

Học viện Công nghệ Bưu Chính Viễn thông- Khoa Công nghệ thông tin I

KHO DỮ LIỆU VÀ KỸ THUẬT KHAI PHÁ

BÀI GIẢNG

DÀNH CHO SINH VIÊN CÔNG NGHỆ
THÔNG TIN

NGUYỄN QUỲNH CHI

GIỚI THIỆU

Học phần Kho dữ liệu và kỹ thuật khai phá cung cấp phương pháp luận và lý thuyết cơ sở dữ liệu về việc xây dựng một kho dữ liệu và ứng dụng vào xử lý phân tích trực tuyến, đồng thời cung cấp các kiến thức cơ bản về các phương pháp tích hợp cơ sở dữ liệu và các phương pháp khai phá dữ liệu để hỗ trợ cho hệ trợ giúp quyết định. Do đối tượng là sinh viên năm cuối của đại học nên chỉ trình bày những phương pháp khai phá cơ bản.

Đối tượng chính của bài giảng này là sinh viên ngành Công nghệ thông tin hệ đại học, ngoài ra sinh viên các hệ và chuyên ngành khác có thể dùng làm tài liệu tham khảo nếu cần. Để hiểu sâu thêm những kiến thức được trình bày trong bài giảng này, sinh viên cần đọc thêm sách được nêu ra trong phần tài liệu tham khảo. Sinh viên cần hoàn thành các môn học: Cơ sở dữ liệu, kỹ thuật lập trình, có khả năng làm việc với một hệ quản trị CSDL, nhập môn xác suất thống kê trước khi tham gia học môn học này.

Đây là một môn học tính điểm trung bình sau khi kết thúc cuối kỳ học, trong đó kiểm tra cuối kỳ chiếm 70%, kiểm tra giữa kỳ chiếm 20%, quá trình tham dự trên lớp chiếm 10%.

Tổng số gồm 3 tín chỉ trong đó 44 tiết lý thuyết giảng trên lớp, 8 tiết cho việc giảng viên giải đáp thắc mắc về bài tập và 2 tiết ôn tập trước khi thi cuối kỳ.

Yêu cầu đọc sách để chuẩn bị bài và làm bài tập lớn theo hướng dẫn của giảng viên trước mỗi buổi tham gia lớp học. Nói chung sinh viên được khuyến khích đặt các câu hỏi và phát biểu ý kiến riêng với những vấn đề đặt ra trong quá trình nghe giảng trên lớp, tránh thái độ thụ động ngồi nghe.

Nội dung của môn học sẽ được trình bày trong mục lục của bài giảng.

Mục lục

CHƯƠNG I: Giới thiệu về kho dữ liệu và khai phá dữ liệu	7
1.1 Khai phá dữ liệu là gì.....	8
1.2 Các loại dữ liệu và kiểu mẫu dữ liệu được khai phá.....	8
1.3 Các bài toán và phương pháp cơ bản trong khai phá dữ liệu.....	10
Định nghĩa bài toán phân loại	10
Định nghĩa bài toán phân cụm	11
Định nghĩa bài toán phát hiện luật kết hợp	12
Bài toán phân loại cho dữ liệu hồi quy	12
Phát hiện sự sai lệch hay dị thường	13
Khai phá dữ liệu và Nguyên lý quy nạp	13
1.4 Sự tích hợp của khai phá dữ liệu với cơ sở dữ liệu hay kho dữ liệu.....	14
Vai trò của khai phá dữ liệu đối với quá trình phát hiện tri thức từ dữ liệu	14
Các bước của quá trình phát hiện tri thức từ dữ liệu	14
Các chuyên ngành khác liên quan tới khai phá dữ liệu	16
So sánh khai phá dữ liệu với phân tích thống kê	16
So sánh khai phá dữ liệu với cơ sở dữ liệu	17
So sánh khai phá dữ liệu với công nghệ kho dữ liệu	17
Kiến trúc của một mô hình phân tích trực tuyến (OLAM)	17
So sánh Cơ sở dữ liệu, xử lý phân tích trực tuyến và khai phá dữ liệu	18
1.5 Ứng dụng của kho dữ liệu và khai phá dữ liệu.....	21
Ứng dụng của bài toán phân lớp (phân loại)	21
Ứng dụng của bài toán phân cụm	22
Ứng dụng của bài toán phát hiện luật kết hợp	22
Những vấn đề chính trong lĩnh vực công nghệ kho dữ liệu và khai phá dữ liệu	23

Câu hỏi ôn tập chương 1.....	24
Chương 2: Các công nghệ và kỹ thuật tích hợp cơ sở dữ liệu	26
2.1 Giới thiệu Mô hình dữ liệu mở rộng XML.....	26
Giới thiệu về ngôn ngữ XML (Extensible Markup Language)	26
Một hệ thống XML điển hình	27
Cú pháp của XML	28
Khai báo kiểu văn bản – Data Type Declaration (DTD)	31
Nhắc lại kiến thức về mô hình thực thể liên kết mở rộng	39
Kiến trúc tích hợp nhiều cơ sở dữ liệu	46
Kỹ thuật chuyển đổi lược đồ quan hệ sang mô hình thực thể liên kết mở rộng	46
Ví dụ về việc chuyển đổi từ lược đồ quan hệ sang mô hình thực thể liên kết	49
2.3 Tích hợp các lược đồ dữ liệu.....	53
Khái niệm về tích hợp dữ liệu	53
Các bước tích hợp ngữ nghĩa dữ liệu	54
Bài thực hành	65
2.4 Chuyển đổi và tích hợp dữ liệu.....	67
Phương pháp luận cho công nghệ kho dữ liệu và OLAP	67
Các cách chuyển đổi dữ liệu	67
Một ví dụ về việc chuyển đổi	71
Tích hợp dữ liệu	75
Câu hỏi ôn tập chương 2.....	81
Chương 3: Công nghệ kho dữ liệu và xử lý phân tích trực tuyến	83
3.1 Khái niệm về kho dữ liệu	83
3.2 Mô hình dữ liệu đa chiều	86
3.3 Kiến trúc của kho dữ liệu	95

3.4 Cài đặt kho dữ liệu	97
3.5 Liên hệ công nghệ kho dữ liệu với khai phá dữ liệu	104
3.6 Xây dựng kho dữ liệu với mục đích hỗ trợ quyết định (DSS).....	106
Nhắc lại một chút về khái niệm kho dữ liệu và những tác nhân liên quan	106
Các giai đoạn xây dựng	106
Thiết kế cơ sở dữ liệu với lược đồ hình sao	109
Nghiên cứu xây dựng một kho dữ liệu	110
Câu hỏi ôn tập chương 3.....	114
Chương 4: Khai phá dữ liệu	116
4.1 Tiền xử lý dữ liệu trước khi khai phá.....	116
Khái niệm về dữ liệu	116
Tiền xử lý dữ liệu	124
4.2 Phương pháp khai phá bằng luật kết hợp.....	129
Nguồn gốc của khai phá luật kết hợp	129
Các ứng dụng của luật kết hợp	129
Khái niệm cơ bản trong bài toán tìm luật kết hợp	130
Cách tiếp cận theo kiểu vét cạn (Brute-force approach)	130
Khai phá luật kết hợp với cách tiếp cận hai bước	132
Phương thức giảm số lượng các ứng cử viên: thuật toán Apriori	133
Một phương pháp sinh tập các mặt hàng thường xuyên FP-growth	139
Sinh luật kết hợp	143
4.3 Phương pháp cây quyết định.....	145
Những khái niệm cơ bản trong bài toán phân loại	145
Phương pháp phân loại bằng cây quyết định	146
Các thuật toán tìm cây quyết định	149

Đánh giá các mô hình phân loại	160
4.4 Phương pháp phân nhóm và phân đoạn.....	164
Khái niệm về phân tích phân cụm	164
Độ đo trong phân cụm	166
Phân loại phân cụm	170
Phương pháp phân cụm	173
Câu hỏi ôn tập chương 4	178
Tài liệu tham khảo	188

CHƯƠNG I: Giới thiệu về kho dữ liệu và khai phá dữ liệu

Vấn đề bùng nổ về dữ liệu: khi các công cụ thu thập dữ liệu tự động và công nghệ về cơ sở dữ liệu đã trở nên hoàn thiện, một lượng lớn dữ liệu được thu thập và lưu trữ trong những cơ sở dữ liệu, kho dữ liệu và các kho lưu trữ thông tin khác.

Lúc này, chúng ta đang có quá nhiều dữ liệu, chưa mang tính phục vụ có mục đích cho người sử dụng. Chúng ta đang thiếu tri thức, dữ liệu đã qua xử lý và phục vụ riêng cho mục đích của người sử dụng. Vấn đề là làm thế nào để khai thác tri thức từ đồng dữ liệu khổng lồ hiện đang có trong tay.

Giải pháp cho việc khai phá ra tri thức chính là sự ra đời của công nghệ kho dữ liệu và các phương pháp khai phá dữ liệu. Giải pháp này liên quan tới những khía cạnh sau đây:

- Công nghệ để xây dựng một kho dữ liệu lớn và các phương thức để xử lý phân tích trực tuyến (sẽ nghiên cứu trong những bài học sau)
- Trích lọc ra tri thức có ích cho con người bao gồm các luật, thể chế, mẫu, và các ràng buộc từ khối lượng lớn dữ liệu của một hay nhiều cơ sở dữ liệu có kích cỡ lớn.

Các lý do cần khai phá dữ liệu trên quan điểm thương mại trong thế giới thực.

- Rất nhiều dữ liệu đã được thu thập trong thế giới thực và được lưu trữ một cách hệ thống trong các kho dữ liệu bao gồm:
 - o Các dữ liệu trên web, các dữ liệu thương mại điện tử
 - o Các dữ liệu mua bán tại các cửa hàng, gian hàng trong siêu thị
 - o Các dữ liệu của giao dịch ngân hàng, thẻ tín dụng
- Máy tính trở nên rẻ hơn và có sức mạnh xử lý dữ liệu hơn
- Sức ép cạnh tranh mạnh mẽ hơn: cần cung cấp các dịch vụ tốt hơn và tùy biến với khách hàng hơn (nhất là trong quan hệ với khách hàng)

Các lý do cần khai phá dữ liệu trên quan điểm khoa học

- Các dữ liệu được thu thập và lưu trữ với tốc độ rất nhanh (GB/h) thông qua
 - o Bộ cảm biến (sensor) điều khiển từ xa trên các trạm vệ tinh
 - o Kính viễn vọng quan sát bầu trời
 - o Dùng công cụ microarray để sinh ra dữ liệu thể hiện đặc tính của gene (gene expression data)
 - o Dùng các bộ mô phỏng khoa học để tạo ra hàng terabyte dữ liệu
- Các kỹ thuật truyền thống không còn khả thi cho lượng lớn các dữ liệu thô

- Các kỹ thuật khai phá dữ liệu có thể sẽ giúp ích được các nhà khoa học hơn trong các công việc
 - o Phân loại và phân mảnh dữ liệu
 - o Hình thành các giả thuyết trong nghiên cứu khoa học

1.1 Khai phá dữ liệu là gì

Khai phá dữ liệu (phát hiện tri thức trong cơ sở dữ liệu sẵn có) là việc trích lọc ra những thông tin có ích (không hiển nhiên, không tường minh, không biết trước, và có ích một cách tiềm năng), những mẫu dữ liệu trong các cơ sở dữ liệu lớn.

Khai phá dữ liệu có một số tên gọi khác khi được sử dụng khi được đề cập đến trong cuộc sống cũng như trong sách và tạp chí khoa học như:

- Khám phá tri thức (knowledge discovery) trong cơ sở dữ liệu (thường được viết tắt theo tiếng anh là KDD).
- Trích lọc tri thức
- Phân tích mẫu/dữ liệu
- Khảo cổ dữ liệu
- Tri thức kinh doanh (business intelligence) và còn nhiều tên khác nữa ít dùng.

Xem xét một ví dụ sau để phân biệt khái niệm khai phá dữ liệu với các khái niệm trong cơ sở dữ liệu, cái mà dễ nhầm tưởng là khai phá dữ liệu

Những xử lý không phải là khai phá dữ liệu	Những xử lý là khai phá dữ liệu
Tra cứu số điện thoại trong danh bạ điện thoại	Xác định những tên được cho là phổ biến ở một địa danh cụ thể nào đó
Truy vấn một mô tơ tìm kiếm thông tin trên Web liên quan tới từ “Amazon”	Gộp nhóm các tài liệu giống nhau được trả về bởi công cụ tìm kiếm thông tin dựa vào ngữ cảnh của chúng (ví dụ như vùng Amazon, hay vùng miền Amazon.com)

1.2 Các loại dữ liệu và kiểu mẫu dữ liệu được khai phá

Khi thực hiện một công việc khai phá dữ liệu, để đưa ra các quyết định cần thiết cho công việc khai phá, chúng ta cần xác định những yếu tố sau:

- Loại cơ sở dữ liệu cần khai phá

Các loại cơ sở dữ liệu có thể dùng cho khai phá bao gồm cơ sở dữ liệu quan hệ, cơ sở dữ liệu giao dịch, hướng đối tượng, cơ sở dữ liệu quan hệ- đối tượng, không gian, cơ sở dữ liệu văn

bản, chuỗi thời gian, đa phương tiện, cơ sở dữ liệu hỗn tạp, cơ sở dữ liệu luật, cơ sở dữ liệu Web, và các loại cơ sở dữ liệu khác nữa.

- **Loại tri thức cần phát hiện ra**

Bao gồm tri thức miêu tả đặc điểm của các cá thể trong tập cá thể đang xét, phân biệt cá thể này với cá thể khác, luật kết hợp, tìm xu hướng, phân loại cá thể trong một tập hợp, phân cụm gộp nhóm các cá thể giống nhau, phân tích tìm ra cá thể ngoại lai và sự khác biệt đối với phần đông các cá thể khác, v.v...

Ngoài ra, tri thức còn là các chức năng tích hợp, đa chức năng và khai phá ở nhiều mức độ khác nhau.

- **Loại kỹ thuật cần được sử dụng để giải quyết vấn đề**

Bao gồm kỹ thuật theo hướng cơ sở dữ liệu, kỹ thuật kho dữ liệu (xử lý phân tích trực tuyến), các phương pháp học máy, các phương pháp thống kê, biểu diễn trực quan, mạng nơron nhân tạo, và các phương pháp khác.

- **Loại ứng dụng cần được xây dựng, áp dụng cho vấn đề khai phá**

Bao gồm các ứng dụng trong lĩnh vực bán lẻ, truyền thông, ngân hàng, phân tích lỗi, khai phá dữ liệu gen, phân tích thị trường chứng khoán, khai phá dữ liệu Web, phân tích Weblog.

Một công việc nữa cần được xác định là nhận thức rõ nhiệm vụ của bài toán khai phá dữ liệu là thuộc loại nào trong hai loại sau đây:

- **Bài toán khai phá dữ liệu dạng mô tả**

Nhiệm vụ của bài toán dạng này là tìm ra các mẫu mô tả dữ liệu mà con người có thể hiểu được.

- **Bài toán khai phá dữ liệu dạng tiên đoán**

Sử dụng một vài biến để tiên đoán các giá trị chưa biết hoặc trong tương lai của các biến khác.

Các nhiệm vụ thường gặp của việc khai phá dữ liệu

- Phân loại: thuộc loại bài toán tiên đoán
- Phân cụm: thuộc loại bài toán mô tả
- Phát hiện luật kết hợp: thuộc loại bài toán mô tả
- Phát hiện mẫu dạng liên tục: thuộc loại bài toán mô tả
- Bài toán hồi quy: thuộc loại bài toán tiên đoán
- Phát hiện sự khác biệt: thuộc loại bài toán tiên đoán

1.3 Các bài toán và phương pháp cơ bản trong khai phá dữ liệu

Định nghĩa bài toán phân loại

- Cho một tập các bản ghi được gọi là tập huấn luyện, mỗi bản ghi chứa một tập các thuộc tính, một thuộc tính trong đó gắn nhãn phân loại được gọi là thuộc tính lớp.
- Nhiệm vụ của bài toán phân loại là tìm ra một mô hình thể hiện thuộc tính lớp là một hàm của giá trị của các thuộc tính khác
- Sau khi tìm được mô hình thích hợp nhất cho bài toán, mục đích cuối cùng là áp dụng mô hình (hàm tìm được) đó để tiên đoán các bản ghi chưa được biết đến trước đó thuộc lớp nào một cách càng chính xác càng tốt.
- Một tập bản ghi kiểm thử được dùng để xác định độ chính xác của mô hình. Thông thường, một tập dữ liệu được đưa ra sẽ được chia thành tập huấn luyện và tập kiểm thử, tập huấn luyện được dùng để xây dựng mô hình và tập kiểm thử được dùng để kiểm tra.

Một ví dụ minh họa cho bài toán phân loại: Cho tập các bản ghi được coi là tập huấn luyện như hình vẽ dưới đây

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Trong đó thuộc tính **Cheat** là thuộc tính phân lớp, thuộc tính Tid không có ý nghĩa trong việc huấn luyện mô hình. Các bản ghi của tập huấn luyện này được sử dụng để tìm ra sự phụ thuộc giữa thuộc tính phân lớp và các thuộc tính còn lại (hàm phụ thuộc). Khi tìm được sự phụ thuộc này (hay còn gọi là bộ phân lớp) chúng ta nói đã huấn luyện xong mô hình phân lớp.

Mô hình phân lớp tìm được sẽ được xác định chính xác thông qua việc áp dụng mô hình phân lớp cho một bộ dữ liệu

Refund	Marital Status	Taxable Income	Cheat
No	Single	75K	?
Yes	Married	50K	?
No	Married	150K	?
Yes	Divorced	90K	?
No	Single	40K	?
No	Married	80K	?

Giá trị của thuộc tính ***Cheat*** sẽ được tính sau khi đưa mỗi bản ghi qua mô hình phân lớp, giá trị đó sẽ được so sánh với giá trị thực của thuộc tính trong bộ dữ liệu được cho trước, để xác định tính chính xác của mô hình phân lớp.

Mô hình tìm được sẽ được sử dụng để phân loại các bản ghi mới với những giá trị thuộc tính (ngoại trừ thuộc tính phân lớp) đã biết, để phục vụ nhu cầu của người sử dụng. Ví dụ minh họa này, với những giá trị sẵn có của một người như tình trạng hôn nhân, thu nhập tính thuế và thông tin có hoàn trả thuế hay không, mô hình phân loại bản ghi đó là thông tin giả hay thật.

Định nghĩa bài toán phân cụm

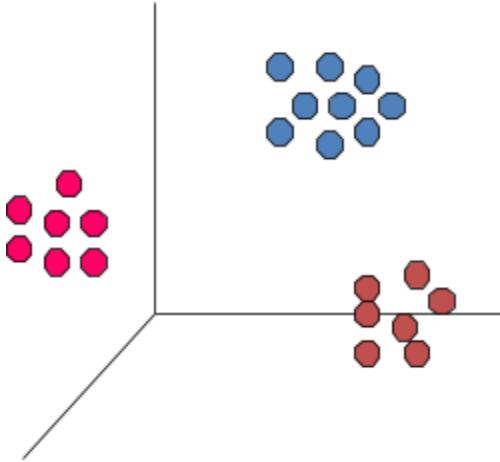
Cho một tập các điểm dữ liệu, mỗi điểm có một tập thuộc tính và có một độ đo sự tương đồng giữa chúng để phân cụm sao cho:

- Những điểm dữ liệu trong cùng một cụm thì có sự tương đồng cao, nhiều hơn với các điểm khác.
- Những điểm dữ liệu trong các cụm riêng rẽ thì ít tương đồng hơn các điểm thuộc cùng một cụm.

Các độ đo sự tương đồng có thể kể đến

- Khoảng cách Euclidean nếu các thuộc tính là giá trị liên tục
- Các độ đo khác theo từng bài toán và lĩnh vực

Mô tả một phân cụm dựa trên khoảng cách Euclidean trong không gian 3 chiều được thể hiện trong hình vẽ dưới đây



Nhìn và hình vẽ thấy rõ các điểm được phân thành 3 cụm thể hiện bởi ba màu đỏ, nâu và xanh sao cho khoảng cách giữa hai điểm bất kỳ trong cùng một cụm là nhỏ nhất có thể và khoảng cách giữa hai điểm bất kỳ của hai cụm khác nhau là lớn nhất có thể.

Định nghĩa bài toán phát hiện luật kết hợp

Cho một tập các bản ghi, mỗi bản ghi đều có chứa một số mặt hàng nằm trong một tập các mặt hàng cho sẵn. Nhiệm vụ của bài toán này là sản xuất ra các luật phụ thuộc, thể hiện sự tiên đoán về sự xuất hiện một mặt hàng này dựa trên sự xuất hiện của các mặt hàng khác.

Bài toán này xuất phát từ nhu cầu thực tế khi con người đi mua bán ở các siêu thị. Một ví dụ mô tả bài toán này như sau: Cho thông tin về các giao dịch mua bán được thể hiện trong bảng dưới đây gồm 2 cột: mã giao dịch và các mặt hàng mua bán trong mỗi giao dịch. Các luật tìm được: {Milk} \rightarrow {Coke}; {Diaper, Milk} \rightarrow {Beer} có nghĩa là nếu một người mua sữa (Milk) thì nhiều khả năng sẽ mua Coca cola (Coke); Và nếu mua tã và sữa (Diaper, Milk) thì nhiều khả năng sẽ mua bia (Beer).

<i>TID</i>	<i>Items</i>
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Bài toán phân loại cho dữ liệu hồi quy

Định nghĩa bài toán

Dự đoán một giá trị của một biến hồi quy dựa trên giá trị của các biến khác với giả định mô hình phụ thuộc là tuyến tính hoặc phi tuyến.

Bài toán này được sử dụng rất nhiều trong nghiên cứu thông kê, và các lĩnh vực của mạng nơron.

Ví dụ của bài toán

- Dự đoán số lượng bán ra của các sản phẩm mới dựa trên chi phí cho việc quảng cáo
- Dự đoán vận tốc của gió như là một hàm số của nhiệt độ, độ ẩm, áp suất...vv
- Tiên đoán theo chuỗi thời gian của chỉ số thị trường chứng khoán

Phát hiện sự sai lệch hay dị thường

Định nghĩa bài toán: Phát hiện những sai phạm đáng kể từ những hành vi bất thường

Ví dụ của bài toán

- Phát hiện xâm phạm thẻ tín dụng: dùng thẻ tín dụng của người khác để mua bán trên mạng
- Phát hiện xâm nhập mạng lưới máy tính để thực hiện các hoạt động không bình thường

Khai phá dữ liệu và Nguyên lý quy nạp

Trong phần này ta xem xét sự liên hệ giữa khai phá dữ liệu và nguyên lý quy nạp và suy diễn. Trước hết ta phân biệt suy diễn và quy nạp.

Suy diễn thông thường đảm bảo tính xác thực của mệnh đề. Một ví dụ cho sự suy diễn này được thể hiện thông qua ba mệnh đề sau:

1. Tất cả các con ngựa đều là loài động vật có vú
2. Tất cả các loài động vật có vú đều có phổi
3. Vì thế, tất cả các loài ngựa đều có phổi

Trong khi đó, suy diễn quy nạp thêm thông tin (chưa chắc đã xác thực). Một ví dụ về suy diễn quy nạp như sau:

1. Tất cả các con ngựa được quan sát từ trước đến nay đều có phổi
2. Vì vậy, tất cả các con ngựa đều có phổi.

Suy diễn theo kiểu quy nạp thường gặp vấn đề: từ các thực tế có thực, chúng ta có thể suy diễn ra một mô hình sai hoặc không đúng trong tất cả các trường hợp. Một ví dụ điển hình cho vấn đề này được thể hiện qua các mệnh đề sau: Tất cả các con thiên nga ở châu Âu đều màu trắng

Dùng suy diễn theo kiểu quy nạp suy ra rằng: tất cả các con thiên nga đều màu trắng như một quy luật chung. Nhưng chúng ta thấy rằng còn loại thiên nga ở châu Úc và loại thiên nga đen nữa. Như vậy kết quả của suy diễn quy nạp là sai trong một số trường hợp. Nguyên nhân việc

suy diễn sai ở đây là do việc chọn tập các mẫu quan sát không ngẫu nhiên và không đại diện cho tập toàn bộ cá thể.

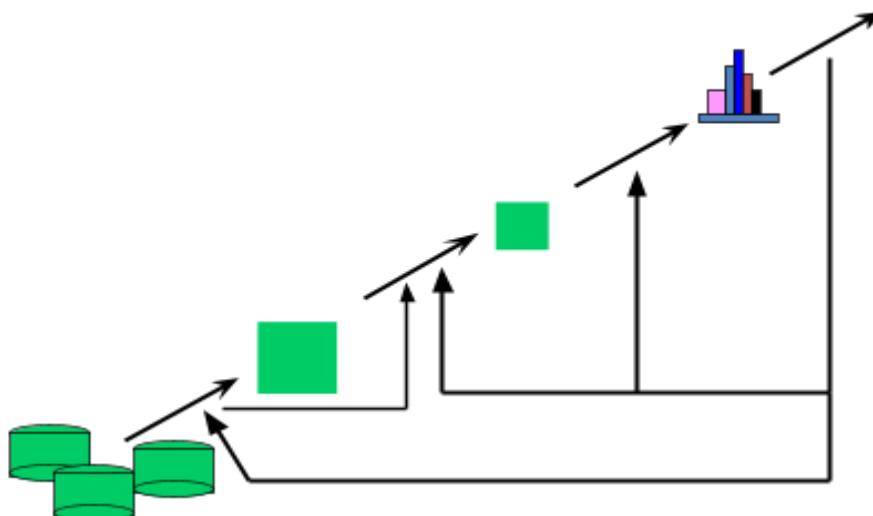
Một ví dụ khác: phân biệt các thùng chứa của Mỹ và của Irắc.

- Để thực hiện việc này chúng ta dùng phương pháp phân loại sử dụng một cơ sở dữ liệu các hình ảnh, và phân chung ra thành tập huấn luyện và tập kiểm thử, mô hình phân loại sẽ được xây dựng dựa trên tập huấn luyện.
- Kết quả của phương pháp này sẽ cho độ chính xác của việc tiên đoán tốt chỉ trên tập kiểm thử, còn sẽ cho kết quả tồi trên các bức ảnh độc lập khác.
- Nguyên nhân của việc cho độ chính xác tồi khi phân loại các hình ảnh độc lập là do các đặc điểm đặc biệt trên các bức ảnh đó.

1.4 Sự tích hợp của khai phá dữ liệu với cơ sở dữ liệu hay kho dữ liệu

Vai trò của khai phá dữ liệu đối với quá trình phát hiện tri thức từ dữ liệu (KDD)

được thể hiện trong hình vẽ dưới đây

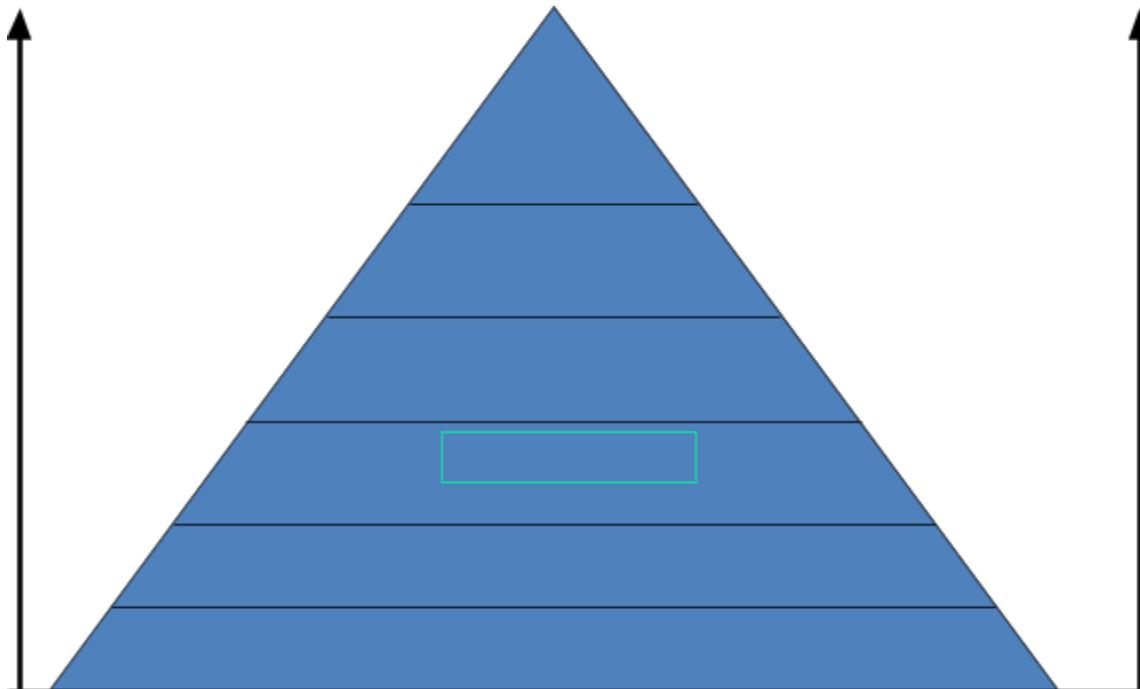


Các bước của quá trình phát hiện tri thức từ dữ liệu

- Học từ lĩnh vực ứng dụng: liên quan tới các tri thức liên quan trước đó và mục tiêu của ứng dụng
- Tạo một tập dữ liệu đích: cần phải lựa chọn dữ liệu cho vào tập dữ liệu này
- Quá trình tiền xử lý và làm sạch dữ liệu: có lẽ chiếm 60% công sức trong toàn bộ
- Chuyển đổi và thu hẹp dữ liệu: quá trình này liên quan tới việc tìm ra những đặc tính có ích, giảm biến và chiều của dữ liệu, tìm ra những phần tử đại diện bất biến

- Lựa chọn những chức năng của khai phá dữ liệu như tổng hợp, phân loại, phân loại cho dữ liệu liên tục, luật kết hợp, phân cụm
- Lựa chọn các thuật toán khai phá
- Khai phá dữ liệu: cần tìm kiếm các mẫu quan tâm
- Đánh giá các mẫu tìm được và biểu diễn tri thức thông qua các phương pháp trực quan, phương pháp chuyển đổi, loại bỏ các mẫu dư thừa, v.v..
- Sử dụng các tri thức phát hiện được cho mục đích khác của người sử dụng

Mối quan hệ giữa Khai phá dữ liệu và Tri thức kinh doanh được thể hiện trong tháp dưới đây



Trục bên trái của tháp thể hiện mức độ hỗ trợ cho việc ra quyết định của các nhà kinh doanh tăng dần của các công việc trong tháp tương ứng với mức đó. Trục bên phải của tháp thể hiện các vai trò của con người thực hiện công việc ở mức tương ứng của tháp.

Dữ liệu được xử lý ở các mức độ khác nhau từ thấp đến cao tính từ đáy đến đỉnh của tháp. Ở mức thấp nhất, nguồn dữ liệu được thu thập từ nhiều kênh khác nhau như từ các tài liệu, tập tin, nhà cung cấp thông tin, các hệ thống cơ sở dữ liệu, hệ thống xử lý giao dịch trực tuyến (OLTP). Sau đó, các dữ liệu được đưa vào kho dữ liệu hoặc các kho dữ liệu theo chiều để cung cấp xử lý phân tích trực tuyến (OLAP), với quản trị dữ liệu đa chiều (MDA). Hai mức này được thực hiện bởi người quản trị hệ thống cơ sở dữ liệu. Tiếp tới các dữ liệu được thăm dò bằng các phương pháp phân tích thống kê, báo cáo và truy vấn và được khai phá để phát hiện ra thông tin bởi các

này phân tích dữ liệu. Cuối cùng, dữ liệu sau khi được khai phá sẽ được trình bày sử dụng các kỹ thuật biểu diễn trực quan, kết quả của việc biểu diễn trực quan này sẽ được các người sử dụng cuối sử dụng trợ giúp cho việc ra quyết định.

Các loại dữ liệu cho khai phá dữ liệu có thể kể đến các loại sau

- Cơ sở dữ liệu quan hệ: đã được học trong học phần Cơ sở dữ liệu
- Kho dữ liệu
- Các cơ sở dữ liệu giao dịch
- Các cơ sở dữ liệu nâng cao và các kho chứa thông tin bao gồm:
 - o các cơ sở dữ liệu hướng đối tượng và cơ sở dữ liệu đối tượng quan hệ,
 - o cơ sở dữ liệu không gian,
 - o dữ liệu thời gian và chuỗi thời gian
 - o Cơ sở dữ liệu văn bản và đa phương tiện
 - o Các cơ sở dữ liệu thông tin bằng chữ và hỗn tạp
 - o Hệ thống trang Web trên toàn cầu

Các chuyên ngành khác liên quan tới khai phá dữ liệu

- Các công nghệ cơ sở dữ liệu
- Các kỹ thuật học máy
- Thống kê
- Khoa học thông tin
- Biểu diễn trực quan và các chuyên ngành khác.

So sánh khai phá dữ liệu với phân tích thống kê

Phân tích thống kê	Khai phá dữ liệu
phù hợp với các loại dữ liệu có cấu trúc và dạng số	Phù hợp với tập dữ liệu lớn, dữ liệu của thế giới thực, có thể có nhiều giá trị bị mất, dữ liệu tồn tại trước đó không phải do người sử dụng tạo ra
Hoàn toàn hướng dữ liệu – không liên quan tới tri thức miền giá trị cả dữ liệu	Hiệu quả và khả năng mở rộng về kích cỡ của thuật toán là quan trọng đối với việc khai phá
Phiên dịch kết quả khó và không rõ ràng	Dữ liệu không tinh- có xu hướng cập nhật thường xuyên

Cần sự hướng dẫn của chuyên gia sử dụng	Cần các phương pháp thu thập dữ liệu hiệu quả có sẵn để dùng
---	--

So sánh khai phá dữ liệu với cơ sở dữ liệu

Để so sánh chúng ta xem xét báo cáo cơ sở dữ liệu thường trả lời những truy vấn chứa các thông tin kiểu như sau:

- Lượng hàng bán được cho mỗi loại dịch vụ của các tháng trước đó
- Lượng hàng bán được cho mỗi loại dịch vụ được gộp nhóm theo từng giới tính của khách hàng hoặc nhóm tuổi của khách hàng
- Liệt kê danh sách các khách hàng không dùng dịch vụ liên tục của công ty

Những câu hỏi trả lời được bởi khai phá dữ liệu kiểu như sau:

- Đặc điểm chung của các khách hàng không dùng liên tục dịch vụ của công ty và sự khác nhau giữa họ và các khách hàng có dùng dịch vụ liên tục
- Loại người dùng bảo hiểm mô tô nào là khách hàng tiềm năng cho loại bảo hiểm đồ đạc trong nhà.

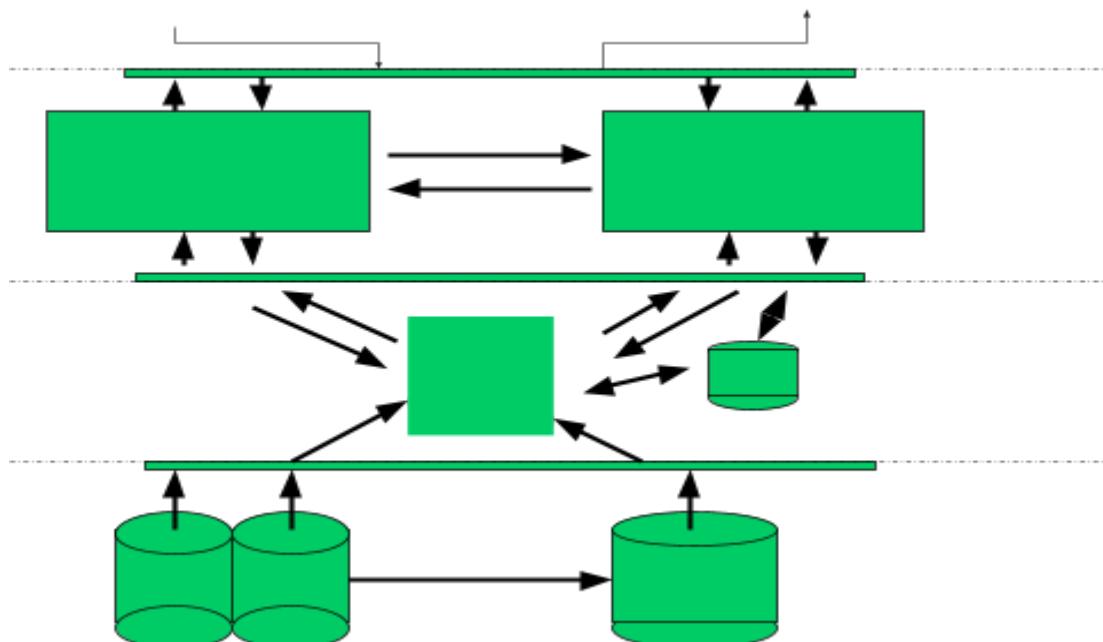
So sánh khai phá dữ liệu với công nghệ kho dữ liệu

- Kho dữ liệu là một kho lưu trữ dữ liệu tập trung có thể được truy vấn cho các lợi ích kinh doanh
- Công nghệ kho dữ liệu có thể
 - Trích lọc các dữ liệu tác nghiệp được lưu trữ
 - Giải quyết được sự không đồng nhất giữa các định dạng dữ liệu văn bản khác nhau
 - Tích hợp dữ liệu trong toàn bộ doanh nghiệp, không phụ thuộc vào vị trí, định dạng hoặc các yêu cầu về truyền thông giao tiếp
 - Phối hợp với các thông tin của chuyên gia và thông tin bổ sung từ bên ngoài
- Xử lý phân tích trực tuyến là chức năng do công nghệ kho dữ liệu cung cấp
- Mô hình dữ liệu nhiều chiều cũng thuộc công nghệ kho dữ liệu
- Các thao tác cơ bản của công nghệ kho dữ liệu bao gồm:
 - Cuộn lên (roll-up)
 - Khoan sâu xuống (drill-down)
 - Cắt dọc (Slice) và cắt ngang (dice)
 - Quay (Rotate)

Kiến trúc của một mô hình phân tích trực tuyến (OLAM)

được thể hiện như hình vẽ dưới đây

Mô hình OLAM bao gồm 4 tầng như hình vẽ trên: Kho lưu trữ dữ liệu, CSDL đa chiều, OALP/OLAM và giao diện với người sử dụng. Giữa mỗi tầng có một giao diện xử lý (API): tầng 1 và 2 là API của cơ sở dữ liệu (Database API), giữa tầng 2 và 3 là API của khối dữ liệu (Data Cube API), giữa tầng 3 và 4 là API giao diện đồ họa với người sử dụng (User GUI API). Dữ liệu của mỗi tầng được lưu trữ dưới dạng CSDL và kho dữ liệu ở tầng 1, CSDL đa chiều ở tầng 2 và dạng của OLAP và OLAM ở tầng 3, tầng 4 là tầng cho người sử dụng (NSD). Ở tầng 4 NSD đưa vào hệ thống những câu truy vấn khai phá và thông qua các mô hình OLAP và OLAM nhận được kết quả khai phá thông qua giao diện đồ họa. Các mũi tên giữa các khối trong hình vẽ thể hiện sự tương tác một chiều (ứng với mũi tên một chiều) hay tương tác qua lại (ứng với mũi tên hai chiều) của các bộ phận trong hệ thống với công việc chính là các nhãn gắn trên mũi tên đó. Ngoài dữ liệu ra, tầng 2 còn có sự góp phần của siêu dữ liệu giúp bổ sung thông tin cho các dữ liệu chính trong hệ thống.



So sánh Cơ sở dữ liệu, xử lý phân tích trực tuyến và khai phá dữ liệu

được thể hiện theo các tiêu chí so sánh bao gồm

- Nhiệm vụ:

- Trích xuất dữ liệu chi tiết và tổng quát của cơ sở dữ liệu (DBMS)
- Tóm tắt, xác định xu hướng và dự đoán của hệ thống xử lý phân tích trực tuyến (OLAP)

- Khai phá dữ liệu từ những thông tin tiềm ẩn bên trong dữ liệu của khai phá dữ liệu (DM)
- Loại kết quả:
 - Thông tin của DBMS
 - Phân tích của OLAP
 - Chi tiết bên trong và dự đoán của DM
- Phương pháp:
 - Suy diễn bằng các câu hỏi và kiểm định với dữ liệu của DBMS
 - Mô hình dữ liệu đa chiều, tích hợp và thống kê của OLAP
 - Quy nạp bằng cách xây dựng mô hình, áp dụng nó với dữ liệu mới và thu thập kết quả cho DM
- Các câu hỏi ví dụ:
 - DBMS có thể trả lời: Ai mua quỹ phúc lợi trong vòng 3 năm gần đây?
 - OLAP có thể trả lời: Thu nhập trung bình của những người mua quỹ phúc lợi theo từng vùng cho từng năm?
 - DM có thể trả lời: Ai sẽ mua quỹ phúc lợi trong 6 tháng tới và tại sao.
- Ví dụ về dữ liệu thời tiết trong cơ sở dữ liệu được cho trong bảng sau

Da y	outlook	temperature	humidity	windy	play
1	sunny	85	85	false	no
2	sunny	80	90	true	no
3	overcast	83	86	false	yes
4	rainy	70	96	false	yes
5	rainy	68	80	false	yes
6	rainy	65	70	true	no
7	overcast	64	65	true	yes
8	sunny	72	95	false	no

9	sunny	69	70	false	yes
10	rainy	75	80	false	yes
11	sunny	75	70	true	yes
12	overcast	72	90	true	yes
13	overcast	81	75	false	yes
14	rainy	71	91	true	no

- Với DBMS khi truy vấn trong DBMS chưa trong bảng trên ta có thể trả lời những câu hỏi như :
 - o Nhiệt độ của ngày Chủ nhật là bao nhiêu? {85, 80, 72, 69, 75}
 - o Những ngày nào có độ ẩm nhỏ hơn 75? {6, 7, 9, 11}
 - o Những ngày nào có nhiệt độ lớn hơn 70? {1, 2, 3, 8, 10, 11, 12, 13, 14}
 - o Những ngày nào có nhiệt độ lớn hơn 70 và độ ẩm lớn hơn 75? {11}
- Với OLAP ta có thể tạo ra mô hình dữ liệu đa chiều (**Multidimensional Model**) hay còn gọi là khối dữ liệu (**Data Cube**).
 - o VD có sử dụng các chiều : **time**, **outlook** và **play** ta có thể tạo ra được mô hình sau

9 / 5	sunny	rainy	overcast
Week 1	0 / 2	2 / 1	2 / 0
Week 2	2 / 1	1 / 1	2 / 0

- Với DM sử dụng phương pháp phân loại bằng cây quyết định ID3 dữ liệu sẽ được biểu diễn dưới dạng cây quyết định như sau
 - o outlook = sunny
 - humidity = high: no
 - humidity = normal: yes

- outlook = overcast: yes
- outlook = rainy
 - windy = true: no
 - windy = false: yes

1.5 Ứng dụng của kho dữ liệu và khai phá dữ liệu

Ứng dụng của bài toán phân lớp (phân loại)

1. Sử dụng trong tiếp thị trực tiếp:

- Mục đích: Phân loại khách hàng để xác định nhóm khách hàng tiềm năng thích mua những sản phẩm máy di động thế hệ mới nhất. Nhờ đó, các nhân viên tiếp thị không tốn tiền gửi thư cho những khách hàng không tiềm năng, chỉ gửi cho nhóm khách hàng tiềm năng này, để tiết kiệm chi phí.
- Cách tiếp cận cho ứng dụng này như sau
 - Sử dụng dữ liệu của một sản phẩm tương tự được giới thiệu trước đó
 - Ta biết được những khách hàng nào mua và những khách hàng nào không mua hàng. Quyết định *{buy, don't buy}* chỉ ra *thuộc tính lớp*.
 - Thu thập các thông tin về nhân khẩu học, phong cách sống, các thông tin liên quan tới việc giao tiếp với công ty của khách hàng
 - Công việc của khách hàng, nơi họ sống, số tiền họ kiếm được, v.v...
 - Sử dụng thông tin này như là các thuộc tính đầu vào để huấn luyện một mô hình phân lớp.

2. Sử dụng trong phát hiện lừa gạt

- Mục đích: Tiên đoán các trường hợp lừa gạt trong các giao dịch bằng thẻ tín dụng.
- Cách tiếp cận:

- Dùng các thông tin của giao dịch bằng thẻ và các thông tin về tài khoản của người dùng như các thuộc tính như khi nào khách hàng mua, anh ta mua cái gì, tần suất anh ta trả tiền đúng hạn v.v..
- Gán nhãn các giao dịch trong quá khứ như những giao dịch gian lận và không gian lận. Điều này xác định thuộc tính lớp.
- Huấn luyện một mô hình cho việc phân lớp của các giao dịch.
- Sử dụng mô hình này để phát hiện ra gian lận bằng cách quan sát những giao dịch bằng thẻ của một tài khoản.

3. Sử dụng trong việc kiểm tra xu hướng giảm số lượng khách hàng

- Mục đích: Tiên đoán xem liệu có để một khách hàng rời vào tay một công ty cạnh tranh hay không.
- Cách tiếp cận:

- Sử dụng các bản ghi chi tiết của các giao dịch của từng khách hàng trong hiện tại và quá khứ để tìm các thuộc tính như tần suất các cuộc gọi của khách hàng, khách hàng gọi ở đâu, thời điểm nào khách hàng hay gọi nhất, tình hình tài chính và tình trạng hôn nhân của khách hàng v.v...
- Gán nhãn cho khách hàng gồm khách hàng lâu năm và không lâu năm.
- Tìm ra mô hình để phân loại khách hàng lâu năm

4. Sử dụng trong phân loại các vật thể khi khảo sát bầu trời

- Mục đích: Tiên đoán phân loại các vật thể trên bầu trời (là sao hay thiên hà), dựa trên những hình ảnh thu được từ kính thiên văn. Ví dụ từ 3000 bức ảnh với 23,040 x 23,040 pixels/ảnh.
- Cách tiếp cận:

- Phân đoạn ảnh.
- Đo các thuộc tính ảnh, thường thì 40 thuộc tính cho mỗi đối tượng ảnh

- Thiết lập mô hình phân lớp dựa trên những thuộc tính này

Ứng dụng của bài toán phân cụm

1. Phân mảnh thị trường
 - Mục đích: chia nhỏ thị trường thành các tập con riêng biệt mà bất kỳ tập con nào cũng có thể được lựa chọn như là một mục tiêu tiếp thị.
 - Cách tiếp cận:
 - Thu thập các thuộc tính khác nhau của khách hàng dựa trên các thông tin liên quan đến lối sống, khu vực sinh sống.
 - Tìm các cụm khách hàng tương đồng.
 - Đánh giá chất lượng phân cụm bằng cách quan sát kiểu mua hàng của những khách hàng thuộc cùng một cụm với các khách hàng thuộc cụm khác.
2. Phân cụm tài liệu
 - Mục đích: Tìm ra những nhóm văn bản có sự tương đồng lẫn nhau dựa trên các thuật ngữ quan trọng xuất hiện trong văn bản.
 - Cách tiếp cận:
 - Xác định những thuật ngữ thường xuất hiện trong văn bản. Chỉ ra độ tương đồng dựa trên tần suất xuất hiện các khái niệm khác nhau. Dùng nó để phân cụm.
 - Kết quả đạt được: Trích lọc thông tin có thể dùng kết quả phân cụm này để liên hệ tới một văn bản mới hoặc tìm kiếm các từ thuật ngữ trong một văn bản đã được phân cụm.

Ứng dụng của bài toán phát hiện luật kết hợp

1. Tiếp thị và Tăng doanh số Bán Hàng

- Giả sử phát hiện ra luật sau : { *Bagels*, ... } --> { *Potato Chips* }
- Potato Chips được coi là hệ quả của việc mua *Bagels*, điều này có thể được dùng để xác định công việc cần thực hiện để tăng doanh số bán hàng lên.
- Bagels được gọi là điều kiện trước và nó có thể được dùng để xác định xem những sản phẩm nào trong siêu thị có thể bị ảnh hưởng nếu dùng bán *Bagels*.
- Bagels là điều kiện trước và Potato chips là hệ quả sau. Điều này xác định rằng các sản phẩm cần được bán cùng với *Bagels* để tăng doanh số bán hàng là *Potato chips*.

2. Quản lý các kệ hàng trong siêu thị

- Mục đích: Phát hiện ra các luật kết hợp để xác định các mặt hàng được mua cùng nhau bởi nhiều khách hàng. Nhờ đó có thể sắp xếp các kệ hàng,gian hàng trong siêu thị một cách hợp lý nhất.
- Cách tiếp cận: xử lý các dữ liệu trọng điểm được thu thập từ việc nhận dạng qua mã quét hàng lúc thanh toán để tìm mối quan hệ giữa các mặt hàng.
- Thường xuất hiện một luật: Nếu một khách hàng mua tã và sữa, thì nhiều khả năng anh ấy sẽ mua bia.

Dự đoán một giá trị của 1 biến liên tục dựa trên giá trị của các biến khác, Giả định 1 mô hình tuyến tính hay phi tuyến của các phụ thuộc

Những vấn đề chính trong lĩnh vực công nghệ kho dữ liệu và khai phá dữ liệu

Một trong những vấn đề cần giải quyết liên quan tới sự đa dạng về loại dữ liệu được dùng trong khai phá cũng như được tích hợp vào kho dữ liệu bao gồm

- Xử lý loại dữ liệu quan hệ và dữ liệu loại tổng hợp và phức tạp
- Khai phá các thông tin từ những cơ sở dữ liệu hỗn tạp và hệ thống lưu trữ thông tin trên oàn cầu như trên hệ thống trang web toàn cầu (www)

Thứ hai là các vấn đề liên quan tới ứng dụng và các ảnh hưởng về mặt xã hội bao gồm

- Các ứng dụng các tri thức khai phá được liên quan tới các công cụ khai phá dữ liệu cho các lĩnh vực cụ thể; Trả lời các câu hỏi thông minh; Kiểm soát xử lý và ra quyết định
- Tích hợp các tri thức phát hiện được với các tri thức đã tồn tại sẵn có. Đây chính là bài toán trộn tri thức.
- Bảo đảm an toàn dữ liệu, toàn vẹn và riêng tư của dữ liệu

Câu hỏi ôn tập chương 1

1.1 Khai phá dữ liệu là gì? Trong câu trả lời của bạn, hãy chỉ ra những điều sau:

- (a) Nó là một sự chuyển đổi đơn giản của công nghệ phát triển từ cơ sở dữ liệu, thống kê, và học máy?
- (b) Giải thích sự tiến triển của công nghệ cơ sở dữ liệu đã dẫn đến khai phá dữ liệu như thế nào.
- (c) Mô tả các bước liên quan đến khai phá dữ liệu khi được xem như một quá trình phát hiện tri thức

1.2 Trình bày một ví dụ trong đó khai phá dữ liệu rất quan trọng đối với sự thành công của một doanh nghiệp. Nếu các chức năng của khai phá dữ liệu mà doanh nghiệp này cần. Chúng có thể

được thực hiện thay thế bằng việc xử lý truy vấn dữ liệu hoặc phân tích thống kê đơn giản được không?

1.3 Giả sử nhiệm vụ của bạn là kỹ sư phần mềm tại Học viện công nghệ bưu chính viễn thông là thiết kế một hệ thống khai thác dữ liệu để nghiên cứu cơ sở dữ liệu các khóa học đại học, bao gồm các thông tin sau: tên, địa chỉ và trạng thái (ví dụ: đại học hoặc sau đại học) của từng sinh viên, các khóa học được thực hiện và điểm trung bình tích lũy (GPA). Hãy mô tả kiến trúc bạn sẽ chọn. Mục đích của mỗi thành phần của kiến trúc này là gì?

1.4 Kho dữ liệu khác với cơ sở dữ liệu như thế nào? Chúng giống nhau như thế nào?

1.5 Mô tả ngắn gọn các hệ thống và ứng dụng của cơ sở dữ liệu nâng cao sau: cơ sở dữ liệu quan hệ đối tượng, cơ sở dữ liệu không gian, cơ sở dữ liệu văn bản, cơ sở dữ liệu đa phương tiện, dữ liệu luồng, World Wide Web.

1.6 Định nghĩa từng chức năng của khai phá dữ liệu sau: phân tích đặc điểm, sự khác biệt, tính kết hợp và phân tích tương quan, phân loại, dự đoán, phân cụm và phân tích tiền hóa. Đưa ra ví dụ về từng chức năng khai phá dữ liệu, sử dụng cơ sở dữ liệu thực tế mà bạn đã quen thuộc.

1.7 Sự khác nhau giữa phân tích sự khác biệt và phân loại là gì? Giữa phân tích đặc tính và phân cụm? Giữa phân loại và dự đoán? Đối với mỗi cặp nhiệm vụ, chúng giống nhau như thế nào?

1.8 Dựa trên quan sát của bạn, mô tả một loại tri thức khác có thể cần phải được phát hiện bằng phương pháp khai phá dữ liệu nhưng chưa được liệt kê trong chương này. Nó có cần phương pháp khai phá hoàn toàn khác với phương pháp được nêu trong chương này không?

1.9 Liệt kê và mô tả năm yếu tố gốc (primitives) để xác định một nhiệm vụ khai phá dữ liệu.

1.10 Mô tả lý do tại sao phân cấp khái niệm hữu ích trong khai phá dữ liệu.

1.11 Các thành phần ngoại lai (outliers) thường bị loại bỏ là tiếng ồn. Tuy nhiên, rác của một người có thể là kho báu của người khác. Ví dụ: các trường hợp ngoại lệ trong giao dịch thẻ tín dụng có thể giúp chúng tôi phát hiện việc sử dụng thẻ tín dụng gian lận. Lấy phát hiện gian lận làm ví dụ, đề xuất hai phương pháp có thể được sử dụng để phát hiện các ngoại lệ và thảo luận phương pháp nào đáng tin cậy hơn.

1.12 Các ứng dụng gần đây đặc biệt chú ý đến luồng dữ liệu không gian và thời gian. Luồng dữ liệu không gian và thời gian chứa thông tin không gian thay đổi theo thời gian và có dạng dữ liệu luồng (tức là luồng dữ liệu vào và ra như các luồng vô hạn).

(a) Trình bày ba ví dụ ứng dụng của luồng dữ liệu không gian và thời gian.

(b) Thảo luận xem loại kiến thức có ích nào có thể được khai phá từ các luồng dữ liệu như vậy, với thời gian và tài nguyên giới hạn.

(c) Xác định và thảo luận những thách thức chính trong khai phá dữ liệu không gian thời gian.

(d) Sử dụng một ví dụ ứng dụng, phác họa một phương pháp để khai phá một loại tri thức từ dữ liệu luồng như vậy hiệu quả.

1.13 Mô tả sự khác biệt giữa các cách tiếp cận sau đây cho việc tích hợp dữ liệu của hệ thống khai phá với một cơ sở dữ liệu hoặc hệ thống kho dữ liệu: không có ghép nối, ghép nối lỏng lẻo, ghép nối khá chặt và ghép nối chặt chẽ. Hãy phát biểu cách tiếp cận bạn nghĩ là phổ biến nhất và nêu lý do tại sao.

1.14 Mô tả ba thách thức đối với khai phá dữ liệu liên quan đến phương pháp khai phá dữ liệu và các vấn đề tương tác với người dùng.

1.15 Những thách thức chính khi khai phá một lượng lớn dữ liệu (như hàng tỷ

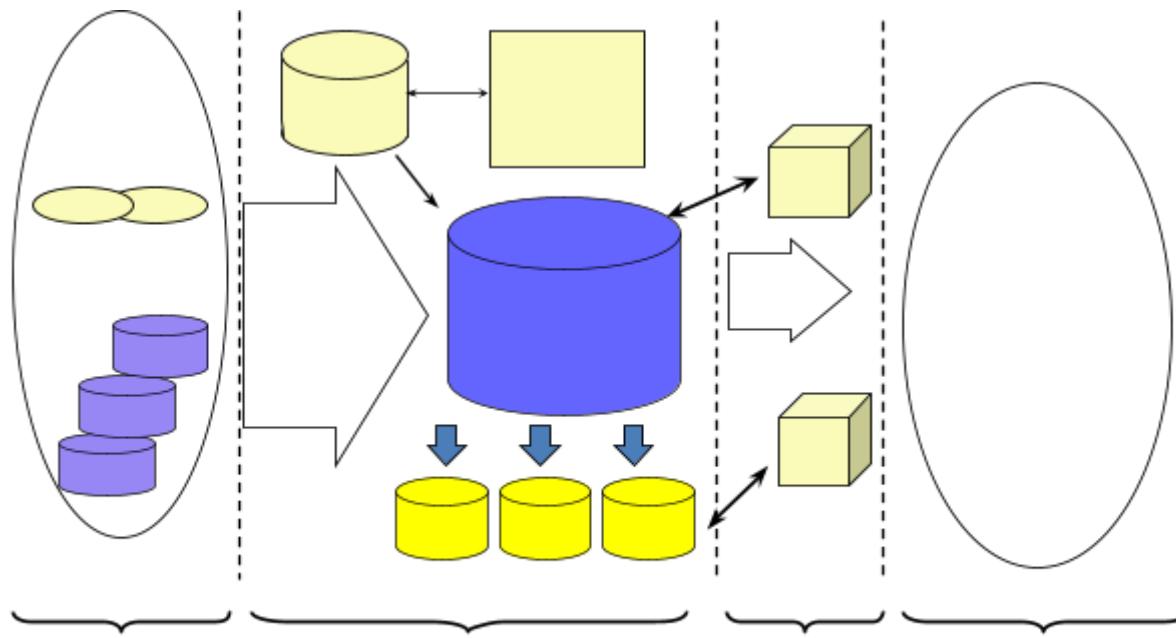
bộ) so với khai phá một lượng nhỏ dữ liệu (chẳng hạn như bộ dữ liệu gồm vài trăm bộ)?

1.16 Phác thảo các thách thức nghiên cứu chính về khai phá dữ liệu trong một miền ứng dụng cụ thể, chẳng hạn như phân tích dữ liệu luồng/cảm biến, phân tích dữ liệu không gian và thời gian hoặc lĩnh vực tin sinh học

Chương 2: Các công nghệ và kỹ thuật tích hợp cơ sở dữ liệu

2.1 Giới thiệu Mô hình dữ liệu mở rộng XML

Phản đầu tiên của môn học sẽ đi vào xem xét các kỹ thuật chuyển đổi và tích hợp dữ liệu vào kho dữ liệu từ các nguồn dữ liệu khác nhau. Trước hết xem xét kiến trúc đa tầng của kho dữ liệu và khai phá dữ liệu thể hiện trong hình vẽ dưới đây



Dữ liệu được thu thập từ nhiều nguồn khác nhau trong đó có cơ sở dữ liệu tác nghiệp và các nguồn dữ liệu khác. Chúng được trích lọc, chuyển đổi và tải vào một nơi lưu trữ được gọi là kho dữ liệu. Ngoài ra dữ liệu còn có thể tích hợp, làm mới để đưa vào kho dữ liệu, sau đó được tổ chức lại để phục vụ cho OLAP và các công cụ đầu cuối của người sử dụng bao gồm công cụ phân tích, truy vấn, báo cáo và khai phá dữ liệu.

Tại thời điểm này chúng ta bắt đầu từ nguồn dữ liệu, ngoài các cơ sở dữ liệu quan hệ được học ở môn học trước, chúng ta xem xét một loại dữ liệu cũng tương đối phổ biến hiện nay là mô hình dữ liệu mở rộng XML.

Giới thiệu về ngôn ngữ XML (Extensible Markup Language)

- Là ngôn ngữ đánh dấu mở rộng, về tính mở rộng thì giống với ngôn ngữ đánh dấu html đã được làm quen từ trước.
- Do tổ chức World Wide Web Consortium (W3C) giới thiệu Version 1.0 vào năm 1998

- Là một ngôn ngữ dùng để miêu tả dữ liệu, chứ không phải hướng dẫn một hệ thống xử lý dữ liệu.
- Cung cấp một công cụ khá mạnh cho việc tích hợp dữ liệu và theo kiểu hướng dữ liệu.
- Giới thiệu một cơ chế xử lý mới và yêu cầu các cách suy nghĩ mới để phát triển web.
- Là một ngôn ngữ siêu đánh dấu nên có một tập các luật để tạo ra những thẻ ngữ nghĩa dùng để miêu tả dữ liệu.
- XML là một ngôn ngữ có khả năng mở rộng khác với HTML
 - o Với HTML, thẻ được sử dụng để đánh dấu tài liệu và cấu trúc của tài liệu HTML được định trước.
 - o Những người sử dụng tài liệu HTML chỉ được sử dụng những thẻ đã được định nghĩa trước trong chuẩn HTML.
 - o XML cho phép người sử dụng định nghĩa thẻ và cấu trúc trong dữ liệu của mình.
- Sử dụng XML mang lại lợi ích bởi những đặc điểm sau
 - o XML có cấu trúc nên dễ học, dễ dùng
 - o Không phụ thuộc vào cấu hình nền phần cứng của hệ thống, cung cấp thông tin văn bản
 - o Là một chuẩn mở
 - o Độc lập với Ngôn ngữ
 - o DOM và SAX là là tập các giao diện mở, độc lập với ngôn ngữ
 - o Sử dụng cho web

Một hệ thống XML điển hình

Hệ thống XML điển hình bao gồm các thành phần được thể hiện như hình vẽ dưới đây



- Thành phần thứ nhất là Tài liệu XML chứa nội dung của văn bản cần thể hiện bằng ngôn ngữ XML
- Thành phần thứ hai là Định dạng kiểu tài liệu XML-DTD, thành phần này xác định cấu trúc và định dạng của văn bản. Đây chính là một thành phần thao tác.
- Thành phần thứ ba là Bộ Phân tích cú pháp XML dùng để xử lý trộn nội dung của văn bản và cấu trúc của văn bản để đưa ra văn bản XML hoàn chỉnh sau khi kiểm tra tính phù hợp của nội dung và định dạng.
- Thành phần thứ tư chính là ứng dụng XML (phân tích đầu ra của bộ phân tích cú pháp để đưa ra được một đối tượng duy nhất)

Sử dụng XML như thế nào?

XML khác biệt với HTML, nó có thể lưu trữ dữ liệu tách biệt khỏi văn bản HTML, chuyển đổi việc thể hiện dữ liệu sang định dạng khác thông qua việc tự định nghĩa cấu trúc DTD.

Cú pháp của XML

Xét một ví dụ văn bản XML đơn giản như sau

```
<?xml version="1.0"?>
<note>
  <to>Tan Siew Teng</to>
  <from>Lee Sim Wee</from>
  <heading>Reminder</heading>
  <body>Don't forget the Golf Championship this weekend!</body>
</note>
```

Ta hiểu ý nghĩa của mỗi dòng trong văn bản XML trên như sau

- Dòng đầu tiên trong tài liệu : khai báo XML version 1.0, dòng này luôn luôn phải có vì nó xác định phiên bản XML của văn bản.
- Trong trường hợp này tài liệu phù hợp với đặc tả 1.0 của XML <?xml version="1.0"?>
- Dòng tiếp theo xác định phần tử đầu tiên của tài liệu (gọi là phần tử gốc hay **root**) <note>
- Bốn dòng tiếp theo định nghĩa 4 phần tử con của **root** là **to**, **from**, **heading** và **body**

```
<to>Tan Siew Teng</to>
<from>Lee Sim Wee</from>
<heading>Reminder</heading>
<body>Don't forget the Golf Championship this weekend!</body>
```

- Dòng cuối cùng định nghĩa sự kết thúc của phần tử **root** bằng thẻ </note>

Một phần tử XML có những đặc điểm sau đây

- Được **tạo bởi** một thẻ bắt đầu hay thẻ mở, một thẻ kết thúc hay thẻ đóng và **dữ liệu ở giữa**

<Sport>Golf</Sport>

- **Tên** của một phần tử được thể hiện bằng các ký tự nằm trong thẻ (tags).
- Thẻ bắt đầu và kết thúc miêu tả dữ liệu trong phạm vi của nó, gọi là **dữ liệu** của một phần tử.

Ví dụ, phần tử XML sau là một phần tử có tên là <player> với dữ liệu là “Tiger Wood.”

<player>Tiger Wood</player>

Thẻ bao gồm 3 loại

- **Thẻ mở:** Trong ví dụ <Sport> là một thẻ mở. Nó xác định các kiểu của một phần tử và thuộc tính đặc tả có thẻ. Ví dụ sau thể hiện thẻ có thuộc tính đặc tả

<Player **firstname=“Wood” lastname=“Tiger”**>

Trong đó **tên** của thẻ là **Player** với hai thuộc tính đặc tả là **firstname** với giá trị **Wood** và **lastname** với giá trị **Tiger**.

- **Thẻ đóng** (End-Tag): Trong **ví dụ** </Sport> là thẻ đóng. Nó xác định kiểu của một phần tử mà thẻ này là kết thúc. Không giống như thẻ mở, thẻ đóng không chứa các thuộc tính đặc tả.
- **Thẻ không có phần tử** (Empty Element Tag): Giống như thẻ mở, nó có **không có** các thuộc tính đặc tả nhưng nó không cần thẻ đóng. Biểu thị phần tử là rỗng, chú ý là có tag “/” trước khi kết thúc. Ví dụ: <Player **firstname=“Wood” lastname=“Tiger”**/>

Một số lưu ý

- Những phần tử XML cần phải có thẻ đóng khác với một số phần tử trong ngôn ngữ HTML. Trong HTML một số phần tử không cần thẻ đóng. Ví dụ mã HTML sau đây là hoàn toàn đúng đắn

<p>This is a paragraph

<p>This is another paragraph

Nhưng không đúng trong XML, trong XML mã phải như sau

<p>This is a paragraph</p>

<p>This is another paragraph</p>

- **Thẻ XML** cần chú ý phân biệt chữ hoa và chữ thường. Ví dụ như thẻ <Message> khác thẻ <message>.

- Thẻ bắt đầu và kết thúc cần phải viết giống nhau. Ví dụ cách viết đầu tiên sau là đúng, còn cách viết thứ hai là không đúng: <message>This is correct</message>
<Message>This is incorrect</message>

Luật đặt tên cho các phần tử XML

- Tên phải bắt đầu bằng chữ cái hoặc dấu gạch dưới
- Phần còn lại của tên có thể chứa chữ cái, chữ số, dấu chấm, dấu gạch dưới hoặc dấu gạch ngang
- Tên không được phép có dấu cách hoặc khoảng trắng
- Tên không được bắt đầu bằng từ khóa “xml”

Các phần tử của XML có thể lồng nhau nhưng phải viết đúng cách

Trong HTML một số phần tử không cần phải lồng theo đúng cách, ví dụ như sau:

```
<b><i>This text is bold and italic</b></i>
```

Trong XML các phần tử phải lồng đúng cách

```
<b><i>This text is bold and italic</i></b>
```

XML phải có một root tag

- XML cần phải bao gồm những cặp thẻ đơn để xác định phần tử root.
- Những phần tử khác phải được lồng trong phạm vi của phần tử root.
- Những phần tử có thẻ có phần tử con.
- Những phần tử con cần phải có trong một cặp và lồng theo đúng trật tự trong phạm vi của phần tử cha. Một mẫu điển hình của văn bản XML như sau trong đó số lượng child và subchild lồng nhau có thể tăng lên cùng cấp hoặc nhiều cấp hơn.

```
<root>
  <child>
    <subchild>
    </subchild>
  </child>
</root>
```

Thuộc tính của XML

- Những thuộc tính XML thường mô tả phần tử XML, hoặc là cung cấp thêm những thông tin về phần tử.

