FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Einführung in Datenbanken
Installation und Vorbereitung

Prof. Dr. rer. nat. Alexander Voß
Inform-Professur

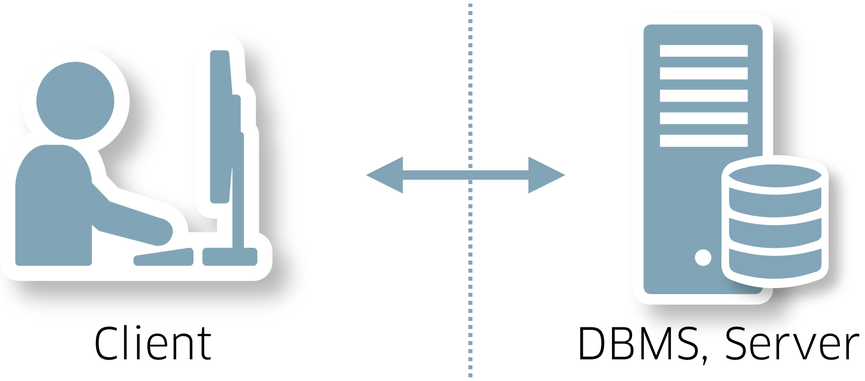
FH Aachen
University of applied Sciences

23.09.2021

Client-Server-Architektur

Für die praktischen Übungen benötigen wir ein sogenanntes Datenbankmanagement-system (DBMS, Server), welches die Daten für unsere Anwendungen (Clients) bereitstellt. Wir haben hier also eine klassische Client-Server-Architektur:

PersNr	Name	Fachgebiet	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2134



Daten Client DBMS, Server

In der Praxis laufen Clients häufig lokal und Server sind über das Netz erreichbar.

Für unsere Übungszwecke ist es nun sinnvoll, eine vergleichbare Situation möglichst einfach und ohne Seiteneffekte herzustellen…

Installation

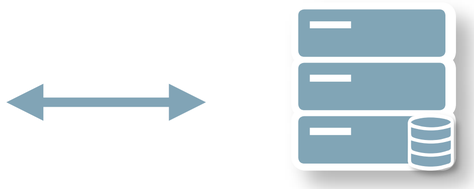
Virtualisierung

… weswegen wir im Folgenden beispielhaft ein DBMS **lokal** und als **Docker-Container** installieren. Auf diese Weise können wir Seiteneffekte mit evtl. bereits installierten DBMS minimieren und sind nicht auf die Verfügbarkeit externer Server angewiesen.

