Combining and Uniting Business Intelligence with Semantic Technologies
Acronym: CUBIST Project No: 257403
Small or Medium-scale Focused Research Project FP7-ICT-2009-5 Duration: 2010/10/01-2013/09/30

FCA Integration in the Triple Store, Version #1

This document describes the FCA Service component, which is part of the 1st version of the CUBIST integrated prototype.

Type:	Prototype
Document ID:	CUBIST D3.3.1
Workpackage:	WP3
Leading partner:	SHU
Author(s):	Constantinos Orphanides (SHU)
Dissemination level:	Public
Status:	Final
Date:	12 April 2012
Version:	0.7

Versioning and contribution history

Version	Description	Contributors
0.1	Draft	Constantinos Orphanides (SHU)
0.2	Feedback / Review	Cassio Melo (CRSA)
0.3	Incorporation of feedback from Cassio Melo	Constantinos Orphanides (SHU)
0.4	Feedback / Review	Alex Simov (Ontotext)
0.5	Incorporation of feedback from Alex Simov	Constantinos Orphanides (SHU)
0.6	Final Draft	Constantinos Orphanides (SHU)
0.7	Finalized Document	Constantinos Orphanides (SHU)

Reviewers

Name	Affiliation
Cassio Melo	CRSA
Alex Simov	Ontotext

Contents

1 Introduction

This document provides an overview of the FCA Service component in CUBIST. This component is used to create formal contexts out of a triple store. The NowaSearch front-end component issues a request to the FCA Service to create a formal context, to be then visualised as a concept lattice by the CUBIX Visual Analytics component.

2 Overview

2.1 Installation and Binaries

For instructions on how to install the FCA Service and to gain access to the binary file please refer to D1.3.1.

2.2 Architecture

The FCA service is built using C# on the Microsoft .NET 4 Framework, using the REpresentational State Transfer[1] (REST) architecture.

2.2.1 Web-Methods

The FCA service exposes two web-methods which are used to create a formal context out of a triple store. The two web-methods accept input and return output using the JavaScript Object Notation[2] (JSON) format. The web-methods are explained below:

Resource	URL	Description
POST formalcontext	http://cubist.hallam.shu.ac.uk/FcaBedrock.svc/formalcontext	This method is an HTTP POST method which accepts the following parameters: <ul><li>repositoryConnection: the information needed to connect to a particular triple store.</li><li>repositoryId: The ID of the repository in the triple store.</li><li>sparqlQuery: The SPARQL query that the FCA service will execute on the triple store to fetch data.</li><li>minSuppObjs: The minimum-support for objects that the High Performance Concept Miner (In-Close) component should apply to the formal context created by the FCA Service.</li></ul>

[1] http://en.wikipedia.org/wiki/Representational_state_transfer
[2] http://en.wikipedia.org/wiki/JSON

		• minSuppAtts: The minimum-support for objects that the High Performance Concept Miner (InClose) component should apply to the formal context created by the FCA Service. The web-method creates a formal context based on parameters 1-5 and returns a unique formal context ID (which represents the formal context created) to the consumer of the service. The formal context ID can be then used to retrieve the actual formal context
GET formalcontext?id={FormalContextID}&format=json	http://cubist.hallam.shu.ac.uk/FcaBedrock.svc/formalcontext?id=030412225657&format=json	This method retrieves the formal context having the FormalContextID issued with the request, or null if the particular ID does not exist.

3 An Example

Following is an example scenario demonstrating how the FCA Service can be used to create formal contexts out of a triple store.

Let us say that we are querying the data from the HWU Use-Case and we are interested in finding which Tissues exist in Theiler Stage 7 and which Genes (if any) are expressed, in each of the Tissues, during that Theiler Stage.

The following HTTP POST JSON request is issued on http://cubist.hallam.shu.ac.uk/FcaBedrock.svc/formalcontext:

```
{
  "sparqlQuery":"PREFIX rdfs:<http://www.w3.org/2000/01/rdf-
schema#> PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-
ns#> PREFIX hwu:<http://www.cubist_project.eu/HWU#> PREFIX :<http://www.cubist_projec
t.eu/HWU#> PREFIX owl:<http://www.w3.org/2002/07/owl#> PREFIX xsd:<http://www.w3.or
g/2001/XMLSchema#> SELECT DISTINCT ?o1 ?a1 WHERE { ?x1 rdf:type :Tissue ; rdfs:label
?o1 . ?x1 hwu:has_theiler_stage hwu:theiler_stage_07 . OPTIONAL { ?x3 rdf:type :Gene ; rdfs:
label ?a1 .  ?x2 rdf:type hwu: Textual_Annotation .  ?x2 hwu:in_tissue ?x1 .  ?x2 hwu:has_inv
olved_gene ?x3 .  ?x2 hwu:has_strength hwu:level_detected .} } ORDER BY ?o1 ? a1",
  "repositoryId":"CUBISTHWU",
  "repositoryConnection":"http://cubist.hallam.shu.ac.uk:8080/openrdf-sesame",
  "minSuppObjs":"0",
  "minSuppAtts":"0"
}
```

Based on the request above, the FCA Service does the following:

1) Connects to triple store located at *http://cubist.hallam.shu.ac.uk:8080/openrdf-sesame*

2) Sets *CUBISTHWU* as the repository to be queried

3) Executes the following SPARQL query on repository *CUBISTHWU:*

```
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX hwu:<http://www.cubist_project.eu/HWU#>
PREFIX :<http://www.cubist_project.eu/HWU#>
PREFIX owl:<http://www.w3.org/2002/07/owl#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
SELECT DISTINCT ?o1 ?a1 WHERE {
```

