Master research Internship

Bibliographic report

Gestion de points de vues sur des données RDF

Domain: Databases - Artificial Intelligence

Author:
Ludivine Duroyon

Supervisor:
François Goasdoué
Ioana Manolescu
SHAMAN (IRISA) / CEDAR
(Inria)

ENSSAT
LANNION

Abstract: Cette bibliographie est un travail préliminaire à mon stage portant sur la gestion de données du journalisme, exprimées dans le modèle RDF du W3C et enrichies avec des informations temporelles et des points de vue. Ce travail bibliographique présente donc le modèle RDF puis fait l'état des lieux des propositions de la littérature pour la représentation du temps et de points de vue sur des faits, afin d'identifier un point de départ de mes travaux de stage à venir.

Table des matières

1 Introduction

Mon stage va s'effectuer dans le contexte du domaine journalistique. Il s'agit de créer une base de données permettant de stocker de nombreuses données venant de différents supports. L'intérêt est d'avoir une vue d'ensemble des données homogène malgré l'hétérogénéité des données.

C'est pourquoi on va s'intéresser au modèle de données RDF (Resource Description Framework). C'est un modèle qui sert actuellement à décrire des données sémantiquement riches du Web et est recommandé par le W3C (*World Wide Web Consortium*).

Dans le domaine journalistique, le temps et les points de vues sont des données qui ont toutes leurs importances. Le temps permet de créer des liens chronologiques entre les données, ce qui peut intéressant pour enquêter sur un sujet. Les points de vues permettent de voir plus facilement les avis et opinions de politiques par exemple. De combiner les deux permettrait sûrement de faire des lien entre un politique qui change de discours et un évènement extérieur qui aurait pu le conduire à ce comportement.

Nous nous intéresserons donc sur les manières d'enrichir RDF avec des points de vues de multiples acteurs et aussi d'y ajouter une dimension temporelle. Par la suite, il sera question de pouvoir intérroger cette base de manière à pouvoir exploiter ces aspects des données.

Je vais donc commencer par présenter le modèle RDF. Puis nous verrons comment enrichir une base de données avec du temps. Nous commencerons par les bases de données relationnelles où les recherches sont abouties puis on traitera la partie RDF. Finalement, nous terminerons par l'ajout de points de vues, là les références sont plus nombreuses en relationnel.

2 Modèle de données RDF et langage de requête SPARQL

Cette section introduit le modèle RDF [11, 12] et son langage d'interrogation SPARQL [13, 14] sur la base de la présentation qui en est faite dans [2].

RDF. Le modèle RDF *Ressource Description Framework* est un modèle de données "graphe" permettant de décrire formellement des ressources Web par des métadonnées. Ce modèle a été développé par le W3C (*World Wide Web Consortium*).

Un graphe RDF est composé de triplets notés $s\ p\ o$. Un triplet représente un arc $s \xrightarrow{p} o$ et indique que le *sujet* s peut être (partiellement) décrit par la *propriété* p ayant la valeur ou *objet* o.

Trois ensembles servent à exprimer de tels triplets : $\mathcal{U}$ l'ensembles des URI *Universal Ressource Identifiers* qui permet d'indentifer chaque resources, $\mathcal{L}$ l'ensemble des litéraux (constantes) et $\mathcal{B}$ l'ensemble des noeuds blancs. Un noeud blanc permet de modéliser une information incomplète, il peut être vu comme une variable. Un triplet bien formé appartient à $(\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{L} \cup \mathcal{B})$.

Les faits. Un graphe RDF de triplets modélise un ensemble de faits, appelés *assertions RDF*, chacune exprimant une instance de *classe* (relation unaire) ou une instance de *propriété* (relation binaire). La syntaxe de ces assertions est donnée dans la Table 1 (en haut) ; rdf:type est la propriété fournie par le standard RDF pour exprimer l'assertion de classe.

Les contraintes ontologiques. Les graphes RDF peuvent être enrichis à l'aide de contraintes ontologiques servant à décrire le domaine d'application. Ces contraintes s'expriment grâce à *RDF Schema* (RDFS) ; elles offrent la possibilité de modéliser des relations liant sémantiquement les

Assertion RDF	Triple
Assertion de classe	$(\mathtt{s}, \text{rdf:type}, \mathtt{o})$
Assertion de propriété	$(\mathtt{s}, \mathtt{p}, \mathtt{o})$ avec $\mathtt{p} \neq \text{rdf:type}$

Assertion RDFS	Triple
Sous-classe	$(\mathtt{s}, \text{rdfs:subClassOf}, \mathtt{o})$
Sous-propriété	$(\mathtt{s}, \text{rdfs:subPropertyOf}, \mathtt{o})$
Type du domaine	$(\mathtt{s}, \text{rdfs:domain}, \mathtt{o})$
Type du range	$(\mathtt{s}, \text{rdfs:range}, \mathtt{o})$

TABLE 1 – Assertions RDF & RDFS.

Règle [12]	Règles de déduction
`rdfs2`	$(\mathtt{p}, \hookleftarrow_d, \mathtt{o}), (\mathtt{s}_1, \mathtt{p}, \mathtt{o}_1) \rightarrow (\mathtt{s}_1, \tau, \mathtt{o})$
`rdfs3`	$(\mathtt{p}, \hookrightarrow_r, \mathtt{o}), (\mathtt{s}_1, \mathtt{p}, \mathtt{o}_1) \rightarrow (\mathtt{o}_1, \tau, \mathtt{o})$
`rdfs5`	$(\mathtt{p}_1, \preceq_{\text{sp}}, \mathtt{p}_2), (\mathtt{p}_2, \preceq_{\text{sp}}, \mathtt{p}_3) \rightarrow (\mathtt{p}_1, \preceq_{\text{sp}}, \mathtt{p}_3)$
`rdfs7`	$(\mathtt{p}_1, \preceq_{\text{sp}}, \mathtt{p}_2), (\mathtt{s}, \mathtt{p}_1, \mathtt{o}) \rightarrow (\mathtt{s}, \mathtt{p}_2, \mathtt{o})$
`rdfs9`	$(\mathtt{s}, \preceq_{\text{sc}}, \mathtt{o}), (\mathtt{s}_1, \tau, \mathtt{s}) \rightarrow (\mathtt{s}_1, \tau, \mathtt{o})$
`rdfs11`	$(\mathtt{s}, \preceq_{\text{sc}}, \mathtt{o}), (\mathtt{o}, \preceq_{\text{sc}}, \mathtt{o}_1) \rightarrow (\mathtt{s}, \preceq_{\text{sc}}, \mathtt{o}_1)$
`ext1`	$(\mathtt{p}, \hookleftarrow_d, \mathtt{o}), (\mathtt{o}, \preceq_{\text{sc}}, \mathtt{o}_1) \rightarrow (\mathtt{p}, \hookleftarrow_d, \mathtt{o}_1)$
`ext2`	$(\mathtt{p}, \hookrightarrow_r, \mathtt{o}), (\mathtt{o}, \preceq_{\text{sc}}, \mathtt{o}_1) \rightarrow (\mathtt{p}, \hookrightarrow_r, \mathtt{o}_1)$
`ext3`	$(\mathtt{p}, \preceq_{\text{sp}}, \mathtt{p}_1), (\mathtt{p}_1, \hookleftarrow_d, \mathtt{o}) \rightarrow (\mathtt{p}, \hookleftarrow_d, \mathtt{o})$
`ext4`	$(\mathtt{p}, \preceq_{\text{sp}}, \mathtt{p}_1), (\mathtt{p}_1, \hookrightarrow_r, \mathtt{o}) \rightarrow (\mathtt{p}, \hookrightarrow_r, \mathtt{o})$

