

# Stacja paliw

## Opis

Sieci stacji paliw potrzebna jest baza danych do przechowywania informacji o sprzedaży, magazynowaniu i logistyce paliw naftowych, pracownikach, produktach sprzedaży detalicznej oraz nieruchomościach.

Nieruchomości muszą mieć określony typ: stacja lub magazyn, adres oraz kierownika. Pracownicy muszą mieć przypisane miejsce zatrudnienia oraz stanowisko. Każdy pracownik ma służbowy e-mail, który służy również jako identyfikator przy połączeniu z zewnętrznymi systemami. Pracownicy magazynów mogą tworzyć i wysyłać transporty. Pracownicy stacji paliw mogą zamawiać paliwa, zarządzać sprzedażą. Zamówienia muszą mieć określony towar oraz ilość, magazyn oraz stację dostawy oraz pracowników zamawiających i realizujących zamówienie. Nieruchomości muszą mieć określone stany magazynowe, zawierające towar oraz jego ilość. Na danych stacjach pracownicy muszą mieć możliwość określenia cen surowców.

## Założenia bazy

### Pracownicy

- muszą mieć określone imię i nazwisko
- muszą mieć określone stanowisko
- muszą mieć określony email służbowy
- muszą mieć określoną stację lub magazyn zatrudnienia

### Pracownicy magazynów

- muszą mieć możliwość podglądu zamówień do realizacji
- muszą mieć możliwość zmiany statusu zamówienia na “w realizacji”

### Kierowcy

- muszą mieć możliwość zmiany statusu zamówienia na “w dostawie”

### Pracownicy stacji

- muszą mieć możliwość składania zamówień na konkretne towary
- muszą mieć możliwość zmiany statusu zamówienia na “zakończone”

### Kierownicy stacji

- muszą mieć możliwość zmieniania cen towarów

## Stacje

- muszą mieć określony typ: stacja lub magazyn
- muszą mieć określony adres

- muszą mieć określonego kierownika
- muszą mieć określone stany magazynowe konkretnych towarów
- muszą mieć określone ceny konkretnych towarów

## **Magazyny**

- muszą mieć określony typ: stacja lub magazyn
- muszą mieć określony adres
- muszą mieć określonego kierownika
- muszą mieć określone stany magazynowe konkretnych towarów

## **Towary**

- muszą mieć określone typy paliwa
- muszą mieć określone nazwy marketingowe

## **Zamówienia**

- muszą mieć określony status
- muszą mieć określoną listę towarów
- muszą mieć określony magazyn realizujący zamówienie
- muszą mieć określoną stację docelową
- muszą mieć określonego pracownika realizującego zamówienie
- muszą mieć określonego pracownika składającego zamówienie
- muszą mieć określonego kierowcę przewożącego zamówienie

# Struktura bazy danych

Nazwa wierzchołka	gas_station	
Nazwa relacji	Wierzchołek docelowy	Nazwa pola w relacji
sells	product	price
		stock
Nazwa pola		
city		
country		
number		
region		
street		
zip_code		

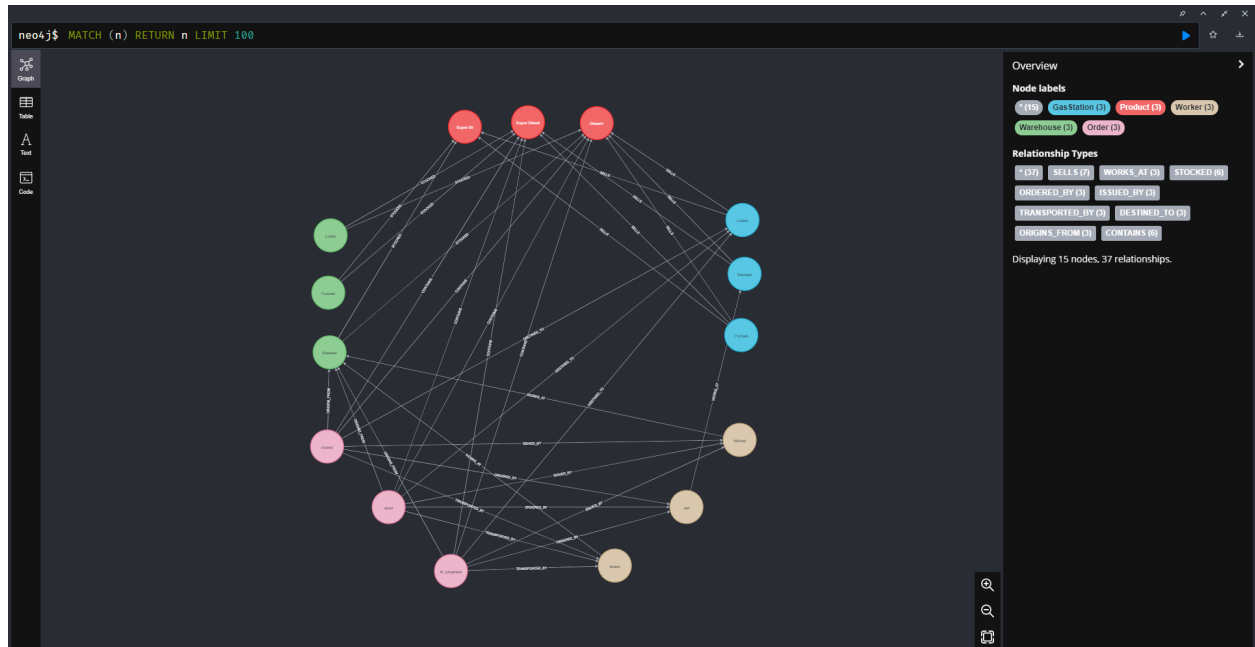
<b>Nazwa wierzchołka</b>	warehouse	
<b>Nazwa relacji</b>	<b>Wierzchołek docelowy</b>	<b>Nazwa pola w relacji</b>
stocked	product	stock
<b>Nazwa pola</b>		
city		
country		
number		
region		
street		
zip_code		

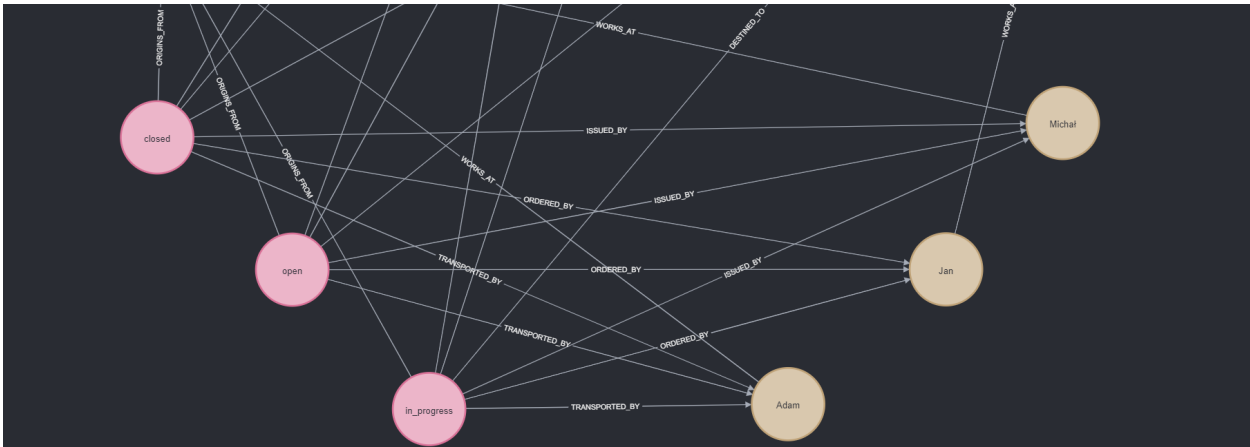
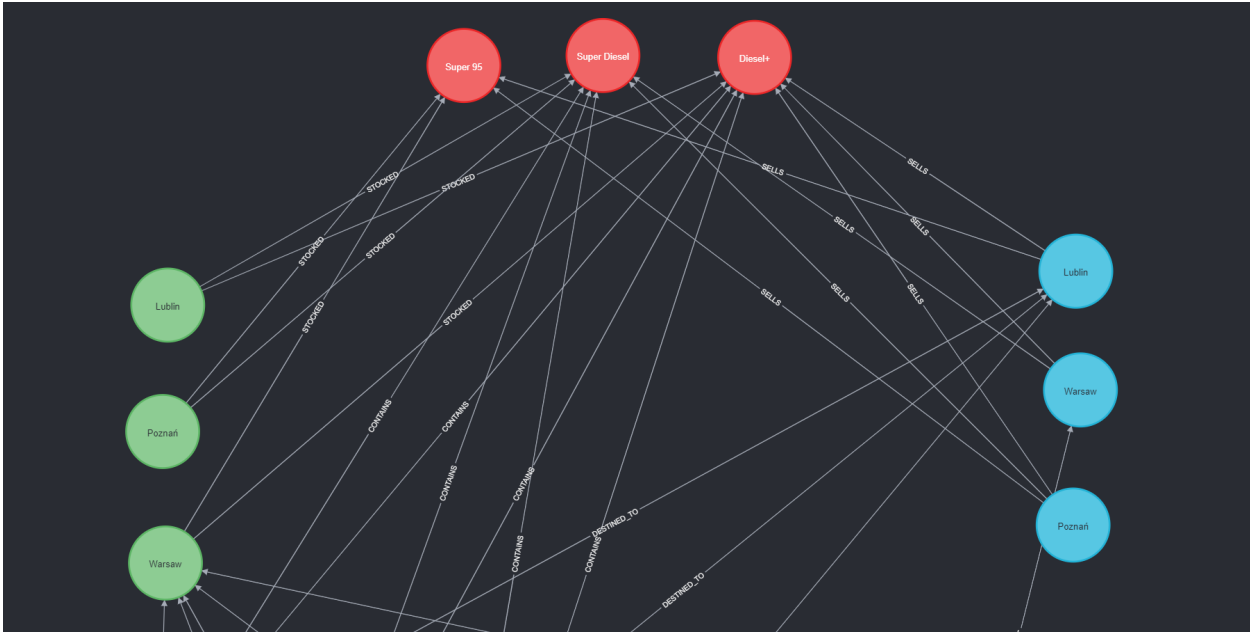
Nazwa wierzchołka	worker	
Nazwa relacji	Wierzchołek docelowy	Nazwa pola w relacji
works at	gas_station/warehouse	job_position
Nazwa pola		
email		
name		
surname		

