Bazy Danych
MySQL

reaktor

PWN

Agenda

1. Wprowadzenie do baz danych
2. Tworzenie nowych baz danych
3. Tworzenie, modyfikacja i usuwanie tabel
4. Dodawanie, modyfikacja i usuwanie rekordów w tabelach
5. Wykonywanie zapytań w bazie danych
6. Wbudowane funkcje MySQL (matematyczne, daty i czasu, tekstowe)
7. Instrukcje warunkowe
8. Filtrowanie wyników
9. Sortowanie wyników
10. Łączenie wielu tabel
11. Agregacja wyników
12. Podzapytania
13. Tworzenie perspektyw
14. Tworzenie wyzwalaczy
15. Projektowanie baz danych

Wprowadzenie

- **Informacja** to wiedza dotycząca obiektów takich jak fakty, zdarzenia, przedmioty, procesy, idee, zawierająca koncepcje, która w określonym kontekście ma określone znaczenie.
- **Dane** to reprezentacja informacji, mająca interpretację, właściwą do komunikowania się, właściwą do przetwarzania.
- **Przetwarzanie danych** to automatyczne wykonywanie operacji na danych.
- **Operacje na danych** to operacje matematyczne, logiczne, sortowanie, kompilowanie, operacje tekstowe, łączenie, zestawianie, wyszukiwanie, drukowanie, redagowanie.

PRZETWARZANIE DANYCH ↔ PRZETWARZANIE INFORMACJI

Czym jest baza danych?

- Baza danych to kolekcja wzajemnie powiązanych danych przechowywana w pamięciach dyskowych i udostępniania jej użytkownikom na określonych zasadach.
- Bazą danych jest na przykład
  - System plików na komputerze.
  - książka adresowa w programie pocztowym.
  - Bazy danych używane są w bankach i większych przedsiębiorstwach do przechowywania informacji o kontach czy też danych personalnych.

reaktor

PWN

Język baz danych

- Język SQL (*Structured Query Language*) służy do manipulowania danymi umieszczonymi w relacyjnych bazach danych.
- Jest językiem uniwersalnym, dzięki czemu praca na różnych systemach baz danych sprowadza się do wydawania tych samych lub podobnych komend tzw. zapytań SQL.
- Język SQL został zaimplementowany w większości relacyjnych systemów baz danych takich jak: DB2, Oracle, InterBase, MySQL, dBase, Paradox.

reaktor

PWN

Składnia SQL

- Język definiowania struktur danych - DDL (Data Definition Language) - jest wykorzystywany do wszelkiego rodzaju operacji na tabelach, takich jak: tworzenie, modyfikacja oraz usuwanie
- Język do wybierania i manipulowania danymi - DML (Data Manipulation Language) - służy do manipulowania danymi umieszczonymi w tabelach, pozwala na wstawienie danych, ich prezentację, modyfikowanie oraz usuwanie
- Język do zapewnienia bezpieczeństwa dostępu do danych - DCL (Data Control Language) - jest używany głównie przez administratorów systemu baz danych do nadawania odpowiednich uprawnień do korzystania z bazy danych.

reaktor

PWN

Relacyjne bazy danych

- Relacyjny system baz danych przechowuje wszystkie dane w tabelach.
- Tabela zawiera dane na konkretny temat, np. dane o klientach, pracownikach, towarach itp.
- System bazy danych zarządza tymi informacjami, pozwala m.in. na szybsze ich wyszukiwanie i zorganizowanie.
- Za każdym razem, kiedy potrzebujemy informacji z bazy danych, musimy "zapytać" system bazy danych w zrozumiałym dla niego języku. Tym językiem jest SQL.

reaktor

PWN

Czym jest MySQL

- MySQL jest najpopularniejszym darmowym systemem obsługi baz danych rozpowszechnianym na zasadach licencji GPL (*General Public License*).
- Jego nowatorska budowa pozwoliła na stworzenie niezwykle szybkiego i niezawodnego serwera obsługującego bazy danych.

reaktor

PWN

Jak się połączyć z MySQL?

- Aby połączyć się z serwerem baz danych potrzebujemy specjalnego programu tzw. klienta lub języka skryptowego (umieszczanego na serwerach WWW), który posiada wbudowaną obsługę baz danych.
- Bardzo dobrym narzędziem przydatnym podczas nauki SQL, jest panel administracyjny do baz danych – MySQL Workbench.

reaktor

PWN

MySQL Workbench

- MySQL Workbench to narzędzie do zarządzania i modelowania baz danych MySQL.
- Za jego pomocą można:
  - edytować konfigurację serwera i jego komponentów,
  - zaprojektować i stworzyć schematy (wizualne reprezentacje tabel, widoków itp.) nowych baz danych,
  - wykonać dokumentację istniejących oraz zapewnić wsparcie przy procesach migracji do MySQL.
- W MySQL Workbench znajdziemy więc funkcjonalność między innymi MySQL Administrator i MySQL Query Browser.
- Najlepsze narzędzie dla administratora, jak i programisty MySQL, pozycja obowiązkowa dla każdego środowiska wyposażonego w serwer MySQL.

reaktor

PWN

Cechy MySQL Workbench

- Silnik graficzny OpenGL,
- Wsparcie dla procesów reverse-engineeringu i synchronizacji baz danych,
- Możliwość generowania skryptów SQL,
- Przeglądowy tryb pracy w którym cały model bazy jest prezentowany w jednym przekrojowym widoku,
- Wsparcie dla projektowania baz na poziomach koncepcyjnym, logicznym i fizycznym,
- Rozszerzalna architektura, eksport modelu jako skryptu SQL typu CREATE, import i eksport modeli DBDesigner4,
- Wizualna reprezentacja tabel, widoków, procedur wbudowanych oraz funkcji oraz pełne wsparcie dla możliwości MySQL 5.

Tworzenie nowych baz danych

Tworzenie baz danych

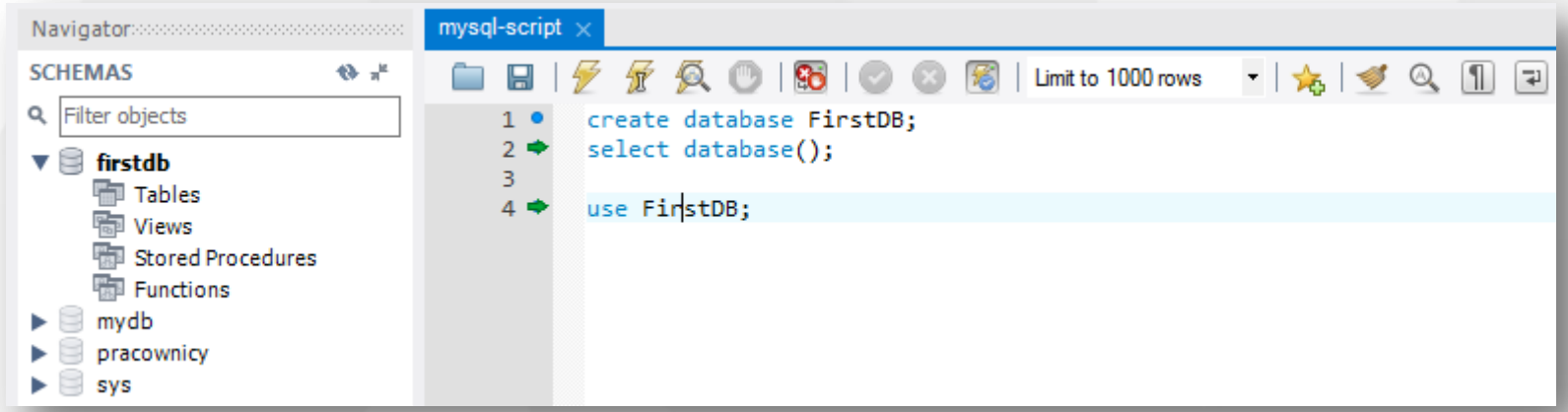
- Poleceniu utworzenia nowej bazy danych:

```
CREATE DATABASE DBname;
```

Ustawienie domyślnej bazy danych

- Procedura ustawienia domyślnej bazy danych, dzięki której wszystkie procedury będą odwoływać się do ustawionej bazy danych:

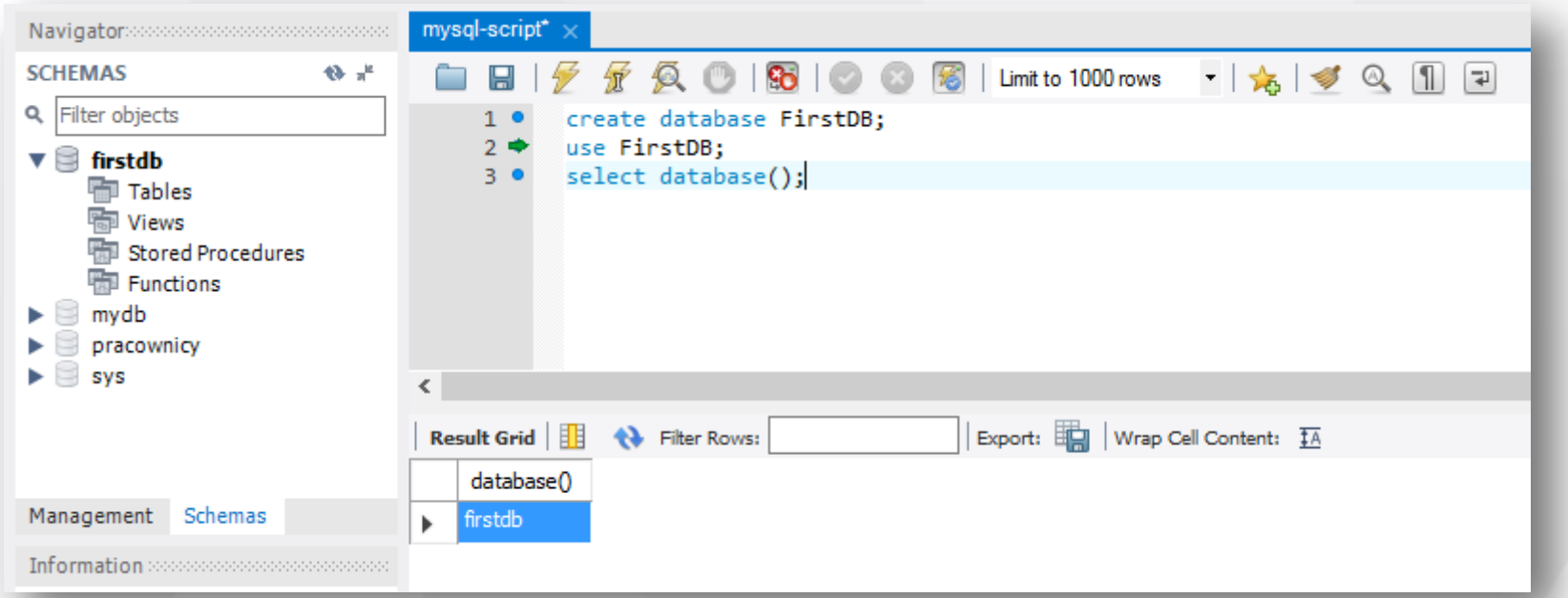
```
USE nazwaBD;
```



Wyświetlanie aktywnej bazy danych

- Polecenie wyświetlenia aktywnej bazy danych:

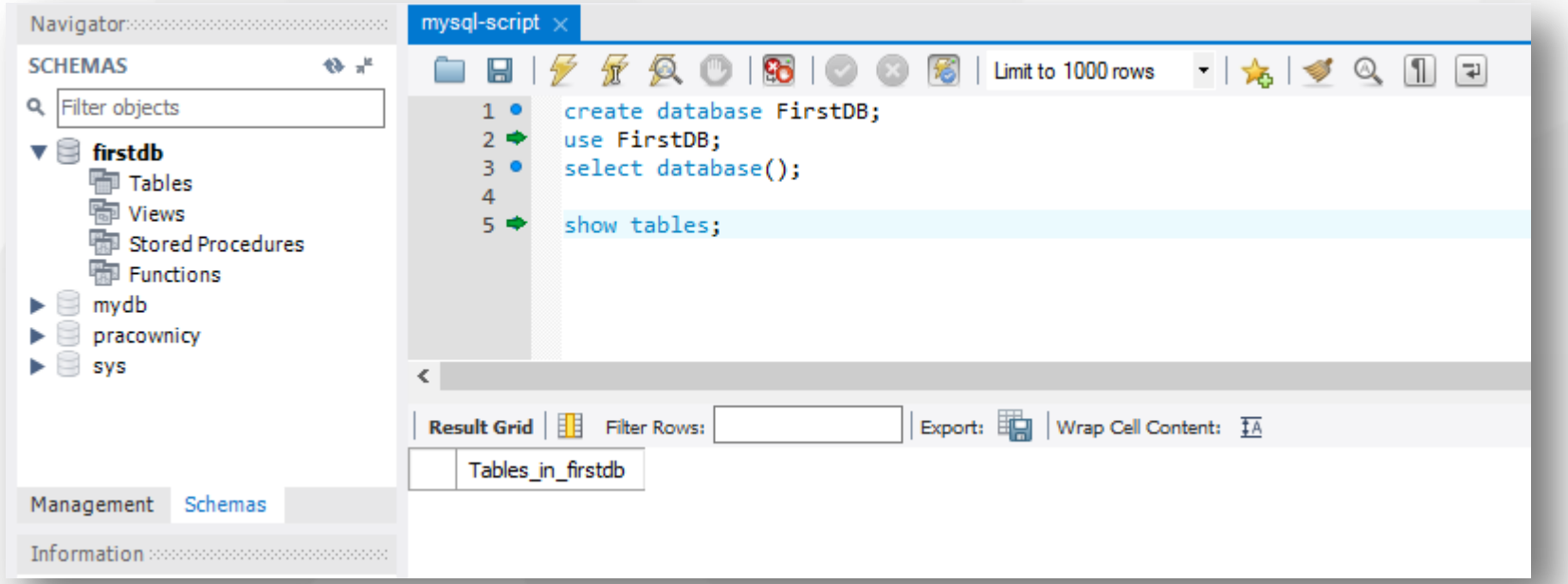
```
SELECT DATABASE();
```



Wyświetlanie tabel w bazie danych

- Polecenie wyświetlenia tabel w bazie danych:

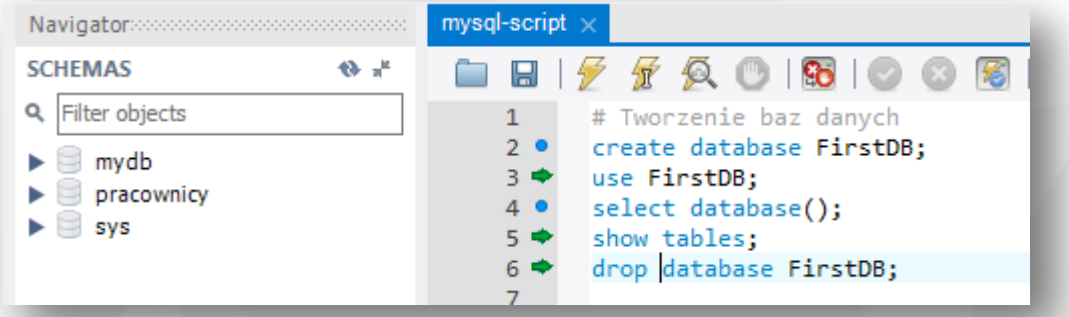
```
SHOW TABLES;
```



Usuwanie bazy danych

- Ostatnią komendą związaną z bazami danych jest usunięcie bazy danych:

```
DROP DATABASE DBname;
```



Tabele w bazie danych

Tworzenie nowej tabeli

- Procedura utworzenia nowej tabeli:

```
CREATE TABLE nazwa_tablicy (
nazwa_kolumny1 typ_danych[rozmiar] typ_warunku [warunek] klauzule,
nazwa_kolumny1 typ_danych[rozmiar] typ_warunku [warunek] klauzule,
…
nazwa_kolumny1 typ_danych[rozmiar] typ_warunku [warunek] klauzule
);
```

Typy danych

- Stałoprzecinkowe:

TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT

- Zmienno przecinkowe:

FLOAT, DOUBLE

- Znakowe:

CHAR, VARCHAR

- Tekstowe:

TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT

- Data i czas:

DATE, TIME, DATETIME, YEAR

Typy warunków

- Klucz główny - powoduje, że dane w kolumnie nie mogą sie powtarzać, służy do identyfikacji rekordu.

PRIMARY KEY

- Klucz obcy - odwołanie do klucza głównego z innej tabeli.

FOREIGN KEY

- Niepowtarzalny – dane nie mogą przyjmować tych samych wartości:

UNIQUE

- Wartości dodatnie - powoduje, że kolumna nie może przechowywać wartości na minusie przy czym zakres pozostaje taki sam, działa tylko dla typów przechowujących liczby całkowite

UNSIGNED

- Wypełnienie zerem - czyli zerowe wypełnienie, w przypadku gdy ilość liczb w polu będzie mniejsza niż ta zadeklarowana przy tworzeniu kolumny wartość pola będzie automatycznie "dopełniana" zerami na początku, działa tylko dla typów przechowujących liczby całkowite, automatycznie tworzy atrybut UNSIGNED

ZEROFILL

Klauzule

- Automatyczne zwiększanie wartości:

AUTO_INCREMENT

- Dopuszczanie tylko wartości niezerowych:

NOT NULL

- Przypisanie wartości domyślnej:

DEFAULT wartość

Utworzenie przykładowej tabeli

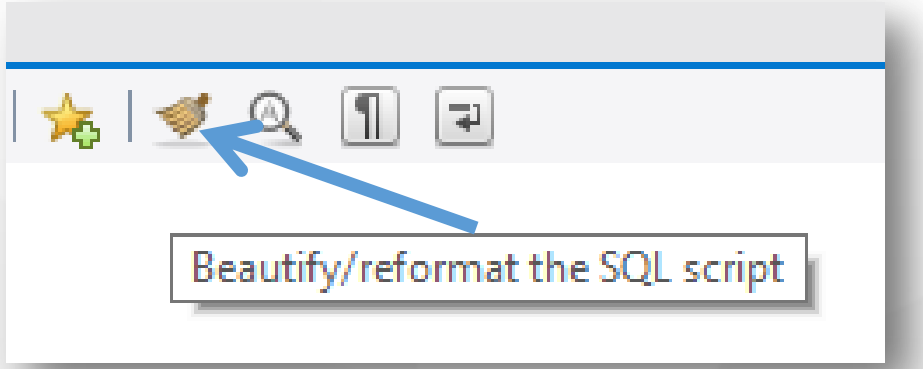
```
create table Uczestnicy(

id int primary key auto_increment,

imie varchar(15) not null,

nazwisko varchar(25) not null

);
```



Wyświetlanie atrybutów kolumn tabeli

Polecenie wyświetlania atrybutów kolumn w tabeli:

```
DESCRIBE TBLname;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
imie	varchar(15)	NO		NULL	
nazwisko	varchar(25)	NO		NULL	

Usuwanie tabel

- Na koniec polecenie usuwania tabeli z bazy danych:

```
DROP TABLE TBLname;
```

Modyfikowanie tabel w bazie danych

Dodawanie klucza głównego do tabeli

- Dodanie klucza głównego do istniejącej tablicy:

```
ALTER TABLE TBLname
ADD PRIMARY KEY (CLMNname);
```

- Dodanie złożonego klucza głównego do istniejącej tablicy:

```
ALTER TABLE TBLname
ADD CONSTRAINT nazwa_klucza_głównego
PRIMARY KEY (CLMNname1, CLMNname2);
```

Zmiana nazw

- Zmiana nazwy tabeli:

```
ALTER TABLE 'stara_nazwa_tablicy'
RENAME AS 'nowa_nazwa_tablicy';
```

- Zmiana nazwy kolumny:

```
ALTER TABLE 'nazwa_tablicy'
CHANGE 'nazwa_kolumny' 'nowa_nazwa_kolumny' 'typ_danych‘ [klauzule];
```

- Zmiana typu danych kolumny:

```
ALTER TABLE 'nazwa_tablicy'
MODIFY 'nazwa_kolumny' 'typ_danych';
```

Dodawanie nowej kolumny do tabeli

- Dodanie kolumny do tabeli:

```
ALTER TABLE 'nazwa_tablicy'

ADD 'nazwa_kolumny' 'typ_danych';
```

reaktor

Ćwiczenie MS1

- Utwórz nową bazę danych o dowolnej nazwie.

reaktor

Ćwiczenie MS2

- Usuń nowo utworzoną bazę danych.

reaktor

Ćwiczenie MS3

- ❑ Utwórz jeszcze raz nową bazę danych o nazwie Skoczkowie.
- ❑ Ustaw nowo utworzoną bazę danych jako domyślną.

reaktor

Ćwiczenie MS4

- Dodaj do bazy tabelę o nazwie skocznie zawierającą następujące dane:

```
id_skoczni          integer,
miasto              text,
kraj_s              text,
nazwa               text,
k                   integer,
sedz                integer
```

reaktor

Ćwiczenie MS5

- Dodaj do bazy tabelę o nazwie trenerzy zawierającą następujące dane:

```
kraj        text,
imie_t      text,
nazwisko_t  text,
data_ur_t   date
```

reaktor

Ćwiczenie MS6

- Dodaj do bazy tabelę o nazwie zawodnicy zawierającą następujące dane:

```
id_skoczka    integer,
imie          text,
nazwisko      text,
kraj          varchar(3),
data_ur       date,
wzrost        integer,
waga          integer
```

reaktor

Ćwiczenie MS7

- Dodaj do bazy tabelę o nazwie zawody zawierającą następujące dane:

```
id_zawodow          integer,

id_skoczni          integer,

data                date
```

reaktor

Ćwiczenie MS8

- Stwórz tabelę o nazwie *kibice* zawierającą kolumny: imie_k, nazwisko_k, data_ur_k, kraj, wzrost.

reaktor

Ćwiczenie MS9

- Stwórz tabelę *składki* z czterema kolumnami: *id_skladki, kwota_skladki, data_skladki, id_kibica*.
- Z tabelki *kibice* usuń kolumnę *kraj*.
- Do tabelki *kibice* dodaj kolumnę *pesel* .
- Do tabelki kibice dodaj ponownie kolumnę *kraj*.

Dodawania rekordów w bazie danych

Dodawanie rekordów do tabeli

- Polecenie wpisania danych do tabeli:

```
INSERT INTO TBLname (kolumna1, …, kolumnaN)

VALUES (wartość1, …, wartośćN);
```

Pobieranie danych z pliku

- Polecenie importu danych z pliku:

```
LOAD DATA LOCAL INFILE ”plik” INTO TABLE TBLname;

plik = adres_bezpośredni/nazwa_pliku.rozszerzenie_pliku
```

UWAGA1! Kolumny dla tego polecenia powinny być oddzielone tabulatorami a wiersze znakami końca wiersza.

UWAGA2! Adres skopiowany z właściwości pliku zawiera „\” należy je zamienić na „/”.

Modyfikacja rekordów

- Polecenie modyfikacji zapisanych rekordów w bazie danych:

```
UPDATE 'nazwa_tabeli'
SET kolumna1 = wyrażenie1, …, kolumnaN = wyrażenieN
[WHERE warunek];
```

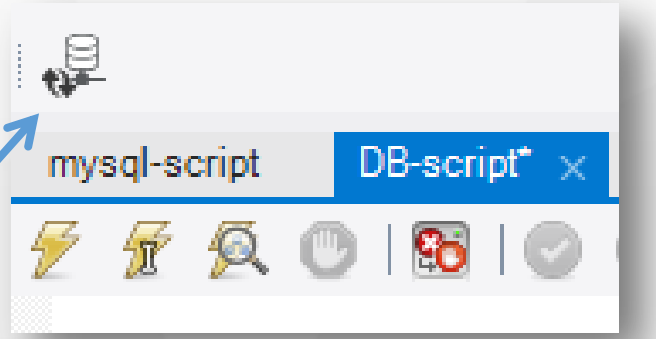
Usuwanie rekordów z bazy danych

- Polecenie usunięcia rekordu/rekordów pod warunkiem:

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

- Polecenie usunięcia wszystkich rekordów z bazy danych:

```
DELETE FROM TBLname;
```



- Uwaga na uprawnienia! Edit->Preferences->SQL Editor->Odznacz „Safe Updates”. Następnie zsynchronizuj zmianę z serwerem.

reaktor

Ćwiczenie MS10

- Uzupełnij tabelę skocznie danymi.

reaktor

Ćwiczenie MS11

- Uzupełnij tabelę trenerzy danymi.

reaktor

Ćwiczenie MS12

- Uzupełnij tabelę zawodnicy danymi.

reaktor

Ćwiczenie MS13

- Uzupełnij tabelę zawody danymi.

reaktor

Ćwiczenie MS14

- Do tabeli *kibice* wpisz trzech kibiców:

Jan Kowalski z Polski,

John Smith, wzrost 172 cm

Anna Zawadzka, ur. 23.12.1977

reaktor

Ćwiczenie MS15

- Z tabeli kibice usuń Johna Smitha.

reaktor

Ćwiczenie MS16

- Zmień datę urodzenia kibica Jana Kowalskiego na 8 grudnia 1974.

reaktor

Ćwiczenie MS17

- Zmień na Krzysztof imię kibica pochodzącego z tego kraju, z którym związany jest trener Kuttin.

reaktor

Ćwiczenie MS18

- Zwiększ o 2 cm wzrost wszystkich zawodników.
- Następnie zmniejsz o 2 cm wzrost wszystkich zawodników.

Tworzenie zapytań w bazie danych

Najprostsze zapytania

- Polecenie do wybierania danych, które mają być wyświetlone, eksportowane lub przetwarzane.:
- Konstrukcje SELECT są zwyczajowo nazywane zapytaniami lub kwerendami.

