BAZY DANYCH

Lab. 6 Połączenia relacji

Laboratorium nr 6

Temat: Połączenia relacji

Dotychczas omawiane zapytania zawsze dotyczyły jednej relacji. Możliwe jest jednak pisanie zapytań, które odczytują i łączą dane z wielu relacji. Celem tego ćwiczenia jest zapoznanie Państwa z mechanizmem połączeń, oraz notacją polecenia SELECT pozwalającą na ich wykonywanie.

PLAN LABORATORIUM:

1. Wprowadzenie do laboratorium
2. Iloczyn kartezjański
3. Połączenia równościowe
4. Połączenia naturalne
5. Połączenia nierównościowe
6. Połączenia zewnętrzne
7. Połączenia zwrotne
8. Połączenia wielu tabel
9. Stara notacja połączeń

1. Wprowadzenie do laboratorium

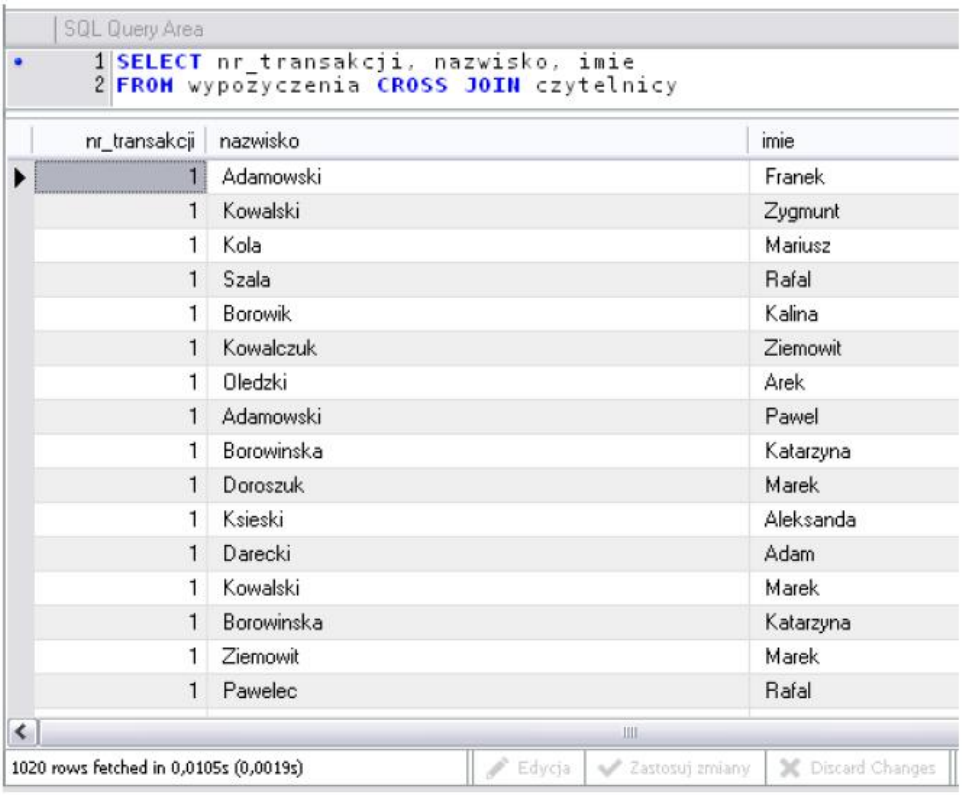
Laboratorium poświęcone jest bardzo ważnemu mechanizmowi wykorzystywanemu przy realizacji zapytań. Jest to mechanizm tzw. „połączeń". Co to są połączenia i jaka jest motywacja stojąca za stworzeniem tego mechanizmu? Przyjrzyjmy się następującemu problemowi. Korzystając z bazy danych poznanej na poprzednich zajęciach, chcemy odnaleźć dla transakcji wypożyczeń nazwisko i imię osoby wypożyczeni. Numery transakcji wypożyczeń są zapisane w osobnej relacji - WYPOZYCZENIA. Pierwszym odruchem przy rozwiązywaniu tego problemu byłoby najpierw odczytać numery wypożyczeń i identyfikatory wypożyczających z relacji WYPOZYCZENIA, a następnie odczytać nazwiska i imiona wypożyczających o odczytanych wcześniej identyfikatorach z relacji CZYTELNICY. Wykorzystując wartości identyfikatorów

