TRƯỜNG ĐẠI HỌC KINH TẾ - KỸ THUẬT CÔNG NGHIỆP
KHOA: KHOA HỌC ỨNG DỤNG.

CHƯƠNG 5
THAO TÁC VỚI SQL TRONG PYTHON

Giảng viên: TS.Cao Diệp Thắng
Khoa: Khoa học Ứng dụng

MỤC TIÊU BÀI HỌC

Sau khi học xong bài này, người học sẽ nắm được các vấn đề sau:

+ Ngôn ngữ truy vấn cơ sở dữ liệu SQL.
+ Cách kết nối Python với các cơ sở dữ liệu như SQLite và MySQL, và thực hiện các thao tác truy vấn cơ bản như SELECT, INSERT, UPDATE, DELETE.
+ Cách sử dụng các thư viện như sqlite3 (cho SQLite) và **mysql-connector** (cho MySQL) để thực hiện các thao tác trên cơ sở dữ liệu.
+ Cách tạo và trực quan hóa dữ liệu bằng biểu đồ (bar chart, line chart, pie chart,...) để hiển thị thông tin một cách dễ hiểu và trực quan.

NỘI DUNG BÀI HỌC

5.1 GIỚI THIỆU VỀ SQLITE

5.2 ỨNG DỤNG DB BROWSER For SQLite

5.3 THAO TÁC VỚI CSDL SQLite VỚI PYTHON.

SO SÁNH CÁC HỆ CƠ SỞ DỮ LIỆU PHỔ BIẾN

Tiêu chí	SQLite	MySQL/PostgreSQL (Client-Server)
Kiến trúc	**File-based** (Không cần Server). CSDL là một tệp tin duy nhất trên máy cục bộ.	**Client-Server** (Cần Server). Dữ liệu nằm trên một máy chủ riêng biệt, cần kết nối qua mạng.
Phù hợp	Học tập, thử nghiệm, ứng dụng nhỏ, lưu trữ cấu hình.	Ứng dụng doanh nghiệp, website, hệ thống có nhiều người dùng truy cập đồng thời.
Thư viện Python	**sqlite3** (Built-in, không cần cài đặt thêm)	**mysql-connector-python**, **pymysql** (cho MySQL) hoặc **psycopg2** (cho PostgreSQL) - **Cần cài đặt** (pip install...)
Tham số kết nối	Chỉ cần **đường dẫn file** (database.db)	Cần **Host** (**IP**/Tên miền), **Port**, **User**, **Password**, **Database Name**.

Thực hành. Minh họa kết nối với CSDL Client-Server

Mục đích: **phân biệt khái niệm**:

1.SQLite (File-based): Kết nối chỉ cần đường dẫn đến một file.

2.MySQL/PostgreSQL (Client-Server):

Kết nối cần **xác thực** (**Host**, **User**, **Password**, **DB** Name) để truy cập Server qua mạng.

Nội dung thực hành. Thực thi theo các bước hướng dẫn chi tiết trong file:

Thuc_hanh_Demo_MySQL_over_Docker.pdf

Kết quả chạy code mong đợi.

```
Đang cố gắng kết nối tới MySQL Server...
✅ Kết nối THÀNH CÔNG tới CSDL 'data_science_db' qua cổng 3307.
Thực hiện truy vấn mẫu... Phiên bản MySQL: 9.5.0
Ngày hiện tại trên Server: 2025-11-27 Đã đóng kết nối CSDL.
```

5.1. GIỚI THIỆU VỀ SQLITE

SQLite là một hệ thống quản lý cơ sở dữ liệu quan hệ (**Relational Database Management System - RDBMS**) nhỏ gọn, dựa trên C, được tích hợp sẵn trong hầu hết các ngôn ngữ lập trình, bao gồm cả Python. Nó không yêu cầu cài đặt hoặc cấu hình phức tạp, cho phép bạn lưu trữ và truy vấn dữ liệu một cách đơn giản và hiệu quả.

Điểm nổi bật

- **Không cần máy chủ**: SQLite là một thư viện nhúng, có nghĩa là nó không yêu cầu một máy chủ riêng biệt để hoạt động. Mọi thao tác trên cơ sở dữ liệu được thực hiện trực tiếp trong tệp cơ sở dữ liệu duy nhất, thường có đuôi .sqlite hoặc .db.
- **Nhỏ gọn và tiết kiệm tài nguyên**: Với kích thước nhỏ và ít yêu cầu tài nguyên, SQLite rất phù hợp cho các ứng dụng di động hoặc các dự án nhẹ.
- **Hỗ trợ cơ bản cho các câu lệnh SQL**: SQLite hỗ trợ một số câu lệnh SQL tiêu chuẩn để tạo, đọc, cập nhật và xóa dữ liệu từ cơ sở dữ liệu.
- **Dễ sử dụng:** Việc làm việc với SQLite rất dễ dàng và hầu như không đòi hỏi bất kỳ kiến thức đặc biệt về cơ sở dữ liệu.
- **Giao tiếp với nhiều ngôn ngữ:** SQLite được tích hợp sẵn trong nhiều ngôn ngữ lập trình như Python, C++, Java, Ruby, PHP và nhiều ngôn ngữ khác.
- SQLite thích hợp cho các ứng dụng nhỏ, đơn giản hoặc cần tích hợp cơ sở dữ liệu cơ bản mà không muốn triển khai một hệ thống quản lý cơ sở dữ liệu phức tạp.

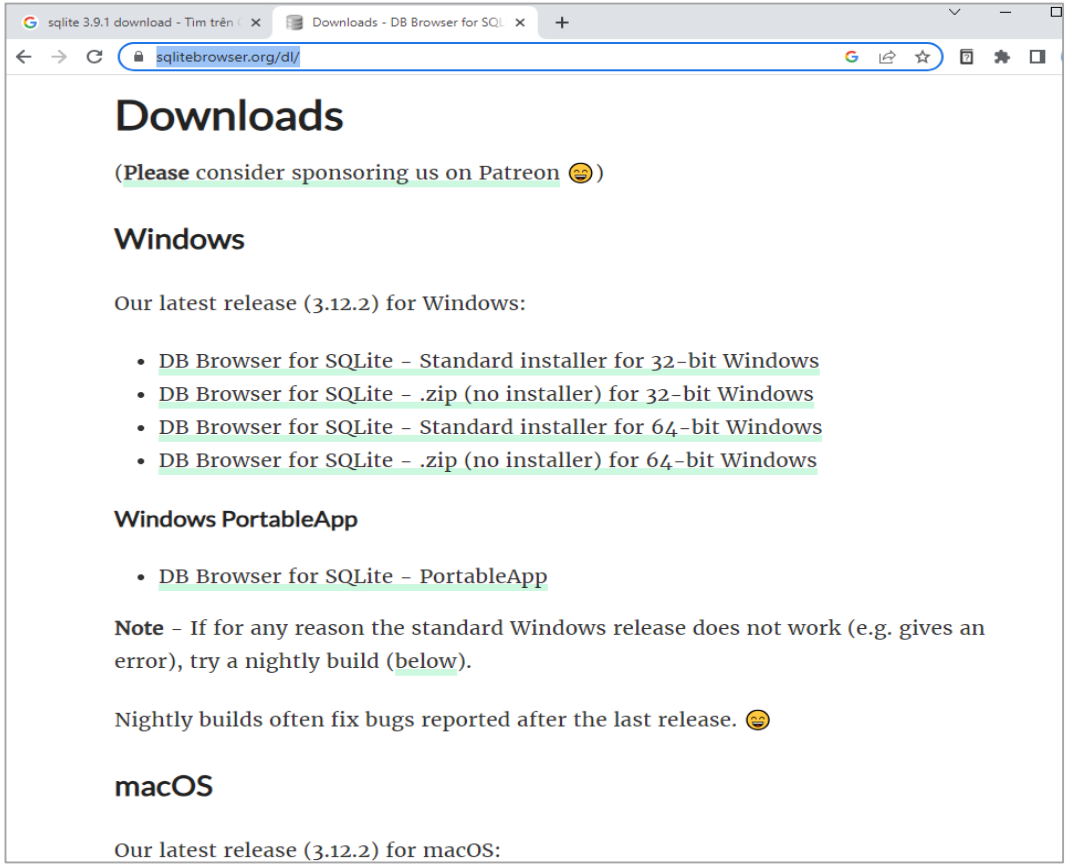
5.2. ỨNG DỤNG DB BROWSER FOR SQLITE

'DB Browser for SQLite', còn được gọi là **'SQLite Database Browser'**, là một công cụ quản lý cơ sở dữ liệu mã nguồn mở dành cho SQLite. Nó cho phép người dùng tạo, truy vấn, duyệt, và quản lý cơ sở dữ liệu SQLite một cách dễ dàng và trực quan. 'DB Browser for SQLite' là một phần mềm miễn phí và rất phổ biến trong cộng đồng phát triển và quản lý cơ sở dữ liệu.

Một số tính năng chính của 'DB Browser for SQLite': xem TLHT trang 244

5.2.2. Cài đặt DB Browser For SQLite

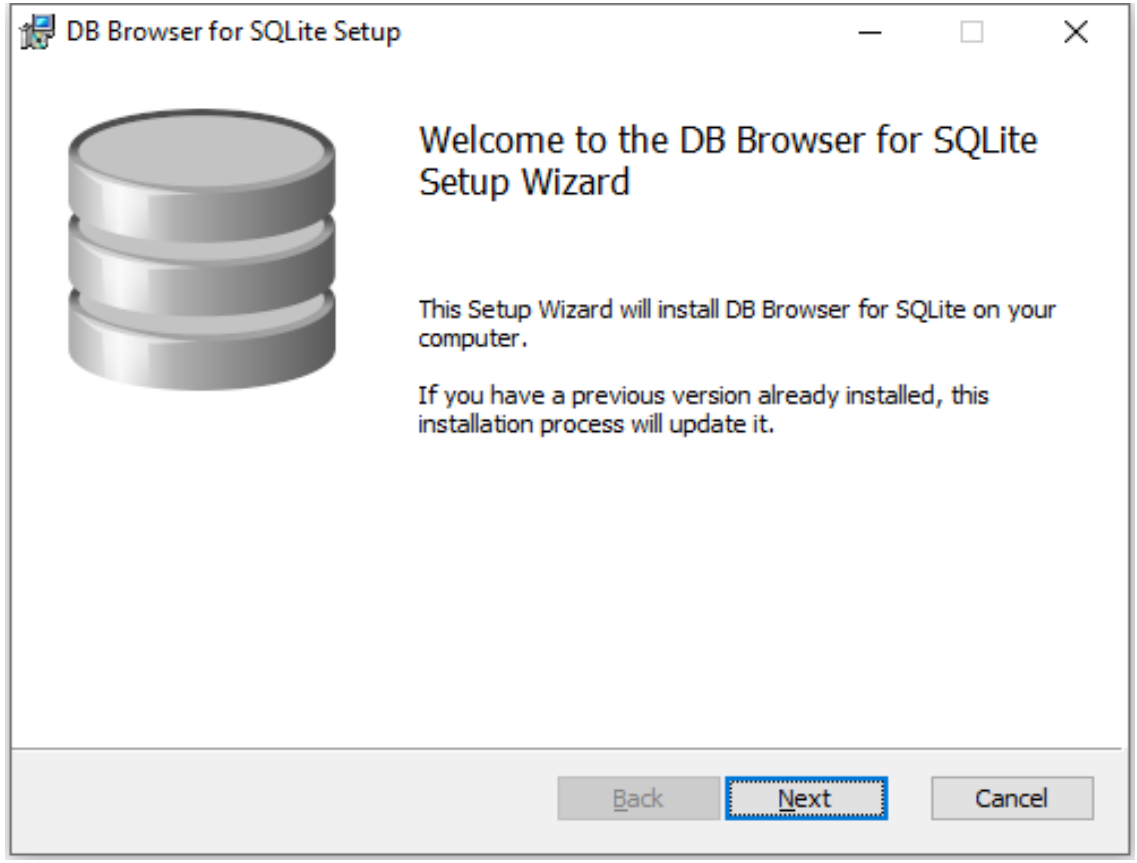
Bước 1. Vào trang https://sqlitebrowser.org/dl/ chọn Download để tải file về và tiến hành cài đặt. Có thể chọn các phiên bản 32/64 bit, trên các hệ điều hành Windows, macOS và Linux,..



Hình 5.1. Tải file cài đặt DB Browser for SQLite.

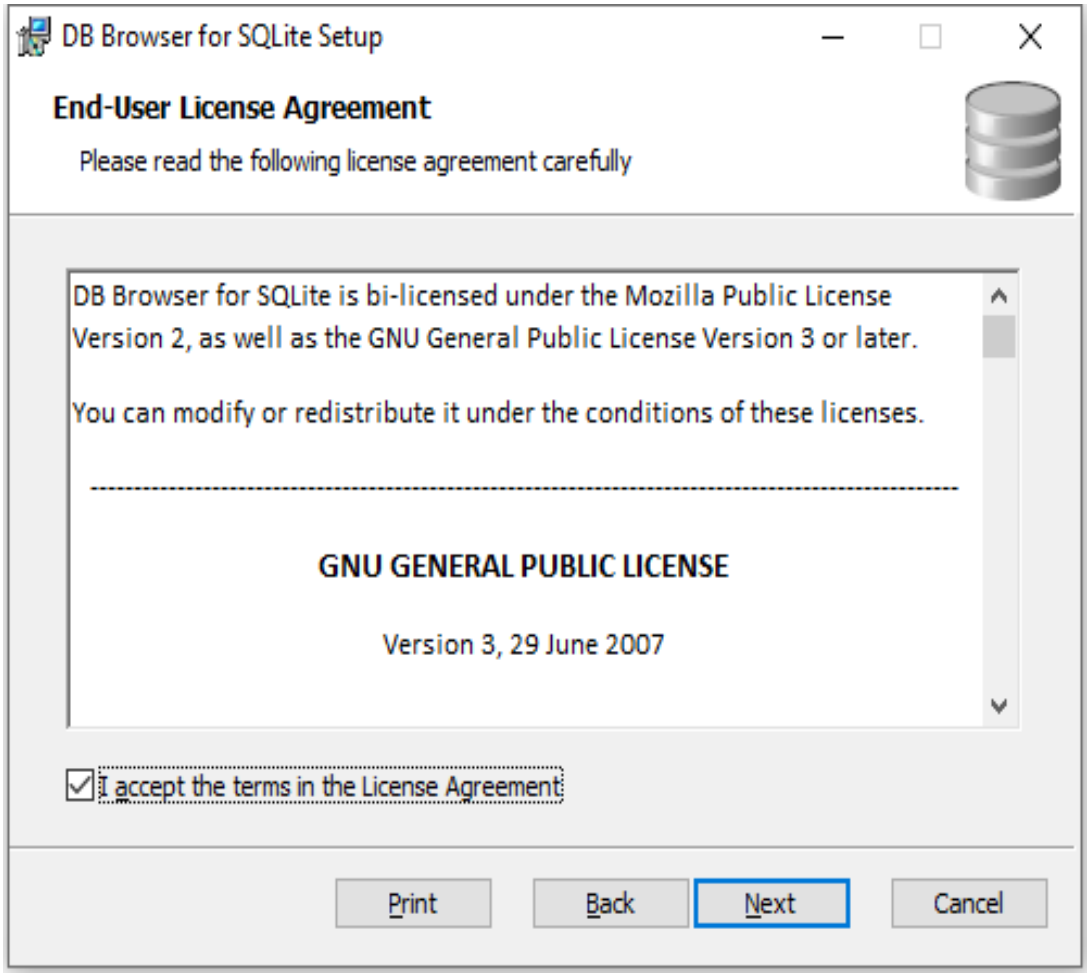
Trong tài liệu này giới thiệu cài đặt bản DB Browser for SQLite phiên bản tiêu chuẩn cho HĐH Windows 64bit. File cài đặt DB.Browser.for.SQLite-3.12.2-win64.msi, kích thước ~ 17MB.

