Универзитет у Нишу
Електронски факултет

Интерна структура и организација складиштења података код Neo4J базе података

Семинарски рад
Системи за управљање базама података

Предметни наставник:
Александар Станимировић

Студент:
Ивана Алексић, бр. индекса 1590

Садржај

1. Увод....2
2. Инсталација стандардне Neo4j базе података....2
3. Манипулација базама у систему....5
3.1. Наредба за приказ база у систему....5
3.2. Наредбе за креирање базе....6
3.3. Наредбе за измену базе података....7
3.5. Брисање базе података....8
4. Интерна структура података у Neo4J....8
4.1 Чворови....9
4.2 Везе....9
5. Организација складиштења података у Neo4J....12
6. Пројектовање графовске базе података у односу на релациону....18
7. Literatura....19

1. Увод [1]

Neo4j је база података и систем за управљање базама података (*DBMS*) . Neo4j као систем за управљање базама података, или *DBMS*, је способан за управљање више база података. *DBMS* може управљати самосталним сервером или групом сервера у кластеру. Neo4J као база је графовска база података (Пример је на слици 1) која се користи за управљање и анализу података који су повезани сложеним релацијама. Ова база података користи графовски модел података, што значи да подаци у њој су организирани као графови с чворовима који представљају ентитете и релацијама које повезују те ентитете. База података је административна раздела *DBMS*-а. У практичним терминима, то је физичка структура датотека организованих унутар директоријума или фасцикле, која има исто име као и база података. Neo4j *DBMS* омогућава управљање локалним и удаљеним стандардним базама података, композитним базама података и алијасима базе података. Овај рад ће бити орјентисан стандардним базама података. Neo4J за претраживање и манипулацију подацима користи се *Cypher* језик упита, који је посебно дизајниран за рад с графовским подацима. Дизајниран је да буде високо скалабилан, па се може користити за управљање великим скуповима података. Такође, подржава трансакције и *ACID* (*Atomicity, Consistency, Isolation, Durability*) својства како би осигурао конзистентност и поузданост података. Пружа скуп алата за анализу података, укључујући и подршку за алгоритме за анализу графова и визуелизацију података. Neo4J се издваја по томе што не мора да рачуна везе између података у току упита јер их већ чува, тако да олакшава упите и јако их брзо разрешава.

2. Инсталација стандардне Neo4j базе података [2]

Подразумевана инсталација Neo4j 5 садржи једну стандардну базу података, чије је име neo4j, која је подразумевана база података за *DBMS*. Различито име може бити конфигурисано пре покретања Neo4j-а први пут. У наставку су побројани параметри које је могуће подесити пре инсталације базе:

- **initial.dmbs.default_database** - односи се на име подразумеване базе података за инстанцу Neo4j. База података ће бити креирана ако не постоји када инстанца почне са радом. Подразумевана вредност: neo4j

- **dbms.max_databases** – односи се на максималан број база података које могу бити коришћене у једној Neo4j инстанци или кластеру. Број укључује све онлајн и офлајн базе података. Вредност је целобројна и има минималну вредност од 2. Када се достигне ограничење, није могуће креирати додатне базе података. Исто тако, ако се ограничење промени на број који је мањи од укупног броја постојећих база података, додатне базе података не могу бити креиране. Подразумевана вредност: 100

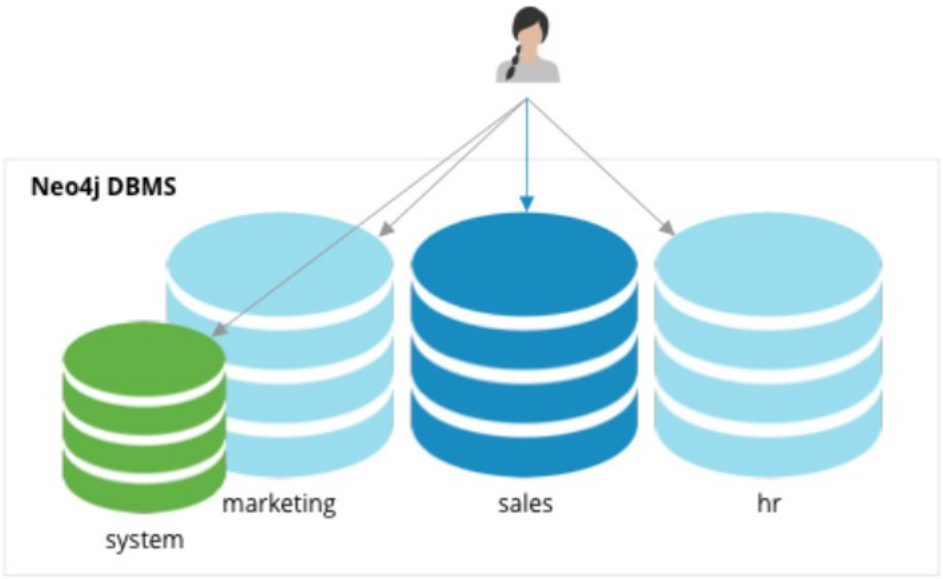
- **server.databases.default_to_read_only** -односи се на подразумевани режим за све базе података. Ако је ова поставка постављена на true, све постојеће и нове базе података биће у режиму само за читање, и тако ће спречити упите за уписивање. Подразумевана вредност: false

