/areisenauer#
```

Abbildung 33: Gleichzeitiges Starten des Performancemonitorings für drei Benutzer

Ein weiterer hilfreicher Befehl ist `ps aux | grep java`. Er zeigt alle laufenden Java-Prozesse an und ist in Abbildung 34 dargestellt.

```
areisenauer@t3rd-ar-3:~$ ps aux | grep java
root      3209  0.9  0.7 6757172 129784 ?      SNl  15:29   0:05 java -jar /home/user3/PerformanceSQL.jar neusaess user3
root      3210  0.9  0.7 6757172 128072 ?      SNl  15:29   0:05 java -jar /home/user2/PerformanceSQL.jar neusaess user2
root      3211  0.9  0.7 6757172 125768 ?      SNl  15:29   0:05 java -jar /home/user1/PerformanceSQL.jar neusaess user1
areisen+  3440  0.0  0.0  13136  1008 pts/2    S+   15:39   0:00 grep --color=auto java
```

Abbildung 34: Anzeige der aktuell laufenden Java-Prozesse

Will man eines der Java-Programme beenden, so kann man entweder den Job mit `atrm + job_id` löschen oder mit `kill + Prozess_id` den Javaprozess beenden.

4.2 Performancemonitoring der MySQL Datenbank

4.2.1 Single-User Zugriff

Das Performancemonitoring wird mit der kleinsten der drei Testdatenbanken begonnen. In der Grafik ist die durchschnittliche Antwortzeit pro Datensatz in Millisekunden dargestellt. Die durchschnittliche Antwortzeit wird aus einer Stichprobe von 10 Messungen ermittelt. In der Grafik ist deutlich zu erkennen, dass sich die durchschnittliche Antwortzeit pro Datensatz ab 750 ausgelesenen Datensätzen deutlich stabilisiert und ca. 13,7 ms beträgt. Bis zu diesem Stabilisierungszeitpunkt sind insgesamt 138703 ms, rund 2 Minuten und 19 Sekunden vergangen.

Eine mögliche Erklärung für dieses Verhalten könnte in der Anzahl der Datensätze liegen, die in der neusaess-Datenbank gespeichert sind. Die Datenbank neusaess enthält etwas mehr als 14.500 Datensätze. Ab 750 ausgelesen Datensätzen steigt möglicherweise die Wahrscheinlichkeit, dass ein Datensatz bereits gelesen wurde, sodass die Datenbank diesen schneller bereitstellen kann.

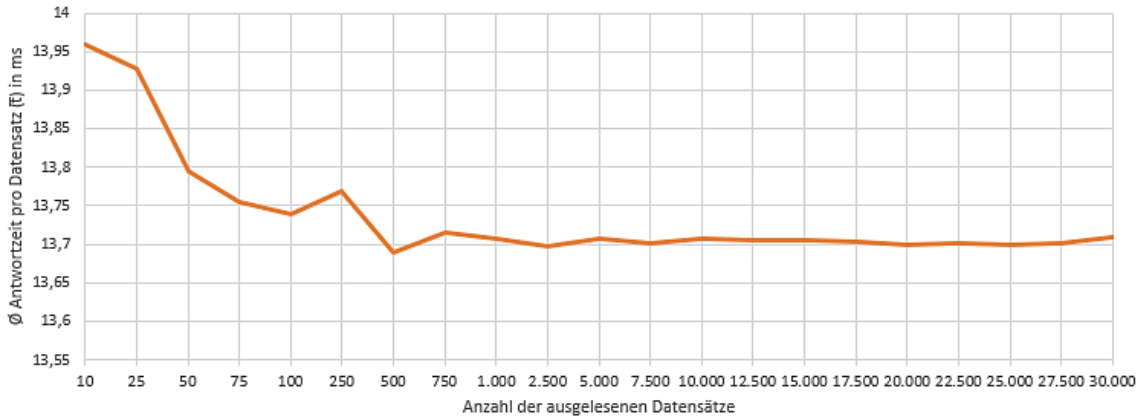


Abbildung 35: MySQL: Ø Antwortzeit pro Datensatz, Datenbank neusaess

Die zweite Analyse findet mit der Datenbank augsburg, die 90.773 Datensätze enthält, statt. Hier dauert die Antwortzeit zu Beginn verhältnismäßig lange. Zwischen 75 und 2500 Datensätzen, ist die Antwortzeit im Vergleich zum Gesamtdurchschnitt dagegen sehr niedrig. Auch bei der Datenbank augsburg lässt sich nach einiger Zeit eine Stabilisierung der Antwortzeit erkennen. Sie findet jedoch erst bei 2500 gelesenen Datensätzen und damit deutlich später als bei der Datenbank neusaess statt. Insgesamt beträgt die durchschnittliche Antwortzeit pro Datensatz 86,81 ms.

