IBM UFO Repository

OBJECT-ORIENTED DATA INTEGRATION

Michael N. Gubanov[1,2], Lucian Popa[2], Howard Ho[2], Hamid Pirahesh[2], Jeng-Yih Chang[2,3], Shr-Chang Chen[2,3]
[1] *University of Washington,* [2] *IBM Almaden Research Center*
[3] *National Yang-Ming University*
mgubanov@cs.washington.edu, {lucian, ho, pirahesh}@almaden.ibm.com
g39528020@ym.edu.tw, sinergy@ibms.sinica.edu.tw

1. INTRODUCTION

Currently, *WWW*, large enterprises, and desktop users suffer from an inability to efficiently access and manage *differently structured* data. The same data objects (e.g. Product) stored by different databases, repositories, distributed web storage systems, *etc* are named, referenced, and combined internally into schemas or data structures differently. This leads to *structural mismatch* of data that often consists of the same semantic objects (e.g. EBay and Yahoo! online auction offers).

Web 2.0 offered a multitude of mashups, microformats, and tagged data without a convenient way to access, deploy, and exchange them. Yahoo! Pipes, Microsoft Popfly, Google mashup editor, IBM DAMIA [22], and other mashup fabrics entered the arena here to bridge the gap between *differently structured* heterogeneous data. Large enterprises, having suffered for years from the *Data Integration Curse* [1], could improve their situations with new data management tools built with significant help from the research community [15, 8, 7, 9, 10, 17, 22, 2, 11, 19, 14, 23, 26, 12, 3, 6, 13, 4, 24, 20, 5]. While this work represents significant progress that surpasses *WWW* and *Web 2.0*, none of the solutions are unsupervised yet, and they require significant human effort.

Finally, biological and medical domains have many public and proprietary databases storing probably the most complex and large structures (e.g. human anatomy) in the world. All of them suffer from the same *curse* and are a subject of significant body of ongoing research in Bio-Informatics (e.g. [17, 18, 21]).

In all these domains the cornerstone of the problem is in *structural mismatch* of data from different sources and insufficient data *modularity* to make it more accessible. Monolithic data representation (e.g. one large XML file) with abundance of low-level storage/representation details are hard to query or extract needed information, because understanding their structure requires significant expert effort upfront. As a general solution, we propose *Object-oriented* data integration that has a collection of *Unified Famous Objects (UFOs)* as a cornerstone. Our proposed *Object-oriented* data integration introduces a higher-level abstraction, the `Unified Famous Object (UFO)`, and leverages it to simplify data management. In the same way that the Java Object hides *implementation* details behind its interface, a Unified Famous Object conceals data *representation* differences. Having a large collection of `UFOs` would significantly simplify data access and exchange by automatically recognizing objects in the incoming data feeds and offering a standard query interface oblivious of the source schemas. Finally, `UFOs` are more general and flexible than schemas in a sense that they can be viewed as abstract building blocks for metadata oriented applications.

Similarly, when the use of object-oriented programming was not widespread, large programs were quite hard to build and maintain. They consisted of a large number (i.e. thousands) of functions and data structures passed among these functions. Debugging and modifying was quite challenging due to poor *modularity.* The object-oriented programming paradigm changed programming practice, so that a program became *modular* and therefore easier to implement and maintain [25, 16]. It consists of a collection of well-defined pieces (i.e. objects) and a sequence of interaction between them. Objects by themselves deal with their own implementation, which is hidden from a program that manipulates them as 'black-box' pieces.

Similarly, by introducing `UFOs` we bring *modularity* that drastically simplifies data access and management. First, the repository detects familiar objects inside the incoming data feeds and, second, it offers *object-oriented* standardized access.

For instance, by querying the standard `UFO-Item`, the user can easily find the best offer among several electronics stores without prior knowledge of original sources schemas. Similarly she would be able to recompose incoming *Web 2.0* mashups such as *Products*, *Reviews*, and *Merchants* to put them on Google maps without prior knowledge of data source schema. To keep it up, the repository offers tools to enrich the `UFOs` collection from external sources.

In this paper we briefly describe the architecture, demo-scenarios illustrated by implementation screenshots and result sets. We conclude by discussing some forward-looking

[1]inability to efficiently manage and integrate differently structured data

statements that were born as we worked on UFOs.

