Лабораторна робота 1. (4 год.)
ОЗНАЙОМЛЕННЯ З СЕРЕДОВИЩЕМ MYSQL WORKBENCH.
ВВЕДЕННЯ ДО МОВИ SQL. ОСНОВНІ ЗАПИТИ МОВ DDL ТА DML.(2+2)

Теоретичні відомості

База даних (БД) — впорядкований набір логічно взаємопов'язаних даних, що використовуються спільно, та зберігаються центрально або розподілено. Якщо коротко, то найпростіша **БД** це звичайна таблиця з рядками та стовпцями у якій зберігається різного роду інформація (прикладом може слугувати таблиця в**Excel**). Так, часто, з **БД** нероздільно пов'язують **Системи управління базами даних (СУБД)**, які надають функціонал для роботи з **БД**. Мова **SQL** якраз і є частиною **СУБД**, яка здійснює керування інформацією в **БД**. Ми будемо вважати **БД** набором звичайних таблиць, які зберігаються в окремих файлах.

SQL - проста мова програмування, яка має небагато команд і якій може навчитися будь-який бажаючий. Розшифровується як **Structured Query Language** - мова структурованих запитів, яка була розроблена для роботи з **БД**, а саме, щоб отримувати/добавляти/змінювати дані, мати можливість опрацьовувати великі масиви інформації та швидко отримувати структуровану та згруповану інформацію. Є багато варіантів мови **SQL**, але у них всіх основні команди майже однакові. Також існує і багато **СУБД**, але основними з них являються: **Microsoft Access, Microsoft SQL Server, MySQL, Oracle SQL, IBM DB2 SQL, PostgreSQL та Sybase Adaptive Server SQL**. Щоб працювати з **SQL** кодом, нам потрібна буде одна з вище перелічених **СУБД**. Для навчання ми будемо використовувати **СУБД MySQL**.

Інструментальним засобом будемо використовувати **MySQL Workbench.** MySQL Workbench дозволяє з легкістю проектувати EER-модель "сутність-зв'язок" та експортувати цю модель в логічну модель (**SQL-дамп .mwb модели).** MySQL Workbench може бути прекрасною альтернативою PHPMyAdmin і в адмініструванні даних. Буде потрібно лише створити підключення до MySQL серверу.

Створення віддаленого підключення до сервера MySQL

На стартовому екрані вибираємо пункт меню "Open connection to start querying" або вибираємо з основного меню "Database → Query database ...".

Щоб створити нове підключення, вибираємо "Database → Manage Connections ...", у вікні, натискаємо кнопку "New". MySQL Workbench пропонує три способи підключення до сервера: пряме підключення через користувача, якому дозволений віддалений доступ до MySQL (зазвичай доступ таких користувачів обмежують по IP), socket / pipe підключення через файл сокета (для UNIX) або pipe (для Windows), а так само підключення через SSH-тунель (вимагає наявність SSH доступу і юзера SSH і MySQL з відповідними правами).

Розглянемо варіанти підключення до віддаленого сервера:

Через віддаленого користувача MySQL (Standard: TCP / IP)

У діалоговому вікні створення підключення вибираємо тип підключення "Standard: TCP / IP":

У полі "Host" вводимо адресу сервера MySQL або адресу сайту (якщо MySQL сервер знаходиться на самому веб сервері)

"Port" за замовчуванням найчастіше +3306

Вводимо ім'я користувача MySQL ("Username"), пароль ("Password") і можна задати ім'я бази даних ("Default Schema") (ця властивість не обов'язкова)

Після створення підключення натискаємо "Test Connection" і чекаємо повідомлення "Connection parameters are correct."

Якщо щось пішло не так, перевіряємо, чи включений у користувача віддалений доступ, а так само наявність IP адреси нашого комп'ютера в списку дозволених для даного користувача.

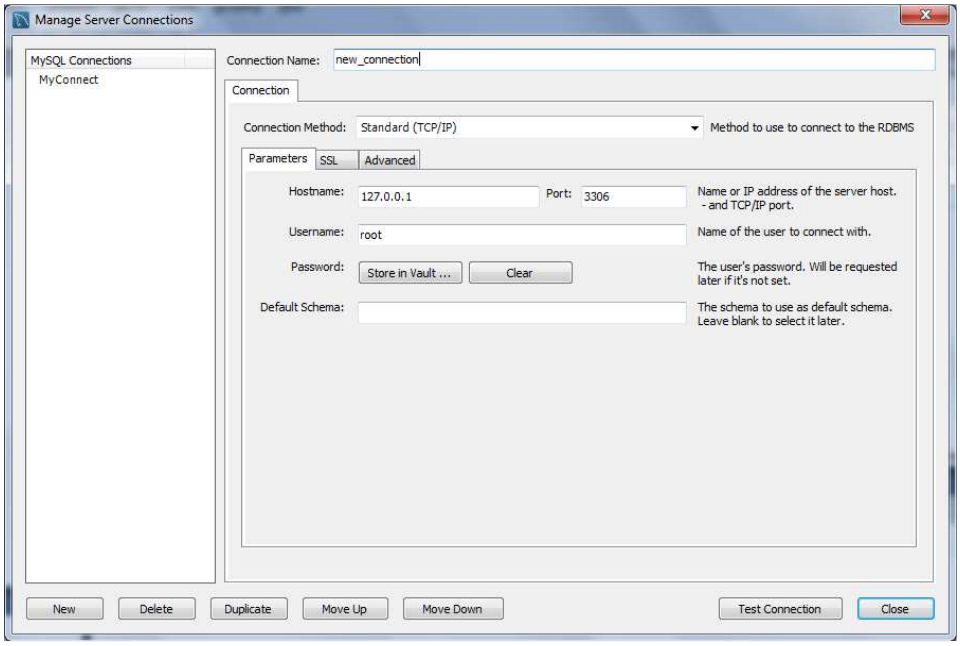

Рис.1.

В наступний раз для доступу до сервера MySQL вибираємо вже наявне підключення. Вибираємо підключення на стартовому екрані або вибираємо "Database → Connect to Database ..." (Ctrl + U) і вибираємо з'єднання зі списку.

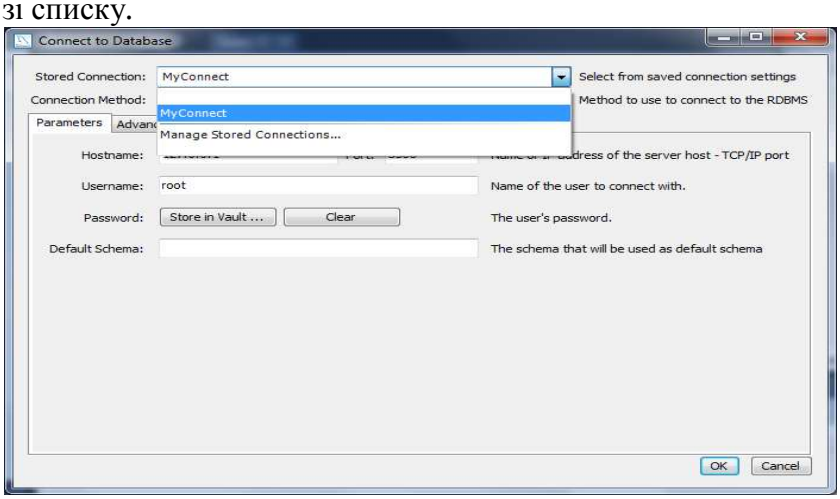

Рис.2.

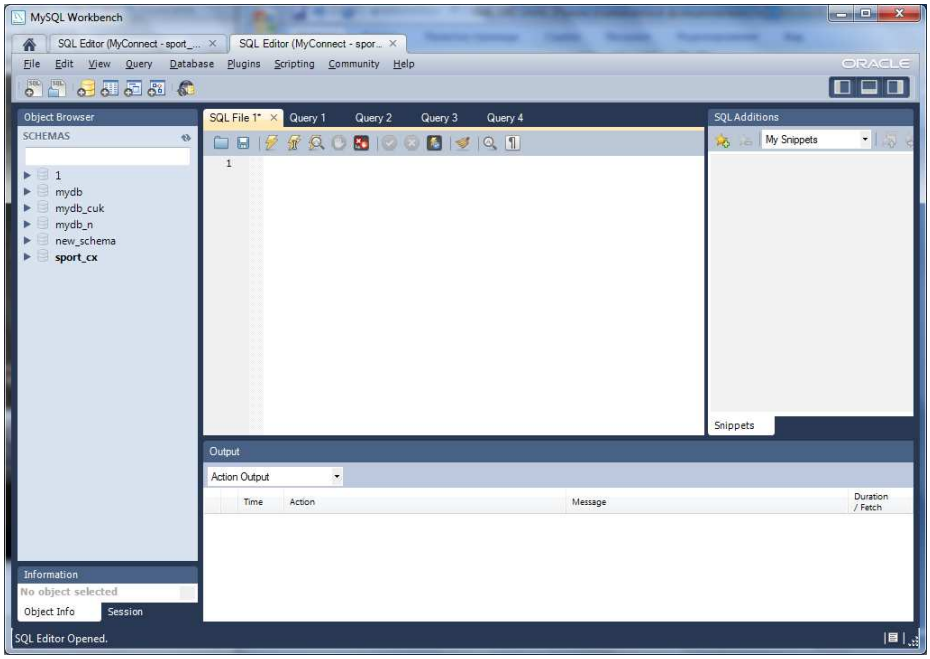

Рис.3.

