TCVN

TIÊU CHUẨN QUỐC GIA

TCVN 7981 - 2 : 2008
ISO/TS 17369 - 2 : 2005

Xuất bản lần 1

TRAO ĐỔI SIÊU DỮ LIỆU VÀ DỮ LIỆU THỐNG KÊ
PHẦN 2: MÔ HÌNH THÔNG TIN: THIẾT KẾ KHÁI NIỆM UML

Statistical data and metadata exchange

Section 2: Information model: UML definition design

HÀ NỘI - 2008

Mục lục Trang

Lời nói đầu

TCVN 7981 - 2 : 2008 hoàn toàn tương đương với **ISO/TS 17369 - 2 : 2005**

TCVN 7981 - 2 : 2008 do Ban kỹ thuật tiêu chuẩn quốc gia TCVN/TC 154 "*Quá trình, các yếu tố dữ liệu và tài liệu trong thương mại, công nghiệp và hành chính*" biên soạn, Tổng cục Tiêu chuẩn Đo lường Chất lượng đề nghị, Bộ Khoa học và Công nghệ công bố.

Bộ tiêu chuẩn TCVN 7981; *Trao đổi siêu dữ liệu và dữ liệu thống kê* gồm các phần sau:

- TCVN 7981-1 : 2008 (ISO/TS 17369 - 1 : 2005); Phần 1: Khung tổng quát về các tiêu chuẩn SDMX.
- TCVN 7981-2 : 2008 (ISO/TS 17369 - 2: 2005); Phần 2: Mô hình thông tin - Thiết kế khái niệm UML.

Bộ tiêu chuẩn ISO/TS 17369 :2005 còn các phần sau:

- ISO/TS 17369 : 2005 Section 3: SDMX-ML schema and documentation (*Tài liệu và lược đồ SDMX-ML*).
- ISO/TS 17369 : 2005 Section 4: SDMX-EDI syntax and documentation (*Tài liệu và cú pháp SDMX-EDI*).
- ISO/TS 17369 : 2005 Section 5: An implementer's guide for SDMX (*Hướng dẫn người thực thi SDMX*).
- ISO/TS 17369 : 2005 Section 6: SDMX guideline for use of web services (*Hướng dẫn sử dụng các dịch vụ web trong SDMX*).

T I Ê U C H U Ẩ N Q U Ố C G I A **TCVN 7981-2 : 2008**

Xuất bản lần 1

Trao đổi siêu dữ liệu và dữ liệu thống kê
Phần 2: Mô hình thông tin – Thiết kế khái niệm UML

Statistical data and metadata exchange (SDMX)
Section 2: Information Model – UML concept design

1 Tổng quan

Tiêu chuẩn tham khảo này đưa ra quan điểm chi tiết về mô hình thông tin, làm cơ sở cho các đặc tả SDMX. Các ký hiệu UML hoặc khái niệm về tập khóa có thể tham khảo trong phụ lục của tiêu chuẩn này như bài tập mở đầu.

1.1 Tài liệu liên quan

Các tài liệu liên quan đến mô hình thông tin SDMX:

MÔ HÌNH THÔNG TIN SDMX: THIẾT KẾ KHÁI NIỆM UML (tiêu chuẩn này)

Tiêu chuẩn này bao gồm định nghĩa đầy đủ về mô hình thông tin, ngoại trừ các giao diện sổ đăng ký. Tiêu chuẩn này giúp các nhà chuyên môn hiểu đầy đủ phạm vi của các tiêu chuẩn kỹ thuật SDMX theo một dạng trung tính về cú pháp.

ĐẶC TẢ SỔ ĐĂNG KÝ SDMX: CÁC GIAO DIỆN LOGIC

Tài liệu này cung cấp đặc tả logic cho các giao diện sổ đăng ký, bao gồm đặt hàng/thông báo, đăng ký/đệ trình dữ liệu, siêu dữ liệu và truy vấn.

HƯỚNG DẪN CHO NGƯỜI THỰC HIỆN SDMX

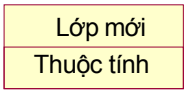
Tài liệu này giải thích các cấu trúc trong mô hình ở dạng sơ đồ bậc cao và ánh xạ các sơ đồ này sang các sơ đồ lớp trong mô hình. Ngoài ra, nó còn đưa ra các ví dụ về cấu trúc đã hoạt động. Tài liệu này giúp các nhà chuyên môn hiểu tổng thể các cấu trúc mô hình theo dạng không chính thức và đầy đủ bằng mô hình được đưa ra trong tiêu chuẩn về thiết kế khái niệm UML.

1.2 Kỹ thuật mô hình hóa và chú thích bằng sơ đồ

Kỹ thuật mô hình hóa được sử dụng cho mô hình thông tin (SDMX-IM) là ngôn ngữ mô hình hóa thống nhất (Unified Modelling Language - UML). Khái quát về các kết cấu UML sử dụng trong SDMX-IM được đưa ra trong phụ lục “Hướng dẫn UML trong mô hình thông tin SDMX”

Việc lập sơ đồ UML cho phép thể hiện một lớp có hoặc không có các ngăn ô đối với cả thuộc tính và thao tác hoặc chỉ đối với thuộc tính hoặc thao tác (đôi khi được gọi là các phương pháp). Trong tiêu chuẩn này

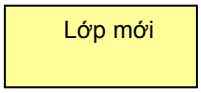
ngăn ô các thao tác không được thể hiện vì không có thao tác nào hoặc.



Hình 1 – Lớp cùng với các thao tác bị bỏ qua

Trong một vài sơ đồ đối với lớp, ngăn ô thuộc tính bị bỏ thông qua mặc dù có một số thuộc tính. Điều này được xem xét và thực hiện giúp sơ đồ rõ ràng hơn. Các quy tắc được sử dụng:

- Các thuộc tính luôn có mặt trên sơ đồ lớp, ở đây; lớp, các thuộc tính và liên kiết của lớp đó được xác định.
- Trong các sơ đồ khác, như sơ đồ kế thừa, các thuộc tính có thể bị lược bỏ khỏi lớp đó để sơ đồ rõ ràng hơn.



Hình 2 – Lớp cùng với các thuộc tính bị bỏ qua

Lưu ý, trong mọi trường hợp, các thuộc tính được kế thừa từ lớp trên không được thể hiện ở lớp con.

Cấu trúc bảng sau đây được sử dụng trong định nghĩa của các lớp, thuộc tính và liên kết.

Lớp	Đặc trưng	Mô tả
`ClassName (tên lớp)`		
	`attributeName` (tên thuộc tính)	
	`associationName` (tên liên kết)	
	`+roleName` (tên vai trò)	

Nội dung trong cột "Đặc trưng" bao gồm hoặc giải thích một trong các đặc trưng sau đây của lớp đó:

- là lớp trừu tượng hoặc không. Các lớp trừu tượng được đưa ra dưới dạng phông chữ *`italicCourier`*;
- Lớp trên của lớp kế thừa này, nếu có
- Các lớp con của lớp này, nếu có
- Thuộc tính – tên thuộc tính (`attributeName`) được đưa ra dưới dạng phông chữ `Courier` ;
- Liên kết – tên liên kết (`associationName`) được đưa ra dưới dạng phông chữ `Courier`. Nếu liên kết được tạo từ liên kết giữa các lớp trên thì định dạng: /tên liên kết được đưa ra dưới dạng phông chữ `Courier` (`/associationName`);
- Vai trò – tên vai trò (`+roleName`) được đưa ra dưới dạng phông chữ `Courier` ;

Cột "Mô tả" đưa ra định nghĩa hoặc giải thích ngắn gọn lớp hoặc đặc trưng đó. Tên lớp UML có thể được sử dụng trong mô tả và thể hiện dưới dạng phông bình thường cùng với các khoảng trống giữa các từ. Ví dụ lớp `CodeList` (danh sách mã) được viết là Code List (danh sách mã).

1.3 Chức năng tổng thể

1.3.1 Các gói mô hình thông tin

Mô hình thông tin SDMX (SDMX-IM) là siêu mô hình khái niệm được phát triển từ các thực thi cụ thể về cú pháp. Mô hình được kết cấu như một tập các gói chức năng giúp cho việc thông hiểu, tái sử dụng và duy trì mô hình đó.

Ngoài ra, để giúp cho việc thông hiểu rõ ràng, mỗi gói được xem xét theo một trong ba lớp khái niệm sau:

- Lớp cơ sở SDMX bao gồm các khối cơ bản, được sử dụng bởi lớp định nghĩa cấu trúc, lớp báo cáo và phổ biến;
- Lớp định nghĩa cấu trúc bao gồm định nghĩa các sản phẩm có cấu trúc cần thiết để hỗ trợ việc báo cáo và phổ biến dữ liệu và siêu dữ liệu;
- Lớp báo cáo và phổ biến bao gồm định nghĩa các phần chứa dữ liệu và siêu dữ liệu được sử dụng cho việc báo cáo và phổ biến.

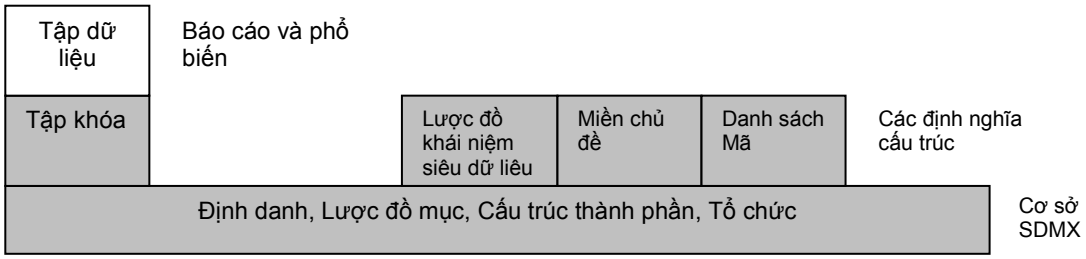
Trong thực tế, các lớp đó không có chức năng cấu trúc hàm ẩn hoặc rõ ràng như bất kỳ gói nào có thể sử dụng mọi kết cấu trong một gói khác.

1.3.2 Phiên bản 1.0

Trong phiên bản 1.0, siêu mô hình hỗ trợ các yêu cầu đối với:

- Định nghĩa tập khóa bao gồm Lược đồ phân loại (lĩnh vực), lược đồ khái niệm (siêu dữ liệu) và danh sách mã;
- Báo cáo và phổ biến dữ liệu và siêu dữ liệu liên quan.

SDMX-IM bao gồm một số các gói. Các gói này đóng vai trò như các ngăn ô thích hợp cho các mô hình con khác nhau trong SDMX-IM. Sơ đồ dưới đây chỉ ra các mô hình con của SDMX-IM trong phiên bản 1.0



Hình 3 – Cấu trúc gói của mô hình thông tin SDMX trong phiên bản 1.0

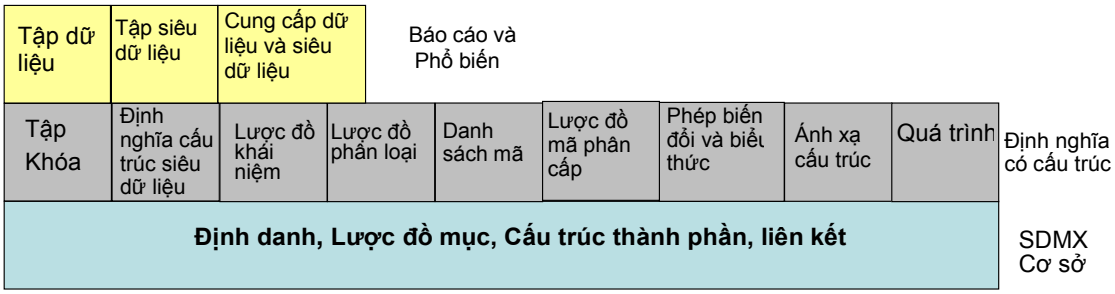
1.3.3 Phiên bản 2.0

Phiên bản 2.0 mở rộng chức năng so với phiên bản 1.0 trong chủ yếu trong phạm vi của siêu dữ liệu, theo các cách khác nhau để xác định cấu trúc, hỗ trợ phân tích dữ liệu bằng hệ thống cùng với kiến thức

về cấu trúc kiểu khối hộp như các hệ thống OLAP[1]. Các gói sau đâu được bổ sung trong phiên bản 2.0:

- Định nghĩa cấu trúc siêu dữ liệu
- Tập siêu dữ liệu
- Lược đồ mã phân cấp
- Định nghĩa khối hộp
- Cung cấp dữ liệu và siêu dữ liệu
- Phép biến đổi và thể hiện

Ngoài ra, định nghĩa cấu trúc dữ liệu đồng nghĩa được gán cho tập khóa, khi hai khái niệm này được sử dụng trong các cộng đồng khác nhau nhưng mang nghĩa giống nhau. Trong tiêu chuẩn này sử dụng thuật ngữ tập khóa.

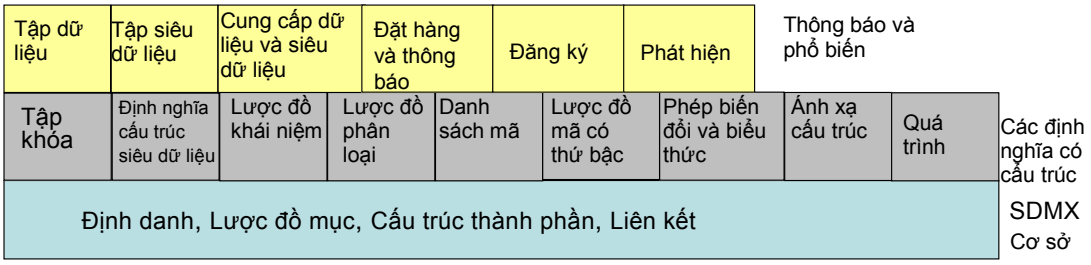


Hình 4 – Cấu trúc gói của mô hình thông tin trong phiên bản 2.0

Các gói bổ sung đặc trưng cho sổ đăng ký dựa trên kịch bản được tìm thấy trong các đặc tả giao diện sổ đăng ký. Chúng được biểu diễn trong sơ đồ ở trên và bao gồm:

- Đặt hàng và thông báo
- Đăng ký
- Phát hiện

Chú ý rằng dữ liệu và siêu dữ liệu được yêu cầu đối với các chức năng sổ đăng ký không bị giới hạn trong ba gói này và sổ đăng ký cũng sử dụng các gói khác trong mô hình thông tin.



Hình 5 – Cấu trúc gói của mô hình thông tin SDMX trong phiên bản 2.0 bao gồm sổ đăng ký

[1] OLAP: Việc xử lý phân tích theo dòng

2 Tác nhân và trường hợp sử dụng

2.1 Tác nhân và trường hợp sử dụng

Để xây dựng các mô hình dữ liệu, cần thiết phải hiểu các chức năng được hỗ trợ từ việc xác định các yêu cầu. Các yêu cầu này được xác định trong mô hình trường hợp sử dụng. Mô hình trường hợp sử dụng bao gồm các tác nhân và trường hợp sử dụng, được định nghĩa như sau:

Tác nhân

"*Tác nhân xác định một tập các vai trò rõ ràng của người sử dụng hệ thống khi tương tác với nó. Một tác nhân có thể đóng vai trò là một cá nhân hoặc một hệ thống bên ngoài.*"

Trường hợp sử dụng

"Trường hợp sử dụng xác định một tập các thể hiện trường hợp sử dụng, trong đó mỗi thể hiện là một trình tự các hành động tiến hành để tạo ra một kết quả giá trị có thể quan sát cho một tác nhân nào đó"

Mục đích tổng thể của mô hình là hỗ trợ việc báo cáo, phổ biến, trao đổi dữ liệu và siêu dữ liệu, trong lĩnh vực dữ liệu thống kê được tập hợp và các siêu dữ liệu liên quan. Để đạt được điều này, mô hình cần hỗ trợ ba khía cạnh cơ bản của quá trình sau:

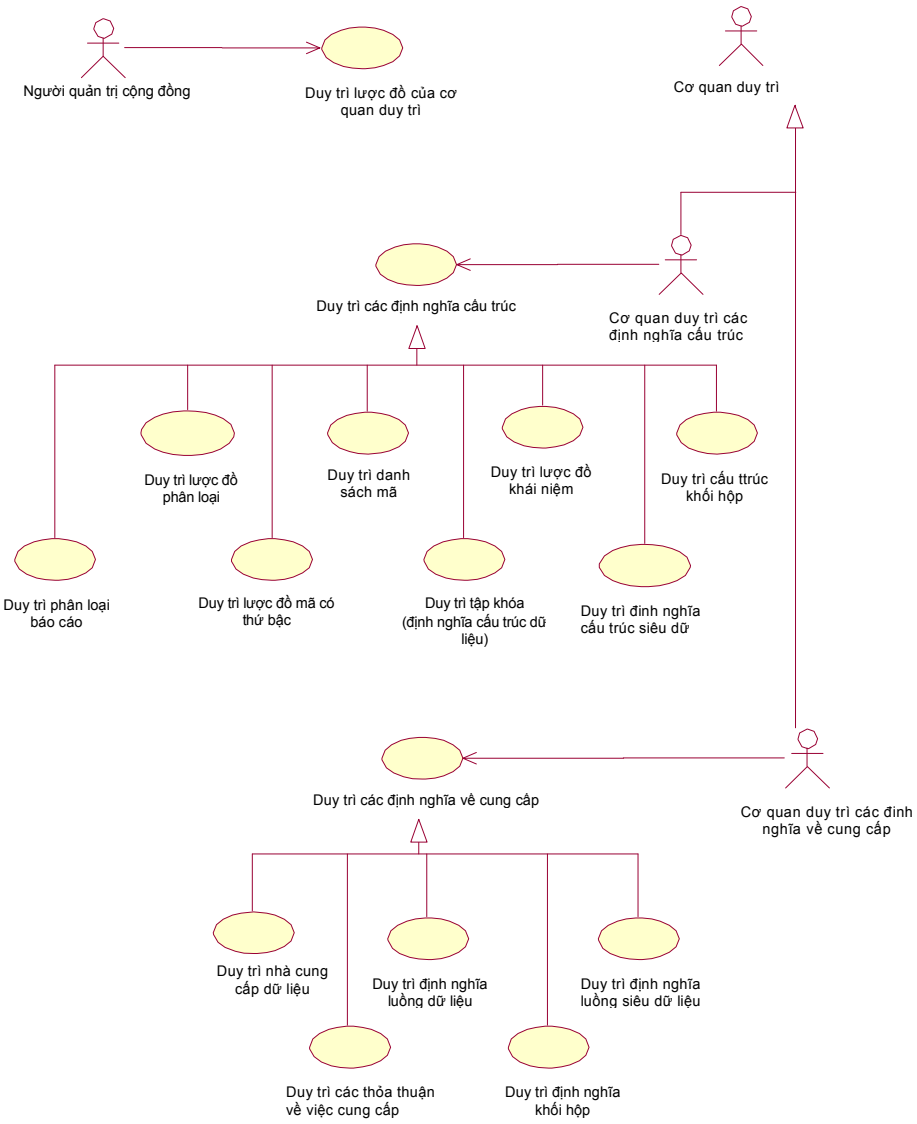
- Duy trì các định nghĩa cung cấp và cấu trúc;
- Công bố (báo cáo) và sử dụng dữ liệu và siêu dữ liệu;
- Truy cập dữ liệu, siêu dữ liệu, các định nghĩa cung cấp và cấu trúc.

Tiêu chuẩn này bao hàm hai khía cạnh đầu tiên, khía cạnh thứ ba được đưa ra trong tài liệu về mô hình logic của sổ đăng ký.

2.2 Sơ đồ trường hợp sử dụng

2.2.1 Duy trì các định nghĩa cung cấp và cấu trúc

2.2.1.1 Trường hợp sử dụng



Hình 6 –Các trường hợp sử dụng đối với việc duy trì các định nghĩa cung cấp và cấu trúc dữ liệu và siêu dữ liệu

2.2.1.2 Giải thích sơ đồ

Để các ứng dụng công bố, sử dụng dữ liệu và siêu dữ liệu, thì cấu trúc và nội dung cho phép của dữ liệu và siêu dữ liệu phải được xác định và sẵn có cho các ứng dụng, cũng như các định nghĩa hỗ trợ quá trình công bố và sử dụng thực tế. Đây là trách nhiệm của cơ quan duy trì (Maintenance Agency - MA).

Tất cả sản phẩm phải được duy trì bởi cơ quan duy trì (MA). Để thuận tiện, tác nhân cơ quan duy trì (MA) được chia thành hai tác nhân nhỏ:

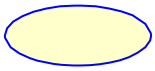
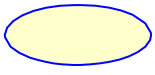
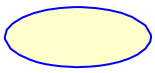
- Duy trì định nghĩa cấu trúc;
- Duy trì định nghĩa cung cấp.

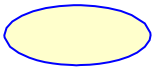
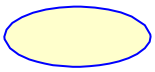
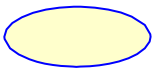
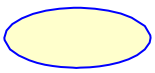
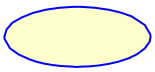
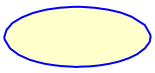
Cả hai chức năng này có thể được thực hiện bởi cùng một người hoặc ít nhất bởi cùng một cơ quan duy trì (MA), mục đích của các định nghĩa đó là khác nhau vì thế các vai trò cũng được phân biệt: các định nghĩa cấu trúc xác định định dạng và nội dung cho phép của dữ liệu và siêu dữ liệu khi được báo cáo hoặc phổ biến. Các định nghĩa cung cấp hỗ trợ quá trình báo cáo và phổ biến (ai báo cáo? Báo cáo cho a?i và báo cáo khi nào?).

Trong một cộng đồng sử dụng hoạt động dựa trên cơ sở kịch bản, trong đó có ít nhất các định nghĩa cấu trúc có thể được chia sẻ, thì điều quan trọng là lược đồ cơ quan duy trì phải được các tổ chức có trách nhiệm duy trì (gọi là người quản trị cộng đồng - CA).

2.2.1.3 Định nghĩa

Tác nhân	Trường hợp sử dụng	Mô tả
 Người quản trị cộng đồng		Tổ chức có trách nhiệm quản trị các định nghĩa cấu trúc chung cho toàn thể cộng đồng.
	 Duy trì lược đồ của cơ quan duy trì	Tạo và duy trì lược đồ của các cơ quan duy trì.
 Cơ quan duy trì		Cơ quan có trách nhiệm duy trì các sản phẩm có cấu trúc như: các danh sách mã, lược đồ khái niệm, định nghĩa cấu trúc tập khóa, định nghĩa cấu trúc siêu dữ liệu, các sản phẩm về cung cấp dữ liệu và siêu dữ liệu như người cung cấp dữ liệu và định nghĩa luồng dữ liệu. Các vai trò phụ là: Cơ quan duy trì định nghĩa cấu trúc ; Cơ quan duy trì định nghĩa việc cung cấp.
 Cơ quan duy trì định nghĩa cấu trúc		Có trách nhiệm duy trì các định nghĩa cấu trúc.

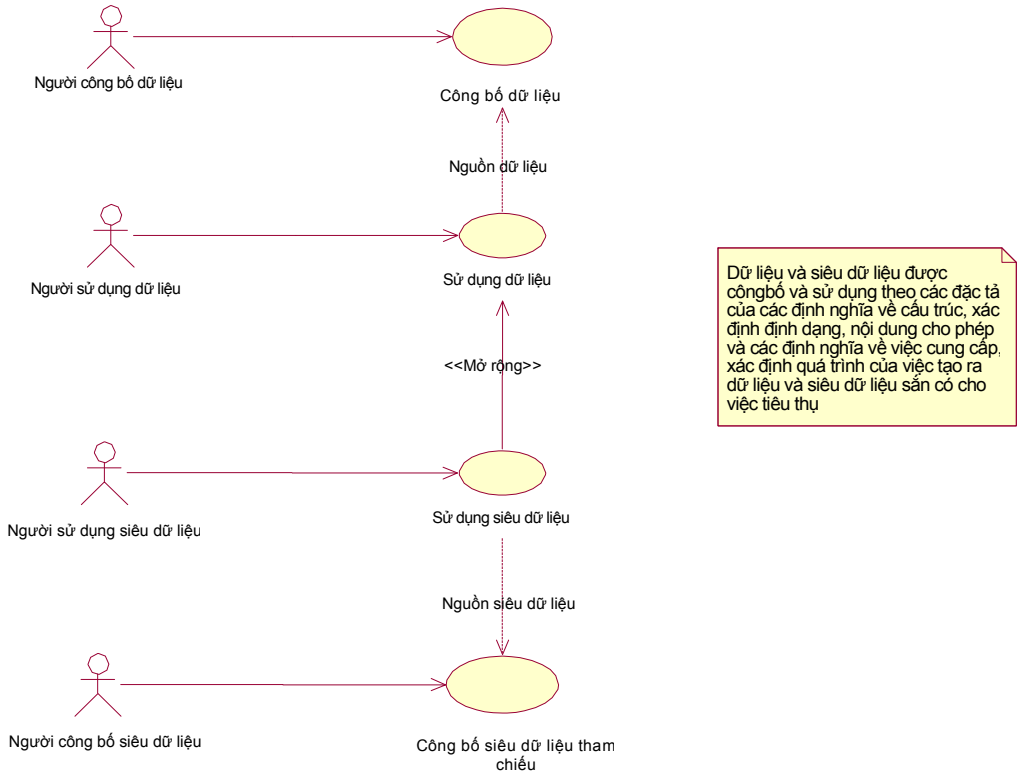
	 Duy trì các định nghĩa cấu trúc	Việc duy trì các định nghĩa cấu trúc. Trường hợp sử dụng này có các trường hợp sử dụng của lớp con cho mỗi sản phẩm có cấu trúc được duy trì.
	 Duy trì danh sách mã  Duy trì lược đồ khái niệm  Duy trì Lược đồ phân loại  Duy trì tập khóa (định nghĩa cấu trúc dữ liệu)  Duy trì định nghĩa cấu trúc siêu dữ liệu  Duy trì cấu trúc khối hộp  Duy trì lược đồ mã có thứ bậc	Việc tạo và duy trì tập khóa (định nghĩa cấu trúc dữ liệu), định nghĩa cấu trúc siêu dữ liệu, cấu trúc khối hộp và việc hỗ trợ các sản phẩm để sử dụng, như danh sách mã và lược đồ khái niệm.
	 Duy trì quy tắc phân loại báo cáo	

 Cơ quan duy trì các định nghĩa cung cấp		Có trách nhiệm duy trì các định nghĩa cung cấp dữ liệu và siêu dữ liệu.
	 Duy trì các định nghĩa về việc cung cấp	Việc duy trì các định nghĩa cung cấp. Trường hợp sử dụng này có các trường hợp sử dụng của lớp con cho mỗi sản phẩm có cấu trúc được duy trì
	 Duy trì nhà cung cấp dữ liệu  Duy trì định nghĩa luồng dữ liệu  Duy trì định nghĩa luồng siêu dữ liệu	Việc tạo và duy trì các sản phẩm để hỗ trợ định nghĩa cung cấp dữ liệu và siêu dữ liệu, ví dụ như danh sách người cung cấp dữ liệu, các định nghĩa luồng dữ liệu, các định nghĩa khối hộp và các thỏa thuận cung cấp để liên kết người cung cấp dữ liệu với các định nghĩa luồng dữ liệu và luồng siêu dữ liệu
	 Duy trì định nghĩa khối hộp  Duy trì thỏa thuận cung cấp	

Hình 7 – Bảng các tác nhân và trường hợp sử dụng đối với việc duy trì các định nghĩa cấu trúc và định nghĩa cung cấp

2.2.2 Công bố và sử dụng dữ liệu và siêu dữ liệu

2.2.2.1 Trường hợp sử dụng



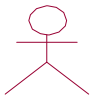
Hình 8 – Tác nhân và trường hợp sử dụng đối với việc công bố và sử dụng dữ liệu và siêu dữ liệu

2.2.2.2 Giải thích sơ đồ

Chú ý rằng trong sơ đồ này “việc công bố” dữ liệu và siêu dữ liệu giống với “việc báo cáo ” dữ liệu và siêu dữ liệu. Trong một số trường hợp có thể tạo ra dữ liệu sẵn có để đáp ứng cả hai chức năng trên. Dữ liệu tập hợp được công bố bởi nhà cung cấp, việc sử dụng các ứng dụng để xử lý dữ liệu, siêu dữ liệu và cấu trúc của nó phải được biết đến. Hơn nữa, việc sử dụng các ứng dụng cũng yêu cầu truy cập siêu dữ liệu (tham chiếu), điều này giúp người sử dụng dữ liệu hiểu hơn về dữ liệu. Đối với dữ liệu, siêu dữ liệu tham chiếu cũng cần được định dạng theo một cấu trúc được duy trì. Người sử dụng dữ liệu và siêu dữ liệu không thể sử dụng dữ liệu và siêu dữ liệu đó trừ khi nó được “công bố”, vì thế có một “nguồn dữ liệu” hoặc “nguồn siêu dữ liệu” phụ thuộc giữa các trường hợp ”sử dụng” và “công bố”.

Trong mọi kịch bản công bố và sử dụng dữ liệu, siêu dữ liệu, việc công bố và sử dụng các ứng dụng sử dụng và công bố cần truy cập đến các định nghĩa cung cấp được duy trì. Các định nghĩa này phải đơn giản, như ai cung cấp dữ liệu, siêu dữ liệu? dữ liệu, siêu dữ liệu gì? cung cấp dữ liệu, siêu dữ liệu cho ai và ở đâu? hoặc các định nghĩa có thể phức tạp hơn cùng với các ràng buộc về dữ liệu, siêu dữ liệu được cung cấp bởi người công bố nào đó. Kịch bản chia sẻ dữ liệu, trong đó dữ liệu và siêu dữ liệu “được lấy” từ các nguồn dữ liệu, các chi tiết của nguồn dữ liệu đó.

2.2.2.3 Định nghĩa

Tác nhân	Trường hợp sử dụng	Mô tả
 Người công bố dữ liệu		Có trách nhiệm công bố dữ liệu theo định nghĩa tập khóa (cấu trúc dữ liệu) và các định nghĩa về việc cung cấp liên quan.
	 Công bố dữ liệu	Công bố một tập dữ liệu. Có thể là một tập dữ liệu tự nhiên hoặc nó có thể tạo ra dữ liệu sẵn có để truy cập ở nguồn dữ liệu ví dụ như cơ sở dữ liệu có thể xử lý truy vấn.
 Người sử dụng dữ liệu		Người sử dụng dữ liệu, có thể là một người sử dụng truy cập thông qua một giao diện sử dụng hoặc một ứng dụng ví dụ như một hệ thống sản phẩm thống kê
	 Sử dụng dữ liệu	Sử dụng dữ liệu được định dạng theo các định nghĩa cấu trúc và tạo ra theo các định nghĩa cung cấp. Dữ liệu thường được kết nối với siêu dữ liệu ở các vị trí khác nhau và được công bố và duy trì một cách độc lập.
 Người công bố siêu dữ liệu		Có trách nhiệm về việc công bố siêu dữ liệu tham chiếu theo định nghĩa cấu trúc siêu dữ liệu và các định nghĩa về việc cung cấp liên quan.
	 Công bố siêu dữ	Công bố một tập siêu dữ liệu tham chiếu. Có thể là tập siêu dữ liệu tự nhiên hoặc nó tạo ra siêu dữ liệu có sẵn để truy cập đến nguồn siêu dữ liệu như kho siêu dữ liệu, có thể xử lý truy vấn.
 Người sử dụng siêu dữ liệu		Người sử dụng dữ liệu hoặc người sử dụng nhiều truy cập thông qua giao diện sử dụng hoặc một ứng dụng ví dụ như hệ thống sản xuất hoặc phổ biến dữ liệu thống kê.

	 Sử dụng siêu dữ liệu	Sử dụng siêu dữ liệu được định dạng theo các định nghĩa cấu trúc và được tạo ra theo các định nghĩa cung cấp

3 Gói SDMX cơ sở

3.1 Giới thiệu

Các kết cấu trong gói SDMX cơ sở bao gồm các khối cơ bản, hỗ trợ nhiều cấu trúc khác nhau trong mô hình. Với lý do này, nhiều lớp trong gói là lớp trừu tượng (ví dụ: chỉ các lớp con được tạo có thể tồn tại trong thực thi).

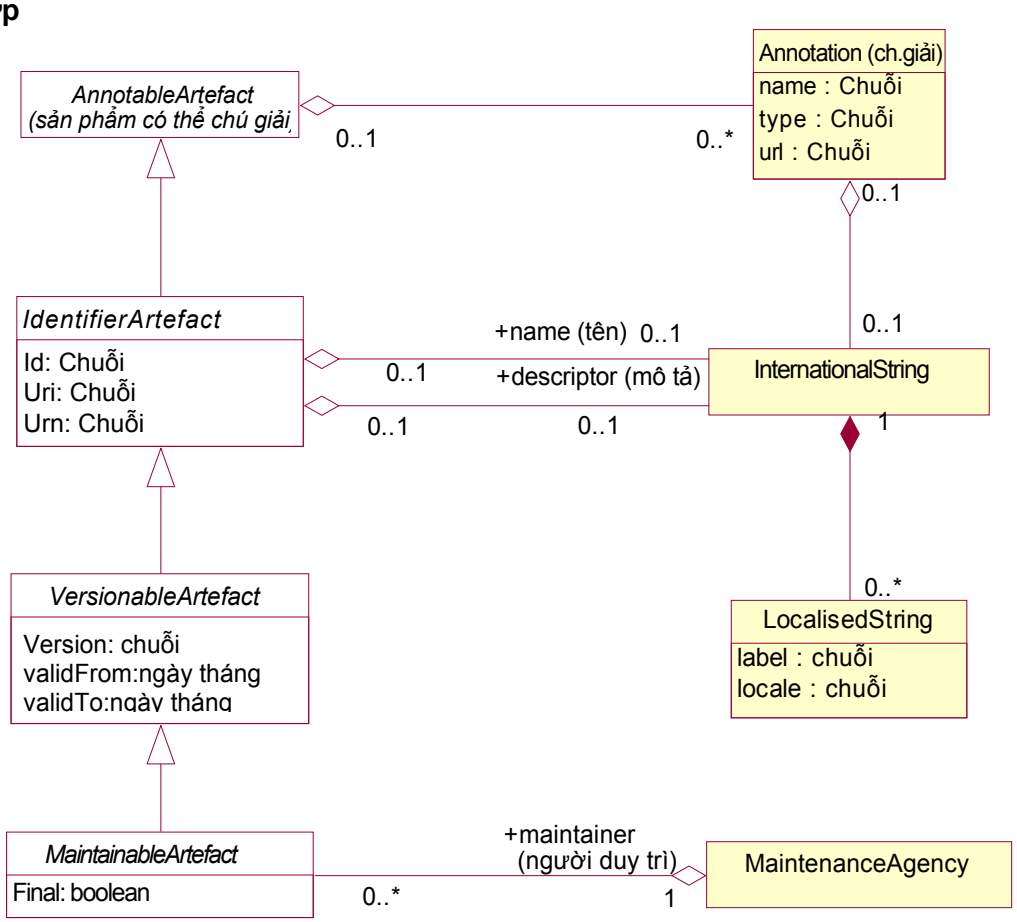
Động cơ thiết lập gói SDMX cơ sở:

- Đây là "Quy phạm hiệu quả nhất" được chấp nhận để định danh các nguyên mẫu cơ bản xuất hiện trong mô hình.
- Định danh các cấu trúc hoặc các "mẫu" chung làm cho việc thông hiểu dễ dàng hơn.
- Định danh các mẫu khuyến khích việc tái sử dụng.

Mỗi sơ đồ lớp trong điều này quan sát các lớp trong gói SDMX cơ sở từ một viễn cảnh khác nhau. Có nhiều cách quan sát chi tiết của các mẫu cụ thể, cùng với các mô tả ngắn gọn chỉ ra tính kế thừa giữa các lớp và quan hệ giữa các lớp.

3.2 Định danh, xác định phiên bản và duy trì

3.2.1 Sơ đồ lớp



Hình 9 – Định danh, xác định phiên bản và duy trì SDMX

3.2.2 Giải thích sơ đồ

3.2.2.1 Diễn giải

Nhóm các lớp này tạo thành trung tâm của các khía cạnh quản trị các đối tượng SDMX. Chúng cung cấp các đặc trưng có thể tái sử dụng bởi các lớp được tạo để hỗ trợ chức năng theo phương ngang như định danh, xác định phiên bản, v.v.

Tất cả các lớp được tạo từ lớp trừu tượng *AnnotableArtefact* có thể có nhiều chú giải (chú thích): điều này hỗ trợ yêu cầu bổ sung các chú thích cho tất cả các phần tử SDMX-ML. Chú thích được sử dụng để truyền thông tin bổ sung mô tả mọi kết cấu SDMX. Thông tin này có thể dưới dạng tham chiếu URL và/hoặc văn bản đa ngôn ngữ (được biểu diễn bởi liên kết với `InternationalString`).

IdentifiableArtefact là một lớp trừu tượng, bao gồm các thuộc tính cơ bản cần thiết cho việc định danh. Tất cả các lớp cụ thể được dựa trên *IdentifiableArtefact* kế thừa khả năng được định danh duy nhất. Chúng cũng kế thừa khả năng chứa các chú thích. Ngoài ra, các vai trò `+description` và `+name` hỗ trợ các mô tả đa ngôn ngữ và tên cho tất cả đối tượng dựa trên *IdentifiableArtefact*. `InternationalString` hỗ trợ biểu diễn một mô tả tại nhiều địa điểm xảy ra (địa điểm tương tự ngôn ngữ nhưng bao gồm sự khác biệt địa lý ví dụ như tiếng Pháp ở Canada, tiếng Anh ở Mỹ v.v). *LocalisedString* hỗ trợ cho việc biểu diễn một mô tả tại một địa điểm.

VersionableArtefact là một lớp trừu tượng kế thừa từ *IdentifiableArtefact* và bổ sung các khả năng xác định phiên bản cho các lớp được tạo từ nó.

MaintainableArtefact bổ sung khả năng cho các lớp được tạo để duy trì thông qua liên kết của nó với MaintenanceAgency. Lớp này có khả năng xác định sản phẩm là bản dự thảo hoặc bản cuối cùng với thuộc tính final.

Chuỗi kế thừa từ *AnnotableArtefact* ***thông qua*** *MaintainableArtefact* cho phép các lớp SDMX kế thừa các đặc trưng chúng cần, từ chú giải đơn giản, thông qua định danh để xác định phiên bản và duy trì.

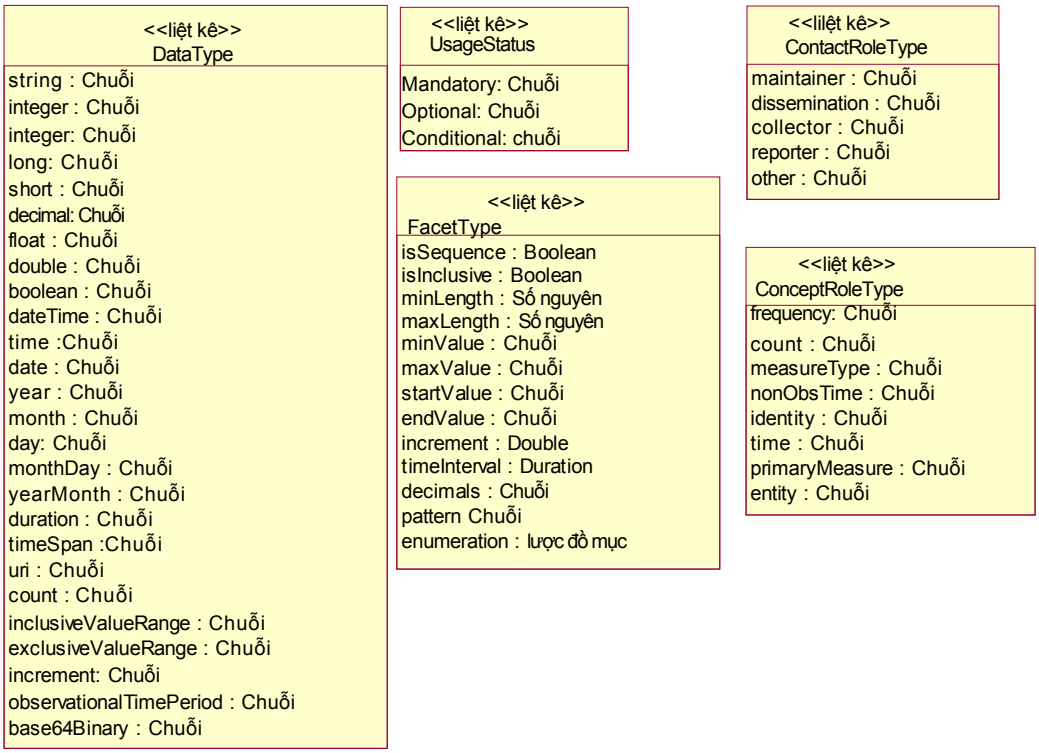
3.2.2.2 Định nghĩa

Lớp	Đặc trưng	Mô tả
AnnotableArtefact	Các lớp con trực tiếp: *IdentifiableArtefact*	Đối tượng của các lớp được tạo từ lớp này có thể đính kèm các chú giải.
Annotation		Thông tin mô tả bổ sung đính kèm với một đối tượng.
	name	Tên được sử dụng để định danh một chú giải.
	type	Quy định cách xử lý một chú giải.
	url	Liên kết với văn bản mô tả bên ngoài.
	+text	Chuỗi quốc tế cung cấp nội dung của chú giải trong văn bản đa ngôn ngữ thông qua vai trò này.
IdentifiableArtefact	Lớp trên: AnnotationArtefact Các lớp con trực tiếp: VersionableArtefact .	Cung cấp định danh cho tất cả các lớp được tạo. Nó cũng cung cấp các chú giải cho các lớp được tạo bởi vì nó là lớp con của sản phẩm có thể chú giải.
	id	Thẻ định danh duy nhất của đối tượng.
	uri	Thẻ định danh tài nguyên chung có thể hoặc không thể phân ly.
	urn	Tên tài nguyên chung - sử dụng trong sổ đăng ký: mọi đối tượng được đăng ký có một urn.
	+description	Mô tả đa ngôn ngữ được cung cấp bởi vai trò này thông qua lớp chuỗi quốc tế

	+name	Tên đa ngôn ngữ được cung cấp bởi vai trò này thông qua lớp chuỗi quốc tế
VersionableArtefact	Lớp trên: *IdentifiableArtefact* Các lớp con trực tiếp: *MaintainableArtefact*	Cung cấp thông tin xác định phiên bản cho tất cả đối tượng được tạo
version		Chuỗi phiên bản theo sau một quy ước đã thỏa thuận
	validFrom	Ngày tháng phiên bản có hiệu lực
	validTo	Ngày tháng phiên bản hết hiệu lực
InternationalString		Chuỗi quốc tế là một tập hợp của chuỗi bản địa hóa và hỗ trợ việc biểu diễn mô tả ở nhiều vị trí.
LocalisedString		Chuỗi bản địa hóa hỗ trợ biểu diễn mô tả ở một địa điểm (địa điểm tương tự như ngôn ngữ khác nhau về địa lý như tiếng Pháp ở Cananda, tiếng Anh ở Mỹ v.v)
	label	Nhãn của một chuỗi
	locale	Vị trí địa lý của chuỗi, ví dụ: tiếng Pháp, Tiếng Pháp ở Canada
MaintainableArtefact	Kế thừa từ VersionableArtefact Các lớp được tạo: *StructureUsage* *Structure* *ItemScheme*	Lớp trừu tượng để nhóm các sản phẩm siêu dữ liệu có cấu trúc chính cùng nhau được duy trì bởi cơ quan duy trì.
	final	Xác định một sản phẩm được duy trì là bản dự thảo hoặc bản cuối cùng
	+maintainer	Cơ quan duy trì sẽ duy trì các lớp được tạo như được quy định bởi vai trò này
MaintenanceAgent		Xem điều về “Tổ chức”

3.3 Kiểu dữ liệu

3.3.1 Sơ đồ lớp



3.3.2 Giải thích sơ đồ

3.3.2.1 Diễn giải

Kiểu số đếm của `UsageStatus` được sử dụng như một kiểu dữ liệu về thuộc tính, trong đó giá trị của thuộc tính là trường hợp của lớp phải lấy một trong các giá trị của `UsageStatus` (ví dụ: bắt buộc (mandatory), tùy chọn (optional) hoặc điều kiện (conditional)).