TABLE 2 – Exemple de règles de déduction RDF.

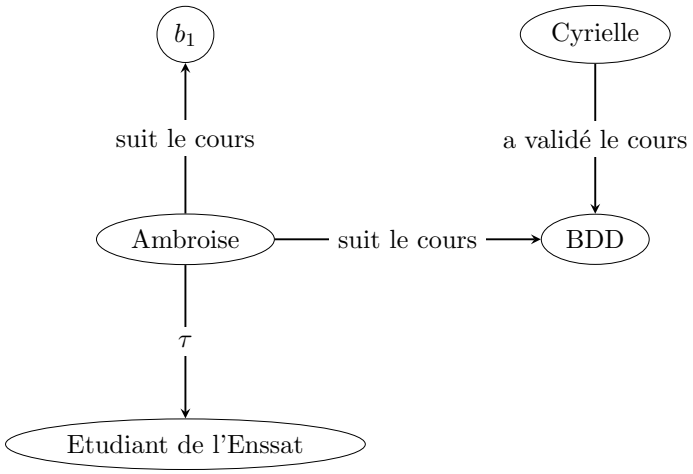

FIGURE 1 – Exemple de graphe RDF $\mathcal{G}$.

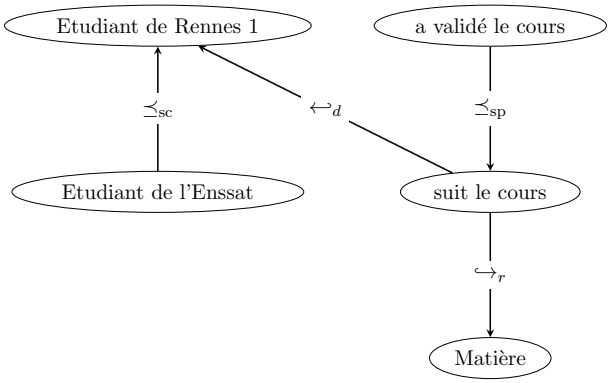

FIGURE 2 – Exemple de graphe RDF $\mathcal{G}'$.

classes et les propriétés utilisées dans les assertions RDF. Ces contraintes, appelées *assertions RDFS*, sont de type : sous-classe, sous-propriété, typage du premier attribut ou *domaine* d'une propriété, et enfin de typage du second attribut ou *range* d'une propriété. La syntaxe de ces assertions est donnée dans la Table 1 (en bas) ; rdfs:subClassOf, rdfs:subPropertyOf, rdfs:domain et rdfs:range sont les propriétés fournies par le standard RDFS pour exprimer les contraintes ontologiques.

Notation. Dans la suite, nous utiliserons les notations suivantes éventuellements avec des indices : s pour un sujet de triplet, p pour une propriété de triplet, o pour un objet de triplet, et b pour un noeud blanc.

Les raccourcis suivants de notation de propriétés issues du standard RDF seront aussi utilisés : τ pour rdf:type, $\preceq_{\text{sc}}$ pour rdfs:subClassOf, $\preceq_{\text{sp}}$ pour rdfs:subPropertyOf, $\hookleftarrow_d$ pour rdfs:domain et $\hookrightarrow_r$ pour rdfs:range.

Exemple. Le graphe RDF $\mathcal{G}$ de la Figure 1 exprime qu'Ambroise suit le cours de BDD, ainsi qu'un autre cours dont la valeur n'est pas disponible, grâce aux triplets $(Ambroise, suit\ le\ cours\ de, BDD)$ et $(Ambroise, suit\ le\ cours\ de, b_1)$. Ce graphe indique également qu'Ambroise est étudiant à l'ENSSAT grâce au triplet $(Ambroise, \tau, Etudiant\ de\ l'ENSSAT)$. Enfin, il exprime aussi que Cyrielle

a validé le cours de BDD suivi par Ambroise grâce au triplet : $(Cyrielle, a\ valide\ le\ cours, BDD)$.

Le graphe RDF $\mathcal{G}'$ de la Figure 2 exprime un ensemble de contraintes ontologiques sur des classes et propriétés, dont certaines sont utilisées dans le graphe RDF $\mathcal{G}$. Il modélise que les étudiants de l'ENSSAT sont des étudiants de Rennes 1 ($Etudiant\ de\ l'ENSSAT, \preceq_{\text{sc}}, Etudiant\ de\ Rennes\ 1$), que seuls des étudiants de Rennes 1 peuvent suivre des cours ($suit\ le\ cours, \hookleftarrow_d, Etudiant\ de\ Rennes\ 1$), que seules des matières peuvent être des cours suivis ($suit\ le\ cours, \hookrightarrow_r, Matiere$) et qu'avoir validé un cours c'est avoir suivi ce cours ($a\ valide\ le\ cours, \preceq_{\text{sp}}, suit\ le\ cours$).

Interprétation des contraintes ontologiques. En gestion de données, il existe deux façons d'interpréter des contraintes [1] : soit sous l'hypothèse du monde fermé ou *closed-world assumption (CWA)*, soit sous l'hypothèse du monde ouvert ou *open-world assumption (OWA)*.

