

Podstawy SQL

Podstawy relacyjnych baz danych oraz języka SQL były przedstawione na innym przedmiocie, ty jest tylko krótkie przypomnienie najważniejszych elementów składni.

Tworzenie bazy danych

Składnia

```
CREATE <nazwa-bazy-danych>  
  DEFAULT CHARACTER SET <zestaw-znaków>  
  DEFAULT COLLATE <sortowanie>
```

Przykład

```
CREATE forum  
  DEFAULT CHARACTER SET utf8mb4  
  DEFAULT COLLATE utf8mb4_polish_ci
```

Kodowanie utf8mb4 (4-Byte UTF-8 Unicode Encoding), obecnie zalecane, umożliwia zapisanie wszystkich znaków zestawu podstawowego (BMP) i znaków pomocniczych

Tworzenie użytkownika

Składnia

```
CREATE USER <nazwa-użytkownika>  
GRANT <uprawnienia> ON <baza-danych>@<host>  
  TO <nazwa-użytkownika>
```

Przykład

```
CREATE USER inaz  
GRANT ALL ON inaz TO 'inaz'@'localhost'
```

Studenci mają wszystkie uprawnienia (ALL) do swojej bazy danych – obejmuje to wszystkie operacje na tabelach i wierszach oprócz nadawania uprawnień innym użytkownikom. Na ćwiczeniach PAI nazwa użytkownika jest taka sama, jak bazy danych – zwykle tak nie jest, a na pewno nie musi być

Tworzenie tabel

Typy danych

- INT - liczba całkowita, odpowiednik int w C++/C#
- BIGINT - liczba całkowita duża, często jest kluczem głównym
- FLOAT - liczba zmiennoprzecinkowa, odp. double w C++/C#
- DATE, TIME, DATETIME – data i czas
- TIMESTAMP – czas utworzenia rekordu lub ostatniej operacji
- VARCHAR (n) - ciąg znaków o zmiennej długości, max n znaków
- TEXT - ciąg znaków o zmiennej długości, maks. 65000 znaków
- BLOB - dane binarne (byte string) o zmiennej długości

Typy TEXT i BLOB omijają ograniczenie objętości wiersza (65e3 bajtów)

Atrybuty danych

- NOT NULL - kolumna nie może być pusta – wymagane jest podanie wartości (za wyjątkiem kolumny, dla której podano DEFAULT lub AUTO_INCREMENT)
- DEFAULT wartość - domyślna wartość
- AUTO_INCREMENT - automatyczna inkrementacja (zwykle dla kluczy głównych)
- PRIMARY KEY - klucz główny
- UNIQUE - nie są dopuszczalne duplikaty wartości

Dyrektywy

- ENGINE = <silnik>
Silniki MySQL/MariaDB:
 - InnoDB – zaawansowane możliwości blokowania/transakcji
 - MyISAM – szybszy, ale nie obsługuje transakcji
 - MEMORY – najszybszy, ale dane nie są trwale zapisywane
- CHARSET = <kodowanie>
Domyślne kodowanie znaków
- COLLATE = <sortowanie>
Domyślne reguły sortowania łańcuchów znaków

Tworzenie tabel

Składnia

```
CREATE TABLE <nazwa-tabeli> (  
    <nazwa-kolumny> <typ> [atrybuty] [,  
    ...]  
) [atrybuty dyrektywy]
```

Przykład 1

```
CREATE TABLE users (  
    userid BIGINT AUTO_INCREMENT PRIMARY KEY,  
    login VARCHAR(20) UNIQUE NOT NULL,  
    pass VARCHAR(30) NOT NULL,  
)  
ENGINE = InnoDB  
CHARSET = utf8mb4  
COLLATE = utf8mb4_polish_ci
```

Przykład 2

```
CREATE TABLE posts (  
    postid BIGINT AUTO_INCREMENT PRIMARY KEY,  
    userid BIGINT,  
    title VARCHAR(200),  
    text VARCHAR(2000),  
    created TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
)  
ENGINE = InnoDB  
CHARSET = utf8mb4  
COLLATE = utf8mb4_polish_ci
```

Usuwanie tabel i baz danych

Składnia

```
DROP TABLE <nazwa-tabeli>
DROP DATABASE <nazwa-bazy-danych>
```

Przykład

```
DROP TABLE users
DROP DATABASE forum
```

Wprowadzanie danych

Składnia

```
INSERT INTO <nazwa-tabeli>
VALUES (wartość, wartość, ...)

INSERT INTO <tabela> (kolumna, kolumna, ...)
VALUES (wartość, wartość, ...)
```

Przykład

```
INSERT INTO users (userid, login, pass)
VALUES (NULL, 'root', 'kernel21')
```

Ważne! Łańcuchy znaków, daty i czas trzeba podawać ograniczone apostrofami (liczby można, ale nie ma wymogu)

Pobieranie danych – jedna tabela

Składnia

```
SELECT <co?> FROM <tabela>
[ WHERE <warunki> ]
```

Jako <co?> można podać:

- * - wszystkie kolumny,
- <kolumna> AS <nazwa> - kolumna ze zmienioną nazwą
- <kolumna> [, <kolumna>, ...] - wybrane kolumny

Przykład

```
SELECT * FROM users
SELECT login AS l, pass AS p FROM users
```

Można pobierać wartości kolumn lub wynik operacji arytmetycznych i/lub funkcji na tych wartościach; Można również nadać nazwę wynikowi takiej operacji:

```
<wyrażenie> AS <nazwa>
```

Przykład

```
SELECT cena * ilość AS wartosc ...
SELECT CONCAT(imie, ' ', nazwisko) AS nazwa
```

Do określania warunków można użyć:

- operatorów relacji (=, <>, >, >=, <, <=)
- operatora IS NULL
- operatora LIKE (w rodzaju: LIKE '<znaki>%')

Przykład

```
SELECT * FROM users WHERE login LIKE 'r%'
SELECT pass FROM users WHERE login='root'
```

Wyrażenia warunkowe można łączyć używając operatorów

- AND – koniunkcja
- OR – alternatywa

Przykład

```
SELECT * FROM posts
WHERE userid=1 AND created>'2024-01-01'
```

W wyrażeniach warunkowych można stosować operatory:

- IN <zbiór-wartości>
- BETWEEN <wartość> AND <wartość>

Przykład

```
SELECT * FROM users WHERE userid IN (1, 3, 5)
SELECT * FROM posts
WHERE created BETWEEN '2024-01-01' AND NOW()
```

Pobieranie danych – więcej tabel

Składnia (1)

```
SELECT <co?> FROM <tabela> [, <tabela>, ...]
[ WHERE <warunki> ]
```

Domyślnie tworzony jest iloczyn kartezjański tabel, dlatego warunki trzeba podać – typowo przez pary kluczy: klucz główny tabeli pierwszej = klucz obcy tabeli drugiej. Przykład

```
SELECT login, created, title
FROM users, posts
WHERE userId = authorId
```

Można podawać nazwy kolumn w notacji <tabela>.<kolumna>, ale jeżeli nazwy kolumn powtarzają się w różnych tabelach, to jest to obowiązkowe

Przykład

```
SELECT users.userId, login, created, title
FROM users, posts
WHERE users.userId = posts.userId
```

Składnia (2)

Jawne podanie warunku łączenia wierszy

```
SELECT <co?> FROM <tabela_1>  
  JOIN <tabela_2> ON <warunek>  
  [ WHERE <warunki> ]
```

Polecenie Join wymusza podanie warunku złączenia tabel (sam warunek – jak w składni 1) i oddziela go od warunków wybierania wierszy. Przykład

```
SELECT login, created, title  
  FROM users  
 JOIN posts ON users.userid = posts.userid  
  WHERE posts.date > NOW() - INTERVAL 7 DAY
```

Składnia (3)

Zamiast klauzuli ON można użyć klauzuli USING – pod warunkiem, że kolumna występuje w obu tabelach, w jednej jako klucz główny, w drugiej jako klucz obcy. Nazwa kolumny musi być w nawiasach:

```
SELECT <co?> FROM <tabela_1>  
  JOIN <tabela_2> USING ( <kolumna> )  
  [ WHERE <warunki> ]
```

Przykład

```
SELECT login, created, title  
  FROM users  
 JOIN posts USING (userid)  
  WHERE posts.date > NOW() - INTERVAL 7 DAY
```

Inne rodzaje złączeń

Jest kilka rodzajów złączeń: wewnętrzne (JOIN), zewnętrzne (OUTER JOIN), które można podzielić na lewe (LEFT JOIN) i prawe (RIGHT JOIN), a także pełne (FULL JOIN) oraz iloczyn kartezjański (CROSS JOIN):

```
SELECT <co?> FROM <tabela_1>  
  [LEFT/RIGHT/...] JOIN <tabela_2> ON <warunek>
```

Złączenie lewe zwróci wszystkie wiersze pierwszej tabeli, nawet jeżeli druga tabela nie ma pasującego wiersza (kolumny prawej tabeli będą uzupełnione wartościami NULL). Przykład:

```
SELECT login, created, title  
  FROM users  
 LEFT JOIN posts ON users.userid = posts.userid  
  WHERE posts.date > NOW() - INTERVAL 7 DAY
```

Dyrektywy ORDER BY i LIMIT

Składnia

```
SELECT <co?> FROM <tabela>
[ WHERE <warunki> ]
[ dyrektywy ]
```

ORDER BY <kolumna> [ASC | DESC] [, <kolumna>, ...]

LIMIT [n,] m – m wierszy od n (numeracja n od zera!)