Kiểu số đếm của `AttributeValueType` được sử dụng như kiểu dữ liệu về giá trị thuộc tính, chỉ ra định dạng của nó.

Kiểu số đếm của `ConceptRoleType` được sử dụng như kiểu dữ liệu về vai trò của thuộc tính để chỉ ra rằng một thành phần đóng vai trò trong tập khóa (định nghĩa cấu trúc dữ liệu). Vai trò này bổ sung cho mọi lớp cấu trúc mô hình như: `Dimension`, *`Measure`* và `DataAttibute`. Việc mô tả các vai trò khác nhau được tìm thấy trong điều về tập khóa (điều 5).

Kiểu số đếm của `DataType` được sử dụng để quy định định dạng hợp lệ về nội dung của `Concept` khi được quy định để sử dụng trên *`Component`* về *`Structure`* (như `Dimension` trong `KeyFamily`). Việc mô tả các kiểu khác nhau có thể được tìm thấy trong lược đồ khái niệm (điều 4.4).

Kiểu số đếm của `FacetType` được sử dụng để đưa ra ngữ cảnh cho một `facetValue` cụ thể. Việc sử dụng giá trị này và sự mô tả các kiểu dữ liệu khác nhau được tìm thấy trong lược đồ khái niệm (điều 4.4).

3.3.2.2 Định nghĩa

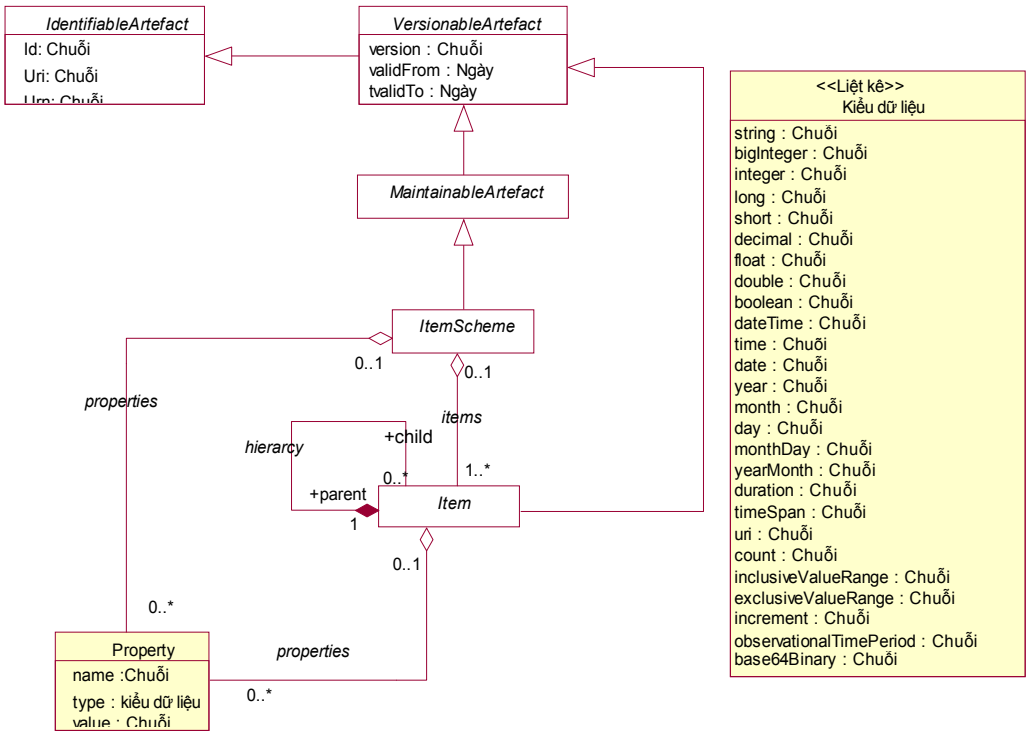
Lớp	Đặc trưng	Mô tả
UsageStatus		Liệt kê các giá trị có thể để một thuộc tính có thể chiếm khi được gán với kiểu dữ liệu của trạng thái sử dụng
	mandatory	Việc sử dụng là bắt buộc
	optional	Việc sử dụng là tùy ý.
	conditional	Việc sử dụng là bắt buộc khi điều kiện nào đó thỏa mãn
ConceptRoleType		Liệt kê các định dạng có thể để giá trị thuộc tính có thể mang khi nó được gán với kiểu dữ liệu về thuộc tính (ví dụ trong vai trò khái niệm). Ý nghĩa ngữ nghĩa của kiểu vai trò trong bảng liệt kê được xác định với cấu trúc mà chúng được sử dụng (ví dụ tập khóa)
Datatype		Liệt kê các định dạng mà giá trị thuộc tính có thể mang khi nó được gán với kiểu dữ liệu về thuộc tính (ví dụ: kiểu). Ngữ nghĩa của các kiểu dữ liệu trong bảng liệt kê được xác định với cấu trúc chúng được sử dụng(ví dụ: Lược đồ khái niệm).
FacetType		Liệt kê các định dạng mà giá trị thuộc tính có thể mang khi nó được gán với kiểu dữ liệu về thuộc tính (ví dụ: Kiểu Facet). Ý nghĩa ngữ nghĩa của kiểu dữ liệu trong bảng liệt kê được xác định với cấu trúc chúng được sử dụng (ví dụ: Lược đồ khái niệm).

3.4 Mẫu lược đồ mục

3.4.1 Ngữ cảnh

Lược đồ mục là mẫu kiến trúc cơ bản cho phép tạo ra các lược đồ danh sách để sử dụng trong các nguyên tắc phân loại đơn giản, ví dụ: ItemScheme là cơ sở cho CategoryScheme, CodeList, ConceptScheme và CodeSet.

3.4.2 Sơ đồ lớp



Hình 10 – Mẫu lược đồ mục

3.4.3 Giải thích sơ đồ

3.4.3.1 Diễn giải

ItemScheme là lớp trừu tượng xác định một tập *Item* (lớp này cũng là lớp trừu tượng). Mục đích chính của nó là xác định cơ chế sử dụng để tạo ra các nguyên tắc phân loại có thể phân loại các phần khác nhau của mô hình thông tin SDMX. *ItemScheme* được tạo từ *MaintainableArtefact,* lớp này giúp cho *ItemScheme* có khả năng được chú thích, có định danh, xác định phiên bản và được liên kết với MaintenanceAgency. Một ví dụ về các lớp cụ thể là CategoryScheme và Category liên kết.

Item kế thừa từ *VerionableArtefact*, lớp này giúp cho *Item* có khả năng chú thích, có định danh, xác định phiên bản, do đó có id, uri và các thuộc tính urn, tên và mô tả theo dạng của InternationalString. Không giống với lớp cha *ItemScheme*, bản thân *Item* không phải là *MaintainableArtefact* do đó không thể có MaintenanceAgency độc lập (ví dụ, *Item* có cùng cơ quan với *ItemScheme*).

Item là lớp phân cấp vì vậy *Item* có các *Item* con. Sự hạn chế của liên kết phân cấp là một *Item* con chỉ có một *Item* cha.

ItemScheme và *Item,* có thể có tất cả Property tùy chọn, đưa ra thêm khả năng vào các đặc tính mở rộng. Giải thích các kiểu dữ liệu khác nhau có thể được tìm thấy trong lược đồ khái niệm (điều 4.4).

3.4.3.2 Định nghĩa

Lớp	Đặc trưng	Mô tả
ItemScheme	kế thừa từ *MaintainableArtefact* Các lớp con trực tiếp: `CategoryScheme` `ConceptScheme` `CodeList` `OrganisationScheme` `ItemSchemeAssociation`	Thông tin mô tả về việc sắp xếp và phân chia các đối tượng thành các nhóm dựa trên các đặc điểm chung của các đối tượng đó.
	`property`	Liên kết với một đặc tính của mục.

Item	kế thừa từ *IdentifiableArtefact*
	Các lớp con trực tiếp:
	`Category`
	`Concept`
	`Code`
	`Association`
		là một mục nội dung trong lược đồ mục. Mục có thể là một nút trong nguyên tắc phân loại hoặc bản thể học, một mã trong danh sách mã v.v.

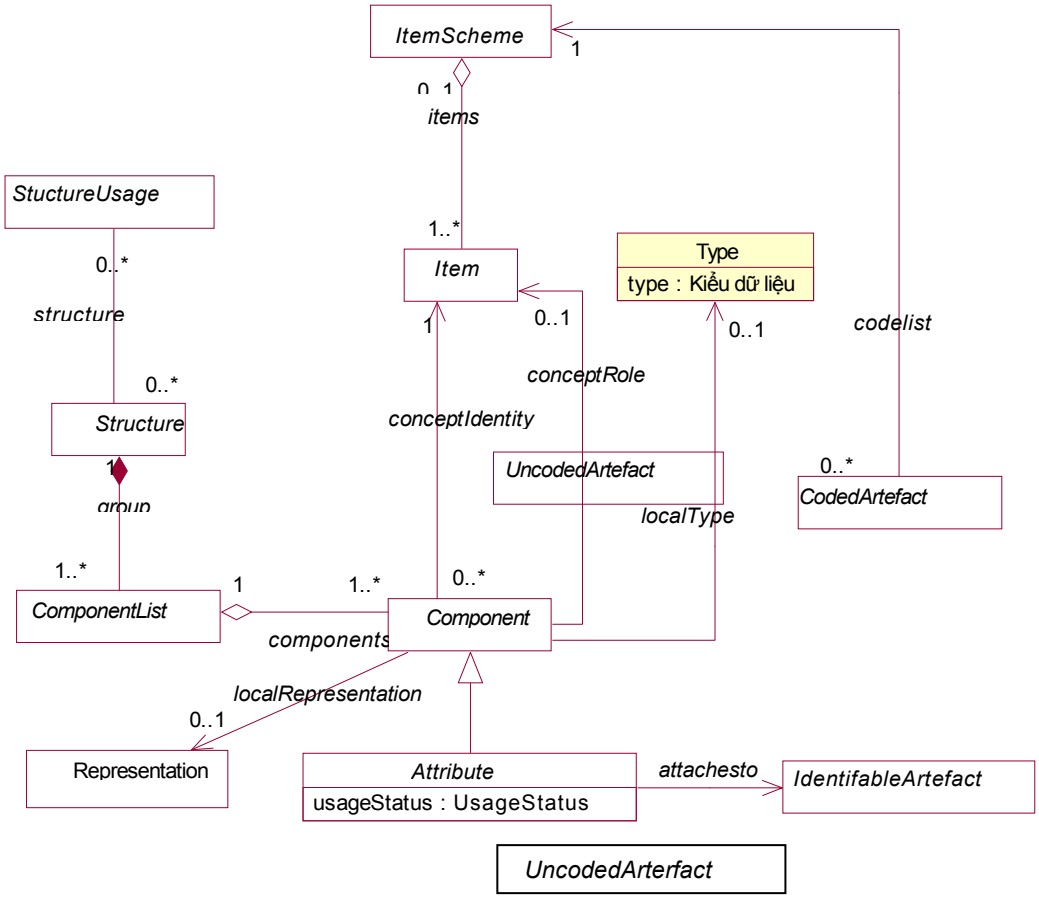
	`hierarchy`	Điều này cho phép một mục tùy chọn có một hoặc nhiều mục con.
	`property`	Liên kết với một đặc tính của mục.
`Property`		Đặc tả giá trị mà ngữ nghĩa được định danh bởi tên của nó.
	`name`	Tên của đặc tính.
	`type`	Quy định kiểu dữ liệu cho đặc tính của thuộc tính. Các kiểu này là danh sách liệt kê trong liệt kê kiểu dữ liệu
	`value`	Giá trị của đặc tính.

3.5 Mẫu cấu trúc

3.5.1 Ngữ cảnh

Cấu trúc này là một mẫu kiến trúc cơ bản cho phép đặc tả các cấu trúc phức tạp được trình bày bằng bảng, điều này thường được tìm thấy trong dữ liệu thống kê (ví dụ như tập khóa, khối hộp và các định nghĩa cấu trúc siêu dữ liệu). Cấu trúc là một tập các danh sách mã có thứ tự. Mẫu làm cơ sở hình thành cấu trúc trình bày bằng bảng, vì vậy, sự phổ biến giữa các định nghĩa cấu trúc này có thể được hỗ trợ bởi phần mềm và các cấu trúc cú pháp chung.

3.5.2 Sơ đồ lớp



Hình 11 – Mẫu cấu trúc

3.5.3 Giải thích sơ đồ

3.5.3.1 Diễn giải

Structure là lớp trừu tượng chứa một tập gồm một hoặc nhiều *ComponentList* (đây cũng là lớp trừu tượng). Ví dụ về *ComponentStructure* là KeyFamily. Các *ComponentList* được nhúng trong *Structure*, được chỉ ra bởi hình thoi trên liên kết nhóm.

ComponentList là danh sách gồm một hoặc nhiều *Component*. *ComponentList* có một vài lớp mô tả dựa trên cơ sở *ComponentList*: KeyDescriptor, GroupKeDescriptor, MeasureDescriptor và AttributeDescriptor ví dụ là KeyFamily. Trong trường hợp *ComponentList* hoạt động như *ComponentList*, thì (các) *Component* của nó là (các) Dimension.

Mỗi Component mang ngữ nghĩa từ một mục trong *ItemScheme* như Concept trong một ConceptScheme. Hơn nữa, *Component* được xác định có một hoặc nhiều vai trò trong cấu trúc đó, điều này được định danh bởi +conceptRole liên kết với một *Item* trong *ItemScheme* để xác định các vai trò. *Component* cũng có một Type quy định localType, điều này cho phép một lớp như Dimension quy định kiểu dữ liệu cục bộ về *Structure* mà nó được chứa trong đó (với Dimension nó sẽ là KeyFamily), do đó có thể ghi đè bất kỳ Type quy định về mục chứa conceptIdentity (trong trường hợp của Dimension sẽ là một Concept).

Lớp con cụ thể của Component là Attribute. Các Attribute được sử dụng trong các Structure

cụ thể (ví dụ như KeyFamily) và được quy định để "có thể đính kèm" với các thành phần cụ thể trong mô hình. Điều này được hỗ trợ bởi liên kết "có thể đính kèm" để liên kết với *IdentifiableArtefact*. Sự liên kết này được quy định trong các mô hình vững chắc, sử dụng mẫu cấu trúc này để quy định các thành phần mô hình thực trong đó thuộc tính có thể được đính kèm.

Structure có thể được sử dụng bởi một hoặc nhiều *StructureUsage*. Ví dụ về điều này trong các thuật ngữ của các lớp là DataflowDefinition (lớp con của *StructureUsage*) có thể sử dụng một KeyFamily cụ thể (lớp con của Structure) và các kết cấu tương tự áp dụng cho MetadataflowDefinition (kết nối với MetadataStructureDefinition) và CubeDefinition (kết nối với CubeStructure).

Cuối cùng, mẫu bao gồm *CodedArtefact* và *UncodedArtefact*. Mô hình này phân biệt hai cách "biểu diễn" cơ bản đối với các thành phần trong cấu trúc. *CodedArtefact* liên kết với *ItemScheme* (thường là một CodeList) để xác định nội dung hợp lệ của nó, trong khi *UncodedArtefact* không có liên kết với danh sách chính thức để quy định nội dung hợp lệ. Tuy nhiên, UncodedArtefact có thể được biểu diễn dưới dạng không mã hoá thay vì dạng văn bản. Các biểu diễn này được mô tả trong điều 4.4 (Lược đồ khái niệm).

3.5.3.2 Định nghĩa

Lớp	Đặc trưng	Mô tả
StructureUsage	kế thừa từ *MaintainableArtefact* Các lớp con trực tiếp: DataflowDefinition (xem hình 22) MetadataflowDefinition (xem hình 22)	Sản phẩm mà các thành phần của nó được mô tả bằng cấu trúc. Trong các thuật ngữ (các lớp con), ví dụ: Định nghĩa luồng dữ liệu được kết nối với một cấu trúc cho trước – trong trường hợp tập khóa này.
	Structure	Liên kết với cấu trúc quy định cấu trúc của sản phẩm.
Structure	kế thừa từ *MaintainableArtefact* Các lớp con trực tiếp: KeyFamily MetadataStructure Definition	Đặc tả trừu tượng của một danh sách mã để định nghĩa một cấu trúc phức tạp trình bày bằng bảng. Ví dụ: các khái niệm thống kê, danh sách mã và tổ chức trong định nghĩa cấu trúc dữ liệu và siêu dữ liệu, xác định bởi một viện trung tâm, thường đối với các trao đổi thông tin thống kê với các đối tác
	grouping	Liên kết tập hợp với một hoặc nhiều thành phần tạo ra danh sách đó.

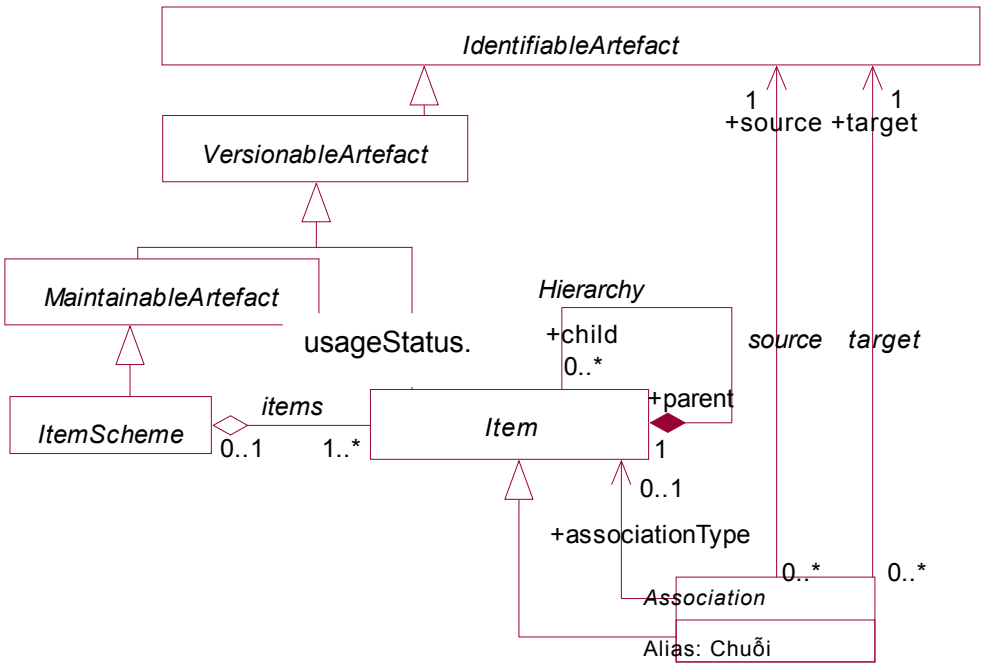
ComponentList	kế thừa từ *IdentifiableArtefact* Các lớp trực tiếp: KeyDescriptor GroupKeyDescriptor MeasureDescriptor AttributeDescriptor TargetIdentifier PartialTargetIdentifier ConceptDescriptor	Định nghĩa trừu tượng của một danh sách các thành phần. Ví dụ: một mô tả khóa, xác định danh sách các chiều kích thước tạo ra khóa cho một tập khóa.
	Components	Liên kết tập hợp tới một hoặc nhiều thành phần tạo ra danh sách
Component	kế thừa từ IdentifiableArtefact Các lớp trực tiếp: Measure Attribute Dimension	Thành phần là một lớp trên trừu tượng được sử dụng để xác định các mục dữ liệu và siêu dữ liệu về số lượng và chất lượng thuộc về một danh sách thành phần cấu trúc. Thành phần được cải tiến thông qua các lớp con của nó.
Attribute	kế thừa từ *Component* Các lớp con trực tiếp: UncodedDataAttribute CodedDataAttribute *MetadataAttribute*	Lớp trừu tượng được sử dụng để cung cấp thông tin về chất lượng.
	usageStatus	Xác định trạng thái sử dụng quy định bởi trạng thái sử dụng kiểu dữ liệu đó.
UncodedArtefact	Các lớp con trực tiếp: UncodedDataAttribute UncodedMetadata Attribute UncodedMeasure	Sản phẩm không mã hóa là một lớp trừu tượng được sử dụng để xác định các giá trị có số lượng và chất lượng hoặc giá trị văn bản tự do lấy từ tập giá trị được duy trì.
CodedArtefact	Các lớp con trực tiếp: Dimension CodedDataAttribute CodedMeasure IdentifierComponent CodedMetadata Attribute	Sản phẩm được mã hóa là một lớp trừu tượng được sử dụng để xác định các giá trị chất lượng được lấy từ tập giá trị được duy trì.
	codelist	Liên kết với lược đồ mục cho phép các lớp con xác định danh sách mã mà từ đó thành phần này lấy giá trị của nó.

3.6 Mẫu liên kết

3.6.1 Ngữ cảnh

Cấu trúc này là mẫu kiến trúc cơ bản cho phép đặc tả các cấu trúc phức tạp được trình bày bằng bảng thường trong dữ liệu thống kê (ví dụ tập khóa).

3.6.2 Sơ đồ lớp



Hình 12 – Sơ đồ lớp của mẫu liên kết

3.6.3 Giải thích sơ đồ

3.6.3.1 Diễn giải

Mẫu liên kết cho phép các liên kết giữa hai *IdentifiableArtefact* bất kỳ. Liên kết này có một kiểu mã hóa được quy định bởi mục trong một *ItemScheme*. Liên kết này là một *VersionableArtefact*, cho phép các liên kết giữa các đối tượng tiến triển theo thời gian. Lớp *Association* cũng là một *Item*, do đó nó có thể chứa các *Association* con. Điều này có lợi cho việc trình bày ánh xạ giữa các danh sách và các hệ thống phân cấp. Ví dụ, một liên kết có thể ánh xạ hai lớp CodeList với nhau và một tập các lớp *Association* con có thể ánh xạ các lớp Code riêng. Một hệ thống phân cấp phức tạp hơn sẽ ánh xạ tất cả thành phần trong một tập khóa, bao gồm các lớp CodeList và Code được sử dụng bởi các thành phần. Điều này được biểu diễn dưới dạng lược đồ như sau:

KeyFamily↬[Dimension, DataAttribute, Measure]↬CodeList↬Code.

Việc sử dụng mẫu này được mô tả trong tập cấu trúc (điều 9).

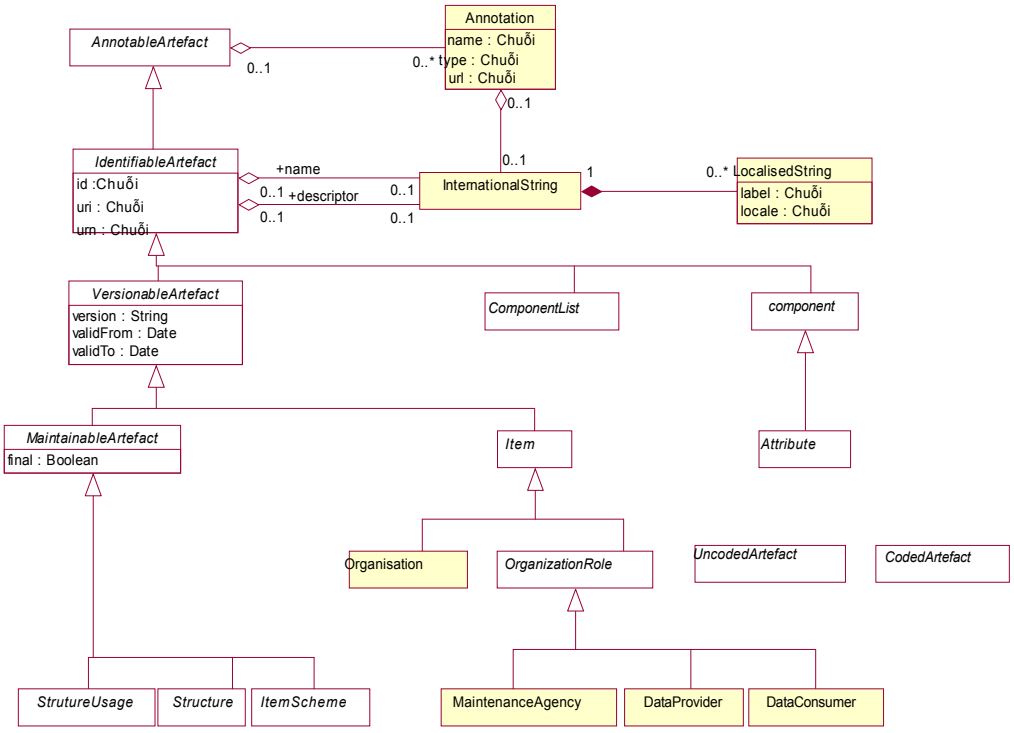
Thuộc tính alias được sử dụng để quy định tên trung tính liên quan đến các ánh xạ theo cặp do đó tạo thuận lợi cho việc truy vấn thông qua tập các sản phẩm được ánh xạ.

3.6.3.2 Định nghĩa

Lớp	Đặc trưng	Mô tả
Association	kế thừa từ	Kết nối hai sản phẩm có thể định danh trong một liên kết nguồn và đích.
	+source	Liên kết với sản phẩm có thể định danh nguồn.
	+target	Liên kết với sản phẩm có thể định danh đích.
	+associationType	Liên kết với một mục quy định vai trò của kết nối giữa sản phẩm có thể định danh nguồn và đích.
	Alias	Quy định một tên trung tính liên quan đến các ánh xạ theo cặp của các sản phẩm có thể định danh.

3.7 Tính kế thừa

3.7.1 Sơ đồ lớp



Hình 13 – Tính kế thừa trong các cấu trúc cơ sở

3.7.2 Giải thích sơ đồ

3.7.2.1 Diễn giải

Sơ đồ ở trên chỉ ra tính kế thừa trong các cấu trúc cơ sở. Nhiều lớp được giới thiệu và xác định trong gói cụ thể liên quan đến chúng, chủ yếu là: Định nghĩa cấu trúc dữ liệu và định nghĩa cấu trúc siêu dữ liệu.

Chú ý rằng cả *CodedArtefact* và *UncodedArtefact* đều không kế thừa từ bất kỳ lớp cơ sở nào và bản thân chúng cũng không có định danh. Các lớp kế thừa từ các lớp này cũng kế thừa từ một lớp có định danh (ví dụ: trong trường hợp thuộc tính dữ liệu là `CodedDataAttribute`)

4 Lược đồ mục cụ thể

4.1 Giới thiệu

Các cấu trúc để sắp xếp các đối tượng vào các hệ thống phân cấp hoặc các danh sách dựa trên các đặc điểm và được duy trì như một nhóm kế thừa từ *ItemScheme*. Các lớp đó:

- `CodeList`
- `ConceptScheme`
- `CategoryScheme`
- `OrganisationScheme`
- `ItemSchemeAssociation`

- `TransformationScheme`

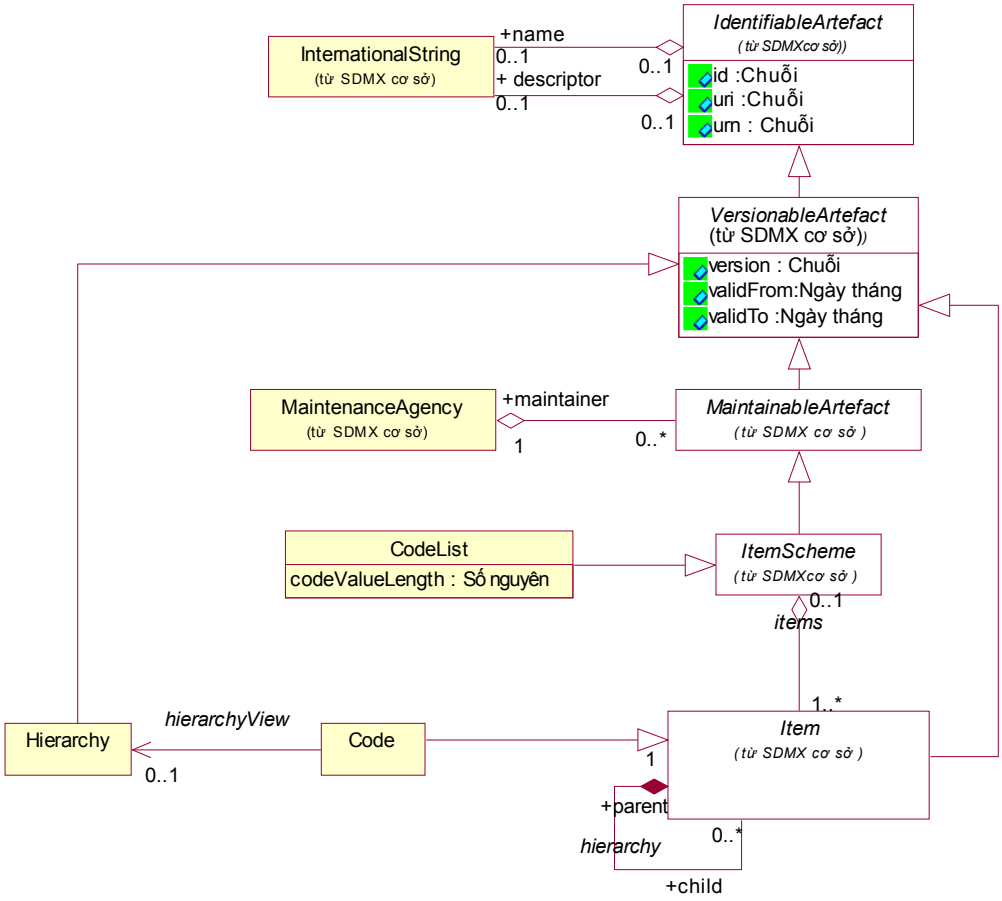
`TransformationScheme` được mô tả trong điều phép biến đổi và biểu thức (điều 12). Điều này mô tả các đặc tả chuyên môn của *`ItemScheme`*.

4.2 Quan điểm tính kế thừa

Các quan điểm về quan hệ và tính kế thừa cũng được chỉ ra trong các sơ đồ dưới đây.

4.3 Danh sách mã

4.3.1 Sơ đồ lớp



Hình 14 – Sơ đồ lớp của danh sách mã

4.3.2 Giải thích sơ đồ

4.3.2.1 Diễn giải

Lớp `CodeList` kế thừa từ *`ItemScheme`* và lớp `Code` kế thừa từ *`Item`*, do đó, cả hai lớp này đều có các thuộc tính sau:

- `id`
- `uri`
- `urn`
- `version`

- validFrom
- validTo

Các lớp này cũng có liên kết với lớp InternationalString để hỗ trợ tên đa ngôn ngữ, một mô tả đa ngôn ngữ tùy chọn, một liên kết với Annotation để hỗ trợ các chú thích (không được hiển thị).

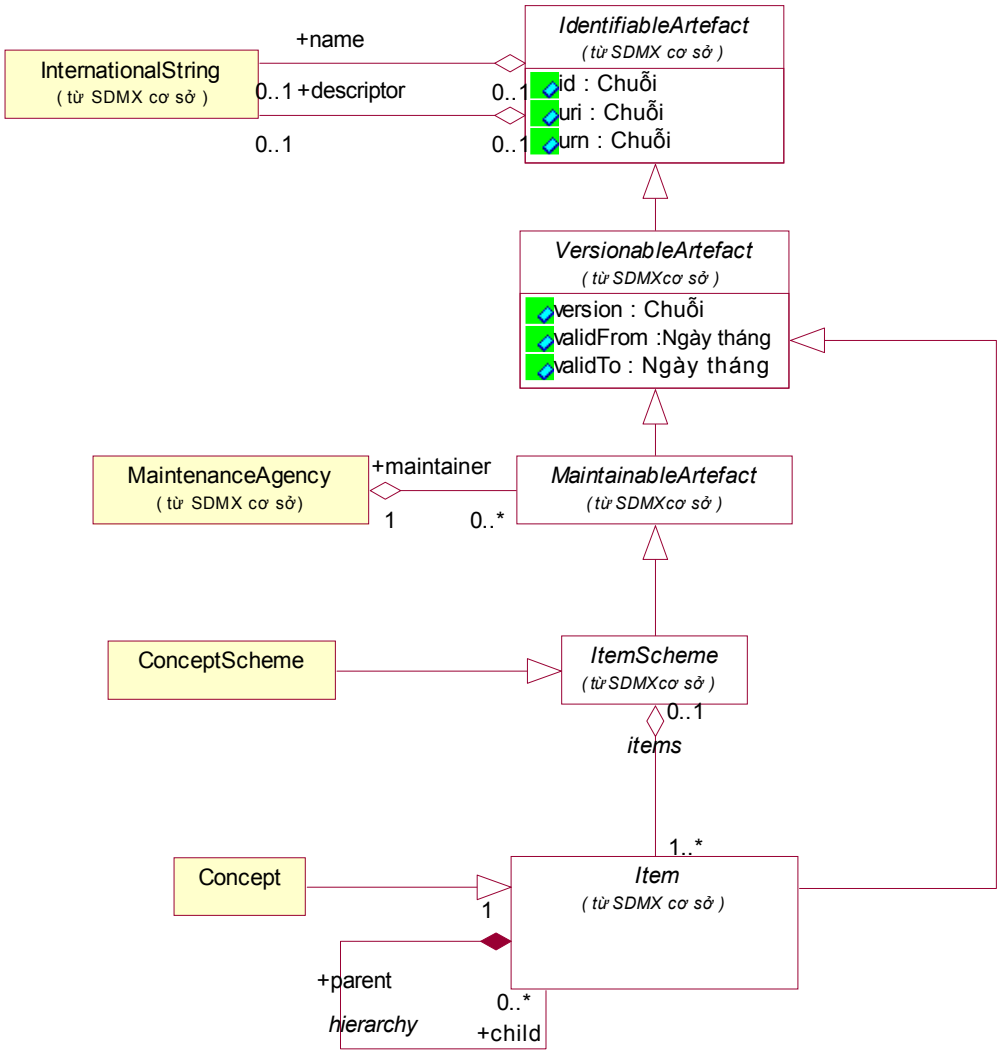
Thông qua việc kế thừa này, lớp CodeList bao gồm một hoặc nhiều Code và bản thân Code này có thể có một hoặc nhiều Code con trong liên kết hierarchy. Chú ý rằng một Code con chỉ có một Code cha trong liên kết này. Một CodeSet phức tạp cho phép nhiều lớp cha và nhiều hệ thống phân cấp được mô tả sau đó. Một HierachicalCodeScheme phức tạp cho phép nhiều lớp cha và hệ thống phân cấp được môt tả sau đó. Trong HierachicalCodeScheme, Code được tham chiếu từ HierarchicalCodeScheme đó, nhưng có yêu cầu kết nối từ Code đến *Hierarchy* trong một HierarchicalCodeScheme (kết nối như vậy sẽ hỗ trợ các ánh xạ mã – xem điều 9) và được hỗ trợ thông qua liên kết hierarchyView.

4.3.2.2 Định nghĩa

Lớp	Đặc trưng	Mô tả
CodeList	kế thừa từ *ItemScheme*	Danh sách mà từ đó các khái niệm thống kê (các khái niệm được mã hóa) lấy các giá trị của chúng. Trong mô hình này, các khái niệm được mã hóa là các lớp con của sản phẩm được mã hóa.
	codeValueLength	Độ dài của một mã (ví dụ: thẻ định danh) trong danh sách mã đó.
	/items	Liên kết với các mã đó.
	/	
Code	Kế thừa từ *Item*	Tập các ký tự, chữ số, ký hiệu độc lập với ngôn ngữ biểu diễn khái niệm mà ý nghĩa của nó được mô tả theo ngôn ngữ tự nhiên.
	hierarchy	Liên kết với các mã cha và con.
	hierarchyView	Liên kết với hệ thống phân cấp.

4.4 Lược đồ khái niệm

4.4.1 Sơ đồ lớp kế thừa



Hình 15 – Sơ đồ lớp của lược đồ khái niệm

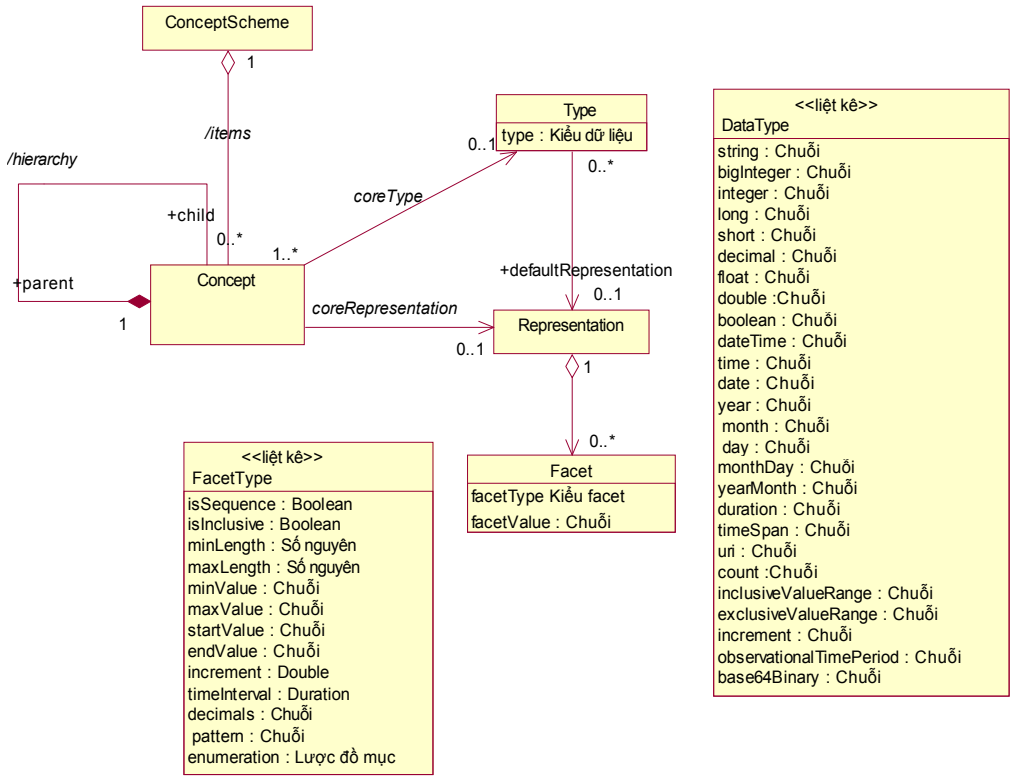
4.4.2 Giải thích sơ đồ

Lớp `ConceptScheme` kế thừa *`ItemScheme`* và lớp `Concept` kế thừa từ *`Item`* , do đó, cả hai lớp này đều có các thuộc tính sau:

- `id`
- `uri`
- `urn`
- `version`
- `validFrom`
- `validTo`

Cả hai lớp này cũng có liên kết với `InternationalString` để hỗ trợ tên đa ngôn ngữ, một mô tả đa ngôn ngữ tùy chọn và một liên kết với `Annotation` để hỗ trợ các chú giải (không được hiển thị).

4.4.3 Sơ đồ lớp quan hệ



Hình 16 – Sơ đồ lớp quan hệ của lược đồ khái niệm

4.4.4 Giải thích sơ đồ

4.4.4.1 Diễn giải

Lớp `ConceptScheme` có thể có một hoặc nhiều `Concept`. Một `Concept` có thể không có hoặc có nhiều `Concept` con, do đó hỗ trợ một hệ thống phân cấp của các `Concept`. Chú ý rằng một `Concept` con chỉ có một `Concept` cha trong liên kết này. Mục đích của hệ thống phân cấp là liên kết các khái niệm có một quan hệ ngữ nghĩa: ví dụ một CONTACT có thể có một PRIMARY_CONTACT như một khái niệm con. Các lược đồ này không có mục đích xác định các cấu trúc báo cáo: các cấu trúc báo cáo này được xác định trong `KeyFamily` hoặc `MetadataStructureDefinition`.

Lớp `Concept` này được xác định phù hợp với một `Type` như dạng chuỗi, số, v.v. `CoreType` của `concept` và `concept` cũng có một `Representation` đó là `coreRepresentation`, ví dụ: `coreType` và `coreRepresentation` là đặc tả của định dạng và miền giá trị `Concept` khi sử dụng trong một cấu trúc như `KeyFamily` hoặc `MetadataStructureDefinition` trừ khi đặc tả của `Type` hoặc `Representation` được ghi đè trong định nghĩa cấu trúc liên quan. Trong một `ConceptScheme` phân cấp `Type` và `Representation` được kế thừa từ `Concept` cha nếu không được ghi đè ở mức `Concept` con.

Chú ý rằng trong khi `Representation` phụ thuộc vào giá trị của `Type.DataType` (đây là liên kết với vai

trò `defaultRepresentation`) điều này không bắt buộc hiển thị trên mô hình, vì một vài lý do về sự phù hợp với phiên bản 1.0, nên không hỗ trợ tất cả *`Representation`*.

