Ontology-Driven Sub-Query Extraction for Distributed Autonomous Information Resources in UnIT-Net IEDI

Vadim Ermolayev, Natalya Keberle, Vladimir Shapar, Vladimir Vladimirov

Dept. of IT
Zaporozhye State Univ.
Zhukovskogo 66, 69063, Zaporozhye, Ukraine
{eva, kenga, wws, vvlad}@zsu.zp.ua

Abstract: The paper reports on the development of UnIT-Net[1] Infrastructure for Electronic Data Interchange (IEDI). The main task of IEDI is to provide a uniform framework for authorized and secure information retrieval from heterogeneous, distributed, autonomous Information Resources (IRs) among the higher educational establishments and state bodies in Ukraine. The focus of the paper is the algorithm for ontology-driven sub-query extraction. The algorithm performs terminological mapping of an initial query in terms of domain ontology to the set of the sub-queries to different IRs in terms of respective IR ontologies. Mapping procedure is based on the raw mappings knowledge taken from mediator mapping ontology and the so called Late Binding technique for determining concepts. Finally, sub-queries are refined to become correct RDQL queries with respect to the specific IR. This algorithm will be used by IEDI mediator to decompose the user queries. Sub-queries will then be executed by the wrappers of the respective IRs. Initial proof-of-concept evaluation and the semi-formal proofs of ontology-driven sub-query extraction algorithm correctness are provided in the discussion part of the paper.

1 Introduction

To achieve and sustain dynamic improvement, service-oriented organizations like Universities, need an infrastructure that underpins flexible and robust management of their activities and decision making support. To a large extent the activities within Universities as well as their coordination and control at National level involve the processing of enterprise data and knowledge. As far as the organizations involved in the educational framework are legally independent, they own and maintain their data and knowledge sources autonomously – i.e. independently from each other and, to a high degree, from the coordination body, like a National Ministry. The fact that these information resources are autonomous implies serious complications for their integration: they might be provided by different hardware and software using various notations and protocols; they might be disparately structured or even have different data models behind them; they are semantically heterogeneous.

[1] **UnIT-Net**: IT in University Management Network. TEMPUS/TACIS MP JEP 23010-2003.

The task of the IEDI, which is developed within UnIT-Net project, is to attempt to overcome some of these complications by providing a uniform framework for authorized and secure information retrieval from heterogeneous, distributed, autonomous IRs. One of the central points of IEDI is the homogeneous and coherent representation of the Domain and IR semantics by means of the family of ontologies. Therefore, the processes of querying IEDI IRs are ontology-driven and cannot be arranged entirely automatically. Preparing an IR to become available for querying requires intensive ontology engineering by human administrators with different roles.

Another aspect to be mentioned with respect to IEDI functional processes is the state of the system distribution. A query may demand to retrieve data from several physically distributed IRs which belong to different legal owners and are physically stored in different places. This is why IEDI processes are composed of a number of tasks and activities performed at distributed nodes. These tasks should of course be executed in a controlled and ordered way. A process normally involves both automated activities performed by the IEDI software and human activities, like ontology harmonization, supplied with appropriate methodologies and software tools. Human activities are performed by various user roles: Authorized User (AU), Mediator Ontologies Engineer (MOE), IR Ontology Engineer (IROE), IR Provider (IRP). Please refer to [Er04] for more details.

The paper presents the mechanism for ontology-driven sub-query extraction by IEDI mediator. The reminder of it is structured as follows. Section 2 presents the related work in the field of distributed information retrieval with special emphasis to the approaches to query decomposition. Section 3 sketches out the architecture of IEDI and outlines the principles on which the architecture was designed. Section 4 presents our algorithm for ontology-driven sub-query extraction. Section 5 and 6 provide the initial proof-of-concept evaluation example and the discussion of the algorithm. Section 7 gives the conclusions and the prospects for future work.

2 Related Work

The genre of the IEDI belongs to the distributed Intelligent Information Retrieval (I2R) domain within the broader area of Intelligent Information Integration (I3). The research activities within this domain have been intense in the past decade. Examples of R&D projects developing formal, algorithmic, architectural frameworks, deploying software prototypes for I2R from distributed, heterogeneous IR-s and Intelligent Information Integration (I3) are BUSTER [St00], DOME ([CJO01], [CJO02]), InfoSleuth [Ba97], KRAFT [Gr97], MOMIS [Be98], OBSERVER [Me00], Ontobroker [De99], PICSEL [LR00], SIMS [AKS96], TSIMMIS [Ga95], and others. A good survey of ontology-based approaches to I2R and I3 may be found in [Wa01].

Although all these projects use different techniques, approaches, and software paradigms for the task they identify similar pitfalls for the domain. The first group of possible pitfalls is the way in which semantic heterogeneity is resolved in the processes of ontology-based information integration. As outlined in [CJO01], this includes the aspects of developing ontologies (bottom-up and top-down approaches), mapping between ontologies, and relationships between ontologies and IRs.

Most projects adopt one of the following approaches to using ontologies [Wa01]: single ontology (SIMS), multiple ontology (OBSERVER), hybrid approach (BUSTER, DOME). Mapping between ontologies is necessary when a system uses several ontologies either "horizontally" (as in multiple ontologies approach) or "vertically" (as in hybrid approach). Mappings between ontologies within the system provide links between equivalent or related elements of ontologies, thus ensuring ontology re-use. Mappings between ontologies and IR schemas maintain correspondences between ontology elements and the elements of data schemas. As stated in [CJO01], mapping between ontology elements and data schema elements makes for transparent execution of user queries within the system as a whole.

The second group of possible pitfalls concerns the aspects of supplying autonomy and dynamic nature of the open system elements. The solutions here advocate one of the mediator architectures: centralized and decentralized. Centralized mediator architecture provides for one centre (e.g., TSIMMIS), which stores all the information about ontologies, IRs, mappings between them, and controls the query formulation and execution. A decentralized mediator architecture provides a separate agent/wrapper for each IR, which stores mappings between global/shared ontology(-ies) and the underlying IR (e.g., RACING [Er03]). The resource broker communicates with resource agents/wrappers and determines relevant and accessible resources for every query personally (e.g., InfoSleuth, SIMS, KRAFT).

The third group of possible pitfalls is formed by the tasks of query formulation, effective query decomposition without loss of information and query results merging and refinement. Known approaches to solving these tasks are: use of ontologies (hypernym/hyponym relationships) to reformulate queries containing terms which do not exist in the ontology(-ies) thus constructing query plans with no loss of information (OBSERVER); use of rewriting techniques together with mappings to produce queries on IRs that most effectively satisfy the input query (PICSEL).

Some of the mentioned problems have received only partial solutions. For example, the problem of semantic interoperability is typically partially solved by committing the participating nodes to a kind of a convention, providing the framework for semantic representations. These partial solutions evidently constrain the application domain and the functionality of the deployed software prototypes for I2R. The constraints for IEDI are as follows:

- IEDI is built on the principles of the mediator-wrapper architecture [Wi92] with the centralized mediator
- IEDI exploits the hybrid approach [Wa01] to knowledge representation
- IEDI uses information resource registration to allow an IR to become available for querying
- IEDI does not provide full automation of ontologies' mapping and alignment
- IEDI components use rewriting techniques with mappings to produce, process, and perform queries

The concept and the architecture of IEDI use some novelties which, in their combination, distinguish IEDI from the predecessors. IEDI Ontologies are specified in

W3C emerging de facto standard language OWL DL [Ow03]. Ontology-driven query formulation and transformation (RACING) is used for query processing. The semantics of a structured IR (e.g., RDB) is formalized by means of a semi-structured Ontology Specification Language (OWL DL), but not by specifications like, e.g., ODM_{I3}+ODL_{I3} [Be98]**.** Web Service technology is used for IR wrappers implementation.

IEDI mediator query language is RDQL [Rd04]. RDQL queries are formulated in terms of IEDI Mediator Domain Ontology (MDO). RDQL query denotes a connected RDF-graph comprising involved MDO concepts and properties. This initial query is then mapped onto the collection of IR Ontologies (IRO). This mapping procedure produces the set of sub-queries to respective IRs. Sub-queries are finally translated to the query languages of particular IRs (e.g., SQL) by their wrappers. This approach to query processing is generally similar to the one of OntoSeek [GMV99]. In OntoSeek a user query is also presented in the form of lexical conceptual graph (LCG), and then compared to the underlying OntoSeek ontology (LCG as well) to find matching elements. However, OntoSeek doesn't solve query decomposition problem – all the knowledge (both concepts and their instances) is stored in one ontology, and queries need not be decomposed.

OBSERVER project uses the family of several related ontologies and underlying resources satisfying particular ontology from the family. It uses special technique to query decomposition without loss of information, based on exploiting semantic relationships ("hyponimy-hypernimy" and other types) described between the concepts in one ontology. Similar technique based on the notion of upward "cotopies" (members of concepts' hierarchy which are equal to the given concept or are its nearest neighbors) is used in SEAL – Semantic Portal [St01] created within Ontobroker project. PICSEL uses rewriting technique for constructing the queries over particular resources. PICSEL uses knowledge about possible transformations between concepts in the form of deductive rules presented in a special language (CARIN).

3 IEDI Architecture in a Nutshell

The main purpose of IEDI is to provide for performing queries over the set of pre-registered, but independent, distributed and semantically heterogeneous IRs. This implies that IEDI is naturally a distributed system.

An important factor which seriously influenced the design of IEDI architecture is semantic heterogeneity of the IRs which are registered to IEDI mediator. This implied the use of the hierarchy of ontologies which actually drive the performance of distributed queries to different IRs. The tasks of merging and alignment of the ontologies describing the semantics of the IRs and the common ontology of the mediator (MDO) are performed manually. IEDI provides reference ontologies and tools for this ontology engineering activities. However, this thorough preparation work allows to further perform query formulation, sub-query extraction, sub-query execution tasks in a straightforward manner and almost automatically. The diagrams of IEDI query performance, IR Registration, and Ontology Coherence Maintenance scenarios are given and described in details in [Er04].