Nazwa wierzchołka	order	
Nazwa relacji	Wierzchołek docelowy	Nazwa pola w relacji
contains	product	stock
ordered by	worker	—
transported by	worker	—
issued by	worker	—
origins from	warehouse	—
destined to	gas_station	—
Nazwa pola		
order_status		

Nazwa wierzchołka	product
Nazwa pola	
name	
type	

# Graf bazy danych

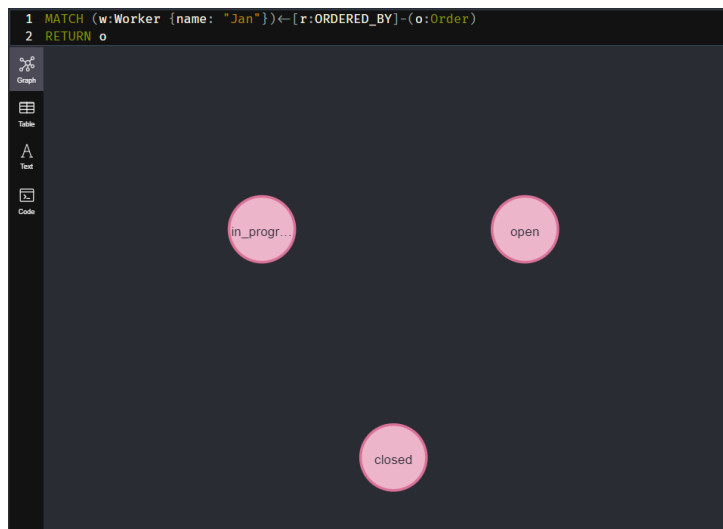




# Zapytania do bazy danych

Zwróć wszystkie zamówienia, które zamawiał Jan.

```
MATCH (w:Worker {name: "Jan"})<-[r:ORDERED_BY]-(o:Order)
RETURN o
```



Zwróć wszystkie stacje paliw z Lublina.

```
MATCH (g:Gas_station {city: "Lublin"})
RETURN g
```



Zwróć liczbę i rodzaj relacji dla każdego pracownika.

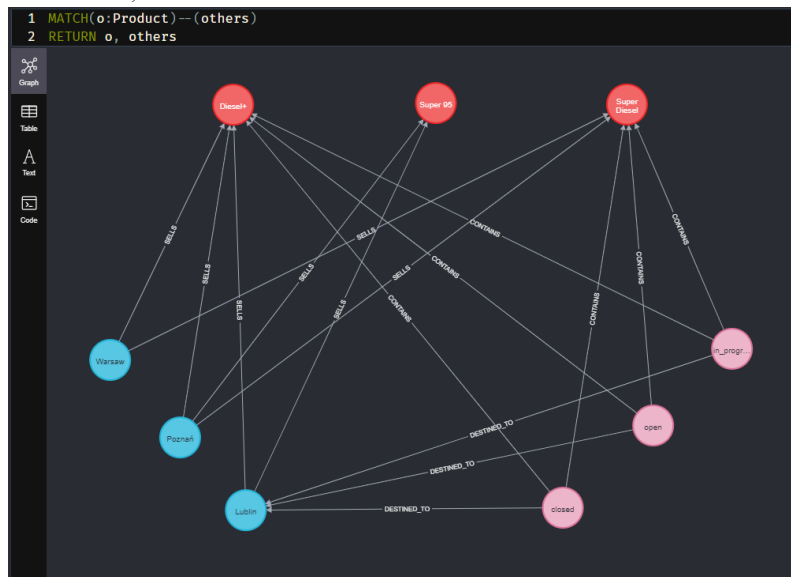
```
MATCH(w:Worker)-[r]->()
RETURN w.name, type(r), count(r)
```

	w.name	type(r)	count(r)
	"Jan"	"WORKS_AT"	1
	"Adam"	"WORKS_AT"	1
	"Michał"	"WORKS_AT"	1

### Zwróć wszystkie relacje z produktami

```
MATCH(o:Product)--(others)
```

```
RETURN o, others
```



### Zwróć stan paliw w stacji w Lublinie

```
MATCH (n:GasStation {city:"Lublin"})-[r:SELLS]->(p:Product)
```

```
RETURN p.name, r.stock
```

```
1 MATCH (n:GasStation {city:"Lublin"})-[r:SELLS]->(p:Product)
2 RETURN p.name, r.stock
```

	p.name	r.stock
1	"Super 95"	4000
2	"Diesel+"	6000

### Zwróć ceny paliw stacji w Poznaniu

```
MATCH (n:GasStation {city:"Poznań"})-[r:SELLS]->(p:Product)
```

```
RETURN p.name, r.price
```

```
1 MATCH (n:GasStation {city:"Poznań"})-[r:SELLS]->(p:Product)
2 RETURN p.name, r.price
```

	p.name	r.price
1	"Super Diesel"	6.2
2	"Diesel+"	7.7
3	"Super 95"	6.7

### Zwróć średnią cenę Diesel+

```
Match (p:Product {name:"Diesel+"})<-[s:SELLS]-(g:GasStation)
```

```
RETURN avg(s.price)
```



```

1 Match (p:Product {name:"Diesel+"})←[s:SELLS]-(g:GasStation)
2 RETURN avg(s.price)

```

avg(s.price)
7.006666666666667

### Zwróć najdroższe paliwo i jego cenę

**MATCH** (g:GasStation)-[s:SELLS]->(p:Product)

**RETURN** p.name, max(s.price)

```

1 MATCH (g:GasStation)-[s:SELLS]->(p:Product)
2 RETURN p.name, max(s.price)

```

p.name	max(s.price)
"Super 95"	6.7
"Diesel+"	7.7
"Super Diesel"	6.7

### Zwróć największy stan magazynowy super 95

**MATCH** (w:Warehouse)-[s:STOCKED]->(p:Product{name:"Super 95"})

**RETURN** w.city,w.street, max(w.number)

```

1 MATCH (w:Warehouse)-[s:STOCKED]->(p:Product{name:"Super 95"})
2 RETURN w.city,w.street, max(w.number)

```

w.city	w.street	max(w.number)
"Poznań"	"Warszawska"	"34"
"Warsaw"	"Marszałkowska"	"51"

### Zwróć posortowane malejąco paliwa na stacji Lublin

**MATCH** (g:GasStation{city:"Lublin"})-[s:SELLS]->(p:Product)

**RETURN** p

**ORDER BY** p.name DESC

```

neo4j$ MATCH (g:GasStation{city:"Lublin"})-[s:SELLS]->(p:Product) RETURN p ORDER BY p.name DESC

```

Graph

Table

Super 95	Diesel+
----------	---------

Text

Code

# Uaktualnienia bazy danych

Zmień imię pracownika Adam na Marek

```
MATCH (w:Worker {name: "Adam"})
```

```
SET w.name = "Marek"
```

```
RETURN w
```

```
1 MATCH (w:Worker {name: "Adam"})
2 SET w.name = "Marek"
3 RETURN w
```



The interface shows a sidebar with icons for Graph, Table, Text, and Code. The main area displays a single node labeled 'Marek' in a circular shape.

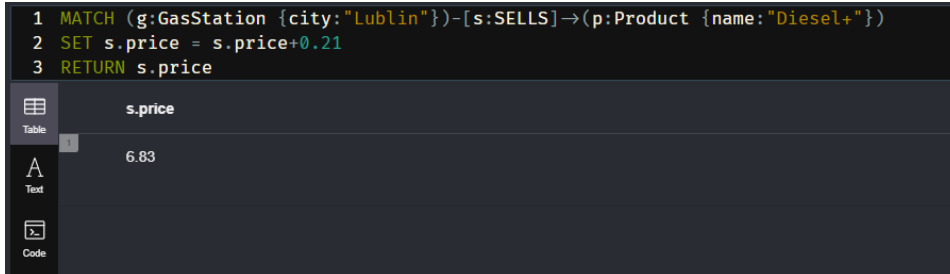
Zwiększ cenę Diesel+ na stacji lublin o 21 groszy

```
MATCH (g:GasStation {city: "Lublin"})-[s:SELLS]->(p:Product {name: "Diesel+"})
```

```
SET s.price = s.price+0.21
```

```
RETURN s.price
```

```
1 MATCH (g:GasStation {city: "Lublin"})-[s:SELLS]->(p:Product {name: "Diesel+"})
2 SET s.price = s.price+0.21
3 RETURN s.price
```



The interface shows a sidebar with icons for Table, Text, and Code. The main area displays a table with one row: s.price, 6.83.

