Jeanette Schmid, Eva Stepanek 4CHIT

GK Document Oriented Middleware using MongoDB

Nennen Sie 4 Vorteile eines NoSQL Repository im Gegensatz zu einem relationalen DBMS.

1. Skalierbarkeit: NoSQL-Datenbanken sind horizontal skalierbar.
2. Flexibilität: Schemafrei, ermöglicht dynamische Änderungen an der Datenstruktur.
3. Performance: Schnelle Lese- und Schreiboperationen, besonders bei großen Datenmengen.
4. Verteilte Speicherung: Gute Unterstützung für verteilte Systeme und Replikation.

Nennen Sie 4 Nachteile eines NoSQL Repository im Gegensatz zu einem relationalen DBMS.

1. Konsistenz: Oft eventual consistency statt ACID-Transaktionen.
2. Fehlende Standardisierung: Kein einheitlicher Abfrage-Standard wie SQL.
3. Weniger komplexe Abfragen: Fehlende Joins erfordern oft zusätzliche Verarbeitung.
4. Begrenzte Unterstützung für Transaktionen: Besonders problematisch für Finanzanwendungen.

Welche Schwierigkeiten ergeben sich bei der Zusammenführung der Daten?

Daten können in unterschiedlichen Formaten, Strukturen und Systemen gespeichert sein, was Konvertierung und Harmonisierung erfordert. Redundanzen und Inkonsistenzen müssen erkannt und gelöst werden. Unterschiedliche Konsistenzmodelle können zu Konflikten führen.

Welche Arten von NoSQL Datenbanken gibt es?

1. Key-Value-Datenbanken
2. Dokumentenorientierte Datenbanken
3. Spaltenorientierte Datenbanken
4. Graphdatenbanken

Nennen Sie einen Vertreter für jede Art?

1. Key-Value: Redis
2. Dokumentenorientiert: MongoDB
3. Spaltenorientiert: Apache Cassandra
4. Graph: Neo4j

Beschreiben Sie die Abkürzungen CA, CP und AP in Bezug auf das CAP Theorem

- CA (Consistency & Availability): Garantiert Konsistenz und Verfügbarkeit, aber keine Partitionstoleranz (z. B. traditionelle relationale DBs).
- CP (Consistency & Partition Tolerance): Garantiert Konsistenz und Partitionstoleranz, aber kann unzugänglich sein (z. B. HBase).
- AP (Availability & Partition Tolerance): Garantiert Verfügbarkeit und Partitionstoleranz, aber nicht immer Konsistenz (z. B. DynamoDB).

Mit welchem Befehl können Sie den Lagerstand eines Produktes aller Lagerstandorte anzeigen?

```
db.products.aggregate([ { $match: { productID: '03-123456' } }  ]);
```

Mit welchem Befehl können Sie den Lagerstand eines Produktes eines bestimmten Lagerstandortes anzeigen?

Jeanette Schmid, Eva Stepanek 4CHIT

GKü:

Nach dem herunterladen des Docker-Images und das implementieren in einen Container, habe ich das bereitgestellte Git-Repository auf ein neues Projekt gepullt.

Dann habe ich den Code um die bereits in der letzten Aufgabe geschriebene Klasse „WarehouseData“ erweitert.
Um mich jetzt mit der Datenbank zu verbinden, musste ich nur noch ein application.properties-File mit den passenden Daten erstellen:

```
# MongoDB Verbindungsdetails
spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=warehouse
```

(Die Datenbank „warehouse“ wird automatisch erstellt, wenn sie noch nicht existiert.)

Zusätzlich habe ich noch jede der Klassen, welche eine Entity repräsentieren, mit einem

‚@Document(collection = <Name der Collection>’

Versehen. Da dies bestimmt in welche Collection ein Objekt hineingespeichert wird.

```
@Document(collection = "products")
public class ProductData {

    2 usages
    @Id
    private String ID;

    4 usages
    private String warehouseID;
    4 usages
    private String productID;
```

(„products“ wird ebenfalls automatisch erstellt. @Id lässt Spring die Id automatisch als ein Unikat erstellen.)

Jeanette Schmid, Eva Stepanek 4CHIT

GKv:

```
warehouse> db.products.find()
[
  {
    _id: ObjectId('67c7195d24f2767e33d57dab'),
    warehouseID: '1',
    productID: '00-443175',
    productName: 'Bio Orangensaft Sonne',
    productCategory: 'Getraenk',
    productQuantity: 2500,
    _class: 'warehouse.model.ProductData'
  },
```

Mit db.products.find() werden alle Objekte in der Collection products ausgegeben.

```
warehouse> db.products.deleteOne({ "productID": '03-893173'})
{ acknowledged: true, deletedCount: 1 }
```

