



# University database

---

dokumentacja

Przedmiot: Bazy danych  
Grupa: Czwartek 12:50 A  
Zespół: Natalia Brzozowska  
Karol Hamielec  
Zuzanna Brzezińska

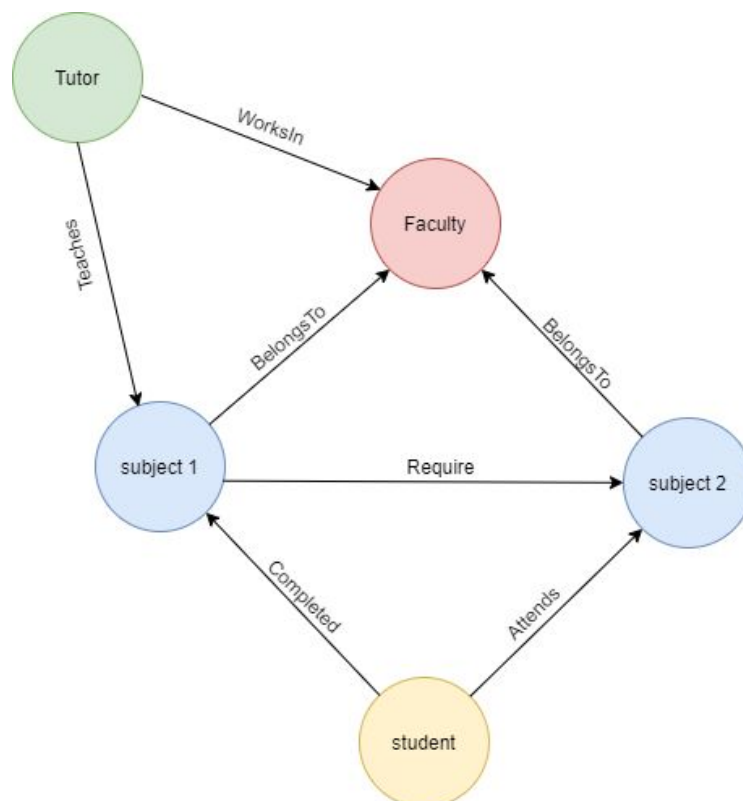
## Cel projektu:

Celem projektu było stworzenie bazy danych wspomagającej działanie uniwersytetu. Tworzony jest on przez jednostki - wydziały- w ramach każdego wydziału zatrudnionych jest pewna liczba prowadzących, oraz organizowane jest wiele kursów. Kursy mają strukturę hierarchiczną - student dany kurs może rozpocząć wtedy i tylko wtedy, gdy ukończył wszystkie wymagane kursy z niższych poziomów. Istnieje 7 poziomów kursów, kursy z poziomu 1 nie wymagają ukończenia żadnych innych kursów, kursy z 2 poziomu wymagają ukończenia pewnych kursów z poziomu 3 itd. Jeden prowadzący należy tylko do jednego wydziału, za to może prowadzić wiele kursów i analogicznie - kurs może mieć wielu prowadzących. Niektóre kursy są organizowane przez więcej niż jeden wydział.

## Technologie użyte w projekcie:

Do stworzenia bazy danych użyto systemu zarządzania bazą danych grafów Neo4J oraz języka Python

## Schemat bazy:



## Opis Węzłów:

- a. student:
  - i. firstname - imię
  - ii. lastname - nazwisko
  - iii. pesel
  - iv. student\_nr - numer albumu
- b. Tutor:
  - i. degree - stopień naukowy
  - ii. firstname - imię
  - iii. lastname - nazwisko
  - iv. mail
- c. Subject
  - i. free\_places - liczba wolnych miejsc
  - ii. name - nazwa kursu
  - iii. tier - poziom kursu
- d. faculty
  - i. name - nazwa wydziału

## Opis relacji:

- a. Attends:
  - student -> subject
  - student uczęszcza na kurs
- b. Completed:
  - student -> subject
  - student ukończył kurs
- c. Require:
  - subject -> subject
  - rozpoczęcie kursu wymaga zaliczenia innego kursu
- d. WorksIn:
  - tutor -> faculty
  - prowadzący jest pracownikiem wydziału
- e. Teaches:
  - tutor -> subject
  - prowadzący uczy danego kursu
- f. BelongsTo:
  - subject -> faculty
  - kurs jest prowadzony przez dany wydział