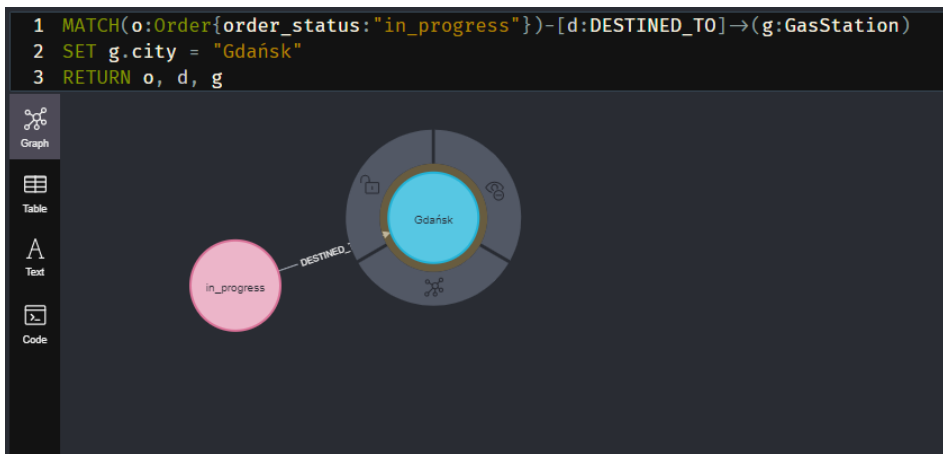
Zmiana nazwy docelowej stacji dla zamówienia o statusie "in\_progress"

```
MATCH(o:Order{order_status:"in_progress"})-[d:DESTINED_TO]->(g:GasStation)
```

```
SET g.city = "Gdańsk"
```

```
RETURN o, d, g
```

```
1 MATCH(o:Order{order_status:"in_progress"})-[d:DESTINED_TO]->(g:GasStation)
2 SET g.city = "Gdańsk"
3 RETURN o, d, g
```



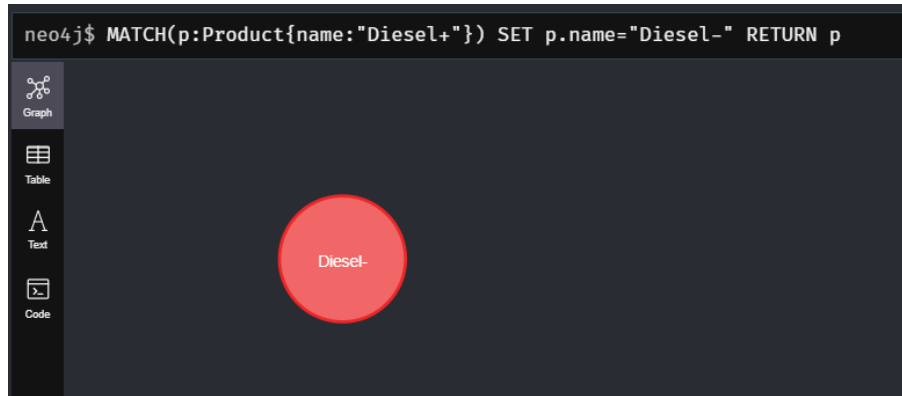
The interface shows a sidebar with icons for Graph, Table, Text, and Code. The main area displays a graph with a node labeled 'in\_progress' connected to a node labeled 'Gdańsk' via a relationship labeled 'DESTINED\_TO'.

Zmiana nazwy paliwa z Diesel+ na Diesel-

```
MATCH(p:Product{name:"Diesel+"})
```

```
SET p.name="Diesel-"
```

```
RETURN p
```

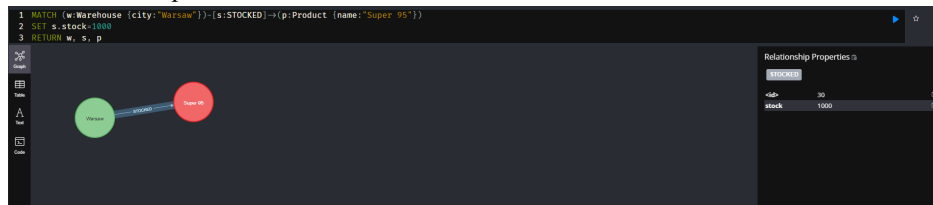


Zmiana stanu magazynowego dla paliwa Super 95 w Warszawie

```
MATCH (w:Warehouse {city:"Warsaw"})-[s:STOCKED]->(p:Product {name:"Super 95"})
```

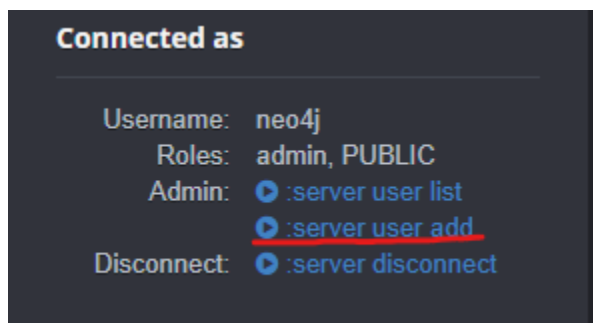
```
SET s.stock=1000
```

```
RETURN w, s, p
```



# Użytkownicy

## Tworzenie nowych użytkowników



Rys. Klikamy przycisk tworzenia nowego użytkownika

\$ :server user list

Username	Add Role	Current Role(s)	Status	Action	Password Change	Delete
Jakub		admin PUBLIC	Active	Suspend	-	Remove
Marcin		reader PUBLIC	Active	Suspend	-	Remove
Michał		editor PUBLIC	Active	Suspend	-	Remove
neo4j		admin PUBLIC	Active	Suspend	-	Remove

Add new user

Rys.Powstała lista użytkowników

## Jakub - admin

\$ :server user add

Add user

Add a user to the current database

Username

Jakub

Password

\*\*\*\*

Confirm password

\*\*\*\*

Roles

admin

☐ Force password change

Add User

See user list

Rys.Tworzenie użytkownika Jakub

## Michał - editor

\$ :server user add

Add user

Add a user to the current database

Username

Michał

Password

\*\*\*\*

Confirm password

\*\*\*\*

Roles

editor

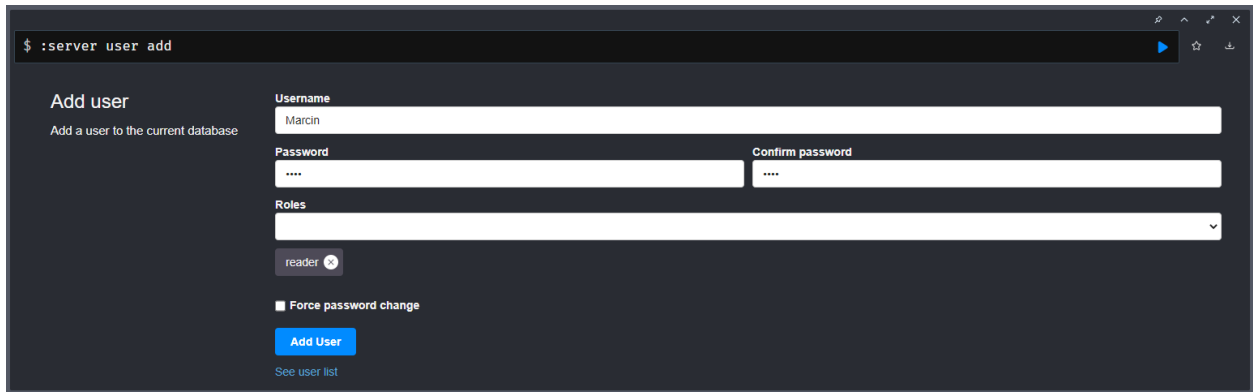
☐ Force password change

Add User

See user list

Rys.Tworzenie użytkownika Michał

## Marcin - reader

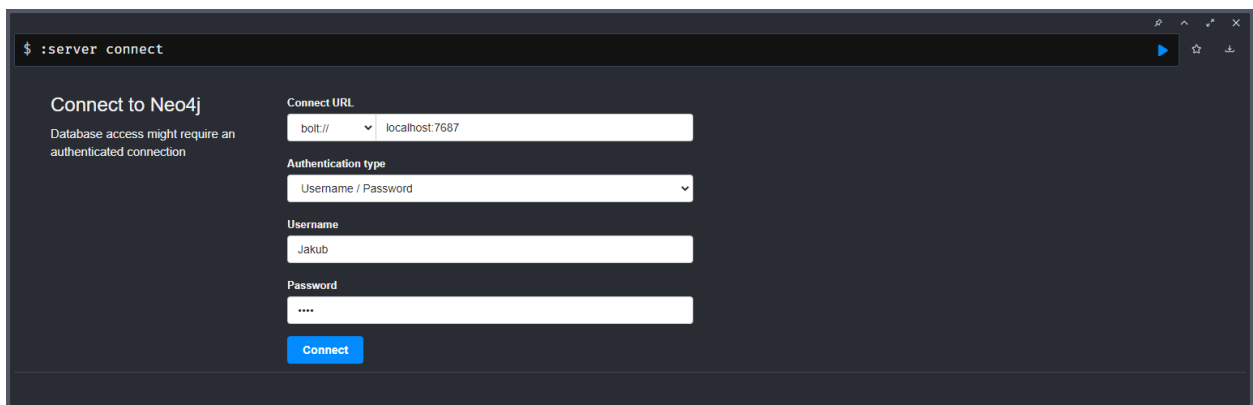


The screenshot shows the Neo4j web interface for adding a new user. The title bar indicates the command `$ :server user add`. On the left, the section "Add user" includes the instruction "Add a user to the current database". The main form contains the following fields: "Username" with the value "Marcin", "Password" and "Confirm password" both masked with four asterisks, and a "Roles" dropdown menu currently showing "reader". Below these fields is a checkbox for "Force password change" which is unchecked. At the bottom of the form is a blue "Add User" button. A link "See user list" is located at the very bottom of the interface.

