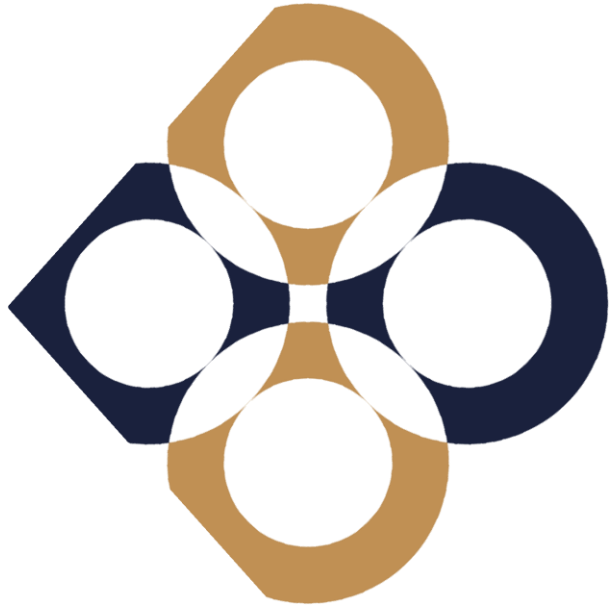
BUDAPESTI CORVINUS EGYETEM

Adatbázisok előadás 09

Gráf adatbázisok



BUDAPESTI CORVINUS EGYETEM

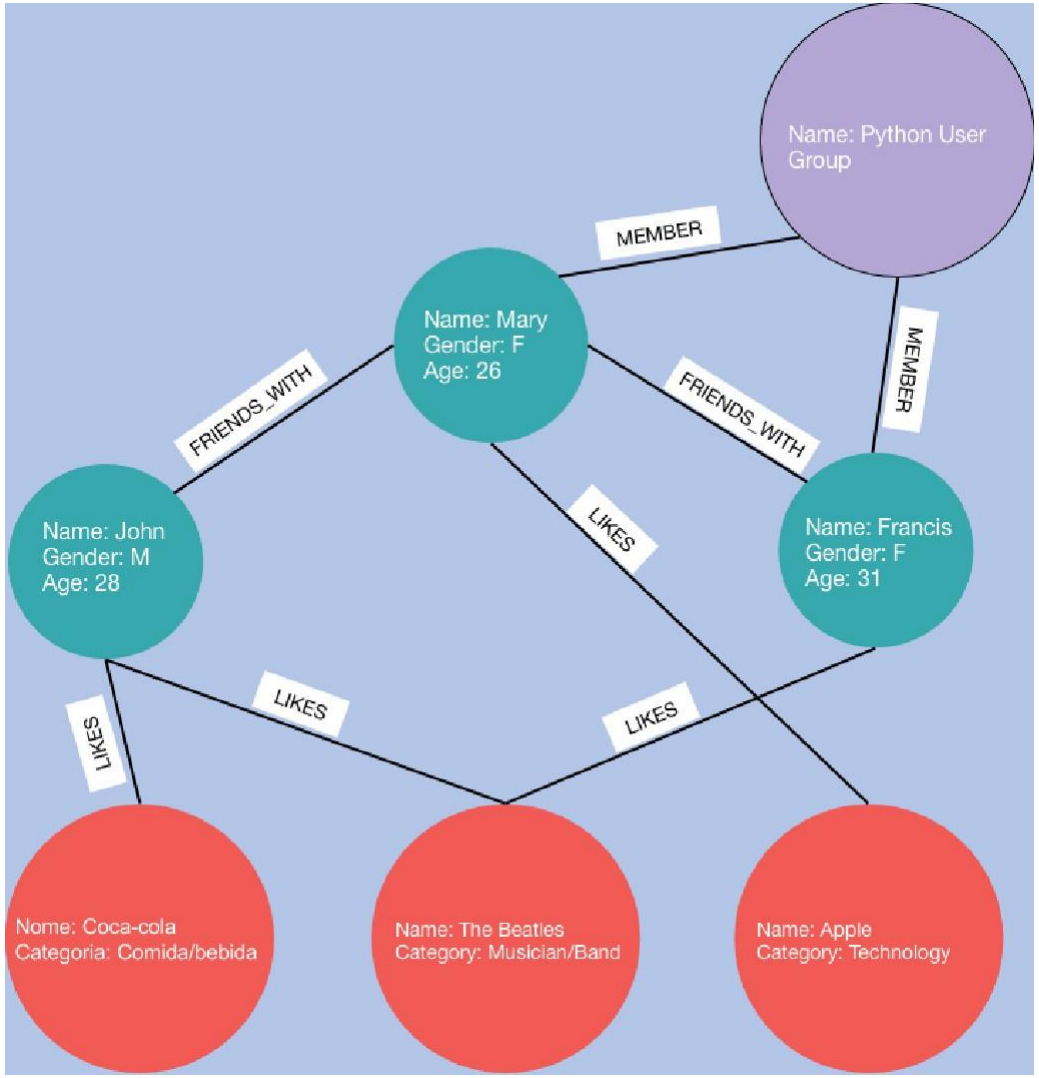
Miről lesz szó?

