MÔ HÌNH HÓA TRI THỨC CHO MỘT CƠ SỞ DỮ LIỆU QUAN HỆ BẰNG ONTOLOGY WEB LANGUAGE

Huỳnh Tuấn Anh[a*]

[a]*Khoa Công nghệ Thông tin, Trường Đại học Nha Trang, Khánh Hòa, Việt Nam*

Lịch sử bài báo
Nhận ngày 10 tháng 01 năm 2017 | Chỉnh sửa ngày 10 tháng 04 năm 2017
Chấp nhận đăng ngày 11 tháng 05 năm 2017

Tóm tắt

Trong bài báo này, chúng tôi trình bày phương pháp mô hình hóa tri thức một cơ sở dữ liệu quan hệ bằng Ontology Web Language (OWL). Kết quả đạt được bao gồm các luật chuyển đổi dữ liệu từ cơ sở dữ liệu quan hệ sang Ontology và các Axiom bổ sung ngữ nghĩa cho một cơ sở dữ liệu quan hệ. Dựa trên các luật này, dữ liệu trong mô hình quan hệ có thể được chuyển đổi thành các bộ ba RDF/OWL cho các ứng dụng Sematic web.

Từ khóa: Mapping; Ontology; OWL; Relational Database; RDF; RDFS; Semantic web.

1. GIỚI THIỆU

Sematic web, hay còn gọi là Web 3.0, biểu diễn các trang web có nội dung mà máy tính có thể hiểu được. Trong Sematic web, dữ liệu được lưu trữ bằng các bộ ba RDF/OWL hay còn gọi là các Ontology. Các thông tin được lưu trữ bằng các Ontology được xem là một cơ sở dữ liệu có khả năng liên kết toàn cầu. OWL là một hình thức đặc tả và liên kết dữ liệu một cách có ngữ nghĩa để cho máy tính có thể hiểu và xử lý dữ liệu một cách tự động. Ngoài ra, dữ liệu của các ứng dụng Semantic web có thể được chia sẻ ở phạm vi toàn cầu. Dữ liệu của một ứng dụng Semantic web có thể được truy vấn từ nhiều nguồn và tích hợp lại với nhau một cách trực tiếp. Tuy nhiên, phần lớn dữ liệu của các ứng dụng trong thế hệ web hiện tại lại được lưu trữ trong các cơ sở dữ liệu quan hệ. Do đó, một bài toán quan trọng là tạo các Ontology từ dữ liệu web hiện có trong các cơ sở dữ liệu quan hệ.

* Tác giả liên hệ: Email: anhht@ntu.edu.vn

Trong bài báo này, tiếp tục phát triển nghiên cứu của Huỳnh (2015), chúng tôi trình bày và bổ sung, hoàn thiện các luật chuyển đổi từ cơ sở dữ liệu quan hệ sang các Ontology. Các bổ sung bao gồm các luật chuyển đổi một số mối kết hợp thành các thuộc tính *owl: TransitiveProperty*, luật chuyển đổi bảng dữ liệu kết hợp (bảng dữ liệu có các thành phần khóa chính là các khóa ngoại), luật chuyển đổi các bản ghi thành các Ontology. Bài báo có cấu trúc như sau: Mục 1 giới thiệu mở đầu, các nghiên cứu liên quan được trình bày ở Mục 2. Mục 3 trình bày các khái niệm về cơ sở dữ liệu quan hệ và OWL Ontology. Mục 4 trình bày các luật chuyển đổi một cơ sở dữ liệu quan hệ sang Ontology. Ví dụ minh họa các luật chuyển đổi được trình bày ở Mục 5. Phần đánh giá các luật được trình bày trong Mục 6.

2. CÁC NGHIÊN CỨU LIÊN QUAN

Ayoub, Mohamed, và Ilias (2015) đề xuất cách ánh xạ một cơ sở dữ liệu quan hệ tới một Ontology sẵn có mà vẫn giữ nguyên cấu trúc của cơ sở dữ liệu. Các bảng, các thuộc tính, khóa chính được ánh xạ thành các đối tượng của các lớp đặc biệt dùng để mô tả các đặc trưng của một cơ sở dữ liệu quan hệ và được lưu thành một tập tin lược đồ có tên "*Abstract.OWL*". Dữ liệu của cơ sở dữ liệu quan hệ sau đó được rút trích thành một tập tin RDF theo các qui tắc trong tập tin lược đồ. Ontology cần thiết được xây dựng bằng các mệnh đề "*CONSTRUCT*" khi thực hiện các truy vấn "*SPARQL*" từ dữ liệu RDF trung gian.

Raji và Nadine (2007); Sufeng, Haiyun, Mei, và Huaiwei (2010); và Mallede, Marir, và Vassilev (2013) đề xuất cách mô tả các cơ sở dữ liệu quan hệ thành các Ontology. Trước hết, cơ sở dữ liệu quan hệ được mô tả thành các Ontology. Các bảng được mô tả thành các lớp, các thuộc tính được mô tả thành các DataType Property. Các thuộc tính khóa ngoại được mô tả thành các Object Property có tính tương hỗ. Tuy nhiên, hầu hết các đề xuất chủ yếu chỉ chú trọng đến các mô tả các bảng, mối kết hợp của cơ sở dữ liệu, việc chuyển đổi các bộ dữ liệu chỉ đơn giản là chuyển mỗi bản ghi thành một đối tượng.

Thực tế, xây dựng các Ontology chính là việc mô hình hóa tri thức cho một lĩnh vực cụ thể, các Ontology phải được xây dựng sao cho chúng hỗ trợ việc suy luận trên các

tri thức đã có. Bên cạnh các lớp, thuộc tính mô tả cơ sở dữ liệu quan hệ, chúng ta cần phải bổ sung thêm các lớp hỗ trợ việc suy luận như: Suy luận bắc cầu, suy luận tương hỗ, suy luận xác định đối tượng... Việc chuyển đổi các đối tượng phải chú trọng đến các tri thức riêng của lĩnh vực được mô hình hóa chứ không đơn thuần là chuyển đổi mỗi bản ghi thành một đối tượng.

3. CƠ SỞ DỮ LIỆU VÀ ONTOLOGY

Trong phần này, chúng tôi sẽ giới thiệu một số khái niệm về cơ sở dữ liệu và OWL Ontology ở Mục 3.1 và 3.2. Dựa vào những đặc tính tương đồng của cả hai, chúng tôi trình bày và bổ sung một số luật chuyển đổi từ mô hình quan hệ sang Ontology trong Mục 4. Ngoài ra, các luật chuyển đổi các mối kết hợp thành các thuộc tính *owl:TransitiveProperty*, *owl:propertyChainAxiom*, *owl:inverseOf* cũng được bổ sung để hỗ trợ việc suy luận của các ứng dụng Semantic web trên các Ontology.

3.1. OWL Ontology

OWL là một ngôn ngữ mô hình hóa tri thức, được thiết kế để trình bày, trao đổi tri thức về một lĩnh vực cụ thể. OWL được xem là một ngôn ngữ đa năng mạnh mẽ để mô hình hóa các lĩnh vực nhất định của tri thức nhân loại. Kết quả của tiến trình mô hình hóa này là các Ontology - Là các thuật ngữ trong biểu diễn tri thức. Một số khái niệm cơ bản của OWL là:

- *Axioms*: Các mệnh đề mà một OWL Ontology biểu diễn. Một Axiom trong OWL luôn được đánh giá đúng.
- *Entities (Các thực thể):* Các phần tử được sử dụng để chỉ các đối tượng trong thế giới thực.
- *Expressions (Các biểu thức):* Sự kết hợp các thực thể để hình thành các biểu diễn phức tạp.
- *Class*: Còn được gọi là khái niệm (*concept*). Một lớp trong OWL được hiểu là một loại thực thể nào đó (Ví dụ: Sinh viên; Giáo viên; Môn học...). Các

lớp có thể được tổ chức theo các phân cấp thông qua việc định nghĩa các lớp con. Ví dụ: Lớp *Động_Vật* là lớp con của Lớp *Sinh_Vật*.

- *Properties (Các thuộc tính)*: Còn gọi là các mối kết hợp (*relations*) dùng để biểu diễn các đặc tính của các đối tượng (*object*) hay mối liên hệ giữa các đối tượng, ví dụ *John Kết_hôn Marry* hay *John Sinh_năm 1980*. Trong OWL, thuộc tính được chia làm hai loại: 1. *DataType Properties* dùng để gán các giá trị dữ liệu cho các đối tượng. 2. *Object Properties* dùng để biểu diễn mối kết hợp giữa các đối tượng. Trong OWL, khác với cơ sở dữ liệu và các ngôn ngữ lập trình hướng đối tượng, các thuộc tính được định nghĩa độc lập với các lớp. Khi chúng được sử dụng, các đối tượng sẽ được nhận biết thuộc về lớp nào dựa vào chủ thể (*domain*) và giá trị (*range*) của các thuộc tính.

- *Restriction (ràng buộc)*: Các Ontology mô tả các ràng buộc về giá trị (*range*) của các thuộc tính cũng như ràng buộc về chủ thể (*domain*) của các thuộc tính.

3.2. Cơ sở dữ liệu quan hệ

Cơ sở dữ liệu quan hệ là một mô hình dữ liệu dựa trên lý thuyết quan hệ. Một cơ sở dữ liệu được tổ chức thành các bảng, mỗi bảng bao gồm nhiều thuộc tính.

- *Bảng dữ liệu*: Tập hợp các bộ dữ liệu có cùng tập thuộc tính, mỗi bộ dữ liệu thường biểu diễn thông tin về một đối tượng.

- *Thuộc tính*: Dùng để mô tả các đặc tính của đối tượng. Mỗi thuộc tính phải có một kiểu dữ liệu gọi là domain. Thuộc tính trong cơ sở dữ liệu quan hệ có thể được chia thành các dạng: 1) Thuộc tính thông thường; 2) Thuộc tính là thành phần của khóa chính; 3) Thuộc tính khóa ngoại. Các thuộc tính dạng 1 và 2 có thể xem là các thuộc tính dạng *DataType Property* trong OWL, còn các thuộc tính dạng 3 tương đồng với các *Object Property* trong OWL.

- *Các ràng buộc (Constraint):* Các ràng buộc mà một cơ sở dữ liệu phải tuân theo để mô hình hóa dữ liệu cho một ứng dụng trong thực tế. Các ràng buộc

có thể bao gồm: Ràng buộc khóa; Ràng buộc toàn vẹn; Ràng buộc trên miền dữ liệu của các thuộc tính; Ràng buộc trên bộ dữ liệu; và các Trigger.

3.3. Một số định nghĩa và ký hiệu

3.3.1. Các ký hiệu

- Ω là tập các bảng trong cơ sở dữ liệu *D*. T là một bảng thuộc tập Ω.
- *PK(T)* là tập thuộc tính làm khóa chính trong bảng *T*.
- *FK(T)* là tập các thuộc tính khóa ngoại trong bảng *T*.
- *FK(T_i, T_j)* là một khóa ngoại, tên *FK,* của bảng T_j tham chiếu đến khóa chính của bảng T_j.
- *A(T)* là tập các thuộc tính không phải khóa chính hoặc khóa ngoại của bảng *T*. Ta ký hiệu *A* là một thuộc tính và *xsdIRI(A)* là *IRI* của kiểu dữ liệu *xsd* tương ứng với kiểu dữ liệu của thuộc tính A.
- *t(T)* là tập các bản ghi của bảng dữ liệu *T*; *t* là một bản ghi, *t.A* là giá trị của thuộc tính *A* trong bộ dữ liệu *t*.

3.3.2. Các định nghĩa

Định nghĩa 1 (*Bảng quan hệ nhị phân*): Một bảng T được gọi là bảng quan hệ nhị phân khi và chỉ khi *PK(T) = FK(T)* và *Card(FK(T))* = 2 và và *A(T) = ∅*.

Tập các bảng quan hệ nhị phân được ký hiệu là Ω_B.

Định nghĩa 2 (*Bảng chuyên biệt hóa*): Một bảng *T* được gọi bảng chuyên biệt hóa của bảng T_P khi và chỉ khi có một khóa ngoại *FK(T, T_P)* và khóa ngoại này cũng chính là khóa chính của *T*.

Ký hiệu bảng *T* là bảng chuyên biệt hóa của bảng T_P là: *T isa* T_P. Tập các bảng chuyên biệt hóa trong Ω được ký hiệu Ω_S.

Định nghĩa 3 (*Bảng kết hợp*): Một bảng *T* được gọi là bảng kết hợp nếu nó không phải là bảng quan hệ nhị phân và $FK(T) \subseteq PK(T)$ và $Card(FK(T) \geq 2$.

Tập các bảng kết hợp được ký hiệu là Ω_R.

Cơ sở dữ liệu quan hệ được đề cập đến trong bài báo này là một cơ sở dữ liệu quan hệ đạt chuẩn 3. Các bảng không phải bảng quan hệ nhị phân hoặc bảng kết hợp đều có khóa chính có số thuộc tính là một.

4. CÁC LUẬT CHUYỂN ĐỔI TỪ CƠ SỞ DỮ LIỆU QUAN HỆ SANG ONTOLOGY

Trong mục này, chúng tôi trình bày các luật chuyển đổi một cơ sở dữ liệu quan hệ sang các Ontology. Các không gian tên có thể được cài đặt một cách thích hợp trong các trường hợp chuyển đổi khác nhau. Trong bài báo này, chúng tôi đề xuất các không gian tên và các *IRI* như sau:

- *IRI* của tiền tố của các lớp, đối tượng, thuộc tính được chuyển đổi:

  @prefix pre: <IRI của cơ sở dữ liệu>

- *IRI* của lớp được chuyển đổi: *pre:Tên bảng dữ liệu*

  IRI của thuộc tính không phải khóa ngoại: *pre: Tên bảng + "-" + Tên thuộc tính*

- IRI của thuộc tính được chuyển đổi từ khóa ngoại:

  pre:Tên Bảng + "-" + Tên thuộc tính + "_" + Tên bảng được tham chiếu

- Mỗi *individual* được chuyển đổi từ một bộ dữ liệu sẽ có *IRI*:

  pre: Tên bảng + "ID_" + Định danh của bộ dữ liệu

Định danh của bộ dữ liệu thường là giá trị của trường khóa hay là sự kết hợp các giá trị của các thuộc tính tham gia vào khóa. Để biểu diễn các luật chuyển đổi một cách ngắn gọn, chúng tôi sử dụng cú pháp tựa như cú pháp hàm trong OWL để mô tả các luật

chuyển đổi. Để cho đơn giản, trong các luật sau đây chúng tôi không thêm *prefix pre* phía trước tên lớp hay tên thuộc tính *OWL*. Ví dụ: *Declaration (DataProperty (proName, C, xsd: string))* là một khai báo thuộc tính *proName* có domain là lớp *C* và range là *xsd:string*.

Các luật: 1, 3, 5, 11, 12, 13 được kế thừa từ kết quả của Zhou và ctg. (2010), luật 2, 4, 9 được kế thừa từ Huỳnh (2015). Luật: 6, 7, 8, 10, 14 là các luật được đề xuất trong bài báo này.

- *Luật 1.* Chuyển đổi bảng dữ liệu.

$$\forall T \in \Omega \wedge (T \notin \Omega_B) \Rightarrow Declaration(Class(T))$$

- *Luật 2.* Chuyển đổi bảng chuyên biệt hóa.

$$\forall T \in \Omega \wedge (T\ isa\ T_P) \Rightarrow SubClassOf\ (T, T_P)$$

- *Luật 3.* Chuyển đổi các thuộc tính không phải khóa chính hoặc khóa ngoại.

$$\forall A \in A(T) \Rightarrow Declaration(\text{DataProperty}(T\text{-}A,\ T,\ xsdIRI(A)))$$

- *Luật 4.* Chuyển đổi các khóa chính chỉ gồm một thuộc tính.

$$\forall A \in \text{PK(T)} \wedge (\text{Card(PK(T))} = 1) \wedge (\text{T} \notin \Omega_S)$$
$$\Rightarrow \text{Declaration(DataProperty(T} - \text{A, T, xsdIRI(A)))}$$

Tùy chọn: *[HasKey(T, T-A)]*

Việc sử dụng tùy chọn *HasKey* tùy thuộc vào mục đích ứng dụng và ý nghĩa của trường khóa *PK*. Nếu *PK* là một khóa tự nhiên như số chứng minh nhân dân, bằng lái xe..., ta nên sử dụng tùy chọn *HasKey* cho mục đích suy luận trên các đồ thị RDF hoặc hỗ trợ cho việc suy luận *SameAs* trên dữ liệu RDF tích hợp từ nhiều nguồn sau này. Nếu *PK* là khóa giả (*artificial key*), tức là khóa chỉ phục vụ mục đích phân biệt các bộ dữ liệu trong bảng, tùy chọn *HasKey* có thể bỏ qua. Chú ý rằng khóa chính của bảng chuyên biệt hóa được bỏ qua và không cần thiết phải chuyển đổi khóa chính này.

- *Luật 5.* Chuyển đổi các thuộc tính khóa ngoại không phải là khóa chính.

$$\forall FK(T_i, T_j) \notin PK(T_i) \Rightarrow$$

$$\begin{cases} Declaration(ObjectProperty(T_i - FK\_T_j, T_i, T_j); \\ Declaration(ObjectProperty(T_j - has\_T_i, T_j, T_i); \end{cases}$$

$$InverseObjectProperties(T_i - FK\_T_j, T_j - has\_T_j)$$

- *Luật 6.* Luật suy luận bắc cầu.

$$\forall FK(T, T) \Rightarrow TransitiveObjectProperty(T - FK\_T)$$

$$TransitiveObjectProperty(T - has\_T)$$

- *Luật 7.* Chuyển đổi hai thuộc tính khóa ngoại trong bảng quan hệ nhị phân.

$$\forall FK_1(T, T_1), FK_2(T, T_2) \wedge (T \in \Omega_B) \Rightarrow OP1, OP2$$

*InverseObjectProperties(OP*1, *OP*2*)*

Ta cần phải khai báo *OP1* và *OP2* là hai *Object Property*

OP1: *Declaration(ObjectProperty(*T_1*–FK*2*_*T_2*,* T_1*,* T_2*))*

OP2: *Declaration(ObjectProperty(*T_2*–FK*1*_*T_1*,* T_2*,* T_1*))*

- *Luật 8.* Luật hỗ trợ suy luận bắc cầu.

Nếu có một chuỗi các bảng T_n, T_{n-1}...T_1 có mối quan hệ khóa ngoại, $FK_1(T_1, T_2)$, $FK_2(T_2, T_3)$... $FK_{n-1}(T_{n-1}, T_n)$ với điều kiện $FK_i \notin PK(T_i)$. Ta khai báo một lớp hình thức *T_Transitive*. Các lớp OWL được chuyển đổi từ các bảng T_i là lớp con của lớp *T_Transitive.*

Declaration(Class(T_Transitive))

SubClassOf(T_i*, T_Transitive)*

và hai *TransitiveObjectProperty* tương hỗ:

Declaration(ObjectProperty(belongTo–T_Transitive,T_Transitive, _Trasitive));

TransitiveObjectProperty(belongTo-T_Transitive);

Declaration(ObjectPropery(has-T_Transitive, T_Transitive, T_Transitive));

TransitiveObjectProperty(has-T_Transitive);

InverseObjectProperties(has-T_Transitive, belongTo-T_Transitive)

Hai *Object Property* được định nghĩa ở Luật 5 lần lượt là *SubObjectProperty* của hai *Object Property belongTo-T_Transitive* và *has-T_Transitive*.

SubObjectPropertyOf(T_i–FK_i_T_{i+1}, belongTo–T_Transitive)

SubObjectPropertyOf(T_{i+1}–has_T_i, has–T_Transitive);

Tùy thuộc vào cơ sở dữ liệu, ta có thể sử dụng tên thay thế cho lớp *T_Transitive* và tên các thuộc tính *belongTo_T_Transitive, has_T_Transitive* một cách phù hợp. Trong trường hợp chuỗi các bảng có mối quan hệ khóa ngoại có độ dài bằng 3, ta nên sử dụng Luật 9.

- *Luật 9.* Luật kết nối chuỗi các thuộc tính.

$\forall FK1(T_1, T_2), FK2(T_2, T_3) \wedge (FK1 \notin PK(T_1)) \wedge (FK2 \notin PK(T_2))$

=> *Declaration(ObjectProperty(T_1–belongTo_T_3, T_1, T_3));*

SubObjectPropertyOf(ObjectPropertyChain(T_1–FK1_T_2,T_2–FK2_T_3),T_1–belongTo_T_3);

Declaration(ObjectProperty(T_3–has_T_1, T_3, T_1))

InverseObjectProperties(T_1–belongTo_T_3, T_3–has_T_1)

Tương tự như Luật 8, tên các thuộc tính *owl:propertyChainAxiom* có thể được thay thế bằng các tên thích hợp tùy vào các cơ sở dữ liệu khác nhau.

- *Luật 10.* Chuyển đổi các thuộc tính khóa chính của bảng kết hợp.

Chuyển các thuộc tính FK thành tập các *ObjectProperty:*

$$\forall FK(T_i, T_j) \wedge (T_i \in \Omega_R) \Rightarrow Declaration(ObjectProperty(T_i - FK\_T_j, T_i, T_j))$$

Chuyển đổi thành phần của khóa chính nhưng không phải là khóa ngoại thành một *DataType Property:*

$$\forall A \in PK(T) \wedge T \in \Omega_R \wedge A \notin FK(T) \Rightarrow$$
$$Declaration(\text{DataProperty}(T - A, T, xsdIRI(A)))$$

- *Luật 11.* Ràng buộc *NOT NULL* cho các thuộc tính không phải khóa ngoại.

Các thuộc tính, *A*, không phải khóa ngoại của bảng *T* có ràng buộc *NOT NULL* khi chuyển đổi sẽ được khai báo thêm ràng buộc *ExactCardinality* = 1**.**

*DataExactCardinality(*1, *T-A)*

- *Luật 12.* Ràng buộc *NOT NULL* cho các thuộc tính khóa ngoại.

Thuộc tính khóa ngoại $FK(T_i,T_j)$ có ràng buộc *NOT NULL* khi chuyển đổi sẽ được khai báo thêm ràng buộc *ObjectExactCardinality* = 1 cho một *ObjectProperty*. Không khai báo ràng buộc này cho thuộc tính *ObjectProperty* tương hỗ còn lại.

*ObjectExactCardinality(*1, T_i–*FK_*T_j*)*

Không khai báo ràng buộc này cho *ObjectProperty* T_j – *has_*T_i .

- *Luật 13.* Ràng buộc Unique.

Thuộc tính, *A* của bảng *T*, không phải khóa chính hoặc khóa ngoại có ràng buộc unique khi chuyển đổi sẽ được khai báo thêm ràng buộc *DataMaxCardinality* = 1.

*DataMaxCardinality(*1, *T-A)*

Thuộc tính khóa ngoại *FK(T_i, T_j)* có ràng buộc Unique khi được chuyển đổi sẽ được khai báo thêm ràng buộc *ObjectMaxCardinality* = 1 cho một Object Property. Không khai báo ràng buộc này cho thuộc tính Object Property tương hỗ còn lại.

*ObjectMaxCardinality(*1, *T_i–FK_T_j)*

- *Luật 14.* Chuyển đổi các bản ghi.

Mỗi bản ghi không thuộc bảng quan hệ nhị phân hoặc bảng chuyên biệt hóa được chuyển thành một đối tượng.

$$\forall t \in t(T) \wedge (T \notin \Omega_B) \wedge (T \notin \Omega_S) \Rightarrow ClassAsser\,tion(T, getIRI(t))$$

Với *getIRI(t)* là hàm tính giá trị định danh của bộ dữ liệu *t* dựa vào giá trị khóa chính của *T*. Trong trường hợp *t* là bộ dữ liệu thuộc bảng kết hợp, định danh của bộ dữ liệu *t* là một sự kết hợp các giá trị trên các thuộc tính khóa của bảng *T*. Ví dụ, chúng ta có thể sử dụng cách đơn giản nhất là nối các giá trị của bộ dữ liệu *t* trên các thuộc tính khóa để làm định danh của bộ dữ liệu *t*.

Chuyển đổi các giá trị *t.A* của thuộc tính *A* không phải giá trị khóa ngoại (Chú ý rằng khóa chính của một bảng chuyên biệt hóa cũng chính là khóa ngoại):

$$\forall t.A \mid t \in t(T) \wedge (A \notin FK(T)) \Rightarrow DataPropertyAssertion(T - A, getIRI(t), getStringValue(t.A)\,\hat{}\hat{}\,xsdIRI(A))$$

Với *getStringValue(t.A)* là hàm trả về giá trị chuỗi của thuộc tính *A* trong bộ dữ liệu *t*. Trong trường hợp *A* là một thuộc tính thuộc bảng chuyên biệt hóa, định danh của bộ dữ liệu *t* chính là định danh của bộ dữ liệu mà giá trị khóa chính của *t* tham chiếu đến.

Chuyển đổi các giá trị khóa ngoại không thuộc bảng quan hệ nhị phân và không phải là khóa chính của bảng chuyên biệt hóa:

$$\forall t.FK \mid t \in t(T) \wedge (FK = FK(T, T_i)) \wedge (T \notin \Omega_B) \wedge (FK \notin PK(T) \mid T \in \Omega_S) \Rightarrow$$
$$ObjectPropertyAssertion(T\text{-}FK\_T_i, getIRI(t), getIRI(t.FK))$$

Trong đó *getIRI(t.FK)* là hàm xác định IRI của bộ dữ liệu được tham chiếu bởi bộ dữ liệu *t* thông qua giá trị của khóa ngoại *FK*.

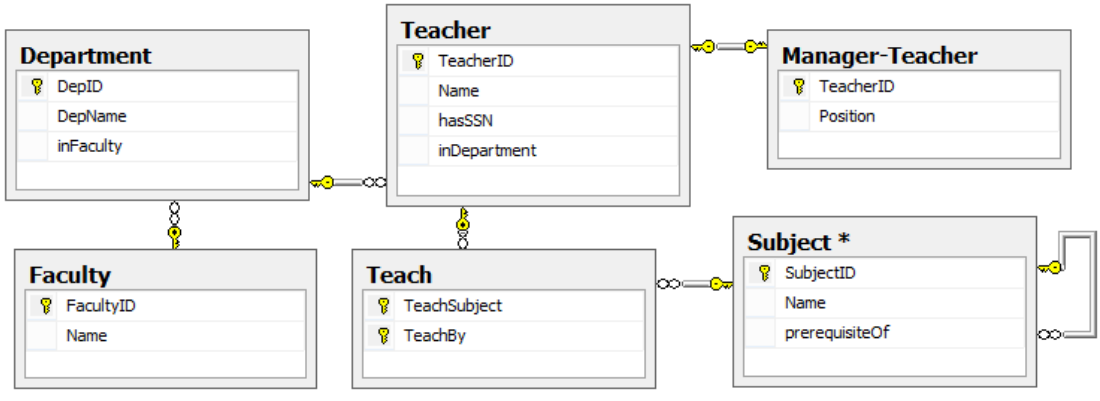
Chuyển đổi các giá trị khóa ngoại thuộc bảng quan hệ nhị phân:

$$\forall t.FK1, t.FK2 \mid t \in t(T) \wedge (T \in \Omega_B) \Rightarrow$$
$$ObjectPropertyAssertion(T_1 - FK2\_T_2, getIRI(t.FK1), getIRI(t.FK2))$$

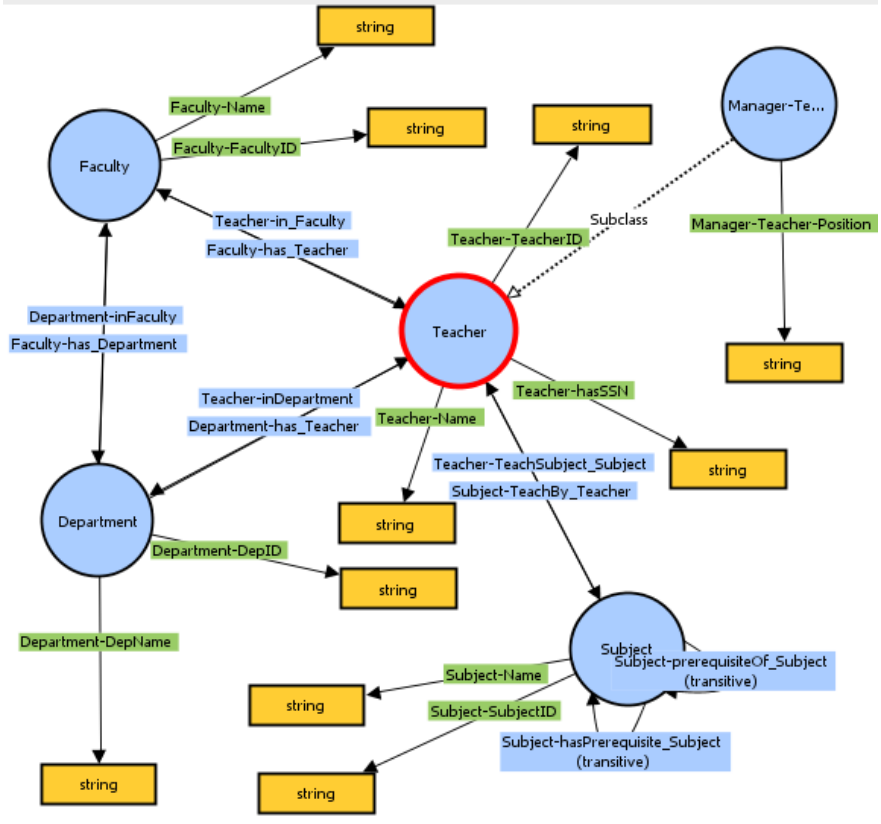
Chú ý rằng theo Luật 14 thì một bản ghi thuộc bảng chuyên biệt hóa hoặc bảng quan hệ nhị phân sẽ không được chuyển đổi thành một đối tượng OWL. Khóa chính của bảng chuyên biệt hóa không xét đến khi chuyển đổi. Các giá trị của các thành phần không phải khóa chính của một bản ghi, *t*, thuộc một bảng chuyên biệt hóa được chuyển đổi thành các giá trị của các thuộc tính OWL tương ứng có domain là một đối tượng đã được chuyển đổi từ một bản ghi có giá trị khóa chính bằng giá trị khóa chính của *t*.

5. MINH HỌA CÁC LUẬT CHUYỂN ĐỔI

Để minh họa các luật đã đề xuất chúng tôi sử dụng một cơ sở dữ liệu minh họa, TeacherDB, lưu trữ thông tin các giảng viên của một trường đại theo lược đồ trong Hình 1 (được vẽ trong MS SQL Server). Trong thực tế, một môn học (*subject*) có thể là điều kiện tiên quyết cho nhiều môn học khác. Tuy nhiên với mục đích minh họa, chúng tôi giả sử rằng một môn học chỉ có thể là điều kiện tiên quyết cho nhiều nhất một môn học khác.



Hình 1. Cơ sở dữ liệu minh họa TeacherDB



Hình 2. Sơ đồ các lớp và thuộc tính OWL đã được chuyển đổi của CSDL TecherDB

Để chuyển đổi cơ sở dữ liệu TeacherDB sang Ontology, chúng tôi sử dụng công cụ Protégé 5.0 của Đại học Stanford để khai báo các lớp, thuộc tính. Hình 2 là sơ đồ lớp và các thuộc tính OWL được chuyển đổi từ cơ sở dữ liệu TeacherDB. Chú ý rằng *Teacher-in_Faculty* là một *owl:propertyChainAxiom.*

Giả sử *pre: IT* là IRI của Khoa Công nghệ Thông tin. Đối với mô hình chuyển đổi trong Hình 2, câu truy vấn liệt kê danh sách giáo viên Khoa Công nghệ Thông tin là:

SELECT ?x WHERE { pre:IT pre:Faculty-has_Teacher ?x .}

Hoặc: *SELECT ?x WHERE {?x pre:Teacher-in_Faculty pre:IT .}*

Câu truy vấn SPARQL liệt kê danh sách tất cả các môn học điều kiện tiên quyết của môn hệ điều hành, pre:OS, là:

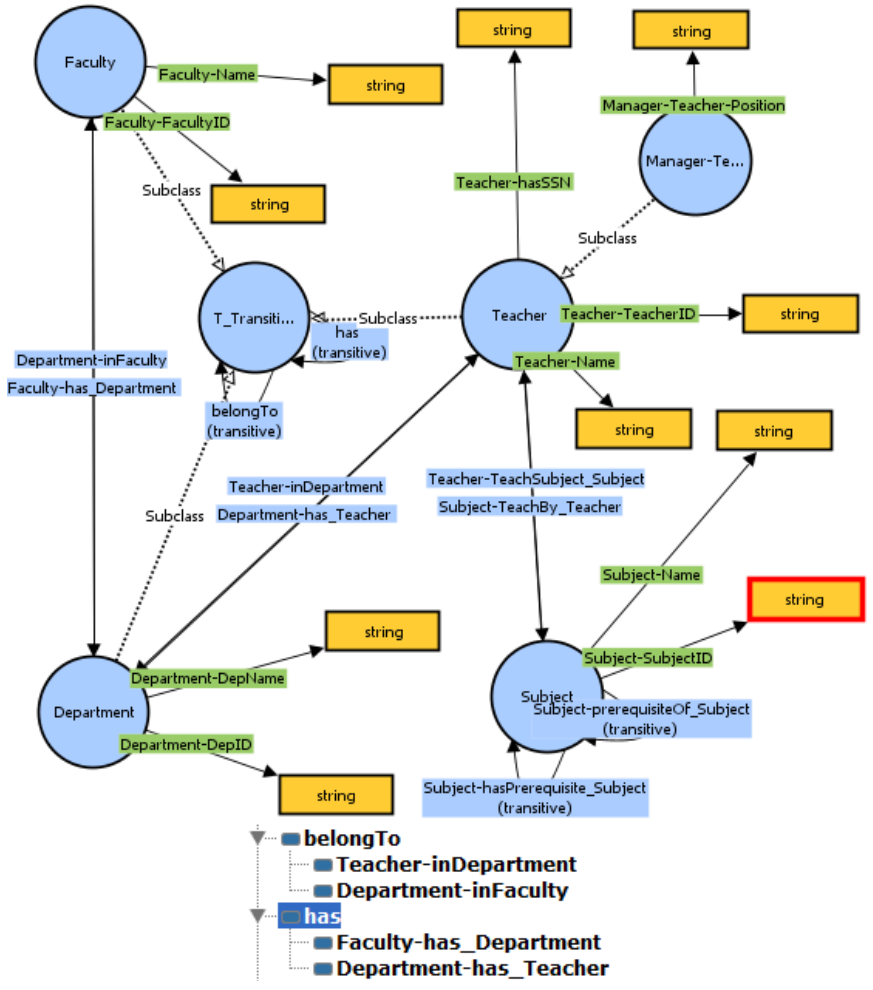
SELECT ?x WHERE {pre:OS pre:Subject-hasPrerequisite_Subject ?x.}

Hình 3 cũng là một sơ đồ lớp và các thuộc tính OWL như Hình 2 nhưng các *owl:propertyChainAxiom* đã được thay thế bằng các *owl:TransitiveProperty* thông qua một lớp *T_Transitive* và hai thuộc tính *TransitiveProperty* là *has* và *belongTo*. Các thuộc tính: *Teacher-in_Department* và *Department-in_Faculty* là *owl:subPropertyOf* của thuộc tính belongTo. Hai thuộc tính *Faculty_has_Department* và *Department-has_Teacher* là *owl:subPropertyOf* của thuộc tính *has*.

Câu truy vấn SPARQL sau sẽ liệt kê tất các *IRI* của giáo viên thuộc Khoa Công nghệ Thông tin ứng với mô hình chuyển đổi trong Hình 3.

SELECT ?x WHERE { pre:IT pre:has ?x . ?x rdf:type pre:Teacher .}

Hoặc: *SELECT ?x WHERE {?x pre:belongTo pre:IT.?x rdf:type pre:Teacher .}*



Hình 3. Thay thế các thuộc tính *owl:PropertyChainAxiom* bằng các *owl:TransitiveProperty*

Chú ý rằng để các câu truy vấn trên đây đưa ra kết quả đúng, chúng ta phải sử dụng các mô đun suy luận khi thực hiện câu truy vấn. Trong thực nghiệm, chúng tôi sử dụng mô đun suy luận HermiT 1.3.8.3 được tích hợp sẵn trong phần mềm Protégé.

6. ĐÁNH GIÁ CÁC LUẬT CHUYỂN ĐỔI

Trong bài báo này chúng tôi đã trình bày các luật chuyển đổi từ một cơ sở dữ liệu sang các Ontology. Vì mô hình dữ liệu RDF chú trọng vào việc biểu diễn các tri thức nên khi chuyển đổi chúng tôi đã bổ sung thêm một số thuộc tính OWL để hỗ trợ suy luận trên các Ontology. So với các nghiên cứu trước đây, chúng tôi đã bổ sung các thuộc tính: *owl:TransitiveProperty*, *owl:propertyChainAxiom* để hỗ trợ các suy luận bắc cầu. Thuộc tính *owl:TransitiveProperty* được chúng tôi ánh xạ một cách tổng quát cho một dãy các bảng có mối quan hệ khóa ngoại chứ không chỉ dừng ở mức 3 bảng. Thuộc tính *owl:propertyChainAxiom* được xem là một trường hợp đặc biệt của *owl:Transitive Property* khi dãy các bảng có mối quan hệ khóa ngoại chỉ gồm 3 bảng. Bên cạnh đó chúng tôi cũng chỉ ra rằng, khóa ngoại của một bảng tham chiếu đến khóa chính của bảng đó cũng có thể được chuyển đổi thành một thuộc tính *owl:TransitiveProperty*.

Việc chuyển đổi các bản ghi cũng được chúng tôi đề cập rất chi tiết. Để việc chuyển đổi từ một cơ sở dữ liệu sang một một mô hình tri thức gần gũi với thực tế, chúng tôi không chuyển đổi các khóa chính của bảng chuyên biệt hóa, các bản ghi của các bảng quan hệ nhị phân hay của bảng chuyên biệt hóa cũng không được chuyển đổi thành một đối tượng. Tuy nhiên giá trị các thuộc tính không phải khóa chính của bảng chuyên biệt hóa vẫn được chuyển đổi thành giá trị của các thuộc tính OWL. Ví dụ, lớp *Manager-Teacher* là một lớp con của lớp *Teacher*, hiển nhiên một đối tượng của lớp *Manager-Teacher* cũng là một đối tượng của lớp *Teacher* và hai đối tượng này phải có chung *IRI*. Vì vậy việc tạo thêm một đối tượng mới cho một bản ghi của bảng *Teacher-Manager* là không cần thiết bởi vì đối tượng này đã tồn tại trong tập các đối tượng của lớp *Teacher*. Khi chuyển đổi, thuộc tính *Manager-Teacher_Position* có domain là lớp *Manager-Teacher*. Dựa vào domain của các thuộc tính, bộ suy luận trên dữ liệu Ontology sẽ chỉ ra lớp cụ thể của một đối tượng. Một đóng góp nữa của bài báo này là việc chuyển đổi các

bảng kết hợp thành các Ontology. Chúng tôi cho rằng đây là những điểm khác biệt so với các nghiên cứu trước đây.

7. KẾT LUẬN

Trong bài báo này, chúng tôi đã trình bày các luật chuyển đổi từ cơ sở dữ liệu quan hệ sang các Ontology. Dựa trên các mối kết hợp giữa các bảng trong cơ sở dữ liệu, chúng tôi cũng đưa ra các Axiom cho các Ontology nhằm hỗ trợ các mô đun suy luận của ứng dụng Sematic web. So với các nghiên cứu đã có, việc chuyển đổi dữ liệu của cơ sở dữ liệu quan hệ sang các Ontology của chúng tôi có chú trọng đến ngữ nghĩa của mô hình tri thức sau chuyển đổi. Mặt khác, các luật chuyển đổi đưa ra rất chi tiết và bao phủ hầu hết các trường hợp của một cơ sở dữ liệu ở chuẩn 3. Dựa vào các luật đã đề xuất, các mô đun có thể được xây dựng để tự động chuyển đổi một cơ sở dữ liệu quan hệ thành các Ontology. Tuy nhiên, để đảm bảo mô hình tri thức cho các Ontology được thể hiện một cách có ý nghĩa, tên của các thuộc tính OWL có thể được sửa đổi một cách thích hợp trước khi chuyển đổi các bản ghi trong cơ sở dữ liệu quan hệ thành các Ontology.

Để các Ontology có thể được chia sẻ và sử dụng một cách có hiệu quả, các Ontology được chuyển đổi có thể được tích hợp với các Ontology khác. Vì vậy, bên cạnh việc đề xuất các luật để hỗ trợ việc suy luận trên các Ontology được chuyển đổi, việc tích hợp các Ontology từ nhiều nguồn khác nhau cũng phải cần được chú trọng nghiên cứu.

TÀI LIỆU THAM KHẢO

Arenas, M. (2012). *A direct mapping of relational data to RDF*. Retrieved from http://www.w3.org/TR/rdb-direct-mapping/

Ayoub, O., Mohamed, B., & Ilias, C. (2015). Creating an RDF graph from a relational database using SPARQL. *Journal of Software*, *10*(4), 384-391.

Das, S. (2012). *R2RML: RDB to RDF mapping language*. Retrieved from http://www.w3.org/TR/r2rml/

Dean, A., & Jim, H. (2011). *Semantic web for the working Ontologist* (2nd ed.). Massachusetts, USA: Morgan Kaufmann.

Huỳnh. T. A. (2015). *Các luật chuyển đổi từ CSDL quan hệ sang Ontology*. Bài báo được trình bày tại Hội thảo khoa học CNTT và Truyền thông ICT 2015, Việt Nam.

Lei, Z., & Jing, L. (2011). Automatic generation of Ontology based on database. *Journal of Computational Information Systems, 7*(4), 1148-1154.

Mallede, W. Y., Marir, F., & Vassilev, V. T. (2013). *Algorithms for mapping RDB schema to RDF for facilitating access to deep web*. Paper presented at The First International Conference on Building and Exploring Web Based Environments, Spain.

Pascal, H., Markus, K., & Bijan, P. (2012). *OWL 2 web Ontology language primer* (2nd ed.). Retrieved from: http://www.w3.org/TR/ 2012/REC-owl2-primer-20121211/

Raji, G., & Nadine, C. (2007). *Database to Ontology mapping generation for semantic interoperability*. Paper presented at the Very Large Database Endowment '07, Austria.

Shufeng, Z., Haiyun, L., Mei, H., & Huaiwei, Z. (2010). Ontology generator from relational database based on Jena. *Computer and Information Science, 3*(2), 21-40.

KNOWLEDGE MODELING FOR A RELATIONAL DATABASE IN ONTOLOGY WEB LANGUAGE (OWL)

Huynh Tuan Anh[a*]

[a]The Faculty of Information Technology, Nhatrang University, Khanhhoa, Vietnam
[*]Corresponding author: Email: anhht@ntu.edu.vn

Article history
Received: January 10th, 2017 | Received in revised form: April 10th, 2017
Accepted: May 11th, 2017

Abstract

This paper presents a knowledge modeling method by Ontology Web Language (OWL) for a relational database. The proposed method contains the rules for transforming data in a relational database into Ontology and Axioms for supplementing meaning of a relational database. Based on these rules, data in a relational database could be converted into triples Resource Description Framework (RDF)/OWL for Semantic web applications.

Keywords: Mapping; Ontology; OWL; Relational Database; RDF; RDFS; Semantic web.