



BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC HOA SEN  
KHOA KHOA HỌC VÀ CÔNG NGHỆ

# KHOÁ LUẬN TỐT NGHIỆP

*Tên đề tài:*

## XÂY DỰNG KIẾN TRÚC CÔNG THÔNG TIN TÌM VIỆC

Giảng viên hướng dẫn : TS. Trần Vũ Bình  
Nhóm sinh viên thực hiện : Lê Trung Hiếu (09015L)  
Phùng Chí Nguyên (09023L)  
Lê Dương Công Phúc (09025L)  
Lớp : QL092L

Tháng 12 năm 2011

< Phiếu giao đê tài >

## TRÍCH YẾU

Hiện tại, cùng với sự phát triển bùng nổ Internet trên thế giới cũng như tại Việt Nam, Internet đã cung cấp cho chúng ta khối lượng thông tin và dịch vụ khổng lồ. Bên cạnh đó, khối lượng thông tin ngày càng lớn phát sinh nhu cầu so khớp thông tin ngày càng nhiều nhưng vẫn phải đảm bảo hiệu năng và độ tin cậy của hệ thống. Các website so khớp thông tin trên Internet hiện nay rất nhiều nhưng hầu như chưa có một kiến trúc chung, cùng với thực trạng các kiến trúc phần mềm giải quyết bài toán nghiệp vụ hiện nay cũng ít được phổ biến. Với mục tiêu xây dựng một kiến trúc phần mềm để giải quyết nhu cầu nói trên, nhóm chúng tôi đã chọn và tham gia thực hiện khoá luận lần này.

Nhóm chúng tôi đã xây dựng thành công nền tảng kiến trúc nghiệp vụ tổ chức và so khớp thông tin một cách hiệu quả, đồng thời áp dụng vào lĩnh vực cụ thể đang rất được quan tâm hiện nay đó là tìm việc trực tuyến. Đây là cơ hội tốt để nhóm có thể vận dụng những kiến thức đã được học áp dụng vào thực tế, Và quan trọng hơn, với những kiến thức đã được học ở trường cũng như tích lũy trong quá trình thực hiện đề tài, kèm theo đó là sự nỗ lực của các thành viên trong nhóm, chúng tôi đã hoàn thành đề tài đúng thời hạn và đúng yêu cầu ban đầu đề ra. Chúng tôi hy vọng kiến trúc này sẽ được tiếp tục phát triển hơn nữa và được ứng dụng, triển khai rộng rãi trên các hệ thống cần tổ chức và so khớp thông tin.

## LỜI CẢM ƠN

Trước tiên, nhóm chúng tôi xin chân thành cảm ơn TS. Trần Vũ Bình đã tận tình hướng dẫn, góp ý và định hướng nhóm chúng tôi trong quá trình thực hiện khoá luận tốt nghiệp này. Khoá luận đã được hoàn thành cung chính là nhờ sự nhắc nhở, đôn đốc và sự giúp đỡ nhiệt tình của thầy.

Xin chân thành cảm ơn các thầy cô Khoa Khoa Học & Công Nghệ đã nhiệt tình giúp đỡ chúng tôi trong suốt thời gian học tập tại trường. Chính thầy cô đã xây dựng cho chúng tôi kiến thức nền tảng và những kiến thức chuyên môn để nhóm hoàn thành khoá luận này và công việc sau này.

Nhóm chúng tôi cũng xin cảm ơn gia đình và bạn bè đã luôn bên cạnh, cổ vũ và động viên chúng tôi những lúc khó khăn để có thể hoàn thành tốt khoá luận này.

Trân trọng.

Nhóm thực hiện khoá luận.

# MỤC LỤC

TRÍCH YẾU .....	i
LỜI CẢM ƠN .....	ii
MỤC LỤC.....	iii
DANH MỤC HÌNH ẢNH .....	vii
DANH MỤC BẢNG .....	ix
DANH MỤC BIỂU ĐỒ.....	x
THUẬT NGỮ .....	xi
NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN .....	xiii
1 Nhập đẽ .....	1
2 Giới thiệu bài toán kiến trúc tổ chức, lưu trữ và so khớp thông tin.....	2
2.1 Bài toán 1: Tổ chức thông tin và sự linh hoạt của hệ thống so khớp .....	4
2.2 Bài toán 2: So sánh mức độ tương quan giữa các thông tin với nhau.....	5
2.3 Bài toán 3: Sử dụng kiến trúc tổ chức thông tin linh hoạt hỗ trợ ra quyết định	5
2.4 Kết quả mong muôn .....	6
3 Cơ sở và nền tảng xây dựng kiến trúc .....	7
3.1 Xu hướng lưu trữ và tìm kiếm thông tin.....	7
3.2 Nhu cầu so khớp thông tin ngày càng nhiều.....	9
3.3 Lưu trữ thông tin dưới dạng cây là tiền đề xây dựng kiến trúc .....	10
3.4 Khi nào thông tin có thể so khớp được với nhau .....	11
3.5 Tính khả thi của một kiến trúc chung cho việc lưu trữ và so khớp thông tin.	12
3.6 Lý thuyết và phương pháp xây dựng kiến trúc tổ chức thông tin linh hoạt và so khớp thông tin .....	13
3.6.1 Tag .....	14

3.6.2	Taxonomy.....	15
3.6.3	Lý thuyết cơ sở để so khớp pattern thông tin .....	20
3.6.4	Cây quyết định.....	22
3.6.5	Các khái niệm cơ bản về kiến trúc phần mềm .....	40
3.6.6	4+1 Architecture View Model.....	43
3.6.7	Các đặc tính chất lượng của kiến trúc phần mềm.....	45
3.6.8	Một số lý thuyết cơ bản OOP Design .....	46
3.6.9	Design Patterns .....	49
3.7	Các vấn đề cần giải quyết khi thực hiện kiến trúc lưu trữ và so khớp thông tin	
	52	
3.7.1	Nắm bắt xu hướng lưu trữ thông tin .....	52
3.7.2	Độ tương quan giữa các tag.....	52
3.7.3	Hệ thống so khớp linh hoạt .....	53
3.7.4	Gợi ý người sử dụng đăng thông tin nhu cầu và “thông tin đáp ứng nhu cầu” bằng hệ thống so khớp.....	53
3.7.5	Gợi ý người sử dụng đăng thông tin nhu cầu và “thông tin đáp ứng nhu cầu” bằng cây quyết định.....	54
3.7.6	Kiến trúc dễ dàng triển khai, mở rộng .....	54
3.7.7	Kiến trúc đảm bảo hiệu năng khi áp dụng cây quyết định và hệ thống so khớp	55
4	Giải quyết bài toán kiến trúc .....	55
4.1	Giải pháp tổ chức thông tin linh hoạt dưới dạng Tag kết hợp với Taxonomy	55
4.2	Giải pháp so khớp thông tin .....	60
4.3	So khớp thông tin kết hợp mức độ tương quan giữa các tag .....	61
4.4	Áp dụng cây quyết định vào bài toán .....	64
4.5	Áp dụng lý thuyết kiến trúc phần mềm vào bài toán .....	68

---

5	Áp dụng giải pháp vào bài toán Job Zoom.....	69
5.1	Thực trạng các website tuyển dụng hiện nay.....	69
5.2	Những vấn đề Job Zoom cần giải quyết .....	69
5.2.1	Vấn đề 1: Hỗ trợ người dùng viết CV theo ngành nghề .....	69
5.2.2	Vấn đề 2: Hỗ trợ nhà tuyển dụng trong việc đăng tải yêu cầu công việc một cách chi tiết và có trọng số theo ngành nghề .....	70
5.2.3	Vấn đề 3: So khớp hồ sơ ứng viên với yêu cầu tuyển dụng .....	70
5.3	Kết quả mong muốn .....	71
6	Những vấn đề cần giải quyết trong JobZoom framework.....	71
6.1	Khái quát phương pháp giải quyết vấn đề .....	71
6.2	Ứng dụng cấu trúc lưu trữ dưới dạng tag kết hợp taxonomy .....	72
6.3	Ứng dụng hệ thống so khớp kết hợp mức độ tương quan .....	72
6.4	Ứng dụng cây quyết định.....	73
6.5	Ứng dụng kiến trúc dễ dàng triển khai và mở rộng .....	74
6.6	Ứng dụng kiến trúc đảm bảo hiệu năng khi áp dụng cây quyết định và hệ thống so khớp.....	74
7	Giải pháp kiến trúc công thông tin tìm việc JobZoom.....	75
7.1	Điều kiện ra đời .....	75
7.2	Kiến trúc JobZoom framework .....	76
7.2.1	Kiến trúc tổng quan framework.....	76
7.2.2	Giải pháp tổ chức và lưu trữ thông tin linh hoạt ứng dụng trong JobZoom framework .....	91
7.2.3	Taxonomy - giải pháp ứng dụng cấu trúc phân loại và lưu trữ thông tin ứng dụng vào JobZoom framework .....	95
7.2.4	Matching tool - Giải pháp so sánh độ tương quan giữa các thông tin được tổ chức theo cấu trúc cây phân cấp. ....	98

---

7.2.5 Data Mining engine - giải pháp ứng dụng Decision Tree vào kiến trúc JobZoom framework .....	101
7.3 So sánh Job Zoom với các website tìm việc hiện tại .....	112
8 Đánh giá và hướng phát triển.....	113
8.1 Những điểm làm được .....	113
8.2 Những điểm hạn chế.....	114
8.3 Hướng phát triển.....	115
8.3.1 Xác thực độ tin cậy của CV và yêu cầu tuyển dụng.....	115
8.3.2 Phát triển Semantic web .....	115
8.3.3 Đánh trọng số cho các thuộc tính dựa vào độ sâu của taxonomy .....	115
8.3.4 Phân tán dữ liệu .....	115
8.3.5 Công cụ hỗ trợ lựa chọn ứng viên .....	115
8.3.6 Thu thập thông tin việc làm tự động.....	115
8.3.7 Áp dụng quy trình tuyển dụng vào hệ thống.....	116
8.3.8 Phát triển kiến trúc phân mềm theo hướng dịch vụ.....	116
TÀI LIỆU THAM KHẢO .....	xiv

## DANH MỤC HÌNH ẢNH

Hình 1. Tổ chức thông tin dạng cấu trúc tương tự nhau có thể so khớp .....	11
Hình 2. Ví dụ về một taxonomy có cấu trúc đơn giản .....	17
Hình 3. Ví dụ về flat taxonomy.....	18
Hình 4. Ví dụ về hierarchical taxonomy.....	18
Hình 5. Ví dụ về faceted taxonomy.....	19
Hình 6. Ví dụ về network taxonomy .....	20
Hình 7. Ví dụ về cây quyết định .....	23
Hình 8. Phân chia bảng D thành những tập con $S_i$ .....	28
Hình 9. Phân chia $S_i$ dựa vào phân lớp.....	30
Hình 10. Độ lợi thông tin thuộc tính Travel cost/Km .....	31
Hình 11. Độ lợi thông tin các thuộc tính còn lại trong Bảng 1 .....	33
Hình 12. Node gốc của cây quyết định sau lần lặp đầu tiên.....	34
Hình 13. Bảng D được phân chia sau lần lặp đầu tiên .....	34
Hình 14. Cây quyết định sau lần lặp đầu tiên .....	35
Hình 15. Dữ liệu cho lần phân lớp thứ 2 .....	35
Hình 16. Tính Impurity degree cho lần phân lớp thứ 2.....	36
Hình 17. Tính Impurity degree các thuộc tính cho lần phân lớp thứ 2 .....	36
Hình 18. Bảng dữ liệu sau khi chia theo phân lớp Gender.....	37
Hình 19. Cây quyết định sau lần phân lớp thứ 2.....	37
Hình 20. Cây quyết định đầy đủ sau 3 lần phân lớp .....	38
Hình 21. Các khái niệm cơ bản về kiến trúc phần mềm.....	42
Hình 22. Mô hình 4+1 View Model.....	43
Hình 23. Weak association .....	46
Hình 24. Strong association .....	46
Hình 25. Aggregation relationship .....	47
Hình 26. Composition relationship .....	47
Hình 27. Mô hình tổ chức thông tin theo tag kết hợp taxonomy .....	57
Hình 28. Ví dụ tổ chức thông tin có cấu trúc linh hoạt .....	58

---

Hình 29. Mô hình giải pháp so khớp thông tin .....	60
Hình 30. Class diagram mức độ tương quan giữa các tag theo góc nhìn .....	62
Hình 31. Ví dụ về lưu trữ thông tin thông qua taxonomy .....	65
Hình 32. Mô hình kiến trúc tổng quan JobZoom framework khi ứng dụng vào thực tế .....	76
Hình 33. Sơ đồ use case JobZoom framework .....	78
Hình 34. Mô hình kiến trúc JobZoom framework .....	82
Hình 35. Kiến trúc phần mềm dưới góc độ lập trình viên .....	83
Hình 36. Quy trình hoạt động của JobZoom framework .....	84
Hình 37. Kiến trúc dưới góc độ triển khai hệ thống vật lý .....	86
Hình 38. Giải quyết bài toán cải thiện hiệu năng xử lý của JobZoom framework .....	90
Hình 39. Mô hình tổ chức thông tin của JobZoom framework .....	91
Hình 40. Tổ chức thông tin yêu cầu một công việc .....	92
Hình 41. Cấu trúc TagAttribute .....	93
Hình 42. Giải pháp mapping dữ liệu .....	94
Hình 43. Mapping dữ liệu khi sử dụng JobZoom framework .....	94
Hình 44. Tổ chức thông tin phân cấp .....	95
Hình 45. Composite pattern .....	96
Hình 46. Class diagram mô tả tổ chức cây phân cấp .....	96
Hình 47. Các thuộc tính phân lớp đảm bảo khả năng so khớp thông tin .....	98
Hình 48. Mô hình giải pháp so sánh độ tương quan giữa các thông tin trong JobZoom .....	99
Hình 49. Dữ liệu đầu vào và đầu ra Decision Tree Engine của JobZoom framework .....	102
Hình 50. Các bước xây dựng cây quyết định trên JobZoom framework .....	103
Hình 51. Cấu trúc bảng DecisionTreeNode và DecisionTreeNodeDistribution .....	103
Hình 52. Cấu trúc dữ liệu cây quyết định #1 .....	104
Hình 53. Cấu trúc cây quyết định khi sử dụng MS Analysis Services API .....	105
Hình 54. Dữ liệu nhập xuất khi sử dụng MS Analysis Services API .....	105
Hình 55. Dữ liệu đầu vào và đầu ra của Decision Tree Engine .....	108
Hình 56. Tiến trình xây dựng cây quyết định của kiến trúc .....	109

---

## DANH MỤC BẢNG

Bảng 1. Ví dụ bảng dữ liệu lựa chọn phương tiện di chuyển .....	22
Bảng 2. Ví dụ về bảng dữ liệu cần dự đoán phương tiện di chuyển .....	24
Bảng 3. Bảng dữ liệu kết quả dự đoán phương tiện di chuyển.....	24
Bảng 4. Lợi ích khi chia bảng D theo thuộc tính “Travel cost/km” .....	32
Bảng 5. Kết quả độ lợi thông tin sau khi phân chia bảng D theo từng thuộc tính.....	34
Bảng 6. Bảng dữ liệu cho lần phân lớp thứ 3 .....	38
Bảng 7. Dữ liệu ví dụ trước khi xử lý bằng kỹ thuật Pivot Transformation .....	59
Bảng 8. Dữ liệu Bảng 7 sau khi xử lý bằng kỹ thuật Pivot Transformation .....	59
Bảng 9. Các thành phần của Microsoft Business Intelligence.....	67
Bảng 10. Bảng mô tả tác nhân trong JobZoom framework.....	78
Bảng 11. Bảng mô tả use case JobZoom framwork.....	82
Bảng 12. Bảng dữ liệu các thông tin tuyển dụng trước khi Pivot trên cột Tag .....	102
Bảng 13. Bảng kết quả sau khi Pivot Transformation.....	102
Bảng 14. Bảng mô tả dữ liệu đầu vào khi sử dụng MS Analysis Services .....	106
Bảng 15. Bảng mô tả dữ liệu đầu ra khi sử dụng MS Analysis Services.....	108
Bảng 16. So sánh JobZoom framework với các website hiện tại .....	113

## DANH MỤC BIỂU ĐỒ

Biểu đồ 1. Mối liên hệ giữa giá trị Entropy cực đại và số lượng phân lớp .....	26
Biểu đồ 2. Mối liên hệ giữa giá trị Gini index cực đại và số lượng phân lớp .....	27

# THUẬT NGỮ

STT	Thuật ngữ	Giải thích
1	Actor	Đối tượng sử dụng hệ thống
2	Attribute	Thuộc tính
3	Blog	Tương tự một tập san cá nhân trực tuyến, một quyền nhật ký dựa trên nền web hay một bản tin trực tuyến nhằm thông báo những sự kiện xảy ra hàng ngày về một vấn đề gì đó.
4	Class	Lớp đối tượng trong lập trình
5	CV	Hay còn gọi là Resume, tương tự sơ yếu lý lịch
6	Database	Cơ sở dữ liệu
7	Email	Thư điện tử
8	Forum	Diễn đàn
9	Framework	Kiến trúc
10	Kiểu dữ liệu Binary	Kiểu dữ liệu có giá trị 0 hoặc 1
11	Kiểu dữ liệu Continuos	Kiểu dữ liệu có giá trị liên tục
12	Kiểu dữ liệu Nominal (biến tên)	Một giá trị thuộc kiểu dữ liệu này sẽ xác định một thực thể từ một thực thể khác, những giá trị này được phân loại để tạo thành các nhóm giá trị
13	Kiểu dữ liệu Ordinal (biến thứ tự)	Một giá trị thuộc kiểu dữ liệu này xác định vị trí của một thực thể so với các thực thể khác như thực thể được đặt ở vị trí thứ nhất, thứ hai hay thứ ba ...
14	Pivot table	Là bảng tổng hợp, phân tích và xử lý dữ liệu từ một danh sách hay một bảng
15	Prefix	Tiền tố

<b>16</b>	Resume	Sơ yếu lý lịch
<b>17</b>	Semantic web	Web ngữ nghĩa
<b>18</b>	Series of rules	Tập các luật
<b>19</b>	Server	Máy chủ
<b>20</b>	Social Network	Mạng xã hội
<b>21</b>	Tag	Thẻ
<b>22</b>	Taxonomy	Hình thái tổ chức, lưu trữ thông tin
<b>23</b>	Training set	Tập dữ liệu đào tạo

# NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

TP. Hồ Chí Minh, ngày tháng 12 năm 2011

**GIẢNG VIÊN HƯỚNG DẪN**

**Trần Vũ Bình**

## 1 Nhập đề

Kiến trúc phần mềm đóng vai trò rất quan trọng trong ngành công nghiệp phần mềm hiện nay. Theo định nghĩa từ The Open Group Architecture Framework (TOGAF), kiến trúc phần mềm có thể chia ra theo các loại: kiến trúc nghiệp vụ (business architecture), kiến trúc ứng dụng (application architecture), kiến trúc dữ liệu (data architecture), kiến trúc công nghệ (technical architecture). TOGAF là kiến trúc mở mà các tập đoàn phần mềm trên thế giới đang áp dụng như: Capgemini, Hewlett-Packard (HP), IBM, Oracle, SAP.. Trong các loại, kiến trúc nghiệp vụ (business application) đa dạng, phong phú hơn tất cả vì lý do đặc thù đa dạng của các nghiệp vụ (business). Mặt khác, hiện nay các ứng dụng phần mềm, website có chức năng nghiệp vụ như đăng tải thông tin, tìm kiếm và so khớp và được thể hiện qua nhiều loại khác nhau (ví dụ như website tìm việc, tìm bạn, tìm mặt hàng...) rất phổ biến. Nhưng hiện nay, dùng như rất ít phổ biến một kiến trúc chung về nghiệp vụ hỗ trợ cho việc tổ chức và so khớp thông tin linh hoạt có thể áp dụng cho nhiều ứng dụng, website có nghiệp vụ nói trên. Từ nhu cầu thông tin nhiều, đa dạng, cần so khớp nhiều và nhanh, nhóm chúng tôi muốn xây dựng một kiến trúc phần mềm giúp người dùng đăng tải và so khớp thông tin một cách dễ dàng, nhanh chóng và hiệu quả hơn.

Trong khuôn khổ thực hiện đề tài, chúng tôi tiến hành nghiên cứu đánh giá các website tìm kiếm và so khớp thông tin hiện nay, từ đó làm tiền đề xây dựng giải pháp kiến trúc nghiệp vụ nói trên, đồng thời áp dụng giải pháp kiến trúc này vào lĩnh vực tuyển dụng. Chúng tôi tạm gọi kiến trúc được áp dụng vào lĩnh vực tuyển dụng này là Job Zoom framework. Từ việc áp dụng vào một lĩnh vực cụ thể đó, chúng tôi sẽ tiến hành đánh giá giải pháp của chúng tôi đưa ra liệu có hợp lý và giải quyết hiệu quả các vấn đề đưa ra hay không.

Phương pháp thực hiện của chúng tôi là: đầu tiên tìm hiểu và phân tích nhu cầu so khớp thông tin của người dùng thông qua các hệ thống website có các chức năng nghiệp vụ về khớp thông tin, đặc biệt các website tìm kiếm việc làm trong và ngoài nước. Với cơ sở phân tích trên, chúng tôi xác định các bài toán chính cần giải quyết. Bám sát các bài toán này, chúng tôi tiến hành tìm hiểu các cơ sở lý thuyết khoa học máy tính, công nghệ phần mềm liên quan. Với các cơ sở lý thuyết cụ thể trong đề này:

lý thuyết về tổ chức thông tin, khai thác dữ liệu và các lý thuyết về xây dựng kiến trúc hệ thống phần mềm, nhóm chúng tôi tiến hành nghiên cứu, đánh giá nhằm đưa ra các giải pháp hiệu quả giải quyết các bài toán đã đề.

Mục tiêu của nhóm chúng tôi khi thực hiện đề tài “**xây dựng kiến trúc hệ thống công thông tin tìm việc**” là cung cấp giải pháp kiến trúc nghiệp vụ linh hoạt giúp người dùng đăng tải và so khớp thông tin, cung cấp một framework đảm bảo tính linh hoạt khi áp dụng trong nhiều hệ thống ngành nghề khác nhau, hỗ trợ người tìm việc hoàn thiện CV tìm việc của mình và hỗ trợ nhà tuyển dụng trong việc đánh giá ứng viên. Đồng thời, doanh nghiệp cung cấp website tìm việc có thể dễ dàng xây dựng hoặc hoàn thiện website của mình dựa trên Job Zoom framework.

Với những ưu điểm của kiến trúc nghiệp vụ đăng tải và so khớp thông tin nói chung và giải pháp Job Zoom framework nói riêng, chúng tôi hy vọng đề tài sẽ được ứng dụng và triển khai rộng rãi trong các website so khớp, đánh giá cũng như các website tìm việc trực tuyến, góp phần nâng cao hiệu quả và tiện ích cho người dùng đưa ra những quyết định hiệu quả và kịp thời.

## 2 Giới thiệu bài toán kiến trúc tổ chức, lưu trữ và so khớp thông tin

Ngày nay, Internet mang lại rất nhiều tiện ích hữu dụng cho người sử dụng, một trong các tiện ích phổ thông của Internet là hệ thống thư điện tử (email), dịch vụ tìm kiếm (search engine), website, mạng xã hội (social network) và nhiều dịch vụ tiện ích khác. Những hệ thống này cung cấp một khối lượng thông tin và dịch vụ khổng lồ trên Internet. Người ta đã tổ chức quản lý những thông tin này như thế nào cho hiệu quả, hãy xem qua những ví dụ sau:

❖ Đã 29 năm trôi qua kể từ ngày 30/8/1982<sup>1</sup>, ngày hệ thống liên lạc bằng thư điện tử (email) ra đời, có thể nói đây là một phương tiện thông tin rất nhanh, vận tốc truyền thư điện tử chỉ vài giây đến vài phút và chi phí rất nhỏ không đáng kể so với gửi qua đường bưu điện. Việc gửi email cho một nhóm thật sự trở nên khó khăn khi phải nhập danh sách người nhận gây mất thời gian. Để giải quyết vấn đề này, giải pháp tổ chức

<sup>1</sup> Nguồn: <http://www.pcworld.com.vn/articles/tieu-dung/song-cong-nghe/2011/08/1227918/email-hanh-trinh-29-nam-lich-su/>

thông tin địa chỉ email theo các nhóm (contact group) giúp việc gửi email cho một nhóm trở nên dễ dàng hơn. Bên cạnh đó, kỹ thuật tổ chức email theo các nhóm thư (message grouping) cũng giúp cho việc phân loại và quản lý dễ dàng số lượng số lớn các thư có trong hộp thư.

❖ Ví dụ thứ hai về tổ chức thông tin thành các nhóm có cấu trúc, diễn đàn (forum) ra đời nhằm giúp hỗ trợ việc trao đổi thông tin giữa một nhóm người được thực hiện một cách dễ dàng và nhanh chóng hơn, những bài viết cũng được phân chia thành từng thể loại (box) cụ thể và rõ ràng hơn. Tuy nhiên, việc phân loại bài viết trên forum cũng không phải là vấn đề đơn giản, một bài viết trong forum chỉ thuộc về duy nhất một chủ đề, do đó, khi một bài viết thuộc về nhiều chủ đề khác nhau, khi chúng ta đăng bài viết trong chủ đề này, thì khi vào chủ đề kia lại không thấy và ngược lại. Để giải quyết vấn đề này, thẻ (tag) đã ra đời. Một bài viết sẽ được tag những tag liên quan, sự liên kết các tag này góp phần giải quyết cho việc phân loại bài viết trở nên dễ dàng hơn.

❖ Dịch vụ tìm kiếm (search engine), cỗ máy tìm kiếm sẽ thu thập thông tin trên khắp các website trên internet không phân biệt đó là website gì, thông tin gì, cấu trúc thể hiện website đó như thế nào. Tiếp theo, dịch vụ kiểm sẽ tổ chức khói thông tin đã thu thập đó. Chức năng chính của các dịch vụ tìm kiếm là cung cấp khả năng tìm kiếm, so khớp thông tin của người dùng nhập vào và trả về kết quả thông tin (còn gọi là keyword based). **Vậy các dịch vụ tìm kiếm đó đã tổ chức thông tin đó linh hoạt như thế nào và khả năng so khớp thông tin theo từ khóa thế nào?**

❖ Một ví dụ khác cũng về dịch vụ tìm kiếm, cũng nhờ việc tổ chức khói lượng thông tin khổng lồ đã thu thập khắp thế giới internet giúp dịch vụ tiềm kiếm có thể **khai thác khói thông tin đó giúp đưa ra nhiều thông tin hữu ích, mang tính chất quyết định, hoặc một tri thức tri thức**. Ví dụ, qua việc khai thác kho thông tin thu thập khổng lồ, nhà dịch vụ tìm kiếm có thể đưa ra thông tin dự đoán xu hướng mua hàng của của người dân Việt Nam trong năm 2012.

Qua những ví dụ trên, chúng tôi nghĩ về nhu cầu rất cần thiết về một kiến trúc phần mềm chung có thể giúp **tổ chức thông tin linh hoạt từ các nguồn thông tin đa dạng khác nhau, qua đó giúp người sử dụng đăng tải và so khớp thông tin một**

*cách dễ dàng, nhanh chóng, hiệu quả.* Kiến trúc này phải giải quyết các bài toán cụ thể sau:

## 2.1 Bài toán 1: Tổ chức thông tin và sự linh hoạt của hệ thống so khớp

Bạn cần tìm kiếm một sản phẩm để mua - nhà sản xuất cần tìm khách hàng để bán sản phẩm, làm sao để bạn có thể tìm kiếm những sản phẩm thoả mãn nhu cầu và doanh nghiệp cũng đạt được lợi nhuận cao nhất từ việc bán hàng? Bạn cần tìm kiếm một người bạn đã lâu không còn liên lạc – người bạn của bạn đã tham gia một mạng xã hội, làm sao để bạn và người bạn ấy có thể tìm ra nhau? Bạn cần tìm một công việc phù hợp với kỹ năng và trình độ của bạn thân – doanh nghiệp cần tìm những ứng viên đáp ứng đầy đủ các yêu cầu tuyển dụng?... *Qua những ví dụ trên, kết luận việc tìm kiếm so khớp thông tin với nhau, giữa các thông tin “yêu cầu” và thông tin “đáp ứng yêu cầu” có nhu cầu rất cần thiết trên internet ngày nay.*

Trong các phần mềm, thông tin của một đối tượng trong thế giới thực (real object) được mô hình hóa thành các đối tượng, các thực thể trong phần mềm (software object). Có nhiều cách để mô hình hóa (schema mapping) một đối tượng trong thế thực: mỗi quan hệ 1-1 (một đối tượng thực được biểu diễn bằng một đối tượng duy nhất trong phần mềm, mỗi quan hệ tập hợp (một đối tượng thực được biểu diễn bằng cách cấu thành bởi nhiều đối tượng trong phần mềm), quan hệ ké thừa... Không những chỉ tổ chức thông tin trong phần mềm (software object) mà còn phải tổ chức về mặt lưu trữ các đối tượng đó xuống cơ sở dữ liệu thì cũng có nhiều cách để tổ chức. *Tóm lại trong quá trình xây dựng phần mềm, việc tổ chức và lưu trữ thông tin rất đa dạng theo nhiều mô hình khác nhau phụ thuộc vào đặc thù của đối tượng thông tin đó và chiến lược của nhà phát triển phần mềm. Làm thế nào thiết kế một kiến trúc phần mềm dùng chung có thể đáp ứng khả năng tổ chức thông tin một cách linh hoạt.*

Như đã đề cập ở trên, thông tin được mô tả, tổ chức dưới nhiều mô hình khác nhau như trên thì giải quyết cho vấn đề tìm kiếm, so khớp những thông tin “yêu cầu” và thông tin “đáp ứng yêu cầu” cũng gặp khó khăn do các thông tin được tổ chức, cấu trúc phức tạp. *Xuất phát từ những vấn đề thực tế trên, bài toán cần phải đạt được là xây dựng kiến trúc phần mềm dùng chung có khả năng tổ chức thông tin một cách*

*linh hoạt, có áp dụng cho nhiều phần mềm, nhiều nghiệp vụ khác nhau mà vẫn đảm bảo được khả năng so khớp giữa những các thông tin nội tại đó.*

## 2.2 Bài toán 2: So sánh mức độ tương quan giữa các thông tin với nhau

Như đã đề cập ở vấn đề của phần giới thiệu bài toán 1: thông tin người dùng của các phần mềm nghiệp vụ về tìm kiếm, so khớp thông tin “nhu cầu” và thông tin “đáp ứng yêu cầu” được tổ chức đa dạng tùy theo đặc điểm thông tin và chiến lược sản phẩm phần mềm đó. *Kết quả của bài toán 1 là một kiến trúc linh hoạt chung có thể đáp ứng khả năng sau: đầu vào là các thông tin đa dạng được tổ chức theo mô hình riêng, có cấu trúc riêng (structured information), khi vào trong kiến trúc các thông tin đó được tổ chức chung lại (unstructured information) và kết quả đầu ra tạo ra những thông tin mới có cấu trúc linh hoạt (structured information) mà có khả năng so khớp với nhau.*

Vấn đề đang đề cập tới việc so khớp các thông tin với nhau như thế nào: các thông tin yêu cầu và thông tin đáp ứng yêu cầu. Kết quả so khớp sẽ giúp người sử dụng tiếp cận thông tin và đưa ra các quyết định hữu ích.

*Tóm lại, bài toán 2 giải quyết vấn đề so sánh giữa các thông tin nội tại trong kiến trúc phần mềm giúp thông tin “yêu cầu” và “thông tin đáp ứng yêu cầu” có thể so khớp với nhau.*

## 2.3 Bài toán 3: Sử dụng kiến trúc tổ chức thông tin linh hoạt hỗ trợ ra quyết định

*Chúng ta cùng xem xét các vấn đề cần được kiến trúc hỗ trợ ra quyết định sau đây:*

- Trong những thuộc tính mô tả thông tin về nhu cầu và “thông tin đáp ứng nhu cầu” thì những thuộc tính nào có yếu tố quyết định trong việc lựa chọn “thông tin đáp ứng nhu cầu”?

- “Thông tin đáp ứng nhu cầu” cần có thêm những thuộc tính nào để đáp ứng phần lớn những nhu cầu của “người phát sinh nhu cầu”?

- “Người phát sinh nhu cầu” khi muốn đăng thông tin về nhu cầu của mình thì những thuộc tính nào được nhiều người dùng quan tâm nhất khi lựa chọn “thông tin

đáp ứng nhu cầu”, những thuộc tính quan trọng giúp họ có khả năng hình dung các thuộc tính cần có của “thông tin đáp ứng nhu cầu” mà họ đang muốn đăng thông tin để tìm kiếm?

Thông tin về nhu cầu và “thông tin đáp ứng nhu cầu” được mô tả bằng tập các thuộc tính do người dùng tự do định nghĩa và được tổ chức linh hoạt (bài toán 1). Tuy nhiên, sự linh hoạt này cần được sử dụng để tạo ra các tri thức mới, các tri thức tiềm năng trong nguồn dữ liệu đã có nhằm giải quyết ba vấn đề cần hỗ trợ ra quyết định như trên.

## 2.4 Kết quả mong muốn

- ❖ Xây dựng giải pháp kiến trúc phần mềm chung hỗ trợ việc tổ chức và lưu trữ thông tin một cách linh hoạt, có thể áp dụng phần mềm có nghiệp vụ tìm kiếm, so khớp các thông tin “yêu cầu” và thông tin “đáp ứng yêu cầu”.
- ❖ Từ giải pháp kiến trúc tổ chức thông tin linh động trên, đưa ra giải pháp so khớp các thông tin nội tại qua việc đánh giá mức độ tương quan giữa các thông tin đó với nhau bằng điểm số cụ thể. Một yêu cầu khác là giải pháp này có dễ dàng mở rộng, cải thiện sau này.
- ❖ Từ giải pháp kiến trúc tổ chức thông tin linh động trên đưa ra các giải pháp khai thác dữ liệu giúp tạo nên các thông tin hữu ích. Cụ thể trong phạm vi đề tài, đưa ra giải pháp áp dụng lý thuyết cây quyết định (decision tree) giúp gợi ý người sử dụng, cải thiện việc việc so khớp “yêu cầu” và thông tin “đáp ứng yêu cầu”, áp dụng vào các nghiệp vụ thực tế.
- ❖ Áp dụng ba giải pháp trên, chúng tôi mong muốn xây dựng một giải pháp framework cụ thể (JobZoom framework) áp dụng vào các website tìm kiếm việc làm, trong đó cung cấp API lập trình cho các lập trình viên dễ dàng sử dụng, implement website giải quyết bài toán trên một cách nhanh chóng, đồng thời có khả năng mở rộng sau này.
- ❖ Ngoài ra, trong quá trình thiết kế kiến trúc phần mềm các giải pháp trên nói chung, và JobZoom framework nói riêng, chúng tôi cân nhắc, quan tâm và giải quyết một phần bài toán hiệu năng (performance) của hệ thống.

### 3 Cơ sở và nền tảng xây dựng kiến trúc

#### 3.1 Xu hướng lưu trữ và tìm kiếm thông tin

Trong khoa học máy tính, cấu trúc dữ liệu là một cách lưu trữ dữ liệu trong máy tính sao cho nó có thể được sử dụng một cách hiệu quả. Sau khi internet bắt đầu bùng nổ năm 1990, các website đã liên tục phát triển và có bước thay đổi chóng mặt. Một trong những thay đổi không thể không nhắc đến đó là cách lưu trữ thông tin. Những website đầu tiên hầu như chỉ là những đoạn văn bản không có cấu trúc, một email giải quyết nhu cầu trao đổi thư điện tử cá nhân cũng chỉ đơn thuần là văn bản. Do nhu cầu người dùng ngày càng cao và đa dạng hơn, lần lượt mail group đã được ra đời để giải quyết nhu cầu gửi thông tin bằng dạng văn bản cho nhiều đối tượng tránh hiểu nhầm thư được gửi là thư rác, kỹ thuật mail grouping giải quyết phân loại email cho từng người dùng ở một mức độ đơn giản. Mặc dù mail group có thể đáp ứng được nhu cầu truyền tải thông tin cho một nhóm người dùng nhanh chóng hơn nhiều so với email, nhưng các thông tin này hoàn toàn chưa có một cấu trúc chung, hay nói khác hơn là chưa phân loại một cách cụ thể để người dùng dễ dàng phân biệt. Forum ra đời đã giải quyết được bài toán phân loại bài viết cho một nhóm người dùng, thông tin từ dạng không được chia theo nội dung, được lưu trữ theo từng chủ đề hay đề mục cụ thể tương ứng với nội dung của thông tin. Tuy nhiên, khi forum xuất hiện khuyết điểm của mình: một bài viết nếu cùng thuộc về nhiều chủ đề thì việc phân loại thật sự bộc lộ khuyết điểm của nó (ví dụ bài viết về nghiên cứu hóa sinh thực phẩm, nếu để trong chủ đề “Hoá học” thì khi người dùng vào chủ đề “Sinh học” lại không có và ngược lại), mặc dù có thể giải quyết bằng cách đăng tải bài viết vào tất cả những chủ đề liên quan, nhưng mỗi bài này đều khác nhau, việc phân hồi thông tin trong bài viết trong từng chủ đề cũng không có mối quan hệ với nhau. Vì vậy, kỹ thuật tag ra đời cùng với blog, một cách phân loại bài viết hoàn toàn mới, một bài viết có thể được tag những từ khóa liên quan. Việc phân loại thực sự trở nên dễ dàng và linh động hơn: bài viết về nghiên cứu hóa sinh thực phẩm, sẽ được tag “Hoá học” và “Sinh học”, khi vào chủ đề “Hoá học” hay “Sinh học” thì bài viết “Nghiên cứu hóa sinh thực phẩm” đều được hiển thị, điểm quan trọng ở đây, là bài viết trong cả hai chủ đề này đều dẫn đến một bài viết duy nhất đó là “Nghiên cứu hóa sinh thực phẩm”. Tuy nhiên, các tổ chức

thông tin này cũng có khuyết điểm, các tag được người dùng định nghĩa tự do hoàn toàn không có cấu trúc, làm sao biết được mối quan hệ giữa các tag, tag này có mức độ tương quan như thế nào với tag kia. Một ví dụ đơn giản, bài viết giới thiệu về cách nấu món bò nấu đậu, người dùng A có thẻ tag thông tin “đậu, bò, ...”, người dùng B có thẻ tag thông tin “beans, beef, ...”, “beans và đậu” hay “beef và bò” có một độ tương quan nhất định với nhau. Xuất hiện trong vài năm gần đây nhưng thật sự đã trở thành một cơn sốt, đó chính là mạng xã hội, mạng xã hội đã ra đời khắc phục cách tổ chức thông tin theo dạng tag của blog. Để khắc phục được nhược điểm này, mạng xã hội lưu trữ thông tin dưới dạng tag có cấu trúc, việc lưu trữ này thể hiện ở việc lưu thông tin của người dùng dưới dạng tag, có mức độ tương đồng giữa các thuộc tính giúp cho việc so sánh hay cụ thể là tìm bạn trên mạng xã hội trở nên dễ dàng hơn. Bạn có thử bằng cách gõ vào thanh tìm kiếm của facebook một từ khoá bất kỳ, kết quả trả về cho bạn là những tên group, pages, những bạn bè của bạn hay những người bạn có thể quen mà hai bạn chưa dùng chức năng kết bạn của facebook để tạo liên kết.

Điểm chung dễ nhận thấy ở đây là cách lưu trữ thông tin, ban đầu thông tin chỉ là những đoạn văn bản thô hoàn toàn không có cấu trúc, dần dần thông tin này được lưu trữ dưới dạng có cấu trúc ở mức đơn giản nhưng cũng bộc lộ nhiều khuyết điểm trong quá trình phân loại. Để khắc phục điều này, thông tin lại được tổ chức dưới dạng không có cấu trúc đó chính là tag. Tuy nhiên một nhu cầu lại đặt ra là giữa các tag người dùng được tự do định nghĩa ấy, thì tag nào giống nhau hay có mức độ tương quan với nhau, “tag lưu trữ theo dạng có cấu trúc” đã giải quyết được bài toán ấy.

Không chỉ trong khoa học máy tính, mà chính trong quá trình phát triển của xã hội loài người cũng thể hiện điều đó, từ khi con người xuất hiện và sống trong môi trường tự nhiên hoàn toàn không có sự phân chia giai cấp, khi xã hội bước sang giai đoạn chiếm hữu nô lệ giữa người và người lại có sự phân chia giữa chủ tớ và nô lệ. Một khi sự phân chia này không còn phù hợp và bộc lộ nhiều khuyết điểm, con người lại đấu tranh đòi hỏi có sự phân chia công bằng, bình đẳng hơn.