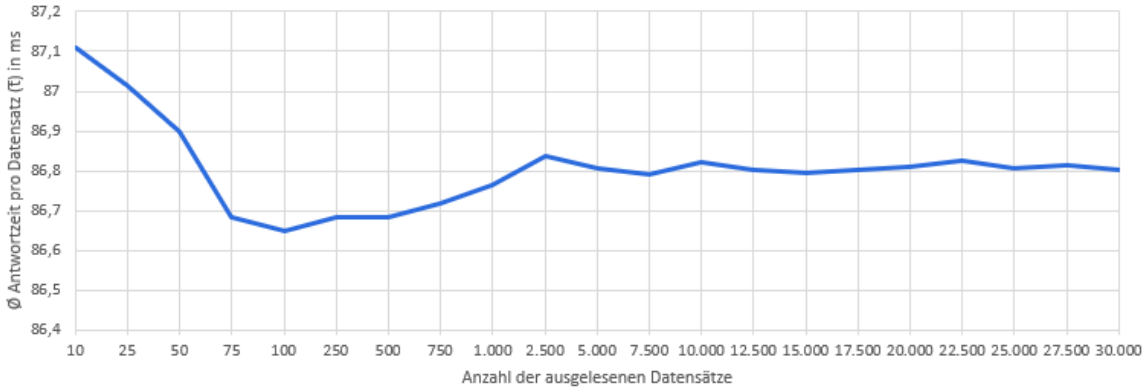


Abbildung 36: MySQL: Ø Antwortzeit pro Datensatz, Datenbank augsburg

Die Datenbank augsburg enthält 6,24 mal so viele Datensätze wie die Datenbank neusaess. Vergleicht man die beiden durchschnittlichen Antwortzeiten pro Datensatz der Datenbank augsburg mit der Datenbank neusaess, stellt man fest, dass auch die Antwortzeit fast genau um etwas mehr als den Faktor 6,24 steigt. Es ist also davon auszugehen, dass die Antwortzeit von der Menge der zu durchsuchenden Datensätze abhängig ist. Tabelle 2 zeigt die Antwortzeit in Bezug auf die Anzahl der zu durchsuchenden Datensätze

Datenbank	Ø Antwortzeit pro Datensatz	Anzahl der Datensätze	Antwortzeit/ Anzahl der Datensätze
neusaess	13,7377426 ms	14.536	0,00094508 ms/ ds
augsburg	86,80568679 ms	90.773	0,00095629 ms/ ds

Tabelle 2: Antwortzeiten der Datenbanken neusaess und augsburg

Die dritte und letzte Performanceanalyse findet mit der Datenbank bayern statt. Da diese Messung sehr zeitintensiv war, wurden lediglich bis zu 1000 Datensätze ausgelesen. Im Gegensatz zu den beiden vorherigen Messungen fällt auf, dass die Antwortzeit zwar sehr langsam, aber mit der steigenden Menge der zu lesenden Datensätze kontinuierlich steigt. Das Auslesen eines Datensatzes aus der Datenbank dauert fast 11 Sekunden.

