Towards Semantic-driven, Flexible and Scalable Framework for Peering and Querying e-Catalog Communities

Boualem Benatallah [a] , Mohand-Said Hacid [b] , Hye-young Paik [a] , Christophe Rey [c] , Farouk Toumani [c]

[a] *CSE, University of New South Wales, Australia*
[b] *LIRIS, University Lyon I, France*
[c] *LIMOS, ISIMA, University Blaise Pascal, France*

Abstract

Given that e-catalogs are often autonomous and heterogeneous, effectively integrating and querying them is a delicate and time-consuming task. More importantly, the number of e-catalogs to be integrated and queried may be large and continuously changing. Conventional approaches where the development of an integrated e-catalog requires the understanding of each of the underlying catalog are inappropriate. In this paper, we use the concept of *e-catalog communities* and *peer relationships* among them to facilitate the querying of a potentially large number of dynamic e-catalogs. E-catalog communities are essentially containers of related e-catalogs. We propose a flexible and user-centric query matching algorithm that exploits both community descriptions and peer relationships to find e-catalogs that best match a user query. The user query is formulated using a description of a given community.

Key words:
E-catalogs communities, P2P and Networked Data Management, Semantic Web, Query rewriting, Description logics

Email addresses: boualem@cse.unsw.edu.au (Boualem Benatallah), mshacid@liris.univ-lyon1.fr (Mohand-Said Hacid), hpaik@cse.unsw.edu.au (Hye-young Paik), rey@isima.fr (Christophe Rey), ftoumani@isima.fr (Farouk Toumani).

1 Introduction

Web portals, such as Expedia.com and Amazon.com, are becoming more and more prominent feature of the World Wide Web. They offer tremendous opportunities for empowering users and organisations in various application domains including electronic commerce, travel, intelligence information gathering and analysis, digital government. For example, citizens and social workers will be able to get timely information and services by accessing state and federal government department Web portals [1]. Travellers will get timely information and services from relevant sources such as route information, travel operations schedule, and entertainment Web portals.

However, the technology to organize, search, integrate, and evolve these portals has not kept pace with the rapid growth of the available information space. Enjoying the above benefits using current Web technologies is a time-consuming, frustrating, and a very complex process for end users. Currently, users need to access a large number of heterogeneous and distributed data sources in order to become self-supported. For example, in digital government example, users need access to several Web portals of different government offices separately, manually filter and organise the search results to get the information about the benefits they are entitled to. This ad-hoc process of searching and combining required information sources has largely hampered a faster pace in allowing users to automate their tasks. Clearly, to effectively realize the potential of Web-based information access, there is a need for facilitating the integrated access to relevant information sources.

At present, the mainstream approach for locating web resources is through keyword-based search engines relying on information retrieval techniques. Although search engines are certainly useful, they present fundamental drawbacks such as lack of information-space organisation [2]. To effectively exploit the Web's expanding information sources, the emerging semantic web efforts employ machine-understandable abstractions for the representation of resource semantics. In particular, the semantic web promotes the use of ontologies as a tool for reconciling semantic heterogeneity between web resources. Efforts in this area include the development of ontology languages such as RDF, DAML, OWL [1] .

Despite these early efforts, many of the semantic web objectives, including the technology to organise, search, integrate, and evolve web-accessible resources, remain difficult to achieve. Ontologies provide essential support for semantics rich queries (e.g., data source vocabulary and structure aware queries). However, several critical issues must be addressed to realize the potential of the

[1] http://www.w3.org/2001/sw/WebOnt/

semantic web vision [3]. In particular, the effective use of a potentially large and dynamic collection of Web resources requires more scalable and flexible information sharing and querying techniques, which leverages and seamlessly extends current mainstream technologies and practices.

In this paper, without loss of generality, we use e-catalog portals (also called e-catalogs or e-commerce portals) as representative Web information sources to illustrate the infrastructure we propose to support information sharing and querying over the Web. We present the design and implementation of the *WS-CatalogNet* system: a Web services based information sharing middleware infrastructure whose aims is to enhance the potential of e-catalogs by focusing on scalability and flexible aspects of their sharing and access. Our research in the WS-CatalogNet builds upon lessons learned in our previous work on Web information integration [2,4], in order to provide a peer-to-peer and semantic-based e-catalogs sharing middleware infrastructure through which: (i) e-catalogs can be categorised and described using domain ontologies, (ii) heterogeneous service ontologies can be linked via peer relationships, (iii) e-catalogs selection caters for flexible and user-centric matching between user queries and e-catalogs descriptions. In a nutshell, the salient features of WS-CatalogNet are:

- **Ontology-based Indexing of e-Catalogs:** At present the main stream approach for providing unified access to multiple e-catalogs is the ad-hoc development of an integrated e-catalog by locating the e-catalogs to be integrated, understanding their interfaces, and statically linking them to the global or integrated e-catalog. This static approach is clearly tedious and hardly scalable because of the volatility and size of the Web. We use the concept of *catalog community* as a way of organizing and integrating a potentially large number of dynamic e-catalogs. A catalog community is a container of alternative e-catalogs (i.e., catalogs offering products of a common domain such as community of `Laptopsproviders` or community of `Accommodationproviders`). It provides an ontological description of desired products (e.g., product categories, product attributes) without referring to any actual provider (e.g., Dell Computers, The Hilton Hotel). Actual providers can register with any community of interest to offer the desired products. The e-catalog communities allow for meaningful organization and division of the information space.
- **Peering e-Catalog Communities:** Existing e-catalog organization techniques usually use centralized categorization and indexing schemes, whereas the participating e-catalogs are distributed and autonomous [5,6]. A centralized categorization and indexing model has several drawbacks including scalability, flexibility, and availability [7,8]. In addition, it is not realistic to assume that a global ontology that contains concepts used in all possible e-catalogs could exist. It is not realistic either to assume that there will be on agreement on the vocabularies used by different ontologies. It is

important to provide mediation facilities between heterogeneous ontologies. Given the highly dynamic and distributed nature of e-catalogs, we believe that novel techniques involving peer-to-peer centric categorization and indexing schemes will become increasingly attractive. Our approach features the use of *peer relationships* among e-catalog communities to allow decentralized sharing of e-catalogs information [9]. Query routing between communities is based on *query forwarding policies*. Query forwarding policies are high-level directives that define interaction modes among communities based on different types of peer relationships (e.g., mappings among community descriptions are fully defined, partially described, not defined). The objective is to achieve scalable information sharing and access through the collaborative data sharing and query processing among peer communities.

- **Flexible and User-centric Selection of Relevant e-Catalogs:** Because of the variety of e-catalogs offering similar information and the large number of available e-catalogs, it is important to provide appropriate support to select those e-catalogs that are relevant to a specific user query before actually querying them. In addition, e-catalog selection techniques should support flexible matching since it is unrealistic to expect queries and e-catalog descriptions to match exactly. In our approach, a user query is expressed using descriptions from a community ontology. We formalize relevant e-catalog selection as a new instance of the query rewriting problem [10,11], where a user query is reformulated in terms of:
  · a local query (i.e., the part of the user query that can be answered by some e-catalogs registered with the actual community),
  · the remaining part of the user query that cannot be answered by the actual community (i.e, cannot be answered by e-catalogs which are registered with the community and it is not possible to exactly determine which peer communities can answer it).

Our approach for e-catalog selection develops novel and more advanced query rewriring techniques for flexible and effective e-catalogs selection. Existing query rewriting approaches, developped in the context of information integration systems (see [10] for a survey), assume the existence of global mediated schema, as well as availability of the description of the underlying data source capabilities. Given a user query expressed over the global schema, the data sources that are relevant to answer the query are selected by means of a rewriting algorithm that allows to reformulate the user query into an equivalent or maximally subsumed (contained) query whose definition refers only to source descriptions. It should be noted that, an underlying assumption behind using such kind of operators (i.e., subsumption or equivalence) in a rewriting algorithm is that the source descriptions are known *a priori* (i.e., they are published in the mediator system).

The query rewriting techniques proposed in this paper go beyond subsumption and equivalence between queries and e-catalog descriptions. First, our

approach enables partial matching between e-catalog descriptions and queries, thereby allowing to cater for flexible selection of e-catalogs in environments where : (i) the existence of one centralized schema or ontology is not realistic (ii) descriptions of external e-catalogs (members of communities which are accessible via peer relationships) are not known when processing a query at a given community. In *WS-CatalogNet*, peer communities do not share the descriptions of their respective members. In addition, a community may form a peer relationship with another community without necessarily providing precise description of the mapping between its ontology and the ontology of the peer community. As a consequence, processing a user query in such a context requires the ability to select among the local members of a community those e-catalogs that are able to answer a part of the query, as well as the ability to compute the remaining part of the query that cannot be answered by the actual community (i.e, cannot be answered by e-catalogs which are registered with the community). The remaining part of a query can be forwarded to peer communities for further processing. Second, we argue that in the presence of multiple possible alternative e-catalogs with overlapping or identical functionality, users will discriminate these alternatives based on their Quality of Service (QoS). QoS is a broad concept that encompasses a number of non-functional properties such as price and reputation [12]. A QoS-aware and user-centric approach to e-catalogs selection is therefore needed.

Our main contribution in this context is the characterization of several types of flexible query rewritings and analysis of their associated computational complexity. Given a user query expressed over a community ontology:

- We consider first the problem of computing those rewritings, called *best cover rewritings*, that minimize the part of the query that cannot be answered by the community. We show that such a rewriting, as well as a best cover rewriting that minimize the number of the e-catalogs selected for answering the query (called *non redundant rewriting*), can be computed efficiently (i.e., in polynomial time).
- We show that computing all the *non redundant rewritings* can be done in incremental subexponential time in the combined size of the query and the number of the e-catalogs.
- We investigate the problem of computing the best cover rewritings that maximize the user satisfaction with respect to a given Quality of Service (QoS) criteria (e.g., communication cost, price, etc). We call such rewritings *best quality rewritings*. We show that, the general problem of computing a best quality rewriting is **NP**-hard. However in some particular cases which depend on the characteristics of the QoS function, a best quality rewriting can be computed efficiently.
- We propose a hypergraph-based algorithm to effectively compute all the best quality rewritings of a given request and we describe first experimental results that show the performance of our algorithm using up to 660

e-catalogs.

The paper proceeds as follows. Section 2 presents the design overview of *WS-CatalogNet*. Section 3 and 4 discuss the flexible query rewriting algorithms. The implementation and evaluation of *WS-CatalogNet* are explained in section 5, followed by application scenarios in section 6. Discussions on the related work are presented in section 7 and finally, section 8 concludes the paper with final remarks.

2 *WS-CatalogNet* Framework

This section presents the *WS-CatalogNet* framework, a middleware infrastructure for flexible integration and querying of heteregeneous and autonomous e-catalogs. We begin by describing the *WS-CatalogNet* metadata model and then we present the cooperative query processing approach which is used to select relevant e-catalogs in communities network.

To cope with the dynamic nature and size of the Web the *WS-CatalogNet* organizes the e-catalogs landscape in a network of *e-catalog communities* [2] [13,2]. Figure 1 shows the metamodel implemented by the *WS-CatalogNet*. A community is a container of e-catalogs of a specific domain that cater for similar customer needs (e.g. e-marketplaces for hardware, vertical portals organized on a special business topic, etc). A community is associated to an ontology that provides a description of a spefic domain of interests. E-catalogs can *register* themselves into a community as *members* by exporting (all or part of) their descriptions. Moreover, to achieve interoperability across similar or overlapping domains, communities themselves can be linked together as *peers* based on inter-ontology relationships (a.k.a. *peer relationship*)).

The exported descriptions of community members are expressed in terms of the community ontology (i.e., as views over the community ontology), thereby *WS-CatalogNet* follows a *LAV* (Local As View)[14,10] approach for integrating a community ontology with the descriptions of its members. This approach is especially useful as a given community can have a potentially large number of e-catalogs. Clearly, a GAV (Global as View) integration approach[14], where the development of a community ontology requires the understanding of both structure and semantics of all descriptions of community members, is hardly applicable in e-catalog environments because of the dynamic nature and size of the Web. Furthermore, the addition of a new member to the community requires only the provision of the exported description of the member. It does not requires any changes to the community ontology.