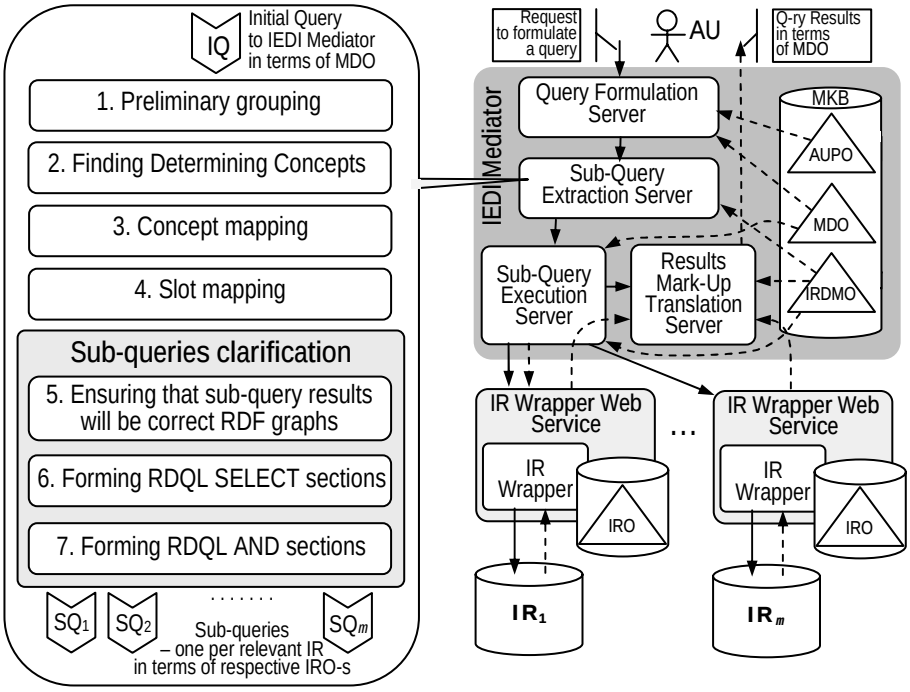

Figure 1: Sub-query extraction in IEDI reference architecture.

IEDI Architectural layering is defined according to the analysis of the IEDI processes and tasks and reflects the mediator-wrapper type of IEDI architecture (Fig. 1). The layering represents the overall IEDI organization and is outlined according to the following points of view:

– What are the Components, the Tools and the User Roles at the specific IEDI layers?
– How do IEDI Clients and Servers interoperate across the layers of its architecture?

IEDI User Layer is the environment for AUs and AU Clients. IEDI IR Wrapper and IR Layers represent autonomous, heterogeneous, and distributed IR holders. IEDI Mediator Layer is the holder for the components and the tools providing the means for mediation between the AU-s formulating queries and retrieving the results from the registered IR-s and respective IR Wrappers to provide the relevant information.

IEDI software components are split into two categories of Clients and Servers according to their functionality. IEDI Clients are related to IEDI AU-s and provide the interfaces for their activities. AU client provides IEDI interfaces for an AU. It functions in generic Web Browser environment (+ Java Virtual Machine) at the User Layer of IEDI Architecture (Fig. 1) and provides the interfaces for the tasks of: User Query Formulation, User Query Approval, Browsing Query Results. AU Client interoperates with the IEDI Query Formulation Tool and with the following IEDI components: IEDI Mediator Access Server and Query Formulation Server (the component of IEDI Mediator Server). MOE Client provides IEDI interfaces for the MOE. It functions in Java Virtual Machine (JVM) environment at the Mediator Layer of IEDI Architecture

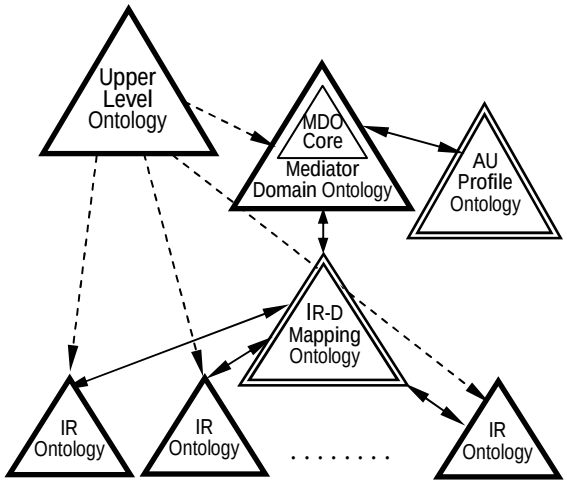

Figure 2: IEDI ontologies hierarchy.

and provides the interfaces for the tasks of IEDI Ontologies Discussion, Merge, Alignment, Editing and Repair. IROE Client provides IEDI interfaces for an IROE and is similar to MOE Client. It operates at the Mediator and the IR Wrapper Layers of IEDI Architecture and provides the interfaces for the tasks of IRO Ontology Discussion, Editing and Repair as well as for the Negotiation on IRO – MDO Merge within the IR Registration Process. MOE and IROE Clients interoperate with the following IEDI tools: Ontology Discussion and Alignment (under development in UnIT-NET), Ontology Editor (Protégé [Fn01]). MOE and IROE Clients interoperate with the following IEDI components: IEDI Mediator Access Server. In full detail IEDI reference architecture is specified in [Er04].

