

WS-CatalogNet: An Infrastructure for Creating, Peering, and Querying e-Catalog Communities

Karim Baïna¹, Boualem Benatallah¹, Hye-young Paik^{1*}, Farouk Toumani², Christophe Rey², Agnieszka Rutkowska¹, and Bryan Harianto¹

¹CSE, University of New South Wales, Sydney, Australia,

²Laboratoire LIMOS, ISIMA, Clermont-Ferrand, France

{kbaina,boualem,hpaik,agr225,bryanh}@cse.unsw.edu.au, {ftoumani,rey}@isima.fr

1 Introduction

E-catalog portals, such as Expedia.com and Amazon.com, are becoming more and more prominent feature of the Web. They aim to offer one-stop shopping experience for the users. However, the users still need to access a number of portals separately, or to use search engines in order to get complete information they are looking for. It is clearly useful to provide a unified interface to access multiple e-catalog portals. The issue here is that the technology to create, organise, integrate and search these portals has not kept pace with the rapid growth of the available information space. Most existing approaches for providing access to integrated e-catalogs as a portal are based on (i) creating centralised product data repository collected from participating e-catalog providers, (ii) statically linking manually (ad-hoc) identified e-catalogs to the portal. Surely, these are not scalable approaches. First, we cannot expect the integrators to understand underlying schemas of thousands of online e-catalogs and produce a single schema. Second, considering the dynamic nature of the Web, the underlying schema of any online e-catalog may change any time and change frequently, which has to be reflected to the central schema.

We have developed WS-CatalogNet to address the two issues: (i) scalable integration of e-catalogs, and (ii) flexible query processing technique whereby a portal collaborate with other portals to locate information

*The author is now at the School of Information Systems, Queensland University of Technology, Brisbane Australia, h.paik@qut.edu.au

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, requires a fee and/or special permission from the Endowment.

Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004

which is not locally available.

More precisely, in WS-CatalogNet, e-catalogs catering for similar customer needs are grouped together into *e-catalog communities* (communities in short) [2]. Individual e-catalogs *register* themselves to a community as *members*. Also, communities are linked together as *peers* based on inter-ontology relationships (e.g. similarity). Query routing among communities occurs to identify a set of e-catalogs that, when put together, can satisfy all constraints specified in the user query.

WS-CatalogNet is a hybrid of peer-to-peer and web-services technologies. Given the highly dynamic and distributed nature of e-catalogs, a novel approach that involves techniques such as peer-to-peer and Web services will become increasingly attractive in building e-catalog portals.

WS-CatalogNet consists of a set of integrated tools that allow for creating communities, registering e-catalog members, creating peer relationships between communities, querying individual communities and routing queries among communities. It has been implemented using Java and the IBM Web Services Development Kit 5.0 (WSDK) which 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 e-catalog communities and the product e-catalogs.

2 WS-CatalogNet Design Overview

The general architecture of WS-CatalogNet contains three main components (see Figure 1): *Community Manager*, *Member Manager* and *Cooperative Query Manager*. These components are built on a panel of libraries and packages which 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).



Figure 1: WS-CatalogNet Architecture

2.1 Community Manager

The **Community Manager** is used to create communities and build peer relationships between communities.

Creating Communities. A community is a *container* of e-catalogs of the same domain (e.g. community of **Flights**). It provides a description of desired products without referring to actual product providers (e.g. **Qantas Airlines**). The schema of a community is described in terms of categories and descriptive attributes. For example, the community **FlightCenter** may have a category **Flights**, which is described using attributes such as **arrival**, **departure**, **price**, etc. In fact, a community schema can be viewed as an ontology (integrated schema) of its underlying e-catalogs.

To provide formal semantics in describing categories and attributes, necessary for precise characterisation of queries over the e-catalogs, we use a class description language that belongs to the family of description logics¹. The users do not describe schemas and queries directly using description logic. Instead, a graphical interface is used to automatically generate these descriptions.

