INFORMATYKA II

ZAPYTANIE SELECT W JĘZYKU SQL

CEL LABORATORIUM

Celem zajęć jest zapoznanie się z zapytaniem SELECT języka SQL. Podczas zajęć zaprezentowana zostanie składnia i przykłady, pozwalające konstruować zapytania języka SQL służące do pobierania informacji z bazy danych. Omówione zostaną sposoby wyboru określonych kolumn, sortowania, czy też wykorzystywania wbudowanych w język SQL funkcji.

CZYM JEST I JAK WYGLĄDA BAZA DANYCH

Czym jest baza danych? Jest to zbiór danych[1] spełniający pewne kryteria. Najczęściej wymienianymi kryteriami jest uporządkowanie i spójność (rozumiana jako jednolitość tematyczna). Z bazami danych możemy się spotkać nie tylko w komputerach. Przykładem rzeczywistej bazy danych może być biblioteka publiczna, czy książka telefoniczna.

Cyfrowe bazy danych to **pliki** o określonej strukturze, przyspieszającej wyszukiwanie i przetwarzanie informacji. Bazy danych, z którymi będziemy pracować podczas zajęć są podzielone na **tabele**. Każda tabela przechowuje fragment danych o takich samych cechach, lecz innych wartościach. Przykładem takiej tabeli może być tabela przechowująca klientów sklepu. Indywidualny klient jest opisany przez takie cechy jak imię, nazwisko i PESEL, lecz ich konkretna wartość jest odmienna i niepowtarzalna dla każdej osoby.

Cechy w tabeli będą oznaczać **kolumny** (w omawianym przypadku kolumn będzie trzy). **Wiersz** tabeli oznacza klienta. Tabela będzie więc zawierać tyle wierszy, ilu było klientów (a każdy z nich będzie opisany polem zawierającym imię, polem zawierającym nazwisko i polem zawierającym PESEL), którzy skorzystali z usług sklepu. Przykład takiej tabeli znajduje się poniżej:

Identyfikator	Imię	Nazwisko	PESEL
1	Adam	Mickiewicz	98122412345
2	Juliusz	Słowacki	09090412345
3	Jan	Brzechwa	98081512345
4	Zbigniew	Herbert	24102912345

Powyższa tabela zawiera jedną dodatkową kolumnę – identyfikator. Każda tabela bazy danych powinna zawierać przynajmniej jedną kolumnę zawierającą unikalne wartości. Kolumnę taką nazywamy **kluczem głównym** tabeli (ang. „*Primary Key*"). Zauważ, że kolumna PESEL również może zawierać wyłącznie unikalne wartości. Kolumnę, która spełnia wymagania

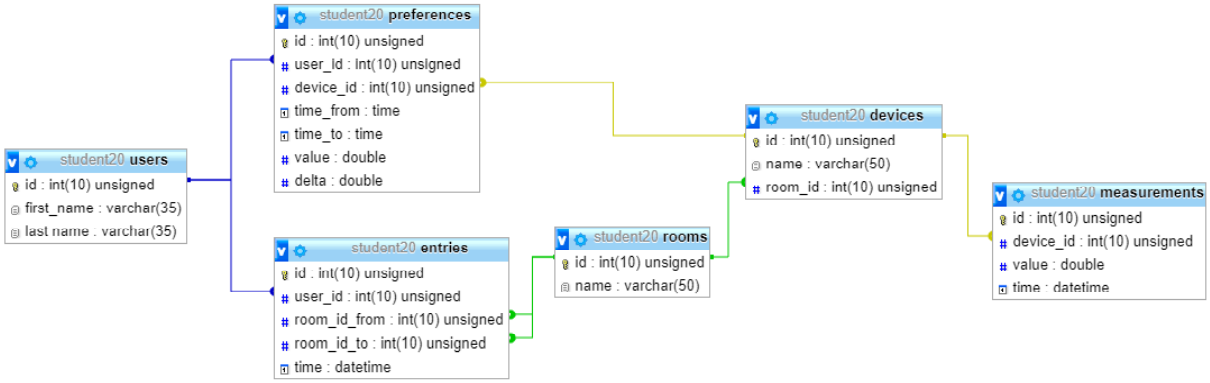
[1] Istnieje formalna definicja „danych", jednak nie będzie ona przedstawiona, gdyż intuicyjna interpretacja jest w pełni wystarczająca w omawianym kontekście. Istnieje również rozróżnienie „danych", od „informacji", ale w niniejszej instrukcji będą to wyrażenia używane zamiennie.

