MIT Sloan School of Management

MIT Sloan Working Paper 4562-05
CISL Working Paper 2005-06

August 2005

Information Aggregation using the Caméléon# Web Wrapper

Aykut Firat, Stuart Madnick, Nor Adnan Yahaya, Choo Wai Kuan, and Stéphane Bressan

Information Aggregation using the Caméléon# Web Wrapper

Aykut Firat, Stuart Madnick, Nor Adnan Yahaya,
Choo Wai Kuan, and Stéphane Bressan

Working Paper CISL# 2005-06

August 2005

Composite Information Systems Laboratory (CISL)
Sloan School of Management, Room E53-320
Massachusetts Institute of Technology
Cambridge, MA 02142

Information Aggregation using the Caméléon# Web Wrapper

Aykut Firat[1], Stuart Madnick[2], Nor Adnan Yahaya[3], Choo Wai Kuan[3], and Stéphane Bressan[4]

[1] Northeastern University, Boston, MA, USA
a.firat@neu.edu
[2] Massachusetts Institute of Technology, Cambridge, MA, USA
smadnick@mit.edu
[3] Malaysia University of Science and Technology, Petaling Jaya, Malaysia
{noradnan,wkchoo}@must.edu.my
[4] National University of Singapore, Singapore
steph@nus.edu.sg

Abstract. Caméléon# is a web data extraction and management tool that provides information aggregation with advanced capabilities that are useful for developing value-added applications and services for electronic business and electronic commerce. To illustrate its features, we use an airfare aggregation example that collects data from eight online sites, including Travelocity, Orbitz, and Expedia. This paper covers the integration of Caméléon# with commercial database management systems, such as MS SQL Server, and XML query languages, such as XQuery.

1 Introduction

We have argued [12] and illustrated in a case study [15] that information aggregation plays a critical role in the success of electronic business and electronic commerce services. Indeed, extraction and aggregation provide the foundation for added services leveraging the large amounts of data available on the public Internet and on Intranet that are waiting to be put in context and turned into information.

In this paper, we present the technology and tools that we have developed to achieve effective and efficient information extraction and aggregation: Caméléon#. Caméléon# is a web data extraction and aggregation tool that automates form submission; and dynamically converts semi-structured data into relational tables and XML documents. These converted data can then be queried with SQL and XQuery to facilitate interoperability across heterogeneous platforms. Caméléon#'s design and implementation make it Web service compliant and allow a seamless integration into a service oriented architecture. We introduce the features of Caméléon# by means of simple yet challenging examples.

In the financial information aggregation example shown in Fig. 1, internal and external semi-structured data sources are treated as if they were relational tables and aggregated into an MS Excel sheet using SQL through the use of Caméléon#.

Caméléon# associates each web source with a simple specification (spec) file that contains a virtual schema declaration with form submission and data extraction rules. Although we do not offer fully-automatic spec file generation, spec file creation is remarkably simple.

During the last decade or so, we have been successfully employing Caméléon# (and its predecessors) for research and teaching purposes. It is part of a larger semantic integration framework ECOIN [5], and has been used in a number of courses to introduce web data management concepts. Compared to other commercial and academic tools, we find Caméléon# better in its balance of simplicity and expressiveness; and capability to connect to problematic sites (e.g., sites that require "on the fly" javascript interpretation).

In the next section, we start with a quick background on wrappers. Then we explain the features of Caméléon# with a practical airfare example that collects price information from eight online airfare sources including Travelocity, Expedia, Orbitz, etc. We also discuss the integration of Caméléon# with a commercial database management system (MS SQL Server) and the XML query language XQuery.