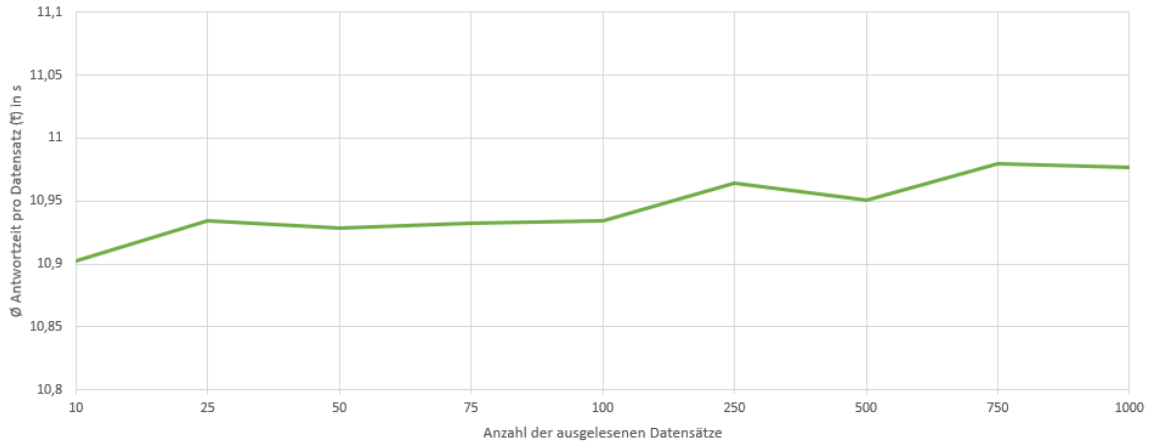


Abbildung 37: MySQL: Ø Antwortzeit pro Datensatz, Datenbank bayern

Setzt man auch bei der Datenbank bayern die Antwortzeit in Verhältnis zur Datenmenge in der Datenbank (siehe Tabelle 3) kann man erkennen, dass sich die Antwortzeit ebenfalls leicht erhöht hat. Es sollten weitere Messungen mit mittelgroßen Datenbanken erfolgen um feststellen zu können, ob die Antwortzeit eventuell linear mit der Größe der Datenbank wächst.

Datenbank	Ø Antwortzeit pro Datensatz	Anzahl der Datensätze	Antwortzeit/ Anzahl der Datensätze
bayern	10944,7679 ms	9.127.738	0,00119907 ms/ ds

Tabelle 3: Antwortzeiten der Datenbank bayern

4.2.2 Multi-User Zugriff

Die Ergebnisse des Multi-User Zugriffs auf die Datenbank neusaess mit 10 Benutzern sind besonders spannend. Bis zum Lesen von bis einschließlich 750 Datensätzen haben die Benutzer sehr verschiedene Antwortzeiten pro Datensatz. Ab 750 gelesenen Datensätzen stabilisiert sich die Antwortzeit und beträgt etwa 36 ms. Vom Beginn der Performancemessung bis zur Stabilisierung der Antwortzeit vergehen mehr als 10 Minuten. Offensichtlich benötigt die Datenbank so lange um zu erkennen viele Benutzer derzeit auf die Datenbank zugreifen und jedem Benutzer die gleichen Ressourcen bei der Datenbankabfrage zur Verfügung zu stellen. Interessant ist auch, dass bei einer Verzehnfachung der Benutzer die Antwortzeit nur um den Faktor 2,8 steigt.

Ein besonderes Phänomen ist der rasante Abfall der Antwortzeit ab 27.500 gelesenen Datensätzen. In Abbildung 38 sind die Messkurven des Multiuserzugriffs dargestellt

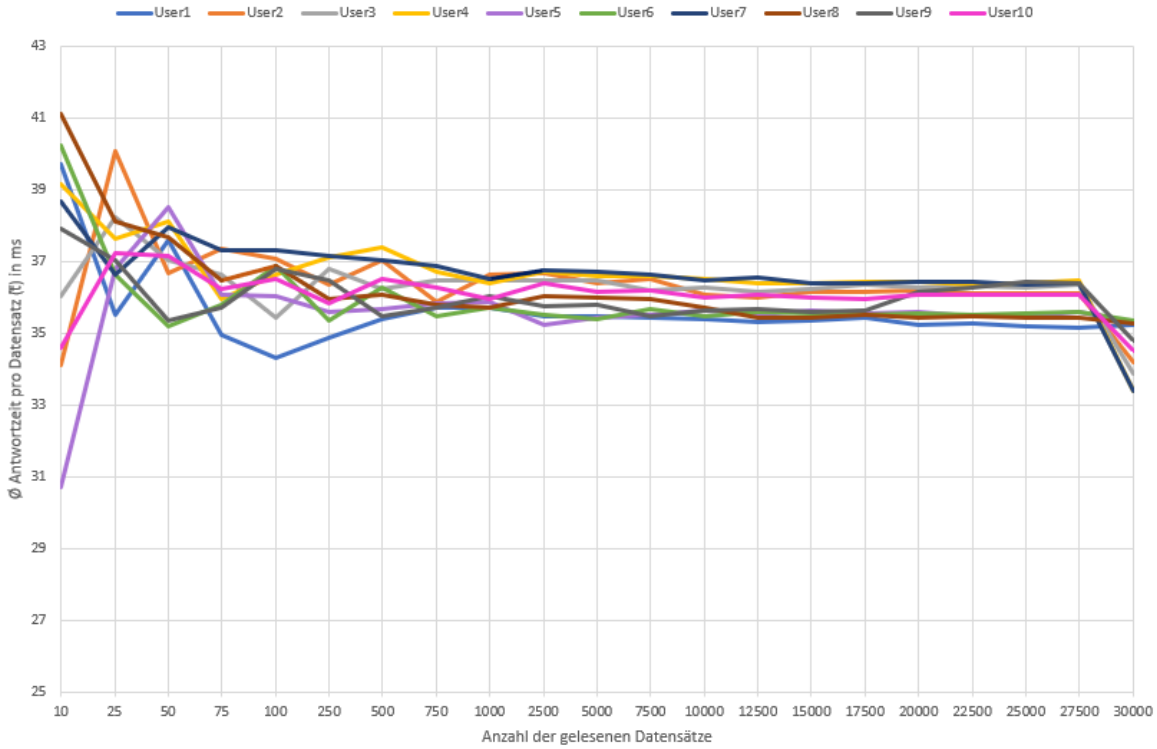


Abbildung 38: MySQL: Ø Antwortzeit pro Datensatz, 10 User

4.3 Performancemonitoring der MongoDB

4.3.1 Single-User Zugriff

Ebenso wie bei der MySQL-Datenbank wird auch bei MongoDB mit der kleinsten der drei Testdatenbanken begonnen. In der Grafik ist die durchschnittliche Antwortzeit pro Datensatz in Millisekunden dargestellt. Die durchschnittliche Antwortzeit wird aus einer Stichprobe von 10 Messungen ermittelt. In der Grafik ist deutlich zu erkennen, dass sich auch bei der MongoDB die Antwortzeit irgendwann stabilisiert. Jedoch erfolgt diese Stabilisierung erst bei 5000 gelesenen Datensätzen. Insgesamt lässt sich feststellen, dass die MongoDB mit einer durchschnittlichen Antwortzeit von 3,2 ms pro Datensatz mehr als 4 mal schneller ist als die MySQL-Datenbank.

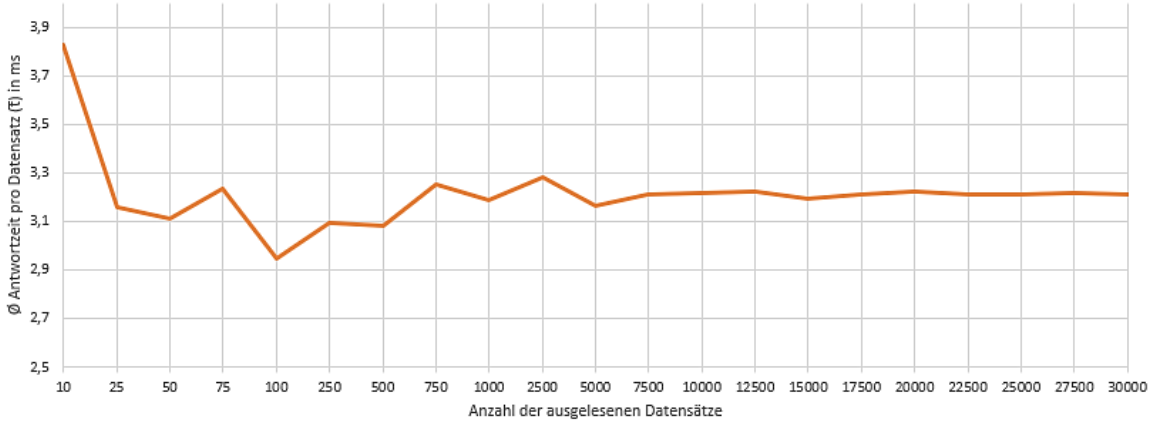


Abbildung 39: MongoDB: Ø Antwortzeit pro Datensatz, Datenbank neusaess

Bei der zweiten Analyse ist die Antwortzeit insgesamt recht stabil. Es gibt jedoch eine extreme Abweichung bei 50 Datensätzen. Möglicherweise ist zu diesem Zeitpunkt ein weiteres Programm, das auf eine andere Datenbank zugreift, gestartet worden. Insgesamt beträgt die durchschnittliche Antwortzeit pro Datensatz 20,64 ms.