Bước 2: Trong thư mục chứa file cài đặt DB.Browser.for.SQLite-3.12.2-win64.msi, tiến hành chọn file, nhấp phím phải (right click) → chọn install. Sau đó làm theo các bước hướng dẫn.



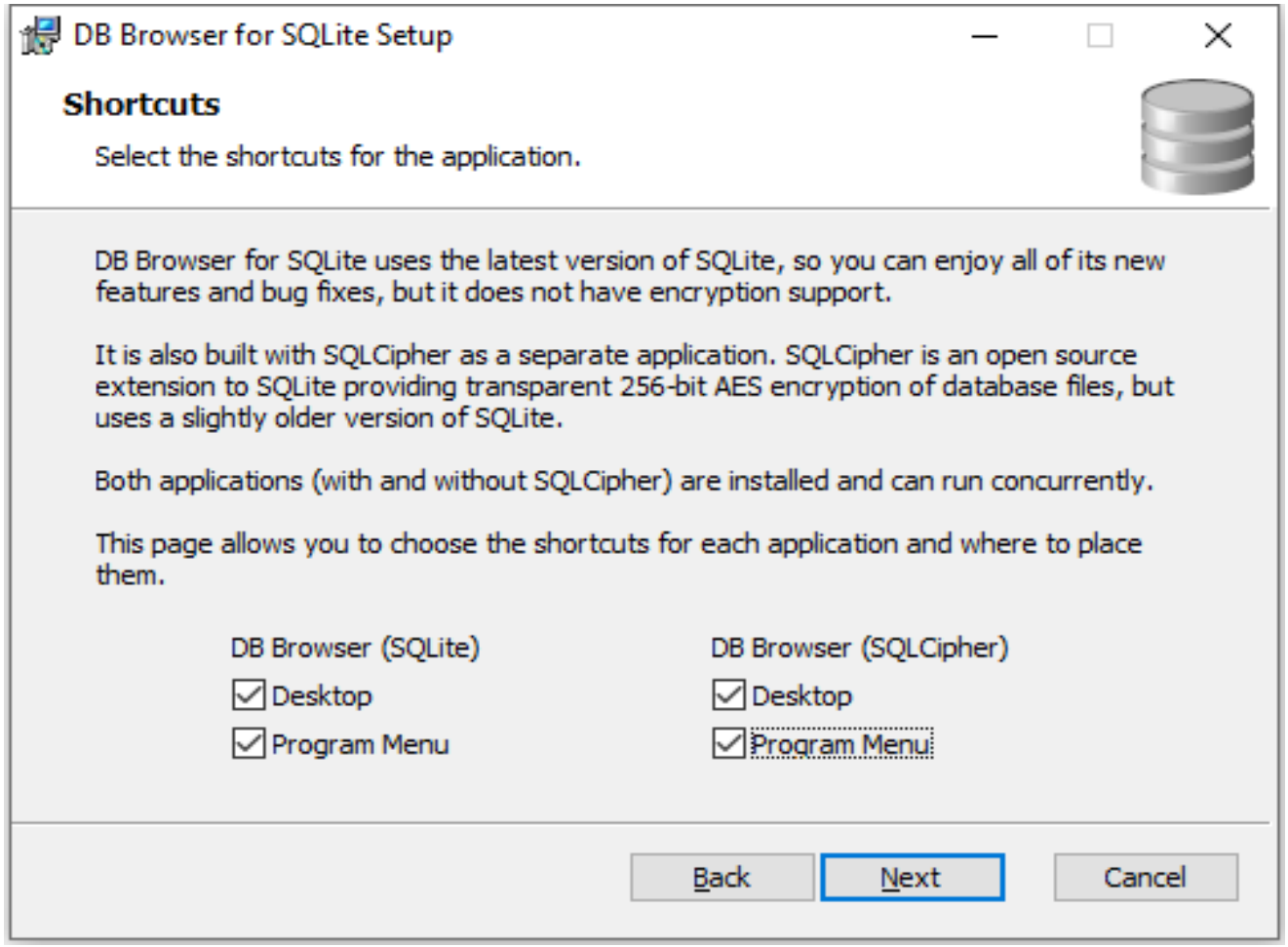
Hình 5.2. Màn hình cài đặt của DB Browser for SQLite.

Trên màn hình xuất hiện cửa sổ như hình 5.3. Chọn ‘I accept the term in the License Agreement’ → chọn Next.

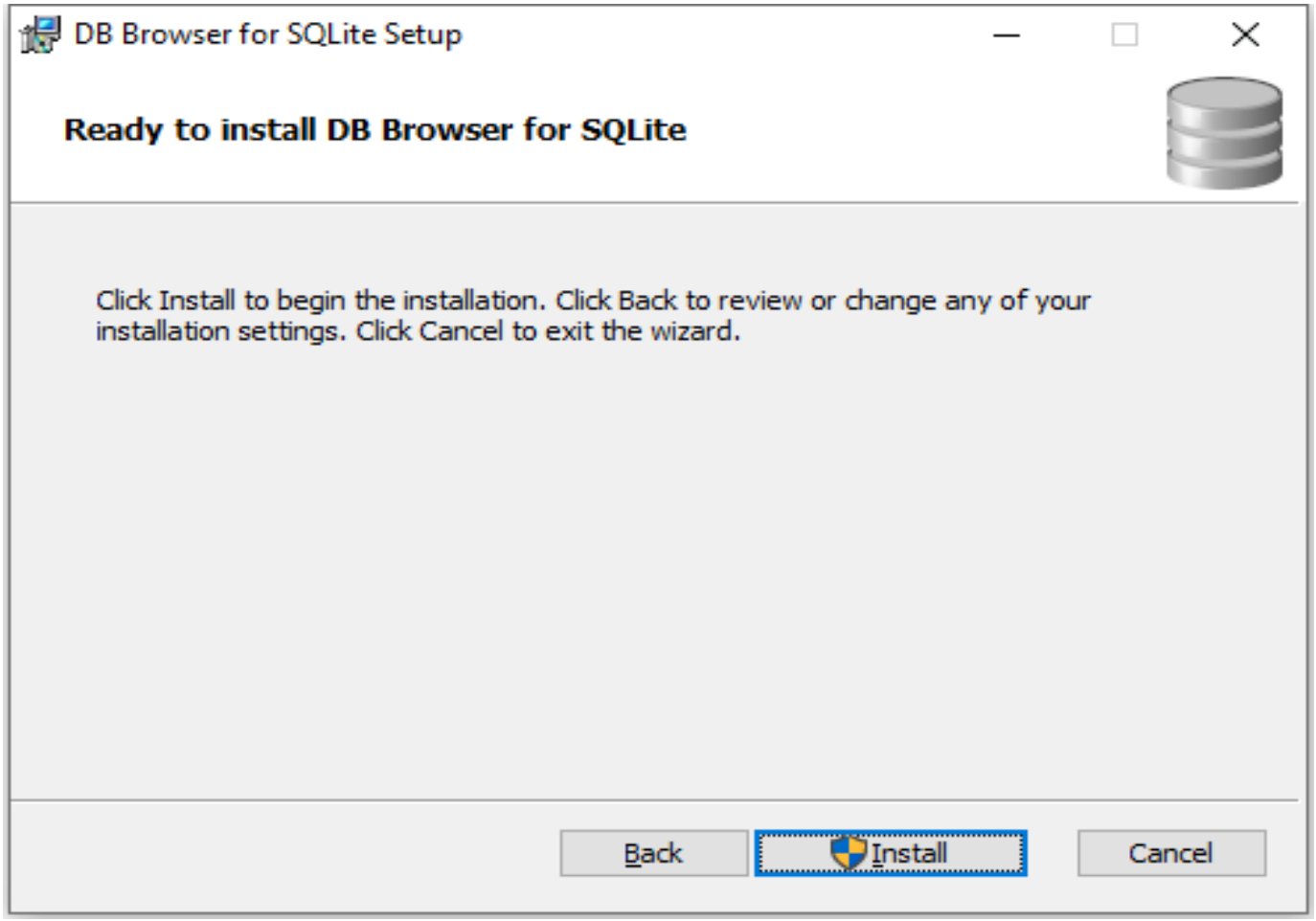


Hình 5.3. Chọn Next DB Browser for SQLite.

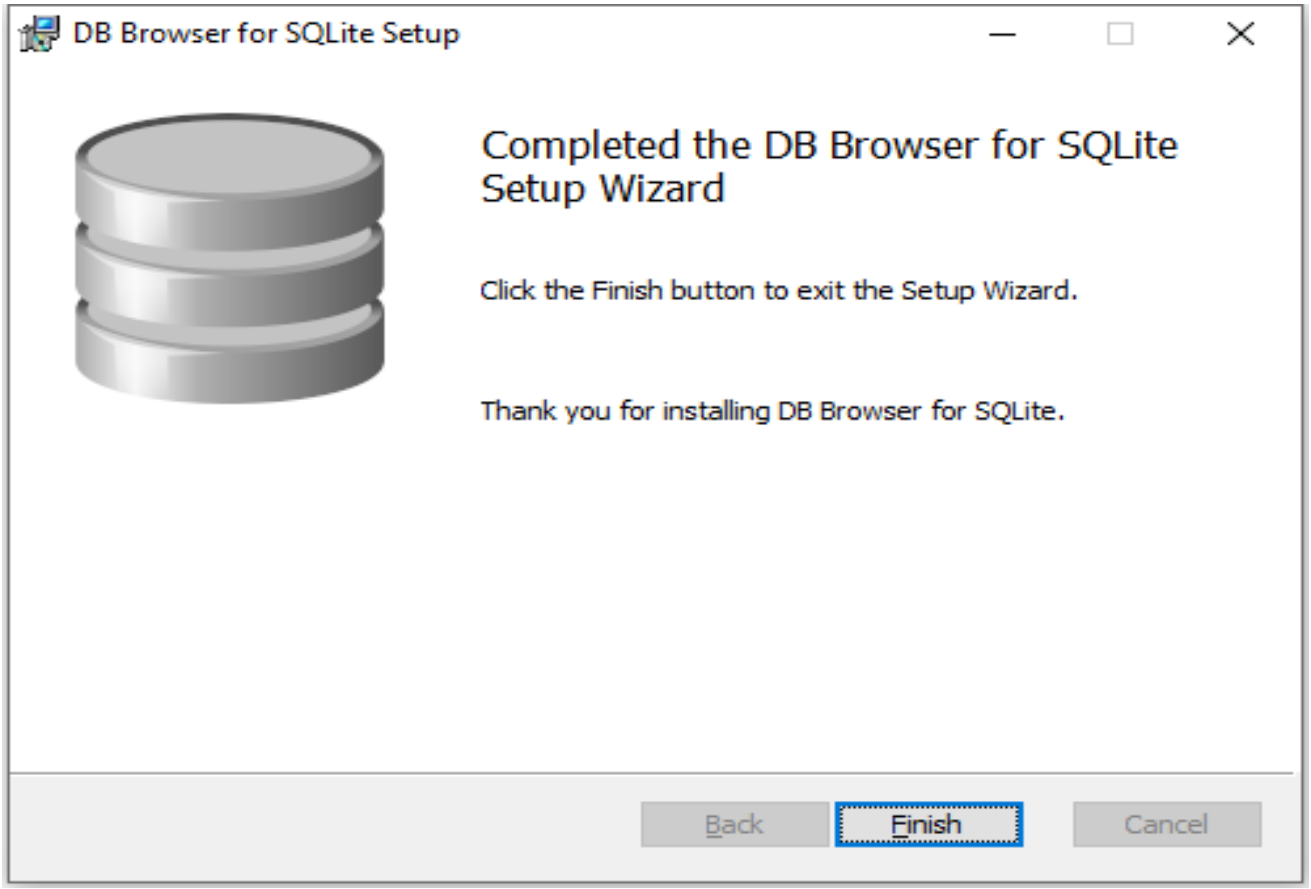
Tiến hành lựa chọn phím tắt (Shortcuts) cho ứng dụng. Ở đây có thể chọn như trên hình 5.4. sau đó bấm Next.



Hình 5.4. Chọn Next để tiếp tục.



Hình 5.5. Chọn Install để cài đặt.



Hình 5.6. Bấm finish để hoàn thành việc cài đặt.

Tùy theo việc lựa chọn trong khi cài đặt (nếu lựa chọn all) khi cài đặt xong có 3 icon trên Desktop là: 'DB Browser (**SQLCipher**), DB Browser (**SQLite**) và DB Browser (**SQLiteStudio**).

'DB Browser (SQLCipher),

DB Browser (SQLite)

DB Browser (SQLiteStudio).

- DB Browser **(SQLCipher)** là phiên bản được hỗ trợ SQLCipher.

  SQLCipher là một thư viện mã nguồn mở cho phép mã hóa cơ sở dữ liệu SQLite. Điều này có nghĩa là dữ liệu trong cơ sở dữ liệu có thể được bảo vệ khỏi truy cập trái phép.