Sous CWA, qui est par exemple l'hypothèse retenue pour les bases de données relationnelles, seul ce qui est explicitement stocké est vrai (tout le reste est faux). Ainsi, sous cette hypothèse, à partir des exemples ci-dessus, le graphe RDF $\mathcal{G}'' = \mathcal{G} \cup \mathcal{G}'$ est inconsistant car, par exemple, Cyrielle a validé le cours de BDD, ce qui implique qu'elle l'ait suivi du fait des contraintes ontologiques (F), hors puisque ce fait n'est pas stocké dans $\mathcal{G}''$, il est faux ($\neg F$).

Sous OWA, qui est l'hypothèse retenue pour les graphes RDF, un graphe RDF ne stocke qu'une partie des triplets vrais, d'autres triplets non stockés pouvant également être vrais, notamment ceux dérivables des contraintes ontologiques. Ainsi, sous cette hypothèse, le graphe $\mathcal{G}''$ ci-dessus est consistant et permet, grâce aux contraintes ontologiques, que puisque Cyrielle a validé le cours de BDD, c'est qu'implicitement elle l'a suivi.

Sémantique d'un graphe RDF. La sémantique d'un graphe RDF, du fait de l'hypothèse OWA retenue par le W3C, correspond à sa *saturation ou clôture*, c'est-à-dire à ce graphe augmenté de tous ses triplets implicites, dérivables à partir de ce graphe et de règles de déduction RDF. La Table 2 illustre un sous-ensemble de règles courramment utilisées du standard RDF ; elles correspondent à la dérivation de nouveaux faits et contraintes ontologiques, à partir de faits ou contraintes ontologiques.

Dans le modèle de données RDF, la saturation $\mathcal{G}^\infty$ d'un graphe RDF est notée $\mathcal{G}^\infty$. Notamment, quel que soit l'ensemble utilisé de règles de déduction RDF, $\mathcal{G}^\infty$ est finie et unique (modulo renommage de ses noeuds blancs jouant le rôle de variables).

La Figure 3 montre les assertions RDF explicites du graphe $\mathcal{G}''$, sous forme d'arcs pleins, ainsi que ses assertions RDF implicites pouvant être dérivées par les règles de déduction de la Table 2, sous forme d'arcs en pointillés.

Les requêtes. Le standard d'interrogation pour les graphes RDF est le langage de requête SPARQL du W3C [13, 14].

Au coeur de ce langage se trouve le fragment de SPARQL permettant d'exprimer des requêtes de sélection-projection-jointure appelées les requêtes *Basic Graph Pattern* (BGP). Un BGP est un ensemble de triplets, dont les sujets, propriétés et objets peuvent être des variables d'un ensemble $\mathcal{V}$ disjoint de $\mathcal{U}$, $\mathcal{L}$ et $\mathcal{B}$. Les BGP généralisent les graphes RDF en comprenant des triplets de $(\mathcal{U} \cup \mathcal{B} \cup \mathcal{V}) \times (\mathcal{U} \cup \mathcal{V}) \times (\mathcal{U} \cup \mathcal{L} \cup \mathcal{B} \cup \mathcal{V})$.

Une requête BGP est notée $q(\bar{x}) \leftarrow t_1, \ldots, t_\alpha$ où $\{t_1, \ldots, t_\alpha\}$ est un BGP et $\bar{x}$ est un sous-ensemble des variables du BGP appelée variables réponses.

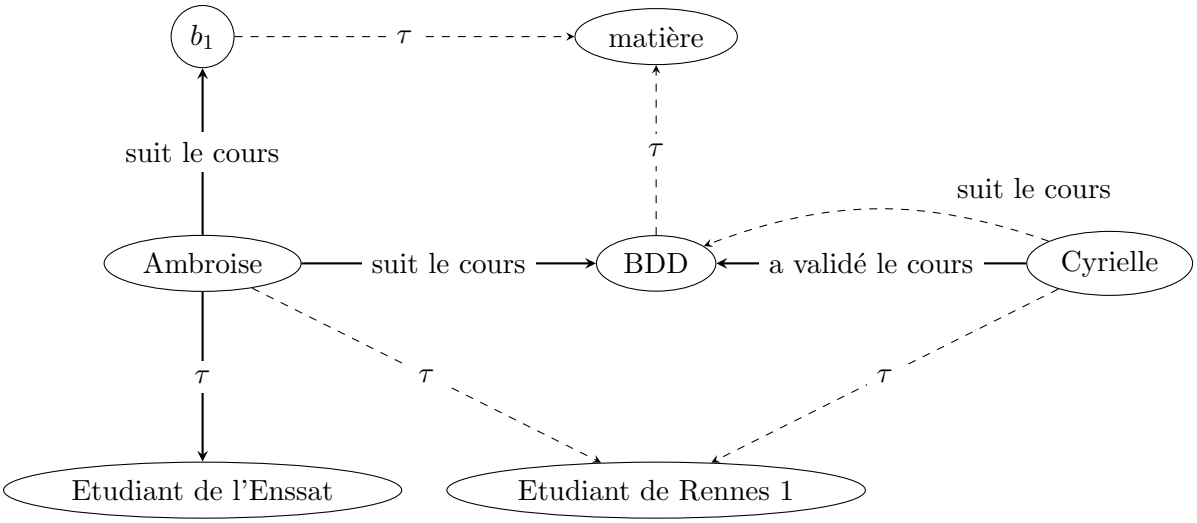

FIGURE 3 – Assertions RDF explicites et implicites de la saturation de $\mathcal{G} \cup \mathcal{G}'$ (cf. Figures 1 et 2).

L'*évaluation* q sur un graphe RDF $\mathcal{G}$, notée $q(\mathcal{G})$, est définie à l'aide d'homomorphismes de graphes par :

$$q(\mathcal{G}) = \{[\bar{x}]_\phi \mid [\{t_1, \ldots, t_\alpha\}]_\phi \subseteq \mathcal{G}\}$$

où ϕ est une assignation des variables et noeuds blancs de q à des valeurs de $\mathcal{G}$ (URI, littéraux et noeuds blancs), $[\{t_1, \ldots, t_\alpha\}]_\phi$ le graphe RDF obtenu sous l'assignation ϕ et $[\bar{x}]_\phi$ le tuple de valeurs de $\mathcal{G}$ sous l'assignation ϕ.