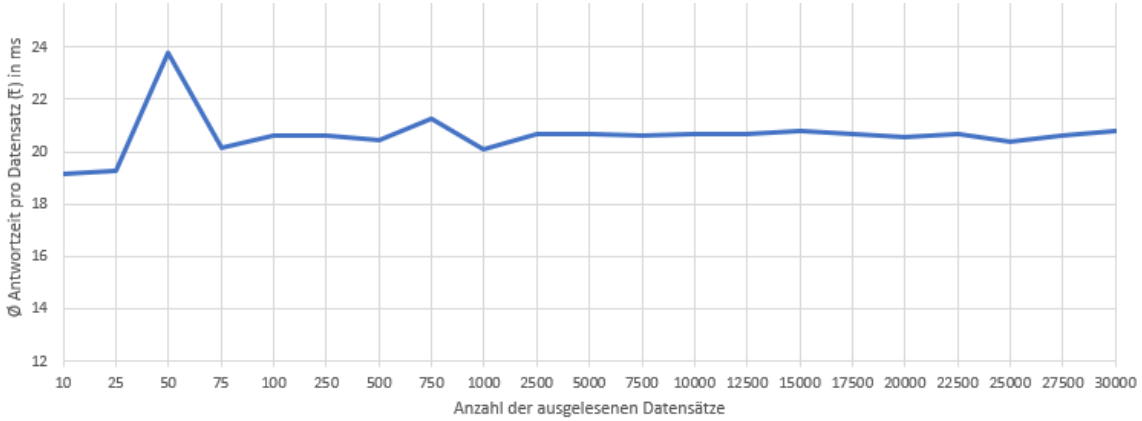


Abbildung 40: MongoDB: Ø Antwortzeit pro Datensatz, Datenbank augsburg

Mit der Standardabweichung kann man ermitteln, wie stark die Streuung der Werte um einen Mittelwert ist. Um die Standardabweichung berechnen zu können, muss man erst den Durchschnitt bilden und im Anschluss daran die Varianz berechnen. Die Standardabweichung ist die Wurzel aus der Varianz. In Excel gibt es dankenswerterweise die Funktion STABW.N(), die die Berechnung der Standardabweichung durchführt. In Abbildung 41 ist die Standardabweichung der jeweiligen Messreihe von der insgesamt durchschnittlichen Antwortzeit (20,64 ms) dargestellt. Auch hier ist die Abweichung der Antwortzeit beim Auslesen von 50 Datzensätzen sehr deutlich zu erkennen. Darüber hinaus lässt sich erkennen, dass mit der wachsenden Anzahl an ausgelesenen Datensätzen die Standardabweichung sinkt.

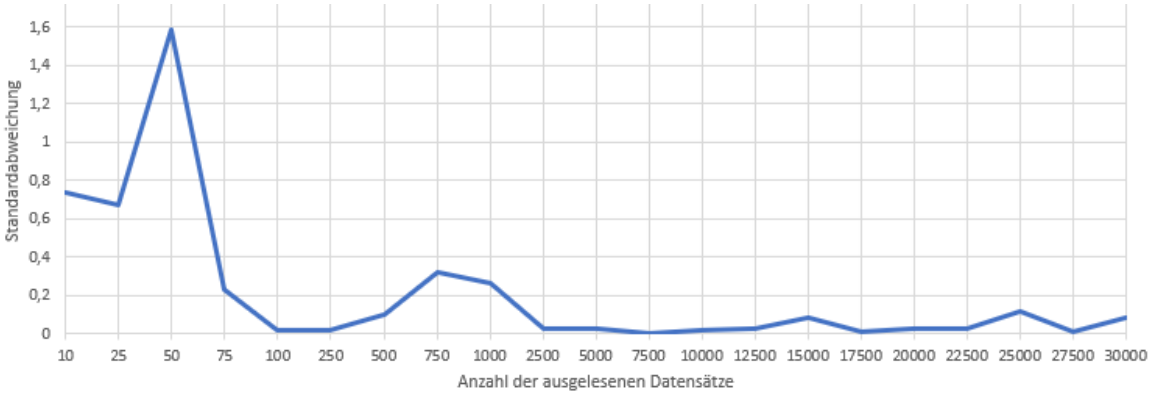


Abbildung 41: MongoDB Standardabweichung, Datenbank augsburg

Am deutlichsten ist der Unterschied zwischen MySQL und MongoDB bei der Datenbank bayern zu erkennen. Hier benötigt die MongoDB durchschnittlich nur 2,39 Sekunden um einen Datensatz auszulesen. Ebenso wie bei allen anderen Messungen ist zu Beginn der Performanceanalyse eine Unregelmäßigkeit in der Antwortzeit zu erkennen.