[2] e-Catalog communities will be referred as communities in short

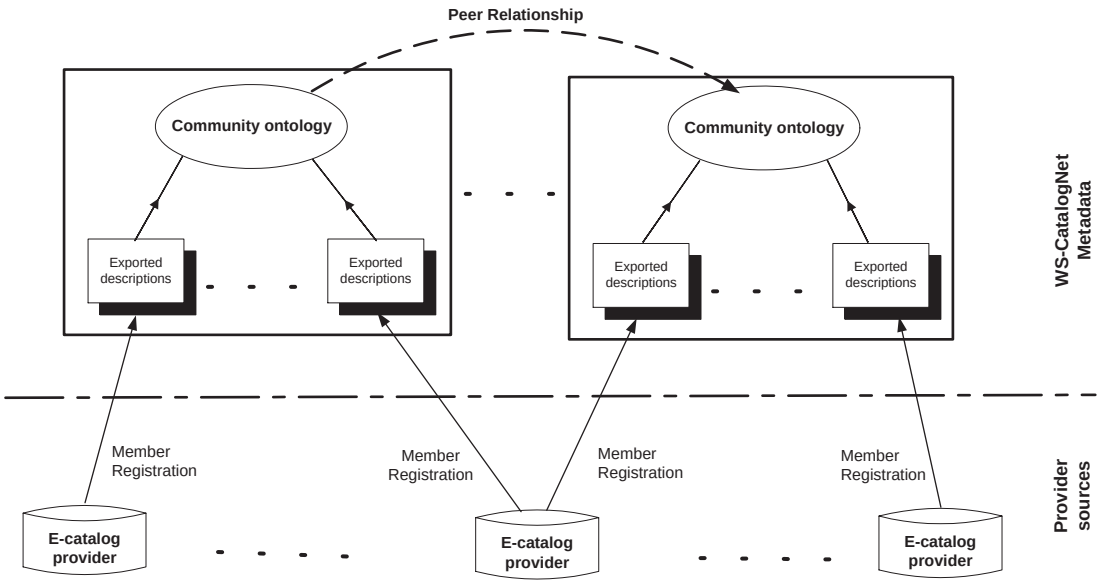

Fig. 1. Metadata representation of community networks

The second main component of the *WS-CatalogNet* framework is the cooperative e-catalogs selection approach. In *WS-CatalogNet*, a community acts as an entry point to query available e-catalogs: user queries are expressed over a community ontology and re-formulated in terms of e-catalog descriptions. As it can happen that a given (part) of a query cannot be answered by the actual members of a community, queries can also be forwarded to peer communities. The main purpose of query forwarding is to select a set of e-catalogs that, when put together, can satisfy as much as possible the requirements of the user query. For instance, if a community does not have members that provide "warranty" information, it will ask its peers for e-catalogs which do.

In the remainder of this section, we discuss how the community networks are formed, and then we introduce the cooperative e-catalogs selection.

2.1 Community Ontologies

As mentionned before, a community is a *container* of e-catalogs of a specific domain (e.g. community of `Flights` providers). Communities provide means for indexing availabe e-catalogs based on their domains and properties via ontologies. We use a simple model based on e-catalog categories (e.g., Flights, accommodation) that is powerful enough for practical applications. Indeed, our analysis of a large number of e-commerce web portals (e.g., amazon.com and expedia.com), confirms that the concept of e-catalog category is commonly used to describe portal content and capabilities. Simplicity is important as we target users who are not necessarily IT professionals. The ontology of a community is described in terms of categories for the domain it represents.

A category is described by a set of attributes. Categories within an ontology may be inter-related via the specialization (subsumption) relationship. For example, the community `FlightCenter` may have a category `Flights`, which is described using attributes such as `arrival`, `departure`, `price`, etc. (see Figure 2).

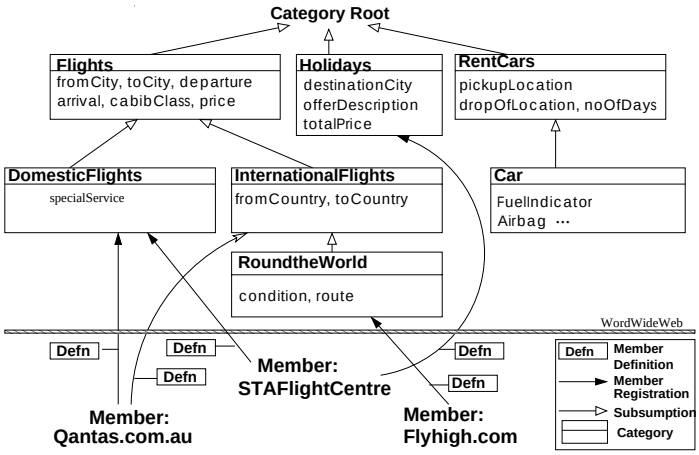

Fig. 2. Community of FlightCentre, its categories and members

To provide formal semantics, necessary for precise and rigorous characterization of queries over e-catalog communities, we propose to use a (concept) class description language that belongs to the family of description logics (DLs) [15]. The proposed language is in fact a simple description logic that is designed to represent ontology descriptions in terms of classes (unary predicates) and attributes (binary predicates). Description logics are a family of logics that were developed for modeling complex hierarchical structures and to provide a specialized reasoning engine to do inferences on these structures. This is essential to provide formal foundations for the envisioned semantic Web paradigm. Indeed, DLs have heavily influenced the development of some semantic Web ontology languages (e.g., DAML-OIL or OWL). The main constructs of the language used in this paper are illustrated below via examples.

In *WS-CatalogNet*, a community ontology is described in terms of *classes* (unary predicates) and *attributes* (binary predicates). Class descriptions are denoted by expressions formed by means of the following constructors:

- class conjunction ($\sqcap$), e.g., the description `Travel` $\sqcap$ `Accommodation` denotes the class of products which are instances of the classes `Travel` and `Accommodation` (e.g., a `Hotel`),
- the universal attribute quantification ($\forall R.C$), e.g., the description $\forall$`arrivalDate`.`Date` denotes the class of products for which all the values of the attribute `arrivalDate` are instances of the class `Date` (i.e., the data type of the attribute `arrivalDate` is `Date`),
- the existential attribute quantification ($\exists R$), e.g., the description $\exists$`Price` denotes the class of products having at least one value for the attribute

`Price`.

It should be noted that, this language is a subset of the *OWL* language. For clarity reasons, in this paper, we use the usual description logic syntax instead of the OWL syntax.

Definition 1 ***(syntax and semantics of class descriptions.)*** *Let $\mathcal{C}N$ be a set of class names and $\mathcal{A}$ be a set of attribute names. Class descriptions are inductively defined as follows:*

- *C is a class description for each class name $C \in \mathcal{C}N$.*
- *Let C, D be class descriptions and $R \in \mathcal{A}$ an attribute name. Then $C \sqcap D$, $\forall R.C$ and $\exists R$ are class descriptions as well.*

A model-theoretic semantics for this language is given by an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. It consists of a nonempty set $\Delta^{\mathcal{I}}$, called the domain of the interpretation, and an interpretation function $\cdot^{\mathcal{I}}$. The interpretation function associates to each class name $C \in \mathcal{C}N$ a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, and to each attribute name $R \in \mathcal{R}$ a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Additionally, the extension of $\cdot^{\mathcal{I}}$ to arbitrary class descriptions has to satisfy the following equations:
$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, and
$(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | \forall y : (x, y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$.
$(\exists R)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | \exists y : (x, y) \in R^{\mathcal{I}}\}$.

Based on this semantics, the notions of subsumption and equivalence between class descriptions are defined as follows. Let C and D be class descriptions:

- C is *subsumed by* D (noted $C \sqsubseteq D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all interpretation $\mathcal{I}$.
- C is *equivalent to* D (noted $C \equiv D$) iff $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all interpretation $\mathcal{I}$.

The basic concepts of the data model we use (i.e., community, category, attribute, member definition, etc) are formally defined using the class description language.

Category Definition. A *category definition* is specified as follows: `Cname` $\equiv$ `CatDescr`, where

- `Cname` is the name of the category,
- `CatDescr` is a class description that defines the category `Cname`.

For example, using the proposed language, the category `Domestic Flights` in Figure 2 can be described as follows:

`DomesticFlights` $\equiv$ `Flights` $\sqcap$ $\forall$`specialService.String` $\sqcap$ $\exists$`specialService`

This definition states that the category `Domestic Flights` inherits all the

attributes of the category `Flights`, and has one additional attribute, namely `specialService`.

We would like to emphasize that the use of a formal language is transparent to community providers and users. Indeed, *WS-CatalogNet* provides a graphical editor that supports a community providers in creating a community and defining the community ontology (see Section 6). After the definition of an ontology, the editor automatically generates the class descriptions of the ontology (i.e., class descriptions of all categories).

Using WordNet in Ontology Creation. It should be noted that, the terms used in community ontology can be different from one community to another. To help solve query mismatch problems, we use synonym-based matching approach. As part of the community ontology, each category (respectively, each attribute) is annotated with a list of synonyms. For example, for the category `Flight`, synonyms may be *air travel, air, trip*. We use WordNet [16] to assist community providers in annotating and extending community ontologies. WordNet defines a variety of semantic relationships between meanings of terms. It can be used, e.g., to derive lexical ontology relationships between terms to help the community provider annotate the ontology with the synonyms, define attributes or to add sub categories. Some of such lexical ontology relationships are:

- synonymy (similar relation): two categories (respectively, attributes) share similar meanings. Hence, one is synonyms of the other. For example, the synonymy construct of WordNet suggests that similar terms of the category `flight` are *air travel, trip and air*.
- hypernymy (super-category relation): one category has broader meaning than the other. Hence, one is super category of the other. For example, the hypernymy construct of WordNet suggests that possible super-categories of the category `flight` are *trip, journey, travel*.
- hyponymy (sub-category relation): one category has narrower meaning than the other. Hence, one is sub category of the other. For example, the hyponymy construct of WordNet suggests that possible sub-categories of the category `flight` are *domestic flight, international flight, nonstop flight*.
- meronymy (part-of relation): A category can be described by several attributes. For example, the meronymy construct of WordNet suggests that the sub category `Cars` of the category `Rentals` can be described by the attributes *air bag, sunroof, airbrake, fuel type* etc.

For the e-catalogs to be accessible through a community, they must register with the community. When registering, the e-catalog provider supplies a capability description which specifies the categories and attributes of the community ontology that are supported by the e-catalog. This form of information

is referred to as *member definition or description*. Thus, the registration process allows an e-catalog and a community to form a *MemberOf* relationship. Member definitions are also converted to class descriptions expressed using the proposed class description language and stored in the community meta-data repository (a repository used to store information about community ontology and descriptions of its members).

A *member definition* specifies the query capabilities of a given e-catalog as follows: `Mname_Dname` $\equiv$ `MDescr` where `Mname_Dname` is a member definition name made of `Mname`, a member name (i.e., an unique identifier of a member), and `Dname`, the name of a description. `MDescr` is a class description that specifies which products are actually provided by this member. For example, the e-catalog `flyhigh.com`, which offers a range of flight information, can be registered with the community `FlightCentre` using the following member definition:

```
flyhigh.com_International
≡ InternatioanlFlights
```

This definition states that `flyhigh.com` supports all attributes in the category `InternationalFlights` as well as all attributes inherited from the category `Flights` (as `InternationalFlights` is a sub-category of `Flights`). Note that, each member can provide several definitions.

When registering with a community, member providers are in charge of providing and maintaining the exported descriptions of their e-catalogs as well as the wrapping utilities required to translate user queries into e-catalog native query models.

Community Definition. A community ontology consists of a tuple $\texttt{CO} = (\texttt{C}, \texttt{S})$, where `C` is a set of category definitions and `S` is a set of community names that have peer relationships with the community. An e-catalog community consists of a tuple $\texttt{CAT} = (\texttt{CO}, \texttt{M})$, where `CAT` is the community name, `CO` is the community ontology, and `M` is a set of member definitions. We assume that the set of class descriptions in a e-catalog community (i.e., category definitions and member definitions) is acyclic, i.e., there does not exist cyclic dependencies between class definitions.

It should be noted that, in our approach, it is also possible that, a community provider (i) searches for an e-catalog which can be member of the community on the Web, (ii) generates the corresponding *WS-CatalogNet* description of the selected e-catalog, (iii) and register the e-catalog into the community. As described in Section 5, this integration mode is implemented using the Froogle search engine [17] (for searching potential e-catalogs) and our HTML2WS (a custom tool that semi-automatically generates class descriptions from a native

e-catalog description). This approach is useful, for instance, when a community provider observes that, the actual members are not answering queries related to specific categories or attributes of the community ontology.

2.2 Peering e-Catalog Communities

In *WS-CatalogNet*, a community can form links (a.k.a. *peer relationship*) with other communities. Once a link is formed, communities can forward queries to each other. This enables sharing of data that is distributed among communities. To form a peer relationship, community providers uses the community registry (hosted by *WS-CatalogNet*) to discover other communities whose domains are relevant/smilar to their communities. Typically, a provider of a community $\texttt{C}_\texttt{i}$ may decide to form a peer relationship with another community $\texttt{C}_\texttt{j}$ if $\texttt{C}_\texttt{i}$ has categories that are considered analogous, or interchangeable to $\texttt{C}_\texttt{j}$'s categories (e.g., category `Accommodation` in community `Travels` and category `Budget Hotel` in community `Hotels`).

The terms used in categories and attributes may be different from one community to another. These differences need to be resolved for the communities to collaborate in answering user queries (i.e. community `Travels` needs to know how a query described in its own ontology should be translated into a query described using the ontology of the peer community `Hotels`). Let us consider that the provider of a community (noted `Source`) forms a peer relationship with another community (noted `Target`). At this point, the provider is presented with the details of categories and attributes of the two communities. She can then defines a mapping which defines how categories (respectively, attributes) in `Source` are related to categories (respectively, attributes) in `Target`. For flexibility, we consider three types of mappings:

- **Full Mapping:** `Source` provides explicit mapping description specifying how the categories (respectively, attributes) in `Source` can be mapped to categories (respectively, attributes) in `Target`. This mapping description is stored in `Source`. Therefore, the query expressed in `Source`'s terms (i.e., `Q`) can be translated into `Target`'s terms.
- **Category Mapping:** `Source` does not specify mapping description for the attributes. In this case, the mapping only describes which category in `Source` maps to which category in `Target`. When `Source` only has mappings for categories, `Source` translate `Q` so that the category name is understood by `Target` (see `FROM` clause in the query `Q.1` given below) [3] . Then, for each attribute in `Q`, a list of synonyms is attached (as shown in `Q.1`). `Target` will use the synonyms to match the attributes.