pozwalające zostać kluczem głównym, ale która nim nie jest, nazywamy **kluczem kandydującym** (lub potencjalnym). W skład klucza głównego może wchodzić więcej niż jedna kolumna (np. kluczem głównym może być para kolumn: numer domu i numer mieszkania). Klucz złożony z więcej niż jednej kolumny nazywamy… **kluczem złożonym**.

Wykorzystywana podczas laboratorium baza danych przechowuje informacje o hipotetycznym inteligentnym domu. Wśród zebranych danych znajdują się informacje o domownikach (w tym dane osobowe i preferencje dotyczące działania domu), pomieszczeniach (w tym o tym, kiedy dany mieszkaniec przeszedł z jednego pomieszczenia do drugiego) oraz urządzeniach (w tym, przede wszystkim, wykonane przez nie pomiary).

MATERIAŁY POMOCNICZE

Diagram bazy danych przedstawia się następująco:



Kod SQL generujący wykorzystywaną w trakcie zajęć bazę danych można pobrać pod adresem http://www.stawarz.edu.pl/informatyka2/smart_house.sql.

Baza danych może zostać uruchomiona na każdym komputerze, który jest zaopatrzony w system zarządzania bazami danych MySQL. Darmowym programem pozwalającym pracować z bazą danych w zaciszu domowym jest program XAMPP, który można pobrać pod adresem https://www.apachefriends.org/pl/index.html.

Program opracowany na poprzednich zajęciach (w wersji okrojonej), można pobrać pod adresem http://www.stawarz.edu.pl/informatyka2/Lab1.rar (lub w wersji ZIP)

MODUŁ INFORMACJI O MIESZKAŃCACH

Uruchom aplikację opracowaną na poprzednich zajęciach. Jeżeli jest niepełna, możesz skorzystać z okrojonej wersji, do której odnośnik znajduje się w poprzednim rozdziale niniejszej instrukcji.

Utwórz nowy plik. Nazwij go „modułObsługiMieszkańców.py". Znajdzie się w nim moduł zwracający informacje o mieszkańcach domu. Umożliwimy użytkownikowi wybranie jednej z poniższych opcji:

- wypisanie wszystkich danych o użytkownikach
- wypisanie listy identyfikatorów użytkowników
- wypisanie łącznej liczby użytkowników w bazie
- wypisanie imienia i nazwiska użytkownika na podstawie jego identyfikatoru

Sam kod będzie wyglądać podobnie do kodu klasy *Aplikacja* i kodu klasy *ModułBazyDanych*. Większość kodu jest łatwa do napisania i nie wymaga zastanowienia. Najbardziej problematyczne będzie opracowanie funkcji realizujących wymienione wcześniej opcje. Na chwilę obecną możemy wszystko przygotować i poprawne implementacje funkcji dostarczyć później.