Rys.Tworzenie użytkownika Marcin

## Logowanie użytkowników

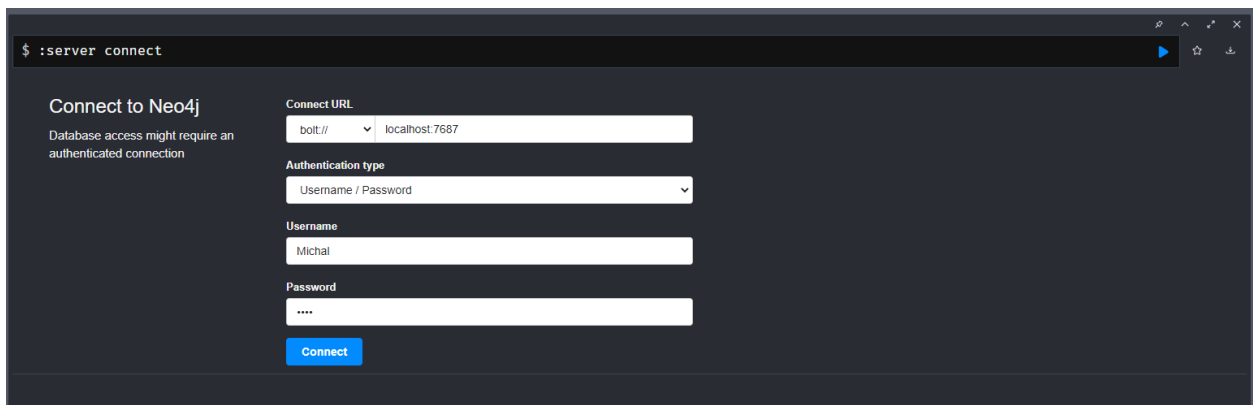
### Jakub - admin



The screenshot shows the Neo4j web interface for connecting to a database. The title bar indicates the command `$ :server connect`. On the left, the section "Connect to Neo4j" includes the instruction "Database access might require an authenticated connection". The main form contains the following fields: "Connect URL" with a dropdown set to "bolt://" and the text "localhost:7687", "Authentication type" dropdown set to "Username / Password", "Username" with the value "Jakub", and "Password" masked with four asterisks. A blue "Connect" button is positioned below the password field.

Rys.Logowanie użytkownika Jakub

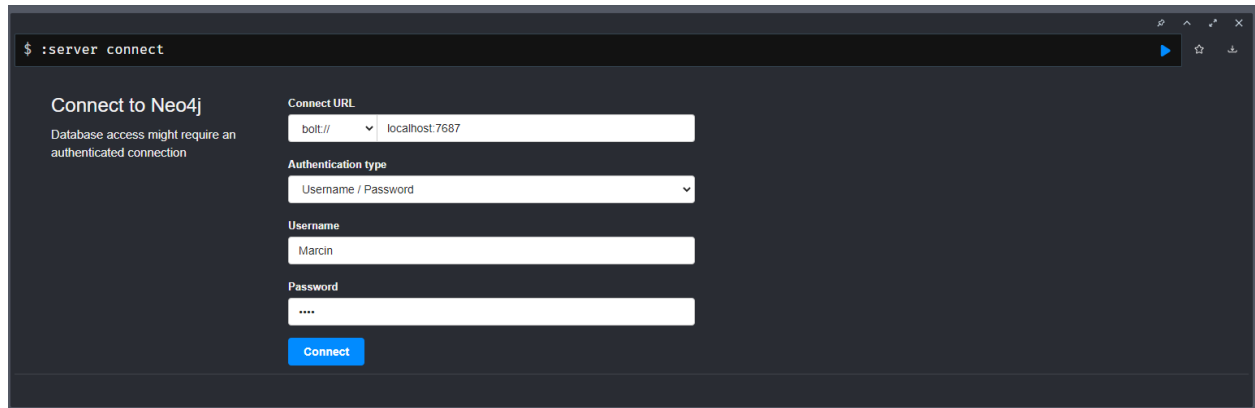
### Michal - editor



The screenshot shows the Neo4j web interface for connecting to a database, similar to the previous one. The title bar indicates the command `$ :server connect`. On the left, the section "Connect to Neo4j" includes the instruction "Database access might require an authenticated connection". The main form contains the following fields: "Connect URL" with a dropdown set to "bolt://" and the text "localhost:7687", "Authentication type" dropdown set to "Username / Password", "Username" with the value "Michal", and "Password" masked with four asterisks. A blue "Connect" button is positioned below the password field.

Rys.Logowanie użytkownika Michal

## Marcin - reader

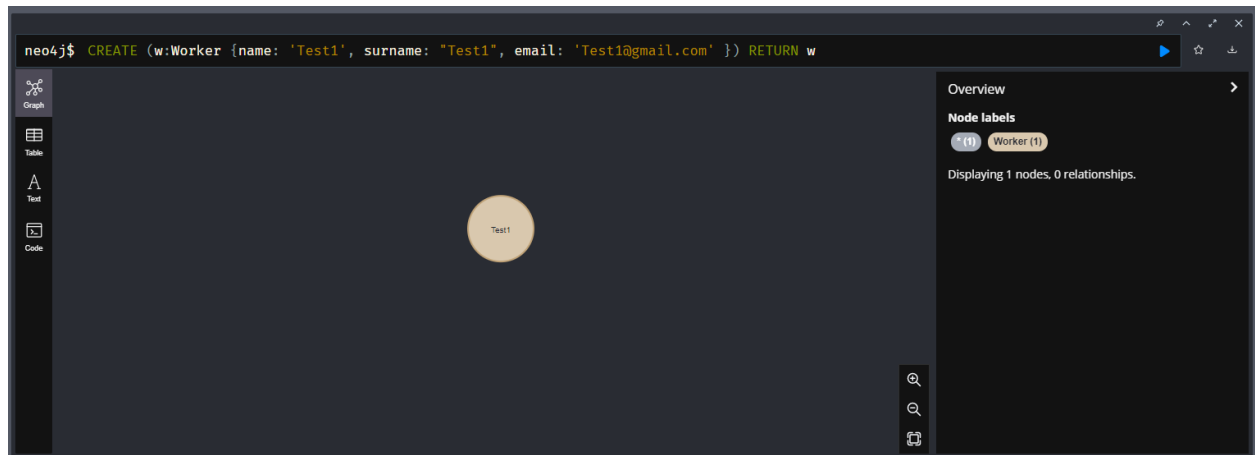


The image shows a 'Connect to Neo4j' dialog box. It has a title bar with standard window controls. The main area contains a 'Connect URL' field with a dropdown set to 'bolt://' and a text field with 'localhost:7687'. Below this is an 'Authentication type' dropdown set to 'Username / Password'. There are two text input fields: 'Username' with the value 'Marcin' and 'Password' with masked characters '\*\*\*\*'. A blue 'Connect' button is at the bottom right. On the left, there is a note: 'Database access might require an authenticated connection'.