lokale Installation

PersNr	Name	Fachgebiet	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2134

Client



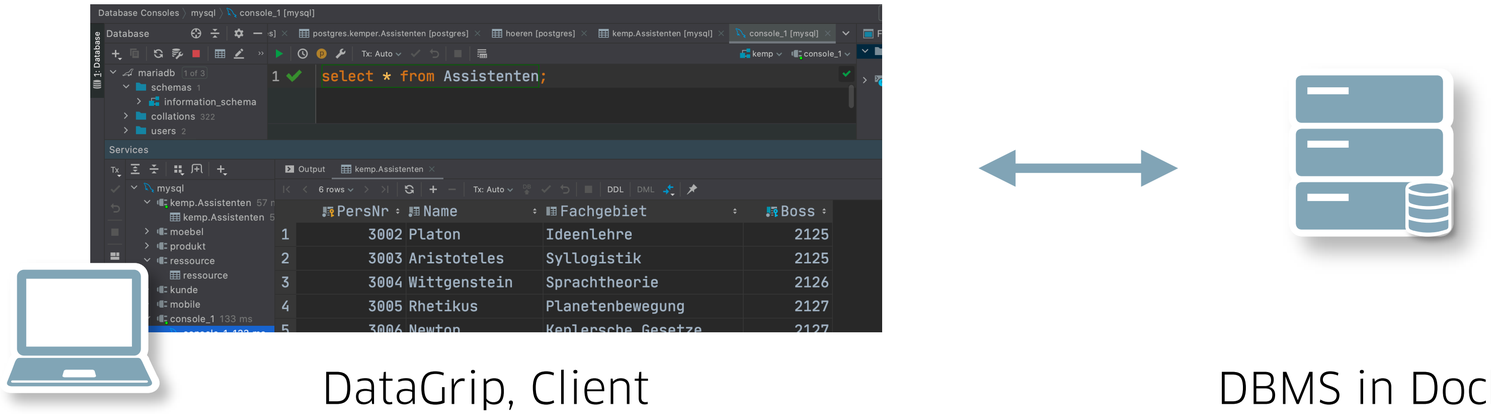
Docker-Container
mit DBMS

Beachten Sie, dass diese Konfiguration keineswegs notwendig ist. Sollten Sie etwa bereits ein DBMS zur Verfügung haben, welches Sie nutzen (und modifizieren!) dürfen, dann ist das auch in Ordnung. Eine Übersicht möglicher Kombinationen findet sich am Ende.

Installation

Client

Später werden wir das DBMS auch programmatisch (Java, Kotlin) ansprechen, aber zunächst nutzen wir einen Client, der speziell auf das Management verschiedener DBMS ausgelegt ist: **DataGrip** (https://www.jetbrains.com/datagrip/). Durch die "educational license for students" können Sie DataGrip zu Lernzwecken ohne weitere Kosten nutzen.



DataGrip, Client

DBMS in Docker

Beachten Sie auch hier, dass andere Tools (MySQL Workbench, pgAdmin, ...) grundsätzlich möglich sind. Sie müssen am Ende zum DBMS und zu den Übungen passen.

Installation

DBMS, Server

Wir betrachten zunächst sogenannte "relationale" DBMS (später auch weitere Typen) und hier beispielhaft **MySQL** (https://www.mysql.com/) bzw. **MariaDB** (https://mariadb.com/). Andere verbreitete DBMS sind PostgreSQL, MSSQL, Oracle oder Db2.

Für die Übungen und die Skripte (Datenabfragedialekt SQL) müssen wir uns einmal festlegen, denn wir können nicht alle Varianten pflegen und unterstützen.

Die Wahl von MySQL oder MariaDB für unsere Zwecke ist nicht wirklich kritisch. Es gibt viele Wege und Tools, um Daten in andere DBMS zu konvertieren. Sie müssen ggf. die Beispieldaten und -skripte anpassen, aber das ist mit vertretbarem Aufwand möglich.

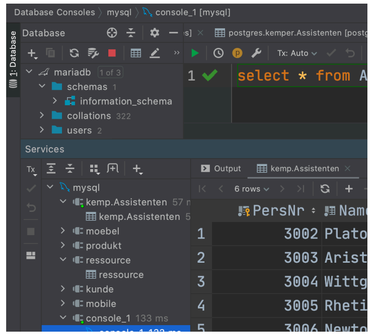
Beide DBMS können Sie lokal installieren, aber beachten Sie bitte, dass dies kein kleiner Eingriff auf Ihrem Rechner ist und ggf. mit bestehenden Installationen in Konflikt gerät. Daher empfehlen wir eine Virtualisierung.

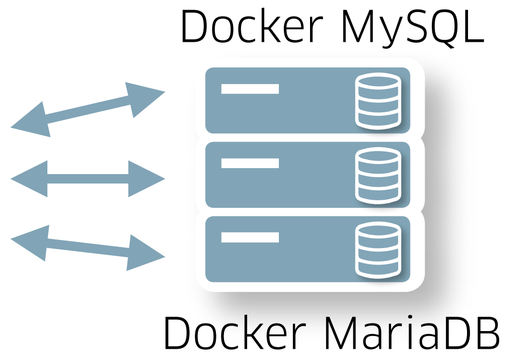
Natürlich gilt: Ist MySQL oder MariaDB schon installiert, können Sie das auch nutzen.

