Graph database project - “E-bilet”

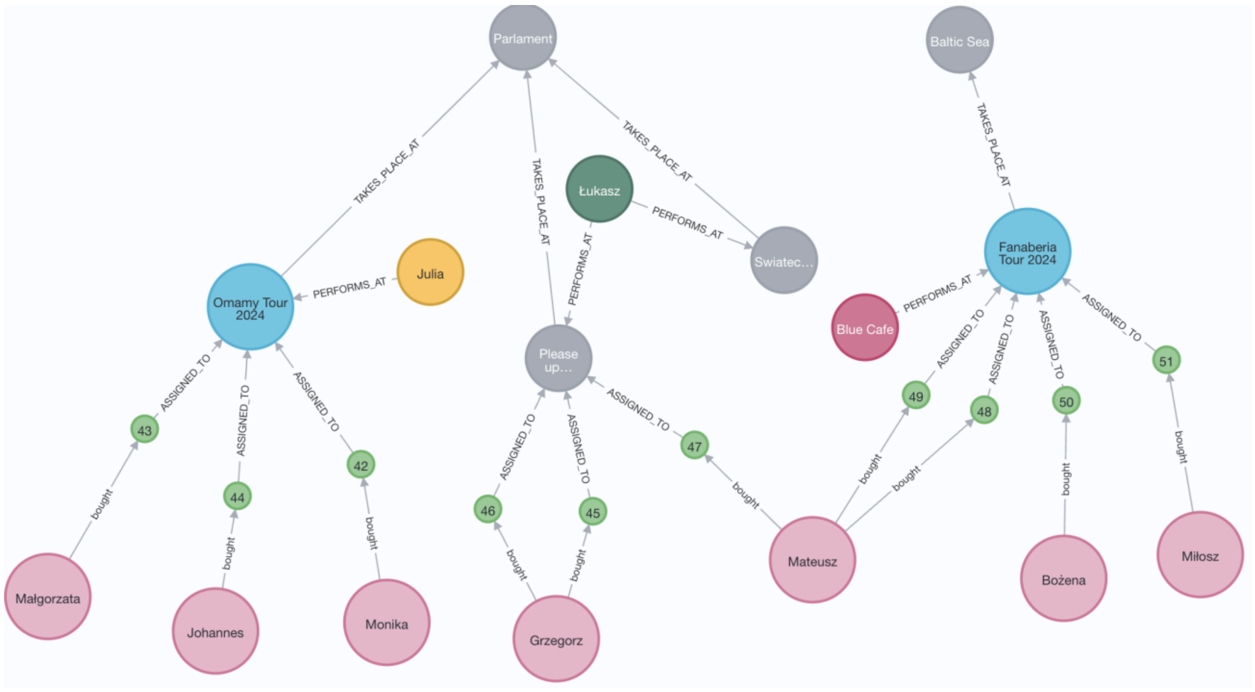
1. Business Domain

The e-bilet platform is a ticketing service designed to sell tickets for a wide range of events such as concerts, stand-up comedy shows, recitals, performances and many more. The platform allows artists to add events, specify the venue, and keep track of what tickets have been sold. The events have assigned specific tickets that are sold to users that are logged into service. The platform displays all upcoming events hosted, allowing users to browse and purchase tickets directly through our website.

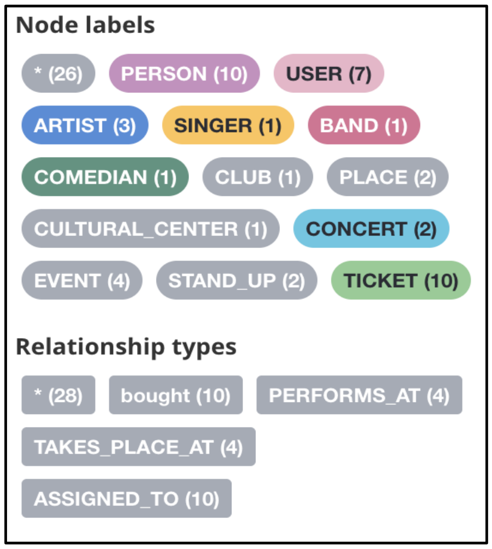
To buy a ticket, users must first create a profile by providing personal information such as their name, surname, email, and phone number (with birthdate being optional). Each ticket has a set price and may include a seat number for reserved seating events or special classifications like VIP tickets. For general admission events, no seat number is needed.

