Instrukcje grupowania w języku SQL, etap III

- Wprowadzenie
- Przeczytaj
- Prezentacja multimedialna
- Sprawdź się
- Dla nauczyciela

Instrukcje grupowania w języku SQL, etap III

Źródło: Tobias Fischer, domena publiczna.

Współczesne bazy danych zawierają bardzo dużo informacji, do których dostęp uzyskujemy za pomocą przeglądarek internetowych i aplikacji webowych. Po stronie serwerów muszą działać wydajne mechanizmy wybierania i analizowania danych zapisanych w bazach. Jednym ze stosowanych rozwiązań są skrypty pisane między innymi w językach Python lub PHP. Pozwalają one na łatwą komunikację z bazami, wykonywanie zapytań w języku SQL i analizę wydobywanych informacji.

Więcej informacji na temat instrukcji grupowania w języku SQL znajdziesz w e-materiałach:

- Instrukcje grupowania w języku SQL, etap I,
- Instrukcje grupowania w języku SQL, etap II,
- Instrukcje grupowania w języku SQL, etap IV.

Twoje cele

- Wykonasz zapytania za pomocą skryptu języka Python.
- Skonstruujesz zapytania grupujące dane.
- Zbadasz działanie funkcji agregujących.

Przeczytaj

W bazie SQLite3 zapisanej w pliku `portal.db` znajdują się dane dotyczące użytkowników pewnego portalu internetowego. Plik pobieramy i zapisujemy w wybranym katalogu.

Plik bazy SQLite3 `portal.db`
Plik o rozmiarze 64.00 KB w języku polskim

Do wykonywania operacji na bazie danych w tym materiale wykorzystamy skrypt napisany w języku Python oraz moduł `sqlite3` przeznaczony do obsługi baz SQLite3.

> Podstawy pracy z bazą danych z wykorzystaniem skryptów Pythona zostały omówione w materiale Instrukcje wyszukiwania w języku SQL, etap III.
>
> Ponieważ pracujemy na bazie SQLite3, instrukcje SQL zamieszczone w tym materiale możemy również wykonywać:
>
> - w programie SQLiteStudio,
> - w wierszu poleceń bazy SQLite3.
>
> Informacje na temat instalacji i korzystania z tych narzędzi zamieszczone zostały w materiale Definiowanie schematu bazy danych w języku SQL, etap II.

Schemat bazy

Poznajmy zawartość bazy. W wybranym edytorze tworzymy skrypt, w którym umieszczamy następujący kod:

```
1  import sqlite3
2
3  def wykonaj_zapytanie(kursor, zapytanie):
4      kursor.execute(zapytanie)
5      for rekord in kursor.fetchall():
6          print(*rekord)
7
8  baza = 'portal.db'
9  pol = sqlite3.connect(baza)  # połączenie z bazą
10 kursor = pol.cursor()  # utworzenie obiektu kursora
11
12 kwerenda = "SELECT name, sql FROM sqlite_master WHERE type='table
13
14 wykonaj_zapytanie(kursor, kwerenda)
```

Do nawiązania połączenia z bazą wykorzystujemy metodę `connect()`. Przekazujemy jej nazwę pliku, w którym zapisana jest baza. Wykonywanie operacji na bazie umożliwia obiekt kursora zwracany przez metodę `cursor()`. Kursor udostępnia metodę `execute()` pozwalającą wykonywać zapytania SQL podane jako argument. Wyniki zapytania zwraca w postaci listy rekordów metoda `fetchall()`.

Plik zapisujemy pod nazwą `kwerendy_portal.py` w tym samym katalogu, w którym zapisaliśmy wcześniej bazę danych.

> **Polecenie 1**
>
> Uruchom skrypt `kwerendy_portal.py`. Przeanalizuj wynik wykonania skryptu i opisz schemat bazy danych.

Zapytania grupujące i funkcje agregujące

> **Polecenie 2**
>
> Napisz zapytanie, które zwróci liczbę użytkowników zapisanych w bazie.