*Tóm lại, xu hướng lưu trữ thông tin được lưu trữ dưới dạng phi cấu trúc – dữ liệu thô và không có sự phân loại sẽ dần dần chuyển sang dạng có cấu trúc, có sự phân loại rõ ràng. Từ dạng có cấu trúc này thông tin lại được trở về dạng phi cấu*

*trúc tuy nhiên ở một mức độ cao hơn, có sự phân loại bằng tag nhưng chưa thể hiện mức độ tương quan giữa các tag ấy và lưu trữ thông tin dưới dạng có cấu trúc cây đã giúp giải quyết các khuyết điểm trên, giúp cho việc lưu trữ thông tin trở nên linh hoạt cũng như giúp cho việc phân loại và so khớp giữa các đối tượng trở nên dễ dàng hơn. Việc thay đổi trong cách lưu trữ thông tin cũng giải quyết được nhu cầu tìm kiếm nhanh và nhiều của người dùng.*

### 3.2 Nhu cầu so khớp thông tin ngày càng nhiều

Theo số liệu thống kê của tổ chức Internet Usage World Stats<sup>2</sup>, tính đến nay có hơn 2,1 tỉ người trên thế giới sử dụng Internet – chiếm hơn 30% dân số thế giới, chỉ tính riêng mạng xã hội facebook số lượng người sử dụng đã lên đến 710 triệu người chiếm gần 34% số lượng người sử dụng internet, tốc độ tăng trưởng mỗi năm khoảng 10,3%. Số liệu trên cho thấy nhu cầu tìm bạn và chia sẻ trên mạng xã hội ngày càng tăng. Làm sao mạng xã hội có thể gợi ý cho người dùng những bạn bè của họ, những quảng cáo trên mạng xã hội facebook cũng được hiển thị dựa theo sở thích và sự quan tâm của người dùng đối với một lĩnh vực cụ thể mà người dùng khai báo trong chương trình. Để làm được điều này, mạng xã hội đã tiến hành so khớp những thông tin giữa người dùng với nhau và so khớp giữa nhu cầu quảng cáo và các lĩnh vực người dùng đang quan tâm.

Bên cạnh đó, nhu cầu mua sắm trực tuyến và sự bùng nổ của các website thương mại điện tử trên thế giới đã phát sinh nhu cầu tìm kiếm, so khớp thông tin sản phẩm và tìm kiếm thông tin thị trường ngày càng nhiều, khách hàng cần tìm sản phẩm ưng ý để mua và nhà sản xuất cũng cần tìm khách hàng để bán sản phẩm của mình.

Các website việc làm cũng ngày càng phát triển, 30% số người dùng Internet đã tham khảo các website tuyển dụng trực tuyến để tìm kiếm thông tin về việc làm cho mình<sup>3</sup>. Việc đăng tải thông tin việc làm và thông tin ứng viên hiện đang được các website này lưu trữ dưới dạng văn bản, cấu trúc không linh hoạt. Cách tổ chức thông tin này bộc lộ nhiều khuyết điểm trong quá trình so khớp thông tin giữa hồ sơ xin việc và yêu cầu tuyển dụng. Hiện tại các website này chỉ dừng lại ở mức đăng tải thông tin

<sup>2</sup> <http://www.internetworkworldstats.com/stats.htm>

<sup>3</sup> Theo Mintel International Group Ltd. (tổ chức quốc tế chuyên nghiên cứu về thị trường và người tiêu dùng)

và tìm kiếm việc làm ở mức đơn giản, chủ yếu cung cấp tìm kiếm thông qua kỹ thuật tìm kiếm dựa trên keyword, chưa đưa ra mức độ so khớp giữa CV người xin việc và yêu cầu công việc trong khi nhu cầu so khớp ở các website này là rất nhiều: doanh nghiệp cần được hỗ trợ đánh giá ứng viên, thông tin ứng viên cung cấp thoả mãn bao nhiêu phần trăm nhu cầu tuyển dụng của họ, hay ứng viên cần so khớp thông tin để viết CV được tốt và tạo được ấn tượng với nhà tuyển dụng hơn...

Tìm bạn, tìm sản phẩm, tìm việc,... là những vấn đề mỗi chúng ta rất thường gặp trong thực tế cũng như trong môi trường toàn cầu hoá thông tin như hiện nay. Nhu cầu so sánh và đánh giá mức độ so khớp giữa “nhu cầu” và “thông tin đáp ứng nhu cầu” đang rất bức thiết và thực sự đang rất cần một kiến trúc nghiệp vụ để hỗ trợ tổ chức và so khớp thông tin.

### 3.3 Lưu trữ thông tin dưới dạng cây là tiền đề xây dựng kiến trúc

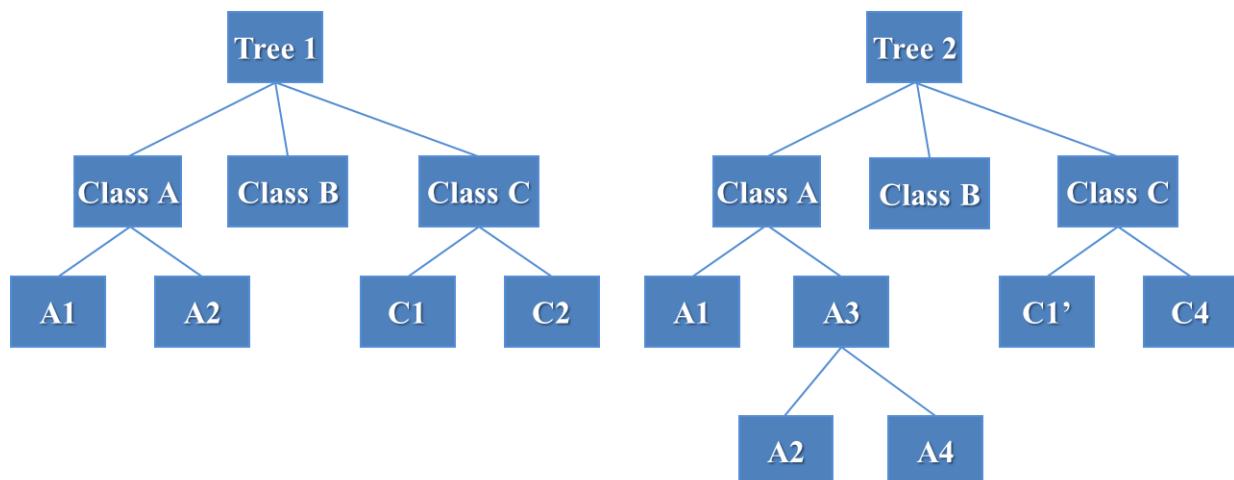
Có nhiều cách lưu trữ thông tin khác nhau: lưu trữ thông tin dạng quan hệ (danh sách liên kết), ngang hàng (dạng mảng), lưu trữ dạng cây và đồ thị. Cây là một cấu trúc dữ liệu có vai trò quan trọng trong việc phân tích và thiết kế các thuật toán. Lưu trữ và biểu diễn dữ liệu kiểu cấu trúc cây có thể thấy trong rất nhiều lĩnh vực của cuộc sống. Ví dụ: một cuốn gia phả lưu trữ thông tin về các thành viên trong một dòng họ, trong đó các thành viên thuộc bậc khác nhau được phân thành các cấp khác nhau trong biểu diễn hình cây của gia phả. Sơ đồ tổ chức của một đơn vị cũng thường được tổ chức thông qua cấu trúc cây. Các đơn vị con nằm ở cấp dưới đơn vị trực tiếp quản lý, các đơn vị ngang hàng nằm cùng cấp. Trong lĩnh vực khoa học máy tính, cách lưu trữ dữ liệu của hệ điều hành cũng áp dụng kiểu lưu trữ cây, các tập tin được lưu trữ trong các cây thư mục, trong đó các thư mục con nằm trong các thư mục cha. Việc lưu trữ thông tin dưới dạng cây cũng có thể dễ dàng phát triển thành mạng thông tin (lưu trữ dạng network), chỉ cần đưa bất kỳ một node của cây thành node gốc, có sự đánh giá mức độ tương quan giữa các tag thông tin với nhau, vô tình cây thông tin ban đầu đã được chuyển sang dạng lưu trữ theo mạng thông tin, làm cho kiến trúc có khả năng phát triển linh hoạt.

Lưu trữ thông tin dạng cây trong lĩnh vực tìm kiếm và so khớp thông tin cũng có nhiều ưu điểm hơn so với các dạng lưu trữ thông tin khác: lưu trữ dạng ngang hàng và

danh sách liên kết rất khó khăn trong việc so khớp dữ liệu, việc tìm kiếm khi lưu trữ theo dạng ngang hàng và danh sách liên kết đều phải dùng phương pháp vét cạn để duyệt gây hao tốn tài nguyên hệ thống, bên cạnh đó, thông tin ban đầu được xây dựng không đủ lớn để có thể tổ chức dưới dạng network. Việc lưu trữ thông tin dưới dạng cấu trúc cây là phù hợp nhất để phát triển hệ thống tìm kiếm và so khớp từ ban đầu. *Tổ chức thông tin theo dạng cây là cách tổ chức thông tin khá đơn giản, đem lại nhiều tiện ích cho việc tra cứu và so khớp thông tin, làm tiền đề cho việc lưu trữ thông tin của kiến trúc.*

### 3.4 Khi nào thông tin có thể so khớp được với nhau

Để so khớp thông tin với nhau đòi hỏi giữa các cây thông tin phải có một cấu trúc tương tự nhau hay giữa các node bất kỳ, có thể xác định được mức độ tương quan giữa chúng [20]. Thông tin ban đầu được chuyển về lưu trữ dạng tag, sau đó, công việc tiếp theo là phải chuyển thông tin về dạng cây – các tag được xây dựng thành cây, được phân loại theo các chủ đề có sẵn để dễ dàng hơn trong việc so khớp.



**Hình 1. Tổ chức thông tin dạng cấu trúc tương tự nhau có thể so khớp**

“Tree 1” và “Tree 2” lưu trữ thông tin về hai đối tượng, cả hai cây này để có cấu trúc phân loại là “Class A”, “Class B” và “Class C”:

- Việc so sánh node A1 của “Tree 1” và “Tree 2” có thể được thực hiện dễ dàng, vì giá trị của hai node này hoàn toàn giống nhau.

- Xét node A2, ở “Tree 1” node A2 là con của phân lớp “Class A” còn ở “Tree 2”, node A2 là con của A3, A3 là con của phân lớp “Class A”, giữa hai node này cũng có thể được so sánh với nhau thông qua mối quan hệ với phân lớp “Class A”

- Xét node C1 và C1’, node A2 và node A3, giữa các cặp node này có mức độ tương quan nhất định với nhau nên có thể so khớp được với nhau.

### 3.5 Tính khả thi của một kiến trúc chung cho việc lưu trữ và so khớp thông tin

Internet chứa hầu như tất cả những thông tin liên quan tới mọi lĩnh vực trong cuộc sống. Nhưng nó rất rộng, rộng đến mức gần như không ai có thể kiểm soát được. Diện mạo của Internet lại thay đổi quá nhanh chóng và mạnh mẽ. Hạt nhân của Internet là Word Wide Web, với số lượng lên tới hàng chục tỉ trang, được lưu trữ trong hàng triệu máy chủ đặt khắp nơi trên toàn thế giới. Có thể ví Internet như một biển dữ liệu khổng lồ, với muôn vàn những viên ngọc quý nằm giữa các hạt sạn. Trong đời sống hàng ngày, *nhu cầu tìm kiếm và so khớp thông tin đóng vai trò vô cùng to lớn, và một trong những vấn đề bức thiết nhất của công nghệ hiện nay là làm sao “đãi cát tìm vàng”, khai thác nguồn tài nguyên này một cách hợp lý, đem lại lợi ích tốt nhất cho con người*. Nói khác hơn, thông tin cần được tổ chức một cách hợp lý và hệ thống so khớp làm việc hiệu quả, hỗ trợ cho việc khai thác nguồn tài nguyên to lớn ấy.

Tìm kiếm và so khớp thông tin trên mạng Internet quả thật là một thách thức lớn lao. Nó không giống như việc bới các hạt đỗ đen nằm lẩn lộn trong thùng gạo, bởi dữ liệu trên mạng Internet do con người đưa vào, chúng cũng có cấu trúc và tổ chức xác định (mặc dù thiếu tính nhất quán), trong khi đó thì các hạt đỗ đen lại nằm rải rác và lộn xộn, không có một vị trí hay quy luật nào. Tuy nhiên, bài toán tìm kiếm và so khớp khó hơn bài toán nhặt đỗ đen rất nhiều. Muốn tìm tất cả các hạt đỗ đen, bạn đơn giản chỉ cần thiết kế một cái sàng hình cầu đủ lớn để có thể đổ cả thùng gạo vào đó, với những chiếc lỗ có kích thước phù hợp sao cho hạt gạo chui lọt còn hạt đỗ đen thì không, và quay đủ số vòng để tất cả các hạt gạo đều có cơ hội bay ra ngoài. Việc tìm kiếm và so khớp thông tin trên internet đòi hỏi thông tin cần được lưu trữ một cách linh hoạt, có cấu trúc giúp cho việc tìm kiếm và so khớp thông tin hiệu quả hơn.

Có tới hàng chục tỉ trang web tràn ngập trên mạng Internet, rất nhiều “thông tin đáp ứng nhu cầu” có thể thoả mãn nhu cầu của người dùng, và vấn đề là làm sao đưa ra những gì ta muốn thu thập sao cho đồng thời thoả mãn hai tiêu chí: Chính xác và nhanh chóng. Hơn thế nữa, người dùng cũng không đủ kiên nhẫn để ngồi duyệt qua tất cả các trang chứa thông tin cần tìm. Trên thực tế, người dùng hiếm khi vào xem quá mười thông tin kết quả, và vì thế, một yêu cầu khó khăn nữa cần giải quyết, đó là: những gì phù hợp nhất phải được đặt lên hàng đầu, nên *hệ thống so khớp cũng đóng vai trò rất quan trọng trong quá trình tìm kiếm thông tin*.

Bên cạnh đó, xu hướng lưu trữ, tìm kiếm và so khớp thông tin ngày càng nhiều nhưng chưa có kiến trúc nghiệp vụ nào giải quyết bài toán này (các kiến trúc hiện nay đi nhiều về giao diện và giao tiếp, kiến trúc giải quyết bài toán nghiệp vụ còn ít). Thông tin có xu hướng lưu trữ dưới *dạng phi cấu trúc* – dữ liệu thô và chưa có sự phân loại chuyển sang *dạng có cấu trúc*, có sự phân loại rõ ràng, từ *dạng có cấu trúc* này thông tin lại được trở về *dạng phi cấu trúc* tuy nhiên ở một mức độ cao hơn, có sự phân loại bằng tag nhưng chưa thể hiện mức độ tương quan giữa các tag ấy và đã được khắc phục bằng việc lưu trữ thông tin dưới *dạng có cấu trúc cây*. *Nhu cầu so khớp thông tin cũng ngày càng gia tăng*: tìm việc, tìm bạn, tìm sản phẩm, ... Từ những luận điểm trên, chúng tôi nhận thấy cần có một kiến trúc chung cho việc lưu trữ và so khớp thông tin và kiến trúc này hoàn toàn có tính khả thi.

### 3.6 Lý thuyết và phương pháp xây dựng kiến trúc tổ chức thông tin linh hoạt và so khớp thông tin

- Về việc tổ chức thông tin về nhu cầu và “thông tin đáp ứng nhu cầu” một cách linh hoạt, chúng tôi đã sử dụng **Tag** trong việc lưu trữ các thuộc tính của nhu cầu và thông tin đáp ứng nhu cầu, tuy nhiên, cần có sự phân cấp giữa các thuộc tính với nhau, nên chúng tôi đã kết hợp **Tag** và **Taxonomy** để lưu trữ thuộc tính dưới dạng cấu trúc cây.

- Bên cạnh đó, để giải quyết vấn đề cây có cấu trúc và có mức độ tương quan, giống nhau giữa các thuộc tính, chúng tôi đã áp dụng “**Độ tương quan giữa các tag**” một cách đơn giản là khi truyền hai từ khoá bất kỳ vào hệ thống, hệ thống sẽ trả về mức độ tương quan của hai thuộc tính đã truyền vào.

- Về vấn đề so khớp giữa nhu cầu và “thông tin đáp ứng nhu cầu” chúng tôi sẽ đề cập trong phần “**Lý thuyết cơ sở để so khớp thông tin**”. Hệ thống so khớp sẽ giúp cải thiện “thông tin đáp ứng nhu cầu”, đặc biệt giúp cho việc đánh giá mức độ so khớp giữa các thuộc tính trong nhu cầu với “thông tin đáp ứng nhu cầu cụ thể”.

- Lý thuyết về “**Cây quyết định**” được chúng tôi áp dụng trong việc gợi ý cải thiện “thông tin đáp ứng nhu cầu” cho toàn bộ những nhu cầu trong lĩnh vực cụ thể và có liên quan với “thông tin đáp ứng nhu cầu”.

### 3.6.1 Tag<sup>4</sup>

Việc phân lớp phân cấp dữ liệu nhằm tổ chức thông tin trong hệ thống có thể được thực hiện bằng nhiều phương pháp khác nhau, một số phương pháp cụ thể như phân chia nội dung theo loại thông tin, theo cấu trúc cây (như các bài viết trong forum được đặt vào một mục cụ thể của cây danh mục). Một phương pháp khá phổ biến để tổ chức, phân loại và chia sẻ một lượng dữ liệu lớn trong cộng đồng mạng xã hội hiện nay là kỹ thuật **Tagging**, nó là *một cách đơn giản và trực quan để tổ chức nguồn thông tin, cho phép người dùng gắn các từ khóa hoặc tag (thẻ) vào thông tin hoặc đối tượng dữ liệu*; còn được biết đến với thuật ngữ social tagging, collaborative tagging, social classification, và social indexing.

“**Tag** là một từ khóa không có thứ tự hay một thuật ngữ nhằm chỉ một mẫu thông tin (như một bookmark trên internet, hình ảnh kỹ thuật số, hoặc một tập tin máy tính). Loại dữ liệu biến đổi này giúp miêu tả một mục tin và cho phép người sử dụng tìm lại mục đó bằng cách duyệt hay dò tìm.”<sup>5</sup>

Có 2 loại tagging phổ biến đó là Author-based tagging và User-based tagging. Author-based tagging được thực hiện bởi người tạo nội dung, được xem là phương pháp phân lớp theo hướng top-down. User-based tagging cho phép người dùng tự quyết định việc phân lớp nội dung mà họ quan tâm, kỹ thuật này là phương pháp phân lớp theo hướng bottom-up. Một số hệ thống có ứng dụng tagging tiêu biểu là Delicious (<http://del.icio.us>), Flickr (<http://www.flickr.com>) và các trang mạng xã hội như Facebook, Google plus...

<sup>4</sup> Tham khảo [5]

<sup>5</sup> Nguồn: <http://vi.wikipedia.org/wiki/Tag>

Thuận lợi và bất lợi của việc sử dụng kỹ thuật tagging:

- ❖ Thuận lợi: trong một hệ thống có sử dụng kỹ thuật tagging để mô tả thông tin mà trong đó nếu không xét về gốc độ ý nghĩa và ngữ nghĩa (semantics) của mỗi thẻ tag, thì ưu điểm của sử dụng kỹ thuật tagging là người sử dụng dễ dàng áp mô tả thông tin của mình qua những thẻ tag một cách linh hoạt.
- ❖ Bất lợi: một khi người sử dụng tự do chọn các thẻ để mô tả thông tin của mình và nếu không kiểm soát các tag họ tự do lựa chọn thì sẽ xảy ra một số vấn đề khó khăn cho hệ thống: có nhiều tag (một thẻ có thể mô tả nhiều nghĩa khác nhau; từ nhiều nghĩa), các thẻ khác nhau nhưng cho nghĩa giống nhau (đồng nghĩa). Điều này sẽ gây khó khăn cho hệ thống tìm kiếm. Vì vậy những hệ thống rất nhiều thẻ mô tả thông tin, gọi là Folksonomies phải kiểm soát các thẻ chặt chẽ hơn.

**Kết luận: theo xu hướng của việc tổ chức và lưu trữ thông tin, thẻ tag đã giải quyết được vấn đề làm sao tổ chức thông tin một cách đơn giản và linh hoạt.**

### 3.6.2 Taxonomy<sup>6</sup>

#### 3.6.2.1 Tại sao sử dụng taxonomy

Như đã nói ở những phần trên, mô tả một thông tin của một đối tượng bằng cách chia nhỏ thông tin đó ra nhiều mảng nhỏ thông tin cụ thể, mỗi mảng nhỏ thông tin đó được mô tả dưới dạng một tag (keyword). Nhưng nội dung thông tin của một đối tượng thì rất phong phú. Vì vậy mô tả thông tin của một đối tượng sẽ được mô tả rất nhiều tag, tập hợp một khối các tag. Folksonomy là một thuật ngữ riêng liên quan tới các thẻ tag, là một hệ thống phân loại xuất phát từ các phương pháp quản lý việc tạo ra các thẻ tag, quản việc chú thích và phân loại nội dung các thẻ, một số thuật ngữ liên quan là **collaborative tagging, social classification, social indexing, and social tagging**. Kết luận: tag giúp mô tả thông tin một cách linh hoạt, dễ dàng tuy nhiên điều cần thiết là phải quản lý các thẻ tag, việc phân loại (**classification**) và tổ chức (**organization**) các thẻ tag một vấn đề cần phải xử lý. Thuật ngữ chuyên sâu về phân loại và tổ chức thông tin là **Taxonomy**.

---

<sup>6</sup> Tham khảo [5], [13], [23]

### 3.6.2.2 Tổng quan về taxonomy

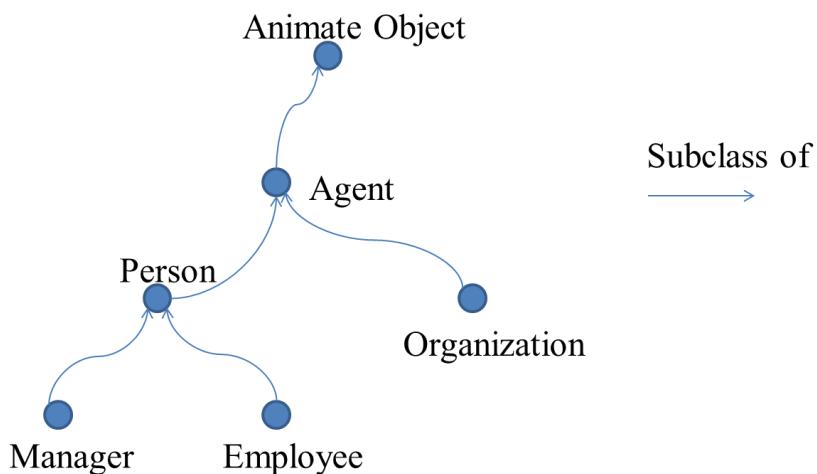
Vào những năm 90 của thế kỷ XX, khái niệm taxonomy được sử dụng trong nhiều lĩnh vực khác nhau như khoa học xã hội, tâm lý học và công nghệ thông tin... để thiết lập sự trùng hợp giữa thuật ngữ của người sử dụng và thuật ngữ của hệ thống. Các chuyên gia đầu tiên phát triển cấu trúc hệ thống web đã dùng thuật ngữ taxonomy để nói đến việc tổ chức nội dung các trang web. Kể từ đó, khái niệm taxonomy được sử dụng rộng rãi với mục đích này. Do được sử dụng trong nhiều lĩnh vực khác nhau, nên cũng có nhiều định nghĩa về taxonomy. Từ năm 2000 đến 2005 có hơn 36 định nghĩa khác nhau, dựa theo tài liệu của Hypertext<sup>7</sup> chúng tôi định nghĩa taxonomy như sau:

**Taxonomy là sự phân loại toàn bộ thông tin trong một hệ thống có phân cấp, sự phân loại này theo một mối quan hệ có trước của các thực thể trong thế giới thực mà nó biểu diễn.**

Một taxonomy thường được mô tả với gốc ở trên cùng, mỗi nút của taxonomy – bao gồm cả gốc – là một thực thể thông tin đại diện cho một thực thể trong thế giới thực. Giữa các nút trong taxonomy có một mối quan hệ đặc biệt gọi là *is subclassification of* nếu hướng liên kết từ nút con lên nút cha hoặc là *is superclassification of* nếu hướng liên kết từ nút cha xuống nút con. Đôi khi những quan hệ này được xác định một cách chặt chẽ hơn là *is subclass of* hoặc *is superclass of*, nếu thực thể thông tin là một lớp đối tượng.

Ví dụ sau đây mô tả một taxonomy đơn giản gồm lớp Person, lớp con của nó là Employee, Manager; Lớp cha của Person là Agent. Khi đi lên từ gốc của taxonomy, các thực thể khái quát hơn. Khi đi xuống những lá ở cuối, thực thể xác định rõ ràng hơn. Ví dụ, Agent chung chung hơn Person, Employee cụ thể hơn Person.

<sup>7</sup> Tham khảo: <http://www.hipertext.net/english/pag1011.htm#origenNota2>



**Hình 2. Ví dụ về một taxonomy có cấu trúc đơn giản**

Taxonomy thể hiện ưu điểm trong việc phân lớp thực thể thông tin theo ngữ nghĩa, chúng thiết lập một quan hệ ngữ nghĩa đơn giản để phân biệt giữa các đối tượng trong một miền thông tin.

Taxonomy đóng vai trò rất quan trọng trong việc tổ chức thông tin và tổ chức tri thức. Nó được sử dụng chủ yếu để giúp cho việc tìm kiếm và duyệt thông tin thuận lợi và nhanh chóng hơn, đặc biệt khi chúng ta chỉ có những thông tin chung chung về vấn đề cần tìm kiếm. Khi tìm kiếm trên Internet, nếu sử dụng từ khoá để tìm kiếm thông tin, kết quả trả về có thể từ vài nghìn đến vài chục nghìn tài liệu về các chủ đề khác nhau. Sử dụng taxonomy để tìm kiếm và duyệt thông tin sẽ tiết kiệm được rất nhiều thời gian cho người dùng để tìm được thông tin cần thiết. Đồng thời, taxonomy cho phép các máy tìm kiếm và các ứng dụng có thể dễ dàng tìm được các thực thể thông tin nhanh và chính xác hơn nhiều.

Taxonomy đã được áp dụng trong nhiều bài toán khác nhau: OU Shi-yan, KHOO Christopher S.G, GOH Dion H<sup>8</sup>. xây dựng taxonomy hỗ trợ việc tóm tắt tự động văn bản; H.T.Kung và C.H.Wu xây dựng taxonomy cho mạng nội dung<sup>9</sup>, Wollersheim và Rahayu<sup>10</sup> xây dựng một taxonomy hỗ trợ việc duyệt cơ sở dữ liệu về y tế.

<sup>8</sup> Tài liệu tham khảo [15]

<sup>9</sup> Tài liệu tham khảo [13]

<sup>10</sup> Tài liệu tham khảo [14]

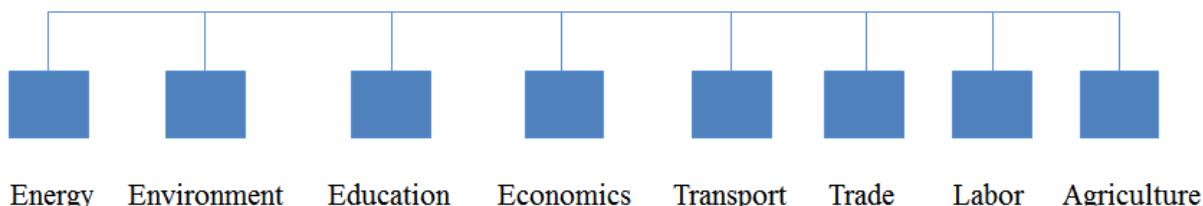
### 3.6.2.3 Phân loại taxonomy

Có 04 loại taxonomy:

- Loại 1: Flat (phân lớp phẳng)
- Loại 2: Hierarchical (cấu trúc dạng cây)
- Loại 3: Faceted
- Loại 4: Network

#### 3.6.2.3.1 Flat taxonomy

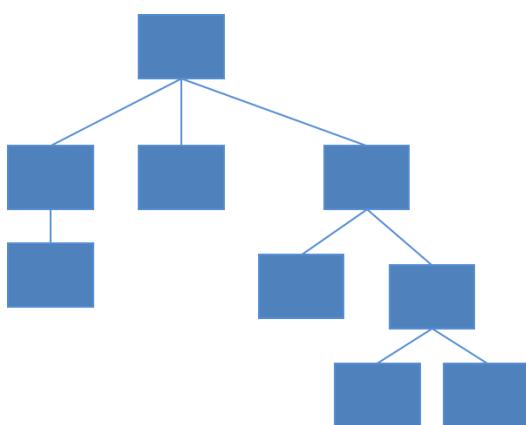
Flat taxonomy gom nhóm thông tin vào một tập hợp danh mục sẵn có, không có quan hệ thuộc về giữa các mục, chúng ngang hàng với nhau. Flat taxonomy thường được ứng dụng trong việc phân nhóm sản phẩm theo alphabet.



**Hình 3. Ví dụ về flat taxonomy**

#### 3.6.2.3.2 Hierarchical taxonomy

Hierarchical taxonomy được thể hiện dưới dạng cấu trúc cây, mỗi node chỉ có một node cha và không giới hạn số node con. Ứng dụng tiêu biểu của Hierarchical taxonomy là cây phân cấp các chủ đề trong forum.

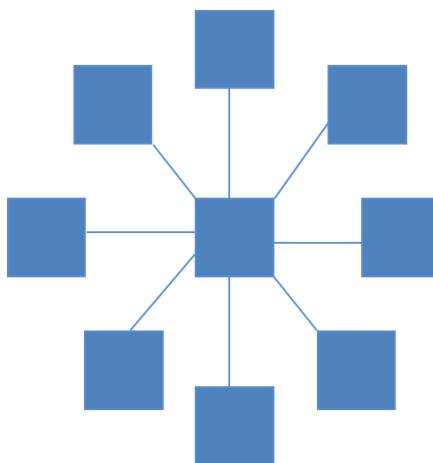


**Hình 4. Ví dụ về hierarchical taxonomy**

Như đã nói ở trên, chúng tôi lựa chọn tổ chức thông tin theo dạng cấu trúc cây làm tiền đề cho việc phát triển kiến trúc của nhóm, việc bắt đầu từ việc lưu trữ theo cấu trúc cây giúp cho ứng dụng có thể dễ dàng phát triển hơn trong tương lai.

### 3.6.2.3.3 Faceted taxonomy

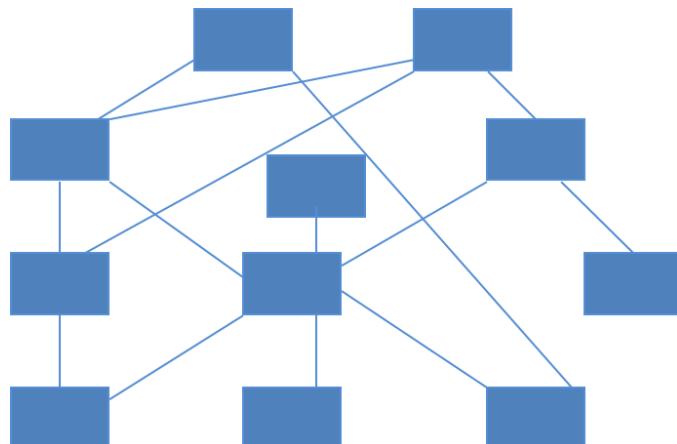
Faceted taxonomy thể hiện sự phân lớp theo cấu trúc dữ liệu ngôi sao. Mỗi node trong cấu trúc này là một điểm trung tâm, có thể liên kết đến những node khác trong cấu trúc ngôi sao khác. Cũng giống như flat taxonomy, không có quan hệ thuộc về, hay quan hệ cha con giữa các mục trong faceted taxonomy. Tất cả các mục trong faceted taxonomy liên kết đến một đối tượng duy nhất – mỗi node mô tả thuộc tính hoặc giá trị hay một khía cạnh của vấn đề. Ví dụ như sách điện tử, mỗi node mô tả một khía cạnh về cuốn sách – tác giả, tên sách, ngày xuất bản...



**Hình 5. Ví dụ về faceted taxonomy**

### 3.6.2.3.4 Network taxonomy

Network taxonomy là một cấu trúc mạng, mỗi node có thể có nhiều node cha. Mỗi node trong network taxonomy có thể liên kết đến một node bất kỳ. Với network taxonomy thông tin được tổ chức theo cả hai dạng cấu trúc cây và dạng liên kết, các quan hệ giữa các node có nhiều ý nghĩa khác nhau. Network taxonomy có thể ứng dụng trong hệ thống quản lý kiến thức như công cụ gợi ý, semantic network (mạng ngữ nghĩa).



Hình 6. Ví dụ về network taxonomy

### 3.6.3 Lý thuyết cơ sở để so khớp thông tin

#### 3.6.3.1 Các loại dữ liệu trong khai thác thông tin (Content types – Data mining)<sup>11</sup>

Một trong những cơ sở để so khớp thông tin với nhau, đó là cần xác định loại dữ liệu của thông tin, loại dữ liệu (content type) biểu hiện cấu trúc hành vi của thông tin được lưu trữ. Ví dụ, nếu thông tin lặp lại trong một khoảng xác định, như các ngày trong tuần là dữ liệu tuần hoàn (cyclical). Các loại dữ liệu trong khai thác dữ liệu được phân loại như sau:

- ❖ **Kiểu rời rạc (Discrete)**

Kiểu dữ liệu rời rạc là các giá trị của nó chỉ nằm trong danh sách giá trị hữu hạn, không có tính liên tục giữa các giá trị. Ví dụ, giới tính là một thuộc tính rời rạc cơ bản, nó thể hiện một giá trị trong danh mục được định sẵn, là nam hoặc nữ.

Các giá trị của thuộc tính rời rạc không thể thực hiện sắp xếp, mặc dù giá trị là kiểu số. Hơn nữa, nếu giá trị của nó là số thì các giá trị này không thể tính toán được. Ví dụ số khu vực là một ví dụ điển hình dữ liệu rời rạc kiểu số.

- ❖ **Kiểu liên tục (Continuous)**

Kiểu dữ liệu liên tục là các giá trị thể hiện dữ liệu số numeric trong một phạm vi rộng. Khác với kiểu rời rạc, thể hiện sự hữu hạn, dữ liệu đếm được, một thuộc tính liên tục thể hiện các phép đo có khả năng mở rộng, và có thể chứa các giá trị dữ liệu vô hạn.

<sup>11</sup> Tham khảo <http://technet.microsoft.com/en-us/library/ms174572.aspx>

### ❖ Kiểu thứ tự (Ordered)

Loại dữ liệu có thứ tự chỉ các giá trị có trình tự, trật tự nhất định. Tuy nhiên, loại dữ liệu này có trình tự nhưng không biểu hiện khoảng cách hay mối quan hệ nào giữa các giá trị trong tập hợp. Ví dụ, nếu một thuộc tính chứa thông tin về trình độ kỹ năng được đánh giá theo thứ tự từ 1 đến 5, thì không thể suy ra khoảng cách giữa các trình độ này, trình độ đạt mức 5 không nhất thiết là cao hơn 5 lần so với mức độ 1.

### ❖ Kiểu tuần hoàn (Cyclical)

Kiểu tuần hoàn nhằm mô tả giá trị thể hiện một tập hợp giá trị tuần hoàn có trình tự. Ví dụ, các thứ trong tuần là một tập hợp tuần hoàn có trình tự, bởi vì ngày thứ hai theo sau ngày thứ bảy.

#### *3.6.3.2 Các phương thức so sánh hai dữ liệu với nhau*

Để so sánh hai dữ liệu cụ thể với nhau, ta có các phương thức so sánh khác nhau hướng đến một mục đích và phạm vi riêng biệt đến tính ra kết quả độ tương đồng giữa hai dữ liệu. Ta có một số phương thức so sánh<sup>12</sup>sau:

##### *3.6.3.2.1 Absolute Match*

Phương pháp so sánh này sẽ thực hiện so sánh hai thuộc tính và chỉ so khớp hoàn toàn. Nếu khớp hoàn toàn thì kết quả được 1 điểm, trường hợp khác sẽ là 0 điểm. Ví dụ thuộc tính thứ nhất là “Apple” và thuộc tính thứ hai cũng là “Apple” vậy là hai thuộc tính khớp hoàn toàn với nhau, nên được 1 điểm, ngược lại so ánh “Apple” và “Application” thì kết quả là 0.

##### *3.6.3.2.2 Soundex Match*

Đây là phương thức so sánh dữ liệu theo phát âm của các từ trong tiếng Anh. Mục đích cơ bản là những từ với cách phát âm tương tự nhau được mã hóa thành cùng một chuỗi để có thể thực hiện so khớp mặc cho sự khác biệt nhỏ về chính tả.

<sup>12</sup> Tham khảo <http://msdn.microsoft.com/en-us/magazine/gg598923.aspx>

### 3.6.3.2.3 Lookup Matching

Lookup matching là phương pháp so sánh bằng cách tìm kiếm từ khóa cần so sánh trong một tập hợp từ khóa có sẵn. Ví dụ như so khớp một tên đường phố có nằm trong danh sách tên đường yêu cầu hay không.

### 3.6.4 Cây quyết định

#### 3.6.4.1 Giới thiệu về cây quyết định<sup>13</sup>

- ✓ Cây quyết định là một cây phân cấp có cấu trúc.
- ✓ Dùng để phân lớp đối tượng dựa vào dãy các luật (series of rules), các luật này được sinh ra từ tập dữ liệu (training set).
- ✓ Các thuộc tính phân lớp thường có kiểu dữ liệu là binary, nominal, ordinal, continuos.
- ❖ Ví dụ “Phương tiện di chuyển”: Cho tập dữ liệu (training set) như sau:

Thuộc tính (Attribute)				Thuộc tính phân lớp (Class)
Gender	Car ownership	Travel cost (\$/km)	Income level	Transportation mode
Male	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Female	1	Cheap	Medium	Train
Female	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Male	0	Standard	Medium	Train
Female	1	Standard	Medium	Train
Female	1	Expensive	High	Car
Male	2	Expensive	Medium	Car
Female	2	Expensive	High	Car

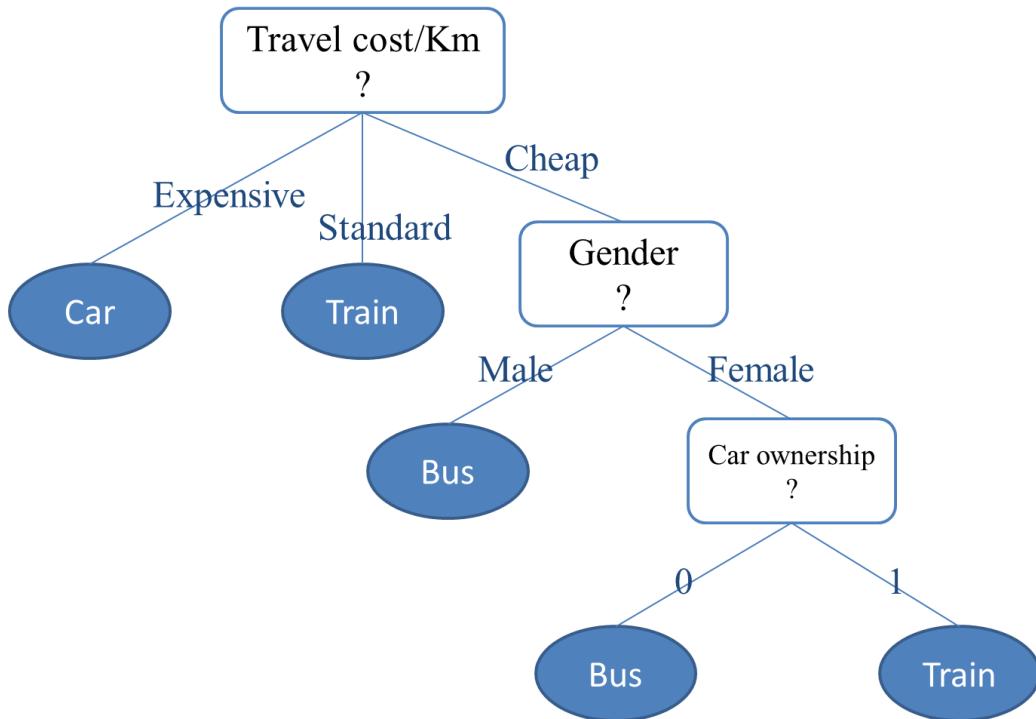
Bảng 1. Ví dụ bảng dữ liệu lựa chọn phương tiện di chuyển

- ❖ Tập dữ liệu trên mô tả 10 đối tượng, mỗi đối tượng được miêu tả bằng 4 thuộc tính là “Gender” (kiểu dữ liệu binary), “Car ownership” (quantitative interger),

<sup>13</sup> Tham khảo [9], [10]

“Travel cost/km” (ordinal), “Income level” (ordinal) và thuộc tính phân loại – category attribute – “Transportation mode” (ordinal).

- ❖ Từ tập dữ liệu trên, chúng ta có thể tạo ra cây quyết định như sau:



**Hình 7. Ví dụ về cây quyết định**

- ❖ Trong cây quyết định trên, thuộc tính “Income level” không xuất hiện trong cây; vì dựa vào Bảng 1, thuộc tính “Travel cost/Km” sẽ sinh ra cây quyết định dùng để phân lớp tốt hơn “Income level”

#### 3.6.4.2 Sử dụng cây quyết định để dự đoán lớp các dữ liệu chưa biết

- ✓ Mục đích chính của cây quyết định là dùng để xác định lớp hay nói khác đi là dự đoán lớp của các dữ liệu chưa biết dựa vào cây quyết định được sinh ra từ tập dữ liệu đào tạo (training data)

- ❖ Ví dụ: dựa vào ví dụ ở phần Giới thiệu về cây quyết định. Cho tập dữ liệu cần dự đoán sau đây:

Name	Gender	Car ownership	Travel cost (\$/km)	Income level	Transportation mode
Ngoc	Female	1	Cheap	High	?

Hieu	Male	0	Standard	High	?
Phuc	Male	1	Cheap	Medium	?
Nguyen	Male	2	Expensive	High	?

