JDEVELOPER 11G R2 JENA ADAPTER EXTENSION

Petar Ristoski
Faculty of Computer Science and Engineering
Skopje, Macedonia

Marjan Efremov
Open Mind Solutions
Skopje, Macedonia

Vladimir Zdraveski
Faculty of Computer Science and Engineering
Skopje, Macedonia

Dimitar Trajanov
Faculty of Computer Science and Engineering
Skopje, Macedonia

ABSTRACT

Supporting the idea of the semantic web, we developed an intelligent GUI extension for the Oracle JDeveloper 11g R2.ThisGUI extension supports the basic Oracle semantic scenarios (set table space, test query with variables, call pre-written query from file and etc.) from the client side and is also a quite extensive system, thus it is very easy to add an additional functionality. The Jena adapter for Oracle 11g is used to connect to the database engine, placed on the server side. Thus, a complete client side development environment for SPARQL programming is now available and the semantic query management, as a simple concatenation of strings inside the code, seems to becoming just a forgotten nightmare of our semantic web programmer's life.

I. INTRODUCTION

The technologies of the Semantic Web allowed development of novel approaches to the data storage and retrieval. A semantic search system is essentially an information retrieval system which employs semantic technologies in order to enhance different parts of the information retrieval process. This is achieved by semantic indexing and annotation of content, query expansion, filtering and ranking the retrieved information [1].

The semantic search also introduces additional possibilities, such as search for an online ontology [2], search for online (distributed) knowledgebase, retrieval of facts from the ontology and knowledgebase and question answering.

In terms of providing better database support for the Semantic Web based applications, Oracle invented the Oracle 11g semantic database [3], which provides environment for storage and retrieval of semantic data, represented as RDF triples [4]. Together with the database, a Jena adapter for Oracle Database was written [5]. By use of the adapter, any Jena-based semantic application can be migrated on the Oracle 11g database. The performance and opportunities obtained by the Oracle 11g are comparable with the basic Jena (file based) solution. Retrieval is made by use of Jena Graph API [6] or with direct use of SPARQL [7].

But many of the semantic application developers describe as a very complex task the development environment and the semantic application development. So we succeeded to implement a JDevloper extension, where the users can easily set the environment and use semantic database directly from GUI interface.

II. RELATED WORK

The Oracle 11g is the latest database release by Oracle, leading and only commercial database with native RDF/OWL data management. It is open, standards-based, scalable, secure, reliable and performing RDF management platform. Based on a graph data model, RDF data (triples) are persisted, indexed and queried, like other object-relational data types. Application developers use the power of the Oracle Database 11g to design and develop a wide range of semantic-enhanced business applications.[8] Stores rich, real-world relationships in the data beyond columns, table joins and Boolean to obtain more semantically complete query results. Enables machine-driven discovery of new relationships using the native Oracle Database inference engine, ontologies, and RDFS/SKOS/OWL semantics and user defined rules. Inferred data is persisted in the database for faster querying. [9]] Oracle Database Strengths are Scalability (Trillions of triples), Availability (tens of thousands of users), Security (protect sensitive business data), Performance (timely load, query & inference), Accessibility (to enterprise applications) and Manageability (leverage IT resources) [10].

On the other hand, there is the well-known Jena. It is a Java framework for building Semantic Web applications and provides an API to extract data from and write to the RDF graphs. The graphs are represented as an abstract "model". A model can be sourced with data from files, databases, URLs or a combination of these. A Model can also be queried through SPARQL and updated through SPARUL [11]. Jena also provides support for OWL [12].

Sesame [13] is another existing standard framework for processing RDF data. It can be deployed on top of a variety of storage systems, relational databases, in-memory, file systems, keyword indexers, and offers a large scale of tools to developers to leverage the power of RDF and related standards. Sesame fully supports the SPARQL query language for expressive querying and offers transparent access to remote RDF repositories using the exact same API as for local access.

