MTG Trading Card API

Dokumentation

des Studienganges Informatik

an der Dualen Hochschule Baden-Württemberg Stuttgart

von

Ole Reimers

vorgelegt am 02.12.2022

Matrikelnummer:	9865392
Kurs:	STG-TINF 20E
Studienjahrgang:	2020
Modul:	Big Data

Inhaltsverzeichnis

1. Installation

Im Folgenden wird beschrieben, wie das Projekt auf einer Kommandozeile mit installiertem Docker aufgesetzt wird.

Zunächst wird ein Netzwerk erstellt, damit die Docker Container kommunizieren können.

```
docker network create --driver bridge bigdatanet
```

Listing 1.1: Docker Netzwerk erstellen

Anschliessend werden die Images von Docker Hub heruntergeladen.

```
docker pull marcelmittelstaedt/spark_base
docker pull ostfrost420/airflow
docker pull ostfrost420/mongodb
docker pull ostfrost420/express
```

Listing 1.2: Images herunterladen

Die Container werden gestartet.

```
docker run -dit --name hadoop \
-p 8088:8088 -p 9870:9870 -p 9864:9864 -p 10000:10000 \
-p 8032:8032 -p 8030:8030 -p 8031:8031 -p 9000:9000 \
-p 8888:8888 --net bigdatanet \
marcelmittelstaedt/spark_base:latest

docker run -dit --name airflow \
-p 8080:8080 \
--net bigdatanet \
ostfrost420/airflow

docker run --name mongodb-container -d \
-p 27017:27017 \
--net bigdatanet \
ostfrost420/mongodb

docker run --name express-container -d \
-p 5000:5000 \
```

```
--net bigdatanet \
ostfrost420/express
```

Listing 1.3: Container starten

Hive und der Hadoop Cluster werden gestartet.

```
docker exec -it hadoop bash
sudo su hadoop
cd
start-all.sh
hiveserver2
```

Listing 1.4: Hive starten

Airflow kann nach der Installation über den Port 8080 und die Webanwendung über den Port 5000 erreicht werden.

2. Aufbau des DAG

In Abbildung 2.1 ist der Aufbau des DAGs in Airflow dargestellt. Die einzelnen Schritte werden im Folgenden beschrieben:

create_import_directory Im Airflow Container wird gegebenenfalls ein neuer Ordner erstellt, der als Zwischenspeicher der Rohdaten dient. Ist der Ordner schon vorhanden, passiert in diesem Schritt nichts.

clear_import_directory Falls noch Dateien von einem früheren Durchlauf im Ordner vorhanden sind, werden diese in diesem Schritt gelöscht.

download_mtg_cards In diesem Schritt wird auf die Magic the Gathering API zugegriffen, um die Informationen aller Spielkarten zu importieren. Die Daten werden in zwei JSON-Dateien (eine für jede Tabelle) zwischengespeichert. Der Speicherort ist der im ersten Schritt angelegte Ordner. Der konkrete Ablauf wird in Kapitel 3 erklärt.

mkdir_hdfs_*_dir In diesen Schritten werden die Ordner der Tabellen im Hadoop File System angelegt. Der Ordnerpfad wird an das Datum des Ausführens angepasst, um alte Daten-

stände nicht zu überschreiben. Ab diesem Punkt läuft die DAG für die Tabellen cards und foreign_cards parallel.

upload_*_to_hdfs Es werden die importierten JSON-Dateien mit den Magic the Gathering Daten in die entsprechenden Ordner im Hadoop File System hochgeladen.

create_*_table Es werden die Tabellen der Rohdaten erstellt, falls sie noch nicht vorhanden sind. Die Tabellen sind nach dem Hochladedatum partitioniert.

add_partition_*_table Es werden neue Partitionen in den Tabellen zum aktuellen Datum angelegt.

dummy Es wird gewartet, bis die parallelen DAGs fertiggestellt wurden.

create_cards_reduced Es wird, falls sie noch nicht vorhanden ist, eine Tabelle angelegt für die reduzierten Daten.

add_jar_dependencies Es werden Abhängigkeiten zu Hive hinzugefügt, die den Umgang mit JSON-Dokumenten ermöglichen

hive_write_cards_reduced_table Der Map-Reduce Algorithmus wird durchgeführt, und die reduzierten Daten werden in die Tabelle cards_reduced gespeichert. Der Schritt des Reduzierens wird in Kapitel 4 weiterführend erklärt

hive_to_mongodb Die reduzierten Daten werden in die Mongo Datenbank gespeichert. Der Schritt wird in Kapitel 5 erläutert.