The venues we work with provide essential details about their locations (such as the address and manager's contact information) and information about the events. This includes details on the performing artists (headliner), and specifics of the event itself like capacity, date, start time, duration, and event name.

2. Information about Graph



Picture 2.1. Screenshot of a graph from Neo4j Desktop



Node labels

* (26) PERSON (10) USER (7)
ARTIST (3) SINGER (1) BAND (1)
COMEDIAN (1) CLUB (1) PLACE (2)
CULTURAL_CENTER (1) CONCERT (2)
EVENT (4) STAND_UP (2) TICKET (10)

Relationship types

* (28) bought (10) PERFORMS_AT (4)
TAKES_PLACE_AT (4)
ASSIGNED_TO (10)

The graph database consists of **26 nodes and 28 relationships**

There are **13 different Node labels**, and **4 different types of relationships**

:SINGER/:BAND/:COMEDIAN – extension labels to an :ARTIST
:CULTURAL_CENTER/:CLUB – extension labels to a :PLACE

:BOUGHT – relationship between :USER and :TICKET
:PERFORMS_AT – relationship between :ARTIST and :EVENT
:TAKES_PLACE_AT – relationship between :EVENT and :PLACE
:ASSIGNED_TO – relationship between :TICKET and :EVENT

Picture 2.2. Screenshot of a graph overview

2.1. Properties of Nodes and Relationships

The nodes of the label may have slightly different attributes, and different number of attributes, that's why we will just enumerate each attribute that appears on the graph:

:USER – name, surname, join_date, date_birth, phone_number, email, nationality

:TICKET – id, price, seat, row, type
:EVENT – date, duration, start_time, title
:PLACE – name, city, address, contact
:ARTIST – name, surname, nationality, genre

[:BOUGHT] - how, when
[:ASSIGNED_TO] - none
[:PERFORMS_AT] - role, setlist
[:TAKES_PLACE_AT] - capacity, rental_price

3. Database Creation

All of the commands that we used to create this database are saved in cypher file (create_database.cypher). Below, we attached just a snippet for each type of nodes.

```
CREATE
(:PERSON:USER {name: "Monika",surname: "Kowalska",ID:10203041, birth_date:date("2001-05-01"), email:"monika123@gmail.com",
phone:"123456789", Nationality:"Polish", join_date:date("2022-10-01")}),
(:PERSON:USER {name: "Małgorzata",surname: "Piec",ID:10203042,birth_date:date("1977-04-21"), email:"malpiec@gmail.com",
phone:"123456788", Nationality:"Polish", join_date:date("2022-10-02")}),
```

```
//02 Artists
CREATE
(:PERSON:ARTIST:SINGER {name: "Julia",surname: "Wieniawa", Nationality:"Polish", genre:"Pop"}),
(:PERSON:ARTIST:BAND {name: "Blue Cafe",  Nationality:"Polish", band_size: 7, genre: "Jazz"}),
```

```
//03 Places
create
(:PLACE:CLUB {name: "Parlament", city:"Gdańsk", address:"Świętego Ducha 2", contact:"manager_parlament@gmail.com"}),
(:PLACE:CULTURAL_CENTER {name:"Baltic Sea", city:"Gdańsk",address:"Korzenna 33/35", contact:"manager_BalticSea@wp.pl"})
```

```
//04 Events
CREATE
(:EVENT:CONCERT {title: "Omamy Tour 2024", date:date("2024-08-10"), start_time:time("19:30"), duration: 120}),
(:EVENT:STAND_UP {title: "Please stand up", date:date("2024-09-16"), start_time:time("18:25"), duration:60}),
```

```
//05 Tickets
CREATE
(:TICKET {ID: "000", price:179}),
(:TICKET {ID: "001", price:219}),
(:TICKET {ID: "002", type:"VIP", seat:3,  price:449}),
(:TICKET {ID: "003", seat:"10", row:"H", price:199}),
```

