BAZY DANYCH

Lab. 2 DDL/DML

Laboratorium nr 2

Temat: DDL/DML

Polecenia DDL

Użycie SQL, zgodnie z jego nazwą, polega na zadawaniu zapytań do bazy danych. Zapytania można zaliczyć do jednego z trzech głównych podzbiorów:

- **SQL DML** (ang. *Data Manipulation Language* – „język manipulacji danymi”),
- **SQL DDL** (ang. *Data Definition Language* – „język definicji danych”),
- **SQL DCL** (ang. *Data Control Language* – „język kontroli nad danymi”).

Instrukcje SQL w obrębie zapytań tradycyjnie zapisywane są wielkimi literami, jednak nie jest to wymóg. Każde zapytanie w SQL-u musi kończyć się znakiem średnika (;).

DDL

Dzięki DDL (Data Definition Language) można operować na strukturach, w których dane są przechowywane – czyli np. dodawać, zmieniać i kasować tabele lub bazy. Najważniejsze polecenia tej grupy to:

- **CREATE** (np. CREATE TABLE, CREATE DATABASE, ...) – utworzenie struktury (bazy, tabeli, indeksu itp.),
- **DROP** (np. DROP TABLE, DROP DATABASE, ...) – usunięcie struktury,
- **ALTER** (np. ALTER TABLE ADD COLUMN ...) – zmiana struktury (dodanie kolumny do tabeli, zmiana typu danych w kolumnie tabeli).

AUTOMATYCZNA INKREMENTACJA

Możliwość automatycznej inkrementacji jest zaletą MySQLa. Gdy dodajemy dane do tabeli i chcemy, aby numerowane były automatycznie (1,2,3…)

PRZYKŁAD:

```
CREATE TABLE Pracownicy
(
ID_Pracownicy INTEGER not null auto_increment,
imie VARCHAR(20),
nazwisko VARCHAR (30),
adres VARCHAR (70),
email VARCHAR (20) default 'brak',
primary key (ID_Pracownik)
);
```

WARTOŚĆ DEFAULT

Dla każdego z pól rekordu można określić wartości domyślne, które będą wstawiane do pól, gdy jawnie nie określimy wartości.

Własność tę można wprowadzić na dwa sposoby:

1. na etapie definiowania struktury tabeli:

```
CREATE TABLE Pracownicy
(
ID_Pracownicy INTEGER not null auto_increment,
imie VARCHAR(20),
nazwisko VARCHAR (30),
adres VARCHAR (70),
email VARCHAR (20) default 'brak',
primary key (ID_Pracownik)
);
```

2. modyfikując już istniejącą strukturę:

   ALTER TABLE Pracownicy MODIFY email VARCHAR(20) DEFAULT 'brak';

EDYCJA TABELI

Po utworzeniu tabeli można jej strukturę poddać edycji.

Ogólna formuła instrukcji ALTER TABLE jest następująca

```
ALTER TABLE nazwa_tabeli określenie_zmian;
```

```
Określenie_zmian:
ADD [COLUMN] definicja_tworzenia [FIRST | AFTER nazwa_kol]
ADD [COLUMN] (definicja_tworzenia, definicja_tworzenia, …)
ADD INDEX [nazwa_indeksu](nazwa_indexu, nazwa_indexu_kol, …)
ADD PRIMARY KEY (nazwa_indexu_kol, …)
ADD UNIQUE (nazwa_indexu, nazwa_indexu_kol, …)
ADD FULLTEXT (nazwa_indexu, nazwa_indexu_kol, …)
ADD [CONSTRAINT symbol] FOREIGN KEY  [nazwa_indexu
(nazwa_indexu, nazwa_indexu_kol, …) [definicja_odwołania]
ALTER [COLUMN] nazwa_kol {SET DEFAUL litera | DROP DEFAULT }
CHANGE [COLUMN]dawna_nazwa_kol definicja_tworzenia [FIRST |
AFTER nazwa_kol]
MODIFY
DROP [COLUMN] nazwa_kol
DROP PRIMARY KEY
DROP INDEX nazwa_indexu
DISABLE KEYS
RENAME [TO] nowa_nazwa_tab
ORDER BY nazwa_kol
```

Większość opcji ma dosłowne znaczenie.