wypożyczających w obu relacjach wynikowych można skojarzyć nazwisko z numerem wypożyczenia. Problemy z tym podejściem są dwa. Po pierwsze konieczne jest wykonanie dwóch zapytań, a po drugie należy zaimplementować własnoręcznie połączenie tych informacji. Mechanizm połączeń w języku SQL pozwala uniknąć tych problemów, gdyż pozwala na nakazanie SZBD aby połączył dane z dwóch, lub więcej tabel. Jeżeli zapytanie SQL zostanie odpowiednio skonstruowane, to system zarządzania bazą danych sam dobierze najbardziej wydajny algorytm połączenia danych z kilku tabel, a wyniki zwróci w postaci jednej relacji wynikowej. Istnieje wiele rodzajów połączeń danych z dwóch tabel: iloczyn kartezjański oraz połączenia: naturalne, równościowe, nierównościowe, zewnętrzne i zwrotne. Każdy z tych rodzajów zostanie omówiony na niniejszych ćwiczeniach.

2. Iloczyn kartezjański

Najprostszym typem połączenia jest tzw. „iloczyn kartezjański" (*cross-join*). W wyniku iloczynu kartezjańskiego powstaje relacja, która zawiera wszystkie atrybuty z obu relacji. Krótki w tej relacji powstają jako każda możliwa kombinacja krótki z pierwszej łączonej relacji, z krotką z drugiej łączonej relacji. Jak łatwo zauważyć, liczba krotek w relacji stanowiącej wynik połączenia poprzez iloczyn kartezjański jest równa iloczynowi rozmiarów oryginalnych relacji (o ile nie wprowadzi się dodatkowych warunków selekcji). Wobec olbrzymich rozmiarów, jakie potrafią przyjmować relacje w zastosowaniach praktycznych, w większości wypadków wystąpienie iloczynu kartezjańskiego sygnalizowane jest błędem w zapytaniu. Iloczyn kartezjański w czystej postaci rzadko bywa przydatny.

W języku SQL, według standardu ANSI, sposób połączenia dwóch lub więcej tabel definiowany jest w klauzuli FROM. Połączenie poprzez iloczyn kartezjański definiowane jest za pomocą operatora połączenia CROSS JOIN umieszczanego pomiędzy nazwami łączonych relacji.

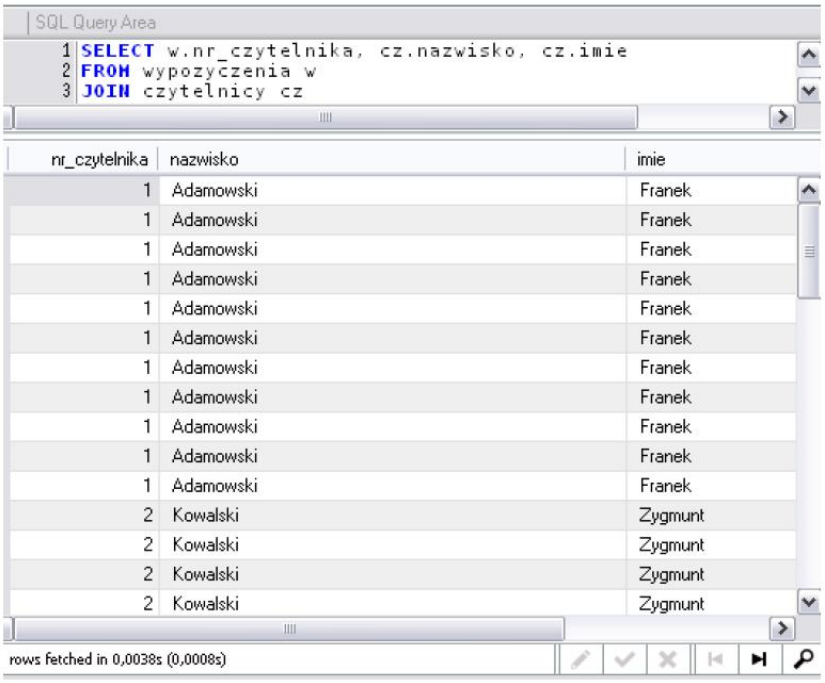


Zapytanie to da nieprawidłowy wynik. Każdemu wypożyczeniu przyporządkuje każdego czytelnika.

