ddi: Mini-Challenge LE3 – Nicht relationale Datenbanken (NoSQL)

Lukas Reber

Wahl der Datenbank

Ich habe für diese Mini-Challenge die Graph-Datenbank Neo4J gewählt. (https://neo4j.com/). Neo4j existiert seit 2007 und ist Open-Source verfügbar. Dies macht sie zu einer der beliebtesten Graph-Datenbanken. Neo4j verfügt über eine eigene Query Language mit dem Namen «Cypher»

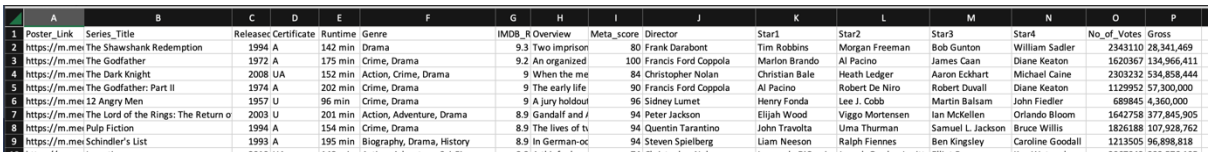
Graph-Datenbanken sind besonders dafür geeignet, Beziehungen zwischen verschiedenen Entitäten zu speichern und aufzuzeigen. Im Unterschied zu einer klassischen relationalen Datenbank werden nicht Tabellen und Spalten miteinander verknüpft, sondern die Daten selbst. So existiert bei einer Graph Datenbank kein Schema wie bei einer SQL-Datenbank und es können beliebig neue Datenpunkte miteinander verknüpft werden.

Use-Case

Für mein Beispiel habe ich einen Datensatz der Top 1000 bestbewerteten Filme auf IMDB verwendet. Der Datensatz ist auf Kaggle frei verfügbar. (https://www.kaggle.com/datasets/bansodesandeep/imdb-top-1000-movies)

Zu jedem der Filme sind Name, Jahr, Genre, Rating, Directors, Schauspieler und weitere Attribute aufgeführt (welche ich jedoch nicht verwende).

Die Daten sehe in der Rohfassung so aus:

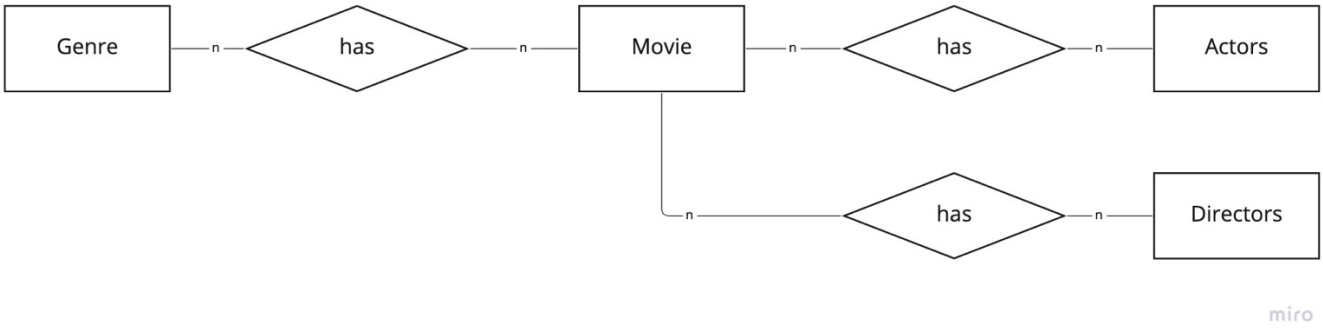


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Poster_Link	Series_Title	Releasec	Certificate	Runtime	Genre	IMDB_R	Overview	Meta_score	Director	Star1	Star2	Star3	Star4	No_of_Votes	Gross
2	https://m.mei	The Shawshank Redemption	1994	A	142 min	Drama	9.3	Two imprison	80	Frank Darabont	Tim Robbins	Morgan Freeman	Bob Gunton	William Sadler	2343110	28,341,469
3	https://m.mei	The Godfather	1972	A	175 min	Crime, Drama	9.2	An organized	100	Francis Ford Coppola	Marlon Brando	Al Pacino	James Caan	Diane Keaton	1620367	134,966,411
4	https://m.mei	The Dark Knight	2008	UA	152 min	Action, Crime, Drama	9	When the me	84	Christopher Nolan	Christian Bale	Heath Ledger	Aaron Eckhart	Michael Caine	2303232	534,858,444
5	https://m.mei	The Godfather: Part II	1974	A	202 min	Crime, Drama	9	The early life	90	Francis Ford Coppola	Al Pacino	Robert De Niro	Robert Duvall	Diane Keaton	1129952	57,300,000
6	https://m.mei	12 Angry Men	1957	U	96 min	Crime, Drama	9	A jury holdout	96	Sidney Lumet	Henry Fonda	Lee J. Cobb	Martin Balsam	John Fiedler	689845	4,360,000
7	https://m.mei	The Lord of the Rings: The Return o	2003	U	201 min	Action, Adventure, Drama	8.9	Gandalf and A	94	Peter Jackson	Elijah Wood	Viggo Mortensen	Ian McKellen	Orlando Bloom	1642758	377,845,905
8	https://m.mei	Pulp Fiction	1994	A	154 min	Crime, Drama	8.9	The lives of tv	94	Quentin Tarantino	John Travolta	Uma Thurman	Samuel L. Jackson	Bruce Willis	1826188	107,928,762
9	https://m.mei	Schindler's List	1993	A	195 min	Biography, Drama, History	8.9	In German-oc	94	Steven Spielberg	Liam Neeson	Ralph Fiennes	Ben Kingsley	Caroline Goodall	1213505	96,898,818

Werden die Daten normalisiert in einer Datenbank hinterlegt, entstehen sehr viele verknüpfte Daten. Denn

- jeder Film gehört zu einem oder mehreren Genres
- jeder Film hat ein Director
- jeder Director hat ein oder mehrere Filme
- jeder Film hat mehrere Schauspieler
- jeder Schauspieler hat einen oder mehrere Filme

Datenmodell



Vergleich Graph-Datenbank zu SQL
In einer relationalen Datenbank wäre z.B. die Beziehung zwischen Film und Genre wie folgt aufgebaut:

Tabelle Movies

ID	Name
1	Schindlers List
2	Shawshak Redeption

Tabelle Movies_Genres

Movie_ID	Genre_ID
1	1
1	2
Usw.	Usw.

