Implementing a modern API for CDS/ISIS, a classic semistructured NoSQL database

Luciano G. S. Ramalho

Departamento de Biblioteconomia e Documentação, Escola de Comunicações e Artes
Universidade de São Paulo – São Paulo, SP – Brazil

BIREME/PAHO/WHO, Latin-American and Caribbean Center on Health Sciences Information

luciano.ramalho@bireme.org

Abstract. CDS/ISIS is a family of semistructured, "NoSQL" database products created by Unesco and used at the SciELO digital library as well as thousands of academic libraries since the 1980s. This paper describes how a database-independent API is being developed to allow the LILACS bibliographic methodology created by BIREME to be implemented over CDS/ISIS and modern semistructured databases such as MongoDB and CouchDB.

1. INTRODUCTION

Much of the data on the Web is organized in hierarchies and is multimedia in nature, therefore consisting of several parts that nevertheless must be treated as a unified whole. Modeling such data in normalized relational databases is difficult, and may result in performance problems due to the cost of joins. De-normalization and horizontal scaling help meet the scalability challenges of the Web, but are non-trivial to implement on relational databases originally designed to enforce consistency at all times [Eure, 2009]. These issues have prompted renewed interest in non-relational databases, collectively known since 2009 by the informal "NoSQL" term.

Apache Cassandra, Redis, Apache CouchDB and MongoDB are some examples of Open Source NoSQL databases. Cassandra was created by Facebook and is also used by Twitter and Digg. Two large-scale deployments of proprietary NoSQL databases are BigTable, by Google, and Dynamo by Amazon.com, used by its S3 storage web service.

While most NoSQL products mentioned have been released in the past 10 years, CDS/ISIS is a non-relational database created by Unesco in the 1960s, ported to PC/DOS microcomputers in the 1980s and then to the Windows and Linux operating systems. Today it is used by thousands of libraries to run their on-line catalogs.

The C language port of CDS/ISIS for Linux and Windows, called CISIS, is used to run two of the largest on-line digital libraries in Latin America, LILACS[1] and SciELO[2], and is deployed in hundreds of cooperating scientific information centers, handling more than 18 million bibliographic records. CISIS was developed and is maintained, since the early 1990s, by BIREME/PAHO/WHO, the Latin American and Caribbean Center on Health Sciences Information, a specialized

[1] LILACS: Latin American and Caribbean Literature in Health Sciences), part of the Virtual Health Library coordinated by BIREME/PAHO/WHO
[2] SciELO: Scientific Electronic Library Online), a partnership between FAPESP, CNPq, FapUNIFESP and BIREME/PAHO/WHO

center of the Pan American Health Organization/World Health Organization, located in São Paulo, Brazil, at the main UNIFESP[3] campus.

2. THE DATA MODEL OF CDS/ISIS

Beyond the informal NoSQL descriptor, CDS/ISIS is more precisely described as a document database implementing the data model of the ISO-2709 Format for Information Exchange standard, which in turn reflects the data model of the Library of Congress MARC[4] format for the representation of bibliographic records.

2.1. Relaxing the restrictions of the First Normal Form

MARC fields can be multivalued and composite. Therefore, the MARC data model violates the First Normal Form (1NF) atomicity requirement: "values in the domains on which each relation is defined are required to be atomic with respect to the DBMS" [Codd, 1990]. Relaxing the 1NF atomicity requirement greatly simplifies the development of bibliographic databases and the exchange of records among libraries.

> *[...] it is interesting to note the absurdity of the NRM [Normalized Relational Model]: if a book has 3 authors and 5 subjects, it will be necessary to represent it in the NRM through a row in the Books table, plus 3 rows in the AuthorNames (which would implement the corresponding multivalued attribute) and another 5 in the Subjects, for a total of 9 rows in three separate tables [...] yet what we hold in our hands in the real world is just one book [...] [Setzer, 2005]* [5]

2.2. The semistructured data model

In the database theory literature, the data model most similar to the CDS/ISIS database structure is the semistructured data model [Abiteboul, 1999].

> *The semi-structured data model is designed as an evolution of the relational data model that allows the representation of data with a flexible structure. Some items may have missing attributes, others may have extra attributes, some items may have two ore more occurrences of the same attribute. The type of an attribute is also flexible: it may be an atomic value or it may be another record or collection. Moreover, collections may be heterogeneous, i.e., they may contain items with different structures. The semi-structured data model is a self-describing data model, in which the data values and the schema components co-exist. [Liu, 2009]*

In a CDS/ISIS record, fields are identified by numeric tags from 1 to 32767. The meaning of each numeric tag is defined by the application. For instance, in the LILACS methodology, field 10 is Author and field 12 is Title. The fact that each field occurrence is preceded by its tag number means that fields may be omitted or repeated as needed.

---

[3] *Universidade Federal de São Paulo*
[4] *MAchine-Readable Cataloging*
[5] *Translated from the Brazilian Portuguese book by Valdemar Setzer "Bancos de dados" (1st ed., 2005)*

Multivalued attributes are represented by repeating field tags. In LILACS, field 10 (Author) may be repeated, and field 12 (Title) may be repeated if the title is available in multiple languages.

2.3. CDS/ISIS data model limitations

CDS/ISIS has restrictions that are not in the general semistructured data model:

1. There is only one level of nesting: fields may contain subfields but subfields may not contain sub-subfields;
2. Subfields must be labeled from A to Z and from 0 to 9, therefore a field can only contain 36 subfields;

In addition to the 36 subfields, a field may have content that is outside of any subfield. Because the syntax for delimiting subfields uses only one marker at the start of the subfield, data outside of the subfields must appear before the first subfield.

10 «Lewis Carroll^1USP^2ECA^pBrasil^cSão Paulo^rEditor»

Figure 1: Sample of an author field (10) showing subfields labeled "1", "2", "p", "c" and "r". The "p" subfield describes the country of the author, in this case "Brasil". The name of the author, "Lewis Carroll" is not preceded by a subfield delimiter, therefore is not part of any subfield.

2.4. Schema definition in CDS/ISIS

Like all semistructured databases, CDS/ISIS is "schemaless" in the sense that a database instance does not have a predefined schema. But in practice, CDS/ISIS databases usually have at least a documented, human-readable schema, describing the semantics of each field tag, and defining which field tags are mandatory or repeatable and the subfields that may be used inside each field tag. A rich example of such a schema is the LILACS Data Dictionary [BIREME, 2008]. Some CDS/ISIS applications, like WinISIS and ABCD, allow the user to define a schema in a Field Definition Table (FDT), which is used to generate data entry forms.

2.5. Handling fields and subfields in the ISIS Formatting Language (IFL)

The CDS/ISIS family has a data extraction language called the "Formatting Language", henceforth the IFL. The IFL is used primarily to generate displays and reports of database records and to generate indexes supporting efficient retrieval.

Table 1: Examples of ISIS Formatting Language expressions

IFL expression	Result	Comment
v10[1]	Lewis Carroll^1USP^2ECA^ pBrasil^cSãoPaulo^rEditor	first occurrence of field tag 10
v10^p[1]	Brasil	content of p subfield in first occurrence of field tag 10
v10^p[1]*0.3	Bra	substring starting at offset zero with length 3 of p subfield in field tag 10

The IFL is flexible but its syntax is terse. Readability is also hindered by the fact that field tags are always numeric, subfield labels are limited to one alphanumeric character, and the language lacks abstraction mechanisms to allow the user to define functions or assign friendlier identifiers to expression results.

3. ISIS-DM, A MODERN API FOR SCHEMA DEFINITION AND ACCESS

In 2007, BIREME/PAHO/WHO started the development of the ISIS Network Based Platform (ISIS-NBP) project [BIREME, 2010], which aims to re-implement the functionality of CISIS in the Python programming language using a service oriented architecture. ISIS-NBP is free software, distributed under the GNU General Public License. Most of the effort from 2007 to 2009 was devoted to the CISIS binary-compatible storage layer and the Formatting Language interpreter.

A recent development within ISIS-NBP is ISIS-DM, the ISIS Data Model application programming interface. The ISIS-DM effort aims to provide an API for:

1. Schema definition through classes, similar to modern object persistence frameworks;
2. Data extraction using operators and methods similar to modern collection and string handling APIs;

In other words, the goal is to allow programmers to express constraints and functionality similar to those of the CDS/ISIS Field Definition Table and the CDS/ISIS Formatting Language, but in contemporary, object-oriented programming languages.

The API aims to be database-independent. Objects defined in the ISIS-DM should be easily persisted in CDS/ISIS and other semistructured databases.

3.1. Schema definition

Listing 1 shows an example of a very simple ISIS-DM schema definition in Python. A schema called Book is being defined as a subclass of CheckedModel, an ISIS-DM class which implements schema constraints and validation. Field types are implemented as

Python descriptors, which control instance attribute access.

```
class Book(CheckedModel):
    title = SingularProperty(required=True, subfields='s')
    creators = PluralProperty(required=True, subfields='yr')
    pages = NumberProperty(validator=gt_zero)
```

Listing 1: Simple schema definition showing required and repeatable fields.

Non-repeatable string fields are instances of SingularProperty; repeatable fields are instances of PluralProperty. Both types may have subfields, which are always optional. The API checks whether subfields entered match those in the field definition. Internally, string data is converted to instances of a CompositeField class which implements several subfield access methods and operators, including iterators over the subfield keys and values.

