

Evaluierung von temporalen Graphdatenbanken am Beispiel von Neo4j

KOLLOQUIUM VON OVE FOLGER

Gliederung

- Aktuelle Datenmodelle
- Graphdatenbank Neo4j
- Versuchsaufbau
- Ergebnisse
- Limitierungen
- Fazit und Ausblick zu Neo4j

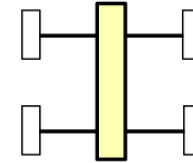
Aktuelle Datenmodelle

SQL Datenbanken

Relational

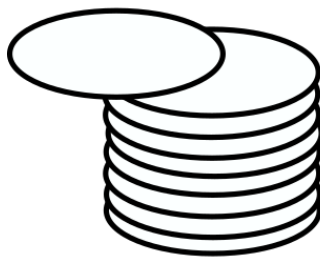


Analytisch (OLAP)

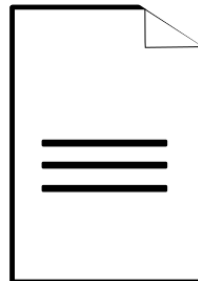


NoSQL Datenbanken

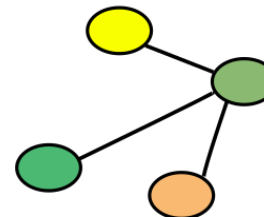
Transaktion (OTLP)



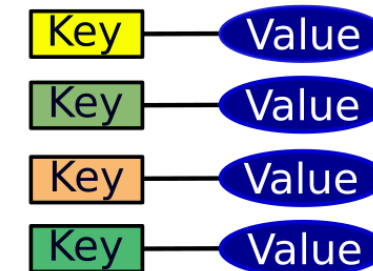
Dokumente



Graph



Schlüssel-Wert



Graphdatenbank Neo4j



Veröffentlicht 2010



Modellierung und
Traversierung von
Netzwerken



Kostenlos in
Community Version



In Cypher und Java
programmierbar



Vorhandene Treiber
für andere Sprachen

Cypher

- () – Entität
- [] – Relation
- {} – Bedingung
- Return – Rückgabe
- Match – Angabe eines Musters
- Beispiel:

```
MATCH (X:Person{Name : 'Peter'})-[:Freund]->(Y:Person)  
RETURN COUNT(DISTINCT(Y))
```

Versuchsaufbau – Aufbau des Graphen



50.004 Knoten und
250.100.000 Kanten



4 Aktions-Knoten



50.000 Person-Knoten



50.000
RELATIONSHIP1 und
RELATIONSHIP2



125.000.000
RELATIONSHIP3 und
RELATIONSHIP4

Beispiel: Grundanfrage 1.3

Semantisch :

- Finde die gemeinsamen Nachbarn von Person1 und Person2

Syntaktisch :

- ```
MATCH (X:Person {name: 'Person1'})-[:Relationship3]->(n1)
WITH COLLECT(n1) as n
MATCH (Y:Person {name: 'Person2'})-[:Relationship3]->(n1)
WHERE n1 in n
RETURN COUNT(DISTINCT(n1))
```

Alternative Syntax:

- ```
MATCH (X:Person {name: 'Person1'})-[:Relationship3]->(n1) <-[:Relationship3]-(Y:Person {name: 'Person2'})
RETURN COUNT(DISTINCT(n1))
```

