Project N°:	**FP7-318338**
Project Acronym:	**Optique**
Project Title:	**Scalable End-user Access to Big Data**
Instrument:	**Integrated Project**
Scheme:	**Information & Communication Technologies**

Deliverable D2.4
Second Prototype of the Optique Platform

Due date of deliverable:	(T0+24)
Actual submission date:	October 31, 2014

Start date of the project: **1st November 2012** Duration: **48 months**

Lead contractor for this deliverable: **FOP**

Dissemination level: **PU** – Public

Final version

Executive Summary:
Second Prototype of the Optique Platform

This document summarises deliverable D2.4 of project FP7-318338 (Optique), an Integrated Project supported by the 7th Framework Programme of the EC. Full information on this project, including the contents of this deliverable, is available online at `http://www.optique-project.eu/`.

This deliverable is a software deliverable and documents the second software prototype of the Optique system, which integrates the software components from the technical workpackages into a central platform. The deliverable describes the design and functionality of shared platform interfaces and explains how the modules for ontology and mapping management, query transformation, query execution and visual query formulation are integrated into the platform. Finally, the document guides the reader through different administration task as well as explains basic the process of visual query building to both IT-Experts and End-User.

List of Authors

Johannes Trame (FOP)
Peter Haase (FOP)
Dimitris Bilidas (UoA)
Ernesto Jiménez-Ruiz (UOXF)
Artem Kozlov (FOP)
Jose Mora (UNIROMA1)
Christoph Pinkel (FOP)
Martín Rezk (FUB)
Ahmet Soylu (UiO)

Contributors

Marco Console (UNIROMA1)
Martin Giese (UiO)
Dag Hovland (UiO)
Evgeny Kharlamov (UOXF)
Marco Ruzzi (UNIROMA1)
Martin G. Skjæveland (UiO)
Guohui Xiao (FUB)

Contents

Chapter 1

Introduction

1.1 Background

A typical problem that end-users face when dealing with Big Data is that of data access. Due to the increasing *volume* of data, the *velocity* of data growth as well as due to the *variety* of different data formats accessing the *relevant* information becomes a difficult task. In situations where an end-user needs data that predefined queries do not provide, the help of IT-experts (e.g., database managers) is required to translate the information need of end-users to specialised queries and optimise them for efficient execution. This process may require several iterations of communication between end users and IT-experts and can take up to several days [2]. This becomes the bottleneck of data access. In Optique we propose to rely on the "Ontology-Based Data Access" (OBDA) to address this problem.

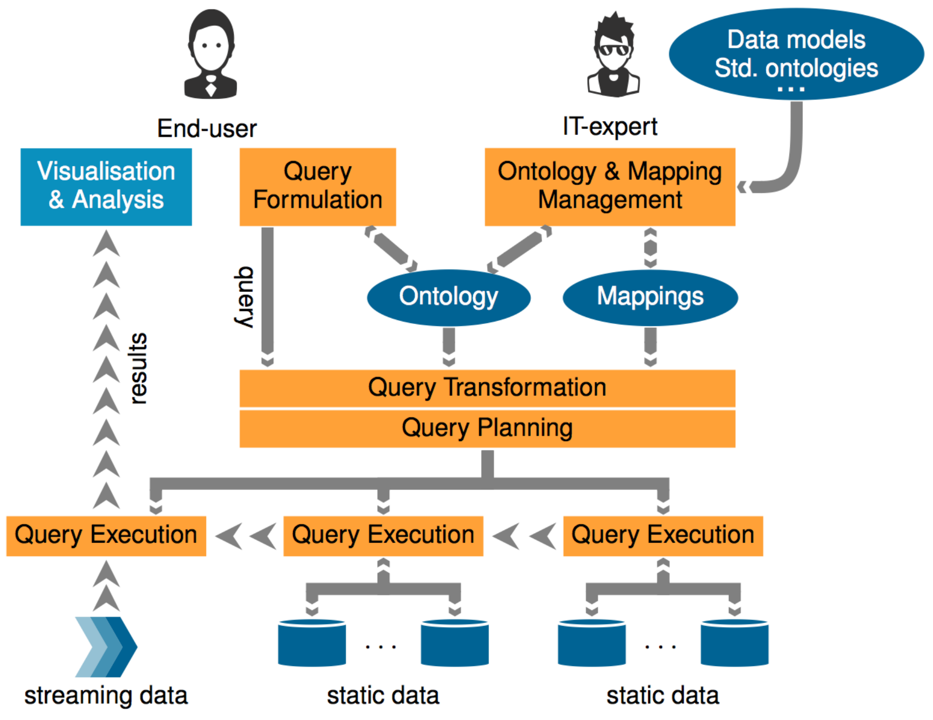


Figure 1.1: The Optique OBDA system

The key idea of OBDA [3, 1] is to use an ontology, which presents to users a semantically rich conceptual model of the problem domain. The users formulate their information requirements (that is, queries) in terms of the ontology, and then receive the answers in the same intelligible form. These requests should be executed over the data automatically, without an IT-expert's intervention. To this end, a set of mappings is maintained which describes the relationship between the terms in the ontology and the corresponding terminology in the data source specifications, e.g., table and column names in relational database schemas. State of the art OBDA systems typically do not address several important challenges that limit their applicability in industry. In particular, they do not offer integration of temporal and streaming data, provide a limited query formulation support for end-users, and a limited system deployment and maintenance support for IT-Experts. The Optique projects aims at developing a next generation OBDA system (cf. Figure 1.1) that overcomes these limitations. A bigger goal of the project is to provide a platform with a generic architecture that can be adapted to any domain that requires scalable data access and efficient query execution.

1.2 Summary of changes and contributions

This document is a successor of the Deliverable 2.3 and has been changed and extended to document the second year prototype in a self-contained way. In the following we summarize contributions in terms of changes over the initial prototype:

- New data source concept & metadata management
  - connecting to disparate data sources through templates and previews
  - support for multiple schemas and different database vendors
  - fully transparent meta data gathering & caching
  - extensible to support new database vendors and types of data sources
- New mapping catalogue and fully integrated, build-in mapping editor
  - fully standard compliant, based on R2RML mapping language
  - auto-suggestions and pre-populated dropdowns during mapping manipulation taking metadata and ontology into account
  - access to all metadata
  - in-line SQL editor, raw data previews
  - triple preview
- Improved ontology management facilities
  - easy export, import, delete and visualization of ontologies
- New query catalogue
  - infrastructure to store, retrieve and execute queries as first order objects from the shared metadata-repository
  - standard compliant, based on SPIN modelling language
  - support for parametrization
  - queries are stored as first order objects and can be reference through-out different places in the platform
- New facilities for platform configuration & debugging (query time-out, log browser)
- New platform life-cycle listener
  - proper registration and configuration of components and services during different states of the platforms' life-cycle
  - enables, for example, seamless registration of new result writers as such as for the new KML/GIS export
- New and improved visualization and export capabilities
  - visualization for large data point collections
  - initial widgets for streamified result visualization
- Built-In SPARQL federation technology (FedX)

  - fully transparent SPARQL federation over different types of semantic repositories
  - covers extended use-cases, complements SQL federation
- Extended integration of component for ontology and mapping management
  - back-end integration of the component for mapping analysis
  - front-end integration of the mapping analysis into the mapping catalogue as well as into the mapping editor
  - close integration of the module for ontology and mapping bootstrapping with the mapping editor and ontology catalogue
- Extended integration of ADP component
  - ADP implementation of the new data source concept for standard, distributed JDBC mode
  - "ADP Federated" implementation of the new data source concept for JDBC federation mode
  - initial templates for automated provisioning using the platforms native cloud management functionalities
- Closer integration of the module for visual query formulation
  - implementation of the platform widget concept to ease configuration and improve back-end-communication
  - integration of extended back-end functionality
  - integration with query catalogue (queries can be stored to and re-opened from the catalogue)
  - initial version of the semi-automatic ontology annotation extractor has been integrated into the ontology catalogue
- Enhanced integration of the query transformation component
  - fully configurable, ability to add user defined database constraints
  - improved exception handling, simplified exception messages are propagated to the end-user
- Initial back-end integration of component for handling time and streams
  - implementation as an extension to the standard query transformation component
  - to operate on a newly implemented ADP "StreamingDataSource", instead of standard JDBC datasource
  - transparent registration and transformation of STAQRL queries to a StreamingDataSource
  - new visualization widget to operate on the streaming data source
- Extended mechanism to easy bundle, install and update modules and domain extensions to installations
- New structure and design of the administration dashboard as a single point of entry for administrative tasks

1.3 Structure of the document

In Chapter 2 we recall the architecture as specified in Deliverable 2.1, before we describe the design and implementation of shared platform interface and their role in integrating the different components. Chapter 3 documents the required steps to install and configure the platform as well as provide instructions for end-user. Chapter 4 summarizes the main achievements and platform features as well as provides an overview of ongoing development and engineering activities.

Chapter 2

Optique Platform Architecture & Implementation

2.1 Architectural Overview

The Optique architecture is designed as a tiered architecture with three layers (cf. Figure 2.1). While the presentation layer addresses specifically end-users needs such as query formulation and query answer visualisation, it provides also functionalities for IT-experts in order to set-up and (re)configure the platform.

The core components such as for query transformation, query answering, ontology and mapping management are integrated on the application layer and interact through specialized application programming interfaces. The data and resource layer is responsible for managing the access to different kind of (re)sources, for example, relational databases, data streams or even access to computational resources in the cloud.

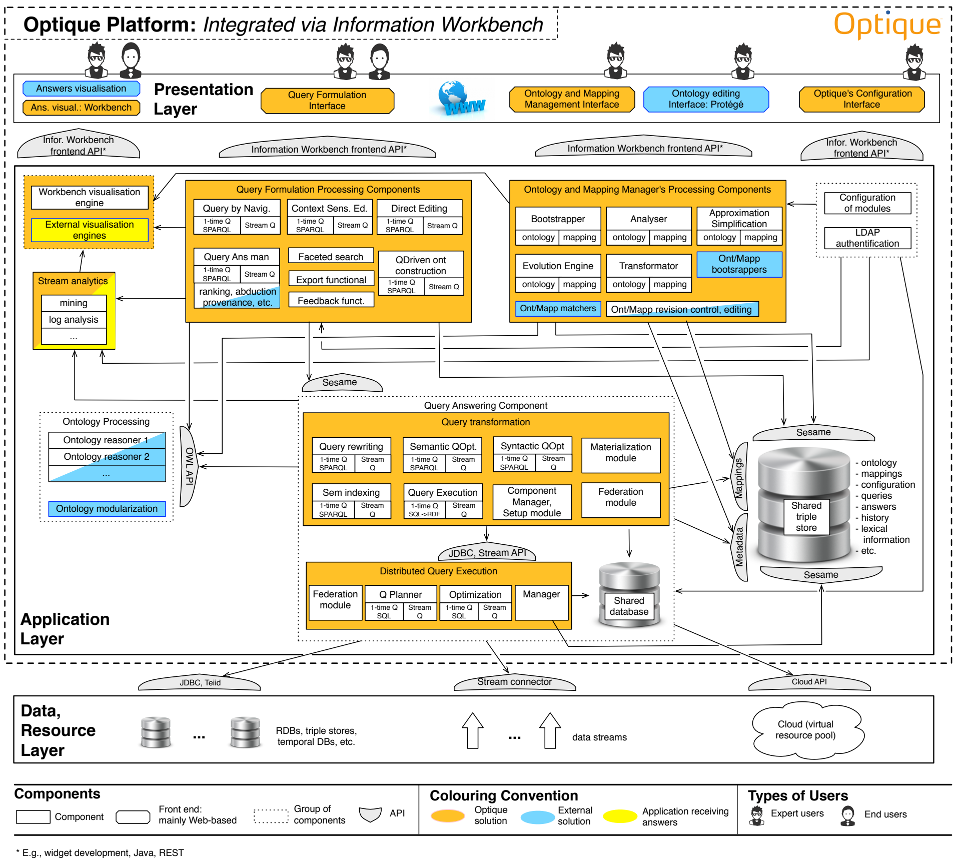


Figure 2.1: The general architecture of the Optique OBDA system

2.2 The Information Workbench as a Platform for Integration

The Optique platform based up on the Information Workbench, an industrial-strength and mature software platform developed and maintained by fluid Operations.[1] The Information Workbench is an open, data-centric development platform that has been specifically designed to support the whole lifecyle of interacting with semantic data – from integration to access, visualization, exploration, and data interaction.

While the Information Workbench provides already a number of extension points in order to automatically plug-in new modules, special programming interfaces have been designed and implemented according to the requirements as specified in Deliverable 2.1. This is to enable a tight integration and seamless interaction of the Optique software components.

