INFORMATYKA II

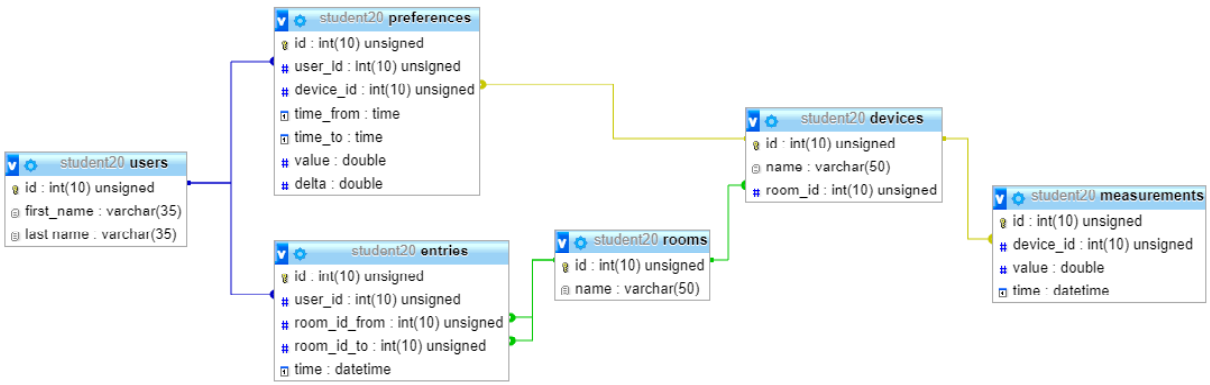
MANIPULACJA DANYMI W JĘZYKU SQL

CEL LABORATORIUM

Celem zajęć jest zapoznanie się z podzbiorem poleceń języka SQL, dotyczącym manipulacji danymi. Podczas zajęć przedstawione zostaną polecenia INSERT INTO, UPDATE SET i DELETE FROM, odpowiednio wstawiające, modyfikujące lub usuwające dane ze wskazanej tabeli. Pokazana zostanie składnia tych poleceń, przykłady ich użycia oraz zalecenia dotyczące bezpieczeństwa.

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 pakiet XAMPP. O tym skąd go pobrać, jak zainstalować i wykorzystać, dowiedzieć się można z materiałów pomocniczych udostępnionych pod adresem http://www.stawarz.edu.pl/informatyka2/dodatek_xampp.pdf.

Program opracowany na poprzednich zajęciach (w wersji okrojonej), można pobrać pod adresem http://www.stawarz.edu.pl/informatyka2/Lab3.rar (lub w wersji ZIP)

NIEBEZPIECZEŃSTWO

Istnieją trzy operacje modyfikacji danych: wstawianie nowych danych, edycja istniejących bądź usuwanie. W przeciwieństwie do poznanego podczas poprzednich zajęć wybierania danych, operacje te powodują **trwałe skutki** w bazie danych i należy

je wykonywać z należytą **uwagą i starannością**. Operacji tych **nie można cofnąć**[1]. Zanim zleci się silnikowi bazy danych wykonanie jakiegokolwiek z tych poleceń, należy upewnić się, że wszystkie warunki są obecne.

UPRAWNIENIA

Baza danych wykorzystywana na zajęciach została skonfigurowana w sposób **uniemożliwiający** użycia poleceń *INSERT, UPDATE i DELETE* na wszystkich tabelach, oprócz tabeli *preferences*. Próba wykonania tych poleceń na innych tabelach skończy się wyświetleniem błędu informującego o **braku dostępu**.

Jeżeli wykonujesz niniejszą instrukcję w domu, to oczywiście polecenia *INSERT, UPDATE i DELETE* będą działać na każdej tabeli, o ile nie zmieniono uprawnień użytkownika.

PRZECHOWYWANIE CZASU W BAZIE DANYCH

Oprócz klasycznych typów danych, takich jak liczby całkowite, zmiennoprzecinkowe, czy ciągi znaków, które są znane z nauki języka Python, bazy danych mogą przechowywać dane w wielu innych formatach. Jednym z najczęściej spotykanych i pożytecznych typów, są typy i funkcje pozwalające przechowywać datę i czas.