Ontologies play an important role in IEDI architecture as they drive its mainstream functionality – distributed query performance. The hierarchy of IEDI ontologies (Fig. 2) reflects the type of IEDI architecture. The following four types of ontologies are used: Upper-Level Ontology (ULO), domain ontology, IR ontology and reference ontology. ULO defines basic upper-level concepts and, by that, allows comparing any two of IEDI ontologies. ULO also serves as the foundation for discussion on concepts semantics between a MOE and any of IROEs. MDO is the domain ontology of IEDI. Its main utility is to represent the domain knowledge and to provide domain concepts to formulate queries. MDO is also the common mediator ontology by the procedure of its incremental design through IR registration. IROs are IEDI IR ontologies. An IRO presents the vision of IROE on the domain. IRO is used in the process of IR registration to IEDI mediator. Each registered IR should have its own IRO because it is used by the IR wrapper to perform queries. IEDI reference ontologies are AU Profile Ontology (AUPO) and IR-Domain Mapping Ontology (IRDMO). These ontologies store the knowledge on concept/slot mappings. AUPO is used for AU profiling and initial AU query formulation [Er03]. IRDMO provides for mapping of MDO concepts and slots to the concepts and the slots of the IROs of registered IRs. IRDMO is extensively used in the process of sub-query extraction.

4 Algorithm for Ontology-Driven Sub-Query Extraction (ODSQE)

One of the important steps in distributed query processing in IEDI is the extraction of the sub-queries from the initial query. Extracted sub-queries are then routed to the respective IR-s for execution by the IR wrappers. Basic idea of the extraction algorithm is to attempt to map MDO concepts/slots used in the initial query to the elements of IRO of

each registered IR. The resulting mappings to certain IROs will be not empty. Corresponding partial queries will be considered as the extracted sub-queries. Obviously the subset of the IRs with empty mappings to their IROs will not be used in the subsequent query processing steps.

Algorithm performs terminological mapping of MDO query to the set of IRO[*m*] queries. This mapping is based on the raw mappings taken from IRDMO and Late Binding procedure. Final refinement of the resulting sub-queries is then performed.

ODSQE starts with grouping the triples related to the same MDO concept – a determining concept. Then, it maps each determining concept to one or more corresponding concepts in IRO[*i*] as proscribed by IRDMO (Step 3). The algorithm then maps the slots from the initial query to corresponding (one or more) slots in IRO[*i*] (Step 4). Obtained Billet Sub-Queries (BSQ) are then refined (Steps 5,6,7) to get correct RDQL notation. This refinement procedure first removes the "hanging" triples which link mapped concepts/slots to unmapped ones. It then removes the variables (SELECT clause) and the conditions (AND clause) related to these "hanging" triples. High-level schema of ODSQE has been given in Fig. 1.

ODSQE input: one correct RDQL query (here and below – MDO query) satisfying current MDO state (all mentioned concepts/slots exist in MDO)

ODSQE output: set of *m* correct RDQL queries (here and below – IRO[*m*] queries) satisfying current IRO[*m*] states, where: *m* – the number of IROs currently possessing the mappings of the terms from the MDO query; $m \leq n$, where *n* – the number of IRs, currently registered to IEDI mediator; IRO[*i*] stands for *i*-th IRO possessing the terms from the input query ($1 \leq i \leq m$).

Step 1. Preliminary grouping
Group all the triples from WHERE clause of MDO query with the same <subj>-part by reordering.

Step 2. Finding Determining Concepts (DC)
For each triple group (**Step 1**) get corresponding slot names from the <predicate>-part and find a determining concept (from MDO) – the highest concept in the hierarchy, to which all these slots are attached by means of the domain property (OWL). These triple groups are hereafter referred to as DC Triple Groups (DCTG)[2].

Step 3. Concept mapping
Let *m*:=0, *flag*:=false
`For` $i = 1,...,n$
  `For` each DCTG
    Use IRDMO to find all concepts, equivalent to the DC in the IRO[*i*]
    Compute *k* – the number of such concepts (the value of *k* may be 0, 1 or more)
    `If` *k*>0 and *flag*=false
     `Then` *flag*:=true;
  `End For`
  `If` *flag* = true

[2] Comments on the existence and the uniqueness of the determining concept are given at the end of the Section.

```
    Then m:=m+1, flag = false
End For
```

Step 4. Slot mapping

Make m ($m \le n$, m is defined at Step 3) copies of the MDO query. Hereafter these copies are referred to as Billet Sub-Queries (BSQs).

```
For  i = 1,..., m
  For  each DCTG in BSQ[i]
       If k =0 (i.e. in the IRO[i] there are no concepts equivalent to the DC)
          Then  delete this DCTG from BSQ[i]
   End For
End For
```

Copies obtained will also be referred to as BSQ[i].

```
For  i = 1,..., m
  For  each remaining DCTG (see above in step 4)
       If  k>1 Then
         Make k-1 copies of BSQ[i]
           For j = 1,…,k
             For each triple in the DCTG under analysis
                 Replace slot name in the <predicate>-part by its IRDMO mapping[3]
             End For
           End For
         Else
         For  each triple in a DCTG under analysis
             Replace slot name in the <predicate>-part by its IRDMO mapping[3]
         End For
       End If
    End For
End For
```