2.3 Shared Platform Interfaces

We distinguish between (i) the shared interfaces among the Optique components and (ii) the interfaces provided by each component itself (e.g. APIs for query formulation). The concrete interfaces for the components and their interplay with the platform are described in the deliverables of the components itself. In the initial architecture specification, we agreed on a number of shared interfaces:

1. API for Ontology Management
2. API for Mapping Management
3. API for Data Source & Relational Metadata Management
4. API for RDF Data Management

The following subsections describe the purpose and high-level functionalities of these four interfaces. The platform ships with a number of help documents (wiki-pages) with detailed technical descriptions (i.e. the specific signatures) and examples of usage.

2.3.1 Ontology Management API

The *Ontology Management API* exposes the functionality for loading, storing, manipulating and reasoning over OWL ontologies. Ontologies are stored in the Optique repository natively as RDF triples, each ontology in its own context (named graph). Through the API, ontologies can be accessed through the object model of the OWLAPI[2]. In this way, much of the basic ontology management functionality can be directly reused from the OWLAPI and seamlessly interacts with Reasoners such as Hermit or Pellet.[3] Ontologies in the Optique repository are uniquely identified via their Ontology URI.[4]
At this point ontology identifiers are supposed to be unique. In the future, we may want to extend the API to support multiple versions of an ontology with the same ontology IRI/URI, but different version identifiers. The API provides the following core functionality:

- Listing of existing ontologies from the Optique repository
- Loading of an ontology and resolving possible imports from the Optique repository
- Storage of an ontology in the Optique repository
- Removal of ontologies from the repository

2.3.2 Data Source & Relational Metadata Management API

The *Data Source API* provides a layer of abstraction to connect the application layer and respective components to disparate data sources. It hides the complexity of connection to various access protocols and data formats and provides convenience methods such as for authentication and secure storage of credentials. In

[1] `http://www.fluidops.com/information-workbench/`
[2] `http://owlapi.sourceforge.net`
[3] see `http://clarkparsia.com/pellet` and `http://hermit-reasoner.com`
[4] `http://www.w3.org/TR/owl2-syntax/#Ontology_IRI_and_Version_IRI`

particular, the *Data Source API* interacts closely with the *Relational Metadata API* to provide a standardized way for accessing relational metadata such as table and column information, datatypes, constraints, and indices. Internally, the RelationalSchemaOntology (RSO) is used to abstract over the relational metadata from various database vendors. While the *Metadata API* serves primarily as common layer for abstraction, it acts also as a caching layer which allows fast access to the metadata. The process of gathering metadata is completely transparent to the administrator. Once a new data source has been registered within the platform's *Data Source API* a in-memory representation of relational schema objects can be retrieved. From schema objects, schema's name and a list of tables can be accessed. Tables objects, in turn, hold all column information and respective constrains. A column has a datatype associated and a position at which it occurs in the table.

2.3.3 R2RML Mapping Management API

The R2RML *Mapping Management API* has been designed for managing and manipulating collections of mappings according to the R2RML standard.[5] At its core, the R2RML API provides in-memory classes for creation and manipulation of mappings rules, whereas the platform API provides specifically methods for

- De-serialization of mappings from an input stream
- Serialization of mapping collection to file
- Maintenance (add, delete, replace) of mapping rules loaded into the central metadata repository
- Access to R2RML specific factory methods for mapping rule construction

The core R2RML library has been released as a standalone and open-source library [4] and is publicly available under the Apache License. [6] Please refer to the project website for extensive documentation, tests and examples of usage.

2.3.4 RDF Data Management API

A variety of assets in Optique (ontology, mapping, relational database metadata, ...) are stored in the central store as RDF. Using the respective platform APIs it is possible to access and manipulate these assets as in-memory JAVA objects. In addition, the RDF data management API allows components to store, query, and manipulate other kinds of RDF data in the central store directly:

- Load RDF data from Input Streams
- Execute SPARQL 1.0/1.1 SELECT, CONSTRUCT, ASK, INSERT/DELETE queries

The API is used, for example, by the backend of the visual formulation interface to store and retrieve the queries in the query catalogue using the SPIN modelling vocabulary[7].

2.4 Integration of Components

2.4.1 Ontology & Mapping Management

OBDA systems crucially depend on the existence of suitable ontologies and mappings. Developing them from scratch is likely to be expensive and a practical OBDA system should support a (semi-) automatic bootstrapping of an initial ontology and set of mappings.

The *Ontology and Mapping (O&M) management component* is in charge of creating and evolving the ontology and mappings, as well as, providing an interface to feed the *Query Formulation component* with the ontology vocabulary in order to guide the formulation of the queries.

The current implementation of the Optique O&M component is equipped with an O&M *bootstrapper*, a routine that takes a database schemata and possibly instances over these schemata as an input, and returns an ontology and a set of mappings connecting the ontology entities to the elements of the input schemata.

[5] `http://www.w3.org/TR/r2rml/`
[6] `https://github.com/R2RML-api/R2RML-api`
[7] `http://spinrdf.org/spin.html`

For this purpose the O&M component retrieves the required metadata from the platforms' shared metadata repository using the Relational Metadata API (cf. Section 2.3.2).

The Optique O&M component also integrates an ontology matching system to align the bootstrapped ontology with state of the art domain ontologies and an ontology approximation module to transform the resulting ontology if it is outside the desired OWL 2 profile.

The O&M bootstrapper uses the Ontology API (cf. Section 2.3.1) and the Mapping Management API (cf. Section 2.3.2) to store the resulting assets back to the platform.

After a set of mappings has been generated by the *bootstrapper*, the set of mappings need to be manually edited and refined. The *bootstrapper* is seamlessly integrated and allows the user to navigate directly to the built-in mapping editor for manual refinements of the mappings. The manual edition of mappings is a complex process, which means that errors can be introduced and they may be hard to detect. A *mapping analysis* routine is provided to assist in the detection of errors and other anomalous situations in sets of mappings. To perform this analysis, the *mapping analysis* routine takes a set of database schemata, an ontology and a set of mappings and returns a list of messages that can be serialized to HTML and displayed to the user. All three inputs (the database schemata, the ontology and the mappings) are specified according to the platforms' APIs for these resources (cf. Sections 2.3.2, 2.3.3 and 2.3.1).

2.4.2 Query Transformation

The *Query Transformation (QT) component* Ontop allows to query virtual RDF graphs defined by a relational database, an ontology, and a set of R2RML mappings. The general architecture is illustrated in Figure 2.2. In a first step a set of mappings, the ontology and the SPARQL query are translated into a set of Datalog rules that represent these objects. Second, the program is optimized using query containment based techniques and Semantic Query Optimization. In particular, *SLD-resolution* is used to compute a *partial evaluation* of the program and the queries are optimized with respect to Primary/Foreign Keys to avoid redundant self-joins. The optimized program is translated into an equivalent relational algebra expression, the SQL query is generated and executed by the DBMS. The relational metadata, together with

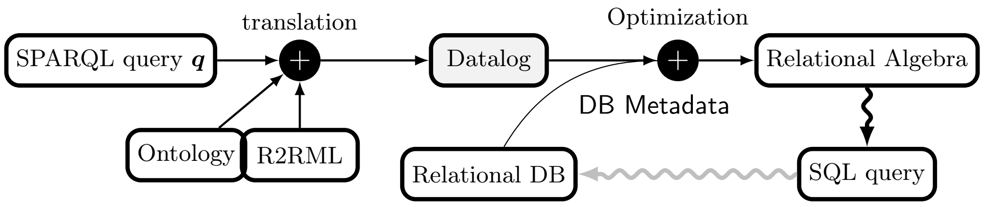


Figure 2.2: Query answering with mappings in Ontop

the R2RML mappings and the OWL ontology are fetched from the platform's shared metadata repository using the respective APIs (cf. Sections 2.3.2, 2.3.3 and 2.3.1) and passed to the Ontop repository during initialization. As Ontop internally makes use of specialized data structures for handling metadata Optique's database metadata object is transformed into a Ontop specific metadata object. Similarly, Ontop supports R2RML mappings by transforming them internally into its own type of mappings.

The QT component is registered within the platform as a Sesame[8] repository. Consequently, native platform functionalities and widgets such as for search and visualization are able to seamless operate on this repository. The platform provides a specialized widget to ease the configuration and initialization of the QT component (cf. Section 3.2.1). As such the administrator can simply select the respective configuration parameters from the pre-populated drop-down in order to change the configuration. Furthermore, advanced configuration parameters such as user-defined database constraints or database settings can be set using the respective configuration properties. During initialization of the QT component basic consistency checks will

[8]http://www.openrdf.org/documentation.jsp

be performed taken the metadata, mappings and ontology into account. Possible errors are being propagated to the administrator using human readable exception message.

The initial integration of the module for managing time and streams has been realised as a extension of the standard query transformation component and fetches respectively meta-data, mappings and the ontology from the platforms' shared APIs. Instead of using JDBC, it operates on a newly implemented ADP "StreamingDataSource".

2.4.3 Distributed Query Execution Engine

ADP, a system for distributed data processing in the cloud, is the component responsible for the query execution in Optique. ADP provides a driver that implements the standard JDBC interface and therefore it can be used as a normal JDBC data source, as described in section 2.3.2. ADP has also the ability of federated query execution. In this mode of execution, one can register other relational databases as endpoints in ADP and then the system will provide a unified view of all the data in the different databases. A special data source, named *ADP Federated JDBC*, has been created in the Optique platform in order to facilitate the usage of ADP in the federated mode. During the instantiation of a data source of this type, the user can set other relational data sources that have already been defined in the platform (cf. Section 2.3.2), to act as endpoints for the newly created data source.

2.4.4 Query Formulation

The Optique *Visual Query Formulation (VQS) system* is designed as a user-interface (UI) mashup built on widgets. A UI mashup aggregates different applications into a common graphical space and orchestrates them for common goals. Widgets[9] are the building blocks of our VQS and refer to portable, self-contained, full-fledged, and mostly client side applications with limited functionality and complexity. Widgets in our system communicate with each other by delivering events, generated by user actions, through a client-side communication channel. Each widget reacts to events either in a preprogrammed way or by considering the semantic and syntactic signatures of events.

The VQS consists of a client-side component that is the presentation and a server-side component that uses platform APIs to serve the ontology and data to the client-side component. The client-side component is purely HTML and JavaScript and uses existing libraries for realising its functionality. JQuery mobile[10] is used to generate widgets, InfoVis[11] is used to visualise query graphs, and JQuery[12] is used for cross-browser compliance. The communication channel is built on HTML 5's[13] message passing support.

The server-side component makes internally use of the *RDF Data Management* and *Ontology API* (cf. Sections 2.3.4 and 2.3.1). The component is implemented as a platform service and exposed through Optique platform's API framework. As such the communication between the client-side and the server-side components is handled via a set of REST calls. Currently the OptiqueVQS uses the following REST methods from query formulation backend-service:

- `getAvailableOntologies()`: gets the list of identifiers of the available ontologies in the triple store.
- `loadOntology(String ontologyURI)`: it loads an ontology given its URI.
- `getCoreConcepts()`: gets the core concepts of the active ontology
- `getConceptFacets(String conceptURI)`: retrieves associated facets given a concept URI
- `getNeighbourConcepts(String conceptURI)`: retrieves associated concept neighbours for given URI.

Each REST call returns the ontology-related information serialised as JSON objects, which will populate the VQS interface.

[9]http://www.w3.org/TR/widgets/
[10]http://jquerymobile.com/
[11]http://philogb.github.io/jit/
[12]http://jquery.com/
[13]http://www.w3.org/TR/html5/

Chapter 3

Documentation and Prototype

This chapter provides detailed instructions for system administrators (IT-Experts) in order to deploy and configure the Optique platform. Beside guidance for the technical system installation of the platform on different operation systems, it explains the basic steps that are required to initialize the platform. Particularly, it shows how to connect the system to a relational datasource (also called "data endpoints"), bootstrapping the system with initial configurations which are required for the initialization of the component for query transformation. Finally, basic functionality of the novel query formulation interface is explained to both, the IT-Expert as well as the End-User.

3.1 Installation Instructions

3.1.1 Obtaining the platform bundle

The Optique platform prototype is available in the restricted download area of the project website. Please contact the project coordinator if you would like to obtain a copy for review or evaluation purpose.

3.1.2 Installation Requirements

Server - Operating System

Windows (64-bit only): Windows 7, Windows Server 2008
Linux (64-bit only): openSUSE 12.1
Java Runtime Environment (JRE >= 1.7.0_25 64 bit)
32-bit systems, other Linux Distributions, different versions of Windows or OS X systems may also work, but are not officially supported..