And here are the snippet for relationships:

```
match (event {title:"Fanaberia Tour 2024"}), (artist:ARTIST {name:"Blue Cafe"}), (place {name:"Baltic Sea"})
create (artist)-[:PERFORMS_AT {role: 'headliner'}]→(event)
create (event)-[:TAKES_PLACE_AT {rental_price: 150, capacity: 45}]→(place);
match (ticket), (event {title:"Fanaberia Tour 2024"})
where ticket.ID IN ["006","007","008","009"]
create (ticket)-[:ASSIGNED_TO]→(event);
```

```
match (ticket{ID:"000"}), (person{ID:10203041})
create (person)-[:bought {when:datetime("2024-07-09T11:13:45"), how:"paypal"}]→(ticket)
match (ticket{ID:"001"}), (person{ID:10203042})
create (person)-[:bought {when:datetime("2024-08-01T14:30:15"), how:"blik"}]→(ticket);
match (ticket{ID:"002"}), (person{ID:10203043})
```

4. Scenario

Whole graph represents the e-bilet platform. This organisation collaborates with 2 venues: Parlament (club), and Baltic Sea (cultural center). To collaborate, an organisation needs to provide address of the venue and contact its manager/representative. There are 2 events that already happened at Parlament, and another one is planned to be held at the end of this year. Stand-up comedian Lotek hosted the event "Please stand up" at Parlament, with 3 tickets sold to 3 different people. Julia Wieniawa headlined the "Omamy Tour 2024", also at Parlament, where 3 tickets were sold to 2 users. The Baltic Sea has hosted only one event so far. At the event Blue Cafe performed for the "Fanaberia Tour", with 4 tickets sold to 3 different users. When the venue hosts an event, it determines the rental price and maximum capacity of the audience.

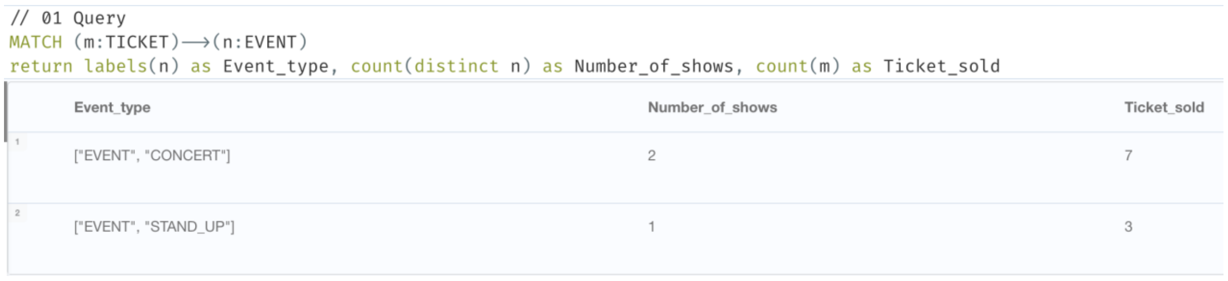
The events are various, it can be a concert, standup, or something else. The event is often assigned to one artist that is the headliner (however, there can be a support artist). On the platform, if a user wants to purchase a ticket, he needs to first provide some basic details about himself: Name, Surname, Contact (Email/Phone number), etc. Once he has done that, he could buy one or more tickets for the show that he wants to see, because ticket is not assigned to person. System saves the payment method and the exact date and time of the purchase.

Each ticket has assigned a price to it, a unique identifier, and additionally a row and a seat number (if the event has assigned seats to a ticket)

5. Competency Questions

We come up with 8 queries that we think are both complicated and realistic (if the organisation would function in real-life). The cypher file with all queries is attached to the project (queries.cypher). Below, we attached the screenshots of the code and results for each question.

