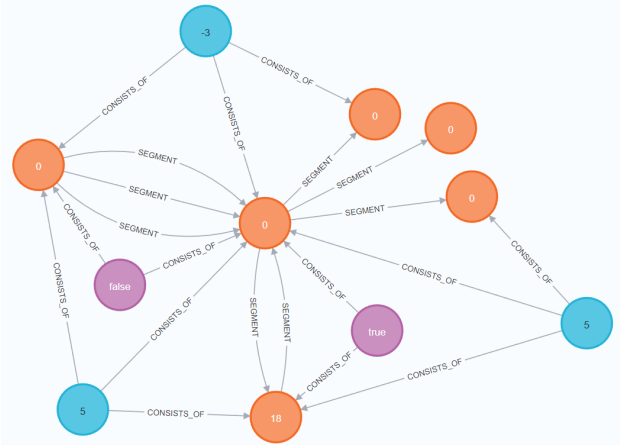
Gruppenaufgabe 1

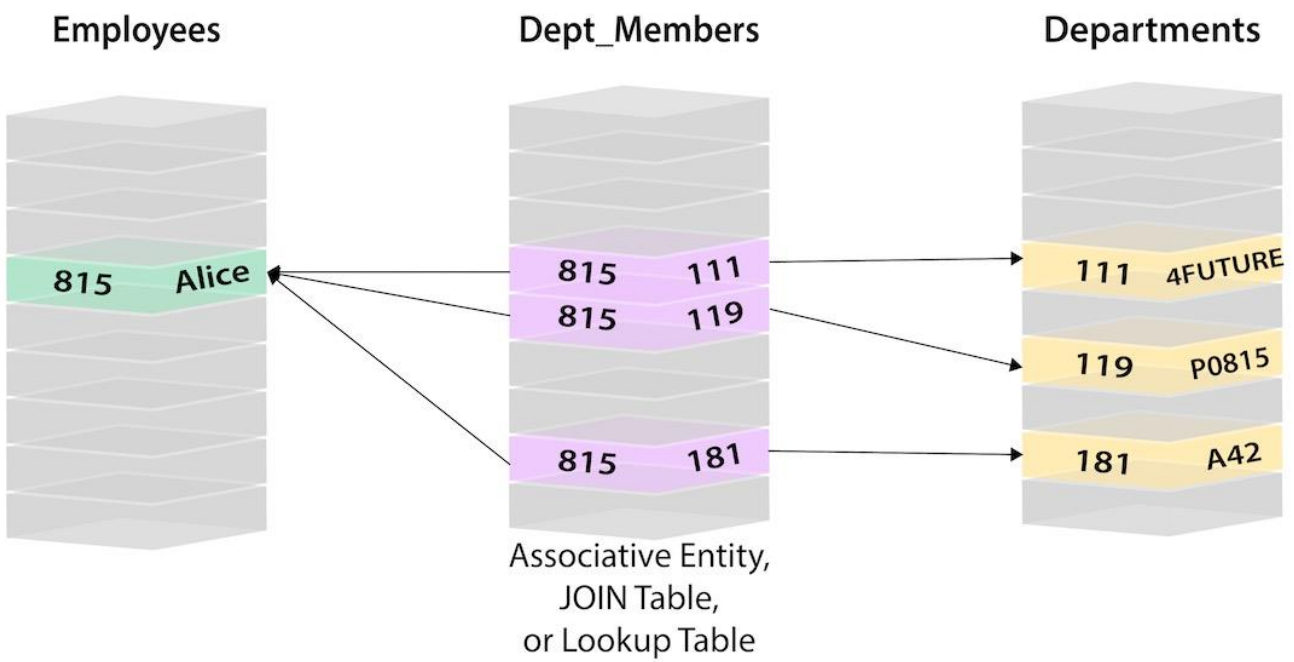
Graphdatenbankpraktikum



Graphdatenbank Motivation

- Repräsentation von Graphen im relationalen Modell
- Performance Probleme, große Joins