Đa số kiểu dữ liệu SDMX thích hợp với kiểu dữ liệu được tìm thấy trong lược đồ XML và tương đương với hầu hết các nền tảng hiện hành:

Kiểu dữ liệu SDMX	Kiểu dữ liệu lược đồ XML	Kiểu khung cơ cấu .NET	Kiểu dữ liệu Java
String	Xsd: string	System.String	java.lang.String
BigInteger	xsd: integer	System.Decimal	java.math.Biginteger
Integer	xsd:int	System.Int32	int
Long	xsd:long	System.Int64	long
Short	xsd:short	System.Int16	short
Decimal	xsd:decimal	System.Decimal	java.math.BigDecimal
Float	xsd:float	System.Single	float
Double	xsd:double	System.Double	double
Boolean	xsd:boolean	System.Boolean	boolean
DateTime	xsd:datetime	System.DateTime	java.xml.datatype.XmlGregorianCalender
Time	xsd:time	System. DateTime	java.xml.datatype.XmlGregorianCalender
Date	xsd:date	System. DateTime	java.xml.datatype.XmlGregorianCalender
Year, month, Day, yearMonth	xsd:g*	System. DateTime	java.xml.datatype.XmlGregorianCalender
Duration	xsd:duration	System.Timespan	java.xml.datatype.Duration
Base64Binary	xsd:base64Binary	System.Byte[]	Byte[]
URI	xsd:anyURI	System.Uri	Java.net.URI hoặc java.lang.String

Một số kiểu dữ liệu SDMX không có các tương ứng trực tiếp ở trên, vì chúng là các biểu diễn hỗn hợp:

- Timespan (Thời gian, ngày tháng bắt đầu + khoảng thời gian)
- ObservationalTimePeriod (khoảng thời gian quan sát) (một kiểu ngày tháng, thời gian, thời gian ngày tháng và một tập các mã về các khoảng thời gian thông thường – xem phần tiêu chuẩn Hướng dẫn người thực thi).

Như đã trình bày ở trên, miền giá trị của một `Type` được thể hiện bởi một `Representation`. `Representation` này được chứa trong các `Facet`, mỗi `Representation` truyền thông tin đặc điểm liên quan đến định nghĩa một miền giá trị. Thông thường một tập các `Facet` cần thiết để truyền ngữ nghĩa yêu cầu. Ví dụ, một trình tự được xác định bởi tối thiểu hai khía cạnh: một để xác định giá trị bắt đầu và một để xác định sự tăng lên. Các kết hợp đúng về ngữ nghĩa của các `Facet` phụ thuộc vào kiểu mà chúng giới hạn, nhưng được lựa chọn từ bảng các `facetType` sau đây:

Kiểu Facet	Giải thích
isSequence (trình tự)	Nếu đúng, phép biểu diễn là một trình tự tăng lên của các giá trị số nguyên (dải giá trị) hoặc các giá trị ngày tháng/thời gian (dải thời gian). Các khía cạnh startValue (giá trị bắt đầu) và interval (khoảng) hoặc timeInterval (khoảng thời gian) cũng phải được quy định cho một trình tự.
isInclusive (bao gồm)	Nếu đúng, các giá trị hợp lệ đối với phép biểu diễn đó nằm trong dải giá trị/thời gian cho trước, nếu không thì nằm ngoài dải giá trị/thời gian đó.
minLength (độ dài tối thiểu)	Quy định số ký tự tối thiểu cho một giá trị.
maxLength (độ dài tối đa)	Quy định số ký tự tối đa cho một giá trị.
minValue (giá trị nhỏ nhất)	Quy định giá trị số nhỏ nhất.
maxValue (giá trị lớn nhất)	Quy định giá trị số lớn nhất.
startValue (giá trị bắt đầu)	Quy định giá trị bắt đầu cho một trình tự (dải thời gian hoặc giá trị)
endValue (giá trị kết thúc)	Quy định giá trị kết thúc cho một trình tự(dải thời gian hoặc giá trị)
Increment (trị số gia tăng)	Được sử dụng để quy định các bước gia tăng của một dải giá trị. Bắt đầu từ startValue (giá trị bắt đầu) và gia tăng một increment (trị số gia tăng) đến khi đạt endValue (giá trị kết thúc). Sau đó trình tự lại bắt đầu từ startValue (giá trị bắt đầu). Nếu không có giá trị endValue (giá trị kết thúc) được quy định thì chuỗi tiếp tục vô hạn.

timeInterval (khoảng thời gian)	Được sử dụng để quy định các bước gia tăng (các giai đoạn) của một dải thời gian. Bắt đầu từ startValue (giá trị bắt đầu) và gia tăng bằng timeInterval (khoảng thời gian) đến endValue (giá trị kết thúc) đạt được. Sau đó trình tự lại bắt đầu từ startValue (giá trị bắt đầu). Nếu không có giá trị endValue (giá trị kết thúc) được quy định thì chuỗi sẽ tiếp tục một cách vô hạn định.
Decimals (số thập phân)	Biểu diễn có một số lượng các số thập phân theo lý thuyết
Pattern (mẫu)	Biểu diễn là một biểu thức có quy tắc (xem XSD) được diễn đạt như một chuỗi
Enumeration (liệt kê)	Biểu diễn là việc liệt kê các mục trong lược đồ cụ thể, ví dụ như danh sách mã được định danh

4.4.4.2 Định nghĩa

Lớp	Đặc trưng	Mô tả
`ConceptScheme`	kế thừa từ *`ItemScheme`*	Thông tin mô tả đối với một sắp xếp hoặc phân chia các khái niệm thành các nhóm dựa trên các đặc điểm chung cho các đối tượng đó.
	`/items`	Liên kết khái niệm.
`Concept`	kế thừa từ *`Item`*	Khái niệm là một đơn vị kiến thức được tạo ra bởi việc kết hợp duy nhất các đặc điểm.
	`/hierarchy`	Liên kết khái niệm cha và con.
	`coreType`	Liên kết một kiểu dữ liệu.
	`coreRepresentation`	Liên kết một biểu diễn.
`Type`	`type`	Quy định như trí nhớ, định dạng hợp lệ của nội dung được báo cáo như: chữ alpha, số, thời gian.

`Representation`	Các lớp con của lớp trừu tượng:
	`ItemScheme`
	`DataRange`
	`NumericRange`
	`Pattern`
		Quy định nội dung của khái niệm khi báo cáo trong một tập dữ liệu hoặc siêu dữ liệu.

`DateRange`		Dải dữ liệu và tính chu kỳ của ngày tháng trong dải đó.
	`startDate`	Ngày bắt đầu của dải ngày tháng
	`endDate`	Ngày kết thúc của dải ngày tháng.

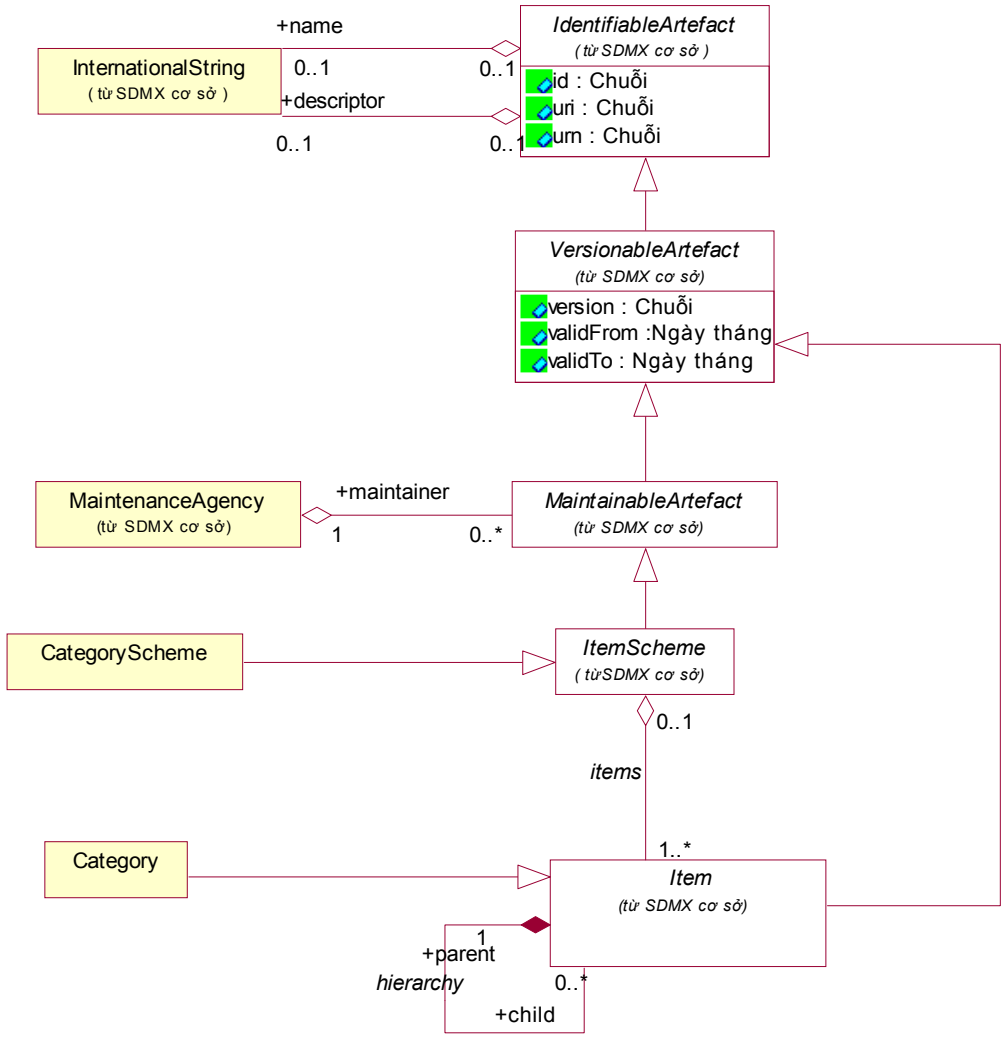
	periodicity	Chu kỳ thời gian trong đó một tập ngày tháng được bao hàm bởi việc gia tăng chu kỳ từ thời gian bắt đầu đến thời gian kết thúc.
NumericRange		Dải chữ số và trị số gia tăng của các số đó.
	maxValue (giá trị lớn nhất)	Giá trị lớn nhất trong dải.
	minValue (giá trị nhỏ nhất)	Giá trị nhỏ nhất trong dải
	increment (trị số gia tăng)	Việc gia tăng tập giá trị có thể được bao hàm bằng cách gia tăng từ giá trị bắt đầu đến giá trị nhỏ nhất.
Pattern		Cách biểu diễn theo một biểu mẫu có thể được trình bày như một biểu thức.
	regularExpression	Biểu thức xác định định dạng nội dung dữ liệu hoặc siêu dữ liệu.
Sequence		Trình tự tất cả các số.
	startValue (giá trị bắt đầu)	Giá trị bắt đầu trong trình tự các giá trị
	increment (trị số gia tăng)	Việc gia tăng tập các giá trị được ám chỉ bằng cách gia tăng từ giá trị bắt đầu hoặc giá trị nhỏ nhất.

4.5 Lược đồ phân loại

4.5.1 Ngữ cảnh

Gói này xác định cấu trúc hỗ trợ định nghĩa và các quan hệ giữa các loại trong Lược đồ phân loại. Điều này tương tự với gói đối với lược đồ khái niệm. Ví dụ về Lược đồ phân loại là lược đồ để phân loại dữ liệu – đôi khi được hiểu như một lược đồ về lĩnh vực chủ đề hoặc Lược đồ phân loại dữ liệu. Một ví dụ khác là lược đồ quy tắc phân loại báo cáo xác định cấu trúc khái niệm của lược đồ báo cáo trong các nhánh, nhiều "tập" riêng về dữ liệu được mô tả bởi cấu trúc cụ thể (đây là một báo cáo trong báo cáo đầu tiên). Các nút riêng trong lược đồ ("loại") có thể được liên kết tới các luồng dữ liệu thực tế, lần lượt kết nối với định nghĩa của cấu trúc luồng dữ liệu (ví dụ KeyFamily).

4.5.2 Sơ đồ lớp



Hình 17 – Sơ đồ lớp của Lược đồ phân loại

4.5.3 Giải thích sơ đồ

4.5.3.1 Diễn giải

Các loại được mô hình hóa như *`ItemScheme`* phân cấp. `CategoryScheme` kế thừa từ *`ItemScheme`*. Và `Category` kế thừa từ *`Item`*, do đó cả hai đều có các thuộc tính sau:

- `id`
- `uri`
- `urn`
- `version`
- `validFrom`
- `validTo`

Cả hai cùng liên kết với `InternationalString` để hỗ trợ tên đa ngôn ngữ, mô tả đa ngôn ngữ tùy

chọn và liên kết với Annotation để hỗ trợ các chú giải (không được hiển thị trên mô hình).

CategoryScheme có thể có một hoặc nhiều Category. Category có thể không có hoặc có một Category con, do đó hỗ trợ một hệ thống phân cấp của các Category. Chú ý rằng một Category có thể chỉ có một Category cha trong liên kết này. Một CodeSet phức tạp hơn cho phép nhiều hệ thống phân cấp hoặc lớp cha được mô hình hóa sau đó.

4.5.3.2 Định nghĩa

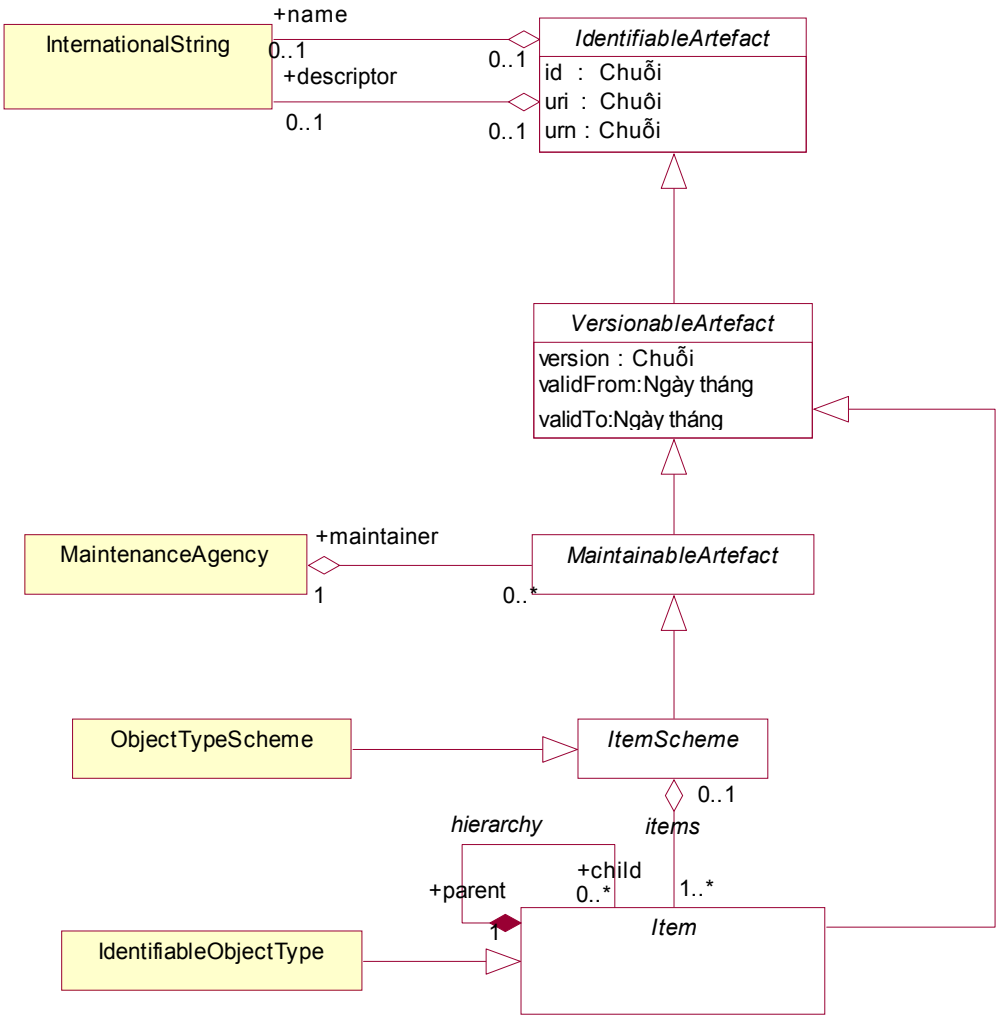
Lớp	Đặc trưng	Mô tả
CategoryScheme	kế thừa từ *ItemScheme*	Thông tin mô tả đối với việc sắp xếp và phân chia các loại thành các nhóm dựa trên các đặc điểm, mà các đối tượng đó có các đặc điểm chung.
	/items	Liên kết loại.
Category	kế thừa từ *Item*	Mục ở bất kỳ mức nào trong một phân loại, các loại sắp xếp thành bảng, các phần, các phần nhỏ, các nhóm, các nhóm nhỏ, các lớp, các lớp con.
	hierarchy	Liên kết với loại cha và con.

4.6 Lược đồ kiểu đối tượng

4.6.1 Ngữ cảnh

Lược đồ này cần thiết trong tài liệu SDMX để định danh kiểu đối tượng trong mô hình SDMX. Ví dụ của một tài liệu như vậy là định nghĩa cấu trúc siêu dữ liệu, quy định tài liệu đính kèm của siêu dữ liệu vào một luồng dữ liệu, một tập khóa hoặc danh sách mã v.v. Đây là một điều cần thiết để định danh kiểu đối tượng và phải được lấy từ “danh sách” hợp lệ của các kiểu đối tượng. Lớp ObjectTypeScheme là một danh sách.

4.6.2 Sơ đồ lớp



Hình 18 – Sơ đồ lớp của lược đồ kiểu đối tượng

4.6.3 Giải thích sơ đồ

4.5.3.1 Diễn giải

Các kiểu đối tượng được mô hình hóa như một *ItemScheme*. `ObjectTypeScheme` kế thừa từ *ItemScheme* và `IdentifiableObjectType` kế thừa từ *Item*, do đó cả hai điều có các thuộc tính sau:

- `id`
- `uri`
- `urn`
- `version`
- `validFrom`
- `validTo`

Cả hai cũng có liên kết với `InternationalString` để hỗ trợ tên đa ngôn ngữ, một mô tả đa ngôn ngữ tùy chọn và một liên kết với `Annotation` hỗ trợ các chú giải (không hiển thị trên mô hình).

`ObjectTypeScheme` có thể có một hoặc nhiều `IdentifiableObjectType`

4.6.3.2 Định nghĩa

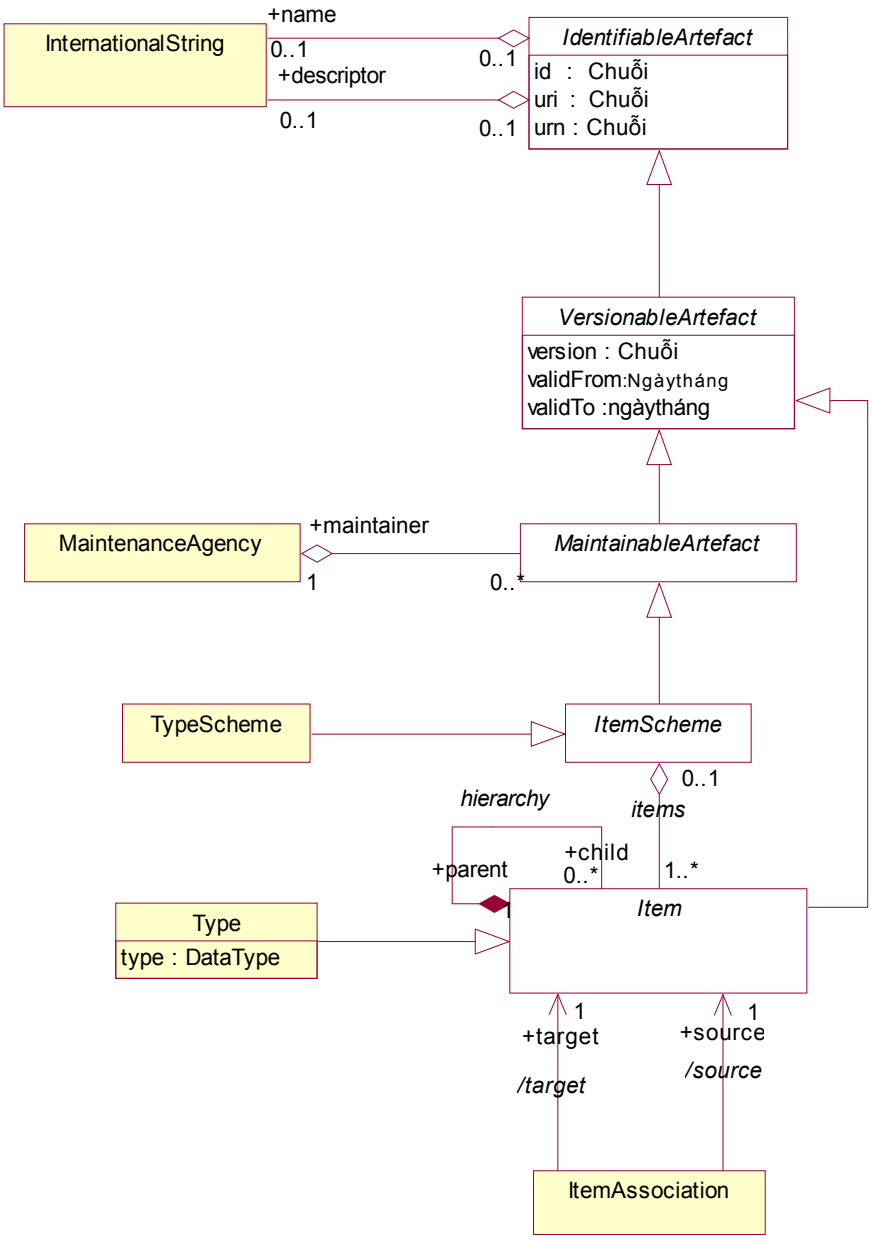
Lớp	Đặc trưng	Mô tả
`ObjectTypeScheme`	kế thừa từ *ItemScheme*	Tập hợp các kiểu đối tượng định danh (được hiểu như các lớp hoặc các thực thể) được chứa trong mô hình dữ liệu hay việc xác định sản phẩm hoặc mô tả kiểu đối tượng
	`/items`	Liên kết với kiểu đối tượng có thể định danh
`IdentifiableObject Type`	kế thừa từ *Item*	Mô tả một tập đối tượng chia sẻ cùng các thuộc tính, thao tác, phương pháp, quan hệ, và ngữ nghĩa, có định danh để kiểu đối tượng(ví dụ. đối tượng riêng lẻ) có thể được tham chiếu.

4.7 Lược đồ về kiểu

4.7.1 Ngữ cảnh

Đây là một lược đồ về kiểu ví dụ như kiểu dữ liệu. Nó được sử dụng để liên kết một kiểu với sản phẩm khác như `ExpressionNode` trong đó một kiểu xác định kiểu dữ liệu về kết quả của biểu thức được xác định trong `ExpressionNode` (xem PHÉP BIẾN ĐỔI VÀ BIỂU THỨC).

4.7.2 Sơ đồ lớp



Hình 19 – Sơ đồ lớp của lược đồ về Kiểu

4.7.3 Giải thích sơ đồ

4.7.3.1 Diễn giải

Các kiểu được mô hình hóa như *ItemScheme*. Lớp `TypeScheme` được kế thừa từ *ItemScheme* và `Type` được kế thừa từ *Item*, do đó cả hai đều có các thuộc tính sau:

- `id`
- `uri`
- `urn`

- version
- validFrom
- validTo

Cả hai lớp này cùng có liên kết với InternationalString để hỗ trợ tên đa ngôn ngữ, một mô tả đa ngôn ngữ tùy chọn và một liên kết với Annotation để hỗ trợ các chú giải (không được hiển thị trên mô hình).

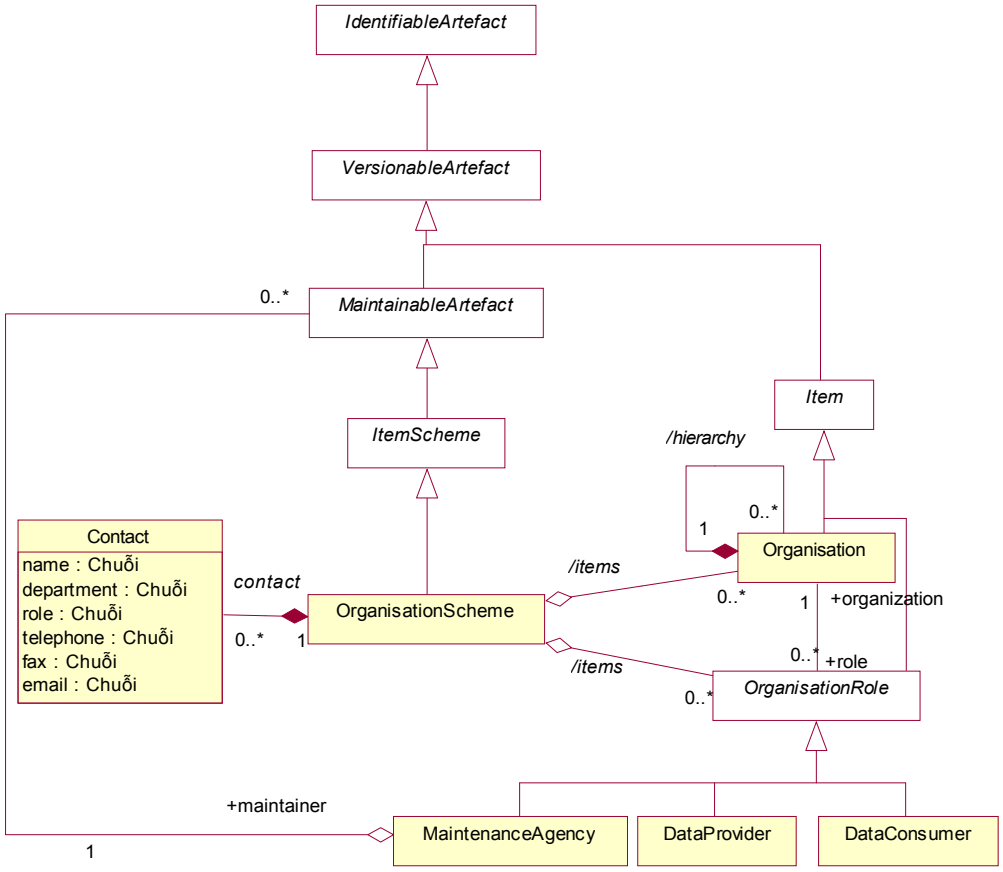
TypeScheme có thể có một hoặc nhiều Type.

4.7.3.2 Định nghĩa

Lớp	Đặc trưng	Mô tả
TypeScheme	kế thừa từ *ItemScheme*	Tập hợp các mục xác định định dạng hợp lệ của dữ liệu để dữ liệu này có thể được xử lý bởi hệ thống máy tính.
	/items	Liên kết với các kiểu trong lược đồ.
Type	kế thừa từ *Item*	Quy định một định dạng dữ liệu do đó có thể được xử lý trong một hệ thống máy tính, ví dụ như số hoặc chuỗi.
	type	Định danh kiểu.

4.8 Lược đồ tổ chức

4.8.1 Sơ đồ lớp



Hình 20 – Sơ đồ lớp của tổ chức

4.8.2 Giải thích sơ đồ

4.8.2.1 Diễn giải

`Organisation` kế thừa từ *`Item`* và cũng có định danh, thông tin về phiên bản và được duy trì trong `OrganisationScheme` (mà bản thân nó là lớp con của lớp *`ItemScheme`*). `Organisation` có thể vận hành một số *`OrganisationRole`*. Hiện tại có ba vai trò được định danh: `DataProvider`; `DataConsumer`; `MaintenanceAgency`. Các lớp được liên kết với các vai trò này được xác định trong (các) gói liên quan tới chúng. Chú ý rằng vai trò của `DataProvider` và `DataConsumer` cũng bao quát hoạt động của việc sử dụng và cung cấp siêu dữ liệu.

Mô hình cho phép `OrganisationScheme` được điều hướng bởi Tổ chức và *`OrganisationRole`* hoặc một trong hai. Tuy nhiên, trong khi `Organisation` có thể thực hiện nhiều *`OrganisationRole`* thì `Organisation` được khuyến cáo rằng bất kỳ `OrganisationScheme` nào chỉ bao gồm *`OrganisationRole`* (ví dụ một trong `DataProvider`, `DataConsumer`, `MaintenanceAgency`).

Siêu dữ liệu được đính kèm với *`OrganisationRole`* bởi các phương tiện của cơ chế đính kèm siêu dữ

liệu. Cơ chế này được giải thích trong Siêu dữ liệu Tham chiếu (xem điều 7). Điều này có nghĩa là mô hình không quy định siêu dữ liệu cụ thể mà được đính kèm với một `DataProvider` hoặc `MaintenanceAgency`, như thông tin liên hệ, điều này có thể được cung cấp theo hình thức động sử dụng cơ chế đính kèm siêu dữ liệu.

Tập thông tin `Contact` giới hạn có thể được đính kèm ở mức `OrganisationScheme`. Nếu nhiều thông tin liên hệ được yêu cầu thì điều này có thể đạt được thông qua Siêu dữ liệu Tham chiếu.

`MaintenanceAgency` có thể duy trì nhiều *`MaintainableArtefact`*. *`MaintainableArtefact`* là một lớp trừu tượng và các lớp cụ thể chỉ được ra tại lúc bắt đầu khi chúng được mô tả ở đó.

4.8.2.2 Định nghĩa

Lớp	Đặc trưng	Mô tả
`OrganisationScheme`	kế thừa từ *`ItemScheme`*	Tập hợp được duy trì của các tổ chức.
	`contact`	Liên kết với thông tin liên hệ trong lược đồ.
	`/items`	Liên kết với các tổ chức trong lược đồ.
	`/items`	Liên kết với các vai trò của tổ chức trong lược đồ.
`Contact`		Thể hiện vai trò của cá nhân hoặc tổ chức (cá nhân hoặc bộ phận của tổ chức) mà (các) mục thông tin, (các) đối tượng tài liệu và/hoặc cá nhận có thể gửi đến hoặc từ một ngữ cảnh.
	`name`	Thiết kế Thông tin liên hệ cá nhân bằng biểu thức ngôn ngữ.
	`department`	Thiết kế cấu trúc tổ chức bằng một biểu thức ngôn ngữ, trong các công việc liên quan đến thông tin liên hệ cá nhân.
	`role`	Trách nhiệm của cá nhân đối với đối tượng mà cá nhân này là một liên lạc
	`telephone`	Số điện thoại để liên lạc
	`fax`	Số fax để liên lạc
	`email`	Địa chỉ email để liên lạc.
`Organisation`	kế thừa từ *`Item`*	Tổ chức là khuôn khổ duy nhất của cơ quan có thẩm quyền nơi cá nhân làm việc hoặc được thiết kế để hoạt động và đạt được các mục đích.
	`/hierarchy`	Liên kết giữa hai tổ chức trong quan hệ cha/con
	`+role`	Liên kết với vai trò của tổ chức

OrganisationRole	kế thừa từ	Chức năng và hoạt động của tổ chức, trong xử lý bằng thống kê như là tập hợp, việc xử lý và phổ biến.
	+organisation	Liên kết với tổ chức.
MaintenanceAgency	kế thừa từ *OrganisationRole*	Cơ quan có trách nhiệm về duy trì các sản phẩm như phân loại bằng thống kê, các định nghĩa cấu trúc của tập khóa và các định nghĩa cấu trúc siêu dữ liệu.
DataProvider	kế thừa từ *OrganisationRole*	Tổ chức mà sản xuất ra dữ liệu hoặc siêu dữ liệu tham chiếu.
DataConsumer	kế thừa từ *OrganisationRole*	Tổ chức sử dụng dữ liệu như một đầu vào để xử lý thêm.
MaintainableArtefact		Xem điều về định danh, xác định phiên bản và duy trì
	+Maintainer	Liên kết với cơ quan duy trì.

4.9 Liên kết lược đồ mục

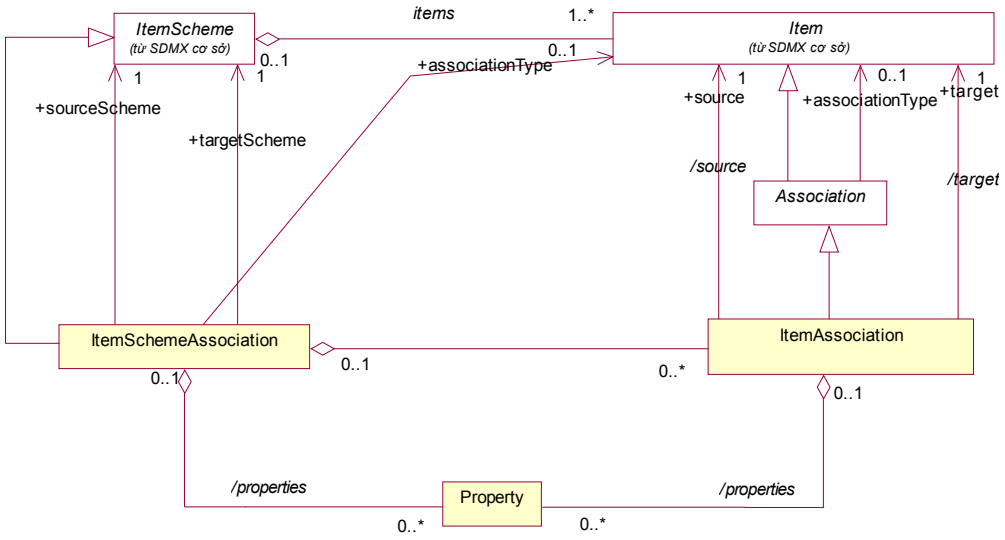
4.9.1 Ngữ cảnh

ItemSchemeAssociation được sử dụng để liên kết các *Item* trong hai *ItemScheme* khác nhau. Đây là cơ chế chung có thể sử dụng để ánh xạ các *Item*. Các mô hình cụ thể tồn tại để ánh xạ các sơ đồ trong đó các *Item* trong *ItemScheme* tương đương với nhau về ngữ nghĩa. Các mô hình hiện có:

- CodeList
- ConceptScheme
- CategoryScheme

Chi tiết thêm xem điều 9 – ÁNH XẠ VÀ TẬP CẤU TRÚC.

4.9.2 Sơ đồ lớp



Hình 21 – Sơ đồ lớp về liên kết lược đồ mục

4.9.3 Giải thích sơ đồ

4.9.3.1 Diễn giải

`ItemSchemeAssociation` kế thừa từ *`ItemScheme`* và `ItemAssociation` kế thừa từ `item` do đó cả hai kế thừa khả năng có được đặc tính liên kết – vì vậy, phải chú ý đến định nghĩa của siêu dữ liệu bổ sung được đính kèm với `ItemSchemeAssociation` và `ItemAssociation`. `AssociationType` xác định vai trò của `ItemSchemeAssociation` và `ItemAssociation`. Chú ý rằng *`Item`* được liên kết bởi `associationType` – nó nằm trong lược đồ cụ thể (danh sách mã) của các vai trò.

4.9.3.2 Định nghĩa

Lớp	Đặc tính	Mô tả
`ItemSchemeAssociation`	kế thừa từ *`ItemScheme`*	Liên kết hai lược đồ mục theo cách được xác định bởi vai trò liên kết.
	`/source`	Liên kết lược đồ mục nguồn.
	`/target`	Liên kết lược đồ mục đích.
	`/items`	Liên kết các liên kết mục mà mỗi liên kết kết nối với một mục nguồn và đích.
	`+associationType`	Đây là một kết nối với mục trong lược đồ mục mà xác định vai trò của liên kết lược đồ mục.
	`/properties`	Liên kết đặc tính với liên kết lược đồ mục.
`ItemAssociation`	kế thừa từ *`Item`*	

	/source	Liên kết mục nguồn
	/target	Liên kết mục đích
	+associationType	Đây là một kết nối với mục trong lược đồ mục mà xác định vai trò của liên kết mục
	/properties	Liên kết đặc tính với liên kết mục

5 Tập khóa (Định nghĩa cấu trúc dữ liệu) và tập dữ liệu

5.1 Giới thiệu

KeyFamily là tên lớp của định nghĩa cấu trúc dữ liệu. Nhiều tổ chức hiểu định nghĩa này là "Định nghĩa cấu trúc dữ liệu" vì vậy hai tên đó là đồng nghĩa với nhau. Thuật ngữ tập khóa được sử dụng trong tiêu chuẩn na.

Nhiều kết cấu trong lớp này của mô hình kế thừa từ lớp SDMX cơ sở. Do đó, cần phải nghiên cứu cả tính kế thừa lẫn các sơ đồ quan hệ để hiểu được chức năng của các gói riêng lẻ. Ở các mô hình con đơn giản, chúng chỉ ra trong cùng một sơ đồ, nhưng được loại bỏ khỏi các lớp con phức tạp để đảm bảo tính rõ ràng. Trong các trường hợp này, sơ đồ dưới đây chỉ ra biểu đồ hình cây về các lớp liên quan đến các định nghĩa cấu trúc dữ liệu.ày

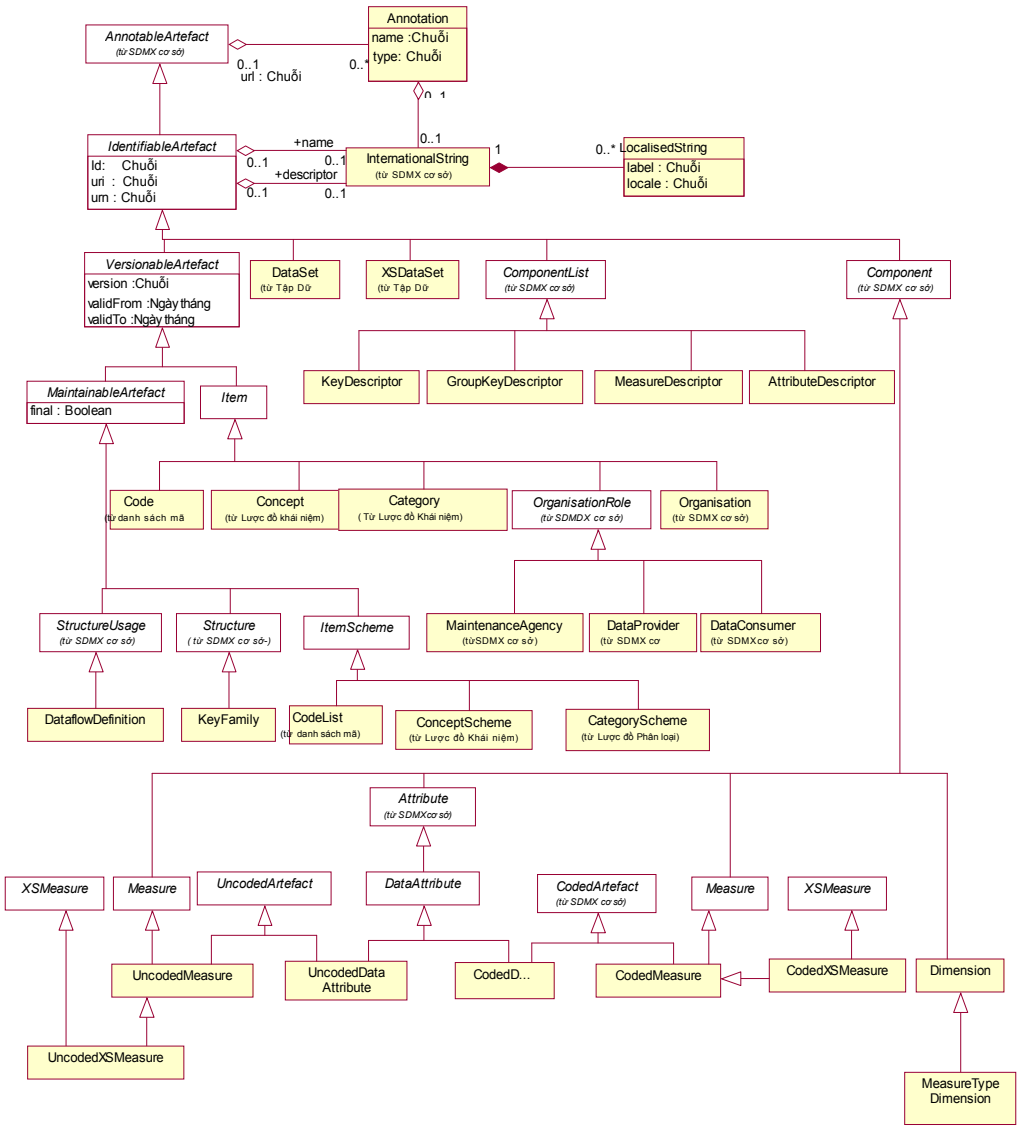
Có rất ít lớp bổ sung trong mô hình con này, thay vì đó chúng được chỉ ra trong sơ đồ kế thừa dưới đây. Nói cách khác, SDMX cơ sở chỉ ra hầu hết cấu trúc của mô hình con này trong cả các thuật ngữ về tính liên kết lẫn các thuật ngữ về thuộc tính. Các sơ đồ quan hệ chỉ ra trong điều này chỉ rõ khi các liên kết này được kế thừa từ SDMX cơ sở (xem Phụ lục "Hướng dẫn ngắn về UML trong Mô hình thông tin SDMX" để xem xét ký pháp bằng sơ đồ sử dụng để mô tả).

Kết cấu thực tế của SDMX cơ sở trong đó các lớp cụ thể kế thừa nhờ các yêu cầu của lớp:

- Chú thích - *AnnotableArtefact*
- Định danh - *IdentifiableArtefact*
- Xác định phiên bản – *VersionableArtefact*
- Duy trì - *MaintainableArtefact*

5.2 Tổng quan về tính kế thừa

5.2.1 Sơ đồ lớp



Hình 22 – Lớp kế thừa trong tập khóa và các gói tập dữ liệu

5.2.2 Giải thích sơ đồ

5.2.2.1 Diễn giải

Các lớp trong siêu mô hình SDMX mà yêu cầu các chú thích kế thừa từ *AnnotableArtefact*. Đó:

- *IdentifiableArtefact*

Các lớp trong siêu mô hình SDMX mà yêu cầu các chú thích, định danh toàn cầu, tên và mô tả ở đa ngôn ngữ được tạo từ *IdentifiableArtefact*. Đó:

- *VersionableArtefact*

Các lớp trong siêu mô hình SDMX mà yêu cầu các chú thích, định danh toàn cầu, tên và mô tả ở đa ngôn

ngữ và xác định bằng phiên bản được tạo từ: *VersionableArtefact*. Đó:

- *MaintainableArtefact*
- *Item*

Các lớp trừu tượng biểu diễn thông tin, được duy trì bởi các cơ quan duy trì, tất cả kế thừa từ *MaintainableArtefact*, chúng cũng kế thừa tất cả các đặc trưng của *VersionableArtefact*, đó:

- *StructureUsage*
- *Structure*
- *ItemScheme*

Tất cả các lớp trên là trừu tượng, điều quan trọng cần hiểu về các lược đồ lớp trong điều này là các lớp cụ thể kế thừa từ các lớp trừu tượng này.

