Język baz danych – SQL cz.1

1. *Wstęp*

SQL (ang. Structured Query Language – Strukturalny Język Zapytań) język umożliwiający dostęp i przetwarzanie danych w relacyjnej bazie danych. Jest międzynarodowym standardem, do którego stosują się wszyscy producenci relacyjnych baz danych.
Na zajęciach pracujemy z mysql-em zainstalowanynym na serwerze „linuxowym", natomiast do ćwiczeń w domu można zainstalować np. pakiet WAMP (www.wampserver.com) lub Xampp (www.apachefriends.org) lub webserv (www.webserv.pl). Jak utworzyć bazę i wprowadzić dane dowiesz się w dalszej części kursu na przykładzie pakietu wampserwer.
Uruchom linię poleceń mysql. (Po uruchomieniu programu kliknij na ikonie w pasku zadań).

Trochę przydatnych poleceń.

1) Tworzenie bazy danych

`CREATE DATABASE <nazwa_bazy>` - tworzy bazę o zadanej nazwie.
Po utworzeniu bazy wystarczy się z nią połączyć
`USE <nazwa_bazy>`
i jeżeli mamy nadane odpowiednie uprawnienia, możemy tworzyć tabele.

Ćwiczenie 1. Utwórz tabelę o nazwie world.

2) Tworzenie tabel

Do zdefiniowania nowej tabeli używamy instrukcji **CREATE TABLE**. Obowiązkowym parametrem instrukcji są: ***nazwa tworzonej tabeli, nazwy wchodzących w skład tabeli kolumn*** oraz ***typ danych*** przechowywanych w poszczególnych tabelach. MySQL nakładają pewne ograniczenia na każdy z tych parametrów. I tak, nazwa tabeli musi być zgodna z regułami nazewnictwa. W przypadku serwera MySQL nazwy tabel i kolumn:

- Mogą zawierać litery, cyfry i znaki specjalne.
- Mogą zawierać litery dowolnej wielkości, przy czym rozróżnianie dużych i małych liter zależy od konfiguracji systemu operacyjnego i samej bazy danych.
- Muszą być unikatowe — niedopuszczalne jest istnienie w bazie kilku tabel lub w pojedynczej tabeli kilku kolumn o tej samej nazwie.
- Nie powinny być terminem zastrzeżonym dla języka, np. nazwą instrukcji, funkcji lub klauzuli — w takim przypadku, posługując się nazwą, musimy umieszczać ją w apostrofach.
- Nie powinny kończyć się znakiem spacji.
- Nie mogą zawierać znaków ukośnika /, odwrotnego ukośnika \ i kropki.

Typy danych
Dla każdej kolumny należy zdefiniować określony typ danych. Określony typ danych powinien w maksymalnym stopniu odpowiadać rodzajowi przechowywanych w konkretnej tabeli danych.

Typy znakowe

Typ	Rozmiar w bajtach
CHAR(x)	Pole tekstowe o stałej długości znaków, może przechowywać od 0 do 255 znaków.
VARCHAR(x)	Pole tekstowe o zmiennej długości, od 0 do 255znaków.
TINYBLOB, TINYTEXT	Łańcuch od 0 do 255 znaków.
BLOB, TEXT	Łańcuch od 0 do 65 535 znaków.
MEDIUMBLOB, MEDIUMTEXT	Łańcuch od 0 do 16 777 215 znaków.
LONGBLOB, LONGTEXT	Łańcuch od 0 do 4 294 967 295 znaków.

Typy daty i czasu

Typ	Rozmiar w bajtach
DATE	3
DATETIME	8
TIMESTAMP	4
TIME	3
YEAR	1

Typy numeryczne

Typ	Rozmiar w bajtach	Wartość minimalna	Wartość maksymalna
TINYINT (ze znakiem)	1	–128	127
TINYINT (bez znaku)	1	0	255
SMALLINT (ze znakiem)	2	–32768	32767
SMALLINT (bez znaku)	2	0	65535
MEDIUMINT (ze znakiem)	3	-8388608	8388607

MEDIUMINT (bez znaku)	3	0	16777215
INT (ze znakiem)	4	–2147483648	2147483647
INT (bez znaku)	4	0	4294967295
BIGINT (ze znakiem)	8	–9223372036854775808	9223372036854775807
BIGINT (bez znaku)	8	0	18446744073709551615
BIT (x)	około (x+7)/8	wyzerowane wszystkie bity	ustawione wszystkie bity
DECIMAL(s,p), NUMERIC(s,p)		–1038–1	1038–1
FLOAT	4	1,79E+308	1,79E+308
DOUBLE [PRECISION], REAL	8	–3,40E+38	3,40E+38

Przykład: Aby utworzyć tabelę o nazwie Miasta zawierającą pola id (liczba całkowita + autonumerowanie), Name (tekst do 35 znaków), CountryCode (kod państwa), District (tekst do 20 znaków), Powierzchnia (liczba całkowita), Population (liczba całkowita) wykonujemy poniższy kod.

```
CREATE TABLE `Miasta` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `Name` varchar(35),
  `CountryCode` char(3),
  `District` varchar(20),
  `Powierzchnia` int(11),
  `Population` int(11),
  PRIMARY KEY (`ID`)
)DEFAULT CHARSET=cp1250;
```

NOT NULL powoduję, że dane pole nie może być puste.

3) Modyfikowanie tabel

Raz utworzone tabele mogą być zmieniane — zawsze możemy dodać lub usunąć kolumnę, możemy również zmodyfikować istniejące kolumny, czy zmienić nazwę tabeli lub poszczególnych kolumn. Wszystkie te zmiany możemy przeprowadzić, wykonując instrukcję **`ALTER TABLE`**.

Przykład: Najpierw usuwamy jedną kolumnę, postem zmieniamy typ i nazwę pozostałej kolumny, a na końcu nazwę całej tabeli

```
ALTER TABLE miasta DROP COLUMN Powierzchnia;
```
usuwa pole Powierzchnia

```
ALTER TABLE miasta CHANGE name varchar(40);
```
zmienia typ pola name na varchar(40)

```
ALTER TABLE miasta RENAME TO City;
```
zmienia nazwę tabeli na City

4) Usuwanie tabel

Aby usunąć tabelę, należy wykonać instrukcję *`DROP TABLE`*

Ćwiczenie 2. Utwórz, a następnie usuń tabelę *miasteczka*.

```
CREATE TABLE miasteczka (id int(3));
DROP TABLE miasteczka;
```

5) Indeksy

Jedynym powodem tworzenia indeksów jest poprawa wydajności bazy danych. Indeksy znacznie wpływają na czas wykonania instrukcji. Jeżeli nie istnieją indeksy, wyszukanie pojedynczej wartości wiąże się z koniecznością odczytania całej tabeli. Jeżeli natomiast istnieją powiązane z tabelą indeksy, znalezienie żądanych danych sprowadza się do znalezienia w indeksie kluczy spełniających podane kryteria i odczytania wyłącznie tych wierszy tabeli, na które wskazują znalezione klucze indeksu.

Aby utworzyć indeks użyjemy polecenia:

```
 ALTER TABLE nazwa_tabeli ADD INDEX (kolumna);
```

Przykład: Aby utworzyć indeks dla kolumny **miasto** ***tabeli*** **City**, ***napiszemy***

```
ALTER TABLE `city` ADD INDEX (`name`);
```

MySQL umożliwia również tworzenie indeksów podczas tworzenia tabel. W takim przypadku po podaniu typu kolumny należy użyć słowa kluczowego INDEX, ewentualnie określić typ indeksu i wskazać indeksowaną kolumnę.

6) Import danych.

Pobierz plik tabele.zip i rozpakuj na dysk. W linii poleceń mysql wykonaj polecenia

```
source d:\city.sql
source d:\countrysql
source d:\countrylanguage.sql
```

Oczywiście zamiast d:\ proszę wstawić scieżkę dostępu do plikow w swoim systemie.

2. *Połączenie z serwerem*

Aby połaczyć się z bazą danych musimy uruchomić system do zarządzania relacyjnymi bazami danych – w naszym przypadku jest to MYSQL i połączyć się z serwerem.
Korzystać z mysqla będziemy w linii poleceń cmd.
Aby połączyć się z serwerem wykonujemy poniższe polecenie.

```
W:\>mysql -h 172.28.30.250 -u labor -p
Enter password: *****
```

W tym przypadku korzystamy z kilku parametrów:
-h – połączenie z serwerem o nazwie elmo (zamiast nazwy może być oczywiście adres IP)
-u – user: nazwa użytkownika bazy
-p – password: kiedy chcemy podać hasło
Po zalogowaniu pojawi nam się znak zachęty **`mysql>`**

3. *Otwarcie i przeglądanie bazy.*

Aby zobaczyć bazy jakie mamy na serwerze użyjemy polecenia **show** databases

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| bank_innodb        |
| bank_myisam        |
| test               |
| world              |
+--------------------+
```

Baza, z którą będziemy pracowali na zajęciach nazywa się ***world***. Aby zobaczyć, co zawiera musimy się do niej podłączyć poleceniem **use**.

```
mysql>use word;
```

Do wyświetlenia tabel bazy użyjemy znów polecenia **show**.

```
mysql> show tables;
+-----------------+
| Tables_in_world |
+-----------------+
| City            |
| Country         |
| CountryLanguage |
+-----------------+
```

4. *Odczytywanie danych z tabel.*

1) Polecenie SELECT

Do pobierania danych z tabel służy polecenie SELECT.

```
SELECT <lista_pól> FROM <nazwa_bazy>.<nazwa_tabeli>;
```

lub

```
SELECT <lista_pól> FROM <nazwa_tabeli>;
```

np.

```
SELECT * FROM City;
```

Otrzymamy długą tabelę, z której trudno wybrać interesujące dane.

```
.
| 4074 | Gaza          | PSE  | Gaza          |   353632 |
| 4075 | Khan Yunis    | PSE  | Khan Yunis    |   123175 |
| 4076 | Hebron        | PSE  | Hebron        |   119401 |
| 4077 | Jabaliya      | PSE  | North Gaza    |   113901 |
| 4078 | Nablus        | PSE  | Nablus        |   100231 |
| 4079 | Rafah         | PSE  | Rafah         |    92020 |
+------+---------------+------+---------------+----------+
```

2) Klauzula WHERE

Możemy jednak stosując odpowiednie wyrażenia zawęzić wyniki zapytania do tych, które nas interesują. Możemy zastosować klauzulę WHERE, w której możemy określić które rekordy chcemy wyświetlić

Ćwiczenie 3. Wyświetl pierwsze dziesięć rekordów.

```
SELECT * FROM City WHERE id<10;
+----+----------------+-------------+-------------+-----------+
| ID | Name           | CountryCode| District     | Population|
+----+----------------+-------------+-------------+-----------+
|  1 | Kabul          | AFG        | Kabol        |    1780000|
|  2 | Qandahar       | AFG        | Qandahar     |     237500|
|  3 | Herat          | AFG        | Herat        |     186800|
|  4 | Mazar-e-Sharif | AFG        | Balkh        |     127800|
|  5 | Amsterdam      | NLD        | Noord-Holland|     731200|
|  6 | Rotterdam      | NLD        | Zuid-Holland |     593321|
|  7 | Haag           | NLD        | Zuid-Holland |     440900|
|  8 | Utrecht        | NLD        | Utrecht      |     234323|
|  9 | Eindhoven      | NLD        | Noord-Brabant|     201843|
+----+----------------+------------+--------------+-----------+
```

Podobny efekt uzyskamy stosując klauzulę **BETWEEN**

```
SELECT * FROM City WHERE id BETWEEN 1 AND 10;
```

Ćwiczenie 4. Wypisz wszystkie miasta polskie.

3) Sortowanie wyników

Do sortowania wyników służy klauzula ORDER BY występująca na końcu zapytania SELECT. Posiada ona dwa specyfikatory ASC (rosnąco -domyślny) i DESC (malejąco).

Ćwiczenie 5. W powyższym przykładzie wypisz dane rosnąco wg. pola Name.

```
SELECT * FROM City WHERE id BETWEEN 1 AND 10 ORDER BY Name ASC;
```

Ćwiczenie 6.Wyświetl państwa w których średnia długość zycia mieszkańców jest większa od 80(%). Wyświetl tylko pola Name, Continent i LifeExpectancy.

4) Ograniczanie liczby wierszy - LIMIT

Język SQL pozwala również ograniczyć liczbę wierszy wyniku zapytania — wystarczy w klauzuli **LIMIT** wpisać odpowiednią liczbę. W efekcie bardzo łatwo możemy np. odczytać nazwy pięciu najbardziej zaludnionych miast.

```
SELECT * FROM City ORDER BY Population DESC LIMIT 3;
+------+-----------------+-------------+-------------+------------+
| ID   | Name            | CountryCode | District    | Population |
+------+-----------------+-------------+-------------+------------+
| 1024 | Mumbai (Bombay) | IND         | Maharashtra |   10500000 |
| 2331 | Seoul           | KOR         | Seoul       |    9981619 |
```

```
|  206 | SŃo Paulo       | BRA         | SŃo Paulo   |    9968485 |
+------+-----------------+-------------+-------------+------------+
```

Ćwiczenie 7. Wyświetl 10 państw w których średnia długość zycia mieszkańców jest największa.

5) Warunki logiczne AND, OR i NOT

Możemy łączyć warunki stosując operatory **OR** i **AND** .

```
SELECT * FROM City
WHERE CountryCode=”POL” AND District=”Malopolskie”;
```

```
+------+--------+-------------+-------------+------------+
| ID   | Name   | CountryCode | District    | Population |
+------+--------+-------------+-------------+------------+
| 2930 | Krakow | POL         | Malopolskie |     738150 |
| 2962 | Tarnow | POL         | Malopolskie |     121494 |
+------+--------+-------------+-------------+------------+
```

Ćwiczenie 8. Wypisz wszystkie miasta polskie i czeskie.

Zobacz, co uzyskasz wykonując poniższy kod.

```
SELECT * FROM City
WHERE District=”Malopolskie” OR District=”Slaskie”;
```

To samo możemy uzyskać stosując wyrażenie **IN**

```
SELECT * FROM City
WHERE District IN(”Malopolskie”, ”Slaskie”);
```

Wyrażenie **NOT** neguje wyrażenia IN i BETWEEN. Poniższe polecenie wyświetli nam wszystkie miasta oprócz powyższych.

```
SELECT * FROM City
WHERE CountryCode=”POL” AND District NOT IN(”Malopolskie”, ”Slaskie”);
```

5. *DODATEK: Nazwy pól w tabelach bazy Word.*

City (`ID`, `Name`, `CountryCode`, `District`, `Population`)

Country (`Code`, `Name`, `Continent`, `Region`, `SurfaceArea`, `IndepYear`, `Population`, `LifeExpectancy`, `GNP`, `GNPOld`, `LocalName`, `GovernmentForm`, `HeadOfState`, `Capital`, `Code2`)

CountryLanguage (`CountryCode, `Language`, `IsOfficial`, `Percentage`)