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

ОРГАНІЗАЦІЯ БАЗ ДАНИХ

Методичні вказівки
до виконання практичних робіт

для здобувачів бакалаврського рівня вищої освіти
за спеціальністю 123 «Комп'ютерна інженерія»

Затверджено
рішенням Науково-методичної ради
ННІ «Кіберпорт»
Протокол №__ від __.__.2023 р.

Харків
2023

УДК 005.931

Схвалено на засіданні кафедри
кібернетики та інформаційних технологій
Протокол № 12 від 31.05.2023 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».

Організація баз даних : методичні вказівки до виконання практичних робіт для здобувачів бакалаврського рівня вищої освіти за спеціальністю 123 «Комп'ютерна інженерія» / О.Д. Міхнова / – Електрон. дані. – Х. : ДБТУ, 2023. 45 с.

Методичні вказівки включають 7 практичних робіт та список літератури до них. Матеріал розкриває сутність проєктування 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

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: Установка MySQL Workbench.

Мета: Провести інсталяцію MySQL Workbench на робочому місці.

Порядок виконання роботи

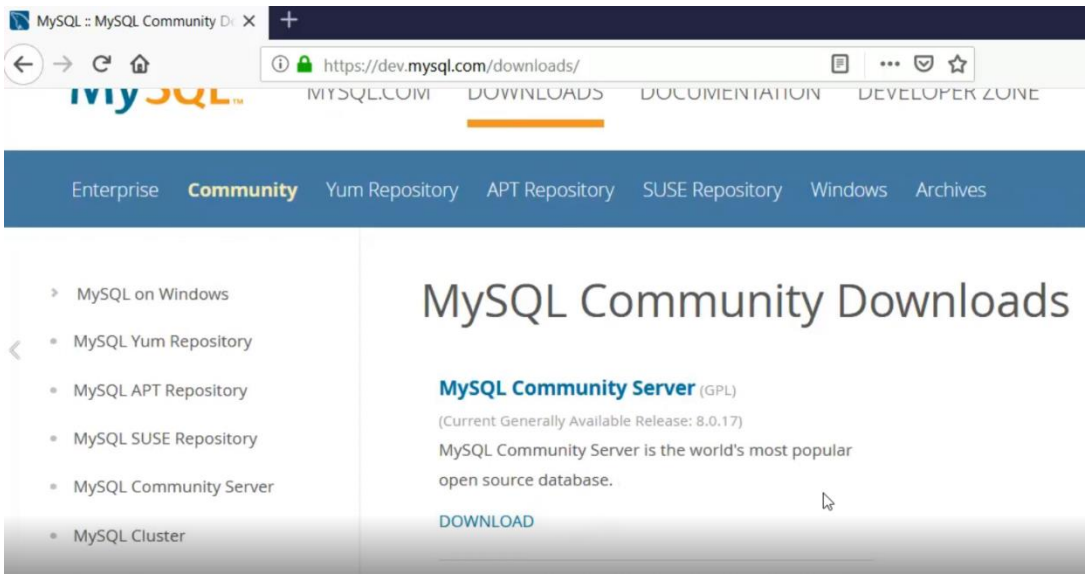
1. Завантажити MySQL Workbench

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

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.

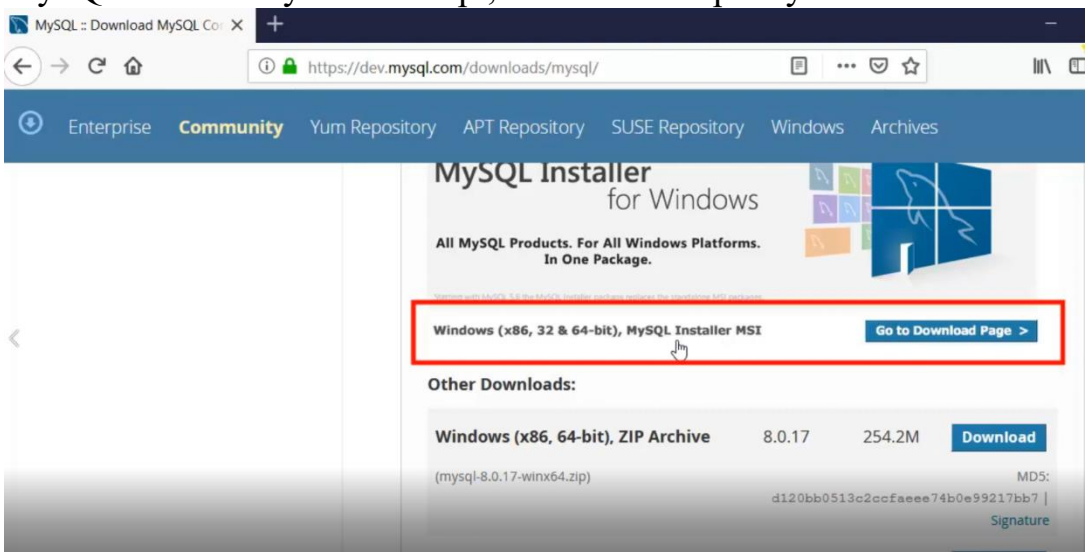
Щоб перейти до установки MySQL 8, спочатку необхідно завантажити дистрибутив цієї програми. Завантажити MySQL 8 в редакції Community можна, як було вже зазначено, абсолютно вільно з офіційного сайту, ось посилання на сторінку завантаження - https://dev.mysql.com/downloads/installer

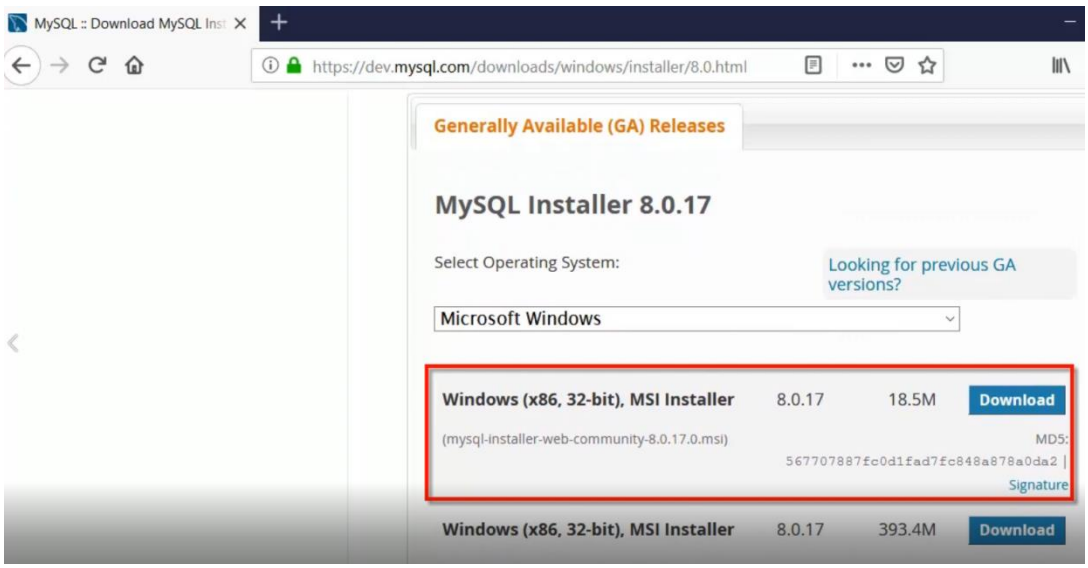


Після переходу на сторінку Ви можете вибрати спосіб завантаження дистрибутива, нам пропонують два способи:

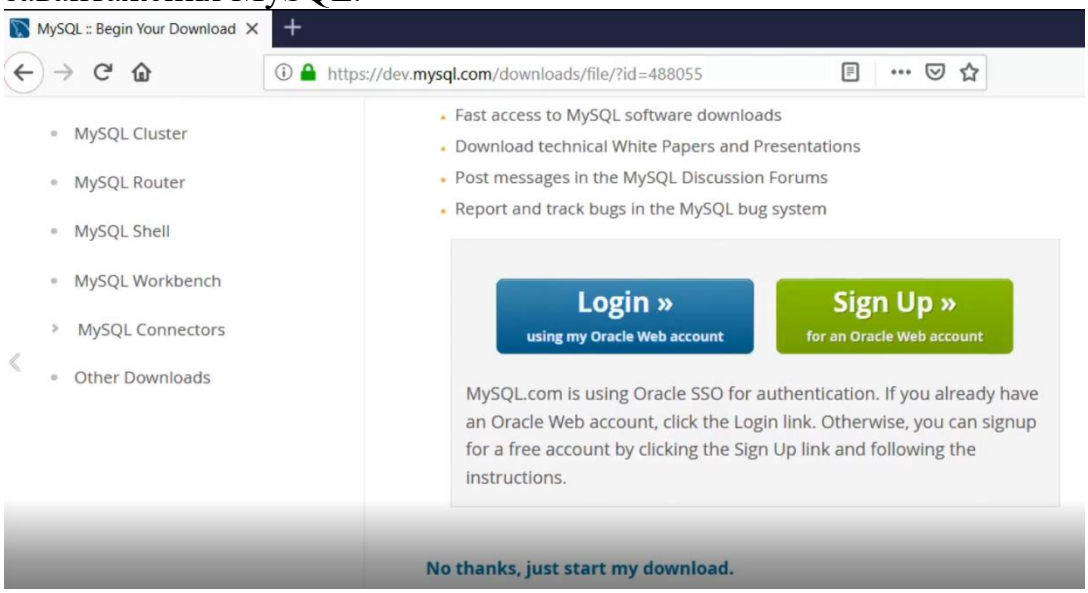
• Завантажити Web-інсталятор - невелика за розміром програма, яка завантажує всі необхідні компоненти для установки MySQL. Можна використовувати для установки MySQL на комп'ютерах, де є інтернет;

• 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 почнеться завантаження MySQL.



Натискаємо на це посилання і чекаємо закінчення завантаження.
В результаті у Вас повинен завантажиться 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 запуститься програма установки MySQL. Спочатку вибираємо тип установки.

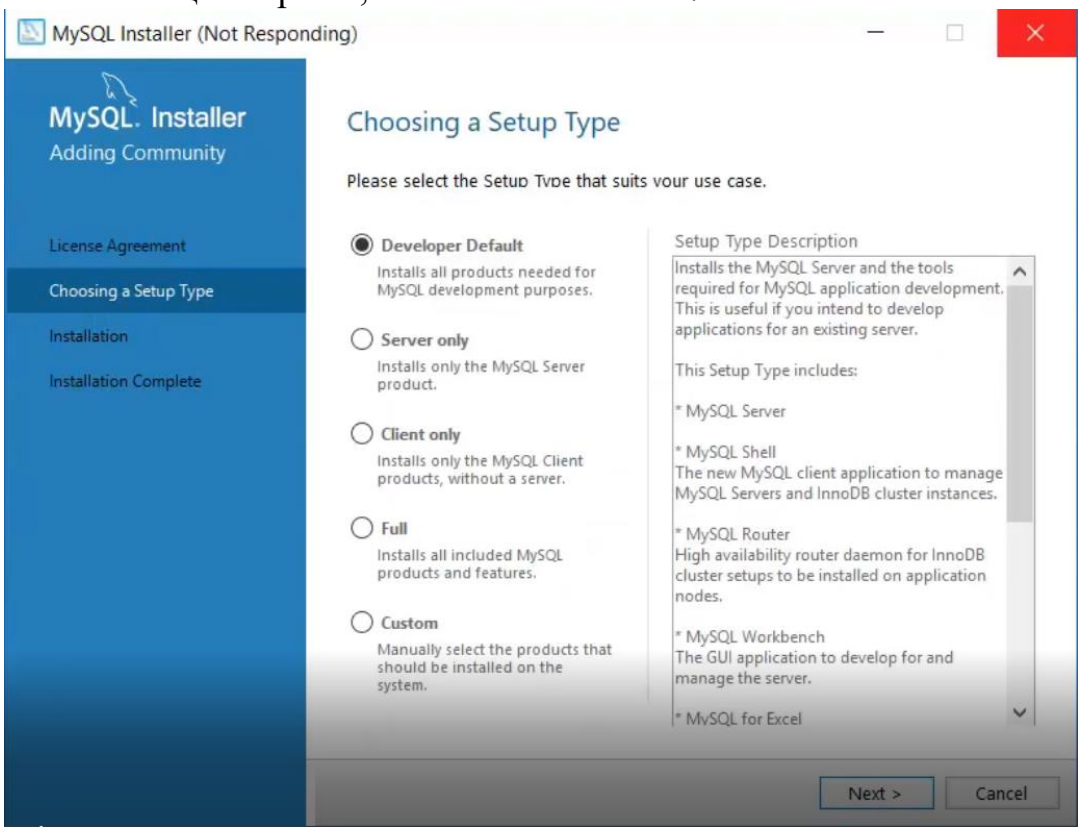
Нам пропонують кілька варіантів:

• 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;

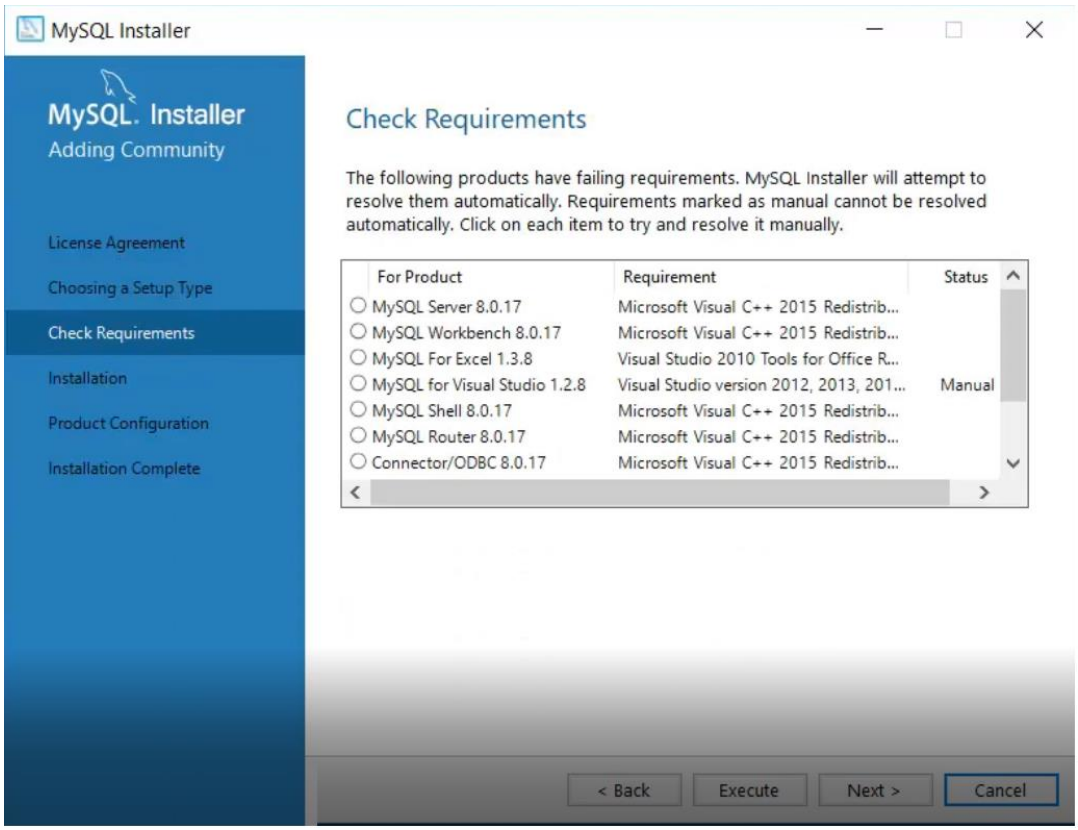
• Client Only - цей варіант передбачає установку тільки клієнтської частини для роботи з MySQL Server, тобто серверна частина встановлюватися не буде;
• Full - установка для всіх включених в дистрибутив компонентів;
• Custom - вибіркова установка, в даному випадку Ви можете встановити лише те, що потрібно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

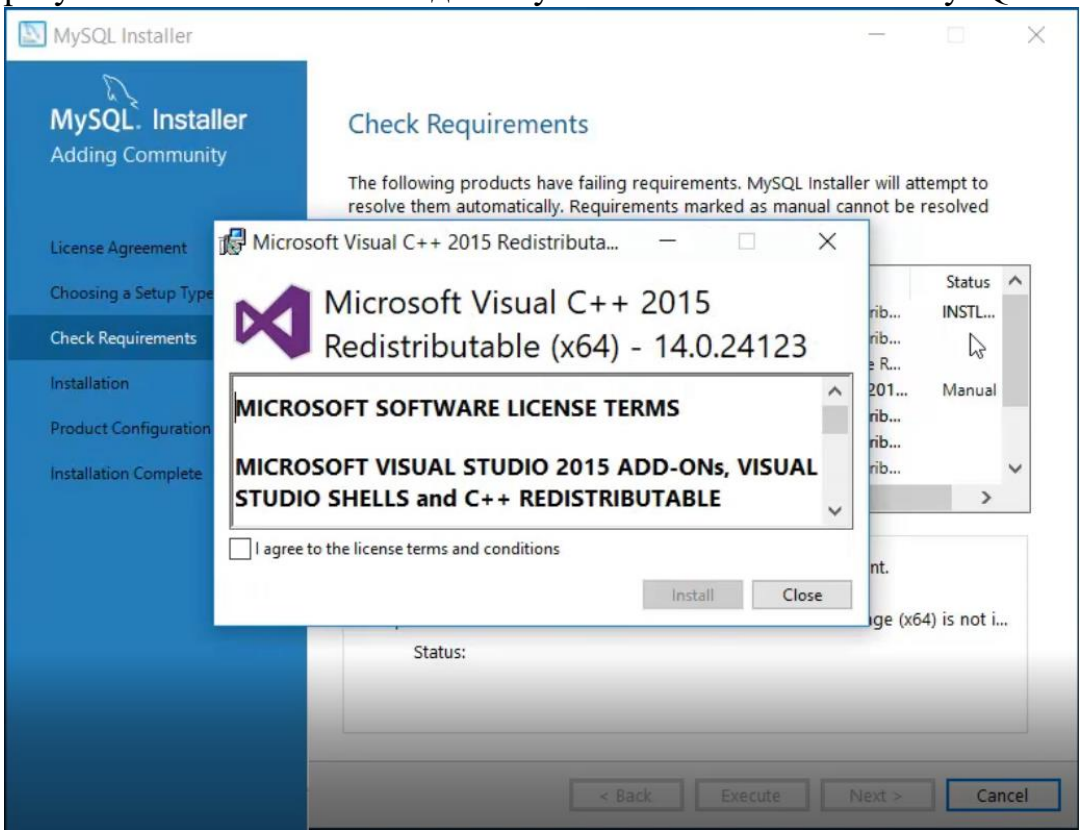


3. Третім етапом відбувається перевірка і установка додаткових компонентів Потім програма установки перевірить систему на наявність компонентів, які потрібні для роботи деяких розширень MySQL. У моєму випадку програма установки видала два попередження для розширень MySQL For Excel і MySQL For Visual 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».

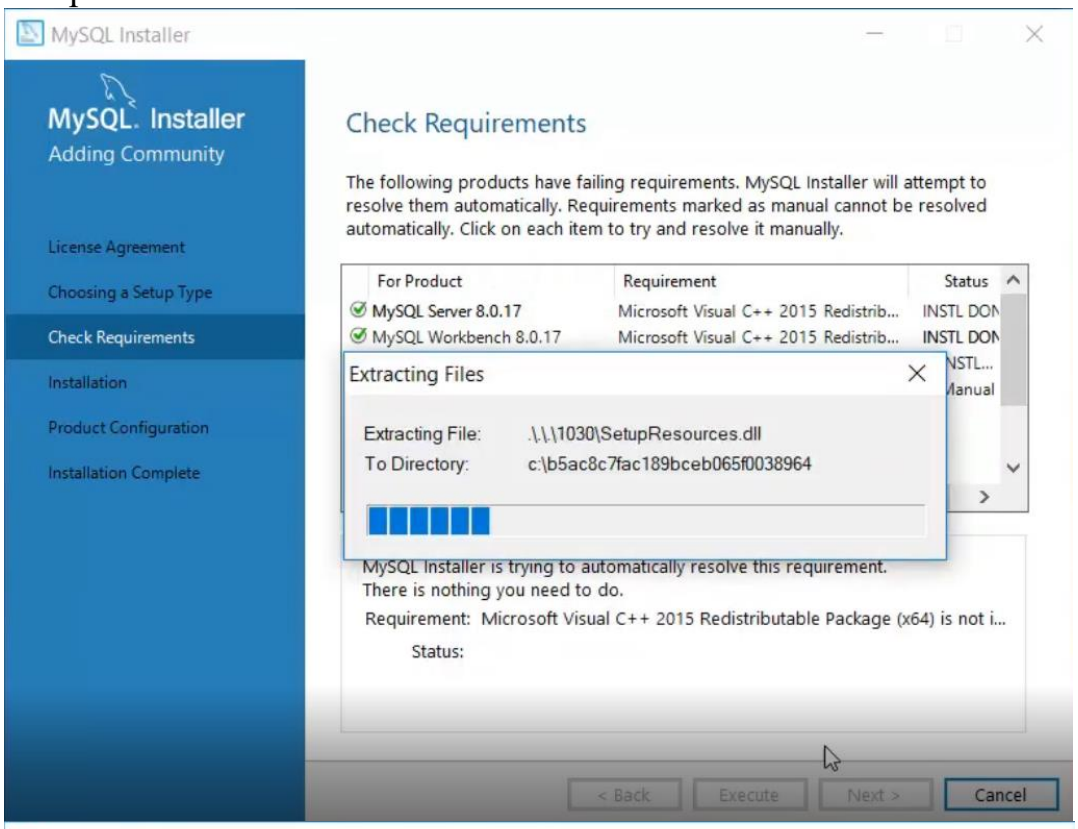


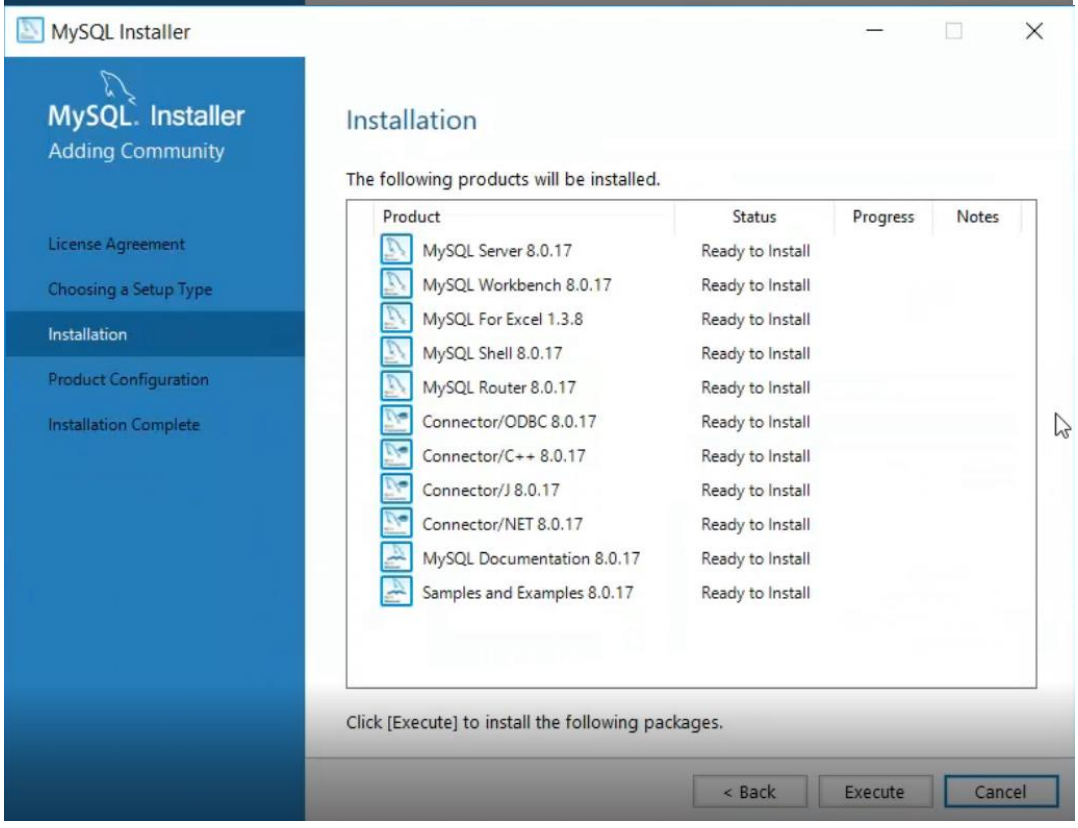
4. Четвертим етапом відбувається установка компонентів MySQL
Все готово для установки MySQL, на даному етапі програма установки видасть всі компоненти MySQL, які готові до установки, натискаємо « Execute» . В результаті почнеться послідовна установка компонентів MySQL.



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

завершена. Натискаємо « Next» .

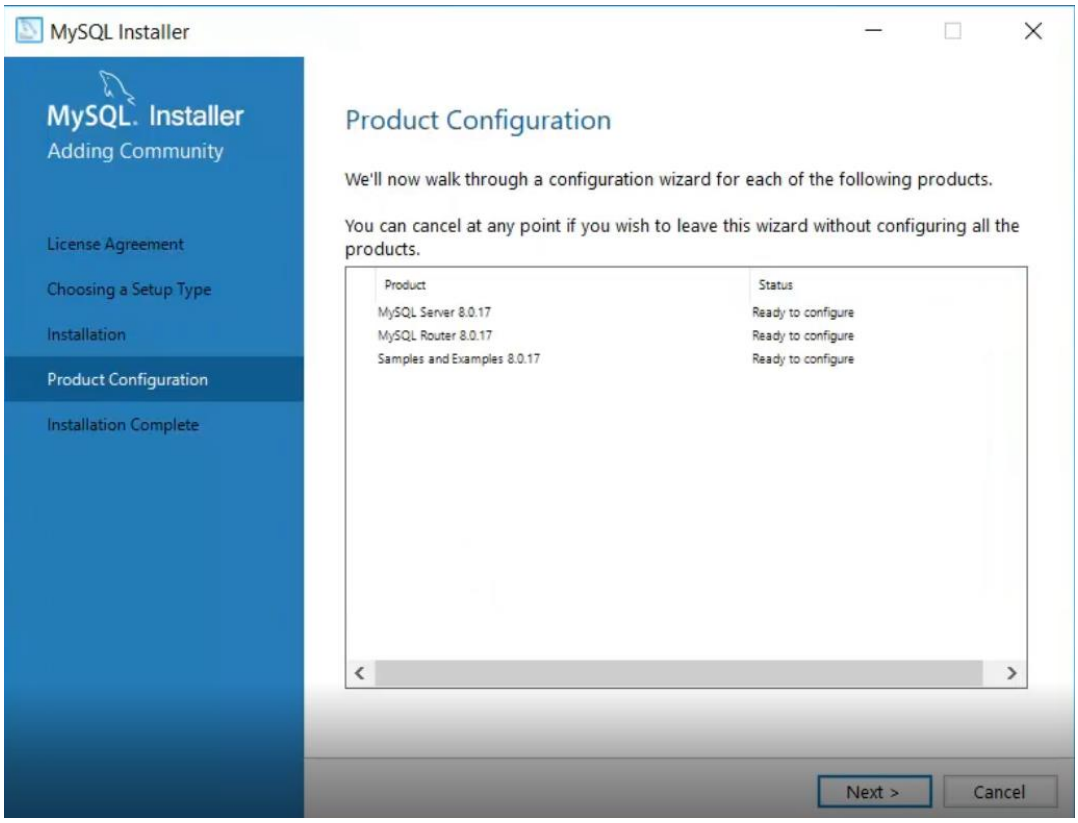




6. Налаштування компонентів MySQL

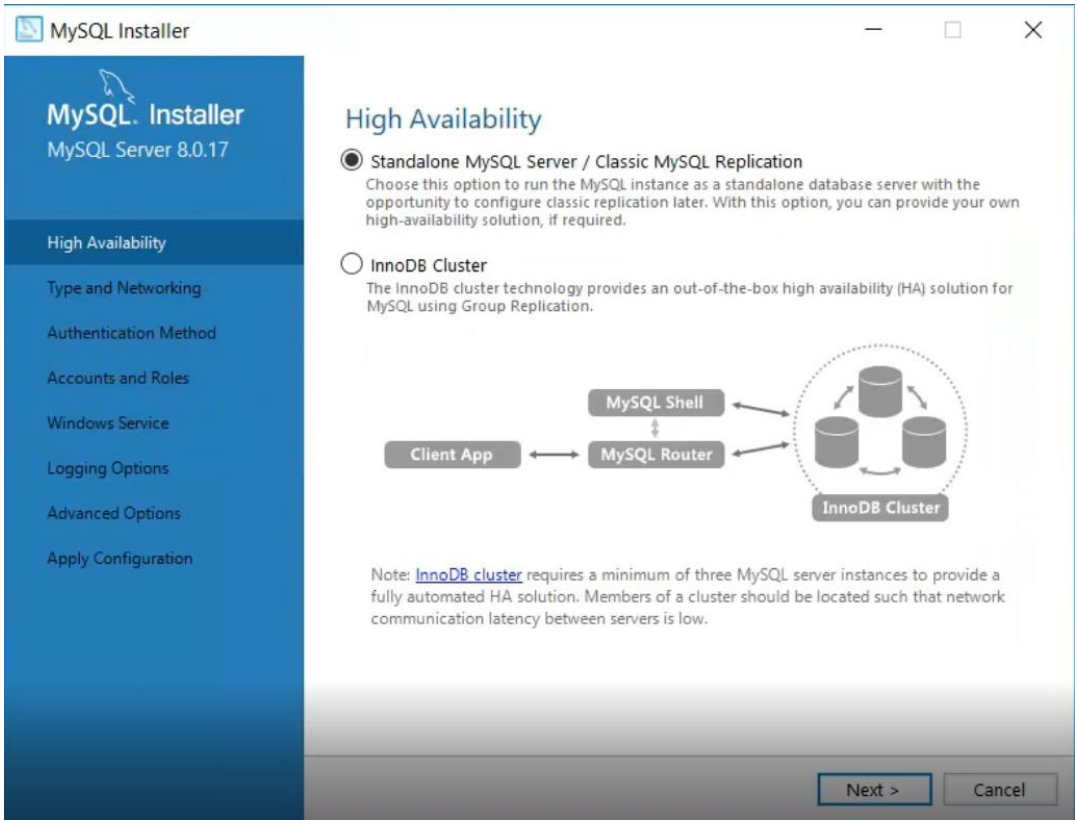
Всі компоненти встановлені, але не всі налаштовані, тому нам необхідно їх налаштувати. Програма установки покаже, які саме компоненти потрібно налаштувати. У нашому випадку це MySQL Server, MySQL Router і Samples and Examples.

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:

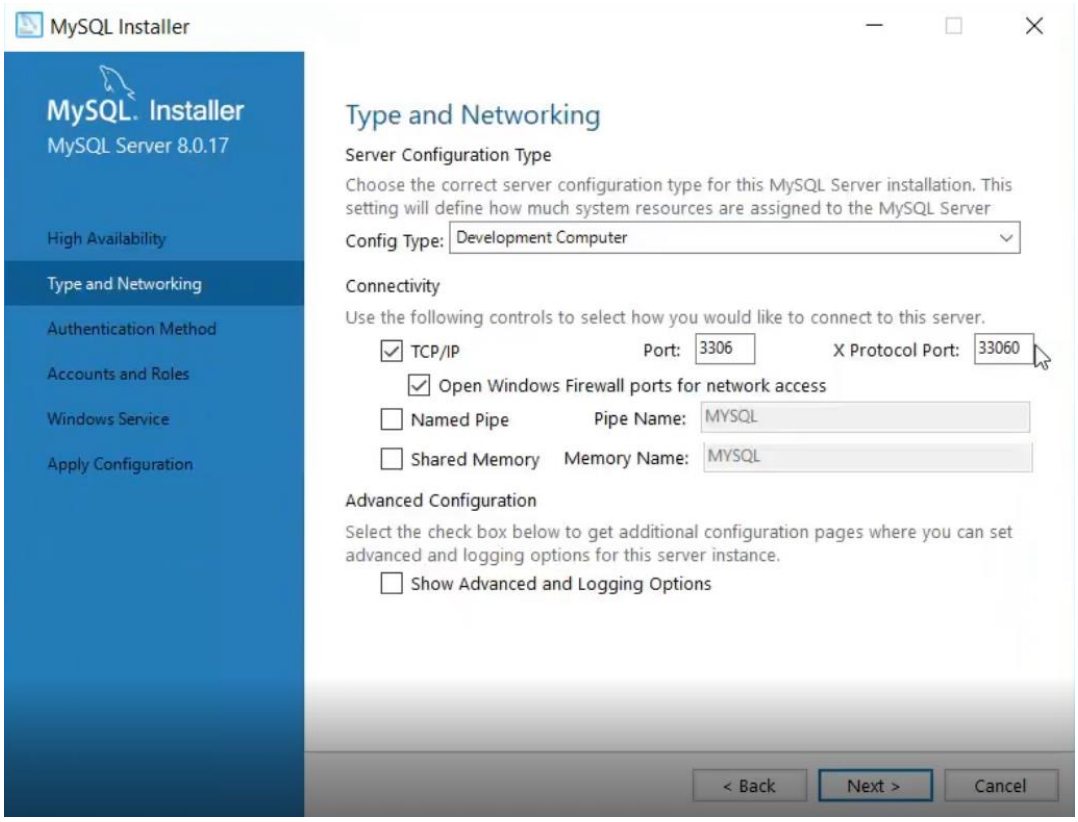
• Тип - Development Computer;

• Протокол - TCP / IP;

• Порт - 3306;

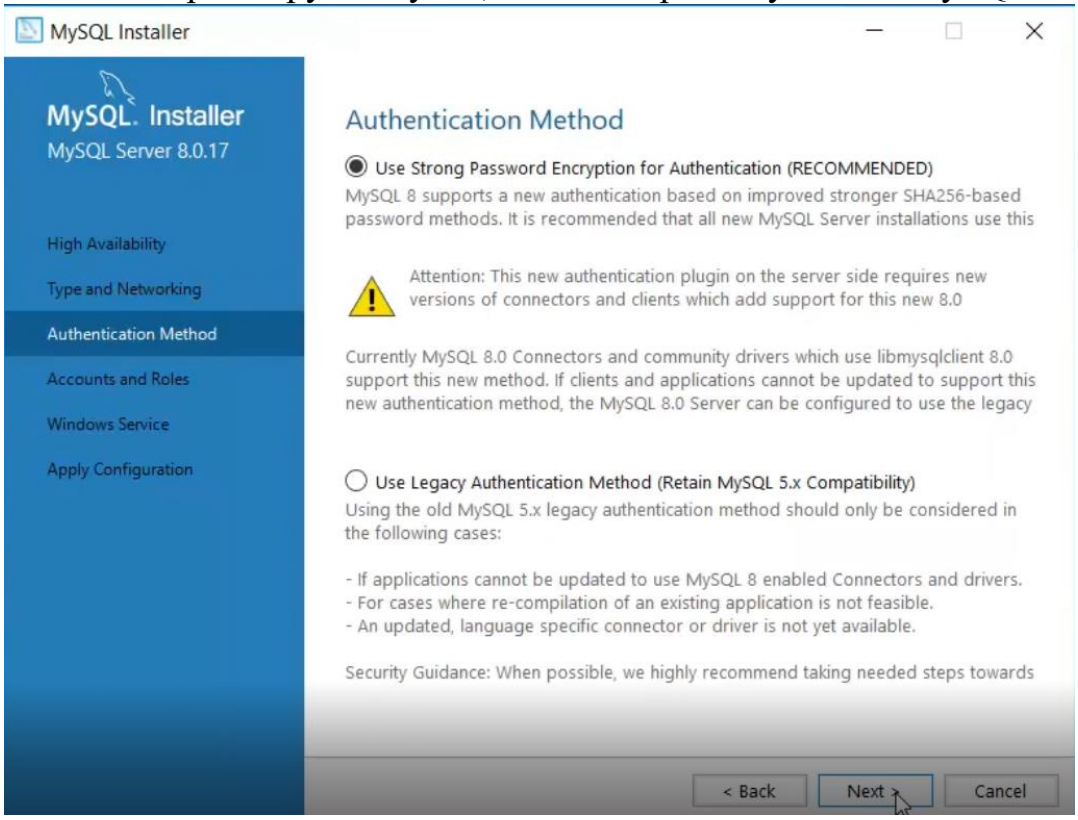
• Галочку «Open Windows Firewall ports for network access» необхідно поставити.

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:

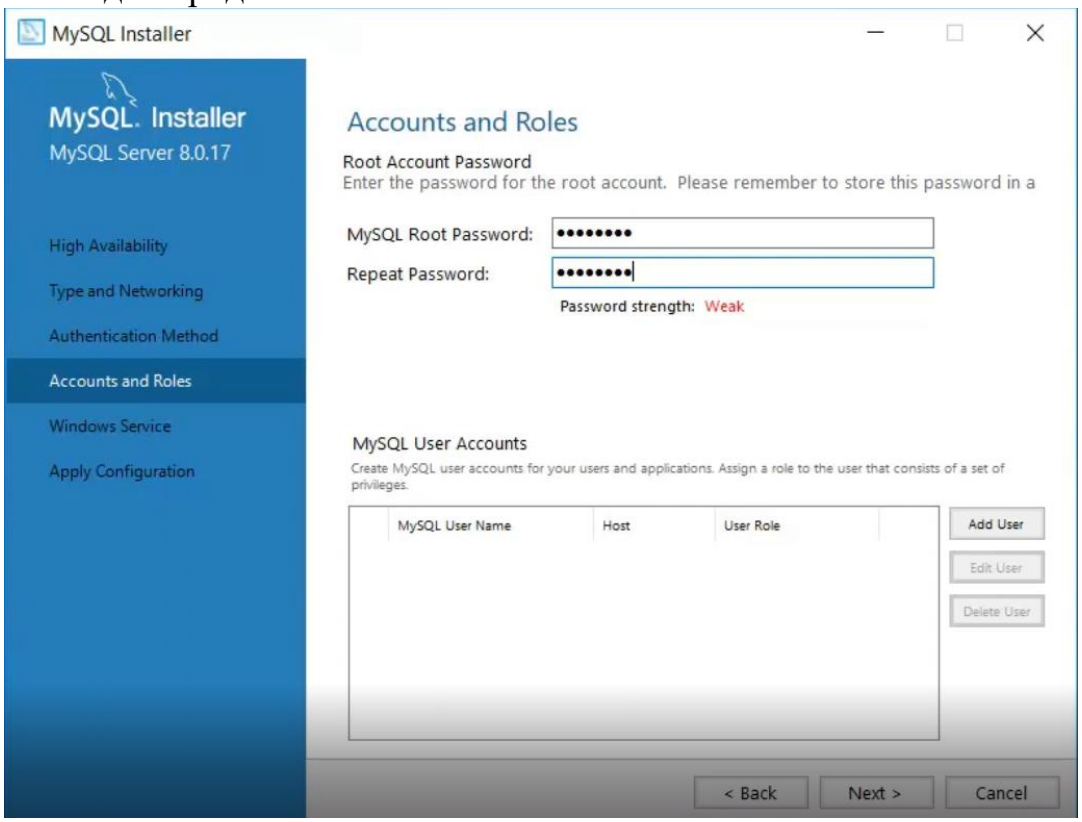
• User Name - вводимо ім'я користувача;

• 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.

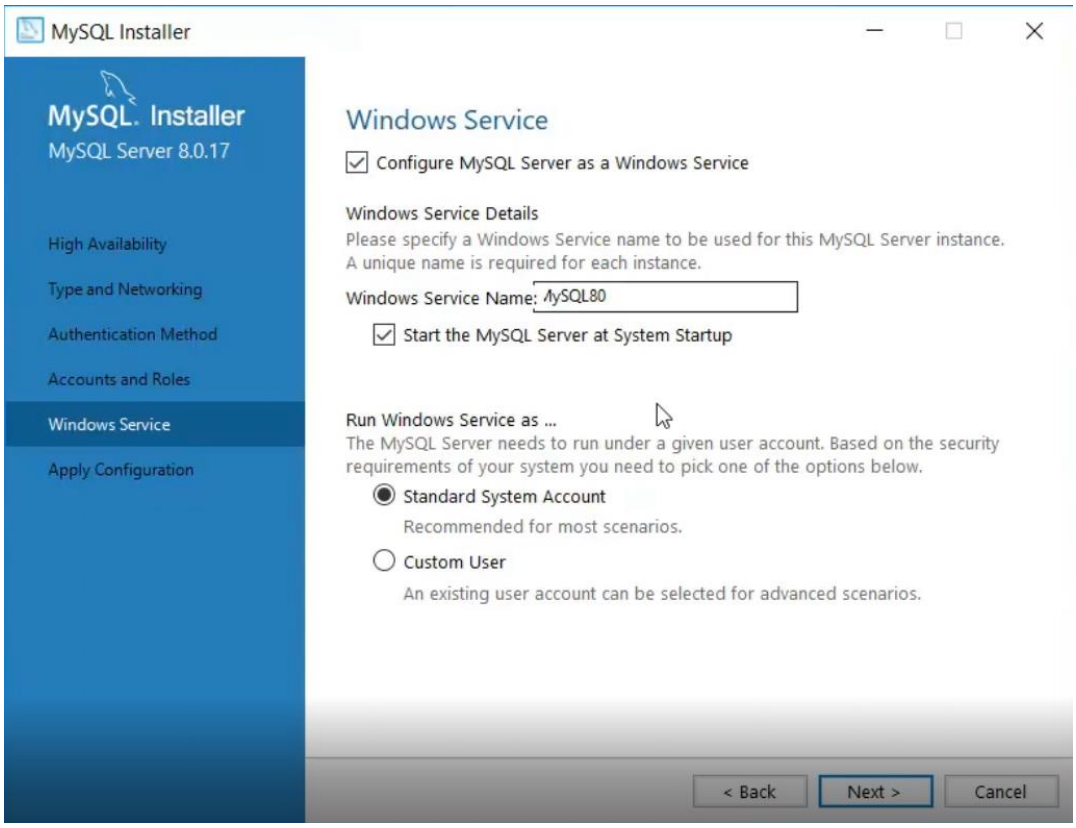
Ми можемо поставити:

• 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;

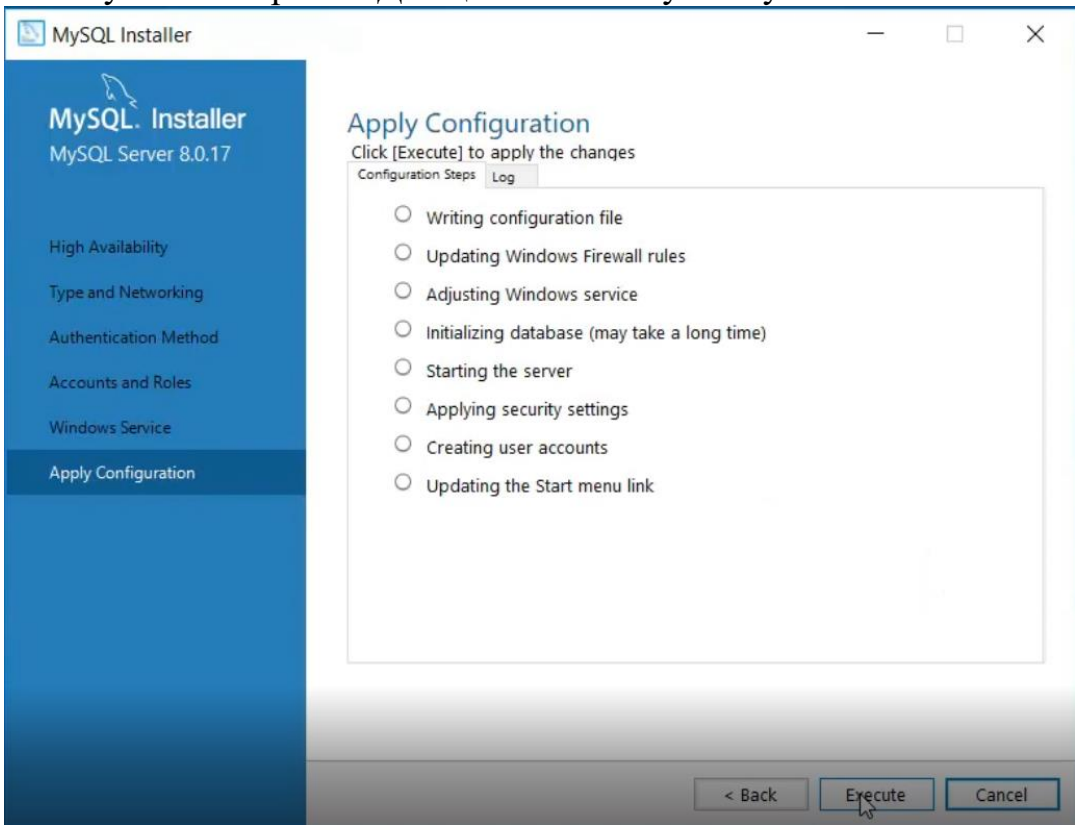
• Run Windows Service - від імені якої облікового запису буде працювати служба MySQL в Windows. Standard System Account - це системна обліковий запис.

Можна все залишити за замовчуванням. Натискаємо « Next» .



12. Застосування параметрів MySQL Server

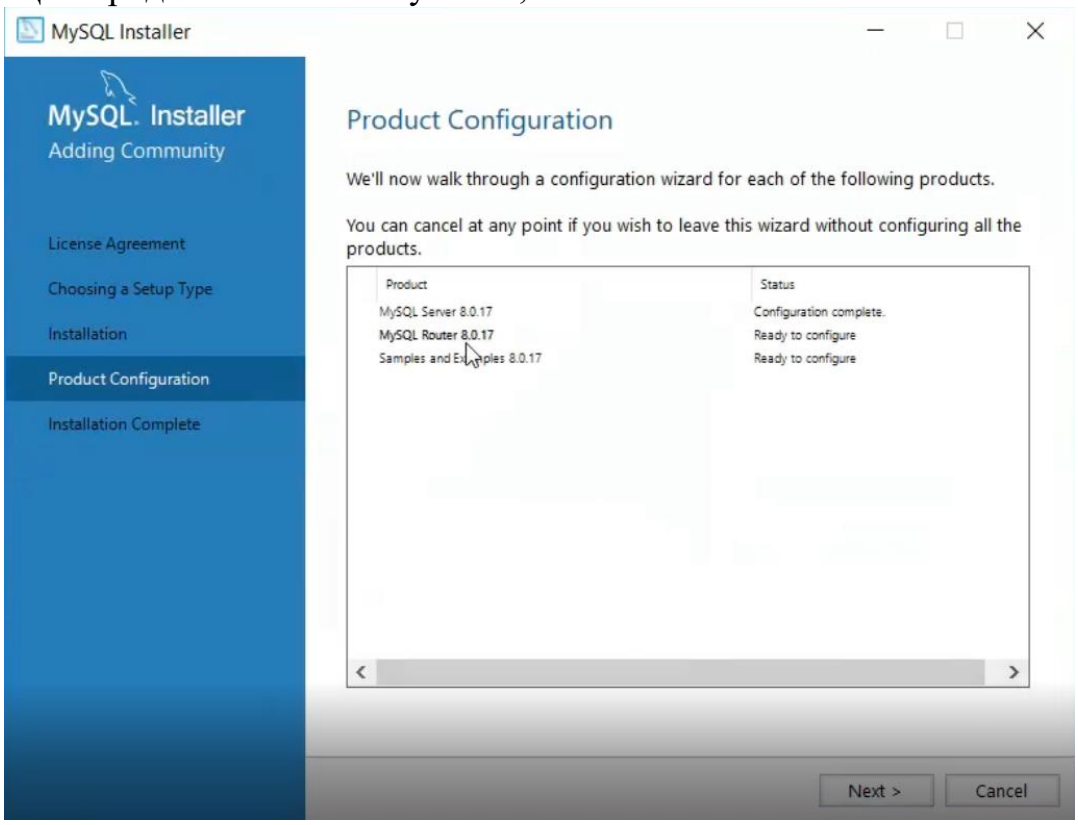
Всі параметри MySQL Server налаштовані, однак нам ще потрібно їх застосувати і зберегти. Для цього на наступному вікні натискаємо « Execute».



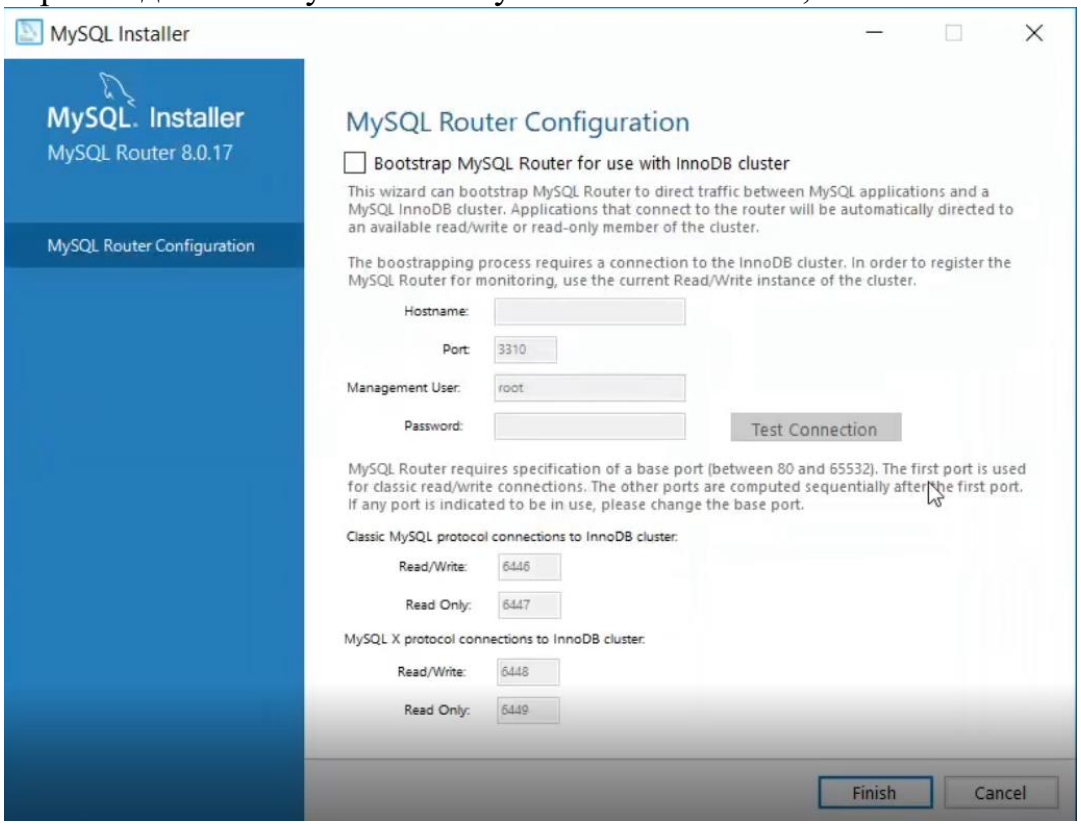
13.Завершення налаштування MySQL Server

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

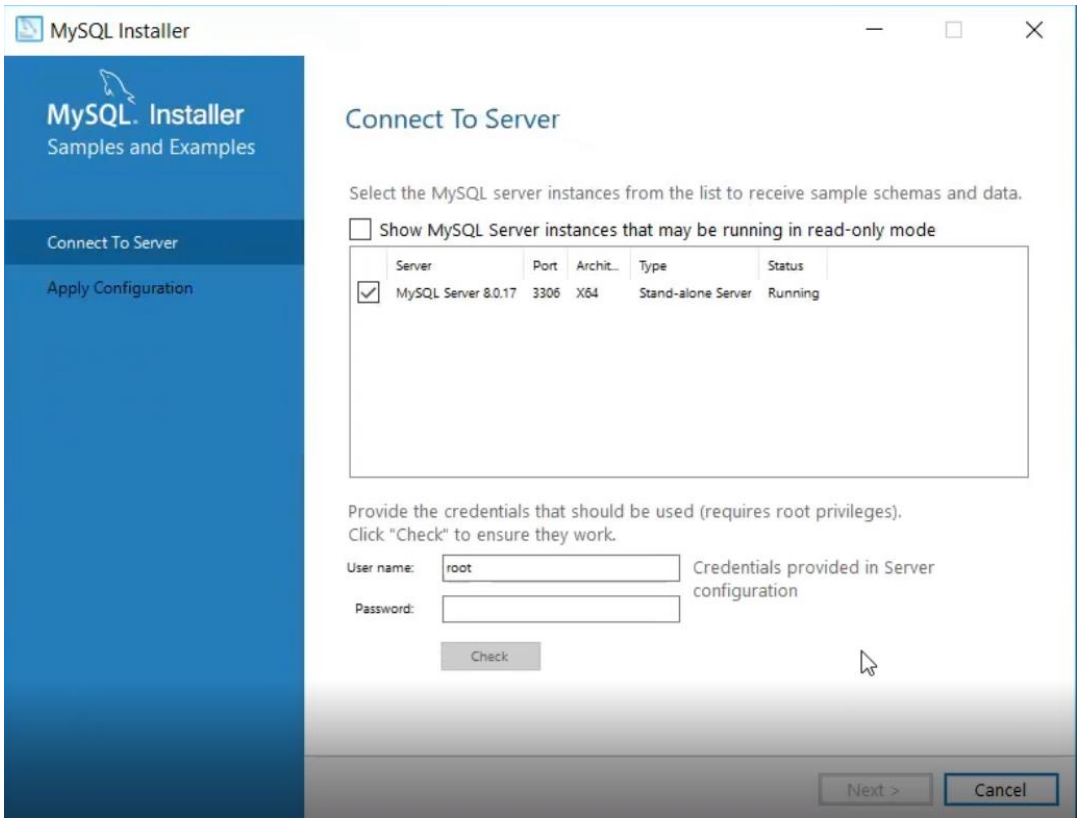
Server буде змінений на «Configuration complete» .
Щоб продовжити налаштування, натискаємо « Next» .



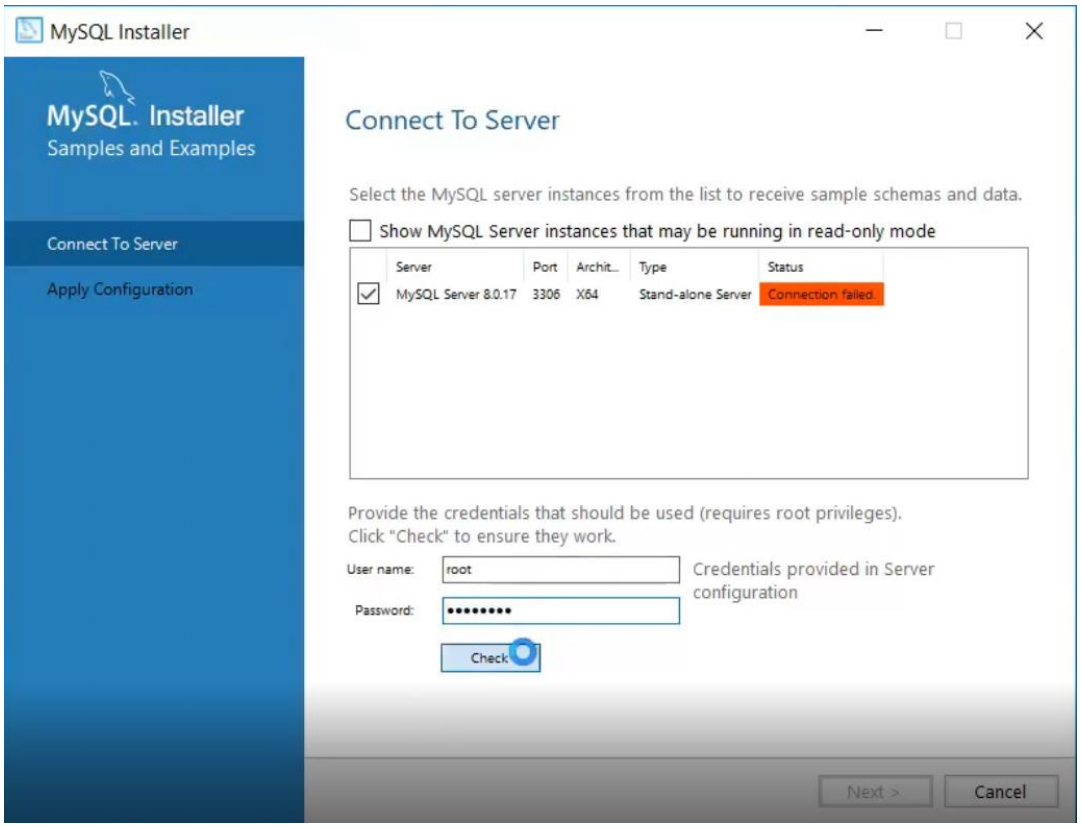
14. Налаштування MySQL Router
В даному випадку нам пропонують налаштувати маршрутизатор MySQL, однак це робити необов'язково, можемо відразу натиснути « Finish» . Для того щоб перейти до налаштування наступного компонента, натискаємо « Next» .



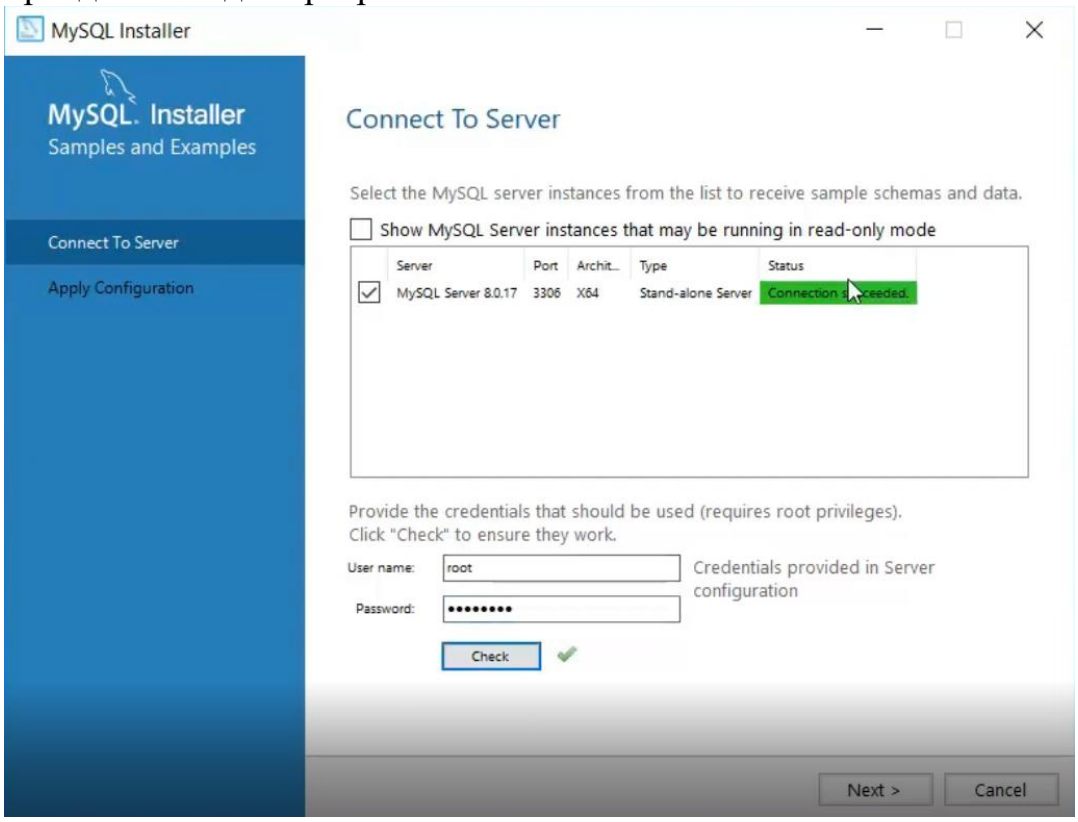
15. Установка тестових даних на MySQL
Далі ми можемо встановити тестові дані на MySQL Server, наприклад, для навчання. Щоб це зробити, вводимо пароль користувача Root і натискаємо « Check» для перевірки, якщо підключення встановлено, значить все добре і можна продовжувати. Натискаємо « Next». Щоб застосувати 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.

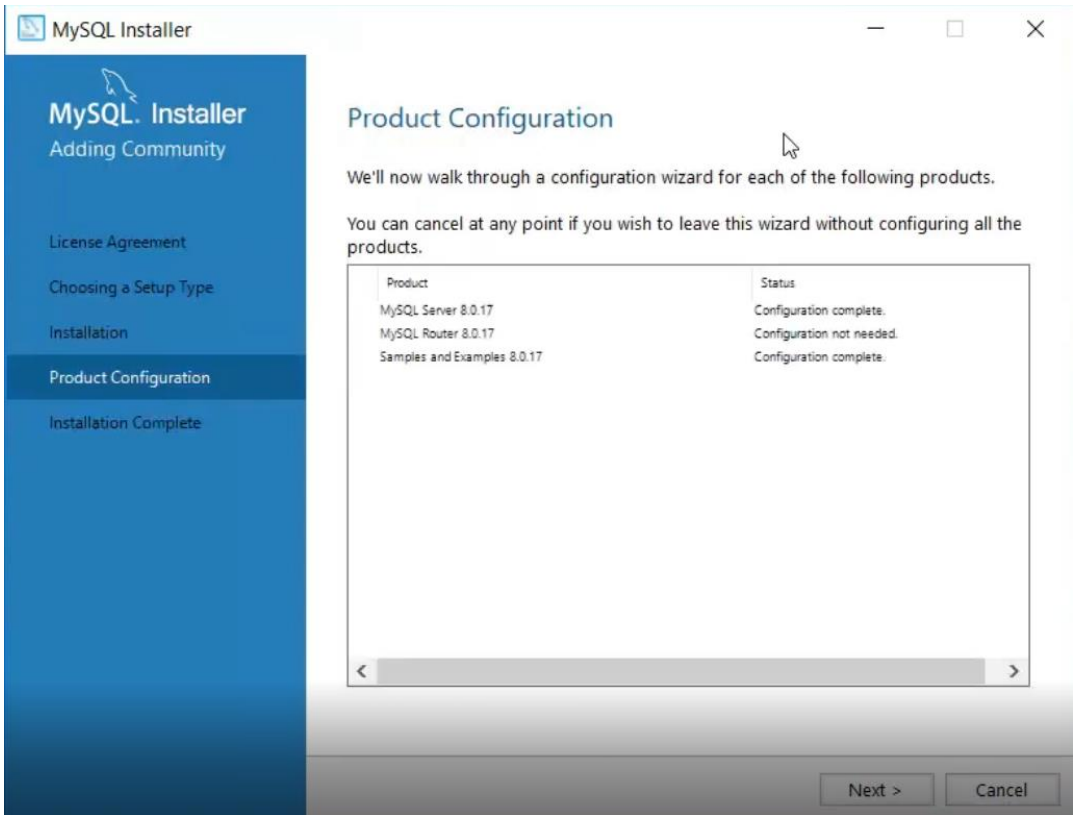


Нижче наведено вікно з правильно введеним паролем та успішним приєднанням до сервера.



16. Завершення налаштування компонентів MySQL

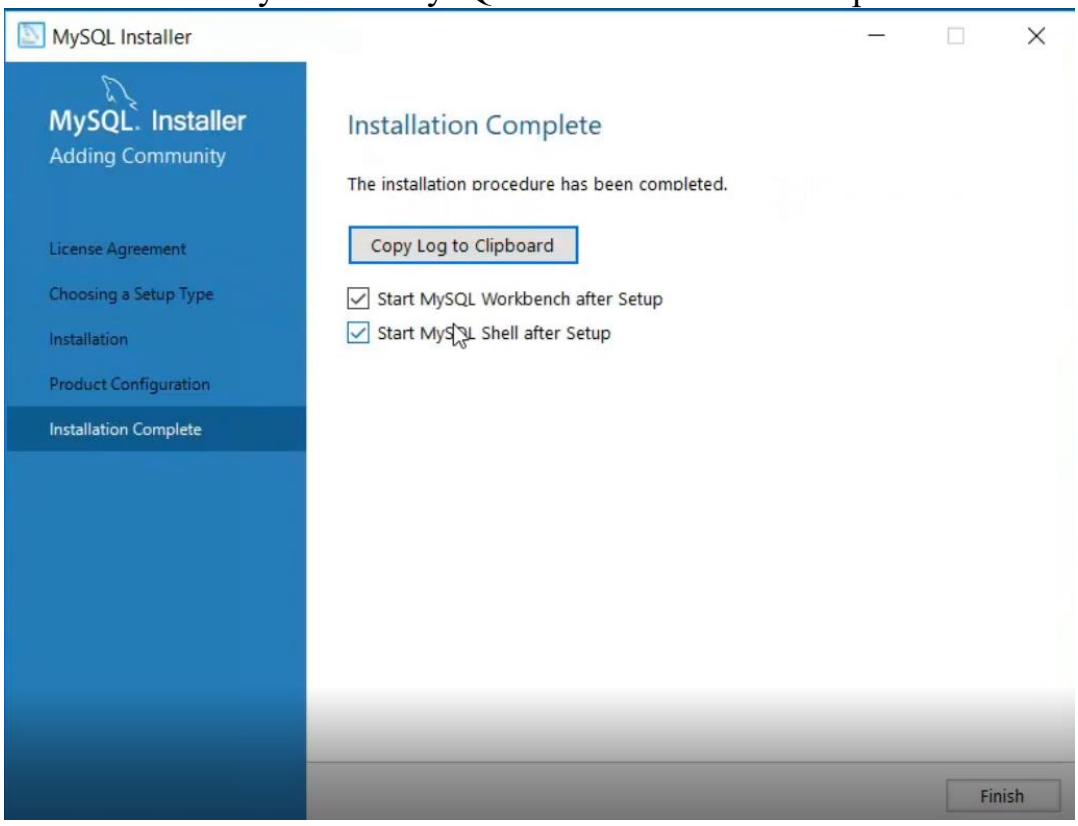
Всі компоненти налаштовані. Натискаємо « Next» .



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

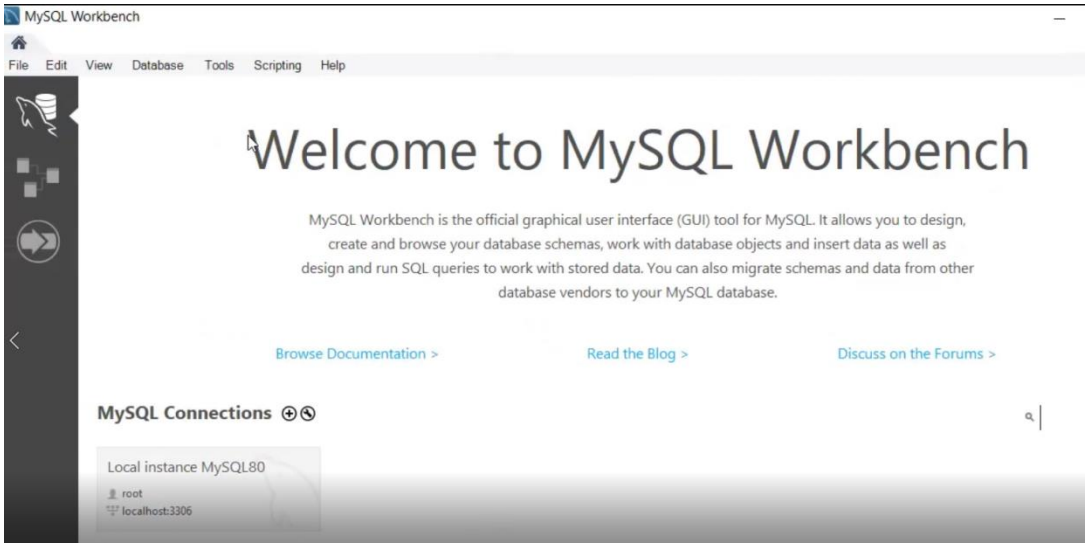
Після установки MySQL в меню Windows 10 з'являться всі необхідні ярлики, які Ви можете використовувати. Наприклад, для запуску середовища «MySQL Workbench 8» можна використовувати відповідний ярлик.

Підключення до MySQL використовуючи MySQL Workbench

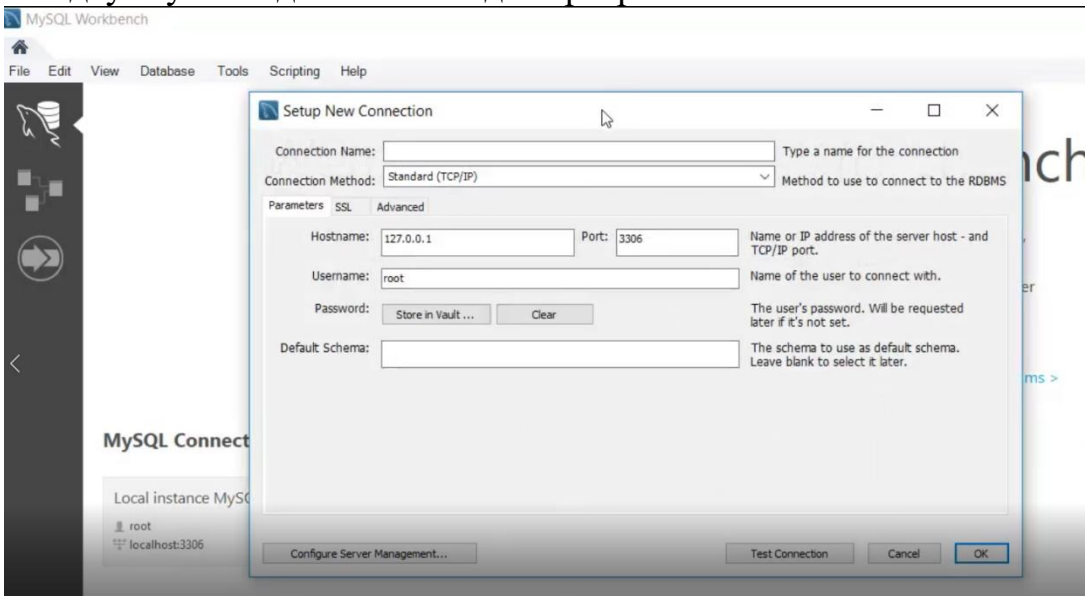
Щоб підключитися до сервера MySQL і подивитися на об'єкти бази даних, на таблиці, уявлення і так далі, запустіть MySQL Workbench, наприклад, як зазначено вище. В результаті відкриється MySQL Workbench, де на стартовій сторінці у нас вже буде одне підключення до локального сервера, яке буде здійснюватися 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.



ЗАВДАННЯ ДЛЯ САМОСТІЙНОЇ РОБОТИ

Провести установку MySQL Workbench на ПК.

Лабораторна робота №2

Тема: Збір та аналіз даних до внесення у таблиці БД.

Мета: Навчитися проводити збір інформації з різних джерел

ЗАВДАННЯ ДЛЯ САМОСТІЙНОЇ РОБОТИ

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

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

Чаї – tea1, tea2 Спеції – sp1, sp2 **Таблиця**.

Товари.

Код товару	Назва	Опис	Ціна	Акція
tea1	Ранок срібносяйний	Завдяки м'яті та мелісі цей чай додає бадьорості та водночас гармонії. Аніс та фенхель стимулюють травлення. Ідеальний чай для початку ранку або для будь-якої пори, коли потрібно підбадьоритись. Ранок срібносяйний — свіжий і легкий, як ранковий світанок, коли сонце, що сходить, підсвічує хмари в прозорому небі над горизонтом.	42	

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	tea1	1
1	Tea10	2
1	Sp1	1

Завдання №4. Провести аналіз документів із замовленнями

Дані обліку замовлення занести у таблицю

Код замов-лення	**Код замов-ника**	Дата замов-лення	Виконання замовлення	Оплата	Знижка
1	**1**				

Лабораторна робота №3

Тема: Логічне конструювання бази даних

Мета: Навчитися проектувати логічну структуру бази даних. Встановити зв'язки між таблицями. Приведення до 3 нормальної форми.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Логічна побудова включає побудову логічної структури та встановлення зв'яків між ними. Таблиці бази даних подаються схематично за допомогою елементів мови UML.

ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

Провести логічне конструювання БД за допомогою UML діаграми

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

Лабораторна робота №4

Тема: Поля таблиці бази даних та їх атрибути

Мета: Створити структуру бази 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 указиватся 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 4 проблеми і в підсумку все одно буде займати 10 символів

- **VARCHAR** : представляє стоку змінної довжини. Довжина

збереженої рядки також указивается 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 заівісімості 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 precisioni 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 yyyy-mm-dd. 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 "yyyy-mm-dd 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 "yyyy-mm-dd". Деякі з прийнятих форматів:

- yyyy-mm-dd - 2018-05-25
- yyyy-m-dd - 2018-5-25
- yy-m-dd - 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.

- yyyymmdd - 20180525
- yyyy.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

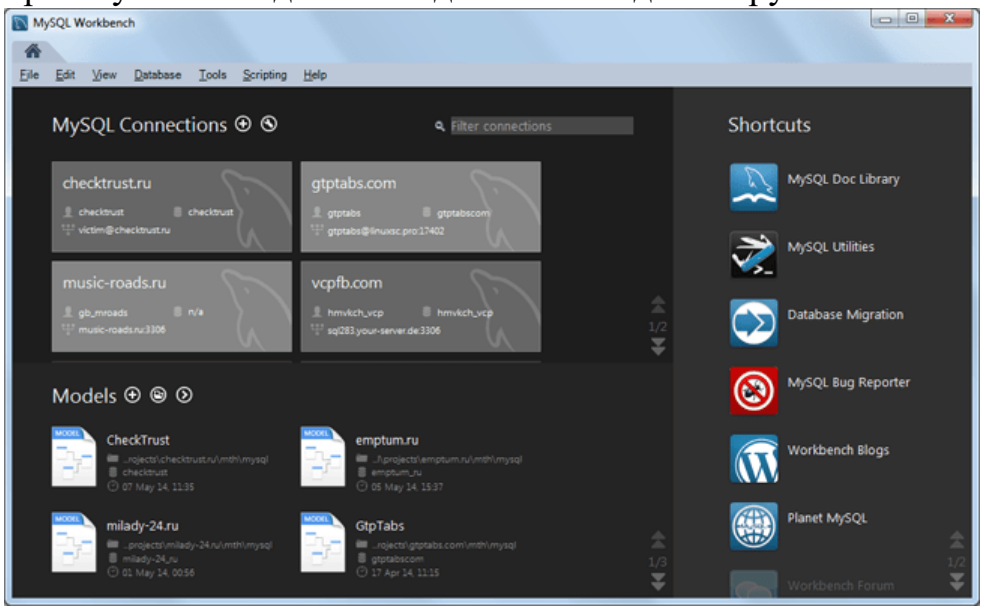
- **ENUM** : зберігає одне значення зі списку допустимих значень. Займає 1-2
- **SET** : може зберігати кілька значень (до 64 значень) з деякого списку допустимих значень. Займає 1-8 байт.

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

Початок роботи у програмі My SQL Workbench

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

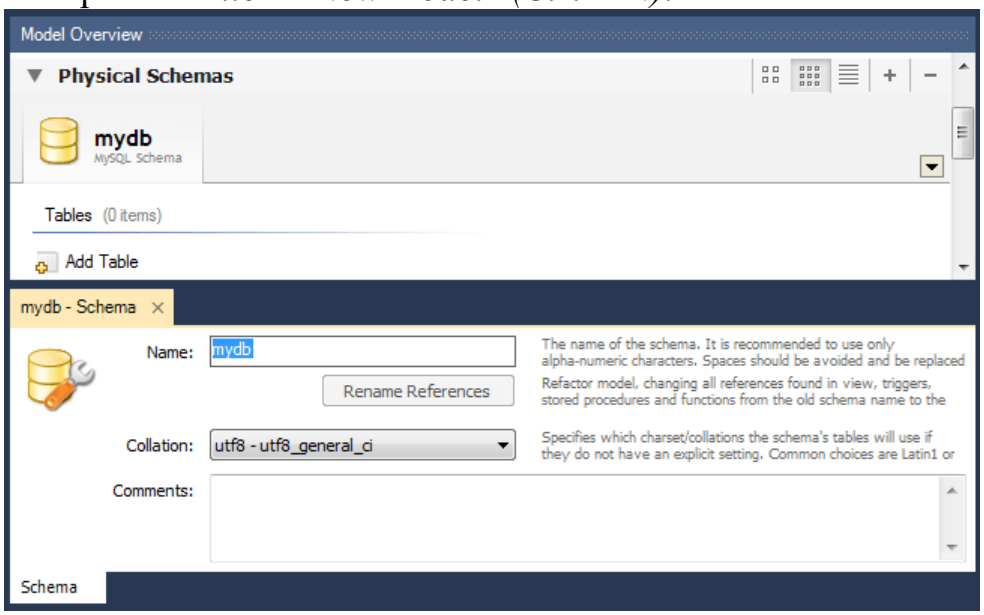


У верхній частині екрану знаходиться список підключень до MySQL серверів ваших проектів, а список останніх відкритих моделей даних - в нижній частині екрана. Робота зазвичай починається з створення схеми даних або завантаження

існуючої структури в MySQL Workbench.

Створення та редагування моделі даних

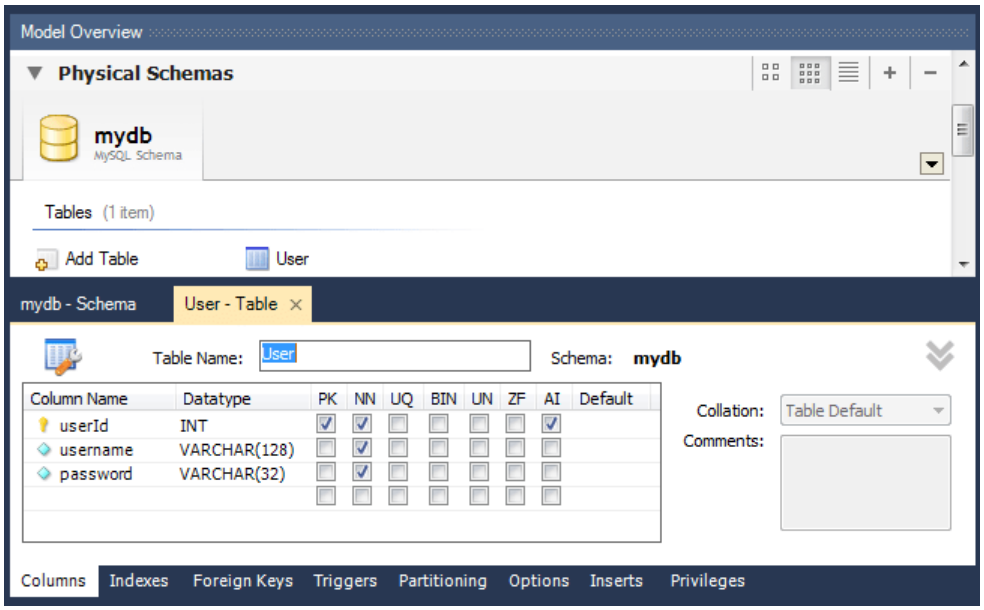
Для додавання моделі натискаємо плюсик поруч із заголовком "Models" або вибираємо *"File → New Model" (Ctrl + N):*



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

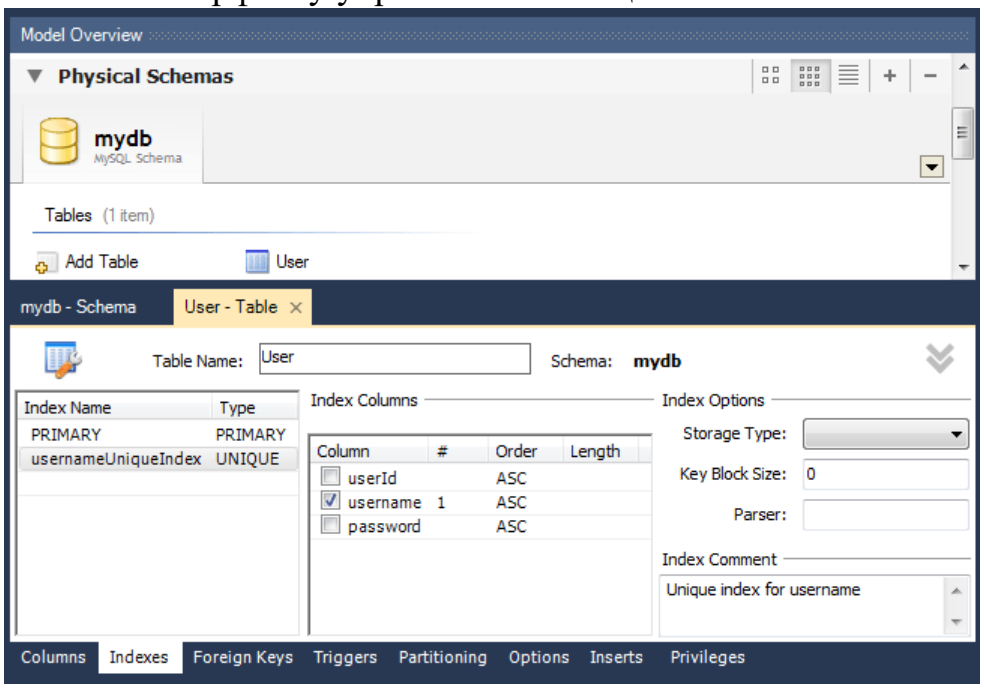
Список баз даних проекту і список таблиць в межах бази даних буде розташовуватися у вкладці *"Physical Schemas"*.Щоб створити таблицю, двічі натискаємо на *"+ Add Table":*



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-інкреміроване (AI)*і *значення за замовчуванням (Default)*.

Управління індексами

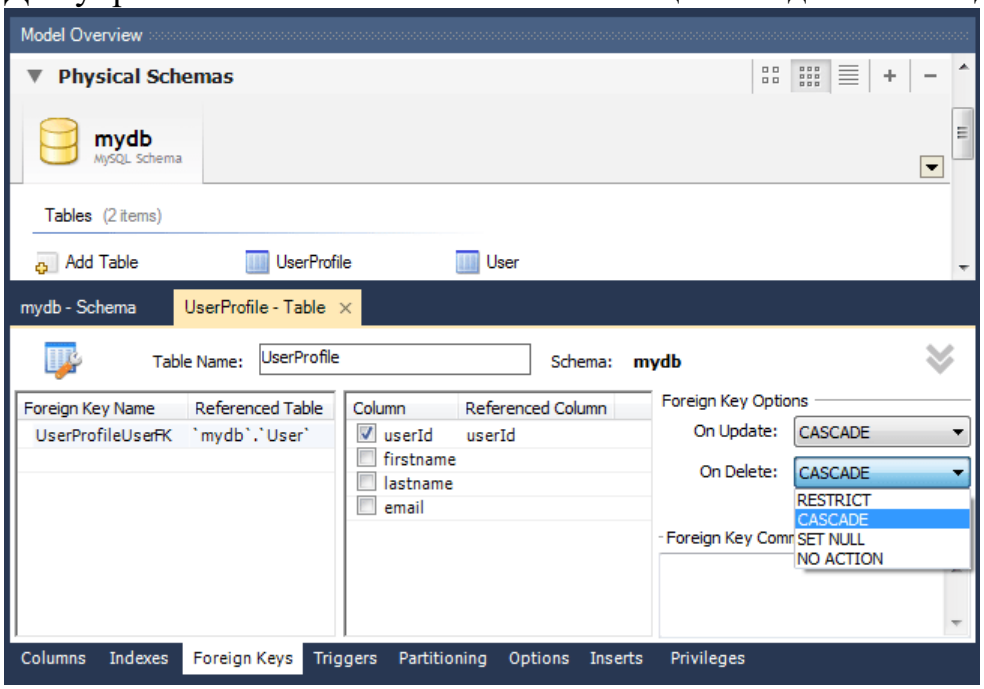
Додавати, видаляти і редагувати індекси таблиць можна у вкладці *"Indexes"*інтерфейсу управління таблицею:



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 полю *username*.

Зв'язки між таблицями

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



Для додавання зв'язку відкриваємо вкладку *"Foreign Keys"* **дочірньої**

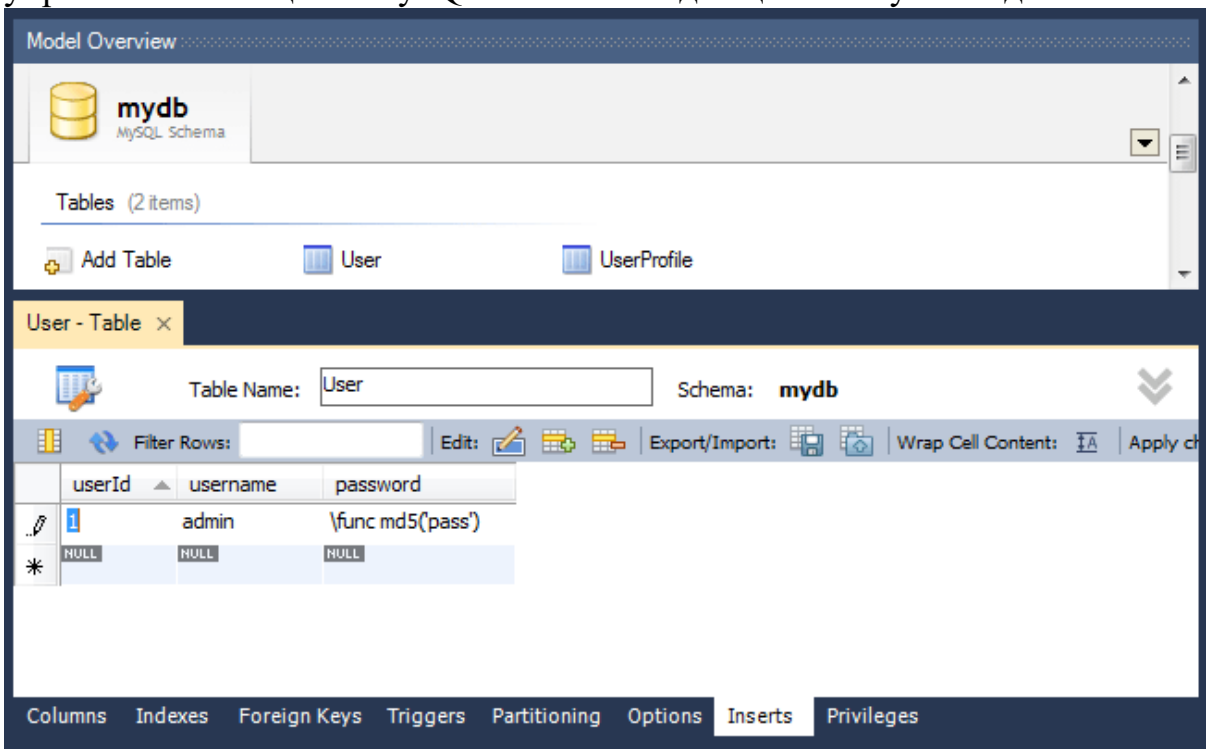
таблиці, Вводимо ім'я зовнішнього ключа і вибираємо **таблицю-батька**. Далі в середній частині вкладки в графі *Column* вибираємо поле-ключ з дочірньої таблиці, а в графі *Referenced Column* – відповідне поле з батьківської таблиці (тип полів повинен збігатися). При створенні зовнішніх ключів **в дочірній 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 базу 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")*.

ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

На основі логічної моделі БД визначити типи полів та атрибути для збереження інформації

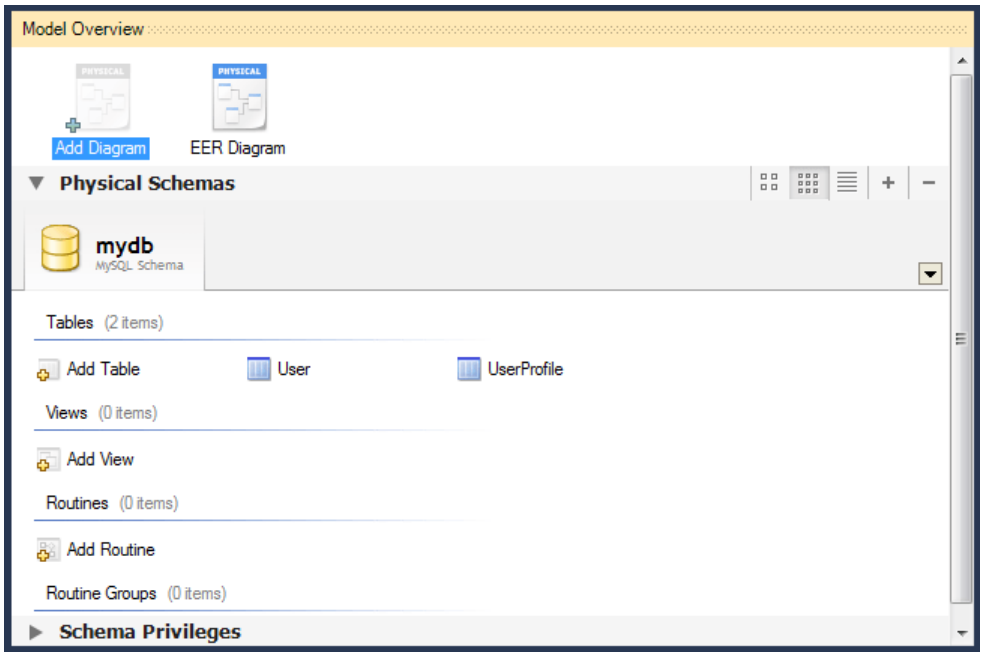
Лабораторна робота №5

Тема: Створення та редагування моделі даних. Робота з таблицями
Мета: Визначити типи полів таблиці бази даних та необхідні атрибути, розробити модель даних

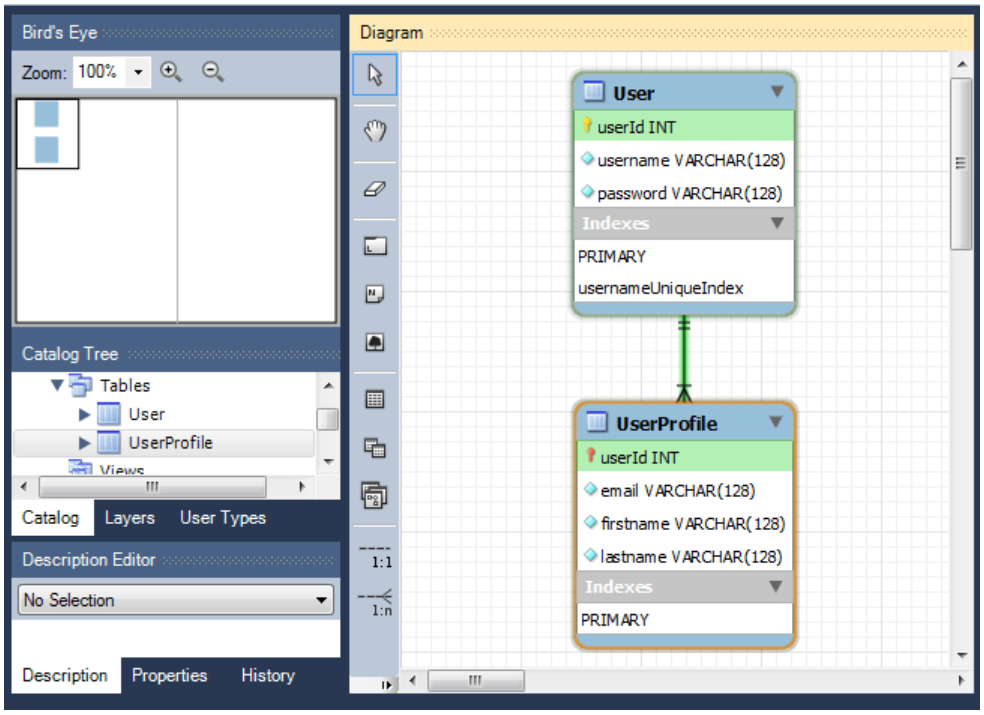
ТЕОРЕТИЧНІ ВІДОМОСТІ

Створення EER діаграми (діаграми "сутність-зв'язок")

Для представлення схеми даних, сутностей і їх зв'язків в графічному вигляді в MySQL Workbench існує редактор EER-діаграм. Для створення діаграми у верхній частині екрану управління базою даних двічі натискаємо на іконку *"+ Add Diagram":*



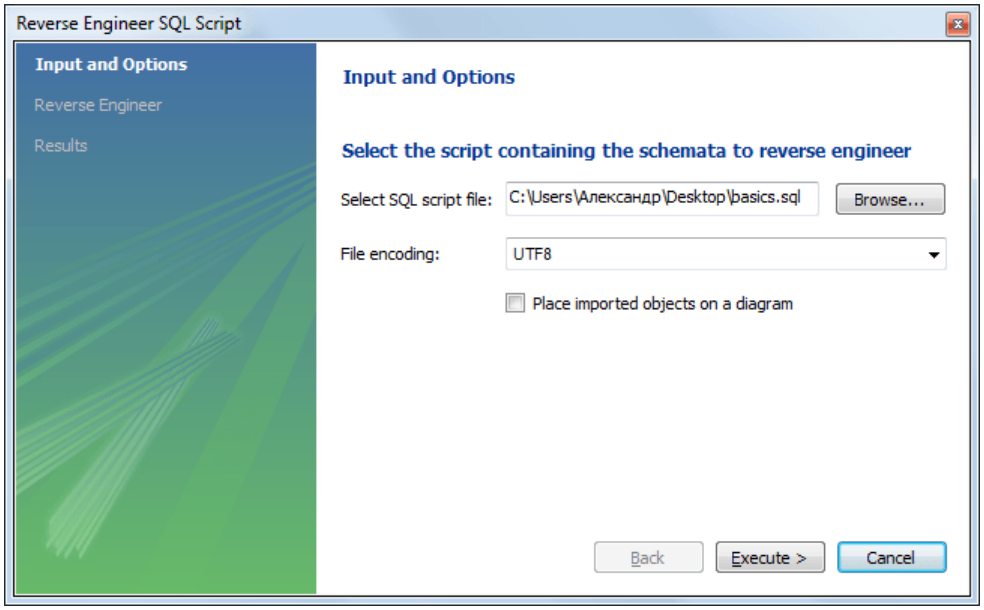
У його інтерфейсі можна створювати і редагувати таблиці, додавати між ними зв'язки різних типів. Щоб додати вже існуючу в схемі таблицю на діаграму, просто перетягніть її з панелі *"Catalog Tree"*.



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

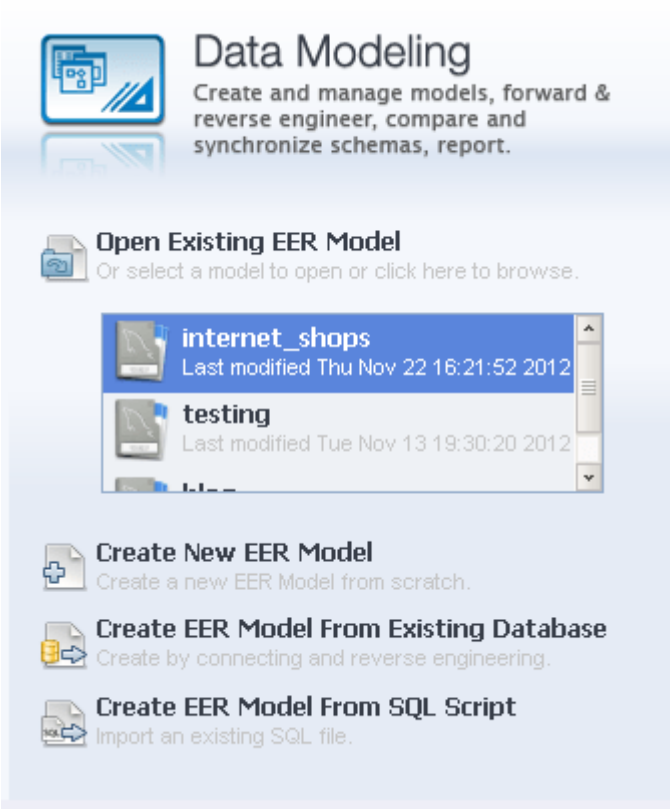
Імпорт існуючої схеми даних (з SQL дампа)

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>"*



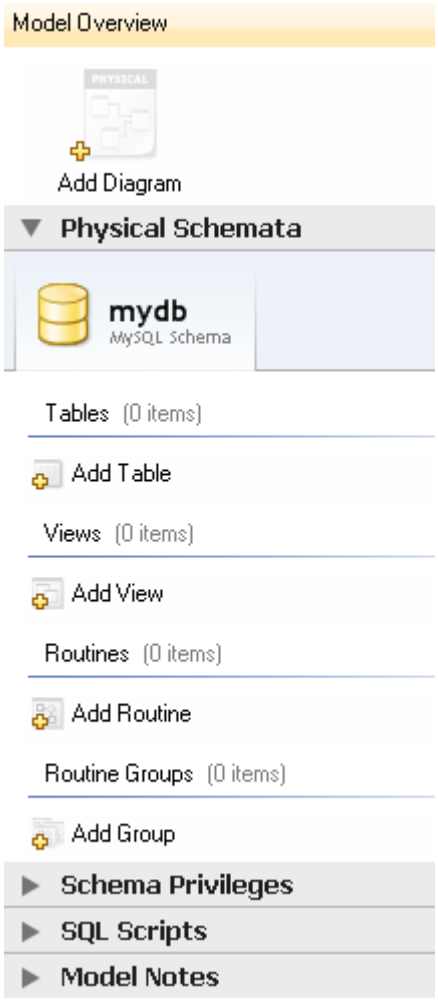
В MySQL Workbench так само передбачений імпорт і синхронізація моделі даних наряму з віддаленим сервером. Для цього потрібно створити підключення віддаленого доступу до MySQL.

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



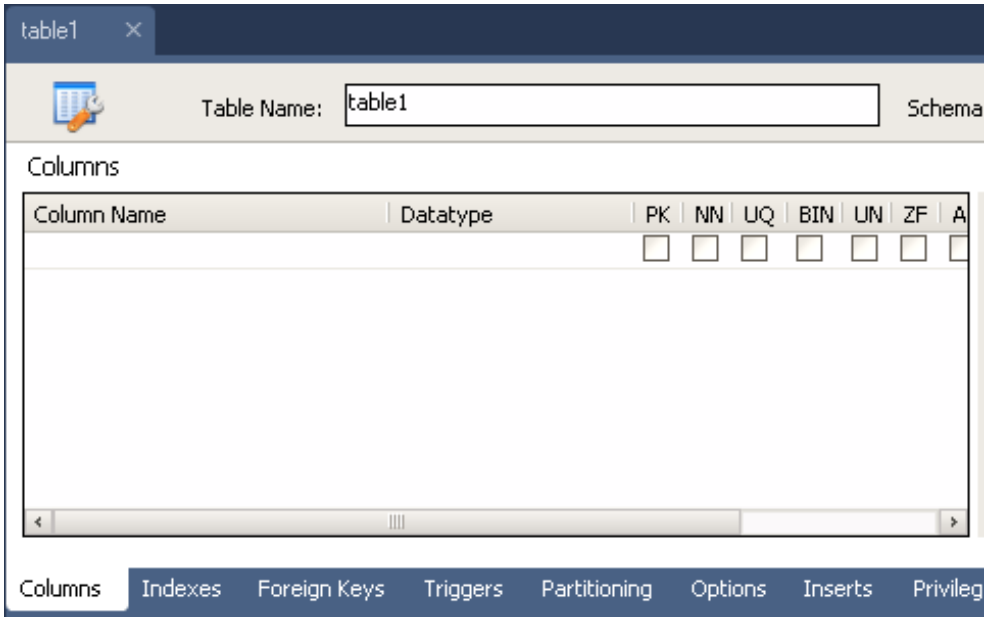
Для того, щоб відкрити існуючу модель треба натиснути на посилання: **Open Existing EER Model**, Для створення нової моделі – вибрати параметр: **Create New EER Model**. Щоб створити модель «сутність-зв'язок» з існуючої бази даних – натиснути на параметр: **Create EER Model From Existing Database**,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

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

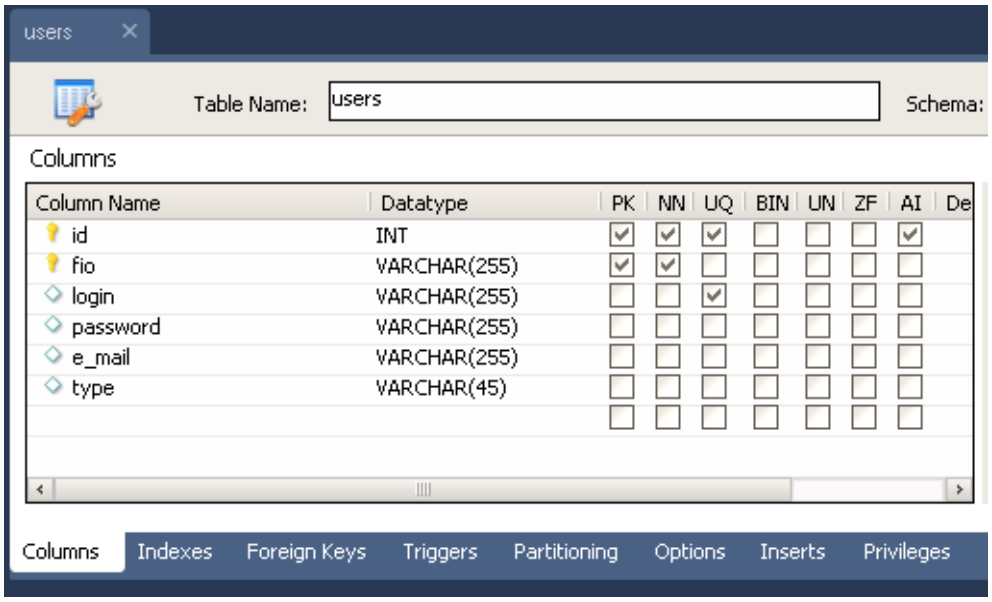


Порядок виконання роботи

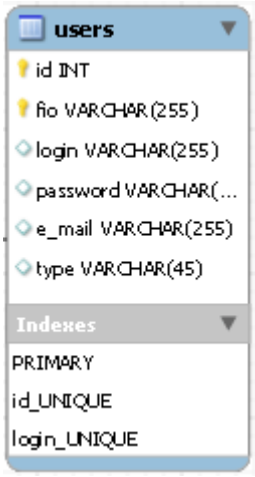
Для початку необхідно створити таблиці, для цього натиснемо на кнопку **Add Table**, з'явиться наступна форма:



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:

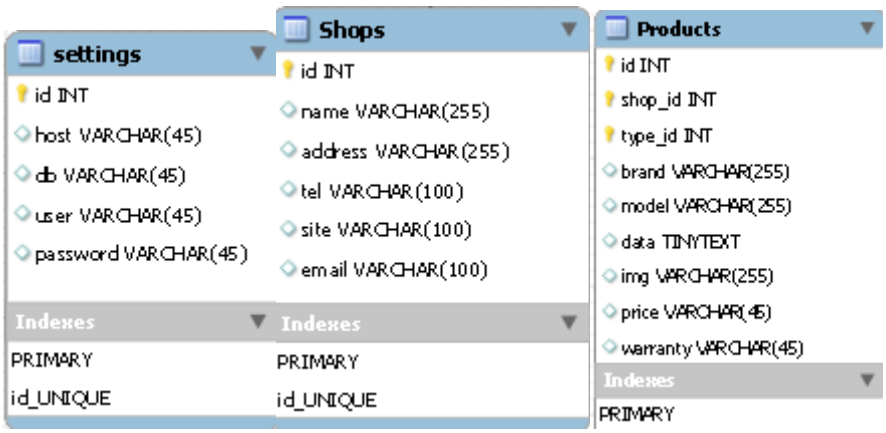


Аналогічним способом створимо таблицю **settings** з налаштуваннями доступу до бази даних ІС, що містить поля **id**, **host** для вказівки імені хоста (адреси сервера), **db** - імені бази даних, **user** і **password** з ім'ям користувача і паролем для установки ІС на віддалений сервер.

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:



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

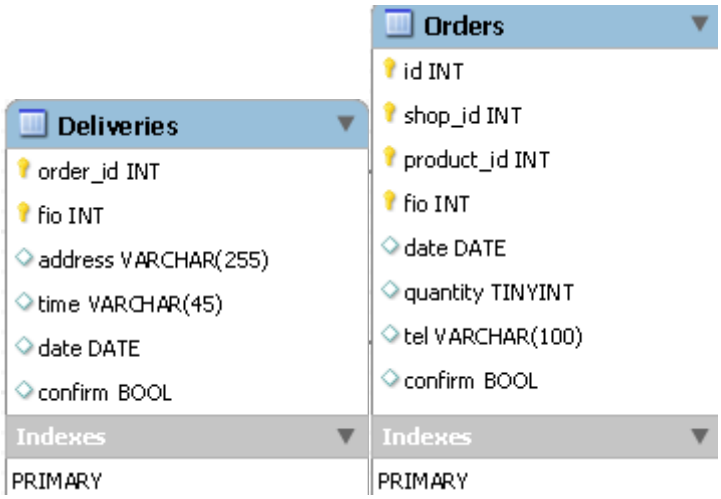
Вид таблиці такий:

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.

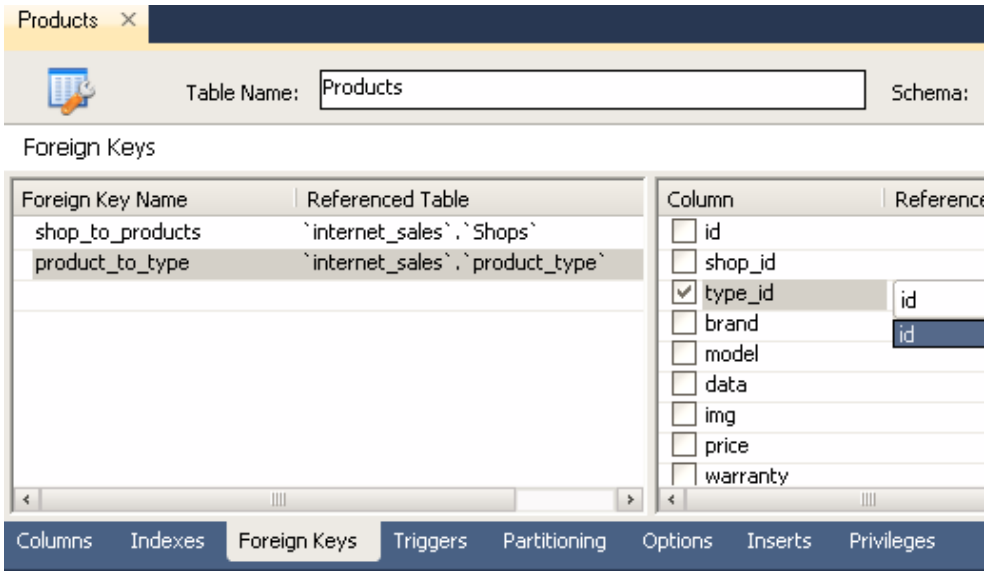
Таблиці **orders** і **deliveries** виглядають наступним чином:



зв'язки таблиць

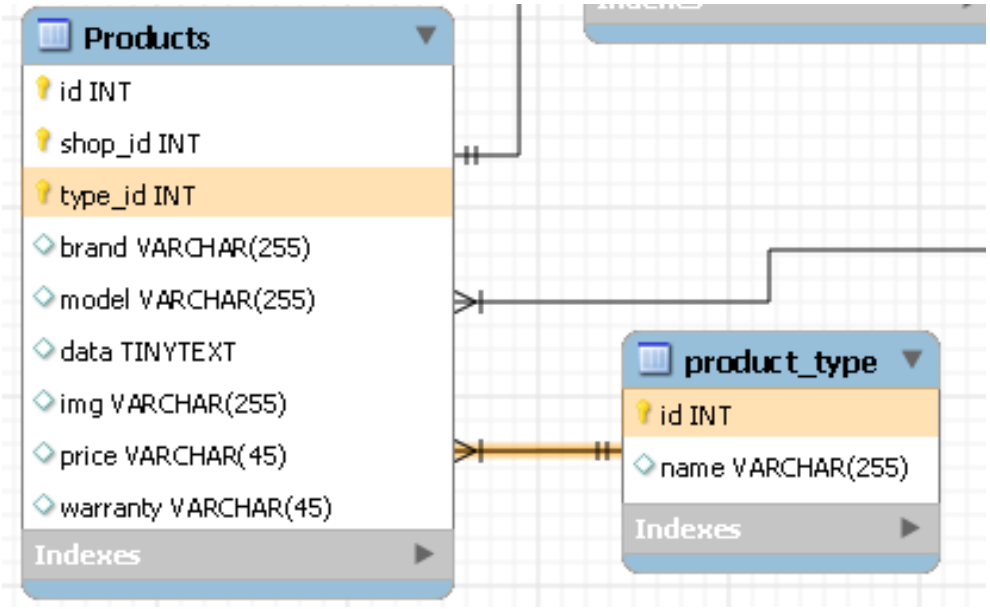
Ми створили базу даних, що складається з семи таблиць, тепер необхідно зв'язати таблиці, ми вже практично розробили основні поля цілочисельного типу, вони і стануть основою для зв'язування.

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:

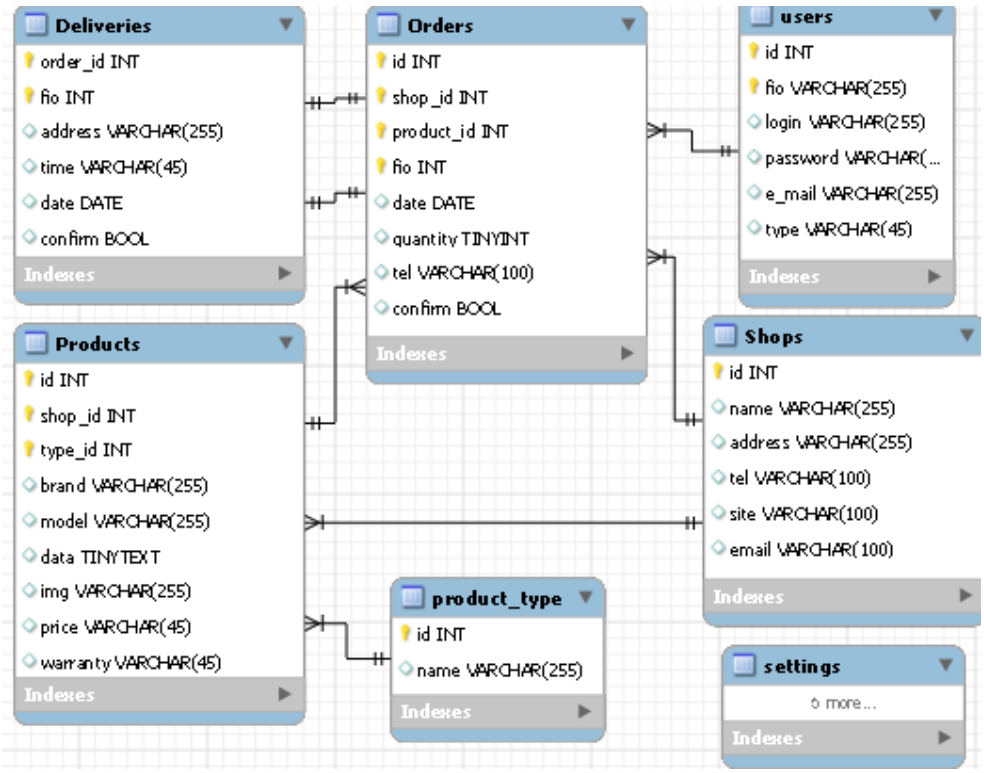


Аналогічним чином пов'язуємо всі ключові поля в таблицях, щоб вони були логічно пов'язані між собою, далі нам необхідно переконатися, що спроектована база даних відповідає третій нормальній формі.

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.

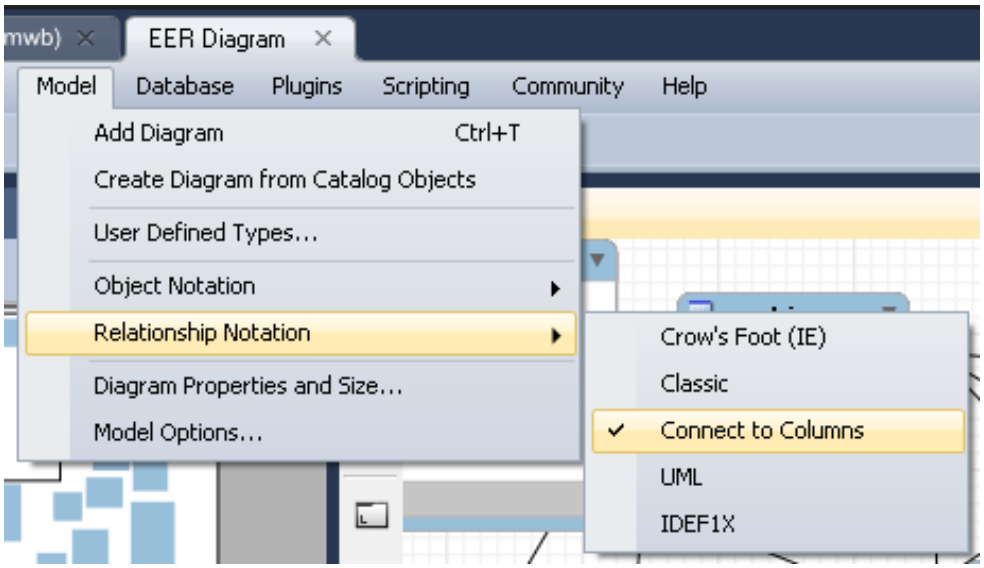
Таким чином, наша база знаходиться в третій нормальній формі, тому що кожен не ключовий стовпець незалежний один від одного. Це наочно видно на діаграмі нашої бази даних:



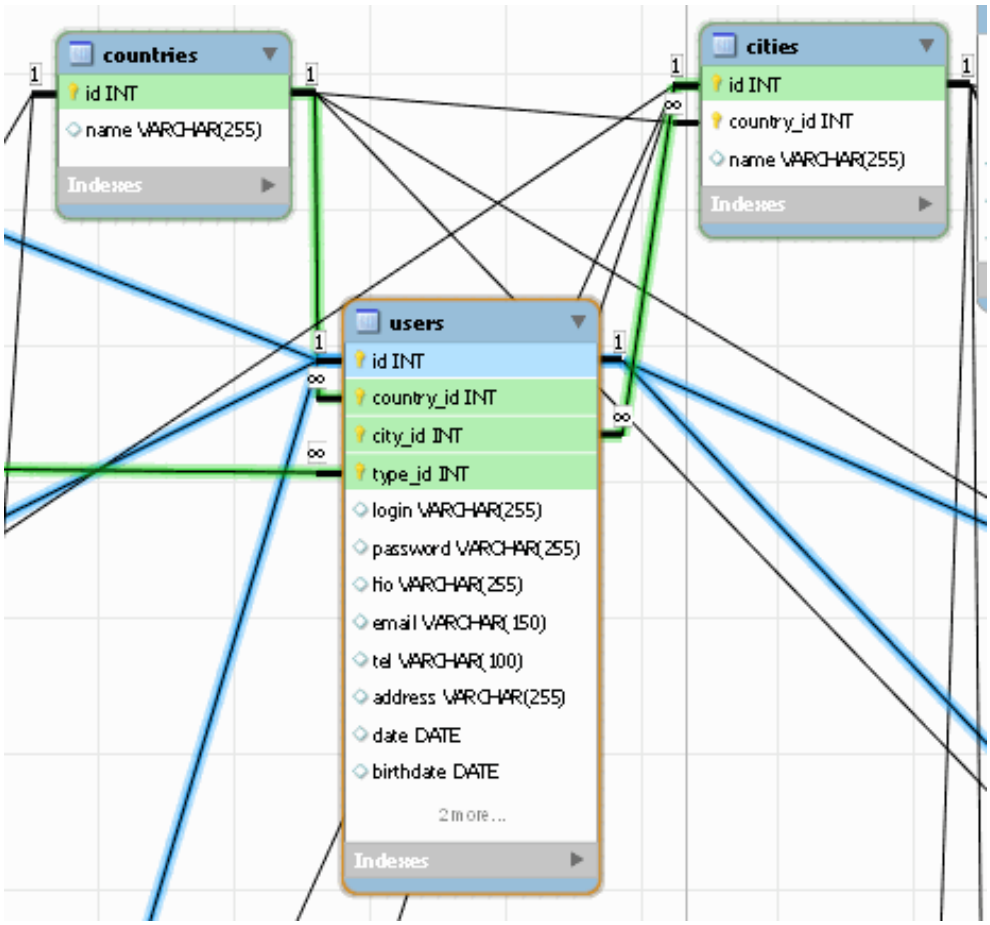
Більшість таблиць знаходяться у відношенні один-до-багатьох, за винятком таблиць **deliveries** і **orders** що у відношенні один-до-одного, тому що доставлений товар може бути тільки в одному замовленні, тобто у одного замовлення тільки одна доставка. Решта зв'язків наочно продемонстровані вище.

Відображення ліній зв'язків таблиць в режимі поле-к-полю

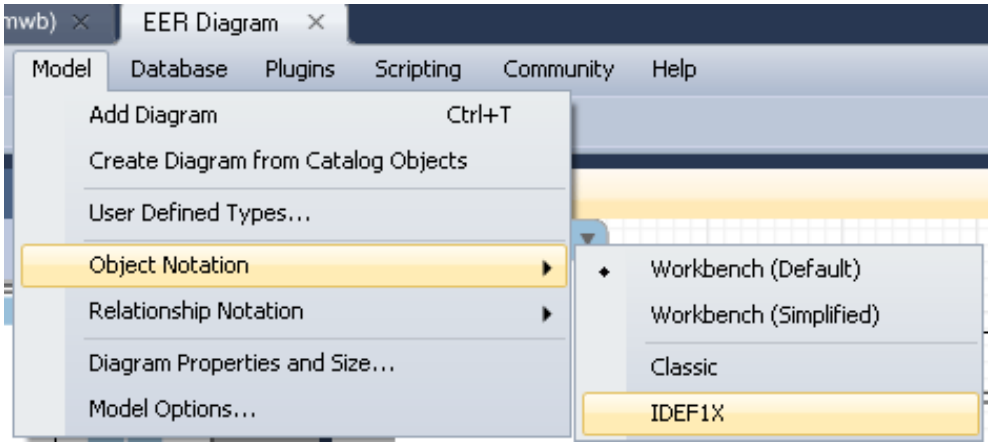
Далі наведено пояснення про те, як змінити вид зв'язків і таблиць, для цього необхідно вибрати наступний параметр в розділі меню **Relationship Notation**:



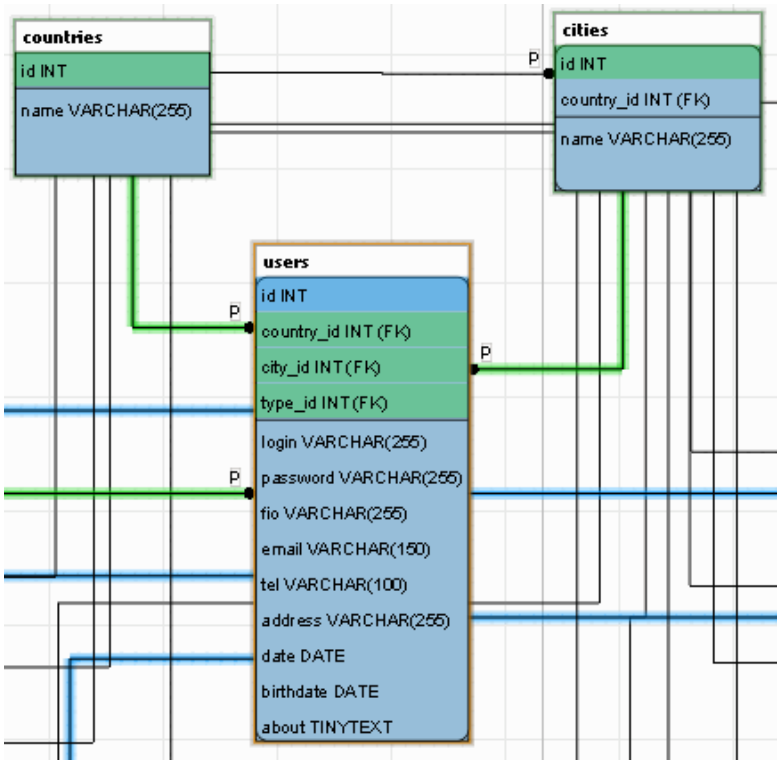
Після цього зв'язки таблиць приймуть вигляд:



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:



ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

Розробити модель даних та таблиці у графічному середовищі

Лабораторна робота №6
Тема: Конструювання бази даних у WorkBench
Мета: Доопрацювання структури бази даних

ТЕОРЕТИЧНІ ВІДОМОСТІ

Створення таблиці

Для створення таблиць використовується команда **CREATE TABLE** . Ця команди 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_стовпцяn тип_даних атрибути_стовпцяn,
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:

```
CREATE DATABASE
productsdb;

USE productsdb;

CREATE TABLE
Customers (
   Id INT,
   Age
   INT,
   FirstName VARCHAR(20),
   LastName VARCHAR(20)
);
```

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.

Перейменування таблиць

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

```
RENAME TABLE старое_название TO
```

Наприклад, перейменуємо таблицю Customers в Clients:

```
RENAME TABLE Customers TO
```

Повне видалення даних

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

```
TRUNCATE TABLE
Clients;
```

Видалення таблиць

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

```
DROP TABLE
Clients;
```

Атрибути полів

За допомогою атрибутів можна налаштувати поведінку стовпців. Розглянемо, які атрибути ми можемо використовувати.

PRIMARY KEY

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

```
USE productsdb;

CREATE TABLE Customers
(
   Id INT PRIMARY KEY,
   Age INT,
   FirstName VARCHAR(20),
   LastName VARCHAR(20)
);
```

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:

```
USE productsdb;
CREATE TABLE
Customers (
   Id  INT,
   Age
   INT,
   FirstName
   VARCHAR(20),
   LastName
   VARCHAR(20),
   PRIMARY KEY(Id)
);
```

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

```
CREATE TABLE OrderLines
(
   OrderId INT,
   ProductId
   INT, Quantity
   INT, Price
   MONEY,
   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.

```
CREATE TABLE Customers
(
   Id INT PRIMARY KEY
   AUTO_INCREMENT, Age INT,
   FirstName VARCHAR(20),
   LastName VARCHAR(20)
);
```

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

На основі логічної моделі БД створити таблиці. Визначити атрибути, встановити зв’язки між таблицями за допомогою SQL-команд

Лабораторна робота №7

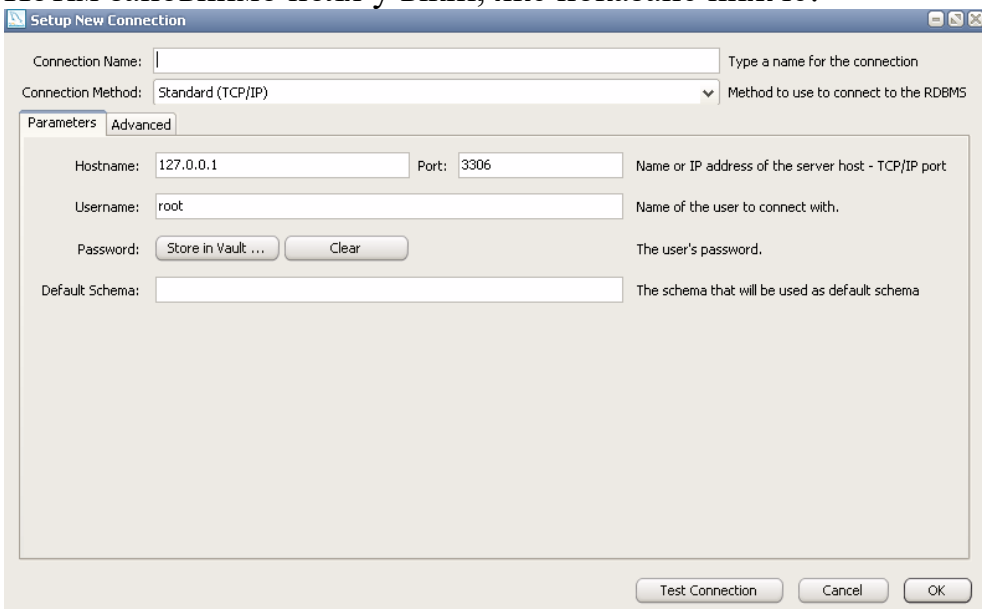
Тема: Завантаження бази даних на сервер

Мета: Навчитись завантажувати базу даних на сервер, навчитись експортувати створену базу даних

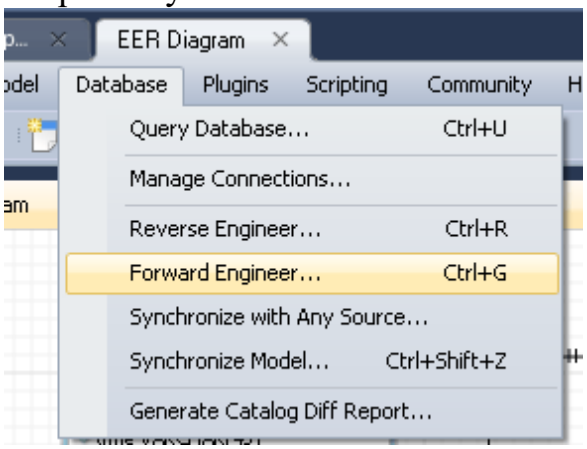
Порядок виконання роботи

Завантажимо нашу базу даних на сервер. Для цього створимо нове підключення до бази даних, клацнувши по посиланню **New connection** в стартовому вікні програми:

Потім заповнимо поля у вікні, яке показане нижче:

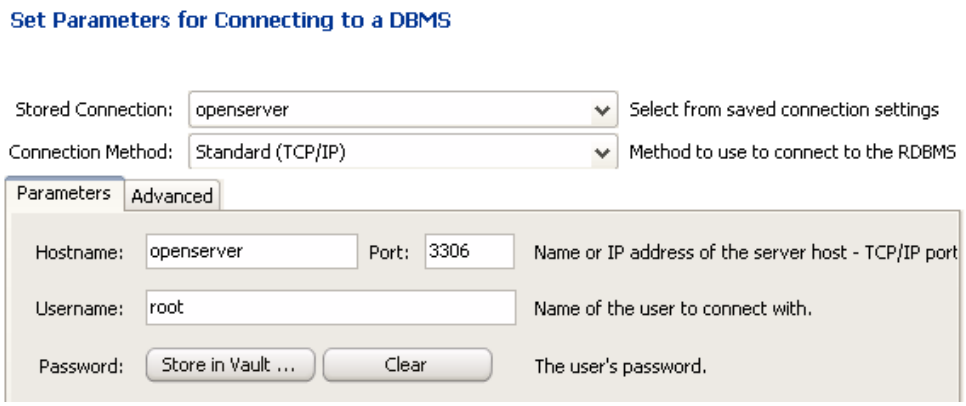


Зазначимо ім'я з'єднання в поле **Connection Name**. Виберемо метод з'єднання в списку **Connection Method**. Задамо ім'я хоста і порт у вкладці **Parameters**. Вкажемо ім'я користувача і пароль, якщо він є і натиснемо на кнопку ОК. Потім відкриємо вкладинку **EER Diagram**. На панелі **EER Diagram** оберемо пункт **Database** і натиснемо на параметр **Forward Engineer**:



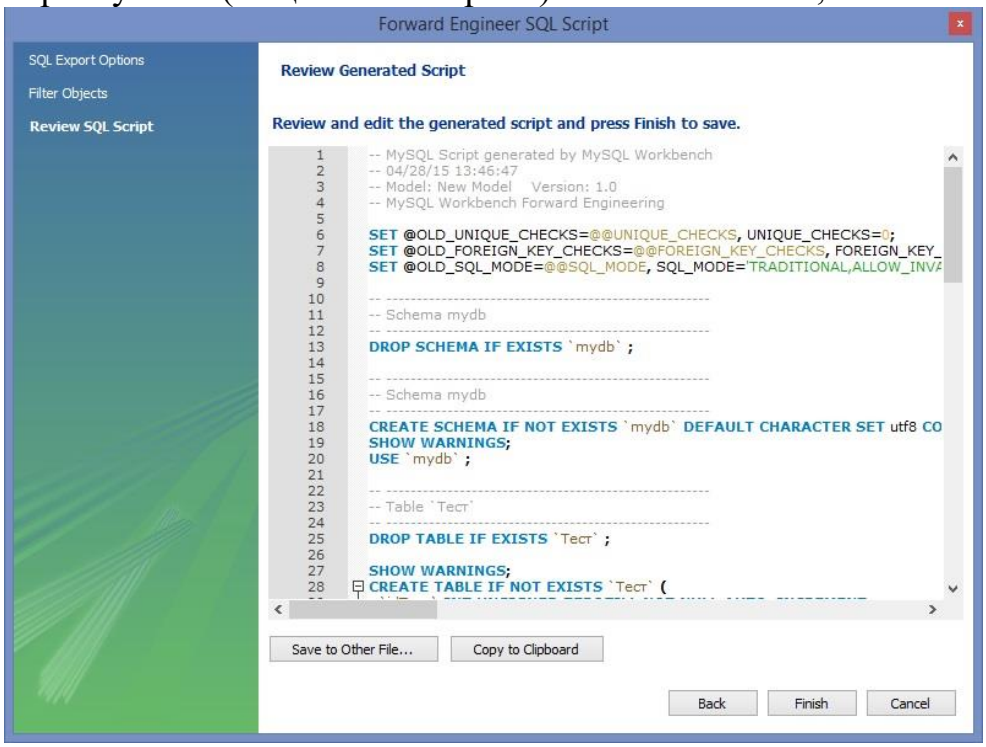
Після того як з'явиться вікно, натискаємо на кнопку **"Next"**. Вибираємо параметр **Export MySQL Table Objects** і натискаємо на кнопку **"Next"**:

Після натискання кнопки з'явиться вкладка з SQL кодом, можна зберегти його натиснувши кнопку **"Save to file"**якщо це необхідно, а потім натиснути на кнопку **"Next"**. З'явиться вікно з параметрами з'єднання:



Перевіряємо, чи правильні параметри підключення і натискаємо на кнопку **"Execute"**, Якщо в SQL коді не міститься помилок, то після виконання коду ми побачимо вікно зі списком таблиць, інакше виведеться повідомлення про помилку. Тепер наша база завантажена на сервер.

Коли Ви закінчите створення структури, ви можете створити базу даних SQL просто експортувавши її. 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.



ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

Завантажити базу даних на сервер, експортувати створену базу даних.

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. Бережани : Бережанський 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

ОРГАНІЗАЦІЯ БАЗ ДАНИХ

Методичні вказівки до виконання практичних робіт

Укладач:
Міхнова Олена Дмитрівна

Формат 60х84/16. Гарнітура Times New Roman Папір для цифрового друку.
Друк ризографічний.
Ум. друк. арк. 1,1.
Наклад пр.
Державний біотехнологічний університет61002, м. Харків, вул. Алчевських, 44