A community implements a common interface of service operations which are invoked by end users or peers (e.g. *addCategory()*, *addPeer()* and *queryCommunity()*). After the community schema is defined, the community manager generates the WSDL which contains details of the interface (e.g. signature of the operations) and description of the schema. The newly created community is deployed and registered to a service directory hosted by WS-CatalogNet.

Building Peer Relationships. Communities may create a peer relationship with each other. When building a relationship, any mismatches of the terms used in categories and attributes between communities need to be resolved. Let us consider that the administrator of a community (noted **Source**) forms a relationship with another community (noted **Target**). The administrator then defines a mapping description which states how the categories (respectively the attributes) in **Source** are mapped to **Target**. For flexibility, we consider three types of mappings: (i) explicit mapping between categories (respectively, attributes) of the corresponding communities (*full mapping*), (ii) explicit mapping between only categories of the corresponding communities (*category mapping*), (iii) no ex-

¹<http://dl.kr.org/>

licit mapping between the corresponding community schemas. When the translation of a query between peers is needed, and explicit mapping description is not available, the **Source** and **Target** use synonyms² to solve mismatches between the corresponding community schemas.

2.2 Member Manager

The **Member Manager** supports registering individual e-catalogs into communities. The e-catalogs are either already web services, or converted to web services through the member manager.

When registering an e-catalog, the e-catalog provider first indicates which categories, in the community schema, the e-catalog belongs to (e.g., **Qantas Airlines** may belong to the category **international flights** of the community **Flights**). Then, for each category chosen, the provider specifies what kind of attributes are supported for the selected categories (called *member definition*). Similarly to describing mappings between peer communities, the e-catalog provider must define mappings between the community and e-catalog descriptions.

2.3 Cooperative Query Manager

Since a community does not store product data locally, processing the query requires locating e-catalogs that are capable of answering the query. We propose a cooperative query processing technique that consists of two steps: (i) identify best combinations of members whose query capabilities, when put together, satisfy the constraints expressed in the query, (ii) resolve the query by sending it to the selected combination of members. The name “cooperative” comes from the fact that the first step involves communities forwarding queries to each other to find the members who can resolve the query.

We adapted a query rewriting algorithm developed by the authors [1] – Best Query Rewriting (BQR). This algorithm identifies which part of the query can be answered by local members of the community and which part of the query cannot (hence, needs help of peers). The algorithm takes as input the community schema, member definitions and the query (all expressed in the class description language) then produces the following output:

- (a) Q_{local} : the part of the query Q that *can be answered* by the community’s local members. It gives the best combinations of the local members that can answer all (or part of) the query.
- (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 peer communities. It is

²Synonyms are defined for each categories and attributes in a community.

noted that the expected answers of the forwarding is the combination of the external members (i.e. members of peer communities) that are capable of answering the part of the query.

Each community has a *query forwarding policy* which controls what should be done with Q_{rest} . The forwarding policy can express (i) when the query should be forwarded (e.g. when no local members can answer, when the community is too busy, etc.) (ii) to which peer (e.g. all, top K , random, etc.) the query should be forwarded, and (iii) how far the query should be forwarded (i.e., hop limit).

After forwarding, the community collects the returned results from the peers and chooses the best combination of e-catalog 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 processes parts of the query that it is capable of processing, and the results are returned to the community.

3 Demonstration Scenario

We have used WS-CatalogNet to deploy an application in the tourism domain. It covers the following 6 communities: **FlightCenter** - international/domestic flight information, **TravelInfo** - flights and accommodation information, **CarRentals** - rent cars information, **TouristAttractionInfos** - tourist attractions in the world, **Accommodations** - hotels, youth hostels, B&B information, **TravelPortals** - travel tips, special deals information.