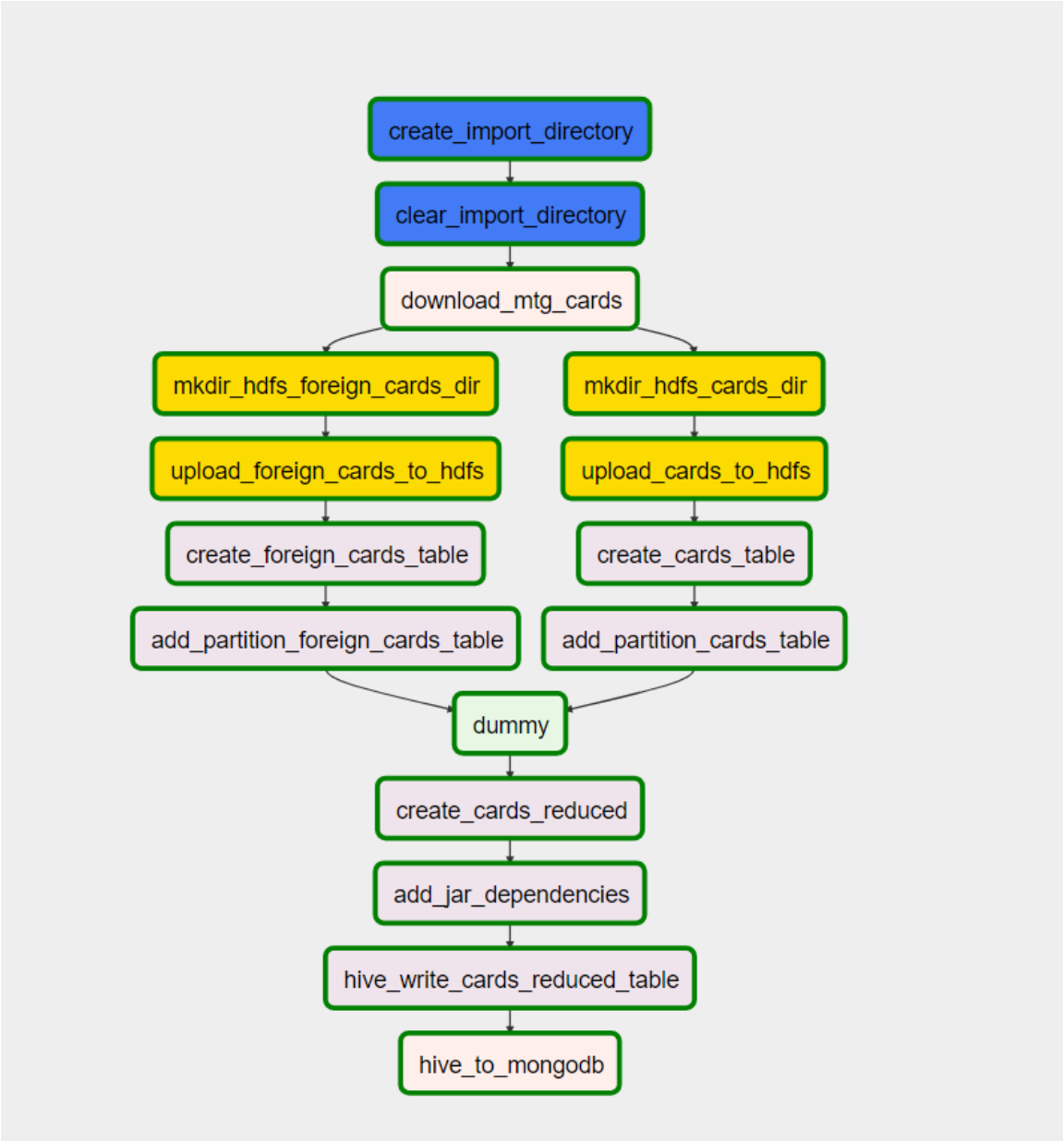

Abbildung 2.1: Aufbau der DAG

3. Importieren der Rohdaten

Die Aufgabe ist es, eine durchsuchbare Datenbank für die Spielkarten des Kartenspiels Magic the Gathering zu erstellen. Im ersten Schritt werden die Daten hierfür aus der entsprechenden Schnittstelle von magicthegathering.io importiert.