- **sarver.databases.read_only** - односи се на списак имена база података за које је потребно спречити упите за уписивање. Овај скуп може садржати и базе података које још увек не постоје, али не и системску базу података. Без обзира на поставке **server.databases.default_to_read_only**, **server.databases.read_only** и **dbms.databases.writ**able, системска база података никада неће бити само за читање и увек ће прихватати упите за уписивање. Још један начин спречавања уписивања је поставити приступ бази података на само за читање користећи Cypher наредбу **ALTER DATABASE**. Пример конфигурације:
  server.databases.read_only=["foo", "bar"]

- **dmbs.databases.writable** – односи се на списак имена база података за које се прихватају упити за уписивање. Овај скуп може садржати и базе података које још увек не постоје. Вредност ове поставке се игнорише ако је **server.databases.default_to_read_only** постављен на false. Ако је име базе података присутно у оба скупа, база података ће бити само за читање и спречиће упите за уписивање. Пример конфигурације:
  dbms.databases.writable=["foo", "bar"]

Иако постоје два приступа у подешавању вредности истог параметра (конфигурацијом и *Cypher* наредбама), важно је разумети разлику између ова два приступа. *Cypher* наредбама параметар добија вредност широм целог *DBMS*-а, док коришћење конфигурације омогућава посебно подешавање за сваку инстанцу. Уколико се користе и *Cypher* наредба и конфигурациони параметри, и садрже различите информације, искористиће се вредност из конфигурације.

Слика 1 илуструје инсталацију Neo4j-а која садржи три стандардне базе података, чија су имена *marketing*, *sales* и *hr*, као и системска база података. Подразумевана база података је *sales*:



Слика 1: Инсталација система база података, са подразумеваном базом

Све инсталације укључују уграђену базу података под именом систем, која садржи метаподатке о *DBMS*-у и конфигурацији безбедности.

Системска база података се понаша различито од свих осталих база података. Над овом базом података може се извршавати само одређени скуп административних наредби, као што су управљање базама података, алијасима, серверима и контролом приступа. Већина доступних административних наредби је ограничена на кориснике са одређеним административним привилегијама.

3. Манипулација базама у систему [1]

3.1. Наредба за приказ база у систему

SHOW DATABASES YIELD name, owner, created_date, modified_date, size, status

name	owner	created_date	modified_date	size	status
system	neo4j	2023-05-15	2023-05-15	50 MB	active
marketing	marketing_team	2023-06-10	2023-06-10	75 MB	active
sales	sales_team	2023-06-12	2023-06-12	80 MB	active
hr	hr_team	2023-06-15	2023-06-16	70 MB	active

У овом примеру, приказ сваеа база података је ограничен на информације о власнику, датуму креирања, датуму последње измене, величини и стању. Постоје многа поља која се могу приказати, као што је alias базе, адреса приступа, привилегије приступа приступ, статус базе, да ли је база подразумевана или не, забране итд.

Резултати наредбе **SHOW DATABASES** се филтрирају у складу са привилегијама приступа корисника. Међутим, неке привилегије омогућавају корисницима да виде додатне базе података без обзира на њихове привилегије приступа:

- Корисници са привилегијама **CREATE/DROP/ALTER DATABASE** или **SET DATABASE ACCESS** могу видети све стандардне базе података.
- Корисници са привилегијама **CREATE/DROP COMPOSITE DATABASE** или **COMPOSITE DATABASE MANAGEMENT** могу видети све композитне базе података.
- Корисници са привилегијама **DATABASE MANAGEMENT** могу видети све базе података.

Ако кориснику нису додељене привилегије приступа ни једној бази података нити било који од горе наведених специјалних случајева, наредба се ипак може извршити, али ће вратити само системску базу података, која је увек видљива.