- Một phần tử có thể bao gồm một hay nhiều thuộc tính. Và một thuộc tính là một cặp tên-giá trị phân cách nhau bằng một dấu =
- Chung nhất thì những thuộc tính được sử dụng để cung cấp thông tin nhưng nó không phải là một phần của nội dung trong XML.
- Thường dữ liệu thuộc tính cần cho bộ phân tích XML hơn là cho người dùng.
- Những thuộc tính luôn luôn bao gồm trong thẻ bắt đầu của một phần tử. Sau đây là một số ví dụ :

<Player firstname="Wood" lastname="Tiger" /> trong đó

Player - là tên phần tử; Firstname - là tên thuộc tính; Wood – là giá trị thuộc tính

Ví dụ HTML

Ví dụ XML <file type="gif">

<person id="3344">

- Phần tử XML có thể có những thuộc tính trong cặp tên/giá trị như HTML
 - Một phần tử có thể bao gồm một hay nhiều thuộc tính.
 - Trong XML thuộc tính giá trị cần phải được chỉ rõ.
 - Một thuộc tính là một cặp giá trị tên-giá trị được phân cách bởi một dấu “=”. Ví dụ:
- <CITY ZIP="01085">Westfield</CITY> trong đó ZIP="01085" là một thuộc tính trong phần tử<CITY>.

Chú thích trong XML

- Chú thích trong XML được sử dụng để trợ giúp thông tin cho người đọc
- Chúng được bộ xử lý XML bỏ qua khi chạy
- Các comments luôn nằm trong dấu <!-- -->. VD <!-- đây là comment-- >

Hướng dẫn xử lý (Processing Instruction)

- Hướng dẫn xử lý cung cấp một đường dẫn để hướng dẫn cho chương trình máy tính hoặc ứng dụng. Chúng nằm trong tag “<?” và “>”
- Ví dụ <? xml:stylesheet type="text/xsl" href="styler.xsl" ?> trong đó:
xml:stylesheet là tên ứng dụng
type="text/xsl" href="styler.xsl" là hướng dẫn cho ứng dụng

Khai báo kiểu văn bản – Data Type Declaration (DTD)

Khái niệm DTD

- DTD là một cơ chế (tập các quy tắc) để miêu tả cấu trúc, cú pháp và từ vựng của các tài liệu XML
- Nó là 1 ngôn ngữ mô hình hóa cho XML nhưng nó không tuân theo các cú pháp như XML
- Xác định các khái niệm luật của một tài liệu XML
- Tập các quy tắc để xác định cấu trúc tài liệu với một danh sách các thành phần hợp lệ
- Được khai báo bên trong tài liệu XML hoặc là một tham chiếu ngoài
- Tất cả các tên là do người sử dụng định nghĩa
- Có nguồn gốc từ SGML, Có định dạng ASCII
- Khi ta định nghĩa một DTD, có thể sử dụng nó cho nhiều tài liệu
- Từ khóa DOCTYPE

Khai báo một phần tử trong XML thông qua DTD

- Những dòng sau hiển thị cú pháp có thể cho việc khai báo một phần tử:

<!ELEMENT reports (employee*)> thể hiện việc khai báo một phần tử có tên là reports và có một phần tử con có tên là employee

<!ELEMENT employee (ss_number, first_name, middle_name, last_name, email, extension, birthdate, salary)> thể hiện việc khai báo một phần tử có tên là employee và có các phần tử con bên trong có tên lần lượt là ss_number, first_name, middle_name, last_name, email, extension, birthdate, salary

<!ELEMENT email (#PCDATA)> thể hiện việc khai báo một phần tử có tên là email có thể chứa văn bản trong đó, không chứa các phần tử con loại khác.

<!ELEMENT extension EMPTY> thể hiện việc khai báo một phần tử có tên là extension, và nó là phần tử lá.

Trong đó: #PCDATA – là một loại dữ liệu của bộ phân tích ký tự, có nghĩa là các phần tử có thể chứa văn bản. Yêu cầu này có nghĩa là không có phần tử con xuất hiện trong phần tử có #PCDATA

EMPTY – Chỉ ra rằng đây là phần tử lá không thể chứa thêm bất cứ phần tử con nào

- Ký pháp để thể hiện số lần xuất hiện của một phần tử con trong một phần tử cha sẽ xuất hiện ở cuối mỗi phần tử. Có mấy loại ký hiệu sau để thể hiện số lần xuất hiện

+ thể hiện phần tử đó có thể xuất hiện 1 hoặc nhiều lần

* thể hiện phần tử đó có thể xuất hiện 0 hoặc nhiều lần

? thể hiện phần tử đó có thể xuất hiện 0 hoặc 1 lần

Không có ký hiệu nào thể hiện phần tử đó chỉ được và phải xuất hiện 1 lần duy nhất

- Ký hiệu phân cách (separator)

Dấu phẩy “,” được dùng khi thành phần ở bên phải và trái của dấu phẩy phải xuất hiện cùng thứ tự

Dấu gạch sô “|” được dùng khi chỉ 1 thành phần ở bên phải hoặc trái của kí tự này phải xuất hiện

Khai báo một thuộc tính trong DTD

- Những dòng sau thể hiện cú pháp khai báo thuộc tính của một phần tử

<! ATTLIST Customer ID CDATA # REQUIRED>

<! ATTLIST Customer Preferred(true|false) “false”>

Trong đó Customer là tên phần tử

ID: thuộc tính loại ID xác định duy nhất 1 phần tử

IDREF: thuộc tính loại IDREF chỉ rõ tới phần tử với 1 thuộc tính ID

Preferred là tên thuộc tính

(true|false): giá trị thuộc tính có thể

False: giá trị thuộc tính mặc định

CDATA: dữ liệu kí tự

#Required: giá trị thuộc tính phải được cung cấp

#Implied: nếu không có giá trị được cung cấp, ứng dụng sẽ sử dụng giá trị mặc định của nó

#FIXED: giá trị thuộc tính phải là giá trị được cung cấp trong DTD

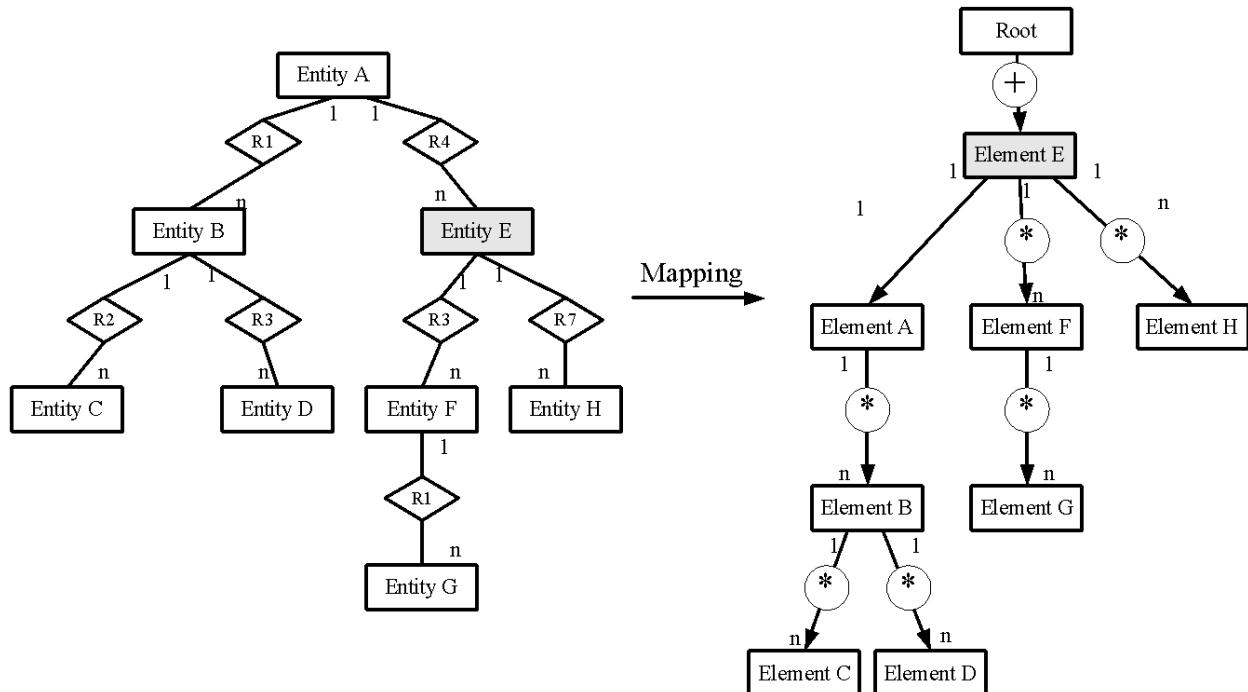
NMTOKEN: tên mã thông báo bao gồm các chữ cái, chữ số, thời gian, gạch dưới, gạch ngang và các kí tự dấu hai chấm

Lý do cần sử dụng DTD

- DTD cung cấp cách chia sẻ dữ liệu độc lập đối với ứng dụng. Hai ứng dụng khác nhau nếu muốn dùng chung định dạng dữ liệu thì có thể dùng chung một DTD
- Với DTD, các nhóm người độc lập nhau có thể thống nhất cách sử dụng một DTD chung cho việc trao đổi dữ liệu, mặc dù dữ liệu có thể khác nhau.
- Ứng dụng của bạn có thể sử dụng DTD chuẩn để xác minh rằng dữ liệu bạn nhận được từ thế giới bên ngoài là hợp lệ
- Bạn cũng có thể sử dụng DTD để xác minh dữ liệu của chính bạn

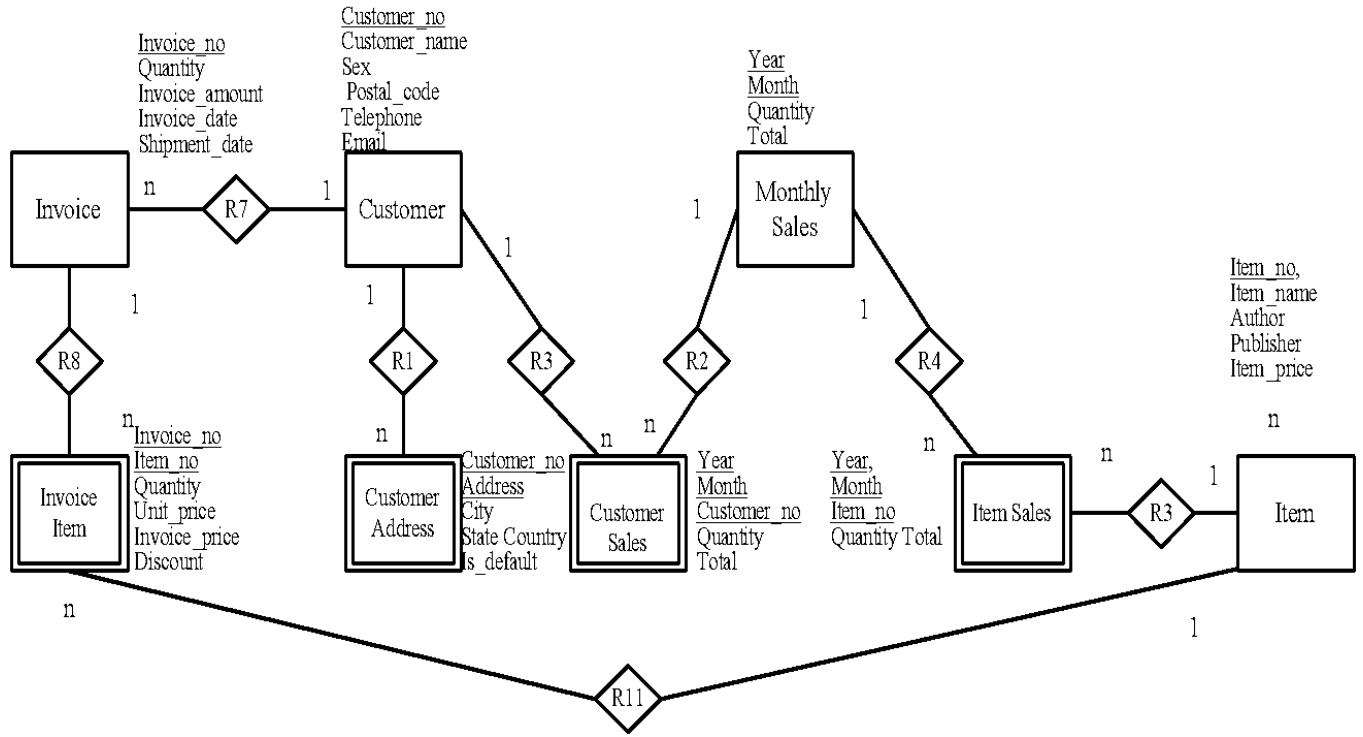
Đồ thị DTD

- Với thông tin DTD của XML được lưu, ta có thể tạo ra một cấu trúc gọi là Đồ thị định nghĩa loại dữ liệu (Data Type Definition Graph) phản ánh cấu trúc của DTD. Mỗi nút trong đồ thị DTD biểu thị một phần tử trong XML bằng hình chữ nhật, một thuộc tính bằng nửa hình tròn và một toán tử trong vòng tròn. Chúng được đặt cùng nhau trong một mô hình phân cấp dưới một nút phần tử gốc, với các nút phần tử ở dưới nút phần tử cha, được phân tách bởi kí hiệu xuất hiện trong vòng tròn.
- Điều kiện thuận lợi là có thể kết nối các thành phần với nhau bằng một ID và IDREF. Một thành phần với IDREF chỉ ra một phần tử với ID. Mỗi ID phải có một địa chỉ duy nhất. Các nút có thể chỉ tới các nút khác bằng cách sử dụng ID và IDREF sao cho các nút với IDREF chỉ tới các nút có ID.
- Một ví dụ về việc chuyển đổi từ mô hình thực thể liên kết mở rộng sang đồ thị DTD được thể hiện trong hình vẽ dưới đây

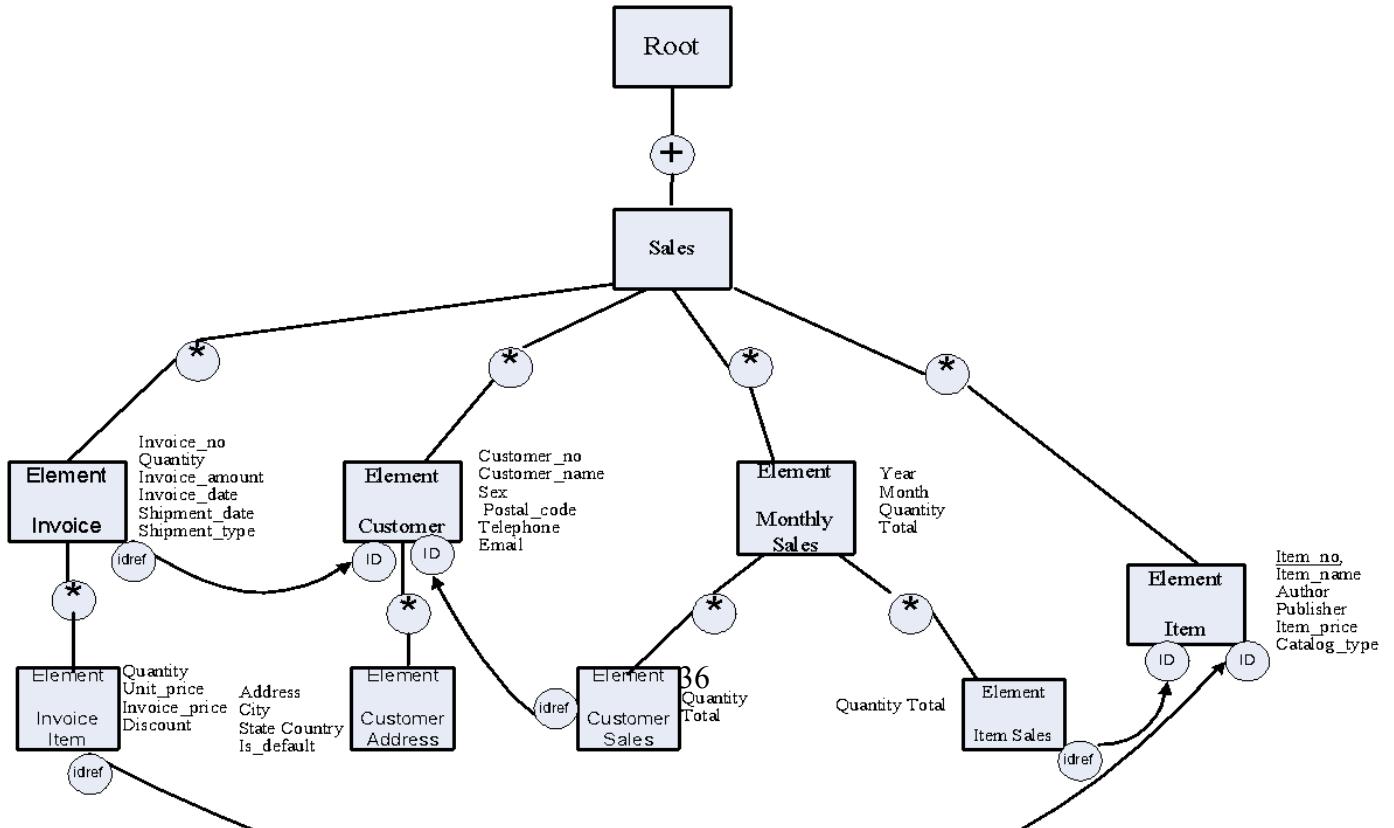


Một ví dụ chuyển đổi từ mô hình thực thể liên kết EER sang XML

Mô hình thực thể liên kết mở rộng cho việc bán hàng được thể hiện trong hình vẽ dưới đây



Mô hình này được chuyển sang đồ thị DTD thể hiện trong hình vẽ dưới đây



Đồ thị DTD được chuyển đổi sang DTD của XML như sau

```
<!ELEMENT Sales (Invoice*, Customer*, Item*, Monthly_sales*)>
<!ATTLIST Sales Status (New | Updated | History) #required>
<!ELEMENT Invoice           (Invoice_item*)>
<!ATTLIST Invoice
  Invoice_no          CDATA    #REQUIRED
  Quantity            CDATA    #REQUIRED
  Invoice_amount      CDATA    #REQUIRED
  Invoice_date        CDATA    #REQUIRED
  Shipment_date       CDATA    #IMPLIED
  Customer_idref     IDREF#REQUIRED>
<!ELEMENT Customer        (Customer_address*)>
<!ATTLIST Customer
  Customer_id         ID      #REQUIRED
  Customer_name       CDATA    #REQUIRED
  Customer_no         CDATA    #REQUIRED
  Sex                 CDATA    #IMPLIED
  Postal_code         CDATA    #IMPLIED
  Telephone          CDATA    #IMPLIED
  Email               CDATA    #IMPLIED>
<!ELEMENT Customer_address EMPTY>
<!ATTLIST Customer_address
  Address_type (Home|Office)      #REQUIRED
  Address          NMTOKENS #REQUIRED
  City             CDATA    #IMPLIED
  State            CDATA    #IMPLIED
```

```

Country          CDATA      #IMPLIED
Customer_idref IDREF      #REQUIRED
Is_default      (Y|N) "Y">
<!ELEMENT Invoice_Item    EMPTY>
<!ATTLIST Invoice_Item
  Quantity      CDATA      #REQUIRED
  Unit_price   CDATA      #REQUIRED
  Invoice_price CDATA      #REQUIRED
  Discount     CDATA      #REQUIRED
  Item_idref   IDREF      REQUIRED>
<!ELEMENT Item           EMPTY>
<!ATTLIST Item
  Item_id       ID         #REQUIRED
  Item_name    CDATA      #REQUIRED
  Author       CDATA      #IMPLIED
  Publisher   CDATA      #IMPLIED
  Item_price   CDATA      #REQUIRED>
<!ELEMENT Monthly_sales (Item_sales*, Customer_sales*)>
<!ATTLIST Monthly_sales
  Year        CDATA      #REQUIRED
  Month       CDATA      #REQUIRED
  Quantity    CDATA      #REQUIRED
  Total       CDATA      #REQUIRED>
<!ELEMENT Item_sales    EMPTY>
<!ATTLIST Item_sales
  Quantity    CDATA      #REQUIRED
  Total       CDATA      #REQUIRED
  Item_idref IDREF#REQUIRED>
<!ELEMENT Customer_sales EMPTY>
<!ATTLIST Customer_sales
  Quantity    CDATA      #REQUIRED

```

Total	CDATA	#REQUIRED
Customer_idref	IDREF	#REQUIRED>

Một tài liệu XML chuẩn tuân thủ một số quy tắc nhất định

- Dòng đầu tiên của một tài liệu XML well-formed phải là một khai báo XML
- Tất cả các thành phần không rỗng phải có tag mở và tag đóng với tên các thành phần
- Tất cả các thành phần rỗng phải kết thúc bằng />
- Tất cả các tài liệu phải chứa 1 thành phần gốc
- Các thành phần con phải nằm trong thành phần cấp cao hơn của chúng
- Các tham chiếu thực thể riêng là: &, ', >, < và "

Một ví dụ của tài liệu XML chuẩn thể hiện dưới đây

```
<?xml version="1.0"?>
<TITLE>
<title>A Well-formed Documents</title>
<first>
  This is a simple
  <bold>well-formed</bold>
  document.
</first>
</TITLE>
```

2.2 Chuyển đổi lược đồ dữ liệu giữa các mô hình

Nhắc lại kiến thức về mô hình thực thể liên kết mở rộng (đã học trong môn Cơ sở dữ liệu)

Mô hình thực thể liên kết được sử dụng rất rộng rãi trong thiết kế cơ sở dữ liệu nhưng vẫn có một số nhược điểm nhất định trong việc thể hiện ngữ nghĩa. Với mô hình thực thể liên kết thông thường thì việc biểu diễn các trường hợp mà một thực thể có thuộc tính thay đổi phụ thuộc vào một đặc tính nào đó.

Chính vì thế, mô hình thực thể liên kết đã được mở rộng thành EER miêu tả ngữ nghĩa nhiều hơn ví dụ như tổng quát hóa, phân loại và tích hợp. Chúng ta sẽ xem xét lần lượt từng loại ngữ nghĩa trên được biểu diễn như thế nào trong mô hình thực thể liên kết mở rộng.

Quan hệ một-một: Mỗi quan hệ 1-1 giữa tập A và tập B được định nghĩa như sau

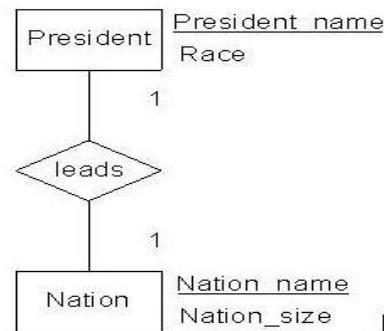
- Với mỗi thuộc tính a thuộc A đều tồn tại nhiều nhất 1 thuộc tính b trong B sao cho a và b có quan hệ với nhau và ngược lại.
- Ví dụ: Một tổng thống (President) đứng đầu một nước.
- Mô hình quan hệ:

Relation President (President_name, Race, *Nation_name)

Relation Nation (Nation_name, Nation_size)

Dưới dấu gạch chân là khóa chính, * là khóa ngoại.

- Mô hình thực thể liên kết mở rộng



Quan hệ nhiều-một: Mỗi quan hệ nhiều-một giữa tập A và tập B được định nghĩa như sau

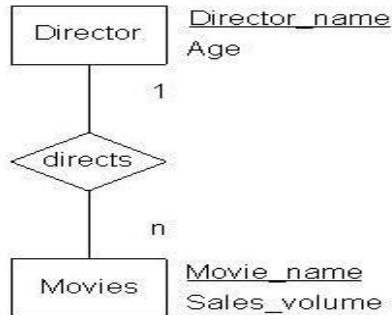
- VỚI mỗi thuộc tính a thuộc A đều tồn tại nhiều nhất 1 thuộc tính b trong B sao cho a và b có quan hệ với nhau và với mỗi thuộc tính b thuộc B, sẽ tồn tại 0 hoặc một a thuộc A sao cho a và b có quan hệ với nhau.

- Ví dụ: Một đạo diễn có nhiều phim.
- Mô hình quan hệ:

Relation Director (Director_name, Age)

Relation Movies (Movie_name, Sales_volume, *Director_name)

- Mô hình thực thể liên kết mở rộng



Quan hệ nhiều-nhiều: Mọi quan hệ nhiều-nhiều giữa tập A và tập B được định nghĩa như sau

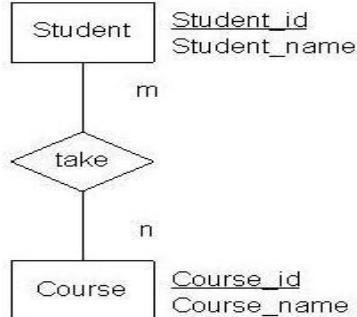
- Với mỗi thuộc tính a thuộc A sẽ tồn tại 0 hoặc nhiều hơn một thuộc tính b thuộc B sao cho a và b có quan hệ với nhau và ngược lại.
- Ví dụ: Nhiều sinh viên tham gia nhiều khóa học, một sinh viên tham gia nhiều khóa học và một khóa học có nhiều sinh viên.
- Mô hình quan hệ:

Relation Student (Student_id, Student_name)

Relation Course (Course_id, Course_name)

Relation take (*Student_id, *Course_id)

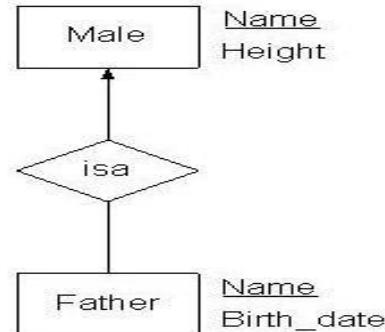
- Mô hình thực thể liên kết mở rộng



Ngữ nghĩa là-một của dữ liệu: Mọi quan hệ là-một (tập con) giữa tập A và tập B được định nghĩa như sau

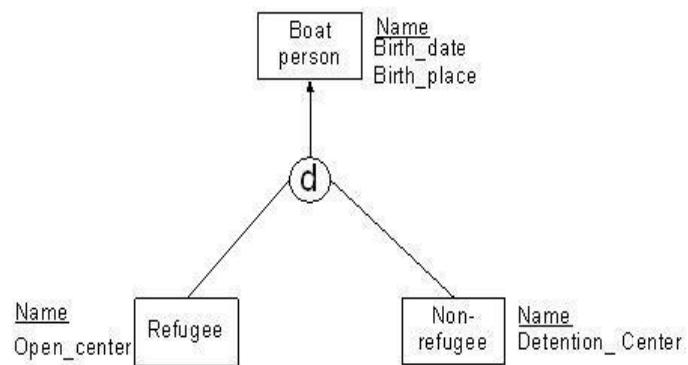
- Quan hệ A là-một B khi A là một loại (trường hợp) đặc biệt của B.

- Ví dụ: Bố là đàn ông trong đó Bố là một trường hợp đặc biệt của đàn ông vì có những đàn ông không phải là Bố.
- Mô hình quan hệ:
 - Relation Male (Name, Height)
 - Relation Father (*Name, Birth_date)
- Mô hình quan hệ thực thể mở rộng:



Ngữ nghĩa khái quát hóa không giao nhau của dữ liệu

- Khái quát hóa là phân loại các thực thể tương tự nhau thành các thực thể đơn. Nhiều hơn một quan hệ là-một có thể hình thành việc trừu tượng hóa dữ liệu này (ví dụ như siêu lớp và lớp con) giữa các thực thể.
- Một thực thể lớp con là một tập con của một thực thể lớp cha (siêu lớp) của nó. Có hai loại khái quát hóa dữ liệu
 - o Thứ nhất là khái quát hóa dữ liệu không giao nhau có nghĩa là các thực thể lớp con loại trừ nhau (không trùng nhau).
 - o Thứ hai là khái quát có giao nhau nghĩa là các thực thể lớp con có thể trùng lặp nhau.
- Ví dụ về khái quát hóa không giao nhau: Một người tị nạn và không phải tị nạn có thể đều là người vượt biên, nhưng người tị nạn chắc chắn không phải người không tị nạn và ngược lại.
- Mô hình quan hệ:
 - Relation Boat_person (Name, Birth_date, Birth_place)
 - Relation Refugee (*Name, Open_center)
 - Relation Non-refugee (*Name, Detention_center)
- Mô hình thực thể liên kết mở rộng



Ngữ nghĩa khái quát hóa không giao nhau của dữ liệu

-Ví dụ: Một người lập trình viên và một nhà phân tích hệ thống cả hai đều là chuyên gia máy tính, một lập trình viên có thể là nhà phân tích và ngược lại.

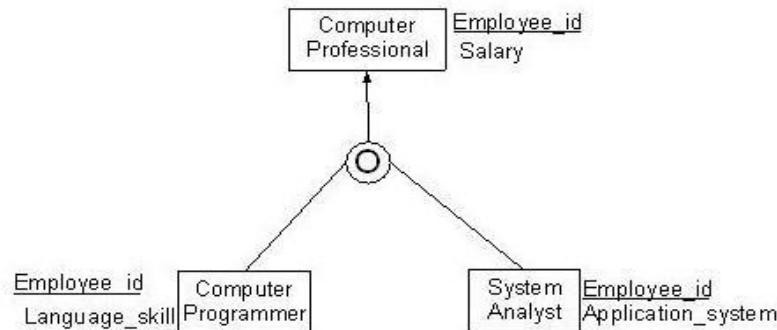
- Mô hình quan hệ:

Relation Computer_professional (Employee_id, Salary)

Relation Computer_programmer (*Employee_id, Language_skill)

Relation System_analyst (*Employee_id, Application_system)

- Mô hình thực thể liên kết mở rộng

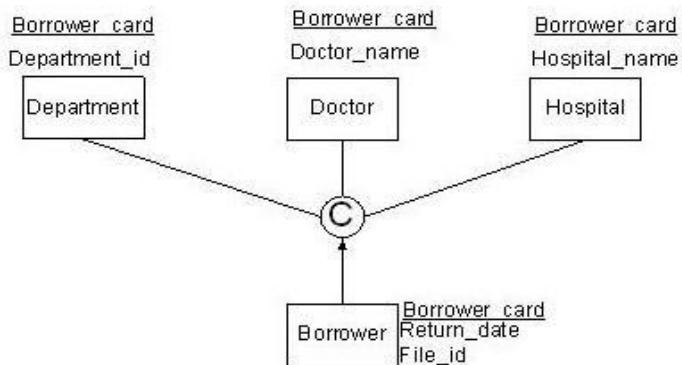


Ngữ nghĩa phân loại của dữ liệu Quan hệ phân loại được định nghĩa như sau

- Trong trường hợp có nhu cầu mô hình hóa quan hệ đơn cha/con với nhiều hơn một siêu lớp, nơi mà các lớp cha đại diện cho các thực thể khác. Trong trường hợp này ta gọi lớp con là một loại.
- Mô hình quan hệ:

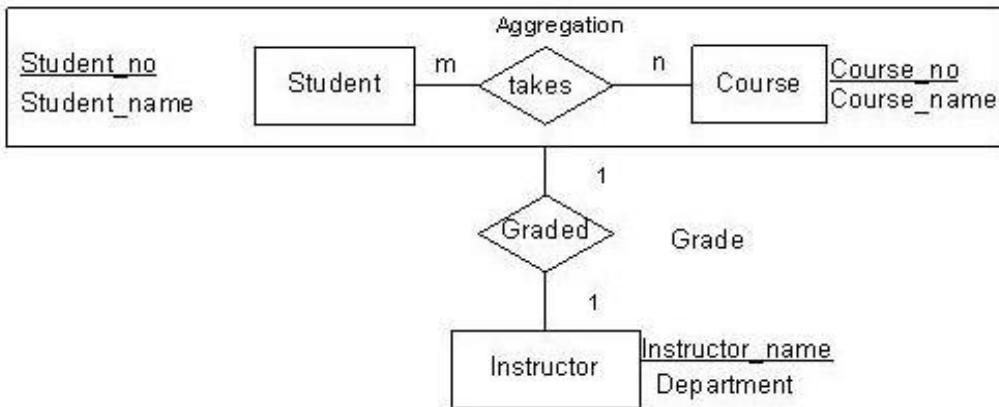
Relation Department (Borrower_card, Department_id)
 Relation Doctor (Borrower_card, Doctor_name)
 Relation Hospital (Borrower_card, Hospital_name)
 Relation Borrower (*Borrower_card, Return_date, File_id)

- Mô hình thực thể liên kết mở rộng



Ngữ nghĩa tích hợp của dữ liệu Quan hệ tích hợp được định nghĩa như sau

- Tích hợp là một phương pháp để hình thành một đối tượng kết hợp từ các thành phần của nó.
Nó tích hợp các giá trị thuộc tính của một thực thể để hình thành một thực thể tích hợp.
- Ví dụ: Quá trình tham gia một khóa học của một sinh viên hình thành một thực thể tích hợp mà có thể được đánh giá bằng điểm số bởi một giảng viên khi sinh viên hoàn thành khóa học đó.
- Mô hình quan hệ:
 - Relation Student (Student_no, Student_name)
 - Relation Course (Course_no, Course_name)
 - Relation Takes (*Student_no, *Course_no, *Instructor_name)
 - Relation Instructor (Instructor_name, Department)
- Mô hình thực thể liên kết mở rộng



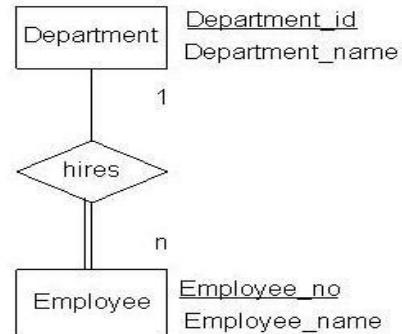
Ngữ nghĩa tham gia toàn bộ của dữ liệu

- Một thực thể tham gia toàn bộ với một thực thể khác khi tất cả thuộc hiện dữ liệu của thực thể đó tham gia vào quan hệ với thực thể kia.
- Ví dụ: Một nhân viên nằm trong 1 phòng ban.
- Mô hình quan hệ:

Relation Department (Department_id, Department_name)

Relation Employee (Employee_id, Employee_name, *Department_id)

- Mô hình thực thể liên kết mở rộng



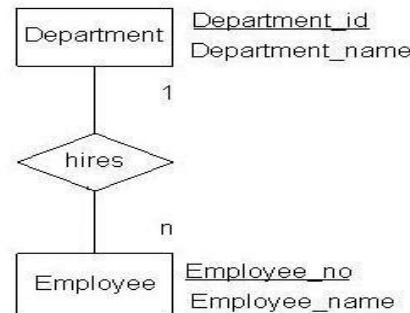
Ngữ nghĩa tham gia một phần của dữ liệu

- Một thực thể tham gia một phần với một thực thể khác nếu dữ liệu của thực thể này không tham gia toàn bộ vào quan hệ với thực thể kia.
- Ví dụ: Một nhân viên có thể được tuyển vào một phòng ban.
- Mô hình quan hệ:

Relation Department (Department_id, Department_name)

Relation Employee (Employee_no, Employee_name, &Department_id) với dấu “&” có nghĩa là được phép nhận giá trị null.

- Mô hình thực thể liên kết mở rộng

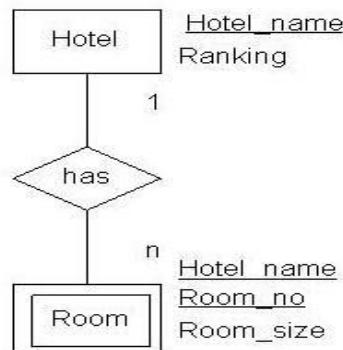


Ngữ nghĩa thực thể yếu của dữ liệu

- Sự tồn tại của một thực thể yếu phụ thuộc vào thực thể mạnh của nó.
- Ví dụ: Một phòng khách sạn phải gắn với tên của khách sạn để xác định.
- Mô hình quan hệ:

Relation Hotel (Hotel_name, Ranking)

Relation Room (*Hotel_name, Room_no, Room_size)



- Mô hình thực thể liên kết mở rộng

Ngữ nghĩa quan hệ nhiều ngôi (N-ary)

- Nhiều thực thể liên kết với nhau để tạo thành quan hệ N-ary.
- Ví dụ: Nhân viên sử dụng hàng loạt các kỹ năng khác nhau để mỗi dự án liên kết với nhau.
- Mô hình quan hệ:

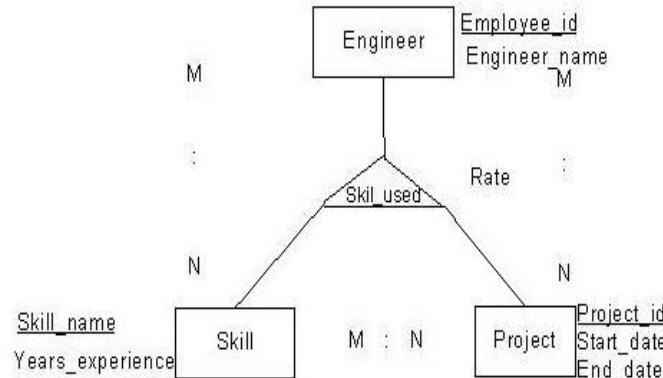
Relation Engineer (Employee_id, Employee_name)

Relation Skill (Skill_name, Years_experience)

Relation Project (Project_id, Start_date, End_date)

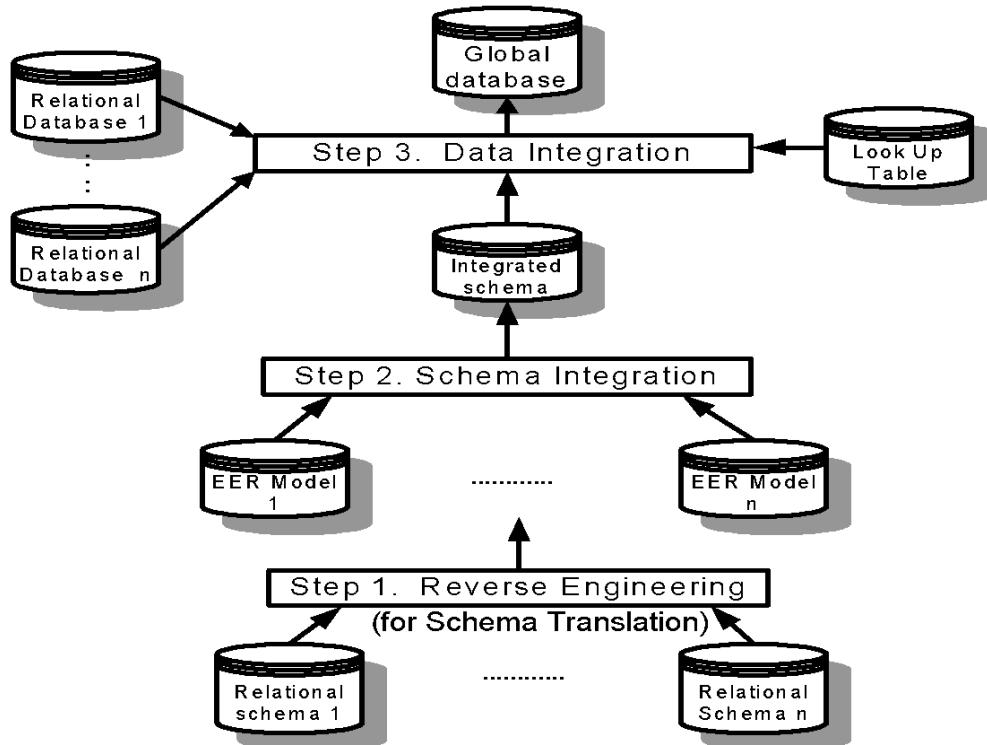
Relation Skill_used (*Employee_id, *Skill_name, *Project_id)

- Mô hình thực thể liên kết mở rộng



Kiến trúc tích hợp nhiều cơ sở dữ liệu

Việc tích hợp nhiều cơ sở dữ liệu được thể hiện trong hình vẽ dưới đây



Trong hình vẽ, ta thấy quá trình tích hợp n CSDL bao gồm ba bước lớn. Bước thứ nhất được gọi là công nghệ chuyển đổi ngược từ các lược đồ quan hệ sang mô hình thực thể liên kết mở rộng. Bước này được gọi là chuyển đổi ngược bởi vì thông thường khi xây dựng hệ thống CSDL chúng ta thường mô tả yêu cầu xây dựng thông qua mô hình thực thể liên kết mở rộng rồi dùng

kỹ thuật chuyển đổi từ mô hình thực thể liên kết sang lược đồ quan hệ (đã được học trong môn Cơ sở dữ liệu). Nhưng ở đây, xuất phát từ các CSDL nguồn với các lược đồ quan hệ tương ứng, ta cần chuyển đổi chúng sang mô hình thực thể liên kết rồi mới tích hợp các lược đồ này lại. Bước thứ hai, ta tích hợp các mô hình thực thể liên kết mở rộng thành một lược đồ thống nhất. Bước thứ ba xuất phát từ lược đồ thực thể thống nhất này, để dữ liệu từ n CSDL nguồn để thu được một CSDL thống nhất toàn cục. Bước thứ ba này cần tham chiếu tới các bảng tra cứu thông tin của dữ liệu (lookup table).

Trong bài học này chúng ta đi vào xem xét bước thứ nhất của kiến trúc trên, kỹ thuật chuyển đổi ngược từ CSDL quan hệ các bước sau sẽ lần lượt được xem xét trong các bài học tiếp theo.

Kỹ thuật chuyển đổi lược đồ quan hệ sang mô hình thực thể liên kết mở rộng

Việc chuyển đổi này gồm nhiều bước được mô tả lần lượt dưới đây

Bước 1: Xác định kiểu quan hệ, khóa và các trường.

Các quan hệ được tiền xử lí bằng cách tạo ra sự thay thế các khóa ứng cử cần thiết như sau:

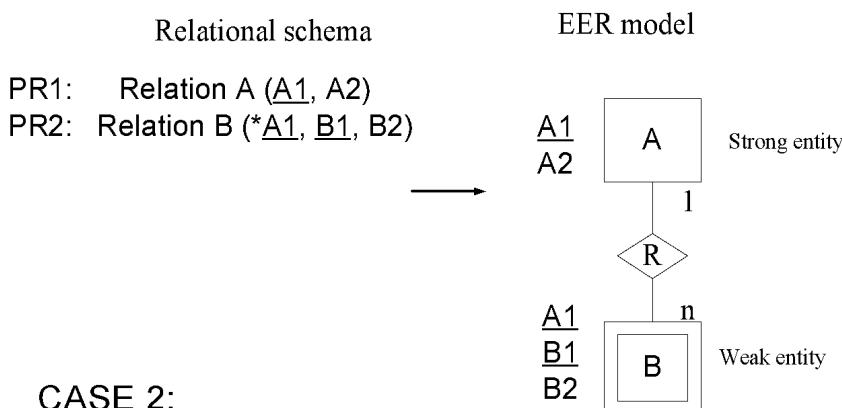
- Loại quan hệ chính PR(primary relation): mô tả những thực thể:
 - PR1: Đây là quan hệ mà khóa chính không chứa khóa của quan hệ khác.
 - PR2: Là quan hệ mà khóa chính chứa khóa của quan hệ khác.
- Loại quan hệ phụ SR (Second relation): là quan hệ mà khóa chính được hình thành đầy đủ hoặc một phần từ khóa chính của các quan hệ khác.
 - SR1: Nếu khóa của SR được hình thành từ khóa chính của quan hệ PR thì là SR1.
 - SR2: Không phải loại 1.
- Thuộc tính khóa (KAP): Là một thuộc tính trong khóa chính của một quan hệ và cũng là khóa ngoại của quan hệ khác.
- Thuộc tính khóa chung (KAG): Tất cả thuộc tính khóa chính khác trong quan hệ SR mà không phải loại KAP.
- Thuộc tính khóa ngoại (FKA): đây là thuộc tính khóa không chính của một quan hệ chính (PR) và đồng thời là khóa ngoại của một quan hệ khác.
- Thuộc tính không khóa (NKA): Những thuộc tính khóa không chính mà không phải là FKA thì là NKA.

Bước 2: Chuyển đổi mỗi PR1 thành một thực thể.

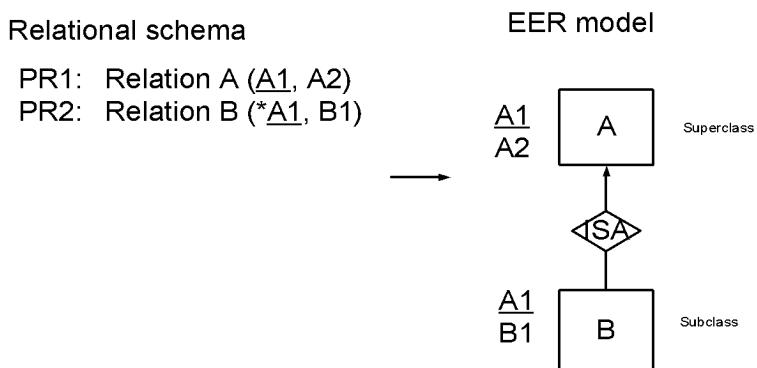
- Đối với mỗi quan hệ kiểu PR1 xác định một loại thực thể tương ứng và xác định nó bằng một khóa chính. Đó là thuộc tính không khóa ánh xạ tới thuộc tính của các loại thực thể với phạm vi giá trị tương ứng.

Bước 3 Ánh xạ mỗi PR2 sang một thực thể lớp con hoặc là một thực thể yếu

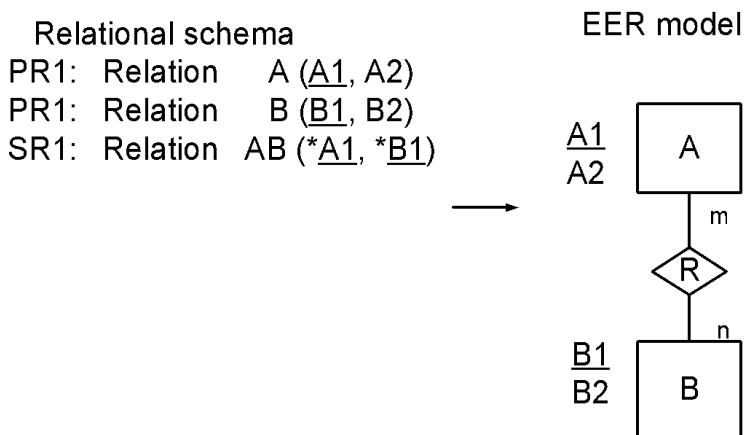
Case 1:



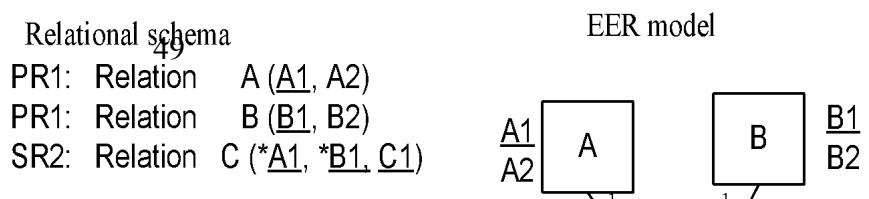
CASE 2:



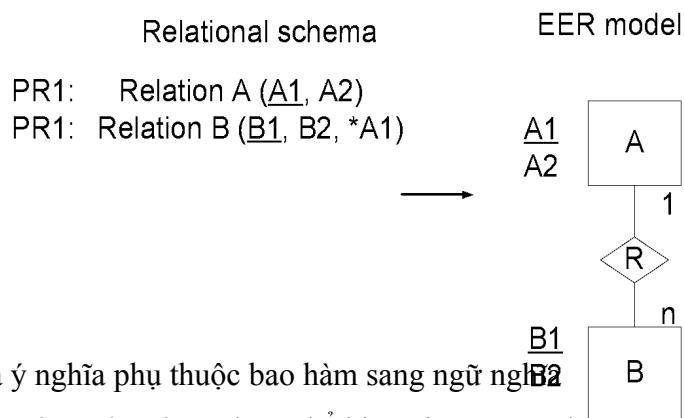
Bước 4: Ánh xạ mỗi SR1 sang quan hệ hai ngôi hoặc quan hệ nhiều ngôi



Bước 5: Ánh xạ mỗi SR2 sang quan hệ hai ngôi hoặc quan hệ nhiều ngôi



Bước 6: Ánh xạ mỗi KFA sang 1 mối quan hệ



Bước 7: Ánh xạ ý nghĩa phụ thuộc bao hàm sang ngũ nghĩa

- Nếu như phụ thuộc bao hàm được thể hiện giữa 2 quan hệ , quan hệ A với khóa chính là a và khóa ngoại là b' , quan hệ B với khóa chính là b và khóa ngoại là a'

Trường hợp 1: *cho phụ thuộc bao hàm a' là con của a* thì thực thể A có quan hệ một-nhiều với B

Trường hợp 2 : *cho phụ thuộc bao hàm a' là con của a, b' là con của b* thì thực thể A có quan hệ 1:1 với thực thể B

Trường hợp 3: *cho phụ thuộc bao hàm a' là con a, b' là con của b, và a' b' là một khóa ghép* thì thực thể A có quan hệ nhiều-nhiều với thực thể B

Bước 8: Vẽ mô hình EER với những kết quả thu được từ các bước trên

Ví dụ về việc chuyển đổi từ lược đồ quan hệ sang mô hình thực thể liên kết

Xét một ví dụ hệ thống đăng ký học ở một trường Đại học bao gồm các mối quan hệ sau

Lược đồ nghĩa tiếng Việt	Lược đồ trong tiếng Anh
Khoa (<u>maKhoa</u> , tenKhoa)	Department (<u>Dept#</u> , Dept_name,)
Giảng Viên (* <u>maKhoa</u> , <u>tenGV</u> , diachiGV)	Instructor (* <u>Dept#</u> , <u>Inst_name</u> , Inst_addr)
Khóa học (<u>maKH</u> , diadiem)	Course (<u>Course#</u> , Course_location)
Điều Kiện (<u>maDK</u> , tenDK ,* <u>maKH</u>)	Prerequisite (<u>Prer#</u> , Prer_title, * <u>Course#</u>)
Sinh Viên (<u>maSV</u> ,tenSV)	Student (<u>Student#</u> , Student_name)
Lớp (* <u>maKhoa</u> ,* <u>maKH</u> ,* <u>tenGV</u> , <u>maSection</u>)	Section (* <u>Dept#</u> , * <u>Course#</u> , * <u>Inst_name</u> , <u>Section#</u>)
Điểm (* <u>maKhoa</u> ,* <u>tenGV</u> ,* <u>maKH</u> ,* <u>maSV</u> ,* <u>maSection</u> ,diem)	Grade (* <u>Dept#</u> , * <u>Inst_name</u> , * <u>Course#</u> , * <u>Student#</u> , * <u>Section#</u> , Grade)

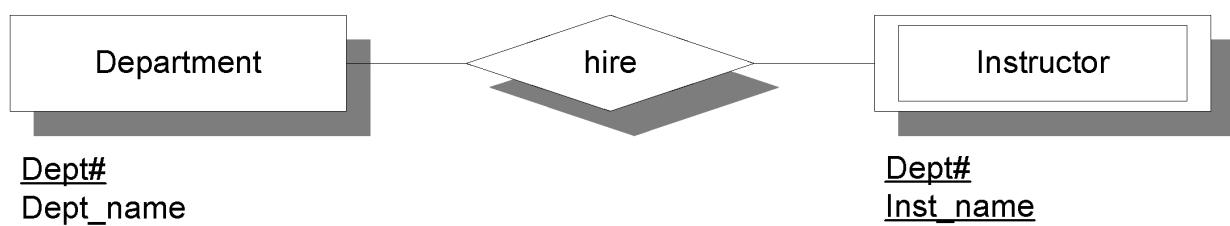
Bước 1: Bảng phân loại các quan hệ và các thuộc tính theo định nghĩa trong phần 3

Relation Name	Reltype	Primary Key	KAP	KAG	FKA	NKA
Khoa	PR1	maKhoa				tenKhoa
Giảng Viên	PR2	maKhoa tenGV	maKhoa	tenGV		diachiGV
Khóa học	PR1	maKH				diadiem
Sinh Viên	PR1	maSV	maKH			tenDK
Điều Kiện	PR1	maDK			maKH	tenSV
Section	SR2	maKH maKhoa maSection tenGV	maKhoa maKH tenGV	maSection		
Điểm	SR1	tenGV maKH maSV maKhoa maSection	maKhoa tenGV maKH maSV maSection			diem

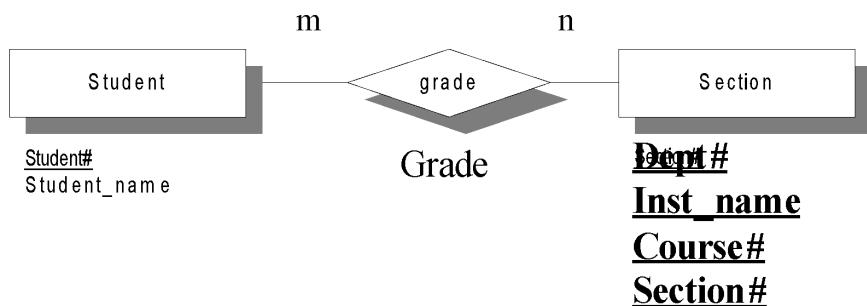
Bước 2: Ánh xạ PR1 thành thực thể

- 
 - 
 - 
 - 
- maKhoa* *maKH* *maDK* *maSV*
 • *tenKhoa* *diadiem* *tenDK* *tenSV*

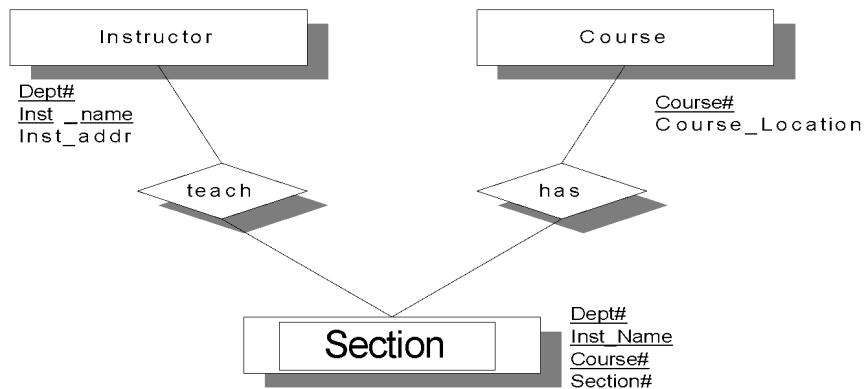
Bước 3: Ánh xạ PR2 sang thực thể yếu



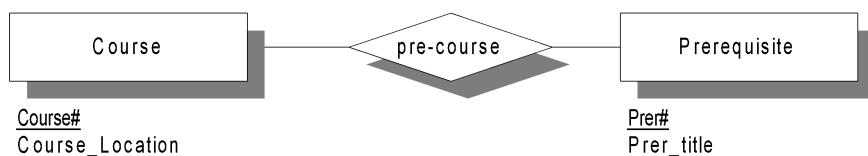
Bước 4: Ánh xạ SR1 thành quan hệ hai ngôi hoặc nhiều ngôi



Bước 5: Ánh xạ SR2 thành quan hệ hai ngôi hoặc nhiều ngôi



Bước 6: Ánh xạ mỗi FKA thành một quan hệ



Bước 7: Ánh xạ phụ thuộc bao hàm sang ngữ nghĩa

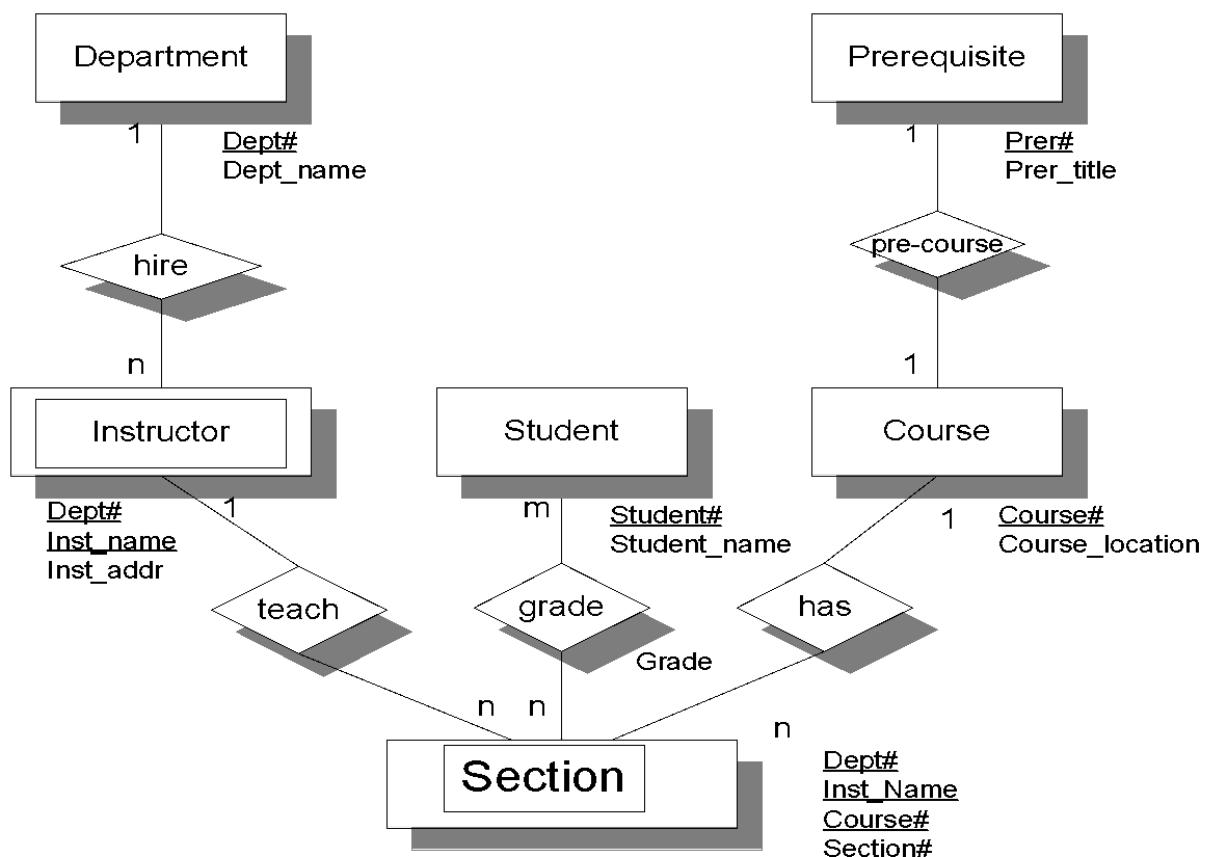
Các phụ thuộc bao hàm phái sinh	Ngữ nghĩa suy ra
Instructor.Dept# ⊆ Department.Dept#	Quan hệ nhiều-một giữa thực thể Instructor và Department

Section.Dept# \subseteq Department.Dept#	Quan hệ một-nhiều giữa thực thể Instructor và Section, giữa Course và Section
Section.Inst_name \subseteq Instructor.Inst_name	
Section.Course# \subseteq Course.Course#	

Grade.Dept# \subseteq Section.Dept#	Quan hệ nhiều-nhiều giữa thực thể Section và Student
Grade.Inst_name \subseteq Section.Inst_name	
Grade.Course# \subseteq Section.Course#	
Grade.Student# \subseteq Student.Student#	

Prerequisite.Course# \subseteq Course.Course#	Quan hệ một-một giữa thực thể Course và Prerequisite
Course.Prer# \subseteq Prerequisite.Prer#	

Bước 8: Từ kết quả của 7 bước trên ta thu được mô hình EER như hình vẽ dưới đây



2.3 Tích hợp các lược đồ dữ liệu

Sau bước đầu tiên là chuyển đổi ngược các lược đồ quan hệ của các cơ sở dữ liệu nguồn sang mô hình thực thể liên kết mở rộng, bước thứ hai trong kiến trúc đa tầng của tích hợp dữ liệu là nhiệm vụ tích hợp các mô hình thực thể liên kết mở rộng thành một lược đồ tích hợp thống nhất. Chúng ta sẽ xem xét công việc đó trong bài học này.

Khái niệm về tích hợp dữ liệu

- Ngữ nghĩa dữ liệu rất quan trọng vì nó xác định mối quan hệ giữa dữ liệu và yêu cầu dữ liệu của người dùng vì thế cần tìm cách để biểu diễn ngữ nghĩa trong quá trình mô hình hóa dữ liệu cũng như trong quá trình tích hợp dữ liệu từ nhiều nguồn vào một mô hình tích hợp duy nhất.
- Ngữ nghĩa dữ liệu được biểu diễn trong lược đồ khái niệm cơ sở dữ liệu như mô hình thực thể liên kết mở rộng EER và đồ thị DTD.
- Một ứng dụng chưa chắc đã cần hết toàn bộ dữ liệu từ các nguồn khác nhau vì thế chỉ các dữ liệu liên quan tới ứng dụng mới được tích hợp vào trong ứng dụng.
- Việc xác định những dữ liệu nào là liên quan và cần thiết cho ứng dụng phụ thuộc phần lớn vào yêu cầu dữ liệu của người dùng cho ứng dụng đó.
- Sự nhất quán của dữ liệu được coi là chuẩn mực cho miền giá trị và định dạng của dữ liệu. Dữ liệu không nhất quán phải được chuyển đổi trước khi tích hợp.
- Việc tích hợp dữ liệu nói chung là cần sự giám sát của người sử dụng. Việc giám sát có nghĩa là người sử dụng đưa đầu vào cho các yêu cầu dữ liệu người dùng, và những yêu cầu đó là cho việc thiết kế dữ liệu và lược đồ tích hợp dữ liệu.

Thuật toán tích hợp lược đồ của các mô hình thực thể liên kết mở rộng

Begin For mỗi cơ sở dữ liệu cần tích hợp **do**

If lược đồ khái niệm chưa tồn tại

then xây dựng lại mô hình thực thể liên kết mở rộng (EER model) của nó bằng kỹ thuật chuyển đổi ngược;

For mỗi cặp lược đồ EER A và B **do**

Begin giải quyết xung đột ngữ nghĩa dữ liệu giữa EER A và B; /*bước 1*/

Kết hợp các thực thể giữa lược đồ EER A và EER B; /*bước 2*/

Kết hợp các quan hệ giữa EER A và EER B; /*bước 3*/

End

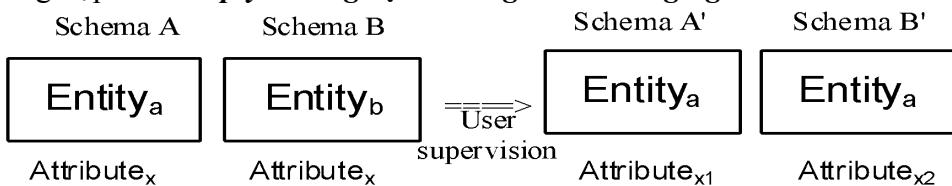
End

Các bước tích hợp ngữ nghĩa dữ liệu

Xét một cặp lược đồ thực thể liên kết A và B, chúng ta cần tích hợp ngữ nghĩa dữ liệu của chúng

Bước 1: Giải quyết xung đột giữa các lược đồ EER

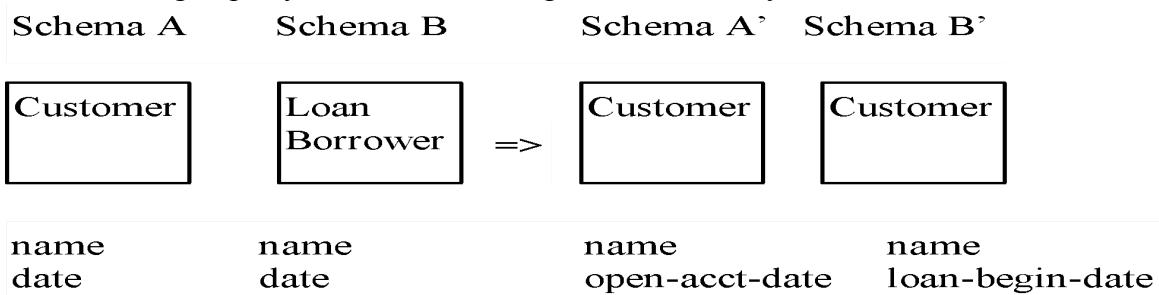
Trường hợp 1: *Giải quyết xung đột về đồng âm và đồng nghĩa*



Hai lược đồ A và B có hai thực thể a và b, bản chất giống nhau nhưng tên gọi khác nhau, đó là trường hợp đồng nghĩa. Với sự giám sát của người sử dụng thì việc đồng nghĩa này sẽ được giải quyết khi tích hợp hai lược đồ A và B vào một lược đồ thống nhất như trên hình vẽ trên có nghĩa là sẽ đổi tên một trong hai thực thể a và b thành tên của thực thể kia để cả hai thực thể cùng bản chất sẽ cùng tên và hợp nhất thành một tập thực thể, ở đây thực thể b được chuyển tên thành thực thể a.

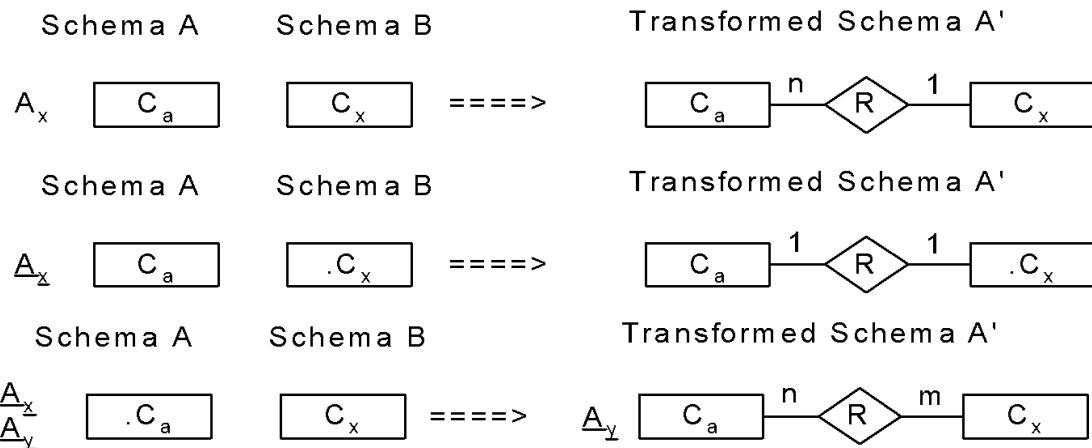
Thực thể a và b của hai lược đồ A và B có thuộc tính với tên là x nhưng bản chất hai thuộc tính này khác nhau, nói cách khác chúng là dạng đồng âm (cùng tên) nhưng khác nghĩa. Giải quyết trường hợp này khi tích hợp, chúng ta cần đổi tên thuộc tính sao cho chúng phân biệt nhau, không còn gây nhầm lẫn khi định danh chúng. Cụ thể trong hình vẽ trên, thuộc tính x của thực thể a được chuyển thành x1 và thuộc tính x của thực thể b được chuyển thành x2.

