# Tài liệu Kỹ thuật: Kiến trúc Quy tắc (Rule Architecture) v1.0

**Dự án:** ITAPIA

**Ngày:** 26/07/2025

**Version:** 1.0

### 1. Tổng quan và Mục tiêu Thiết kế

Kiến trúc Quy tắc (Rule Architecture) là nền tảng cốt lõi cho module **Advisor** của hệ thống ITAPIA. Nó định nghĩa một "ngôn ngữ" chuẩn hóa để biểu diễn, lưu trữ, thực thi và giải thích các quy tắc ra quyết định.

Các mục tiêu thiết kế chính cho phiên bản này bao gồm:

- **Tính biểu cảm (Expressiveness):** Có khả năng biểu diễn các quy tắc logic và toán học phức tạp, kết hợp nhiều nguồn dữ liệu từ các báo cáo phân tích ( `Technical` , `News` , `Forecasting` ).
- **Tính Nhất quán & An toàn (Consistency & Safety):** Đảm bảo các quy tắc có cấu trúc rõ ràng, được xác thực và an toàn khi thực thi.
- **Tính Giải thích được (Interpretability):** Mọi quy tắc phải có khả năng được "dịch" sang dạng văn bản mà con người có thể đọc và hiểu.
- **Khả năng Lưu trữ và Vận chuyển (Portability):** Các quy tắc phải có thể được lưu trữ bền vững trong cơ sở dữ liệu và dễ dàng vận chuyển giữa các service.
- **Sẵn sàng cho Tương lai (Future-Proof):** Kiến trúc phải đủ linh hoạt để làm nền tảng cho module **Evolution ( `evo-worker` )** trong tương lai mà không cần tái thiết kế lớn.

### 2. Kiến trúc Ba Lớp (Three-Layer Architecture)

Để đáp ứng các mục tiêu trên, kiến trúc được xây dựng dựa trên ba lớp biểu diễn riêng biệt, mỗi lớp phục vụ một mục đích khác nhau. Việc chuyển đổi giữa các lớp này được thực hiện bởi các hàm "parser" và "compiler" chuyên dụng.

1. **Lớp Biểu diễn JSON (JSON Representation Layer):**
    - **Mục đích: Lưu trữ & Giao tiếp (Storage & Communication).** Đây là định dạng "trên dây" (on-the-wire) và "trong cơ sở dữ liệu" (in-database).
    - **Đặc điểm:** Dựa trên cấu trúc cây JSON lồng nhau, độc lập với ngôn ngữ, dễ dàng tuần tự hóa và xác thực. Đây là **Nguồn Chân lý Duy nhất (Single Source of Truth)** cho định nghĩa của một quy tắc.

2. **Lớp Biểu diễn Trong Bộ nhớ (In-Memory Representation Layer):**
    - **Mục đích: Thực thi (Execution).** Khi một quy tắc cần được áp dụng, phiên bản JSON của nó sẽ được phân tích cú pháp (parse) thành một cây các đối tượng Python trong bộ nhớ.
    - **Đặc điểm:** Tối ưu hóa cho hiệu năng. Mỗi node trong cây là một đối tượng Python ( `OperatorNode` , `VariableNode` , `ConstantNode` ) có một phương thức `evaluate()` , cho phép thực thi quy tắc một cách đệ quy và hiệu quả.
3. **Lớp Biểu diễn Văn bản (Plain-text Representation Layer):**
    - **Mục đích: Hiển thị & Giải thích (Display & Explanation).** Dành cho con người đọc, phục vụ cho module `Explainer` và mục đích gỡ lỗi (debug).
    - **Đặc điểm:** Một chuỗi văn bản mô tả logic của quy tắc một cách dễ hiểu, được tạo ra bằng cách duyệt qua cây In-memory.

### 3. Cấu trúc Cây Biểu thức (Symbolic Expression Tree)

Trái tim của kiến trúc này là việc sử dụng **Cây Biểu thức (Expression Tree)**, một cấu trúc dữ liệu tự nhiên để biểu diễn các công thức toán học và logic. Một quy tắc sẽ được thực thi để trả về một **giá trị số liên tục** (ví dụ: trong khoảng [-1.0, 1.0]), đại diện cho sức mạnh và hướng của một tín hiệu (ví dụ: -1.0 = Bán rất mạnh, +1.0 = Mua rất mạnh).