3. Połączenia rownościowe

Problem, o którym wspomniano na początku polegający na znalezieniu dla każdego numeru wypożyczenia nazwiska i imienia czytelnika, który wypożyczył daną książkę - można rozwiązać za pomocą tzw. „połączenia rownościowego" (*equi join*). W wyniku połączenia rownościowego powstaje relacja, która zawiera wszystkie atrybuty z obu łączonych relacji, jednak w przeciwieństwie do iloczynu kartezjańskiego, krótki w takiej relacji są konstruowane w inny sposób. Powstają one poprzez znalezienie wszystkich par krotek, z których jedna pochodzi z pierwszej łączonej relacji, a druga z drugiej i spełniają one tzw. „warunek połączenia". Każda taka para jest łączona i tworzy nową krotkę w relacji wynikowej. Ważne jest, aby warunki połączeniowe porównywały jedynie wartości atrybutów pochodzących z łączonych relacji. W połączeniach równościowych warunki te muszą być oparte o operator równości ('='). Podobnie jak w przypadku iloczynów kartezjańskich, połączenie równościowe jest również definiowane w klauzuli FROM.

Zapytanie zwróci numery transakcji i imiona i nazwiska osób, które dokonały wypożyczenia:



Powyższe zapytanie jest rozwiązaniem problemu zdefiniowanego na początku, który polegał na odnalezieniu dla każdej transakcji wypożyczenia , nazwisko i imię czytelnika, który dokonał wypożyczenia. Nazwisko czytelnika można zidentyfikować korzystając z numeru nr_czytelnika, który jest każdemu czytelnikowi przypisany.

4. Polaczenia naturalne

Połączenia naturalne są specjalnym rodzajem połączeń równościowych. Połączenie naturalne dwóch relacji to połączenie równościowe relacji, w którym warunki równości dotyczą wszystkich par atrybutów o takich samych nazwach. Podstawową różnicą, pomiędzy zapytaniami równościowymi, a naturalnymi, jest lista atrybutów relacji powstającej w wyniku połączenia. W wyniku połączenia naturalnego atrybut (albo atrybuty) połączeniowe występują tylko raz, podczas gdy w wyniku połączenia równościowego występują oba atrybuty połączeniowe z obu łączonych relacji. Istnieją dwie notacje dla połączeń naturalnych:

```
SELECT relacjal.atrybut, alias2.atrybut

FROM relacjal [aliasl] NATURAL JOIN relacja2 [alias2]

WHERE ....

ORDER BY (atrybutl,atrybut2,
```

lub:

```
SELECT relacja1.atrybut, alias2.atrybut
FROM relacja1 [alias1] JOIN relacja2 USING [alias2]
WHERE ....
ORDER BY (atrybut1,atrybut2,
```

Różnica pomiędzy tymi notacjami jest taka, że pierwsza notacja automatycznie wymaga, aby wszystkie pary atrybutów o takich samych nazwach w obu łączonych relacjach były równe, a druga pozwala określić, które z par atrybutów, o takich samych nazwach, powinny być równe. W celu lepszej ilustracji działania połączeń naturalnych, na slajdzie przedstawiono trzy równoważne zapytania.

```
SELECT wypozyczenia.nr_czytelnika, nazwisko, imie
FROM wypozyczenia
JOIN czytelnicy
ON wypozyczenia.nr_czytelnika=czytelnicy.nr_czytelnika
```

```
SELECT nr_czytelnika, nazwisko, imie FROM wypozyczenia
NATURAL JOIN czytelnicy
```

```
SELECT nr_czytelnika, nazwisko, imie FROM wypozyczenia
JOIN czytelnicy USING (Nr_czytelnika)
```

Zapytanie (1) jest identyczne z zapytaniem omawianym przy okazji połączeń równościowych. Zapytania (2) i (3) wykorzystują połączenia naturalne do realizacji tego samego zadania, co zapytanie (1). Przeanalizujmy zapytanie (2). Relacje Wypożyczenia i Czytelnicy są łączone (wklauzuli FROM) za pomocą operatora NATURAL JOIN. Ponieważ ta odmiana połączenia naturalnego wymaga, aby wszystkie pary atrybutów o takich samych nazwach były równe, a jedynymi takimi atrybutami w obu tych relacjach są atrybuty o nazwie nr_czytelnika, to relacje zostaną połączone równościowo zgodnie z warunkiem

WYPOZYCZENIA.NR_CZYTELNIKA=CZYTELNICY.NR_CZYTELNIKA.