[3] The SQL-like syntax is used here for illustration purpose. We will show in Section 7, how queries are constructed in our approach

```
(Q.1) SELECT fromCity, toCity (destination)
FROM
 Flights
WHERE
 price (retail price, listed price) < 3000
```

- **No Mapping:** `Source` does not specify any explicit mapping description with `Target`. In this case, it is left to `Target` to figure out how to answer `Source`'s queries. When there is no mapping available, synonyms for attributes (respectively for categories) are identified and attached to `Q` before forwarding (as shown in `Q.2`).

```
(Q.2) SELECT fromCity, toCity (destination)
FROM
 Flights (AirFares)
WHERE
 price (retail price, listed price) < 3000
```

  `Target` refers to the synonyms to find alternative attribute/category names to match the terms in the query with `Target`'s own terms.

Using WordNet for Ontology Mapping. WordNet is also used to assist community providers in defining mapping between ontologies of two communities. It can be used, e.g., to derive lexical ontology relationships between categories (respectively attributes) of `Source` and `Target`. For example, let us assume that the community `FlightCentre` is `Source` and the provider of this community decides to map the category `Flight` to a related category in the `Target`. WordNet would suggest terms such as *trip, travel* as hypernymy, *nonstop flights, international, domestic, connecting*, etc as hyponymy. If such categories exist in `Target`, the mapping can be described accordingly.

We are also investigating the possibility of using WordNet to automatically derive possible mapping descriptions between the ontologies of two communities. The algorithm that produces the mapping uses the WordNet constructs (e.g., synonymy, hypernymy, hyponymy) and distance metrics between terms. It takes, from `Source` and `Target`, the categories, attributes and the annotation (e.g., synonyms defined for them) as input. Then it generates *possible* mapping descriptions of `Source` ontology to `Target`. For example, the suggested mapping description might indicate that `Flight` category in `Source` is similar to `Air Travel` in `Target`.

2.3 Cooperative Selection of Relevant e-Catalogs

We now describe how the peers collaboratively participate in a query answering. The query interface in *WS-CatalogNet* lets the user easily formulate

a query (by pointing&clicking). A query is expressed in terms of categories and attributes of a given community ontology. That is, the user will click a category and then select attributes to be queried on, and specify values for the attributes if desired (e.g., "*category:domestic flights, attributes: fromCity, toCity, cabinClass, price, values: cabinClass* $\geq$ *business, price* $\leq$ *1000*"). The query interface automatically converts the user formulated query to a class description.

Since a community does not store product data locally, processing the query requires locating e-catalogs that are capable of answering the query. These e-catalogs can be selected from the local members of the community or the members of the peers. We propose a cooperative query processing technique that consists of two steps: (i) identify combination of members whose query capabilities, when put together, satisfy as much as possible the constraints expressed in the query, (ii) answer the query by sending it to the selected combination of members.

For the first step, we proposed a query rewriting algorithm, called Best Quality Rewriting (BQR), which allows to identify: (i) the part of the query can be answered by local members of the community and (ii) the part of the query that cannot by be answered by local members and hence can be forwarded to peer communities. The algorithm takes as input the community definition, member definitions and the query (all in their class descriptions format) and produces the following output:

(a) Q_{local}: the part of the query Q that *can be answered* by the community's local members. It also gives the combination of the local members that can answer all (or part of) the query. For each selected local member, we compute the part of the query to be sent to the member.
(b) Q_{rest}: the part of the query that *cannot be answered* by the local members. This part of the query will be forwarded to peers. It should be noted that the expected answers of the forwarding is the combination of the external members that are capable of answering the part of the query.

Each community defines a forwarding policy that controls what should be done with Q_{rest}. The basic structure of a forwarding policy is as follows:

```
forwarding policy:
    what theQuery| theRest
    when empty| always| expand| busy
    whom all| random m| [community...]
    hop n
```

- `what`: This is used to decide which part of the query should be forwarded.
  · *theQuery*: the original query Q is forwarded.
  · *theRest*: the Q_{rest} part of Q is forwarded.

- `when`: This is used to decide when the forwarding should be initiated
  - *empty*: the query is forwarded only when the size of Q_{local} is zero. That is, there is no member that can answer the query.
  - *expand*: the query is forwarded even when the size of Q_{local} is not zero. This is used when, for example, the community wishes to expand the number of relevant e-catalogs that can answer the query.
  - *busy*: the query is forwarded when the community is temporarily overloaded with many incoming queries. Each community has predefined value that indicates number of allowed incoming queries per time unit.
  - *always*: the query is always forwarded.
- `whom`: This is used to decide to whom the query is forwarded.
  - *all*: all similar peer communities are selected for forwarding.
  - *random m*: maximum m number of peers are randomly selected.
  - [*community...*]: only the peers that appear in the community clause are selected.
- `hop`: The hop limit is the number of subsequent forwardings that can occur once the forwarding has been initiated.

The community collects the returned results from the peers and chooses the best combination of members (local and external) based on the quality of the members (e.g., reliability) and user preferences. After all necessary members are selected, each of the selected member (both local and external) processes parts of the query they are able process, and the results are returned to the community. The community is responsible for performing the join operation on the results before displaying them to the user [4].

3 Community Query Rewriting

In this section, we discuss our approach for community query reformulation. We propose a rewriting algorithm, called BQR [5], that allows to reformulate a community query [6] Q, expressed as a class description over a community ontology, into queries that are expressed in terms of community member definitions, as well as queries that cannot be answered by the community local members. The algorithm takes as input an e-catalog community $\texttt{CAT} = (\texttt{CO}, \texttt{M})$ and a query Q over the ontology `CO` and computes a set of rewritings $R(Q) = \{r_i(Q)\}$. A rewriting r_i is a couple $r_i = (Q_{local}, Q_{rest})$ where:

[4] It should be noted that, although important, the issue of assembling actual results returned by selected catalogs, is outside the scope for this paper.
[5] The readers are referred to [9] for details and proofs related to this algorithm.
[6] We use the terms user query and community query interchangeably.

- $Q_{local} = \{(q_j, m_j)\}$ is a set of pairs (q_j, m_j) where q_j is the part of the query Q that can be answered by the member definition m_j.
- Q_{rest} is the part of the query Q that cannot be answered by the members of the actual community.

Note that a query Q can lead to several alternative rewritings. For example, the Q_{local} part of a given query Q is not necessarily unique (e.g., a same part q_j of a query Q can be provided by different members). In section 3.2, we introduce an utility function that can be used to select only those rewritings, called best quality rewritings, that maximize the user satisfaction with respect to a given Quality of Service criteria. In section 3.3, we show the computation costs of computing one or all relevant rewritings.

In a nutshell, the query rewriting problem can be formally stated as follows: Let $\mathcal{C} = \{c_i \equiv description_i,\ i \in [1, n]\}$ be a set of class definitions corresponding to member definitions, and let Q be a class definition that denotes a community query. Then, can Q be reformulated as a conjunction of class names $E \equiv c_{i_1} \sqcap \ldots \sqcap c_{i_m}$, with $1 \leq m \leq n$ and $c_{i_j} \in \mathcal{C}\ for\ 1 \leq j \leq m$, such that E contains as much as possible of *common information* with Q? (E is called a rewriting of Q using $\mathcal{C}$.)

To formally define this kind of query rewriting, it is necessary to characterize the notion of *"extra information"*, i.e., the information contained in one class description and not contained in the other. For that, a *difference or subtraction* operation on class descriptions is required. To tackle this issue, we use known techniques in description logics [18] as described below.

3.1 Difference Operation on Class Descriptions.

An extension of description logics with a difference operation is studied in [18]. Roughly speaking, the difference of two descriptions C and D, denoted as $C - D$, is defined as being a description that contains all information which is a part of the description C but not part of the description D. This definition of difference operation requires that the second operand subsumes the first one. However, in case the operands C and D are incomparable w.r.t the subsumption relationship, then the difference $C - D$ can be computed by determining the least common subsumer of C and D, that is, $C - D := C - lcs(C, D)$, where $lcs(C, D)$ denotes the least common subsumer of C and D. A least common subsumer of a set of (concept) class descriptions corresponds to the most specific description which subsumes all the given descriptions [15]. In the sequel, for a sake of clarity we use the expression $C - D$ to denote $C - lcs(C, D)$.

Teege [18] provides sufficient conditions to characterize the logics where the difference operation is always semantically unique (i.e., the result of $C - D$

is unique modulo the equivalence of descriptions) and can be implemented in a simple syntactical way by constructing the set difference of sub-terms in a conjunction. From the results presented in [18], we can derive the following properties of our class description language:

- Each class description C can be expressed using a normal form, called *Reduced Clause Form* (RCF), as a conjunction of atomic clauses (e.g., $A_1 \sqcap \ldots \sqcap A_m$). In our language, a RCF of a class description C is obtained using the following normalization process:
  · unfolding C (i.e., replacing each class name that appears in C by its description until no more defined class name occurs in C).
  · recursively rewriting each description $\forall R.(B \sqcap D)$, where B, D are class descriptions, that appear in the description C into the equivalent description $\forall R.B \sqcap \forall R.D$.

  As the set of class descriptions in a community is acyclic, the normalization process is guaranteed to terminate. At the end of the normalization process, each conjunct that appears in the normal form of the description C constitutes an atomic clause of C.
- The difference between two class descriptions C and D can be computed using the simple *set difference* operation between the sets of atomic clauses of C and D.

Moreover, we define the size $|C|$ of a class description C as being the number of clauses in its RCF. Note that, in the used language, a given class description has only one RCF.

3.2 Problem Statement.

Now let us introduce some basic definitions to formally define the query rewriting problem in the context of e-catalog communities. Let $\mathcal{C} = \{c_i \equiv description_i, i \in [1, n]\}$ be a set of class definitions corresponding to member definitions of a given e-catalog community, and Q be a query over the community ontology.

Definition 2 ***(query rewriting)*** *A rewriting of Q using $\mathcal{C}$ is a conjunction $E = c_{i_1} \sqcap \ldots \sqcap c_{i_m}$, with $1 \leq m \leq n$ and $c_{i_j} \in \mathcal{C}$ for $1 \leq j \leq m$, of some class names c_{i_j} from $\mathcal{C}$ such that:*

$Q - E \not\equiv Q$.

Hence, a rewriting of a query Q using $\mathcal{C}$ is defined as being a conjunction of class names occurring in $\mathcal{C}$ that shares some information with Q. We use the expression $rest_E(Q) = Q - E$, to denote the part of a query Q that cannot be answered by the rewriting E (i.e., the Q_{rest} part of Q, if E is selected as

relevant to answer Q). In practical situations, however, we are not interested in all kinds of rewritings. Therefore, we define additional criteria to characterize the notion of *relevant rewritings*. For example, it is clearly not interesting to consider those rewritings that do not minimize the Q_{rest} part of the query Q. That is, the part of Q that cannot be answered by the local members.

Definition 3 ***(best cover rewriting)*** *A conjunction E of some class names c_i from $\mathcal{C}$ is a best cover rewriting of Q using $\mathcal{C}$ iff:*

- *E is a rewriting of Q using $\mathcal{C}$.*
- *there does not exist a rewriting E' of Q using $\mathcal{C}$ such that: $|(rest_{E'}(Q)| < |rest_E(Q)|$.*

Best cover rewritings correspond to those rewritings that minimize the size of Q_{rest} part of a query Q. Hence, they are clearly relevant rewritings in practical situations. However, usually it may not be interesting or efficient to compute all the possible best cover rewritings. The following two definitions characterize, among the best cover rewritings, those that are more relevant in practical situations.

Definition 4 ***(non redundant rewriting)*** *A conjunction $E = c_{i_1} \sqcap \ldots \sqcap c_{i_m}$, with $1 \leq m \leq n$ and $c_{i_j} \in \mathcal{C}$ for $1 \leq j \leq m$, is a non redundant rewriting of Q using $\mathcal{C}$ iff:*

- *E is a best cover rewriting of Q using $\mathcal{C}$.*
- *$\forall j \in [1, m]$, $E' = c_{i_1} \sqcap \ldots \sqcap c_{i_{j-1}} \sqcap c_{i_{j+1}} \sqcap \ldots \sqcap c_{i_m}$ is not a best cover rewriting of Q using $\mathcal{C}$.*

Definition 4 states that it is not possible to remove any class name from the description of a *non redundant rewriting* without modifying the Q_{rest} of the query Q. In other words, the notion of non redundant rewriting characterizes those rewritings that select only the local members that minimize the size of the Q_{rest} part of the query Q. Such rewritings allow minimizing the number of e-catalogs that will be selected for providing the answer to a given community query. This may be useful, for example, to minimize the communication cost.