Step 5. IRO[*m*] query clarification – making the query result a correct RDF graph

```
For  i = 1,..., m
  For each DCTG in BSQ[i]
       For each <obj>-part in DCTG
         Let x:=<obj>-part
         If  no triples with <subj>-part = x in the WHERE-section of the BSQ[i]
            Then  remove all triples with such <obj>-part
       End For
    End For
End For
```

Step 6. IRO[*m*] query clarification – forming RDQL SELECT-section

```
For  i = 1,..., m
  For  each variable in the SELECT-section of the BSQ[i]
```

[3] It is assumed in IEDI that MDO-IRO[i] slot mapppings are unique for each MDO concept.

```
        If  in WHERE-section of BSQ[i] no triples with this variable in any part
            of a triple
         Then remove the  variable from the SELECT-section of the BSQ[i]
  End For
End For
```

Step 7. IRO[*m*] Query clarification – forming RDQL AND-section

```
For  i = 1,..., m
  For  each variable in the AND-section of the BSQ[i]
       If  in WHERE-section of BSQ[i] there are no triples with this variable
           in any part of a triple
        Then remove all logical conditions comprising the variable
             from the AND-section of the BSQ[i]
  End For
End For
```

At the second step of the algorithm it is assumed that there exists the unique DC for each DCTG. The reasons are as follows. First – the DC exists for each DCTG by definition. Otherwise there will be a slot in MDO query, which is not an MDO slot. Second – the determining concept should be unique for each triples group (Section 1). In the cases when there are several concepts possessing the same slot sets (it is possible for subconcepts and superconcepts) ODSQE will take the highest concept in the concepts hierarchy, which has all the slots from the mentioned set.

ODSQE algorithm returns $k_{DC_1} * k_{DC_2} * ... * k_{DC_p}$ sub-queries for the same IRO[i], where p shows how many DC-s have mapping(-s) in the IR[i], and k_{DC_p} shows how many mappings the same DC has in the IR[i].

5 Evaluation Example

Initial proof-of-concept implementation has been done to evaluate ODSQE by the typical analytical query example from University Management Domain. Example query in natural language was formulated as follows:

> *Retrieve the list of the 1-st year students who have received maximum grade (5) in Mathematics at the University entrance examinations and have failed to pass the 1-st semester examination in any basic course in Mathematics (got unsatisfactory grade - 2). Display Student's Name, Given Name, Surname, Speciality Name, and the Name of the Course.*

It was supposed in the example that IEDI grants access to two IR-s: "University Entrants" IR implemented as MS SQL database and a "University Students" IR implemented as MS Access database. The fragments of MDO and respective IROs are given in Fig. 3. The specificity of this example is:

– Involved IRs possess only partial information with respect to the query. “University Entrants” doesn’t provide the 1-st semester examination marks. “University Students” lacks entrance examination marks.
– The semantics of “Mathematics” concept is different for the used IRs. “University Entrants” IRO says that “Mathematics” is the single subject at secondary school and is the instance of “Discipline” class. “University Students” IRO doesn’t provide the concept which has “Mathematics” as its instance. It provides courses like “Mathematical Analysis“, “Linear Algebra“, etc. as the instances of “Exams” class “Mathematics” is related to the family of University courses only in MDO as the instance of the “Discipline” class which “includes” the mentioned subjects as the other “Discipline” instances.

Initial RDQL query in terms of MDO for the given example is as follows:

```
SELECT ?firstName, ?secondName, ?lastName, ?specialityName,
?sessionExTitle
WHERE
 (?x, stud:first_name, ?firstName), (?x, stud:second_name, ?secondName),
 (?x, stud:last_name, ?lastName), (?x, stud:exams_passes, ?y),
 (?x, stud:exams_passes, ?z), (?x, stud:on_spec, ?a),
 (?y, stud:exam_title,?entrantExTitle), (?y, stud:exam_type, ?examType1),
 (?y, stud:entrant_grade, ?entrantGrade), (?y, stud:examOnDiscipline,?r1),
 (?z, stud:exam_title,?sessionExTitle), (?z, stud:exam_type, ?examType2),
 (?z, stud:session_grade, ?sessionGrade),
 (?z, stud:semesterNum,?semesterNum),
 (?z, stud:examOnDiscipline,?r2),
 (?a, stud:specialityName, ?specialityName)
 (?r1,stud:disciplineName,?entrDiscName), (?r1,stud:includes, ?i1),
 (?r2,stud:disciplineName,?sessionDiscName), (?r2,stud:includes, ?i2),
 (?i1,stud:disciplineName,?discName1),
 (?i2,stud:disciplineName,?discName2)
AND (?examType1 eq "Exam"), (?examType2 eq "Exam")
AND (?entrDiscName eq "Mathematics"), (?sessionDiscName eq "Mathematics")
AND ((?entrantExTitle eq ? discName1) || (?sessionExTitle eq ?discName2))
AND ((?sessionExTitle eq "Linear Algebra") ||
     (?sessionExTitle eq "Mathematical Analysis"))
AND (?entrantGrade eq "5")
AND (?sessionGrade eq "2")
AND (?semesterNum eq "1")
USING stud FOR <MDO-URL#>
```

The result of sub-query extraction for the example is given it Table 1.