Client -Browsers

Firefox >=17.x (ESR)
Internet Explorer >=8
Safari >=5.1.7
Other browsers may also work, but are not officially supported.

3.1.3 Installation

The Optique platform supports both Windows and Linux based operating systems.

Windows

Installation from the zip-distribution

It is recommended to use a 64bit Windows operating system with a 64Bit Java SE Runtime Environment in version 1.7 (taken from the JDK). The reference version shipped with the installer is JRE SE 1.7.0_25 64bit. This is also the version used in steps a) and b) below. Unpack the distribution into a directory of any choice (e.g. `C:\\OPTIQUE`). In the following, we will refer to the absolute pathname of this directory by `<OPTIQUE_HOME_DIRECTORY>`.

Running the Optique platform as executable

Execute `<OPTIQUE_HOME_DIRECTORY>/start.cmd`

Running the Optique platform on a 32bit Windows operating system

The Optique platform can be run on a 32bit Windows operating system by following the steps below:

1) Download and install Java SE 32-Bit JDK version 1.7.[1]

2) Set the path of java.exe in the file `<OPTIQUE_HOME_DIRECTORY>/fiwb/backend.conf`, examples:
`wrapper.java.command=C:\Program Files\Java\jdk1.7.0_25\bin\java` (absolute path)
`wrapper.java.command=java` (if the java command is in the Path environment)

3) Execute the Optique platform as described above.

Linux

To run the Optique platform under Linux a Java SE Runtime Environment version 1.7 (taken from the JDK) must be installed. Note that the Optique platform does not ship a reference version bundled with the release. Unpack the distribution into a directory of any choice (e.g. /opt/optique). In the following, we will refer to the absolute pathname of this directory by `<OPTIQUE_HOME_DIRECTORY>`. Download and install Java SE 64-Bit JDK version 1.7. Make sure the java command is added to the command-path of the user root.

a) Running as service

Create a user under which the optique platform shall run, e.g. “fluid” (in the following we will refer to this user as `<OPTIQUE_USER>`).
If “fluid” has not been chosen as user, the script `<OPTIQUE_HOME_DIRECTORY>/fiwb/iwb.sh` has to be adapted accordingly: Search for `RUN_AS_USER=fluid` and replace fluid by `<OPTIQUE_USER>`
Exceute the script linux-install.sh in `<OPTIQUE_HOME_DIRECTORY>` as user root, like follows:

```
bash -eu linux-install.sh <OPTIQUE_USER>
```

This installs an init-script as `/etc/init.d/iwb` and starts the application. To make sure this script is executed on reboot create corresponding links in the run-level specific directories.
Depending on the unix distribution, this can be done with: `chkconfig -a iwb` or with `insserv iwb`

b) Running the Optique platform as executable

Make sure all script are executable by executing in `<OPTIQUE_HOME_DIRECTORY>`:

```
chmod +x *.sh fiwb/*.sh fiwb/wrapper-linux*
```

If the Optique platform needs to be executed as a user different from “fluid”, the script `<OPTIQUE_HOME_DIRECTORY>/fiwb/iwb.sh` has to be adapted accordingly: Search for `RUN_AS_USER=fluid` and replace “fluid” by any preferred user (this user must exist on the system).
Execute start.sh in `<OPTIQUE_HOME_DIRECTORY>`.

Mac OS X

Please note that, while we have successfully installed and run the Optique platform on Mac OS X, this platform is not officially supported. To run the Optique platform on Mac OS X it requires a compatible version of the Java runtime (ideally, version 1.7). OS X may ask whether to install a Java runtime automatically if it detects that one is needed but missing the Optique platform distribution (.zip file). To get started, proceed as follows:

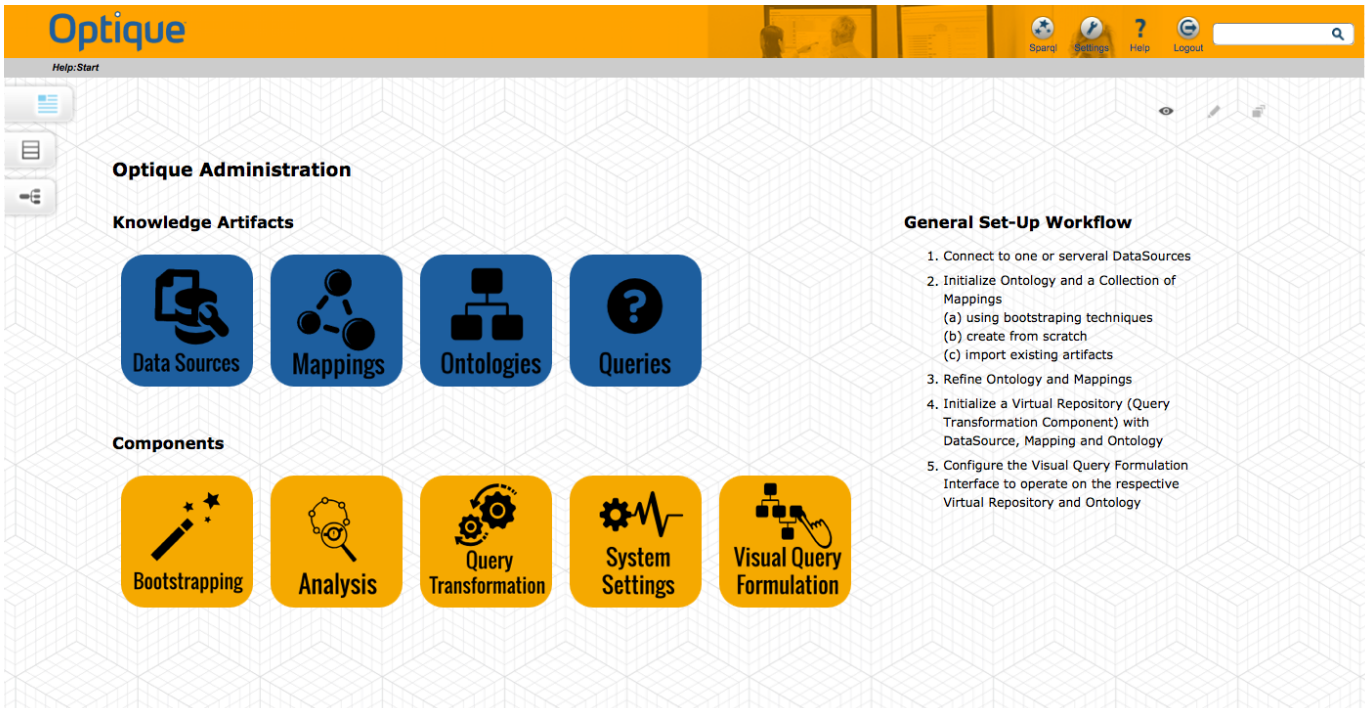
- Unpack the Optique platform zip distribution.

[1] `http://www.oracle.com/technetwork/java/javase/downloads/index.html`

- Open the terminal application and cd into the unpacked distribution (e.g. `cd Desktop/IWB' [ENTER]`)
- Make scripts executable by typing `chmod +x *.sh fiwb/*.sh fiwb/wrapper* [ENTER]`)
- In the file, `fiwb/iwb.sh`, modify the value for `RUN_AS_USER=fluid` to the user name that is intended to run the Optique platform. (Alternatively, create a user account with the name `fluid`.)
- To start the Optique platform, now type `./start.sh' [ENTER]`
- After a few seconds, the Optique platform should be accessible locally from any browser under the address `http://localhost:8888`

3.1.4 Opening the Optique platform

Once the Optique platform is running (initial startup may take a few minutes), it can be accessed at `http://localhost:8888` . The start page (cf. Figure 3.1) provides links to mange all nessecariy knowledge artefacts as well as to configure the components.



General Set-Up Workflow

1. Connect to one or serveral DataSources
2. Initialize Ontology and a Collection of Mappings
   (a) using bootstraping techniques
   (b) create from scratch
   (c) import existing artifacts
3. Refine Ontology and Mappings
4. Initialize a Virtual Repository (Query Transformation Component) with DataSource, Mapping and Ontology
5. Configure the Visual Query Formulation Interface to operate on the respective Virtual Repository and Ontology

Figure 3.1: Start page of the Optique Platform, describing the basic steps for initial configuration.

By default, an administrator account with credentials "admin/iwb" is created. It is highly recommended to change the password in the Admin area after the first login.

3.1.5 Shutting down the Optique platform

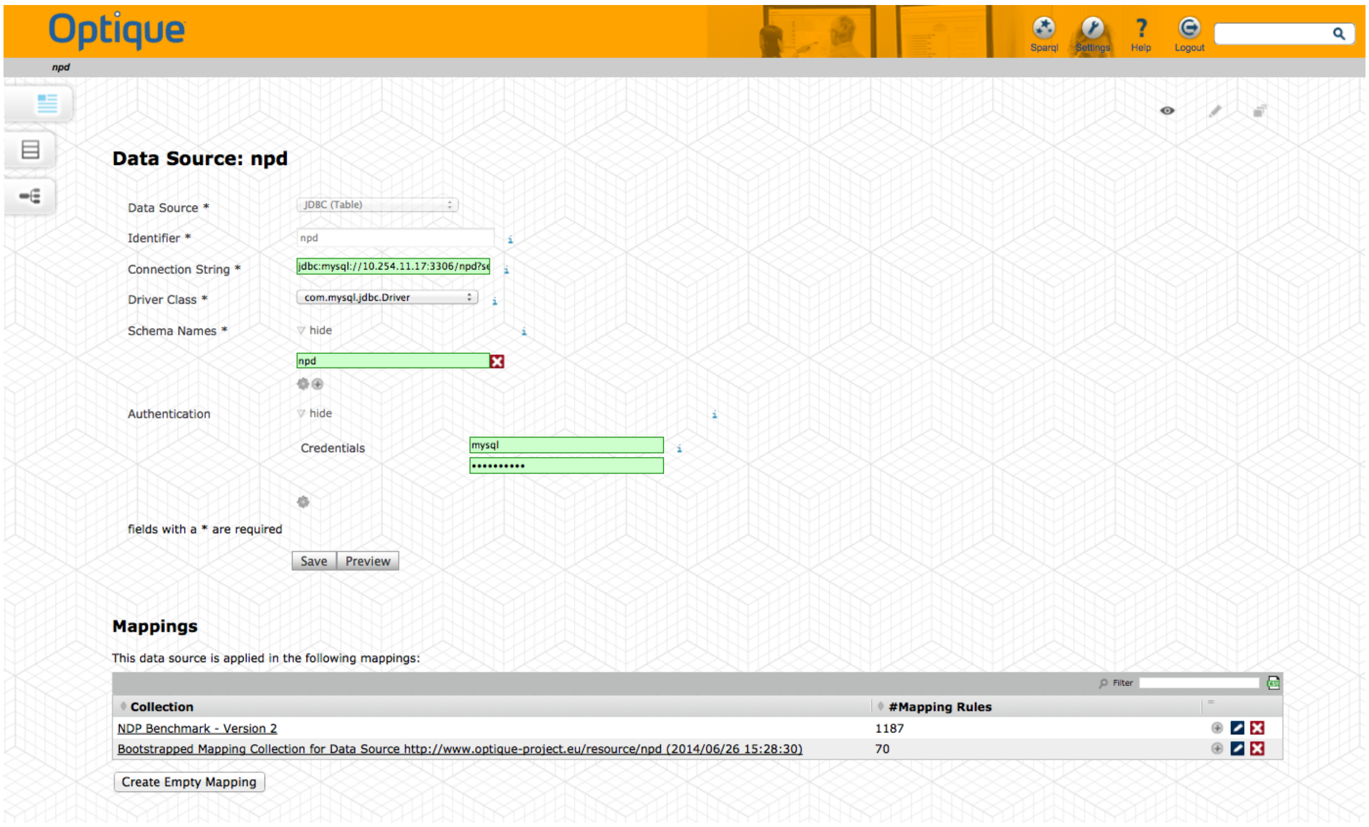
Windows Service: Go to the Services configuration tool and stop the service
Linux Daemon: Invoke `/etc/init.d/iwb stop`
Command line (all OS): Exit the Optique platform by clicking in the command line window and pressing `Ctrl + c`.
IMPORTANT: Never exit the Optique platform by closing the command line window without proper shutdown. This can result in corrupting the data store and loss of data.

3.2 Administration Guide

The initial set-up of the platform for a specific domain, requires to go through several pref-configuration steps (Section 3.2.1) before the system can be fully customized (Section 3.2.2). We assume that this is to be done by an IT-Expert rather than the End-User (cf. Figure 1.1).