Warto wspomnieć, że istnieją zarówno typy pozwalające przechować sam czas, samą datę jak i obie te wartości jednocześnie. Sposób przechowywania zależy od silnika bazy danych oraz od typu pola. Tabela poniżej prezentuje wybrane typy danych:

Typ	Domyślny sposób przechowywania danych
TIME	Przechowuje informacje o czasie, ograniczając się do godzin, minut i sekund. Format to „HH:mm:ss", gdzie „HH" oznacza godzinę w formacie 24-godzinnym, z wiodącym zerem, „mm" oznacza minuty, a „ss" sekundy. Przykładem poprawnej wartości zapisanej w polu o typie TIME może być np. „04:00:00", czy „12:59:59".
DATE	Przechowuje informacje o czasie, z dokładnością do dnia. Format to „YYYY-MM-DD", gdzie „YYYY" to rok w formacie czterocyfrowym, „MM" to miesiąc, a „DD" to dzień miesiąca. Przykładem poprawnej wartości zapisanej w polu o typie DATE może być np. „1000-01-01", czy „2019-03-25".
DATETIME	Przechowuje informacje o czasie, w tym zarówno godzinie, jak i dniu. Format to „YYYY-MM-DD HH:mm:ss". Przykładem poprawnej wartości zapisanej w polu o typie DATETIME może być np. „2020-03-25 13:35:27", czy „9999-12-31 23:59:59".
TIMESTAMP	Przechowuje te same informacje co typ DATETIME, ale w innym formacie. Informacja jest przechowywana w postaci ilości sekund jakie minęły od czasu rozpoczęcia epoki Unixa (północ 1 stycznia 1970 roku). Pobieranie i zwracanie informacji odbywa się w takim samym formacie jak dla typu DATETIME, różni się tylko metoda przechowywania.

[1] Możliwe jest cofnięcie zmian wykorzystując transakcje i komendę ROLLBACK, ale nie zostaną one przedstawione podczas zajęć.

Istnieje możliwość zmiany formatów reprezentacji danych wspomnianych w tabeli powyżej, jednak w rozważanym przez nas przypadku formaty zapisu są zgodne. Dane czasowe przekazujemy jak zwykły ciąg znaków. Ważne jest więc by pamiętać, żeby otoczyć je cudzysłowami.

TABELA PREFERENCES

Struktura tabeli preferences, z którą będziemy pracować, wygląda następująco:

NAZWA KOLUMNY	TYP	KOMENTARZE
Id	Liczba całkowita	Automatycznie zwiększany
user_id	Liczba całkowita	Klucz obcy
device_id	Liczba całkowita	Klucz obcy
time_from	Czas w formacie TIME	Domyślnie NULL
time_to	Czas w formacie TIME	Domyślnie NULL
value	Liczba zmiennoprzecinkowa	
delta	Liczba zmiennoprzecinkowa	Domyślnie NULL

Podczas dodawania i edycji danych należy zwrócić uwagę czy wartości kluczy obcych wskazują na poprawne dane w stosownych tabelach oraz czy wartości pól *time_from* i *time_to* są w odpowiednim formacie.

MODUŁ USTAWIEŃ

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

Utwórz nowy plik. Nazwij go „modułUstawień.py". Znajdzie się w nim moduł pozwalający wyświetlać i modyfikować preferencje użytkowników dotyczące pracy poszczególnych urządzeń w domu. W pliku utwórz nową klasę dziedziczącą z klasy Moduł i nazwij ją „ModułUstawień". Przypisz jej identyfikator „V" i nazwę „Moduł ustawień". Zaimplementuj metodę przejmijKontrolę w sposób analogiczny do innych modułów.

Pamiętaj, że moduł należy zarejestrować w aplikacji, zanim będziemy mogli go wykorzystać.