Finally, Definition 5 given below characterizes the notion of *best quality rewriting*, i.e., a rewriting that maximize the user satisfaction with respect to a given set of Quality of Service (QoS) criteria. We assume that there is a *scoring function*, denoted `qual(E)`, that returns a positive value which measures the quality of the rewriting E (i.e., the higher the value of `qual(E)`, the higher the quality of E). This can be, for example, a multi-attribute utility function that computes a quality score of an e-catalog based on pre-defined non-functional properties of the e-catalog (e.g., reliability, query execution time) [12]. Further discussion about the used QoS model is outside the scope of this paper due to space reasons (e.g., see [12] for more details about QoS modeling).

Definition 5 ***(best quality rewriting)*** *A conjunction E of some class names c_i from $\mathcal{C}$ is a best quality rewriting of Q using $\mathcal{C}$ iff:*

- *E is a best cover rewriting of Q using $\mathcal{C}$.*
- *there doesn't exist a rewriting E' of Q using $\mathcal{C}$ such that $qual(E) < qual(E')$.*

Best quality rewritings are defined as being the highest quality rewritings that minimize the size of Q_{rest} (as they are also best cover rewritings).

3.3 Mapping Rewritings to Hypergraph Transversals

In this section, we investigate the computational problems associated to our proposed query rewritings (e.g., computing all the *best quality rewritings* of a query Q). To achieve this task, we provide a full characterization of query rewriting in terms of hypergraph transversals. Hypergraphs and related minimal transversals generation problems have been deeply studied in recent years (e.g., see [19,20]) and have proved to be useful in a large number of applications in many areas of Computer Science, including Distributed Systems, Databases, and Artificial Intelligence [21]. Our motivation in using hypergraphs is to reuse and adapt existing techniques in hypergraph theory to solve the computational issues related to our proposed query rewritings. First, we look at the computation of each kind of rewriting (e.g., *best quality rewritings*) and determine the complexity for obtaining a solution for it. Then, in the next section, we propose a hypergraph-based algorithm for computing the best quality rewritings of a query Q using a set of class definitions $\mathcal{C}$.

Let us first recall some basic definitions regarding hypergraphs. For more details about hypergraphs theory, we refer the reader to [19,20].

Definition 6 ***(hypergraph and transversals)*** *[20]*
An hypergraph $\mathcal{H}$ is a pair (Σ, Γ) of a finite set $\Sigma = \{V_1, \ldots, V_n\}$ and a set Γ of subsets of Σ. The elements of Σ are called vertices, and the elements of Γ are called edges. A set $T \subseteq \Sigma$ is a transversal of $\mathcal{H}$ if for each $\varepsilon \in \Gamma$, $T \cap \varepsilon \neq \emptyset$. A transversal T is minimal if no proper subset T' of T is a transversal. The set of the minimal transversals of an hypergraph $\mathcal{H}$ is noted $Tr(\mathcal{H})$.

We formulate rewritings computation as a problem of finding hypergraph transversals. Given a query Q and a set of class definitions $\mathcal{C} = \{c_i \equiv description_i,\ i \in [1, n]\}$, the first step is to build an associated hypergraph $\mathcal{H}_{\mathcal{C}Q}$ as follows:

- each class c_i in $\mathcal{C}$ is associated to a vertex V_{c_i} in the hypergraph $\mathcal{H}_{\mathcal{C}Q}$. Thus $\Sigma = \{V_{c_i}, i \in [1, n]\}$.
- each clause A in the normal form description of the description of the query

Q is associated to an edge, noted w_A, in the hypergraph $\mathcal{H}_{\mathcal{C}Q}$. The edge is labeled by those classes that have in their RCFs a clause A' that is equivalent to A.

For the sake of clarity, we introduce the following notation: for any set of vertices $X = \{V_{c_l}, \ldots, V_{c_q}\} \subseteq \Sigma$, we use the expression $E_X \equiv c_l \sqcap \ldots \sqcap c_q$ to denote the class definition obtained from the conjunction of class names corresponding to the vertices in X. Inversely, for a rewriting $E \equiv c_l \sqcap \ldots \sqcap c_q$, we use the expression $X_E = \{V_{c_l}, \ldots, V_{c_q}\}$ to denote the set of vertices corresponding to the class names in E.

Using lemmas 1 and 2 given below, we show that computing a best cover rewriting of Q using $\mathcal{C}$ (i.e., a rewriting that minimizes the Q_{rest}) amounts to computing a transversal of $\mathcal{H}_{CQ}$ by considering only the non empty edges.

Lemma 1 ***(characterization of the minimal rest)*** *Let $\mathcal{H}_{CQ} = (\Sigma, \Gamma)$ be the hypergraph built from a set of class definitions $\mathcal{C}$ and a RCF of a query $Q = A_1 \sqcap \ldots \sqcap A_k$. The minimal rest (i.e., the rest whose size is minimal) of rewriting Q using $\mathcal{C}$ is: $Rest_{min} \equiv A_{j_1} \sqcap \ldots \sqcap A_{j_l}$, $\forall j_i \in [1, k] \mid w_{A_{j_i}} = \emptyset$.*

From the previous lemma, we know that the minimal rest of a query Q using $\mathcal{C}$ is always unique and is equivalent to $Rest_{min}$. Hence, a given rewriting E of Q using $\mathcal{C}$ is a best cover rewriting if and only if $rest_E(Q) \equiv Rest_{min}$.

Lemma 2 ***(characterization of best cover rewritings)*** *Let $\widehat{\mathcal{H}_{\mathcal{C}Q}} = (\Sigma, \Gamma')$ be the hypergraph built by removing the empty edges from $\mathcal{H}_{CQ}$. A rewriting $E \equiv c_{i_1} \sqcap \ldots \sqcap c_{i_m}$, where $1 \leq m \leq n$ and $c_{i_j} \in \mathcal{C}$ for $1 \leq j \leq m$, is a best cover rewriting of Q using $\mathcal{C}$ iff $X_E = \{V_{c_{i_j}}, j \in [1, m]\}$ is a transversal of $\widehat{\mathcal{H}_{\mathcal{C}Q}}$.*

This lemma characterizes the best cover rewritings in terms of hypergraph transversals. The following characterization of non redundant rewritings can be straightforwardly derived from lemma 2.

Lemma 3 ***(characterization of non redundant rewritings)*** *A rewriting $E \equiv c_{i_1} \sqcap \ldots \sqcap c_{i_m}$, with $1 \leq m \leq n$ and $c_{i_j} \in \mathcal{C}$ for $1 \leq j \leq m$, is a non redundant rewriting of Q using $\mathcal{C}$ iff $X_E = \{V_{c_{i_j}}, j \in [1, m]\}$ is a minimal transversal of $\widehat{\mathcal{H}_{\mathcal{C}Q}}$.*

Now let us consider the characterization of the best quality rewritings. Although, a detailed description of the used quality model is outside the scope of this paper, we assume l essential quality criteria (e.g., reliability, execution time) that are common to all community members:

$$q = \{q_1, q_2, \ldots, q_l\} \tag{1}$$

Therefore, we assume that for each potential member c_i, there is a *quality vector* $q(c_i)$ which represents the *goodness* of selecting this member in relation to all essential quality criteria. This vector is defined as:

$$q(c_i) = (q_1(c_i), q_2(c_i), \ldots, q_l(c_i)) \tag{2}$$

For simplicity, we assume that the value of each $q_k(c_i)$ has been scaled to [0, 1] and the higher the value is, the higher the quality is.

Finally, to be able to evaluate the quality of a given rewriting, we assume that there is a function f, called *optimization function*, that computes the quality of a set of member definitions based on their respective quality vectors. More precisely, given a rewriting $E \equiv c_{i_1} \sqcap \ldots \sqcap c_{i_m}$, the quality of E is determined as follows:

$$qual(E) = f(q(c_{i_1}), \ldots, q(c_{i_m}))$$

As before, we assume that the function f is scaled to [0, 1] and the higher the value is, the higher the quality of the rewriting is. Let us now characterize the notion of quality of a rewriting in the context of hypergraphs.

Definition 7 ***(quality of a set of vertices)*** *Let $X = \{V_{c_i}, \ldots, V_{c_j}\}$ be a set of vertices of the hypergraph $\mathcal{H}_{\mathcal{C}Q}$. We define the notion of a quality of a set of vertices as: $qual(X) = f(q(c_i), \ldots, q(c_j))$.*

Computing the best quality rewritings consists of determining, from the best cover rewritings, the ones that have the highest quality (See definition 5). In a hypergraph, this computation can be characterized as follows.

Lemma 4 ***(characterization of best quality rewritings)*** *A rewriting $E \equiv c_{i_1} \sqcap \ldots \sqcap c_{i_m}$, with $1 \leq m \leq n$ and $c_{i_j} \in \mathcal{C}$ for $1 \leq j \leq m$, is a best quality rewriting of Q using $\mathcal{C}$ iff:*

- *$X_E = \{V_{c_{i_j}}, j \in [1, m]\}$ is a transversal of $\widehat{\mathcal{H}_{\mathcal{C}Q}}$,*
- *there doesn't exist a transversal $X_{E'}$ of $\widehat{\mathcal{H}_{\mathcal{C}Q}}$ such that $qual(X_E) < qual(X_{E'})$.*

3.4 Illustrating Example

To illustrate the proposed rewriting approach, let us consider the following member definitions of the community `FlightCenter` depicted at Figure 2:

$$
\begin{aligned}
\texttt{Quantas\_DomesticFlights} &\equiv \texttt{DomesticFlights}\\
\texttt{Quantas\_IntFlights} &\equiv \texttt{InternationalFlights}\\
\texttt{Flyhigh\_Flights} &\equiv \texttt{RoundtheWorld}\\
\texttt{STA\_Flights} &\equiv \exists\texttt{fromCity} \sqcap \forall\texttt{fromCity.String} \sqcap \exists\texttt{toCity} \sqcap\\
&\quad \forall\texttt{toCity.String} \sqcap \exists\texttt{price} \sqcap \forall\texttt{price.Float}
\end{aligned}
$$

In this example, the e-catalog `Quantas` registers two member definitions, namely `Quantas_DomesticFlights` and `Quantas_IntFlights`, that respectively provide all the attributes of the categories `DomesticFlights` and `InternationalFlights` of the `FlightCenter` community. The e-catalog `Flyhigh` provides all the attributes of the category `RoundtheWorld` while the e-catalog `STA` only provides the attributes `fromCity`, `toCity` and `price` of the category `Flights`.

Let us consider now the following query which is expressed over the ontology of the community `FlightCenter`:

$$
\begin{aligned}
&\texttt{Q} \equiv \exists\ \texttt{fromCity} \sqcap \forall\ \texttt{fromCity.String} \sqcap \exists\ \texttt{toCity} \sqcap \forall\ \texttt{toCity.String}\\
&\sqcap \exists\ \texttt{price} \sqcap \forall\ \texttt{price.Float} \sqcap \exists\ \texttt{cabinClass} \sqcap \forall\ \texttt{cabinClass.String}\\
&\sqcap \exists\ \texttt{travelInsurance} \sqcap \forall\ \texttt{travelInsurance.String}
\end{aligned}
$$

The associated hypergraph $\mathcal{H}_{\mathcal{T}Q} = (\Sigma, \Gamma)$ (see Figure 3) consists of the following sets of vertices and edges:

$$
\begin{aligned}
\Sigma = \ & \{V_{Quantas\_DomesticFlights}, V_{Quantas\_IntFlights}, V_{Flyhigh\_Flights}, V_{STA\_Flights}\}\\
\Gamma = \ & \{e_1 = w_{(\exists\texttt{fromCity})}, e_2 = w_{(\forall\texttt{fromCity.String})}, e_3 = w_{(\exists\texttt{toCity})}, e_4 = w_{(\forall\texttt{toCity.String})},\\
& e_5 = w_{(\exists\texttt{price})}, e_6 = w_{(\forall\texttt{price.Float})}, e_7 = w_{(\exists\texttt{cabinClass})}, e_8 = w_{(\forall\texttt{cabinClass.String})},\\
& e_9 = w_{(\exists\texttt{travelInsurance})}, e_{10} = w_{(\forall\texttt{travelInsurance.String})}\}
\end{aligned}
$$

We can see that no member definition provides the edges $w_{\exists travelInsurance}$ and $w_{\forall travelInsurance.String}$ because $w_{\exists travelInsurance} = w_{\forall travelInsurance.String} = \emptyset$. Since these are the only empty edges in Γ, the best cover rewritings of Q using the member definitions of the `FlightCenter` community have exactly the following rest: $Q_{rest} \equiv \exists\texttt{travelInsurance} \sqcap \forall\texttt{travelInsurance.String}$. If we consider the hypergraph $\widehat{\mathcal{H}}_{\mathcal{T}Q}$, any subset Σ that contains the vertices $V_{Quantas\_DomesticFlights}$, $V_{Quantas\_IntFlights}$ or $V_{Flyhigh\_Flights}$ is a transversal of the hypergraph $\widehat{\mathcal{H}}_{\mathcal{T}Q}$ and hence constitutes a best cover rewriting of Q. The minimal transversals of $\widehat{\mathcal{H}}_{\mathcal{T}Q}$ are: $\{V_{Quantas\_DomesticFlights}\}$, $\{V_{Quantas\_IntFlights}\}$ and $\{V_{Flyhigh\_Flights}\}$. Therefore, the definitions of *Quantas* (i.e., *Quantas_DomesticFlights* and *Quantas_IntFlights*) and *Flyhigh* (i.e., *Flyhigh_Flights*) constitute three non redundant rewritings

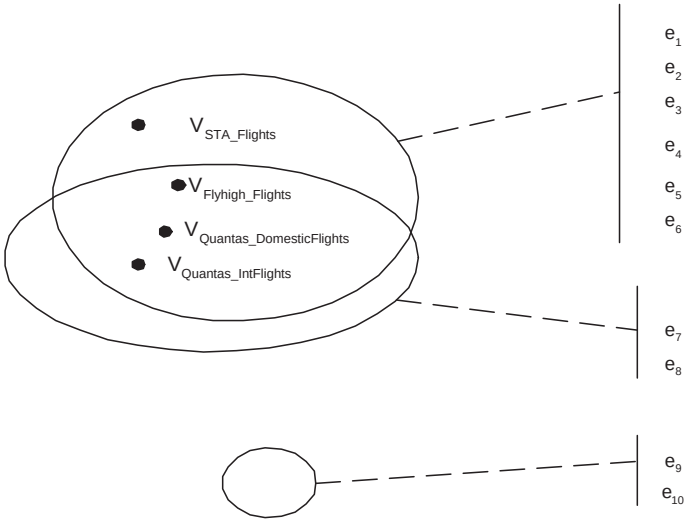

Fig. 3. Example of a hypergraph.

of the query Q. A specific quality scoring function can be used to rank these rewritings and determine the best quality rewritings of Q.