3.2.1 Setup and Configuration Steps

Configuration of Datasource



Data Source: npd

Data Source *	JDBC (Table)
Identifier *	npd
Connection String *	jdbc:mysql://10.254.11.17:3306/npd?se
Driver Class *	com.mysql.jdbc.Driver
Schema Names *	hide
	npd
Authentication	hide
Credentials	mysql
	••••••••••

fields with a * are required

Save Preview

Mappings

This data source is applied in the following mappings:

Filter

Collection	#Mapping Rules
NDP Benchmark - Version 2	1187
Bootstrapped Mapping Collection for Data Source http://www.optique-project.eu/resource/npd (2014/06/26 15:28:30)	70

Create Empty Mapping

Figure 3.2: Creating a new JDBC data sources.

Before the platform can be connected to a data source, there are two things to ensure beforehand:

1. JDBC access to a (remote) database (i.e. typical constrains are user rights, network settings etc.)
2. the platform has the respective JDBC driver installed (the page Admin:JDBCDriver lists registered drivers and provides further instructions to add missing ones, for example, due to incompatible license)

When navigating from the start-page (cf. Figure 3.1) to the data source page, a dashboard will list all registered data sources. The link "Create new Data Source" leads to an empty data source form. First the type of data source needs to be selected. For the time being, the Optique platform supports to configure the following data sources from the UI:

1. simple tabular files to be processed in memory (Excel, TSV, CSV Data Source)
2. relational databases engines such using JDBC connections (JDBC DataSource)

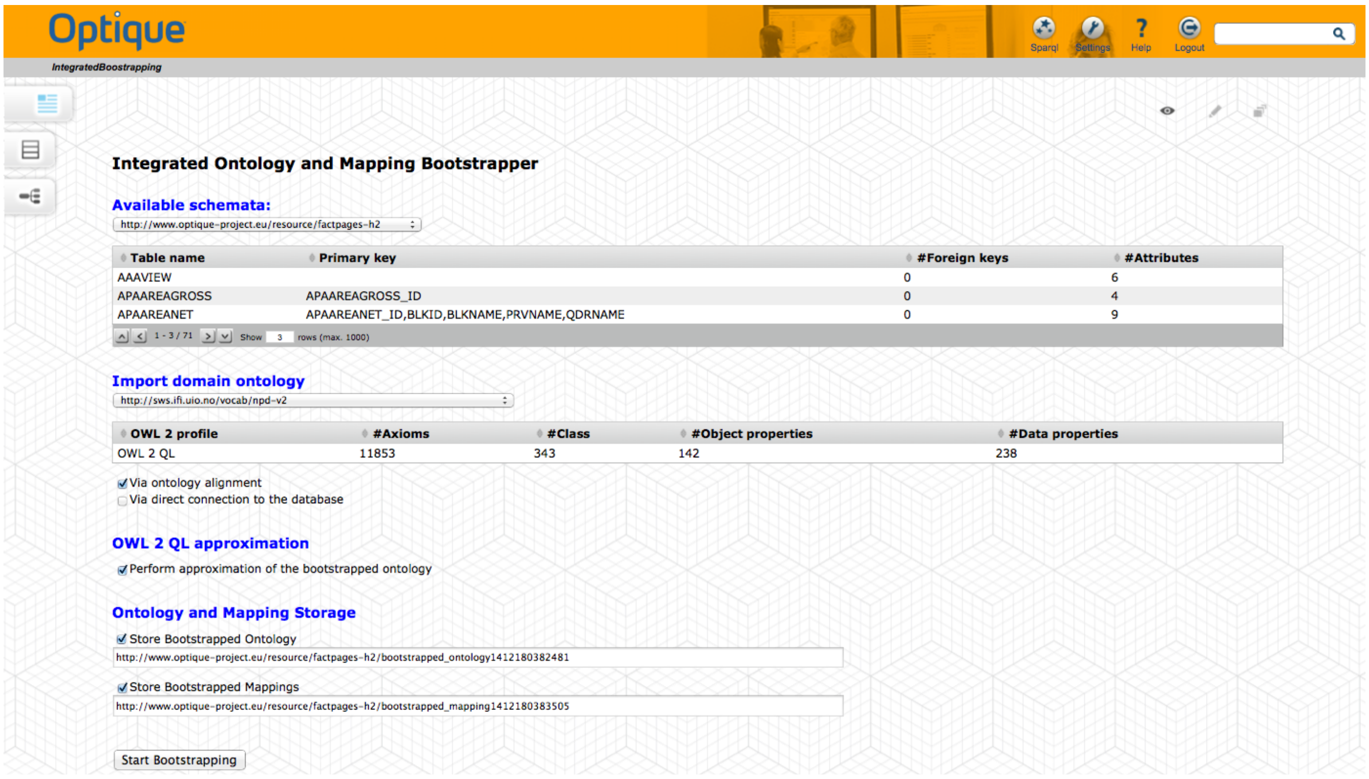
A special implementation of a JDBC data source is the ADP Data Source in order to connected via JDBC to the distributed query execution engine (ADP) in a distributed and/or federated setting. However, this requires access to an a virtualization environment and needs to be set-up beforehand. The configuration of streaming data sources is not yet fully supported from the UI.

Bootstrapping of Initial System Configurations: Ontologies and Mappings

Once the platform has been connected successfully to a relational data source, the integrated bootstrapping widget can be utilized in order to:

- bootstrap an initial ontology directly from the relation metadata
- bootstrap a set of direct mappings
- automatically align the bootstrapped ontology to an imported domain ontology
- if needed, approximate the ontology language profile to the OWL 2 QL profile

The bootstrapper stores the generated knowledge artifacts in the platform's shared metadata repository.



Integrated Ontology and Mapping Bootstrapper

Available schemata:

http://www.optique-project.eu/resource/factpages-h2

Table name	Primary key	#Foreign keys	#Attributes
AAAVIEW		0	6
APAAREAGROSS	APAAREAGROSS_ID	0	4
APAAREANET	APAAREANET_ID,BLKID,BLKNAME,PRVNAME,QDRNAME	0	9

1 - 3 / 71 Show 3 rows (max. 1000)

Import domain ontology

http://sws.ifi.uio.no/vocab/npd-v2

OWL 2 profile	#Axioms	#Class	#Object properties	#Data properties
OWL 2 QL	11853	343	142	238

✓ Via ontology alignment
☐ Via direct connection to the database

OWL 2 QL approximation

✓ Perform approximation of the bootstrapped ontology

Ontology and Mapping Storage

✓ Store Bootstrapped Ontology
http://www.optique-project.eu/resource/factpages-h2/bootstrapped_ontology1412180382481

✓ Store Bootstrapped Mappings
http://www.optique-project.eu/resource/factpages-h2/bootstrapped_mapping1412180383505

Start Bootstrapping

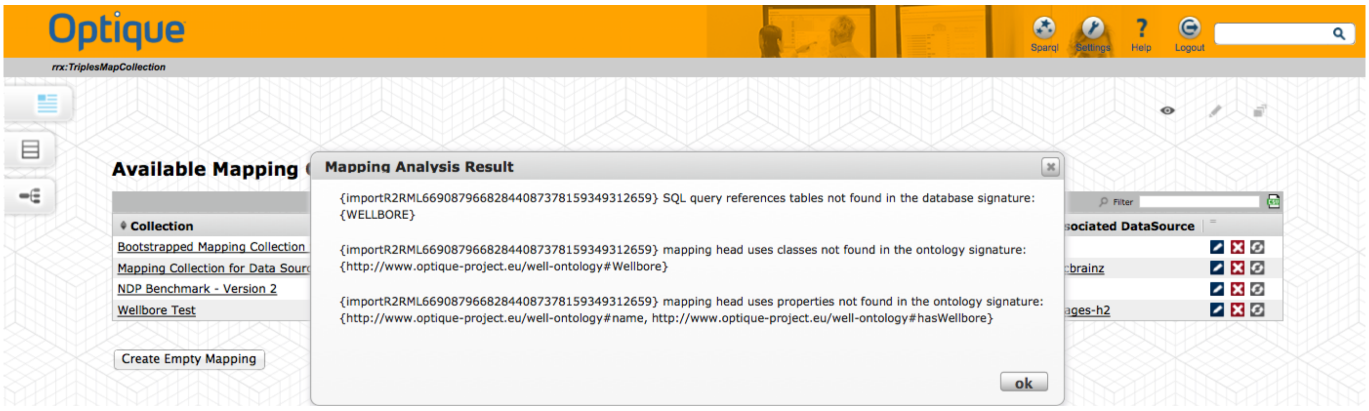
Figure 3.3: Widget for integrated bootstrapping.

Ontology and Mapping Refinement

After a set of mappings has been generated by the *bootstrapper*, the mapping rules and the ontology need to be refined manually. When navigating from the start-page (cf. Figure 3.1) to the mapping page, the platform lists all availabe mapping collections and provides functionality for creation, deletion, renaming, import/export and analysis of collections (cf. Figure 3.4). Each mapping collection consists of one or several mapping rules. New mapping rules can be created by either referencing database tables directly (cf. Figure 3.5) or by defining custom SQL queries as input. Once a mapping rule is created, it can be edited in-line by browsing to the respective instance page (cf. Figure 3.6).[2]

The manual edition of R2RML mappings is a complex process, which means that errors can be introduced and they may be hard to detect. A *mapping analysis* routine is provided to assist in the detection of errors. The mapping analysis routine can be called from the mapping collection overview page (cf. Figure 3.4).

[2]Please refer to the R2RML specfication for further details about the structure of R2RML mapping rules: `http://www.w3.org/TR/r2rml/`



Available Mapping

Collection	ociated DataSource
Bootstrapped Mapping Collection	
Mapping Collection for Data Sourc	:brainz
NDP Benchmark - Version 2	
Wellbore Test	ages-h2

Create Empty Mapping

Mapping Analysis Result

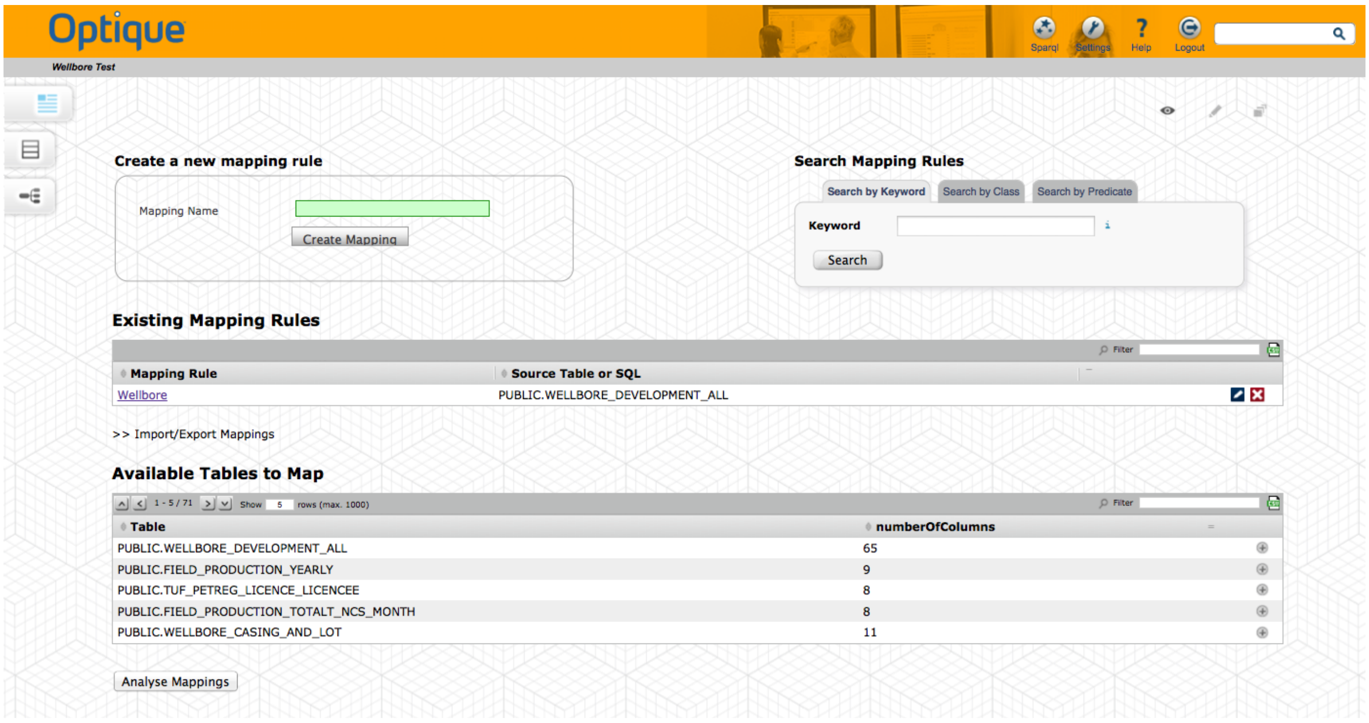
```
{importR2RML66908796682844087378159349312659} SQL query references tables not found in the database signature:
{WELLBORE}

{importR2RML66908796682844087378159349312659} mapping head uses classes not found in the ontology signature:
{http://www.optique-project.eu/well-ontology#Wellbore}

{importR2RML66908796682844087378159349312659} mapping head uses properties not found in the ontology signature:
{http://www.optique-project.eu/well-ontology#name, http://www.optique-project.eu/well-ontology#hasWellbore}
```

ok

Figure 3.4: Overview of available mapping collections in the platform. Mapping collections can be renamed, deleted and analysed. The figure shows the result of such a syntactic mapping analysis.



