XI Encontro Nacional de Pesquisa em Ciência da Informação
Inovação e inclusão social: questões contemporâneas da informação
Rio de Janeiro, 25 a 28 de outubro de 2010

GT 8: Tecnologia e Informação
Modalidade de apresentação: Comunicação Oral

MODELOS DE DADOS PARA BASES BIBLIOGRÁFICAS: LEGADO, PADRÕES E ALTERNATIVAS

Marcos Luiz MUCHERONI[1]
Universidade de São Paulo
Luciano G. S. RAMALHO[2]
BIREME

Resumo: A escolha de um modelo de dados e de um Sistema Gerenciador de Banco de Dados (SGBD) deve ser consequência da natureza dos dados que se deseja armazenar e dos tipos de operações que se pretende realizar com eles. Neste artigo, comparamos brevemente o modelo de dados relacional, predominante no mercado, com o modelo de dados semiestruturado, característico de padrões como MARC, ISO-2709, CDS/ISIS. À luz das necessidades de Sistemas de Gerenciamento Eletrônico de Dados (SGED) para bases de dados bibliográficas, constatamos que o uso continuado da família ISIS nas bibliotecas se justifica, e exploramos o potencial de novos sistemas de bancos de dados semiestruturados, como CouchDB e MongoDB, em aplicações em catalogação, documentação e arquivística.

Palavras-chave: Sistemas de informação, bancos de dados, relacional, semiestruturado

---

[1] Prof. Dr. Marcos Luiz Mucheroni, professor de pós-graduação em Ciência da Informação, CBD-ECA-USP.

[2] Bacharelando em Biblioteconomia na Escola de Comunicação e Artes (ECA) da Universidade de São Paulo (USP), e supervisor de desenvolvimento de sistemas na BIREME/OPAS/OMS

1. Introdução

O modelo relacional de dados, com sua sólida base teórica e forte apoio da indústria domina as aplicações de banco de dados nas últimas duas décadas, incluindo um conjunto amplo de EDI (Electronic Data Interchange).

EDIs fazem parte de um conjunto de Sistemas de Gerenciamento Eletrônico de Dados (SGED) onde ocorrem integração e intercâmbio de dados, com uso de sistemas de coleta e linguagens de marcação como XML (Hasselbring, 2000).

Uma forma de EDIs, que é o gerenciamento de bases de dados bibliográficas, tem na família de sistemas ISIS um padrão que continua sendo largamente utilizado, principalmente em países em desenvolvimento. Tais sistemas não se enquadram no modelo relacional que muitos EDIs utilizam: são exemplos do modelo de dados semiestruturados.

Enquanto a difusão do modelo relacional foi acompanhada de uma importante produção de pesquisas e bibliografia de apoio, o modelo de dados do ISIS não despertou o mesmo interesse acadêmico.

Porém, a partir de 1995, grupos de pesquisadores como o U**Penn Database Research Group (University of Pennsylvania, 2010) e a equipe do projeto Lore, na Universidade Stanford (Infolab, 2010) formalizaram o modelo de dados semiestruturado, do qual os modelos de dados dos padrões MARC, ISO-2709 e ISIS são casos especiais. Dezenas de artigos foram produzidos por estes pesquisadores, e o livro seminal *Data on The Web: From Relations to Semistructured Data and XML* (Abiteboul, 1999), com contribuições de ambos grupos. Embora o foco da pesquisa no modelo semiestruturado tenha se voltado no final da década de 1990 para as aplicações de XML, a teoria é útil também para o estudo dos modelos de dados utilizados em bases bibliográficas. Segundo Abiteboul (1999), uma das motivações para a formalização do modelo de dados semiestruturado foi justamente facilitar o intercâmbio de dados entre bases heterogêneas com esquemas de dados distintos, que é um dos problemas centrais dos Sistemas de Gerenciamento Eletrônico de Dados.**

2. Terminologia: bancos de dados, bases de dados e termos relacionados

Ao escrever sobre "bancos de dados", uma dificuldade inicial é que o termo é empregado com sentidos diferentes, mesmo em textos acadêmicos, e às vezes na mesma oração. Uma modesta contribuição deste trabalho é tentar introduzir alternativas ao termo "banco de dados" para suas diferentes acepções. O que estamos buscando são definições básicas, e neste sentido justifica-se começar pelas definições do dicionário Aurélio (Ferreira, 2009):

> **banco de dados:** 1. Coleção organizada e inter-relacionada de dados persistentes; base de dados. 2. Programa especializado em gerenciar um banco de dados (1).