- Gráf adatbázisok jellemzői
- Neo4J adatbázis
  - Jellemzők
  - Lekérdezések – Cypher nyelv
  - CRUD műveletek
  - Indexek
  - Terminál
  - Elérés Python-ból

Gráf adatbázisok

Olyan adatbázisok, amelyek az adatok tárolására és megjelenítésére gráf struktúrát alkalmaznak

- A gráf csúcsaiban vannak az adatok
  - Az adatok sémája nem rögzített
- A gráf élei jelentik a kapcsolatokat
  - Az élek irányítottak
  - Az éleknek adott név a kapcsolatra jellemző



https://medium.com/labcodes/graph-databases-talking-about-your-data-relationships-with-python-b438c689dc89

Gráf adatbázisok – előnyök és hátrányok

Előnyök	Hátrányok
• Flexibilis séma • Logikus, jól érthető lekérdezések • Gyors adatelérés • Nagymértékben összefüggő adatok kezelése • Sokféle feladathoz megfelelők	• OLTP rendszerekhez nem a legjobbak • A nagy adatmennyiséget érintő lekérdezések nem optimálisak • Sok esetben egy szerveren tárolódnak

BUDAPESTI CORVINUS EGYETEM

Gráf adatbázisok – hol használják őket?

- Tudásbázisok
- Csalás felderítés
- Termék ajánló rendszer
- Social media
- Törzsadatok kezelése
- Hálózati infrastruktúra monitorozás

BUDAPESTI CORVINUS EGYETEM

Gráf adatbázisok vs. Relációs adatbázisok

Gráf adatbázisok	Relációs adatbázisok
Node	Tábla
Nincs séma	Fix séma
A kapcsolatok direkt módon definiáltak	A kapcsolatok idegen kulcsokkal valósulnak meg
A kapcsolódó adatok megjelenítése minták segítségével	A kapcsolódó adatok megjelenítése JOIN-okkal

BUDAPESTI CORVINUS EGYETEM

Gráf adatbázisok vs. Dokumentum adatbázisok

Gráf adatbázisok	Dokumentum adatbázisok
Node	Document
Nincs séma	Nincs séma
Kapcsolatok a modellben	Kapcsolatok beágyazással vagy „idegen” kulcsokkal
A kapcsolódó adatok megjelenítése minták segítségével	A kapcsolódó adatok megjelenítése beágyazással vagy join-okkal

BUDAPESTI CORVINUS EGYETEM

Gráf adatbázisok - Példák

Gráf adatbázisok – Neo4j

- A legismertebb gráf adatbázis
- A csúcsok ~ entitások, objektumok
  - Lehetnek tulajdonságaik (kulcs-érték párok)
    - Az értékek primitív adattípusok
    - A tulajdonságok (részben) indexelhetők
    - Megadható UNIQUE kényszer
    - Nincs NULL elem
  - Lehetnek címkéik
- A kapcsolatok
  - Van nevük
  - Lehetnek tulajdonságaik
  - Indexelhetők

BUDAPESTI CORVINUS EGYETEM

Gráf adatbázisok – Neo4j

- Java-alapú gráf adatbázis
- Egyidejű hozzáférések kezelése (MVCC)
  - Minden tranzakció egy konzisztens snapshot-ot lát
  - Egyszerre több tranzakció tud írni/olvasni
- Tranzakciók esetén a konzisztencia teljesülése a preferált
- Egy szerveres rendszerben az ACID feltételek is teljesülhetnek
- Flexibilis séma
- Elosztott rendszerben magas rendelkezésre állás és nagy teljesítmény
- Beépített gráf algoritmusok
- Index-mentes navigáció
- Szerepkör-alapú biztonság