3.2. Наредбе за креирање базе

Креирање базе података се врши следећом наредбом

CREATE DATABASE customers

Приказ резултата: **SHOW DATABASES YIELD** name

```
+-------------+
| name        |
+-------------+
| "customers" |
| "marketing" |
| "hr"        |
| "system"    |
| "sales"     |
+-------------+
```

Наредба **CREATE DATABASE** опционо је идемпотентна, са подразумеваним понашањем да ће фалирати са грешком ако база података већ постоји. Постоје два начина да се обиђе ово понашање. Прво, додавање **IF NOT EXISTS** наредби осигурава да не буде враћена грешка и да се ништа не деси уколико база података већ постоји:

CREATE DATABASE customers **IF NOT EXISTS**

Друго, додавање **OR REPLACE** наредби резултује у брисању сваке постојеће базе података и креирању нове.

CREATE OR REPLACE DATABASE customers

Ова наредба долази уз разне опције као што је подешавање начина обраде постојећих података на диску при креирању базе података, постваљање сервера који ће се користити за постављање података креиране базе података итд.

3.3. Наредбе за измену базе података

Наредба за измену базе је **ALTER DATABASE**. Пример коришћења измене привилегија приступа бази:

ALTER DATABASE customers **SET ACCESS READ ONLY**

Након извршене наредбе, промену можемо видети на следећи начин:

SHOW DATABASES yield name, access

Резултат:

```
+---------------------------+
| name        | access      |
+---------------------------+
| "customers" | "read-only"  |
| "movies"    | "read-write" |
| "neo4j"     | "read-write" |
| "system"    | "read-write" |
+---------------------------+
```

3.4. Стопирање и покретање базе података

Наредба **STOP** се користи да би се привремено зауставила база података. Након извршене наредбе, онемогућава се приступ бази података. База остаје на серверу, спремна да се покрене поново.Ова функционалност је корисна у ситуацијама када је потребно привремено зауставити рад базе података ради одржавања, ажурирања или других административних задатака или ради заштитите од неовлашћеног приступа:

STOP DATABASE customers
SHOW DATABASE customers **YIELD** name, requestedStatus, currentStatus

```
+-----------------------------------------------+
| name        | requestedStatus | currentStatus |
+-----------------------------------------------+
| "customers" | "offline"       | "offline"     |
+-----------------------------------------------+
```

Покретање базе се врши следећом командом:

START DATABASE customers
SHOW DATABASE customers **YIELD** name, requestedStatus, currentStatus

```
+-----------------------------------------------+
| name        | requestedStatus | currentStatus |
+-----------------------------------------------+
| "customers" | "online"        | "online"      |
+-----------------------------------------------+
```

3.5. Брисање базе података

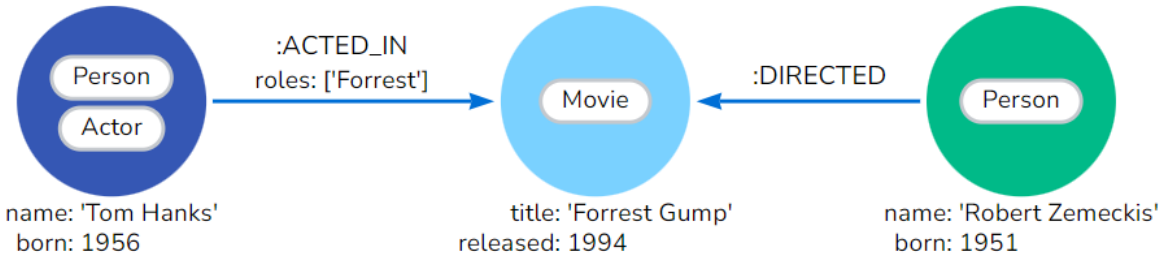
Брисање се врши коришћењем следеће наредбе:

DROP DATABASE customers

Као и у случају креирања базе, брисање базе може бити идемпотентна, са подразумеваним понашањем да ће се неуспех манифестовати грешком уколико база података не постоји. Додавање **IF EXISTS** наредби осигурава да никаква грешка не буде враћена и ништа се не дешава уколико база података не постоји. Увек ће бити враћена грешка ако постоји алијас који је усмерен ка бази података. У том случају, алијас мора бити уклоњен пре него што се уклони база података.