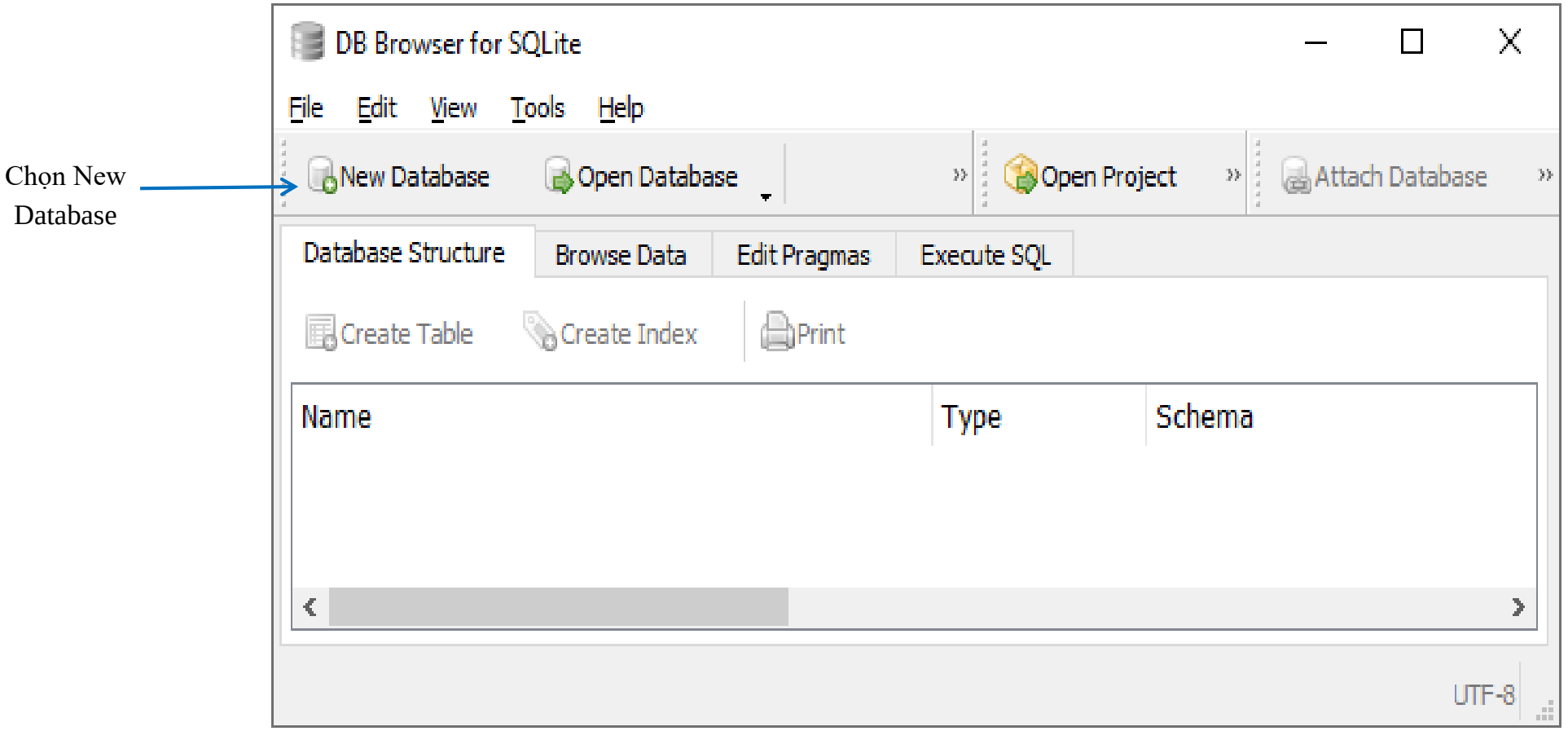
- DB Browser **(SQLite)** là phiên bản cơ bản của DB Browser. Nó không hỗ trợ SQLCipher, vì vậy dữ liệu trong cơ sở dữ liệu sẽ không được mã hóa.
- DB Browser (**SQLiteStudio**) là phiên bản nâng cao của DB Browser. Nó cung cấp một số tính năng bổ sung so với DB Browser (SQLite), bao gồm:
  - Trình soạn thảo SQL nâng cao
  - Trình quản lý bảng và cột nâng cao
  - Trình quản lý giao dịch nâng cao
  - Trình xem lịch sử nâng cao

Quyết định sử dụng phiên bản nào là phụ thuộc vào nhu cầu của người dùng.

5.2.3. Sử dụng DB Browser for SQLite

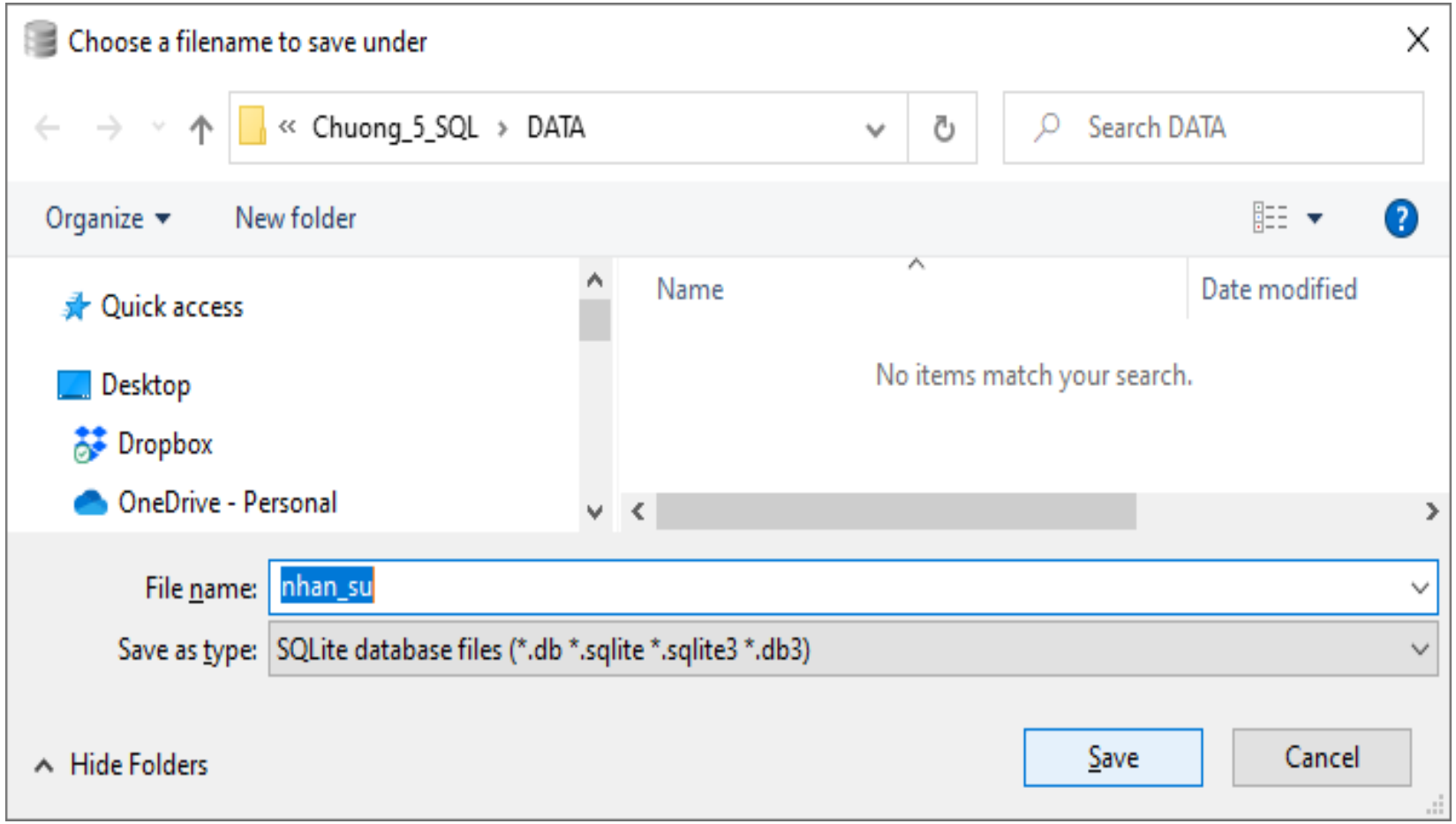
a. Tạo cơ sở dữ liệu Database

Mở DB Browser for SQLite, sau đó chọn **New Database** (xem hình 5.7.)



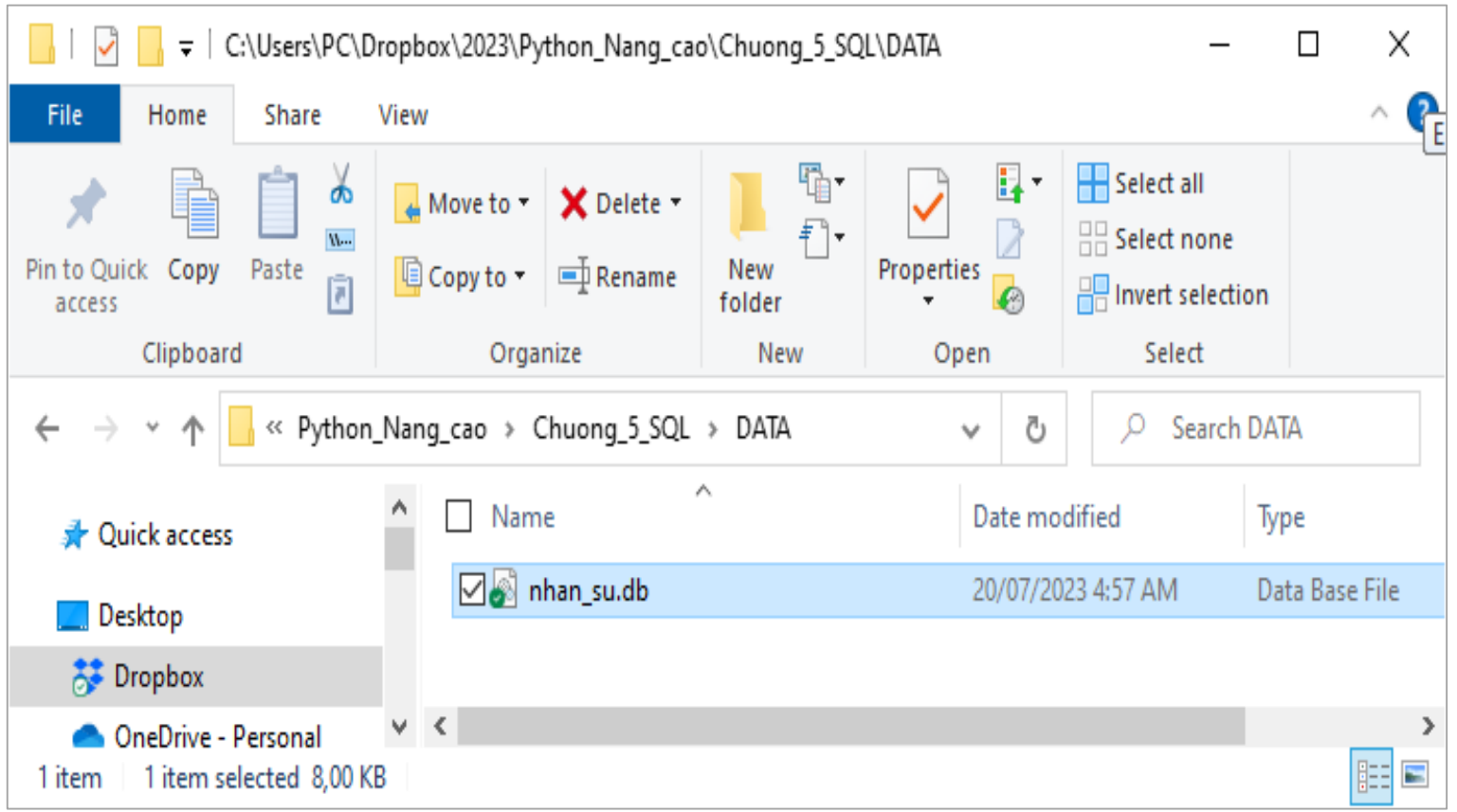
Hình 5.7. Tạo CSDL mới.

Ở hình 5.8, trong ô Filename gõ vào tên database là : nhan_su, sau đó chọn save. Một file tên là nhan_su.db sẽ được tạo ra và chúng ta có thể lưu vào địa chỉ theo đường dẫn chẳng hạn:…\DATA\nhan_su.db.



Hình 5.8. Lưu tên CSDL vào thư mục

Sau khi lựa chọn đường dẫn để lưu file nhan_su.db, tiếp tục xuất hiện màn hình Edit table database yêu cầu chúng ta tạo bảng cho cơ sở dữ liệu nhan_su.db. Lúc này, chúng ta có thể tiếp tục tạo bảng Table cho database. Tuy nhiên, tạm thời chọn **cancel** để đóng màn hình Edit table database lại. Lúc này file nhan_su.db đã được lưu lại xem hình 5.9. Edit table database

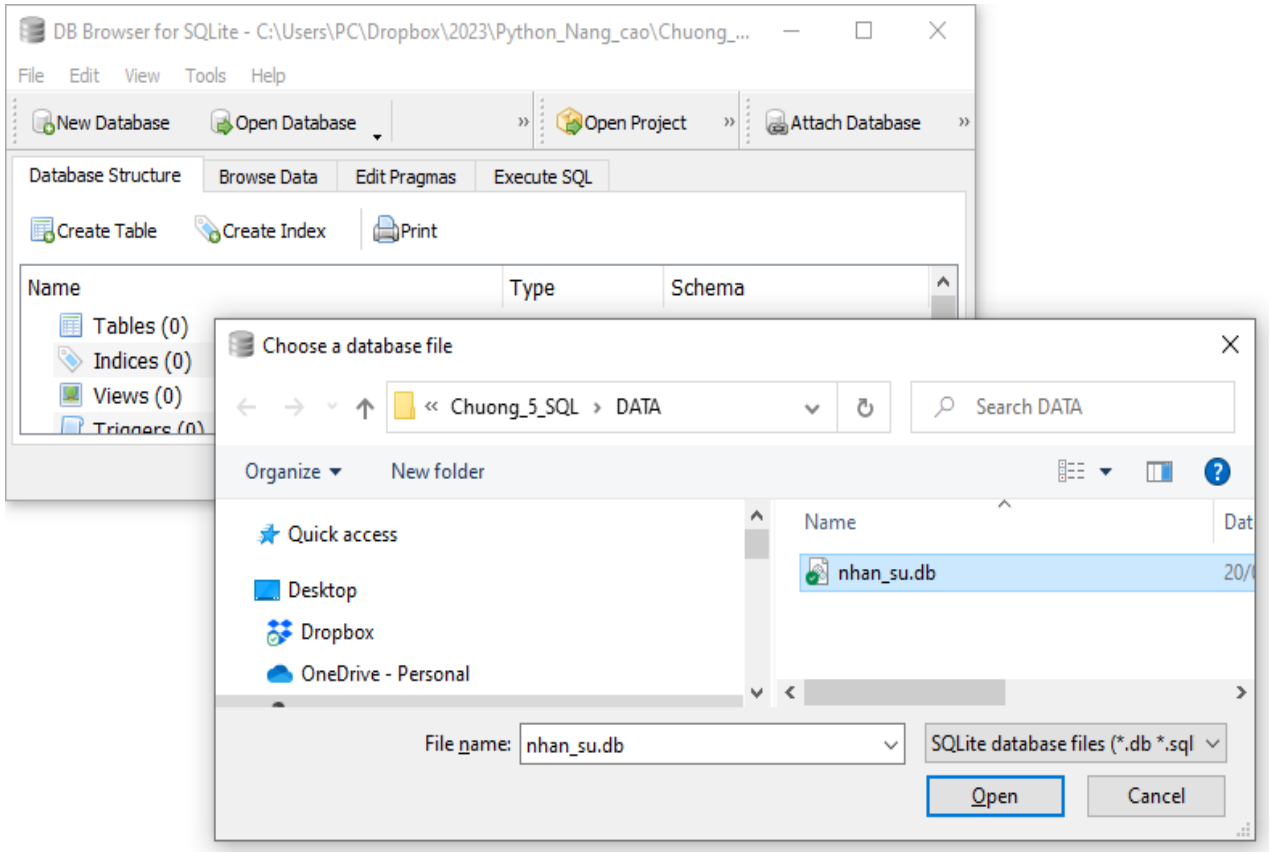


Hình 5.9. File nhan_su.db được lưu trong thư mục DATA

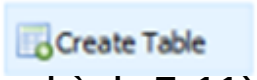
b. Cài đặt cơ sở dữ liệu với SQlite – Tạo bảng (Create Table)