Rozwiązanie polecenia wymaga zliczenia rekordów z tabeli *uzytkownicy*. Użyjemy funkcji agregującej `COUNT()`, która zlicza rekordy pasujące do zapytania:

```
SELECT COUNT(*) AS liczba_użytkowników
FROM uzytkownicy;
```

> **Ważne!**
>
> W skrypcie kod SQL kolejnych kwerend dla uproszczenia możemy zapisywać w zmiennej `kwerenda` umieszczanej przed końcową funkcją `wykonaj_zapytanie(kursor, kwerenda)`. Kolejne wystąpienia zmiennej będą nadpisywały poprzednie. Tym samym przy każdym uruchomieniu skryptu wykonywana będzie tylko jedna, ostatnia kwerenda. Dodatkowo możemy używać potrójnych cudzysłowów, aby umieszczać w skrypcie kwerendy zapisane dla czytelności w wielu liniach.
>
> Kod po dodaniu zapytania odpowiedzi na bieżące pytanie będzie wyglądał następująco:

```
kwerenda = "SELECT name, sql FROM sqlite_master WHERE type='tab
kwerenda = """
    SELECT COUNT(*) AS liczba_użytkowników
    FROM uzytkownicy;
```

```
"""

wykonaj_zapytanie(kursor, kwerenda)
```

Po wykonaniu kwerendy dowiemy się, że w bazie zapisano dane 84 użytkowników.

> **Polecenie 3**
>
> Napisz zapytanie, które zwróci nazwy państw oraz liczbę użytkowników z tych państw.

Odpowiedź wymaga pogrupowania użytkowników według identyfikatorów państw, a następnie zliczenia rekordów w poszczególnych grupach.

```sql
SELECT idp, COUNT(*) AS liczba_użytkowników
FROM uzytkownicy
GROUP BY idp;
```

W tym przykładzie widzimy, że grupowanie pozwala przeprowadzać obliczenia na polach rekordów wyodrębnionych według wartości kolumny podanej w klauzuli `GROUP BY`.

Wynik dotychczasowego zapytania nie jest czytelny, identyfikatory państw warto byłoby zastąpić nazwami. W tym celu dodamy do zapytania złączenie, które pozwoli pobrać nazwy państw z drugiej tabeli. Dodatkowo wyniki uporządkujemy malejąco według liczby użytkowników:

```sql
SELECT panstwa.nazwa, COUNT(*) AS liczba_użytkowników
FROM uzytkownicy
INNER JOIN panstwa ON uzytkownicy.idp = panstwa.idp
GROUP BY panstwa.nazwa
ORDER BY liczba_użytkowników DESC;
```

Początkowe zwrócone rekordy:

```
Stany Zjednoczone 8
Wielka Brytania 6
Niemcy 5
Polska 4
Kanada 4
```

> **Polecenie 4**
>
> Napisz zapytanie zwracające nazwy państw, z których w bazie mamy po jednym użytkowniku.

Możemy zauważyć, że potrzebujemy zapytania podobnego do poprzedniego, ale z dodatkowym warunkiem. W którym miejscu należy go dodać? Może w klauzuli `WHERE`? To jednak błędne przypuszczenie, ponieważ warunki z klauzuli `WHERE` nakładane są na rekordy przed ich grupowaniem. Ponadto żeby wybrać państwa z jednym użytkownikiem, trzeba wcześniej tych użytkowników policzyć. Z tego wynika, że warunek musi zostać nałożony po zgrupowaniu i policzeniu rekordów. W takich przypadkach stosujemy klauzulę `HAVING`, w której możemy używać funkcji agregujących. Kwerenda przyjmie następującą postać:

```
SELECT panstwa.nazwa
FROM panstwa, uzytkownicy
USING(idp)
GROUP BY panstwa.nazwa
HAVING COUNT(*) = 1;
```

W zapytaniu użyliśmy również klauzuli `USING()`, która może zastąpić klauzulę `INNER JOIN ON`, jeżeli pola tworzące złączenie mają taką samą nazwę w obu tabelach.