SQL–РЕДАКТОР

Зліва направо це:

Create a New SQL Script File: Створює нову вкладку SQL скрипт, де код SQL може бути введено.

Open a SQL Script File: При натисканні на цю кнопку дозволяє відкрити будь-який збережений SQL скрипт, готовий для завантаження та виконання. Сценарій буде відображатися в області SQL запитів.

Save SQL Script to File: Натискання на цю кнопку дозволяє в даний час завантажений SQL скрипт зберегти у файл, вказаний користувачем.

Save SQL to Snippets List: SQL фрагменти коду, можуть бути збережені. Вони з'являться в палітрі SQL Фрагменти в бічній панелі редактора SQL.

Execute SQL Script in Connected Server: Виконує поточний завантажений SQL скрипт. Результати відображаються на одній або кількох вкладках результатів.

Execute Current SQL Statement in Connected Server: Виконує поточну запит SQL. Результати відображаються на одній або кількох вкладках результатів.

Explain (All or Selection): Пояснення запиту SQL або обраний оператор.

Stop the query being executed: зупиняє виконання виконуваного в даний момент скрипта SQL. Це перезапускає підключення до сервера бази даних.

Commit: Виконує транзакцію.

Rollback: відкат транзакції.

Toggle Auto-Commit Mode: У разі вибору, запит автоматично буде здійснено.

Reconnect to DBMS: Перевстановлює з'єднання з базою даних.

Refresh state of database structures: Обновляє вид схеми, таблиці, таблиці і підпрограми, яка з'являється в Tabsheet. Наприклад, якщо SQL скрипт створює нову таблицю, воно не буде показано на вкладці Огляд, поки кнопка на панелі інструментів оновлення не буде натиснута.

Toggle whether query result tabs should be kept between queries by default: Зазвичай, коли сценарій виконується будь-які результати, отримані від попередніх виконань сценарію губляться, і нові результати відображаються на вкладці результатів. Якщо це кнопка перемикання натиснута, так що штифт заданий, результати будуть збережені між виконанням. Кожне виконання скрипта буде створювати нову вкладку результатів, що містить набір результатів.

РОЗРОБЛЕННЯ SQL СКРИПТУ ДЛЯ СТВОРЕННЯ ТАБЛИЦІ БД.

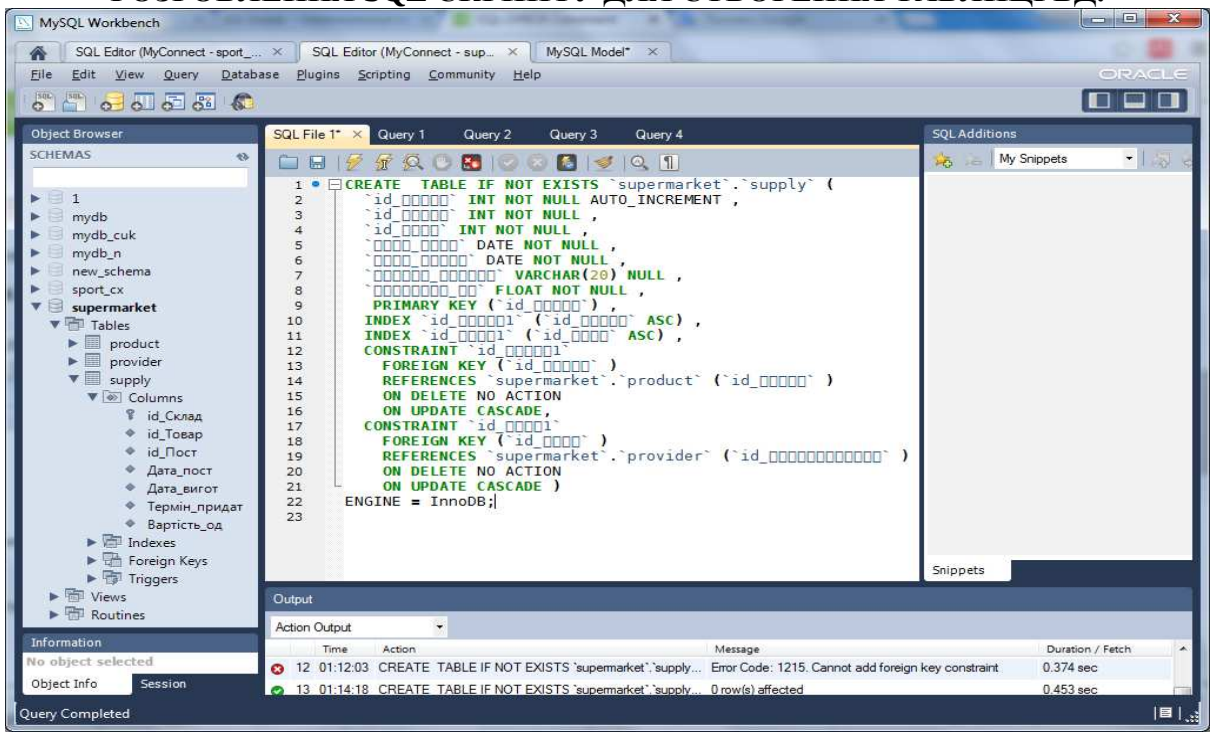

Рис.4.

АДМІНІСТРУВАННЯ БАЗИ ДАНИХ

Список доступних для редагування баз даних можна побачити в лівій панелі в розділі "SCHEMAS". Відкривши потрібну базу даних, можна побачити список таблиць в ній (рис. 5).

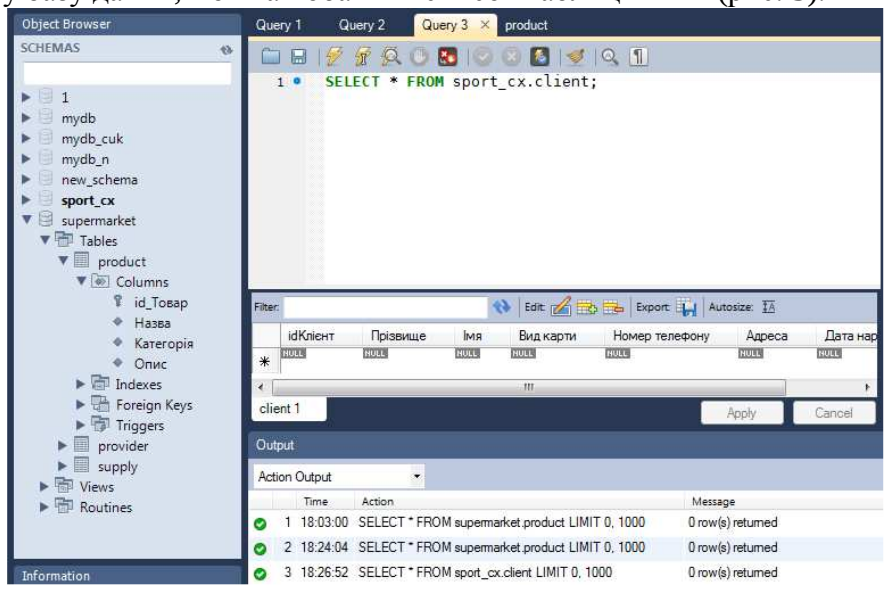

Рис.5.

У правій же області ми можемо побачити вкладки різних видів, наприклад, на скріншоті вище відкрита вкладка складання SQL запиту, в якій є маса корисних кнопок, що допомагають у його редагуванні. Виконувати запити можна комбінацією клавіш Ctrl + Enter або за допомогою кнопки *Execute* (кнопка у вигляді блискавки).

Для перегляду, створення або редагування записів натискаємо на потрібну таблицю правою кнопкою і вибираємо "*Select Rows - Limit 1000*" або виконуємо потрібний для вибірки SQL запит (рис.6).