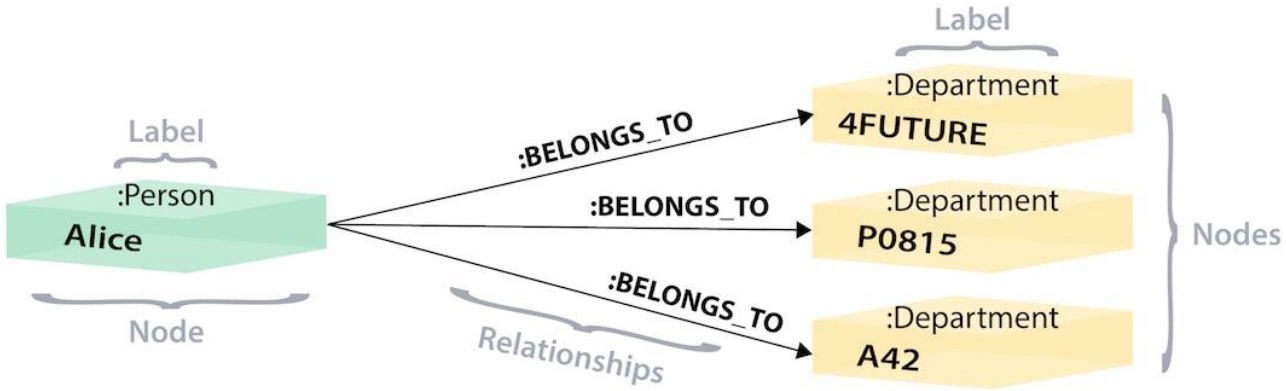
-> Strukturierung als Graph



Quelle: https://neo4j.com/developer/relational-to-graph-modeling/

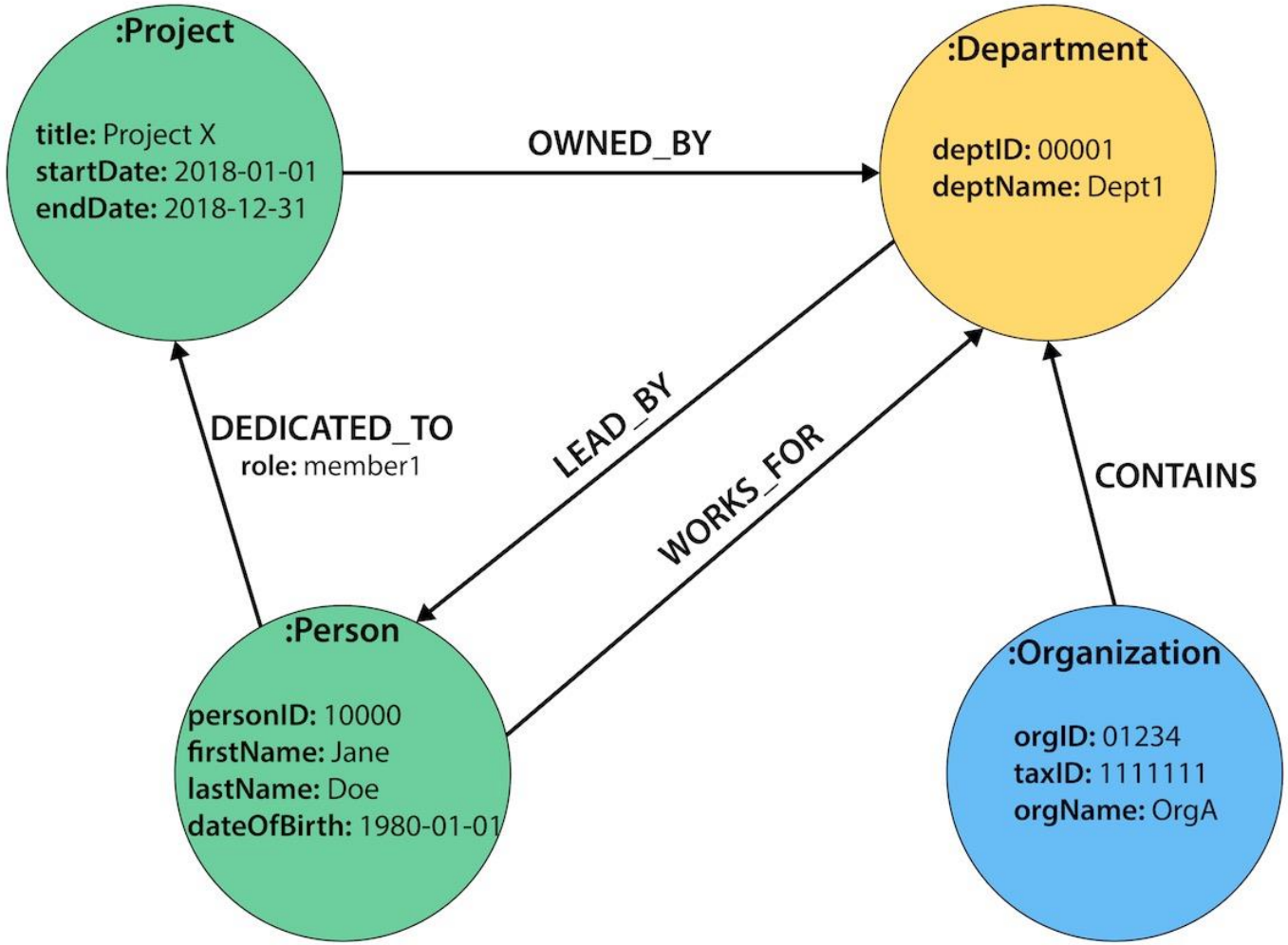
Graphdatenbank

- 4 (Kern-)Elemente: Node, Label, Relationships, Attribute
- Label = Tabelle
- Node = Zeile in Tabelle
- Relationships = Fremdschlüssel
- Attribut = Spalte in Tabelle



