Dokumentacja

Program wspomagający Dział Obsługi Doktorantów na uczelni

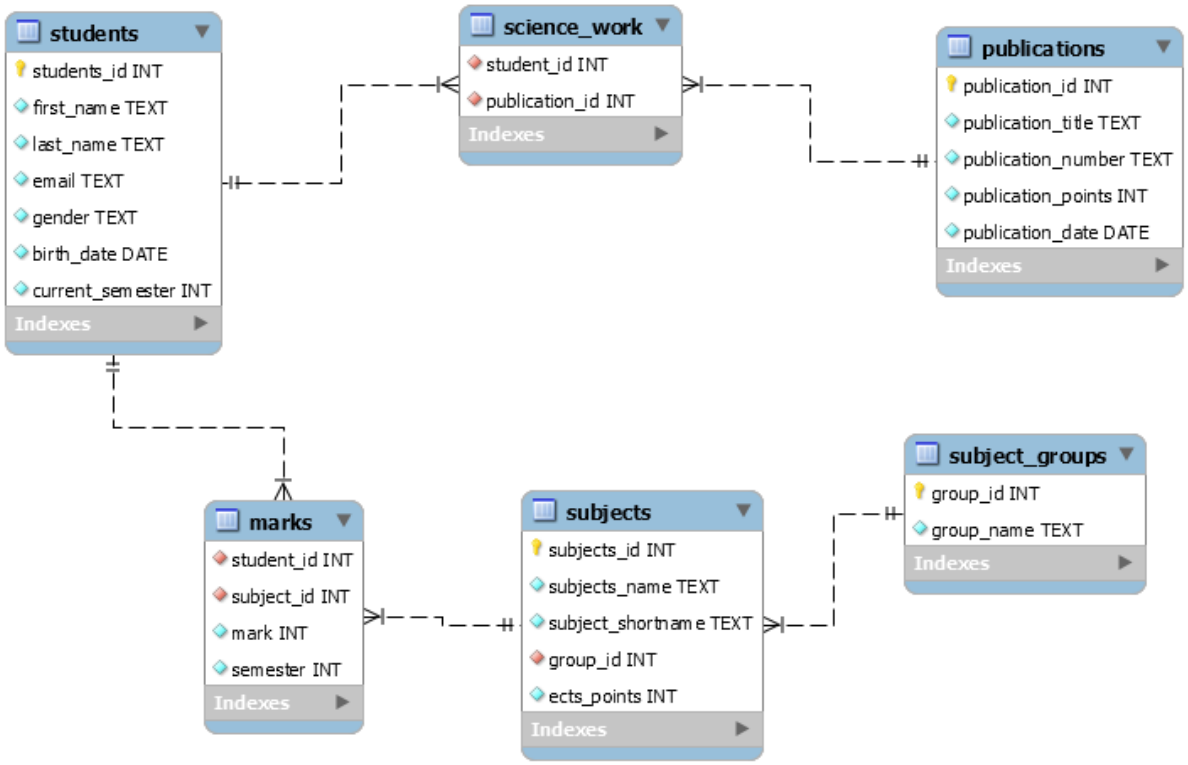
Pavel Klimuk

Opis projektu

Projekt polega na stworzeniu bazy danych, i programu obsługującego tę bazę.

Opis implementacji

Struktura bazy danych została przedstawiona poniżej.



Program składa się z dwóch plików głównych: program.py i main.py, pliku testowego: test_project.py, bazy danych: project_database_1.db i dokumentacji: doc.pdf.

Dla współpracy z bazą danych przez Python została użyta biblioteka: sqlalchemy, dla uruchomienia programu z konsoli : argparse.

Sqlalchemy została wybrana dlatego, że działa w oparciu na ORM(Object Relational Mapping) i zachowuje całą „elastyczność” SQL’a, co otwiera duże możliwości do działania.

W pliku program.py znajduje się ciało programu. Klasy – reprezentują główne tabele w bazie danych, takie jak Students, Subjects, Publications i Subject_groups. Tabele science_work i marks zostały zaimplementowane jako tabele asocjacyjne.