```
SELECT 6;

SELECT 3,4,5;
```

Jak się wykonuje zapytanie?

```
SELECT 3, 4, 5;
```

- Wykonywanie tego zapytania odbywało się następująco:
  - Zostaje stworzona tabelka wynikowa.
  - W niej zostają utworzone trzy kolumny, bo poprosiliśmy o trzy wartości.
  - Zostaje utworzony jeden wiersz.
  - W kolejne pola tego wiersza zostają wstawione wartości 3, 4 i 5.

Proste zapytania

- Polecenie prostego zapytania wyszukującego wszystkie rekordy w tabeli:

```
SELECT * FROM TBLname
```

- Polecenie prostego zapytania wyszukującego wszystkie dane z określonych kolumn tabeli:

```
SELECT CLMNname1,…,CLMNnameN FROM TBLname
```

Proste zapytania

- Polecenie prostego zapytania wyszukującego dane z określonych kolumn tabeli, ale tylko te które spełniają określony warunek:

```
SELECT CLMNname1,…,CLMNnameN FROM TBLname [WHERE warunek]
```

Jak się wykonuje zapytanie w tabeli

```
SELECT imie, nazwisko from zawodnicy;
```

- Proste zapytanie odbywa się następująco:
  - Zostaje stworzona tabela wynikowa,
  - W niej zostają stworzone dwie kolumny, bo poprosiliśmy o dwie wartości,
  - Następnie dla każdego wiersza tabeli zawodnicy zostają wykonane poniższe kroki:
    - w tabeli wynikowej zostaje stworzony wiersz,
    - w pierwszym polu tego wiersza zostaje umieszczona wartość wyrażenia imię wyliczona w kontekście aktualnego wiersza,
    - w drugim polu tego wiersza zostaje umieszczona wartość wyrażenia nazwisko wyliczona w kontekście aktualnego wiersza,

Nieco bardziej złożone zapytanie

- Przeanalizujemy jak odbywa się następujące zapytanie:

```
SELECT imię, nazwisko, wzrost+3

FROM zawodnicy;
```

Usuwanie powtórzeń w zapytaniach

- Aby usunąć wszystkie powtórzenia należy użyć klauzuli DISTINCT:

```
SELECT DISTINCT CLMNname1,  ..., CLMNnameN
FROM TBLname;
```

Nazywanie (alias) kolumn wynikowych

- Jeżeli chcemy nazwać kolumnę, która powstała jako wynik wyrażenia musimy posłużyć się konstrukcją AS nazwa:

```
SELECT CLMNname1 AS newCLMNname1,…, CLMNnameN AS newCLMNnameN
FROM TBLname;
```

Wartość NULL

- Teoria baz danych obejmuje sytuację, w której dana komórka tabeli jest pusta, nie ma żadnej wartości - NULL.
- Nie jest to ani łańcuch pusty, ani liczba 0.
- Istnienie NULL na tyle komplikuje bazy danych, że w systemy zarządzania bazami danych zawsze w specjalny sposób traktują te osobliwą wartość.
- Oprócz podstawowego działania każdy operator oddzielnie obsługuje NULL. Zwykle oznacza to, że wystąpienie NULL w podwyrażeniu powoduje, że całe wyrażenie staje się NULL!

reaktor

Ćwiczenie MS19

- Wypisz zawartość tabeli *zawodnicy*.
- Wypisz zawartość tabeli *trenerzy*.

reaktor

Ćwiczenie MS20

- Z tabeli *zawodnicy* wypisz kolumny *kraj*, *imie*, *nazwisko*.

reaktor

Ćwiczenie MS21

- Narty skoczka narciarskiego nie mogą być dłuższe niż 146% jego wzrostu.
- Wypisz listę skoczków z ich maksymalną długością nart.

reaktor

Ćwiczenie MS22

- Wskaźnik BMI (*Body Mass Index*) to iloraz masy człowieka i kwadratu jego wzrostu wyrażonego w metrach.
- Wypisz listę skoczków z ich wskaźnikiem BMI.

Funkcje i operatory matematyczne

Podstawowe operatory arytmetyczne

- Podstawowe operatory arytmetyczne:
  - Dodawanie (+)
  - Odejmowanie (-)
  - Mnożenie (*)
  - Dzielenie (/)

```
SELECT 2+4;

SELECT 15*3;
```

Operatory arytmetyczne

- MOD() lub % – reszta z dzielenia

```
SELECT 8 MOD 3 -- 2
SELECT 8 % 3 -- 2
```

- PI() – funkcja zwraca wartość Pi

```
SELECT PI(); -- 3.141593
```

- POW() lub POWER() – zwraca wartość podanej liczby podniesionej do określonej potęgi. W poniższym przypadku “2" podniesione do potęgi “8" daje “256".

```
SELECT POW(2,8); -- 256
SELECT POWER(2,8); -- 256
```

Operatory logiczne

- AND - operator logiczny ‘i’ wymagający spełnienia obu warunków.
- OR - operator logiczny ‘lub’ wymagający spełnienia przynajmniej jednego z warunków.
- IN - operator sprawdzający przynależność elementu do zadanego zbioru. Elementy zbioru podaje się w nawiasach okrągłych i wymienia po przecinkach.
- BETWEEN - operator sprawdzający przynależność elementu do danego przedziału
- NOT - operator zaprzeczenia. Można go łączyć z innymi operatorami np. NOT BETWEEN czy NOT IN

Więcej operatorów w dokumentacji:

https://dev.mysql.com/doc/refman/5.7/en/comparison-operators.html

Operatory porównania

- > Większe niż.
- >= Większe lub równe (uwaga na kolejność!).
- < Mniejsze niż.
- <= Mniejsze lub równe.
- = Równe.
- <> lub != Różne.

Wykrywanie NULL

- Jakiekolwiek porównania z NULL dają NULL. Do wykrywania NULL służą dwie konstrukcje:
  - wyrażenie IS NULL,
  - wyrażenie IS NOT NULL

Funkcje matematyczne

- ABS() – funkcja zwracająca wartość bezwzględną podanej liczby

```
SELECT ABS(-56); -- 56
```

- ACOS() – funkcja zwraca wartość arcus cosinus dla podanego argumentu

```
SELECT ACOS(0.32); -- 1.2450668395002664
```

- ASIN() – funkcja zwraca wartość arcus sinus dla podanego argumentu

```
SELECT ASIN(0.32); -- 0.3257294872946302
```

- ATAN() – funkcja zwraca wartość arcus tangens dla podanego argumentu bądź argumentów

```
SELECT ATAN(0.32); -- 0.3097029445424562
```

Funkcje matematyczne

- CEIL() – zaokrąglenie w górę do pełnej wartości całkowitej

```
SELECT CEIL(0.32); -- 1
```

- CONV (liczba, aktualny system, konwertowany system) – konwersja wartości pomiędzy podanymi systemami.

```
SELECT conv(9,10,2); -- 1001
```

- SIN() - zwraca wartość funkcji sinus dla podanej wartości kąta w radianach

```
SELECT SIN(90); -- 0.8939966636005579
```

- COS() – zwraca wartość funkcji cosinus dla podanej wartości kąta w radianach

```
SELECT COS(60); -- -0.9524129804151563
```

Funkcje matematyczne

- TAN() – zwraca wartość funkcji tangens dla podanej wartości kąta w radianach

```
SELECT TAN(45); -- 1.6197751905438615
```

- COT() – zwraca wartość funkcji cosinus dla podanej wartości kąta w radianach

```
SELECT COT(45); -- 0.617369623783551
```

- DEGREES() – konwersja radianów na stopnie

```
SELECT DEGREES(1.5707963267948966); -- 90
```

Funkcje matematyczne

- RADIANS() – konwersja stopni na radiany

```
SELECT RADIANS(90)          -- 1.5707963267948966
```

- DIV() – operator dzielenia (wynikiem dzielenia jest wartość całkowita)

```
SELECT 8 div 3              -- 2
```

- EXP() – funkcja zwracająca wartość “e”, odwrotność funkcji LOG()/LN()

```
SELECT EXP(1);              -- 2.718281828459045
```

- FLOOR() – zaokrąglenie w dół do pełnej wartości całkowitej

```
SELECT FLOOR(4.32);         -- 4
```

Funkcje matematyczne

- LN() – logarytm z podanej liczby

```
SELECT LN(8)                    -- 2.079441541679835
```

- LOG() – funkcja wyznaczająca logarytm dla podanych argumentów, czyli do jakiej liczby podnieść “2" aby otrzymać “8"

```
SELECT LOG(2,8)                 -- 3
```

- LOG10() – funkcja wyznaczająca logarytm dziesiętny dla podanego argumentu, czyli do jakiej liczby podnieść “10" aby otrzymać “100"

```
SELECT LOG10(100)               -- 2
```

- LOG2() - funkcja wyznaczająca logarytm dwójkowy dla podanego argumentu, czyli do jakiej liczby podnieść “2" aby otrzymać “8"

```
SELECT LOG2(8)                  -- 3
```

Funkcje matematyczne

- RAND() – funkcja generująca wartość losową

```
SELECT rand();                    -- 0.9159928401149743
```

- ROUND() – zaokrąglenie podanej wartości do określonej liczby miejsc po przecinku. Bez parametru wartość zaokrąglana jest do liczby całkowitej.

```
SELECT ROUND(123.6543);           -- 124
```

- SIGN() – funkcja zwracająca znak dla podanej wartości. Gdy wartość jest mniejsza od zera zwraca “-1", gdy zero zwraca “0", gdy większa od zera zwraca “1"

```
SELECT SIGN(-1212);               -- -1
```

Funkcje matematyczne

- SQRT() – zwraca pierwiastek kwadratowy z podanej liczby

```
SELECT SQRT(9); -- 3
```

- TRUNCATE() – funkcja “obcinająca” części dziesiętne (bez zaokrąglania), gdy podany parametr jest większy od “0", lub “zerująca” części całkowite gdy parametr mniejszy od “0"

```
SELECT TRUNCATE(1.263,2); -- 1.26
```

Funkcje daty i czasu

Aktualna data i czas

- CURDATE() – funkcja zwracająca aktualną datę w formacie YYYY-MM-DD, np: 2016-04-05

```
SELECT CURDATE();
```

- UTC_DATE() – funkcja zwracająca aktualną datę UTC w formacie YYYY-MM-DD, np: 2016-04-05

```
SELECT UTC_DATE();
```

- CURTIME() – funkcja zwracająca aktualny czas w formacie HH:MM:SS, np: 09:06:34

```
SELECT CURTIME();
```

- UTC_TIME() – funkcja zwracająca aktualny czas UTC (-2 godziny)w formacie HH:MM:SS, np: 07:06:34

```
SELECT UTC_TIME();
```

Aktualna data i czas

- NOW() – funkcja zwraca bieżącą datę oraz godzinę, np: 2016-04-05 09:06:34

```
SELECT NOW();
```

- SYSDATE() – funkcja zwraca bieżącą datę oraz godzinę systemową, np: 2016-04-05 09:06:34

```
SELECT SYSDATE();
```

- UNIX_TIMESTAMP() – funkcja zwraca bieżącą datę oraz godzinę systemową, np: 2016-04-05 09:06:34

```
SELECT UNIX_TIMESTAMP();
```

- UTC_TIMESTAMP() – funkcja zwraca bieżącą datę UTC oraz godzinę UTC (-2 godziny), np: 2016-04-05 07:06:34

```
SELECT UTC_TIMESTAMP();
```

Obcinanie daty

- YEAR() – funkcja zwraca rok dla podanej daty, np: 2016

```
SELECT YEAR('2016-04-05');
```

- MONTH() – funkcja zwraca miesiąc dla podanej daty, np: 4

```
SELECT MONTH('2016-04-05');
```

- DAY() – funkcja zwraca miesiąc dla podanej daty, np: 5