WSTAWIANIE DANYCH

Aby umieścić w bazie danych nowy wiersz, należy użyć polecenia **INSERT INTO**. Wywołując polecenie musimy podać wartości **wszystkich** pól nowego wiersza, które nie mają wartości domyślnych. Często możemy uniknąć podania wartości klucza głównego tabeli, jeżeli jest on opisany typem liczbowym oraz wartości pola przechowującego czas, gdyż domyślnie zostanie pobrany aktualny czas z systemu.

Składnia polecenia INSERT wygląda następująco:

```
INSERT INTO tabela [ (listaNazwKolumn) ]
VALUES (wartośćPola1, wartośćPola2, … , wartośćPolaN);
```

Lista kolumn jest **opcjonalna**. Nazwy kolumn powinny zostać oddzielone przecinkiem. Jeżeli lista kolumn nie zostanie podana, system bazy danych będzie oczekiwał, że podamy wartości wszystkich kolumn tabeli.

Lista wartości musi zawierać tyle wartości, ile kolumn zawiera lista kolumn. Jeżeli nie podaliśmy listy kolumn, lista wartości musi zawierać wartości **wszystkich** kolumn tabeli **w kolejności** takiej, jaka jest kolejność pól w bazie danych.

Przykładowo, jeżeli chcielibyśmy dodać nowego mieszkańca do naszej bazy danych, możemy zrobić to w następujący sposób:

```
INSERT INTO users
VALUES (12, „Imię”, „Nazwisko”);
```

Możemy też zdecydować się ominąć pole id, ale musimy to wyraźnie zaznaczyć:

```
INSERT INTO users (first_name, `last name`)
VALUES („Imię”, „Nazwisko”);
```

Natomiast wstawienie danych do tabeli preferences może wyglądać następująco:

```
INSERT INTO preferences (id, user_id, device_id, time_from, value)
VALUES (NULL, 7, 6, „20:30:00”, 3.5);
```

Podanie wartości *NULL* dla pola *id* oznacza, że *id* zostanie wygenerowane automatycznie przez bazę danych.

EDYCJA DANYCH

Dane obecne w bazie danych można również zmienić. W tym celu należy użyć polecenia **UPDATE**. Składnia polecenia *UPDATE* prezentuje się następująco:

```
UPDATE nazwaTabeli
SET kolumna1 = nowaWartość1 [, kolumna2 = nowaWartość2 [, … ] ]
[ WHERE warunek ];
```

Zbiór nazw kolumn wraz z nowymi wartościami może być dowolnie długi, same zaś kolumny mogą być w dowolnej kolejności. Zmienione zostaną wartości kolumn tylko tych wierszy, które **spełniają warunek** podany w klauzuli *WHERE*. Jeżeli klauzula WHERE zostanie ominięta, zmienione zostaną wartości **wszystkich** wierszy tabeli.

Chcąc zmienić nazwisko **wszystkich** użytkowników na „Nowak", wykonalibyśmy więc następujące polecenie języka SQL:

```
UPDATE users
SET `last name` = „Nowak";
```

Natomiast chcąc zmienić nazwisko wyłącznie Adama Kowalskiego na „Nowak", wykonalibyśmy następujące polecenie języka SQL:

```
UPDATE users
SET `last name` = „Nowak"
WHERE first_name = „Adam" AND `last name` = „Kowalski";
```

USUWANIE DANYCH

Ostatnią (i zarazem mogącą wyrządzić **największe szkody**) operacją jest usuwanie danych. Służy do tego polecenie **DELETE**. Składnia tego polecenia jest zadziwiająco prosta i wygląda w następujący sposób:

```
DELETE FROM nazwaTabeli
[ WHERE warunek ];
```

Tak jak i w przypadku polecenia **UPDATE**, pominięcie klauzuli WHERE spowoduje wykonanie działania (co w przypadku polecenia DELETE oznacza usunięcie) na wszystkich wierszach tabeli. Przykładowo, poniższa komenda usunie z bazy danych wszystkie dane o pokojach:

```
DELETE FROM rooms;
```