Les *réponses* à q sur $\mathcal{G}$ sont obtenues en évaluant q sur $\mathcal{G}^\infty$, c'est-à-dire $q(\mathcal{G}^\infty)$, afin de prendre en compte la sémantique de ce graphe.

Exemple. Soit la requête suivante : $q(x_1, x_2) \leftarrow (x_1, \textit{suit le cours}, x_2)$.

L'évaluation de cette requête sur le graphe $\mathcal{G}''$ ci-dessus est

$$\{(Ambroise, BDD), (Ambroise, b_1)\}$$

car seuls les triplets explicites de la Figure 3 sont considérés.

La réponse à cette requête sur le même graphe est

$$\{(Ambroise, BDD), (Ambroise, b_1), (Cyrielle, BDD)\}$$

car les triplets explicites et implicites de la Figure 3 sont considérés.

3 Le temps

Dans le contexte journalistique mais aussi dans bien d'autres, nous avons besoin d'avoir des repères temporels. C'est pour cela que nous allons nous intéresser à la façon de stocker des variations de temps dans les bases de données. Dans la Section 3.1, nous montrons comment le temps est représenté dans les bases de données temporelles. Dans la Section 3.2, nous présentons les contreparties de ces modèles spécifiques au modèle RDF.

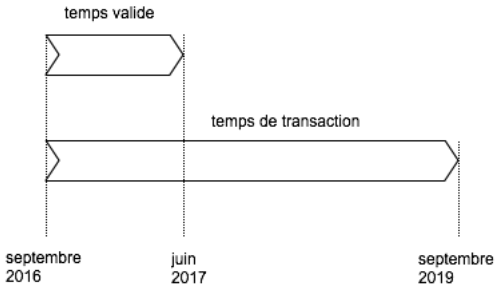

FIGURE 4 – temps de validité et temps de transaction.

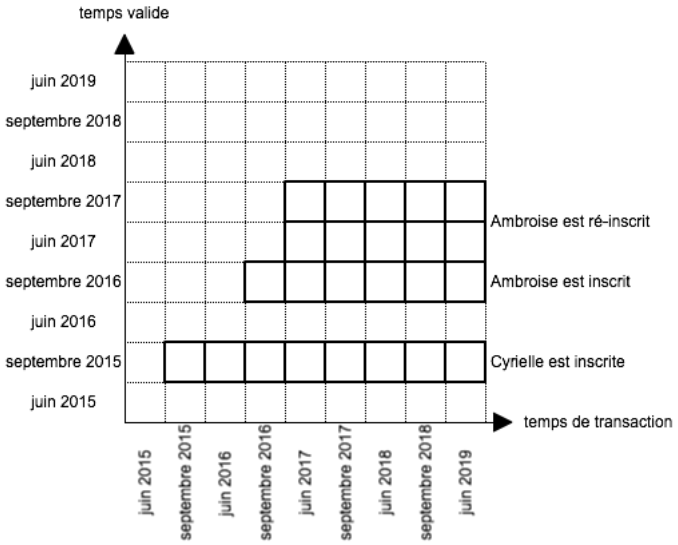

FIGURE 5 – Illustration du modèle.

Prénom	Cours suivi	T_d	T_f	V_d	V_f
Cyrielle	BDD	sept 15	UC	sept 15	juin 16
Ambroise	BDD	sept 16	juin 17	sept 16	juin 17
Ambroise	BDD	juin 17	UC	sept 16	juin 18

TABLE 3 – Table temporelle INSCRIPTION.

3.1 Le temps dans les bases de données relationnelles

Les premiers travaux auxquels on s'intéresse sont ceux dont le but a été de représenter de façon complète le temps dans les bases de données relationnelles. En effet, les travaux dans le domaine relationnel sont bien avancés. Pour cette partie je vais principalement m'inspirer de l'article « *Temporal Data Management* » de Christian Jensen et Richard Snodgrass [7], très reconnus pour leurs travaux dans ce domaine.

Aspects temporels. Les deux concepts fondamentaux à la base des représentations utilisées sont le temps de validité et le temps de transaction.

Le *temps de validité* est lié aux faits représentés dans la base de données ; c'est le moment où un fait est vrai.

Le *temps de transaction* représente le moment où l'information est stockée dans la base. La durée commence donc au moment où on l'ajoute dans la base et se termine quand on le supprime.

Par exemple, prenons le fait que Cyrielle suit le cours de BDD pendant l'année scolaire 2015/2016. Elle s'est inscrite en septembre de 2015 à ce cours et on a enregistré l'information dans notre base de données à cette date là. De plus, on sait que les cours s'arrêtent en juin 2016 et que les données sont conservées trois ans.

Le temps de validité pour cette information est donc de septembre 2015 à juin 2016 et le temps de transaction est de septembre 2015 à septembre 2019, comme montré dans la figure 4.

Représentation du temps. On peut représenter le temps de différentes façons. On peut le discrétiser et le représenter par des points temporels, par exemple, "midi heure de Paris, le 22 janvier 2017". Ou alors on peut choisir d'utiliser des intervalles de temps, par exemple "de midi à quatorze heures, heure de Paris, le 22 janvier 2017". On remarque que dès que l'on peut représenter des points (moments) dans le temps, il est facile de représenter des intervalles en les représentant par leur point de départ et leur point de fin. Plus généralement, on peut considérer des ensembles d'intervalles temporels. Par exemple, un ensemble de deux intervalles temporels peut être utilisé pour représenter la période dans laquelle Julie a travaillé pour l'entreprise ACME du 1er juin au 1er september 2015, puis depuis le 1er juillet 2016 jusqu'à la fin de 2016.

Modèle de données temporelles. Le modèle que l'on va illustrer ici est le modèle *Bitemporal Conceptual Data Model (BCDM)*. Il s'agit d'un modèle conceptuel, qui permet de modéliser à la fois le temps de validité et le temps de transaction.

Prenons la situation suivante :

— Une année scolaire dure de septembre à juin.
— Cyrielle s'est inscrite pour le cours de BDD en septembre 2015. Elle était donc inscrite dans ce cours durant l'année scolaire 2015/2016.
— Ambroise s'est inscrit pour suivre le cours de BDD en juin 2016. Il est donc inscrit jusu'à la fin de l'année scolaire.
— Arrivent les résultats des examens en juin 2017, on apprend que Ambroise ne valide pas BDD, il est donc réinscrit (ce mois ci) pour le cours de BDD pour l'année 2017/2018.

