Rafał Gaweł

Bazy Danych
Neo4j

3. Zaimplementować funkcję (wystarczy wykonać jedno zapytanie typu MATCH WHERE i wyświetlić wynik).

Template do wykonania zadań

```java
import org.neo4j.driver.v1.*;

public class GraphDatabase implements AutoCloseable
{
    private final Driver driver;

    public GraphDatabase(String uri, String user, String password )
    {
        driver = org.neo4j.driver.v1.GraphDatabase.driver( uri, AuthTokens.basic( user, password ) );
    }

    @Override
    public void close() { driver.close(); }

    private void zadanie(){
        Session session = driver.session();
        Transaction tx = session.beginTransaction();

        // Miejsce na kod

        tx.close();
        tx.success();
        session.close();
    }


    public static void main(String[] args) {
        try ( GraphDatabase graphDatabase = new GraphDatabase( uri: "bolt://localhost:7687", user: "neo4j", password: "1234" ) )
        {
            graphDatabase.zadanie();
        }
    }
}
```

Zapytanie cypher

```
MATCH (n:Movie)
WHERE n.released > 2008
RETURN n.title, n.tagline, n.released
```

Kod w javie

```
private void zadanie3(){
    Session session = driver.session();
    Transaction tx = session.beginTransaction();

    StatementResult result = tx.run( s: "MATCH (n:Movie) " +
            "WHERE n.released > 2008 " +
            "RETURN n.title, n.tagline, n.released" );

    while ( result.hasNext() )
    {
        Record record = result.next();
        System.out.println( record.get( "n.title" ) + ", " +
                record.get("n.tagline") + ", " +
                record.get("n.released"));
    }

    tx.success();
    tx.close();
    session.close();
}
```

Rezultat

```
"Cloud Atlas", "Everything is connected", 2012
"Ninja Assassin", "Prepare to enter a secret world of assassins", 2009
```

4. Stworzyć kilka nowych węzłów reprezentujących film oraz aktorów w nim występujących, następnie stworzyć relacje ich łączące (np. ACTED_IN)

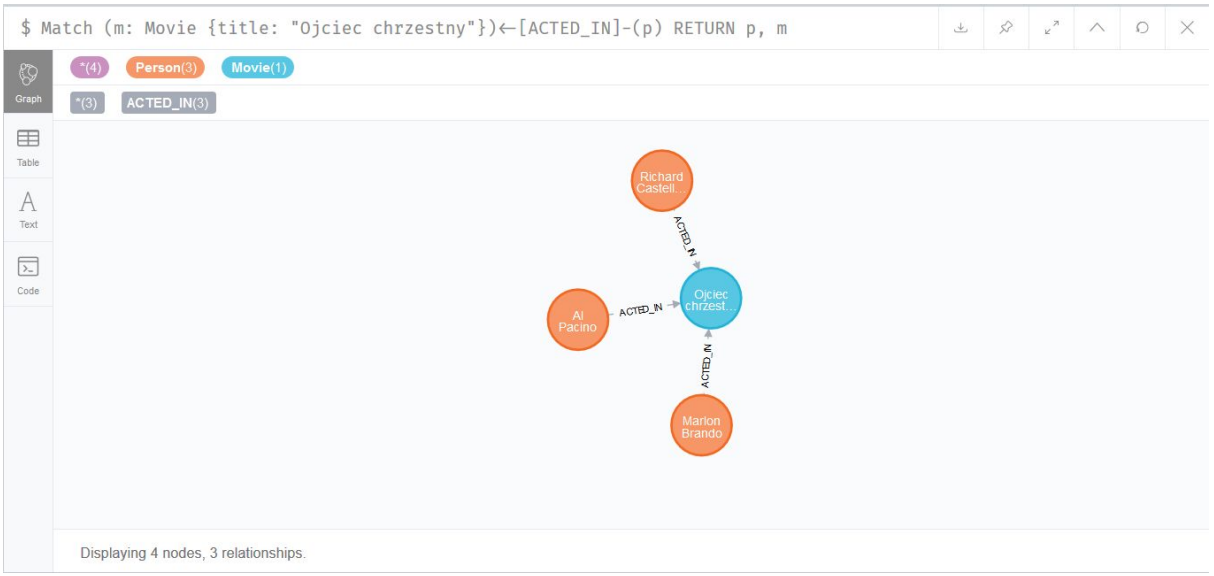
Zapytanie cypher

```
CREATE (n: Movie { title: "Ojciec chrzestny", tagline: "Przychodzisz do mnie w dniu ślubu mojej córki
i prosisz bym mordował za pieniądze", released: 1972 });
CREATE (n: Person { name: "Al Pacino", born: 1940 });
CREATE (n: Person { name: "Marlon Brando", born: 1924 });
CREATE (n: Person { name: "Richard Castellano", born: 1933 });
MATCH (n: Movie { title: "Ojciec chrzestny" }), (m: Person { name: "Al Pacino" })
CREATE (m)-[:ACTED_IN]→(n);
MATCH (n: Movie { title: "Ojciec chrzestny" }), (m: Person { name: "Marlon Brando" })
CREATE (m)-[:ACTED_IN]→(n);
MATCH (n: Movie { title: "Ojciec chrzestny" }), (m: Person { name: "Richard Castellano" })
CREATE (m)-[:ACTED_IN]→(n);
```