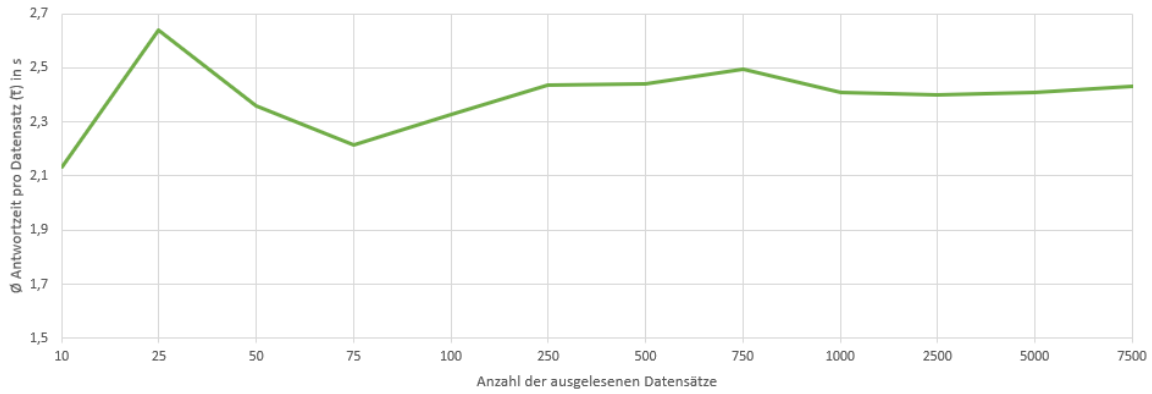


Abbildung 42: MongoDB: Ø Antwortzeit pro Datensatz, Datenbank bayern

Auch bei der MongoDB wird die durchschnittliche Antwortzeit im Verhältnis zu den ausgelesenen Datensätzen betrachtet. Mit der steigenden Datenmenge in der Datenbank erhöht sich die Antwortzeit.

Datenbank	Ø Antwortzeit pro Datensatz	Anzahl der Datensätze	Antwortzeit/ Anzahl der Datensätze
neusaess	3,21344467 ms	14.536	0,00022107
augsburg	20,64070187 ms	90.773	0,00022739
bayern	2391,74512 ms	9.127.738	0,00026203

Tabelle 4: Antwortzeiten der Datenbanken neusaess, augsburg und bayern

4.3.2 Multi-User Zugriff

Besonders interessant ist der Vergleich des Multi-User Zugriffs zwischen MongoDB und MySQL. Beide Datenbanken zeigen ab dem Lesen von bis einschließlich 750 Datensätzen eine Stabilisierung der Antwortzeit. Da die MongoDB jedoch insgesamt deutlich weniger Zeit benötigt um einen Datensatz aus der Datenbank auszulesen, findet die Stabilisierung der Antwortzeit bei der MongoDB bereits nach etwa 3 Minuten und 15 Sekunden statt. Interessant ist auch, dass bei einer Verzehnfachung der Benutzer die Antwortzeit bei der MongoDB um den Faktor 3,375 steigt. Bei der MySQL-Datenbank steigt die Antwortzeit dagegen nur um den Faktor 2,8. Ein unerklärliches Phänomen ist der rasante Abfall der Antwortzeit ab 27.500 gelesenen Datensätzen, der sowohl bei MongoDB als auch bei

MySQL sehr deutlich zu erkennen ist. In Abbildung 43 sind die Messkurven des Multiuser-zugriffs dargestellt.