BUDAPESTI CORVINUS EGYETEM

Neo4j - Cypher

- A Neo4j preferált lekérdező nyelve
- Deklaratív (nem procedurális)nyelv
- Minta egyezéseket vizsgál
- Az emberi gondolkodáshoz közel álló nyelv
- Záradékok használata (pl: WHERE, ORDER BY)

BUDAPESTI CORVINUS EGYETEM

Cypher – hatékony, jól olvasható

#3: A Language For Connected Data
Cypher Query Language

```
(SELECT T.directReportees AS directReportees, sum(T.count) AS count
FROM (
SELECT manager.pid AS directReportees, 0 AS count
  FROM person_reportee manager
  WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
UNION
  SELECT manager.pid AS directReportees, count(manager.directly_manages) AS count
FROM person_reportee manager
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION
SELECT manager.pid AS directReportees, count(reportee.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee reportee
ON manager.directly_manages = reportee.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION
SELECT manager.pid AS directReportees, count(L2Reportees.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee L1Reportees
ON manager.directly_manages = L1Reportees.pid
JOIN person_reportee L2Reportees
ON L1Reportees.directly_manages = L2Reportees.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
) AS T
GROUP BY directReportees)
UNION
(SELECT T.directReportees AS directReportees, sum(T.count) AS count
FROM (
SELECT manager.directly_manages AS directReportees, 0 AS count
FROM person_reportee manager
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
UNION
SELECT reportee.pid AS directReportees, count(reportee.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee reportee
ON manager.directly_manages = reportee.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION

SELECT depth1Reportees.pid AS directReportees,
count(depth2Reportees.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee L1Reportees
ON manager.directly_manages = L1Reportees.pid
JOIN person_reportee L2Reportees
ON L1Reportees.directly_manages = L2Reportees.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
) AS T
GROUP BY directReportees)
UNION
(SELECT T.directReportees AS directReportees, sum(T.count) AS count
  FROM(
  SELECT reportee.directly_manages AS directReportees, 0 AS count
FROM person_reportee manager
JOIN person_reportee reportee
ON manager.directly_manages = reportee.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION
SELECT L2Reportees.pid AS directReportees, count(L2Reportees.directly_manages)
AS count
FROM person_reportee manager
JOIN person_reportee L1Reportees
ON manager.directly_manages = L1Reportees.pid
JOIN person_reportee L2Reportees
ON L1Reportees.directly_manages = L2Reportees.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
) AS T
GROUP BY directReportees)
UNION
(SELECT L2Reportees.directly_manages AS directReportees, 0 AS count
FROM person_reportee manager
JOIN person_reportee L1Reportees
ON manager.directly_manages = L1Reportees.pid
JOIN person_reportee L2Reportees
ON L1Reportees.directly_manages = L2Reportees.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
)
```

```
MATCH (boss)-[:MANAGES*0..3]->(sub),
      (sub)-[:MANAGES*1..3]->(report)
WHERE boss.name = "John Doe"
RETURN sub.name AS Subordinate,
  count(report) AS Total
```

openCypher

Less time writing queries
- More time understanding the answers
- Leaving time to ask the next question

Less time debugging queries:
- More time writing the next piece of code
- Improved quality of overall code base

Code that's easier to read:
- Faster ramp-up for new project members
- Improved maintainability & troubleshooting

https://twitter.com/amyhodler/status/1233437495624253442

BUDAPESTI CORVINUS EGYETEM

Fontosabb Cypher adattípusok

Típus	Példa	Megjegyzés
Integer	13	Tulajdonság típus
Float	3.14	Tulajdonság típus
String	'Hello', "World"	Tulajdonság típus
Boolean	true, false	Tulajdonság típus
Date	"2019-06-01"	Tulajdonság típus
Time	"21:40:32"	Tulajdonság típus
DateTime	"2019-09-25T06:29:39Z"	Tulajdonság típus
Node	(a:Actor)	Szerkezet típus
Relationship	[d:Directed]	Szerkezet típus
Path	(a:Actor)-[:Acted_in]->(m:Movie)	Szerkezet típus
List	[0, 1, 2]	Összetett típus
Map	{kulcs1: érték1, kulcs2: érték2 …}	Összetett típus