Dans la figure 5 on peut voir ces faits sur un graphique dont les deux axes sont le temps de validité en ordonnée, et respectivement en abscisse le temps de transaction. La figure montre qu'en septembre 2016 on savait que Cyrielle a été inscrite au cours depuis 2015, et Ambroise l'est pour un an (2016-2017). On voit bien que c'est qu'à partir de juin 2017 que l'on sait qu'Ambroise a été inscrit sur deux ans (2016-2017, et 2017-2018).

La table 3 illustre une façon présentée par Jensen pour stocker les données temporalles dans une table. On a utilisé les notations suivantes : T_d qui représente le point de départ d'un intervalle de temps de transaction et T_f le point de fin, de même V_d et V_f indiquent le début et la fin d'un intervalle de temps de validité. De plus, UC sert à modéliser "aujourd'hui" (le moment courant). Ce symbole est typiquement utilisé pour modéliser la fin de l'intervalle validité des informations que nous pensons encore valides.

Les requêtes. La dimension temporelle rajoutée aux données peut être exploitées à l'aide de requêtes SQL élargies. Il serait possible de faire cela avec le langage SQL standard, en représentant le temps par des attributs et en utilisant des prédicats logiques SQL pour exprimer les conditions souhaitées sur le temps. Cela permettrait, dans le cas de notre exemple, de savoir qui a suivi quel cours et quand. Mais cela demande d'écrire des requêtes SQL très complexes.

C'est pourquoi de nombreux langages ont été définis, la plupart sont des extensions de SQL, comme par exemple TSQL2 [9].

Prenons un exemple en requêtant notre table INSCRIPTION. Cherchons à savoir qui est inscrit au cours de BDD pour l'année scolaire 2016/2017, c'est à dire qui était inscrit (en parlant de temps de validité) en septembre 2016. La requête s'écrit :

```
VALIDTIME SELECT Prenom
FROM INSCRIPTION
WHERE V= 'septembre 2016'
```

Le résultat obtenu sera ici seulement *Ambroise*. En effet seul Ambroise est inscrit en septembre 2016. Cyrielle l'a été mais pour l'année précédente.

Maintenant nous souhaitons chercher tous les élèves qui, en septembre 2016, étaient connus comme inscrits au cours de BDD. Ici on s'intéresse donc au temps de transaction. La requête s'écrit :

```
TRANSACTIONTIME SELECT Prenom
FROM INSCRIPTION
WHERE T= 'septembre 2016'
```

Le résultat ici est *(Ambroise, Cyrielle)*. En effet Cyrielle s'était inscrite en septembre 2015 et ce fait était toujours stocké dans la base en septembre 2016 ; Ambroise venait de s'y inscrire.

3.2 Modèles temporels pour RDF

Nous nous intéressons maintenant à comment enrichir le modèle RDF avec une représentation du temps. Section 3.2.1 montre comment réaliser cela par une approche à base d'étiquetage. Ensuite nous nous pencherons sur le cas où notre information temporelle est incomplète. Puis nous verrons une manière différente de mettre du temps avec son langage de requêtes. Et finalement nous verrons une manière d'annoter RDF qui d'adapte assez bien à l'aspect temporel.

3.2.1 Etiquetage des triplets

Nous présentons ici une première approche dont les idées sont exposées dans [4].

Concepts Une des possibilités consiste à étiqueter les triplets par les dimensions temporelles qu'on y attache ; celles-ci deviennent ainsi des propriétés du triplet. Cette approche peut supporter les deux dimensions du temps (temps de validité et de transaction), cela dit c'est plutôt le temps de validité qui est étudié. L'étiquetage a été fait conserve l'esprit d'extensibilité et de description des aspects différents de la réalité par des propriétés différentes, spécifique à RDF.

Une autre possbilité est de faire du *versioning*, c'est à dire de stocker des versions successives du graphe tel qu'il évolue dans le temps. Dans le cas où des changements sont fréquents, l'approche par versions successives peut rendre complexet coûteux le requêtage du graphe ; c'est la raison pour laquelle l'étiquetage est plutôt retenu.

Encodage du temps Le temps est encodé de manière discrète et linéairement ordonné. Le domaine temporel est représenté par des points temporels, et les intervalles seront donc définis par un point de départ et un point de fin.

Notations Un triplet avec une étiquette de temps est noté $(s, p, o) : [t]$, où t denote un point temporel. On utilise aussi $(s, p, o) : [t_1, t_2]$ pour ajouter les contraintes suivantes sur $t : t_1 \leq t \leq t_2$. Pour désigner le jour d'aujourd'hui on utilise le point temporel *maintenant*.

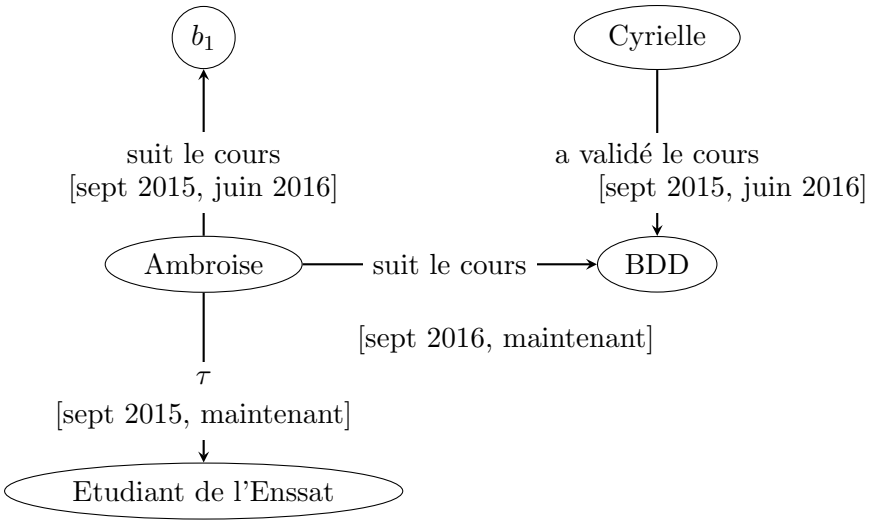

FIGURE 6 – Graphe RDF temporel.

Exemple Prenons un exemple de graphe temporel. On se place dans un contexte où l'on annote seulement avec le temps de validité.

