# Лабораторна робота № 1

Адміністрування баз даних SQL Server

# Завдання 0. Встановлення та налаштування MYSQL Server

За рекомендаціями наведеними <http://www.php.su/prepare/?mysql_install_win> завантажити, встановити та налаштувати MYSQL Server.

Завантажити та встановити MY SQL Workbench.

Завантажити та встановити MY SQL Connector.

# Завдання 2. Створити БД за індивідуальним варіантом.

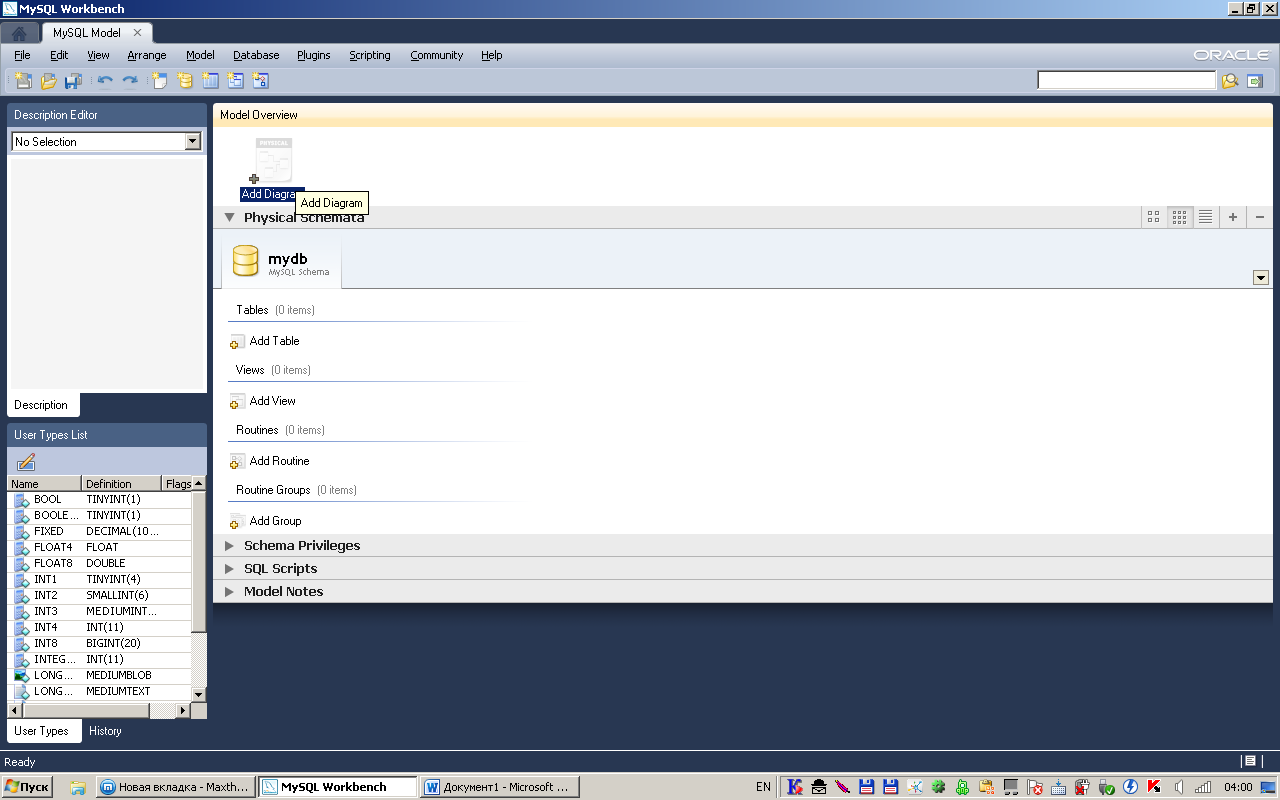
Створити БД, таблиці, встановити між ними зв’язки, вставити записи та реалізувати запити. Один запит реалізуватия як представлення, та один процедуру, що зберігається із параметрами.

Приклад:

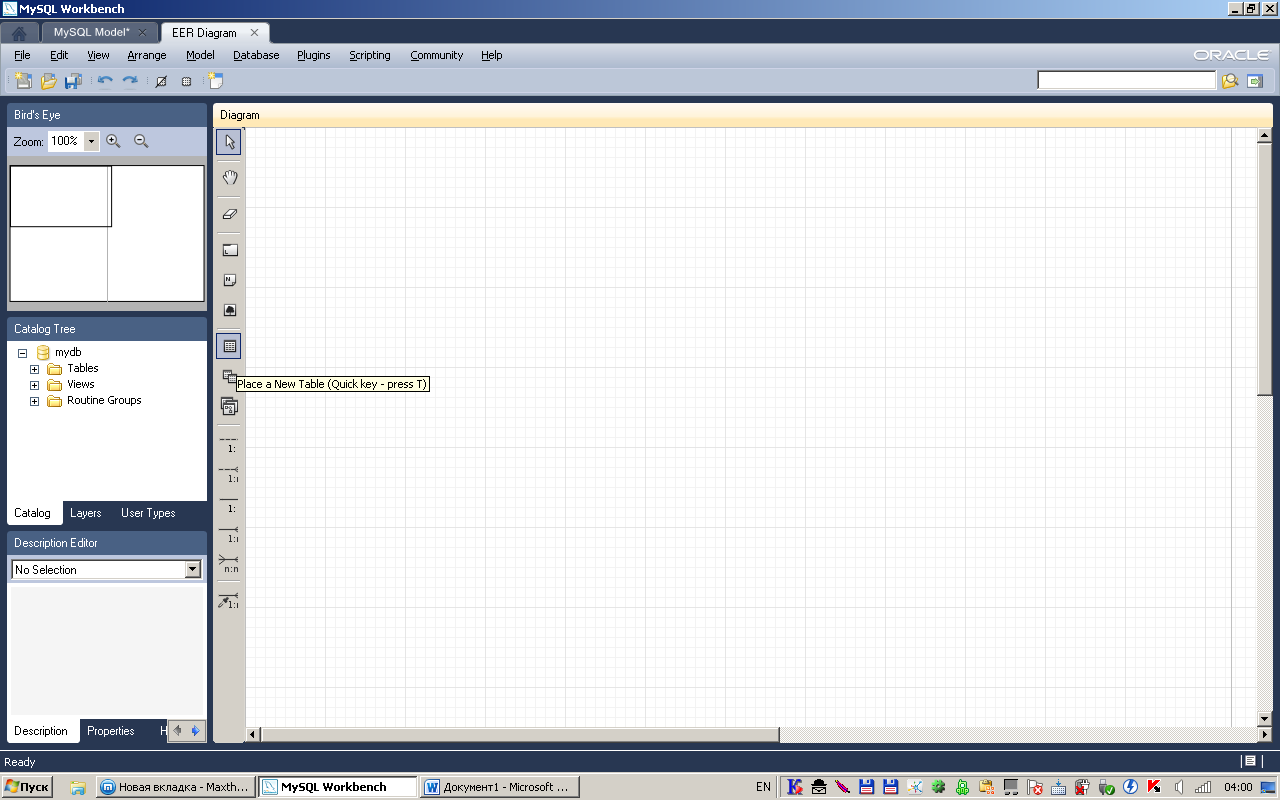
1. Для концептуального моделювання схеми БД в MSSQL Workbench необхіідно обрати пункут Create New EER Model у розділі Data Modelling:



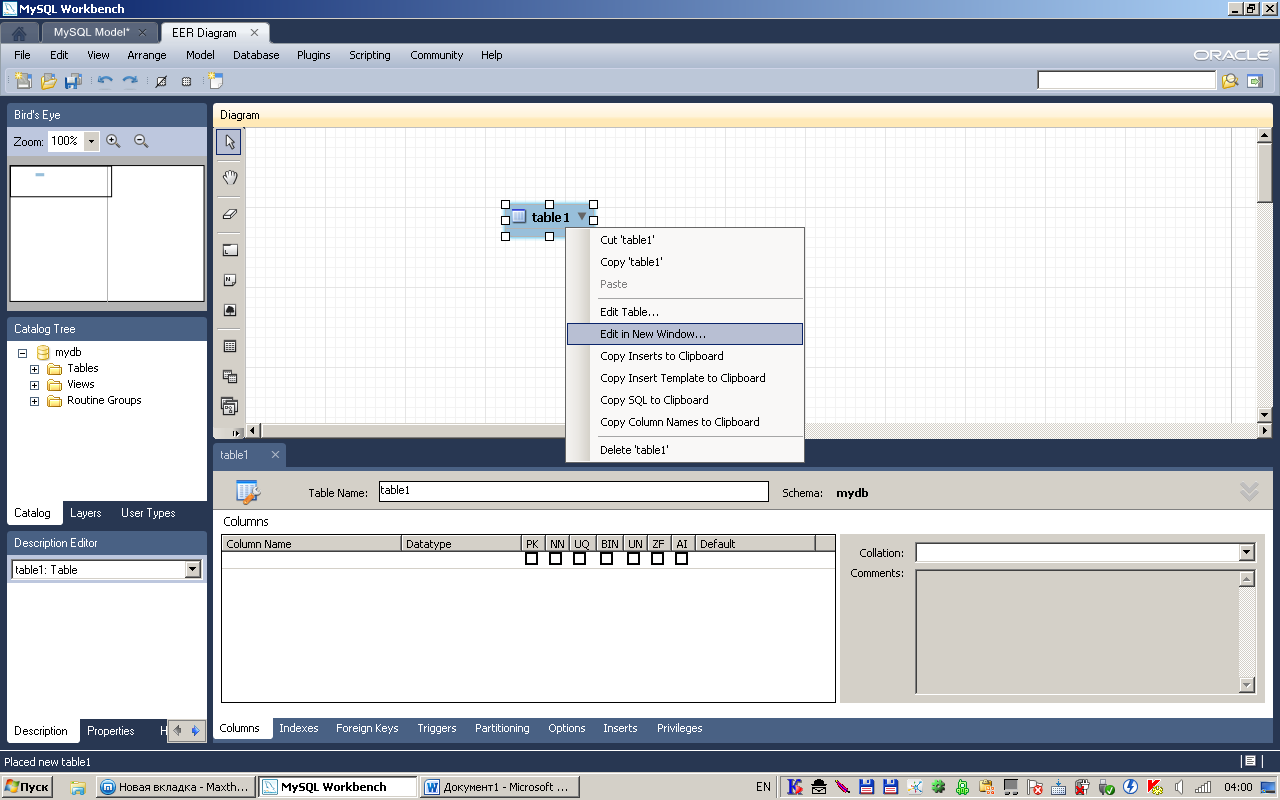
2. Для створення нової схеми даних необхідно додати нову діаграму командою Add Diagramm:



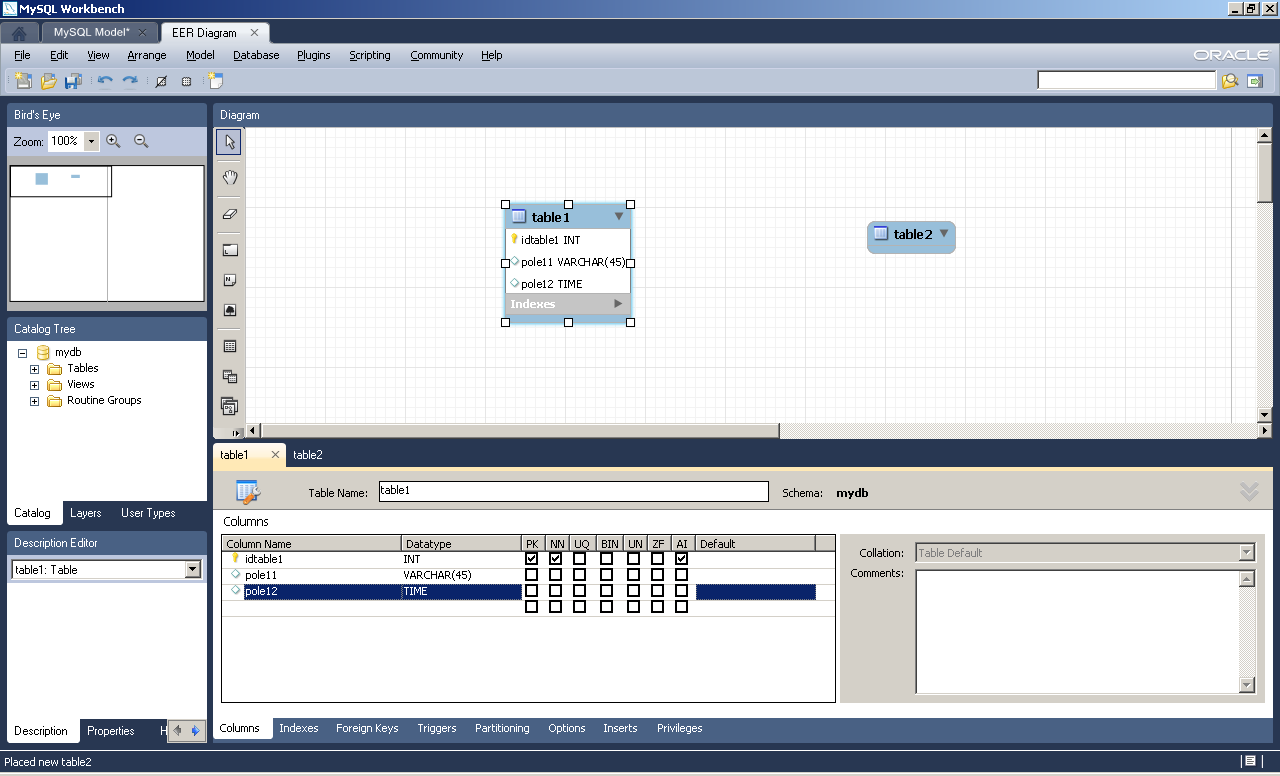
3. Для визначення сутностей, що моделюються, на панелі інструментів використаємо відповідний об’єкт:



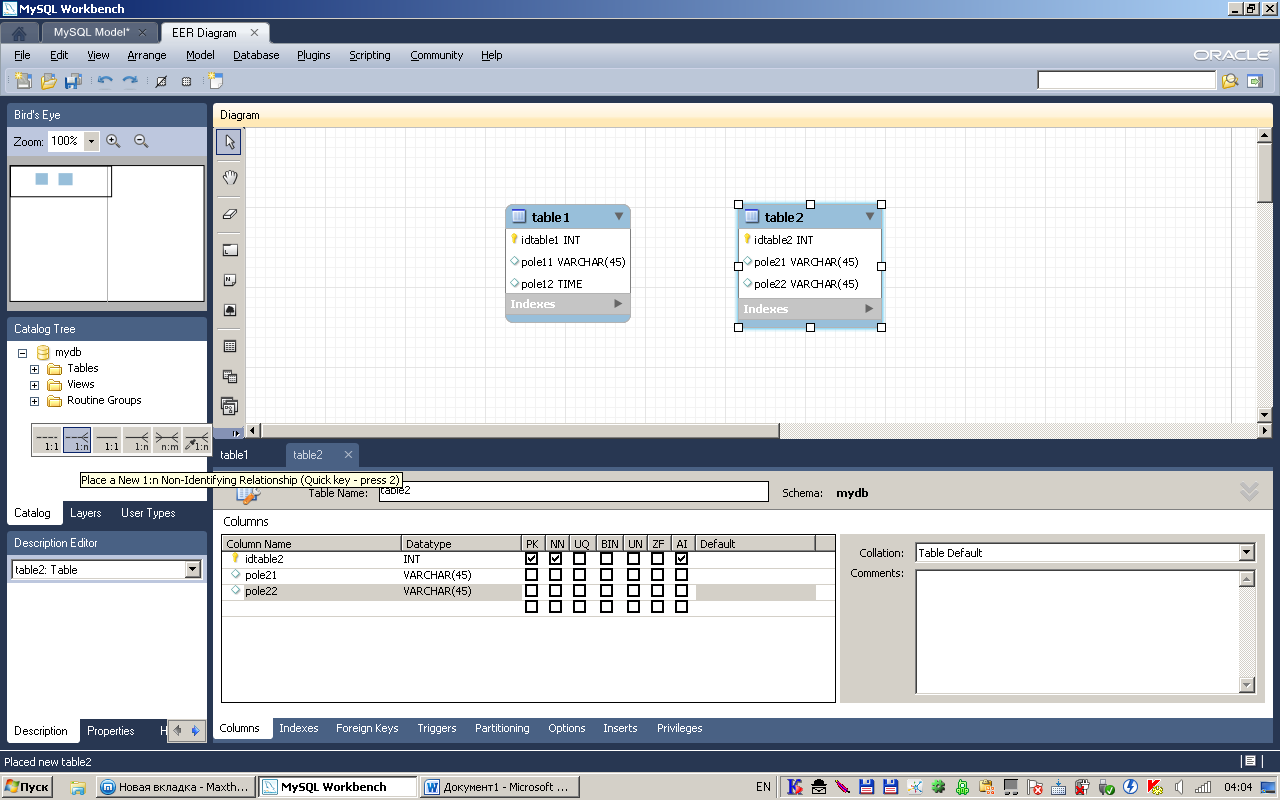
4. Для редагування структури сутностей та визначення атрибутів використаємо контекстне меню, пункт Edit Table:



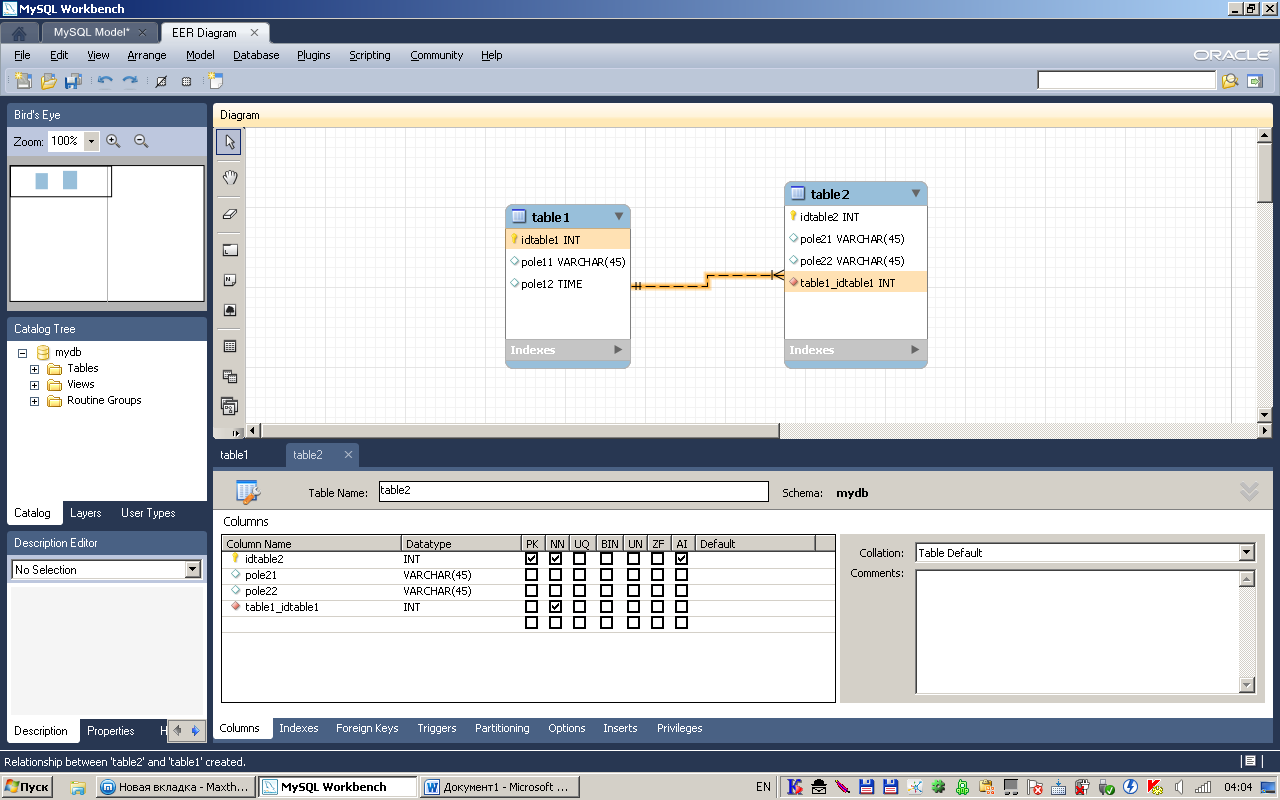
5. Під час редагування задаємо параметри: тип даних, первинний ключ, підтримка Null та інші:



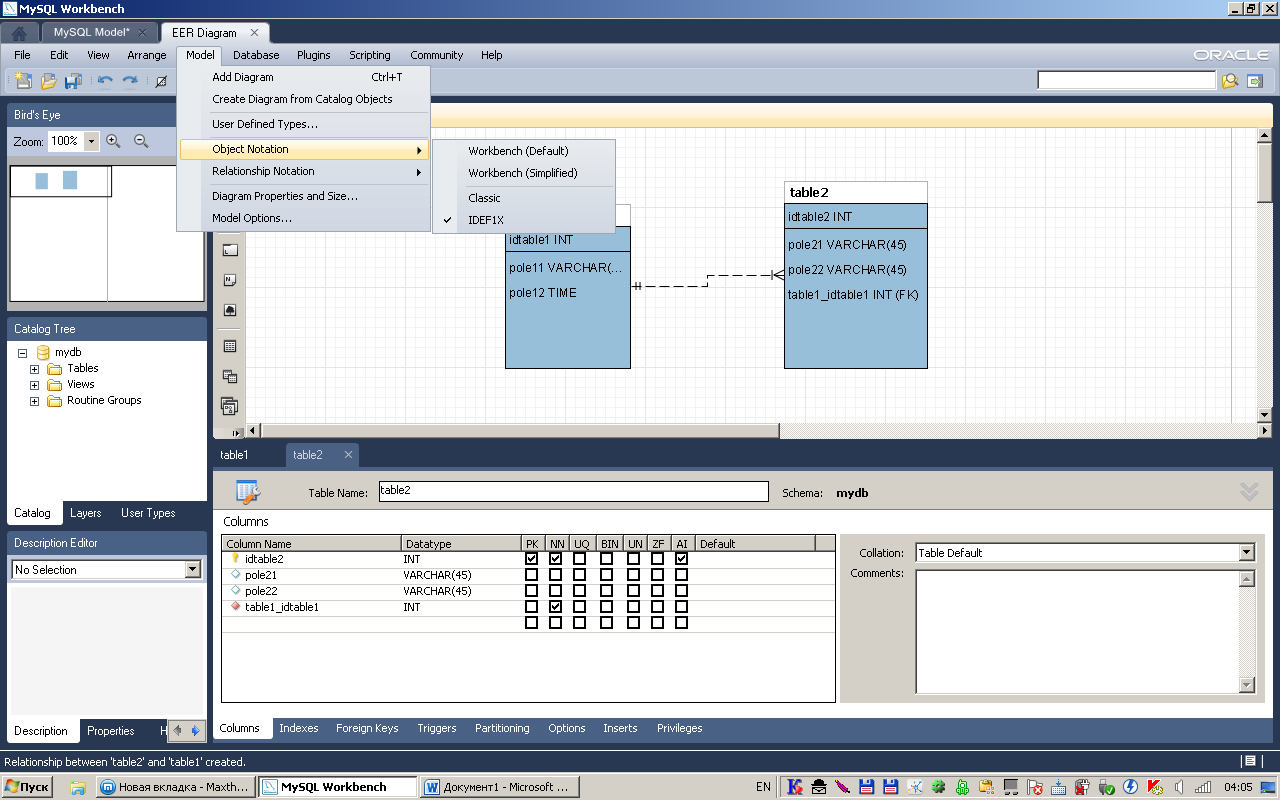
7. Аналогічним чином повторюємо дії 3-5 для інших об’єктів. Для визначенн зв’язків між створеними об’єктами оберемо тип зв’язку на панелі інструментів:



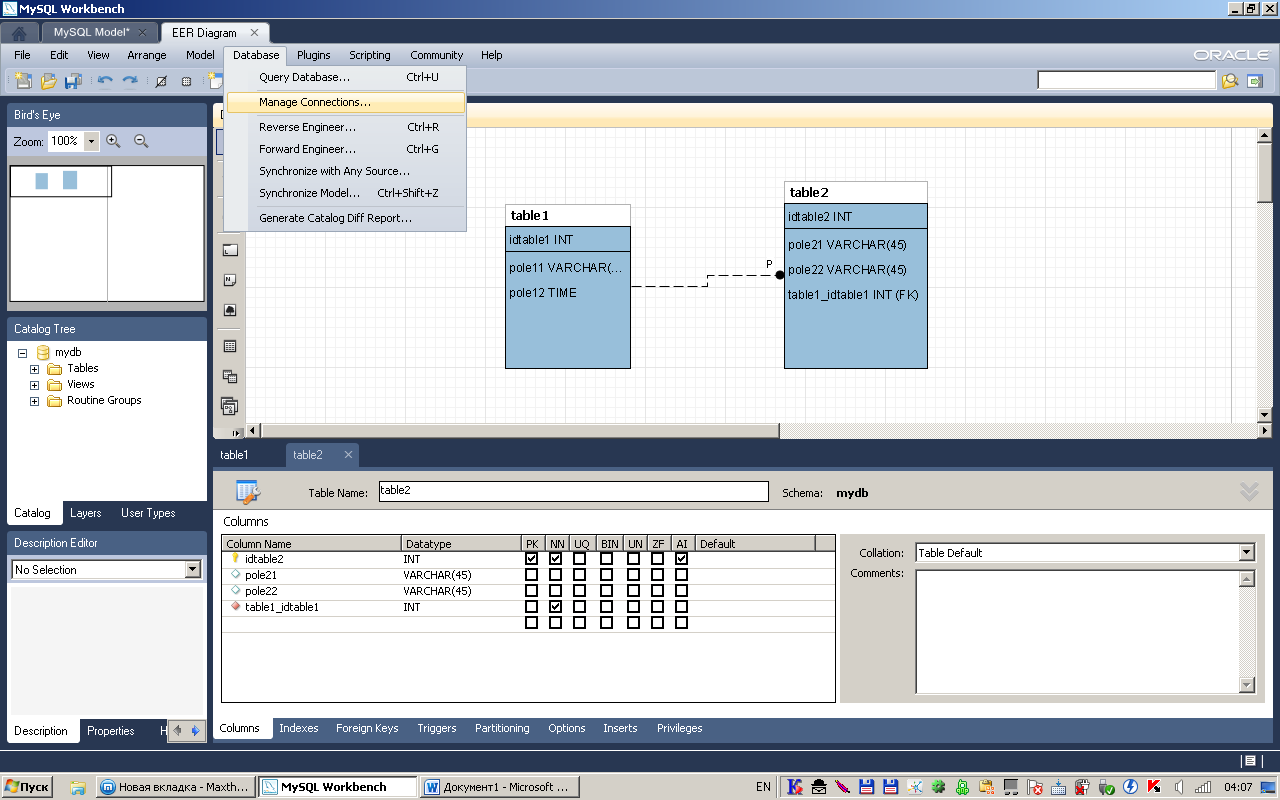
8. Зветніть увагу, напрямок зв’язку визначається від дочірньої до батьківської сутності (тобто при встановленні спочатку виділяємо підлеглу таблицю, потім головну). Результат наведено на рисунку:



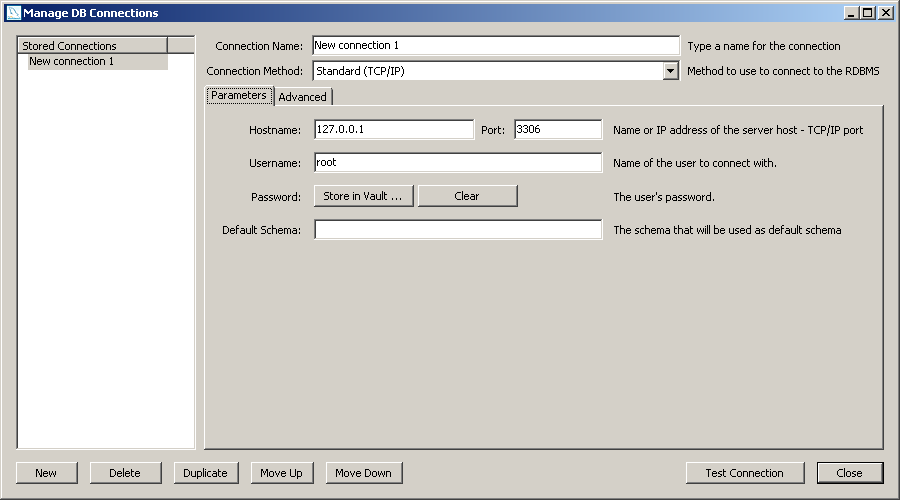
9. Якщо дана форма відображення є незручною для сприйняття її можна замінити на звичну у форматі IDEF 1X:



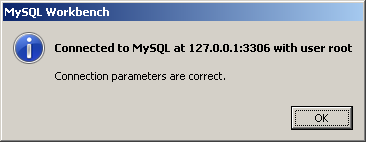
10. Для генерації БД, необхідно під’єднатись до сервера. Порожню базу при цьому створювати не потрібно. БД буде створено автоматично, назва нової БД збігаєтсья з назвою моделі:



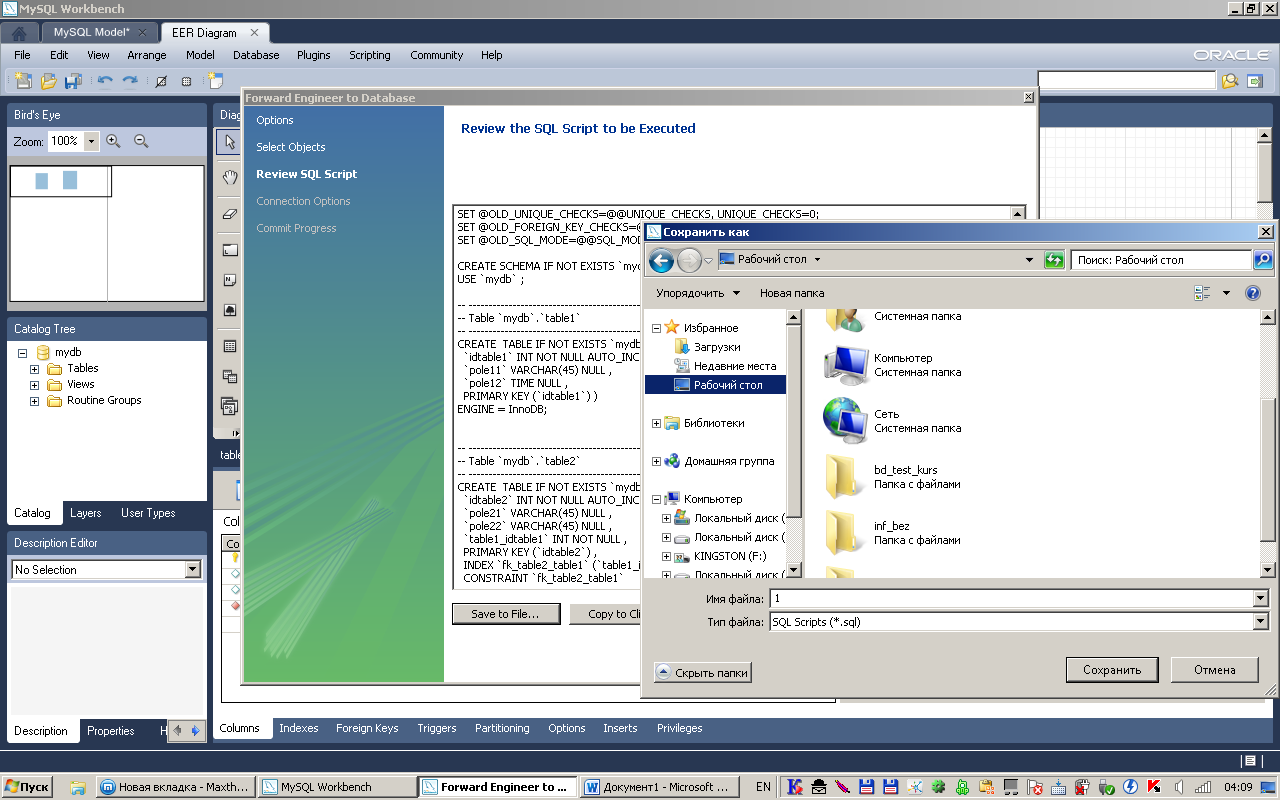
11. За необхідності параметри можна змінити:



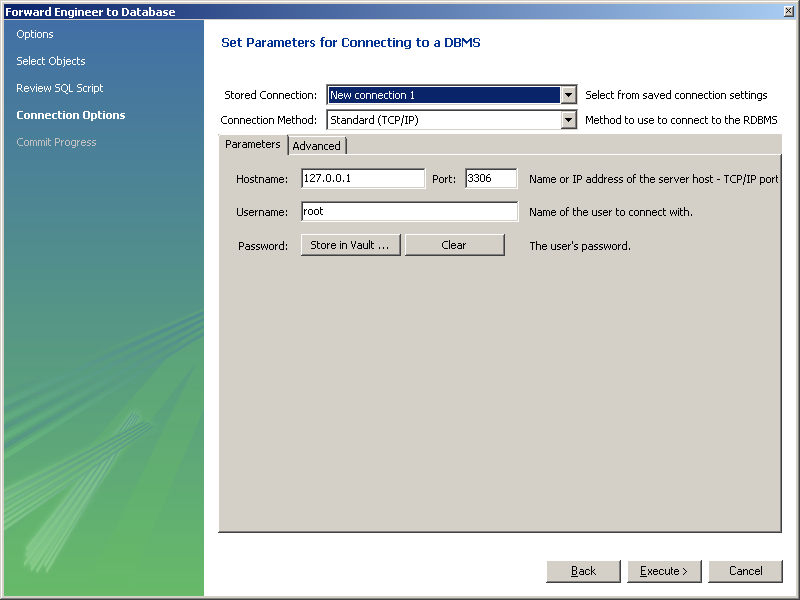
12. При успішному під’єднанні буде висвітлене відповідне повідомлення.



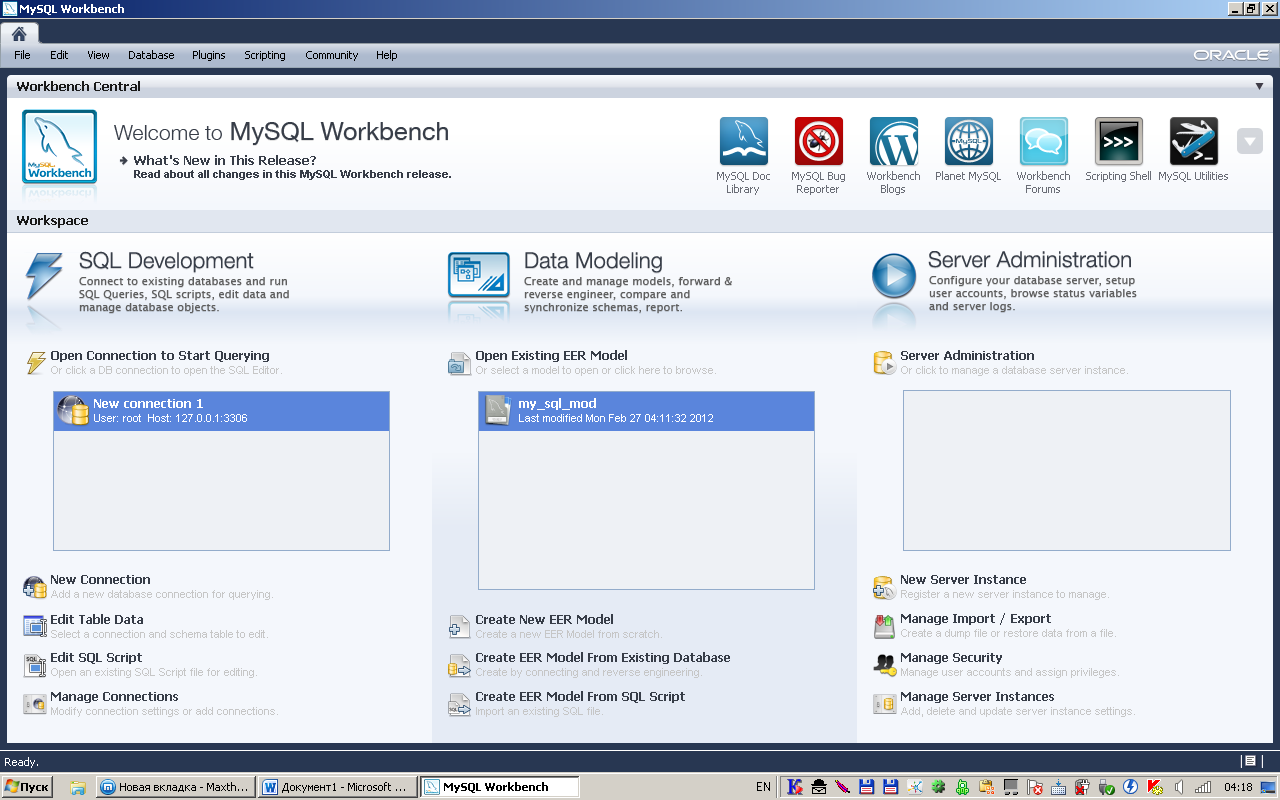
13. Для подальшої роботи під час генерації доцільно зберегти скріпти якими виконувалось створення об’єкту.



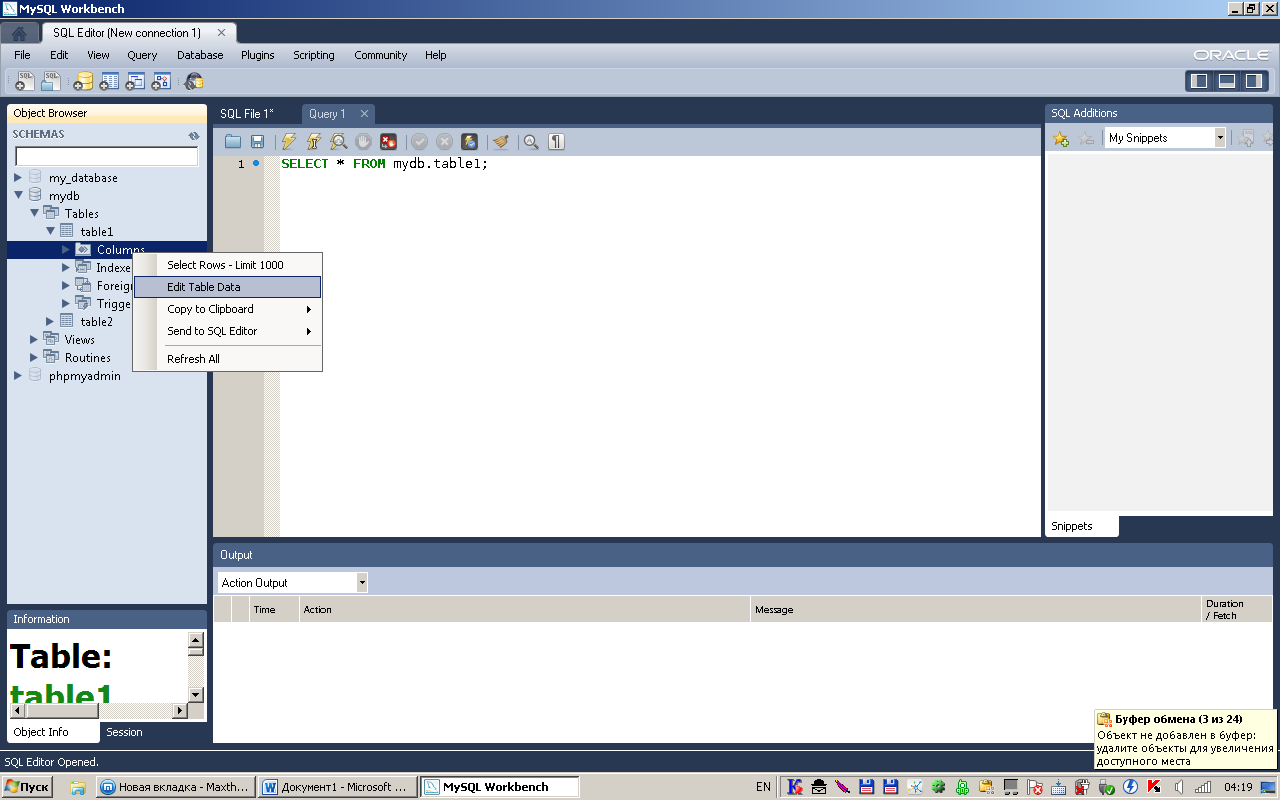
14. Вибрати створене з’єднання:



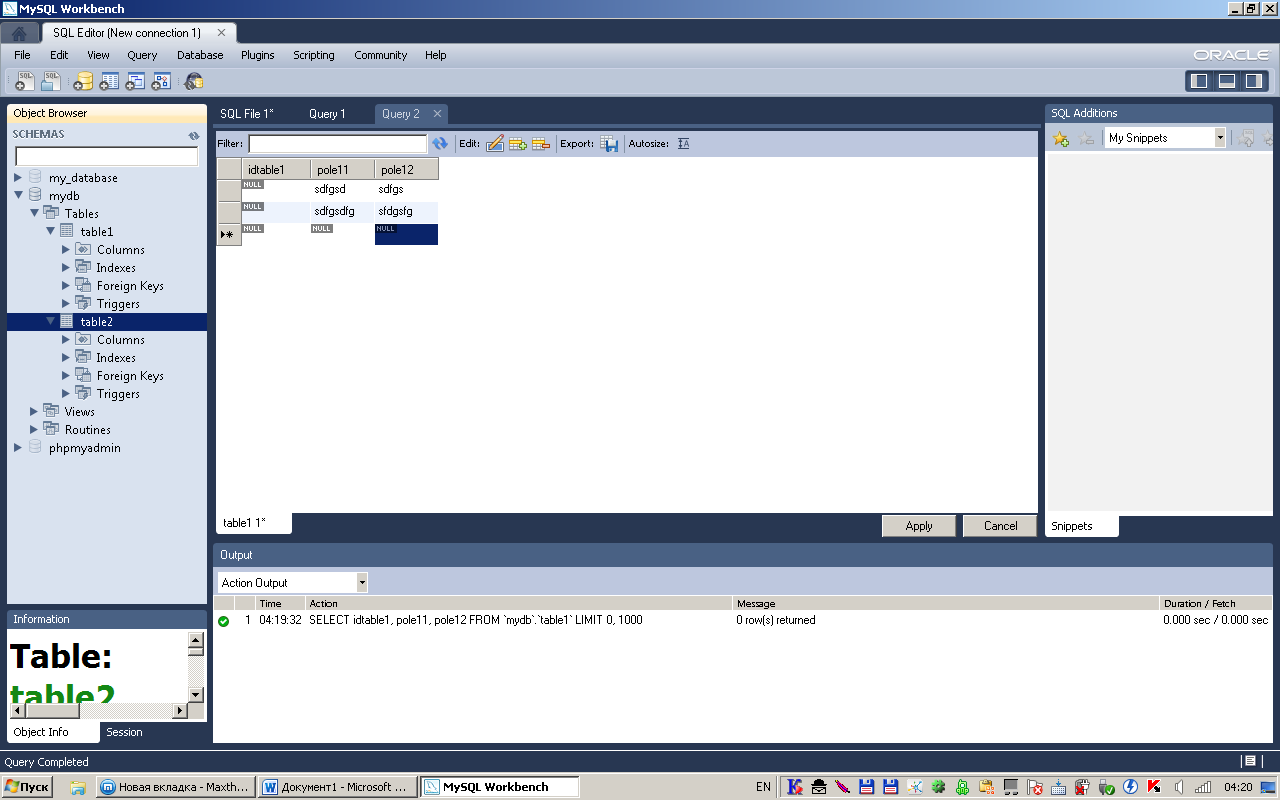
14. Переглянути результати можна відкривши з’єднання у області SQL Development.



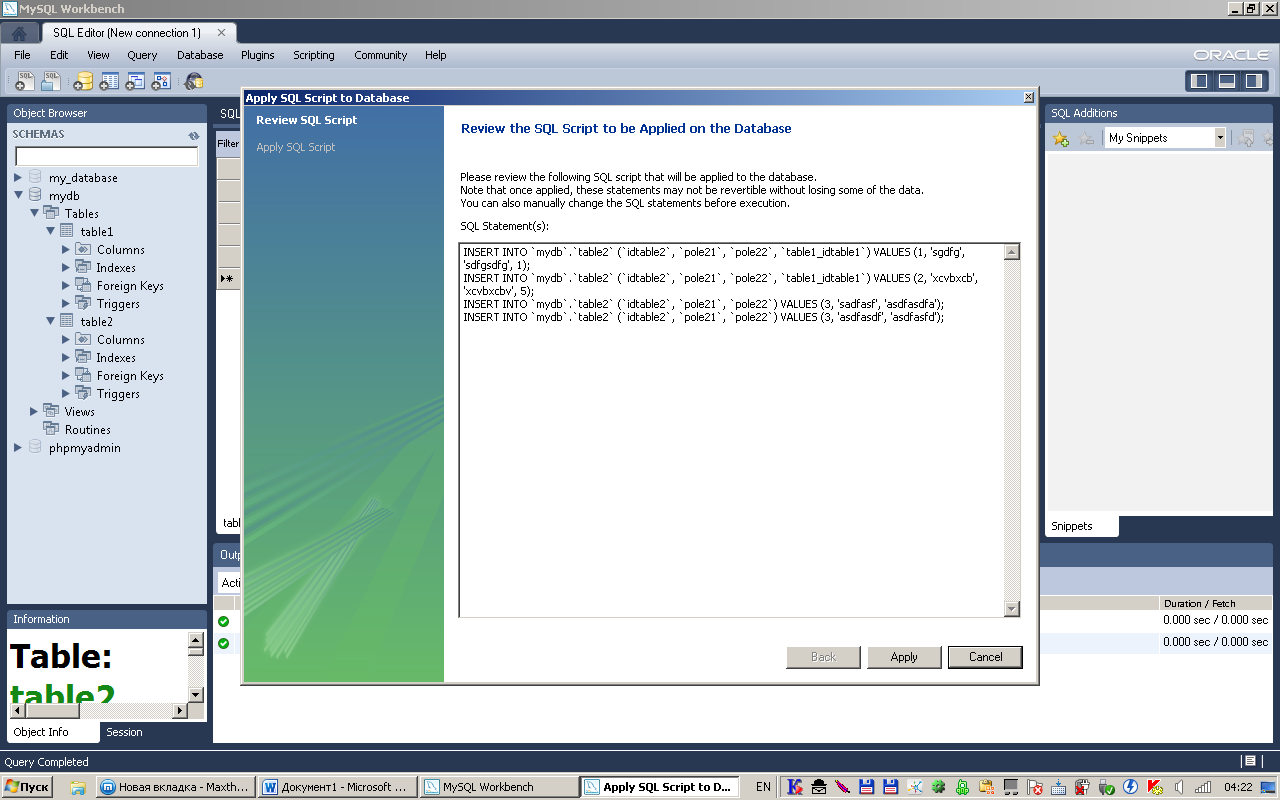
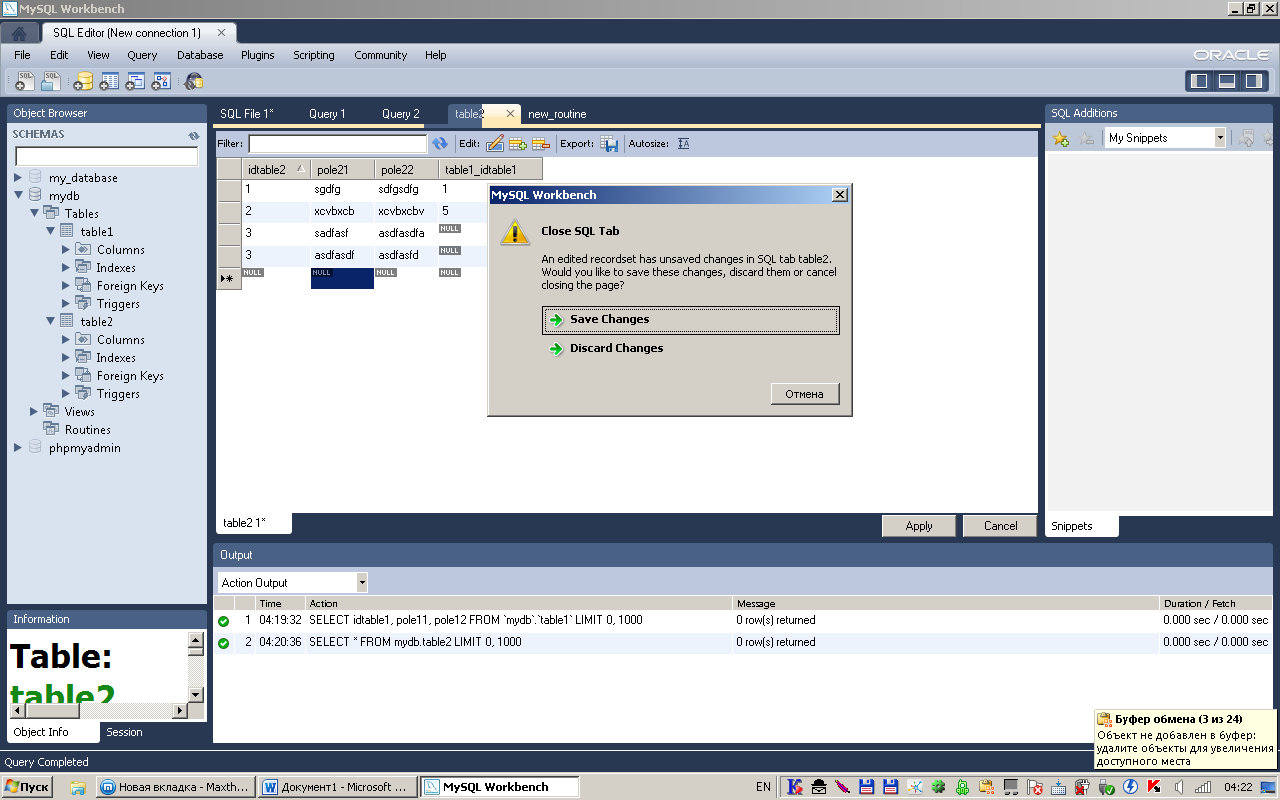
15. Для тестування коректності побудови БД потрібно ввести не менше 3-х записів до кожної таблиці:



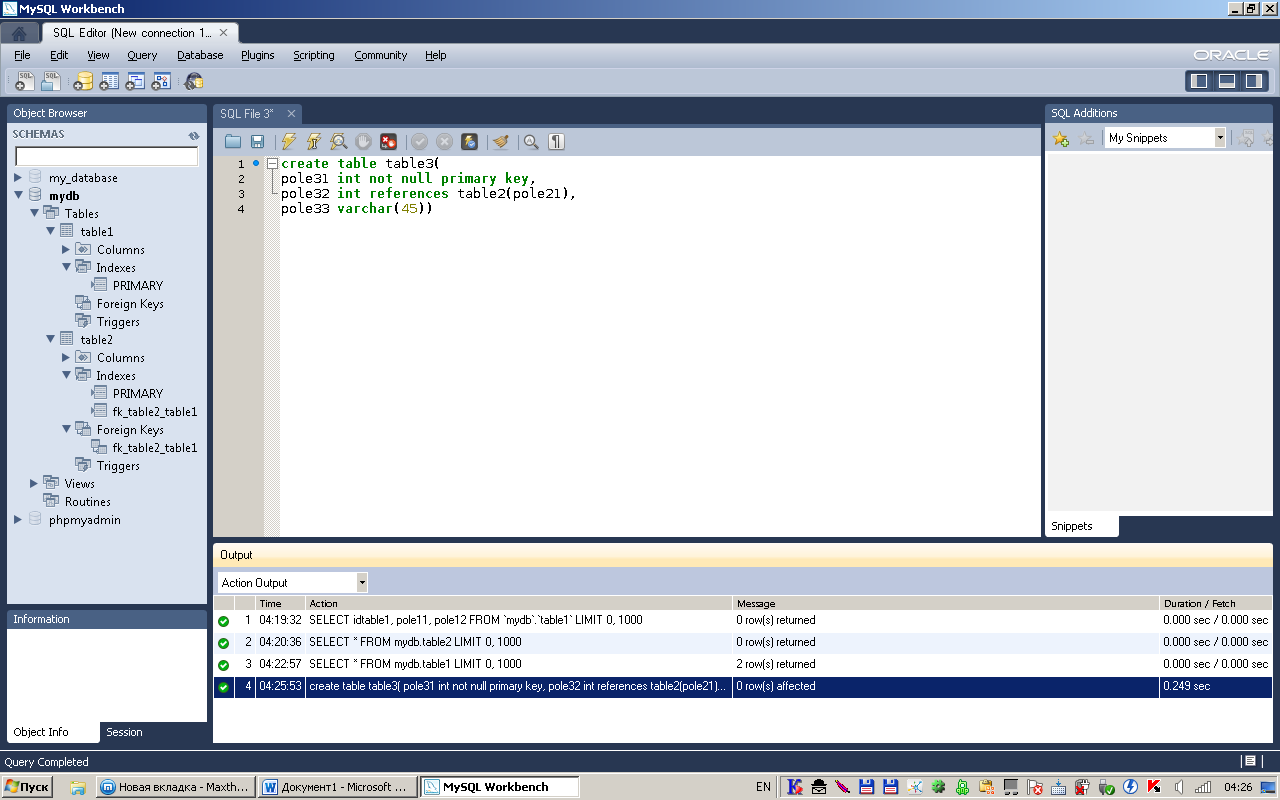
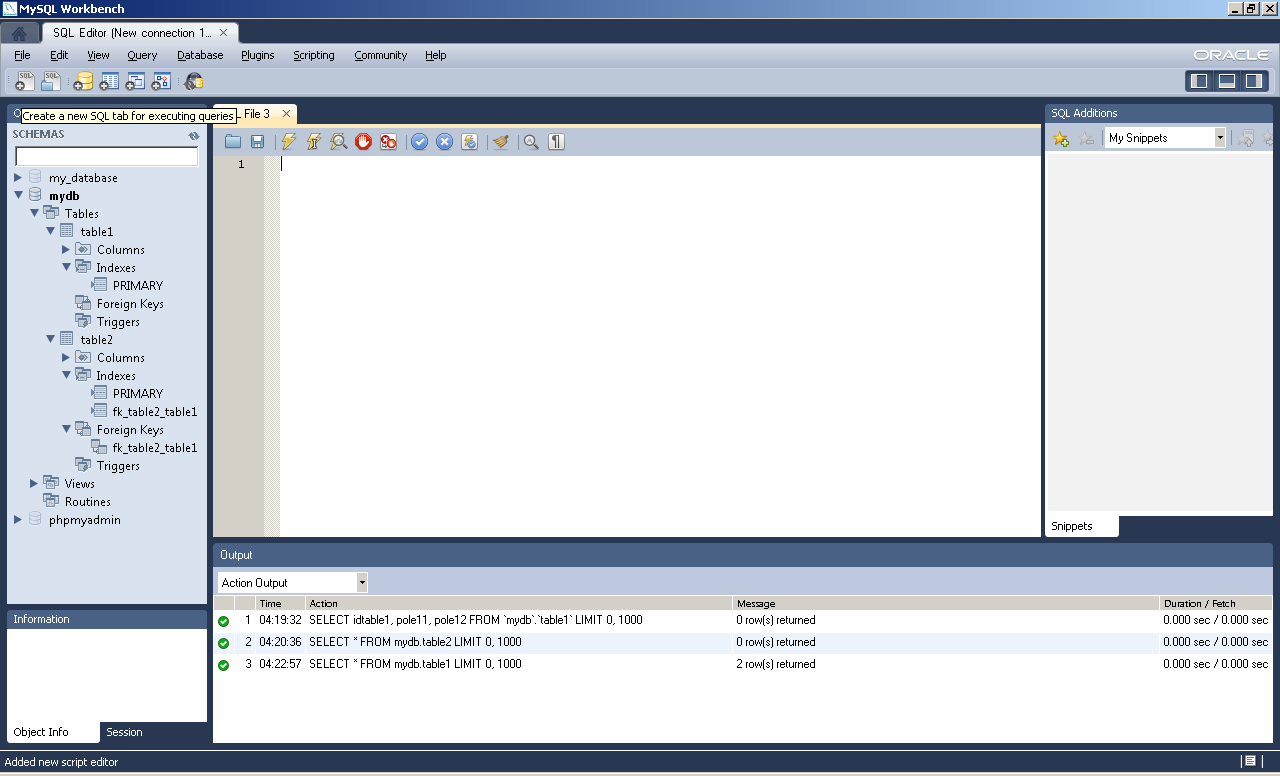
17. Підтвердження введення запису – кнопка Apply, або записи буде збережено автоматично під час закриття об’єкту (особливості клієнт-серверної архітектури).



18. Якщо дані суперечать правилам підтримки цілісності даних, то виникне повідомлення про помилку та відповідний звіт:



19. Для створення таблиць вручну можна використати вікно запитів:



Завдання 2. Створити БД за наведеним завданням використавши визначення на мові SQL:

Побудувати БД що складається з двох таблиць: Дзвінки з атрибутами номер телефону, дата, час дзвінка, тривалість дзвінка та Тарифи з атрибутами – тип дзвінка, вартість за 1 хв. Встановити між таблицями реляційний зв‘язок.

Приклад:

CREATE TABLE table1 (

`idtable1` INT NOT NULL AUTO\_INCREMENT ,

`pole11` VARCHAR(45) NULL ,

`pole12` TIME NULL ,

PRIMARY KEY (`idtable1`) )

CREATE TABLE IF NOT EXISTS `mydb`.`table2` (

`idtable2` INT NOT NULL AUTO\_INCREMENT ,

`pole21` VARCHAR(45) NULL ,

`pole22` VARCHAR(45) NULL ,

`table1\_idtable1` INT NOT NULL ,

PRIMARY KEY (`idtable2`) ,

INDEX `fk\_table2\_table1` (`table1\_idtable1` ASC) ,

CONSTRAINT `fk\_table2\_table1`

FOREIGN KEY (`table1\_idtable1` )

REFERENCES `mydb`.`table1` (`idtable1` ))

Для встановлення зв’язків можна також використовувати запит:

ALTER TABLE *Table\_name1*  ADD FOREIGN KEY (*Table\_name1.FK\_name*) REFERENCES *Table\_name1*  (*PK\_name*).

Приклад створення процедури, що зберігається

delimiter |

create procedure reorder (in grNazva varchar(10), out result varchar(10))

begin

declare a, b, c int;

declare curs cursor for select id\_st from student inner join groupa on student.id\_gr=groupa.id\_gr

where groupa.GoupNazva = grNazva order by Sname\_st;

declare continue handler for not found set c = 0;

open curs;

set a=1;

set c=1;

while c=1 do

fetch curs into b;

update student set N\_ingroup = a where id\_st = b;

set a=a+1;

end while;

close curs;

select \* from student;

set result="ura";

end

Для того, щоб викликати процедуру необхідно використати наступний запит:

call reorder ('PI-47', @pr);

select @pr;

# Завдання 2. Управлінння доступом до БД

* Створити матрицю доступу до БД за метою курсової роботи.
* Для визначених користувачів (не менше 2 типів):
  + Створити обліковий запис та ім’я входу користувач SQL Server
  + Визничити для даного логіну користувача сервера БД з правами на створення нових баз даних
  + Перевірити наявність даного користувача
  + Для тестової БД даних створити користувача з правами на модифікацію схеми даних та виконання операцій вставки та перегляду таблиць.
  + Перевірити надані права доступу.

Завдання виконувати із використанням SQL запитів GRANT, REVOKE та інструкцій ІNSERT, UPDATE для таблиць HOST, USER, DB

Перед реалізацією завдання ознайомитись з матеріалами наведеними в розділах [4.2 Общие проблемы безопасности и система привилегий доступа MySQL](http://www.php.su/mysql/manual/?page=Privilege_system) - <http://www.php.su/mysql/manual/?page=Privilege_system>

та [4.3 Управление учетными записями пользователей MySQL](http://www.php.su/mysql/manual/?page=User_Account_Management) - <http://www.php.su/mysql/manual/?page=User_Account_Management>

Додаткова інформація про користувачів таблиці User та вибір прав користувача для поточного сеансу <http://winitpro.ru/index.php/2016/04/19/vypolnenie-mysql-zaprosov-iz-powershell/>.

**Приклади до опрацювання**

**Створення матриці доступу**

Найбільші права доступу до даних БД мають адміністратори. Для них необхідно призначити стандартні серверні ролі та ролі БД.

Для групи користувачів відносяться керівники підрозділів. Для цієї групи необхідно передбачити можливість передавати їх права та привілеї. Користувачі типу Wokers мають обмежений доступ до об’єктів БД. Більш детально перелік об’єктів БД, доступ до яких надано користувачам відділу (на прикладі відділу продаж) наведено в табл. 1.1. На перетині рядків і стовпців зазначено дії, які може виконувати користувач даного типу ролі: 0 – нема доступу, 1 – перегляд даних, 2 – редагування, 3 – видалення, 4 – вставка даних, 5 - повний доступ(виконання процедури).

Таблиця 1.1

Матриця доступу для співробітників відділу продажу

|  |  |  |  |
| --- | --- | --- | --- |
|  | Керівник відділу | Менеджер | Комірник |
| Товари | 5\* | 1 | 1,2,3 |
| Замовлені товари | 5\* | 5 | 1,2 |
| Клієнти | 1\*, 2\*, 3 | 1,2,3 | 1 |
| Доступ до стовпців (адреса, телефон, примітка) | 5\* | 5 | 2 |
| Працівники | 1\* | 1 | 1 |
| Замовлення | 5\* | 5 | 1 |
| Постачальники | 1 | 1 | 1 |
| Процедура «Періоцінка» | 5 | 0 | 0 |
| Процедура «Залишки» | 5 | 0 | 5 |

### Створення користувачів

Користувачів можна додавати двома різними способами - за допомогою команди GRANT або безпосередньо в таблиці призначення привілеїв MySQL. Бажано використовувати команду GRANT - цей спосіб простіше і дає менше помилок.

Існує також велика кількість програм (таких як phpmyadmin), які служать для створення і адміністрування користувачів.

У наведених нижче прикладах демонструється, як використовувати клієнт mysql для завдання нових користувачів. У прикладах передбачається, що привілеї встановлені відповідно до набутих за умовчанням значень, описаних в попередньому розділі. Це означає, що для внесення змін на тому ж комп'ютері, де запущений mysqld, необхідно під'єднатися до сервера як користувач MySQL root, і у користувача root має бути привілей INSERT для бази даних mysql, а також адміністративний привілей RELOAD. Крім того, якщо був змінений пароль користувача root, його необхідно вказати тут для команди mysql.

Нових користувачів можна додавати, використовуючи команду GRANT :

GRANT ALL PRIVILEGES ON \*.\* TO monty@localhost

IDENTIFIED BY 'some\_pass' WITH GRANT OPTION;

GRANT ALL PRIVILEGES ON \*.\* TO monty@"%"

IDENTIFIED BY 'some\_pass' WITH GRANT OPTION;

GRANT RELOAD, PROCESS ON \*.\* TO admin@localhost;

GRANT USAGE ON \*.\* TO dummy@localhost;

Ці команди GRANT створюють трьох нових користувачів:

* **monty**

Повноцінний суперкористувач - він може під'єднуватися до сервера звідки завгодно, але повинен використовувати для цього пароль some\_pass. Зверніть увагу на те, що ми повинні застосувати операторів GRANT як для monty@localhost, так і для monty@"%". Якщо не додати запис з localhost, запис анонімного користувача для localhost, яка створюється за допомогою mysql\_install\_db, матиме перевагу при під’єднанні з локального комп'ютера, оскільки в ній вказано певніше значення для поля Host, і вона розташована раніше в таблиці user.

* **admin**

Користувач, який може під'єднуватися з localhost без пароля; йому призначені адміністративні привілеї RELOAD і PROCESS. Ці привілеї дозволяють користувачеві запускати команди mysqladmin reload, mysqladmin refresh і mysqladmin flush-\*, а також mysqladmin processlist. Йому не призначено ніяких привілеїв, що відносяться до баз даних (їх можна призначити пізніше, додатково застосувавши оператора GRANT).

* **dummy**

Користувач, який може під'єднуватися до сервера без пароля, але тільки з локального комп'ютера. Усі глобальні привілеї встановлені в значення 'N '- тип привілею USAGE, який дозволяє створювати користувачів без привілеїв. Передбачається, що що відносяться до баз даних привілею будуть призначені пізніше.

Можна безпосередньо додати таку саму інформацію про користувача за допомогою оператора INSERT, а потім дати серверу команду перезавантажити таблиці призначення привілеїв :

INSERT INTO user VALUES('localhost ',' monty ', PASSWORD('some\_pass')

''Y ',' Y ',' Y ',' Y ',' Y ',' Y ',' Y ',' Y ',' Y ',' Y ',' Y ',' Y ',' Y ',' Y');

INSERT INTO user VALUES('%',' monty ', PASSWORD('some\_pass')

''Y ',' Y ',' Y ',' Y ',' Y ',' Y ',' Y ',' Y ',' Y ',' Y ',' Y ',' Y ',' Y ',' Y');

INSERT INTO user SET Host='localhost ', User='admin

Reload\_priv='Y ', Process\_priv='Y';

INSERT INTO user (Host, User, Password)

VALUES('localhost ',' dummy'');

FLUSH PRIVILEGES;

Залежно від версії MySQL в прикладі, приведеному вище, може вказуватися різна кількість значень 'Y' (у версіях до Version 3.22.11 було менше стовпців привілеїв). Для користувача admin використовується легший для читання розширений синтаксис команди INSERT, який доступний починаючи з версії 3.22.11.

Зверніть увагу: щоб створити суперкористувача, необхідно створити запис таблиці user з полями привілеїв, встановленими в значення 'Y'. Немає необхідності задавати значення в записах таблиць db або host.

Стовпці привілеїв в таблиці user в останньому операторові INSERT (для користувача dummy) не були задані явно, тому цим стовпцям був присвоєно набутого за умовчанням значення 'N'. Так само діє команда GRANT USAGE.

У наведеному нижче прикладі додається користувач custom, який може під'єднуватися з комп'ютерів localhost, server.domain і whitehouse.gov. Він хоче діставати доступ до бази данныхbankaccount тільки з комп'ютера localhost, до бази даних expenses - тільки з whitehouse.gov, і до бази даних customer - з усіх трьох комп'ютерів, а також використовувати пароль stupid при під’єднанні з усіх трьох комп'ютерів.

Щоб задати ці привілеї користувача за допомогою оператора GRANT, виконаєте наступні команди:

GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP

ON bankaccount.\*

TO custom@localhost

IDENTIFIED BY 'stupid';

GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP

ON expenses.\*

TO custom@whitehouse.gov

IDENTIFIED BY 'stupid';

GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP

ON customer.\*

TO custom@'%'

IDENTIFIED BY 'stupid';

Привілеї для користувача custom ми призначаємо тому, що цей користувач хоче діставати доступ до MySQL як з локального комп'ютера через сокети Unix, так і з видаленого комп'ютера whitehouse.gov через протокол TCP/IP.

Щоб задати привілеї користувача шляхом безпосереднього внесення змін до таблиць призначення привілеїв, виконаєте наступні команди (зверніть увагу на команду FLUSH PRIVILEGES у кінці прикладу) :

INSERT INTO user (Host, User, Password)

VALUES('localhost ',' custom ', PASSWORD('stupid'));

INSERT INTO user (Host, User, Password)

VALUES('server.domain ',' custom ', PASSWORD('stupid'));

INSERT INTO user (Host, User, Password)

VALUES('whitehouse.gov ',' custom ', PASSWORD('stupid'));

INSERT INTO db

( (Host, Db, User, Select\_priv, Insert\_priv, Update\_priv, Delete\_priv

Create\_priv, Drop\_priv)

VALUES

(' ('localhost ',' bankaccount ',' custom ',' Y ',' Y ',' Y ',' Y ',' Y ',' Y');

INSERT INTO db

( (Host, Db, User, Select\_priv, Insert\_priv, Update\_priv, Delete\_priv

Create\_priv, Drop\_priv)

VALUES

(' ('whitehouse.gov ',' expenses ',' custom ',' Y ',' Y ',' Y ',' Y ',' Y ',' Y');

INSERT INTO db

( (Host, Db, User, Select\_priv, Insert\_priv, Update\_priv, Delete\_priv

Create\_priv, Drop\_priv)

VALUES('%',' customer ',' custom ',' Y ',' Y ',' Y ',' Y ',' Y ',' Y');

FLUSH PRIVILEGES;

Перші три оператори INSERT додають в таблицю user записи, які дозволять користувачеві customподключаться з різних комп'ютерів з вказаним паролем, але не дають йому ніяких привілеїв (усі привілеї встановлені в набутого за умовчанням значення 'N'). Наступні три оператори INSERTдобавляют запису в таблицю db, в якій призначаються привілеї для користувача custom по відношенню до баз даних bankaccount, expenses і customer, але тільки якщо доступ здійснюється з певних комп'ютерів. Як завжди, після внесення змін безпосередньо до таблиць призначення привілеїв серверу необхідно дати команду на перезавантаження цих таблиць (за допомогою FLUSH PRIVILEGES), щоб внесені зміни набули чинності.

Якщо необхідно надати певному користувачеві доступ з будь-якого комп'ютера до певного домена, можна скористатися оператором GRANT таким чином:

GRANT ..

ON \*.\*

TO myusername" %.mydomainname.com"

IDENTIFIED BY 'mypassword';

Щоб зробити те ж саме шляхом безпосереднього внесення змін до таблиць призначення привілеїв, виконаєте наступні дії:

INSERT INTO user VALUES ('%.mydomainname.com ', 'myusername

PASSWORD('mypassword'),..);

FLUSH PRIVILEGES;

Загальний формат оператора призначення привілеїв для об'єкту типу таблиця матиме наступний синтаксис:

GRANT {[SELECT][,INSERT][,DELETE][,UPDATE (<список стовпців>)]}

ON <ім'я\_таблиці>

TO {<ім'я\_користувача> | PUBLIC } [WITH GRANT OPTION ]

Можна виконати наступні призначення привелеїв:

GRANT INSERT

ON Tab1

TO user2 ;

GRANT SELECT

ON Tab1

TO user3

Операція призначення привілеїв користувачеві user3 із правом оновлення даних:

GRANT SELECT, UPDATE (Pole2)

ON Tab1

TO user3

Надати користувачу усі права по роботі із створеною таблицею Tab1.

GRANT ALL PRIVILEGES

ON Tab1

TO user4

WITH GRANT OPTION

Призначення права на введення нових рядків в таблицю командою

GRANT INSERT

ON Tab1

TO user5

При використанні операції :

REVOKE ALL PRIVILEGES

ON Tab1

FROM user4 CASCADE

будуть скасовані привілею і користувача user5, якому користувач user4 встиг присвоїти привілеї.

Параметр RESTRICT обмежує відміну привілеїв тільки користувачеві, безпосередньо згаданому в операторові REVOKE. Але за наявності делегованих привілеїв цей оператор не буде виконаний. Так, наприклад, операція:

REVOKE ALL PRIVILEGES

ON Tab1 TO user4 RESTRICT

не буде виконана, тому що користувач user4 передав частину cвоих повноважень користувачеві user5.

## ВИМОГИ ДО ЗВІТНОСТІ

Звіт містить 3 файли:

* файл БД
* лістинг запитів та процедур створення користувачів
* документ зі скріншотами екрану, що містять результати виконання завдань.

ЗАВДАННЯ ДО САМОСТІЙНОЇ РОБОТИ

За матеріалами наведеними у виконати інтеграцію користувачів MYSQL із ACTIVE DIRECTORY - <http://winitpro.ru/index.php/2016/04/19/vypolnenie-mysql-zaprosov-iz-powershell/>.