2. ARCHITECTURE

The UFO repository components are in Figure 1. The system provides uniform access to data residing in a variety of data sources ranging from *Web 2.0* mashups, online auction data feeds, to biological, medical, and enterprise structured data. It accumulates and maintains a large collection of UFOs; leverages it to *automatically* discover familiar objects in incoming data feeds; offers standard querying interface that is oblivious to the original data source representation. This allows to rise the abstraction level and increase automation by hiding the specific object structure differences under a standard UFO interface. The information is accessed by querying a collection of standard UFOs; new data feeds are imported by having the repository discover and map objects similar to existing UFOs.

Neither data warehousing nor ETL [19, 11] attempt to introduce a common abstraction (i.e. UFO) and leverage it to raise automation level. In some sense, those approaches are less general in that they put main efforts on the specifics of data mapping and transformation and are built for *specific* query languages, schema, and data formats. By contrast, we try to crystallize higher-level abstractions that conceal lower-level technological details to manipulate these abstractions instead.

Another approach would be federation [10] of incoming data together with query reformulation to the original sources, which is currently a subject of ongoing work. Next we describe the repository components in more detail.

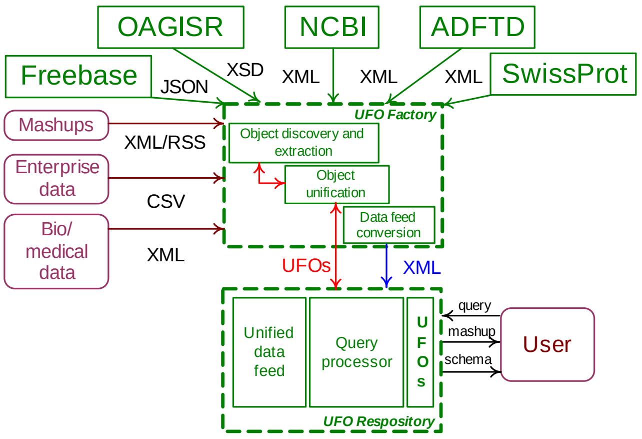


Figure 1: UFO repository architecture

Interface: Users and applications interact with the system through UFOs currently in the system (shown in Figure 2). The collection includes UFOs such as U-Item, U-Genome, U-Protein, etc. Currently, the repository uses a simple (i.e. flat) model of UFOs, but there is a clear need of subclasses and polymorphism (e.g. U-Truck is a subclass of U-Car).

Browsing: UFOs stored in the repository can be explored using a GUI in Figure 2 that displays domain hierarchy on the left and the UFO structure with relations to other UFOs on the right. The demo also implements *keyword*-search over UFOs.

Table 1 illustrates a fragment of U-Item. It is stored as XML entity and has standardized entity/attribute names

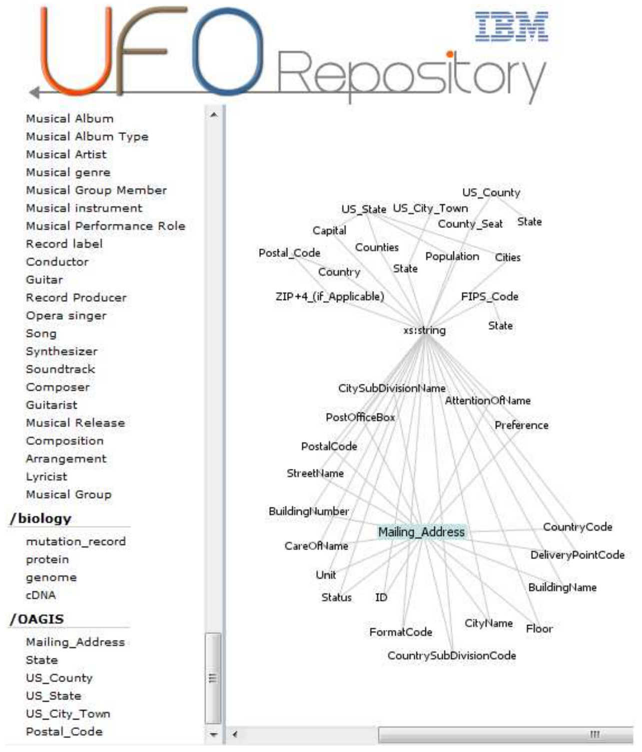


Figure 2: UFO Browser

```
<e name="Item" ufo_name="Item|Product|Catalog">
   <a name="Price"
        ufo_name="(BuyItNow|(Converted)?Current)?
                        Price(To|From)?" />
   <a name="URL"
        ufo_name="(ViewItem)?URL(ForNaturalSearch)?)" />
   <a name="Title"
        ufo_name="Title|(Product)?Name" />
 ...
</e>
```

Table 1: U-Item fragment in XML

(name field) that are queried by user and accumulated names (ufo_name field) that are invisible to user and used by object discovery algorithm. For instance, Title attributes from different Item representations encountered by the repository are accumulated as regular expression $Title|(Product)?Name$ that is used later on to match similar attributes.