Poniższy kod realizuje postawione przed nami zadanie. Przepisz go do pliku „modułObsługiMieszkańców":

```
from moduł import *
import mysql.connector

class ModułObsługiMieszkańców(Moduł):

    def __init__(self):
        self.identyfikator = "M"
        self.nazwa = "Moduł obsługi mieszkańców"

    def przejmijKontrolę(self):
        print("\nModuł obsługi mieszkańców:")
        print("========================")
        kursor = self.aplikacja.połączenie.cursor(dictionary=True)
        while (True):
            print("[1] Wypisz dane mieszkańców")
            print("[2] Wypisz identyfikatory mieszkańców")
            print("[3] Wypisz liczbę mieszkańców ")
            print("[4] Wyszukaj mieszkańca po identyfikatorze")
            print("[KONIEC] Wyjdź z modułu")

            opcja = input("Wybierz opcję z menu: ")
            if opcja.upper() == "KONIEC":
                break
            elif(opcja == "1"):
                self.wypiszMieszkańców(kursor)
            elif(opcja == "2"):
                self.wypiszIdentyfikatory(kursor)
            elif(opcja == "3"):
                self.wypiszLiczbęMieszkańców(kursor)
            elif(opcja == "4"):
                self.wyszukajMieszkańca(kursor)
            else:
                print("Nie ma takiej opcji w menu. Spróbuj ponownie!")
            print("\n")
        kursor.close()

    def wypiszMieszkańców (self, kursor):
        print("Ups! Tej opcji jeszcze nie zaimplementowano!")

    def wypiszIdentyfikatory(self, kursor):
        print("Ups! Tej opcji jeszcze nie zaimplementowano!")

    def wypiszLiczbęMieszkańców (self, kursor):
```

```
        print("Ups! Tej opcji jeszcze nie zaimplementowano!")

    def wyszukajMieszkańca(self, kursor):
        print("Ups! Tej opcji jeszcze nie zaimplementowano!")
```

Żeby móc korzystać z nowego modułu, musimy jeszcze go zarejestrować w naszej aplikacji. W tym celu, do pliku *main* należy dodać dwie linijki kodu:

```
from modułObsługiMieszkańców import *
```

oraz

```
aplikacja.zarejestrujModuł(ModułObsługiMieszkańców())
```

Zadbaj o to by te dwie linijki kodu znajdowały się w **odpowiednim** miejscu pliku *main*.

SKŁADNIA POLECENIA SELECT

Język Python, który do tej pory poznaliśmy, zaliczamy do rodziny języków imperatywnych[2] (od łac. *imperō* – władać, wydawać rozkazy, polecenia). Języki imperatywne charakteryzują się tym, że pisane w nich programy składają się z serii poleceń. Polecenia informują komputer **jak** ma coś wykonać.

Do pracy z bazami danych służy język SQL. Język SQL nie jest językiem imperatywnym, ale językiem deklaratywnym (od łac. *declaratio* – wyjaśniać). Język SQL nie mówi jak konkretnie wykonać daną rzecz (tym zajmie się sama baza danych), zamiast tego opisuje on co chcemy osiągnąć.

Najbardziej podstawowym poleceniem języka SQL jest polecenie *SELECT* (ang.: *wybierz*). Jego uproszczona składnia przedstawia się następująco:

```
SELECT kolumna1[, kolumna2[, kolumnaN] ]
FROM tabela
[ WHERE warunek ]
[ ORDER BY kolumna[ ASC | DESC ] ];
```

Fragmenty otoczone nawiasami kwadratowymi są **opcjonalne**. Słowa kluczowe wyróżniono kolorem niebieskim. Przykładowe proste zapytanie *SELECT* zostało przedstawione na poprzednich zajęciach:

```
SELECT * FROM users;
```

[2] Tak właściwie to Python jest językiem wieloparadygmatowym. Można użyć języka Python do programowania funkcjonalnego. W trakcie zajęć laboratoryjnych skupimy się jednak na wykorzystaniu języka Python do programowania imperatywnego.

Gwiazdka (*) zamiast listy kolumn oznacza, że interesują nas **wszystkie** kolumny. W przypadku tabeli *users*, działanie poniższego polecenia będzie identyczne:

```
SELECT id, first_name, `last name` FROM users;
```

Zwróć uwagę, że nazwa kolumny *last name* została otoczona **grawisami**[3]. Ogólnie rzecz biorąc, każda nazwa **kolumny** oraz **tabeli** może być otoczona grawisami, ale nie musi. **Wymagane** jest to wyłącznie w momencie, gdy nazwa zawiera w sobie białe znaki (tak jest i w naszym wypadku). Zapytanie to można, przeczytane na głos, brzmiałoby „*wybierz (SELECT) identyfikatory (id), imiona (first_name) i nazwiska (last name) z (FROM) tabeli przechowującej użytkowników (users)*". Zwróć uwagę jak prosto utworzyć takie zapytanie w naszym języku i jak łatwo je później przetłumaczyć na język SQL.

