Milestone 3:
Migration Relational-NoSQL

Team:
Volobuieva Karyna, a01568897
Lanyi Lujza, a01634518

1. NoSQL database design

Wir haben beschlossen, Kollektionen für alle Tabellen zu erstellen. Jedes Feld entspricht der Spalte in MySQL. Die Beziehungen sind also eher normalisiert. Dies liegt daran, wie über die PHP-Dateien auf die Daten zugegriffen wird.

1.1 Die Kochschule

```
{
    "_id": ObjectID("5e301acac5f335538041dc76"),
    "AbteilungsNr": 1,
    "Name": "Gusto",
    "Ort": "Wien",
    "PLZ": "1020",
    "Strasse": "Taborstrasse 1-3"
}
```

Implementiert:

In Kochschule werden nur die Attributen von Kochschule gespeichert.

Alternative:

Wir können die Küche als Array values hier speichern, weil die Küche ist eine schwache Entität. Aber die Küche werden sehr oft geändert und dann die Zeit für die Abfrage wird sich verlangsamen.

1.2 Die Findet_statt

Implementiert:

In MySQL hatten wir eine Tabelle Findet_statt, um diese Tabelle in MongoDB zu migrieren haben wir entscheidet, die Zeit aus dieser Tabelle in andere Kollektionen zu speichern, anstatt unabhängige Kollektion zu implementieren.

Alternative:

Wir können für die Findet_statt Tabelle eine unabhängige Kollektion erstellen und drinnen die Zeit entweder als die eingebetteten Dokumenten oder als Referenz auf Zeit Dokument speichern. Die Küche und der Kochkurs werden als eingebettete Dokumenten gespeichert. Diese Kollektion wird sehr oft verwendet, weil die eine von Hauptfunktionen von System ist. Mehrere Daten wurden dort gespeichert und die Zeit für Abfragen wird langsam. Also wir haben entscheidet die bessere Alternative: in zwei verschiedene Kollektionen entkoppeln.

1.3 Die Küche


```
{
    "_id": ObjectID("5e301acbc5f335538041dc77"),
    "AbteilungsNr": ObjectID("5e301acac5f335538041dc76"),
    "Nummer": 1,
    "Fassungsvermoegen": 15,
    "Ausstattung": "2x Backoefen,2x Herde, 2x Kuehlschraenke und ein Geschirrspueler",
    "Zeit": [
        {
            "_id": ObjectID("5e301ad0c5f335538041f7f6"),
            "ZeitBlock": "1. 10:00-14:00",
            "Datum": "2020-01-10"
        },
        {
            "_id": ObjectID("5e301ad0c5f335538041f900"),
            "ZeitBlock": "2. 14:15-18:15",
            "Datum": "2020-01-17"
        }
    ]
}
```


Implementiert:

In Küche werden die Attributen von Küche und die AbteilungsNr wird als Referenz auf Kochschule gespeichert. Auch hier wird die Zeit als Array values gespeichert, weil wir in unsure ER-Diagramm die ternäre Beziehung zwischen Küche, Zeit und Kochkurs haben. Die Küche und der Kochkurs haben beide Kardinalität 1 und die Zeit Kardinalität m. Unserer Meinung nach, die Zeit gehört immer zu einer Küche und es wird logisch die Zeit Array hier speichern.

Alternative:

Wie oben geschrieben,, können wir die Küche als Array in Kochschule Kollektion speichern. Da jedoch im Laufe der Zeit neue Termine für die Durchführung von Kursen auftauchen, nimmt die Größe eines Attributs (d. h. eines Arrays) ständig zu und es wird schwierig, auf Daten zuzugreifen. Auch andere Küchenattribute können häufig geändert werden.

1.4 Der Mitarbeiter


```
{
    "_id": ObjectID("5e301acbc5f335538041dce4"),
    "MId": 1,
    "Nachname": "Koch",
    "Vorname": "Florence",
    "Gehalt": 1200.6,
    "Strasse": "Haasgasse",
    "Ort": "Wien",
    "PLZ": 1020,
    "Geburtsdatum": "1983-04-23",
    "LeiterMId": 1,
    "AbteilungsNr": ObjectID("5e301acac5f335538041dc76")
}
```


Implementiert:

In Mitarbeiter Kollektion werden sowohl die Attributen von Mitarbeiter als auch die Referenz in AbteilungsNr auf Kochschule Dokument gespeichert.