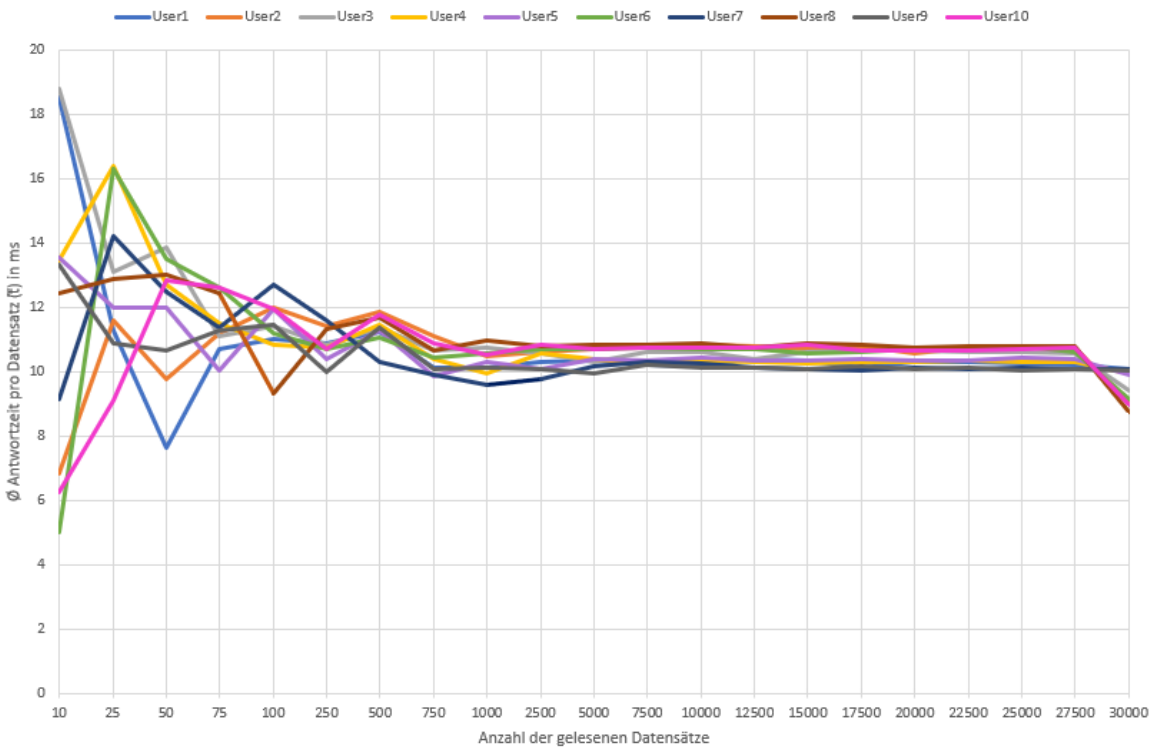


Abbildung 43: MongoDB: Ø Antwortzeit pro Datensatz, 10 User

5 Vergleich und Interpretation der Ergebnisse des Performancemonitorings

Vergleicht man die Ergebnisse der jeweiligen Messungen erkennt man deutlich den Zeitunterschied zwischen MongoDB und MySQL. Es ist nicht verwunderlich, dass die MongoDB deutlich niedrigere Antwortzeiten erzeugt, da das Speicherkonzept der MongoDB für einen schnellen Datenzugriff optimiert ist.

Einige Phänomene treten sowohl bei der MongoDB als auch bei der MySQL-Datenbank auf. Beide Datenbanken benötigen einige Zeit bis sich die Antwortzeit stabilisiert hat. Interessanterweise findet die Stabilisierung der Antwortzeit bei beiden Datenbankarten nach der gleichen Menge an gelesenen Datensätzen statt. Darüber hinaus lässt sich beim Multi-User Zugriff bei beiden Datenbanken ein Abfall der Antwortzeit ab 27.500 gelesenen Datensätzen erkennen. Dieser Effekt sollte mit weiteren Datenbanken untersucht werden.

6 Zusammenfassung und Ausblick

In der vorliegenden Praxisarbeit wurde zunächst das Erstellen eines Ubuntu 18.04 Servers gezeigt. Anschließen wurden auf dem einen Server die Datenbank MySQL mit der zugehörigen Weboberfläche phpMyAdmin installiert. Auf dem zweiten Server fand die Installation einer MongoDB statt. Die grafische Oberfläche Robo3T wurde auf dem lokalen Rechner installiert und erleichtert die Arbeit deutlich.