Kod w javie

```
private void zadanie4(){
    Session session = driver.session();
    Transaction tx = session.beginTransaction();

    String newMovie = "CREATE (n: Movie { title: \"%s\", tagline: \"%s\", released: \"%d\" })";
    String newPerson = "CREATE (n: Person { name: \"%s\", born: \"%d\" })";
    String newActedIn = "MATCH (n: Movie { title: \"%s\" }), (m: Person { name: \"%s\" }) " +
            "CREATE (m)-[:ACTED_IN]->(n)";



    tx.run(String.format(newMovie, "Ojciec chrzestny", "Przychodzisz do mnie w dniu ślubu mojej córki " +
            "i prosisz bym mordował za pieniądze" , 1972));

    tx.run(String.format(newPerson, "Al Pacino", 1940));
    tx.run(String.format(newPerson, "Marlon Brando", 1924));
    tx.run(String.format(newPerson, "Richard Castellano", 1933));

    tx.run(String.format(newActedIn, "Ojciec chrzestny", "Al Pacino"));
    tx.run(String.format(newActedIn, "Ojciec chrzestny", "Marlon Brando"));
    tx.run(String.format(newActedIn, "Ojciec chrzestny", "Richard Castellano"));

    tx.success();
    tx.close();
}
```

Rezultat


$ Match (m: Movie {title: "Ojciec chrzestny"})←[ACTED_IN]-(p) RETURN p, m
Graph
Table
Text
Code
*(4)
Person(3)
Movie(1)
*(3)
ACTED_IN(3)
Richard Castell...
ACTED_IN
Al Pacino
ACTED_IN
Ojciec chrzest...
ACTED_IN
Marlon Brando
Displaying 4 nodes, 3 relationships.

5. Dodać zapytaniem nowe własności nowo dodanych węzłów reprezentujących aktorów (np. birthdate oraz birthplace)

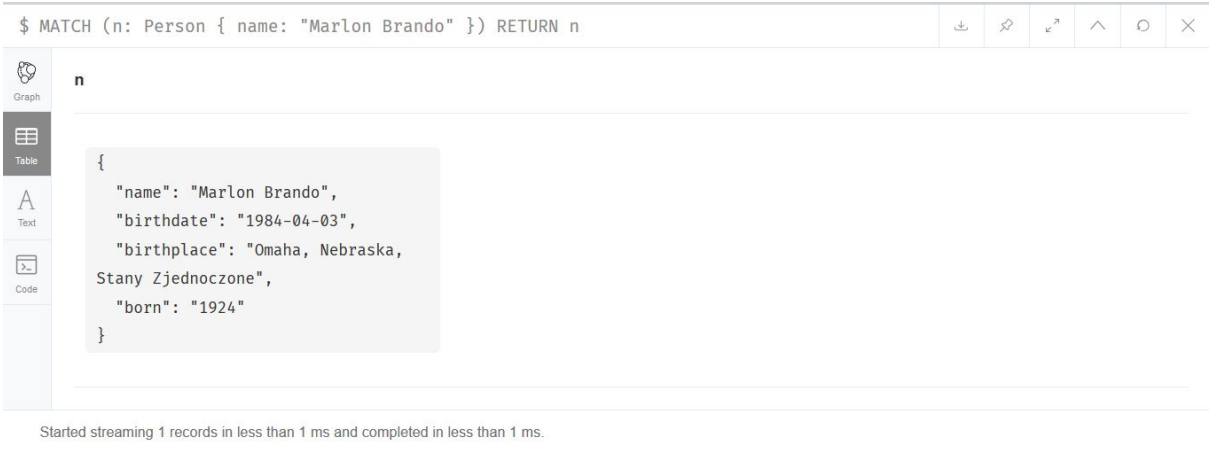
Zapytanie cypher

```
PROFILE MATCH (n: Person { name: "Marlon Brando" })
SET n.birthplace = "Omaha, Nebraska, Stany Zjednoczone", n.birthdate = date({ year:1984, month:4, day:3 })
```

Kod w javie

```
private void zadanie5(){
    Session session = driver.session();
    Transaction tx = session.beginTransaction();

    String query = "MATCH (n: Person { name: \"%s\" }) " +
            "SET n.birthplace = \"%s\", n.birthdate = date({ year:%d, month:%d, day:%d})";

    tx.run(String.format(query, "Marlon Brando", "Omaha, Nebraska, Stany Zjednoczone", 1984, 4, 3));
    tx.success();
    tx.close();
}
```

Rezultat



6. Ułożyć zapytanie, które zmieni wartość atrybutu węzłów danego typu, jeżeli inny atrybut węzła spełnia kryterium.

Dla filmów wydanych po roku 2007 tagline zmienimy na “Hasta la vista, baby”

Zapytanie cypher

```
MATCH (n: Movie) WHERE n.released > 2007
SET n.tagline = "Hasta la vista, baby"
```

Kod w javie

```
private void zadanie6(){
    Session session = driver.session();
    Transaction tx = session.beginTransaction();

    String query = "MATCH (n: Movie) WHERE n.released > 2007 " +
            "SET n.tagline = \"%s\"";

    tx.run(String.format(query, "Hasta la vista, baby"));
    tx.success();
    tx.close();
}
```

Rezultat

7. Zapytanie o aktorów którzy grali w co najmniej 2 filmach (użyć collect i length) i policzyć średnią wystąpień w filmach dla grupy aktorów, którzy wystąpili w co najmniej 3 filmach.

Aktorzy którzy grali w co najmniej dwóch filmach

Zapytanie cypher

```
MATCH (p:Person) -[:ACTED_IN]→ (m:Movie)
WITH p, SIZE(COLLECT(m)) as cnt
WHERE cnt ⩾ 2
RETURN p.name, cnt
```

Kod w javie

```
private void zadanie7(){
    Session session = driver.session();
    Transaction tx = session.beginTransaction();

    String query = "MATCH (p:Person) -[:ACTED_IN]-> (m:Movie) " +
            "WITH p, SIZE(COLLECT(m)) as cnt " +
            "WHERE cnt >= 2 " +
            "RETURN p.name, cnt ";

    StatementResult result = tx.run(query);

    System.out.println(result.keys());
    while ( result.hasNext() )
    {
        Record record = result.next();
        System.out.println( record.get( "p.name" ) + ", " + record.get("cnt"));
    }
    tx.success();
    tx.close();
}
```

Rezultat

```
[p.name, cnt]
"Bill Paxton", 3
"Tom Hanks", 12
"Gary Sinise", 2
"Kevin Bacon", 3
"Sam Rockwell", 2
"Oliver Platt", 2
"Marshall Bell", 2
"Jerry O'Connell", 2
"Kiefer Sutherland", 2
"Max von Sydow", 2
"Robin Williams", 3
"Cuba Gooding Jr.", 4
"Meg Ryan", 5
"Nathan Lane", 2
```

Średnia wystąpień w filmach dla grupy aktorów, którzy wystąpili w co najmniej 3 filmach

Zapytanie cypher

```
MATCH (p:Person) -[:ACTED_IN]→ (m:Movie)
WITH p, SIZE(COLLECT(m)) as cnt
WHERE cnt ⩾ 3
RETURN AVG(cnt) as avg
```

Kod w javie

```
private void zadanie7b(){
    Session session = driver.session();
    Transaction tx = session.beginTransaction();

    String query = "MATCH (p:Person) -[:ACTED_IN]-> (m:Movie) " +
            "WITH p, SIZE(COLLECT(m)) as cnt " +
            "WHERE cnt >= 3 " +
            "RETURN AVG(cnt) as avg";

    StatementResult result = tx.run(query);

    System.out.println(result.next().get("avg"));

    tx.success();
    tx.close();
}
```

Rezultat

```
4.3333333333333334
```

9. Zmienić wartość wybranego atrybutu w węzłach na ścieżce pomiędzy dwoma podanymi węzłami.

W węzłach pomiędzy Marlonem Brando a Keanu Reevsem ustawiam atrybut attribute=true.

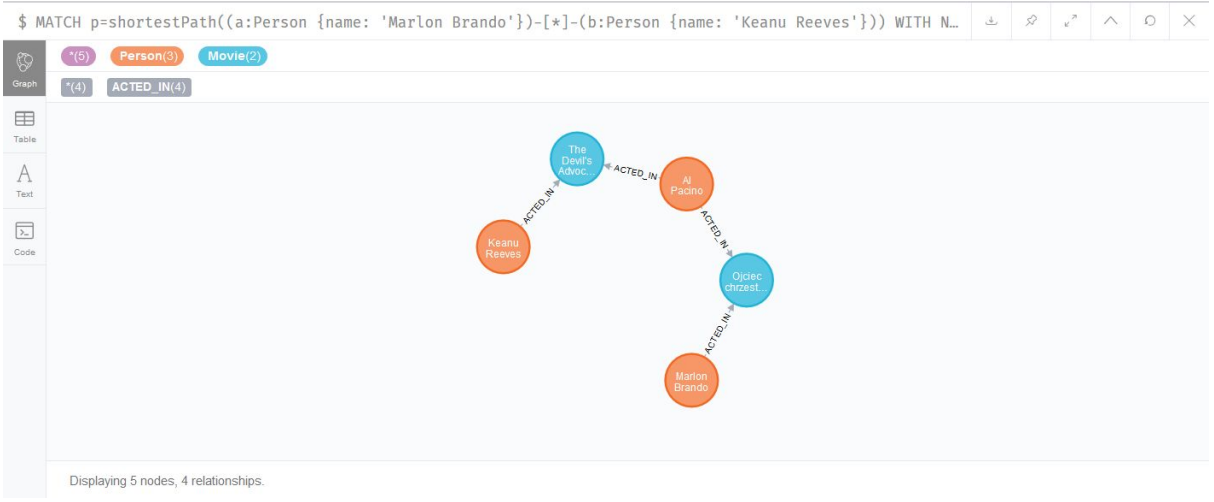
Zapytanie cypher

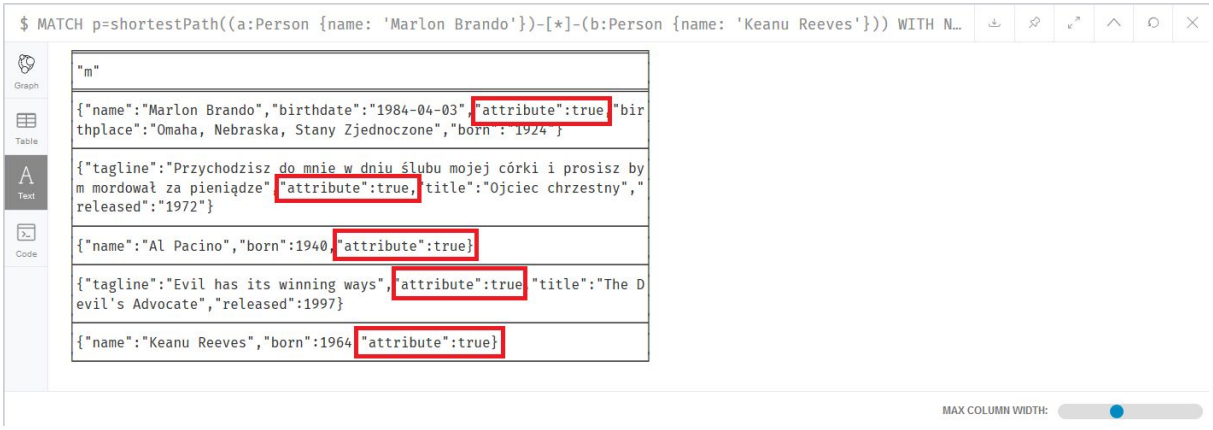
```
MATCH p=shortestPath((a:Person {name: 'Marlon Brando'})-[*]-(b:Person {name: 'Keanu Reeves'}))
WITH NODES(p) AS n
UNWIND n AS m
SET m.attribute = true
RETURN m
```