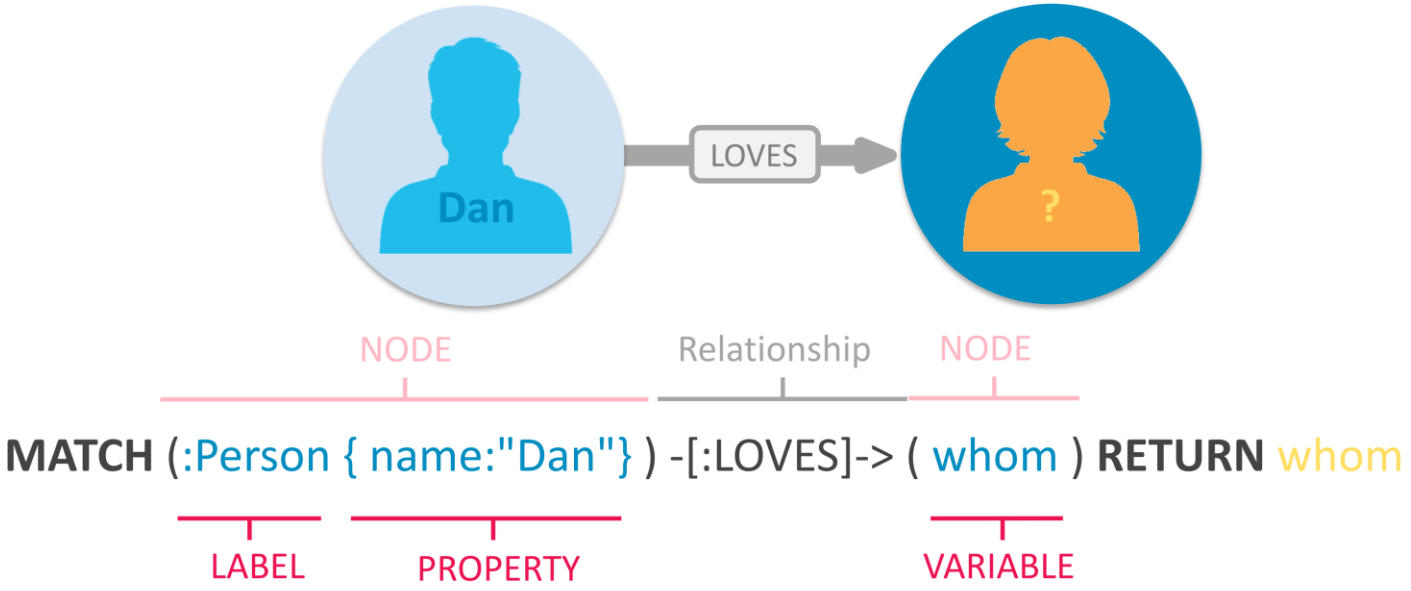
Quelle: https://neo4j.com/developer/relational-to-graph-modeling/