Cách 1: Sử dụng giao diện (Làm việc trực tiếp với DB Browser for SQLite)

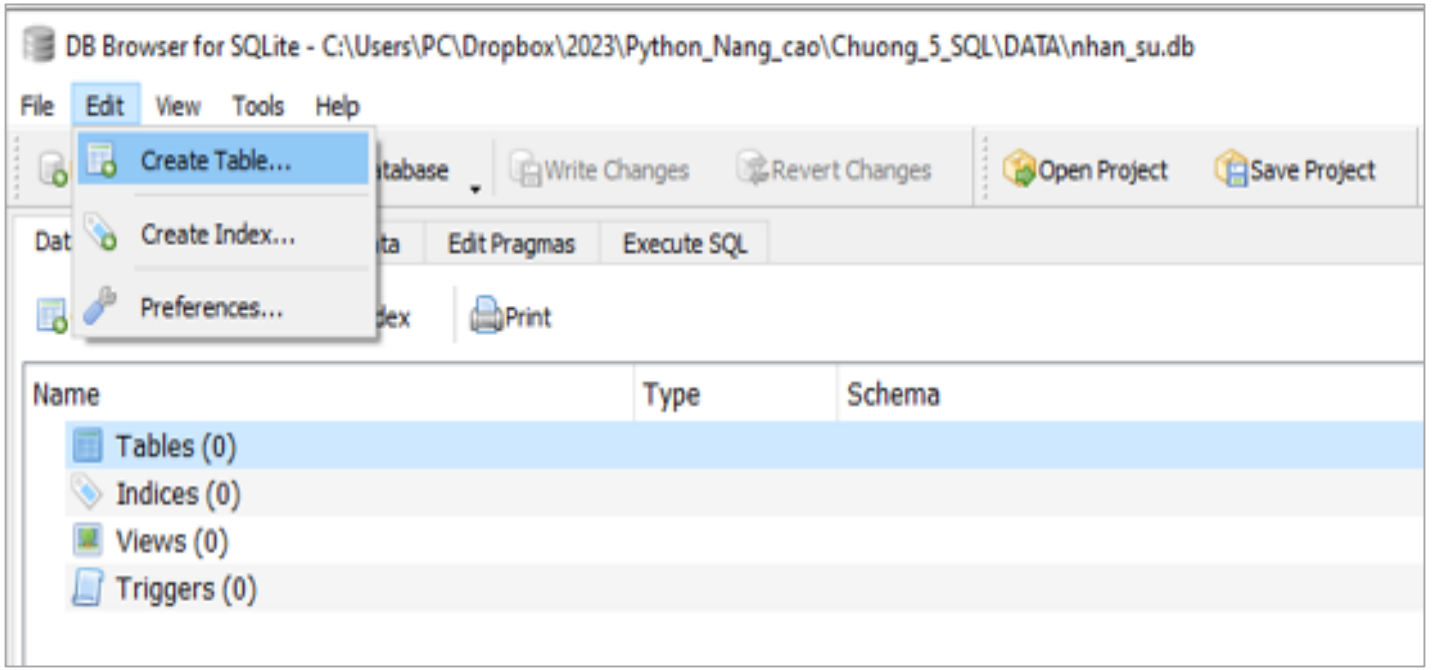
Bước 1. Chọn open database → xuất hiện màn hình (xem hình 5.10), chọn file nhan_su sau đó bấm open)



Hình 5.10. Tạo CSDL nhan_su.db.

Bước 2. chọn vào mục 

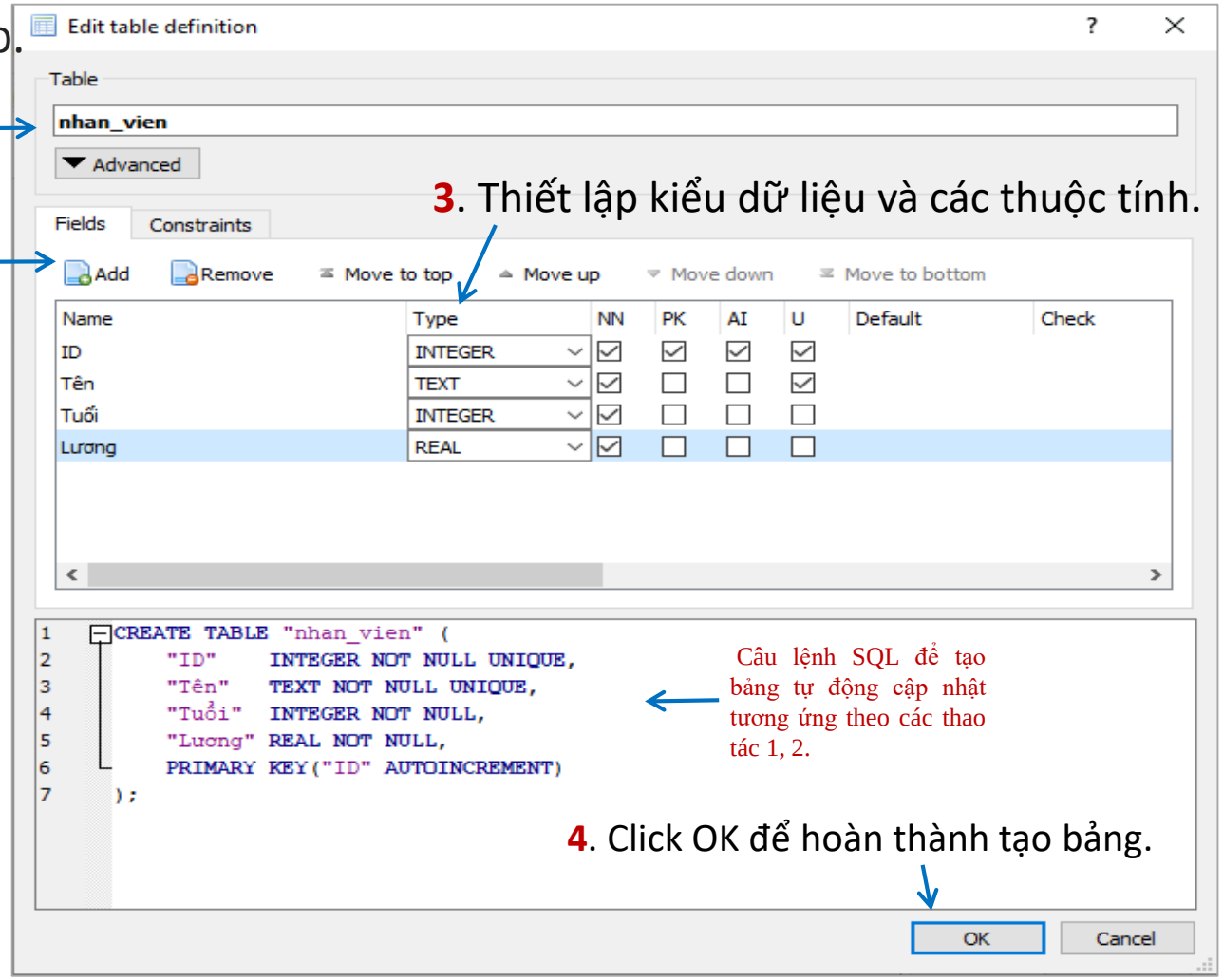
ở trên thanh công cụ tools bar, hoặc trên menu chọn Edit → Create Table (xem hình 5.11) → xuất hiện màn hình tạo bảng Edit table definition (Hình 5.12), lúc này chúng ta đặt tên table là nhan_vien,



Hình 5.11. Tạo bảng

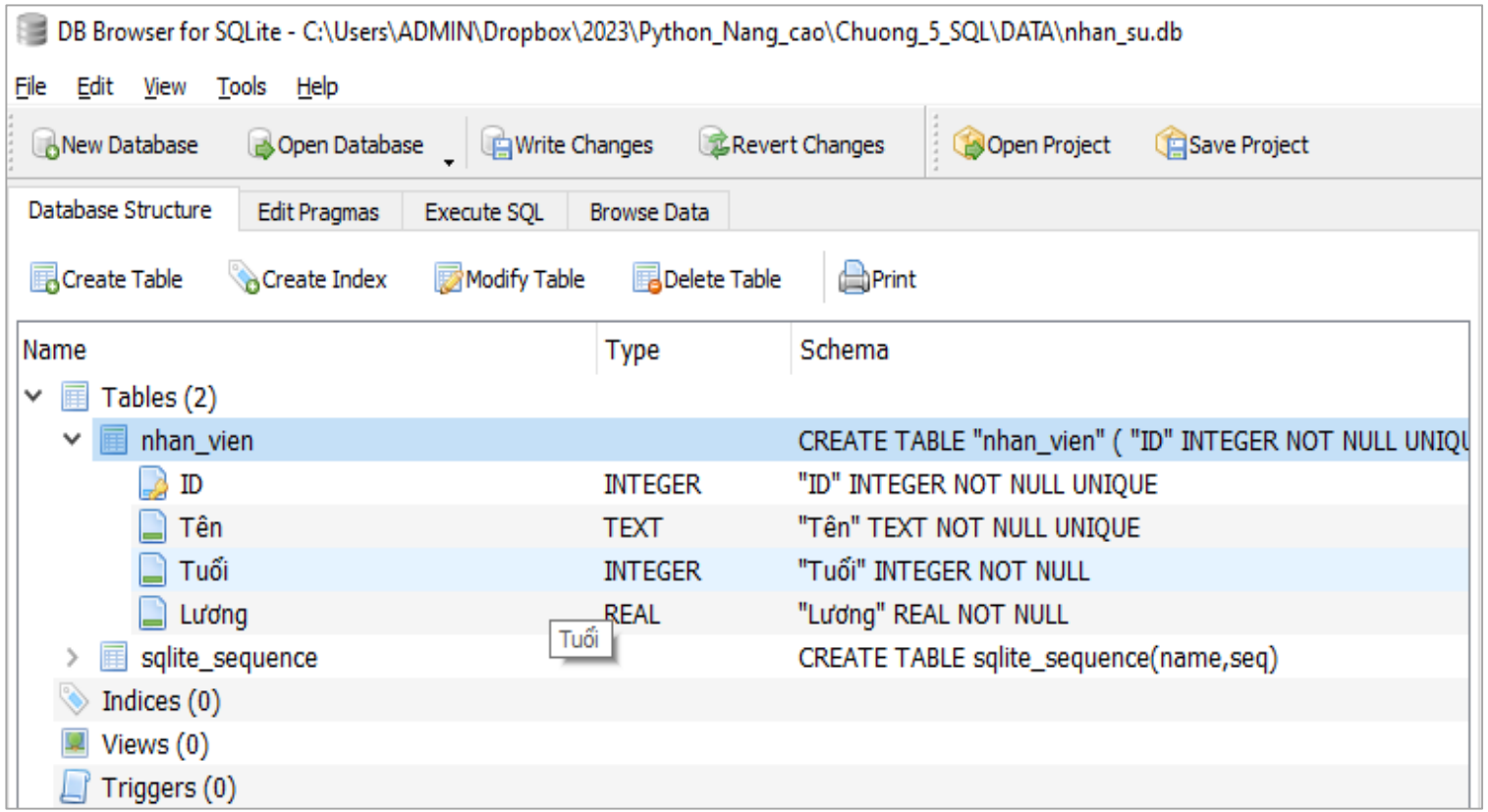
Trong màn hình (hình 5.12), Chọn **Add ->** nhập tên cột, chỉ định kiểu dữ liệu, nhập kích thước, thiết lập ràng buộc null -> chọn OK (Thực hiện thao tác này nhiều lần để tạo nhiều cột cho bảng).

Type (Kiểu dữ liệu) của trường: INTEGER, TEXT, BLOG, NUMERIC.
NN (Not Null): Không được để trống.
PK (Primary Key): khóa chính.
AI (Auto Increcement): Tự động tăng giá trị.
U(Unique): Giá trị không được trùng lặp.



Hình 5.12. Tạo bảng nhan_vien

Kết quả bảng table **nhan_vien** được tạo ra như trong hình 5.13.



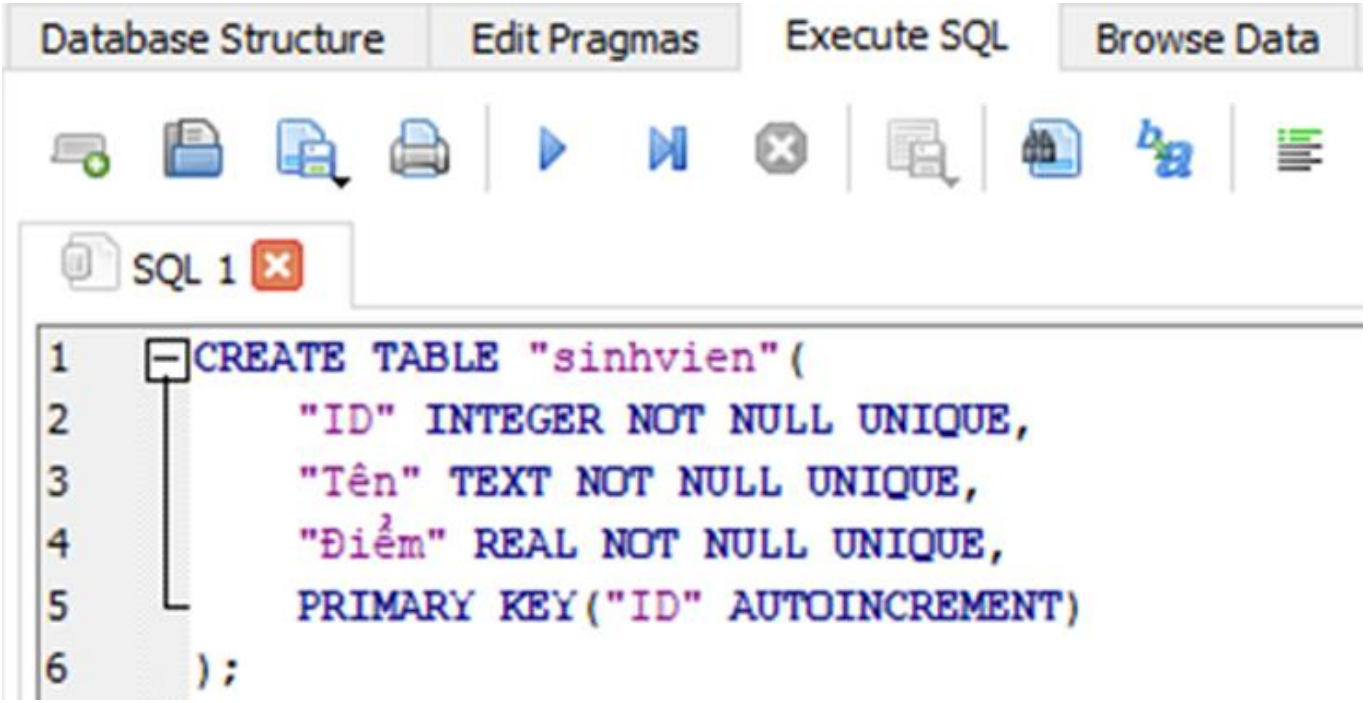
Hình 5.13. Bảng nhan_vien đã được tạo.