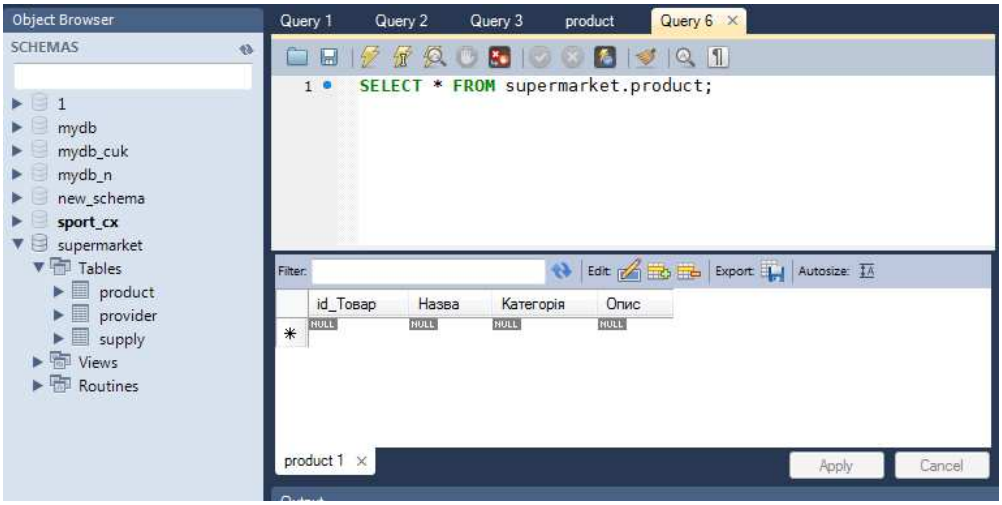

Рис.6.

Поки-що база даних порожня. Щоб додати дані виберіть пункт *"Edit Table Data"*.

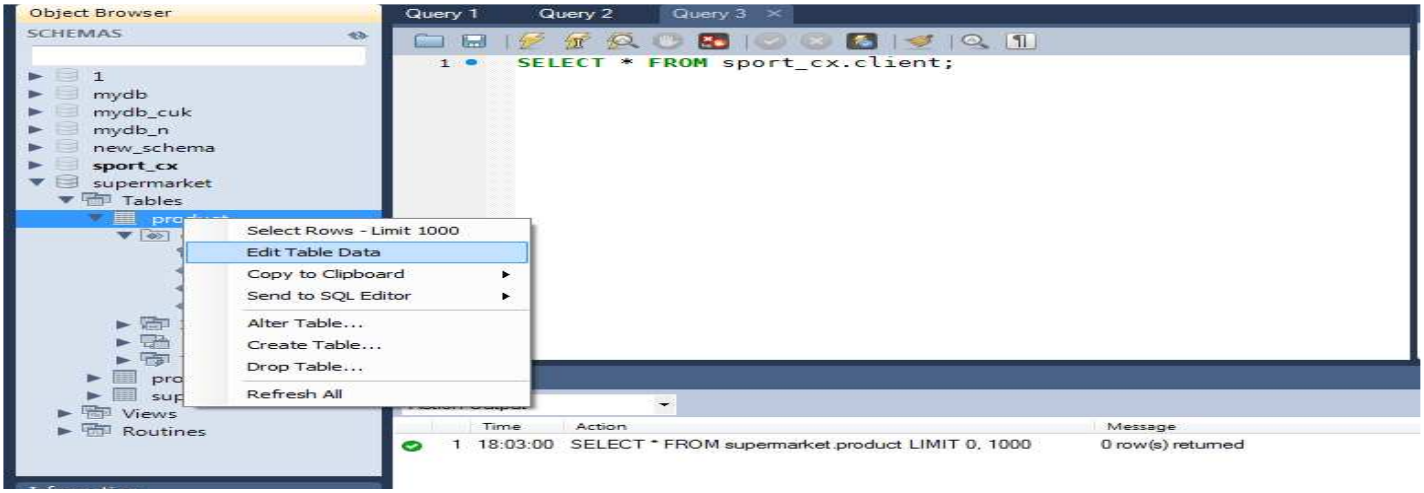

Рис.7.

Вводимо відповідні дані і обов'язково натискаємо кнопку ***Apply*** для підтвердження та внесення змін в БД.

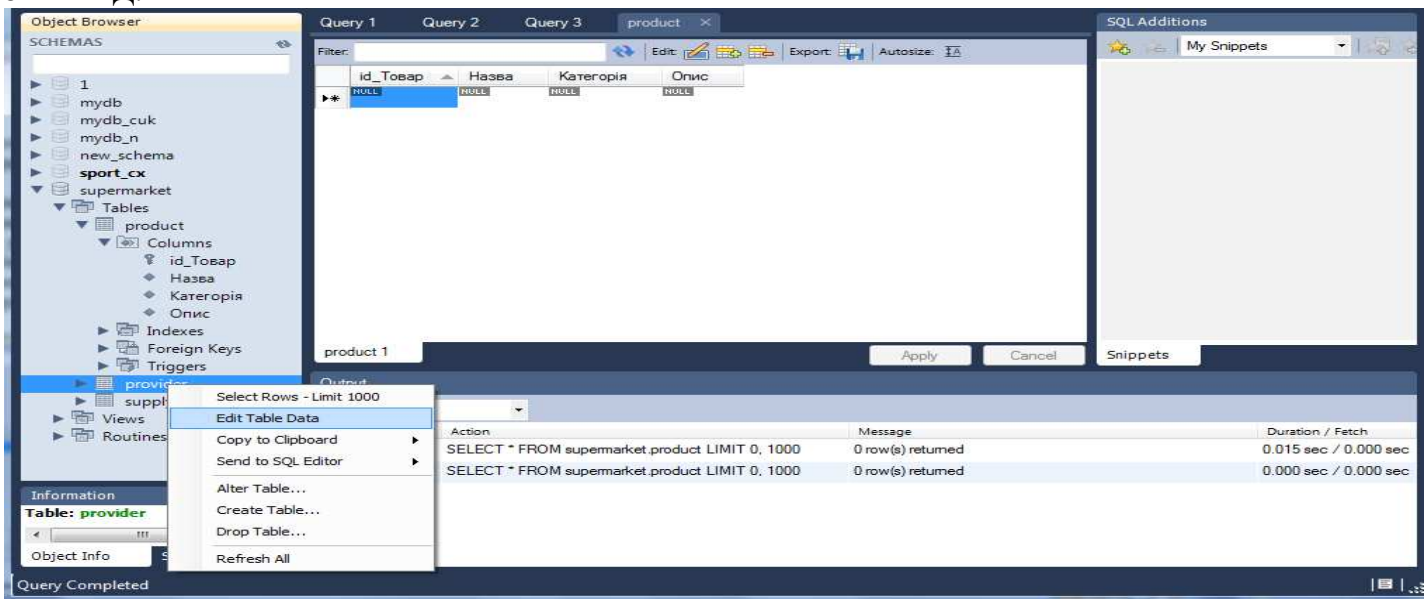

Рис.8.

СКЛАД МОВИ SQL

- мова визначення даних (Data Definition Language, DDL);
- мова маніпулювання даними (Data Manipulation Language, DML);
- мова керування даними (Data Control Language, DCL).

ВИЗНАЧЕННЯ DDL

Мова визначення даних (DDL) являє собою мову, яка призначена для опису схеми БД або її частини. Мова DDL визначає:

- типи даних;
- структуру даних;
- зв'язки.

В Базі Даних MySQL є три основні типи даних: **text, number, and Date/Time**.

Тип даних **Text**.

Тип даних	Опис даних
CHAR(size)	Зберігає строкові дані фіксованого обєму (можуть бути літери, цифри та спеціальні знаки). Фіксований розмір вказується в круглих дужках. Може зберігати до 255 символів
VARCHAR(size)	Зберігає змінної величини строкові дані(можуть бути літери, цифри та спеціальні знаки). Максимальний розмір визначений в круглих дужках. Зберігає до 255 символів. **Зауважте:** Якщо поле буде мати більше ніж 255 символів, то воно буде конвертовано в тип даних **TEXT**
TINYTEXT	Зберігає строкові дані максимальної довжини в 255 символів.
TEXT	Зберігає строкові дані довжиною в 65,535 символів.
BLOB	Для даних (Binary Large OBjects). Зберігає до 65,535 біт даних.
MEDIUMTEXT	Зберігає строкові дані довжиною в 16,777,215 символів.
MEDIUMBLOB	Для даних (Binary Large OBjects). Зберігає до 16,777,215 біт даних.
LONGTEXT	Зберігає строкові дані довжиною в 4,294,967,295 символів.
LONGBLOB	Для даних (Binary Large OBjects). Зберігає до 4,294,967,295 біт даних.
ENUM(x,y,z,etc.)	Дозволяє ввести список можливих значень. Список може зберігати до 65535 значень в **ENUM** списку. Якщо значення, що має бути записане в базу даних, відсутнє в списку,- то буде занесене пусте значення. Зауваження: Значення будуть відсортовані в тому порядку в якому ви їх запишете. Можливі значення вводяться в такому форматі: ENUM('X','Y','Z')
SET	Схожий тип даних до **ENUM** за виключенням, **SET** допускає список із 64 елементів

Тип даних **Number**.