Przykład

```
SELECT login, created, title FROM users
JOIN posts USING (userid)
WHERE posts.date > NOW() - INTERVAL 7 DAY
ORDER BY date DESC, login ASC
LIMIT 0, 20
```

Agregacja danych

Agregacja polega na połączeniu wierszy mających we wskazanej kolumnie tę samą wartość w jeden wiersz; Polecenie SELECT może pobierać wyniki funkcji agregujących:

- COUNT(*) – liczba zagregowanych wierszy
- COUNT(DISTINCT) – liczba unikalnych wartości
- SUM(<wyrażenie>) – suma
- MIN(<wyrażenie>) – wartość minimalna
- MAX(<wyrażenie>) – wartość maksymalna
- AVG(<wyrażenie>) – średnia

gdzie <wyrażenie> to kolumna lub wynik operacji arytmetycznych i/lub funkcji na wartościach kolumn, np. SUM(cena * liczbaSztuk).

Można też pobierać wartości dowolnych kolumn, ale trzeba je wymienić PRZED funkcjami agregującymi

Składnia

```
SELECT <co?> FROM <tabela>
[ WHERE <warunek> ]
GROUP BY <kolumna> [, <kolumna>]
[ HAVING <warunek> ]
```

GROUP BY <kolumna> [, <kolumna>] - kolumna lub lista kolumn, według których następuje agregacja danych

WHERE <warunek> - wybiera wiersze **do** agregacji

HAVING <warunek> - wybiera wiersze **po** agregacji

Przykład – autor i liczba jego wpisów w ciągu ostatniego miesiąca oraz data najnowszego wpisu, tylko dla autorów mających co najmniej 10 wpisów, posortowane malejąco:

```

SELECT login,
       COUNT(*) AS numPosts,
       MAX(created) AS lastPostDate
FROM users
     JOIN posts USING (userid)
WHERE created >= NOW() - INTERVAL 30 DAY
GROUP BY users.userid
HAVING numPosts >= 10
ORDER BY numPosts DESC

```

Kolejność dyrektyw

Składnia

```

SELECT <co?> FROM <tabela>
     JOIN <tabela> ON <warunki>
WHERE <warunki>
GROUP BY <kolumna>
HAVING <warunki>
ORDER BY <kolumny>
LIMIT <start>, <liczba-wierszy>

```

Wszystkie dyrektywy są opcjonalne, na ogół niezależne (wyjątek: HAVING można użyć tylko jeżeli jest też GROUP BY), ale jeżeli są używane, to w podanej wyżej kolejności

Modyfikacja danych

Składnia

```

UPDATE <tabela>
     SET <kolumna> = <wartość> [, <kol>=<wart>]
     [ WHERE <warunki> ]

```

Przykład

```

UPDATE users
     SET pass = 'kernel21'
     WHERE login = 'root'

```

Ważne! Bez podania warunków w dyrektywie WHERE zmienione zostaną wszystkie wiersze.

Usuwanie danych

Składnia

```

DELETE FROM <tabela>
     [ WHERE <warunki> ]

```

Przykład

```

DELETE FROM users WHERE login = 'pik'
DELETE FROM posts WHERE title LIKE '%covid%'
DELETE FROM posts

```

Ważne! Bez podania warunków w dyrektywie WHERE usunięte zostaną wszystkie wiersze.

PHP i bazy danych

Język PHP może zostać rozszerzony o mechanizmy dostępu do różnych baz danych:

- MySQL
- PostgreSQL
- Oracle
- MS SQL Server

Istnieje również rozszerzenie zapewniające wspólny interfejs dostępu do baz danych, niezależnie od ich typu – PDO (PHP Data Objects). PDO umożliwia nawiązywanie połączeń, wykonywanie zapytań i zarządzanie danymi. Od wersji 5.1 jest standardową częścią PHP.

Przebieg czynności podczas korzystania z bazy danych

- Nawiązywanie połączenia z serwerem baz danych
- Wybór bazy danych oraz uwierzytelnienie użytkownika tej bazy danych
- Ustawienie zestawu znaków dla połączenia z serwerem bazy danych
- Wykonanie operacji na bazie danych
- Wykorzystanie wyników zwróconych z bazy danych
- Zamknięcie połączenia

W przypadku MySQL i modułu mysqli, większość czynności wykonuje obiekt mysqli, tworzony w pierwszym kroku.

Nawiązywanie połączenia, wybór bazy danych oraz uwierzytelnienie

Potrzebne dane:

- Nazwa lub adres IP serwera baz danych (w praktyce *localhost*)
- Nazwa użytkownika bazy danych
- Hasło
- Nazwa bazy danych (użytkownik musi mieć uprawnienia do tej bazy danych)

Wszystkie potrzebne czynności wykonuje konstruktor obiektu mysqli:

```
$conn = new mysqli (<host>, <użytkownik>,  
                    <hasło>, <baza-danych>);  
  
$conn = new mysqli ("localhost", "inaz", "???", "inaz");
```

W większości wypadków serwery PHP i MySQL są na tym samym komputerze, a ponadto ze względów bezpieczeństwa dostęp z zewnątrz jest zabroniony, dlatego jako <host> należy podać *localhost*

Ustawienie zestawu znaków

Zestaw znaków dla połączenia z serwerem bazy danych ustawia zapytanie SQL SET:

```
$sql = "SET NAMES 'utf8'";  
$res = $conn->query ($sql);
```

Ustawienie tego samego zestawu znaków do komunikacji serwer bazy danych – PHP oraz PHP – przeglądarka gwarantuje prawidłowe zapisywanie i odczytywanie danych.