Cách 2. Sử dụng câu lệnh (Thường được sử dụng khi viết code)

Cú pháp:

```
CREATE TABLE ten_bang (
    cot1_kieu_du_lieu,
    cot2_kieu_du_lieu,
    ...
);
```

Ví dụ: tạo CSDL quản lý điểm học phần (diem_hoc_phan), sau đó tạo bảng sinh_vien bằng cách sử dụng câu lệnh SQL trong DB Browser SQLite như sau. (xem hình 5.14)



Hình 5.14. Tạo bảng với câu lệnh SQL trong DB Browser SQLite.

Bước 1. Tạo/mở file database diem_hoc_phan.db.

Bước 2. Trên menu chọn **Execute SQL**, sau đó trong cửa sổ SQL gõ vào nội dung:

```
1  CREATE TABLE "sinhvien"(
2     "ID" INTEGER NOT NULL UNIQUE,
3     "Tên" TEXT NOT NULL UNIQUE,
4     "Điểm" REAL NOT NULL UNIQUE,
5     PRIMARY KEY("ID" AUTOINCREMENT)
6  );
```

Bước 3. Kiểm ta lại cú pháp (ngữ pháp) và bấm vào biểu tượng  để thực hiện câu lệnh SQL, nếu không có lỗi trên màn hình sẽ hiện thông báo sau:

```
Execution finished without errors.
Result: query executed successfully. Took 0ms
At line 1:
CREATE TABLE "sinhvien"(
            "ID" INTEGER NOT NULL UNIQUE,
            "Tên" TEXT NOT NULL UNIQUE,
            "Điểm" REAL NOT NULL UNIQUE,
            PRIMARY KEY("ID" AUTOINCREMENT)
);
```

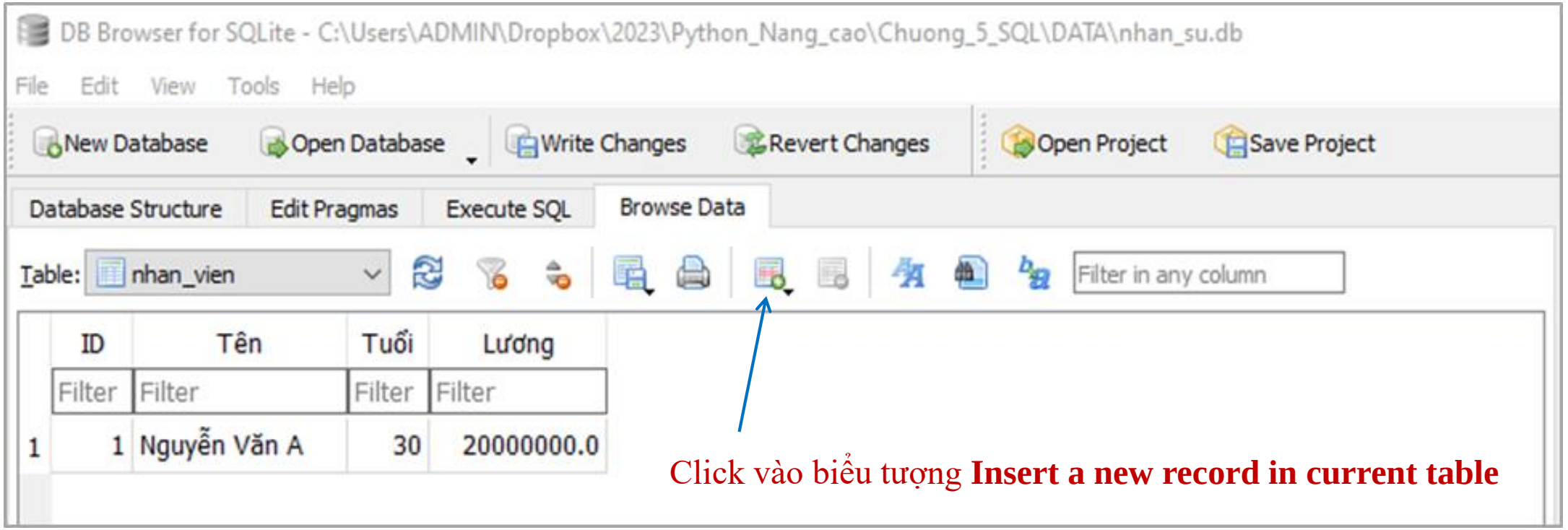
c. Thêm dữ liệu vào bảng:

Cách 1: Sử dụng giao diện (Làm việc trực tiếp với DB Browser for SQLite)

Ví dụ: Thêm dữ liệu vào bảng nhan_vien trong CSDL nhansu.db.

Bước 1. Chọn tab Browse Data, sau đó chọn biểu tượng  (Insert a new record in current table). (Xem hình 5.15)

Bước 2. Nhập các giá trị "ID" =1, "Tên" = 'Nguyễn Văn A', "Tuổi" =30, "Lương" = 20000000.0 vào bảng

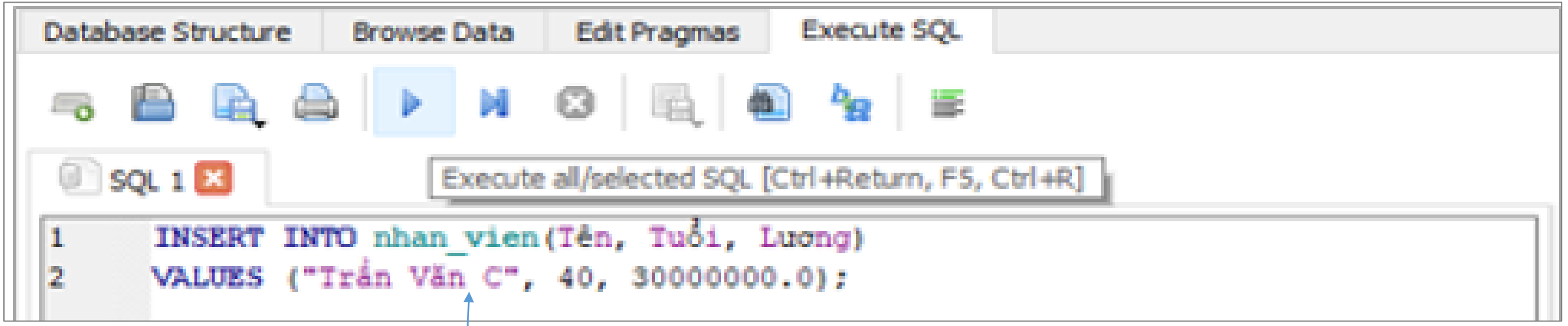


Cách 2. Sử dụng câu lệnh INSERT INTO để thêm dữ liệu mới vào bảng.

Cú pháp:

```
INSERT INTO ten_bang (cot1, cot2, ...)
VALUES (gia_tri_cot1, gia_tri_cot2, ...);
```

Ví dụ: Thêm nhân viên với ID = 2, Tên = 'Trần Văn B', tuổi = 40 và lương = 30000000.0 vào bảng nhan_vien.

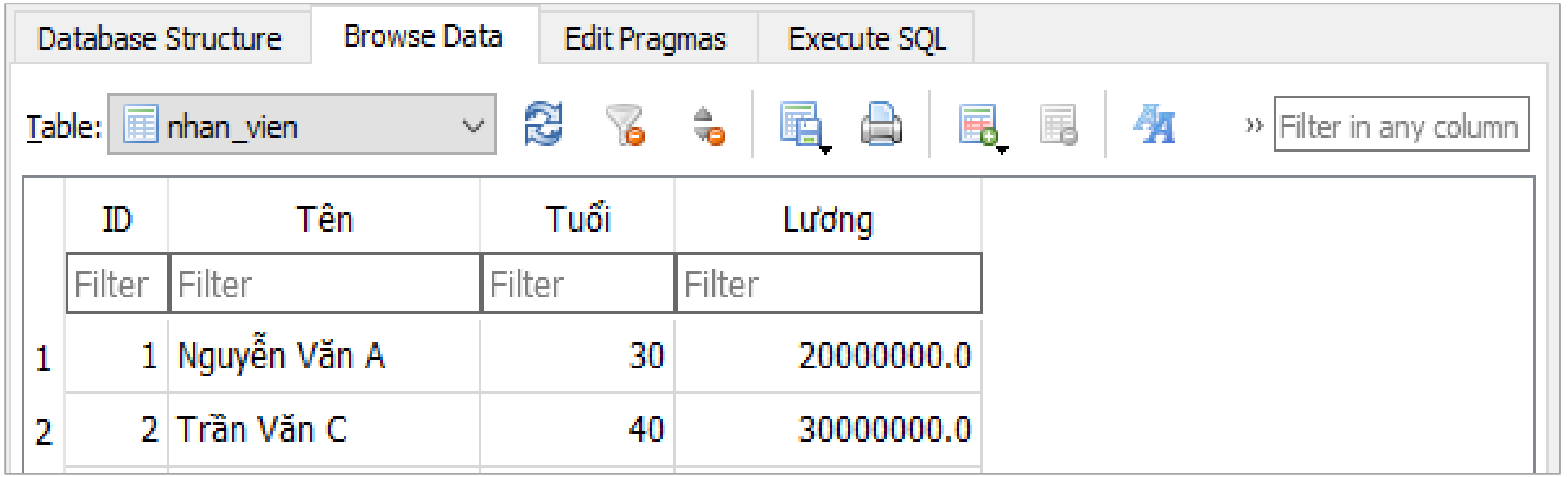


Bước 1. Chọn tab Excute SQL.

Bước 2: Gõ vào câu lệnh như trên hình

bấm F5 hoặc click vào biểu tượng  hoặc gõ tổ hợp phím Ctr+Return/Ctr+R để thực hiện lệnh SQL.

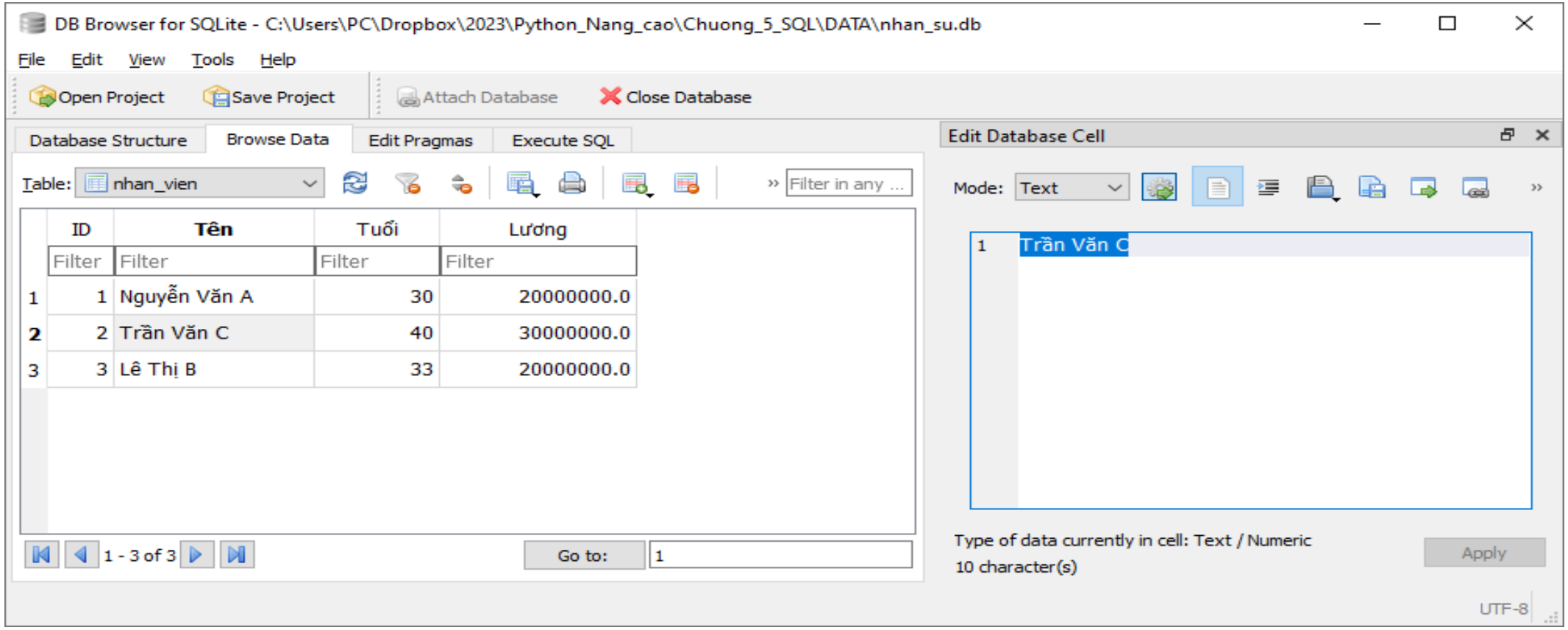
Sau đó chúng ta có thể chuyển qua tab Browse Data để xem kết quả:



Lưu ý: Do trường ID dặt thuộc tính là Auto Increcement nên trong câu lệnh SQL chúng ta không cần nhập giá trị cho trường ID.

d. Cập nhật/chỉnh sửa dữ liệu.