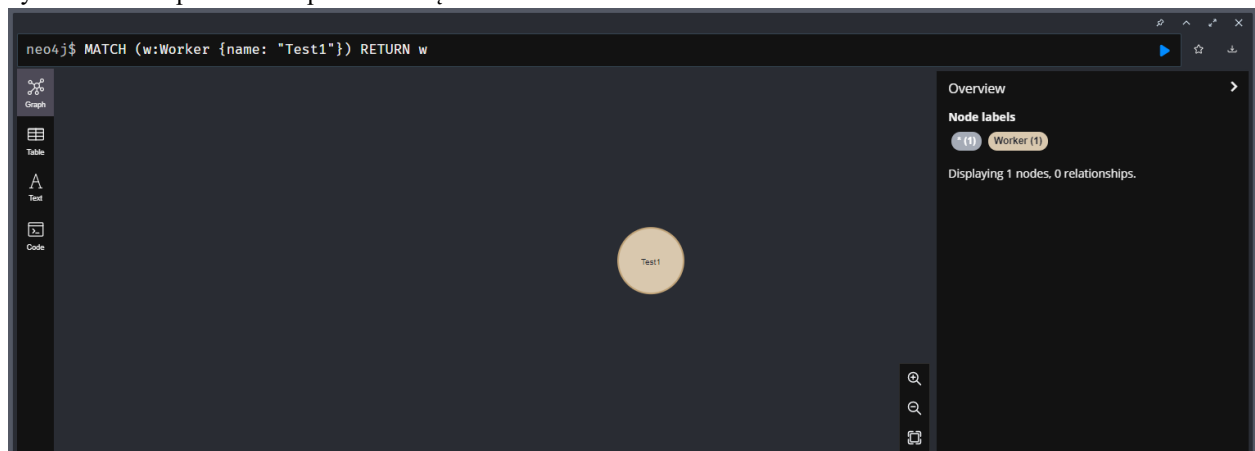
Rys. Logowanie użytkownika Marcin

## Testowanie

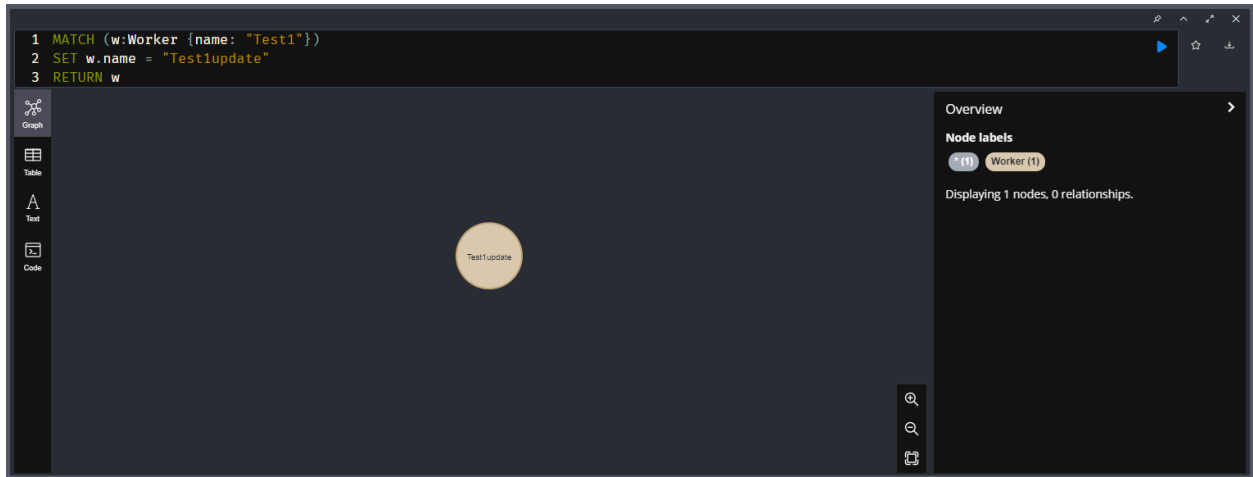
### Jakub - admin



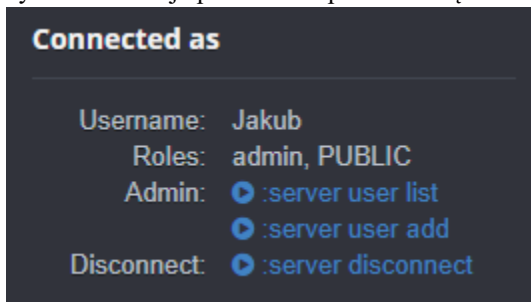
Rys. Tworzenie pracownika powiodło się



Rys. Zwracanie pracownika powiodło się

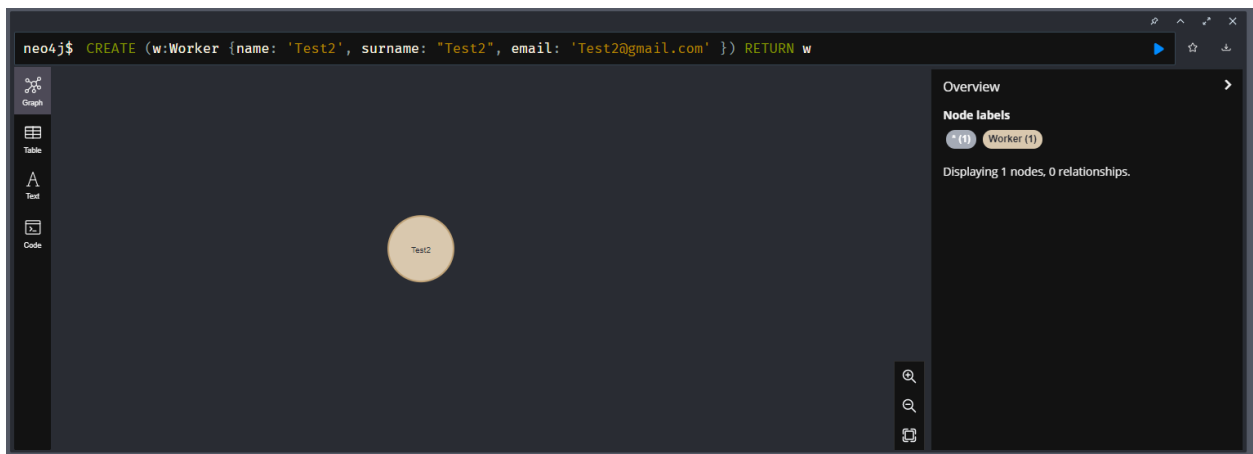


Rys. Aktualizacja pracownika powiodła się

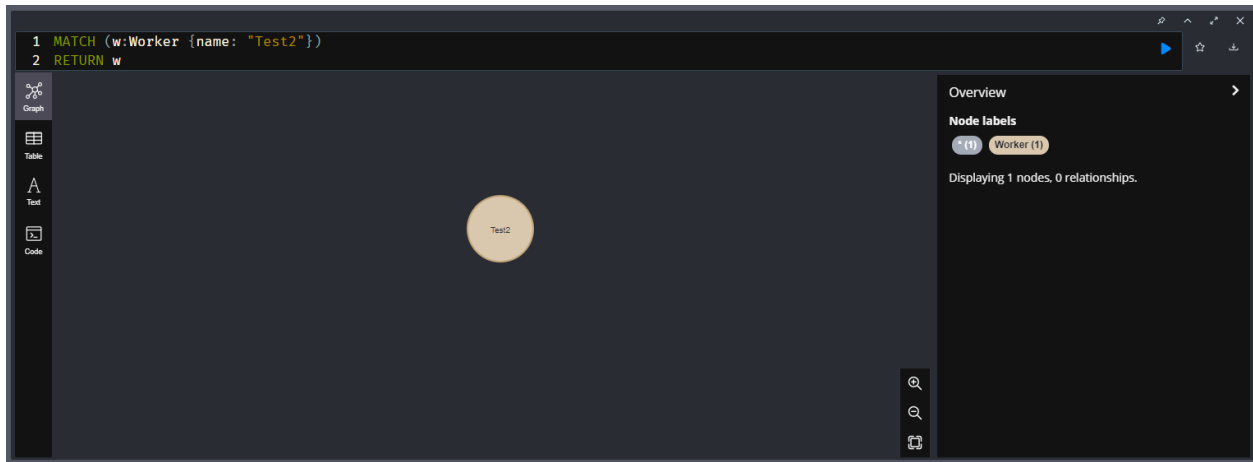


Rys. Użytkownik może zarządzać użytkownikami

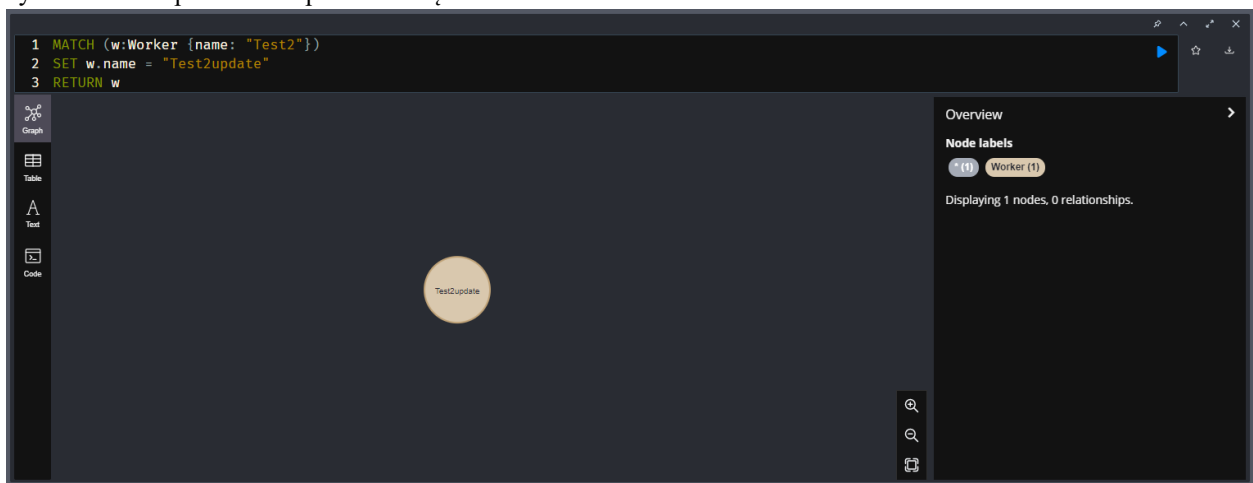
## Michał - editor



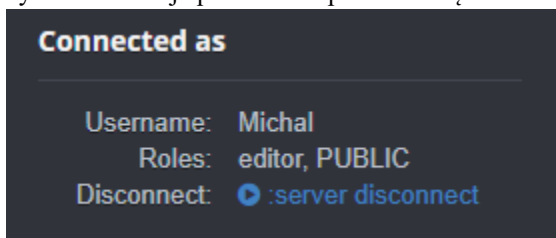
Rys. Tworzenie pracownika powiodło się



Rys. Zwracanie pracownika powiodło się

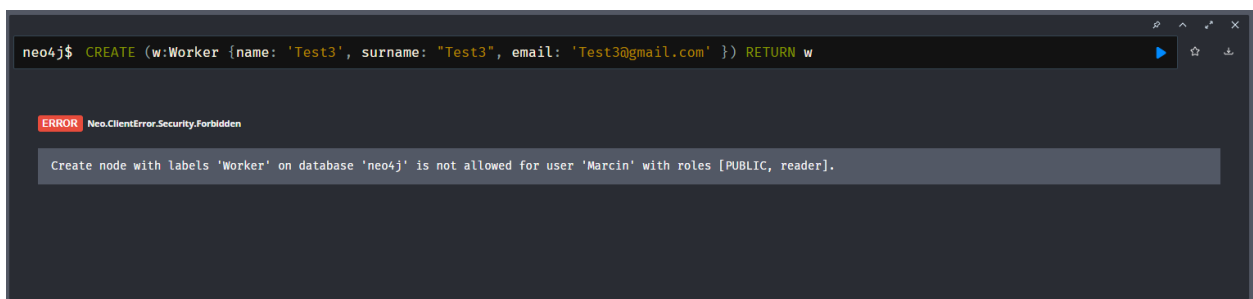


Rys. Aktualizacja pracownika powiodła się



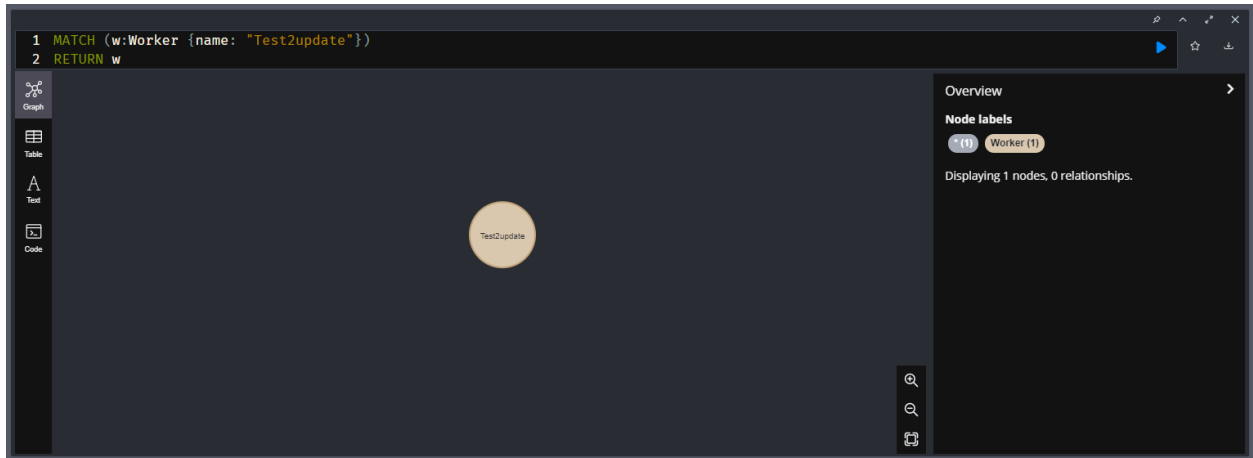
Rys. Użytkownik nie może zarządzać użytkownikami