Kod w javie

```
private void zadanie9(){
    Session session = driver.session();
    Transaction tx = session.beginTransaction();

    String query = "MATCH p=shortestPath((a:Person {name: \"%s\"})-[*]-(b:Person {name: \"%s\"})) " +
            "WITH NODES(p) AS n " +
            "UNWIND n AS m " +
            "SET m.attribute = true";

    tx.run(String.format(query, "Marlon Brando", "Keanu Reeves"));
    tx.success();
    tx.close();
}
```

Rezultat




$ MATCH p=shortestPath((a:Person {name: 'Marlon Brando'})-[*]-(b:Person {name: 'Keanu Reeves'})) WITH N…
Graph
Table
Text
Code
"m"
{"name":"Marlon Brando","birthdate":"1984-04-03","attribute":true,"birthplace":"Omaha, Nebraska, Stany Zjednoczone","born":1924}
{"tagline":"Przychodzisz do mnie w dniu ślubu mojej córki i prosisz bym mordował za pieniądze","attribute":true,"title":"Ojciec chrzestny","released":"1972"}
{"name":"Al Pacino","born":1940,"attribute":true}
{"tagline":"Evil has its winning ways","attribute":true,"title":"The Devil's Advocate","released":1997}
{"name":"Keanu Reeves","born":1964,"attribute":true}
MAX COLUMN WIDTH:

10. Wyświetlić węzły, które znajdują się na 2 miejscu na ścieżkach o długości 4 pomiędzy dwoma wybranymi węzłami.

Zapytanie cypher

```
MATCH (a:Person {name: 'Marlon Brando'})-[r1]-(n)-[*2]-(m)-[r3]-(b:Person {name: 'Keanu Reeves'})
RETURN n,  m
```

Kod w javie

```
private void zadanie10(){
    Session session = driver.session();
    Transaction tx = session.beginTransaction();

    String query = "MATCH (a:Person {name: 'Marlon Brando'})-[r1]-(n)-[*2]-(m)-[r3]-(b:Person {name: 'Keanu Reeves'})\n" +
            "RETURN n,  m";

    StatementResult result = tx.run(String.format(query, "Marlon Brando", "Keanu Reeves"));
    while ( result.hasNext() )
    {
        Record record = result.next();
        System.out.println( record.get("n").get("title"));
        System.out.println( record.get("m").get("title"));
    }
    tx.success();
    tx.close();
}
```

Rezultat

```
"Ojciec chrzestny"
"The Devil's Advocate"
```

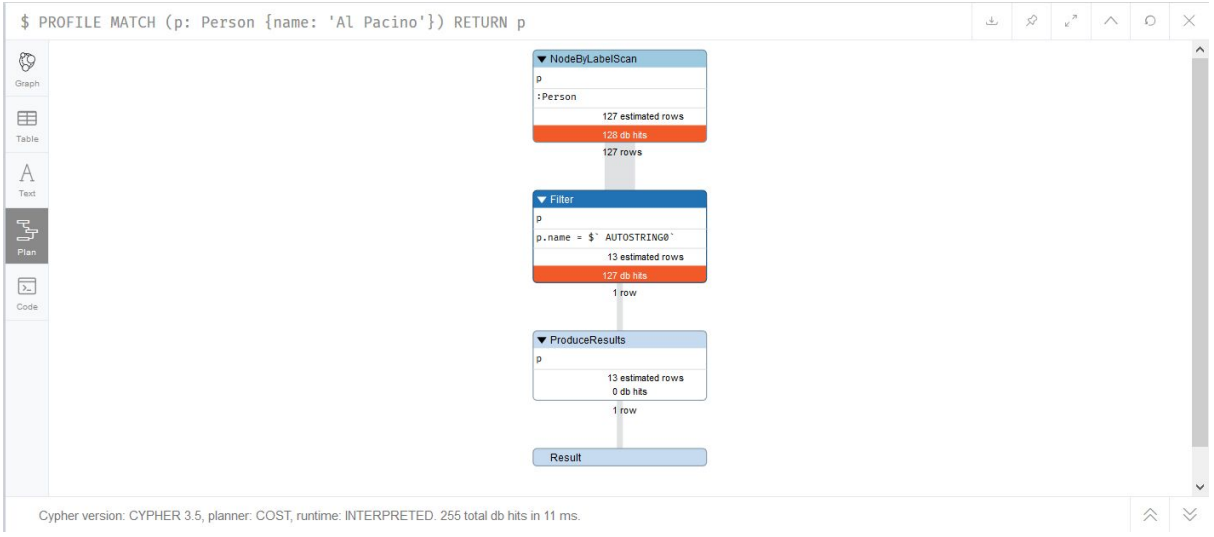
Najkrótsza ścieżka między Marlonem Brando i Keanu Reevesem ma długość 4 i jest jedyną. Węzły na drugich miejscach są to filmy “Ojciec chrzestny” i “The Devil’s Advocate”.