Bảng 2. Ví dụ về bảng dữ liệu cần dự đoán phương tiện di chuyển

- ❖ Dựa vào cây quyết định (Hình 7), cây quyết định sẽ được duyệt từ nút gốc “Travel cost/km”, dãy các luật sau sẽ được sinh ra:
  - Nếu “Travel cost/Km” là **Expensive** thì người đó sẽ chọn phương tiện di chuyển là **car**.
  - Nếu “Travel cost/Km” là **Standard** thì người đó sẽ chọn phương tiện di chuyển là **train**
  - Nếu “Travel cost/Km” là **Cheap**, chúng ta sẽ xem xét thuộc tính “Gender”:
    - Nếu “Gender” là **Male**, người đó sẽ chọn phương tiện là **bus**
    - Nếu “Gender” là **Female**, thì xem người đó sở hữu bao nhiêu xe hơi (thuộc tính “Car ownership”). Nếu số xe sở hữu là **0**, thì người đó sẽ chọn phương tiện di chuyển là **bus**; ngược lại, nếu số xe sở hữu lớn hơn hay bằng **1**, thì người đó sẽ chọn phương tiện di chuyển là **train**.
- ❖ Bảng 2 sẽ được dự đoán như sau:

Name	Gender	Car ownership	Travel cost (\$/km)	Income level	Transportation mode
Ngoc	Female	1	Cheap	High	Train
Hieu	Male	0	Standard	High	Train
Phuc	Male	1	Cheap	Medium	Bus
Nguyen	Male	2	Expensive	High	Car

Bảng 3. Bảng dữ liệu kết quả dự đoán phương tiện di chuyển

- ✓ Một số chú ý khi sử dụng cây quyết định:
  - Phụ thuộc rất nhiều vào training data, tập dữ liệu training data càng lớn thì cây quyết định sẽ đáng tin cậy hơn.
  - Không thể nói cây quyết định được sinh ra từ cây quyết định trên là tập luật tốt nhất

- Có nhiều thuật toán phân lớp: ID3, J48, C4.5, C5, CART (Classification and Regression Tree), ... Việc lựa chọn thuật toán phụ thuộc vào rất nhiều yếu tố, trong đó yếu tố cấu trúc dữ liệu ảnh hưởng rất nhiều đến kết quả của thuật toán. Chẳng hạn, thuật toán ID3 và CART hiệu quả cho việc phân lớp đối với các dữ liệu số (quantitative value), trong khi đó, thuật toán J48, C4.5 có hiệu quả hơn đối với dữ liệu Qualitative value (ordinal, binary, nominal)

### 3.6.4.3 Một số độ đo thông dụng

- ✓ Cho bảng dữ liệu bao gồm các thuộc tính và thuộc tính phân lớp, chúng ta có thể đo được tính đồng nhất hay không đồng nhất thông qua thuộc tính phân lớp. Bảng dữ liệu có tính đồng nhất nếu nó có duy nhất một phân lớp. Ngược lại, nếu nó có nhiều phân lớp khác nhau, thì bảng dữ liệu có tính không đồng nhất hay tính pha trộn. Chúng ta có thể đo được mức độ pha trộn (Impurity Degree); Entropy, độ đo Gini và classification error là những cách tính mức độ pha trộn thông dụng nhất.

$$\text{Entropy} = \sum_j -p_j \log_2 p_j$$

$$\text{Gini index} = 1 - \sum_j p_j^2$$

$$\text{Classification error} = 1 - \max\{p_j\}$$

$P_j$ : xác suất xảy ra phân lớp j

- ❖ Ví dụ: Chúng ta cùng nhìn lại Bảng 1. Ví dụ bảng dữ liệu lựa chọn phương tiện di chuyển. “Transportation mode” có 3 nhóm Bus, Car và Train. Bảng 1 có 10 dòng dữ liệu, trong đó “Transportation mode” có 4 buses, 3 cars, 3 trains (4B, 3C, 3T).

- ❖ Xác suất để xảy ra cho từng phân lớp là:

$$P(\text{Bus}) = \frac{4}{10} = 0.4$$

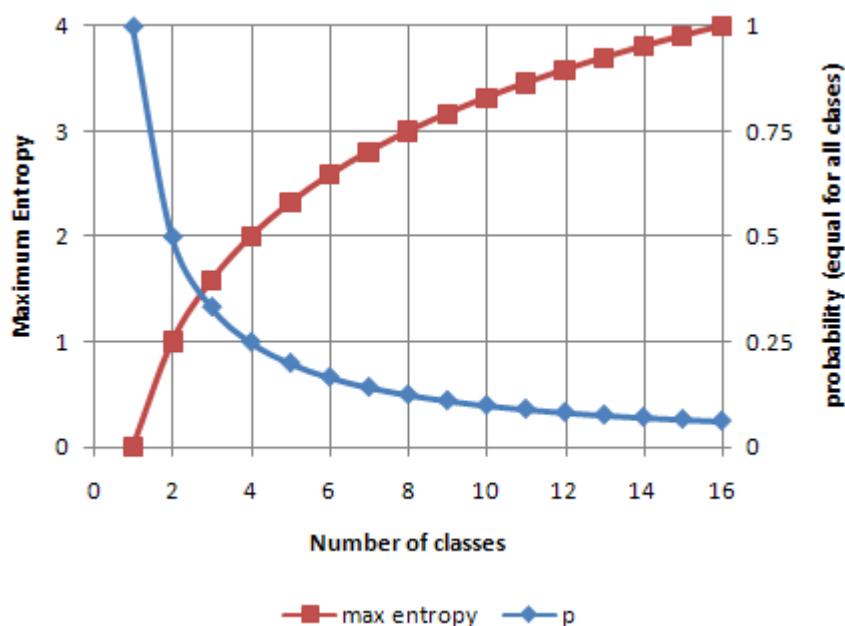
$$P(\text{Car}) = \frac{3}{10} = 0.3$$

$$P(\text{Train}) = \frac{3}{10} = 0.3$$

### 3.6.4.3.1 Entropy

$$\begin{aligned}
 Entropy &= \sum_j -p_j \log_2 p_j \\
 &= -P(Bus) \log_2 P(Bus) - P(Car) \log_2 P(Car) \\
 &\quad - P(Train) \log_2 P(Train) = -0.4 \log_2 0.4 - 0.3 \log_2 0.3 \\
 &\quad - 0.3 \log_2 0.3 = 1.571
 \end{aligned}$$

Khi bảng dữ liệu có duy nhất 1 phân lớp thì Entropy sẽ bằng 0 vì xác suất là 1 và  $\log_2 1 = 0$ . Entropy sẽ đạt giá trị cực đại khi tất cả thuộc tính phân lớp có xác suất xảy ra bằng nhau. Biểu đồ dưới đây thể hiện giá trị cực đại của Entropy sẽ thay đổi phụ thuộc vào số lượng thuộc tính phân lớp n, trong trường hợp xác xuất tất cả thuộc tính phân lớp  $P = \frac{1}{n}$  thì  $Entropy_{MAX} = -n * p * \log_2 p$ . Giá trị của Entropy sẽ lớn hơn 1 khi số lượng thuộc tính phân lớp nhiều hơn 2



Biểu đồ 1. Mối liên hệ giữa giá trị Entropy cực đại và số lượng phân lớp

### 3.6.4.3.2 Gini index

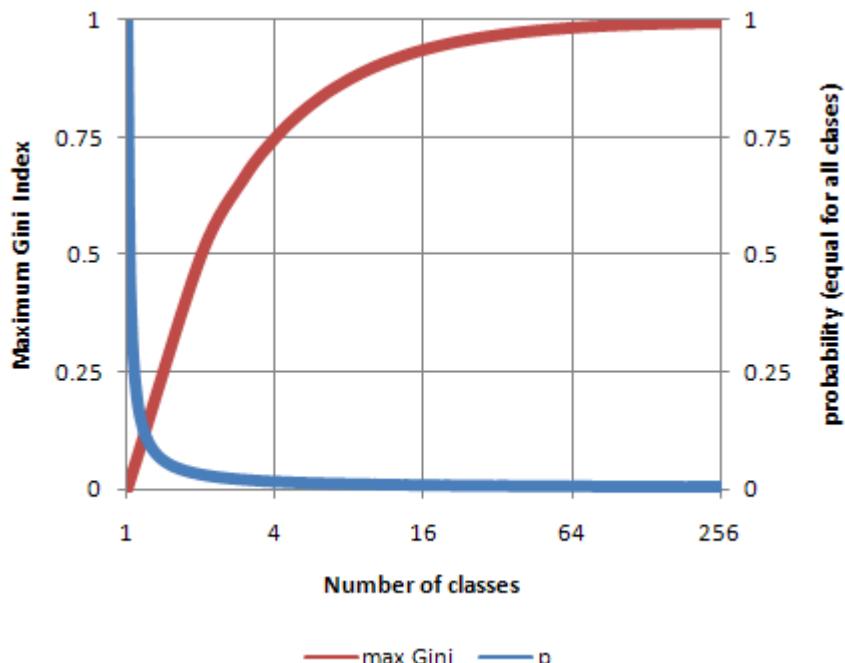
- ✓ Một cách khác để tính “Impurity degree”.

$$Gini\ index = 1 - \sum_j p_j^2$$

- ❖ Chúng ta cùng quay lại ví dụ ở trên

$$\text{Gini index} = 1 - \sum_j p_j^2 = 1 - (0.4^2 + 0.3^2 + 0.3^2) = 0.660$$

- ✓ Khi bảng dữ liệu có duy nhất 1 phân lớp thì Gini index sẽ bằng 0 vì xác suất bằng 1 và  $1 - 1^2 = 0$ . Cũng giống như Entropy, Gini index sẽ đạt giá trị cực đại khi tất cả thuộc tính phân lớp có xác suất xảy ra bằng nhau. Biểu đồ dưới đây thể hiện giá trị cực đại của Gini index sẽ khác nhau phụ thuộc vào số lượng phân lớp n, khi xác xuất tất cả thuộc tính phân lớp  $P = \frac{1}{n}$
- ✓  $0 \leq \text{Gini index} \leq 1$



**Biểu đồ 2. Mối liên hệ giữa giá trị Gini index cực đại và số lượng phân lớp**

#### 3.6.4.3.3 Classification error

$$\text{Classification error} = 1 - \max\{p_j\}$$

- ❖ Trong ví dụ trên

$$\text{Classification error} = 1 - \max\{p_j\} = 1 - \max\{0.4, 0.3, 0.3\} = 1 - 0.4 = 0.6$$

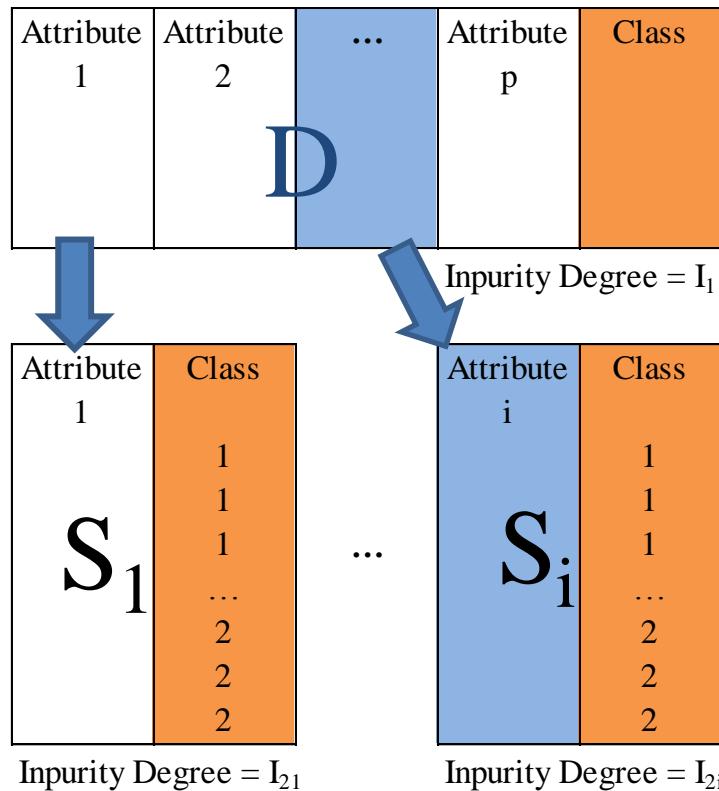
- ✓ Cũng giống như Entropy và Gini index, Classification error sẽ bằng 0 khi bảng dữ liệu có duy nhất 1 phân lớp vì xác suất bằng 1 và  $1 - \max(1) = 0$ .
- ✓  $0 \leq \text{Classification error} \leq 1$

### 3.6.4.4 Nguyên tắc hoạt động của một số thuật toán xây dựng cây quyết định thường dùng

- ✓ Các thuật toán xây dựng cây quyết định thường được sử dụng nhất là ID3, C4.5 và CART (classification and regression trees).
- ✓ Nhìn chung, các thuật toán xây dựng cây quyết định đều được xây dựng trên quy tắc đệ quy. Ví dụ, Hunt là thuật toán xây dựng cây quyết định, đệ quy theo nút của cây, bắt đầu từ nút gốc. Mặc dù kết quả đạt được từ thuật toán Hunt không được tối ưu, nhưng đây là một trong những một trong những thuật toán dùng để xây dựng cây quyết định sớm nhất.

#### 3.6.4.1 Lần lặp đầu tiên

- ✓ Giả sử, chúng ta có một bảng dữ liệu chứa các thuộc tính và thuộc tính phân lớp, tạm gọi bảng này là D. Từ bảng D, chúng ta sẽ lấy ra từng cột thuộc tính trong bảng để đối chiếu với các giá trị của thuộc tính phân lớp. Nếu chúng ta có p cột dữ liệu, và lấy từng phần tử p là tập con của D, tạm gọi từng tập hợp này là  $S_i$ . Bảng D là tập hợp những  $S_i$  và thuộc tính phân lớp.



**Hình 8. Phân chia bảng D thành những tập con  $S_i$**

✓ Sau khi phân chia bảng dữ liệu D như trên, chúng ta sẽ tính mức độ pha trộn (tham khảo cách tính tại phần 3.6.4.3 Một số độ đo thông dụng)

❖ Ví dụ, dựa vào Bảng 1. Ví dụ bảng dữ liệu lựa chọn phương tiện di chuyển. chúng ta có thể tính “Impurity degree” dựa vào thuộc tính phân lớp “Transportation mode”. “Transportation mode” có 4 busses, 3 cars và 3 trains (4B, 3C, 3T)

Thuộc tính (Attribute)				Thuộc tính phân lớp (Class)
Gender	Car ownership	Travel cost (\$/km)	Income level	Transportation mode
Male	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Female	1	Cheap	Medium	Train
Female	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Male	0	Standard	Medium	Train
Female	1	Standard	Medium	Train
Female	1	Expensive	High	Car
Male	2	Expensive	Medium	Car
Female	2	Expensive	High	Car

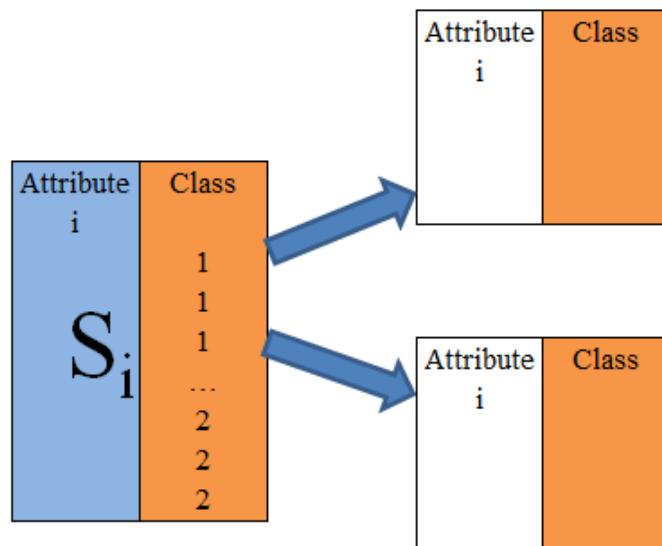
4B, 3C, 3T

Entropy 1.571

Gini index 0.660

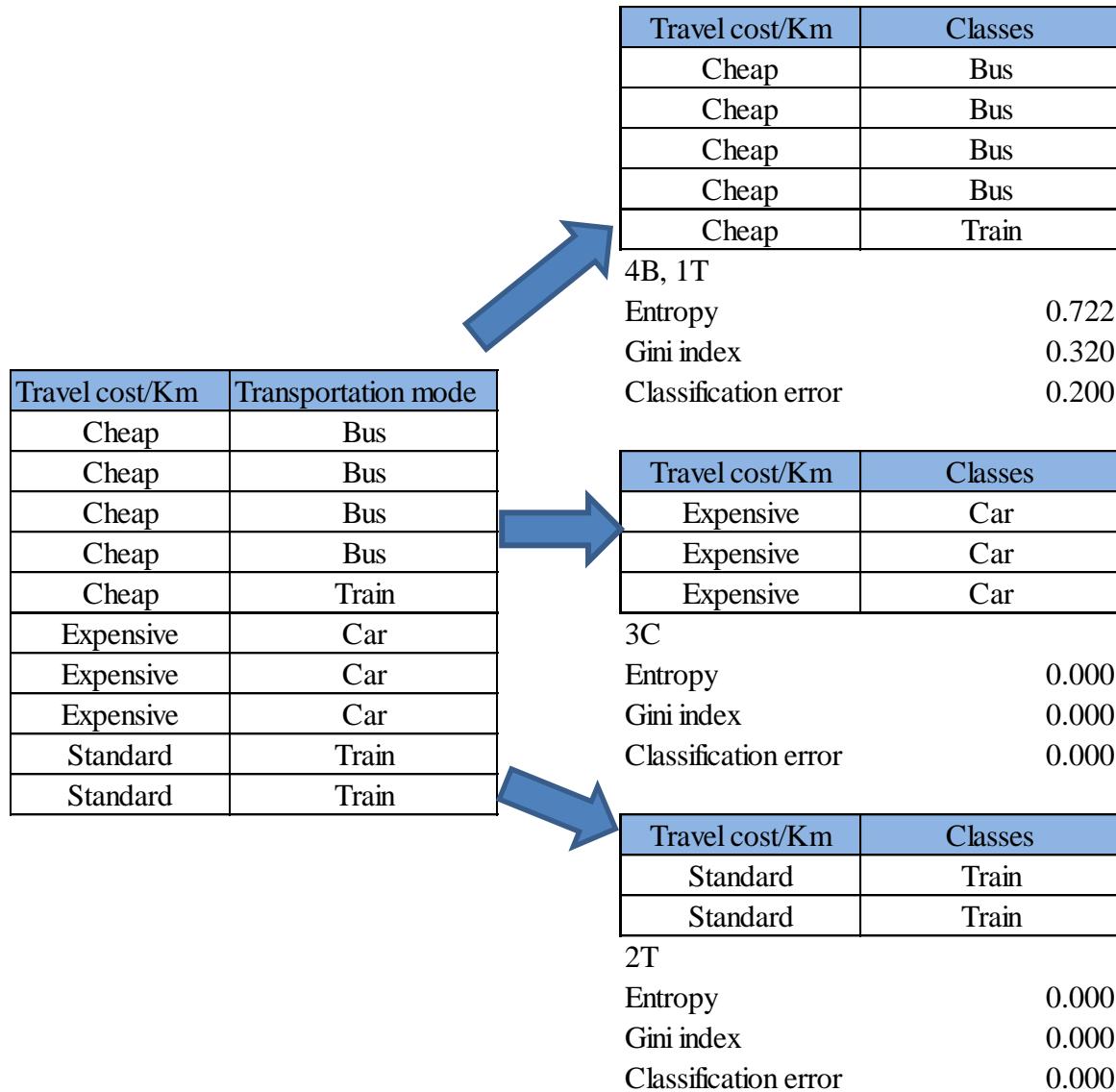
Classification index 0.600

✓ Sau khi phân chia bảng D thành từng tập con  $S_i$ , chúng ta tiếp tục phân chia  $S_i$  để tính “Impurity degree” cho từng giá trị thuộc tính trong từng tập  $S_i$



Hình 9. Phân chia  $S_i$  dựa vào phân lớp

- ❖ Tiếp tục ví dụ trên, thuộc tính “Travel cost/km” có 3 giá trị, đó là “Cheap”, “Standard” và “Expensive”.

**Hình 10. Độ lợi thông tin thuộc tính Travel cost/Km**

✓ **Độ lợi thông tin (Information Gain):**

- Có sự khác nhau giữa các cách tính mức độ pha trộn “Impurity degree” giữa bảng D và các tập con Si, chúng ta sẽ tiến hành so sánh “Impurity degree” trước và sau khi phân chia bảng thành những tập con (ví dụ như phân chia D thành từng tập con Si). Information gain là một phương pháp để đo lường sự khác nhau này. Chúng ta sẽ so sánh lợi ích khi chia bảng dữ liệu theo các giá trị của thuộc tính, từ đó chọn ra thuộc tính tối ưu để phân chia

*Information gain*

$$= \text{Entropy of Parent table} - \sum_n \left( \frac{k}{n} * \text{Entropy of each value } k \text{ of } S_i \right)$$

- ❖ Xét ví dụ trên, bảng D có 3 phân lớp 4B, 3C và 3T và có Entropy là 1.571. Bây giờ chúng ta sẽ sử dụng thuộc tính “Travel cost/km” để chia bảng thành 3 phần:
- ❖ “Travel cost/km” có giá trị là Cheap, thuộc tính phân lớp có 4B và 1T

$$P(Bus) = \frac{4}{5} = 0.8$$

$$P(Train) = \frac{1}{5} = 0.2$$

$$\text{Entropy} = \sum_j -p_j \log_2 p_j = -0.8 \log_2 0.8 - 0.2 \log_2 0.2 = 0.722$$

- ❖ “Travel cost/km” có giá trị là Standard, thuộc tính phân lớp có 2T, Entropy = 0 vì lúc này chỉ có duy nhất 1 thuộc tính phân lớp)
- ❖ “Travel cost/km” có giá trị là Expensive, thuộc tính phân lớp có 3C, Entropy = 0 vì lúc này chỉ có duy nhất 1 thuộc tính phân lớp)

*Information gain*

$$= \text{Entropy of Parent table} - \sum_n \left( \frac{k}{n} * \text{Entropy of each value } k \text{ of } S_i \right)$$

$$= 1.571 - \left( \frac{5}{10} * 0.722 + \frac{2}{10} * 0 + \frac{3}{10} * 0 \right) = 1.210$$

- ❖ Tương tự chúng ta sẽ tính được độ đo Gini index và Classification error:

Lợi ích khi chia theo thuộc tính “Travel cost/km” dựa vào	
Entropy	1.210
Gini index	0.500
Classification error	0.500

Bảng 4. Lợi ích khi chia bảng D theo thuộc tính “Travel cost/km”

- ❖ Chúng ta thực hiện lần lượt cho các thuộc tính còn lại của bảng D: “Gender”, “Car ownership” và “Income level”

**Subset**

Gender	Classes
Female	Bus
Female	Car
Female	Car
Female	Train
Female	Train

**1B, 2C, 2T**

Entropy	1.522
Gini index	0.640
Classification error	0.600

Gender	Classes
Male	Bus
Male	Bus
Male	Bus
Male	Car
Male	Train

**3B, 1C, 1T**

Entropy	1.371
Gini index	0.560
Classification error	0.400

Car ownership	Classes
0	Bus
0	Bus
0	Train

**2B, 1T**

Entropy	0.918
Gini index	0.444
Classification error	0.333

Income level	Classes
High	Car
High	Car

**2C**

Entropy	0.000
Gini index	0.000
Classification error	0.000

Car ownership	Classes
1	Bus
1	Bus
1	Car
1	Train
1	Train

**2B, 1C, 2T**

Entropy	1.522
Gini index	0.640
Classification error	0.600

Income level	Classes
Low	Bus
Low	Bus

**2B**

Entropy	0.000
Gini index	0.000
Classification error	0.000

Car ownership	Classes
2	Car
2	Car

**2C**

Entropy	0.000
Gini index	0.000
Classification error	0.000

Income level	Classes

**2B, 1C, 3T**

Entropy	1.459
Gini index	0.611
Classification error	0.500

**Gain of Gender based on**

Entropy	0.125
Gini index	0.060
Classification error	0.100

**Gain of Car ownership based on**

Entropy	0.534
Gini index	0.207
Classification error	0.200

**Gain of Income level based on**

Entropy	0.695
Gini index	0.293
Classification error	0.300

**Hình 11. Độ lợi thông tin các thuộc tính còn lại trong Bảng 1**

- ❖ Bảng dưới đây sẽ cho chúng ta thấy độ lợi thông tin cho tất cả 4 thuộc tính trong bảng D. Chúng ta không cần tính “Impurity degree” dựa trên cả 3 độ đo Entropy, Gini index và Classification error, chỉ cần chọn lựa 1 trong 3 độ đo trên.

**Kết quả của lần lặp đầu tiên**

Độ lợi	Gender	Car ownership	Travel cost/Km	Income level
Entropy	0.125	0.534	1.210	0.695
Gini index	0.060	0.207	0.500	0.293
Classification error	0.100	0.200	0.500	0.300

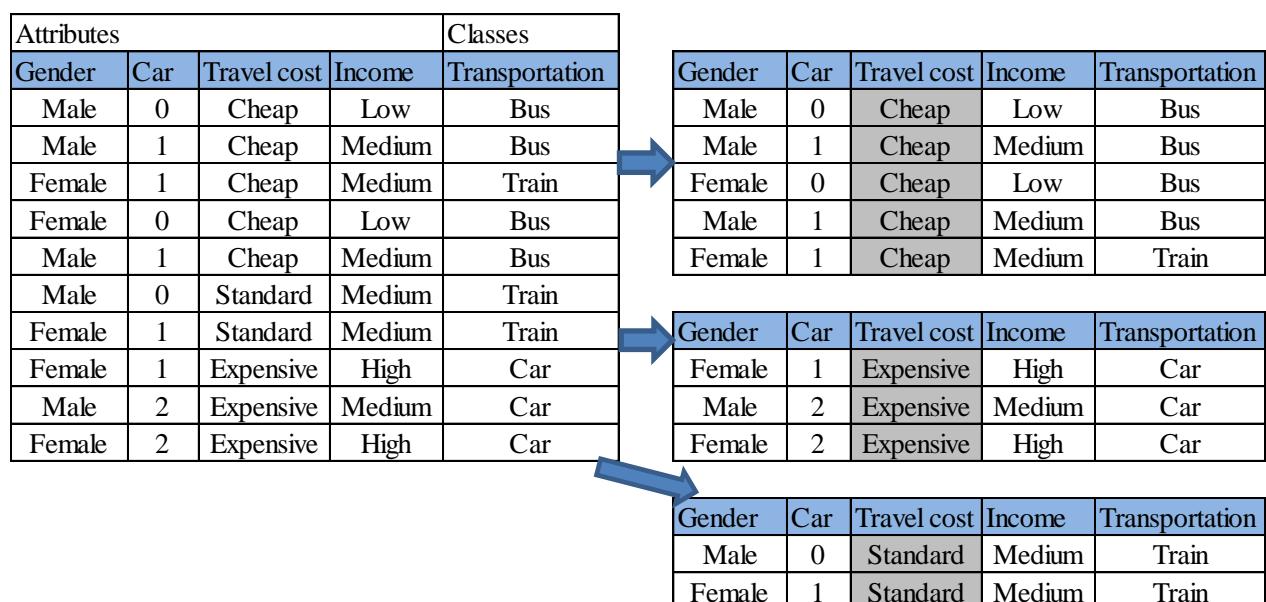
**Bảng 5. Kết quả độ lợi thông tin sau khi phân chia bảng D theo từng thuộc tính**

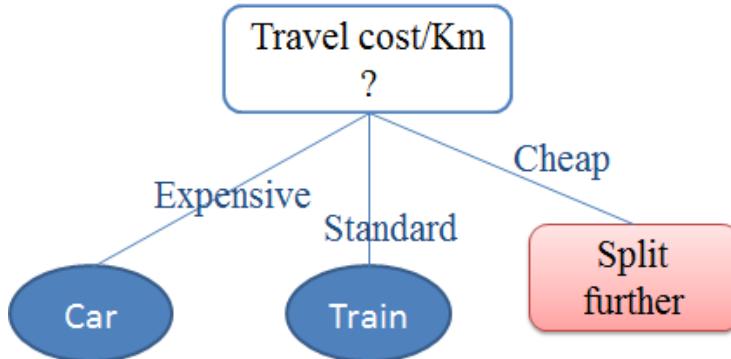
- ❖ Sau khi tính được độ lợi thông tin cho từng thuộc tính, chúng ta sẽ lựa chọn thuộc tính có độ lợi thông tin cao nhất để làm node gốc cho cây quyết định. Bảng trên cho thấy, độ lợi thông tin khi phân lớp theo thuộc tính “Travel cost/km” là cao nhất nên chúng ta sẽ chọn “Travel cost/km” là node gốc của cây.

Travel cost/Km  
?

**Hình 12. Node gốc của cây quyết định sau lần lặp đầu tiên**

- ❖ Bảng D sẽ được phân chia theo thuộc tính “Travel cost/km” như sau:

**Hình 13. Bảng D được phân chia sau lần lặp đầu tiên**



Hình 14. Cây quyết định sau lần lặp đầu tiên

#### 3.6.4.4.2 Những lần phân lớp tiếp theo

Trong phần này sẽ sử dụng lại ví dụ ở phần 3.6.4.4.1

##### 3.6.4.4.2.1 Lần phân lớp thứ 2

- ✓ Sau lần lặp đầu tiên, chúng ta cần cập nhật lại bảng dữ liệu. Khi chúng ta phân chia bảng D theo node gốc “Travel cost/km”, những dòng dữ liệu có giá trị thuộc tính “Travel cost/km” là Expensive và Standard đã là phân lớp thuần khiết, chúng ta không cần sử dụng lại những dòng dữ liệu này để phân lớp. Trong lần lặp này, “Travel cost/km” chỉ còn lại giá trị Cheap, chúng ta sẽ loại bỏ thuộc tính này

Attributes				Classes
Gender	Car ownership	Travel cost/Km	Income level	Transportation mode
Male	0	Cheap	Low	Bus
Female	0		Low	Bus
Male	1		Medium	Bus
Male	1	Cheap	Medium	Bus
Female	1	Cheap	Medium	Train

Bảng dữ liệu trong phân lớp thứ 2

Attributes			Classes
Gender	Car ownership	Income level	Transportation mode
Male	0	Low	Bus
Female	0	Low	Bus
Male	1	Medium	Bus
Male	1	Medium	Bus
Female	1	Medium	Train

Hình 15. Dữ liệu cho lần phân lớp thứ 2

- ✓ Tính “Impurity degree” cho bảng chính

**Bảng dữ liệu trong phân lớp thứ 2**

Attributes			Classes
Gender	Car ownership	Income level	Transportation mode
Male	0	Low	Bus
Female	0	Low	Bus
Male	1	Medium	Bus
Male	1	Medium	Bus
Female	1	Medium	Train

**4B, 1T**

Entropy	0.722
Gini index	0.320
Classification error	0.200

**Hình 16. Tính Impurity degree cho lần phân lớp thứ 2**

- ✓ Tính “Impurity degree” cho các thuộc tính của bảng chính

Gender	Classes
Female	Bus
Female	Train

Car ownership	Classes
0	Bus
0	Bus

Income level	Classes
Low	Car
Low	Car

**1B, 1T**

Entropy	1.000
Gini index	0.500
Classification error	0.500

**2B**

Entropy	0.000
Gini index	0.000
Classification error	0.000

**2B**

Entropy	0.000
Gini index	0.000
Classification error	0.000

Gender	Classes
Male	Bus
Male	Bus
Male	Bus

Car ownership	Classes
1	Bus
1	Bus
1	Train

Car ownership	Classes
0	Bus
0	Bus
0	Train

**3B**

Entropy	0.000
Gini index	0.000
Classification error	0.000

**2B, 1T**

Entropy	0.918
Gini index	0.444
Classification error	0.333

**2B, 1T**

Entropy	0.918
Gini index	0.444
Classification error	0.333

**Gain of Gender based on**

Entropy	0.322
Gini index	0.120
Classification error	0.000

**Gain of Car ownership based on**

Entropy	0.171
Gini index	0.053
Classification error	0.000

**Gain of Income level based on**

Entropy	0.171
Gini index	0.053
Classification error	0.000

**Hình 17. Tính Impurity degree các thuộc tính cho lần phân lớp thứ 2**

- ✓ Độ lợi thông tin khi phân lớp theo thuộc tính “Gender” lớn nhất. Chúng ta sẽ phân chia bảng dữ liệu như sau:

Xây dựng kiến trúc công thông tin tìm việc| Cơ sở và nền tảng xây dựng kiến trúc

Attributes			Classes
Gender	Car ownership	Income level	Transportation mode
Female	0	Low	Bus
Female	1	Medium	Train

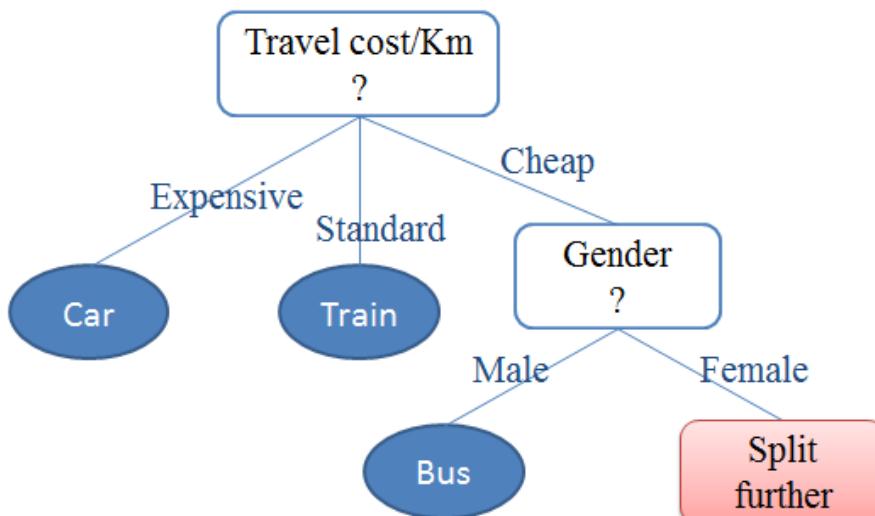
Attributes			Classes
Gender	Car ownership	Income level	Transportation mode
Male	0	Low	Bus
Female	0	Low	Bus
Male	1	Medium	Bus
Male	1	Medium	Bus
Female	1	Medium	Train

Attributes			Classes
Gender	Car ownership	Income level	Transportation mode
Male	0	Low	Bus
Male	1	Medium	Bus
Male	1	Medium	Bus

Hình 18. Bảng dữ liệu sau khi chia theo phân lớp Gender

- ✓ Cây quyết định sau lần phân lớp thứ 2:



Hình 19. Cây quyết định sau lần phân lớp thứ 2

#### 3.6.4.4.2.2 Lần phân lớp thứ 3

Tương tự sau lần phân lớp thứ 2, bảng dữ liệu cho lần phân lớp thứ 3 như sau:

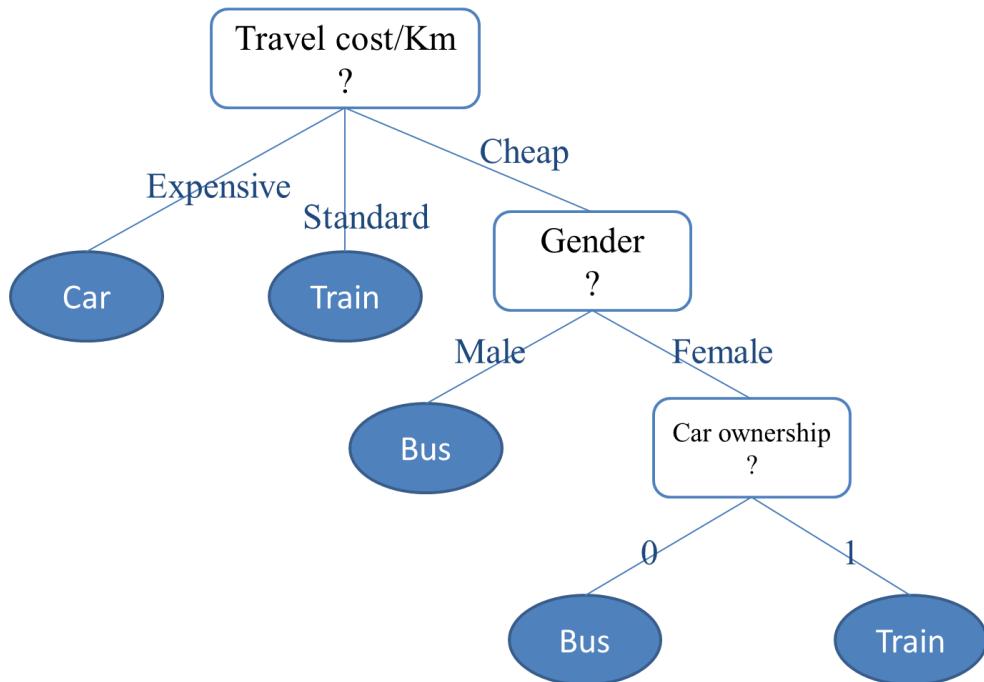
Attributes			Classes
Gender	Car ownership	Income level	Transportation mode
Female	0	Low	Bus
Female	1	Medium	Train

Attributes			Classes
Car ownership	Income level	Transportation mode	
0	Low	Bus	
1	Medium	Train	

**Bảng 6. Bảng dữ liệu cho lần phân lớp thứ 3**

Nếu chia bảng trên theo “Car ownership” hay “Income level” thì bảng dữ liệu được chia đều có duy nhất 1 phân lớp. Chúng ta có thể chia bảng dữ liệu theo 1 trong 2 thuộc tính này.

**Hình 20. Cây quyết định đầy đủ sau 3 lần phân lớp**

### **3.6.4.5 Đánh giá cây quyết định trong lĩnh vực khai thác dữ liệu**

#### **3.6.4.5.1 Điểm mạnh**

- ✓ **Cây quyết định sinh ra các quy tắc hiểu được**

Cây quyết định có thể sinh ra các quy tắc được chuyển đổi sang tiếng Anh hay câu lệnh SQL. Thậm chí đối với những tập dữ liệu lớn làm cho hình dáng cây quyết định lớn và phức tạp thì việc duyệt cây cũng rất dễ dàng. Bất cứ một sự phân lớp hay dự đoán nào đều tương đối minh bạch. Đây là ưu điểm nổi bật của cây quyết định.

- ✓ **Cây quyết định có thể thực thi trong những lĩnh vực hướng quy tắc**

Cây quyết định là sự lựa chọn hoàn hảo cho những lĩnh vực có quy tắc, từ lĩnh vực di truyền đến các những quá trình công nghiệp chứa các quy tắc ẩn, không rõ ràng (underlying rules), dữ liệu lỗi chưa được tiền xử lý khá phức tạp và tối nghĩa. Cây quyết định là một lựa chọn tối ưu khi cần tìm ra những quy tắc ẩn, không rõ ràng trong kho dữ liệu (data warehouse)

- ✓ **Dễ dàng tính toán trong khi phân lớp**

Những thuật toán xây dựng cây quyết định thường tạo ra cây với số phân nhánh thấp và kiểm tra đơn giản tại từng node. Những thuật toán này thường kiểm tra bằng cách so sánh số, xem xét phần tử của một tập hợp hay các phép nối đơn giản, những thao tác này sẽ được chuyển thành các toán hàm logic và số nguyên, đây là những toán hạng thực thi nhanh và chi phí tối ưu. Trong môi trường thương mại, các mô hình dự đoán này được sử dụng để phân lớp hàng triệu, thậm chí hàng tỷ bản ghi, bởi vậy có thể nói đây là một ưu điểm quan trọng của cây quyết định.

- ✓ **Cây quyết định xử lý với cả thuộc tính liên tục và thuộc tính rời rạc**

Các thuộc tính liên tục hay rời rạc đều có thể xử lý bằng cây quyết định. Tuy nhiên, thuộc tính liên tục cần nhiều tài nguyên tính toán hơn, được phân chia bằng việc chọn ra một ngưỡng trong tập các giá trị đã được sắp xếp của thuộc tính đó.

- ✓ **Thể hiện rõ ràng những thuộc tính quan trọng nhất cho việc dự đoán phân lớp**

Các thuật toán xây dựng cây quyết định chỉ ra những thuộc tính dùng để phân chia tốt nhất tập dữ liệu đào tạo bắt đầu từ node gốc của cây. Chính vì vậy, chúng ta

có thể thấy được thuộc tính nào là quan trọng, có mức độ phụ thuộc lớn cho việc dự đoán hay phân lớp

#### 3.6.4.5.2 Điểm yếu

Mặc dù có những điểm mạnh nổi bật trên, cây quyết định vẫn không tránh khỏi những điểm yếu. Cây quyết định không thích hợp lăm với những bài toán mục tiêu, như là dự đoán giá trị của thuộc tính liên tục: thu nhập, huyết áp, lãi suất ngân hàng... Bên cạnh đó, cây quyết định cũng khó giải quyết với những dữ liệu thời gian liên tục.

- ✓ **Cây quyết định dễ xảy ra lỗi khi có nhiều phân lớp**

Một số thuật toán chỉ tao tác với những lớp giá trị nhị phân dạng “có/không” hay “đồng ý/từ chối”. Số khác có thể chỉ định các bản ghi vào một số lớp bất kỳ nhưng dễ xảy ra lỗi khi ứng với một phân lớp có số lượng dữ liệu đào tạo nhỏ.

- ✓ **Chi phí tính toán để đào tạo cao**

Điểm này nghe có vẻ mâu thuẫn với điểm mạnh phía trên. Quá trình xây dựng cây quyết định khác đắt về mặt tính toán. Điều này cũng dễ hiểu vì *cây quyết định có nhiều node trong trước khi đi đến node lá cuối cùng*, ứng với từng node, chúng ta cần tính một độ đo (hay tiêu chuẩn phân chia) trên từng thuộc tính. Bên cạnh đó, đối với thuộc tính liên tục, chúng ta còn phải thêm thao tác sắp xếp lại dữ liệu theo thứ tự giá trị của thuộc tính đó. Quá trình này chọn ra những thuộc tính để phân lớp tốt nhất. Một số thuật toán còn sử dụng tổ hợp các thuộc tính kết hợp với nhau có trọng số để phát triển cây. Quá trình cắt tỉa cây chi phí cũng khá cao vì trong quá trình cắt tỉa, nhiều cây con sẽ được tạo ra và so sánh.