BUDAPESTI CORVINUS EGYETEM

Fontosabb Cypher operátorok

Operátor típus	Példák
Matematikai	+, -, *, /, %, ^
Összehasonlító	=, <, >, <>, <=, >=, IS NULL, IS NOT NULL
Szöveg összehasonlító	STARTS WITH, ENDS WITH, CONTAINS
Logikai	NOT, AND, OR, XOR
Szöveg	+ (összefűzés), =~ (regex)
Aggregációs	DISTINCT
Tulajdonság (property)	. (csomópont vagy kapcsolat tulajdonság elérése) = (csomópont vagy kapcsolat tulajdonságok felülírása += (csomópont vagy kapcsolat tulajdonság módosítása, hozzáadása)
Lista	IN (tartalmazást vizsgál) + (összefűz) [ ] (listaelemek elérése)

BUDAPESTI CORVINUS EGYETEM

Fontosabb Cypher függvények

Függvény típus	Példák
Matematikai	abs(), round(), rand(), sqrt(), log(), sin(), cos(),
Szöveg	left(), right(), toLower(), toUpper(), trim(), substring()
Predikátum	exists(), all(), any(), isEmpty()
Skalár	id(), type(), toFloat(), toInteger, toBoolean()
Lista	labels(), nodes(), relationships(), range()
Dátum/Idő	date(), datetime(), time()

BUDAPESTI CORVINUS EGYETEM

A Case kifejezés Cypher-ben

```
CASE kifejezés
   WHEN értéke1 THEN eredmény1
   WHEN értéke2 THEN eredmény2
   …
   [ELSE default_érték]
END
```

Neo4j - lekérdezések

MATCH( ) - Csúcsok, kapcsolatok, tulajdonságok, címkék és minták keresése az adatbázisban

- A SQL SELECT-hez hasonló elven működik
- A lekérdezés által visszaadott értékeket a RETURN kulcsszó után adhatjuk meg
- A lekérdezés eredményét a WHERE kulcsszó után megadott feltételekkel szűrhetjük
- A megjelenítendő eredményt a LIMIT kulcsszóval korlátozhatjuk
- Az eredményt többféle nézetben (Graph, Table, Text, Code) is megtekinthetjük

Neo4j – Egyszerű lekérdezések I.

```
MATCH (n)
RETURN n
```

Listázza az összes csúcsot

```
MATCH (p:Person)
RETURN p
LIMIT 1
```

Megjeleníti a legelső személyt

```
MATCH (p:Person {name: 'Tom Hanks'})
RETURN p
```

Megjeleníti Tom Hanks adatait

```
MATCH (:Person {name: 'Tom Hanks'})-[:DIRECTED]->(movie:Movie)
RETURN movie.title
```

Megjeleníti, hogy Tom Hanks milyen film(ek)et rendezett

Neo4j – Egyszerű lekérdezések II.

```
MATCH (p:Person {name:'Tom Hanks'})-[rel:DIRECTED]-(m:Movie)
RETURN p.name AS name, p.born AS `Year Born`, m.title AS title,
m.released AS `Year Released`
```

Megjeleníti Tom Hanks és az általa rendezett film egyes adatait

```
MATCH (:Person)-[:DIRECTED]->(m:Movie)
RETURN DISTINCT m.released
```

Megjeleníti azon éveket, amikor filmeket rendeztek

```
MATCH (j:Person)
WHERE j.born = 1955
RETURN j
```

Megjeleníti az 1955-ben született személyeket

```
MATCH (j:Person)
WHERE NOT j.born = 1955
RETURN j
```

Megjeleníti azokat, akik nem 1955-ben születtek

Neo4j – Egyszerű lekérdezések III.

```
MATCH (p:Person)
WHERE p.name STARTS WITH 'M'
RETURN p.name
```

Megjeleníti az M betűvel kezdődő személyeket

```
MATCH (p:Person)
WHERE p.name CONTAINS 'a'
RETURN p.name
```

Megjeleníti azon személyeket, akik nevében van „a” betű

```
MATCH (p:Person)
WHERE p.name ENDS WITH 'n'
RETURN p.name
```