Normierte Verteilung der Grundanfrage 1.3

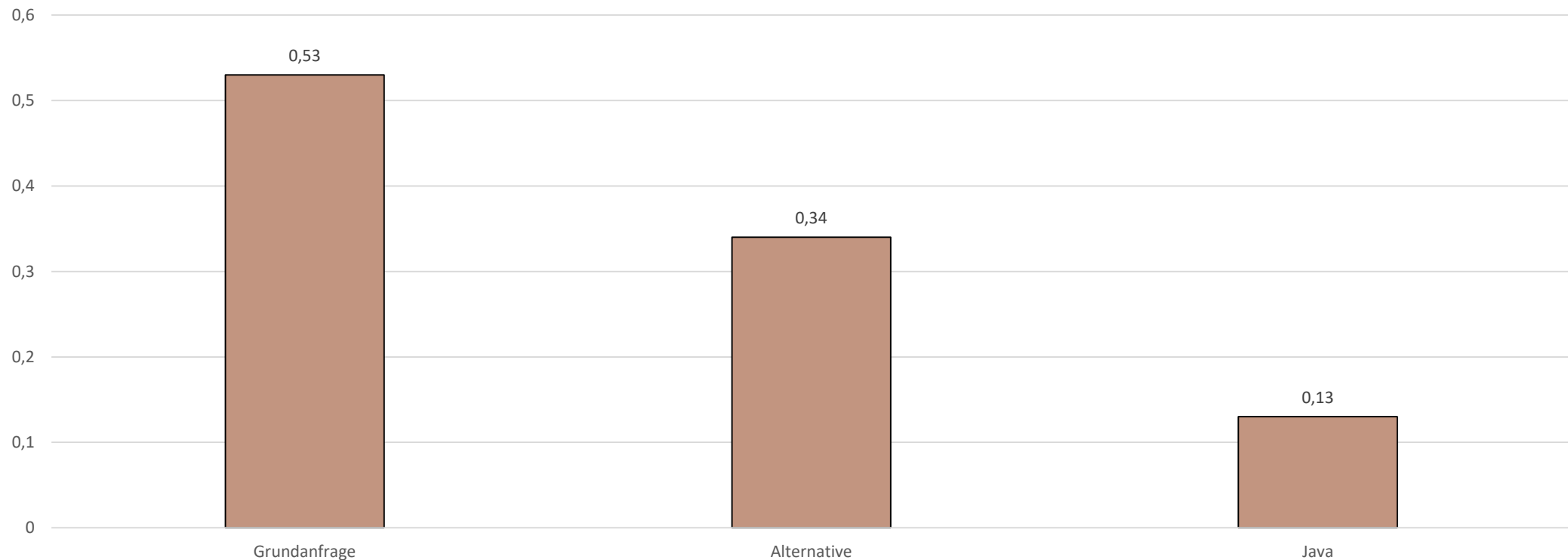
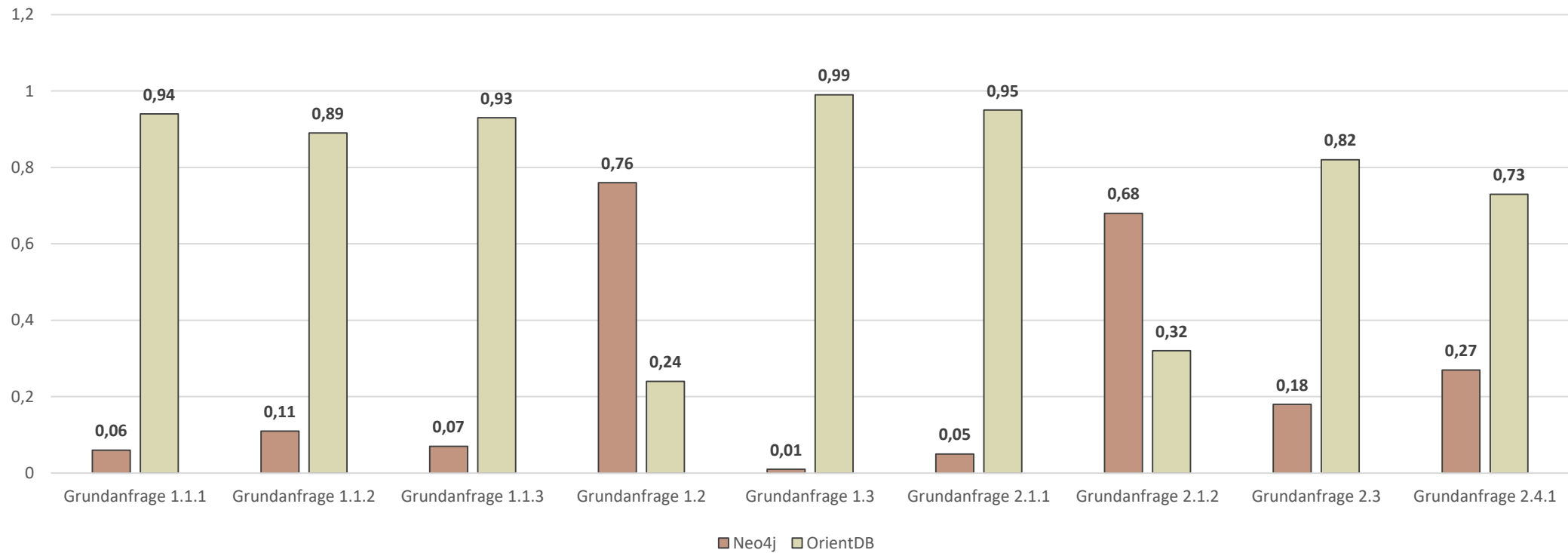


Tabelle 1: Verteilung Grundanfrage 1.3

Anfrage	Grundanfrage	Alternative	Java
Zeit	13,36 ms	8,66 ms	3,2 ms

OrientDB und Neo4j mit Cypher



Grundanfragen 2.1.1-2.1.2

Grundanfrage 2.1.1:

MATCH (p:Person {name: 'Person1'})-[:RELATIONSHIP3*2]->(p1:Person) RETURN COUNT(DISTINCT(p1))

Grundanfrage 2.1.2:

MATCH (p:Person {name: 'Person1'})-[:RELATIONSHIP3*3]->(p1:Person) RETURN COUNT(DISTINCT(p1))

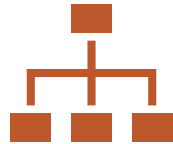
Tabelle 2: Verteilung Grundanfrage 2.1.1-2.1.2

Anfrage	Grundanfrage 2.1.1	Grundanfrage 2.1.2	
Cypher	3501 ms	234312 ms	66,9
OrientDB	72092 ms	110064 ms	1,5
	20,6	0,5	Anstiegsfaktor

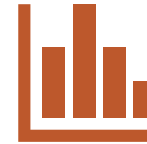
Limitierungen



Begrenzte Anzahl an
Anfragen



Keine Verteilung der
Datenbank



Fehlen von
Referenzstudien



Nur 4 Relationstypen
und 2 Entitätstypen

Fazit zu Neo4j



Verglichen mit OrientDB:

205 mal schneller mit Java und 1,9 mal schneller mit Cypher

2,4-facher Speicherbedarf als OrientDB



Einfach Bedienung durch Cypher



Flexibler Gebrauch durch Java

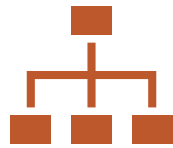


Unterstützt mehrere Sprachen

Zukünftige Arbeit



Vergleich mit reiner
Graphdatenbank



Verteilung des Systems
nutzen



Relationale
Datenbanken betrachten



Verwenden von
GraphQL