## Marcin - reader

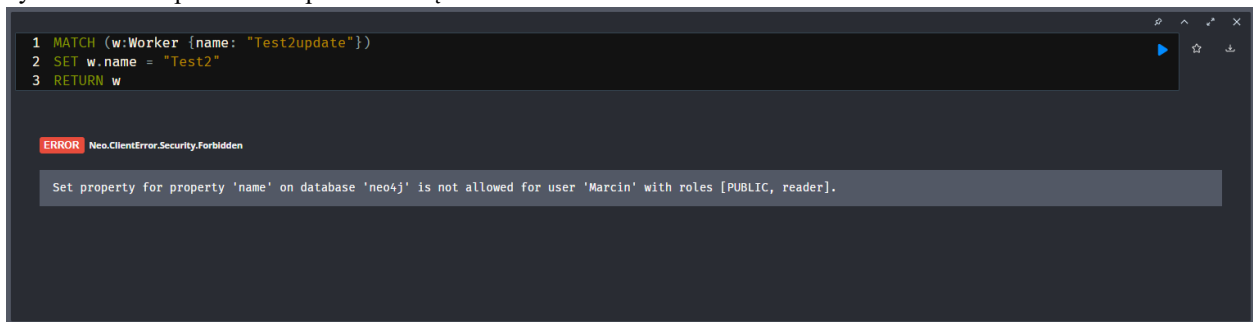


Rys. Tworzenie pracownika nie powiodło się

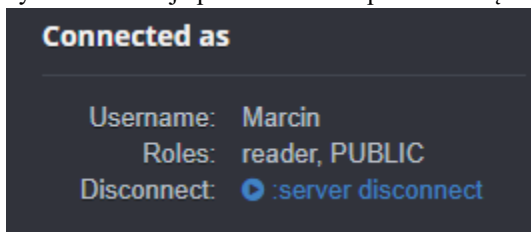




Rys. Zwracanie pracownika powiodło się



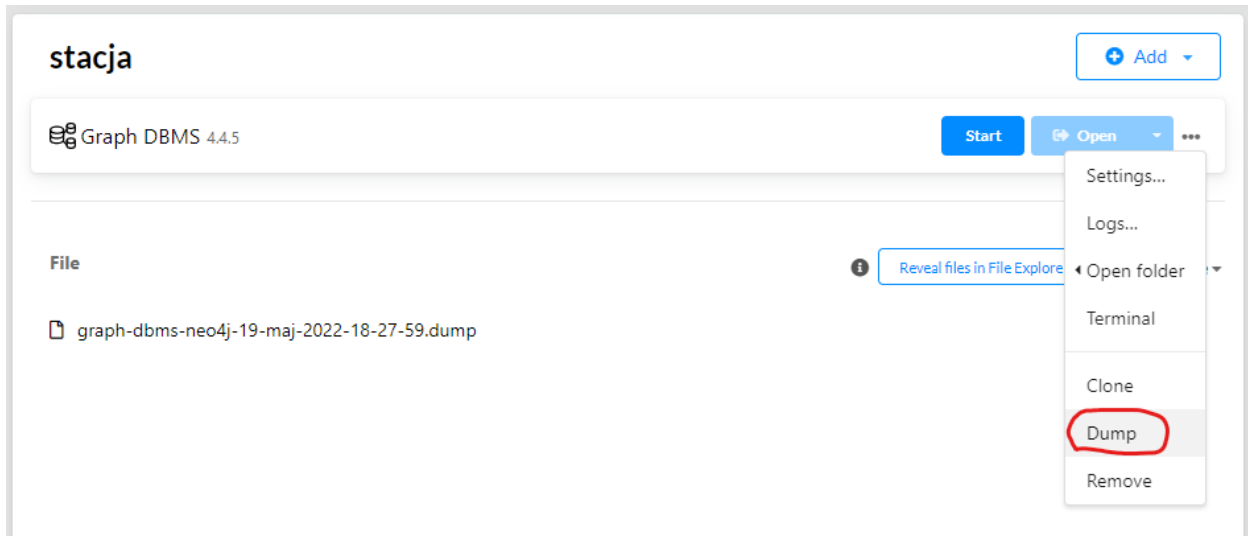
Rys. Aktualizacja pracownika nie powiodła się



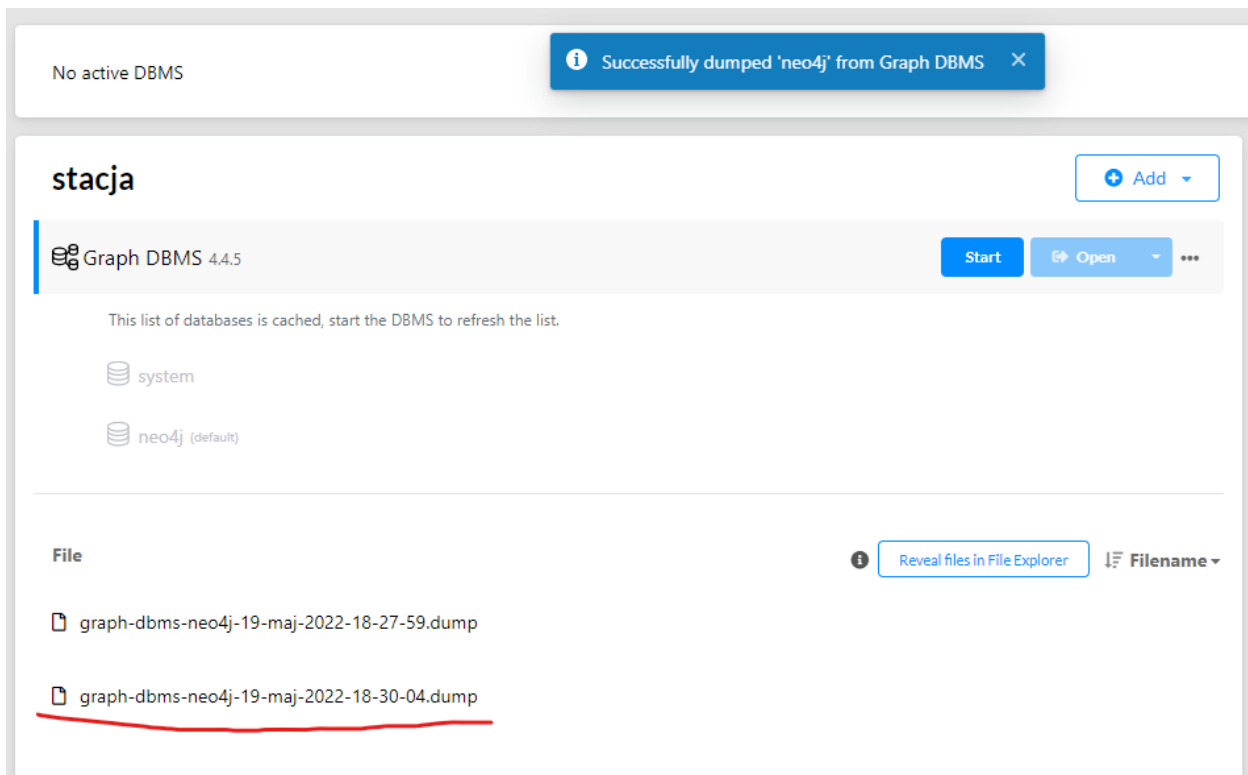
Rys. Użytkownik nie może zarządzać użytkownikami