> W systemach bazodanowych innych niż SQLite składnia klauzuli `USING()` wymaga klauzuli `JOIN`:
>
> ```
> SELECT panstwa.nazwa
> FROM panstwa JOIN uzytkownicy
> USING(idp)
> GROUP BY panstwa.nazwa
> HAVING COUNT(*) = 1;
> ```

Początkowe zwrócone rekordy:

```
Algieria
Armenia
Austria
Chorwacja
Dania
```

> **Polecenie 5**
>
> Napisz zapytanie zwracające imiona i nazwiska kobiet, które zrobiły najwięcej zdjęć.

Zacznijmy od pogrupowania rekordów z tabeli *zdjęcia* według identyfikatorów użytkowników i policzenia ich:

```
SELECT COUNT(*) FROM zdjecia GROUP BY idu";
```

Żeby wyeliminować z grupowania mężczyzn, musimy dołączyć tabelę *uzytkownicy* i dodać odpowiedni warunek, tym razem w klauzuli `WHERE`.

```
SELECT imie, nazwisko, COUNT(*) AS liczba_zdjęć
FROM zdjecia, uzytkownicy
USING(idu)
WHERE plec = 'K'
GROUP BY zdjecia.idu
ORDER BY COUNT(*) DESC;
```

Przeanalizujmy działanie zapytania: wszystkie rekordy z tabeli *zdjecia* złączone z odpowiednimi rekordami z tabeli *uzytkownicy* zostają wybrane, następnie odrzucane są te, które nie spełniają warunku z klauzuli `WHERE`, reszta jest grupowana według identyfikatora użytkownika z tabeli *zdjecia* i zliczana, na koniec zostają posortowane malejąco według liczby zdjęć.

> **Ćwiczenie 1**
>
> Załóżmy, że chcemy zobaczyć dane tylko pięciu użytkowników, którzy dodali do bazy najwięcej zdjęć. Wystarczy, że uzupełnimy zapytanie klauzulą `LIMIT 5`. Przetestuj to rozwiązanie.

> **Ważne!**
>
> Przed grupowaniem można pozbyć się rekordów, które nas nie interesują, za pomocą warunku w klauzuli `WHERE`.
>
> **Polecenie 6**
>
> Napisz zapytanie zwracające imiona, nazwiska i liczbę znajomości zawartych przez użytkowników portalu.

Rozwiązanie zadania zaczniemy od wyświetlenia imion i nazwisk osób, które zawarły znajomość. W tabeli *znajomosci* mamy niepowtarzające się pary identyfikatorów użytkowników: *idu1* i *idu2*. Kwerenda, która pokaże te identyfikatory oraz imię i nazwisko pierwszego użytkownika, może wyglądać tak:

```
SELECT idu1, imie, nazwisko, idu2
FROM uzytkownicy
INNER JOIN znajomosci ON idu=idu1;
```

Początkowe zwrócone rekordy:

```
1 Joshua King 22
1 Joshua King 26
1 Joshua King 36
1 Joshua King 46
1 Joshua King 47
1 Joshua King 69
1 Joshua King 74
```

Żeby wyświetlić imiona i nazwiska osób, których identyfikator zawarty jest w polu *idu2*, przekształcimy powyższą kwerendę w podzapytanie:

```
SELECT q1.idu1, q1.imie, q1.nazwisko, q1.idu2, u.imie, u.nazwisko
FROM (
    SELECT idu1, imie, nazwisko, idu2
    FROM uzytkownicy
    INNER JOIN znajomosci ON idu=idu1
    ) q1, uzytkownicy u
WHERE u.idu =q1.idu2;
```

Początkowe zwrócone rekordy:

```
1 Joshua King 22 Frank Dijkstra
1 Joshua King 26 Joshua Nelson
1 Joshua King 36 Milan Petrovic
1 Joshua King 46 Laura Mello
1 Joshua King 47 Mia Hoffmann
1 Joshua King 69 Grace Doherty
```

```
1 Joshua King 74 Luka Wagner
...
22 Frank Dijkstra 2 Leonardo De Luca
22 Frank Dijkstra 8 Samuel Ramirez
22 Frank Dijkstra 36 Milan Petrovic
22 Frank Dijkstra 62 Filip Kralj
22 Frank Dijkstra 67 Artjoms Klavins
22 Frank Dijkstra 75 Elise Wouters
```

Po przejrzeniu wszystkich wyników zauważymy, że aby zliczyć wszystkie znajomości nawiązane przez użytkowników, trzeba wziąć pod uwagę znajomości, które nawiązali oni, i znajomości, które nawiązano z nimi.