Graphdatenbank



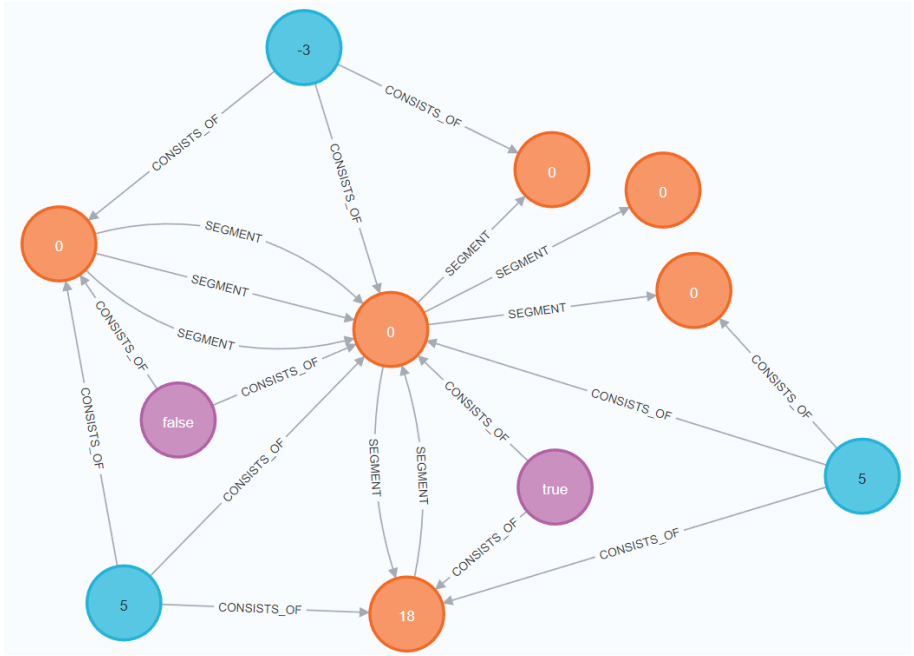
Cypher

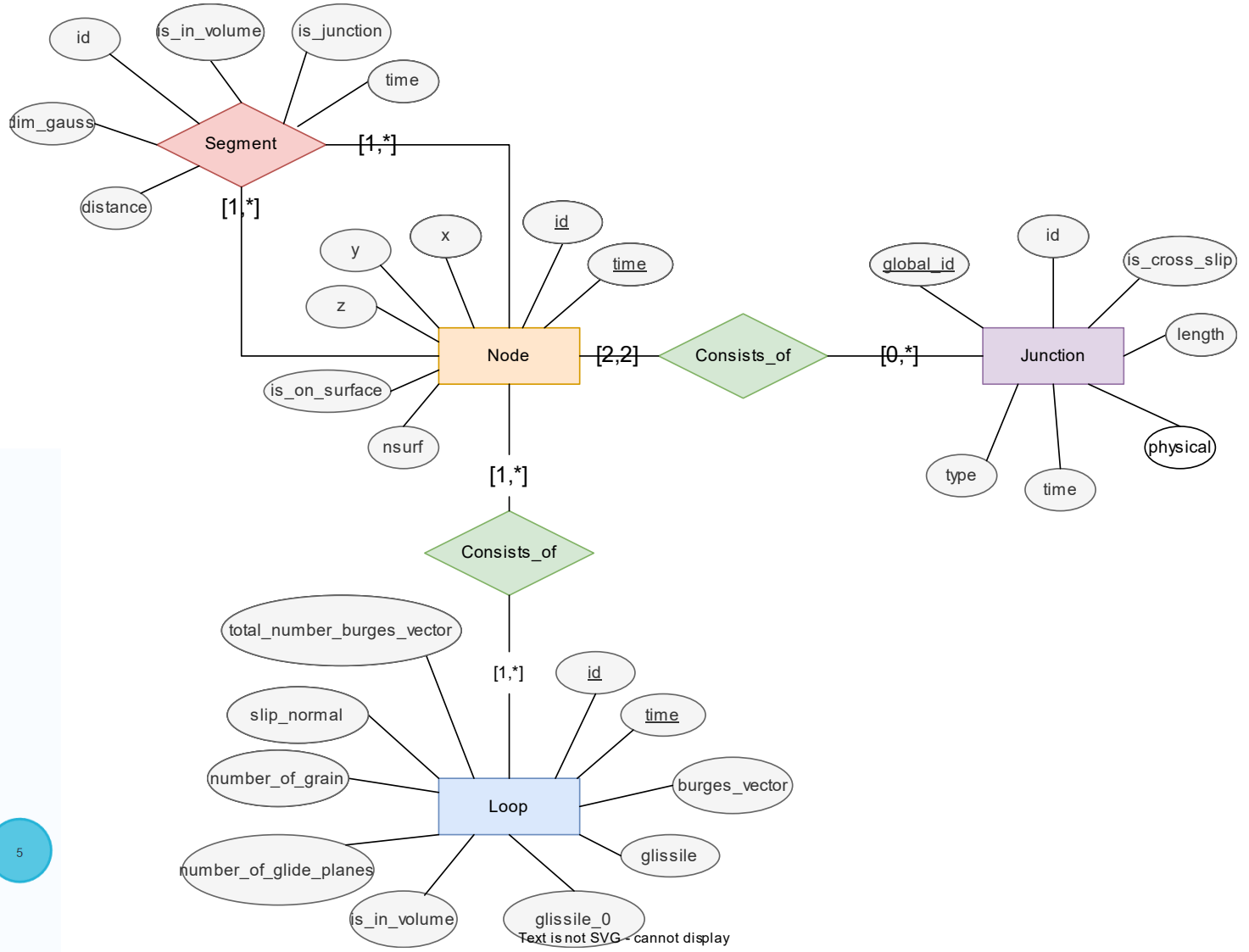
- Querysprache für Graphdatenbanken
- Syntax: MATCH {Pattern} WHERE {Attribute} RETURN {Variablen}
- Pattern: Beschreibender Ausdruck von Beziehungen, enthält Variablen