```
SELECT DAY('2016-04-05');
```

Obcinanie godziny

- HOUR() – funkcja zwraca godzinę od podanego czasu, np: 9

```
SELECT HOUR('09:06:34');
```

- MINUTE() – funkcja zwraca minuty od podanego czasu, np: 6

```
SELECT MINUTE('09:06:34');
```

- SECOND() – funkcja zwraca sekundy od podanego czasu, np: 34

```
SELECT SECOND('09:06:34');
```

Operacje na datach i czasie

- DATE_FORMAT() – funkcja formatuje podaną datę do określonego formatu

```
SELECT DATE_FORMAT('2016-04-05', '%d.%m.%Y');
```

- QUARTER() – funkcja zwraca numer kwartału dla podanej daty, np: 2

```
SELECT QUARTER('2016-04-05');
```

- DAYOFYEAR() – funkcja zwraca dzień w roku, np. dzień 2016-04-05 to 96 dzień w roku 2016

```
SELECT DAYOFYEAR('2016-04-05');
```

- DAYOFWEEK() – funkcja zwraca numer dnia tygodnia, np: 3

```
SELECT DAYOFWEEK('2016-04-05');
```

Operacje na datach i czasie

- DAYOFMONTH() – funkcja zwraca numer dnia miesiąca, np: 5

```
SELECT DAYOFMONTH('2016-04-05');
```

- MONTHNAME() – funkcja zwraca nazwę miesiąca dla podanej daty, np: April

```
SELECT MONTHNAME('2016-04-05');
```

- DAYNAME() – funkcja zwraca nazwę dnia tygodnia dla podanej daty, np: Tuesday

```
SELECT DAYNAME('2016-04-05');
```

- WEEK() – funkcja zwraca numer tygodnia dla podanej daty, np: 14

```
SELECT WEEK('2016-04-05');
```

Operacje na datach i czasie

- WEEKOFYEAR() – funkcja zwraca numer dnia tygodnia dla podanej daty, np: 14

```
SELECT WEEKOFYEAR('2016-04-05');
```

- WEEKDAY() – funkcja zwraca numer dnia tygodnia, np: 1 gdzie: 0 – poniedziałek, 1 – wtorek, 2 – środa, 3 – czwartek 4 – piątek, 5 – sobota, 6 – niedziela

```
SELECT WEEKDAY('2016-04-05');
```

- LAST_DAY – funkcja zwraca ostatni dzień miesiąca dla podanej daty, np: 2016-04-30

```
SELECT LAST_DAY('2016-04-05');
```

- TO_DAYS() – funkcja zwraca ilość dni od roku “0", (0000-01-01), np: 736424

```
SELECT TO_DAYS('2016-04-05');
```

Operacje na datach i czasie

- FROM_DAYS() – funkcja zwraca datę na podstawie podanej ilości dni licząc od roku “0"

```
SELECT FROM_DAYS(736424);
```

- DATEDIFF() – funkcja zwraca różnicę dni między dwoma datami, (liczba dni od 2001-01-01 do 2016-04-05)

```
SELECT DATEDIFF('2016-04-05','2000-01-01');
```

- TIME_TO_SEC() – liczba sekund, która upłynęła od określonej godziny, np: 32794

```
SELECT TIME_TO_SEC('09:06:34');
```

- ADDDATE(), DATE_ADD() – dodawanie określonej liczby, dni, miesięcy, lat do określonej daty, np: 2016-04-14

```
SELECT ADDDATE('2016-04-04', INTERVAL 10 DAY);
```

```
SELECT DATE_ADD('2016-04-04', INTERVAL 10 DAY);
```

Operacje na datach i czasie

- SUBDATE(), DATE_SUB() – odejmowanie określonej liczby, dni, miesięcy, lat od określonej daty, np: 2016-03-25

```
SELECT DATE_SUB('2016-04-04', INTERVAL 10 DAY);
```

- ADDTIME() – dodawanie daty i czasu do podanego w parametrze czasu/daty, np: 2016-04-06 10:17:45

```
SELECT ADDTIME('2016-04-05 09:06:34','1 1:11:11');
```

Powyższe zapytanie zwróci nam zatem wynik: 10:17:45

- GET_FORMAT() – funkcja zwracająca format daty/czasu, np: %d.%m.%Y

```
SELECT GET_FORMAT(DATE,'EUR');
```

Formatowanie daty i czasu

- Formatowanie daty do określonego formatu

```
SELECT DATE_FORMAT('2016-04-05', GET_FORMAT(DATE,'EUR'));
```

- Formatowanie bieżącej daty do określonego formatu

```
SELECT DATE_FORMAT(CURDATE(),'%D %M %Y');
```

Funkcje tekstowe

Dane w tabelach

- Każda kolumna w bazodanowej tabelce ma określony typ danych, które możemy w niej przechowywać.
- Kolumny mogą być na przykład typu integer (przechowujemy liczby całkowite), date (przechowujemy daty) czy text (przechowujemy tekst).
- Kiedy pobieramy z bazy wartość typu date, to w zasadzie serwer przesyła klientowi datę jako taką i dopiero klient zajmuje się wyświetleniem jej w taki lub inny sposób (miesiąc cyfrą rzymską lub arabską, rok na końcu lub początku).
- Nie możemy więc zapytaniem SQL-owym określić, czy chcemy otrzymać datę sformatowaną po amerykańsku czy po europejsku. Możemy natomiast użyć w wyrażeniu funkcji, która zamieni datę (lub liczbę) na napis (typ text) formatując ją po drodze w odpowiedni sposób.

Funkcje tekstowe

- Zmiana wielkości liter:
  - LOWER() / LCASE() – zmiana na małe litery

```
SELECT LOWER(CLMNname) FROM TBLname;
```

  - UPPER() / UCASE() – zmiana na wielkie litery

```
SELECT UPPER(CLMNname) FROM TBLname;
```

Funkcje tekstowe

- Obliczanie długości ciągu:
  - BIT_LENGTH() – zwraca długość ciągu w bitach

```
SELECT BIT_LENGTH(CLMNname);
```

  - LENGTH() / CHAR_LENGTH() / CHARACTER_LENGTH() / OCTET_LENGTH() – zwraca długość ciągu podanego w argumencie

```
SELECT LENGTH(CLMNname);
```

Funkcje tekstowe

- Usuwanie pustych znaków:
  - TRIM() – usuwa puste znaki (spacje) na początku i końcu podanego ciągu

```
SELECT TRIM(kolumna);
```

  - LTRIM() / RTRIM() – usuwa puste znaki (spacje) na początku / końcu podanego ciągu

```
SELECT LTRIM(kolumna);
SELECT RTRIM(kolumna);
```

Funkcje tekstowe

- Łączenie ciągów:
  - CONCAT() – łączenie zawartości kolumn jak i dowolnego tekstu.

```
SELECT CONCAT(CLMNname1, CLMNname2)
FROM TBLname;
```

  - CONCAT_WS() – łączenie ciągów z zastosowaniem określonego separatora.

```
SELECT CONCAT_WS(separator, CLMNname1, CLMNname2)
FROM TBLname;
```

  - LPAD() / RPAD() – uzupełnienie ciągu z lewej / prawej strony o określony ciąg do określonej długości.

```
SELECT LPAD(CLMNname, długość max, wypełnienie);
SELECT RPAD(CLMNname, długość max, wypełnienie);
```

Funkcje tekstowe

- Łączenie ciągów:

  - **SPACE()** – funkcja wstawia określoną ilość spacji.

```
SELECT SPACE(liczba spacji);
```

  - **INSERT()** – funkcja wstawia do określonego ciągu zdefiniowany w funkcji ciąg począwszy od podanej pozycji zamieniając określoną ilość znaków.

```
SELECT INSERT(ciąg bazowy, pozycja początkowa podstawienia do
    cb, liczba znaków usuwanych z cb, ciąg do podstawienia);
```

  - **REPEAT()** – funkcja powtarza podany w parametrze ciąg określoną ilość razy.

```
SELECT REPEAT(ciąg bazowy, liczba powtórzeń cb);
```

Wycinanie ciągów i zastępowanie znaków

- **REPLACE()** – funkcja zamienia w podanym ciągu wskazany ciąg znaków na inny określony ciąg znaków.

```
SELECT REPLACE(ciąg bazowy, co zmieniamy, na co zamieniamy);
```

- **LEFT() / RIGHT()** – funkcja zwraca określoną ilość znaków licząc od lewej / prawej strony dla podanego ciągu.

```
SELECT LEFT(ciąg bazowy,  liczba znaków do wyświetlenia);
SELECT RIGHT(ciąg bazowy,  liczba znaków do wyświetlenia);
```

- **MID()** – funkcja wycina z podanego ciągu określoną liczbę znaków.

```
SELECT MID(ciąg bazowy, pozycja start, zakres);
```

- **SUBSTR()** lub **SUBSTRING()** – funkcja wycina z podanego ciągu określoną liczbę znaków zaczynając od znaku określonego parametrem.

```
SELECT SUBSTR(ciąg bazowy, pozycja start, zakres);
```

Funkcje porównujące

- LIKE() – funkcja porównująca dwa argumenty, w przypadku gdy są takie same zwraca “1", gdy różne zwraca “0"

```
SELECT 'A' like 'A'; -- 1

SELECT 'A' like 'B'; -- 0
```

- NOT LIKE() – funkcja porównująca dwa argumenty, w przypadku gdy są takie same zwraca “0", gdy różne zwraca “1"

```
SELECT 'A' NOT like 'A'; -- 0

SELECT 'A' NOT like 'B'; -- 1
```

- STRCMP() – funkcja porównuje dwa ciągi i zwraca: “0" – jeżeli oba ciągi są takie same, “-1" – gdy pierwszy ciąg jest mniejszy niż drugi, “1" – w pozostałych przypadkach

```
SELECT STRCMP('Wyraz','Wyraz'); -- 0

SELECT STRCMP('Wyraz','Inny wyraz'); -- 1
```

Funkcje porównujące

- REGEXP() lub RLIKE() – funkcja pozwalająca sprawdzić czy podany ciąg odpowiada zdefiniowanemu wzorcowi. W przypadku gdy porównanie da wynik pozytywny wtedy funkcja zwróci “1", w przeciwnym przypadku “0".

```
SELECT '60-300' REGEXP '[0-9][0-9]-[0-9][0-9][0-9]'; -- 1

SELECT '60-3XA' REGEXP '[A-Z][0-9]-[0-9][0-9][0-9]'; -- 0
```

- NOT REGEXP() lub NOT RLIKE() – funkcja pozwalająca sprawdzić czy podany ciąg nie pasuje do zdefiniowanego wzorca. W przypadku gdy porównanie da wynik negatywny wtedy funkcja zwróci “1", w przeciwnym przypadku “0".

```
SELECT '60-300' NOT REGEXP '[0-9][0-9]-[0-9][0-9][0-9]'; -- 1

SELECT '60-3XA' NOT REGEXP '[0-9][0-9]-[0-9][0-9][0-9]'; -- 0
```

Funkcje wyszukujące

- ELT() – funkcja zwraca wskazany argument podanego ciągu. W tym przypadku zwróci 3 argument, czyli ciąg: “Trzeci”

```
SELECT ELT(3, 'Pierwszy', 'Drugi', 'Trzeci'); -- Trzeci
```

- FIELD() – funkcja zwraca pozycję wystąpienia danego ciągu w podanych ciągach. Zatem podany ciąg ‘Drugi’ zostanie znaleziony w zbiorze wartości ‘Pierwszy’, ‘Drugi’, ‘Trzeci’ na drugim miejscu

```
SELECT FIELD('Drugi', 'Pierwszy', 'Drugi', 'Trzeci'); -- 2
```

- FIND_IN_SET() – funkcja zwraca pozycję wystąpienia danego wyrazu w podanym ciągu

```
SELECT FIND_IN_SET('Drugi', 'Pierwszy,Drugi,Trzeci'); -- 2
```

Funkcje wyszukujące

- INSTR() – funkcja wyszukuje w podanym ciągu określonego ciągu podając pozycję na której on występuje po raz pierwszy.

```
SELECT INSTR('ToJestPrzykladoweZdanie', 'rz'); -- 8
```

- LOCATE() lub POSITION() – funkcja zwraca miejsce wystąpienia określonego ciągu w danym ciągu.