Tworzymy dwa zapytania. Pierwsze pokaże imię, nazwisko oraz liczbę znajomości, które zawarli użytkownicy – rekordy zostaną zgrupowane według pola *idu1*. Drugie pokaże imię, nazwisko oraz liczbę znajomości, które z użytkownikami zawarły inne osoby – rekordy zgrupujemy według pola *idu2*. W kwerendach wykorzystamy aliasy:

```sql
SELECT imie, nazwisko, COUNT(idu1) AS z1
FROM uzytkownicy
INNER JOIN znajomosci ON idu=idu1
GROUP BY idu1;
```

Początkowe zwrócone rekordy:

```
Joshua King 7
Leonardo De Luca 8
Aleksandra Krawczyk 8
Luka Schulte 2
Jayden Harper 8
```

```sql
SELECT imie, nazwisko, COUNT(idu2) AS z2
FROM uzytkownicy
INNER JOIN znajomosci ON idu=idu2
GROUP BY idu2;
```

Początkowe zwrócone rekordy:

```
```

```
Joshua King 7
Leonardo De Luca 4
Aleksandra Krawczyk 13
Luka Schulte 5
Jayden Harper 6
```

Jeżeli popatrzymy na wyniki powyższych kwerend, zauważymy na przykład to, że użytkownik Leonardo De Luca zawarł osiem znajomości, a z nim zawarto cztery znajomości. Więc aby pokazać, ile znajomości zawarł użytkownik, musimy dopasować rekordy z obydwu kwerend według nazwisk użytkowników i dodać do siebie pola aliasowane jako *z1* oraz *z2*. W ten sposób przygotowane kwerendy wykorzystamy jako podzapytania.

```sql
SELECT q1.imie, q1.nazwisko, q1.z1 + q2.z2 AS liczba_znajomosci
FROM (
    SELECT imie, nazwisko, COUNT(idu1) AS z1
    FROM uzytkownicy
    INNER JOIN znajomosci ON idu=idu1
    GROUP BY idu1
    ) q1
INNER JOIN (
    SELECT imie, nazwisko, COUNT(idu1) AS z2
    FROM uzytkownicy
    INNER JOIN znajomosci ON idu=idu2
    GROUP BY idu2
    ) q2
ON q1.nazwisko = q2.nazwisko
ORDER BY liczba_znajomosci DESC;
```

Na początku wykonana będzie pierwsza kwerenda po klauzuli `FROM` nazwana `q1`, następnie druga wstawiona po klauzuli `INNER JOIN` nazwana `q2`. Rekordy obydwu kwerend zostaną połączone na podstawie identycznych nazwisk. Wtedy z kwerendy `q1` wybrane zostaną pola *imie* i *nazwisko*, a następnie przeprowadzone będzie sumowanie pól `z1` i `z2`, przez co otrzymamy pole wyliczeniowe nazwane *liczba_znajomosci*. Wyniki zostaną uporządkowane malejąco według liczby znajomości.

Początkowe zwrócone rekordy:

```
Aleksandra Krawczyk 21
Grace Doherty 21
Filip Kralj 20
```

```
Lautero Quiroga 18
Andreas Jakobsson 18
```

Słownik

funkcje agregujące

funkcje umożliwiające wykonywanie obliczeń na grupach rekordów, jak również wyszukiwanie i zliczanie rekordów spełniających określone warunki

grupowanie

tworzenie zbiorów rekordów na podstawie takich samych wartości z podanej kolumny; grupowanie zazwyczaj poprzedza zastosowanie funkcji agregującej, która przeprowadza obliczenia na grupie rekordów

podzapytanie

zapytanie `SELECT` umieszczone wewnątrz innego zapytania `SELECT`, podzapytanie wykonywane w pierwszej kolejności, a jego wyniki są źródłem dla zewnętrznego zapytania

pole wyliczeniowe

pole w zapytaniu, którego wartość obliczana jest dynamicznie z wykorzystaniem danych pobieranych z bazy

Prezentacja multimedialna

Polecenie 1

Zapoznaj się z prezentacją, w której omówiono zastosowanie funkcji agregujących i klauzuli `GROUP BY` języka SQL, a następnie wykonaj polecenie 2.