Megjeleníti azon személyeket, akik neve n-re végződik

```
MATCH (p:Person)
WHERE p.name =~ 'Jo.*'
RETURN p.name
```

Reguláris kifejezéssel szűr a személyek nevére

Neo4j – Egyszerű lekérdezések IV.

```
MATCH (m:Movie)
WHERE ID(m) IN [0, 5, 9]
RETURN m
```

Megjeleníti a 0, 5 és 9 azonosítójú filmeket

```
MATCH (p:Person)-[d:REVIEWED]->(m:Movie)
RETURN p, d, m
```

Megjeleníti, hogy melyik személy milyen filmekről írt kritikát

```
MATCH (p:Person)-[d:WROTE]->(m:Movie)
WHERE not exists ((p)-[:ACTED_IN]->(m))
RETURN p, d, m
```

Megjeleníti azokat a személyeket és filmeket, ahol az író nem szerepelt a filmben

```
MATCH (p:Person)-[:ACTED_IN]->(m:Movie)<-[:ACTED_IN]-(p2:Person)
WHERE p.name= 'Gene Hackman'
AND exists( (p2)-[:DIRECTED]->(m) )
RETURN p, p2, m
```

Kivel és milyen filmben szerepelt együtt Gene Hackman, ha a másik szereplő egyben rendező is volt?

Neo4j – Egyszerű lekérdezések V.

```
MATCH (p:Person)
WHERE p.name STARTS WITH 'J'
OPTIONAL MATCH (p)-[:DIRECTED]->(m)
RETURN p.name, m.title
```

Megjeleníti a személyeket és az általuk rendezett filmet (ha van olyan)

```
MATCH (p:Person)
RETURN count(*)
```

Megjeleníti, hogy hány személy van az adatbázisban

```
MATCH (p:Person)-[:FOLLOWS]->(p2:Person)
WITH p, count(*) AS db
RETURN p.name, db
```

Megjeleníti azt, hogy melyik személy hány másikat követ

```
MATCH (p:Person)-[:WROTE]->(m:Movie)
RETURN p.name, collect(m.title) AS filmek
```

Megjeleníti, hogy melyik személy milyen filmeket rendezett

Neo4j – Egyszerű lekérdezések VI.

```
MATCH (p:Person)-[:ACTED_IN]->(m:Movie)
RETURN p.name, size(collect(m.title)) AS db
```

Megjeleníti, hogy melyik személy hány filmben szerepelt

```
MATCH  (p:Person)-[:ACTED_IN]->(m:Movie)
RETURN p.name, size(collect(m.title)) as db
ORDER BY db DESC, p.name
LIMIT 5
```

Megjeleníti, hogy kik szerepeltek a legtöbb filmben – az első 5

```
MATCH (p:Person)-[r]->(p2:Person)
RETURN type(r), count(*)
```

Megjeleníti azt, hogy milyen típusú és hány db kapcsolat van a személyek között

```
MATCH (p:Person)-[r]->(m:Movie)
WHERE p.born IS NULL
RETURN p.name, type(r), m.title, avg(date().year-m.released)
```

Megjeleníti, hogy azok a személyek, akiknek nincs megadva a születési évük, milyen filmekkel vannak kapcsolatban, és a filmek átlagosan hány éve jelentek meg

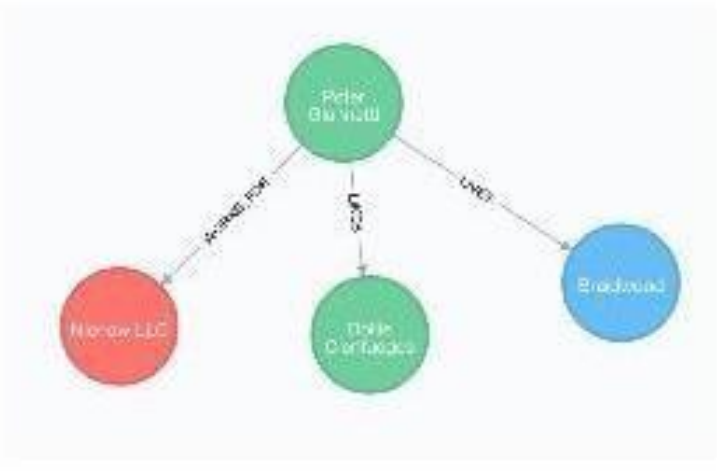
BUDAPESTI CORVINUS EGYETEM

Cypher példa*

Cypher: Example Query

neo4j

```
MATCH (p:Person {fullName :"Peter Giannetti"})-[r]-(n)
RETURN p, r, n
```