```
SELECT LOCATE('je', 'To jest zdanie normalne'); -- 4
```

Funkcje wyszukujące

- MAKE_SET() – funkcja zwraca ciąg określony “bitowo” przez pierwszy parametr, wyjaśnienie poniższego przykładu:

```
0 - 000 - wynik:
1 - 100 - wynik: pierwszy
2 - 010 - wynik: drugi
3 - 110 - wynik: pierwszy, drugi
4 - 001 - wynik: trzeci
5 - 101 - wynik: pierwszy, trzeci
6 - 011 - wynik: drugi, trzeci
7 - 111 - wynik: pierwszy, drugi, trzeci
```

```
SELECT MAKE_SET(5,'pierwszy','drugi','trzeci'); -- pierwszy,trzeci
```

Funkcje wyszukujące

- EXPORT_SET() – funkcja zwracająca ciąg tekstowy według podanego wzorca. Dla poniższego przykładu będzie to ciąg składający się z “10" znaków rozdzielonych między sobą znakiem “|” gdzie na czwartej pozycji wystąpi “X” zaś pozostałe znaki będą “o”. Wartość “8" podana jako parametr oznacza 4 bit.

```
SELECT EXPORT_SET(8,'X','o','|',10); -- o|o|o|X|o|o|o|o|o|o
```

Zatem patrząc “od tyłu” 2^0 2^1 2^2 2^3 – 0001 oznacza binarnie “8". Inny przykład:

```
SELECT EXPORT_SET(17,'X','o','|',10); -- X|o|o|o|X|o|o|o|o|o
```

Zatem patrząc “od tyłu” 2^0*1 2^1*0 2^2*0 2^3*0 2^4*1 – 10001 oznacza binarnie 17"

Formatowanie ciągów liter

- FORMAT() – funkcja zwraca podaną wartość w określonym formacje, w poniższym przykładzie zaokrągloną do 4 miejsc po “kropce”. Warto zwrócić uwagę na separator tysięczny, którym jest znak “przecinka”

```
SELECT FORMAT(1234.123456, 4); -- 1,234.1235
```

- QUOTE() – funkcja wstawiająca podany ciąg w cudzysłów

```
SELECT QUOTE('Teskt w cudzyslowiu'); -- 'Teskt w cudzyslowiu'
```

- REVERSE() – funkcja odwraca “odbija” podany ciąg (kolejność liter)

```
SELECT REVERSE('To jest zdanie normalne'); -- enlamron einadz tsej oT
```

Formatowanie ciągów liter

- CHAR() – podaje znak dla określonego kodu ASCII

```
SELECT CHAR(68); -- D
```

- ASCII() lub ORD() – podaje kod ASCII dla określonego znaku (funkcja odwrotna do powyższej)

```
SELECT ASCII('D'); -- 68
```

- BIN() – funkcja zwraca binarną wartość dla podanej liczby

```
SELECT BIN(17); -- 10001
```

Formatowanie ciągów liter

- HEX() – zamienia podany ciąg/liczbę na postać heksadecymalną (szesnastkową)

```
SELECT HEX('Tekst'); -- 54656B7374
```

```
SELECT HEX(46548); -- B5D4
```

- UNHEX() – zamienia podaną wartość heksadecymalną (szesnastkową) na ciąg alfanumeryczny

```
SELECT UNHEX('54656B7374'); -- Tekst
```

reaktor

Ćwiczenie MS23

- Obok imion i nazwisk skoczków wypisz ich daty urodzenia w formacie typowym dla języka polskiego, czyli np. "*07.02.2006 r.*"
- Jak wyżej, tylko datę wypisz w postaci "*038 dnia 2006 roku*".

reaktor

Ćwiczenie MS24

- Obok imion i nazwisk skoczków wypisz ich BMI z dokładnością do 2 i 3 miejsc po przecinku.

reaktor

Ćwiczenie MS25

- Zakładając, że narty mogą mieć jedynie długość wyrażoną całkowitą ilością centymetrów, podaj dla każdego zawodnika maksymalną długość jego nart.

reaktor

Ćwiczenie MS26

- Wypisując imiona i nazwiska zamień wielkość liter w nazwiskach, tak by tylko pierwsza litera była wielka.

reaktor

Ćwiczenie MS27

- Wypisz listę zawodników w formacie *imię nazwisko (kraj)*, np. "*Adam Małysz (POL)*".

reaktor

Ćwiczenie MS28

- Obok imienia i nazwiska każdego zawodnika wypisz jego wiek z dokładnością do dnia i z dokładnością do lat.

reaktor

Ćwiczenie MS29

- FIS dba, aby skoczkowie narciarscy nie byli zbyt szczupli i wymaga, aby ich BMI wynosiło co najmniej 20.
- Wypisz listę zawodników wraz z informacją czy mają odpowiednią wagę w stosunku do swojego wzrostu (informacja powinna być osobnym polem o wartości typu boolean).

reaktor

Ćwiczenie MS30

- Wypisz listę trenerów, wraz z informacją czy dla danego trenera znana jest data jego urodzenia (informacja powinna być osobnym polem o wartości typu boolean).

Warunkowe wybieranie wierszy

Wybieranie wyników

- Omówiona dotychczas postać polecenia SELECT pozwalała manipulować jedynie wyświetlanymi kolumnami.
- Aby wpływać na to, które wiersze będą wyświetlane, należy posłużyć się kolejnym cechą zapytań.
- Polecenie SELECT można rozszerzyć o klauzulę WHERE. Zapytanie ma wtedy postać:

```
SELECT CLMNname1, …, CLMNnameN
FROM TBLname WHERE warunek;
```

Wyrażenia warunkowe

- Standard SQL definiuje wyrażenie warunkowe w postaci:

```
CASE  WHEN warunek THEN wynik
      [WHEN warunek_1 THEN wynik_1...]
      [ELSE wynik_n]
END
```

Wyrażenia warunkowe

- Zamiast rozbudowanej konstrukcji CASE można skorzystać z funkcji COALESCE(...) zwracającej wartość swojego pierwszego argumentu, który nie jest NULL.
- Jeśli argument jest null to wyrażenie zwraca kolejny argument (w poniższym przykładzie napis o treści ‘brak napisu’).
- Należy zwrócić uwagę, aby wszystkie argumenty były tego samego typu!

```
..., COALESCE (napis,'brak napisu'), ...
```

reaktor

Ćwiczenie MS31

- Wypisz listę skoczków wraz z informacją o ich wadze. Jeżeli ich BMI jest za niskie obok skoczka powinna pojawić się informacja "*zawodnik za lekki*", w przeciwnym przypadku - "*waga zawodnika w normie*".

reaktor

Ćwiczenie MS32

- Wypisz listę trenerów. Jeżeli data urodzenia trenera nie jest znana, zamiast niej powinna pojawić się informacja "*brak danych*".

Filtrowanie wyników

Wypisywanie części wierszy wyniku

- Klauzula LIMIT ilość umieszczana w końcowej części zapytania (po ORDER BY) służy do ograniczania ilości wypisywanych wierszy.
- Parametr ilość określa maksymalną liczbę zwróconych wierszy.

```
SELECT CLMNname1, …, CLMNnameN
FROM TBLname
WHERE warunki
ORDER BY CLMNname1, …, CLMNnameM
LIMIT ilość;
```

Zastosowanie ograniczania ilości wyników

```
SELECT imie, nazwisko

FROM zawodnicy

WHERE kraj != 'POL'

ORDER BY data_ur

LIMIT 4;
```

Pomijanie części wierszy wyniku

- Ostatnią omawianą klauzulą, która może wystąpić w poleceniu SELECT jest OFFSET ilość, która powoduje pominięcie wskazanej ilości pierwszych wierszy. Dopiero po ich pominięciu odliczana jest ilość podana w LIMIT.
- Oczywiście tak samo jak w poprzednim przypadku, zastosowanie OFFSET ma sens jedynie na posortowanym wyniku.

```
SELECT wyrażenia
FROM tabele
WHERE warunki
ORDER BY wyrażenia
LIMIT ilość
OFFSET ilość;
```

Zastosowanie pominięcia wyników

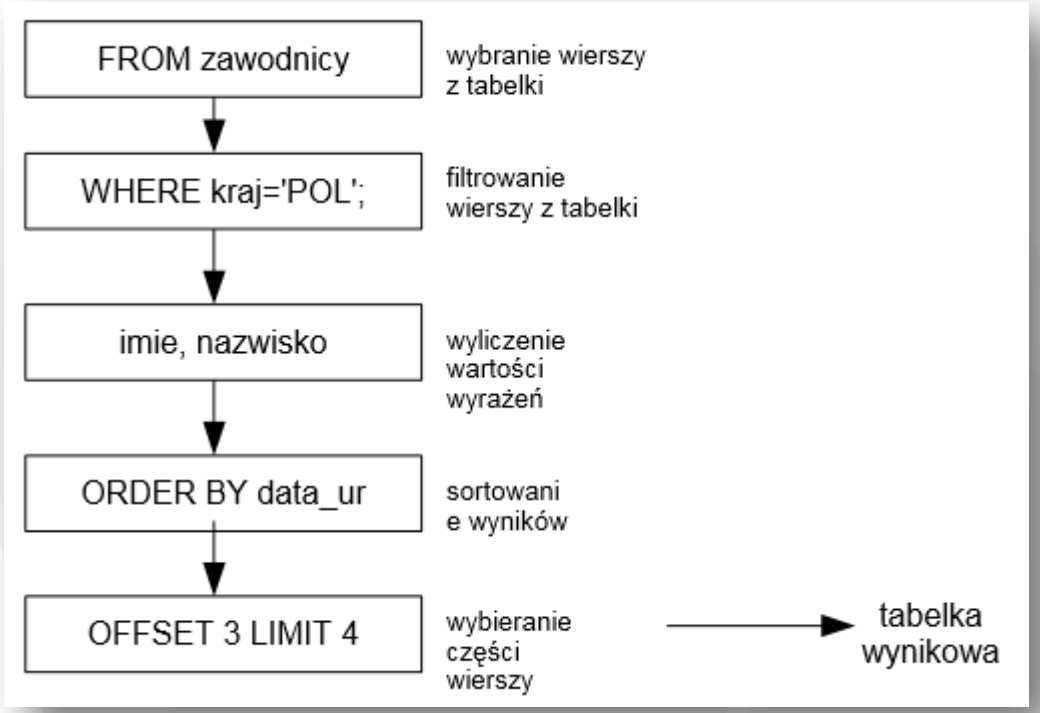
```
SELECT imie, nazwisko

FROM zawodnicy

WHERE kraj != 'POL'

ORDER BY data_ur

LIMIT 4;
```



Sortowanie wyników

Porządek wypisywania danych

- Poleceniem SELECT można wypisać dane w zadanej kolejności.
- Traktowanie relacji jako tabeli może zawieść w momencie modyfikacji, ale także i zwykłego zapytania – wiersze baz danych nie są uporządkowane.
- Do wymuszenia uporządkowania wyniku zapytania służy konstrukcja ORDER BY. Polecenie SELECT ma wtedy postać:

```
SELECT CLMNname1, …, CLMNnameN

FROM tabela

WHERE warunek ORDER BY CLMNname1, …, CLMNnameM;
```

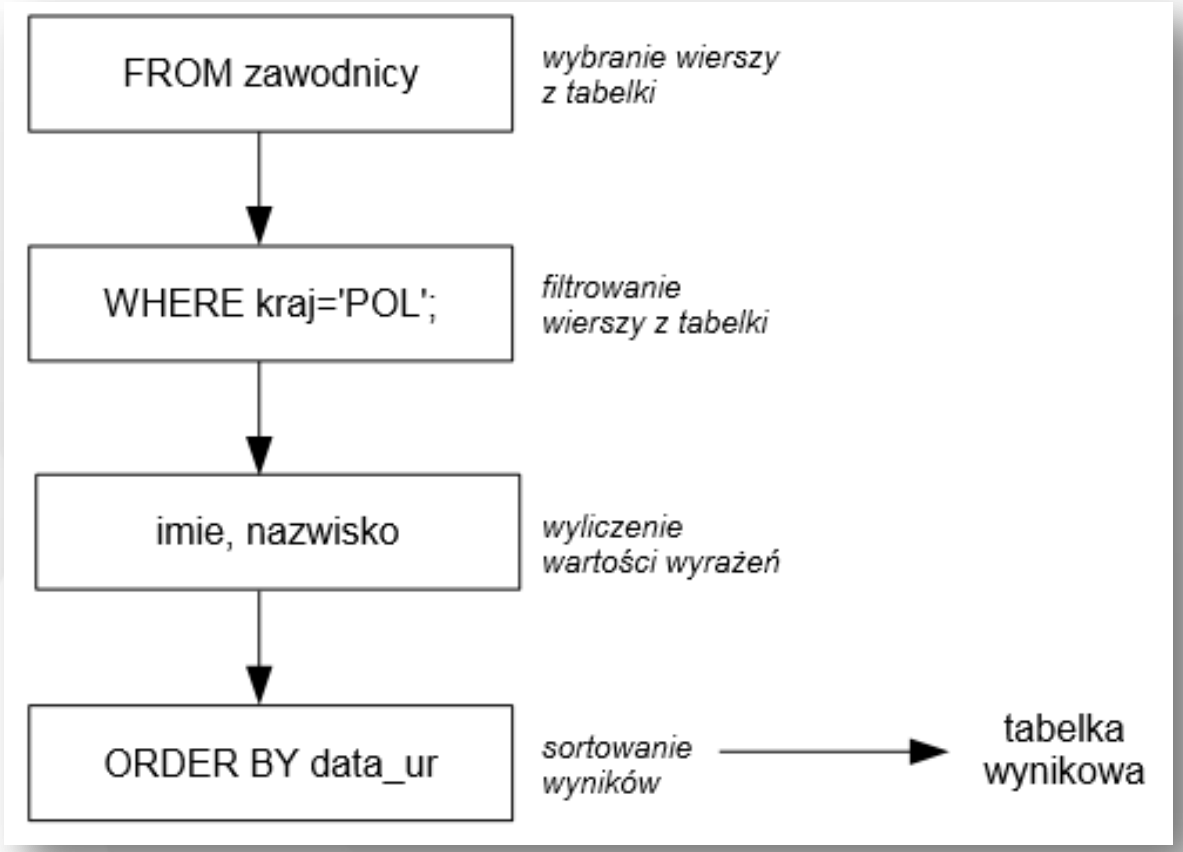
Zastosowanie sortowania wyników

