МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт
з дисципліни
“ОРГАНІЗАЦІЯ БАЗ ДАНИХ”

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт
з дисципліни
«Організація баз даних»

для студентів спеціальності
174 «Автоматизація, комп'ютерно-інтегровані
технології та робототехніка»

Затверджено
редакційно-видавничою
радою університету,
протокол № 1 від 16.02.2023 р.

Харків
НТУ «ХПІ»
2023

Методичні вказівки до виконання лабораторних робіт з дисципліни «Організація баз даних» / уклад. І. Л. Красніков, А.К. Бабіченко, І.Г. Лисаченко, О.Г. Шутинський. – Харків : НТУ «ХПІ». – 53 с.

Укладач І.Л. Красніков

Рецензент О.М. Дзевочко

Кафедра автоматизації технологічних систем та екологічного моніторингу

ЛАБОРАТОРНА РОБОТА № 1

ЗНАЙОМСТВО З MYSQL. РОБОТА З КОНСОЛЬНИМ КЛІЄНТОМ MYSQL

Мета роботи – Знайомство з СУБД *MYSQL*. Отримання навичок роботи з консольним клієнтом *mysql*.

1.1 Загальні положення

MySQL (правильно вимовляється як "Май Эс Кью Эль") це одна з найпопулярніших систем управління реляційних базами даних (СУБД) з відкритим програмним кодом, яка відрізняється великою продуктивністю, високим рівнем безпеки, та може бути встановлена практично на усі існуючі операційні систем. Існує як платна (*MySQL Enterprise Edition*), так і безкоштовна (*MySQL Community*) версії програми. Безкоштовну версію можна завантажити за адресою http://dev.mysql.com/downloads/mysql

MySQL складається з двох частин: серверної та клієнтської. Серверна частина *MySQL* постійно працює на комп'ютері, який виділений під сервер. Клієнтська частина *MySQL* може бути встановлена як на тому ж комп'ютері, так і на будь-якому іншому, до якого є доступ по мережі. До стандартної поставки *MySQL* сервера входить консольний клієнт *mysql* (*MySQL* – це сервер баз даних, а файл *mysql.exe* – це консольний клієнт, для роботи з *MySQL*).

Для підключення до сервера *MySQL* з консольного клієнта *mysql* (*mysql.exe)* потрібно знати: *IP*-адресу або доменне ім'я сервера, порт (стандартне значення 3306), логін і пароль. Ці значення передаються як параметри клієнта *mysql*. Побачити список команд програми *mysql* можна, запустивши її з параметром --*help*:

C:\> mysql --help

Основні опції команди *mysql*, що будуть використовуватись при виконанні лабораторних робіт наведені в табл.1.1.

Таблиця 1.1. Основні опції команди *mysql*

-?, --help	Довідка
-h,--hostname=[hostname]	Ім’я (адреса) сервера *MYSQL*
-u, --user=[user]	Ім’я користувача, що має доступ до *MYSQL*
-p,--password=[password]	Пароль користувача для доступу до *MYSQL*
-P,--port=[port]	Порт для з’єднання з сервером (за замовчуванням 3306).
-V, --version	Інформація о версії

Наприклад одержати інформацію о версії сервера можна за допомогою наступних команд

```
C:\Users\ikl>mysql -V
mysql  Ver 14.14 Distrib 5.7.7-rc, for Win64 (x86_64)
```

або

```
C:\Users\ikl>mysql --version
mysql  Ver 14.14 Distrib 5.7.7-rc, for Win64 (x86_64)
```

Початково після установки існує єдиний обліковий запис користувача, якому надається право входу в *MYSQL* з ім'ям *root*, який не має пароля, що не є доцільним з точки зору безпеки.

Для підключення з того ж комп’ютера, на якому встановлено сервер через стандартный порт необхідно в командній консолі (*cmd*) вказати повний шлях до *MySQL* і ввести команду:

C:\> mysql –u Name –p

Name ім’я користувача. Параметр *-p* вказує, що при підключенні потрібно ввести пароль. При вдалому підключенні клієнт *mysql* перейде в інтерактивний режим і можна побачити приблизно наступну картинку.

```
c:\Program Files\MySQL\MySQL Server 5.7\bin>mysql -uroot -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 62
Server version: 5.7.7-rc-log MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Пароль може бути переданий і безпосередньо з командного рядка, тоді він вводиться відразу після ключа *-p* без пробілів, наприклад, так:

```
mysql> -u Name -pmypassword
```

Для виходу з клієнта *mysql* передбачені команди *exit* або *quit:*

```
mysql> quit
Bye
```

Якщо сервер *MySQL* знаходиться на віддаленому комп'ютері (хості) – його мережну адресу необхідно вказати за допомогою опції *-h*.

```
c:\Program Files\MySQL\MySQL Server 5.7\bin>mysql -h 127.0.0.1 -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 77
Server version: 5.7.7-rc-log MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

В наведеному вище прикладі адреса 127.0.0.1 (або *localhost*) це адреса для доступу клієнта до сервера на тій же машині. При підключенні по мережі з іншої машини необхідно замість 127.0.0.1 ввести *IP*-адресу або доменне ім’я сервера.

Після підключення до серверу клієнт *mysql* працює в інтерактивному режимі в якому результати запиту представляються в *ASCII*-форматі. *SQL*-запит повинен закінчуватись одним із символів: «;», «\g» або «\G». Якщо ввести в командному рядку *help* без параметру, програма *mysql* виведе список усіх підтримуваних нею команд, але якщо після команди вказати ім'я функції або оператора *MySQL*, то з сервера буде завантажена довідка за зазначеною функцією.

Наприклад:

```
mysql> help USE;
Name: 'USE'
Description:
Syntax:
USE db_name

The USE db_name statement tells MySQL to use the db_name database as
the default (current) database for subsequent statements. The database
remains the default until the end of the session or another USE
statement is issued:

USE db1;
SELECT COUNT(*) FROM mytable;   # selects from db1.mytable
USE db2;
SELECT COUNT(*) FROM mytable;   # selects from db2.mytable

URL: http://dev.mysql.com/doc/refman/5.7/en/use.html
```

Подивитися яка версія *MySQL* використовується і дані про поточного користувача можна командою *status*.

```
mysql> status;
--------------
mysql  Ver 14.14 Distrib 5.7.7-rc, for Win64 (x86_64)

Connection id:          4
Current database:
Current user:           root@localhost
SSL:                    Not in use
Using delimiter:        ;
Server version:         5.7.7-rc-log MySQL Community Server (GPL)
Protocol version:       10
Connection:             localhost via TCP/IP
Server characterset:    utf8
Db     characterset:    utf8
Client characterset:    cp866
Conn.  characterset:    cp866
TCP port:               3306
Uptime:                 1 hour 3 min 39 sec

Threads: 2  Questions: 12  Slow queries: 0  Opens: 108  Flush tables: 1  Open
 Queries per second avg: 0.003
--------------
```

Всі введені команди, а також їх результати можна зберегти в текстовий файл за допомогою опцій: *T*, *tee* та *t*, *notee.*

Наприклад після введення наступної команди усі дані будуть записуватись до файлу test.txt:

```
mysql> tee test.txt
Logging to file 'test.txt'
mysql>
```

Закінчити вивід до файлу можна командою *notee* або *t*:

```
mysql> \t
Outfile disabled.
mysql>
```

Після встановлення сервера *MySQL*, підключитися до нього можна тільки з цієї ж машини. Зроблено це з метою безпеки системи. Для того, щоб підключиться віддалено потрібно на сервері створити користувача і присвоїти йому відповідні права.

1.2 Порядок виконання роботи

1.2.1 Переглянути список команд консольного клієнту *mysql*

1.2.2 Підключитися через консольний клієнт *mysql (mysql.exe)* до серверу (*login:* ***student****, password:* ***student****, порт за замовчуванням* ***3306***). *IP*-адресу сервера взяти у викладача.

1.2.3 Набрати команду *help* та переглянути список команд

1.2.4 Забезпечити збереження усіх дій у текстовий файл, який необхідно додати до звіту з лабораторної роботи (команда *T* або *tee*).

1.2.5 Отримати інформацію про статус сервера (команда \s)

1.2.6 Вивести поточну дату та ім'я користувача командою

```
mysql> select now(), user();
```

1.2.7 Подивитися привілеї які призначені користувачеві *student*

```
mysql> show grants for 'student'@'%';
```

або

```
mysql> show grants for student;
```

Результат має приблизно таку структуру:

```
mysql> show grants for 'student'@'%';
+-----------------------------------------------------------------+
| Grants for student@%                                            |
+-----------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'student'@'%'                             |
| GRANT SELECT ON 'mysql'.'user' TO 'student'@'%'                 |
| GRANT ALL PRIVILEGES ON 'education'.'mysql' TO 'student'@'%'    |
+-----------------------------------------------------------------+
3 rows in set (0.00 sec)
```

В даному випадку користувач з ім'ям *student* має право робити вибірку даних з таблиці *user* бази даних *mysql* та має усі привілеї при роботі з таблицею *mysql* бази даних *education* з усіх комп'ютерів та не має жодних привілей для усіх інших баз даних.

1.2.8 Подивитися список баз даних доступних користувачеві *student* на сервері командою

```
mysql> show databases;
```

1.2.9 Обрати поточною системну базу даних *mysql*, в якій зберігаються відомості для адміністрування сервера

```
mysql> use mysql;
```

Також обрати поточну базу даних можна на етапі підключення. Наприклад наступна команда обирає для користувача *student* поточною базу даних *education*:

```
C:\> mysql –D education –uStudent –pStudent
```

1.2.10. Перевірити яка база є поточною

```
mysql> select database();
```

1.2.11. Вивести на екран список таблиць поточної бази даних, для доступу до яких користувачеві надані права:

```
mysql> show tables;
```

1.2.12. Вивести на екран структуру таблиці *user*, в якій зберігаються дані про зареєстрованих користувачів системи командою *describe* або скорочено *desc*;

```
mysql> describe user;
```

або

```
mysql> desc user;
```

З метою уникнення помилок більш доцільно використовувати повне ім'я таблиці

```
mysql> describe mysql.user;
```

Якщо таблиця не вміщається у вікні консолі по ширині можна замість «**;**» використовувати команди «**\g**» або «**\G**».

```
mysql> desc mysql.user \g
```

```
mysql> desc mysql.user \G
```

1.2.13. Вивести на екран список існуючих користувачів (*user*) і список хостів (*host*) с яких їм дозволено доступ до сервера (% означає доступ з будь-якої адреси в мережі, *localhos* – з машини на якій встановлено сервер) за допомогою *SQL* запиту до таблиці *user* системної бази даних *mysql.*

```
mysql> SELECT user, host FROM mysql.user;
```

1.2.14. Відключити вивід у файл та розірвати з'єднання з сервером.

1.2.15 Скласти звіт до лабораторної роботи, який повинен містити файл з усіма використаними командами.

Контрольні питання

1) Який порт за замовчуванням має сервер *MYSQL*?
2) Яка команда дозволяє побачити права користувача?
3) Як показати, що користувач має дозвіл на доступ до серверу з будь-якої машини мережі?
4) Як подивитися список баз даних, що доступні користувачу?
5) Як записати усі консольні команди поточного сеансу роботи з сервером у файл?

ЛАБОРАТОРНА РОБОТА № 2

РОБОТА З ОБЛІКОВИМИ ЗАПИСАМИ КОРИСТУВАЧІВ СУБД MYSQL

Мета роботи – отримання навичок керування обліковими записами користувачів і визначення їх привілеїв.

2.1 Загальні положення

СУБД *MySQL* є багатокористувацьким середовищем, тому для доступу до таблиць БД повинні бути створені різні облікові записи з різним рівнем привілеїв.

За замовчуванням створюється єдиний обліковий запис користувача, якому надається право входу *root*, який не має пароля.

Обліковий запис користувача має форму *'username' @ 'hostname'*, де *username* – ім'я користувача, а *hostname* – найменування хоста, з якого користувач може звертатися до сервера. (Якщо частина *username* або *hostname* не містить таких спеціальних символів, як '-' або '%', його необов'язково брати в лапки. Проте безпечніше використовувати лапки).

Наприклад, записи *'root'* @ '127.0.0.1' і *'student'*@'172.17.124.17' означають, що користувач з ім'ям *root* може звертатися з хоста, на якому розташований сервер, а *'student'* – тільки з хоста з *IP*-адресою 172.17.124.17. *IP*-адреса 127.0.0.1 завжди відноситься до локального хосту. *IP*-адреса 127.0.0.1 має псевдонім *localhost*, тому облікові записи виду '*username*'@'127.0.0.1' можна записувати у вигляді *'username'@'localhost'*. Якщо необхідно забезпечити доступ користувачеві до сервера з декількох комп'ютерів в імені хоста використовується спеціальний символ "%". Так, обліковий запис 'student'@'%' дозволяє користувачеві *student* звертатися до сервера *MySQL* з будь-яких комп'ютерів мережі.

Створення нового облікового запису. Створити обліковий запис дозволяє оператор

```
CREATE USER 'username'@'hostname' [IDENTIFIED BY [PASSWORD] 'пароль'];
```

Ключове слово *IDENTIFIED BY* вказує на наступний за ним пароль облікового запису. *IDENTIFIED BY* може бути відсутнім у реченні *GRANT*. У цьому випадку пароль облікового запису не потрібен. Однак з метою безпеки необхідно завжди призначати пароль.

Наприклад, створення нового облікового запису може виглядати наступним чином:

```
CREATE USER 'test'@'%' IDENTIFIED BY 'test2015';
```

Ця команда створює нового користувача з ім’ям *test*, який може підключатися до серверу з будь-якого комп’ютера мережі з паролем *test2015.*

Видалення облікового запису. Видалити обліковий запис дозволяє оператор

```
DROP USER 'sername'@'hostname';
```

Наприклад, наступний оператор видаляє користувача з ім’ям *test*

```
DPOP USER 'test'@'%';
```

Зміна імені користувача в обліковому записі здійснюється за допомогою оператора:

```
RENAME USER старе_імя TO нове_імя;
```

Усі облікові записи зберігаються в таблиці *user* системної бази даних з ім'ям *mysql*.

Призначення привілеїв. Розглянуті вище оператори дозволяють створювати, видаляти і редагувати облікові записи, але вони не дозволяють змінювати привілеї користувача – повідомляти *MySQL*, який користувач має

право тільки на читання інформації, який на читання і редагування, а кому надані права змінювати структуру БД і створювати облікові записи.

Для вирішення цих завдань призначені оператори *GRANT* (призначає привілеї) і *REVOKE* (видаляє привілеї). Якщо облікового запису, який вказаний в операторі *GRANT*, не існує, то він автоматично створюється. Видалення всіх привілеїв за допомогою оператора *REVOKE* не призводить до автоматичного знищення облікового запису. Для видалення користувача необхідно скористатися оператором *DROP USER*.

Перелік привілей, які можуть надаватися користувачеві наведені в табл. 2.1.

Таблиця 2.1 Привілеї користувачів MYSQL

Привілей	Операція, дозволена привілеєм
ALL [PRIVILEGES]	Дозволяє усі привілеї, за виключенням привілеї *WITH GRANT OPTION*, яка задається окремо
ALTER	Дозволяє перейменовувати таблиці, вставляти нові поля в таблицю, видаляти поля з таблиці, а також модифікувати їх.
ALTER ROUTINE	Дозволяє редагувати або видаляти збережену процедуру
CREATE	Дозволяє створювати нові бази даних, а також нові таблиці в базі даних
CREATE ROUTINE	Дозволяє створювати збережену процедуру
CREATE TEMPORARY TABLES	Дозволяє створювати тимчасові таблиці, які видаляються по закінчені сесії
CREATE USER	Дозволяє працювати з обліковими записами с допомогою *CREATE USER, DROP USER, RENAME USER* і *REVOKE ALL PRIVILEGES*
CREATE VIEW	Дозволяє створювати уявлення
DELETE	Дозволяє видаляти записи з таблиць
DROP	Дозволяє видаляти таблиці, або бази даних
EXECUTE	Дозволяє виконувати збережені процедури
FILE	Дозволяє робити вибірку записів і записувати дані в файл, а також зчитувати їх звідти.
INDEX	Дозволяє працювати з індексами, зокрема, використовувати оператори *CREATE INDEX* и *DROP*

Привілей	Операція, дозволена привілеєм
	INDEX
INSERT	Дозволяє додавати в таблицю нові записи
LOCK TABLES	Дозволяє здійснювати блокування таблиць за допомогою операторів LOCK TABLES і UNLOCK TABLES.
SELECT	Дозволяє робити вибірку записів з таблиць баз даних.
SHOW DATABASES	Дозволяє переглядати список всіх таблиць на сервері
SHOW VIEW	Дозволяє використовувати оператор *SHOW CREATE VIEW*
UPDATE	Дозволяє оновлювати вміст таблиць (оновлювати записи)
USAGE	Синонім для статусу «відсутні привілеї»

Наприклад, запит на надання привілей може виглядати наступним чином:

```
mysql> GRANT ALL ON *.* TO 'test'@'%' IDENTIFIED BY 'test2015';
Query OK, 0 rows affected, 1 warning (0.34 sec)
```

Даний запит створює користувача з ім'ям *test* і паролем *test2015*, який може звертатися до сервера з будь-якого комп'ютеру мережі і має всі права (*ALL*) для всіх баз даних (*.*). Якщо такий користувач вже існує, то команду *IDENTIFED BY* вводити не потрібно. Замість ключового слова *ALL* можна використовувати будь-яке з ключових слів, або їх комбінацій, представлених в табл.2.1 **(***к*ористувач може передати іншому користувачу тільки ті привілеї, які має сам).

Відмінити надані права можна за допомогою команди *REVOKE.*

```
mysql> REVOKE DELETE ON *.* FROM 'test'@'%';
Query OK, 0 rows affected (0.07 sec)
```

Ця команда забороняє користувачеві з ім'ям *test* видаляти записи з таблиць усіх існуючих БД.

Ключове слово *ON* в операторі *GRANT* задає рівень привілеїв, які можуть бути задані на одному з чотирьох рівнів, представлених в табл. 2.2.

Таблица 2.2. Рівень привілеїв

Ключове слово *ON*	Рівень привілеїв
*ON *.**	Глобальний рівень – користувач з повноваженнями на глобальному рівні може звертатися до всіх БД і таблиць, що входять до їх складу
*ON db.**	Рівень бази даних – привілеї поширюються на таблиці бази даних db
ON db.tbl	Рівень таблиці – привілеї поширюються на таблицю tbl бази даних db
ON db.tbl	Рівень стовпця - привілеї стосуються окремих стовпців в таблиці tbl бази даних db. Список стовпців вказується в дужках через кому після ключових слів *SELECT*, *INSERT*, *UPDATE*

Для таблиць можна встановити тільки наступні типи привілеїв: *SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, GRANT OPTION, INDEX* и *ALTER*.

Для стовпця – тільки *SELECT*, *INSERT* і *UPDATE*.

Речення *WITH GRANT OPTION* дозволяє керувати привілеями інших користувачів, без даної привілеї неможливо виконати оператори *GRANT* і *REVOKE*. Слід пам'ятати, що *GRANT OPTION* дає користувачеві здатність передати іншим користувачам будь привілеї, які цей користувач має в певному рівні доступу. Два користувача з різними привілеями можуть бути здатні поєднати свої привілеї. Щоб відмінити привілею, яка надана командою *GRANT*, використовується команда *REVOKE*.

```
mysql> REVOKE GRANT OPTION ON ... FROM ...;
```

2.2 Порядок виконання лабораторної роботи

2.2.1 Підключитися через консольний клієнт *mysql* до серверу (*login:* ***student****, password:* ***student****, порт за замовчуванням (3306*). *IP*-адресу сервера взяти у викладача).

2.2.2 Переглянути список баз даних до яких має доступ користувач *student.*

2.2.3 Переглянути привілеї, які має користувач *student*

2.2.4 За допомогою команди *CREATE* створити два нових облікових записа користувача (прізвище студента латинськими літерами): перший з доступом лише з *IP*-адреси Вашого комп'ютера, другий – з будь-якого комп'ютера глобальної мережі.

Вимоги до паролю: ***мінімум 8 символів; мінімум 1 буква велика; мінімум 1 цифра та 1 службовий знак.***

2.2.5 Переглянути права надані користувачам за замовченням.

2.2.6 Передати користувачам права, що має обліковий запис *student.*

2.2.7 Видалити права 1-го облікового запису (з обмеженим доступом) (*REVOKE*).

2.2.8 Перейменувати і потім видалити цей обліковий запис (*DROP*).

2.2.9 Підключитися до серверу під новим ім'ям. Переглянути свої привілеї.

2.2.10 Скласти звіт до лабораторної роботи, який повинен містити усі використані команди.

Контрольні питання

1) До чого відноситься IP-адреса 127.0.0.1?
2) Як визначити, що користувач має дозвіл на доступ до серверу з будь-якої машини мережі?
3) Яка команда відміняє права користувача?
4) Як створити користувача с певними правами?
5) Як видалити користувача?

ЛАБОРАТОРНА РОБОТА №3

СТВОРЕННЯ ТА ЗАПОВНЕННЯ БАЗИ ДАНИХ *MYSQL*

Мета роботи – Отримання навичок створення та заповнення бази даних в консольному клієнті *mysql*.

3.1 Завдання до лабораторної роботи

Особливості створення бази в консольному клієнті *mysql* розглянемо на прикладі простої бази даних, яка містить загальні відомості про студентів.

База складається з двох таблиць:

Таблиця STUDENT (Студент)	
idStud	**Код студента (номер залікової книжки)**
SFam	Прізвище студента
SName	Ім’я студента
Stip	Розмір стипендії студента
Groupname	Код групи, де навчається студент

Таблиця GR (Група)	
idGr	**Код групи**
Starosta_Fam	Прізвище старости групи

В таблиці STUDENT первинним ключем є ***idStud***, а в таблиці GR – ***idGr***. Зв'язок між таблицями здійснюється за допомогою зовнішнього ключа *Grupname* в таблиці STUDENT, який пов'язаний з первинним ключем таблиці GR.

3.2 Загальні положення

3.2.1 Створення бази даних

Для створення бази даних необхідно ввести команду

```
CREATE DATABASE db_Name;
```

Переконатися в тому, що база даних створена, можна за допомогою оператора

```
SHOW DATABASES;
```

Перед початком роботи необхідно обирати створену базу даних поточною за допомогою команди

```
USE db_Name;
```

Тепер всі дії за замовчуванням будуть застосовуватися саме до цієї БД.

3.2.2 Створення таблиць здійснюється командою *CREATE TABLE,* синтаксис якої в загальному виді виглядає наступним чином:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS]
tbl_name [(CREATE_DEFINITION,...)];
```

де *tbl_name* задає ім'я таблиці, яка буде створена в поточній базі даних. Якщо жодна база даних на момент виклику команди *CREATE TABLE* не була обрана поточною, то виникне помилка виконання команди. У цьому випадку можна явно вказати базу даних, в якій буде створена нова таблиця, за допомогою синтаксису *db_name.tbl_name*.

Необов'язкове ключове слово *TEMPORARY* використовується для створення таблиць, які будуть існувати тільки в поточному сеансі роботи з БД і будуть автоматично видалені, коли сеанс завершиться.

При використанні виразу *IF NOT EXISTS* таблиця буде створена тільки в тому випадку, якщо ще немає таблиці із зазначеним ім'ям *tbl_name*.

CREATE_DEFINITION визначає внутрішню структуру створюваної таблиці (назви і типи полів, ключі, індекси і т.п.).

У визначені полів таблиці (*CREATE_DEFINITION*) найбільш часто використовуються наступні:

```
col_name TYPE [NOT NULL | NULL] [DEFAULT
DEFAULT_VALUE] [AUTO_INCREMENT] [PRIMARY KEY]
[REFERENCE_DEFINITION]
```

Параметри в *CREATE_DEFINITION* означають наступне:

- *col_name* задає ім'я стовпця в таблиці, що створюється;
- *TYPE* задає тип даних для стовпця *col_name* (типи даних, що застосовуються в *MYSQL* наведені у додатку);
- *[NOT NULL | NULL]* – вказує що стовпцю заборонено або не заборонено приймати значення *NULL*). За замовчуванням – *NULL*;
- *[DEFAULT default_value]* –Задає значення за замовчуванням для даного стовпця. При вставці нового запису в таблицю командою *INSERT* якщо значення для поля *col_name* явно зазначено не було, то встановлюється значення *default_value;*
- *[AUTO_INCREMENT]* – при вставці нового запису в таблицю поле з цим атрибутом автоматично отримає числове значення, на 1 більше найбільшого значення для цього поля в поточний момент часу. Стовпець, для якого застосовується атрибут *AUTO_INCREMENT*, повинен мати цілочисельний тип. У таблиці може бути тільки один стовпець з атрибутом *AUTO_INCREMENT;*
- *PRIMARY KEY* – оголошує стовпець первинним ключем.

Наступний приклад створює таблицю *users* з 3 полями, де перше поле – унікальний ідентифікатор запису (первинний ключ), друге поле – ім'я користувача, а третє поле – його вік:

```
CREATE TABLE users (
    id INT(11) NOT NULL AUTO_INCREMENT,
    name CHAR(30) NOT NULL,
    age SMALLINT(6) NOT NULL,
    PRIMARY KEY(id)
    );
```

Перевірити структуру створеної таблиці можна за допомогою команди *DESCRIBE table_name.*

Наприклад перевіримо структуру таблиці *users.*

```
mysql> DESCRIBE users;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| id    | int(11)     | NO   | PRI | NULL    | auto_increment |
| name  | char(30)    | NO   |     | NULL    |                |
| age   | smallint(6) | NO   |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

Встановлення зв’язків між таблицями та забезпечення цілісності бази даних за посиланнями здійснюється за допомогою зовнішніх ключів (*FOREIGN KEYS),* які задаються конструкцією:

```
    foreign key [name_key] (col1, ... )
    REFERENCES  tbl (tbl_col, ... )
    [ON DELETE {CASCADE | SET NULL | NO ACTION | RESTRICT
| SET DEFAULT}]
    [ON UPDATE {CASCADE | SET NULL | NO ACTION  |
RESTRICT | SET DEFAULT}]
```

Конструкція дозволяє задати зовнішній ключ з необов'язковим ім'ям *name_key* на стовбцях, які задаються в круглих дужках (один або декілька). Ключове слово *REFERENCES* вказує таблицю *tbl*, на яку посилається зовнішній ключ, в круглих дужках вказуються імена стовпців. Необов'язкові конструкції *ON DELETE* і *ON UPDATE* дозволяють задати поведінку СУБД при видаленні і оновлені рядків з таблиці-предка. Параметри, наступні за цими ключовими словами, мають такі значення:

- *CASCADE* – при видаленні або оновленні запису в батьківській таблиці, що містить первинний ключ, записи з посиланнями на це значення в дочірній таблиці видаляються або оновлюються автоматично;

- *SET NULL* – при видалення або оновлення запису в батьківській таблиці, що містить первинний ключ, записи з посиланнями на це значення в дочірній таблиці, встановлюються в *NULL*;
- *NO ACTION* – при видаленні або оновленні записів, що містять первинний ключ, з дочірньою таблицею ніяких дій не проводиться;
- *RESTRICT* – якщо в дочірній таблиці є записи, що посилаються на первинний ключ батьківської таблиці, при видаленні або оновленні записів з таким первинним ключем повертається помилка;

Приклад створення двох таблиць та встановлення зв'язок між ними за допомогою зовнішнього ключа.

```
CREATE table GR(
idGr varchar (10) NOT NULL,
Starosta_Fam varchar (50),
PRIMARY KEY(idGr));

CREATE table STUDENT(
    idStud smallint(5) NOT NULL,
    SFam varchar (50) NOT NULL,
    SName varchar (50),
    Stip real,
GroupName varchar (10),
PRIMARY KEY(idStud),
FOREIGN KEY (GroupName) REFERENCES Lab3.GR(idGr)
ON UPDATE CASCADE
);
```

Вводити данні до таблиць у клієнті *mysql* можна за допомогою команди *INSERT*, що має наступну структуру:

```
INSERT INTO tbl_name [(col_name,...)]
        VALUES (expression,...),(...),...);
```

Наприклад

```
INSERT
INTO GR(idGr,Starosta_Fam)
```

```
VALUES
('ІКМ-719А','Агєєв'),
('ІКМ-719Б','Борисов'),
('ІКМ-719В','Семенов');
```

Для оновлення даних використовується оператор *UPDATE*

```
UPDATE tbl_name
    SET col_name1=expr1 [, col_name2=expr2, ...]
    [WHERE where_definition]
```

де ***tbl_name*** - задає ім'я таблиці, в якій будуть оновлюватися запису. На момент запуску команди *UPDATE* таблиця з таким ім'ям повинна існувати в базі даних.

У виразі *SET* вказується, які саме стовпці слід модифікувати і які величини повинні бути в них встановлені. У виразі *WHERE*, якщо воно присутнє, задається, які рядки підлягають оновленню. В інших випадках оновлюються всі рядки.

Приклад

```
mysql> UPDATE persondata SET age=age+1;
```

Структура таблиці змінюється за допомогою оперfтора *ALTER TABLE*. Оператор *ALTER TABLE* забезпечує можливість змінювати структуру існуючої таблиці. Наприклад, можна додавати або видаляти стовпці, створювати або знищувати індекси або перейменовувати стовпці або саму таблицю. Можна також змінювати коментар для таблиці та її тип.

Для того щоб додати новий стовпець з ім'ям Name:

```
mysql> ALTER TABLE tabl ADD Name char(20);
```

3.3 Порядок виконання лабораторної роботи

3.3.1 Підключитися через консольний клієнт *mysql* до серверу за допомогою облікового запису, що був створений у лабораторній роботі №2.

3.3.2 Створити базу даних на ім'я L3**FAM** (**FAM** - Ваше прізвище) та обрати її поточною.

3.3.3. Перевірити права надані Вашому обліковому запису. Для виконання лабораторної роботи повинні бути надані наступні права : CREATE, DROP, ALTER, INSERT, REFERENSES, SELECT, UPDATE, SHOW DATABASES

3.3.4 Створити таблиці GR та STUDENT та перевірити відповідність їх структури завданню (DESCRIBE). Забезпечити цілісність бази даних за допомогою зовнішнього ключа та передбачити можливість автоматичного оновлення даних в дочірній таблиці (STUDENT) при оновлені даних в батьківській таблиці (GR)

3.3.5 Ввести 3-4 записи до кожної таблиці (команда INSERT)

3.3.6 Превірити введені дані (SELECT * FROM Table_Name)

3.3.7 Змінити назву однієї групи у таблиці GR та перевірити зміни у таблиці STUDENT

3.3.8 Змінити структуру таблиці GR, додавши ім'я старости (Starosta_Name) (*ALTER TABLE)*

3.3.9 Збільшити стипендію усім студентам на 20% (*UPDATE*)

Контрольні питання

1) Що буде, якщо змінити номер групи в таблиці STUDENT
2) Як можна перевірити структуру таблиці
3) Як додати у таблицю додатковий атрибут?
4) Яка конструкція забезпечує цілісність бази даних за посиланнями?

ЛАБОРАТОРНА РОБОТА №4

РОБОТА З ГРАФІЧНОЮ УТИЛІТОЮ *MySQL Workbench*

Мета роботи – Отримання навичок роботи в *MySQL Workbench*

4.1 Загальні положення

MySQL Workbench – інструмент для візуального проектування баз даних, інтегруючий проєктування, моделювання, створення та експлуатацію БД в єдине оточення для системи баз даних *MySQL*.

MySQL Workbench дозволяє вирішувати три основні задачі:

1. Адмініструвати бази даних: Замінює програму адміністрування *mysql*. Наявний графічний інтерфейс служить для запуску/зупинки серверів, створення облікових записів, редагування конфігураційних файлів і т.ін.

2. Дозволяє користувачеві підключатися до існуючої бази даних для редагування і виконання *SQL* запитів.

3. Моделювати дані: здійснювати повне візуальне проектування та моделювання баз даних.

MySQL Workbench можна безкоштовно скачати та встановити за адресою *http://dev.mysql.com/downloads/workbench/*.

Стартовий екран програми складається з 3-х секцій***:*** *MySQL Connection, Models ma Shortcuts,* який показано на рис.4.1.

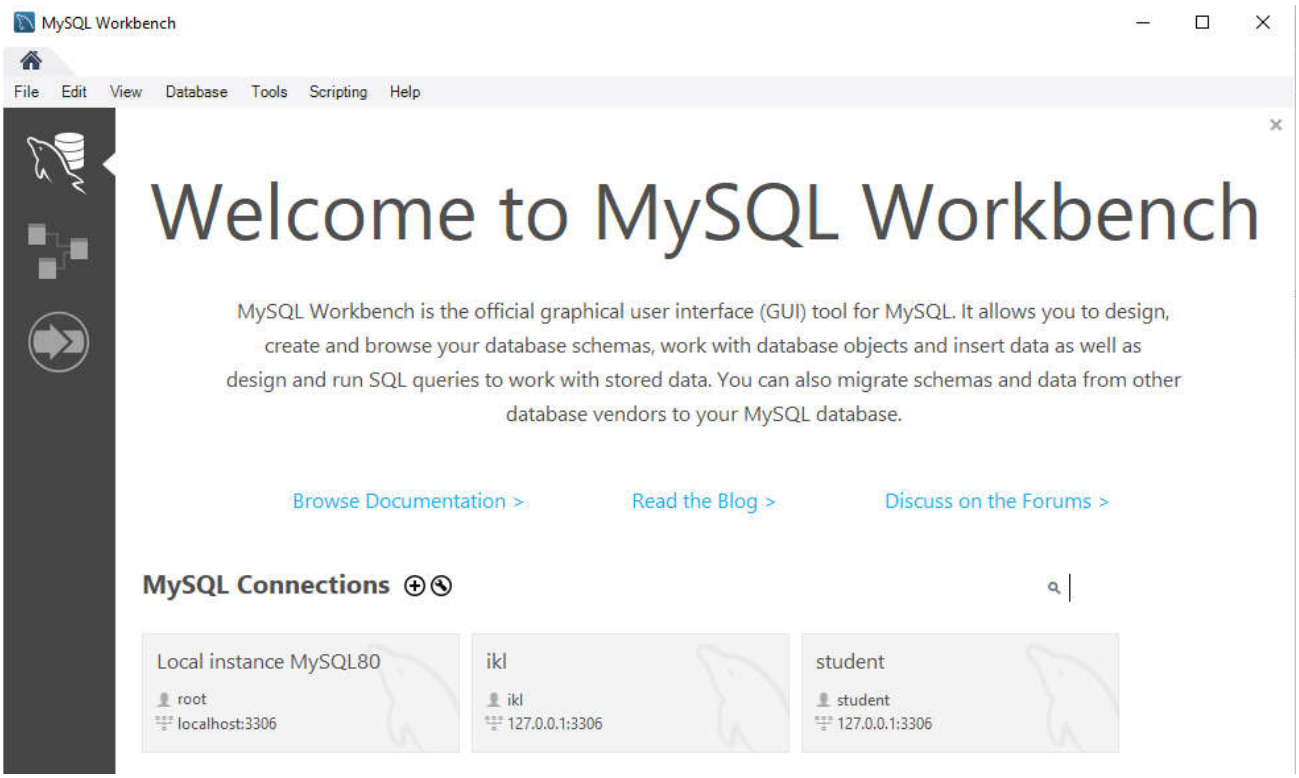


Рис. 4.1. Головний екран програми *MySQL Workbench* 8.3

MySQL Connection. У цьому розділі розташовані з'єднання до всіх серверів MySQL (див.рис 4.2)ю

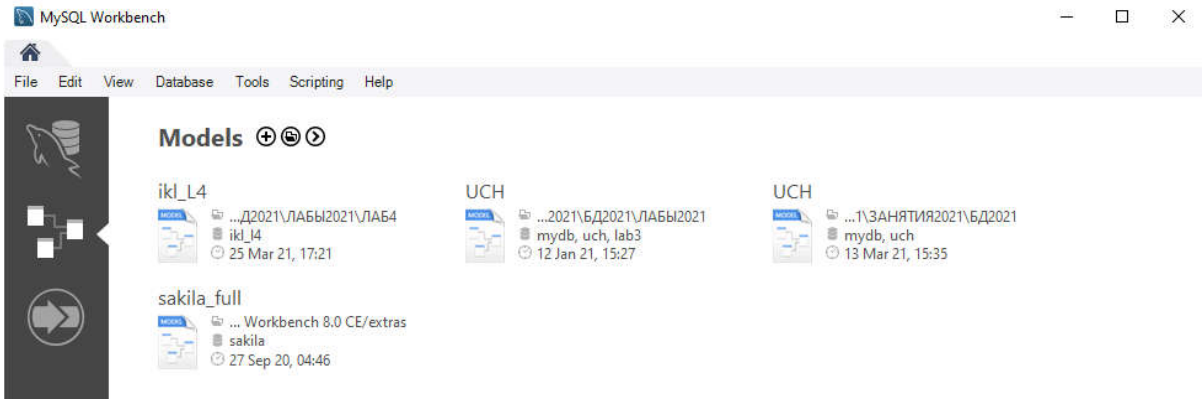


Рис. 4.2. Екран Models програми *MySQL Workbench* 8.3

Вкладка *Models* містить інформацію про бази даних, що використовуються найчастіше. Біля кожної панелі показані шлях до моделі та дата останнього відкриття. Праворуч від *Models* розташовані три

іконки: (+) додає нову модель, (тека) відкриває існуючу модель з диска і (>) відкриває контекстне меню для додаткових команд, таких як *Create EER Model from Database* (створити *EER* модель з бази даних).

4.2 Створення нового з'єднання з сервером MySQL

Кнопка (+) праворуч від заголовка ***MySQL Connection*** відкриває форму *Setup New Connection*, яка показана на рис.4.3.

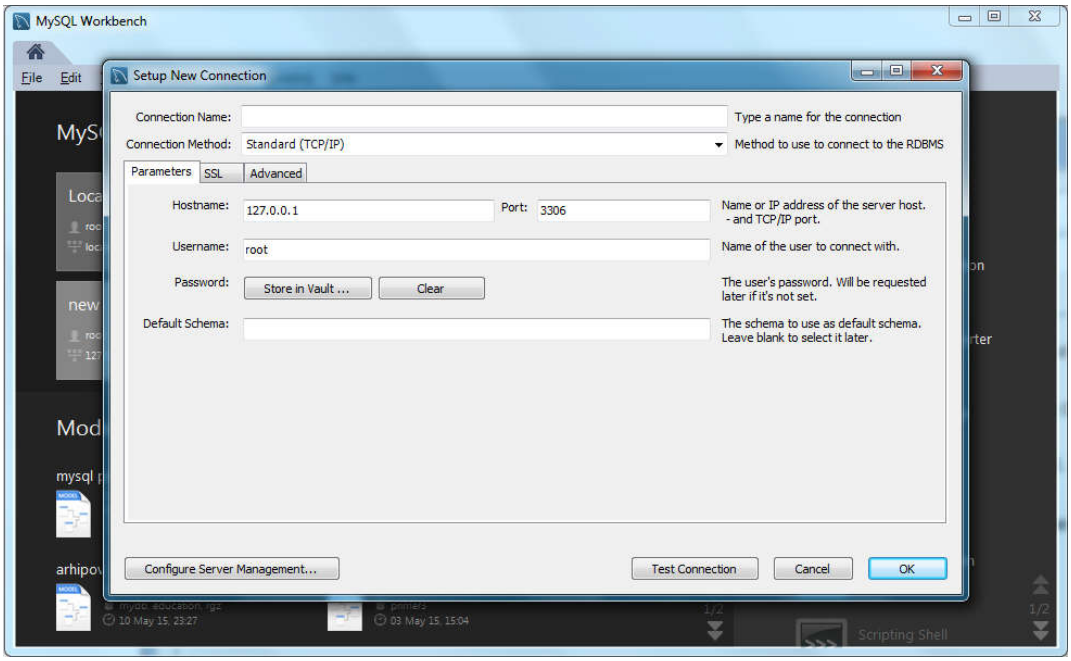


Рис.4.3. Створення нового з’єднання з сервером *MySQL*

У вікні необхідно ввести ім’я з’єднання (*Connection Name*), *IP*-адресу серверу (*Hostname*) та ім’я користувача (*Username*). Перевірити чи має користувач з такими параметрами права для підключення до сервера можна кнопкою *Test Connection***.** Нове з’єднання з’явиться в стартовому вікні.

4.3 Створення нової бази даних

Запустивши *MySQL Workbench* в розділі *Models* (Моделювання) оберемо Кнопка (+). З’явиться наступне вікно (див. рис.4.4).

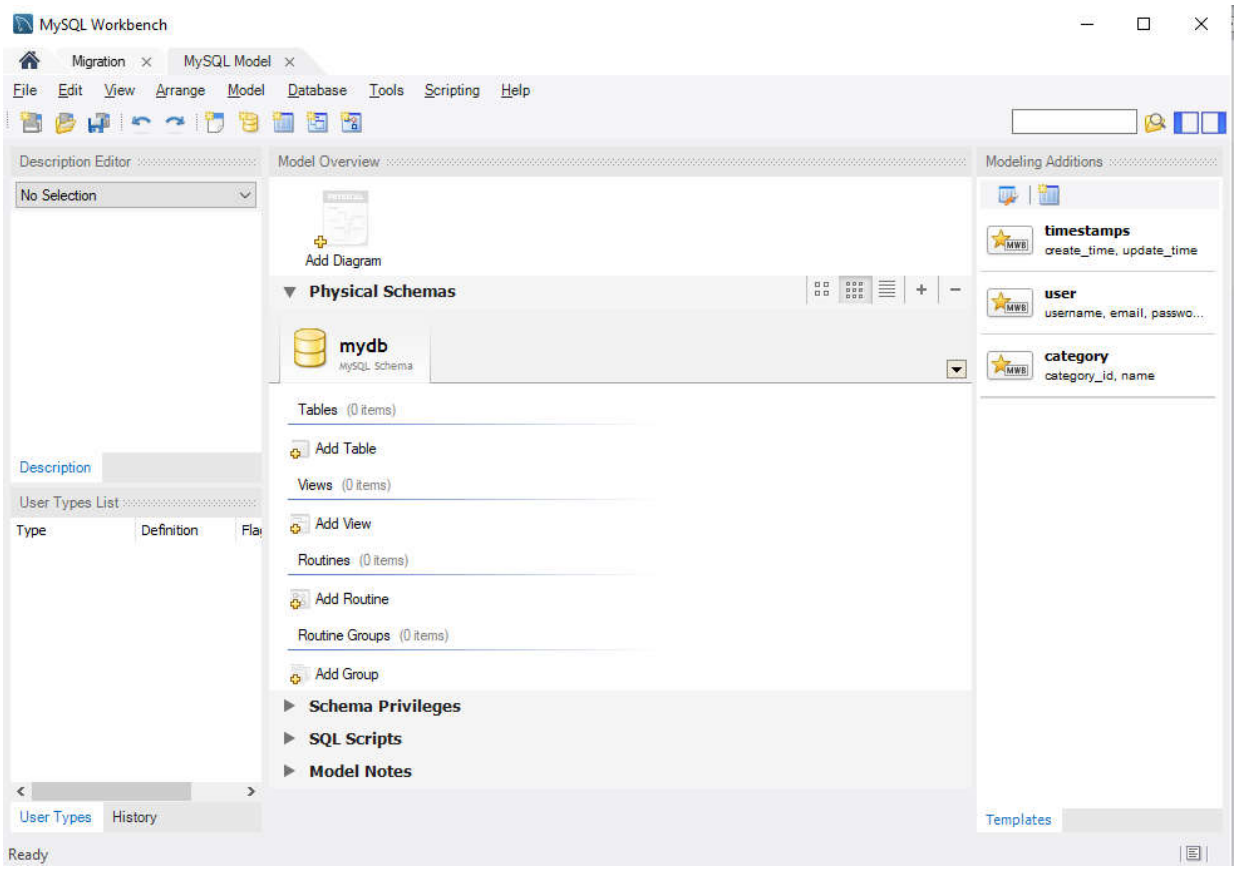


Рис. 4.4. Вікно редактора *Workbench* для створення нової бази даних

За замовчуванням нова модель бази даних містить схему *MyDB*, яку можна перейменувати. *Workbench* має свій особливий формат для збереження моделі бази даних: *mwb*. Модель складається з декількох частин: графічної і фізичної.

Графічна схема бази даних складається з *ERR*-діаграм. Фізична схема містить все необхідне для визначення бази даних: таблиці, стовпці, типи, індекси, обмеження і т.ін. Кожен об'єкт, доданий в графічну модель, також відображається у фізичній схемі.

EER означає *Extended Entity Relationship* (розширена діаграма сутність-зв'язок).

База даних складається з таблиць. Таблиці можна створювати як у фізичній схемі так і у графічній.

4.3.1 Створення таблиці у фізичній схемі

Створення таблиці у фізичній схемі відбувається натисканням на кнопку *Add Table* (Додати таблицю). Редактор таблиць відкриється у вигляді вкладки, як показано на рис.4.5.

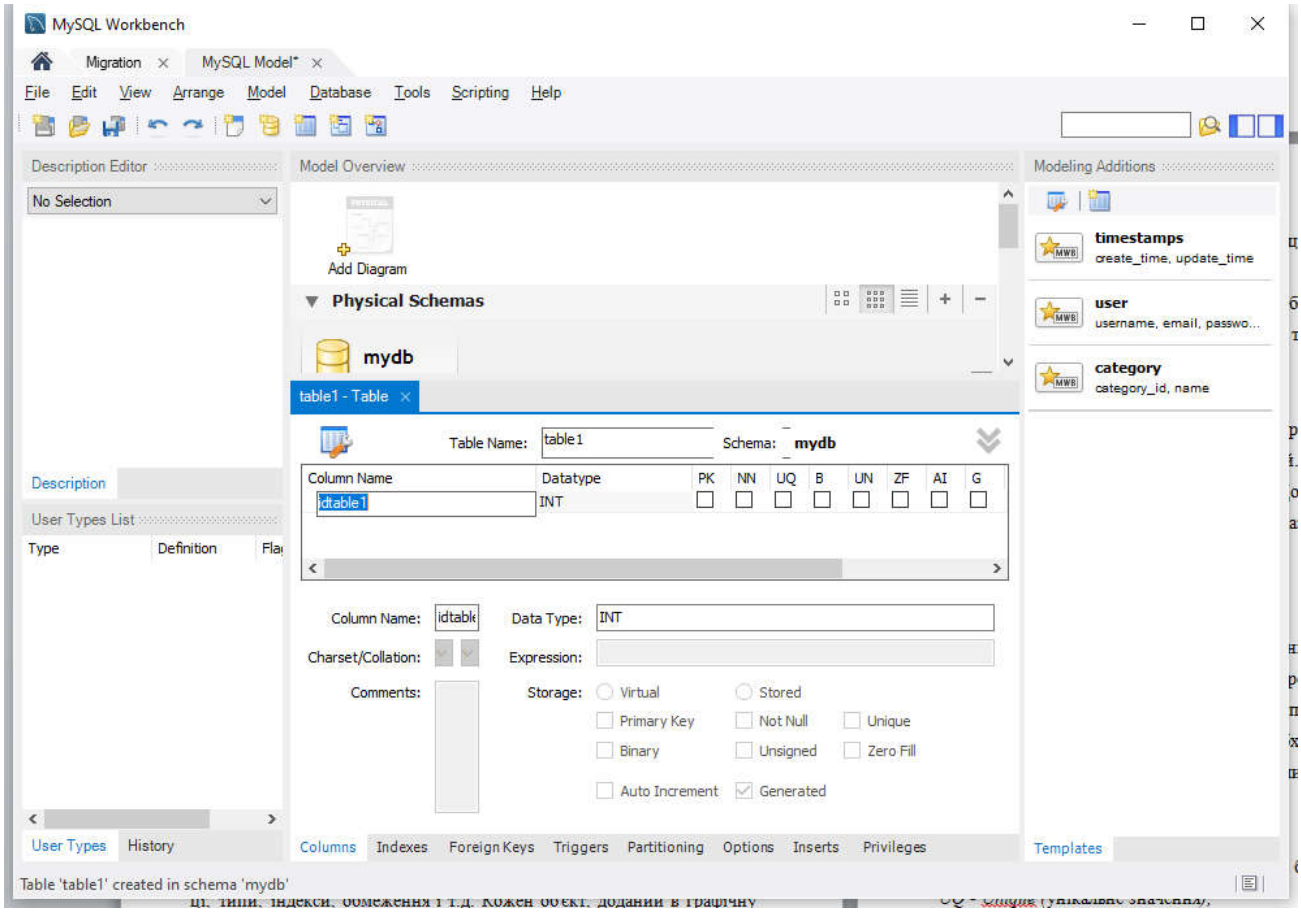


Рис. 4.5. Вікно редактора таблиць *Workbench*

Використовуючи редактор таблиць можна змінити ім'я таблиці та створити її структуру. Ми можемо вибрати тип даних (є випадаючий

список всіх типів даних *MySQL*), привласнити, якщо це необхідно, значення за замовчуванням, і у нас є сім прапорців, щоб відзначити будь-яку з наступних властивостей:

• *PK – Primary key (*первинний ключ*);*

• *NN - Not null (null,* тобто значення повинно бути визначено*);*

• *UQ - Unique (*унікальне значення*);*

• *BIN - Binary (*двійкове значення*);*

• *UN - Unsigned (*беззнакове значення*);*

• *ZF - Zero fill (*заповнене нулями*);*

• *AI - Autoincrement (*автоінкремент – автоматичне збільшення на 1).

Натиснувши на кнопку *Add Diagram* можна побачити, що створена таблиця не відображається у вікні діаграми. Для відображення таблиць, створених у фізичній схемі у вікні діаграм необхідно обрати пункт меню: *Model/Create Diagram from Catalog Objects* (Модель/ Створення діаграми з каталогу об'єктів).

4.3.1 Створення таблиці у графічній схемі

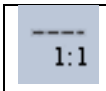
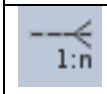
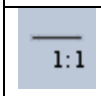
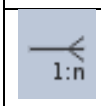
Для створення таблиці у вікні діаграми необхідно натиснути *Add Diagram,* та обрати в колонці зліва значок таблиці і потім кликнути у будь-якому місці вікна.

Два рази кликнувши на таблиці відкриємо вікно редактора.

4.3.1 Створення зав’язків між таблицями у графічній схемі

У вертикальній панелі інструментів зліва є шість інструментів, доступних для створення зав’язків між таблицями, як показано в табл.4.1.

Таблиця 4.1. Типи зав’язків між таблицями *Workbench*

	один до одного, неідентифікуючий
	один до багатьох, неідентифікуючий
	один до одного, ідентифікуючий
	один до багатьох, ідентифікуючий
	багато до багатьох, ідентифікуючий
	один до багатьох, з використанням існуючих атрибутів

Зв'язок є ідентифікуючим тільки в тому випадку, коли частина зовнішнього ключа дочірньої сутності є первинним ключем батьківської, тобто запис у дочірній таблиці не може існувати без відповідного запису у батьківській таблиці. В іншому випадку зв'язок завжди неідентифікуючий.

Для створення зв’язку між таблицями необхідно кликнути по значку, а потім по таблицям, які необхідно зв’язати. Для зв’язку *один до багатьох* спершу треба кликнути по таблиці багато, а потім по таблиці один. У дочірній таблиці автоматично створюється зовнішній ключ, який зв’язаний з первинним ключем у батьківській таблиці. Тип даних зовнішнього ключа той самий, що і у первинного ключа батьківської таблиці.

Приклад створеного зв'язку між таблицями показано на рис.4.6.

Якщо користувач хоче створити зовнішні ключі самостійно, їх необхідно ввести до таблиці на етапі її створенні та обрати в редакторі діаграм нижній шостий значок  (або можна натиснути 6). Спочатку

буде запропоновано обрати поле зовнішнього ключа в дочірній таблиці, а потім поле первинного ключа в батьківській.

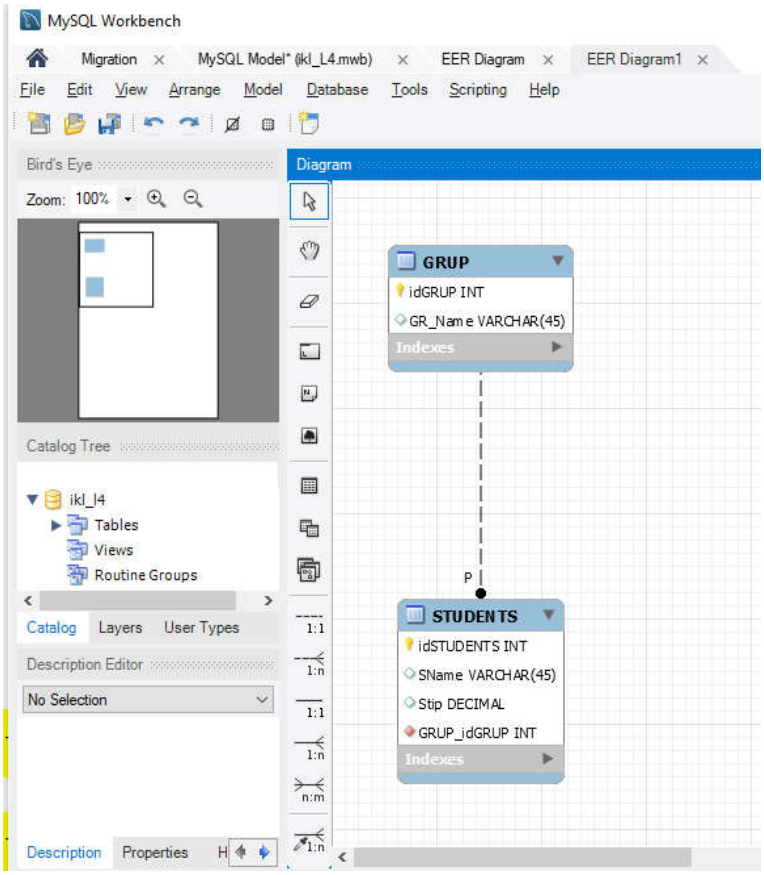


Рис.4.6. Зв'язок між таблицями типу один до багатьох.

За замовчуванням зв'язок має властивості *ON UPDATE NO ACTION* та *ON DELETE NO ACTION*. Для того, щоб змінити дії при модифікації або видалені первинного ключа в батьківській таблиці необхідно виконати відповідні дії у редакторі дочірньої таблиці (див. рис.4.6).

ERR-діаграму можна експортувати в графічний файл за допомогою команд меню *"File → Export"*, а потім один з варіантів *(PNG, SVG, PDF, PostScript File)*.

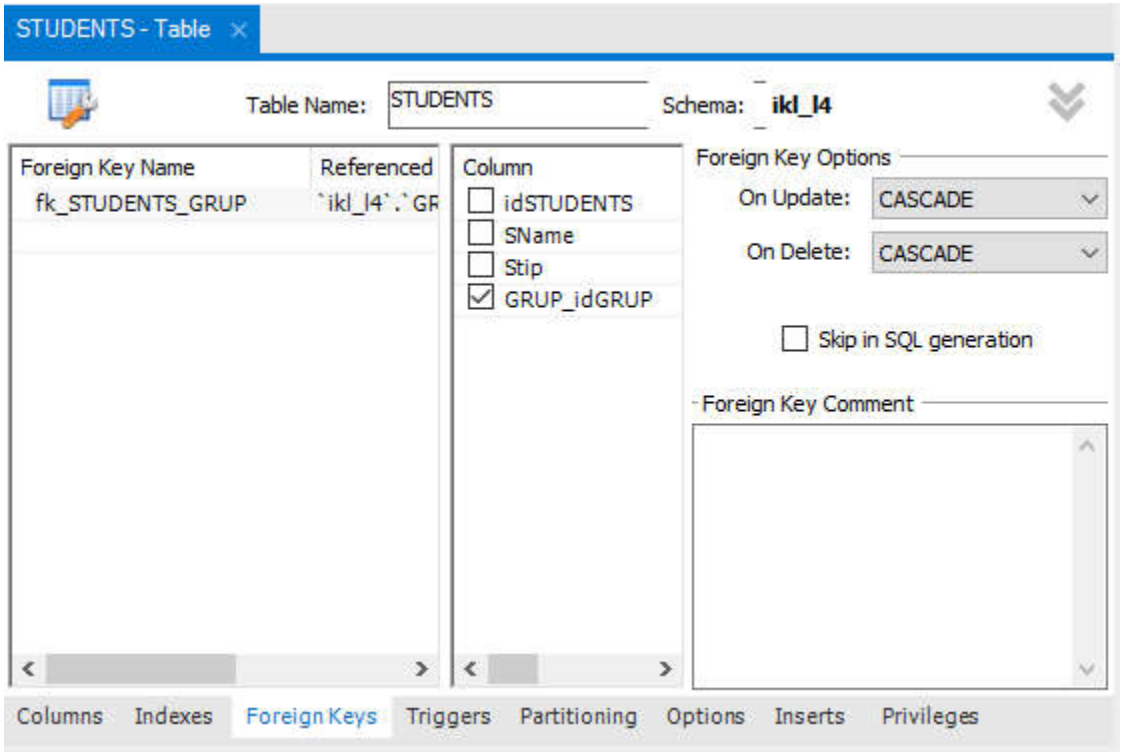


Рис.4.6. Встановлення властивостей зовнішнього ключа

4.4 Перенос локальної моделі на сервер

Створену *mwb* модель необхідно перенести на сервер. Найшвидший шлях для того, щоб схема даних з *MySQL Workbench* потрапила на сервер – створення *SQL*-дампа *mwb* моделі. Для цього відкриваємо нашу модель і вибираємо "*File → Export → Forward Engineer SQL CREATE Script...*" *(Ctrl + Shift + G):*

Якщо потрібно записати дамп в файл, вказуємо шлях до файлу в полі "*Output SQL Script File*" (якщо залишити поле порожнім, *SQL* скрипт можна буде скопіювати на останньому кроці в буфер обміну).

Галочка "*Generate INSERT Statements for Tables*" включає в дамп дані, які були внесені до таблиць. Після натискання "*Next*" ми бачимо список того, що взагалі можна експортувати. Для експорту таблиць вибираємо "*Export MySQL Table Objects*", а щоб експортувати їх вибірково, натискаємо "*Show Filter*" і вибираємо потрібні нам таблиці.

Натиснувши "*Next*" ми побачимо у вікні готовий скрипт *SQL*, звідки зможемо скопіювати його в буфер обміну або ж записати в який-небудь файл.

Тепер цей скрипт може бути використаний для створення бази даних на сервері. Наприклад, його можна завантажити у командному рядку *mysql* клієнта:

```
mysql> SOURCE scriptName.sql,
```

або підключитися до серверу *MYSQL* за допомогою *Workbench* та відкрити збережений скрипіт командою меню "*File → Open SQL Script" (Ctrl + Shift + O):*

Щоб виконати сценарій (script) необхідно кликнути значок блискавки і модель буде створена.

4.5 Порядок виконання лабораторної роботи

4.5.1 Запускаємо *Workbench* (без підключення до серверу) та створюємо нову модель ("*File → New Model"* або *Ctrl* +N)

4.5.2 Перейменовуємо схему за замовченням (mydb) в Lab4_**СВОЄ ПРИЗВИЩЕ**

4.5.3 Створюємо таблиці GR і STUDENT згідно завдання до лабораторної роботи № 3.

Таблицю GR створюємо у фізичній схемі.

Таблицю STUDENT створюємо у графічній схемі.

4.5.4 Створюємо зв’язок між таблицями у графічній схемі.

В нашому прикладі в одній групі може навчатися багато студентів, а один студент може навчатися не більш чим в одній групі, тому зв'язок між таблицями *один до багатьох*. Студент може в якийсь момент не бути записаний до жодної групи, тому цей зв'язок неідентифікуючий.

Необхідно розглянути два варіанти створення вторинного ключа у таблиці STUDENT (автоматичний і ручний).

4.5.5 Заповнюємо таблиці даними у редакторі таблиць (3-4 строки).

4.5.6 Зберігаємо отриману модель у файл з розширенням .mwb ("*File → Save Model As")*

Експортуємо *ERR*-діаграму в графічний файл за допомогою команд меню *"File → Export"*

4.5.7 Створюємо *SQL*-дампа нашої *mwb* моделі, передбачивши збереження введених даних.

4.5.8 Підключаємося до серверу *MySQL* у програмі *Workbench* за допомогою свого облікового запису.

4.5.9 Відкриваємо збережений SQL скрипіт та виконуємо його.

4.5.6 Підготувати звіт з лабораторної роботи, який повинен включати *ERR*-діаграму і SQL-скрипт розробленої моделі.

Контрольні питання

1) Що робити, якщо таблиця створена і фізичній схемі не відображається у вікні діаграм?
2) Що таке ідентифікуючий та неідентифікуючий зв’язок?
3) Як створити зовнішній ключ самостійно?
4) Яке розширення має модель БД?
5) Як перенести створену модель з локального хоста на сервер?

ЛАБОРАТОРНА РОБОТА № 5

ПРОЄКТУВАННЯ БАЗИ ДАНИХ МЕТОДОМ СЕМАНТИЧНОГО МОДЕЛЮВАННЯ (*ER*-моделювання)

Мета роботи – Отримання навичок створення бази даних на основі *ERR*-моделі в *MySQL Workbench*.

5.1 Створення *ERR*-діаграми в *MySQL Workbench*

Створити нову базу даних в редакторі *MySQL Workbench* можна за допомогою меню *"File → New Model"* **(***Ctrl* + N**)** див.рис 5.1.

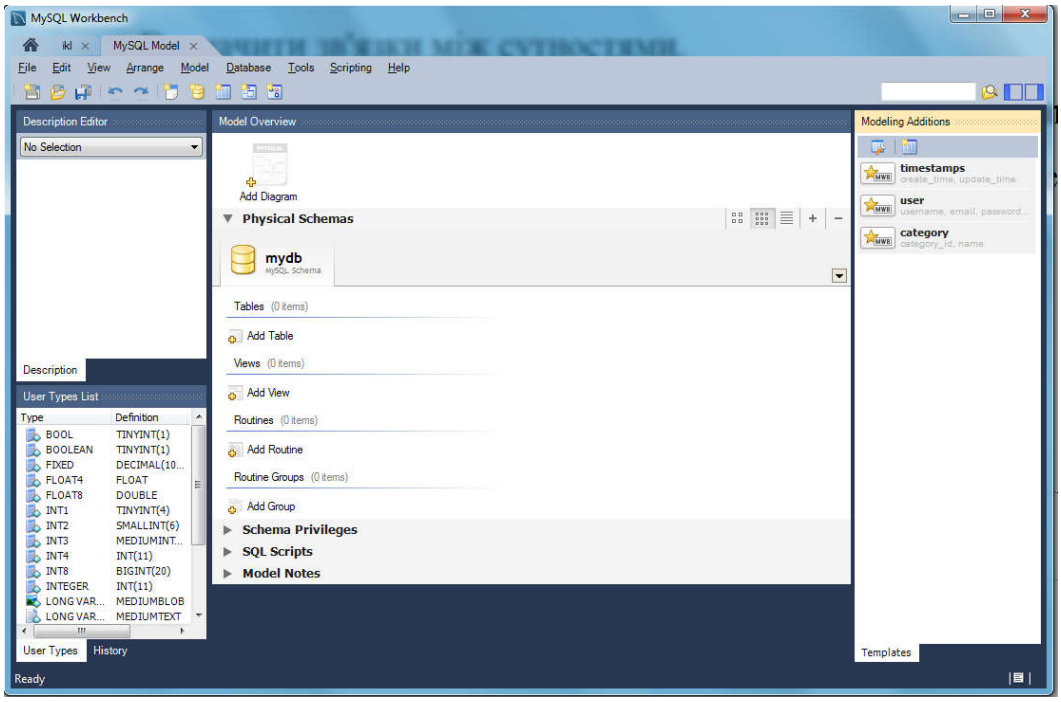


Рис.5.1. Вікно редактора в *mySQL Workbench* при створенні нової моделі

У вікні *Model Overview* можна будувати як логічну (у вигляді ERR-діаграми) та і фізичну моделі бази даних.

Для переходу до створення ERR-діаграми клацнемо по пункту *Add diagram*, завантажиться порожнє вікно діаграми (рис 5.2).

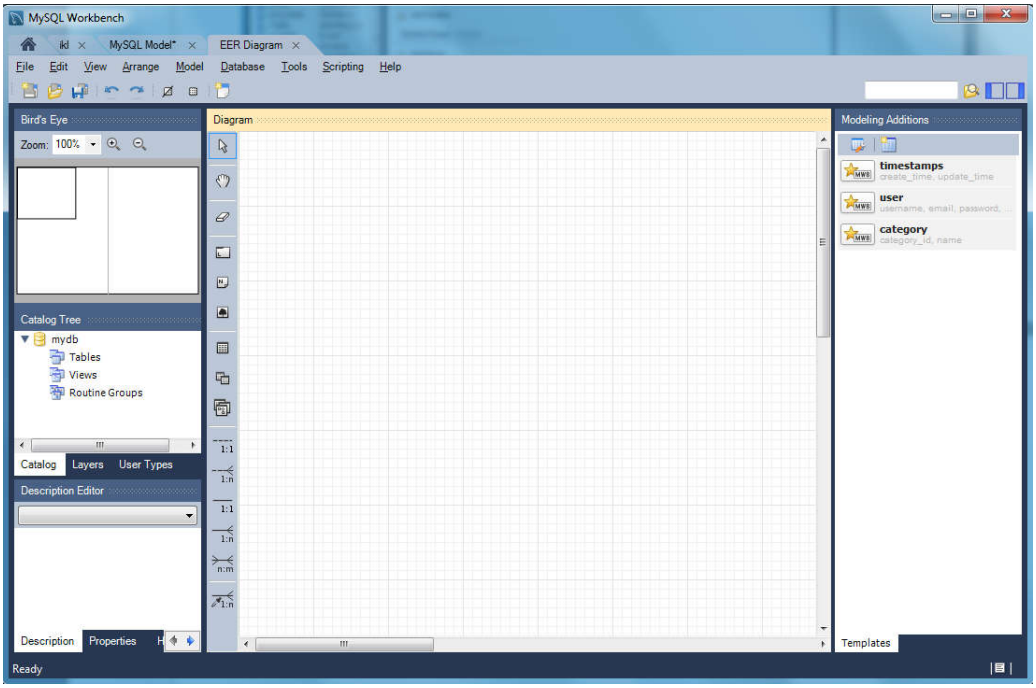


Рис.5.2.

Перейменуємо базу даних за замовченням (*mydb*) в LAB5_СВОЕ ПРИЗВИЩЕ

Створити нову таблицю можна за допомогою піктограми  . Потрібно клацнути по цій піктограмі, а потім клацнути в робочій області діаграми. На цьому місці з'явиться таблиця з назвою за замовчуванням table1. Подвійне клацання по цій таблиці відкриває вікно редагування, в якому можна змінити ім'я таблиці та налаштувати її структуру.

Створимо таблицю PREDMET (предмети) з наступними атрибутами:

PNum – унікальний номер дисципліни (*PRIMERY KEY*);

PName – назва дисципліни;

Semestr – семестр в якому викладається дисципліна;

Hours – кількість годин відведених на дисципліну.

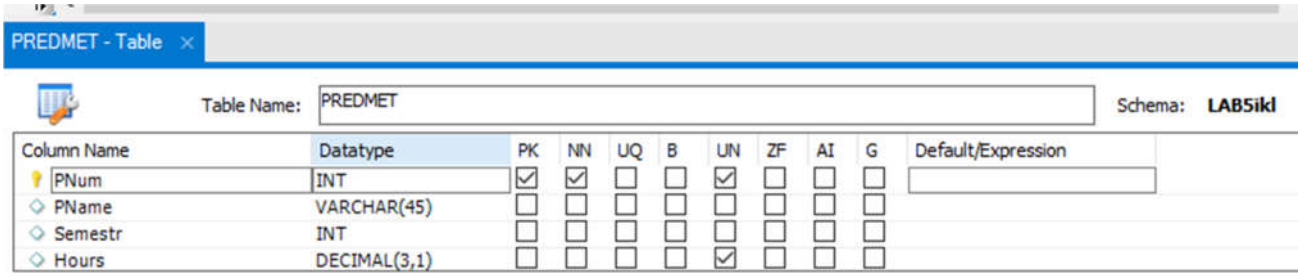


Рис.5.3. Структура таблиці PREDMET

Кожен атрибут (стовпчик) таблиці обов'язково має певний тип даних (Datatype) та може мати додаткові властивості:

- PK (primary key) – первинний ключ;
- NN (not null) – не допускаються порожні значення;
- UQ (unique) – значення має бути унікальним в межах стовпця;
- BIN – бінарний стовпець;
- UN (unsigned) – стовпець має лише позитивні значення;
- ZF – порожні числові значення доповнюються нулями до заданої довжини;
- AI (auto incremental) – Для стовпця первинного ключа з типом INT властивість означає, що поле буде автоматично заповнюватися натуральними числами: 1, 2, 3, і т.д;
- DEFAULT – значення за замовчуванням, тобто, значення, яке при додаванні нового рядка в таблицю автоматично вставляється в комірку сервером, якщо користувач залишив комірку порожньою.

Створимо таблицю TEACHER (Викладач) з наступною структурою (див рис. 5.4):

TNum – унікальний номер викладача;

TFam – прізвище викладача;

TName – ім'я викладача.

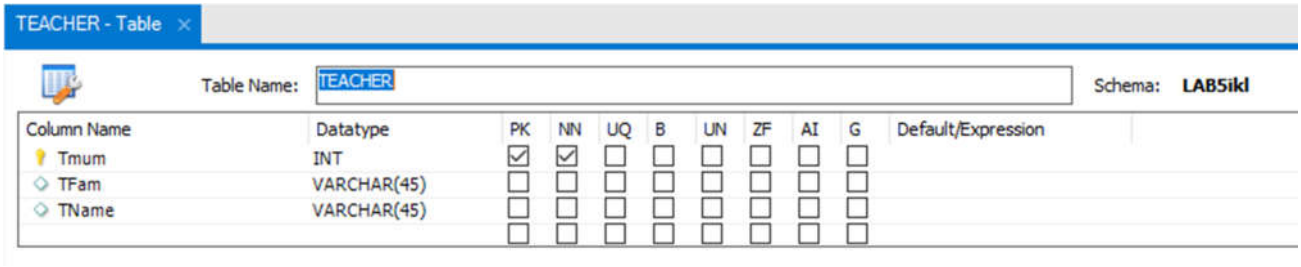
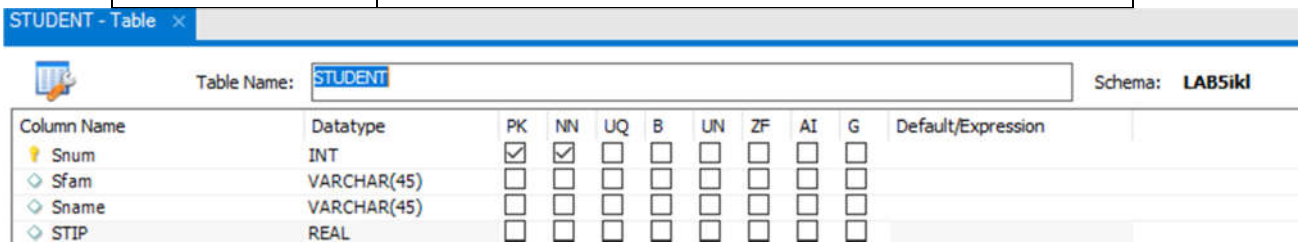
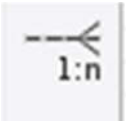


Рис. 5.4. Структура таблиці TEACHER

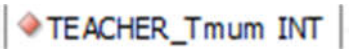
Аналогічно створимо таблицю STUDENT

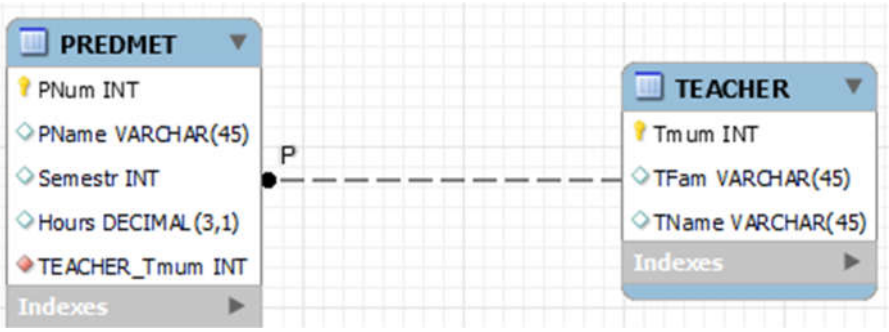
Таблиця STUDENT (Студент)	
SNum	**Код студента**
SFam	Прізвище студента
SName	Ім’я студента
Stip	Розмір стипендії студента



Тепер зв'яжемо ці таблиці. Кожен викладач читає декілька предметів, але кожен предмет викладає лише один викладач. Зв'язок між таблицями TEACHER і PREDMET один до багатьох. Таблиця TEACHER називається батьківською, а PREDMET – дочірньою. Для встановлення зв’язку використовуємо кнопку 

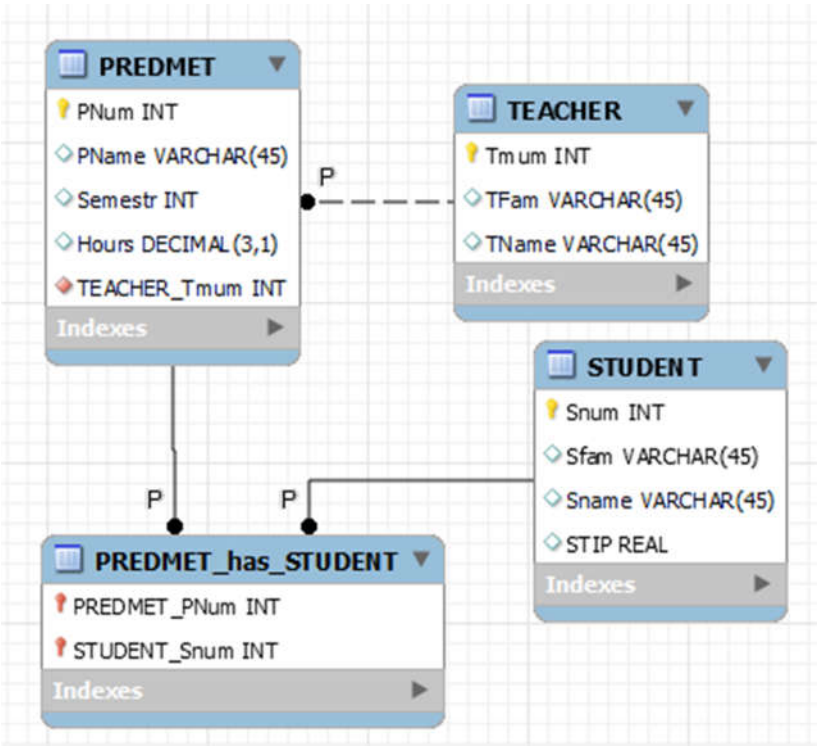
. Спочатку обираємо таблицю PREDMET, а потім таблицю TEACHER. В таблиці PREDMET автоматично створюється зовнішній ключ (FOREIGN KEY):



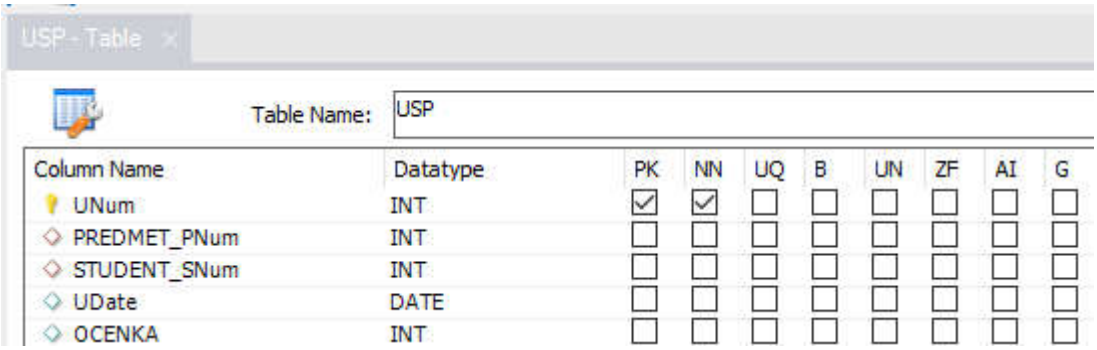


Один студент вивчає декілька предметів і по кожному предмету кожен студенти отримує оцінку. Тому зв'язок між таблицями STUDENT та PREDMET «багато до багатьох». Згідно з правилами проєктування баз даних такий зв'язок повинен бути змінений на два зв’язки типу «один до багатьох». Створюємо зв’язок між таблицями STUDENT та PREDMET кнопкою 

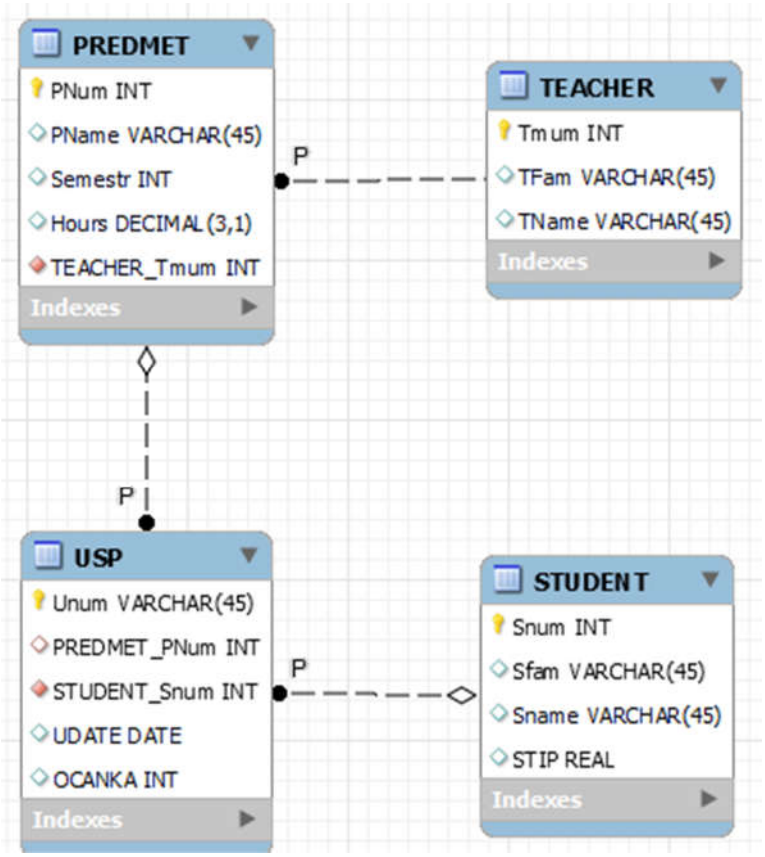
. *Workbench* автоматично створює нову таблицю, що з’єднує таблиці STUDENT та PREDMET.



Перейменовуємо цю таблицю в USP. Змінюємо структуру таблиці відповідно до схеми:



При цьому схема бази даних повинна прийняти наступний вигляд:



Заповнимо таблиці бази даних

STUDENTS

SNUM	SFAM	SNAME	STIP
3412	Поляков	Анатолій	1000
3413	Старова	Любов	600
3414	Гриценко	Володимир	0
3415	Котенко	Анатолій	0
3416	Нагорний	Євген	600

TEACHERS

TNUM	TFAM	TNAME
4001	Викулина	Валентина
4002	Костиркін	Олег
4003	Казанко	Віталій
4004	Позднякова	Любов
4005	Загорайчук	Ігор

PREDMET

PNUM	PNAME	TNUM	HOURS	SEMESTR
2001	Фізика	4001	32	1
2002	Хімія	4002	64	1
2003	Математика	4003	64	1
2004	Філософия	4005	16	2
2005	Економіка	4004	16	3

USP

UNUM	OCENKA	UDATE	SNUM	PNUM
1001	5	10.06.2020	3412	2001
1002	4	10.06.2020	3413	2003
1003	3	11.06.2020	3414	2005
1004	4	12.06.2020	3412	2003
1005	5	12.06.2020	3416	2004

Останнім етапом моделювання даних є експорт розробленої моделі на сервер MySQL.

1. В головному меню обираємо *Database — Forward Engineering*
2. У вікні *Set Parameters for Connecting to DBMS* вибираємо з'єднання з потрібним хостом і тиснемо Next
3. у вікні *Set Options for Database to be Created* проставляємо потрібні галочки (див. рис.5.3) і тиснемо *Next*
4. У наступному вікні *Select Objects to Forward Engineer* обираємо об’єкти бази даних для експорту
5. У вікну *Review the SQL Script to be Executed* бачимо сформований *SQL*-скріпт, який можна скопіювати в файл або в буфер обміну. Натиснувши *Next* синхронізуємо нашу модель з сервером

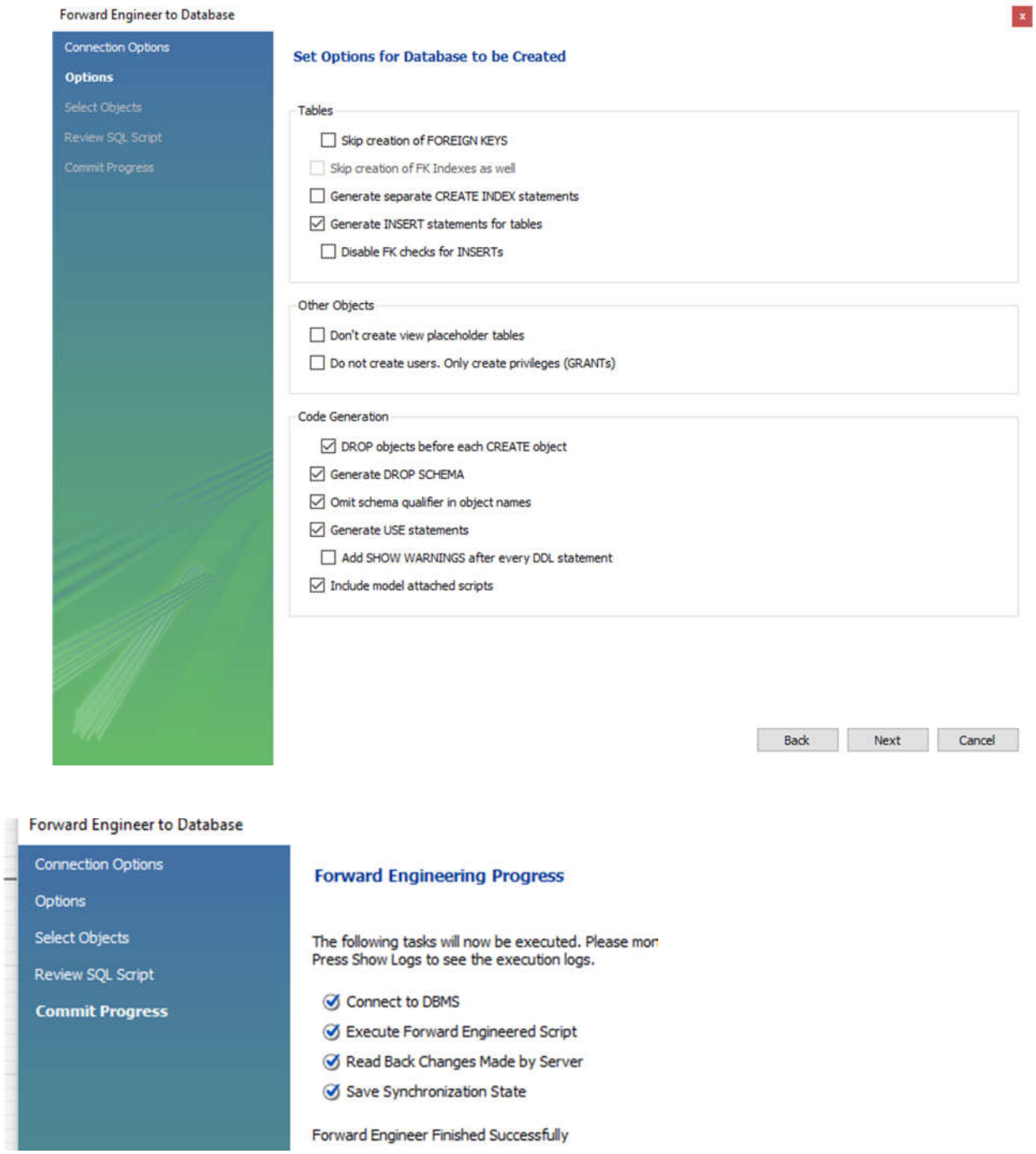


Рис.5.3 Пернос моделі на сервер.

Контрольні питання

1) Як створити зв’язок «багато до багатьох»?
2) Що необхідно зробити для переносу внесених у БД даних на сервер?
3) Що означає галочка *DROP objects before each create object*?

ЛАБОРАТОРНА РОБОТА № 6

УЯВЛЕННЯ (VIEW) В *MYSQL*

Мета роботи – робота з уявленнями (view) *MYSQL*

6.1 Загальні положення

Уявлення (*VIEW*) – об'єкт бази даних, що є результатом виконання запиту до бази даних, визначеного за допомогою оператора *SELECT*, в момент звернення до уявлення.

Уявлення *VIEW* — це віртуальні таблиці, які не зберігають власні дані, а відображають дані, збережені в інших таблицях. Уявлення може містити всі або кілька рядків таблиці. Уявлення *MySQL* може відображати дані з однієї таблиці або багатьох таблиць.

Для створення уявлення використовується оператор *CREATE VIEW*, синтаксис якого має наступний вид:

```
CREATE [OR REPLACE]
[ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
VIEW view_name [(column_list)]
AS select_statement
[WITH [CASCADED | LOCAL] CHECK OPTION]
```

де `view_name` – ім'я уявлення, що створюється; `select_statement` – оператор *SELECT*, що вибирає дані з таблиць і / або інших уявлень, які будуть міститися в уявлені.

Оператор *CREATE VIEW* містить 4 необов'язкові конструкції:

OR REPLACE — при використанні даної конструкції у разі існування уявлення з таким ім'ям старе буде видалено, а нове створено. В іншому випадку

виникне помилка, яка інформує про існування подання з таким ім'ям і нове уявлення створено не буде. Слід зазначити одну особливість - імена таблиць та уявлень у межах однієї бази даних мають бути унікальні, тобто. не можна створити уявлення з ім'ям вже існуючої таблиці. Однак конструкція OR *REPLACE* діє тільки на уявлення та заміщати таблицю не буде;

ALGORITM – визначає алгоритм, що використовується при зверненні до подання;

column_list – задає імена полів уявлення;

WITH CHECK OPTION – при використанні даної конструкції всі рядки, що додаються або змінюються, перевірятимуться на відповідність визначенню уявлення. У разі невідповідності ця зміна не буде виконана. Зверніть увагу, що при використанні даної конструкції для поновлення уявлення виникне помилка і уявлення не буде створено.

Приклад створення уявлення в консольному клієнті *MySQL*:

За замовчуванням колонки подання мають ті ж імена, що й поля, що обертаються оператором *SELECT* у визначенні подання. При явному указанні імен полів уявлення *column_list* повинен включати по одному імені кожного поля розділених комою.

```
CREATE DATABASE PRIMER1;
USE PRIMER1;
CREATE TABLE t (quantity INT, price INT);
INSERT INTO t VALUES(3, 50);
CREATE VIEW v AS SELECT quantity, price,
quantity*price AS value FROM t;
SELECT * FROM v;
```

```
mysql> USE PRIMER1;
Database changed
mysql> CREATE TABLE t (quantity INT, price INT);
Query OK, 0 rows affected (0.03 sec)

mysql> INSERT INTO t VALUES(3, 50);
Query OK, 1 row affected (0.00 sec)

mysql> CREATE VIEW v AS SELECT quantity, price, quantity*price AS value FROM t;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM v;
+----------+-------+-------+
| quantity | price | value |
+----------+-------+-------+
|        3 |    50 |   150 |
+----------+-------+-------+
1 row in set (0.00 sec)
```

6.2 Порядок виконання лабораторної роботи

6.2.1 Підключитися через консольний клієнт *mysql* до серверу

6.2.2 Обрати поточною свою базу даних, що була створена в лабораторній роботі 5 (LAB5_СВОЕ ПРИЗВИЩЕ)

6.2.3. Створити уявлення (*VIEW*), в якому зберігаються студенти відмінники з таблиці USP на ім’я ВІДМИННИКИ

use Им'я+бази_даних ;

```
CREATE OR REPLACE
VIEW ВІДМИННИКИ AS
SELECT  SNUM, PNUM, OCENKA
FROM  usp
WHERE (OCENKA = 5);
```

6.2.4 Перевірити створене уявлення за допомогою оператора SELECT

	SNUM	PNUM	OCENKA
▶	3412	2001	5
	3413	2003	5
	3416	2004	5

6.2.5 Вийти з консольного клієнту та підключитися до серверу через WORKBENCH

6.2.5 Створити уявлення ПРОТОЧНА УСПІШНІСТЬ

```
USE Ім'я+бази_даних;
CREATE OR REPLACE
    VIEW `ПОТОЧНА УСПІШНІСТЬ` AS
SELECT USP.SNUM,USP.PNUM,USP.OCENKA
FROM USP;
```

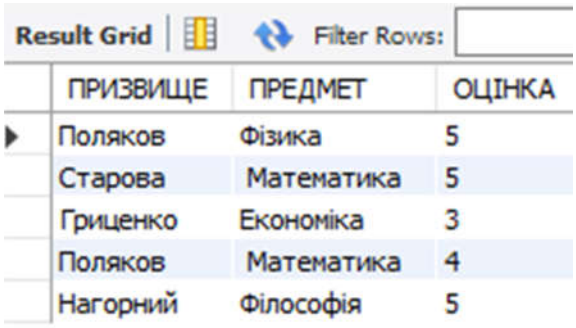
Result Grid | Filter Ro

	SNUM	PNUM	OCENKA
▶	3412	2001	5
	3413	2003	5
	3414	2005	3
	3412	2003	4
	3416	2004	5

6.2.6 Замінити НОМЕР СТУДЕНТА на його ПРИЗВИЩЕ, а НОМЕР ПРЕДМЕТА на його НАЗВУ використовуючи зв’язки між таблицями

```
USE Имя+бази_даних;
CREATE OR REPLACE
    VIEW `ПОТОЧНА УСПІШНІСТЬ` AS
Select STUDENTS.SFAM AS `ПРИЗВИЩЕ`, PREDMET.PNAME AS
`ПРЕДМЕТ`, USP.OCENKA AS `ОЦІНКА`
FROM USP
```

```
JOIN STUDENTS
ON STUDENTS.SNUM=USP.SNUM
JOIN PREDMET
ON PREDMET.PNUM=USP.PNUM;
```



Result Grid | Filter Rows:

ПРИЗВИЩЕ	ПРЕДМЕТ	ОЦІНКА
Поляков	Фізика	5
Старова	Математика	5
Гриценко	Економіка	3
Поляков	Математика	4
Нагорний	Філософія	5

6.2.7 Змінити наприклад назву Предмету МАТЕМАТИКА на ВИЩА МАТЕМАТИКА в таблиці PREDMET і перевірити зміни у уявлені

Контрольні питання

1) Що таке уявлення (VIEW)?
2) Як видалити старе уявлення перед створенням нового?
3) Яку назву мають атрибути уявлення за замовчуванням?
4) Як змінити назву атрибутів уявлення?

ЛАБОРАТОРНА РОБОТА №7

ПРОЦЕДУРИ MYSQL

Мета роботи – робота зі збереженими процедурами *mysql.*

7.1 Загальні положення

Збережена процедура (*stored procedure*) – попередньо скомпільований набір операторів мови *SQL*, який зберігається на сервері.

Часто при виконанні рутинних операцій потрібно здійснювати послідовність однакових запитів. Збережені процедури дозволяють об'єднати послідовність таких запитів і зберегти їх на сервері. Після цього клієнтам не доведеться посилати серверу послідовність запитів, достатньо надіслати один запит на виконання збереженої процедури.

Загальний синтаксис створення збереженої процедури має такий вигляд

```
DELIMITER //
CREATE PROCEDURE sp_name ([parameter [,...]])
routine_body
```

Оператор *DELIMITER* // змінює признак кінця оператора *SQL*. За замовчуванням такою ознакою є символ точка з комою, який сприймається як сигнал до надсилання запиту на сервер. Однак у збереженій процедурі можуть біти вирази, що закінчуються точкою з комою. Тому необхідно на час створення процедури змінити його, наприклад на //.

Тут *sp_name* ім'я збереженої процедури, в дужках, наступних за ім'ям, передається необов'язковий список параметрів, перерахованих через кому. Кожен *parameter* дозволяє передати в процедуру або з неї вхідні дані або результат роботи функції і має наступний синтаксис:

[*IN |OUT |INOUT] param_name type*

Ключове слово *IN* перед іменем параметра вказує на те, що в цьому параметрі передаватиметься значення в процедуру. Ключове слово *OUT* вказує на те, що в даному параметрі буде розміщене значення, що повертатиме процедура. *INOUT* вказує на те, що деяке значення параметра передаватиметься у процедуру, там буде змінюватися, і результат повернеться у тому ж параметрі.

Якщо ні один із модифікаторів не вказано, вважається, що параметр об'явлено з ключовими словами *IN*.

Тілом процедури (*routine_body*) є складовий оператор *BEGIN...END*, всередині якого можуть розташовуватися інші оператори: *IF…THEN…ELSE , REPEAT…UNTIL, WHILE, LOOP, LEAVE, ITERATE, CASE, RETURN* та коментарі.

Для виклику збереженої процедури використовується оператор *CALL*. Його вигляд є таким:

CALL sp_name (значення_вхідної_змінної [,...n], @змінна [,...n]}

7.2 Завдання до лабораторної роботи

Створити процедуру, яка вносить дані в таблицю TEACHERS

Підключаємося до серверу через Workbench .

Обираємо поточною свою базу даних (лабораторна робота 5)

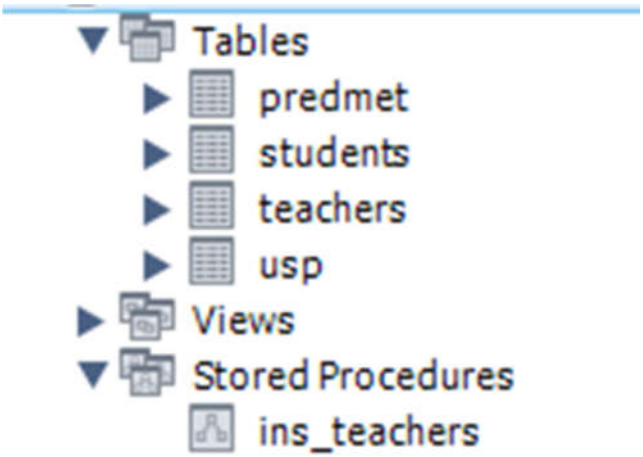
Відкриваємо вікно для вводу команд SQL (*Fil → New Query Tab*)

Виконуємо наступний код:

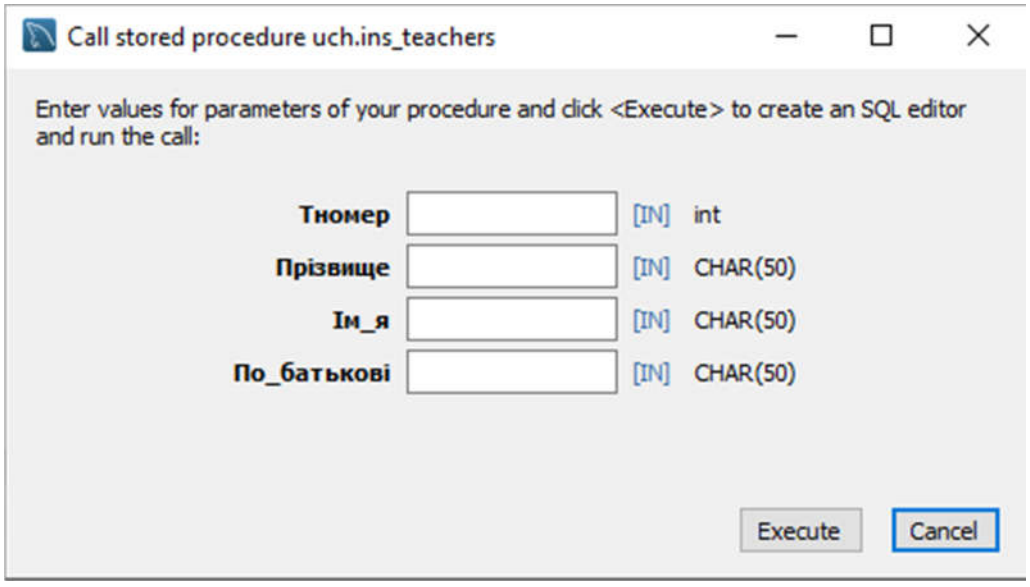
```
DELIMITER //
DROP PROCEDURE IF EXISTS ins_teachers //
CREATE  PROCEDURE ins_teachers(Тномер int, Прізвище
CHAR(50), Ім_я CHAR(50), По_батькові CHAR(50))
begin
```

```
        INSERT INTO TEACHERS(TNUM,TFAM,TNAME,TOTCH)
value(Тномер,Прізвище,Ім_я,По_батькові);
end
```

Перевіряємо наявність процедури ins_teachers



Викликаємо збережену процедуру та вводимо нового викладача



Вводимо дані нового викладача і перевіряємо чи з'явився він у таблиці TEACHERS.

Аналогічно створити ПРОЦЕДУРУ для вводу нового студента в таблицю STUDENTS.

Контрольні питання

1) Що таке збережена процедура?
2) Яку функцію виконує оператор *DELIMITER*?
3) Як відокремлюється тіло процедури?
4) Які оператори можуть бути розташовані у тілі процедури?
5) Як автоматично створити процедуру замість вже існуючої?

Навчальне видання

Методичні вказівки
до виконання лабораторних робіт
з дисципліни
«Організація баз даних»

для студентів спеціальності
174 «Автоматизація, комп'ютерно-інтегровані
технології та робототехніка»

Укладачі:
КРАСНІКОВ Ігор Леонідович
БАБІЧЕНКО Анатолій Костянтинович
ЛИСАЧЕНКО Ігор Григорович
ШУТИНСЬКИЙ Олексій Григорович

Відповідальний за випуск Подустов М.О.
Роботу рекомендував до друку Дуднік О.В.

В авторській редакції

План 2023 р., поз. 138

Підп. до друку 16.02.2023.
Гарнітура Times New Roman.

Видавничий центр НТУ «ХПІ».
Свідоцтво про державну реєстрацію ДК № 5478 від 21.08.2017 р.
61002, Харків, вул. Кирпичова, 2

Самостійне електронне видання