Ví dụ của trường hợp này được thể hiện trong hình vẽ dưới đây



Trong ví dụ này, thực thể của hai lược đồ là đồng nghĩa, đều là khách hàng nên được chuyển thành thực thể Customer trước khi tích hợp. Hai thuộc tính cùng tên là name date nhưng có ý nghĩa khác nhau, ở lược đồ A là ngày mở tài khoản của khách hàng, còn ở lược đồ B là ngày bắt đầu tính nợ. Vì thế, chúng được đổi tên để phân biệt nhau.

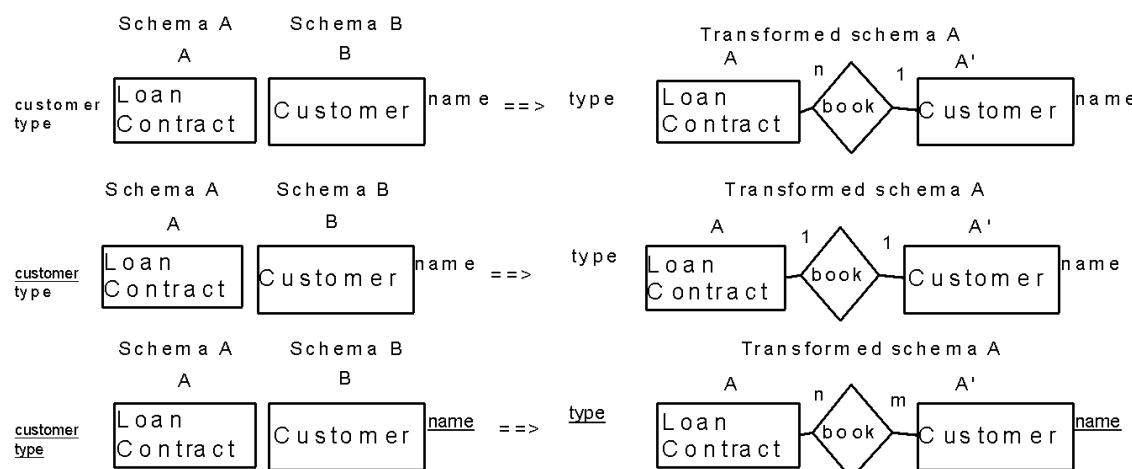
Trường hợp 2: *Giải quyết xung đột kiểu dữ liệu*



Lược đồ A có thuộc tính A_x có chung kiểu dữ liệu với một thuộc tính trong lược đồ B. Khi tích hợp hai lược đồ này ta sinh thêm một quan hệ giữa hai lược đồ A và B bởi chúng có thuộc tính chung kiểu dữ liệu và bản chất giống nhau. Ta xét ba trường hợp sau:

- Nếu thuộc tính A_x không phải là khóa thì khi tích hợp A và B thêm một quan hệ giữa hai lược đồ này: B thành A' ; A' và A có quan hệ 1-nhiều.
- Nếu thuộc tính A_x là khóa chính thì khi tích hợp A và B thêm một quan hệ giữa hai lược đồ này: B thành A' ; A' và A có quan hệ 1-1.
- Nếu thuộc tính A_x và A_y là các thuộc tính tạo khóa chính thì khi tích hợp A và B thêm một quan hệ giữa hai lược đồ này: B thành A' ; A' và A có quan hệ nhiều-nhiều và khóa chính của A là A_y

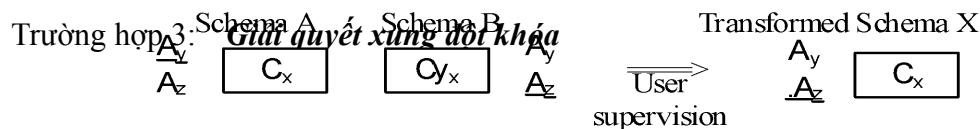
Ví dụ:



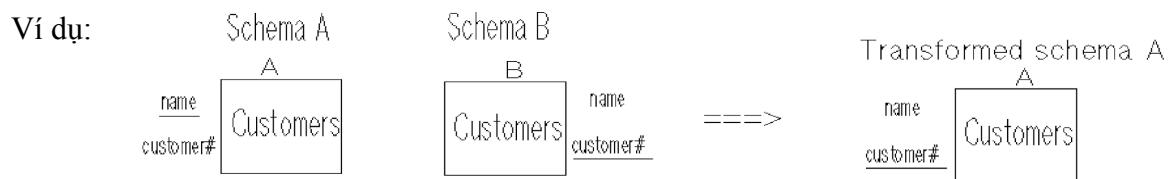
Lược đồ A có hai thuộc tính là customer và type, lược đồ B có thuộc tính name trong đó name và customer có chung kiểu dữ liệu. Khi tích hợp hai lược đồ này ta sinh thêm một quan hệ giữa hai

lược đồ A và B bởi chúng có thuộc tính chung kiểu dữ liệu và bản chất giống nhau. Ta xét ba trường hợp sau:

- Nếu thuộc tính customer không phải là khóa thì khi tích hợp A và B thêm một quan hệ **book** giữa hai lược đồ này: B thành A'; A' và A có quan hệ 1-nhiều.
- Nếu thuộc tính customer là khóa chính thì khi tích hợp A và B thêm một quan hệ **book** giữa hai lược đồ này: B thành A'; A' và A có quan hệ 1-1.
- Nếu thuộc tính customer là thuộc tính tạo khóa chính thì khi tích hợp A và B thêm một quan hệ **book** giữa hai lược đồ này: B thành A' ; A' và A có quan hệ nhiều-nhiều và khóa của A là **type** và khóa của A' là **name**

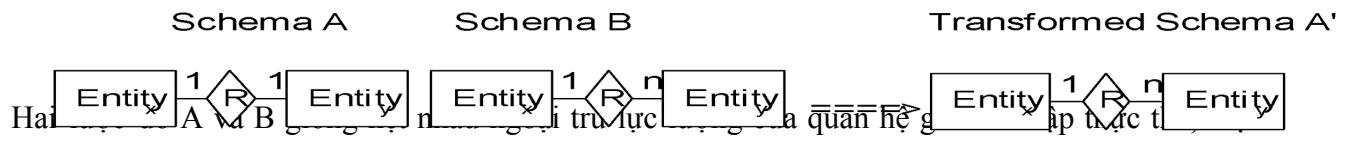


Hai lược đồ A và B có cùng một số thuộc tính là A_y và A_z trong đó khóa của A là A_y còn khóa của B là A_z . Bản chất của hai tập thực thể A và B là như nhau nhưng khóa khác nhau nên khi tích hợp chúng lại thành một tập thực thể X, chúng ta cần giải quyết xung đột về khóa bằng cách chọn một trong hai khóa của A và B làm khóa cho X. Trong hình vẽ trên ta chọn A_z để làm khóa cho X.



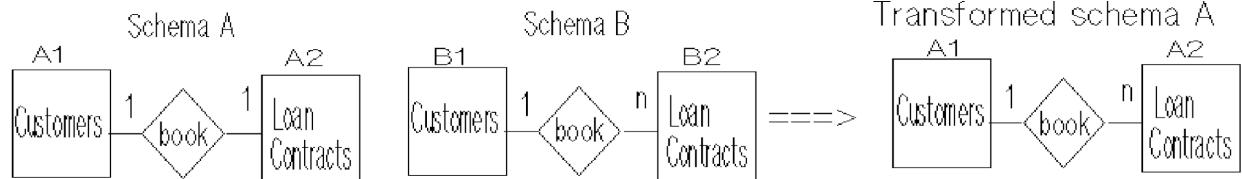
Hai lược đồ A và B có cùng một số thuộc tính là **name** và **customer#** trong đó khóa của A là **name** còn khóa của B là **customer#**. Bản chất của hai tập thực thể A và B là như nhau nhưng khóa khác nhau nên khi tích hợp chúng lại thành một tập thực thể X, chúng ta cần giải quyết xung đột về khóa bằng cách chọn một trong hai khóa của A và B làm khóa cho X. Trong hình vẽ trên ta chọn **customer#** để làm khóa cho X.

Trường hợp 4: Giải quyết xung đột về lực lượng



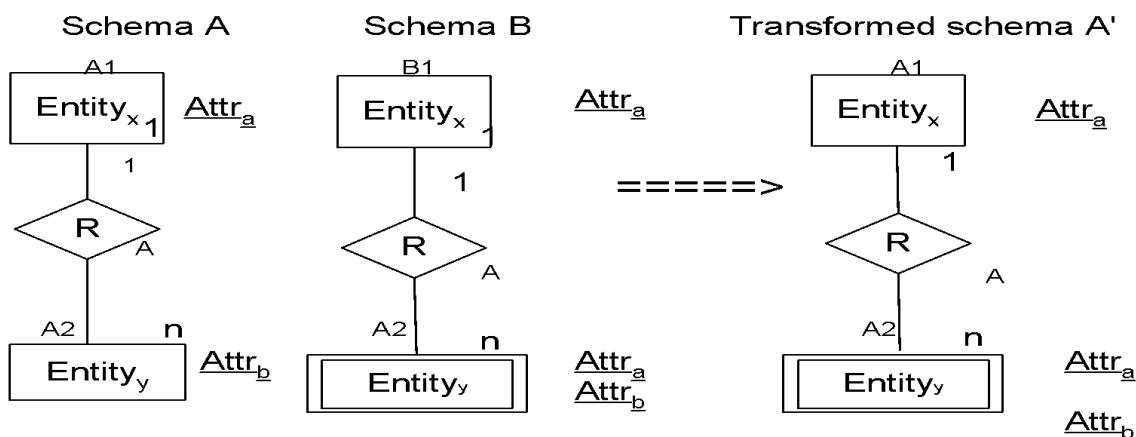
lực đồ có quan hệ 1-1, còn quan hệ kia có quan hệ 1-nhiều. Khi tích hợp hai lực đồ này chúng ta chuyển thành một lực đồ có hai tập thực thể đó và một quan hệ một-nhiều giữa hai tập thực thể.

Ví dụ:

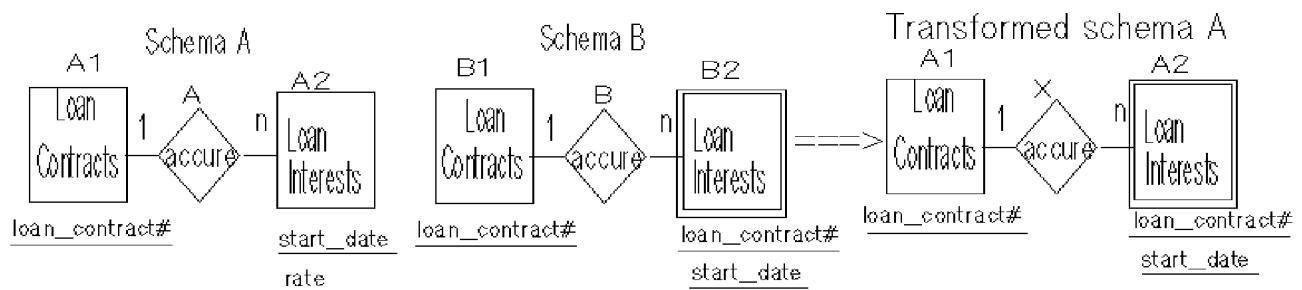


Trường hợp 5: Giải quyết xung đột của thực thể yếu

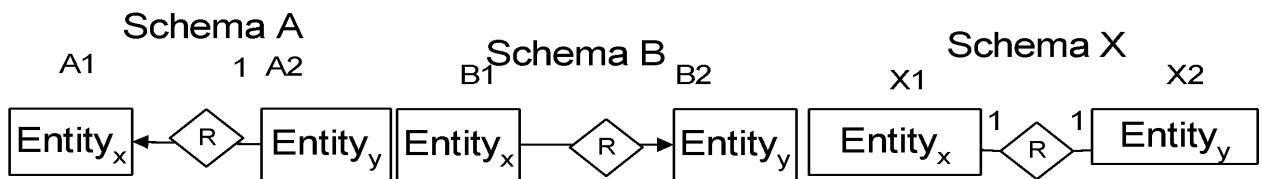
Hai lực đồ A và B giống nhau, chỉ khác là tập thực thể Y là thực thể yếu trong lực đồ B, nhưng trong lực đồ A là tập thực thể bình thường (xem hình vẽ minh họa dưới đây). Khi tích hợp hai lực đồ này chúng ta có một lực đồ giống hệt lực đồ B, có nghĩa là tập thực thể Y được chuyển sang thành tập thực thể yếu sau khi tích hợp.



Ví dụ:

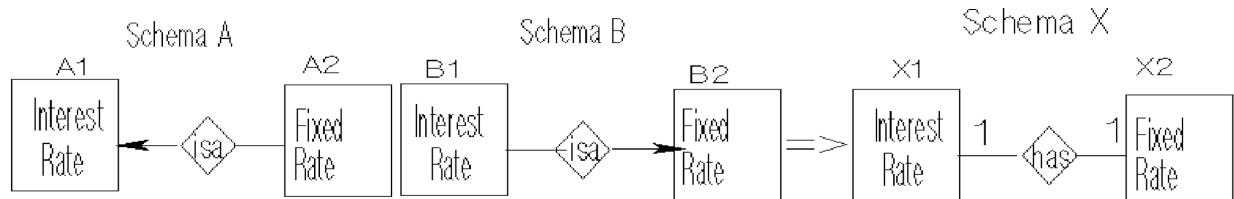


Trường hợp 6: Giải quyết xung đột trên thực thể kiểu con



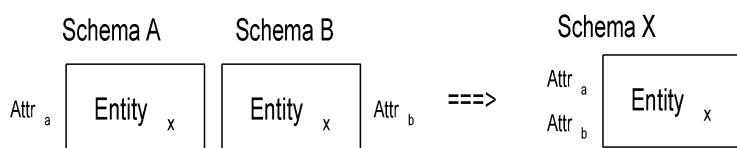
Cho hai tập thực thể X và Y có ràng buộc như trên hình vẽ: X là con của Y, Y là con của X, khi tích hợp lại ta có X và Y có quan hệ 1-1.

Ví dụ:



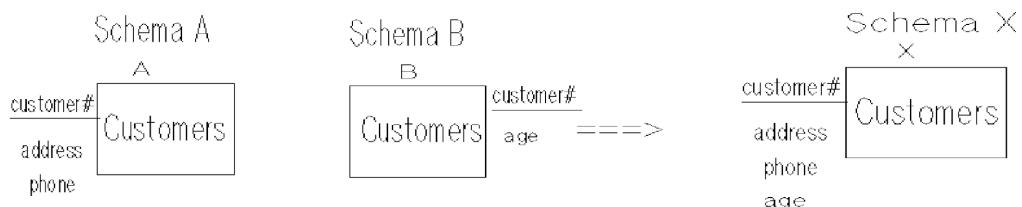
Bước 2: Trộn các thực thể

Trường hợp 1: Trộn các tập thực thể bằng phép hợp

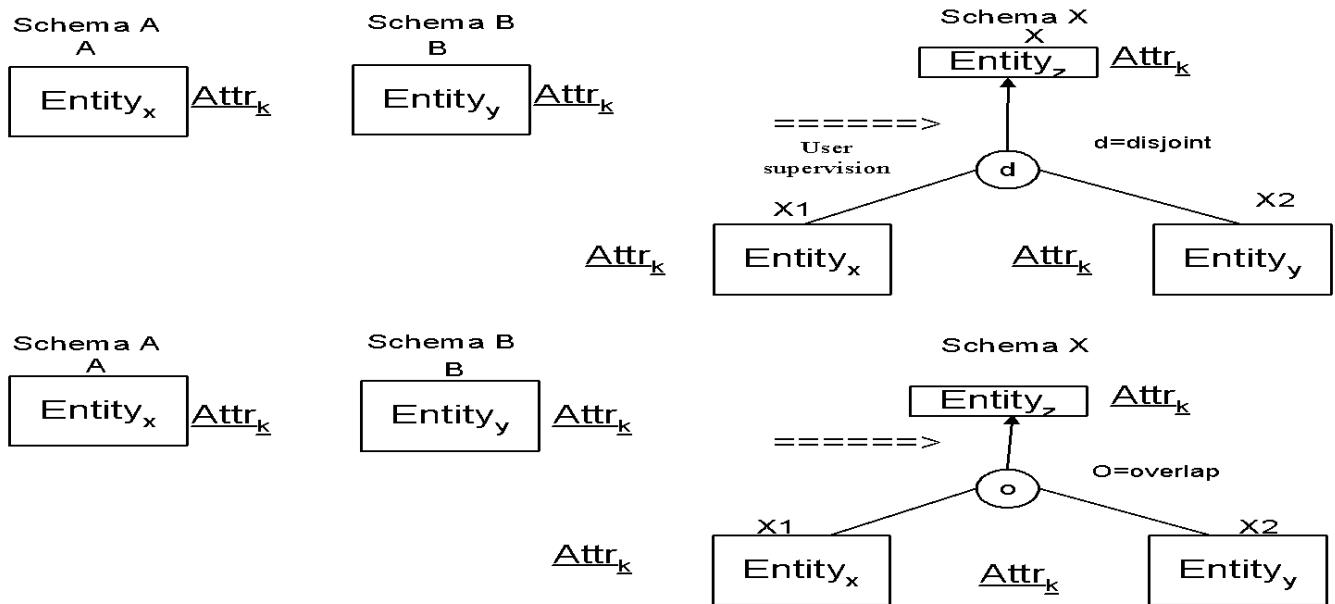


Nếu hai lược đồ A và B có hai tập thực thể X và Y có thuộc tính a và b khác nhau thì sau khi tích hợp lại chúng ta có lược đồ X với tập thực thể X có hai thuộc tính a và b (chính là hợp các thuộc tính của hai tập thực thể A và B).

Ví dụ: lược đồ X có tập thực thể Customers chứa các hợp của các thuộc tính trong lược đồ A và B (như hình vẽ dưới đây)

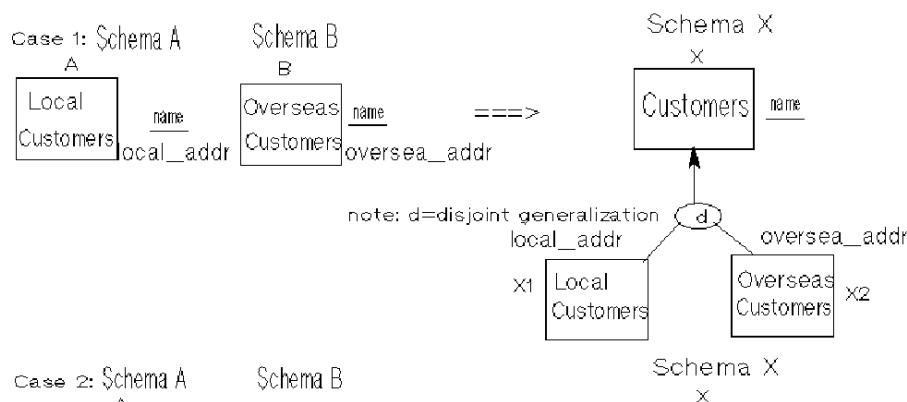


Trường hợp 2: Trộn EER bằng cách khái quát hóa

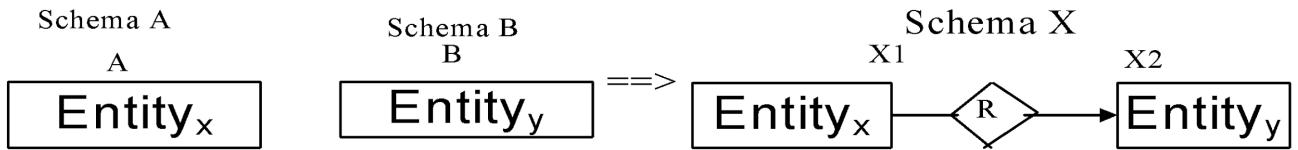


Hai lược đồ A và B có hai tập thực thể X và Y có cùng khóa là thuộc tính k. Nếu hai tập thực thể không có phần tử nào giao nhau thì khi tích hợp lại sẽ được một lược đồ khái quát không giao nhau như hình đầu tiên trong hình vẽ trên. Nếu hai tập thực thể có giao nhau thì kết quả sẽ được hình thứ hai. Trong cả hai trường hợp giao và không giao nhau, tập thực thể Z đều có khóa là thuộc tính k.

Ví dụ được thể hiện trong hình vẽ dưới đây. Trong đó hai loại thực thể khách hàng là khách hàng địa phương và khách hàng nước ngoài được khái quát hóa thành tập thực thể khách hàng nói chung với hai trường hợp tập thực thể khách hàng nước ngoài và địa phương là có giao nhau (case 2) hoặc không giao nhau (case 1).



Trường hợp 3: Trộn EER bằng quan hệ loại con

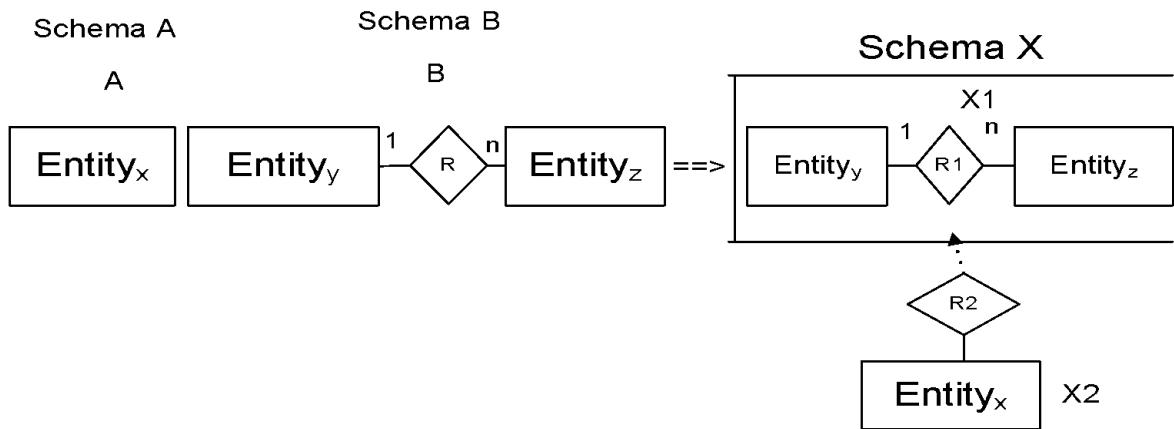


Hai lược đồ A và B có hai tập thực thể X và Y và chung nhau khóa K. Nếu hai tập thực thể này có quan hệ cha con thì sau khi tích hợp ta thu được một lược đồ với hai tập thực thể và một quan hệ cha con (subtype) giữa chúng như hình vẽ trên.

Ví dụ được thể hiện trong hình vẽ dưới đây: hai tập thực thể hiện tỉ lệ lãi suất được tích hợp lại thành lược đồ X có quan hệ là-một (cha con) giữa hai tập thực thể này.

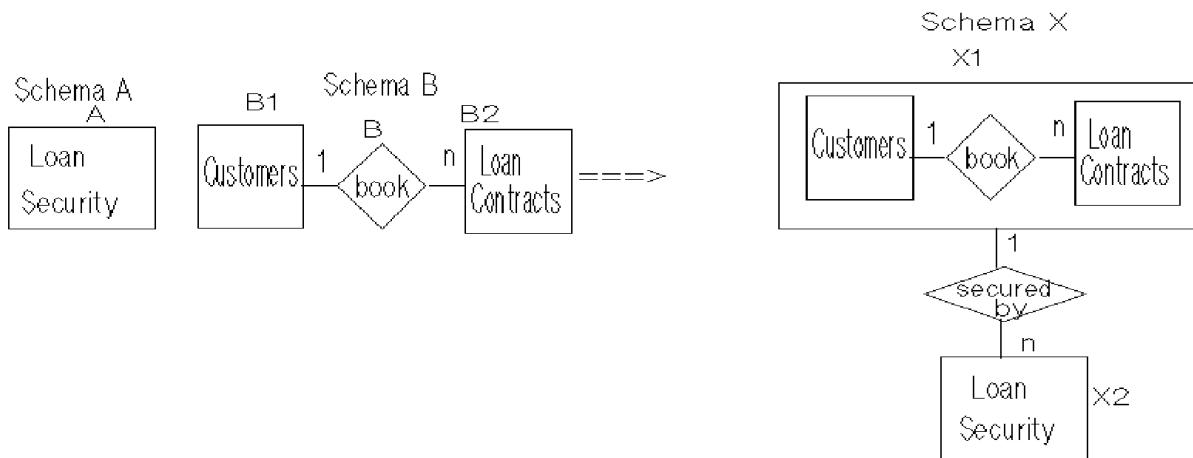


Trường hợp 4: Trộn EER bằng cách dùng thực thể tích hợp (Aggregation)

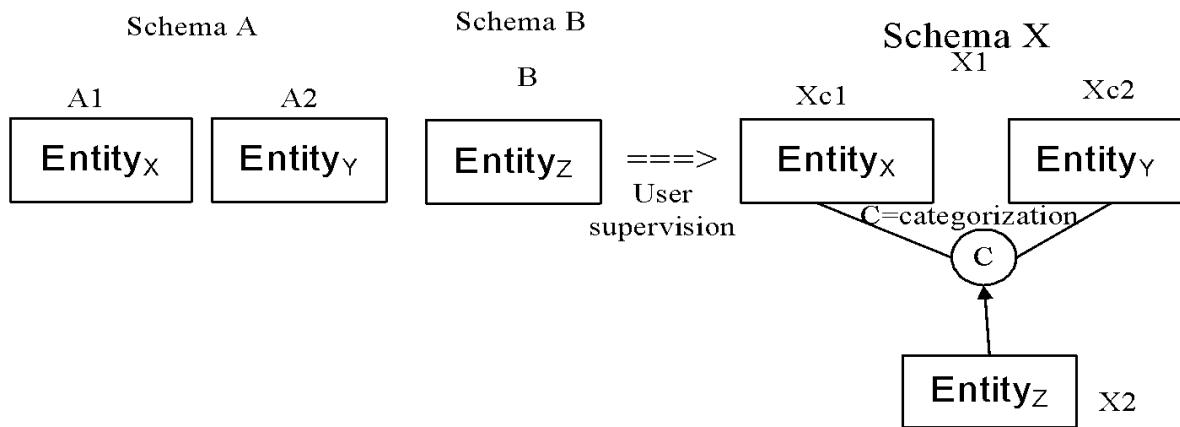


Hai tập thực thể Y và Z của lược đồ B đều có quan hệ R2 với tập thực thể X của lược đồ A, do đó sau khi tích hợp hai lược đồ thành lược đồ X, ta thiết lập một thực thể tích hợp X1 chứa lược đồ B có quan hệ R2 với tập thực thể X (giờ được đổi tên là X2) của lược đồ A.

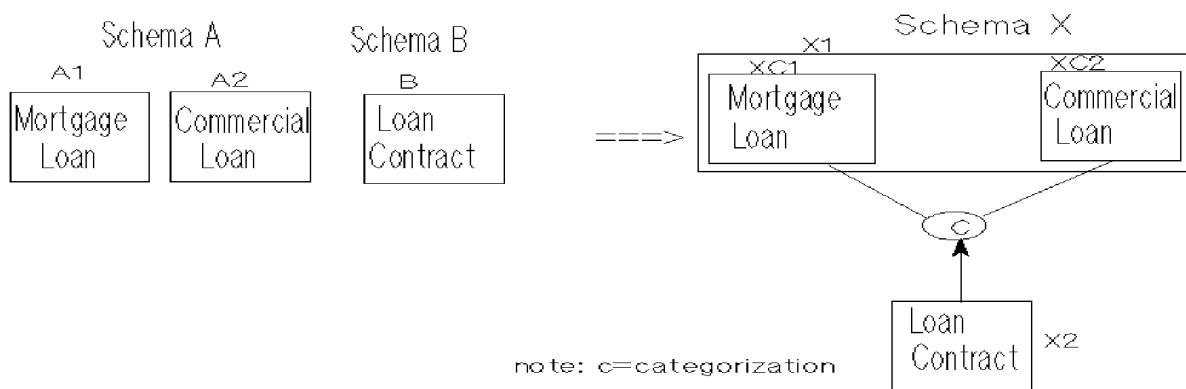
Ví dụ xem hình vẽ dưới đây trong đó cả hai tập thực thể Khách hàng và Giao dịch vay nợ đều được đảm bảo bằng các phương thức an toàn vay nợ. Chính vì thế khi tích hợp ta tạo một lược đồ X1 là thực thể tích hợp gồm hai loại thực thể và quan hệ của lược đồ B, X1 có quan hệ 1-nhiều với tập thực thể X2 là các phương thức an toàn vay nợ.



Trường hợp 5: Trộn EER bằng cách dùng phân loại

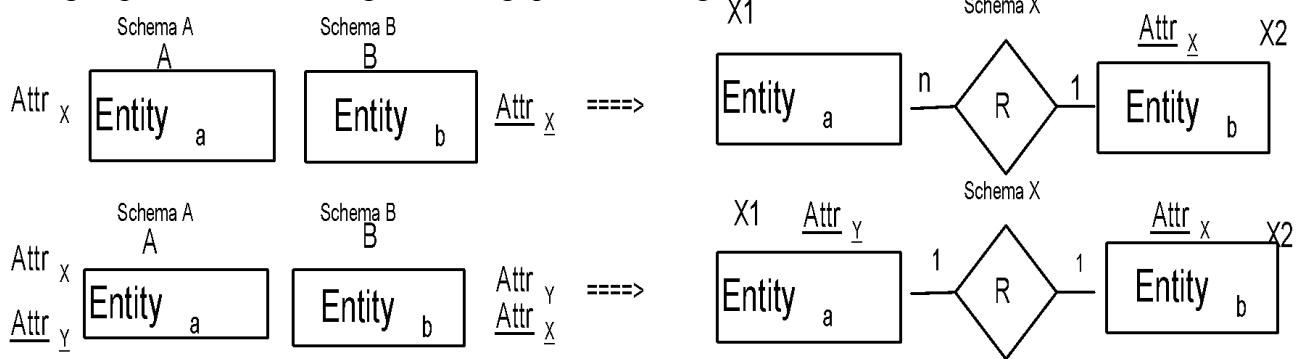


Lược đồ A bao gồm hai tập thực thể X và Y, lược đồ B bao gồm tập thực thể Z. Nếu ta nhận thấy tập thực thể Z bao gồm hai loại thực thể X và Y như hai loại đặc biệt của Z thì khi tích hợp A, B thành lược đồ X, ta có X2 được phân loại thành Xc1 và Xc2.



Ví dụ: Hợp đồng vay nợ (tập thực thể B trong lược đồ B) gồm hai loại vay nợ thương mại (tập thực thể A2) và vay nợ mua nhà (tập thực thể A1), khi chuyển đổi sẽ thành lược X trong đó tập thực thể X2 được phân loại thành Xc1 và Xc2 như trong hình vẽ trên.

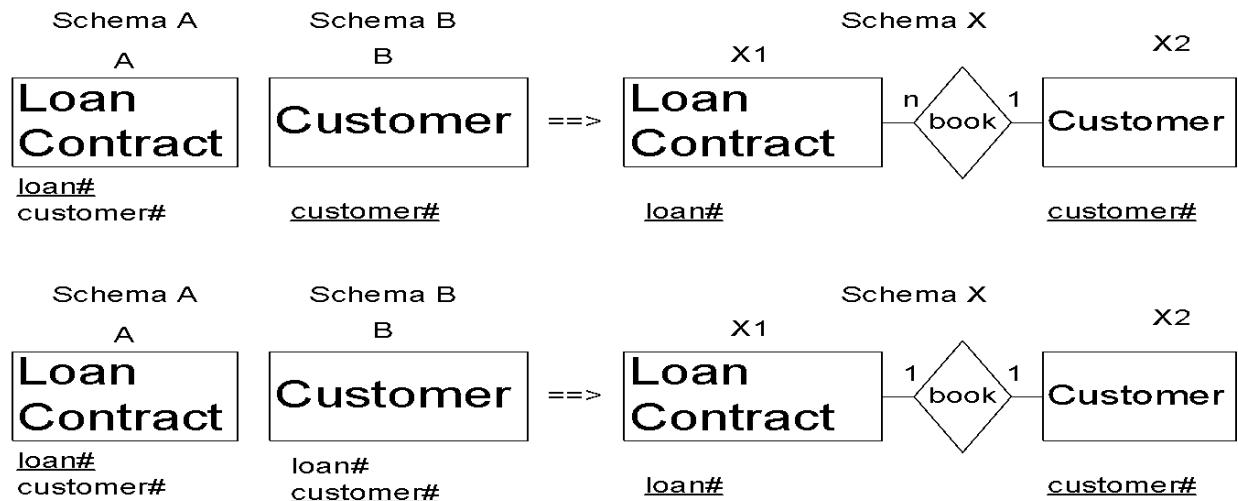
Trường hợp 6: Trộn EER bằng cách dùng quan hệ hai ngôi



Trường hợp thứ nhất lược đồ A chứa tập thực thể A với thuộc tính X không phải khóa chính, lược đồ B chứa tập thực thể B cũng có thuộc tính X là khóa chính thì khi trộn hai lược đồ đó thu được lược đồ X với hai thực thể A và B và một quan hệ một-nhiều R giữa chúng (xem hình vẽ trên).

Trường hợp thứ hai lược đồ A chứa tập thực thể A với thuộc tính X không phải khóa chính và thuộc tính Y là khóa chính, lược đồ B chứa tập thực thể B cũng có thuộc tính X là khóa chính và Y không là khóa chính thì khi trộn hai lược đồ đó thu được lược đồ X với hai thực thể A và B và một quan hệ một-một R giữa chúng (xem hình vẽ trên).

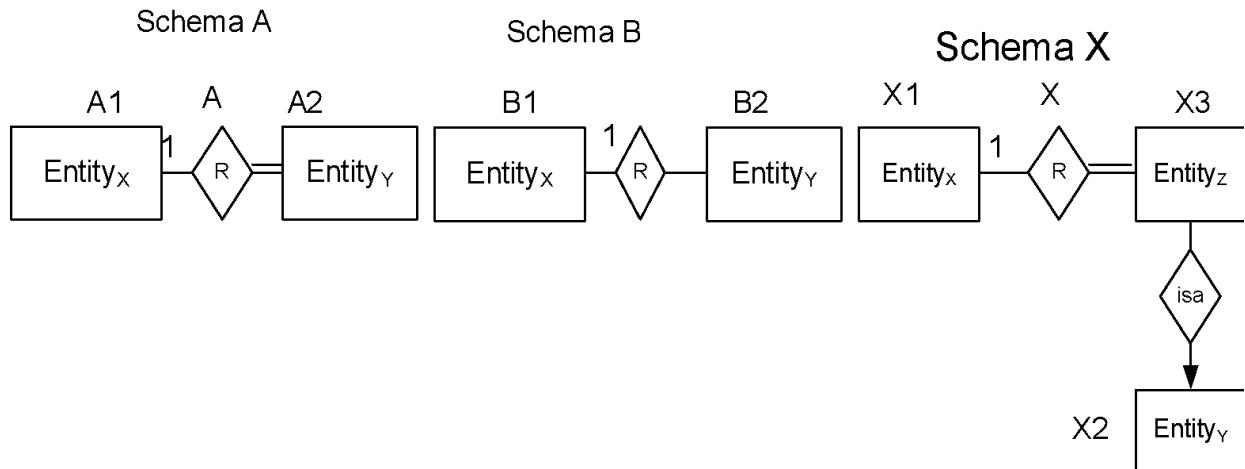
Ví dụ lược đồ A chứa tập thực thể hợp đồng vay nợ, lược đồ B chứa tập thực thể khách hàng, có các thuộc tính khóa và không khóa như hình vẽ dưới đây. Sau khi trộn hai lược đồ thu được lược đồ X có thêm quan hệ Book.



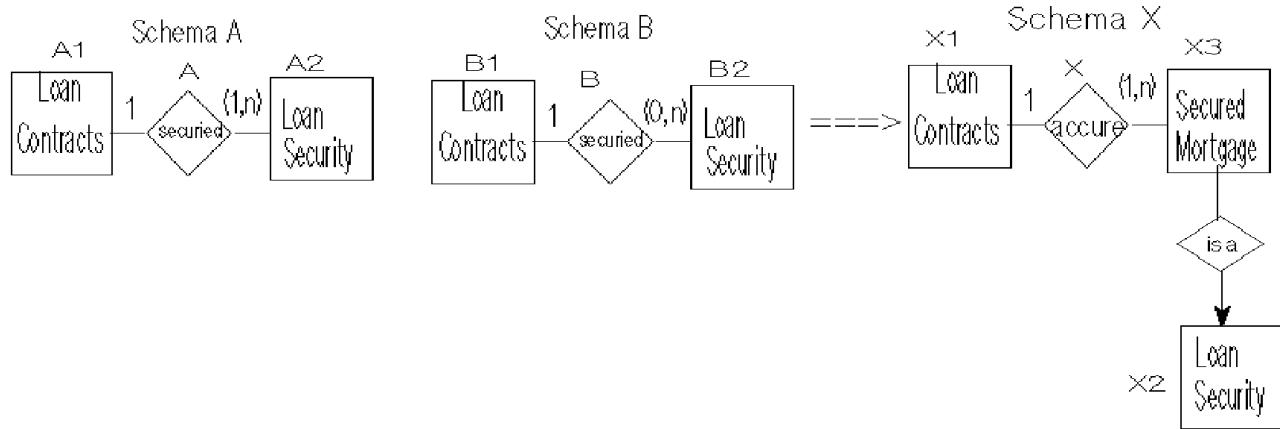
Bước 3: Trộn các quan hệ

Trường hợp 1: Trộn các quan hệ bằng quan hệ cha con

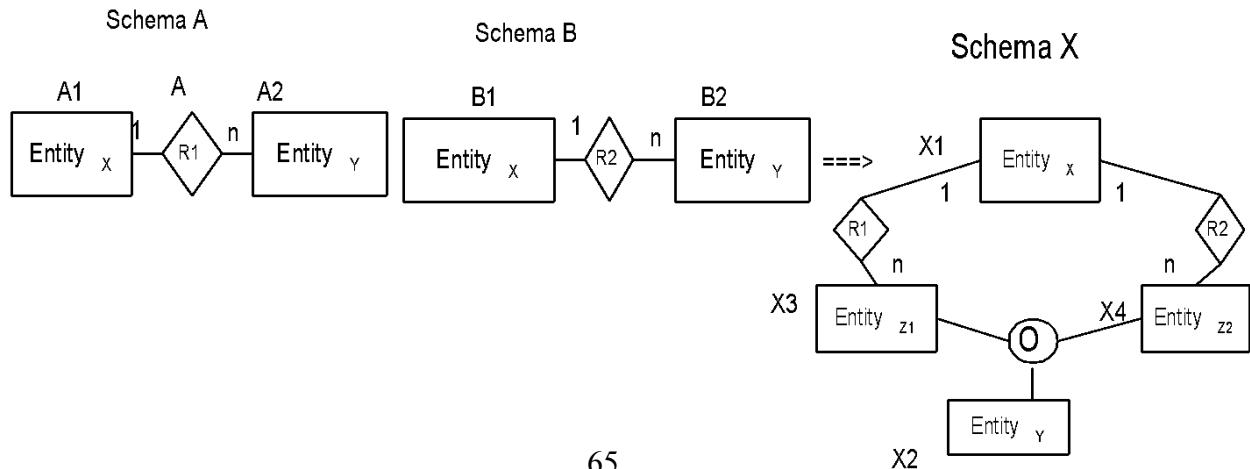
Lược đồ A và B phần lớn là giống nhau chỉ khác tập thực thể Y tham gia toàn phần vào quan hệ R trong lược đồ A, còn tham gia một phần vào quan hệ R trong lược đồ B. Khi trộn hai lược đồ, ta đưa thêm vào một tập thực thể Z có quan hệ R toàn phần với X và quan hệ là-một với Y (như hình vẽ dưới đây). Nhờ quan hệ R và quan hệ là-một, X quan hệ R với Y nhưng là một phần vì không phải tất cả các thực thể trong Y đều nằm trong Z, mà Z thì quan hệ toàn phần với X.



Ví dụ cho việc trộn trên được thể hiện trong hình vẽ dưới đây

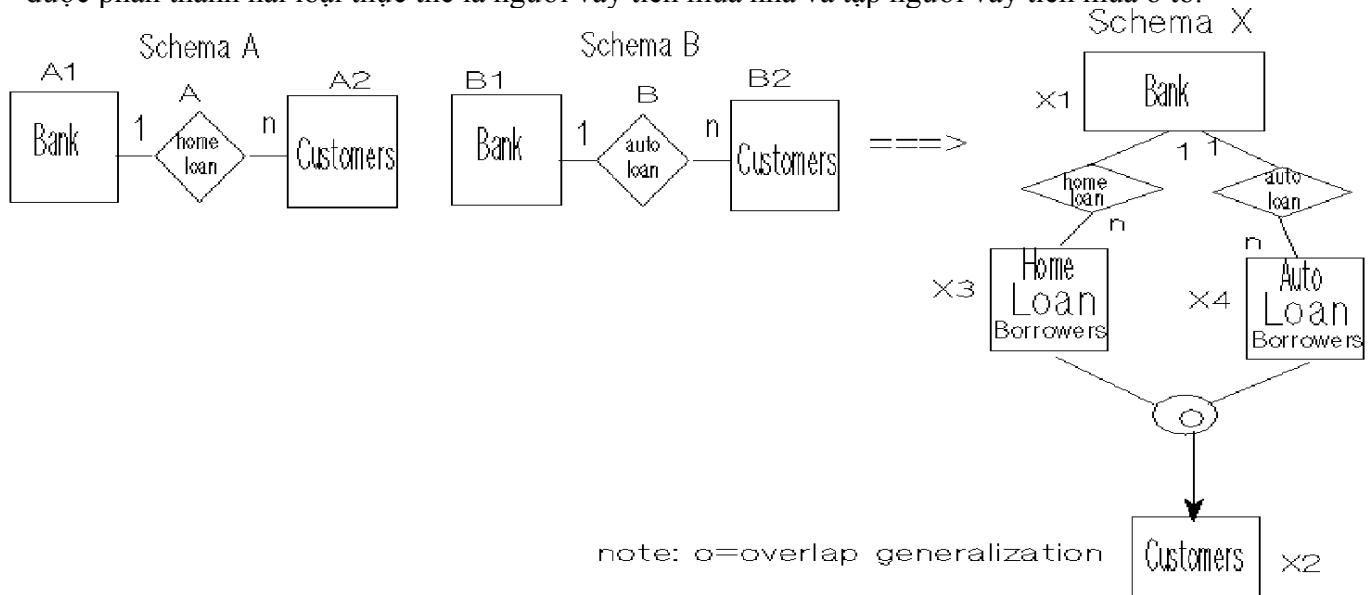


Trường hợp 2: Trộn các quan hệ bằng khái quát hóa trùng lặp



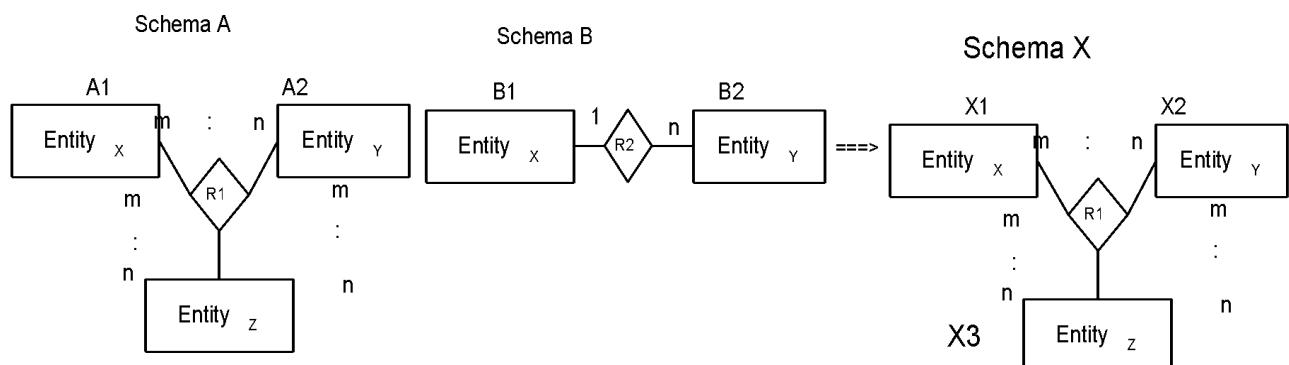
Hai lược đồ A và B có hai tập thực thể X và Y quan hệ 1-nhiều với nhau thông qua quan hệ R1 và R2. Sau khi trộn hai lược đồ, ta thu được lược đồ X chứa một khái quát hóa có thể trùng lặp: thực thể Y được phân thành hai tập thực thể Z1 và Z2, Z1 có quan hệ R1 với X và Z2 có quan hệ R2 với X, thể hiện được lược đồ A và B, đồng thời thể hiện rõ không phải tất cả thực thể thuộc Y đều tham gia R1 và R2.

Ví dụ của trường hợp trộn này được thể hiện ở hình vẽ dưới đây trong đó: thực thể Khách hàng được phân thành hai loại thực thể là người vay tiền mua nhà và tập người vay tiền mua ô tô.

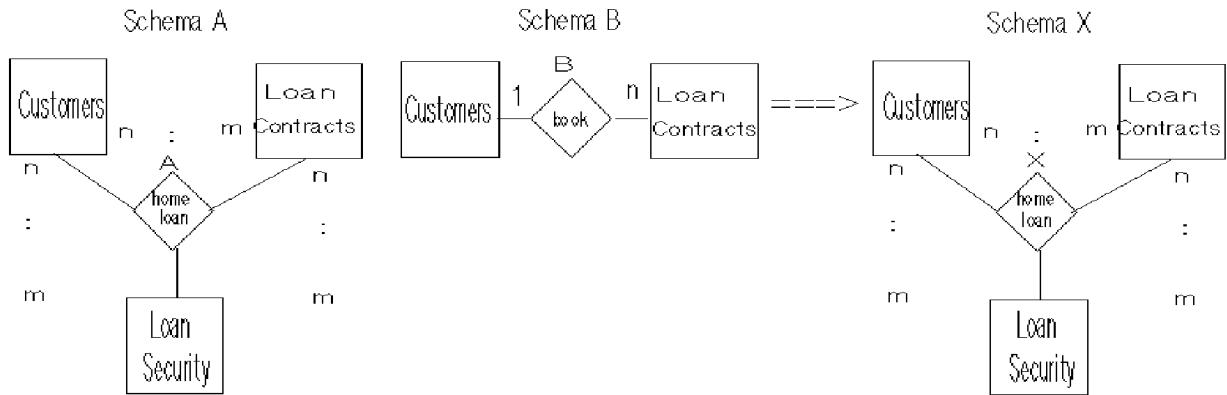


Trường hợp 3: Sát nhập các quan hệ ít ngôi thành một quan hệ có ngôi cao hơn

Tập thực thể X, Y và Z có quan hệ R1 3 ngôi dạng nhiều-nhiều trong lược đồ A, tập thực thể X và Y có quan hệ R2 một-nhiều trong lược đồ B. Khi trộn lược đồ A và B ta thu được lược đồ X 3 ngôi giữa X, Y và Z có quan hệ R1 nhiều-nhiều, lúc này ta nói R2 được sát nhập vào R1 trong lược đồ X (xem hình vẽ dưới đây)

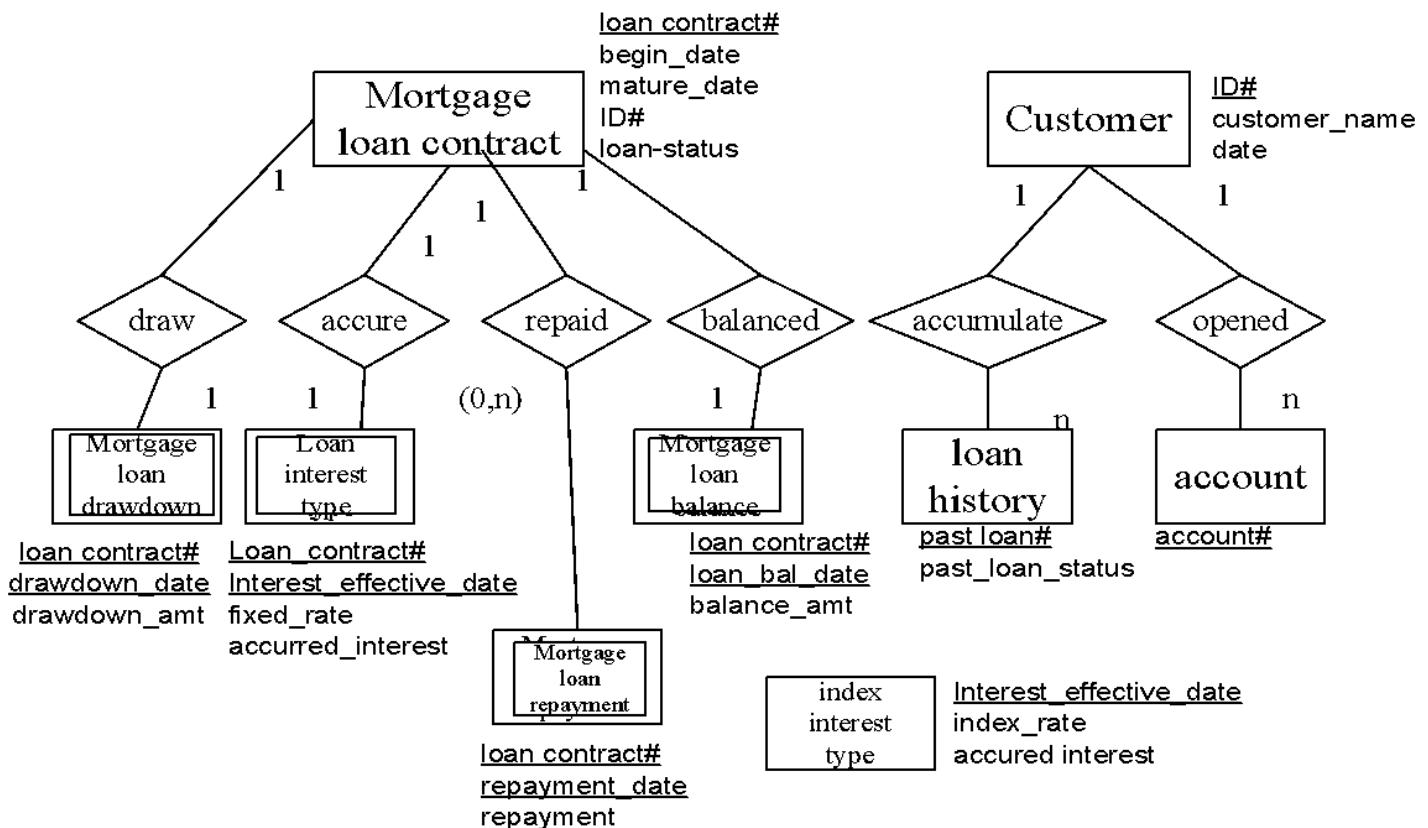


Ví dụ cho trường hợp này được thể hiện trong hình vẽ dưới đây



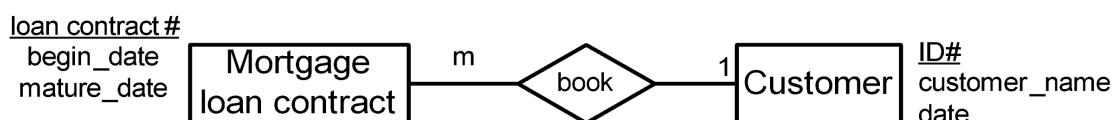
Bài thực hành

Một ngân hàng gồm có các cơ sở dữ liệu với các lược đồ tương ứng sau: một lược đồ khách hàng, một cho các hợp đồng vay nợ mua nhà, và lược đồ thứ ba cho tỉ lệ lãi suất. Chúng ta cần tích hợp chúng thành một hệ thống vay nợ ngân hàng quốc tế.

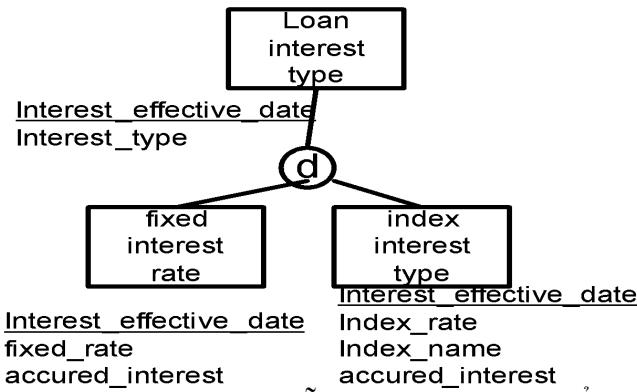


Bước 2: Trộn các tập thực thể bằng các quan hệ hai ngôi và khái quát hóa

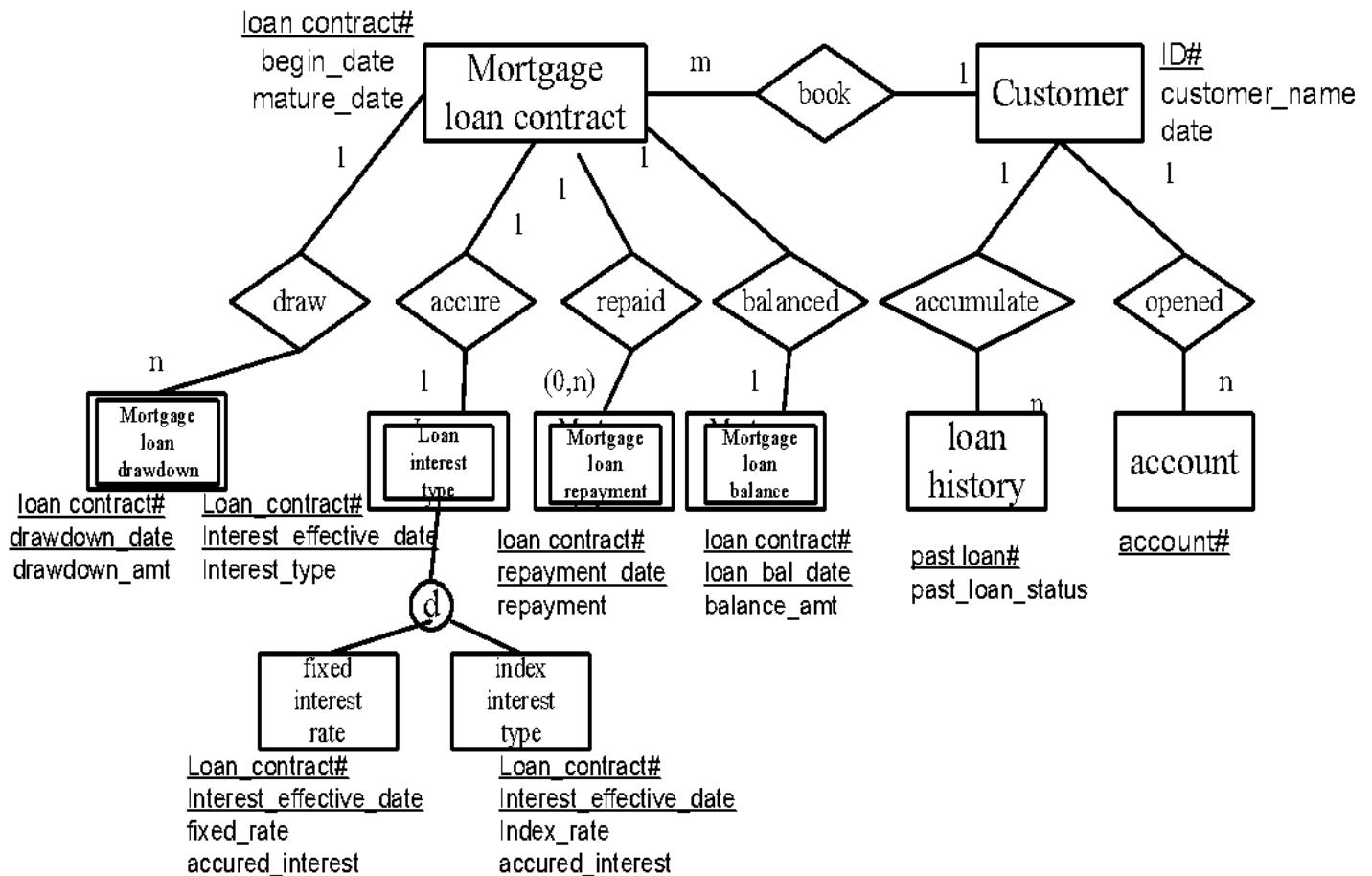
Vì dataID# xuất hiện trong thực thể Mortgage Loan Contract và thực thể Customer, chúng được thể hiện trong hình vẽ sau



Vì Fixed và Index Interest Rate có cùng khóa nên chúng có thể được khai quát hóa không giao nhau thể hiện trong hình vẽ dưới đây

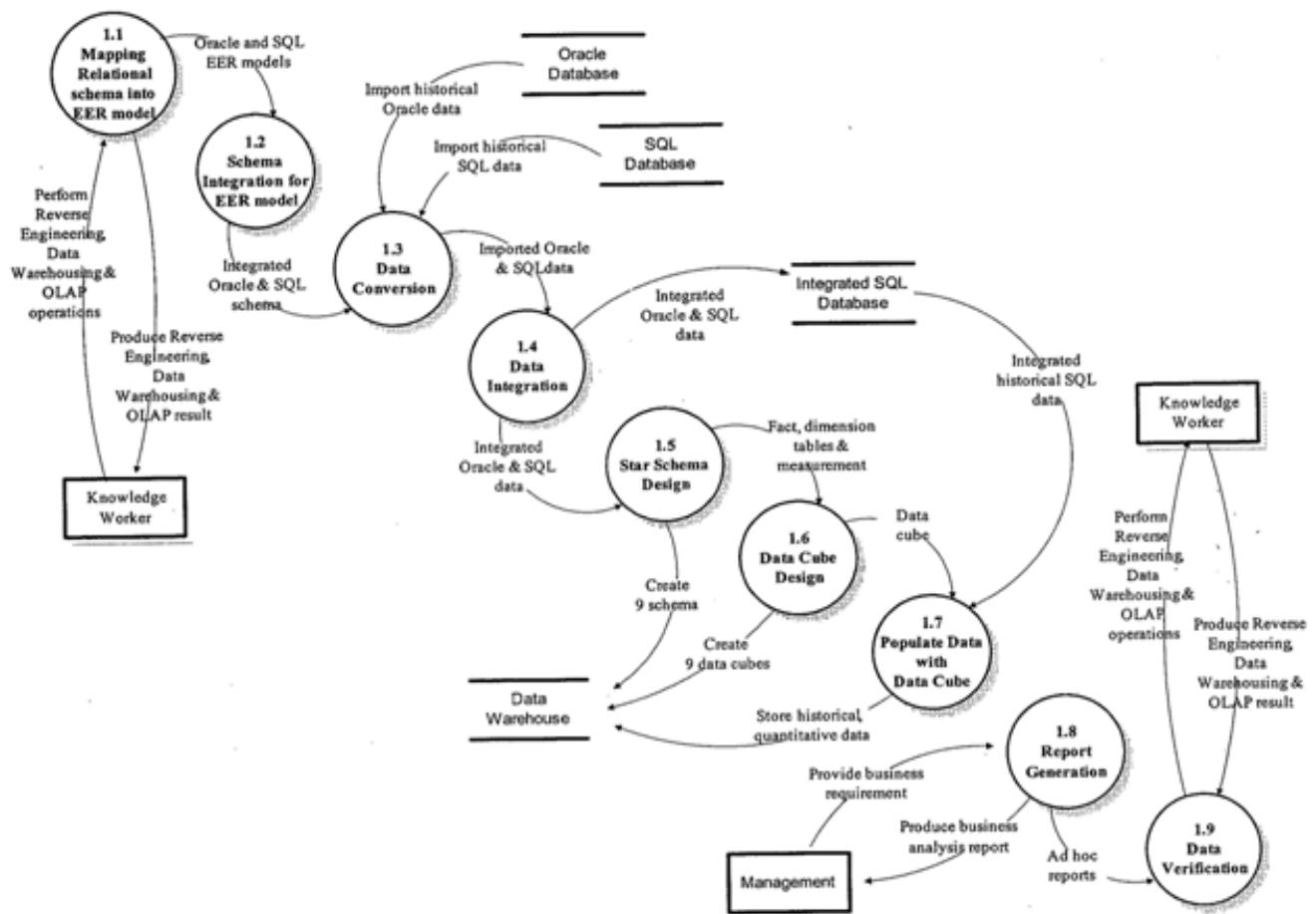


Vì vậy, chúng ta có thể xác định lực lượng cho các quan hệ suy diễn ra giữa các thực thể và tích hợp hai lược đồ này thành một mô hình EER như hình vẽ dưới đây



2.4 Chuyển đổi và tích hợp dữ liệu

Phương pháp luận cho công nghệ kho dữ liệu và OLAP



Hình vẽ trên mô tả các bước cần thiết trong công nghệ kho dữ liệu và xử lý trực tuyến OLAP, trong đó gồm 9 bước cơ bản và những tác nhân bên ngoài và cơ sở dữ liệu cần thiết cho mỗi bước cũng như các kết quả trung gian được chú thích trên đó. Bước 1 của quá trình này được thể hiện trong phần 2.2 thực hiện việc ánh xạ lược đồ quan hệ sang mô hình thực thể liên kết mở rộng, bước 2 được trình bày trong phần 2.3 thực hiện việc tích hợp các mô hình thực thể liên kết mở rộng thành một lược đồ thống nhất. Bước thứ 3 là chuyển đổi dữ liệu được trình bày trong phần 2.4 này rồi đến việc tích hợp dữ liệu cũng được trình bày trong phần 2.5. Bước 5 đến 9 sẽ được trình bày lần lượt ở các phần tiếp theo.

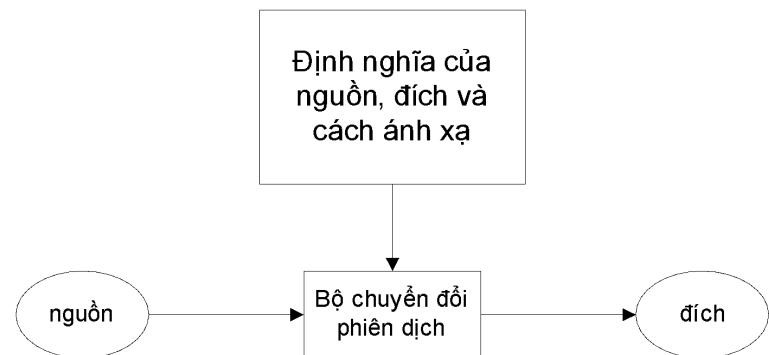
Các cách chuyển đổi dữ liệu

Chuyển đổi dữ liệu bằng cách dùng chương trình

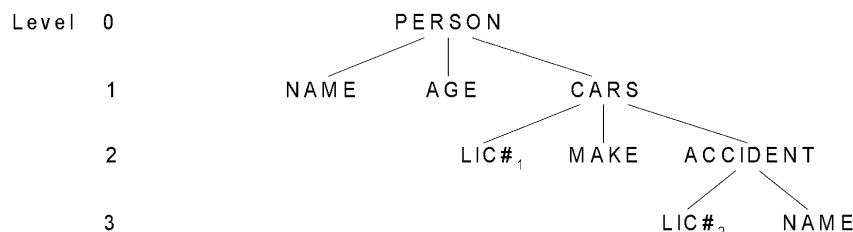
- Một cách tiếp cận chung cho việc chuyển đổi dữ liệu là xây dựng các chương trình tùy biến để chuyển đổi dữ liệu từ một môi trường này sang một môi trường khác. Tuy nhiên, cách tiếp cận này là cực kỳ tốn kém bởi vì nó đòi hỏi một chương trình riêng biệt được viết cho từng M tập nguồn và N tập đích. Hơn nữa, các chương trình này chỉ được sử dụng một lần. Kết quả là, việc hoàn toàn phụ thuộc vào chương trình tùy biến cho dữ liệu chuyển đổi là không thể quản lý, quá tốn kém và mất thời gian.

Chuyển đổi dữ liệu bằng bộ chuyển đổi phiên dịch

Cách thức được thể hiện trong hình vẽ dưới đây



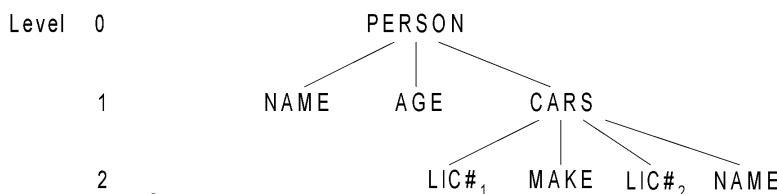
Ví dụ cấu trúc Cobol sau có thể được thể hiện bằng các sử dụng các báo cáo



Cấu trúc dữ liệu
Thể hiện
Bộ N
Mối Quan hệ
Các bộ N

PERSON <NAME, AGE>
CARS <LIC#1, MAKE>
ACCIDENTS <LIC#2, NAME>
PERSON-CAR <NAME, LIC#1>
CAR-ACCIDENT <LIC#1, LIC#2>

Để dịch 3 mức trên thành cấu trúc dữ liệu hai mức như hình vẽ



Câu lệnh TDL cần như sau

FORM NAME FROM NAME

FORM LIC#1 FROM LIC#1

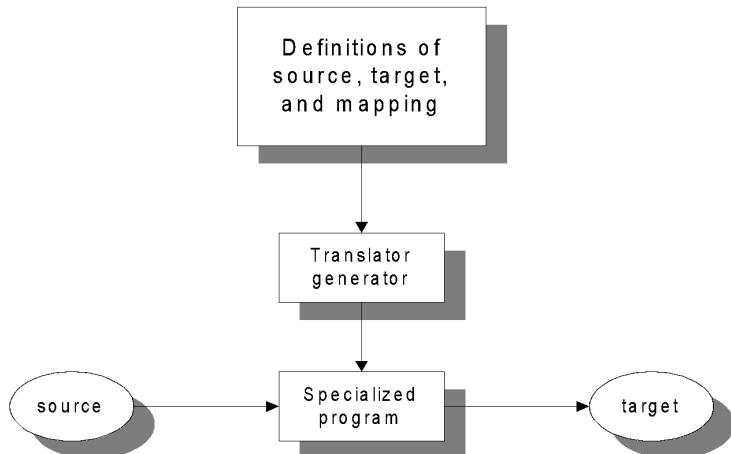
:

FORM PERSON IF PERSON

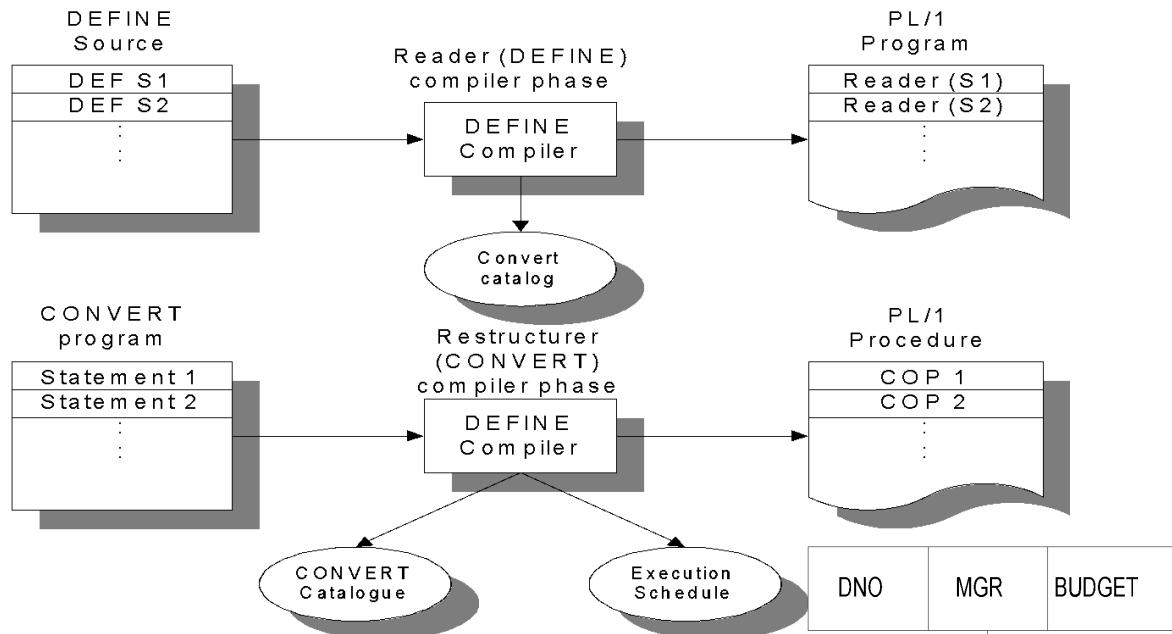
FORM CARS IF CAR AND ACCIDENT

Chuyển đổi dữ liệu bằng bộ chuyển đổi biên dịch

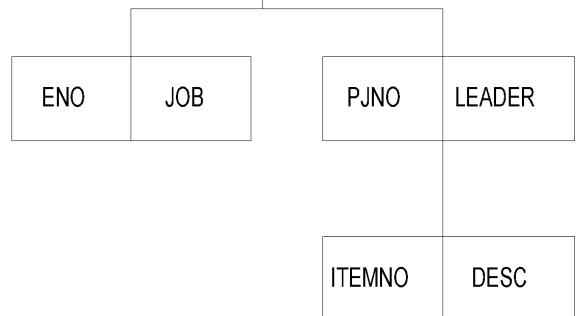
Cách thức gồm các bước được thực hiện như hình vẽ dưới đây



DEFINE và CONVERT trong pha biên dịch được thể hiện trong hình vẽ dưới đây



Xét một cơ sở dữ liệu phân cấp sau để làm ví dụ



Câu lệnh DEFINE có thể được mô tả như dưới đây trong đó với mỗi câu lệnh DEFINE, một mã nguồn được tạo ra để dành vị trí trong bộ nhớ đệm bên trong cho một cây con mới.