11. Porównać czas wykonania zapytania o wybranego aktora bez oraz z indeksem w bazie nałożonym na atrybut name (DROP INDEX i CREATE INDEX oraz użyć komendy PROFILE/EXPLAIN).

Zapytanie

```
MATCH (p: Person {name: 'Al Pacino'}) RETURN p
```

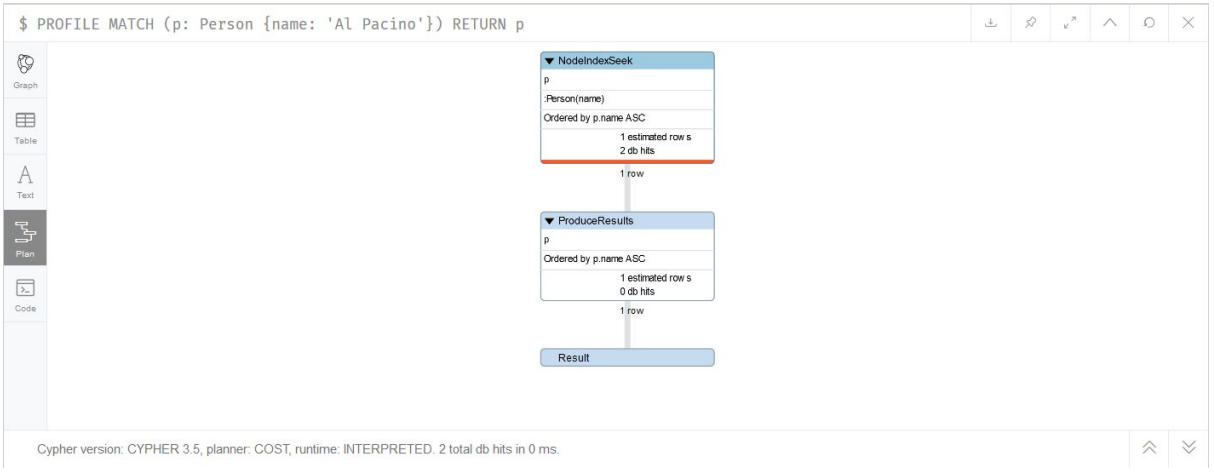
Bez indeksu

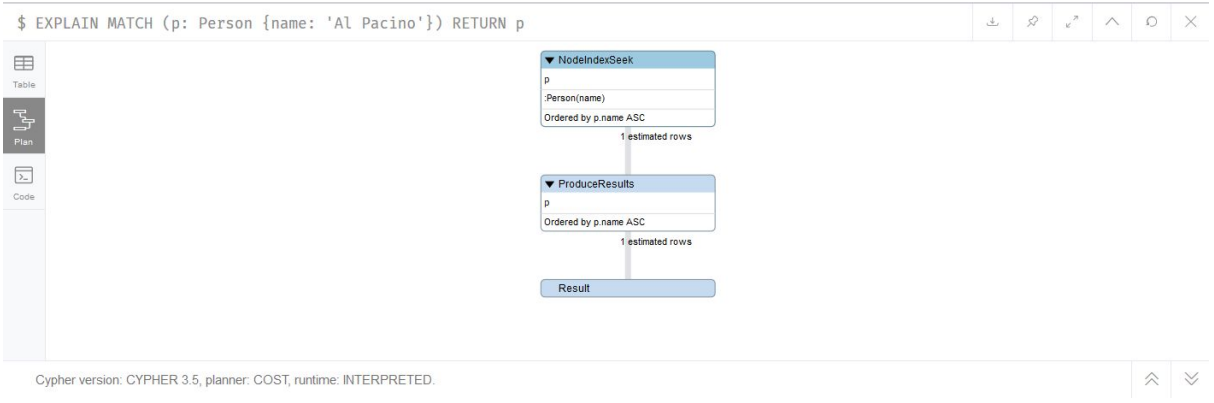


Założenie indeksu

```
1 CREATE INDEX ON :Person(name)
```

Z indeksem





Bez indeksu egzekutor zapytań musi przeszukać wszystkie węzły z etykietą Person. Z indeksem egzekutor wie gdzie jest poszukiwany węzeł i od razu do niego przechodzi.

Czas bez indeksu: 11ms. Czas z indeksem: 0ms.

