**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА**

**ПОЛІТЕХНІКА”**

Кафедра систем штучного інтелекту



**Лабораторна робота 2**

з організації баз даних та знать

**Виконав:**

Студент групи КН-208

Воробель Адріан

**Викладач:**

Якимишин Х.М.

Львів – 2019р.

**Мета роботи:** Побудувати даталогічну модель бази даних; визначити типи, розмірності та обмеження полів; визначити обмеження таблиць; розробити SQL запити для створення спроектованих таблиць.

**Короткі теоретичні відомості.**

Щоб створити нову базу даних у командному рядку клієнта MySQL (mysql.exe) слід виконати команду CREATE DATABASE, опис якої подано нижче. Тут і надалі, квадратні дужки позначають необов’язковий аргумент команди, символ "|" позначає вибір між аргументами.

**CREATE {DATABASE | SCHEMA}** [IF NOT EXISTS] *ім’я\_бази*

[[DEFAULT] CHARACTER SET *кодування*]

[[DEFAULT] COLLATE *набір\_правил*]

*ім’я\_бази* – назва бази даних (латинські літери і цифри без пропусків); *кодування* – набір символів і кодів (koi8u, latin1, utf8, cp1250 тощо); *набір\_правил* – правила порівняння рядків символів (див. результат команди

show collation).

Нижче наведені деякі допоміжні команди для роботи в СУБД MySQL. Кожна команда і кожен запит в командному рядку повинні завершуватись розділяючим символом ";".

1. Перегляд існуючих баз даних:

SHOW DATABASES

2. Вибір бази даних для подальшої роботи:

USE DATABASE *ім’я\_бази*

3. Перегляд таблиць в базі даних:

SHOW TABLES [FOR *ім’я\_бази*]

4. Перегляд опису таблиці в базі:

DESCRIBE *ім’я\_таблиці*

5. Виконати набір команд з зовнішнього файлу:

SOURCE *назва\_файлу*

6. Вивести результати виконання подальших команд у зовнішній файл:

\T *назва\_файлу*

Для роботи зі схемою бази даних існують такі основні команди:

ALTER DATABASE – зміна опису бази даних; CREATE TABLE – створення нової таблиці; ALTER TABLE – зміна структури таблиці; DELETE TABLE – видалення таблиці з бази даних;

CREATE INDEX – створення нового індексу (для швидкого пошуку даних);

DROP INDEX – видалення індексу;

DROP DATABASE – видалення бази даних.

Розглянемо команду створення таблиці в MySQL та її основні аргументи.

**CREATE** [TEMPORARY] **TABLE** [IF NOT EXISTS] ім’я\_таблиці

[**(**опис\_таблиці,...**)**] [додаткові\_параметри] ... [вибірка\_даних]

**опис\_таблиці**: назва\_поля опис\_поля

| [CONSTRAINT [ім’я\_обмеження]] PRIMARY KEY (назва\_поля,...) [тип\_обмеження]

| {INDEX|KEY} [ім’я\_обмеження] (назва\_поля,...)[ тип\_обмеження]

| [CONSTRAINT [ім’я\_обмеження]] UNIQUE [INDEX|KEY] [ім’я\_обмеження](назва\_поля,...) [тип\_обмеження]

| {FULLTEXT|SPATIAL} [INDEX|KEY] [ім’я\_обмеження] (назва\_поля,...)

[тип\_обмеження]

| [CONSTRAINT [ім’я\_обмеження]] FOREIGN KEY [ім’я\_обмеження] (назва\_поля,...) опис\_зв’язку

| CHECK (вираз)

**опис\_поля:**

тип\_даних [NOT NULL | NULL] [DEFAULT *значення\_за\_замовчуванням*] [AUTO\_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]

**опис\_зв’язку:**

REFERENCES *ім’я\_таблиці* (назва\_поля, ...) [ON DELETE дія]

[ON UPDATE дія]

**дія:**

CASCADE

Одночасне видалення, або оновлення відповідного значення у зовнішній таблиці.

RESTRICT

Аналог NO ACTION. Дія над значенням поля ігнорується, якщо існує відповідне йому значення у зовнішній таблиці. Опція задана за замовчуванням.

SET NULL

При дії над значенням у первинній таблиці, відповідне значення у зовнішній таблиці замінюється на NULL.