Create a new mapping rule

Mapping Name

Create Mapping

Search Mapping Rules

Search by Keyword | Search by Class | Search by Predicate

Keyword

Search

Existing Mapping Rules

Mapping Rule	Source Table or SQL
Wellbore	PUBLIC.WELLBORE_DEVELOPMENT_ALL

>> Import/Export Mappings

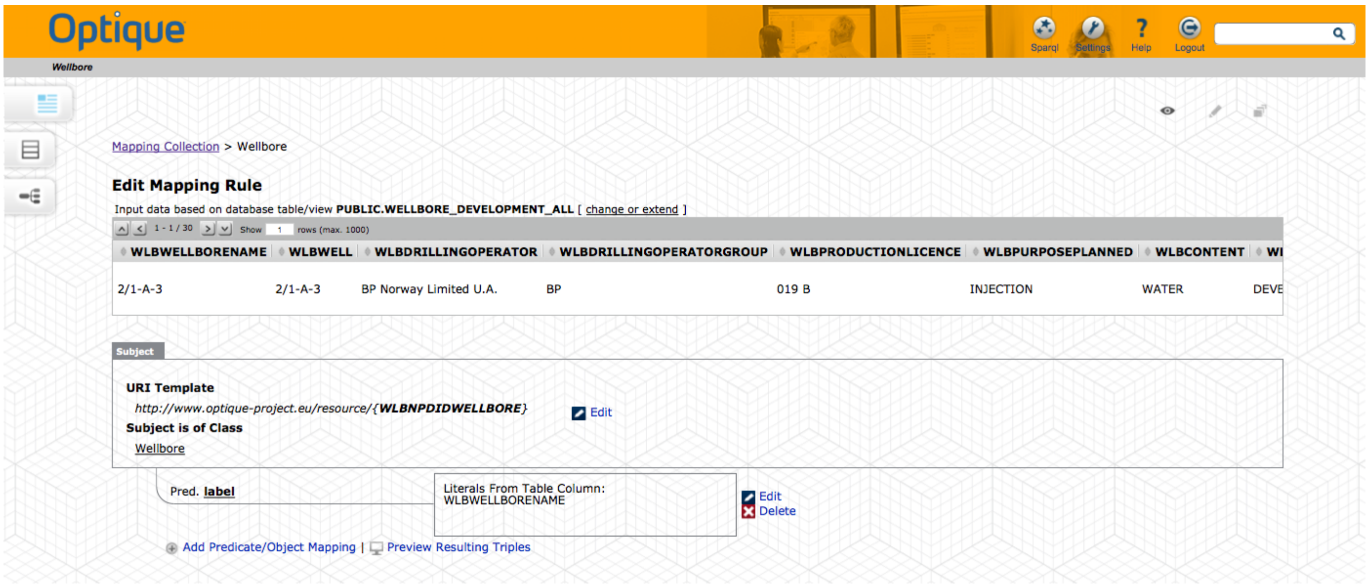
Available Tables to Map

1 - 5 / 71 Show 5 rows (max. 1000)

Table	numberOfColumns
PUBLIC.WELLBORE_DEVELOPMENT_ALL	65
PUBLIC.FIELD_PRODUCTION_YEARLY	9
PUBLIC.TUF_PETREG_LICENCE_LICENCEE	8
PUBLIC.FIELD_PRODUCTION_TOTALT_NCS_MONTH	8
PUBLIC.WELLBORE_CASING_AND_LOT	11

Analyse Mappings

Figure 3.5: Overview of existing mapping rules in a mapping collection. Mappings can be searched and ordered by different criteria. New mapping rules can be created either from scratch (top) or by referencing database tables or views directly (bottom).



Mapping Collection > Wellbore

Edit Mapping Rule

Input data based on database table/view **PUBLIC.WELLBORE_DEVELOPMENT_ALL** [change or extend]

1 - 1 / 30 Show 1 rows (max. 1000)

WLBWELLBORENAME	WLBWELL	WLBDRILLINGOPERATOR	WLBDRILLINGOPERATORGROUP	WLBPRODUCTIONLICENCE	WLBPURPOSEPLANNED	WLBCONTENT	WI
2/1-A-3	2/1-A-3	BP Norway Limited U.A.	BP	019 B	INJECTION	WATER	DEVE

Subject

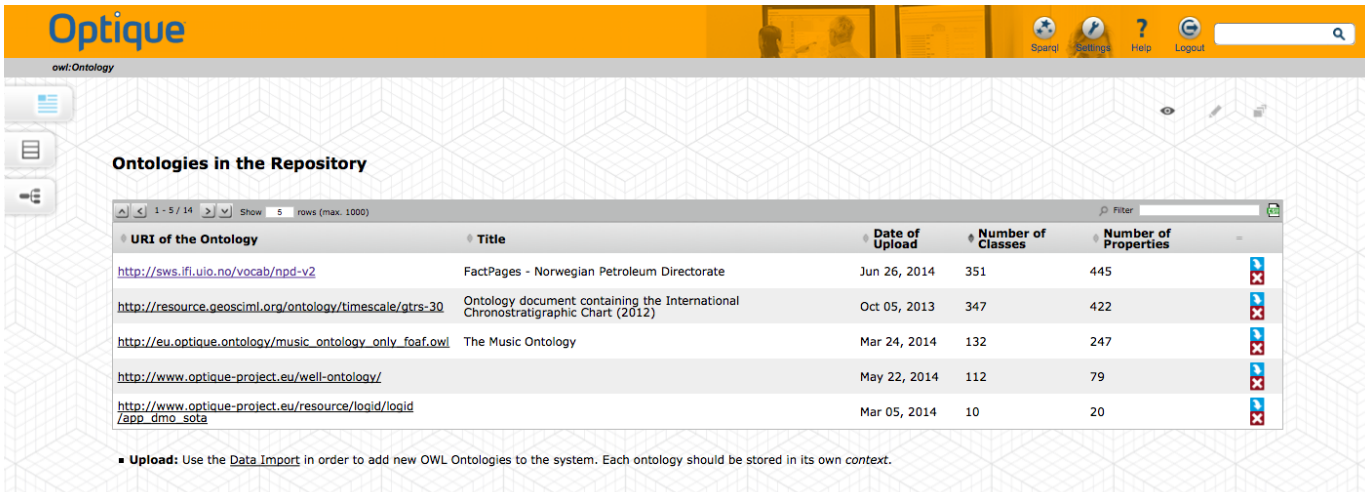
URI Template
*http://www.optique-project.eu/resource/{**WLBNPDIDWELLBORE**}* Edit
Subject is of Class
Wellbore

Pred. **label** | Literals From Table Column: WLBWELLBORENAME | Edit Delete

Add Predicate/Object Mapping | Preview Resulting Triples

Figure 3.6: Editing a single mapping rule in the mapping editor. The table in the top displays the raw data as defined in the logical table. The input can be changed and can either be a custom SQL query or direct SQL Table/View reference. One mapping rule consists of exactly one subject definition and 1:m object-predicate definitions. The subject as well as existing object-predicate definitions can be edit. New object-predicate pairs can be added in the bottem.

For the time being the platform has limited support for in-line ontology editing and visualization. However, ontologies from the platform can easily be exported to matured, third party tools such as Protege.[3] When navigating from the start-page (cf. Figure 3.1) to the ontology page, the platform will list all known ontologies from the metadata repository (cf. Figure 3.7) and provides functionality for deletion, export, import of ontologies .



Ontologies in the Repository

1 - 5 / 14 Show 5 rows (max. 1000) Filter

URI of the Ontology	Title	Date of Upload	Number of Classes	Number of Properties
http://sws.ifi.uio.no/vocab/npd-v2	FactPages - Norwegian Petroleum Directorate	Jun 26, 2014	351	445
http://resource.geosciml.org/ontology/timescale/gts-30	Ontology document containing the International Chronostratigraphic Chart (2012)	Oct 05, 2013	347	422
http://eu.optique.ontology/music_ontology_only_foaf.owl	The Music Ontology	Mar 24, 2014	132	247
http://www.optique-project.eu/well-ontology/		May 22, 2014	112	79
http://www.optique-project.eu/resource/logid/logid/app_dmo_sota		Mar 05, 2014	10	20

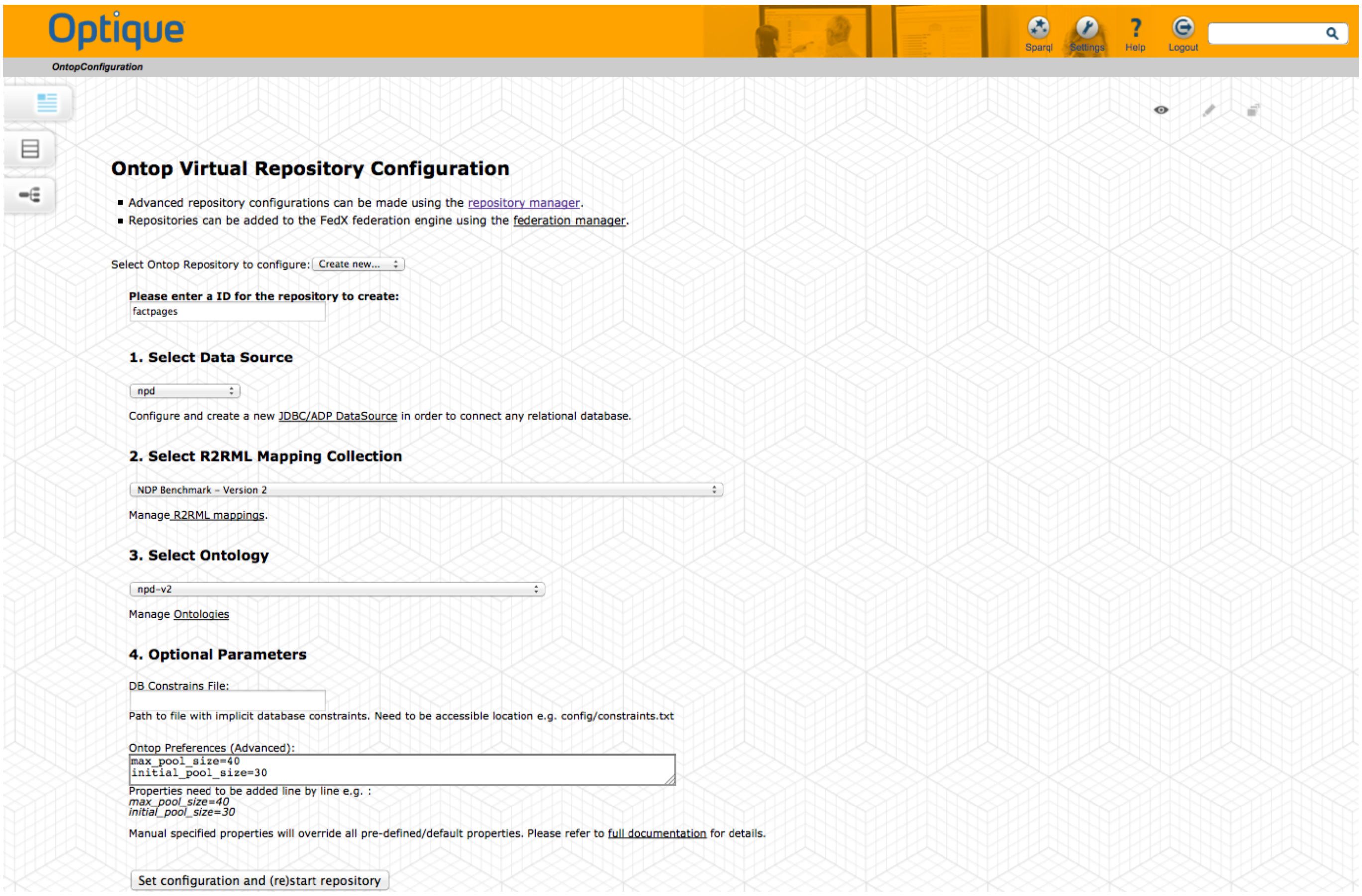
▪ **Upload:** Use the Data Import in order to add new OWL Ontologies to the system. Each ontology should be stored in its own *context*.

Figure 3.7: List of available ontologies in the platform. Ontologies can be deleted, exported and imported.

Configuration and Initialization of the Query Transformation Component

After refinement of the ontology and the respective mappings a new virtual repository (query transformation component) needs to be initialized. For this purpose, the platform provides a specific widget (cf. Figure 3.8)

[3]`http://protege.stanford.edu/`



Ontop Virtual Repository Configuration

- Advanced repository configurations can be made using the repository manager.
- Repositories can be added to the FedX federation engine using the federation manager.

Select Ontop Repository to configure: Create new...

Please enter a ID for the repository to create: factpages

1. Select Data Source

npd

Configure and create a new JDBC/ADP DataSource in order to connect any relational database.

2. Select R2RML Mapping Collection

NDP Benchmark - Version 2

Manage R2RML mappings.

3. Select Ontology

npd-v2

Manage Ontologies

4. Optional Parameters

DB Constrains File:

Path to file with implicit database constraints. Need to be accessible location e.g. config/constraints.txt

Ontop Preferences (Advanced):

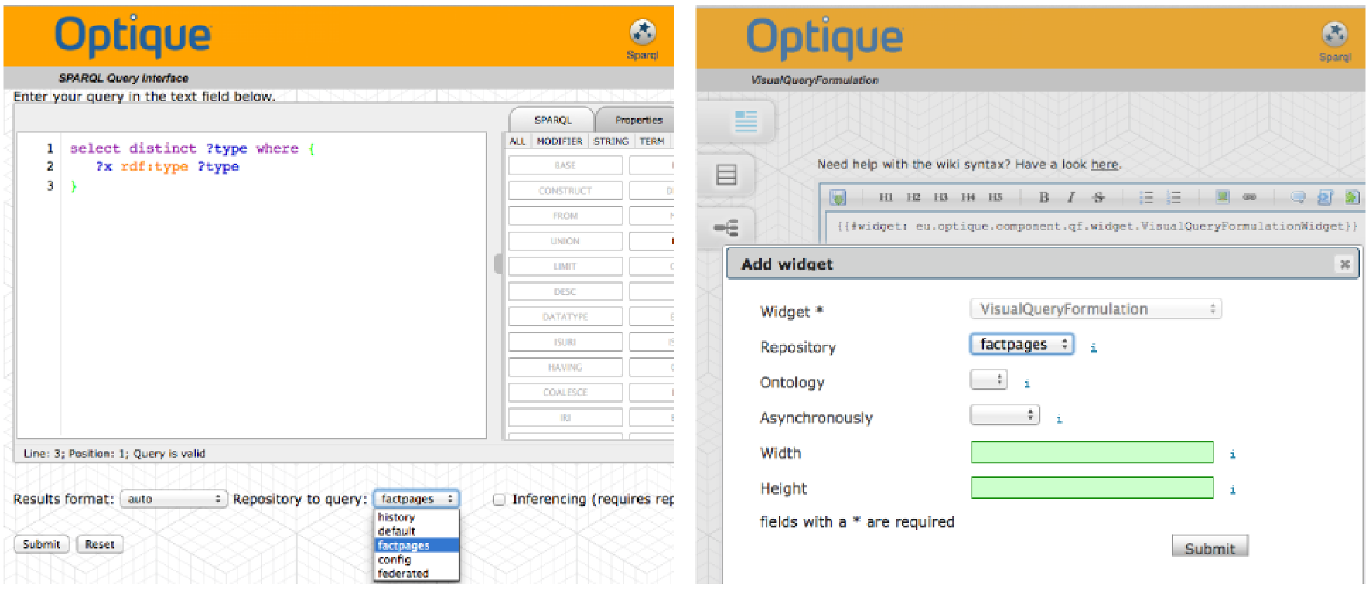
```
max_pool_size=40
initial_pool_size=30
```

Properties need to be added line by line e.g. :
max_pool_size=40
initial_pool_size=30

Manual specified properties will override all pre-defined/default properties. Please refer to full documentation for details.

Set configuration and (re)start repository

Figure 3.8: Widget to configure a virtual repository with a data source, mappings and an ontology.



Enter your query in the text field below.

```
select distinct ?type where {
    ?x rdf:type ?type
}
```

Line: 3; Position: 1; Query is valid

Results format: auto Repository to query: factpages (history, default, factpages, config, federated) Inferencing (requires rep

Submit Reset

Need help with the wiki syntax? Have a look here.

```
{{#widget: eu.optique.component.qf.widget.VisualQueryFormulationWidget}}
```

Add widget

Widget *	VisualQueryFormulation
Repository	factpages
Ontology	
Asynchronously	
Width	
Height	

fields with a * are required

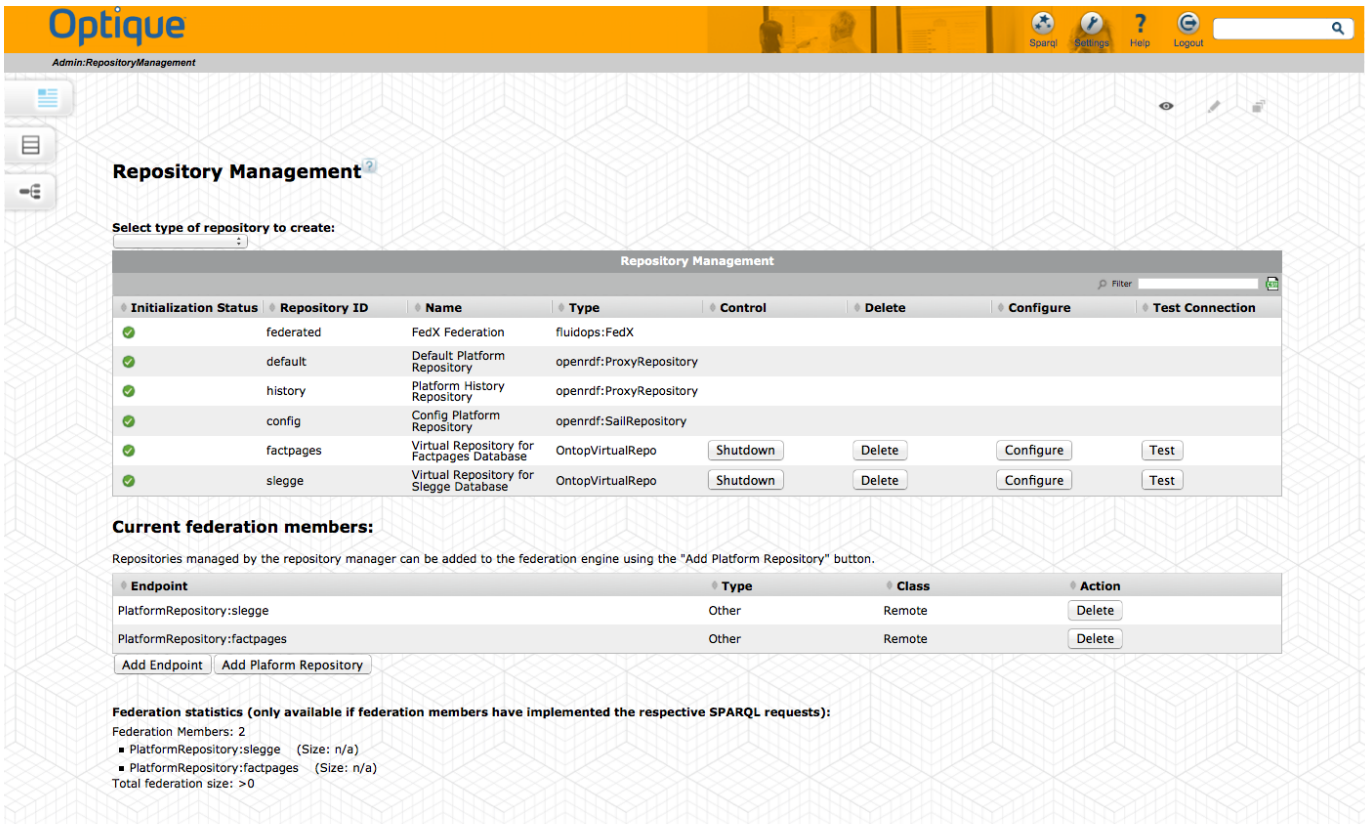
Submit

Figure 3.9: Left: Selection of repository to execute a query from the syntactic SPARQL interface against. Right: Selection of repository from the visual query formulation widget configuration. By selecting the repository in the widget's configuration, the widget will operate directly on the repository i.e. queries constructed in the visual query formulation interface will be executed against the respective repository.

to easy the configuration. Once the required configuration parameters have been set and stored, the query transformation component will be initialized and registered as a semantic repository within the platform. Depending on the size of the meta data, the mappings and the ontology, the initialization may take some time. After successful initialization, the platform offers automatically to operate on the virtualised repository, for example, from within widgets, the SPARQL interface or the visual query formulation interface (cf. Figure 3.9.

Repository Management & SPARQL Federation

The platform supports to connect to several materialized and virtualized SPARQL repositories at the same time. Therefore, the platform repository manager serves as a central registry for managing any kind of repository and connecting those to the platform. Federation over several platform repositories can be enabled by using the platform's native federation engine FedX. FedX is a framework to transparently evaluate queries against virtually integrated data sources. Different kinds of semantic repositories (native, remote SPARQL endpoints) registered in the repository manager can be added seamlessly to the federation engine during runtime. The federation engine itself is registered as a standard repository in the repository manager as well and is identified by a fixed repository ID "federated". Visualization widgets or the query formulation widget can be configured to operate on a particular repository by specifying the ID respectively (cf. Figure 3.9).



Admin:RepositoryManagement

Repository Management

Select type of repository to create:

Repository Management

Filter

Initialization Status	Repository ID	Name	Type	Control	Delete	Configure	Test Connection
✓	federated	FedX Federation	fluidops:FedX				
✓	default	Default Platform Repository	openrdf:ProxyRepository				
✓	history	Platform History Repository	openrdf:ProxyRepository				
✓	config	Config Platform Repository	openrdf:SailRepository				
✓	factpages	Virtual Repository for Factpages Database	OntopVirtualRepo	Shutdown	Delete	Configure	Test
✓	slegge	Virtual Repository for Slegge Database	OntopVirtualRepo	Shutdown	Delete	Configure	Test

Current federation members:

Repositories managed by the repository manager can be added to the federation engine using the "Add Platform Repository" button.

Endpoint	Type	Class	Action
PlatformRepository:slegge	Other	Remote	Delete
PlatformRepository:factpages	Other	Remote	Delete

Add Endpoint | Add Plaform Repository

Federation statistics (only available if federation members have implemented the respective SPARQL requests):
Federation Members: 2
- PlatformRepository:slegge (Size: n/a)
- PlatformRepository:factpages (Size: n/a)

Total federation size: >0

Figure 3.10: List of all initialized SPARQL repositories in the platform. Two (virtualized) repositories have been added to the FedX SPARQL federation engine.

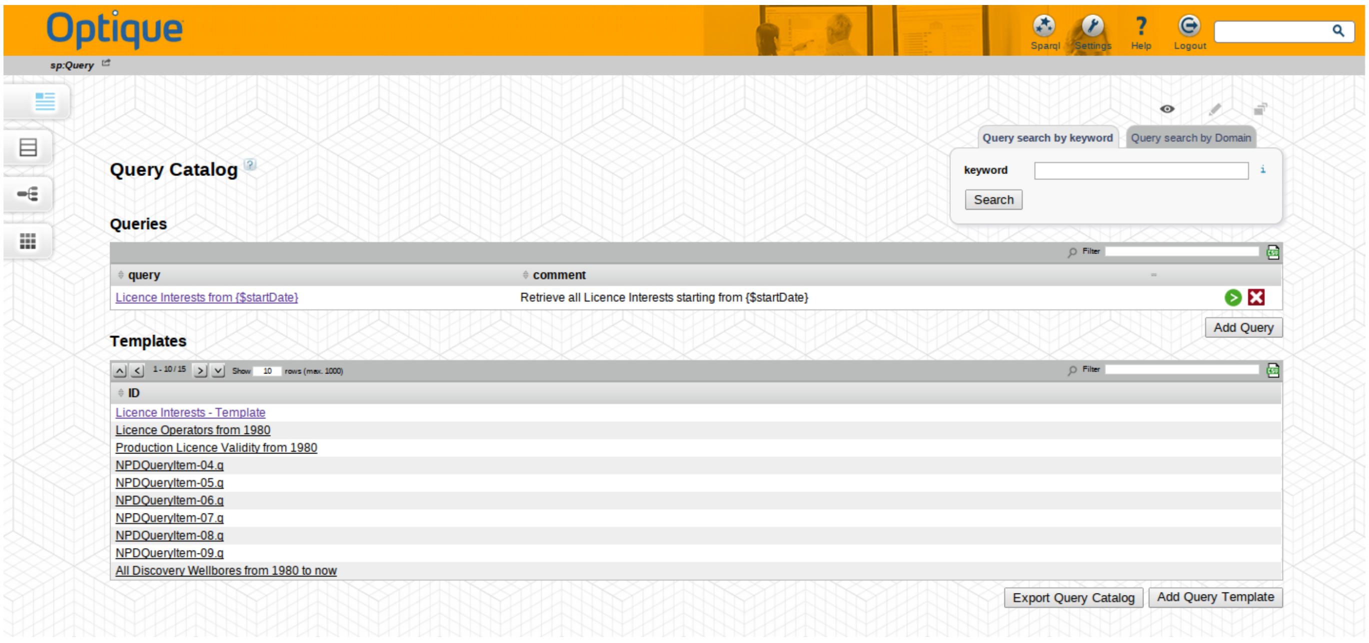
Query Catalog

Formalized information needs (i.e. SPARQL queries) can be stored as first order objects in the query catalogue. This capability enables one to build a dedicated catalogue of information needs. Queries can be

- manipulated

- parametrized using templates
- executed against different repositories
- organized and searched according to their natural language description and categories

Queries can be entered into the catalogue using either the syntactical SPARQL Editor or the novel visual query formulation interface. Queries constructed in the visual query formulation interface can be stored in the catalogue for later (manual) refinement or parametrization.