By some means, the repository interface (a collection of UFOs) it can be viewed as an extensible, standardized *mediated schema* [17] over multiple data feeds. It is extensible, because a collection grows when it accumulates new object representations. The more representations are accumulated, the easier it is to discover familiar objects inside the incoming data feeds later on. It is standardized, because the UFOs query interface is static in contrast to varying source schemas and therefore easier to query. Whereas the repository currently offers one generic UFO collection split by domain, we are considering ways to reuse UFOs across domains by leveraging polymorphism.

`Querying, mapping:` `UFOs` in the repository represent an interface that supports XQuery. We demonstrate this by importing XML feeds from three online stores - eBay, Yahoo!, NewEgg; discovering and unifying the objects present in the feeds, and querying the integrated data through the standardized `U-Item UFO` interface.

Table 2 illustrates two queries:

- against merged eBay, Yahoo!, and NewEgg offers via `U-Item UFO` and
- against merged NCBI, AD&FTD, and SwissProt via `U-Sequence UFO`.

The first query automatically returns the best offers for a 32" TV among the three stores. The second query takes the sites from the patient's genetic profile and makes automatic disease diagnosis by querying mutation records for a specific gene. Only standardized `UFO` attributes are queried by the user, thereby lower level source differences are concealed.

Q1:

```
for $tv in db2-fn:xml('UOFFERS.AUCTIONS')/UFO/Item
where fn:contains($tv/Category, "Television")
        and fn:contains($tv/Title, "32")
order by ($tv/Price)
return ($tv/url, $tv/Price, $tv/Title)[1];
```

Results: http://cgi.sandbox.ebay.com/ws/...; $878.00; TV 32"LCD 1080P FULL HD

Q2:

```
for $y in db2-fn:xmlcolumn('USEQUENCE.MUTATION')/
                            MUTATION_RECORD,
where $y/before_mutation='E'
   and $y/after_mutation='G'
   and $y/site='280'
return ($y/disease, $y/description, $y/frequency,
        $y/tissue_specific);
```

Results: **Alzheimer disease, Spastic Paraparesis**; Mean of onset Ages: 44.8y Mean of ages at Death: 58.3y; AD3 is the most severe form of the disease, with complete penetrance and an onset occurring as early as 30 years of age. The second form is late-onset AD (LOAD), with mean age of onset greater than 58 years. AD is an autosomal dominant neurodegenerative disorder characterized by progressive dementia, parkinsonism, and deposition of fibrillar amyloid proteins as intraneuronal neurofibrillary tangles, extracellular amyloid plaques and vascular amyloid deposits. The major protein found within these deposits is a small, insoluble and highly aggregating polypeptide, beta-amyloid protein (beta-APP42)...; 6; Expressed in a wide range of tissues including various regions of the brain;

Table 2: Source-oblivious U-Item, U-Sequence XQueries. Without `UFOs`**, prior knowledge of all sources' structure (eBay, Yahoo!, NewEgg, NCBI, Swissprot, AD&FTD) and multiple different queries would be required to get the result set returned by one** `UFO`**- query.**

`Object discovery and accumulation:` Mappings generated by the *Clio* **UFO** Factory in Figure 3 can be used to discover and unify objects into UFOs. Also, to extract objects from an XML data feed, it is parsed, traversed in inorder and flattened to extract objects. Next, the objects are matched against the repository and unified with the best matching `UFO`.

`Exporting:` `UFOs` can be selected by the user and exported to compose a schema. For instance, a user can export a schema for *Banking* domain by selecting its properties like *cardinality of relationship* between entities, *level of detail*, and *denormalization.* Based on the user choices, the system generates XQuery that produces the output XML Schema shown in Figure 4.

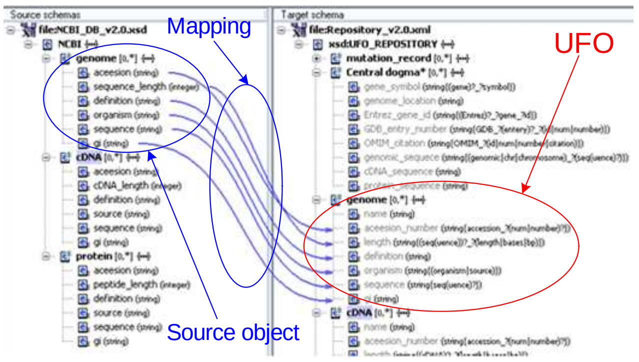


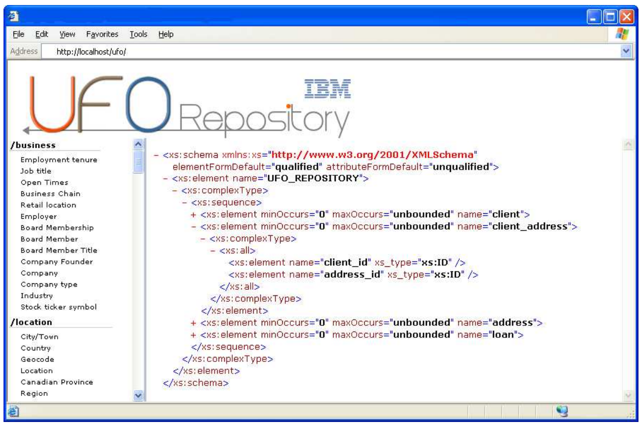
Figure 3: UFO Factory: Genome is discovered and unified from the incoming NCBI XML feed.

Object sources: The main design premise of `UFO` Repository is that it should be able to import, unify, and standardize differently structured objects from wide variety of sources:

`Schemas:` Clearly, an excellent source of object metadata are (xml, relational) schemas. Here, the entities are discovered, and unified with existing `UFOs` in the repository. For instance, OAGIS [1] provides a large collection of schemas for many domains that have objects in abundance.

`HTML:` Web contains many objects hidden within HTML, which if appropriately wrapped however, can provide a rich set of objects and instances. For instance, NCBI, AD&FTD[2], Swiss-prot (protein knowledgebase) are just a few of many biological databases on the Web that provide online access to their data. In one of our demo scenarios, we used wrapped objects to create `U-Protein, U-cDNA, U-Genome,`

[2]Alzheimer Disease & Frontotemporal Dementia



```
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" attributeFormDefault="unqualified">
  - <xs:element name="UFO_REPOSITORY">
    - <xs:complexType>
      - <xs:sequence>
        + <xs:element minOccurs="0" maxOccurs="unbounded" name="client">
        - <xs:element minOccurs="0" maxOccurs="unbounded" name="client_address">
          - <xs:complexType>
            - <xs:all>
                <xs:element name="client_id" xs_type="xs:ID" />
                <xs:element name="address_id" xs_type="xs:ID" />
              </xs:all>
            </xs:complexType>
          </xs:element>
        + <xs:element minOccurs="0" maxOccurs="unbounded" name="address">
        + <xs:element minOccurs="0" maxOccurs="unbounded" name="loan">
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

Figure 4: UFO schema export

`U-Sequence` `UFOs`, and demonstrate automatic disease diagnosis.

`Freebase` [2] is a general-purpose collection of structured data on a wide-variety of topics. The domains include *music*, *sport*, *publishing*, etc. It is created manually by a huge community effort and supports queries returning *JSON* results.

`XML Mashups:` *Web 2.0* offers a wide variety of differently structured mashups that poses many challenges for Information Management Community. By leveraging already accumulated `UFOs` the repository is able to automatically discover objects inside incoming XML feeds; extract, and unify them with the appropriate `UFOs`; convert the incoming XML feed into *unified* XML where original objects are replaced with standard `UFOs`.

UFO Factory: Since the metadata managed by the `UFO Repository` is very heterogeneous and comes from many sources it is crucial that the representation differences are detected and reconciled seamlessly. Observe that reconciliation is much more challenging in our context, because the metadata is represented rather differently in all the sources.

Clio `UFO Factory` is a tool we developed on top of *Clio* to offer GUI for semi-automatic object discovery [9]. Figure 3 illustrates semi-automatic discovery of *Genome* from NCBI (on the left) and its unification into `U-Genome` (on the right). After unification is done, the updated UFOs are used to detect objects in incoming feeds and serve as standardized interface for user queries.

Storage: The repository stores `UFOs` and data in IBM DB2 pureXML. The `UFOs` and data instances are represented in XML and can be queried with XQuery. *Clio* UFO Factory (Figure 3) exports UFOs after discovery and unification is done as well as loads existing `UFOs` to discover familiar objects (Figure 1).

3. FORWARD-LOOKING STATEMENTS

`UFO Repository` currently has two main applications. The first is *object-oriented* data integration. It simplifies access to information inside raw data hidden behind its different representation barrier. The second is *object-oriented* data exchange. Data objects in different sources can be mapped and exchanged through `UFOs` easier. This is currently the subject of ongoing work.

We believe `UFO Repository` is only a start on the way to *modular* and *fluid* data world, where the problem to retrieve and access needed data from multiple sources is significantly alleviated or no longer exists.

4. REFERENCES

[1] Oagis: http://www.oasis-open.org.

[2] K. Bollacker and T. Sturge. Freebase: A collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.

[3] S. Ding, G. Cong, C.-Y. Lin, and X. Zhu. Extracting question-context-answer triples from online forums. In *ACL*, 2008.

[4] X. Dong and A. Halevy. Indexing dataspaces. In *SIGMOD*, 2007.

[5] T. J. Green, G. Karvounarakis, N. E. Taylor, O. Biton, Z. G. Ives, and V. Tannen. Orchestra: facilitating collaborative data sharing. In *SIGMOD*, 2007.

[6] M. N. Gubanov. Distributed component architecture and a library for web xml applications. In *TELEMATIKA*, 2002.

[7] M. N. Gubanov and P. A. Bernstein. Structural text search and comparison using automatically extracted schema. In *WebDB*, 2006.

[8] M. N. Gubanov, P. A. Bernstein, and A. Moshchuk. Model management engine for data integration with reverse-engineering support. In *ICDE*, 2008.

[9] L. M. Haas, M. A. Hernández, H. Ho, L. Popa, and M. Roth. Clio grows up: from research prototype to industrial tool. In *SIGMOD*, 2005.

[10] L. M. Haas, E. T. Lin, and M. A. Roth. Data integration through database federation. *IBM Syst. J.*, 41(4), 2002.

[11] M. Hernández, S. Dessloch, R. Wisnesky, A. Radwan, and J. Zhou. Orchid: Integrating schema mapping and etl. *ICDE*, 2008.

[12] L. Lim, H. Wang, and M. Wang. Semantic data management: Towards querying data with their meaning. 2007.

[13] L. Lim, H. Wang, and M. Wang. Unifying data and domain knowledge using virtual views. In *VLDB*, 2007.

[14] J. Madhavan, P. A. Bernstein, K. Chen, A. Halevy, and P. Shenoy. Corpus-based schema matching. In *IJCAI*, 2003.

[15] S. Melnik, E. Rahm, and P. A. Bernstein. Rondo: a programming platform for generic model management. In *SIGMOD*, 2003.

[16] M. Mezini and K. Ostermann. Integrating independent components with on-demand remodularization. In *OOPSLA*, 2002.

[17] P. Mork, R. Shaker, and P. Tarczy-Hornoch. The multiple roles of ontologies in the biomediator data integration system. In *DILS*, 2005.

[18] C. Re, J. F. Brinkley, K. P. Hinshaw, and D. Suciu. Distributed xquery. In *IIWeb*, 2004.

[19] M. Roth, M. A. Hernández, P. Coulthard, L. Yan, L. Popa, H. C.-T. Ho, and C. C. Salter. Xml mapping technology: making connections in an xml-centric world. *IBM Syst. J.*, 45(2), 2006.

[20] S. Sekine. On-demand information extraction. In *COLING/ACL*, 2006.

[21] M. Shaw, L. T. Detwiler, J. F. Brinkley, and D. Suciu. Generating application ontologies from reference ontologies. In *AMIA*, 2008.

[22] D. Simmen, M. Altinel, V. Markl, S. Padmanabhan, and A. Singh. Damia: Data mashups for intranet applications. In *SIGMOD*, 2008.

[23] Y. Sismanis, B. Reinwald, and H. Pirahesh. Document-centric olap in the schema-chaos world. In *BIRTE*, 2006.

[24] O. Udrea, L. Getoor, and R. J. Miller. Leveraging data and structure in ontology integration. In *SIGMOD*, 2007.

[25] D. Ungar and R. B. Smith. Self: The power of simplicity. In *OOPSLA*, 1987.

[26] C. Yu and H. V. Jagadish. Querying complex structured databases. In *VLDB*, 2007.