

Public State, Private State và Multiple State Schemas trong LangGraph

Ví Dụ Minh Họa: Cửa Hàng Sửa Máy Tính

1. Public State - Giao Tiếp Đơn Giản Với Khách Hàng

- **Kịch bản:** Bạn mang máy tính bị hỏng đến cửa hàng sửa chữa.
- **Cuộc trò chuyện:**
 - Bạn: "Máy tính của tôi không hoạt động nữa, tôi không biết lý do."
 - Nhân viên: "Chúng tôi sẽ kiểm tra và báo lại cho bạn sau."
- **Đặc điểm:**
 - Thông tin **đơn giản**, dễ hiểu.
 - Không đi vào chi tiết kỹ thuật.

→ Đây chính là **public state** trong Landgraaf: trạng thái mà người dùng cuối (khách hàng) tương tác trực tiếp.

2. Private State - Giao Tiếp Nội Bộ Kỹ Thuật

- **Kịch bản:** Nhân viên chuyển máy tính cho đội kỹ thuật.
- **Cuộc trò chuyện:**
 - Kỹ thuật viên 1: "Kiểm tra nguồn điện, có thể do adapter."
 - Kỹ thuật viên 2: "Test RAM và ổ cứng, có thể bị bad sector."
- **Đặc điểm:**
 - Thông tin **kỹ thuật phức tạp**, chỉ dành cho nội bộ.
 - Khách hàng không cần biết chi tiết này.

→ Đây là **private state**: trạng thái nội bộ để xử lý logic phức tạp.

3. Kết Quả Cuối Cùng - Quay Lại Public State

- Kỹ thuật viên tổng hợp kết quả và báo lại cho nhân viên:
 - "Máy bị lỗi ổ cứng, cần thay thế với chi phí 3 triệu đồng."
- Nhân viên thông báo lại cho khách hàng:

- "Máy của bạn bị hỏng ổ cứng, chúng tôi sẽ sửa trong 2 ngày."

→ Thông tin được **đơn giản hóa** từ private state trở lại public state.

Áp Dụng Vào LangGraph

1. Public State Schema

- **Vai trò:** Giao diện tương tác với người dùng.
- **Đặc điểm:**
 - Dữ liệu đầu vào/đầu ra đơn giản.
 - Ví dụ: Form đăng ký, kết quả tìm kiếm.

2. Private State Schema

- **Vai trò:** Xử lý logic nghiệp vụ phức tạp.
- **Đặc điểm:**
 - Chứa các biến và thuật toán kỹ thuật.
 - Ví dụ: Xử lý dữ liệu từ database, tính toán AI.

3. Chuyển Đổi Giữa Các Trạng Thái

- **Node 1 (Nhận Public State → Tạo Private State):**
 - Nhận thông tin đơn giản từ người dùng.
 - "**Tăng độ phức tạp lên 1000 lần**" để xử lý nội bộ.
- **Node 2 (Nhận Private State → Trả Public State):**
 - Tổng hợp kết quả phức tạp.
 - "**Giảm độ phức tạp 999 lần**" để trả về người dùng.

Tại Sao Điều Này Quan Trọng?

1. **Bảo Mật:** Private state giúp che giấu logic nhạy cảm khỏi người dùng.
2. **Hiệu Suất:** Tách biệt xử lý phức tạp khỏi giao diện người dùng.
3. **Dễ Bảo Trì:** Thay đổi logic kỹ thuật mà không ảnh hưởng đến trải nghiệm người dùng.

Lời Khuyên Khi Áp Dụng

- **Ghi nhớ khái niệm:** Hiểu rõ sự khác biệt giữa public/private state.
- **Đừng sa lầy vào chi tiết ngay:** Tập trung vào bài học thực hành trước.
- **Quay lại khi cần:** Khi xây dựng ứng dụng phức tạp, hãy tra cứu lại tài liệu này.

“LandGraph không chỉ dành cho siêu anh hùng - nó đơn giản nếu bạn hiểu cách tổ chức trạng thái!”

Kết Luận

Việc phân chia **public/private state** giống như cách một cửa hàng sửa chữa hoạt động:

- **Public state** = Giao tiếp với khách hàng.
- **Private state** = Thảo luận nội bộ kỹ thuật.

Hiểu được điều này sẽ giúp bạn thiết kế ứng dụng LangGraph mạnh mẽ và dễ bảo trì!

Multiple State Schemas Qua Ví Dụ Cửa Hàng Sửa Máy Tính

1. Multiple State Schemas Là Gì?

Trong LangGraph, **Multiple State Schemas** (Đa lược đồ trạng thái) là cách phân chia trạng thái ứng dụng thành nhiều "lớp" khác nhau, mỗi lớp có:

- **Mục đích riêng** (giao tiếp với người dùng vs xử lý nội bộ)
- **Độ phức tạp riêng** (đơn giản vs kỹ thuật)
- **Phạm vi truy cập riêng** (public vs private)

Giống như một cửa hàng có khu vực bán hàng (public) và xưởng sửa chữa (private).