Die Schnittstelle ist unter der URL „https://api.magicthegathering.io/v1/cards“ zu erreichen.

Die Karten werden im JSON Format, Seitenweise zurückgegeben. Eine spezifische Seite kann über den „page“ Paramter angefordert werden. Die Anzahl Karten pro Seite kann über den „pageSize“ Parameter übergeben werden und ist auf maximal 100 Karten pro Seite begrenzt.

Zum Importieren aller Karten werden deshalb die Daten aller Seiten iterativ angefordert, zusammengefügt und abschliessend wieder in eine JSON-Datei gespeichert.

Zunächst werden die Karten der ersten Seite angefordert. Im Header der Antwort befindet sich im Parameter „Total-Count“ die Anzahl aller Karten. Über die Kartenanzahl und die PageSize kann die Anzahl der Seiten berechnet werden. Die übrigen Aufrufe werden zur Verbesserung der Performance parallel von mehreren Threads durchgeführt, bis alle Karten angefordert wurden.

Die Kartenobjekte beinhalten neben ihren Attributen zusätzlich Verweise auf Internationalisierungen der Karte, die einige Informationen in verschiedenen Übersetzungen angibt. Es bietet sich an, diese Informationen, um die Komplexität der SQL Querys im Map-Reduce Schritt zu vereinfachen, in zwei Tabellen aufzuteilen. Die Daten werden dementsprechend in ein JSON-Dokument mit Informationen zu den Karten und ein JSON-Dokument mit Informationen zu den Übersetzungen aufgeteilt. Die Einträge werden über die ID der Karte verknüpft.

Damit Hadoop die JSON-Dateien einlesen kann, müssen die einzelnen Objekte durch Zeilenumbrüche getrennt werden. Die Karten- und Internationalisierungsobjekte werden deshalb in das JSON-Format formattiert und durch Zeilenumbrüchen getrennt in Dateien geschrieben.

4. Reduzieren der Daten

Für die Webanwendung zum Durchsuchen der Magic Karten sollen das Bild der deutschen Karte, der englische Titel und die ID in die Mongo Datenbank übertragen werden. Alle anderen Informationen sind für die Anwendung irrelevant. Vor der Übertragung wird deshalb ein Map Reduce Job durchgeführt. Hierzu wird zunächst eine neue Tabelle cards_reduced angelegt, die nur die notwendigen Spalten besitzt. Anschliessend werden die Einträge aus den zwei Tabellen mit der folgenden Query in die Tabelle eingefügt:

```
INSERT OVERWRITE TABLE cards_reduced
SELECT
  c.name,
  c.multiverseid,
  f.imageUrl
FROM
  cards c
  JOIN foreign_cards f ON (c.id = f.cardid)
WHERE
  f.language = "German";
```

Listing 4.1: Map Reduce Query

Der englische Name und die Multiverseid werden aus der Karten Tabelle übernommen. Die URL des deutschen Bildes der Karte aus der Internationalisierungstabelle. Bestehende Daten werden überschrieben.

5. Übertragen der Daten in die Mongo Datenbank

Um die Daten in die Mongo Datenbank zu übertragen, wird ein kleines Pythonskript ausgeführt. Das Skript baut eine Verbindung zu Hive auf und liest alle Daten der cards_reduced Tabelle ein.

Anschliessend werden diese Daten aufbereitet, um beispielsweise fehlende oder fehlerhafte Felder mit passenden Daten zu ersetzen. Zum Beispiel wird ein fehlender Name und eine fehlende oder fehlerhafte ID mit dem „-“ String ersetzt. Eine fehlende Bild-URL wird mit einer URL zu einem Bild des Magic The Gathering Kartenrückens ersetzt.

Die aufbereiteten Daten werden im letzten Schritt in die Mongo Datenbank gespeichert. Veraltete Daten in der Datenbank werden davor gelöscht.