4. Интерна структура података у Neo4J [1]

У Neo4J бази, подаци су организовани у чворове који могу бити повезани. Чворови описују ентитете (дискретне објекте) домена. Могу имати нула или више ознака за дефинисање (класификацију) типа чворова који представљају. Везе описују везу између изворног чвора и циљног чвора. Везе увек имају правац (један правац) и морају имати тип (један тип) за дефинисање типа веза које представљају. Чворови и везе могу имати својства (парове кључ-вредност), који их додатно описују.



Слика 2: Пример графа

Креирање графа са слике 2 у *Cypher* – у:

CREATE

(:Person:Actor {name: 'Tom Hanks', born: 1956})
-[:ACTED_IN {roles: ['Forrest']}]→
(:Movie {title: 'Forrest Gump', released: 1994})
←[:DIRECTED]-
(:Person {name: 'Robert Zemeckis', born: 1951})

4.1 Чворови

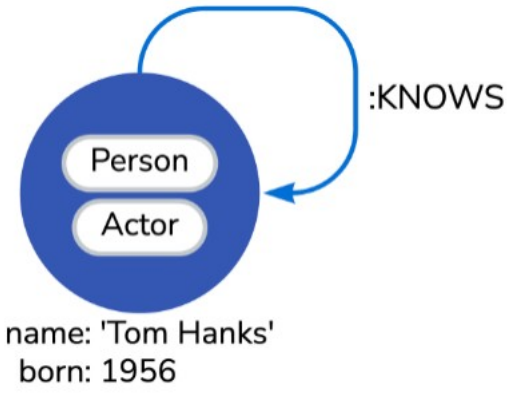
Чворови у графовској бази описују ентитете домена. Најједноставнији случај графа је граф који чини један чвор који нема веза. Ознаке обликују домен груписањем чворова у скупове где сви чворови са одређеном ознаком припадају истом скупу.

На пример, сви чворови који представљају кориснике могу бити означени ознаком "Корисник". На тај начин, може се тражити од Neo4j-а да изврши операције само над чворовима корисника, као што је налажење свих корисника са датим именом.

Ознаке се могу додавати и уклањати током извршавања програма, такође могу бити коришћене за означавање привремених стања за чворове. Ознака "Суспендован" може се користити да означи банкарске рачуне који су суспендовани, а ознака "Сезонски" може означавати површе које је тренутно у сезони. Чвор може имати нула или више ознака, а оне могу да садрже и метаподатке као што су су информације о индексу или ограничењу .

4.2 Везе

Веза описује како су повезани изворни чвор и циљни чвор. Могуће је да чвор има везу са самим собом (Приказано на слици 3). Овква веза се користи када постоји потреба да се покаже да чвор утиче сам на себе, што је корисно код рекурзивних структура или самореференцијалних информација (Приказано на слици 3).



Слика 3: Пример везе чвора са самим собом

Креирање графа са слике 3 у *Cypher* – у:

CREATE

(:Person:Actor {name: 'Tom Hanks', born: 1956})

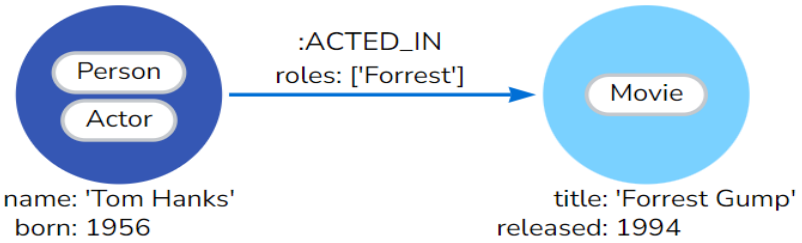
-[:KNOWS]→

(:Person:Actor {name: 'Tom Hanks', born: 1956})

Везе имају један правац и један тип којим се веза класификује, а могу имати и парове кључ-вредност који је додатно описују. Смер везе се може игнорисати када није користан. Ово значи да нема потребе да се везе дуплирају у супротном смеру осим ако није потребно да се исправно опише модел података.

Иако је веза увек усмерена, могућа је ефикасна навигација кроз граф у било ком смеру због уграђених бидирекционих својстава везе. Везе организују чворове у структуре, омогућавајући графу да личи на листу, стабло, мапу или комплексни ентитет - било који од ових може бити комбинован у још комплексније, богато повезане структуре. Број веза једног чвора не утиче на перформансе.

Пример на слици 4 приказује везу „Глумио у“, а додатни описи везе су улога и изведба.