Initial proof-of-concept evaluation showed that ODSQE worked correctly for the chosen example. Detailed evaluation results may be retrieved from [Er04].

6 Algorithm Discussion

The environment in which ODSQE is used has the following peculiarities.

Peculiarity 1. IRs’ autonomy implies that not every MDO concept/slot must have correspondences in every registered IRO. Reaching such autonomy is beyond the scope of the IEDI mediator. Instead, one of the main requirements to IEDI is the ability of the system to answer queries on a limited set of concepts which were presented in the core part of the MDO – the so called Mediator Core Ontology (MCO). IR providers are hence committed to ensure that their IROs possess the concepts/slots correspondent to MCO concepts/slots. Appearance of other concepts (with their respective slots) “mappable” to MDO\MCO is only the desired option.

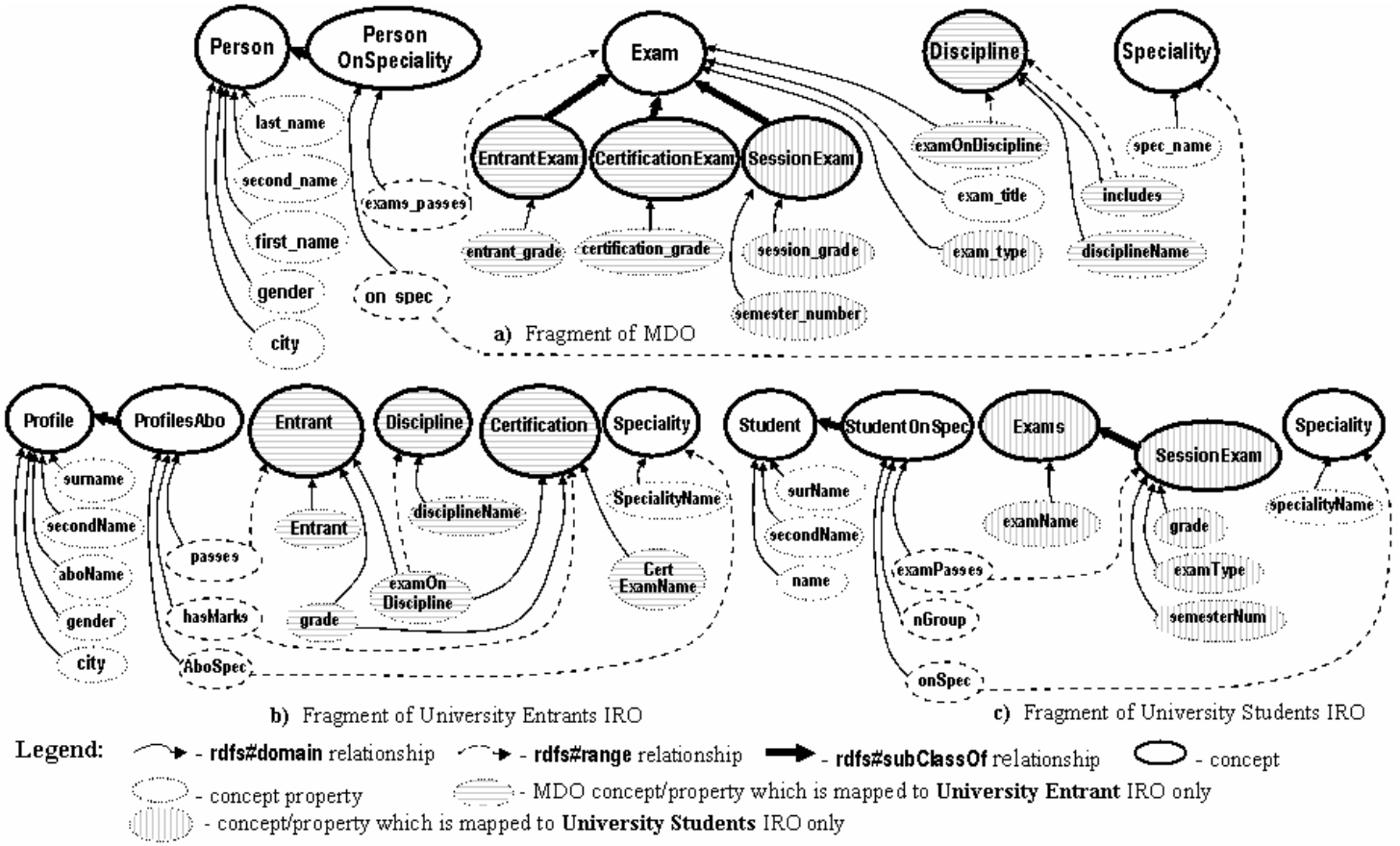

Figure 3: Fragments of MDO and IROs for evaluation example.

Results of sub-query extraction by ODSQE Table 1.