**додаткові\_параметри:**

{ENGINE|TYPE} [=] тип\_таблиці

| AUTO\_INCREMENT [=] *значення\_приросту\_лічильника*

| AVG\_ROW\_LENGTH [=] *значення*

| [DEFAULT] CHARACTER SET [=] кодування

| CHECKSUM [=] {0 | 1}

| [DEFAULT] COLLATE [=] *набір\_правил*

| COMMENT [=] 'коментар до таблиці'

| DATA DIRECTORY [=] 'абсолютний шлях'

| DELAY\_KEY\_WRITE [=] {0 | 1}

| INDEX DIRECTORY [=] 'абсолютний шлях'

| MAX\_ROWS [=] *значення*

| MIN\_ROWS [=] *значення*

| ROW\_FORMAT {DEFAULT|DYNAMIC|FIXED|COMPRESSED|REDUNDANT|COMPACT}

**вибірка\_даних:**

[IGNORE | REPLACE] [AS] SELECT ... (вибір даних з інших таблиць)

**вираз:**

Логічний вираз, що повертає TRUE або FALSE.

**Опис аргументів:**

ім’я\_таблиці

Назва таблиці. Або назва\_бази.назва\_таблиці.

тип\_таблиці

В MySQL крім типів таблиць MyISAM та InnoDB існують типи MEMORY, BDB, ARCHIVE тощо.

тип\_обмеження

Задає тип індексу для ключового поля: USING {BTREE | HASH | RTREE}.

TEMPORARY

Створення тимчасової таблиці, яка буде знищена після завершення зв’язку з сервером.

CONSTRAINT

Вказує на початок оголошення PRIMARY KEY, UNIQUE, або FOREIGN KEY

обмеження.

NULL | NOT NULL

Директива, що дозволяє/забороняє null-значення для даного поля.

PRIMARY KEY

Вказує, що дане поле буде первинним ключем в таблиці.

UNIQUE

Вказує на те, що в даному полі будуть зберігатися унікальні значення.

FOREIGN KEY ... REFERENCES

Створює зовнішній ключ, зв’язаний із вказаним полем (полями).

AVG\_ROW\_LENGTH

Приблизне значення середньої довжини рядків зі змінною довжиною.

DATA DIRECTORY

Вказує шлях, за яким таблиця має зберігатись у файловій системі.

CHECKSUM

Якщо параметр = 1, то для рядків таблиці буде рахуватись контрольна сума. Це сповільнює оновлення таблиці, але робить легшим пошук пошкоджених таблиць.

ROW\_FORMAT

Вказує на спосіб зберігання рядків таблиці (залежно від типу таблиці).

FULLTEXT|SPATIAL

Тип індексу (повнотекстовий/просторовий; тільки для таблиць типу MyISAM).

|  |  |
| --- | --- |
| Основні типи даних у СУБД MySQL: Тип даних | Опис |
| TINYINT[(k)] [UNSIGNED] | Ціле число з *k*-біт: -127 .. 128. UNSIGNED: 0 .. 255. |
| BOOL | Логічний тип (1-бітне число). Число 0 – фальш,  відмінне від нуля – істина. |
| SMALLINT[(k)] [UNSIGNED] | Ціле число з *k*-біт: -32768 .. 32767.  UNSIGNED: 0 .. 65535. |
| MEDIUMINT[(k)] [UNSIGNED] | Ціле число з *k*-біт: -8388608 .. 8388607.  UNSIGNED: 0 .. 16777215. |
| INT[(k)] [UNSIGNED] | Ціле число з *k*-біт: -2147483648 .. 2147483647.  UNSIGNED: 0 .. 4294967295. |
| BIGINT[(k)] [UNSIGNED] | -9223372036854775808 .. 9223372036854775807.  UNSIGNED: 0 .. 18446744073709551615. |
| SERIAL | Синонім для типу BIGINT UNSIGNED NOT NULL AUTO\_INCREMENT UNIQUE |
| FLOAT[(n,m)] [UNSIGNED] | Число з плаваючою крапкою, де *n* – кількість всіх цифр, *m* – кількість цифр після крапки.  Від -3.402823466E+38 до -1.175494351E-38  UNSIGNED: 1.175494351E-38 .. 3.402823466E+38 |
| DOUBLE[(n,m)] [UNSIGNED] | Від -1.7976931348623157E+308 до  -2.2250738585072014E-308  UNSIGNED: від 2.2250738585072014E-308 до 1.7976931348623157E+308. |
| DECIMAL[(n[,m])] [UNSIGNED] | Число з фіксованою крапкою. *n* – кількість цифр (максимально – 65), *m* – кількість цифр після крапки (максимально – 30, за замовчуванням – 0).  UNSIGNED: від’ємні значення заборонені. |
| DATE | Дата. Від "1000-01-01" до "9999-12-31". |
| DATETIME | Дата і час.  Від "1000-01-01 00:00:00" до "9999-12-31 23:59:59". |
| TIMESTAMP | Часова мітка. Може присвоюватись автоматично.  Від "1970-01-01 00:00:01" до "2038-01-09 03:14:07" |
| TIME | Час у форматі "HH:MM:SS" (рядок або число). |
| CHAR[(n)] | Рядок з *n*-символів (макс. – 255, за замовчуванням – 1). |
| VARCHAR(n) | Рядок змінної довжини. Для кодування utf8  максимальна довжина складає 21844 символи. |
| TEXT(n) | Рядок змінної довжини. Максимальна кількість  однобайтових символів – 65535. |
| MEDIUMTEXT | 16777215 однобайтових символів (16 Мб тексту). |
| BLOB | Бінарні дані (65535 байт). |
| MEDIUMBLOB | Бінарні дані (16 Мб) |
| LONGBLOB | Бінарні дані (4 Гб, залежно від налаштувань системи) |
| ENUM('знач1','знач2',...) | Перелік значень. Зберігається лише одне. |
| SET('знач1','знач2',...) | Множина значень. Зберігається одне, або більше  (максимально – 64). |