Polecenie 2

Zaprojektuj kwerendę zwracającą nazwy klas, imiona i nazwiska uczniów z klas humanistycznych, których średnia ocen jest większa lub równa niż 3.0. Nazwy klasy humanistycznych kończą się na literę „c”. Wyniki powinny zostać uporządkowane rosnąco według klas, a w obrębie klas alfabetycznie według nazwisk.

W pliku `uczniowie2.db` zamieszczono bazę danych SQLite3. Pobierz i zapisz plik. Do pracy z bazą możemy użyć:

- skryptu języka Python,
- programu SQLiteStudio,
- wiersza poleceń bazy SQLite3.

Baza danych SQLite3 `uczniowie2.db`
Plik o rozmiarze 832.00 KB w języku polskim

1

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBfR38Mj

Zapoznajmy się ze schematem bazy `Uczniowie`:

W tabeli *uczniowie* występują pola:

- *id_ucz* – `INTEGER`, klucz główny, identyfikator ucznia,
- *nazwisko*, *imie*, *ulica*, *nr_domu*, *klasa* – `VARCHAR`, dane tekstowe.

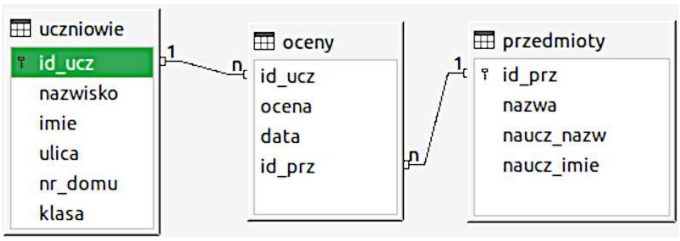
W tabeli *przedmioty* występują pola:

- *id_prz* – `INTEGER`, klucz główny, identyfikator przedmiotu,
- *nazwa*, *naucz_nazwa*, *naucz_imie* – `VARCHAR`, dane tekstowe.

W tabeli *oceny* występują pola:

- *id_ucz*, *id_prz* – `INTEGER`, klucze obce, identyfikatory ucznia i przedmiotu,
- *ocena* – `DECIMAL`, pole zawierające oceny,
- *data* – `DATE`, pole przechowujące datę wystawienia oceny w formacie ISO8601 "YYYY-MM-DD".

2



Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBfR38Mj

Tabele *uczniowie* i *przedmioty* łączy relacja wiele-do-wielu definiowana przez relacje jeden-do-wielu z każdej z tych tabel do tabeli *oceny*, w której zgromadzono informacje o ocenach każdego ucznia z różnych przedmiotów.

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBfR38Mj

Zasadę grupowania rekordów można zilustrować na kilka sposobów. Zacznijmy od napisania kwerendy zliczającej oceny wystawione przez poszczególnych nauczycieli.

Na początku wybierzemy wszystkie oceny i nauczycieli, którzy je wystawili. Dane pobierzemy z dwóch tabel: *przedmioty* i *oceny*.

W kwerendzie użyliśmy dla wygody aliasów. Kwerenda zwróci tyle rekordów, ile jest wystawionych ocen w bazie, tzn. `16 965`.

Podczas wykonywania kwerendy za pomocą skryptu języka Python (lub w wierszu poleceń bazy SQLite3) przy tak dużej liczbie wyników możemy mieć problem z zobaczeniem początkowych rekordów. Możemy wtedy ograniczyć liczbę zwracanych wyników (np. do 10) za pomocą umieszczenia na końcu zapytania klauzuli `LIMIT 10`.