Adapters that enable semantic frameworks to be deployed over Oracle Database 11g were also released. The Jena Adapter for Oracle Database provides a Java-based interface to Oracle Database Semantic Technologies by implementing the well-known Jena Graph and Model APIs. Similarly, The Sesame Adapter for Oracle Database [14] integrates the

popular Sesame Java APIs with Oracle Semantic Technologies support.

All these tools are not quite simple to use, especially for beginners in semantic Web and Oracle database. And the biggest drawback of these concepts is that the users have to set the development environment manually, which is not an easy task. Second biggest drawback is that the users have to build their queries as a concatenation of strings and send the whole query every time they need to execute it. Thus, the management of the queries becomes quite chaotic in large applications and sometimes it is almost impossible to code and/or make changes. Our main problem is that there isn't (wasn't) an environment similar to the stored procedures framework or of any other kind that provides those functionalities for the Semantic Web querying process.

III. SOLUTION DESCRIPTION

In the context of solving the described problem, we developed a JDeveloper11g R2 [15] extension that encapsulates and extends many of the Jena adapter functionalities in order to store and retrieve semantic data into the Oracle 11g database. The extension was developed using the extension SDK for JDeveloper 11g R2.

The extension is accessible via the context menu in the navigation pane, for every project, displayed as submenu, Fig. 1.The extension uses the external Jena adapter libraries, to access the Oracle 11g database, and our custom made extension API, to enable access to the server configuration parameters from the new project's java code. When any of the functionalities is used, all the needed libraries are automatically added to the classpath configuration of the project.

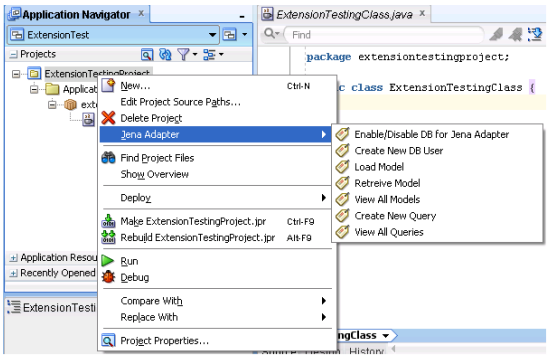

Figure1: Extension submenu inside the context menu in navigation pane.

As is shown on Fig. 1, there are 7 entries in the Jena adapter submenu in the current version of the extension. We will take a closer look on each of them.

A. *Enable/Disable DB for Jena Adapter*

If the user selects this entry from the submenu, the dialog from Fig. 2 will be shown. With the use of this dialog the user can easily enable or disable the semantic technologies environment inside the Oracle 11g DB.

In order to do that, the user has to provide credentials as sysdba user. After the user enters the jdbc URL to database and the password for the sys user, it has the option to test the connection for the given login information. If the connection is successful the user should fill the rest of the fields. The "tablespace" attribute is the name of the tablespace, used to store the users and their semantic data. The user also needs to provide the Oracle home directory path of the Oracle 11g database, choose if the database is installed locally, or on a remote server. If the database is installed on a remote server, the server has to support ssh connection, and login credentials of the ssh user should be provided.

The user can choose whether to enable or disable semantic technologies environment in the database. If the database is on a remote server, a ssh connection is established directly via the java sshj API. The following steps are the same ina remote as in a local machine scenario. The appropriate sql script is executed, with the privileges as sysdba user. The semantic tablespace and the initial semantic network are created/removed respectively.

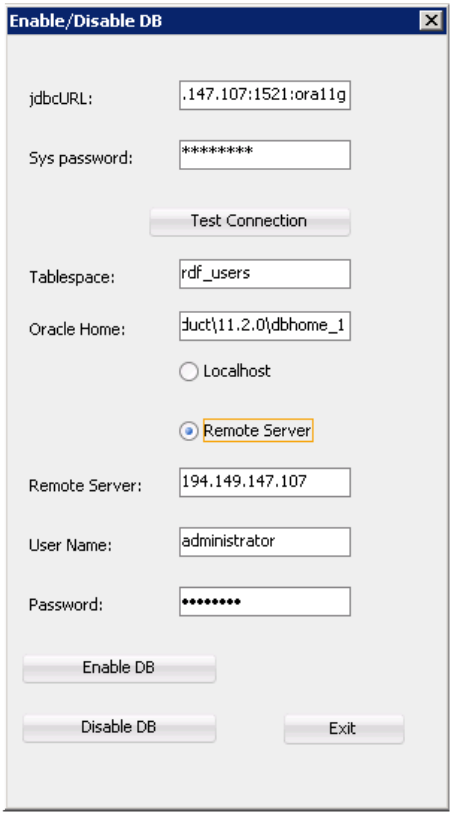

Figure2: Enable/Disable semantic technologies environment in database wizard

If the user chooses to enable/disable already enabled/disabled semantic environment, he will be prompted and notified that is not allowed to do that action.

B. Create a New Database User

After the semantic technologies environment in the database has been enabled, a database user, that will have the privileges to work with semantic data, has to be created.

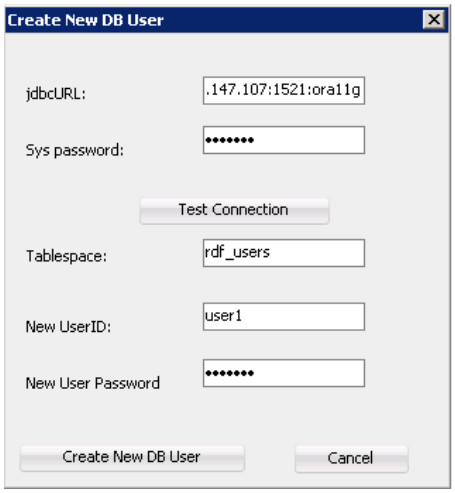

Figure3: Create new database user wizard

The user has to provide the sys password, the tablespace, where the new user will be added, and login information about the new user. After the user clicks the "Create new DB user"-button, a connection to the database is established as sysdba user, new user is created and the needed privileges are granted to him. After that, the new user is ready to work with semantic data.

C. *Load/Create Model*

If the user selects this entry from the submenu, the dialog from Fig. 4 will be shown. The user can load existing models from files or to create new one. First the user needs to provide the database connection information and enter the model name. If the user wants to load existing model from file, he should select the file, select the loading type, which can be incremental, bulk or batch, and has the option to perform OWL prime inference over the model.

If the user wants to create a new model, then the file field should be left blank.

When the user clicks the "Load model"-button, a connection to database is established and the model is loaded to database or a new model is created.

If this is the first loaded or created model in the current working project, then a config.xml file is added to the project. This file is used by the extension to store the configurations about the extension, the database connections, the data models and the user queries.

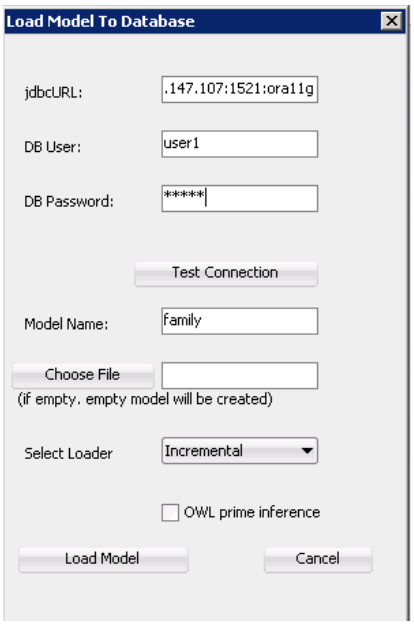

Figure4: Load/Create model to database wizard

D. Retrieve Model

The Retrieve Model command shows the dialog from Fig. 5.From this dialog the user can read models from his database.

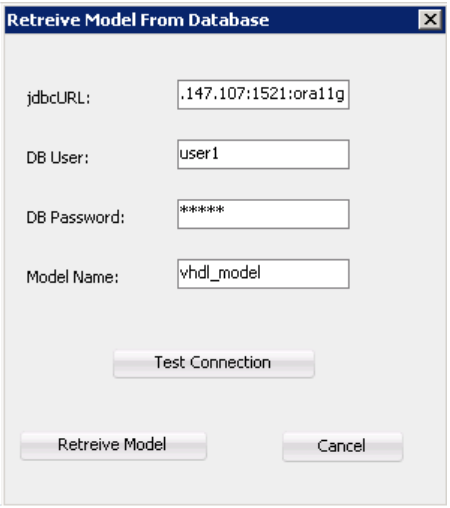

Figure5: Retrieve model from database wizard

The user should provide all the database connection information and the name of the model that wants to retrieve. When the user clicks the "Retrieve model"-button, a connection to the database is established and the model information is stored in the config.xml file. If the config.xml file does not exist, it will be created.

E. View All Models

Via this dialog form, shown in Fig. 6,the user can preview the information about all the models saved in the config.xml.

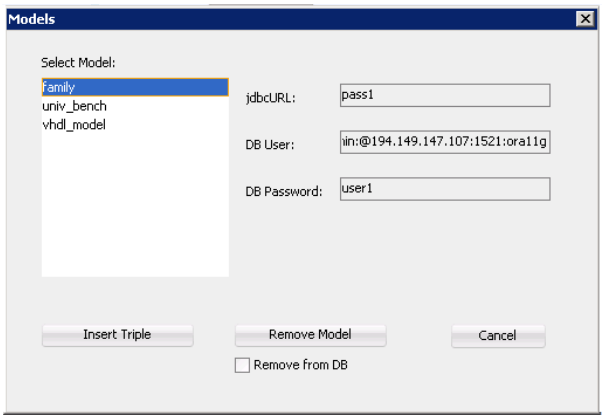

Figure6: View all models wizard

The user can add triples to the model by clicking "Insert triple to model"-button and filling the dialog form from Fig. 7. The user can enter the subject, predicate, object of the triple, he wants to insert into the model. After the "Insert triple"-button click, the triple will be inserted in the previously selected semantic model.

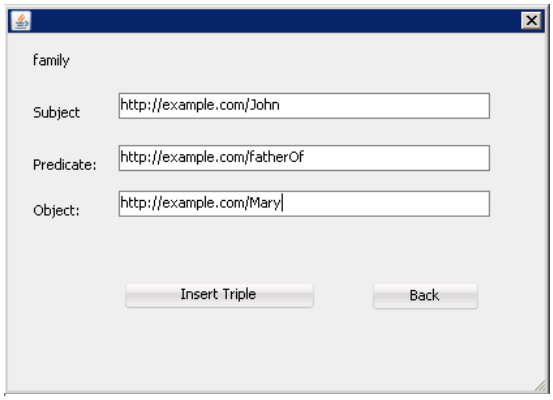

Figure 7: Insert triple in model wizard

F. *Create New Query*

In this dialog, Fig. 8, the user can create and store query, for some of the created or retrieved models. When the dialog loads the list will be filled with all the available models stored in the config.xml. The user has to select one of the models, and then he can write the query.

In this point, we proposed a trivial solution to the problem of constructing complex queries. Namely, we made a simple extension to the SPARQL language, by defining a notation for a SPARQL query variable parameter.

Inside the query, the user can use variables, that should be marked with the predefined notation, ${variable}$. Using variables in the query reduces writing the same query for different values and is a step forward to creation of a semantic stored procedures environment, presented in [16], for SEM_MATCH queries. The user has an option to test the written query by clicking the "Test query"-button. The dialog on Fig. 9 will be shown. The query is parsed in background and all the variables are detected. In this dialog an input field is shown for each variable, expecting the user to explicitly set the values.

Then the query will be executed against the previously selected model and if the query is correct, the result will be shown to the user. Otherwise, the appropriate error will be displayed to the user.

When the user has finished the query, he can save it as a new query.

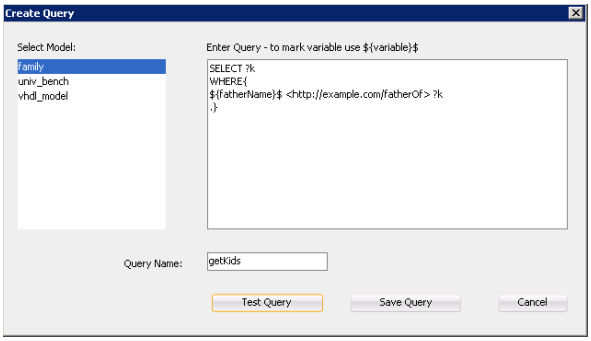

Figure8: Create a new query wizard

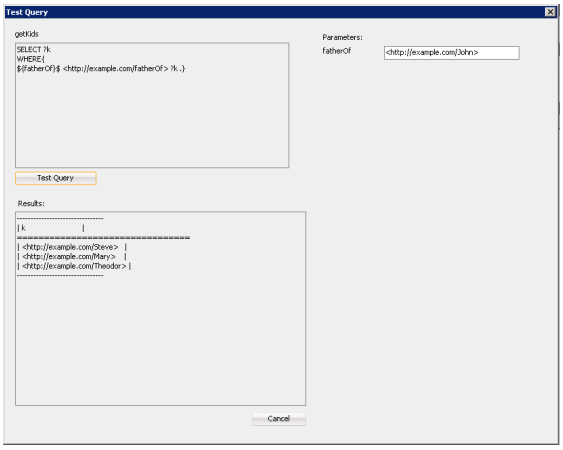

Figure9: Test query wizard

G. *View All Queries*

If the user selects this entry from the submenu, the dialog from Fig. 10 will be shown. In the first list are shown the models stored in the config.xml file. When the user selects a model, the queries from the selected model are listed in the second list. If the user selects a query, the query will be shown and the user can modify, delete or test it using the command buttons.

H. *Extension API*

We saw how the users can easily load or create the models and create and test queries. It is the first step in the application development, when the user sets the data tier of his application. The next step is manipulation with the data from the business tier. With this API the user can quickly retrieve models or execute queries, previously created with the extension GUI, directly from their Java code.

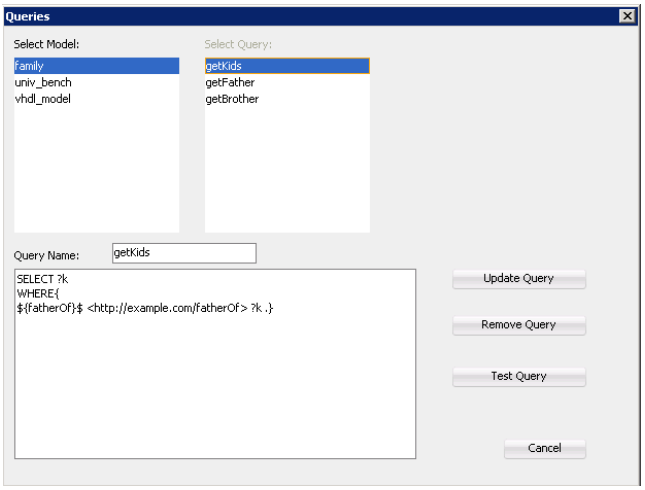

Figure10: View all queries wizard

Thus, the API describes the main point of our extension, which is the ability of independent query building and testing. After the query is completed, the user can use it only by reference to its source file. This can be treated as some kind of a client-side stored procedures scenario, which will provide all the required functionalities during the implementation phase. So, the programmers will not have to write those complex string concatenations in order to test and correct their query.

