Import von strukturierten XML-Daten in neo4j

Contents

1 Inhalt

{:.no_toc}

- Will be replaced with the ToC, excluding the “Contents” header {:toc}

2 Import von strukturierten XML-Daten in neo4j

In diesem Kapitel wird der Import von strukturierten XML-Daten in die Graphdatenbank neo4j beschrieben. Strukturiert meint hierbei, dass es sich nicht um Text handelt, der beispielsweise in TEI-XML ausgezeichnet ist, sondern um Daten in einer datenbank-ähnlichen Struktur. Die Daten

stammen aus einem Projekt meines Kollegen Thomas Kollatz, der sie mir freundlicherweise zur Verfügung gestellt hat. Ziel des Kapitels ist es, die Struktur der XML-Daten im Graphen nachzubilden und anschließend den Import durchzuführen.

2.1 Das XML-Beispiel

In der folgenden Abbildung wird ein Auszug aus den Daten gezeigt.

```
<collection>
    <work id="1">
        <title>Be'ur millot ha-higgajon</title>
        <autor>[1] Mose ben Maimon</autor>
        <autor>[2] Nieto, David ben Pinchas</autor>
        <kommentator>Mendelssohn, Moses</kommentator>
        <druckort>Berlin</druckort>
    </work>
    <work id="2">
        <title>Be'ur millot ha-higgajon</title>
        <autor>Mose ben Maimon</autor>
        <kommentator>Mendelssohn, Moses</kommentator>
        <druckort>Berlin</druckort>
    </work>
    <work id="3">
        <title>Be'ur millot ha-higgajon</title>
        <autor>Mose ben Maimon</autor>
        <kommentator>Mendelssohn, Moses</kommentator>
        <druckort>Berlin</druckort>
    </work>
```

Figure 1: Auszug aus dem XML-Beispiel (Quelle: Kuczera)

Das root-Element der XML-Datei ist `<collection>`. Innerhalb der `collection` finden sich Angaben zu verschiedenen Büchern. Zu jedem Buch werden folgende Angaben gemacht:

- Titel des Buches
- Autor(en) des Buches
- Kommentator des Buches
- Druckort des Buches

2.2 Knotentypen

Für die Modellierung dieser Datenstruktur in der Graphdatenbank müssen zunächst die verschiedenen Knotentypen festgelegt werden. Als erstes scheint es sinnvoll einen Knoten vom Typ `Werk` anzulegen, wie es auch im XML über das `<work>`-Element modelliert ist. Die dem `<work>`-Element untergeordneten Elemente `<title>` `<autor>`, `<kommentator>` und `<druckort>` können dann als Properties des `<work>`-Knotens angelegt werden. Damit wären alle Informationen des XMLs auch im Graphen gespeichert.

Die Graphmodellierung geht hier jedoch einen Schritt weiter. In einem ersten Schritt sind die in den Daten vorhandenen Entitäten zu identifizieren. Neben dem oben schon genannten Knoten vom Typ `Werk` ergeben sich noch Knoten vom Typ `Person` und `Ort`. In den `Ort`-Knoten werden die Druckorte abgelegt, die Angaben zu Autoren und Kommentatoren werden in den `Person`-Knoten gespeichert, da es sich ja um Personen handelt, die aber je nach Situation verschiedene Rollen einnehmen können.

2.3 Kantentypen

Nach den Knotentypen sind nun die Kantentypen festzulegen. Sie geben an, in welcher Beziehung die verschiedenen Knoten zueinander stehen. Sieht man sich die XML-Vorlage an, ergeben sich folgene Typen von Kanten:

- GEDRUCKT_IN
- AUTOR_VON
- KOMMENTIERT_VON

Mit der `GEDRUCKT_IN`-Kante werden ein Werk und ein Ort verbunden und damit angegeben, dass dieses Buch in jenem Ort gedruckt worden ist.

Die `AUTOR_VON`-Kante verbindet einen Personenknoten mit einem Werkknoten und ordnet damit den Autor dem von ihm geschriebenen Buch zu.

