CƠ SỞ DỮ LIỆU SQLITE

GIẢNG VIÊN: THS.LÂM DU ĐẠT

Email: datld@donga.edu.vn

Trường Đại học Đông Á : Khoa Điện – Điện tử

CƠ SỞ DỮ LIỆU SQLITE

MỤC TIÊU BÀI HỌC

- **Hiểu**: Vai trò của CSDL (SQLite) trong dự án IoT
- **NẮM VỮNG**: 4 thao tác CRUD (Create, Read, Update, Delete) bằng câu lệnh SQL.
- **BIẾT**: Cách dùng thư viện sqlite3 của Python để thực thi các lệnh CRUD.
- **HIỂU**: Lợi ích của việc chuyển đổi dữ liệu thô (tuple) sang Đối tượng (Data Classes) (như SensorData, SettingData).

CHUẨN ĐẦU RA

- **Viết**: Được các hàm Python để thực thi 4 thao tác CRUD
- **Chuyển đổi**: Được kết quả truy vấn (dạng tuple) thành danh sách các đối tượng

KHỞI TẠO DỮ LIỆU VỚI SQLITE

ĐẶT VẤN ĐỀ

VẤN ĐỀ 1: MẤT DỮ LIỆU LỊCH SỬ

- Ở chế độ "auto", Arduino gửi dữ liệu nhiệt độ, độ ẩm mỗi 5 phút để app "lưu vào database
- Câu hỏi: Nếu app Python tắt, làm sao chúng ta vẽ biểu đồ (IoT Dashboard...) hoặc xem lại lịch sử? Dữ liệu trên RAM sẽ mất.

VẤN ĐỀ 2: MẤT CÀI ĐẶT NGƯỠNG

- Ở chế độ "manual", người dùng "thiết lập ngưỡng" (ví dụ: 35°C).
- Câu hỏi: Khi tắt app và mở lại, làm sao app nhớ được ngưỡng 35°C mà người dùng đã cài?

GIÀI PHÁP LÀ GÌ?:

Chúng ta cần LƯU TRỮ DỮ LIỆU LÂU DÀI (Persistent Storage).

SQLITE

- Một thư viện CSDL (có sẵn trong Python).
- Siêu nhẹ, không cần cài đặt.
- Lưu toàn bộ CSDL vào 1 file duy nhất (trong dự án là iot.db).
- Hoàn hảo cho ứng dụng IoT.

NÓI CHUYỆN TRỰC TIẾP VỚI CSDL (SỬ DỤNG CLI)

- Mở terminal/CMD
- Mở cơ sở dữ liệu/Tạo mới: `sqlite3 iot.db`
- Kiểm tra danh sách bảng: `.tables`
- Tạo bảng sensor_data:

```
CREATE TABLE sensor_data (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    temperature REAL,
    humidity REAL,
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

- Kiểm tra cấu trúc bảng sensor_data: `.schema sensor_data`
- Thoát cửa sổ gõ lệnh: `.exit`

THỰC HÀNH TẠO BẢNG `fan_data`

Hãy tạo thêm bảng **fan_data** với thông số sau bằng CLI

- Mục đích: Lưu trữ lịch sử trạng thái Bật/Tắt ('ON'/'OFF') của quạt

Tên cột	Kiểu dữ liệu	Thuộc tính	Ý nghĩa
id	INTEGER	PRIMARY KEY AUTOINCREMENT	ID tự tăng, không trùng lặp
value	TEXT	DEFAULT "OFF"	Trạng thái, mặc định là "OFF"
timestamp	DATETIME	DEFAULT CURRENT_TIMESTAMP	Mốc thời gian dữ liệu được ghi

THỰC HÀNH TẠO BẢNG `setting_data`

Hãy tạo thêm bảng setting_data với thông số sau bằng CLI

- Mục đích: Lưu trữ ngưỡng cài đặt (cấu hình) của người dùng. Chỉ có 1 dòng duy nhất (id = 1) trong bảng này.

Tên cột	Kiểu dữ liệu	Thuộc tính	Ý nghĩa
id	INTEGER	PRIMARY KEY AUTOINCREMENT	ID tự tăng
temperature_threshold	REAL	DEFAULT 35.0	Ngưỡng nhiệt độ mặc định 35.0
humidity_threshold	REAL	DEFAULT 70.0	Ngưỡng độ ẩm mặc định 70.0
timestamp	DATETIME	DEFAULT CURRENT_TIMESTAMP	Mốc thời gian dữ liệu được ghi

CRUD QUA COMMAND LINE

CRUD: CREATE READ UPDATE DELETE

INSERT (THÊM DỮ LIỆU - CREATE)

- Bối cảnh: Đây là lúc Arduino gửi dữ liệu hoặc ta cài đặt giá trị mặc định cho app.

Thêm dữ liệu cảm biến