Gdyby interesowały nas jedynie wyniki spełniające pewien warunek, rozbudowalibyśmy nasze zapytanie, dodając do niego klauzulę **WHERE** (ang.: *gdzie*). Na przykład, gdyby interesowały nas wyłącznie imiona użytkowników, których identyfikator jest większy niż 5, zadalibyśmy następujące zapytanie:

```
SELECT first_name FROM users WHERE id>5;
```

Od bazy danych można pobrać także wyniki posortowane według danej kolumny (lub wielu kolumn). Dopuszczalne jest zarówno sortowanie w kolejności rosnącej jak i malejącej, niezależnie od typu danych (liczby, ciągi znaków, a nawet daty). Służy do tego klauzula **ORDER BY**:

```
SELECT * FROM users WHERE `last name`!="Kowalski" ORDER BY id;
```

Domyślnie przyjmowana jest kolejność rosnąca (**ASC**, od angielskiego *ascending - rosnąco*). Możemy ją zmienić na malejącą dodając słowo kluczowe **DESC** (od angielskiego *descending – malejąco*):

```
SELECT id FROM users ORDER BY first_name DESC;
```

Gdybyśmy chcieli sortować wyniki po więcej niż jednej kolumnie, składnia zapytania prezentowałaby się następująco:

```
SELECT * FROM users ORDER BY first_name DESC, `last_name` ASC;
```

[3] Znak grawisu (`) znajduje się na tym samym klawiszu klawiatury co tylda (~). Nie myl go z pojedynczym cudzysłowem ('), są to dwa różne znaki. o innym znaczeniu w języku SQL!

WYPISYWANIE DANYCH O MIESZKAŃCACH

Zacznijmy od implementacji pierwszej funkcji – zwracania wszystkich informacji o mieszkańcach. Jak zostało przedstawione wcześniej (na poprzednich zajęciach oraz powtórzone w poprzedniej, opcjonalnej części niniejszej instrukcji), służy do tego polecenie języka SQL *SELECT * FROM users*, gdzie *users* jest nazwą tabeli zawierającej dane, a znak „*" oznacza „wszystkie pola".

Naśladując rozwiązanie zaproponowane w klasie ModułBazyDanych, zaimplementujemy funkcje w następujący sposób:

```
def wypiszMieszkańców(self, kursor):
    try:
        kursor.execute("SELECT * FROM users")

    except mysql.connector.Error as błąd:
        print("Błąd w zapytaniu SQL:",błąd)

    else:
        for wiersz in kursor:
            print(wiersz)
```

Możesz uruchomić aplikację i spróbować wybrać odpowiednią opcję z menu. Wynik możesz zaobserwować w konsoli:

```
Lista modułów:
======================
[ M ]    Moduł obsługi mieszkańców
[ BD ]   Moduł bazy danych
[ KONIEC ]       Zakończ program
Wybierz moduł: M

Moduł obsługi mieszkańców:
======================
[1] Wypisz dane mieszkańców
[2] Wypisz identyfikatory mieszkańców
[3] Wypisz liczbę mieszkańców
[4] Wyszukaj mieszkańca po identyfikatorze
[KONIEC] Wyjdź z modułu
Wybierz opcję z menu: 1
{'id': 3, 'first_name': 'Jan', 'last name': 'Kowalski'}
{'id': 4, 'first_name': 'Janina', 'last name': 'Kowalska'}
{'id': 5, 'first_name': 'Adam', 'last name': 'Kowalski'}
{'id': 6, 'first_name': 'Adam', 'last name': 'Mickiewicz'}
{'id': 7, 'first_name': 'Celina', 'last name': 'Mickiewicz'}
```

Miej na uwadze, że niniejsze zajęcia mogą rozpocząć się od bardzo krótkiego sprawdzianu wiedzy. Niezależnie od pytania, odpowiedź należy oddać na kartce „w kratkę", z jednolicie zamalowaną trzecią kratką od dołu, z prawej strony. Jest to warunek otrzymania oceny pozytywnej, udowadniający, że uczestnik zapoznał się z instrukcją zanim rozpoczął zajęcia.