# Eksport/Import bazy danych

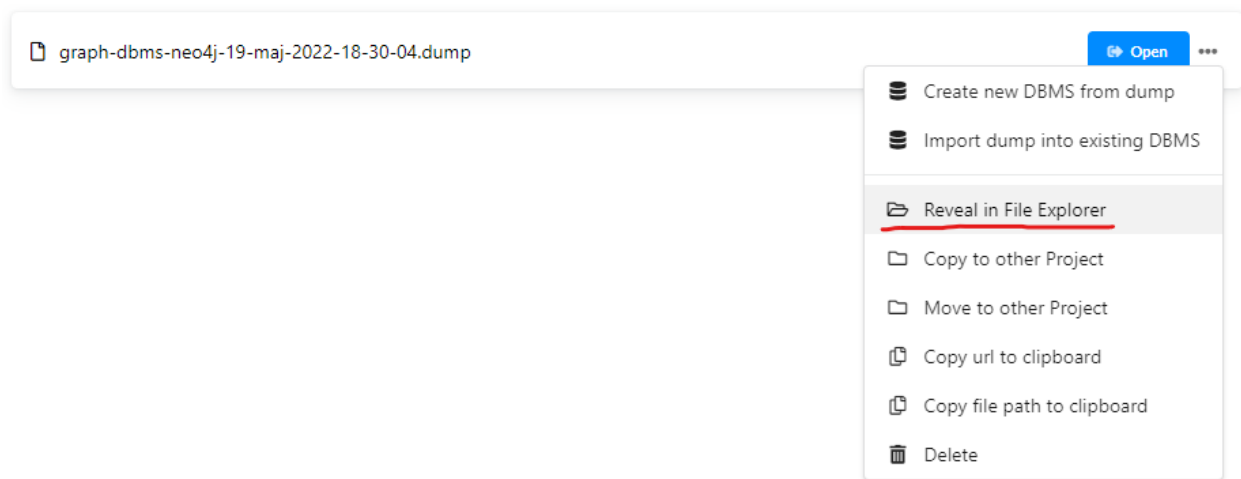
## Eksport



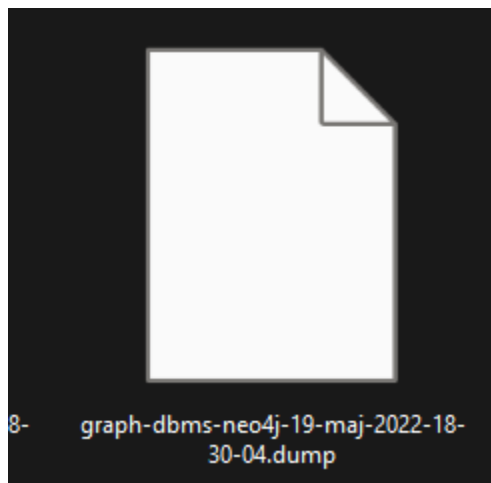
Rys. Klikamy przycisk Dump



Rys. Tworzy się nowy plik

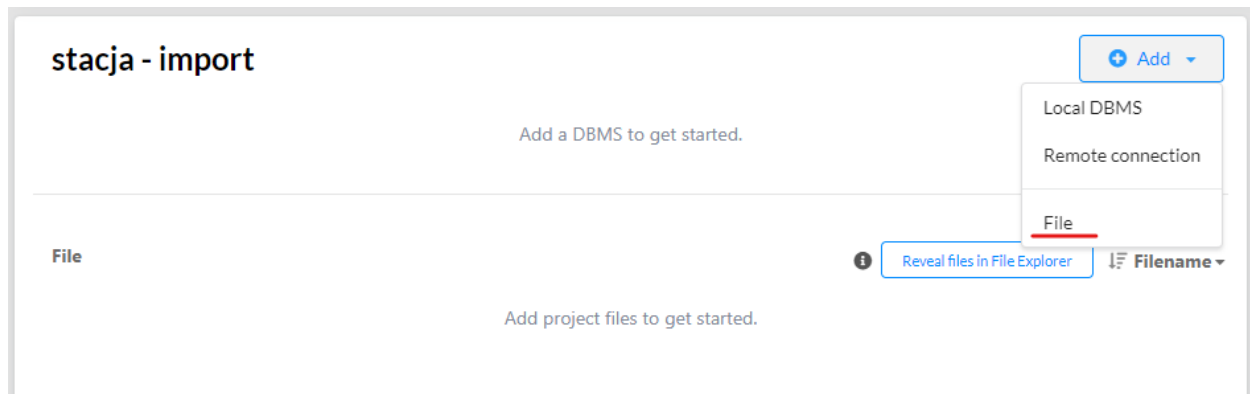


Rys. Wyświetlamy plik w explorerze

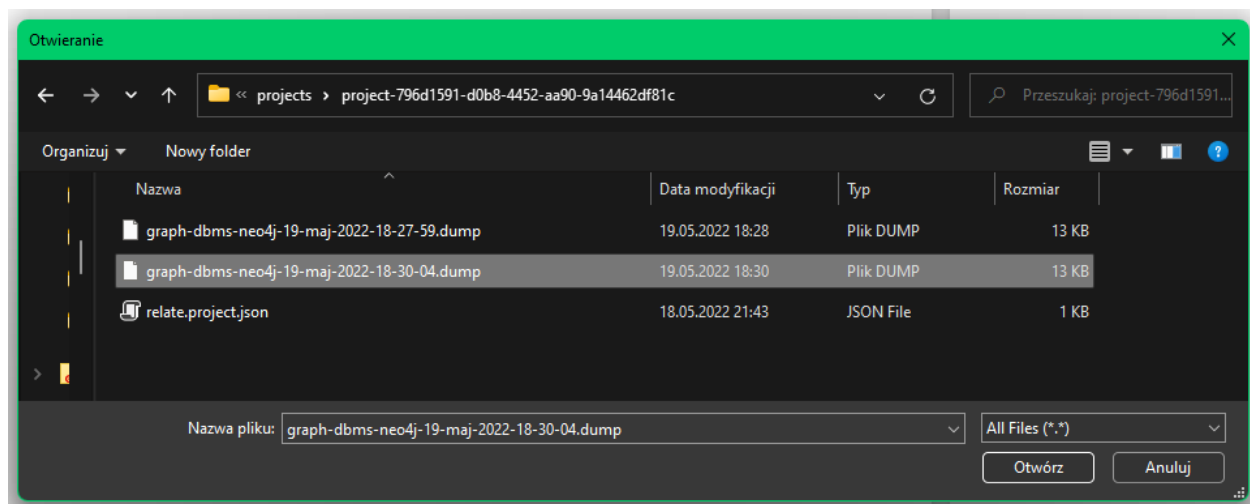


Rys. Powstały plik

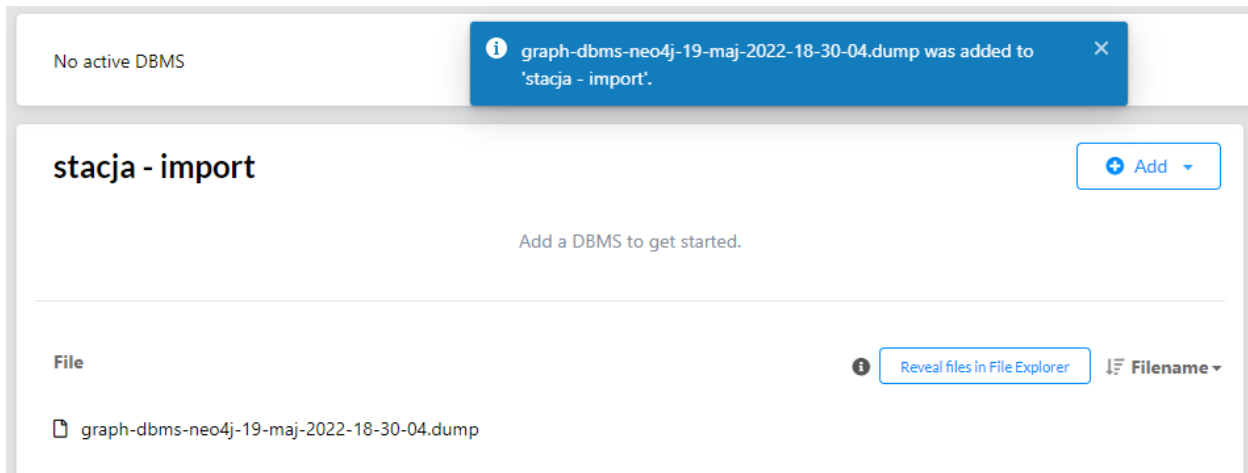
# Import



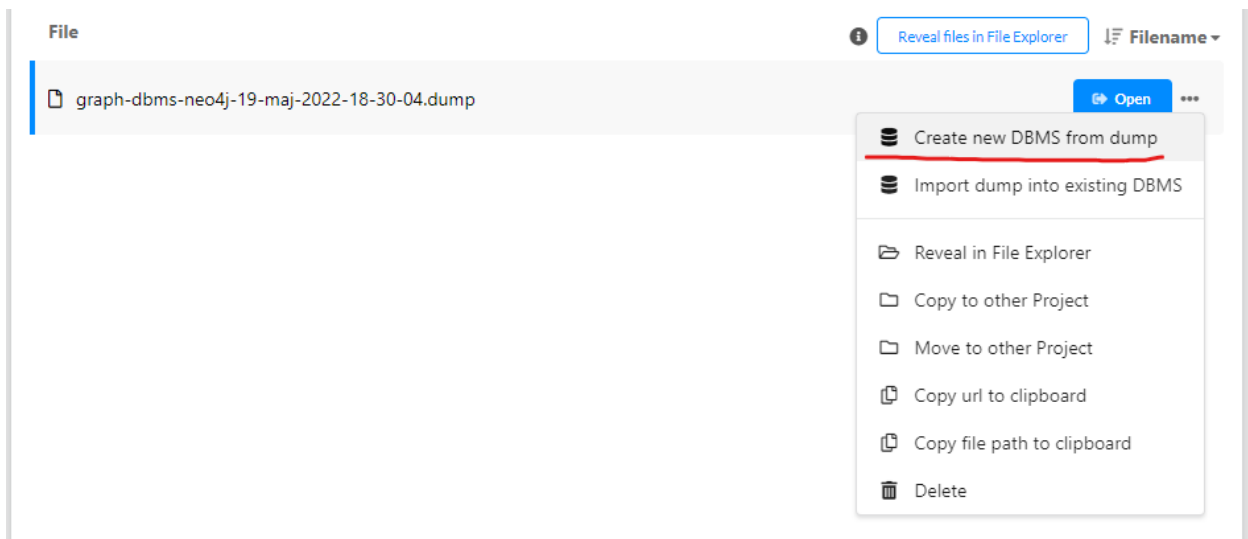
Rys. Klikamy przycisk importu pliku



Rys. Wybieramy plik



Rys. W projekcie pojawia się zaimportowany plik



Rys. Wybieramy opcję utworzenia bazy z pliku

# stacja - import

Graph DBMS

.....