Quelle: https://neo4j.com/developer/relational-to-graph-modeling/

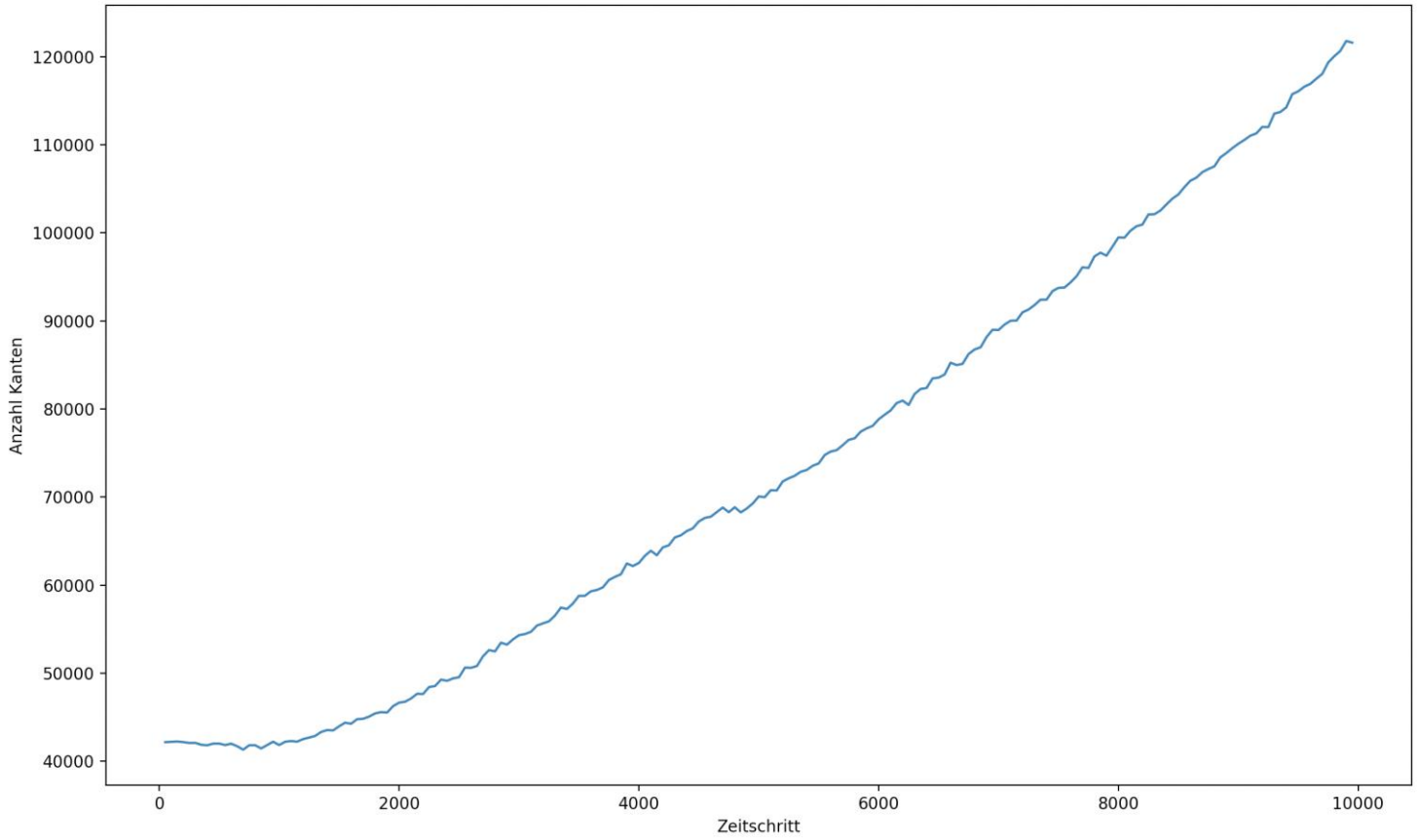
Aufgabe 1: ER-Diagramm



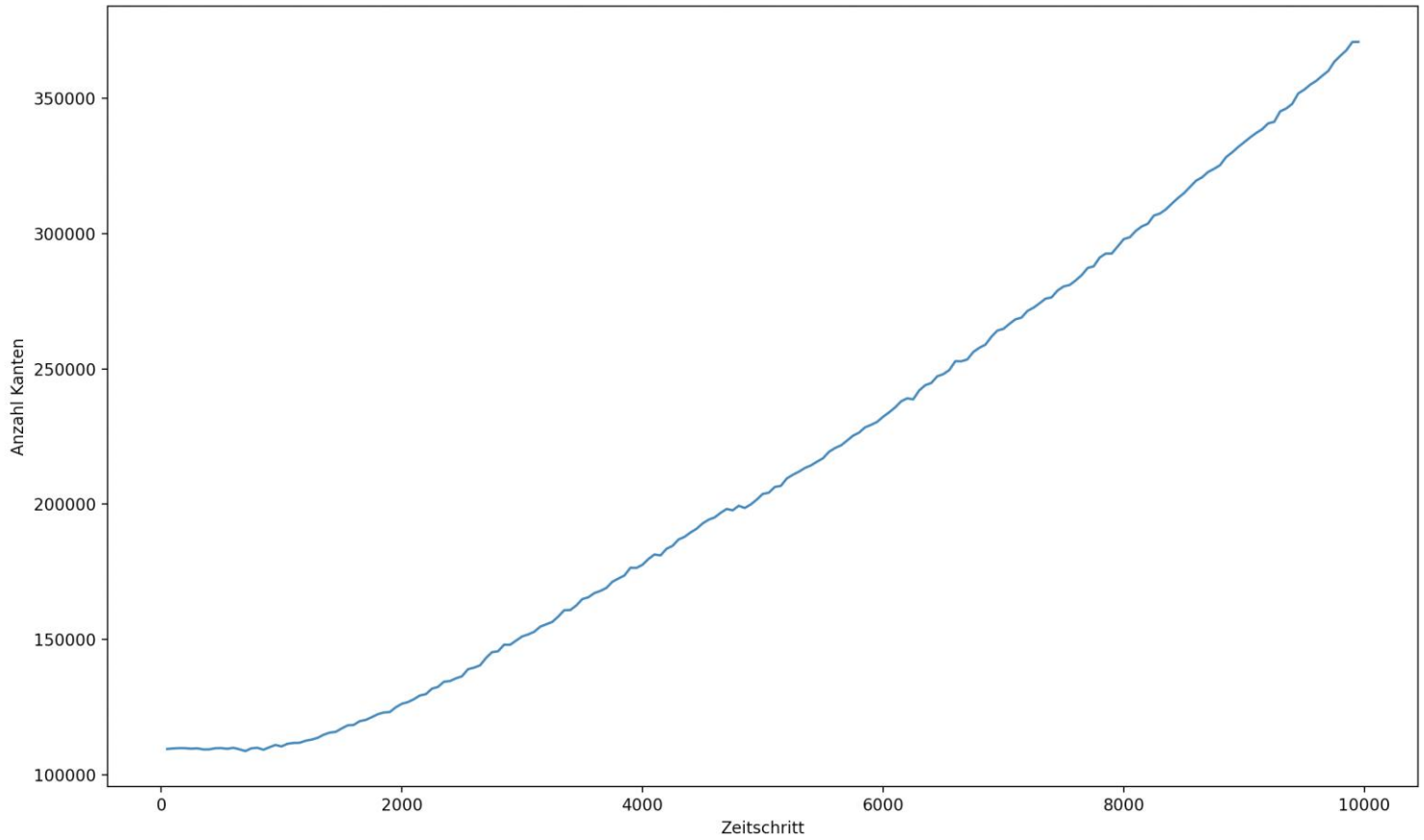


2a) Wie viele Knoten existieren pro Zeitschritt, wie entwickelt sich deren Anzahl über alle Zeitschritte?