Cách 1: Sử dụng giao diện (Làm việc trực tiếp với DB Browser for SQLite)

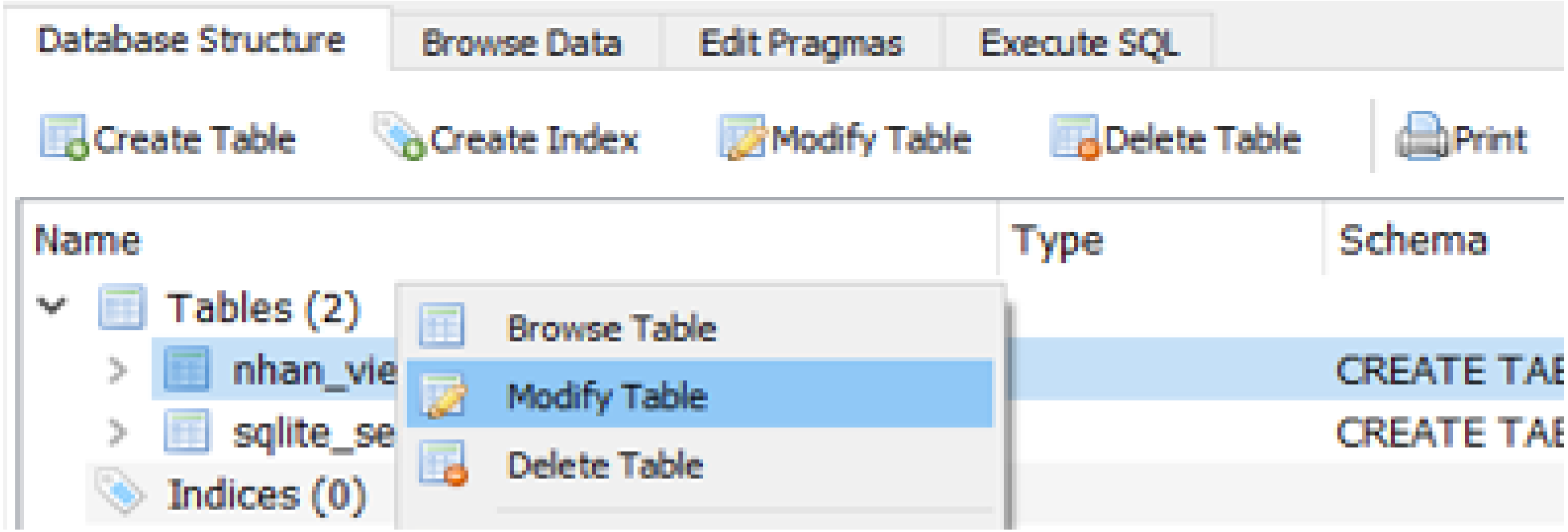


Hình 5.18. Cập nhật dữ liệu

Nhấp đúp trực tiếp vào ô cần cập nhật trong bảng và thực hiện cập nhật trong mục Edit Database Cell. (Hình 5.18).

Cập nhật/chỉnh sửa bảng

Để chỉnh sửa bảng, chúng ta mở tab Database Structure → chọn bảng cần chỉnh sửa → right click→ chọn Modify Table (hình 5.19) để mở màn hình Edit table definition sau dó tiến hành chỉnh sửa như khi tạo bảng.



Hình 5.19. Chỉnh sửa bảng

Lưu ý: Trước khi chỉnh sửa bảng DB Browser SQLite sẽ nhắc chúng ta lưu lại dữ liệu của bảng.

e. Xóa dữ liệu.

Xóa dữ liệu trong bảng:

Mở tab Browser Data:

Xóa từng dòng: Đặt con trỏ vào dòng cần xóa, sau đó chọn nút xóa dòng  , khi đưa chuột đến nút xóa dòng sẽ xuất hiện thông báo “Delete the curent recod” ngay dưới nút, và ở phía dưới màn hình Browser Data cũng xuất hiện dòng cảnh báo với nội dung: “This button deletes the record or records curently selected in the table”.

Trường hợp muốn xóa nhiều dòng chúng ta có thể chọn nhiều dòng và sau đó thực hiện thao tác xóa như trên.

g. Xóa bảng:

Mở tab Database Structure, sau đó chọn bảng cần xóa, Right click → chọn Delete Table.(xem hình 5.19)

5.3. THAO TÁC VỚI CSDL SQLITE VỚI PYTHON.

5.3.1. Kết nối với SQL

a. import mô đun SQLite

Cú pháp: import **sqlite3**

b. Tạo kết nối - Create Connection

Sử dụng phương thức **connect()**, được cung cấp bởi thư viện SQLite3 trong Python.

Ví dụ:

```
conn = sqlite3.connect('mydatabase.db')
```

- Nếu file cơ sở dữ liệu 'mydatabase.db' đã tồn tại, phương thức connect() sẽ kết nối tới cơ sở dữ liệu đó.
- Nếu không, SQLite sẽ tạo một file mới với tên 'mydatabase.db' để sử dụng làm cơ sở dữ liệu.

conn: Là một connection **object** (đối tượng kết nối), **conn** đại diện cho kết nối giữa chương trình Python và cơ sở dữ liệu SQLite. Từ đối tượng này, chúng ta có thể thực thi các lệnh SQL hoặc quản lý CSDL.

Lưu ý: *cẩn thận khi sử dụng tùy chọn tạo mới và đảm bảo rằng dữ liệu quan trọng đã được sao lưu trước khi thực hiện.*

c. Thực thi câu lệnh SQL-excute() : cursor = conn.**cursor()**

- **cursor**: Là một cursor object (**đối tượng con trỏ**).
  - Con trỏ này được sử dụng để thực hiện các thao tác trên cơ sở dữ liệu, như thực thi các câu lệnh SQL (**SELECT, INSERT, UPDATE, DELETE**, v.v.).
  - Con trỏ cũng giúp lấy kết quả trả về từ các câu truy vấn (như truy vấn SELECT).

Sử dụng đối tượng **cursor object** để gọi đến phương thức **execute()** và truyền vào tên của câu lệnh sql làm tham số cho nó.

Ví dụ 5.3.1.1.

```
cursor.execute('''CREATE TABLE IF NOT EXISTS users(id INTEGER PRIMARY KEY,
                  name TEXT, age INTEGER) ''')
cursor.execute("INSERT INTO users (name, age) VALUES('Nguyễn V A', 30)")
```

Lưu ý: *Để viết câu lệnh SQL trên nhiều dòng trong Python, chúng ta có thể sử dụng cặp dấu nháy đơn ba lần nháy đơn ''' hoặc ba lần nháy kép """. Điều này cho phép viết câu lệnh SQL dưới dạng chuỗi trên nhiều dòng một cách dễ đọc hơn.*

5.3.1. Kết nối với SQL, …

Đối tượng kết nối (**connection object**):

Quản lý kết nối, giao dịch,

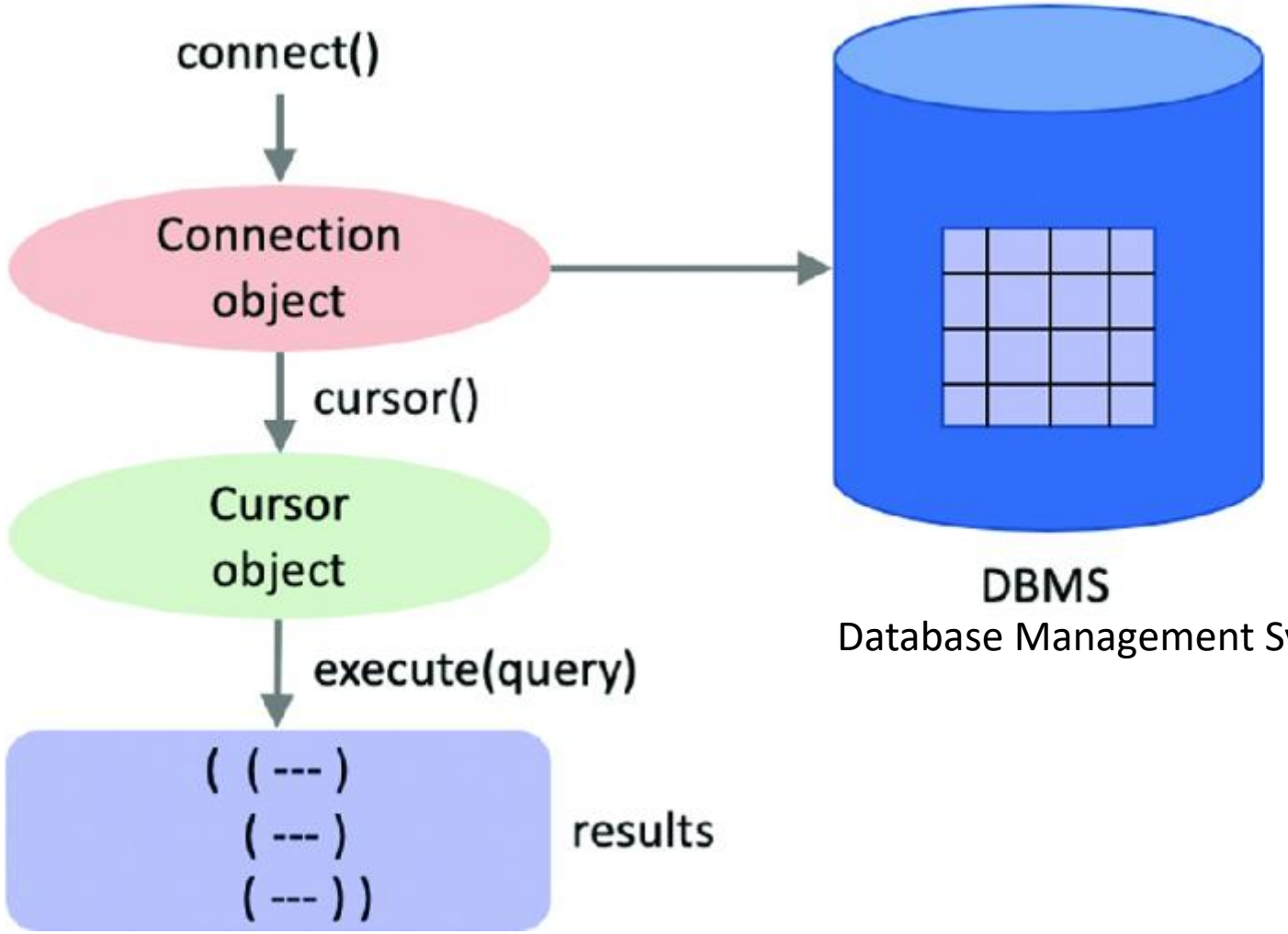
và trạng thái cơ sở dữ liệu.

Đối tượng con trỏ (**cursor object**):

Thực thi các lệnh SQL,

quản lý kết quả truy vấn, và

cung cấp phương thức xử lý dữ liệu..



Minh họa vai trò của đối tượng kết nối và đối tượng con trỏ

5.3.1. Kết nối với SQL, …

d. Lưu thay đổi - commit(): phương thức commit() được sử dụng để lưu các thay đổi đã thực hiện vào cơ sở dữ liệu. Khi người dùng thực hiện các câu lệnh INSERT, UPDATE hoặc DELETE, các thay đổi sẽ không được lưu trực tiếp vào cơ sở dữ liệu mà chỉ ở trạng thái tạm thời.

Để đảm bảo rằng các thay đổi được lưu lại và cơ sở dữ liệu được cập nhật, chúng ta cần gọi phương thức **commit()** trên đối tượng kết nối.

Cú pháp:

```
sqliteConnection.commit()
```

e. Đóng kết nối - close(): sử dụng phương thức close() để đóng kết nối đến cơ sở dữ liệu.

Ví dụ 5.3.1.2.

```
1  import sqlite3
2
3  # Kết nối đến cơ sở dữ liệu (hoặc tạo cơ sở dữ liệu mới nếu nó không tồn tại)
4  conn = sqlite3.connect('mydatabase.db')
5
6  # Tạo một đối tượng con trỏ (Cursor object) từ đối tượng kết nối
7  cursor = conn.cursor()
8
9  # Thực thi câu lệnh SQL và truy xuất dữ liệu được đặt trong cặp dấu '''...'''
10 cursor.execute('''
11                  CREATE TABLE IF NOT EXISTS users (
12                  id INTEGER PRIMARY KEY,
13                  "Tên" TEXT,
14                  "Tuổi" INTEGER
15                  )
16                  ''')
17 cursor.execute('''
18                  INSERT INTO users ("Tên", "Tuổi")
19                  VALUES ("Nguyễn Văn A", 30)
20                  ''')
21
22 # Lưu các thay đổi vào cơ sở dữ liệu (nếu cần)
23 conn.commit()
24
25 # Truy xuất dữ liệu từ bảng users
26 cursor.execute("SELECT * FROM users")
27 rows = cursor.fetchall()
28
29 # In kết quả
30 for row in rows:
31     print(row)
32
33 # Đóng con trỏ (Cursor object) và đóng kết nối
34 cursor.close()
35 conn.close()
```

Kết quả thực hiện chương trình:

```
(1, 'Nguyễn Văn A', 30)
```

5.3.2 Cập nhật cơ sở dữ liệu SQLite

Cú pháp:

```
UPDATE ten_bang
SET cot1 = gia_tri_moi1, cot2 =
gia_tri_moi2, ...
WHERE dieu_kien;
```

Trong đó:

- ten_bang: Tên của bảng muốn cập nhật bản ghi.
- cot1, cot2, ...: Các cột muốn cập nhật giá trị.
- gia_tri_moi1, gia_tri_moi2, ...: Giá trị mới muốn gán cho các cột tương ứng.
- dieu_kien: Điều kiện xác định các bản ghi mà chúng ta muốn cập nhật. Nếu không có điều kiện, câu lệnh UPDATE sẽ cập nhật tất cả các bản ghi trong bảng.

Ví dụ:

```
UPDATE nhan_vien
SET lương = 25000000.0
WHERE tên = 'Nguyễn Văn A';
```

Sau khi thực thi câu lệnh này, bản ghi của nhân viên có tên là "Nguyễn Văn A" trong bảng "nhan_vien" sẽ có mức lương mới là 25000000.0.

Ví dụ 5.3.2.1.

```
1  # code for update operation
2  import sqlite3
3
4  # Kết nối đến cơ sở dữ liệu, tên cơ sở dữ lệu được truyền như một tham số của connect()
5  conn = sqlite3.connect('mydatabase.db')
6
7  # cập nhật bản ghi nhân viên từ bảng users
8  conn.execute("UPDATE users SET 'Tên' = 'Nguyễn Vân Anh' where id='1'")
9  conn.commit()
10
11 print("Tổng số dòng được cập nhật :", conn.total_changes)
12
13 cursor = conn.execute("SELECT * FROM users")
14 for row in cursor:
15    print(row)
16 # Đóng kết nối sau khi hoàn tất công việc
17 conn.close()
```

Kết quả thực hiện chương trình:

```
(1, 'Nguyễn Vân Anh', 30)
```

b. Xóa bản ghi trong cơ sở dữ liệu SQLite

Cú pháp:
```
DELETE FROM ten_bang
WHERE dieu_kien;
```

Trong đó:

- ten_bang: Tên của bảng mà người dùng muốn xóa bản ghi.
- dieu_kien: Điều kiện xác định bản ghi muốn xóa. Nếu không có điều kiện, câu lệnh DELETE sẽ xóa tất cả các bản ghi trong bảng.