Query Catalog

Queries

query	comment
Licence Interests from {$startDate}	Retrieve all Licence Interests starting from {$startDate}

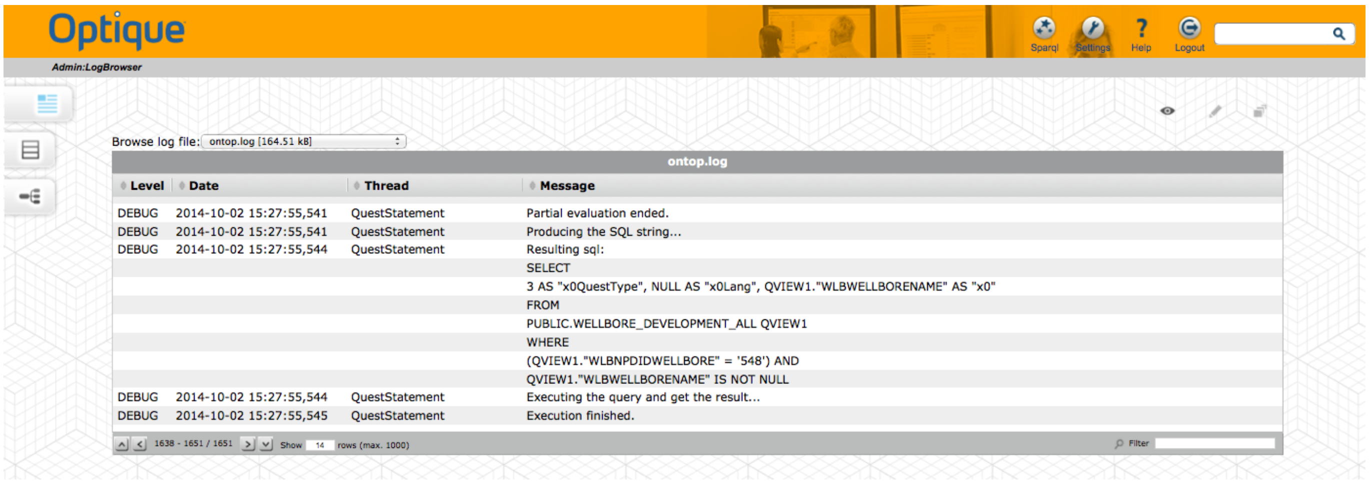
Templates

ID
Licence Interests - Template
Licence Operators from 1980
Production Licence Validity from 1980
NPDQueryItem-04.q
NPDQueryItem-05.q
NPDQueryItem-06.q
NPDQueryItem-07.q
NPDQueryItem-08.q
NPDQueryItem-09.q
All Discovery Wellbores from 1980 to now

Figure 3.11: Overview of query instances and query templates in the query catalogue.

Advanced Configuration and Debugging

The settings page (linked from the top menu bar and the start-page) provides access to advanced system configuration parameters in order to set, for example, query time-outs. Furthermore, a log browser (cf. Figure 3.12) can be utilized to access different log files. Particularly, log messages from the query transformation component can be used to trace and debug, for example, erroneous or long running queries.



```
ontop.log
Level   Date                      Thread           Message
DEBUG   2014-10-02 15:27:55,541   QuestStatement   Partial evaluation ended.
DEBUG   2014-10-02 15:27:55,541   QuestStatement   Producing the SQL string...
DEBUG   2014-10-02 15:27:55,544   QuestStatement   Resulting sql:
                                                   SELECT
                                                   3 AS "x0QuestType", NULL AS "x0Lang", QVIEW1."WLBWELLBORENAME" AS "x0"
                                                   FROM
                                                   PUBLIC.WELLBORE_DEVELOPMENT_ALL QVIEW1
                                                   WHERE
                                                   (QVIEW1."WLBNPDIDWELLBORE" = '548') AND
                                                   QVIEW1."WLBWELLBORENAME" IS NOT NULL
DEBUG   2014-10-02 15:27:55,544   QuestStatement   Executing the query and get the result...
DEBUG   2014-10-02 15:27:55,545   QuestStatement   Execution finished.
```

Figure 3.12: Detailed tracing log from the query transformation component.

3.2.2 Full Administrator Guide

A full documentation of features for administration of the platform core as well as for developing customized domain extensions, ships with the platform bundle. The documentation covers in particular topics such as system settings, wiki management, access to the platform's APIs via REST and CLI and the management of user rights. The developers guide explains basic plugin and extension mechanism for developing customized solutions, for example, domain specific query answer visualisation widgets.

3.3 End User Documentation

3.3.1 General Platform Features for Exploration, Search, Authoring and Visualisation

Platform features that specifically target End-User needs such as different kinds of widgets for query result visualization and browsing through the data, are documented. The help is included in the platform bundle as well as available through the online reference (see above). It documents the underlying key concepts of template mechanism for browsing, searching and semantic authoring of resource using a semantic wiki approach.

3.3.2 Visual Query Formulation

The Optique *Visual Query Formulation (VQS) system* initially includes three widgets as depicted in Figure 3.13:

- The first widget (W1 - see the bottom-left part of Figure 3.13) is a menu-based query by navigation widget and allows users to navigate concepts through pursuing relationships between them, hence joining relations in a database.
- The second widget (W2 - see the bottom-right part of Figure 3.13) is a form-based widget, which presents the attributes of a selected concept for selection and projection operations.
- The third widget (W3 - see the top part of Figure 3.13) is a diagram-based widget and provides an overview of the constructed query and affordances for manipulation.

These three widgets are orchestrated by the system, through harvesting event notifications generated by each widget as a user interacts, to jointly extract and represent the information need of a user.
In a typical query construction scenario, a user first selects a kernel concept, i.e., the starting concept, from W1, which initially lists all domain concepts accompanied with icons, descriptions, and the potential/approximate number of results. The selected concept becomes the focus/pivot concept (i.e., the node coloured in orange or highlighted), appears on the graph (i.e., W3) as a variable-node, W2 displays its attributes, and W1 displays all concept-relationship pairs pertaining to this concept.

The user can select attributes to be included in the result list (i.e., using the "eye" button) and/or impose constraints on them through form elements (i.e., W2). Currently, the attributes selected for output appear on the corresponding variable-node in black with a letter "o", while constrained attributes appear in blue with letter "c". Note that W1 does not purely present relationships, but combine relationship and concept pairs (i.e., relationship and range) into one selection; this helps us to reduce the number of navigational levels that a user has to pass through. The user can select any available option from the list, which results in a join between two variable-nodes over the specified relationship and moves focus to the selected concept (i.e., pivot). The user has to follow the same steps to involve new concepts in the query and can always jump to a specific part of the query by clicking on the corresponding variable-node. The arcs that connect variable-nodes do not have any direction, since for each active node only outgoing relationships, including inverse relationships, are presented for selection in W1; this allows queries to always be read from left to right.

An example query is depicted in Figure 3.13 for the Statoil use case. The query asks for all fields that contain an oil producing facility and are operated by the Statoil company. In the output, we would like to

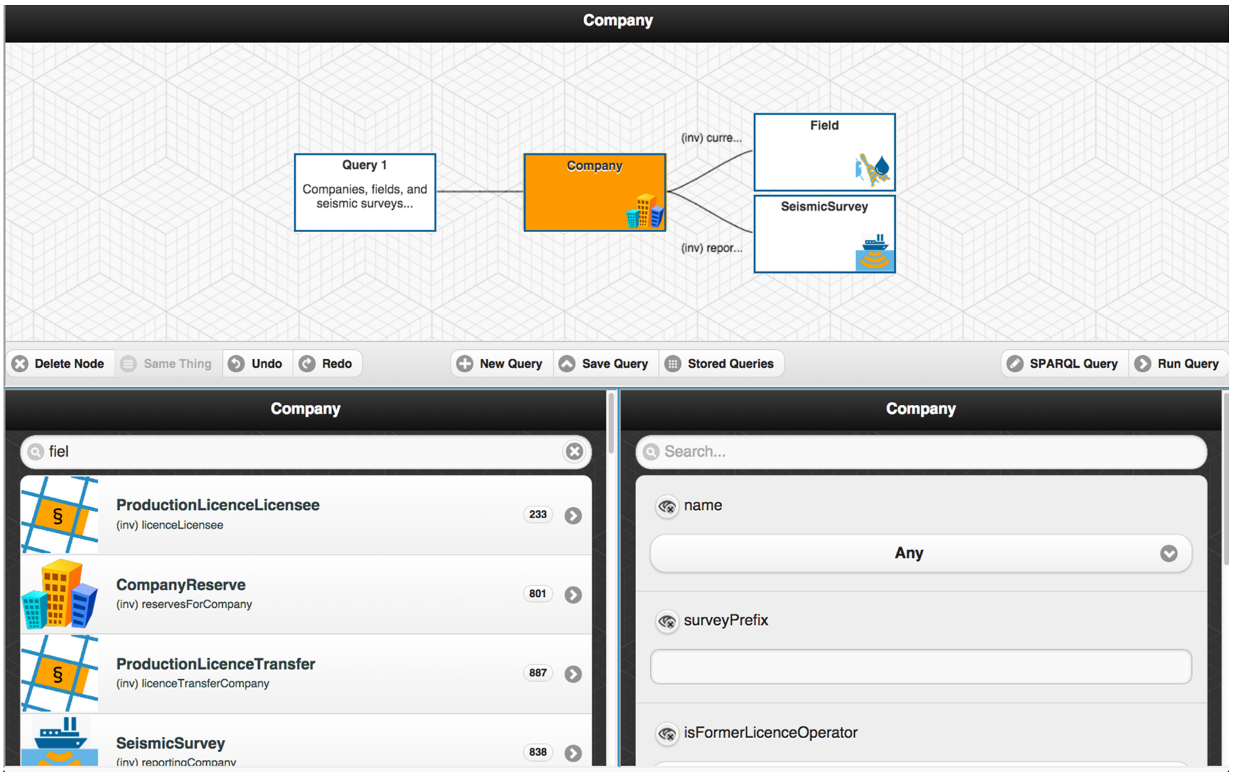


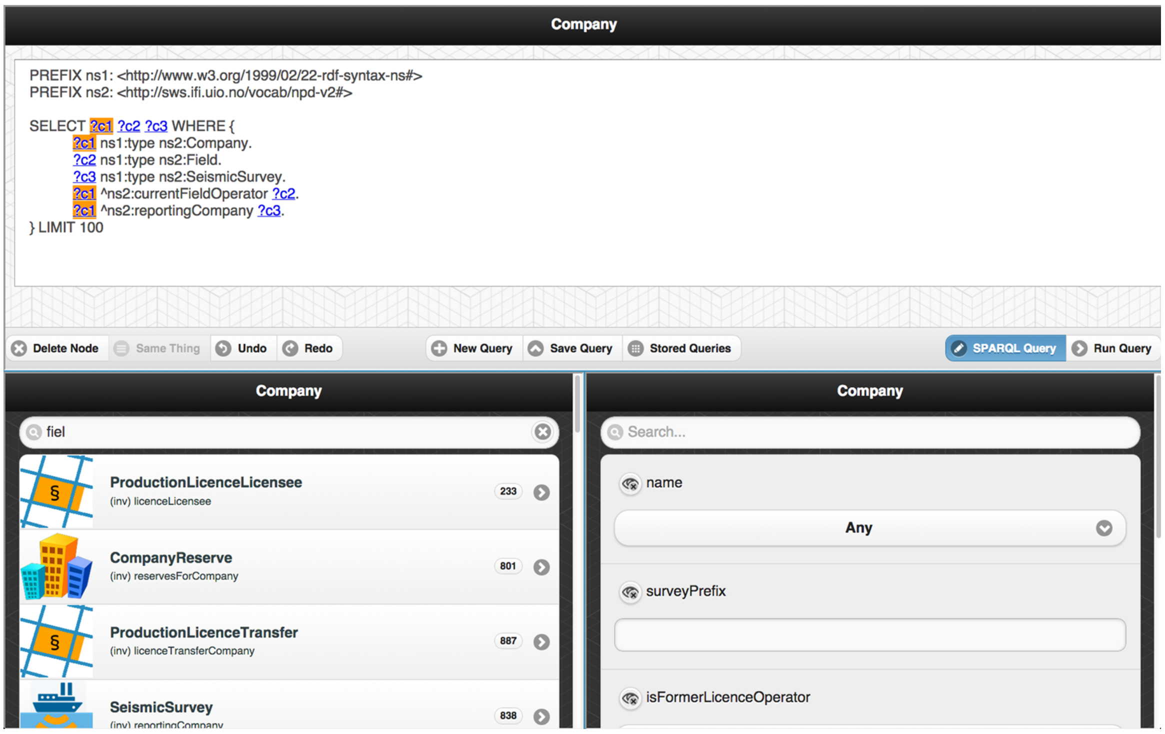
Figure 3.13: VQS – an example query is depicted for the Statoil use case.