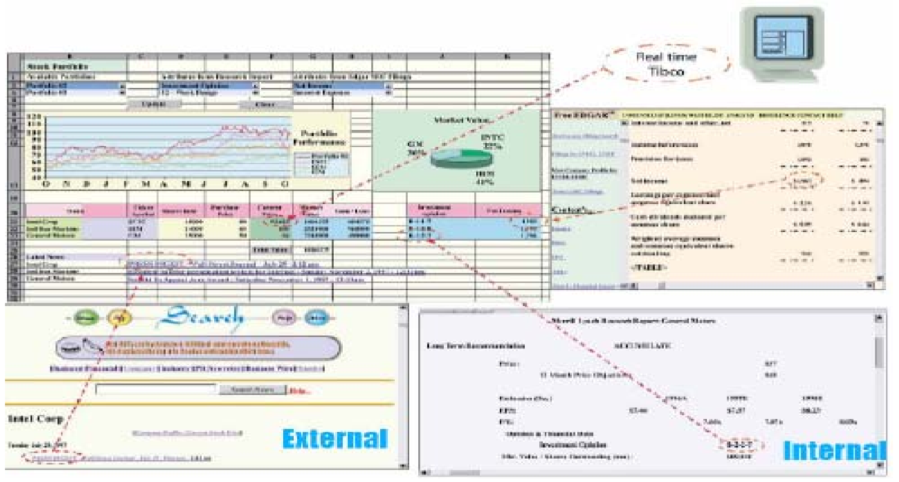


Fig. 1. Caméléon# in Financial Information Aggregation: Internal and External information sources are aggregated as if they were relational tables into an MS Excel sheet with SQL.

2 Background

During the boom years of the Internet, especially with the emergence of aggregators [12], there has been a proliferation of data extraction technologies, often-called Web wrappers (or wrappers for short).

A web wrapper is an engine capable of responding to some type of query by retrieving a web page S, based on:

1. A specification of path and parameterized inputs to get to the Web page S containing a set of implicit objects (and any other page S' similar to S); and then extracting data items based on:
2. A mapping specification W that postulates a data repository R with the objects in S. The mapping W must also be capable of recognizing and extracting data from any other page S' similar to S (see [11]).

We can classify wrappers according to how they treat documents (Web pages); how their mapping specifications are generated; and how declarative they are.

Wrappers treat Web pages either as a document tree or as a data stream. Wrapper engines like W4F [14] and Lixto [2] parse Web pages using Document Object Model (DOM) into a tree, and mapping specifications are expressed primarily in terms of the DOM. Other wrapper engines such as TSIMMIS [6] and Caméléon# ignore the HTML tag-based hierarchy and treat Web pages as a sequence of characters. Mapping specifications in this category are usually expressed in terms of regular expressions.

Wrappers can be manual, semi-automatic, or automatic based on how their mapping specifications are generated. In the manual approach (e.g., Jedi [7]), users create general extraction rules by analyzing a representative set of web pages, and are responsible for updating the specification files when necessary. In automatic generation, users first have to annotate a number of training examples through a visual interface (e.g., SoftMealy [8]). Machine learning algorithms, such as inductive learning, are then applied to generate the mappings (e.g., Wien [10], Stalker [13]). Semi-automatic approaches do not use any machine-learning algorithms but try to make the spec file creation easier through mappings between the visual and text/DOM views, by making suggestions on patterns that need to be approved or modified by the user.

Manual approaches are known to be tedious, time-consuming and require some level of expertise concerning the wrapper language. In addition, when web sites change, specification files have to be updated manually as well. Given the state of the art in automatic wrapper creation, however, automatic approaches are not very successful in creating *robust* wrappers. The maintenance costs of current automatic approaches are also comparable to manual and semi-automatic approaches, since in the automatic approach the user has to annotate new training samples when the wrapped web pages are modified. In fact, as noted by [9], it is unrealistic to assume that a user is willing and has the skills to browse a large number of documents in order to identify a set of informative training examples. While new approaches are being suggested that require a small number of training samples [9], their applicability is limited to simpler Web pages that do not contain various sorts of exceptions. On difficult web pages the lack of informative examples would lead to low accuracy.

