CubicWeb : vers un outil pour des applications clé en main dans le Web Sémantique

Fabien Amarger, Simon Chabot, Nicolas Chauvat, Elodie Thiéblin

CubicWeb : vers un outil pour des applications clé en main dans le Web Sémantique

Fabien Amarger[1], Simon Chabot[1], Nicolas Chauvat[1], Elodie Thiéblin[1]

LOGILAB, 104 boulevard Louis-Auguste Blanqui Paris, France
prénom.nom@logilab.fr

Résumé : CubicWeb est un cadriciel pour le développement d'applications Web, qui permet d'exposer des ressources en RDF et gère la négociation de contenu pour prendre part au Web de données liées.

Mots-clés : Web de données liées, Déréférencement, Cadriciel, Python, RDF, Développement d'applications

1 Introduction et motivations

Le développement de CubicWeb, un cadriciel libre écrit en Python, a commencé en 2001, année de publication de l'article fondateur (Berners-Lee *et al.*, 2001) et a été mené en ayant connaissance des bases de données relationnelles, des standards objets tels que UML (*Unified Modeling Language*) et MOF (*Meta-Object Facility*) et des travaux tels que DAML+OIL.

CubicWeb ayant été conçu pour faciliter la réalisation d'applications déployées pour les clients de Logilab, un choix initial a été de s'appuyer sur des fondations solides en matière de gestion de données, à savoir des bases de données relationnelles, plutôt que sur des *TripleStores*, jugés trop instables à l'époque. Dans le but de se rapprocher d'une part des modélisations de type UML et OWL et d'autre part des langages d'interrogation de graphes, deux langages spécifiques ont été créés : YAMS (*Yet Another Magic Schema* [1]) pour décrire les modèles et RQL (*Relation Query Language* [2]) pour interroger les données. CubicWeb traduit le modèle YAMS en un modèle SQL, compile le RQL en des requêtes SQL et fournit un moteur de génération semi-automatique de l'interface utilisateur. Sur ces bases, il est possible d'initialiser une application web complète à partir d'un simple modèle YAMS.

Pour mettre les applications en production, il a été nécessaire dès le début de traiter les questions relatives à l'authentification des utilisateurs, la gestion des droits, la personnalisation des interfaces, la migration des données au fur et à mesure de l'évolution des modèles...

Fort de ces fonctionnalités CubicWeb a pu répondre aux besoins de projets conséquents tels que DataBnf (`https://data.bnf.fr`) (Simon *et al.*, 2013), FranceArchives (`https://francearchives.fr/`) ou DataPOC (`https://datapoc.mnhn.fr/`).

Même si les idées et principes du Web sémantique étaient présents dès la création de CubicWeb, il a été difficile de suivre les processus de normalisation menés au W3C, car il aurait fallu dégager les ressources suffisantes pour faire évoluer le cœur de CubicWeb et les applications déjà déployées chez les clients. En 2009, une première tentative a été faite de permettre l'interrogation de CubicWeb en SPARQL [3] et de convertir des modèles de OWL vers YAMS. Nous avons repris cet effort en 2018 en y ajoutant l'interrogation en GraphQL, puis la négociation de contenu RDF.

Nous présentons ici l'architecture de CubicWeb et les derniers travaux en terme de négociation de contenu, puis nous détaillons un scénario d'utilisation et concluons en mettant CubicWeb en perspective avec d'autres travaux et en présentant la suite des travaux prévus.

---

1. `https://cubicweb.readthedocs.io/en/3.27/book/devrepo/datamodel/definition/`
2. `https://cubicweb.readthedocs.io/en/3.27/book/annexes/rql/language/`
3. `https://www.cubicweb.org/blogentry/344822`

2 Architecture de CubicWeb

CubicWeb fonctionne par composants, appelés *cubes* (`https://www.cubicweb.org/project`), exploitables par plusieurs applications. Il est possible de combiner plusieurs cubes pour créer une application (qui est elle-même un cube réutilisable). Un cube est composé :

i) d'un schéma (ou modèle données) exprimé en YAMS, un langage qui permet de représenter un schéma entité-association et les permissions associées en python ;

ii) d'une logique applicative ;

iii) de vues (interfaces graphiques, ou types d'export de données).