Wyrażenia CHANGE i MODIFY oznaczają to samo. Pozwalają zmienić definicję kolumny lub jej pozycje w tabeli. Przy zmianie jedynie właściwości pola posługujemy się komendą MODIFY, a zmianie zarówno właściwości jak i nazwy pola - CHANGE

DROP COLUMN powoduje usunięcie kolumny z tabeli, a DROP PRIMARY KEY oraz DROP INDEX powodują usunięcie indeksu związanego z dana kolumną.

Wyrażenie DISABLE KEY powoduje zaprzestanie uaktualniania indeksów; uaktualnianie indeksów można przywrócić za pomocą wyrażenia ENABLE KEYS.

RENAME zmienia nazwę tabeli

Wyrażenie ORDER BY sprawi, ze do zmienianej tabeli wiersze będą wstawiane w określonej kolejności. Ta kolejność nie zostanie jednak utrzymana, gdy dane będą się zmieniały w czasie.

Przykład

Zmiana struktury istniejącej tabeli: dodanie indeksu nazwisko do tabeli pracownicy

```
ALTER TABLE Pracownicy ADD INDEX nazwisko nazwisko(nazwisko);
```

Przykład
Zmiana struktury istniejącej tabeli: dodanie kolumny płeć do tabeli studenci (za kolumna nazwisko)

```
ALTER TABLE pracownicy ADD email VARCHAR(20) AFTER nazwisko;
```

ZRZUT BAZY

Jedną z ważnych czynności administracyjnych przy używaniu baz danych jest zabezpieczanie ich zawartości. Najprostsza metoda zabezpieczenia zawartości polega na wykonaniu zrzutu struktur i danych do pliku sql i zabezpieczenie jego zawartości na dodatkowym nośniku.

Zrzutu dokonuje się następującą komendą:

```
mysqldump –h host –u identyfikator –p nazwa_bazy > nazwa_plik.sql
```

Domyślnie plik zapisuje się w katalogu c:\mysql\bin. Zawiera on komendy SQL definiujące strukturę i ładujące dane do bazy.

W celu wykonania tej operacji należy uruchomić okno z wierszem poleceń i przejść do katalogu c:\mysql\bin. Po wpisaniu komendy program poprosi o wpisanie hasła dla użytkownika, którego dane podajemy po opcji –u.

Ćwiczenie

1. Proszę utworzyć nową bazę danych o nazwie składającej się z imienia i pierwszej litery nazwiska, a następnie ustawić nową bazę danych na aktywną;
2. Proszę utworzyć tabelę w nowej bazie na podstawie podanych niżej danych (proszę zrobić zrzut ekranu z utworzoną tabelą)

```
Pracownik
    ID INTEGER, automatyczna inkrementacja, wartość niepusta
    Nazwisko VARCHAR (20)
    Imie VARCHAR (20)
    E_mail VARCHAR (20)
    Wynagrodzenie DECIMAL(10,2)
    Premia DECIMAL(10,2) DEFAULT 0
    Data_urodzenia DATE
    PESEL VARCHAR(11)
    Data _zatrudnienia DATE
```

3. Zmień nazwę tabeli na Uzytkownik
4. Dodaj do tabeli na początek (przed ID) nową kolumnę Kraj z domyślną wartością ‘Polska’
5. Dodaj do tabeli nową kolumnę TEL (po kolumnie E_mail);
6. Zmień nazwę E_mail na email
7. Dodaj po kolumny wynagrodzenie wartość domyślną ‘2000’ zł
8. Proszę zrobić zrzut bazy do pliku na dysku (w formie pliku SQL)

Polecenia DML

DML

DML (Data Manipulation Language) służy do wykonywania operacji na danych – do ich umieszczania w bazie, kasowania, przeglądania, zmiany. Najważniejsze polecenia z tego zbioru to:

- **SELECT** – pobranie danych z bazy,
- **INSERT** – umieszczenie danych w bazie,
- **UPDATE** – zmiana danych,
- **DELETE** – usunięcie danych z bazy.