Một cây bao gồm ba loại node cơ bản:

#### a. `ConstantNode` (Node Hằng số):

- **Mô tả:** Đại diện cho một giá trị số không đổi.
- **Ví dụ JSON:**

```
{ "type": "constant", "value": 0.75 }
```

#### b. `VariableNode` (Node Biến - "Biến Thông minh"):

- **Mô tả:** Đại diện cho một điểm dữ liệu được trích xuất từ `QuickCheckReport` . Node này "thông minh" ở chỗ nó chứa cả thông tin về cách lấy và **mã hóa (encode)** dữ liệu gốc thành một giá trị số.
- **Ví dụ JSON (Biến hạng mục - Categorical):**

```
{
  "type": "variable",
  "path": "technical.daily.trend.ma_direction",
  "data_type": "categorical",
  "encoding": {
    "method": "mapping",
    "map": { "uptrend": 1.0, "downtrend": -1.0, "undefined": 0.0 },
```

```
      "default": 0.0
    }
  }
```

- **Ví dụ JSON (Biến số - Numerical):**

```
{
  "type": "variable",
  "path": "technical.daily.key_indicators.rsi_14",
  "data_type": "numerical",
  "encoding": {
    "method": "normalization",
    "range": [0, 100],
    "target_range": [-1, 1]
  }
}
```

#### c. `OperatorNode` (Node Toán tử):

- **Mô tả:** Đại diện cho một hàm hoặc một phép toán nhận một hoặc nhiều node con làm đối số.
- **Ví dụ JSON (Phép cộng):**

```
{
  "type": "operator",
  "value": "+",
  "children": [
    { "type": "constant", "value": 0.5 },
    { "type": "variable", "path": "news.sentiment.score", ... }
  ]
}
```

### 4. Vị trí trong Hệ thống ( `sharedlib` )

Toàn bộ logic để định nghĩa, phân tích cú pháp, thực thi và giải thích các cấu trúc cây này sẽ được đặt trong thư viện chung ( `itapia_common.rules` hoặc tương tự). Điều này đảm bảo cả hai service:

- `ai-service-quick` : Có thể tải các quy tắc JSON từ DB, chuyển thành cây In-memory và **thực thi** chúng để ra quyết định.
- `evo-worker` : Có thể đọc, **thao tác (mutate/crossover)**, và ghi các quy tắc JSON.

### 5. Luồng Hoạt động cho Quy tắc Thủ công (Manual Rules)

Giai đoạn hiện tại tập trung vào việc cho phép `Decision Maker` sử dụng các quy tắc được định nghĩa thủ công.

1. **Định nghĩa Quy tắc:** Một nhà phân tích (hoặc chính bạn) sẽ viết một quy tắc dưới dạng **JSON** và lưu nó vào cơ sở dữ liệu.
2. **Tải Quy tắc:** Khi `Decision Maker` cần ra quyết định, nó sẽ truy vấn DB để lấy chuỗi JSON của quy tắc cần thiết.
3. **Phân tích cú pháp (Parsing):** `Decision Maker` sử dụng một hàm parser từ `sharedlib` để chuyển chuỗi JSON thành một cây các đối tượng `RuleTree` trong bộ nhớ.
4. **Thực thi (Execution):**
   - `Decision Maker` gọi phương thức `rule_tree.evaluate(quick_check_report)` .
   - Cây sẽ tự đệ quy để tính toán. Các `VariableNode` sẽ trích xuất và mã hóa dữ liệu từ báo cáo, các `OperatorNode` sẽ thực hiện phép toán.
   - Kết quả cuối cùng là một điểm số (ví dụ: `0.85` ).
5. **Ánh xạ sang Quyết định:** `Decision Maker` nhận điểm số và áp dụng một logic ánh xạ cuối cùng để chuyển nó thành một quyết định có thể hành động.
   - `if score > 0.7: return 'Strong Buy'`
   - `if score < -0.7: return 'Strong Sell'`
   - ...

Kiến trúc này cung cấp một nền tảng cực kỳ mạnh mẽ, có cấu trúc và sẵn sàng cho tương lai, đồng thời giải quyết được ngay lập tức nhu cầu thực thi các quy tắc thủ công một cách an toàn và minh bạch.