МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних наук та інформаційних технологій

Кафедра систем штучного інтелекту

Звіт до лабораторної роботи №2

з дисципліни
“ОБДЗ”

Виконав:
ст. гр. КН-211
Качмарик Віктор

Викладач:
Якимишин Х.М.

Львів – 2019

Лабораторна робота №2

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.

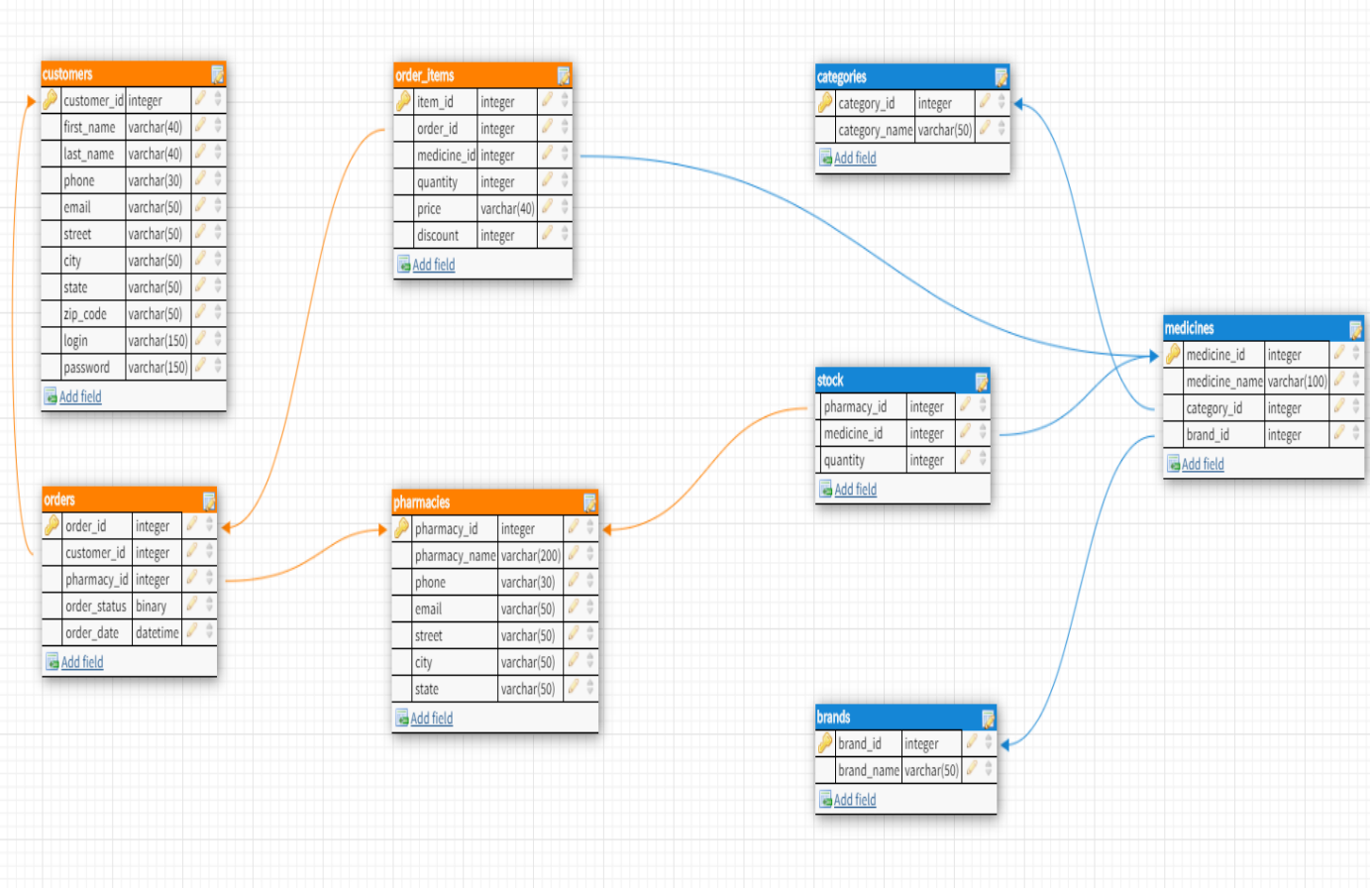
Можна дати декілька порад щодо розробки схеми бази даних і вибору типів даних. Вони дозволять уникнути повільного виконання запитів і потреби модифікації таблиць в майбутньому.

