# Тестовое задание по графовым базам данных

## 1. Очищаем все переменные

In [43]:
```
%reset -f
```

## 2. Импортируем библиотеки: pandas (работа с данными), neo4j (работа с граф БД), json (rest), random (выбор случайного ФИО)

In [44]:
```
import pandas as pd
from neo4j import GraphDatabase
import json
import random
```

## 3. Считываем данные из файла .csv

In [45]:
```
data = pd.read_csv("../files/data_test.csv",
                   names=['id_event', 'fullname1', 'fullname2'],
                   header=0, sep=";")
data.head(10)
```

Out[45]:

| | id_event | fullname1 | fullname2 |
|---|---|---|---|
| **0** | 189 | Галчевская Карина Владимировна | Белоновская Анастасия Семеновна |
| **1** | 206 | Офицеров Олег Романович | Сапожник Борис Валерьевич |
| **2** | 445 | Жандарова Лариса Германовна | Чемодуров Дамир Русланович |
| **3** | 503 | Масимова Яна Дамировна | Мингажетдинов Рамиль Семенович |
| **4** | 571 | Мухтарова Алена Яковлевна | Щербатенко Ольга Робертовна |
| **5** | 595 | Русских Кира Константиновна | Федутинов Артем Артурович |
| **6** | 637 | Федонкина Кристина Маратовна | Маргиев Григорий Иванович |
| **7** | 741 | Чернолусская Екатерина Марселевна | Бухановская Лидия Радиковна |
| **8** | 996 | Оранский Владимир Артурович | Скотников Виктор Ильич |
| **9** | 1210 | Ардатов Андрей Анатольевич | Ханыгин Дмитрий Юрьевич |

## 4. Формируем словарь со свойствами для каждого node и запрос для внесения данных в БД
## props - словарь со свойствами
## query - запрос для внесения данных (создаем Participants1, Participants2, Event)

In [46]:
```
def get_multiple_nodes_properties(df: object) -> dict:
    records = {
    "props": []
    }

    for row in data.itertuples(index=False):
        records["props"].append({'id_event':  row[0],
                                 'fullname1': row[1],
                                 'fullname2': row[2]})
    return records

props = get_multiple_nodes_properties(data)
query = """
```

```
UNWIND $props as user
MERGE(p1:  Participants1 {name: user.fullname1, id_event: user.id_event })
MERGE(p2:  Participants2 {name: user.fullname2, id_event: user.id_event })
MERGE(e: Event {name: user.id_event})
"""
```

### 5. Проверяем количество значений словаря `props`

In [47]:
```
len(props.get('props'))
```

Out[47]: 5000

### 6. Подключаемся к БД и заносим данные на основе `query` и `props`

In [48]:
```
class LocalDatabaseNeo4j:
    def __init__(self, uri, db, user, password):
        self.driver = GraphDatabase.driver(uri, database=db, auth=(user, password))

    def execute_queries(self, query, props=None):
        with self.driver.session() as session:
            session.run(query, props)

    def close(self):
        self.driver.close()


graphs = LocalDatabaseNeo4j("bolt://localhost:7687", "graphdata", "mark", "123")
graphs.execute_queries(query, props)
graphs.close()
```

### 7. Rest сервис

In [49]:
```
all_participants = data.fullname1.to_list()

name = ''.join(random.choices(all_participants))
query_rest = """
MATCH (properties: Participants1 | Participants2)
WHERE properties.name = $name
RETURN properties
"""
print("Поиск:", name)
```

```
Поиск: Белехов Владимир Никитович
```

In [50]:
```
class LocalDatabaseNeo4j:
    def __init__(self, uri, db, user, password):
        self.driver = GraphDatabase.driver(uri, database=db, auth=(user, password))

    def execute_queries(self, query, name=None):
        with self.driver.session() as session:
            return session.run(query, name=name).data()

    def close(self):
        self.driver.close()


graphs = LocalDatabaseNeo4j("bolt://localhost:7687", "graphdata", "mark", "123")
rest = json.dumps(graphs.execute_queries(query_rest, name),  ensure_ascii=False,
                                                             indent=2)
graphs.close()
```

In [51]:
```
print(rest)
```

```
[
  {
    "properties": {
      "name": "Белехов Владимир Никитович",
      "id_event": 413843
    }
  }
]
```

In [ ]: