J. Hammer, M. Stonebraker, and O. Topsakal, "THALIA: Test Harness for the Assessment of Legacy Information Integration Approaches." University of Florida, Gainesville, FL, Technical Report TR05-001, January 2005.

THALIA: Test Harness for the Assessment of Legacy Information Integration Approaches

Joachim Hammer[†], Mike Stonebraker[‡], and Oguzhan Topsakal[†]

[†]*Department of Computer & Information Science & Engineering*
CSE Building
University of Florida
Gainesville, FL 32611-6120
{jhammer,otopsaka}@cise.ufl.edu

[‡]*Computer Science and Artificial Intelligence Laboratory*
The Stata Center, Building 32
Massachusetts Institute of Technology
32 Vassar Street - Cambridge, MA 02139
stonebraker@lcs.mit.edu

Abstract

We introduce a new, publicly available testbed and benchmark called THALIA (Test Harness for the Assessment of Legacy information Integration Approaches) to simplify the evaluation of existing integration technologies and to enable more thorough and more focused testing. THALIA provides researchers with a collection of downloadable data sources representing University course catalogs, a set of twelve benchmark queries, as well as a scoring function for ranking the performance of an integration system. Our benchmark focuses on syntactic and semantic heterogeneities since we believe they still pose the greatest technical challenges. A second important contribution of this paper is a systematic classification of the different types of syntactic and semantic heterogeneities, which directly lead to the twelve queries that make up the benchmark. A sample evaluation of two integration systems at the end of the paper is intended to show the usefulness of THALIA in identifying the problem areas that need the greatest attention from our research community if we want to improve the usefulness of today's integration systems.

1. Introduction

Information integration refers to the unification of related, heterogeneous data from disparate sources, for example, to enable collaboration across domains and enterprises. Despite the significant efforts in developing effective solutions and tools for automating information integration, current techniques are still mostly manual, requiring significant programmatic set-up with only limited reusability of code. As a result, these approaches do not scale to data integration problems involving a large number of sources, especially if the source schemas and contents are dynamic. For example, the time and investment needed to integrate heterogeneous data from different enterprises has acted as a significant barrier to the adoption of collaborative environments (e.g., electronic supply networks), which require efficient and flexible sharing across hundreds of sources.

In defense of ongoing research in this area, information integration is a very challenging problem, which must be solved at various levels of abstraction [15], ranging from *system-level* heterogeneities (i.e., differences at the hardware level), to *structural* (*syntax)-level* heterogeneities (i.e., differences at the schema level), to heterogeneities at the *semantic level* (i.e., differences in the meaning and usage of similar data). Information integration has been an active area of research since the early 80's and produced a plethora of techniques and approaches to integrate heterogeneous information. As a result, determining the quality and applicability of a solution is a daunting task and has been the focus of several studies (e.g., [1]). What makes such comparisons even more challenging is the lack of available test data of sufficient richness and volume to allow meaningful and fair evaluations. Instead, researchers use their own test data and evaluation techniques, which are tailored to the strengths of the approach and often hide any existing weaknesses.

The goal of this paper is to introduce our new, publicly available *testbed and benchmark called THALIA*[1] (Test Harness for the Assessment of Legacy information Integration Approaches) for testing and evaluating integration technologies. THALIA[2] provides researchers with a collection of over 25 downloadable data sources[3] representing University course catalogs from computer science departments around the world. In addition, THALIA provides a set of twelve benchmark queries as well as a scoring function for

[1] Derived from Greek *thallein* meaning "to blossom".
[2] See http://www.cise.ufl.edu/research/dbintegrate/
[3] Expected to reach 45 sources by August 2004.

ranking the performance of an integration system. Our benchmark focuses on syntactic and semantic heterogeneities since we believe they still pose the greatest technical challenges to the research community. A second important contribution of this paper is a *systematic classification of the different types of syntactic and semantic heterogeneities*, which directly lead to the twelve queries that make up the benchmark. We believe that THALIA will not only simplify the evaluation of existing integration technologies but also help researchers improve the accuracy and quality of future approaches by enabling more thorough and more focused testing.

The rest of this paper is organized as follows. In Sec. 2 we describe the structure, implementation, and contents of our testbed using the course catalog of Brown University as an example. Sec. 3 provides our classification of the different types of heterogeneities and the corresponding benchmark queries, each posing a specific integration challenge. Sect. 4 briefly summarizes the state-of-the-art in integration research and illustrates the use of the benchmark by evaluating two sample integration systems, Cohera's Content Integration System and the University of Florida's Integration Wizard [6]. Sec. 5 concludes the paper and provides some thoughts on future directions in information integration.

2. Introduction to THALIA

As stated in the introduction, there is a need for a publicly available testbed to compare integration technologies and to determine the overall effectiveness of an approach. For example, under the recently announced SEIII program[4], the National Science Foundation specifically supports the creation of shared resources environments containing archived data in support of information integration research. While testbeds for evaluation of integration technologies are not new[5], what distinguishes THALIA is the fact that it combines rich test data with a set of benchmark queries and associated scoring function to enable the objective evaluation and comparison of integration systems.

2.1 Testbed

Our testbed currently provides access to course information from 25 computer science departments at Universities around the world. We have chosen course information as our domain of discourse because it is well known and easy to understand. Furthermore, there is an abundance of data sources publicly available that allowed us to develop a testbed exhibiting all of the syntactic and semantic heterogeneities that we have identified in our classification (Sec. 3).

Figure 1 shows a sample course catalog from the CS department at Brown University providing information such as course number, instructor, title, time and location in tabular format. Of particular interest are the columns labeled "Instructor" and "Title/Time." The Instructor column contains a hyperlinked string which points to the home page of the instructor for additional information. This means that the name value must also be interpreted as a pointer to the continuation of the value, in case additional information about an instructor, for example, first name, specialty, etc. is to be retrieved. The Title/Time column contains a concatenation of the course title (some of which are hyperlinked) and time when the course is offered. As we can see, not all of the time information is regular (e.g., CS034) making extraction difficult. Finally, the challenge in interpreting the meaning of attribute values can be seen in the case of the column labeled "Room" which not only contains the location of the lecture but in some cases that of the lab as well.

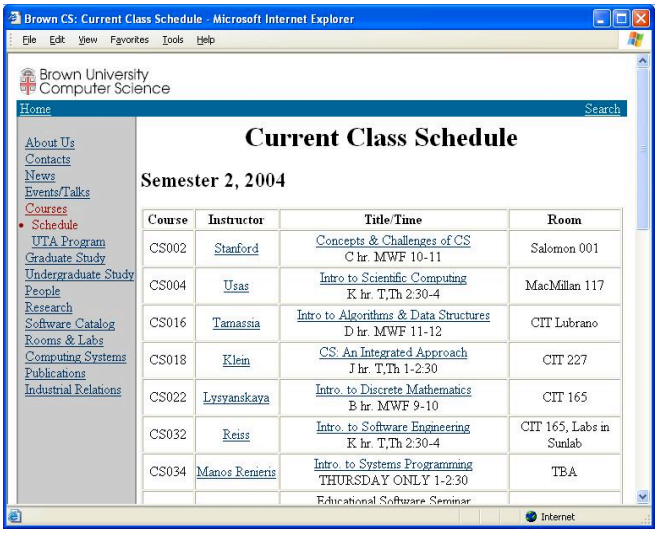

Figure 1: Original Course Catalog from Brown University.

In THALIA, University course catalogs are represented using well-formed and valid XML according to the extracted schema for each course catalog. Extraction and translation from the original representation is done using a source-specific wrapper. Although the wrapper removes heterogeneities at the system-level it preserves structural and semantic heterogeneities that exist among the different catalogs.

We used the Telegraph Screen Scraper (TESS)[6] system developed at UC Berkeley to extract the source data

[4] http://www.nsf.gov/pubs/2004/nsf04528/nsf04528.htm

[5] See, for example, the *Web-Based Knowledge Representation Repositories* project at the Vrije Universieit Amsterdam (wbkr.cs.vu.nl/) or the *UIUC Web Integration Repository* (metaquerier.cs.uiuc.edu/repository).

[6] http://telegraph.cs.berkeley.edu/tess/

and produce the XML output. For each source, a configuration file specifies which fields TESS should extract; beginning and ending points for each field are identified using regular expressions. Although the original TESS system could successfully extract information from catalog with simple structure such as the one from Brown University (Fig. 1), it could not parse complex catalogs such as the one from the University of Maryland shown in Figure 2, for example. The combination free-form structure and nested table required modification to TESS, which was not designed to extract multiple lines from a nested structure. In addition, we needed to provide TESS with additional configuration information specifying the structure of those fields that contain nested substructure.

When TESS is invoked on a source for which a configuration file exists, the XML output is produced according to the specifications provided in the configuration file. The astute reader will of course notice that this approach only works as long as the source remains unchanged. Any syntactic changes to the underlying source must also be reflected in the configuration file for this source. Given the usual turnover time of course catalogs, which is roughly 2-3 times a year, we do not expect many changes to the configuration files once they are created.

Figure 2: Snapshot of the CS course catalog of the University of Maryland.

As a final note, TESS is currently only capable of extracting data from a single Web page or document. Extraction of content from multiple pages or documents that are linked (e.g., instructor's home page or course description in Fig. 1) requires multiple configuration files (one for each link). We have not implemented this type of deep-extraction yet and return the URL of the link (instead of the contents of the linked page) as the extracted value.

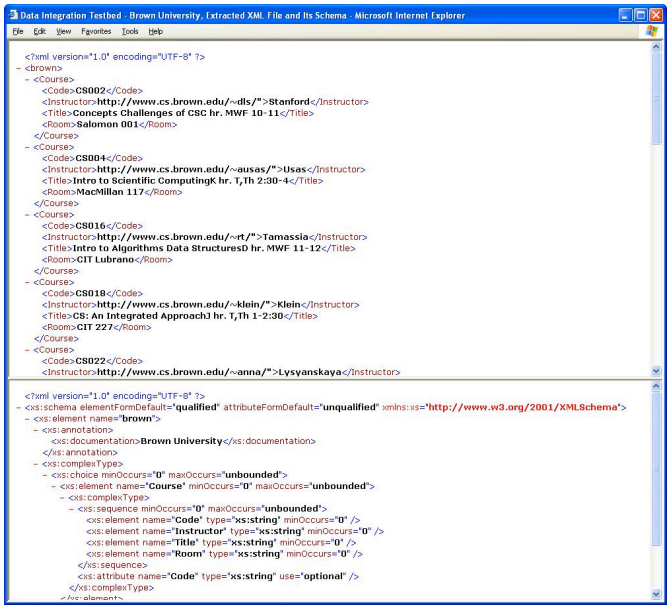

Figure 3: XML representation of Brown University's course catalog and corresponding schema file.

Figure 3 depicts the extracted XML output and corresponding XML schema for Brown University. As we can see, the schema of the XML document remains as close to the original schema of the corresponding catalog as possible. We preserve any semantic heterogeneities by retaining the names of the schema elements. If the source is a table-formatted HTML page, we create a `Course` tag and put all the information in a row under this tag. Also, we create a tag for each column in a row and select the column's title as the name of that tag.

2.2 Web Site

We have created a Web site to simplify the access to the testbed and the benchmark described in Sec. 3 below. A snapshot of the home page is shown in Figure 4 depicting the available interface options in the left-hand frame. Specifically, THALIA supports the browsing of the available course catalogs in their original representation ('University Course Catalogs' button), as well as the viewing of the extracted XML documents and corresponding schemas ('Browse Data and Schema'). It is important to note that the THALIA

testbed contains cached snapshots of the University course schedules. This decision was the result of several weeks of testing during which many of the University sources exhibited frequent connection problems or became otherwise unavailable for prolonged periods. Since up-to-dateness of the catalog data is much less important than its availability, we believe this option makes sense.

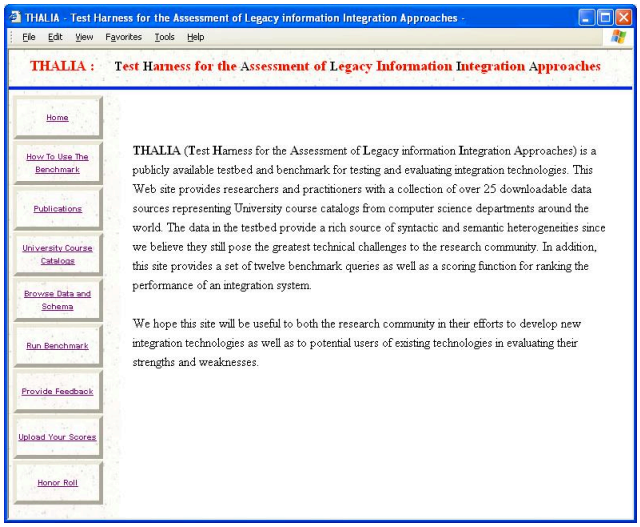

Figure 4: Snapashot of THALIA's Home Page.

The 'Run Benchmark' option provides the user with three choices: (1) Download a zipped file containing the XML and XML Schema files of all available course catalogs. We believe this option will be useful to those researchers and developers looking for test data for the experimentation with new integration solutions. (2) Download a zipped file containing the twelve benchmark queries as well as the corresponding test data sources. We believe this option will be useful to researchers wishing to evaluate the capabilities of their integration solution using a third-party, independent benchmark. Furthermore, the benchmark scores will be the first publicly available measure of integration technologies allowing the objective comparison of integration solutions. Note that integration systems that do not provide query processing can still use the benchmark by providing an integrated schema over the two data sources associated with each benchmark query. (3) Download a zipped file containing sample solutions to each of the benchmark queries including an XML Schema file describing the structure of the integrated XML result.

In order to facilitate the comparison of technologies, we invite users of the benchmark to upload their benchmark scores ('Upload Your Scores') which can be viewed by anybody using the 'Honor Roll' button. We now describe the benchmark queries and underlying heterogeneities in more detail.

3. Integration Benchmark

In this section, we start by illustrating the different types of heterogeneities that are exhibited by the data sources in our testbed. For each type of heterogeneity, we have formulated a *benchmark query* against two data sources from our testbed that requires a particular integration activity in order to provide the correct result. We are tacitly assuming that an integration system is capable of processing each of the benchmark queries by breaking it into subqueries, which can be answered separately using the extracted XML data from the underlying sources, and by merging the results into an integrated whole. The details of how query decomposition and result merging are achieved are not important in this context. Using THALIA, integration systems can be evaluated and compared based on the number of correctly answered benchmark queries as well as on the amount of programmatic "integration effort" that must be invested in order to answer each query.

3.1 Twelve Benchmark Queries

In the context of THALIA we are only interested in structural and semantic heterogeneities which we have further divided into twelve separate cases. For each case, we have formulated a benchmark query which must be answered against at least two schemas from the testbed: a *reference schema*, which is used to formulate the query, as well as a *challenge schema* which exhibits the type of heterogeneity that is to be resolved by the integration system. Note, in some cases (e.g., Benchmark Query 9), a query may illustrate additional types of heterogeneities that are also showcased in other queries. Queries are written in XQuery version 1.0.

In what follows, we briefly describe each heterogeneity and the associated benchmark query including sample data[7] from its reference and challenge schemas. Rather than adopting the commonly-used but very general distinction between structural and semantic heterogeneities, we grouped the twelve cases based on commonalities among the different heterogeneities. In each of the three groups, heterogeneities are shown in increasing order of complexity of the effort that is needed to resolve them.

Attribute Heterogeneities: These are heterogeneities that exist between two related attributes in different schemas and include the following five cases.

1. *Synonyms:* Attributes with different names that convey the same meaning. For example, "instructor" vs. "lecturer."

[7] All sample data elements are taken directly from the translated course catalogs; any irregularities are part of the test data.

Benchmark Query: List courses taught by instructor "Mark"

```
FOR $b in
doc("gatech.xml")/gatech/Course
WHERE $b/Instructor = "Mark"
RETURN $b
```

Reference University: Georgia Tech University
Sample Element:

```
<Course>
  ...
  <CRN>20381</CRN>
  ...
  <Instructor>Mark</Instructor>
  ...
  </Course>
```

Challenge University: Carnegie Mellon University
Sample Element:

```
<Course>
  <Code>15-567*</Code>
  ...
  <Lecturer>Mark</Lecturer>
  ...
</Course>
```

Challenge: Determine that in CMU's course catalog the instructor information can be found in a field called "Lecturer".

2. *Simple Mapping:* Related attributes in different schemas differ by a mathematical transformation of their values. For example, representing time values using a 24 hour vs. 12 hour clock.

   Benchmark Query: Find all database courses that meet at 1:30pm on any given day.

```
FOR $b in doc("cmu.xml")/cmu/Course
WHERE $b/Course/Time='1:30 - 2:50'
RETURN $b
```

   Reference University: Carnegie Mellon University[8]
   Sample Element:

```
<Course>
  ...
  <CourseTitle>Database System Design
  and Implementation</CourseTitle>
  ...
  <Time>1:30 - 2:50</Time>
  ...
</Course>
```

   Challenge University: University of Massachusetts
   Sample Element:

```
<Course>
  <Code>CS430</Code>
  ...
  <Times>16:00-17:15</Times>
  ...
</Course>
```

   Challenge: Conversion of time represented in 12 hour-clock to 24 hour-clock.

3. *Union Types:* Attributes in different schemas use different data types to represent the same information. For example, representing course description as a single string vs. complex data type composed of strings and links (URLs) to external data.
   Benchmark Query: Find all courses with the string 'Data Structures' in the title.

```
FOR $b in doc("umd.xml")/umd/Course
WHERE $b/CourseName='%Data
Structures%'
RETURN $b
```

   Reference University: University of Maryland
   Sample Element:

```
<Course>
  <Code>CMSC420</Code>
  <CourseName>Data
Structures;</CourseName>
  ...
</Course>
```

   Challenge University: Brown University
   Sample Element:

```
<Course>
  <Code>CS016</Code>
  ...

  <Title>http://www.cs.brown.edu/co
  urses/cs016/">Intro to Algorithms
  & Data Structures D hr. MWF 11-
  12</Title>
  ...
</Course>
```

   Challenge: Map a single string to a combination external link (URL) and string to find a matching value. Note, in addition, this query exhibits a synonym heterogeneity (CourseName vs. Title).

4. *Complex Mappings:* Related attributes differ by a complex transformation of their values. The transformation may not always be computable from first principles. For example, the attribute "Units" represents the number of lectures per week vs. textual description of the expected work load in field "credits"

   Benchmark Query: List all database courses that carry more than 10 credit hours.

[8] On the actual Web site, columns titled Room, Day and Time are mislabeled. We assume that this was unintentional and corrected the error in the extracted XML file.

```
FOR $b in doc("cmu.xml")/cmu/Course
WHERE $b/Units >10 AND
$b/CourseName='%Database%'
RETURN $b
```

Reference University: Carnegie Mellon University
Sample Element:

```
<Course>
  ...
  <CourseTitle>Database System Design
  and Implementation</CourseTitle>
  ...
  <Units>12</Units>
</Course>
```

Challenge University: Swiss Federal Institute of Technology Zürich (ETH)
Sample Element:

```
<Unterricht Nummer="251-0317-00">
  <Titel>XML und Datenbanken</Titel>
  <Umfang>2V1U</Umfang>
</Unterricht>
```

Challenge: Apart from the language conversion issues, the challenge is to develop a mapping that coverts the numeric value for credit hours into a string that describes the expected scope (German: "Umfang") of the course.

5. *Language Expression:* Names or values of identical attributes are expressed in different languages. For example, the English term "database" is called "Datenbank" in the German language.

   Benchmark Query: Find all courses with the string 'database' in the course title.

```
FOR $b in
doc("umd.xml")/umd/Course
WHERE $b/CourseName='%Database%'
RETURN $b
```

Reference University: University of Maryland
Sample Element:

```
<Course>
  ...
  <CourseName>Database
  Design;</CourseName>
  ...
</Course>
```

Challenge University: Swiss Federal Institute of Technology Zürich (ETH)
Sample Element:

```
<Unterricht Nummer="251-0317-00">
  ...
  <Titel>XML und Datenbanken</Titel>
  ...
</Unterricht>
```

Challenge: For each course in the catalog of ETH, convert the German tags into their English counterparts to locate the one representing course name. Convert the English course title 'Database' into its German counterpart 'Datenbank' or 'Datenbanksystem' and retrieve those courses from ETH that contain that substring..

Missing Data: These are heterogeneities that are due to missing information (structure or value) in one of the schemas.

6. *Nulls:* The attribute (value) does not exist. For example, some courses do not have a textbook field or the value for the textbook field is empty.

   Benchmark Query: List all textbooks for courses about verification theory.

```
FOR $b in
doc("toronto.xml")/toronto/course
WHERE $b/title='%Verification%'
RETURN $b/text
```

Reference University: University of Toronto
Sample Element:

```
<course No="CSC 2108"
level="graduate" offeredTerm="Fall
2003">
  <title>Automated
  Verification</title>
  ...
  <text>'Model Checking', by Clarke,
  Grumberg, Peled, 1999, MIT
  Press.</text>
</course>
```

Challenge University: Carnegie Mellon University
Sample Element:

```
<Course>
  ...
  <CourseTitle>Specification and
  Verification</CourseTitle>
  ...
</Course>
```

Challenge: Proper treatment of NULL values. In the above query, the integrated result must include the fact that no textbook information was available for CMU's course.

7. *Virtual Columns:* Information that is explicitly provided in one schema is only implicitly available in the other and must be inferred from one or more values. For example, course prerequisites is provided as an attribute in one schema but exists only in comment form as part of a different attribute in another schema.

   Benchmark Query: Find all entry-level database courses.

```
FOR $b in
doc("umich.xml")/umich/Course
WHERE $b/prerequisite='None'
RETURN $b
```

Reference University: University of Michigan
Sample Element:

```
<course subject="EECS"
catalognumber="484">
  <name>Database Management
  Systems</name>
  <prerequisite>None</prerequisite>
  ...
  </course>
```

Challenge University: Carnegie Mellon University
Sample Element:

```
<Course>
  ...
  <CourseTitle>Database System
  Design and Implementation
  First course in
  sequence</CourseTitle>
  ...
</Course>
```

Challenge: Infer the fact that the course is an entry-level course from the comment field that is attached to the title.

8. *Semantic incompatibility:* A real-world concept that is modeled by an attribute does not exist in the other schema. For example, the concept of student classification ('freshman', 'sophomore', etc.) at American Universities does not exist in German Universities.

   Benchmark Query: List all database course open to juniors.

```
FOR $b in
doc("gatech.xml")/gatech/Course
WHERE $b/Course restricted=%JR%'
RETURN $b
```

Reference University: Georgia Tech
Sample Element:

```
<Course>
  ...
  <Title>Intro-Network
  Management</Title>
  ...
  <Course restricted>JR or
  SR</Course restricted>
</Course>
```

Challenge University: The Swiss Federal Institute of Technology Zurich, ETH
Sample Element:

```
<Unterricht Nummer="251-0019-00">
  <Titel>Vernetzte Systeme (3.
  Semester)</Titel>
  ...
</Unterricht>
```

Challenge: Although one could return a NULL value for ETH, such an answer is quite misleading. To deal intelligently with this query one must support more than one kind of NULL. Specifically, one must distinguish "data missing but could be present" (see case 6) from "data missing and cannot be present".

Structural Heterogeneities: These are heterogeneities that are due to discrepancies in the way related information is modeled/represented in different schemas.

9. *Same attribute in different structure:* The same or related attribute may be located in different positions in different schemas. For example, the attribute `Room` is an attribute of `Course` in one schema while it is an attribute of `Section` which in turn is an attribute of `Course` in another schema.

   Benchmark Query: Find the room in which the database course is held.

```
FOR $b in
doc("brown.xml")/brown/Course
WHERE $b/Title ='Software
Engineering'
RETURN $b/Room
```

Reference University: Brown University
Sample Element:

```
<Course>
  ...
  <Title>Intro. to Software
  EngineeringK hr. T,Th 2:30-
  4</Title>
  <Room>CIT 165, Labs in
  Sunlab</Room>
</Course>
```

Challenge University: University of Maryland
Sample Element:

```
<Course>
  ...
  <CourseName>Software
  Engineering;</CourseName>
  <Section>
    <Time>TuTh..... 2:00pm- 3:15pm
    (CSI 1121)</Time>
  </Section>
  ...
  </Course>
```

Challenge: Determine that room information in the University of Maryland's course catalog is available as part of the time element located under the Section element.