3.5 Complexity Analysis.

Based on known results in hypergraphs theory [20], we provide hereafter complexity analysis with respect to the computation of the proposed query rewritings.

Let $\mathcal{C}$ be a set of class definitions and Q be a community query. Let n be the total number of class definitions in $\mathcal{C}$ and $m = |Q|$ be the size of a query Q.

- A best cover rewriting of Q using $\mathcal{C}$ can be computed in time $O(m)$.
- A non redundant rewriting can be computed in time $O(m.n)$.
- Computing all the non redundant rewritings:
  From lemma 3, this problem amounts to computing the minimal transversals of a hypergraph. It is known that computing the minimal transversals of an hypergraph is inherently exponential since the size of the outputs (i.e., the number of the minimal transversals) is exponential in the input size [20]. However, whether there is an *output-polynomial* time algorithm (i.e., an algorithm that works in polynomial time if the number of minimal transversals is taken into account) for computing the minimal transversals of a hypergraph is still an open problem. In [22], it is shown that the generation of hypergraph transversals, and hence computation of minimal transversals, can be done in incremental subexponential time $k^{O(logk)}$, where k is the combined size of the hypergraph (i.e., the number of the vertices and the edges) and the number of minimal transversals. To the best of our knowledge, this

is the best theoretical upper time bound for computing minimal transversals of a hypergraph.

- The general problem of computing a best quality rewriting is **NP**-hard. This result is based on a polynomial reduction of the **NP**-hard problem of finding a minimal cardinality transversal of a hypergraph [20] to a particular instance of the problem of computing the best quality rewritings. However, in some particular cases, depending on the characteristics of the optimization function f, best quality rewritings can be computed efficiently. For example, in the case of additive functions, it suffices to select from each edge of the considered hypergraph the member with the highest quality score to obtain a best quality rewriting. Hence, in this case a best quality rewriting can be computed in time $O(m.n)$.

4 Computing Best Quality Rewritings

In this section, we describe an algorithm, called *BestQRC*, for computing best quality rewritings. Let $\mathcal{C} = \{c_i \equiv description_i,\ i \in [1, n]\}$ be a set of class definitions and Q be a class definition that denotes a community query. The *BestQRC* computes the best rewritings of the query Q using $\mathcal{C}$ with respect to a given optimization function f. We recall that the function f is defined on a set of quality vectors $q(c_i)$, where c_i denotes a class name. Each class name corresponds to a member definition name. When designing this algorithm, we considered an optimization function f that satisfies the following requirements [7] :

(R1) f is strictly monotonic (i.e, $Y \subset Y'$ implies that $f(Y) > f(Y')$, where Y, Y' denote sets of quality vectors). This means that the higher the number of selected sources (i.e, local members), the lower the quality of the rewriting. Consequently, in this case, the best quality rewritings should be selected among the non redundant rewritings.

(R2) f is a global optimization function in the sense that it must be performed on the whole rewriting to get its quality score (i.e., $f(Y)$ cannot be computed incrementally using a given intermediary result $f(Y')$ where $Y' \subset Y$).

These two requirements lead to a computation problem that is hard to deal with. Indeed, based on the analysis presented in the previous section and with respect to the requirement (R2), it can be shown that computing *best quality non redundant rewritings* in this case is an **NP**-hard problem. However, these two requirements also correspond to a likely realistic and desirable situations.

[7] The requirements (R1) and (R2) are inspired from a real life situation encountered during the application of our work in the context of an European project (cf. section 5).

Based on the analysis presented in the previous section, it can be shown that computing best quality non redundant rewritings can be mapped to finding the minimal transversals, with maximal quality, of the hypergraph $\widehat{\mathcal{H}}_{\mathcal{CQ}}$.

A classical algorithm for computing the minimal transversals of a hypergraph is presented in [19,20]. Using this algorithm, computing the minimal transversals with the highest quality scores consists of (i) computing all the minimal transversals, and then (ii) choosing those transversals that have the highest quality scores. The *BestQRC* algorithm presented below makes the following improvements (i.e., optimizations) with respect to the classical algorithm:

(1) it reduces the number of candidates in the intermediary steps by generating only the minimal transversals, and
(2) it uses, at the intermediary steps, a combinatorial optimization technique, namely *Branch-and-Bound* [23], in order to prune those candidate transversals which will not generate transversals with a maximal quality.

Algorithm 1 Algorithm *BestQRC* (skeleton)

Require: a query $Q = A_1 \sqcap \ldots \sqcap A_k$ provided by its RCF and an e-catalog community $CAT = (CO, M)$, with $CO = (C, S)$.
Ensure: The set $R(Q) = \{r(Q)\}$ of the best quality rewritings of Q using M.
1: Let $\mathcal{C} = M$.
2: Build the associated hypergraph $\mathcal{H}_{\mathcal{CQ}} = (\Sigma, \Gamma)$.
3: compute $Q_{rest} = A_{j_1} \sqcap \ldots \sqcap A_{j_l}$, $\forall j_i \in [1, k]$ such that $w_{A_{j_i}} = \emptyset$.
4: Build the associated hypergraph $\widehat{\mathcal{H}}_{\mathcal{CQ}} = (\Sigma, \Gamma')$.
5: $R(Q) \leftarrow \emptyset$ – Initialization of the best quality rewriting set.
6: $Tr \leftarrow \emptyset$ – Initialization of the minimal transversal set.
7: Compute a minimal transversal Y of $\widehat{\mathcal{H}}_{\mathcal{CQ}}$
8: $QualEval \leftarrow qual(Y)$. – Initialization of QualEval
9: **for all** edge $\epsilon \in \Gamma'$ **do**
10: $Tr \leftarrow$ the new generated set of the minimal transversals. – Using Theorem 1.
11: Remove from Tr the transversals whose quality is less than $QualEval$.
12: **end for**
13: **for all** $X = \{V_{c_{i_1}}, \ldots V_{c_{i_m}}\} \in Tr$ such that $qual(X) = QualEval$ **do**
14: $Q_{local} = \{(q_{i_p}, c_{i_p}),\ p \in [1, m]\}$, where $q_{i_p} = A_{j_1} \sqcap \ldots \sqcap A_{j_l}$, $\forall j_i \in [1, k]$ such that $V_{c_{i_p}} \in w_{A_{j_i}}$.
15: $R(Q) = R(Q) \cup \{r(Q) = (Q_{local}, Q_{rest})\}$
16: **end for**
17: return $R(Q)$

The first optimization generates minimal transversals at each iteration (line 10 of the algorithm). We use a necessary and sufficient condition (provided by Theorem 1 described below) to describe a pair (X_i, c_j) that will generate a non minimal transversal at iteration i, where X_i is a minimal transversal generated at iteration $i-1$ and c_j is a vertex of the i^{th} edge. Details and proofs

of Theorem 1 are presented in [9].

Theorem 1
Let $Tr(\mathcal{H}) = \{X_i, i = 1..m\}$ be the set of minimal transversals of the hypergraph $\mathcal{H}$, and $E = \{c_j, j = 1..n\}$ be an edge of $\mathcal{H}$. Assume that $\mathcal{H}' = \mathcal{H} \cup E$. Then, we have : $X_i \cup \{c_j\}$ is a non-minimal transversal of $\mathcal{H}' \Leftrightarrow$ there exists a minimal transversal X_k of $\mathcal{H}$ such that $X_k \cap E = \{c_j\}$ and $X_k \setminus \{c_j\} \subset X_i$

The second optimization consists of a *Branch-and-Bound* like enumeration of transversals. First, a simple heuristic is used to efficiently compute the quality of a *good* transversal (i.e., a transversal expected to have a high quality). The quality score is stored in the variable $QualEval$ (line 8 of the algorithm). As we consider candidates in intermediate steps, any candidate transversal that has a quality score less than $QualEval$ is eliminated from the set Tr of the candidate minimal transversals (line 11). As the optimization function f is strictly monotonic, the quality score of an eliminated candidate transversal cannot be better than that of an already computed minimal transversal.

At the end of the algorithm (lines 13 to 16), each computed minimal transversal $X \in Tr$ is translated into a rewriting $r(Q) = (Q_{local}, Q_{rest})$ which constitutes a best quality non redundant rewriting of the query Q using $\mathcal{C}$. The Q_{rest} part of a query is computed at the beginning of the algorithm (line 3). The Q_{local} part of a given rewriting is computed as follows (lines 14 and 15): for vertices $V_{c_{i_p}}$ in the transversal X, a pair (q_{i_p}, c_{i_p}) is created and added to the Q_{local} part of the actual rewriting if c_{i_p} is a member definition name. The associated query q_{i_p} consists of the conjunction of the clauses A_j of the query Q such that the corresponding hypergraph edge w_{A_j} contains the vertices $V_{c_{i_p}}$.

5 *WS-CatalogNet* Implementation

To evaluate our approach, we have implemented a prototype called *WS-CatalogNet*, which is a web service based environment for building e-catalog communities [24]. Web services [25] are emerging as promising technology for the effective automation of application integration across networks and organisations. They build upon XML, as vehicle for exchanging messages across applications, and upon the three major standards WSDL (Web Service Description Language), UDDI (Universal Description, Discovery and Integration), and SOAP, which provide the building blocks for service description, discovery, and interaction.

WS-CatalogNet has been implemented using Java and web service technologies. We used a toolkit, the IBM Web Services Development Kit 5.0 (WSDK), which has supports for the web service protocols (e.g. UDDI, WSDL and

SOAP). The toolkit also provides several components for developing Web services. In particular, we used the UDDI Java API (UDDI4J) to access a private UDDI registry (i.e. hosted by the *WS-CatalogNet*), as well as the WSDL generation tool for creating the WSDL documents and SOAP service descriptors for catalog communities and the product catalogs.

The general architecture of *WS-CatalogNet* contains three main components (see Figure 4): *Community Manager*, *Member Manager* and *Cooperative Query Manager*. These components are built on a panel of libraries and packages that the authors have either developed or integrated into WS-CatalogNet (e.g. HTML2WS: HTML to Web Service Wrapper, BQR: Best Query Rewriting algorithm and WordNet). In WS-CatalogNet, both e-catalogs and communities

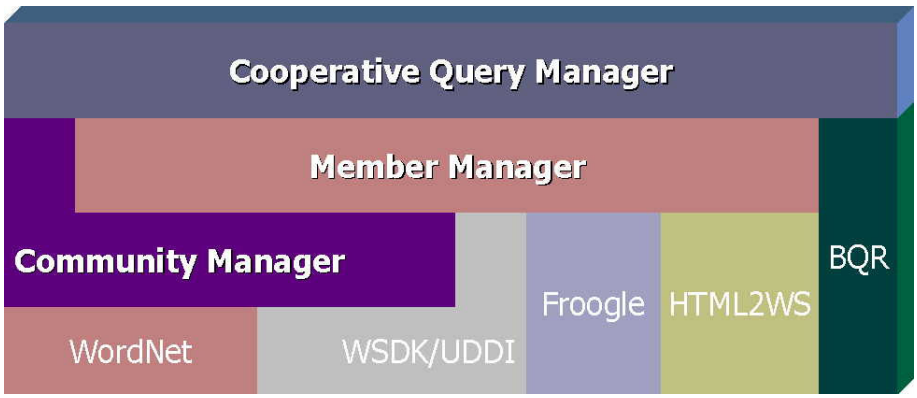

Fig. 4. *WS-CatalogNet* Architecture

are represented as web services. In UDDI registry, every web service is assigned to a tModel. A tModel provides a semantic classification of a service's functionality and a canonical description of its interface. We have designed specific tModels for e-catalogs and for communities. In Table 1, we list few of the operations in the tModels [8] .