2. Áp Dụng Vào Ví Dụ Cửa Hàng Sửa Máy Tính

Schema 1: Public State Schema

(Khu vực tiếp tân)

- **Đặc điểm:**
 - Ngôn ngữ: Đơn giản, không chuyên môn
 - Dữ liệu: Chỉ thông tin cần thiết cho khách hàng
- **Ví dụ:**

```
// Public State Schema
{
  customerName: "Nguyễn Văn A",
  device: "Laptop Dell XPS",
  problem: "Không khởi động được",
  status: "Đang chẩn đoán"
```

}

Schema 2: Private State Schema

(Xưởng kỹ thuật)

- **Đặc điểm:**
 - Ngôn ngữ: Kỹ thuật, chi tiết
 - Dữ liệu: Đầy đủ thông tin để sửa chữa

- **Ví dụ:**

```
// Private State Schema
{
  deviceId: "DELL-XPS-2023-SN-12345",
  diagnosticLogs: [
    { test: "Power Supply", result: "12V rail unstable" },
    { test: "HDD S.M.A.R.T", result: "Reallocated sectors: 512" }
  ],
  repairPlan: [
    { action: "Replace PSU", part: "PSU-DELL-120W" },
    { action: "Clone HDD to SSD", tools: ["Acronis", "USB-SATA adapter"] }
  ]
}
```

3. Cách Chuyển Đổi Giữa Các State Schema

Quy Trình Xử Lý

1. Public → Private (Node 1)

- Nhận yêu cầu đơn giản từ khách
- **"Mở rộng" thành thông số kỹ thuật**

```
def public_to_private(public_state):
    private_state = {
        'deviceId': lookup_device_id(public_state['device']),
        'diagnosticLogs': run_hardware_tests(),
        'repairPlan': generate_repair_options()
    }
    return private_state
```

2. Private → Public (Node 2)

- Tổng hợp kết quả kỹ thuật
- **"Rút gọn" thành thông báo dễ hiểu**

- ```
def private_to_public(private_state):
 public_state_update = {
 'status': "Đã xác định lỗi",
 'solution': f"Thay {private_state['repairPlan'][0]['part']}",
 'cost': calculate_cost(private_state)
 }
 return public_state_update
```

## 4. Lợi Ích Của Multiple State Schemas

| Tiêu Chí      | Public State | Private State            |
|---------------|--------------|--------------------------|
| Đối tượng     | Khách hàng   | Kỹ thuật viên            |
| Độ phức tạp   | 1x (dễ hiểu) | 1000x (chi tiết)         |
| Bảo mật       | Ai cũng thấy | Nội bộ                   |
| Ví dụ thực tế | App Mobile   | Database + Microservices |

### Ưu điểm:

- Giảm tải thông tin cho người dùng cuối
- Dễ dàng thay đổi logic nghiệp vụ mà không ảnh hưởng giao diện
- Bảo mật thông tin nhạy cảm (giá vốn, lỗi hệ thống...)

## 5. Sai Lầm Cần Tránh

### 1. Nhồi nhét private state vào public

→ Làm người dùng bối rối với thông báo kiểu:  
*"Lỗi 0x7F do sector 512 bị bad, cần remap cluster"*

### 2. Thiếu chuyển đổi giữa các schema

→ Dẫn đến mất mát thông tin khi giao tiếp giữa các thành phần

### 3. Dùng chung schema cho mục đích khác nhau

→ Như bắt kỹ thuật viên dùng form đơn giản của khách để ghi chép

# Câu Hỏi Thực Hành

Hãy thử thiết kế Multiple State Schemas cho các tình huống sau:

### 1. Ứng dụng ngân hàng

- Public: Số dư tài khoản
- Private: Lịch sử giao dịch chi tiết + thuật toán chống gian lận

## 2. Trợ lý ảo y tế

- Public: Triệu chứng đơn giản ("Đau đầu 3 ngày")
- Private: Dữ liệu EHR + mô hình chẩn đoán AI

“☐ **Mẹo:** Luôn tự hỏi "*Thông tin này có CẦN THIẾT cho người dùng cuối không?*" khi thiết kế schema.

Tác giả: **Đỗ Ngọc Tú**  
Công Ty Phần Mềm **VHTSoft**

Phiên bản #2

Được tạo 23 tháng 4 2025 04:40:27 bởi Đỗ Ngọc Tú

Được cập nhật 23 tháng 4 2025 05:17:15 bởi Đỗ Ngọc Tú