https://neo4j.com/docs/developer-manual/current/cypher/

https://www.opencypher.org/

Neo4j – CRUD műveletek

CREATE (változónév:címke {tulajdonságok:értékek}) -- Csomópont létrehozása

SET – Cimkék, tulajdonságok és kapcsolatok módosítása

REMOVE – Cimkék és tulajdonságok törlése

DELETE – Csomópontok és kapcsolatok törlése

Neo4j – CRUD műveletek I.

```
CREATE (:Movie {title: 'Félelem',released: 2011, tagline:'Amit mindenki érez' })
```

Létrehoz egy új filmet

```
create (:Person {name: 'Kiss Ilona', born:1988 }),
(:Person {name: 'Nagy Béla', born:2000 })
```

Létrehoz két új személyt

```
MATCH (a:Person), (b:Movie)
WHERE a.name = 'Kiss Ilona' AND b.title = 'Félelem'
CREATE (a)-[:FOLLOWS]->(b)
```

Létrehoz új kapcsolatot meglévő csúcsok között

```
create (p:Person {name:'Fekete Edit', born:1997})-[:WROTE]->(m:Movie {title:'A hősnő', released:2021})
return (p)-[]-(m)
```

Egyszerre hoz létre új személyt és filmet, valamint kapcsolatot közöttük

Neo4j – CRUD műveletek II.

```
MATCH (p:Person
{name: 'Fekete Edit'})
SET p.born = 2010
RETURN p
```

Módosítja az adott személy születési évét

```
MATCH (p:Person {name: 'Fekete Edit'})
SET (case when p.born < 2015 then p end).born = 2015
RETURN p
```

Módosítja az adott személy születési évét, ha teljesül egy feltétel

```
MATCH (p:Person {name: 'Fekete Edit’})
REMOVE  p.born
RETURN p
```

Törli az adott személy születési évét

```
MATCH (n {name: 'Fekete Edit'})
REMOVE n:Person
RETURN n.name, labels(n)
```

Törli az adott csúcs címkéjét

Neo4j – CRUD műveletek II.

```
MATCH (n:Person {name: 'Fekete Edit'})
DELETE n
```

Törli az adott csomópontot

```
MATCH (p:Person {name: 'Kiss Ilona'})-[r:FOLLOWS]->(m:Movie)
DELETE r
RETURN p
```

Egy adott kapcsolat törlése

```
MATCH (n {name: 'Kiss Ilona'})
DETACH DELETE n
```

Törli az adott csomópontot és minden kapcsolatát

```
MATCH (n)
DETACH DELETE n
```

Töröl minden csomópontot és kapcsolatot

Neo4j - Indexek

CREATE INDEX – index létrehozása

SHOW INDEXES [VERBOSE]– indexek listázása

DROP INDEX – index törlése

- A VERBOSE segítségével opcionálisan részletesebb lista jeleníthető meg
- A PROFILE utasítással megjeleníthető a végrehajtási terv
- Az EXPLAIN utasítás hasonlóan működik, de magát az utasítást nem hajtja végre, csak a végrehajtási tervet jeleníti meg

Neo4j – Indexek - példák

```
profile
match (p:Person)
return p
```

A lekérdezés és végrehajtási terv

```
CREATE INDEX i_name
IF NOT EXISTS FOR (n:Person)
ON (n.name)
```

A személyeket indexeli név alapján, ha még nincs index

```
CREATE INDEX i_filmek
FOR (m:Movie)
ON (m.title, m.released)
```