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBfR38Mj

Skoro mamy wszystkie oceny wraz z nazwiskami nauczycieli, którzy je wystawili, możemy oceny pogrupować według nazwisk nauczycieli, a następnie je zliczyć.

W klauzuli `GROUP BY` podajemy nazwę pola, według którego grupujemy rekordy, funkcję agregującą `COUNT()` (lub inną) stosujemy w klauzuli `SELECT` na polu, którego wartości

chcemy zliczać. Dodatkowo sortujemy rekordy malejąco według liczby ocen.

Początkowe zwrócone rekordy:

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBfR38Mj

Gdybyśmy chcieli wiedzieć, ile ocen wystawiono z poszczególnych przedmiotów, wystarczy, że w poprzedniej kwerendzie zamienimy pole *naucz_nazw* z nazwiskiem nauczyciela na pole *nazwa* z nazwą przedmiotu:

Początkowe zwrócone rekordy:

Po przeanalizowaniu wyników warto zauważyć, że liczby zwróconych ocen są takie same jak w poprzedniej kwerendzie, ponieważ każdy przedmiot nauczany jest przez jednego nauczyciela.

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBfR38Mj

Kolejnym zadaniem będzie policzenie ocen wystawionych przez podanego nauczyciela (lub z podanego przedmiotu) w poszczególnych klasach.

Na początku wybierzemy nazwisko nauczyciela, przedmiot, klasę oraz wystawioną ocenę. Dane pobieramy z trzech tabel:

Początkowe zwrócone rekordy:

7

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBfR38Mj

Podobnie jak w kwerendach zliczających oceny wystawione przez nauczycieli lub z poszczególnych przedmiotów, tak i teraz rekordy zwrócone przez ostatnią kwerendę należy pogrupować według klas, a oceny zliczyć:

Początkowe zwrócone rekordy:

8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBfR38Mj

Przydatną informacją byłoby zestawienie pokazujące liczbę ocen wystawionych z przedmiotów w poszczególnych klasach.

Aby otrzymać takie wyniki, wystawione oceny trzeba zgrupować na początku według klas, a następnie w obrębie klas według przedmiotów. Na końcu należy je policzyć. Pierwsze zadanie wykonaliśmy w poprzedniej kwerendzie, teraz wystarczy dołożyć kolejny poziom grupowania:

Początkowe zwrócone rekordy:

9

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBfR38Mj

Jak widzieliśmy w ostatnim przykładzie, w klauzuli `GROUP BY` może wystąpić kilka pól, przy czym ich kolejność nie jest ważna. Zobaczmy jednak, co się stanie, jeżeli zmienimy kolejność grupowania w ostatnim zapytaniu:

Jak można zaobserwować poniżej, liczba ocen nie ulegnie zmianie, zmienia się natomiast kolejność zwracanych rekordów.

Początkowe zwrócone rekordy:

10

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBfR38Mj

Na zgrupowanych danych możemy wykonywać wiele operacji jednocześnie. Na przykład do zestawienia liczby ocen z poszczególnych przedmiotów w klasach możemy dodać średnią arytmetyczną ocen z przedmiotów. W tym celu dodajemy funkcję agregującą `AVG()` w klauzuli `SELECT`:

Początkowe zwrócone rekordy:

Średnie arytmetyczne ocen zostały dla czytelności zaokrąglone do dwóch miejsc po przecinku z wykorzystaniem funkcji `ROUND()`.

11

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBfR38Mj

Odczytanie z wyników poprzedniej kwerendy informacji o tym, z którego przedmiotu klasa ma największą średnią, może być trudne. Uporządkujmy więc średnie ocen rosnąco w obrębie klas.

Podczas sortowania, inaczej niż w grupowaniu, zachowujemy odpowiednią kolejność: na początku porządkujemy rekordy według klas rosnąco (domyślnie), a następnie w ramach klas według średniej ocen malejąco.

Początkowe zwrócone rekordy:

12

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBfR38Mj

Wyniki omawianych kwerend możemy filtrować w różny sposób. Najprostsza możliwość to nakładanie warunków w klauzuli `WHERE`.