Alternative:

Wir können die Kochschule hier als ein eingebettetes Dokument zu speichern. Unseres System ist für eine Kochschule geplant, Referenz kann helfen die Speicherplatz zu speichern.

1.5 Der Manager

Implementiert:

In Manager Kollektion werden die Attributen von Manager und die Referenz in MId auf Mitarbeiter Dokument gespeichert. Wir haben solches Modell gewählt, weil das der Zugriff auf der Webseite vereinfacht.

Alternative:

Wir können der Manager als eingebettetes Dokument in Mitarbeiter Dokumentation speichern, weil das “is-a” Beziehung ist.

1.6 Der Führt

Implementiert:

In MySQL hatten wir eine Tabelle Fuehrt. Das repräsentiert Many-to-Many Beziehung. Also das ist logisch die Referenzen verwenden, um die Daten zu speichern. Wir haben kurze Abfragezeiten erstellt. Wir speichern eine Array von ObjectIds aus der Koch Kollektion in der Kochkurs Kollektion, um festzustellen, welcher Koch wird dieser Kochkurs führen. Ebenso speichern wir im Dokument Koch ein Array der Objekt-IDs aus der Kochkurs Kollektion,um zu identifizieren, welche Kochkurse werden unter der Aufsicht dieses Kochs abgehalten.
Das Array in beide Kollektionen heißt Fuehrung.

Alternative:

Wir können die unabhängige Kollektion Fuehrung erstellen und in Dokumenten die Arrays von Koch und Kochkurs speichern, aber das ist die ineffizienz Lösung.

1.7 Der Koch

```
{
    "_id": ObjectID("5e301acbc5f335538041ddad"),
    "KochID": 2,
    "Rang": "Demi Chef de Partie",
    "Ausbildung": "International Culinary Center",
    "MId": ObjectID("5e301acbc5f335538041dd16"),
    "Fuehrung": [
        ObjectID("5e301acbc5f335538041de99"),
        ObjectID("5e301acbc5f335538041df24"),
        ObjectID("5e301acbc5f335538041df3e"),
        ObjectID("5e301acbc5f335538041dfa1")
    ]
}
```

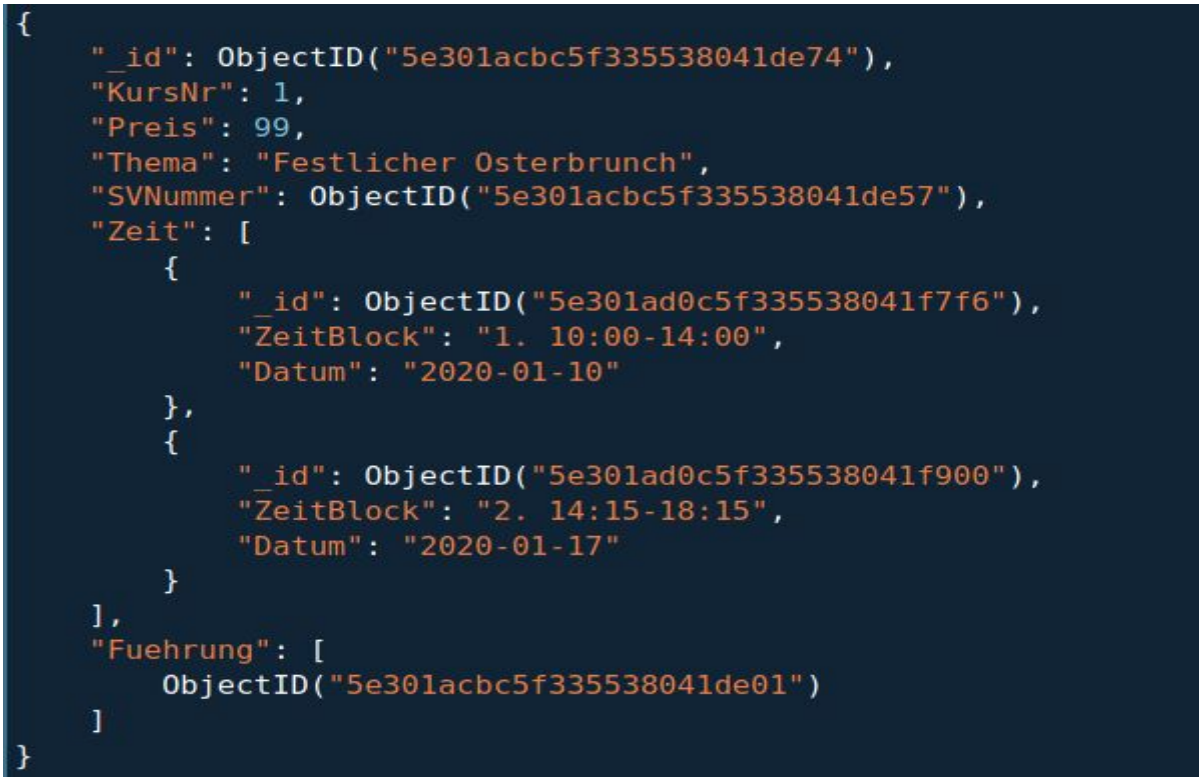
Implementiert:

In Koch Kollektion werden die Attributen von Koch, sowohl die Referenz in MId auf Mitarbeiter Dokument als auch Array von ObjectIDs von Kochkurse gespeichert. Wir haben solches Modell für MId gewählt, weil das der Zugriff auf der Webseite vereinfacht.

Alternative:

Wir können der Koch als eingebettetes Dokument in Mitarbeiter Dokumentation speichern, weil das “is-a” Beziehung ist.

1.8 Der Kochkurs



```
{
    "_id": ObjectID("5e301acbc5f335538041de74"),
    "KursNr": 1,
    "Preis": 99,
    "Thema": "Festlicher Osterbrunch",
    "SVNummer": ObjectID("5e301acbc5f335538041de57"),
    "Zeit": [
        {
            "_id": ObjectID("5e301ad0c5f335538041f7f6"),
            "ZeitBlock": "1. 10:00-14:00",
            "Datum": "2020-01-10"
        },
        {
            "_id": ObjectID("5e301ad0c5f335538041f900"),
            "ZeitBlock": "2. 14:15-18:15",
            "Datum": "2020-01-17"
        }
    ],
    "Fuehrung": [
        ObjectID("5e301acbc5f335538041de01")
    ]
}
```

Implementiert:

In Kochkurs Kollektion werden die Attribute von Kochkurs, die Referenz in SVNummer auf Manager Dokument, Array von Dokumenten von Zeit und Array von ObjectIDs von Koch gespeichert. Wir haben solches Modell für SVNummer gewählt, weil das der Zugriff auf der Webseite vereinfacht. Und wie oben geschrieben, speichern wir hier Array von Zeit wegen ternäre Beziehung zwischen Zeit, Kochkurs und Küche. In ER-Diagramm hat der Kochkurs die Kardinalität 1. Unserer Meinung nach, die Zeit gehört immer zu einem Kochkurs und es wird logisch die Zeit Array hier speichern.

Alternative:

Wir können Kochkurs als eingebettetes Dokument in Manager Dokumentation speichern, weil das “1:1” Beziehung ist. Aber dann wird eine solche Struktur komplex sein und die Zeit für die Abfrage wird sehr langsam.

1.9 Der Kursteilnehmer


```
{
    "_id": ObjectID("5e301accc5f335538041ec37"),
    "KursteilnehmerNr": 24,
    "Vorname": "Hayley",
    "Nachname": "Maldonado",
    "EMail": "HayleyMaldonado23@yahoo.com23",
    "TelefonNr": "+4366083523",
    "AbteilungsNr": ObjectID("5e301acac5f335538041dc76"),
    "KursNr": ObjectID("5e301accc5f335538041e65b")
}
```


Implementiert:

In Kursteilnehmer sind die Attributen von Kursteilnehmer, die Referenz in AbteilungsNr auf Kochschule Dokument und die Referenz in KursNr auf Kochkurs gespeichert. Die Referenzen helfen uns die Zeit für Abfrage speichern.

Alternative:

Wir können Kursteilnehmer als eingebettetes Dokument in Kochkurs speichern, weil das “1:1” Beziehung ist. Aber Kochkurs hat schon ein Array von Dokumenten und um die Speicherplatz zu speichern, haben wir die optimale Lösung gefunden.

1.10 Die Zeit


```
{
    "_id": ObjectID("5e301ad0c5f335538041f7f6"),
    "ZeitBlock": "1. 10:00-14:00",
    "Datum": "2020-01-10"
}
```


Implementiert:

In Zeit Kollektion werden die Attributen von Zeit gespeichert. Die unabhängige Kollektion, weil es den Zugriff vereinfacht.