```
INSERT INTO sensor_data (temperature, humidity)
VALUES (30.5, 75.2);
```

Thêm cài đặt mặc định

```
INSERT INTO setting_data (temperature_threshold, humidity_threshold)
VALUES (35.0, 70.0);
```

THỰC HÀNH LỆNH `insert`

- Hãy insert dữ liệu sau vào bảng sensor_data với thông tin sau:

temperature	humidity	timestamp
30.5	65.2	'2025-10-03 08:00:00'
40.9	70.1	'2025-10-03 08:10:00'
50.7	75.5	'2025-10-03 08:20:00'
80.7	90.5	'2025-10-03 08:30:00'
70.7	70.5	'2025-10-03 08:40:00'

- Hãy insert dữ liệu sau vào bảng fan_data với thông tin sau:

value	timestamp
'ON'	'2025-10-29 08:30:40'
'OFF'	'2025-10-29 09:00:40'
'ON'	'2025-10-29 09:00:40'

SELECT (ĐỌC/TRUY VẤN DỮ LIỆU - READ)

Lấy tất cả dữ liệu trong bảng: SELECT *

```
SELECT * FROM sensor_data;
```

Lấy dữ liệu có điều kiện: SELECT ... WHERE

Lấy record đầu tiên trong bảng `sensor_data`

```
SELECT * FROM sensor_data WHERE id = 1;
```

Lấy dữ liệu mới nhất: SELECT .. ORDER BY...LIMIT

Lấy 10 record mới nhất trong bảng `sensor_data`

```
SELECT * FROM sensor_data ORDER BY timestamp DESC LIMIT 10;
```

SELECT NÂNG CAO:(THỐNG KÊ DỮ LIỆU)

link: SQLite Aggregate Functions

Cú pháp: SELECT *Aggregate Functions* ... FROM ... WHERE ...

Tính nhiệt độ trung bình: Hàm AVG

```
SELECT AVG(temperature) FROM sensor_data;
```

Tính nhiệt độ, độ ẩm trung bình trong ngày 3/10

```
SELECT AVG(temperature) , AVG(humidity) FROM sensor_data WHERE timestamp
BETWEEN '2025-10-03 00:00:00' AND '2025-10-03 23:59:59';
```

Tính max , min nhiệt độ: Hàm MAX, MIN

```
SELECT MAX(temperature), MIN(temperature) FROM sensor_data;
```

THỰC HÀNH LỆNH SELECT

SInh viên thực hành làm theo các bước sau

- Thêm dữ liệu cho bảng `sensor_data`

temperature	humidity	timestamp
30.5	65.2	'2025-10-29 08:00:00'
40.9	70.1	'2025-10-29 08:10:00'
50.7	75.5	'2025-10-29 08:20:00'
80.7	90.5	'2025-10-29 08:30:00'
70.7	70.5	'2025-10-29 08:40:00'

- Lấy tất cả thông tin bảng `sensor_data`
- Lấy tất cả bản ghi trong bảng `sensor_data` khi điều kiện nhiệt độ >35 độ
- Lấy bản ghi mới nhất trong bàng `sensor_data`
- Lấy nhiệt độ Max, Min trong ngày 29/10
- Lấy nhiệt độ trung bình trong ngày 29/10

UPDATE (CẬP NHẬP DỮ LIỆU - UPDATE)

Bối cảnh: Người dùng kéo thanh trượt (Scale) trên giao diện Tkinter (IoT Dashboard...) để đổi ngưỡng nhiệt độ từ 35 lên 37.

- Cú pháp UPDATE ... SET... WHERE ...

```
UPDATE setting_data SET temperature_threshold = 37.0 WHERE id = 1;
```

- Kiểm tra: Gõ lại `SELECT * FROM setting_data;` để thấy giá trị đã thay đổi.
- Chú ý: Nếu UPDATE mà quên WHERE, tất cả các dòng sẽ bị cập nhật!

THỰC HÀNH LỆNH UPDATE

Bối cảnh: Người dùng kéo thanh trượt (Scale) trên giao diện Tkinter (IoT Dashboard...) để đổi ngưỡng độ ẩm từ 70 xuống 60

- Thực hiện lệnh `update` xuống bảng `setting_data`
- kiểm tra lại bằng lệnh `select`

DELETE(XÓA DỮ LIỆU)

Bối cảnh: Thường trong **hệ thống Iot** dữ liệu cũ sẽ được xóa trong khoảng thời gian nhất định (1 tháng , 6 tháng, 1 năm) nhằm giải phóng bộ nhớ

- Cú pháp: DELETE FROM ... WHERE ...
- Xóa dữ liệu cảm biến trong khoảng thời gian 03/10/2025 08:00 đến 03/10/2025 08:20

```
DELETE FROM sensor_data
WHERE timestamp BETWEEN '2025-10-03 08:00:00' AND '2025-10-03 08:20:00';
```

- **Chú ý**: Khi `DELETE` luôn kèm theo điều kiện `WHERE` nếu không sẽ xóa hết tất cả dữ liệu của bảng

THỰC HÀNH LỆNH DELETE

Bối cảnh : Thực hành xóa dữ liệu `sensor_data`

- Dùng lệnh `SELECT` để đọc tất cả dữ liệu
- Dùng lệnH `DELETE` để xóa dữ liệu theo cá bước sau
  - Xóa dữ liệu theo `id`: chọn một `id` bất kỳ và xóa dữ liệu
  - Xóa dữ liệu quá nóng: Thực hiện xóa dữ liệu khi điều kiện nhiệt độ >40 độ

TỰ ĐỘNG HÓA CRUD BẰNG PYTHON

MỤC TIÊU

- SV hiểu cách dùng thư viện sqlite3 để thực thi 4 lệnh CRUD (INSERT, SELECT, UPDATE, DELETE).
- SV hiểu tầm quan trọng của việc dùng Placeholder (?) để chống lỗi SQL Injection.
- SV phân biệt được fetchone() (lấy 1) và fetchall() (lấy nhiều).

CHUẨN ĐẦU RA

- Viết được các hàm Python để INSERT, SELECT, UPDATE, DELETE
- Biết cách "gói" dữ liệu tuple (kết quả thô từ CSDL) vào các Data Class (như SensorData, SettingData)

ĐẶT VẤN ĐỀ

Chúng ta đã biết (Tiết 1, 2):

- Dùng CLI (sqlite3 iot.db) để "gõ tay" các lệnh SQL (CRUD) và thấy kết quả ngay.

Vấn đề

- Ứng dụng Python (IoT Dashboard...) không thể "gõ tay" như vậy.
- Làm sao App của chúng ta tự động:
  - INSERT dữ liệu sensor_data khi Arduino gửi lên?
  - UPDATE ngưỡng setting_data khi người dùng kéo thanh trượt?
  - SELECT dữ liệu để vẽ biểu đồ?

Giải pháp

Dùng thư viện **sqlite3** của Python để "gửi" các lệnh SQL này đi.

5 BƯỚC KẾT NỐI VỚI sqlite

