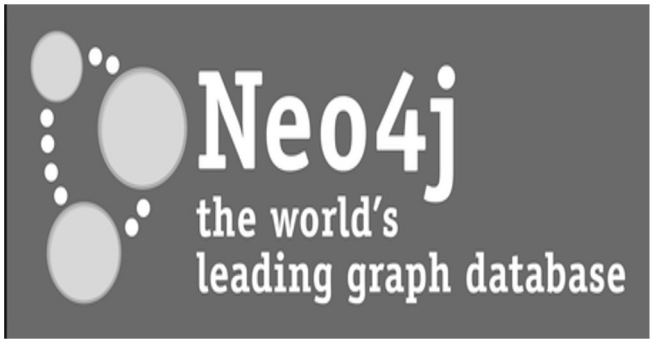
Neo4j, grafni nerelacijski SUPB

- Kaj je Neo4j?
- Grafni nerelacijski SUPB
- Povpraševalni jezik Cypher
- Povezava z aplikacijami (Python)
- Kdaj ga izbrati?

- Namestitev:
  - Prenos (OS in arhitektura sistema)
  - Preprosta namestitev (le specifikacija avtentikacije, kasneje jo lahko izklopite)
  - Neo4j desktop (integrirano okolje za spoznavanje)

Kaj je Neo4j

- Produkt podjetja Neo Technologies
- Najpopularnejši grafni SUPB
- Implementiran v Javi
- Odprtokoden



(www.neo4j.org)

Neo4j - Licenca GPL v3

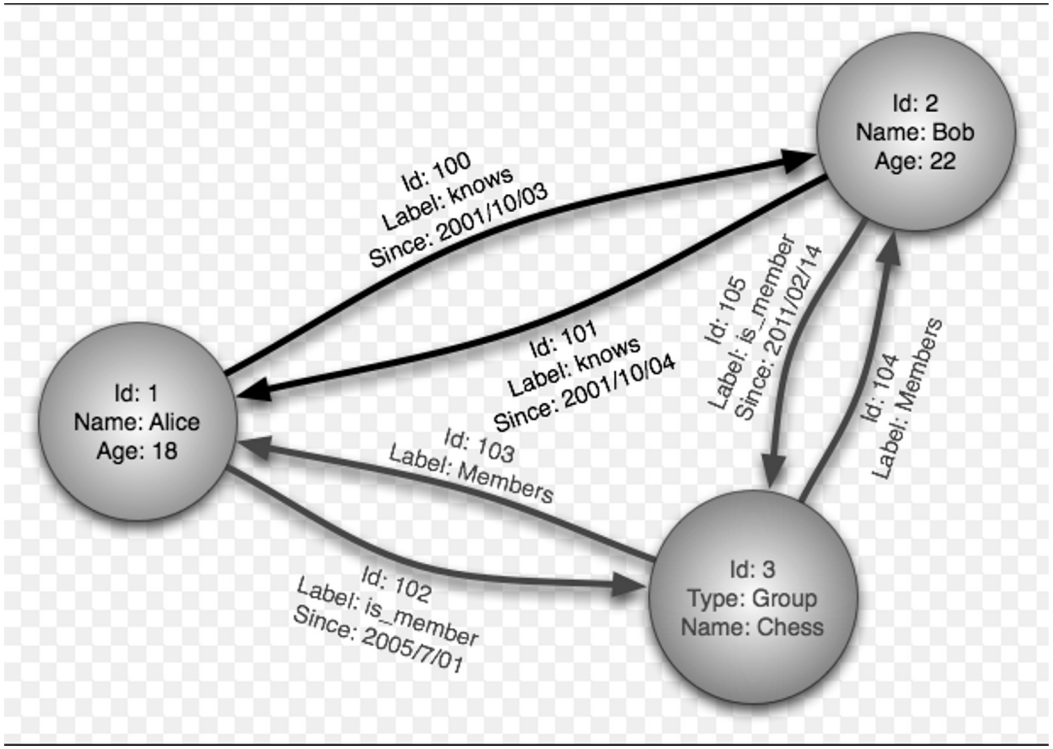
- GNU General Public License version 3
- **Copyleft:** Vsako delo, ki izhaja iz programske opreme z licenco GPL v3, mora biti prav tako licencirano pod GPL v3.
- **Dostop do izvorne kode:** Uporabniki imajo pravico do dostopa do izvorne kode programske opreme.
- **Svoboda spreminjanja:** Uporabniki lahko programsko opremo spreminjajo in prilagajajo svojim potrebam.
- **Svoboda deljenja:** Uporabniki lahko programsko opremo prosto delijo z drugimi, tako v izvorni kot v spremenjeni obliki.
- **Komercialna uporaba**: mogoča, ob upoštevanju licenčnih zahtev

(www.neo4j.org)

Grafni SUPB

- Podatkovni model je graf z lastnostmi (omrežje, "property graph")
  - Podatki se hranijo v vozliščih, povezavah in njihovih dodanih lastnostih
  - To je **podatkovni model** grafnih SUPB
- Vsako vozlišče vsebuje kazalce na svoje naslednike
  - Posledično ne potrebujemo nujno (lahko pa) dodatnih indeksov
- Povezave vsebujejo najpomembnjši del informacije, saj povezujejo
  - Vozlišča z drugimi vozlišči
  - Vozlišča z njihovimi lastnostmi