A NumberProperty is non-repeatable and may only contain integers or floatingpoint numbers. For any type of property, a validator function may be specified, returning an error message when the field value fails a test.

```
>>> book = Book(title='The Annotated Alice^sDefinitive Edition',
...             pages=352,
...             creators=['Lewis Carroll^y1832-1898^rAuthor',
...                       'John Tenniel^rIllustrator',
...                       'Martin Gardner^y1914-2010^rEditor'])
>>> book.title
u'The Annotated Alice^sDefinitive Edition'
>>> print book.title.s
Definitive Edition
>>> print book.creators[0].y
1832-1898
>>> for creator in book.creators:
...     print '%-12s: %s' % (creator.r, creator[0])
Author      : Lewis Carroll
Illustrator : John Tenniel
Editor      : Martin Gardner
```

Listing 2: Instance creation and attribute access. Fields may be handled as a whole or by subfield, output is formatted with the Python % formatting operator.

The CheckedModel.check instance method verifies the presence of required fields (see listing 3). It returns a dictionary of attribute names and error messages, useful for displaying in Web or GUI forms. Validator functions are applied upon instance creation or attribute change, and generate exceptions with custom messages.

```
>>> b1 = Book()
>>> b1.check()
{'creators': u'Required value missing.', 'title': u'Required
value missing.'}
>>> b2 = Book(title='The Annotated Alice^sDefinitive Edition',
...           pages=352)
>>> b2.check()
{'creators': u'Required value missing.'}
>>> b2.creators = ['Lewis Carroll']
>>> b2.pages = 0
Traceback (most recent call last):
  ...
ValueError: <pages> must be greater than zero
```

Listing 3: Demonstration of .check method and validator functions.

4. CONCLUSION

4.1. Lessons learned

Using modern language features such as metaclasses, operator overloading and object properties (Python descriptors), part of the functionality of the CDS/ISIS Field Definition Table and of the Formatting Language was implemented in less than 700 lines of code, including automated tests (doctests), in the isisdm.properties package composed of three Python modules.[6]

[6] Source code available at the BIREME/PAHO/WHO RedDes public repository: http://reddes.bvsalud.org/projects/isisnbp/browser/isisdm/properties or via http://bit.ly/isisdmrepo

The CheckedModel class and the various property classes provide a means of defining the schema and validating records in a way that will allow automatic generation of data entry forms. To allow ordered retrieval of field definitions it was necessary to define a special metaclass for an OrderedModel, which collaborates with an OrderedProperty descriptor class. OrderedModel is the superclass of CheckedModel, and OrderedProperty is the superclass of all properties shown here.

Besides the functionality shown here, features already implemented include:

- Controlled-vocabulary checking via lists provided at field definition;
- Alternate syntaxes to access subfields defined with numeric keys;
- Various iterators for flexible field and subfield access;

The CDS/ISIS Formatting Language implements both presentation and data extraction functions. Many of its presentation features were designed for fixed width displays with monospaced fonts, while today all CISIS applications in production at BIREME/ PAHO/WHO use a Web interface where those formatting assumptions are no longer valid. On the other hand, the richer data extraction API made possible with Python and other modern languages allow more convenient handling of fields and subfields in the presentation layer, where sometimes data will be rendered directly as HTML elements and other times as JSON or XML for delivery and display via Ajax.

4.2. Future work

The ISIS-DM API aims to be language-independent. The current implementation is in Python, but a PHP version would be valuable, because of its popularity and its use in current BIREME/PAHO/WHO systems which are widely distributed. Also, storage drivers should be developed for ISIS-NBP, CISIS, and at least a couple of the most popular semistructured databases, such as Cassandra, CouchDB, MongoDB and Google Datastore. The CouchDB implementation is planned as part of a graduation monograph for a bachelors degree in Library Sciences which the author is currently working on.

REFERENCES

Abiteboul, S.; Buneman P.; Suciu, D. (1999), Data on the web: from relations to semistructured data and XML, San Francisco: Morgan Kaufmann, 1999.

BIREME (2008), Diccionario de datos del modelo LILACS Versión 1.6a , São Paulo: BIREME/PAHO/WHO, 2008.

BIREME (2010), ISIS-NBP Project Repository, São Paulo: BIREME/PAHO/WHO http://reddes.bvsalud.org/projects/isisnbp/, accessed June 2010.

Codd, E. F. (1990) The Relational Model for Database Management Version 2, Reading, MA: Addison-Wesley, 1990.

Eure, I. (2009) “Looking to the future with Cassandra”, In: Digg Technology Blog http://about.digg.com/blog/looking-future-cassandra, September 9, 2009.

Liu, L.; Özsu, M. T. (2009) Encyclopedia of database systems : Springer, 2009

Setzer, V.; Corrêa da Silva, F. (2005) “Bancos de dados aprenda o que são, melhore seu conhecimento, construa os seus”, 1ª ed. São Paulo: Edgard Blücher, 2005.