Obsługa błędów połączenia

Do sprawdzenia powodzenia tej operacji służy funkcja `mysqli_connect_errno()`, zaś funkcja `mysqli_connect_error()`, zwraca komunikat o błędzie:

```
$conn = new mysqli ("localhost", "inaz", "???", "inaz");
if (mysqli_connect_errno() != 0)
{
    $msg = mysqli_connect_error();
    echo "wystąpił błąd: $msg";
}
else ...
```

Wykonanie operacji na bazie danych

Do wykonywania wszelkich operacji na bazie danych służy metoda `query()` obiektu `mysqli`:

```
$sql = "<zapytanie-SQL>";
$res = $conn->query ($sql);
```

Metoda ta zwraca jeden z trzech możliwych rezultatów:

- `FALSE` jeżeli wystąpi błąd,
- `TRUE` jeżeli operacja kończy się powodzeniem, ale nie zwraca rezultatu (np. `DELETE`, `UPDATE`)
- Obiekt `mysqli_result`, z którego można pobierać wyniki, np. przy pomocy funkcji `fetch_assoc`.

Obiekt `mysqli_result` może być pusty, co w automatycznych konwersjach PHP jest równe `NULL` – z tego względu do sprawdzania wyniku metody `query()` należy używać operatora identyczności (`===`) a nie równości (`==`).

Wykorzystanie wyników zwróconych z bazy danych

Jeżeli zapytanie SQL – np. `SELECT` – zwraca wynik, to ma on postać obiektu `mysqli_result`. Kolejne wiersze można odczytywać funkcją `fetch_assoc()`, najlepiej w pętli `while`:

```
$sql = "SELECT * FROM posts";
$res = $conn->query ($sql);
if ($res===FALSE)
    // obsługa błędu
else
{
    while (($row = $res->fetch_assoc()) !== NULL)
        // korzystamy z tabeli $row
}
```

Każdy taki wiersz – odczytany przez `fetch_assoc()` – jest tablicą asocjacyjną, w której kluczami są nazwy kolumn, a wartościami – ich wartości: `<kolumna> -- <wartość>`:

```
echo "<table>";
echo "<tr> <th>Data <th>Tytuł";
while (($row = $res->fetch_assoc()) !== NULL)
{
    echo "<tr> <td>{$row["created"]} "
        . "<td>{$row["title"]} ";
}
echo "</table>";
```

Zamknięcie połączenia

Połączenie z bazą danych zamyka metoda `close()` obiektu `mysqli` utworzonego podczas nawiązywania połączenia:

```
$conn = new mysqli(...);  
  
// dostęp do bazy danych  
  
$conn->close();
```

Niepotrzebne połączenie należy zamknąć, aby zwolnić wykorzystywane przez nie zasoby.

Połączenie wszystkich czynności:

```
$conn = new mysqli ("localhost", "...", "...", "...");  
if (mysqli_connect_errno() != 0)  
{  
    $msg = mysqli_connect_error();  
    echo "wystąpił błąd: $msg";  
}  
else  
{  
    $sql = "SELECT * FROM Users";  
    $res = $conn->query ($sql);  
    if ($res === FALSE)  
    {  
        $msg = mysqli_error($conn);  
        echo "Błąd zapytania SQL: $msg";  
    }  
    else  
    {  
        while (($row = $res->fetch_assoc()) !== NULL)  
        {  
            echo "... {$row["column"]} ...";  
        }  
    }  
    $conn->close();  
}
```

Zmiany w PHP 8

W PHP 8 każdy błąd (połączenia lub zapytania) powoduje rzucenie wyjątku programowego. Można nadal korzystać z funkcji `mysqli_connect_error` i `mysqli_error`, ale trzeba najpierw przechwycić wyjątek. W najprostszym przypadku można całkiem pominąć obsługę błędów:

```
$conn = new mysqli ("...", "...", "...", "...");  
  
$sql = "SELECT * FROM Users";  
$res = $conn->query ($sql);  
  
while (($row = $res->fetch_assoc()) !== NULL)  
{  
    echo "... {$row["column"]} ...";  
}
```

PHP i bazy danych

Wygodniejsze podejście do SQL

Cały proces jest dość złożony, a ponieważ są to zasoby zewnętrzne, to musi tak być. Jeżeli z bazy danych korzysta wiele skryptów, to połączenie z bazą danych i zapytanie najlepiej zdefiniować w funkcji:

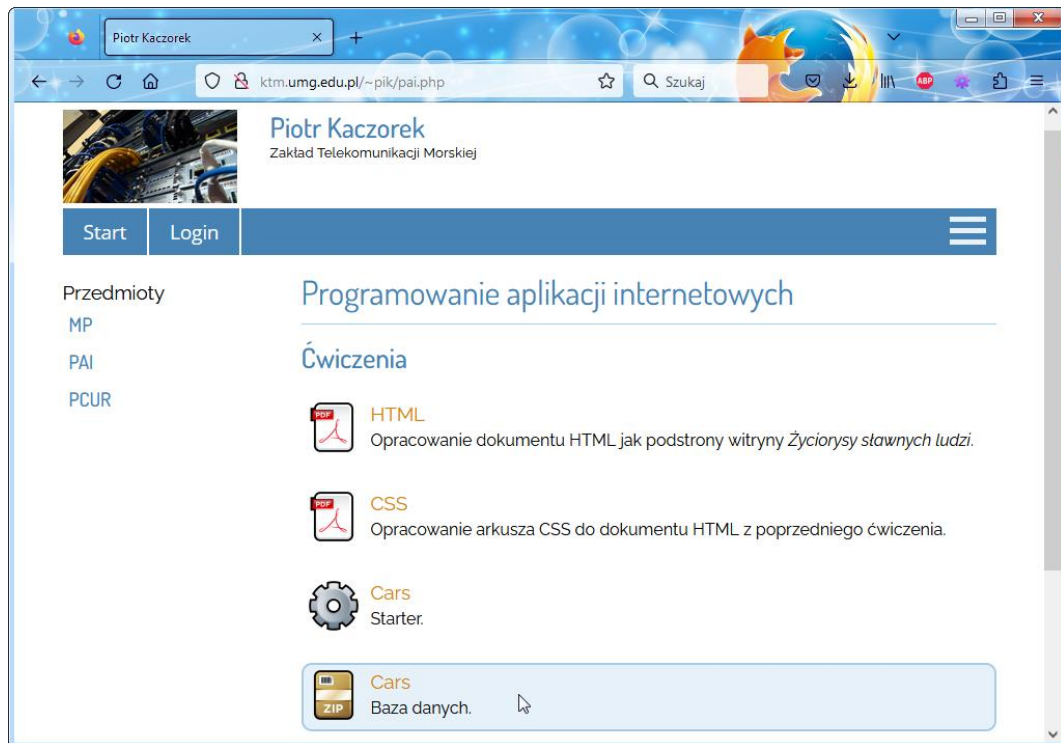
```
function dbQuery($sql) {  
    $conn = new mysqli (...);  
    $result = ...;  
    if (...) {  
    }  
    return $result;  
}  
  
$users = dbQuery("SELECT * FROM users");
```

Funkcja powinna też obsługiwać wystąpienie błędów, ustawiać kodowanie znaków itp.

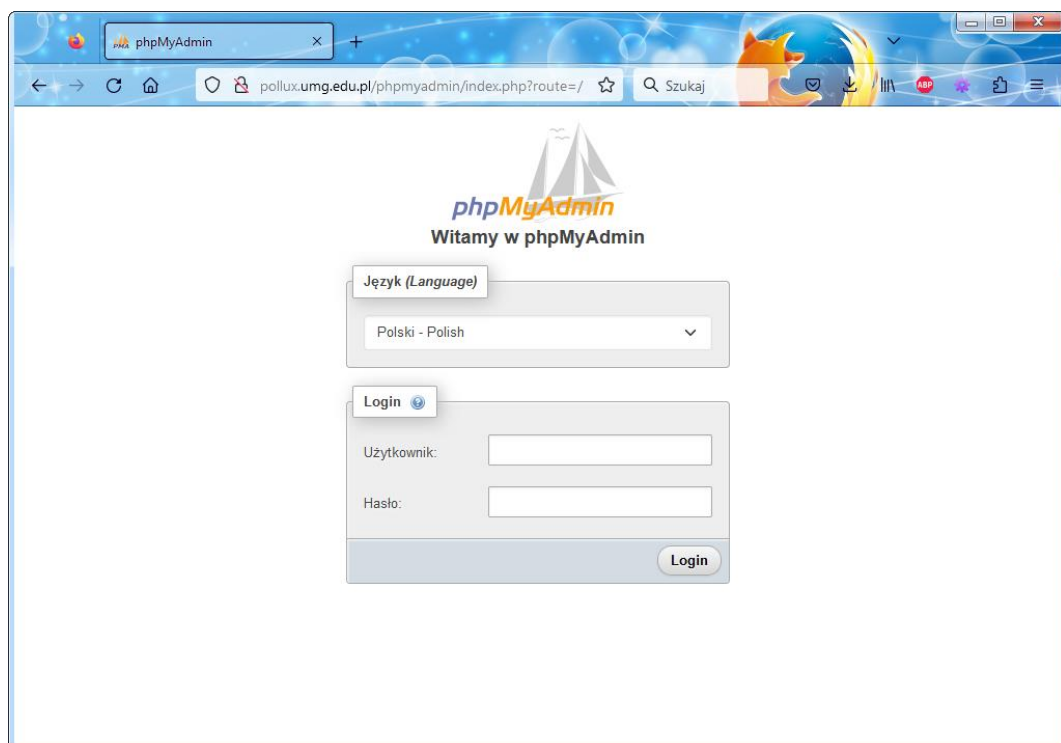
Zadania

Proszę zalogować się na swoje konto w serwerze baz danych i zaimportować bazę danych.