```
MATCH (n:Node)
RETURN n.time, COUNT(*)
```



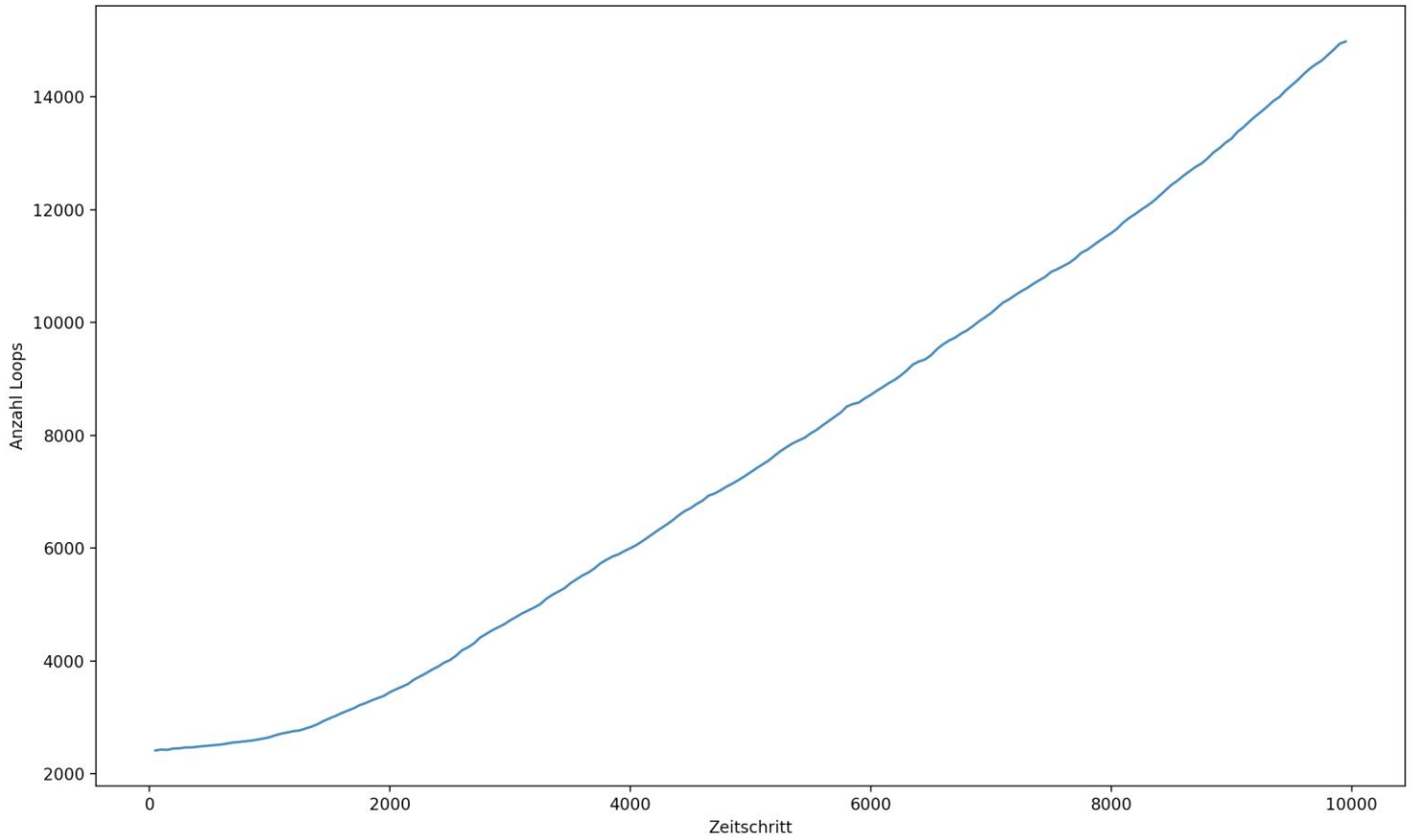
2b) Wie viele Kanten existieren pro Zeitschritt, wie entwickelt sich deren Anzahl?




```
MATCH
(:Node)-[s:SEGMENT]-(:Node)
RETURN s.time, COUNT(*)
```