In the demonstration, we first create a community **FlightCenter** using **Community Manager**. We will show how **FlightCenter** builds a peer relationship with another community **TravelInfo**. Then we register two e-catalog members **STAFlightCenter** and **Qantas.com** through **Member Manager**. Here, we will demonstrate that we support two kinds of e-catalogs: one that is already a web service, the other that is converted to a web service through our custom developed tool. Finally, we show how the community **FlightCenter** is queried and Q_{rest} part of a query is forwarded to its peer.

3.1 Creating Community FlightCenter

Figure 2 shows how the administrator of the community **FlightCenter** creates the hierarchy of the categories using the **Community Manager**. For example, the **Add Category** operation lets the user add a category **DomesticFlights** as sub category of **Flights**. Using the **Add Attribute** operation, the user adds attributes such as **fromCity** and **toCity** to the category **Flights**.

Using WordNet. To demonstrate how WordNet assists the user in defining the community schema,

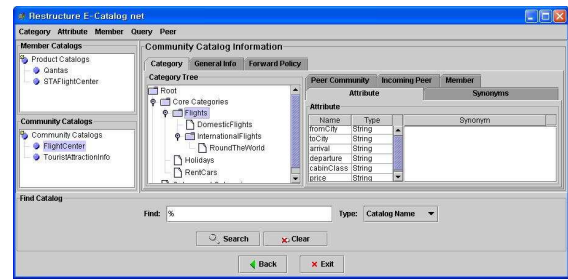


Figure 2: Creating categories

we show the following. After creating the category **RentCars**, the user uses **Add Category with WordNet** operation to create a sub category **Car** under **RentCars**. WordNet proposes that there are five different *meanings* in the category **Car**, and for each meaning, it suggests **Car**'s sub categories and their synonyms. The user chooses the one s/he wants and customises the proposed sub categories of the categorie **Car** (e.g. by removing unwanted sub categories, or editing synonyms, etc.). Figure 3 displays the sub categorise of **Car** suggested by WordNet (e.g. **Convertible**, **Coupe**, **Limousine**, **Sedan**, etc.) and its synonyms (e.g. **auto**, **automobile**, etc.). After creating the categories, the user uses **Add Attributes with WordNet** operation to see possible attributes of the categorie **Car**. WordNet proposes a large list of attributes, and their synonyms (e.g. **accelerator**, **airbag**, **sunroof**, etc.). The user customises the proposed attributes (e.g. by removing unwanted attributes, or adding synonyms, etc.). After editing the community schema is completed, the **Community Manager** generates corresponding class descriptions for the community.

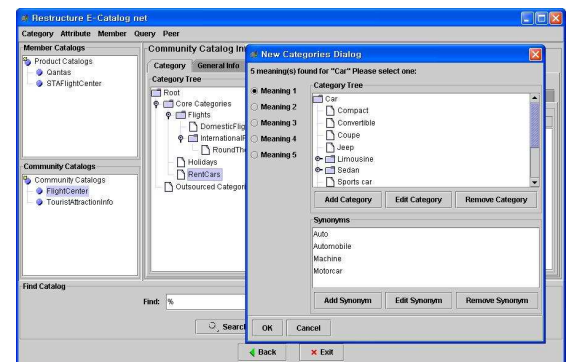


Figure 3: Creating categories using WordNet

3.2 Peering communities

The administrator of the community **FlightCenter** first searches for other communities in WS-CatalogNet via **Community Manager**'s search functionality. When s/he selects the community **TravelInfo**, the **Community Manager** displays categories and attributes

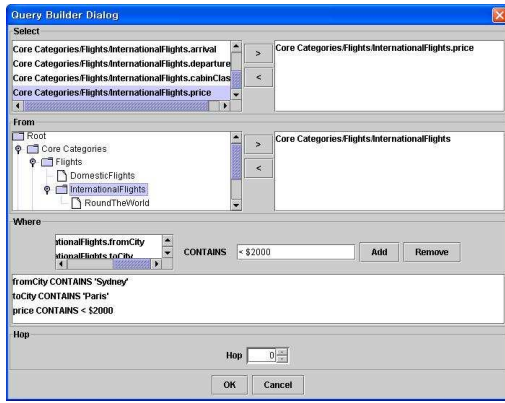


Figure 4: Expressing a query on FlightCenter

of TravelInfo. To create a peer relationship with category mapping information only, s/he clicks the category Holidays of the community FlightCenter and drag it to the category holidaysOffers of TravelInfo community.