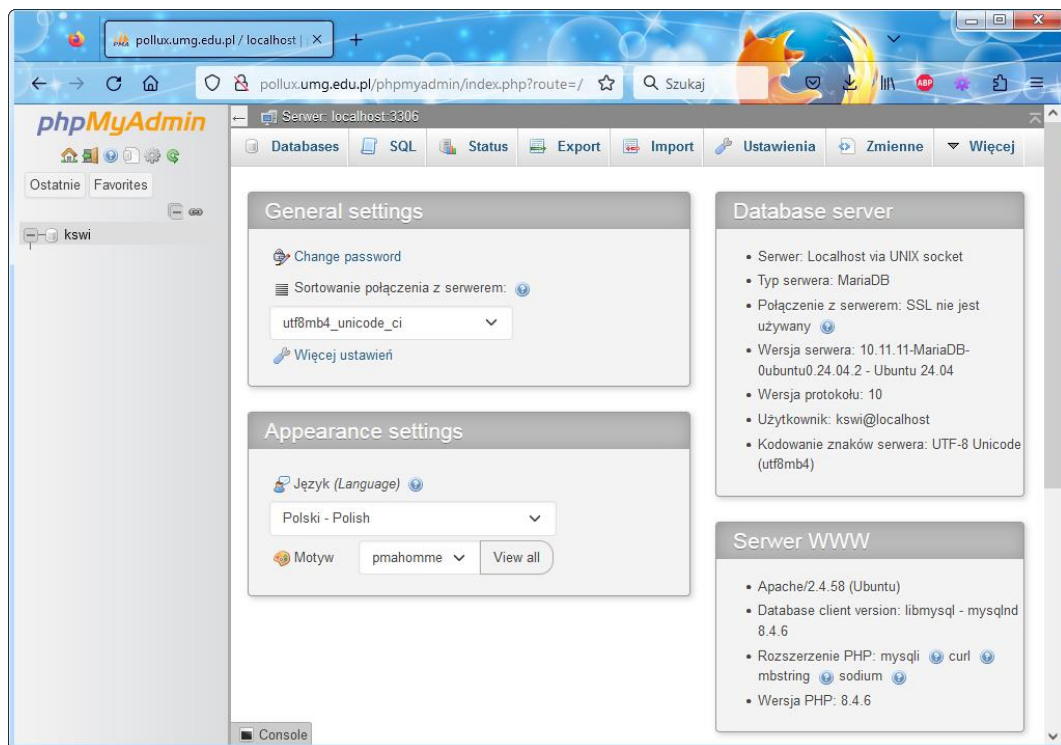
1. Pobrać plik bazy danych ze strony <http://ktm.umg.edu.pl/~pik/pai.php>



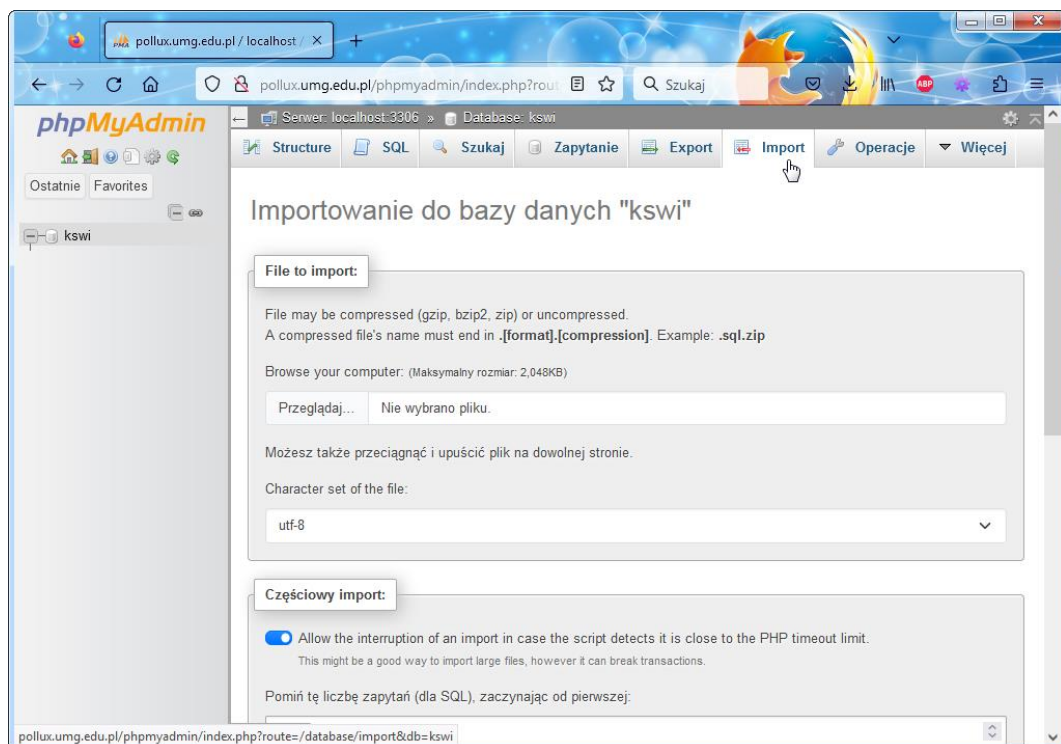
2. W przeglądarce wpisać adres <http://pollux.umg.edu.pl/phpmyadmin/>, wpisać login i hasło do bazy danych i zalogować się