- 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:

Даталогічна модель вимагає визначення конкретних полів бази даних, їхніх типів, обмежень на значення, тощо. На рисунку зображено даталогічну модель проектованої бази даних.



Створимо нову базу даних, виконавши такі команди:

```
CREATE TABLE `pharmacies` (
	`pharmacy_id` INT NOT NULL AUTO_INCREMENT,
	`pharmacy_name` varchar(200) NOT NULL,
	`phone` varchar(30) NOT NULL,
	`email` varchar(50) NOT NULL,
	`street` varchar(50) NOT NULL,
	`city` varchar(50) NOT NULL,
	`state` varchar(50) NOT NULL,
	PRIMARY KEY (`pharmacy_id`)
);

CREATE TABLE `customers` (
	`customer_id` INT NOT NULL AUTO_INCREMENT,
	`first_name` varchar(40) NOT NULL,
	`last_name` varchar(40) NOT NULL,
	`phone` varchar(30) NOT NULL UNIQUE,
	`email` varchar(50) NOT NULL,
	`street` varchar(50) NOT NULL,
	`city` varchar(50) NOT NULL,
	`state` varchar(50) NOT NULL,
	`zip_code` varchar(50) NOT NULL,
	`login` varchar(150) NOT NULL UNIQUE,
	`password` varchar(150) NOT NULL,
	PRIMARY KEY (`customer_id`)
);
```

```
CREATE TABLE `orders` (
	`order_id` INT NOT NULL,
	`customer_id` INT NOT NULL UNIQUE,
	`pharmacy_id` INT NOT NULL UNIQUE,
	`order_status` BINARY NOT NULL,
	`order_date` DATETIME NOT NULL,
	PRIMARY KEY (`order_id`)
);

CREATE TABLE `order_items` (
	`item_id` INT NOT NULL AUTO_INCREMENT,
	`order_id` INT NOT NULL,
	`medicine_id` INT NOT NULL UNIQUE,
	`quantity` INT NOT NULL,
	`price` varchar(40) NOT NULL,
	`discount` INT NOT NULL,
	PRIMARY KEY (`item_id`)
);

CREATE TABLE `medicines` (
	`medicine_id` INT NOT NULL AUTO_INCREMENT,
	`medicine_name` varchar(100) NOT NULL AUTO_INCREMENT,
	`category_id` INT NOT NULL,
	`brand_id` INT NOT NULL,
	PRIMARY KEY (`medicine_id`)
);

CREATE TABLE `categories` (
	`category_id` INT NOT NULL AUTO_INCREMENT,
	`category_name` varchar(50) NOT NULL,
	PRIMARY KEY (`category_id`)
);
```

```
CREATE TABLE `stock` (
    `pharmacy_id` INT NOT NULL,
    `medicine_id` INT NOT NULL,
    `quantity` INT
);

CREATE TABLE `brands` (
    `brand_id` INT NOT NULL AUTO_INCREMENT,
    `brand_name` varchar(50) NOT NULL,
    PRIMARY KEY (`brand_id`)
);
```

ALTER TABLE `orders` ADD CONSTRAINT `orders_fk0` FOREIGN KEY (`customer_id`) REFERENCES `customers`(`customer_id`);

ALTER TABLE `orders` ADD CONSTRAINT `orders_fk1` FOREIGN KEY (`pharmacy_id`) REFERENCES `pharmacies`(`pharmacy_id`);

ALTER TABLE `order_items` ADD CONSTRAINT `order_items_fk0` FOREIGN KEY (`order_id`) REFERENCES `orders`(`order_id`);

ALTER TABLE `order_items` ADD CONSTRAINT `order_items_fk1` FOREIGN KEY (`medicine_id`) REFERENCES `medicines`(`medicine_id`);

ALTER TABLE `medicines` ADD CONSTRAINT `medicines_fk0` FOREIGN KEY (`category_id`) REFERENCES `categories`(`category_id`);

ALTER TABLE `medicines` ADD CONSTRAINT `medicines_fk1` FOREIGN KEY (`brand_id`) REFERENCES `brands`(`brand_id`);

ALTER TABLE `stock` ADD CONSTRAINT `stock_fk0` FOREIGN KEY (`pharmacy_id`) REFERENCES `pharmacies`(`pharmacy_id`);

ALTER TABLE `stock` ADD CONSTRAINT `stock_fk1` FOREIGN KEY (`medicine_id`) REFERENCES `medicines`(`medicine_id`);

Висновок: на цій лабораторній роботі було завершено моделювання і засобами SQL створено базу даних, що складається з восьми таблиць.