```
# 1. Import thư viện
import sqlite3
# 2. Kết nối (Connect)
# Lệnh này sẽ tạo file 'iot.db' nếu nó chưa tồn tại
conn = sqlite3.connect('iot.db')
# 3. Tạo con trỏ (Cursor)
# Con trỏ là đối tượng để ta ra lệnh SQL
cursor = conn.cursor()
# 4. Tạo lệnh sql và thực thi (Execute)
sql_command = "..."
cursor.execute(sql_command)
# 5. Lưu thay đổi (Commit) và Đóng (Close)
conn.commit() # Bắt buộc phải có để LƯU lại thay đổi
conn.close()  # Đóng kết nối
```

GHI DỮ LIỆU VÀO DATABASE (INSERT)

Arduino gửi dữ liệu cảm biến (ví dụ: T=30.5, H=70)

```
import sqlite3
conn = sqlite3.connect('iot.db')
cursor = conn.cursor()
new_temp = 30.5
new_hum = 70.0
# 1. Câu lệnh SQL
sql = "INSERT INTO sensor_data (temperature, humidity) VALUES (?, ?)"
# 2. Dữ liệu (phải là 1 tuple)
params = (new_temp, new_hum)
# 3. Thực thi
cursor.execute(sql, params)
# 4. BẮT BUỘC KHI THAY ĐỔI DỮ LIỆU!
conn.commit()
conn.close()
```

BẢO MẬT: LỖI SQL INJECTION LÀ GÌ?

- Nếu trong code ta viết câu lệnh như sau

```
sql = f"INSERT INTO sensor_data (temperature, humidity) VALUES
({new_temp}, {new_hum})"
```

- Chyện gì xảy ra khi kẻ tấn công nhập trên giao diện

`new_temp_doc_hai = "30.5); DROP TABLE setting_data; --"`

Lúc này lệnh sql sẽ là

```
sql = INSERT INTO sensor_data (temperature, humidity) VALUES (30.5);
DROP TABLE setting_data; --, 70.0)
```

Kết quả: Xóa sạch dữ liệu database: **DROP TABLE setting_data**

GIẢI PHÁP: LUÔN DÙNG PLACEHOLDER (?) thực thi lệnh sql với tham số

TRUY VẤN DỮ LIỆU (SELECT)

- Ghi dữ liệu thì phải commit(). Nhưng đọc dữ liệu (SELECT) thì KHÔNG cần commit(). Thay vào đó, chúng ta cần 'LẤY' (Fetch) kết quả về."
- Các hàm `Fetch`
  - `cursor.fetchone()`: Lấy **1** dòng đầu tiên (trả về 1 tuple). Nếu không có, trả về None.
  - `cursor.fetchall()`: Lấy **TẤT CẢ** các dòng còn lại (trả về 1 list các tuple). Nếu không có, trả về list rỗng [].

SELECT + fetchone

- Khi app khởi động, nó cần lấy ngưỡng cài đặt hiện tại (chỉ 1 dòng) từ setting_data để hiển thị

```
# ... (kết nối)
sql = "SELECT * FROM setting_data WHERE id = 1"
cursor.execute(sql)

# Lấy 1 dòng
data_tuple = cursor.fetchone()
# data_tuple sẽ là: (1, 38.0, 85.0, '2025-10-29...')

if data_tuple:
    print(f"Ngưỡng nhiệt (kiểu tuple): {data_tuple[1]}")
    print(f"Ngưỡng ẩm (kiểu tuple): {data_tuple[2]}")

conn.close()
```

SELECT + fetchall

- Để vẽ biểu đồ, app cần lấy 10 giá trị cảm biến mới nhất