10. *Handling sets:* A set of values is represented using a single, set-valued attribute in one schema vs. a collection of single-valued attributes organized in a hierarchy in another schema. For example, a course with multiple instructors can have a single

attribute `instructors` or multiple `section-instructor` attribute pairs.

Benchmark Query: List all instructors for courses on software systems.

```
FOR $b in doc("cmu.xml")/cmu/Course
WHERE $b/CourseTitle ='%Software%'
RETURN $b/Lecturer
```

Reference University: Carnegie Mellon University
Sample Element:

```
<Course>
  ...
  <CourseTitle>Secure Software
  Systems</CourseTitle>
  <Lecturer>Song/Wing</Lecturer>
  ...
</Course>
```

Challenge University: University of Maryland
Sample Element:

```
<Course>
  ...
  <CourseName>Software
  Engineering;</CourseName>
  ...
  <Section>
    <Title>0101(13795) Singh,
    H.</Title>
    ...
  </Section>
  <Section>
    <Title>0201(13796) Memon, A.
    (Seats=40, Open=2,
    Waitlist=0)</Title>
    <Time>TuTh..... 9:30am-10:45am
    (CSI 2107)</Time>
  </Section>
  ...
</Course>
```

Challenge: Determine that instructor information is stored as part of the Section information in the University of Maryland catalog. Specifically, the instructor information must be gathered by extracting the name part from all of the section titles rather than from a single field called Lecturer in CMU's case.

11. *Attribute name does not define semantics*: The name of the attribute does not adequately describe the meaning of the value that is stored there.

    Benchmark Query: List instructors for the database course.

```
FOR $b in doc("cmu.xml")/cmu/Course
WHERE $b/Course Title ='%Database'
RETURN $b/Lecturer
```

Reference University: Carnegie Mellon University
Sample Element:

```
<Course>
  ...
  <CourseTitle>Database System
   Design and
   Implementation</CourseTitle>
  <Lecturer>Ailamaki</Lecturer>
  ...
</Course>
```

Challenge University: University of California, San Diego
Sample Element:

```
<Course>
  ...
  <Title>Database System
  Implementation</Title>
  <Fall2003>Yannis</Fall2003>
  <Winter2004>-</Winter2004>
  <Spring2004>Deutsch</Spring2004>
</Course>
```

Challenge: In the case of the catalog for the Univ. of San California, San Diego, one must associate columns labeled "Fall 2003", "Winter 2004" etc. with instructor information.

12. *Attribute composition:* The same information can be represented either by a single attribute (e.g., as a composite value) or by a set of attributes, possibly organized in a hierarchical manner.

    Benchmark Query: List the title and time for computer networks courses.

```
FOR $b in doc("cmu.xml")/cmu/Course
WHERE $b/CourseTitle ='%Computer
Networks%'
RETURN <Course>
        <Title> $b/Title </title>
          <Day>  $b/Day </Day>
        <Time> $b/Time </time>
   </course>
```

Reference University: Carnegie Mellon University
Sample Element:

```
<Course>
  ...
  <CourseTitle>Computer
  Networks</CourseTitle>
  ...
  <Day>F</Day>
  <Time>1:30 - 4:20</Time>
  ...
 </Course>
```

Challenge University: Brown University
Sample Element:

```
<Course>
  ...
  <Title>Computer NetworksM hr. M 3-
  5:30</Title>
  ...
</Course>
```

Challenge: Determine that course title, day and time information in the catalog of Brown University are represented as part of the title attribute rather than as separate attributes as in the case of CMU. Extract the correct title, day and time values from the title column in the catalog of Brown University.

The above list of twelve heterogeneities is not an exhaustive list as different application domains may exhibit additional heterogeneities not seen in the current collection of course catalogs. However, the cases above cover most of the heterogeneities discussed in the literature (e.g., [10, 15]) and thus represent a reasonable subset of cases that must be resolved by an integration system. Since we are still adding new data sources, we will also update the benchmark queries if necessary.

3.2 Running the Benchmark and Evaluating the Outcome

The benchmark can be used to evaluate the performance of an integration system in two ways: (1) In cases where the integration system is capable of answering queries against multiple heterogeneous sources, the system can attempt to answer each of the benchmark queries against the provided test sources directly. If necessary the benchmark queries may need to be translated into the native language of the integration system first. (2) Integration subsystems that focus on the reconciliation of heterogeneities and do not have a query processor can be evaluated on their ability to integrate the given test sources for each of the twelve heterogeneities. The output should correspond to the schema that is described by the benchmark query.

The performance of a benchmark run can be evaluated as follows. Each correct answer is worth 1 point for a total of 12 points max. Sample answers to all benchmark queries are available at the THALIA Web site. For those queries for which the integration system has to invoke an external function in order to aid in the generation of the answer, the level of complexity of the function must be scored on a scale of low (1 point), medium (2 points), and high (3 points). This complexity score will be used to further rank those systems which get the same number of correct answers: The higher the complexity score, the lower the level of sophistication of the integration system. We demonstrate the usage of the benchmark and scoring function in the section below, when we describe two integration systems as part of our summary of the current state-of-the-art.