Kwerenda, która wyświetli liczbę i średnią ocen z matematyki i fizyki w klasach pierwszych, może wyglądać następująco:

Na początku z wybieranych rekordów eliminowane są te, które nie spełniają nałożonych warunków w klauzuli `WHERE`, pozostałe zostają zgrupowane według klasy, a w ramach klas według przedmiotów i dopiero wtedy wykonywane jest zliczanie oraz obliczanie średniej.

Początkowe zwrócone rekordy:

13

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBfR38Mj

Stosując warunki w klauzuli `WHERE`, pamiętamy, że wykluczane w taki sposób rekordy nie są później brane pod uwagę podczas grupowania i stosowania funkcji agregujących. Jeżeli chcemy nakładać warunki na dane pogrupowane, na których wykonane zostały jakieś operacje agregujące, używamy klauzuli `HAVING`.

14

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBfR38Mj

Przygotujemy kwerendę, która wyświetli nazwy klas trzecich, które z polskiego mają średnią większą lub równą 3.0.

Zacznijmy od wybrania ocen spełniających założone warunki:

15

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBfR38Mj

Do poprzedniej kwerendy dodajemy grupowanie i funkcję agregującą:

16

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBfR38Mj

W ostatnim kroku na końcu zapytania dodajemy warunek z użyciem klauzuli `HAVING`:

Wynik kwerendy:

17

D. Richard Hipp, logo SQLite.
Źródło: en.wikipedia.org, domena publiczna.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PBfR38Mj

Wykonaj polecenie 2. Konstruując kwerendę, odpowiedz na pytania:

1. Z których tabel trzeba pobrać dane?
2. Według którego pola należy dane zgrupować?
3. Jaką funkcję agregującą i na jakim polu zastosować?
4. Kiedy przefiltrować rekordy względem nazw klas?

5. Według których pól uporządkować wyniki?

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Ustaw we właściwej kolejności podane niżej operacje, aby można było wykonać za ich pomocą zapytanie przy użyciu skryptu języka Python:

```
cur.fechall()
```

```
con = sqlite3.connect('baza.db')
```

```
cur = con.cursor()
```

```
cur.execute("SELECT * FROM tablica;")
```

Ćwiczenie 2

Dokończ zdanie.
Funkcje agregujące, np. `COUNT()` lub `AVG()` mogą działać...

- ◯ tylko na jednej grupie rekordów z tabeli.
- ◯ tylko na wszystkich rekordach tabeli.
- ◯ na jednej lub wielu grupach rekordów tabeli.

Ćwiczenie 3

Dokończ zdanie.
Klauzula `GROUP BY` działa...

- ◯ tylko na części rekordów tabeli.
- ◯ na wszystkich lub na części rekordów tabeli.
- ◯ tylko na wszystkich rekordach tabeli.

Ćwiczenie 4

Dokończ zdanie, wybierając właściwe wyrażenie.

Jeżeli będziemy chcieli usunąć z grupowania niektóre rekordy, użyjemy [ ].

`klauzuli USING` | `klauzuli HAVING.` | `klauzuli WHERE`

Ćwiczenie 5

Dokończ zdanie.
Grupowanie polega na łączeniu rekordów...

- ◯ w obrębie jednej tabeli na podstawie tej samej wartości we wskazanym polu.
- ◯ z różnych tabel na podstawie tożsamej wartości jakiegoś pola.
- ◯ w jednej tabeli lub z wielu tabel na podstawie klauzuli warunkowej `WHERE`.

Ćwiczenie 6

Uporządkuj klauzule.