#### 3.6.5 Các khái niệm cơ bản về kiến trúc phần mềm<sup>14</sup>

Khái niệm kiến trúc phần mềm của một chương trình máy tính hay một hệ thống tính toán là cấu trúc của các thành phần trong hệ thống đó. Kiến trúc phần mềm bao gồm các phần tử phần mềm, các thuộc tính và mối quan hệ giữa chúng. Ngoài ra, thuật ngữ “kiến trúc phần mềm” cũng đề cập đến các tài liệu kiến trúc phần mềm của một hệ thống, thuận tiện cho việc trao đổi thông tin giữa các thành viên trong một dự án. Kiến trúc phần mềm giúp việc quyết định ở mức cao trong thiết kế phần mềm dễ

<sup>14</sup> Tham khảo <http://www.sei.cmu.edu/architecture/start/moderndefs.cfm>, <http://www.sei.cmu.edu/architecture/start/classicdefs.cfm>, [24]

dàng hơn và cho phép tái sử dụng các thành phần và mẫu thiết kế (design pattern) của các dự án.

### Software Architecture = {Components, Connectors, Constraint}

- Components: là một phần của hệ thống
- Connectors: là phương tiện trung gian cho các tương tác của các thành phần
- Constraints: là các ràng buộc cho các thành phần (component) và các kết nối (connector) bao gồm các luật về topology, các chuẩn giao diện, các thuộc tính cần thiết.

Năm 1998, tại hội thảo quốc tế đầu tiên về kiến trúc phần mềm, phân loại một số khía cạnh về kiến trúc phần mềm:

- Structure models: bao gồm các thành phần phần mềm (components), sự kết nối giữa các components đó và thường kèm theo một số khía cạnh sau: sự cấu hình (configuration), các ràng buộc (constraint), semantics, các phân tích (analysis)
- Framework models: cung cấp mô hình giải quyết các bài toán kiến trúc phần mềm cụ thể (domain-specific). Ví dụ như CORBA (Common Object Request Broker Architecture)

*Giải quyết bài toán kiến trúc trong hệ thống thông tin về tìm kiếm việc làm cũng với mục đích xây dựng một Framework cụ thể muốn hướng tới, tạm gọi là JobZoom Framework.*

#### ❖ **Tầm quan trọng của kiến trúc phần mềm:**

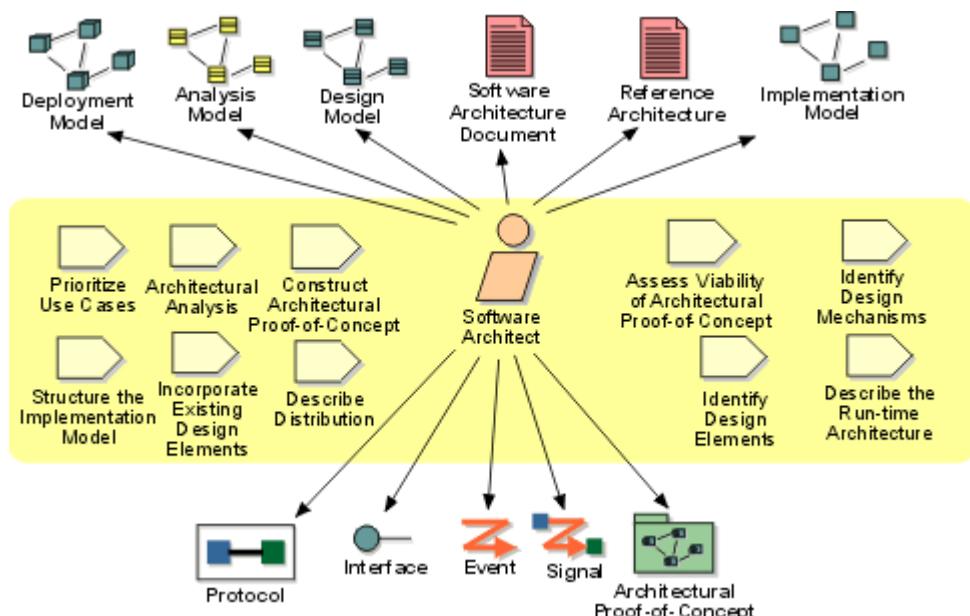
- *Hỗ trợ việc giao tiếp:* Hỗ trợ việc giao tiếp với các thành viên trong dự án và kiến trúc phần mềm tái hiện một vẻ bề ngoài trừu tượng của hệ thống. Với sự trừu tượng hóa hệ thống với các khái niệm dễ hiểu, những thành viên trong dự án sẽ chỉ cần vận dụng các kiến thức cơ bản của mình vào hệ thống.

- *Giúp ra quyết định sớm hơn:* việc ra quyết định được thực hiện sớm hơn. Kiến trúc phần mềm biểu thị các quyết định thiết kế dành cho hệ thống. Như vậy các đội tham gia phát triển, triển khai, kiểm thử và bảo trì phần mềm cũng như các nhóm người dùng và các cấp quản lý sẽ có cái nhìn tổng quan hơn cũng như sớm hơn về hệ thống ngay từ khi nó còn sơ khai. Mỗi đội đó sẽ có các đóng góp ý kiến của mình, các đề xuất cũng như các phản bác của mình khi mọi chuyện chưa quá muộn. Nếu không

có quyết định sớm, thì khi phần mềm đã được xây dựng hoàn chỉnh hoặc khá hoàn chỉnh mà đột ngột xuất hiện các yêu cầu thay đổi từ phía nhóm này hoặc nhóm khác, ngoài việc gây trì trệ cho tiến độ công việc mà còn có thể gây ra tâm lý căng thẳng, mâu thuẫn giữa các đội tham gia trong dự án.

- *Tính khả chuyển cho hệ thống:* kiến trúc phần mềm không phụ thuộc vào một ngôn ngữ cụ thể nào cả mà chỉ tuân theo một số chuẩn của các ngôn ngữ đặc tả nó. Ngoài ra, kiến trúc phần mềm khi được xây dựng cho một hệ thống, nó tạo thành một mô hình có sự gắn kết tương đối với hệ thống. Kiến trúc phần mềm còn chỉ ra cách thức mà phần mềm làm việc với hệ thống. Do vậy, khi ta muốn chuyển phần mềm sang làm việc ở các hệ thống khác có những điểm tương đồng nhất định với hệ thống cũ thì phần mềm này cũng sẽ có các thuộc tính chất lượng và các yêu cầu chức năng được đảm bảo là không quá khác so với khi tồn tại ở hệ thống cũ.

*Tóm lại,* kiến trúc phần mềm ra đời khoảng hơn 30 năm. Vài thập niên trước, người ta đã thấy nó là công việc có ý nghĩa trong công nghệ phần mềm. Kiến trúc sư phần mềm đóng vai trò quan trọng trong việc hình thành một giải pháp nhằm vào các mục đích kinh doanh và công nghệ thông tin của doanh nghiệp.

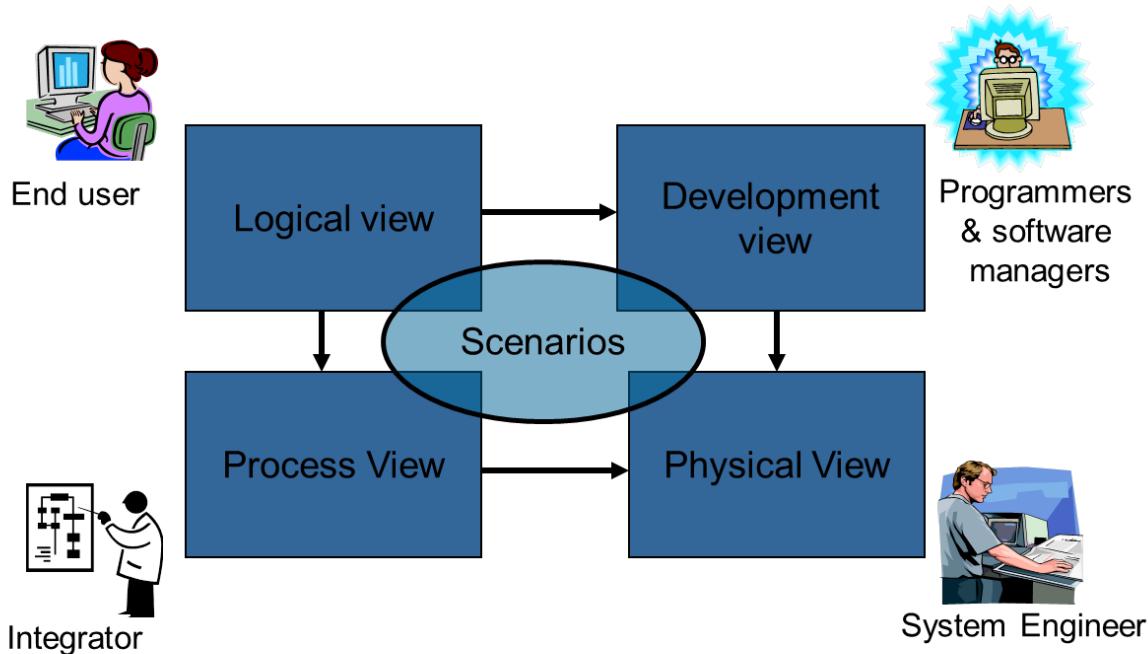


Hình 21. Các khái niệm cơ bản về kiến trúc phần mềm

### 3.6.6 4+1 Architecture View Model<sup>15</sup>

Trong bài toán về xây dựng kiến trúc của một hệ thống phần mềm: đòi hỏi nhà kiến trúc sư phần mềm (software architect) phải làm việc độc với các bên liên quan (stackholder) bao gồm: người sử dụng đầu cuối (end-user), lập trình viên (developer), kỹ sư hệ thống (system engineer), quản trị dự án phần mềm (project manager), v.v... để giải quyết những những góc độ khác nhau của kiến trúc hệ thống (tạm gọi Views): chức năng hệ người dùng đầu cuối, giải pháp cho người phát triển, giải pháp tích hợp, triển khai hệ thống, giải pháp hiệu năng...

4+1 Architecture View được thiết kế bởi Philippe Kruchten. Mô hình cho phép mô tả các góc độ khác nhau của bài toán kiến trúc phần mềm qua các Views khác nhau:



Hình 22. Mô hình 4+1 View Model

❖ **Scenarios (Use Case view):** thể hiện các vấn đề và các giải pháp liên quan đến chức năng tổng quát của hệ thống.

*Đối tượng:* tất cả các stakeholders của hệ thống, bao gồm cả người dùng cuối.

*Mô tả:* diễn tả các chức năng quan trọng, trung tâm của hệ thống. Mô tả những actors và các use cases của hệ thống.

<sup>15</sup> Tham khảo [12], [21]

❖ **Logical view:** thể hiện các vấn đề liên quan đến cấu trúc thiết kế của hệ thống, mô tả thiết kế tổ chức của hệ thống thành các hệ thống con, thành phần, lớp, giao diện, và làm thế nào mà các yếu tố này kết hợp với nhau để tạo ra các chức năng đã miêu tả trong phần Use-Case View. Ngoài ra, Logical View cũng rất quan tâm đến những chức năng mà hệ thống cung cấp cho người dùng, thể hiện qua lược đồ Class, lược đồ tuần tự.

*Đối tượng:* Designers.

*Mô tả:* miêu tả phương thức mà các chức năng của hệ thống sẽ được cung cấp. Ngược lại với use case view, Logical view nhìn vào phía bên trong hệ thống; miêu tả các lớp, các đối tượng, quan hệ.

❖ **Process view:** thể hiện các vấn đề liên quan đến việc xử lý giao tiếp và đồng bộ trong hệ thống. Process View quan tâm đến khía cạnh động của hệ thống, giải thích về các quy trình hoạt động của hệ thống và cách mà các thành phần giao tiếp với nhau, tập trung vào các hành vi của hệ thống vào thời điểm hoạt động. Nó minh họa quá trình phân hủy hệ thống thành biểu đồ của các lớp và các hệ thống con bởi một luồng thực thi tác vụ, một tiến trình hoạt động của chức năng hệ thống. Process View được thể hiện chi tiết ở lược đồ hoạt động

*Đối tượng:* Integrators.

*Mô tả:* chia hệ thống thành các tiến trình (process) có thể được thực thi song song, hướng nhìn này cũng phải quan tâm đến vấn đề giao tiếp và đồng bộ hóa các tiến trình đó.

❖ **Deployment view:** thể hiện các vấn đề liên quan đến việc triển khai hệ thống. Deployment View mô tả cách mà các tiến trình thực thi tác vụ được cấp phát cho phần cứng bên dưới hệ thống, môi trường thực thi, và nó còn thể hiện con đường truyền thông giao tiếp giữa các phần cứng với nhau.

*Đối tượng:* Deployment managers.

*Mô tả:* khai chỉ cho chúng ta sơ đồ triển khai về mặt vật lý của hệ thống, ví dụ như các máy tính cũng như các máy móc và sự liên kết giữa chúng với nhau. Hướng nhìn này cũng bao gồm sự ánh xạ các thành phần của hệ thống vào cấu trúc vật lý.

❖ **Implementation view:** thể hiện các vấn đề liên quan đến việc tổ chức các thành phần trong hệ thống. Implementation view mô tả các thành phần của hệ thống đã được ta nhận ra bởi quan điểm Logical View, thể hiện sự phụ thuộc giữa các thành phần này bên trong hệ thống.

*Đối tượng:* Programmers.

*Mô tả:* chỉ ra khía cạnh tổ chức của các thành phần code, mô tả các lớp và các hệ thống con của ứng dụng.

### 3.6.7 Các đặc tính chất lượng của kiến trúc phần mềm<sup>16</sup>

Đặc tính chất lượng của phần mềm là tiêu chuẩn đánh giá hành vi của hệ thống, các đặc tính chất lượng cung cấp phương pháp để đo chất lượng và độ phù hợp của sản phẩm. Hiện nay có một số đặc tính chất lượng của kiến trúc phổ biến sau:

- ❖ **Tính khả dụng (Usability):** khả năng người dùng có thể học cách tương tác với hệ thống.
- ❖ **Tính hiệu năng (Performance):** là thời gian phản hồi, khả năng thực thi của và hiệu năng của hành vi hệ thống.
- ❖ **Tính mở rộng (Scalability):** khả năng phát triển thêm các thành phần (component) cho hệ thống
- ❖ **Tính có thể thay đổi (Modifiability):** các thành phần (component) có thể thay đổi nhằm chỉnh sửa sai sót, tăng hiệu năng, hoặc thay đổi đáp ứng sự đổi mới của môi trường hay nhu cầu sử dụng.
- ❖ **Tính bảo mật (Security):** kiến trúc cần đảm bảo mức độ bảo mật nhất định do mình xác định
- ❖ **Tính sẵn sàng (Availability):** đảm bảo khả năng truy xuất vào hệ thống hay các thành phần (component) khi cần sử dụng.
- ❖ **Tính tích hợp (Integrability):** khả năng mà các thành phần riêng biệt của hệ thống làm việc một cách chính xác với nhau.
- ❖ **Tính di động (Portability):** khả năng chạy trong các môi trường khác nhau.
- ❖ **Tính kiểm thử (Testability):** hệ thống có khả năng dễ dàng thực hiện kiểm thử bằng các phương pháp khác nhau.

<sup>16</sup> Tham khảo [22]

- ❖ **Tính hỗ trợ (Supportability):** tính hỗ trợ cài đặt, sửa lỗi, khắc phục sự cố trong quá trình sử dụng của người dùng.

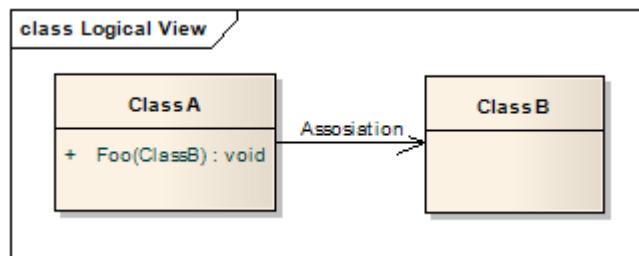
### 3.6.8 Một số lý thuyết cơ bản OOP Design<sup>17</sup>

Có ba loại mối quan hệ (Relationship) chính giữa các Object: association, aggregation, composition. Việc sử dụng đúng relationship rất quan trọng trong việc thiết kế kiến trúc của một hệ thống phần mềm.

#### 3.6.8.1 Association relationship

##### Weak association

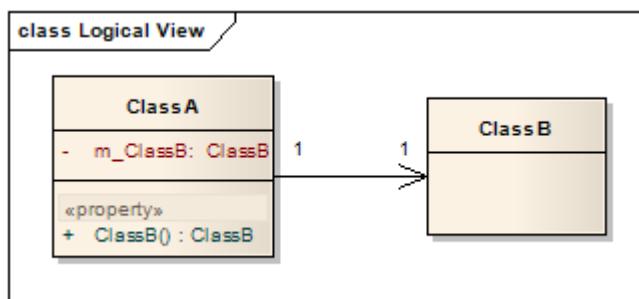
Class A được liên kết với Class B thông qua phương thức trong Class A có include parameter đầu vào là một instance của Class B



Hình 23. Week association

##### Strong association

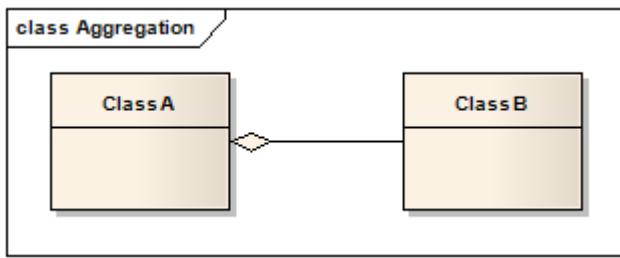
Class A khai báo sử dụng Class B như thuộc tính của mình



Hình 24. Strong association

<sup>17</sup> Tham khảo <http://aviadezra.blogspot.com/2009/05/uml-association-aggregation-composition.html>

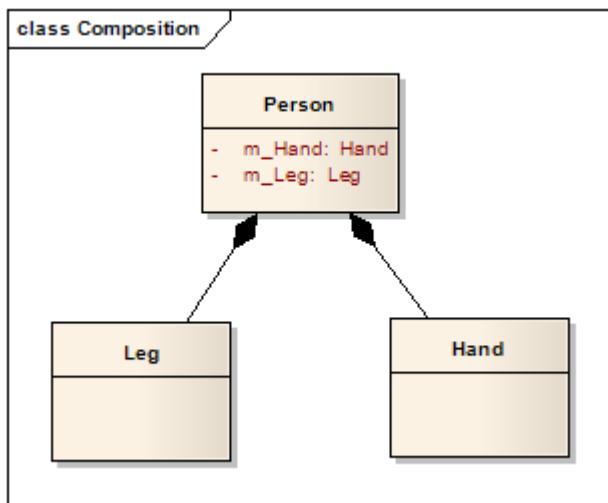
### 3.6.8.2 Aggregation relationship



Hình 25. Aggregation relationship

Đôi khi một class bao gồm một số class thành phần (component class). Đây là loại đặc biệt của mối quan hệ được gọi là sự tập hợp (*aggregation*). Quan hệ giữa các thành phần với class tạo nên quan hệ bộ phận-tổng thể (*part-whole*). Như ở hình trên, Class B là bộ phận của Class A.

### 3.6.8.3 Composition relationship



Hình 26. Composition relationship

Một cấu thành (composite) là một loại aggregation chặt chẽ hơn. Mỗi thành phần trong một composite chỉ có thể thuộc một tổng thể. Trong ví dụ trên, con người được cấu tạo từ nhiều thành phần trong đó có tay và chân. Biểu tượng cho composite tương tự như cho aggregation, ngoại trừ hình thoi màu đen.

### 3.6.8.4 Inheritance<sup>18</sup>

Đặc tính này cho phép một đối tượng có thể có sẵn các đặc tính mà đối tượng khác đã có thông qua kế thừa. Điều này cho phép các đối tượng chia sẻ hay mở rộng các đặc tính sẵn có mà không phải tiến hành định nghĩa lại.

### 3.6.8.5 Polymorphism<sup>19</sup>

Thể hiện thông qua việc gửi các thông điệp (*message*). Việc gửi các thông điệp này có thể so sánh như việc gọi các hàm bên trong của một đối tượng. Các phương thức dùng trả lời cho một thông điệp sẽ tùy theo đối tượng mà thông điệp đó được gửi tới sẽ có phản ứng khác nhau. Người lập trình có thể định nghĩa một đặc tính (chẳng hạn thông qua tên của các phương thức) cho một loạt các đối tượng gần nhau nhưng khi thi hành thì dùng cùng một tên gọi mà sự thi hành của mỗi đối tượng sẽ tự động xảy ra tương ứng theo đặc tính của từng đối tượng mà không bị nhầm lẫn.

### 3.6.8.6 Abstraction<sup>20</sup>

Đây là khả năng của chương trình bỏ qua hay không chú ý đến một số khía cạnh của thông tin mà nó đang trực tiếp làm việc lên, nghĩa là nó có khả năng tập trung vào những cốt lõi cần thiết. Mỗi đối tượng phục vụ như là một "động tử" có thể hoàn tất các công việc một cách nội bộ, báo cáo, thay đổi trạng thái của nó và liên lạc với các đối tượng khác mà không cần cho biết làm cách nào đối tượng tiến hành được các thao tác. Tính chất này thường được gọi là sự trừu tượng của dữ liệu. Tính trừu tượng còn thể hiện qua việc một đối tượng ban đầu có thể có một số đặc điểm chung cho nhiều đối tượng khác như là sự mở rộng của nó nhưng bản thân đối tượng ban đầu này có thể không có các biện pháp thi hành. Tính trừu tượng này thường được xác định trong khái niệm gọi là lớp trừu tượng hay lớp cơ sở trừu tượng.

### 3.6.8.7 Encapsulation & information hiding<sup>21</sup>

Tính chất này không cho phép người sử dụng các đối tượng thay đổi trạng thái nội tại của một đối tượng. Chỉ có các phương thức nội tại của đối tượng cho phép thay

<sup>18</sup> [http://vi.wikipedia.org/wiki/Lập\\_trình\\_hướng đối\\_tượng](http://vi.wikipedia.org/wiki/Lập_trình_hướng đối_tượng)

<sup>19</sup> [http://vi.wikipedia.org/wiki/Lập\\_trình\\_hướng đối\\_tượng](http://vi.wikipedia.org/wiki/Lập_trình_hướng đối_tượng)

<sup>20</sup> [http://vi.wikipedia.org/wiki/Lập\\_trình\\_hướng đối\\_tượng](http://vi.wikipedia.org/wiki/Lập_trình_hướng đối_tượng)

<sup>21</sup> [http://vi.wikipedia.org/wiki/Lập\\_trình\\_hướng đối\\_tượng](http://vi.wikipedia.org/wiki/Lập_trình_hướng đối_tượng)

đổi trạng thái của nó. Việc cho phép môi trường bên ngoài tác động lên các dữ liệu nội tại của một đối tượng theo cách nào là hoàn toàn tùy thuộc vào người lập trình. Đây là tính chất đảm bảo sự toàn vẹn của đối tượng.

### 3.6.9 Design Patterns<sup>22</sup>

Design Pattern là những thiết kế đã được sử dụng và được đánh giá tốt giúp giải quyết những vấn đề thiết kế thường gặp, Design pattern chú trọng việc giúp cho bản thiết kế có tính uyển chuyển, dễ nâng cấp, thay đổi.

#### 3.6.9.1 Vai trò của design pattern:

- Cung cấp phương pháp giải quyết những vấn đề thường gặp trong thực tế đã được đánh giá, kiểm nghiệm.
- Là biện pháp tái sử dụng tri thức các chuyên gia phần mềm.
- Hình thành kho tri thức, ngữ vựng trong giao tiếp giữa những người làm phần mềm
- Giúp người tìm hiểu về design pattern nắm vững hơn đặc điểm của ngôn ngữ lập trình, nhất là lập trình hướng đối tượng.

#### 3.6.9.2 Khả năng ứng dụng của design pattern

- Tìm kiếm đối tượng: Chúng ta thường gặp khó khăn trong việc phân chia hệ thống thành một tập hợp các đối tượng hoạt động hiệu quả, việc sử dụng các mẫu thiết kế giúp đưa ra những đối tượng thường gặp trong những trường hợp thiết kế tương tự đã gặp trước đây.
- Xác định số lượng và kích thước đối tượng: trong trường hợp hệ thống cần ràng buộc số lượng xác định đối tượng đang hoạt động hoặc người thiết kế đang băn khăn về việc nén tập trung một số chức năng nào đó vào trong một đối tượng hay tách ra thành nhiều đối tượng.
- Xác định interface và hiện thực (implement) của đối tượng. “Program to an interface, not an implementation”: lập trình bắt đầu từ interface sau đó hiện thực chúng.

---

<sup>22</sup> Tài liệu tham khảo: [16], [17], [18]

- Giúp thiết kế theo hướng tái sử dụng và linh động bằng cách sử dụng mối quan hệ giữa các đối tượng (tính bao đóng, tính thừa kế...) một cách phù hợp và thiết kế theo hướng tiên đoán trước các thay đổi trong tương lai

### **3.6.9.3 Phân loại Object Oriented Design Patterns**

- Mỗi nhóm design pattern có các pattern về lớp (class patterns) và pattern về đối tượng (object patterns)
- **Class patterns** dựa trên mối quan hệ thừa kế giữa các lớp, mối quan hệ này là tĩnh (xác định tại thời điểm dịch), do đó class patterns thích hợp cho hệ thống không cần thay đổi động trong thời gian chạy
- **Object patterns** dựa trên mối quan hệ giữa các đối tượng, do đó có thể thay đổi ở thời điểm chạy

### **3.6.9.4 Design patterns có ba nhóm chính:**

- **Structural:** Cung cấp cơ chế xử lý những lớp không thể thay đổi (lớp thư viện của third party...), ràng buộc muộn (lower coupling) và cung cấp các cơ chế khác để kế thừa
- **Creational:** Khắc phục các vấn đề khởi tạo đối tượng, hạn chế sự phụ thuộc vào nền tảng (platform).
- **Behavioral:** Che giấu hiện thực của đối tượng, che giấu giải thuật, hỗ trợ việc thay đổi cấu hình đối tượng một cách linh động.

### **3.6.9.5 Nhóm Structural patterns**

- Nhóm này tập trung giải quyết các vấn đề về kết hợp các lớp, các đối tượng thành một kiến trúc lớn hơn
- Các mẫu cấu trúc lớp (structural class patterns) sử dụng kế thừa nhằm kết hợp các class hay các interface với nhau. Tương tự quá trình đa thừa kế, một lớp thừa kế từ nhiều lớp cha hay interface sẽ mang đặc điểm của tất cả các lớp cha hay interface gộp lại.
- Các mẫu cấu trúc đối tượng (structural object patterns) tập trung vào việc kết hợp các đối tượng để thực hiện một chức năng cụ thể

- Structural patterns bao gồm: Adapter, composite, proxy, decorator, façade và flyweight

#### 3.6.9.6 Nhóm *Creational patterns*

- Creational design patterns giúp xây dựng hệ thống linh động về mặt khởi tạo, quản lý và sử dụng các đối tượng. Chúng cho phép hệ thống chủ động trong việc xác định đối tượng nào được tạo ra, ai tạo ra đối tượng, cách thức và thời điểm khởi tạo.
- Đặc điểm nổi bật trong creationnal patterns là chương trình cần sử dụng đối tượng không trực tiếp sinh ra đối tượng mà nhờ các phần tử trung gian để tăng độ linh động
- Class creational patterns sử dụng đặc điểm thừa kế để thay đổi class sẽ được sử dụng để sinh ra đối tượng.
- Object creational patterns truyền quá trình khởi tạo đối tượng cho một đối tượng khác.
- Creational patterns bao gồm Abstract Factory, Factory Method, Prototype, Singleton và Builder.

#### 3.6.9.7 Nhóm *Behavioral patterns*

- Nhóm behavioral tập trung vào giải thuật và sự phân bổ công việc giữa các object.
- Behavioral class patterns sử dụng kế thừa để chuyển giao công việc giữa các class.
- Object patterns sử dụng tính đa hình và bao đóng để truyền công việc từ đối tượng này sang đối tượng khác.
- Behavioral patterns bao gồm Chain of Responsibility, Template method, Strategy, Command, State, Observer

### 3.7 Các vấn đề cần giải quyết khi thực hiện kiến trúc lưu trữ và so khớp thông tin

#### 3.7.1 Nắm bắt xu hướng lưu trữ thông tin

##### 3.7.1.1 Thông tin được lưu trữ dựa trên dạng tag

Tag là một từ khóa không có cấu trúc (non-hierarchical) hay một thuật ngữ nhằm chỉ một mẫu thông tin. Loại dữ liệu biến đổi này giúp miêu tả một mục tin và cho phép người sử dụng tìm lại mục đó bằng cách trình duyệt hay dò tìm. Lưu trữ thông tin dưới dạng tag, giúp chúng tôi có thể dễ dàng phân loại các tag với nhau.

*Công việc khó khăn mà kiến trúc phải thực hiện được đó là: từ khối dữ liệu lớn của người dùng, làm sao chuyển các cấu trúc này về dạng tag, nói khác đi, đó chính là việc mapping dữ liệu người dùng, sau đó lưu trữ các thông tin này dưới dạng tag để xử lý ở các giai đoạn tiếp theo.*

##### 3.7.1.2 Chuyển dạng phi cấu trúc tag về dạng có cấu trúc

Tìm kiếm là một thao tác rất quan trọng đối với nhiều ứng dụng tin học. Tìm kiếm nói khác đi là việc thu thập một số thông tin nào đó từ một khối lượng thông tin lớn đã được lưu trữ trước đó. Sau khi thông tin được phân rã thành tag, các tag này sẽ được phân loại có cấu trúc theo dạng cây để dễ dàng trong việc tìm kiếm và so khớp thông tin. Mặc dù, việc lưu trữ thông tin dưới dạng tag giúp dễ dàng trong việc phân loại dữ liệu, tuy nhiên, để cải thiện được tốc độ của việc tìm kiếm và so khớp, thì các tag thông tin này phải được tổ chức theo dạng cấu trúc – lưu trữ và phân loại các tag có mối quan hệ với nhau theo cấu trúc cây, hiệu quả hơn trong việc tìm kiếm và đặc biệt hơn là việc so khớp thông tin.

#### 3.7.2 Độ tương quan giữa các tag

Nếu so sánh với các phương pháp phân loại dữ liệu trước đây (categories/folder), tag là cách đơn giản, linh hoạt và là một công cụ phân loại mạnh mẽ. Tuy nhiên, *tag có một số khuyết điểm đáng kể đó là chúng không thể cung cấp thông tin về ngữ nghĩa của chúng*, ngữ nghĩa của các tag có thể được bao gồm: Tính nhiều nghĩa (cùng một từ có thể tham khảo các khái niệm khác nhau), tính đồng nghĩa (khái niệm

tương tự có thể được chỉ ra bằng cách sử dụng từ ngữ khác nhau), các hình thức từ vựng khác nhau (hình thức danh từ khác nhau, chia động từ khác nhau, từ viết tắt, ngôn ngữ khác nhau), lỗi chính tả là một số vấn đề mà phát sinh khi sử dụng các tag. Ví dụ, tag "orange" có thể liên quan đến trái cây hay màu sắc, và tag "movie" và "film" thường được mô tả cùng một khái niệm. Thiếu sự phân biệt ngữ nghĩa này dẫn đến các kết nối không phù hợp giữa các mục, làm cho chúng khó tìm kiếm và duyệt.

*Như vậy, giữa hai tag khác nhau sẽ tồn tại một mối quan hệ nhất định. Đó là một con số thể hiện mức độ tương quan giữa tag A và tag B, thường có giá trị từ -1 đến 1, hoặc giữa 0 và 1, trong đó giá trị 1 biểu hiện tính tương đồng rất cao, và giá trị 0 có nghĩa ít hoặc không có sự liên hệ nào. Đứng dưới mỗi góc nhìn khác nhau, mức độ tương quan giữa các tag cũng có giá trị khác nhau, đây là vấn đề cần giải quyết khi xác định độ tương quan giữa các tag (sự phụ thuộc độ tương quan vào góc nhìn sẽ được ví dụ dễ hiểu hơn trong phần 4.2)*

### 3.7.3 Hệ thống so khớp linh hoạt

Một khi thông tin đã được lưu trữ theo dạng taxonomy, hệ thống có thể dễ dàng so khớp thông tin giữa các đối tượng bất kỳ với nhau. Tuy nhiên, để tăng sự linh hoạt của việc so khớp và khắc phục tình trạng các tag có độ tương quan nhưng không thể khớp với nhau, đó chính là vấn đề tiếp theo mà kiến trúc cần giải quyết. Chính việc kết hợp giữa hệ thống so khớp cùng với độ tương quan giữa các tag đã góp phần làm cho hệ thống so khớp được linh hoạt và uyển chuyển hơn, đem lại kết quả tốt hơn khi so khớp thông tin.

### 3.7.4 Gợi ý người sử dụng đăng thông tin nhu cầu và “thông tin đáp ứng nhu cầu” bằng hệ thống so khớp

Việc so khớp không chỉ dừng lại ở việc cung cấp mức độ so khớp giữa các đối tượng với nhau mà còn giúp cho việc gợi ý cải thiện “nhu cầu” và “thông tin đáp ứng nhu cầu”.

*Có thể giải quyết được việc này bằng hệ thống so khớp?*

Khi so khớp giữa nhu cầu và “thông tin đáp ứng nhu cầu cụ thể”, hệ thống so khớp sẽ cung cấp mức độ so khớp: thuộc tính nào trong “thông tin đáp ứng nhu cầu”

thoả mãn nhu cầu được đặt ra và thoả mãn bao nhiêu phần trăm, hệ thống so khớp còn đưa ra những thuộc tính nào chưa được thoả mãn. Từ tập các thuộc tính chưa được thoả mãn này, kiến trúc có thể đưa ra các thuộc tính gợi ý cho người dùng cải thiện “sản phẩm” của mình, đáp ứng tốt hơn nhu cầu của khách hàng hay đổi tượng phát sinh nhu cầu.

### 3.7.5 Gợi ý người sử dụng đăng thông tin nhu cầu và “thông tin đáp ứng nhu cầu” bằng cây quyết định

Không chỉ dừng lại ở vấn đề gợi ý cải thiện “thông tin đáp ứng nhu cầu”, việc ứng dụng cây quyết định vào kiến trúc sẽ giúp gợi ý cải thiện “thông tin đáp ứng nhu cầu” đối với loại nhu cầu có liên quan đến sản phẩm đó. Ví dụ: Nhu cầu của bạn muốn mua một bộ ghế salon, bạn không quan tâm đến việc mình sẽ lựa chọn một nhãn hiệu cụ thể nào đó, công việc của kiến trúc là gợi ý cho bạn, làm sao đăng tải thông tin tìm được một bộ ghế salon vừa ý bạn nhất.

Bên cạnh đó, kiến trúc phải gợi ý cho nhà sản xuất đăng thông tin cho “thông tin đáp ứng nhu cầu của họ”, làm sao đăng thông tin một cách nhất quán, làm sao bán được sản phẩm được tiêu thụ nhanh chóng. Ví dụ: khi bạn muốn đăng bán một bộ loa, kiến trúc sẽ xem xét tất cả những nhu cầu mua những bộ loa trên thị trường, từ đó phân tích các thuộc tính, gợi ý cho bạn những thuộc tính phù hợp, giúp bạn bán đi sản phẩm của mình.

Để giải quyết các vấn đề nói trên, việc áp dụng cây quyết định là phương pháp tối ưu để xác định mức độ phụ thuộc, liên quan giữa các thuộc tính với nhau. Cây quyết định còn giúp cho việc xây dựng kiến trúc taxonomy.

### 3.7.6 Kiến trúc dễ dàng triển khai, mở rộng

Thông tin về nhu cầu và thông tin đáp ứng nhu cầu được lưu trữ dưới dạng tag, từ các tag sẽ được xây dựng thành hệ thống taxonomy. Chúng tôi cung cấp một phương pháp xây dựng và phân lớp các tag theo việc phân chia các tag này theo từng chủ đề cụ thể, các tag có liên quan sẽ thuộc về cùng chủ đề tương ứng. Việc xây dựng cây quyết định hay hệ thống so khớp cũng khá hoàn thiện, tuy nhiên trong vấn đề xây dựng kiến trúc, chúng tôi phải xem xét các yếu tố sau:

- Làm sao kiến trúc nghiệp vụ được xây dựng ra có thể dễ dàng áp dụng được khi các lập trình viên có nhu cầu sử dụng.
- Làm sao có thể mở rộng, tối ưu việc xây dựng cây, thuật toán khai thác dữ liệu, cách hoạt động của việc so khớp? Kiến trúc phải hỗ trợ lập trình viên trong việc mở rộng kiến trúc, kế thừa các interface, class của kiến trúc để định nghĩa cho hệ thống nghiệp vụ của mình có những đặc thù riêng: có thể sửa lại cách tính điểm của hệ thống so khớp, cách xây dựng cây, thuật toán khai thác dữ liệu...

### 3.7.7 Kiến trúc đảm bảo hiệu năng khi áp dụng cây quyết định và hệ thống so khớp

Hệ thống so khớp và cây quyết định làm việc rất tốn nhiều thời gian và chi phí tính toán, làm sao đảm bảo hiệu năng khi xây dựng cây quyết định và hệ thống so khớp. Đây cũng là một trong những vấn đề trọng tâm chúng tôi cần giải quyết khi xây dựng kiến trúc này

## 4 Giải quyết bài toán kiến trúc

Lý thuyết và phương pháp ở trên đã được chúng tôi áp dụng vào việc giải quyết bài toán kiến trúc được đề cập ở phần “**2. Giới thiệu bài toán kiến trúc**”.

Tổ chức thông tin và sự linh hoạt của hệ thống so khớp được giải quyết bằng việc lưu trữ thông tin về nhu cầu và “thông tin đáp ứng nhu cầu” dưới dạng **tag** kết hợp **taxonomy**. Đồng thời, **taxonomy** và **cây quyết định** sẽ gợi ý cho actor xây dựng cây thông tin nhu cầu và “thông tin đáp ứng nhu cầu” một cách nhanh chóng và hiệu quả. Bài toán 2 được giải quyết bằng việc lưu trữ “độ tương quan giữa các tag” thông qua bảng **SimilarityTerm**. So khớp kết hợp mức độ tương quan giữa các tag nhằm giải quyết bài toán kiến trúc còn lại.

### 4.1 Giải pháp tổ chức thông tin linh hoạt dưới dạng Tag kết hợp với Taxonomy

#### Vấn đề 1: Làm thế nào mô tả thông tin bất kỳ (*structured information*)

Đã đề cập trong bài toán thứ nhất, Trong các phần mềm, thông tin của một đối tượng trong thế giới thực (real object) được mô hình hóa thành các đối tượng, các thực thể trong phần mềm (software object). Có nhiều cách để mô hình hóa (schema mapping) một đối tượng trong thế thực: mỗi quan hệ 1-1 (một đối tượng thực được biểu diễn

bằng một đối tượng duy nhất trong phần mềm, mỗi quan hệ tập hợp (một đối tượng thực được biểu diễn bằng cách cấu thành bởi nhiều đối tượng trong phần mềm), quan hệ kế thừa... Không những chỉ tổ chức thông tin trong phần mềm (software object) mà còn phải tổ chức về mặt lưu trữ các đối tượng đó xuống cơ sở dữ liệu thì cũng có nhiều cách để tổ chức.

### ***Giải pháp vấn đề 1: sử dụng phương pháp tagging mô tả những nội dung thông tin cần thiết thành các thẻ tag.***

Như đã phân tích xu hướng lưu trữ và tìm kiếm thông tin (phần 3.1) và lý thuyết cơ bản về tag (phần 3.6.1), ta nhận thấy đặc điểm quan trọng của tag: vì tag còn được gọi là keyword (metadata) nên không phụ thuộc vào cách tổ chức thông tin có cấu trúc ban đầu của người dùng (structured information).