Sub-query to "University Entrants IR	Sub-query to "University Students" IR
SELECT ?*firstName*, ?*secondName*, ?*lastName*, ?*specialityName* WHERE (?*x*, abo:aboName, ?*firstName*), (?*x*, abo:secondName, ?*secondName*), (?*x*, abo:surname, ?*lastName*), (?*x*, abo:passes, ?*y*), (?*x*, abo:AboSpec, ?*a*), (?*y*, abo:EntrantExamName, ?*entrantExTitle*), (?*y*, abo:examType, ?*examType1*), (?*y*, abo:grade, ?*entrantGrade*), (?*y*, abo:examOnDiscipline,?*r1*), (?a, abo:specialityName, ?*specialityName*) (?*r1*,abo:disciplineName,?*entrDiscName*), (?*r1*,abo:includes, ?*i1*), (?*i1*,abo:disciplineName,?*discName1*), AND (?*examType1* eq "Exam") AND (?*entrDiscName* eq "Mathematics") AND ((?*entrantExTitle* eq ? *discName1*) AND (?*entrantGrade* eq "5") USING abo FOR <IRO Entrant-URL#>	SELECT ?*firstName*, ?*secondName*, ?*lastName*, ?*specialityName*, ?*sessionExTitle* WHERE (?*x*, stud:name, ?*firstName*), (?*x*, stud:secondName, ?*secondName*), (?*x*, stud:surName, ?*lastName*), (?*x*, stud:examPasses, ?*z*), (?*x*, stud:onSpec, ?*a*), (?*z*, stud:examName,?*sessionExTitle*), (?*z*, stud:examType, ?*examType2*), (?*z*, stud:grade, ?*sessionGrade*), (?*z*, stud:semesterNum,?*semesterNum*), (?a, stud:specialityName,?*specialityName*) AND (?*examType2* eq "Exam") AND ((?*sessionExTitle* eq "Linear Algebra") || (?*sessionExTitle* eq "Mathematical Analysis")) AND (?*sessionGrade* eq "2") AND (?*semesterNum* eq "1") USING stud FOR <IRO-Faculty URL#>
Which means in English:	Which means in English:
Retrieve the list of the 1-st year students who have received maximum grade (5) in Mathematics at the University entrance examinations. *Display Student's Name, Given Name, Surname, Speciality Name.*	*Retrieve the list of the 1-st year students who have failed to pass the 1-st semester examination in any basic course in Mathematics (grades 2 and 1).* *Display Student's Name, Given Name, Surname, Speciality Name, and the Name of the Course.*

Peculiarity 2. IRDMO is constructed in a way to contain the minimally necessary mappings for the MDO-IRO[i] pairs. Hence, IRDMO provides the mappings only for non-inherited (the ones defined for this very class, but not inherited from its superclass) slots of each MDO concept.

Peculiarity 3. Construction of IRDMO does not require the bijective mapping between MDO concept/slot and IRO concept/slot. Each MDO concept may have several correspondents within the same IRO. For our example (Section 5) MDO concept "Exam" has the correspondences "EntrantExam" and "CertificationExam" in "University Entrants" IRO (multiple horizontal correspondence). The same concept "Exam" has the correspondences "SessionExam" and "Exams" in "University Students" IRO (multiple vertical correspondence through concepts hierarchy). Graphical representation of the example ontologies is given in [Er04].

Mentioned peculiarities imply the usage of a kind of a Late Binding technique to find correspondences between IRO-s and MDO concepts. Late Binding here means that the decision on the choice of relevant MDO concepts used in the query is not necessarily determined by IRDMO mapping (see peculiarity 2). It is elaborated by ODSQE "lately" at run time. Of course, ODSQE uses MDO to get the knowledge on class-superclass relationships to generate proper MDO-IRO[i] concept "bindings".

Multiple concepts mapping problem may arise at ODSQE Step 3. A concept from MDO query may have several mappings to IRO[*i*] concepts. E.g., given the query "Find all titles of exams to be passed to enter the speciality "Applied Mathematics". If the user has no intention to get the titles of entrant exams only (i.e. user does not choose "EntrantExam" at the query formulation stage), then query decomposition algorithm will take the concept "Exam" as the only concept with the slot "exam_title". However

IRDMO contains two mappings for the concept “Exam” – “EntrantExam” and “SertificationExam”. The solution used in ODSQE is to get all mappings of the “problem” concept by making *k* copies of billet sub-queries at Step 4, produce all slot mappings and further to add correspondent queries to the resulting set of IRO[*i*] sub-queries.

It may be stated that ODSQE:

(i) Will build the only set of IRO[*i*] queries for a specific MDO (existence and uniqueness) and

(ii) The hypothetic result of the MDO query will be completely covered by the set of the results of produced IRO[*i*] queries (complete coverage).

The following **Statements 1** and **2** prove (i).

Statement 1. For each DCTG there exists the only DC.

Statement 2. For each DC∈ MCO there exists one or more mappings to each IRO[*i*] and for each DC∈ MDO\MCO there exist zero or more mappings to each IRO[*i*].

The proofs are based on the principles of MDO construction.

The proof of (ii) is based on the idea that in the set of produced IRO[*i*] queries there should be all requested MCO DC mappings and in at least one IRO[*i*] query per MDO\MCO DC there will be the mapping of this very DC. Hence the *recall* of the set of the results of IRO[*i*] queries will not be less than the recall of the hypothetical result of the MDO query.