Mit db.products.deleteOne() wird das erste Element, welches die passenden Attribute in der Klammer hat, gelöscht. Es gibt noch eine zweite Variante nämlich deleteMany(). Diese löscht alle Elemente mit den passenden Attributen, nicht nur das Erste.

```
warehouse> db.products.insertOne({ "warehouseID": '2', "productID": '00-123456', "productName": 'Shampoo', "pro
ductCategory": 'Waschmittel', "productQuantity": 123})
{
  acknowledged: true,
  insertedId: ObjectId('67c7211346ff8b597a51e944')
}
```

Mit db.products.insertOne() wird ein Produkt mit den eingegebenen Attributen in die Collection eingefügt.

```
warehouse> db.products.updateOne({"productID": '00-123456'}, {$set: {"productName": 'Seife'}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Mit db.products.updateOne() wird ein Produkt, welches mit der angegebenen ID übereinstimmt, verändert werden. Hier zum Beispiel wurde der Name geändert.

Jeanette Schmid, Eva Stepanek 4CHIT

EKü:

Das Programm wurde um ein WarehouseData und ein WarehouseRepository erweitert.

```java
@Document(collection = "warehouses")
public class WarehouseData{
    3 usages
    @Id
    private String warehouseID;

    4 usages
    private String warehouseName;
    4 usages
    private String warehouseAddress;
    4 usages
    private String warehousePostalCode;
    4 usages
    private String warehouseCity;
    4 usages
    private String warehouseCountry;
    4 usages
    private Timestamp timestamp;

    4 usages
    private List<ProductData> productData;
```

Vom Prinzip her funktioniert es wie ProductData und ProductRepository. Es wird eine separate Collection für warehouses erstellt, sobald die Klasse zum ersten Mal aufgerufen wird. Das Repository kommuniziert mit der Collection und gibt Daten vom Typ WarehouseData zurück.

```java
public interface WarehouseRepository extends MongoRepository<WarehouseData, String>
    Jeanette Schmid
    public List<WarehouseData> findAll();
    1 usage  Jeanette Schmid
    public WarehouseData findByWarehouseID(String warehouseID);
    no usages  Jeanette Schmid
    public void deleteByWarehouseID(String warehouseID);

}
```

Es wurden 3 Methoden in das Interface geschrieben: findAll() um alle Warehouses auszugeben, findByWarehouseID(String warehouseID) um ein bestimmtes Warehouse über seine ID auszugeben und deleteByWarehouseID(String warehouseID) um ein Warehouse über seine ID aus der Collection zu löschen.

Das ProductRepository wurde um 2 Methoden erweitert:

```java
public interface ProductRepository extends MongoRepository<ProductData, String> {

    2 usages  Jeanette Schmid
    public ProductData findByProductID(String productID);
    2 usages  Jeanette Schmid
    public List<ProductData> findByWarehouseID(String warehouseID);
    Jeanette Schmid
    public List<ProductData> findAll();

    1 usage  Jeanette Schmid
    public void deleteByProductID(String productID);
}
```

findAll() um alle Produkte auszugeben und deleteByProductID(String productID) um ein Product über seine ID aus der Collection zu löschen.

Jeanette Schmid, Eva Stepanek 4CHIT

EKv:

Wie ist der Gesamtbestand eines bestimmten Produkts (nach productID) über alle Lagerstandorte?

```
warehouse> db.products.aggregate([ {$match: {productID: "00-443175"} }, {$group: {_id: "$productID", total
Stock: {$sum: "$productQuantity"} }}, {$project: {_id: 0, productID: "$_id", totalStock: 1}} ]);
[ { totalStock: 2, productID: '00-443175' } ]
```

Welche Produkte haben einen Gesamtbestand von weniger als 10 Stück über alle Lager hinweg?

```
warehouse> db.products.aggregate([ {$group: { _id: "$productID", totalStock: { $sum: "$productQuantity"} }
}, {$match: {totalStock: {$lt: 10} } }, {$project: { _id: 0, productID: "$_id", totalStock:1}} ]);
[
  { totalStock: 1, productID: '00-871895' },
  { totalStock: 1, productID: '01-926885' },
  { totalStock: 2, productID: '00-443175' }
]
```

Welche Lagerstandorte haben insgesamt einen Lagerbestand von weniger als 1000 Artikeln (über alle Produkte zusammen)?

```
warehouse> db.products.aggregate([ {$group: { _id: "$warehouseID", totalStock: {$sum: "$productQuantity" }
}}, {$match: { totalStock: {$lt: 1000} } }, {$project: { _id: 0, warehouseID: "$_id", totalStock: 1 }} ]);

[ { totalStock: 3, warehouseID: '1' } ]
```

Erklärung der Befehle:

$group: gruppiert alle Produkte nach entweder warehouseID oder productID und die Menge der Produkte im Lager bzw. über alle Lager (je nach Fragestellung) wird zusammengezählt.

$match: Filtert die Lager oder die Produkte, welche unter einer bestimmten Menge gefallen sind.

$project: Gibt das Ergebnis in übersichtlicher Form aus (warehouseID oder productID + totalStock)