2c) Wie viele “Loops” existieren und wie entwickelt sich deren Anzahl?

```
MATCH (n:Loop)
RETURN n.time, COUNT(*)
```



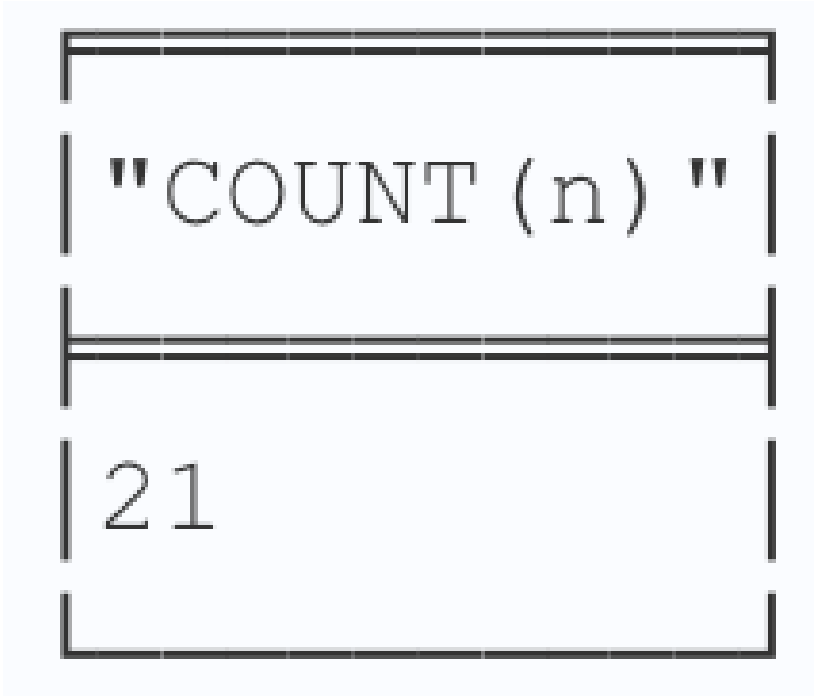
2d) Wie viele Verbindungen zu anderen Loops haben Loops im Durchschnitt?

```
MATCH (n:Loop)
WITH n, size((n)-[:CONSISTS_OF]->(:Node)<
-[:CONSISTS_OF]-(:Loop)) as connections
RETURN AVG(connections)
```

```
╒═══════════════════════╕
│"AVG(connections)"     │
╞═══════════════════════╡
│16.874770585593915     │
└───────────────────────┘
```

2e) Wie viele ”Loops” existieren, welche keine Verbindungen mit anderen Loops eingehen?

```
MATCH (n:Loop)
WHERE NOT (n)--()--(:Loop)
RETURN COUNT(n)
```



2f) Wie viele Loops existieren pro Zeitschritt mit den Attributen burges_vector = 1, slip_normal = 1 und >= 10 Knoten?

```
MATCH (n:Loop)
WHERE size((n)
-(:Node)) >= 10 AND n.burges_vector = 1
AND n.slip_normal = 1
RETURN n.time,COUNT(n)
```