```
SELECT imie, nazwisko

FROM zawodnicy

WHERE kraj != 'POL'

ORDER BY data_ur;
```



reaktor

Ćwiczenie MS33

- Wypisz listę wszystkich polskich zawodników.

reaktor

Ćwiczenie MS34

- Wypisz listę wszystkich zawodników z krajów niemieckojęzycznych.

reaktor

Ćwiczenie MS35

- Wypisz listę wszystkich zawodników o zbyt małej masie ciała.

reaktor

Ćwiczenie MS36

- Wypisz listę zawodników starszych niż 40 lat.

reaktor

Ćwiczenie MS37

- Wypisz listę trenerów bez podanej daty urodzenia.

reaktor

Ćwiczenie MS38

- Wypisz listę zawodników urodzonych w sezonie od listopada do marca.

reaktor

Ćwiczenie MS39

- Wypisz listę zawodników od najniższego do najwyższego.
- Co się stanie, jeśli dwóch zawodników ma ten sam wzrost? Który będzie pierwszy na liście?

reaktor

Ćwiczenie MS40

- Wypisz zawodników od najniższego do najwyższego. Jeśli dwu zawodników ma ten sam wzrost, niech będą oni posortowani po nazwisku.

reaktor

Ćwiczenie MS41

- Wypisz zawodników posortowanych według imienia i nazwiska. Kto byłby pierwszy na takiej liście – Andrzej Zieliński czy Zenon Andrzejewski?

reaktor

Ćwiczenie MS42

- Wypisz listę zawodników zebranych w ekipy narodowe.

reaktor

Ćwiczenie MS43

- Wypisz listę zawodników od najmłodszego do najstarszego.

reaktor

Ćwiczenie MS44

- Wypisz alfabetyczną listę zawodników w formie "*imię nazwisko (KRAJ)*".

reaktor

Ćwiczenie MS45

- Wypisz listę trenerów uporządkowaną według daty urodzenia.

reaktor

Ćwiczenie MS46

- Wypisz listę trenerów uporządkowaną według daty urodzenia – od najstarszych do najmłodszych.
- Ale ci, którzy nie mają podanej daty urodzenia, niech będą na początku.

reaktor

Ćwiczenie MS47

- Wypisz zawodników posortowanych według BMI.

reaktor

Ćwiczenie MS48

- Wypisz listę wszystkich zawodników podzieloną na dwie części: tych o odpowiedniej wadze i tych zbyt lekkich.

Ćwiczenie MS49

- W ramach każdej z tych części zawodnicy niech będą ułożeni alfabetycznie według nazwisk.

reaktor

Ćwiczenie MS50

- Wypisz listę zawodników w kolejności losowej.

reaktor

Ćwiczenie MS51

- Wskaż zawodnika drugiego co do wzrostu.
- Czy jest tylko jeden taki zawodnik?

Operacje teoriomnogościowe

reaktor

PWN

Czym są operacje teoriomnogościowe?

- Wszystkie relacje, jak i wyniki zapytań są zbiorami.
- Fakt ten objawia się m. in. nieokreślonym porządkiem przy wykonywaniu zapytań.
- Jego efektem ubocznym jest to, że wyniki zapytań podlegają dokładnie tym samym operacjom, co zbiory.
- Do tych operacji należy:
  - suma,
  - różnica - nie jest wspierana w MySQL
  - produkt (część wspólna) - nie jest wspierana w MySQL

Suma zbiorów

- Konstrukcja UNION odpowiada sumie relacji. Przyjmuje ona jako dwa parametry dwa zapytania (jedno z lewej, drugie z prawej strony), których wyniki zostaną zsumowane.

```
SELECT wszystkie dotychczas omówione klauzule...

UNION

SELECT wszystkie dotychczas omówione klauzule...;
```

reaktor

PWN

Prosty przykład sumowania zbiorów

```
SELECT 1, 2

UNION

SELECT 3, 4;
```

reaktor

PWN

Operacje teoriomnogościowe

- Zapytania powinny w wyniku dawać tabele o jednakowej ilości kolumn tych samych typów.
- Jeżeli po UNION nastąpi słówko ALL zostaną zwrócone wszystkie krotki obu relacji.
- W przeciwnym przypadku w wyniku nie pojawią się duplikaty – każda krotka będzie unikalna.

reaktor

Ćwiczenie MS52

- Wypisz wspólną listę zawodników i trenerów wraz z informacją o kraju.

reaktor

Ćwiczenie MS53

- Do poprzedniego zapytania dodaj kolumnę *rola*, w której przy zawodnikach będzie słowo *zawodnik*, a przy trenerach – *trener*.

Łączenie wielu tabel

reaktor

PWN

Łączenie tabel

- Dotychczasowe zapytania ograniczały się do jednej tabeli.
- Siłą języka SQL i systemów zarządzania bazami danych jest możliwość wyciągania danych pochodzących z różnych tabel i łącznego wypisywania ich w jednej tabeli wynikowej.
- Aby wypisać dane z kilku tabel w klauzuli FROM należy podać listę tych tabel.
- Dla każdej z tabel można podać alias tak, jak w przypadku kolumn i wyrażeń występujących bezpośrednio po poleceniu SELECT.
- Słowo AS w tym przypadku jest opcjonalne.

Łączenie tabel – iloczyn kartezjański

- Tak skonstruowane zapytanie zwróci iloczyn kartezjański dwóch (trzech, czterech, n) tabel. Iloczyn kartezjański jest zbiorem wszystkich możliwych par (trójek, czwórek, n-tek) krotek z obu tabel i nie jest na ogół pożądanym wynikiem

```
SELECT CLMNnameN AS nazwa_kN, CLMNnameM AS nazwa_kM, ...
FROM TBLname1, TBLname2, ...
WHERE warunek
ORDER BY ...;
```

reaktor

PWN

Łączenie tabel

- Język SQL umożliwia dokonywanie złączeń w bardziej naturalny sposób, tzn. nie przez warunki w klauzuli WHERE, ale poprzez zaznaczenie faktu złączenia w klauzuli FROM.
- JOIN działa bardzo podobnie do tego łączenia dwu tabel, które właśnie robiliśmy. Każdy wiersz pierwszej tabeli jest łączony z każdym wierszem tabeli drugiej.
- Następnie wybierane są tylko te wiersze, które spełniają pewien warunek. Jak określać ten warunek – zaraz się nauczymy.

Łączenie tabel

```
TBLname1 [NATURAL] rodzaj_złączenia TBLname2 [warunek złączenia]
```

- W miejscu rodzaju_złączenia może się pojawić:
  - JOIN oznaczający pełny iloczyn kartezjański;
  - LEFT JOIN oznaczający złączenie, w którym zostaną wykorzystane wszystkie wiersze tabeli_1 (z lewej strony); jeżeli nie będzie odpowiedniego do złączenia wiersza w tabeli_2 pola odpowiadające kolumnom tej tabeli zostaną wypełnione NULLami;
  - RIGHT JOIN – jw. tylko, że zostaną wykorzystane wszystkie wiersze prawej tabeli;

Łączenie tabel

- Warunek złączenia można określić na trzy sposoby:
  - umieszczając słówko NATURAL przed rodzajem złączenia – warunkiem złączenia będzie równość wartości na atrybutach o tych samych nazwach,
  - umieszczając konstrukcję ON warunki po nazwie drugiej tabeli, gdzie warunki są tym co musiałoby się znaleźć w odpowiedniej klauzuli WHERE i takie jest ich znaczenie,
  - umieszczając konstrukcję USING (kolumna_1, kolumna_2...) po nazwie drugiej tabeli – warunkiem złączenia będzie równość wartości na wskazanych kolumnach (wszystkie kolumny muszą wystąpić w obu tabelach).

Schemat łączenia tabel

reaktor

Ćwiczenie MS54

- Wypisz imiona i nazwiska zawodników wraz ich trenerami (na podstawie kraju).

reaktor

Ćwiczenie MS55

- Wypisz listę zawodów wraz z nazwami skoczni i miastami, w których się mieszczą.

reaktor

Ćwiczenie MS56

- Wypisz imiona i nazwiska zawodników wraz ich trenerami (na podstawie kraju).
- Jeśli jakiś zawodnik nie ma trenera, niech nie będzie go na tej liście.

reaktor

Ćwiczenie MS57

- Przerób zapytanie z poprzedniego punktu tak, żeby na liście byli również ci zawodnicy, którzy nie mają trenerów.

reaktor

Ćwiczenie MS58

- Wypisz imiona i nazwiska trenerów z imionami i nazwiskami ich zawodników.
- Jeśli jakiś trener nie trenuje żadnego zawodnika, niech też będzie na tej liście.

reaktor

Ćwiczenie MS59

- Znajdź trenerów, którzy nie trenują żadnych zawodników.

reaktor

Ćwiczenie MS60

- Znajdź trenerów, którzy trenują jakichś zawodników.

reaktor

Ćwiczenie MS61

- Znajdź zawodników, którzy nie mają trenera.

reaktor

Ćwiczenie MS62

- Wypisz listę zawodów wraz z nazwami skoczni i miastami, w których się mieszczą.

reaktor

Ćwiczenie MS63

- Dla każdego zawodnika pokaż wszystkie zawody, które odbywają się w jego kraju.

reaktor

Ćwiczenie MS64

- Znajdź takich zawodników, którzy są starsi od swoich trenerów.
- Znajdź takich zawodników, którzy są młodsi od swoich trenerów.

reaktor

Ćwiczenie MS65

- Wypisz listę wszystkich par zawodników tej samej narodowości takich, że pierwszy z zawodników jest wyższy od drugiego.

Agregacja (grupowanie) wyników

Grupowanie wyników

- Kolejnym bardzo ważnym mechanizmem języka SQL jest agregacja.
- Pozwala on na zebranie wielu wierszy w jeden pod warunkiem, że mają one identyczną wartość wskazanych atrybutów.
- Oprócz wartości tych atrybutów można także wypisywać wartości pewnych funkcji bazujących na pozostałych atrybutach.