Portanto, "banco de dados" pode ser sinônimo de base de dados, no sentido de "coleção de dados", ou um programa que gerencia tal coleção. Mas, na prática, os dois conceitos são bem distintos. No mesmo dicionário, o verbete "base de dados" tem apenas uma definição:

> **base de dados:** 1. Banco de dados (1).

Ou seja, para os lexicógrafos do Aurélio, base de dados é sempre uma coleção de dados (definição 1 de banco de dados). Assim, evitaremos ambiguidade usando sempre o termo base de dados neste sentido. Por exemplo: "LILACS[3] é uma base de dados que reúne a produção científica em saúde da América Latina e do Caribe".

Na terminologia técnica de administração de sistemas de banco de dados, há ainda mais um significado para "banco de dados": pode ser um conjunto de tabelas inter-relacionadas, identificado por um nome e configurado com determinadas regras de controle de acesso.[4]

Para diferenciar os três sentidos de "banco de dados", adotaremos aqui os seguintes termos:

---

[3] LILACS: Latin American and Caribbean Literature in Health Sciences, catálogo cooperativo que faz parte da rede de Bibliotecas Virtuais em Saúde, coordenada pela BIREME/OPAS/OMS.

[4] Veja por exemplo a documentação online do MySQL: <http://dev.mysql.com/doc/refman/4.1/pt/database-use.html>. Acesso em 4 abr. 2010.

base de dados: coleção de dados, conforme a definição 1 do Aurélio.

sistema de banco de dados: software integrado ou conjunto de componentes de software para manipular bases de dados; definição 2 do Aurélio.

objeto banco de dados: conjunto nomeado de tabelas ou coleções de dados em um sistema de banco de dados.

Além disso, ao discutir diferentes sistemas de bancos de dados (definição 2 do Aurélio), usaremos termos mais específicos:

sistema gerenciador de banco de dados (SGBD): sistema de banco de dados projetado para permitir e controlar o acesso e a manipulação dos dados por múltiplos processos ou usuários remotos via rede, simultaneamente, garantindo sua consistência contra operações conflitantes e mesmo sob certas condições de falha. Por exemplo, o PostgreSQL é um sistema gerenciador de banco de dados relacional, e o Apache CouchDB é um SGBD semiestruturado.

sistema eletrônico de gerenciamento de dados (SGED): sistema que garante a interoperabilidade sintática dos dados a partir de alguma linguagem padrão ou por formatos abertos (Martinez Usero e Lara Navarra, 2007).

motor de banco de dados (*database engine*): componente de software projetado para ser embutido em um sistema maior, que permite o acesso a um objeto banco de dados. Um motor de banco de dados normalmente não faz controle de acesso nem gerencia acessos concorrentes ou remotos, sendo estas, funções do aplicativo no qual o motor está embutido. Por exemplo, o SQLite[5] é um motor de banco de dados relacional e CISIS é um motor de banco de dados semiestruturado.

3. O modelo relacional e registros bibliográficos

No modelo de dados relacional, a informação é organizada em relações (ou tabelas), que são conjuntos de tuplas (ou linhas), nas quais cada tupla corresponde a um registro. Por definição, os elementos de uma tupla não podem ser repetidos, e todas as tuplas de

[5] SQLite database engine: <http://www.sqlite.org/>, acesso em 24 ago. 2010.

uma relação têm a mesma estrutura, ou seja, todos os campos ocorrem em todos os registros, e não há campos repetitivos. Além disso, “values in the domains on which each relation is defined are required to be atomic with respect to the DBMS” (Codd, 1990), ou seja, o conteúdo de cada campo é indivisível. Esta última restrição é a chamada Primeira Forma Normal (1NF, na sigla em inglês). As regras de normalização visam eliminar redundância e assim evitar anomalias na atualização dos dados.

Em conjunto, as restrições do modelo relacional e as regras de normalização obrigam a divisão dos campos de um registro bibliográfico em várias tabelas:

> [...] seria interessante notar o absurdo do padrão do MRN [Modelo Relacional Normalizado]: se um livro tiver 3 autores e 5 assuntos, será necessário representá-lo no MRN por meio de uma linha na tabela Livros, mais 3 na Nomes-de-autores (que implementaria o atributo multivalorado correspondente) e mais 5 na de Assuntos, num total de 9 linhas em três tabelas distintas [...]. Mas o que se vê e e pega-se na mão no mundo real é um livro só, e não um picadinho de livro! (Setzer, 2005)

Esta representação de um único registro bibliográfico em vários registros no sistema de banco de dados relacional complica o intercâmbio de dados, e portanto dificulta a catalogação cooperativa, uma prática essencial para a construção de grandes bases de dados bibliográficas como a LILACS.