DBMS, Server – Virtualisierung mit Docker

Was macht Docker genau und speziell ein Docker-Container? Docker stellt, kurz gesagt, eine fertig eingerichtete Anwendung, hier unser DBMS, als abgekapselten *Container* bereit und läßt diesen, im Wesentlichen getrennt von bereits installierten Anwendungen, laufen. Die Vorteile liegen dabei auf der Hand:

- bestehende Installationen bleiben unangetastet;
- unterschiedliche Versionen eines DBMS und auch verschiedene DBMS können, sogar gleichzeitig, verwendet werden;
- ein Neustart des DBMS und eine Neuinstallation der Daten sind, falls notwendig, einfach möglich.





In unserem Fall stellen wir also einen Docker-Container mit einem DBMS bereit, starten diesen und können uns dann mit dem Client (DataGrip) dorthin verbinden.

DBMS, Server – Virtualisierung mit Docker und Docker-Compose – Teil I

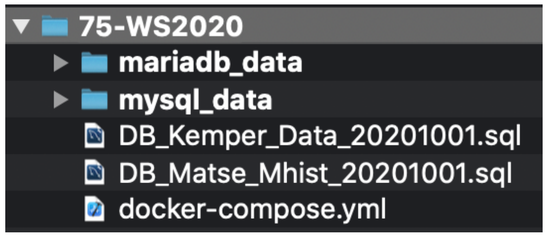
Was macht Docker Compose genau? "Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration."

Wir stellen also ein `docker-compose.yml` mit Einträgen für MySQL und MariaDB bereit und im Fall einer korrekten Docker-Installation war es das schon. Alles weitere, d.h. Download der DBMS-Container und starten der Services, wird von Compose erledigt.

Eine Information ist noch für das Verständnis wichtig: Da ein Docker-Container immer im gleichen Ausgangszustand startet (Feature!), überleben zunächst einmal auch keine Daten, die verändert wurden – was für ein DBMS nicht so sexy erscheint. Daher gibt man dem Container in der Regel spezielle Ablageorte für Daten mit, also konkret "geteilte" Ordner, wo dann die Daten gesichert werden können (siehe "volumes").

DBMS, Server – Virtualisierung mit Docker und Docker-Compose – Teil II

- Installiert: DataGrip, Docker inkl. Docker Compose
- Verzeichnis: "geteilte" Ordner `mariadb_data` und `mysql_data` sowie `docker-compose.yml`



- `docker-compose.yml`:
  - Passwort: Klartext nur zum Test!
  - `ports`: Port-Mapping von 3308 bzw. 3310 auf 3306
  - `volumes`: "geteilter" Ordner für die Daten, zeigt auf `/var/lib/mysql`

```yaml
services:
  mysql:
    image: mysql:latest
    environment:
      MYSQL_ROOT_PASSWORD: root
    ports:
      - "3308:3306"
    volumes:
      - ./mysql_data:/var/lib/mysql
    restart: always
```

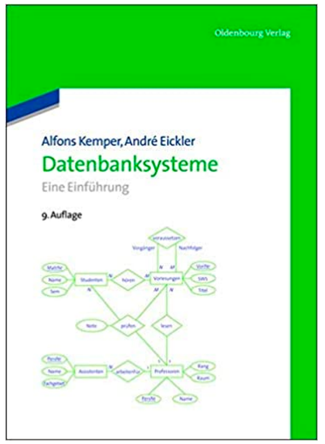
```yaml
mariadb:
  image: mariadb:latest
  environment:
    MYSQL_ROOT_PASSWORD: root
  ports:
    - "3310:3306"
  volumes:
    - ./mariadb_data:/var/lib/mysql
  restart: always
```

- Befehle zum Starten, Stoppen und eine Übersicht aller laufenden Container finden sich im Anhang.