Der Import der Geodaten, die in Form einer csv-Datei vorlagen, wurden mittels `LOAD DATA` in die MySQL-Datenbank realisiert. Der Datenimport in die MongoDB wurde mit `mongoimport()` durchgeführt.

Das Performancemonitoring zeigt eine deutlich höhere Geschwindigkeit beim Auslesen der Daten aus der MongoDB.

Für das LDBV sollte deswegen ein Technologiewechsel nicht ausgeschlossen werden. Vorab ist jedoch zu klären, ob alle im LDBV verwendeten Softwareprodukte mit MongoDB kompatibel sind. Besonders berücksichtigt sollten dabei Anwendungen werden, die auf dem relationalen System der Oracle-Datenbank aufbauen. Darüber hinaus sollte das Performancemontitoring für die im LDBV eingesetzte Oracle-Datenbank vor Ort wiederholt werden um aussagekräftige Messungen unter realen Bedingungen zu schaffen. Für erste Tests im Produktivsystem sollten zunächst die beiden Datenbanksysteme Oracle und MongoDB parallel betrieben werden. Erweist sich MongoDB nach der ersten Testphase von etwa einem halben Jahr als geeignete Alternative und steht das notwendige Personal zur Verfügung ist im Sinne des technologischen Fortschritts eine Umstellung des Datenbanksystems anzustreben.

7 Literaturverzeichnis

7.1 Buch (Monographie)

The phpMyAdmin devel team (2018): phpMyAdmin Documentation.

Ullenboom, Christian (2018): Java ist auch eine Insel. Einführung, Ausbildung, Praxis. 13., aktualisierte und überarbeitete Auflage (Rheinwerk Computing).

Wolf, Jürgen (2019): C von A bis Z. Das umfassende Handbuch. 3., aktualisierte und erweiterte Auflage 2009, 7. korrigierter Nachdruck 2019. Bonn: Rheinwerk Verlag.

7.2 Online-Quellen

Alex Howansky (2016): Stack Overflow. Import CSV with semicolon separator into MongoDB database. Online verfügbar unter https://stackoverflow.com/questions/39044556/import-csv-with-semicolon-separator-into-mongodb-database, zuletzt aktualisiert am 19.08.2016, zuletzt geprüft am 06.03.2019.

Braun Heider GmbH: Shellbefehle - chown, zuletzt geprüft am 30.12.2018.

Can’t log into phpMyAdmin: mysqli_real_connect: (HY000/1698): Access denied for user ‘root’@’localhost’ (2018). Online verfügbar unter https://devanswers.co/phpmyadmin-access-denied-for-user-root-localhost/, zuletzt aktualisiert am 21.08.2018, zuletzt geprüft am 03.01.2019.

Datendurchsatz (2018). Online verfügbar unter https://de.wikipedia.org/wiki/Datendurchsatz, zuletzt aktualisiert am 06.07.2018, zuletzt geprüft am 15.01.2019.

F. Kalhammer (Hg.) (2001): Wichtige Unix-Befehle. Befehle zum Umgang mit Dateien. Online verfügbar unter http://www.linux-praxis.de/linux1/befehle2.html, zuletzt aktualisiert am 2001, zuletzt geprüft am 03.01.2019.

MongoDB Manual. Data Modeling Introduction (2019a). Online verfügbar unter https://docs.mongodb.com/manual/core/data-modeling-introduction/, zuletzt aktualisiert am 28.02.2019, zuletzt geprüft am 06.03.2019.

MongoDB Manual. mongoimport (2019b). Online verfügbar unter https://docs.mongodb.com/manual/reference/program/mongoimport/, zuletzt aktualisiert am 28.02.2019, zuletzt geprüft am 01.03.2019.

MySQL 5.7 Reference Manual. Geometry Format Conversion Functions. Online verfügbar unter https://dev.mysql.com/doc/refman/5.7/en/gis-format-conversion-functions.html#function_st-astext, zuletzt geprüft am 12.03.2019.

Oracle (Hg.): MySQL 8.0 Reference Manual. LOAD DATA Syntax. Online verfügbar unter https://dev.mysql.com/doc/refman/8.0/en/load-data.html, zuletzt geprüft am 25.01.2019.

8 Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde nach meiner besten Kenntnis bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Hof, den 13.03.2019 _____
(Andrea Reisenauer)