W zapytaniu (3) użyto drugiej notacji stosowanej w połączeniach naturalnych. Relacje są łączone, tak jak w przypadku połączeń równościowych, za pomocą operatora JOIN. W przeciwieństwie jednak do połączeń równościowych, za nazwą drugiej relacji użyto słowa kluczowego USING, a nie ON, i podano wspólną nazwę atrybutów z obu łączonych relacji, które

mają zostać wykorzystane do połączenia. Podobnie jak poprzednio, warunek użyty do połączenia relacji będzie następujący: WYPOZYCZENIA.NR_CZYTELNIKA=CZYTELNICY.NR_CZYTELNIKA. Jak zatem łatwo zauważyć, połączenia we wszystkich 3 zapytaniach są równoważne. Porównajmy obecnie klauzulę SELECT zapytania (1) z klauzulami SELECT zapytań (2) i (3). Jedyną różnicą pomiędzy tymi klauzulami jest to, iż w zapytaniu (1) nazwę atrybutu nr_czytelnika poprzedzono aliasem relacji WYPOZYCZENIA, podczas gdy w zapytaniach (2) i (3) tego nie zrobiono. Przyczyną nie podania aliasu, bądź nazwy relacji, przed nazwą atrybutu nr czytelnika jest fakt, Że jest to atrybut połączeniowy, a, jak wspominano na początku, atrybuty połączeniowe występują w wyniku połączenia jedynie raz. Ponieważ atrybut nr_czytelnika nie należy już do Żadnej konkretnej relacji nie może być poprzedzany nazwą relacji, bądź jej aliasem.

5. Połączenia nierównościowe

Połączenia nierównościowe są połączeniami, w których warunek połączeniowy nie używa operatora równości, ale dowolny inny operator. Podobnie jak w przypadku połączenia równościowego, w wyniku połączenia nierównościowego powstaje relacja, która zawiera wszystkie atrybuty z obu relacji. Krotki są również tworzone w podobny sposób. Znajdowane są wszystkie pary krotek, z których jedna pochodzi z pierwszej łączonej relacji, a druga z drugiej i spełniają one warunki połączenia. Każda taka para jest łączona i tworzy nową krotkę w relacji powstającej w wyniku połączenia. Ogólna notacja połączeń jest taka sama jak dla połączeń równościowych (zmieniają się tylko warunki połączeniowe):

```
SELECT relacja1.atrybut, alias2.atrybut
FROM relacja1 [alias1] JOIN relacja2 [alias2] ON
warunek_połączenia
WHERE . . . .
ORDER BY ....
```

6. Połączenia zewnętrzne

We wszystkich opisanych dotychczas rodzajach połączeń, w relacji powstającej w wyniku połączenia, znajdują się jedynie krotki, które spełniają warunki połączenia. Taki typ połączeń nazywany jest „połączeniem wewnętrznym" *(inner join).* Istnieją również „połączenia zewnętrzne" *(outer join),* w których można zażądać, aby wszystkie krotki z jednej, albo z obydwu łączonych relacji znalazły się w wyniku połączenia, nawet takie, które nie spełniają warunków połączenia (nie znalazły pary). Aby móc zachować wszystkie krotki z jednej relacji, do drugiej relacji wprowadzana jest „wirtualna" krotka, która wypełniona jest wartościami pustymi. Wszystkie krotki z relacji, które nie mogą znaleźć swojej pary, łączone są z "wirtualną" krotką w drugiej relacji.

Ogólna składnia połączeń zewnętrznych wygląda następująco:

```
SELECT relacja1.atrybut, alias2.atrybut

FROM relacja1 [alias1] [NATURAL] {LEFT|RIGHT|FULL}
[OUTER] JOIN
relacja2 [alias2]
{ON (warunek_połączenia1) | USING (atrybut) | 0}

WHERE ....
ORDER BY ..
```

W ogólności podział typów połączeń ze względu na to które krotki trafiają do relacji wynikowej (wewnętrzne, zewnętrzne - lewostronne, prawostronne, pełne) jest ortogonalny względem podziału połączeń ze względu na warunek połączenia (równościowe, naturalne, nierównościowe). Każdą z kombinacji tych typów połączeń można skonstruować (za wyjątkiem połączenia typu iloczyn kartezjański, który jest zupełnie osobnym typem połączenia).