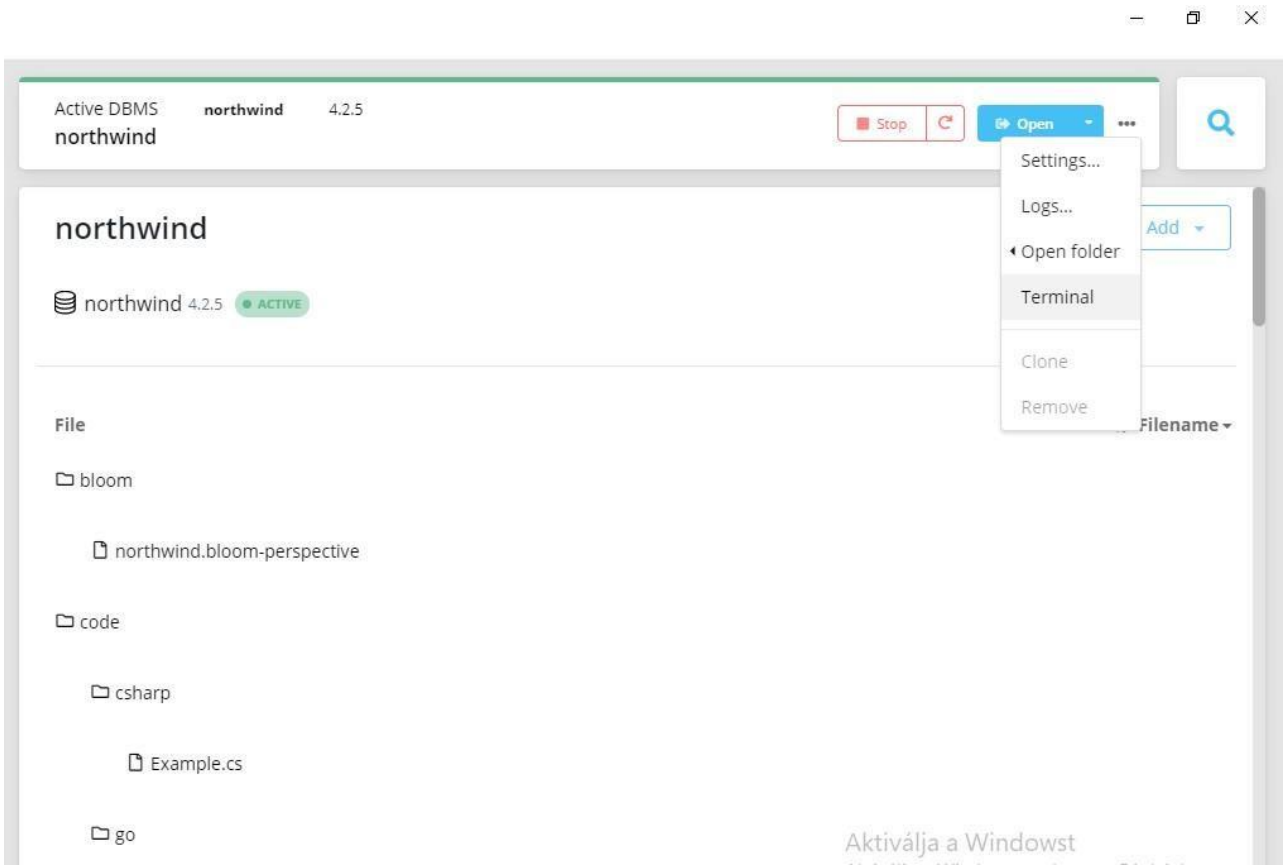
Összetett index létrehozása

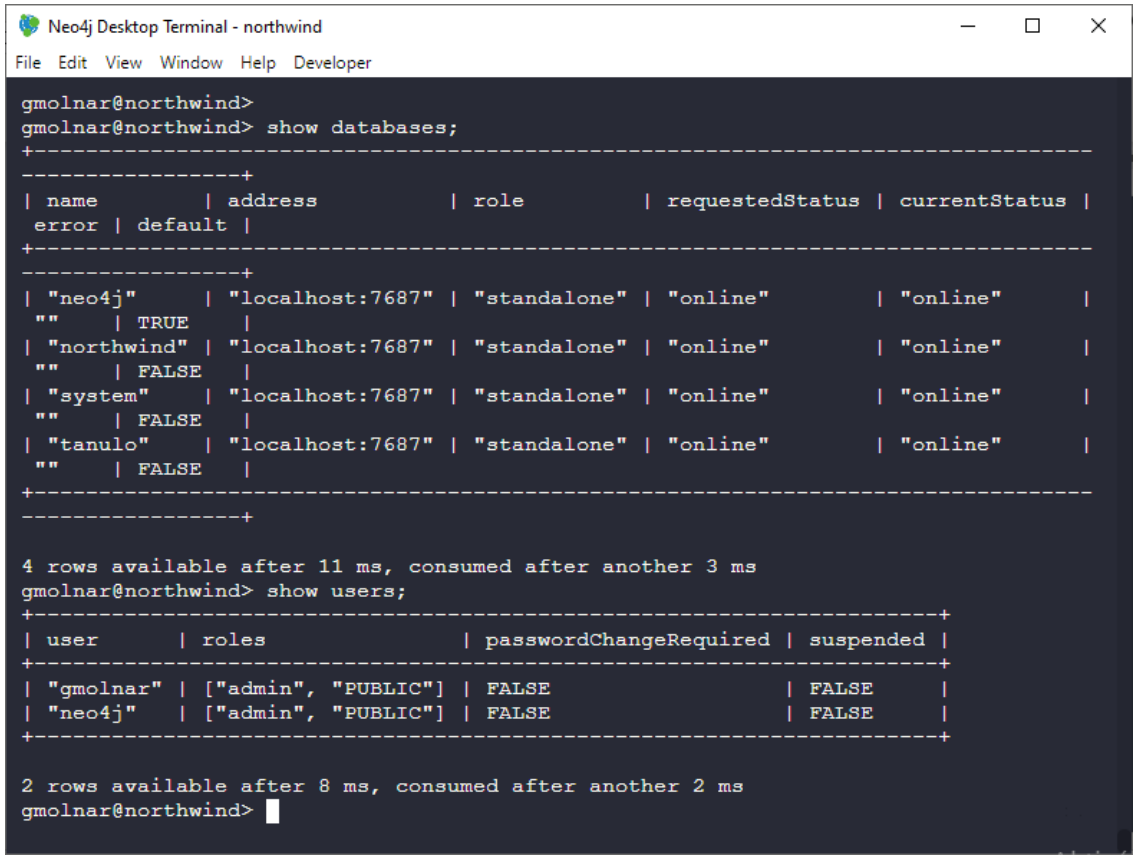
```
DROP INDEX i_filmek
```

Törli az adott indexet

BUDAPESTI CORVINUS EGYETEM

Cypher-shell terminal





Elérés Python-ból*

```
!pip install neo4j
from neo4j import GraphDatabase

class Neo4jConnection:
  …
return response

conn = Neo4jConnection(uri="bolt://localhost:7687", user="neo4j", pwd="neo4j")

query_string = 'match (n) return n limit 4’

conn.query(query_string, db='northwind')
```

A teljes kód a mellékelt python.ipynb fájlban található

Feladatok megoldása I.

A Neo4J Desktop-ban hozzon létre új projektet, majd egy új adatbázist tanulo néven! Nyissa meg a Neo4J Browsert, majd tegye aktívvá az új adatbázist!

a. Hozzon létre :Tanulo és :Tanar csomópontokat az alábbi ábra alapján:

TANULO		
Nev	Eletkor	Atlag
Kiss Béla	22	3.5
Nagy Ilona	23	4.4

TANAR	
Nev	Szak
Tóth Ottó	Matematika
Nagy Ivett	Informatika

Feladatok megoldása II.

Az előző feladatban létrehozott tanulo adatbázisban hozzon létre két új kapcsolatot :Tanit néven az alábbiak szerint:

a. Tóth Ottó tanítja Kiss Bélát

b. Nagy Ivett tanítja Nagy Ilonát

c. A szükséges utasításokat adja meg válaszként!

Feladatok megoldása III.

A 7. feladatban létrehozott tanulo adatbázisban végezze el a következő módosításokat:

a. Nagy Ilona átlaga legyen 5.0

b. Tóth Ottó szakja legyen Fizika

c. A szükséges utasításokat adja meg válaszként!

BUDAPESTI CORVINUS EGYETEM

Feladatok megoldása IV.

A Neo4J Desktop-ban tegye aktívvá a tanulo projektet, majd nyisson új terminált az adatbázis melletti Open gombnál mellett lévő három pont (...) kiválasztásával! Utána lépjen be a bin mappába, majd adja ki a cypher-shell parancsot!

a. Szükség esetén adja meg a felhasználónevet és a jelszót

b. Csatlakozzon a tanulo adatbázishoz (:use tanulo;)

c. Kérdezze le az első két csúcsot!

Köszönöm a figyelmet!