GROUP DEPT:

```
OCCURS FROM 1 TIMES;  
FOLLOWED BY EOF;  
PRECEDED BY HEX '01';  
:  
END EMP;  
GROUP PROJ:  
OCCURS FROM 0 TIMES;  
PRECEDED BY HEX '03';  
:  
END PROJ;
```

END DEPT;

Cho mỗi câu lệnh CONVERT do người sử dụng viết, chúng ta có thể sinh ra một chương trình tùy biến. Lấy DEPT FORM từ câu lệnh DEFINE ở trên ta có

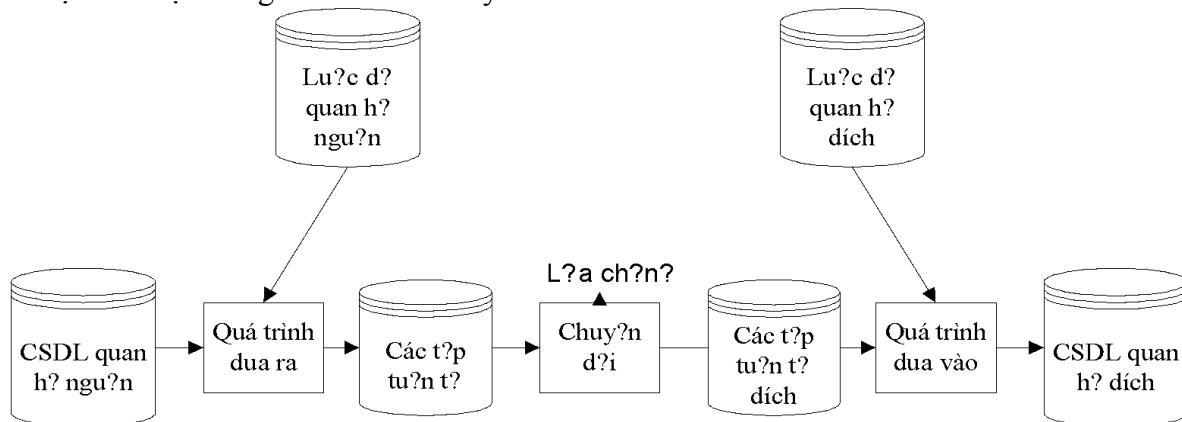
```
T1 = SELECT (FROM DEPT WHERE BUDGET GT '100'); sẽ sinh ra chương trình sau đây  
/* PROCESS LOOP FOR T1 */
```

```
DO WHILE (not end of file);  
    CALL GET (DEPT);  
    IF BUDGET > '100'  
        THEN CALL BUFFER_SWAP (T1, DEPT);
```

END

Cách chuyển đổi dữ liệu dạng biên dịch ở mức logic

Biểu đồ luồng hệ thống cho việc chuyển đổi dữ liệu từ CSDL quan hệ nguồn tới CSDL quan hệ đích được thể hiện trong hình vẽ dưới đây



Một ví dụ về việc chuyển đổi

Dưới đây là một ví dụ về chuyển đổi dữ liệu của một cơ sở dữ liệu quan hệ từ DB2 sang Oracle

Yêu cầu nghiệp vụ: Một công ty có 2 khu vực hoạt động A và B

- Mỗi khu vực có văn phòng riêng
- Mỗi văn phòng duyệt một số chuyến đi trong một năm
- Mỗi nhân viên đi nhiều chuyến công tác trong 1 năm
- Trong mỗi chuyến công tác, mỗi nhân viên cần thuê xe cho việc di chuyển
- Mỗi chiếc xe được thuê đều có thể chở được nhiều nhân viên
- Một nhân viên có thể là một người quản lý hoặc một kỹ sư

Yêu cầu về dữ liệu bao gồm các quan hệ và các ràng buộc sau đây

Relation Department(*Department_id, Salary)

Relation Region_A (Department_id, Classification)

Relation Region_B (Department_id, Classification)

Relation Trip (Trip_id, *Car_model, *Staff_id, *Department_id)

Relation People (*Staff_id, Name, DOB)

Relation Car (Car_model, Size, Description)

Relation Assignment(*Car_model, *Staff_id)

Relation Engineer (*Staff_id, Title)

Relation Manager (*Staff_id, Title)

ID: Department.Department_id \subseteq (Region_A.Department_id \cup Region_B.Department_id)

ID: Trip.Department_id \subseteq Department.Department_id

ID: Trip.Car_model \subseteq Assignment.Car_model

ID: Trip.Staff_id \subseteq Assignment.Staff_id

ID: Assignment.Car_model \subseteq Car.Car_model

ID: Assignment.Staff_id \subseteq People.Staff_id

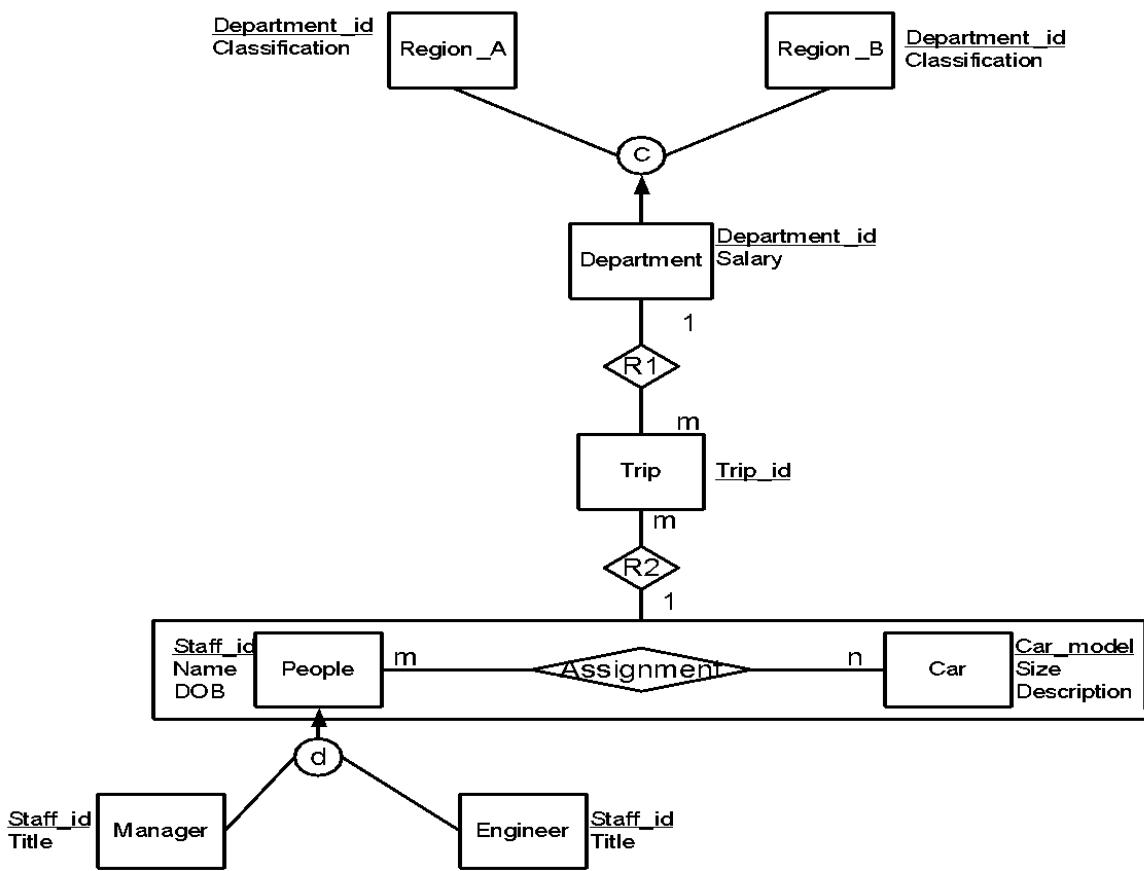
ID: Engineer.Staff_id \subseteq People.Staff_id

ID: Manager.Staff_id \subseteq People.Staff_id

Với ID = Inclusion Dependence hay sự phụ thuộc bao gồm, Các thuộc tính được gạch chân là khóa chính và dấu “*” ở đầu thuộc tính thể hiện là các khóa ngoại.

Mô hình thực thể liên kết mở rộng cho dữ liệu TRIP

Được thể hiện trong lược đồ dưới đây



Lược đồ quan hệ cho cơ sở dữ liệu TRIP ở trên

Tạo bảng Car

(CAR_MODEL character (10),
SIZE character (10),
DESCRIPT character (20),
STAFF_ID character (5),
primary key (CAR_MODEL))

Tạo bảng Depart

(DEP_ID character (5),
SALARY numeric (8),
primary key (DEP_ID))

Tạo bảng People

(STAFF_ID character (4),
NAME character (20),
DOB datetime,
primary key (STAFF_ID))

Tạo bảng Reg_A

(DEP_ID character (5),
DESCRIP character (20),
primary key (DEP_ID))

Tạo bảng trip

(TRIP_ID character (5),
primary key (TRIP_ID))

Tạo bảng Engineer

(TITLE character (20),
STAFF_ID character (4),
Foreign Key (STAFF_ID)
REFERENCES People(STAFF_ID),
Primary key (STAFF_ID))

Tạo bảng Manager

(TITLE character (20),
STAFF_ID character (4),
Foreign Key (STAFF_ID)
REFERENCES People(STAFF_ID),
Primary key (STAFF_ID))

Tạo bảng Assign

(CAR_MODEL character (10),
Foreign Key (CAR_MODEL)
REFERENCES Car(CAR_MODEL),

Tạo bảng Reg_B
`(DEP_ID character (5),
 DESCRIPT character (20),
 primary key (DEP_ID))`

`STAFF_ID character (4),
 Foreign Key (STAFF_ID)
 REFERENCES People(STAFF_ID),
 primary key
 (CAR_MODEL,STAFF_ID))`

`ALTER TABLE trip ADD CAR_MODEL character (10) null`

`ALTER TABLE trip ADD STAFF_ID character (4) null`

`ALTER TABLE trip ADD Foreign Key (CAR_MODEL,STAFF_ID) REFERENCES
 Assign(CAR_MODEL,STAFF_ID)`

Chèn dữ liệu vào cơ sở dữ liệu TRIP

```
INSERT INTO trip_ora.CAR (CAR_MODEL,CAR_SIZE,DESCRIPT,STAFF_ID) VALUES ('DA-02      ', '165  

', 'Long car      ','A001 ')  

INSERT INTO trip_ora.CAR (CAR_MODEL,CAR_SIZE,DESCRIPT,STAFF_ID) VALUES ('MZ-18      ', '120  

', 'Small sportics  ','B004 ')  

INSERT INTO trip_ora.CAR (CAR_MODEL,CAR_SIZE,DESCRIPT,STAFF_ID) VALUES ('R-023     ', '150      ',  

'Long car      ','A002 ')  

INSERT INTO trip_ora.CAR (CAR_MODEL,CAR_SIZE,DESCRIPT,STAFF_ID) VALUES ('SA-38      ', '120  

', 'New dark blue  ','D001 ')  

INSERT INTO trip_ora.CAR (CAR_MODEL,CAR_SIZE,DESCRIPT,STAFF_ID) VALUES ('WZ-01      ', '1445  

', 'Middle Sportics ','B004 ')  

INSERT INTO trip_ora.DEPART (DEP_ID,SALARY) VALUES ('AA001', 35670)  

INSERT INTO trip_ora.DEPART (DEP_ID,SALARY) VALUES ('AB001', 30010)  

INSERT INTO trip_ora.DEPART (DEP_ID,SALARY) VALUES ('BA001', 22500)  

INSERT INTO trip_ora.DEPART (DEP_ID,SALARY) VALUES ('BB001', 21500)  

INSERT INTO trip_ora.PEOPLE (STAFF_ID,NAME,DOB) VALUES ('A001', 'Alexender      ','7-1-1962')  

INSERT INTO trip_ora.PEOPLE (STAFF_ID,NAME,DOB) VALUES ('A002', 'April      ','5-24-1975')  

INSERT INTO trip_ora.PEOPLE (STAFF_ID,NAME,DOB) VALUES ('B001', 'Bobby      ','12-5-1987')  

INSERT INTO trip_ora.PEOPLE (STAFF_ID,NAME,DOB) VALUES ('B002', 'Bladder      ','1-3-1980')  

INSERT INTO trip_ora.PEOPLE (STAFF_ID,NAME,DOB) VALUES ('B003', 'Brent      ','12-15-1979')  

INSERT INTO trip_ora.PEOPLE (STAFF_ID,NAME,DOB) VALUES ('B004', 'Brelendar      ','8-18-1963')  

INSERT INTO trip_ora.PEOPLE (STAFF_ID,NAME,DOB) VALUES ('C001', 'Calvin      ','4-3-1977')  

INSERT INTO trip_ora.PEOPLE (STAFF_ID,NAME,DOB) VALUES ('C002', 'Cheven      ','2-2-1974')  

INSERT INTO trip_ora.PEOPLE (STAFF_ID,NAME,DOB) VALUES ('C003', 'Clevarance      ','12-6-1987')  

INSERT INTO trip_ora.PEOPLE (STAFF_ID,NAME,DOB) VALUES ('D001', 'Dave      ','8-17-1964')  

INSERT INTO trip_ora.PEOPLE (STAFF_ID,NAME,DOB) VALUES ('D002', 'Davis      ','3-19-1988')  

INSERT INTO trip_ora.PEOPLE (STAFF_ID,NAME,DOB) VALUES ('D003', 'Denny      ','8-5-1985')  

INSERT INTO trip_ora.PEOPLE (STAFF_ID,NAME,DOB) VALUES ('D004', 'Denny      ','2-21-1998')  

INSERT INTO trip_ora.REG_A (DEP_ID,DESCRIP) VALUES ('AA001','Class A Manager      ')
```

```

INSERT INTO trip_ora.REG_A (DEP_ID,DESCRIP) VALUES ('AB001','Class A Manager ')
INSERT INTO trip_ora.REG_A (DEP_ID,DESCRIP) VALUES ('AC001','Class A Manager ')
INSERT INTO trip_ora.REG_A (DEP_ID,DESCRIP) VALUES ('AD001','Class A Assisnat ')
INSERT INTO trip_ora.REG_A (DEP_ID,DESCRIP) VALUES ('AE001','Class A Assisnat ')
INSERT INTO trip_ora.REG_A (DEP_ID,DESCRIP) VALUES ('AF001','Class A Assisnat ')
INSERT INTO trip_ora.REG_A (DEP_ID,DESCRIP) VALUES ('AG001','Class A Clark ')
INSERT INTO trip_ora.REG_A (DEP_ID,DESCRIP) VALUES ('AH001','Class A Assistant ')
INSERT INTO trip_ora.REG_B (DEP_ID,DESCRIPT) VALUES ('BA001','Class B Manager ')
INSERT INTO trip_ora.REG_B (DEP_ID,DESCRIPT) VALUES ('BB001','Class B Manager ')
INSERT INTO trip_ora.REG_B (DEP_ID,DESCRIPT) VALUES ('BC001','Class B Manager ')
INSERT INTO trip_ora.REG_B (DEP_ID,DESCRIPT) VALUES ('BD001','Class B Assistant ')
INSERT INTO trip_ora.REG_B (DEP_ID,DESCRIPT) VALUES ('BE001','Class B Assistant ')
INSERT INTO trip_ora.REG_B (DEP_ID,DESCRIPT) VALUES ('BF001','Class B Assistant ')
INSERT INTO trip_ora.REG_B (DEP_ID,DESCRIPT) VALUES ('BG001','Class B Clark ')
INSERT INTO trip_ora.REG_B (DEP_ID,DESCRIPT) VALUES ('BH001','Class B Assistant ')
INSERT INTO trip_ora.TRIP (TRIP_ID) VALUES ('T0001')
INSERT INTO trip_ora.TRIP (TRIP_ID) VALUES ('T0002')
INSERT INTO trip_ora.TRIP (TRIP_ID) VALUES ('T0003')
INSERT INTO trip_ora.TRIP (TRIP_ID) VALUES ('T0004')
INSERT INTO trip_ora.ENGINEER (TITLE,STAFF_ID) VALUES ('Electronic Engineer ','A001')
INSERT INTO trip_ora.ENGINEER (TITLE,STAFF_ID) VALUES ('Senior Engineer ','B003')
INSERT INTO trip_ora.ENGINEER (TITLE,STAFF_ID) VALUES ('Electronic Engineer ','B004')
INSERT INTO trip_ora.ENGINEER (TITLE,STAFF_ID) VALUES ('Junior Engineer ','C002')
INSERT INTO trip_ora.ENGINEER (TITLE,STAFF_ID) VALUES ('System Engineer ','D001')
INSERT INTO trip_ora.ENGINEER (TITLE,STAFF_ID) VALUES ('Material Engineer ','D002')
INSERT INTO trip_ora.ENGINEER (TITLE,STAFF_ID) VALUES ('Parts Engineer ','D003')
INSERT INTO trip_ora.MANAGER (TITLE,STAFF_ID) VALUES ('Sales Manager ','A002')
INSERT INTO trip_ora.MANAGER (TITLE,STAFF_ID) VALUES ('Marketing Manager ','B001')
INSERT INTO trip_ora.MANAGER (TITLE,STAFF_ID) VALUES ('Sales Manager ','B002')
INSERT INTO trip_ora.MANAGER (TITLE,STAFF_ID) VALUES ('General Manager ','C001')
INSERT INTO trip_ora.MANAGER (TITLE,STAFF_ID) VALUES ('Marketing Manager ','C003')
INSERT INTO trip_ora.ASSIGN (CAR_MODEL,STAFF_ID) VALUES ('MZ-18 ','A002')
INSERT INTO trip_ora.ASSIGN (CAR_MODEL,STAFF_ID) VALUES ('MZ-18 ','B001')
INSERT INTO trip_ora.ASSIGN (CAR_MODEL,STAFF_ID) VALUES ('MZ-18 ','D003')
INSERT INTO trip_ora.ASSIGN (CAR_MODEL,STAFF_ID) VALUES ('R-023 ','B004')
INSERT INTO trip_ora.ASSIGN (CAR_MODEL,STAFF_ID) VALUES ('R-023 ','C001')
INSERT INTO trip_ora.ASSIGN (CAR_MODEL,STAFF_ID) VALUES ('R-023 ','D001')
INSERT INTO trip_ora.ASSIGN (CAR_MODEL,STAFF_ID) VALUES ('SA-38 ','A001')
INSERT INTO trip_ora.ASSIGN (CAR_MODEL,STAFF_ID) VALUES ('SA-38 ','A002')

```

```

INSERT INTO trip_ora.ASSIGN (CAR_MODEL,STAFF_ID) VALUES ('SA-38 ', 'D002')
INSERT INTO trip_ora.ASSIGN (CAR_MODEL,STAFF_ID) VALUES ('WZ-01 ', 'B002')
INSERT INTO trip_ora.ASSIGN (CAR_MODEL,STAFF_ID) VALUES ('WZ-01 ', 'B003')
INSERT INTO trip_ora.ASSIGN (CAR_MODEL,STAFF_ID) VALUES ('WZ-01 ', 'C002')
UPDATE trip_ora52.TRIP SET DEPT_ID= 'AA001', CAR_MODEL= 'MZ-18 ', STAFF_ID= 'A002'
, CAR_MODEL= 'MZ-18 ', STAFF_ID= 'A002' WHERE TRIP_ID= 'T0001'
UPDATE trip_ora52.TRIP SET DEPT_ID= 'AA001', CAR_MODEL= 'MZ-18 ', STAFF_ID= 'B001'
, CAR_MODEL= 'MZ-18 ', STAFF_ID= 'B001' WHERE TRIP_ID= 'T0002'
UPDATE trip_ora52.TRIP SET DEPT_ID= 'AB001', CAR_MODEL= 'SA-38 ', STAFF_ID= 'A002'
, CAR_MODEL= 'SA-38 ', STAFF_ID= 'A002' WHERE TRIP_ID= 'T0003'
UPDATE trip_ora52.TRIP SET DEPT_ID= 'BB001', CAR_MODEL= 'SA-38 ', STAFF_ID= 'D002'
, CAR_MODEL= 'SA-38 ', STAFF_ID= 'D002' WHERE TRIP_ID= 'T0004'

```

Tích hợp dữ liệu

Tích hợp bằng phép hợp nhất

Relation Ra

A ₁	A ₂
a ₁₁	a ₂₁
a ₁₂	a ₂₂

Relation Rx

A ₁	A ₂	A ₃
a ₁₁	a ₂₁	null
a ₁₂	a ₂₂	null
a ₁₃	null	a ₃₁
a ₁₄	null	a ₃₂

Relation Rb

A ₁	A ₃
a ₁₃	a ₃₁
a ₁₄	a ₃₂

==>

Trộn các quan hệ bằng tổng quát hóa

Relation Ra

A ₁	A ₂
a ₁₁	a ₂₁
a ₁₂	a ₂₂

Relation Rx1

A ₁	A ₂
a ₁₁	a ₂₁
a ₁₂	a ₂₂

Relation Rx

A ₁	A ₂	A ₃
a ₁₁	a ₂₁	null
a ₁₂	a ₂₂	null
a ₁₃	null	a ₃₁
a ₁₄	null	a ₃₂

Relation Rb

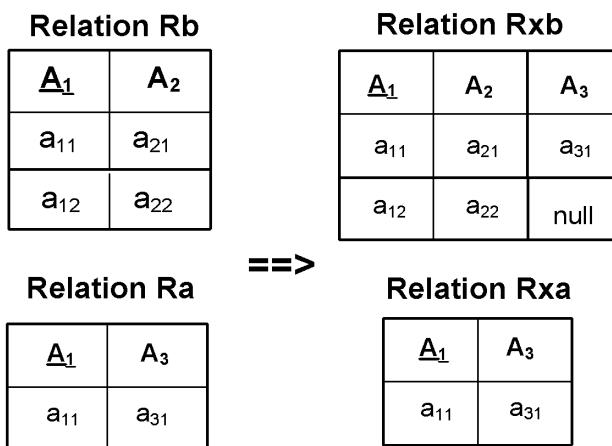
A ₁	A ₃
a ₁₃	a ₃₁
a ₁₄	a ₃₂

==>

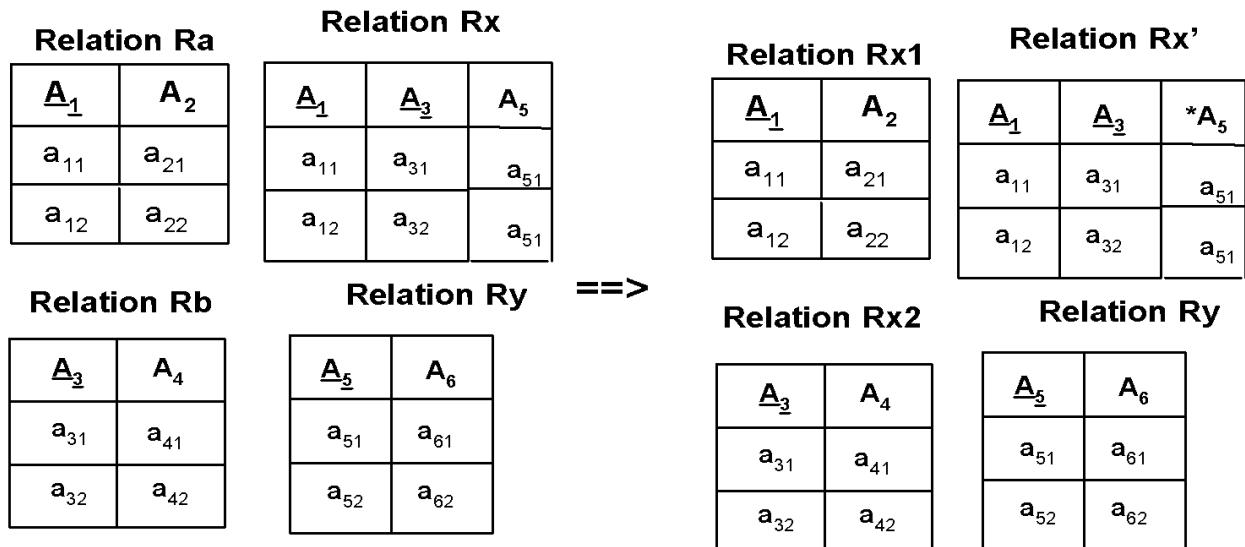
Relation Rx2

A ₁	A ₃
a ₁₃	a ₃₁
a ₁₄	a ₃₂

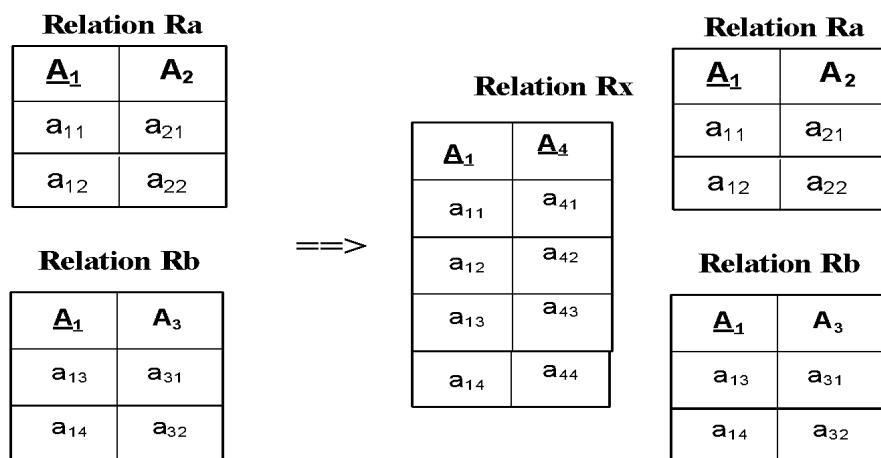
Trộn các quan hệ bằng kề thừa



Trộn các quan hệ bằng tích hợp



Trộn các quan hệ bằng phân loại



Trộn các quan hệ bằng quan hệ ngũ ý giữa chúng

Relation Ra		Relation Rb		Relation Xa		Relation Xb	
A ₁	A ₂	A ₃	A ₁	A ₁	A ₂	A ₃	*A ₁
a ₁₁	a ₂₁	a ₃₁	a ₁₁	a ₁₁	a ₂₁	a ₃₁	a ₁₁
a ₁₂	a ₂₂	a ₃₂	a ₁₂	a ₁₂	a ₂₂	a ₃₂	a ₁₂

==>

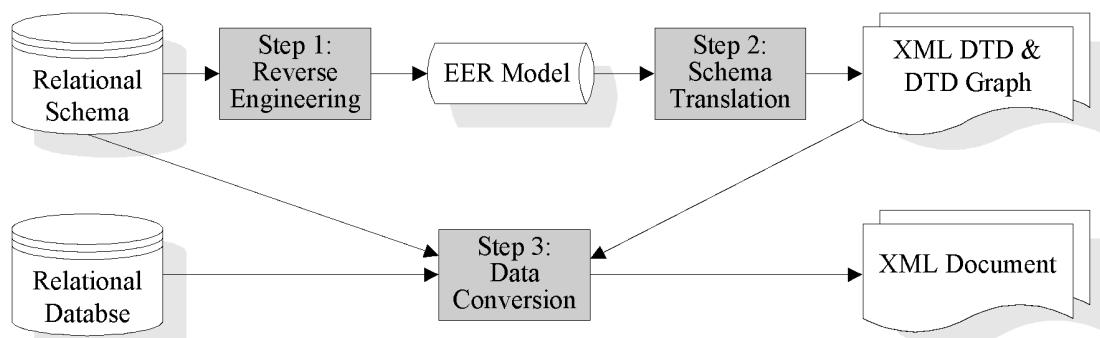
Trộn các quan hệ bằng kiểu con

Relation Ra		Relation Rb		Relation Xa		Relation Xc	
A ₁	A ₂	A ₃	*A ₁	A ₁	A ₂	*A ₃	*A ₁
a ₁₁	a ₂₁	a ₃₁	a ₁₁	a ₁₁	a ₂₁	a ₃₁	a ₁₁
a ₁₂	a ₂₂	a ₃₂	a ₁₂	a ₁₂	a ₂₂	a ₃₂	a ₁₂
Relation Ra'		Relation Rb'		Relation Xa		Relation Xb	
A ₁	A ₂	A ₃	A ₁	A ₁	A ₂	A ₃	A ₁
a ₁₁	a ₂₁	a ₃₁	a ₁₁	a ₁₁	a ₂₁	a ₃₁	a ₁₁
a ₁₂	a ₂₂	a ₃₃	null	a ₁₂	a ₂₂	a ₃₃	null

==>

Chuyển đổi dữ liệu từ dạng quan hệ sang dạng XML

Kiến trúc chuyển đổi ngược từ cơ sở dữ liệu quan hệ sang dạng văn bản XML được thể hiện trong hình vẽ dưới đây



Phương pháp luận cho việc chuyển đổi cơ sở dữ liệu quan hệ sang XML

Là kết quả của việc phiên dịch lược đồ, chúng ta dịch một mô hình thực thể liên kết sang một khung nhìn khác của lược đồ XML là DTD dựa trên phần tử gốc được lựa chọn của chúng. Cho mỗi lược đồ XML được dịch, chúng ta có thể đọc các quan hệ nguồn tương ứng một cách tuân tự

bằng các câu lệnh nhúng SQL bắt đầu từ một quan hệ cha. Các bộ sau đó có thể được tải vào văn bản XML dựa vào cấu trúc DTD của XML. Chúng ta sau đó có thể đọc các bộ quan hệ con và tải chúng vào văn bản XML.

Thuật toán chuyển đổi cơ sở dữ liệu quan hệ sang dạng XML

```
begin
  while not end of element do
    read an element from the translated target DTD;
    read the tuple of a corresponding relation of the element from the source relational database;
    load this tuple into a target XML document;
    read the child elements of the element according to the DTD;
    while not at end of the corresponding child relation in the source relational database do
      read the tuple from the child relation such that the child's corresponding to the processed
      parent relation's tuple;
      load the tuple to the target XML document;
    end loop // end inner loop
  end loop // end outer loop
end
```

Các bước chuyển đổi từ cơ sở dữ liệu quan hệ vào văn bản XML

Bước 1: Chuyển đổi ngược lược đồ quan hệ về mô hình thực thể liên kết

Bằng cách sử dụng bảng phân loại để xác định mối quan hệ giữa các phím và các thuộc tính trong tất cả các mối quan hệ, chúng ta có thể phục hồi dữ liệu của các ngữ nghĩa trong các hình thức của mô hình EER

Bước 2: Chuyển đổi dữ liệu từ quan hệ vào văn bản XML

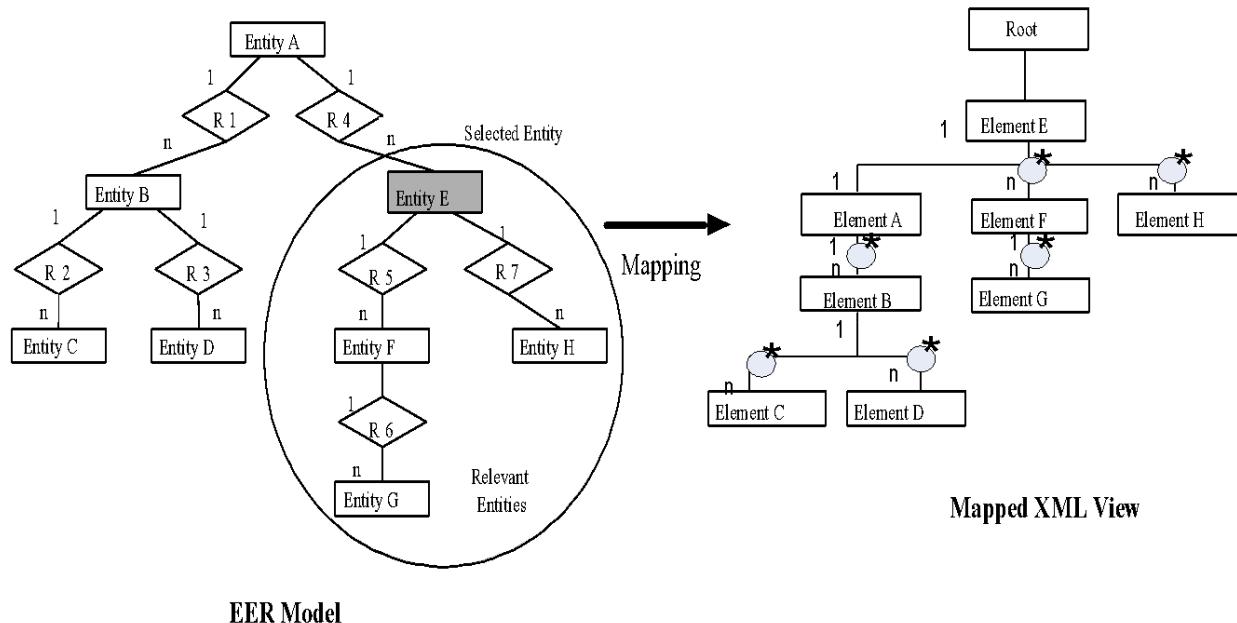
Chúng ta có thể ánh xạ ngữ nghĩa của dữ liệu trong mô hình EER vào đồ thị DTD dựa trên những sự ràng buộc phụ thuộc dữ liệu. Những sự ràng buộc này sau đó có thể được chuyển đổi thành DTD như lược đồ XML.

Bước 2.1 Xác định phần tử gốc

Để lựa chọn 1 phần tử gốc chúng ta phải đưa thông tin liên quan của nó vào lược đồ XML. Những thông tin liên quan này liên đới tới những thực thể mà liên quan đến một thực thể được chọn bởi người sử dụng. Các quan hệ liên quan bao gồm thực thể được lựa chọn và tất cả những thực thể liên quan tới chúng mà được định hướng.

Trong một mô hình EER, chúng ta có thể đi từ một thực thể này tới một thực thể khác tương ứng với một mô hình phân cấp trong XML.

Ví dụ như hình vẽ, bên trái là mô hình EER và được chuyển đổi sang bên phải là DTD của XML



Bước 2.2 Ánh xạ các loại mối quan hệ từ RDB đến XML

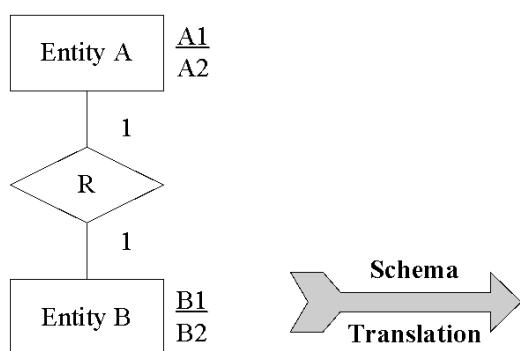
Trong DTD chúng ta dịch

- quan hệ 1-1 vào phần tử cha và phần tử con và
- quan hệ 1-nhiều vào phần tử cha và phần tử con có xuất hiện nhiều lần

Trong quan hệ nhiều-nhiều, chúng được ánh xạ vào DTD của một cấu trúc phân cấp với một định danh ID và tham chiếu định danh IDREF.

Quan hệ 1-1 được dịch như hình vẽ sau đây

EER Model



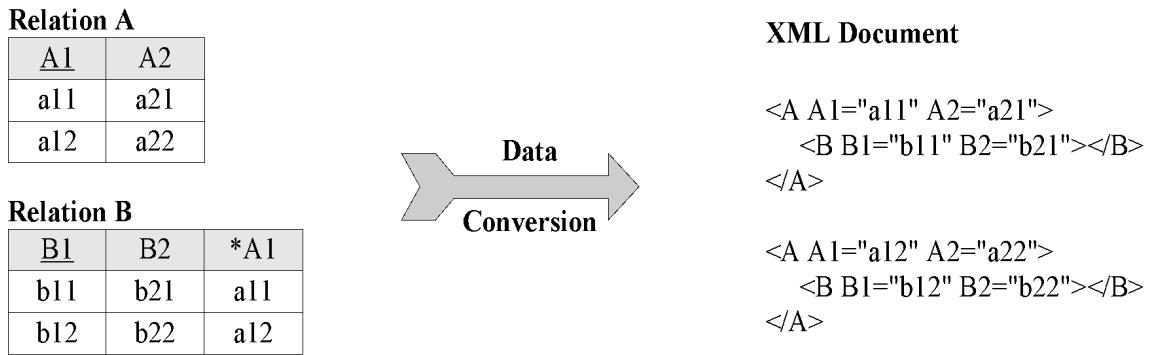
DTD Graph

Relational Schema

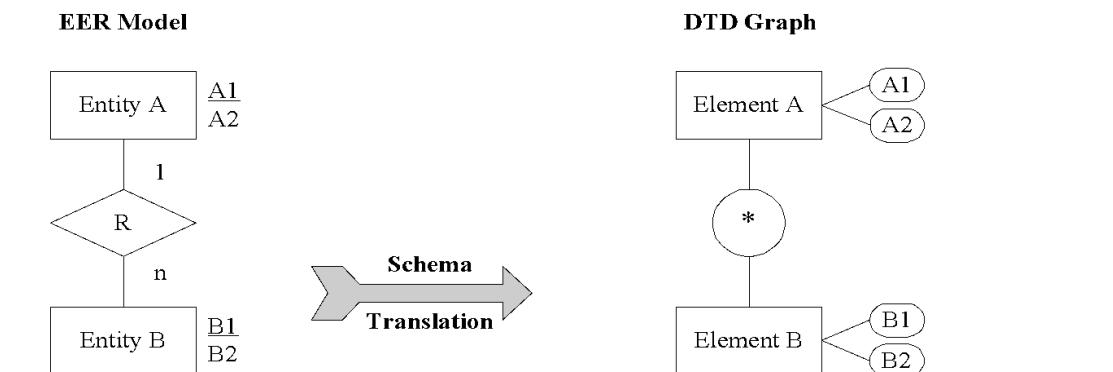
Relation A(A1, A2)
Relation B(B1, B2, *A1)

DTD

```
<!ELEMENT A(B)>
<!ATTLIST A A1 CDATA #REQUIRED>
<!ATTLIST A A2 CDATA #REQUIRED>
<!ELEMENT B EMPTY>
<!ATTLIST B B1 CDATA #REQUIRED>
<!ATTLIST B B2 CDATA #REQUIRED>
```



Quan hệ 1-nhiều được dịch như hình vẽ sau đây



Relational Schema

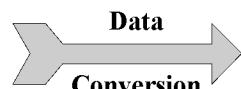
Relation A(A1, A2)
Relation B(B1, B2, *A1)

Relation A

A1	A2
a11	a21
a12	a22

Relation B

B1	B2	*A1
b11	b21	a11
b12	b22	a12
b13	b23	a12



DTD

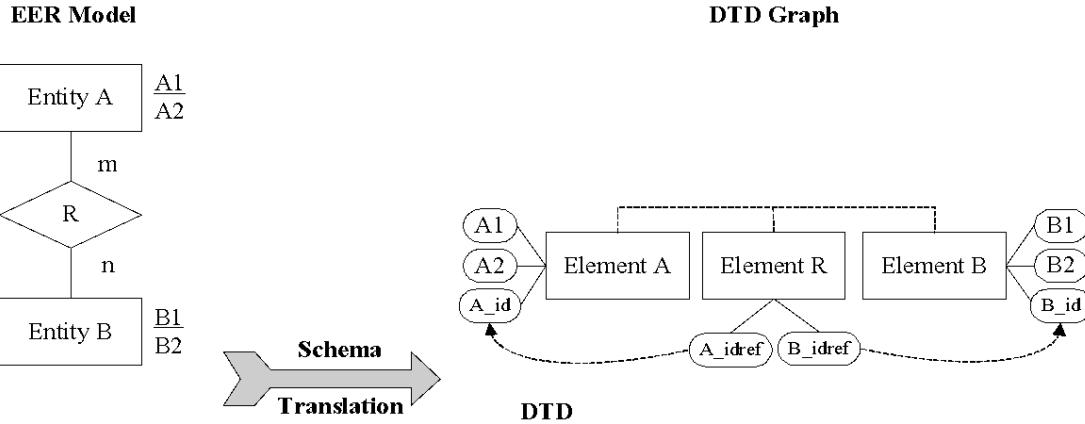
```
<!ELEMENT A(B)*>
<!ATTLIST A A1 CDATA #REQUIRED>
<!ATTLIST A A2 CDATA #REQUIRED>
<!ELEMENT B EMPTY>
<!ATTLIST B B1 CDATA #REQUIRED>
<!ATTLIST B B2 CDATA #REQUIRED>
```

XML Document

```
<A A1="a11" A2="a21">
  <B B1="b11" B2="b21"></B>
</A>

<A A1="a12" A2="a22">
  <B B1="b12" B2="b22"></B>
  <B B1="b13" B2="b23"></B>
</A>
```

Quan hệ nhiều-nhiều được dịch như hình vẽ sau đây



Relational Schema

Relation A(A₁, A₂)
Relation B(B₁, B₂)
Relation R(*A₁, *B₁)

```

<!ELEMENT A EMPTY>
<!ATTLIST A A1 CDATA #REQUIRED>
<!ATTLIST A A2 CDATA #REQUIRED>
<!ATTLIST A A_id ID #REQUIRED>
<!ELEMENT R EMPTY>
<!ATTLIST R A_idref IDREF #REQUIRED>
<!ATTLIST R B_idref IDREF #REQUIRED>
<!ELEMENT B EMPTY>
<!ATTLIST B B1 CDATA #REQUIRED>
<!ATTLIST B B2 CDATA #REQUIRED>
<!ATTLIST B B_id ID #REQUIRED>

```

Relation A

A1	A2
a11	a21
a12	a22

Relation B

B1	B2
b11	b21
b12	b22

Relation R

*A1	*B1
a11	b11
a12	b12
a11	b12



XML Document

```

<A A1="a11" A2="a21" A_id="1"></A>
<B B1="b11" B2="b21" B_id="2"></B>
<R A_idref="1" B_idref="2"></R>

<A A1="a12" A2="a22" A_id="3"></A>
<B B1="b12" B2="b22" B_id="4"></B>
<R A_id="3" B_idref="4"></R>

<R A_id="1" B_idref="4"></R>

```

Câu hỏi ôn tập chương 2

2.1 Trình bày sự giống và khác nhau giữa định nghĩa kiểu văn bản (DTD của XML) và một văn bản có cấu trúc tốt.

2.2 Trình bày sự khác nhau chính giữa tổng quát hóa và phân loại hóa liên quan tới các thẻ hiện dữ liệu trong thực thể/các thực thể lớp cha và lớp con? Có trường hợp đặc biệt nào mà hai ngữ nghĩa dữ liệu này có thể giao nhau không?

2.3 Các lược đồ quan hệ có thể tích hợp được lại với nhau thành một lược đồ quan hệ không?
Giải thích câu trả lời.

2.4 Trình bày các bước giám sát cần thiết của người dùng đối với việc tích hợp hai mô hình thực
thể liên kết mở rộng thành một mô hình thực thể liên kết mở rộng. Giải thích câu trả lời.

2.5 Bạn so sánh ưu điểm và nhược điểm khi sử dụng “cách tiếp cận dịch mức logic” với “cách
tiếp cận sử dụng chương trình tùy biến” trong công việc chuyển đổi dữ liệu.

Chương 3: Công nghệ kho dữ liệu và xử lý phân tích trực tuyến

Nội dung chương này bao gồm:

1. Khái niệm về kho dữ liệu
2. Mô hình dữ liệu đa chiều
3. Kiến trúc kho dữ liệu
4. Cài đặt kho dữ liệu
5. Tương lai phát triển công nghệ khai phá dữ liệu
6. Từ công nghệ kho dữ liệu đến khai phá dữ liệu

3.1 Khái niệm về kho dữ liệu

Kho dữ liệu được định nghĩa theo nhiều cách khác nhau, nhưng không chặt chẽ:

- Có thể được coi như một cơ sở dữ liệu hỗ trợ quyết định mà được duy trì một cách **riêng biệt** từ cơ sở dữ liệu tác nghiệp của một tổ chức.
- Hỗ trợ **xử lý thông tin bằng cách cung cấp một nền tảng vững chắc dữ liệu lịch sử và hợp nhất** cho việc phân tích.

Một trong những định nghĩa hay gấp của kho dữ liệu của W.H .Inmon được phát biểu như sau:

"**Một kho dữ liệu là một bộ dữ liệu hướng chủ đề, tích hợp, biến động theo thời gian, và không mất đi** được sử dụng để hỗ trợ quá trình ra quyết định quản lý "

Khái niệm công nghệ kho dữ liệu được dùng để chỉ quá trình xây dựng và sử dụng kho dữ liệu. Tiếp tới chúng ta cùng phân tích kỹ hơn các đặc điểm của kho dữ liệu được nêu ra trong định nghĩa trên.

- Đặc điểm hướng chủ đề của khi dữ liệu được thể hiện:
 - o Qua việc cung cấp một khung nhìn xúc tích và đơn giản xung quanh các vấn đề của một chủ đề cụ thể. Chúng ta có thể thực hiện đặc điểm này bằng cách loại trừ các dữ liệu không hữu ích trong tiến trình hỗ trợ quyết định.
 - o Qua việc được tổ chức xung quanh các đối tượng chính, chẳng hạn như khách hàng, sản phẩm, bán hàng.
 - o Qua việc tập trung vào mô hình hóa và phân tích các dữ liệu cho những người ra quyết định, không phải cho các hoạt động tác nghiệp hàng ngày hoặc cho xử lý giao dịch.
- Kho dữ liệu có tính tích hợp được thể hiện qua các đặc điểm sau:

- Được xây dựng bằng cách tích hợp nhiều nguồn dữ liệu không đồng nhất như cơ sở dữ liệu quan hệ, các tệp lưu trữ bằng văn bản, các bản ghi của giao dịch trực tuyến
 - Dữ liệu được làm sạch và tích hợp bằng các kỹ thuật tích hợp dữ liệu để đảm bảo tính nhất quán trong quy ước đặt tên, mã hóa cấu trúc, các độ đo thuộc tính, v.v... giữa các nguồn dữ liệu khác nhau ví dụ như Giá khách sạn bao gồm đơn vị tiền tệ, thuế, giá ăn sáng ...
 - Khi dữ liệu được chuyển đến kho, nó cần được chuyển đổi sang một dạng thống nhất.
- Kho dữ liệu biến đổi theo thời gian được thể hiện ở các đặc điểm sau:
- Trục hoành chỉ thời gian trong các kho dữ liệu dài hơn đáng kể so với các hệ thống tác nghiệp vì trong cơ sở dữ liệu tác nghiệp thể hiện giá trị hiện hành của dữ liệu còn trong kho dữ liệu cung cấp thông tin từ một khung nhìn lịch sử (ví dụ, dữ liệu qua 5-10 năm)
 - Tất cả các cấu trúc quan trọng trong kho dữ liệu đều chứa yếu tố về thời gian một cách tường minh hoặc không tường minh. Cách thể hiện thời gian một cách tường minh là trong cấu trúc của dữ liệu có một thuộc tính thời gian, cách không tường minh thường được thể hiện thông qua một số các thuộc tính khác mà không nói rõ đó là thời gian, sự không tường minh được thể hiện khác nhau trong các trường hợp khác nhau. Tuy nhiên, khóa của dữ liệu tác nghiệp có thể chứa hoặc không chứa những "yếu tố thời gian".
- Kho dữ liệu là không mất đi khi tắt điện được thể hiện ở những đặc điểm sau
- Là một kho lưu trữ riêng biệt về mặt vật lý của những dữ liệu được chuyển đổi từ môi trường tác nghiệp vào.
 - Thao tác cập nhật dữ liệu tác nghiệp không nhất thiết xảy ra trong môi trường kho dữ liệu. Lý do là vì
 - Kho dữ liệu không chứa thông tin về xử lý giao dịch, phục hồi dữ liệu và các cơ chế kiểm soát việc xảy ra đồng thời.
 - Thường chỉ đòi hỏi có hai thao tác truy xuất dữ liệu là tải dữ liệu vào kho lúc khởi tạo và truy nhập dữ liệu có sẵn trong kho.

So sánh kho dữ liệu với cơ sở dữ liệu hỗn tạp

Chúng ta dễ nhầm lẫn kho dữ liệu với cơ sở dữ liệu hỗn tạp bởi cả hai đều chứa nhiều loại dữ liệu khác nhau và các dữ liệu này được tích hợp lại trong một thể thống nhất. Tuy nhiên kho dữ liệu khác hẳn cơ sở dữ liệu hỗn tạp về bản chất:

- Việc tích hợp cơ sở dữ liệu hỗn tạp truyền thống cần
 - o Xây dựng một thành phần đóng gói (trung gian, bao bọc) cho toàn bộ các dữ liệu hỗn tạp khác nhau
 - o Có cách tiếp cận hướng truy vấn có nghĩa là một truy vấn do người dùng yêu cầu được dịch sang các câu truy vấn tương ứng với từng thành phần hỗn tạp; kết quả trả về sẽ được tích hợp thành một tập câu trả lời toàn cục.
 - o Liên quan tới việc lọc thông tin phức tạp
 - o Cạnh tranh tài nguyên tại các nguồn cục bộ địa phương
- Kho dữ liệu có xu hướng chỉ thực hiện các thao tác cập nhật với hiệu năng cao
 - o Thông tin từ các nguồn hỗn tạp được tích hợp trước và được lưu trữ trong kho để phân tích và truy vấn trực tiếp.

So sánh kho dữ liệu với cơ sở dữ liệu tác nghiệp

Hai loại này khác nhau chủ yếu về cách sử dụng dữ liệu được lưu trữ trong chúng:

- Nhiệm vụ của cơ sở dữ liệu quan hệ truyền thống là thực hiện xử lý giao dịch trực tuyến, chủ yếu thực hiện các hoạt động hàng ngày như mua bán, thống kê hàng tồn kho, giao dịch ngân hàng, quản lý sản xuất, tiền lương, đăng ký, kế toán sổ sách, v.v...
- Nhiệm vụ của kho dữ liệu là thực hiện các phân tích trực tuyến, chủ yếu thực hiện các công việc phân tích và hỗ trợ ra quyết định cho một tổ chức nào đó.
- Đặc điểm phân biệt phân tích trực tuyến (OLAP) và xử lý trực tuyến (OLTP) chính là sự khác nhau cơ bản của kho dữ liệu và cơ sở dữ liệu truyền thống, được thể hiện ở những điểm sau:

- o OLTP có định hướng người sử dụng còn OLAP có định hướng hệ thống: OLTP phục vụ khách hàng còn OLAP phục vụ thị trường
- o Nội dung dữ liệu khác nhau: OLTP mang nội dung hiện tại, chi tiết còn OLAP mang nội dung lịch sử, hợp nhất.
- o Thiết kế cơ sở dữ liệu: OLTP sử dụng mô hình thực thể liên kết ER đi cùng với ứng dụng còn OLAP sử dụng mô hình sao cùng với chủ thể

- Khung nhìn dữ liệu: OLTP cung cấp khung nhìn hiện tại, cục bộ còn OLAP cung cấp khung nhìn dữ liệu cho các thao tác chỉ đọc nhưng cho những truy vấn phức tạp.
- Các mẫu truy cập dữ liệu: OLTP cần các truy vấn cập nhật đơn giản là chính trong khi OLAP sử dụng các truy vấn chỉ đọc nhưng phức tạp.

Tách rời kho dữ liệu khỏi cơ sở dữ liệu tác nghiệp

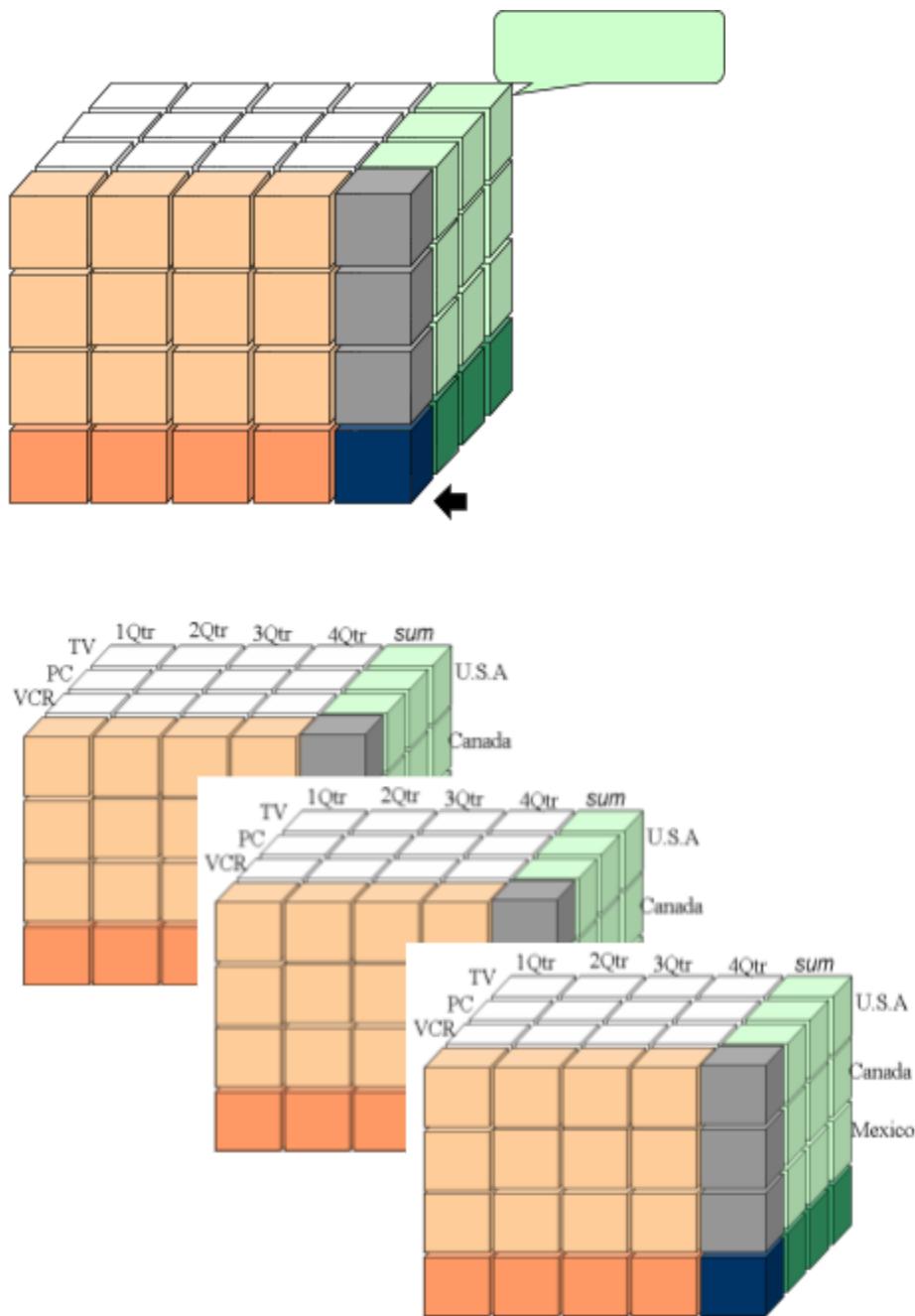
Cần thực hiện việc này bởi những lý do sau:

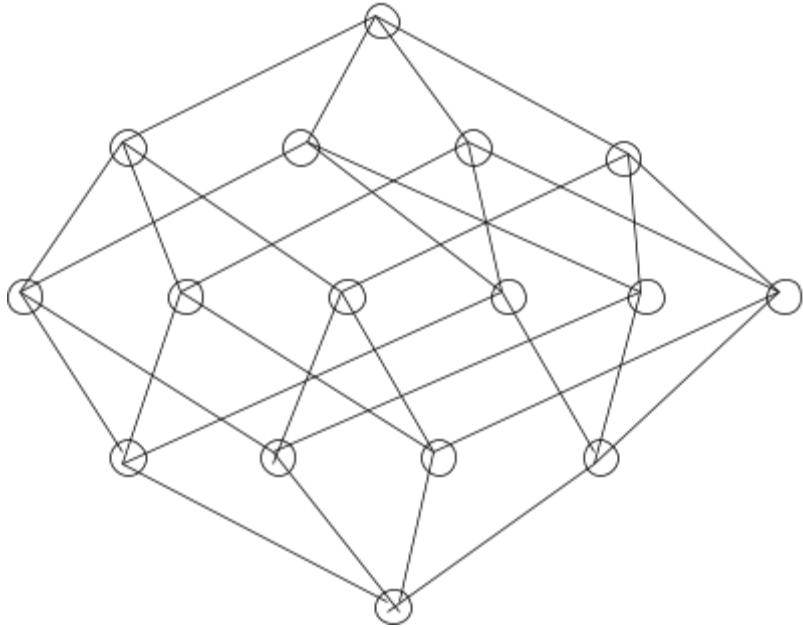
- Hiệu suất cao cho cả hai hệ thống:
 - Hệ quản trị cơ sở dữ liệu đã được thiết kế tốt để phục vụ cho các hoạt động xử lý trực tuyến về các phương pháp truy cập dữ liệu, đánh chỉ mục, kiểm soát xử lý đồng thời, phục hồi dữ liệu.
 - Kho dữ liệu được thiết kế tốt phục vụ các hoạt động xử lý phân tích trực tuyến bao gồm xử lý các truy vấn OLAP, tạo khung nhìn đa chiều và cung cấp dữ liệu.
- Chức năng khác nhau và dữ liệu khác nhau:
 - Hệ hỗ trợ quyết định mà kho dữ liệu cung cấp yêu cầu dữ liệu lịch sử trong khi đó cơ sở dữ liệu tác nghiệp thường không duy trì.
 - Hệ hỗ trợ quyết định cần cung cấp dữ liệu (tích hợp, tổng hợp) từ các nguồn dữ liệu hỗn tạp khác nhau.
 - Các nguồn khác nhau thường sử dụng dữ liệu không nhất quán, mã số và các định dạng của chúng cần phải được đổi chiều.

3.2 Mô hình dữ liệu đa chiều

- Một kho dữ liệu được thiết kế dựa trên một mô hình dữ liệu đa chiều, mô hình cung cấp khả năng xem dữ liệu dưới dạng một khối dữ liệu.
- Một khối dữ liệu cho phép dữ liệu được mô hình hóa và xem ở nhiều chiều (thuộc tính) khác nhau:
 - Các chiều của một khối dữ liệu được thể hiện thông qua bảng theo chiều (Dimension), chẳng hạn như bảng các mặt hàng bao gồm các thuộc tính (tên mặt hàng, thương hiệu, loại hàng), hoặc bảng thời gian bao gồm các thuộc tính (ngày, tuần, tháng, quý, năm).
 - Bảng sự kiện (Fact) chứa các giá trị đo lường được (như số tiền bán được -dollars_sold) và các khóa tới mỗi bảng theo chiều liên quan.

- Trong các tài liệu nói về công nghệ kho dữ liệu, một khối dữ liệu cơ sở n chiều (n-D base cube) được gọi là một khối (hình được bao bọc bởi 6 mặt- cuboid) cơ bản. Khối ở mức trên cùng là khối 0-D với số chiều là 0 chứa số liệu tổng hợp ở mức cao nhất được gọi là khối dữ liệu mức đỉnh. Lưới chia ngăn của các khối dữ liệu đó tạo thành các khối dữ liệu mức thấp hơn. Chúng ta sẽ xem xét ví dụ dưới đây để hiểu rõ hơn về khái niệm khối dữ liệu này.
- Một ví dụ về khối dữ liệu được thể hiện trong hình vẽ dưới đây



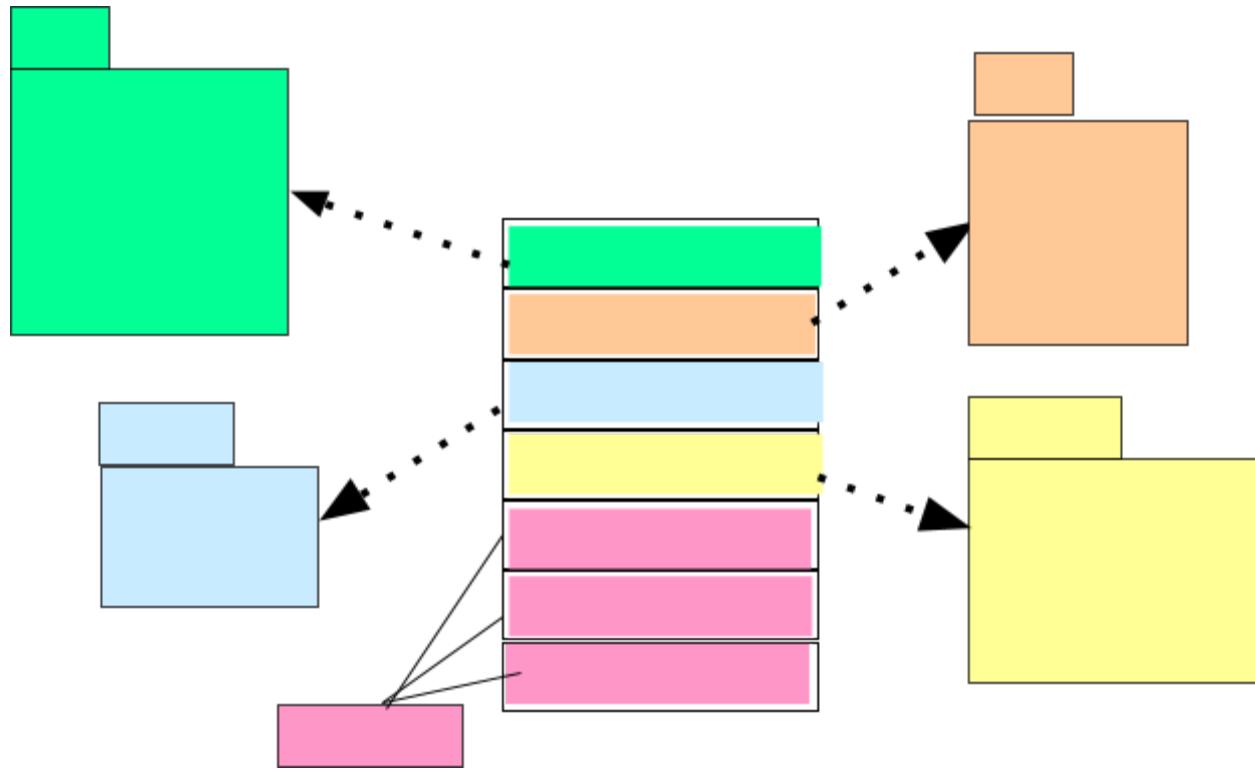


Mô hình hóa dữ liệu cho kho dữ liệu

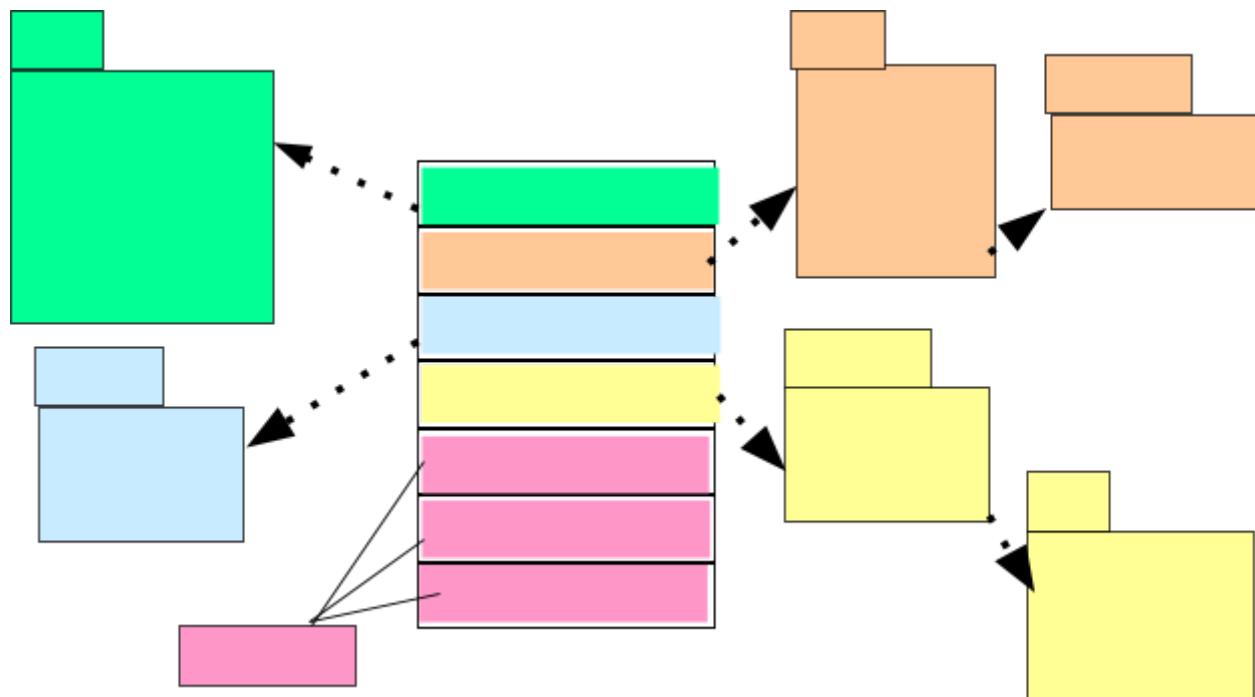
Bao gồm 3 loại lược đồ: hình sao, hình bông tuyết và dải thiên hà

- Lược đồ hình sao: Một bảng Fact ở giữa kết nối với nhiều bảng theo chiều
- Lược đồ hình bông tuyết: Là một dạng chuẩn hóa của mô hình hình sao trong đó sự phân cấp của các chiều được chuẩn hóa thành một tập hợp các bảng theo chiều có kích thước nhỏ hơn, tạo thành một hình dạng tương tự như bông tuyết.
- Lược đồ dải thiên hà: gồm nhiều bảng Fact có chung các bảng theo chiều, mà mỗi bảng Fact cùng các bảng theo chiều là mô hình hình sao, được coi như một ngôi sao vì thế mô hình này được xem như là một bộ sưu tập của các ngôi sao, và được gọi là lược đồ dải thiên hà hay một dải các bảng fact.

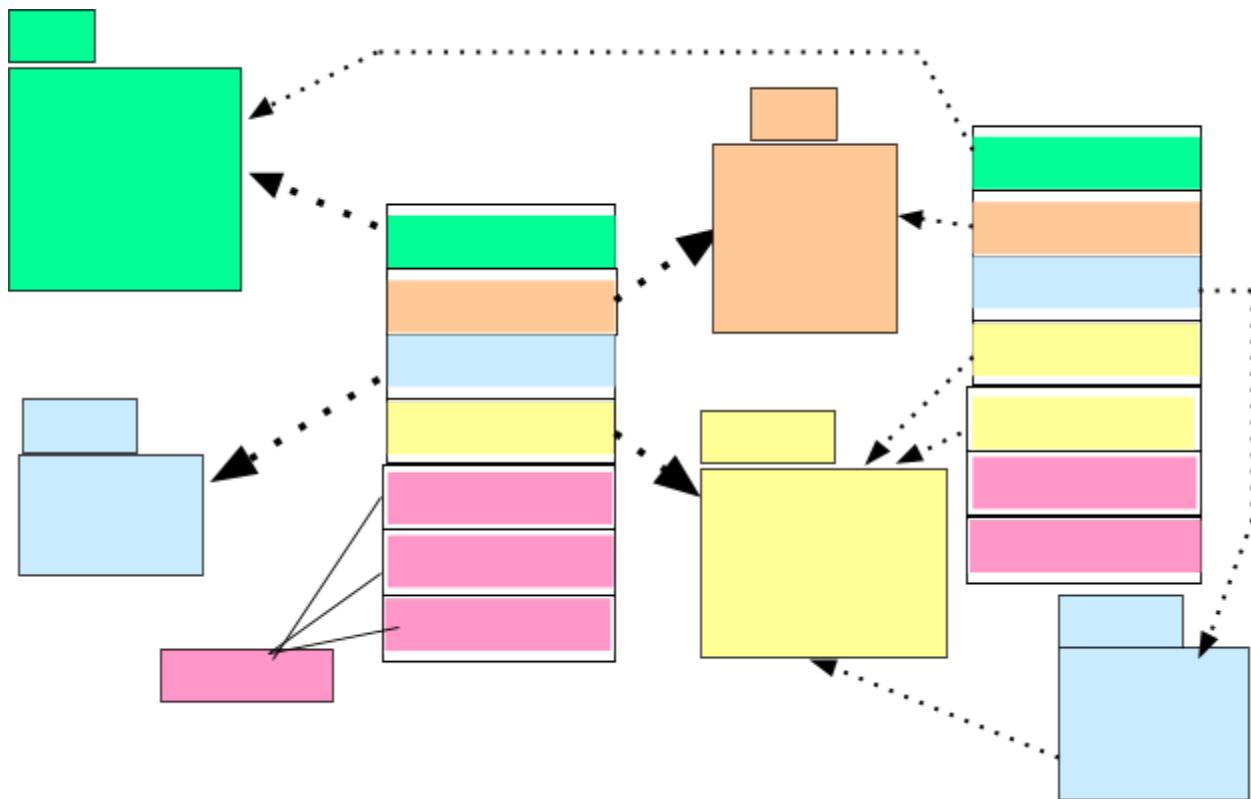
Ví dụ về lược đồ hình sao



Ví dụ lược đồ hình bông tuyết



Ví dụ lược đồ dải thiên hà



Ngôn ngữ truy vấn khai phá dữ liệu (Data Mining Query Language)

Bao gồm các hàm nguyên thủy như sau

- Định nghĩa một khối dữ liệu: vì bảng Fact cũng là một khối dữ liệu ở mức thấp nhất, chứa thông tin ở mức chi tiết nhất nên đây cũng là câu lệnh để định nghĩa một bảng Fact
define cube <tên_khối> [<danh sách các chiều>]: <danh sách các độ đo>
- Định nghĩa các chiều (bảng theo chiều)
define dimension <tên_chiều> as (<danh sách các thuộc tính hoặc thuộc tính con>)
- Trường hợp đặc biệt (dùng chung các bảng theo chiều)
 - o Bảng theo chiều được khai báo lần đầu khi định nghĩa khối dữ liệu liên quan
 - o Nếu muốn dùng lại bảng theo chiều này trong những trường hợp về sau thì khai báo với cú pháp như sau**define dimension <tên_chiều> as <tên_chiều_được_khai_báo_lần_đầu> in cube <tên_khối_đầu_tiên_sử_dụng_chiều_đó>**

Sử dụng các hàm nguyên thủy định nghĩa lược đồ hình sao được thể hiện trong ví dụ trên như sau:

define cube sales_star [time, item, branch, location]:

```
dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars), units_sold = count(*)  
define dimension time as (time_key, day, day_of_week, month, quarter, year)  
define dimension item as (item_key, item_name, brand, type, supplier_type)  
define dimension branch as (branch_key, branch_name, branch_type)  
define dimension location as (location_key, street, city, province_or_state, country)
```

Định nghĩa lược đồ hình bông tuyết được thể hiện trong ví dụ trên như sau:

```
define cube sales_snowflake [time, item, branch, location]:  
dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars), units_sold = count(*)  
define dimension time as (time_key, day, day_of_week, month, quarter, year)  
define dimension item as (item_key, item_name, brand, type, supplier(supplier_key,  
supplier_type))  
define dimension branch as (branch_key, branch_name, branch_type)  
define dimension location as (location_key, street, city(city_key, province_or_state, country))
```

Định nghĩa lược đồ dài thiên hà được thể hiện trong ví dụ trên như sau:

```
define cube sales [time, item, branch, location]:  
dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars), units_sold = count(*)  
define dimension time as (time_key, day, day_of_week, month, quarter, year)  
define dimension item as (item_key, item_name, brand, type, supplier_type)  
define dimension branch as (branch_key, branch_name, branch_type)  
define dimension location as (location_key, street, city, province_or_state, country)  
define cube shipping [time, item, shipper, from_location, to_location]:  
dollar_cost = sum(cost_in_dollars), unit_shipped = count(*)  
define dimension time as time in cube sales  
define dimension item as item in cube sales  
define dimension shipper as (shipper_key, shipper_name, location as location in cube sales,  
shipper_type)  
define dimension from_location as location in cube sales  
define dimension to_location as location in cube sales
```

Thuộc tính độ đo trong lược đồ được là một hàm được tính toán trên những dữ liệu đã được tích hợp lại dựa trên những cặp giá trị theo chiều cho trước. Thuộc tính độ đo có thể thuộc một trong ba loại như sau:

- **Phân phối:** nếu hàm có thể được tính theo phương cách phân phối. Ví dụ như các hàm count(); sum(); min(),max().
- **Đại số:** nếu nó có thể được tính từ các đối số thu được bằng cách áp dụng các chức năng phân phối tổng hợp. Ví dụ: avg()=sum()/count(), min_N(), standard_deviation().
- **Loại khác** nếu như không phải đại số. Ví dụ median(),mode(),rank()

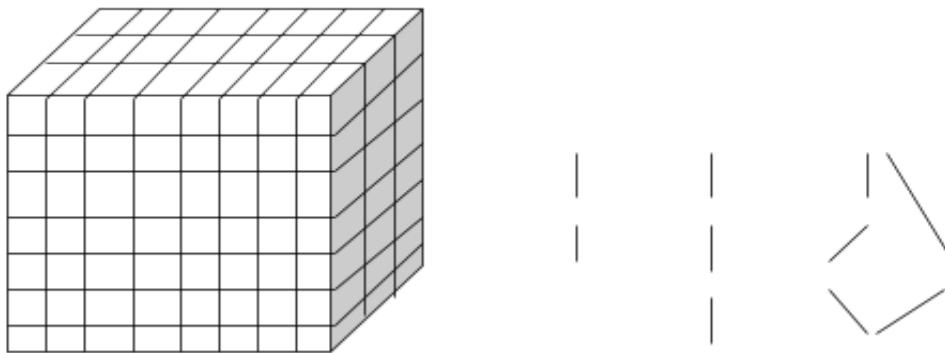
Các loại hàm phân phối và đại số thích hợp một cách lý tưởng cho việc tính toán các khối dữ liệu bởi chúng phân mảnh nhỏ hơn được, phương thức áp dụng tính toán ở mức thấp rồi tích hợp lên mức cao được áp dụng dễ dàng đối với các hàm loại này. Việc tính toán các độ đo ở mức độ chi tiết hơn (sẽ được xem xét đến trong phần cài đặt kho dữ liệu) sẽ được sử dụng lại trong quá trình tính toán ở các cấp chi tiết cao hơn. Mặt khác, các hàm thuộc loại khác khó tính toán một cách hiệu quả như vậy, chỉ thường có thể tính toán xấp xỉ một cách hiệu quả.

Khái niệm phân cấp

Các phân cấp khái niệm cho phép dữ liệu có thể được xử lý tại các mức trùu tượng khác nhau. Ta cùng xét ví dụ cho khái niệm phân cấp với một khối dữ liệu ba chiều được thể hiện trong hình vẽ dưới đây bao gồm các chiều thể hiện sản phẩm (Product), thời gian là các tháng (month) và vị trí địa lý của sản phẩm là vùng miền sản xuất (Region). Mỗi một chiều dữ liệu có thể có các phân cấp khác nhau để thể hiện các mức độ trùu tượng khác nhau của dữ liệu. Ở ví dụ này, chiều dữ liệu Product có một phân cấp Industry□Category□ Product để thể hiện một sự tổng quát hóa(summaried) hay chi tiết hóa các dữ liệu trong một chiều. Ý nghĩa của phân cấp này nói rằng: một ngành công nghiệp (Industry) thì có nhiều loại sản phẩm (Category) và mỗi loại sản phẩm thì có nhiều sản phẩm khác nhau vì thế khi biết số lượng hàng bán được của mỗi sản phẩm thì sẽ tổng hợp được số hàng bán được của mỗi loại sản phẩm, cũng như tổng hợp được số hàng bán được của mỗi ngành công nghiệp, nhờ phân cấp chúng ta thiết kế ra cho kho dữ liệu.

Tương tự như vậy, ta thiết kế một phân cấp cho chiều Region nếu như chúng ta muốn phân tích dữ liệu cho chiều này ở các mức chi tiết (trùu tượng) khác nhau. Cụ thể phân cấp như sau: Region□Country□City□Office. Ý nghĩa của phân cấp này nói rằng: mỗi vùng địa lý trên thế giới có nhiều nước, mỗi nước có nhiều thành phố, và mỗi thành phố có thể có một hoặc nhiều văn phòng đại diện của công ty đó. Vì thế, nếu biết số lượng tiền lãi thu được của mỗi văn phòng đại diện thì sẽ tổng hợp được tiền lãi của từng thành phố, của từng quốc gia và của từng vùng địa

lý. Phân cấp cho chiều thời gian từ năm đến ngày được thể hiện trong hình vẽ dưới đây



Nhờ có phân cấp, chúng ta biết được mức độ chi tiết của dữ liệu trong từng chiều của kho dữ liệu và thiết kế được cách tổng hợp dữ liệu sử dụng cho việc phân tích từ mức cao tới mức thấp hay ngược lại.

Các thao tác cơ bản của xử lý phân tích trực tuyến

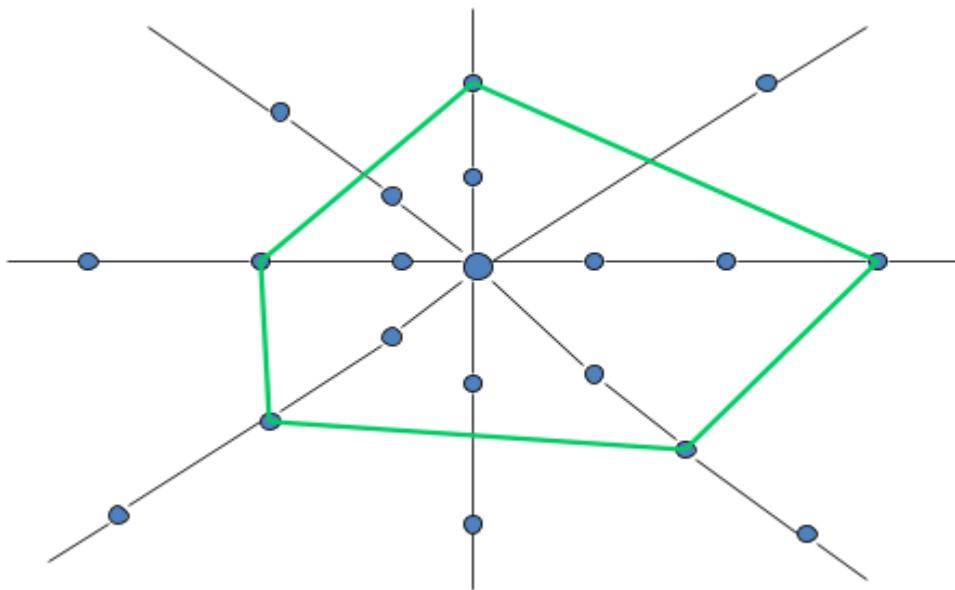
Xử lý phân tích trực tuyến bao gồm các thao tác cơ bản sau đây

- **Cuộn lên (Drill-up):** dùng để tổng hợp dữ liệu từ mức thấp lên mức cao. Thao tác được dùng khi tổng hợp dữ liệu ở các mức độ khác nhau từ thấp đến cao trong một phân cấp của một chiều nào đó hoặc khi muốn giảm chiều dữ liệu, cung cấp dữ liệu tổng hợp theo một số lượng chiều ít hơn số lượng chiều dữ liệu ban đầu của khối dữ liệu (sẽ xem xét ví dụ minh họa sau)
- **Khoan xuống (Drill-down):** ngược với thao tác cuộn lên, dùng để xem dữ liệu ở mức độ chi tiết hơn của một phân cấp theo một chiều nào đó hoặc khi muốn tăng số chiều của dữ liệu, xem chi tiết theo số lượng chiều nhiều hơn dữ liệu hiện tại.
- **Cắt ngang (dice) và cắt dọc (slide):** giống như phép chiếu và phép chọn trong đại số quan hệ
- **Xoay (pivot):** dùng để định hướng hay xoay lại khối dữ liệu theo một số chiều mà người sử dụng quan tâm, hoặc để thể hiện biểu diễn dữ liệu một cách trực quan, chuyển dạng biểu diễn 3 chiều thành một chuỗi các biểu diễn 2 chiều.
- Các thao tác khác: xuất phát từ những thao tác cơ bản trên
 - **Khoan chéo (Drill across):** bản chất giống thao tác khoan xuống và cuộn lên nhưng ở đây thao tác không chỉ liên quan tới một bảng Fact mà liên quan tới

nhiều bảng Fact có chung một số thuộc tính (chiều) để cuộn lên xuống ở các mức khác nhau

- **Khoan xuyên suốt (Drill through)**: cuộn lên từ đáy của khôi dữ liệu lên đến mức các bảng quan hệ đầu cuối (sử dụng ngôn ngữ SQL để thực hiện)

Để thực hiện một truy vấn trong kho dữ liệu chúng ta có thể sử dụng một mô hình **mạng hình sao** (Star-Net) để thiết lập câu lệnh. Xét ví dụ trong hình vẽ dưới đây để hiểu được mô hình này



Giả sử cho sẵn một **kho dữ liệu** có các 8 chiều như hình vẽ trên bao gồm: Time (thời gian), Shipping Method (phương thức chuyển hàng), Customer Orders (các loại đơn đặt hàng của khách hàng), Customers (Khách hàng), Product (Sản phẩm), Organization (loại tổ chức bán hàng), Promotion (loại khuyến mại), Location (vị trí bán hàng). Một số chiều có phân cấp được thể hiện trong hình vẽ, như chiều Time có phân cấp theo annually (hàng năm), quarterly (hàng quý), daily (hàng ngày), chiều Organization được phân cấp thành các Division (chi nhánh), District (một nhóm người), Sales Person (người bán hàng), Customer Orders được phân cấp thành mua theo Hợp đồng (Contracts) và mua theo từng đơn đặt hàng một (Order) v.v...

Chúng ta cần trả lời **một câu truy vấn** như sau: hãy **tính số tiền thu được của việc bán hàng theo mỗi nhóm sản phẩm, của từng nhóm người, của từng nước, hàng quý và của từng hợp đồng** được đặt mua. Câu truy vấn này sẽ cần lấy dữ liệu theo 5 chiều là Product, Organization, Region, Time và Customer Orders, trong đó mỗi chiều đều có phân cấp dữ liệu ở mức độ chi tiết khác nhau. Mức độ ở chiều Product là Product Group (vì cần tổng hợp theo từng nhóm mặt hàng), ở chiều

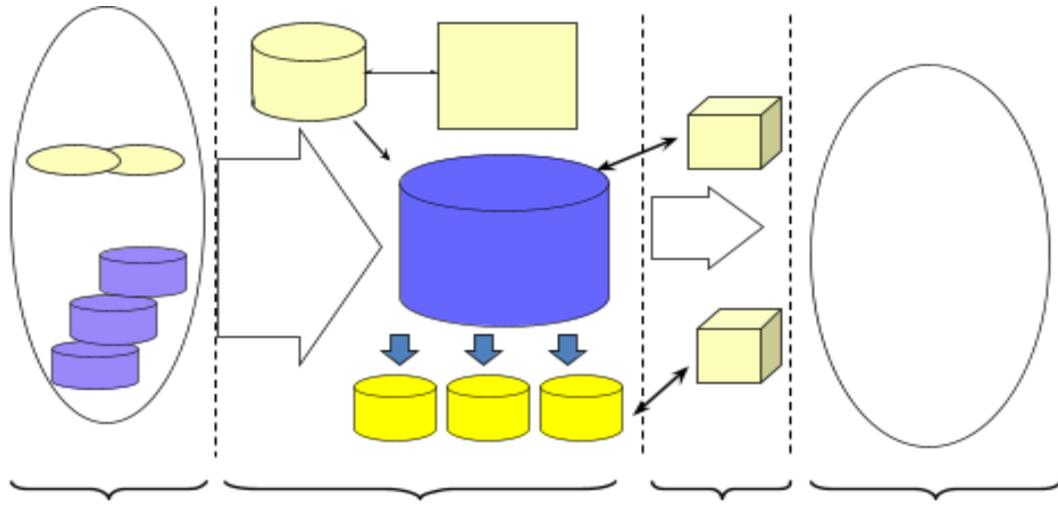
Organization là District (vì theo từng nhóm người), ở chiều Location là Country (vì theo từng nước), ở chiều Time thì là Qtrly, ở chiều Customer Orders là Contracts (vì theo từng hợp đồng), câu truy vấn lấy dữ liệu ở 5 chiều trong tổng số 8 chiều và ở các mức như trên nên được thể hiện bởi một hình đa giác 5 cạnh như trên hình vẽ, đây chính là mô hình mạng sao của truy vấn. Quay trở lại thiết kế của kho dữ liệu, bảng Fact chứa dữ liệu với số chiều lớn nhất và ở mức thấp nhất ở mỗi chiều có phân cấp. Qui chiếu vào mô hình mạng sao trên thì bảng Fact chính là tâm điểm của sơ đồ trên.

3.3 Kiến trúc của kho dữ liệu

Tiến trình thiết kế một kho dữ liệu

- Một số cách tiếp cận sau để thiết kế kho dữ liệu: Cách tiếp cận từ trên xuống (Top-down), từ dưới lên (bottom-up) hoặc sự kết hợp của 2 phương pháp đó
 - Từ trên xuống: Bắt đầu với thiết kế tổng thể và lập kế hoạch, thường thì được áp dụng đối với các đội dự án có kinh nghiệm
 - Từ dưới lên: Bắt đầu với các thử nghiệm và nguyên mẫu, thường áp dụng đối với những dự án cần tốc độ xây dựng nhanh
- Trên quan điểm của công nghệ phần mềm, xây dựng kho dữ liệu có thể tuân thủ theo một trong các mô hình sau đây:
 - Mô hình thác nước (Waterfall): trong đó việc phân tích cấu trúc và hệ thống được thực hiện từng bước một trước khi tiến hành bước kế tiếp.
 - Mô hình xoắn ốc (Spiral): xây dựng thêm các chức năng một cách nhanh chóng, sửa đổi nhanh, thích ứng kịp thời với thiết kế và công nghệ mới.
- Tiến trình thiết kế kho dữ liệu điển hình
 - Chọn một tiến trình kinh doanh (business process) hoặc một công việc nào đó để mô hình, ví dụ: quản lý đặt hàng, hóa đơn....
 - Chọn dữ liệu ở mức độ nhỏ nhất (grain) của quá trình kinh doanh cần lưu trữ
 - Chọn các chiều mà sẽ áp dụng cho mỗi bản ghi của bảng Fact
 - Chọn độ đo được sinh ra cho mỗi bản ghi của bảng Fact

Kiến trúc đa tầng của công nghệ kho dữ liệu



Ba loại mô hình kho dữ liệu

- Kho dữ liệu doanh nghiệp (Enterprise warehouse): Thu thập tất cả thông tin về chủ thể bao trùm toàn bộ tổ chức.
- Kho dữ liệu con theo chủ đề (Data mart): Một tập hợp con dữ liệu của toàn doanh nghiệp có giá trị cho một nhóm người sử dụng cụ thể. Phạm vi của nó có giới hạn cho một nhóm người sử dụng được lựa chọn trước, ví dụ như quảng cáo doanh nghiệp
 - o Có hai loại kho dữ liệu chủ đề: Độc lập và phụ thuộc với kho dữ liệu
- Kho dữ liệu ảo:
 - o Bao gồm một tập hợp các khung nhìn của cơ sở dữ liệu tác nghiệp
 - o Chỉ một số khung nhìn có khả năng tổng hợp dữ liệu lên mức cao.

Các kiến trúc của máy chủ cho việc xử lý phân tích trực tuyến (OLAP)

- OLAP quan hệ (Relational OLAP) hay còn gọi là ROLAP
 - o Dùng hệ quản trị cơ sở dữ liệu quan hệ hoặc quản hệ mở rộng để lưu trữ và quản lý kho dữ liệu
 - o Bao gồm sự tối ưu hóa các công việc nền tảng của cơ sở dữ liệu cũng như các công cụ phụ trợ bổ sung và các dịch vụ
 - o Có khả năng mở rộng lớn hơn
- OLAP đa chiều (Multidimensional OLAP) hay còn gọi là MOLAP

The diagram illustrates the multidimensional representation of a sales fact table. On the left, a 2D table is shown with columns: pid, timeid, locid, and sales. The data consists of 10 rows with values ranging from 8 to 50. To the right, this data is visualized as a 3D cube. The cube's dimensions are labeled: pid (11, 12, 13) on the vertical axis, locid (1, 2, 3) on the horizontal axis, and timeid (1, 2, 3) on the depth axis. The sales values are placed at the intersections of these axes. For example, the value 8 is at pid=11, locid=1, timeid=1; the value 10 is at pid=12, locid=1, timeid=2; and so on.

- Có **mô hình lưu trữ mảng dữ liệu đa chiều dựa trên cấu trúc mảng** (sử dụng các kỹ thuật với các ma trận thưa)
- Lập chỉ mục nhanh để tính toán trước khi tổng hợp dữ liệu
- OLAP lai (Hybrid OLAP) hay còn gọi là HOLAP
 - Người dùng sử dụng ROLAP và MOLAP một cách linh hoạt (thường truy vấn mức thấp thì sử dụng ROLAP, còn mức cao dùng mảng hay MOLAP).
- Các máy chủ SQL chuyên dụng
 - Chuyên hỗ trợ cho các truy vấn SQL trên lược đồ hình sao hay lược đồ bông tuyết

3.4 Cài đặt kho dữ liệu

Khi cài đặt kho dữ liệu với cách thiết kế đã được trình bày ở trên, chúng ta cần quan tâm tới một số vấn đề trình bày dưới đây

Tính toán khối dữ liệu một cách hiệu quả

- **Khối dữ liệu** có thể xem như là **mạng lưới** của các **khối cơ bản**
 - **Khối dữ liệu** ở đáy dưới cùng của khối dữ liệu được xem là **khối cơ sở**
 - **Khối** trên đỉnh cao nhất của khối dữ liệu chỉ chứa **một ô**
 - Chúng ta cùng xác định xem có bao nhiêu khối lập phương trong khối dữ liệu **n** chiều với mỗi chiều có **L** mức phân cấp khác nhau. Ta có chiều thứ **(i)** có **L_i** mức nên nhận $L_i + 1$ giá trị. Vì thế tổng số khối lập **n** chiều là
$$T = \prod_{i=1}^n (L_i + 1)$$
- **Tổng hợp** khối dữ liệu

- Chúng ta có thể tổng hợp mọi khối dữ liệu cơ bản (được gọi là tích hợp toàn bộ) hoặc không khống cơ bản nào (được gọi là không tích hợp) hoặc một vài khối cơ bản (được gọi là tích hợp một phần)
- Khi tổng hợp dữ liệu, chúng ta cần chọn lựa những khối cơ bản để tích hợp dựa trên kích cỡ của các khối, những phần giao nhau của các khối đó, tần suất truy nhập, v.v...

Các phép toán đối với khối dữ liệu

- Định nghĩa khái và tính toán trong DMQL

define cube sales [item, city, year]: sum(sales _in _dollars)

Phép toán trên định nghĩa một khái dữ liệu có tên là sales gồm 3 chiều item, city và year, khái dữ liệu này sẽ lưu trữ thông tin về số tiền bán được tính theo đơn vị đôla của từng mặt hàng (item), từng thành phố (city) và của mỗi năm (year) dựa trên những khái dữ liệu ở mức chi tiết hơn (hay mức thấp hơn) ví dụ như số tiền bán được của tivi Sony bán tại thành phố New York của năm 2011 cho các loại khách hàng (ở đây số chiều của khái dữ liệu thấp hơn là 4, thêm chiều khách hàng)

compute cube sales

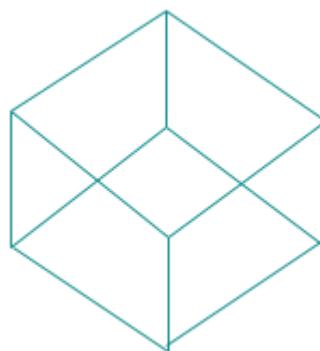
Phép toán trên tính toán các bản ghi cho khái dữ liệu tên Sales đó.

- Các phép toán trên có thể được biểu diễn thông qua ngôn ngữ dạng SQL, chúng ta biến đổi chúng thành một dạng ngôn ngữ giống như SQL với một toán tử mới **cube by** được giới thiệu bởi nhóm nghiên cứu Gray vào năm 1996) như sau:

SELECT item, city, year, **SUM** (amount)

FROM SALES

CUBE BY item, city, year



- Cân tính các khói dữ liệu ở mức cao hơn tức là số chiều giảm đi ví dụ như tính số tiền bán được của các nhóm sau
 - o 3 chiều (*date, product, customer*): *của mỗi ngày, mỗi sản phẩm, mỗi khách hàng* từ các khói dữ liệu 4 chiều (*date, product, customer, location*)
 - o 2 chiều (*date, product*): *của mỗi ngày, mỗi sản phẩm* từ các khói dữ liệu 3 chiều (*date, product, customer*). tương đương với câu lệnh *Group-by* sau


```
SELECT item, city, year, SUM(amount)  
FROM SALES  
GROUP BY item, year
```

 - o 2 chiều (*date, customer*): *của mỗi ngày và mỗi khách hàng*
 - o chiều (*product, customer*): *của mỗi sản phẩm và mỗi khách hàng*
 - o 1 chiều (*date*): *của mỗi ngày tính từ các khói dữ liệu 2 chiều có chiều date*
 - o 1 chiều (*product*) *của mỗi sản phẩm* tính từ các khói dữ liệu 2 chiều có chiều *product*
 - o 1 chiều (*customer*): *của mỗi khách hàng* tính từ các khói dữ liệu 2 chiều có *customer*
 - o 0 chiều () *thể hiện tổng số tiền bán được của tất cả có thể tính từ khói dữ liệu 1 chiều*
- Các thuật toán tính toán khói dữ liệu dựa trên ROLAP được sử dụng để
 - o Đánh địa chỉ dựa trên khóa
 - o Các phép sắp xếp, băm, và gộp nhóm được áp dụng tới các thuộc tính chiều để sắp xếp lại trật tự hoặc gộp nhóm các bộ có liên quan tới nhau theo một tiêu chí nào đó
 - o Các tích hợp dữ liệu có thể được tính toán từ những tích hợp được tính trước đó hơn là được tính từ dữ liệu trong những bảng Fact cơ bản
- Các thuật toán tính toán khói dữ liệu dựa trên MOLAP được sử dụng để
 - o Đánh địa chỉ mảng trực tiếp
 - o Phân mảng các mảng thành các khói dữ liệu nhỏ (chunk) mà vừa vặn với bộ nhớ để đỡ tốn chi phí truy nhập bộ nhớ nhiều lần.
 - o Tính toán các tích hợp dữ liệu bằng cách duyệt các khói dữ liệu con (vừa được phân mảng ở bước trên) của khói dữ liệu cần tính.

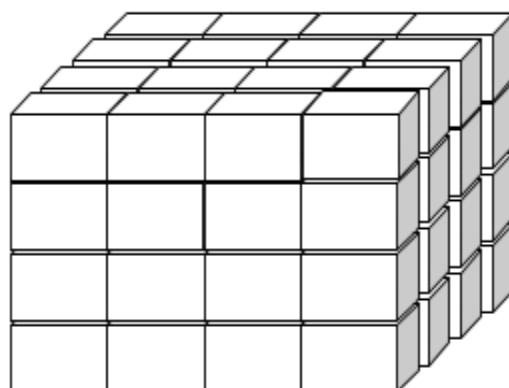
- Cần xác định một trật tự có thể duyệt các khối dữ liệu (chunk) cho một tính toán nhanh hơn và tối ưu hơn.

Mô tả nguyên tắc việc tích hợp nhiều chiều dữ liệu trong MOLAP để tham khảo

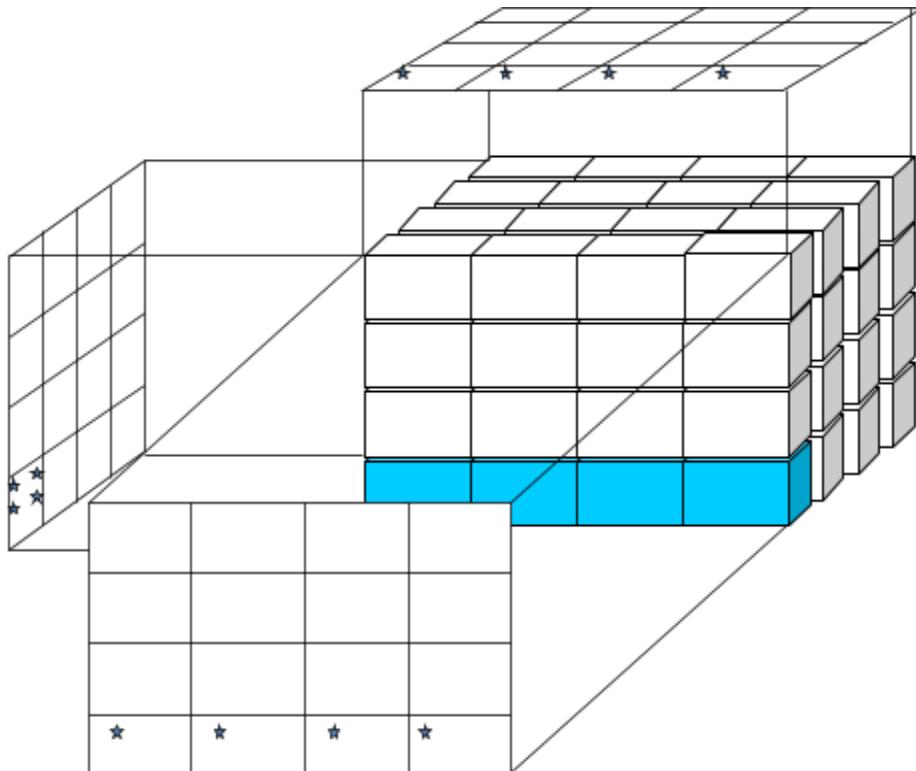
- Ta biết rằng trong MOLAP, dữ liệu được lưu trữ dưới dạng mảng nhiều chiều. Bước đầu tiên ta cần phân mảnh các cấu trúc mảng thành các khối dữ liệu con như đã nói đến ở trên.
- Đánh địa chỉ cho mảng thưa dữ liệu đã được nén theo dạng (chunk_id, offset)
 - Chunk_id: mã của khối con dữ liệu
 - Offset: vị trí lệch với đầu của khối con dữ liệu
- Tính toán các tích hợp trong khối đa chiều bằng cách duyệt các ô lướt (khối dữ liệu nhỏ nhất trong một khối dữ liệu nhiều chiều) trong khối dữ liệu theo một trật tự sao cho tối thiểu hóa số lần thăm viếng của mỗi ô lướt, để giảm thiểu việc truy nhập bộ nhớ và chi phí lưu trữ những giá trị tính toán trung gian.

Xét một khối dữ liệu như hình vẽ dưới đây, trong đó có 3 chiều dữ liệu A, B và C. Chiều A nhận các giá trị a0, a1, a2, a3, chiều B nhận các giá trị b0, b1, b2, b3 và chiều C nhận các giá trị c0, c1, c2, c3. Mỗi ô lướt trong khối dữ liệu này nhận một giá trị lưu trữ của bảng Fact 3 chiều. Vẫn đề là phải xác định được một trật tự thăm viếng tối ưu để tích hợp dữ liệu được theo nhiều chiều cho các giá trị tổng hợp khác nhau phục vụ người phân tích trực tuyến mà lại giảm thiểu được số lần viếng thăm của mỗi ô lướt để giảm thiểu phép toán và giảm thiểu không gian lưu trữ trung gian.

Trên hình vẽ giá trị số trong mỗi ô là trật tự được viếng thăm của chúng trong quá trình tích hợp dữ liệu các mức cao hơn để đảm bảo tối ưu như trình bày trên.



Sau khi duyệt qua các ô {1,2,3,4} ta thấy khôi con b0c0 được tổng hợp, a0c0 và a0b0 chưa được tính toán tổng hợp.



Sau khi duyệt 1-13 khôi a0c0 và b0c0 được tổng hợp, a0b0 vẫn chưa được tổng hợp (ta sẽ cần duyệt 1-49 để thực hiện việc này)

- Phương pháp: các mặt phẳng phải được sắp xếp và tính toán dựa trên kích cỡ của chúng theo trật tự tăng dần
 - o Trật tự duyệt như trong ví dụ trên là tối ưu nếu như lực lượng của C > lực lượng của B > lực lượng của A
- Tính toán khôi dữ liệu theo MOLAP nhanh hơn ROLAP
- Hạn chế của MOLAP là chỉ tính toán tốt chỉ với một số lượng chiều ít
- Nếu số chiều lớn, sử dụng cách tính toán cho khôi dữ liệu mà trong đó chỉ xử lý những khôi con dữ liệu dày đặc.

Đánh chỉ mục và xử lý truy vấn phân tích trực tuyến

Đánh chỉ mục dữ liệu OLAP dùng chỉ mục dạng Bitmap

- Thích hợp cho các tên miền với lực lượng (số lượng giá trị có thể nhận của thuộc tính đó) thấp
- Đánh chỉ số trên một cột cụ thể

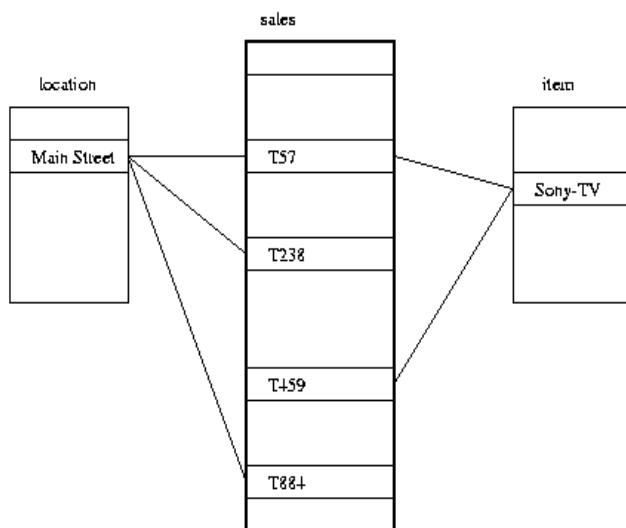
- Mỗi giá trị trong cột là một vectơ dạng bit có ưu điểm là các phép toán trên bit rất nhanh
- Độ dài của vecto bit được xác định bằng số lượng của bản ghi trong bảng cơ sở
- Cột bit thứ i được thiết lập (nhận giá trị là 1) nếu hàng thứ i của bảng cơ sở có giá trị cho cột được lập chỉ mục

Xem ví dụ sau đây

Cust	Region	Type	RecID	Asia	Europe	America	RecID	Retail	Dealer
C1	Asia	Retail	1	1	0	0	1	1	0
C2	Europe	Dealer	2	0	1	0	2	0	1
C3	Asia	Dealer	3	1	0	0	3	0	1
C4	America	Retail	4	0	0	1	4	1	0
C5	Europe	Dealer	5	0	1	0	5	0	1

Đánh chỉ mục dữ liệu OLAP dạng chỉ mục kết nối

- Chỉ mục kết nối có ích cho việc kết nối trong mô hình quan hệ và tăng tốc độ kết nối giữa các bảng, trong khi phép toán kết nối có một chi phí khá lớn.
- Trong kho dữ liệu, chỉ mục kết nối liên hệ các giá trị của các chiều trong một lược đồ hình sao tới các bản ghi tương ứng trong bảng Fact. Ví dụ: bảng Fact tên Sales có hai



chiều là location (vị trí) và item (các mặt hàng)

- Một chỉ mục kết nối trên bảng location là một danh sách các cặp $\langle \text{loc_name}, T_{\text{id}} \rangle$ được sắp xếp bởi location, trong đó T_{id} là định danh của bản ghi có chứa loc_name trong bảng Sales
 - Một chỉ mục kết nối trên location-và-item là một danh sách bộ ba $\langle \text{loc_name}, \text{item_name}, T_{\text{id}} \rangle$ được sắp xếp bởi location (vị trí) và item name (mặt hàng)
- Tìm kiếm trên một chỉ mục kết nối có thể vẫn lâu vì thế phải dùng kết hợp thêm một cách khác
 - Chỉ mục kết nối dạng bitmap cho phép tăng tốc việc tìm kiếm bằng cách sử dụng các vecto bit thay vì tên thuộc tính các chiều.

Tích hợp trực tuyến

Xem xét một truy vấn tích hợp sau đối với ví dụ được trình bày ở trên: "hãy tìm trung bình lượng hàng bán được của mỗi bang trong nước Mỹ"

Liệu chúng ta có thể cung cấp cho người sử dụng một số thông tin trước khi số lượng hàng trung bình chính xác được tính toán cho tất cả các bang không?

- Giải pháp: có thể đưa ra con số trung bình hiện tại cho mỗi bang khi việc tính toán đang tiến hành
- Thậm chí tốt hơn nữa, nếu chúng ta sử dụng các kỹ thuật thống kê và các bộ mẫu để tích hợp thay vì đơn giản là duyệt bảng cần tích hợp, cũng có thể cung cấp các giá trị biên ví dụ như "giá trị trung bình cho bang Wisconsin là 2000 ± 102 với xác suất 95%.

Xử lý các truy vấn OLAP một cách hiệu quả

Chúng ta cần làm những thao tác sau

- Xác định phép toán nào cần được thực hiện trên các khối sẵn có: chuyển đổi các phép toán khoan sâu xuống, cuộn lên v.v... thành các câu lệnh SQL tương ứng cùng với các phép toán OLAP ví dụ dice=selection + projection
- Xác định các phép toán liên quan cần được áp dụng tới những khối dữ liệu con đã được tích hợp nào.
- Xem xét các cấu trúc chỉ mục và các cấu trúc mảng nén hay dày đặc trong MOLAP (sẽ có sự đánh đổi giữa việc lập chỉ mục với chi phí không gian lưu trữ)

Kho siêu dữ liệu

- Siêu dữ liệu là các đối tượng dữ liệu để định nghĩa kho dữ liệu. Nó bao gồm các loại sau:

- o Mô tả các cấu trúc của kho dữ liệu như lược đồ dữ liệu, các khung nhìn, các chiều dữ liệu, các phân cấp, các định nghĩa dữ liệu phái sinh, các vị trí và nội dung của kho dữ liệu theo chủ đề
- o Các siêu dữ liệu thao tác bao gồm nguồn gốc lịch sử phát triển dữ liệu như dữ liệu được đổ vào từ đâu, như thế nào, các đường dẫn chuyển đổi ra sao hay còn gọi là gia phả dữ liệu.Thêm nữa, loại siêu dữ liệu này còn có thông tin về trạng thái hiện tại (hoạt động, được lưu trữ hay cần được loại bỏ), các thông tin theo dõi (các thông kê sử dụng kho dữ liệu, các báo cáo lỗi, các vết kiểm tra tính xác thực của dữ liệu).
- o Các thuật toán được sử dụng để tổng hợp dữ liệu
- o Việc ánh xạ dữ liệu từ môi trường tác nghiệp sang kho dữ liệu
- o Các dữ liệu liên quan tới hiệu năng của hệ thống như lược đồ kho dữ liệu, các khung nhìn và các định nghĩa dữ liệu được phái sinh
- o Các dữ liệu về hệ thống đang quản lý như các thuật ngữ chuyên môn và các định nghĩa, quyền sở hữu dữ liệu, các chính sách phân quyền

Các công cụ và tiện ích đầu cuối của kho dữ liệu

- Trích lọc dữ liệu: dùng để lấy dữ liệu từ nhiều nguồn dữ liệu hỗn tạp bên ngoài
- Làm sạch dữ liệu: dùng để phát hiện lỗi trong dữ liệu và sửa chữa chúng khi có thể
- Chuyển đổi dạng dữ liệu: dùng để chuyển đổi dữ liệu từ dạng ngữ nghĩa hoặc của máy khách tới dạng của kho dữ liệu.
- Tải dữ liệu: dùng để sắp xếp, tổng hợp, cung cấp và tính toán các khung nhìn, kiểm tra tính toàn vẹn dữ liệu và xây dựng các tệp chỉ mục và tệp phân mảnh.
- Làm mới dữ liệu: dùng để truyền tải những cập nhật từ nguồn dữ liệu tới kho dữ liệu

3.5 Liên hệ công nghệ kho dữ liệu với khai phá dữ liệu

Cách sử dụng kho dữ liệu

- Ba loại ứng dụng kho dữ liệu
 - o Xử lý thông tin: hỗ trợ việc truy vấn thông tin, phân tích thống kê cơ bản và làm báo cáo sử dụng các bảng tham chiếu chéo, các bảng, các biểu đồ và đồ thị
 - o Xử lý phân tích: dùng cho phân tích đa chiều của kho dữ liệu, hỗ trợ các thao tác OLAP cơ bản, cắt ngang, cắt dọc, khoan sâu, xoay

o Khai phá dữ liệu bao gồm:

- Phát hiện tri thức từ mô hình ẩn
- Hỗ trợ luật kết hợp, xây dựng các mô hình phân tích, thực hiện phân loại và dự đoán, trình bày các kết quả khai phá sử dụng các công cụ trực quan

Ba nhiệm vụ này khác nhau về bản chất và ý nghĩa ứng dụng, một cho truy vấn báo cáo đơn giản, một cho phân tích để đưa ra các kết quả tổng hợp, cái cuối cùng để dành cho việc phát hiện tri thức tiềm ẩn trong dữ liệu

Từ xử lý phân tích trực tuyến (OLAP) tới khai phá phân tích trực tuyến (OLAM)

- Tại sao cần khai phá phân tích trực tuyến
 - o Vì dữ liệu trong kho dữ liệu được lưu trữ với chất lượng cao do chứa những dữ liệu đã được làm sạch, đồng nhất và tích hợp
 - o Vì cấu trúc xử lý thông tin sẵn có xung quanh các kho dữ liệu như ODBC (kết nối dữ liệu), OLEDB (những cơ sở dữ liệu), truy nhập Web, các dịch vụ tiện tích, các công cụ OLAP và báo cáo.
 - o Vì cần phân tích dữ liệu thăm dò dựa trên OLAP: có thể khai phá với các phép toán khoan sâu, cắt lát, xoay, v.v...
 - o Lựa chọn trực tuyến các chức năng khai phá dữ liệu: tích hợp và hoán đổi nhiều chức năng khai thác khác nhau, các thuật toán và nhiệm vụ khác nhau.
- Kiến trúc của OLAM: được trình bày trong bài 1

3.6 Xây dựng kho dữ liệu với mục đích hỗ trợ quyết định (DSS)

Nhắc lại một chút về khái niệm kho dữ liệu và những tác nhân liên quan.

Một cơ sở dữ liệu là tập hợp các dữ liệu được tổ chức bởi một hệ thống quản trị cơ sở dữ liệu. Một kho dữ liệu là một cơ sở dữ liệu phân tích chỉ đọc được sử dụng để vận hành một hệ thống hỗ trợ quyết định.

Một kho dữ liệu để hỗ trợ quyết định thường được lấy dữ liệu từ các nền tảng khác nhau, cơ sở dữ liệu, và các tập tin như dữ liệu nguồn. Việc sử dụng các công cụ tiên tiến và công nghệ chuyên ngành có thể cần thiết trong sự phát triển của các hệ thống hỗ trợ quyết định, thứ ảnh hưởng đến nhiệm vụ, các sản phẩm phân phối, việc đào tạo, và các thời hạn của dự án.

Một kho dữ liệu cần có sự thân thiện và sẵn sàng được tạo ra bởi các nhà phân tích cho người dùng cuối, ngay cả những người không quen thuộc với cấu trúc cơ sở dữ liệu.

Kho dữ liệu là một tập hợp các cơ sở dữ liệu đã được chuẩn hóa tích hợp với nhau cho hiệu năng đáp ứng nhanh.

Nói chung, một kho dữ liệu phải được dự trù cho sự tăng trưởng về dữ liệu dài hạn ít nhất 5 năm. Trong bài này, chúng ta cùng bàn luận đến các giai đoạn xây dựng một hệ thống kho dữ liệu trên quan điểm của một hệ trợ giúp quyết định

Các giai đoạn xây dựng

1. Lập kế hoạch
2. Thu thập yêu cầu về dữ liệu và mô hình hóa
3. Thiết kế và Phát triển cơ sở dữ liệu vật lý
4. Dữ liệu bản đồ và sự biến đổi
5. Khai thác dữ liệu và tải
6. Tự động hóa việc Quy trình quản lý dữ liệu
7. Phát triển ứng dụng-Tạo bộ starter của báo cáo
8. Xác Nhận và kiểm tra dữ liệu
9. Đào tạo
10. Triển khai

Pha 1: Lập kế hoạch cho một kho dữ liệu có liên quan tới

- Xác định phạm vi dự án
- Tạo ra kế hoạch dự án
- Xác định các nguồn lực cần thiết, cả trong và ngoài

- Xác định nhiệm vụ và các sản phẩm phân phối
- Xác định thời hạn của dự án
- Xác định sản phẩm phân phối cuối cùng của dự án

Lập kế hoạch về hiệu năng của dự án cần thực hiện những công việc sau đây

- Tính toán kích cỡ bản ghi cho mỗi bảng.
- Ước tính số lượng bản ghi ban đầu cho mỗi bảng
- Xem lại các yêu cầu truy cập kho dữ liệu để dự đoán yêu cầu về tệp chỉ mục
- Xác định các yếu tố tăng trưởng cho mỗi bảng
- Xác định bảng mục tiêu lớn nhất dự kiến trong một giai đoạn thời gian được lựa chọn và thêm khoảng 25-30% dự trù tới kích thước bảng để xác định kích thước lưu trữ tạm thời

Pha 2: Thu thập các yêu cầu dữ liệu và mô hình hóa

Để thu thập các yêu cầu về dữ liệu, cần phải trả lời những câu hỏi sau:

- Người sử dụng thực hiện các công việc nghiệp vụ như thế nào?
- Hiệu suất của người dùng được đo như thế nào?
- Những thuộc tính nào người sử dụng cần?
- Các phân cấp trong nghiệp vụ kinh doanh của hệ thống là gì?
- Những dữ liệu nào người dùng hiện nay đang sử dụng và họ muốn có dữ liệu nào trong tương lai?
- Người dùng cần dữ liệu tổng hợp hay chi tiết ở mức độ nào?

Mô hình hóa dữ liệu: chọn một trong hai loại mô hình hóa dưới đây

- Một mô hình dữ liệu logic bao phủ phạm vi của dự án phát triển bao gồm các mối quan hệ, loại liên kết giữa các quan hệ, các thuộc tính, và các khóa ứng cử viên (candidate keys).
- Một mô hình nghiệp vụ nhiều chiều được thể hiện qua các bảng Fact, các chiều, các phân cấp, các mối quan hệ và các khóa ứng cử viên cho các phạm vi phát triển của dự án.

Pha 3: Thiết kế và Phát triển cơ sở dữ liệu vật lý

- Thiết kế cơ sở dữ liệu, bao gồm các bảng Fact, các bảng quan hệ, và các bảng mô tả (dùng để tra cứu).
- Phi chuẩn dữ liệu
- Xác định các khóa
- Tạo các chiến lược lập chỉ mục

- Tạo các đối tượng cơ sở dữ liệu thích hợp.

Pha 4: Ánh xạ và chuyển đổi dữ liệu

- Xác định hệ thống nguồn.
- Xác định cách bố trí tập tin.
- Phát triển các yêu cầu chi tiết kỹ thuật chuyển đổi bằng văn bản cho các biến đổi phức tạp
- Ánh xạ nguồn tới dữ liệu đích
- Xem xét lại các kế hoạch về hiệu năng

Pha 5: Hình thành kho dữ liệu

- Phát triển các thủ tục để trích xuất và di chuyển dữ liệu vào kho.
- Phát triển các thủ tục để nạp dữ liệu vào kho.
- Phát triển chương trình phần mềm hoặc dùng các công cụ chuyển đổi dữ liệu để chuyển đổi và tích hợp dữ liệu.
- Kiểm thử việc trích xuất, chuyển đổi và các thủ tục tải dữ liệu

Pha 6: Thủ tục quản lý dữ liệu tự động

- Tự động hóa và lập lịch cho quá trình tải dữ liệu.
- Tạo sao lưu dữ liệu và các thủ tục phục hồi.
- Tiến hành một thử nghiệm đầy đủ của tất cả các thủ tục tự động.

Pha 7: Phát triển ứng dụng – tạo ra một tập khởi đầu cho các báo cáo

- Tạo tập khởi đầu cho các báo cáo được định trước.
- Phát triển các báo cáo cơ bản quan trọng
- Kiểm thử tính đúng đắn của các báo cáo
- Viết tài liệu cho ứng dụng
- Phát triển các đường dẫn để điều hướng

Pha 8: Xác nhận và kiểm thử dữ liệu

- Xác nhận dữ liệu bằng cách sử dụng tập khởi đầu cho các báo cáo.
- Xác nhận dữ liệu bằng cách sử dụng các quy trình chuẩn
- Lặp đi lặp lại thay đổi dữ liệu

Pha 9: Đào tạo

Để đạt được giá trị kinh doanh thực tế từ việc phát triển kho dữ liệu, những người dùng của tất cả các cấp sẽ cần phải được đào tạo về:

- Phạm vi của dữ liệu trong kho
- Công cụ truy nhập đầu cuối và cách thức hoạt động nó.
- Việc ứng dụng các DDS hoặc tập khởi tạo các báo cáo bao gồm cả các khả năng ứng dụng và đường dẫn chuyển hướng.
- Liên tục đào tạo và hỗ trợ người sử dụng khi hệ thống thay đổi

Pha 10: Triển khai

- Cài đặt cơ sở hạ tầng vật lý cho tất cả người dùng
- Phát triển ứng dụng DDS
- Tạo thủ tục cho việc thêm các báo cáo mới và mở rộng việc áp dụng DSS
- Thiết lập các thủ tục để sao lưu các ứng dụng DSS, không phải chỉ là kho dữ liệu
- Tạo thủ tục điều tra và giải quyết các vấn đề liên quan tới toàn vẹn dữ liệu

Thiết kế cơ sở dữ liệu với lược đồ hình sao

Các mục tiêu của một cơ sở dữ liệu hỗ trợ quyết định thường được thực hiện bằng một thiết kế cơ sở dữ liệu gọi là một lược đồ sao. Một thiết kế lược đồ sao là một cấu trúc đơn giản với một vài bảng quan hệ và các đường dẫn kết nối giữa chúng được xác định rõ ràng. Thiết kế cơ sở dữ liệu này, trái ngược với cấu trúc được chuẩn hóa, cái được sử dụng cho cơ sở dữ liệu xử lý giao dịch, cung cấp thời gian đáp ứng các truy vấn nhanh chóng và một lược đồ đơn giản dễ hiểu đối với các nhà phân tích và người dùng cuối.

Các bảng sự kiện (Fact) và bảng theo chiều (dimension)

Một lược đồ hình sao bao gồm có hai kiểu bảng, các bảng sự kiện và bảng theo chiều. Các **bảng sự kiện** chứa các dữ liệu định lượng hoặc thực tế về một doanh nghiệp - thông tin được truy vấn đến. Thông tin này thường là các con số đo lường và có thể bao gồm nhiều cột và hàng triệu dòng. Bảng theo chiều nhỏ hơn và giữ dữ liệu mô tả phản ánh các yếu tố (chiều) ảnh hưởng tới một doanh nghiệp. Truy vấn SQL sau đó sẽ sử dụng các đường dẫn kết nối định sẵn và do người dùng xác định giữa bảng Fact và bảng theo chiều để trả về thông tin được lựa chọn.

Để xác định các bảng Fact và bảng theo chiều ta cần thực hiện những công việc như sau:

- Tìm các giao dịch cơ bản trong quá trình kinh doanh. Việc này sẽ xác định các thực thể là ứng cử viên trong bảng Fact
- Xác định các chiều chính cho từng bảng Fact. Điều này xác định các thực thể là những ứng viên cho các bảng theo chiều.

- Kiểm tra một ứng cử bảng Fact không thực sự là một bảng theo chiều có nhúng bảng Fact.
- Kiểm tra xem một ứng viên cho bảng theo chiều hướng không thực sự là một bảng Fact trong ngữ cảnh các yêu cầu của hỗ trợ quyết định.

Một ví dụ về thiết kế lược đồ cơ sở dữ liệu cho kho dữ liệu

Bước 1 Hãy tìm những giao dịch cơ bản trong quá trình kinh doanh

- Bước đầu tiên trong quá trình xác định các bảng sự kiện là nơi mà chúng ta kiểm tra việc kinh doanh, và xác định các giao dịch có thể được quan tâm. Chúng sẽ có xu hướng giao dịch trong đó mô tả các sự kiện cơ bản cho doanh nghiệp

Bước 2 Xác định kích thước chính áp dụng cho từng sự kiện

- Bước tiếp theo là xác định kích thước chính cho mỗi ứng cử viên bàn thực tế. Điều này có thể đạt được bằng cách nhìn vào mô hình hợp lý, và tìm hiểu các thực thể được liên kết với các tổ chức đại diện cho bảng thực tế. Thách thức ở đây là để tập trung vào các thực thể kích thước phím.

Bước 3 Kiểm tra xem một sự kiện không phải là ứng cử viên thực sự là một bảng kích thước với sự kiện được bình thường hóa

- Tìm chiều bình thường hóa trong các bảng sự kiện ứng viên. Nó có thể là trường hợp thực tế bảng ứng cử viên là một chiều hướng có chứa các nhóm lặp đi lặp lại các thuộc tính thực tế.

Bước 4 Kiểm tra ứng cử viên một chiều không phải là bảng thực tế

Nếu yêu cầu nghiệp vụ là hướng phân tích của đơn vị, đó là ứng cử viên thể hiện một chiều hướng, cơ hội được rằng nó có lẽ là thích hợp hơn để làm cho nó một bảng thực tế

Lược đồ sao đơn giản

Mỗi bảng phải có một khóa chính, mà là một cột hoặc một nhóm các cột có nội dung xác định duy nhất mỗi hàng. Trong một lược đồ sao đơn giản, các khóa chính cho bảng thực tế gồm một hoặc nhiều khóa ngoại. Khi cơ sở dữ liệu được tạo ra, các báo cáo SQL được sử dụng để tạo ra các bảng sẽ chỉ định các cột được để tạo thành khóa chính và khóa ngoại.

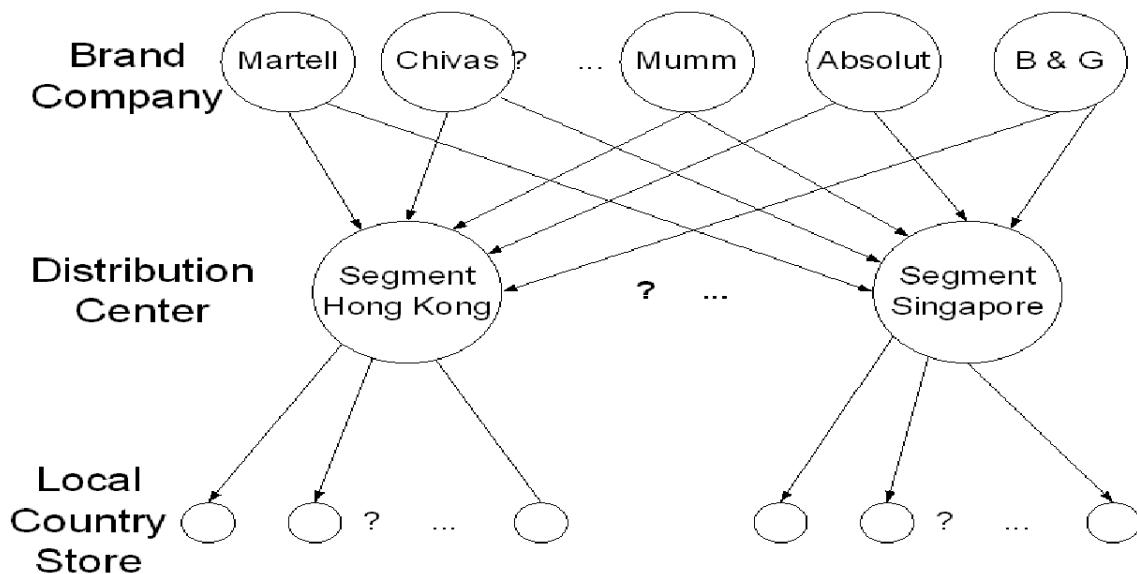
Lược đồ bông tuyết

Lược đồ bông tuyết là một lược đồ sao trong đó lưu trữ tất cả các chiều thông tin trong ba bảng mẫu thông thường, trong khi thực tế vẫn giữ các bảng cấu trúc giống nhau.

Nghiên cứu xây dựng một kho dữ liệu

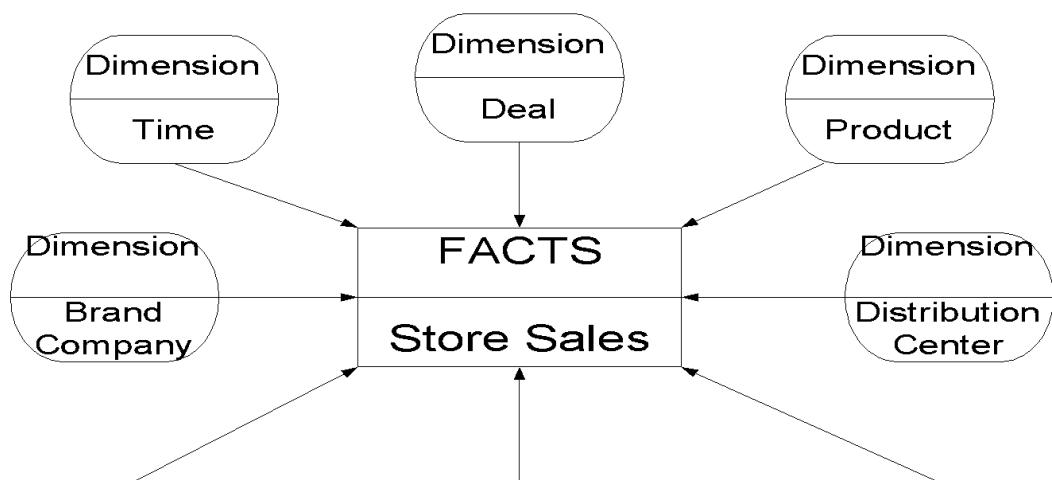
Bước 1: Lập kế hoạch

Segram Company Supply and Demand Chain



Năng suất kế hoạch

- Với kích thước thời gian: 2 năm x 365 ngày
- Kích thước sản phẩm: 5 sản phẩm trung bình cho mỗi giao dịch
- Kích thước khuyến mại: 1 kiểu xúc tiến cho mỗi loại giao dịch
- Kích thước của cửa hàng : 10 cửa hàng địa phương
- Kích thước của khách hàng: 1 khách hàng cho mỗi giao dịch
- Số lượng giao dịch bán hàng : 200 mỗi ngày cho khách hàng lớn
- Kết quả là, số lượng hồ sơ cơ sở thực tế = $2 \times 365 \times 5 \times 1 \times 200 = 7,3$ triệu hồ sơ
- Giả sử số lĩnh vực trọng điểm = 5, số trường thực tế = 7, tổng số lĩnh vực = 12
Do đó, kích thước thực tế cần cù bảng = $7,3$ triệu x 12 x 4 byte cho mỗi lĩnh vực = 350 MB (kích thước của kích thước bảng là không đáng kể).



Bước 2: Yêu cầu về dữ liệu và mô hình hóa

Bước 3 Thiết kế và phát triển cơ sở dữ liệu vật lý

Ví dụ: Thiết kế một lược đồ sao đơn giản từ một lược đồ quan hệ

- Xác định các lĩnh vực đo lường trong một bảng thực tế.
- Xác định tiêu chí lựa chọn của các đo lường như là chìa khóa trong một bảng thực tế.
- Xây dựng các bảng kích thước bắt nguồn từ các phím trong bảng thực tế.

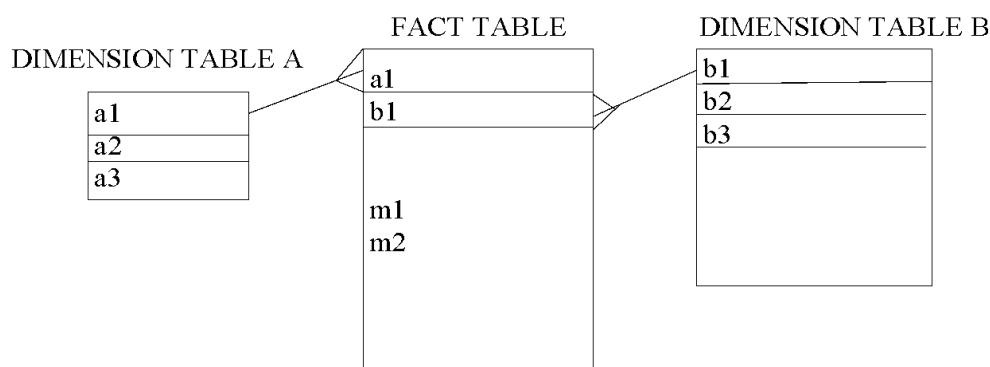
Xác nhận lược đồ sao đơn giản như SR1 loại liên quan.

Cho Quan hệ A (a₁, a₂, a₃)

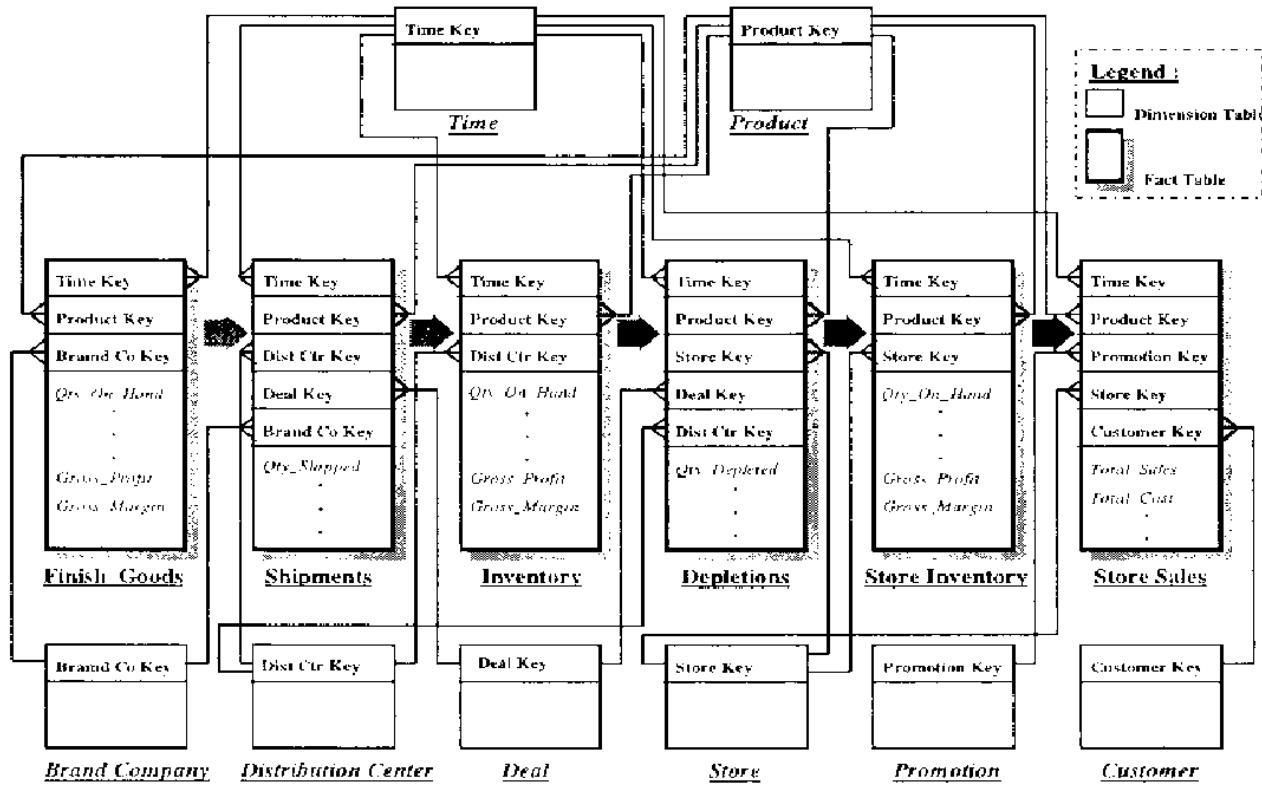
Quan hệ B (b₁, b₂, b₃)

Quan hệ C (*a₁, *b₁, m₁, m₂)

Được suy ra từ lược đồ sao đơn giản



Supply & Demand Chain Star Schema



Bước 4: Ánh xạ mô hình tác nghiệp thành một kho dữ liệu

Bước 5: Trích lọc dữ liệu và tải vào kho

Hãy tàng kỹ thuật phải được thực hiện để hỗ trợ với các giai đoạn trung gian cho việc ánh xạ dữ liệu, chuyển đổi, trích xuất và tải bao gồm:

1. Tri thức quản trị cơ sở dữ liệu
2. Tri thức và huấn luyện công cụ chuyển đổi dữ liệu
3. Các chiến lược cập nhật / làm mới
4. Chiến lược tải
5. Các hoạt động / lập lịch trình cho các công việc
6. Các thủ tục bảo đảm chất lượng
7. Tri thức lập kế hoạch năng lực

Bước 6: Tự động hóa quá trình quản lý dữ liệu

Một kho dữ liệu sử dụng hai chế độ hoạt động. Hầu hết các kho dữ liệu được sử dụng trực tuyến 16-22 giờ mỗi ngày trong một chế độ chỉ đọc. Kho dữ liệu không trực tuyến (off-line) cho 2-8 giờ trong một khoảng nhỏ của buổi sáng để tải dữ liệu, lập chỉ mục, đảm bảo chất lượng dữ liệu, và giải phóng dữ liệu.

Bước 7: Phát triển ứng dụng- Tạo những bộ báo cáo khởi tạo

Các báo cáo cho các hệ thống thông tin thực thi ví dụ như:

- Có thực sự cần lưu trữ nhiều kích cỡ khác nhau của các loại sản phẩm nào đó?
- Công ty cạnh tranh đang bán 10 mặt hàng phổ biến nhất nào mà chúng ta không bán?
- Mùa nào bán được nhiều rượu Cognac nhất trong năm ngoái?
- Khách hàng/cửa hàng nào mua nhiều hàng giảm giá nhất trong năm 2011?
- Những sản phẩm nào mang lại lợi nhuận nhiều nhất trong năm 2011 ở Hà nội?
- Lợi nhuận tổng trong tháng 4 năm nay là bao nhiêu?

Câu hỏi ôn tập chương 3

3.1 Hãy giải thích tại sao, để tích hợp nhiều nguồn thông tin không đồng nhất, nhiều công ty trong ngành thích phương pháp tiếp cận hướng cập nhật (xây dựng và sử dụng kho dữ liệu), thay vì cách tiếp cận hướng truy vấn (áp dụng các chương trình bao bọc và tích hợp). Mô tả các tình huống mà cách tiếp cận hướng truy vấn thích hợp hơn cách tiếp cận hướng cập nhật.

3.2 So sánh ngắn gọn các khái niệm sau. Bạn có thể sử dụng một ví dụ để giải thích (các) quan điểm.

- (a) Lược đồ bông tuyết, chòm sao thực tế, mô hình truy vấn hình sao
- (b) Làm sạch dữ liệu, chuyển đổi dữ liệu, làm mới dữ liệu
- (c) Kho dữ liệu toàn doanh nghiệp, kho dữ liệu chủ đề, kho dữ liệu ảo

3.3 Kho dữ liệu có thể được mô hình hóa bằng lược đồ hình sao hoặc lược đồ bông tuyết. Mô tả ngắn gọn các điểm tương đồng và sự khác biệt của hai mô hình, và sau đó phân tích lợi thế và bất lợi của chúng. Cho ý kiến của bạn về việc mô hình nào có thể hữu ích hơn về mặt thực nghiệm và nêu rõ lý do đằng sau câu trả lời của bạn.

3.4 Việc cài đặt một kho dữ liệu phô biến là xây dựng một cơ sở dữ liệu đa chiều, được gọi là khối dữ liệu. Thật không may, điều này thường có thể tạo ra một ma trận đa chiều rất lớn, nhưng rất thưa thớt. Trình bày ví dụ minh họa một khối dữ liệu khổng lồ và thưa thớt như vậy.

3.5 Trong công nghệ kho dữ liệu, một khung nhìn đa chiều có thể được thực hiện bởi một kỹ thuật cơ sở dữ liệu quan hệ (ROLAP), hoặc bằng một kỹ thuật cơ sở dữ liệu đa chiều (MOLAP), hoặc bởi một kỹ thuật cơ sở dữ liệu lai (HOLAP).

- (a) Mô tả ngắn gọn từng kỹ thuật cài đặt này.
- (b) Đối với mỗi kỹ thuật, giải thích cách thức của từng chức năng sau đây có thể được triển khai:
 - i. Tạo ra một kho dữ liệu (bao gồm cả việc tổng hợp)
 - ii. Cuộn lên
 - iii. Khoan xuống
 - iv. Cập nhật gia tăng

Bạn thích kỹ thuật triển khai nào hơn và tại sao?

3.6 Giả sử một kho dữ liệu chứa 20 chiều, mỗi chiều có khoảng năm mức phân cấp

- (a) Người dùng chủ yếu quan tâm đến bốn chiều cụ thể, mỗi chiều có ba mức truy cập thường xuyên để cuộn lên và khoan xuống. Bạn sẽ thiết kế như thế nào một cấu trúc khối dữ liệu để hỗ trợ hiệu quả tùy chọn này?
- (b) Đôi khi, một người dùng có thể muốn khoan xuyên qua khối lập phương, xuống đến dữ liệu thô cho một hoặc hai chiều cụ thể. Bạn sẽ hỗ trợ tính năng này như thế nào?

3.7 Sự khác biệt giữa ba loại hình chính của việc sử dụng kho dữ liệu là gì: xử lý thông tin, xử lý phân tích và khai phá dữ liệu? Thảo luận về động cơ đằng sau việc khai phá phân tích trực tuyến OLAP (OLAM)

3.8 Trình bày ba phương pháp để cài đặt một câu lệnh xử lý phân tích trực tuyến. Hãy đưa ra một ví dụ cho mỗi phương pháp đó.

Chương 4: Khai phá dữ liệu

4.1 Tiền xử lý dữ liệu trước khi khai phá

Khái niệm về dữ liệu

- Dữ liệu là tập hợp các đối tượng và thuộc tính của nó.
- Một thuộc tính là một tính chất hoặc một đặc điểm của đối tượng.
 - Ví dụ: màu mắt của một người, nhiệt độ...
 - Thuộc tính cũng được gọi là biến, trường, đặc tính, đặc điểm.
- Một tập các thuộc tính mô tả một đối tượng.
 - Đối tượng cũng được gọi là bản ghi (record), điểm (point), trường hợp (case), mẫu (sample), thực thể (entity) hoặc một thể hiện (instance)

Ví dụ: bảng sau đây cung cấp một dữ liệu gồm 10 đối tượng tương ứng với 10 hàng trong bảng, và 5 thuộc tính tương ứng với 5 cột trong bảng

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Giá trị thuộc tính

- Giá trị thuộc tính là số hoặc ký hiệu được gán cho một thuộc tính.
- Phân biệt giữa thuộc tính và giá trị thuộc tính.
 - Thuộc tính giống nhau có thể ánh xạ đến các giá trị thuộc tính khác nhau. Ví dụ: chiều cao có thể đo bằng các đơn vị đo lường khác nhau như feet của Anh hoặc mét.

- Các thuộc tính khác nhau có thể được ánh xạ tới cùng một tập giá trị. Ví dụ: giá trị thuộc tính định danh ID và tuổi đều là số tự nhiên. Nhưng tính chất của giá trị thuộc tính có thể khác nhau ví dụ như ID không có giới hạn nhưng tuổi thì có giới hạn lớn nhất và nhỏ nhất.

Các loại kiểu thuộc tính

- Tên (Nominal) Ví dụ : số định danh ID, màu mắt, mã vùng (zip codes_
- Thứ tự (Ordinal) Ví dụ: xếp loại (e.g., vị của món khoai tây rán trong khoảng từ 1-10); cấp: chiều cao {cao, trung bình, thấp}
- Khoảng (Interval) Ví dụ: nhiệt độ theo độ C hoặc độ F (Fahrenheit)
- Tỉ lệ (Ratio) Ví dụ: độ Kelvin, chiều dài, thời gian..

Thuộc tính có nhiều loại khác nữa, bất kể một miền giá trị nào thỏa mãn những đặc tính dưới đây đều được coi là kiểu thuộc tính

Kiểu của một giá trị thuộc tính phụ thuộc vào một số tính chất

- Tính riêng biệt (Distinctness): thực hiện được các phép $= \neq$
- Tính thứ tự (Order): thực hiện được các phép $< >$
- Tính cộng (Addition): thực hiện được các phép $+ -$
- Tính nhân (Multiplication): thực hiện được các phép $* /$

Ví dụ:

- Thuộc tính tên (Nominal attribute) có tính riêng biệt
- Thuộc tính thứ tự có tính riêng biệt và tính thứ tự
- Thuộc tính khoảng có tính riêng biệt, tính thứ tự và tính cộng
- Thuộc tính tỉ lệ: cả 4 tính chất trên.

Bảng so sánh các kiểu thuộc tính thể hiện dưới đây

Các thuộc tính liên tục và rời rạc

- Thuộc tính rời rạc là các thuộc tính thỏa mãn những tiêu chí sau
 - Chỉ có một tập hữu hạn hoặc vô hạn đếm được các giá trị. Ví dụ: mã vùng, số đếm, hoặc tập các từ trong văn bản
 - Thường được biểu diễn bằng các biến nguyên
 - Lưu ý: Các thuộc tính nhị phân là trường hợp đặc biệt của thuộc tính rời rạc
- Thuộc tính liên tục là các thuộc tính thỏa mãn những tiêu chí sau
 - Giá trị thuộc tính là những số thực. Ví dụ: nhiệt độ, độ cao, cân nặng.
 - Thực tế, các giá trị thực chỉ có thể được đo và được biểu diễn bằng cách sử dụng một số hữu hạn các chữ số.
 - Thuộc tính liên tục thông thường được biểu diễn bằng các biến số thực.

Các kiểu tập dữ liệu

- Kiểu bản ghi (Record) bao gồm
 - Ma trận dữ liệu: dạng ma trận hai chiều
 - Dữ liệu văn bản
 - Dữ liệu giao dịch chứa thông tin về các giao dịch gồm các thuộc tính khác nhau
- Kiểu đa quan hệ (Multi-Relational) trong đó mỗi dữ liệu là dạng lược đồ hình sao hoặc bông tuyết (snowflake)

- Kiểu Đồ họa(Graph) trong đó mỗi dữ liệu là một trang Web trong hệ thống World Wide Web hoặc là một cấu trúc phân tử trong hóa sinh học
- Kiểu có thứ tự (Ordered) trong đó
 - Dữ liệu không gian: mỗi dữ liệu là thể hiện đặc điểm trong không gian
 - Dữ liệu thời gian: mỗi dữ liệu là thể hiện đặc điểm thời gian.
 - Dữ liệu tuần tự: thể hiện một chuỗi có thứ tự các đối tượng

Các đặc điểm quan trọng của dữ liệu có cấu trúc

- Đa chiều: số chiều của dữ liệu chính là số các thuộc tính mà mỗi đối tượng được mô tả.
 - Thách thức: quá nhiều chiều sẽ gây nhiều khó khăn
 - Tính thừa thót: Dữ liệu thừa thót là dữ liệu mà giá trị nhiều thuộc tính bằng 0. Thách thức: dữ liệu thừa thót yêu cầu xử lý đặc biệt
 - Dải phạm vi của thuộc tính
 - Các thuộc tính dữ liệu thường được đo bằng các dải giá trị khác nhau
 - Thách thức: Quyết định một dải tốt nhất là một công việc khó

Dạng dữ liệu bản ghi

- Dữ liệu bao gồm nhiều bản ghi, mỗi bản ghi lại chứa 1 tập các thuộc tính giống trong cơ sở dữ liệu quan hệ

Dạng ma trận dữ liệu

- Nếu các đối tượng dữ liệu có cùng một tập cố định số các thuộc tính, thì các đối tượng dữ liệu có thể được coi là điểm trong một không gian đa chiều, nơi mà mỗi chiều đại diện cho một thuộc tính khác biệt
- Tập dữ liệu như vậy có thể được đại diện bởi một ma trận $m \times n$, m hàng, mỗi hàng là một đối tượng, và n cột, mỗi cột là một thuộc tính
- Ví dụ về ma trận dữ liệu được thể hiện trong bảng sau trong đó có 2 hàng và 5 cột

Projection of x Load	Projection of y load	Distance	Load	Thickness
10.23	5.27	15.22	2.7	1.2
12.65	6.25	16.22	2.2	1.1

Dữ liệu văn bản

- Mỗi văn bản tài liệu sẽ trở thành một vectơ các thuật ngữ hay từ khóa

- Mỗi từ khóa là một thành phần (thuộc tính) của vecto,
- Giá trị của mỗi thành phần là số lần từ khóa tương ứng xuất hiện trong tài liệu

Ví dụ của dữ liệu loại này được thể hiện trong hình vẽ dưới đây trong đó có 3 tài liệu, mỗi tài liệu được biểu diễn bằng vecto 10 chiều, mỗi chiều ứng với một từ khóa, chứa số lần xuất hiện của mỗi từ khóa đó trong tài liệu tương ứng

	team	coach	pla	ball	score	game	wi	lost	timeout	season
Tài liệu 1	3	0	5	0	2	6	0	2	0	2
Tài liệu 2	0	7	0	2	1	0	0	3	0	0
Tài liệu 3	0	1	0	0	1	2	2	0	3	0

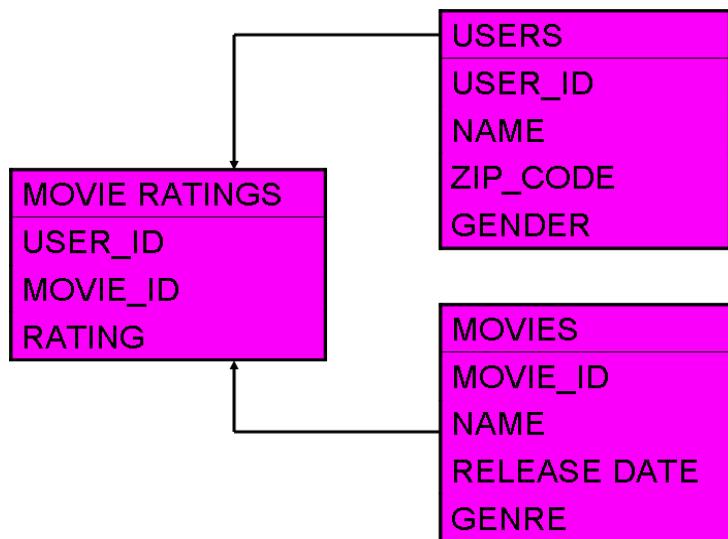
Dữ liệu giao dịch

- Là một kiểu dữ liệu bản ghi đặc biệt, mà mỗi bản ghi (giao dịch) bao gồm một tập các mục
- Ví dụ: Xét một cửa hàng tạp hóa trong đó bộ các sản phẩm mà khách hàng mua trong một lần đi mua sắm gọi là một giao dịch, mỗi sản phẩm gọi là một mặt hàng, thể hiện trong hình vẽ dưới đây, mỗi hàng là thông tin của một giao dịch trong đó cột 1 chỉ mã số giao dịch, cột 2 chứa các mặt hàng được mua trong giao dịch đó.

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

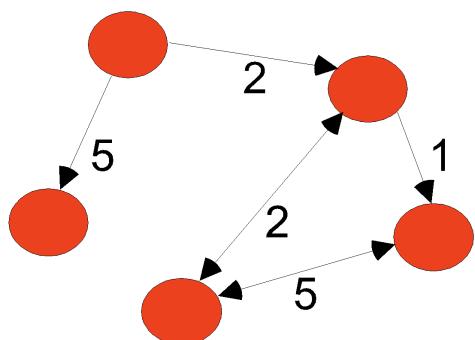
Dữ liệu đa quan hệ

Trong loại dữ liệu này, mỗi đối tượng chính là một quan hệ, các thuộc tính của mỗi đối tượng chính là quan hệ. Ví dụ như sau trong đó 3 bảng quan hệ và 2 mối quan hệ giữa chúng thể hiện một dữ liệu đa quan hệ, thuộc tính của đối tượng này là 3 quan hệ và 2 mối quan hệ thể hiện bằng hai mũi tên đen trong hình vẽ.



Dữ liệu đồ thị

Mỗi đối tượng trong loại này là một đồ thị, thuộc tính của chúng là các nút và kết nối giữa các nút đó. Ví dụ một đồ thị tổng quát và liên kết HTML như sau là dạng dữ liệu đồ thị

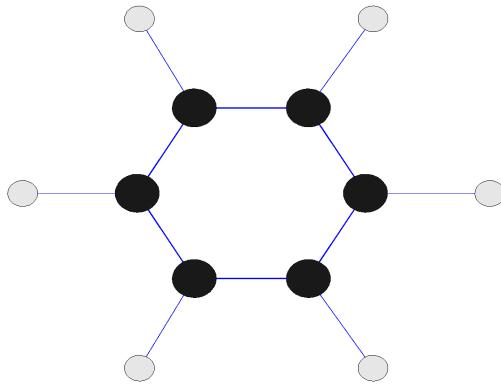


```

<a href="papers/papers.html#bbbb">  
Data Mining </a>  
<li>  
<a href="papers/papers.html#aaaa">  
Graph Partitioning </a>  
<li>  
<a href="papers/papers.html#aaaa">  
Parallel Solution of Sparse Linear System of Equations </a>  
<li>  
<a href="papers/papers.html#ffff">  
N-Body Computation and Dense Linear System Solvers
    
```

Dữ liệu hóa học

Mỗi đối tượng của loại này có thể là một cấu trúc phân tử của một chất nào đó. Ví dụ như cấu trúc phân tử của Benzene C₆H₆ được thể hiện trong hình vẽ dưới đây



Dữ liệu có thứ tự

Là một chuỗi các giao dịch hay các chuỗi di truyền DNA, ví dụ như hình vẽ dưới đây



Chất lượng dữ liệu

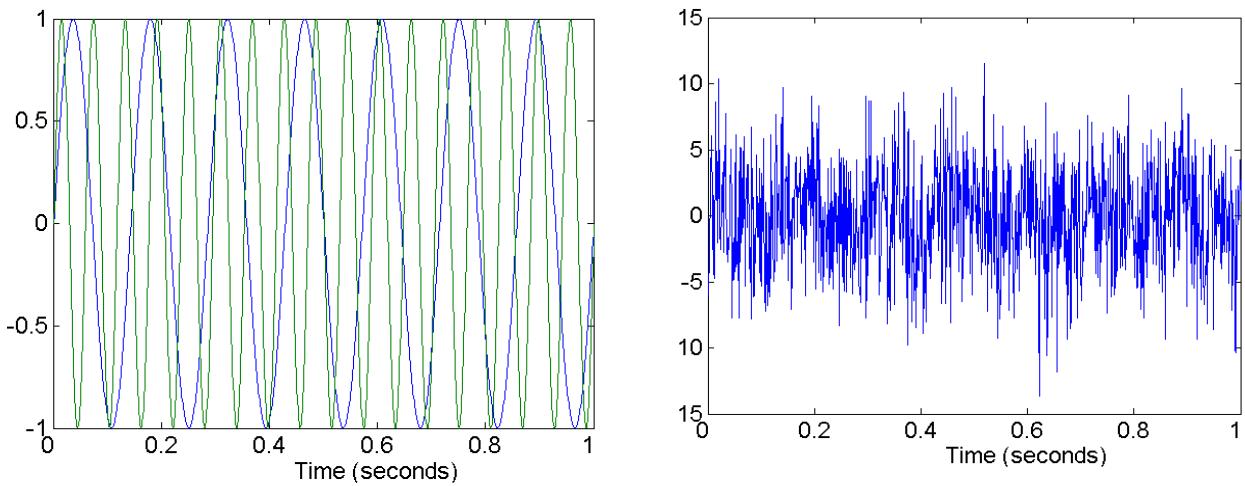
Nói đến chất lượng của dữ liệu, chúng ta quan tâm tới giải quyết 3 câu hỏi liên quan dưới đây

- Về chất lượng dữ liệu bao gồm những kiểu vấn đề gì?
- Làm thế nào chúng ta có thể phát hiện những vấn đề với dữ liệu?
- Chúng ta có thể làm gì với những vấn đề này?

Một số ví dụ về vấn đề chất lượng dữ liệu như dữ liệu có nhiễu và yếu tố ngoại lai hay dữ liệu có một số giá trị bị mất, hay dữ liệu bị lặp lại

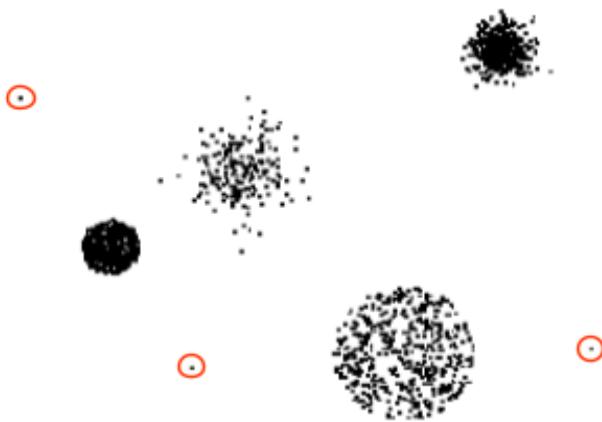
Dữ liệu có nhiễu

Nhiều liên quan tới sự thay đổi của các giá trị nguyên gốc ban đầu, một ví dụ về nhiễu là sự méo tiếng của một người đang nói chuyện được thể hiện trong hình vẽ dưới đây (bên trái thể hiện giọng nói bởi hai sóng hình sin, bên phải thể hiện giọng nói bị méo đi do có nhiễu



Dữ liệu có thành phần ngoại lai

Thành phần ngoại lai là các đối tượng dữ liệu có các đặc điểm khác biệt nhiều so với các đối tượng khác trong cùng một tập dữ liệu. Ví dụ dưới đây biểu diễn thành phần ngoại lai màu đỏ



Dữ liệu bị thiếu giá trị

Trường hợp này là do thông tin không được thu thập đầy đủ (ví dụ như có người từ chối không đưa chiều cao và cân nặng của họ cho người đi thu thập dữ liệu) hoặc do các thuộc tính có thể không được áp dụng cho mọi trường hợp (ví dụ như thuộc tính thu nhập hàng năm không được áp dụng cho trẻ em)

Để giải quyết vấn đề thiếu giá trị này chúng ta có một số cách như loại bỏ các đối tượng dữ liệu bị thiếu đó hoặc ước lượng để điền vào những giá trị còn thiếu, hoặc có thể bỏ qua những giá trị thiếu đó trong quá trình phân tích.

Dữ liệu bị lặp lại Các tập dữ liệu có thể bao gồm các đối tượng dữ liệu là bản sao của nhau, hoặc gần như là bản sao của nhau. Lý do chính của việc lặp lại này là khi kết hợp dữ liệu từ các nguồn khác nhau. Ví dụ: cùng một người có nhiều địa chỉ email.

Để giải quyết vấn đề này chúng ta có thể làm sạch dữ liệu thông qua các tiến trình xử lý các vấn đề về trùng lặp dữ liệu.

Tiền xử lý dữ liệu

Do dữ liệu có những vấn đề nên trên nên trước khi áp dụng các thuật toán khai phá dữ liệu chúng ta cần thực hiện việc tiền xử lý dữ liệu. Các công việc đó bao gồm các kỹ thuật sau:

- Tích hợp
- Lấy mẫu
- Giảm số chiều
- Lựa chọn tập thuộc tính con đặc trưng
- Tạo mới thuộc tính đặc trưng
- Rời rạc hóa và nhị phân hóa
- Chuyển đổi các thuộc tính

Kỹ thuật tích hợp

Là kết hợp 2 hay nhiều thuộc tính (đối tượng) thành 1 thuộc tính (đối tượng) đơn nhằm mục đích:

- Giảm dữ liệu do giảm số lượng thuộc tính hoặc đối tượng
- Thay đổi quy mô ví dụ các thành phố kết hợp thành các vùng, tiểu bang, nước...
- Làm dữ liệu ổn định hơn do dữ liệu tổng hợp có xu hướng ít bị biến đổi hơn

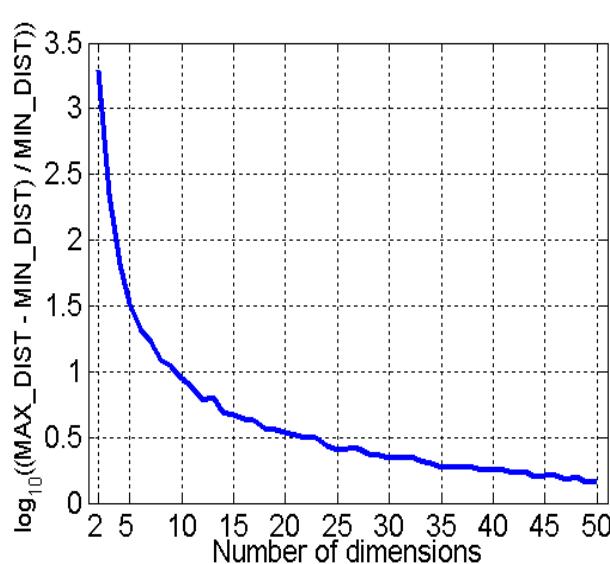
Kỹ thuật lấy mẫu (Sampling)

- Lấy mẫu là kỹ thuật chính được sử dụng cho việc lựa chọn dữ liệu. Nó thường được dùng cho cả việc điều tra sơ bộ dữ liệu và phân tích dữ liệu cuối cùng.
- Nhà thống kê lấy mẫu bởi vì việc thu được tập dữ liệu toàn bộ theo mong muốn là quá đắt hoặc tiêu tốn nhiều thời gian.
- Lấy mẫu được dùng trong khai phá dữ liệu vì việc xử lý tập dữ liệu toàn bộ là quá tốn kém
- Nguyên tắc quan trọng cho việc lấy mẫu hiệu quả là:
 - Sử dụng các mẫu sẽ làm việc tốt gần như với các tập dữ liệu toàn bộ, nếu lấy các mẫu điển hình

- Một mẫu gọi là điển hình nếu nó có các tính chất gần giống như tập dữ liệu ban đầu
- **Các kiểu lấy mẫu**
 - Lấy mẫu ngẫu nhiên đơn giản: Lựa chọn một item bất kỳ có xác suất bằng nhau
 - Lấy mẫu mà không thay thế: khi một item được chọn, nó sẽ bị xóa khỏi tập hợp
 - Lấy mẫu và thay thế: Đối tượng sẽ không bị xóa khi nó được chọn để lấy mẫu
 - Trong lấy mẫu có thay thế, cùng một đối tượng có thể được chọn nhiều hơn một lần
 - Lấy mẫu phân tầng: Chia nhỏ dữ liệu thành nhiều phân vùng, sau đó rút ra các mẫu ngẫu nhiên từ mỗi phân vùng

Những khó khăn của tính chất đa chiều

- Khi số chiều tăng lên, dữ liệu sẽ càng thưa thớt trong không gian mà nó chiếm
- Việc định nghĩa mật độ và khoảng cách giữa các điểm, một điều quan trọng trong việc phân cụm và phát hiện thành phần ngoại lai, trở nên kém ý nghĩa
- Sự phụ thuộc của mật độ và khoảng cách giữa các điểm đối với số lượng các chiều được thể hiện trong hình vẽ dưới đây.



- Ngẫu nhiên tạo ra 500 điểm
- tính toán sự khác biệt giữa khoảng cách max và min giữa 2 điểm bất kỳ

Kỹ thuật giảm số chiều

- Mục đích:
 - Tránh các khó khăn về đa chiều

- Giảm thiểu thời gian và dung lượng bộ nhớ cho các thuật toán khai phá dữ liệu
 - Cho phép dữ liệu được hình dung dễ dàng hơn
 - Có thể giúp loại bỏ các đặc điểm không liên quan hoặc làm giảm nhiễu
- Các kỹ thuật
 - Phân tích thành phần chính (không trình bày trong chương trình, gọi mở để sinh viên đọc thêm tài liệu nếu muốn)
 - Phân ly giá trị đơn (không trình bày trong chương trình gọi mở để sinh viên đọc thêm tài liệu nếu muốn)
 - Kỹ thuật khác: Kỹ thuật giám sát và phi tuyển

Kỹ thuật lựa chọn tập con đặc tính

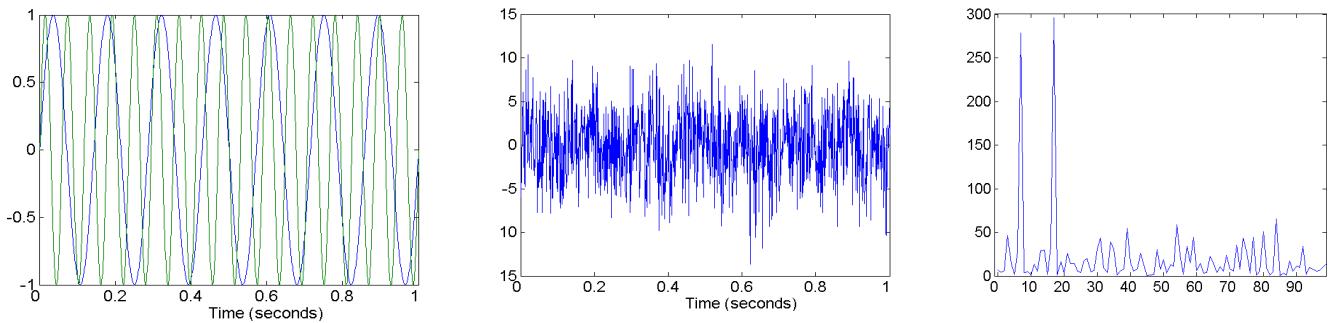
đây là một cách để giảm số chiều dữ liệu

- Có thể loại bỏ các đặc tính dư thừa: những đặc tính dư thừa là những đặc tính trùng lặp nhiều hoặc tất cả các thông tin đó có chứa trong một hoặc nhiều thuộc tính khác. Ví dụ: giá mua của một sản phẩm và số tiền thuế bán hàng đã nộp
- Có thể loại bỏ các đặc điểm không liên quan: các đặc tính không liên quan là các đặc tính không chứa thông tin hữu ích cho việc khai phá dữ liệu. Ví dụ: định danh hay mã số (ID) của sinh viên thường không liên quan đến việc dự đoán điểm trung bình các môn học (GPA) của sinh viên đó.
- Các kỹ thuật chính để lựa chọn tập con đặc trưng
 - Phương pháp vét cạn (Brute-force) là phương pháp mà chúng ta cần
 - Thủ tất cả các tập con đặc trưng có thể làm đầu vào cho thuật toán khai phá dữ liệu
 - Các phương pháp nhúng
 - Việc lựa chọn đặc trưng diễn ra một cách tự nhiên như là một phần của thuật toán khai phá dữ liệu
 - Các phương pháp lọc:
 - Các đặc trưng được lựa chọn trước khi thuật toán khai phá dữ liệu chạy
 - Các phương pháp bọc:

- Sử dụng các thuật toán khai thác dữ liệu như là một hộp đen để tìm tập con thuộc tính tốt nhất

Kỹ thuật tạo mới đặc tính

- Là cách tạo các thuộc tính mới mà chúng có thể chứa đựng thông tin quan trọng trong một tập dữ liệu có hiệu quả hơn nhiều so với các thuộc tính ban đầu
- Ba phương pháp luận chung để tạo đặc tính mới là
 - Trích từ một đặc tính ban đầu: cách này phụ thuộc nhiều vào một miền dữ liệu cụ thể của đặc tính được trích lọc cũng như đặc tính được tạo mới
 - Ánh xạ dữ liệu sang một không gian mới
- Xây dựng đặc tính mới bằng cách kết nối các đặc tính gốc khác nhau để tạo được một đặc tính mới có ích hơn.
- Ví dụ của việc ánh xạ dữ liệu sang một không gian dữ liệu mới thông qua kiến thức cũ đã học đó là khai triển Fourier và Wavelet được thể hiện trong hình vẽ dưới đây, trong đó hình số 1 từ bên trái sang là khai triển Fourier của tín hiệu âm thanh, hình 2 là có nhiễu và hình 3 là thể hiện trong không gian tần số (Frequency)

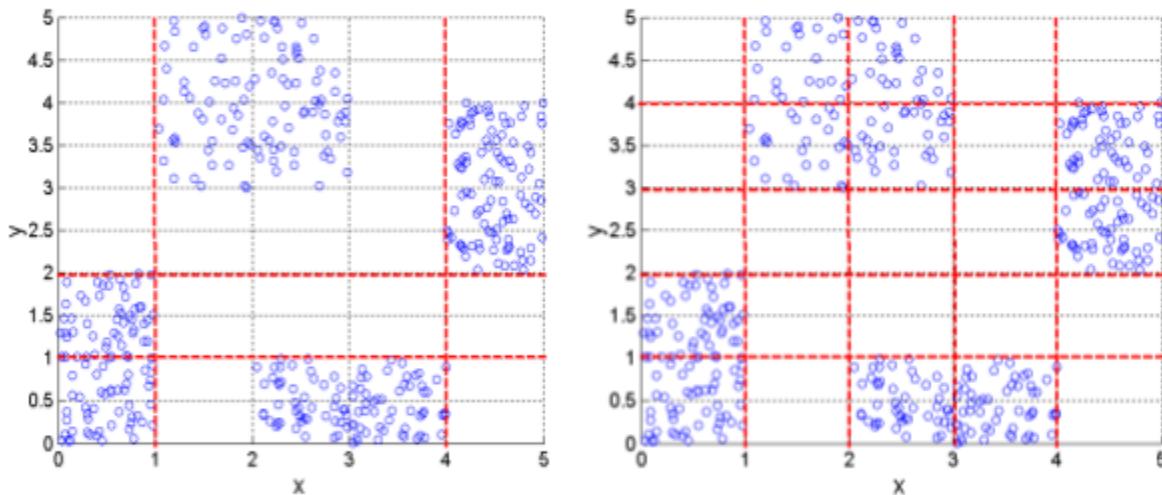


Kỹ thuật rời rạc hóa và nhị phân hóa

- Các kỹ thuật khai phá dữ liệu khác nhau đòi hỏi các dạng dữ liệu cụ thể khác nhau cho mỗi đầu vào của thuật toán. Ví dụ như dạng dữ liệu đầu vào phải ở dạng phân loại (rời rạc) hoặc ở dạng nhị phân (nhị phân hóa) hoặc dạng phạm vi, tỉ lệ (một dạng tổng quát hơn của nhị phân)
- Vì thế nếu dữ liệu cần khai phá ban đầu không ở dạng thích hợp với dạng dữ liệu đầu vào cho thuật toán chúng ta dùng để khai phá thì việc rời rạc hóa dữ liệu hay nhị phân hóa dữ liệu trở nên cần thiết. Ví dụ dữ liệu thu thập được ở dạng liên tục nhưng dữ liệu đầu vào

cho thuật toán cần dạng rời rạc thì chúng ta cần thực hiện rời rạc hóa dữ liệu trước khi tiến hành khai phá chúng.

- Rời rạc hóa có thể hiểu là chuyển đổi các thuộc tính khoảng (trong một phạm vi liên tục) thành thuộc tính loại rõ ràng
- Nhị phân hóa có thể hiểu là chuyển đổi các thuộc tính không phải nhị phân sang nhị phân
- Ví dụ về việc rời rạc hóa sử dụng phân lớp: ban đầu dữ liệu là dạng các điểm trong hệ tọa độ hai chiều x và y, phân bố trong một phạm vi nào đó dựa trên phương pháp tính entropy của chúng, chúng ta sử dụng nhãn phân loại theo hai chiều x và y để rời rạc hóa dữ liệu như hình vẽ dưới đây trong đó hình bên trái thể hiện 3 loại cho x ([0,1], [1,4] và >4) và y ([0,1], [1,2] và >2), hình bên phải thể hiện 5 loại cho x ([0,1], [1,2], [2,3], [3,4] và >4) và y ([0,1], [1,2], [2,3], [3,4] và >4)



Kỹ thuật chuyển đổi thuộc tính

- Kỹ thuật này sử dụng một hàm ánh xạ toàn bộ các giá trị của một thuộc tính sang một tập mới các giá trị thay thế sao cho mỗi giá trị cũ có thể được xác định với một giá trị mới
- Có thể sử dụng một số hàm đơn giản như x^k , $\log(x)$, e^x , $|x|$ hay một số hàm chuẩn hóa để đưa miền giá trị của thuộc tính về một miền giá trị với biên độ nhỏ, hiệu quả hơn trong quá trình tính toán của thuật toán khai phá. Ví dụ: trong kỹ thuật mạng nơron nhân tạo, chúng ta thường phải đưa chuyển đổi miền giá trị của thuộc tính về khoảng [0,1] để kết quả khai phá thu được chính xác hơn (thuật toán khai phá nhanh hội tụ hơn).

4.2 Phương pháp khai phá bằng luật kết hợp

Nguồn gốc của khai phá luật kết hợp

Bài toán có nguồn gốc từ việc đi mua hàng tại các siêu thị, thông thường quan sát thấy rằng khi một người mua mặt hàng này thì sẽ mua thêm những mặt hàng nhất định khác đi kèm với nó. Thực tế này được phát biểu thành bài toán về luật kết hợp như sau:

Cho trước một tập các giao dịch, tìm các luật có thể tiên đoán sự xuất hiện của một mặt hàng này dựa trên sự xuất hiện của các mặt hàng khác trong giao dịch đó.

Xét một ví dụ về khai phá luật kết hợp: các mặt hàng mua bán trong siêu thị của các giao dịch được thể hiện trong bảng sau gồm hai cột, cột thứ nhất là mã của giao dịch, cột thứ hai là các mặt hàng mua trong lần giao dịch đó.

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Luật kết hợp quan sát được từ dữ liệu trong bảng bên gồm
 $\{Diaper\} \rightarrow \{Beer\}$,
 $\{Milk, Bread\} \rightarrow \{Eggs, Coke\}$,
 $\{Beer, Bread\} \rightarrow \{Milk\}$,

Luật kết hợp $\{Diaper\} \rightarrow \{Beer\}$ có nghĩa là nếu mua Diaper (tã giấy cho trẻ con) thì có nhiều khả năng là người đó sẽ mua thêm Beer (bia). Luật này thể hiện sự xảy ra đồng thời của hai sự kiện mua tã giấy và mua bia chứ không phải là quan hệ nhân quả.

Bài toán đặt ra cho khai phá dữ liệu ở đây là tìm ra tất cả những luật kết hợp kiểu như trên đối với một tập giao dịch cho trước.

Các ứng dụng của luật kết hợp

- Được sử dụng trong việc thống nhất bảo dưỡng các gian bán hàng trong một cửa hàng hay siêu thị. Với những luật kết hợp phát hiện ra được, chúng ta có thể trả lời câu hỏi những gian hàng nào nên được bảo dưỡng đồng thời trong quá trình bán hàng
- Được sử dụng trong bán đồ điện gia dụng, trả lời câu hỏi những sản phẩm gia dụng nào nên lưu trữ nhập kho cùng nhau?
- Được sử dụng trong việc gửi kèm thư quảng cáo trực tiếp các sản phẩm kèm theo việc bán sản phẩm khác vì chúng ta biết được các sản phẩm nào hay được bán cùng nhau
- Được sử dụng trong khuyến mại bán hàng và quảng cáo tiếp thị

- Được sử dụng trong việc quản lý xếp đặt các gian hàng trong siêu thị

Trước khi trình bày thuật toán tìm luật kết hợp, chúng ta cùng xem xét một số khái niệm cơ bản cho bài toán này với ví dụ minh họa là bảng giao dịch mua hàng gồm 5 bản ghi được trình bày ở trên.

Khái niệm cơ bản trong bài toán tìm luật kết hợp

- Tập các mặt hàng là một tập hợp bao gồm một hoặc nhiều hơn một mặt hàng. Ví dụ: {Milk, Bread, Diaper}
- k-tập mặt hàng: là tập bao gồm k mặt hàng
- Độ số hỗ trợ (σ) là tần số xuất hiện của một tập mặt hàng. Ví dụ $\sigma(\{Milk, Bread, Diaper\}) = 2$ trong ví dụ minh họa
- Độ hỗ trợ là tỷ lệ các giao dịch có chứa một tập mặt hàng nào đó. Ví dụ $s(\{Milk, Bread, Diaper\}) = 2/5$
- Tập mặt hàng thường xuyên là một tập mặt hàng có độ hỗ trợ lớn hơn hoặc bằng một ngưỡng được gọi là minsup hay độ hỗ trợ nhỏ nhất.
- Luật kết hợp được thể hiện dưới dạng của $X \rightarrow Y$, ở đó X, Y là tập các mặt hàng
- Các độ đo để đánh giá một luật kết hợp
 - o Độ hỗ trợ (s) là tỷ lệ các giao dịch chứa cả X và Y.
 - o Độ tin cậy (c) dùng để đo các mặt hàng của Y xuất hiện trong các giao dịch có chứa X.
 - o Ví dụ với luật kết hợp $\{Milk, Diaper\} \Rightarrow Beer$ thì

$$s = \frac{\sigma(Milk, Diaper, Beer)}{|T|} = \frac{2}{5} = 0.4 \quad \text{và} \quad c = \frac{\sigma(Milk, Diaper, Beer)}{\sigma(Milk, Diaper)} = \frac{2}{3} = 0.67$$

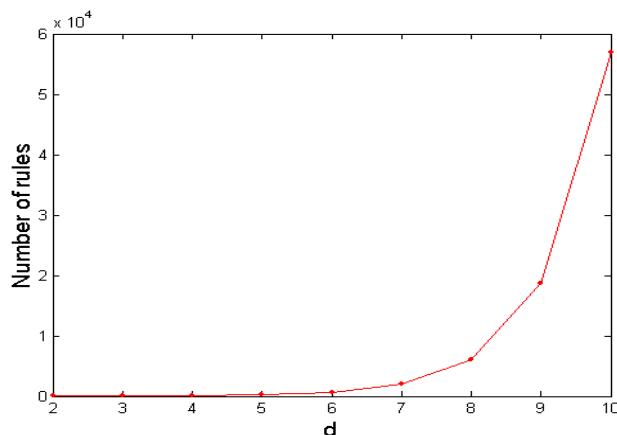
- Nhiệm vụ khai phá tìm luật kết hợp: Cho trước một tập các giao dịch T, mục tiêu của khai phá luật kết hợp là tìm ra tất cả các luật có độ hỗ trợ \geq ngưỡng $minsup$ và độ tin cậy \geq ngưỡng $minconf$

Cách tiếp cận theo kiểu vét cạn (Brute-force approach)

- Liệt kê tất cả các luật kết hợp có thể có
- Tính toán độ hỗ trợ và độ tin cậy của mỗi luật
- Loại bỏ những luật không thỏa mãn lớn hơn hoặc bằng ngưỡng $minsup$ và $minconf$

- Một nhược điểm lớn nhất của cách tiếp cận này là số lượng phép tính rất lớn do số lượng luật kết hợp có thể có rất lớn, tăng theo cấp lũy thừa hay nói một cách khác độ phức tạp tính toán quá lớn, không thể thực hiện được.
- Xem xét độ phức tạp tính toán trong ví dụ sau: giả sử xét d mặt hàng khác nhau thì tổng số tập mặt hàng sẽ là 2^d và tổng số các luật kết hợp có thể có là

$$R = \sum_{k=1}^{d-1} \left[\binom{d}{k} \times \sum_{j=1}^{d-k} \binom{d-k}{j} \right] = 3^d - 2^{d+1} + 1. \text{ Nếu } d=6 \text{ thì } R=602 \text{ luật.}$$



Biểu đồ sự phụ thuộc của tổng số luật cần xét với số lượng mặt hàng d được thể hiện dưới đây cho thấy số luật tăng rất nhanh theo cấp lũy thừa.

- Một cách phân tách hai độ đo của thuật toán luật kết hợp

Quan sát một nhóm các luật như sau

$\{\text{Milk}, \text{Diaper}\} \rightarrow \{\text{Beer}\}$ ($s=0.4, c=0.67$)

$\{\text{Milk}, \text{Beer}\} \rightarrow \{\text{Diaper}\}$ ($s=0.4, c=1.0$)

$\{\text{Diaper}, \text{Beer}\} \rightarrow \{\text{Milk}\}$ ($s=0.4, c=0.67$)

$\{\text{Beer}\} \rightarrow \{\text{Milk}, \text{Diaper}\}$ ($s=0.4, c=0.67$)

$\{\text{Diaper}\} \rightarrow \{\text{Milk}, \text{Beer}\}$ ($s=0.4, c=0.5$)

$\{\text{Milk}\} \rightarrow \{\text{Diaper}, \text{Beer}\}$ ($s=0.4, c=0.5$)

- Ta thấy tất cả những luật trên là phân vùng nhị phân của cùng một tập các mặt hàng $\{\text{Milk}, \text{Diaper}, \text{Beer}\}$
- Những luật bắt nguồn từ cùng một tập các mặt hàng có sự hỗ trợ giống nhau nhưng có thể có độ tin cậy khác nhau. Do đó chúng ta có thể tách riêng việc kiểm tra yêu cầu về độ hỗ trợ và yêu cầu về độ tin cậy. Dẫn đến một cách tiếp cận khác được trình bày tiếp sau đây

Khai phá luật kết hợp với cách tiếp cận hai bước

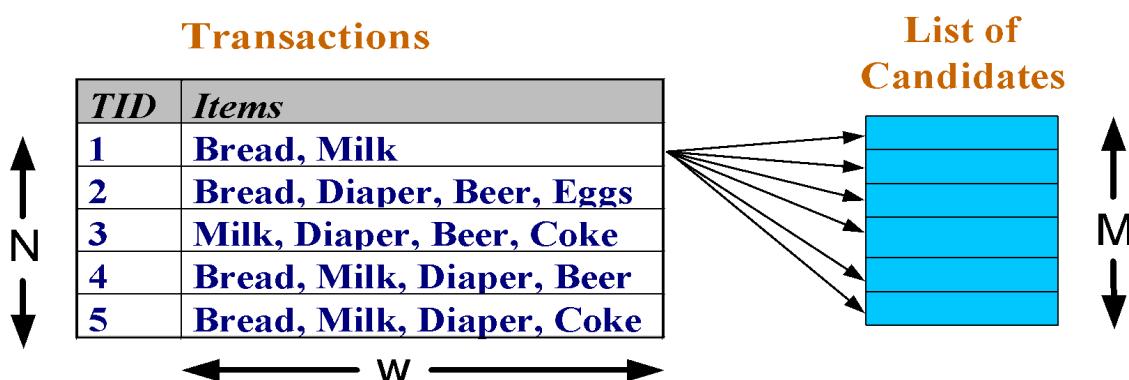
Bước 1: Sinh các tập mặt hàng thường xuyên có nghĩa là sinh ra tất cả các tập mặt hàng mà có độ hỗ trợ lớn hơn hoặc bằng giá trị ngưỡng *minsup*

Bước 2: Sinh các luật bằng cách sinh ra các luật từ tập các mặt hàng thường xuyên có độ tin cậy lớn hơn trong đó mỗi luật chính là một sự phân chia nhị phân (làm hai phần) của một tập mặt hàng thường xuyên.

Việc sinh ra tập các mặt hàng thường xuyên vẫn có độ phức tạp tính toán rất lớn do số lượng tập các mặt hàng có thể có rất nhiều, chính vì thế độ phức tạp tính toán chung của cách tiếp cận này vẫn rất lớn

Sinh các tập mặt hàng thường xuyên theo cách tiếp cận vét cạn

- Mỗi tập các mặt hàng trong cột 2 của bảng các giao dịch xét làm ví dụ dưới đây được coi là một ứng cử viên cho tập mặt hàng thường xuyên
- Sau đó ta cần đếm số giao dịch hỗ trợ cho mỗi ứng cử viên này bằng cách duyệt cơ sở dữ liệu



- Trong hình vẽ trên danh sách các ứng cử viên được thể hiện trong danh sách bên phải với số phần tử của danh sách là M, mỗi ứng cử viên được nối với một giao dịch liên quan trong bảng
- Độ phức tạp tính toán của việc sinh tập các mặt hàng thường xuyên theo cách này là $O(NMw)$ là rất cao vì trong đó $M = 2^d$, N là số phần giao dịch trong bảng, w là số mặt hàng lớn nhất trong mỗi giao dịch.
- Chúng ta muốn giảm độ phức tạp tính toán xuống để cách tiếp cận này khả thi hơn, để giảm độ phức tạp này, có các cách sau đây
 - Giảm số lượng ứng cử viên M: nếu tìm kiếm toàn bộ thì $M=2^d$, ta có thể dùng các kỹ thuật cắt cành để giảm số lượng ứng cử viên bằng cách loại bỏ những nhóm ứng viên không thỏa mãn cùng một lúc cho nhanh.

- Giảm số lượng các giao dịch N: giảm số lượng giao dịch N khi kích cỡ của tập các mặt hàng tăng, có một cách là thực hiện lấy mẫu điển hình của N giao dịch
- Giảm số lượng phép toán so sánh độ hỗ trợ với $minsup$, mỗi ứng cử viên cần đếm số giao dịch hỗ trợ bằng cách duyệt qua N giao dịch nên tốn N phép toán, mà có M ứng cử viên nên tổng số phép toán so sánh là NM. Để giảm việc so sánh này chúng ta có thể
 - Sử dụng một cấu trúc dữ liệu hiệu quả để lưu trữ các ứng cử viên hoặc các giao dịch
 - Không nhất thiết phải kiểm tra xem mỗi ứng cử viên có trong mỗi giao dịch hay không

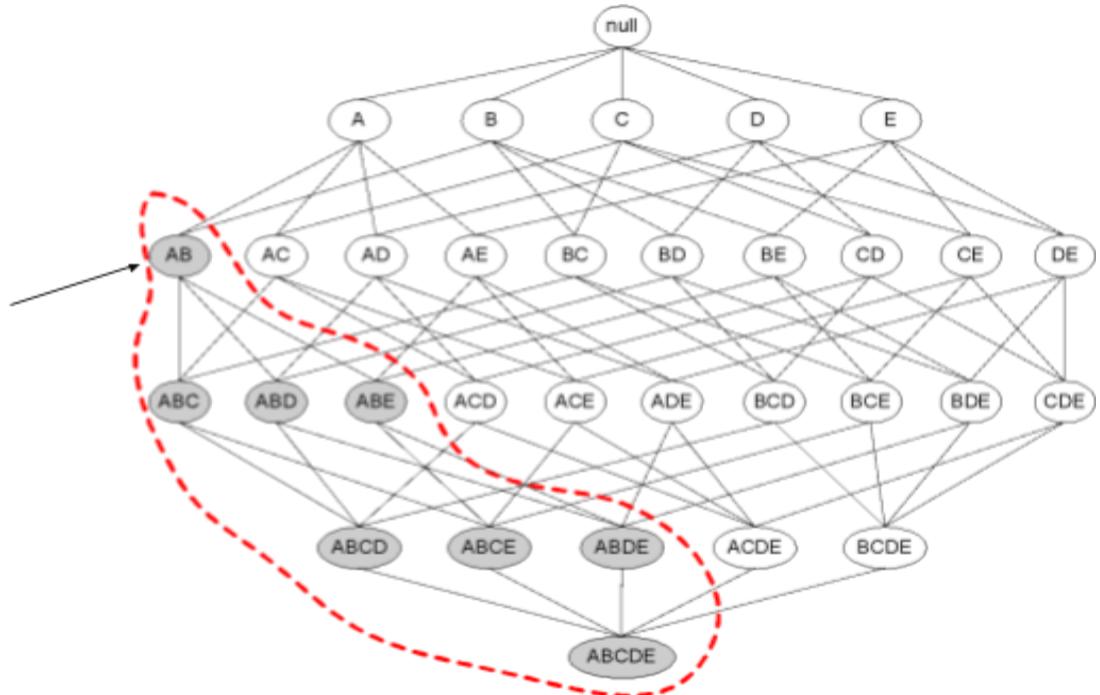
Chúng ta sẽ lần lượt xem xét các phương thức giảm độ phức tạp tính toán trong phần tiếp theo

Phương thức giảm số lượng các ứng cử viên: thuật toán Apriori

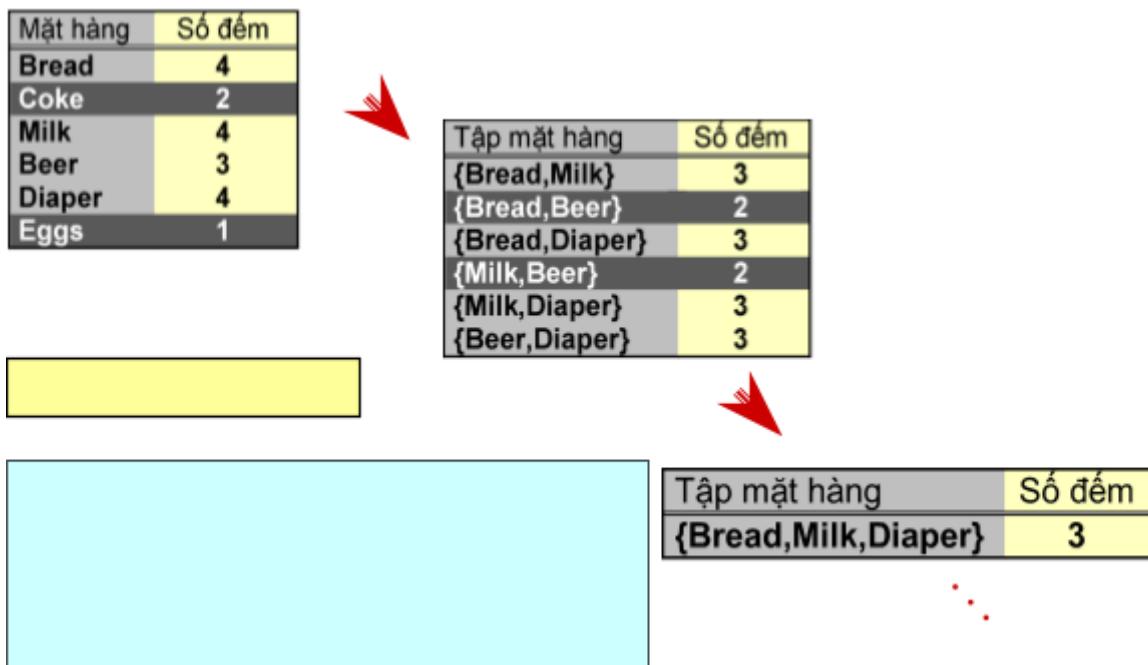
Nguyên tắc Apriori: Nếu một tập các mặt hàng là thường xuyên thì tất cả các tập con của nó cũng là thường xuyên.

Nguyên lý Apriori đúng đắn bởi thuộc tính sau đây của độ đo độ hỗ trợ là đúng
 $\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$

- Độ hỗ trợ của một tập mặt hàng không bao giờ vượt quá độ hỗ trợ của các tập con của nó. Tính chất này được gọi là thuộc tính không đơn điệu của độ hỗ trợ. Hình vẽ sau mô tả nguyên lý Apriori trong đó tập mặt hàng {AB} được tìm thấy là không thường xuyên nên tất cả các tập mặt hàng mà nhận {AB} là tập con đều được cho là không thường xuyên và được cắt bỏ toàn bộ (phần được khoanh màu đỏ trong hình vẽ). Nhờ có nguyên lý này, số lượng các ứng cử viên giảm đi đáng kể khi xác định được một tập mặt hàng nào đó không thỏa mãn là tập thường xuyên.



Việc giảm số lượng ứng cử viên khi áp dụng nguyên lý Apriori được thể hiện qua ví dụ về con số. Với ví dụ được thể hiện trong bảng giao dịch trên, lợi ích của việc sử dụng nguyên lý Apriori được thể hiện trong hình dưới đây



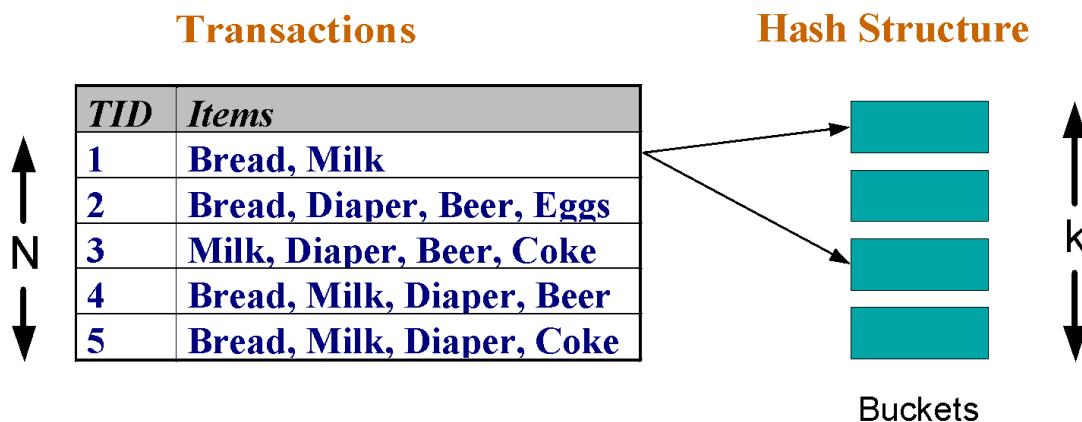
Thuật toán Apriori để giảm số lượng các ứng cử viên

- Gán k=1
- Sinh ra một tập mặt hàng với độ dài là 1

- Lặp cho tới khi không còn tập mặt hàng mới nào được xác định
 - Sinh ra các tập mặt hàng với $(k+1)$ phần tử từ các tập mặt hàng với k phần tử
 - Cắt bỏ những tập mặt hàng chứa tập con có độ dài k mà không phải là tập thường xuyên
 - Đếm số hỗ trợ của mỗi ứng viên bằng cách quét toàn bộ cơ sở dữ liệu
 - Loại bỏ những ứng viên không phải thường xuyên, chỉ để lại những tập mặt hàng thường xuyên.

Thuật toán Apriori để giảm số lượng phép toán so sánh

- Đếm số lượng ứng cử viên bằng cách duyệt hết bảng dữ liệu chứa các giao dịch để xác định số đếm hỗ trợ của mỗi tập mặt hàng là ứng cử viên
- Để giảm số lượng phép so sánh, cần lưu trữ các ứng cử viên trong một cấu trúc băm. Nếu lưu trữ như vậy thay vì phải kiểm tra xem mỗi ứng cử viên có trong dữ liệu của mỗi giao dịch không thì chúng ta có thể kiểm tra xem nó có được chứa trong các ngăn của cấu trúc băm không. Như đã được biết, lưu trữ dạng cấu trúc băm sẽ làm giảm độ phức tạp tìm kiếm đi nhiều so với tìm kiếm tuyến tính như ban đầu.



Cài đặt một thuật toán Apriori sử dụng cấu trúc cây hàm băm

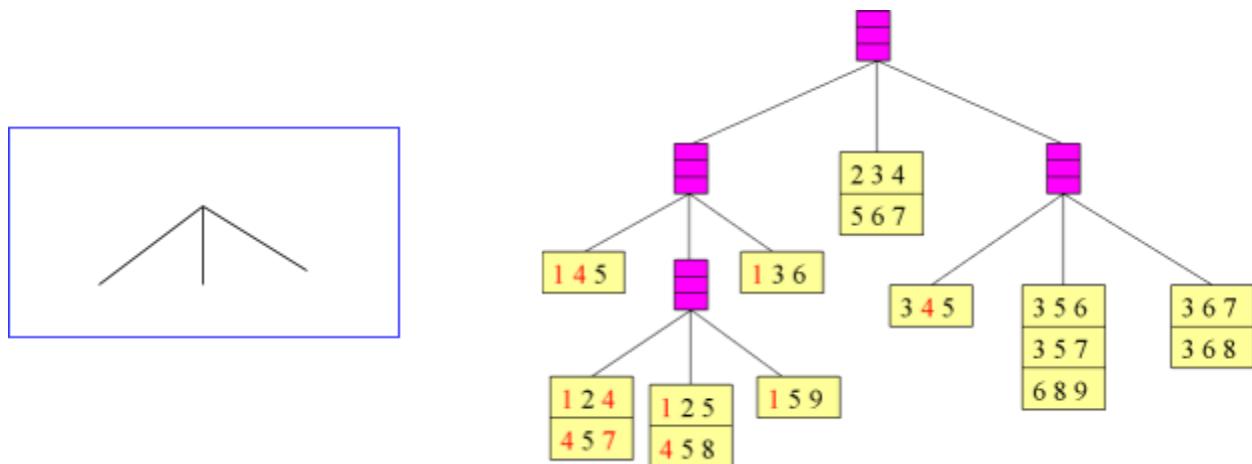
Chúng ta xem xét một ví dụ để hiểu được cách cài đặt thuật toán này. Giả sử bạn được cho 15 tập mặt hàng ứng cử viên với độ dài là 3 như sau: {1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}.

Bạn cần định nghĩa hai thông số sau để có thể tổ chức những ứng cử viên trên thành một cấu trúc cây băm

- Một hàm băm để tính toán các giá trị của mỗi ứng viên khi xây dựng cây hàm băm với các ứng cử viên là thành phần của cây (nút trong hoặc lá của cây)

- Kích cỡ lớn nhất của nút lá hay số lượng tập các mặt hàng lớn nhất có thể chứa trong mỗi nút lá. Nếu số lượng các tập mặt hàng ứng viên này vượt quá kích cỡ lớn nhất thì nút lá đó cần phân tách ra làm hai nút lá mới.

Trong ví dụ này ta định nghĩa hàm băm bởi một phép toán module 3 cho một cây 3 nhánh. Đầu tiên để xây dựng cây hàm băm thì ta cần mã hóa các mặt hàng dưới dạng số tự nhiên để tính toán. Nếu giá trị của mặt hàng chia 3 dư 1 thì được lưu ở nhánh thứ nhất, nếu chia 3 dư 2 thì lưu ở nhánh thứ hai, chia 3 dư 0 thì được lưu ở nhánh thứ ba. Các ứng viên được lưu trữ dưới dạng cây băm như hình vẽ dưới đây

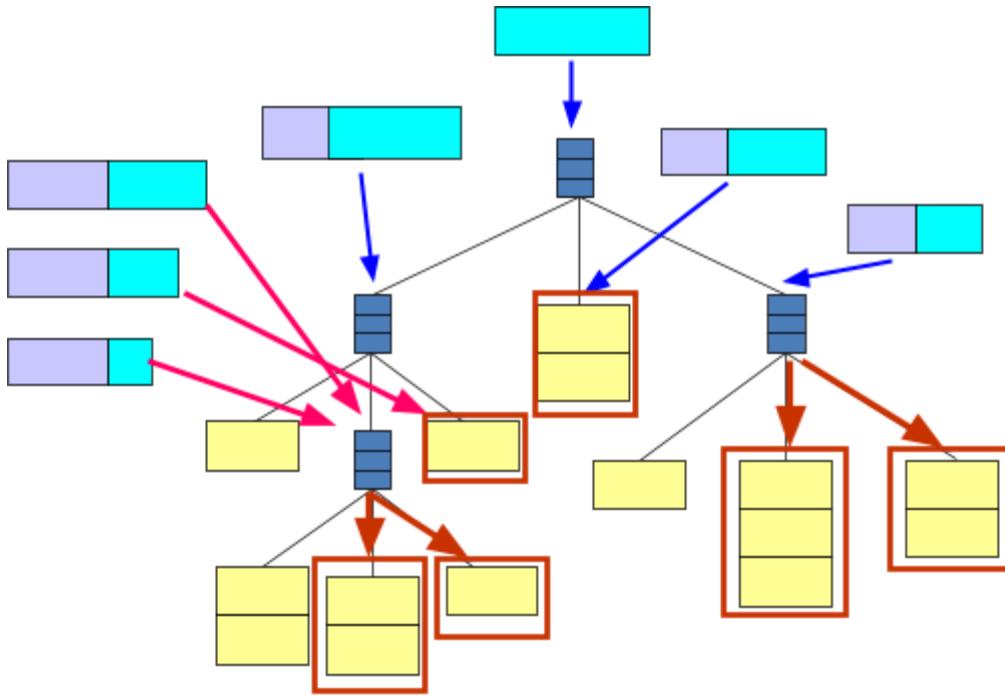


Để xây dựng cây hàm băm thì chúng ta cứ xét lần lượt từng vị trí (1,2,3) trong mỗi tập mặt hàng ứng viên để đưa từng tập vào đúng vị trí của nó, chúng ta không mô tả từng bước ở đây, mà chỉ mô tả quá trình như hình vẽ dưới đây trong đó đưa 11 ứng viên trong số 15 ứng viên vào đúng vị trí của nó trong cây (yêu cầu sinh viên tự thực hiện từng bước để hiểu được cách cài đặt thuật toán)

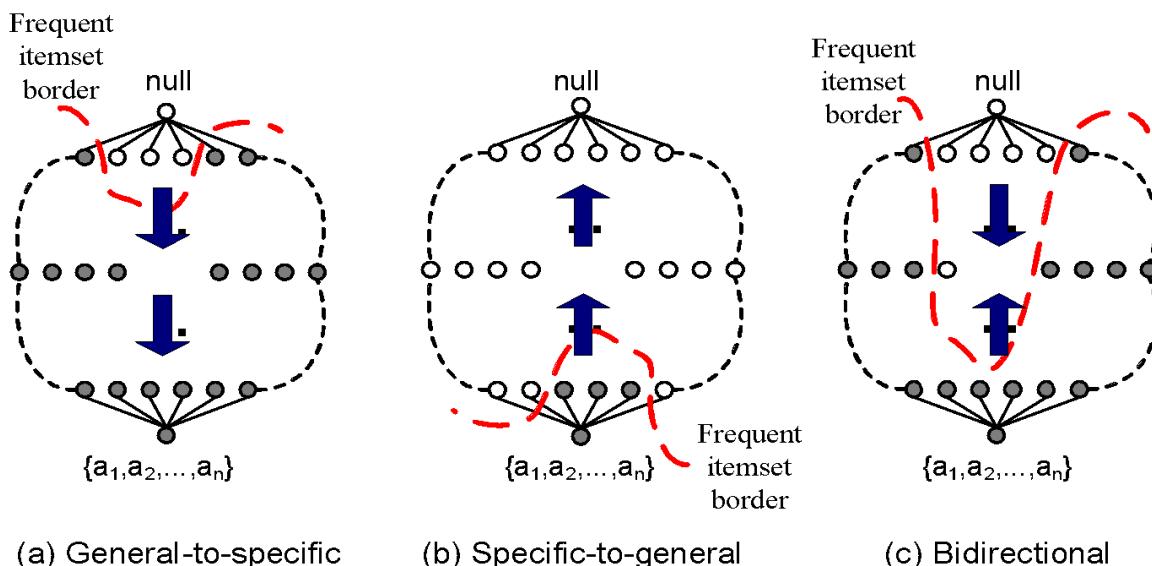
Các phương pháp tìm kiếm dựa trên thuật toán Apriori

Để thực hiện việc so sánh, nếu không dùng cấu trúc băm mà lưu trữ tuyến tính như ban đầu chúng ta cần duyệt các phần tử trong toàn bộ không gian tìm kiếm, với những kiểu duyệt sau đây

- Từ tổng quát tới cụ thể (từ tập mặt hàng có số lượng ít tới số lượng nhiều - General-to-specific)
- Từ cụ thể tới tổng quát (từ tập mặt hàng có số lượng nhiều tới số lượng ít - Specific-to-general)
- Kiểu trộn giữa hai kiểu trên



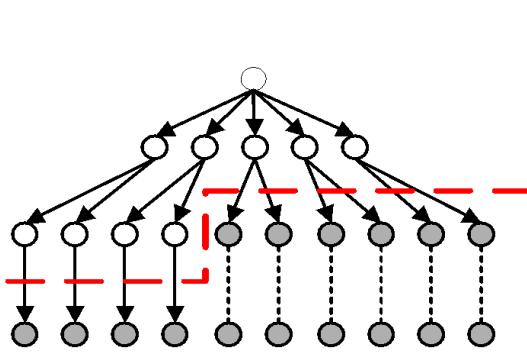
Hình vẽ sau thể hiện các kiểu tìm kiếm đó



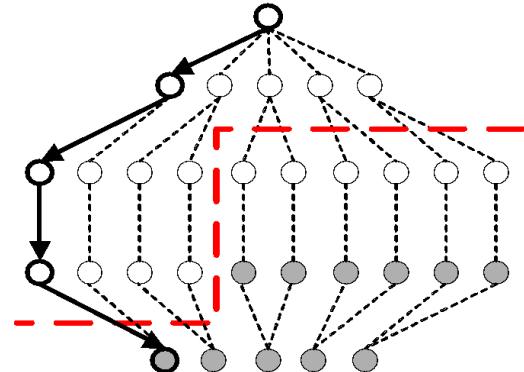
Trong đó **Frequent itemset border** là vùng biên giữa tập mặt hàng thường xuyên và không thường xuyên. Các tập mặt hàng thường xuyên được biểu diễn bằng một hình tròn màu trắng, còn không thường xuyên bằng hình tròn nhỏ màu xám.

Nếu dùng cấu trúc băm để lưu trữ ứng cử viên thì chúng ta có thể dùng các thuật toán tìm kiếm theo chiều sâu (Depth-first) hay theo chiều rộng (Breadth-first) để tìm kiếm trong cây hàm băm,

rõ ràng là thuật toán tìm kiếm loại này có độ phức tạp tính toán ít hơn rất nhiều so với cách tìm kiếm tuần tự ở trên



(a) Breadth first



(b) Depth first

Việc tìm kiếm như mô tả trong thuật toán Apriori ban đầu sẽ gây ra hiện tượng thắt cổ chai vì

- Việc sinh thêm ứng cử viên có thể cho kết quả là một tập ứng viên với số lượng khổng lồ:
 - o 10^4 tập mặt hàng có một phần tử sẽ sinh ra 10^7 ứng viên có 2 phần tử
 - o Để phát hiện ra một tập mặt hàng xuyên kích thước 100, ví dụ như $\{a_1, a_2, \dots, a_{100}\}$, cần sinh ra $2^{100} \sim 10^{30}$ ứng viên
- Cần duyệt cơ sở dữ liệu nhiều lần: cần $n+1$ lần duyệt toàn bộ cơ sở dữ liệu với n là độ dài lớn nhất của tập mặt hàng đang xét.

Một phương pháp khác cho việc sinh ra tập mặt hàng thường xuyên ECLAT

Với mỗi mặt hàng, chúng ta lưu trữ một danh sách các mã định danh các giao dịch có chứa mặt hàng đó. Nói một cách khác, ta không lưu trữ các giao dịch dưới dạng các bảng thông thường (horizontal data layout) trong đó mỗi hàng chứa thông tin của một giao dịch mà lưu trữ dưới dạng cột (vertical data layout) trong đó mỗi cột tương ứng với một mặt hàng và các giao dịch chứa mặt hàng đó. Hình vẽ dưới đây thể hiện một ví dụ cho việc lưu trữ này.

Horizontal Data Layout

TID	Items
1	A,B,E
2	B,C,D
3	C,E
4	A,C,D
5	A,B,C,D
6	A,E
7	A,B
8	A,B,C
9	A,C,D
10	B

Vertical Data Layout

A	B	C	D	E
1	1	2	2	1
4	2	3	4	3
5	5	4	5	6
6	7	8	9	
7	8	9		
8	10			
9				

Tính độ hỗ trợ của mỗi tập k mặt hàng bằng cách lấy giao của danh sách mã giao dịch của hai tập mặt hàng con có $k-1$ phần tử. Ví dụ: để đếm độ hỗ trợ của $\{AB\}$ ta lấy giao của hai danh sách mã giao dịch của tập $\{A\}$ và $\{B\}$ như hình vẽ dưới đây

A	B	AB
1	1	1
4	2	5
5	5	7
6	7	
7	8	
8	10	8
9		

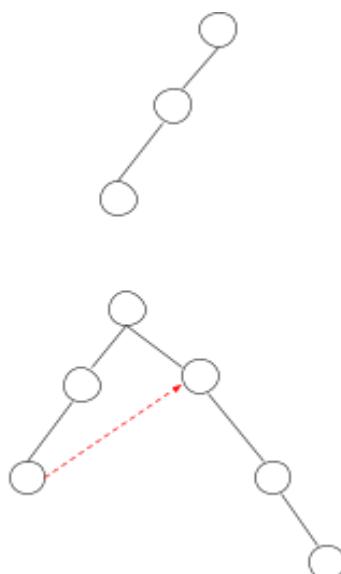
Ưu điểm của phương pháp này là cung cấp một cách tính độ hỗ trợ khá nhanh nhưng cũng có nhược điểm là tốn bộ nhớ để lưu trữ danh sách các mã giao dịch trung gian mà thường thì các danh sách trung gian này có kích cỡ khá lớn.

Một phương pháp sinh tập các mặt hàng thường xuyên FP-growth

Ý tưởng chính của phương pháp này là giảm thiểu số lượng các phép toán so sánh NM bằng cách sử dụng một cách biểu diễn nén cơ sở dữ liệu về các giao dịch trong một cây FP thay vì biểu diễn các ứng cử viên dưới dạng cấu trúc cây như phương pháp trình bày ở trên. Việc đầu tiên của phương pháp là xây dựng một cây FP, sau đó nó sử dụng một cách tiếp cận chia để trị để quy để khai phá ra các tập mặt hàng thường xuyên. Nhờ có việc lưu trữ dữ liệu về các giao dịch trong một cấu trúc hình cây nên việc tìm kiếm được thực hiện nhanh hơn.

Việc xây dựng cây FP được mô tả như hình vẽ sau:

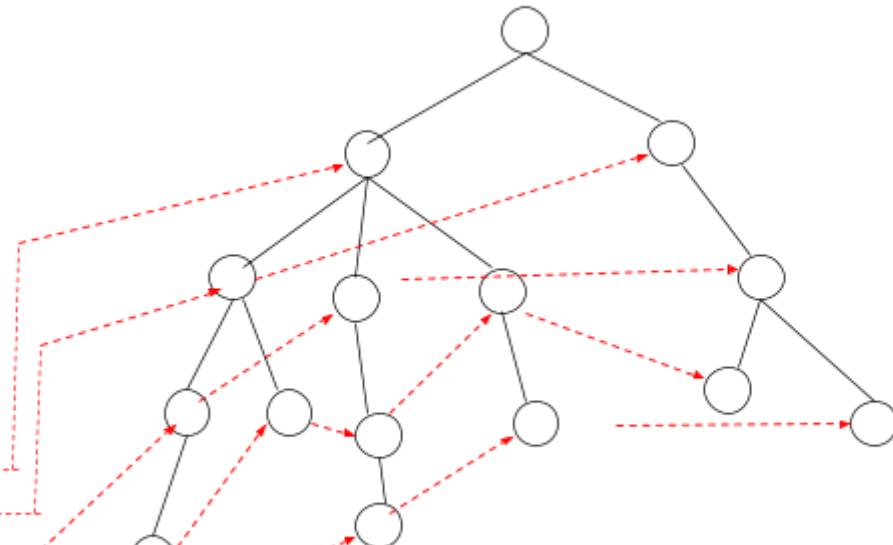
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}



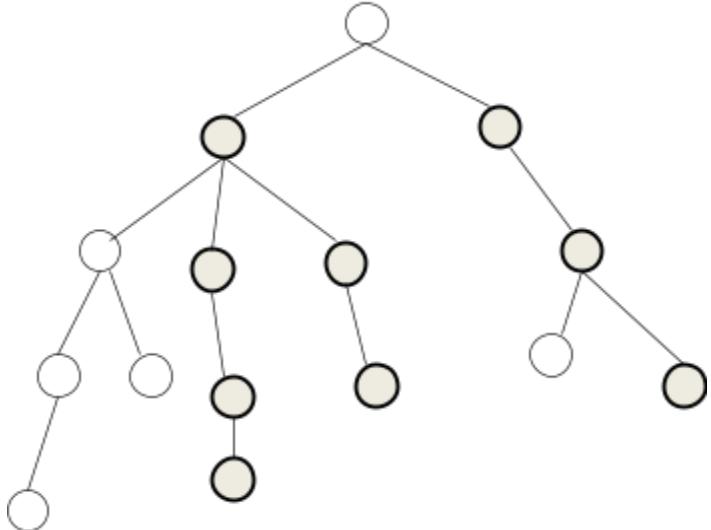
Lưu ý mỗi nút trong cây có nhãn dạng X:n trong đó X là mặt hàng trong giao dịch và n là số lần mặt hàng đó xuất hiện trong các giao dịch của cơ sở dữ liệu. Mỗi lần đọc một giao dịch trong cơ sở dữ liệu, các mặt hàng được đưa vào từ trái qua phải và từ gốc xuống đến lá. Sau khi cả 10 giao dịch được đọc và đưa vào cây FP ta thu được kết quả cuối cùng như hình vẽ dưới đây

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

Item	Pointer
A	-----
B	-----
C	-----
D	-----
E	-----



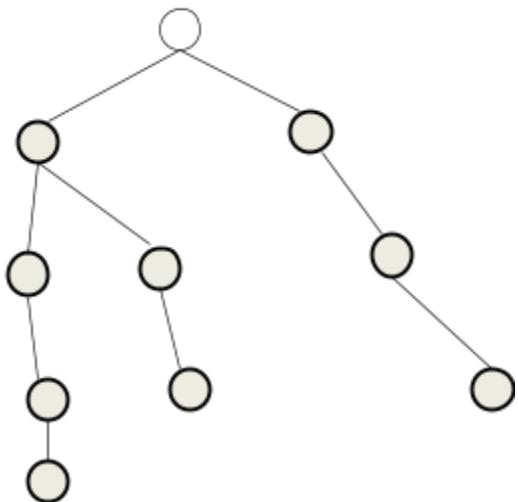
Trong hình vẽ trên, ta thấy cùng một mặt hàng X (như B, C, D, E) xuất hiện nhiều lần trong cây. Để tránh lưu trữ lặp lại trong bộ nhớ, chúng ta chỉ lưu trữ thông tin X một lần và dùng thêm một bảng tiêu đề để lưu thông tin của các con trỏ trỏ tới X. Khi X xuất hiện lần thứ hai trong cây, ta không dùng bản thân X để thể hiện mà dùng con trỏ trỏ tới X để lưu trữ, cho tiết kiệm bộ nhớ. Tiếp theo, sau khi đã xây dựng xong cây FP, để tìm kiếm các tập mặt hàng thường xuyên, ta xây dựng các cơ sở mẫu điều kiện cho từng tập mặt hàng ứng viên, xuất phát từ loại có một phần tử.. Ví dụ xây dựng cơ sở điều kiện cho E là tập hợp P trong đó mỗi phần tử chính là đường đi tới E (gồm một tập hợp có trật tự các mặt hàng) với số lần xuất hiện của mỗi mặt hàng trong đường đi đó.



Cây điều kiện cho E

$P = \{(A:1, C:1, D:1),$
 $(A:1, D:1),$
 $(B:1, C:1)\}$

Tiếp đến áp dụng FP-growth trên cây
 P một cách dễ qui cho cơ sở mẫu điều
 kiện



Cây điều kiện cho D nằm trong cây điều kiện
 cho E

Cây cơ sở điều kiện của E là

$P = \{(A:1, C:1, D:1, E:1), (A:1, D:1, E:1),$
 $(B:1, C:1, E:1)\}$

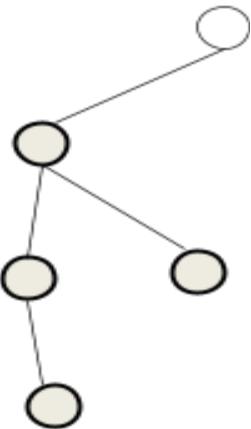
Đếm hỗ trợ cho E là 3 lớn hơn minsup nên
 $\{E\}$ là tập mặt hàng thường xuyên
 Có thể tiếp tục đệ quy FP-growth tại P vì E là
 tập mặt hàng thường xuyên

Cây cơ sở điều kiện cho D nằm trong cơ sở
 điều kiện cho E:

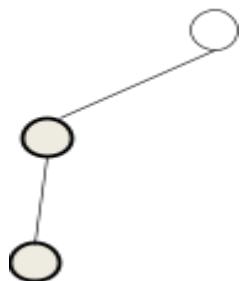
$P = \{(A:1, C:1, D:1),$
 $(A:1, D:1)\}$

Đếm hỗ trợ cho D là 2 nên $\{D, E\}$ là tập mặt
 hàng thường xuyên

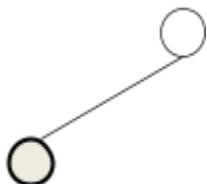
Có thể tiếp tục đệ qui FP-growth tại P



Cây điều kiện cho C nằm trong D nằm trong E

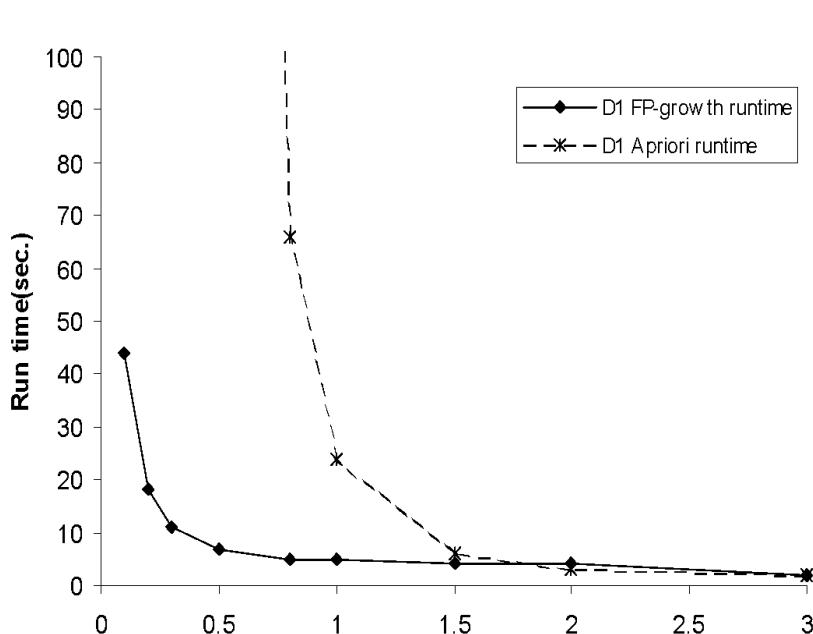


Cây điều kiện cho A nằm trong D nằm trong E



Việc đệ quy cứ tiếp tục như vậy cho đến khi tìm ra cây điều kiện cho A (chỉ có nút A).

Ưu điểm của cấu trúc cây FP



Cây cơ sở điều kiện cho C nằm trong D nằm trong E:

$$P = \{(A:1, C:1)\}$$

Đếm hỗ trợ cho C là $1 < 2 = \text{minsup}$ nên $\{C, D, E\}$ là tập mặt hàng không thường xuyên
Quá trình Đệ qui FP-growth tại P dừng, chuyển sang đệ quy ở nút trên nút đó trong cây P, tức là nút A

Đếm hỗ trợ cho A là $2 = \text{minsup}$ nên $\{A, D, E\}$ là tập mặt hàng thường xuyên

Quá trình Đệ qui FP-growth dừng vì đã đến gốc, bước tiếp theo xây dựng cây điều kiện cho C bên trong cây điều kiện cho E

Về hiệu quả hoạt động thì FP-growth nhanh hơn Apriori.

Lý do là vì

- Không phải tạo ứng cử viên, không phải kiểm tra ứng viên đó có phù hợp không

- Sử dụng cấu trúc dữ liệu nén
- Loại bỏ được việc duyệt CSDL nhiều lần
- Phép toán cơ bản là đếm và xây dựng cây FP

Độ phức tạp của khai phá luật kết hợp

Độ phức tạp của khai phá luật kết hợp phụ thuộc vào những yếu tố trình bày dưới đây

- Việc lựa chọn ngưỡng độ hỗ trợ nhỏ nhất ảnh hưởng rất lớn tới độ phức tạp tính toán
 - o Nếu hạ thấp giá trị của ngưỡng hỗ trợ số lượng các tập mặt hàng thường xuyên sẽ tăng
 - o Điều này có thể làm tăng số lượng các ứng cử viên và độ dài lớn nhất của các tập mặt hàng thường xuyên
- Số chiều của tập dữ liệu (số lượng các mặt hàng) cũng ảnh hưởng đến độ phức tạp tính toán vì
 - o Cần nhiều chỗ hơn để lưu trữ số đếm hỗ trợ của mỗi mặt hàng
 - o Nếu số lượng của các mặt hàng thường xuyên tăng, cả lượng tính toán cũng như chi phí vào ra cũng sẽ tăng
- Kích cỡ của cơ sở dữ liệu cũng là một yếu tố ảnh hưởng bởi vì thuật toán Apriori duyệt CSDL nhiều lần nên thời gian chạy thuật toán sẽ tăng theo số lượng giao dịch
- Độ rộng trung bình (số mặt hàng trung bình) của các giao dịch cũng gây ảnh hưởng: khi độ rộng của các giao dịch tăng theo độ dày đặc của tập dữ liệu. Điều này có thể làm tăng độ dài lớn nhất của các tập mặt hàng thường xuyên và chiều dài cần duyệt của cây hàm băm (số lượng tập con trong một giao dịch tăng theo độ dài của nó)

Sinh luật kết hợp

Sau khi tìm được các tập mặt hàng thường xuyên, nhiệm vụ tiếp theo là xác định luật kết hợp từ các tập mặt hàng thường xuyên đó.

Bài toán được phát biểu như sau: Cho một tập các mặt hàng thường xuyên L, hãy tìm tất cả các tập con f không rỗng của L sao cho $f \rightarrow L - f$ thỏa mãn yêu cầu về độ tin cậy nhỏ nhất.

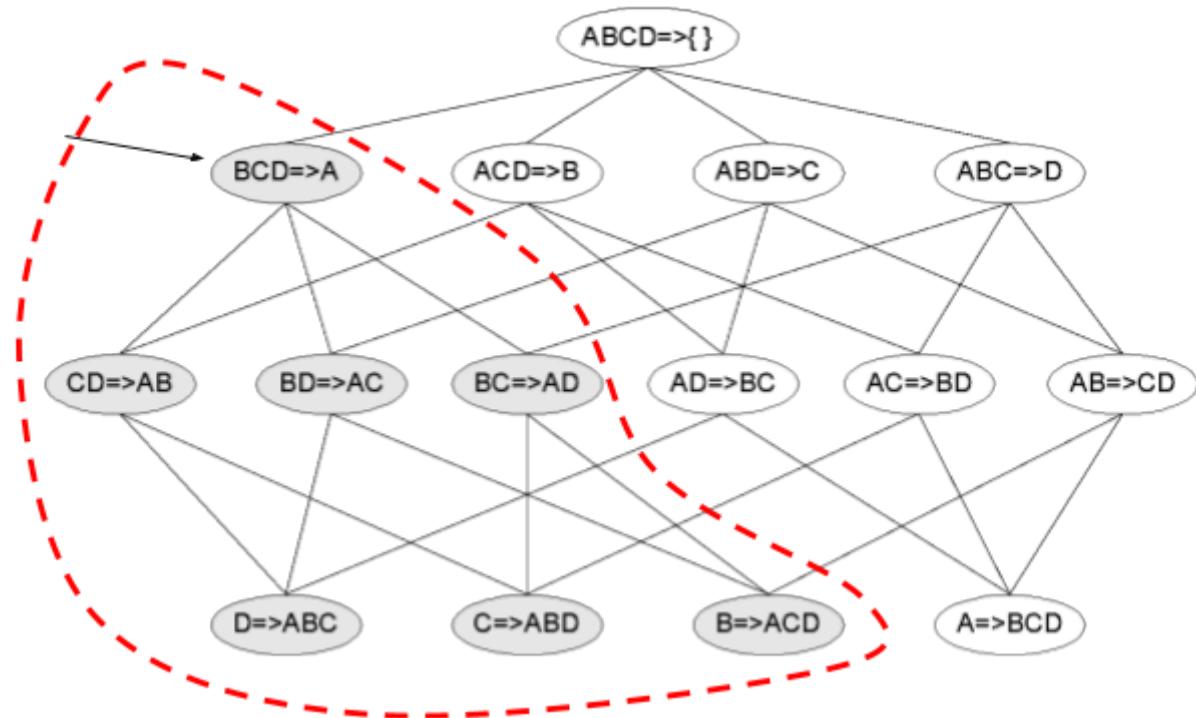
Nếu $\{A, B, C, D\}$ là một tập mặt hàng thường xuyên, các luật ứng cử viên là:

$ABC \rightarrow D$, $ABD \rightarrow C$, $ACD \rightarrow B$, $BCD \rightarrow A$,
 $A \rightarrow BCD$, $B \rightarrow ACD$, $C \rightarrow ABD$, $D \rightarrow ABC$
 $AB \rightarrow CD$, $AC \rightarrow BD$, $AD \rightarrow BC$, $BC \rightarrow AD$,
 $BD \rightarrow AC$, $CD \rightarrow AB$,

Nếu $|L| = k$, thì sẽ có $2^k - 2$ luật kết hợp được sinh ra (trừ 2 luật $L \rightarrow \emptyset$ và $\emptyset \rightarrow L$).

Ta thấy số lượng luật kết hợp khá lớn (cấp lũy thừa) nên có vấn đề về độ phức tạp tính toán. Vấn đề đặt ra ở đây là làm thế nào để sinh luật kết hợp từ các tập mặt hàng thường xuyên một cách có hiệu quả. Nhìn chung, độ tin cậy không có thuộc tính không đơn điệu như nguyên lý Apriori, có nghĩa là $c(ABC \rightarrow D)$ có thể lớn hơn hay nhỏ hơn $c(AB \rightarrow D)$, nhưng độ tin cậy của các luật được sinh ra từ cùng một tập mặt hàng lại có thuộc tính đơn điệu, ví dụ $L = \{A, B, C, D\}$: $c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$, chính vì vậy ta có thể áp dụng kỹ thuật cắt cành (loại bỏ) để giảm số lượng các luật sinh ra thỏa mãn yêu cầu về độ tin cậy.

Một ví dụ về việc sinh luật và loại bỏ các nhóm luật không thỏa mãn được thể hiện trong hình vẽ dưới đây



4.3 Phương pháp cây quyết định

Những khái niệm cơ bản trong bài toán phân loại

Định nghĩa bài toán phân loại

Trong phần này chúng ta nhắc lại khái niệm bài toán phân loại đã được trình bày trong bài đầu tiên của bài giảng này để tiện theo dõi những phần tiếp theo.

Cho một tập các bản ghi (còn được gọi là *tập huấn luyện*) trong đó mỗi bản ghi chứa một tập các thuộc tính, một trong những thuộc tính đó là thuộc tính lớp (dùng làm chỉ thị phân loại).

Mục tiêu của bài toán là tìm một *mô hình* cho thuộc tính lớp như là một hàm của giá trị các thuộc tính khác. Nhờ có hàm tìm được này giá trị của thuộc tính phân lớp sẽ được tính cho các bản ghi mới chưa thấy trước đây, các bản ghi này được phân loại càng chính xác càng tốt.

Một tập *kiểm thử* được sử dụng để xác định độ chính xác của mô hình. Thông thường, tập dữ liệu có sẵn đang xét được phân chia thành một tập dùng để huấn luyện mô hình và một tập để kiểm thử, với tập huấn luyện được sử dụng để xây dựng mô hình và tập kiểm thử được sử dụng để xác định tính đúng đắn của mô hình đó.

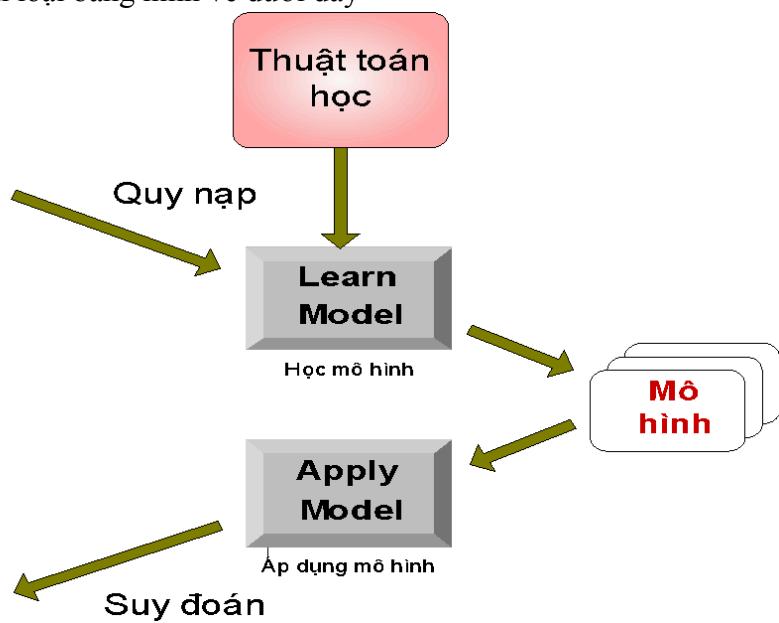
Chúng ta có thể minh họa bài toán phân loại bằng hình vẽ dưới đây

Tid	?Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Tập huấn luyện

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Tập kiểm thử



Các ứng dụng của bài toán phân loại

- Dự đoán khối u là lành tính hay ác tính.
- Phân loại các phiên giao dịch thẻ tín dụng là hợp lệ hay là gian lận.

- Phân loại cấu trúc thứ cấp của protein
- Phân loại tin tức tài chính, thời tiết, giải trí, thể thao, vv...

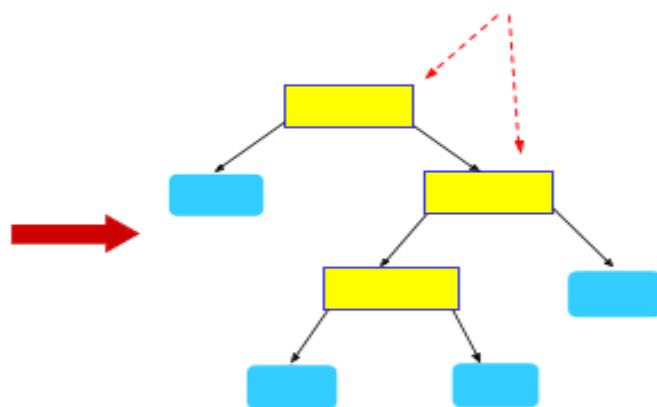
Phương pháp phân loại bằng cây quyết định

Các phương pháp phân loại được sử dụng trong khai phá dữ liệu

- Các phương pháp trên cơ sở cây quyết định
- Các phương pháp trên cơ sở luật kết hợp
- Các phương pháp lập luận trên cơ sở ghi nhớ
- Mạng Nơ ron nhân tạo
- Mạng Bayes đơn giản và mạng Bayes tổng quát
- Máy vector hỗ trợ

Trong phạm vi của chương trình môn học chúng ta chỉ đề cập tới phương pháp dựa trên cây quyết định. Trước khi đi sâu vào phương pháp dựa trên cây quyết định, chúng ta cùng xem xét một ví dụ về một cây quyết định như hình vẽ dưới đây: từ các bản ghi của tập dữ liệu huấn luyện được lưu trữ trong một bảng quan hệ thể hiện ở bên trái ở hình, ta có thể xây dựng được một cây quyết định như hình bên phải

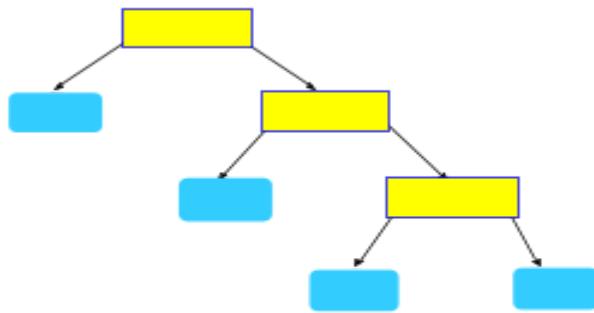
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



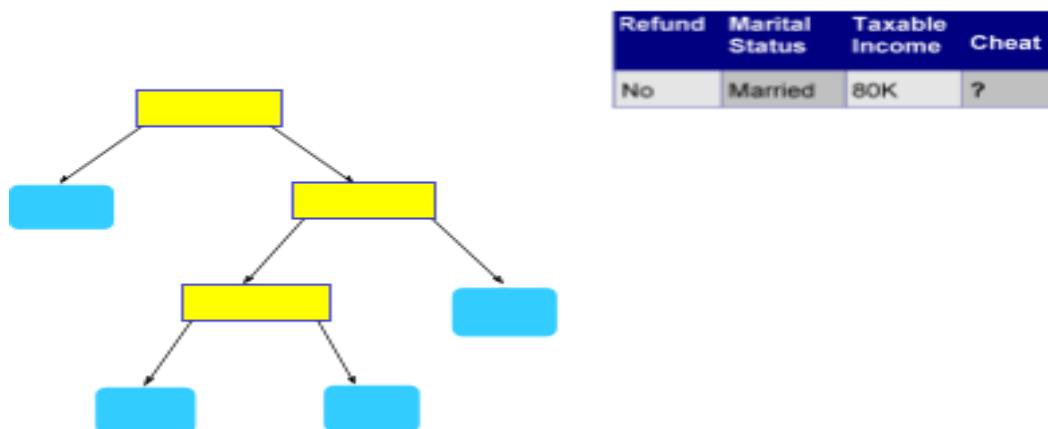
Cấu tạo của một cây quyết định bao gồm các nút trong cây và nút lá, nút trong cây là các thuộc tính và nút lá chính là các giá trị của lớp. Trong ví dụ minh họa trên các thuộc tính được thể hiện bởi các nút trong màu vàng, các nút lá được thể hiện bằng màu xanh và nhận hai giá trị của thuộc tính lớp YES và NO. Dễ thấy rằng có thể xây dựng nhiều cây quyết định từ cùng một dữ liệu vì

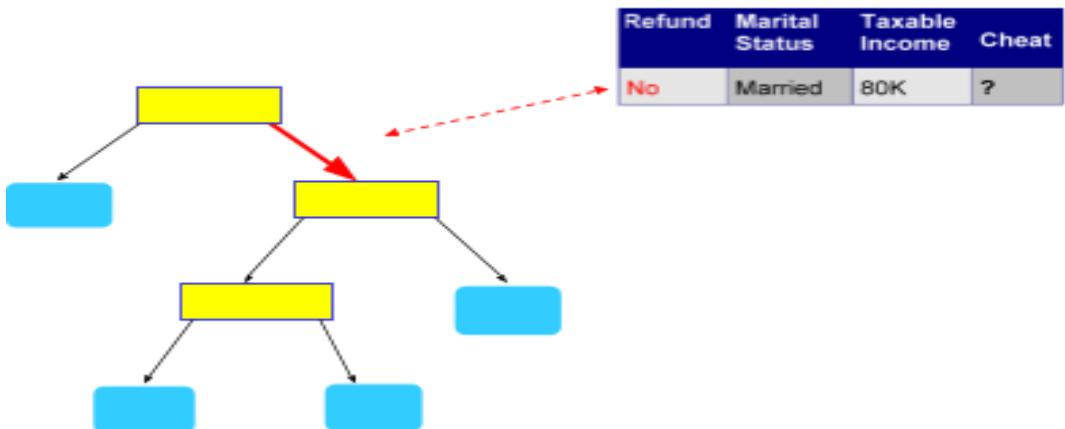
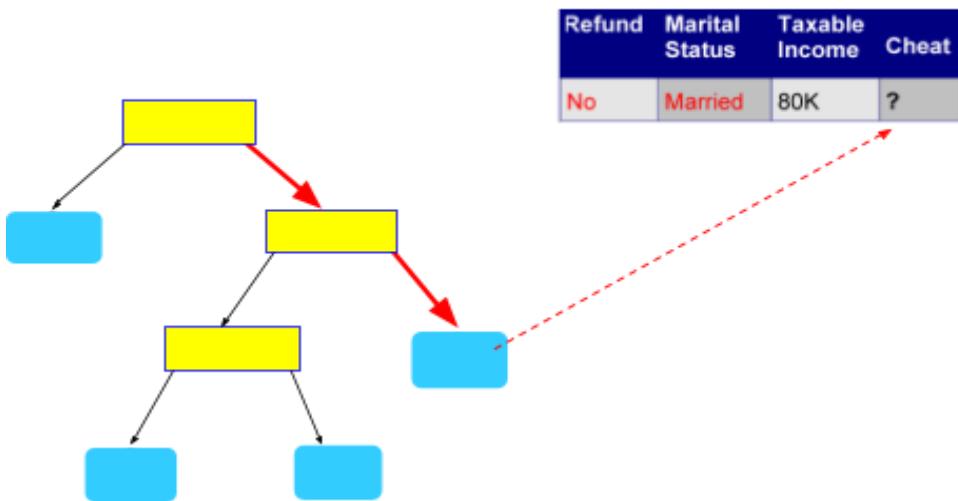
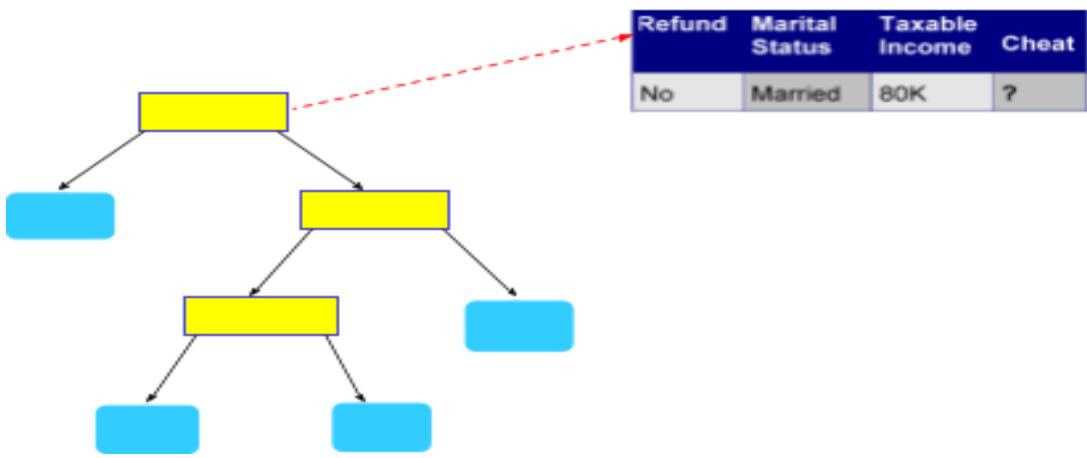
với mỗi nút trong có nhiều khả năng chọn một trong những thuộc tính của bảng dữ liệu để đại diện cho nó. Ví dụ một cây quyết định khác từ dữ liệu huấn luyện ở trên được thể hiện trong hình vẽ dưới đây

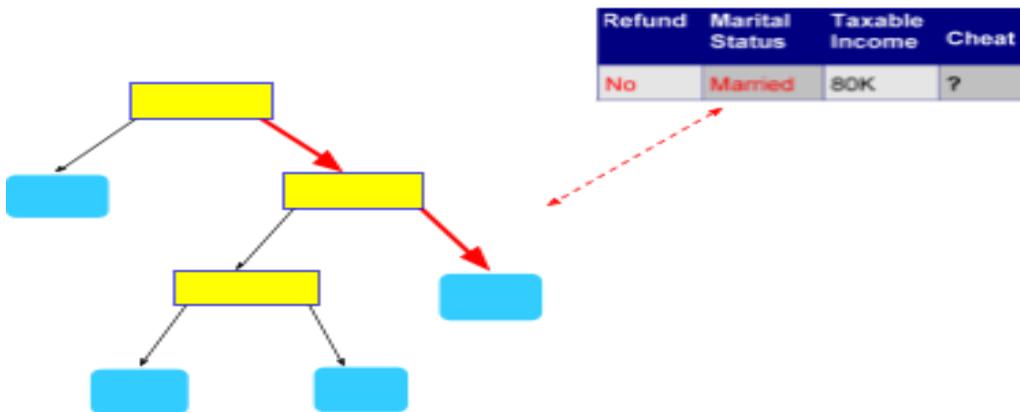
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Với dữ liệu huấn luyện như trên, một mô hình cây quyết định sẽ được huấn luyện, sau đó người sử dụng có thể dùng mô hình đó để phân loại các bản ghi mới. Để kiểm tra tính đúng đắn của mô hình, chúng ta áp dụng mô hình cho tập kiểm thử, xuất phát bắt đầu từ gốc cây (là một thuộc tính), rồi dựa vào giá trị của thuộc tính tại nút gốc đó, xác định hướng đi tiếp xuống sâu trong cây, đến nút trong tiếp theo, lại dựa vào giá trị của thuộc tính ở tại nút trong đó, cứ như vậy cho đến khi chạm nút lá, giá trị của nút lá chính là lớp của bản ghi. Minh họa cho việc áp dụng mô hình với dữ liệu kiểm thử được thể hiện trong một loạt hình vẽ dưới đây, mỗi hình vẽ là một bước trong quá trình được mô tả bằng lời ở trên







Nhiệm vụ phân loại bằng cây quyết định được mô tả như hình vẽ dưới đây (cụ thể hóa hình vẽ mô tả bài toán phân loại)

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Tập huấn luyện

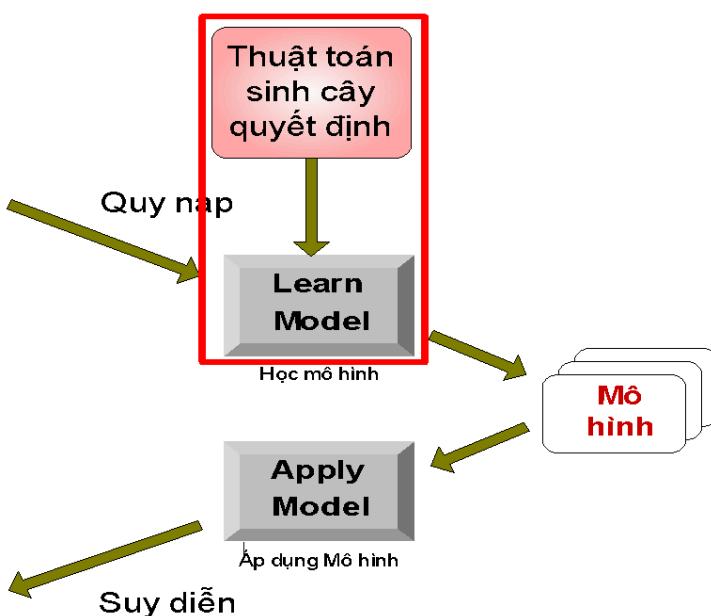
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

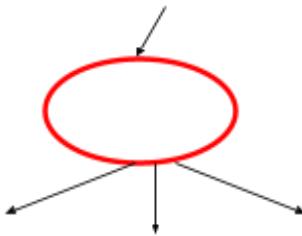
Tập kiểm thử

Các thuật toán tìm cây quyết định

Có nhiều nhóm thuật toán được áp dụng để xây dựng cây quyết định bao gồm

- Các thuật toán của Hunt: là một nhóm thuật toán ra đời sớm nhất
- Nhóm thuật toán CART
- Nhóm thuật toán ID3,C4.5
- Nhóm SLIQ , SPRINT





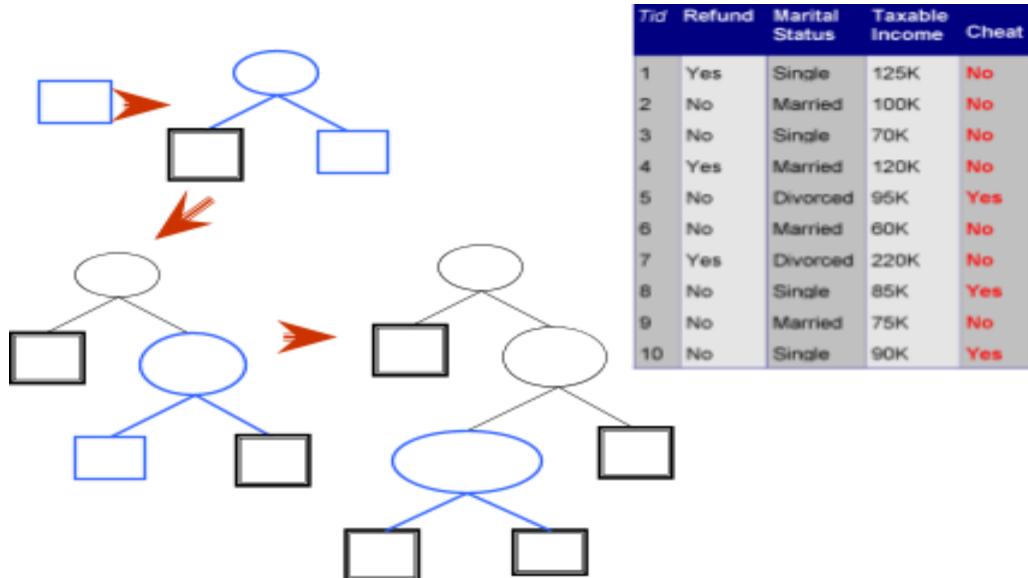
Nhóm thuật toán Hunt

Cấu trúc chung của thuật toán Hunt: xét ví dụ được đề cập đến ở trên

- Gọi D_t là tập các bản ghi huấn luyện mà hướng tới nút t
- Thủ tục gán nhãn chung cho các nút của cây như sau:
 - o Nếu D_t chứa các bản ghi thuộc cùng một lớp y_t , thì t là một nút lá được gán nhãn y_t
 - o Nếu D_t là một tập rỗng, t là một nút lá được gán nhãn bởi lớp mặc định, y_d
- Nếu D_t chứa các bản ghi thuộc nhiều hơn một lớp, sử dụng một thuộc tính kiểm tra để phân chia dữ liệu thành các tập con nhỏ hơn. Áp dụng thủ tục trên một cách đệ quy cho mỗi tập con.

Hình vẽ dưới đây minh họa cho thuật toán Hunt từng bước một

Thuộc tính Refund được xét đến đầu tiên, tất cả các bản ghi có thuộc tính Refund=Yes thì đều thuộc loại Don't Cheat (Cheat=No). Còn các bản ghi có thuộc tính Refund=No thì thuộc cả hai loại Cheat (Cheat=Yes) và Don't Cheat (Cheat=No) do đó cần dùng một thuộc tính thứ hai để phân loại, ở đây chọn là thuộc tính Marital Status (tình trạng hôn nhân), với thuộc tính này, xét tương tự, tất cả các bản ghi có thuộc tính Marital Status = Married đều thuộc loại Don't Cheat, còn Marital Status = Single hoặc Divorced thuộc cả hai loại nên lại phải dùng thuộc tính thứ ba để phân loại. Tương tự với thuộc tính thứ ba hay nút trong thứ ba của cây quyết định, chọn là Taxable Income, đây là một thuộc tính dạng liên tục nên cần nhị phân hóa giá trị của nó để sinh hai nhánh cây (với trường hợp này cây có hai nhánh, có thể tổng quát lên nhiều nhánh). Kết quả cuối cùng được mô tả như trong hình vẽ dưới đây



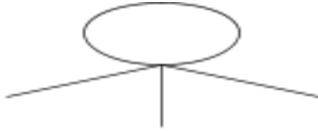
Khi xây dựng cây quyết định chúng ta áp dụng chiến lược tham lam: phân tách các bản ghi thành các nhánh dựa trên một phép kiểm tra giá trị thuộc tính để tối ưu một tiêu chí nào đó. Nếu tối ưu như vậy, mỗi tập huấn luyện sẽ cho một cây quyết định tối ưu chứ không còn cho nhiều cây quyết định như ban đầu nữa. Để xây dựng được cây tối ưu chúng ta cần giải quyết một số bài toán sau:

- Xác định phân tách các bản ghi trong tập huấn luyện tại một nút thành các nhánh trong cây thế nào thông qua việc
 - o Xác định cụ thể các điều kiện kiểm tra thuộc tính
 - o Xác định xem việc phân tách đó đã là tốt nhất chưa
- Xác định xem khi nào không cần phân tách một nút trong cây nữa

Chúng ta sẽ xem xét lần lượt từng ván đề, trước hết là làm thế nào để xác định cụ thể điều kiện kiểm tra ở một nút phân tách. Việc này phụ thuộc vào kiểu của thuộc tính là loại biến tên, biến có trật tự hay biến liên tục. Mỗi loại biến có cách xác định điều kiện kiểm tra khác nhau. Việc tìm điều kiện kiểm tra còn phụ thuộc vào số lượng nhánh muốn phân tách: có thể phân tách thành hai nhánh hay nhiều nhánh.

Phân tách cho các thuộc tính tên có hai trường hợp.

- Nếu là dạng phân tách nhiều nhánh, ta có thể dùng các giá trị phân biệt khác nhau cho mỗi nhánh như ví dụ sau chẳng hạn thuộc tính điều kiện là Car Type có 3 giá trị thì nút trong đại diện cho thuộc tính này được phân làm 3 nhánh như hình vẽ

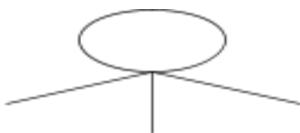


- Nếu là dạng phân tách nhị phân (thành hai nhánh) ta cần chia các giá trị thành hai tập con nên vấn đề ở đây là cần tìm ra sự phân tách tối ưu

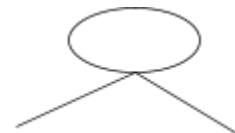


Tương tự với biến có trật tự

- Nếu là dạng phân tách nhiều nhánh, ta có thể dùng các giá trị phân biệt khác nhau cho mỗi nhánh như ví dụ sau chẳng hạn thuộc tính điều kiện là Size có 3 giá trị (Small, Medium, Large) thì nút trong đại diện cho thuộc tính này được phân làm 3 nhánh như hình vẽ



- Nếu là dạng phân tách nhị phân (thành hai nhánh) ta cần chia các giá trị thành hai tập con nên vấn đề ở đây là cần tìm ra sự phân tách tối ưu



Với biến liên tục, có nhiều cách để phân tách chúng thành các nhánh

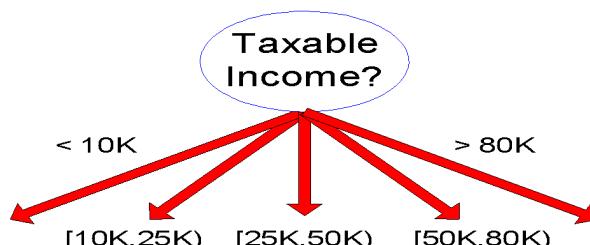
- Rời rạc hóa để tạo ra một thuộc tính loại và có trật tự
 - o Có thể tạo biến tĩnh: rời rạc hóa một lần ngay ban đầu
 - o Có thể tạo biến động: vùng phạm vi có thể được tạo ra bằng cách phân thành các vùng bằng nhau hoặc phân cụm
- Phân thành hai nhánh bằng một quyết định nhị phân ($A < v$) hoặc ($A \geq v$)
 - o Cần cân nhắc tất cả các cách phân nhánh có thể và tìm thấy cách tốt nhất
 - o Cách này có thể cần thêm nhiều tính toán

Ví dụ về tạo phân nhánh trên các thuộc tính liên tục được thể hiện trong hình vẽ dưới đây. Thuộc tính thu nhập có thuế là một thuộc tính liên tục, nếu dùng nó làm thuộc tính để phân nhánh trong cây thì cần phải rời rạc hóa nó. Có hai cách

- Phân nhánh nhị phân hay làm hai nhánh thì dùng ngưỡng 80000 (80K) để tạo điều kiện kiểm tra: Taxable Income > 80K
- Phân nhiều nhánh thì dùng các khoảng ngưỡng khác nhau để tạo điều kiện kiểm tra như trong hình vẽ

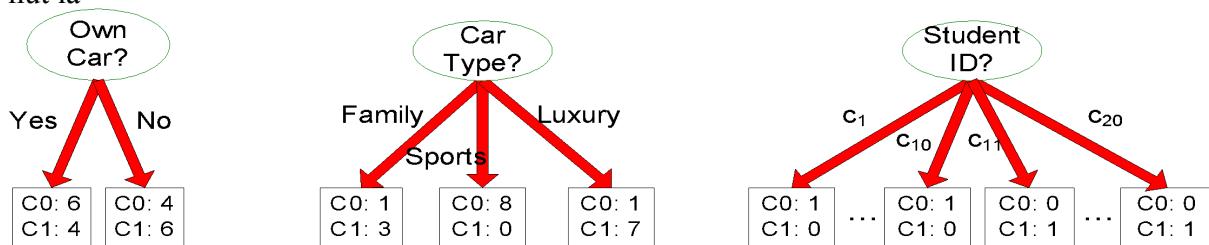


(i) Phân 2 nhánh



(ii) Phân nhiều nhánh

Tiếp đến, chúng ta cùng xem xét làm thế nào để chọn được phân nhánh tốt nhất cho cây quyết định. Xét một ví dụ để thấy rõ được điều đó: sử dụng bảng dữ liệu ở trên, trước khi phân nhánh có 10 bản ghi thuộc lớp 0, 10 bản ghi thuộc lớp 1. Khi xây dựng cây quyết định thì đầu tiên phải chọn điều kiện để phân nhánh, vấn đề ở đây là điều kiện tốt nhất để phân nhánh. Ví dụ có 3 cách phân nhánh như hình vẽ dưới đây, mỗi cách phân nhánh có số bản ghi của mỗi lớp được ghi lại trong nút lá



Theo cách tiếp cận tham lam, các nút có sự phân nhánh với việc phân phối lớp đồng nhất được ưu tiên hơn. Vì thế chúng ta cần đề cập tới một độ đo tính không đồng nhất của một nút trong cây quyết định. Để tính độ không đồng nhất (hay đồng nhất) của một nút trong cây thì ta cần đếm số bản ghi thuộc mỗi lớp của mỗi nhánh dựa vào điều kiện được xác định ở nút đó. Ví dụ một sự phân nhánh với số lượng bản ghi ở lớp 0 là 5, lớp 1 là 5 được gọi là phân nhánh đồng nhất. Trường hợp phân nhánh mà dẫn tới số bản ghi của lớp 0 là 9, của lớp 1 là 1 trong tổng số 10 bản ghi được gọi là có tính không đồng nhất. Với chiến lược tham lam, sự phân nhánh nào có tính đồng nhất hơn sẽ được lựa chọn hay nói cách khác thì phân nhánh có mức độ không đồng nhất thấp hơn (lộn xộn hơn) được ưu chuộng hơn.

C0: 5
C1: 5

C0: 9
C1: 1

Đo độ không đồng nhất của các nút trong cây quyết định

- Chỉ số GINI
- Độ Entropy
- Lỗi phân loại

Chỉ số GINI của một nút t được tính theo công thức sau đây

$$GINI(t) = 1 - \sum_j [p(j | t)]^2$$

- Trong đó $p(j | t)$ là tần suất của lớp j tại nút t,
- Chỉ số GINI nhận giá trị lớn nhất ($1 - 1/n_c$) khi các bản ghi được phân bổ bằng nhau giữa tất cả các lớp, kéo theo thông tin có ích thấp nhất
- Giá trị GINI nhận giá trị nhỏ nhất (0.0) khi tất cả các bản ghi thuộc một lớp, kéo theo thông tin có ích nhất (vì muốn phân loại thông tin thuộc một lớp nào đó)
- Ví dụ tính chỉ số GINI trong một số trường hợp dưới đây, mỗi bảng được gọi là ma trận đếm để tính chỉ số GINI

C1	0
C2	6
Gini=0.000	

C1	1
C2	5
Gini=0.278	

C1	2
C2	4
Gini=0.444	

C1	3
C2	3
Gini=0.500	

Ví dụ về tính toán GINI được thể hiện như trong hình vẽ dưới đây

C1	0
C2	6

C1	1
C2	5

C1	2
C2	4

Chỉ số GINI được sử dụng trong các thuật toán CART, SLIQ, SPRINT.

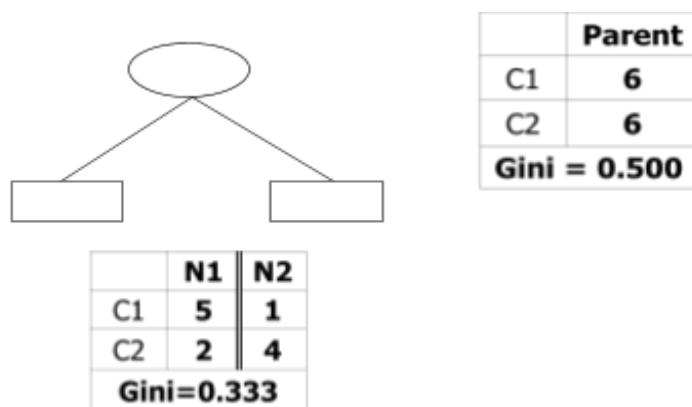
Khi nút p phân chia thành k phần, chất lượng của sự phân chia (split) được tính toán theo công thức

$$GINI_{split} = \sum_{i=1}^k \frac{n_i}{n} GINI(i)$$

trong đó n_i = số các bản ghi tại phần con thứ i,

n = số các bản ghi tại nút p

Với các thuộc tính nhị phân, việc phân nhánh thường chia thành hai nhánh, tính chỉ số GINI cho từng cách phân nhánh. Ví dụ: ta thấy chỉ số GINI của nút cha trong cây lớn hơn chỉ số GINI của toàn bộ phân nhánh.



Với các thuộc tính phân loại, cho mỗi giá trị khác nhau, đếm số bản ghi cho mỗi lớp, sau đó sử dụng ma trận đếm để đưa ra quyết định. Ví dụ sự phân nhánh theo điều kiện của thuộc tính CarType nêu ra ở trên: có thể phân thành nhiều nhánh hoặc phân làm hai nhánh

Phân làm nhiều nhánh có ma trận đếm

Phân làm hai nhánh, ta có hai cách phân chia như sau

CarType		
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

CarType		
	Family	Sports
C1	1	2
C2	4	1
Gini	0.400	

CarType		
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	0.419	

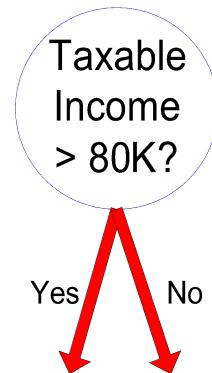
Với thuộc tính liên tục, việc tính toán chỉ số GINI phức tạp hơn

- Nếu muốn phân làm hai nhánh thì cần sử dụng một quyết định nhị phân dựa trên một giá trị ngưỡng, để chia làm hai phạm vi, lớn hơn hoặc bằng ngưỡng và nhỏ hơn giá trị ngưỡng

- Có một số lựa chọn để xác định giá trị ngưỡng nói trên, dựa trên nguyên tắc số lượng ngưỡng cần = số lượng các giá trị phân biệt cần quan tâm theo yêu cầu của bài toán
- Mỗi giá trị ngưỡng sẽ có một ma trận đếm tương ứng với nó để đếm số lượng bản ghi trong mỗi phần của mỗi lớp (có nghĩa là giả sử giá trị ngưỡng là v thì đếm số lượng bản ghi của mỗi lớp mà thuộc tính $A < v$ và $A \geq v$ rồi lưu lại vào ma trận đếm)
- Phương pháp đơn giản để chọn giá trị ngưỡng v tốt nhất: cho mỗi giá trị v , duyệt toàn bộ cơ sở dữ liệu để đếm số lượng tương ứng trong ma trận đếm và tính chỉ số GINI của nó. Một nhược điểm của phương pháp này là tính toán không hiệu quả vì các công việc bị lặp lại nhiều lần.

Ví dụ về việc tính ngưỡng cho bảng dữ liệu được thể hiện trong bảng bên trái với thuộc tính liên tục là Taxable Income (thu nhập chịu thuế). Ngưỡng lựa chọn ở đây là 80K (hay 80000) để phân các bản ghi thành hai nhánh như hình vẽ

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



- Để tính toán một cách có hiệu quả: đối với mỗi thuộc tính chúng ta thực hiện
 - o Sắp xếp các giá trị của thuộc tính
 - o Duyệt tuyến tính những giá trị này, mỗi lần cập nhật ma trận tính và tính chỉ số GINI luôn
 - o Lựa chọn vị trí ngưỡng phân chia sao cho có chỉ số GINI thấp nhất
 - o Ví dụ về việc tính toán kiểu này được thể hiện trong bảng dưới đây trong đó thuộc tính Thu nhập chịu thuế được xét trong nhiều ngưỡng khác nhau để chọn được ngưỡng có chỉ số GINI tương ứng thấp nhất là 0.300 tương ứng với ngưỡng giá trị 97K

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No	No
	Taxable Income										
	60	70	75	85	90	95	100	120	125	172	220
	55	65	72	80	87	92	97	110	122	172	230
	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >
Yes	0 3 0 3 0 3 1 2 2 1 3 0 3 0 3 0 3 0 3 0 3 0										
No	0 7 1 6 2 5 3 4 3 4 3 4 3 4 4 3 5 2 6 1 7 0										
Gini	0.420 0.400 0.375 0.343 0.417 0.400 0.300 0.343 0.375 0.400 0.400 0.420										

Độ đo Entropy dựa trên tiêu chuẩn lượng thông tin chứa đựng trong nó

Entropy tại nút t được tính theo công thức

$$Entropy(t) = -\sum_j p(j|t) \log p(j|t)$$

Trong đó $p(j|t)$ là tần suất của lớp j tại nút t.

- Dùng để đo lường sự đồng nhất của một nút.
 - Nhận giá trị lớn nhất ($\log n_c$) khi các bản ghi được phân bổ bằng nhau giữa tất cả các lớp, kéo theo việc chứa đựng lượng thông tin ít nhất.
 - Nhận giá trị nhỏ nhất (0.0) khi tất cả các bản ghi thuộc một lớp, kéo theo việc chứa đựng nhiều thông tin nhất
- Cách tính toán giá trị Entropy tương tự như tính toán chỉ số GINI
- Ví dụ về việc tính toán Entropy được thể hiện như sau

C1	0
C2	6

C1	1
C2	5

C1	2
C2	4

- Đại lượng Information Gain để đo lượng thông tin thu được của việc phân nhánh được tính theo công thức sau

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^k \frac{n_i}{n} Entropy(i) \right)$$

Trong đó

nút cha p được phân chia thành k phần và n_i là số bản ghi trong phần i

- Dùng để đo việc giảm Entropy hay việc giảm lượng thông tin được chứa đựng trong nút đó do nút đó được phân nhánh. Chúng ta sẽ chọn sự phân nhánh giảm lượng thông tin thấp nhất (hay độ đo GAIN lớn nhất)
- Được sử dụng trong thuật toán ID3 và C4.5
- Nhược điểm: Có xu hướng lựa chọn sự phân chia có nhiều nhánh, mỗi nhánh nhỏ nhưng đồng nhất, không thực tế lắm.
- Để khắc phục nhược điểm của đại lượng trên một đại lượng khác được sử dụng có tên là tỉ lệ lượng thông tin có công thức sau

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO} \quad SplitINFO = -\sum_{i=1}^k \frac{n_i}{n} \log \frac{n_i}{n}$$

node cha p được phân chia thành k phần; với n_i là số bản ghi trong phần thứ (i)

- Đại lượng này tính tỉ số lượng thông tin đạt được thêm nhờ có sự phân nhánh với lượng thông tin chứa đựng trong nút đó, điều chỉnh đại lượng Information Gain bằng entropy của sự phân chia (SplitINFO). Với công thức này, sự phân chia có Entropy cao hơn (số lượng lớn các nhánh nhưng mỗi nhánh nhỏ) bị cấm.
- Đại lượng này được sử dụng trong thuật toán C4.5, khắc phục được nhược điểm của đại lượng Information Gain.

Độ đo lỗi phân loại

- Lỗi phân loại tại một nút t được tính bằng công thức $Error(t) = 1 - \max_i P(i | t)$
- Dùng để đo lỗi phân loại tạo ra bởi một nút.
 - Nhận giá trị lớn nhất ($1 - 1/n_c$) khi các bản ghi được phân bố bằng nhau giữa tất cả các lớp, dẫn đến việc chứa đựng thông tin có ích thấp nhất
 - Nhận giá trị nhỏ nhất (0.0) khi tất cả các bản ghi thuộc một lớp, dẫn đến việc chứa đựng thông tin có ích nhiều nhất.
- Ví dụ về việc tính toán lỗi được thể hiện ở hình vẽ dưới đây

C1	0
C2	6

C1	1
C2	5

C1	2
C2	4

Điều kiện dừng phân nhánh trong cây quyết định

Khi xây dựng cây quyết định, chúng ta liên tiếp xác định các nút trong và sự phân chia các tập bản ghi huấn luyện tại các nút trong đó. Vấn đề tiếp theo chúng ta bàn tới là khi nào thì kết thúc quá trình phân nhánh cho cây hay nói cách khác thuật toán Hunt dừng khi nào.

Các điều kiện để dừng không xây dựng cây quyết định một cách quy nạp như sau:

- Dừng phân nhánh một nút khi tất cả các bản ghi là cùng một lớp
- Dừng phân nhánh một nút khi tất cả các bản ghi có các giá trị thuộc tính giống nhau
- Kết thúc sớm trong một số trường hợp đặc biệt (sẽ được thảo luận sau)

Đánh giá phương pháp phân loại dựa trên cây quyết định

Ưu điểm:

- Chi phí xây dựng không đắt
- Vô cùng nhanh trong việc phân loại các bản ghi chưa biết
- Dễ dàng để giải thích ý nghĩa cho cây có kích thước nhỏ
- Độ chính xác có thể so sánh ngang bằng với các kỹ thuật phân loại khác nhưng chỉ với những tập dữ liệu đơn giản

Một thuật toán phân loại dựa trên cây quyết định là thuật toán C4.5

Thuật toán này có những đặc điểm như sau

- Là một dạng cây được xây dựng đơn giản theo kiểu chiết sâu trước
- Sử dụng độ đo Information Gain
- Thực hiện công việc sắp xếp các thuộc tính liên tục tại mỗi nút.
- Cần toàn bộ dữ liệu lưu được trong bộ nhớ.
- Không phù hợp với tập dữ liệu lớn.

Có thể lấy phần mềm từ một kết nối trang Web trên mạng Internet để thử nghiệm

Đánh giá các mô hình phân loại

Độ đo để đánh giá hiệu quả của một mô hình

Để trả lời câu hỏi làm thế nào để đánh giá một mô hình, chúng ta cần xác định một độ đo tính hiệu quả của mô hình.

Việc đánh giá một mô hình dựa vào khả năng dự đoán của nó hơn là tốc độ phân loại hay xây dựng mô hình và khả năng mở rộng kích cỡ của mô hình.

Sử dụng một ma trận đo độ lộn xộn (Confusion Matrix) để đo hiệu quả của việc phân loại. Ma trận đó như sau:

		Lớp dự đoán	
		Lớp =Yes	Lớp =No
Lớp thực tế	Lớp =Yes	a (TP)	b (FN)
	Lớp =No	c (FP)	d (TN)

Trong đó

a là số trường hợp phân lớp trong thực tế và dự đoán bằng mô hình cùng nhận giá trị yes hay được gọi là true positive

b là số trường hợp phân lớp trong thực tế là yes nhưng dự đoán bằng mô hình thuộc lớp No, hay được gọi là false negative

c là số trường hợp phân lớp trong thực tế là No nhưng dự đoán bằng mô hình thuộc lớp Yes, hay được gọi là false positive

d là số trường hợp phân lớp trong thực tế là No và dự đoán bằng mô hình thuộc lớp No, hay được gọi là true negative

Độ đo được sử dụng rộng rãi nhất là độ chính xác được tính theo công thức sau

$$\text{Độ chính xác} = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + TN + FP + FN}$$

Tuy vậy độ chính xác có hạn chế nhất định, xem xét bài toán phân loại thành hai lớp để thấy rõ điều đó. Số phần tử thuộc lớp 0 là 9990 và số phần tử thuộc lớp 1 là 10. Như vậy nếu mô hình dự đoán mọi trường hợp đều là thuộc lớp 0 thì độ chính xác là $9990/10000=99,9\%$, là một con số rất tốt đối với một mô hình phân lớp bất kỳ nhưng thực tế lại không đúng vì chẳng có phần tử nào được xếp vào lớp 1 cả.

Để đo tính hiệu quả của một mô hình có thể dùng một ma trận chi phí để tính toán. Ma trận chi phí được thể hiện như sau (khá giống với ma trận lộn xộn) trong đó $C(i|j)$: Chi phí phân loại lỗi lớp i trong thực tế thành là lớp j

		Lớp dự đoán	
		C(i j)	Lớp=Yes
Lớp thực tế	C(i j)	Lớp=Yes	Lớp>No
	Lớp=Yes	C(Yes Yes) =p	C(No Yes)=q
Lớp=No	C(Yes No) =q	C(No No)=p	

Xét một ví dụ: cho ma trận chi phí được thể hiện trong bảng đầu tiên dưới đây và ma trận chi phí lộn xộn của hai mô hình M_1 và M_2 như sau, dựa vào 3 bảng đó ta tính được độ chính xác và chi phí của từng mô hình.

Ma trận chi phí	Lớp dự đoán		
	C(i j)	+	-
Lớp thực tế	+	-1	100
	-	1	0

Mô hình M_1	Lớp dự đoán		
		+	-
Lớp dự đoán	+	150	40
	-	60	250

Mô hình M_2	Lớp dự đoán		
		+	-
Lớp thực tế	+	250	45
	-	5	200

Dựa vào kết quả trên, khó có thể xác định mô hình nào tốt hơn, mô hình thứ nhất có độ chính xác thấp hơn nhưng chi phí cần cũng ít hơn, mô hình thứ hai có độ chính xác cao hơn và chi phí cũng cao hơn. Nếu một mô hình có độ chính xác cao hơn nhưng chi phí thấp hơn thì được coi là tốt hơn mô hình kia.

Việc tính toán chi phí và độ chính xác có thể tính dựa vào nhau, chúng tỷ lệ thuận với nhau nếu $C(Yes|No) = C(No|Yes) = q$ và $C(Yes|Yes) = C(No|No) = p$.

Gọi $N = a + b + c + d$ thì độ chính xác = $(a + d) / N$

chi phí = $p(a + d) + q(b + c) = p(a + d) + q(N - a - d) = qN - (q - p)(a + d) = N[q - (q-p) \times$
độ chính xác]

Một số độ đo khác liên quan tới chi phí

$$\text{Precision}(p) = \frac{a}{a+c}$$

$$\text{Recall}(r) = \frac{a}{a+b}$$

$$\text{F-measure}(F) = \frac{2rp}{r+p} = \frac{2a}{2a+b+c}$$

Độ chính xác Precision thể hiện mối quan tâm thiên về C(Yes|Yes) & C(Yes|No)

Độ đo Recall thể hiện mối quan tâm thiên về C(Yes|Yes) & C(No|Yes)

Độ đo F thể hiện mối quan tâm thiên về C(Yes|Yes) & C(No|Yes) và C(Yes|No) (tất cả loại trừ C(No|No))

Một cách đo nữa sử dụng trọng số trong trường hợp vai trò của a, b, c, d không như nhau, trường hợp nào có vai trò quyết định hơn trong việc đánh giá độ chính xác trong thực tế thì được đánh trọng số cao hơn. Độ đo được tính theo công thức sau

Độ chính xác có trọng số $= \frac{w_1a + w_4d}{w_1a + w_2b + w_3c + w_4d}$ trong đó w là các trọng số tương ứng với a,b,
c,d

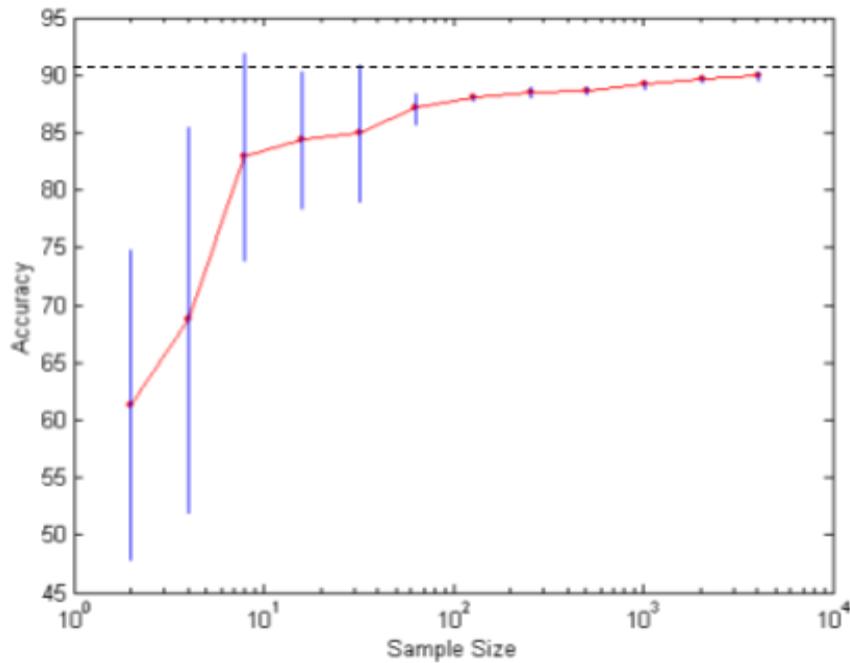
Các phương pháp đánh giá hiệu quả

Trả lời câu hỏi làm thế nào để đạt được ước lượng đáng tin cậy?

Hiệu quả của một mô hình còn phụ thuộc vào các yếu tố khác ngoài thuật toán học ra, đó là các yếu tố:

- Sự phân bố các phân tử trong một lớp hay số lượng các phân tử khác nhau trong mỗi lớp và sự phân bố chúng trong các lớp khác nhau có ảnh hưởng rất lớn đối với hiệu quả phân lớp của một mô hình
- Chi phí của sự phân sai lớp
- Kích cỡ của tập dữ liệu dùng để huấn luyện và tập dùng để kiểm thử

Đường cong học trong hình vẽ dưới đây thể độ chính xác thay đổi thế nào khi kích cỡ của tập huấn luyện thay đổi. Để vẽ được đường cong này, chúng ta cần một lịch lấy mẫu để thay đổi kích cỡ lấy mẫu bằng cách lấy mẫu số học hoặc lấy mẫu dạng hình học. Hiệu quả của kích cỡ lấy mẫu nhỏ thể hiện trong sự sai khác về ước lượng, phương sai trong ước lượng



Các phương pháp ước lượng

- Phương pháp giữ lại một phần (holdout): dùng 2/3 số bản ghi của tập dữ liệu cho việc huấn luyện và 1/3 còn lại cho việc kiểm thử. Việc lựa chọn 2/3 lượng bản ghi nào là ngẫu nhiên
- Phương pháp lấy mẫu con: là cách dùng lại phương pháp giữ một phần trình bày ở trên lặp đi lặp lại
- Phương pháp xác nhận chéo:
 - o Chia dữ liệu thành k tập con không giao nhau
 - o Lặp lại k lần công việc sau: huấn luyện mô hình trên k-1 tập con và kiểm thử mô hình trên tập con thứ k còn lại
 - o Trong trường hợp đặc biệt k=n có nghĩa là tập dữ liệu được chia thành n tập con (n là số bản ghi trong tập dữ liệu), mỗi lần huấn luyện sẽ dùng n-1 bản ghi và kiểm thử trên bản ghi còn lại. Lặp công việc đó n lần cho toàn bộ bản ghi trong tập dữ liệu.
- Phương pháp lấy mẫu: dùng các tập con của tập dữ liệu thu được bằng cách áp dụng các phương pháp lấy mẫu để huấn luyện và kiểm thử mô hình. Cần quan tâm tới các vấn đề lấy mẫu với số lượng quá nhiều hoặc quá ít, khiến cho chất lượng huấn luyện mô hình và kiểm thử không cao.

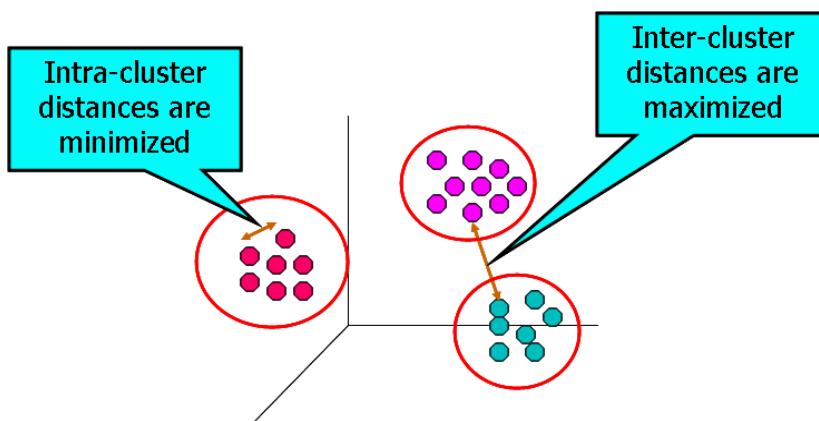
- Phương pháp khởi động nhanh: là cách lấy mẫu nhưng có thay thế

4.4 Phương pháp phân nhóm và phân đoạn

Khái niệm về phân tích phân cụm

Cụm là một tập hợp các đối tượng dữ liệu giống nhau theo một tiêu chí nào đó, hai đối tượng giống nhau thì nằm trong cùng một cụm, hai đối tượng khác nhau thì nằm ở hai cụm khác nhau.

Việc phân cụm các điểm thành 3 cụm (màu đỏ, màu tím và màu xanh) được mô tả trong hình dưới đây trong đó chúng ta thấy khoảng cách giữa các phần tử trong cùng một cụm được tối thiểu hóa, còn khoảng cách giữa hai phần tử trong hai cụm khác nhau cần được tối đa hóa.



Phân tích phân cụm là việc gộp nhóm một tập các đối tượng dữ liệu vào các cụm. Việc phân cụm là một cách phân loại không giám sát có nghĩa là các loại không được xác định trước và cũng không biết là có bao nhiêu loại.

Các ứng dụng điển hình của việc phân cụm là

- tìm hiểu bản chất bên trong của dữ liệu
- được coi như là một bước tiền xử lý dữ liệu trước khi khai phá
- được sử dụng để nhận dạng mẫu
- Dùng trong phân tích dữ liệu không gian:
 - tạo các bản đồ trong hệ thống định vị toàn cầu bằng cách phân cụm các không gian đặc tính
 - Phát hiện những cụm không gian và giải thích ý nghĩa của chúng trong việc khai phá dữ liệu không gian
- Dùng trong lĩnh vực xử lý ảnh
- Dùng trong khoa học kinh tế đặc biệt là trong nghiên cứu thị trường
- Dùng trong khám phá hệ thống Web toàn cầu như

- Phân loại các tài liệu trên các trang Web
- Phân cụm các dữ liệu Weblog để phát hiện ra các nhóm mẫu truy nhập giống nhau

Các ví dụ của ứng dụng phân cụm

- Ứng dụng trong tiếp thị: giúp cho những người tiếp thị phát hiện ra những nhóm đặc biệt trong dữ liệu khách hàng của họ sau đó sử dụng những tri thức này để phát triển các chương trình tiếp thị có mục tiêu
- Ứng dụng trong việc sử dụng đất: Xác định các vùng sử dụng đất giống nhau trong cơ sở dữ liệu quan sát toàn trái đất
- Ứng dụng trong bảo hiểm: Xác định các nhóm người có bảo hiểm ô tô với chi phí trung bình được chi trả bảo hiểm cao.
- Ứng dụng trong lập kế hoạch cho thành phố: Xác định các nhóm nhà dựa trên kiểu nhà, giá trị và vị trí địa lý ngôi nhà của họ
- Nghiên cứu động đất: địa chấn của các trận động đất được phân cụm dựa trên các lối lục địa

Khái niệm phân cụm tốt

Một phương pháp phân cụm tốt sẽ sinh ra các phân cụm có chất lượng cao trong đó hai phần tử cùng một cụm có độ giống nhau cao sự giống nhau của hai phần tử bất kỳ khác cụm

Chất lượng của kết quả phân cụm phụ thuộc vào độ đo sự giống nhau được sử dụng cho phương pháp phân cụm và việc cài đặt độ đo đó. Chất lượng của một phương pháp phân cụm được đo bởi khả năng phát hiện một hoặc tất cả các mẫu tiềm ẩn trong dữ liệu.

Những yêu cầu của việc phân cụm trong khai phá dữ liệu

Tồn tại rất nhiều phương pháp phân cụm trong khai phá dữ liệu, muốn đề xuất một phương pháp mới phải thỏa mãn các đặc tính sau đây

- Tính mở rộng về kích cỡ
- Khả năng phân cụm với những kiểu thuộc tính khác nhau
- Có thể phát hiện những cụm với một hình thù bất kỳ
- Đòi hỏi yêu cầu nhỏ nhất về tri thức miền dữ liệu để xác định các tham số đầu vào
- Có khả năng giải quyết với nhiều và các thành phần ngoại lai
- Trật tự của các bản ghi đầu vào cho phương pháp không làm ảnh hưởng tới kết quả của việc phân cụm
- Có khả năng làm việc dữ liệu có nhiều chiều
- Cho phép chấp nhận thêm các ràng buộc do người sử dụng định nghĩa

- Có tính phiên dịch và sử dụng được

Độ đo trong phân cụm

Cấu trúc dữ liệu được sử dụng trong việc phân cụm

Trong phân cụm, dữ liệu được lưu trữ dưới hai dạng: dạng ma trận dữ liệu hai chiều có n hàng và p cột trong đó n là số phần tử trong tập dữ liệu đang xét và p là số đặc tính quan tâm khi phân cụm của một phần tử nào đó và dạng ma trận sự khác nhau

$$\begin{bmatrix} x_{11} & \dots & x_{1f} & \dots & x_{1p} \\ \dots & \dots & \dots & \dots & \dots \\ x_{i1} & \dots & x_{if} & \dots & x_{ip} \\ \dots & \dots & \dots & \dots & \dots \\ x_{n1} & \dots & x_{nf} & \dots & x_{np} \end{bmatrix} \quad \begin{bmatrix} 0 \\ d(2,1) & 0 \\ d(3,1) & d(3,2) & 0 \\ \vdots & \vdots & \vdots \\ d(n,1) & d(n,2) & \dots & \dots & 0 \end{bmatrix}$$

Trong ma trận thể hiện sự khác nhau, phần tử $d(i,j)$ thể hiện khoảng cách hay sự khác nhau giữa phần tử thứ i và thứ j. Ma trận này là ma trận đối xứng vì sự khác nhau giữa phần tử i và phần tử j cũng là giữa j và i, đồng thời đường chéo của ma trận là 0 vì là khoảng cách của một phần tử i và chính nó.

Đo sự giống nhau

Đơn vị đo lường để đo sự giống nhau hay sự khác nhau của hai phần tử trong tập dữ liệu đang xét được biểu diễn bởi một hàm khoảng cách, là một độ đo điển hình $d(i,j)$ là khoảng cách giữa phần tử (i) và phần tử (j)

Để đánh giá chất lượng của một cụm, một hàm chất lượng riêng biệt được sử dụng tới. Việc định nghĩa hàm khoảng cách thường rất khác nhau đối với các biến phạm vi, phân khoảng, biến nhị phân, phân loại, biến có trật tự và biến tỉ lệ.

Các trọng số có thể được sử dụng với các biến khác nhau dựa trên các loại ứng dụng và ngữ nghĩa của dữ liệu. Rất khó để định nghĩa thế nào là “đủ giống nhau” và “đủ tốt” trong quá trình xác định sự giống nhau giữa các phần tử trong tập dữ liệu. Câu trả lời thường mang tính chủ quan, khác nhau đối với từng đối tượng và cảm nhận của từng đối tượng.

Đối với các biến có giá trị nằm trong một khoảng nào đó, chúng ta cần chuẩn hóa dữ liệu trước khi thực hiện phân cụm. Sở dĩ như vậy là vì khiến cho các thuật toán phân cụm đưa ra kết quả chính xác hơn. Chuẩn hóa bằng cách

- Tính toán trung bình phương sai bình phương

$$s_f = \frac{1}{n}(|x_{1f} - m_f|^2 + |x_{2f} - m_f|^2 + \dots + |x_{nf} - m_f|^2)$$

trong đó $m_f = \frac{1}{n}(x_{1f} + x_{2f} + \dots + x_{nf})$ sau đó tính độ đo được chuẩn hóa (z-score) của mỗi phần tử trong ma trận hai chiều

$$z_{if} = \frac{x_{if} - m_f}{s_f}$$

Sử dụng phương sai trung bình tuyệt đối (không phải phương sai trung bình trung bình ở trên) có thể khiến thuật toán được thực hiện nhanh chóng hơn sử dụng phương sai chuẩn.

Sự giống nhau và khác nhau giữa các đối tượng dữ liệu

Các khoảng cách thông thường được sử dụng để đo sự giống nhau hay khác nhau giữa hai đối tượng dữ liệu

Một số các khoảng cách phổ biến thường được dùng bao gồm

- Khoảng cách Minkowski được tính theo công thức

$$d(i, j) = \sqrt[q]{(|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \dots + |x_{ip} - x_{jp}|^q)}$$

Trong đó

$i = (x_{i1}, x_{i2}, \dots, x_{ip})$ và $j = (x_{j1}, x_{j2}, \dots, x_{jp})$ là hai đối tượng dữ liệu có p chiều và q là một số nguyên dương

Nếu $q=1$ thì d được gọi là khoảng cách Manhattan

$$d(i, j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \dots + |x_{ip} - x_{jp}|$$

Nếu $q=2$ thì d là khoảng cách Oclit (Euclit)

$$d(i, j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \dots + |x_{ip} - x_{jp}|^2)}$$

Với các thuộc tính

- $d(i, j) \geq 0$
- $d(i, i) = 0$
- $d(i, j) = d(j, i)$
- $d(i, j) \leq d(i, k) + d(k, j)$

Ngoài ra chúng ta có thể sử dụng các khoảng cách có trọng số, các tương hỗ tích mômen có tham số Pearson hoặc các độ đo sự khác nhau khác, miễn là thỏa mãn các thuộc tính khoảng cách trình bày trên.

Độ giống nhau Cosin

Nếu d_1 và d_2 là hai vectơ tài liệu thì $\cos(d_1, d_2) = (d_1 \cdot d_2) / \|d_1\| \|d_2\|$ trong đó \cdot thể hiện phép toán nhân vectơ và $\|d\|$ là độ dài của vectơ d

Ví dụ: $d_1 = 3 \ 2 \ 0 \ 5 \ 0 \ 0 \ 0 \ 2 \ 0 \ 0$

$$d_2 = \mathbf{1} \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 2$$

$$d_1 \cdot d_2 = 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5$$

$$\|d_1\| = (3*3+2*2+0*0+5*5+0*0+0*0+0*0+2*2+0*0+0*0)^{0.5} = (42)^{0.5} = 6.481$$

$$\|d_2\| = (1*1+0*0+0*0+0*0+0*0+0*0+0*0+1*1+0*0+2*2)^{0.5} = (6)^{0.5} = 2.245$$

$$\cos(d_1, d_2) = .3150$$

- Độ giống nhau của biến nhị phân

Bảng sau được sử dụng để tính độ giống nhau của biến nhị phân, thành phần của mỗi phần tử trong bảng là số thuộc tính của đối tượng i và đối tượng j nhận giá trị là 0 hay 1. Chẳng hạn a là số thuộc tính của đối tượng i và j có giá trị là 1, còn b là số thuộc tính của đối tượng i là 1 và còn của đối tượng j là 0.

	1	0	sum
1	a	b	a+b
0	c	d	c+d
sum	a+c	b+d	p

Đo độ tương đồng giữa hai đối tượng, có thể dùng một khoảng cách đơn giản sau (là bất biến nếu biến nhị phân là đối xứng)

$$d(i, j) = \frac{b+c}{a+b+c+d}$$

Nếu biến nhị phân là không đối xứng thì độ tương quan là không bất biến, được gọi là độ tương quan Jaccard có công thức tính như sau

$$d(i, j) = \frac{b+c}{a+b+c}$$

Ví dụ: Cho một bảng dữ liệu như sau

Name	Gender	Fever	Cough	Test-1	Test-2	Test-3	Test-4
Jack	M	Y	N	P	N	N	N
Mary	F	Y	N	P	N	P	N
Jim	M	Y	P	N	N	N	N

Trong đó Gender là một thuộc tính đối xứng thể hiện giới tính của các đối tượng có tên trong thuộc tính Name đang xét. Các thuộc tính còn lại lần lượt là sự biểu hiện có sốt, có ho không và kết quả xét nghiệm 1, 2, 3, 4. Tất cả các thuộc tính còn lại này đều thuộc loại nhị phân không đối xứng. Giả sử giá trị Y (yes-có) và P (positive- dương tính) được đặt là 1 và giá trị N (No- không hay âm tính) được đặt là 0 thì ta có các khoảng cách Jaccard như sau

$$d(jack, mary) = \frac{0+1}{2+0+1} = 0.33$$

$$d(jack, jim) = \frac{1+1}{1+1+1} = 0.67$$

$$d(jim, mary) = \frac{1+2}{1+1+2} = 0.75$$

- Độ giống nhau của biến tên

Là một sự tổng quát hóa của biến nhị phân trong đó biến có thể có nhiều hơn hai trạng thái (trong khi biến nhị phân chỉ có hai trạng thái là 0 và 1), ví dụ như biến dạng tên có thể nhận các giá trị màu sắc đỏ, vàng, xanh nước biển và xanh lá cây

Để đo độ giống nhau của những biến này chúng ta có thể dùng một số cách để xuất sau đây

Cách 1: so sánh giống nhau một cách đơn giản

$d(i, j) = \frac{p - m}{P}$ Trong đó p là tổng số thuộc tính của mỗi đối tượng, còn m là số thuộc tính của hai đối tượng i và j có cùng giá trị.

Cách 2: sử dụng một số lượng lớn các biến nhị phân bằng cách tạo ra một biến nhị phân mới cho mỗi một trong M trạng thái của biến tên.

- Độ đo giống nhau cho biến trật tự:

Một biến có trật tự có thể là rời rạc hoặc liên tục. Trật tự của biến là rất quan trọng ví dụ như biến đó thể hiện sự phân bậc của một đối tượng.

Chúng ta có thể coi loại biến có trật tự này như dạng biến trong một khoảng phạm vi bằng cách

- o Thay thế x_{if} bằng cấp bậc của nó $r_{if} \in \{1, \dots, M_f\}$
- o Ánh xạ phạm vi của mỗi biến vào khoảng [0,1] bằng cách thay thế đối tượng thứ i trong biến thứ f bằng z-score của nó $z_{if} = \frac{r_{if} - 1}{M_f - 1}$
- o Tính toán sự khác nhau sử dụng cách thức dùng cho các biến trong khoảng phạm vi.

- Độ đo cho biến tỉ lệ

Biến tỉ lệ là một đơn vị đo lường dương trên một phạm vi phi tuyến hoặc dạng lũy thừa xấp xỉ ví dụ như Ae^{Bt} hoặc Ae^{-Bt}

Cách thức để tính toán độ đo sự giống/khác nhau của các biến loại này như sau:

- Coi chúng như các biến phạm vi tuy không phải là một lựa chọn tốt
- Áp dụng một số chuyển đổi logarit $y_{if} = \log(x_{if})$

- Coi chung như dữ liệu có trật tự và liên tục và coi cấp bậc của chúng như khoảng phạm vi.
- Độ đo cho loại dữ liệu hỗn hợp (trộn của nhiều loại khác nhau)

Một cơ sở dữ liệu có thể chứa tất cả sáu loại biến: nhị phân đối xứng, nhị phân không đối xứng, loại tên thường, loại có trật tự, loại tỉ lệ và phạm vi.

Chúng ta có thể sử dụng một công thức có trọng số để kết hợp các hiệu quả của chúng vào với nhau theo công thức sau

$$d(i, j) = \frac{\sum_{f=1}^P \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^P \delta_{ij}^{(f)}}$$

Trong đó

- f là dạng nhị phân hoặc dạng tên $d_{ij}^{(f)} = 0$ nếu $x_{if} = x_{jf}$, hoặc $d_{ij}^{(f)} = 1$
- f là dạng phạm vi: sử dụng khoảng cách chuẩn
- f là dạng trật tự hoặc khoảng tỉ lệ thì cần

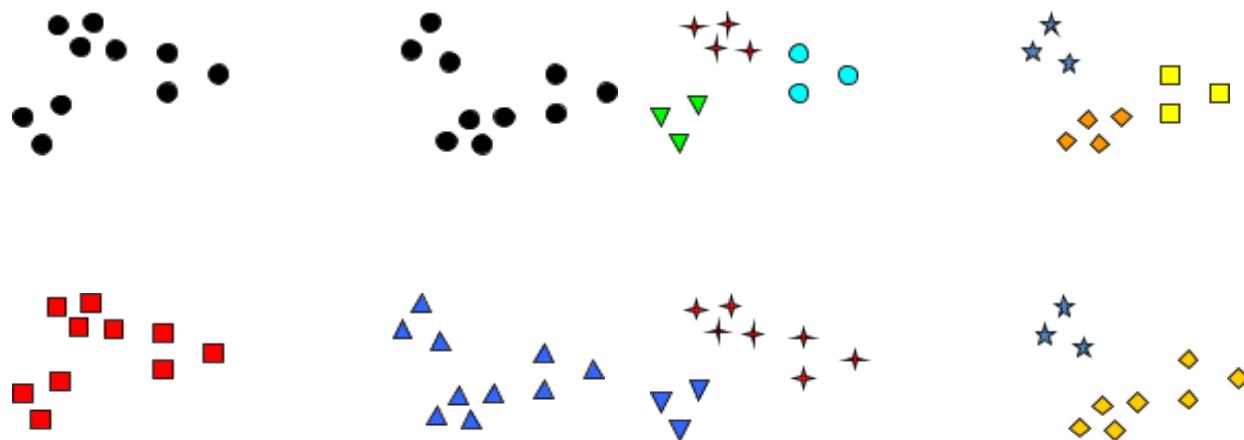
o tính cấp bậc r_{if} và

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

Phân loại phân cụm

Khái niệm không rõ ràng về một cụm trong không gian các điểm dữ liệu

Đây là một vấn đề trong quá trình phân cụm: với một tập hợp các điểm trong không gian dữ liệu, có nhiều cách phân cụm với các số lượng cụm khác nhau và với các tiêu chí khác nhau. Xem ví dụ sau đây để minh họa điều đó với một không gian dữ liệu ban đầu có thể phân thành 2 cụm, 4 cụm hay 6 cụm như hình vẽ dưới đây (mỗi phần tử trong cùng một cụm được thể hiện bởi một hình giống nhau)

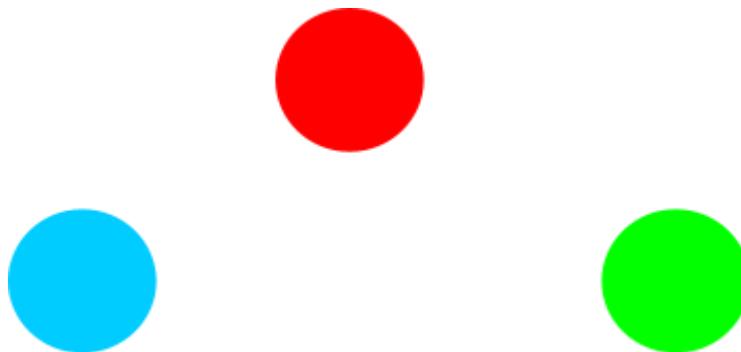


Sự khác biệt giữa các loại phân cụm

- Loại trừ và không loại trừ: trong phân cụm không loại trừ, các điểm trong không gian dữ liệu có thể thuộc nhiều phân cụm và có thể đại diện trong nhiều lớp hoặc là các điểm nằm trên biên giới giữa các lớp.
- Cụm mờ hoặc không mờ:
 - o Trong phân cụm mờ, một điểm thuộc vào một cụm nào đó với một trọng số (xác suất điểm thuộc cụm đó) giữa 0 và 1
 - o Tổng các trọng số phải bằng 1
 - o Các phân cụm xác suất có đặc tính giống nhau
- Cụm một phần hoặc toàn bộ: trong một số trường hợp chúng ta chỉ muốn phân cụm một vài dữ liệu chứ không phải toàn bộ dữ liệu
- Hỗn tạp và đồng điệu: các cụm có kích cỡ, hình dạng và mật độ khác nhau thuộc loại hỗn tạp, còn nếu tương đồng nhau thì thuộc loại đồng điệu

Các loại cụm

- Cụm phân chia rõ ràng: Là loại cụm bao gồm một tập các điểm mà mọi điểm trong một cụm gần (hay giống) mọi điểm khác trong cụm đó hơn tới một điểm bất kỳ trong một cụm khác. Ví dụ về loại cụm phân chia rõ ràng được thể hiện trong hình vẽ sau với 3 cụm được biểu diễn bởi 3 hình tròn màu đỏ, màu xanh da trời và màu xanh nõn chuối sau



- Cụm có tâm điểm:
 - o Là loại cụm bao gồm một tập các đối tượng sao cho một đối tượng trong một cụm là gần (hay giống) “trung tâm” của cụm đó hơn là “trung tâm” của một cụm bất kỳ nào khác
 - o Trung tâm của một cụm thường được gọi là “tâm điểm” (centroid), là trung bình của tất cả các điểm trong một cụm hoặc gọi là medoid, là điểm đại diện nhất của một cụm.

- Ví dụ về loại cụm dựa vào tâm điểm được mô tả trong hình vẽ dưới đây trong đó có 4 cụm được thể hiện bằng 4 hình tròn 4 màu khác nhau



- Cụm dựa trên sự tiếp giáp (hay hàng xóm gần nhất hoặc bắc cầu)
 - Là loại cụm bao gồm tập hợp các điểm sao cho một điểm trong một cụm gần (hay giống) một hoặc nhiều điểm khác trong cùng cụm đó hơn là tới một điểm bất kỳ không nằm trong cụm ấy.
 - Ví dụ về loại cụm này được thể hiện trong hình vẽ dưới đây trong đó mỗi màu thể hiện một cụm

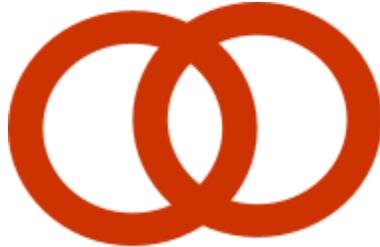


- Cụm dựa trên mật độ:
 - Là loại cụm trong đó một cụm là một vùng các điểm dày đặc, tách biệt với các vùng có mật độ điểm thưa thớt, từ các vùng có mật độ dày đặc.
 - Được sử dụng khi các cụm có hình dạng đặc biệt hoặc gắn kết với nhau hoặc khi có nhiều và các phần tử ngoại lai xuất hiện trong tập dữ liệu
 - Ví dụ về loại cụm này được thể hiện trong hình vẽ dưới đây trong đó mỗi màu thể hiện một cụm tách nhau



- Cụm theo thuộc tính hoặc khái niệm

- Là các cụm có chia sẻ một thuộc tính chung hoặc thể hiện một khái niệm cụ thể nào đó
- Ví dụ về loại cụm này được mô tả như hình vẽ dưới đây



Phương pháp phân cụm

Các cách tiếp cận phân cụm chính

- Các thuật toán phân mảnh: xây dựng nhiều mảnh khác nhau sau đó đánh giá chúng theo một tiêu chí nào đó
- Các thuật toán phân cấp: tạo một sự phân chia theo cấp của một tập các dữ liệu (hoặc đối tượng) sử dụng tiêu chí nào đó
- Các thuật toán dựa trên mật độ: dựa trên các hàm kết nối và hàm mật độ để phân cụm các đối tượng dữ liệu
- Các thuật toán dựa trên lưới: dựa trên một cấu trúc lõi đa mức
- Các thuật toán dựa trên mô hình: Một mô hình được giả thiết mỗi một cụm và ý tưởng là tìm ra một mô hình phù hợp nhất với mỗi cụm.

Trong phạm vi bài giảng này chúng ta chỉ xem xét một loại nhóm thuật toán đầu tiên.

Phương pháp phân cụm K-means

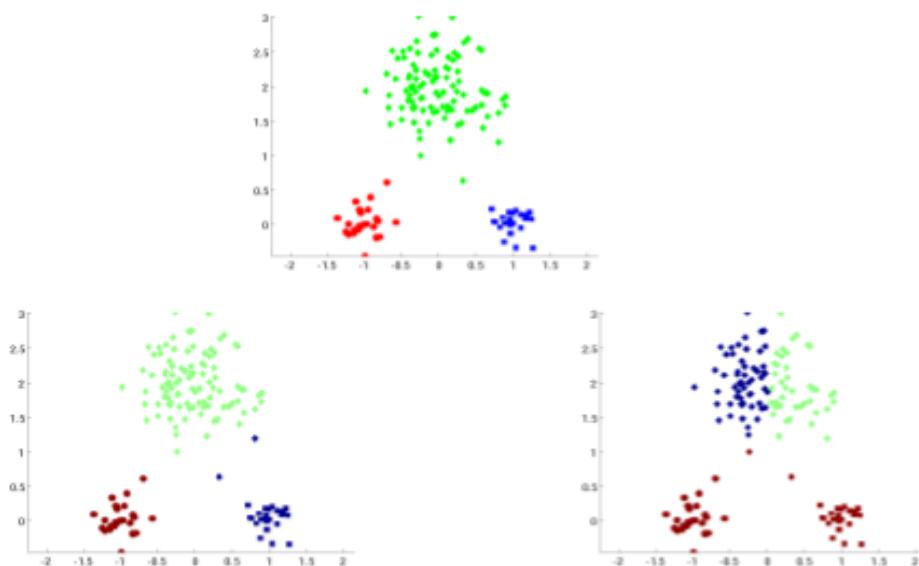
Phương pháp này có một số đặc tính sau:

- Đây là một cách tiếp cận phân cụm dạng phân mảnh
 - Mỗi cụm liên quan tới một tâm điểm (được gọi là centroid)
 - Mỗi điểm sẽ được gán tới một cụm mà có tâm điểm gần nó nhất
 - Số lượng các cụm là K, phải được xác định cụ thể từ đầu
 - Ý tưởng của thuật toán cơ bản rất đơn giản được thể hiện dưới dạng mã giả dưới đây
1. Lựa chọn K điểm làm tâm điểm khởi tạo của các cụm
 2. Lặp các công việc sau
 3. Hình thành K cụm bằng cách gán tất cả các điểm tới tâm điểm gần nó nhất
 4. Tính toán lại tâm điểm của mỗi cụm
 5. Cho đến khi các tâm điểm không thay đổi nữa

Một số nhận xét đối với K-means

- Tâm điểm khởi tạo thường được chọn một cách ngẫu nhiên vì thế trên thực tế chúng ta sẽ thấy các cụm được sinh ra thay đổi trong các lần chạy thuật toán khác nhau
- Tâm điểm thường là kết quả trung bình của các điểm trong cụm
- Đặc tính “gần nhau” được đo bằng khoảng cách Euclidean, sự giống nhau Cosine, độ tương hỗ, v.v...
- Thuật toán K-mean sẽ hội tụ cho hầu hết các độ đo độ tương tự phổ biến được đề cập đến ở trên
- Hầu hết sự hội tụ xảy ra trong một vài vòng lặp lại đầu tiên.
 - o Thông thường điều kiện dừng được chuyển thành “tới khi chỉ còn một ít điểm thay đổi cluster”
- Độ phức tạp của thuật toán là $O(n * K * I * d)$ trong đó $n =$ số điểm trong không gian dữ liệu đang xét, $K =$ số cluster được xác định khi khởi tạo, $I =$ số vòng lặp, $d =$ số thuộc tính của dữ liệu

Khi chạy thuật toán K-mean, kết quả có thể khác nhau trong các lần chạy cho dù chọn số cụm như nhau bởi tâm điểm khởi tạo được chọn ngẫu nhiên nên mỗi lần chạy là được sinh các giá trị khác nhau. Xét ví dụ minh họa trong hình vẽ dưới đây để thấy được các kết quả khác nhau có thể sinh ra khi chạy K-mean trên cùng một tập dữ liệu



Để đánh giá các cụm được tìm thấy bằng phương pháp K-mean chúng ta dùng một độ đo lỗi phô biến nhất là tổng bình phương lỗi (Sum of Squared Error –SSE)

- Đối với mỗi điểm, lỗi được tính là khoảng cách tới cụm gần nhất
- Để tính được SSE, các lỗi tính được ở trên được bình phương lên và lấy tổng của chúng, theo như công thức dưới đây

$$SSE = \sum_{i=1}^K \sum_{x \in C_i} dist^2(m_i, x)$$

Trong đó

- x là một điểm dữ liệu trong cụm C_i và m_i là điểm đại diện cho cluster C_i
- Nếu kết quả chạy thuật toán cho 2 cụm thì chúng ta thường chọn cụm với lỗi nhỏ nhất

Thông thường chúng ta muốn lỗi nhỏ nhất có thể để thu được cách phân cụm tốt nhất có thể. Một cách đơn giản có thể làm giảm SSE là tăng số lượng K cụm, khi K tăng thì SSE sẽ giảm nhưng điều này không có ý nghĩa trong thực tế vì nếu K tăng lên giá trị lớn nhất chính bằng số điểm trong không gian dữ liệu thì lỗi SSE là nhỏ nhất và bằng 0 nhưng không có ý nghĩa. Vì thế lưu ý là một cách phân cụm tốt với số cụm K nhỏ có thể có lỗi SSE nhỏ hơn một phân cụm tồi với số lượng K lớn hơn.

Như đã phân tích ở trên việc lựa chọn tâm điểm khởi tạo có thể gây ảnh hưởng lớn tới kết quả chạy của thuật toán (tới thời gian hội tụ, và các kết quả phân cụm khác nhau). Một số giải pháp có thể thực hiện để giải quyết vấn đề này

- Chạy nhiều lần
- Lấy mẫu và sử dụng phương pháp phân cụm dạng phân cấp để xác định các tâm điểm khởi tạo ban đầu
- Có thể lựa chọn nhiều hơn K tâm điểm ban đầu và sau đó lựa chọn trong số những tâm khởi tạo này với độ phân tách rộng nhất
- Dùng các phương pháp hậu xử lý dữ liệu (xử lý sau khi tìm được các cụm)
- Dùng phương pháp K-means dạng phân đôi: không dễ bị các vấn đề thường xảy ra khi khởi tạo.

Giải quyết vấn đề có cụm rỗng trong kết quả phân cụm

- Thuật toán K-mean có thể cho kết quả là những cụm rỗng (cụm không có phần tử nào)
- Một số chiến lược có thể được sử dụng để loại bỏ những cụm rỗng vô nghĩa này

- Lựa chọn các điểm có đóng góp nhiều nhất tới tổng bình phương lỗi SSE và đưa điểm đó vào cụm dữ liệu rỗng
- Lựa chọn một điểm trong cụm có SSE cao nhất và đưa vào cụm rỗng đó để giảm SSE nhiều nhất có thể đồng thời làm cụm rỗng có phần tử
- Nếu có nhiều cụm rỗng thì công việc trên được lặp lại nhiều lần

Quá trình phân cụm cần qua trình tiền xử lý dữ liệu và hậu xử lý dữ liệu

cũng giống như khi sử dụng một số các phương pháp chung trong khai phá dữ liệu.

Tiền xử lý dữ liệu cần thiết trong quá trình phân cụm vì dữ liệu cần được chuẩn hóa hoặc loại bỏ các phần tử ngoại lai trước khi đưa vào thuật toán

Hậu xử lý dữ liệu cần thiết trong những trường hợp sau:

- Cần loại bỏ những cụm nhỏ (số lượng phần tử trong cụm ít) vì có thể chúng chỉ chứa đựng các phần tử ngoại lai, không có ý nghĩa trong ứng dụng thực tế
- Phân chia những cụm lỏng lẻo (hay mật độ các phần tử trong cụm không đồng đều, chỗ dày đặc, chỗ thưa thớt), hay nói cách khác là các cụm có tổng bình phương lỗi lớn thành các cụm nhỏ
- Trộn các cụm có khoảng cách khá gần nhau, hay có tổng bình phương lỗi SSE khá thấp.
- Có thể sử dụng những bước này trong quá trình phân cụm

Phương pháp K-mean phân đôi

Là một biến đổi của K-mean mà có thể sinh ra một sự phân cụm có phân cấp hoặc phân cụm dạng phân mảnh

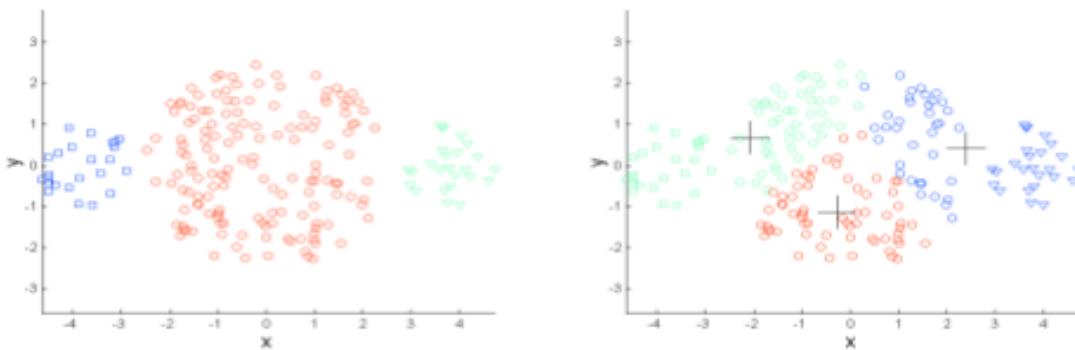
Thuật toán được thể hiện như các bước dưới đây

1. Khởi tạo danh sách L các cụm để chứa các cụm tìm được, ban đầu chỉ chứa có một cụm bao gồm tất cả các điểm
2. Lặp các bước sau
3. Chọn một cụm trong danh sách L các cụm trên
4. **For** i=1 to số lượng vòng lặp định trước **do**
5. Phân đôi cụm được lựa chọn thành hai phân cụm bằng phương pháp K-mean
6. **End for**
7. Thêm hai phân cụm kết quả của những lần phân đôi cụm trên với tổng bình phương lỗi SSE nhỏ nhất vào danh sách các cụm

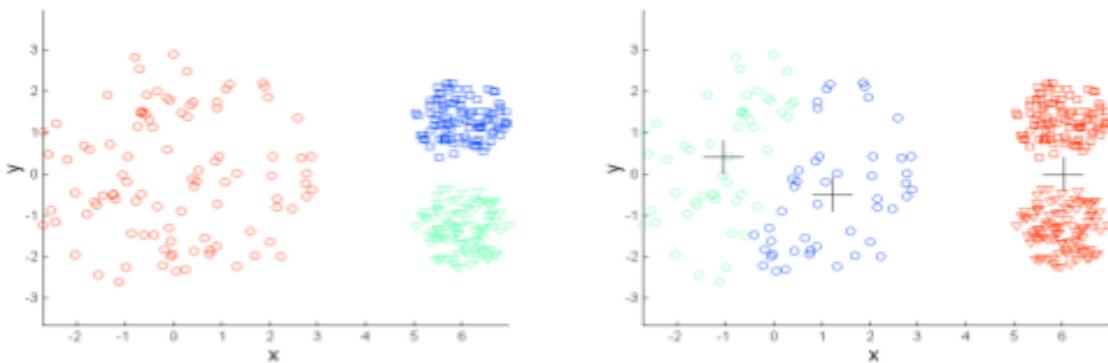
8. Cho đến khi danh sách các cụm chứa K cụm thì dừng.

Những hạn chế của K-means

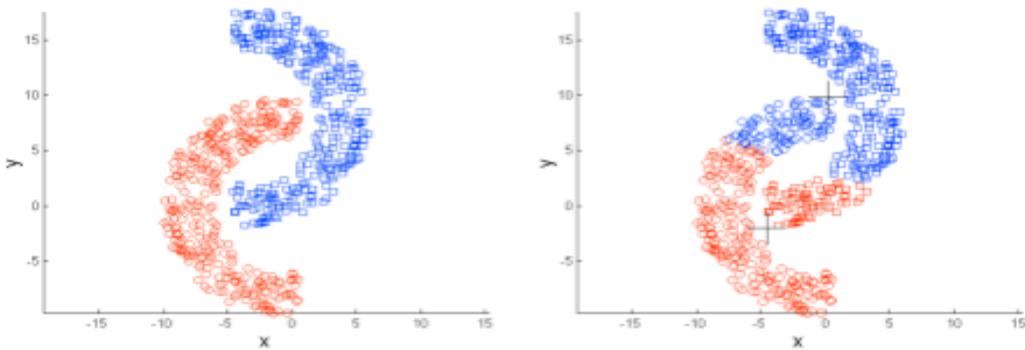
- K-mean có nhiều vấn đề khi các cụm khác nhau
- Về kích cỡ: nếu trong số các cụm có cụm có kích cỡ lớn hơn nhiều so với các cụm khác thì khi dùng K-mean để phân cụm sẽ cho kết quả sai nhiều bởi kích cỡ của các cụm kết quả của phương pháp thường là tương đương nhau. Ví dụ minh họa như hình vẽ dưới đây



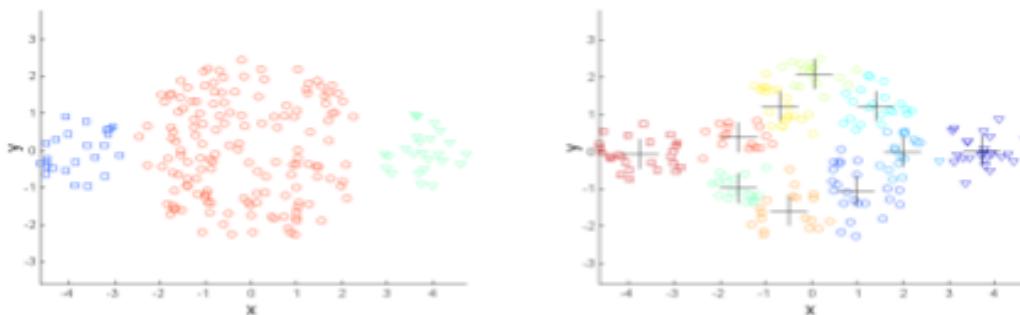
- Về mật độ dữ liệu: khi mật độ dữ liệu không đủ dày đặc trong cùng một cụm sẽ khiến cho chúng bị phân tách làm nhiều cụm khác nhau khi sử dụng phương pháp K-means, ngược lại khi mật độ tương đối dày đặc thì hai cụm gần nhau dễ bị ghép lại thành một cụm như trong hình vẽ minh họa sau đây

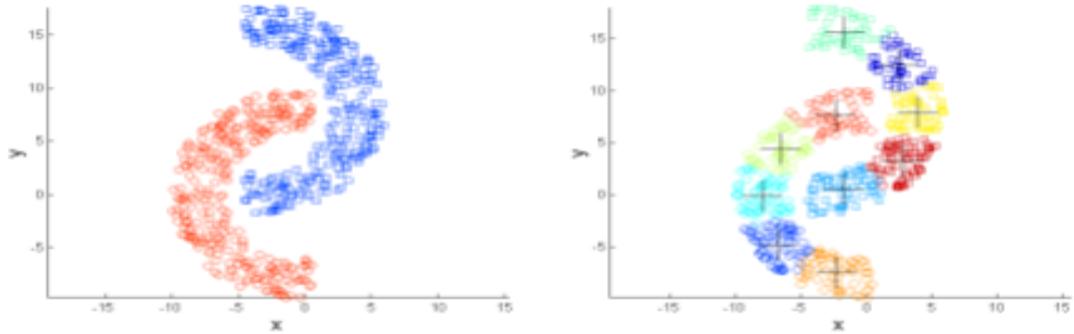


- Hình dạng không phải hình cầu: với trường hợp các điểm dữ liệu phân bố theo một hình dạng không phải hình cầu (không phải hình lồi) cũng gây ảnh hưởng lớn tới kết quả của phương pháp K-means. Ví dụ như được minh họa trong hình vẽ dưới đây, hình dạng của hai cụm dữ liệu (màu xanh và màu đỏ) ban đầu là dạng phi cầu nên khi dùng K-means để phân cụm sẽ có lỗi như trong hình vẽ



- K-mean cũng có vấn đề khi dữ liệu chứa phần tử ngoại lai
Để giải quyết những hạn chế của phương pháp K-means khi có sự khác nhau về kích cỡ dữ liệu và mật độ của các điểm dữ liệu trong các cụm, một giải pháp được đưa ra là sử dụng nhiều cụm, lúc này các cụm to trong kết quả sẽ được phân thành nhiều phân cụm khác nhau, sau đó cần kết hợp chúng lại với nhau để thành cụm to ban đầu. Hình vẽ sau mô tả điều đó





Câu hỏi ôn tập chương 4

4.1 Chất lượng dữ liệu có thể được đánh giá về độ chính xác, đầy đủ và nhất quán. Đề xuất hai chiều khác của chất lượng dữ liệu.

4.2 Giả sử rằng các giá trị cho một tập dữ liệu nhất định được nhóm thành các khoảng giá trị. Các khoảng giá trị và tần số tương ứng như sau.

tần số	tuổi
1–5	200
5–15	450
15–20	300
20–50	1500
50–80	700
80–110	44

Tính toán giá trị trung vị xấp xỉ cho dữ liệu.

4.3 Cung cấp thêm độ đo thống kê thường được sử dụng (ví dụ: không được minh họa trong chương này) để mô tả đặc tính phân tán dữ liệu và thảo luận cách chúng có thể được tính toán hiệu quả trong các cơ sở dữ liệu lớn.

4.4 Trong nhiều ứng dụng, các bộ dữ liệu mới được bổ sung thêm vào các tập dữ liệu lớn hiện có. Do đó, một cân nhắc quan trọng để tính toán tóm tắt dữ liệu mô tả là liệu một độ đo có thể

được tính hiệu quả theo cách gia tăng. Sử dụng hàm đếm số lượng, độ lệch chuẩn và trung vị làm ví dụ để cho thấy rằng đo lường phân phối hoặc đại số tạo điều kiện thuận lợi tính toán gia tăng hiệu quả, trong khi đo lường toàn diện thì không.

4.5 Trong dữ liệu trong thế giới thực, các bộ dữ liệu có giá trị thiếu ở một số thuộc tính là một sự xuất hiện phổ biến. Hãy mô tả các phương pháp khác nhau để xử lý vấn đề này.

4.6 Phạm vi giá trị của các phương pháp chuẩn hóa sau là gì?

- (a) chuẩn hóa min-max
- (b) chuẩn hóa z-score
- (c) chuẩn hóa theo tỷ lệ thập phân

4.7 Sử dụng hai phương pháp bên dưới để chuẩn hóa nhóm dữ liệu sau: 200, 300, 400, 600, 1000

- (a) chuẩn hóa min-max bằng cách thiết lập $\text{min} = 0$ và $\text{max} = 1$
- (b) chuẩn hóa z-score

4.8 Sử dụng sơ đồ để tóm tắt các thủ tục sau đây để lựa chọn tập con các thuộc tính:

- (a) lựa chọn chuyển tiếp từng bước (stepwise forward selection)
- (b) loại bỏ lùi từng bước (stepwise backward elimination)
- (c) kết hợp lựa chọn chuyển tiếp và loại bỏ lùi

4.9 Giả sử một nhóm 12 bộ giá bán đã được sắp xếp như sau: 5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215. Phân vùng chúng thành ba thùng theo từng phương pháp sau:

- (a) phân vùng tần số bằng nhau (equidepth)
- (b) phân vùng theo chiều rộng bằng nhau
- (c) phân cụm

4.10 Tính trung vị là một trong những biện pháp tổng thể quan trọng nhất trong phân tích dữ liệu. Đề xuất một số phương pháp cho xác định trung vị. Phân tích độ phức tạp tương ứng trong các cài đặt thông số khác nhau và quyết định mức độ nào giá trị thực có thể xác định. Hơn nữa, đề xuất

một chiến lược heuristic để cân bằng giữa tính chính xác và phức tạp và sau đó áp dụng nó cho tất cả các phương pháp bạn đã đưa ra.

4.11 Điều quan trọng là phải xác định hoặc chọn các độ đo sự tương tự trong quá trình phân tích dữ liệu. Tuy nhiên, không có một độ đo tương tự chủ quan được chấp nhận phổ biến. Sử dụng các độ đo tương tự khác nhau có thể suy ra các kết quả khác nhau. Tuy nhiên, một số các độ đo tương tự khác nhau có thể tương đương sau một số phép biến đổi.

Giả sử chúng ta có tập dữ liệu hai chiều sau đây:

	A1	A2
X ₁	1.5	1.7
X ₂	2	1.9
X ₃	1.6	1.8
X ₄	1.2	1.5
X ₅	1.5	1.0

- (a) Xem xét dữ liệu dưới dạng các điểm dữ liệu hai chiều. Cho điểm dữ liệu mới, $x = (1.4, 1.6)$ như một truy vấn, xếp hạng các điểm cơ sở dữ liệu dựa trên sự giống nhau với truy vấn sử dụng (1) Khoảng cách Euclidean và (2) độ tương tự cosin.
- (b) Chuẩn hóa tập dữ liệu để làm cho norm của mỗi điểm dữ liệu bằng 1. Sử dụng khoảng cách Euclidean trên dữ liệu được chuyển đổi để xếp hạng các điểm dữ liệu

4.12 Đề xuất một thuật toán, viết bằng mã giả hoặc bằng ngôn ngữ lập trình yêu thích của bạn, cho những điều sau đây:

- (a) Tự động tạo hệ thống phân cấp khái niệm cho dữ liệu phân loại dựa trên số giá trị khác biệt của các thuộc tính trong lược đồ đã cho
- (b) Tự động tạo hệ thống phân cấp khái niệm cho dữ liệu số dựa trên quy tắc phân vùng có chiều rộng bằng nhau
- (c) Tự động tạo hệ thống phân cấp khái niệm cho dữ liệu số dựa trên quy tắc phân vùng tần số bằng nhau

4.13 Tải dữ liệu nhanh đặt ra một thách thức trong các hệ thống cơ sở dữ liệu vì dữ liệu đầu vào là thường bẩn. Trong nhiều trường hợp, bản ghi đầu vào có thể có nhiều giá trị bị thiếu và một số bản ghi có thể bị ô nhiễm (tức là, với một số giá trị dữ liệu nằm ngoài phạm vi hoặc của một kiểu khác loại dữ liệu thực tế). Làm việc với thuật toán tải và làm sạch dữ liệu tự động để dữ liệu sai sẽ được đánh dấu và không được chèn may vào cơ sở dữ liệu trong quá trình tải dữ liệu.

4.14 (Bài tập cài đặt) Thực hiện ba thuật toán khai phá tập các mặt hàng thường xuyên được giới thiệu trong chương này: (1) Apriori, (2) Tăng trưởng FP và (3) ECLAT (khai phá sử dụng định dạng dữ liệu theo chiều dọc), sử dụng ngôn ngữ lập trình mà bạn quen thuộc với, chẳng hạn như C ++ hoặc Java. So sánh hiệu suất của từng thuật toán với các loại tập dữ liệu lớn khác nhau. Viết báo cáo để phân tích các tình huống (chẳng hạn như kích thước dữ liệu, phân phối dữ liệu, cài đặt ngưỡng hỗ trợ tối thiểu và mật độ mẫu) trong đó một thuật toán có thể hoạt động tốt hơn các thuật toán khác và nêu rõ lý do.

4.15 Giả sử rằng một cửa hàng lớn có một cơ sở dữ liệu giao dịch được lưu trữ phân tán trong số bốn vị trí. Các giao dịch trong mỗi cơ sở dữ liệu thành phần có cùng định dạng, cụ thể là $T_j: \{i_1, \dots, i_m\}$, trong đó T_j là mã định danh giao dịch và i_k ($1 \leq k \leq m$) là mã định danh của một mặt hàng được mua trong giao dịch. Đề xuất một thuật toán hiệu quả để khai phá các luật kết hợp toàn cục (không xem xét các kết hợp đa cấp). Bạn có thể trình bày thuật toán dưới dạng một phác thảo. Thuật toán của bạn không yêu cầu gửi tất cả dữ liệu vào một vị trí và không được gây ra truyền thông trên mạng quá mức.

4.16 Giả sử rằng tập các mặt hàng thường xuyên được lưu cho một cơ sở dữ liệu giao dịch lớn, DB . Thảo luận cách khai phá hiệu quả các luật kết hợp (toàn cầu) với cùng ngưỡng hỗ trợ tối thiểu nếu một tập hợp các giao dịch mới, được biểu thị là ΔDB , là (từng bước) được thêm vào?

4.17 Các thuật toán khai phá mẫu xuất hiện thường xuyên nhất chỉ xem xét các mặt hàng khác biệt trong một giao dịch. Tuy nhiên, nhiều lần xuất hiện của một mặt hàng trong cùng một giỏ mua hàng, chẳng hạn như bốn chiếc bánh và ba bình sữa, có thể rất quan trọng trong phân tích dữ liệu giao dịch. Làm thế nào tôi có thể khai phá tập các mặt hàng thường xuyên một cách hiệu quả khi xem xét nhiều lần xuất hiện của các mặt hàng? Đề xuất sửa đổi các thuật toán nổi tiếng, chẳng hạn như Apriori và FP-tăng trưởng, để thích nghi với tình huống như vậy.

4.18 (Bài tập cài đặt) Cài đặt ba phương pháp khai phá các mặt hàng thường xuyên đóng (1) A-Close [PBTL99] (dựa trên phần mở rộng của Apriori [AS94b]), (2) CLOSET + [WHP03] (dựa trên phần mở rộng của tăng trưởng FP [HPY00]) và (3) CHARM [ZH02] (dựa trên mở rộng ECLAT [Zak00]). So sánh hiệu suất của chúng với nhiều loại bộ dữ liệu lớn. Viết báo cáo để trả lời các câu hỏi sau:

- (a) Tại sao việc khai phá bộ các tập thường xuyên đóng thường hấp dẫn hơn khai phá tập hợp các mặt hàng thường xuyên hoàn chỉnh (dựa trên các thử nghiệm của bạn trên cùng một tập dữ liệu ở Bài tập 4.14)?
- (b) Phân tích các tình huống (chẳng hạn như kích thước dữ liệu, phân phối dữ liệu, thiết lập ngưỡng hỗ trợ tối thiểu và mật độ mẫu) và tại sao một thuật toán hoạt động tốt hơn những thuật toán khác.

4.19 Đề xuất và phác thảo một cách tiếp cận khai phá mức chia sẻ để khai phá luật kết hợp đa cấp trong đó mỗi mục được mã hóa theo vị trí cấp của nó và một lần quét ban đầu của cơ sở dữ liệu thu thập số lượng cho mỗi mục ở mỗi cấp độ khái niệm, xác định các mục thường xuyên và ít thường xuyên. Nhận xét về chi phí xử lý khai phá các luật kết hợp đa cấp với phương pháp này so với khai phá luật kết hợp đơn cấp.

4.20 (Bài tập cài đặt) Nhiều kỹ thuật đã được đề xuất để cải thiện hơn nữa hiệu suất của các thuật toán khai phá tập các mặt hàng thường xuyên. Lấy các thuật toán tăng trưởng mẫu thường xuyên dựa trên FP-tree, chẳng hạn như tăng trưởng FP, làm ví dụ, triển khai một thuật toán các kỹ thuật tối ưu hóa sau đây và so sánh hiệu suất của kỹ thuật mới của bạn thực hiện với một kỹ thuật không kết hợp tối ưu hóa như vậy.

- (a) Khai thác mẫu thường xuyên được đề xuất trước đây với FP-tree tạo điều kiện cơ sở mẫu bằng cách sử dụng kỹ thuật chiểu từ dưới lên (tức là, dự án trên đường dẫn tiền tố của một mặt hàng p). Tuy nhiên, người ta có thể phát triển kỹ thuật chiểu từ trên xuống (tức là, dự án trên đường dẫn hậu tố của một mặt hàng p trong quá trình tạo mẫu cơ sở có điều kiện). Thiết kế và cài đặt phương pháp khai phá cây FP từ trên xuống dưới và so sánh hiệu suất của bạn với phương pháp chiểu từ dưới lên.

(b) Các nút và con trỏ được sử dụng thống nhất trong cây FP trong thiết kế của thuật toán FP tăng trưởng. Tuy nhiên, cấu trúc như vậy có thể tiêu tốn rất nhiều không gian khi dữ liệu thưa thớt. Một thiết kế thay thế có thể là được khám phá cài đặt lai dựa trên mảng và con trỏ, trong đó một nút có thể lưu trữ nhiều mặt hàng khi nó không chứa điểm chia tách cho nhiều nhánh con. Phát triển việc cài đặt như vậy và so sánh nó với bản gốc.

(c) Việc tốn nhiều thời gian và không gian để tạo ra nhiều cơ sở mẫu có điều kiện trong quá trình khai phá mẫu tăng trưởng. Một thay thế thú vị là đẩy phải (push right) các nhánh đã được khai phá cho một mặt hàng cụ thể p , tức là đẩy chúng vào các nhánh còn lại của cây FP. Điều này được thực hiện để ít cơ sở mẫu điều kiện hơn phải được tạo và việc chia sẻ thêm có thể được khai thác khi khai phá các chi nhánh còn lại của cây FP. Thiết kế và cài đặt một phương pháp kiểu như vậy và tiến hành một nghiên cứu hiệu suất trên đó.

4.21 Đưa ra một ví dụ ngắn để chỉ ra rằng các mục trong một luật kết hợp mạnh có thể là một tương quan âm (negatively correlated) trong thực tế

4.22 Bảng sau đây tóm tắt dữ liệu giao dịch của siêu thị, nơi mặt hàng *hotdogs* đè cập đến các giao dịch có chứa xúc xích, *nohotdogs* đè cập đến giao dịch không chứa xúc xích, *hamburgers* đè cập đến các giao dịch có chứa bánh mì kẹp thịt và *nohamburgers* đè cập đến các giao dịch không chứa bánh mì kẹp thịt

	<i>hotdogs</i>	<i>nohotdogs</i>	Σ_{row}
<i>hamburgers</i>	2000	500	2500
<i>nohamburgers</i>	1000	1500	2500
Σ_{col}	3000	2000	5000

(a) Giả sử rằng luật kết hợp “*hotdogs* \Rightarrow *hamburgers*” được khai phá ra. Với ngưỡng hỗ trợ tối thiểu là 25% và ngưỡng tin cậy tối thiểu là 50%, đó có phải là luật kết hợp mạnh không?

(b) Dựa trên dữ liệu đã cho, việc mua xúc xích có độc lập với việc mua bánh mì kẹp thịt không?

4.23 Trong phân tích dữ liệu đa chiều, thật thú vị khi trích xuất các cặp ô tương tự với các thay đổi đáng kể về phép đo trong một khối dữ liệu, các trường hợp được coi là tương tự nếu chúng có liên quan theo dạng cuộn lên (tức là, tổ tiên), khoan xuống (tức là con cháu), hoặc đột biến một chiều (tức là, anh chị em ruột). Phân tích như vậy được gọi là phân tích gradient khối. Giả sử

độ đo của khối dữ liệu là phép toán trung bình (average). Một người sử dụng một bộ các tê bào thăm dò và muốn tìm các bộ tương ứng của các gradient ô, mỗi một trong số đó thỏa mãn một ngưỡng gradient nhất định. Ví dụ, tìm tập hợp các ô gradient tương ứng có giá bán trung bình lớn hơn 20% so với các ô thăm dò đã cho. Phát triển một thuật toán rồi khai phá hiệu quả tập hợp các ô gradient bị hạn chế trong một khối dữ liệu lớn.

4.24 Khai phá luật kết hợp thường tạo ra một số lượng lớn các luật. Thảo luận về các phương thức hiệu quả có thể được sử dụng để giảm số lượng các luật được tạo ra trong khi vẫn bảo toàn hầu hết các luật có giá trị.

4.25 Các mẫu tuần tự có thể được khai phá trong các phương thức tương tự như khai phá các luật kết hợp. Thiết kế một thuật toán hiệu quả để khai phá các mẫu tuần tự đa cấp từ cơ sở dữ liệu giao dịch. Một ví dụ về mẫu như sau: “Khách hàng mua PC sẽ mua phần mềm của Microsoft trong vòng ba tháng,” người dùng có thể tìm hiểu về phiên bản cải tiến của mẫu, chẳng hạn như “Khách hàng mua máy tính Pentium sẽ mua Microsoft Office trong vòng ba tháng.”

4.26 Giá của mỗi sản phẩm trong một cửa hàng không âm. Người quản lý cửa hàng chỉ quan tâm đến luật có dạng: “một mặt hàng miễn phí có thể được nhận với một giao dịch có tổng số tiền mua hàng là 200 đô la”. Hãy phát biểu cách khai phá các luật đó một cách hiệu quả.

4.27 Giá của từng mặt hàng trong cửa hàng là không âm. Đối với mỗi trường hợp sau, xác định các loại ràng buộc mà chúng đại diện và thảo luận ngắn gọn về cách khai phá các luật kết hợp như vậy một cách hiệu quả

- (a) Chứa ít nhất một trò chơi Nintendo
- (b) Chứa các mặt hàng có tổng giá thấp hơn \$150
- (c) Chứa một mặt hàng miễn phí và các mặt hàng khác với tổng giá của nó ít nhất là \$200
- (d) Trường hợp giá trung bình của tất cả các mặt hàng là từ \$100 đến \$500

4.28 Tóm tắt các bước chính của việc phân loại bằng cây quyết định.

4.29 Tại sao kỹ thuật tia cành cây hữu ích trong việc suy diễn trên cây quyết định? Nhược điểm của việc sử dụng một tập hợp các bộ dữ liệu riêng biệt để đánh giá việc cắt tia?

4.30 Với cây quyết định, bạn có tùy chọn (a) chuyển đổi cây quyết định thành các luật sau đó cắt tia các luật kết quả, hoặc (b) cắt tia cây quyết định và sau đó chuyển đổi cây tia cành thành các luật. Lợi thế của (a) hơn (b) như thế nào?

4.31 Việc xác định độ phức tạp tính toán trong trường hợp tồi tệ nhất của thuật toán cây quyết định là quan trọng. Cho trước một tập dữ liệu D, số thuộc tính n và số lượng các bộ dữ liệu dùng để huấn luyện $|D|$, hãy chỉ ra rằng chi phí tính toán của việc dựng cây tối đa là $n \times |D| \times \log(|D|)$.

4.32 Cho một tập dữ liệu 5 GB với 50 thuộc tính (mỗi thuộc tính chứa 100 giá trị riêng biệt) và 512 MB bộ nhớ chính trong máy tính xách tay của bạn, phác thảo một phương pháp hiệu quả mà xây dựng cấu trúc cây quyết định trong các tập dữ liệu lớn như vậy. Giải thích câu trả lời của bạn bằng cách tính toán sơ bộ sử dụng bộ nhớ chính.

4.33 Rất khó để đánh giá độ chính xác phân loại khi các đối tượng dữ liệu riêng lẻ có thể thuộc về cho nhiều hơn một lớp tại một thời điểm. Trong những trường hợp như vậy, hãy bình luận về những tiêu chí bạn sẽ sử dụng để so sánh các bộ phân loại khác nhau được mô hình hóa trên cùng một bộ dữ liệu.

4.34 Tóm tắt phác thảo cách tính toán sự khác biệt giữa các đối tượng được mô tả bởi các loại biến sau:

- (a) Các biến số (dạng phân khoáng)
- (b) Biến nhị phân bất đối xứng
- (c) Các biến phân loại
- (d) Biến số tỷ lệ
- (e) Đối tượng vector phi độ đo

4.35 Cho các phép đo sau cho biến *tuổi*: 18, 22, 25, 42, 28, 43, 33, 35, 56, 28, chuẩn hóa biến bằng cách sau:

- (a) Tính toán độ lệch tuyệt đối trung bình của *tuổi*.
- (b) Tính toán z-score trong bốn phép đo đầu tiên.

4.36 Cho hai đối tượng đại diện bởi các bộ dữ liệu (22, 1, 42, 10) và (20, 0, 36, 8):

- (a) Tính toán khoảng cách Euclidean giữa hai đối tượng trên.
- (b) Tính khoảng cách Manhattan giữa hai đối tượng trên.
- (c) Tính toán khoảng cách Minkowski giữa hai đối tượng trên, sử dụng $q = 3$.

4.37 Cả hai thuật toán k-means và k-medoids đều có thể thực hiện phân cụm hiệu quả. Minh họa sức mạnh và điểm yếu của k-means so với thuật toán k-medoids.

4.38 Ví dụ về cách các phương thức phân cụm cụ thể có thể được tích hợp, ví dụ, nơi một thuật toán phân cụm được sử dụng làm bước tiền xử lý cho một thuật toán khác. Ngoài ra, cung cấp lý do tại sao việc tích hợp hai phương pháp đôi khi có thể dẫn tới cải thiện chất lượng và hiệu quả phân cụm.

4.39 Phân cụm đã được công nhận rộng rãi như một nhiệm vụ khai phá dữ liệu quan trọng với phạm vi rộng các ứng dụng. Đưa ra một ví dụ ứng dụng cho mỗi trường hợp sau:

- (a) Một ứng dụng lấy phân cụm như một chức năng khai phá dữ liệu chính
- (b) Một ứng dụng lấy phân cụm như một công cụ tiền xử lý để chuẩn bị dữ liệu cho các nhiệm vụ khai phá dữ liệu khác.

4.40 Mô tả từng thuật toán phân cụm sau đây với các tiêu chí sau: (i) hình dạng của các cụm có thể được xác định; (ii) các thông số đầu vào phải được xác định; và (iii) hạn chế.

- (a) k-means
- (b) k-medoids
- (c) CLARA
- (d) DBSCAN

Tài liệu tham khảo

1. Jiawei Han and Micheline Kamber, “Data Mining: Concepts and Techniques”, Morgan Kanfmann Publishers, Second Edition.
2. Joseph Fong, “Information Systems Reengineering and Integration”, Springer Verlag, 2006, ISBN 978-1-84628-382-6, Second edition
3. http://www-sal.cs.uiuc.edu/~hanj/DM_Book.html
4. <http://www-users.cs.umn.edu/~kumar/csci5980/index.html>
5. <http://www.cs.cityu.edu.hk/~jfong/course/cs5483/>
6. <http://www.ist.temple.edu/~vucetic/cis526fall2004.htm>