```
SELECT CLMNname1, …, CLMNnameN, funkcja(CLMNnameX)
FROM TBLname
WHERE warunki
GROUP BY CLMNname1, …, CLMNnameN
ORDER BY wyrażenia;
```

reaktor

PWN

Grupowanie wyników

- Należy pamiętać, że kolumny nie wymienione w klauzuli GROUP BY mogą pojawić się jedynie jako argumenty funkcji agregujących.
- Podstawowe funkcje agregujące zostały zebrane w tabeli.

Funkcja agregująca	Opis
count(*)	Ilość wartości
count(wyrażenie)	Ilość wartości wyrażenia różnych od NULL
count(distinct wyrażenie)	Ilość unikatowych wartości wyrażenia
sum(wyrażenie)	Suma wartości wyrażeń
max(wyrażenie)	Maksimum z wartości wyrażeń
min(wyrażenie)	Minimum z wartości wyrażeń
avg(wyrażenie)	Średnia z wartości wyrażeń
stddev(wyrażenie)	Odchylenie standardowe wartości wyrażeń

reaktor

PWN

Klauzula HAVING

- Grupowanie wierszy zgodnie z klauzulą GROUP BY następuje po ich wybraniu na podstawie warunków z klauzuli WHERE.
- Wynika z tego, że w części WHERE nie można umieścić żadnych warunków dotyczących zagregowanych wierszy, w tym wartości funkcji agregujących.
- Do nakładania warunków na wartości funkcji agregujących służy klauzula HAVING.
- Umieszczona po GROUP BY może się już do nich odwoływać.

Klauzula HAVING

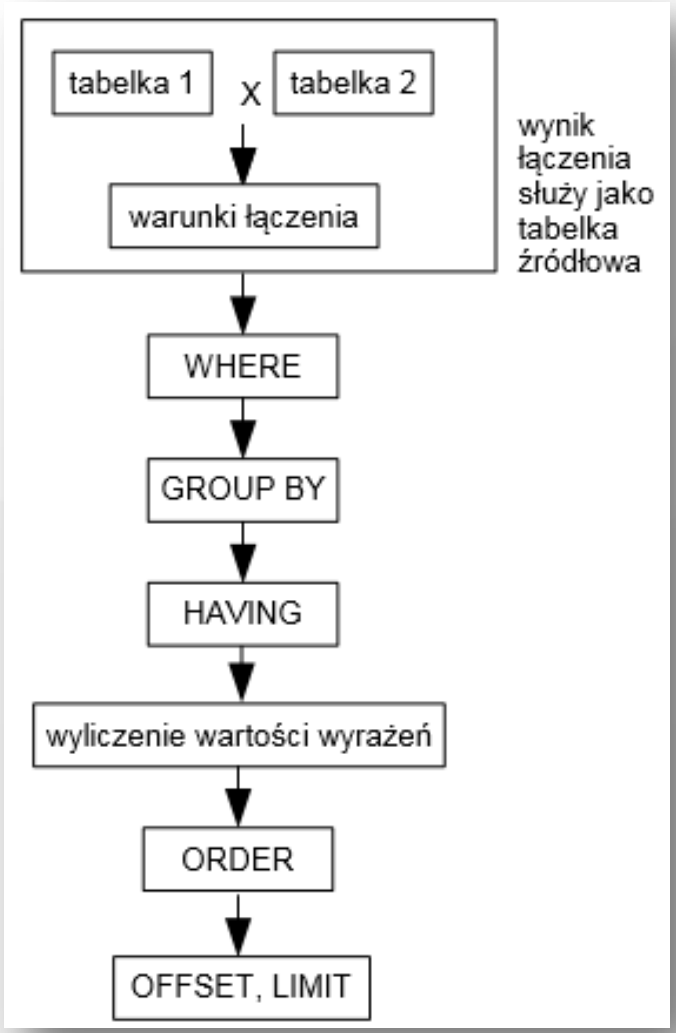
```
SELECT wyrażenia...

FROM tabele...

WHERE warunki...

GROUP BY wyrażenia... HAVING warunki...

ORDER BY wyrażenia...;
```



reaktor

Ćwiczenie MS66

- Podaj wielkości drużyn narodowych.

reaktor

Ćwiczenie MS67

- Policz, ilu jest wszystkich zawodników.

reaktor

Ćwiczenie MS68

- Podaj ilości zawodników wyższych niż 180 cm w poszczególnych drużynach.

reaktor

Ćwiczenie MS69

- Podaj listę ekip uporządkowaną według średniego wzrostu zawodników.

reaktor

Ćwiczenie MS70

- Sprawdź, jaki jest największy wzrost w poszczególnych krajach.

reaktor

Ćwiczenie MS71

- Sprawdź, jaki jest największy wzrost wśród wszystkich.

reaktor

Ćwiczenie MS72

- Policz, ilu zawodników urodziło się w poszczególnych kwartałach.

reaktor

Ćwiczenie MS73

- Policz, ilu zawodników urodziło się w poszczególnych latach w poszczególnych kwartałach.

reaktor

Ćwiczenie MS74

- Policz, jaka jest średnia wielkość ekipy narodowej.

reaktor

Ćwiczenie MS75

- Wypisz listę ekip. Dla każdej podaj, ilu jest w niej zawodników mających powyżej 180 cm wzrostu. W liście nie uwzględniaj tych ekip, w których takich zawodników jest mniej niż dwóch.

reaktor

Ćwiczenie MS76

- Wypisz tylko te ekipy, w których średnia wzrostu jest równa co najmniej 180 cm.

Podzapytania

Podzapytania

- Rozwiązaniem, które pozwala w pełni wykorzystać możliwości grupowania i funkcji agregujących są podzapytania.
- Poznane dotychczas polecenie SELECT może być zagnieżdżane. Może się ono pojawić w innym zapytaniu jako:
  - **źródło danych w części FROM – wynik zapytania traktowany jest tak, jak wszystkie pozostałe tabele, jedynym wymaganiem jest obowiązek nadania podzapytaniu nazwy (aliasu),**
  - **wartość skalarna wszędzie tam, gdzie jest ona dopuszczalna (np. w warunkach WHERE) – jedynym wymaganiem jest to, żeby wynik takiego zapytania był jednym wierszem jednokolumnowej tabeli.**
- Podzapytanie otaczamy nawiasami, aby uniknąć trudno wykrywalnych błędów i niejednoznaczności należy tak dobierać aliasy, żeby nie powtarzały się poszczególnych częściach zapytania i podzapytań.

reaktor

Ćwiczenie MS77

- Znajdź zawodników wyższych od Małysza.

reaktor

Ćwiczenie MS78

- Znajdź zawodników cięższych od Małysza.

reaktor

Ćwiczenie MS79

- Znajdź zawodnika wyższego niż najcięższy.

reaktor

Ćwiczenie MS80

- Znajdź zawodników starszych niż Heinz Kuttin

reaktor

Ćwiczenie MS81

- Znajdź zawodników o wzroście takim samym jak Janne Ahonen

reaktor

Ćwiczenie MS82

- Znajdź imię i nazwisko najwyższego zawodnika.

reaktor

Ćwiczenie MS83

- Wypisz zawodników cięższych niż średnia wśród wszystkich.

reaktor

Ćwiczenie MS84

- Wypisz zawodników cięższych niż przeciętny zawodnik z Polski.

reaktor

Ćwiczenie MS85

- Wypisz zawodników cięższych niż średnia w danej ekipie.

reaktor

Ćwiczenie MS86

- Podaj ilości zawodników wyższych niż 180 cm w poszczególnych drużynach, ale tak, żeby w tabelce istniały wiersze także dla krajów, w których nie ma takich zawodników.
- W takim wypadku oczywiście w kolumnie oznaczającej liczbę zawodników powyżej 180 cm powinna być wartość 0.

Tworzenie widoków (perspektyw)

Widoki (perspektywy)

- Czasami podczas pracy z bazami danych pojawia się sytuacja, że korzystamy często z jakiejś tabeli z stałymi parametrami warto w takim momencie stworzyć widok, który zaoszczędzi nam czasu.
- Widokiem (lub perspektywą) nazywam trwałą definicję tabeli pochodnej, która to definicja przechowaywana jest w bazie danych.

```
CREATE VIEW nazwa_widoku AS SELECT … ;
```

Zastosowanie widoków

```
CREATE VIEW zawodnicy_pl

AS SELECT

      imie, nazwisko, kraj

FROM

      zawodnicy

WHERE

      kraj = 'POL';

SELECT * FROM zawodnicy_pl;
```

Modyfikowanie widoków

- Za pomocą polecenia ALTER VIEW możemy modyfikować istniejący widok:

```
ALTER VIEW nazwa_widoku as SELECT … ;
```

- Za pomocą polecenia DROP możemy usunąć całkowicie widok:

```
DROP VIEW nazwa_widoku;
```

reaktor

Ćwiczenie MS87

- Utwórz widok ułatwiający pracę z tabelą zawodnicy.
- Zmodyfikuj utworzony widok.
- Usuń ten widok.

Tworzenie wyzwalaczy

Wyzwalacze

- Triger, zwany także wyzwalaczem jest to skrypt (fragment kodu) wykonywany w przypadku zajścia jakiegoś zdarzenia w bazie danych (np. dodania danych, ich modyfikacji, czy usunięcia).
- Istnieje kilka typów wyzwalaczy -skoncentrujemy się na dwóch: BEFORE i AFTER.
- Dla każdego typu istnieją trzy zdarzenia powodujące wykonanie wyzwalacza i są to:
  - **AFTER DELETE** – wykonanie wyzwalacza **po** operacji usunięcia rekordu
  - **AFTER INSERT** - wykonanie wyzwalacza **po** dodaniu rekordu
  - **AFTER UPDATE** - wykonanie wyzwalacza **po** zmodyfikowaniu rekordu
  - **BEFORE DELETE** - wykonanie wyzwalacza **przed** operacji usunięcia rekordu
  - **BEFORE INSERT** - wykonanie wyzwalacza **przed** dodaniu rekordu
  - **BEFORE UPDATE** - wykonanie wyzwalacza **przed** zmodyfikowaniu rekordu

Konstrukcja wyzwalacza