?x1 rdf:type hwu:Tissue ; rdfs:label ?o1 .
?x1 hwu:has_theiler_stage hwu:theiler_stage_07 .
OPTIONAL
{ ?x3 rdf:type hwu:Gene ; rdfs:label ?a1 .
?x2 rdf:type hwu:Textual_Annotation .
?x2 hwu:in_tissue ?x1 .
?x2 hwu:has_involved_gene ?x3 .
?x2 hwu:has_strength hwu:level_detected .}
}
ORDER BY ?o1 ?a1

4) Creates a formal context out of the data returned by the SPARQL query on step 3 and applies minimum-support to the formal context, using In-Close.
5) Assigns a unique formal context ID to the formal context.
6) Returns the formal context ID to the consumer, e.g. *100412163045*

The consumer can then issue an HTTP GET request on http://cubist.hallam.shu.ac.uk/FcaBedrock.svc/formalcontext?id={id}&format=json (by replacing {id} with the actual formal context ID) to retrieve the formal context. An example of what the output the FCA Service produces looks like is shown in Figure 1 below.

```
[
   "B",
   "",
   "15",
   "19",
   "",
   "11685",
   "11682",
   "11684",
   "707",
   "5209",
   "5630",
   "5126",
   "11683",
   "5631",
   "2012",
   "6592",
   "774",
   "773",
   "5328",
   "5327",
   "Gene-Bmp4",
   "Strength-detected",
   "Theiler_Stage-20",
   "Theiler_Stage-19",
   "Theiler_Stage-18",
    "Tissue-epithelium",
    "Tissue-inner ear",
    "Strength-strong",
    "Tissue-embryo",
    "Tissue-telencephalon",
    "Tissue-mesenchyme",
    "Tissue-medial-nasal process",
    "Tissue-mandibular component",
    "Tissue-apical ectodermal ridge",
    "Tissue-otocyst",
    "Tissue-eye",
    "Tissue-handplate",
    "Tissue-latero-nasal process",
    "Tissue-footplate",
    "XXX...............X",
    "X.X....X.........X.",
    "XXX.............X..",
    "XX..X..........X...",
    "XX..X.........X....",
    "XX.X.........X.....",
    "XX..X.......X......",
    "X.X....X...X.......",
    "XX.X......X........",
    "XXX......X.........",
    "XX.X....X..........",
    "XX.X..X............",
    "XX.X..X............",
    "XXX..X.............",
    "XXX..X............."
]
```

Figure 1: A formal context returned (in JSON format) by issuing the HTTP GET request above.

Consumers of the request (e.g. CUBIX) can then use the formal context to build a formal concept lattice (Figure 2).

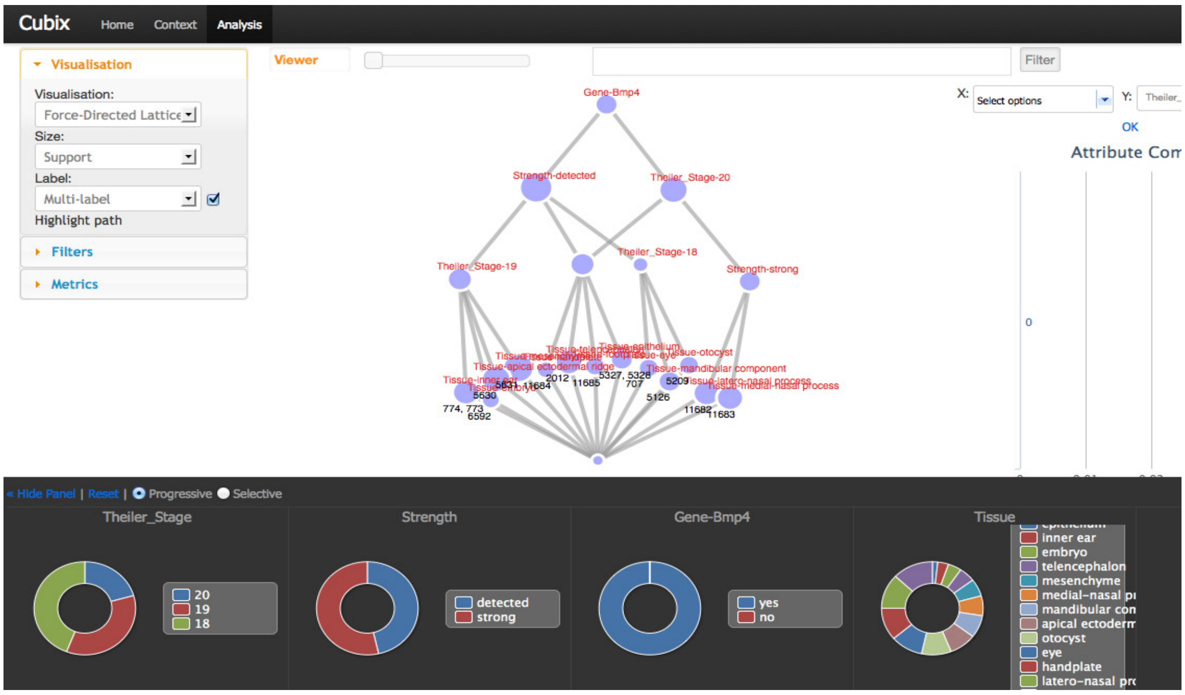

Figure 2: The formal context in Figure 1, visualised in CUBIX.