Daten

Übungsdaten für das DBMS

Das Standardwerk zu Datenbanken von Kemper und Eickler wird häufig zitiert und ihm entstammen zahlreiche Beispiele, u.a. auch Daten rund um einen fiktiven Uni-Betrieb. Modelliert sind u.a. Professoren, Assistenten, Prüfungen usw. und wir stellen diese "Kemper"-Datenbank als Import bereit.



PersNr	Name	Rang	Standort	Raum
2125	Sokrates	C4	Jülich	226
2126	Russel	C4	Jülich	232
2127	Kopernikus	C3	Aachen	310
2133	Popper	C3	Aachen	52
2134	Augustinus	C3	Aachen	309

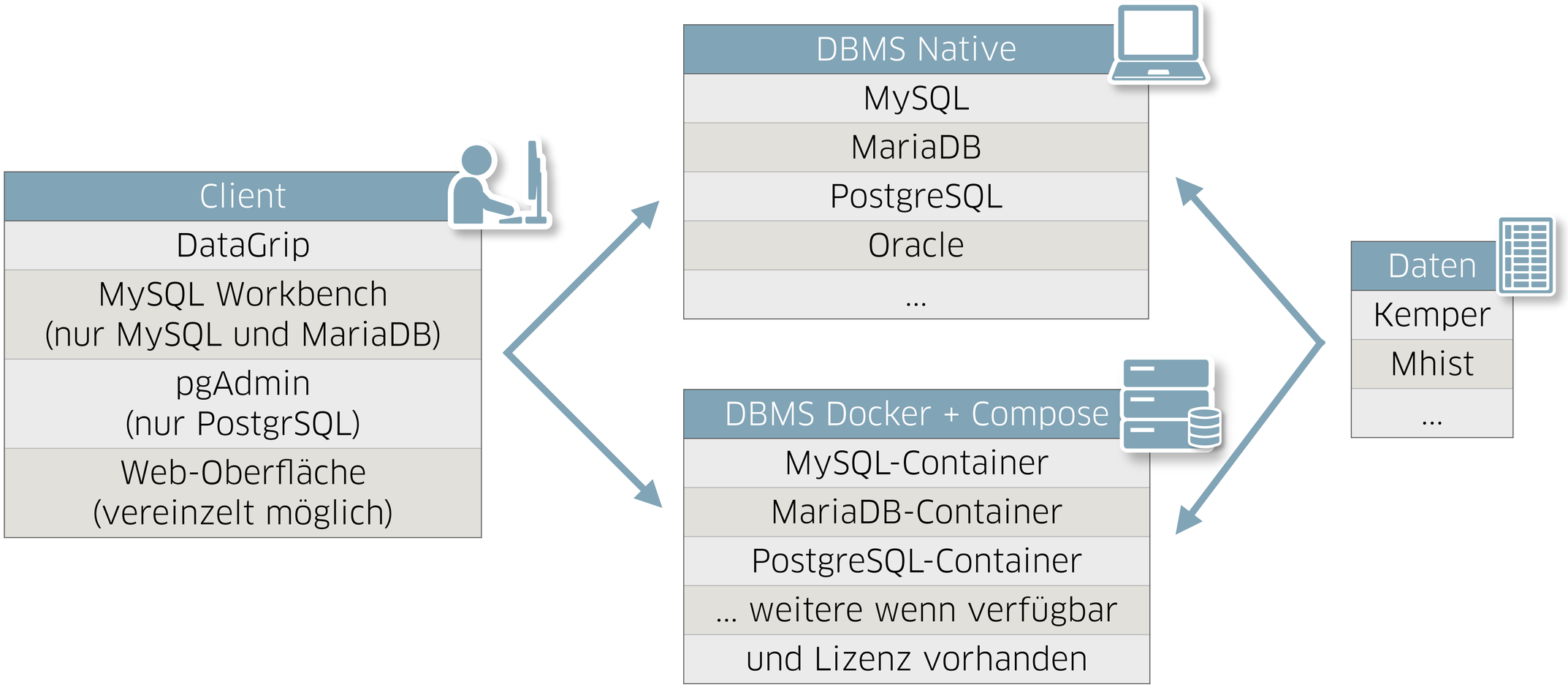
Professoren

PersNr	Name	Fachgebiet	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127

Assistenten

Darüber hinaus gibt es weitere Übungsdaten, z.B. eine fiktive "Mhist"-Unternehmens-datenbank, die Sie ebenfalls importieren können.