A third grouping can be made according to how declarative mapping specifications are. In this context, "*declarative*" implies a clear separation of mapping specifications from the computational behavior of the wrapping engine. "Lowly declarative" wrapper engines mix mapping specifications with a programming language (e.g., W4F with Java) or offer a programming language of their own (e.g., Compaq's WebL) (see [4]). In "highly declarative" wrapper engines, extraction rules are separated from the computation logic and do not require any compilation of the rules into executable code.

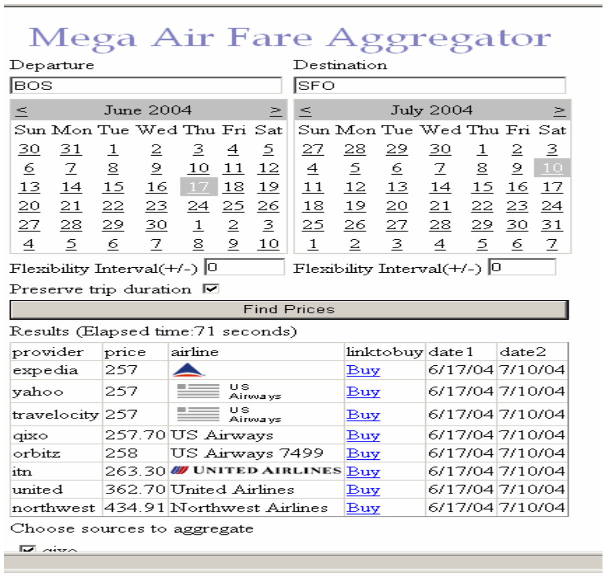
Based on these three dimensions, existing academic wrappers can be classified as shown in Table 1. A recent survey of commercial engines can be found in [4].

Table 1. Classification of Web Wrapper Projects (see [4] and [11] for references)

	Highly Declarative		Lowly Declarative	
	DOM-based	*Stream-based*	*DOM-based*	*Stream-based*
Manual		Tsimmis	Jedi	Araneus
Semi-automatic	NoDoSe	Caméléon#	W4F	
Automatic	Lixto	WIEN, Stalker	XWrap	

3 Airfare Aggregation with Caméléon#

One of several applications built with Caméléon# is 'Mega Air Fare Aggregator' shown in Fig. 2 (after an execution to find prices between Boston and San Francisco).



Mega Air Fare Aggregator

Departure: BOS
Destination: SFO

<			June 2004			>
Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	1	2	3
4	5	6	7	8	9	10

<			July 2004			>
Sun	Mon	Tue	Wed	Thu	Fri	Sat
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

Flexibility Interval(+/-) 0
Flexibility Interval(+/-) 0
Preserve trip duration ☑

Results (Elapsed time:71 seconds)

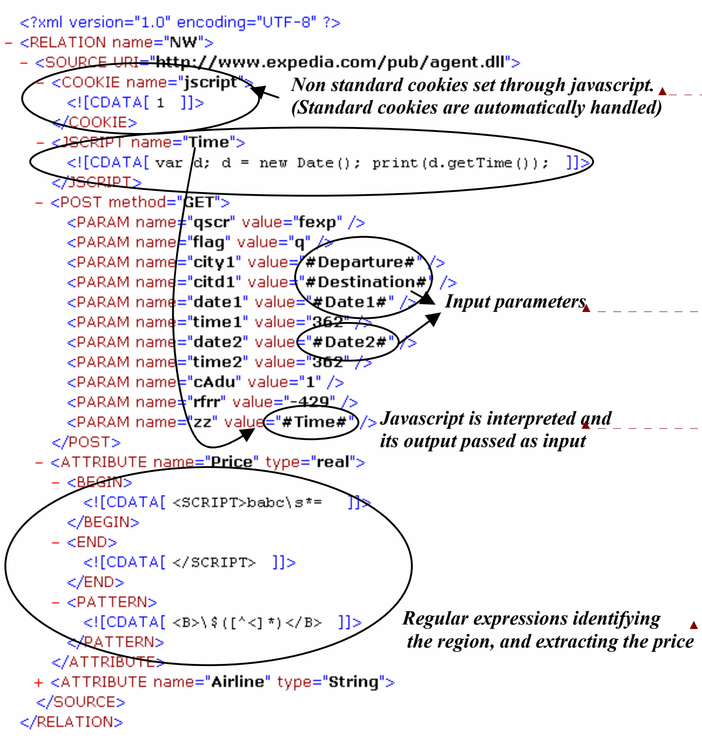
provider	price	airline	linktobuy	date1	date2
expedia	257		Buy	6/17/04	7/10/04
yahoo	257	US Airways	Buy	6/17/04	7/10/04
travelocity	257	US Airways	Buy	6/17/04	7/10/04
qixo	257.70	US Airways	Buy	6/17/04	7/10/04
orbitz	258	US Airways 7499	Buy	6/17/04	7/10/04
itn	263.30	UNITED AIRLINES	Buy	6/17/04	7/10/04
united	362.70	United Airlines	Buy	6/17/04	7/10/04
northwest	434.91	Northwest Airlines	Buy	6/17/04	7/10/04

Choose sources to aggregate
☑ qixo

Fig. 2. Mega Airfare Aggregator

The core of this application is a SQL query in the form of

```
(Select provider, price, airline, linktobuy, date1,
date2
From expedia
Where date1= '6/17/04' and date2= '7/10/04' and Depar-
ture= 'BOS' and Destination = 'SFO'
UNION
…
UNION
Select provider, price, airline, linktobuy, date1,
date2
From travelocity
Where date1= ‘6/17/04’ and date2= ‘7/10/04’ and Depar-
ture= ‘BOS’ and Destination = ‘SFO’)
Order By price ASC
```



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <RELATION name="NW">
  - <SOURCE URI="http://www.expedia.com/pub/agent.dll">
    - <COOKIE name="jscript">
        <![CDATA[ 1  ]]>
      </COOKIE>
    - <JSCRIPT name="Time">
        <![CDATA[ var d; d = new Date(); print(d.getTime());  ]]>
      </JSCRIPT>
    - <POST method="GET">
        <PARAM name="qscr" value="fexp" />
        <PARAM name="flag" value="q" />
        <PARAM name="city1" value="#Departure#" />
        <PARAM name="citd1" value="#Destination#" />
        <PARAM name="date1" value="#Date1#" />
        <PARAM name="time1" value="362" />
        <PARAM name="date2" value="#Date2#" />
        <PARAM name="time2" value="362" />
        <PARAM name="cAdu" value="1" />
        <PARAM name="rfrr" value="-429" />
        <PARAM name="zz" value="#Time#" />
      </POST>
    - <ATTRIBUTE name="Price" type="real">
      - <BEGIN>
          <![CDATA[ <SCRIPT>babc\s*=   ]]>
        </BEGIN>
      - <END>
          <![CDATA[ </SCRIPT>  ]]>
        </END>
      - <PATTERN>
          <![CDATA[ <B>\$([^<]*)</B>  ]]>
        </PATTERN>
      </ATTRIBUTE>
    + <ATTRIBUTE name="Airline" type="String">
    </SOURCE>
  </RELATION>
```

Non standard cookies set through javascript.
(Standard cookies are automatically handled)
Input parameters
Javascript is interpreted and its output passed as input
Regular expressions identifying the region, and extracting the price

Formatted: Font: Bold, Italic

Formatted: Font: Bold, Italic

Formatted: Font: Bold, Italic

Formatted: Font: Bold, Italic

Fig. 3. Specification File for Expedia.

Here, web sites are treated as relational tables through their specification files. As an example, the spec file for the Expedia web site is shown in Fig. 3. In this example, air fare prices from Expedia are obtained through a single form submission; therefore the spec file has a single source declaration. Despite single page traversal, Expedia is a difficult site for two reasons. First, there are cookies, which are set in a non-standard way through Javascript. Because of that, automatic cookie handling will not be able to acquire and supply them. In Caméléon#, custom cookies can be specified as shown in Fig. 3. Second, the Expedia site requires an input form parameter (Time) whose value is determined by some Javascript code. Failure to interpret Javascript will also make it impossible for wrappers to connect to this site. In Caméléon#, we take advantage of Microsoft's .Net framework, which allows mixing different languages with the provision of common intermediate layer CLR (like Java's bytecode). This way, we are able to interpret Javascript code dynamically.

In Fig. 3, after specification of form parameters (those enclosed with # signs are input parameters that are expected in the where clause of a SQL query to Caméléon#), the name of the attribute and its data type are specified. For each attribute, regular expressions inside begin and end tags denote a region in a document, and the expression inside the pattern tag extracts the values for the attribute.

Once spec files for all the airfare sites are constructed, they can be treated as relational tables. It then becomes trivial to construct the airfare aggregation previously shown.

4 Integration with RDBMS

While *core* Caméléon# creates the illusion of an RDBMS to query web data sources, its query support is limited to simple queries in the form of '`Select ... From ... Where`'. To support full SQL, additional steps must be taken. Below we explore three ways of achieving this goal.

4.1 OLE-DB Provider for Caméléon#

OLE-DB is Microsoft's way of building common access to data sources including text and XML files, (although they are being deprecated and replaced by .NET data providers). Sources with OLE-DB providers can be linked to SQL Server and utilize its industry strength query planner and execution engine.

We have built an OLE-DB provider for Caméléon#, and the details of it are described in [3]. With this OLE-DB provider, it is possible to issue arbitrary SQL queries with the help of openrowset function (in SQL Server).