6. Interaktion im Webinterface

Die Webanwendung besteht aus einem Express-Server, der ein in Angular geschriebenes Frontend und eine Schnittstelle zur Datenbanksuche bereitstellt. Das Angular Frontend besitzt eine

Suchleiste und eine Tabelle, in der die Ausgabe angezeigt wird. Es ist in Abbildung 6.1 zu sehen. Nach dem Eingeben eines Suchbegriffs und Klicken des „Search“ Buttons wird zunächst

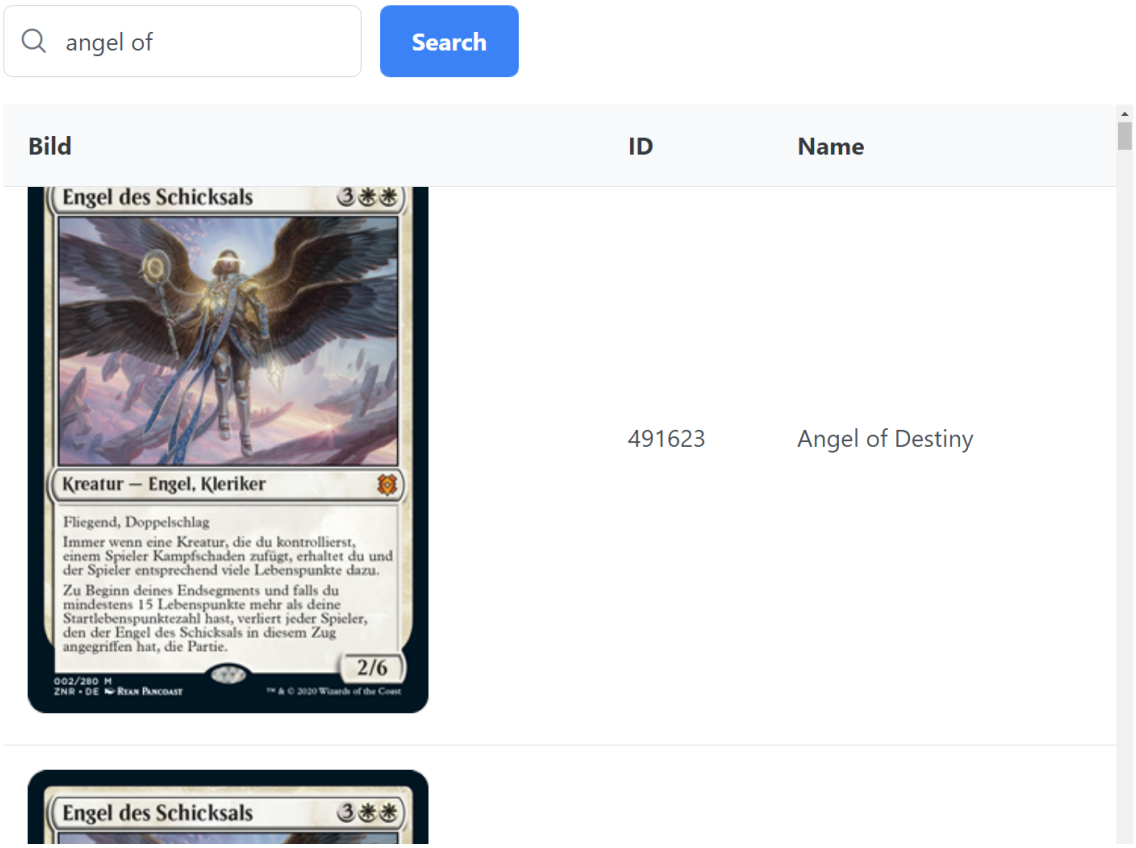

Abbildung 6.1: Screenshot des Frontends

eine Anfrage an den Express-Server gestellt. Dieser sucht in der Mongo Datenbank nach Ergebnissen, in denen der Suchbegriff Teil des Namens oder genau die ID ist. Die Ergebnisse werden an das Frontend zurückgegeben. Dieses zeigt sie im Anschluss in der Tabelle an.

A. Verzeichnisse

A.1 Abbildungsverzeichnis

A.2 Listings