4.4.5

Cancel

Create

Rys.Nazywamy nową bazę i nadajemy hasło

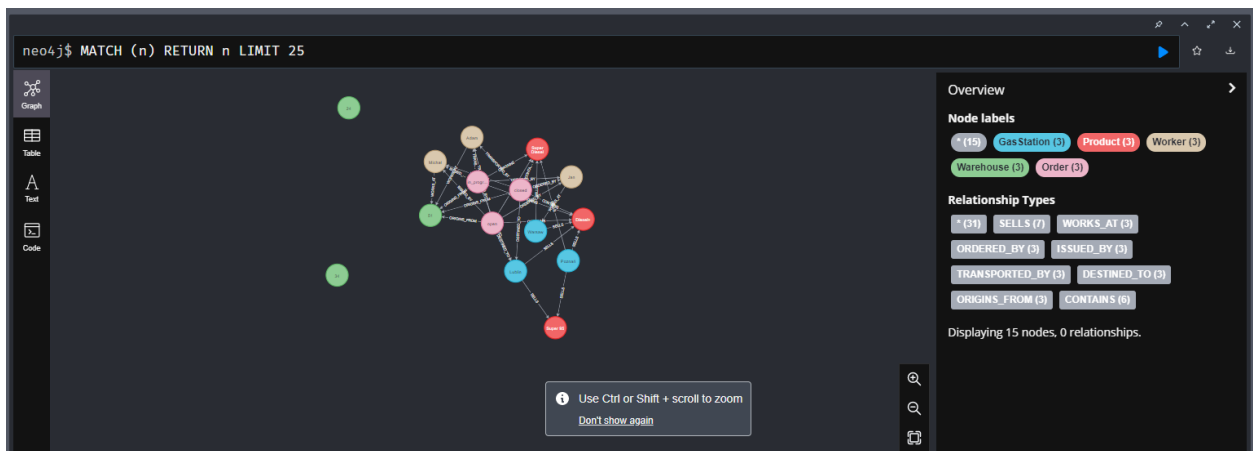
No active DBMS

Successfully loaded 'graph-dbms-neo4j-19-maj-2022-18-30-04.dump' into 'neo4j' in 'Graph DBMS'

# stacja - import

Graph DBMS 4.4.5

Rys.Baza importuje się pomyślnie



Rys.Zaimportowana baza

# Polecenia tworzenia bazy

```
:begin
CREATE CONSTRAINT ON (node:`UNIQUE IMPORT LABEL`) ASSERT (node.`UNIQUE IMPORT ID`) IS
UNIQUE;
:commit
CALL db.awaitIndexes(300);
:begin
UNWIND [{_id:0, properties:{number:"51", country:"Poland", city:"Warsaw", street:"Marszałkowska",
region:"Mazowieckie", zip_code:"00-123"}}, {_id:1, properties:{number:"24", country:"Poland", city:"Lublin",
street:"Nadbystrzycka", region:"Lubelskie", zip_code:"01-123"}}, {_id:2, properties:{number:"34",
country:"Poland", city:"Poznań", street:"Warszawska", region:"Wielkopolskie", zip_code:"24-123"}}] AS row
CREATE (n:`UNIQUE IMPORT LABEL` {'UNIQUE IMPORT ID': row._id}) SET n += row.properties SET
n:GasStation;
UNWIND [{_id:9, properties:{number:"51", country:"Poland", city:"Warsaw", street:"Marszałkowska",
region:"Mazowieckie", zip_code:"00-123"}}, {_id:10, properties:{number:"24", country:"Poland", city:"Lublin",
street:"Nadbystrzycka", region:"Lubelskie", zip_code:"01-123"}}, {_id:11, properties:{number:"34",
country:"Poland", city:"Poznań", street:"Warszawska", region:"Wielkopolskie", zip_code:"24-123"}}] AS row
CREATE (n:`UNIQUE IMPORT LABEL` {'UNIQUE IMPORT ID': row._id}) SET n += row.properties SET
n:Warehouse;
UNWIND [{_id:12, properties:{order_status:"in_progress"}}, {_id:13, properties:{order_status:"closed"}}, {_id:14,
properties:{order_status:"open"}}] AS row
CREATE (n:`UNIQUE IMPORT LABEL` {'UNIQUE IMPORT ID': row._id}) SET n += row.properties SET
n:Order;
UNWIND [{_id:3, properties:{name:"Super 95", type:"95"}}, {_id:4, properties:{name:"Diesel+", type:"Diesel"}},
{_id:6, properties:{name:"Super Diesel", type:"Diesel"}}] AS row
CREATE (n:`UNIQUE IMPORT LABEL` {'UNIQUE IMPORT ID': row._id}) SET n += row.properties SET
n:Product;
UNWIND [{_id:5, properties:{surname:"Kowalski", name:"Jan", email:"jk@gmail.com"}}, {_id:7,
properties:{surname:"Nowak", name:"Adam", email:"an@gmail.com"}}, {_id:8, properties:{surname:"Małysz",
name:"Michał", email:"mm@gmail.com"}}] AS row
CREATE (n:`UNIQUE IMPORT LABEL` {'UNIQUE IMPORT ID': row._id}) SET n += row.properties SET
n:Worker;
:commit
:begin
UNWIND [{start: {_id:0}, end: {_id:6}, properties:{price:6.7, stock:5000}}, {start: {_id:0}, end: {_id:4},
properties:{price:6.7, stock:4000}}, {start: {_id:1}, end: {_id:4}, properties:{price:6.62, stock:6000}}, {start:
{_id:1}, end: {_id:3}, properties:{price:5.7, stock:4000}}, {start: {_id:2}, end: {_id:3}, properties:{price:6.7,
stock:3000}}, {start: {_id:2}, end: {_id:4}, properties:{price:7.7, stock:5600}}, {start: {_id:2}, end: {_id:6},
properties:{price:6.2, stock:3000}}] AS row
MATCH (start:`UNIQUE IMPORT LABEL` {'UNIQUE IMPORT ID': row.start._id})
MATCH (end:`UNIQUE IMPORT LABEL` {'UNIQUE IMPORT ID': row.end._id})
CREATE (start)-[r:SELLS]->(end) SET r += row.properties;
UNWIND [{start: {_id:12}, end: {_id:7}, properties:{}}, {start: {_id:14}, end: {_id:7}, properties:{}}, {start:
{_id:13}, end: {_id:7}, properties:{}}] AS row
MATCH (start:`UNIQUE IMPORT LABEL` {'UNIQUE IMPORT ID': row.start._id})
MATCH (end:`UNIQUE IMPORT LABEL` {'UNIQUE IMPORT ID': row.end._id})
CREATE (start)-[r:TRANSPORTED_BY]->(end) SET r += row.properties;
UNWIND [{start: {_id:12}, end: {_id:9}, properties:{}}, {start: {_id:13}, end: {_id:9}, properties:{}}, {start:
{_id:14}, end: {_id:9}, properties:{}}] AS row
MATCH (start:`UNIQUE IMPORT LABEL` {'UNIQUE IMPORT ID': row.start._id})
MATCH (end:`UNIQUE IMPORT LABEL` {'UNIQUE IMPORT ID': row.end._id})
CREATE (start)-[r:ORIGINS_FROM]->(end) SET r += row.properties;
UNWIND [{start: {_id:5}, end: {_id:0}, properties:{job_position:"gas_station_worker"}}] AS row
MATCH (start:`UNIQUE IMPORT LABEL` {'UNIQUE IMPORT ID': row.start._id})
```

```

MATCH (end:`UNIQUE IMPORT LABEL`{'`UNIQUE IMPORT ID`: row.end._id'})
CREATE (start)-[r:WORKS_AT]->(end) SET r += row.properties;
UNWIND [{start: {_id:13}, end: {_id:8}, properties:{}}, {start: {_id:14}, end: {_id:8}, properties:{}}, {start:
{_id:12}, end: {_id:8}, properties:{}}] AS row
MATCH (start:`UNIQUE IMPORT LABEL`{'`UNIQUE IMPORT ID`: row.start._id'})
MATCH (end:`UNIQUE IMPORT LABEL`{'`UNIQUE IMPORT ID`: row.end._id'})
CREATE (start)-[r:ISSUED_BY]->(end) SET r += row.properties;
UNWIND [{start: {_id:14}, end: {_id:6}, properties:{stock:2000}}, {start: {_id:13}, end: {_id:6},
properties:{stock:3000}}, {start: {_id:13}, end: {_id:4}, properties:{stock:6000}}, {start: {_id:12}, end: {_id:4},
properties:{stock:4000}}, {start: {_id:12}, end: {_id:6}, properties:{stock:5000}}, {start: {_id:14}, end: {_id:4},
properties:{stock:6000}}] AS row
MATCH (start:`UNIQUE IMPORT LABEL`{'`UNIQUE IMPORT ID`: row.start._id'})
MATCH (end:`UNIQUE IMPORT LABEL`{'`UNIQUE IMPORT ID`: row.end._id'})
CREATE (start)-[r:CONTAINS]->(end) SET r += row.properties;
UNWIND [{start: {_id:14}, end: {_id:1}, properties:{}}, {start: {_id:13}, end: {_id:1}, properties:{}}, {start:
{_id:12}, end: {_id:1}, properties:{}}] AS row
MATCH (start:`UNIQUE IMPORT LABEL`{'`UNIQUE IMPORT ID`: row.start._id'})
MATCH (end:`UNIQUE IMPORT LABEL`{'`UNIQUE IMPORT ID`: row.end._id'})
CREATE (start)-[r:DESTINED_TO]->(end) SET r += row.properties;
UNWIND [{start: {_id:7}, end: {_id:9}, properties:{job_position:"driver"}}, {start: {_id:8}, end: {_id:9},
properties:{job_position:"storekeeper"}}] AS row
MATCH (start:`UNIQUE IMPORT LABEL`{'`UNIQUE IMPORT ID`: row.start._id'})
MATCH (end:`UNIQUE IMPORT LABEL`{'`UNIQUE IMPORT ID`: row.end._id'})
CREATE (start)-[r:WORKS_AT]->(end) SET r += row.properties;
UNWIND [{start: {_id:12}, end: {_id:5}, properties:{}}, {start: {_id:14}, end: {_id:5}, properties:{}}, {start:
{_id:13}, end: {_id:5}, properties:{}}] AS row
MATCH (start:`UNIQUE IMPORT LABEL`{'`UNIQUE IMPORT ID`: row.start._id'})
MATCH (end:`UNIQUE IMPORT LABEL`{'`UNIQUE IMPORT ID`: row.end._id'})
CREATE (start)-[r:ORDERED_BY]->(end) SET r += row.properties;
:commit
:begin
MATCH (n:`UNIQUE IMPORT LABEL`) WITH n LIMIT 20000 REMOVE n:`UNIQUE IMPORT LABEL`
REMOVE n.`UNIQUE IMPORT ID`;
:commit
:begin
DROP CONSTRAINT ON (node:`UNIQUE IMPORT LABEL`) ASSERT (node.`UNIQUE IMPORT ID`) IS
UNIQUE;
:commit

```