Ví dụ 5.3.2.2: Xóa một bản ghi cụ thể trong bảng "users" dựa trên điều kiện nào đó (chẳng hạn: id = 1):

```
DELETE FROM users
WHERE id = 1;
```

Lưu ý: việc xóa bản ghi bằng câu lệnh DELETE là không thể đảo ngược, nghĩa là sau khi bản ghi đã bị xóa, chúng ta sẽ không thể khôi phục lại dữ liệu đó. Vì vậy, hãy chắc chắn rằng đã xác định rõ điều kiện xóa và đảm bảo là không bỏ sót bất kỳ bản ghi nào quan trọng trong cơ sở dữ liệu trước khi thực hiện câu lệnh DELETE.

Ví dụ 5.3.2.3.

```
1  # code thực hiện xóa bản ghi
2  import sqlite3
3
4  # Kết nối đến cơ sở dữ liệu,
5  conn = sqlite3.connect('mydatabase.db')
6
7  # xóa bản ghi của bảng users từ database mydatabase.db
8  conn.execute("DELETE from users where id=1")
9  conn.commit()
10 print("Tổng số bản ghi được xóa :", conn.total_changes )
11
12 cursor = conn.execute("SELECT * FROM users")
13 for row in cursor:
14    print(row)
15
16 # Đóng kết nối sau khi hoàn tất công việc
17
18 conn.close()
```

Kết quả thực hiện chương trình:

```
Tổng số bản ghi được xóa : 1
(2, 'Trần Văn C', 45)
(3, 'Lê Thị B', 45)
```

c. Thêm (insert) dữ liệu do người dùng nhập vào csdl SQLite

Cú pháp:

```
INSERT INTO ten_bang (cot1, cot2, ...)
VALUES (gia_tri1, gia_tri2, ...);
```

- ten_bang: Tên của bảng muốn chèn dữ liệu.
- cot1, cot2, ...: Tên các cột trong bảng muốn chèn dữ liệu.
- gia_tri1, gia_tri2, ...: Các giá trị tương ứng của các cột mà chúng ta muốn chèn vào bảng.

TH 1: Chèn một bản ghi mới vào bảng "users" với các giá trị cụ thể:

```
INSERT INTO users ("Tên", "Tuổi")
VALUES ('Nguyễn Văn K', 35);
```

TH2: Chèn nhiều bản ghi mới cùng một lúc bằng câu lệnh INSERT với tham số định danh:

```
INSERT INTO users ("Tên", "Tuổi")
VALUES ('Trần Văn M', 25),
       ('Nguyễn Thị B', 35),
       ('Lê Văn N', 28);
```

TH3: sử dụng cú pháp "INSERT INTO ... SELECT" để chèn dữ liệu từ một bảng khác hoặc từ một câu lệnh SELECT.

```
INSERT INTO users ("Tên", "Tuổi")
SELECT "Tên", "Tuổi"
FROM tap_su
WHERE "Tuổi" > 25;
```

Lưu ý: câu lệnh INSERT sẽ thêm dữ liệu mới vào bảng, nếu bảng không tồn tại thì nó sẽ tạo mới bảng. Tuy nhiên, cần đảm bảo rằng tên cột và giá trị được cung cấp phù hợp với cấu trúc của bảng để tránh lỗi và đảm bảo tính chính xác của dữ liệu trong cơ sở dữ liệu.

Ví dụ 5.3.2.4: về cách thêm dữ liệu do người dùng nhập vào cơ sở dữ liệu SQLite:

```
1  import sqlite3
2
3  # Kết nối đến cơ sở dữ liệu (hoặc tạo cơ sở dữ liệu mới nếu nó không tồn tại)
4  conn = sqlite3.connect('mydatabase.db')
5
6  # Tạo một đối tượng con trỏ (Cursor object) từ đối tượng kết nối
7  cursor = conn.cursor()
8
9  # Nhập dữ liệu từ người dùng
10 name = input("Nhập tên người dùng: ")
11 age = int(input("Nhập tuổi người dùng: "))
12
13 # Thực hiện câu lệnh INSERT để chèn dữ liệu từ người dùng vào cơ sở dữ liệu
14 cursor.execute('INSERT INTO users ("Tên", "Tuổi") VALUES (?, ?)',
15 (name, age))
16
17 # Lưu các thay đổi vào cơ sở dữ liệu
18 conn.commit()
19
20 #In ra các dòng trong bảng sau khi cập nhật
21 cursor = conn.execute("SELECT * FROM users")
22 for row in cursor:
23     print(row)
24
25 # Đóng kết nối
26 conn.close()
```

'Đối tượng giữ chỗ (**placeholder**)', được sử dụng để cung cấp dữ liệu cho bản ghi mới.

Kết quả thực hiện chương trình:	
	Nhập tên người dùng: Nguyễn Văn C Nhập tuổi người dùng: 30 (2, 'Trần Văn C', 45) (3, 'Lê Thị B', 45) (4, 'Nguyễn Văn Thêm', 50) (5, 'Nguyễn Văn C', 30)

5.3.3.Vẽ biểu đồ trong SQLite

Một số dạng biểu đồ trong thư viện Matplotlib là: biểu đồ cột (bar chart), biểu đồ đường(Line Chart), biểu đồ hình bánh (Pie Chart), …

a. Biểu đồ cột (bar chart).

Ví dụ 5.3.3.1

```
import sqlite3
import matplotlib.pyplot as plt

# Kết nối đến cơ sở dữ liệu
conn = sqlite3.connect('mydatabase.db')
cursor = conn.cursor()

# Truy vấn dữ liệu từ cơ sở dữ liệu
cursor.execute("SELECT 'Tên', 'Tuổi' FROM users")
rows = cursor.fetchall()

# Tách dữ liệu thành hai danh sách riêng biệt cho tên và tuổi
Tên = [row[0] for row in rows]
Tuổi = [row[1] for row in rows]

#In ra các dòng trong bảng sau khi cập nhật
cursor = conn.execute("SELECT * FROM users")
for row in cursor:
   print(row)
# Đóng kết nối
conn.close()

# Vẽ biểu đồ cột
plt.bar(Tên, Tuổi)
plt.xlabel('Tên')
plt.ylabel('Tuổi')
plt.title('Biểu đồ tuổi của người dùng')
plt.show()
```

Lấy kết quả truy vấn

rows = cursor.**fetchall()**

- Phương thức **fetchall()** lấy tất cả kết quả từ câu lệnh SQL vừa thực hiện và trả về dưới dạng một danh sách (list) các bản ghi.
- Mỗi bản ghi là một tuple, trong đó mỗi phần tử trong tuple tương ứng với một cột của bản ghi đó.

Ví dụ: Nếu bảng users có dữ liệu sau:

Tên	Tuổi
Nguyễn Văn A	28
Trần Văn C	25
Lê Thị B	30

Biến **rows**

```
rows = [
    ('Nguyễn Văn A', 28),
    ('Trần Văn C', 30),
    ('Lê Thị B', 35)
]
```

list comprehension.

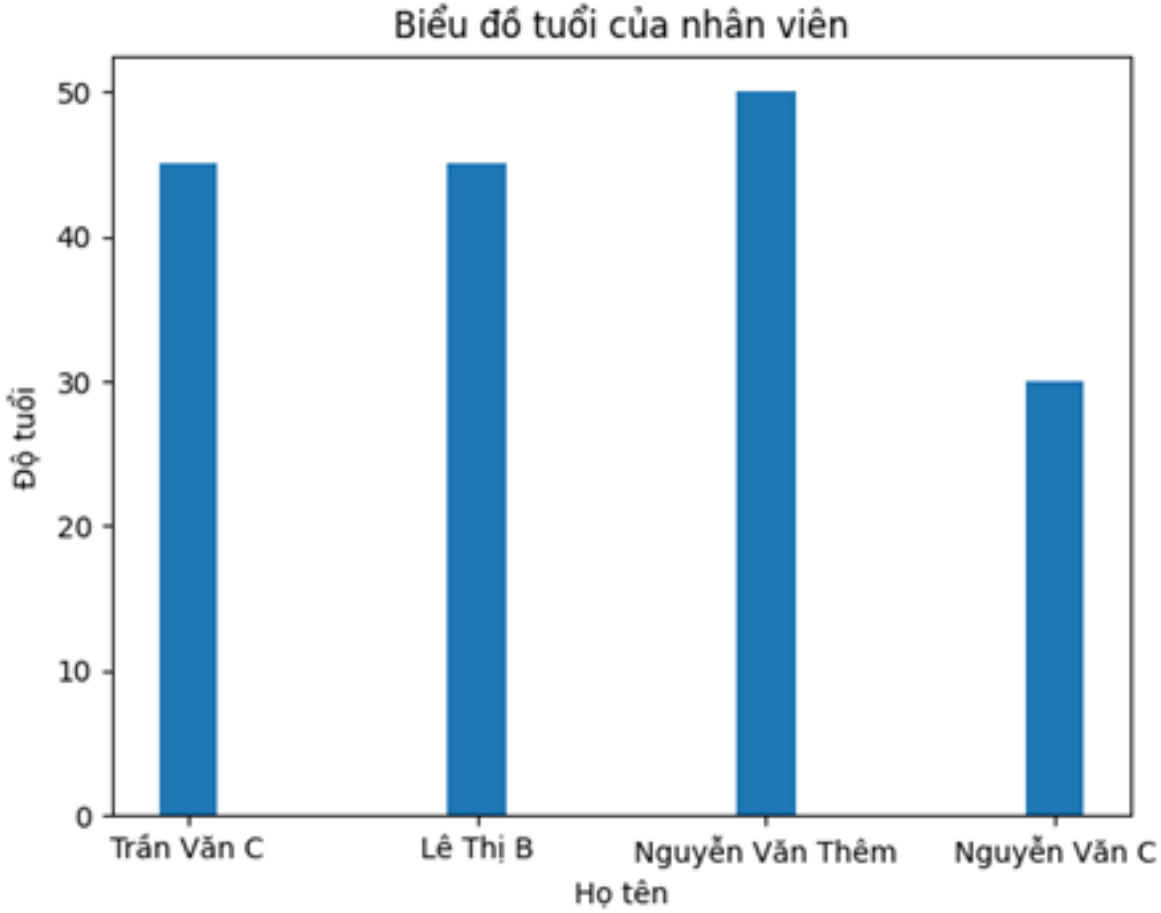
```
Tên = [row[0] for row in rows]
```

Với mỗi bản ghi (row) trong rows, truy cập phần tử đầu tiên (row[0], cột "Tên") và đưa nó vào danh sách Tên.

```
Tuổi= [row[1] for row in rows]
```

Tương tự, mỗi bản ghi (row) trong rows, truy cập phần tử thứ hai (row[1], cột "Tuổi") và đưa nó vào danh sách Tuổi.

Kết quả thực hiện chương trình:



b. Biểu đồ đường (Line Chart)

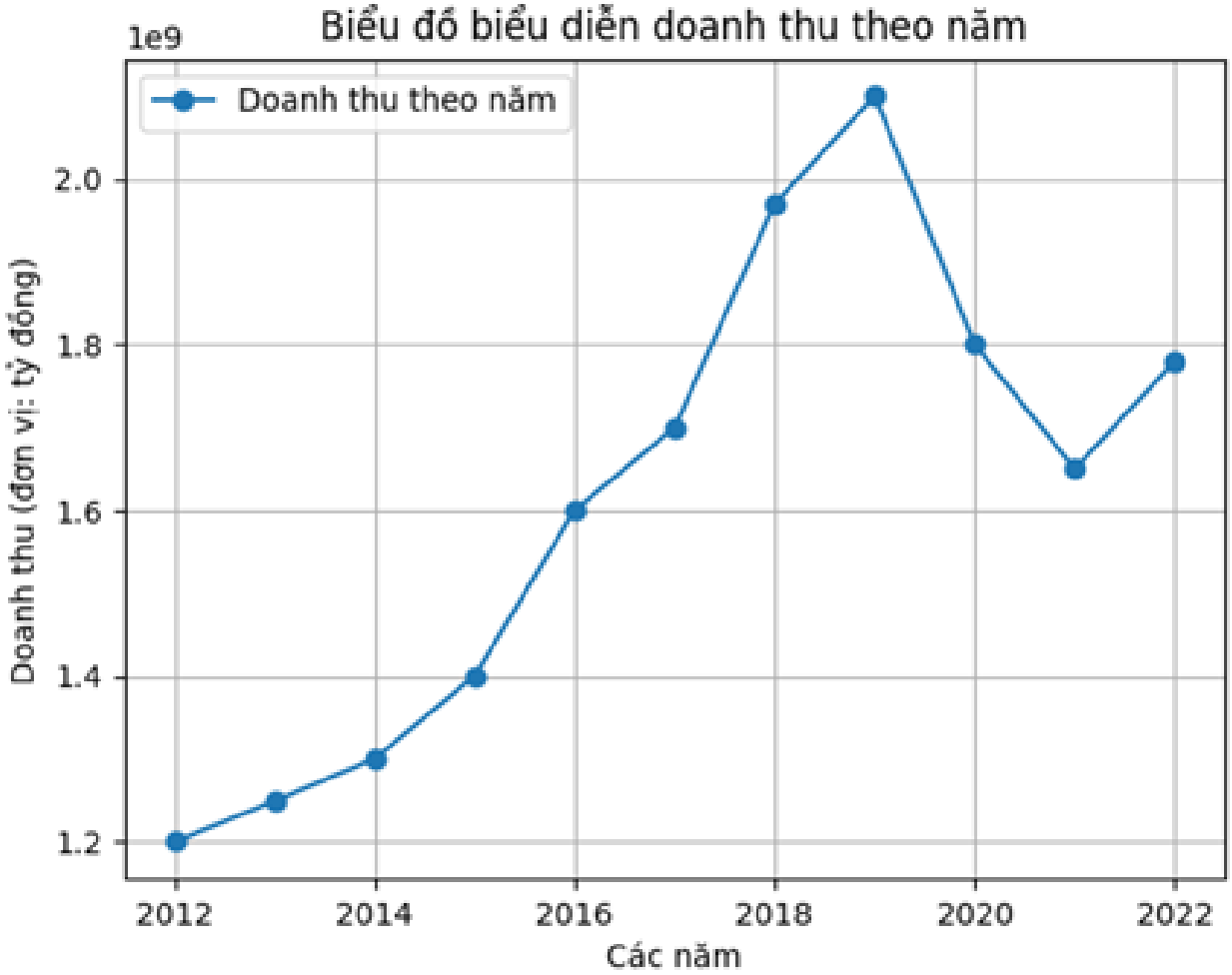
Đặc điểm của biểu đồ Line Chart:

- Thể hiện sự biến đổi của dữ liệu theo thời gian hoặc một biến số khác.
- Được tạo bởi việc nối các điểm dữ liệu bằng các đoạn đường thẳng.
- Dễ dàng nhìn thấy xu hướng, đỉnh, đáy và các biến đổi trong dữ liệu.
- Thích hợp cho dữ liệu có tính chất liên tục và có thời gian.