In the current release, the API provides the following three functions:

- *Model getModel(String name);* returns the model identified by the model name.
- *String getQuery(String modelName, String queryName, List<String>variablesValues);* returns the query string, identified by the model name and the query name, and the variables are replaced with the list of variable Values, if any.
- *ResultSetexecuteQuery(String modelName, String queryName, List<String>variablesValues);* executes the query identified by the model name and the query name, and the variables are replaced with the list of variable Values, if any variables, and returns the result as an object of the ResultSet class.

IV. CONCLUSION AND FUTURE WORK

The application developers often are trying to obviate the use of semantic web and the semantic web technologies, besides the advantages of using them, only due to the complexity to set up the development environment. With the use of this extension, it is simple to set the semantic data tier for any application, just by completing several easy steps from the GUI user interface, so the users can quickly move to the development of the business and the presentation tier of their application.

The Jena adapter extension can be improved in several directions. The UI can be redesigned to a more user friendly one. The SPARQL editor will offer syntax highlighting and code completion, so the user can easily construct complex queries. Also a visual editor can be added, so the user can build the queries with drag and drop of the keyword/construct. There will be also an option to choose the format of the result from the executed query. For example, the result will be shown in visual editor as highlighted nodes in tree, where the tree is visualization of the whole data model, or the result can be exported as a XML file, or it can be displayed in a custom output, defined by the user. Besides the SPARQL queries, the users will be able to write SEM_MATCH queries, which are natively support in Oracle 11g database.

In this solution the queries are stored locally in the project directory, and sent to the database each time when executed. In the future extension releases, the queries will be stored in the database as stored procedures, similar to the solution described in [16], extended to SPARQL queries instead of SEM_MATCH ones.

We expect that this extension will encourage the application developers to use the semantic web technologies more often,t hat will lead to a development of more advanced and customized semantic applications.

REFERENCES

[1] Strasunskas, D. and Tomassen, S.L. On Variety of Semantic Search Systems and Their Evaluation Methods. Proceedings of International Conference on Information Management and Evaluation, University of Cape Town, South Africa, 25-26 March 2010, Academic Conferences Publishing, pp. 380-387.

[2] Pan, J.Z., Thomas, E. and Sleeman, D. (2006) "Ontosearch2: Searching and querying web ontologies", In Proc. of the IADIS International Conference, pp 211-218.

[3] Oracle 11g Database - http://www.oracle.com/technetwork/database/enterprise-edition/overview/index.html

[4] RDF, Resource Description Framework, http://www.w3.org/RDF/, 2010

[5] Jena Adapter for Oracle Database - http://download.oracle.com/docs/cd/E18283_01/appdev.112/e11828/sem_jena.htm

[6] Jena – A semantic web, java framework, Official API documentation and examples for Jena libraries, http://jena.sourceforge.net/,2010

[7] SPARQL - http://www.w3.org/TR/rdf-sparql-query/

[8] Semantic Technologies Center - Oracle http://www.oracle.com/technetwork/database/options/semantic-tech/whatsnew/index-088828.html

[9] Oracle Database 11g Semantic Features http://www.oracle.com/technetwork/database/options/semantic-tech/whatsnew/semtech-makes-ed-sm-195114.html

[10]Oracle Database 11g Semantics Technical Talkhttp://www.oracle.com/technetwork/database/options/semantic-tech/whatsnew/oracle-33.pdf?ssSourceSiteId=otnjp

[11]SPARUL, SPARQL update, http://www.w3.org/Submission/SPARQL-Update/

[12]OWL, Web Ontology Language, http://www.w3.org/TR/owl-features/

[13]Sesame, A semantic web, java framework, http://www.openrdf.org/about.jsp

[14]Sesame Adapter for Oracle Database – http://download.oracle.com/docs/cd/E18283_01/appdev.112/e11828/sem_sesame.htm

[15]JDeveloper 11g R2, http://www.oracle.com/technetwork/developer-tools/jdev/jdev-11gr2-nf-404365.html

[16] M. Efremov , V. Zdraveski, P. Ristoski, D. Trajanov – "Semantic Stored Procedures Programming Environment and performance analysis" – "ICT Innovations 2011", September 2011