Тип даних	Опис даних
TINYINT(size)	Цілий від -128 до 127. Від 0 до 255 UNSIGNED*. Максимальне число цифр задається в дужках.
SMALLINT(size)	Від -32768 до 32767. Від 0 до 65535 UNSIGNED*. Максимальне число цифр задається в дужках.

MEDIUMINT(size)	Від -8388608 до 8388607. Від 0 до 16777215 UNSIGNED*. Максимальне число цифр задається в дужках.
INT(size)	Від -2147483648 до 2147483647. Від 0 до 4294967295 UNSIGNED*. Максимальне число цифр задається в дужках.
BIGINT(size)	Від -9223372036854775808 до 9223372036854775807. Від 0 до 18446744073709551615 UNSIGNED*. Максимальне число цифр задається в дужках.
FLOAT(size,d)	Число з плаваючою крапкою. Максимальне число цифр задається в параметрі size. Максимальне число цифр після десяткової крапки задається в параметрі d.
DOUBLE(size,d)	Точніше число з плаваючою крапкою. Максимальне число цифр задається в параметрі size. Максимальне число цифр після десяткової крапки задається в параметрі d.
DECIMAL(size,d)	DOUBLE, що зберігається як рядок з фіксованою крапкою. Максимальне число цифр задається в параметрі size. Максимальне число цифр після десяткової крапки задається в параметрі d.

UNSIGNED*- Цілі типи мають додаткову опцію UNSIGNED (беззнаковий).

Тип даних **Date/Time**.

Тип даних	Опис даних
DATE()	Дата. Формат: YYYY-MM-DD Зауваження: Підтримується діапазон від '1000-01-01' до '9999-12-31'
DATETIME()	Формат: YYYY-MM-DD HH:MM:SS *. Зауваження: Підтримується діапазон від '1000-01-01 00:00:00' до '9999-12-31 23:59:59'
TIMESTAMP()	Значення TIMESTAMP зберігаються як кількість секунд з початку епохи Unix ('1970-01-01 00:00:00' UTC). Формат: YYYY-MM-DD HH:MM:SS* Зауваження: Підтримується діапазон від '1970-01-01 00:00:01' UTC до '2038-01-09 03:14:07' UTC
TIME()	Час. Формат: HH:MM:SS Зауваження: Підтримується діапазон від '-838:59:59' до '838:59:59'
YEAR()	Рік в двоцифровому, або чотирицифровому форматі (формат YYYY чи YY). Зауваження: Значення, що дозволені в чотирицифровому форматі: від 1901 до 2155. Значення дозволені в двоцифровому форматі: від 70 до 69, що відповідає 1970 та 2069.

***- Навіть якщо DATETIME та TIMESTAMP повертають однакові формати, вони працюють дуже по різному. В запиті INSERT або UPDATE TIMESTAMP автоматично встановлює поточний час і дату. Також TIMESTAMP приймає різні формати, як YYYYMMDDHHMMSS, YYMMDDHHMMSS, YYYYMMDD, чи YYMMDD.**

Команди DDL:

- команди створення;
- команди модифікації;
- команди видалення.

Команди створення DDL

Назва команди	**Опис**
CREATE DATABASE	Створити базу даних
CREATE TABLE	Створити таблицю
CREATE VIEW	Створити віртуальну таблицю
CREATE INDEX	Створити індекс
CREATE TRIGGER	Створити тригер
CREATE PROCEDURE	Створити збережену процедуру

Команди модифікації DDL

Назва команди	**Опис**
ALTER DATABASE	Модифікувати базу даних
ALTER TABLE	Модифікувати таблицю
ALTER VIEW	Модифікувати віртуальну таблицю
ALTER INDEX	Модифікувати індекс
ALTER TRIGGER	Модифікувати тригер
ALTER PROCEDURE	Модифікувати збережену процедуру

Команди видалення DDL

Назва команди	**Опис**
DROP DATABASE	Видалити базу даних
DROP TABLE	Видалити таблицю
DROP VIEW	Видалити віртуальну таблицю
DROP INDEX	Видалити індекс
DROP TRIGGER	Видалити тригер
DROP PROCEDURE	Видалити збережену процедуру

SQL-запит створення Бази Даних CREATE DATABASE

Синтаксис SQL-запиту **CREATE DATABASE**.

```
CREATE DATABASE dbname;
```

Наступний запит створює Базу Даних **"my_db"**:

```
CREATE DATABASE my_db;
```

База Даних може також бути добавлена за допомогою SQL-запиту **CREATE TABLE**.

SQL-запит створення таблиці CREATE TABLE

SQL-запит CREATE TABLE використовується якщо необхідно створити таблицю в Базі Даних.

Таблиці організовані в рядки і стовпці; і кожна таблиця повинна мати назву.

Синтаксис SQL-запиту **CREATE TABLE**.

```
CREATE TABLE table_name
(
column_name1 data_type(size),
column_name2 data_type(size),
column_name3 data_type(size),
....
);
```

Параметр **column_name** вказує ім'я стовпця таблиці.

Параметр **data_type** вказує якого типу параметр буде мати стовпця таблиці (тобто varchar, integer, decimal, date, etc.).

Параметр **size** вказує якої максимальної довжини буде стовпець таблиці.

Приклад створення SQL-запиту **CREATE TABLE**.

```
CREATE TABLE Persons
(
PersonID int,
LastName varchar(255),
FirstName varchar(255),
Address varchar(255),
City varchar(255)
);
```

Стовпець **PersonID**; тип даних **int** і мають бути цифрові значення.

Стовпця **LastName, FirstName, Address, та City** ; тип даних **varchar** і має утримувати символьні значення даних, максимальна довжина поля 255 символів.

Пуста таблиця **"Persons"** буде мати наступний вираз:

PersonID	LastName	FirstName	Address	City

Заповнити таблицю можна використовуючи оператор **INSERT INTO**

SQL-запит створення таблиці CREATE TABLE. Обмеження в SQL-виразах

Обмеження в SQL-виразах використовується, якщо необхідно вказати певні правила для зберігання даних Бази Даних.
Якщо є порушення виконання запитів згідно SQL-обмежень, то ці дії виконуватись не будуть.

Зазвичай обмеження в SQL-виразах застосовуються при створенні таблиці (всередині SQL-виразу CREATE TABLE) чи після того, як таблиця була створена (всередині SQL-виразу ALTER TABLE).

SQL CREATE TABLE + синтаксис обмеження

```
CREATE TABLE table_name
(
column_name1 data_type(size) constraint_name,
column_name2 data_type(size) constraint_name,
column_name3 data_type(size) constraint_name,
....
);
```

В SQL є наступні обмеження

NOT NULL - вказує, що стовпець не повинна зберігати NULL значення даних;
UNIQUE - гарантовано вказує, що кожен рядок стовпця має унікальне значення;
PRIMARY KEY - комбінація NOT NULL та UNIQUE. Гарантує, що стовпець(чи комбінація двох і більше колонок) мають унікальний ідентифікатор, що допомагає віднайти частину записів в таблиці більш легко та швидко;
FOREIGN KEY - гарантує наслідування одних даних таблиці у відповідності до іншої таблиці;
CHECK - гарантує, що значення стовпця таблиці виконує окремі умови;
DEFAULT - вказує початкові дані за замовчуванням.

Застосування умови NOT NULL в SQL-виразах

За замовчуванням значення поля таблиці даних може приймати **NULL** значення.
Обмеження **NOT NULL** в SQL-виразах примушує завжди вносити значення в поле даних, тобто ви не зможете внести чи обновити запис, без внесення значення в поле таблиці.
Наступний SQL-вираз примушує вносити дані в поле стовпця **"P_Id"** та **"LastName"**.