## Opis plików i funkcji:

Funkcjonalności bazy danych - plik db\_helpers.py:

1. Funkcja zwracająca:

- a. Wszystkie kursy prowadzone przez danego pracownika uniwersytetu  
`tutors_courses(tx, firstname, lastname)`
- b. Wydział który zatrudnia danego pracownika:  
`tutors_department(tx, firstname, lastname)`
- c. Pracownicy prowadzący więcej niż jeden przedmiot:  
`tutors_who_teaches_many_subjects(tx, number)`
- d. Kursy, których ukończenie jest potrzebne do rozpoczęcia wybranego kursu:  
`required_subjects(tx, sub=None)`
- e. Kursy, które dany student musi jeszcze ukończyć żeby zapisać się na dany kurs:  
`missing_required_subjects(tx, sub=None, album_nr = None)`
- f. Wszystkie kursy prowadzone przez dany wydział:  
`faculty_subjects(tx, faculty_name)`
- g. Liczbę studentów uczestniczących w kursie:  
`students_in_subject(tx, course_name)`
- h. Kursy, dostępne w danym momencie dla danego studenta:  
`courses_available_for_student(tx, album_nr)`
- i. Informacje dot. danego studenta:  
`get_student_info(tx, firstname=None, lastname=None, pesel=None, student_nr=None)`
- j. Informacje dot. danego prowadzącego:  
`get_tutor_info(tx, firstname=None, lastname=None, degree=None, mail=None)`
- k. Najbardziej optymalną "ścieżkę" kursów, które trzeba zdać by móc rozpocząć dany kurs:  
`shortest_subject_path(tx, album_nr, subject_name)`
- l. Kursy, prowadzone przez więcej niż jeden wydział:  
`subjects_belong_to_few_departments(tx)`
- m. Kursy, które ukończył dany student:  
`get_student_completed_courses(tx, firstname=None, lastname=None, pesel=None, student_nr=None)`

- n. Kursy, na które uczęszcza dany student:  
`get_student_attends_courses(tx, firstname=None, lastname=None, pesel=None, student_nr=None):`

2. Metoda:

- a. Dodająca nowego studenta  
`add_student(tx, firstname, lastname, pesel, album_nr)`
- b. Dodająca nowego prowadzącego  
`add_tutor(tx, degree, firstname, lastname, mail, faculty)`
- c. Umożliwiająca zapisanie się na kurs  
`sign_up(tx, course_name, student_nr)`
- d. Umożliwiająca odnotowanie ukończenia kursu  
`complete_course(tx, course_name, student_nr)`

Wstawianie danych z plików csv do bazy - plik data\_insert.py:

- a. Wstawianie kursów:  
`create_subjects(tx, filename, faculty_name)`
- b. Wstawianie prowadzących:  
`create_tutors(tx, lecturers_file, faculty_name):`
- c. Wstawianie studentów:  
`create_students(tx, filename)`
- d. Odnotowywanie ukończenia kursów przez wstawionych studentów:  
`sign_students(tx, filename)`
- e. Odnotowywanie zapisania na kursy wstawionych studentów:  
`set_attends_rel(tx, filename)`

Pobieranie prowadzących ze strony agh i wstawianie ich do pliku csv - plik tutors\_scrapper.py

Pobieranie listy przedmiotów ze strony agh i wstawianie ich do pliku csv - plik przedmioty.py

Wywołanie funkcji z db\_helpers.py i wyświetlenie wyników - plik cli.py

## Uruchomienie programu:

### 1. Przygotowanie:

Należy pobrać plik requirements.txt i zainstalować znajdujące się w nim biblioteki

### 2. Utworzenie bazy :

Aby wypełnić bazę danymi należy pobrać zawartość folderu Data oraz plik data\_inserter.py z folderu src. W pliku w linijce :

```
driver1 = GraphDatabase.driver("bolt://bazy.flemingo.ovh:7687", auth=("neo4j",  
    "marcjansiwikania"))
```

wpisać dane swojej bazy i uruchomić funkcję main.

### 3. Obsługa bazy:

Do obsługi bazy potrzebne będzie pobranie zawartości folderu src oraz pliku cli.py. po uruchomieniu funkcji main w pliku cli.py pojawi się linia komend.

Wpisanie

-h wypisuje wszystkie dostępne komendy

nazwa komendy -h wypisuje opis komendy wraz z atrybutami potrzebnymi do jej wywołania