Слика 4: Тип везе - „Глумио у"

Креирање графа са слике 4 у *Cypher* – у:

CREATE

(:Person:Actor {name: 'Tom Hanks', born: 1956})

-[:ACTED_IN {roles: ['Forrest']}]→

(:Movie {title: 'Forrest Gump', released: 1994})

Тип везе казује да је чвор са ознаком ,,Том Хенкс“ изворни чвор, а циљни чвор има ознаку ,,Форест Гамп“. Такође, може се приметити чвор ,,Том Хенкс“ има излазну везу, а чвор ,,Форест Гамп“ има улазну везу.

Својства су парови кључ-вредност који се користе за чување података на чворовима и везама. Могу имати различите типове података, као што су број, низ или *boolean* вредност.

Својства чвора и везе су раздвојени како би се третирали на различите начине. Важно је раздвојити вредност и референцу чворова како би се обезбедило оптимално складиштење и приступ за оба.

5. Организација складиштења података у Neo4J [3]

Датотеке базе података у Neo4j се чувају на диску због трајности. Датотеке везане за податке из базе налазе се у директоријуму **data/databases/graph.db (в3.x+)** према задатим подешавањима у Neo4j-овом директоријуму за податке. Врсте датотека које се налазе на овој локацији са префиксом neostore.* и које чувају:

- **nodestore** - Чува податке везане за чворове графа
- **relationship** - Чува податке о везама у графу
- **property** - Чува својства кључ/вредност из графа
- **label** - Чува податке о индексу ознака из графа

Пошто је Neo4j база података без шеме, користе се фиксне дужине записа за чување података и прате се офсети у овим датотекама како би било могуће вратити податке као одговор на упите. У наставку је приказана табела фиксних величина које Neo4j користи за врсту Java објеката који се чувају:

```
Store File                          | Record size   | Contents
------------------------------------------------------------------------------
----------------------------------------
neostore.nodestore.db               | 15 B          | Nodes
neostore.relationshipstore.db       | 34 B          | Relationships
neostore.propertystore.db           | 41 B          | Properties for nodes and
relationships
neostore.propertystore.db.strings   | 128 B         | Values of string properties
neostore.propertystore.db.arrays    | 128 B         | Values of array properties
Indexed Property                    | 1/3 * AVG(X)  | Each index entry is
                                                      approximately 1/3 of the
                                                      average property value size
```

Сви чворови у бази података се складиште у датотеци за складиште чворова. Сваки запис чвора заузима фиксних 15 бајтова. Распоред је следећи:

- Прва бајт — *isInUse* (да ли се користи)
- 2.-5. бајт — *ID* чвора
- 6. бајт — *ID* прве везе
- 7. бајт — *ID* првог својства
- 8. - 12. бајт — Складиште ознака
- Преостали бајтovi — за будућу употребу

Прvi бајт се користи за одређивање да ли је запис у употреби или је избрисан. Ако није, запис ће бити коришћен за новије уносе. Следећа 3 сектора чине *ID*-јеви самог чвора, прве везе, првог својства и складишта ознака. Неке од ознака се чувају у самом чвору, ако је то могуће, ради смањења скокова. Остали бајтови се чувају за будућу употребу.

Сваки запис везе је фиксни запис од 34 бајта. Распоред записа везе је следећи:

- *ID* почетног чвора
- *ID* завршног чвора
- Показивач на тип везе
- Показивач на следећи и претходни запис везе за сваки од почетног чвора и завршног чвора

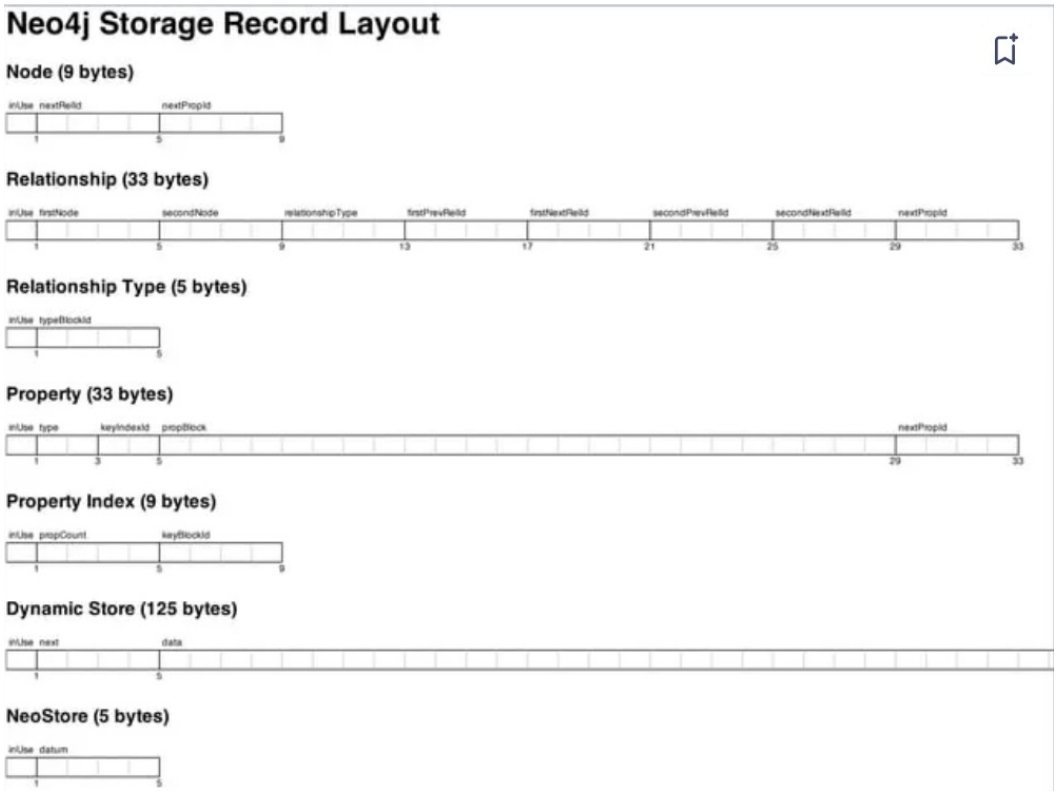
Сваки запис везе деле два чвора, почетни и завршни чвор. Свака веза такође има придружен тип, који означава који тип везе повезује та два чвора. Показивач на тип везе помаже у одређивању овог типа.

Запис везе садржи још 4 показивача или неправилности на записе веза. Они показују на претходни и следећи запис везе и на почетни и на завршни чвор, слично као како се понашају двоструко повезане ланчане листе.

Neo4j користи стабла за обезбеђивање могућности индексирања за достизање почетног чвора одакле се може почети пролазак кроз граф.

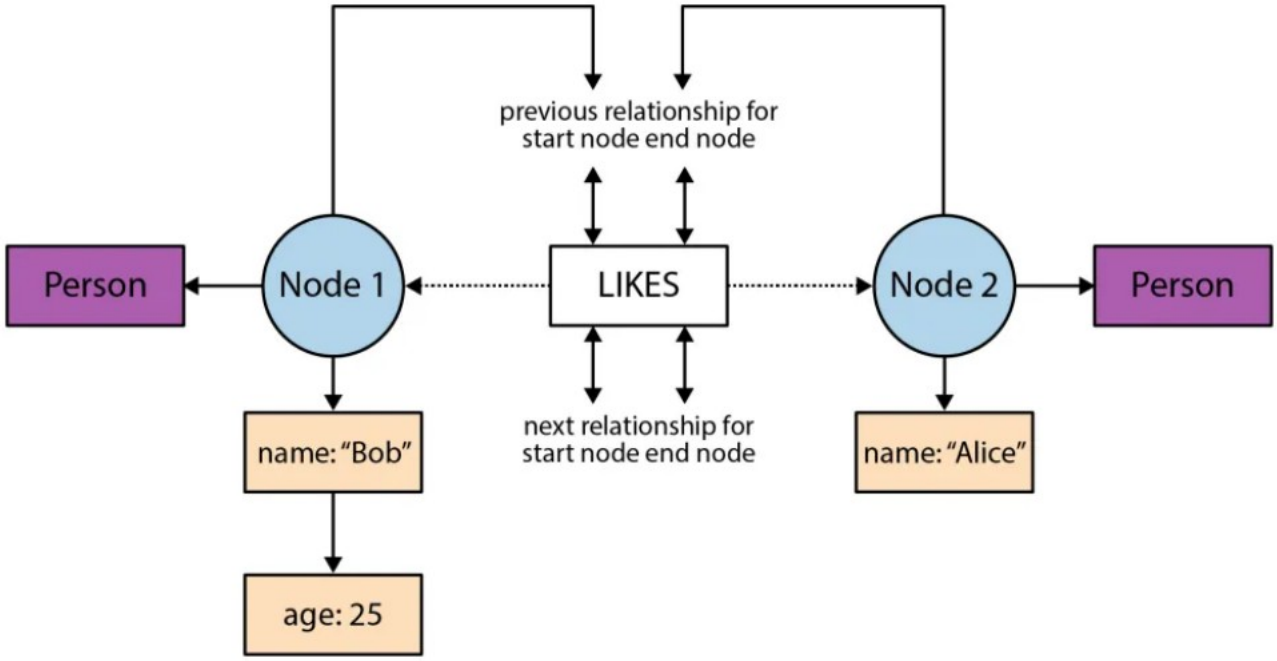
Да би се стигло до одговарајућег чвора, мора се итерирати кроз повезану ланчану листу веза од почетног чвора, итерирати док се не пронађе одговарајући запис везе и затим применити формулу за налажење одговарајућег чвора из записа везе.