Ważnym elementem programu jest *Engine,* który jest punktem startowym każdego programu, korzystającego z sqlalchemy. Jest to „baza domowa” dla aktualnej bazy danych.

Kolejnym bardzo ważnym elementem jest *Session*. Faktycznie to ona nawiązuje połączenie z bazą danych i jest „strefą przechowywania” dla wszystkich obiektów, które zostały załadowane do niej w trakcie jej „życia”.

Zapytania – *Query* wykorzystując połączenie utworzone przez *Session* trafiają do bazy, a wyniki później są przechowywane w strukturze nazywanej *Identity Map*.

Obiekty *Query* filtrowane są za pomocą metody *filter.*

Na przykładzie metody *points_for_publication* z klasy *Publications* pokażę jaki jest algorytm komunikacji z bazą.

```
 def points_for_publication(publication_id):
        Session = sessionmaker(bind=engine)
        session = Session()
        default_points = session.query(Publications.publication_points).\
            filter(Publications.publication_id == publication_id).scalar()
        number_of_authors = Publications.number_of_publication_authors(publica
tion_id)
        session.commit()
        points = round(default_points / number_of_authors, 2)
        return points
```

1. Utworzenie połączenia z bazą (session)
2. Wysłanie zapytania (session.query)
3. Filtrowanie wyniku (filter)
4. Zakończenie połączenia (session.commit())

Podobny algorytm wykorzystuje się również dla usuwania / dodawania / modyfikacji elementów.

Podsumowanie wykonanych prac

Program spełnia wszystkie wymagania funkcjonalne.

Pierwszym problemem przy wykonywaniu projektu było stworzenie bazy danych, próbowałem zrobić to różnymi sposobami i w wyniku pomógł mi w tym program *DB Browser for SQLite* w którym i stworzyłem bazę. Dane do bazy w większości zostały wegenerowane za pomocą strony internetowej

mockaroo.com. Kolejnym krokiem był wybór biblioteki, za pomocą której można komunikować się z bazą. Wybrałem SQLAlchemy, bo przeczytałem, że jest to bardzo pomocnicze narzędzie dla współpracy z bazą w Python'ie. Miałem małe trudności przy opisywaniu różnego rodzaju zależności, a mianowicie sytuacji, gdy z bazy usuwa się student i wraz z nim muszą zniknąć oceny, przywiązania do publikacji i t.p., ale po krótkim czasie dało się to osiągnąć. Po ogarnięciu zapytań(*query*) wykonywanie projektu było już samą przyjemnością =) . Po napisaniu „ciała" projektu zająłem się interfejsem, za pomocą biblioteki argparse, co nie sprawiło większych problemów. Dzięki pracy nad projektem zapoznałem się z podstawami baz danych, co zmieniło mój stosunek do nich. Wcześniej myślałem, że są nudne, ale po małym zagłębieniu w ten temat zrozumiałem, że jest inaczej. Przy pracy z Python'em w większości utrwalałem swoją wiedzę, zdobytą na warsztatach i poznaną z innych źródeł. Przedostatnim problemem było napisanie testów jednostkowych, co spowodowało zmiany w moim kodzie, moim zdaniem na lepsze. Ostatnim problemem okazały się wyjątki(*Exceptions*). Opisałem ich w pliku *program.py*, ale, niestety, nie zdążyłem zrobić tego w pliku *main.py*. Oprócz wyjątków, jestem bardzo zadowolony z tego, co udało mi się osiągnąć, szczególnie biorąc pod uwagę to, że w momencie kiedy w pierwszy raz przeczytałem zadanie nie wiedziałem nawet od czego zacząć. Na koniec chciałbym podziękować za wiedzę przekazaną na warsztatach i za pomoc na konsultacjach. Dziękuję!

Instrukcja obsługi programu

Przed uruchomieniem projektu należy stworzyć i uruchomić wirtualne środowisko(https://docs.python.org/3/library/venv.html). Zainstalować bibliotekę SQLAlchemy(w wierszu poleceń wpisać **pip install sqlalchemy**). Upewnić się, że zainstalowane są biblioteki *argparse, datetime,* zrobić to można wpisując do wiersza poleceń **pip freeze**.