Можна дати декілька порад щодо розробки схеми бази даних і вибору типів даних. Вони дозволять уникнути повільного виконання запитів і потреби модифікації таблиць в майбутньому.

* Слід використовувати якомога менший тип даних для полів таблиць. Наприклад, для зберігання чисел від 1 до 64 краще використати тип TINYINT(6) замість SMALLINT. Це впливає на швидкість пошуку і вибірки даних.
* Слід використовувати рядки фіксованої довжини, якщо це можливо. Для цього всі поля таблиці повинні бути фіксованої довжини. Тобто, варто уникати типів VARCHAR, TEXT і BLOB. Це пришвидчить вибірку даних з середини рядків, оскільки ці дані будуть мати фіксовану адресу. При потребі використання полів з типами TEXT або BLOB, їх можна виділити в окрему таблицю.
* Якщо можливо, варто завжди використовувати поля з обмеженням NOT NULL. Хоча це може збільшувати об’єм бази на диску.
* MySQL дозволяє використовувати різні типи таблиць в одній базі даних. Слід використовувати переваги різних типів (MyISAM, INODB тощо) залежно від характеру майбутнього використання таблиці.
* Потрібно створювати індекси, які пришвидчать пошук і вибірку даних.
* В рідкісних випадках можна денормалізувати схему з метою зменшення кількості операцій з об’єднання таблиць при складних запитах. Але при цьому ускладнюється задача збереження цілісності бази даних.

**Хід роботи.**

Взявши за основу модельовану схему БД я приступив то її створення.

Спочатку створив БД під назвою AutoShop та встановив кодування utf8.

CREATE DATABASE IF NOT EXISTS AutoShop CHARACTER SET utf8;

Зайшов у контекст створеної БД

USE AutoShop;

Створив першу таблицю User

CREATE TABLE IF NOT EXISTS Users

Та надав їй поля та їх властивості

(

id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

login VARCHAR(18) NOT NULL,

pass VARCHAR(10) NOT NULL,

PRIMARY KEY(id)

);

Поле id отримало властивість первинного ключа, тип даних – цілі числа, не пусте, автоінкремент(буде збільшуватись на 1 при кожному новому рядку в таблиці); login – типом даних є набір символів(до 18), не пусте; pass – типом даних є набір символів(до 10), не пусте.

Надалі поле id з однаковим набором властивостей буде зустрічатись в кожній таблиці.

Наступна частина скрипта демонструє створення таблиці із зовнішнім ключем та зв’язком один до одного

CREATE TABLE IF NOT EXISTS UsersContacts(

id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

u\_name VARCHAR(30) NOT NULL,

u\_surname VARCHAR(30),

u\_email VARCHAR(35) NOT NULL,

u\_phone VARCHAR(13),

user\_id INT UNSIGNED NOT NULL,

PRIMARY KEY(id),

CONSTRAINT fk\_contact\_user FOREIGN KEY(user\_id)

REFERENCES AutoShop.Users(id)

ON DELETE CASCADE ON UPDATE CASCADE

);

Таблиця UserContacts має зв’язуватись із таблицею User. Тому в таблиці присутнє поле user\_id, яке є зовнішнім ключем FOREIGN KEY(user\_id) який посилається на поле id таблиці Users REFERENCES AutoShop.Users(id) та володіє властивістю каскадного видалення та оновлення ON DELETE CASCADE ON UPDATE CASCADE.