7. Połączenia zwrotne

„Połączenia zwrotne" *(self join)* są specjalnym przypadkiem połączeń, w których łączymy tabelę z samą sobą. Połączeniem zwrotnym może być dowolny typ połączenia (wewnętrzne, zewnętrzne, równościowe i nierównościowe), za wyjątkiem połączenia naturalnego, co wynika z faktu, że łączenie równościowe relacji z samą sobą według atrybutów o tej samej nazwie nic

nie daje (co najwyżej oryginalną relację). Ogólna składnia połączenia zwrotnego jest taka sama, jak każdego innego typu połączenia omawianego poprzednio. Jedyną różnicą jest tutaj podanie tej samej nazwy relacji po obu stronach operatora definiującego połączenie. Dodatkowo, przy pisaniu zapytań z połączeniem zwrotnym należy pamiętać, Żeby nadać różne aliasy obu wystąpieniom nazwy relacji w zapytaniu. Jest to konieczne aby możliwe było rozróżnienie z którego wystąpienia relacji pochodzi atrybut.

8. Łączenie wielu tabel

Jak wspomniano wcześniej, w wyniku połączenia powstaje relacja, która jest następnie dalej przetwarzana w celu realizacji zapytania (selekcja, projekcja, grupowanie itp.). Ponieważ wynik połączenia jest relacją, to nic nie stoi na przeszkodzie, aby nie można jej było połączyć z kolejną relacją. W ten sposób można wykonywać dowolną liczbę połączeń. Ostateczna składnia polecenia SELECT z uwzględnieniem możliwości definicji dowolnej liczby połączeń wygląda następująco:

```
SELECT relacja1.atrybut, alias2.atrybut .....
FROM relacja
WHERE ....
ORDER BY .......
```

Gdzie „relację" można, w sposób rekursywny, zdefiniować następująco:

```
- nazwa relacji [alias]
- (relacja)
- relacja1 CROSS JOIN relacja2
- relacja1 [NATURAL] [{LEFT|RIGHT|FULL} [OUTER]]

JOIN relacja2 {ON (warunek_połączenia1) | USING (atrybut) | 0}
```

Jak łatwo zauważyć, dla każdego połączenia definiowany jest warunek połączenia (za wyjątkiem iloczynu kartezjańskiego). Ponieważ połączeń jest o jedno mniej niż łączonych relacji, tyle też należy w zapytaniu zdefiniować warunków połączeniowych. Dodatkową ważną uwagą jest to, iż operator połączenia jest łączny lewostronnie, chociaż priorytet połączeń można zmieniać za pomocą nawiasów (stąd nawiasy w rekursywnej definicji przedstawionej powyżej).

9. Stara notacja połączeń

Dotychczas opisano sposób łączenia tabel zdefiniowany w późniejszych wersjach standardu SQL. W starszych wersjach stosowano inny zapis, który teraz zostanie pokrótce przedstawiony. Starsze połączenia były wszystkie definiowane w oparciu o pomysł filtrowania wyniku iloczynu kartezjańskiego za pomocą standardowej klauzuli służącej do selekcji (WHERE). W klauzuli FROM definiowano zatem jedynie iloczyn kartezjański poprzez wymienienie po przecinku wszystkich relacji wchodzących w jego skład. Zapytanie (1) definiuje właśnie iloczyn kartezjanski relacji WYPOZYCZENIA i CZTELNICY. W sytuacji, gdy konieczne było wykonanie połączenia równościowego, z wyniku takiego iloczynu wybierano, za pomocą klauzuli WHERE, jedynie krotki spełniające warunki połączenia (zapytanie (2)). W podobny sposób wykonywano połączenia nierównościowe.