SQL CREATE TABLE + синтаксис обмеження

```
CREATE TABLE PersonsNotNull
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

Застосування умови PRIMARY KEY в SQL-виразах

Застосування умови **PRIMARY KEY** в SQL-виразах створює унікальний ідентифікатор для кожного запису поля даних Бази Даних.

Значення поля **PRIMARY KEY** має бути унікальним.

Значення поля **PRIMARY KEY** не повинно мати **NULL** значення.

PRIMARY KEY автоматично створюється як UNIQUE поля даних.

Одна таблиця має лише один **PRIMARY KEY**.

Умова PRIMARY KEY в SQL-виразі при створенні таблиці CREATE TABLE

Наступний SQL-вираз створює колонку **"P_Id"** з умовою **PRIMARY KEY** при створенні таблиці **"Persons"** (для MySQL)

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
PRIMARY KEY (P_Id)
)
```

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CONSTRAINT pk_PersonID PRIMARY KEY (P_Id,LastName)
)
```

Зауважте: В прикладі поданому вище, лише один PRIMARY KEY (pk_PersonID), але це значення pk_PersonID складається з двох колонок (P_Id та LastName).

Використання індексу AUTO INCREMENT таблиці Бази Даних

Параметр AUTO INCREMENT дозволяє створювати новий унікальний індекс при внесенні нових даних в таблицю.

Досить часто нам необхідно, щоб автоматично створювався новий унікальний індекс (primary key) кожного разу як лише вносяться нові дані. В такому випадку ми застосовуємо автоматично зростаючий індекс поля даних.

Синтаксис SQL-виразу з параметром AUTO INCREMENT (для MySQL)

В поданому SQL-запиті вказано, що стовпець індексу **"ID"** буде мати параметр **AUTO INCREMENT** таблиці **"Persons"**:

```
CREATE TABLE Persons
(
ID int NOT NULL AUTO_INCREMENT,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
PRIMARY KEY (ID)
)
```

MySQL використовує ключове слово **AUTO_INCREMENT** для того, щоб відбувалось автоматичне зростання індексу. За замовчуванням, значення **AUTO_INCREMENT** розпочинається з 1, і він збільшується на 1 з кожним новим записом. Щоб встановити інше значення поля **AUTO_INCREMENT** можна застосувати наступний вираз:

```
ALTER TABLE Persons AUTO_INCREMENT=100
```

З внесенням нових даних в таблицю **"Persons"** ми не встановлюємо ніяких значень для стовпця **"ID"** (новий унікальний індекс буде створений автоматично)

```
INSERT INTO Persons (FirstName,LastName)
VALUES ('Lars','Monsen')
```

З поданому вище SQL-запиті вносяться нові дані в таблицю **"Persons"**.
В колонку **"ID"** буде автоматично внесено унікальне значення.
В колонку **"FirstName"** буде внесено **"Lars"**, а в колонку **"LastName"** буде внесено**"Monsen"**.

Умова PRIMARY KEY в SQL-виразі при внесенні змін до таблиці ALTER TABLE

Наступний SQL-вираз створює колонку **"P_Id"** з умовою **PRIMARY KEY** коли таблиця**"Persons"** вже створена.0

```
ALTER TABLE Persons
ADD PRIMARY KEY (P_Id)
```

Для декількох колонок

```
ALTER TABLE Persons
ADD CONSTRAINT pk_PersonID PRIMARY KEY (P_Id,LastName)
```

Вилучення умови PRIMARY KEY

```
ALTER TABLE Persons
DROP PRIMARY KEY
```

Застосування SQL запиту FOREIGN KEY (зовнішній ключ)

Поле таблиці з атрибутом **FOREIGN KEY** встановлює однозначний зв'язок однієї таблиці з полем **PRIMARY KEY** іншої таблиці.
Краще проілюструвати особливості foreign key (зовнішній ключ) на прикладі.
Погляньте на наступні дві таблиці.

Таблиця "Persons"(Люди):

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Таблиця "Orders"(замовлення):

O_Id	OrderNo	P_Id
1	77895	3
2	44895	3
3	24562	1

Зауважимо, що стовпець **"P_Id"** в таблиці "Orders" вказує на колонку **"P_Id"** в таблиці "Persons".
Стовпець **"P_Id"** в в таблиці **"Persons"**(Люди) є **PRIMARY KEY** (первиний ключ).
Стовпець **"P_Id"** в в таблиці **"Orders"**(замовлення) є **FOREIGN KEY** (зовнішній ключ)
Атрибут **FOREIGN KEY** застосовується для запобігання діям, що можуть порушити зв'язки між таблицями.

Атрибут **FOREIGN KEY** також запобігає внесенню неіснуючих даним внаслідок внесення змін в поле з атрибутом foreign key, тому що має бути відповідність з даними таблиці, на яке поле **FOREIGN KEY** посилається.

Застосування SQL запиту FOREIGN KEY при створенні таблиці CREATE TABLE даних

Наступний SQL запит створює **FOREIGN KEY** для Стовпця **"P_Id"** в таблиці "Orders".

```
CREATE TABLE Orders
(
O_Id int NOT NULL,
OrderNo int NOT NULL,
P_Id int,
PRIMARY KEY (O_Id),
FOREIGN KEY (P_Id) REFERENCES Persons (P_Id)
)
```

Для того, щоб присвоїти назву для поля **FOREIGN KEY**, та для визначення безпосередньо **FOREIGN KEY** в таблиці, що має багато колонок з даними, використовується наступний запис.

```
CREATE TABLE Orders
(
O_Id int NOT NULL,
OrderNo int NOT NULL,
P_Id int,
PRIMARY KEY (O_Id),
CONSTRAINT fk_PerOrders FOREIGN KEY (P_Id)
REFERENCES Persons(P_Id)
)
```

Застосування SQL запиту FOREIGN KEY при змінах в таблиці ALTER TABLE

Якщо необхідно додатково створити поле з **FOREIGN KEY** в таблиці, що вже існує, тоді використовується запис:

```
ALTER TABLE Orders
ADD FOREIGN KEY (P_Id)
REFERENCES Persons(P_Id)
```

Для того, щоб присвоїти назву для поля **FOREIGN KEY**, та для визначенняя безпосередньо **FOREIGN KEY** в таблиці, що має багато колонок з даними, використовується наступний запис.

```
ALTER TABLE Orders
ADD CONSTRAINT fk_PerOrders
FOREIGN KEY (P_Id)
REFERENCES Persons(P_Id)
```

Вилучення FOREIGN KEY з таблиці даних

Для вилучення поля **FOREIGN KEY** з таблиці використовується запис:

```
ALTER TABLE Orders
DROP FOREIGN KEY fk_PerOrders
```

Застосування умови DEFAULT в SQL-виразах

Застосування умови **DEFAULT** в SQL-виразах використовується для внесення початкових значень поля даних Бази Даних.
Початкове значення буде встановлене у всі нові записи, якщо інші значення не вказані.

DEFAULT в SQL-виразі при створенні таблиці CREATE TABLE

Наступний SQL-вираз надає для стовпця **"City"** початкове значення при створенні таблиці **"Persons"**.

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255) DEFAULT 'Sandnes'
)
```

Зручно використовувати **DEFAULT** якщо необхідно внести значення дати, застосовуючи функцію **GETDATE():**

```
CREATE TABLE Orders
(
O_Id int NOT NULL,
OrderNo int NOT NULL,
P_Id int,
OrderDate date DEFAULT GETDATE()
)
```

DEFAULT в SQL-виразі при внесенні змін до таблиці ALTER TABLE

Наступний SQL-вираз вносить дані стовпця **"City"**, коли таблиця **"Persons"** вже створена.(Для MySQL)

```
ALTER TABLE Persons
ALTER City SET DEFAULT 'SANDNES'
```

Вилучення умови DEFAULT

```
ALTER TABLE Persons
ALTER City DROP DEFAULT
```

Видалення індексів, таблиці, бази даних
DROP INDEX, DROP TABLE, та DROP DATABASE

Індекси, таблиці, Бази Даних можуть бути легко очищені від даних за допомогою SQL-виразу DROP.

SQL-вираз видалення індексів DROP INDEX

Зауважте: Для різних Баз Даних Різний синтаксис, в даному випадку для MySQL:

```
ALTER TABLE table_name DROP INDEX index_name
```

SQL-вираз видалення таблиці DROP TABLE

```
DROP TABLE table_name
```

SQL-вираз видалення бази даних DROP DATABASE

```
DROP DATABASE database_name
```

SQL-вираз видалення даних з таблиці TRUNCATE TABLE

Якщо необхідно видалити всі дані з таблиці, але залишити саму таблицю (пусту) використовується наступний синтаксис.

```
TRUNCATE TABLE table_name
```

Зміна вигляду таблиць ALTER TABLE

SQL-вираз ALTER TABLE призначений для зміни, видалення чи модифікації колонок існуючих таблиць.