Alternative:
Wie können die Tabelle aus MySQL als ein Dokument speichern und dort die Array von Zeitwerten haben, aber das ist schlechte Idee, weil die Dokumentgröße das Limit erreichen kann.

Indexing and discussion on optimized sharding of NoSQL DBMS for the IS

MongoDB erstellt standardmäßig einen Index für Dokumenten. Das Index hat der Name "_id" und die Type ObjectID. Es ist auch möglich mehr als ein Index verwenden um die Abfrage zu erstellen. Für unseres Programm können wir die zusammengesetze Indexes verwenden, um die Suche auf der Webseite zu verbessern. Die Dokumenten in unseren Datenbank speichern auch die Arrays, also die Multikey Index sind auch hilfreich.

Sharding könnte bei Kursteilnehmer nützlich sein, da sie die meisten Datensätze bilden im Gegensatz zu Kursen, Küchen, usw. Zum Beispiel könnten diese auf mehrere shards verteilt sein um den Hauptserver zu entlasten.

2. Data Migration

Wir haben das Java-Programm erstellt, um die Migration durchzuführen. Wir haben Java 11(Version 11.0.2) verwendet.
Das Programm liest die Daten aus der relationalen Datenbank ein und konvertiert sie, um die Daten in der NoSQL-Datenbank zu speichern. Jede Tabelle hat eine eigene Klasse zum Konvertieren des Objekts in Kollektionen und zum Migrieren der Daten. Alle Klassen erben die Struktur von der abstrakten Klasse AMigration.
Wie das Programm ausgeführt wird, kann man in der README-Datei nachlesen.

3.3. Implementation IS (NoSQL)

Die folgenden Use Cases funktionieren mit der MongoDB-Verbindung:

1. **Use Case "Kochkurs organisieren"**
2. **Use Case "Den Kochkurs führen"**
3. **Use Case "Am Kochkurs anmelden"**

Sie haben das gleiche Funktionsprinzip wie in MS2 geschrieben wurde.
Die Use Case "Ein Bericht über die Kursteilnahme generieren" funktioniert nicht.

Work protocol

Date, Time	Length in Minutes	Task	Activity	Result	Responsibility	Remarks
20.01.2020, 15:00 - 15:30	30,00	Discussion	We talked about what we have to do and how to split up tasks	Ready for working on MS3	All	
20.01.2020, 15:30-15:50	20	Setting up mongoDB	We had to add mongodb and mongodb express to the docker-compose .yml and properly set it up to interact with other containers	Mongodb and the tool is now integrated into the docker system	All	
20.01.2020		Data Migration	Creating a java program that migrates data from the MySQL database to the Mongo database	The java program has been created and can now be tested for function	Karyna Volobuieva	We decided to take a similar approach to how we did it with the data filling it is going to be executed immediately by the container
20.01.2020, -17:20		Modifying docker-compose.yml			All	
22.01.2020,13:00-15:00		Transforming SQL queries into Mongo queries	After performing research we first tried the MongoClient class, upon learning it was deprecated, we tried MongoDB\Driver\Manager	The queries are set up for MongoClient	Lujza Lanyi	The MongoDB\Driver\Manager is very low level and the MongoClient syntax was simple ans easy to use. We found out that there is a composer package that wraps the new MongoDB driver and provides a similar syntax to the legacy MongoClient

23.01.2020		Creating Migration Java Program			Karyna Volobuieva	
24.12.2020, 14:00-17:00		Installing composer and MongoDB\Client in the project and adapting the code	To use MongoDB\Client (the package that wraps the low level driver as it was mentioned above) we had to install composer in the container and let composer set up the package	The client is now fully integrated in the code and the queries are set up to be used with it	Lujza Lanyi	
27.01.2020, 20:30-21:00	30	Fixing connections	I changed 'localhost' to the container name and added password and user name to the connection strings	Connecting and inserting into MongoDB now works	Lujza Lanyi	
27.01.2020, 18:00-22:00	240	Creating migrations for each table	We created migrations for each table. The data gets converted from table columns to collection fields	All tables have been migrated.	Karyna Volobuieva	
28.01.2020, 14:00-15:00	60	Writing the overall report	Describing all parts of project	The report is done	Karyna Volobuieva	