When building a community, the community provider has to download three special classes named `QueryProcessor`, `QueryRouter` and `ResultAssembler`, which are provided by our system. The class `QueryProcessor` provides methods for processing the community queries. It implements the *BestQRC* rewriting algorithm. Community metadata (i.e., descriptions of categories, members, mapping and forwarding policies) are stored as XML documents. The class `QueryRouter` provides methods for routing queries based on the forwarding policies and the peer mappings. The class `ResultAssembler` provides methods for combining and selecting relevant e-catalogs. All classes are lightweight and the only infrastructure that they require are standard Java libraries, a JAXP-compliant XML parser, and a SOAP server.

[8] For clarity reason, we omit detailed signature of the operations.

Table 1
Main operations in tModel for members and communities

Operations for Member `M`	**Description**
Query()	Invoked to query `M`
GetInterface()	Invoked to get the categories of `M` and their descriptions
Operations for Community `C`	**Description**
Query()	To query `C`
GetInterface()	To get the categories of `C` and their descriptions
ForwardQuery()	To forward queries to other communities
AddPeer()	To add a peer community
RemovePeer()	To remove a peer community
RegisterMember()	To register a member with a community

Performance evaluation. In order to evaluate the performance of the *BestQRC* algorithm, we built a simulation testbed. In this testbed, we have implemented the two optimizations presented in section 4 as two separate options of the *BestQRC* algorithm, namely option Pers for the optimization provided by the theorem 1 and option BnB for the optimization that uses the *Branch-and-Bound* technique. We have then evaluated up to 6 versions of the *BestQRC* algorithm corresponding to different combinations of these optimization options. The simulation testbed includes a tool, based on the IBM XML Generator (http://www.alphaworks.ibm.com/tech/xmlgenerator), that generates random XML-based test community schemas, member definitions and community queries. All experiments have been performed using a PC with a Pentium III 500 MHz and 384 Mo of RAM.

We have considered three test scenarios with differences in the size of community schema, number of e-catalogs, and the size of query expressions (see Table 2).

Table 2
Configurations

Configurations	**Case 1**	**Case 2**	**Case 3**
Number of defined categories in communities	365	1334	3405
Number of e-catalogs	366	660	570
Number of (atomic) clauses in the query	6	33	12

We have run the 6 versions of the *BestQRC* algorithm on the test cases. The overall execution time results are given in Figure 5 [9]. This figure shows

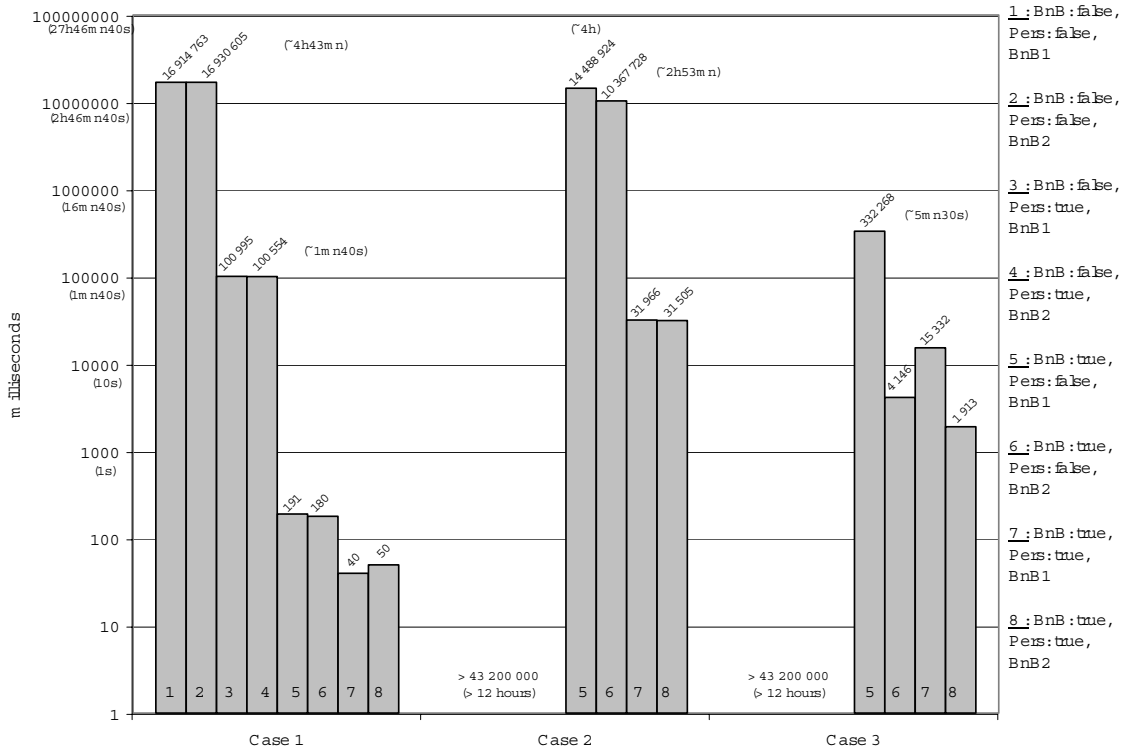

Fig. 5. Execution time

that for cases 1 and 3 (respectively, case 2), there is at least a version of the algorithm that runs in less than two seconds (respectively, in less than 30 seconds). Although Figure 5 shows that there are significant differences in performance between the different versions of the algorithm, it is worth noting that in each case, there is at least one efficient version of the algorithm even when community schema is quite large (i.e., large number of categories).

It should be noted that this preliminary experiment only concerns the performance of the rewriting algorithm. Other experiments include performance and scalability simulation studies that involve interactions (i.e., forwarding queries) in a network of communities formed via peer relationships and restructuring of this network to improve its effectiveness. These experiments are reported in [26]. We are also investigating a simulation framework for evaluating more advanced features of the proposed query rewriting techniques including QoS aspects.

[9] Note that versions 1 and 2 of the algorithm (respectively, 3 and 4) are similar as both run *BestQRC* without BnB, and what distinguishes 1 from 2 (respectively, 3 from 4) is the way the option BnB is implemented (BnB1 or BnB2).

6 Application Scenario

We have used *WS-CatalogNet* to deploy an application in tourism domain. The scenario involves a number communities and members providing various travel related information. The communities built include `FlightCenter` which provides international/domestic flight information; `TouristAttractionInfo` which provides information about tourist attractions in the world; `CarRentals` which provides rental cars information; `Accommodations` which provides accommodation information; `TravelPortals` which provides various general information related to travel. A few of the members include: `Qantas.com`, `STAFlightCentre`, `Virginblue.com.au`, `Europa.com`, etc. All of them are registered with one or two communities.

In the scenario, *WS-CatalogNet* provides the user with an integrated environment where (i) new communities can be created, (ii) e-catalog providers can discover and join communities of interests, (iii) community providers can discover other communities to form peer relationships, and (iv) users can access travel information via communities.

6.1 Developing Communities Net

We show the community `FlightCenter` as an example for creating a community. The categories and attributes of the community `FlightCenter` are shown in Figure 2. As shown in Figure 6, the community provider defines the ontology (i.e., hierarchy of the categories and their attributes) of the community using `Community Manager`. For example, the `Add Category` operation lets the user add `DomesticFlights` as sub category of `Flights`. Using the `Add Attribute` operation, the user adds attributes such as `fromCity` and `toCity` to the category `Flights`.

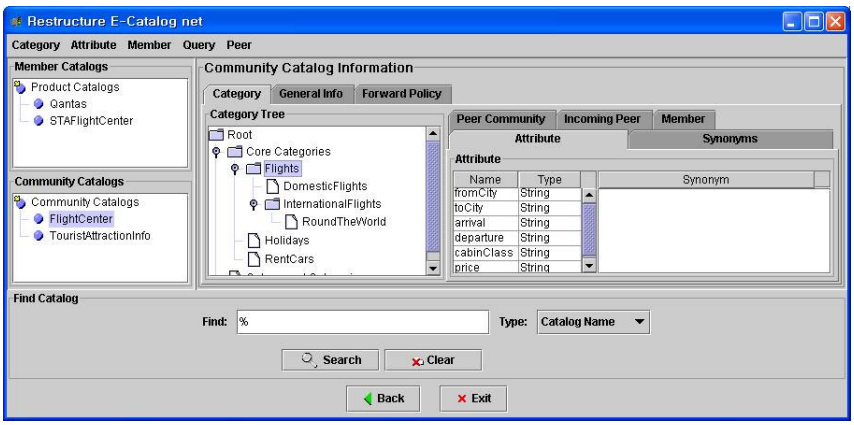

Fig. 6. Creating categories

Figure 7 illustrates how WordNet is used to assist in creating sub categories semi-automatically. After creating the category `RentCars`, the provider chooses `Add Category with` WordNet operation to create a sub category `Car` of `RentCars`. WordNet proposes five meanings of the word `Car`. For each meaning, it suggests `Car`'s sub categories, and synonyms. The provider chooses the appropriate meaning that suits his/her intention, and customises the suggested sub categories, synonyms of `Car` (e.g. by removing unwanted sub categories, or editing synonyms, etc.). Figure 7 shows WordNet suggested description for the first meaning (Meaning 1) of `Car` (e.g. `Convertible`, `Coupe`, `Limousine`, `Sedan`, etc.) and its synonyms (e.g. `auto`, `automobile`, etc.) After the category `Car` is created, the provider uses `Add Attributes with` WordNet operation to discover possible attributes of `Car`. WordNet may propose a large list of attributes, and their synonyms (e.g. *accelerator*, *airbag*, *sunroof*, etc.). The provider customises the proposed attributes list (e.g, by removing unwanted attributes, or adding synonyms, etc.) and confirms the creation of attributes. Figure 8 shows the final `Car` category description with sub categories, attributes, and synonyms. After editing the community ontology,

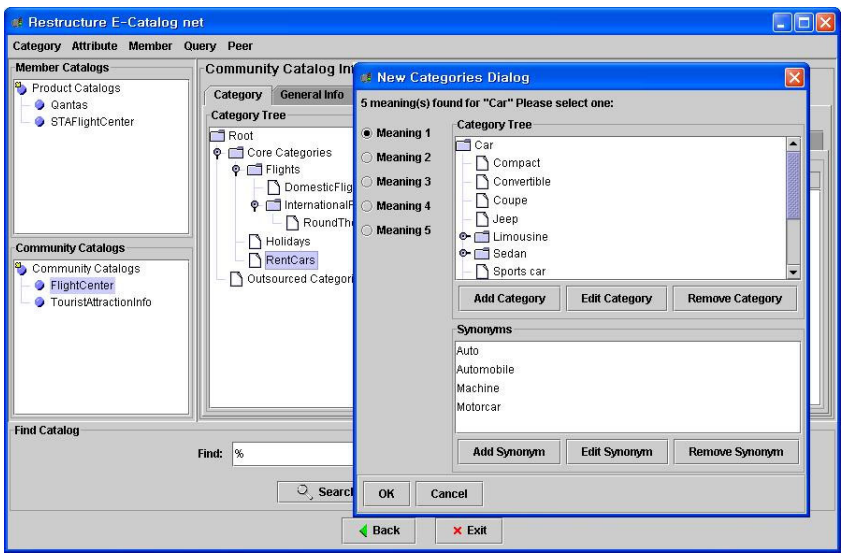

Fig. 7. Creating categories using WordNet

`Community Manager` editor generates the corresponding class descriptions.

Also, being a web service, a community in *WS-CatalogNet* implements a standard set of operations which can be invoked by the user or other communities (e.g., *addCategory()*, *addPeer()*, *queryCommunity()* etc.). The final step of community creation involves generating the WSDL which contains details of the standard operations (e.g., signature of the operations) and the community ontology (i.e., categories and attributes). The newly created community is deployed and registered in a private UDDI hosted by *WS-CatalogNet*.

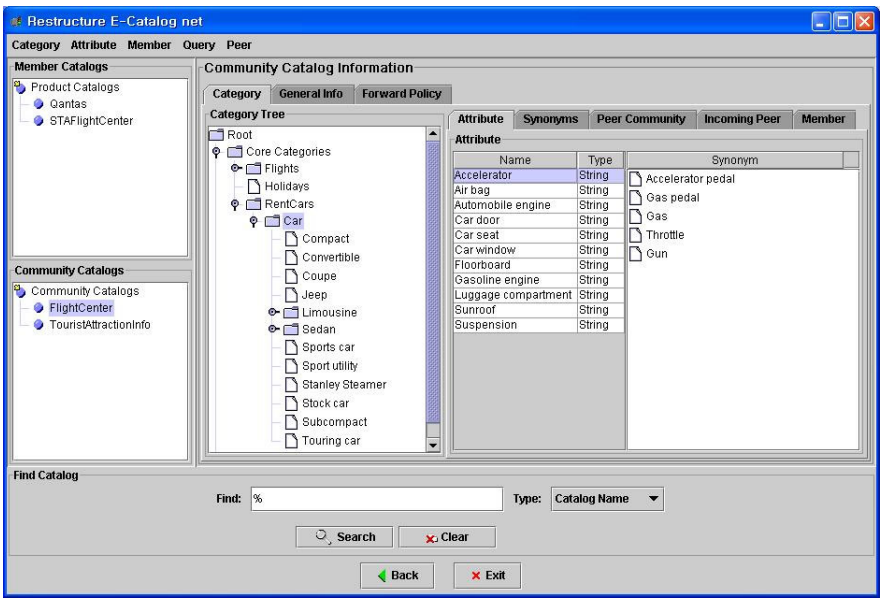

Fig. 8. Creating attributes using WordNet

6.2 Peering Communities

The provider of `FlightCenter` searches for other communities using `Community Manager`. When s/he selects the community `TouristAttractionInfo`, the `Community Manager` displays categories and attributes of the selected community. To create a peer relationship, the administrator maps the category `Holidays` in the community `FlightCenter` to the category `holidaysOffers` of `TouristAttractionInfo` community using the interface shown in Figure 9.