Takie podejście na pierwszy rzut oka wydaje się być bardzo niewydajne, jednak większość SZBD jest w stanie wykryć typ połączenia na podstawie warunków w klauzuli WHERE i zastosować najbardziej wydajny algorytm. Ten sposób łączenia tabel nie uwzględniał połączeń zewnętrznych. Stało się to przyczyną powstania rozwiązań specyficznych dla SZBD, np. umieszczenie w klauzuli WHERE, przy jednym z atrybutów w warunku połączeniowym, operator (+). Znaczenie tego operatora jest następujące: „dla tego połączenia, umieść wirtualną krotkę (krotkę z pustymi wartościami) w relacji, z której pochodzi atrybut, przy którym umieszczono niniejszy operator". W konsekwencji, w wyniku połączenia, wszystkie krotki z drugiej relacji, która uczestniczyła w połączeniu (nie tej przy której umieszczono operator) znajdowały się w wyniku. W związku z tym, wszystkie krotki z relacji CZYTELNICY znajdą się w rozwiązaniu. Niestety, w tej notacji nie jest możliwe zdefiniowanie pełnego połączenia zewnętrznego i jeżeli zachodzi potrzeba wykonania takiego połączenia, to należy zapytanie rozbić na dwa, a wynik połączyć za pomocą operatora UNION. Zapytanie (3) pokazuje sposób wykonania połączenia kilku tabel. Podobnie jak w poprzednich przykładach wykonywany jest tutaj iloczyn kartezjański wszystkich tabel, a następnie, za pomocą warunków umieszczonych w klauzuli WHERE, wybierane są jedynie te krotki, o które chodzi. Stara notacja nie pozwala również na tworzenie połączeń naturalnych.

(1)

```
1 SELECT nazwisko, imie, CONCAT(kod,' ', miasto) adres
2 FROM wypozyczenia ,czytelnicy
```

nazwisko	imie	adres
Adamowski	Franek	21500 Biala Podlaska
Kowalski	Zygmunt	21500 Biala Podlaska
Kola	Mariusz	21500 Biala Podlaska
Szala	Rafal	21500 Biala Podlaska
Borowik	Kalina	21500 Biala Podlaska
Kowalczuk	Ziemowit	21500 Biala Podlaska
Oledzki	Arek	21500 Biala Podlaska
Adamowski	Pawel	19500 Lublin
Borowinska	Katarzyna	19500 Goldap
Doroszuk	Marek	19500 Lublin
Ksieski	Aleksanda	19500 Lublin
Darecki	Adam	21500 Biala Podlaska
Kowalski	Marek	21500 Biala Podlaska
Borowinska	Katarzyna	21500 Biala Podlaska
Ziemowit	Marek	21500 Biala Podlaska
Pawelec	Rafal	21040 Piaski
Ziemowit	Rafal	21048 Lublin

1020 rows fetched in 0,0138s (0,0018s) Edycja Zastosuj zmiany Discard Char

(2)

SQL Query Area

```
1 SELECT nazwisko, imie, CONCAT(kod,' ', miasto) adres
2 FROM wypozyczenia ,czytelnicy
3 WHERE wypozyczenia.nr_czytelnika=czytelnicy.nr_czytelnika
```

nazwisko	imie	adres
Adamowski	Franek	21500 Biala Podlaska
Adamowski	Franek	21500 Biala Podlaska
Adamowski	Franek	21500 Biala Podlaska
Adamowski	Franek	21500 Biala Podlaska
Adamowski	Franek	21500 Biala Podlaska
Adamowski	Franek	21500 Biala Podlaska
Adamowski	Franek	21500 Biala Podlaska
Adamowski	Franek	21500 Biala Podlaska
Adamowski	Franek	21500 Biala Podlaska
Adamowski	Franek	21500 Biala Podlaska
Adamowski	Franek	21500 Biala Podlaska
Kowalski	Zygmunt	21500 Biala Podlaska
Kowalski	Zygmunt	21500 Biala Podlaska
Kowalski	Zygmunt	21500 Biala Podlaska
Kowalski	Zygmunt	21500 Biala Podlaska
Kowalski	Zygmunt	21500 Biala Podlaska

60 rows fetched in 0,0094s (0,0008s) Edycja Zastosuj zmiany Discard Changes

(3)

```
1 SELECT czytelnicy.nazwisko, czytelnicy.imie, tytul,
2        CONCAT(kod,' ', miasto) adres
3 FROM wypozyczenia ,czytelnicy, ksiazki
4 WHERE wypozyczenia.nr_czytelnika=czytelnicy.nr_czytelnika
5 AND wypozyczenia.sygnatura=ksiazki.sygnatura
```