Chcąc ograniczyć się do konkretnego pokoju (lub pokojów), należy rozszerzyć polecenie o klauzulę WHERE. Usunięcie z bazy danych wyłącznie rekordów dotyczących salonu i kuchni, miałoby następującą postać:

```
DELETE FROM rooms
WHERE name = „Salon" OR name = „Kuchnia";
```

ZADANIA DO SAMODZIELNEGO WYKONANIA

ZADANIE 1

Do modułu ustawień dodaj funkcję *wypiszPreferencjeMieszkańca*. Po jej wywołaniu użytkownik zostanie poproszony o podanie imienia i nazwiska mieszkańca. Po podaniu wartości, program powinien wypisać wszystkie preferencje, jakie ma dany mieszkaniec, posortowane względem czasu rozpoczęcia i w następującej formie:

```
Nazwa urządzenia (od H1:M1:S1 do H2:M2:S2) – wartość (+/- delta)
```

Na przykład:

```
Grzejnik (od 06:00:00 do 12:00:00) – 100 (+/- 10)
```

Przetestuj działanie funkcji dla Adama Mickiewicza. Weź pod uwagę, że czasy i dopuszczalna różnica (delta) mogą mieć wartość NULL.

ZADANIE 2

Do modułu ustawień dodaj funkcję *dodajPreferencję*. Po jej wywołaniu użytkownik zostanie poproszony o podanie imienia i nazwiska mieszkańca. Następnie program powinien zapytać o nazwę urządzenia. Jeżeli w domu znajduje się wiele urządzeń danego typu, program powinien zapytać o nazwę pokoju, w którym urządzenie się znajduje, i na tej podstawie ustalić, o które dokładnie urządzenie chodzi. Na koniec użytkownik powinien zostać poproszony o podanie czasu rozpoczęcia, zakończenia, wartości i dopuszczalnej różnicy wartości. Po otrzymaniu od użytkownika tych informacji, dodaj wpis do bazy danych.

ZADANIE 3

Do modułu ustawień dodaj funkcję *edytujPreferencję*. Po jej wywołaniu użytkownik zostanie poproszony o podanie imienia i nazwiska mieszkańca. Następnie program powinien zapytać o nazwę urządzenia. Jeżeli w domu znajduje się wiele urządzeń danego typu, program powinien zapytać o nazwę pokoju, w którym urządzenie się znajduje, i na tej podstawie ustalić, o które dokładnie urządzenie chodzi. Jeżeli istnieje preferencja dotycząca tego mieszkańca i tego ustawienia, użytkownik powinien zostać poproszony o podanie czasu rozpoczęcia, zakończenia, wartości i dopuszczalnej różnicy wartości. Po otrzymaniu od użytkownika tych informacji, zmień odpowiedni wpis w bazie danych. Jeżeli preferencja nie istniała, należy wyświetlić komunikat „użytkownik nie ma preferencji dotyczących tego urządzenia".

ZADANIE 4 Ⓟ

Do modułu ustawień dodaj funkcję *wypiszPreferencjeUrządzenia*. Po jej wywołaniu użytkownik zostanie poproszony o podanie nazwy urządzenia. Jeżeli w domu znajduje się wiele urządzeń danego typu, program powinien zapytać o nazwę pokoju, w którym urządzenie się znajduje, i na tej podstawie ustalić, o które dokładnie urządzenie chodzi. Po otrzymaniu wartości program powinien wypisać wszystkie preferencje dotyczące danego urządzenia, posortowane względem nazwiska mieszkańca i w następującej formie:

```
Nazwisko Imię (od H1:M1:S1 do H2:M2:S2) – wartość (+/- delta)
```

Na przykład:

```
Mickiewicz Adam (od 06:00:00 do 12:00:00) – 100 (+/- 10)
```

Weź pod uwagę, że czasy i dopuszczalna różnica (delta) mogą mieć wartość NULL. Jeżeli żaden mieszkaniec nie ma preferencji dotyczących danego urządzenia, powinien zostać wyświetlony komunikat „Brak preferencji".