Ví dụ 5.3.3.2: Giả sử có số liệu cơ sở dữ liệu SQLite ‘sales.db’ về doanh thu hàng năm của một công ty từ năm 2012 đến năm 2022. Hãy vẽ biểu đồ dạng line Chart thể hiện doanh thu theo các năm của công ty đó.

```python
import sqlite3
import matplotlib.pyplot as plt

# Kết nối đến cơ sở dữ liệu
conn = sqlite3.connect('sales.db')
cursor = conn.cursor()

# Truy vấn dữ liệu từ cơ sở dữ liệu
cursor.execute("SELECT year, revenue FROM sales")
rows = cursor.fetchall()

# Tách dữ liệu thành hai danh sách riêng biệt cho năm và doanh thu
years = [row[0] for row in rows]
revenues = [row[1] for row in rows]

# Đóng kết nối
conn.close()

# Vẽ biểu đồ Line Chart
plt.plot(years, revenues, marker='o', linestyle='-')
plt.xlabel('Các năm')
plt.ylabel('Doanh thu (đơn vị: tỷ đồng)')
plt.title('Biểu đồ biểu diễn doanh thu theo năm')

plt.legend(['Doanh thu theo năm'])  # Thêm chú thích vào biểu đồ
plt.grid(True)  # Thêm lưới trên biểu đồ
plt.show()
```

Kết quả thực hiện chương trình:



c. Biểu đồ tròn (Pie Chart)

Một số đặc điểm và ưu điểm của biểu đồ Pie Chart:

- Biểu diễn tỷ lệ phần trăm: Biểu đồ Pie Chart cho phép dễ dàng nhìn thấy tỷ lệ phần trăm của từng thành phần trong tổng thể. Khi nhìn vào biểu đồ, ta có thể nhanh chóng nhận biết các thành phần chiếm bao nhiêu phần trăm trong tổng số.
- Thể hiện phân chia rõ ràng: Biểu đồ Pie Chart thể hiện phân chia rõ ràng giữa các nhóm hoặc loại dữ liệu. Điều này giúp hiển thị mối quan hệ giữa các thành phần và giúp người đọc dễ dàng so sánh tỷ lệ giữa các phần trong biểu đồ.
- Dễ hiểu và trực quan: Biểu đồ Pie Chart dễ hiểu và trực quan, thích hợp cho việc trình bày tỷ lệ phần trăm một cách trực quan và rõ ràng. Nó thường được sử dụng trong các báo cáo, bài thuyết trình, và công việc trình bày dữ liệu.

Hạn chế của biểu đồ Pie Chart:

- Không phù hợp cho nhiều phần: Khi số lượng thành phần trong biểu đồ quá nhiều, biểu đồ Pie Chart có thể trở nên khó nhìn và khó hiểu.
- Có thể gây nhầm lẫn: Khi các phần trong biểu đồ có tỷ lệ gần nhau, có thể dễ gây ra nhầm lẫn và khó đọc chính xác các giá trị.

Ví dụ 5.3.3.3: Giả sử chúng ta có database về mức chi tiêu hàng tháng của một hộ gia đình như sau:

Thực_phẩm	Đi lại	Nhà ở	Giáo dục	Giải trí	Y tế	Điện nước	iết kiệm
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
5000000.0	2000000.0	6000000.0	1500000.0	2000000.0	1000000.0	1500000.0	2000000.0

Viết chương trình vẽ biểu đồ thể hiện tỷ lệ chi tiêu của các khoản chi.

```
1  import sqlite3
2  import matplotlib.pyplot as plt
3
4  # Kết nối đến cơ sở dữ liệu
5  conn = sqlite3.connect('ct_giadinh.db')
6  cursor = conn.cursor()
7
8  # Truy vấn dữ liệu từ bảng 'khoan_chi'
9  cursor.execute("SELECT * FROM khoan_chi")
10 rows = cursor.fetchall()
11
12 # Lấy thông tin về các tên cột
13 column_names = [description[0] for description in
14 cursor.description][1:]
15
16 # Tách dữ liệu thành hai danh sách riêng biệt cho loại chi tiêu
17 và tổng số tiền
18 loai_chi = column_names
19 tien_chi = rows[0][1:]
20
21 # Đóng kết nối
22 conn.close()
```

```
23  # Vẽ biểu đồ Pie Chart
24  plt.pie(tien_chi, labels=loai_chi, autopct='%1.2f%%')
25  plt.title('Biểu đồ tỉ lệ mức chi tiêu một hộ gia đình')
26  plt.axis('equal')  # Để đảm bảo hình tròn
27  plt.show()
```

Kết quả thực hiện chương trình:

5.3.4. Sử dụng Python để thêm biến vào bảng cơ sở dữ liệu

Bước 1. Tạo cơ sở dữ liệu

```
conn = sqlite3.connect('pythonDB.db')
c = conn.cursor()
```

tạo mới một cơ sở dữ liệu nếu cơ sở dữ liệu đó không tồn tại. Nếu đã tồn tại một cơ sở dữ liệu có cùng tên thì sẽ không tạo mới nữa.

sử dụng một phương thức cursor() để khởi tạo cơ sở dữ liệu và kích hoạt trạng thái active – đang hoạt động.

Bước 2. Tạo bảng cơ sở dữ liệu

```
def create_table():
    c.execute('CREATE TABLE IF NOT EXISTS RecordONE (Number REAL, Name TEXT)')
```

tạo ra bảng cơ sở dữ liệu nếu bảng đó chưa tồn tại

Bước 3. Chèn thêm dữ liệu vào bảng cơ sở dữ liệu

```
def data_entry():
    number = 1234
    name = "KHDL"
    c.execute("INSERT INTO RecordONE (Number, Name) VALUES(?, ?)",
                                                    (number, name))
    conn.commit()
```

khai báo thêm một hàm nữa có tên là data_entry

lưu lại những thay đổi đã thực hiện ở trên.

Bước 4: gọi tới các phương thức và đóng kết nối cơ sở dữ liệu

```
create_table()
data_entry()

c.close()
conn.close()
```

Ví dụ 5.3.4.1. Sử dụng SQLite để tạo một bảng "bang_diem" với hai cột "ten" và "diem". Sau đó, nó chèn dữ liệu có giá trị 8.5 và tên sinh viên "Nguyễn Văn A" vào bảng ‘bang_diem.

```python
import sqlite3

# Kết nối đến cơ sở dữ liệu (hoặc tạo cơ sở dữ liệu mới nếu nó không tồn tại)
conn = sqlite3.connect('sinhvien.db')
c = conn.cursor()

# Tạo bảng và thêm dữ liệu
def tao_bang():
    c.execute('CREATE TABLE IF NOT EXISTS bang_diem (diem REAL, ten TEXT)')

def nhap_diem():
    diem = 8.5
    ten = 'Nguyễn Văn A'
    c.execute("INSERT INTO bang_diem (diem, ten) VALUES(?, ?)", (diem, ten))
    conn.commit()

tao_bang()
nhap_diem()

# In ra các dòng trong bảng sau khi cập nhật
cours = conn.execute("SELECT * FROM bang_diem")
for row in cours:
    print(row)

c.close()
conn.close()
```

Giải thích:

Dòng 1: là để import module sqlite3, cho phép chúng ta làm việc với cơ sở dữ liệu SQLite trong Python.

Dòng 4: tạo kết nối tới cơ sở dữ liệu SQLite có tên là 'pythonDB.db'. Nếu tệp SQLite không tồn tại, nó sẽ được tạo mới; nếu tệp đã tồn tại, kết nối sẽ được thiết lập đến tệp này.

Dòng 5: tạo một đối tượng con trỏ (cursor object) từ kết nối. Đối tượng con trỏ được sử dụng để thực thi câu lệnh SQL và thao tác với cơ sở dữ liệu.

Dòng 8-9 :định nghĩa hàm **tao_bang**() để tạo một bảng có tên là "bang_diem" trong cơ sở dữ liệu. Hàm này sử dụng phương thức execute() để thực thi câu lệnh SQL "**CREATE TABLE IF NOT EXISTS bang_diem (Number REAL, Name TEXT)**". Nếu bảng đã tồn tại, câu lệnh này không tạo bảng mới.

Dòng 11-15: định nghĩa hàm **nhap**() để chèn/thêm dữ liệu vào bảng "bang_diem". Hàm này sử dụng phương thức execute() kết hợp với tham số dạng chỗ trống ? để thêm dữ liệu vào bảng. Sau khi chèn dữ liệu, hàm gọi conn.commit() để lưu các thay đổi vào cơ sở dữ liệu.

Dòng 17 : gọi hàm tao_bang() để tạo bảng "bang_diem", nếu bảng chưa tồn tại.

Dòng 18: Gọi hàm nhap_diem() để chèn dữ liệu vào bảng "bang_diem".

Dòng 22: thực hiện truy vấn SQL "SELECT * FROM bang_diem", trong đó "SELECT *" có nghĩa là lấy tất cả các cột từ bảng "bang_diem". Kết quả truy vấn được lưu vào biến cours..

Dòng 23: sử dụng một vòng lặp for để lặp qua từng dòng trong kết quả truy vấn (cours). Mỗi lần lặp, biến row sẽ chứa một dòng dữ liệu từ bảng "bang_diem".

Dòng 24: in ra giá trị của biến row, tức là in ra dòng dữ liệu từ bảng "bang_diem" trong cơ sở dữ liệu.

Dòng 26: đóng đối tượng con trỏ.

Dòng 27: đóng kết nối tới cơ sở dữ liệu

5.3.5. Kết nối MySQL trong Python

a. Giới thiệu MySQL

b. Kết nối từ xa đến CSDL MySQL bằng ngôn ngữ Python

Bước 1. Cài đặt MySQL: Cài đặt MySQL Server

Tải và cài đặt MySQL Server tại: https://dev.mysql.com/downloads/mysql/

Bước 2. Cài đặt thư viện kết nối MySQL cho Python

Cách 1. Sử dụng Pip: Trong cửa sổ cmd (ở chế độ Administrator), gõ lệnh:

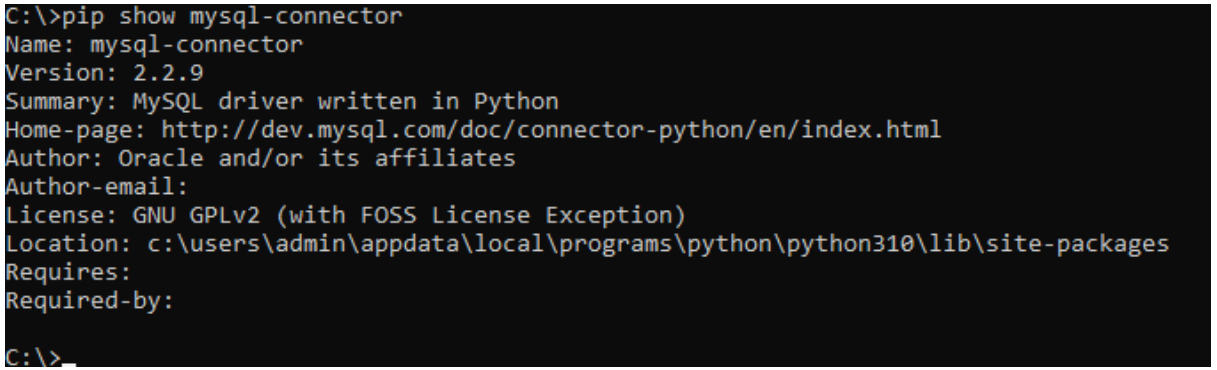
```
python -m pip install mysql-connector
```

Cách 2. Cài đặt phần mềm mySQL.connector.

Vào link sau và tải driver cài đặt tương ứng với HĐH:

https://dev.mysql.com/downloads/connector/python/

Bước 3. Kiểm tra kết nối? gõ lệnh C:\>**pip** show mysql-connector



Bước 4. Kiểm tra đã tồn tại CSDL mySQL?

B1. Mở chế độ SQL CLI: Trong cửa sổ cmd (ở chế độ Administrator), gõ lệnh: **mysql -u** root **-p**

```
C:\>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 9.1.0 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

B1. Tại dấu nhắc 'mysql>' , gõ lệnh: SHOW DATABASES;

mysql> **SHOW** DATABASES**;**

CSDL mysql đã tồn tại

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| mysqldb            |
| performance_schema |
| sakila             |
| sys                |
| world              |
+--------------------+
7 rows in set (0.00 sec)

mysql>
```

Nếu chưa tồn tại CSDL có thể tạo mới bằng cách:

mysql> **CREATE** DATABASE **mySQLdb;**

Query OK, 1 row affected (0.01 sec)

Ví dụ 5.3.5.1: Viết chương trình kết nối đến cơ sở dữ liệu mySQLdb

Ví dụ 5.3.5.1:

```python
import mysql.connector

def mysqlconnect():
    # Thử kết nối đến cơ sở dữ liệu MySQL
    try:
        db_connection = mysql.connector.connect(
            host="localhost",
            user="root",
            password="uneti",
            database="mySQLdb"
        )
    # Nếu connection không thành công
    except mysql.connector.Error as err:
        print("Không thể kết nối đến database mySQLdb:", err)
        return 0

    # Nếu kết nối thành công
    print("Đã kết nối đến Database.")

    # Tạo đối tượng con trỏ để thực thi truy vấn
    cursor = db_connection.cursor()

    # Thực thi truy vấn
    cursor.execute("SELECT CURDATE();")

    # Lấy dữ liệu từ kết quả truy vấn
    m = cursor.fetchone()

    # In kết quả
    print("Hôm nay là ngày :", m[0])

    # Đóng kết nối cơ sở dữ liệu
    db_connection.close()

# Gọi hàm kết nối
mysqlconnect()
```

```
Đã kết nối đến Database.
Hôm nay là ngày : 2024-11-25
```

5.3.6 . Quản lý cơ sở dữ liệu bằng PostgreSQL

Ví dụ 5.3.6.1:

```
import psycopg2

try:
    connection = psycopg2.connect(
        dbname="my_database",
        user="postgres",
        password="uneti",
        host="localhost",
        port="5432"
    )
    print("Đã kết nối đến cơ sở dữ liệu PostgreSQL.")
except Exception as e:
    print("Không thể kết nối đến PostgreSQL:", e)

#Thực thi truy vấn SQL
cursor = connection.cursor()
cursor.execute("SELECT * FROM employees;")
rows = cursor.fetchall()
for row in rows:
    print(row)

connection.close()
```

CÂU HỎI THẢO LUẬN

1. Giải thích SQLite là gì?
2. Nêu các bước cơ bản để kết nối chương trình Python với CSDL MySQLite.
3. Nêu sự khác nhau giữa SQL và SQLite?
4. Nêu các ưu điểm của hệ quản trị CSDL SQLite?
5. Khi nào nên sử dụng SQLite và khi nào không sử dụng SQLite?

BÀI TẬP VẬN DỤNG:

Các bài tập 5.1, 5.2 ---, 5.11 trang 275, 276 TLHT