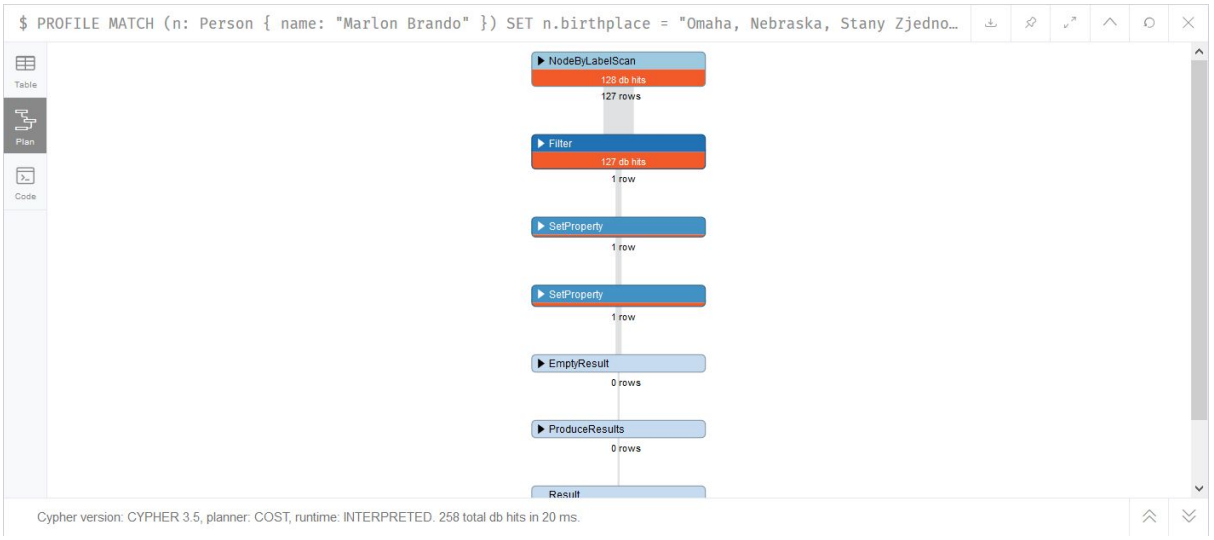
12. Spróbuj dokonać optymalizacji wybranych dwóch zapytań z poprzednich zadań.

Zapytanie 1

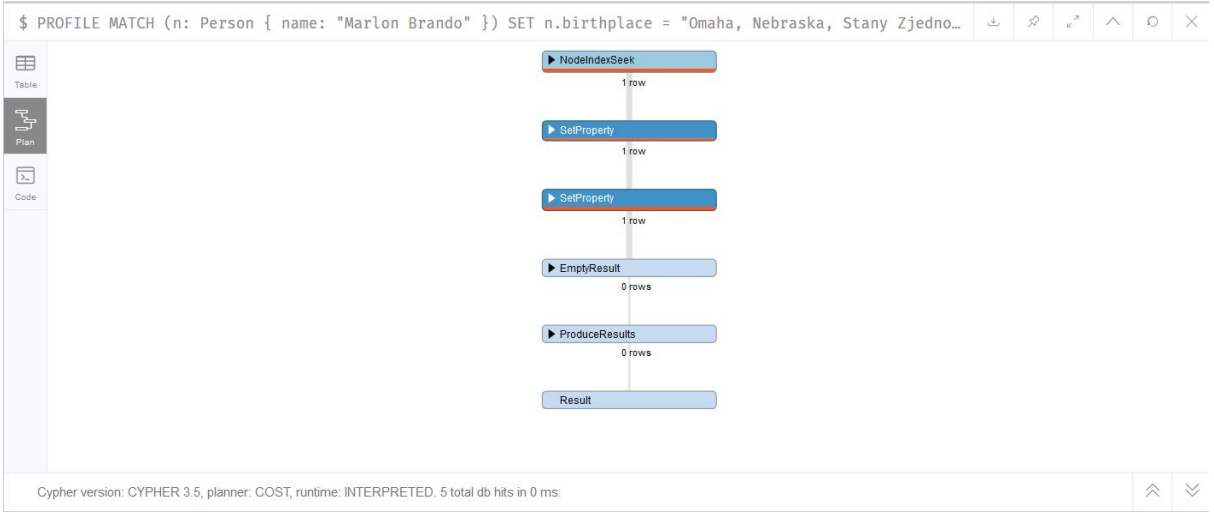
```
PROFILE MATCH (n: Person { name: "Marlon Brando" })
SET n.birthplace = "Omaha, Nebraska, Stany Zjednoczone", n.birthdate = date({ year:1984, month:4, day:3 })
```

Optymalizacja będzie polegała na dodaniu indeksu na polu name.