Konfigurationen - Übersicht



mind. ein Client benötigt

mind. ein DBMS benötigt

alle Daten benötigt

Finaler Benchmark

Ein Dank an Inform GmbH für die Bereitstellung der Piktogramme.

Ziel der Vorbereitung

Das haben Sie erreicht, wenn Sie

- mit einem DBMS-Client (z.B. DataGrip) auf
- das laufende DBMS (z.B. MySQL als Docker-Container) zugreifen können und
- eine einfache SQL-Abfrage wie
  "`select * from Professoren;`"
  als Ergebnis das nebenstehende Resultat ergibt.

PersNr	Name	Rang	Standort
2125	Sokrates	C4	Jülich
2126	Russel	C4	Jülich
2127	Kopernikus	C3	Aachen
2133	Popper	C3	Aachen
2134	Augustinus	C3	Aachen

Viel Erfolg!

Die Installation können auch Sie im Video nachvollziehen.

Docker-Compose Kommandobeispiele

Im Hintergrund die Services aus docker-compose.yml starten:

```
> docker-compose up -d
> docker compose up -d
```

```
Creating network "75-ws2020_default" with the default driver
Creating 75-ws2020_mysql_1   ... done
Creating 75-ws2020_mariadb_1 ... done
```

Shutdown der Services:

```
> docker-compose down
> docker compose down
```

```
Stopping 75-ws2020_mariadb_1 ... done
Stopping 75-ws2020_mysql_1   ... done
Removing 75-ws2020_mariadb_1 ... done
Removing 75-ws2020_mysql_1   ... done
```

Bei neuerem `Docker` ist `Compose` Teil des Interfaces, dann ist der zweite Befehl (ohne Bindestrich) direkt über Docker auch möglich.

Docker-Compose Kommandobeispiele

Übersicht der laufenden Prozesse:

```
> docker-compose ps
> docker compose ps
```

```
        Name                       Command               State                 Ports
-------------------------------------------------------------------------------------
75-ws2020_mariadb_1   docker-entrypoint.sh mysqld   Up      0.0.0.0:3310->3306
75-ws2020_mysql_1     docker-entrypoint.sh mysqld   Up      0.0.0.0:3308->3306
```

Eine Shell ("bash") im Docker-Container (Name "75-ws2020_mariadb_1") starten:

```
> docker exec -it 75-ws2020_mariadb_1 bash
```

Von hier aus können weitere Befehle, z.B. aus den SQL-Tools, abgesetzt werden.

Im- und Export Kommandobeispiele

Daten können oftmals per Menübefehl aus einem (GUI)Client im- oder exportiert werden, oder aber per Kommando aus einer Shell heraus, z.B. wie vorher gesehen.

Für letzteres nutzen wir die MySQL-Tools `mysql` und `mysqldump` und gehen davon aus, dass in der Datenbank ein sog. Schema `kemper` (die Datenbank) existiert und wir als User `root` (-u) mit Passwort `root` (-p) in einem Docker-Container agieren und `/var/lib/mysql` der "geteilte" Ordner ist.

Datenbank in eine Datei exportieren:

```
> mysqldump -uroot -proot kemper > /var/lib/mysql/dump3.sql
```

Datenbank aus einer Datei importieren:

```
> mysql -uroot -proot --database=kemper < /var/lib/mysql/dump3.sql
```

Für Interessierte: `dump3.sql` enthält SQL-Befehle, um die Datenbank aufzubauen.