On peut prendre notre exemple précédent en RDF auquel on rajoute du temps. On rajoute les informations suivantes :

— Cyrielle a validé le cours de BDD qui s'est déroulé de septembre 2015 à juin 2016. Ambroise a suivi un cours pendant la même période ;
— Ambroise suit le cours de BDD de septembre 2016 à maintenant ;
— Ambroise est étudiant à l'ENSSAT à partir de septembre 2015 jusqu'à maintenant.

Cet exemple est illustré dans la Figure 6.

Déduction Un des intérêts des graphes RDF est de permettre la déduction (le raisonnement). Celle-ci reste possible sur des graphes temporels. Une notion de saturation de graphe temporel est donc proposée, ainsi que des règles d'inférence associées. Pour définir la saturation, on utilise la notion de *snapshot* : cela consiste à regarder quels sont les triplets valides à un instant t.

Le problème de savoir si $\mathcal{G}'$ est une fermeture de $\mathcal{G}$ est DP-complet [1].

Syntaxe Une question à résoudre est l'encodage concret de ces étiquettes dans le graphe RDF. On se rappelle que le modèle RDF n'autorise que des triplets, or, les étiquettes temporelles feraient un quatrième élément. Pour résoudre ce problème, il est proposé de *réifier* les triplets, c'est à dire de les représenter comme des noeuds, puis de décrire leur temps de validité comme une propriété spéciale, nommée *tpl* (temporel), dont la valeur peut être un instant ou un intervalle.

La Figure 7 illustre ceci en utilisant un intervalle. On a représenté ici le triplet "Cyrielle a validé le cours de BDD qui s'est déroulé de septembre 2015 à juin 2016". b_1 est le noeud qui représente la réification du triplet « Cyrielle a validé le cours de BDD », t_1 est le temps qui est rattaché au

1. La classe de complexité DP est la classe des langages qui peuvent être écrits comme $L_1 \cap L_2$, où L_1 est un langage NP et L_2 est le complément d'un langage NP.

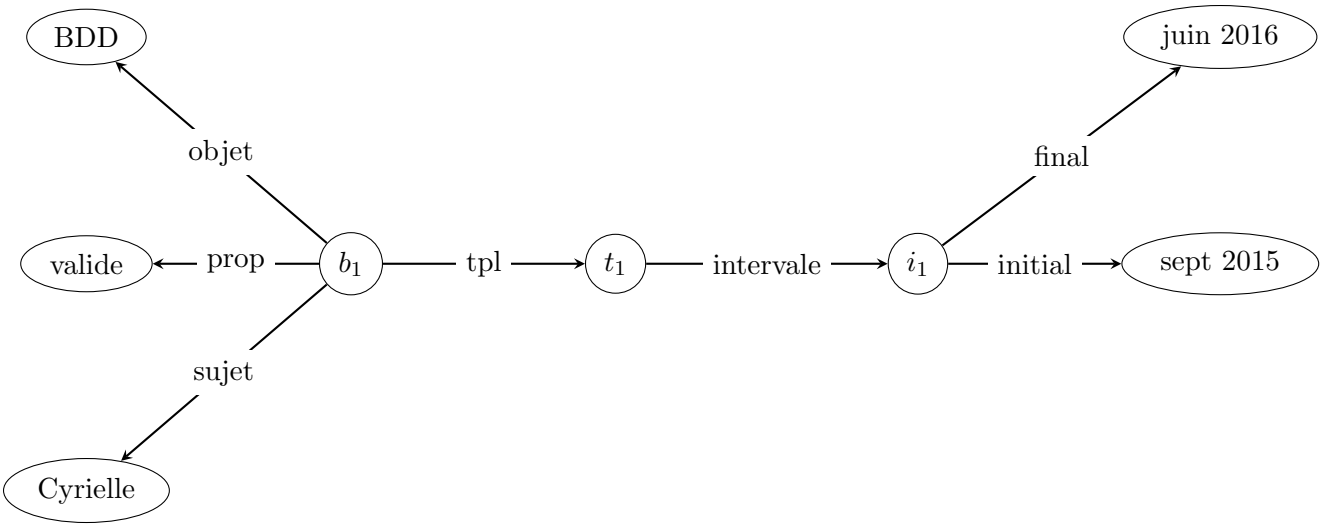

FIGURE 7 – Réification.

triplet par l'intermédiaire de b_1, et i_1 est l'intervalle de validité de notre triplet. Ses points initial et final sont explicités via des propriétés.

Les requêtes Un langage de requête est proposé afin de répondre à des requêtes conjonctives du type :

$$q(x_1) \leftarrow (x_1, a\ valide\ le\ cours, x_2) : [T], T < sept2017$$

Cette requête demande quels étudiants ont validé un cours avant septembre 2017. La réponse contient Cyrielle, avec x_2=BDD. On peut remarquer que pour cette requête il a fallu un langage permettant la comparaison de temps (intervalles et points temporels).

Gutierrez montre que l'ajout du temps dans les requêtes n'a pas un impact important surl'évaluation de requêtes.Il montre aussi que pour une base de données fixée, savoir si la réponse d'une requête est non vide est un problème NP-complet.

3.2.2 Extension : annotations temporelles potentiellement inconnues

Le contexte Hurtado [6] propose d'enrichir le modèle proposé par Gutierrez. En effet, il propose de raisonner sur le temps, et d'ajouter "un ensemble d'intervalles anonymes". Ce qui va permettre aux informations temporelles d'être incomplètes comme si on pouvait mettre des noeuds blancs temporels, seulement ce n'est pas le même ensemble de variables.

Nous allons ci-dessous nous intéresser plus particulièrement aux intervalles.

Les contraintes Le fait d'ajouter des intervalles anonymes ne permet plus de comparer un intervalle à un autre. En effet, si on connait la date de début et de fin de chaque intervalle, on peut les placer sur une frise chronologique assez facilement. Lorsque des intervalles anonymes sont introduits, on peut par exemple connaître le début d'un intervalle mais pas sa fin, ce qui peut être embêtant par exemple pour le comparer avec un autre dont on connaîtrait la fin mais pas le début. C'est pourquoi les auteurs de cet article proposent d'utiliser les concepts de la logique d'Allen, décrits dans la Table 4, comme contraintes.

Relations	Signification
$[l_1,l_2]$ before $[l_3,l_4]$	$l_2 < l_3$
$[l_1,l_2]$ meets $[l_3,l_4]$	$l_2 = l_3$
$[l_1,l_2]$ overlaps $[l_3,l_4]$	$l_3 < l_2 < l_4$ et $l_1 < l_3 < l_2$
$[l_1,l_2]$ starts $[l_3,l_4]$	$l_1 = l_3$ et $l_2 < l_4$
$[l_1,l_2]$ during $[l_3,l_4]$	$l_3 < l_1$ et $l_2 < l_4$
$[l_1,l_2]$ ends $[l_3,l_4]$	$l_1 > l_3$ et $l_2 = l_4$
$[l_1,l_2]$ equals $[l_3,l_4]$	$l_1 = l_3$ et $l_2 = l_4$

TABLE 4 – Relations possibles entre intervalles temporels, connues sous le noms de logique d'Allen.

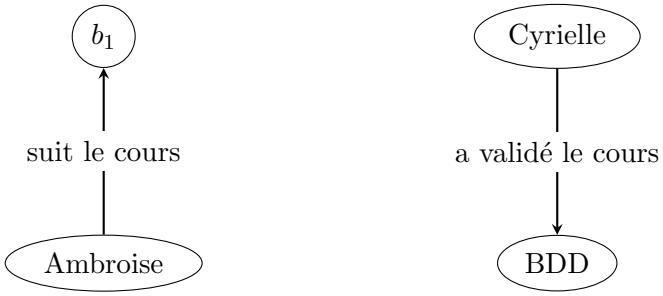

FIGURE 8 – Graphe nommé de l'intervalle [sept 2015, juin 2016].

La déduction Bien que cet article soit inspiré du précédent, les auteurs sont contraints de redéfinir les règles de déduction et la saturation des graphes temporels qui correspondent à leur nouveau modèle enrichi. La complexité pour trouver la saturation d'un graphe est polynomiale. Le problème de savoir si $\mathcal{G}'$ implique $\mathcal{G}$ est NP-complet [6].

Les requêtes Les auteurs n'introduisent pas un langage de requête spécifique à ce modèle.

3.2.3 Applications du modèle RDF temporel

Graphe nommé La solution présentée par Tappolet [10] est plus appliquée et concerne le temps de validité. Elle consiste à prendre en compte un graphe pour un intervalle de temps ; on retrouve l'idée du *snapshot*, ou encore du versioning. On utilise ainsi un *graphe nommé* pour chaque intervalle.

Par exemple, la figure 8 représente le graphe nommé qui correspond à l'intervalle [sept 2015, juin 2016].

Ce choix de faire du versionning a été fait pour raison d'efficacité. En utilisant la réification, beaucoup plus de triplets sont utilisés pour exprimer la même quantité d'information. C'est donc une question d'espace de stockage et de temps de récupération des données.

Modèle de données Pour gérer les triplets, les auteurs proposent un ensemble de simplifications ou restrictions du modèle de données. Ainsi, ils proposent de renommer les noeuds blancs en des URI (ceci change la sémantique du graphe RDF), et interdit les doublons de triplets . De plus, on impose qu'aucun triplet ne peut être dans deux graphes nommés différents.

En ce qui concerne le temps, ils utilisent OWL-Time [5], qui est une extension de RDFS.

Langage de requête Une partie importante de cet article porte sur l'introduction du langage τ-SPARQL, qui est une extension de SPARQL. Voici un exemple de requête en τ-SPARQL :

```
SELECT ?etudiant
FROM SNAPSHOT septembre 2016
WHERE ?etudiant τ Etudiant de l'ENSSAT
```

Cette requête permet de trouver des réponses vraies à un moment précis, en cherchant dans les graphes dont les intervalles contiennent ce moment.

3.3 Annotation temporelle via un framework d'annotation RDF

Contexte Lopes [8] propose une façon d'annoter RDF dénommée *Annotated RDFS* de manière assez générale et un langage de requête *AnQL* correspondant.

Annotated RDFS Les annotations proposées ressemblent à la façon dont proposée dans [4]. On associe à un triplet une annotation qui peut être de la forme :

$$(Cyrielle, avalide, BDD) : [sept2015, juin2016]$$

Ici j'ai donné l'exemple d'une annotation temporelle sous forme d'intervalle, qui correspond à notre exemple habituel. Mais il est aussi possible, d'utiliser d'autres annotations comme un paramètre flou. Par exemple :

$$(Cyrielle, \tau, Riche) : 0,4$$

Ce qui veut dire que Cyrielle est relativement riche, $\ll$ seulement à 40% $\gg$. De même on pourrait mettre une valeur de croyance. Le dernier exemple d'utilisation de ces annotations est la provenance. On pourrait annoter de quel document vient les triplets.

AnQL Ce langage est une extension de SPARQL pour requêter les annotations. Ce n'est pas un langage qui se restreint aux requêtes conjonctives, il a été étendu à des pattern de graphes annotés : (P AND P'), (P OPTIONAL P'), (P UNION P'), (P FILTER R) où P et P' sont des graphes annotés et R une expression de filtre.

Exemples avec OPTIONAL :

```
SELECT ?Prenom ?matiere ?t
WHERE (?Prenom, suit, ?matiere) : ?t OPTIONAL?Prenom, τ, Etudiant de l'ENSSAT : ?t
```

Le résultat de la requête sera (Ambroise, BDD, [sept 2016,maintenant]) et (Ambroise, b_1,[sept 2015, juin 2016]).

Ce langage de requête a tout de même une limite pour le temporel. Si l'on a deux personnes (p_1 et p_2) qui vivent à Paris, que p_1 y a vécu et est parti pendant 10 ans avant d'y revenir et que p_2 y a habiter pendant cette période de 10 ans. Si l'on cherche qui y a habiter avant l'autre p_2 va quand même nous être donné comme réponse, alors que ce n'est pas une réponse attendue.

id	hypothèse	matière	croyances
f1	stats	maths	A $f1^+$, C $f1^-$
f2	interro surprise	maths	A $f2^+$, C $f2^+$

Table 5 – Table des faits.

id	fait	commentaire	croyances
c1	f1	le sujet est tombé l'année dernière	C $c1^+$

Table 6 – Table des commentaires.

4 Les Points de vue

Les travaux sur les points sont principalement pour les bases de données relationnelles. Cependant, on peut voir que dans [8] le modèle proposé peut supporter d'autres annotation que le temps. On remarque aussi que [7] parle des différents modèles de temps, et sur le temps de validité on peut percevoir qu'il y a une notion assez subjective du temps (les différent mini-mondes). On peut donc imaginer des notions de temps qui dépendent de points de vues.

4.1 Croyances

Contexte Cet article de Gatterbauer [3] consiste à mettre des annotations sur des faits dans des bases de données relationnelles. Ces annotations sont des annotations de croyance, elles permettent que chacun mette son avis sur une donnée. Par exemple lors de travaux colaboratifs, il se peut qu'une personne soit en désaccord avec le tuple qu'une autre a mis.

Exemple Prenons le cas où Ambroise et Cyrielle essai de deviner le sujet de l'examen de maths de cette année. Ambroise pense que le sujet sera des statistiques. Cyrielle ne pense pas que ce sera des statistiques qui sont tombées l'année précédente. Cyrielle veut mettre un commentaire sur le tuple d'Ambroise pour soulever que ce sujet est tombé l'année dernière. Ambroise ajoute qu'il y aura aussi une interrogation surprise. Cyrielle est d'accord avec ça.

On peut voir ces faits dans la Table 5 et le commentaire dans la Table 6. Pour la colonne croyance, A+ signifie qu'Ambroise y croit, C+ que Cyrielle y croit et C- qu'elle n'y croit pas.

Monde de croyance Dans l'article, sont définis des mondes de croyances. Cela correspond à faire des tables personnelles, où dans la colonne croyance il n'y aurait que des + ou -. Les faits où l'on a pas de croyance n'apparaissent pas.

La base de données de croyances serait donc la collection des tables de croyances.

Base théoriques Cet article se base sur la logique épistémique multi-agents. La sémantique de cette logique peut être représenté par la structure canonique de Kripke.

Requêtes Un langage de requête est proposé sous formes de requêtes conjonctives. Par exemple, en notant F ma table de faits :

q(X) :- X F^{-}(y,z,u), C_y F^{+}(y,z,u)

est une requête qui me permet de savoir qui a un avis négatif sur les tuples dont Cyrielle a un avis positif.

5 Conclusion

Après avoir vu le modèle RDF. Nous nous sommes intéressés à enrichir des données avec un aspect temporel et des points de vues. Pour ça, nous avons vu ce qu'il se fait dans le monde des bases de données relationnelles et plus récemment directement en RDF.

En ce qui concerne le temporel, on a vu qu'il y a avait deux aspects, le temps de validité et le temps de transaction. Il s'avère que dans le domaine journalistique les deux peuvent être intéressants.

En effet, si un journaliste écrit un article sur un candidat à la présidentielle et qu'une fois publié il se rend compte, qu'il a oublié un fait majeur. On peut déjà se demander si le fait avait bien eu lieu avant qu'il rende son article, là c'est le temps de validité qui rentre en compte. Puis on peut aussi se demander si il avait la possibilité d'accéder à ce fait, s'il était dans sa base de données, là c'est le temps de transaction qui rentre en compte.

Il va donc falloir essayer entre le versionning et la réification. Un point de départ pourrait être de faire des annotations qui nous permettent de mettre du temporel et des points de vues, comme vu précédemment avec l'article [8].

En effet, il serait intéressant de pouvoir rajouter les points de vues, cela pourrait par exemple aider à suivre les opinions et les dires d'un politique lors de sa campagne.

Pour cela, il y a plusieurs approches sur lesquelles je peux me diriger, il y a les bases théoriques du relationnel, sachant qu'en relationnel on est en CWA ce qui est plus restrictif que le RDF où l'on est en OWA. La deuxième approche est d'exploiter ce qui a été en anotation sur le RDF pour l'adapter aux points de vues.

Références

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] Damian Bursztyn, François Goasdoué, and Ioana Manolescu. Optimizing reformulation-based query answering in RDF. In *Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015, Brussels, Belgium, March 23-27, 2015.*, pages 265–276, 2015.

[3] Wolfgang Gatterbauer, Magdalena Balazinska, Nodira Khoussainova, and Dan Suciu. Believe it or not : Adding belief annotations to databases. *PVLDB*, 2(1) :1–12, 2009.

[4] Claudio Gutiérrez, Carlos A. Hurtado, and Alejandro A. Vaisman. Temporal RDF. In *The Semantic Web : Research and Applications, Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29 - June 1, 2005, Proceedings*, pages 93–107, 2005.

[5] Jerry R. Hobbs and Feng Pan. An ontology of time for the semantic web. *ACM Trans. Asian Lang. Inf. Process.*, 3(1) :66–85, 2004.

[6] Carlos A. Hurtado and Alejandro A. Vaisman. Reasoning with temporal constraints in RDF. In *Principles and Practice of Semantic Web Reasoning, 4th International Workshop, PPSWR 2006, Budva, Montenegro, June 10-11, 2006, Revised Selected Papers*, pages 164–178, 2006.

[7] Christian S. Jensen and Richard T. Snodgrass. Temporal data management. *IEEE Trans. Knowl. Data Eng.*, 11(1) :36–44, 1999.

[8] Nuno Lopes, Axel Polleres, Umberto Straccia, and Antoine Zimmermann. AnQL : SPARQLing Up Annotated RDFS. In *The 9th International Semantic Web Conference*, pages 518–533, 2010.

[9] Richard Snodgrass, Ilsoo Ahn, Gad Ariav, Don Batory, James Clifford, Curtis E. Dyreson, Ramez Elmasri, Fabio Grandi, Christian S. Jensen, Wolfgang Kaefer, Nick Kline, Krishna Kulkarni, T. Y. Cliff Leung, Nikos Lorentzos, John F. Roddick, Arie Segev, Michael D. Soo, and Suryanarayana M. Sripada. *The TSQL2 Temporal Query Language*. Kluwer Academic Publishers, 1995. 674+xxiv pages.

[10] Jonas Tappolet and Abraham Bernstein. Applied temporal RDF : efficient temporal querying of RDF data with SPARQL. In *The Semantic Web : Research and Applications, 6th European Semantic Web Conference, ESWC 2009, Heraklion, Crete, Greece, May 31-June 4, 2009, Proceedings*, pages 308–322, 2009.

[11] RDF. `https://www.w3.org/TR/rdf11-concepts`.

[12] RDF Schema. `https://www.w3.org/TR/rdf11-mt`.

[13] SPARQL 1.1 Query Language. `https://www.w3.org/TR/sparql11-query`.

[14] SPARQL 1.1 Entailment Regimes.