Bez indeksu



Z indeksem



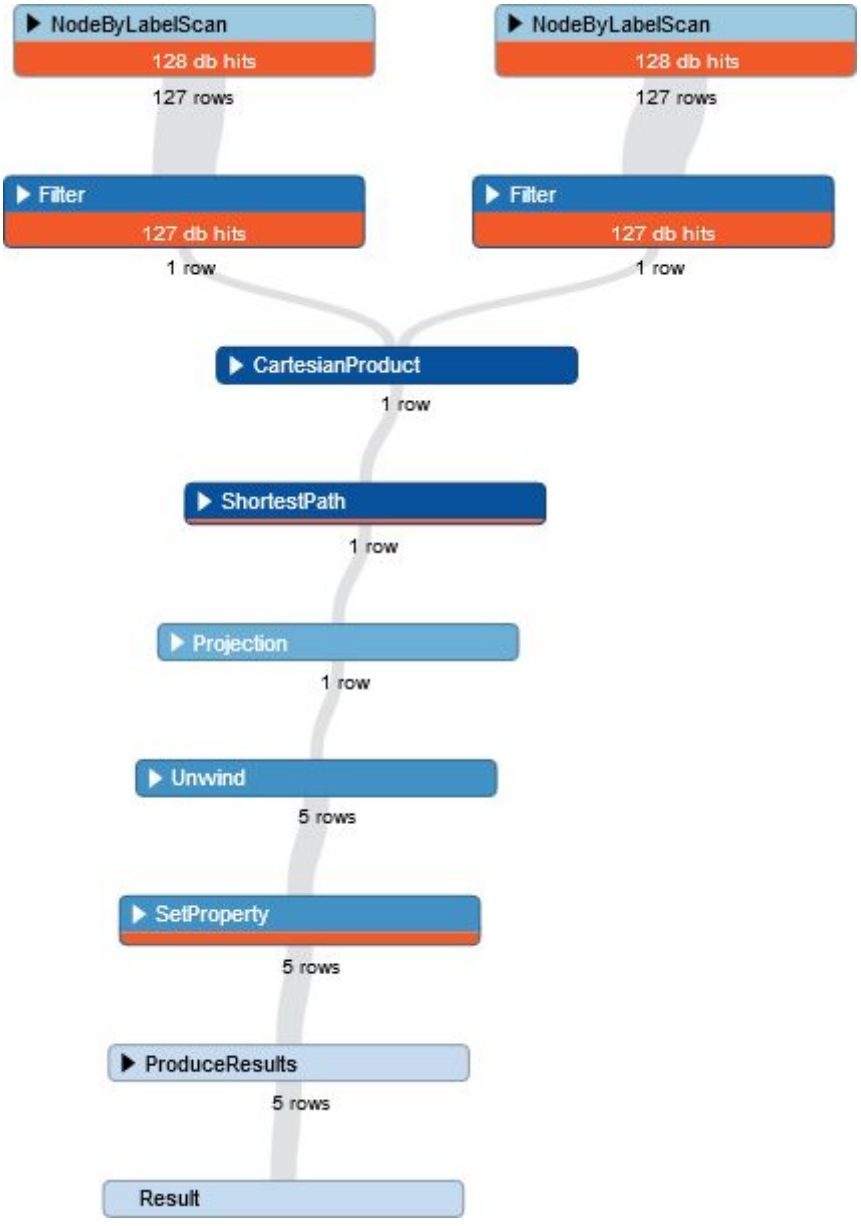
Dodanie indeksu poprawiło czas wykonania z 20ms do 0ms.

Zapytanie 2

```
MATCH p=shortestPath((a:Person {name: 'Marlon Brando'})-[*]-(b:Person {name: 'Keanu Reeves'}))
WITH NODES(p) AS n
UNWIND n AS m
SET m.attribute = true
RETURN m
```

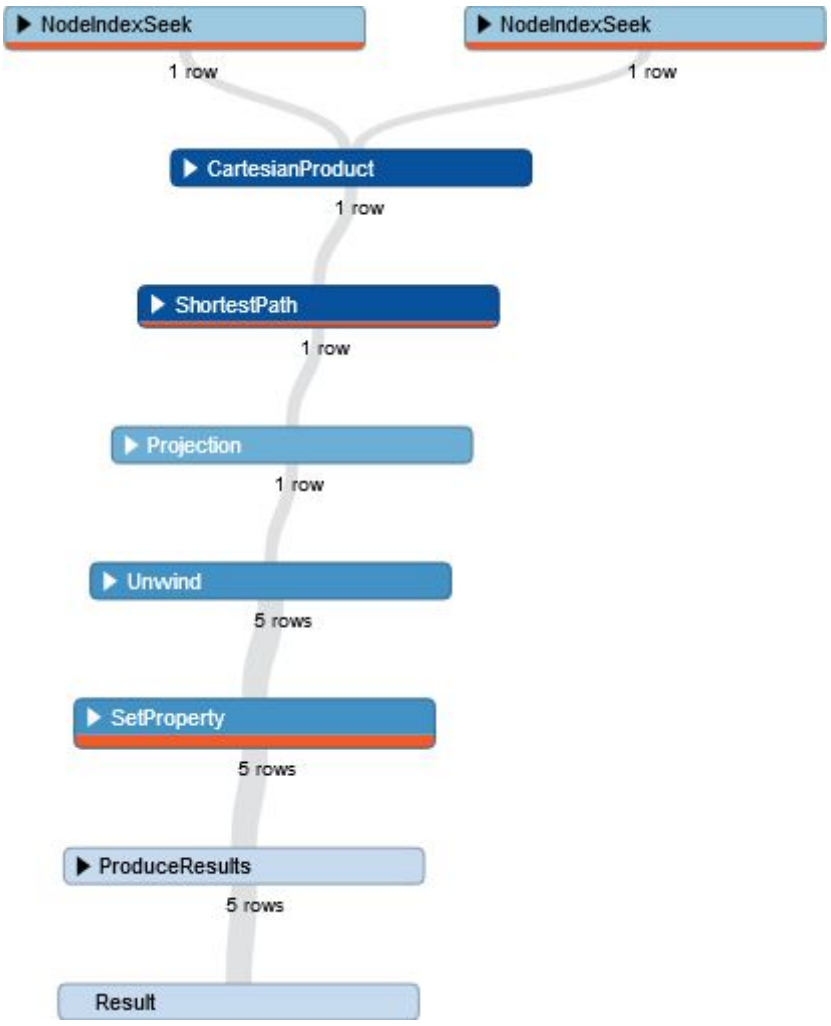
Optymalizacja będzie polegała na dodaniu indeksu na polu name.

Bez indeksu



Cypher version: CYPHER 3.5, planner: COST, runtime: INTERPRETED. 516 total db hits in 50 ms.

Z indeksem



Cypher version: CYPHER 3.5, planner: COST, runtime: INTERPRETED. 10 total db hits in 10 ms.