Складиште својстава и складиште ознака су једноставна складишта, слична складишту чворова. Запис својстава је величине 32 бајта и подељено је на четири блока од по 8 бајтова. Свако поље може да задржи или кључ или вредност, или и кључ и вредност. Кључ и тип заузимају 3,5 бајтова (кључ 4 бита, тип 24 бита). Boolean, byte, short, int, char, float, small long се уклапају у преостали простор истог блока, big long или double се чувају у засебном блоку (што значи да ће тај запис користити два блока). Референца ка складишту string-ова или низова се чува у истом блоку као кључ (приказано на слици 5).



Слика 5: Складиште својстава

Сви подаци који се чувају на диску су ланчане листе чији су чворови фиксних величина. Својства се чувају као ланчана листа записа својстава, сваки од њих садржи кључ и вредност и показује на следеће својство. Сваки чвор и веза имају референцу на свој први запис својства. Чворови такође имају референцу на прву везу у њиховом ланцу веза. Свака веза има референцу на свој почетни и крајњи чвор. Такође има референцу на претходни и следећи запис веза за почетни и крајњи чвор, респективно, приказано на слици 6.



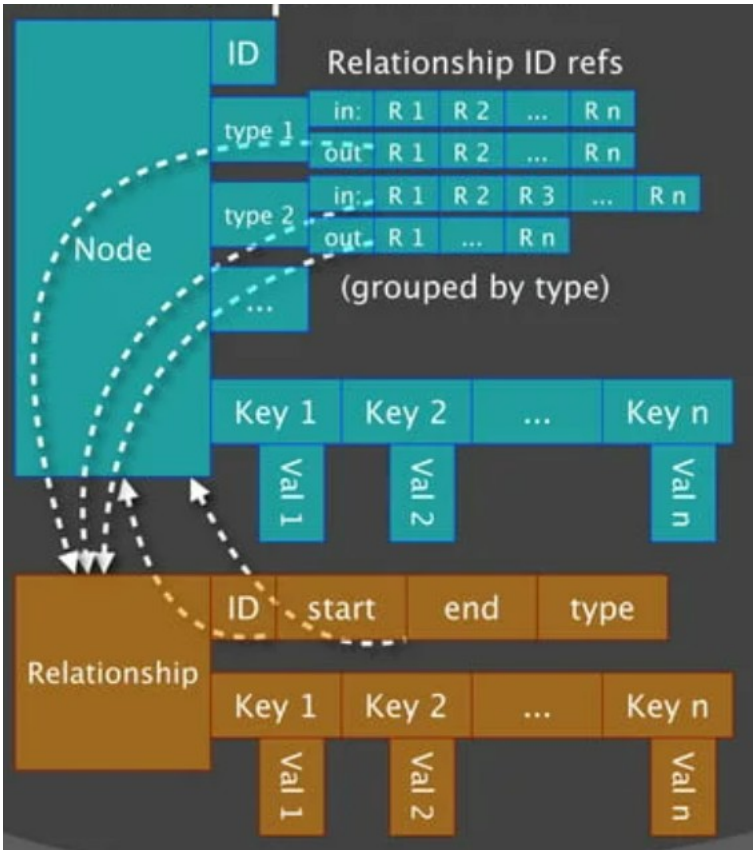
Слика 6: Приказ орīанизације складиштења īодаīака

Фиксни записи и пролазак кроз граф помоћу показивача уместо скенирања табеле користећи индексе, воде ка много бржем и ефикаснијем начину налажења веза.

Када се пронађе запис почетног чвора, користећи први *ID* везе, може се пронаћи веза у складишту веза множењем *ID*-а са величином записа везе. Може се добити запис везе, а из њега се проналази други чвор у складишту чворова користећи сличну формулу.