Tabelle Genres

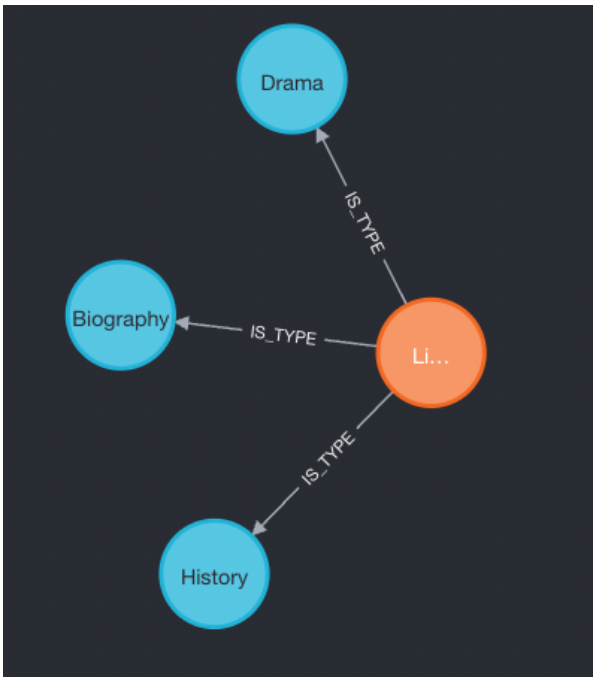
ID	Name
1	Drama
2	Biography
Usw.	Usw.

Es existiert eine Zwischentabelle für die Tabellen Genres und Movies, welche die Beziehungen zueinander aufzeigt. Diese Tabellen könne über die entsprechende Joins zusammengefügt werden.

In der Graph Datenbank sind Verknüpfungen direkt hinterlegt und eine Abfrage bedingt weder Zwischentabellen noch Joins:

```
Bsp Query Neo4J Cypher:
MATCH (m:Movie)-[d:IS_TYPE]->(g:Genre)
WHERE m.name = 'Schindlers List'
RETURN *
```

Resultat



Die Graph Datenbank kennt somit nur die folgenden drei Einheiten:

- Nodes: Dies sind die Knoten, es können beliebig viele Knotenarten sowie Knoten erstellt werden. Im aufgeführten Beispiel sehen wir 1 Node des Typ «Movie» sowie 3 Nodes des Typ «Genre»
- Properties: Jeder Node kann beliebig viele Properties haben. Dies sind Attribute welche zu einem bestimmten Node gehören.
- Relationships: Nodes können miteinander über Relationsships verbunden sein. Die Relationsship erhält hierbei einen Namen. In unserem Beispiel sind Movies mit Genres über die Relationship «is_type» verbunden.

Zentrale Stärken

Die Vorteile einer Graph Datenbank sind:

- Die Abfrage verknüpfter Daten ist sehr schnell
- Der Syntax für verknüpfte Daten abzufragen ist im Vergleich zu SQL sehr einfach
- Die Datenbank verfügt über kein vordefiniertes Schema, es können somit beliebig weitere Daten hinzugefügt werden.

Vergleich zu anderen NoSQL Datenbanken

Geeignet:

Datenbanken mit welcher sich verschiedene Entitäten abbilden lassen und sich die Daten verknüpfen lassen, sind grundsätzlich geeignet. So könnte der Datensatz auch in einer schemalosen Document based Datenbank wie MongoDB oder CouchDB gut abgebildet werden. Jedoch wäre der Syntax für die Abfrage einiges komplizierter als bei Neo4j und je nach Abfrage weniger performant.

Ungeeignet:
Grundsätzlich als ungeeignet sind für den angegeben Use Case Datenbanksystem welche keine Verknüpfung der Daten zulassen. Als Beispiel kann hier eine TimeSeries Datenbank wie InfluxDB genannt werden. Denn ohne Verknüpfungen kommt die Stärke einer Graph Datenbank nicht zum tragen.

Beschreibung der Implementierung
Der Code der Umsetzung befindet sich im folgenden Notebook:
https://github.com/lukasreber/ddi/blob/main/mc_le3.ipynb

Um den Code auszuführen, muss vorgängig eine lokale Version von Neo4j installiert werden. Neo4j kann über folgenden Link heruntergeladen werden: https://neo4j.com/download/

Das Notebook ist wie folgt aufgebaut:

1. Laden der nötigen Packages
2. Verbinden mit lokaler Neo4J Datenbank
3. Laden der Daten aus dem csv welche in die Datenbank importiert werden sollen.
4. Sämtliche Daten welche aktuell in der DB sind löschen (falls bereits Daten vorhanden sind, damit mit einem sauberen Setup gestartet werden kann)
5. Laden der Entitäten Personen, Movies, Genres
6. Erstellen der Verknüpfungen zwischen Movies-Genres, Movies-Directores und Movies-Actors.
7. Abfragen der eingelesenen Daten