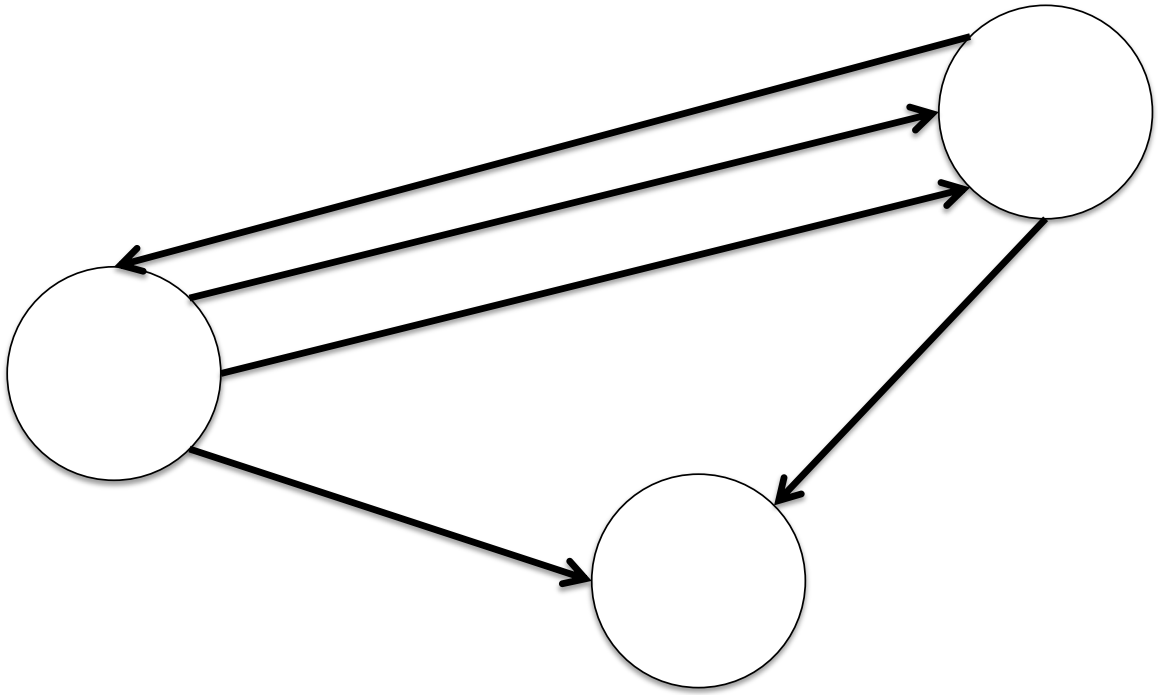
Grafni SUPB – omrežje (podatkovni model)



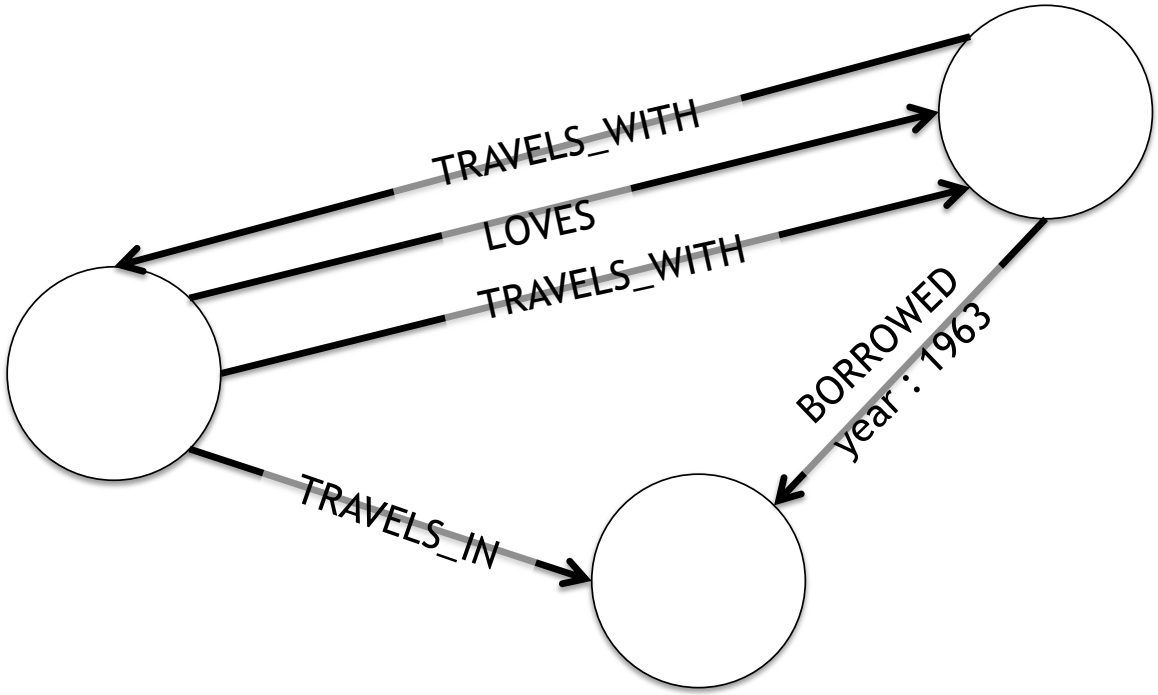
Vozlišča lahko vsebujejo

- Identifikator (id, ime)
- Lastnosti (properties): pari (ime lastnosti, vrednost)
- Oznako ali tip (label/type): pripadnost skupini/tipu vozlišč