(no changes, no records)

=> Keine Ergebnisse gefunden

3a) Gruppieren Sie alle Loops anhand Ihrer Lebenszeit

```
MATCH (l:Loop)
WITH l.id AS ids,
max(l.time)-min(l.time) AS Lifetime
RETURN Lifetime,
collect(ids) as Loops
```

	Lifetime	Loops
1	9900	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37,
2	9850	[2413, 2414, 2415, 2416, 2417, 2418, 2419, 2420, 2421, 2422, 2423, 2424, 2425, 2426, 2427, 2428, 2429]
3	9750	[2430, 2431, 2432, 2433, 2434, 2435, 2436, 2437, 2438, 2439, 2440, 2441, 2442, 2443, 2444, 2445, 2446, 2447]
4	9700	[2448, 2449, 2450]
5	9650	[2451, 2452, 2453, 2454, 2455, 2456, 2457, 2458, 2459, 2460, 2461, 2462, 2463, 2464, 2465, 2466]
6	9550	[2467, 2468, 2469, 2470, 2471, 2472, 2473, 2474, 2475, 2476, 2477, 2478]

3b) Erzeugen Sie für jede Loop eine Liste aller ihrer zugehörigen Junctions

	Loop	Junction
1	2072	[1, 140, 140, 141, 141, 304, 499, 499, 511, 511, 745, 780, 780, 786, 786, 797, 939, 939, 958, 1172, 1
2	1655	[1, 1, 39, 39, 132, 132, 249, 249, 263, 263, 456, 754, 754, 853, 1007, 1090, 1090, 1283, 1283, 1523,

```
MATCH (l:Loop)-[:CONSISTS_OF]->(:Node)<-[:CONSISTS_OF]-(j:Junction)
RETURN distinct l.id AS Loop, collect(j.id) AS Junction
```

3c) Erzeugen Sie für jeden Zeitschritt eine Liste aller Loops mit dem Junction-Typ 5

```
MATCH (l:Loop)-[:CONSISTS_OF]-> (n:Node)<-[:CONSISTS_OF]-(j:Junction{type: 5})

RETURN l.time AS TimeStep, collect(distinct l.id) AS LoopType5
```

	TimeStep	LoopType5
1	50	[1580, 553, 2125, 714, 709, 434, 793, 367, 2075, 811, 153, 2074, 351, 851, 368, 1389, 852, 2337, 863,
2	100	[1594, 559, 2141, 723, 718, 439, 800, 372, 2090, 818, 156, 2089, 356, 858, 373, 1400, 859, 868, 866,
3	150	[1588, 558, 2133, 717, 722, 438, 800, 372, 2082, 2081, 818, 356, 154, 1395, 858, 857, 373, 867, 865,
4	200	[1605, 562, 2156, 728, 723, 443, 806, 376, 2105, 2104, 824, 360, 157, 1409, 864, 863, 377, 2372, 875
5	250	[1606, 567, 2159, 733, 728, 448, 808, 382, 2108, 2107, 827, 366, 160, 1410, 867, 866, 383, 876, 874,

3d) Erzeugen Sie für jeden Zeitschritt und jeden Junction-Typ eine Liste aller zugehörigen Loops

```
MATCH (l:Loop)-[:CONSISTS_OF]->(n:Node)<-[:CONSISTS_OF]-(j:Junction)
RETURN l.time AS TimeStep, j.type AS JunctionType, collect(distinct l.id) AS Loop
```

TimeStep	JunctionType	Loop
50	4	[2072, 1655, 604, 602, 601, 600, 2411, 1236, 1237, 244, 1987, 1569, 317,
50	2	[815, 811, 385, 2253, 1203, 1200, 770, 1205, 1202, 573, 1987, 1410, 1408,
50	1	[1726, 1129, 12, 2296, 1112, 198, 1186, 793, 554, 553, 1478, 302, 878, 215
50	5	[1580, 553, 2125, 714, 709, 434, 793, 367, 2075, 811, 153, 2074, 351, 851,
50	7	[2018, 2014, 2011, 1331, 1328, 979, 977, 976, 980, 365, 505, 1335, 1332,

Aufgabe 3e

Finde für jede Loop mit dem Junction Typ 1 alle Nachbarloops mit den Junction Typen 2 und 5

```
MATCH (l:Loop)-[:CONSISTS_OF]->(n:Node)<-[:CONSISTS_OF]-(j:Junction{type: 1})
MATCH (n)<-[:CONSISTS_OF]-(a:Loop)-[:CONSISTS_OF]->(:Node)<-[:CONSISTS_OF]-(j2:Junction)
WHERE j2.type in [2, 5]
RETURN l.id, collect(a.id)
```