1. How many tickets have been sold for each event type and how many number of shows were performed? (aggregation + distinct + labels())

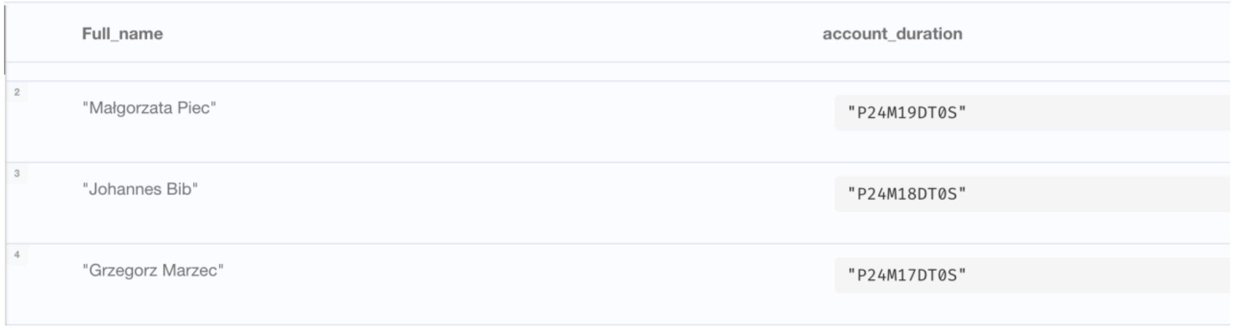


```
// 01 Query
MATCH (m:TICKET)⟶(n:EVENT)
return labels(n) as Event_type, count(distinct n) as Number_of_shows, count(m) as Ticket_sold
```

	Event_type	Number_of_shows	Ticket_sold
1	["EVENT", "CONCERT"]	2	7
2	["EVENT", "STAND_UP"]	1	3

2. How long has each user had their account? (duration + concat using +)

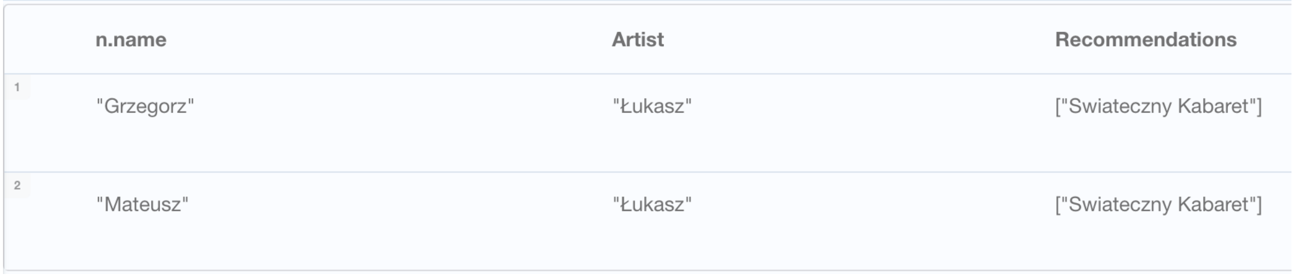
```
// 02 Query
match(n:USER)
return n.name+" "+n.surname as Full_name, duration.between(n.join_date, date()) as account_duration
```



	Full_name	account_duration
2	"Małgorzata Piec"	"P24M19DT0S"
3	"Johannes Bib"	"P24M18DT0S"
4	"Grzegorz Marzec"	"P24M17DT0S"

3. The event recommendations for users : it shows the upcoming shows of the artists that user has seen before based on ticket purchases? (traversing through graph, date calculations)