- `SELECT`
- `FROM`
- `WHERE`
- `ORDER BY`
- `LIMIT`
- `INNER JOIN`
- `GROUP BY`
- `ON`

Ćwiczenie 7

Wskaż, co zwróci przedstawiona kwerenda.

```
SELECT p2.nazwa, COUNT(p2.nazwa)
FROM znajomosci z
INNER JOIN uzytkownicy u1 ON z.idu1 = u1.idu
INNER JOIN uzytkownicy u2 ON z.idu2 = u2.idu
INNER JOIN panstwa p1 ON u1.idp = p1.idp
INNER JOIN panstwa p2 ON u2.idp = p2.idp
WHERE u1.idp = u2.idp
GROUP BY p2.nazwa
ORDER BY COUNT(p2.nazwa) DESC;
```

- ◯ listę państw uporządkowaną malejąco według znajomości zawartych pomiędzy obywatelami tego samego państwa
- ◯ listę nazw państw z uporządkowaną malejąco liczbą zawartych znajomości przez ich obywateli
- ◯ uporządkowaną malejąco według zawartych znajomości listę nazw państw

Ćwiczenie 8

W oparciu o bazę danych `Portal` (omówioną w sekcji „Przeczytaj”) napisz zapytanie, które pokaże liczbę zdjęć dodanych do portalu przez kobiety i mężczyzn.

Baza danych SQLite3:

Plik bazy SQLite3 `portal.db`
Plik o rozmiarze 64.00 KB w języku polskim

Ćwiczenie 9

W oparciu o bazę danych `Portal` (omówioną w sekcji „Przeczytaj”) napisz zapytanie, które zwróci nazwy państw i liczbę zdjęć dodanych przez użytkowników z danego państwa. Dane powinny być uporządkowane malejąco według liczby dodanych zdjęć.

Dla nauczyciela

Autor: GroMar.eu
Przedmiot: Informatyka

Temat: Instrukcje grupowania w języku SQL, etap III

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.
>
> > Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:
> >
> > > 4) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym:
> > >
> > > > d) projektuje i tworzy relacyjną bazę złożoną z wielu tabel oraz sieciową aplikację bazodanową dla danych związanych z rozwiązywanym problemem, formułuje kwerendy, tworzy i modyfikuje formularze oraz raporty, stosuje język SQL do wyszukiwania informacji w bazie i do jej modyfikacji, uwzględnia kwestie integralności danych, bezpieczeństwa i ochrony danych w bazie,

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Wykonasz zapytania za pomocą skryptu języka Python.
- Skonstruujesz zapytania grupujące dane.
- Zbadasz działanie funkcji agregujących.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie MySQL w wersji 8.0 (lub nowszej).

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Instrukcje grupowania w języku SQL, etap III". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wyświetla uczniom temat, wskazuje cele zajęć oraz ustala z uczestnikami zajęć kryteria sukcesu.
2. Prowadzący prosi uczniów, aby zgłaszali swoje propozycje pytań do tematu. Jedna osoba może zapisywać je na tablicy. Gdy uczniowie wyczerpią swoje pomysły, a pozostały jakieś ważne kwestie do poruszenia, nauczyciel je dopowiada.

Faza realizacyjna:

1. **Praca z tekstem.** Nauczyciel ocenia, na podstawie informacji na platformie, stan przygotowania uczniów do zajęć. Jeżeli jest ono niewystarczające, prosi wybraną osobę o przedstawienie najważniejszych informacji z sekcji „Przeczytaj". W kolejnym kroku uczniowie w parach testują na swoich komputerach rozwiązania omówione w sekcji.
2. **Praca z multimedium.** Uczniowie zapoznają się z treścią prezentacji z sekcji „Prezentacja multimedialna". Następnie w parach wykonują polecenie 2, a kiedy skończą, porównują rozwiązanie z inną grupą.
3. Uczniowie w parach wykonują ćwiczenia nr 1–7. Nauczyciel sprawdza poprawność wykonanych zadań, omawiając je wraz z uczniami.

Faza podsumowująca:

1. Na koniec zajęć nauczyciel raz jeszcze wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie". W odniesieniu do ich realizacji dokonuje szczegółowej oceny rozwiązania zastosowanego przez wybranego ucznia.
2. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności.

Praca domowa:

1. Uczniowie wykonują ćwiczenie 8 z sekcji „Sprawdź się".

Wskazówki metodyczne:

- Treści w sekcji „Prezentacja multimedialna" można wykorzystać na lekcji jako podsumowanie i utrwalenie wiedzy uczniów.