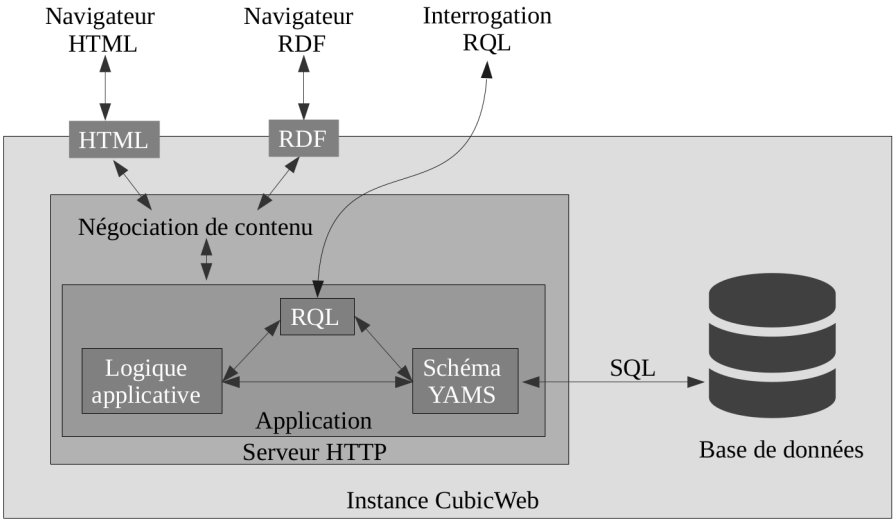

FIGURE *1 – Schéma d'une instance CubicWeb en fonctionnement*

Une instance applicative, appelée instance CubicWeb dans la suite de l'article, peut-être créée à partir d'un ou plusieurs cubes. À sa création, le modèle de données en YAMS est utilisé pour générer une base de données SQL qui sera alimentée au cours de l'utilisation de l'application. Les données dans CubicWeb sont gérées en monde fermé. La Figure 1 représente l'architecture d'une instance CubicWeb à l'utilisation. L'instance CubicWeb contient également un serveur HTTP qui gère (depuis peu) la négociation de contenu orientée serveur [4]. L'en-tête `Accept` d'une requête HTTP permet à un agent de préciser au serveur le format attendu lors de l'interrogation d'une ressource. La logique de l'application interagit avec la base de données par le biais du schéma YAMS et/ou du langage de requête RQL. Les vues de l'instance sont générées automatiquement mais un mécanisme de sélection de vues permet de dissocier données et interfaces pour faciliter leur gestion et leur personnalisation. Les vues contiennent également des interfaces d'administration permettant aux utilisateurs autorisés d'ajouter, éditer, supprimer les entités (instance d'un type YAMS). C'est la logique applicative qui se charge de générer la description en RDF d'une entité lorsque celle-ci est demandée. Les équivalences entre le schéma YAMS et la ou les ontologie(s) OWL utilisées dans l'export RDF sont spécifiées dans le cube ou les cubes sur lesquels repose l'instance CubicWeb.

4. `https://www.w3.org/TR/dwbp/#dataAccess`

3 Scénario d'utilisation

Prenons pour exemple le cube Blog [5], dans lequel les concepts clés sont les blogs et les billets de blogs, décrit en YAMS par :

```
class Blog(EntityType):
    title = String(maxsize=50, required=True)
    description = RichString()

class BlogEntry(WorkflowableEntityType):
    __permissions__ = {
        'read': (
            'managers',
            'users',
            ERQLExpression('X in_state S, S name "published"'),
        ),
        'add': ('managers', 'users'),
        'update': ('managers', 'owners'),
        'delete': ('managers', 'owners')
    }
    title = String(required=True, fulltextindexed=True)
    content = RichString(required=True, fulltextindexed=True)
    entry_of = SubjectRelation('Blog')
```

La classe `Blog` contient un titre et une description. La classe `BlogEntry`, qui décrit un billet de blog, contient un titre et un contenu. Une relation `entry_of` relie un billet de blog à un blog. La classe `BlogEntry` hérite de la classe `WorkflowableEntityType` : elle est liée à un `State` (par défaut *draft* ou *published*) par une relation `in_state`. Les billets de blog ont des permissions particulières définies grâce à l'attribut `__permissions__`. Ces permissions spécifient qu'un billet de blog est accessible en lecture par le groupe *managers*, le groupe *users* (un utilisateur connecté) et à tout le monde si le billet de blog est publié. Le groupe *owners* est un groupe virtuel, qui contient le propriétaire (généralement le créateur) de l'entité. Cet exemple simple illustre que CubicWeb permet la gestion des permissions sous la forme d'*ACL* (*Access Control List*) et d'utiliser des requêtes *RQL* pour exprimer les permissions aussi finement que souhaité. CubicWeb offre un fonctionnement similaire concernant les relations.

Une instance CubicWeb basée sur le schéma YAMS ci-dessus, est publiquement accessible en ligne à cette adresse : `https://pfia.demo.logilab.fr/`. CubicWeb propose nativement une visualisation du schéma, servi sur le chemin relatif `/schema`, ainsi qu'une traduction en OWL de ce schéma YAMS, sur `/view?vid=owl`. Une interface d'aide à l'écriture de requêtes RQL est disponible, *via* un cube spécifique, sur `/browse`.