ZADANIE 5 Ⓟ

Do modułu ustawień dodaj funkcję *usuńPreferencję*. Po jej wywołaniu użytkownik zostanie poproszony o podanie imienia i nazwiska mieszkańca. Następnie program powinien zapytać o nazwę urządzenia. Jeżeli w domu znajduje się wiele urządzeń danego typu, program powinien zapytać o nazwę pokoju, w którym urządzenie się znajduje, i na tej podstawie ustalić, o które dokładnie urządzenie chodzi. Jeżeli istnieje preferencja dotycząca tego mieszkańca i tego ustawienia, należy ją usunąć i wyświetlić komunikat „preferencja usunięta".

ZADANIE 6 Ⓟ

Do modułu ustawień dodaj funkcję *usuńPreferencjeMieszkańca*. Po jej wywołaniu użytkownik zostanie poproszony o podanie imienia i nazwiska mieszkańca. Jeżeli istnieją preferencje dotyczące tego mieszkańca, należy je usunąć i wyświetlić komunikat „Usunięto X preferencji", gdzie „X" jest liczbą usuniętych preferencji.

ZADANIE 7 Ⓟ

Do modułu ustawień dodaj funkcję *usuńPreferencjeUrządzenia*. Po jej wywołaniu użytkownik zostanie poproszony o podanie nazwy urządzenia. Jeżeli w domu znajduje się wiele urządzeń danego typu, program powinien zapytać o nazwę pokoju, w którym urządzenie się znajduje, i na tej podstawie ustalić, o które dokładnie urządzenie chodzi. Jeżeli istnieją preferencje dotyczące tego urządzenia, należy je usunąć i wyświetlić komunikat „Usunięto X preferencji", gdzie „X" jest liczbą usuniętych preferencji.

ZADANIE 8 Ⓟ (NAGRADZANE 2 PLUSAMI)

Do modułu ustawień dodaj funkcję skopiujPreferencjeMieszkańca. Po jej wywołaniu użytkownik zostanie poproszony o podanie imienia i nazwiska mieszkańca. Jeżeli istnieją preferencje dotyczące tego mieszkańca, należy zapytać użytkownika o podanie imienia i nazwiska innego mieszkańca, a następnie skopiować wszystkie preferencje mieszkańca pierwszego i przypisać do mieszkańca drugiego. Po skończonej operacji program powinien wyświetlić komunikat „Skopiowano X preferencji", gdzie „X" jest liczbą skopiowanych preferencji.

ZADANIE 9 Ⓟ (NAGRADZANE 2 PLUSAMI)

Do modułu ustawień dodaj funkcję przenieśPreferencjeMieszkańca. Po jej wywołaniu użytkownik zostanie poproszony o podanie imienia i nazwiska mieszkańca. Jeżeli istnieją preferencje dotyczące tego mieszkańca, należy zapytać użytkownika o podanie imienia i nazwiska innego mieszkańca, a następnie przenieść wszystkie preferencje mieszkańca pierwszego i przypisać do mieszkańca drugiego, po czym wyświetlić komunikat „Przeniesiono X preferencji", gdzie „X" jest liczbą przeniesionych preferencji.

ZADANIE 10 Ⓟ (NAGRADZANE 2 PLUSAMI)

Do modułu ustawień dodaj funkcję skopiujPreferencjeUrządzenia. Po jej wywołaniu użytkownik zostanie poproszony o podanie nazwy urządzenia. Jeżeli w domu znajduje się wiele urządzeń danego typu, program powinien zapytać o nazwę pokoju, w którym urządzenie się znajduje, i na tej podstawie ustalić, o które dokładnie urządzenie chodzi. Jeżeli istnieją preferencje dotyczące tego urządzenia, należy zapytać użytkownika o podanie nazwy innego urządzenia (i znów o pokój, jeżeli to konieczne), a następnie skopiować wszystkie preferencje urządzenia pierwszego i przypisać do urządzenia drugiego. Po skończonej operacji program powinien wyświetlić komunikat „Skopiowano X preferencji", gdzie „X" jest liczbą skopiowanych preferencji.

Autor:	Mgr inż. Paweł Stawarz, 27.03.2020
Korekta:	Jakub Rzucidło, .Net Developer, Sii sp. z.o.o.