Dane tekstowe muszą być zawsze ujęte w znaki pojedynczego cudzysłowu (').

Polecenie SELECT zostanie omówione dokładnie na kolejnych laboratoriach.

1. DODAWANIE REKORDÓW - INSERT

Składnia tego polecenia przedstawia się następująco:

```
INSERT INTO {nazwa_tabeli} VALUES()
```

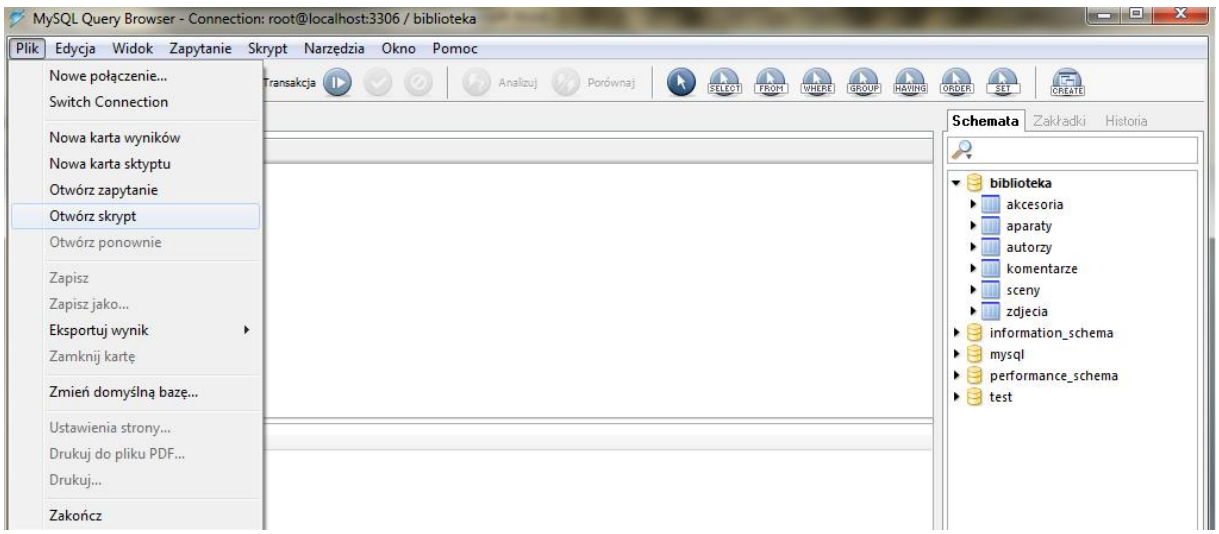
W nawiasach okrągłych po słowie kluczowym **VALUES** wypisujemy listę wartości dla pól rekordu, które zamierzamy podać, oddzielone przecinkami.

Np.

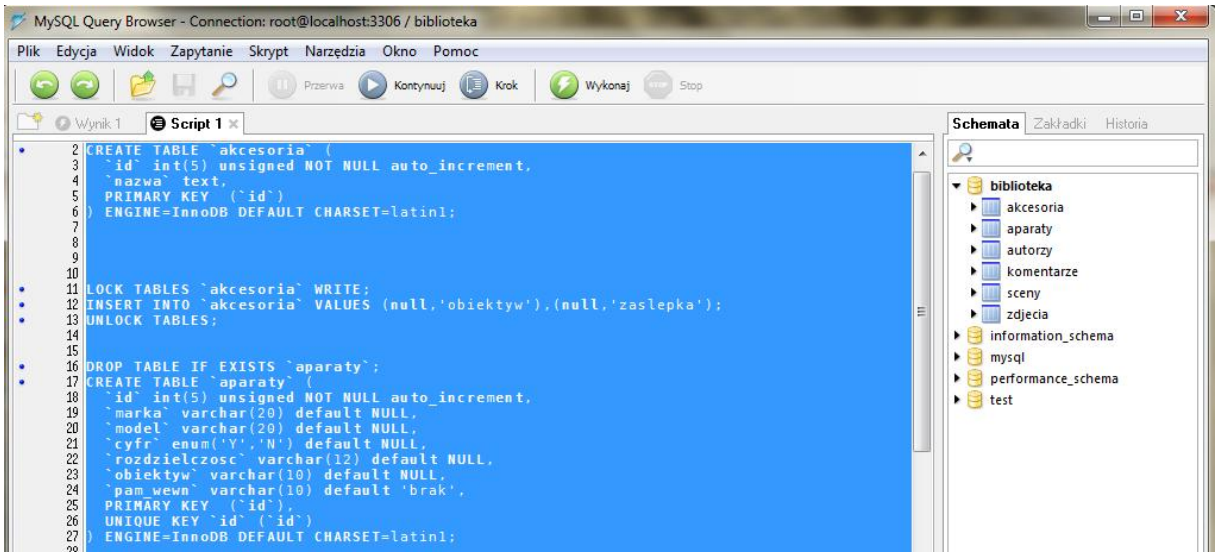
```
INSERT INTO egzaminatorzy VALUES (

'0001','Popiolek','Marek','20618','Biala_Podlaska','Sidorska',
'102','833449902','833449901', 'marekp@psw.bialapodlaska.pl');
```

Zamiast wpisywać kolejne numery **id_pracownik**, możemy podawać za każdym razem null. Pole **id_pracownik** jest typu **auto_increment** i nie ma znaczenia, co do niego wpisujemy (i tak będzie miało odpowiednio inkrementowaną wartość). Dlatego lepiej jest dodawać rekordy bez podawania pola **id**.

ŁADOWANIE DANYCH DO BAZY ZA POMOCA SKRYPTU (MySQL Query Browser)

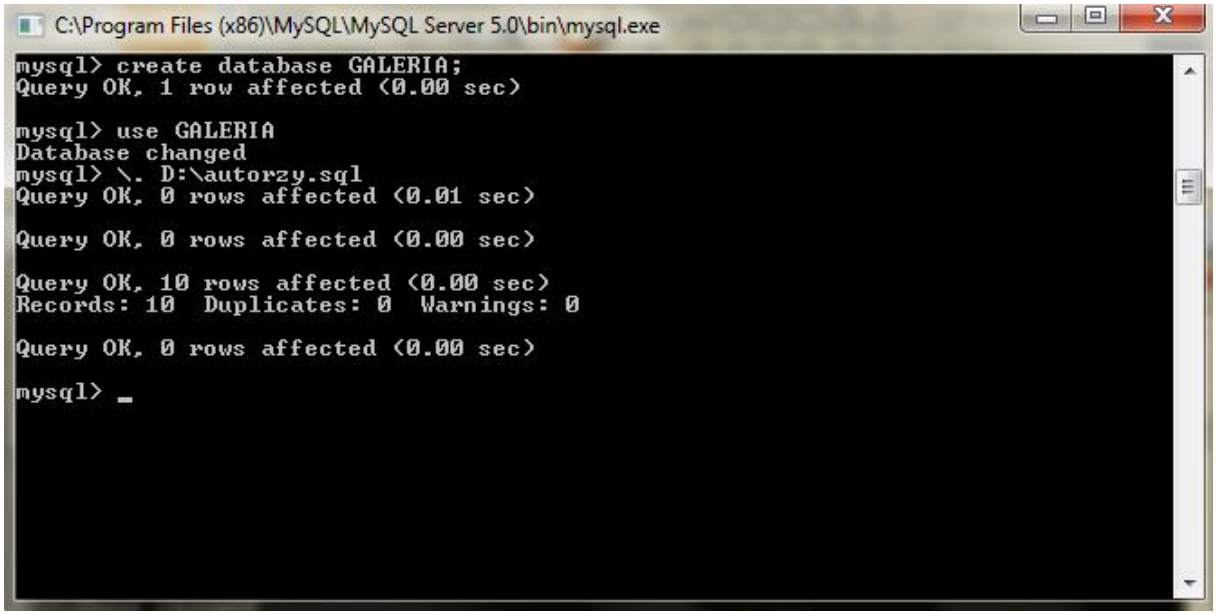


Wykorzystując opcję Plik – Otwórz skrypt – pobieramy plik sql lub txt. Następnie należy wybrać odpowiedni plik i automatycznie załaduje się jego kod źródłowy. Ponieważ w skrypcie istnieje wiele instrukcji SQL oddzielonych średnikami, aby wykonać cały skrypt należy zaznaczyć go i dopiero użyć opcji  'Wykonaj'



WYKORZYSTANIE DYREKTYWY SOURCE DO ŁADOWANIA DANYCH DO TABELI

- Przygotowanie pliku z komendami INSERT i danymi
- Załadowanie do bazy instrukcją: `\. nazwa_pliku.sql`
  (polecenie to działa z wiersza poleceń MySQL)



ŁADOWANIE DANYCH MECHANIZMEM LOAD DATA

- Przygotowanie pliku w formacie z tabulatorem: pola puste oznaczamy, wpisując \N
- Ładowanie do bazy odbywa się za pomocą instrukcji:

LOAD DATA LOCAL INFILE 'PLIK.SQL' INTO TABLE TABELA;

2. DODAWANIE REKORDÓW – REPLICE

Instrukcja REPLACE działa podobnie jak instrukcja INSERT. Różni się od niej wyłącznie tym, że gdy przy wstawianiu wartości pojawi się kolizja kluczy, nowy wiersz zastąpi istniejący

3. WYBIERANIE REKORDÓW – SELECT

Polecenie SELECT służy do wybierania w rozmaity sposób rekordów z bazy danych. Podstawowa składnia tego polecenia wygląda następująco:

```
SELECT lista_pól FROM nazwa_tabeli;
```

Znak * użyty zamiast listy pól oznacza, że chcemy wybrać wszystkie dostępne pola tabeli.

Możesz oczywiście pobrać tylko jedną kolumnę, na przykład tę zawierającą numery id_pracownik. Polecenie SELECT zostanie szerzej omówione na kolejnych ćwiczeniach.

4. MODYFIKACJA REKORDÓW - UPDATE

Modyfikacji rekordów dokonuje się za pomocą polecenia **UPDATE**. Składnia tego polecenia przedstawia się następująco:

```
UPDATE {nazwa_tabeli} SET {nazwa_pola} = 'wartosc' WHERE {warunek}
```

Np. Poprawienie wartości pola stanowisko na „ADMINISTRATOR_BAZ_DANYCH” dla Id_pracownik=586

```
UPDATE pracownik
SET stanowisko = ADMINISTRATOR_BAZ_DANYCH
WHERE Id_pracownik=586;
```

5. USUWANIE REKORDÓW - DELETE

Do usuwania rekordów służy polecenie **DELETE**

```
DELETE FROM {nazwa_tabeli} WHERE {warunek}
```

Np.

```
DELETE FROM pracownik;
```

Spowoduje usunięcie wszystkich wierszy z tabeli pracownik.

```
DELETE FROM pracownik WHERE nazwisko =’Kowalski’;
```

Spowoduje usuniecie pracownika o nazwisku Kowalski

Jeżeli warunek **WHERE** nie zostanie podany, zawartość całej tabeli zostanie usunięta.

Wartości numerów **id** nie przeindeksowują się (pole **auto_increment**) a następny numer **id**, jaki zostanie dodany będzie miał wartość o jeden większą od największego.

6. USUWANIE REKORDÓW – TRUNCATE

Instrukcja TRUNCATE umożliwia usunięcie wszystkich wierszy w tabeli np.

```
TRUNCATE TABLE pracownik;
```

To zapytanie usunęłoby wszystkich pracowników z tabeli pracownik. Jest ono szybsze niż instrukcja DELETE, ponieważ powoduje usunięcie tabeli i utworzenie nowej – pustej. Instrukcja ta nie zapewnia jednak bezpieczeństwa właściwego dla transakcji.

Zadania

Zadanie 1

Proszę utworzyć sprawozdanie z laboratorium w formie pliku pdf. Plik ma zawierać instrukcje i polecenia języka SQL wykonane w trakcie realizacji zadań.

1. Proszę pobrać ze strony internetowej prowadzącego zajęcia plik biblioteka1.sql i wgrać do bazy.
2. Proszę sprawdzić jakie tabele są w bazie
3. [INSERT] Do każdej tabeli należy dodać dane (po 5 wierszy do każdej tabeli). Dane maja być prawidłowe – zgodne z rzeczywistością.
4. [SELECT] Należy sprawdzić jakie dane są w tabelach.
5. [DELETE] Proszę usunąć z tabeli pracownicy osobę, która pracuj najkrócej.
6. [UPDATE] Proszę dokonać zmian w bazie danych:
    a. Zmienić nazwisko pracownika o identyfikatorze 3 na ‘Mikolajczuk’
    b. Dokonać zmiany daty zwrotu książek o identyfikatorach 1,2,4 na dzisiejszą datę
    c. Zmienić stanowisko pracownika o identyfikatorze 1 na ‘kierownik’
    d. Zanotować w bazie zdarzenie: Pracownik o identyfikatorze 4 dokonał dzisiaj wypożyczenia najnowszej w bazie książki, czytelnikowi o

identyfikatorze 1

Zadanie 2

POWTÓRZENIE DO KOLOKWIUM

1. Utwórz bazę danych GALERIA
2. Utwórz plik w Notatniku, który pozwoli załadować do tabeli autorzy dane – min 10 osób. Wczytaj go do bazy. Pamiętaj o automatycznej inkrementacji

```
DROP TABLE IF EXISTS autorzy;

CREATE TABLE autorzy (

  id int(5) unsigned NOT NULL auto_increment,

  imie varchar(20) default NULL,

  nazwisko varchar(30) default NULL,

  plec enum('K','M') default NULL,

  email varchar(20) default 'brak',

  PRIMARY KEY  (id));

INSERT INTO autorzy VALUES
(null,'Piotr','Kot','M','kot@wp.pl'), (null,……
```

3. Pobierz plik galeria.sql
4. Załaduj plik do bazy
5. Sprawdź czy w bazie jest tabela autorzy
6. Sprawdź poprawność danych za pomocą instrukcji SELECT
7. Obejrzyj strukturę tabeli aparaty. (DESC)
8. Zmodyfikuj tabelę aparaty (MODYFI COLUMN) tak, aby marka aparatu uzyskała atrybut NOT NULL
9. Sprawdź co się zmieniło w strukturze tabeli aparaty. Gdzie widoczna jest ta zmiana?
10. Wykonaj instrukcję INSERT na tabeli aparaty. Dodaj nowe aparat marki SONY, CANON, KODAK, NIKON (poprze instrukcje Insert lub plik SQL)
11. Dodaj do tabeli zdjęcia nowe wiersze wg instrukcji prowadzącego