6.3 Registering Members

The provider of the e-catalog `STAFlightCenter` uses the `Member Manager` to display the ontology of `FlightCenter` community. To register with this community (as shown in Figure 2), the provider chooses `FlightCenter`'s categories and attributes that s/he can support. Then s/he maps these attributes to his/her own attributes (i.e., the member definitions). Figure 9 shows the member definitions of `STAFlightCenter` over the community `FlightCentre`.

Similarly to the community creation, after editing the member definitions, the `Member Manager` generates the corresponding class descriptions. For example, for the member `STAFlightCenter`, the following description is generated:

$$\texttt{STAFlightCenter\_Domestic} \equiv \texttt{DomesticFlights}$$
$$\texttt{STAFlightCenter\_Holiday} \equiv \texttt{Holidays}$$

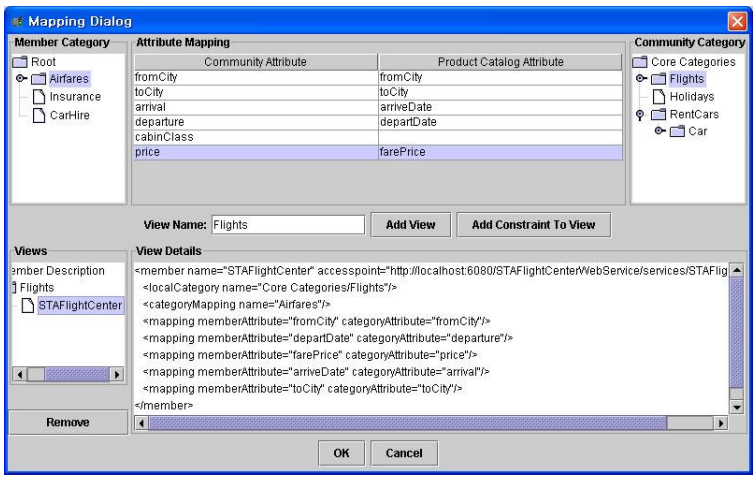

Fig. 9. Defining `STAFlightCenter` member

This definition states that this member supports all attributes in the categories `DomesticFlights` and `Holidays`.

WS-CatalogNet also provides community providers with ability to discover potential members on the Web and register them with their communities. For example, say that the provider of the `FlightCenter` community requires members for the category `InternationalFlights`. In `Member Manager`, the provider clicks on this category and uses the *Froogle*[10] search engine to discover URLs of product catalogs (e-catalog portals) that match this category. Froogle returns top ten URLs matching the keywords `international flights`. Assume that, among others, the provider chooses `Qantas.com` product catalog. Then, s/he can use HTML2WS[11] to annotate the `Qantas.com` web portal, and determine `Qantas.com` attributes (e.g. `Price`, `FlightNo`, `DepartureTime`, etc.). Figure 10 shows the first step of the wrapper in action, where the provider of community `FlightCentre` extracts the possible attributes (i.e., `price`, `departure`, `FlightNo`, `From`, `To`, etc.) of the member `Qantas.com` from its actual HTML output. Finally, HTML2WS generates a web service that will wrap `Qantas.com`. The generated `Qantas.com` web service can be registered trough the `Member manager` as described above (see Figure 9).

6.4 Querying Communities Net

As shown in Figure 11, the user displays the `FlightCenter` community and expresses, by pointing&clicking, a query, say "*category:InternationalFlights, at-*

[10] Froogle (froogle.google.com) (c.f., figure 10) is a search engine for discovering product catalog portals.
[11] A tool that semi-automatically generates a web service from a web portal.

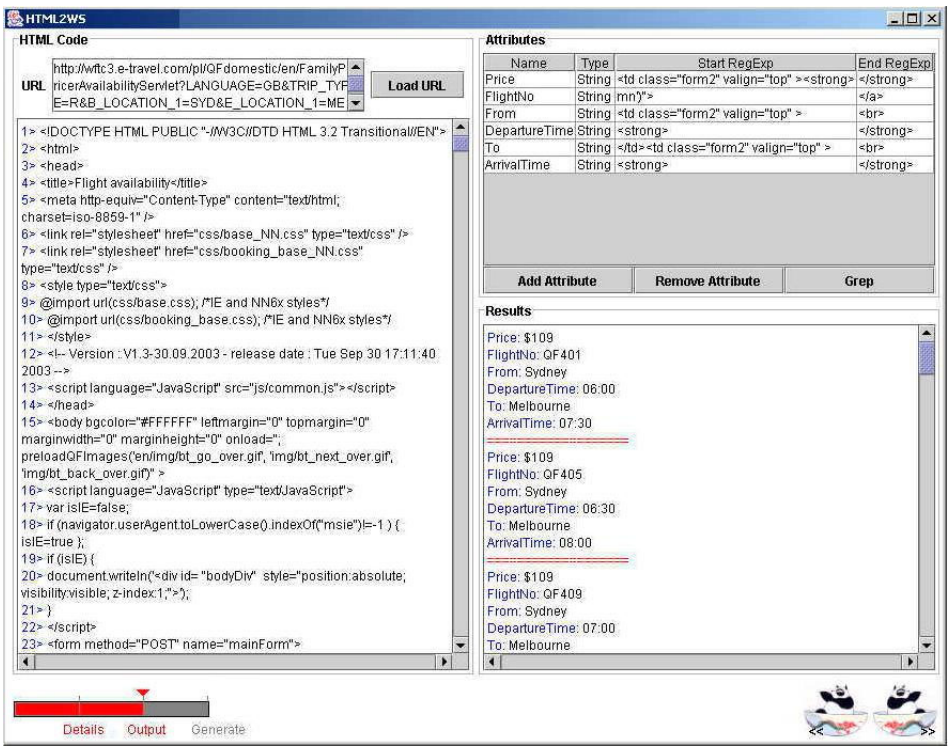

Fig. 10. The HTML2WS: HTML to Web Service Wrapper Tool

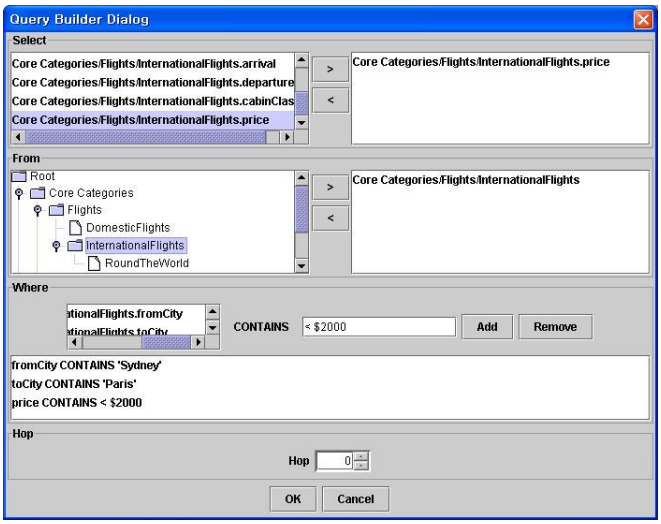

Fig. 11. Expressing a query on `FlightCenter`

tributes: fromCity=Sydney, toCity=Paris, price $\leq$ *2000, travelInsurance: full* "). The `Query Manager` generates the following class description corresponding to the user query:

$$\texttt{Q} \equiv \exists\ \texttt{fromCity} \sqcap \forall\ \texttt{fromCity.String} \sqcap \exists\ \texttt{toCity} \sqcap \forall\ \texttt{toCity.String} \sqcap \exists\ \texttt{price} \sqcap \forall\ \texttt{price.Float} \sqcap \exists\ \texttt{travelInsurance} \sqcap \forall\ \texttt{travelInsurance.String}$$

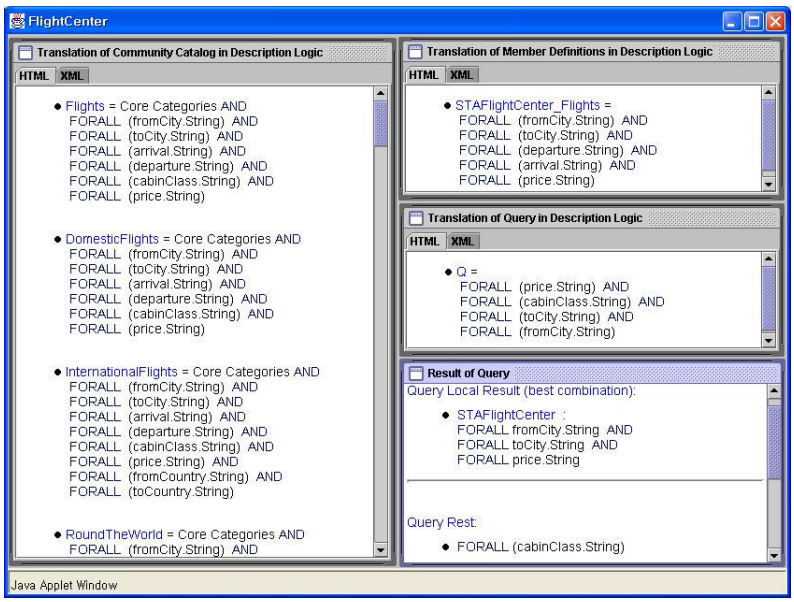

Fig. 12. Querying `FlightCenter` using BQR

From the query written in class description, the `Query Manager` uses the Best Quality Rewriting algorithm to process the query. In this case, the local member `STAFlightCenter` is selected as a relevant e-catalog (i.e., Q_{local}) to answer the user query (it provides the attributes: `fromCity, toCity` and `price`) (Figure 12). The Q_{rest} part of the query contains only the attribute (`travelInsurance`). In this scenario, a predefined forwarding policy is used to route queries among communities.

7 Related Work

In this section, we examine work done information sharing and querying across multiple sources. We also overview related efforts in Web services and semantic Web areas.

7.1 Information Sharing

Existing e-catalogs and data integration approaches typically rely on a global schema integration style [6,27,10,28]. For instance, [5] proposes an e-catalog integration approach, in which all categories of products are organised in a single graph structure and each leaf links to source catalog's product attribute tree which represent local catalog's product classification scheme. In the context of data integration, two main approaches have been investigated depending on how the relationship between the global schema and the sources is defined:

the GAV (Global As View) approach [29] defines the global schema as a set of views over the data sources while the LAV (Local As View) approach [30,31] defines the sources as a set of views over the global schema. Each of these two approaches has advantages and drawbacks. It is well known that query rewriting is easier in the context of a GAV approach while a LAV approach is more flexible (i.e., it enables easy addition of new sources). *WS-CatalogNet* follows a LAV approach to define the mappings between a community ontology and the exported descriptions of the community members (i.e., member definitions are expressed as views over the community ontology). As mentioned before, a GAV approach, where the development of an integrated schema requires the understanding of both structure and semantics of all schemas of sources to be integrated, is hardly scalable because of the potentially large number and dynamic nature of available e-catalogs.

P2P information sharing is currently a very active research and development area. From information sharing point of view, the first generation of P2P systems (e.g., Gnutella, Napster) focused on files sharing (e.g., music, video clips). An interesting survey on files sharing P2P architectures is presented in [32]. Query routing among peers in such approaches is discussed [33]. These systems support limited querying capabilities (e.g., keyword based search). Effective data sharing requires support for more structured querying capabilities to exploit the inherent semantics in data ([34,35]). [36] proposes a super-peer based routing [37] in RDF-based P2P information sources. The proposed approach focuses on indexing RDF resources.

Few approaches that leverage database-like information sharing and querying techniques in P2P environments emerged recently [38,34,39,40]. PeerDB [8] uses a relational model to describe the schema of a peer data source. Relations and attributes are associated to keywords (i.e, synonyms of relation and attribute names). PeerDB uses an Information Retrieval (IR) approach for query routing to avoid the explicit specifications of mapping among peer schemas. The issues of information space organisation and flexible query rewriting are not considered. Piazza [7] considers the issue of schema mediation in P2P environments. It uses a relational model to describe peer schemas. It proposes a language for specifying mappings among peers. It also proposes a query reformulation algorithm for the proposed mediation framework. [35] proposes the notions of mutant query plan (MQP) and multi-hierarchy namespaces to support query processing in P2P environments. A MQP includes verbatim XML encoded data, references to actual resource locations (URL) and references to abstract resource names (URN). A peer can mutate an incoming MQP by either resolving URNs to URLs, or substituting a sub-plan with the evaluated XML encoded data. [41] presents the framework that allows for unifying Web services and peer-to-peer computing technologies together. The framework is based on the idea of super-peers which act as registries of web services that have similar behaviours. The super-peers could be compared with the concept

of communities in *WS-CatalogNet*. However, different types of relationships between super-peers and specification of forwarding policies are not discussed in [41]. [41] does not consider partial matching between service descriptions and requests.