Синтаксис SQL-виразу ALTER TABLE

Щоб добавити колонку в таблицю, необхідно записати наступний вираз:

```
ALTER TABLE table_name
ADD column_name datatype
```

Щоб видалити колонку з таблиці: (але відмітьте собі, що деякі Системи Керування Баз Даних не дозволяють видалення колонок).

```
ALTER TABLE table_name
DROP COLUMN column_name
```

Внесення змін щодо типу даних таблиці:
My SQL / Oracle:

```
ALTER TABLE table_name
MODIFY COLUMN column_name datatype
```

Приклад внесення змін у вигляд таблиці ALTER TABLE

Таблиця **"Persons"**

P_Id	LastName	FirstName	City
1	Hansen	Ola	Sandnes
2	Svendson	Tove	Sandnes
3	Pettersen	Kari	Stavanger

Нам потрібно добавити поле **"DateOfBirth"**(День народження)
Запишемо наступний вираз:

```
ALTER TABLE Persons
ADD DateOfBirth date
```

Нова стовпець **"DateOfBirth"** повинна мати записи даних з типом даних **date**.

Нова таблиця буде мати вигляд:

P_Id	LastName	FirstName	City	DateOfBirth

1	Hansen	Ola	Sandnes	
2	Svendson	Tove	Sandnes	
3	Pettersen	Kari	Stavanger	

Приклад зміни типу даних таблиці

```
ALTER TABLE Persons
MODIFY COLUMN DateOfBirth year
```

Тип даних поля **"DateOfBirth"** тепер **year**.

Видалення стовпця даних таблиці DROP COLUMN

```
ALTER TABLE Persons
DROP COLUMN DateOfBirth
```

Таблиця буде мати вигляд:

P_Id	LastName	FirstName	City
1	Hansen	Ola	Sandnes
2	Svendson	Tove	Sandnes
3	Pettersen	Kari	Stavanger

ВИЗНАЧЕННЯ DML

Мова маніпулювання даними *(DML)* (або мова запитів до БД) представляється зазвичай системою команд маніпулювання даними.

Приклад команд DML

- провести вибірку з бази даного по його найменуванню;
- провести вибірку з бази всіх даних певного типу, значення яких задовольняють певними ознаками;
- знайти в базі позицію даного і помістити туди його нове значення.

Команди DML

Назва команди	**Опис**
SELECT	Вибрати
INSERT	Вставити
UPDATE	Оновити
DELETE	Видалити

Приклад команд DCL

Мова керування даними *(DCL)* - призначена для адміністратора системи і дозволяє визначати повноваження користувачів, забезпечує його засобами захисту, оптимізації і підтримки цілісності.

ВИБІРКА ОКРЕМИХ ПОЛІВ.

SELECT Product FROM Sumproduct

Бачимо, що наш **SQL запит** відібрав колонку **Product** з таблиці **Sumproduct**.

ВИБІРКА КІЛЬКОХ ПОЛІВ.

Припустимо, нам необхідно вибрати назву та кількість реалізованого товару. Для цього просто перераховуємо необхідні поля через кому:

SELECT Product, Quantity FROM Sumproduct

ВИБІРКА ВСІХ СТОВПЦІВ.

Якщо ж нам необхідно отримати всю таблицю зі всіма полями, тоді просто ставимо знак зірочка (*****):

SELECT * FROM Sumproduct

PS. Всі оператори в **SQL** нечутливі до регістру, тому ви можете їх писати як великими буквами, так і маленькими (як правило, їх прийнято писати великими буквами, щоб розрізняти від назв полів та таблиць). Назви же таблиць та полів є навпаки чутливими до регістру та мають писатися точно як в **БД**.

Синтаксис SQL запиту SELECT

Приклад вибірки * SELECT

Представлений нижче вираз вибирає всі стовпця з таблиці **"Customers"**

```
Приклад:
SELECT * FROM Customers;
```

Варто відмітити порядок опрацювання виразу SELECT:

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT
6. ORDER BY

Цей порядок дещо не відповідає синтаксичному порядку загального виразу SELECT ;

```
SELECT
[FROM table_references]
[WHERE where_condition]
[GROUP BY]
[HAVING where_condition]
[ORDER BY]
[LIMIT ]
```

В загальному випадку:

```
SELECT

  [ALL | DISTINCT | DISTINCTROW ]

    [HIGH_PRIORITY]

    [MAX_STATEMENT_TIME = N]
    [STRAIGHT_JOIN]
    [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
    [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
  select_expr [, select_expr ...]
  [FROM table_references
    [PARTITION partition_list]
  [WHERE where_condition]
  [GROUP BY {col_name | expr | position}
    [ASC | DESC], ... [WITH ROLLUP]]
  [HAVING where_condition]
  [ORDER BY {col_name | expr | position}
    [ASC | DESC], ...]
  [LIMIT {[offset,] row_count | row_count OFFSET offset}]
  [PROCEDURE procedure_name(argument_list)]
  [INTO OUTFILE 'file_name'
      [CHARACTER SET charset_name]
      export_options
    | INTO DUMPFILE 'file_name'
    | INTO var_name [, var_name]]
```

```
[FOR UPDATE | LOCK IN SHARE MODE]]
```

SQL запит SELECT

SQL запит **SELECT** використовується для вибірки даних з Бази Даних.
Результат зберігається в таблиці, з назвою "результуюча вибірка"

Синтаксис запиту **SELECT**:

```
SELECT column_name,column_name
FROM table_name;
```

та

```
SELECT * FROM table_name;
```

Приклад вибірки стовпця за допомогою SELECT

Представлений нижче вираз вибирає стовпця з іменем споживача **"CustomerName"**та містом **"City"** з таблиці **"Customers"**

```
Приклад:
SELECT CustomerName,Address FROM Customers;
```

Синтаксис SQL запиту SELECT DISTINCT

SQL запит **SELECT DISTINCT** використовується для вибірки несхожих (різних) даних з Бази Даних.
В таблиці, в стовпецьх можуть зберігатися багато однакових значень, і буває необхідним переглянути список лише різних даних.
Ключове слово DISTINCT в SQL запиті використовується для вибірки лише несхожих даних.

Синтаксис запиту **SELECT DISTINCT**:

```
SELECT DISTINCT column_name,column_name
FROM table_name;
```

Оператор умови WHERE в SQL запиті

Оператор умови **WHERE** використовується для зчитування даних лише тих записів, які задовільняють особливим критеріям.

Синтаксис SQL запиту з використанням фільтру **WHERE**:

```
SELECT column_name,column_name
FROM table_name
WHERE column_name operator value;
```

Приклад вибірки стовпця за допомогою оператора умови WHERE

Представлений нижче вираз вибирає з таблиці **"Customers"** всіх (символ вибору *) споживачів з країни Mexico.

```
Приклад:
SELECT * FROM Customers
```

```
WHERE Country='Mexico';
```

Текстові чи числові поля

SQL запити вимагають текст вставляти в одинарні лапки (більшість Баз Даних дозволяють вставляти і в подвійні лапки).
Що ж до числових значень, то вони не мають бути в лапках.

```
Приклад:
SELECT * FROM Customers WHERE ID=3;
```

Оператори порівняння при фільтрації з WHERE

Наступні оператори можуть використовуватись в умові **WHERE**.

Оператор	**Призначення**
=	Одинакові
<>	Не одинакові. Зауважте, в деяких версіях знак може бути як !=
>	Більше чим
<	Менше чим
>=	Більше чим або рівно з умовою
<=	Менше чим або рівно з умовою
BETWEEN	Між вказаними значеннями
LIKE	шукати за зразком
IN	стовпець специфічних можливих значень

Можливий такий запис: стовпець **IS** [NOT] NULL
А також функція: **ISNULL**(column, ‘Name instead of NULL’)

Оператор ORDER BY в SQL запитах

Ключове слово **ORDER BY** використовується для того, щоб відсортувати результуючі дані з Бази Даних з однієї чи декількох колонок.

За замовчуванням при використанні оператора **ORDER BY** в в SQL запитах відбувається зростаючий порядок записів. Для вибірки спадаючого порядку записів, можна використати ключове слово **DESC**.