Aby uruchomić program należy po aktywacji środowiska za pomocą wiersza poleceń przejść do katalogu, w którym znajduje się program(**cd** …). Wpisać **main.py -cm** wybrać jedno z poleceń poniżej i wpisać dodatkowe podpolecenia, określające np. id studenta, imię studenta i t.p. Przykłady wywoływania każdego polecenia zostaną podane poniżej.

Polecenia:

create_student	Dodaje doktoranta do bazy
delete_student	Usuwa doktoranta z bazy
modify_student	Modyfikuje informację o doktoranciecie
studying_history	Wyświetla historię studiów doktoranta
publications_history	Wyświetla historię publikacji doktoranta
points_for_publications	Wyświetla punkty za publikacje
requirements_verification	Weryfikuje postep doktoranta wzgledem wymagań na dany etap studiow
ranking_list	Wyświetla listę doktorantów dostających stependium
avarage_mark	Wyświetla średnią skumulowaną doktoranta

Podpolecenia:

-cm	Przyjmuje polecenie(Wymagane)
-si	--student_id
-sf-	--student_first_name
-sl	--student_last_name
-se	--student_email
-sg	--student_gender
-sb	--student_birth_date
-sc	-student_current_semester
-d1	--date_1

-d2	--date_2
-l	--limit
-subg	--subject_group
-ep	--ects_points

Dodanie doktoranta:

Aby dodać doktoranta należy podać polecenie *create_student* i wskazać:

Imię doktoranta, nazwisko, email, płeć(Male/Female), datę urodzenia(m/d/r), aktualny semestr studiów

main.py -cm create_student -sf Jj -sl Smith -se j.@m.com -sg Male -sb 03/03/1999 -sc 3

Usuwanie doktoranta:

Aby usunąć doktoranta należy podać polecenie *delete_student* i wskazać:

Id doktoranta

main.py -cm delete_student -si 50

Modyfikacja doktoranta:

Aby zmodyfikować doktoranta należy podać polecenie *modify_student* i wskazać:

Id doktoranta, imię doktoranta, nazwisko, email, płeć(Male/Female), datę urodzenia(m/d/r), aktualny semestr studiów

main.py -cm modify_student -si 52 -sf Aj -sl Smith -se j.@m.com -sg Male -sb 03/03/1999 -sc 3

Historia studiów:

Aby wyświetlić historię studiów doktoranta należy podać polecenie *studying_history* i wskazać:

Id doktoranta

main.py -cm studying_history -si 15

Historia publikacji naukowych:

Aby wyświetlić historię publikacji doktoranta należy podać polecenie *publications_history* i wskazać:

Id doktoranta

main.py -cm publications_history -si 15

Punkty uzyskane za publikacje:

Aby wyświetlić punkty za publikacje doktoranta w jakimś okresie należy podać polecenie *points_for_publications* i wskazać:

Id doktoranta, datę rozpoczynającą okres(m/d/r), datę kończącą okres(m/d/r)

main.py -cm points_for_publications -si 5 -d1 10/10/2012 -d2 10/10/2020

Wyryfikacja postępu:

Aby zweryfikować, czy doktorant spełnia wymagania należy podać polecenie *requirements_verification* i wskazać:

Id doktoranta, grupę przedmiotową(Matematyka, Fizyka, Informatyka, Elektronika), punkty ects(wartość domyślna to 11)

main.py -cm requirements_verification -si 10 -subg Elektronika -ep 11

Lista stypendialna:

Aby wyświetlić listę stypendialną należy podać polecenie ranking_list i wskazać(opcjonalnie):

Limit(wartość domyślna to 20 %)

main.py -cm ranking_list -l 30

Średnia skumulowana:

Aby wyświetlić średnią skumulowaną doktoranta należy podać polecenie *avarage_mark* i wskazać:

Id doktoranta

main.py -cm avarage_mark -si 5