Наступний блок скрипта продемонструє таблицю із зв’язком багато до багатьох

CREATE TABLE IF NOT EXISTS CarDistributor(

id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

car\_id INT UNSIGNED NOT NULL,

distrib\_id INT UNSIGNED NOT NULL,

PRIMARY KEY(id),

CONSTRAINT fk\_car\_carDistrib FOREIGN KEY(car\_id)

REFERENCES AutoShop.Car(id)

ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT fk\_distrib\_carDistrib FOREIGN KEY(distrib\_id)

REFERENCES AutoShop.Distributor(id)

ON DELETE CASCADE ON UPDATE CASCADE

);

Тут створені два зовнішніх ключа, які посилаються на два різних поля id в таблицях carDistrib та Distributor відповідно. Вони володіють такими ж властивостями каскадного видалення та оновлення.

Повний скрипт БД

CREATE DATABASE IF NOT EXISTS AutoShop

CHARACTER SET utf8;

USE AutoShop;

CREATE TABLE IF NOT EXISTS Users(

id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

login VARCHAR(18) NOT NULL,

pass VARCHAR(10) NOT NULL,

PRIMARY KEY(id)

);

CREATE TABLE IF NOT EXISTS UsersContacts(

id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

u\_name VARCHAR(30) NOT NULL,

u\_surname VARCHAR(30),

u\_email VARCHAR(35) NOT NULL,

u\_phone VARCHAR(13),

user\_id INT UNSIGNED NOT NULL,

PRIMARY KEY(id),

CONSTRAINT fk\_contact\_user FOREIGN KEY(user\_id)

REFERENCES AutoShop.Users(id)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS Car(

id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

mark VARCHAR(20) NOT NULL,

model VARCHAR(30) NOT NULL,

product\_date DATE NOT NULL,

price INT NOT NULL,

car\_type ENUM('Sedan','Universal','Hatchback','Minivan','Crossover',

'Coupe','Cabriolet','Pickup') NOT NULL,

PRIMARY KEY(id)

);

CREATE TABLE IF NOT EXISTS CarDescription(

id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

car\_engine VARCHAR(10) NOT NULL,

wheel\_drive VARCHAR(3) NOT NULL,

sit\_places INT NOT NULL,

trunk INT NOT NULL,

transmission ENUM('Automatic','Manual'),

car\_id INT UNSIGNED NOT NULL,

PRIMARY KEY(id),

CONSTRAINT fk\_description\_car FOREIGN KEY (car\_id)

REFERENCES AutoShop.Car(id)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS Distributor(

id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

distrib\_name VARCHAR(30) NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE IF NOT EXISTS DistributorContacts(

id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

address VARCHAR(40) NOT NULL,

phone VARCHAR(13) NOT NULL,

email VARCHAR(50) NOT NULL,

distrib\_id INT UNSIGNED NOT NULL,

PRIMARY KEY (id),

CONSTRAINT fk\_distrib\_contacts FOREIGN KEY (distrib\_id)

REFERENCES AutoShop.Distributor(id)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS CarDistributor(

id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

car\_id INT UNSIGNED NOT NULL,

distrib\_id INT UNSIGNED NOT NULL,

PRIMARY KEY(id),

CONSTRAINT fk\_car\_carDistrib FOREIGN KEY(car\_id)

REFERENCES AutoShop.Car(id)

ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT fk\_distrib\_carDistrib FOREIGN KEY(distrib\_id)

REFERENCES AutoShop.Distributor(id)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS Orders(

id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

user\_id INT UNSIGNED NOT NULL,

carDistrib\_id INT UNSIGNED NOT NULL,

PRIMARY KEY(id),

CONSTRAINT fk\_orders\_user FOREIGN KEY (user\_id)

REFERENCES AutoShop.Users(id)

ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT fk\_orders\_carDistrib FOREIGN KEY(carDistrib\_id)

REFERENCES AutoShop.CarDistributor(id)

ON DELETE CASCADE ON UPDATE CASCADE

);

**Висновок**

В ході лабораторної роботи я навчився писати скрипт бази даних у середовищі MySQL Workbench. Освоїв частину синтаксису мови MySQL. Реалізував здобуті знання у прикладі описаному вище. Зрозумів схему роботи зовнішніх та первинних ключів, а також зв’язків один до одного, один до багатьох, багато до багатьох.