```
# ... (kết nối)
num_records = 10
sql = "SELECT * FROM sensor_data ORDER BY timestamp DESC LIMIT ?"
cursor.execute(sql, (num_records,)) # tuple
# Lấy TẤT CẢ các dòng (tối đa 10)
list_of_tuples = cursor.fetchall()
# [(5, 31.0, ...), (4, 30.5, ...), (3, ...)]
for row_tuple in list_of_tuples:
    print(f"Nhiệt độ: {row_tuple[1]}") # Vẫn dùng index
conn.close()
```

CẬP NHẬP DỮ LIỆU (UPDATE)

Tình huống: Người dùng đang ở chế độ "manual" Họ kéo thanh trượt (Scale) để thay đổi ngưỡng nhiệt độ.

Nhiệm vụ: Ứng dụng Python phải ngay lập tức gửi lệnh UPDATE đến CSDL để cập nhật dòng id=1 trong bảng `setting_data`

```
# ... (kết nối)
# Giả sử new_temp = 37.5, new_hum = 82.0
sql = "UPDATE setting_data SET temperature_threshold = ?,
humidity_threshold = ? WHERE id = ?"
params = (37.5, 82.0, 1)
cursor.execute(sql, params)
conn.commit()
```

XÓA DỮ LIỆU (DELETE)

- **Tình huống**: Bảng sensor_data ngày càng lớn (dữ liệu 5 phút/lần)
- **Nhiệm vụ**: Chúng ta cần "dọn dẹp"

Xóa bản ghi theo id (trường hợp lỗi)

```
# Giả sử muốn xóa bản ghi có id = 5
sql = "DELETE FROM sensor_data WHERE id = ?"
params = (5,) # Phải là tuple (5,) chứ không phải (5)
cursor.execute(sql, params)
conn.commit()
```

Xóa bản ghi sau 30 ngày

```
sql = "DELETE FROM sensor_data WHERE timestamp < datetime('now', '-30
days')"
cursor.execute(sql)
conn.commit()
```

MODEL CLASS (LỚP MÔ HÌNH)

ĐẶT VẤN ĐỀ

- Sau khi chạy `cursor.execute("SELECT * ...")`
  - `cursor.fetchone()` → 1 tuple, ví dụ: `(1, 35.0, 70.0, '2025-10-30...')`
  - `cursor.fetchall()` → 1 list các tuple, ví dụ: `[(1, 30.5, 75.2, ...), (2, 31.0, 76.0, ...)]`
- Khi ta dùng kết quả này trong ứng dụng

```
# ... (kết nối, execute) ...
sensor_list = cursor.fetchall() # Trả về list các tuple
for data in sensor_list:
    nhiet_do_cam_bien = data[1] # vấn đề phát sinh
```

- Cách làm rất tệ: Khó đọc (Dùng Magic number): [1] là gì? [2] là gì?

Giải pháp: Chúng ta sẽ sử dụng các "khuôn" (class) đã được định nghĩa sẵn

THỰC HÀNH TẠO MODEL CLASS: sensor_data

BƯỚC 1: TẠO MODEL CLASS

```
class SensorData:
    def __init__(self, id, temperature, humidity, timestamp):
        self.id = id
        self.temperature = temperature
        self.humidity = humidity
        self.timestamp = timestamp

    def __repr__(self):
        # Hàm này giúp print object ra đẹp hơn
        return f"<SensorData T={self.temperature}, H={self.humidity}>"
```

BƯỚC 2: TRUY VẤN VÀ TRẢ VỀ MODEL CLASS

```
def get_sensors_from_num(num)
 results_list = [] # List rỗng để chứa các object
 conn = sqlite3.connect('iot.db')
 cursor = conn.cursor()
 sql = "SELECT * FROM sensor_data ORDER BY timestamp DESC LIMIT ?"
 params = (num,)
 cursor.execute(sql, params)
 list_of_tuples = cursor.fetchall()
 for row in list_of_tuples:
   data_obj = SensorData(id=row[0],temperature=row[1],
   humidity=row[2],
   timestamp=row[3])
   results_list.append(data_obj)
 conn.close()
 return results_list
```

KẾT THÚC