see the name of the field and the name of the facility. The user can delete nodes by switching to delete mode or assert that two variable-nodes indeed refer to the same variable (i.e., cyclic query). Affordances for these are provided by the buttons at the bottom-left part of the W3. Currently, users can delete nodes, manage queries, and undo/redo their actions over a working query.

The user can also switch to SPARQL mode and see the textual form of the query by clicking on "SPARQL Query" button at the bottom-right part of the W3 as depicted in Figure 3.14. The user can keep interacting with the system in textual form and continue to the formulation process by interacting with the widgets. For this purpose, pivot/focus node is highlighted and every variable-node is made clickable to allow users to change focus. Currently, the textual SPARQL query is non-editable and is for didactical purposes, so that advanced end-users, who are eager to learn the textual query language, could switch between two modes and see the new query fragments added after each interaction.

Users can save and load queries directly from the VQS as Figure 3.15. This not only serves query management needs, but also enables users to see and use queries crafted by the other users. This passive form of collaboration allows users to construct complex queries by modifying and adapting existing queries.

Different input widgets can be attached to form elements in W2, for instance in Figure 3.16, a map widget is attached to name element of the Field concept via a location icon. This way, the user can select a Field instance from the map, rather than typing its name. This example is a step towards addressing geospatial queries in VQS and a similar widget-driven solution is available for the temporal and stream-based queries.



```
PREFIX ns1: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ns2: <http://sws.ifi.uio.no/vocab/npd-v2#>

SELECT ?c1 ?c2 ?c3 WHERE {
        ?c1 ns1:type ns2:Company.
        ?c2 ns1:type ns2:Field.
        ?c3 ns1:type ns2:SeismicSurvey.
        ?c1 ^ns2:currentFieldOperator ?c2.
        ?c1 ^ns2:reportingCompany ?c3.
} LIMIT 100
```

Figure 3.14: VQS – the example query is depicted in SPARQL form.

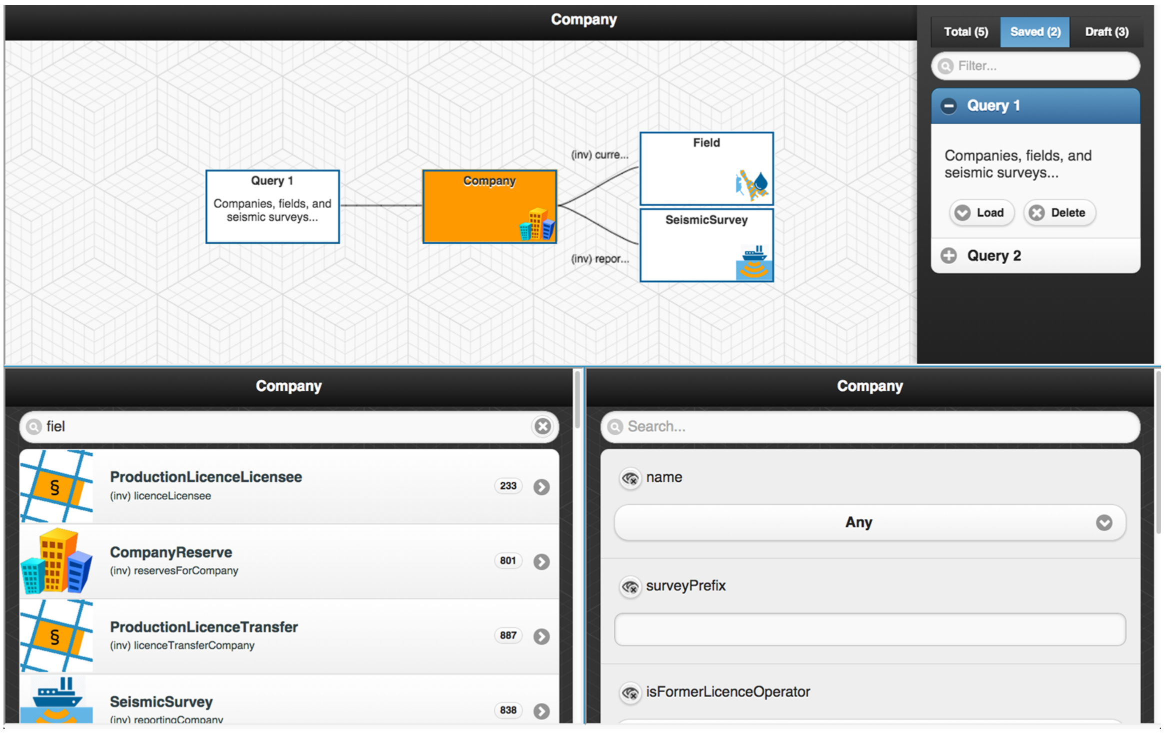


Figure 3.15: VQS – the query management.

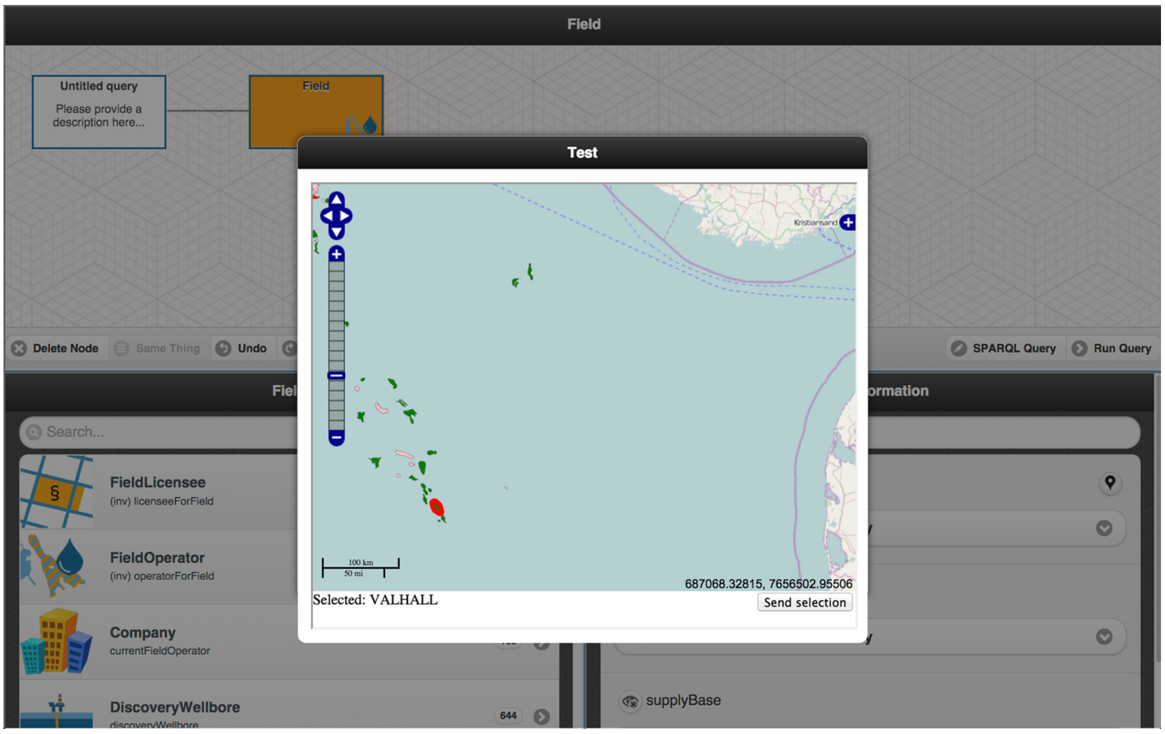


Figure 3.16: VQS – map widget.

Chapter 4

Conclusion and Outlook

In this deliverable we presented the second prototype of the Optique platform. In its core the Optique platform rests on the Information Workbench, which serves as a semantic platform for integration. The platform has been extended with specialized programming interfaces according to the requirements specification and overall architecture as defined in Deliverable 2.1. The shared interfaces enable a seamless integration and interaction of components through a unified semantic layer of abstraction. More specifically, the platform allows components to access required OBDA assets such as ontologies, mappings, metadata and queries from the platform's shared metadata repository based on open standards (e.g. OWL, R2RML, SPIN, SPARQL) and in a convenient way (cf. Section 2.4).

As a result, the Optique platform provides a single point of entry for administrative tasks (e.g. management of mappings and ontologies) as well as visual components through which end users can satisfy their information needs in interacting with Big Data (cf. Section 3.2.1).

Compared to the initial prototype (cf. Deliverable 2.3) the shared programming interfaces have been refined and matured both in terms of standard compliance and functionality. Particularly, the re-factored *Data Source & Relational Metadata Management API* eases the process of connection to different data sources and seamlessly integrates the distributed query execution engine. The core of the R2RML *Mapping Management API* have been released as open source project and moved to a public repository underpinning its maturity.

On the user interface side we have improved the overall usability to better support IT-Experts in setting up and maintaining the Optique system. More specifically, we have added new back-end functionality and front-end components to, for example, to be seamlessly edit mappings and queries.

The platform has been successfully deployed on Siemens and Statoil premises. Relevant user interface components such as those for visual query formulation and browsing have been presented to a group of End-Users during the respective workshops.

Ongoing work with respect to the third year, focuses on cross-component optimization and tuning of the query transformation and query execution engine. While we have prototypically integrated an initial version of the module for handling time and stream, we expect to have a full working version available for the third year's prototype. We continuously improve the usability in order to enhance the overall user experience as well as to lower the barrier of entry for both IT-Experts and End-User.

Bibliography

[1] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. The MASTRO system for ontology-based data access. *Semantic Web*, 2(1):43–53, 2011.

[2] Jim Crompton. Keynote talk at the W3C Workshop on Semantic Web in Oil & Gas Industry: Houston, TX, USA, 9–10 December, 2008. available from `http://www.w3.org/2008/12/ogws-slides/Crompton.pdf`.

[3] Mariano Rodriguez-Muro and Diego Calvanese. High Performance Query Answering over DL-Lite Ontologies. In *KR*, 2012.

[4] Marius Strandhaug. An R2RML mapping management API in java – making an API independent of its dependencies. Master's thesis, Department of Informatics, University of Oslo, 2014.

Glossary

ADP	Athena Distributed Processing
API	Application Programming Interface
CLI	Command Line Interface
DL	Description Logic
IWB	FOP Information Workbench
JDBC	Java Database Connectivity
NPD	Norwegian Petroleum Dictorate
OBDA	Ontology-based Data Access
OS	Operating System
OWL	Web Ontology Language
O&M	Ontology and Mapping
QA	Query Answering
PoJo	Plain Old Java Object
QF	Query Formulation
QT	Query Transformation
RDB	Relational Data Base
RDBMS	Relational Data Base Management System
RDF	Resource Description Framework
REST	Representational State Transfer
R2RML	RDB to RDF Mapping Language
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
URI	Uniform Resource Identifier
VQS	Visual Query Formulation System
W3C	World Wide Web Consortium