Neo4J користи два типа кеша:

- **Кеш фајл система (Filesystem Cache)** – сваки сачувани фајл се дели на регионе једнаке величине. У кеш меморији се чува фиксан број региона сваког фајла, а региони се избацују из кеша користећи политику засновану на најмање коришћеним региону. Када кеш достигне капацитет, региони који се најмање користе се избацују како би направили простор за нове регионе који требају да буду кеширани. Оперативни систем се користи за мапирање делова меморије у кешу, што значи да се меморијски ресурси који се користе за кеширање додељују или резервишу на начин који је директно повезан са механизмом управљања меморијом оперативног система. Ово може укључити коришћење функција оперативног система за мапирање делова меморије у виртуелни адресни простор процеса који користи базу података Neo4j. Оваква имплементација омогућава ефикасно коришћење оперативне меморије и управљање ресурсима на нивоу оперативног система за кешовање података у Neo4j. Већина информација на диску се налази у записима о везама, према којима се само референцирају чворови. У кешу, ово се обрће: чворови чувају референце на све своје везе. Везе су једноставне, чувају само своја својства. Везе за сваки чвор су груписане по типу везе да би омогућиле брзо претраживање одређеног типа. Све референце (приказане испрекиданим стрелицама) су по *ID*, и проласком кроз граф врше индиректно претраживање кроз кеш (приказано на слици).



Слика 7: Функционисање кеша фајл система

- **Кеш чворова/веза (Node/Relationship Cache)** - кеш механизам који је специјално дизајниран да би био ефикаснији за операције проласка кроз граф, при чему се обрађују или извлаче подаци из чворова и/или веза у складу са задатим условима. Пролазак кроз граф се дели на два дела, горњи слој – слој са којим интерагујемо и логику испод њега. У горњем слоју, користе се:
  - Проширивачи веза (**RelationshipExpanders**) који одређују које везе ће бити истраживане за наставак проласка кроз граф. На основу чвора у претходној стази, проширивачи веза одређују које везе треба додати у следећи корак.
  - Оцењивачи (**Evaluators**) који одређују да ли треба наставити траверзалу на тренутној грани или не, као и да ли треба укључити чвор у резултатни скуп или не. Они могу донети одлуку на основу својства чвора или стазе.

  Пролазак се завршава када се достигне одређени крајњи чвор или услов. Пролазак кроз чворове може бити подвргнут различитим стратегијама за управљање путањама, одабиром веза и оцењивањем стања чворова. Ова техника је основа за многе операције тражења и анализе у Neo4j бази података. Алгоритам претраге:

1. Кренути од почетног чвора.
2. Провери да ли је податак о чвору у кешу:
   1. Ако јесте, преузети га из кеша.
   2. Ако није, преузети га из складишта и сачувати у кеш.
3. Добити везе из кешираног чвора:
   1. Ако нису преузете, преузети их из складишта.
4. Проширити везе до следећих чворова на основу дефинисаних услова:
   1. Продужити процес ако су испуњени одређени критеријуми.
   2. Укључити чвор у резултатни скуп ако су испуњени одређени услови.
5. Поновити процес за сваки од добијених следећих чворова.
6. Завршити када су сви чворови обрађени или када је достигнут крајњи услов за завршетак.

6. Пројектовање графовске базе података у односу на релациону [1]

Графовске базе су се показале као погодне за моделовање веза и односа између ентитета. Ова флексибилност омогућава природно представљање комплексних структура података, као што су друштвене мреже, мреже препорука, путнички руте, и многе друге. У графовском моделу, упити који траже повезане податке обично су бржи и ефикаснији него у релационском моделу. Ово је посебно важно у сценаријима где су везе између података од кључног значаја за анализу или операције. Структура података се може лако мењати и прилагођавати без потребе за комплексним операцијама као што су додавање или брисање табела и колона у релационском моделу.

Паралела у пројектовању графовске базе у односу на релациону:

- Табела се слика у ознаку чвора - свака ентитетска табела у релационом моделу постаје ознака на чворовима у графичком моделу.
- Ред се слика у чвор - сваки ред у ентитетској табели у релационом моделу постаје чвор у графу.
- Колона се слика у својство чора - колоне (поља) на релационским табелама постају својства чворова у графу.
- Уклањају се технички примарни кључеви, а задржавају се пословни примарни кључеви.
- Додају се јединствена ограничења за пословне примарне кључеве
- Додају се индекси за честе атрибуте претраге
- Страни кључеви се сликају у везе – замењују се страни кључеви ка другој табели везама

7. Literatura

[1] Neo4j

[2] Memory-configuration

[3] Neo4j-storage-internals

[4] How-neo4j-stores-data-internally

[5] Overview-of-neo4j-internals