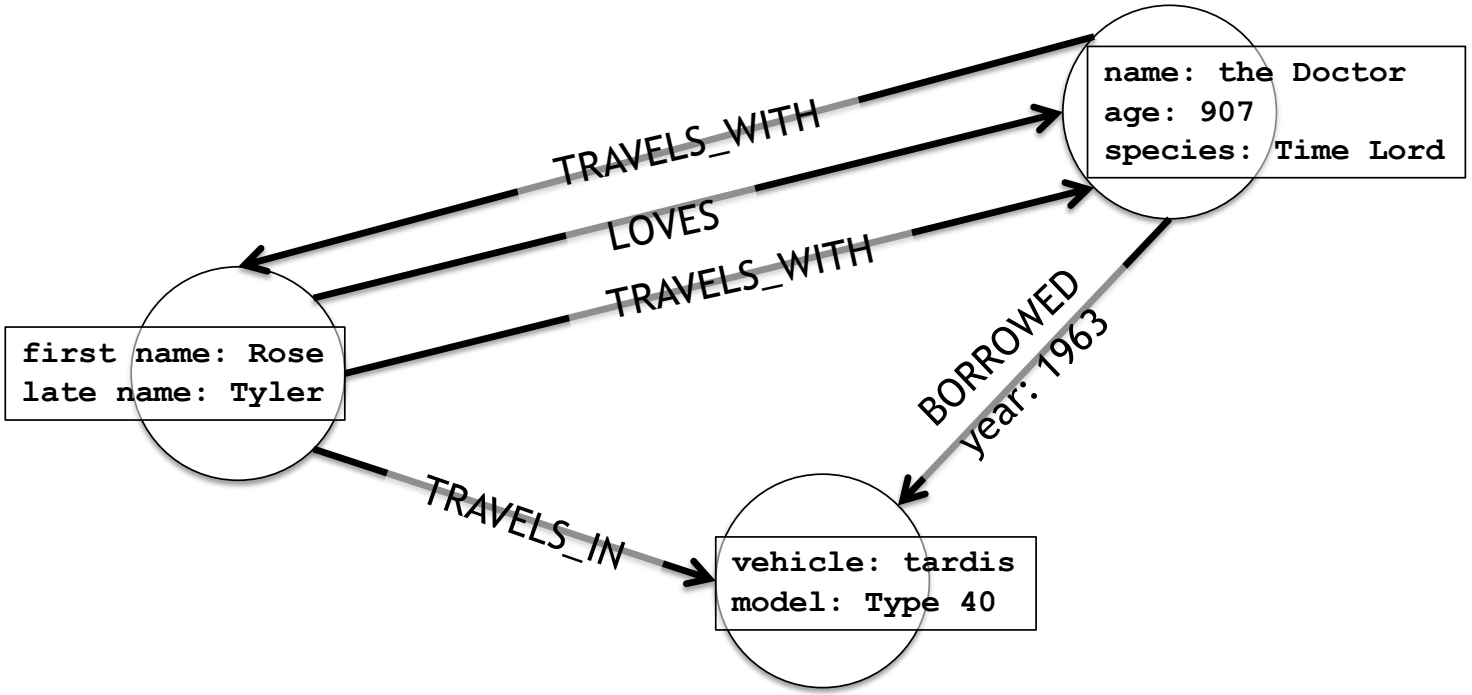
Podatkovni model: omrežje oz. graf z lastnostmi (property graph)



Podatkovni model: omrežje oz. graf z lastnostmi (property graph)