```
select  *
from openrowset ('OLEDBCamProv.CamProv', '', ' Select
provider, price, airline, linktobuy, date1, date2 from
expedia where date1= '6/17/04' and date2= '7/10/04' and
Departure= 'BOS' and Destination = 'SFO'')
```

One problem with the openrowset function, however, is that the SQL Server query planner treats it as a black box; and does not use it in the planning and optimization phase of the query. Queries in the openrowset are executed as they are without any optimization. To overcome this problem, Caméléon# engine must satisfy minimum conformance requirements to SQL 92, which is not a trivial task to undertake in developing OLE-DB providers. Besides, there is no clear indication that OLE-DB providers can be developed for functional sources, which require certain input parameters to be specified every time a query is issued against existing tables.

Ignoring optimization issues, the OLE-DB provider for Caméléon# does provide integration with SQL Server.

4.2 Parameterized Views

In SQL-Server it is possible to model web data as functions that return tables. For example, the Expedia web site could be modelled with the following function like a parameterized view:

```
CREATE FUNCTION fnexpedia (@DepDate smalldatetime,
@ArrDate smalldatetime, @DepCity char(3), @ArrCity
char(3))
returns @fnexpedia table (DepCity char(3),
ArrCity char(3), DepDate smalldatetime, ArrDate small-
datetime, Price real, Airline varchar(30))
AS
BEGIN
DECLARE @query VARCHAR(255)
DECLARE @Date1 char(8), @Date2 char(8)
SET @Date1=CONVERT(char(8), CAST (@DepDate AS small-
datetime), 1)
SET @Date2=CONVERT(char(8), CAST (@ArrDate AS small-
datetime), 1)
SET @query = 'CaméléonSQL "Select Price, Airline From
expedia where Departure="' + @DepCity +'" and Destina-
tion="' + @ArrCity + '" and Date1="' + @Date1 +'" and
Date2="' + @Date2 + '" "'
EXEC master..xp_cmdshell  @query
insert     @fnexpedia
Select  @DepCity , @ArrCity, @DepDate, @ArrDate, Price,
Airline From expedia
RETURN
END
```

In the above function Caméléon# executes the query, creates a temporary table and bulk loads the results into that table. The users can then call the Expedia web site as if it was a parameterized view as follows:

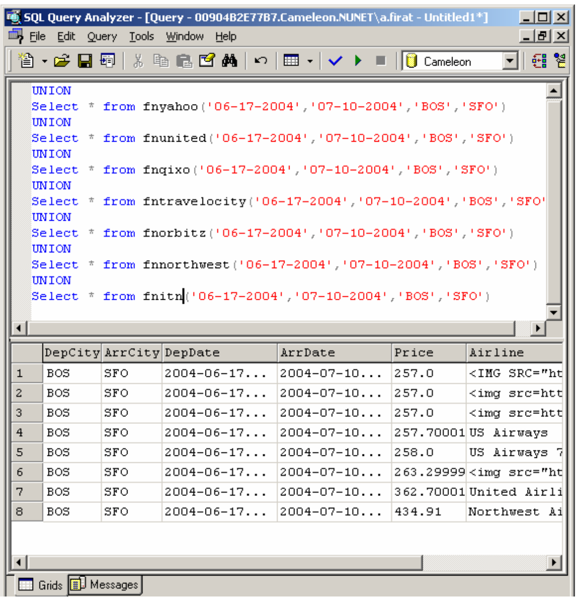
```
Select *
from fnexpedia('06-17-2004','07-10-2004','BOS','SFO')
```

Airfare prices can then be obtained from SQL Server Client as shown in Fig. 4.

One difficulty with this approach, however, is that it is not possible to use these functions with variables in a SQL statement. For example, the following statement would not be meaningful in SQL:

```
Select price
from fnexpedia('06-17-2004','07-10-2004','BOS',Des),
targetcitycodes t
where  Des = t.Destination
```

Furthermore, contrary to expectation this union query in SQL Server is not executed in parallel.


SQL Query Analyzer - [Query - 00904B2E77B7.Cameleon.NUNET\a.firat - Untitled1*]
File Edit Query Tools Window Help
Cameleon
Grids
Messages

```
UNION
Select * from fnyahoo('06-17-2004','07-10-2004','BOS','SFO')
UNION
Select * from fnunited('06-17-2004','07-10-2004','BOS','SFO')
UNION
Select * from fnqixo('06-17-2004','07-10-2004','BOS','SFO')
UNION
Select * from fntravelocity('06-17-2004','07-10-2004','BOS','SFO'
UNION
Select * from fnorbitz('06-17-2004','07-10-2004','BOS','SFO')
UNION
Select * from fnnorthwest('06-17-2004','07-10-2004','BOS','SFO')
UNION
Select * from fnitn('06-17-2004','07-10-2004','BOS','SFO')
```

	DepCity	ArrCity	DepDate	ArrDate	Price	Airline
1	BOS	SFO	2004-06-17...	2004-07-10...	257.0	<IMG SRC="ht
2	BOS	SFO	2004-06-17...	2004-07-10...	257.0	<img src=htt
3	BOS	SFO	2004-06-17...	2004-07-10...	257.0	<img src=htt
4	BOS	SFO	2004-06-17...	2004-07-10...	257.70001	US Airways
5	BOS	SFO	2004-06-17...	2004-07-10...	258.0	US Airways 7
6	BOS	SFO	2004-06-17...	2004-07-10...	263.29999	<img src="ht
7	BOS	SFO	2004-06-17...	2004-07-10...	362.70001	United Airli
8	BOS	SFO	2004-06-17...	2004-07-10...	434.91	Northwest Ai

Fig. 4. Parameterized Views & SQL Server Client.

4.3 Custom Planner/Optimizer/Execution Engine

Finally, we mention our “capabilities aware” custom planner, optimizer and execution (POE) engine that works on top of the Caméléon# core. The central concept in this custom POE engine is the concept of a capability record to represent the capability restrictions of Web sources. An example capability record for a currency exchange web site, olsen, is shown below:

```
relation(cameleon, olsen, [ ['Exchanged',string], ['Ex-
pressed',string], ['Rate',number],
['Date',string]],cap([[b(1),b(1),f,b(1)]],
['<','>','<>','<=','>=']).
```

This simple capability record expresses binding restrictions as a list of all possible binding combinations of the attributes in the virtual relation. A binding combination specifies attributes that need to be bound; attributes that need to be free; and attributes that can be either free or bound. It is represented with a list of binding specifiers for each of the attributes in the relation. A binding specifier can be one of the following: b, b(N), f, and ?. b indicates that the attribute has to be bound. b(N) indicates that the attribute has to be bound with N keys-at-a-time binding restriction. f indicates that the attribute must be free. ? indicates that the attribute can be either bound or free. The record for operator restrictions is a list of the operators, which cannot be used in queries on the relation.

Note that key-at-a time restrictions are quite common among the web wrapped relations. The olsen source can only bind one key at a time for its attributes Exchanged, Expressed, and Date. Key-at-a-time restrictions that can bind more than key at a time (N>1) are also common. A good example of this is a stock quote server like finance.yahoo.com, which allows up to 50 stock quote symbols to be entered at one time.

Based on capability records our custom POE engine produces an optimized plan respecting the capability restrictions. The core Caméléon# and a local RDBMS are then used to execute the plan. More details on this can be found in [1].

5 Integration with XQuery

Since Caméléon# can return results in XML, it becomes trivial to integrate it with XQuery. The airfare results can be obtained with the following XQuery:

```
<Airfare>
{let $travelocity :=
doc(http://interchange.mit.edu/Cameleon_sharp/camserv.asp
x?query=Select Airline, Price from expedia where Destina-
tion="SFO" and Departure="BOS" and Date1="6/12/04" and
Date2= "7/12/04"&amp; format=xml")//price
           …
let $expedia :=
doc(http://interchange.mit.edu/Cameleon_sharp/camserv.asp
x?query=Select Airline, Price from expedia where Destina-
tion="SFO" and Departure="BOS" and Date1="6/12/04" and
Date2= "7/12/04"&amp; format=xml")//price
           …
return
  <Results>
           <travelocity>{ $travelocity }</travelocity>
           <itn>{ $itn }</itn>
           <qixo>{ $qixo }</qixo>
           <yahoo>{ $yahoo }</yahoo>
           <orbitz>{ $orbitz }</orbitz>
           <united>{ $united }</united>
           <northwest>{ $northwest }</northwest>
           <expedia>{ $expedia }</expedia>
  </Results>
  }
  </Airfare>
```

XQuery implementations execute this query in parallel, and under a minute, which is quite remarkable given that in SQL-Server the union query took almost 5 minutes to complete.

6. Spec File Management

Caméléon# reduces the effort of aggregating data to the definition of the spec file. Yet, this effort is not insignificant. The scalability of the Caméléon# approach depends on opportunities for re-using and sharing spec files in communities of users.

For this reason, we developed a spec file management system to help store, manage and share spec files, as shown in Fig. 5.

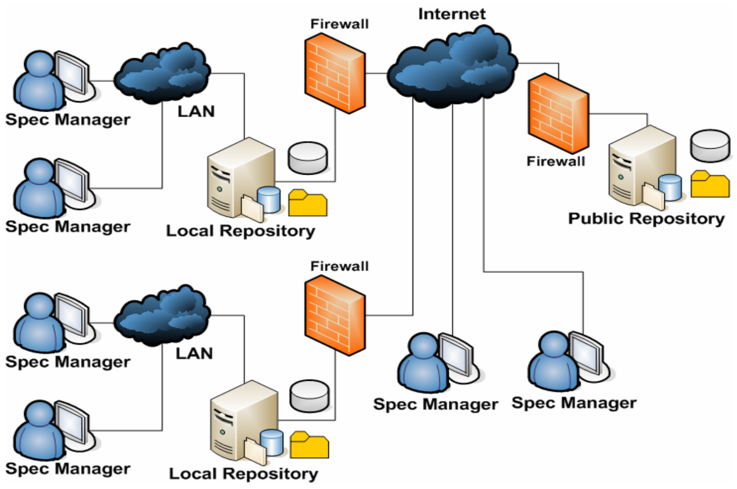


Fig. 5. Spec File Repository Architecture.

A public repository is created to archive all spec files within a community of interest (a company, a group of users, etc.). Public here does not mean its access is not controlled, it is public within a community. There is only one such repository. However, it is connected to several local repositories. Local repositories are usually available only to the internal network of a community. The local repositories periodically communicate their spec files to the public repository. The architecture of the spec file repository is shown in Fig. 5.

The spec manager is a suite developed to assist user in the spec file creation, edition and publication. The spec manager client consists of tools such as web browser, spec file editor, spec file tester, regular expression tester and spec file searcher.

7 Conclusion

We described Caméléon#, a tool for extraction and aggregation of data from various sources. We illustrated the simplicity of use and the power of Caméléon# with the example construction of an application such as the mega airfare aggregator. Caméléon# is used in research and teaching as well as in industrial applications.