7 Concluding Remarks

The paper reports on the development of UnIT-Net IEDI. The main task of IEDI is to provide a uniform framework for authorized and secure information retrieval from heterogeneous, distributed, autonomous Information Resources (IR) among the higher educational establishments and state bodies in Ukraine. The focus of the paper is the algorithm for ontology-driven sub-query extraction. This algorithm will be used by IEDI mediator to decompose the user queries in terms of the MDO. Sub-queries will then be executed by the wrappers of the respective IRs. Initial proof-of-concept evaluation and the proofs of ODSQE correctness are provided in the discussion part of the paper. The future work with respect to ODSQE is planned in the following directions: research prototype implementation as the part of IEDI mediator; prototype evaluation experiments for different IRs and IROs provided by the members of UnIT-Net consortium.

Bibliography

[AKS96] Arens, Y.; Knoblock, C.A.; Shen, W.: Query Reformulation for Dynamic Information Integration. J. of Intelligent Information Systems. 1996.

[Ba97] Bayardo et al.: InfoSleuth: Semantic Integration of Information in Open and Dynamic Environment. In Proceedings of the 1997 ACM International Conference on the Management of Data (SIGMOD), Tucson, Arizona, May 1997.

[Be98] Bergamaschi, S. et al.: An Intelligent Approach to Information Integration. In: Proc. of Formal Ontology in Information Systems (FOIS-98), June, 1998.

[CJO01] Cui, Z.; Jones, D.; O'Brien, P.: Issues in Ontology-based Information Integration. In: (A. Gomez-Perez, M. Gruninger, H. Stuckenschmidt, M. Uschold) Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing, Seattle, USA, August 4-5, 2001, 141-146.

[CJO02] Cui, Z.; Jones, D.; O'Brien, P.: Semantic B2B Integration: Issues in Ontology-based Applications. SIGMOD Record, 31(1), March 2002. 43-48

[De99] Decker, S. et al.: Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In R. Meersman et al. (eds.): Semantic Issues in Multimedia Systems. Proceedings of DS-8. Kluwer Academic Publisher, Boston, 1999, 351-369.

[Er03] Ermolayev, V. et al.: Capturing Semantics from Search Phrases: Incremental User Personification and Ontology-Driven Query Transformation. In: Proc. of the 2-nd Int. Conf. on Information Systems Technology and its Applications (ISTA'2003), Kharkiv, Ukraine, June 19-21, 2003, 9-20

[Er04] Ermolayev, V., et al.: The Infrastructure for Electronic Data Interchange. Reference Architecture Specification. Version 1.0. UNIT-NET Deliverable No D2.2.D.1. URL: http://www.compscipreprints.com/comp/Preprint/eva/20040228/1

[Ga95] Garcia-Molino, H. et. al.: The TSIMMIS Approach to Mediation: Data Models and Languages. In: Proc. Next Generation Information Technologies and Systems (NGITS), June 1995.

[Fn01] Fridman-Noy, N. et al.: Creating Semantic Web Contents with Protege-2000. IEEE Intelligent Systems 16(2):60-71, 2001.

[GMV99] Guarino, N.; Masolo, C.; Vetere, G: Content-Based Access to the Web. IEEE Intelligent Systems, May/June 1999, 70-80.

[Gr97] Gray. P. et al.: KRAFT: Knowledge Fusion From Distributed Databases and Knowledge Bases. In: Proc. 8th Intl. Workshop on Database and Expert System Applications (DEXA-97), IEEE Press, 1997, 682-691.

[LR00] Lattes V.; Rousset M.-C.: The Use of CARIN Language and Algorithms for Information Integration: The PICSEL System. Int J. of Cooperative Information Systems, 9(4), 2000, 383-401.

[Me00] Mena, E. et al.: OBSERVER: An Approach for Query Processing in Global Information Systems Based on Interoperation Across Pre-Existing Ontologies. Distributed and Parallel Databases 8(2), 2000, 223-271

[Ow03] OWL Web Ontology Language Reference. W3C Proposed Recommendation. 15 December 2003. URL: http://www.w3.org/TR/owl-ref/

[Rd04] RDQL – A Query Language for RDF. W3C Member Submission, 9 January 2004, URL: http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/

[St00] Stuckenschmidt H. et al.: Enabling technologies for interoperability. In: Visser, U., Pundt H. (Eds.): Workshop on the 14th International Symposium of Computer Science for Environmental Protection, Bonn, Germany, 2000, 35-46.

[St01] Stojanovic, N., et al.: SEAL: a Framework for Developing SEmantic PortALs. Proc. Int.Conf. on Knowledge Capture, 2001.

[Wa01] Wache, H. et al.: Ontology-Based Integration of Information - A Survey of Existing Approaches. In: (A. Gomez-Perez, M. Gruninger, H. Stuckenschmidt, M. Uschold) Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing, Seattle, USA, August 4-5, 2001, 108-118

[Wi92] Wiederhold, G.: Mediators in the Architecture of Future Information Systems. *IEEE Computer*, 25, 3 (March), 1992, 38–49.