### ***Vấn đề 2: Làm thế nào tổ chức thông tin linh động để có thể so khớp với nhau và có khả năng khai thác thông tin sau này.***

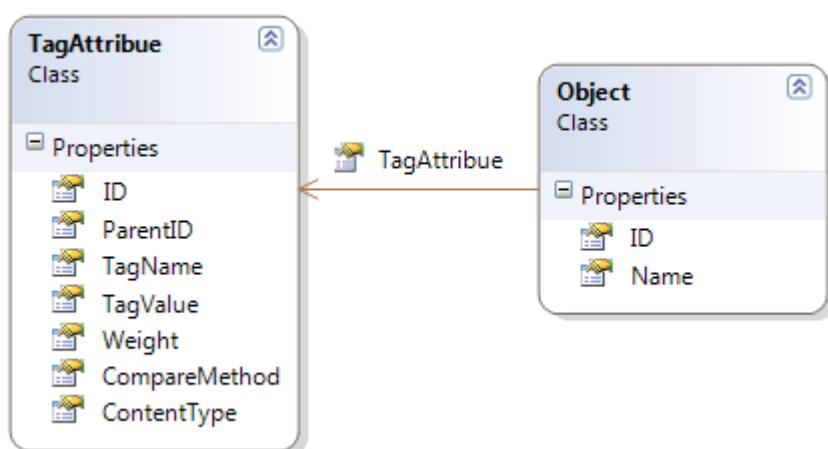
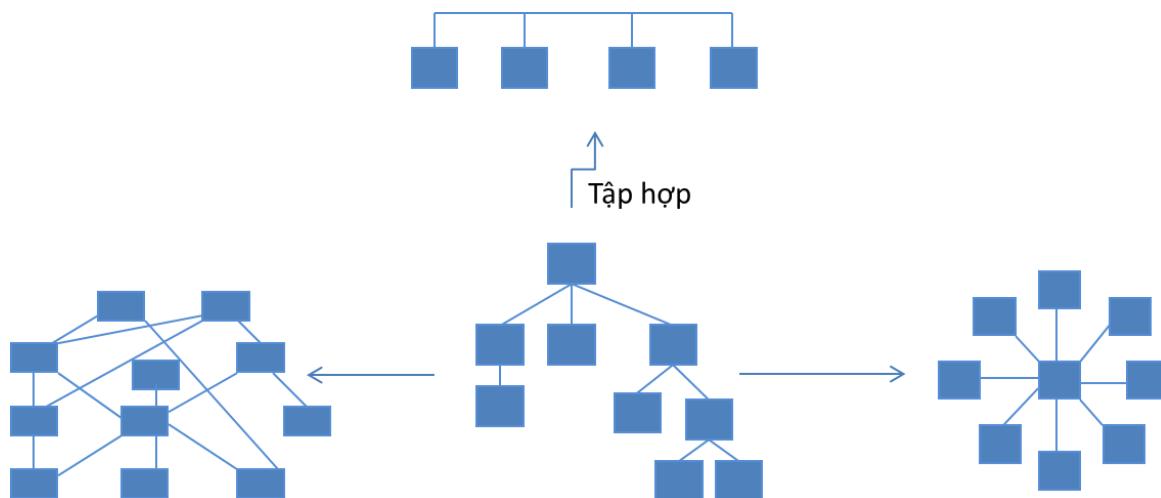
Đối với một bài toán nghiệp vụ thì thường ta sẽ đưa ra giải pháp cho các vấn đề cụ thể, thiết kế và xây dựng cấu trúc lưu trữ thông tin phức tạp để giải quyết cho vấn đề đó, từ đó sẽ tạo ra khó khăn trong việc quản lý thông tin, tìm kiếm và thống kê. Hơn nữa, việc tổ chức thông tin có cấu trúc phức tạp sẽ gây khó khăn đến việc tái sử dụng hay mở rộng hệ thống. Vậy tính linh hoạt của một cấu trúc thông tin là gì? Làm sao tổ chức thông tin có cấu trúc linh hoạt, đánh giá giải pháp tổ chức thông tin linh hoạt. Đó là những vấn đề sẽ được làm rõ trong phần này.

### ***Giải pháp vấn đề 2: sử dụng cấu trúc phân cấp (hierarchical) để tổ chức và lưu trữ thông tin.***

- ❖ Bám sát mục tiêu của bài toán về việc tổ chức thông tin có cấu trúc linh hoạt phải đảm bảo các đặc điểm sau:
  - Cấu trúc thông tin có thể mô tả thông tin đa dạng, nhiều cấu trúc (schema scenarios) của một phần mềm nghiệp đăng tải và so khớp thông tin: riêng lẻ, tập hợp, kế thừa. Thông tin của một đối tượng được mô tả một cách linh hoạt.
  - Có sự quản lý và phân loại thông tin hiệu quả, dễ dàng tìm kiếm và so khớp các đối tượng với nhau.
  - Dễ dàng thay đổi và mở rộng hệ thống tổ chức thông tin sau này.

- Khả năng tái sử dụng hay ứng dụng vào các bài toán hoặc vấn đề tương tự, phù hợp với nhiều lĩnh vực khác nhau.

- ❖ Áp dụng giải pháp tổ chức thông tin theo cây phân cấp (hierarchy) trong các phương pháp phân loại và tổ chức thông tin (taxonomy): cây phân cấp là dạng cấu trúc linh hoạt, trung gian có thể đáp ứng về các schema thông tin linh động của nghiệp vụ đăng tải và so khớp thông tin, đồng thời hierarchy là cấu trúc dễ dàng linh động mở rộng thay đổi sau này: faced, network giúp giải quyết các bài toán chuyên sâu hơn về khai thác thông tin và trí tuệ nhân tạo.

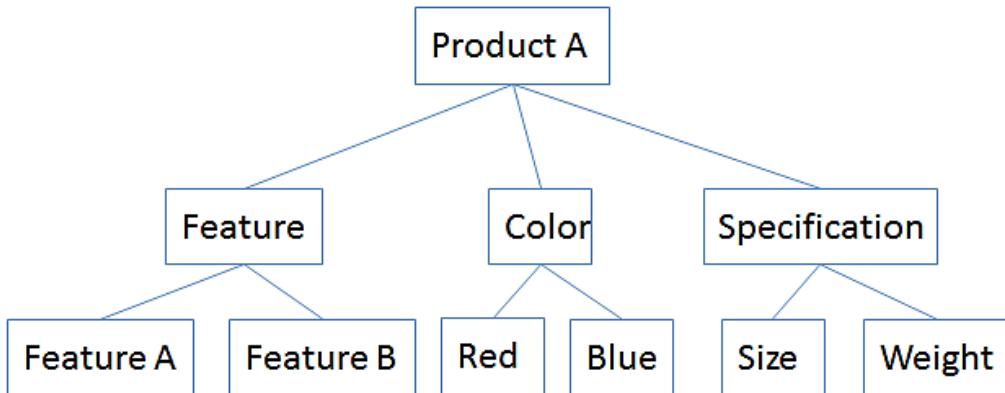


Hình 27. Mô hình tổ chức thông tin theo tag kết hợp taxonomy

- ❖ *Mô tả thông tin object linh hoạt dưới dạng cây*

Dựa trên việc lưu trữ thông tin dưới dạng cấu trúc cây là tiền đề xây dựng kiến trúc, thông tin sẽ được phân rã thành tag, các tag này sẽ được phân loại có cấu trúc

theo dạng cây để dễ dàng trong việc tìm kiếm và so khớp thông tin. Các tag này sẽ liên kết trực tiếp với object giúp bổ sung thông tin cho object, tạo sự linh hoạt trong việc mô tả thông tin cho object. Như đã nói trên, các tag được phân loại theo cấu trúc cây phân lớp Hierarchy Taxonomy, vì thế thông tin của object sẽ được thể hiện theo dạng cấu trúc cây phân cấp.



**Hình 28. Ví dụ tổ chức thông tin có cấu trúc linh hoạt**

- ❖ Vì mục tiêu phân lớp các thông tin đảm bảo khả năng so khớp thông tin theo các tiêu chí, các thuộc tính linh hoạt (classification): đưa ra giải pháp tổ chức thông tin trong cây phân cấp theo 3 cấp (level): Root node (level 1), các Node phân lớp – Classification nodes (level 2), node lá (level 3) kết hợp giải pháp tạo mối tương quan giữa các node lá (similar term), cách tổ chức này giúp thông tin của đối tượng được mô tả linh hoạt hỗ trợ cho việc so khớp dễ dàng hơn [20].

Tổ chức thông tin dạng cây như trên giúp sử dụng kỹ thuật Pivot Transformation trong SQL Server dễ dàng hơn. Ví dụ chúng ta có bảng dữ liệu như sau:

Actor	Tên thông tin đáp ứng nhu cầu	Tên tag
A	Áo thun nam	Tay dài
A	Áo thun nam	Sọc caro
B	Áo thun nam	Tay dài
C	Áo thun nam	Cổ cao

<b>C</b>	Áo thun nữ	Tay dài
<b>D</b>	Áo thun nữ	Có túi
<b>D</b>	Áo thun nữ	Không cỗ
...	...	...

**Bảng 7. Dữ liệu ví dụ trước khi xử lý bằng kỹ thuật Pivot Transformation**

Sau khi xử lý bằng kỹ thuật *Pivot Transformation*, dòng sẽ được hoán đổi thành cột:

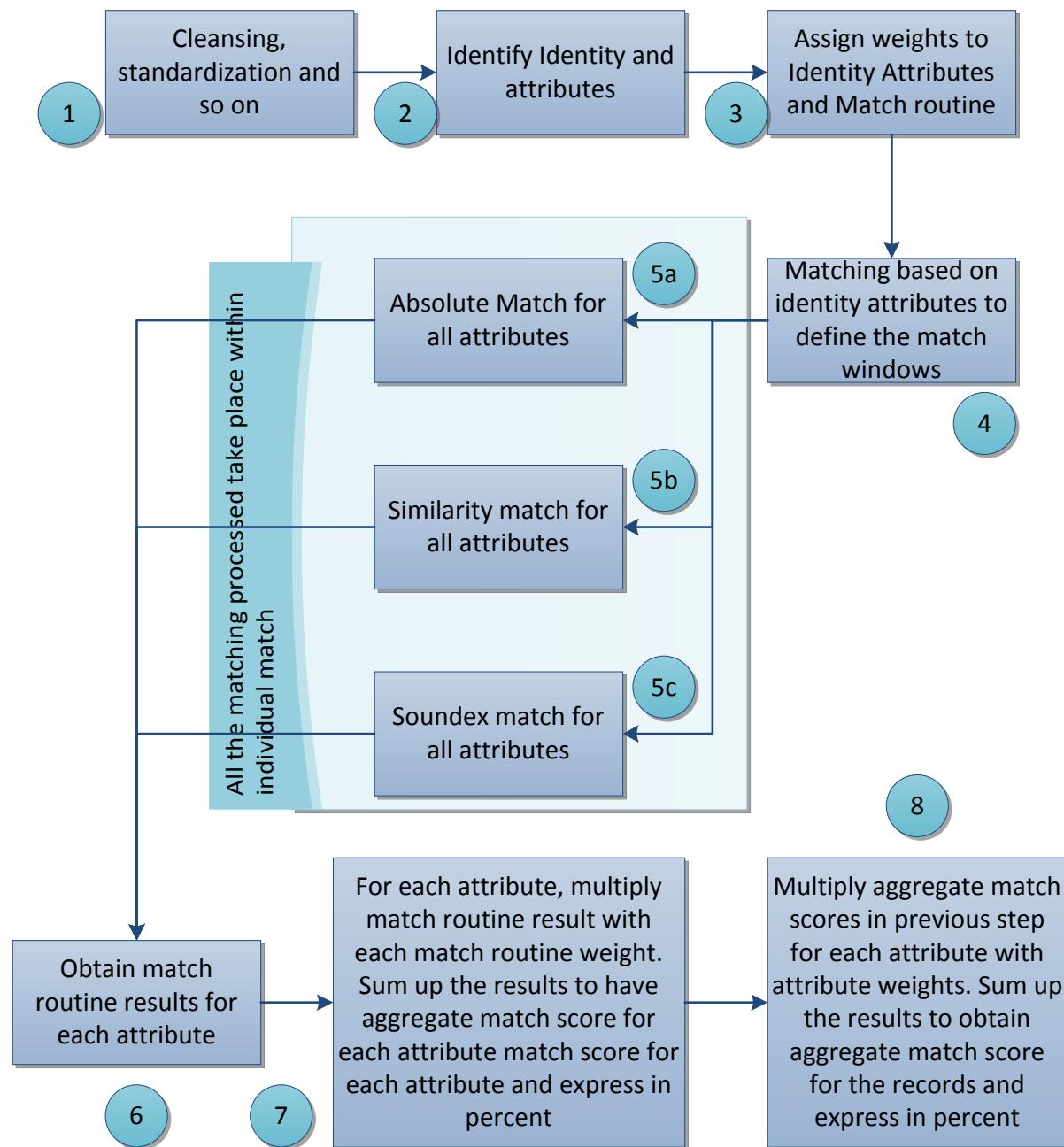
Actor	Thông tin đáp ứng nhu cầu	Tay dài	Sọc caro	Cỗ cao	Có túi	Không cỗ	...
<b>A</b>	Áo thun nam	<b>True</b>	<b>True</b>	False	False	False	...
<b>B</b>	Áo thun nam	<b>True</b>	False	False	False	False	...
<b>C</b>	Áo thun nam	False	False	<b>True</b>	False	False	...
<b>C</b>	Áo thun nữ	<b>True</b>	False	False	False	False	...
<b>D</b>	Áo thun nữ	False	False	False	<b>True</b>	<b>True</b>	...

**Bảng 8. Dữ liệu Bảng 7 sau khi xử lý bằng kỹ thuật Pivot Transformation**

Kỹ thuật *Pivot Transformation* giúp cho việc phân lớp được thực hiện dễ dàng hơn.

Với kiến trúc tổ chức thông tin như trên, đảm bảo tính mở rộng của kiến trúc, khả năng tích hợp vào một hệ thống khác dễ dàng và việc so khớp thông tin giữa các đối tượng khác nhau được thực thi hiệu quả hơn so với mô hình kiến trúc nghiệp vụ thông thường.

## 4.2 Giải pháp so khớp thông tin<sup>23</sup>



**Hình 29. Mô hình giải pháp so khớp thông tin**

❖ **Bước 1:** Làm sạch và chuẩn hoá dữ liệu.

Lấy thông tin về nhu cầu và “thông tin đáp ứng nhu cầu”, chuẩn hoá và làm sạch thông tin chuẩn bị cho việc so khớp.

❖ **Bước 2:** Xác định các thuộc tính cần so khớp

<sup>23</sup> Tham khảo <http://msdn.microsoft.com/en-us/magazine/gg598923.aspx>

Dựa vào những thuộc tính miêu tả nhu cầu để chọn ra những thuộc tính cần lấy ra từ “thông tin đáp ứng nhu cầu”.

❖ **Bước 3:** Xác định “Attribute Weight” (tầm quan trọng) cho từng thuộc tính và phương pháp so khớp cho mỗi kiểu dữ liệu trong khai thác thông tin. Ứng với từng loại kiểu dữ liệu sẽ có những phương pháp so khớp phù hợp, ứng với mỗi phương pháp sẽ có một trọng số “Match Routine Weight” tương ứng

❖ **Bước 4:** Dựa vào các thuộc tính cần so khớp định nghĩa các tiến trình so khớp

❖ **Bước 5:** Ứng với mỗi tiến trình thực hiện các phương pháp so khớp phù hợp cho mỗi kiểu dữ liệu. Có các phương pháp so khớp sau:

*Absolute match:* So khớp tuyệt đối, A phải hoàn toàn khớp với B. Kết quả so khớp giữa A và B là 0% hoặc 100%

*Similarity match:* So khớp có mức độ tương quan, A tương quan B và có một số điểm tương ứng. Kết quả so khớp giữa A và B trong khoảng từ 0% đến 100%

*Soundex match:* So khớp các từ có phát âm tương tự nhau. Kết quả so khớp giữa A và B trong khoảng từ 0% đến 100%

❖ **Bước 6:** Tập hợp kết quả so sánh cho từng thuộc tính

❖ **Bước 7:** Ứng với mỗi kết quả so khớp thuộc tính, kết quả so khớp của từng phương pháp so khớp sẽ được nhân với weight (tầm quan trọng) của mỗi phương pháp, sau đó tổng hợp để đưa ra số điểm cho mỗi thuộc tính. Kết quả được thể hiện dưới dạng phần trăm.

$$\text{Attribute Score (\%)} = \frac{\sum \text{Match Routine Score (\%)} * \text{Match Routine Weight}}{\sum \text{Match Weight}}$$

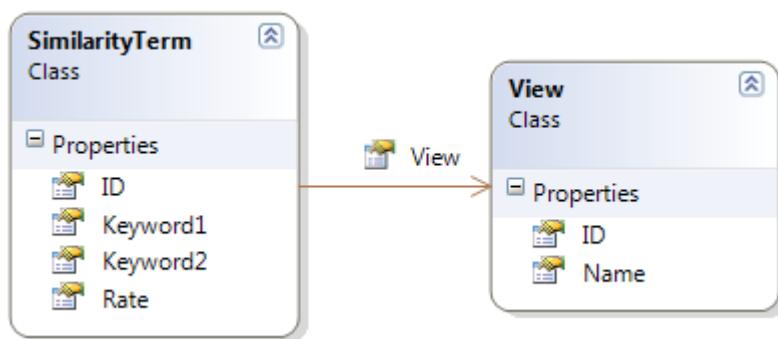
❖ **Bước 8:** Nhân mỗi số điểm tổng hợp ở bước 7 với weight của từng thuộc tính được xác định ở bước 3. Tổng hợp để tính trung bình. Kết quả của bước 8 là mức độ so khớp giữa thông tin nhu cầu và “thông tin đáp ứng nhu cầu”

$$\text{Score (\%)} = \frac{\sum \text{Attribute Score (\%)} * \text{Attribute Weight}}{\sum \text{Attribute Weight}}$$

#### 4.3 So khớp thông tin kết hợp mức độ tương quan giữa các tag

Trong quá trình thực hiện đề tài, khi nghiên cứu phương pháp so khớp thông tin dựa trên các tập thuộc tính Tag của đối tượng, chúng tôi nhận thấy vẫn đề độ tương quan ngữ nghĩa giữa các từ khóa, như đã được đề cập ở phần trên, nhưng do phạm vi

đề tài và thời gian hạn chế nên chúng tôi quyết định không nghiên cứu phương pháp tính độ tương quan ngữ nghĩa của hai từ khóa. Thay vào đó chúng tôi cung cấp giải pháp lưu trữ kết quả tính toán độ tương quan vào bảng SimilerityTerm để truy vấn sử dụng, sau đó xây dựng phương pháp so khớp thông tin kết hợp so sánh ngữ nghĩa để tính mức độ tương quan giữa các tag và cập nhật dữ liệu cho bảng SimilerityTerm. Nhờ vậy kiến trúc đảm bảo khả năng mở rộng và phát triển Semantic web nhằm xây dựng hệ thống so khớp hoàn thiện hơn.



**Hình 30. Class diagram mức độ tương quan giữa các tag theo góc nhìn**

❖ **So khớp linh hoạt theo góc nhìn:**

- Đứng dưới mỗi góc nhìn khác nhau, mức độ tương quan giữa các tag cũng khác nhau. Ví dụ: xét mức độ tương quan giữa tôi và thầy hướng dẫn, đứng dưới góc nhìn là quan hệ gia phả, thì rõ ràng tôi và thầy không có mối quan hệ ruột thịt hay họ hàng với nhau, nói cụ thể hơn là mức độ tương quan giữa tôi và thầy là bằng không, nhưng dưới góc nhìn là quan hệ thầy trò, thì thầy là người đã tận tình chỉ dạy và hướng dẫn chúng tôi từ lúc bắt đầu khoá luận này. Như vậy, khi nhu cầu và “thông tin đáp ứng nhu cầu” cụ thể được mô tả thông qua các thuộc tính, dưới một góc nhìn về một lĩnh vực cụ thể nào đó thì thuộc tính A có thể có mức độ tương quan với thuộc tính B, tuy nhiên, khi xét dưới góc nhìn khác, thì mức độ tương quan này sẽ không còn phù hợp nữa, việc này đòi hỏi giữa các góc nhìn khác nhau, các thuộc tính sẽ có mức độ tương quan thay đổi phù hợp với chính góc nhìn đó. Một ví dụ cụ thể, chúng ta có thuộc tính A lưu trữ giá trị là “quả cam xanh” và thuộc tính B lưu trữ giá trị là “quả cam màu cam”, đứng dưới góc nhìn của một người nội trợ, hai thuộc tính này có mức độ tương quan với nhau, quả cam xanh hay quả cam màu cam đều có thể chế biến món ăn được, tuy nhiên, nếu xét dưới góc nhìn của một người họa sĩ thì lại khác,

việc vẽ “quả cam xanh” hay “quả cam màu cam” đòi hỏi người họa sĩ phải dùng những màu khác nhau để vẽ, mức độ tương quan giữa hai thuộc tính này gần như không có.

Như hình trên, SimilarityTerm và View (góc nhìn) có quan hệ phụ thuộc với nhau, ứng với mỗi góc nhìn khác nhau thì mức độ tương quan giữa các tag sẽ khác nhau.

❖ **So khớp linh hoạt theo thời gian:**

Mức độ tương quan giữa các tag có thể thay đổi theo thời gian. Nhu cầu và “thông tin đáp ứng nhu cầu” được so khớp với nhau, từ đó hệ thống sẽ đưa ra mức độ so khớp giữa hai đối tượng này nhằm mục đích gợi ý cho người dùng lựa chọn. Tuy nhiên, “thông tin đáp ứng nhu cầu” có được người sử dụng lựa chọn hay không chỉ dựa vào một phần của sự so khớp này. Khi hệ thống có cái nhìn khách quan từ người dùng, các tập dữ liệu training sẽ được thay đổi nhằm cập nhật kịp thời mức độ tương quan giữa các tag, xu hướng “người phát sinh nhu cầu” lựa chọn cho mình những “thông tin đáp ứng nhu cầu” nào, sự so khớp thể hiện ở đâu, những thuộc tính nào có mức độ tương quan với nhau. Khi áp dụng phương pháp khai thác dữ liệu, mức độ tương quan này sẽ được cập nhật theo thời gian, giúp cho hệ thống so khớp hoạt động khách quan hơn từ ý kiến của người sử dụng. Do thời gian có giới hạn, chúng tôi chỉ cung cấp giải pháp mức độ tương quan giữa các tag thông qua class SimilarityTerm, class này lưu trữ thông tin sau khi phân tích mức độ tương quan giữa các tag, giúp cho việc truy xuất mức độ tương quan nhanh chóng và kịp thời hơn, người dùng chỉ cần cho biết hai từ khóa cần so sánh mức độ tương quan và việc so sánh dừng dưới góc nhìn cụ thể nào, class SimilarityTerm sẽ trả lời cho người dùng biết mức độ tương quan giữa chúng. Thông qua điểm mạnh dễ mở rộng của kiến trúc, hệ thống tính mức độ tương quan có thể được lập trình viên triển khai và tích hợp vào kiến trúc một cách dễ dàng.

Về phương pháp so khớp, nhóm thực hiện bằng cách duyệt và so sánh từng thuộc tính trong hai cây thuộc tính của hai đối tượng khác nhau. Nếu hai từ khóa so sánh không khớp hoàn toàn với nhau, sẽ tìm từ khóa tương quan trong SimilarityTerm, sau đó sử dụng từ khóa thực hiện so sánh lần nữa. Kết quả so sánh của hai đối tượng là số phần trăm khớp với nhau. Tuy nhiên, đối với các lĩnh vực

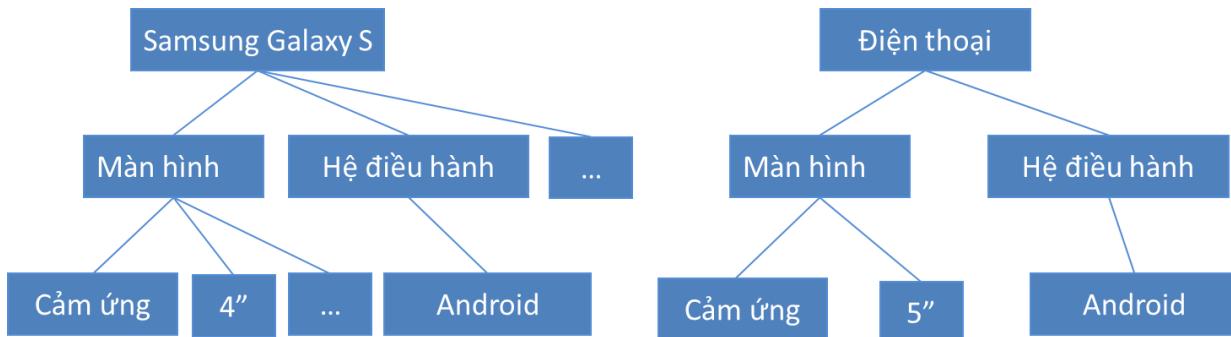
nghiệp vụ khác nhau thì cách so khớp có một vài điểm khác biệt cho nên kiến trúc phải cho phép lập trình viên viết lại giải thuật so khớp cho phù hợp với nghiệp vụ của họ.

*Việc mức độ tương quan có giá trị thay đổi theo góc nhìn và thời gian nói trên đã góp phần tạo được sự linh hoạt cho hệ thống xác định mức độ tương quan và hệ thống so khớp. Đồng thời, khi lưu trữ kết quả so sánh mức độ truy vấn giữa hai đối tượng thông qua SimilarityTerm giúp cải thiện tốc độ truy xuất giúp cho hệ thống so khớp làm việc nhanh và hiệu quả hơn.*

#### 4.4 Áp dụng cây quyết định vào bài toán

Kiến trúc lưu trữ thông tin bằng hierarchy taxonomy, việc so khớp giữa các cây taxonomy có cấu trúc tương tự nhau nhằm đưa ra mức độ so khớp giữa các cây taxonomy này. Tuy nhiên, giữa các tag mô tả thông tin trong taxonomy vẫn chưa thể xác định được mức độ phụ thuộc và tầm quan trọng trong việc người dùng lựa chọn “thông tin đáp ứng nhu cầu” để thoả mãn nhu cầu của mình. Giữa các cây taxonomy, các thông tin được mô tả rời rạc và gắn kết với nhau thông qua các thuộc tính phân lớp, cần có một giải pháp giúp xác định mức độ phụ thuộc và tầm quan trọng của thuộc tính trong việc lựa chọn “thông tin đáp ứng nhu cầu”.

Khi xác định được mức độ phụ thuộc và tầm quan trọng của các thuộc tính trong việc lựa chọn “thông tin đáp ứng nhu cầu”, hệ thống có thể gợi ý cho actor liệt kê cho mình các thuộc tính mô tả phù hợp cho “thông tin đáp ứng nhu cầu”, không những vậy, đối với những thuộc tính được đánh giá có tầm quan trọng trong việc quyết định lựa chọn sản phẩm của actor phát sinh nhu cầu nhằm cải thiện chất lượng và bổ sung vào sản phẩm các thuộc tính chưa có, giúp cho actor cải thiện “thông tin đáp ứng nhu cầu của mình”. Việc gợi ý này nhằm bổ sung thiếu sót của hệ thống so khớp bằng taxonomy – chỉ gợi ý cho actor những thuộc tính đối với một lĩnh vực cụ thể của “thông tin đáp ứng nhu cầu”. Ví dụ: Samsung cần bán dòng sản phẩm Samsung Galaxy S cho bạn và bạn đang cần mua một điện thoại di động có màn hình cảm ứng và hệ điều hành Android, thông tin về nhu cầu của nhà sản xuất và sản phẩm của bạn cần mua được lưu trữ theo dạng cấu trúc tương tự như sau:



**Hình 31. Ví dụ về lưu trữ thông tin thông qua taxonomy**

Trong ví dụ trên, hệ thống so khớp sẽ tiến hành so khớp giữa hai taxonomy này và trả về kết quả so khớp giữa hai cây. Giả sử mức độ tương quan giữa kích thước màn hình 5" và 4" bằng không, Samsung đang muốn bán sản phẩm này để giải quyết nhu cầu cho bạn, lúc này hệ thống so khớp sẽ gợi ý cho nhà sản xuất kích thước màn hình giữa Samsung Galaxy S và nhu cầu của bạn đang không khớp với nhau, nếu Samsung muốn bán sản phẩm này cho bạn thì kích thước màn hình của sản phẩm cần phải là 5". Lúc này, hệ thống so khớp gợi ý để cải thiện sản phẩm dựa trên nhu cầu cụ thể là nhu cầu của bạn. Tuy nhiên, nếu chỉ để bán sản phẩm này cho bạn thôi thì quả là một sự thiếu sót của Samsung, hệ thống sẽ gợi ý những thuộc tính nào mà đa số những người tiêu dùng quyết định khi lựa chọn sản phẩm điện thoại để Samsung có thể sản xuất các sản phẩm đáp ứng hầu hết các nhu cầu của các nhóm người dùng khác nhau.

Bên cạnh đó, kiến trúc cần có một giải pháp gợi ý cho actor những xu hướng lựa chọn “thông tin đáp ứng nhu cầu” của đa số người dùng. Như ở ví dụ trên, người tiêu dùng cần được gợi ý những tiêu chí mà đa số những người tiêu dùng khác lựa chọn khi quyết định mua điện thoại, giúp cho người tiêu dùng bớt phân vân trong việc lựa chọn những tính năng phù hợp cho một chiếc điện thoại mà họ sắp mua.

Để giải quyết vấn đề xác định mức độ phụ thuộc và tầm quan trọng của thuộc tính trong việc lựa chọn “thông tin đáp ứng nhu cầu” nói trên, chúng ta cần có một công cụ khai thác dữ liệu, dựa vào đối tượng phân lớp “quyết định lựa chọn thông tin đáp ứng nhu cầu” để dự đoán mức độ phụ thuộc giữa các thuộc tính, chúng tôi nhận thấy có ba phương pháp khai thác dữ liệu phù hợp sau đây [2]:

- Decision Tree (Cây quyết định): như đã trình bày ở những phần trên, cây quyết định là một trong những thuật toán điển hình hỗ trợ phân loại và hồi quy, sử dụng rất tốt các mô hình dự đoán. Thuật toán này sẽ khảo sát sự ảnh hưởng của mỗi thuộc tính

trong tập dữ liệu và kết quả của thuộc tính dự đoán, sau đó sử dụng các thuộc tính đầu vào với các quan hệ rõ ràng để tạo thành một nhóm phân hoá (các node). Sự liên kết với nhau theo mức độ phụ thuộc giữa các node với nhau thiết lập nên cấu trúc dạng cây.

- Clustering (Phân cụm): Thuật toán này sử dụng kỹ thuật lặp nhầm mục tiêu nhóm các bản ghi từ một tập hợp dữ liệu vào một phân nhóm có đặc điểm giống nhau. Sử dụng phân nhóm này chúng ta có thể khám phá dữ liệu, tìm hiểu về các quan hệ đã tồn tại – các quan hệ này không dễ dàng tìm được một cách hợp lý thông qua quan sát ngẫu nhiên.

- Naïve Bayes: thuật toán này tính toán khả năng có thể xảy ra trong mỗi trường hợp lẻ của thuộc tính đầu vào, gán cho mỗi trường dữ liệu một thuộc tính có thể dự đoán, thuật toán này là sự lựa chọn tốt để khai phá dữ liệu, khám phá các thuộc tính đầu vào được phân bố trong các trường khác nhau của thuộc tính dự đoán như thế nào. Tuy nhiên, thuật toán này không được chúng tôi lựa chọn vì thuộc tính phân lớp đã được chúng tôi xác định cụ thể (thuộc tính “quyết định lựa chọn thông tin đáp ứng nhu cầu” được xác định làm thuộc tính phân lớp)

Đối với bài toán này, cây quyết định là một giải pháp tối ưu hơn phân cụm [19]. Cây quyết định giúp cho việc xác định được mức độ quan hệ giữa các thuộc tính nhằm đưa ra các thuộc tính có yếu tố quyết định trong việc lựa chọn “thông tin đáp ứng nhu cầu”, từ đó có thể dễ dàng hơn trong việc gợi ý cải thiện “thông tin đáp ứng nhu cầu”, hỗ trợ người dùng đề ra các nhu cầu phù hợp cho mình dựa vào thống kê xác suất tại các node lá của cây quyết định.

Về công cụ hỗ trợ tạo cây quyết định, chúng tôi dựa vào Microsoft Business Intelligence (BI). Microsoft BI được thiết kế xây dựng dựa trên nền tảng dữ liệu có khả năng mở rộng cho việc sắp xếp, phân tích, báo cáo dữ liệu và cung cấp các công cụ trực quan và mạnh mẽ để người dùng có thể sử dụng truy cập và phân tích các thông tin. Hạt nhân cuối cùng trong Microsoft BI đó là Microsoft SQL Server (hỗ trợ tốt nhất từ phiên bản 2008 trở lên). Microsoft Business Intelligence bao gồm các thành phần sau:

Thành phần	Mô tả
SQL Server Database Engine	Mang đến một phương tiện lưu trữ dữ liệu hiệu suất cao và có khả năng mở rộng cho các bản dữ liệu rất lớn. Điều đó làm cho nó trở thành một lựa chọn lý tưởng cho việc hợp nhất dữ liệu doanh nghiệp từ toàn bộ doanh nghiệp vào một trung tâm dữ liệu để thuận tiện cho việc phân tích và báo cáo.
SQL Server Integration Services	Một nền tảng toàn diện cho việc trích rút, biến đổi và tải (ETL), các hành động này cho phép cung cấp và đồng bộ kho lưu trữ dữ liệu của bạn với dữ liệu từ các nguồn khác nhau được sử dụng bởi các ứng dụng doanh nghiệp trong toàn bộ tổ chức.
SQL Server Analysis Services	Cung cấp phương tiện phân tích cho các giải pháp phân tích xử lý trực tuyến (OLAP), cụ thể gồm có việc thu nạp các tham số doanh nghiệp qua nhiều kích thước và các chỉ thị hiệu suất chính (KPI), và cho các giải pháp khai thác dữ liệu có sử dụng các thuật toán đặc biệt để nhận dạng mẫu, xu hướng và quan hệ trong dữ liệu doanh nghiệp.
SQL Server Reporting Services	Một giải pháp báo cáo mở cho phép tạo, xuất bản và phân phối các báo cáo doanh nghiệp chi tiết một cách dễ dàng cả bên trong và bên ngoài tổ chức.

**Bảng 9. Các thành phần của Microsoft Business Intelligence**

SQL Server Database Engine và SQL Server Analysis Services sẽ giúp cho việc triển khai xây dựng cây quyết định dễ dàng hơn. *Việc thực hiện tạo cây quyết định có thể mất nhiều thời gian tính toán. Giải pháp nhằm đảm bảo hiệu năng hệ thống khi áp dụng cây quyết định được giải quyết như sau:*

- Thông tin về các bảng cần khai thác dữ liệu bằng cây quyết định sẽ được chuyển về một hệ thống máy chủ khác, hạn chế việc truy vấn liên tục vào hệ thống máy chủ chính
- Việc tính toán xây dựng cây quyết định thông qua SQL Analysis Services sẽ được triển khai độc lập với hệ thống máy chủ chính, hệ thống này có tách rời hay dùng chung với hệ thống lưu trữ dữ liệu truy xuất xây dựng cây.

- Dữ liệu sau khi phân tích xong sẽ được lưu vào hệ thống máy chủ chính để người dùng truy xuất. Việc truy xuất dữ liệu đã phân tích này giúp cải thiện đáng kể hiệu năng của hệ thống khi ứng dụng cây quyết định.

Tóm lại, nhằm xác định được mức độ phụ thuộc và tầm quan trọng trong việc người dùng lựa chọn “thông tin đáp ứng nhu cầu” để thoả mãn nhu cầu của mình, giữa các cây taxonomy, các thông tin được mô tả rời rạc và gắn kết với nhau thông qua các thuộc tính phân lớp, giải pháp xây dựng cây quyết định giúp xác định mức độ phụ thuộc và tầm quan trọng của thuộc tính trong việc lựa chọn “thông tin đáp ứng nhu cầu”. Giải pháp này hỗ trợ bổ sung thiếu sót của hệ thống so khớp bằng taxonomy, gợi ý cải thiện “thông tin đáp ứng nhu cầu” và gợi ý người dùng đưa ra các nhu cầu phù hợp nhưng vẫn đảm bảo hiệu năng của hệ thống. Chúng tôi lựa chọn Microsoft Business Intelligent cụ thể là Microsoft Analysis Services để hỗ trợ xây dựng cây quyết định cho kiến trúc.

#### 4.5 Áp dụng lý thuyết kiến trúc phần mềm vào bài toán

Như đã trình bày tổng quan một số lý thuyết cơ sở về kiến trúc phần mềm (được đề cập trong báo cáo từ phần 3.6.5 đến 3.6.9). Vậy làm thế nào áp dụng các cơ sở lý thuyết trên vào xây dựng kiến trúc phần mềm hiện thực giải pháp ba bài toán chính trên.

- Cơ sở lý thuyết về mô hình 4+1 architecture view model: giúp mô tả giải pháp của mỗi bài toán dưới nhiều góc độ khác nhau: Scenarios, Logical, Process, Development, Physical. Từ đó xác định 3 khía cạnh chính của kiến trúc phần mềm: Components, Connecters, Constraint.
- Cơ sở lý thuyết về OOP Paradigm, Design Pattern giúp thiết kế kiến trúc mức chi tiết.
- Ngoài ra, với các yêu cầu khác về chất lượng của kiến trúc phần mềm (như tính mở rộng, tính khả chuyển, tính hiệu năng...) mà áp dụng sự tổng hợp các cơ sở lý thuyết về kiến trúc để thiết kế kiến trúc phần mềm.

## 5 Áp dụng giải pháp vào bài toán Job Zoom

### 5.1 Thực trạng các website tuyển dụng hiện nay

#### Về phía người tìm việc:

- Đăng tải thông tin CV theo khuôn mẫu nhất định
- Hạn chế trong việc hỗ trợ người tìm việc mô tả, minh chứng cho những kinh nghiệm hay kỹ năng của họ
  - Không hỗ trợ ứng viên viết CV, không thể hiện kiến thức, kỹ năng nào có mức độ quan trọng đối với công việc hiện tại mà họ muốn ứng tuyển, không thể hiện cho ứng viên biết kỹ năng nào ứng viên còn thiếu sót cần học tập, trau dồi thêm. Các website hiện tại, chức năng gợi ý cho ứng viên bổ sung thông tin hồ sơ còn đơn giản (chỉ gợi ý ứng viên nên cập nhật các thông tin quan trọng như kinh nghiệm làm việc, học vấn), không hướng đến một vị trí làm việc hay ngành nghề mà ứng viên mong muốn, không gợi ý những thông tin có thể quyết định cơ hội việc làm cho ứng viên.

#### Về phía nhà tuyển dụng

- Đăng tải thông tin việc làm theo dạng văn bản theo khuôn mẫu nhất định
- Chưa hỗ trợ nhà tuyển dụng đăng tải thông tin về công việc
- Hỗ trợ tốt việc tìm kiếm công việc, hồ sơ ứng viên nhưng chưa thấy được sự khớp giữa yêu cầu tuyển dụng và hồ sơ của ứng viên.

### 5.2 Những vấn đề Job Zoom cần giải quyết

#### 5.2.1 Vấn đề 1: Hỗ trợ người dùng viết CV theo ngành nghề

**Bạn là người tìm việc?** Có thể bạn đã từng gặp khó khăn trong việc viết sơ yếu lý lịch (CV hay còn gọi là resume) để tạo ấn tượng cho nhà tuyển dụng, tiếp thị cho bản thân mình. Bên cạnh đó, mẫu CV sẽ khác nhau đối với từng ngành nghề khác nhau. Để lọt vào danh sách ứng viên hàng đầu của nhà tuyển dụng, bạn cần phải có một sơ yếu lý lịch ngắn gọn, súc tích nhưng cần đầy đủ và rõ ràng về quá trình làm việc, trình độ chuyên môn, học vấn và các kỹ năng mềm khác, việc viết CV này thật sự trở nên khó khăn khi bạn vẫn loay hoay chưa biết phải trình bày hay liệt kê những thông tin như thế nào. Những ứng dụng tìm việc hiện tại không thể gợi ý cho bạn tạo một CV chuẩn cho một ngành nghề cụ thể hay nói khác đi là không hỗ trợ người dùng

tạo CV tìm việc đầy đủ và rõ ràng đối với ngành nghề mà người tìm việc muốn ứng tuyển.

Khi người tìm việc muốn tìm một công việc nào đó, bên cạnh những thông tin cá nhân, ứng dụng sẽ gợi ý một CV chuẩn về ngành nghề, người dùng sẽ tạo ra sơ yếu lý lịch cho mình dựa vào các thuộc tính chương trình gợi ý. Ngoài ra, khi cần ứng tuyển vào một vị trí của một nhà tuyển dụng nào, ứng dụng sẽ gợi ý người dùng những thông tin cần bổ sung cho vị trí tìm việc.

### 5.2.2 Vấn đề 2: Hỗ trợ nhà tuyển dụng trong việc đăng tải yêu cầu công việc một cách chi tiết và có trọng số theo ngành nghề

**Bạn là nhà tuyển dụng?** Các website tìm việc hiện nay, đều cho nhà tuyển dụng đăng thông tin dưới dạng văn bản thô, không đưa ra gợi ý cho bạn về các ứng viên có khả năng đáp ứng yêu cầu công việc dựa vào các trọng số các yêu cầu của bạn đưa ra. Khi nhà tuyển dụng muốn đăng thông tin tuyển dụng, các website tìm việc hiện tại không thể đánh giá mức độ đáp ứng của những ứng viên dựa vào những thông tin nhà tuyển dụng cung cấp, không thể biết ứng viên đáp ứng được bao nhiêu phần trăm những yêu cầu này, nhà tuyển dụng phải tự đánh giá bằng việc đọc những CV của ứng viên sau đó lựa chọn, việc này gây mất nhiều thời gian và chi phí.

Với Job Zoom framework, ứng dụng hỗ trợ nhà tuyển dụng đăng tải thông tin tuyển dụng linh động hơn và có thể đặt trọng số cho từng yêu cầu của công việc. Ứng dụng đánh giá mức độ phù hợp của ứng viên và liệt kê những ứng viên tiềm năng dựa vào thông tin của nhà tuyển dụng cung cấp.

### 5.2.3 Vấn đề 3: So khớp hồ sơ ứng viên với yêu cầu tuyển dụng

Sau quá trình phân tích nghiệp vụ tuyển dụng, chúng tôi nhận thấy cấu trúc thông tin của yêu cầu tuyển dụng và hồ sơ ứng viên có một cấu trúc tương tự với nhau. Vì vậy ta có thể thực hiện so khớp thông tin của chúng với nhau.

**Đối với người tìm việc:** Hệ thống so khớp gợi ý người dùng hoàn thiện, bổ sung những thiếu sót của CV, tăng cơ hội ứng tuyển vào một vị trí tại một công ty cụ thể.

Đối với nhà tuyển dụng: Hệ thống so khớp đánh giá ứng viên xin ứng tuyển vào công việc, hỗ trợ cho nhà tuyển dụng trong việc đánh giá ứng viên.

### 5.3 Kết quả mong muốn

Áp dụng “*Giải quyết bài toán kiến trúc*” vào giải quyết các vấn đề trên.