Синтаксис оператора ORDER BY в SQL запитах

```
SELECT column_name,column_name
FROM table_name
ORDER BY column_name,column_name ASC|DESC;
```

Приклад вибірки стовпця даних за допомогою оператора ORDER BY

Представлений нижче вираз вибирає дані з таблиці **"Customers"**, що відсортировані за умовою "Country" (країни).

```
Приклад:
SELECT * FROM Customers
ORDER BY Country;
```

Приклад вибірки стовпця даних за допомогою оператора ORDER BY та ключового слова DESC

Представлений нижче вираз вибирає дані з таблиці **"Customers"**, що відсортировані за умовою "Country" (країни) в спадаючому порядку.

```
Приклад:
SELECT * FROM Customers
ORDER BY Country DESC;
```

Приклад вибірки стовпця даних за допомогою оператора ORDER BY для декількох колонок

```
Приклад:
SELECT * FROM Customers
ORDER BY Country,CustomerName;
```

СИНТАКСИС INSERT INTO В SQL ЗАПИТАХ

SQL запит **INSERT INTO** використовується для внесення нового запису в таблицю даних.

Є можливість добавити запис двома способами:

1. Перша форма запису, не вказує стовпця куди необхідно внести дані, а лише їхні значення:

```
INSERT INTO table_name
VALUES (value1,value2,value3,...);
```

2. Інша форма запису вказує і колонку і значення, що необхідно внести:

```
INSERT INTO table_name (column1,column2,column3,...)
VALUES (value1,value2,value3,...);
```

Розглянемо наступну таблицю Бази Даних.
Таблиця **"Customers"**(Споживачі).

ID	CustomerName	ContactName	Address
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57
2	Ana Trujillo	Ana Trujillo	Constitucion 2222
3	Antonio Moreno	Antonio Moreno	Mataderos 2312
4	Around the Horn	Thomas Hardy	120 Hanover Sq.
5	Berglunds Snabbkop	Christina Berglund	Berguvsvagen 8

Приклад внесення даних за допомогою SQL запиту INSERT INTO

Якщо ми бажаємо внести нові дані в таблицю, то маємо написати такий вираз:

```
Приклад:
INSERT INTO Customers (CustomerName, ContactName, Address)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21');
```

 **Ви звернули увагу, що в SQL запиті INSERT INTO не було прописано ніякого номеру для**

поля ID ?
Поле ID таблиці автоматично оновлюється з унікальним номером для кожного запису таблиці даних.

Внесення даних лише в окремі поля запису

Можливо внести дані лише в окремі поля таблиці.

Наступний вираз вносить дані в поле таблиці **"Customers"** (Споживачі) лише в поле**"CustomerName** (**CustomerID** звичайно обновиться автоматично).

Приклад:
INSERT INTO Customers (CustomerName)
VALUES ('Cardinal');

ФУНКЦІЇ SQL

Елемент SQL	Призначення
AVG()	Функція розрахунку середнього значення SELECT AVG(column_name) FROM table_name
COUNT()	Функція розрахунку кількості записів рядків SELECT COUNT(column_name) FROM table_name;
FIRST()	Визначення першого елементу стовпця Баних Даних SELECT column_name FROM table_name ORDER BY column_name ASC LIMIT 1;
LAST()	Визначення останнього елементу стовпця Баних Даних SELECT column_name FROM table_name ORDER BY column_name DESC LIMIT 1;
MAX()	Функція розрахунку найбільшого значення вибраної стовпця SELECT MAX(column_name) FROM table_name;
MIN()	Функція розрахунку найменшого значення вибраної стовпця SELECT MIN(column_name) FROM table_name;
SUM()	Функція розрахунку суми елементів стовпця Баних Даних SELECT SUM(column_name) FROM table_name;
GROUP BY	Об'єднання агрегатних функцій за групами результуючих даних SELECT column_name, aggregate_function(column_name) FROM table_name WHERE column_name operator value GROUP BY column_name;
Функції SQL	SQL має багато вбудованих функцій для виконання операцій з даними.

HAVING	Параметр HAVING визначає умови вибірки даних агрегатних функцій SQL, які визначені спільним признаком GROUP BY. SELECT column_name, aggregate_function(column_name) FROM table_name WHERE column_name operator value GROUP BY column_name HAVING aggregate_function(column_name) operator value;
UCASE ()	Функція UCASE() перетворює значення поля у верхній регістр SELECT UCASE(column_name) FROM table_name;
LCASE ()	Функція LCASE() перетворює значення поля у нижній регістр SELECT LCASE(column_name) FROM table_name;
MID()	Функція MID() застосовується, якщо необхідно вибрати вказану кількість символів з поля таблиці Бази Даних. SELECT MID(column_name,start[,length]) AS some_name FROM table_name;
LEN()	Функція LEN() повертає кількість символів текстового поля таблиці Бази Даних. SELECT LEN(column_name) FROM table_name;
ROUND()	Функція ROUND() округлює числове значення до вказаної в умові кількості знаків після коми. SELECT ROUND(column_name,decimals) FROM table_name;
NOW()	Функція NOW() повертає поточне значення дати та часу. SELECT NOW() FROM table_name;
FORMAT()	SQL-функція FORMAT() вказує в якому форматі мають бути виведені значення даних. SELECT FORMAT(column_name,format) FROM table_name;

В таблиці вказані найбільш важливіші функції часу при застосуванні Бази даних MySQL:

Функція	**Призначення**
NOW()	Повертає поточний день та час
CURDATE()	Повертає поточну дату
CURTIME()	Повертає поточний час
DATE()	Вилучає порцію дати з даних формату дата та дата/час
EXTRACT()	Повертає просто частину дати/часу
DATE_ADD()	Додає проміжок часу додати
DATE_SUB()	Віднімає вказаний інтервал часу від дати
DATEDIFF()	Повертає кількість днів між двома датами
DATE_FORMAT()	Відображає дату/час в різних форматах

Завдання 1.
(В звіті - виконання кожної дії представити скріншотом)

1) Створити базу даних. Створити в ній таблицю «співробітник», де вказати наступну інформацію: табельний номер, прізвище, ім’я та по-батькові, стать, посада, рік народження, стаж роботи. До існуючого відношення (таблиці) додати атрибут (стовпець) "домашня адреса" (команда ALTER TABLE). Ввести принаймні 5 записів в таблицю (команда INSERT INTO). • Знайти всіх співробітників пенсійного віку. • Знайти інформацію про всіх співробітників з найбільшим стажем роботи.

2) Створити базу даних. Створити в ній таблицю «комп’ютерне обладнання», де вказати наступну інформацію: інвентарний номер, найменування, балансова вартість, дата придбання. До існуючого відношення (таблиці) додати атрибут (стовпець) "кількість" (команда ALTER TABLE). Ввести принаймні 5 записів в таблицю (команда INSERT INTO). • Знайти інформацію про все обладнання, балансова вартість якого найбільша. • Знайти обладнання, загальна вартість (кількість * балансова вартість) якого є найменшою.

3) Створити базу даних. Створити в ній таблицю «студент», де вказати наступну інформацію: факультет, спеціальність, рік вступу, прізвище, ім’я та по-батькові, рік народження. До існуючого відношення (таблиці) додати атрибут (стовпець) "домашня адреса" (команда ALTER TABLE). Ввести принаймні 5 записів в таблицю (команда INSERT INTO). • Знайти інформацію про всіх студентів, що найдовше вчаться. • Знайти наймолодших по віку студентів.

4) Створити базу даних. Створити в ній таблицю «стипендія», де вказати наступну інформацію: факультет, курс, прізвище, ім’я та по-батькові, оцінка предмет1, оцінка предмет2. До існуючого відношення (таблиці) додати атрибут (стовпець) "оцінка предмет3" (команда ALTER TABLE). Ввести принаймні 5 записів в таблицю (команда INSERT INTO). • Знайти студентів з максимальною оцінкою за предмет3. • Знайти інформацію про всіх студентів, що мають успішність вищу за середню.

5) Створити базу даних. Створити в ній таблицю «книга», де вказати наступну інформацію: автор, назва, рік видання, кількість за обліком. До існуючого відношення (таблиці) додати атрибут (стовпець) "ціна" (команда ALTER TABLE). Ввести принаймні 5 записів в таблицю (команда INSERT INTO). • Знайти інформацію про всі книги, що мають найбільшу ціну. • Знайти найстарішого видання книги з найменшою кількістю за обліком.

6) Створити базу даних. Створити в ній таблицю «автомобіль», де вказати наступну інформацію: державний номер, марка, колір, рік випуску, літраж двигуна. До існуючого відношення (таблиці) додати атрибут (стовпець) "пробіг" (команда ALTER TABLE). Ввести принаймні 5 записів в таблицю (команда INSERT INTO). Знайти інформацію про всі автомобілі з найбільшим пробігом. • Знайти всі автомобілі червоного кольору найстарішого року випуску.