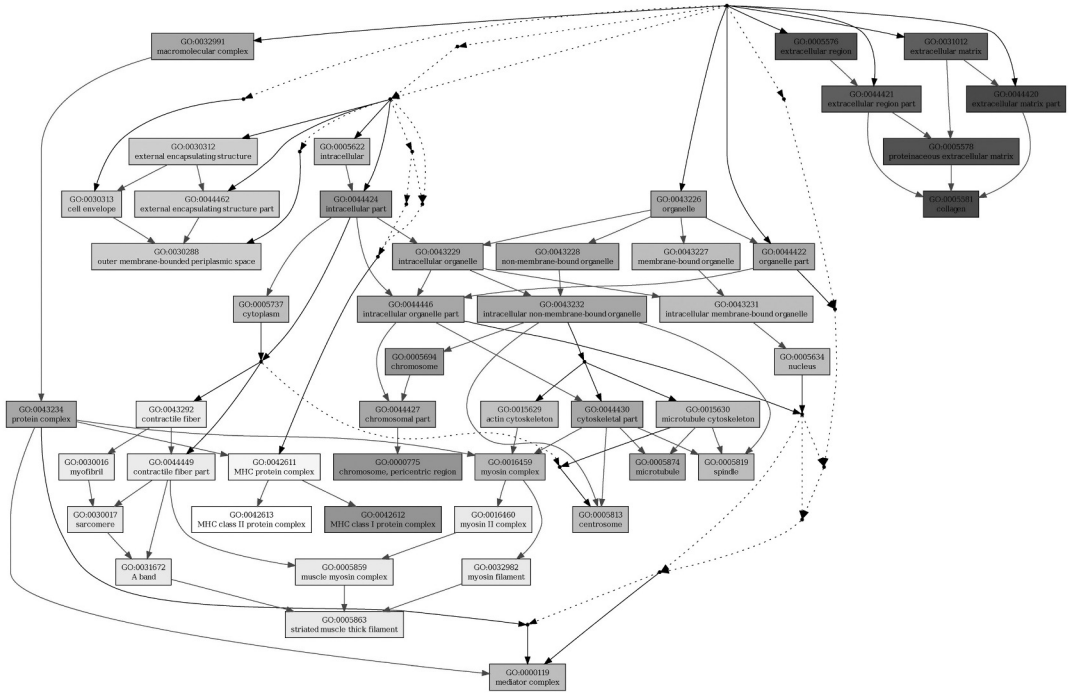


Podatkovni model: omrežje oz. graf z lastnostmi (property graph)



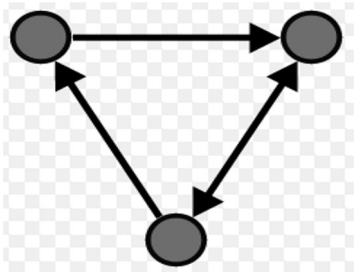
Prednosti grafnih SUPB

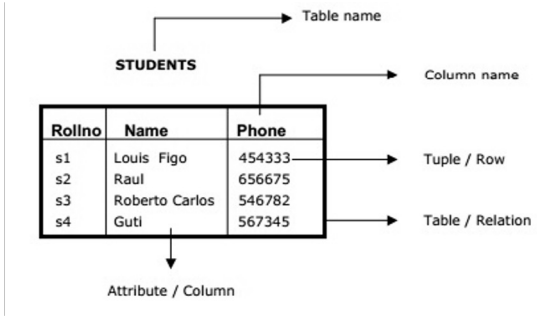
- Če nas zanimajo povezave (razmerja) med entitetami (vozlišči) so grafni SUPB zelo uporabni zaradi svojega podatkovnega modela
- Grafni SUPB so zelo primerne za asociativne podatkovne množice
  - Podobno kot družabna omrežja
- Omogočajo direktno preslikavo objekno usmerjenih struktur in aplikacij
  - Klasifikacija objektov
  - Hierarhije (razmerja starš - potomec)
- **Zelo fleksibilen podatkovni model**

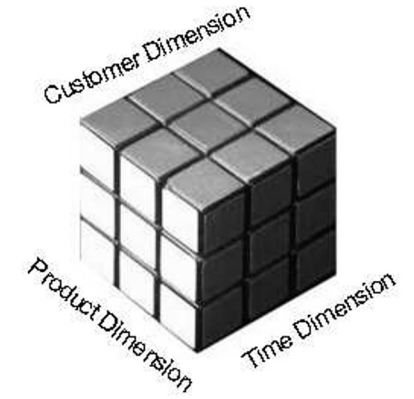


Pomanjkljivosti grafnih SUPB

- Manj primerno za tabelarične podatke z malo razmerji
- Slaba podpora za OLAP in druge večdimenzionalne strukture
  - Problematična uporaba za "poslovno" podatkovno analitiko
  - Aktivno področje, potrebne nadaljnje raziskave in razvoj
- V določenih primerih problematično horizontalno skaliranje





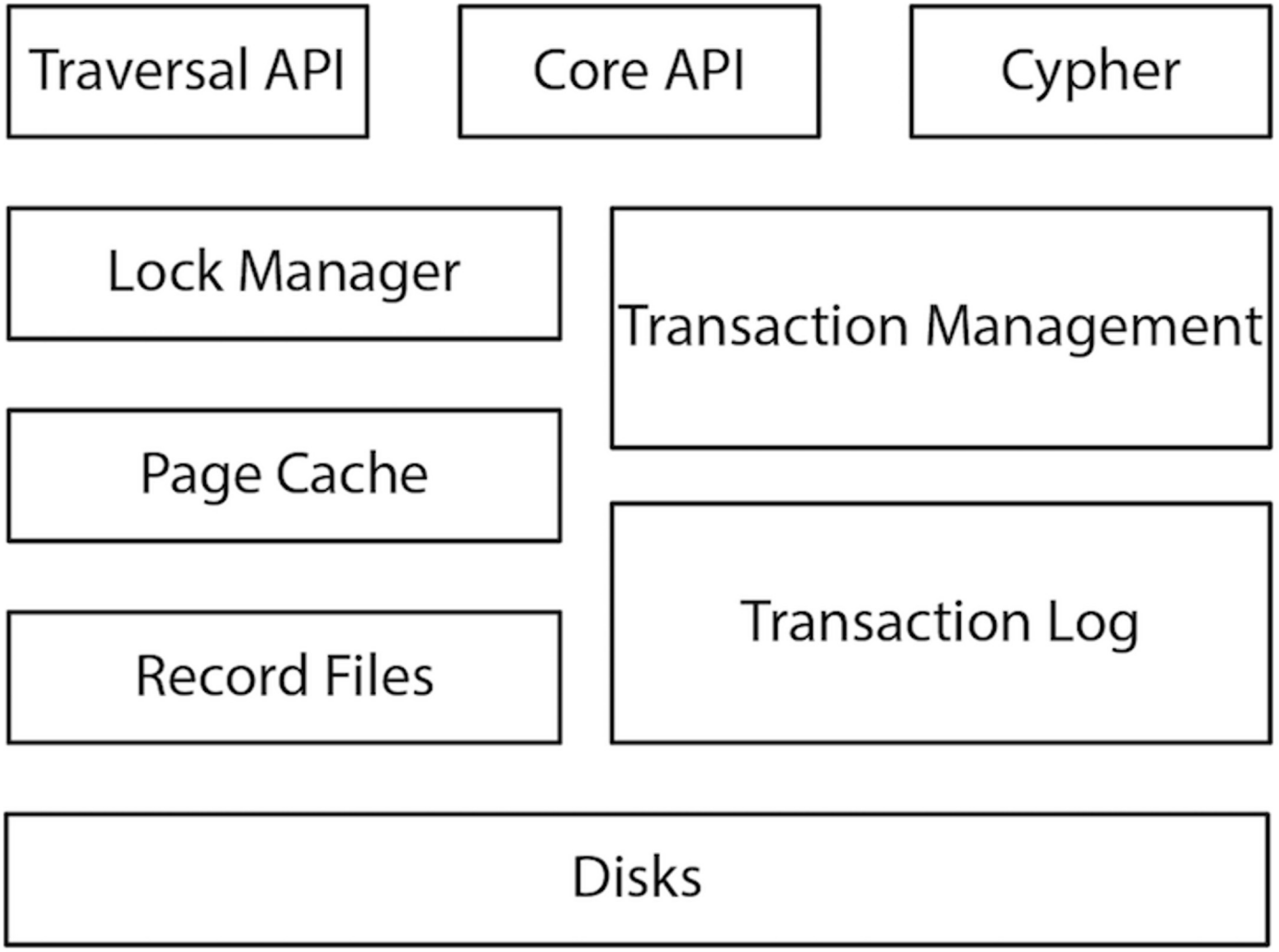


Enostavnost in učinkovitost agregacije

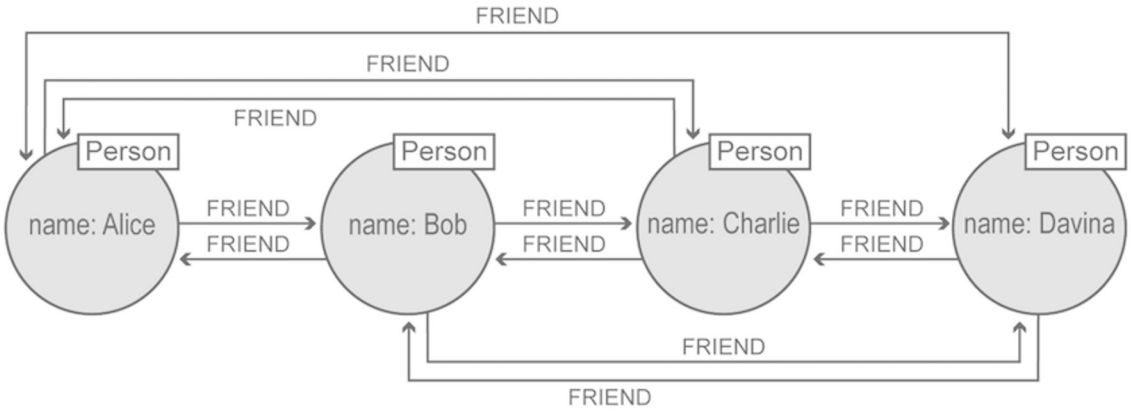
Primerjava Neo4j in relacijskih SUPB

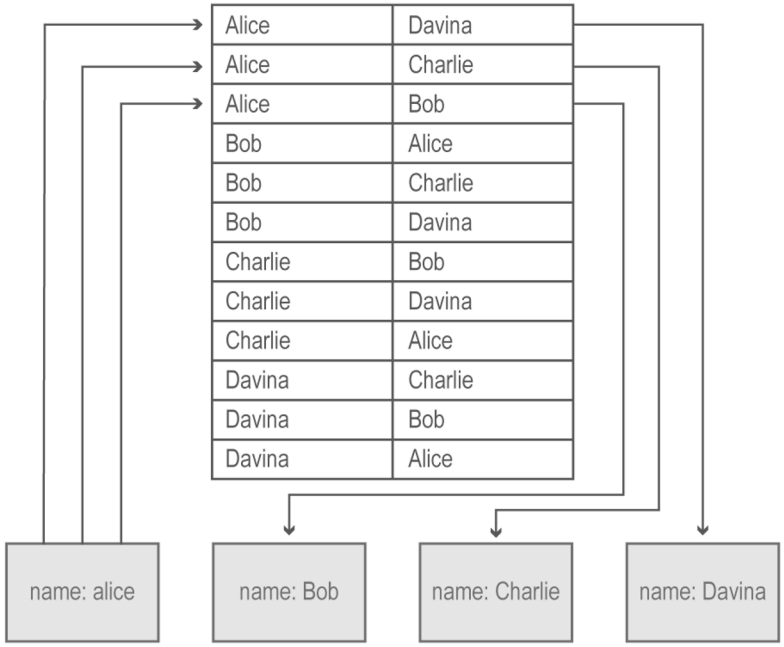
- Nima vnaprej definirane sheme/konvencije, ki se je mora držati
  - Vsako vozlišče/povezava ima lahko poljuben nabor lastnosti
- Podpora ACID transakcijam (logičnim enotam dela)
- Povpraševalni jezik Cypher (nekoliko podoben SQL)
  - Iniciativa openCypher (odprti viri/materiali za implementacije jezika)
  - Del iniciative SQL/PGQ (Property Graph Queries in SQL), integracija v SQL ISO standard SQL/PGQ:2023 (prilagojena podmnožica sintakse Cypher)
  - Velik vpliv na ISO GQL (Graph Query Language) – april 2024
- Enostaven za učenje in uporabo
- Dobra dokumentacija, velika podpora skupnosti
- Podpora sodelovanju z drugimi jeziki
  - Java, Python, Perl, Scala, …