References

1. Alatovic, T.: Capabilities Aware, Planner, Optimizer, Executioner for Context Interchange Project. Thesis (S.M.) M.I.T, Dept. of EE & CS (2001)
2. Baumgartner, R., Flesca, S., Gottlob, G. (2001). "Declarative Information Extraction, Web Crawling, and Recursive Wrapping with Lixto". In Proc. LPNMR'01, Vienna, Austria, 2001.
3. Chan, C.: OLE DB for the Context Interchange System. Thesis (S.M.) M.I.T, Dept. of EE & CS (2000)
4. Chuang, S.W.: A Taxonomy and Analysis of Web Wrapping Technologies. Thesis (S.M.) M.I.T, Technology and Policy Program (2004)
5. Firat, A.: Information Integration Using Contextual Knowledge and Ontology Merging Thesis (Ph.D.) M.I.T. (2003)
6. Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, V., Ullman, J., Widom, J. (1995). Integrating and Accessing Heterogeneous Information Sources in TSIMMIS. In Proceedings of the AAAI Symposium on Information Gathering, pp. 61-64, Stanford, California, March 1995
7. Huck, G., Fankhauser, P., Aberer, K., Neuhold, E. J.: JEDI: Extracting and Synthesizing Information from the Web; submitted to COOPIS 98, New York; IEEE Computer Society Press, (1998)
8. Hsu, C., and Dung, M. (1998). Wrapping semistructured web pages with finite-state transducers. In Proceedings of the Conference on Autonomous Learning and Discovery CONALD-98.
9. Knoblock, C., Lerman, K., Minton, S., Muslea, I.: Accurately and reliably extracting data from the web: A machine learning approach, IEEE Data Engineering Bulletin, 23(4), (2000)
10. Kushmerick, N., Doorenbos, R., Weld., D. (1997) Wrapper Induction for Information Extraction. IJCAI-97, August 1997.
11. Laender, A., Ribeiro-Neto, B., Silva, A. and Teixeira, J.: A Brief Survey of Web Data Extraction Tools, SIGMOD Record, 31(2), (2002)
12. Madnick, S. and Siegel, M: Seizing the Opportunity: Exploiting Web Aggregation, MIS Quarterly Executive, 1(1), (2002)
13. Muslea, I., Minton, S., and Knoblock, C. (1998) STALKER: Learning extraction rules for semistructure, Web-based information sources. In Proc. of AAAI'98: Workshop on AI and Information Integration.
14. Sahuguent, A. and Azavant, F.: W4F: the WysiWyg Web Wrapper Factory. Technical Report, University of Pennsylvania, Department of Computer and Information Science, (1998)
15. Zhu, H., Siegel, M. and Madnick, S.: Information Aggregation – A Value-added E-Service, Proc. of the International Conference on Technology, Policy, and Innovation: Critical Infrastructures, (2001)