LISTY ASOCJACYJNE

Zwróć uwagę na formę w jakiej zwrócony został wynik. Nie jest to zwykła lista, gdyż indeksy nie są liczbami (0,1,2...), ale ciągami znaków ('id', 'first_name', 'last name'). Takie struktury nazywamy **listami asocjacyjnymi**.

Obsługa list asocjacyjnych różni się jedynie w niewielkim stopniu od obsługi list klasycznych. Odwołując się do konkretnego elementu takiej listy, odwołujemy się podając ciąg znaków. Przykładowo, następujący kod wypisze nazwisko domownika wskazanego przez kursor:

```
for wiersz in kursor:
    print(wiersz['last name'])
```

Składnia definiowania nowej listy asocjacyjnej jest następująca:

```
nowyMieszkaniec = {'id': 10, 'first_name': "Nowa", 'last name': "Osoba"}
```

OBSŁUGA FUNKCJI *WYPISZIDENTYFIKATORY* ORAZ *WYSZUKAJMIESZKAŃCA*

Implementacja pozostałych funkcji będzie wyglądać bardzo podobnie do implementacji funkcji *wypiszMieszkańców*. Różnić się będą dwa fragmenty – treść zapytania SQL skierowanego do bazy danych oraz sposób wypisania informacji na ekranie. W funkcji *wyszukajMieszkańca* dodatkowo poprosimy użytkownika naszej aplikacji o podanie identyfikatora.

Zacznijmy od funkcji *wypiszIdentyfikatory*. Wynik wypiszemy w jednej linijce, będzie to wyglądało bardziej estetycznie dla użytkownika naszej aplikacji. W związku z tym implementacja funkcji przyjmie następującą postać:

```
def wypiszIdentyfikatory(self, kursor):
    try:
        kursor.execute("SELECT id FROM users")

    except mysql.connector.Error as błąd:
        print("Błąd w zapytaniu SQL:",błąd)

    else:
        for wiersz in kursor:
            print(wiersz['id'],  end =" ")
```

Poświęć stosowną ilość czasu na analizę powyższego kodu. Zwróć uwagę na treść zapytania SQL oraz na to, w jaki sposób dokonujemy wypisania wyniku.

Ten sam sposób wykorzystamy do implementacji funkcji *wyszukajMieszkańca*. Zapytania rozbudujemy o klauzulę *WHERE*. Warunkiem dopasowania wiersza do wyniku będzie fakt, że kolumna *id* zawiera podany przez nas identyfikator.

```
def wyszukajMieszkańca(self, kursor):
    id = int(input("Podaj identyfikator mieszkańca: "))
    try:
        kursor.execute("SELECT `last name`,first_name FROM users "
                       "WHERE id={}".format(id))

    except mysql.connector.Error as błąd:
        print("Błąd w zapytaniu SQL:",błąd)

    else:
        for wiersz in kursor:
            print("ID {} posiada mieszkaniec {} {}"
                  .format(id,wiersz['first_name'],wiersz['last name']))
```

W jaki sposób wypisać liczbę mieszkańców? Moglibyśmy wynik zapisać do listy i pobrać ilość zapisanych w niej elementów za pomocą funkcji *len*. Język SQL oferuje prostszy sposób na wykonanie postawionego przed nami zadania – funkcję wbudowaną *COUNT*.

PODSTAWOWE FUNKCJE W JĘZYKU SQL

Język SQL oferuje wiele funkcji, które mogą okazać się pomocne w tworzeniu i przetwarzaniu zapytań.

Funkcje z których będziemy korzystać w trakcie zajęć przedstawiono w poniższej tabeli:

Nazwa	Przykład	Opis
MIN	MIN(`id`)	Wybiera najmniejszą wartość spośród wszystkich wartości danej kolumny zwróconych w wyniku
MAX	MAX(`id`)	Analogicznie do MIN, ale zwraca wartość maksymalną
AVG	AVG(`id`)	Oblicza i zwraca średnią wartość wszystkich wartości danej kolumny zwróconych w wyniku
SUM	SUM(`id`)	Sumuje wszystkie wartości danej kolumnę zwrócone w wyniku
COUNT	COUNT(*)	Zlicza wiersze zwrócone w wyniku
AND	-	Iloczyn logiczny (‘i’)
OR	-	Suma logiczna (‘lub’)
NOT	-	Negacja logiczna (‘nie’)