Un blog contenant un billet de Blog ont été créés par l'administrateur de l'instance CubicWeb. Le billet de blog exemple se trouve à l'adresse `https://pfia.demo.logilab.fr/blogentry/992`. Le cube "Blog" implémente la génération de triplets RDF en fonction de son schéma et permet donc la négociation de contenu. Nous souhaitons récupérer la description en RDF de ce billet de blog au format turtle. Pour cela, nous pouvons exécuter la commande suivante :

```
curl -H "Accept: text/turtle" https://pfia.demo.logilab.fr/blogentry/992
```

Nous obtenons alors la réponse suivante :

```
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix sioc: <http://rdfs.org/sioc/types#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<https://pfia.demo.logilab.fr/992> a sioc:BlogPost ;
    dcterms:date "2020-02-12T16:02:00.439309+00:00"^^xsd:dateTime ;
    dcterms:modified "2020-02-12T16:02:18.030359+00:00"^^xsd:dateTime ;
    dcterms:title "un premier billet de blog" ;
    sioc:container <https://pfia.demo.logilab.fr/989> ;
    sioc:content "voici le contenu de mon premier billet de blog" .
```

5. `https://www.cubicweb.org/project/cubicweb-blog`

4 Conclusion et perspectives

Plusieurs outils permettent la gestion et la publication de données RDF sur le Web (Heath & Bizer, 2011). Par exemple, les plateformes de données liées telles que Apache Marmotta [6] ou CarbonLDP [7] supportent la négociation de contenu et fournissent une interface d'administration des données mais ne permettent pas de créer une application avec interfaces graphiques de manière directe. Pubby [8] permet, à l'inverse, de partir d'un endpoint SPARQL pour générer des interfaces de consultation des ressources. PSPS [9] se base sur des données RDF existantes et des vues personnalisables pour générer un site Web, sans toutefois offrir d'interface d'administration. Enfin, Virtuoso [10] dans sa version payante offre des interfaces d'administration et de consultation.

CubicWeb permet de publier des données sur le Web et fournit des interfaces d'administration et de consultation de ces ressources dont les permissions sont gérées finement.

Dans la notion de « *clé en main* », il nous semble important qu'un créateur d'application puisse importer directement des données formalisées en OWL et RDF dans CubicWeb. Des travaux pour extraire un schéma YAMS d'une ontologie ont été initiés et il sera important de déterminer quel fragment de OWL peut être traduit sans perte. Ensuite, le peuplement de la base de données CubicWeb à partir de ressources RDF pourra être effectué automatiquement. Afin de conserver l'esprit de mutualisation des fonctionnalités de CubicWeb, un alignement entre l'ontologie en entrée et les classes définies dans les cubes existants peut être envisagé. Par exemple si l'ontologie fait référence à la classe `sioc:Blog`, utiliser le cube `Blog` plutôt que de recréer une classe du même nom.

Toujours dans la même dynamique d'alignement vers les standards du Web Sémantique, nous avons commencé à rendre possible l'interrogation en SPARQL des données CubicWeb. Pour l'instant il n'est possible que d'effectuer des requêtes simples de projection avec uniquement des triplets sans variable pour les relations. Néanmoins nous envisageons de continuer ces travaux pour prendre en compte un maximum d'éléments de SPARQL sans prétendre arriver à couvrir toute la recommandation.

En suivant ces perspectives, nous espérons faciliter la publication de données sur le Web de données liées, leur administration et le développement d'applications les utilisant. Une première étape a été de proposer la négociation de contenu nativement dans CubicWeb. La gestion des requêtes SPARQL, l'import d'ontologies OWL et de données RDF permettront une intégration complète de la chaîne de publication. Nous pourrons alors répondre à la question « *Une fois que nous avons nos données en RDF, qu'en faisons nous ?* » par la réponse « *Mettons les dans CubicWeb pour en faciliter la découverte, la consultation, le partage et* in fine *l'accès à de nouveaux savoirs* ».

Références

BERNERS-LEE T., HENDLER J. & LASSILA O. (2001). The semantic web. *Scientific american*, **284**(5), 34–43.

HEATH T. & BIZER C. (2011). *Linked Data : Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers.

SIMON A., WENZ R., MICHEL V. & MASCIO A. D. (2013). Publishing bibliographic records on the web of data : Opportunities for the bnf (french national library). In *ESWC*, volume 7882 of *Lecture Notes in Computer Science*, p. 563–577 : Springer.

6. `https://marmotta.apache.org`
7. `https://carbonldp.com/`
8. `http://wifo5-03.informatik.uni-mannheim.de/pubby`
9. `https://github.com/factsmission/psps`
10. `https://virtuoso.openlinksw.com`