4. State-of-the-art in Information Integration

4.1 Overview

There has been a large amount of work on federating disparate data bases, dating back to the mid 80's. In fact, Google has 104,000 citations for this topic. Hence, it would be silly to try to cover any significant part of this space. Many organizations that built distributed data bases, object-oriented data bases, gateways, and adaptors have offerings in this area. Also, the research community has been incredibly active on this topic, with efforts ranging from view integration [16] sharing architectures [15] and languages [8] including multi-source query processing [19] to schema matching [13] and data translation [12] and cleansing [14]. As a result we merely touch on a few points in this section. The interested reader can consult one of any number of textbooks for more information, e.g. [11].

The distributed data bases from the early 1980's (e.g. R* [20] and Distributed Ingres [18]) had essentially no support for heterogeneity. Ingres corporation implemented Ingres* in the late 1980's, and supported access to a variety of foreign DBMSs through gateways. More recently, Cohera extended this support to include user-defined functions to specify mappings from local schemas to a global schema. IBM's distributed data system, Data Joiner [9], now called the Information Integrator, contains the same sort of capabilities, which were first explored in the TSIMMIS [5] and Garlic [7] projects.

The work of Halevy [3] extends the framework above by allowing federated data bases which have no global schema. In effect, any local schema can be used to specify a query, and pairwise transformations are allowed between local schemas. The system looks for transformation paths through the distributed system to answer queries.

Although information integration has been widely addressed, we believe that it is far from solved. For example, we know of no system that can score well on the THALIA benchmark, and the next section discusses two example integration systems.

4.2 Two Integration Systems and their Performance on the Benchmark

This section indicates how the federated DBMS Cohera and the University of Florida's Integration Wizard [4] projects would work on the THALIA benchmark. Since Cohera, a commercial version of Mariposa [17], was purchased in 2001, and is not available currently, it is impossible to actually run the benchmark. Hence, we suggest how it would have performed.

Cohera supported the notion of a **local schema** at each site to be integrated, as well as a **global schema**, on which users ran queries. Cohera allowed transformations that could map from one schema to the other. There were a couple of different ways to specify these transformations. First, Cohera included a "web site wrapper", which was the inspiration for the TESS system used in the THALIA test harness. This wrapper constructed records from a web page in a very flexible way. Moreover, Cohera gave users the power of Postgres in specifying local to global schema transformations. Of note is the ability to write user-defined functions in C to perform transformations.

With this background, Cohera would implement the benchmark as follows:

- Query 1 (renaming columns): supportable by the Cohera local to global schema mapping.
- Query 2 (24 hour clock): supportable with a user-defined function – small amount of code
- Query 3 (union data types): supportable by a user defined union type and appropriate conversion routines – moderate amount of code.
- Query 4 (meaning of credits): no easy way to deal with this, without large amounts of custom code.
- Query 5 (language translation): no easy way to deal with this, without large amounts of custom code.
- Query 6 (nulls): Postgres had direct support for nulls
- Query 7 (virtual attributes): same answer as query 3.
- Query 8 (semantic incompatibility): no easy way to deal with this
- Query 9 (attribute in different places): supportable by the Cohera local to global schema mapping
- Query 10 (sets): supportable by the Cohera local to global schema mapping
- Query 11 (name does not define semantics): same answer as queries 3 and 7.
- Query 12 (run on columns): same answer as queries 3, 7 and 11.

In summary, Cohera could do 4 queries with no code, and another 5 with varying amounts of user-defined code. The other 3 queries look very difficult.

The Integration Wizard (IWIZ) combines the data warehousing and mediation approaches to offer integrated access to heterogeneous Web sources. IWIZ allows end-users to issue queries based on a global schema to retrieve information from data sources without knowledge about API and data source representation. However, unlike existing systems, queries that can be satisfied using the contents of the IWIZ warehouse, are answered quickly and efficiently without connecting to the sources.

IWIZ provides wrappers for translating data from local schemas into the global IWIZ schema as well as a mediator for processing queries against the global schema. The mediator is also capable of merging and cleansing heterogeneous data from multiple sources. IWIZ does not support user-defined functions per se. However, it provides a 4GL to specify local to global transformations in the wrappers as well as special-purpose cleansing and integration procedures in the mediator at build-time. IWIZ uses XML as its internal data model.

