

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

**Інститут комп'ютерних наук та інформаційних технологій**

**Кафедра систем штучного інтелекту**



**Звіт до лабораторної роботи №2**

з дисципліни  
“ОБДЗ”

**Виконав:**  
ст. гр. КН-211  
Ільків Андрій

**Викладач:**  
Якимишин Х.М.

Львів – 2019

## Лабораторна робота №2

**Мета роботи:** Побудувати даталогічну модель бази даних; визначити типи, розмірності та обмеження полів; визначити обмеження таблиць; розробити SQL запити для створення спроектованих таблиць.

### Короткі теоретичні відомості

Щоб створити нову базу даних у командному рядку клієнта MySQL (mysql.exe) слід виконати команду CREATE DATABASE, опис якої подано нижче. Тут і надалі, квадратні дужки позначають необов'язковий аргумент команди, символ "|" позначає вибір між аргументами.

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] ім'я_бази  
[[DEFAULT] CHARACTER SET кодування] [[DEFAULT] COLLATE  
набір_правил]
```

ім'я\_бази – назва бази даних (латинські літери і цифри без пропусків);  
кодування – набір символів і кодів (koі8u, latin1, utf8, cp1250 тощо);  
набір\_правил – правила порівняння рядків символів (див. результат команди show collation).

Нижче наведені деякі допоміжні команди для роботи в СУБД MySQL. Кожна команда і кожен запит в командному рядку повинні завершуватись розділяючим символом ";".

1. Перегляд існуючих баз даних: SHOW DATABASES
2. Вибір бази даних для подальшої роботи: USE DATABASE ім'я\_бази
3. Перегляд таблиць в базі даних: SHOW TABLES [FOR ім'я\_бази]
4. Перегляд опису таблиці в базі: DESCRIBE ім'я\_таблиці
5. Виконати набір команд з зовнішнього файлу: SOURCE назва\_файлу
6. Вивести результати виконання подальших команд у зовнішній файл: \T назва\_файлу

Для роботи зі схемою бази даних існують такі основні команди: ALTER DATABASE – зміна опису бази даних; CREATE TABLE – створення нової таблиці; ALTER TABLE – зміна структури таблиці; DELETE TABLE – видалення таблиці з бази даних; CREATE INDEX – створення нового індексу (для швидкого пошуку даних); DROP INDEX – видалення індексу; DROP DATABASE – видалення бази даних.

Розглянемо команду створення таблиці в MySQL та її основні аргументи.

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] ім'я_таблиці  
    [(опис_таблиці,...)]  
    [додаткові_параметри] ...  
    [вибірка_даних]
```

**опис\_таблиці:**

назва\_поля опис\_поля

```
| [CONSTRAINT [ім'я_обмеження]] PRIMARY KEY (назва_поля,...)  
[тип_обмеження]  
| {INDEX|KEY} [ім'я_обмеження] (назва_поля,...) [тип_обмеження]  
| [CONSTRAINT [ім'я_обмеження]] UNIQUE [INDEX|KEY]  
[ім'я_обмеження](назва_поля,...) [тип_обмеження]  
| {FULLTEXT|SPATIAL} [INDEX|KEY] [ім'я_обмеження] (назва_поля,...)  
[тип_обмеження]  
| [CONSTRAINT [ім'я_обмеження]] FOREIGN KEY [ім'я_обмеження]  
(назва_поля,...) опис_зв'язку  
| CHECK (вираз)
```

**опис\_поля:**

```
тип_даних [NOT NULL | NULL] [DEFAULT значення_за_замовчуванням]  
[AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
```

**опис\_зв'язку:**

```
REFERENCES ім'я_таблиці (назва_поля, ...)  
    [ON DELETE дія]  
    [ON UPDATE дія]
```

**дія:**

CASCADE

Одночасне видалення, або оновлення відповідного значення у зовнішній таблиці.

RESTRICT

Аналог NO ACTION. Дія над значенням поля ігнорується, якщо існує відповідне йому значення у зовнішній таблиці. Опція задана за замовчуванням.

## SET NULL

При дії над значенням у первинній таблиці, відповідне значення у зовнішній таблиці замінюється на NULL.

### **додаткові параметри:**

{ENGINE|TYPE} [=] тип\_таблиці  
| AUTO\_INCREMENT [=] значення\_приросту\_лічильника  
| AVG\_ROW\_LENGTH [=] значення  
| [DEFAULT] CHARACTER SET [=] кодування  
| CHECKSUM [=] {0 | 1}  
| [DEFAULT] COLLATE [=] набір\_правил  
| COMMENT [=] 'коментар до таблиці'  
| DATA DIRECTORY [=] 'абсолютний шлях'  
| DELAY\_KEY\_WRITE [=] {0 | 1}  
| INDEX DIRECTORY [=] 'абсолютний шлях'  
| MAX\_ROWS [=] значення | MIN\_ROWS [=] значення  
| ROW\_FORMAT {DEFAULT|DYNAMIC|FIXED|COMPRESSED|REDUNDANT|COMPACT}

### **вибірка даних:**

[IGNORE | REPLACE] [AS] SELECT ... (вибір даних з інших таблиць)

### **вираз:**

Логічний вираз, що повертає TRUE або FALSE.



Можна дати декілька порад щодо розробки схеми бази даних і вибору типів даних. Вони дозволять уникнути повільного виконання запитів і потреби модифікації таблиць в майбутньому.


- Слід використовувати якомога менший тип даних для полів таблиць. Наприклад, для зберігання чисел від 1 до 64 краще використати тип TINYINT(6) замість SMALLINT. Це впливає на швидкість пошуку і вибірки даних.

- Слід використовувати рядки фіксованої довжини, якщо це можливо. Для цього всі поля таблиці повинні бути фіксованої довжини. Тобто, варто уникати типів VARCHAR, TEXT і BLOB. Це пришвидчить вибірку даних з середини рядків, оскільки ці дані будуть мати фіксовану адресу. При потребі використання полів з типами TEXT або BLOB, їх можна виділити в окрему таблицю.
- Якщо можливо, варто завжди використовувати поля з обмеженням NOT NULL. Хоча це може збільшувати об'єм бази на диску.
- MySQL дозволяє використовувати різні типи таблиць в одній базі даних. Слід використовувати переваги різних типів (MyISAM, INODB тощо) залежно від характеру майбутнього використання таблиці.
- Потрібно створювати індекси, які пришвидчать пошук і вибірку даних.
- В рідкісних випадках можна денормалізувати схему з метою зменшення кількості операцій з об'єднання таблиць при складних запитах. Але при цьому ускладнюється задача збереження цілісності бази даних.



### **Хід роботи:**



Даталогічна модель вимагає визначення конкретних полів бази даних, їхніх типів, обмежень на значення, тощо. На рисунку зображено даталогічну модель проектованої бази даних.



| Student   |            |              |
|---|------------|--------------|
|                            | student_id | int          |
|   | first_name | varchar(255) |
|   | last_name  | varchar(255) |
|   | group      | binary       |
|  <a href="#">Add field</a> |            |              |

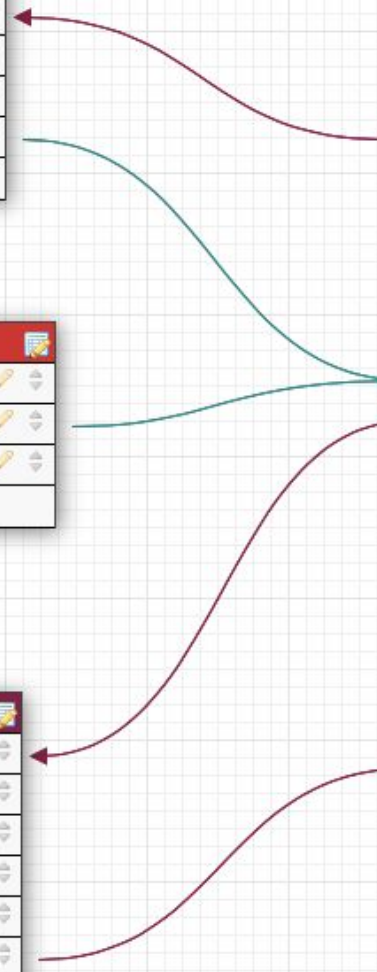
| StudentInfo  |                       |              |
|--|-----------------------|--------------|
|  | identification_number | varchar(255) |
|  | civil_number          | varchar(255) |
|  | medical_info          | varchar(255) |
|  | student               | int(255)     |
|  <a href="#">Add field</a> |                       |              |

| Schedule  |                    |              |
|---|--------------------|--------------|
|   | lection_time       | time         |
|   | group              | int          |
|   | section_decription | varchar(255) |
|  <a href="#">Add field</a> |                    |              |

| Group  |            |              |
|--|------------|--------------|
|                            | group_id   | int          |
|  | teacher_id | int          |
|  | lectionId  | int          |
|  | category   | varchar(255) |
|  <a href="#">Add field</a> |            |              |

| Teacher   |            |              |
|---|------------|--------------|
|                             | teacher_id | int          |
|   | first_name | varchar(255) |
|   | last_name  | varchar(255) |
|   | email      | varchar(255) |
|   | phone      | varchar(255) |
|   | car_id     | int          |
|  <a href="#">Add field</a> |            |              |

| Car  |          |              |
|--|----------|--------------|
|                             | car_id   | int          |
|  | model    | varchar(255) |
|  | category | varchar(255) |
|  <a href="#">Add field</a> |          |              |



Створимо нову базу даних, виконавши такі команди:

```
CREATE DATABASE Bass;
```

```
USE Bass;
```

```
CREATE TABLE `Car` (  
    `car_id` INT NOT NULL AUTO_INCREMENT,  
    `model` varchar(255) NOT NULL,  
    `category` varchar(255) NOT NULL,  
    PRIMARY KEY (`car_id`)  
);
```

```
CREATE TABLE `Teacher` (  
    `teacher_id` INT NOT NULL,  
    `first_name` varchar(255) NOT NULL,  
    `last_name` varchar(255) NOT NULL,  
    `email` varchar(255) NOT NULL,  
    `phone` varchar(255) NOT NULL,  
    `car_id` INT NOT NULL,  
    PRIMARY KEY (`teacher_id`),  
    CONSTRAINT `Teacher_fk0` FOREIGN KEY (`car_id`) REFERENCES `Car`(`car_id`) ON  
DELETE CASCADE ON UPDATE CASCADE  
);
```

```
CREATE TABLE `Group` (  
    `group_id` INT NOT NULL AUTO_INCREMENT,  
    `teacher_id` INT NOT NULL,  
    `lectionId` INT NOT NULL,  
    `category` varchar(255) NOT NULL,  
    PRIMARY KEY (`group_id`),  
    CONSTRAINT `Group_fk0` FOREIGN KEY (`teacher_id`) REFERENCES  
`Teacher`(`teacher_id`) ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
CREATE TABLE `Student` (  
    `student_id` INT NOT NULL AUTO_INCREMENT,  
    `first_name` varchar(255) NOT NULL,  
    `last_name` varchar(255) NOT NULL,  
    `group` INT NOT NULL,  
    PRIMARY KEY (`student_id`),  
    CONSTRAINT `Student_fk0` FOREIGN KEY (`group`) REFERENCES `Group`(`group_id`)  
ON DELETE CASCADE ON UPDATE CASCADE
```

);

```
CREATE TABLE `StudentInfo` (  
    `identification_number` varchar(255) NOT NULL,  
    `civil_number` varchar(255) NOT NULL,  
    `medical_info` VARCHAR(255) NOT NULL,  
    `student` INT(255) NOT NULL,  
    CONSTRAINT `StudentInfo_fk0` FOREIGN KEY (`student`) REFERENCES  
`Student`(`student_id`) ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
CREATE TABLE `Schedule` (  
    `lection_time` TIME NOT NULL,  
    `teacher` INT NOT NULL,  
    `group` INT NOT NULL,  
    `section_decription` varchar(255) NOT NULL,  
    CONSTRAINT `Schedule_fk0` FOREIGN KEY (`teacher`) REFERENCES  
`Teacher`(`teacher_id`) ON DELETE CASCADE ON UPDATE CASCADE,  
    CONSTRAINT `Schedule_fk1` FOREIGN KEY (`group`) REFERENCES  
`Group`(`group_id`) ON DELETE CASCADE ON UPDATE CASCADE  
);
```

**Висновок:** на цій лабораторній роботі було завершено моделювання і засобами SQL створено базу даних, що складається з восьми таблиць.