Other complementary research proposals include [34,42], which focus on generating mappings between schemas and schema change management aspects in integrated environments.

7.2 Concept and Query Rewriting

From the technical point of view, our e-catalog selection technique belongs to the general framework of *rewriting using terminologies* proposed in [11]. This framework is defined as follows:

- Given a terminology $\mathcal{T}$ (i.e., a set of class descriptions), a class description Q that does not contain class names defined in $\mathcal{T}$ and a binary relation ρ between class descriptions, can Q be rewritten into a description E, built using (some) of the names defined in $\mathcal{T}$, such that $Q \rho E$?
- Additionally, some optimality criterion is defined in order to select the relevant rewritings.

Already investigated instances of this problem are the minimal rewriting problem [11] and rewriting queries using views [31,43,10].

Minimal rewriting is concerned with the problem of rewriting a concept description Q into a shorter but *equivalent* description (hence, ρ is equivalence modulo $\mathcal{T}$ and the size of the rewriting is used as the optimality criterion). Here, the focus is on determining a rewriting that is shorter and more readable than the original description.

The problem of rewriting queries using views has been intensively investigated in the database area (see [10] for a survey). The purpose here is to rewrite a query Q into a query expression that uses only a set of views $\mathcal{V}$. Two types of rewritings have been studied:

- Maximally-contained rewritings where ρ is the subsumption and the optimality criterion is the inverse subsumption. This kind of rewriting plays an important role in many applications such as information integration and data warehousing.
- Equivalent rewriting where ρ is the equivalence and the optimality criterion is minimization of the cost of the corresponding query plan. This kind of rewriting has been mainly used for query optimization purposes.

The e-catalog selection technique proposed in this paper can be viewed as a new instance of the problem of rewriting concepts using terminologies where:

- ρ correponds to the notion of cover (hence, it is neither equivalence nor subsumption), and
- the optimality criterion is a quality function.

In our approach, the proposed selection technique finds rewritings that 'best match' a given query with respect to a quality function, where the relationships between a query and its rewritings goes beyond containment or equivalence. We believe that there is a need for such a flexible query rewriting approach to cope with the high dynamicity and heterogeneity of the Web-based environments.

7.3 Web Services and Semantics Web

Current mainstream web services infrastructure has serious limitations with respect to meeting the challenges of automation of Web information sharing. For example, UDDI provides limited search facilities, supporting only keyword-based search of businesses, services, category names, and identifiers (i.e. so-called tModels). To cope with this limitation, emerging approaches rely on semantic web technology, and in particular web ontology languages such as DAML-OIL or OWL [44,45], to support service description and discovery [46]. Unfortunately, ongoing research in this area focuses on matchmaking techniques based on simple subsumption and equivalence relationships. Our approach, on the other hand, builds upon existing ontology description languages to develop novel and more advanced techniques including: (i) ontology and peer-to-peer indexing schemes imparting domain-based and scalable organisation of a potentially large number of e-catalogs, (ii) advanced matching techniques which go beyond subsumption and equivalence between queries and e-catalog descriptions and allow flexible and efficient selection of e-catalogs through partial matching.

8 Conclusions

Our work provides complementary contributions to related work on Web information sharing and querying. We focus on providing support for achieving effective and efficient access to e-catalogs resident data. Since scalability is of great importance in e-catalog environments, the information space is organised in communities that are inter-related using peer relationships. The model that we use to describe community ontologies relies on simple concepts (categories

and attributes) that we found to be useful and commonly used to describe e-catalogs content and capabilities. We consider different types of peer relationships between communities (i.e, companionship and similarity relationships). Such flexibility allows to establish different interaction types among communities. The specification of mappings in our approach combines: (i) IR style where no explicit mapping description is provided (synonym-based matching approach) and (ii) full mapping description when it is possible or desired (e.g., in the case of companionship relationships). We formalised e-catalogs selection as a rewriting process for e-catalog communities. Query routing among peer communities is based on forwarding policies. We proposed a novel hypergraph-based algorithm to effectively select relevant e-catalogs for a given query. The proposed algorithm finds rewritings that 'best match' a given query with respect to a quality function, where the relationships between a query and its rewritings goes beyond containment or equivalence. We believe that there is a need for such a flexible query rewriting approach to cope with the high dynamicity and heterogeneity of the Web-based environments.

Our ongoing work concerns developing and evaluating self-monitoring and adaptation techniques. We believe that in large and dynamic environments, e-catalog communities should be self-adaptive in the sense that, mappings among their ontologies should be restructured over time to improve the effectiveness of the overall communities network. This will enable e-catalogs networks to be adaptive and responsive to the ways users interact with the e-catalogs and communities thereby increasing their usability. Another interesting future research direction of our is to study load balancing among services communities in order to improve the performance of query answering in a communities network.

References

[1] B. Medjahed, A. Rezgui, A. Bouguettaya, M. Ouzzani, Infrastructure for E-Government Web Services, IEEE Internet Computing 7 (1) (2003) 58–65.

[2] A. Bouguettaya, B. Benatallah, L. Hendra, M. Ouzzani, J. Beard, Supporting Dynamic Interactions among Web-based Information Sources, IEEE TKDE 12 (5) (2000) 779–801.

[3] D. Beneventano, S. Bergamaschi, F. Guerra, M. Vincini, Synthesizing an Integrated Ontology, IEEE Internet Computing 7 (5) (2003) 42–51.

[4] H. Paik, B. Benatallah, Building Adaptive E-Catalogs Based on User Interaction Patterns, IEEE Intelligent Systems, IEEE Society.

[5] S. Navathe, H. Thomas, M. S. A., A. Datta, A Model to Support E-Catalog Integration, in: IFIP Conference on Database Semantics, Hong Kong, 2001.

[6] G. Yan, W. Ng, E. Lim, Product Schema Integration for Electronic Commerce– A Synonym Comparison Approach, IEEE TKDE 14 (3).

[7] A. Halevy, Z. Ives, D. Suciu, I. Tatarinov, Schema Mediation in Peer Data Management Systems, in: ICDE'03, Bangalore, India, 2003.

[8] W. Ng, B. Ooi, K. Tan, A. Zhou, PeerDB: A P2P-based System for Distributed Data Sharing, in: ICDE'03, Bangalore, India, 2003.

[9] B. Benatallah, M.-S. Hacid, H.-Y. Paik, C. Rey, F. Toumani, Peering and Querying e-Catalog Communities (extended version), ftp://ftp.cse.unsw.edu.au/pub/doc/papers/UNSW/0319.pdf (2003).

[10] A. Y. Halevy, Answering queries using views: A survey, VLDB Journal 10 (4) (2001) 270–294.

[11] F. Baader, R. Küsters, R. Molitor, Rewriting Concepts Using Terminologies, in: KR'00, Colorado, USA, 2000, pp. 297–308.

[12] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, Q. Sheng, Quality-driven Web Service Composition, in: WWW'03, Budapest, Hungary, 2003.

[13] H. Paik, B. Benatallah, R. Hamadi, Dynamic Restructuring of E-Catalog Communities Based on User Interaction Patterns, WWW Journal 5 (4) (2002) 325–366.

[14] J. D. Ullman, Information integration using logical views, in: F. N. Afrati, P. G. Kolaitis (Eds.), Database Theory - ICDT '97, 6th International Conference, Delphi, Greece, January 8-10, 1997, Proceedings, Springer, 1997, pp. 19–40.

[15] F. Baader, D. Calvanese, D. McGuinness, e. D. Nardi andP. Patel-Schneider, The Description Logic Handbook. Theory, Implementation and Applications, Cambridge University Press, 2003.

[16] Wordnet, www.cogsci.princeton.edu/∼wn/.

[17] Froogle, froogle.google.com.

[18] G. Teege, Making the difference: A subtraction operation for description logics, in: KR'94, San Francisco, CA, 1994.

[19] C. Berge, Hypergraphs, Vol. 45 of North Holland Mathematical Library, Elsevier Science Publishers B.V., 1989.

[20] T. Eiter, G. Gottlob, Identifying the minimal transversals of a hypergraph and related problems, SIAM Journal on Computing 24 (6) (1995) 1278–1304.

[21] G. G. T. Eiter, Hypergraph Transversal Computation and Related Problems in Logic and AI, in: S. Flesca, S. Greco, N. Leone, G. Ianni (Eds.), JELIA 2002, Cosenza, Italy,, Vol. 2424 of LNCS, Springer, 2002, pp. 549–564.

[22] M. Freidman, L. Khachiyan, On the complexity of dualization of monotone disjunctive normal forms, Journal of Algorithms 21 (1996) 618–628.

[23] G. Nemhauser, L. Wolsey, Integer and Combinatorial Optimization, John Wiley&Sons (New York), 1988.

[24] K. Baina, B. Benatallah, H. Paik, F. Toumani, C. Rey, A. Rutkowska, H. Susanto, WS-CatalogNet: An Infrastructure for Creating, Peering, and Querying e-Catalog Communities, in: Proc. of 30th International Conference on Very Large Data Bases (VLDB 2004), Toronto, Canada, 2004, demonstration paper, to appear.

[25] F. Casati, M.-C. Shan, D. G. (editors), The VLDB Journal: Special Issue on E-Services, 10(1), Springer-Verlag Berlin Heidelberg (2001).

[26] H. young Paik, Community Based Integration and Adaptation of Electronic Catalogs, PHD Thesis, University of New South Wales, Sydney, Australia (March 2004).

[27] J. Ullman, Information integration using logical views, Theor. Comput. Sci. 239 (2) (2000) 189–210.

[28] M. Lenzerini, Data Integration: A Theoretical Perspective, in: L. Popa (Ed.), PODS'02, Madison, Wisconsin, USA, 2002, pp. 233–246.

[29] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos, J. Widom, The tsimmis approach to mediation: Data models and languages, Journal of Intelligent Information Systems (JIIS) 8 (2) (1997) 117–132.

[30] A.Y.Levy, A. .Rajaraman, J. Ordille, Querying Heterogeneous Information Sources Using Source Descriptions, in: VLDB'96, Mumbai (Bombay), India, Morgan Kaufmann, 1996, pp. 251–262.

[31] C. Beeri, A. Levy, M.-C. Rousset, Rewriting Queries Using Views in Description Logics, in: L. Yuan (Ed.), PODS'97, New York, USA, 1997, pp. 99–108.

[32] B. Yang, H. Garcia-Molina, Comparing Hybrid Peer-to-Peer Systems, in: VLDB'01, Rome, Italy, 2001.

[33] A. Crespo, H. Garcia-Molina, Routing Indices For Peer-to-Peer Systems, in: ICDCS'02, Vienna, Austria, 2002.

[34] P. Bernstein, F. Giunchigiloa, A. Kementsietsidis, J. Mylopoulos, L. Serafini, I. Zaihrayeu, Data Management for Peer-to-Peer Computing: A Vision, in: WebDB'02, Madison, Wisconsin, 2002.

[35] V. Papadimos, D. Maier, K. Tufte, Distributed Query Processing and Catalogs for Peer-to-Peer Systems, in: CIDR'03, Asilomar, CA, 2003.

[36] W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, A. Lser, Super-Peer-Based Routing and Clustering Strategies for RDF- Based Peer-To-Peer Networks, in: WWW'03, Budapest, Hungary, 2003.

[37] B. Yang, H. Garcia-Molina, Designing a Super-Peer Network, in: ICDE'03, Bangalore, India, 2003.

[38] K. Aberer, P. Cudre-Mauroux, M. Hauswirth, The Chatty Web: Emergent Semantics Through Gossiping, in: WWW'03, Budapest, Hungary, 2003.

[39] D. Calvanese, G. D. Giacomo, M. Lenzerini, R. Rosati, Logical foundations of peer-to-peer data integration, in: PODS'04 (to appear), 2004.

[40] F. Goasdoué, M.-C. R. V. Lattès, Querying Distributed Data through Distributed Ontologies: A Simple but Scalable Approach, IEEE Intelligent Systems 18 (5) (2003) 60–65.

[41] M. Papazoglou, B. Kramer, J. Yang, Leveraging Web-Services and Peer-to-Peer Networks, in: CAiSE'03, Klagenfurt, Austria, 2003.

[42] A. Kementisietsidis, M. Arenas, R. Miller, Mapping Data in Peer-to-Peer Systems: Semantics and Algorithmic Issues, in: ACM SIGMOD'03, San Diego, CA, 2003.

[43] F. Goasdoué, M.-C. R. V. Lattès, The Use of CARIN Language and Algorithms for Information Integration: The PICSEL System, IJICIS 9 (4) (2000) 383–401.

[44] D. Fensel, C. Bussler, and A. Maedche. Semantic Web Enabled Web Services. In International Semantic Web Conference, Sardinia, Italy, pages 1-2, Jun. 2002.

[45] Ian Horrocks. DAML+OIL: A Reasonable Web Ontology Language. In Proc. of the EDBT'2002 Prague, Czech Republic, pages 2-13, Mar. 2002.

[46] M. Paolucci, T. Kawmura, T. Payne, K. Sycara, Semantic matching of web services capabilities, in: Proceedings of the First International Semantic Web Conference, 2002.