Arhitektura Neo4j



Primerjava predstavitve grafov





- Zgoraj: grafna baza, način sprehajanja po grafu je očiten in preprost
- Levo: relacijska baza, sprehajanje po grafu ni preprosto

Povpraševalni jezik Cypher

- Eden od načinov za povpraševanje v Neo4j (ostalo: REST API , Core API, povezave z zunanjimi jeziki, …)
- Formulacija povpraševanj, ki temelji na razmerjih med podatki
- Specifikacija z vzorci (patterns)
- Primer vzorca (ustvarjanje povezave med konkretnima vozliščema):

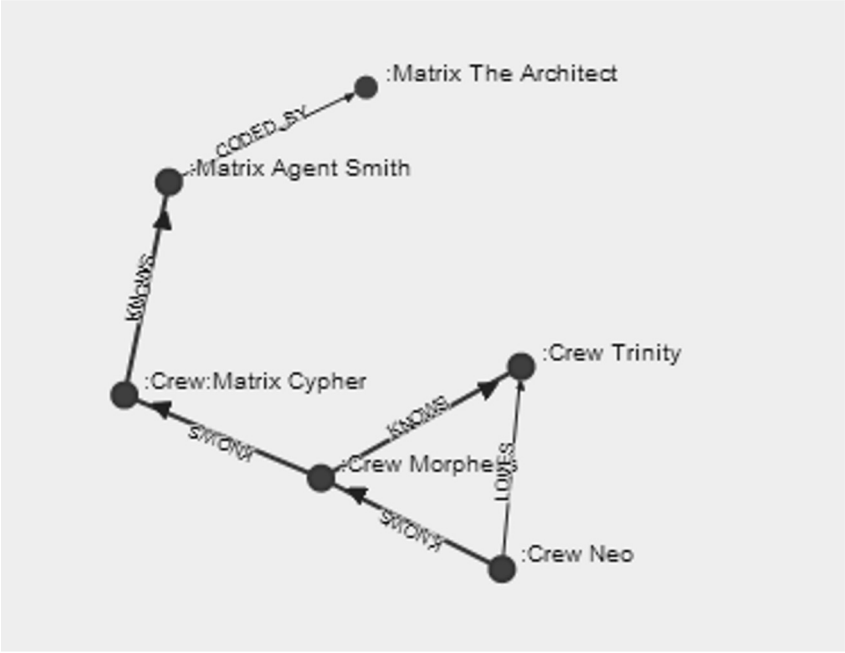
```
MATCH (a:Person { name: 'Ann' }), (b:Person { name: 'Dan' })
    CREATE (a)-[:KNOWS]->(b)
```

- Mnoge značilnosti izhajajo iz želje po odpravi težav relacijskih SUPB (npr. odprava stičnih tabel)

Cypher

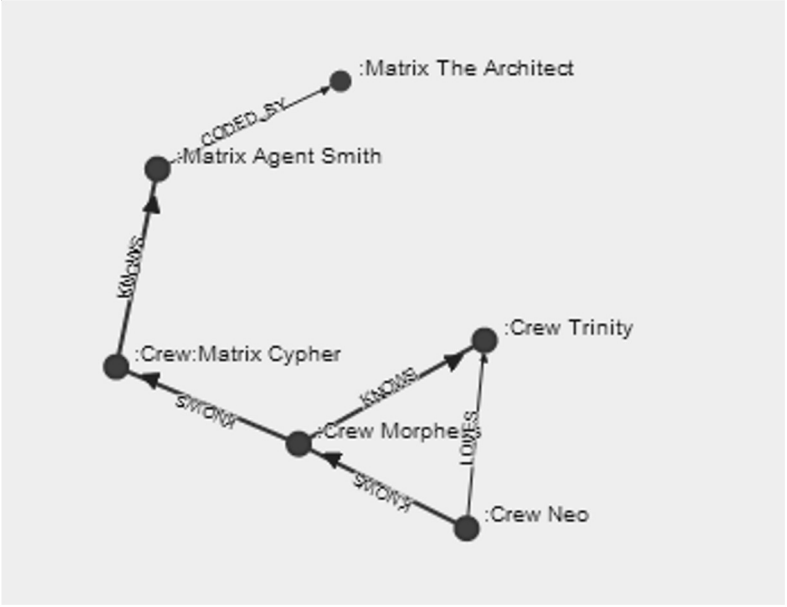
```
CREATE (Neo:Crew { name:'Neo' })
```

```
(Neo)-[:KNOWS]->(Morpheus)
```