nazwisko	imie	tytul	adres
Adamowski	Franek	Fotografowanie aparatem cyfrowym - samouczek	21500 Biala Podlaska
Kowalski	Zygmunt	Fotografowanie aparatem cyfrowym - samouczek	21500 Biala Podlaska
Adamowski	Franek	Fotografowanie aparatem cyfrowym - samouczek	21500 Biala Podlaska
Kowalski	Zygmunt	Fotografowanie aparatem cyfrowym - samouczek	21500 Biala Podlaska
Kola	Mariusz	Fotografowanie aparatem cyfrowym - samouczek	21500 Biala Podlaska
Adamowski	Franek	Fotografowanie aparatem cyfrowym - samouczek	21500 Biala Podlaska
Kowalski	Zygmunt	Fotografowanie aparatem cyfrowym - samouczek	21500 Biala Podlaska
Adamowski	Franek	Fotografowanie aparatem cyfrowym - samouczek	21500 Biala Podlaska
Kowalski	Zygmunt	Fotografowanie aparatem cyfrowym - samouczek	21500 Biala Podlaska
Kola	Mariusz	Fotografowanie aparatem cyfrowym - samouczek	21500 Biala Podlaska
Szala	Rafal	Fotografowanie aparatem cyfrowym - samouczek	21500 Biala Podlaska
Kola	Mariusz	Fotografowanie aparatem cyfrowym - samouczek	21500 Biala Podlaska
Borowik	Kalina	Strategia bekitnego oceanu	21500 Biala Podlaska

60 rows fetched in 0,0083s (0,0014s) Edycja Zastosuj zmiany Discard Changes Pierwsz

Ćwiczenia do samodzielnej realizacji

Proszę pobrać nowy plik biblioteka.sql i wgrać do bazy.

1. Ile razy każdy z czytelników wypożyczał już książki. Podaj numer czytelnika, nazwisko, imię i wynik zapytania.
2. Ile razy wypożyczana była każda książka? Podaj sygnaturę, książkę i wynik.
3. Wyświetl książki (sygnaturę i tytuł), które wypożyczane były przynajmniej 5 razy
4. Ile jest książek z działów ‘Informatyka” i Literatura”. Wyświetl identyfikator działu, nazwę działu oraz wynik.
5. Wyświetl nazwy stanowisk i informacje ilu pracowników z miasta Lublin, którzy zarabiają powyżej 2000 zł pracuje na danym stanowisku.
6. Wyświetl informacje, ile książek zostało wypożyczonych przez grupy osób zajmujących poszczególne stanowiska, np. Bibliotekarz 53, Kierownik 3.
7. Wyświetl w jednej kolumnie imię, nazwisko, czytelnika i tytuł książki, którą już wypożyczał. Dane posortuj malejąco względem nazwiska. Usuń powtarzające się wyniki
8. Wyświetl w jednej kolumnie imię, nazwisko czytelnika i liczbę ile różnych książek już wypożyczył
9. Jakie książki wypożyczał już czytelnik o id=1 u pracownika o id 1 lub 3. Usuń powtarzające się odpowiedzi.
10. Wyświetl czytelników: imię i nazwisko czytelników - studentów, którzy nie oddali wypożyczonych książek. Dane wyświetl w jednej kolumnie, w nagłówku ma być wpisane „Lista studentów, którzy maja oddać książki przed wakacjami". Dane posortuj alfabetycznie względem nazwiska
11. Podaj dane czytelników, którzy wypożyczali książki od 1 stycznia do 1 lipca 2009 r. Usuń powtarzające się wyniki. Dane posortuj alfabetycznie po nazwisku
12. Wyświetl wynik: ilu czytelników ma nazwisko zaczynające się na literę M lub na literę S? W nagłówku wpisz test: „liczba czytelników"
13. Wyświetl w jednej kolumnie dane o pracownikach biblioteki, którzy nie wypożyczali książek czytelnikom.
14. Wyświetl czytelników, którzy dokonali wypożyczeń 5-03-2010 u pracownika o identyfikatorze 13. Wyniki posortuj względem nazwiska czytelnika malejąco

15. Wyświetl osoby, które wypożyczyły książki w tych samych dniach, kiedy Aleksandra Daniluk. W wyniku nie wyświetlaj już danych Aleksandry Daniluk. Dane posortuj względem identyfikatorów malejąco.
16. Wyświetl 5 ostatnich wyników zapytania: podaj daty wypożyczeń i nazwiska czytelników którzy wypożyczali książki w datach 11-05-2008 - 08-11-2010.