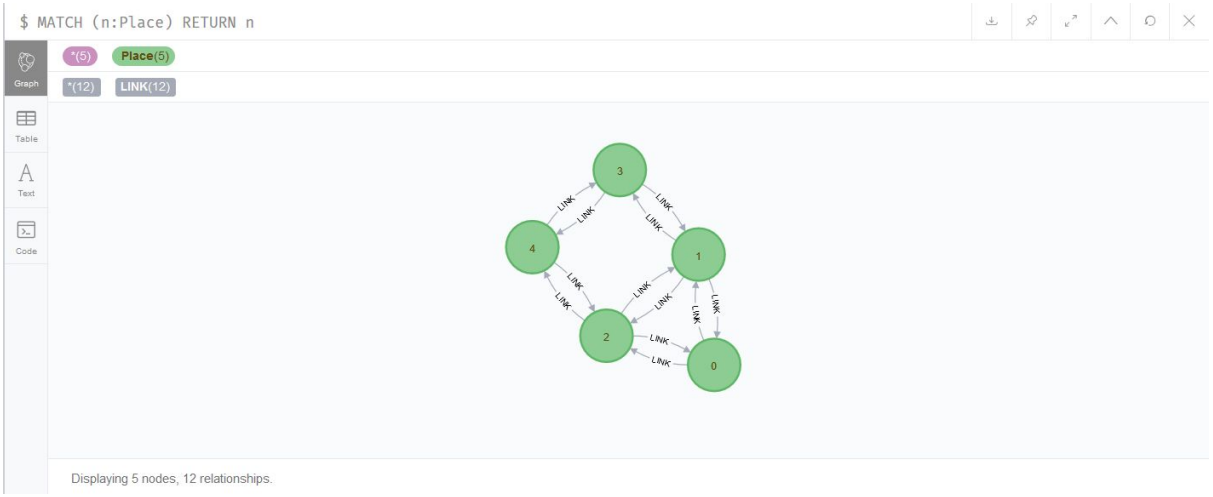
Dodanie indeksu poprawiło czas wykonania z 50ms do 10ms.

13. Napisać kod, który wygeneruje drzewo rozpinające z bazy (z poziomu javy lub pythona, nie musi być minimalne) - (można wygenerować własny mały graf do realizacji zadania, zadanie na liczbę punktów powyżej 5.0)

Napiszę w javie kod który przy użyciu algorytmy Prima wygeneruje minimalne drzewo rozpinające

Generuję spójny nieskierowany graf z wagami;

```
MERGE (a:Place {id:0})
MERGE (b:Place {id:1})
MERGE (c:Place {id:2})
MERGE (d:Place {id:3})
MERGE (e:Place {id:4})

MERGE (d)-[:LINK {cost:4}]→(b)
MERGE (b)-[:LINK {cost:4}]→(d)
MERGE (d)-[:LINK {cost:6}]→(e)
MERGE (e)-[:LINK {cost:6}]→(d)
MERGE (b)-[:LINK {cost:1}]→(a)
MERGE (a)-[:LINK {cost:1}]→(b)
MERGE (b)-[:LINK {cost:3}]→(c)
MERGE (c)-[:LINK {cost:3}]→(b)
MERGE (a)-[:LINK {cost:2}]→(c)
MERGE (c)-[:LINK {cost:2}]→(a)
MERGE (c)-[:LINK {cost:5}]→(e)
MERGE (e)-[:LINK {cost:5}]→(c)
```



Kod w javie

```
private void zadanie13(){
    Session session = driver.session();
    Transaction tx = session.beginTransaction();

    String query = "MATCH (a)-[r]->(b) RETURN a.id, b.id, r.cost";

    StatementResult result = tx.run(query);

    int n =  tx.run( s: "MATCH (a) RETURN a").list().size();

    MST t = new MST(n);
    int[][] graph = new int[n][n];
    while ( result.hasNext() )
    {
        Record record = result.next();
        graph[record.get("a.id").asInt()][record.get("b.id").asInt()]=record.get("r.cost").asInt();
    }
    t.primMST(graph);

    tx.success();
    tx.close();
}
```

```java
package neo4j;

import java.lang.*;

public class MST {
    private int V ;

    public MST(int V) {
        this.V = V;
    }

    private int minKey(int[] key, Boolean[] mstSet)
    {
        int min = Integer.MAX_VALUE, min_index = -1;
        for (int v = 0; v < V; v++)
            if (!mstSet[v] && key[v] < min) {
                min = key[v];
                min_index = v;
            }
        return min_index;
    }

    public void primMST(int[][] graph)
    {
        int[] parent = new int[V];
        int[] key = new int[V];
        Boolean[] mstSet = new Boolean[V];

        for (int i = 0; i < V; i++) {
            key[i] = Integer.MAX_VALUE;
            mstSet[i] = false;
        }
        key[0] = 0;
        parent[0] = -1;

        for (int count = 0; count < V - 1; count++) {
            int u = minKey(key, mstSet);
            mstSet[u] = true;
            for (int v = 0; v < V; v++)
                if (graph[u][v] != 0 && !mstSet[v] && graph[u][v] < key[v]) {
                    parent[v] = u;
                    key[v] = graph[u][v];
                }
        }
        System.out.println("Edge \tWeight");
        for (int i = 1; i < V; i++)
            System.out.println(parent[i] + " - " + i + "\t" + graph[i][parent[i]]);
    }
}
```

Rezultat

Edge	Weight
0 - 1	1
0 - 2	2
1 - 3	4
2 - 4	5

Na wygenerowany wcześniej graf nanoszę rezultat działania algorytmu: