



Міністерство освіти і науки України
ДЕРЖАВНИЙ БІОТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ
ННІ Кіберпорт
Кафедра інформаційних технологій,
кібернетики та захисту інформації

ПРОЄКТУВАННЯ ТА РОЗРОБКА БАЗ ДАНИХ В MYSQL WORKBENCH

Методичні вказівки
до виконання практичних робіт

для здобувачів бакалаврського рівня вищої освіти
за спеціальністю 163 «Біомедична інженерія»

Затверджено
рішенням Науково-методичної ради
ННІ «Енергетики, робототехніки та
комп'ютерних технологій»
Протокол №1 від 31.10.2024 р.

Харків
2024

УДК 005.931

Схвалено на засіданні кафедри
інформаційних технологій, кібернетики та захисту інформації
Протокол № 3 від 20.10.2023 р.

Рецензенти:

Машталір С.В., доктор технічних наук, професор, професор кафедри інформатики, Харківського національного університету радіоелектроніки;

Коваленко С.М., кандидат технічних наук, доцент, доцент кафедри програмної інженерії та інтелектуальних технологій управління, Національного технічного університету «Харківський політехнічний інститут».

Проектування та розробка баз даних в MySQL Workbench: методичні вказівки до виконання практичних робіт для здобувачів бакалаврського рівня вищої освіти за спеціальністю 163 «Біомедична інженерія» / О.Д. Міхнова / – Електрон. дані. – Х. : ДБТУ, 2024. 70 с.

Методичні вказівки включають 8 практичних робіт та список літератури до них. Матеріал розкриває сутність проектування баз даних засобами MySQL Workbench: створення таблиць, взаємозв'язків між ними, побудову схеми даних та розробку запитів, адміністрування бази даних. Призначено для здобувачів технічних спеціальностей закладів вищої освіти, аспірантів, викладачів, наукових і практичних працівників.

УДК 004.932

Відповідальний за випуск: О.Д. Міхнова, к.т.н., доцент

© Міхнова О.Д., 2024

© ДБТУ, 2024

ЗМІСТ

Лабораторна робота №1	4
Тема: Встановлення програми MySQL Workbench. Огляд можливостей. Визначення структури бази даних.	4
Лабораторна робота №2.....	25
Тема: Основні операції з базою даних. Робота в командному рядку та за допомогою графічного інтерфейсу.....	25
Лабораторна робота №3.....	32
Тема: Операції над структурними елементами БД, додавання, редагування та видалення записів таблиці.....	32
Лабораторна робота №4.....	38
Тема: Первинний ключ і зовнішній ключ. Композитний (складений) первинний ключ.	38
Лабораторна робота №5.....	44
Тема: Створення та редагування моделі даних.....	44
Лабораторна робота №6.....	60
Тема: Оцінка ефективності запиту	60
Лабораторна робота №7.....	64
Тема: Міграція та резервування бази даних.....	64
Лабораторна робота №8.....	66
Тема: Адміністрування бази даних.....	66

ПЛАН СКЛАДАННЯ ЗВІТУ З ЛАБОРАТОРНО-ПРАКТИЧНИХ РОБІТ

Звіт до лабораторної роботи виконується на аркушах формату А4 з одного боку і повинен містити нижчевказані пункти.

1. Титульний лист встановленого зразка з темою роботи і прізвищами виконавців.
2. Мета роботи.
3. Скріншоти зі стислим описом порядку виконання роботи.
4. Висновок по роботі.

У висновку необхідно проаналізувати отримані практичні результати, порівняти їх з теоретичними даними, сформулювати і записати помічені особливості.

Лабораторна робота №1

Тема: Встановлення програми MySQL Workbench. Огляд можливостей. Визначення структури бази даних.

Мета: Провести інсталяцію MySQL Workbench на робочому місці. Оглянути функціональні можливості СУБД. Ознайомитись із наявними типами даних

Теоретичні відомості

При визначенні стовпців таблиці для них необхідно вказати тип даних. Кожен стовпець повинен мати тип даних. Тип даних визначає, які значення можуть зберігатися в стовпці, скільки вони будуть займати місця в пам'яті. MySQL надає наступні типи даних, які можна розбити на ряд груп.

Символьні типи

- **CHAR** : представляє стоку фіксованої довжини.

Довжина збереженої рядки указується в дужках, наприклад, CHAR(10)- рядок з десяти символів. І якщо в таблицю в даний стовпець зберігається рядок з 6 символів (тобто менше встановленої довжини в 10 символів), то рядок доповнюється пробілами і в підсумку все одно буде займати 10 символів

- **VARCHAR** : представляє стоку змінної довжини. Довжина збереженої рядки також указується в дужках, наприклад, VARCHAR(10). Однак на відміну від CHAR збережена рядок буде займати саме стільки місця, як необхідна. Наприклад, якщо визначення довжини в 10 символів, але в стовпець зберігається рядок в 6 символів, то збережена рядок так і буде займати 6 символів плюс додатковий байт, який зберігає довжину рядка.

Починаючи з MySQL 5.6 типи CHAR і VARCHAR за замовчуванням використовують кодування UTF-8, яка дозволяє використовувати до 3 байт для зберігання символу в залежності від мови (для багатьох європейських мов по 1 байту на символ, для ряду східно-європейських і близькосхідних - 2 байта, а для китайського, японського, корейського - по 3 байта на символ).

Ряд додаткових типів даних представляють текст невизначеної довжини:

- **TINYTEXT** : представляє текст довжиною до 255 байт.
- **TEXT** : представляє текст довжиною до 65 КБ.
- **MEDIUMTEXT** : представляє текст довжиною до 16 МБ
- **LARGETEXT** : представляє текст довжиною до 4 ГБ

числові типи

- **TINYINT** : представляє цілі числа від -127 до 128, займає 1 байт
- **BOOL** : фактично не представляє окремий тип, а є лише псевдонімом для типу TINYINT(1) і може зберігати два значення 0 і 1. Однак даний тип може також в якості значення приймати вбудовані константи **TRUE** (представляє число 1) і **FALSE** (надає число 0). Також має псевдонім **BOOLEAN**.

- **TINYINT UNSIGNED** : представляє цілі числа від 0 до 255, займає 1

байт

- **SMALLINT** : представляє цілі числа від -32768 до 32767, займає 2 байта
- **SMALLINT UNSIGNED** : представляє цілі числа від 0 до 65535, займає 2 байта
- **MEDIUMINT** : представляє цілі числа від -8388608 до 8388607, займає 3 байти
- **MEDIUMINT UNSIGNED** : представляє цілі числа від 0 до 16777215, займає 3
- **INT** : представляє цілі числа від -2147483648 до 2147483647, займає 4 байта
- **INT UNSIGNED** : представляє цілі числа від 0 до 4294967295, займає 4 байта
- **BIGINT** : представляє цілі числа від -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807, займає 8 байт
- **BIGINT UNSIGNED** : представляє цілі числа від 0 до 18 446 744 073 709 551 615, займає 8 байт
- **DECIMAL** : зберігає числа з фіксованою точністю. Даний тип може приймати два параметри precision і scale: DECIMAL(precision, scale).

Параметр precision представляє максимальну кількість цифр, які може зберігати число. Це значення повинно знаходитися в діапазоні від 1 до 65.

Параметр scale представляє максимальну кількість цифр, які може містити число після коми. Це значення повинно знаходитися в діапазоні від 0 до значення параметра precision. За замовчуванням воно дорівнює 0.

Наприклад, у визначенні у наступній колонці:

```
1 salary  
DECIMAL(5,  
2)
```

Число 5 - precision, а число 2 - scale, а тому цей стовпець може зберігати значення з діапазону від -999.99 до 999.99.

Розмір даних в байтах для DECIMAL залежить від значення, що зберігається. Даний тип також має псевдоніми **NUMERIC** , **DEC** , **FIXED** .

• **FLOAT** : зберігає дробові числа з плаваючою точкою одинарної точності від $-3.4028 * 10^{38}$ до $3.4028 * 10^{38}$, займає 4 байта

Може приймати форму FLOAT(M,D), де M – загальна кількість цифр, а D – кількість цифр після коми.

• **DOUBLE** : зберігає дробові числа з плаваючою точкою подвійної точності від $-1.7976 * 10^{308}$ до $1.7976 * 10^{308}$, займає 8 байт. Також може приймати форму DOUBLE(M,D), де M- загальна кількість цифр, а D – кількість цифр після коми.

Даний тип також має псевдоніми **REAL** і **DOUBLE PRECISION** , які можна використовувати замість DOUBLE.

Типи для роботи з датою і часом

- **DATE** : зберігає дати з 1 січня 1000 року до 31 деабря 9999 роки (с

"1000-01- 01" до "9999-12-31"). За замовчуванням для зберігання використовується формат уууу-мм-дд. Займає 3 байта.

• **TIME** : зберігає час від -838: 59: 59 до 838: 59: 59. За замовчуванням для зберігання часу застосовується формат "hh: mm: ss". Займає 3 байта.

• **DATETIME** : об'єднує час і дату, діапазон дат і часу - з 1 січня 1000 року по 31 грудня 9999 року (з "1000-01-01 00:00:00" до "9999-12-31 23:59:59") . Для зберігання за замовчуванням використовується формат "уууу-мм-дд hh: mm: ss". Займає 8 байт

• **TIMESTAMP** : також зберігає дату і час, але в іншому діапазоні: від "1970-01- 01 00:00:01" UTC до "2038-01-19 3:14:07" UTC. Займає 4 байти

• **YEAR** : зберігає рік у вигляді 4 цифр. Діапазон доступних значень від 1901 до 2155. Займає 1 байт.

Тип Date може приймати дати в різних форматах, однак безпосередньо для зберігання в самій бд дати наводяться до формату "уууу-мм-дд". Деякі з прийнятих форматів:

- уууу-мм-дд - 2018-05-25
- уууу-м-дд - 2018-5-25
- уу-м-дд - 18-05-25

В такому форматі двозначні числа від 00 до 69 сприймаються як дати в діапазоні 2000-2069. А числа від 70 до 99 як діапазон чисел 1970 - 1999.

- ууууmmdd - 20180525
- уууу.mm.dd - 2018.05.25

Для часу тип Time використовує 24-годинний формат. Він може приймати час в різних форматах:

- hh:mi- 3:21(збережене значення 03:21:00)
- hh:mi:ss - 19:21:34
- hhmiss - 192134

Приклади значень для типів DATETIME і TIMESTAMP:

- 2018-05-25 19:21:34
- 2018-05-25(збережене значення 2018-05-25 00:00:00)

Складові типи

• **ENUM** : зберігає одне значення зі списку допустимих значень. Займає 1-2

• **SET** : може зберігати кілька значень (до 64 значень) з деякого списку допустимих значень. Займає 1-8 байт.

бінарні типи

• **TINYBLOB** : зберігає бінарні дані у вигляді рядка довжиною до 255 байт.

• **BLOB** : зберігає бінарні дані у вигляді рядка довжиною до 65 КБ.

• **MEDIUMBLOB** : зберігає бінарні дані у вигляді рядка довжиною до 16 МБ

• **LARGEBLOB** : зберігає бінарні дані у вигляді рядка довжиною до 4 ГБ

Порядок виконання роботи

1. Завантажити MySQL Workbench


Дистрибутив MySQL Workbench доступний на офіційному сайті. Перед скачуванням потрібно вибрати одну з наступних платформ:

- Microsoft Windows (Доступні MSI Installer і ZIP архів)
- Ubuntu Linux
- Fedora
- Red Hat Enterprise Linux / Oracle Linux
- Mac OS X

🏠 <https://downloads.mysql.com/archives/>

MySQL Product Archives

This site provides archived versions of various MySQL products. We provide these as a courtesy to our users, who may need to duplicate an existing installation based on older versions.

 **Please note that these are old versions. New releases will have recent bug fixes and features!**
To download the latest releases of these products, please visit [MySQL Downloads](#).

- [MySQL Community Server](#)
- [MySQL Cluster](#)
- [MySQL Installer](#)
- [MySQL Workbench](#)
- [MySQL Router](#)
- [MySQL Utilities](#)
- [MySQL Shell](#)
- [MySQL NDB Operator](#)
- [MySQL Operator](#)
- [MySQL Proxy](#)
- [MySQL Connector/C](#)
- [MySQL Connector/C++](#)
- [MySQL Connector/J](#)
- [MySQL Connector/Node.js](#)
- [MySQL Connector/MXJ](#)
- [MySQL Connector/NET](#)
- [MySQL Connector/ODBC](#)
- [MySQL Connector/PHP](#)
- [MySQL Connector/Python](#)
- [MySQL Control Center](#)
- [MySQL Administrator](#)
- [MySQL Migration Toolkit](#)
- [MySQL GUI Tools](#)
- [MySQL Query Browser](#)
- [MySQL for Visual Studio](#)
- [MySQL for Excel](#)
- [MySQL Notifier](#)

← ↻ 🏠 🔒 <https://downloads.mysql.com/archives/workbench/>

MySQL Product Archives

← MySQL Workbench (Archived Versions)

⚠️ Please note that these are old versions. New releases will have recent bug fixes and features!
To download the latest release of MySQL Workbench, please visit [MySQL Downloads](#).

Product Version:

Operating System:

- Select Operating System...
- Microsoft Windows
- Ubuntu Linux
- Red Hat Enterprise Linux / Oracle Linux
- Fedora
- macOS
- Source Code

Windows (x86, 64-bit) (mysql-workbench-community-8.0.33-1)	Mar 27, 2023	45.5M	MDS: 28a2b1831a958
--	--------------	-------	--------------------

! We suggest that you use the [MD5 checksums](#) and [GnuPG signatures](#) to verify the integrity of the packages you download.

MySQL open source software is provided under the [GPL License](#).

ORACLE © 2023 Oracle

[Privacy / Do Not Sell My Info](#) | [Terms of Use](#) | [Trademark Policy](#) | [Напаштування кукія](#)

← ↻ 🏠 🔒 <https://downloads.mysql.com/archives/shell/>

MySQL Product Archives

← MySQL Shell (Archived Versions)

⚠️ Please note that these are old versions. New releases will have recent bug fixes and features!
To download the latest release of MySQL Shell, please visit [MySQL Downloads](#).

Product Version:

Operating System:

- Select Operating System...
- Microsoft Windows
- Ubuntu Linux
- Debian Linux
- SUSE Linux Enterprise Server
- Red Hat Enterprise Linux / Oracle Linux
- Fedora
- Linux - Generic
- macOS
- Source Code

Windows (x86, 64-bit) (mysql-shell-8.0.33-windows)	Mar 31, 2023	68.8M	Download
Windows (x86, 64-bit) (mysql-shell-8.0.33-windows)	Mar 30, 2023	76.8M	Download

! We suggest that you use the [MD5 checksums](#) and [GnuPG signatures](#) to verify the integrity of the packages you download.

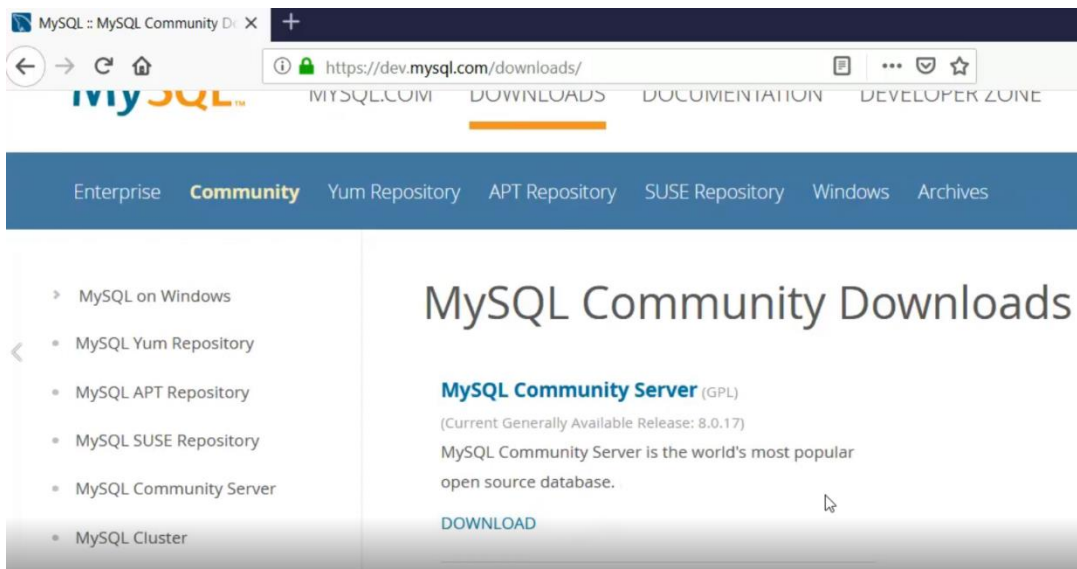
MySQL open source software is provided under the [GPL License](#).

ORACLE © 2023 Oracle

[Privacy / Do Not Sell My Info](#) | [Terms of Use](#) | [Trademark Policy](#) | [Напаштування кукія](#)

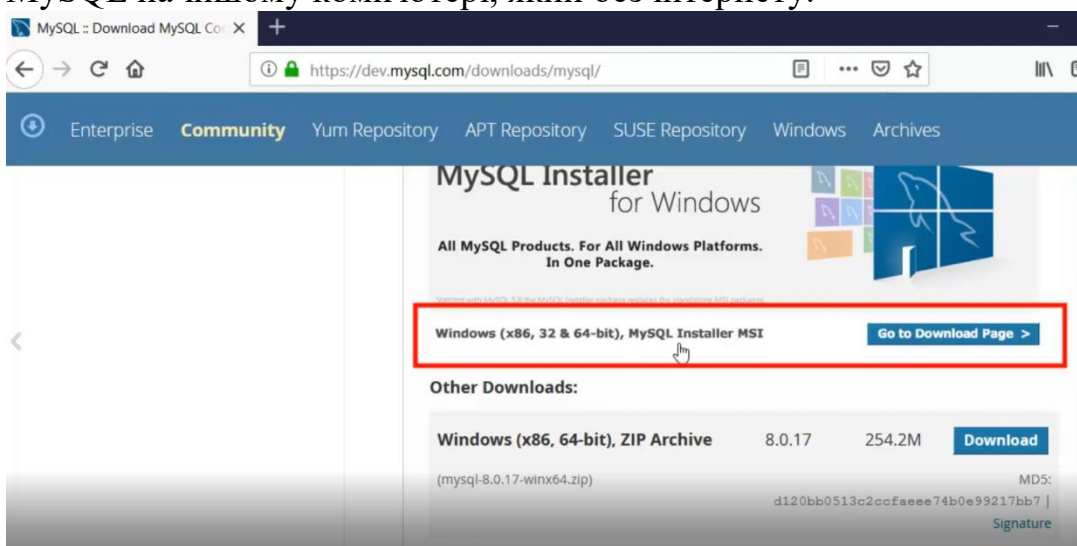
Після вибору платформи вам пропонують зареєструватися або авторизуватися в Oracle. Якщо не хочете, внизу є посилання "No thanks, just start my download", можна натиснути на нього.

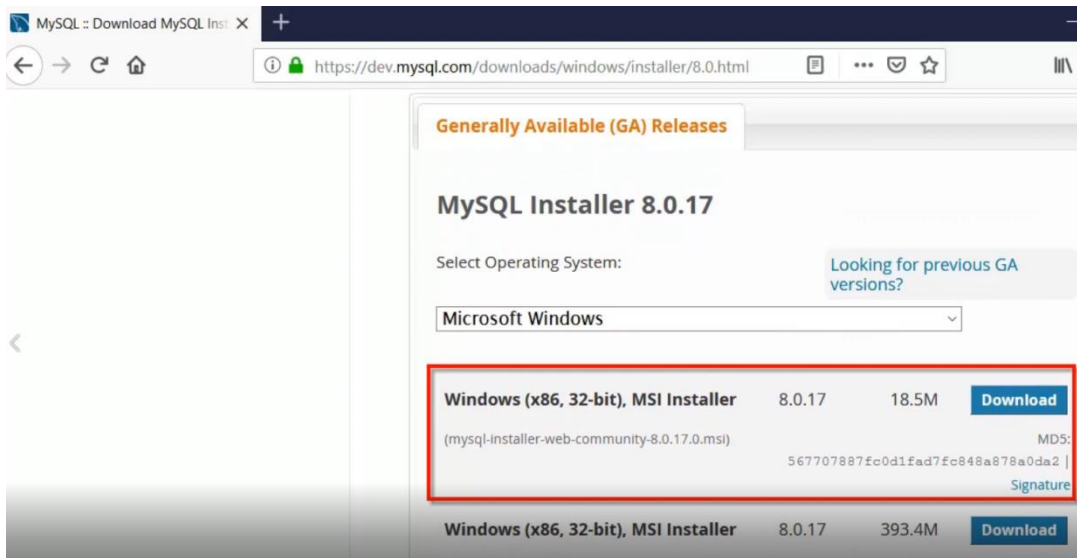
Щоб перейти до установки MySQL 8, спочатку необхідно завантажити дистрибутив цієї програми. Завантажити MySQL 8 в редакції Community можна, як було вже зазначено, абсолютно вільно з офіційного сайту, ось посилання на сторінку завантаження - <https://dev.mysql.com/downloads/installer>



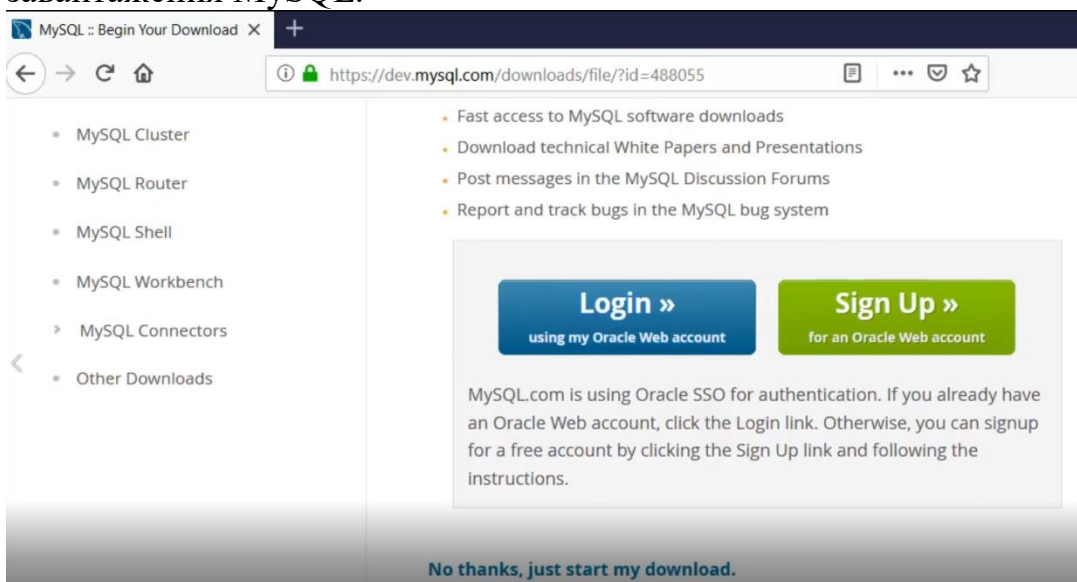
Після переходу на сторінку Ви можете вибрати спосіб завантаження дистрибутива, нам пропонують два способи:

- Завантажити Web-інсталятор - невелика за розміром програма, яка завантажує всі необхідні компоненти для установки MySQL. Можна використовувати для установки MySQL на комп'ютерах, де є інтернет;
- Завантажити повний установник - дистрибутив програми MySQL, який включає всі необхідні компоненти. В даному випадку Ви можете використовувати цей дистрибутив для установки MySQL на комп'ютері як з доступом, так і без доступу до інтернету. Наприклад, Ви можете завантажити цей файл на комп'ютері, де є інтернет, а використовувати його для установки MySQL на іншому комп'ютері, який без інтернету.





Потім нам пропонують авторизуватися, використовуючи облікові дані Oracle, якщо вони є, або зареєструватися, тим самим створити обліковий запис Oracle. Однак якщо Ви цього не хочете, то в нижній частині є посилання «No thanks, just start my download», натиснувши на яку відразу почнетесь завантаження MySQL.



Натискаємо на це посилання і чекаємо закінчення завантаження.

В результаті у Вас повинен завантажитися файл `mysql-installer-community8.0.18.0.msi` розміром приблизно 415 мегабайт.

2. Другим етапом відбувається запуск установки і вибір типу установки MySQL.

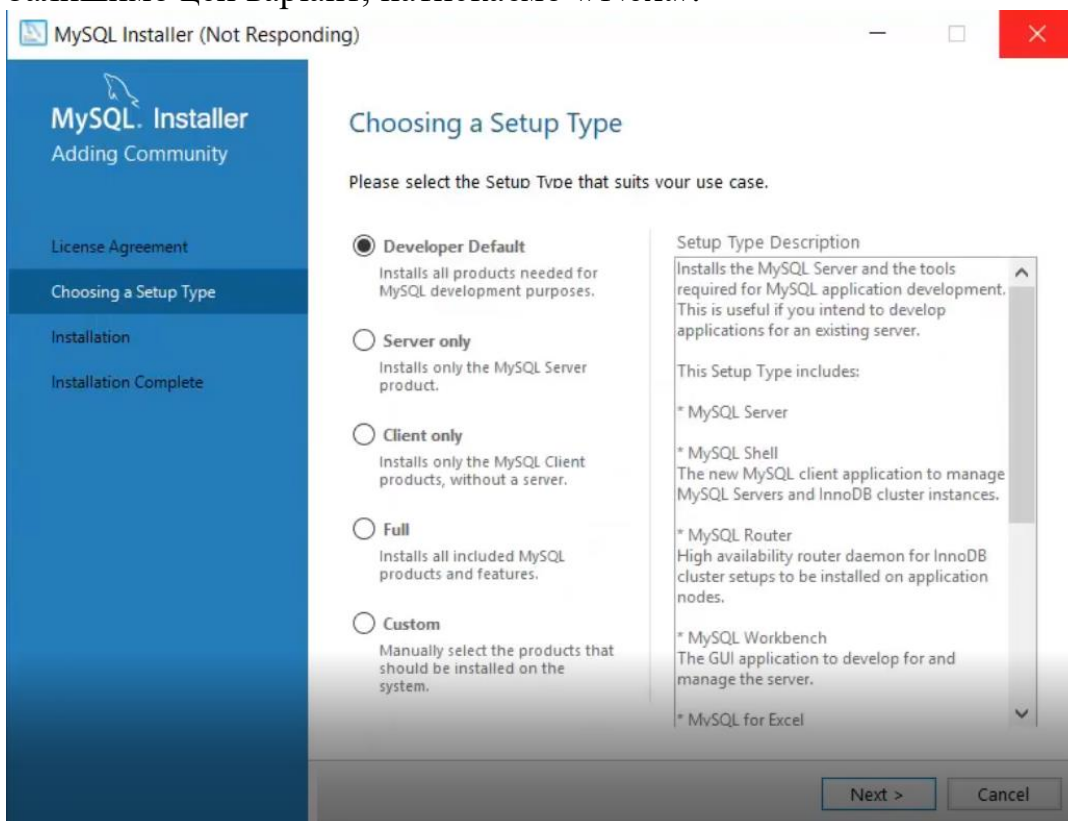
Далі запускаємо скачав файл, в результаті запуситься програма установки MySQL. Спочатку вибираємо тип установки.

Нам пропонують кілька варіантів:

- **Developer** - це варіант за замовчуванням, він передбачає установку всього того, що потрібно розробнику, це і MySQL Server, і MySQL Workbench і інші інструменти для роботи з MySQL;
- **Server Only** - цей варіант передбачає установку тільки сервера MySQL;

- Client Only - цей варіант передбачає установку тільки клієнтської частини для роботи з MySQL Server, тобто серверна частина встановлюватися не буде;
- Full - установка для всіх включених в дистрибутив компонентів;
- Custom - вибіркова установка, в даному випадку Ви можете встановити лише те, що потрібно Вам.

Варіант за замовчуванням, тобто Developer, підійде для більшості випадків, особливо якщо Ви встановлюєте MySQL на домашньому комп'ютері для якихось своїх цілей (вивчення MySQL, вивчення SQL і так далі). Залишимо цей варіант, натискаємо « Next ».

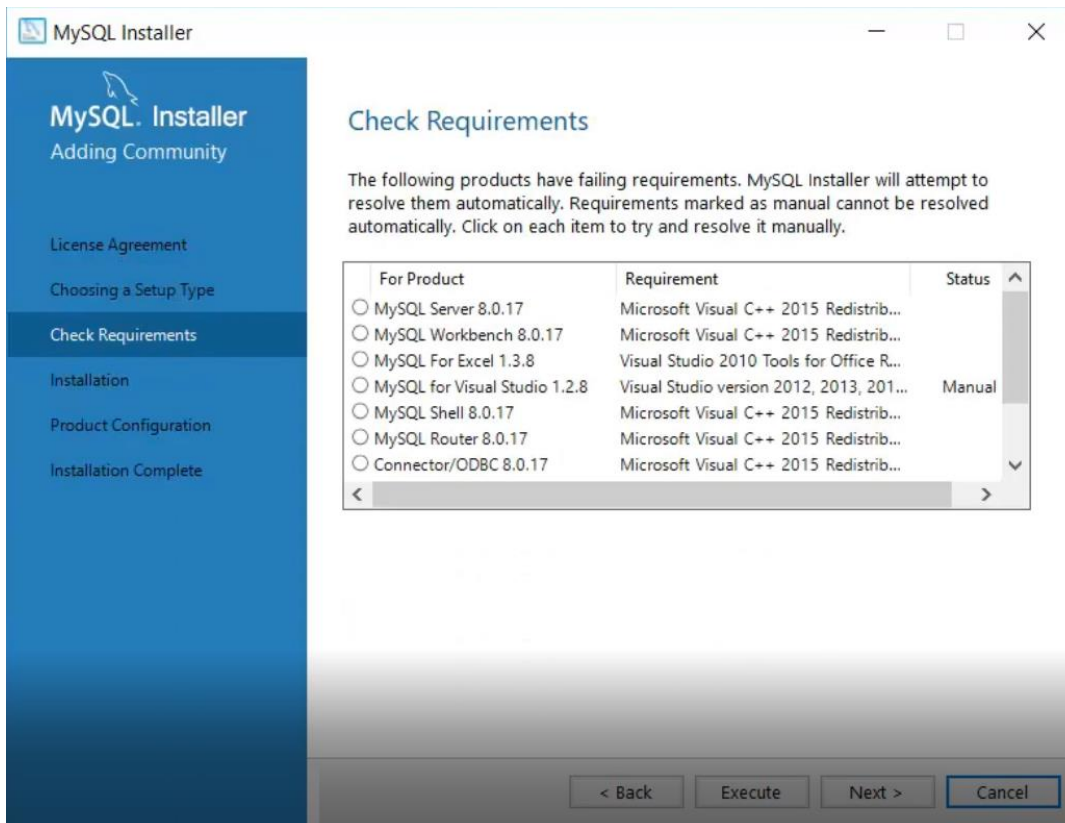


3. Третім етапом відбувається перевірка і установка додаткових компонентів. Потім програма установки перевірить систему на наявність компонентів, які потрібні для роботи деяких розширень MySQL. У моєму випадку програма установки видала два попередження для розширень MySQL For Excel і MySQL For Visual Studio. Якщо Ви плануєте користуватися цими компонентами, то Вам потрібно усунути ці зауваження. Для цього в свою чергу Вам необхідно встановити ці компоненти (тобто виділити їх і натиснути «Execute», деякі можуть встановитися і автоматично).

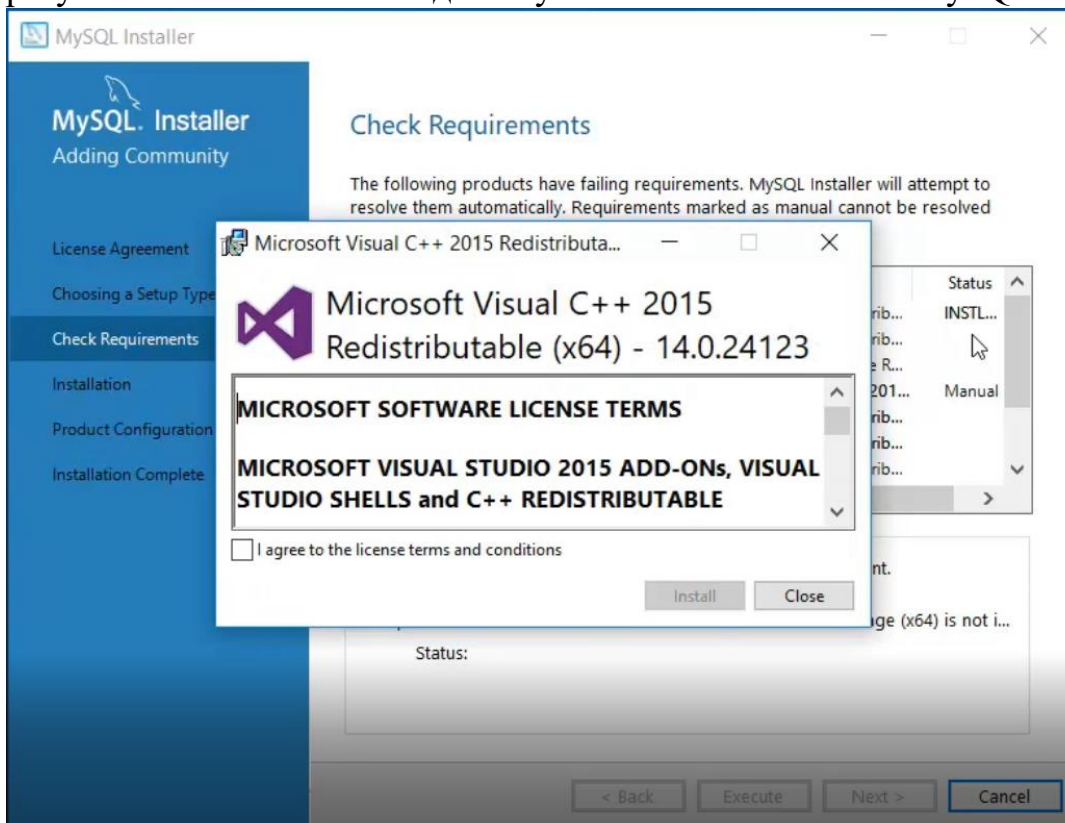
Однак якщо Ви не будете використовувати розширення MySQL For Excel і MySQL For Visual Studio, то можете відразу натиснути «Next» .

Замітка ! Установка Visual Studio 2019 Community на Windows 10.

Програма установки видасть попередження, пов'язане з відсутністю деяких компонентів, натискаємо «Yes».

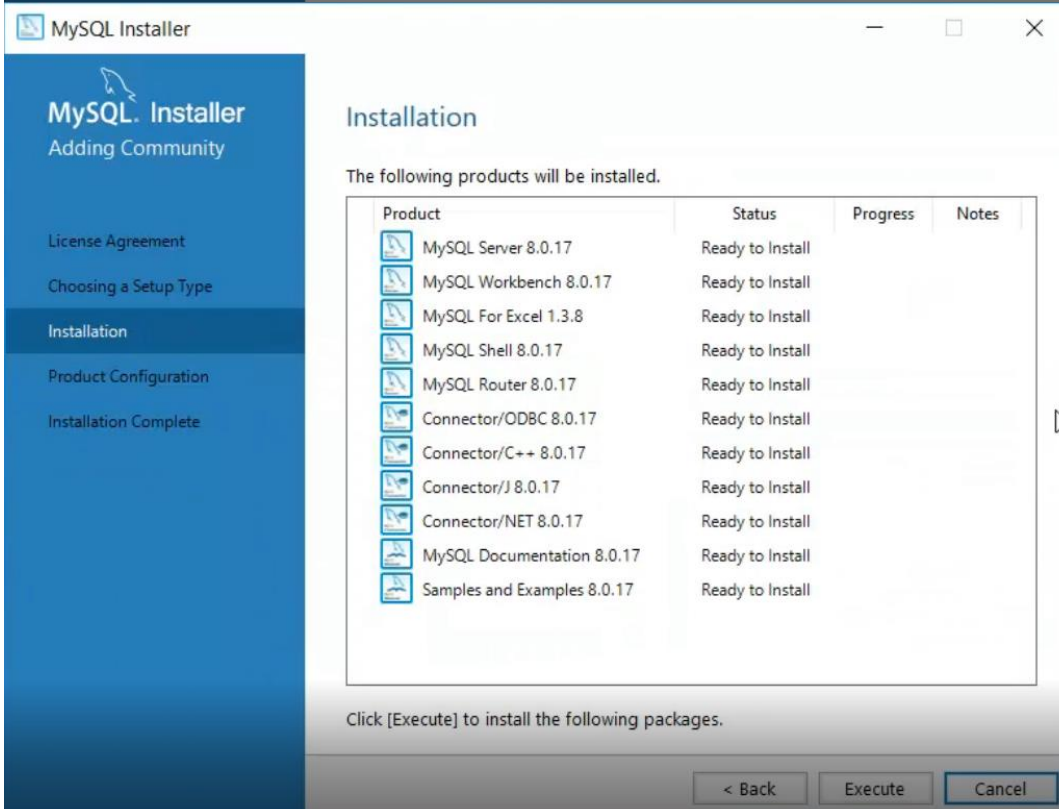
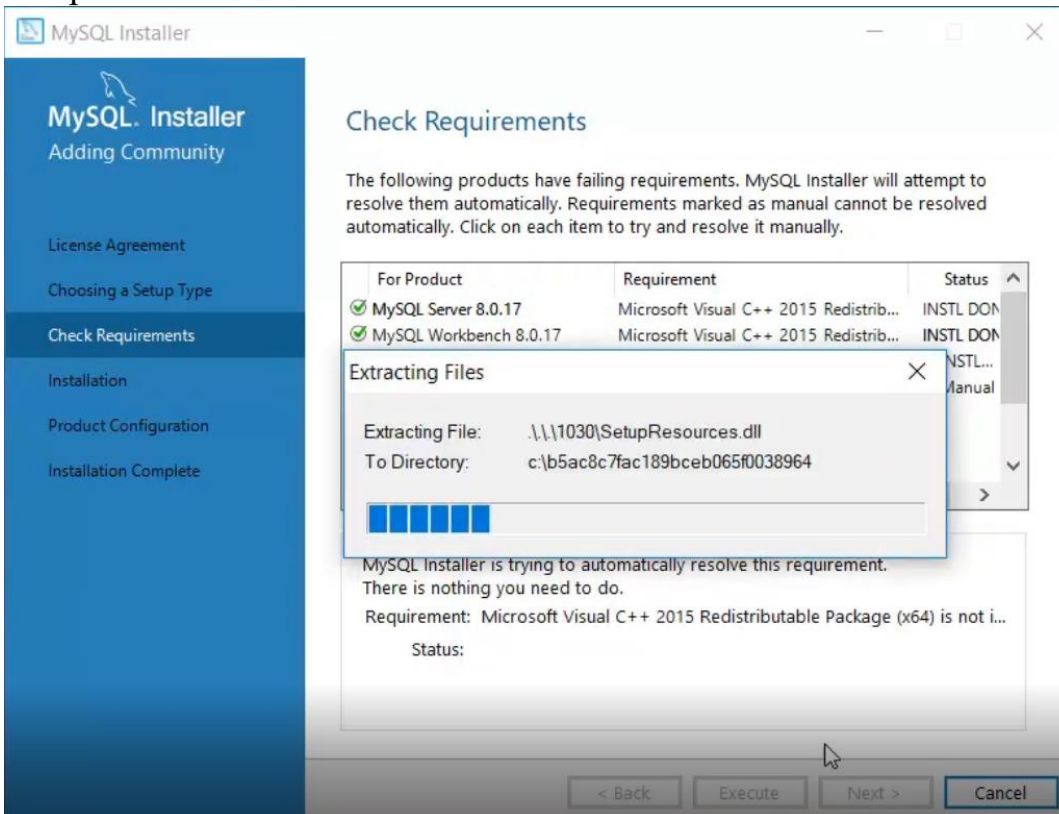


4. Четвертим етапом відбувається установка компонентів MySQL. Все готово для установки MySQL, на даному етапі програма установки видасть всі компоненти MySQL, які готові до установки, натискаємо «Execute». В результаті почнеться послідовна установка компонентів MySQL.



5. Завершення установки компонентів MySQL. Коли навпроти кожного пункту з'явиться зелена галочка, установка буде завершена.

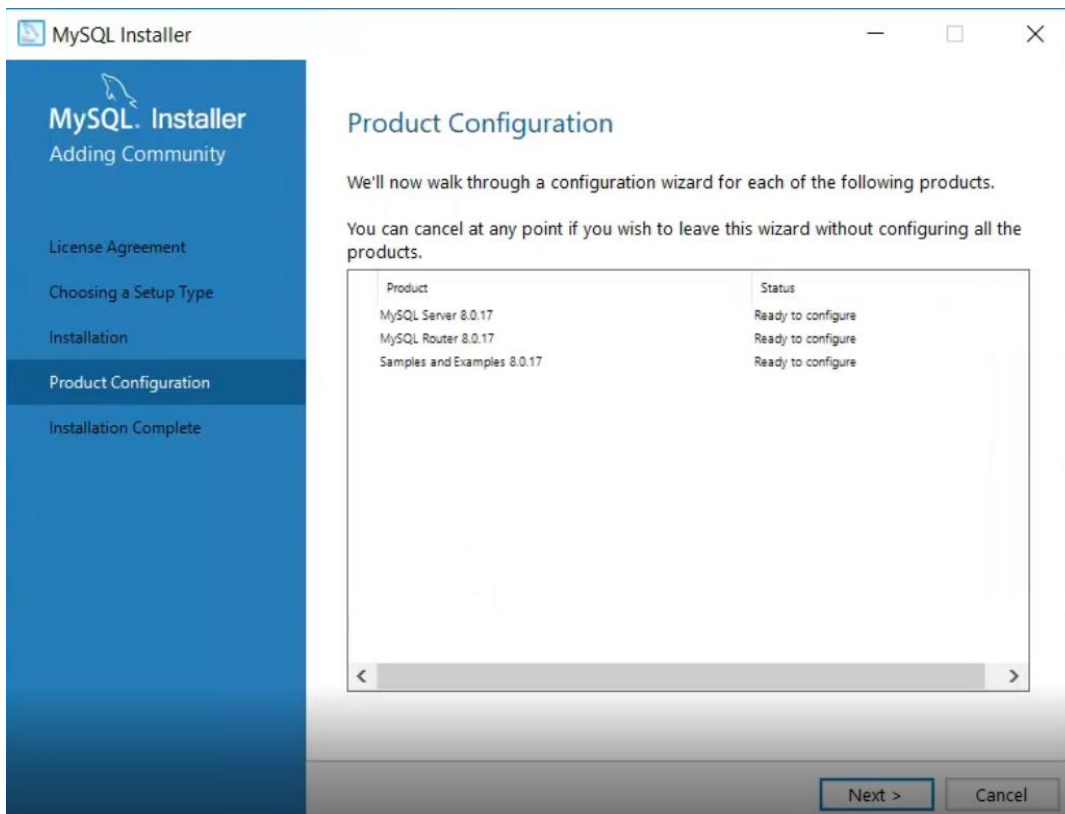
завершена. Натискаємо « Next » .



6. Налаштування компонентів MySQL

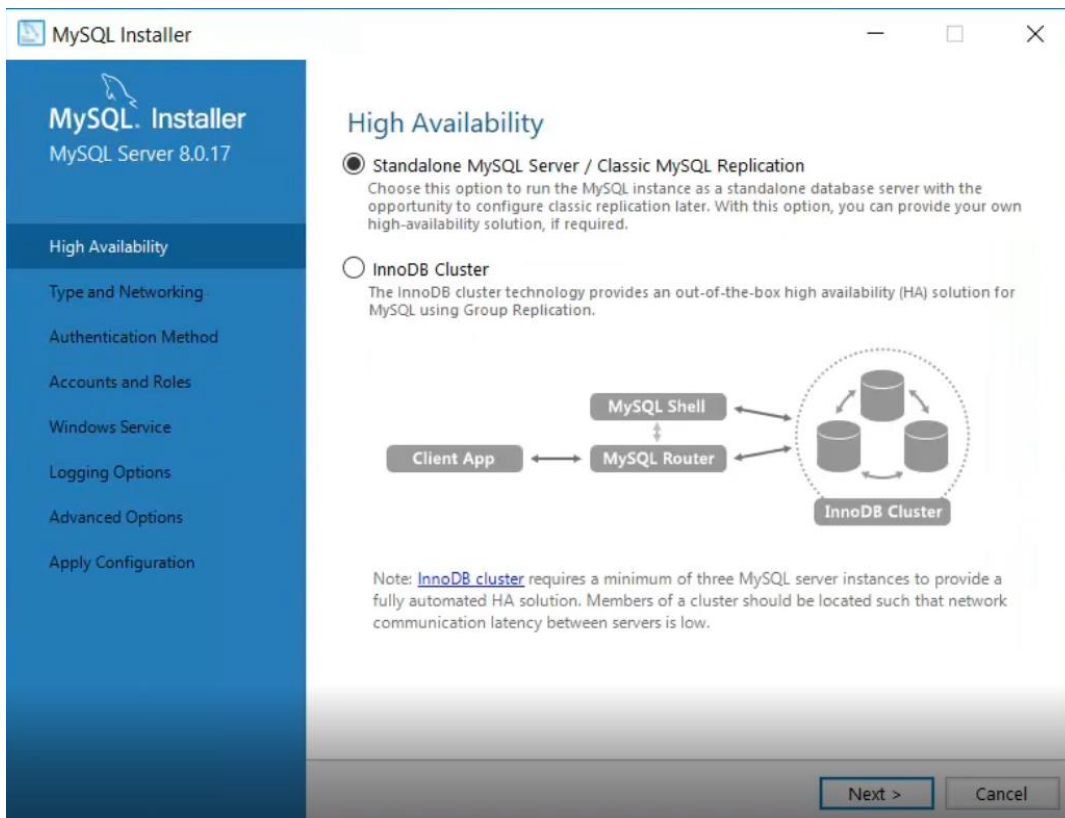
Всі компоненти встановлені, але не всі налаштовані, тому нам необхідно їх налаштувати. Програма установки покаже, які саме компоненти потрібно налаштувати. У нашому випадку це MySQL Server, MySQL Router і Samples and Examples.

Натискаємо «Next».



7. Налаштування MySQL Server (параметр High Availability)

Спочатку нам обов'язково необхідно налаштувати MySQL Server. Насамперед налаштовуємо параметр « High першими Availability» (Висока доступність) , який відповідає за те, як буде встановлено MySQL сервер. У нашому випадку, як і в більшості інших, потрібно стандартний « Standalone MySQL Server» (Автономний сервер) - це класичний варіант установки MySQL Server. Тому вибираємо перший пункт і натискаємо «Next».

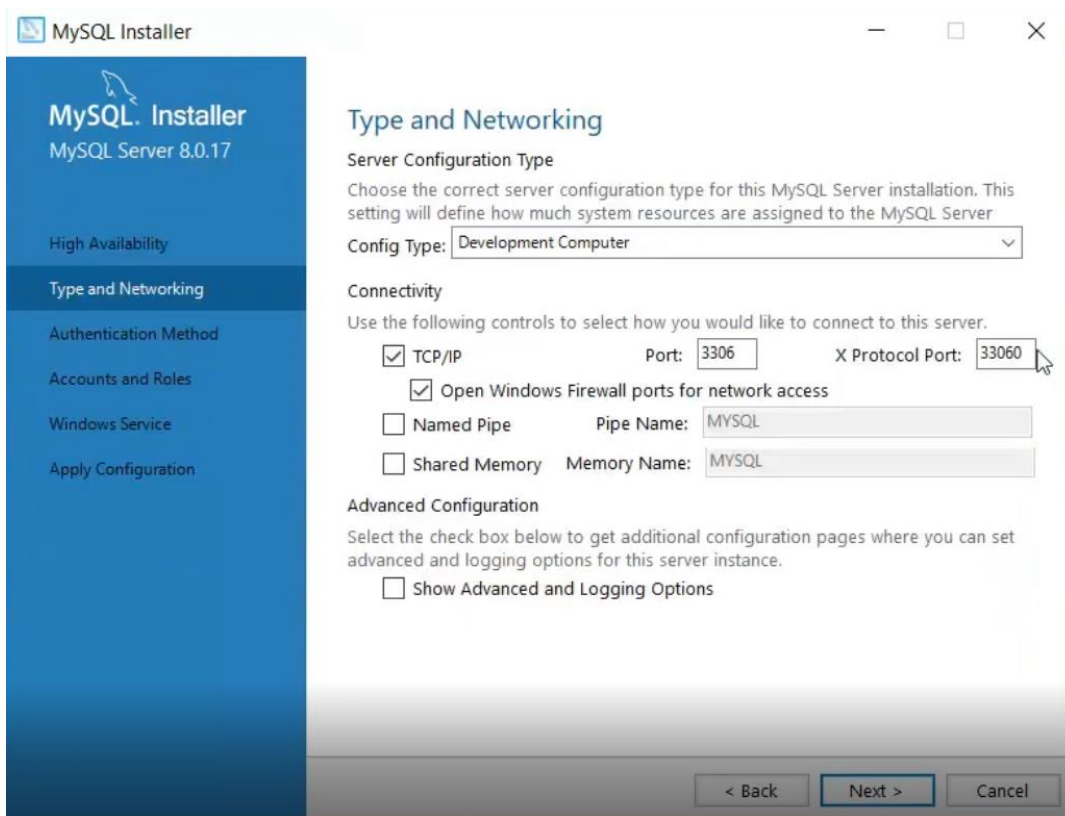


8. Налаштування MySQL Server (Type and Networking)

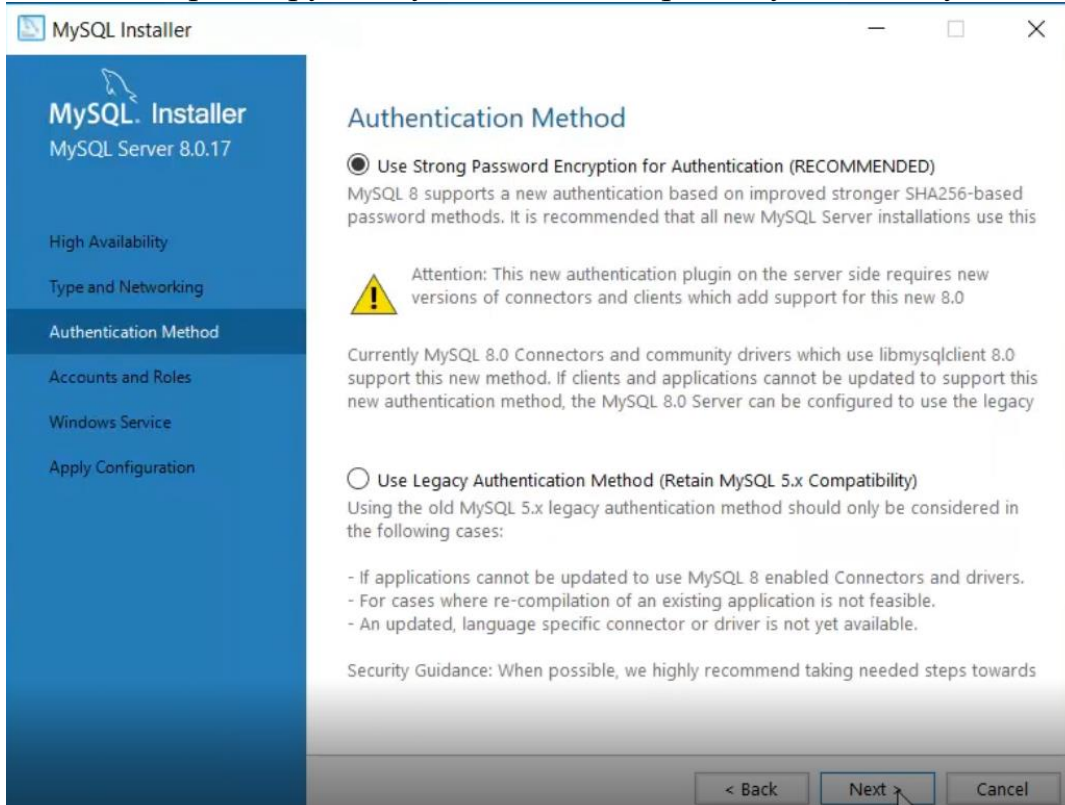
Далі налаштовуємо тип сервера і мережу. У нашому випадку можна залишити все за умовчанням:

- Тип - Development Computer;
- Протокол - TCP / IP;
- Порт - 3306;
- Галочку «Open Windows Firewall ports for network access» необхідно поставити.

У разі необхідності Ви можете більш тонко налаштувати MySQL Server, використавши для цього розширені параметри. Щоб це зробити, поставте галочку « Show Advanced and Logging Options» . Для продовження натискаємо «Next».



9. Налаштування методу аутентифікації в MySQL Server 8 версія MySQL підтримує новий метод аутентифікації, який і рекомендовано використовувати, залишаємо як є і натискаємо «Next». У разі необхідності Ви можете вибрати другий пункт, який використовувався в MySQL 5.



10. Налаштування користувачів MySQL
Після цього нам потрібно придумати пароль для користувача root (це головний

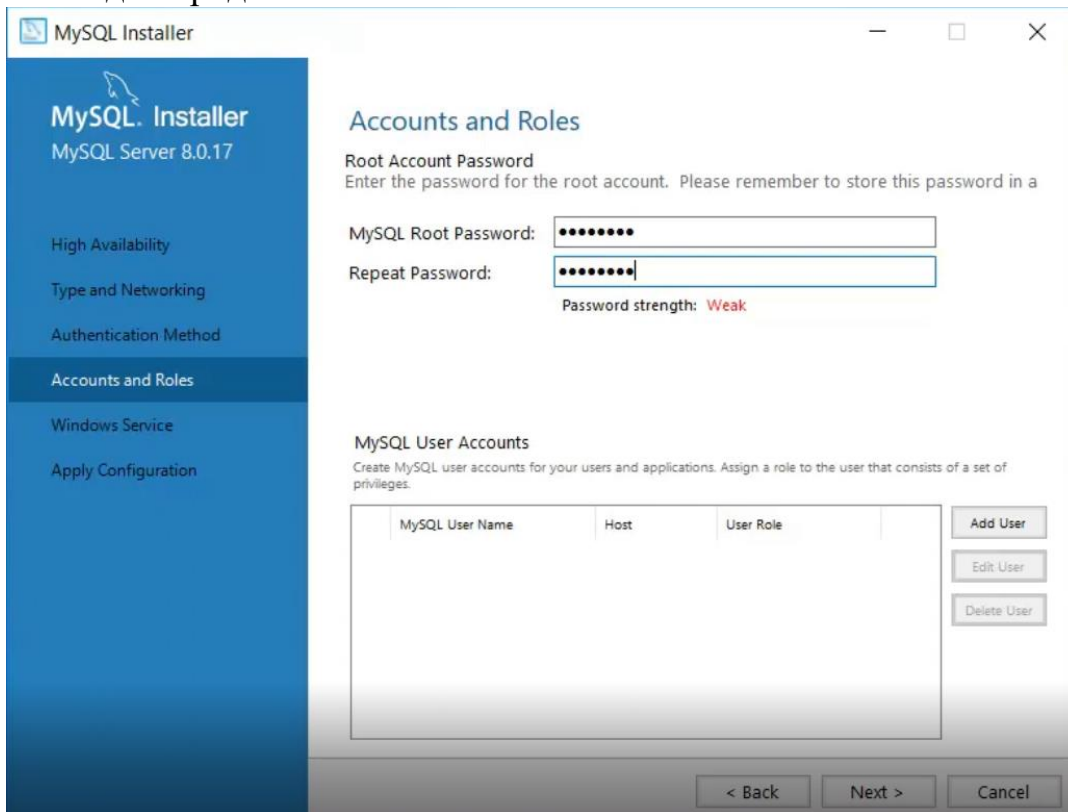
адміністратор MySQL).

Крім цього, ми можемо додати додаткових користувачів, щоб це зробити, необхідно натиснути на кнопку « Add User » .

І ввести необхідні дані:

- User Name - вводимо ім'я користувача;
- Host - залишаємо «All Hosts» ;
- Role - вибираємо роль користувача;
- Password і Confirm Password - придумуємо і вводимо пароль, який буде у цього користувача.

Натискаємо «ОК». Після цього користувач буде створений. Натискаємо « Next » для продовження.



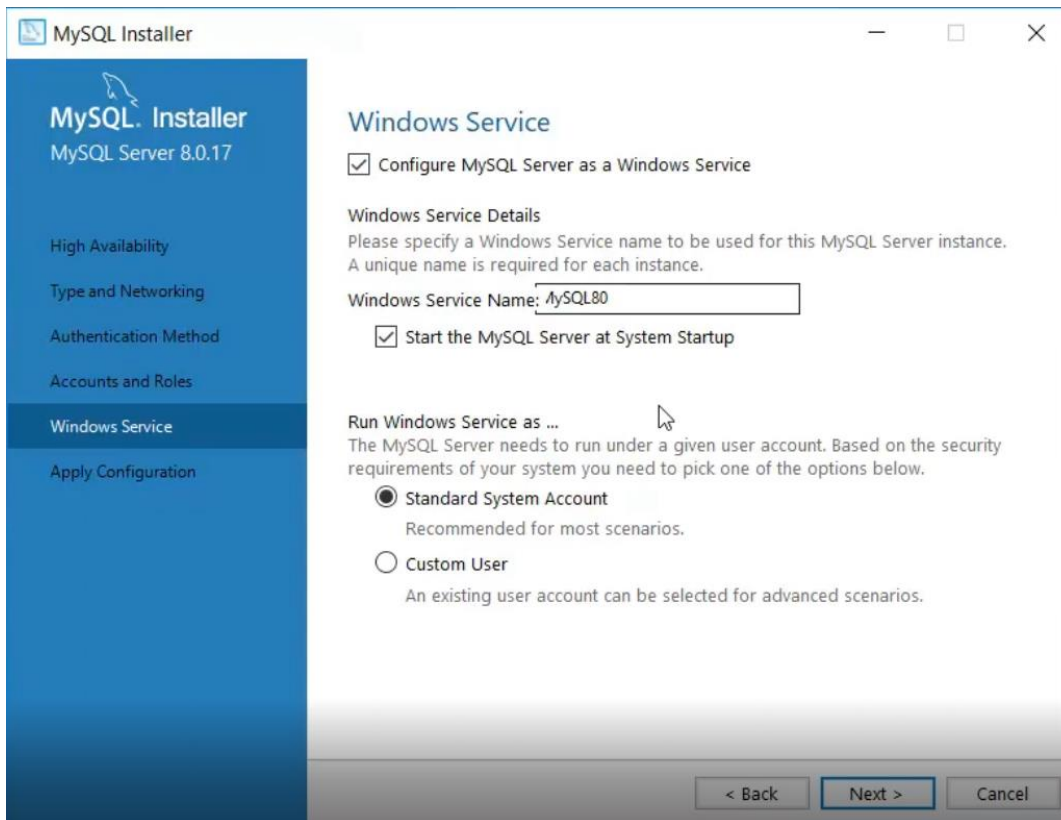
11. Налаштування служби MySQL в Windows

Тепер нам необхідно налаштувати службу MySQL, яка буде працювати в Windows.

Ми можемо поставити:

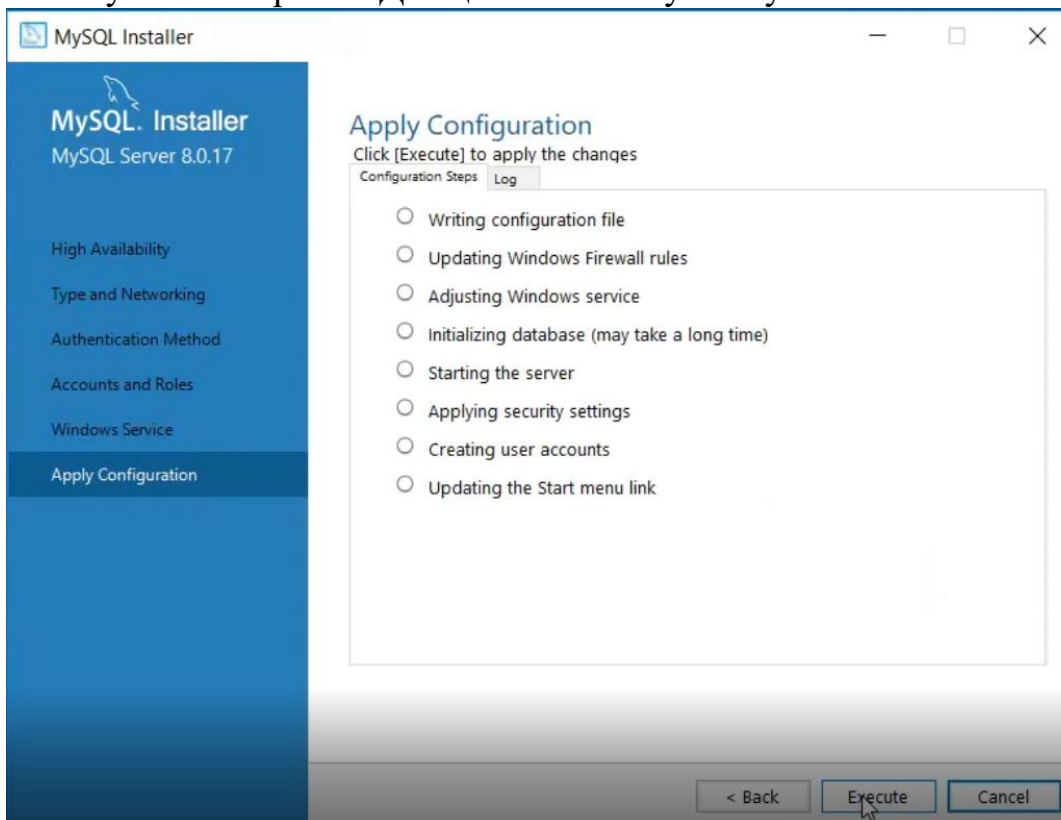
- Windows Service Name - Ім'я служби;
- «Start the MySQL Server at System Startup» - якщо ця галочка стоїть, то служба MySQL запускатиметься разом із запуском Windows;
- Run Windows Service - від імені якої облікового запису буде працювати служба MySQL в Windows. Standard System Account - це системна обліковий запис.

Можна все залишити за замовчуванням. Натискаємо « Next » .



12. Застосування параметрів MySQL Server

Всі параметри MySQL Server налаштовані, однак нам ще потрібно їх застосувати і зберегти. Для цього на наступному вікні натискаємо «Execute».

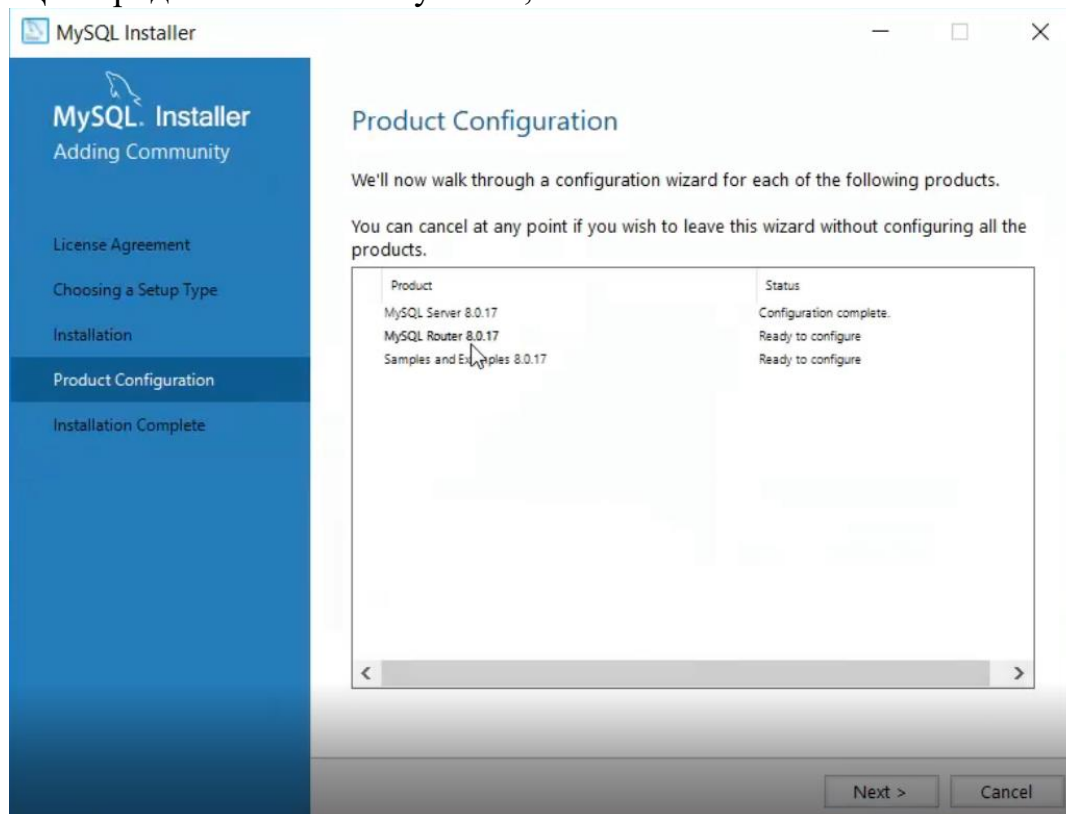


13. Завершення налаштування MySQL Server

Коли всі галочки будуть проставлені і відобразиться повідомлення «Successful», процес буде завершений, натискаємо «Finish». Статус MySQL Server буде

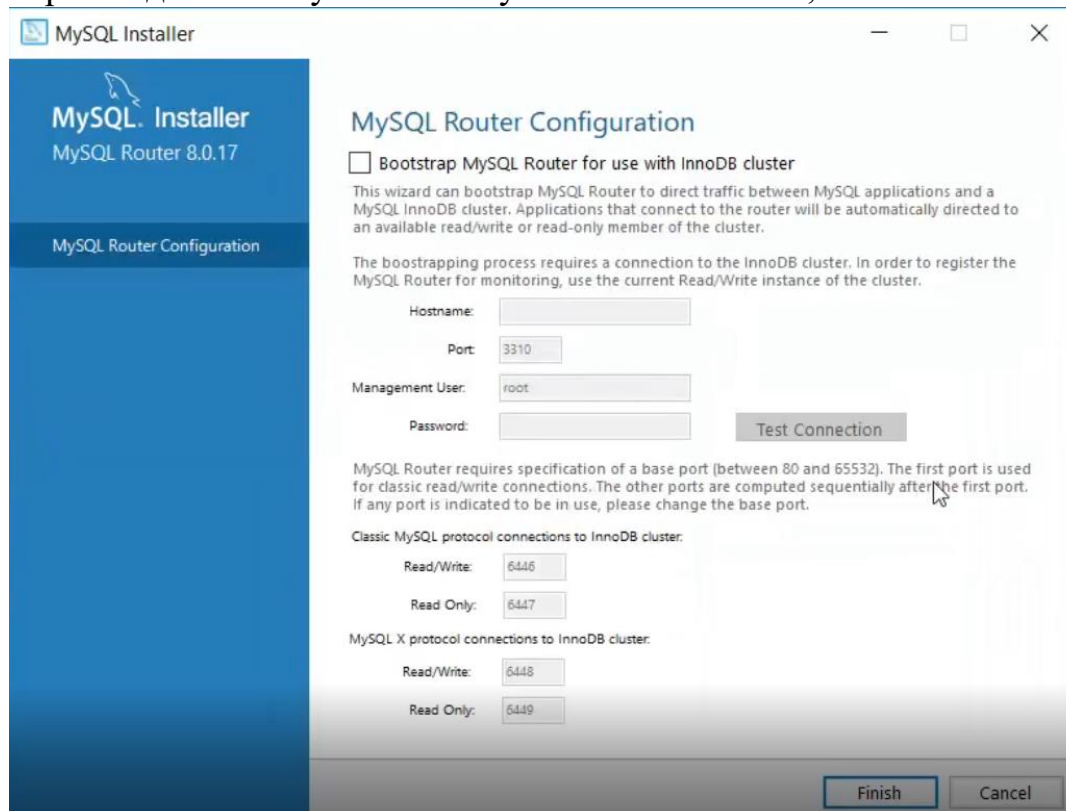
змінений на «Configuration complete» .

Щоб продовжити налаштування, натискаємо « Next » .



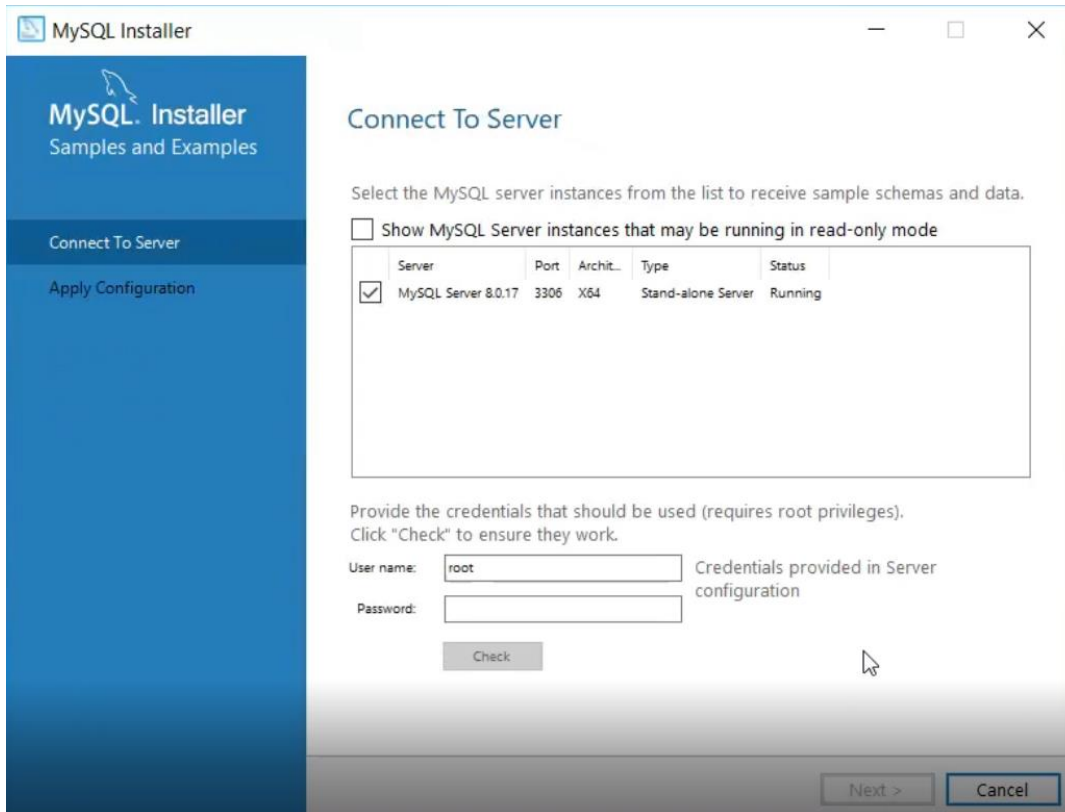
14. Налаштування MySQL Router

В даному випадку нам пропонують налаштувати маршрутизатор MySQL, однак це робити необов'язково, можемо відразу натиснути « Finish » . Для того щоб перейти до налаштування наступного компонента, натискаємо « Next » .

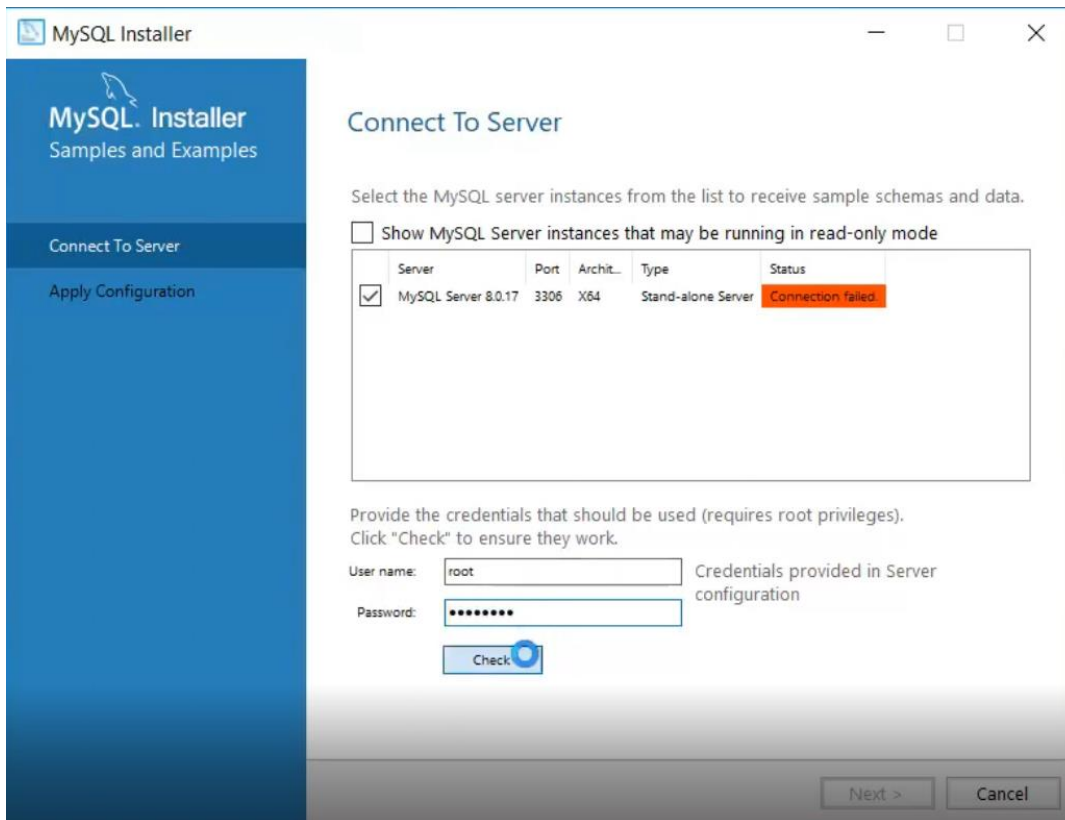


15. Установка тестових даних на MySQL

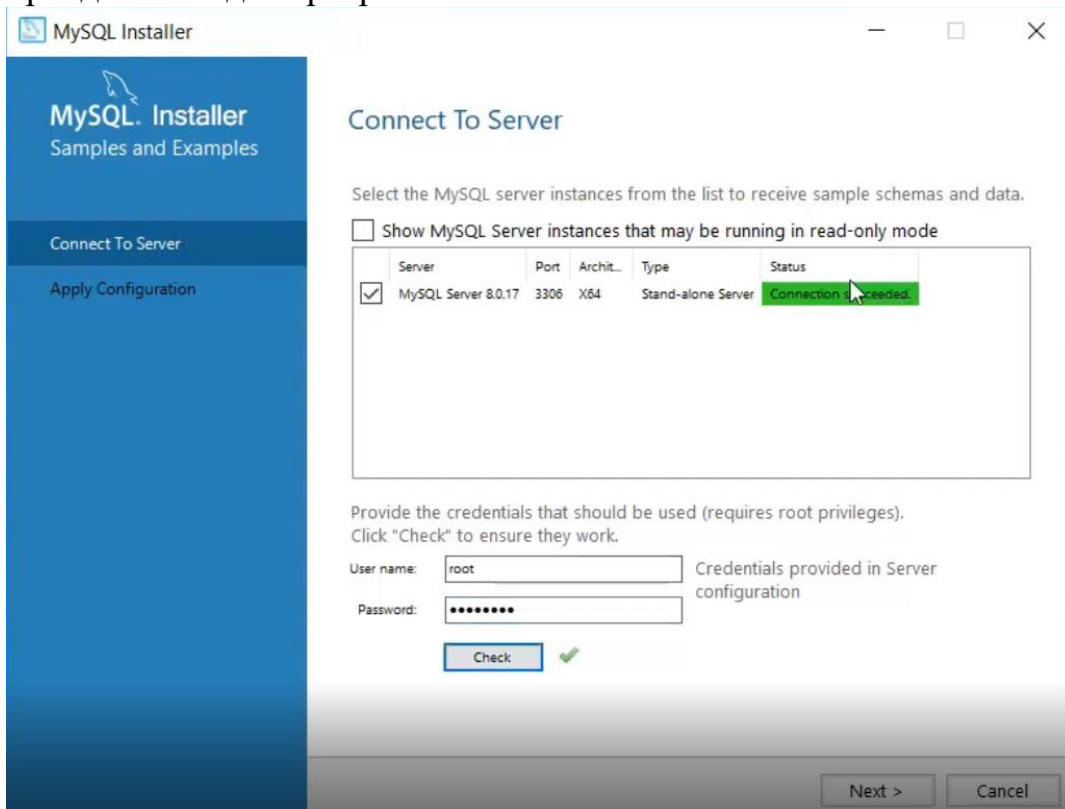
Далі ми можемо встановити тестові дані на MySQL Server, наприклад, для навчання. Щоб це зробити, вводимо пароль користувача Root і натискаємо «Check» для перевірки, якщо підключення встановлено, значить все добре і можна продовжувати. Натискаємо «Next». Щоб застосувати всі параметри і почати витяг тестових даних на MySQL Server, натискаємо «Execute». Процес буде завершено, коли галочки будуть проставлені і відобразиться повідомлення «Successful». Натискаємо «Finish».



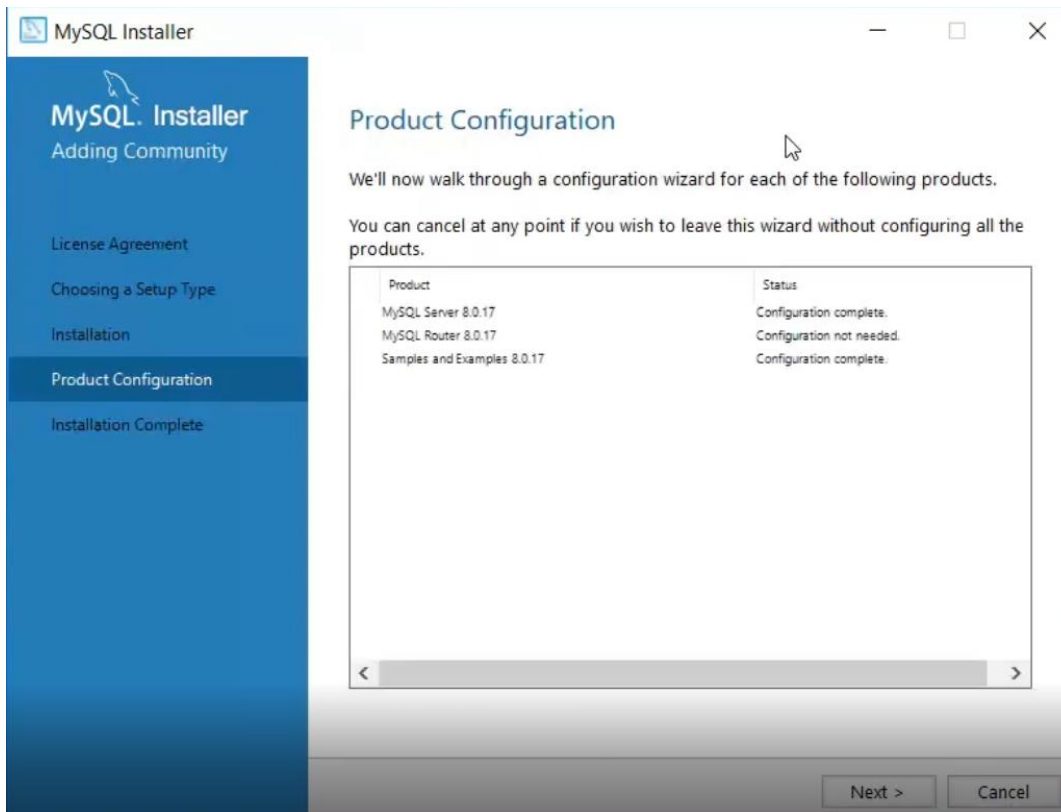
Коли пароль буде введено неправильно, з'єднатись із сервером не вдасться.



Нижче наведено вікно з правильно введеним паролем та успішним приєднанням до сервера.

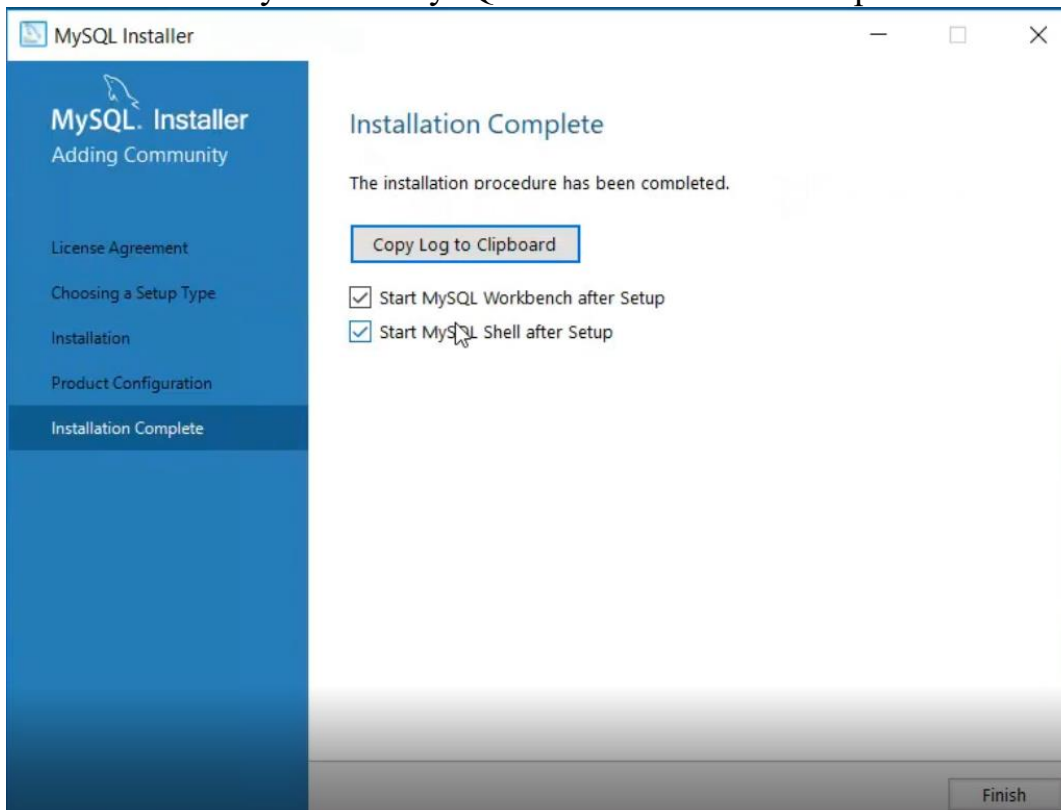


16. Завершення налаштування компонентів MySQL
Всі компоненти налаштовані. Натискаємо « Next » .



17. Завершення всього процесу установки і настройки MySQL MySQL Server і всі необхідні для роботи з ним компоненти встановлені і налаштовані: це і MySQL Workbench, і різні коннектори, і документація, і навіть тестова база даних.

Щоб відразу ж запустити MySQL Workbench після завершення установки, поставте галочку « Start MySQL Workbench after Setup» . Натискаємо « Finish» .

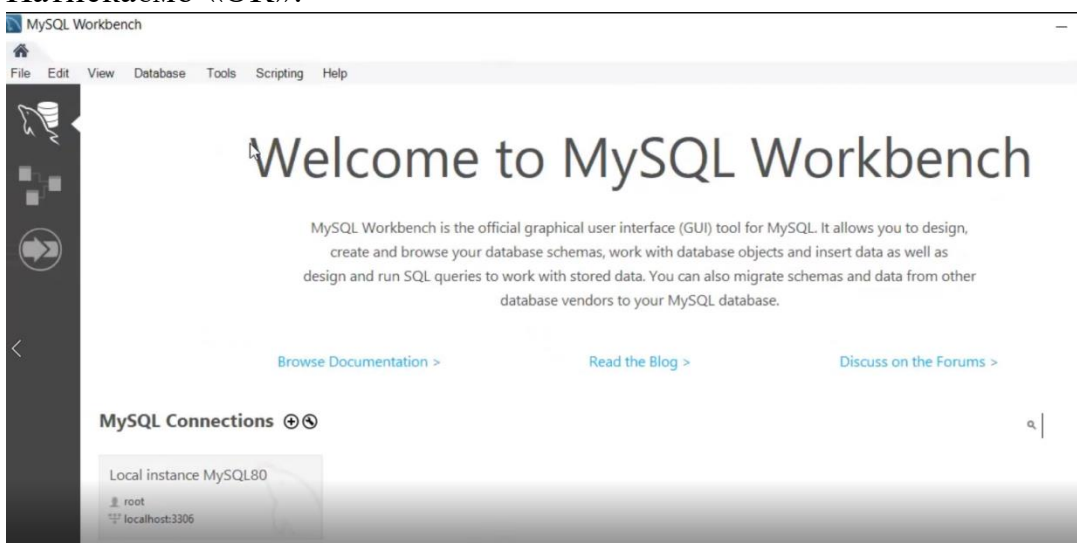


18. Запуск MySQL Workbench

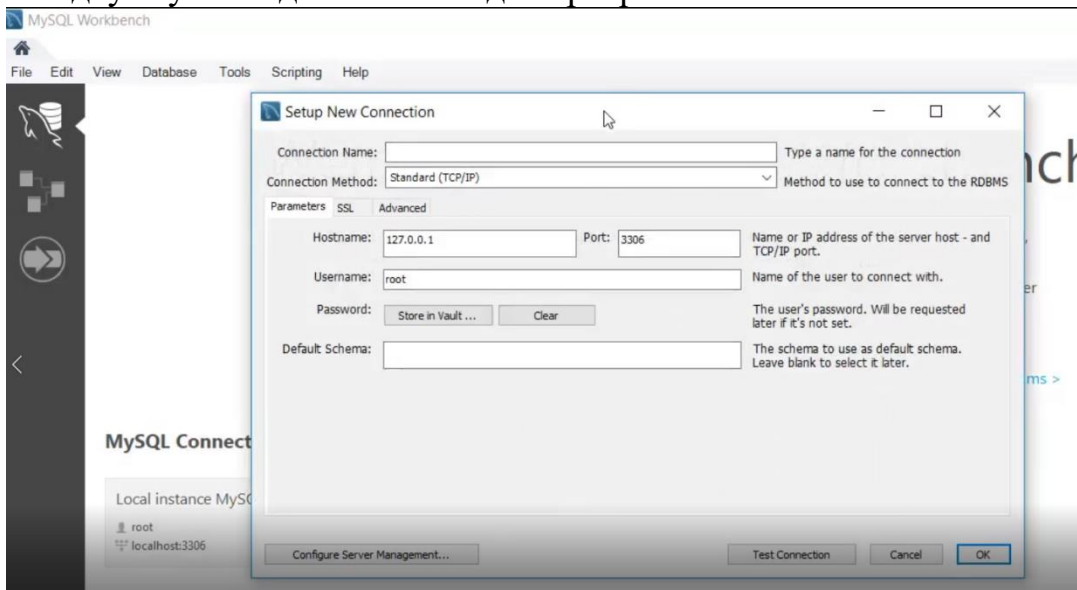
Після установки MySQL в меню Windows 10 з'являться всі необхідні ярлики, які Ви можете використовувати. Наприклад, для запуску середовища «MySQL Workbench 8» можна використовувати відповідний ярлик.

Підключення до MySQL використовуючи MySQL Workbench

Щоб підключитися до сервера MySQL і подивитися на об'єкти бази даних, на таблиці, уявлення і так далі, запустіть MySQL Workbench, наприклад, як зазначено вище. В результаті відкриється MySQL Workbench, де на стартовій сторінці у нас вже буде одне підключення до локального сервера, яке буде здійснюватися від імені користувача root. Натискаємо на нього. Потім вводимо пароль, щоб зберегти його і не вводити кожен раз при підключенні ми можемо поставити галочку « Save password in vault». Натискаємо «ОК».



У підсумку ми підключимося до сервера.



Для тих одиничних випадків, коли встановити СУБД неможливо, для тестування SQL-інструкцій можна використати SQL онлайн редактори та

компілятори коду, на кшталт <https://onecompiler.com/mysql>. Для побудови діаграм сутність зв'язок у такому випадку можна використати один із існуючих графічних за стосунків, наприклад онлайн-варіант – це <https://www.mindonmap.com/uk/blog/lucidchart-er-diagram-tutorial>. Але це буде лише ER-діаграма окремо, без прив'язки до БД, тому бажано все ж таки опанувати усі функції СУБД.

ЗАВДАННЯ ДЛЯ САМОСТІЙНОЇ РОБОТИ

Завдання №1. Провести установку MySQL Workbench на ПК.

Завдання №2. Створити базу даних за допомогою MySQL Workbench двома способами (через графічну оболонку та консоль).

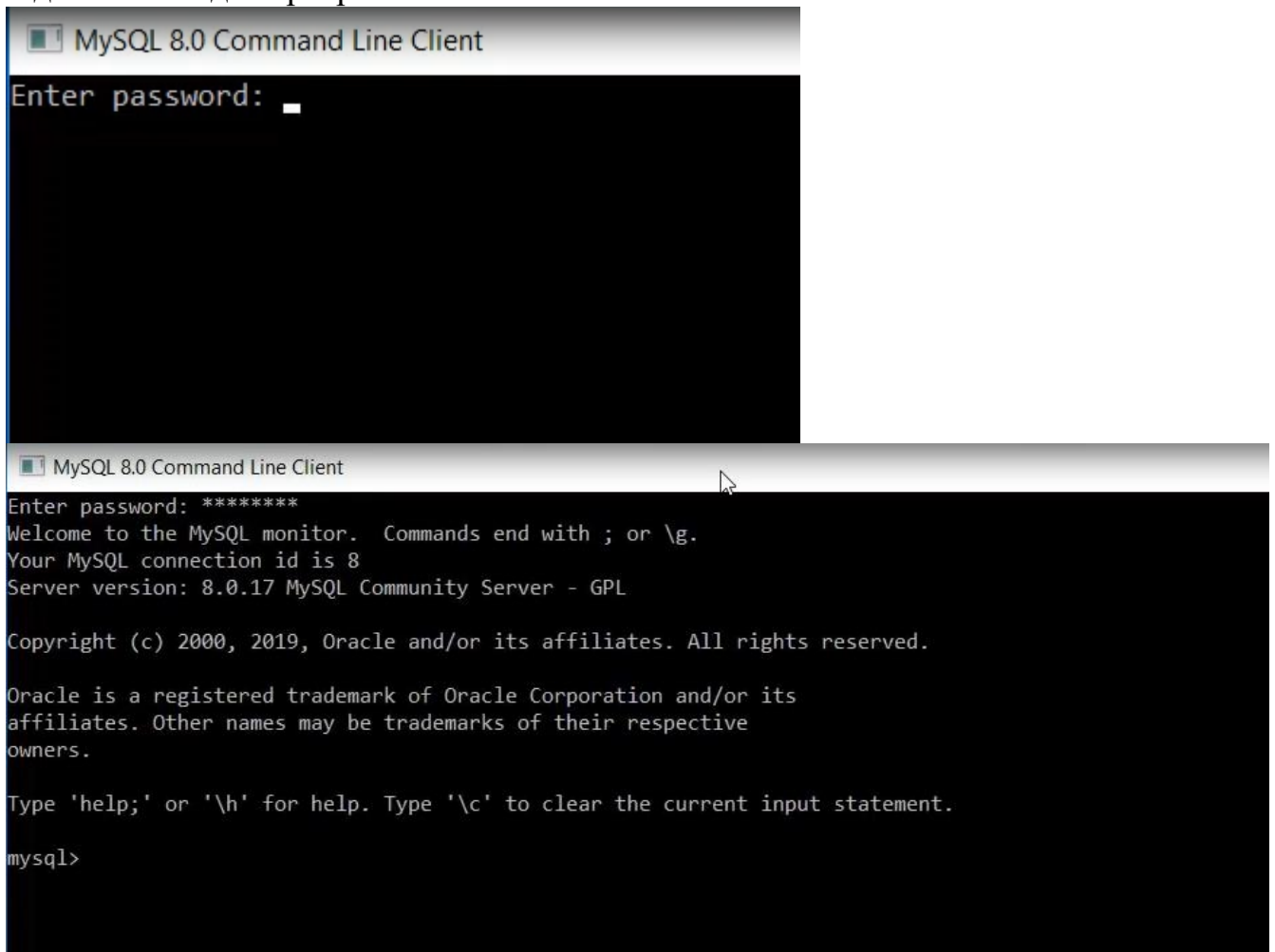
Лабораторна робота №2

Тема: Основні операції з базою даних. Робота в командному рядку та за допомогою графічного інтерфейсу

Мета: Навчитися проводити збір інформації з різних джерел до внесення у таблиці БД. Навчитись переглядати та створювати БД та таблиці у командному та графічному режимах

ТЕОРЕТИЧНІ ВІДОМОСТІ

Робота через командний рядок починається із введення пароля та підключення до сервера:



```
MySQL 8.0 Command Line Client
Enter password: █

MySQL 8.0 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.17 MySQL Community Server - GPL

Copyright (c) 2000, 2019, 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>
```

Далі, можна переглянути бази даних, розташовані на сервері за допомогою команди SHOW DATABASES:

```
MySQL 8.0 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.17 MySQL Community Server - GPL

Copyright (c) 2000, 2019, 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> show databases;
```

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| productsdb |
| sakila |
| sys |
| world |
+-----+
7 rows in set (0.02 sec)
```

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| productsdb |
| sakila |
| sys |
| world |
+-----+
7 rows in set (0.00 sec)
```

Встановлюємо БД за замовчуванням:

```
mysql> use world;
```

```
mysql> use world;
Database changed
mysql> _
```

Переглядаємо перелік таблиць:

```
mysql> show tables;_
```

```
+-----+
| Tables_in_world |
+-----+
| city |
| country |
| countrylanguage |
+-----+
3 rows in set (0.17 sec)
```

Та перелік записів таблиць:

```
mysql> select * from city;
```

```
mysql> select * from city;
```

ID	Name	CountryCode	District	Population
1	Kabul	AFG	Kabul	1780000
2	Qandahar	AFG	Qandahar	237500
3	Herat	AFG	Herat	186800
4	Mazar-e-Sharif	AFG	Balkh	127800
5	Amsterdam	NLD	Noord-Holland	731200

Створення таблиці

Для створення таблиць використовується команда **CREATE TABLE**. Ця команда застосовує ряд операторів, які визначають стовпці таблиці і їх атрибути. Загальний формальний синтаксис команди CREATE TABLE:

```
1 CREATE TABLE назва_таблиці
2 (назва_стовпця1 тип_даних атрибути_стовпця1,
3 назва_стовпця2 тип_даних атрибути_стовпця2,
4 .....
5 назва_стовпцяп тип_даних атрибути_стовпцяп,
6 атрибути_рівня_таблиці
7 )
```

Після команди CREATE TABLE йде назва таблиці. Ім'я таблиці виконує роль її ідентифікатора в базі даних, тому воно повинно бути унікальним. Потім в дужках перераховуються назви стовпців, їх типи даних і атрибути. У самому кінці можна визначити атрибути для всієї таблиці. Атрибути стовпців, а також атрибути таблиці вказувати необов'язково.

Створимо найпростішу таблицю. Для цього виконаємо наступний скрипт:

```
1 CREATE DATABASE
2 productsdb;
3
4 USE productsdb;
5
6 CREATE TABLE
7 Customers (
8 Id INT,
9 Age
10 INT,
11 FirstName VARCHAR(20),
12 LastName VARCHAR(20)
13 );
```

Таблиця не може створюватися сама по собі. Вона завжди створюється в певній базі даних. Спочатку тут створюється база даних productsdb. І потім, щоб вказати, що всі подальші операції, в тому числі створення таблиці, будуть проводитися з цією базою даних, застосовується команда **USE**.

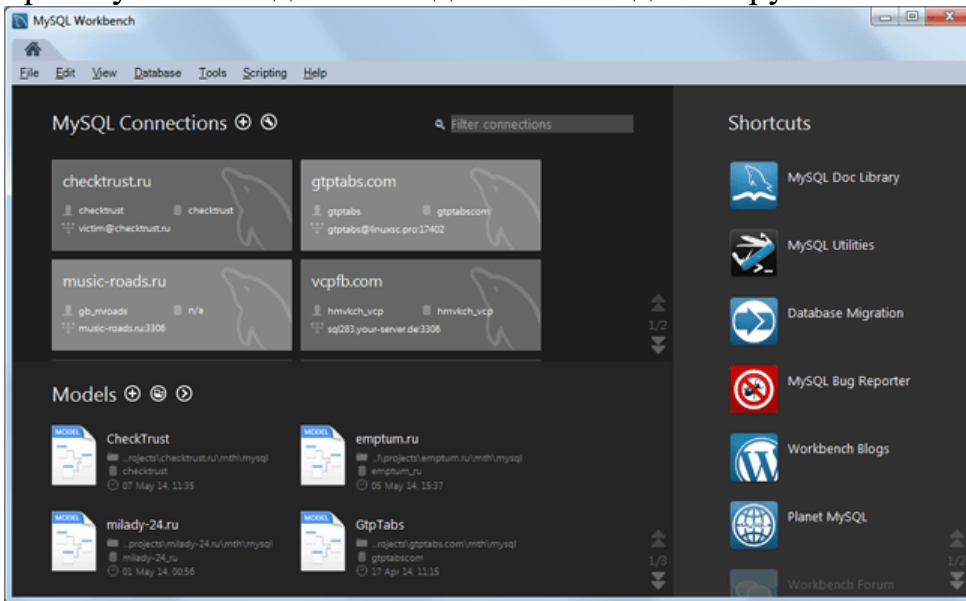
Далі власне йде створення таблиці, яка називається Customers. Вона визначає чотири стовпці: Id, Age, FirstName, LastName. Перші два стовпці представляють ідентифікатор клієнта і його вік і мають тип INT, тобто будуть зберігати числові значення. Наступні стовпці представляють ім'я і прізвище клієнта і мають тип VARCHAR(20), тобто представляють рядок

довжиною не більше 20 символів. В даному випадку для кожного стовпця визначено ім'я і тип даних, при цьому атрибути стовпців і таблиці в цілому відсутні.

І в результаті виконання цієї команди буде створена база даних productsdb, в якій буде створена таблиця Customers.

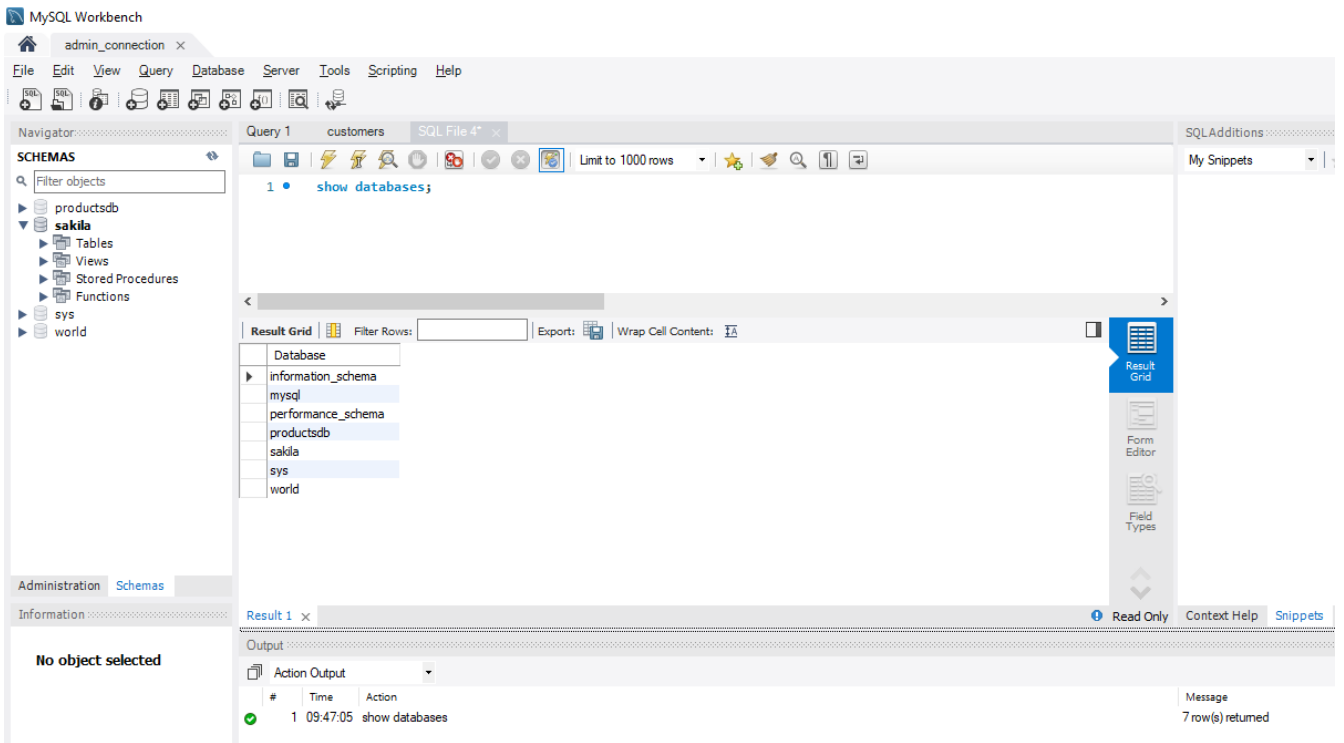
Початок роботи у програмі My SQL Workbench

Стартовий екран програми відображає основні напрямки її функціональності - проектування моделей баз даних та їх адміністрування:

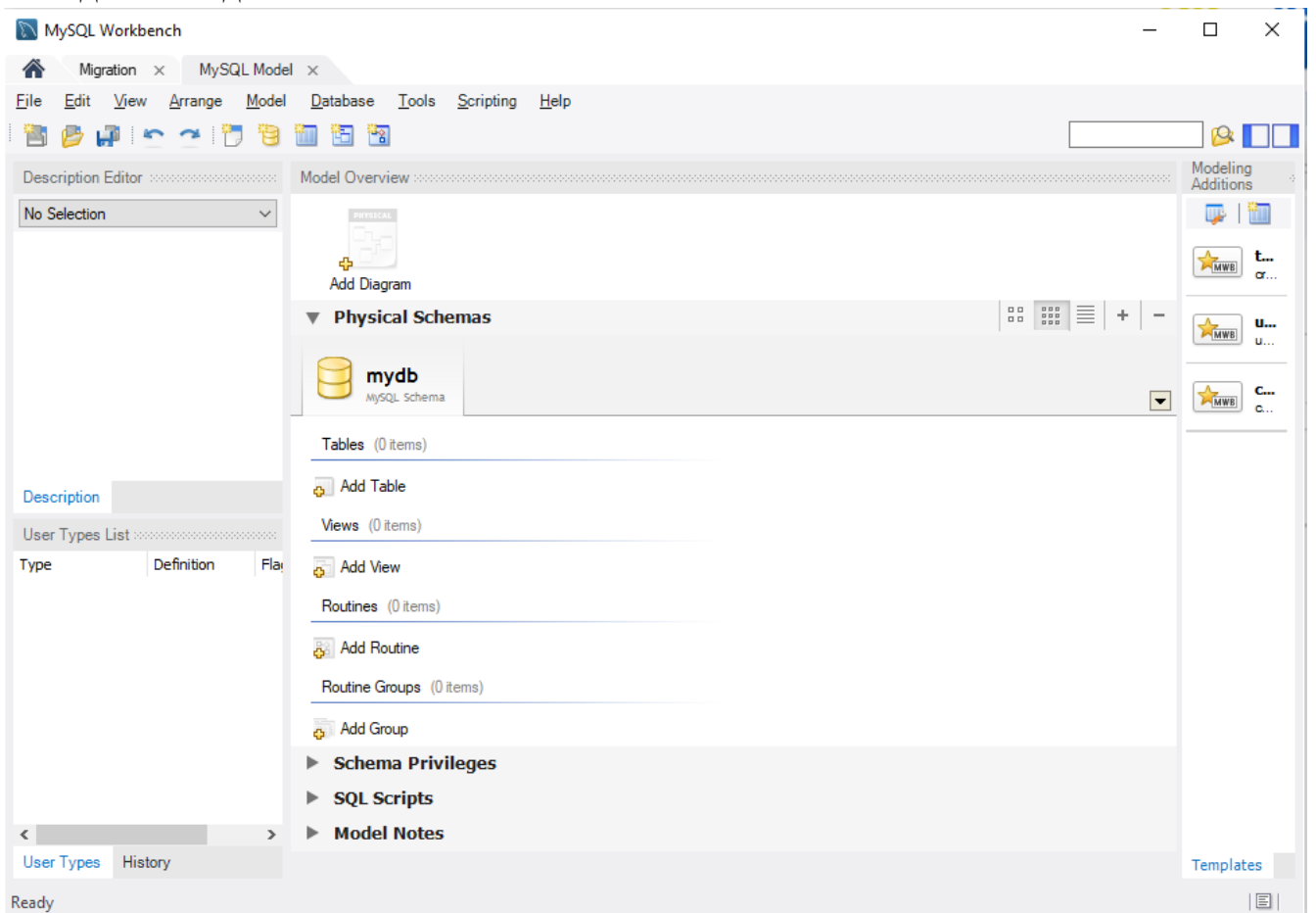


У верхній частині екрану знаходиться список підключень до MySQL серверів ваших проектів, а список останніх відкритих моделей даних - в нижній частині екрана. Робота зазвичай починається зі створення схеми даних або завантаження існуючої структури в MySQL Workbench.

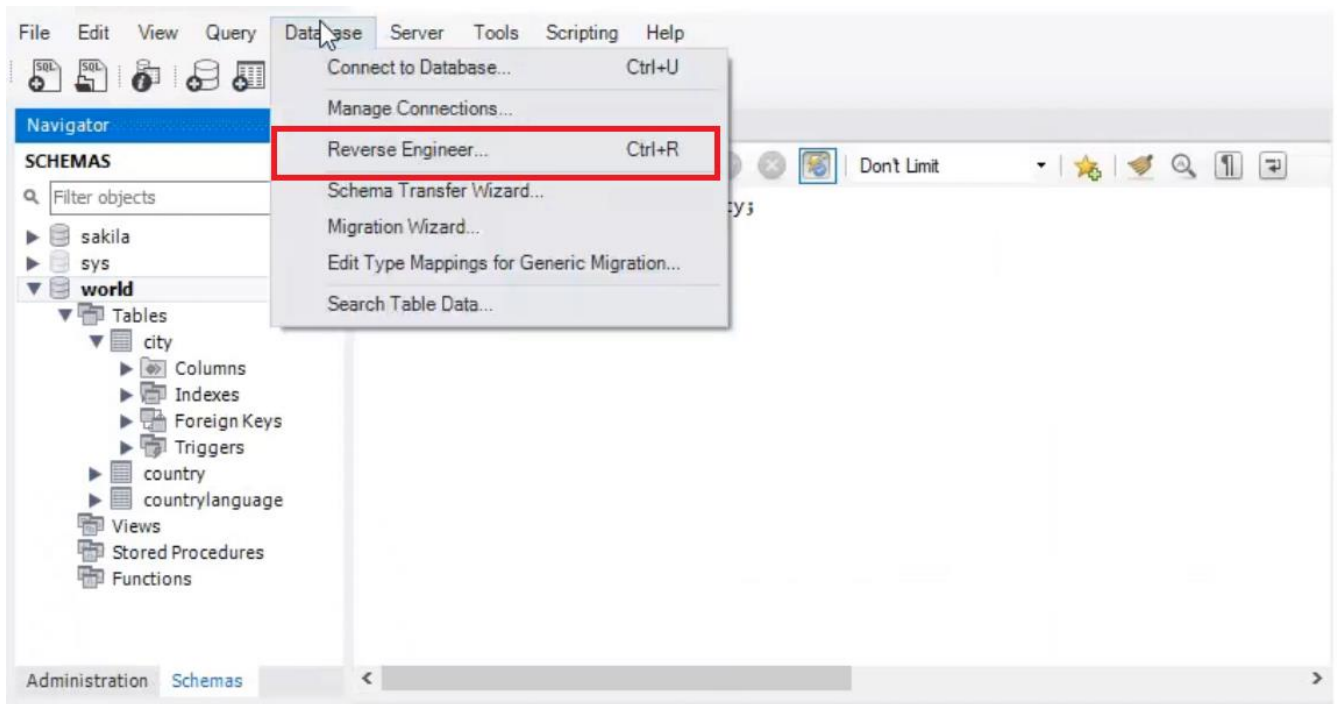
Перегляд переліку БД через графічну оболонку (вкладка Schemas):



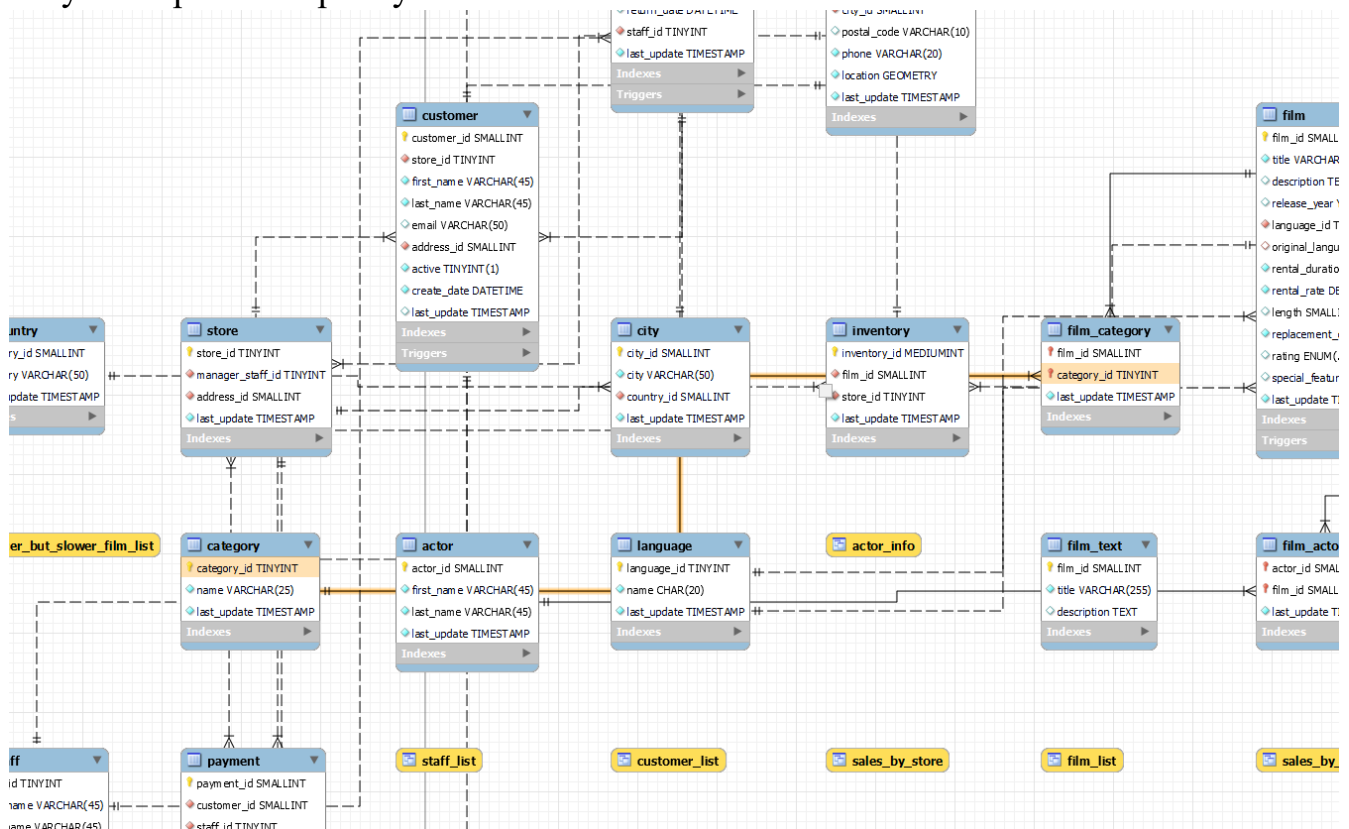
Вкладка моделювання:



Рейнженеринг (створення діаграми за наявними таблицями):



Результат реінженерингу



ЗАВДАННЯ ДЛЯ САМОСТІЙНОЇ РОБОТИ

Завдання №1. Провести аналіз довільного сайту з продажу товарів медичного призначення.

Дані про товари занести у таблицю. Таблицю створити та заповнити у текстовому редакторі. Наприклад:

Таблиця. Товари.

Код товару	Назва	Опис	Ціна	Акція
1	Костилі для дорослих	Розраховано на зріст людини від 1,5м до 2м	2000	
2	Пояс підтримуючий	Для вагітних	1000	

Завдання №2. Провести аналіз документів із замовленнями
Дані про користувачів занести у таблицю

Таблиця . Особисті дані замовника

Код замовника	Прізвище	Ім'я	Телефон	Електронна пошта	Нова пошта
1					
2					

Завдання №3. Провести аналіз документів із замовленнями
Дані про зміст замовлення занести у таблицю

Таблиця 4. Зміст замовлення

Код замовлення	Код товару	Кількість
1		1
1		2
1		1

Завдання №5. Провести аналіз документів із замовленнями
Дані обліку замовлення занести у таблицю

Код замовлення	Код замовника	Дата замовлення	Виконання замовлення	Оплата	Знижка

Завдання №6. Відкрити MySQL Command Line Client, MySQL Shell, MySQL Workbench. Спробувати переглянути допомогу, переглянути бази даних на сервері, змінити поточну базу даних, переглянути її таблиці та їх вміст.

Завдання №7. Відкрити MySQL Workbench. Згенерувати EER-діаграму за створеною базою даних.

Завдання №8. Створити таблицю з різними типами даних

Лабораторна робота №3

Тема: Операції над структурними елементами БД, додавання, редагування та видалення записів таблиці

Мета: Навчитися додавати, редагувати та видаляти дані у таблиці. Навчитися змінювати та видаляти стовпці таблиці бази даних та їх атрибути

ТЕОРЕТИЧНІ ВІДОМОСТІ

Перейменування таблиць

Якщо після створення таблиці ми захочемо її перейменувати, то для цього потрібно використовувати команду **RENAME TABLE**, яка має наступний синтаксис:

```
1 RENAME TABLE старое_название TO новое_название;
```

Наприклад, перейменуємо таблицю Customers в Clients:

```
1 RENAME TABLE Customers TO Clients;
```

Повне видалення даних

Для повного видалення даних, очищення таблиці застосовується команда **TRUNCATE TABLE**. Наприклад, очистимо таблицю Clients:

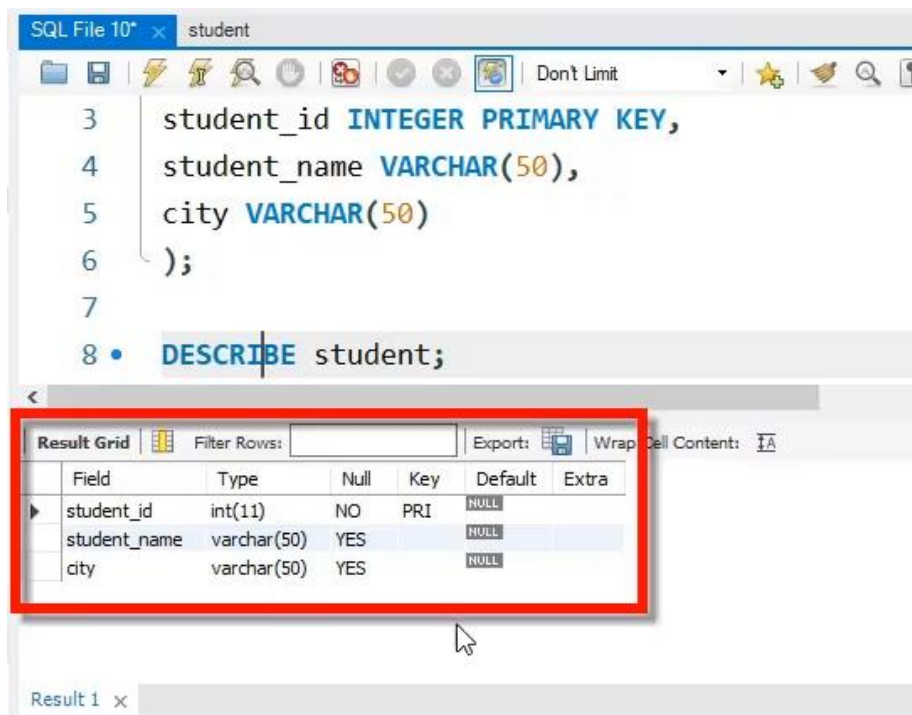
```
1 TRUNCATE TABLE  
Clients;
```

Видалення таблиць

Для видалення таблиці з БД застосовується команда **DROP TABLE**, після якої вказується назва видаляється таблиці. Наприклад, видалимо таблицю Clients:

```
1 DROP TABLE  
Clients;
```

Команда **DESCRIBE** дозволяє передивитись, з яких стовпців складається таблиця



Команда add column використовується для додавання нового стовпця в таблицю MySQL. Ви можете вказати ім'я, тип і позицію нового стовпця. Команда add використовується для додавання кількох стовпців одночасно.

Приклад команди add column:

```
ALTER TABLE books ADD COLUMN book_category NVARCHAR (200);
```

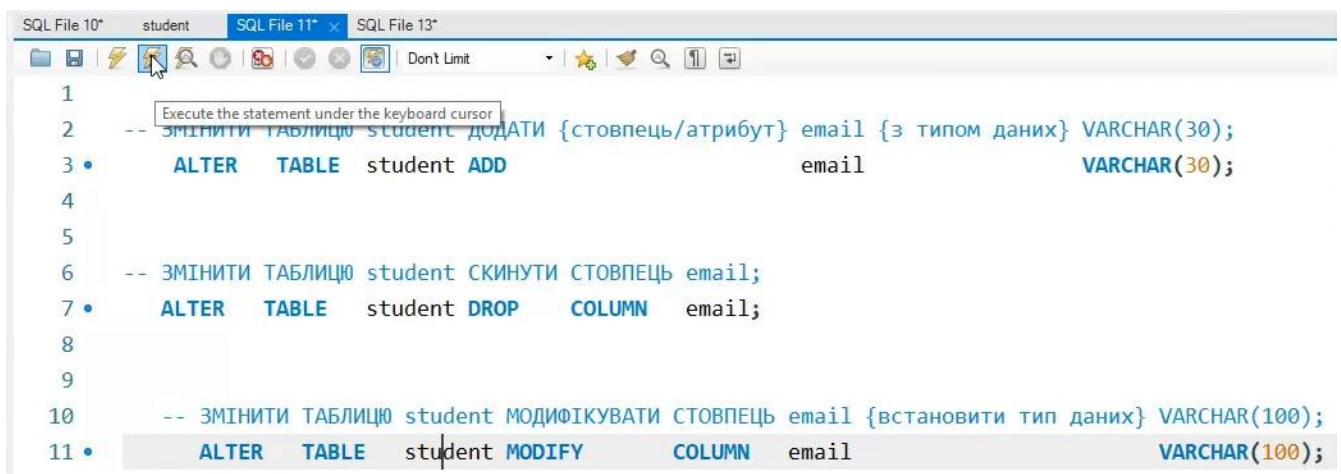
Приклад команди add:

```
ALTER TABLE books ADD book_title VARCHAR (100), ADD book_author VARCHAR (50);
```

DELETE використовується для видалення власне даних

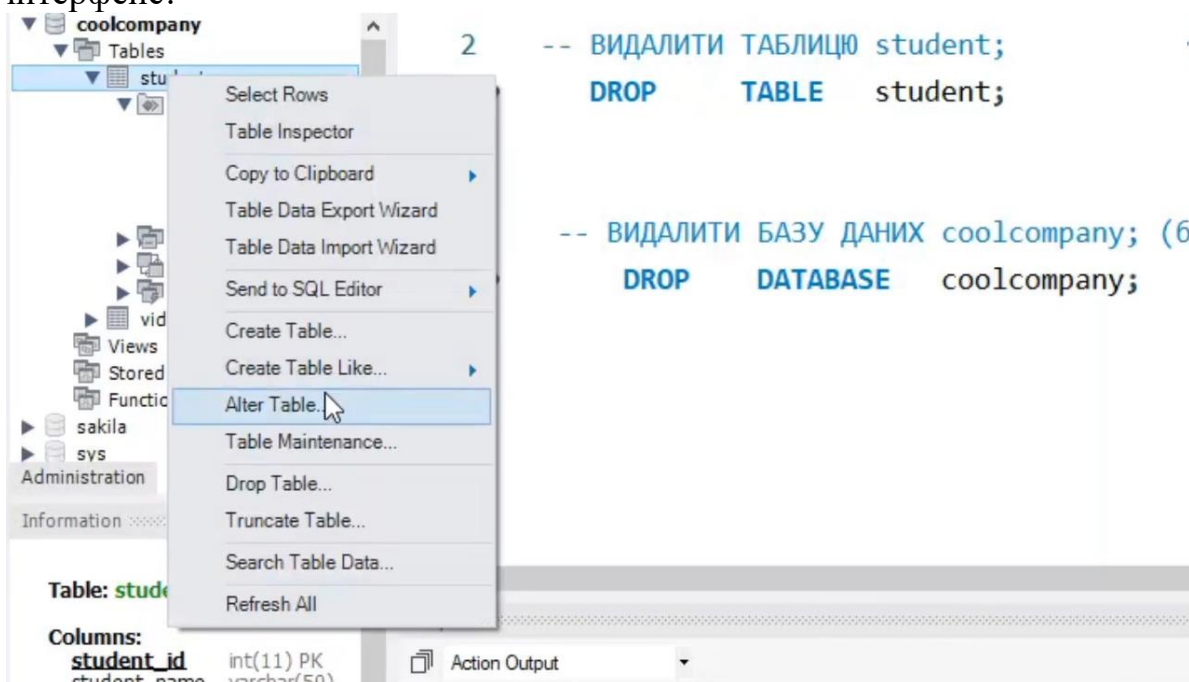
DROP для видалення структурних об'єктів БД

```
7 • ALTER TABLE student DROP COLUMN email;
```

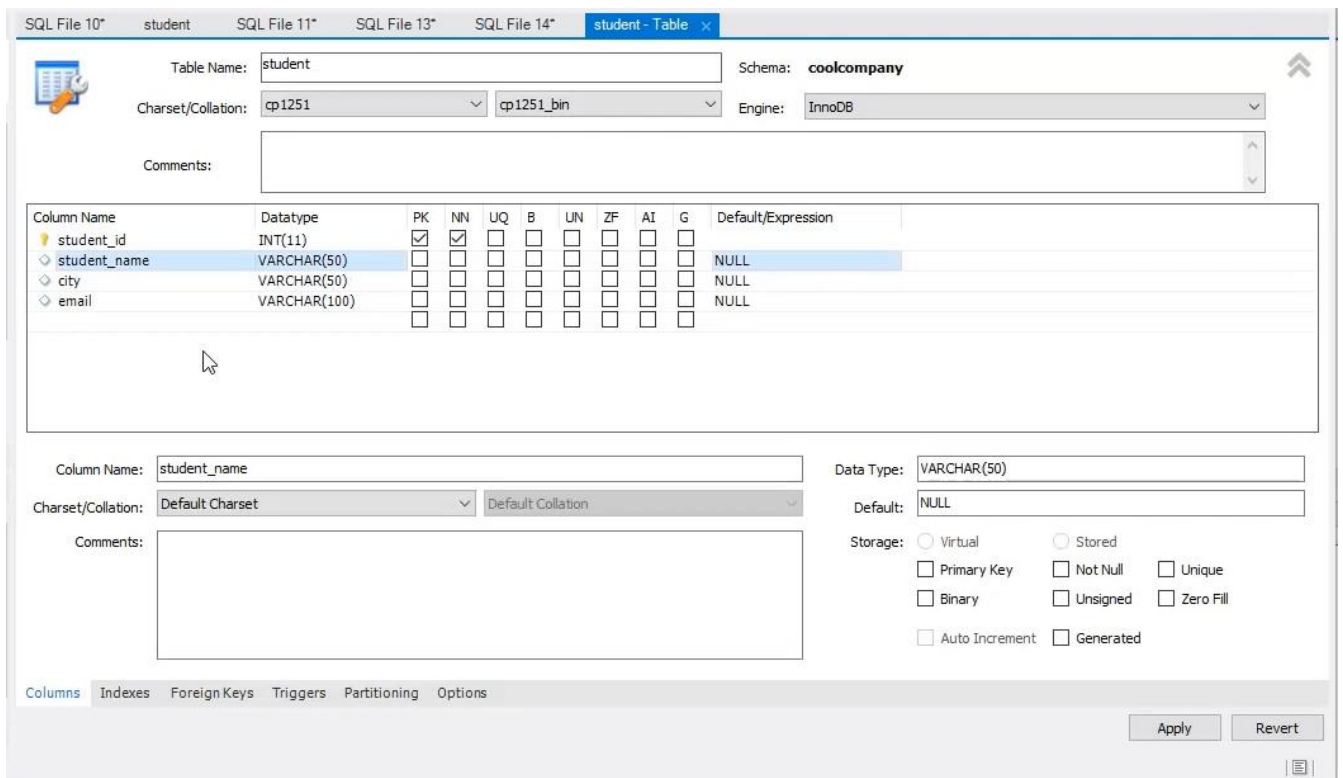


```
SQL File 10* student SQL File 11* SQL File 13* SQL File 14* x
[Icons] | Don't Limit
1
2 -- ВИДАЛИТИ ТАБЛИЦЮ student;
3 • DROP TABLE student;
4
5
6 -- ВИДАЛИТИ БАЗУ ДАНИХ coolcompany; (безповоротно)
7 • DROP DATABASE coolcompany;
```

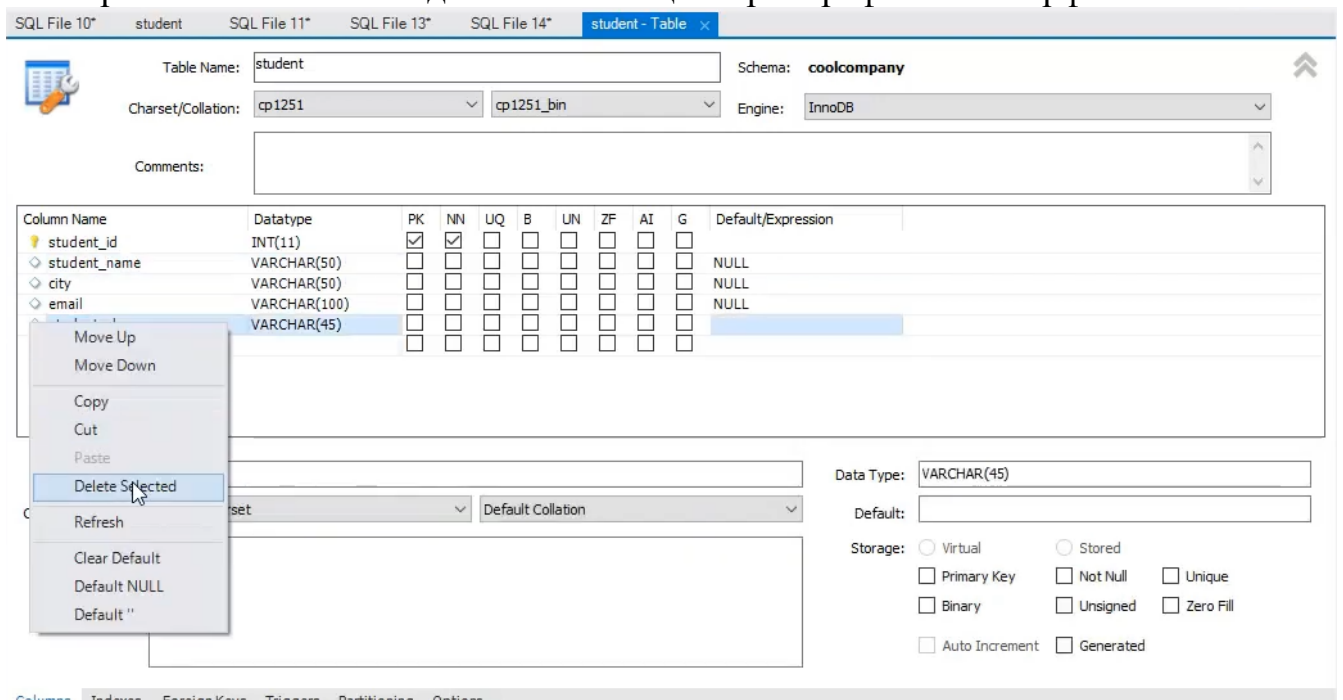
Альтернативний спосіб внесення змін у таблиці через графічний інтерфейс:



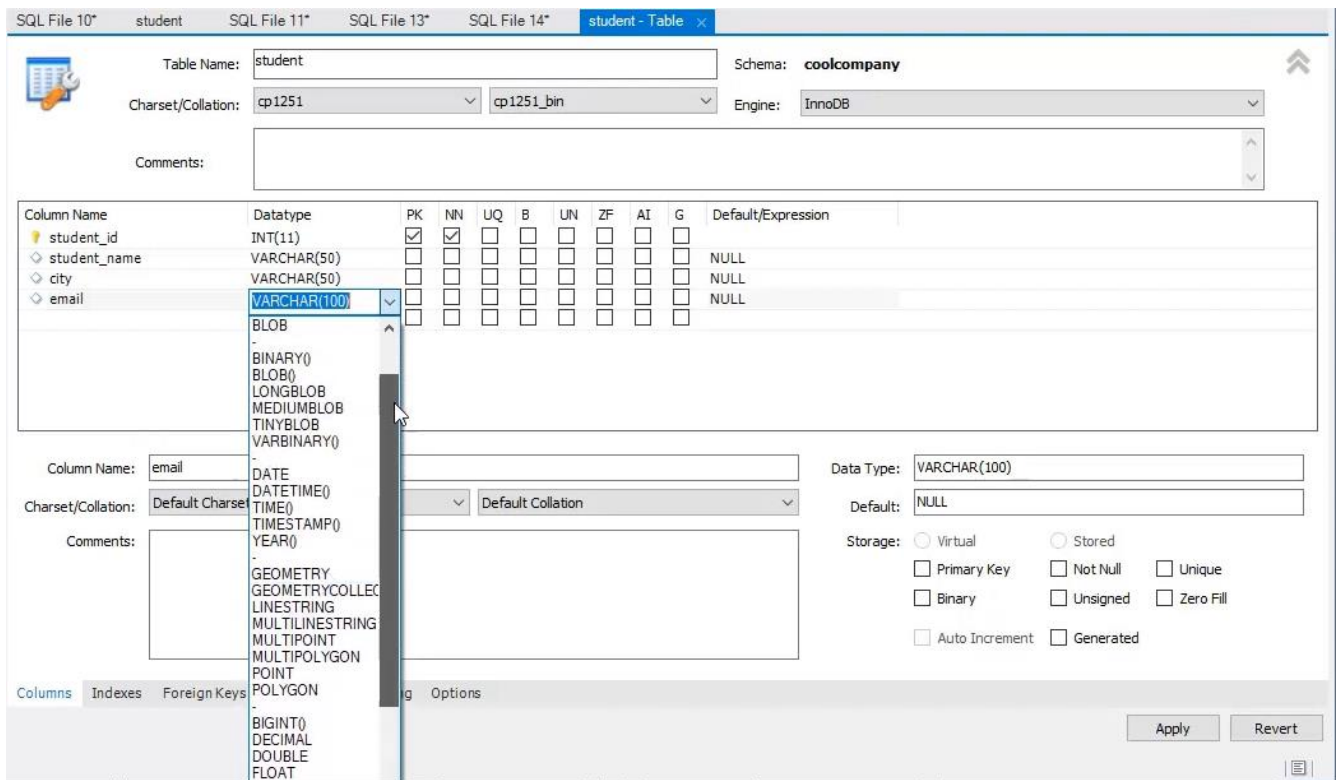
Так виглядатиме діалогове вікно Alter Table:



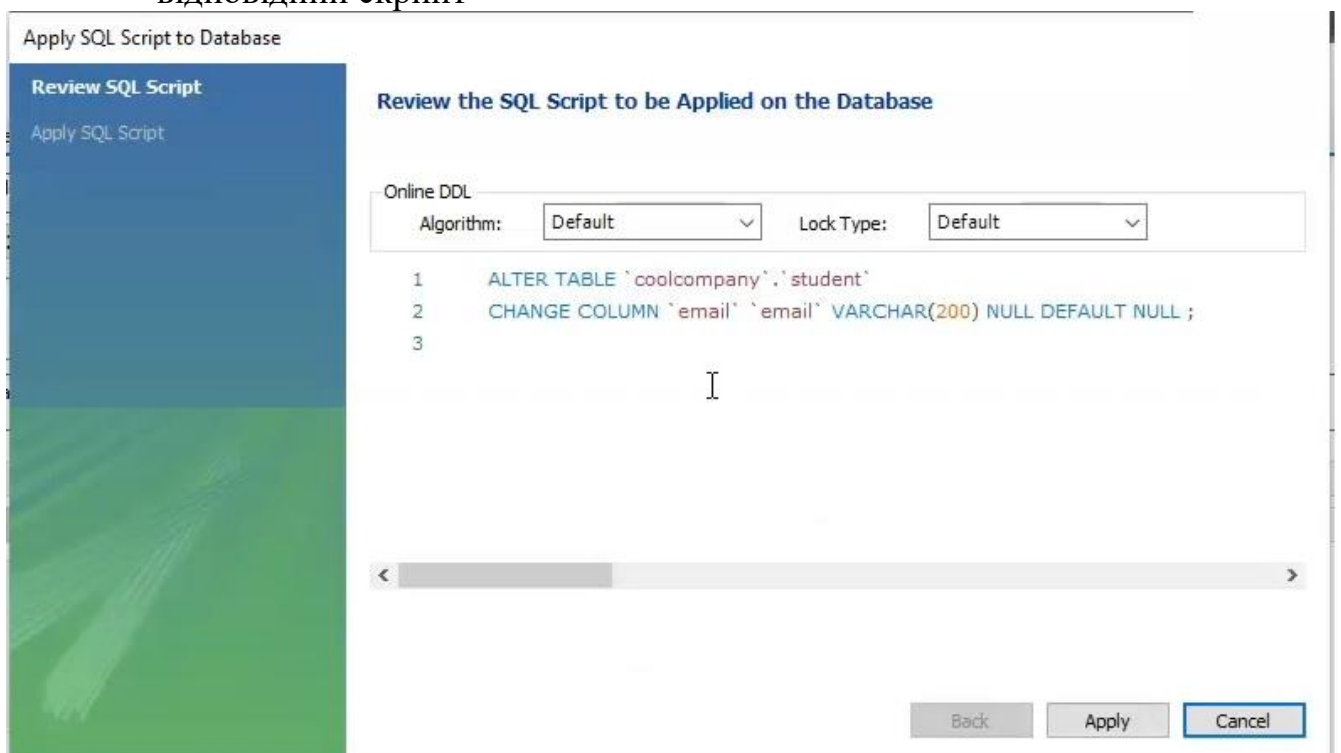
Альтернативний спосіб видалення стовпців через графічний інтерфейс:



Змінюємо тип даних у графічний спосіб:



Після натискання кнопки Apply, MySQL Workbench генерую відповідний скрипт

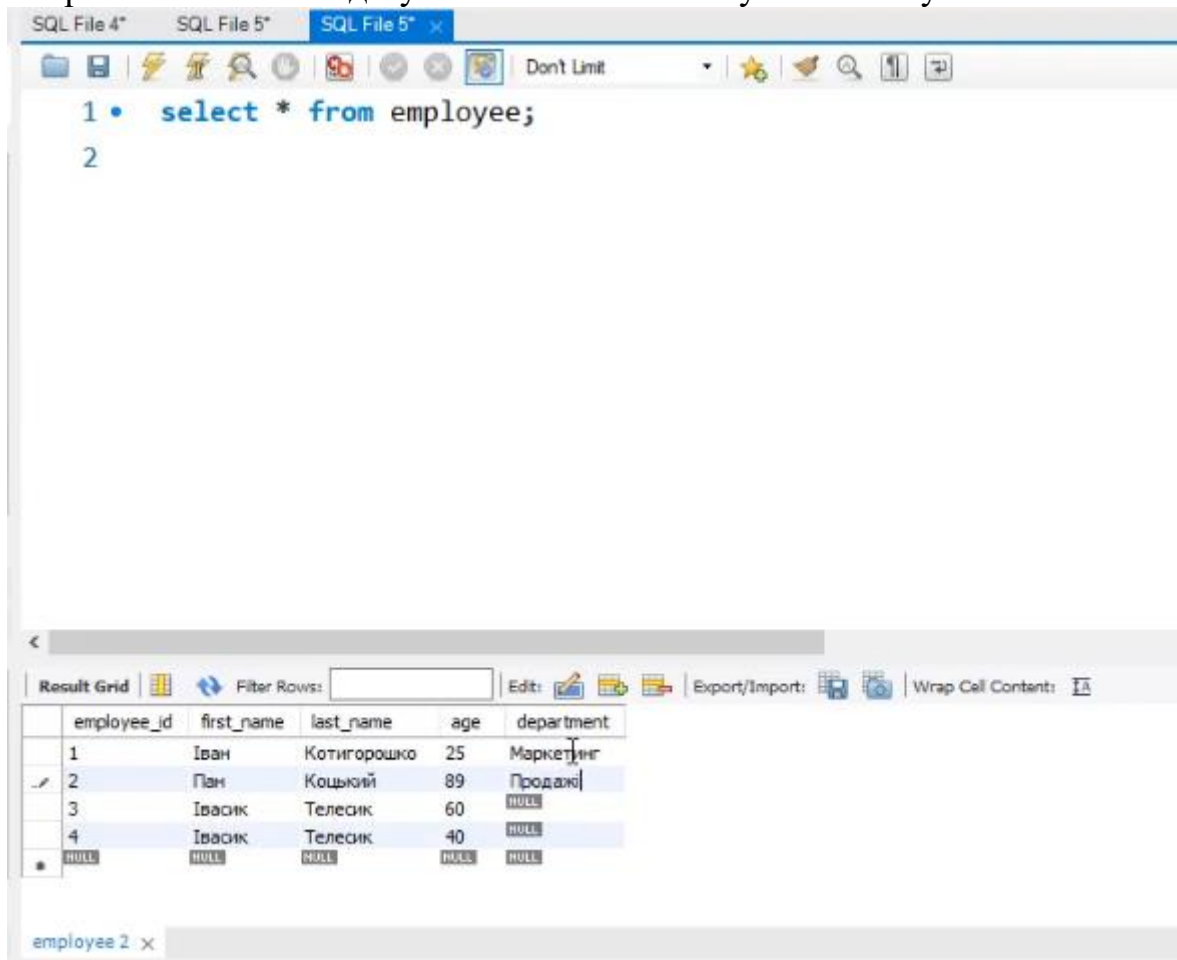


Для наповнення таблиці контентом можна використовувати команду INSERT, наприклад (якщо попередньо створити таблицю customer з двома текстовими стовпцями):

```
6 • INSERT INTO customer
7 VALUES ('a', 'b');
```

Несумісні типи даних вставляти не можна (коли тип даних, що додається відрізняється від того типу, який був заданий при створенні таблиці), в області Output буде виведено повідомлення про помилку.

Наповнення таблиці контентом у графічний спосіб виконується подвійним кліком миші на комірку, але це можливо тільки для таблиць, в яких задано первинний ключ. Редагування записів виконується наступним чином:



ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

Завдання 1. Створити таблицю у БД. Спробувати переіменувати та видалити її.

Завдання 2. Наповнити таблиці бази даних значимим контентом, відсортувати один стовпець у спадаючому порядку, а інший у порядку збільшення значень.

Завдання 3. Відредагувати дані та спробувати видалити їх із таблиці програмним та графічним способами.

Завдання 4. Додати стовпець, змінити його довжину та тип даних. Зробити це програмним способом та за допомогою засобів графічного інтерфейсу

Завдання 5. Написати SQL-команду для створення та видалення таблиці БД. За допомогою графічної оболонки Workbench створити, а потім видалити таблицю із БД

Завдання 6. Написати SQL-запит, який відбирає лише деякі стовпці таблиці (не всі) за певними критеріями, скориставшись умовою WHERE, та продемонструвати його виконання. Бажано використати оператори IN, NOT IN, !=, <>, LIKE, BETWEEN і т.ін.

Завдання 7. Спробувати написати кілька запитів з використанням агрегатних функцій COUNT(*), COUNT(column), MIN(column), MAX(column), AVG(column), SUM(column).

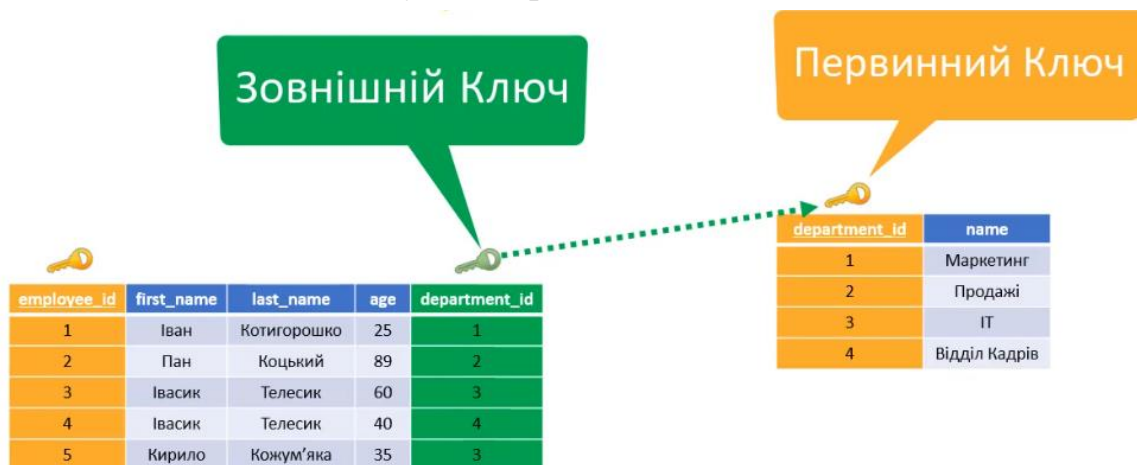
Лабораторна робота №4

Тема: Первинний ключ і зовнішній ключ. Композитний (складений) первинний ключ.

Мета: Навчитися створювати таблиці з первинним та зовнішнім ключами.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Коли в нас більше одного разу повторюються у БД текстові записи, це ознака необхідності створення ще однієї таблиці і додавання зовнішнього ключа. Зовнішній ключ вказує на первинний ключ в іншій таблиці.



Зв'язок між таблицями можна зробити, використовуючи інструкцію SQL INNER JOIN:

```

1 ALTER TABLE employee -- змінити таблицю employee
2 ADD CONSTRAINT fk_employee_on_dept -- додати ОБМЕЖЕННЯ {з іменем} fk_employee_on_dept
3 FOREIGN KEY (department_id) -- яке буде {типом} "зовнішній ключ" {для стовпчика} (department_
4 REFERENCES department (department_id); -- і буде прив'язаним до {стовпця} (department_id) в т
5

```

#	Time	Action	Message	Duration / Fetch
2	19:01:14	SELECT * FROM coolcompany.employee	6 row(s) returned	0.000 sec / 0.000 sec
3	19:01:34	ALTER TABLE employee -- змінити таблицю employee ADD CONSTRAINT fk_...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.344 sec

Дочірна таблиця не може містити неіснуючі записи із батьківської, якщо такого значення зовнішнього ключа не існує, цей принцип у базах даних називається достовірністю посилання зовнішнього ключа (referential integrity).

Атрибути полів

За допомогою атрибутів можна налаштувати поведінку стовпців.

Розглянемо, які атрибути ми можемо використовувати.

PRIMARY KEY

Атрибут **PRIMARY KEY** задає первинний ключ таблиці.

```

1 USE productsdb;
2
3 CREATE TABLE Customers
4 (
5     Id INT PRIMARY KEY,
6     Age INT,
7     FirstName VARCHAR(20),
8     LastName VARCHAR(20)
9 );

```

Первинний ключ унікально ідентифікує рядок в таблиці. В якості первинного ключа необов'язково повинні виступати стовпці з типом int, вони можуть представляти будь-який інший тип.

Установка первинного ключа на рівні таблиці:

```

1 USE productsdb;
2 CREATE TABLE
3 Customers (
4     Id INT,
5     Age
6     INT,
7     FirstName
8     VARCHAR(20),
9     LastName
10    VARCHAR(20),
11    PRIMARY KEY(Id)
12 );

```

Первинний ключ може бути складеним. Такий ключ використовує відразу декілька стовпців, щоб унікально ідентифікувати рядок в таблиці, наприклад:

```

1 CREATE TABLE OrderLines
2 (
3     OrderId INT,
4     ProductId
5     INT, Quantity
6     INT, Price
7     MONEY,
8     PRIMARY KEY(OrderId,
        ProductId)
    )

```

Тут поля OrderId і ProductId разом виступають як складової первинний ключ. Тобто в таблиці OrderLines не може бути двох рядків, де для обох з цих полів одночасно були б одні і ті ж значення.

AUTO_INCREMENT

Атрибут **AUTO_INCREMENT** дозволяє вказати, що значення стовпця буде автоматично збільшуватися при додаванні нового рядка. Даний атрибут працює для стовпців, які представляють цілочисельний тип або числа з плаваючою крапкою.

```

1 CREATE TABLE Customers
2 (
3     Id INT PRIMARY KEY
4     AUTO_INCREMENT, Age INT,
5     FirstName VARCHAR(20),
6     LastName VARCHAR(20)
7 );

```

В даному випадку значення стовпця Id кожної нової доданої рядка буде збільшуватися на одиницю.

UNIQUE

Атрибут **UNIQUE** вказує, що стовпець може зберігати тільки унікальні значення.

```

1 CREATE TABLE Customers
2 (
3     Id INT PRIMARY KEY
4     AUTO_INCREMENT, Age INT,
5     FirstName VARCHAR(20),
6     LastName VARCHAR(20),
7     Phone VARCHAR(13)
8     UNIQUE
    );

```

В даному випадку стовпець Phone, який представляє телефон клієнта, може зберігати тільки унікальні значення. І ми не зможемо додати в таблицю два рядки, у яких значення для цього стовпця буде збігатися.

Також ми можемо визначити цей атрибут на рівні таблиці:

```

1 CREATE TABLE Customers
2 (
3     Id INT PRIMARY KEY
4     AUTO_INCREMENT, Age INT,
5     FirstName
6     VARCHAR(20),
7     LastName
8     VARCHAR(20), Email
9     VARCHAR(30), Phone
10    VARCHAR(20),
        UNIQUE(Email, Phone)
    );

```


NULL і NOT NULL

Щоб вказати, чи може стовпець приймати значення **NULL**, при визначенні стовпця йому можна задати атрибут **NULL** або **NOT NULL**. Якщо цей атрибут явно не буде використаний, то за замовчуванням стовпець буде допускати значення **NULL**. Винятком є той випадок, коли стовпчик виступає в ролі первинного ключа - в цьому випадку за замовчуванням стовпець має значення **NOT NULL**.

```
1 CREATE TABLE Customers
2 (
3   Id INT PRIMARY KEY
4   AUTO_INCREMENT, Age INT,
5   FirstName VARCHAR(20) NOT
6   NULL, LastName VARCHAR(20)
7   NOT NULL, Email
8   VARCHAR(30) NULL,
9   Phone VARCHAR(20) NULL
   );
```

В даному випадку стовпець **Age** за замовчуванням буде мати атрибут **NULL**.

DEFAULT

Атрибут **DEFAULT** визначає значення за замовчуванням для стовпця. Якщо при додаванні даних для стовпця нічого очікувати передбачено значення, то для нього буде використовуватися значення за замовчуванням.

```
1 CREATE TABLE Customers
2 (
3   Id INT PRIMARY KEY
4   AUTO_INCREMENT, Age INT
5   DEFAULT 18,
6   FirstName VARCHAR(20) NOT NULL,
7   LastName VARCHAR(20) NOT NULL,
8   Email VARCHAR(30) NOT NULL
9   UNIQUE, Phone VARCHAR(20) NOT
   NULL UNIQUE
   );
```

Тут стовпець **Age** як значення за замовчуванням має число 18.

CHECK

Атрибут **CHECK** задає обмеження для діапазону значень, які можуть зберігатися в стовпці. Для цього після **CHECK** вказується в дужках умова, якому повинен відповідати стовпець або декілька стовпців. Наприклад, вік клієнтів не може бути менше 0 або більше 100:

```
1 CREATE TABLE Customers
2 (
3   Id INT AUTO_INCREMENT,
4   Age INT DEFAULT 18 CHECK(Age >0 AND Age
5   < 100), FirstName VARCHAR(20) NOT NULL,
6   LastName VARCHAR(20) NOT NULL,
7   Email VARCHAR(30) CHECK(Email
8   !="), Phone VARCHAR(20)
9   CHECK(Phone !=")
   );
```

Крім перевірки віку тут також перевіряється, що стовпці **Email** і **Phone** не можуть мати порожній рядок як значення (порожній рядок не є еквівалентною значенням **NULL**).

Для з'єднання умов використовується ключове слово **AND**.

Умови можна задати у вигляді операцій порівняння більше (>), менше (<), не дорівнює (! =).

Також CHECK можна використовувати на рівні таблиці:

```
1 CREATE TABLE Customers
2 (
3     Id INT AUTO_INCREMENT,
4     Age INT DEFAULT 18,
5     FirstName VARCHAR(20) NOT NULL,
6     LastName VARCHAR(20) NOT NULL,
7     Email VARCHAR(30),
8     Phone VARCHAR(20),
9     CHECK((Age >0 AND Age <100) AND (Email !='') AND (Phone !=''))
10 );
```

Оператор CONSTRAINT. Встановлення імені обмежень

За допомогою ключового слова **CONSTRAINT** можна задати ім'я для обмежень. Вони вказуються після ключового слова **CONSTRAINT** перед атрибутами на рівні таблиці:

```
1 CREATE TABLE Customers
2 (
3     Id INT
4     AUTO_INCREMENT,
5     Age INT,
6     FirstName VARCHAR(20) NOT
7     NULL, LastName VARCHAR(20)
8     NOT NULL, Email
9     VARCHAR(30),
10    Phone VARCHAR(20) NOT NULL,
11    CONSTRAINT customers_pk PRIMARY KEY(Id),
12    CONSTRAINT customer_phone_uq
    UNIQUE(Phone),
    CONSTRAINT customer_age_chk CHECK(Age >0 AND
    Age <100)
);
```

В даному випадку обмеження для **PRIMARY KEY** називається **customers_pk**, для **UNIQUE** - **customer_phone_uq**, а для **CHECK** – **customer_age_chk**. Сенс встановлення імен обмежень полягає в тому, що згодом через ці імена ми зможемо управляти обмеженнями – видаляти або змінювати їх.

ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

Завдання 1. Провести логічне конструювання БД

Таблиця 1. Облік замовлень

1. Код замовлення
2. Код замовника
3. Код товару
4. Кількість
5. Дата замовлення
6. Знижки
7. Акції

Таблиця 2. Особисті дані замовника

1. Код замовника
2. Прізвище

3. Ім'я
4. Телефон
5. Електронна пошта
6. Нова пошта

Таблиця 3. Товари.

1. **Код товару**
2. Назва
3. Опис
4. Ціна
5. Акція

Таблиця 4. Облік

1. **Код замовлення**
2. **Код замовника**
3. Дата замовлення
4. Виконання замовлення
5. Оплата
6. Знижка

Таблиця 5. Зміст замовлення

1. Код замовлення
2. **Код товару**
3. Кількість

Завдання 2. Доповнити структуру БД новою таблицею Ціна

Таблиця 6. Ціна

1. Код товару
2. Назва товару
3. Ціна

Завдання №3. Відкрити MySQL Workbench, під'єднатись до сервера, створити нову БД і таблиці в ній з первинним та зовнішнім ключем за обраною тематикою. Наприклад, можна додати таблицю зі зв'язком через зовнішній ключ для кольору або марки автомобілів. Можна створити таблицю з переліком апаратного забезпечення ПК та окрему таблицю із моделями процесорів, або материнських плат, які будуть підставлятись у базову таблицю.

Завдання №4. Створіть дві таблиці, щоб зовнішній ключ однієї був первинним для іншої, і навпаки. Тобто, із взаємним зв'язком, наприклад, як відділи пов'язані із робітниками, і в кожному відділі є керівник, який є також робітником.

Завдання №5. Створіть таблицю, в якій стовпець із зовнішнім ключем відсилається на первинний ключ у цій же таблиці.

Завдання №6. Створіть 3 таблиці, одна з яких буде із композитним ключем, один із стовпців таблиць зробіть унікальним індексом.

Завдання №7. Створити таблицю з автоінкрементом для первинного ключа.

Лабораторна робота №5

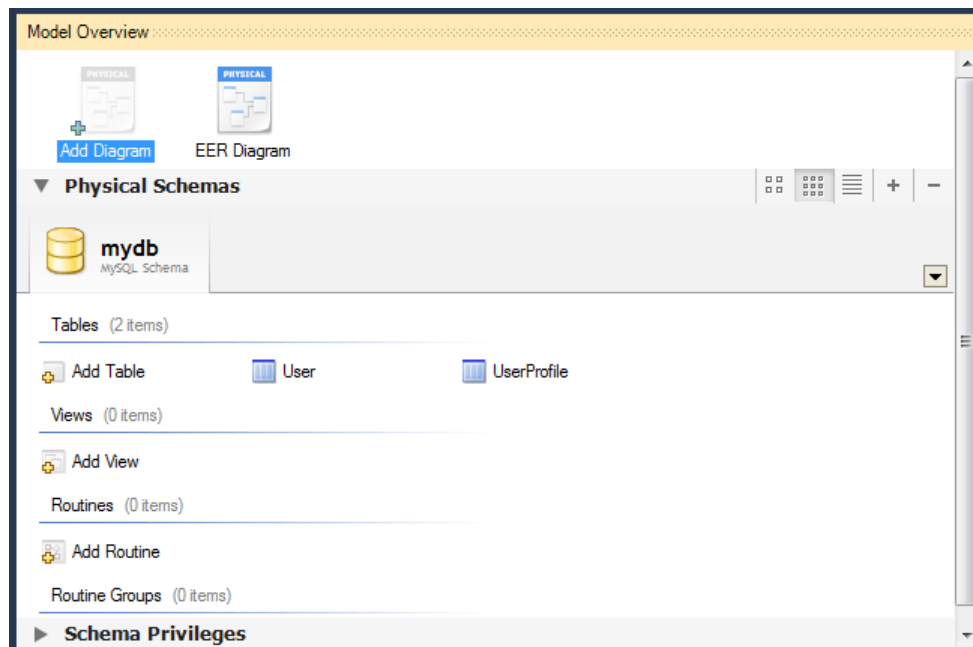
Тема: Створення та редагування моделі даних

Мета: Навчитися проектувати логічну структуру бази даних. Встановлювати зв'язки між таблицями.

ТЕОРЕТИЧНІ ВІДОМОСТІ

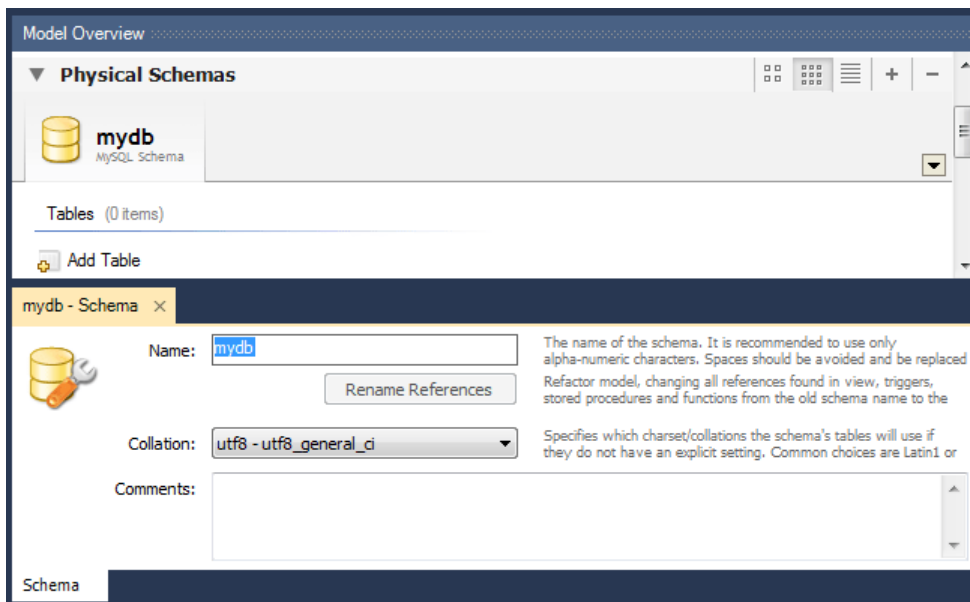
Логічна модель даних включає побудову логічної структури БД та встановлення зв'язків між таблицями. Таблиці бази даних подаються схематично. Діаграми "сутність-зв'язок" (EER model, Enhanced Entity–Relationship або Extended Entity–Relationship model) – це візуальний інструмент, призначений для розуміння читачами зв'язків сутностей у системі. Зазвичай він візуалізує інформацію в системах баз даних. Крім того, ця діаграма також може допомогти концептуалізувати загальний дизайн і структуру, включно з різними способами взаємодії об'єктів один з одним.

Для представлення схеми даних, сутностей і їх зв'язків в графічному вигляді в MySQL Workbench існує редактор EER-діаграм. В інтерфейсі EER-діаграм можна створювати і редагувати таблиці, додавати між ними зв'язки різних типів. Для створення діаграми у верхній частині екрану управління базою даних двічі натискаємо на іконку "+ Add Diagram":



Створення моделі даних

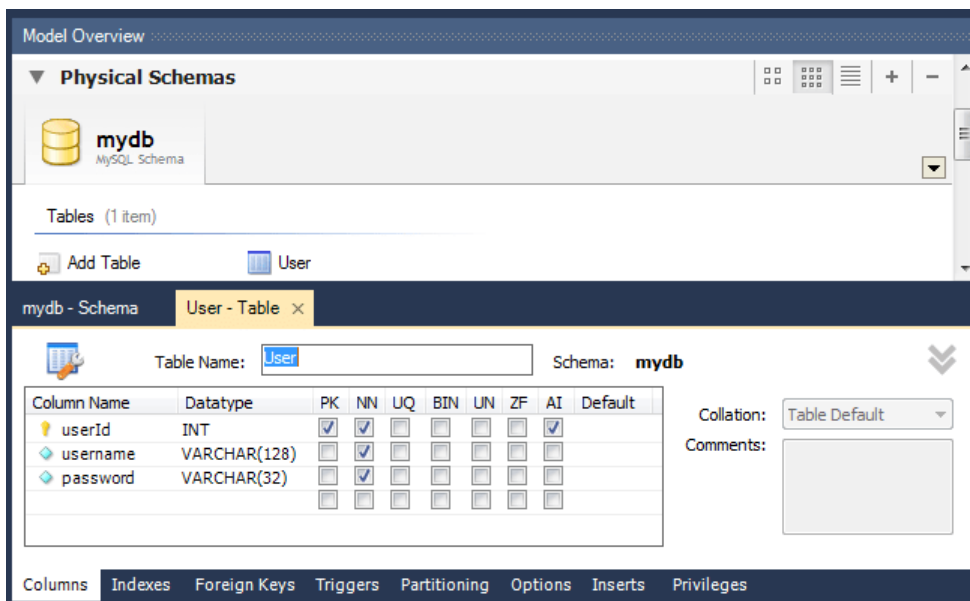
Для додавання моделі натискаємо плюсик поруч із заголовком "Models" або обираємо "File → New Model" (Ctrl + N):



На цьому екрані вводимо ім'я бази даних, вибираємо кодування за замовчуванням і, якщо потрібно, заповнюємо поле коментаря. Можна приступати до створення таблиць.

Додавання та редагування таблиці

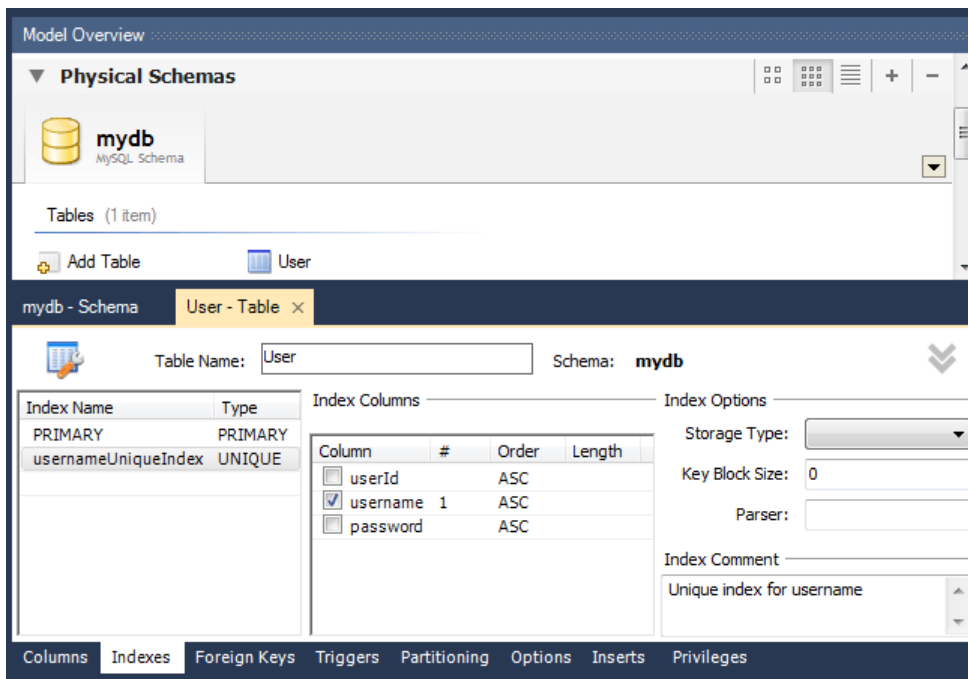
Список баз даних проекту і список таблиць в межах бази даних буде розташовуватися у вкладці "Physical Schemas". Щоб створити таблицю, двічі натискаємо на "+ Add Table":



Відкриється зручний інтерфейс для редагування списку полів і їх властивостей. Тут ми можемо поставити назву поля, тип даних, а так само встановити для полів різні атрибути: призначити поле *первинним ключем (PK)*, Позначити його *Not Null (NN)*, *бінарним (BIN)*, *унікальним (UQ)* та інші, встановити для поля *авто-інкрементування (AI)* і *значення за замовчуванням (Default)*.

Управління індексами

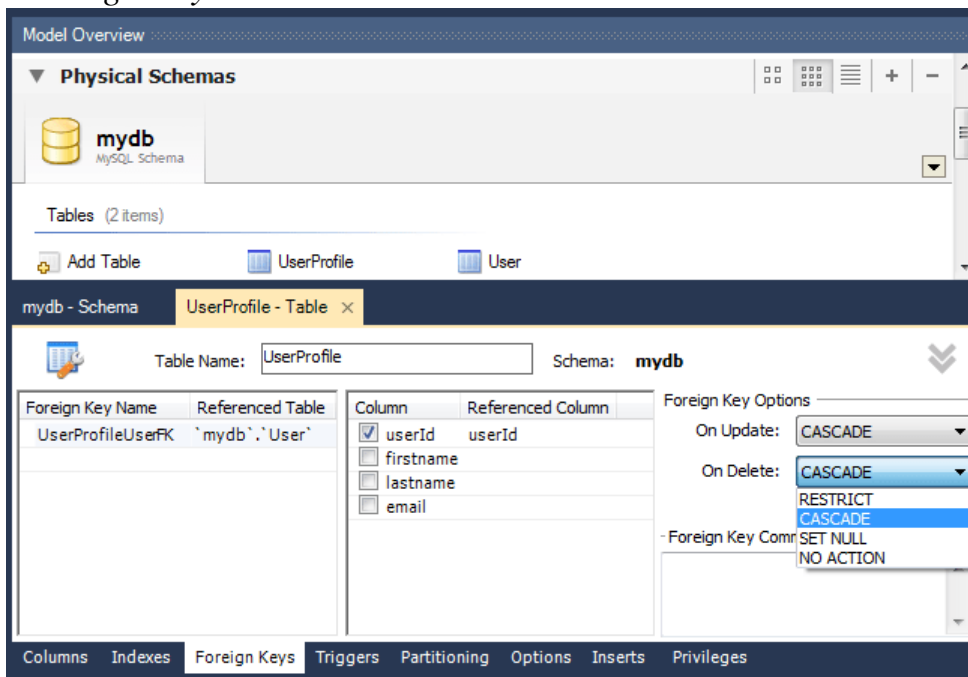
Додавати, видаляти і редагувати індекси таблиць можна у вкладці "Indexes" інтерфейсу управління таблицею:



Вводимо назву індексу, вибираємо його тип, потім галочками помічаємо в потрібному порядку список полів, що беруть участь в даному індексі. Порядок полів буде відповідати порядку, в якому були проставлені галочки. В даному прикладі використовується унікальний індекс до поля *username*.

Зв'язки між таблицями

Установка зовнішніх ключів і зв'язування таблиць можливе тільки для таблиць *InnoDB* (ця система зберігання даних обирається за замовчуванням). Для управління зв'язками в кожній таблиці використовується вкладка "*Foreign Keys*":



Для додавання зв'язку відкриваємо вкладку "*Foreign Keys*" дочірньої таблиці, вводимо ім'я зовнішнього ключа і вибираємо таблицю-батька. Далі в середній частині вкладки в графі *Column* вибираємо поле-ключ з дочірньої

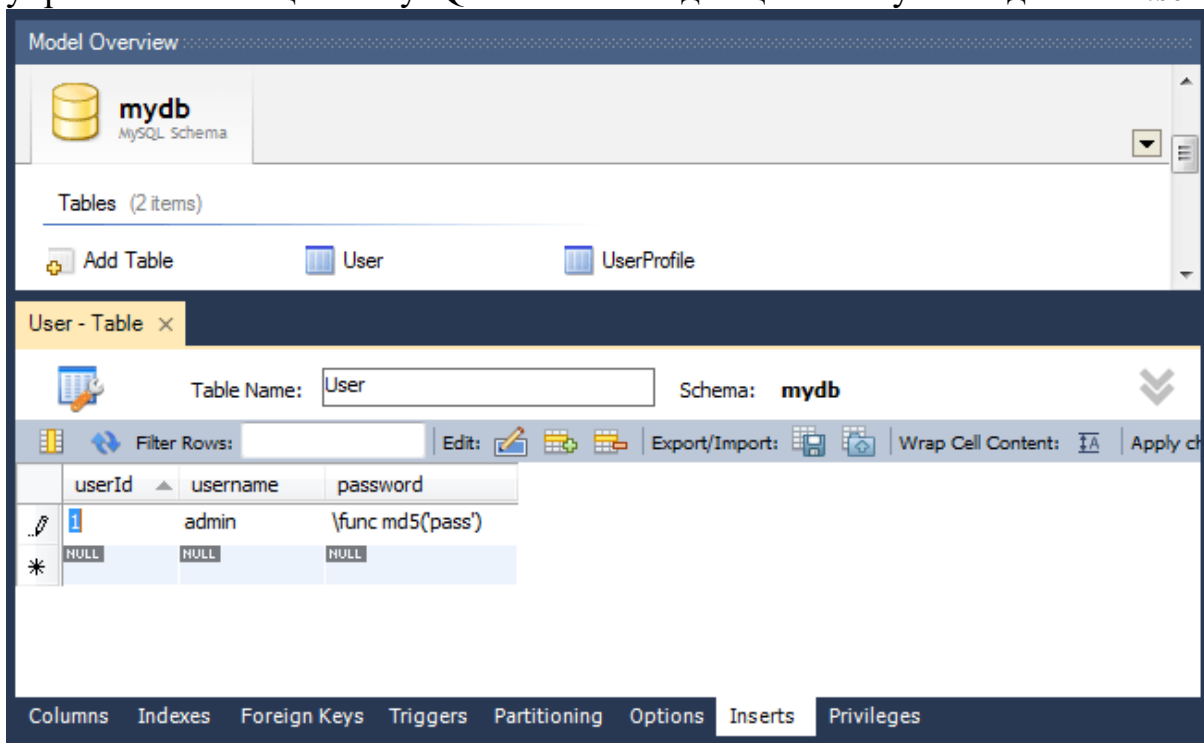
таблиці, а в графі *Referenced Column* – відповідне поле з батьківської таблиці (тип полів повинен збігатися). При створенні зовнішніх ключів в дочірній таблиці автоматично створюються відповідні індекси.

В розділі "*Foreign Key Options*" налаштовуємо поведінку зовнішнього ключа при зміні відповідного поля (*ON UPDATE*) і видаленні (*ON DELETE*) батьківського запису:

- *RESTRICT*- видавати помилку при зміні / видаленні батьківського запису
- *CASCADE*- оновлювати зовнішній ключ при зміні батьківського запису, видаляти дочірній запис при видаленні батька
- *SET NULL*- встановлювати значення зовнішнього ключа *NULL* при зміні / видаленні батька (Неприйнятно для полів, у яких встановлено прапор *NOT NULL!*)
- *NO ACTION*- не робити нічого, однак за фактом ефект аналогічний *RESTRICT*

У наведеному прикладі ми додаємо до дочірньої таблиці *UserProfile* зовнішній ключ для зв'язку з батьківською таблицею *User*. При редагуванні поля *userId* і видаленні позицій з таблиці *User* аналогічні зміни будуть автоматично відбуватися і з пов'язаними записами з таблиці *UserProfile*.

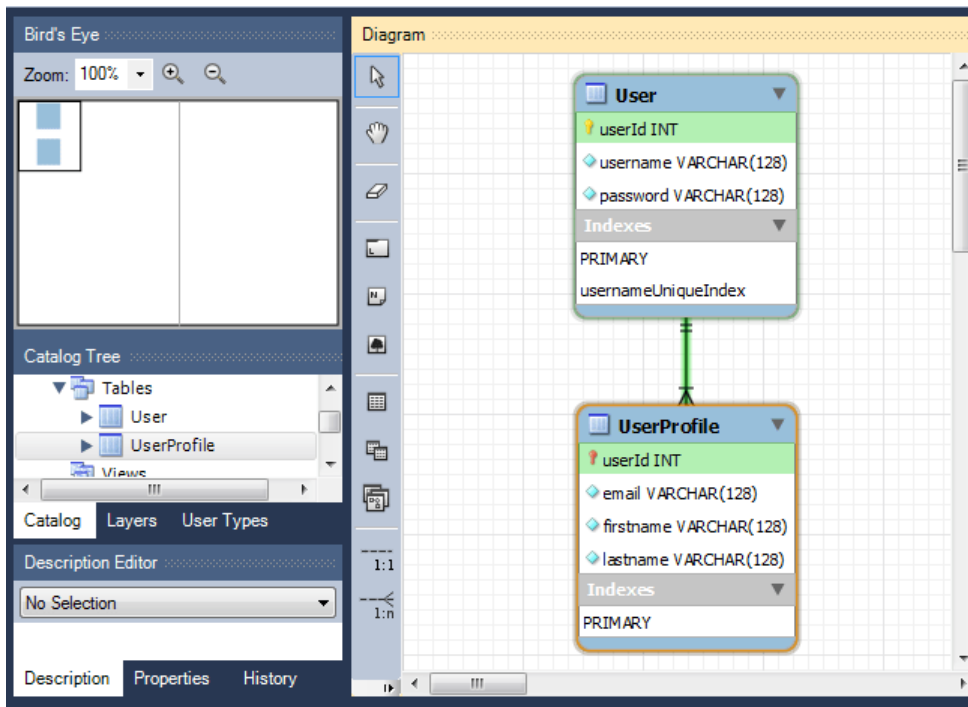
При створенні проекту в бази даних часто потрібно додавати стартові дані. Це можуть бути кореневі категорії, користувачі-адміністратори і т.д. В управлінні таблицями MySQL Workbench для цього існує вкладка "*Inserts*":



Як видно з прикладу, в разі, якщо перед записом в базу даних до даних потрібно застосувати якусь функцію MySQL, це робиться за допомогою синтаксису *\ Func functionName ("data")*, Наприклад, *\ Func md5 ("password")*.

Редагування EER діаграм

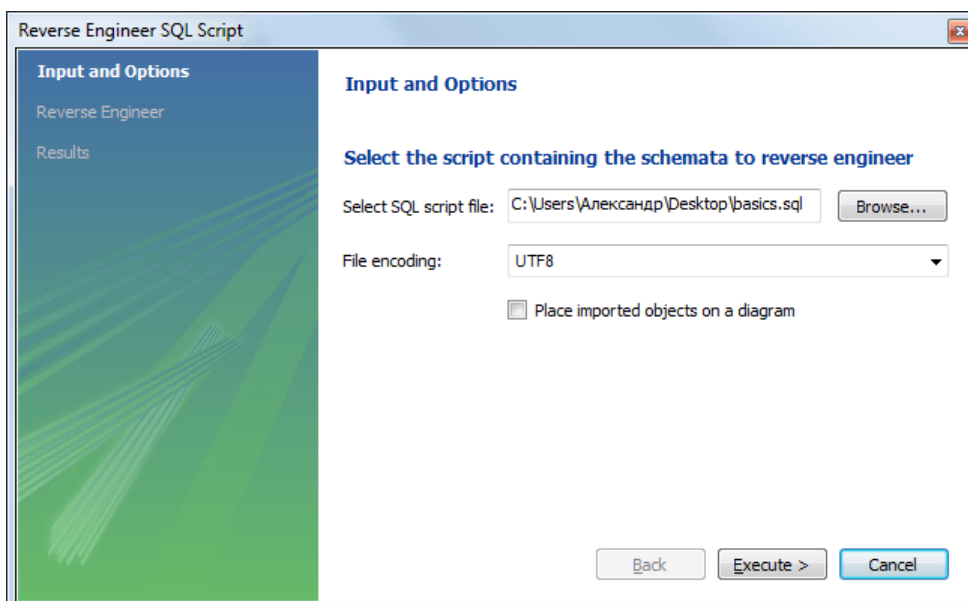
Щоб додати вже існуючу в схемі таблицю на діаграму, просто перетягніть її з панелі "Catalog Tree".



Для експорту схеми даних в графічний файл виберіть "File → Export", А потім один із варіантів (PNG, SVG, PDF, PostScript File).

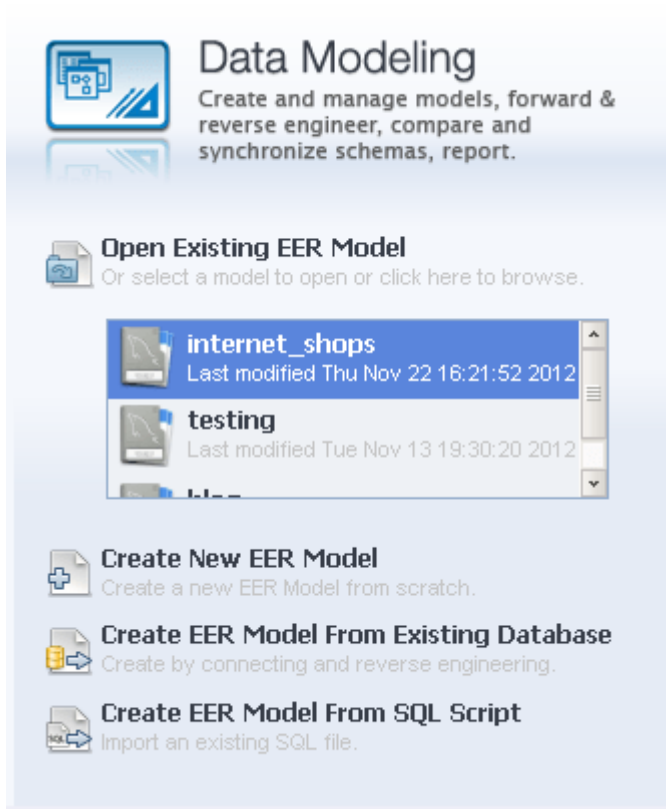
Імпорт існуючої схеми даних (з SQL дампа)

Якщо у нас вже є схема даних, її можна без зайвих зусиль імпортувати в MySQL Workbench для подальшої роботи. Для імпорту моделі з SQL-файлу обираємо "File → Import → Reverse Engineer MySQL Create Script ...", після чого обираємо потрібний SQL файл і натискаємо "Execute >"



В MySQL Workbench так само передбачений імпорт і синхронізація моделі даних наряду з віддаленим сервером. Для цього потрібно створити підключення віддаленого доступу до MySQL.

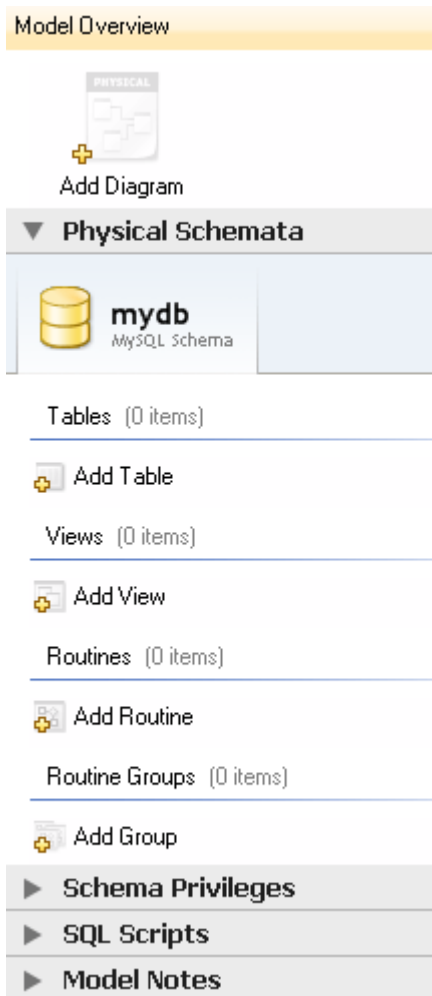
Зовнішній вигляд вікна програми, розділ «Моделювання даних» виглядає так:



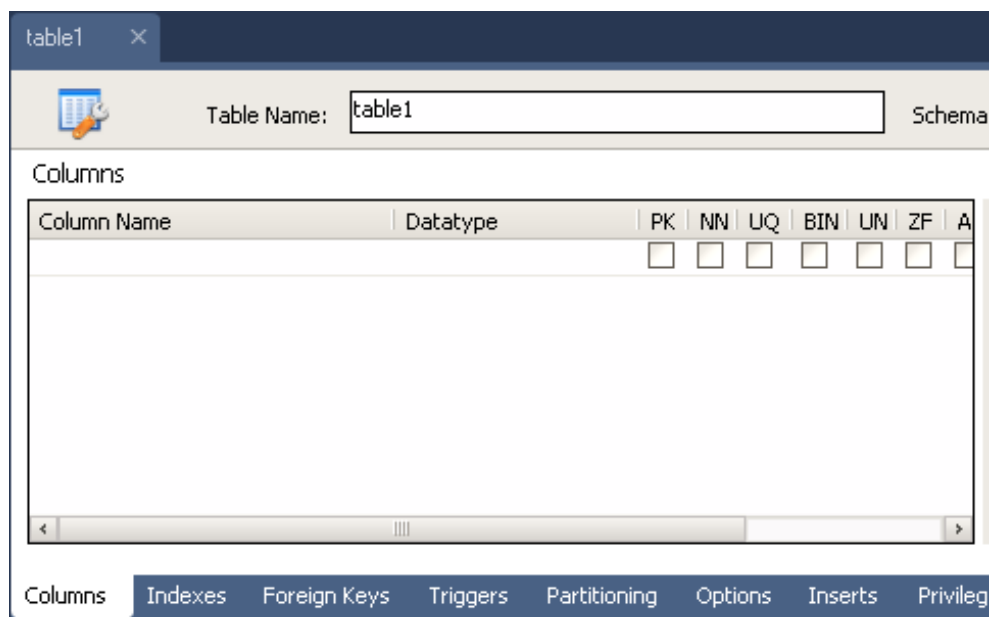
Для того, щоб відкрити існуючу модель треба натиснути на посилання: **Open Existing EER Model**, Для створення нової моделі – вибрати параметр: **Create New EER Model**. Щоб створити модель «сутність-зв'язок» з існуючої бази даних – натиснути на параметр: **Create EER Model From Existing Database**, а для створення EER моделі з SQL-скрипта потрібно вибрати: **Create EER Model From SQL Script**.

Порядок виконання роботи

Для створення нової моделі, скористаємося посиланням Create New EER Model, після натискання на нього з'явиться вікно з параметрами:



Для початку необхідно створити таблиці, для цього натиснемо на кнопку **Add Table**, з'явиться наступна форма:



Спочатку створимо таблицю **users**, яка буде зберігати дані про користувачів інформаційної системи, в поле **table Name** впишемо ім'я таблиці,

в розділі форми **Columns** створимо поля таблиці:

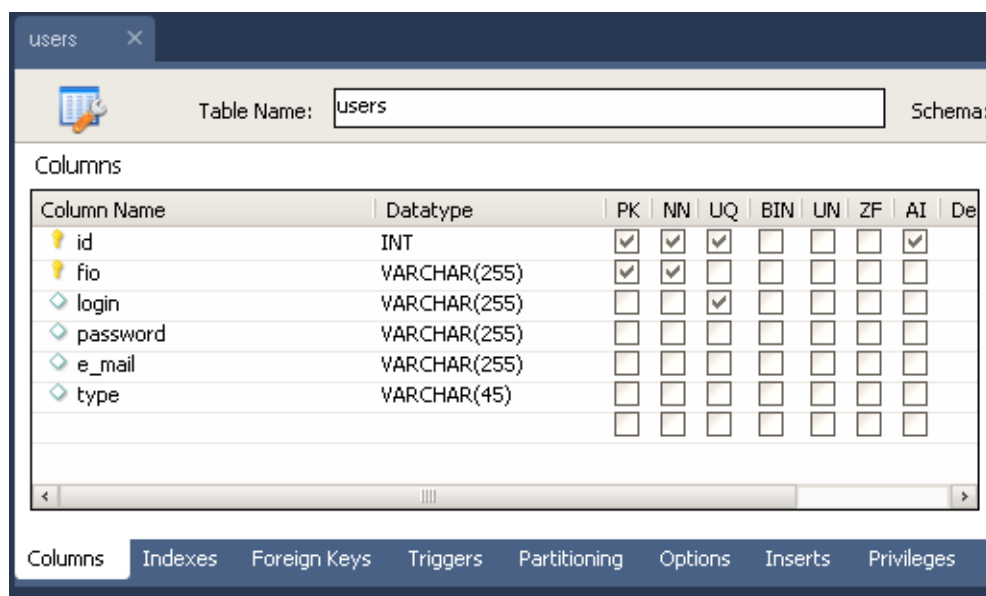
- Перше поле **id** буде містити унікальний номер користувача, задамо йому властивості: **Auto Increment, Not Null, Primary key** і **Unique**, в розділі **Data type** виберемо цілочисельний тип **integer**.

- Друге поле **fio**, де буде зберігається **П.І.Б.** користувача, встановимо поля властивості: **Not Null, Primary key**, в розділі **Data type** виберемо строковий тип **VARCHAR 255**.

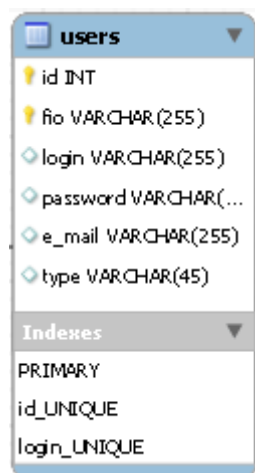
- Третє поле **login** буде містити логін користувача, воно повинно бути унікальним, як і поле **id**, тому встановимо йому властивість **Unique** і задамо кількість символів **255**.

- Наступні поля: **password**, що містить пароль, **e_mail** містить адресу електронної пошти і поле **type**, що містить тип користувача будуть без особливих властивостей, зі строковим типом **VARCHAR** завдовжки **255** символів, за винятком останнього поля **type**, якому вистачить **45** символів.

Після виконаних маніпуляцій форма з ім'ям таблиці **users** буде виглядати так:



На діаграмі з'явиться таблиця **users** с полями і індексами:



Аналогічним способом створимо таблицю **settings** з налаштуваннями доступу до бази даних ІС, що містить поля **id**, **host** для вказівки імені хоста (адреси сервера), **db** - імені бази даних, **user** і **password** з ім'ям користувача і паролем для установки ІС на віддалений сервер.

Далі за вже відомою процедурою створимо таблицю **shops**, яка буде зберігати дані по магазинах в полях: **id** типу **integer** – ключове, нульове, унікальне з автоінкрементом; поле **name**, що зберігає назву магазину; поле **address** – його фізичну адресу, поле **tel** – телефон магазину, **site** – інтернет сайт магазину і поле **email** з електронною адресою магазину.

Потім створимо таблицю **products**, що зберігає дані про продукцію магазину в полях: **id** типу **integer** – ключове, нульове, унікальне з автоінкрементом, поле **name** зберігає назву магазину, ключове, нульове поле цілочисельного типу **shop_id**, що зберігає номер магазину, поле **type_id** з інформацією про номер товару з таблиці видів продукції. Поле **brand** - бренд виробника завдовжки 255 символів, поле **model** з моделлю товару, поле **data** - з даними і характеристиками товару типу **Tinytext**, поле **img** з повною адресою до зображення товару завдовжки 255 символів, і поля **price** з ціною товару, поле **warranty** з інформацією про терміни гарантії на товар завдовжки 45 символів.

Створені нами таблиці **settings**, **shops** і **products** виглядають наступним чином:

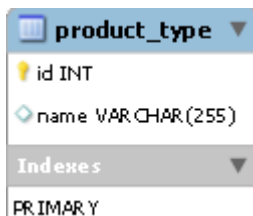
Table Name	Field Name	Field Type
settings	id	INT
	host	VARCHAR(45)
	db	VARCHAR(45)
	user	VARCHAR(45)
	password	VARCHAR(45)
Shops	id	INT
	name	VARCHAR(255)
	address	VARCHAR(255)
	tel	VARCHAR(100)
	site	VARCHAR(100)
	email	VARCHAR(100)
Products	id	INT
	shop_id	INT
	type_id	INT
	brand	VARCHAR(255)
	model	VARCHAR(255)
	data	TINYTEXT
	img	VARCHAR(255)
	price	VARCHAR(45)
warranty	VARCHAR(45)	

Indexes:

Table Name	Index Name	Index Type
settings	id_UNIQUE	PRIMARY
Shops	id_UNIQUE	PRIMARY
Products		PRIMARY

Далі нам знадобиться таблиця, яка зберігає тип продуктів **product_type**, вона складається з унікального, ненульового, ключового поля **id** з автоінкрементом цілочисленного типу, і унікального поля **name** завдовжки 255 символів, яке містить назву виду продуктів.

Вид таблиці такий:

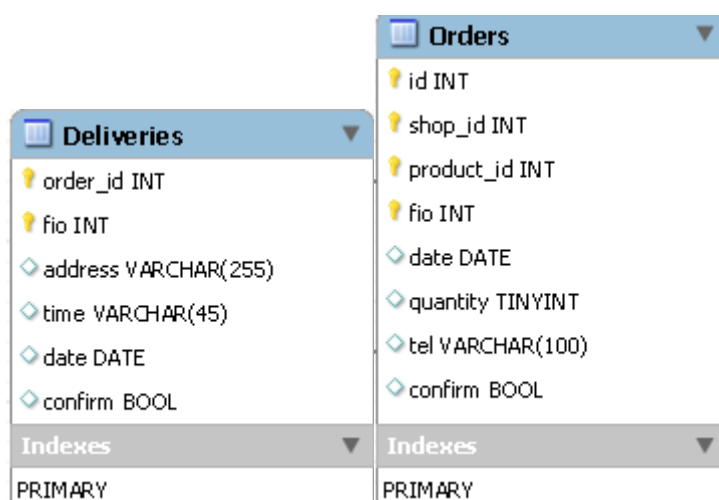


Останні дві таблиці це **orders** і **deliveries**, перша містить інформацію про замовлення клієнтів, а остання – дані про доставку продукції.

Поля таблиці **orders**: **id** – ключове, нульове, унікальне поле целочисленного типу з автоінкрементом, поле **shop_id**, що містить номер магазину – ключове, нульове целочисленного типу, поле **product_id**, що зберігає номер продукту – ключове, нульове целочисленного типу, поле **fio date** з датою замовлення – типу **DATE**, поле **quantity** з кількістю замовлених товарів - целочисленного типу, поле **tel** з номером телефону замовника - строкового типу довжиною 255 символів і поле **confirm** містить інформацію про підтвердження замовлення - логічного типу.

Поля таблиці **deliveries**: **order_id** з номером замовлення - ключове, нульове, унікальне поле целочисленного типу з автоінкрементом, поле **fio** з номером користувача, який зробив замовлення - ключове, нульове целочисленного типу, поле **address**, що зберігає адресу доставки товару, зазначену клієнтом - строкового типу довжиною 255 символів, поле **time** зберігає бажаний час доставки товару - строкового типу довжиною 255 символів, поле **date** з датою здійснення замовлення клієнтом - типу **DATE** і поле логічного типу **confirm**, що зберігає інформацію про доставку товару.

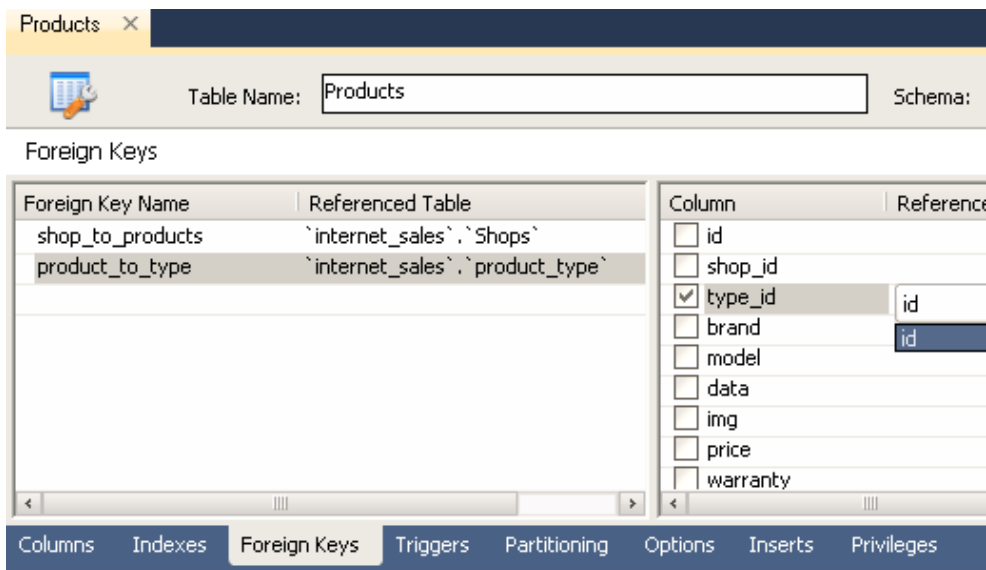
Таблиці **orders** і **deliveries** виглядають наступним чином:



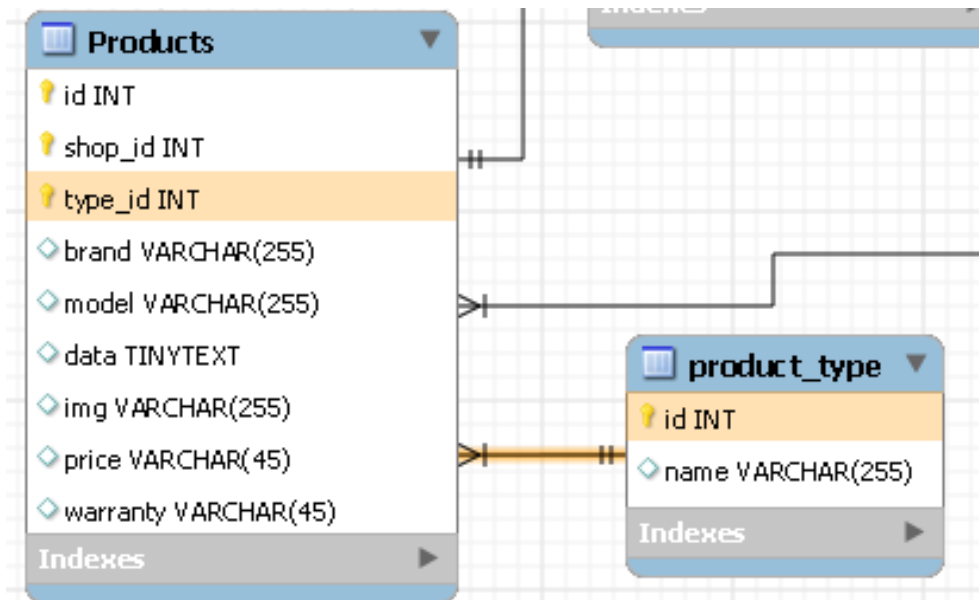
Зв'язки таблиць

Ми створили базу даних, що складається з семи таблиць, тепер необхідно зв'язати таблиці, ми вже практично розробили основні поля цілочисельного типу, вони і стануть основою для зв'язування.

Для того щоб зв'язати дві таблиці наприклад **products** і **product_type**, необхідно двічі клацнути лівою кнопкою миші на діаграму з таблицею products і вибрати вкладку **Foreign keys** (зовнішні ключі). Далі в поле **Foreign key name** ввести унікальне ім'я зовнішнього ключа, двічі клацнути по вкладці **Referenced table** і вибрати таблицю **product_type**. Потім у формі розташованій праворуч вибрати поле **type_id** і вибрати в спливаючому списку поле **id**.



Таким чином, обидва поля таблиці виявляються пов'язані, потім потрібно задати тип відносин зв'язку між таблицями, відкриємо вікно, клікнувши на інформацію, що з'явилася, і виберемо вкладку **Foreign Key**. В розділі **Cardinality** виберемо тип зв'язку один до багатьох, і закриємо вікно. На діаграмі відобразиться зв'язок таблиць:

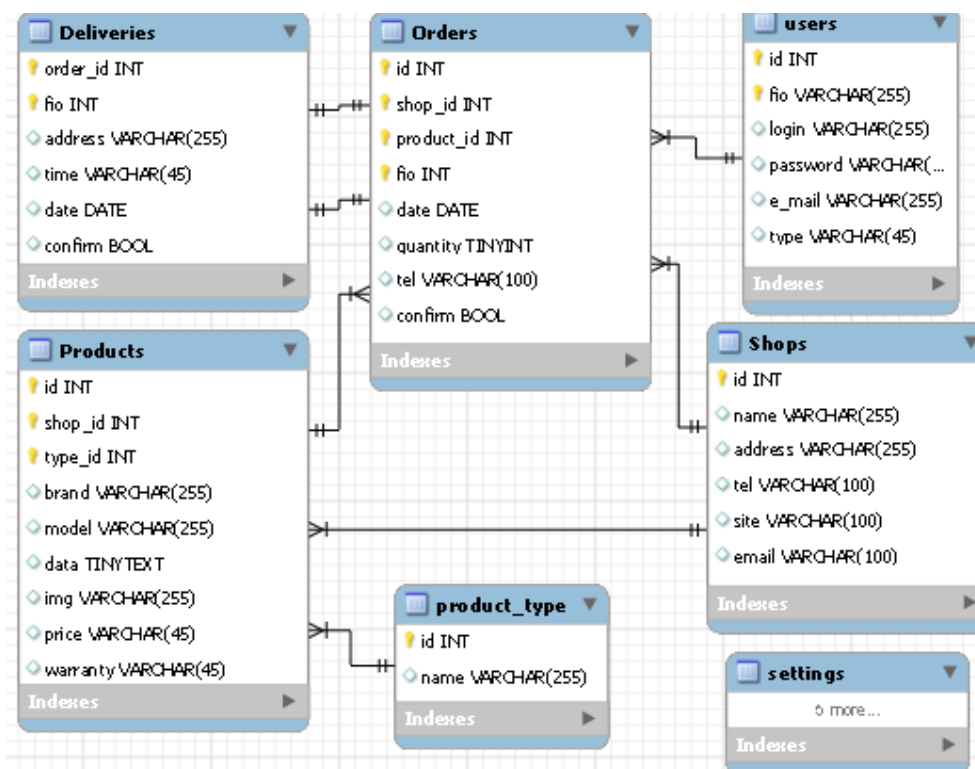


Аналогічним чином пов'яжемо всі ключові поля в таблицях, щоб вони були логічно пов'язані між собою, далі нам необхідно переконатися, що спроектована база даних відповідає третій нормальній формі.

Нормальна форма – якість зв'язку в реляційній моделі даних, що характеризує його з точки зору надмірності, яка потенційно може привести до логічно помилкових результатів вибірки або зміни даних. Нормальна форма визначається як сукупність вимог, яким має задовольняти відношення.

У реляційної моделі відношення завжди знаходиться в першій нормальній формі за визначенням поняття відношення. Що ж стосується різних таблиць, то вони можуть не бути правильними уявленнями відносин і, відповідно, можуть не перебувати в першій нормальній формі. Відносини знаходиться в другій нормальній формі тоді і тільки тоді, коли вони знаходиться в першій нормальній формі і кожен не ключовий атрибут функціонально повно залежить від його потенційного ключа. База даних буде знаходитися в третій нормальній формі, якщо вона приведена до другої нормальної форми і кожен не ключовий стовпець незалежний один від одного.

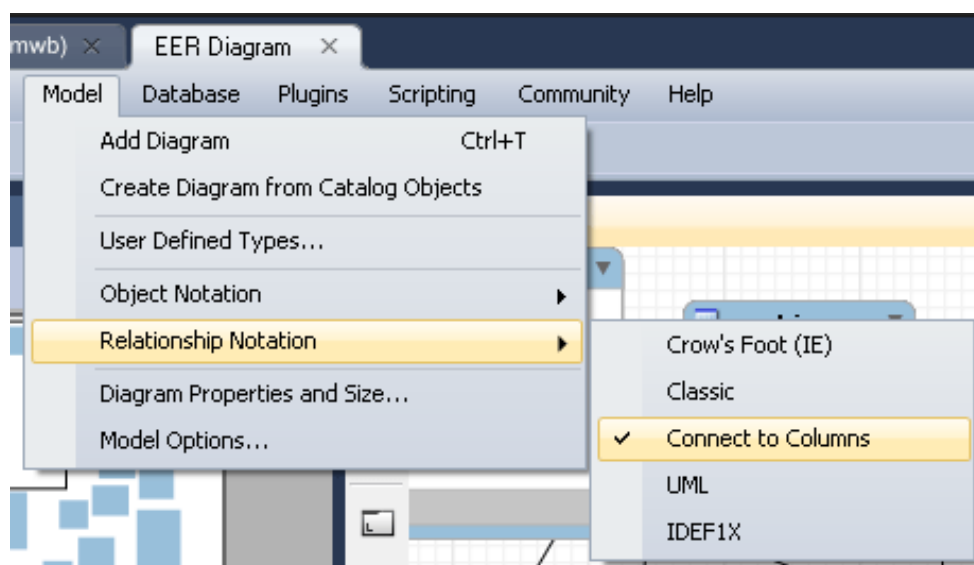
Таким чином, наша база знаходиться в третій нормальній формі, тому що кожен не ключовий стовпець незалежний один від одного. Це наочно видно на діаграмі нашої бази даних:



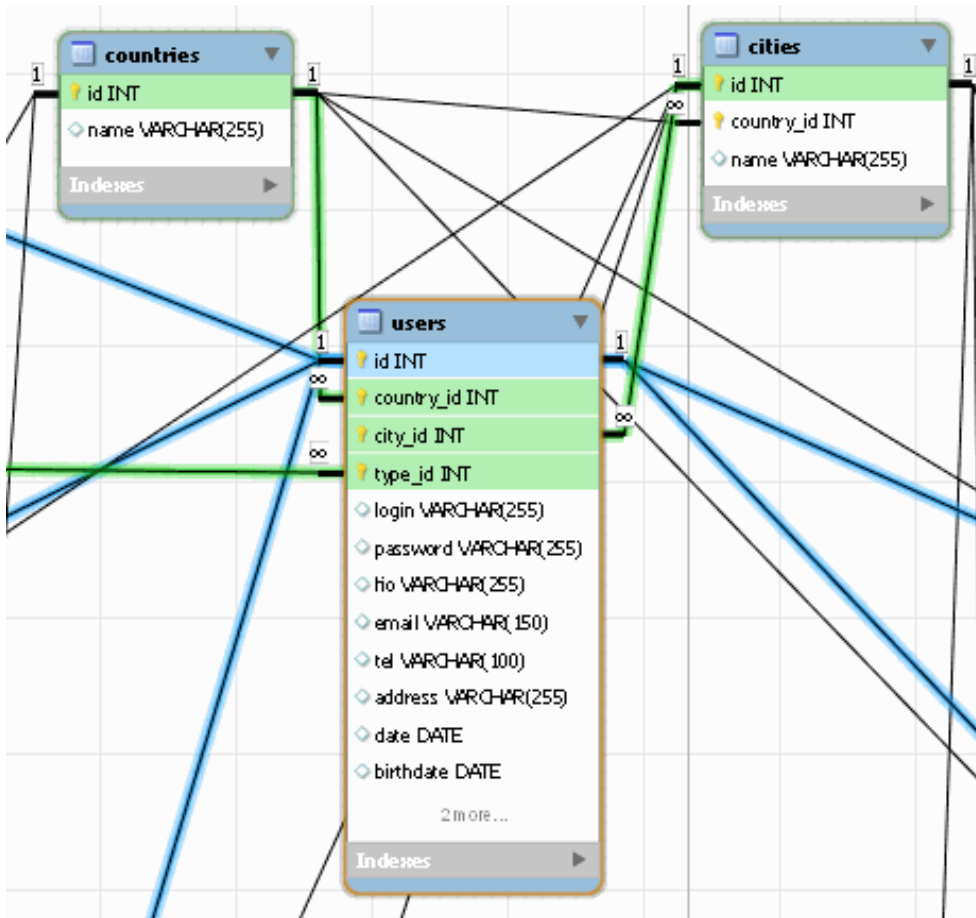
Більшість таблиц знаходяться у відношенні один-до-багатьох, за винятком таблиць **deliveries** і **orders** що у відношенні один-до-одного, тому що доставлений товар може бути тільки в одному замовленні, тобто у одного замовлення тільки одна доставка. Решта зв'язків наочно продемонстровані вище.

Відображення ліній зв'язків таблиць в режимі поле-до-поля

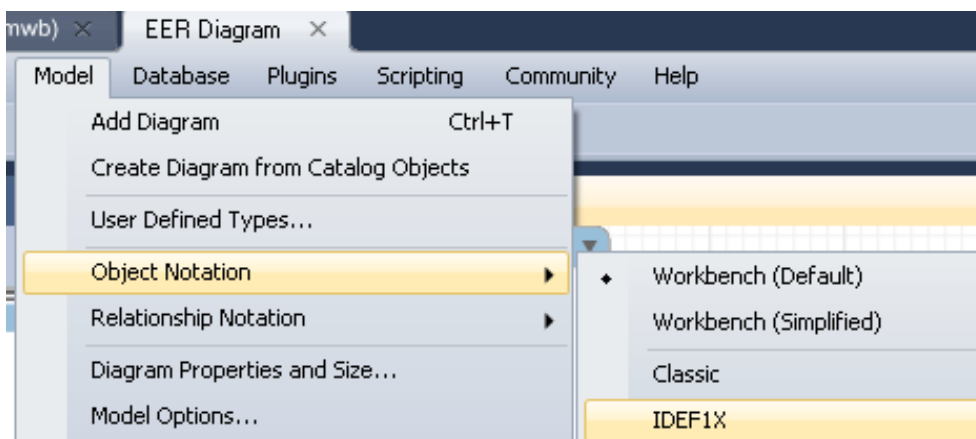
Далі наведено пояснення про те, як змінити вид зв'язків і таблиць, для цього необхідно вибрати наступний параметр в розділі меню **Relationship Notation**:



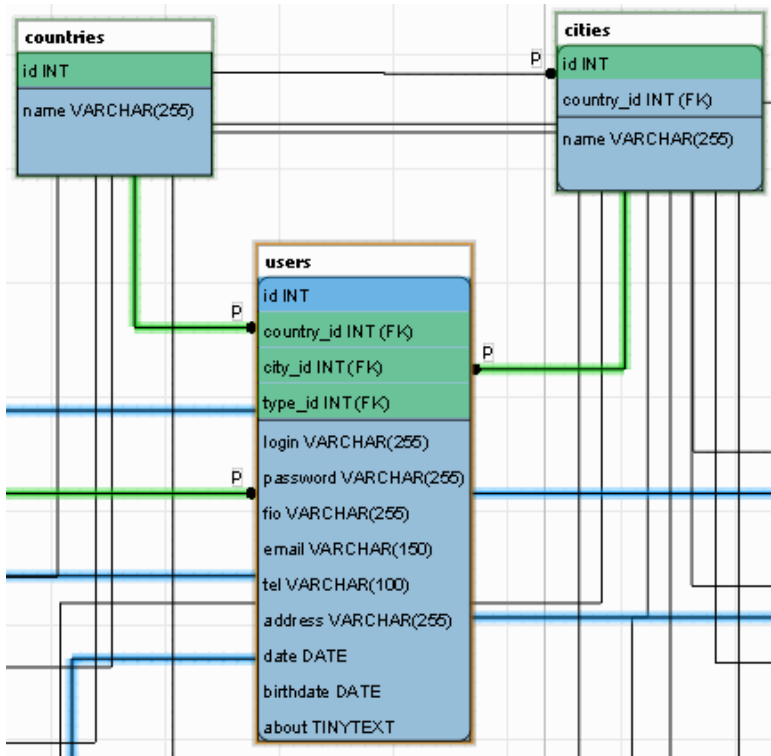
Після цього зв'язки таблиць приймуть вигляд:



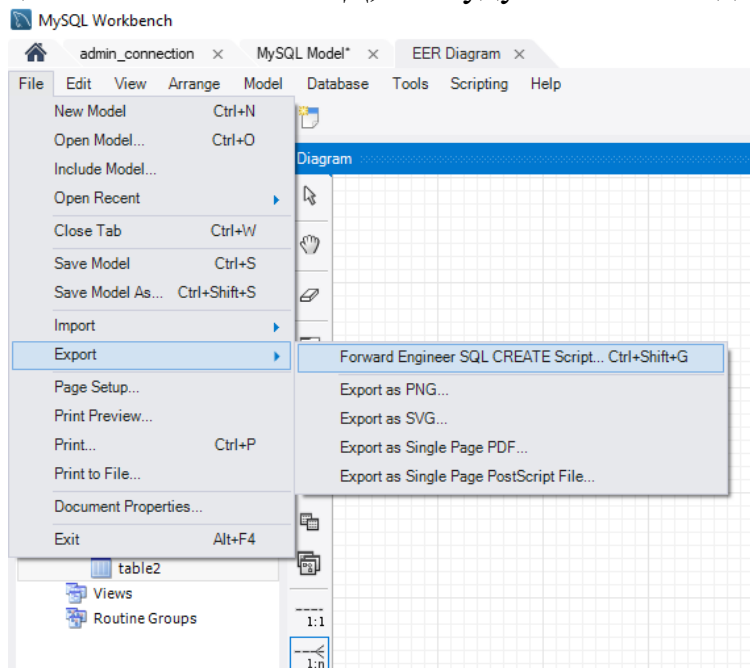
Також є можливість змінити вид таблиць, для цього потрібно поставити галочку у вищевказаному розділі меню і обрати опцію **Object Notation**:

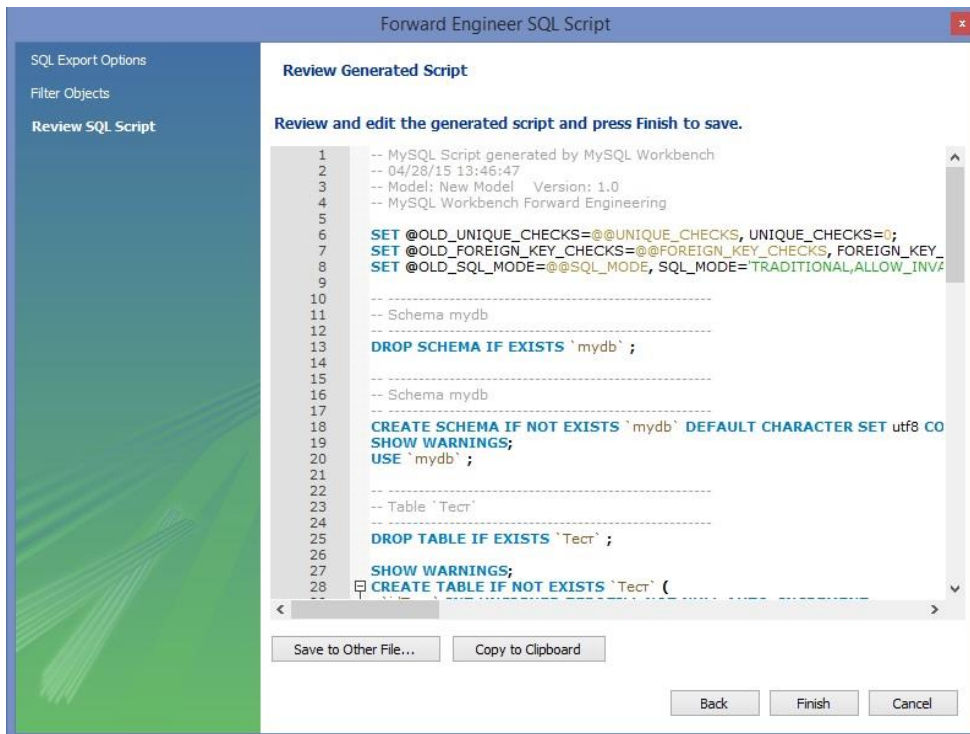
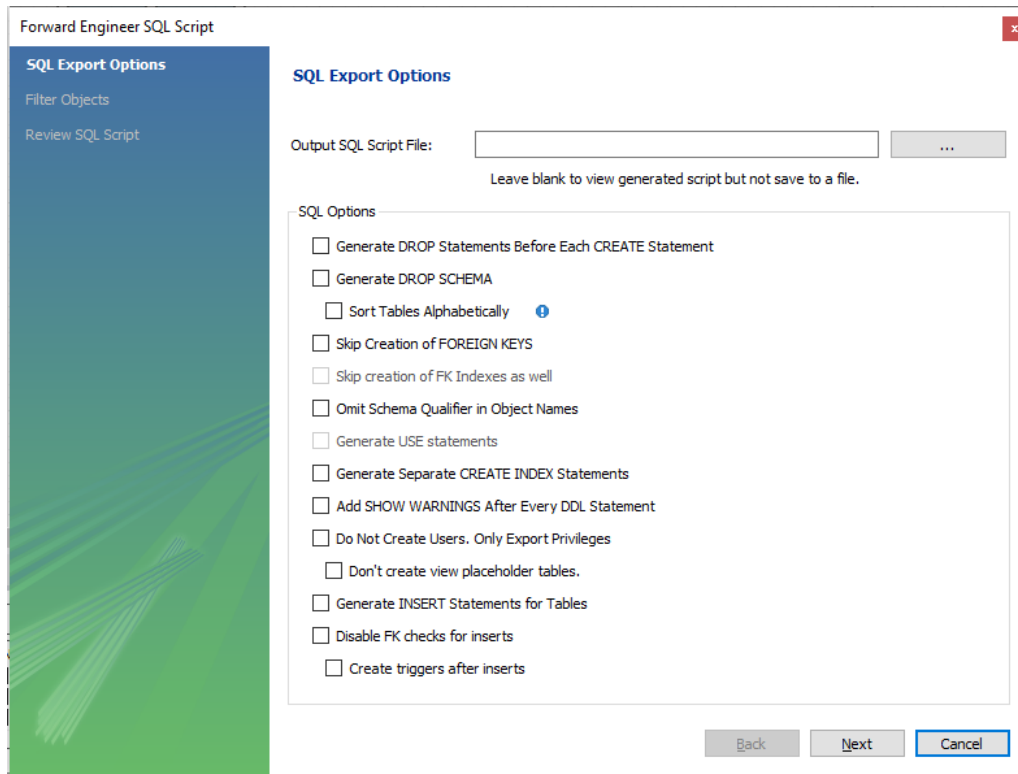


Ось так виглядає таблиця на діаграмі, наведена до стандарту IDEF1X:



Коли Ви закінчите створення структури, ви можете створити базу даних через набір SQL-команд шляхом експорту створеної схеми. Щоб зробити це, виберіть меню Файл -> Експорт -> і виберіть потрібний варіант, дані в основному являють собою таблиці, і користувачів (якщо такі створено). На кшталт того, як показано нижче. В процесі експорту можна задати саме ті таблиці та інші елементи БД, які будуть включені до файлу, що експортується.





Згенерований скрипт можна скопіювати до буфера обміну та вставити потім до нового запиту і згенерувати відповідні створені таблиці і зв'язки між ними.

ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

Завдання 1. На основі логічної моделі БД визначити типи полів та атрибути для збереження інформації. Додати зовнішні ключі. Створити зв'язки між таблицями.

Завдання №2. Експортувати створену схему бази даних у вигляді графічного або pdf-файлу.

Завдання №3. Експортувати схему бази даних у вигляді sql-скрипту, запустити його на виконання, передивитись чи коректно створено таблиці і зв'язки.

Завдання №4. Переглянути створену діаграму у різних поданнях за допомогою опції **Object Notation**. Зробити висновки, яке подання є найзручнішим для певних цілей.

Завдання №5. Відкрити базу даних sakila та передивитись EER-діаграму для неї, внести пропозиції стосовно її вдосконалення.

Лабораторна робота №6

Тема: Оцінка ефективності запиту

Мета: Навчитись проводити аналіз можливостей оптимізації запитів.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Індекси – шлях до оптимізації продуктивності MySQL. Індекс в MySQL - це алгоритм впорядкування даних в таблиці за певним полем, щоб прискорити пошук. Індекси допомагають оптимізувати запити на отримання даних, але уповільнюють додавання та зміну записів. Ви можете створювати та видаляти індекси без впливу на самі дані.

Деякі різновиди індексів:

- 1) Первинний ключ.
- 2) В-дерево.
- 3) Унікальний індекс (коли поле є унікальним, подальший пошук після першого знайденого співпадіння не відбувається, проте, коли дані наприкінці таблиці, це все одно буде перегляд до кінця таблиці).
- 4) Хеш-індекс (використовується для текстового пошуку, де неможливо виконати пошук В-tree. Наприклад, MD5, який є одним з видів хеш-функцій – це алгоритм, який перетворює будь-які дані на короткий і унікальний рядок символів, тобто завжди певна довжина. Хеші MD5, SHA-1 та SHA-256 використовуються для перевірки цілісності та автентичності завантажених даних. Можна порівняти хеш файлу з очікуваним хешем, щоб переконатися, що вони співпадають.)
- 5) Full text index (повнотекстовий індекс, на кшталт індексу до книги, з індикацією сторінок, де про що йдеться мова. Використовується при необхідності пошуку по словах, тобто для пришвидшення такого пошуку, але існують і зовнішні рішення для повнотекстового пошуку, окрім того, що реалізовано в MySQL.)

Пришвидшуючи пошук, ідекси:

1) Збільшують обсяг даних.

2) Збільшують час, необхідний для виконання операцій INSERT та UPDATE (адже виконується перебудова структури даних для зберігання індексу).

На маленьких таблицях навіть коли індекс створено, він у 99% не використовується, адже СУБД швидше виконати повний перегляд, ніж будувати структури даних під індекс.

Порядок виконання роботи

1) Передивляємось план виконання запитів за допомогою інструкції EXPLAIN. Спочатку виконуємо запит по таблиці без індексу, коли окрім первинного ключа у таблиці немає індексів.

Column Name	Datatype	PK	NN	UQ	BIN	U
emp_no	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
birth_date	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
first_name	VARCHAR(14)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
last_name	VARCHAR(16)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
gender	ENUM('M', 'F')	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
hire_date	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Структура таблиці

Index Name	Type	Column	#	Order	Length
PRIMARY	PRIMARY	<input checked="" type="checkbox"/> emp_no	1	ASC	0
		<input type="checkbox"/> birth_date		ASC	0
		<input type="checkbox"/> first_name		ASC	0
		<input type="checkbox"/> last_name		ASC	0
		<input type="checkbox"/> gender		ASC	0
		<input type="checkbox"/> hire_date		ASC	0

Структура індексів

The screenshot displays the Oracle SQL Developer interface. At the top, a SQL query is entered in the editor:

```

1 • SELECT
2 *
3 FROM
4 employees.employees
5 WHERE
6 first_name LIKE 'G%';
  
```

A tooltip above the cursor indicates: "Execute the EXPLAIN command on the statement under the cursor". Below the editor is the "Result Grid" showing 16 rows of employee data:

#	emp_no	birth_date	first_name	last_name	gender	hire_date
1	10001	1953-09-02	Georgi	Facello	M	1986-06-26
2	10015	1959-08-19	Guoxiang	Nooteboom	M	1987-07-02
3	10055	1956-06-06	Georgy	Dredge	M	1992-04-27
4	10063	1952-08-06	Gino	Leonhardt	F	1989-04-08
5	10075	1960-03-09	Gao	Dolinsky	F	1987-03-19
6	10121	1962-07-14	Guoxiang	Ramsay	M	1989-05-03
7	10124	1962-05-23	Geraldo	Marwedel	M	1991-09-05
8	10133	1963-12-12	Giri	Isaak	M	1985-12-15
9	10202	1956-01-05	Greger	Lichtner	M	1991-10-06
10	10207	1955-05-28	Girolamo	Anandan	F	1992-10-11
11	10219	1952-05-02	Genta	Kolvik	M	1993-03-31
12	10321	1955-12-12	Guenter	Ravishankar	M	1991-09-24
13	10336	1961-07-16	Goa	Rothe	F	1992-01-07
14	10399	1957-06-13	Guenter	Marchegay	F	1985-05-07
15	10422	1954-06-02	Greger	Rubsam	F	1990-12-07
16	10458	1961-01-06	Gina	Engelmann	F	1988-07-23

Below the result grid, the "Action Output" shows the execution of the query:

#	Time	Action	Message	Duration / Fetch
7	16:44:34	SELECT * FROM employees.employees WHERE first_name...	1000 row(s) returned	0,0034 sec / 0,0067 sec

The "Tabular Explain" section shows the execution plan:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	employees		ALL					299733	11.11	Using where

Інтерпретуємо отримані результати

2) Додаємо індекс та виконуємо ті ж самі операції, дивимось на вплив індексу на показники у стовпцях filtered та rows.

Стовпець filtered вказує на оцінку відсотка рядків таблиці, які будуть відфільтровані за умовою таблиці. Тобто, rows показує оцінку кількості рядків, які будуть перевірені, а $rows \times filtered / 100$ показує кількість рядків, які будуть об'єднані з попередніми таблицями. Якщо значення у стовпці filtered дуже низьке, це може означати, що умова таблиці не дуже точна або потребує індексування.

ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

Завдання №1. Проаналізувати план виконання запитів для таблиці з індексом та без, порівняти результати.

Завдання №2. Для однієї таблиці спочатку створити 2 окремі індекси по 2 полям, проаналізувати як працює запит. Створити один індекс із двох

проіндексованих полів, зробити висновки. У висновках вказати, чи використовує СУБД створені індекси під час виконання запитів, який саме індекс використовується, а який ні та чому на Вашу думку відбувається саме так.

Завдання №3. Проаналізувати вже створені індекси у таблиці на вибір із БД sakila, запропонувати шляхи оптимізації або обґрунтувати доцільність саме такої структури індексів.

Завдання №4. Реалізувати приклад повнотекстового пошуку із застосуванням відповідного індексу на таблиці film БД sakila.

Завдання №5. Створити згенерований стовпець (GENERATED) із довільним розрахунковим значенням та проаналізувати доцільність додавання до нього індексу. Зробити висновки.

Лабораторна робота №7

Тема: Міграція та резервування бази даних

Мета: Навчитись створювати та розгортати дамп (копію) бази даних MySQL, переносити та зберігати БД, завантажувати базу даних на сервер, експортувати створену базу даних.

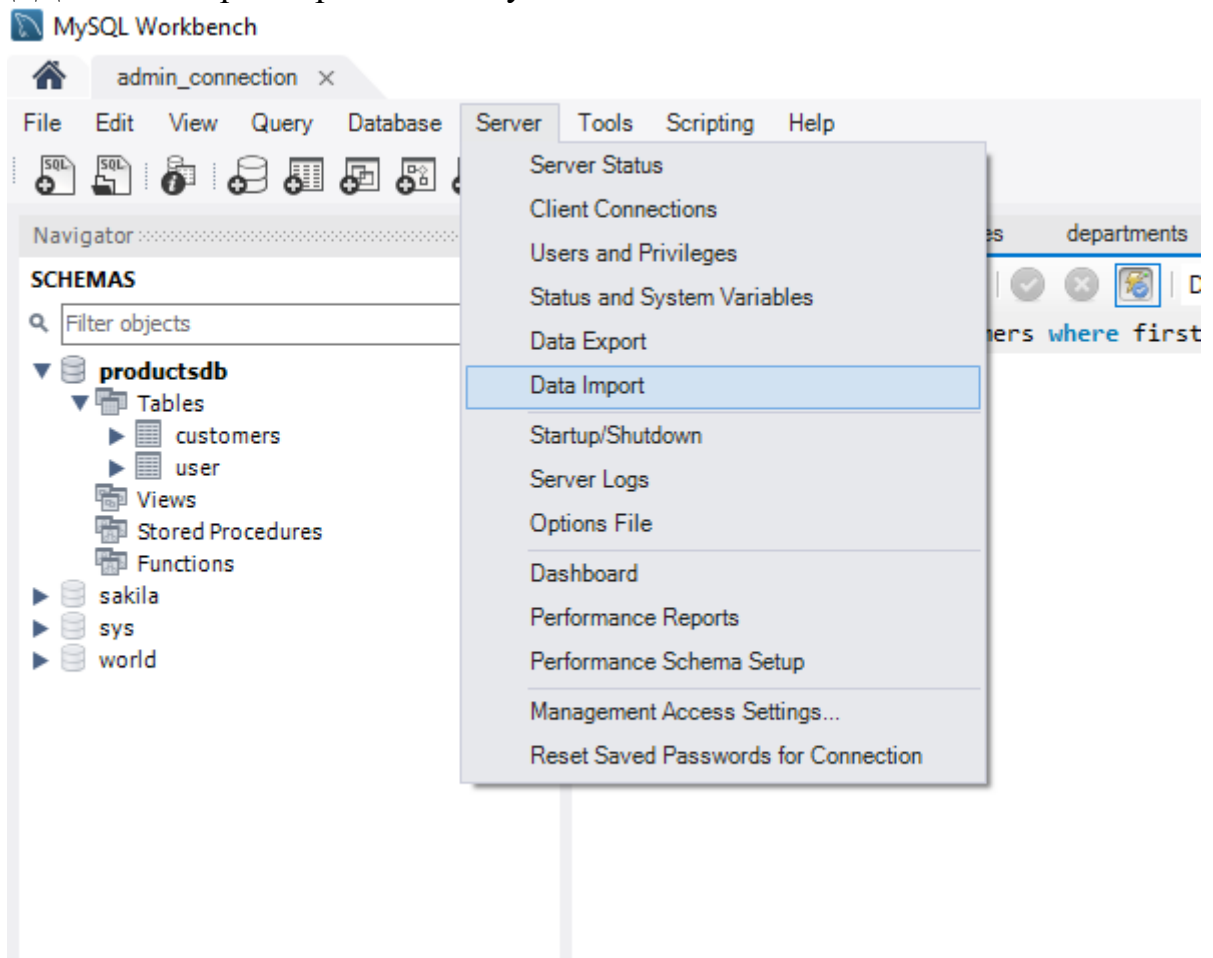
ТЕОРЕТИЧНІ ВІДОМОСТІ

MySQL Workbench – це графічний інструмент для роботи з базами даних MySQL, який має функцію міграції даних з інших систем керування базами даних. Ви можете використовувати майстер міграції MySQL Workbench, щоб налаштувати, виконати та запланувати процес перенесення даних із вихідного джерела даних (source) до місця призначення (target). Ви також можете створити резервну копію своєї бази даних MySQL за допомогою MySQL Workbench, використовуючи функцію експорту та імпорту даних.

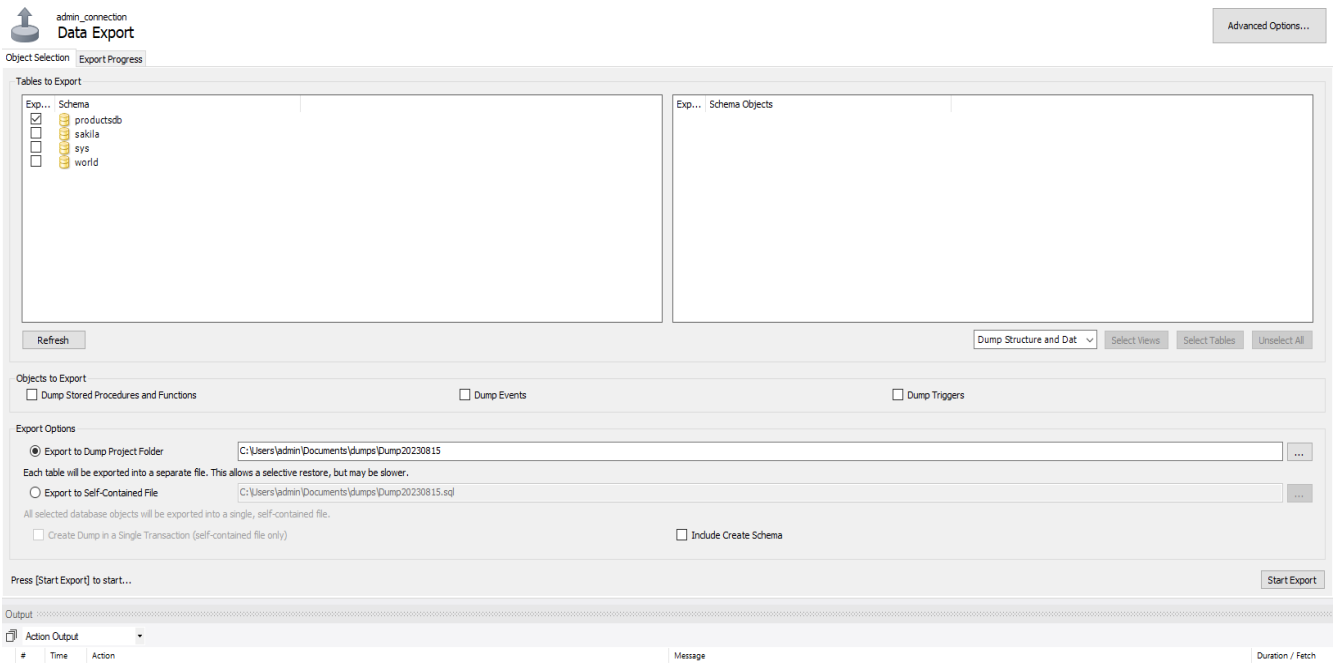
Можна імпортувати та експортувати як окремо тільки структуру БД, так і структуру разом із даними.

Порядок виконання роботи

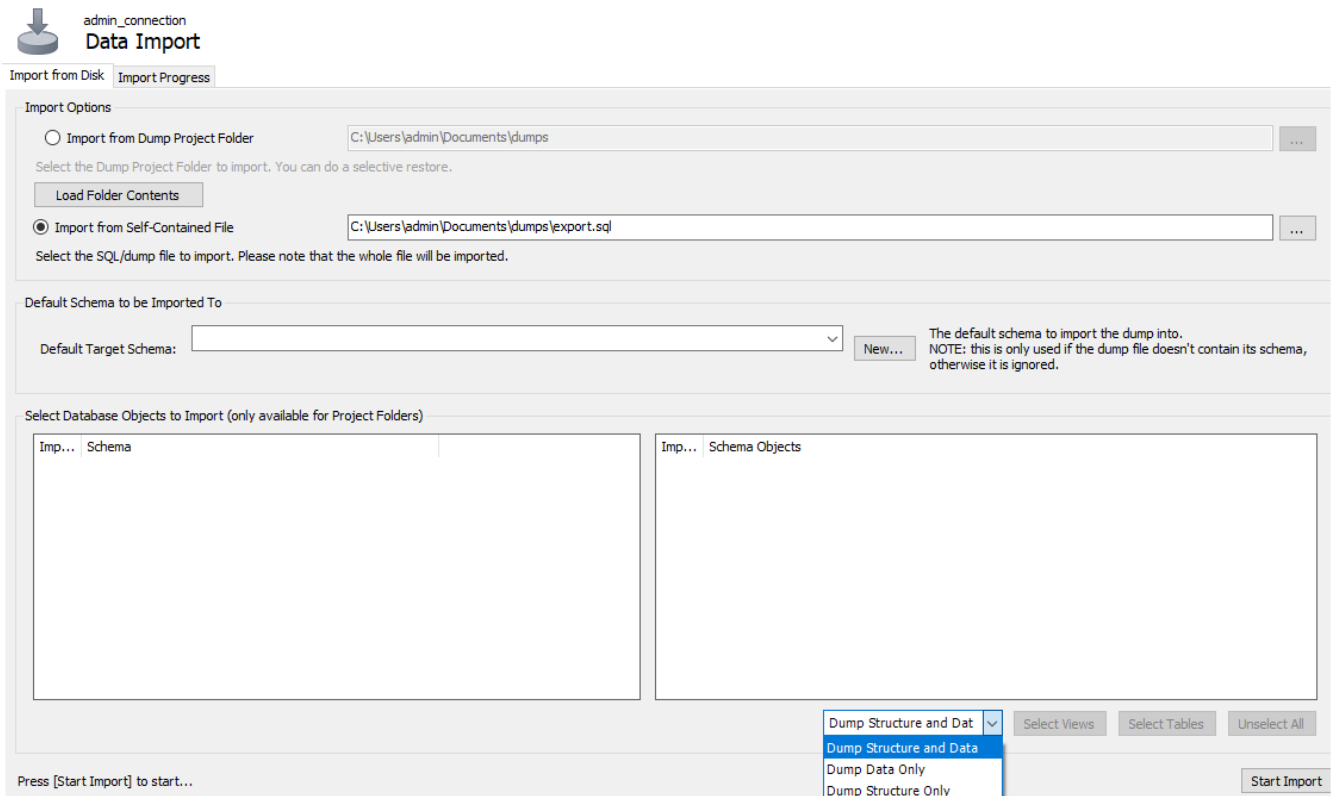
Екпортуємо створену структуру БД разом із даними, потім імпортуємо її у нову БД. Для цього треба зробити наступне.



Команди експорту-імпорту



Вікно експорту БД в MySQL Workbench



Діалогове вікно імпорту БД

ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

Завдання №1. Зробити експорт створеної бази даних разом із її структурою та даними у файл з розширенням sql.

Завдання №2. Імпортувати sql-файл до бази даних з новим ім'ям.

Лабораторна робота №8

Тема: Адміністрування бази даних

Мета: Навчитись створювати користувачів та надавати їм привілеї.

ТЕОРЕТИЧНІ ВІДОМОСТІ

За замовчуванням у MySQL буде лише один користувач — root. Це адміністратор з доступом до всіх баз даних та таблиць усередині цих баз.

Розглянемо, як створити нового користувача MySQL та надати йому права доступу за допомогою командного рядка.

Це зручно, якщо потрібно надати доступ до конкретної таблиці іншій людині, але з обмеженими правами. Наприклад, щоб вона могла її редагувати, але не могла видалити.

Порядок виконання роботи

1. Підключитися до сервера та зайти до MySQL
2. Створити користувача MySQL
3. Налаштувати права доступу
4. Забрати права доступу
5. Видалити користувача
6. Вийти з MySQL

Підключитися до сервера та зайти до MySQL

Спочатку підключіться до сервера SSH. Зазвичай хостинг-провайдери надсилають логін та пароль для підключення у листі про активацію хостингу. У нас цей лист називається по-різному залежно від типу хостингу: «Ваш віртуальний хостинг готовий до використання», «Це ваш VPS готовий?» або «Виділений сервер замовляли?».

Після підключення до сервера авторизуйтеся як root користувач MySQL за допомогою команди:

```
mysql -u root -p
```

Створити користувача MySQL

Припустимо, у раніше створеній базі даних є таблиця зі списком товарів для книгарні. Нам необхідно передати управління таблицею іншій людині. Тому потрібно створити нового користувача MySQL. Використовуйте для цього команду CREATE USER з таким синтаксисом:

```
CREATE USER "ім'я_користувача"@хост IDENTIFIED BY "пароль";
```

Параметр «хост» у команді означає пристрій, з якого людина зможе підключатися до сервера MySQL. Тут можна використовувати кілька варіантів:

- **localhost.** Означає, що доступ до MySQL з'явиться у користувача лише після того, як він підключиться до сервера;
- **IP-адреса пристрою в мережі,** з якого людина буде підключатися до MySQL;

- **%**. Означає, що користувач матиме доступ до MySQL з будь-якого пристрою.

В останніх двох варіантах додатково підключатися до сервера не потрібно, але доведеться спочатку дозволити віддалені підключення в налаштуваннях MySQL.

Якщо ви не знаєте, який хост вказати, використовуйте localhost. Це найбезпечніший варіант. Доведеться додатково підключатися до сервера, зате зловмисникам буде складніше отримати доступ до бази даних.

Тепер підставимо до команди реальні дані. Не забудьте, що всі команди SQL повинні закінчуватись крапкою з комою.

```
CREATE USER "bookstore_administrator"@"localhost" IDENTIFIED BY "Hgy467Hkgd";
```

Налаштувати права доступу

Користувача створили, але він нічого не може зробити з базою даних, доки ви не призначите йому привілеї. Для цього використовуйте команду GRANT з таким синтаксисом:

```
GRANT ПРАВО, ПРАВО ON база_даних.таблиця TO "ім'я_користувача"@"хост";
```

Право — це, наприклад, можливість редагувати базу даних або створювати нового користувача. Якщо потрібно надати відразу кілька прав, пропишіть їх у команді через кому. Ось список прав, які часто використовуються:

- **ALL** — дати всі права до бази даних, за винятком GRANT OPTION. Якщо ви не вкажете назву конкретної бази даних, користувач отримає повний доступ до всього сервера MySQL;
- **CREATE** — право створювати нові бази даних та таблиці;
- **DELETE** — право видаляти рядки з таблиці;
- **DROP** — право видаляти бази даних або таблиці;
- **GRANT OPTION** — право призначати чи відбирати права. Але вдасться дати або відібрати тільки ті права, якими володіє користувач, який використовує команду;
- **INSERT** — право створювати рядки у таблиці;
- **SELECT** — право переглядати рядки в таблицях;
- **UPDATE** — право змінювати зміст рядків у таблицях.

Тепер давайте надамо нашому користувачеві *bookstore_administrator* право на створення таблиць у базі даних *bookstore*. А також дозволимо додавати записи до цих таблиць:

```
GRANT CREATE, INSERT ON bookstore.* TO "bookstore_administrator"@"localhost";
```

Зірочка у нашому прикладі означає «всі таблиці всередині бази даних bookstore». Ви також можете замінити ім'я бази даних на зірочку, щоб надати користувачеві права для всіх баз даних на сервері.

Коли закінчите надавати привілеї, перезавантажте їх командою FLUSH PRIVILEGES, щоб вони набули чинності:

```
FLUSH PRIVILEGES;
```

Тепер подивимося, які доступи є у *bookstore_administrator* :

```
SHOW GRANTS FOR "bookstore_administrator"@"localhost";
```

Результат буде таким:

```
MariaDB [(none)]> SHOW GRANTS FOR "bookstore_administrator"  
+-----  
| Grants for bookstore_administrator@localhost  
+-----  
| GRANT USAGE ON *.* TO 'bookstore_administrator'@'localhost'  
| GRANT INSERT, CREATE ON `bookstore`.* TO 'bookstore_administrator'@'localhost'  
+-----  
2 rows in set (0.01 sec)
```

Якщо через деякий час вашому користувачеві знадобляться додаткові привілеї, додайте їх за допомогою тієї ж команди. Припустимо, ми хочемо дати користувачеві *bookstore_administrator* усі права.

```
GRANT ALL ON bookstore.* TO "bookstore_administrator"@"localhost";
```

Перезавантажуємо привілеї командою `FLUSH PRIVILEGES` та перевіряємо командою `SHOW GRANTS`:

```
MariaDB [(none)]> SHOW GRANTS FOR "bookstore_administrator"  
+-----  
| Grants for bookstore_administrator@localhost  
+-----  
| GRANT USAGE ON *.* TO 'bookstore_administrator'@'localhost'  
| GRANT ALL PRIVILEGES ON `bookstore`.* TO 'bookstore_administrator'@'localhost'  
+-----  
2 rows in set (0.00 sec)
```

Забрати права доступу

Якщо привілеї потрібно навпаки забрати, запустіть команду `REVOKE`:

```
REVOKE ПРАВО ON база_даних.таблиця FROM "ім'я_користувача"@хост";
```

Допустимо, нам потрібно забрати у користувача *bookstore_administrator* право видаляти таблиці в базі даних *bookstore*. Тоді команда буде такою:

```
REVOKE DROP ON bookstore.* FROM "bookstore_administrator"@"localhost";
```

Не забудьте знову перезавантажити привілеї командою `FLUSH PRIVILEGES`.

Видалити користувача MySQL

Щоб видалити користувача MySQL, використовуйте команду `DROP USER`:

```
DROP USER "ім'я_користувача"@хост";
```

Ця команда не вимагає додаткового підтвердження. Користувач бази даних видаляється з першого разу і скасувати це не можна, тому використовуйте команду обережно.

Вийти з MySQL

Наприкінці роботи відключіться від MySQL-сервера за допомогою команди:

ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

Завдання №1. Створити користувача MySQL через консольний клієнт

Завдання №2. Налаштувати права доступу до БД через консольний клієнт

Завдання №3. Забрати права доступу через консольний клієнт

Завдання №4. Видалити користувача MySQL через консольний клієнт

Завдання №5. Виконати ті ж самі операції через графічний інтерфейс

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Добролюбова М. В. Програмування баз даних: конспект лекцій : навч. посіб. Київ : КПІ ім. Ігоря Сікорського, 2021. 275 с.
2. Вербіцький В. В. Бази даних та інформаційні системи : метод. вказівки. Одеса : Одес. нац. ун-т ім. І.І. Мечникова, 2022. 82 с.
3. Качурівський В. О. Конструювання бази даних : метод. рекомендації. Березани : Березанський агротехнічний інститут, 2021. 36 с.
4. Каштан В.Ю., Іванов Д.В. Конспект лекцій з дисципліни “Бази даних в інформаційних системах”. Частина 1. Дніпро : НТУ “ДП”, 2020. 58 с.
5. Coronel C., Morris S. Database Systems: Design, Implementation, & Management. 13th Edition. Boston : Cengage Learning, 2018. – 800 p.
6. Анісімов А. В., Кулябко П. П. Інформаційні системи та бази даних: Навчальний посібник для студентів факультету комп’ютерних наук та кібернетики. Київ, 2017. 110 с.
7. Лавренчук С. В., Коцюба А. Ю. Бази даних та мова SQL: конспект лекцій. Луцьк : Луцький НТУ, 2016. 76 с.
8. Мулеса О. Ю. Інформаційні системи та реляційні бази даних. Навч. посібник. Електронне видання, 2018. 118 с.
9. Харів Н. О. Бази даних та інформаційні системи : навч. посіб. Рівне : НУВГП, 2018. 127 с.
10. SQL Підручник. W3schoolsUA. Українською. [Електронний ресурс] Режим доступу: <https://w3schoolsua.github.io/sql/index.html>

Навчальне видання

БАЗИ ДАНИХ ТА ІНФОРМАЦІЙНІ СИСТЕМИ

Методичні вказівки до виконання практичних робіт

Укладач:
Міхнова Олена Дмитрівна

Формат 60x84/16. Гарнітура Times New
Roman Папір для цифрового друку.
Друк ризографічний.
Ум. друк. арк. 1,5.
Наклад пр.
Державний біотехнологічний
університет 61002, м. Харків, вул.
Алчевських, 44