Pełną listę funkcji można znaleźć w dokumentacji języka: https://dev.mysql.com/doc/refman/8.0/en/func-op-summary-ref.html

WYPISYWANIE LICZBY MIESZKAŃCÓW

Do implementacji zliczania mieszkańców wykorzystamy właśnie funkcję COUNT. Nasze zapytanie będzie miało następującą postać:

```
SELECT COUNT(*) FROM users;
```

Dobrym rozwiązaniem jest także zmiana tekstu komunikatu w zależności od tego, czy w naszej bazie znajduje się jeden, czy więcej mieszkańców. W związku z tym, kod języka Python będzie wyglądać tak, jak przedstawiono poniżej:

```
def wypiszLiczbęMieszkańców(self, kursor):
    try:
        kursor.execute("SELECT COUNT(*) FROM users")

    except mysql.connector.Error as błąd:
        print("Błąd w zapytaniu SQL:",błąd)

    else:
        for wiersz in kursor:
            liczbaUżytkowników = wiersz["COUNT(*)"]
            if(liczbaUżytkowników==1):
                print("W bazie znajduje się 1 mieszkaniec.")
            else:
                print("W bazie znajduje się {} mieszkańców."
                      .format(liczbaUżytkowników))
```

Przetestuj działanie całego modułu. Jeżeli wszystko zostało wykonane poprawnie, wszystkie funkcje powinny zwracać prawidłowe wyniki. Jeżeli występują błędy, poświęć czas, by je naprawić.

MODUŁ INFORMACJI O URZĄDZENIACH

Utwórz nowy plik i nazwij go *modułInformacjiOUrządzeniach*.py. Utwórz w nim klasę *ModułInforamcjiOUrządzeniach*. Przypisz jej identyfikator „*U*" oraz nazwę „*Moduł informacji o urządzeniach*". Zarejestruj ją na liście modułów w pliku *main*.

Moduł będzie oferował następujące opcje:

Opcja	Funkcja w programie	Kod języka SQL
wypisanie wszystkich danych o urządzeniach	wypiszUrządzenia(self, kursor)	`SELECT * FROM devices`
wypisanie listy urządzeń w konkretnym pomieszczeniu	wypiszWPomieszczeniu(self, kursor)	`SELECT * FROM devices WHERE room_id = {}`
wypisanie rodzajów urządzeń	wypiszRodzaje(self, kursor)	`SELECT DISTINCT name FROM devices`

Zadbaj o estetykę wypisywanych informacji oraz o poprawność działania. W funkcji *wypiszWPomieszczeniu* należy pobrać od użytkownika aplikacji id pomieszczenia.

ZADANIA DO SAMODZIELNEGO WYKONANIA

ZADANIE 1 Ⓟ

Zmień działanie funkcji *wypiszWPomieszczeniu* w taki sposób, aby użytkownik aplikacji mógł podać nazwę pomieszczenia zamiast jego identyfikatora. Na przykład, po podaniu na wejście „*Przedpokój*", program powinien wypisać dane urządzeń o identyfikatorach 6 i 7 (odpowiednio *Żyrandol* i *Termometr*).

ZADANIE 2 Ⓟ

Zmień działanie funkcji *wypiszRodzaje* w taki sposób, aby dodatkowo informowała ile urządzeń tego typu znajduje się w domu.

ZADANIE 3 Ⓟ

Dodaj do modułu informacji o użytkownikach opcję „Wyświetl użytkowników według płci". Użytkownik programu powinien móc wybrać płeć (kobieta/mężczyzna), a program, wykonując jedno zapytanie na bazie danych, zwrócić tylko osoby o konkretnej płci. Do identyfikacji płci, posłuż się zapisanym w bazie imieniem (imiona kobiet kończą się na literę „a").

Autor:	Mgr inż. Paweł Stawarz, 4.03.2020
Korekta:	Mgr inż. Mateusz Cząstka, programista, Aspexi