3.3 Registering members

The e-catalog STAFlightCenter which provides cheap international flight tickets is already implemented as a web service. It has three categories Airfares, Insurance and CarHire. The provider of the e-catalog STAFlightCenter uses the Member Manager to display the categories of FlightCenter community and decides to register its own Airfares category with FlightCenter's Flights category. To register with this community, the provider points & clicks FlightCenter's categories and attributes that s/he can support (i.e., Flights category and its attributes). For each attribute in the category clicked, the provider selects, from his/her own list of attributes, the one that maps to it (e.g., Flights's price attribute is mapped to Airfares's farePrice)³. After mapping information is entered, the member manager generates corresponding class descriptions.

3.4 Querying the community FlightCenter

As shown in figure 4, the Query Manager displays the community FlightCenter's categories and attributes to the user. The user then formulate a query, mainly by pointing&clicking, "category:InternationalFlights, attributes: fromCity=Sydney, toCity=Paris, price ≤ 1000, travelInsurance= full ". The Query Manager generates corresponding class description of the user query. Then it runs the Best Quality Rewriting algorithm to compute Q_{local} and Q_{rest} . 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). The

³Corresponding figure not shown due to space reasons.

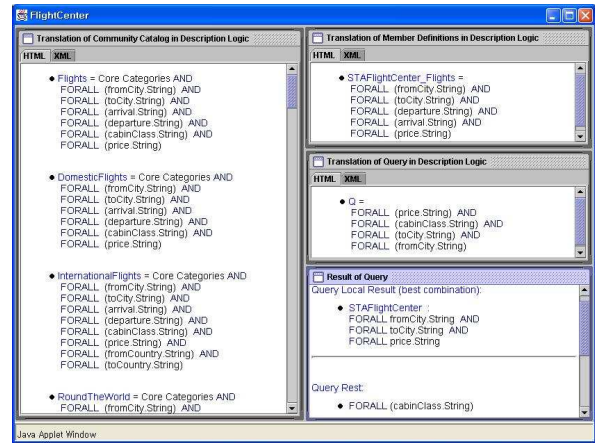


Figure 5: Querying FlightCenter using BQR

Q_{rest} part of the query contains only the attribute (travelInsurance). In this scenario, the community FlightCenter uses a predefined query forwarding policy. It specifies that Q_{rest} should be forwarded to all of its peers with hop limit of 3. The Q_{rest} (travelInsurance) is forwarded to the communities TravelInfo and WebJetDeal. After forwarding, both TravelInfo and WebJetDeal return SmileTravel and BestTravel as members who can answer travelInsurance respectively. The community FlightCenter decides to combine the e-catalogs STAFlightCenter (local) and SmileTravel (external, referred by TravelInfo) and send the query to them. Figure 5 shows the first result of running BQR.

Other scenarios of more complex queries will be presented in the demo to illustrate how peer communities collaborate to answer a user query. We show how a query is routed among several communities, where at each community the part of the query that can be answered by the local members is identified and the remaining part of the query is forwarded to other peers according to the community forwarding policy.

References

- [1] B. Benatallah, M.-S. Hacid, A. Leger, C. Rey, and F. Toumani. On Automating Web Services Discovery. *To appear in VLDB Journal*, 2004.
- [2] H.-Y. Paik, B. Benatallah, and F. Toumani. WS-CatalogNet: Building Peer-to-Peer E-Catalogs. In *FQAS 2004, Lyon, France.*, 2004.