Các lớp cụ thể trong các gói siêu mô hình về tập dữ liệu và tập khóa SDMX, yêu cầu được duy trì bởi các cơ quan duy trì, tất cả kế thừa (qua các lớp trừu tượng khác) từ *MaintainableArtefact*, đó:

- DataflowDefinition
- KeyFamily

Các cấu trúc thành phần là các danh sách của các danh sách, kế thừa trực tiếp từ *Structure*. *Structure* chứa một vài danh sách các phần tử (ví dụ. KeyFamily chứa danh sách các chiều kích thước, danh sách các đo lường và các thuộc tính). Về tập khóa (các định nghĩa cấu trúc dữ liệu) lớp (cấu trúc) cụ thể về các định nghĩa cấu trúc dữ liệu:

- KeyFamily

Các lớp trừu tượng kế thừa từ *ComponentList* và các thành phần phụ của KeyFamily:

- KeyDescriptor
- GroupKeyDescriptor
- MeasureDescriptor
- AttributeDescriptor

Các lớp kế thừa từ *Component* (ví dụ: chúng là các thành phần cụ thể của các lớp trên):

- *Measure*
- Dimension
- *Attribute*

Thuộc tính có thêm một lớp trừu tượng của:

- *DataAttribute*

Các lớp cụ thể mà kế thừa từ các lớp trừu tượng *Measure* và *DataAttribute*:

- CodedMeasure

- UncodedMeasure
- CodedDataAttribute
- UncodedDataAttribute

Hơn nữa, các sản phẩm không được mã hóa (UncodedDataAttribute và UncodedMeasure) kế thừa từ *UncodedArtefact* và các sản phẩm được mã hóa (CodedDataAttribute và CodedMeasure) kế thừa từ *CodedArtefact*. Sự khác nhau giữa *CodedArtefact* và *UncodedArtefact* (được trình bày chi tiết hơn trong đoạn giải thích các cấu trúc cơ sở):

- *CodedArtefact* có một liên kết với *ItemScheme*, trong ngữ cảnh của KeyFamily là lớp CodeList con.
- *UncodedArtefact* không có liên kết này nhưng nó lại có các thuộc tính bổ sung để mô tả kiểu và định dạng của nó.

Các đo lường phần giao là các lớp con của các đo lường chuỗi thời gian, của lớp *XSMeasure* trừu tượng

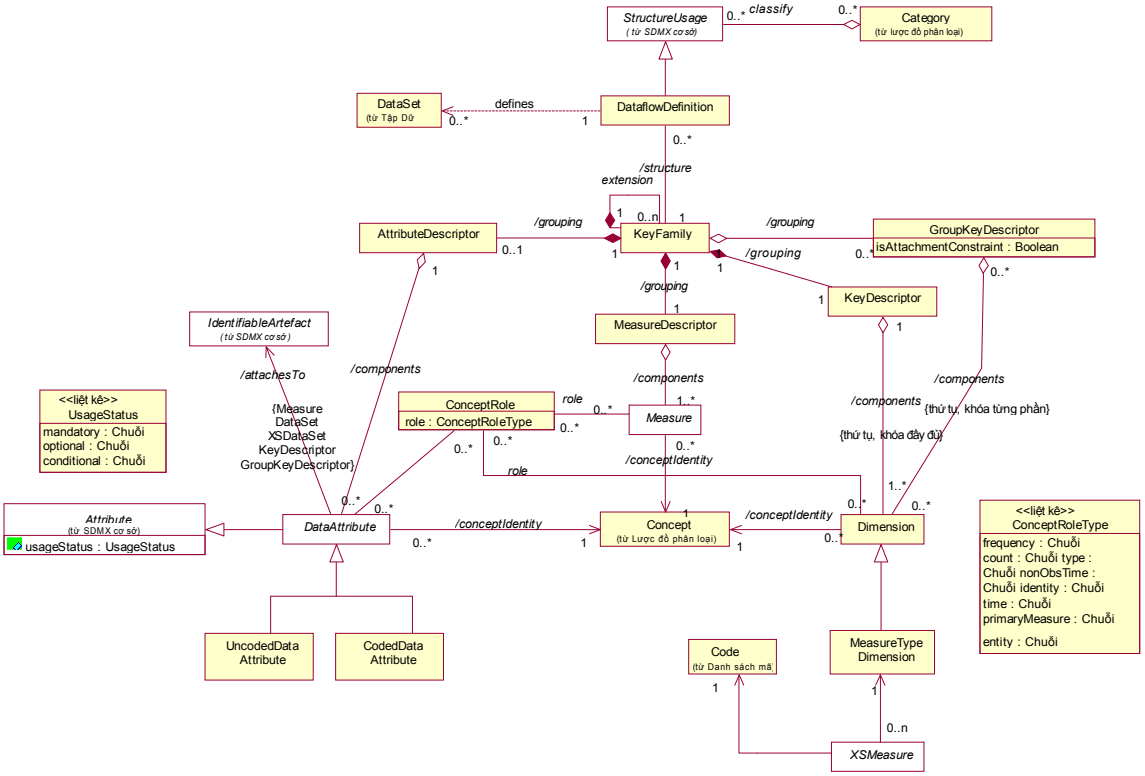
- UncodedXSMeasure kế thừa từ UncodedMeasure và *XSMeasure*
- CodedMeasure kế thừa từ CodedMeasure và *XSMeasure*

Cuối cùng, MeasureTypeDimension là lớp con của Dimension khi nó có các liên kết cụ thể về bản thân Dimension (xem sơ đồ quan hệ bên dưới). Ngoại trừ MeasureTypeDimension vai trò cụ thể được thể hiện bởi các Dimension được hỗ trợ bởi liên kết với vai trò và không được mô tả như các lớp con.

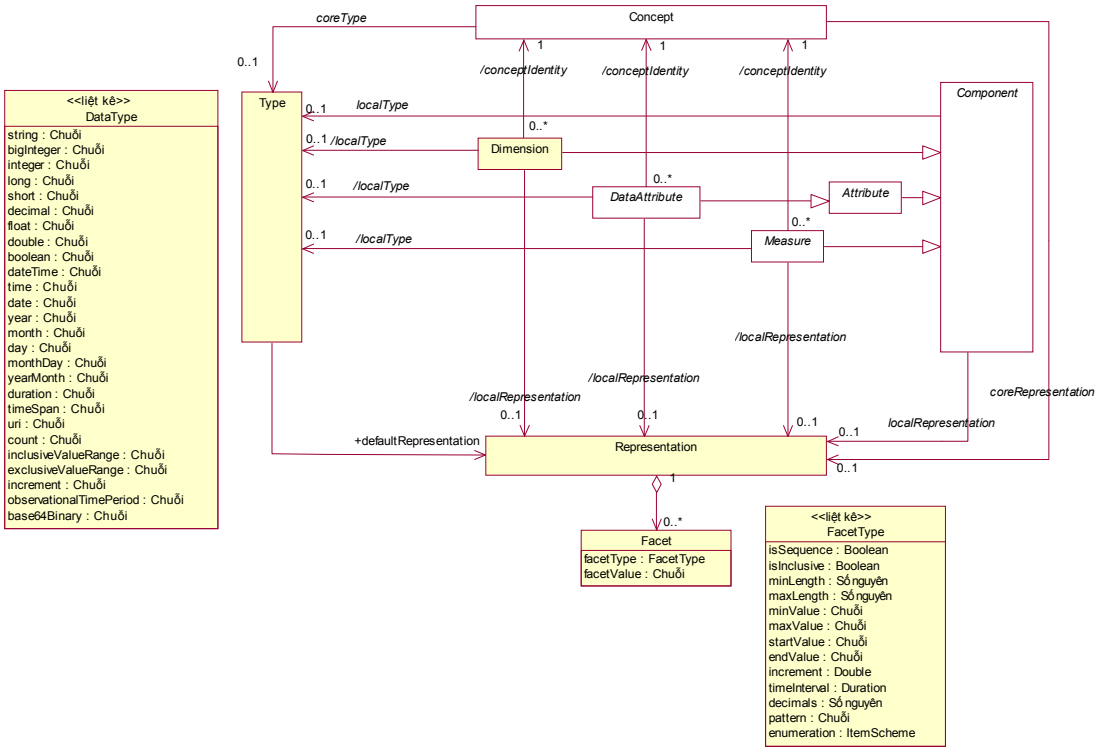
Các lớp cụ thể được định danh ở trên là toàn bộ các lớp được yêu cầu để xác định siêu mô hình về KeyFamily. Các sơ đồ và giải thích trong đoạn còn lại của điều này chỉ ra cách các lớp cụ thể hỗ trợ chức năng yêu cầu.

5.3 Tổng quan về Quan hệ tập khóa

5.3.1 Sơ đồ lớp



Hình 23 – Sơ đồ lớp về quan hệ tập khóa bao gồm biểu diễn



Hình 24 – Sơ đồ lớp về quan hệ của việc biểu diễn tập khóa

5.3.2 Giải thích các sơ đồ

5.3.2.1 Diễn giải

KeyFamily xác định các Dimension, *DataAttribute*, *Measure* và liên kết Representation bao gồm cấu trúc dữ liệu và siêu dữ liệu hợp lệ liên quan có trong DataSet, được xác định bởi DataflowDefinition.

DataflowDefinition liên kết KeyFamily với một hoặc nhiều Category (từ các CategoryScheme khác nhau) thông qua lớp cha của DataflowDefinition – *StructureUsage*. Nó chi ra một hệ thống có khả năng chỉ rõ các DataSet có thể được báo cáo/phổ biến đối với Category cho trước và các DataSet có thể được báo cáo sử dụng định nghĩa KeyFamily. DataflowDefinition cũng có thể có các siêu dữ liệu bổ sung mà xác định thông tin có chất lượng và các ràng buộc khi sử dụng KeyFamily ví dụ như tập con của các Code sử dụng trong một Dimension (nó được nhắc đến sau – xem điều 9 " Sự cung cấp và các ràng buộc về Dữ liệu"). Mỗi DataflowDefinition phải có một KeyFamily xác định cấu trúc của bất kỳ DataSet nào được báo cáo/phổ biến.

Dimension, *DataAttribute* và *Measure* mỗi cái được kết nối với Concept xác định tên và ngữ nghĩa của nó. Các giá trị hợp lệ về một Dimension, Measure hoặc *DataAttribute*, khi được sử dụng trong KeyFamily này, được xác định bởi Representation. Representation được lấy ra từ định nghĩa Concept (coreRepresentation) nếu nó không được ghi đè trong KeyFamily (localRepresentation).

Dimension có thể được nhóm theo hai cách:

1. Sẽ luôn luôn có một nhóm KeyDescriptor định danh tất cả Dimension bao gồm khóa đầy đủ.

2. Có thể tùy chọn nhiều GroupKeyDescriptor mỗi GroupKeyDescriptor định danh nhóm các Dimension mà có thể tạo khóa từng phần. GroupKeyDescriptor phải được định danh (GroupKeyDescriptor.id) và được sử dụng trong GroupKey của DataSet để nhóm các tập khóa tới *DataAttribute* được đính kèm.

Measure là hiện tượng quan sát và một tập các *Measure* trong KeyFamily được nhóm bởi MeasureDescriptor đơn lẻ. Một *Measure* có thể được mã hóa (CodedMeasure) hoặc không mã hóa (UncodedMeasure) – các lớp con cụ thể của Measure này không được chỉ ra trên sơ đồ.

DataAttribute xác định một đặc điểm của dữ liệu mà được tập hợp hoặc phổ biến và được nhóm trong KeyFamily bởi một AttributeDescriptor đơn lẻ. *DataAttribut* có thể được quy định là bắt buộc, điều kiện hoặc tùy chọn (khi xác định trong usageStatus - kế thừa từ lớp *Attribute* cha).

DataAttribute là lớp trừu tượng và còn là một CodedDataAttribute hoặc UncodedDataAttribute.

DataAttribute được quy định "có thể đính kèm với" một phần của cấu trúc KeyFamily. DataAttribute có thể được quy định có thể đính kèm với một tập bắt buộc của các IdentifiableArtefact. Tập bắt buộc đó:

- Measure (Phép đo)

- DataSet (Tập dữ liệu)
- XSDataSet (Tập dữ liệu XS)
- KeyDescriptor (Mô tả khóa)
- GroupKeyDescriptor (Mô tả Khóa mật mã)

Điều này quy định rằng *DataAttribute* được đính kèm với tập con chuỗi khóa hoặc tập con các giá trị mà thành phần có thể mang (ví dụ như Dimension). Điều này được quy định bằng việc công bố trong GroupKeyDescriptor rằng có một AttachmentConstraint (isAttachmentConstraint) quy định tập con này. Id của AttachmentConstraint giống với Id của GroupKeyDescriptor. Các AttachmentContraint được mô tả trong điều 10.3. Nếu có AttachmentConstraint thì GroupKeyDescriptor không quy định bất kỳ Dimension nào, khi chiều kích thước ràng buộc được xác định trong AttachmentConstraint.

Các cấu trúc hợp lệ về định nghĩa KeyFamily trong đó *DataAttribute* được quy định có thể đính kèm và cấu trúc thực tế trong DataSet trong đó AttributeValue được đính kèm:

- DataSet và XSDataSet – AttributeValue đính kèm với DataSet hoặc XSDataSet
- GroupKeyDescriptor(định danh thêm bởi GroupKeyDescriptor.id) AttributeValue đính kèm với GroupKey,Group, Section
- KeyDescriptor – AttributeValue đính kèm với TimeSeriesKey
- *Measure*-AttributeValue đính kèm với Observation hoặc XSObservation

Nếu có yêu cầu gắn siêu dữ liệu với các sản phẩm của KeyFamily khác như Dimension hoặc thậm chí với KeyFamily hoặc tới các lát cắt của khối dữ liệu trong đó không có AttachmentConstraint nào được quy định trong KeyFamily, thì các lớp này được quy định trong định nghĩa cấu trúc siêu dữ liệu, điều này sẽ được giải thích sau.

Các Concept được sử dụng cho mỗi Dimension, *Measure* và *DataAttribute* đóng vai trò cụ thể trong KeyFamily và liên kết với ConceptRole. Các vai trò được quy định trong kiểu dữ liệu ConceptRoleType và mỗi kiểu thành phần được quy định bởi vai trò thể hiện trong bảng dưới đây

Vai trò	Mô tả	Hợp lệ với kiểu thành phần	Vai trò được thực hiện bởi nhiều thành phần
frequency	định danh concept đóng vai trò thường xuyên	DimensionDataAtribute	No
count	định danh concept đóng vai trò của thẻ định danh trong đó thẻ định danh được lấy từ hệ thống đếm	DimensionDataAtribute	Yes
measureType	định danh concept đóng vai trò của việc định danh phép đo	Dimension	Yes

entity	định danh concept đóng vai trò đối tượng cho người tham khảo dữ liệu (ví dụ: đại lý có báo cáo chính, quốc gia có báo cáo thứ cấp)	DimensionDataAtribute	No
time	định danh concept quy định thời gian quan sát của primaryMeasure	Dimension	No
nonObsTime	định danh concept đóng vai trò thẻ định danh ngày tháng trong KeyFamily mà không liên quan tới thời gian quan sát	DimensionDataAtribute	Yes
primaryMeasure	định danh concept đóng vai trò quan sát chuỗi thời gian	Measure	No
identity	định danh concept đóng vai trò thẻ định danh được lấy từ lược đồ định danh đã biếtn	DimensionDataAtribute	Yes

Mỗi Dimension, Measure và *DataAttribute* có thể có một Type và *Representation* quy định (sử dụng các liên kết localType và localRepresentation). Nếu Type *Representation* không được quy định trong định nghĩa KeyFamily thì Type *Representation* được lấy từ liên kết xác định với Concept (các liên kết coreType và coreRepresentation). Sơ đồ lớp ở hình 24 đã mô tả hiệu quả điều đó.

1. Concept có một liên kết với *Representation* (coreRepresentation) và với Kiểu (coreType)

2. *Component* có một liên kết với *Representation*(localRepresentation) và với Kiểu (localType).

3. Dimension, DataAttribute và Measure tất cả kế thừa từ Component do đó kế thừa các liên kết localRepresentation và localType – chỉ ra trong sơ đồ như các liên kết kế thừa (/localRepresentation, /localType)

Định nghĩa của các *Facet* và Type có thể được tìm thấy trong điều 4.4.

MeasureTypeDimension liên kết với CodeList, các Code của MeasureTypeDimension sẽ trở thành các *XSMeasure* trong tập khóa của phần giao và hỗ trợ sự biến đổi của tập dữ liệu phần giao thành chuỗi thời gian và ngược lại: các Concept là các *XSMeasure* trong tập khóa của phần giao là các Code trong CodeList liên kết với MeasureTypeDimension. Mỗi *XSMeasure* có một hướng liên kiết với MeasureTypeDimension và Code. Code này được chứa trong CodeList liên kết với MeasureTypeDimension. Có thể có nhiều hơn một MeasureTypeDimension trong KeyFamily.

Hơn nữa, CodeList được đính kèm với mỗi CodedDataAttribute xác định các đặc điểm của phép đó (ví dụ như đơn vị đo lường) của mỗi *XSMeasure* trong tập dữ liệu phần giao được nối với nhau thành CodeList đơn lẻ để xác định các đặc điểm về đo đạc của *Measure* liên quan trọng chuỗi thời gian tương ứng.

Ví dụ, nếu có ba *XSMeasure* Concept gọi là *Cân nặng(Weight)*, *Giá trị(Value), Âm lượng (Volume)* thì khi biến đổi thành chuỗi thời gian *XSMeasure* Concept trở thành Dimension bổ sung (MeasureTypeDimension) với ba giá trị trong CodeList liên kết (*cân nặng, giá trị, âm lượng*). Measure đơn lẻ trong chuỗi thời gian có thể có *Đơn vị đo lường* CodedAttribute được liên kết với CodeList: CodeLis này phải có tất cả giá trị của ba CodeList sử dụng cho ba *XSMeasure*

Định nghĩa KeyFamily có thể được mở rộng để tạo ra một KeyFamily. Việc mở rộng của KeyFamily được giới hạn cho:

- Bổ sung các Dimension, *DataAttribute* và *Measure*
- Quy định việc bổ sung của các GroupDescriptor
- Thay đổi usageStatus với một *DataAttribute*
- Thay đổi CodeList được sử dụng cho một Dimension hoặc *DataAttribute*
- Thay đổi *DataAttribute* từ CodedDataAttribute thành UncodedDataAttribute hoặc ngược lại

5.3.2.2 Định nghĩa

Lớp	Đặc trưng	Mô tả
StructureUsage		Xem “SDMX Cơ sở”.
	classify	Liên kết với một hoặc nhiều loại trong các lược đồ, xác định việc phân loại dữ liệu theo các thuật ngữ của dữ liệu được báo cáo hoặc phổ biến
Category		Xem “Lược đồ phân loại”.
DataflowDefinition	kế thừa từ *StructureUsage*	Khái niệm trừu tượng(ví dụ: cấu trúc không có dữ liệu) của một luồng dữ liệu mà các nhà cung cấp cung ứng các giai đoạn tham chiếu khác nhau.
	structure	Liên kết định nghĩa luồng dữ liệu với một tập khóa.
KeyFamily		Tập hợp các khái niệm siêu dữ liệu, cấu trúc của chúng và việc sử dụng khi thu thập và phổ biến dữ liệu.
	/grouping	Liên kết với tập các khái niệm siêu dữ liệu có vai trò cấu trúc định danh trong tập khóa.
	classify	Liên kết loại được thực hiện bởi luồng dữ liệu phân loại này.

GroupKeyDescriptor	kế thừa từ *ComponentList*	Tập các khái niệm siêu dữ liệu, xác định một khóa từng phần được tạo từ mô tả khóa trong tập khóa.
	isAttachment Constraint	Quy định xem liệu có Ràng buộc Đính kèm quy định tập con Chiều kích thước, Đo lường hoặc các giá trị Thuộc tính trong đó Thuộc tính có thể được đính kèm.
	/components	Liên kết với thành phần trong tập các thành phần.
KeyDescriptor	kế thừa từ *ComponentList*	Tập các khái niệm siêu dữ liệu kết hợp, phân loại một chuỗi thống kê, ví dụ chuỗi thời gian và giá trị của nó, khi kết hợp, ví dụ như một tập dữ liệu, định danh duy nhất một chuỗi cụ thể.
	/components	Liên kết với một thành phần trong tập các thành phần.
AttributeDescriptor	kế thừa từ componentList	Tập các khái niệm siêu dữ liệu định nghĩa các thuộc tính của tập khóa.
	/components	Liên kết với thành phần trong một tập các thành phần.
MeasureDiscriptor	kế thừa từ *ComponentList*	Tập các khái niệm siêu dữ liệu xác định các đo lường của tập khóa.
	/components	Liên kết với thành phần trong một tập các thành phần
Demension	kế thừa từ *Component* Các lớp con: *MeasureTypeDimension*	Một khái niệm thống kê được sử dụng (cùng với hầu hết các khái niệm thống kê khác) để định danh chuỗi thống kê, như chuỗi thời gian, ví dụ: một khái niệm thống kê bao gồm hoạt động kinh tế nào đó hoặc một khu vực tham chiếu địa lý
	conceptIdentity	Liên kết với khái niệm siêu dữ liệu mà xác định ngữ nghĩa của thành phần
	/localType	Liên kết một Kiểu (kiểu dữ liệu) mà ghi đè bất kỳ kiểu lõi quy định cho khái niệm đó
	/localRepresentation	Liên kết với một biểu diễn mà ghi đè bất kỳ biểu diễn chính quy định cho Khái niệm đó

MeasureTypeDimension	kế thừa từ Dimension	Một khái niệm siêu dữ liệu được sử dụng để đề cập tới và định danh một chiều kích thước trong chuỗi thời gian, xác định các khái niệm về đo lường khi dữ liệu phần giao được biểu diễn trong chuỗi thời gian
DataAttribute	Lớp trừu tượng Các lớp con:	Đặc điểm của một đối tượng hoặc một mục
	/localType	Liên kết một Kiểu (kiểu dữ liệu) ghi đè bất kỳ kiểu lõi quy định cho khái niệm đó)
	/localRepresentation	Liên kết với một biểu diễn mà ghi đè bất kỳ biểu diễn chính nào quy định cho Khái niệm đó
UncodedDataAttribute	kế thừa từ *DataAttribute*	Đặc điểm của một đối tượng hoặc thực thể có một biểu diễn bằng bản tự do.
CodedDataAttribute	Kế thừ từ *DataAttribute*	Đặc điểm của một đối tượng hoặc thực thể lấy giá trị từ danh sách mã
Measure	kế thừa từ *Component* Các lớp con: *CodedMeasure* *UncodedMeasure*	Khái niệm là một hiện tượng được đo ở tập dữ liệu về chuỗi thời gian. Trong một tập dữ liệu thì trường hợp đo này thường được gọi là quan sát
	/localType	Liên kết một Kiểu (kiểu dữ liệu) ghi đè bất kỳ kiểu lõi nào quy định cho khái niệm đó
	/localRepresentation	Liên kết với một biểu diễn mà ghi đè bất kỳ biểu diễn chính quy định nào quy định cho Khái niệm đó
CodedMeasure	kế thừa từ *Measure* Các lớp con: *CodedXSMeasure*	Đo lường về chuỗi thời gian được mã hóa.
UncodedMeasure	kế thừa từ *Measure* Các lớp con: *UncodedXSMeasure*	Đo lường về chuỗi thời gian không được mã hóa
CodedXSMeasure	kế thừa từ CodedMeasure	Đo lường phần giao được mã hóa.

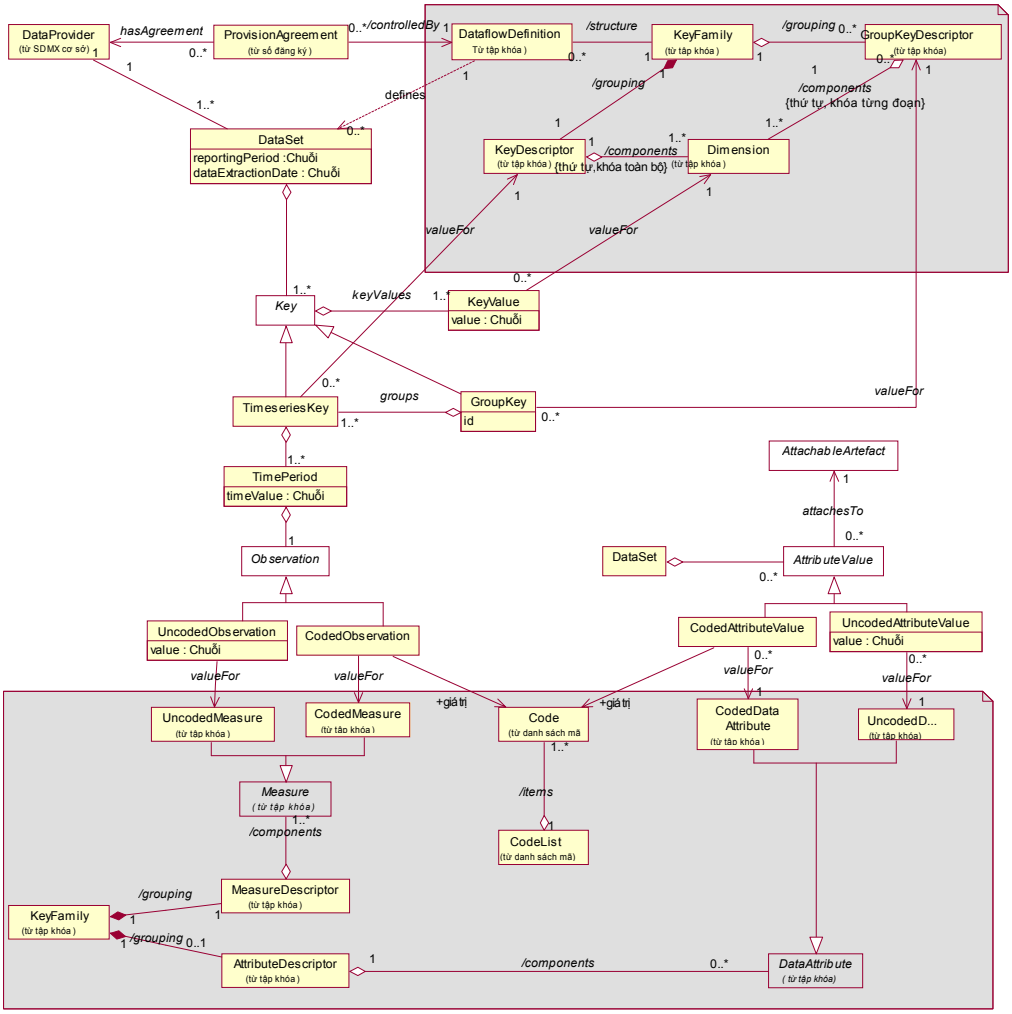
UncodedMeasure	kế thừa từ Measure	Đo lường phần giao không được mã hóa
	XSMeasure	
XSMeasure		Hiện tượng được đo trong tập dữ liệu phần giao
ConceptRole		Quy định vai trò mà một khái niệm thực hiện khi sử dụng trong thành phần cấu trúc, ví dụ như một chiều kích thước trong tập khóa
	role	Định danh vai trò cụ thể

5.4 Quan điểm về quan hệ theo chuỗi thời gian - tập dữ liệu

5.4.1 Ngữ cảnh

Tập dữ liệu bao gồm tập hợp các giá trị dữ liệu và siêu dữ liệu liên kết, được tập hợp hoặc phổ biến theo định nghĩa tập khóa đã biết.

5.4.2 Sơ đồ lớp



Hình 25 – Sơ đồ lớp về tập siêu dữ liệu chuỗi thời gian

5.4.3 Giải thích sơ đồ

5.4.3.1 Diễn giải

Ghi nhớ rằng DataSet phải phù hợp với định nghĩa KeyFamily, liên kết với DataflowDefinition trong đó DataSet là "trường hợp của dữ liệu". Trong khi mô hình biểu diễn việc liên kết với các lớp của KeyFamily, điều này được dùng cho các mục đích về khái niệm để chỉ ra kết nối với KeyFamily. Trong DataSet hiện tại khi trao đổi phải có, tuy nhiên, nó là một tham chiếu với DataflowDefinition, định nghĩa KeyFamily không cần thiết trao đổi với dữ liệu. Do đó, các lớp KeyFamily được biểu diễn trong các vùng xám, khi chúng không phải là một phần của DataSet.

Một tổ chức trong vai trò của DataProvider có thể có trách nhiệm cho một hoặc nhiều DataSet. DataProvider có thể có một DataflowAgreement mà kết nối với DataflowDefinition trong đó DataSet này đang được cung cấp. DataflowAgreement và DataflowDefinition được mô tả sau đó trong đoạn về Việc cung cấp Dữ liệu.

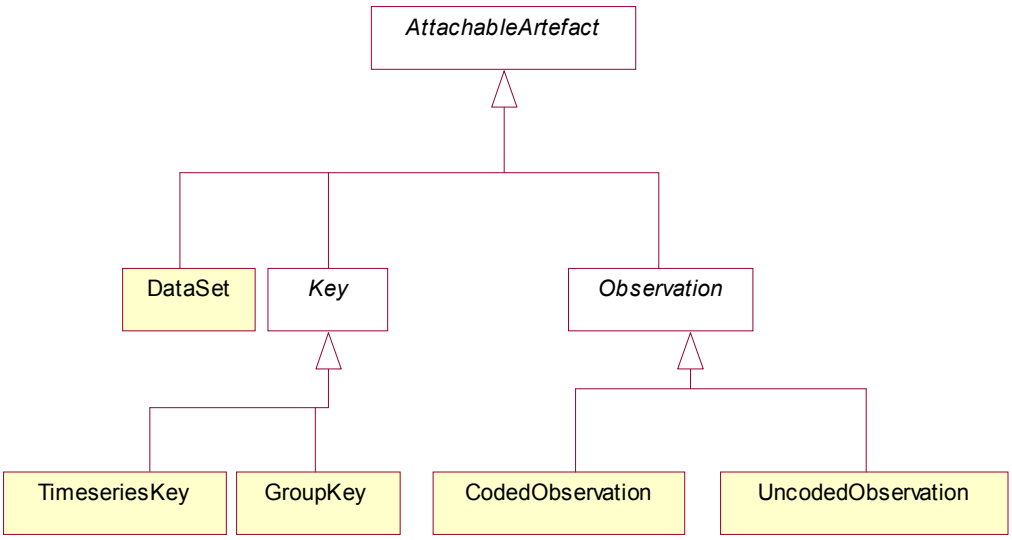
Chuỗi thời gian của DataSet là một tập hợp của các Observation mà chia sẻ cùng một chiều kích thước, được quy định bởi một tập Dimension duy nhất xác định trong KeyDescriptor của KeyFamily, cùng kết hợp với các *AttributeValue* xác định các đặc điểm cụ thể về Observation, Key hoặc DataSet.

Về chuỗi thời gian, mỗi kết hợp duy nhất của KeyValue (TimeseriesKey) kết hợp với một TimePeriod, định danh một Observation.

Observation là một giá trị của biến được đo cho Concept, liên kết với *Measure* trong MeasureDescriptor của KeyFamily. *Observation* có thể liên quan tới CodedMeasure – đây là CodedObservation – hoặc một UncodedMeasure – đây là UncodedObservation.

GroupKey là một đơn vị con của Key có cùng chiều kích thước như TimeseriesKey, nhưng xác định một tập con của các KeyValue của TimeseriesKey. Cấu trúc chiều kích thước phụ của nó được xác định trong GroupKeyDescriptor của KeyFamily định danh bởi cùng một id như GroupKey. Id định danh một "kiểu" nhóm và mục đích của GroupKey là định danh một tập TimeseriesKey riêng để một hoặc nhiều AttributeValue có thể gán với nhóm này. Có thể có nhiều kiểu nhóm trong một DataSet.

Mỗi DataSet, TimeseriesKey, GroupKey và Observation có thể không có hoặc có nhiều AttributeValue mà xác định một vài siêu dữ liệu về đối tượng mà nó liên kết. Các Concept và các đối tượng cho phép mà các siêu dữ liệu này được liên kết, xác định trong KeyFamily. Việc kết nối với đối tượng trong DataSet được biểu diễn bởi sự liên kết với AttachableArtefact. Sơ đồ dưới đây chỉ ra hai kiểu đối tượng mà AttributeValue được đính kèm.



Hình 26 – Giá trị thuộc tính đính kèm với tập dữ liệu theo chuỗi thời gian

Do đó `AttributeValue` kết nối với kiểu đối tượng (`DataSet`, `TimeseriesKey`, `GroupKey`, `CodedObservation`, `UncodedObservation`) và đối tượng hiện tại được định danh bởi khóa của nó (ví dụ. `DataSet`, các `KeyValue` của `TimeseriesKey` hoặc `GroupKey`, hoặc `Observation` (`timeseriesKey` cộng với `TimePeriod`).

5.4.3.2 Định nghĩa

Lớp	Đặc trưng	Mô tả
`DataSet`		Tập hợp dữ liệu có tổ chức.
	`reportingPeriod`	Khoảng thời gian cụ thể trong hệ thống hệ thống khoảng thời gian được biết định danh giai đoạn của một báo cáo.
	`dataExtractionDate`	Khoảng thời gian cụ thể định danh ngày tháng và thời gian mà dữ liệu được trích từ nguồn dữ liệu.
	`describedBy`	Liên kết một định nghĩa luồng dữ liệu và liên quan tới tập khóa của tập dữ liệu

`Key`	Lớp trừu tượng
	Các lớp con
	`TimeseriesKey`
	`GroupKey`
		Bao gồm sản phẩm về giá trị của các chiều kích thước trong đó định danh duy nhất một chuỗi thống kê ví dụ như chuỗi thời gian

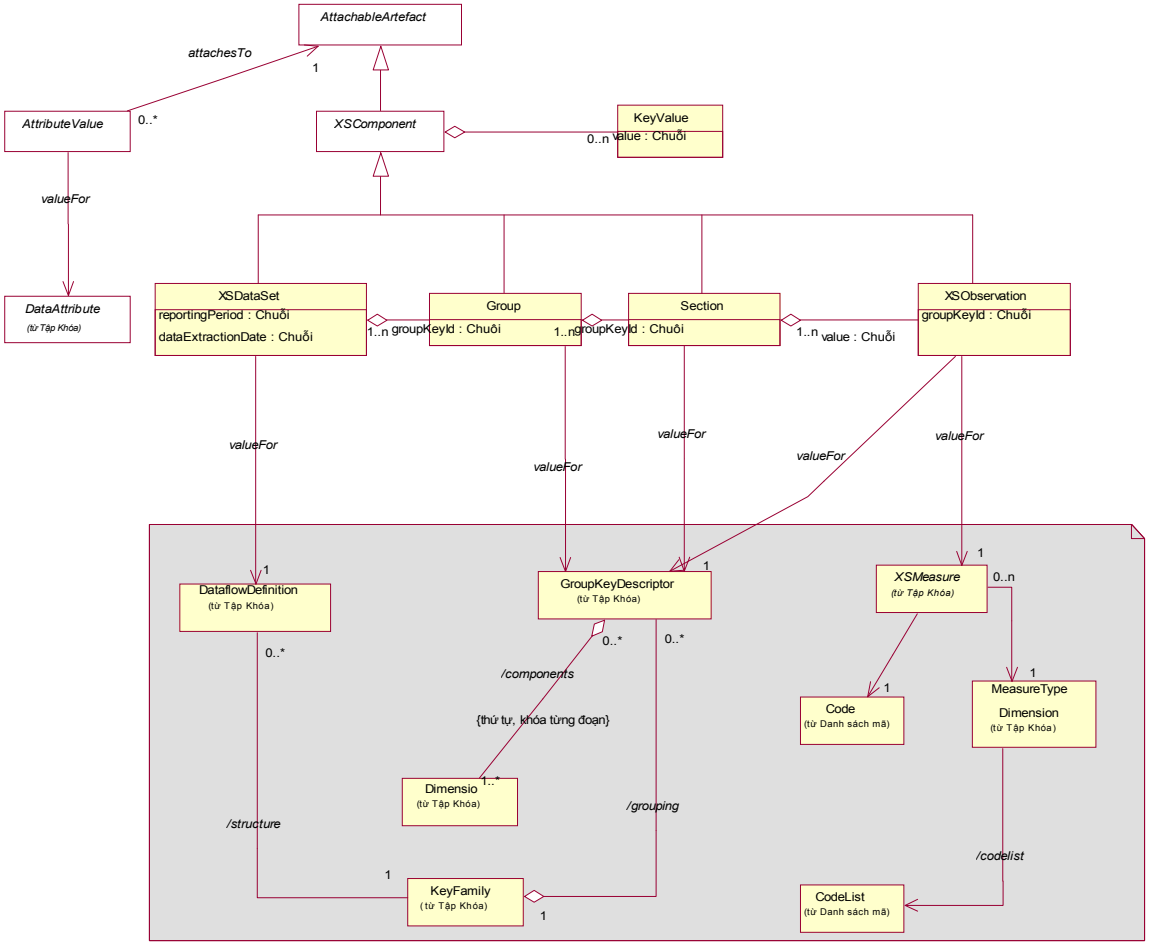
	`keyValues`	Liên kết các giá trị khóa riêng bao gồm khóa.

KeyValue		Giá trị của thành phần khóa ví dụ như giá trị của trường hợp Chiều kích thước trong cấu trúc đa chiều kích thước, giống như việc mô tả khóa trong tập khóa.
	value	Giá trị của thành phần khóa.
	valueFor	Liên kết một chiều kích thước với Giá trị khóa và với Khái niệm là ngữ nghĩa của Chiều kích thước.
GroupKey	kế thừa từ Key	Tập các giá trị khóa bao gồm khóa từng phần, của cùng chiều kích thước như Khóa của Chuỗi Thời gian và cùng nhóm một tập các khóa của chuỗi (ví dụ. phạm vi của các khóa của chuỗi Thời gian được định danh bởi tập khóa xác định bằng cách sử dụng cùng chiều kích thước như Khóa của Chuỗi Thời gian)
	valueFor	Liên kết mô tả nhóm khóa xác định trong tập khóa.
	groups	Liên kết một tập các khóa của Chuỗi Thời gian.
TimeseriesKey	kế thừa từ Key	Bao gồm sản phẩm của các giá trị về các chiều kích thước định danh duy nhất một chuỗi thời gian.
TimePeriod		Một khoảng thời gian trong một hệ thống các gian đoạn thời gian đã biết.
	timeValue	Giá trị của một khoảng thời gian.
Observation	Lớp trừu tượng Các lớp con UncodedObservation CodedObservation	Giá trị ở giai đoạn riêng của một biến riêng.
UncodedObservation	kế thừa từ *Observation*	Quan sát có giá trị nguyên bản.
	value	Giá trị nguyên bản của quan sát.
	valueFor	Liên kết đo lường không được mã hóa xác định trong tập khóa.

CodedObservation	kế thừa từ *Observation*	quan sát lấy giá trị từ một mã trong danh sách mã.
	valueFor	Liên kết đo lường được mã hóa xác định trong tập khóa.
	+value	Liên kết với mã, là giá trị của quan sát.
AttributeValue	Các lớp con của lớp trừu tượng *UncodedAttributeValue* *CodedAttributeValue*	Giá trị của thuộc tính, ví dụ như trường hợp của thuộc tính được mã hóa hoặc của thuộc tính không mã hóa trong cấu trúc như tập khóa.
	attachesTo	Liên kết thuộc tính với đối tượng được đính kèm.
AttachableArtefact		Đối tượng mà giá trị thuộc tính được đính kèm.
UncodedAttributeValue	kế thừa từ *AttributeValue*	Giá trị thuộc tính mà có giá trị nguyên bản.
	value	Giá trị nguyên bản của thuộc tính.
	valueFor	Liên kết thuộc tính dữ liệu được mã hóa xác định trong tập khóa.
CodedAttributeValue	kế thừa từ *AttributeValue*	Thuộc tính mà lấy giá trị từ một mã trong danh sách mã.
	valueFor	Liên kết thuộc tính dữ liệu không được mã hóa xác định trong tập khóa.
	+value	Liên kết với mã, đó là giá trị của quan sát.

5.5 Quan điểm quan hệ của tập dữ liệu phần giao

5.5.1 Sơ đồ lớp



Hình 27 – Sơ đồ lớp về tập dữ liệu phần giao

5.5.2 Giải thích sơ đồ

5.5.2.1 Diễn giải

Tập dữ liệu phần giao - `XSDataSet` – khác với `DataSet` chuỗi thời gian theo các cách sau đây:

1. Không có “khóa đầy đủ” được quy định vì vậy không có khái niệm của “khóa phần giao” khi có khái niệm của một khóa của chuỗi thời gian trong chuỗi thời gian của tập khóa: dữ liệu phần giao được định danh bởi một hoặc nhiều khóa từng phần bao gồm cùng với “khóa đầy đủ”
2. Ý nghĩa của “nhóm” khác với chuỗi thời gian: trong một chuỗi thời gian `GroupKey` nhóm chuỗi thời gian riêng để các thuộc tính chung có thể được đính kèm. Vai trò của `Group` trong tập dữ liệu phần giao: nó mô tả khóa từng phần (phải được kết hợp với các khóa trong các thành phần phụ để định danh đầy đủ việc quan sát); và nó là một cấu trúc mà các thuộc tính có thể được đính

kèm.

3. Các giá trị Dimension (KeyValue) có thể được diễn đạt ở một trong ba mức trong cấu trúc: GroupKey, Section và XSObservation. Do đó, các khóa từng phần có thể được biểu thị ở mỗi mức trong các mức này, cùng tạo ra khóa đầy đủ.
4. Tương tự, các AttributeValue có thể được liên kết ở bất kỳ mức nào trong ba mức, công với mức của bản thân XSDataSet.
5. Nếu giờ hiện hữu trong XSDataSet thì nó được biểu thị ở bất kỳ mức nào của Group.

Chú ý rằng định nghĩa KeyFamily không cần bắt buộc Dimension hoặc Attribute riêng được báo cáo ở mức riêng: nó là bản chất của nhiều chuỗi liên lĩnh vực cho khía cạnh này. Điều kiện tối thiểu trong định nghĩa KeyFamily để hỗ trợ tập dữ liệu miền liên lĩnh vực:

- Biểu thị GroupKeyDescriptor chứa tất cả Dimension
- Tạo ra tất cả MetadataAttribute có thể đính kèm ở mức này.

Rõ ràng, định nghĩa KeyFamily có thể mang tính quy tắc hơn và xác định các nội dung chính xác của mỗi Group, Section và XSObservation bằng cách biểu thị nhiều GroupKeyDescriptor, mỗi GroupKeyDescriptor riêng được định danh bởi GroupKeyDescriptor.id.

Định danh XSObservation lấy từ Code trong CodeList được MeasureTypeDimension sử dụng trong định nghĩa KeyFamily. Có thể có nhiều XSObservation trong một Section, mỗi XSObservation chứa giá trị báo cáo về một trong nhiều Code (chú ý rằng mỗi Code cũng định danh các KeyValue và AttributeValue đã đề cập ở trên).

Liên kết với các kết cấu KeyFamily được biểu diễn bởi các lớp trong hộp mầu xám. Với chuỗi thời gian của DataSet, sẽ có một tham chiếu tới DataFlowDefinition trong XSDataSet.

5.5.2.2 Định nghĩa

Lớp	Đặc trưng	Mô tả
XSComponent	Lớp trừu tượng Các lớp con: DataSet Group	
KeyValue		Giá trị thành phần của một khóa như giá trị của trường hợp một chiều kích thước trong cấu trúc đa chiều kích thước, giống mô tả khóa của tập khóa.
XSDataSet		Tập hợp có tổ chức của dữ liệu miền liên lĩnh vực.
Group	kế thừa từ XSComponent	Tập các giá trị khóa mà bao gồm một khóa từng phần, cùng chiều kích thước như khóa đầy đủ, nhóm cùng với một tập các phần (ví dụ, phạm vi của đoạn này được nhóm lại bởi Nhóm xác định bằng cách sử dụng một tập thành phần các chiều kích thước giống nhau khi xác định trong khóa đầy đủ).
	valueFor	Liên kết mô tả nhóm khóa mà xác định khóa từng phần.
Section	kế thừa từ XSComponent	Tập các giá trị khóa mà bao gồm một khóa từng phần, của cùng chiều kích thước như khóa đầy đủ và cùng nhóm với một tập các quan sát chéo (ví dụ, phạm vi của quan sát XS nhóm bởi đoạn được xác định sử dụng một tập thành phần của các chiều kích thước khác nhau khi xác định trong khóa đầy đủ).
	valueFor	Liên kết GroupKeyDescriptor xác định khóa từng phần.
XSObservation	kế thừa từ XSComponent	Quan sát trong tập dữ liệu phần giao xác định tùy ý một tập các giá trị khóa, của cùng chiều kích thước như khóa đầy đủ.
	valueFor (XSMeasure)	Liên kết XSMeasure (đo lường XS) trong đó đo lường XS là khái niệm của quan sát.
	valueFor (GroupKeyDescriptor)	Liên kết GroupKeyDescriptor (mô tả Khóa mật mã) trong đó mô tả Khóa mật mã xác định khóa từng phần.

6 Khối hộp

6.1 Ngữ cảnh

Một vài hệ thống thống kê hình thành các dạng dữ liệu dựa trên cấu trúc "khối hộp". Thực chất, khối hộp là đối tượng có n chiều kích thước trong đó giá trị mỗi chiều kích thước có thể được tạo từ danh sách mã phân cấp. Tiện ích của các hệ thống khối hộp này có khả năng "mở rộng" hoặc "thu hẹp" mỗi mức của hệ thống phân cấp về chiều kích thước quy định mức yêu cầu đưa ra "dạng "dữ liệu – một vài chiều kích thước có thể được mở rộng, số khác thu hẹp. Các hệ thống này đưa ra một cách nhìn thoáng về dữ liệu, với các giá trị tổng về các vị trí chiều kích thước mở rộng. Ví dụ, các quốc gia riêng có thể mở rộng thành vùng kinh tế như EU hoặc vùng địa lý như Europe, chiều kích thước khác, như "kiểu đường" có thể thu hẹp thành mức thấp hơn. Đo lường kết quả (như "số vụ tai nạn") do đó có thể là tổng giá trị về mỗi quốc gia riêng đối với kiểu đường cụ thể.

Các hệ thống khối hộp này không dựa vào các danh sách mã đơn giản mà dựa vào các tập mã phân cấp (xem điều 8).

6.2 Hỗ trợ khối hộp trong Mô hình thông tin

Dữ liệu báo cáo sử dụng một cấu trúc tập khóa (nơi mà mỗi giá trị về chiều kích thước, nếu mã hóa, sẽ lấy từ một danh sách mã phẳng) có thể được mô tả bởi định nghĩa khối hộp và xử lý bởi các hệ thống nhận biết khối hộp. SDMX-IM cung cấp định nghĩa của các khối hộp này theo cách sau đây:

- `HierachicalCodeScheme` xác định các hệ thống phân cấp (thường phức tạp) của các mã
- `StructureSet`
    - nhóm `KeyFamily` mô tả khối hộp
    - cung cấp cơ chế ánh xạ giữa các mã trong danh sách mã phẳng được sử dụng `KeyFamily` và `HierarchicalCodeScheme`

7 Định nghĩa cấu trúc siêu dữ liệu và tập siêu dữ liệu

7.1 Ngữ cảnh

Siêu mô hình SDMX cho phép siêu dữ liệu:

1. Được trao đổi mà không cần gắn nó vào đối tượng mà nó đang mô tả.
2. Để được lưu trữ riêng biệt với đối tượng mà nó mô tả, được kết nối với nó (ví dụ, một tổ chức có một kho chứa siêu dữ liệu hỗ trợ cho việc phổ biến siêu dữ liệu do các yêu cầu về siêu dữ liệu được tạo ra bởi các hệ thống hoặc dịch vụ dẫn tới đối tượng mà siêu dữ liệu liên quan đến).
3. Được chỉ mục để hỗ trợ việc tìm kiếm (ví dụ: một dịch vụ đăng ký có thể xử lý một báo cáo siêu dữ liệu và trích dẫn thông tin về cấu trúc cho phép nó ghi vào danh mục siêu dữ liệu theo cách đảm báo cho người sử dụng có thể truy vấn về nó).
4. Được báo cáo theo cấu trúc xác định

Để đạt được điều này, các cấu trúc sau đây được mô hình hóa

- Định nghĩa cấu trúc siêu dữ liệu có các thành phần sau đây:
    - Các kiểu đối tượng mà siêu dữ liệu được liên kết (đính kèm)
    - Các linh kiện, bao gồm một khóa duy nhất của kiểu đối tượng
    - Cấu trúc báo cáo bao gồm các thuộc tính siêu dữ liệu có thể được đính kèm với các kiểu đối tượng khác nhau (các thuộc tính này có thể là cấu trúc trong một hệ thống phân cấp), cùng với bất kỳ ràng buộc nào có thể áp dụng (ví dụ. liên kết với danh sách mã chứa các giá trị hợp lệ về thuộc tính khi báo cáo trong tập siêu dữ liệu)
- Tập siêu dữ liệu, chứa siêu dữ liệu báo cáo

7.2 Tính kế thừa

7.2.1 Giới thiệu

Với các định nghĩa cấu trúc, nhiều cấu trúc trong lớp mô hình này kế thừa từ lớp SDMX cơ sở. Do đó, cần nghiên cứu cả tính kế thừa lẫn các sơ đồ quan hệ để hiểu chức năng của các gói riêng lẻ. Sơ đồ dưới đây chỉ ra một sơ đồ hình cây về tính kế thừa đầy đủ về các lớp liên quan tới `MetadataStructureDefinition` và the `MetadataSet`. Sơ đồ không bao gồm các lớp đã mô tả rồi nhưng được sử dụng trong các mô hình siêu dữ liệu tham chiếu (xem 8.3.2).

Có rất ít các lớp bổ sung trong gói `MetadataStructureDefinition` mà không tự kế thừa từ các lớp trong SDMX cơ sở. Nói cách khác, SDMX cơ sở đưa ra hầu hết cấu trúc của mô hình con này kể cả trong các thuật ngữ của các liên kết lẫn các tuật ngữ của các thuộc tính. Các sơ đồ quan hệ được trình bày trong Điều này chỉ rõ khi các liên kết được kế thừa từ SDMX cơ sở (xem phụ lục"Hướng dẫn ngắn gọn về UML trong Mô hình thông tin SDMX" để thấy ký pháp bằng sơ đồ được sử dụng để mô tả điều này. Điều quan trọng cần chú ý là các cấu trúc SDMX cơ sở sử dụng cho `MetadataStructureDefinition` giống với các cấu trúc sử dụng cho `KeyFamily`, thậm chí cách sử dụng không khác nhau là mấy, cách xác định một `MetadataStructureDefinition` là tương tự với cách xác định một `KeyFamily`.

Kết cấu SDMX cơ sở trong đó các lớp cụ thể kế thừa nhờ các yêu cầu của lớp về:

Chú giải – *`AnnotableArtefact`*

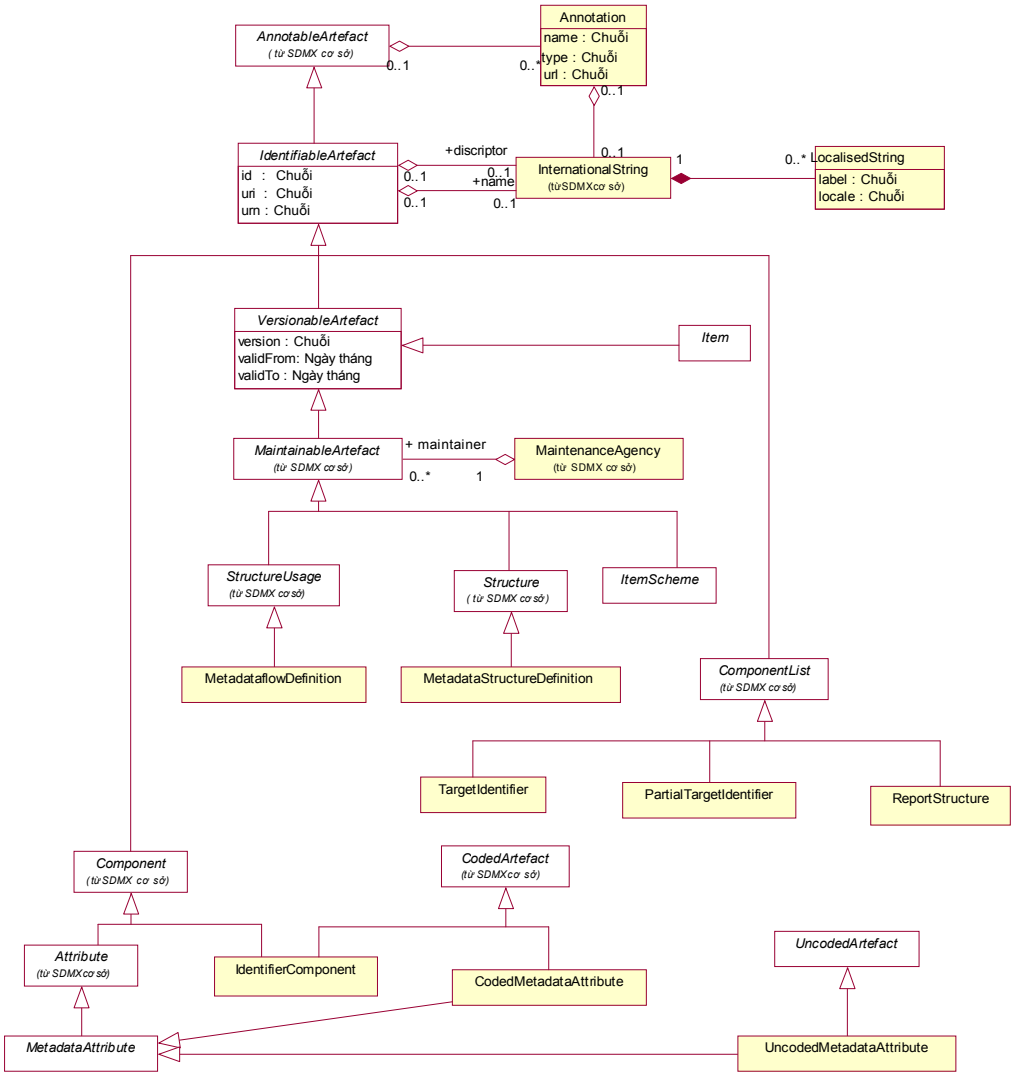
Định danh - *`IdentifiableArtefact`*

Xác định phiên bản – *`VersionableArtefact`*

Duy trì - *`MaintainableArtefact`*

Khả năng có các siêu dữ liệu bổ sung được xác định đính kèm - *`AttachableArtefact`*

7.2.2 Sơ đồ về lớp kế thừa



Hình 28 – Lớp kế thừa trong định nghĩa cấu trúc siêu dữ liệu và các gói Tập Siêu dữ liệu

7.2.3 Giải thích sơ đồ

7.2.3.1 Diễn giải

Điều quan trọng là phải nắm được các lược đồ về lớp quan hệ thể hiện trong điều này để định danh các lớp cụ thể kế thừa từ các lớp trừu tượng.

Các lớp cụ thể trong đoạn này yêu cầu được duy trì bởi các cơ quan duy trì, tất cả kế thừa `MaintainableArtefact`, chúng:

- *`StructureUsage`* (lớp cụ thể là `MetadataflowDefinition`)
- *`Structure`* (lớp cụ thể là `MetadataStructureDefinition`)

Các lớp này cũng kế thừa các khía cạnh về định danh và xác định phiên bản của *`IdentifiableArtefact`* và *`VersionableArtefact`*.

Một *Structure* chứa một vài danh sách các thành phần. Các lớp cụ thể kế thừa từ *ComponentList* và bản thân là các thành phần phụ của MetadataStructureDefinition:

- TargetIdentifier
- PartialTargetIdentifier
- ReportStructure

ComponentList chứa các Component. Các lớp mà kế thừa từ *Component*

- IdentifierComponent
- *MetadataAttribute*

Lớp kế thừa lớp trừu tượng *Attribute* mà liên quan đến siêu dữ liệu tham chiếu và các mô hình tập siêu dữ liệu:

- *MetadataAttribute*

MetadataAttribute là một lớp trừu tượng và có hai lớp cụ thể con:

- CodedMetadataAttribute
- UncodedMetadataAttribute

Thêm vào sự kế thừa từ *MetadataAttribute* CodedMetadataAttribute kế thừa từ *CodedArtefact* và UncodedMetadataAttribute kế thừa từ *UncodedArtefact*.

7.3 Định nghĩa cấu trúc siêu dữ liệu

7.3.1 Giới thiệu

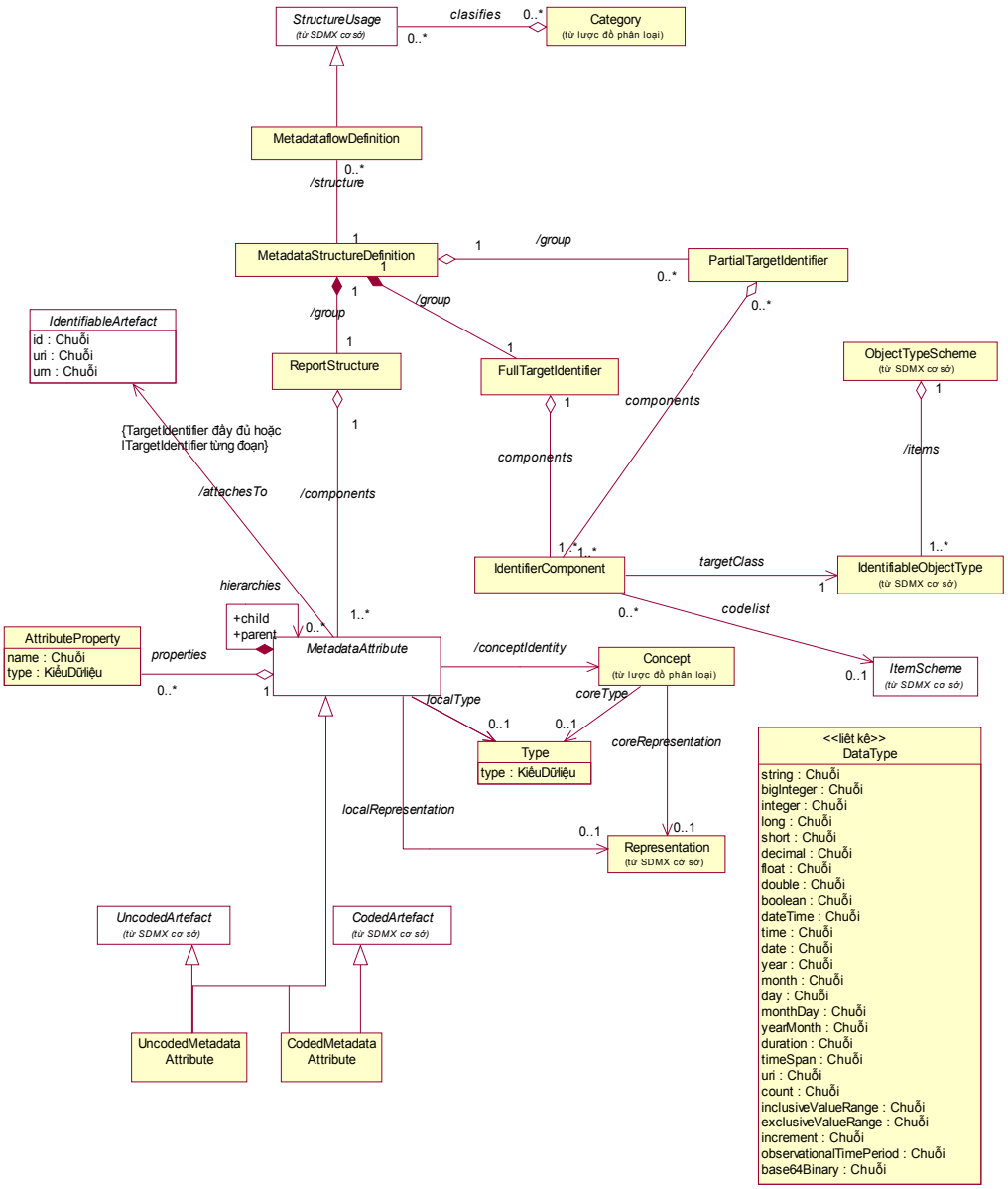
Chỉ với một ngoại lệ, các lớp cụ thể định danh ở trên có thể có nhiều lớp hơn được yêu cầu xác định siêu mô hình về các định nghĩa cấu trúc siêu dữ liệu. Các sơ đồ và giải thích trong đoạn còn lại của điều này chỉ ra cách các lớp cụ thể này hỗ trợ chức năng yêu cầu. Trường hợp ngoại lệ này là AttributeProperty không kế thừa từ bất kỳ lớp SDMX cơ sở nào.

7.3.2 Các cấu trúc đã được mô tả

MetadataStructureDefinition sử dụng các cấu trúc *ItemScheme* khi các lớp cụ thể trong mô hình hoặc các danh sách bao gồm miền giá trị của một IdentifierComponent.

- CategoryScheme
- ConceptScheme
- CodeList
- organisationScheme

7.3.3 Sơ đồ lớp



Hình 29 – Sơ đồ lớp quan hệ của định nghĩa cấu trúc siêu dữ liệu

7.3.4 Giải thích sơ đồ

7.3.4.1 Diễn giải

Nói ngắn gọn, một `MetadataStructureDefinition` xác định:

- Kiểu đối tượng mà siêu dữ liệu có thể được liên kết (`IdentfiableArtefactType`).
- Các thành phần (`IdentifierComponent`)bao gồm thẻ định danh đối tượng của của đối tượng đích (`FullTargetidentifier` và `PartialTargetIdentifier`).
- `ReportStructure` bao gồm các *`MetadataAttribute`* mà có thể được liên kết với kiểu đối

tượng và cấu trúc phân cấp của các thuộc tính.

FullTargetIdentifier bao gồm nhiều IdentifierComponent, bao gồm phạm vi của MetadataStructureDefinition trong các thuật ngữ kiểu đối tượng có thể được định danh bằng cách sử dụng định nghĩa này. Mỗi IdentifierComponent phải được liên kết với một IdentifiableArtefactType mà bản thân lấy từ lược đồ duy trì của các ObjectType. Trong ngữ cảnh của mô hình thông tin này các ObjectType sẽ là bất kỳ lớp hoặc nhóm các lớp (được xác định bởi các IdentifierComponent) trong mô hình có sự định danh, nó là các trường hợp của các kiểu đối tượng hoặc các nhóm của chúng mà siêu dữ liệu có thể được đính kèm trong một MetadataSet.

Các trường hợp của IdentifierComponent (ví dụ. IdentifierComponentValue thực tế xác định trong MetadataSet) được duy trì trong *ItemScheme* (hoặc, chính xác hơn, một sản phẩm cụ thể được tạo từ ItemScheme ví dụ như CodeList, ConceptScheme, CategoryScheme hoặc OrganisationScheme). Ví dụ nếu targetClass của IdentifierComponent là DataProvider thì sự chuyên môn hóa của (ví dụ. kiểu của) *ItemScheme* phải là OrganisationScheme chứa danh sách các DataProvider. Thông thường, *ItemScheme* này có thể được quy định trong MetadataStructureDefinition. Tuy nhiên, có các trường hợp trong đó điều này không có khả năng xảy ra. Một ví dụ về nó trong đó IdentifierComponent là Dimension trong KeyFamily – khi các Dimension riêng có thể sử dụng các Concept từ các ConceptScheme khác nhau, điều đó cần thiết cho ứng dụng đọc định nghĩa KeyFamily để làm cho nó hợp lê, Concept chính xác được tham chiếu trong IdentifierComponentValue của MetadataSet.

PartialTargetIdentifier định danh một tập con các IdentifierComponent của FullTargetIdentifier. Mục tiêu ở đây là đảm bảo rằng MetadataStructureDefinition đơn có thể được xác định cho một tập các kiểu đối tượng cụ thể liên quan: do đó, ví dụ, định nghĩa đơn có thể được xây dựng để xác định siêu dữ liệu mà có thể được đính kèm với bất kỳ phần nào của tập khóa hoặc điều đó có thể được đính kèm với bất kỳ sản phẩm liêu quan tới việc báo cáo chất lượng siêu dữ liệu (ví dụ như nhà cung cấp dữ liệu và loại dữ liệu). FullTargetIdentifier định danh tất cả các kiểu đối tượng liên quan có trong phạm vi định nghĩa, trong khi đó PartialTargetIdentifier định danh tập con của các dữ liệu này, chúng tạo nên "khóa" của targetClass về PartialTargetIdentifier. Ví dụ, trong một tập khóa targetClass có thể là một chiều kích thước, do đó chúng là các IdentifierComponent mà định danh duy nhất một chiều kích thước (ngẫu nhiên, là tập khóa và khái niệm).

Cấu trúc Báo cáo bao gồm một tập các *MetadataAttribute* được xác định như một hệ thống phân cấp. Mỗi *MetadataAttribute* định danh Concept được báo cáo hoặc phổ biến trong MetadataSet sử dụng MetadataStructureDefinition này. Concept phải là Concept hợp lệ duy trì trong một ConceptScheme. Nó không bắt buộc tất cả *MetadataAttribute* được kết nối với các Concept từ ConceptScheme.

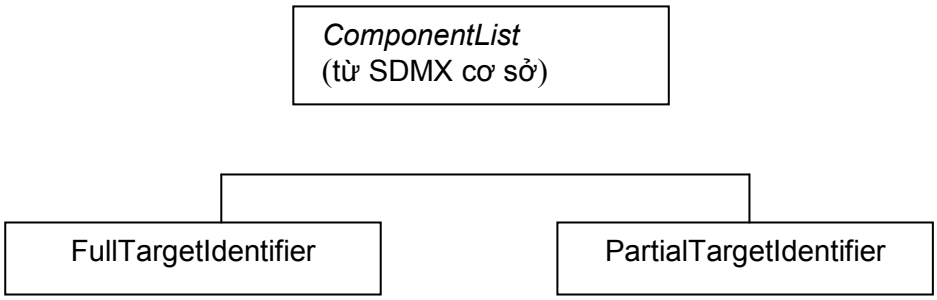
MetadataAttribute có thể được quy định là bắt buộc, có điều kiện hoặc tùy chọn (assignmentStatus – kế thừa từ *Attribute*).

MetadataAttribute là lớp trừu tượng và còn là CodedMetadataAttribute hoặc

UncodedMetadataAttribute. CodedMetadataAttribute được liên kết với CodeList chứa tập các giá trị hợp lệ có thể được báo cáo về CodedMetadataAttribute trong MetadataSet.

Nó có khả năng xác định cấu trúc phụ về *MetadataAttribute* bằng việc sử dụng AttributeProperty. AttributeProperty cho phép *MetadataAttribute* có một văn bản có thể định danh (như là URL). Tuy nhiên, không có hỗ trợ về trình tự và các ứng dụng để biết cách hợp nhất giá trị của thuộc tính đã gửi trong MetadataSet với bất kỳ giá trị đã gửi trong tổ chức của UncodedMetadataAttribute hoặc CodedMetadataAttribute.

MetadataAttribute có thể được quy định có thể đính kèm với một hoặc nhiều *IdentifiableArtefact*. Sơ đồ dưới đây chỉ ra các lớp kế thừa từ *IdentifiableArtefact* trong ngữ cảnh của siêu dữ liệu tham chiếu.



Hình 30 – Định nghĩa đính kèm thuộc tính siêu dữ liệu

Có thể thấy rằng đặc tả về các kiểu đối tượng trong đó *MetadataAttribute* có thể được đính kèm là gián tiếp: *MetadataAttribute* được đính kèm với một hoặc nhiều FullTargetIdentifier hoặc PartialTargetIdentifier và đối tượng thực tế được định danh bởi targetClass ở đó hoặc PartialTargetIdentifier FullTargetIdentifier được liên kết. Điều này đưa ra một cơ chế linh hoạt trong đó các kiểu đối tượng thực tế không cần được xác định trong MetadataStructureDefinition, trong cùng một cách như các khóa trong đó việc quan sát dữ liệu được "đính kèm" trong một định nghĩa KeyFamily. Theo cách này MetadataStructureDefinition có thể được sử dụng để xác định tập các MetadataAttribute và tập đối tượng chúng được đính kèm.

MetadataAttribute có thể có một Type và *Representation* được quy định (sử dụng các liên kết localType và localRepresentation). Nếu không được quy định trong MetadataStructureDefinition thì Type và *Representation* được lấy từ lớp xác định về Concept (các liên kết coreType và coreRepresentation).

Định nghĩa các kiểu khác nhau của *Representation* và Type có thể được tìm thấy trong điều 4.4.

MetadataStructureDefinition được kết nối với một MetadataflowDefinition. MetadataflowDefinition không có bất kỳ thuộc tính cụ thể nào nhưng lại có siêu dữ liệu bổ sung đính kèm sử dụng bản thân có chế siêu dữ liệu tham chiếu.

Thực tế, MetadataflowDefinition liên kết MetadataStructureDefinition với một hoặc nhiều Category (có thể từ cácCategoryScheme khác nhau). Điều này đưa ra một hệ thống có khả năng chỉ

rõ các MetadatataSet được báo cáo/phổ biến về Category cho trước và các MetadataSet có thể được báo cáo bằng cách sử dụng MetadataStructureDefinition.

7.3.4.2 Định nghĩa

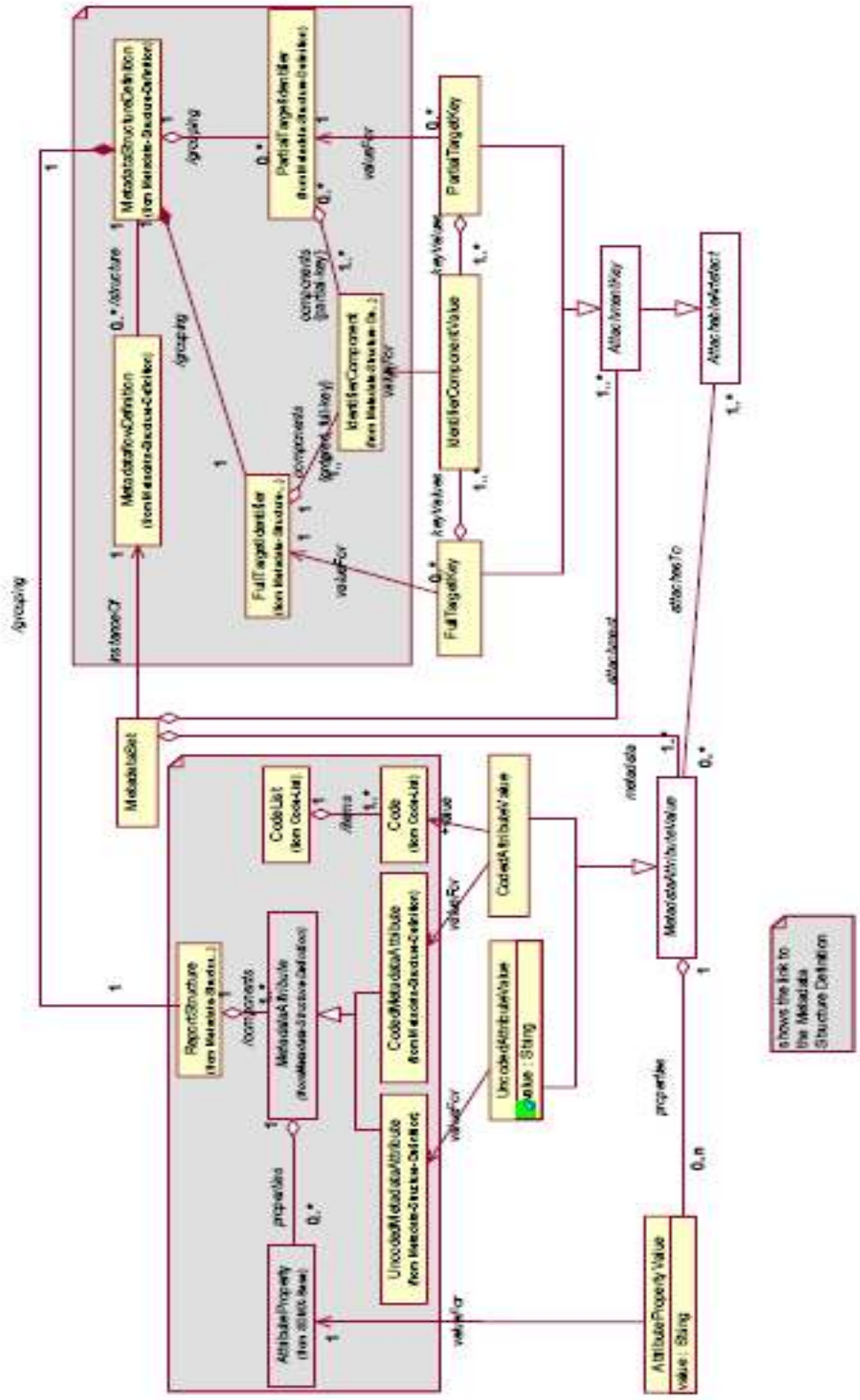
Lớp	Đặc trưng	Mô tả
StructureUsage		Xem "SDMX cơ sở".
	classify	Liên kết một hoặc nhiều loại trong các lược đồ trong đó chúng định việc phân loại siêu dữ liệu, dữ liệu trong các thuật ngữ của siêu dữ liệu được báo cáo hoặc phổ biến.
Category		Xem "Lược đồ phân loại".
Metadataflow Definition	kế thừa từ *StructureUsage*	Khái niệm trừu tượng (ví dụ: cấu trúc không có siêu dữ liệu) của dòng siêu dữ liệu mà các nhà cung cấp cung cấp các giai đoạn tham chiếu khác nhau.
	/structure	Liên kết một định nghĩa cấu trúc siêu dữ liệu.
MetadataStructure Definition		Tập hợp các khái niệm siêu dữ liệu, cấu trúc và cách sử dụng của nó khi được sử dụng để tập hợp hoặc phổ biến siêu dữ liệu tham chiếu.
	/grouping	Liên kết với tập các khái niệm siêu dữ liệu mà có một vai trò cấu trúc được định danh trong định nghĩa cấu trúc siêu dữ liệu.
FullTarget Identifier	kế thừa từ ComponentList	Tập các thành phần xác định khóa của một kiểu đối tượng mà siêu dữ liệu có thể được đính kèm.
	/components	Liên kết các thành phần thẻ định danh trong đó nó xác định một khóa.
	targetClass	Liên kết với kiểu đối tượng có thể được định danh mà thẻ định danh Đích định danh.
PartialTarget Identifier	kế thừa từ ComponentList	Tập các thành phần mà xác định khóa của kiểu đối tượng mà siêu dữ liệu có thể được đính kèm và là một khóa từng phần của đối tượng định danh trong thẻ định danh đích đầy đủ.
	/components	Liên kết các thành phần thẻ định danh mà xác định khóa từng phần.

	targetClass	Liên kết với kiểu đối tượng có thể định danh mà thẻ định danh đích từng phần định danh.
IdentifierComponent		Khái niệm được sử dụng đề cập và định danh một phần của thẻ định danh trong một định nghĩa cấu trúc siêu dữ liệu.
	targetClass	Liên kết với kiểu đối tượng có thể định danh mà thẻ định danh từng phần định danh
	codelist	Liên kết lược đồ mục, ví dụ như: Danh sách mã, lược đồ khái niệm và lược đồ phân loại.
IdentifierComponent		Khái niệm được sử dụng đề cập và định danh một phần của thẻ định danh trong một định nghĩa cấu trúc siêu dữ liệu.
	targetClass	Liên kết với kiểu đối tượng có thể định danh mà thẻ định danh từng phần định danh
	codelist	Liên kết lược đồ mục, ví dụ như: Danh sách mã, lược đồ khái niệm và lược đồ phân loại.
ItemScheme	Các lớp con: CodeList ConceptScheme CategoryScheme OrganisationScheme	Danh sách các giá trị xác định miền giá trị của Thành phần Thẻ định danh.
ConceptDescriptor	kế thừa từ *ComponentList*	Tập các khái niệm siêu dữ liệu xác định các thuộc tính siêu dữ liệu của định nghĩa cấu trúc siêu dữ liệu.
	/components	Liên kết với các thuộc tính siêu dữ liệu liên quan tới Định nghĩa cấu trúc siêu dữ liệu.
MetadataAttribute	Lớp trừu tượng Các lớp con: *CodedMetadataAttribute* *UncodedMetadataAttribut*	Giá trị của một thuộc tính, như là trường hợp của thuộc tính được mã hóa hoặc không được mã hóa trong một định nghĩa cấu trúc siêu dữ liệu.
	/conceptIdentity	Liên kết với khái niệm.
	properties	Cho phép một hoặc nhiều đặc tính Thuộc tính mà xác định ngữ nghĩa của thuộc tính.

	/localType	Liên kết kiểu (kiểu dữ liệu) ghi đè mọi kiểu chính quy định về khái niệm đó.
	/localRepresentation	Liên kết một biểu diễn (kiểu dữ liệu) mà ghi đè bất kỳ Sự biểu diễn chính quy định về khái niệm đó.
Concept	kế thừa từ Item	Khái niệm siêu dữ liệu mà xác định ngữ nghĩa của Thuộc thính Siêu dữ liệu trong định nghĩa cấu trúc siêu dữ liệu
AttributeProperty		Đặc tính cụ thể của cấu trúc được định danh vởi tên và kiểu của nó.
	name	Tên của đặc tính thuộc tính
	type	Quy định kiểu dữ liệu về đặc tính thuộc tính. Kiểu là một danh sách được liệt kê trong liệt kê kiểu dữ liệu.
Identifiable *Artefact*		Quy định các sản phẩm mà thuộc tính siêu dữ liệu có thể được đính kèm. Điều này là bắt buộc cho thẻ định danh đích đầy đủ hoặc thẻ định danh đích từng
CodedMetadata Attribute	kế thừa từ *MetadataAttribute* *CodedArtefact*	Thuộc tính siêu dữ liệu mà lấy các giá trị của nó từ một danh sách mã.
	/codelist	Liên kết một danh sách.
UncodedAttribute	kế thừa từ *MetadataAttribute* *UncodedArtefact*	Thuộc tính siêu dữ liệu mà nội dung của nó không được mã hóa.

7.4 Tập siêu dữ liệu

7.4.1 Sơ đồ lớp



Hình 31 – Tập siêu dữ liệu

7.4.2 Giải thích sơ đồ

7.4.2.1 Diễn giải

Các lớp trong các hộp đậm mầu ở sơ đồ lớp bao gồm các lớp trong MetadataStructureDefinition. Các lớp chứa trong sơ đồ này chỉ ra lien kết giữa các nội dung của MetadataSet và các cấu trúc trong MetadataStructureDefinition. Nhờ vào việc các cấu trúc thực hiện, nó có thể bao gồm một tham chiếu đến MetadataflowDefinition trong trường hợp MetadataSet (khi MetadataflowDefinition chỉ sử dụng một MetadataStructureDefinition).

MetadataSet bao gồm một tập các MetadataAttributeValue đưa ra ý nghĩa bổ sung tới đối tượng được định danh bởi FullTargetKey hay PartialTargetKey. Cấu trúc thành phần của khóa được quy định trong FullTargetIdentifier hoặc PartialTargetIdentifier được xác định ở MetadataStructureDefinition.

Tập IdentifierComponentValue về TargetIdentifier được xác định trong TargetKey và về PartialTargetIdentifier chúng được xác định trong PartialTargetKey.

MetadataSet chứa các *MetadataAttributeValue*, mỗi cái được liên kết với (đính kèm với) một each *AttachableArtefact*. *AttachmentKey* là một sự chuyên môn hóa của *AttachableArtefact*, như các lớp cụ thể, FullargetKey và PatialTargetKey. Do đó một *MetadataAttributeValue* có thể được đính kèm với một hoặc cả hai FullTargetKey và PartialTargetKey. Giá trị nguyên bản đơn giản về thuộc tính sử dụng UncodedAttributeValue, lớp con của *MetadataAttributeValue* trong khi đó một giá trị được mã hóa sử dụng lớp con CodedAttributeValue.

Siêu dữ liệu báo cáo về một *MetadataAttributeValue* có thể có một hoặc nhiều AttributePropertyValuee, nếu AttributeProperty được quy định như đã cho phép về *MetadataAttribute* trong MetadataStructureDefinition.

7.4.2.2 Định nghĩa

Lớp	Đặc trưng	Mô tả
MetadataSet		Mọi tập hợp siêu dữ liệu được tổ chức.
	effectiveDate	Ngày tháng có hiệu lực của toàn bộ siêu dữ liệu.
	instanceOf	Liên kết định nghĩa siêu dữ liệu mà tập siêu dữ liệu này là một ví dụ.
	attachmentKey	Liên kết các khóa đối tượng mà siêu dữ liệu được đính kèm.

	metadata	Liên kết các giá trị thuộc tính siêu dữ liệu được liên kết với đối tượng hoặc các đối tượng được định danh bởi một khóa.
AttachableArtefact	Lớp trừu tượng Lớp con: AttachmentKey	Kết nối với đối tượng mà các siêu dữ liệu được đính kèm.
AttachmentKey	Lớp trừu tượng Các lớp con: TargetKey PartialTargetKey	Định danh khóa của đối tượng mà siêu dữ liệu được đính kèm.
FullTargetKey	kế thừa từ AttachmentKey	Khóa của một đối tượng của kiểu riêng quy định trong thẻ định danh đích đầy đủ của định nghĩa cấu trúc siêu dữ liệu.
	keyValues	Liên kết các giá trị thành phần thẻ định danh của thẻ định danh đích.
	valueFor	Liên kết thẻ định danh đích trong đó thẻ định danh đích này định danh kiểu đối tượng và cấu trúc thành phần của khóa.
PartialTargetKey	kế thừa từ AttachmentKey	Khóa của một đối tượng của kiểu riêng quy định trong thẻ định danh đích từng phần của định nghĩa cấu trúc siêu dữ liệu.
	valueFor	Liên kết Thẻ định danh đích từng phần mà định danh kiểu đối tượng và cấu trúc thành phần của khóa đích từng phần.
	keyValues	Liên kết các giá trị của thành phần thẻ định danh của thẻ định danh đích.
IdentifierComponent Value		Giá trị một thành phần riêng của thẻ định danh đích hoặc Thẻ định danh đích từng phần. Sự móc nối của các giá trị thẻ định danh bao gồm khóa của một đối tượng riêng.
MetadataAttribute Value	Lớp trừu tượng Các lớp con: *UncodedAttributeValue* *CodedAttributeValue*	Giá trị của một thuộc tính siêu dữ liệu
	valueFor	Liên kết với thuộc tính siêu dữ liệu trong định nghĩa cấu trúc siêu dữ liệu mà định danh khái niệm, danh sách mã, các đặc tính và kiểu dữ liệu của thuộc tính.
	properties	Liên kết với một hoặc nhiều giá trị thuộc tính.

	attachesTo	Liên kết với sản phẩm có thể đính kèm (ví dụ. Khóa đích và khóa đích từng phần) trong đó giá trị thuộc tính siêu dữ liệu liên quan đến.
AttributeProperty Value		Giá trị của đặc tính đưa ra siêu dữ liệu bổ sung cho giá trị thuộc tính siêu dữ liệu.
	value	Nội dung của siêu dữ liệu đặc tính.
	valueFor	Liên kết với đặc tính về thuộc tính siêu dữ liệu trong định nghĩa cấu trúc siêu dữ liệu mà định danh tên và kiểu của giá trị đặc tính.
UncodedAttribute Value	kế thừa từ *MetadataAttributeValue* Lớp con: *XMLAttributeValue*	Nội dung văn bản của thuộc tính.
CodedAttributeValue	Inherits from *MetadataAttributeValue*	Nội dung được mã hóa của thuộc tính.
	+value	Liên kết với mã trong danh sách mã, trong đó mã là giá trị của thuộc tính

8 Lược đồ mã phân cấp

8.1 Phạm vi

CodeList được mô tả trong điều này là các định nghĩa cấu trúc hỗ trợ một hệ thống phân cấp đơn giản về các Code và giới hạn Code con chỉ có một Code cha. Cấu trúc này có lợi cho việc hỗ trợ các yêu cầu cần thiết của KeyFamily và MetadataStructureDefinition, nó không đủ để cung cấp nhiều liên kết phức tạp giữa các mã được tìm thấy trong các lược đồ mã như đồ phân loại. Thông thường, CodeList sử dụng trong KeyFamily được tạo từ một lược đồ mã phức tạp hơn. Liên kết với lược đồ mã này có thể hỗ trợ các ứng dụng, như các ứng dụng OLAP, đưa ra nhiều dạng dữ liệu hơn là với CodeList đơn giản sử dụng trong KeyFamily.

Chú ý rằng một danh sách mã phân cấp không cần thiết phải là sơ đồ cây. Một sơ đồ cây trong đó các mức được xác định trước và cố định, (ví dụ, một mức luôn luôn có cùng tập mã và bất kỳ mã nào cũng có một quan hệ cha và con cố định với các mã khác). Một phân loại thống kê là ví dụ của một sơ đồ cây và hỗ trợ cho một hệ thống phân cấp là một tập con và trường hợp đặc biệt của danh sách mã phân cấp.

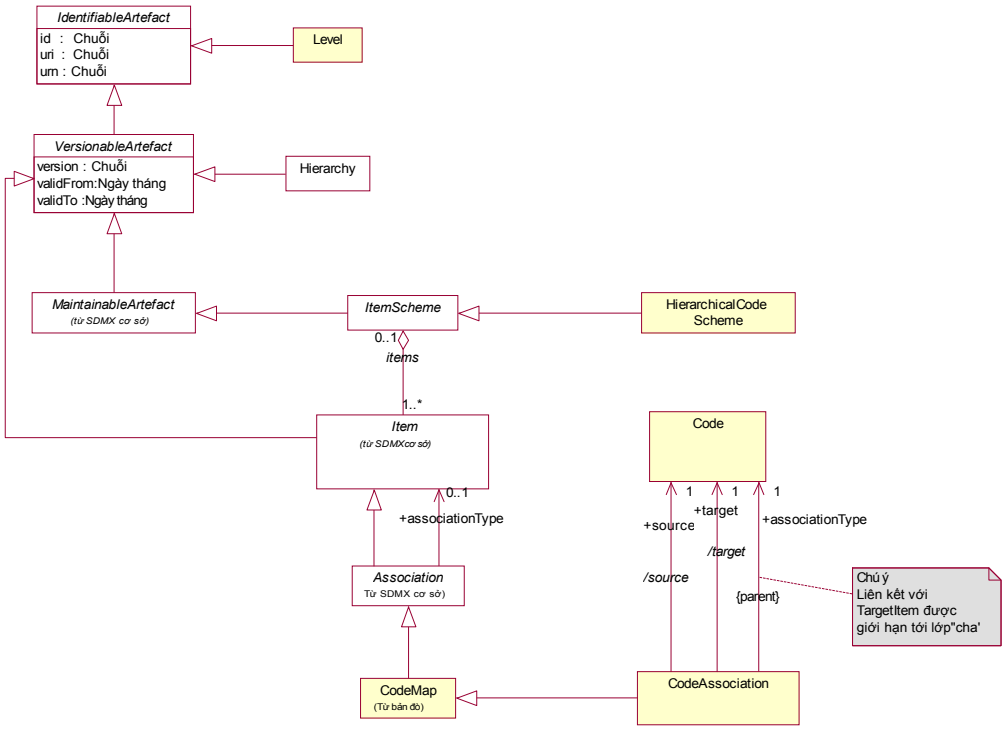
Một số đặc trưng của lược đồ Mã Phân cấp:

1. Mã con có có thể có nhiều hơn một mã cha.
2. Có thể có nhiều hơn một mã mà không có mã cha (ví dụ. nhiều hơn một nút gốc)

3. Có thể có nhiều hệ thống phân cấp (hay "các dạng") xác định, trong các thuật ngữ của các liên kết giữa các mã. Mỗi hệ thống phân cấp đáp ứng một mục đích riêng trong báo cáo, phân tích hoặc phổ biến dữ liệu.

8.2 Tính kế thừa

8.2.1 Sơ đồ lớp



Hình 32 – Sơ đồ lớp về tính kế thừa tập mã

8.2.2 Giải thích sơ đồ

8.2.2.1 Diễn giải

[Chú thích chung: Các ràng buộc về liên kết kế thừa (ví dụ: giữa `CodeAssociation` và `Code`) được chỉ ra trong ngữ cảnh về chức năng của `HierarchicalCodeScheme`. Điều này không có nghĩa là các vai trò liên kết không thể được đặt trên một `Code` trong `HierarchicalCodeScheme` (như đã được xác định trong Bản đồ Mã – xem điều 9). Sơ đồ lớp chỉ giới hạn hoặc quy định các liên kết mà việc sử dụng yêu cầu hỗ trợ chức năng của `HierarchicalCodeScheme`.]

`HierarchicalCodeScheme` kế thừa từ *`ItemScheme`* do đó *`MaintainableArtefact`* với việc định danh, xác định phiên bản và cơ quan duy trì. `CodeAssociation` kế thừa từ `CodeMap` (xem điều 9) do đó là *`VersionableArtefact`*. *`Hierarchy`* kế thừa trực tiếp từ *`VersionableArtefact`*. Hai lớp này có định danh và xác định phiên bản. `Level` là một *`IdentifiableArtefact`* do đó có một Id, tên đa ngôn ngữ và mô tả đa ngôn ngữ.

Điều quan trọng phải hiểu rằng các `Code` trong `HierarchicalCodeScheme` không chứa trong lược đồ - chúng được tham chiếu từ lược đồ và được duy trì trong một hoặc nhiều `CodeList`. Điều này được giải thích trong sơ đồ lớp quan hệ dưới đây.

Các liên kết giữa `CodeAssociation` và `Code` được tạo từ các liên kết giữa `CodeMap` và `Code`. Tuy nhiên, các liên kết kế thừa được quy định như sau:

- Liên kết xác định quan hệ giữa các mã nguồn và đích được giới hạn tới quan hệ “cha ”(ví dụ. `Code` đích là cha)

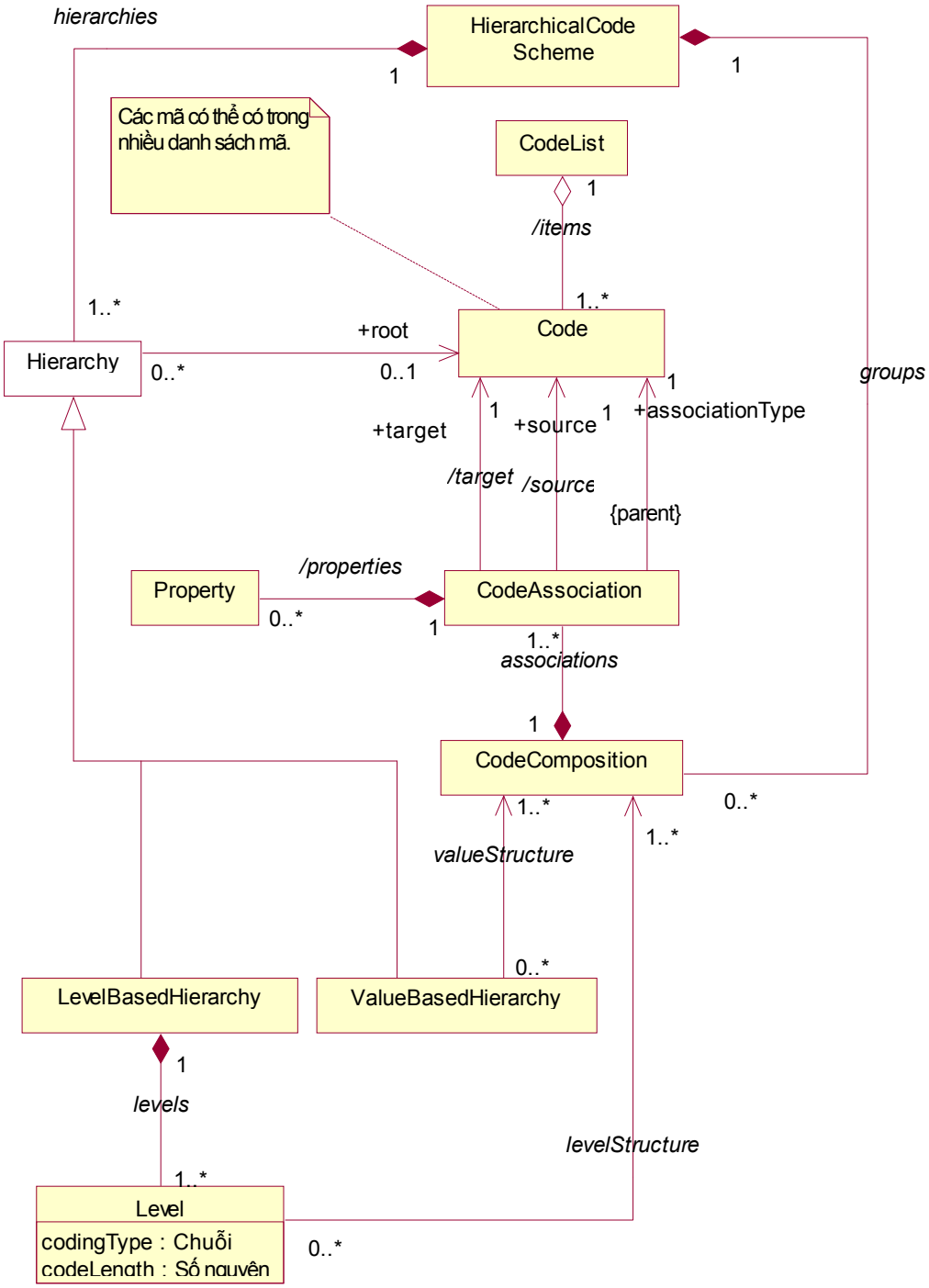
Chú ý rằng `Code` được liên kết bằng `associationType` không cùng `CodeList` cũng như mã nguồn hoặc mã đích – đó là một `CodeList` cụ thể của các vai trò.

8.2.2.2 Định nghĩa

Các định nghĩa về các lớp, các thuộc tính và các liên kết khác nhau được biểu diễn trong đoạn dưới đây:

8.3 Quan hệ

8.3.1 Sơ đồ lớp



Hình 33 – Sơ đồ lớp quan hệ của lược đồ mã phân cấp

8.3.2 Giải thích sơ đồ

8.3.2.1 Diễn giải

Các liên kết và khả năng điều hướng của các liên kết trong `HierarchicalCodeScheme` được quy định để đảm bảo cho biểu diễn nhất quán của `HierarchicalCodeScheme` trong các thuật ngữ về chức năng cơ bản của khóa:

HierarchicalCodeScheme là một đặc tả của các code bao gồm lược đồ và đặc tả về cấu trúc của các code trong lược đồ theo các thuật ngữ của một hoặc nhiều hierarchy.

Các code trong HierarchicalCodeScheme bản thân chúng không phải là một phần của lược đồ, chúng là các tham chiếu tới các code trong một hoặc nhiều codelist bên ngoài.

Các code này có thể tham gia vào một hoặc nhiều hierarchy và một hoặc nhiều codeComposition để đưa ra cấu trúc cho HierarchicalCodeScheme.

Các liên kết giữa bất kỳ hai mã nào được quy đinh trong một CodeAssociation. Liên kết được giới hạn để định danh một code và code cha của nó.

Các quan hệ

Các quan hệ giữa các mã được xác định trong CodeComposition, bao gồm một số CodeAssociation. CodeAssociation kết nối một Code (source) với một Code (target) cha. Quy định là mã cha trong mỗi CodeAssociation của CodeComposition phải cùng một Code. CodeAssociation có thể có một hoặc nhiều Property mà công nhận định nghĩa các đặc tính, ví dụ. một trình tự số hoặc trọng lượng liên quan về một Code (con) đối với sự phân tích Code cha của nó.

Code có thể tham gia vào một hoặc nhiều CodeAssociation, đóng vai trò source (con) or target (cha). Một mã có thể đóng cả hai vai trò nhưng CodeAssociation khác nhau được kết nối với các CodeComposition khác nhau.

Các hệ thống phân cấp

Có khả năng xác định các hệ thống phân cấp chính thức của các Code và một HierarchicalCodeScheme có thể có nhiều hơn một *Hierarchy* này. Mỗi *Hierarchy* có thể định danh Code gốc. Có hai kiểu *Hierarchy* – giá trị cơ sở và mức cơ sở.

Một ValueBasedHierarchy bao gồm một tập prises CodeComposition (bất kỳ kết hợp nào cũng có thể được cho phép theo nguyên tắc).

Một LevelBasedHierarchy hỗ trợ yêu cầu trong đó các mức chính thức cần được xác định. Mỗi Level bao gồm một tập CodeComposition. Ràng buộc của LevelBasedHierarchy là mỗi Code trong Level có một và chỉ một mã cha trong Level cao hơn. Chú ý rằng các phân loại thống kê thường được cấu trúc như một LevelBasedHierarchy.

Level kế thừa từ *IdentifiableArtefact* do đó có một Id, name đa ngôn ngữ, description đa ngôn ngữ, Annotation.

[Chú ý rằng các tổ chức mong muốn tuân theo các mô hình về các phân loại thống kê để bảo đảm rằng Id là số kết hợp với Level, trong đó các Level được đánh số liên tiếp bắt đầu với mức 1 cho tới Level cao nhất].

ItemProperty chấp nhận một hoặc nhiều đặc tính tùy chọn được xác định cho CodeAssociation. Trong ngữ cảnh của HierarchicalCodeScheme, một đặc tính có thể là trình tự trong đó mã nguồn tham gia vào CodeComposition.

8.3.2.2 Định nghĩa

Lớp	Đặc trưng	Mô tả
HierarchicalCode Scheme	kế thừa từ ItemScheme	Tập hợp có tổ chức của các mã mà có thể tham gia vào các quan hệ cha/con với các mã khác trong lược đồ, được xác định bởi một hoặc nhiều hệ thống phân cấp của lược đồ.
	groups	Liên kết với các nhóm của các mã.
	hierarchies	Liên kết với các hệ thống phân cấp của các mã.
CodeComposition		Nhóm các mã trong đó tất cả các mã trong nhóm có một liên kết với cùng một mã cha.
	associations	Liên kết với một liên kết của hai Mã.
CodeAssociation	kế thừa từ	Liên kết giữa hai Mã.
	+source	Liên kết với mã nguồn
	+target	Liên kết với mã đích.
	+associationType	Vai trò của sự liên kết giữa Mã nguồn và Mã đích. Điều này được quy định cho
Code		Mã nguồn hoặc đích
	/items	Liên kết với Danh sách mã chứa Mã.
CodeList		Danh sách mã chứa Mã.
Hierarchy	Lớp trừu tượng Các lớp con: LevelBasedHierar	Cấu trúc phân loại được sắp xếp theo các mức chi tiết từ mức rộng nhất đến mức chi tiết nhất.
	+root	Liên kết với mã có mức cao trong hệ thống phân cấp.
LevelbasedHierarchy	kế thừa từ Hierarchy	Cấu trúc của hệ thống phân cấp trong đó cấu trúc được sắp xếp theo các mức chi tiết từ mức rộng nhất đến mức chi tiết nhất. Mỗi mức được xác định trong các thuật ngữ của các loại ở mức hệ thống phân cấp thấp hơn tiếp theo
	levels	Liên kết với các mức trong hệ thống phân cấp.

Level		Nhóm các mã được mô tả bởi sự mã hóa đồng nhất, trong đó lớp cha của mỗi Mã trong nhóm có cùng mức cao hơn của hệ thống phân cấp.
	codingType	Chỉ ra liệu các mã ở mức đó có theo thứ tự abc, số hoặc vừa có chữ vừa có số
	codeLength	Số các ký tự mà các mã ở mức này có.
	levelStructure	Liên kết các nhóm mã bao gồm mức.
ValueBasedHierarchy	kế thừa từ Hierarchy	Cấu trúc của hệ thống phân cấp trong đó các thuật ngữ trong hệ thống phân cấp không có cấu trúc ở mức chính thức.
	valueStructure	Liên kết với các nhóm mã bao gồm hệ thống phân cấp.

9 Tập cấu trúc và các ánh xạ

9.1 Phạm vi

StructureSet cho phép các thành phần trong một cấu trúc được ánh xạ tới các thành phần trong cấu trúc khác của cùng một kiểu cấu trúc. Trong ngữ cảnh này thuật ngữ "cấu trúc" được sử dụng để bao gồm các kiểu *ItemScheme*, *Structure* và *StructureUsage*. Các cấu trúc cho phép được ánh xạ, các thành phần được ánh xạ trong các cấu trúc này:

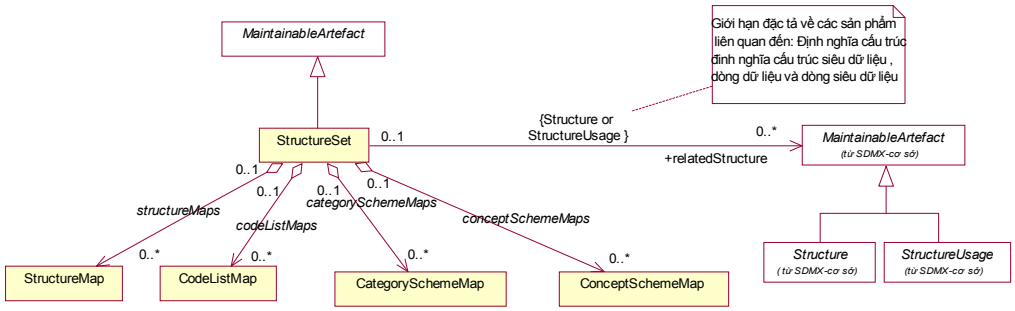
Kiểu Cấu trúc	Kiểu Thành phần
Danh sách mã	Mã
Lược đồ phân loại	Loại
Lược đồ khái niệm	Khái niệm
Định nghĩa cấu trúc dữ liệu (Tập khóa)	Chiều kích thước, Thuộc tính Dữ liệu, Phép đo
Định nghĩa cấu trúc siêu dữ liệu	Thành phần thẻ định danh, thuộc tính siêu dữ liệu
Định nghĩa luồng dữ liệu	Định nghĩa cấu trúc dữ liệu (Tập khóa)
Định nghĩa Dòng Siêu dữ liệu	Định nghĩa luồng dữ liệu

StructureSet có thể chứa một hoặc nhiều "bản đồ" và có thể xác định một hệ thống phân cấp của các bản đồ mà nhóm các bản đồ thành phần phụ liên quan. Một ví dụ:

Định nghĩa luồng dữ liệu ↛ Định nghĩa cấu trúc dữ liệu ↛ [Chiều kích thước, Thuộc tính Dữ liệu, Phép đo]↛ Danh sách mã ↛ mã.

9.2 Tập cấu trúc

9.2.1 Sơ đồ lớp



Hình 34 – Sơ đồ lớp của Tập Cấu trúc

9.2.2 Giải thích sơ đồ

9.2.2.1 Diễn giải

`StructureSet` là *`MaintainableArtefact`*. Nó bao gồm:

1. Tập các tham chiếu tới các lớp con cụ thể của *`Structure`* và *`StructureUsage`* (`KeyFamily`, `MetadataStructureDefinition`,`DataflowDefinition` hoặc `MetadataflowDefinition`) để chỉ ra rằng có một quan hệ về ngữ nghĩa tồn tại giữa chúng. Ví dụ có thể là nhóm của `KeyFamily` cùng tạo ra định nghĩa khối hộp, mỗi `KeyFamily` xác định một phần của khối hộp.
2. Tập các `StructureMap` xác định các thành phần của một cấu trúc tương đương với các thành phần của cấu trúc khác.
3. Tập các `CodeListMap` xác định cách các `Code` được ánh xạ giữa các `CodeLists` hoặc *`Hierarchy`*.
4. Tập các `CategorySchemeMap` xác định cách các `Category` được ánh xạ giữa các `CategoryScheme`
5. Tập các `ConceptSchemeMap` xác định cách các `Concept` được ánh xạ giữa các `ConceptScheme`.

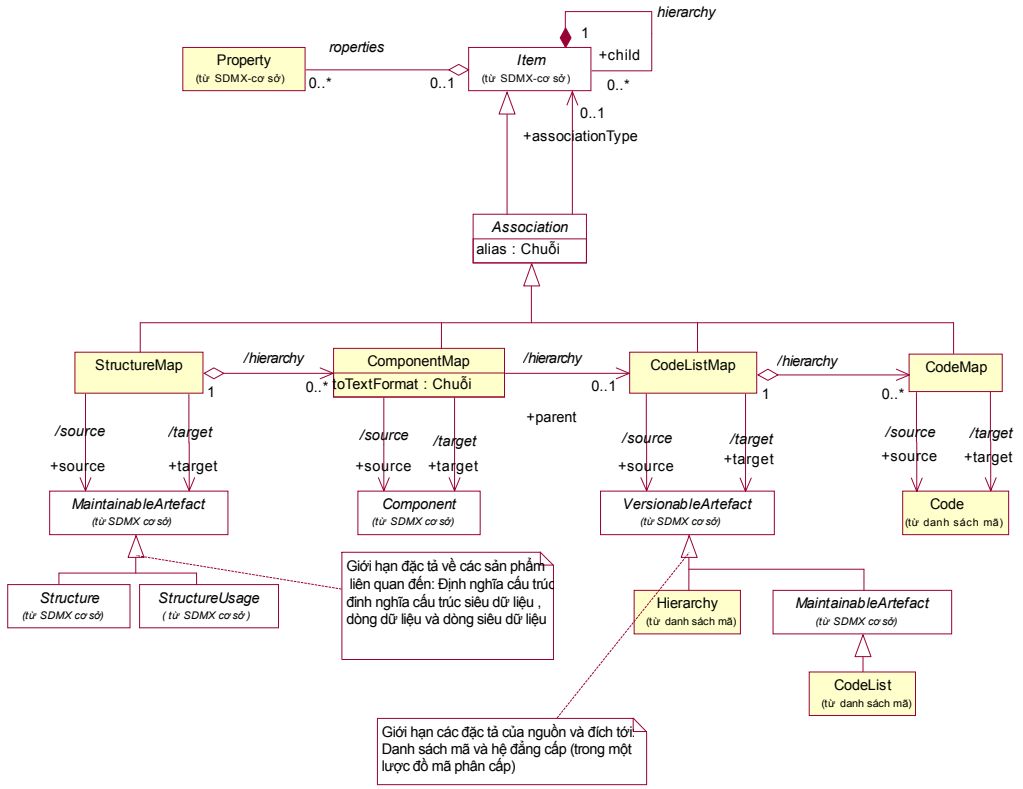
9.2.2.2 Định nghĩa

Lớp	Đặc trưng	Mô tả
`StructureSet`		Tập hợp các bản đồ cấu trúc kết nối các thành phần với nhau trong quan hệ nguồn/đích trong đó có sự tương đương về ngữ nghĩa giữa các thành phần nguồn và đích.

	+relatedStructure	Liên kết với một: tập khóa (Định nghĩa cấu trúc dữ liệu); Định nghĩa cấu trúc siêu dữ liệu; Định nghĩa luồng dữ liệu; Định nghĩa dòng siêu dữ liệu.
	structureMaps	Liên kết với các bản đồ cấu trúc.
	codeListMaps	Liên kết với các bản đồ danh sách mã.
	categorySchemeMaps	Liên kết với Lược đồ phân loại
	conceptSchemeMaps	Liên kết với các lược đồ khái niệm

9.3 Bản đồ cấu trúc

9.3.1 Sơ đồ lớp



Hình 35 – Sơ đồ lớp của bản đồ cấu trúc

9.3.2 Giải thích sơ đồ

9.3.2.1 Diễn giải

`StructureMap` tham chiếu các *`Structure`* hoặc *`StructureUsage`*. Trong các thuật ngữ cụ thể các tham chiếu này sẽ là các `DataStructureDefinition`, `MetadataStructureDefinition`, `DataflowDefinition` hoặc `MetadataflowDefinition`. `StructureMap` chứa một tập `ComponentMap`, mỗi `ComponentMap` chỉ ra sự tương đương giữa các thành phần của *`Structure`* *hoặc* *`StructureUsage`* được tham chiếu. `ComponentMap` có thuộc tính `toTextFormat`, lấy các giá

trị: id, name, description. Điều này chỉ ra các công cụ ánh xạ sử dụng id, tên hoặc mô tả một thành phần được mã hóa để xác định sự tương đương với một giá trị của thành phần không được mã hoá. Với mỗi sự tương đương về *Component* được chỉ ra (đây là sự tương đương Concept), một CodeListMap có thể được xác định.

Ví dụ về ComponentMap kết nối với *Component* nguồn, đó là một Dimension trong KeyFamily nguồn (được định danh trong StructureMap) tới *Component* đích tương đương, đó là một Dimension trong KeyFamily đích.

CodeListMap tham chiếu hai CodeList và Hierarchy (trong HierarchicalCodeScheme). CodeListMap chứa các tập CodeMap, mỗi CodeMap chỉ ra sự tương đương giữa các Code của CodeList được tham chiếu. Một lần nữa, thuộc tính bí danh có thể cung cấp tên cho tất cả các mã tương đương trong các CodeList "liên kết đôi một" để thuận lợi cho việc truy vấn. CodeListMap có thể được kết nối phân cấp với ComponentMap hoặc nó có thể được quy định độc lập với một ComponentMap.

Mỗi bản đồ kế thừa từ *Association* do đó kế thừa liên kết với Property, vì vậy cho phép xác định các đặc tính bổ sung đối với bản đồ.

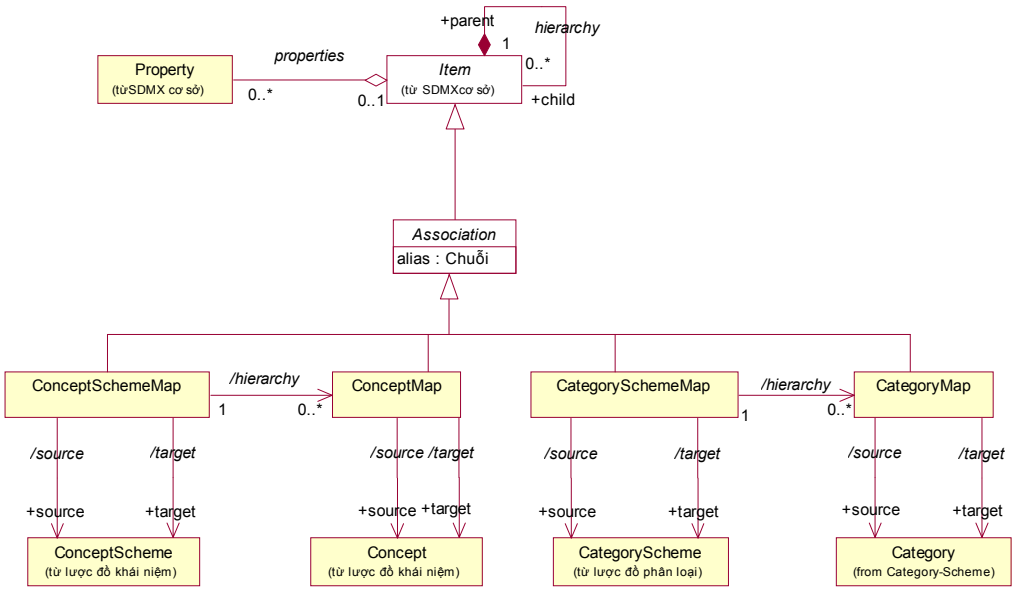
9.3.2.2 Định nghĩa

Lớp	Đặc tính	Mô tả
StructureMap	kế thừa từ *Association*	Kết nối một cấu trúc nguồn và đích trong đó có sự tương đương về ngữ nghĩa giữa các cấu trúc nguồn và đích.
	+source	Liên kết với cấu trúc nguồn.
	+target	Liên kết với cấu trúc đích.
	/hierarchy	Liên kết với các bản đồ thành phần.
ComponentMap		Kết nối một thành phần nguồn và đích trong đó có sự tương đương về ngữ nghĩa giữa các thành phần nguồn và đích.
	+source	Liên kết với thành phần nguồn.
	+target	Liên kết với thành phần đích.
	/hierarchy	Liên kết với các bản đồ danh sách mã.
CodeListMap		Kết nối một danh sách mã hoặc hệ thống phân cấp nguồn với danh sách mã hoặc hệ thống phân cấp đích trong đó có sự tương đương về ngữ nghĩa giữa danh sách mã hoặc hệ thống phân cấp nguồn và đích.
	+source	Liên kết với hệ thống phân cấp hoặc danh sách mã nguồn.
	+target	Liên kết với hệ thống phân cấp hoặc danh sách mã đích.

	/hierarchy	Liên kết với các bản đồ mã.
CodeMap		Kết nối một mã nguồn và đích trong đó có sự tương đương về ngữ nghĩa giữa các mã nguồn và đích.
	+source	Liên kết với mã nguồn.
	+target	Liên kết với mã đích.

9.4 Lược đồ khái niệm và lược đồ phân loại

9.4.1 Sơ đồ lớp



Hình 36 – Sơ đồ lớp của lược đồ khái niệm và lược đồ phân loại

9.4.2 Giải thích sơ đồ

9.4.2.1 Diễn giải

`ConceptSchemeMap` cung cấp một cơ chế cho việc quy định sự tương đương về ngữ pháp giữa các khái niệm. `ConceptSchemeMap` định danh hai `ConceptScheme` trong đó các `Concept` của nó được ánh xạ. Chú ý rằng nhiều lược đồ có thể cùng tham gia thông qua một tập các ánh xạ theo cặp. `ConceptMap` chỉ rõ các `Concept` tương đương về ngữ nghĩa và một bí danh được quy định liên quan đến tập các khái niệm ánh xạ để tạo thuận lợi cho việc truy vấn.

`CategorySchemeMap` tương tự với `ConceptSchemeMap`, ngoại trừ cách sử dụng của nó nhằm vào việc diễn đạt sự tương đương về ngữ nghĩa trong các `CategoryScheme` như lược đồ lĩnh vực chủ đề.

9.4.2.2 Định nghĩa

Lớp	Đặc trưng	Mô tả
ConceptSchemeMap		Liên kết một lược đồ khái niệm nguồn và đích trong đó có sự tương đương về ngữ nghĩa giữa các lược đồ nguồn và đích.
	+source	Liên kết với lược đồ khái niệm nguồn.
	+target	Liên kết với lược đồ khái niệm đích.
	/hierarchy	Liên kết với các bản đồ khái niệm.
Concept Map		Liên kết một Khái niệm nguồn và đích trong đó có sự tương đương về ngữ nghĩa giữa các Khái niệm nguồn và đích.
	+source	Liên kết với khái niệm nguồn.
	+target	Liên kết với khái niệm đích.
CategorySchemeMap		Liên kết một Lược đồ phân loại nguồn và đích trong đó có sự tương đương về ngữ nghĩa giữa các lược đồ nguồn và đích.
	+source	Liên kết với Lược đồ phân loại nguồn.
	+target	Liên kết với Lược đồ phân loại đích.
	/hierarchy	Liên kết với các bản đồ phân loại.
Concept Map		Liên kết một phân loại nguồn và đích trong đó có sự tương đương về ngữ nghĩa giữa các phân loại nguồn và đích.
	+source	Liên kết với phân loại nguồn.
	+target	Liên kết với phân loại đích

10 Các ràng buộc về dữ liệu và việc cung cấp

10.1 Phạm vi

Điều này mô tả hỗ trợ trong siêu mô hình về việc quy định truy cập và nội dung của nguồn dữ liệu. Thông tin có thể được lưu trữ trong nguồn ví dụ như sổ đăng ký sử dụng các ứng dụng để định vị dữ liệu và siêu dữ liệu có sẵn thông qua Internet.

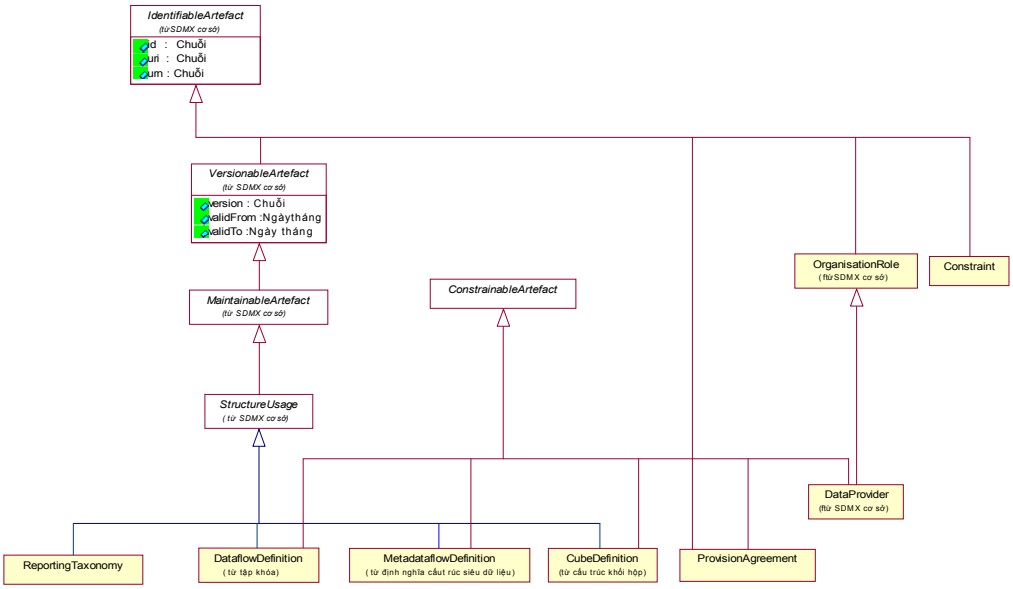
Chú ý rằng trong siêu mô hình này thuật ngữ nguồn dữ liệu liên quan đến cả nguồn dữ liệu và siêu dữ liệu, người cung cấp dữ liệu liên quan đến cả người cung cấp dữ liệu và siêu dữ liệu.

Nguồn dữ liệu có thể là một tệp dữ liệu hoặc siêu dữ liệu đơn giản (trong định dạng SDMX-ML) hoặc kho chứa siêu dữ liệu hoặc cơ sở dữ liệu. Một nguồn dữ liệu có thể chứa dữ liệu về nhiều dữ liệu hoặc nhiều dòng siêu dữ liệu (được gọi là `DataflowDefinition`, `CubeDefinition` và `MetadataflowDefinition` trong mô hình) và các cơ chế mô tả trong điều này cho phép `DataProvider` quy định phạm vi về nội dung của nguồn dữ liệu.

Bản thân các `DataflowDefinition`, `MetadataflowDefinition` và `CubeDefinition` có thể được quy định khi chỉ chứa một tập tất cả các khóa được tạo từ `KeyFamily`, `MetadataStructureDefinition` hoặc `CubeStructure`. `DataProvider` có thể chứa thêm tập các khóa này bằng cách mô tả tập con có sẵn trong nguồn dữ liệu hoặc siêu dữ liệu. Các đặc tả này được gọi là *`Constraint`* trong mô hình này.

10.2 Kế thừa

10.2.1 Sơ đồ lớp về tính kế thừa của sản phẩm có thể ràng buộc và việc cung cấp



Hình 37 – Sơ đồ lớp về tính kế thừa của các sản phẩm có tính ràng buộc và cung cấp

10.2.2 Giải thích sơ đồ

10.2.2.1 Diễn giải

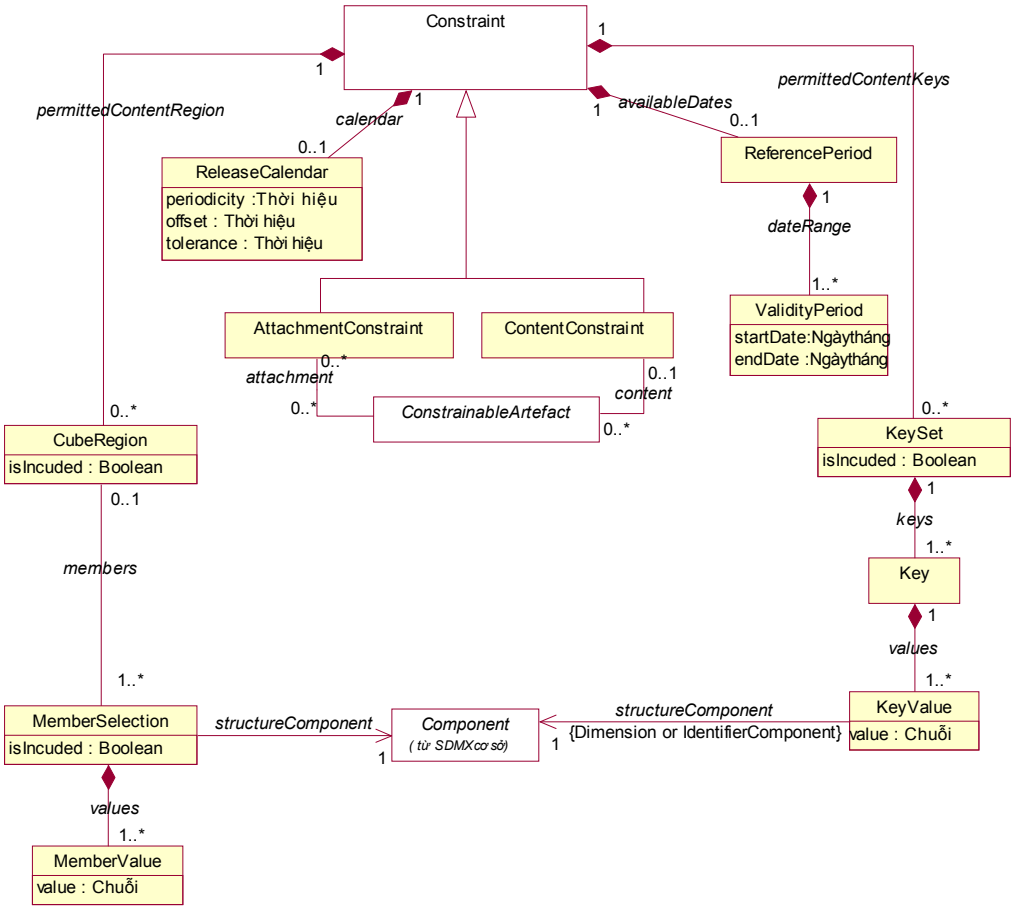
Mọi sản phẩm được tạo từ *`ConstrainableArtefact`* có thể có các ràng buộc được xác định. Các sản phẩm mà có siêu dữ liệu ràng buộc đính kèm:

- `DataflowDefinition`
- `ProvisionAgreement`
- `DataProvider`

- MetadataflowDefinition
- CubeDefinition

10.3 Ràng buộc

10.3.1 Sơ đồ lớp quan hệ của siêu dữ liệu ràng buộc



Hình 38 – Sơ đồ lớp về quan hệ biểu diễn siêu dữ liệu ràng buộc

10.3.2 Giải thích sơ đồ

10.3.2.1 Diễn giải

Cơ chế ràng buộc cho phép các ràng buộc cụ thể được đính kèm với một *ConstrainableArtefact*. Ngoại trừ ReleaseCalendar, các ràng buộc này quy định tập con của toàn bộ tập giá trị hoặc khóa có trong DataSet hoặc MetadataSet. Toàn bộ tập giá trị được kết luận từ định nghĩa cấu trúc liên quan (KeyFamily, MetadataStructureDefinition và CubeStructure).

Ví dụ KeyFamily quy định về mỗi Dimension, danh sách các giá trị mã cho phép. Tuy nhiên, DataflowDefinition cụ thể sử dụng KeyFamily có thể chứa một tập con các dải khóa mà có thể thực hiện được về mặt lý thuyết từ định nghĩa KeyFamily (tổng số khả năng đôi khi được gọi là sản phẩm của các giá trị về chiều kích thước). Thêm vào đó, DataProvider có khả năng cung cấp dữ liệu theo DataflowDefinition có ProvisionAgreement và DataProvider có thể cung cấp siêu dữ

liệu ràng buộc mà quy định thêm các khả năng để mô tả dữ liệu các nhà cung cấp có thể cung ứng.

ConstrainableArtefact có thể có hai kiểu *Constraint*:

1. ContentConstraint – được sử dụng duy nhất như một cơ chế để quy định một tập khóa có sẵn (KeySet) hoặc tập các giá trị thành phần (CubeRegion) trong một *DataSource* ví dụ như DataSet hoặc cơ sở dữ liệu (*QueryDatasource*). Chỉ có ràng buộc này có thể hiện diện cho *ConstrainableArtefact*.
2. AttachmentConstraint – được sử dụng như một cơ chế để xác định các khoanh của tập dữ liệu đầy đủ và các kiểu đối tượng khác trong mô hình (ví dụ như CubeComponent – sẽ thấy dòng chữ **Error! Reference source not found.**) được đính kèm. Các khoanh này có thể xác định như một tập các khóa (KeySet) hoặc một tập các giá trị thành phần (CubeRegion). Có thể có nhiều AttachmentConstraint quy định về AttachableArtefact cụ thể.

Constraint là một *IdentifiableArtefact* do đó có thể được liên kết với một hoặc nhiều *AttachableArtefact*. Tuy nhiên, vì Constraint có thể quy định một tập con các giá trị thành phần bao hàm *Structure* cụ thể (ví dụ như KeyFamily và CubeStructure cụ thể) do đó toàn bộ *AttachableArtefact* phải được liên kết với cùng *Structure* cụ thể.

Một *Constraint* có hai cách quy định các tập con về giá trị:

1. Khi tập các khóa (KeySet) có mặt trong DataSet hoặc MetadataSet. KeySet quy định số lượng các Key trong các thuật ngữ của các KeyValue của chúng. Mỗi KeyValue là giá trị mà có thể hiện diện cho *Component* (đặc biệt Dimension or IdentifierComponent) của cấu trúc khi chứa trong DataSet hoặc MetadataSet.
2. Khi tập các vùng khối hộp, trong đó mỗi vùng khối hộp xác định một “khoanh” của toàn bộ cấu trúc trong các thuật ngữ của một hoặc nhiều giá trị mà có thể hiện diện cho một Thành phần (có thể là bất kỳ kiểu Thành phần nào) của cấu trúc khi chứa trong Tập dữ liệu hoặc tập siêu dữ liệu.

Sự khác biệt giữa (1) and (2) ở trên: trong (1) khóa hoàn thiện được xác định trong khi ở (2) CubeRegion xác định danh sách các giá trị có khả năng cho mỗi *Component* nhưng không quy định các sự kết hợp cụ thể. Thêm vào đó, trong (1) liên kết giữa *Component* và KeyValue bị ràng buộc với các thành phần bao gồm khóa và thẻ định danh, trong khi ở (2) nó có thể chứa các thành phần khác (ví dụ như các thuộc tính). Giá trị trong KeyValue.value và MemberValue.value phải nhất quán với *Representation* biểu thị về *Component* trong KeyFamily hoặc MetadataStructureDefinition kết nối với DataflowDefintion hoặc MetadataflowDefinition. Chú ý rằng trong tất cả trường hợp “toán tử” về value được cho rằng là “ngang bằng”

Có khả năng xác định về KeySet, CubeRegion và MemberSelection. Liệu một tập được bao gồm (isIncluded = “đúng”) hoặc loại trừ (isIncluded=”sai”) từ định nghĩa ràng buộc hoặc không. Thuộc tính này có ích nếu chỉ có một tập con các giá trị có khả năng xảy ra được bao gồm trong tập, lúc đó tập con nhỏ hơn có thể được xác định và loại trừ khỏi sự ràng buộc.

Thêm vào đó các KeySet hoặc CubeRegion, Constraint có thể có:

- `ReferencePeriod` xác định một hoặc nhiều dải (`ValidityPeriod`) quy định các khoảng thời gian về các dữ liệu và siêu dữ liệu có sẵn.
- `ReleaseCalendar` quy định giai đoạn ấn bản dữ liệu và siêu dữ liệu.

`ReleaseCalendar` xác định kế hoạch về việc ấn bản dữ liệu được sắp xếp trong các thuật ngữ định kỳ và đưa ra thông tin đầy đủ để tính toán kế hoạch ấn bản. Khoảng trống được tính từ thời gian bắt đầu như đã được xác định ở ISO 8601 ví dụ: tất cả giai đoạn bắt đầu từ ngày liên quan đầu tiên hoặc sau ngày 1 tháng 1 vì vậy một phần tư giai đoạn sẽ bắt đầu ngày 1 tháng 1, ngày 1 tháng 4, ngày 1 tháng 7, ngày 1 tháng 10 và chu kỳ theo tuần sẽ bắt đầu từ tuần mà có ngày thứ 5 là ngày đầu tiên của năm.

10.3.2.2 Định nghĩa

Lớp	Đặc trưng	Mô tả

Constrainable
Artefact
	Lớp trừu tượng
	Các lớp con:
	`DataflowDefinition`
	`Metadataflow Definition`
	`CubeDefinition`
	`ProvisionAgreement`
	`DataProvider`
		Một sản phẩm mà có các ràng buộc được quy định.

	`content`	Liên kết siêu dữ liệu mà ràng buộc nội dung được tìm thấy trong nguồn dữ liệu hoặc siêu dữ liệu kết nối với Sản phẩm Ràng buộc
	`attachment`	Liên kết siêu dữ liệu mà ràng buộc nội dung hợp lệ của tập dữ liệu hoặc siêu dữ liệu trong đó một sản phẩm ràng buộc (ví dụ như Mục Khối hộp với vai trò "thuộc tính") có thể được đính kèm

Constraint	Lớp trừu tượng. Các lớp con:
	`AttachmentConstraint`
	`ContentConstraint`
		Quy định một tập con của định nghĩa nội dung tập dữ liệu hoặc siêu dữ liệu cho phép trong các thuật ngữ nội dung về dữ liệu, trong các thuật ngữ của tập các kết hợp chính trong đó các thuộc tính cụ thể (được xác định bởi Cấu trúc) có thể được đính kèm

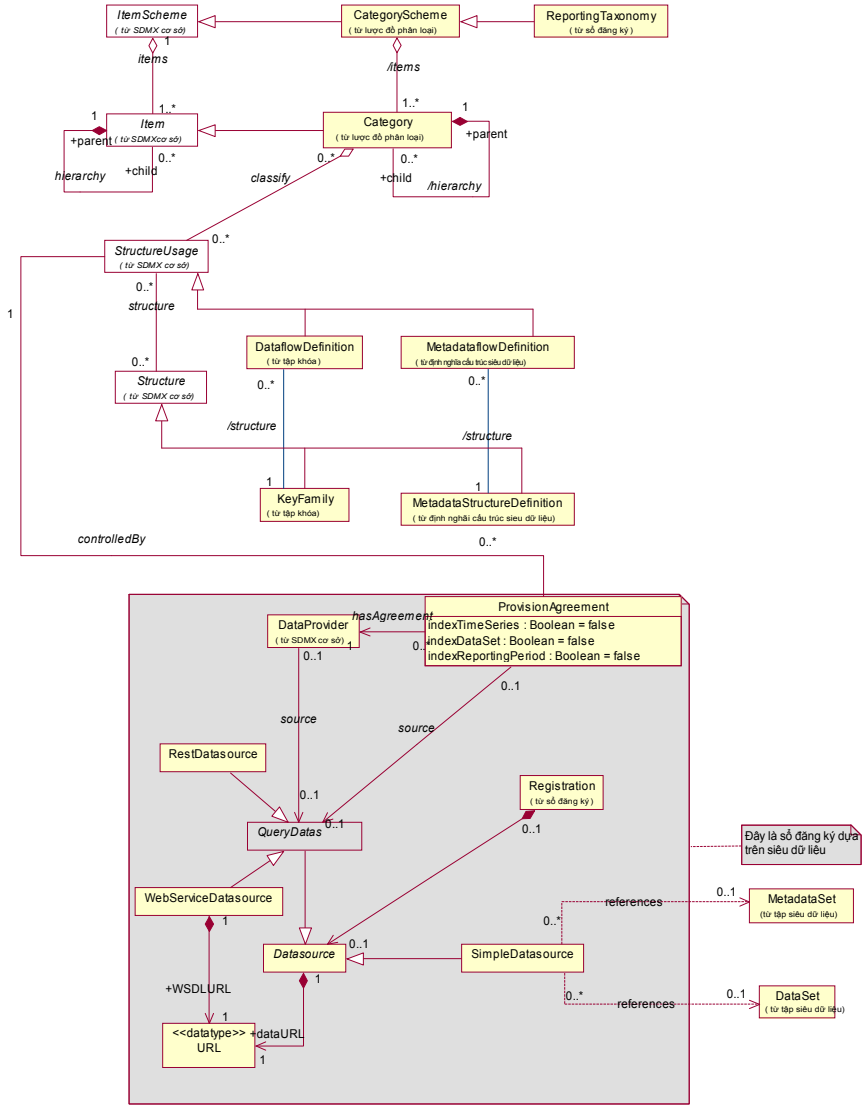
	availableDates	Liên kết với tập các khoảng thời gian mà định danh các dải thời gian mà dữ liệu có sẵn trong nguồn dữ liệu.
	permittedContentKeys	Liên kết với một tập các khóa (ví dụ. sự kết hợp giá trị) mà có thể được tạo từ định nghĩa Cấu trúc trong đó Sản phẩm Ràng buộc được kết nối.
	permittedContent Region	Liên kết với một tập các giá trị thành phần (ví dụ: sự kết hợp giá trị) mà có thể được tạo từ định nghĩa Cấu trúc trong đó Sản phẩm Ràng buộc được kết nối.
	calendar	Liên kết với lịch biểu xác định ngày tháng mà sản phẩm có hiệu lực.
ContentConstraint	kế thừa từ *Constraint*	Xác định một Ràng buộc trong các thuật ngữ của nội dung được tìm thấy ở các tập dữ liệu và siêu dữ liệu kết nối với Sản phẩm Ràng buộc trong đó ràng buộc này được liên kết.
Attachment Constraint	kế thừa từ *Constraint*	Xác định Ràng buộc trong các thuật ngữ của sự kết hợp các giá trị thành phần được tìm thấy trong tập dữ liệu, Sản phẩm Ràng buộc có thể được liên kết trong một định nghĩa cấu trúc.
KeySet		Tập các khóa.
	isIncluded	Chỉ ra liệu tập khóa được bao gồm trong định nghĩa ràng buộc hoặc có bị loại khỏi định nghĩa ràng buộc hoặc không.
	keys	Liên kết các khóa.
Key		Tập các giá trị khóa bao gồm khóa.
	values	Liên kết các giá trị khóa.
KeyValue		Giá trị của thành phần bao gồm một phần của khóa.

	structureComponent	Liên kết Thành phần trong Cấu trúc mà Sản phẩm Ràng buộc được kết nối, xác định Biểu diễn hợp lệ cho Giá trị khóa.
Component		Xem 3.5.3.2
CubeRegion		Tập các thành phần và giá trị của nó mà xác định một tập con hoặc "khoanh" của toàn bộ nội dung có thể thực hiện trong Cấu trúc trong đó Sản phẩm Ràng buộc được kết nối.
	isIncluded	Chỉ ra liệu Khu vực Khối hộp được bao gồm trong định nghĩa ràng buộc hoặc có bị loại khỏi định nghĩa ràng buộc hoặc không.
	members	Liên kết tập các thành phần mà xác định tập con của các giá trị.
MemberSelection		Tập các giá trị cho phép về một thành phần của trục.
	isIncluded	Chỉ ra liệu Lựa chọn Thành viên được bao gồm trong định nghĩa ràng buộc hoặc có bị loại khỏi định nghĩa ràng buộc
	structureComponent	Liên kết với Thành phần trong Cấu trúc mà Sản phẩm Ràng buộc được kết nối, xác định Biểu diễn hợp lệ về các Giá trị Thành viên.
MemberValue		Giá trị của thành phần của một tập Thành viên.
	value	Giá trị của Thành phần.
ReleaseCalendar		Xác định kế hoạch ấn bản trong các thuật ngữ của chu kỳ và thời gian.
	periodicity	Chu ký của các ấn bản trong các thuật ngữ của danh sách các chu kỳ thời gian được biết tới (ví dụ, hàng tháng, hàng quý)

	offset	Khoảng trống trong các ngày từ lúc bắt đầu khoảng thời gian.
	tolerance	Số ngày được chấp nhận mà ấn bản có thể trước hoặc sau ngày ngoại lệ.
ReferencePeriod		Tập các ngày mà ràng buộc nội dung có thể được tìm thấy trong tập dữ liệu hoặc siêu dữ liệu.
	dateRange	Liên kết với các Giai đoạn Hợp lệ.
ValidityPeriod		Một khoảng thời gian xác định giá trị hợp lệ.
	startDate	Thời gian bắt đầu giai đoạn.
	endDate	Thời gian cuối của giai đoạn.

10.4 Cung cấp dữ liệu

10.4.1 Sơ đồ lớp



Hình 39 – Sơ đồ lớp về tính kế thừa và quan hệ của việc cung cấp dữ liệu

10.4.2 Giải thích sơ đồ

10.4.2.1 Diễn giải

Mô hình phụ này kết nốt rất nhiều sản phẩm trong SDMX-IM và còn là mấu chốt của sổ đẳng ký siêu dữ liệu trong SDMX, khi tất cả các sản phẩm trong mô hình phụ này có thể truy cập tới một ứng dụng mà có trách nhiệm về việc đăng ký dữ liệu và siêu dữ liệu hoặc về ứng dụng mà yêu cầu truy cập tới dữ liệu và siêu dữ liệu.

Trong khi đó một sổ đăng ký có thể chứa tất cả siêu dữ liệu được mô tả trong sơ đồ trên, các lớp nằm trong vùng mầu xám đặc trưng cho sổ đăng ký dựa trên kịch bản trong đó các nguồn dữ liệu (các tập dữ liệu hoặc siêu dữ liệu tự nhiên hoặc các kho chứa siêu dữ liệu và cơ sở dữ liệu) được đăng ký. Có nhiều chi tiết hơn về cách sử dụng các lớp này trong sổ đăng ký có thể được tìm thấy trong Giao diện Tài liệu Đăng ký của SDMX.

ProvisionAgreement kết nối tất cả các sản phẩm mà xác định cách các dữ liệu và siêu dữ liệu được sắp xếp và phân loại (*StructureUsage*) cho DataProvider và nó kết nối với Datasource, liệu đây có phải là tệp SDMX có trên website hoặc không (SimpleDatasource) hoặc một dịch vụ cơ sở dữ liệu có khả năng hỗ trợ, truy vấn SDMX và giáp đáp các tài liệu SDMX(*QueryDatasource*) hoặc không.

StructureUsage có các lớp DataflowDefinition, MetadataflowDefinition và CubeDefinition cụ thể định danh KeyFamily, MetadataStructureDefinition hoặc CubeStructure tương ứng, nó kết nối với một hoặc nhiều Category trong CategoryScheme ví dụ như một lược đồ lĩnh vực chủ đề, trong đó *StructureUsage* có thể được phân loại (ví dụ, để giúp cho việc thu hẹp các lĩnh vực chủ đề để tìm ra dữ liệu hoặc siêu dữ liệu liên quan).

ReportingTaxonomy cho phép tổ chức xác định lược đồ báo cáo trong đó lược đồ báo cáo xác định nhiều gói dữ liệu riêng lẻ, mỗi gói có kết cấu khác nhau, sau đó kết hợp trong một tập báo cáo. Bản thân ReportingTaxonomy không có *Structure* chi tiết, đúng hơn nó có một mức cao xác định trong CategoryScheme. Các lược đồ này phổ biến trong báo cáo đầu tiên và được môt tả sau đó (xem 10.5).

SimpleDatasource kết nối với DataSet hoặc MetadataSet trên website (điều này được chỉ rõ trên sơ đồ và được gọi là “các tham chiếu”). sourceURL đạt được trong suốt quá trình đăng ký DataSet hoặc MetadataSet. Siêu dữ liệu về nội dung của SimpleDatasource được lưu trữ ở sổ đăng ký trong các thuật ngữ của ContentConstraint (xem10.3) về Registration.

QueryDatasource kết nối với kho chứa dữ liệu và siêu dữ liệu mà chứa các dữ liệu và siêu dữ liệu. sourceURL đạt được trong suốt quá trình đăng ký *QueryDatasource*. Siêu dữ liệu về nội dung của *QueryDatasource* được lưu trữ ở sổ đăng ký trong các thuật ngữ của ContentConstraint (xem 10.3) về ProvisionAgreement. Trong một số trường hợp về DataProvider. Trường hợp sau không được mong đợi nhiều bởi vì cơ sở dữ liệu thực tế giống hầu hết ProvisionAgreement về DataProvider, nó có thể là ContentConstraint về ProvisionAgreement rõ ràng hơn cho ứng dụng truy vấn sổ đăng ký về nguồn dữ liệu để hoàn thiện phạm vi truy vấn.

Có hai kiểu *QueryDatasource*, RestDatasource được yêu cầu sử dụng HTTP “get” và WebServiceDatasource phù hợp với hiện trạng của ngôn ngữ xác định dịch vụ web (WSDL) có sẵn từ wsdURL.

10.4.2.2 Định nghĩa

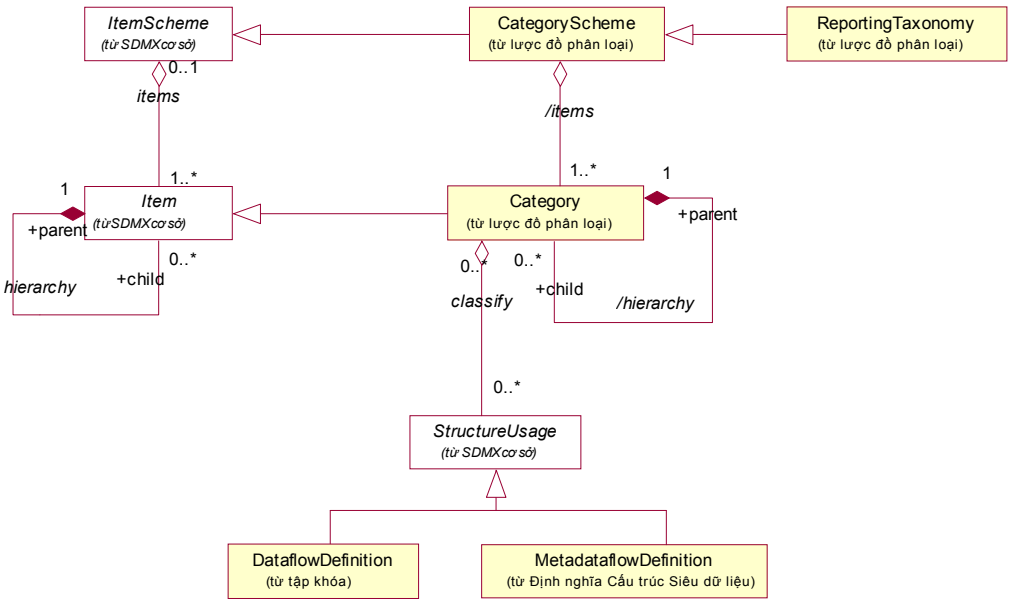
Lớp	Đặc trưng	Mô tả
StructureUsage	Lớp trừu tượng: Các lớp con là: Định nghĩa luồng dữ liệu Định nghĩa dòng siêu dữ liệu Định nghĩa khối hộp Nguyên tắc phân loại báo cáo	Xem 3.5.3.2

	controlledBy	Liên kết các Thỏa thuận cung cấp bao gồm siêu dữ liệu liên quan tới việc cung cấp dữ liệu.
DataProvider		Xem 4.8.2.2.
	hasAgreement	Liên kết với Thỏa thuận cung cấp mà Nhà cung cấp cung ứng dữ liệu và siêu dữ liệu.
	source	Liên kết với nguồn dữ liệu hoặc siêu dữ liệu mà xử lý truy vấn dữ liệu hoặc siêu dữ liệu.
ProvisionAgreement		Kết nối nhà cung cấp dữ liệu với Việc sử dụng Cấu trúc liên quan (ví dụ. Định nghĩa luồng dữ liệu hoặc Dòng siêu dữ liệu) mà nhà nhà cung cấp cung ứng dữ liệu hoặc siêu dữ liệu. Sự thỏa thuận có thể ràng buộc phạm vi của dữ liệu hoặc siêu dữ liệu được cung cấp.
	source	Liên kết với nguồn dữ liệu hoặc siêu dữ liệu mà có thể xử lý truy vấn dữ liệu hoặc siêu dữ liệu.
Datasource	Lớp trừu tượng: Các lớp con: *QueryDatasource*	Định danh vị trí hoặc dịch vụ từ nơi dữ liệu hoặc siêu dữ liệu có thể đạt được.
	sourceURL	URL của nguồn dữ liệu hoặc siêu dữ liệu.
QueryDatasource	Lớp trừu tượng: kế thừa từ *Datasource*	Nguồn dữ liệu hoặc siêu dữ liệu mà có thể xử lý truy vấn dữ liệu hoặc siêu dữ liệu.
RestDatasource		Nguồn dữ liệu có thể truy cập thông qua giao diện Rest
WebService Datasource		Một nguồn dữ liệu phù hợp với giao diện web.
	wsdlURL	URL của hiện trạng ngôn ngữ xác định dịch vụ web.

Registration		Lớp này không được thể hiện chi tiết ở đây nhưng được chỉ ra như một kết nối giữa SDMX-IM và Dịch Vụ Đăng ký API. Nó biểu thị tài liệu đăng ký dữ liệu và siêu dữ liệu.

10.5 Nguyên tắc phân loại báo cáo

10.5.1 Sơ đồ lớp



Hình 40 – Sơ đồ lớp về nguyên tắc phân loại báo cáo

10.5.2 Giải thích sơ đồ

Lớp	Đặc trưng	Mô tả
ReportingTaxonomy		Một lược đồ xác định cấu trúc tổng hợp của một báo cáo dữ liệu trong đó mỗi thành phần có thể được mô tả bởi một Định nghĩa luồng dữ liệu độc lập.

10.5.2.1 Diễn giải

Trong một vài môi trường báo cáo dữ liệu và các môi trường riêng trong báo cáo đầu tiên, báo cáo có thể bao gồm nhiều dữ liệu không đồng nhất, mỗi dữ liệu được mô tả bởi Cấu trúc khác nhau. Định nghĩa của tập các báo cáo phụ được hỗ trợ bởi ReportingTaxonomy.

ReportingTaxonomy là một dạng chuyên môn hóa của CategoryScheme. Mỗi Category của ReportingTaxonomy có thể kết nối với *StructureUsage* mà bản thân nó có thể là DataflowDefinition hoặc MetadataflowDefinition. Trong một ReportingTaxonomy cụ thể mỗi Category sẽ được kết nối với cùng một lớp (ví dụ: tất cả Category trong lược đồ sẽ kết nối với DataflowDefinition). Chú ý rằng Category có thể có Category con và theo cách này nó có khả

năng xác định `ReportingTaxonomy` phân cấp. Trong nguyên tắc phân loại này một vài `Category` được xác định để đưa ra cấu trúc báo cáo.

Ví dụ:

Phần 1

DatafowDefinition_1

DatafowDefinition_2

Phần 2

DatafowDefinition_3

DatafowDefinition_4

Ở đây, các nút của phần 1 và phần 2 không được kết nối với `DataflowDefinition` nhưng ở phần khác thì có thể kết nối được (và sau đó là `KeyFamily`).

10.5.2.2 Định nghĩa

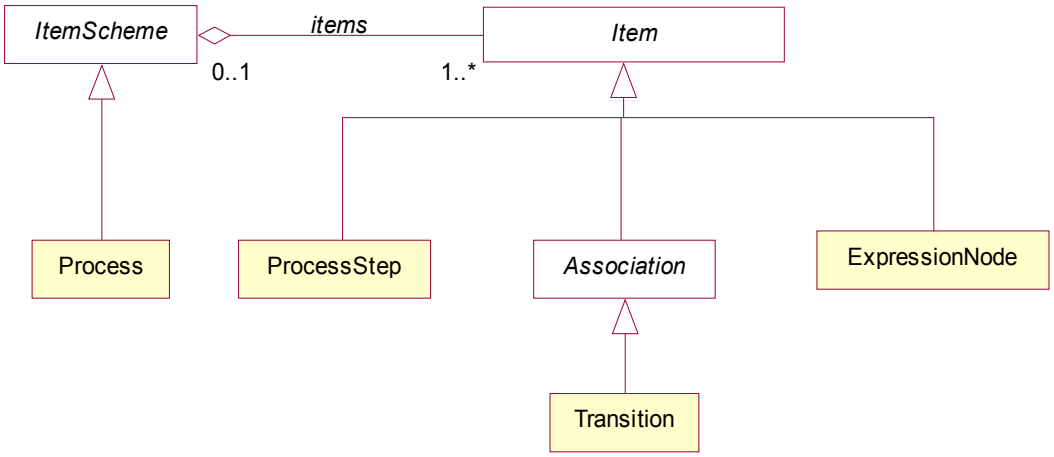
11 Quá trình và sự chuyển tiếp

11.1 Giới thiệu

Trong bất kỳ hệ thống xử lý dữ liệu và siêu dữ liệu, bản thân hệ thống là chuỗi các quá trình và mỗi quá trình xử lý dữ liệu hoặc siêu dữ liệu phải trải thông qua một chuỗi chuyển tiếp. Đây là một hướng đi đúng từ dữ liệu thô tới dữ liệu và siêu dữ liệu công bố. Mô hình quá trình được thể hiện ở đây là mô hình chung có thể lấy được thông tin chính về các giai đoạn này theo cách nguyên bản và chính thức bằng cách sử dụng các biểu thức, khả năng kết nối với các đối tượng định danh cụ thể.

11.2 Quan điểm kế thừa - Mô hình

11.2.1 Sơ đồ lớp



Hình 41 – Sơ đồ lớp về tính kế thừa của quá trình và chuyển tiếp

11.2.2 Giải thích sơ đồ

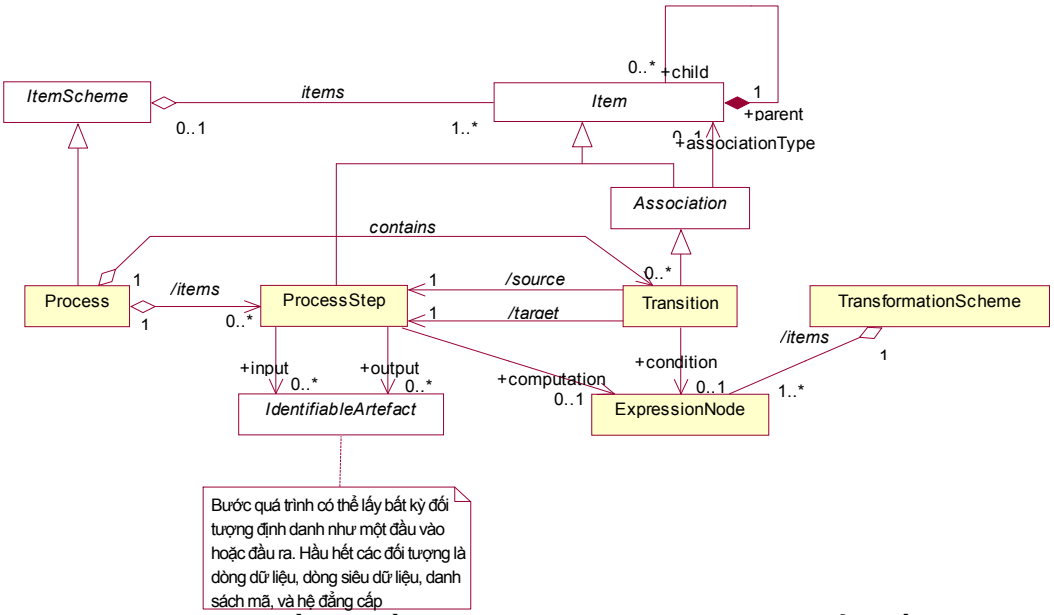
11.2.2.1 Diễn giải

`Process` là *`ItemScheme`*, `ProcessStep` là *`Item`*, do đó `Process` là các `ProcessStep` hình cây. Điều này ngụ ý rằng bất kỳ `ProcessStep` có thể bao gồm số lượng `ProcessStep` phụ tùy ý. `ExpressionNode` cũng là t *`Item`* và được sử dụng để mô tả các tính toán chứa trong `ProcessStep` và để xác định sự điều hướng từ `Process` đến `Process`.

Định nghĩa về các lớp này có thể được tìm thấy dưới đây.

11.3 Tổng quan về quan hệ mô hình

11.3.1 Sơ đồ lớp



Hình 42 – Sơ đồ lớp về quan hệ của quá trình và chuyển tiếp

11.3.2 Giải thích sơ đồ

11.3.2.1 Tường thuật

`Process` là lược đồ của các `ProcessStep` phân cấp. Mỗi `ProcessStep` có thể lấy hoặc không lấy nhiều `ProcessStep` như đầu ra và đầu vào. Thông thường, chúng là các `DataflowDefinition`, `Hierarchy` và `CodeList`-nhưng cũng có thể là bất kỳ cái gì trong mô hình. Việc tính toán được biểu diễn bởi `ProcessStep` mô tả tùy ý bởi `ExpressionNode`, có thể biểu diễn một biểu thức liên quan đến bất kỳ đối tượng mô hình định danh nào. `ProcessStep` cũng có thể được mô tả nguyên bản trong đa ngôn ngữ. `Transition` điều chỉnh việc thực hiện của các `ProcessStep` từ `ProcessStep` nguồn đến `ProcessStep` đích dựa trên việc đánh giá điều kiện xác định trong `ExpressionNode`. Sự chuyển tiếp có thể được sử dụng cho vòng lặp và việc thực hiện có điều kiện của các `ProcessStep`.

Nội dung của điều về CÁC BIỂU THỨC VÀ CÁC PHÉP BIẾN ĐỔI giải thích cấu trúc của `ExpressionNode` và `TransformationScheme`.

Thao tác biểu diễn bằng dữ liệu để kế thừa thông tin mới theo một tập quy tắc cho trước.

11.3.2.2 Định nghĩa

Lớp	Đặc trưng	Mô tả
process	kế thừa từ ItemScheme	Một lược đồ xác định hoặc tài liệu hóa các thao tác được trình diễn bằng dữ liệu để làm hợp lệ dữ liệu hoặc để kế thừa thông tin mới theo tập các quy tắc cho trước
	/items	Liên kết các Các bước Quá trình
	contains	Liên kết các chuyển tiếp
ProcessStep		Một thao tác cụ thể, trình diễn bằng dữ liệu để làm hợp lệ hoặc để kế thừa thông tin mới theo một tập các quy tắc cho trước
	+input	Liên kết các Sản phẩm Có thể định danh mà là các đầu vào dẫn đến Bước Quá trình
	+output	Liên kết các Sản phẩm Có thể định danh mà là các đầu ra của Bước Quá trình
Transition		Một biểu thức theo cách nguyên bản hoặc chính thức của sự biến đổi dữ liệu dữ giữa hai thao tác cụ thể được trình diễn bằng dữ liệu.
	/source	Liên kết Bước Quá trình mà là nguồn của Sự chuyển tiếp
	/target	Liên kết Bước Quá trình mà là đích của Sự chuyển tiếp
	+condition	Liên kết một Nút của Biểu thức

12 Phép biến đổi và biểu thức

12.1 Phạm vi

Mục đích của gói này trong mô hình là khả năng kiểm tra sự kế thừa của dữ liệu. Điều này tương tự như ở khái niệm về dòng dõi trong kho dữ liệu – ví dụ: dữ liệu thu được hoặc được tạo.

Chức năng của phần này cho phép việc định danh và tài liệu hóa các chức năng biểu diễn (các chức năng được tự động hóa một cách thông thường, lập trình các chức năng), cũng như xác định các cấu

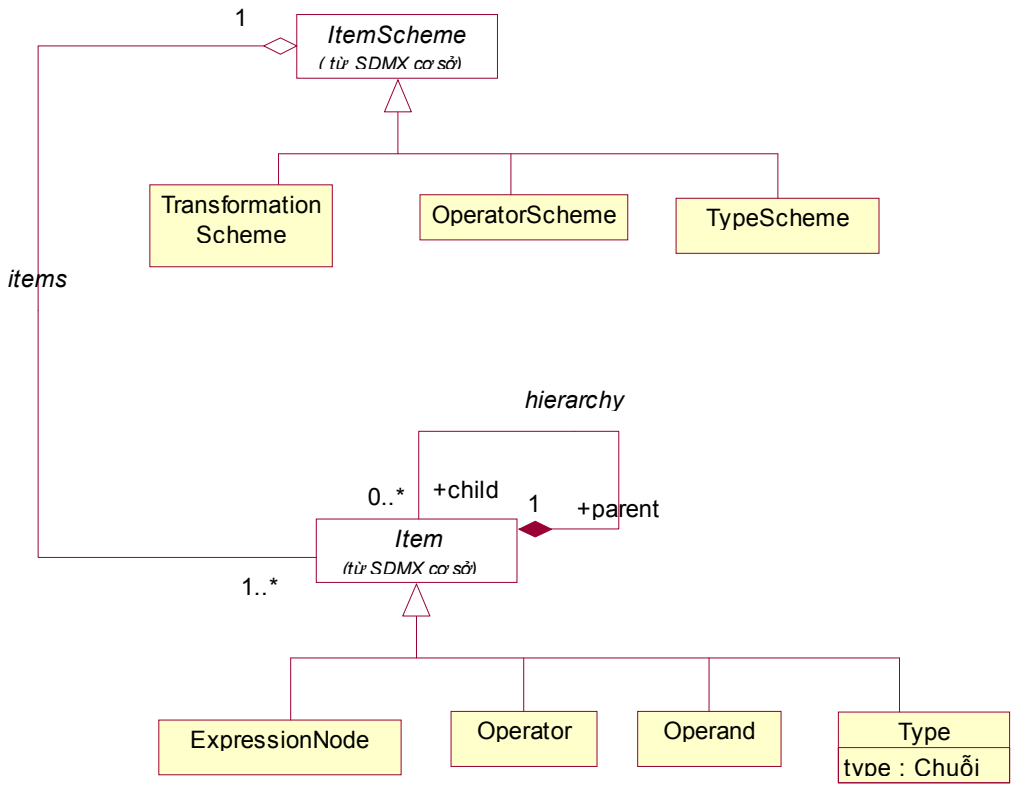
trúc hỗ trợ biểu thức cú pháp trung tính "ngữ pháp" có thể quy định các chức năng ở mức hình hột để một chương trình có thể "đọc" siêu dữ liệu và soạn chức năng được yêu cầu ở mọi ngôn ngữ máy tình phù hợp.

Cần chú ý rằng mô hình được biểu diễn ở trên tương tự với phạm vi và nội dung của siêu mô hình Biểu thức trong Kho Siêu Mô hình Chung (CWM) được xây dựng bởi Nhóm Quản lý Đối tượng (OMG). Đặc tả này có thể được tìm thấy ở: http://www.omg.org/cwm

Siêu mô hình Biểu thức được mô tả ở Điều 8.5 Phần 1 của đặc tả về CWM. Sơ đồ lớp được biểu diễn dưới đây là một giải thích siêu mô hình Biểu thức CWM được trình bày trong các lớp cơ sở của SDMX-IM.

12.2 Tổng quan về kế thừa mô hình

12.2.1 Sơ đồ lớp



Hình 43 – Sơ đồ lớp về tính kế thừa của các lớp biến đổi

12.2.2 Giải thích sơ đồ

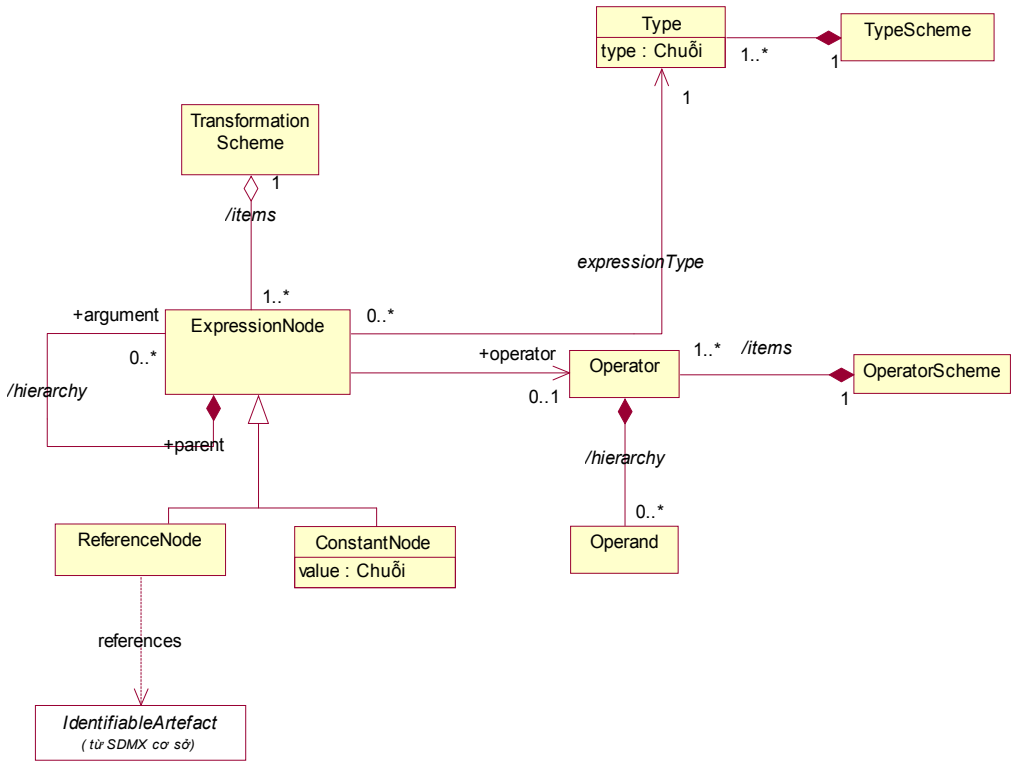
12.2.2.1 Diễn giải

Có ba kiểu *ItemScheme* liên quan tới mô hình này.

1. TransformationScheme bao gồm một hoặc nhiều ExpressionNode.
2. OperatorScheme bao gồm một hoặc nhiều *Operator* mà các Operand của nó là các *Item* con của Operator.
3. Lược đồ Type bao gồm các Type, biểu diễn kết quả của biểu thức.

12.3 Quan niệm về quan hệ - Mô hình

12.3.1 Sơ đồ lớp



Hình 44 – Sơ đồ lớp về quan hệ của các biểu thức

12.3.2 Giải thích sơ đồ

12.3.2.1 Diễn giải

Mô hình được thể hiện ở đây là một khung tổng quát sử dụng cho các biểu thức và các phép biến đổi, nhưng yêu cầu nhiều công đoạn để hợp nhất mô hình và việc sử dụng thực tế trong mô hình. Đây sẽ là một ấn bản tiêu chuẩn trong tương lai và có thể yêu cầu sự hài hòa về nội dung của `OperatorScheme` và `TypeScheme`.

Khái niệm biểu thức trong SDMX-IM mang đến một cái nhìn tổng quan về hàm của các sơ đồ hình cây của biểu thức, dẫn đến khả năng vài biểu thức quan hệ biểu diễn một dải các biểu thức. Mỗi hàm, toán tử hoặc toán hạng xuất hiện trong một biểu thức được biểu diễn bởi `+operator` vai trò liên kết với `Operator` (có các mục con là các `Operand`). Ví dụ phép cộng số học “a+b” được hiểu bởi hàm “sum (a+b)”. “Sum” là `Operator` và a, b là các `Operand`. Các ngữ nghĩa thực tế của hàm hoặc thao tác cụ thể được phó mặc các công cụ thực hiện cụ thể và không bị SDMX-IM giữ lại.

Tính chất phân cấp về việc biểu diễn các biểu thức của SDMX-IM đạt được bởi tính chất đệ quy của liên kết `ExpressionNode`. Liên kết này cho phép các hệ thống phân cấp phụ trong biểu thức được xử lý như các thông số thực của các nút cha.

Mô hình được sử dụng để xác định các việc kế thừa dữ liệu và xác định việc kiểm tra tính toàn vẹn (ví dụ: Tổng của A+B phải bằng C).

Định dạnh về kết quả của biểu thức (ví dụ: sự biểu diễn) được hỗ trợ bởi liên kết với `Type` xác định trong lược đồ gồm nhiều kiểu. Mặc dù mô hình xác định các cấu trúc dữ liệu được sử dụng để chứa biểu thức cú pháp trung tính, bản thân mô hình không quy định ngữ pháp của biểu thức cú pháp trung tính. Hàm có thể được mô tả trong dạng văn bản như một sự giải thích ko có cấu trúc của hàm hoặc ngôn ngữ chính thức hơn như BNF2. Định nghĩa hoặc mô tả nguyên bản được hỗ trợ bởi `ExpressionNode` là *`VersionableArtefact`* (khi nó kế thừa từ *`Item`*), do đó có thể có nhiều mô tả.

Các cấu trúc dữ liệu hoạt động như sau:

Các hàm toán học thực tế cần được biểu diễn (ví dụ: tổng, nhân, chia, gán (=, <, >)v.v) và các thông số của chúng được xác định trong `OperatorScheme` bao gồm một hoặc nhiều `Operator`,mỗi `Operator` là một toán tử trong đó các `Operand` của nó là các mục con của `Operator`, cùng với `Operator` xác định các nội dung của biểu thức.

Các biểu thức được xác định trong `TransformationScheme` bao gồm các `ExpressionNode`.

`ExpressionNode` tham chiếu `Operator` trong `OperatorScheme`. Số lượng các `Operand` con mà `Operator` có, xác định số lượng và thứ tự của các thông số chính thức mà `Operator` mang đến. Khi `ExpressionNode` đề cập tới `Operator`, nó phải xác định các `ExpressionNode` tương ứng với mỗi thông số của các `Operand` trong một trình tự đúng. Các thông số chính thức và các đối số tương ứng có thể là các kết cấu tổng ví dụ như định nghĩa khóa nhiều chiều kích thước mà có ngữ nghĩa về `KeyDescriptor` (of `KeyFamily`).

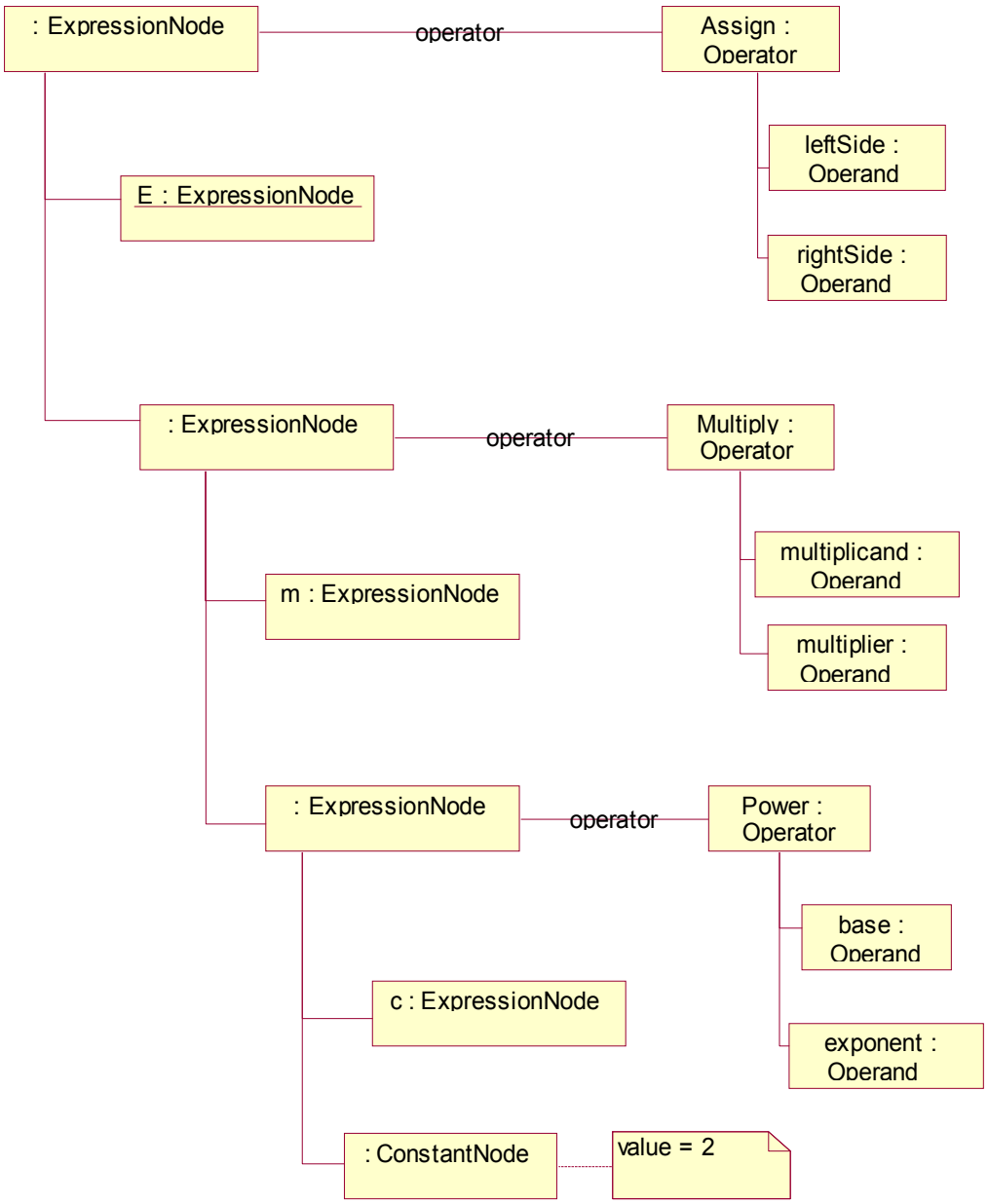
`ExpressionNode` (con) có thể có thêm các `ExpressionNode` xác định (đệ quy), mỗi `ExpressionNode` có thể là `Constant` hoặc là một tham chiếu tới *`IdentifiableArtefact`* (`ReferenceNode`) hoặc `ExpressionNode` khác. Tất cả *`IdentifiableArtefact`* trong SDMX-IM có urn duy nhất bao gồm các giá trị của các đối tượng riêng mà định danh nó. Cấu trúc của urn này được xác định trong Đặc tả Sổ đăng ký. Một ví dụ về urn của mã bao gồm cơ quan: code-list-id.code-id – ví dụ thực tế là "urn:sdmx:org.sdmx.infomodel.codelist.Code=TFFS:CL_AREA.1A".

Chú ý rằng nó có khả năng sử dụng `ReferenceNode` để định danh một khóa hoàn thiện của giá trị đo lường (bằng cách tham chiếu `KeySet` cụ thể quy định trong *`Constraint`*(xem đoạn sau)).

Sơ đồ sau đây chỉ ra việc biểu diễn một sơ đồ biểu thức hình cây của SDMX-IM về phương trình Einstein đã được biết đến E = mc2. Để hiểu rõ hơn cách phương trình được ánh xạ thành sơ đồ hình cây, công thức có thể được viết lại trong một ký pháp hàm như sau:

Gán (E, Multiply(m, Power(c, 2)))

Phương trình này được ánh xạ thành một tập các trường hợp `ExpressionNode` được biểu diễn ở hình sau:



Hình 45 – Sơ đồ hợp tác biểu diễn biểu thức E=mc2

12.3.2.2 Định nghĩa

Lớp	Đặc trưng	Mô tả
`Transformation Scheme`	kế thừa từ *`ItemScheme`*	Một lược đồ xác định hoặc tài liệu hóa các phép biến đổi được yêu cầu để kế thừa hoặc làm hợp lệ dữ liệu từ dữ liệu khác.
`OperatorScheme`	kế thừa từ *`ItemScheme`*	Mô hình xác định các toán tử và các toán hạng.

ExpressionNode	kế thừa từ *Item*	Nút trong hệ thống phân cấp gồm nhiều nút trong đó cùng xác định hoặc tài liệu hóa một biểu thức.
	expressionType	Liên kết với một Kiểu trong đó định danh định dạng kết quả của biểu thức.
	+operator	Liên kết với Toán tử và các Toán hạng con của nó trong đó xác định toán tử của Nút Biểu thức.
	+arguments	Các đối số toán học của Nút Biểu thức.
Constant	kế thừa từ ExpressionNode	Kiểu cụ thể của Nút Biểu thức chứa một giá trị không thay đổi.
ReferenceNode	kế thừa từ ExpressionNode	Kiểu cụ thể của Nút Biểu thức mà tham chiếu một đối tượng cụ thể.
	references	Liên kết với sản phẩm có thể định danh trong đó nó là đối tượng tham chiếu.
Operator		Toán tử trong Lược đồ Toán tử.
	/hierarchy	Liên kết với các Toán hạng của Toán tử.
Operand		Toán hạng trong Lược đồ Toán tử.

13 Phụ lục 1: Hướng dẫn về UML theo mô hình thông tin SDMX

13.1 Phạm vi

Phạm vi của tiêu chuẩn này đưa ra một cái nhìn tổng quan về ký pháp sơ đồ được sử dụng trong UML. Các ví dụ sử dụng trong tiêu chuẩn này được lấy từ mô hình SDMX UML.

13.2 Trường hợp sử dụng

Để xây dựng các mô hình dữ liệu, điều cần thiết là phải hiểu các chức năng yêu cầu được hỗ trợ. Chúng được xác định trong một mô hình trường hợp sử dụng. Mô hình trường hợp sử dụng bao gồm các tác nhân và các trường hợp sử dụng, chúng được xác định dưới đây:

Tác nhân có thể được định nghĩa như sau:

"*Một tác nhân xác định một tập các vai trò rõ ràng của người sử dụng hệ thống khi tương tác với nó. Một tác nhân có thể được thể hiện là một cá nhân hoặc một hệ thống ngoài.*"

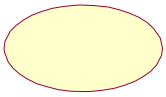
Một tác nhân được mô tả như hình dưới đây:

Nhà công bố dữ liệu

Hình 46 – Tác nhân

Trường hợp sử dụng có thể được định nghĩa như sau:

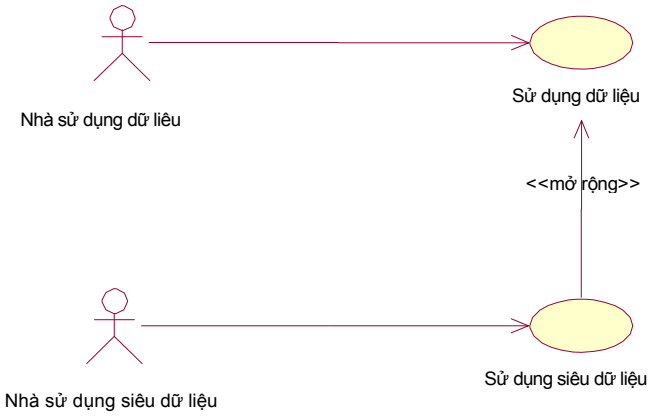
"Một Trường hợp sử dụng xác định một tập các trường hợp các trường hợp sử dụng, trong đó mỗi trường hợp là các hoạt động trình tự được thực hiện bởi hệ thống để đạt được một kết quả có giá trị cho một tác nhân nào đó"



Công bố dữ liệu

Hình 47 – Trường hợp sử dụng

Hình 48 – Tác nhân và trường hợp sử dụng



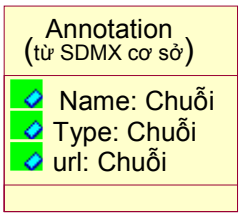
Hình 49 – Các trường hợp sử dụng mở rộng

Trường hợp sử dụng mở rộng là trường hợp sử dụng có thể được mở rộng tùy ý bởi trường hợp sử dụng độc lập với việc sử dụng trường hợp sử dụng. Mũi tên ở các điểm liên kết đóng vai trò sở hữu trường hợp sử dụng mở rộng. Trong ví dụ trên trường hợp sử dụng “Sử dụng dữ liệu” được mở rộng bởi trường hợp sử dụng “Sử dụng siêu dữ liệu”

13.3 Lớp và thuộc tính

13.3.1 Tổng quát

Lớp là vấn đề được người sử dụng quan tâm. Tên gọi tương đương trong mô hình quan hệ thực thể (E-R model) là thực thể và thuộc tính. Thực tế, nếu UML được sử dụng như các phương tiện của dữ liệu mô hình hóa, thì có một sự khác biệt nhỏ giữa một lớp và một thực thể.



Hình 50 – Lớp và các thuộc tính của nó

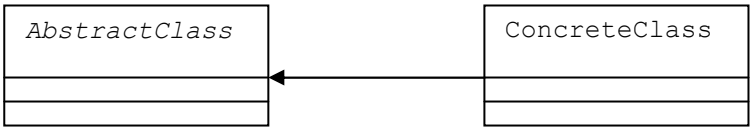
Hình 50 chỉ ra một lớp được biểu diễn bởi hình chữ nhật chia làm ba phần. Phần trên cùng là tên lớp,

phần thứ hai là các thuộc tính và phần cuối cùng là các thao tác. Chỉ có phần đầu tiên là bắt buộc. Tên của lớp là `Annotation` và nó thuộc về gói SDMX cơ sở. `Annotation` Thông thường để nhóm các sản phẩm liên quan (các lớp, các trường hợp sử dụng, v.v) trong các gói. `Annotation` có ba thuộc tính "Chuỗi" - `name`, `type`, and `url`. Định danh đầy đủ của thuộc tính bao gồm các lớp của nó, ví dụ: tên thuộc tính là `Annotation.name`.

Theo quy ước tên lớp sử dụng `UpperCamelCase` – các từ được ràng buộc với nhau và chữ cái đầu của mỗi từ được viết hoa. Thuộc tính sử dụng `lowerCamelCase` – chữ cái đầu tiên của từ đầu tiên không được viết hoa, các từ còn lại viết hoa chữ đầu tiên.

13.3.2 Lớp trừu tượng

Lớp trừu tượng là một cách hữu ích của việc nhóm các lớp, tránh đưa ra sơ đồ phức tạp với nhiều dòng liên kết, nhưng nó không biết trước được rằng lớp còn đáp ứng được các mục đích khác (ví dụ: lớp trừu tượng luôn được thực hiện như một trong các lớp con của nó). Trong sơ đồ này lớp trừu tượng được mô tả với tên dưới dạng in nghiêng và có mầu trắng.

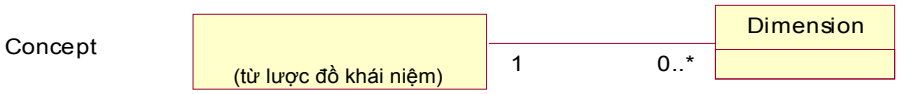


Hình 51 – Các lớp trừu tượng và các lớp cụ thể

13.4 Liên kết

13.4.1 Tổng quát

Trong một mô hình E-R, các liên kết được hiểu là các quan hệ. Mô hình UML có thể đưa ra nhiều ý nghĩa cho các liên kết hơn là được đưa ra trong quan hệ E-R. Ngoài ra, ký pháp UML là cố định (không thay đổi trong các liên kết). Trong sơ đồ E-R, có nhiều công nghệ sơ đồ hóa và nó là quan hệ trong một sơ đồ E-R trong đó có nhiều dạng, phụ thuộc vào ký pháp E-R cụ thể được sử dụng.



Hình 52 – Liên kết đơn giản

13.4.2 Liên kết đơn giản

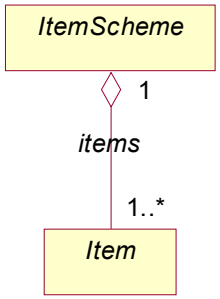
Lớp `Dimension` trong hình 52 có một liên kết với lớp `Concept`. Sơ đồ chỉ ra rằng `Dimension` có thể có một liên kết với `Concept` (1) và `Concept` có thể kết nối với nhiều `Dimension` (0..*). Đôi khi liên kết được đặt tên để đưa ra nhiều ngữ nghĩa hơn.

Trong UML, các liên kết có khả năng quy định nhiều quy tắc. Các quy tắc thông thường nhất:

- 0 hoặc 1 (0..1)
- 0 hoặc nhiều (0..*)
- 1 hoặc nhiều (1..*)
- Nhiều (*)
- Không quy định (bỏ trống)

13.4.3 Tập hợp

Tập hợp đơn giản

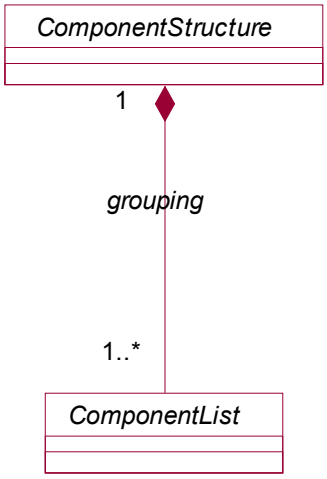


Hình 53 – Tập hợp đơn giản

Liên kết với một quan hệ tập hợp chỉ ra rằng một lớp là một lớp con (hoặc một phần) của lớp khác. Trong quan hệ tập hợp này, lớp con có thể nằm ngoài lớp cha của nó. Để biểu diễn quan hệ tập hợp, vẽ ra một đường kẻ liền từ lớp cha tới lớp con và một hình con rô rỗng trên liên kết cuối của lớp cha. Hình 53 chỉ ra ví dụ của quan hệ tập hợp giữa `ItemScheme` và `Item`.

Tập hợp ghép

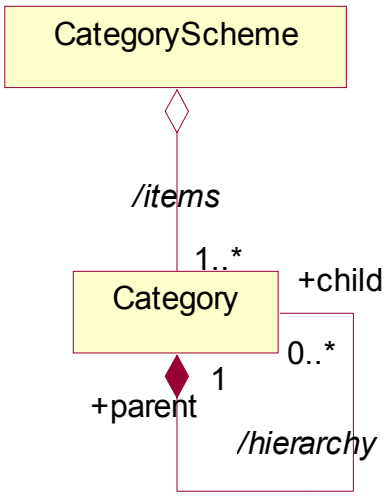
Quan hệ tập hợp ghép chỉ là dạng khác của quan hệ tập hợp, nhưng vòng đời của lớp con phụ thuộc vào vòng đời của lớp cha. Trong hình 54, chỉ ra quan hệ tập hợp ghép giữa lớp `ComponentStructure` và lớp `ComponentList`, để ý rằng quan hệ ghép được đưa ra giống với quan hệ tập hợp, nhưng ở đây hình con rô được phủ kín.



Hình 54 – Tập hợp ghép

13.4.4 Tên liên kết và các tên giới hạn liên kết

Điều này hữu ích cho việc đặt tên các liên kết khi đưa ra một vài ngữ nghĩa về mô hình, ví dụ: mục đích của việc liên kết. Hai lớp có khả năng nối lại với nhau bởi hai hoặc nhiều liên kết và trong trường hợp này các lớp này hữu ích cho việc đặt tên mục đích của liên kết. Hình 55 chỉ ra tập hợp đơn giản giữa `CategoryScheme` và `Category` được gọi là */items* (điều này có nghĩa là */items* được tạo từ liên kết giữa các lớp trên – trong trường hợp này giữa *`ItemScheme`* và *`Item và`* giữa `Category` được gọi là */hierarchy*.



Hình 55 – Các tên liên kết và các tên giới hạn liên kết

Hơn nữa, tên liên kết có khả năng đưa các tên vai trò cho các giới hạn liên kết nhằm mục đích cho ra nhiều ngữ nghĩa hơn - ví dụ như cha và con trong liên kết cấu trúc hình cây. Vai trò được chỉ ra với "+" đặt trước tên vai trò (ví dụ: sơ đồ về ngữ nghĩa của liên kết trong đó `Category` có thể có hoặc không có

`Category` cha và có hoặc có không nhiều `Category` con).

13.4.5 Khả năng điều hướng

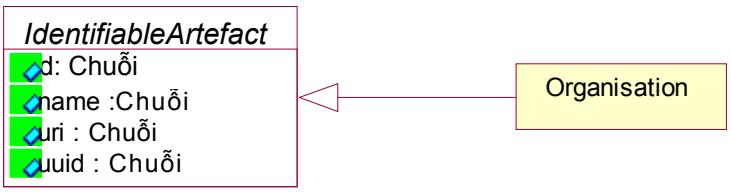
Các liên kết có khả năng điều hướng cả hai phía. Với mô hình dữ liệu thì không cần thiết đưa ra nhiều ngữ nghĩa hơn. Tuy nhiên, nếu có ý định thực hiện mô hình trong cơ sở dữ liệu hoặc cấu trúc thông điệp, thì liên kết này có lợi ích để đặt tên khi liên kết không có khả năng điều hướng (ví dụ: không có ý định hoặc sự cần thiết thực hiện một điều hướng theo một hướng riêng).

Hình 56 – Liên kết một chiều

Ở đây liên kết một chiều có khả năng điều hướng từ A sang B, nhưng không có yêu cầu (ví dụ: không có yêu cầu về chức năng) để điều hướng từ B sang A sử dụng liên kết này.

13.4.6 Kế thừa

Đôi khi tính kế thừa hữu ích cho việc nhóm các thuộc tính và liên kết thông thường với nhau trong một lớp trên. Điều này hữu ích nếu nhiều lớp sử dụng chung các liên kết với các lớp khác và có nhiều (nhưng không cần thiết phải tất cả) thuộc tính chung. Sự kế thừa được biểu diễn bằng một hình tam giác ở lớp trên.

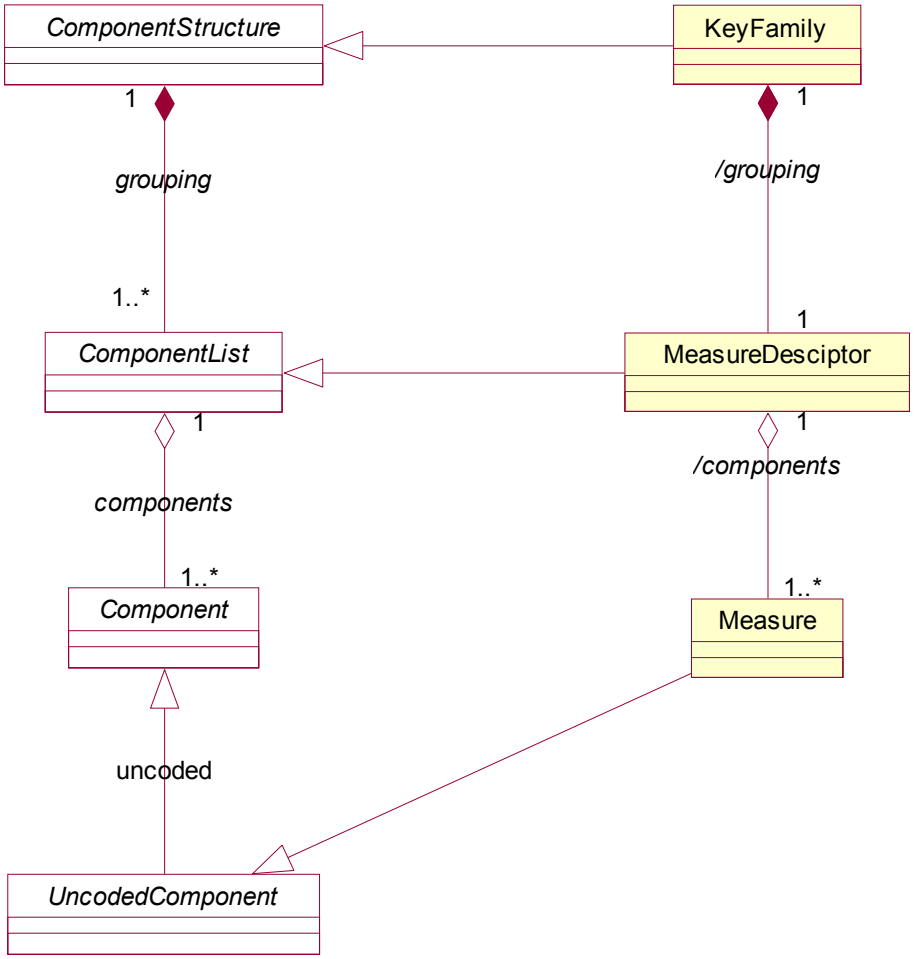


Hình 57– Kế thừa

Ở đây tổ chức được tạo từ *`IdentifiableArtefact`*, là lớp trên trừu tượng. Lớp này kế thừa từ các thuộc tính và liên kết của lớp trên. Lớp trên này có thể là lớp cụ thể (ví dụ: nó thực sự tồn tại) hoặc lớp trừu tượng.

13.4.7 Liên kết được tạo

Liên kết này thường hữu ích trong một sơ đồ quan hệ để chỉ ra các liên kết giữa các lớp con được tạo từ các liên kết của các lớp trên mà các lớp con kế thừa. Các liên kết được tạo được biểu diễn bởi dấu "/" ở trước tên liên kết ví dụ. / *`name`*.

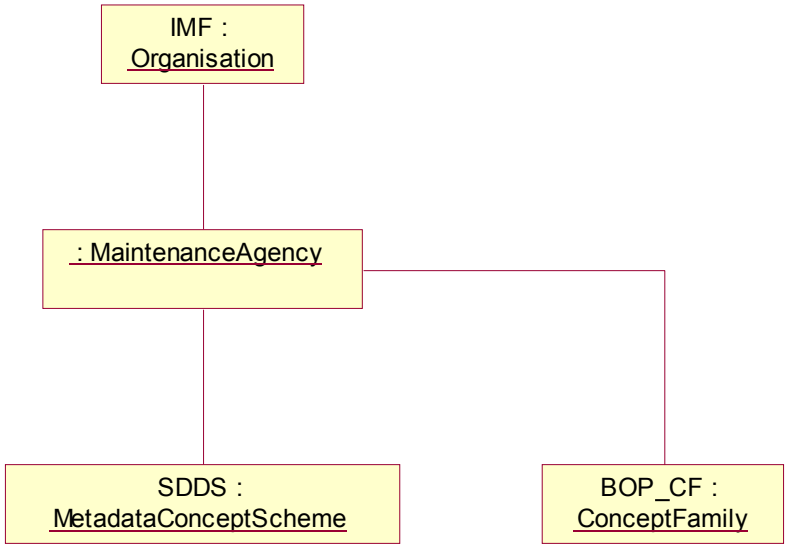


Hình 58 – Các liên kết được tạo

Chú ý rằng vô số các giới hạn liên kết được tạo ra hạn chế hơn trong liên kết được tạo. Trong ví dụ trên các liên kết *grouping* là 1..* trong khi đó liên kết */grouping* là 1.

13.5 Sơ đồ hợp tác

Sơ đồ hợp tác biểu diễn một ví dụ về các lớp (trường hợp lớp được gọi là đối tượng). Trường hợp về lớp có một tên duy nhất.



Hình 59 – Sơ đồ hợp tác

Ở đây có đối tượng của lớp `Organisation` gọi là IMF. Trong vai trò của `MaintenanceAgency` khi IMF duy trì `MetadataConceptScheme` gọi là SDDS và `ConceptFamily` gọi là BOP_CF.

Đôi khi sơ đồ hợp tác không hữu ích cho việc đưa ra tên cho đối tượng. Ở đây đối tượng vẫn là một trường hợp của lớp (ví dụ: `MaintenanceAgency`) nhưng không có tên – vì vậy đối tượng có nghĩa là "bất kỳ" hoặc nó không có ý nghĩa với một tên"

Các đối tượng được nối lại với nhau sử dụng một kết nối đối tượng.

14 Phụ lục 2: Tập khóa – Hướng dẫn

14.1 Giới thiệu

Tiêu chuẩn này giải thích các “tập khóa” cho người chưa quen với khái niệm này. Các tập khóa là một phần quan trọng của bộ tiêu chuẩn SDMX về việc trao đổi dữ liệu thống kê và chúng được mô hình hóa và giải thích chi tiết hơn trong các tài liệu khác. Tuy nhiên, các tài liệu đó không nhằm mục đích giải thích các điều cơ bản, điều đó sẽ tạo khó khăn cho người chưa quen với khái niệm này. Tiêu chuẩn này cung cấp một hướng dẫn cơ bản giúp cho người đọc hiểu được bộ tiêu chuẩn này.

14.2 Khái niệm tập khóa

Để trả lời được câu hỏi này, chúng ta cần xem lại các dữ liệu thống kê. Dữ liệu thống kê được biểu diễn với các số, ví dụ:

17369

Nếu thể hiện bằng số như trên – không hiểu được số đó biểu diễn gì. Bạn nên biết rằng nó là một phần của dữ liệu thống kê, do đó là đo lường của một vài hiện tượng – cũng được hiểu là một “quan sát” – nhưng bạn không thể tự khẳng định nó là đo lường của cái gì. Trong đầu bạn xuất hiện ngay các câu hỏi sau:

- chủ đề đo?
- Đơn vị đo?
- quốc gia hoặc khu vực áp dụng?
- Thời điểm tiến hành đó?

Danh sách các câu hỏi là không giới hạn. Đằng sau mỗi câu hỏi này là một khái niệm cụ thể, được sử dụng để mô tả dữ liệu. Trong các câu hỏi ở trên, các khái niệm mô tả là chủ đề, Đơn vị của phép đo, Quốc gia và Thời gian. Nếu tôi nói với bạn các câu trả lời của câu hỏi này, dữ liệu sẽ bắt đầu tạo ra:

- chủ đề là "tổng dân số"
- Đơn vị của đo lường "Hàng nghìn người"
- Quốc gia là "Quốc gia ABC"
- Thời gian là "1 /1/ 2001"

Đây là một ví dụ hư cấu và được đơn giản hóa, nhưng không giải thích cách chúng ta nắm bắt dữ liệu thống kê như thế nào với một tập các khái niệm mô tả. Giờ đây chúng ta biết rằng con số biểu diễn tổng dân số của Quốc gia ABC vào ngày 1 tháng 1 năm 2001, là 17,369,000.

Giải thích đơn giản nhất của một tập khóa: tập khóa là một tập các khái niệm mô tả, liên kết với tập dữ liệu, cho phép chúng ta hiểu được dữ liệu là gì.

14.3 Dữ liệu nhóm

Các số thường được nhóm với nhau theo nhiều cách khác nhau, đáp ứng các gói thông tin hữu ích. Phương pháp tiếp cận thông thường nhất là tập các quan sát – được hiểu như “chuỗi” hoặc “chuỗi thời

gian ” – được tạo ra trên một khoảng thời gian. Điều này cho phép chúng ta thấy được các xu hướng về hiện tượng đang được tính toán. Do đó, nếu tôi tính tổng dân số ở Quốc gia ABC vào ngày 1 tháng 1 năm mỗi năm, tôi có thể thấy dân số tăng hoặc giảm. Chuỗi thời gian luôn luôn có một “tần suất”. Đây là một khái niệm mô tả, trong đó mô tả các khoảng thời gian giữa các quan sát. Thông thường, đây là khoảng thời gian chuẩn mực, để tần suất có thể được trình bày như “hàng năm” hoặc “hàng tháng” hoặc “hàng tuần”. Đôi khi, các khoảng thời gian là không chuẩn mực. Chú ý rằng một quan sát đơn lẻ không có tần suất - chỉ có chuỗi các quan sát mới có tần suất. Tần suất là một ví dụ của khái niệm mô tả mà chỉ áp dụng cho chuỗi dữ liệu.

Có một số nhóm dữ liệu ở mức cao hơn. Số lượng chuỗi thường được nhóm với nhau thành một “Nhóm”. Nói theo cách truyền thống, Nhóm được hiểu là “Sibling Group” và chứa tập Chuỗi không kể chúng được tính với các các tần suất khác nhau. Do đó, hiện tượng được đưa ra sẽ được tính như: hàng ngày, hàng tháng và hàng năm, các Chuỗi này được lấy ra cùng với nhau, gọi là “Sibling Group”.

Có khả năng có các Nhóm trong đó có các giá trị thay đổi về các khái niệm mô tả hơn là tần suất, tuy nhiên: nếu tôi muốn trình bày một tỉ giá US hàng ngày về tiền tệ thế giới thông qua một năm, tôi có một loại nhóm khác nhau. Tất cả mô tả “tần suất” sẽ là giống nhau – “hàng ngày” – nhưng khái niệm mô tả đưa ra “ngoại tệ” sẽ khác nhau với mỗi chuỗi.

Cũng có một mức gói cao hơn được hiểu là “Tập dữ liệu”. Gói này biểu diễn tập dữ liệu trong đó có thể bao gồm một vài nhóm. Điển hình, nó được duy trì và công bố bởi một cơ quan, để trở thành một nguồn dữ liệu thống kê được biết đến.

Cấu trúc cơ bản: Có các quan sát được nhóm thành chuỗi, thành các nhóm và thành các tập dữ liệu.

CHÚ THÍCH Có một cách đóng gói các quan sát khác, mà chúng ta gọi là dữ liệu “phần giao”. Trong dữ liệu phần giao, số lượng lớn các quan sát liên quan được thể hiện cho điểm riêng hoặc khoảng thời gian. Tổ chức dữ liệu này tương tự với dữ liệu Chuỗi Thời gian theo cách một tập các khái niệm mô tả có thể được liên kết với nó. Tập khóa có thể được sử dụng để mô tả cả dữ liệu phần giao và dữ liệu chuỗi thời gian. Với mục đích của phần hướng dẫn này, chúng ta sẽ tập trung vào dữ liệu chuỗi thời gian. Chúng ta mô tả tập khóa về dữ liệu chuỗi thời gian, sau đó quay lại và tìm hiểu dữ liệu phần giao được tổ chức như thế nào.

Tập khóa là gì? (Câu trả lời thứ 1)

Tập khóa là cách liên kết một tập các khái niệm mô tả với tập các dữ liệu thống kê cụ thể, cũng như kỹ thuật đóng gói hoặc tổ chức tập dữ liệu đó thành các nhóm hoặc các nhóm nhỏ. Đây chỉ đơn thuần là một cách hiểu cấu trúc và ý nghĩa các dữ liệu thống kê, nhưng cách này cung cấp cho chúng ta một mô hình chung.

14.4 Mức đính kèm

Một vài khái niệm mô tả không có ý nghĩa ở mức quan sát, nhưng lại có ý nghĩa ở mức cao hơn. Một ví dụ: tần suất, không có ý nghĩa đơn, nhưng có ý nghĩa khi áp dụng cho chuỗi các quan sát. Bởi vì nó biểu diễn khoảng thời gian giữa các quan sát. Thời gian, nói cách khác, có ý nghĩa ở mức quan sát – mỗi quan sát được liên kết với một thời điểm hoặc một khoảngThời gian cụ thể. Các tập khóa cung cấp thông tin về mức trong đó một khái niệm mô tả riêng được đính kèm: ở mức quan sát, mức chuỗi, mức nhóm

hoặc mức tập dữ liệu. Điều này được hiểu như “mức đính kèm”của khái niệm mô tả.

Nếu chúng ta đề cập tới các Nhóm, đặc biệt, chúng ta muốn biết nó làm việc thế nào. Trong một nhóm, một vài khái niệm mô tả có các giá trị giống với tất cả các chuỗi trong Nhóm, trong khi các khái niệm mô tả khác luôn thay đổi. Đối với nhóm được mô tả ở trên, toàn bộ tỉ giá hối đoái ở Mỹ hàng ngày về tiền tệ thế giới, các khái niệm mô tả của chủ đề (“tỉ giá hối đoái ở Mỹ”) và tần suất (“hàng ngày”) sẽ giống với các thành phần của nhóm. Khái niệm mô tả “ngoại tệ”, tuy nhiên, sẽ thay đổi mỗi chuỗi trong nhóm: sẽ có một chuỗi về “Francs Thụy sĩ, “Chuỗi về “châu Âu,” Chuỗi về “đô la New Zealand” v.v…

Quy tắc: các khái niệm mô tả được “đính kèm” với mức nhóm trong đó chúng trở thành biến số. Do đó, nếu trong tập dữ liệu đơn, tất cả nội dung của Chuỗi chia sẻ giá trị đơn về khái niệm mô tả thì khái niệm mô tả đó nên được đính kèm ở mức chuỗi. Quy tắc này cũng giả thiết một điều: mức được lựa chọn là mức có cấu trúc cao nhất trong đó tất cả các nhóm phụ sẽ chia sẻ cùng một giá trị. (Trong khi thực tế là tất cả chuỗi trong Nhóm trong đó quốc gia “Thụy sĩ” chia sẻ một giá trị đơn, nếu mỗi Nhóm trong Tập dữ liệu luôn có giá trị “Thụy sĩ” với quốc gia, thì mức đính kèm là tập dữ liệu, chứ không phải là Nhóm.)

Các mức đính kèm của các khái niệm mô tả luôn luôn là các mức ít nhất mà khái niệm có ý nghĩa: đo đó, bạn không thể đính kèm tần suất khái niệm mô tả ở mức quan sát, bởi vì khi là một khái niệm thì chỉ điều hành ở mức chuỗi (với nhiều Quan sát tạo ra trên một khoảng thời gian).

14.5 Khóa

Tên gọi "tập khóa" được tạo ra bởi thuật ngữ "khóa". "Khóa" đề cập đến các giá trị về khái niệm mô tả mà mô tả và định danh một tập dữ liệu riêng. Xem ví dụ sau:

Có một tập dữ liệu thống kê sử dụng các khái niệm mô tả sau đây:

- Thời gian
- Tần suất
- chủ đề
- Quốc gia

Thời gian luôn được gắn ở mức quan sát – giá trị về Thời gian là thời điểm tạo ra Quan sát đó. Thời gian là một khái niệm đối với tất cả dữ liệu thống kê – không cấu thành một đoạn của khóa. Các khái niệm mô tả khác – tần suất, chủ đề và quốc gia – tất cả được đính kèm ở mức chuỗi. Với bất kỳ Chuỗi Quan sát được đưa ra, chúng sẽ có tất cả giá trị đơn.

Nếu Chuỗi dữ liệu là phép tính tổng dân số hàng tháng của Quốc gia ABC, thì chúng ta sẽ có một khóa bao gồm các giá trị về mỗi khái niệm mô tả sau đây:

- Tần suất = "hàng tháng"
- chủ đề = "tổng dân số"
- Quốc gia = "Quốc gia ABC"

Tập các giá trị này – “Hàng tháng” – tổng dân số - Quốc gia ABC “ là “khóa” về Chuỗi dữ liệu này: nó định danh dữ liệu gì.

Các khóa được liên kết hầu hết với dữ liệu ở mức chuỗi, nhưng chúng cũng tồn tại ở các mức khác. Ví dụ, chúng ta có thể mở rộng ví dụ như sau: một nhóm bao gồm dữ liệu về tổng số dân của các quốc gia trên thế giới được thống kê hàng tháng. Ở mức nhóm, tần suất có giá trị "hàng tháng" và chủ đề có giá trị "tổng dân số", nhưng không quy định khái niệm mô tả quốc gia, bởi vì nó thay đổi từ chuỗi này sang chuỗi khác. Đối với nhóm khóa được hiểu là một "Khóa mật mã"- nó định danh nhóm, hơn là định danh chuỗi. (Nhằm hiểu đầy đủ nhóm, tuy nhiên, chúng ta cũng cần biết các khái niệm mô tả thay đổi thế nào – trong trường hợp quốc gia.)

Các giá trị khóa được đính kèm ở mức chuỗi và được đưa ra trong trình tự cố định. Tần suất là một khái niệm mô tả đầu tiên và các khái niệm khác được gán theo thứ tự cho tập dữ liệu riêng đó. Điều này giúp cho việc chia sẻ và hiểu dữ liệu thống kê dễ dàng hơn nhiều.

Nếu bạn nhìn lại cách sử dụng ban đầu, bạn sẽ thấy rằng chúng ta đang thảo luận về khái niệm mô tả "Đơn vị đo". Bởi "khóa" chỉ chứa các giá trị về các khái niệm mô tả mà định danh dữ liệu. Nếu chúng ta có các đo lường hàng nghìn hoặc hàng triệu, dữ liệu cũng giống như vậy – chúng có thể tạo từ một dữ liệu khác bằng cách tăng gấp bội số lượng dữ liệu bởi một thừa số biến đổi thích hợp.

Điều này chỉ ra một khác biệt lớn giữa hai kiểu khái niệm mô tả: một kiểu định danh và mô tả dữ liệu gọi là "các chiều kích thước", còn kiểu kia cái chỉ mô tả gọi là "các thuộc tính". Không chỉ "các chiều kích thước" mà các khái niệm mô tả cũng định danh dữ liệu – được sử dụng trong "khóa", bởi vì "khóa" là cách định danh cơ bản một tập dữ liệu.

14.6 Danh sách mã và các biểu diễn khác

Để trao đổi và hiểu dữ liệu, một tập khóa cho biết các giá trị có khả năng xảy ra cho mỗi chiều kích thước. Danh sách các giá trị có khả năng xảy ra này được hiểu như một "danh sách mã". Mỗi giá trị trong đó danh sách được đưa ra chữ viết tắt của một ngôn ngữ nào đó – " mã " và mô tả cụ thể về ngôn ngữ nào đó. Điều này giúp chúng ta tránh được các vấn đề về thông dịch trong việc mô tả dữ liệu: mã có thể được dịch thành các mô tả ở bất kỳ ngôn ngữ nào mà không phải thay đổi mã liên kết với dữ liệu đó. Bất cứ nơi nào có thể, các giá trị về danh sách mã được lấy ra từ các chuẩn quốc tế, như đã được ISO cung cấp về quốc gia và tiền tệ.

Các chiều kích thước luôn luôn được biểu diễn với các mã. Các thuộc tính đôi khi được biểu diễn với các mã, nhưng đôi khi được biểu diễn bởi số hoặc các giá trị văn bản tự do.

Điều này được cho phép bởi các thuộc tính không đáp ứng được một chức năng định danh, mà chỉ mô tả dữ liệu.

Tập khóa là gì? (Câu trả lời thứ 2)

Chúng ta có thêm một cách hiểu nữa về tập khóa: tập khóa quy định một tập các khái niệm mô tả và định danh tập dữ liệu. Tập khóa cho chúng ta thấy được các danh sách mã cung cấp các giá trị đối với các chiều kích thước như thế nào và các giá trị đối với các thuộc tính, các danh sách mã, số hoặc các trường văn bản tự do.

14.7 Cấu trúc dữ liệu phần giao

Đưa ra giải thích về các tập khóa, chúng ta phải hiểu rằng tp Khóa liên kết các khái niệm mô tả với dữ liệu, một vài trong số đó cũng đáp ứng việc định danh dữ liệu – các khái niệm “chiều kích thước” tạo ra Khóa.

Các cấu trúc dữ liệu phần giao không áp dụng các tập khái niệm khác nhau cho các dữ liệu: các khái niệm giống nhau vẫn áp dụng trong việc mô tả và định danh dữ liệu. Dữ liệu phần giao đính kèm các khái niệm với dữ liệu một cách khác biệt, để tạo ra các diễn dữ liệu khác nhau. Quay lại ví dụ trước, chúng ta có các khái niệm sau:

- Thời gian
- Tần suất
- chủ đề
- Quốc gia

Nếu bạn muốn lấy một tập dữ liệu được mô tả và định danh bởi tập các khái niệm này và thể hiện chúng theo dạng “phần giao”, chúng ta sẽ không thay đổi các khái niệm này – mà chỉ thay đổi cách chúng được biểu diễn – đó là đính kèm với cấu trúc dữ liệu.

Theo ví dụ trên, tổng dân số của mỗi quốc gia trên thế giới vào ngày 1, tháng 1 năm 2001 là tập dữ liệu. Trong ví dụ đó, chúng ta tính dân số của Quốc gia ABC thông qua giai đoạn các năm. Thời gian là khái niệm chúng ta sử dụng để tổ chức dữ liệu trong một trình tự các quan sát.

Nếu chúng ta tổ chức dữ liệu để phản ánh chỉ một điểm thời gian đơn lẻ - trong trường hợp này, ngày 1 tháng 1, 2001 – tổ chức dữ liệu trên một khoảng thời gian sẽ làm giảm ý nghĩa. Đó vẫn là cách tổ chức dữ liệu, nhưng chúng ta có thể coi như một phần giao.

Đề cập tới thuật ngữ “phần giao” – nó được hiểu là một nhóm chuỗi song song diễn ra trong khoảng thời gian, trong đó một phần được lấy ra. Vì thế, phần giao được tạo ra trong thời điểm đó.

Trong ví dụ, chúng ta dễ dàng nhận thấy cách nó áp dụng: thay vì tổ chức dữ liệu trên một khoảng thời gian – đó là sử dụng khái niệm thời gian- chúng ta chọn cách tổ chức dữ liệu thông qua khái niệm Quốc gia. Do đó, thay vì có một dữ liệu đơn về Tần suất, chủ đề và Quốc gia về toàn bộ Quan sát trong chuỗi, với một giá trị thời gian liên kết với mỗi quan sát, chúng ta sẽ có một giá trị quốc gia liên kết với mỗi quan sát và giá trị đơn về Tần suất, chủ đề và Thời gian. Thay vì việc gọi nhóm các quan sát “Chuỗi”, hiện tại chúng ta sử dụng thuật ngữ “Phần”

Ở ví dụ trước, chúng ta có một khóa tồn tại ở hầu hết mức chuỗi:

Tần suất = "hàng tháng"

chủ đề = "tổng dân số"

Quốc gia = "Quốc gia ABC"

Thời gian – khái niệm còn lại, được liên kết với các quan sát, với giá trị khác nhau. Do đó, có thể có một chuỗi như sau:

1 tháng 1, 2001 – 17369

1 tháng 2, 2001 – 17370

1 tháng 3, 2001 – 17405

Với việc thể hiện phần giao, có thể có hầu hết khóa ở mức phần (hoặc ở mức nhóm):

Tần suất = "hàng tháng"

chủ đề = "tổng dân số"

Thời gian = "1 tháng 1, 2001"

Với mỗi quan sát, chúng ta có giá trị quốc gia, thay vì giá trị thời gian:

Quốc gia ABC = “17369”

Quốc gia XYZ = “24982”

Quốc gia HIJ = “37260”

Trong phần thể hiện “phần giao” về tập dữ liệu này, chúng ta chọn cách thể hiện mỗi quan sát ghép đối với giá trị quốc gia, được lấy từ Danh sách mã của các giá trị về khái niệm Quốc gia. Các chiều kích thước khác có thể dễ dàng đem lại một cách nhìn về phần giao, bằng cách đính kèm giá trị của chúng ở mức quan sát, thay vì giá trị về Quốc gia, như trong ví dụ.

Bởi vì bản thân các khái niệm không thay đổi, nên chỉ có một cách là chúng được đính kèm với cấu trúc dữ liệu, tập khóa đơn có thể được sử dụng để mô tả cả chuỗi thời gian lẫn sự thể hiện phần giao.

Trong phiên bản 1.0, các định dạng có khả năng thể hiện dữ liệu phần giao về các khái niệm chiều kích thước đơn lẻ, cũng giống việc thể hiện dữ liệu như một chuỗi thời gian. Nhà sáng tạo tập khóa có nhiệm vụ lựa chọn khái niệm phi thời gian, được sử dụng như một chiều kích thước để tổ chức việc biểu diễn phần giao. Các phiên bản tương lai có khả năng hỗ trợ phần giao hoàn thiện hơn cho tập khóa.