IWIZ implements the benchmark as follows:

- Query 1 (renaming columns): supported by the IWIZ local to global schema mappings – small amount of code.
- Query 2 (24 hour clock): supported with a special-purpose function – small amount of code
- Query 3 (union data types): supported by a customized union type and appropriate conversion routines – moderate amount of code.
- Query 4 (meaning of credits): no easy way to deal with this, without large amounts of custom code.
- Query 5 (language translation): no easy way to deal with this, without large amounts of custom code.
- Query 6 (nulls): no direct support for nulls; requires moderate amount of custom code to indicate missing values in the result.
- Query 7 (virtual attributes): same answer as query 3.
- Query 8 (semantic incompatibility): no easy way to deal with this
- Query 9 (attribute in different places): supported by the IWIZ local to global schema mapping – small amount of code.
- Query 10 (sets): supported by IWIZ's local to global schema mapping – small amount of code.
- Query 11 (name does not define semantics): same answer as queries 3 and 7.
- Query 12 (run on columns): same answer as queries 3, 7 and 11.

In summary, IWIZ could do 9 queries with small to moderate amounts of custom integration code. The remaining 3 queries cannot be answered by IWIZ.

5. Conclusion and Proposed Directions

This paper has presented a benchmark and an easy to use collection of test data to support it. The Lowell Report [2] asked for exactly this sort of scheme to stimulate further research in real world integration problems. As can be seen in the preceding section, current systems do not score well, and we hope that THALIA will be an inducement for research groups to construct better solutions.

We are currently adding new data sources to the testbed, which will provide additional sources of heterogeneity. We are also soliciting feedback on the

usefulness of the benchmark including the results of running our benchmark on as many of the existing integration systems as possible.

References

[1] H.-H. Do, S. Melnik, and E. Rahm, "Comparison of schema matching evaluations," in *GI-Workshop on Web and Databases*. Erfurt, Germany, 2002.

[2] J. Gray, H. Schek, M. Stonebraker, and J. Ullman, "The lowell report," in *2003 ACM SIGMOD International Conference on Management of Data*, San Diego, CA, 2003.

[3] A. Halevy, Z. Ives, D. Suciu, and I. Tatarinov, "Schema mediation in peer data management systems," in *International Conference on Data Engineering*, 2003.

[4] J. Hammer, "The information integration wizard (iwiz) project," University of Florida, Gainesville, FL, Technical Report TR99-019, November 1999.

[5] J. Hammer, H. Garcia-Molina, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom, "Integrating and accessing heterogeneous information sources in tsimmis," in *AAAI Symposium on Information Gathering*, Stanford, CA, 1995.

[6] J. Hammer and C. Pluempitiwiriyawej, "Overview of the integration wizard project for querying and managing semistructured data in heterogeneous sources," in *Fifth Computer Science and Engineering Conference (NCSEC 2001)*, Chiang Mai University, Chiang Mai, Thailand, 2001.

[7] V. Josifovski, P. Schwarz, L. Haas, and E. Lin, "Garlic: A new flavor of federated query processing for db2," in *SIGMOD 2002*, Madison, WI, USA, 2002.

[8] L. V. S. Lakshmanan, F. Sadri, and I. N. Subramanian, "Schemasql - a language for interoperability in relational multi-database systems," in *Twenty-Second International Conference on Very Large Databases*. Mumbai, India: Morgan Kaufman, 1996.

[9] P. G. a. E. T. Lin., "Datajoiner: A practical approach to multidatabase access," in *Intl. IEEE Conf. on Parallel and Distributed Information Systems*, Austin, TX, USA, 1994.

[10] S. B. Navathe, R. ElMasri, and J. Larson, "Integrating user views in database design," in *IEEE Computer*, 1986.

[11] M. T. Ozsu and P. Valduriez, *Principles of distributed database systems*: Prentice-Hall publishers, 552 pp., 1991.

[12] Y. Papakonstantinou, A. Gupta, H. Garcia-Molina, and J. Ullman, "A query translation scheme for rapid implementation of wrappers," in *Fourth International Conference on Deductive and Object-Oriented Databases*, Singapore, 1995.

[13] E. Rahm and P. A. Bernstein, "A survey of approaches to automatic schema matching," *VLDB Journal: Very Large Data Bases*, vol. 10, pp. 334-350, 2001.

[14] E. Rahm and H. H. Do, "Data cleaning: Problems and current approaches," in *IEEE Data Engineering Bulletin*, vol. 23, 2000.

[15] A. Sheth and J. A. Larson, "Federated database systems for managing distributed, heterogeneous, and autonomous databases," *ACM Computing Surveys*, vol. 22, pp. 183-236, 1990.

[16] A. P. Sheth, J. A. Larson, and E. Watkins, "Tailor, a tool for updating views," in *International Conference on Extending Database Technology: Advances in Database Technology*, 1988.

[17] M. Stonebraker, P. M. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu, "Mariposa: A wide-area distributed database system," *VLDB Journal: Very Large Data Bases*, vol. 5, pp. 48-63, 1996.

[18] Michael Stonebraker, *The ingres papers: Anatomy of a relational database system*: Addison-Wesley (Reading MA), pp. 452, ACM CR 8604-0296, 1986.

[19] J. D. Ullman, "Information integration using logical views," in *International Conference on Database Theory*, 1997.

[20] R. Williams and et al., "R*: An overview of the architecture," IBM Research, San Jose, CA, Technical Report RJ3325, December 1981.