Cypher

```
Query:
MATCH (n:Crew)-[r:KNOWS*]-m
WHERE n.name='Neo'
RETURN n AS Neo,r,m
```



Neo	r	m
{name:"Neo"}	[(0)-[0:KNOWS]->(1)]	(1:Crew {name:"Morpheus"})
{name:"Neo"}	[(0)-[0:KNOWS]->(1), (1)-[2:KNOWS]->(2)]	(2:Crew {name:"Trinity"})
{name:"Neo"}	[(0)-[0:KNOWS]->(1), (1)-[3:KNOWS]->(3)]	(3:Crew:Matrix {name:"Cypher"})
{name:"Neo"}	[(0)-[0:KNOWS]->(1), (1)-[3:KNOWS]->(3), (3)-[4:KNOWS]->(4)]	(4:Matrix {name:"Agent Smith"})

Kdo vse uporablja Neo4j?

Tudi:

- Ebay
- LinkedIn
- Walmart
- …

(www.neo4j.org)

Kdaj uporabiti grafni SUPB?

- Ključna vprašanja
  - Ali bomo obravnavali predvsem razmerja?
  - Kakšna bodo poizvedovanje?
- Grafni SUPB Neo4j je med najpogosteje uporabljanimi nerelacijskimi SUPB, popularnejši so le MongoDB (dokumentni), Cassandra (stolpčni), Redis (ključ-vrednost), Hbase (stolpčni).
- Najpopularnejši grafni SUPB, poudarjeni so čisti grafni SUPB, neo4j ima več kot 50% delež

1. **Neo4j**
2. Microsoft Azure Cosmos DB
3. ArangoDB
4. OrientDB
5. Virtuoso
6. **JanusGraph**
7. Amazon Neptune
8. GraphDB
9. **Dgraph**
10. **Giraph**

Neo4j / grafni SUPB in horizontalno skaliranje

- Horizontalno skaliranje grafov in izvajanje grafnih algoritmov je v splošnem problematično!
- Zakaj: tipično visoka povezanost netrivialnih grafov (*six degrees of separation*), zato so potrebne poizvedbe med vozlišči
- Iskanje vzorcev je pogosto možno dobro paralelizirati (neodvisnost)
- Možne rešitve:
  - Popolna replikacija podatkov (grafov) na vseh vozliščih – ni vedno mogoče. Enostavno, vendar prostorsko potratno!
  - Omejevanje globine lokalnih povezav oz. poti (npr. do 3 lokalno, globlje/daljše lahko tudi med vozlišči gruče). Relativno zahteven pristop!
- Neo4j - tipična topologija: porazdeljeno branje, centralizirano zapisovanje, popolna replikacija podatkov

Programski dostop do Neo4j

- Python:
  - neo4j (osnovni, nižjenivojski pristop, Cypher, podobno kot ODBC)
  - py2neo (delno OGM pristop), najenostavnejši
  - neomodels (OGM pristop)
  - …

- OGM: Object-Graph Mapping

Povezava na lokalni Neo4j

```
from py2neo import Graph, Node, Relationship

graph = Graph("bolt://localhost:7687", auth=("neo4j", "neo4jneo4j"))

nicole = Node("Person", name="Nicole", age=24)
mtdew = Node("Drink", name="Mt. Dew", calories=9000)

graph.create(nicole | mtdew)
graph.create(Relationship(nicole, "LIKES", mtdew))
```

Poizvedbe, vzorci, parametri

```
# Poizvedba z vzorcem, parametrom in agregacijo
query = """
        MATCH (p:Person)-[:LIKES]->(drink:Drink)
        WHERE p.name = $name
        RETURN p.name AS name, AVG(drink.calories) AS avg_calories
"""

# Parameter $name dobi vrednost "Nicole"
data = graph.run(query, name='Nicole')
```

Neo4j - literatura

- Ian Robinson, Jim Webber, Emil Eifrém: Graph Databases, 2nd Edition, O'Reilly Media, https://neo4j.com/books/ (free)
- http://www.neo4j.org
- http://www.neo4j.org/learn/cypher

+ mnogo tiskanih in spletnih virov

Vektorske baze – problem, ki ga rešujejo

- Problem večine podatkovnih baz (tudi relacijskih):
  - Neučinkovito vektorsko eksaktno ali približno iskanje
  - KNN (K-nearest neaighbour, K-najbližjih sosedov)

- Imejmo:
  - Množica številskih vektorjev enake dolžine
- Problem:
  - Najti K vektorjev, ki so (po nekem kriteriju) najbolj podobni danemu vektorju $\boldsymbol{v}$

- Rešitev:
  - Specializirane "vektorske" baze

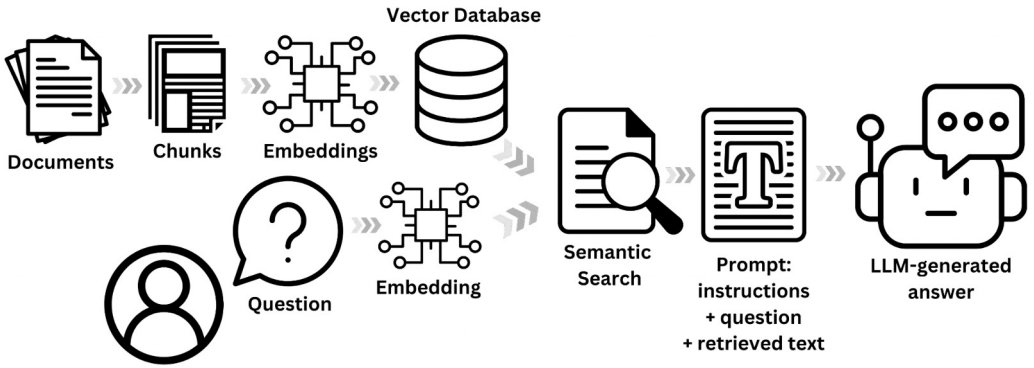
Scenariji uporabe in mere podobnosti

- KNN klasifikator
- Priporočilni sistemi na podlagi podobnosti
- Prilagajanje velikih jezikovnih modelov (LLM), npr. ChatGPT
  - RAG – retrieval augmented generation

- Tipične mere podobnosti:
  - Evklidska razdalja (L2 norma)
  - Kosinusna razdalja (kosinus kota med vektorjema)
  - Pearsonov korelacijski koeficient
  - … in še množica drugih

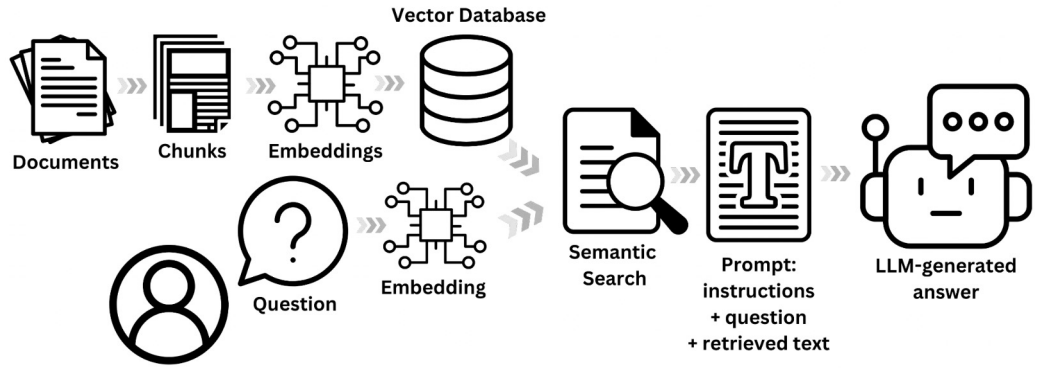
https://rapidfork.medium.com/various-similarity-metrics-for-vector-data-and-language-embeddings-23a745f7f5a7

LLM in RAG



- RAG predstavlja poceni in hitro alternativo prilagajanju (fine-tuning) LLM za lastne potrebe
- Preprečuje/omejuje halucinacije LLM
- Združuje prednosti iskanja in LLM
- LLM-ju predloži (dele) dokumentov (virov), na podlagi katerih nah generira svoj odgovor
- Kako **hitro** in **učinkovito** najti relevantne dokumente/dele dokumentov?

LLM, RAG in vektorske baze



1. Delitev relevantnih dokumentov na primerno manjše dele (chunk), tipično nekaj 100 znakov
2. Opis delov dokumentov z **vložitvami** (embeddings) – dolgi številski vektorji fiksne dolžine, običajno nekaj 100 (razpon 64 do 4096); to naredijo posebej za to namenjeni **vložitveni modeli**
3. Ob vprašanju se generira tudi vložitev vprašanja
4. Sledi **iskanje** delov dokumentov, katerih vložitve (vektorji) so najbolj **podobni** vložitvi (vektorju) vprašanja
5. LLM na koncu generira odgovor na podlagi najdenih dokumentov (ali njihovih delov)

Popularni vektorski SUPB

- Uporaba kot knjižnica, samostojna, ali porazdeljena implementacija
- Specializirane: **Chroma**, Milvus, Qdrant, Pinecone, Weaviate, …
- Razširitve obstoječih SUPB: PostgreSQL (PG Vector), MongoDB (Atlas vector search), Oracle AI vector search, …
- Tipično dobra integracija z orodji umetne inteligence in LLM-ji (ChatGPT, Gemini, Claude, Mistral, **Llama**, …)
- Dobro dodelani recepti in cevovodi za delo
- Več bomo povedali kasneje, ko bomo obravnavali bogate podatkovne tipe in delo z njimi.

(možna je tudi kakšna tema za seminarsko nalogo)