```
CREATE TRIGGER nazwa_triggera

BEFORE INSERT ON -- zdarzenie określające kiedy trigger zostanie wyzwolony

TBLname -- tabela na której triger zostanie założony

FOR EACH ROW BEGIN

    ...  -- skrypt wykonywany przez triger

END
```

reaktor

Ćwiczenie MS88

- Dodaj do tabeli trenerów kolumnę liczba_zawodnikow.
- Wprowadź przykładowe dane dot. liczby trenowanych zawodników przez danego trenera.
- Utwórz trigger, który będzie przyporządkowywał zawodnika do danego trenera zgodnie z kolumną kraj.

Projektowanie baz danych

reaktor

PWN

Projektowanie baz danych

- Zależnie od stopnia komplikacji dziedziny, o której dane będziemy przechowywać w naszej bazie, zaprojektowanie jej może okazać się proste lub trudniejsze, jest zaś bardzo ważne, aby projekt był dobry, gdyż przez resztę czasu spędzonego nad rozwijaniem aplikacji będziemy musieli ponosić konsekwencje ewentualnych błędnych decyzji.
- Proces projektowania bazy danych ma na jednym swoim końcu (czyli na początku) koncepcję jakie dane chcemy przechowywać, zaś na drugim końcu (tym, do którego chcemy szczęśliwie dotrzeć), konkretny kształt fizycznej bazy danych.
- Sytuację dodatkowo utrudnia fakt, że wspomniana koncepcja może nie znajdować się w naszej głowie, a w głowie klienta, szefa, zleceniodawcy.
- Jak powszechnie wiadomo klient taki zazwyczaj nie wie czego chce, a jeśli nawet wie, to przecież nam nie powie, a przynajmniej nie bez walki.

reaktor

PWN

Diagram ERD

- Jednym z diagramów, których rysowanie i aktualizowanie ma głęboki sens jest tak zwany diagram związków encji, inaczej ERD, od angielskiego Entity-Relationship Diagram. Diagram ten może obrazować kolejne stadia projektu bazy danych, od modelu koncepcyjnego aż do konkretnego kształtu tabel w bazie danych.
- Na diagramie ERD, zgodnie z nazwą, pojawiać się będą encje i zależności między nimi. W momencie kiedy doprowadzamy proces projektowania bazy danych do szczęśliwego końca, encja staje się tabelą.

reaktor

PWN

Czym są klucze tabeli

- Klucze to zbiory atrybutów mających określoną właściwość.
- Dzięki nim, możemy jednoznacznie identyfikować każdy pojedynczy wiersz.
- Znajomość pojęć kluczy podstawowych i obcych jest niezbędna do tworzenia zapytań, odwołujących się do wielu tabel.

reaktor

PWN

Klucz podstawowy

- To wybrany (zazwyczaj najkrótszy), jednoznacznie identyfikujący każdy, pojedynczy wiersz, zbiór atrybutów (kolumn) danej relacji (tabeli).
- Jest to pierwszy z wymienionych do tej pory kluczy, który ma faktyczne, fizyczne odwzorowania w implementacji bazy danych.
- Każda tabela może mieć tylko jeden taki klucz.

reaktor

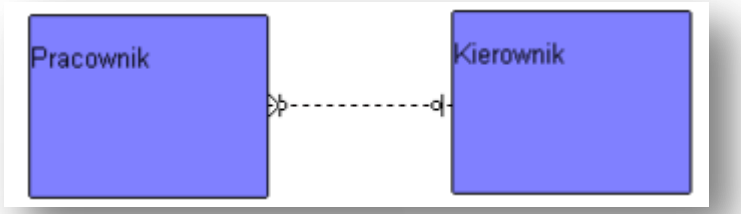
PWN

Klucz obcy

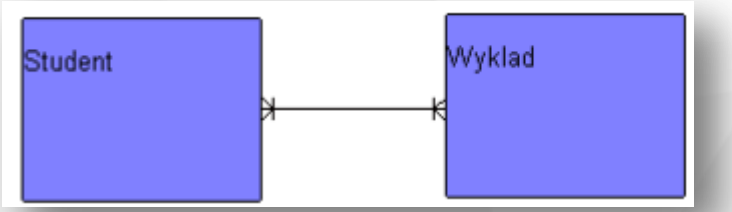
- Atrybut lub zbiór atrybutów, wskazujący na KLUCZ GŁÓWNY w innej RELACJI (tabeli).
- Klucz obcy to nic innego jak związek, relacja między dwoma tabelami.
- W tabeli powiązanej kluczem obcym, trzeba powielić tą strukturę (zbiór atrybutów) aby móc jednoznacznie wiązać rekordy z dwóch tabel.
- Definicja klucza obcego, pilnuje aby w tabeli powiązanej, w określonych atrybutach, znaleźć się mogły tylko takie wartości które istnieją w tabeli docelowej jako klucz główny.
- Klucz obcy może dotyczyć również tej samej tabeli.

Związki między encjami

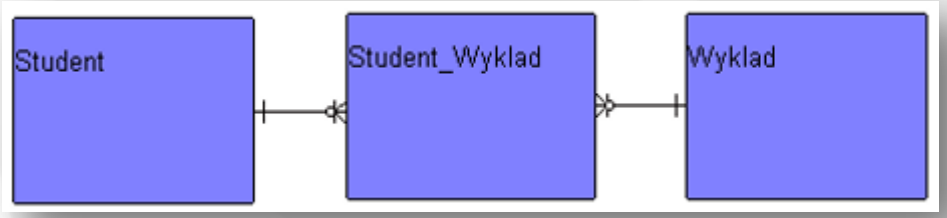
- Jeden do wielu (1:n)



- Wielu do wielu (n:m)



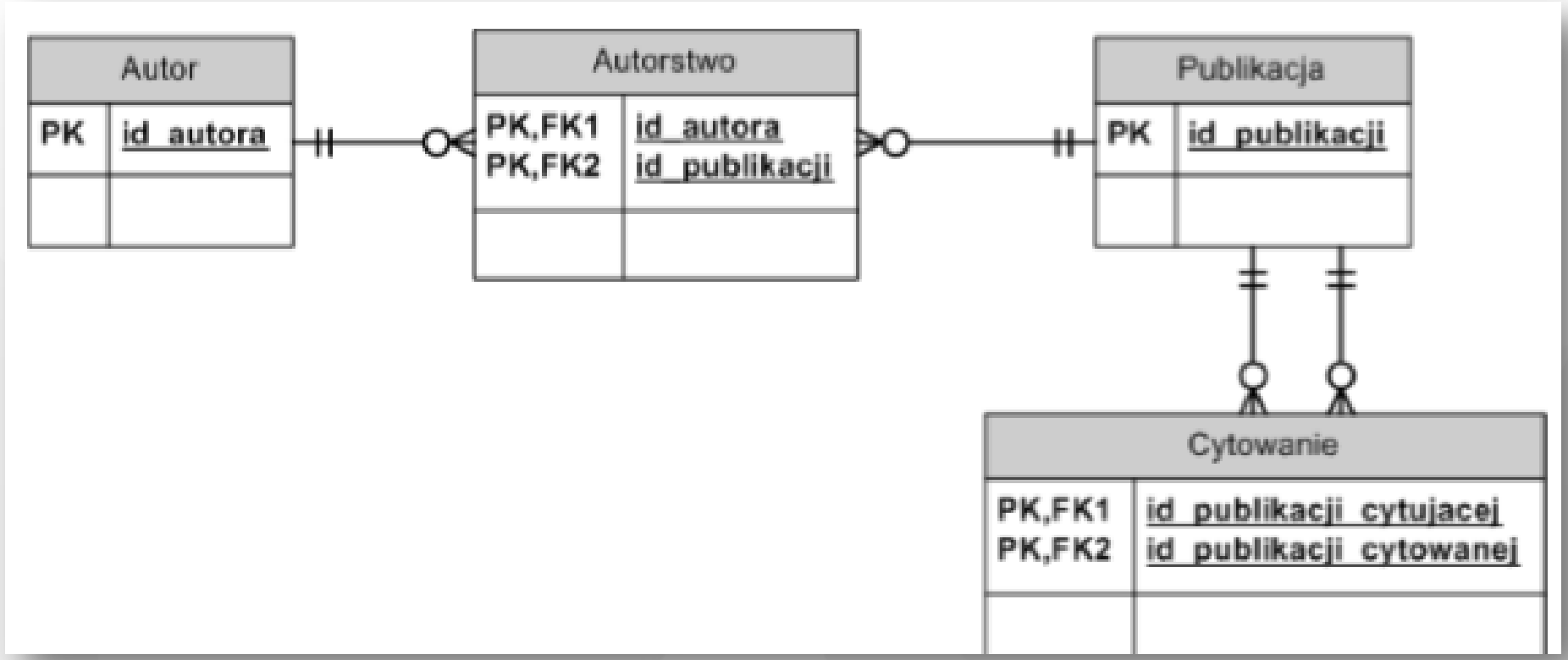
- Jeden do jednego (1:1)



Atrybuty encji

- Na diagramie związków encji możemy, a nawet powinniśmy w miarę przechodzenia od modelu koncepcyjnego do modelu fizycznego uwzględniać atrybuty (czyli kolumny tabel).

Prosty diagram ERD



Pierwsza postać normalna

- Pierwsza postać normalna
  - wszystkie atrybuty w relacji przyjmują wyłącznie wartości atomowe (niepodzielne).

id_zawodnika	nazwisko_zawodnika	lista_zawodow
1027	Adam Małysz	1, 5, 7, 3
1028	Jan Kowalski	2, 3, 18
...	...	...

id_zawodow	nazwa_zawodow
1	Zakopane 2001
2	Sarajewo 2003
...	...

id_zawodnika	nazwisko_zawodnika
1027	Adam Małysz
1028	

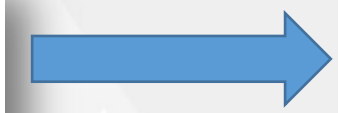
id_zawodow	nazwa_zawodow
1	Zakopane 2001
2	Sarajewo 2003
...	...

id_zawodnika	id_zawodow
1027	1
1027	5
1027	7
1027	3
1028	2
1028	3
1028	18
...	...

Druga postać normalna

- Druga postać normalna
  - atrybuty niekluczowe zależą od całego klucza.

id_studenta	nazwisko_studenta	id_wykładu	nazwa_wykładu
173	Piotr Polkowski	96	bazy danych
174	Artur Dworakowski	91	podstawy PHP
...	...	...	...



id_studenta	nazwisko_studenta
...	...

id_wykładu	nazwa_wykładu
...	...

id_studenta	id_wykładu
...	...

Trzecia postać normalna

- Trzecia postać normalna
  - atrybuty niekluczowe zależą tylko od klucza.

id_wykładu	nazwa_wykładu	wykładowca	pokój_wykładowcy
1	walidacja formularzy	Jan Kowalski	517
2	javascript	Jan Kowalski	405
...	...	...	...

id_wykładu	nazwa_wykładu	wykładowca
...	...	...

wykładowca	pokój_wykładowcy
...	...

Dziękuję za uwagę!

Dodatek 1
Dodawanie uprawnień dla użytkowników

Uprawnienia użytkowników

- Każda instalowana na serwerze aplikacja, która korzysta z baz danych powinna czynić to za pomocą dedykowanego użytkownika z prawami dostępu wyłącznie do konkretnej bazy.
- Należy unikać sytuacji gdy ten sam użytkownik, a tym bardziej root, obsługuje różne aplikacje. Tym bardziej, że stworzenie dedykowanych baz danych i użytkowników to kwestia raptem kilku komend.

Logowanie do MySQL

- Do administracji bazami MySQL najczęściej używa się domyślnie tworzonego konta o nazwie root, o nieograniczonych możliwościach.
- Czasami, ze względów bezpieczeństwa, tworzy się osobne konto o tych samych uprawnieniach jednak z inną nazwą użytkownika.

Tworzenie nowego użytkownika

- Kolejnym krokiem jest utworzenie użytkownika poleceniem:

```
CREATE USER `username'@'localhost`

IDENTIFIED BY `password';
```

- username – nazwa użytkownika
- localhost – użytkownik może łączyć się z serwerem lokalnym na którym zainstalowany jest MySQL
- password – najlepiej wygenerowane losowo hasło użytkownika

Przypisywanie uprawnień do użytkownika

- Utworzony użytkownik niewiele zdziała bez odpowiednich praw, dlatego w zależności od potrzeb, można je nadać na całą bazę lub na konkretną tabelę:

```
GRANT ALL PRIVILEGES ON *.* TO `username'@'localhost';

GRANT ALL PRIVILEGES ON DBname.* TO `username'@'localhost';
```

- Na sam koniec trzeba przeładować uprawnienia poleceniem:

```
FLUSH PRIVILEGES;
```

- Połączenie z bazą danych zamyka się poleceniem:

```
QUIT
```

Koniec

reaktor

MS89

- Korzystając z bazy danych faktury i płatności sprawdź czy Huta Szkla Plock opłaciła wszystkie przelewy

reaktor

MS90

- Korzystając z bazy danych faktury i płatności oblicz sumę faktur opłaconych przez firmę PCK Opole w roku 2015

reaktor

MS91

- Korzystając z bazy danych faktury i płatności oblicz kwotę faktur wystawionych w poszczególnych latach

reaktor

MS92

- Korzystając z bazy danych płatności oblicz kwotę faktur wystawionych w poszczególnych kwartałach i latach

reaktor

MS93

- Korzystając z bazy danych faktury i płatności oblicz kwotę faktur opłaconych w poszczególnych kwartałach i latach

reaktor

MS94

- Korzystając z bazy danych faktury i płatności podaj ogólne saldo płatności

reaktor

MS95

- Korzystając z bazy danych faktury i płatności podaj ogólne saldo płatności

reaktor

MS96

- Korzystając z bazy danych faktury i płatności podaj saldo płatności dla firmy PCK Opole

reaktor

MS97

- Korzystając z bazy danych faktury i płatności podaj saldo płatności dla firmy PCK Opole i dodatkowo jeśli saldo jest dodatnie niech w osobnej kolumnie pojawi się napis - saldo dodatnie w przeciwnym razie - saldo ujemne

reaktor

MS98

- Korzystając z bazy danych kraje wypisz liczbę krajów występujących w poszczególnych kontynentach

reaktor

MS99

- Korzystając z bazy danych kraje wypisz nazwy krajów znajdujących się na poszczególnych kontynentach – jeśli dany kontynent nie zajmuje żaden kraj to niech wyświetli się null

reaktor

MS100

- Korzystając z bazy danych kraje wypisz kraj, który posiada najdłuższą nazwę flagi.

reaktor

MS101

- Na podstawie bazy danych kraje przekształć tak strukturę bazy danych aby można było przedstawić ją w postaci relacji na diagramie ERD.

reaktor

MS102

- Na podstawie bazy danych faktury i płatności przekształć tak strukturę bazy danych aby można było przedstawić ją w postaci relacji na diagramie ERD.

Projekt 1

Projekt 1: wybór tematyki

- ❑ Baza danych sklepu internetowego
- ❑ Baza danych ewidencji pracowników w firmie
- ❑ Baza danych firmy logistycznej
- ❑ Baza danych e-biblioteki
- ❑ Baza danych firmy organizującej eventy
- ❑ Baza danych kliniki stomatologicznej
- ❑ Baza danych hotelu
- ❑ Baza danych restauracji

reaktor

PWN

Projekt 1: implementacja

- ❑ Zaprojektuj i zaimplementuj wybraną bazę danych tak aby składała się z minimum 5 tabel.
- ❑ Wprowadź przykładowe dane do powyższych tabel.
- ❑ Zastosuj wszystkie klauzule poznane w module: Bazy danych.
- ❑ Utwórz minimum 3 widoki ułatwiające wyświetlanie ważnych informacji.
- ❑ * Zastosuj trigger, który zautomatyzuje pewne operacje.

reaktor

PWN

Projekt 1: warunki zaliczenia

- Kompletny zbiór komend potrzebnych do utworzenia bazy danych i wszystkich jej elementów oraz wykonania operacji na danych znajdujących się w tabelach wysyłamy na adres: .....
- Deadline: 27.07.2017
- Krótkie prezentacje projektów: 28.07.2017