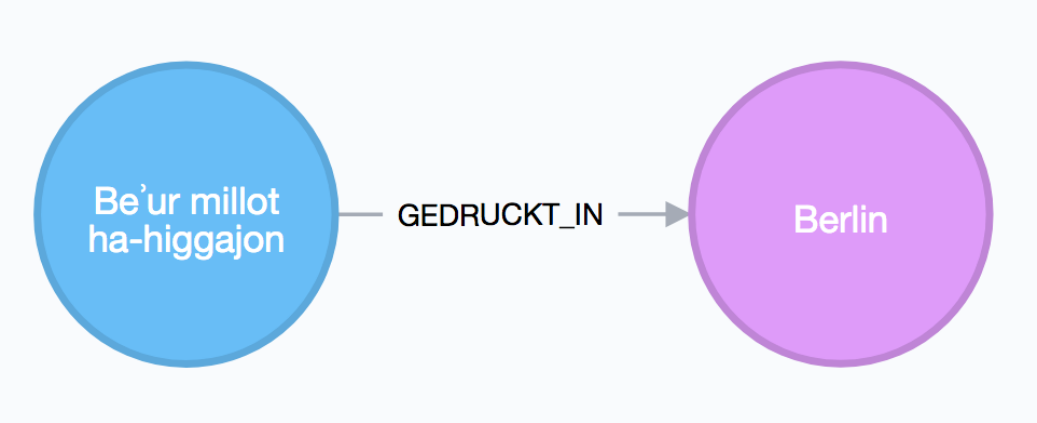

Figure 2: Verbindung zwischen einem `Werk`- und einem `Ort`-Knoten (Quelle: Kuczera).

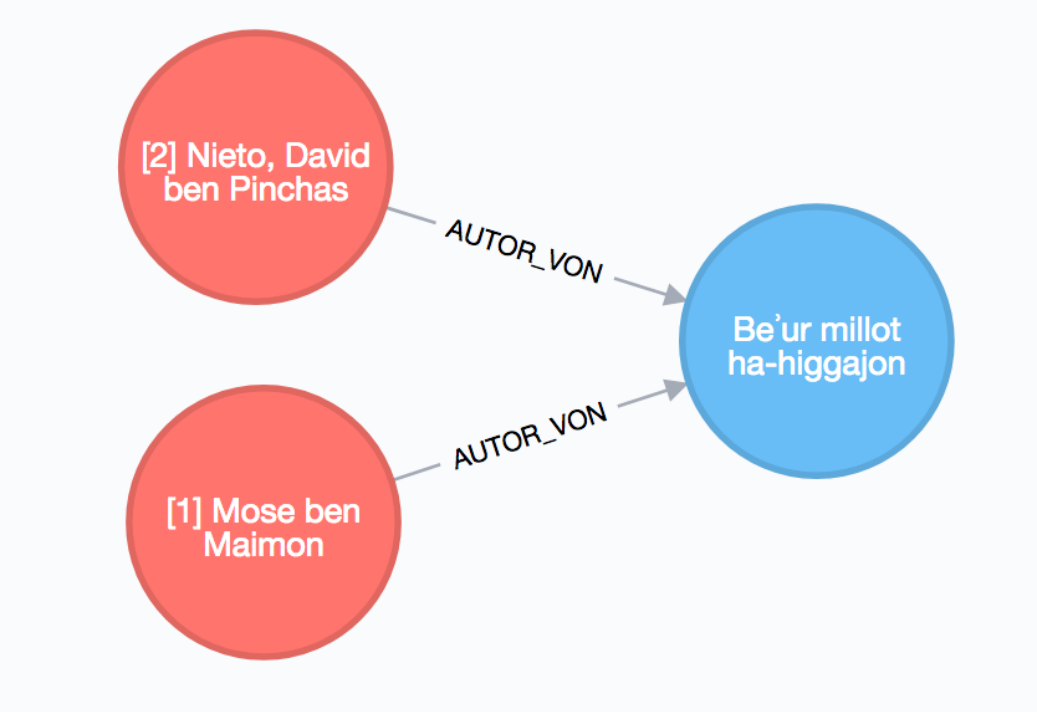

Figure 3: Verbindung zwischen einem `Werk`- und einem `Ort`-Knoten (Quelle: Kuczera).