7) Створити базу даних. Створити в ній таблицю «книги», де вказати наступну інформацію: автор, назва, кількість за обліком, кількість в наявності. До існуючого відношення (таблиці) додати атрибут (стовпець) "рівень попиту" (команда ALTER TABLE). Ввести принаймні 5 записів в таблицю (команда INSERT INTO). • Знайти інформацію про всі книги, що мають найвищий попит. • Знайти всі книги, що мають найменшу кількість в наявності.

8) Створити базу даних. Створити в ній таблицю «готельні номери», де вказати наступну інформацію: номер, категорія номеру, поверх, кількість кімнат, опис. До існуючого відношення (таблиці) додати атрибут (стовпець) "ціна" (команда ALTER TABLE). Ввести принаймні 5 записів в таблицю (команда INSERT INTO). • Визначити найменшу вартість номерів, що знаходяться на останньому поверсі. • Знайти інформацію про всі номери, найвищої категорії з найбільшою кількістю кімнат.

9) Створити базу даних. Створити в ній таблицю «продовольчі товари», де вказати наступну інформацію: найменування товару, номер партії, обсяг постачання, ціна за одиницю, дата постачання. До існуючого відношення (таблиці) додати атрибут (стовпець) "постачальник" (команда ALTER TABLE). Ввести принаймні 5 записів в таблицю (команда INSERT INTO). • Визначити товар з найбільшим обсягом постачання та найменшої сукупною вартістю постання товару. • Знайти інформацію про всі товари, що постачались протягом 2017 року, та обсяги постачання цих товарів.

10) Створити базу даних. Створити в ній таблицю «автопродаж», де вказати наступну інформацію: марка автомобіля, дата постачання, номер партії, кількість, ціна закупки. До існуючого відношення (таблиці) додати атрибут (стовпець) "ціна продажу" (команда ALTER TABLE). Ввести принаймні 5 записів в таблицю (команда INSERT INTO). • Визначити найнижчу ціну закупки автомобілів в кожній партії. • Знайти інформацію про 2 автомобіля (команда LIMIT), що приносять найбільший дохід.

11) Створити базу даних. Створити в ній таблицю «співробітник», де вказати наступну інформацію: табельний номер, прізвище, ім’я та по-батькові, стать, посада, рік народження, стаж роботи. До існуючого відношення (таблиці) додати атрибут (стовпець) "домашня адреса" (команда ALTER TABLE). Ввести принаймні 5 записів в таблицю (команда INSERT INTO). • Знайти всіх співробітників пенсійного віку. • Знайти інформацію про всіх співробітників з найбільшим стажем роботи.

12) Створити базу даних. Створити в ній таблицю «комп’ютерне обладнання», де вказати наступну інформацію: інвентарний номер, найменування, балансова вартість, дата придбання. До існуючого відношення (таблиці) додати атрибут (стовпець) "кількість" (команда ALTER TABLE). Ввести принаймні 5 записів в таблицю (команда INSERT INTO). • Знайти інформацію про все обладнання, балансова вартість якого найбільша. • Знайти обладнання, загальна вартість (кількість * балансова вартість) якого є найменшою.

13) Створити базу даних. Створити в ній таблицю «студент», де вказати наступну інформацію: факультет, спеціальність, рік вступу, прізвище, ім’я та по-батькові, рік народження. До існуючого відношення (таблиці) додати атрибут (стовпець) "домашня адреса" (команда ALTER TABLE). Ввести принаймні 5 записів в таблицю (команда INSERT INTO). • Знайти інформацію про всіх студентів, що найдовше вчаться. • Знайти наймолодших по віку студентів.

14) Створити базу даних. Створити в ній таблицю «стипендія», де вказати наступну інформацію: факультет, курс, прізвище, ім’я та по-батькові, оцінка предмет1, оцінка предмет2. До існуючого відношення (таблиці) додати атрибут (стовпець) "оцінка предмет3" (команда ALTER TABLE). Ввести принаймні 5 записів в таблицю (команда INSERT INTO). • Знайти студентів з максимальною оцінкою за предмет3. • Знайти інформацію про всіх студентів, що мають успішність вищу за середню.

15) Створити базу даних. Створити в ній таблицю «книга», де вказати наступну інформацію: автор, назва, рік видання, кількість за обліком. До існуючого відношення (таблиці) додати атрибут (стовпець) "ціна" (команда ALTER TABLE). Ввести принаймні 5 записів в таблицю (команда INSERT INTO). • Знайти інформацію про всі книги, що мають найбільшу ціну. • Знайти найстарішого видання книги з найменшою кількістю за обліком.

16) Створити базу даних. Створити в ній таблицю «автомобіль», де вказати наступну інформацію: державний номер, марка, колір, рік випуску, літраж двигуна. До існуючого відношення (таблиці) додати атрибут (стовпець) "пробіг" (команда ALTER TABLE). Ввести

принаймні 5 записів в таблицю (команда INSERT INTO). Знайти інформацію про всі автомобілі з найбільшим пробігом. • Знайти всі автомобілі червоного кольору найстарішого року випуску.

17) Створити базу даних. Створити в ній таблицю «книги», де вказати наступну інформацію: автор, назва, кількість за обліком, кількість в наявності. До існуючого відношення (таблиці) додати атрибут (стовпець) "рівень попиту" (команда ALTER TABLE). Ввести принаймні 5 записів в таблицю (команда INSERT INTO). • Знайти інформацію про всі книги, що мають найвищий попит. • Знайти всі книги, що мають найменшу кількість в наявності.

18) Створити базу даних. Створити в ній таблицю «готельні номери», де вказати наступну інформацію: номер, категорія номеру, поверх, кількість кімнат, опис. До існуючого відношення (таблиці) додати атрибут (стовпець) "ціна" (команда ALTER TABLE). Ввести принаймні 5 записів в таблицю (команда INSERT INTO). • Визначити найменшу вартість номерів, що знаходяться на останньому поверсі. • Знайти інформацію про всі номери, найвищої категорії з найбільшою кількістю кімнат.

19) Створити базу даних. Створити в ній таблицю «продовольчі товари», де вказати наступну інформацію: найменування товару, номер партії, обсяг постачання, ціна за одиницю, дата постачання. До існуючого відношення (таблиці) додати атрибут (стовпець) "постачальник" (команда ALTER TABLE). Ввести принаймні 5 записів в таблицю (команда INSERT INTO). • Визначити товар з найбільшим обсягом постачання та найменшої сукупною вартістю постання товару. • Знайти інформацію про всі товари, що постачались протягом 2017 року, та обсяги постачання цих товарів.

20) Створити базу даних. Створити в ній таблицю «автопродаж», де вказати наступну інформацію: марка автомобіля, дата постачання, номер партії, кількість, ціна закупки. До існуючого відношення (таблиці) додати атрибут (стовпець) "ціна продажу" (команда ALTER TABLE). Ввести принаймні 5 записів в таблицю (команда INSERT INTO). • Визначити найнижчу ціну закупки автомобілів в кожній партії. • Знайти інформацію про 2 автомобіля (команда LIMIT), що приносять найбільший дохід.

Завдання 2.

Додати нову таблицю в розроблену базу даних.
Самостійно придумати запити для оновлення та видалення з бази даних:
• записів,
• певних атрибутів (стовпців),
• таблиць.
Експериментувати з новоствореною таблицею.

Завдання 3.

В розробленій базі даних створити допоміжну та похідну таблицю, зв’язану з основною таблицею. Виконати запит, що передбачає співставлення значень полів (через оператор WHERE).

Завдання 4.

Потрібно детально опрацювати та потренуватись із запитами до БД на сайті https://www.w3schools.com/sql/sql_select.asp

В ОБОВ'ЯЗКОВОМУ ПОРЯДКУ!
Запити повинні бути складні, повинні відрізнятися від заданих прикладів на сайті, повинні включати оператори:

- SQL WHERE Clause,
- SQL LIKE Operator,
- SQL ORDER BY,
- SQL HAVING Clause,
- SQL Joins,
- MySQL Functions (протестувати 10 – 15 різних функцій)

РЕСУРСИ ДЛЯ ДОПОМОГИ:

http://moonexcel.com.ua/%D1%83%D1%80%D0%BE%D0%BA%D0%B8-sql_ua (уроки-sql_ua)
http://bestwebit.biz.ua/w3c_1/mysql_w3c_syntax.php
http://dev.mysql.com/doc/refman/5.7/en/sql-syntax-data-definition.html
http://www.tutorialspoint.com/sql/sql-useful-functions.htm