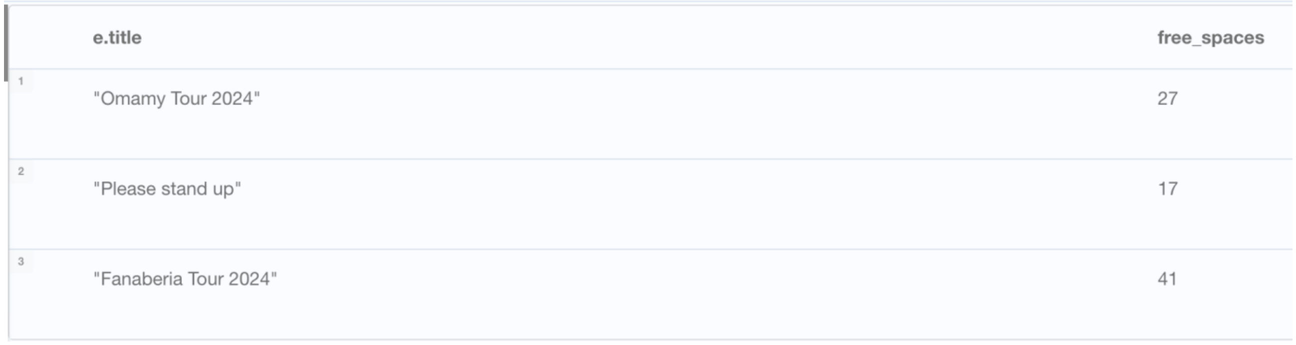
```
//03 Query
MATCH (n:USER)-->(t:TICKET)-->(e:EVENT)<--(m:ARTIST)-->(e2:EVENT)
where e2.date > date()
RETURN  n.name, m.name as Artist,collect(distinct e2.title) as Recommendations
```



	n.name	Artist	Recommendations
1	"Grzegorz"	"Łukasz"	["Swiateczny Kabaret"]
2	"Mateusz"	"Łukasz"	["Swiateczny Kabaret"]

4. How many free tickets are left for events? (aggregations)

```
// 04 Query
match (t:TICKET)-->(e:EVENT)-[r:TAKES_PLACE_AT]->()
return e.title,  max(r.capacity) - count(t)  as free_spaces
```



	e.title	free_spaces
1	"Omamy Tour 2024"	27
2	"Please stand up"	17
3	"Fanaberia Tour 2024"	41

5. How much money did artists made for each show? (aggregations)

```
// 05 Query
//sum of money earned on tickets
match (t:TICKET)-->(e:EVENT)-[r:TAKES_PLACE_AT]->()
return e.title,  sum(t.price)-max(r.rental_price)  as sum_earned;
```

	e.title	sum_earned
1	"Omamy Tour 2024"	547
2	"Please stand up"	297
3	"Fanaberia Tour 2024"	276

6. Who bought a ticket with a seat no. 5 in a row 'J' for "Please Stand up"?

```
// 06 Query
//is seat 5 in row J on Stand up  performance "Omamy Tour 2024" taken, used so nobody will buy the same seat twice
match (n:USER)-->(t:TICKET{seat:"5", row:"J"})-->(e:EVENT{title:"Please stand up"})
return  n.name, n.surname
```

	n.name	n.surname
1	"Grzegorz"	"Marzec"

7. Which place has the most experience in holding an event (the more events held the more experience)? (order by + limit + aggregations)

```
// 07 Query
//the place has the most experiance in holding en event(the more events held the more experience), for artists to help
to choose where to held your event
match (e:EVENT)-[TAKES_PLACE_AT]->(p:PLACE)
return  p.name as PLACE_NAME, count(e) as NUMBER_OF_EVENTS
order by  NUMBER_OF_EVENTS desc
limit 1
```

	PLACE_NAME	NUMBER_OF_EVENTS
1	"Parlament"	3

8. Who have bought more than one ticket? (where statement + aggregations)

```
// 08 Query
//who bought more than 1 ticket?
match (u: USER)-[:bought]->(t: TICKET)
with u.ID AS user_identificator, count(t) AS tickets_amount
where tickets_amount > 1
return user_identificator
```

	user_identificator
1	10203044
2	10203045

6. Project Summary

This project provided valuable experience using the tool Neo4j and helped us familiarize with CYPHER language by going through Cypher Manual and discovering new functions. We think it was the perfect and user-friendly project as an introduction to NoSQL.