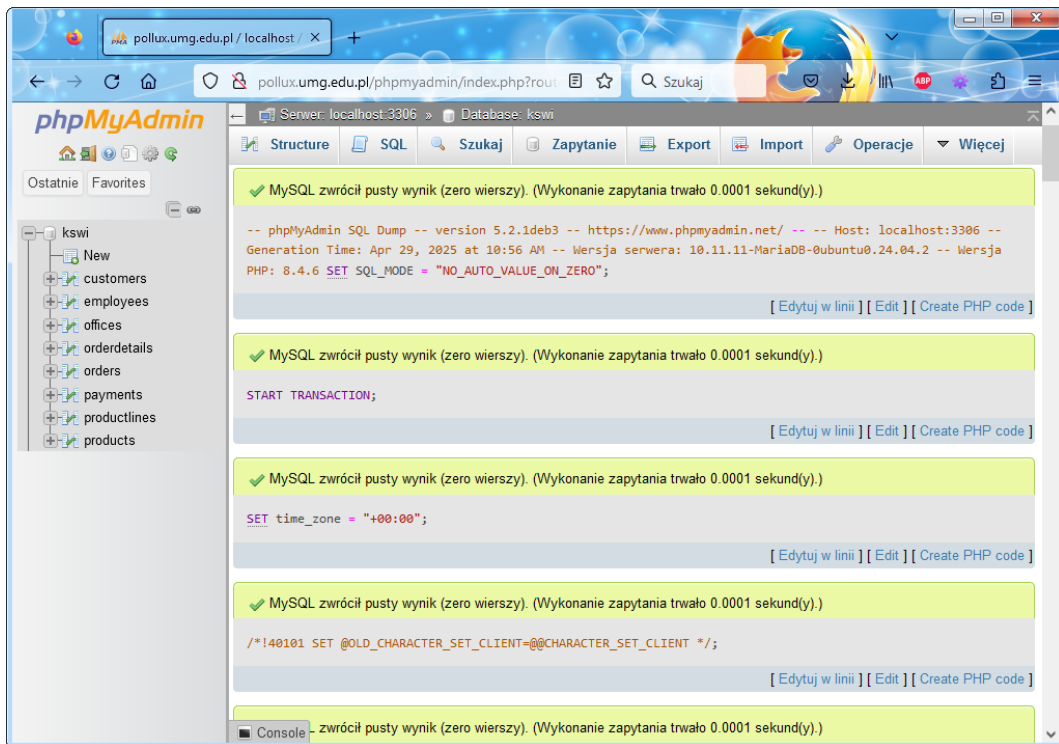
3. Zapoznać się z ustawieniami konta i bazy danych.



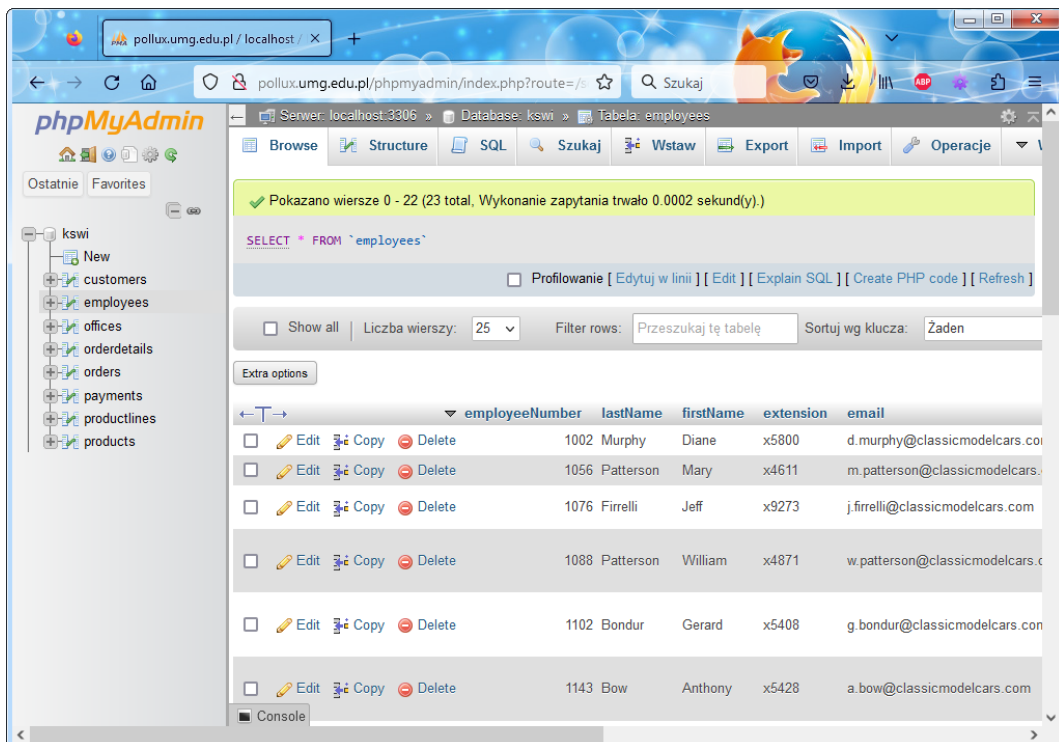
4. Wybrać (kliknięciem) swoją jedyną bazę danych i kliknąć przycisk *Import*.



5. W polu plik do importu (*File to import*) wskazać pobrany wcześniej plik *Cars.sql.zip*, przewinąć do końca formularz i kliknąć przycisk Import



6. Zapoznać się z zawartością poszczególnych tabel, zwracając szczególną uwagę na relacje między tabelami



Proszę wybrać (kliknięciem) najbardziej do zadania odpowiednią tabelę, a następnie w polu *Edytuj w linii* wprowadzić zapytanie SQL, które pozwoli pobrać z bazy danych:

1. Listę wszystkich pracowników (*employees*)
2. Listę numerów, imion i nazwisk oraz stanowisk wszystkich pracowników
3. Listę numerów, imion i nazwisk oraz stanowisk wszystkich pracowników, którzy są na stanowisku *Sales Rep*
4. Listę numerów, imion i nazwisk oraz stanowisk wszystkich pracowników, którzy nie są na stanowisku *Sales Rep*
5. Listę imion i nazwisk oraz numerów biur wszystkich pracowników, którzy są na stanowisku *Sales Rep*, posortowane według nazwisk i imion
6. Listę imion i nazwisk oraz numerów biur wszystkich pracowników, którzy są na stanowisku *Sales Rep*, posortowane według numerów biur, nazwisk i imion
7. Listę imion i nazwisk oraz numerów biur wszystkich pracowników, niezależnie od stanowiska, posortowane według numerów biur, nazwisk i imion, ale tylko z biur o numerach 3, 4 oraz 6
8. Listę imion i nazwisk oraz numerów biur wszystkich pracowników, którzy są na stanowisku *Sales Rep*, posortowane według numerów biur, nazwisk i imion, ale tylko z biur o numerach pomiędzy 3 a 5
9. Listę imion i nazwisk pracowników oraz nazw miast biur, z tabel *employees* i *offices*
10. Listę imion i nazwisk oraz nazw miast, w których poszczególni pracownicy pracują
11. Listę imion i nazwisk oraz nazw miast, w których pracownicy pracują, posortowane według numerów (nie nazw miast) biur oraz nazwisk i imion pracowników
12. Listę imion i nazwisk oraz nazw miast i krajów, w których pracownicy pracują, posortowane według nazw krajów i miast biur
13. Listę nazw miast i krajów biur oraz liczb pracowników, którzy pracują w tych biurach
14. Listę nazw miast i krajów biur oraz liczb pracowników, którzy pracują w tych biurach, posortowane malejąco według liczby pracowników
15. Listę nazw miast i krajów biur oraz liczb pracowników, którzy pracują w tych biurach, mających ponad 2 pracowników, posortowane malejąco według liczby pracowników
16. Listę nazw miast i krajów biur oraz liczb pracowników, którzy pracują w tych biurach, posortowane malejąco według liczby pracowników, ale tylko 3 pierwsze biura
17. Listę nazw, miast i krajów klientów (*customers*) oraz imion i nazwisk pracowników (*employees*), którzy są przypisani do tych klientów
18. Listę nazw, miast i krajów klientów oraz imion i nazwisk pracowników, którzy są przypisani do tych klientów, ale również klientów, którzy nie mają przypisanych pracowników
19. Listę imion i nazwisk pracowników oraz liczbę klientów, do których są przypisani
20. Listę imion i nazwisk pracowników oraz liczbę klientów, do których są przypisani, posortowane malejąco według liczby klientów, ale tylko 5 pierwszych osób

21. Listę wszystkich danych z tabeli szczegółów zamówienia (*orderdetails*), należące do zamówienia o numerze 10120
22. Listę kodów produktów oraz ich koszt (iloczyn liczby sztuk i ceny jednostkowej), należące do zamówienia o numerze 10120
23. Listę numerów zamówień oraz ich łącznych kosztów (sum iloczynów liczby sztuk i ceny jednostkowej)
24. Listę nazw klientów i numerów zamówień oraz łącznych kosztów zamówień (sum iloczynów liczby sztuk i ceny jednostkowej)
25. Listę nazw klientów, numerów i dat zamówień oraz łącznych kosztów zamówień (sum iloczynów liczby sztuk i ceny jednostkowej), dla zamówień złożonych w 2024 r.
26. Listę nazw klientów, numerów i dat zamówień oraz łącznych kosztów zamówień (sum iloczynów liczby sztuk i ceny jednostkowej), dla zamówień złożonych w 2024 r., posortowane malejąco według łącznych kosztów zamówień i tylko 3 najlepszych klientów