Mit der KOMMENTIERT_VON-Kante wird auch ein Personenknoten einem Werkknoten zugeordnet, diesmal nimmt die Person aber die Rolle des Kommentierenden ein.

Figure 4: Verbindung zwischen einem Werk- und einem Ort-Knoten (Quelle: Kuczera).

Im der folgenden Abbildung werden alle Knoten und Kanten des Beispiels gemeinsam dargestellt.

Damit steht das Graphmodell fest und im nächsten Abschnitt geht es an den Import.

2.4 Der Import mit apoc.load.xmlSimple

Für den Import von XML-Daten steht in der apoc-Bibliothek der Befehl apoc.load.xmlSimple zur Verfügung. Im folgenden wird zunächst der gesamte Befehl für den Import gelistet.

```
CALL apoc.load.xmlSimple("https://docs.google.com/
document/u/0/export?format=txt&id=1Ujx9cdequEvuXx6
DxK7xfxbuS63Aq9n_xpLOAcjU_xc&token=AC4w5Vj0yTkdPLo
b1lo1ajkaG75fynNZ0Q%3A1490694667158") yield value
UNWIND value._work as wdata
    MERGE (w1:Werk{eid:wdata.id})
    set w1.name=wdata._title._text
    FOREACH (name in wdata._autor |
```

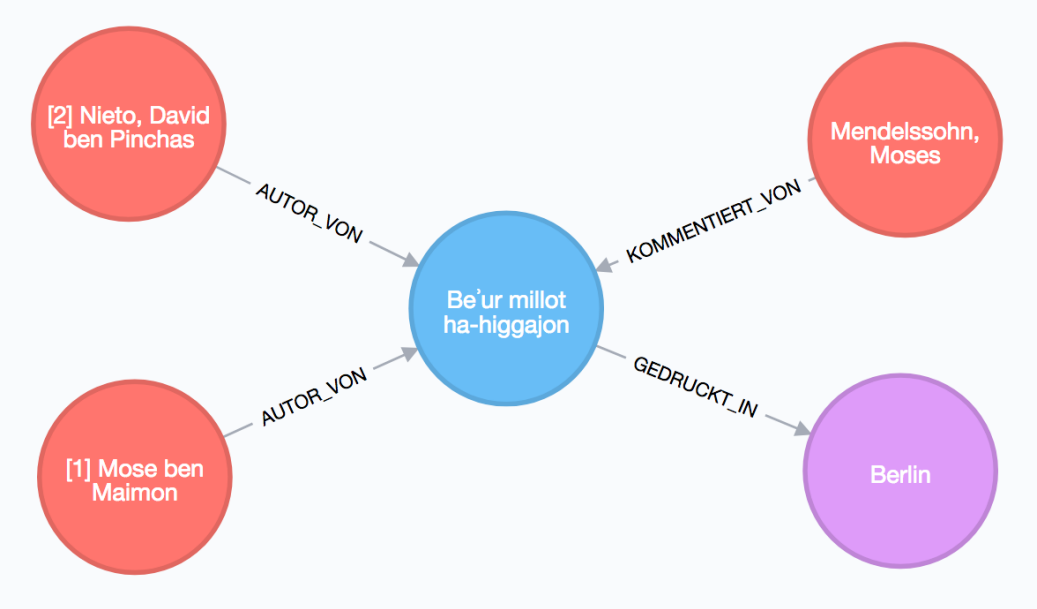

Figure 5: Verbindung zwischen einem `Werk`- und einem `Ort`-Knoten (Quelle: Kuczera).

```
    MERGE (p1:Person {Name:name._text})
    MERGE (p1)-[:AUTOR_VON]->(w1) )
FOREACH (name in wdata._kommentator |
    MERGE (p1:Person {Name:name._text})
    MERGE (p1)-[:KOMMENTIERT_VON]->(w1))
FOREACH (druckort in [x in
    wdata._druckort._text where x is not null] |
     MERGE (o1:Ort{name:druckort})
     MERGE (w1)-[:GEDRUCKT_IN]->(o1));
```