- ✓ Xây dựng JobZoom framework đảm bảo thông tin về CV và yêu cầu công việc được lưu trữ linh hoạt, dễ dàng so khớp.
  - ✓ Áp dụng so khớp kết hợp độ tương quan giữa các tag, hỗ trợ nhà tuyển dụng đánh giá ứng viên và giúp ứng viên nhận ra khuyết điểm của mình, hoàn thiện CV.
  - ✓ Kết hợp phương pháp khai thác dữ liệu bằng cây quyết định tạo nền tảng cải thiện khả năng thích ứng của hệ thống với từng loại ngành nghề, gợi ý ứng viên viết CV và nhà tuyển dụng đăng yêu cầu công việc một cách nhanh chóng và hiệu quả.
  - ✓ Xây dựng được kiến trúc phần mềm đảm bảo hiệu năng của hệ thống với số lượng người sử dụng lớn khi triển khai cây quyết định và hệ thống so khớp.
  - ✓ JobZoom framework dễ dàng mở rộng và triển khai.

## 6 Những vấn đề cần giải quyết trong JobZoom framework

Trong phần này, chúng tôi sẽ giới thiệu các lý thuyết và phương pháp chúng tôi áp dụng được từ giải pháp kiến trúc mà chúng tôi xây dựng ở trên: “kiến trúc lưu trữ và so khớp thông tin”, sau đó sẽ giới thiệu kiến trúc một cách chi tiết hơn ở phần “Giải pháp kiến trúc công thông tin tìm việc JobZoom”.

### 6.1 Khái quát phương pháp giải quyết vấn đề

Những vấn đề của JobZoom framework xoay quanh bài toán kiến trúc của chúng tôi. Những phương pháp để ra để thực hiện “kiến trúc lưu trữ và so khớp thông tin” đã được chúng tôi hiện thực hóa, áp dụng vào lĩnh vực cụ thể đó là lĩnh vực tìm kiếm việc làm và tuyển dụng trực tuyến. Nhằm hỗ trợ so khớp giữa các hồ sơ của ứng viên và yêu cầu tuyển dụng, đòi hỏi thông tin về hồ sơ ứng viên cũng như yêu cầu tuyển dụng cần phải đạt mức độ linh hoạt, có mối quan hệ tương đồng trong quá trình phân loại, lưu trữ thông tin sao cho có thể dễ dàng so khớp.

JobZoom framework còn ứng dụng cây quyết định và hệ thống so khớp của kiến trúc được xây dựng sẽ giúp nhà tuyển dụng dễ dàng hơn trong việc đăng tải yêu cầu công việc, giúp cho người tìm việc dễ dàng viết một hồ sơ xin việc một cách chi tiết, rõ ràng hơn.

Bên cạnh đó, giải pháp đảm bảo hiệu năng và kiến trúc dễ dàng triển khai mở rộng, cũng được chúng tôi chứng minh và hiện thực thông qua JobZoom framework.

## 6.2 Ứng dụng cấu trúc lưu trữ dưới dạng tag kết hợp taxonomy

Khi vào các website tuyển dụng hiện nay, bạn có thể thấy thông tin ứng viên được khai báo theo những cụm thuộc tính đã được liệt kê sẵn, hoàn toàn không có sự linh động. Đồng thời, thông tin về yêu cầu tuyển dụng cũng được lưu trữ dưới dạng văn bản, gây khó khăn trong việc so khớp giữa hồ sơ xin việc và yêu cầu tuyển dụng của doanh nghiệp. Chúng tôi nhận thấy giải pháp kiến trúc của chúng tôi khi áp dụng vào lĩnh vực tuyển dụng rất hiệu quả. Các website tuyển dụng cần có một tổ chức thông tin linh hoạt và so khớp nhanh và hiệu quả.

Khi ứng dụng lưu trữ dưới dạng tag kết hợp taxonomy, CV của người tìm việc và yêu cầu tuyển dụng của doanh nghiệp được mô tả rất linh hoạt. Thông tin được lưu trữ dưới dạng cấu trúc cây phân cấp, có mức độ tương đồng giữa cây thông tin về CV và yêu cầu tuyển dụng nên việc so khớp có thể được thực hiện.

## 6.3 Ứng dụng hệ thống so khớp kết hợp mức độ tương quan

Thông tin yêu cầu tuyển dụng và hồ sơ ứng viên được tổ chức linh hoạt bằng tag và được phân loại bằng hierarchy taxonomy nhằm tạo ra một cấu trúc chung giữa yêu cầu tuyển dụng và hồ sơ của ứng viên, nói cách khác là tạo ra cơ sở để thực hiện việc so khớp. Hệ thống so khớp của kiến trúc được áp dụng vào việc so khớp yêu cầu tuyển dụng và hồ sơ ứng viên, tìm kiếm điểm tương đồng giữa chúng.

Vấn đề đặt ra là thông tin của một yêu cầu công việc và một hồ sơ của ứng viên bất kỳ không có khớp hoàn toàn với nhau, do các từ khóa được người dùng tự do gắn kết vào thông tin (nhà tuyển dụng và người xin việc có thể sử dụng những thuật ngữ để mô tả thông tin khác nhau) nhưng các từ khóa này sẽ có một mối tương quan nào đó. Và mối tương quan này cũng khác nhau nếu ta nhìn dưới một góc nhìn (view)

khác nhau, ví dụ đối với vị trí lập trình viên thì kỹ năng thuyết trình và kỹ năng truyền thông có độ tương quan cao, nhưng nếu xét ở một vị trí nhân viên marketing thì hai kỹ năng này có sự phân biệt rõ ràng hơn, tức là độ tương quan thấp hơn. Vậy làm sao hệ thống có thể so khớp hai cây thông tin này để đưa ra mức độ khớp của chúng ?

#### 6.4 Ứng dụng cây quyết định

Không chỉ dừng lại ở vấn đề gợi ý cho ứng viên viết CV gây ấn tượng tốt, có được cơ hội việc làm cao nhất, việc ứng dụng cây quyết định vào JobZoom framework còn giúp gợi ý nhà tuyển dụng đăng thông tin tuyển dụng của mình, các thuộc tính nào doanh nghiệp cần và những thuộc tính đó thường được ứng viên miêu tả ra sao đối với vị trí tuyển dụng của họ đang cần đăng tuyển.

Thuộc tính phân lớp “Is Approve” (thuộc tính xác định ứng viên có được nhà tuyển dụng nhận hay không sau khi nộp hồ sơ ứng tuyển) đóng vai trò là thuộc tính phân lớp. Như vậy, khi khai thác dữ liệu, hệ thống sẽ sinh ra rất nhiều cây quyết định theo từng vị trí làm việc, những cây này sẽ được tổng hợp lại giúp việc truy vấn kết quả trở nên dễ dàng hơn.

Kết quả khi tạo cây quyết định thể hiện chúng ta có thể thấy mức độ phụ thuộc, tầm quan trọng của các thuộc tính trong việc lựa chọn ứng viên vào một vị trí cụ thể. Những thuộc tính này gợi ý cho người dùng những kỹ năng, kiến thức người tìm việc nên có đối với vị trí làm việc mà họ đang tìm. Như vậy, khi người dùng sử dụng chương trình, chương trình sẽ đặt câu hỏi và liệt kê các thuộc tính (kỹ năng, kinh nghiệm) có mức độ phụ thuộc cao vào xác suất ứng viên được nhận, ứng viên có thể thay đổi CV cho phù hợp với vị trí mình đang cần ứng tuyển. Nếu một trong những thuộc tính mà Job Zoom gợi ý người tìm việc không thể đáp ứng được, thì Job Zoom có thể gợi ý cho ứng viên những thuộc tính khác hoặc ứng viên có thể học tập, rèn luyện để bổ sung kịp thời những thuộc tính đó nhằm tăng xác suất và tính cạnh tranh của họ khi apply vào công việc. Bên cạnh đó, khi áp dụng kiến trúc trên, JobZoom framework còn đưa ra danh sách các tập thuộc tính mô tả công việc thường dùng đối với vị trí mà doanh nghiệp đang cần tuyển dụng, giúp cho doanh nghiệp thực hiện đăng tải yêu cầu công việc một cách hiệu quả hơn.

Thông qua cây quyết định và việc gom nhóm cây, vấn đề 1 của JobZoom framework đã được giải quyết. Tuy nhiên, mức độ giải quyết này đáp ứng cho những ứng viên muốn ứng tuyển vào một vị trí công việc bất kỳ, xác suất của ứng viên được nhận cao khi ứng viên apply vào nhiều công ty tuyển dụng tương ứng với vị trí đó. Nhưng chính việc ứng dụng taxonomy được đề cập ở trên đã giải quyết phần còn lại của vấn đề 1, khi ứng viên apply vào vị trí công việc tại một công ty cụ thể, thì những thuộc tính nào giúp ứng viên apply vào vị trí đó có xác suất được nhận cao nhất.

### 6.5 Ứng dụng kiến trúc dễ dàng triển khai và mở rộng

Áp dụng “kiến trúc lưu trữ và so khớp thông tin” nói trên giúp JobZoom framework có thể dễ dàng triển khai và mở rộng:

- Kiến trúc nghiệp vụ được xây dựng ra có thể dễ dàng áp dụng được khi các lập trình viên có nhu cầu sử dụng.
- Có thể mở rộng, tối ưu việc xây dựng cây, thuật toán khai thác dữ liệu, cách hoạt động của việc so khớp. Kiến trúc hỗ trợ lập trình viên trong việc mở rộng kiến trúc, kế thừa các interface, class của kiến trúc để định nghĩa cho hệ thống nghiệp vụ của mình có những đặc thù riêng: có thể sử lại cách tính điểm của hệ thống so khớp, cách xây dựng cây, thuật toán khai thác dữ liệu…

### 6.6 Ứng dụng kiến trúc đảm bảo hiệu năng khi áp dụng cây quyết định và hệ thống so khớp

Hệ thống so khớp và cây quyết định làm việc rất tốn nhiều thời gian và chi phí tính toán, JobZoom framework giải quyết vấn đề đảm bảo hiệu năng khi xây dựng cây quyết định và hệ thống so khớp cụ thể như sau:

- Những công việc và profile cụ thể được lựa chọn và sử dụng kỹ thuật Pivot Transformation để tạo các view. Các view này được lưu trữ trên hệ thống server hỗ trợ xây dựng cây quyết định. Analysis Services được kết nối đến hệ thống lưu trữ dữ liệu này lấy dữ liệu xử lý và tiến hành khai thác dữ liệu. Hệ thống này có thể xử lý mất hàng tuần thậm chí kéo dài hàng tháng, tuy nhiên nó hoạt động độc lập so với hệ thống máy chủ vận hành website nên không làm ảnh hưởng đến hiệu năng của hệ thống. Sau khi kết quả được khai thác xong sẽ được chuyển về hệ thống server chính

để truy vấn sử dụng. Nhờ khai thác dữ liệu trên hệ thống máy chủ độc lập với hệ thống chính và lưu trữ dữ liệu đã khai thác vào database của hệ thống server chính giúp cho việc sử dụng và truy vấn nhanh chóng và dễ dàng hơn.

- Dữ liệu về độ tương quan được lưu trữ sẵn, chỉ cần truyền vào hai tag cần so sánh mức độ tương quan, JobZoom framework truy vấn nhanh chóng kết quả mức độ tương quan giữa hai tag này thông qua bảng SimilarityTerm giúp cho hệ thống so khớp hoạt động hiệu quả mà vẫn đảm bảo hiệu năng.
- Hệ thống so khớp làm việc bằng cây có cấu trúc nên so khớp trên cây cũng đã cải thiện một phần hiệu năng của hệ thống này.

## 7 Giải pháp kiến trúc cổng thông tin tìm việc JobZoom

### 7.1 Điều kiện ra đời

Ngày nay, internet đã dần chiếm hữu cuộc sống hiện tại, hết thảy mọi việc đều được “số hoá” cũng là lúc tuyển dụng và người tìm việc đều chọn các website tìm việc làm cầu nối cho mình.

Theo số liệu thống kê của tổ chức Internet Usage World Stats<sup>24</sup>, tính đến nay có hơn 2,1 tỉ người trên thế giới sử dụng Internet. Bên cạnh đó, nghiên cứu về người tiêu dùng do Mintel International Group Ltd. (tổ chức quốc tế chuyên nghiên cứu về thị trường và người tiêu dùng) tiến hành cho thấy có đến 30% số người dùng Internet đã tham khảo các website tuyển dụng trực tuyến để tìm kiếm thông tin về việc làm. Chúng ta có thể thấy xu hướng tìm việc trực tuyến đang ngày càng phát triển với tốc độ rất nhanh.

Việt Nam là quốc gia đang phát triển với dân số trên 90 triệu người với trên 28,6 triệu người sử dụng internet chiếm khoảng 31,6% dân số (2011)<sup>25</sup>. Các trang web về lao động việc làm ra đời đã ngày càng đem lại nhiều thông tin tổng hợp, đa dạng về công việc, mở ra những cơ hội lớn hơn cho người tìm việc, đồng thời giúp doanh nghiệp giải quyết nhu cầu nhân sự nhanh chóng và hiệu quả hơn. Tuy nhiên, thực trạng chung của các website tìm việc hiện nay chưa ứng dụng triệt để khai thác dữ liệu

<sup>24</sup> <http://www.internetworldstats.com/stats.htm>

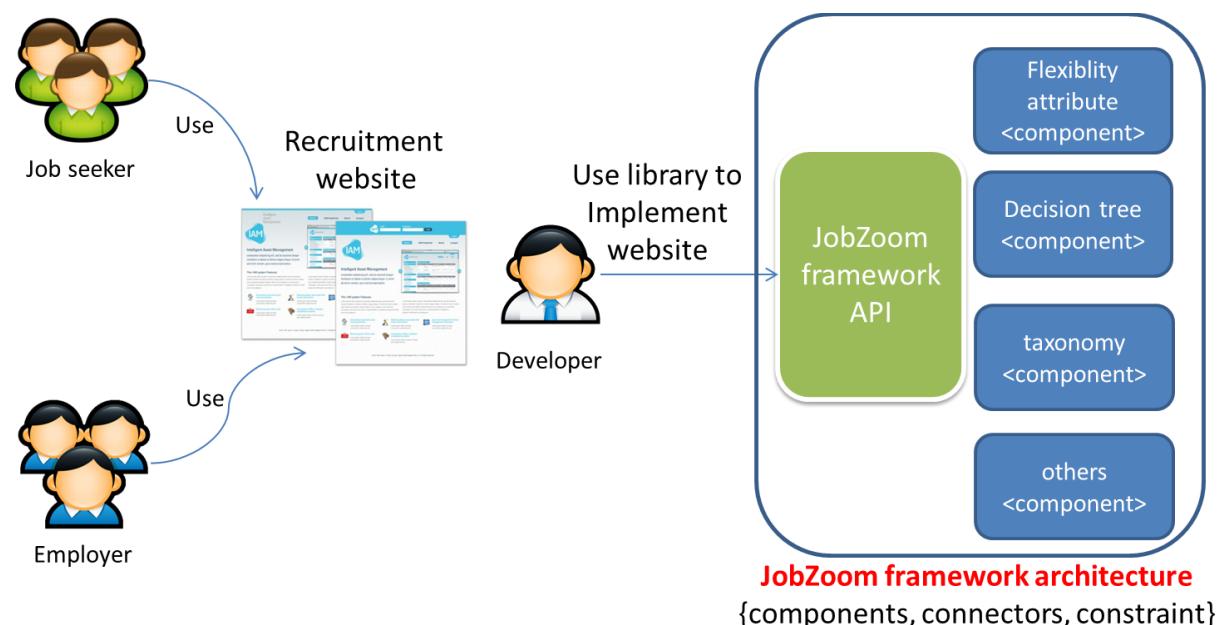
<sup>25</sup> <http://www.internetworldstats.com/asia.htm#vn>

trong việc gợi ý người dùng viết CV hay hỗ trợ doanh nghiệp trong việc đánh giá ứng viên thông qua thông tin ứng viên cung cấp. Để góp phần đáp ứng những yêu cầu thực tiễn này, đề tài “**xây dựng kiến trúc hệ thống công thông tin tìm việc**” được ra đời.

JobZoom framework được phát triển dựa theo phần 4 – Giải quyết bài toán kiến trúc.

## 7.2 Kiến trúc JobZoom framework

### 7.2.1 Kiến trúc tổng quan framework



**Hình 32. Mô hình kiến trúc tổng quan JobZoom framework khi ứng dụng vào thực tế**

Kiến trúc tổng quan của JobZoom framework được mô tả cụ thể qua mô hình 4+1 architecture view với cái nhìn với nhiều góc độ khác nhau:

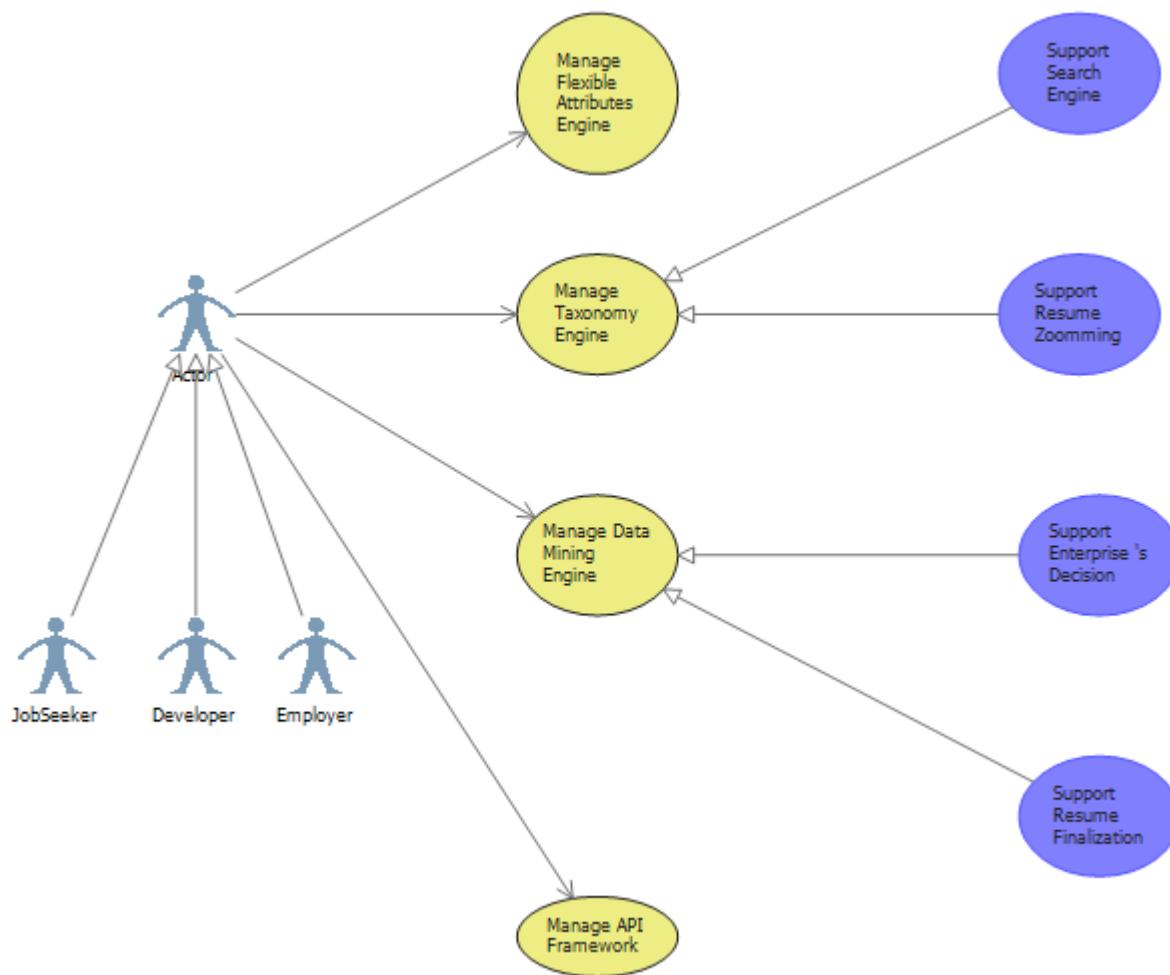
- ❖ *Scenarios*: mô tả các trường hợp sử dụng cụ thể của JobZoom framework, được mô tả thông qua sơ đồ Use case model.
- ❖ *Logical view*: mô tả kiến trúc JobZoom framework dưới góc độ của người sử dụng framework. Ở đây có 2 mô hình mô tả logical view đó là: class diagram và layer diagram.

- ❖ *Development view*: mô tả kiến trúc JobZoom framework dưới góc độ của người lập trình viên phát triển JobZoom framework được thể hiện thông qua sơ đồ component diagram.
- ❖ *Process view*: mô tả kiến trúc JobZoom framework dưới góc độ của người tích hợp hệ thống cần nắm các quy trình xử lý của hệ thống, được thể hiện qua mô hình activity diagram.
- ❖ *Physical view*: mô tả kiến trúc JobZoom framework dưới góc độ của người triển khai hệ thống phải làm việc với các Tier mức vật lý của hệ thống, được thể hiện qua mô hình logic vật lý.

#### ***7.2.1.1 Senarios (Use Case) – các trường hợp sử dụng cơ bản của JobZoom framework và trong ứng dụng vào thực tiễn.***

Theo phần 7.1 về điều kiện ra đời và phần 7.2 về kiến trúc tổng quan của JobZoom framework: JobZoom framework cung cấp cho lập trình viên phát triển các website tìm kiếm việc làm một thư viện lập trình linh hoạt giao tiếp với JobZoom framework phục vụ cho việc ứng dụng giải quyết các 3 bài toán lớn.

- ❖ JobZoom framework library là component trung gian giữa các người sử dụng JobZoom framework vì framework library (API) làm việc với tất cả các component khác trong framework (được mô tả cụ thể qua layer diagram trong phần 7.2.1.2 Logical View).
- ❖ Vì JobZoom library (API) có vai trò trung gian vì vậy các trường hợp sử dụng của JobZoom framework cũng được thể hiện tất cả trong framework library (API) này:

**Hình 33. Sơ đồ use case JobZoom framework**

Actor	Mô tả
<Actor>	Actor tổng quát của Framework, mô tả người sử dụng Framework này vào nhiều mục đích khác nhau. Ở mỗi mục đích, Actor đó có một thể hiện riêng cụ thể: Người tìm kiếm việc (Job Seeker), Doanh nghiệp tuyển việc (Employer), Lập trình viên phát triển ứng dụng dựa trên Framework này (Developer)
Job seeker	Ứng viên tìm việc sử dụng hệ thống (ứng dụng JobZoom framework)
Employer	Nhà tuyển dụng sử dụng hệ thống (ứng dụng JobZoom framework)
Developer	Lập trình viên phát triển hệ thống tìm việc (ứng dụng JobZoom framework) sử dụng trực tiếp Jobzoom framework library (api)

**Bảng 10. Bảng mô tả tác nhân trong JobZoom framework**

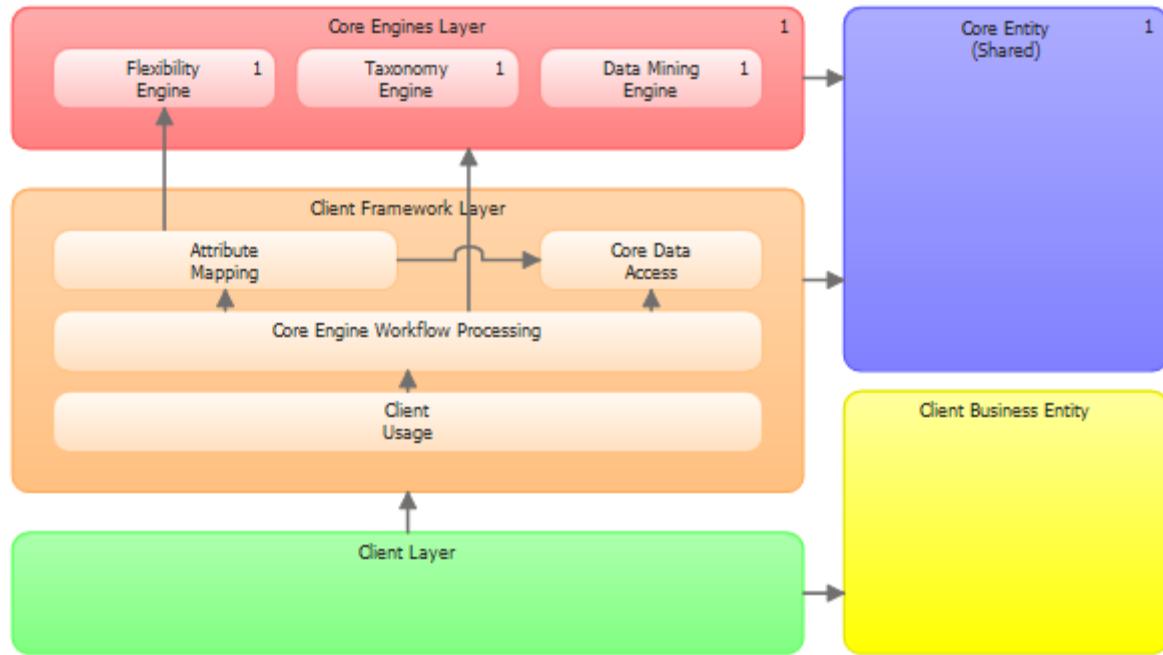
ID	Use Case	Actor	Mô tả
UC01	Manage Flexible Attribute	Developer	<p>Framework cung cấp khả năng quản lý tổ chức và lưu trữ thông tin linh hoạt (bài toán 1) phục vụ cho khả năng so khớp thông tin (bài toán 2). Thông tin người dùng hệ các hệ thống website tìm kiếm việc làm được triển khai, xây dựng từ Jobzoom framework sẽ được tổ chức và lưu trữ thông tin sang dạng tag và cây phân cấp phục vụ cho việc so khớp thông tin và hỗ trợ đưa ra quyết định sau này.</p> <p><b>Pre-condition:</b> thông tin người được mapping chuyển đổi sang dạng tag, tiền xử lý các tag, xử lý và lưu trữ dưới dạng cây phân cấp.</p> <p><b>Post-condition:</b> thông tin có cấu trúc (schema của các business entity riêng theo chiến lược xây dựng website) được chuyển đổi (mapping), tổ chức và lưu trữ sang dạng tag.</p>
UC02	Manage Taxonomy Engine	Website owner Developer	<p>Website owner: tùy theo chiến lược của mỗi website tìm kiếm việc mà đưa ra chiến lược đưa ra các thông tin phân hoạch (các thông tin giúp dựa vào đó mô tả thông tin của người tìm kiếm việc và yêu cầu công việc) trở thành các tiêu chí so khớp thông tin.</p> <p>Developer: implement cây phân cấp tùy biến theo chiến lược website mình.</p> <p><b>Pre-condition:</b> phải có thông tin các thuộc tính phân hoạch tùy theo chiến lược của mỗi website tìm kiếm việc.</p> <p><b>Post-condition:</b> tổ chức và lưu trữ thông tin phân cấp dựa theo các thông tin phân hoạch.</p>

UC02	Manage Data Mining Engine	Developer	<p>Sử dụng framework library để xây dựng cây quyết định dựa trên thông tin của người sử dụng website. Use case này sẽ làm việc trực tiếp với các thành phần khác trong framework bao gồm hệ quản trị cơ sở dữ liệu và JobZoom datamining service (windows service) để phục vụ cho cung cấp công cụ khai thác dữ liệu của hệ thống.</p> <p><b>Pre-condition:</b> JobZoom datamining service (windows service) đã được thiết lập, cấu hình và đã khởi động. Cơ sở dữ liệu của framework (core database) đã được triển khai trên DBMS. Lập trình phải cung cấp chuỗi kết nối đến database này trong thư viện quản lý Datamining service.</p> <p><b>Post-condition:</b> khai thác dữ liệu từ Core database và xây dựng cây quyết định (decision tree).</p>
UC03	Manage framework library (API)	Developer	<p>Developer: Use case mô tả framework cung cấp một thư viện lập trình cho các developer xây dựng website tìm kiếm việc làm giúp phục vụ giải quyết các bài toán trọng tâm của đề tài đã trình bày.</p> <p><b>Pre-condition:</b> bản mô tả chiến lược phát triển website, trong đó mô tả về các thuộc tính phân lớp thông tin của người dùng.</p> <p><b>Post-condition:</b> giao tiếp với các chức năng của framework thông qua thư viện lập trình.</p>
UC04	Support Search Engine	Employer Job seeker	<p>Use case này giúp hỗ trợ cho sự khớp thông tin giữa hồ sơ người tìm việc và thông tin công việc của nhà tuyển dụng.</p> <p>Đối với nhà tuyển dụng: dễ dàng tìm ra các hồ sơ tìm kiếm (các ứng viên) phù hợp với yêu cầu thông tin của công việc.</p>

			<p>Đối với người tìm việc: dễ dàng tìm ra các công việc phù hợp với mình.</p> <p><b>Pre-condition:</b> các hierarchy tree mô tả thông tin người dùng đã được xây dựng.</p> <p><b>Post-condition:</b> thông tin của người dùng có thể so khớp với nhau.</p>
UC05	Support Resume Zoomming	Job seeker	<p>JobZoom framework giúp cho các website có thể xây dựng chức năng hỗ trợ các mẫu (resume template) để giúp người tìm kiếm việc mô tả thông tin resume phù hợp với công việc cần dự tuyển.</p> <p>Người tìm kiếm việc có thể được hệ thống gợi ý nên chú trọng vào những thông tin gì mà nhà doanh nghiệp đó quan tâm, chú trọng, hoặc những thông tin, tiêu chí gì mà mình đang thiếu, cần bổ sung.</p> <p><b>Pre-condition:</b> thông tin về các công việc được tổ chức và lưu trữ với giải pháp taxonomy. Đồng thời giải pháp cây quyết định về thông tin của một công việc phải được triển khai thành công.</p> <p><b>Post-condition:</b> người kiếm việc được gợi ý các thông tin để chú trọng làm nổi bật hồ sơ hoặc được gợi ý các thông tin cần bổ sung, cần trao đổi, học hỏi để có thể được doanh nghiệp tuyển chọn.</p>
UC06	Support Employer's Decision	Employer	<p>Hỗ trợ nhà tuyển dụng đưa ra quyết định chọn ứng viên nào dựa trên cây quyết định.</p> <p><b>Pre-condition:</b> cây quyết định của một công việc đã được xây dựng.</p> <p><b>Post-condition:</b> hỗ trợ doanh nghiệp chọn lọc ra các hồ sơ thỏa mãn các yêu cầu phổ biến của công việc đó.</p>

**Bảng 11. Bảng mô tả use case JobZoom framwork**

### 7.2.1.2 Logical View – Các mức khái niệm và kiến trúc mức Layer của hệ thống JobZoom framework

**Hình 34. Mô hình kiến trúc JobZoom framework**

Kiến trúc mức Layer của JobZoom framework bao gồm:

- ❖ Core entity layer: chứa lớp Entity của core framework bao gồm entity từ: flexible attribute engine, taxonomy engine, dataminining.

**Vấn đề:** nhu cầu sử dụng cần thiết liên tục các entities này từ các component khác bao gồm: component quản lý việc truy xuất cơ sở dữ liệu, component quản lý workflow xử lý tiến trình của framework, truy xuất từ 3 core engine.

**Giải pháp:** tách các entity dùng chung của cả framework vào Layer dùng chung Core entity layer này.

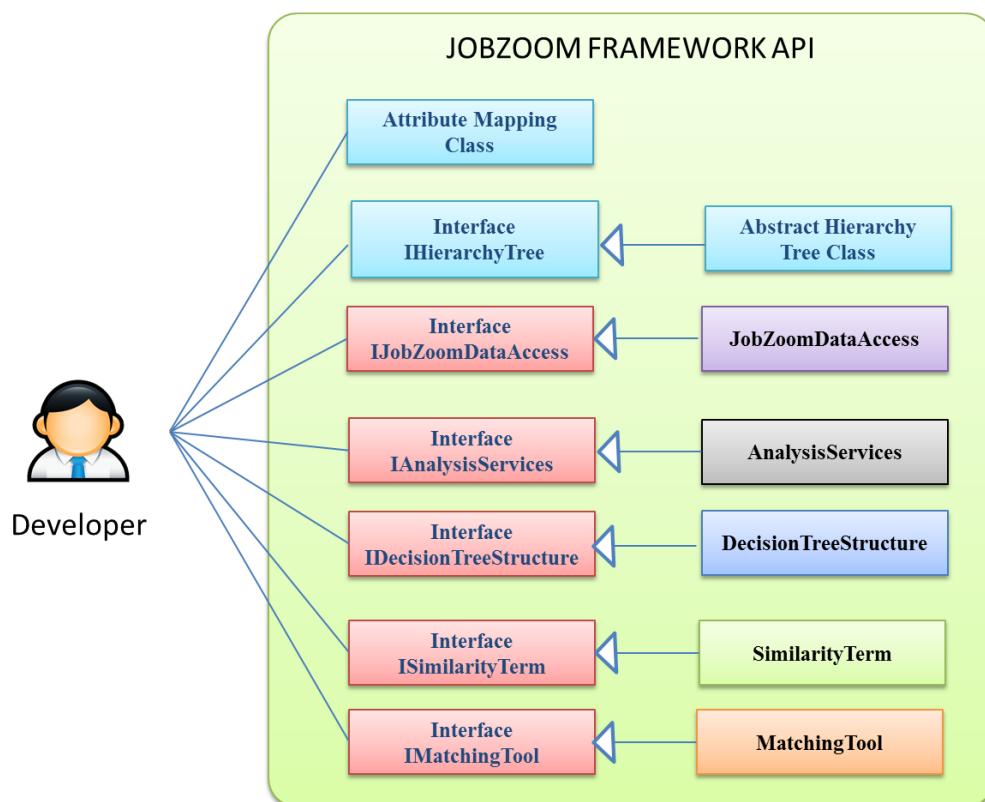
- ❖ Core engine layer: chứa các component xử lý chuyên biệt bao gồm: component Flexiblity engine giải quyết bài toán tổ chức thông tin linh động của hệ thống. component Taxonomy engine giải quyết bài toán tổ chức thông tin thành cây phân cấp hỗ trợ việc so khớp thông tin giữa các thông tin (matching) và cuối cùng là component Data mining engine giải quyết bài toán khai thác dữ liệu của hệ thống, cụ thể trong

JobZoom framework cũng cấp giải pháp xây dựng cây quyết định từ việc tổ chức thông tin linh động trên.

❖ Client framework layer: layer này là phần thể hiện của framework library (API) cung cấp giải pháp đưa ra các thư viện lập trình cho lập trình viên implement hệ thống sử dụng kiến trúc của JobZoom framework bao gồm các component:

- *Attribute mapping*: component quản lý việc chuyển đổi (mapping) các thực thể
- *Data access*: component quản lý việc framework truy xuất cơ sở dữ liệu.
- *Framework workflow processing*: component quản lý các trình tự xử lý các core engine để giải quyết các bài toán.
- *Client usage*: component được thiết kế để đưa ra cho lập trình viên sử dụng các thư viện hàm để lập trình hệ thống implement kiến trúc JobZoom framework.

#### 7.2.1.3 Development View - kiến trúc phần mềm dưới góc độ lập trình viên

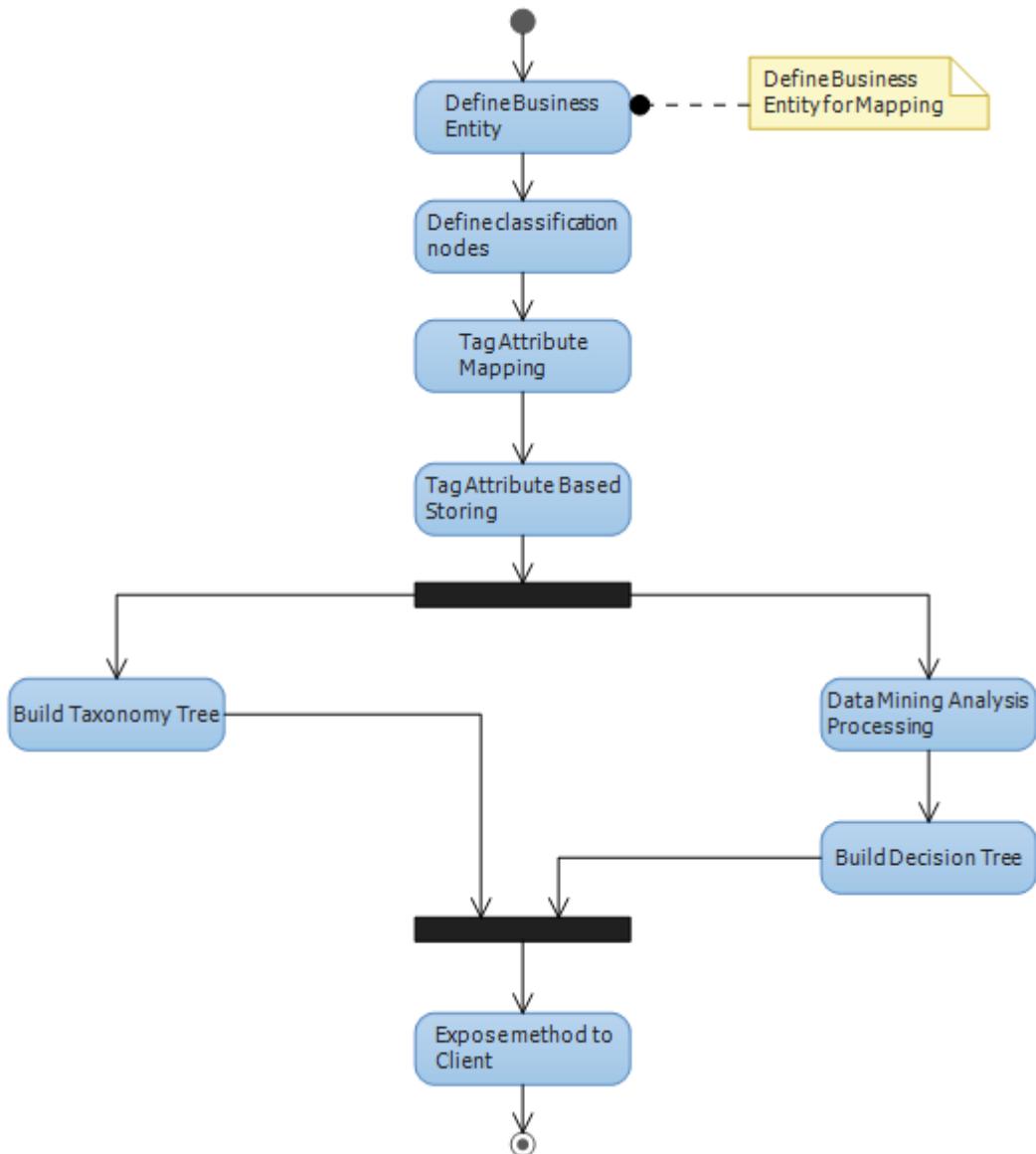


**Hình 35. Kiến trúc phần mềm dưới góc độ lập trình viên**

- ❖ Attribute Mapping Class: Mapping dữ liệu của Business Entity vào kiến trúc
- ❖ Interface IHierarchyTree: Quản lý cây phân cấp taxonomy
- ❖ Interface IJobZoomDataAccess: Quản lý truy xuất dữ liệu

- ❖ Interface IAnalysisServices: Quản lý phân tích dữ liệu trên Analysis Services và xuất dữ liệu từ OLAP Database ra Database Engine
- ❖ Interface IDecisionTreeStructure: Cấu trúc dữ liệu cho cây quyết định
- ❖ Interface ISimilarityTerm: Độ tương quan giữa các thuộc tính
- ❖ Interface IMatchingTool: Hệ thống so khớp của kiến trúc.

#### 7.2.1.4 Process view - các quy trình, các bước hoạt động của JobZoom framework



Hình 36. Quy trình hoạt động của JobZoom framework

*Define business entity:* các website tìm kiếm việc đều có các business entity riêng được tổ chức theo những schema riêng theo chiến lược xây dựng phần mềm,

chiến lược về business riêng. Để website đó có thể sử dụng kiến trúc của JobZoom framework để giải quyết ba bài toán trọng điểm đã trình bày đòi hỏi giải pháp phải chuyển đổi (mapping) các dữ liệu từ dạng schema có cấu trúc sang tổ chức thông tin theo Tag attribute. Phần 7.2.2.2 sẽ trình bày rõ hơn giải pháp mapping các schema. Vì vậy bước đầu tiên của quy trình xử lý framework đòi hỏi lập trình viên phải định nghĩa các Business entity chuẩn bị cho giải pháp mapping dữ liệu.

*Define classification nodes:* như đã trình bày trong phần 3.4 về cơ sở so khớp thông tin với nhau và phần 4.3 về giải pháp so khớp các thông tin được tổ chức theo cấu trúc cây đòi hỏi phải định nghĩa các thuộc tính phân lớp (classification node).

*Attribute mapping:* bước xử lý mapping các business entity schema sang tổ chức thông tin và lưu trữ dạng Tag attribute.

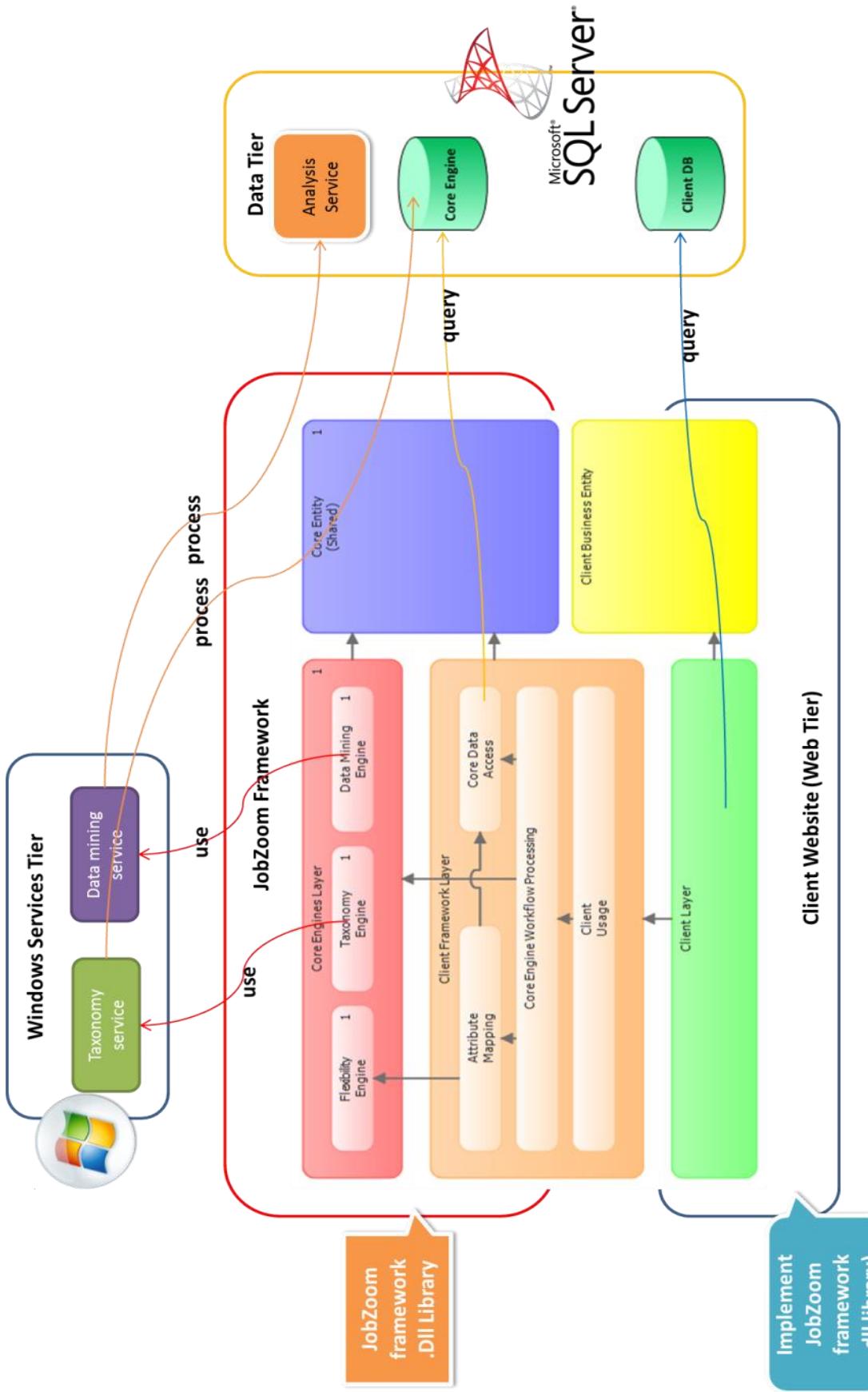
*Tag based storing:* xử lý lưu trữ dữ liệu dưới dạng Tag attribute.

*Build hierarchy tree:* xử lý tổ chức thông tin có cấu trúc cây từ tổ chức thông tin theo Tag attribute.

*Data mining analysis processing:* đồng thời trong lúc đó xử lý việc khai thác dữ liệu qua giải pháp JobZoom data mining service.

*Build decision tree:* xây dựng cây quyết định từ thông tin được khai thác dữ liệu.

#### 7.2.1.5 Physical view - kiến trúc dưới góc độ triển khai hệ thống vật lý



Hình 37. Kiến trúc dưới góc độ triển khai hệ thống vật lý

Như đã trình bày thiết kế kiến trúc JobZoom framework dưới góc độ chia theo các Layer ở phần 7.2.1.2, ta thấy tùy thuộc đặc điểm, nhiệm vụ của mỗi component mà ta nhóm lại các component thuộc layer nào. Cụ thể trong thiết kế JobZoom framework ta có 3 layers: Core engine layer, Framework library layer, Core entity layer.

*Core engine layer:* bao gồm các component xử lý 3 bài toán chính, trong hệ thống được đặt tên là các engines. Trong đó có 2 engines: Taxonomy, Data mining engine phải xử lý hầu hết các lượng thông tin của bài toán kiến trúc tổ chức thông tin. Với đặc điểm trên, nhu cầu về năng lực xử lý tính toán là một yếu tố quan trọng được xem xét tới. Vì vậy giải pháp triển khai engine này dưới dạng các Windows services chạy trên Windows Client (Desktop, Server..) là giải pháp lựa chọn.

*Core entity layer:* bao gồm component chứa các entity của framework, được gọi là core entity. Đặc điểm các core entity là được định nghĩa dưới dạng các entity classes. Vì vậy chúng được đóng gói trong thư viện lập trình framework library (API) để cung cấp cho lập trình viên sử dụng.

*Framework management layer:* chứa các component liên quan đến việc hoạt động, quy trình của framework và có thể cung cấp các lời gọi hàm được thể hiện thông qua thư viện lập trình JobZoom framework libarry cung cấp cho lập trình viên. Vì vậy layer này được đóng gói dưới dạng thư viện lập trình (library).

Với những đặc điểm của những component được đặc tả như trên, giải pháp thiết kế kiến trúc triển khai trên các hệ thống vật lý như sau:

❖ **Windows services tier:** triển khai các engines: Data mining, taxonomy thành các Windows services: JobZoom data mining service và JobZoom taxonomy service để xử lý 2 bài toán về khai thác dữ liệu và tổ chức, lưu trữ thông tin linh động tận dụng sức mạnh về năng lực xử lý của hệ thống máy chủ.

❖ **Data tier:**

Xét trong hệ thống framework: thì Data mining engine và Taxonomy engine xử lý thông tin qua truy xuất dữ liệu từ hệ quản trị cơ sở dữ liệu Microsoft SQL Server truy Cơ sở dữ liệu Core database và sự dịch vụ Analysis Service của Microsoft SQL Server để khai thác dữ liệu (Data mining).

Xét về hệ thống website tuyển dụng: cũng có cơ sở dữ liệu riêng của họ.

Vì vậy về Data Tier sẽ bao gồm: hệ quản trị cơ sở dữ liệu, cơ sở dữ liệu của framework gọi là Core database, cơ sở dữ liệu của website tuyển dụng được gọi là Client database.

❖ JobZoom framework library tier: như mô tả chi tiết ở trên về thư viện lập trình được đóng gói thành một thư viện lập trình. Trong bài toán kiến trúc của JobZoom thì thư viện này được đóng gói dưới dạng file .dll. Thư viện này được website tuyển dụng sử dụng. Vì vậy thư viện này phụ thuộc vào Tier vật lý của website tuyển dụng implement thư viện đó.

❖ Client website tier (web tier): xét về góc độ triển khai vật lý của website tuyển dụng. Họ có thể triển khai tùy ý phụ thuộc vào chiến lược xây dựng và tổ chức website đơn vị quản lý website.

#### 7.2.1.6 Giải quyết bài toán cải thiện hiệu năng xử lý của JobZoom framework

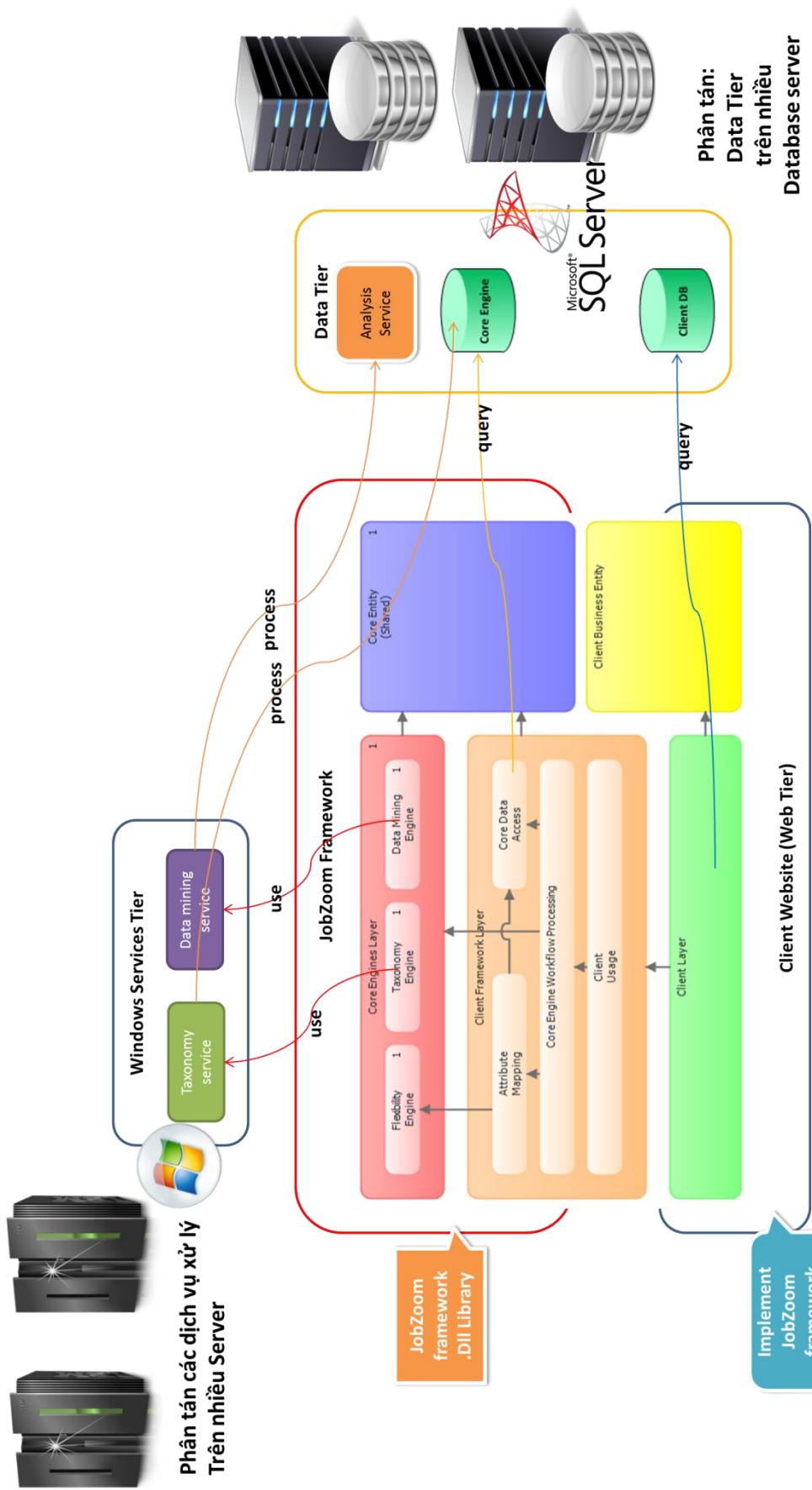
❖ Phân tán việc lưu trữ dữ liệu: trong một hệ thống kìm kiềm việc làm với nhiều người sử dụng, các doanh nghiệp, các người tìm kiếm việc. Trong xu thế web 2.0 và mạng xã hội, JobZoom framework mong muốn được triển khai, ứng dụng vào các mạng xã hội tìm kiếm việc làm. Với nhiều người sử dụng, từ đó lượng thông tin sẽ rất nhiều, cộng với số lượng giao dịch, sử dụng website cũng rất nhiều, sẽ dẫn đến cơ sở dữ liệu khổng lồ, hệ quản trị cơ sở dữ liệu xử lý số lượng thông tin khổng lồ, hiệu năng các máy chủ (server) sẽ giảm. Vì vậy giải pháp kiến trúc của JobZoom framework giải quyết vấn đề lưu trữ dữ liệu bằng cách phân tán dữ liệu ra làm 2 mức: mức kiến trúc framework và mức dữ liệu người dùng. Phân tán Data Tier trên nhiều Database Server.

- Tổ chức lưu trữ mức kiến trúc (Core framework): cơ sở dữ liệu (Core database) xử lý về dữ liệu của framework như xử lý lưu trữ thông tin theo tag và cấu trúc cây, xử lý tổ chức thông tin cho việc khai thác dữ liệu.

- Tổ chức lưu trữ Website người sử dụng: cơ sở dữ liệu riêng của website tìm việc. Từ mức này, có thể triển khai phân tán ra thành nhiều server nhỏ theo cách phân tán.

❖ Phân tán các tiện ích, dịch vụ xử lý thông tin: trong một hệ thống, việc xử lý, khai thác thông tin đòi hỏi hiệu năng. Vì vậy kiến trúc vật lý của framework cho phép triển khai phân tán các dịch vụ này trên nhiều Server.

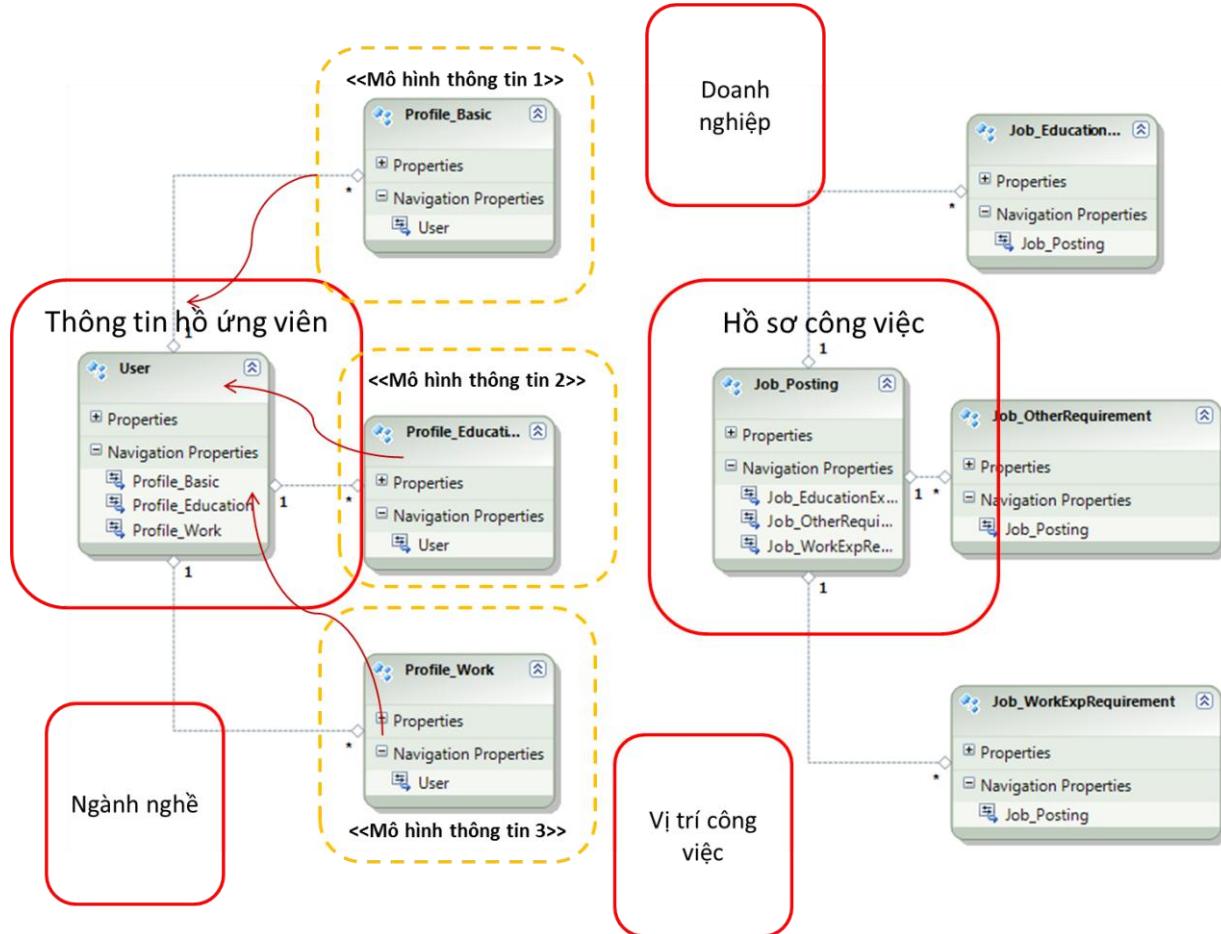
- JobZoom data mining service
- Taxnonomy data mining service



Hình 38. Giải quyết bài toán cải thiện hiệu năng xử lý của JobZoom framework

## 7.2.2 Giải pháp tổ chức và lưu trữ thông tin linh hoạt ứng dụng trong JobZoom framework

### 7.2.2.1 Tổ chức và lưu trữ thông tin linh động (tập các attribute) theo dạng Tag



**Hình 39. Mô hình tổ chức thông tin của JobZoom framework**

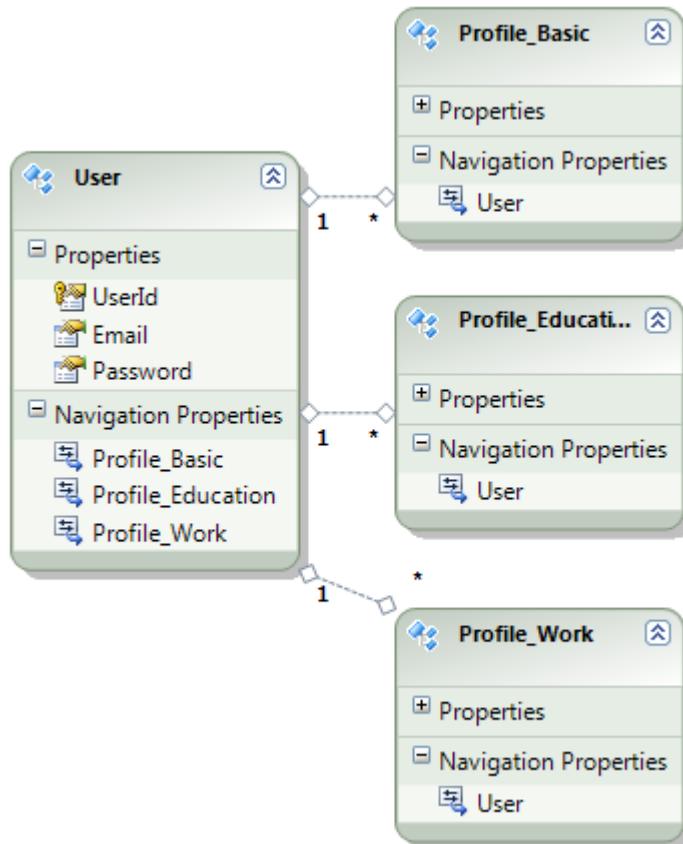
*Vấn đề: về tổ chức thông tin có cấu trúc phức tạp của các Business Entity trong hệ thống tìm việc*

*Trong các hệ thống tìm việc hai khái niệm cơ bản nhất (Business Entity) là:*

- Thông tin hồ sơ ứng viên & Thông tin yêu cầu của một công việc cụ thể.
- Trong mỗi hồ sơ này có rất nhiều loại thông tin khác nhau, tùy theo đặc điểm của mỗi hệ thống tìm việc người ta tổ chức thông tin cho các Business Entity này cũng hoàn toàn khác nhau.

**Ví dụ 1:** Về cách tổ chức thông tin về “Hồ sơ thông tin của người tìm việc” bao gồm tập hợp các loại: thông tin cơ bản, quá trình học vấn, quá trình công tác..

**Ví dụ 2:** Về cách tổ chức thông tin về “Thông tin yêu cầu của một công việc” bao gồm tập hợp các loại: thông tin cơ bản về công việc, các yêu cầu về bằng cấp, các yêu cầu về kỹ năng...



**Hình 40. Tổ chức thông tin yêu cầu một công việc**

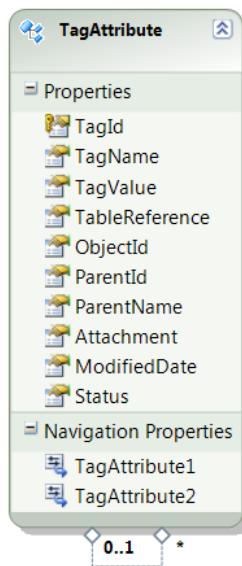
Phân tích hai ví dụ trên, ta rút kết luận rằng việc tổ chức thông tin các Business Entity trong một hệ thống là có cấu trúc phức tạp. Khi các business entity đó được tổ chức thông tin có cấu trúc phức tạp thì vấn đề tạo mối tương quan giữa các business entity càng phức tạp hơn.

Trong phần 6.2 đã trình bày vấn đề khó khăn về tổ chức thông tin phức tạp trong các hệ thống tìm kiếm việc làm và phương pháp giải quyết bằng cách đơn giản và tổ chức lại thông tin cần thiết (các tập attribute) theo dạng Tag.

#### *Giải pháp:*

- Biểu diễn mối quan hệ tập hợp: tập hợp các thông tin giống nhau lại thành một cụm thông tin và phân biệt với cụm khác thông qua ObjectId.
- Biểu diễn mối quan hệ cha con giữa các thông tin trong tập hợp.

- Biểu diễn mối quan hệ phân hoạch lớp của thông tin để phục vụ việc tạo mối tương quan giữa các thông tin với nhau.
- Ví dụ & giải thích



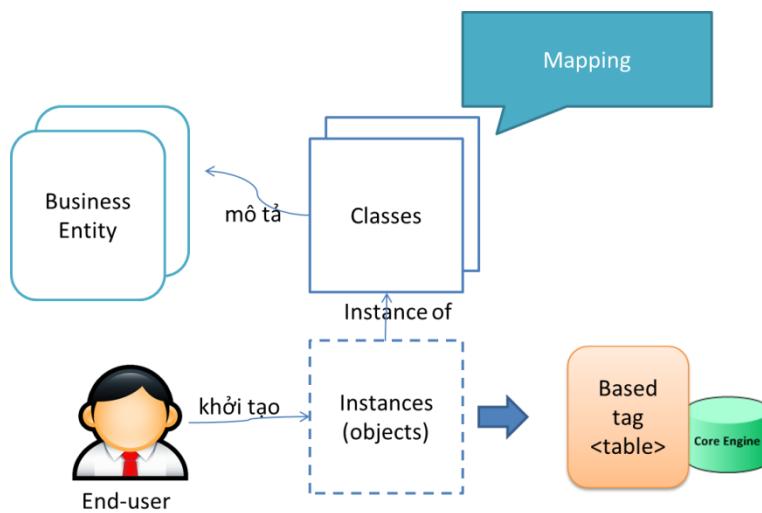
**Hình 41. Cấu trúc TagAttribute**

#### 7.2.2.2 Mapping - Giải pháp mô hình hóa thông tin cấu trúc của người dùng thành tập các attribute dạng Tag

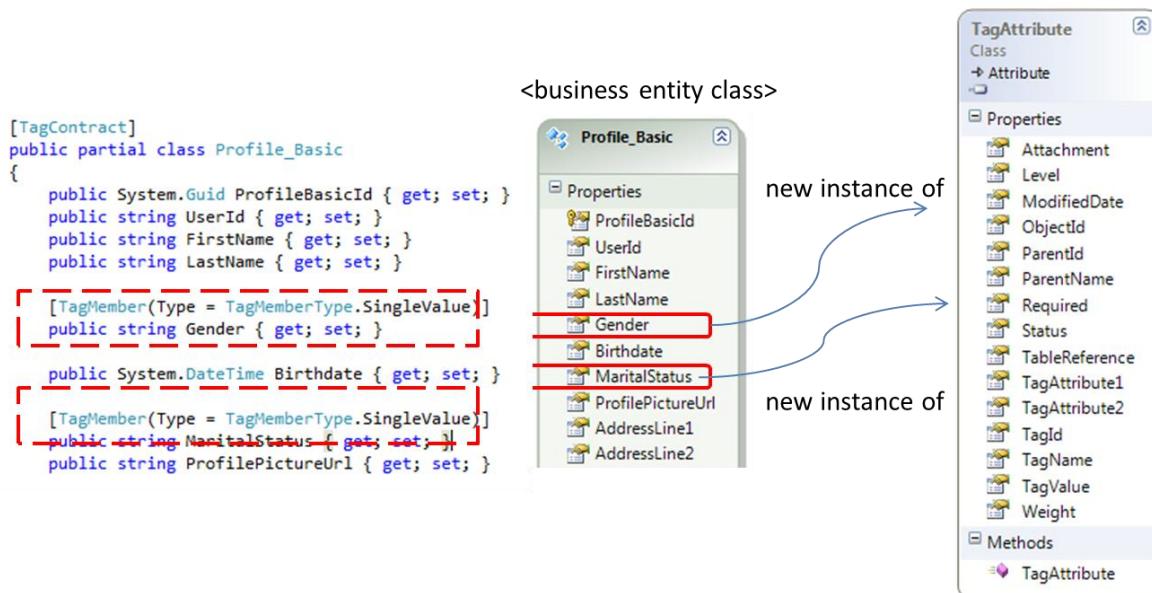
**Vấn đề:** trong phần 7.2.2.1 đã trình bày vấn đề về các website tìm kiếm việc phải tổ chức thông tin có cấu trúc phức tạp và phương pháp giải quyết là tổ chức lại thông tin dạng Tag. Vấn đề đặt ra framework hỗ trợ việc chuyển đổi thông tin cấu trúc thành dạng Tag thế nào?

##### *Giải pháp:*

❖ Dưới góc độ thiết kế kiến trúc hệ thống nói chung và góc độ lập trình viên thì thông tin của một ứng dụng phần mềm nói chung thường được thể hiện dưới hai hình thức; tổ chức thông tin các Business entity (entity classes) và tổ chức thông tin Cơ sở dữ liệu (relational tables). Cả hai có mối quan hệ chắc chắn với nhau. Các Business Entity được định nghĩa trong các Class; mỗi instance của Class này chính là thông tin dữ liệu do người dùng khởi tạo. Sau đó các objects được hệ thống xử lý và lưu trữ xuống các table cơ sở dữ liệu, ở đây cụ thể sẽ được lưu trữ xuống table TagAttribute (tổ chức thông tin lưu trữ dưới dạng Tag)

**Hình 42. Giải pháp mapping dữ liệu**

- ❖ Vì vậy giải pháp mapping giúp mô hình hóa tổ chức thông tin có cấu trúc (các business entity) sẽ được thực hiện thông qua các Classes định nghĩa các business entity này (Business entity classes)
- ❖ Giải pháp mapping cụ thể mà framework dùng kỹ thuật khai báo (declaration) trong 1 business entity class thì các Property nào sẽ được chuyển đổi (mapping) thành thông tin dưới dạng Tag. Mỗi property của các business entity object sẽ được mapping thành một record của bảng TagAttribute.

**Hình 43. Mapping dữ liệu khi sử dụng JobZoom framework**

### 7.2.3 Taxonomy - giải pháp ứng dụng cấu trúc phân loại và lưu trữ thông tin ứng dụng vào JobZoom framework

#### 7.2.3.1 Tổ chức thông tin theo cây phân cấp

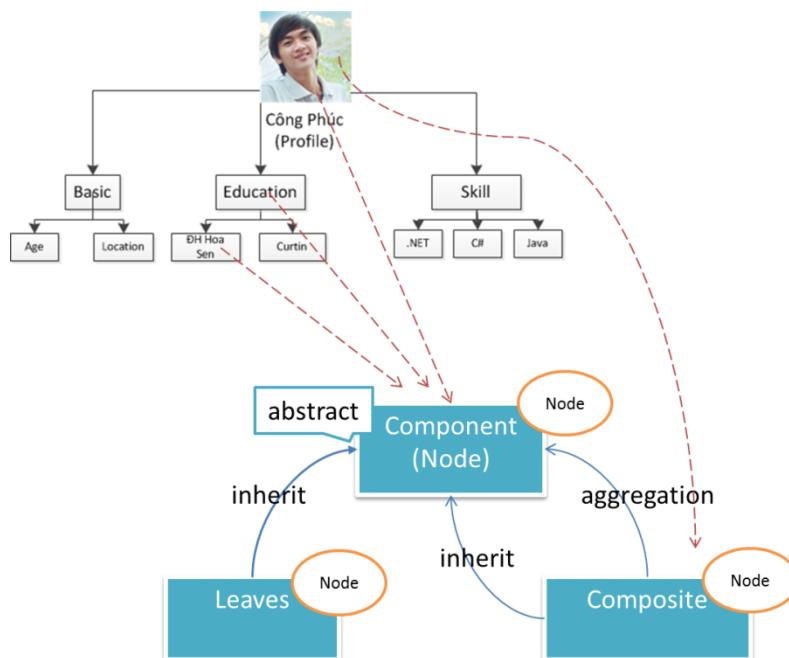
**Vấn đề:** *Làm thế nào để tổ chức thông tin theo cây phân cấp hiệu quả nhất trong phạm vi JobZoom*

❖ Theo hướng giải quyết 6.2 (lưu trữ dưới dạng tag kết hợp tổ chức thông tin theo taxonomy) và trong kiến trúc tổng quan của JobZoom Framework trình bày các Use case của một hệ thống thông tin tìm việc hướng đến hai đối tượng cụ thể (người dùng đầu cuối): người tìm kiếm việc (Job Seeker) và Doanh nghiệp tuyển dụng (Employer).

❖ Vì vậy thông tin tổ chức, lưu trữ tập trung mô tả 2 nguồn thông tin chính: Job Seeker và Employer.

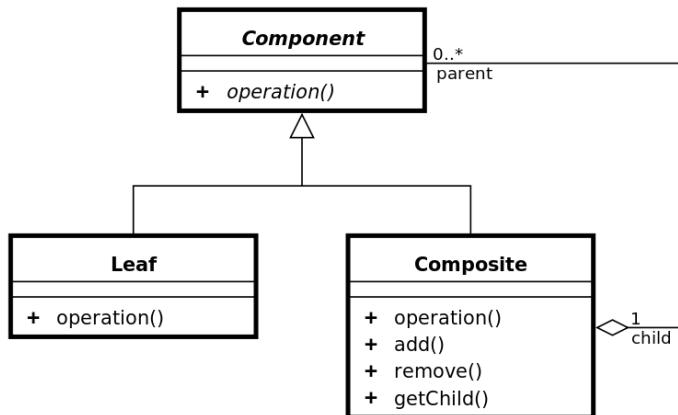
#### *Giải pháp:*

❖ *Theo hướng giải pháp tổ chức cây phân cấp trong phần 4.1, tổ chức thông tin phân cấp hierarchy theo 3 cấp bao gồm: Root node (level 1), các Node phân lớp - Classification nodes (level 2), node lá (level 3) kết hợp giải pháp tạo mối tương quan giữa các node lá (similar term).*



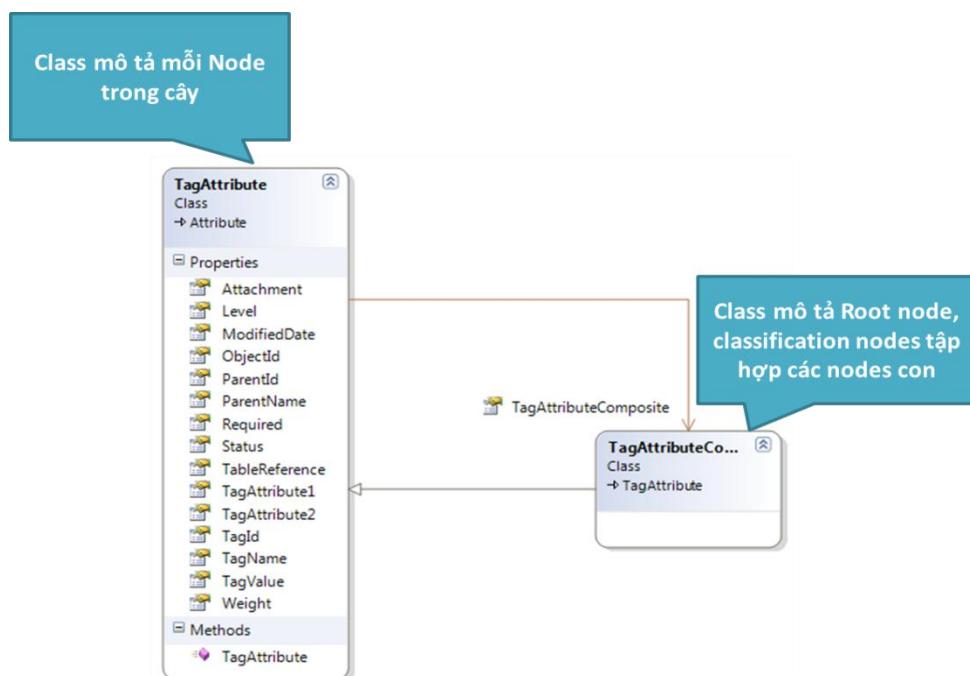
Hình 44. Tổ chức thông tin phân cấp

- ❖ Áp dụng Composite pattern (design pattern) vào thiết kế các classes của Cây phân cấp



**Hình 45. Composite pattern**

- Áp dụng composite pattern vào bài toán cây phân cấp trong JobZoom như sau:
- Mỗi node (root node, classification nodes, leaf nodes) trên cây, được mô tả bởi TagAttribute Class.
- Các nodes mang tính đặc điểm tập hợp các nodes con (root node, classification nodes)



**Hình 46. Class diagram mô tả tổ chức cây phân cấp**

### 7.2.3.2 Giải pháp tạo mối tương quan giữa các cây phân cấp

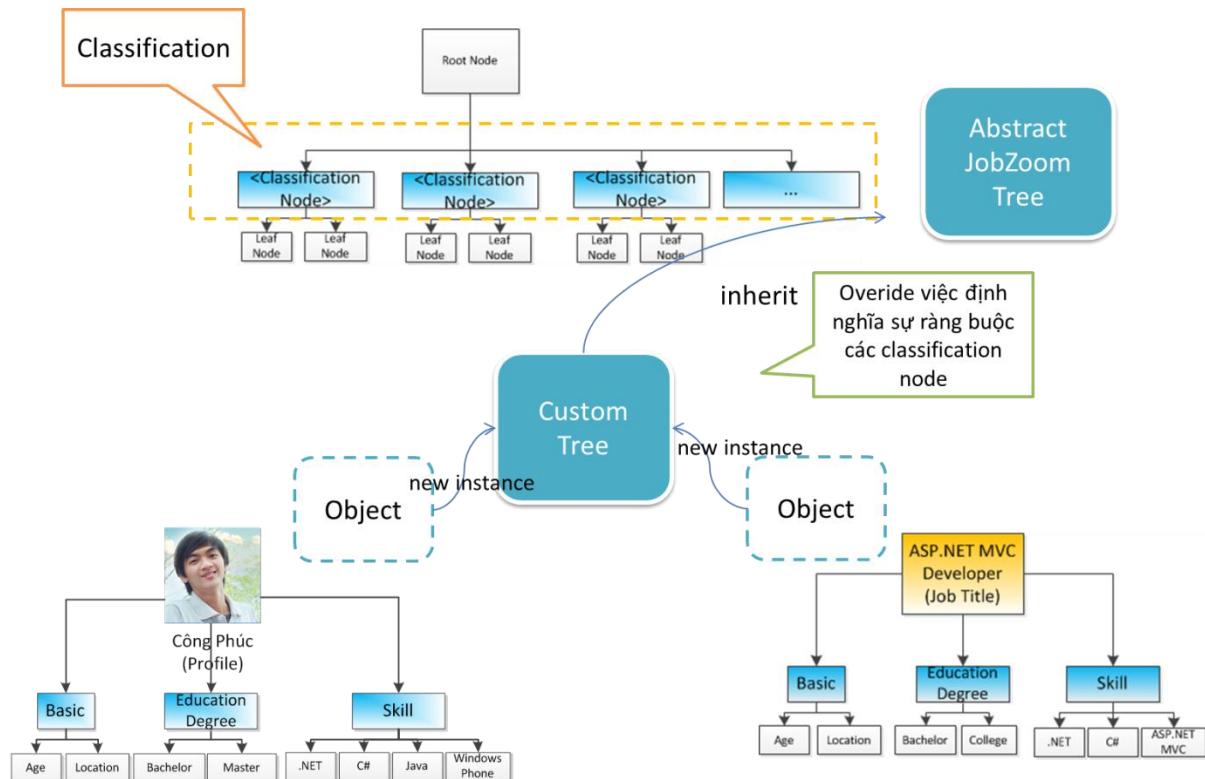
**Vấn đề:** chúng ta đã giải quyết bài toán 1 về việc tổ chức và lưu trữ thông tin linh hoạt giúp khả năng so khớp thông tin. Trong phần 7.2.2.2 đã trình bày giải pháp mapping các entity schema để phục vụ tổ chức thông tin theo dạng Tag. Trong bài toán JobZoom nhận diện hai thông tin chính: thông tin người tìm việc (job seeker) và thông tin về công việc cụ thể (job title requirement).

Để hai thông tin này có thể so khớp nhau đòi hỏi việc tổ chức thông tin hai thông tin này phải có các Node phân hoạch lớp (Classification nodes) mô tả các tiêu chí phân lớp của cây phân cấp thông tin (jobseeker profile, job title requirement).

Vấn về mỗi website tìm kiếm việc làm đều có chiến lược định nghĩa các thuộc tính phân lớp các thông tin theo cây phân cấp khác nhau. Ví dụ có website có chiến lược chia thông tin theo: thông tin cơ bản (basic), bằng cấp, kỹ năng. Có website thì có doanh nghiệp thì định nghĩa theo các tiêu chí: thông tin cơ bản (basic), quá trình học tập (education), quá trình làm việc (work), kỹ năng (skill).

**Vấn đề** *vậy làm sao kiến trúc tổ chức thông tin theo cây phân cấp cho phép người xây dựng website có thể định nghĩa các tiêu chí so khớp, tức là các thuộc tính phân lớp (classification nodes) đảm bảo khả năng so khớp thông tin các thông tin trên các tiêu chí phân lớp đó.*

*Giải pháp:*



**Hình 47. Các thuộc tính phân lớp đảm bảo khả năng so khớp thông tin**

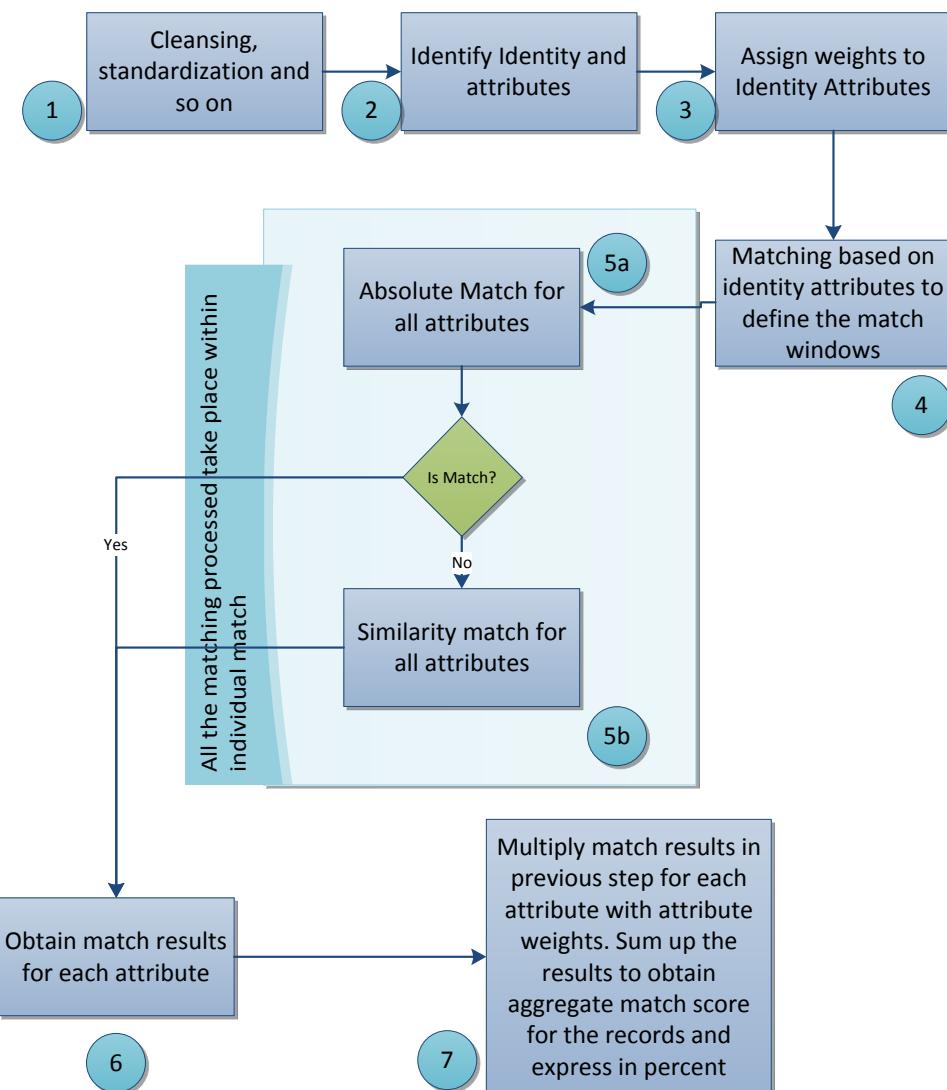
- ❖ Từ giải pháp thiết kế trên 7.2.3.1 định nghĩa cấu trúc cây phân cấp thông tin chung. Để tăng khả năng mở rộng Class thiết kế này, chuyển đổi sang Abstract Class.
- ❖ JobZoomHierarchyTree định nghĩa giao diện:
  - Định nghĩa nội dung các Classification Nodes
- ❖ Tùy theo mục đích sử dụng cây phân cấp này vào việc mô tả thông tin nào (ví dụ jobseeker profile hay jobtitle requirement..) mà Website thiết kế Class mô tả cây phân cấp thông tin của riêng mình (tạm gọi là Custom Hierarchy Tree).
- ❖ Tạo nên các thông tin riêng theo cấu trúc cây phân cấp từ Instance của Custom Hierarchy Tree (ví dụ như tạo 2 Object về <jobseeker profile tree> và <jobtitle profile tree> (trong ví dụ hình trên website quản lý hai thông tin về Profile Công Phúc và Job ASP.NET MVC)

#### 7.2.4 Matching tool - Giải pháp so sánh độ tương quan giữa các thông tin được tổ chức theo cấu trúc cây phân cấp.

- ❖ **Vấn đề:** So khớp giữa các thông tin được tổ chức theo cấu trúc phân cấp

Thông tin về người tìm việc và yêu cầu tuyển dụng được tổ chức dưới dạng taxonomy đa cấp. Việc so khớp các thông tin này với nhau nhằm hỗ trợ doanh nghiệp đánh giá ứng viên và hỗ trợ ứng viên nhận ra các kỹ năng, những thuộc tính còn thiếu cần bổ sung ứng với một vị trí làm việc tại một doanh nghiệp cụ thể.

❖ **Giải pháp:**



**Hình 48. Mô hình giải pháp so sánh độ tương quan giữa các thông tin trong JobZoom**

❖ **Bước 1:** Làm sạch và chuẩn hoá dữ liệu.

Lấy thông tin về hồ sơ ứng viên và yêu cầu tuyển dụng, chuẩn hoá và làm sạch thông tin chuẩn bị cho việc so khớp.

❖ **Bước 2:** Xác định các thuộc tính cần so khớp

Dựa vào những thuộc tính miêu tả yêu cầu tuyển dụng để chọn ra những thuộc tính cần lấy ra từ hồ sơ ứng viên

❖ **Bước 3:** Xác định Weight (tầm quan trọng) cho từng thuộc tính (Weight là số nguyên có giá trị từ 1-5)

❖ **Bước 4:** Dựa vào các thuộc tính cần so khớp định nghĩa các tiến trình so khớp

❖ **Bước 5:**

Đầu tiên, các thuộc tính trong CV sẽ được so khớp theo phương pháp “Absolute Match” để so sánh với các thuộc tính trong yêu cầu tuyển dụng. Nếu thuộc tính A trong CV hoàn toàn khớp với thuộc tính B trong yêu cầu tuyển dụng, thì kết quả so khớp cho thuộc tính này là 100%. Nếu A không khớp hoàn toàn với B, sẽ tiếp tục thực hiện “Similarity Match” – so sánh có mức độ tương quan giữa các thuộc tính. Kết quả so khớp là mức độ tương quan giữa thuộc tính A và B có đơn vị tính là phần trăm.

❖ **Bước 6:** Tập hợp kết quả so sánh cho từng thuộc tính

❖ **Bước 7:** Ứng với mỗi kết quả so khớp thuộc tính, kết quả so khớp của từng thuộc tính sẽ được nhân với weight (tầm quan trọng) của mỗi thuộc tính, sau đó tổng hợp để đưa ra số điểm so khớp giữa CV ứng viên và yêu cầu tuyển dụng.

$$Score (\%) = \frac{\sum \text{Attribute Score (\%)} * \text{Attribute Weight}}{\sum \text{Attribute Weight}}$$

#### 7.2.4.1 Absolute Match

Absolute Match là so khớp tuyệt đối, A phải hoàn toàn khớp với B. Kết quả so khớp giữa A và B là 0% hoặc 100%

**Giá trị đầu vào:** Tập hợp TagName, TagValue và CompareType của yêu cầu tuyển dụng, tập hợp TagName và TagValue của CV ứng viên.

**Giá trị đầu ra:** Kết quả so khớp bằng phương pháp “Absolute Match”

Ứng với kiểu dữ liệu rời rạc (Discrete), CompareType là equals (=)

Ứng với kiểu dữ liệu thứ tự (Ordered), CompareType là GreaterThan (>=)

Ứng với kiểu dữ liệu liên tục (Continuous), có thể áp dụng mọi loại CompareType

Hệ thống so khớp sẽ tiến hành so khớp TagName, nếu TagName giữa yêu cầu tuyển dụng và TagName của ứng viên giống nhau, sẽ đi so sánh các TagValue theo CompareType ra cho ra kết quả.

Ví dụ: Trong yêu cầu tuyển dụng, tập thuộc tính có duy nhất một thuộc tính có giá trị TagName = “Age”, TagValue = “19”, CompareType = “GreaterThan”, ứng viên nộp hồ sơ dự tuyển có thuộc tính TagName = “Age”, TagValue = “21”. Đầu tiên, hệ thống sẽ so khớp TagName với nhau, hai TagName này hoàn toàn giống nhau nên sẽ chuyển qua so sánh TagValue, “21 >= 18”, kết quả của biểu thức so sánh này là true, nên kết quả so khớp giữa ứng viên này và yêu cầu tuyển dụng là 100%. Ngược lại, nếu CompareType là “EqualTo” thì biểu thức “21 = 18” cho kết quả sai, kết quả so khớp giữa ứng viên và yêu cầu tuyển dụng là 0% và được chuyển qua so khớp bằng Similarity Match

#### **7.2.4.2 Similarity Match**

So khớp có mức độ tương quan, A tương quan B và có một số điểm tương ứng. Kết quả so khớp giữa A và B trong khoảng từ 0% đến 100%

**Giá trị đầu vào:** Tập hợp TagName của yêu cầu tuyển dụng, tập hợp TagName của CV ứng viên, tên công việc tuyển dụng đang xét và kết quả

**Giá trị đầu ra:** Kết quả so khớp bằng phương pháp “Similarity Match”

**Điều kiện:** Kết quả Absolute Match bằng KHÔNG, kiểu dữ liệu rời rạc

Hệ thống tiến hành lấy về mức độ tương quan giữa hai từ khoá TagName của nhà tuyển dụng và TagName của ứng viên, với góc nhìn là công việc ứng viên đang dự tuyển. Hệ thống sẽ trả về mức độ so khớp giữa hai TagName này, đây cũng là kết quả của Similarity Match

#### **7.2.5 Data Mining engine - giải pháp ứng dụng Decision Tree vào kiến trúc JobZoom framework**

##### **❖ Pivot transformation:**

Trong quá trình xây dựng cây quyết định thông qua giải pháp kiến trúc lưu trữ và so khớp thông tin, kỹ thuật Pivot transformation được chúng tôi ứng dụng như sau:

Thông tin cho từng vị trí làm việc được lưu trữ theo dạng Tag. Ví dụ dữ liệu của thông tin tuyển dụng gồm vị trí đăng tuyển (Job Title), tên công ty (Company), và một tập hợp các yêu cầu/ tiêu chí, với mỗi hàng là một tiêu chí cụ thể.

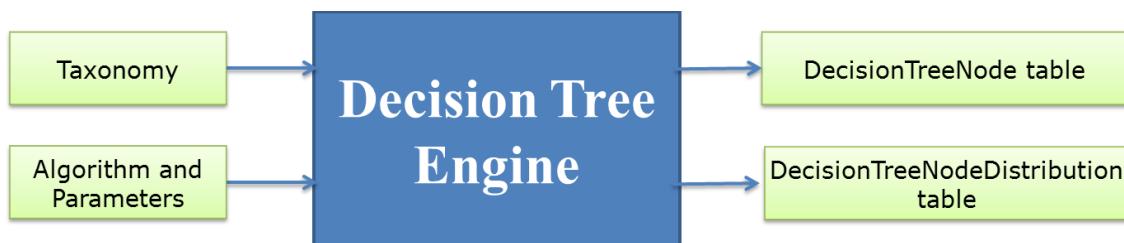
Usage	SetKey	PivotKey
Column Name	Job Title	Tag
<b>Data Records</b>	Developer	OOP
	Developer	Software design
	Developer	DBMS
	Developer	.Net Framework
	Tester	Testing Technique
	Tester	.Net Framework
	Tester	Automation Testing
	Tester	DBMS

Bảng 12. Bảng dữ liệu các thông tin tuyển dụng trước khi Pivot trên cột Tag

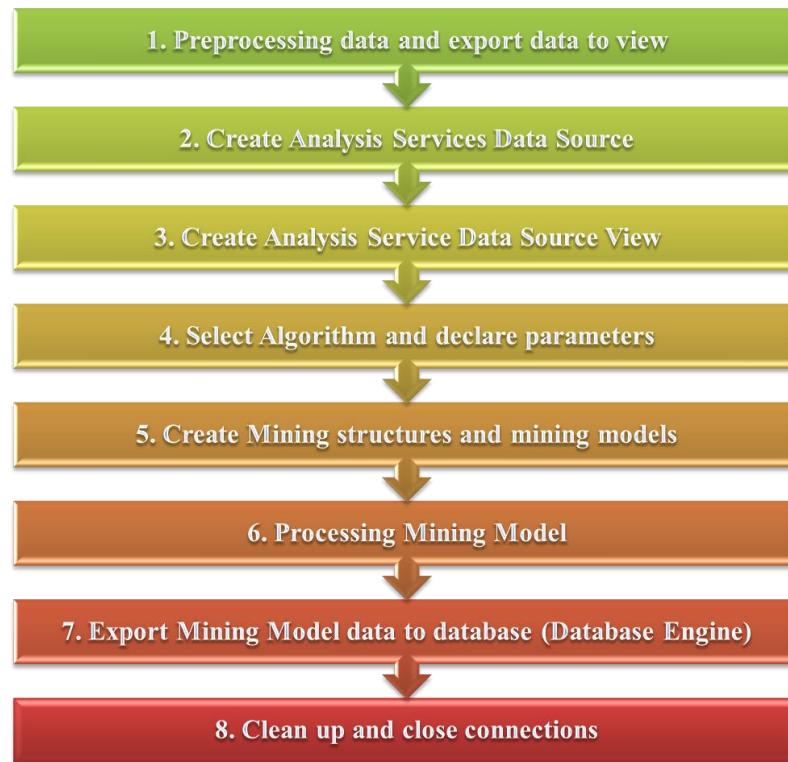
Dựa vào thông tin vị trí đăng tuyển (JobTitle) làm khóa, sẽ chuyển đổi các tag thành tên cột, giá trị của các cột này là kiểu boolean thể hiện có hay không yêu cầu tiêu chí này đối với vị trí đăng tuyển. Bảng sau minh họa kết quả quá trình xử lý dữ liệu.

Column Name	Job Title	OOP	Software design	DBMS	.Net Framework	Testing Technique	Automation Testing
Data Records	Developer	True	True	True	True	False	False
	Tester	False	False	True	True	True	True

Bảng 13. Bảng kết quả sau khi Pivot Transformation



Hình 49. Dữ liệu đầu vào và đầu ra Decision Tree Engine của JobZoom framework



Hình 50. Các bước xây dựng cây quyết định trên JobZoom framework

<b>DecisionTreeNode</b>		
Column Name	Data Type	Allow Nulls
NODE_ID	varchar(100)	<input type="checkbox"/>
MODEL_NAME	varchar(100)	<input type="checkbox"/>
NODE_TYPE	int	<input type="checkbox"/>
NODE_CAPTION	nvarchar(256)	<input checked="" type="checkbox"/>
CHILDREN_CARDINALITY	int	<input type="checkbox"/>
PARENT_ID	varchar(100)	<input checked="" type="checkbox"/>
NODE_DESCRIPTION	ntext	<input checked="" type="checkbox"/>
NODE_RULE	ntext	<input checked="" type="checkbox"/>
MARGINAL_RULE	ntext	<input checked="" type="checkbox"/>
NODE_PROBABILITY	float	<input type="checkbox"/>
MARGINAL_PROBABILITY	float	<input checked="" type="checkbox"/>
NODE_SUPPORT	float	<input type="checkbox"/>
MSOLAP_MODEL_COLUMN	nvarchar(256)	<input checked="" type="checkbox"/>
MSOLAP_NODE_SCORE	float	<input checked="" type="checkbox"/>
MSOLAP_NODE_SHORT_CAPTION	nvarchar(256)	<input checked="" type="checkbox"/>
ATTRIBUTE_NAME	nvarchar(256)	<input checked="" type="checkbox"/>

<b>DecisionTreeNodeDistribution</b>		
Column Name	Data Type	Allow Nulls
NODE_ID	varchar(100)	<input type="checkbox"/>
ATTRIBUTE_NAME	nvarchar(256)	<input checked="" type="checkbox"/>
ATTRIBUTE_VALUE	nvarchar(7)	<input type="checkbox"/>
SUPPORT	float	<input type="checkbox"/>
PROBABILITY	float	<input type="checkbox"/>
VARIANCE	float	<input type="checkbox"/>
VALUETYPE	int	<input type="checkbox"/>

Hình 51. Cấu trúc bảng DecisionTreeNode và DecisionTreeNodeDistribution

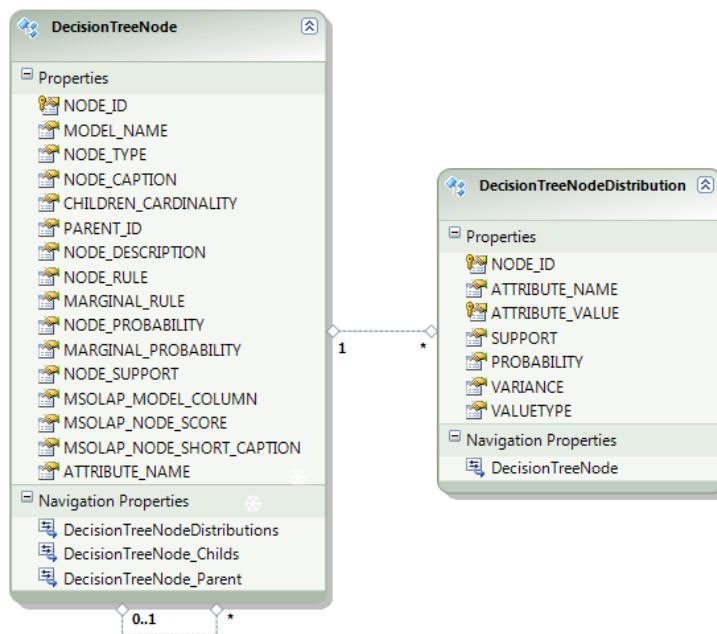
Dữ liệu đầu vào và đầu ra, các bước xây dựng cây quyết định và hai bảng lưu trữ dữ liệu đã khai thác của JobZoom framework hoàn toàn giống với giải pháp kiến trúc

lưu trữ và so khớp của chúng tôi đưa ra. Việc tiến hành xây dựng cây quyết định của JobZoom cũng trải qua các giai đoạn theo thứ tự như sau:

- Tiền xử lý dữ liệu
- Tạo Data source
- Tạo Data source view
- Định nghĩa và truyền giá trị tham số thuật toán
- Tạo Mining Structures và Mining Models
- Tiến hành khai thác dữ liệu
- Xuất dữ liệu đã khai thác vào database engine
- Dọn dẹp và đóng các kết nối.

Vận dụng giải pháp xây dựng cây quyết định để xác định được cung cấp bởi “nền tảng kiến trúc lưu trữ và so khớp thông tin” nói trên, chúng tôi nhận thấy giải pháp này rất khả thi trong vấn đề gợi ý cho người dùng viết CV, cải thiện những điểm yếu của bản thân và giúp doanh nghiệp đăng tuyển yêu cầu công việc một cách dễ dàng hơn so với các website tìm việc hiện nay.

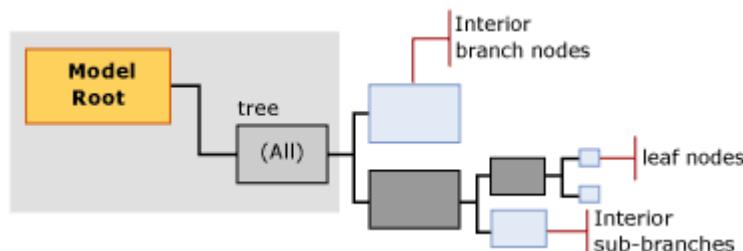
#### *7.2.5.1 Giải pháp hỗ trợ tổ chức thông tin hỗ trợ xây dựng cây quyết định*



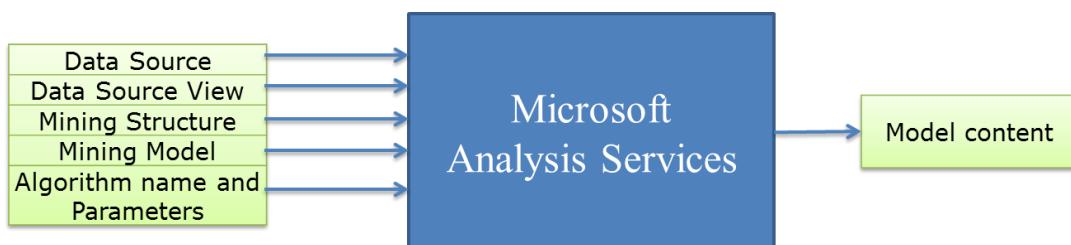
**Hình 52. Cấu trúc dữ liệu cây quyết định #1**

### 7.2.5.2 Giải pháp ứng dụng giải pháp Business Intelligence của Microsoft SQL Server 2008 trong việc xây dựng cây quyết định

Microsoft SQL Server Denali cung cấp một môi trường làm việc với các mô hình khai thác dữ liệu được gọi là “Business Intelligence Development Studio”. Môi trường này bao gồm các thuận toán khai thác dữ liệu và các công cụ xây dựng giải pháp toàn diện cho các dự án khác nhau dễ hơn. Microsoft SQL Server Analysis Services là một thành phần của Microsoft SQL Server Denali chạy trên nền tảng của Microsoft Visual Studio cung cấp nhiều giải pháp, ứng với từng giải pháp là Microsoft tập hợp nhiều thuật toán có thể sử dụng trong khai thác dữ liệu. Sau khi tiến hành sử dụng và đánh giá một số công cụ hỗ trợ tạo cây quyết định hiện có như Weka, Microsoft Analysis Services API,... chúng tôi quyết định lựa chọn Microsoft Analysis Services API để ứng dụng giải quyết bài toán của nhóm. Việc sử dụng Microsoft Analysis Services API giúp chúng tôi có thể dễ dàng xây dựng cây quyết định một cách độc lập, không phụ thuộc vào “Business Intelligence Development Studio”.



Hình 53. Cấu trúc cây quyết định khi sử dụng MS Analysis Services API<sup>26</sup>



Hình 54. Dữ liệu nhập xuất khi sử dụng MS Analysis Services API

Dữ liệu đầu vào	Mô tả
Data source	Khai báo database chứa các bảng dữ liệu

<sup>26</sup> Nguồn: <http://msdn.microsoft.com/en-us/library/cc645758.aspx>

<b>Data source view</b>	Ứng với Data source, lựa chọn các table hay view dùng để khai thác với các phương pháp khai thác dữ liệu được cung cấp sẵn trên MS Analysis Services
<b>Mining structure</b>	Cấu trúc của dữ liệu khai thác (liên kết với Data Source View)
<b>Mining model</b>	Khai báo một model từ Mining Structure để tiến hành khai thác dữ liệu
<b>Algorithm name and Parameter</b>	Tên thuật toán và các tham số cần cung cấp cho thuật toán. Khai báo kèm theo Mining model

**Bảng 14. Bảng mô tả dữ liệu đầu vào khi sử dụng MS Analysis Services**

Model content	Mô tả
<b>MODEL_CATALOG</b>	Tên database chứa model
<b>MODEL_NAME</b>	Tên model
<b>ATTRIBUTE_NAME</b>	Tên thuộc tính dự đoán (phân lớp)
<b>NODE_NAME</b> <b>(NODE_UNIQUE_NAME)</b>	Mã node
<b>NODE_TYPE</b>	Kiểu node. <i>Kiểu 1 - Model:</i> node gốc của model <i>Kiểu 2 - Tree:</i> node gốc của cây, có NODE_CAPTION là “All” <i>Kiểu 3 - Interior:</i> node có node con <i>Kiểu 4 - Distribution:</i> node lá <i>Kiểu 5 - Regression tree:</i> node gốc của cây đê quy, có NODE_CAPTION là “All”
<b>NODE_CAPTION</b>	Tên đầu đề của node Ví dụ: “.NET = True”: các node con của node này bao gồm chính nó đều có các dòng dữ liệu chứa thuộc tính .NET
<b>CHILDREN_CARDINALITY</b>	Số lượng node con cấp 1

	Node lá luôn có giá trị bằng 0 (KHÔNG)
<b>PARENT_UNIQUE_NAME</b>	Node cha
<b>NODE_DESCRIPTION</b>	<p>Tập hợp các NODE_CAPTION để dẫn đến node hiện tại, phụ thuộc vào độ sâu của cây.</p> <p>Ví dụ: Java = False and C# = True, để đến node này, các dòng record phải không chứa Java và có chứa C#, node có đầu đề là “Java = False” sẽ là node cha của node “C# = True”</p>
<b>NODE_RULE</b>	Mô tả NODE_DESCRIPTION bằng XML
<b>MARGINAL_RULE</b>	Mô tả NODE_CAPTION bằng XML
<b>NODE_PROBABILITY</b>	Xác suất xảy ra node
<b>MARGINAL_PROBABILITY</b>	Xác suất xảy ra node khi xảy ra node cha
<b>NODE_DISTRIBUTION</b>	<p>Sự phân bố giá trị phân lớp của node NODE_DISTRIBUTION là bảng lồng, được lưu trữ trong Mining model. Bảng NODE_DISTRIBUTION bao gồm các thuộc tính sau:</p> <p>ATTRIBUTE_NAME: tên thuộc tính phân lớp</p> <p>ATTRIBUTE_VALUE: các giá trị có thể xảy ra của thuộc tính phân lớp</p> <p>SUPPORT: số lượng trường hợp có thuộc tính phân lớp mang giá trị bằng ATTRIBUTE_VALUE ứng với NODE_SUPPORT.</p> <p>PROBABILITY: bằng tỷ số giữa SUPPORT và NODE_SUPPORT</p>

	VARIANCE: Phương sai. Đối với các giá trị rời rạc, phương sai sẽ bằng 0 (KHÔNG) VALUETYPE: kiểu giá trị (Missing hoặc Discrete)
NODE_SUPPORT	Số lượng dòng dữ liệu tương ứng với node (số trường hợp xảy ra)
MSOLAP_MODEL_COLUMN	Chỉ ra các cột có thể dùng để dự đoán
MSOLAP_NODE_SCORE	Hiển thị đến số điểm tương ứng với node
MSOLAP_NODE_SHORT_CAPTION	Phục vụ cho mục đích hiển thị

Bảng 15. Bảng mô tả dữ liệu đầu ra khi sử dụng MS Analysis Services

#### 7.2.5.3 Các bước triển khai xây dựng cây quyết định



Hình 55. Dữ liệu đầu vào và đầu ra của Decision Tree Engine



**Hình 56. Tiến trình xây dựng cây quyết định của kiến trúc**

#### 7.2.5.3.1 Tiền xử lý dữ liệu

Dữ liệu đầu vào, cụ thể là taxonomy sẽ được Decision Tree Engine tiền xử lý để cho kết quả tối ưu nhất.

- Hệ thống sẽ tạo ra các view tương ứng đối với từng loại đối tượng trong taxonomy. Lọc các dòng dữ liệu lưu trữ “nhu cầu” hay “thông tin đáp ứng nhu cầu”, các “nhu cầu” hay “thông tin đáp ứng nhu cầu” này được phân nhóm theo tên và có một tiền tố (prefix) tương ứng (ví dụ “nhu cầu” có tiền tố là “NC” trước mỗi tên view).

- Dữ liệu lưu trữ dưới dạng taxonomy phải thông qua kỹ thuật Pivot Transformation của SQL Server nhằm giúp cho việc khai thác dữ liệu bằng cây quyết định thực hiện dễ dàng hơn.

- Lọc các ký tự đặc biệt mà MS Analysis Services không thể xử lý bao gồm “.,;`^\_:\*/?"`\${!}+=()[]{}<>” để đảm bảo giữ nguyên tên thuộc tính cho “nhu cầu” và “thông tin đáp ứng nhu cầu”. Việc mã hóa các ký tự đặc biệt khắc phục được khuyết điểm của Microsoft Analysis Services cũng như của “Business Intelligence

Development Studio (BIDS)" (trong BIDS các ký tự đặc biệt không được giữ lại, tên các thuộc tính có ký tự đặc biệt sẽ bị BIDS lược bỏ)

**Đầu vào:** taxonomy

**Đầu ra:** các view lưu trữ cho từng "nhu cầu" và từng "thông tin đáp ứng nhu cầu", có tiền tố để phân biệt giữa view lưu trữ "nhu cầu" và view lưu trữ "thông tin đáp ứng nhu cầu". Các view được xử lý bằng kỹ thuật Pivot Transformation và được mã hoá các ký tự đặc biệt

#### 7.2.5.3.2 Tạo Data source

Tạo data source dựa vào thông số kết nối đến Database Engine lưu trữ taxonomy.

**Đầu vào:** Thông số kết nối đến Database Engine lưu trữ taxonomy

**Đầu ra:** Data source object

#### 7.2.5.3.3 Tạo Data source view

Dựa vào Data Source, tự động dựa vào các tiền tố được tạo ra ở bước tiền xử lý dữ liệu, từ đó tạo ra các Data source view. Các view có tiền tố phù hợp sẽ được tạo view hỗ trợ tạo Mining Structure.

**Đầu vào:** Data source object, prefix được định nghĩa ở bước tiền xử lý dữ liệu

**Đầu ra:** Data source view object

#### 7.2.5.3.4 Định nghĩa các tham số thuật toán

Lựa chọn thuật toán xây dựng cây quyết định và các tham số tương ứng.

*Các tham số thuật toán bao gồm:*

- HoldoutMaxPercent: số lượng dữ liệu dùng để kiểm tra (đơn vị tính: %)
- SCORE\_METHOD: Tên thuật toán dùng để xây dựng cây quyết định (Entropy, Bayesian with K2 Prior, hay Bayesian Dirichlet Equivalent (BDE) Prior)
- COMPLEXITY\_PENALTY: kiểm soát mức độ phát triển của cây
- SPLIT\_METHOD: Phương thức phân nhánh (Binary, Complete, Both)
- MAXIMUM\_INPUT\_ATTRIBUTES: Số thuộc tính đầu vào tối đa
- MAXIMUM\_OUTPUT\_ATTRIBUTES: Số thuộc tính đầu ra tối đa

- **MINIMUM\_SUPPORT:** Chỉ định số lượng tối thiểu các trường hợp phải có các tập phổ biến trước khi thuật toán tạo ra một quy tắc.

**Đầu vào:** KHÔNG

**Đầu ra:** các tham số tham gia vào quá trình xây dựng cây quyết định

#### 7.2.5.3.5 Tạo Mining Structures và Mining Models

Các tham số thuật toán được tạo ở giai đoạn trên sẽ được sử dụng để tạo ra Mining Structure và Mining Model. Data source và Data source view cùng với prefix sẽ xác định các model cần tiến hành xây dựng cây quyết định, các view có trong Data source view có prefix tương ứng sẽ được tiến hành xây dựng cây ở bước tiếp theo nhằm tránh sự lãng phí tài nguyên hệ thống.

**Đầu vào:** Data source, Data source view, tham số thuật toán, prefix được định nghĩa bước tiền xử lý dữ liệu

**Đầu ra:** Mining Structures và Mining Models được tạo

#### 7.2.5.3.6 Tiến hành khai thác dữ liệu

Sau khi các Mining Structures và Mining Models được tạo ra, dữ liệu sẽ được nạp và xử lý, quá trình xử lý có thể kéo dài hàng giờ, hàng tuần hay hàng tháng phụ thuộc vào số lượng dữ liệu. Việc tiến hành khai thác dữ liệu trên một server độc lập với server chính sẽ góp phần đảm bảo hiệu năng của hệ thống khi xây dựng cây quyết định.

**Đầu vào:** Mining Structures và Mining Models được tạo

**Đầu ra:** Mining Models đã được xử lý

#### 7.2.5.3.7 Xuất dữ liệu đã khai thác vào Database Engine

Sau khi dữ liệu đã được phân tích xong sẽ được lưu trữ về Server chính để dễ dàng truy xuất. Các ký tự đặc biệt được mã hoá ở giai đoạn tiền xử lý dữ liệu sẽ được giải mã ở giai đoạn này.

Một “Linked Server” được tạo ra trên Database Engine nhằm kết nối, lấy dữ liệu từ Analysis Services thông qua ngôn ngữ truy vấn MDX và ngôn ngữ truy vấn SQL. “Linked Server” là kỹ thuật liên kết các Server lại với nhau, giúp truy vấn dữ liệu từ một database đặt tại server khác (hoặc một instance khác của SQL Server).

Trong quá trình xuất dữ liệu, **các Mining Model sẽ được liên kết với nhau thông qua node gốc của model** (NODE\_NAME = “0”, NODE\_TYPE = 1), các node gốc của cây (NODE\_CAPTION = “All”, NODE\_TYPE = 2) sẽ được chuyển thành tên của “nhu cầu” hay “thông tin đáp ứng nhu cầu” để dễ dàng truy vấn.

**Đầu vào:** Mining Models đã được xử lý

**Đầu ra:** DecisionTreeNode và DecisionTreeNodeDistribution chứa kết quả khai thác (các cây quyết định đã được liên kết với nhau)

#### 7.2.5.3.8 Dọn dẹp và đóng các kết nối

Sau khi cây quyết định được hoàn thành và lưu trữ trở về database chính, cần dọn dẹp các bảng dữ liệu tạm và đóng các kết nối sử dụng trong quá trình tạo cây.

**Đầu vào:** Kết quả của quá trình xây dựng cây quyết định.

**Đầu ra:** Dọn dẹp dữ liệu và đóng các kết nối.

### 7.3 So sánh Job Zoom với các website tìm việc hiện tại

Mô hình kiến trúc khắc phục những điểm yếu của các website tìm việc hiện có:

STT	Các website tìm việc hiện tại	JobZoom framework
1	Đăng thông tin ứng viên và thông tin về việc làm dưới dạng văn bản	Đăng thông tin ứng viên và yêu cầu tuyển dụng dưới dạng tag, đạt được sự linh hoạt và dễ dàng so khớp hơn.
2	CV theo khuôn mẫu, hạn chế trong việc hỗ trợ người tìm việc mô tả minh chứng cho những kinh nghiệm hay kỹ năng của họ	Hỗ trợ ứng viên viết CV: JobZoom gợi ý những thuộc tính quan trọng giúp ứng viên tạo ấn tượng với CV của mình, đồng thời, những kỹ năng mà JobZoom gợi ý ứng viên chưa có để ứng viên kịp thời bổ sung kiến thức hay kinh nghiệm cho bản thân.
3	Chức năng gợi ý ứng viên bổ sung thông tin hồ sơ còn đơn giản (chỉ gợi ý ứng viên nên cập nhật các thông tin quan trọng như kinh nghiệm làm việc, học vấn), không có hướng dẫn một vị trí làm việc hay ngành nghề mà ứng viên muốn làm việc, không gợi ý những thông tin có thể quyết định cơ	

	hội việc làm cho ứng viên.	
4	Hỗ trợ tốt việc tìm kiếm công việc, hồ sơ ứng viên nhưng chưa thấy được sự so khớp giữa yêu cầu tuyển dụng và hồ sơ của ứng viên.	Có thể so khớp giữa yêu cầu tuyển dụng và hồ sơ của ứng viên
5	Không hỗ trợ nhà tuyển dụng đăng thông tin tuyển dụng	JobZoom hỗ trợ nhà tuyển dụng đăng thông tin tuyển dụng bằng phương pháp khai thác dữ liệu

**Bảng 16. So sánh JobZoom framework với các website hiện tại**

## 8 Đánh giá và hướng phát triển

### 8.1 Những điểm làm được

- Kiến trúc giải quyết được nhu cầu lưu trữ thông tin nhiều, đa dạng, dễ dàng trong việc tìm kiếm và so khớp
- Tổ chức thông tin linh động dưới dạng tag kết hợp taxonomy. Thông tin được lưu trữ dưới dạng cây đa cấp và có mức độ tương quan giữa các tag.
- Hệ thống so khớp linh hoạt, kết hợp mức độ tương quan giữa các tag giải quyết được khuyết điểm của việc so sánh object (A phải khớp hoàn toàn với B).
- Kiến trúc dễ dàng triển khai, mở rộng, đồng thời hiệu năng của hệ thống vẫn được đảm bảo khi áp dụng cây quyết định và hệ thống so khớp để gợi ý cho actor xây dựng cây thông tin nhu cầu và “thông tin đáp ứng nhu cầu” một cách nhanh chóng và hiệu quả.
- Khắc phục hạn chế của “Microsoft Analysis Services<sup>27</sup>”: không hỗ trợ xử lý ký tự đặc biệt. Hoàn toàn không phụ thuộc vào công cụ “SQL Server Business Intelligence Development Studio<sup>28</sup>”
- Ứng dụng kiến trúc vào bài toán JobZoom nhằm đánh giá khả năng tổ chức thông tin linh hoạt, hiệu quả của hệ thống so khớp, đồng thời đánh giá tính mở rộng,

<sup>27</sup> Microsoft Analysis Services: Thành phần của Microsoft SQL Server, dùng để khai thác dữ liệu

<sup>28</sup> SQL Server Business Intelligence Development Studio: Công cụ hỗ trợ khai thác dữ liệu của Microsoft SQL Server

dễ dàng triển khai và tái sử dụng của kiến trúc khi áp dụng trong các lĩnh vực cần lưu trữ và so khớp thông tin.

- Nhóm đã xây dựng và phát triển JobZoom framework ứng dụng những điểm mạnh của bài toán kiến trúc:

- Người tìm việc tìm kiếm công việc phù hợp nhanh chóng, viết CV dễ dàng hơn dựa vào gợi ý từ kết quả khai thác dữ liệu và hệ thống so khớp

- Nhà tuyển dụng dễ dàng đăng tuyển yêu cầu công việc một cách chi tiết và lựa chọn ứng viên phù hợp với vị trí làm việc thông qua hệ thống so khớp và cây quyết định. Hệ thống so khớp giúp đánh giá ứng viên cho kết quả cụ thể (yêu cầu nào đạt hay không đạt và số điểm ứng viên đạt được)

- JobZoom Framework dễ dàng triển khai, mở rộng, kết hợp được với phương pháp khai thác dữ liệu “cây quyết định” tạo nền tảng cải thiện khả năng thích ứng của hệ thống với từng loại ngành nghề.

## 8.2 Những điểm hạn chế

Tuy có những ưu điểm như trên nhưng do kinh nghiệm, thời gian hạn chế nên kiến trúc hệ thống của nhóm chúng tôi cũng còn một số hạn chế cần khắc phục:

- Cây đa cấp còn ở mức trừu tượng thông qua bảng SimilarityTerm để thể hiện mức độ tương quan giữa các tag, việc xác định độ tương quan giữa các tag sẽ được triển khai dễ dàng dựa vào tính dễ dàng mở rộng của kiến trúc.

- Chưa được áp dụng vào thực tế nên bài toán kiến trúc vẫn còn ở mức trừu tượng, cần điều chỉnh để đảm bảo tính khách quan và linh hoạt của hệ thống so khớp. Dữ liệu kiểm tra vẫn chưa nhiều chưa chứng minh được hiệu quả của kiến trúc khi áp dụng vào hệ thống nhiều ngành nghề.

- Chưa triển khai được kiến trúc JobZoom framework trong thực tế nên khó kiểm tra tính khả thi và tốc độ tính toán, so khớp và xây dựng cây quyết định do nhóm phát triển khi triển khai với số lượng người dùng lớn. Tuy nhiên, hiệu năng của hệ thống được cải thiện thông qua việc xây dựng cây quyết định trên một hệ thống độc lập sau đó lưu trữ kết quả về database chính để truy vấn nhanh chóng, duyệt thông tin được lưu trữ dưới dạng cây nhanh hơn lưu trữ dưới dạng văn bản.

### 8.3 Hướng phát triển

#### 8.3.1 Xác thực độ tin cậy của CV và yêu cầu tuyển dụng

Các đối tượng tham gia vào những cộng đồng trực tuyến thường là các đối tượng “ảo” và thường không được xác định tính chính xác, tính xác thực của các thông tin đăng tải. Việc xác thực độ tin cậy của CV và yêu cầu tuyển dụng sẽ giúp hệ thống vận hành tốt hơn, tránh rủi ro và tạo uy tín với người sử dụng. Với kiến trúc linh hoạt của hệ thống ta có thể dễ dàng tích hợp công cụ xác định độ tin cậy của người dùng từ đó xác thực thông tin của họ.

#### 8.3.2 Phát triển Semantic web

Với hệ thống hiện tại của chúng tôi chỉ mô hình lại kết quả phân tích về tương quan ngữ nghĩa (Semantic similarity) giữa các tag và tích hợp vào hệ thống bằng một interface cơ bản. Việc xây dựng và triển khai hệ thống Semantic web sẽ hỗ trợ người dùng tìm kiếm thông tin nhanh chóng, chuẩn xác, thông minh và trả ra kết quả so khớp chính xác hơn.

#### 8.3.3 Đánh trọng số cho các thuộc tính dựa vào độ sâu của taxonomy

Phát triển cây đa cấp dựa vào semantic web, những tag có liên quan với những tag khác, khi matching sẽ được một số điểm tương ứng, số điểm này được dựa vào độ sâu của tag.

#### 8.3.4 Phân tán dữ liệu

Ứng dụng những điểm mạnh của cơ sở dữ liệu phân tán nhằm tăng hiệu năng cho hệ thống.

#### 8.3.5 Công cụ hỗ trợ lựa chọn ứng viên

Công cụ hỗ trợ lựa chọn ứng viên: phỏng vấn trực tuyến, thuyết trình qua internet... có thể được phát triển để hỗ trợ nhà tuyển dụng và ứng viên.

#### 8.3.6 Thu thập thông tin việc làm tự động

Giúp thu thập thông tin việc làm từ các cổng thông tin tìm việc hiện tại.

### 8.3.7 Áp dụng quy trình tuyển dụng vào hệ thống

Kiến trúc đưa ra các ứng viên có số điểm cao cho nhà tuyển dụng quyết định, đây chỉ là một khâu trong quy trình tuyển dụng ứng viên. Hệ thống nên có thêm quy trình tuyển dụng, quản lý quy trình từ lúc ứng viên nộp hồ sơ đến khi ứng viên trở thành nhân viên của công ty, giúp nhà tuyển dụng dễ dàng trong việc quản lý quy trình tuyển dụng của mình.

### 8.3.8 Phát triển kiến trúc phân mềm theo hướng dịch vụ

## TÀI LIỆU THAM KHẢO

- [1] Lior Rokach & Oded Maimon, *Data Mining with Decision Trees: Theory and Applications*, 2008.
- [2] Jamie MacLennan, ZhaoHui Tang & Bogdan Crivat, *Data Mining with Microsoft SQL Server 2008*, Wiley Publishing Inc., 2008
- [3] Ian H. Witten, Eibe Frank and Mark A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques - Third Edition*, Elsevier, 2011
- [4] Rafael Olivas, *Decision Trees - A Primer for Decision-making Professionals*, 2007
- [5] Scott A. Golder & Bernardo A. Huberman, *The Structure of Collaborative Tagging Systems*
- [6] Adam Freeman & Steven Sanderson, *Pro ASP.NET MVC 3 Framework – Third Edition*, Apress
- [7] Tomislav Piasevoli, *MDX with Microsoft SQL Server 2008 R2 Analysis Services: Cookbook*, Packt Publishing, 2011
- [8] Arttennick, *Practical MDX Queries for Microsoft SQL Server Analysis Services 2008*, McGraw-Hill Companies
- [9] <http://bis.net.vn/forums/p/378/661.aspx>
- [10] Nguyễn Thị Thùy Linh, *Thuật toán phân lớp cây quyết định*, Khóa luận tốt nghiệp đại học, Trường Đại học Công nghệ, 2005.
- [11] K. Cwalina, B. Abrams, *Framework Design Guidelines: Conventions, Idiom, and Patterns for Reusable .NET libraries Second Edition*, Addison Wesley, October 2008
- [12] Kruchten, Philippe (1995, November). *Architectural Blueprints — The “4+1” View Model of Software Architecture*
- [13] H.T.Kung, C.H.Wu *Content Networks: Taxonomy and New Approaches The Internet as a Large-Scale Complex System*, Kihong Park and Walter Willinger (Editors), published by Oxford University Press as part of Sante Fe Institute series, 2002.
- [14] D.Wollersheim, W.J.Rahayu *Using Medical Test Collection Relevance*

*Judgement to Identify Ontological Relationships Useful for Query Expansion* 21st International Conference on Data Engineering 2005.

[15] OU Shi-yan, KHOO Christopher S.G, GOH Dion H. Division of Information Studies, *Constructing a taxonomy to support multi-document summarization of dissertation abstracts*. Proceedings Issue of the 1st International Conference on Universal Digital Library (ICUDL 2005).

[16] E. Gamma, R. Helm, R. Johnson & J. Vlissides, *Design Patterns*, Addison-Wesley, 1998

[17] G. Booch, *Object-Oriented Analysis and Design with Applications*, The Benjamin Cummings Publishing Company, 1994

[18] I. Jacobso Grady Booch, J. Rumbaugh, *The unified software development process*, Addison-Wesley, 1999

[19] <http://blog.data-miners.com/2008/10/decision-trees-and-clustering.html>

[20] S. Acharya, P. Carlin, C. Galindo-Legaria, *Relational support for flexible schema scenerios*, One Microsoft Way

[21] P. Kruchten, *Architectural Blueprints – The “4+1” view model of software architecture*, Rational Software Corp., November 1995

[22] M.R. Barbacci, Software Quality Attributes and Architecture Tradeoffs, Carnegie Mellon University, 2003

[23] Nguyễn Thị Hương Thảo, Phân lớp phân cấp taxonomy văn bản web và ứng dụng, Khoa luận tốt nghiệp Đại học Công Nghệ, 2006.

[24] Len Bass, Paul Clements, Rick Kazman, *Software Architecture in Practice*, Second Edition. Addison Wesley