4. Bases bibliográficas e o modelo de dados semiestruturado

O esquema de dados da base bibliográfica LILACS é definido no Dicionário de Dados da Metodologia LILACS (BIREME, 2008a). Neste esquema, constam campos repetitivos, campos opcionais, e campos compostos por subcampos, conceitos não aderentes ao paradigma relacional de Codd, porém presentes nos padrões de registros MARC e ISO-2709, que têm suas origens na Biblioteca do Congresso dos EUA.

O modelo de dados implementado nos motores de bancos de dados da família ISIS é derivado diretamente do ISO-2709. Na literatura de teoria de banco de dados, o modelo de dados mais próximo do padrão ISO-2709 é o modelo semiestruturado **(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)[6]

Em um registro ISIS, cada campo é identificado por um marcador (tag) numérico entre 1 e 32767. O significado de cada tag é definido pela aplicação. Por exemplo, no dicionário de dados da LILACS, em um registro de nível analítico, o campo 10 é para o nome do autor individual, e o campo 12 é para o título do artigo. Uma vez que cada ocorrência de um campo é precedida de um tag, é possível repetir ou omitir campos, ao contrário da estrutura rígida do modelo relacional. Na metodologia LILACS o campo 10 (autor) pode ser repetido ou omitido. Mesmo o campo 12 (título) pode ser repetido, permitindo a catalogação multilíngue, com o título em vários idiomas.

---

[6] Tradução: O modelo semiestruturado foi projetado como uma evolução do modelo de dados relacional, permitindo a representação de dados com estrutura flexível. Alguns itens podem ter atributos a menos, outros podem ter atributos a mais, alguns itens podem ter duas ou mais ocorrências do mesmo atributo. O tipo de um atributo também é flexível: pode ser um valor atômico ou pide ser outro registro ou coleção. Além disso, as coleções podem ser heterogêneas, ou seja, pode conter itens com diferentes estruturas. O modelo de dados semiestruturado é um modelo autodescritivo, no qual os valores de dados e os componentes do esquema coexistem.

O modelo semiestruturado, no entanto, é mais geral que o modelo de dados descrito na norma ISO-2709 e implementado nos sistemas ISIS. No modelo semiestruturado, o valor de um campo pode ser um valor simples ou outra coleção de valores, recursivamente. Em outras palavras, é possível representar campos, subcampos, subsubcampos e assim sucessivamente, sem um limite teórico de aninhamento. O modelo do ISIS é limitado a um nível de aninhamento: existem apenas campos e subcampos. Além disso, no modelo semiestruturado não há limite para os rótulos que identificam os subcampos, mas no modelo ISIS o delimitador de subcampo é o caractere ^ seguido de um caractere de A a Z (sem distinção entre maiúsculas e minúsculas) ou um dígito de 0 a 9, o que efetivamente limita o número de subcampos a 36. Finalmente, a sintaxe de marcação do ISIS permite que o marcador do primeiro subcampo seja omitido,

```
10 «Lewis Carroll^1USP^2ECA^pBrasil^cSão Paulo^rEditor»
10 «John Tenniel^1USP^2IME^pBrasil^cSão Paulo^rIlustrador»
12 «Jaguadarte^sEstudo comparado de traduções»
```

desde que este subcampo seja obrigatório[7] (figura 1).

a.

Figura 1: *Duas ocorrências do campo autor (10) com subcampos rotulados "1", "2", "p", "c" and "r". O subcampo "p" descreve o país do autor, nestes casos "Brasil". O subcampo contendo o nome de cada autor, p. ex. "Lewis Carroll", não é precedido de um delimitador de subcampo, uma simplificação permitida quando o subcampo está sempre presente. No exemplo, o campo 12, título, tem um subcampo "s" para o subtítulo.*

5. Sistemas de bancos de dados semiestruturados

Enquanto a demanda do mercado estimulou a criação de inúmeros SGBD e motores de bancos de dados relacionais, o campo dos sistemas semiestruturados não foi tão produtivo, até recentemente. Especificamente a família ISIS ainda não inclui um verdadeiro SGBD, mas apenas implementações como CISIS e J-ISIS, que são motores de bancos de dados.

[7] UNESCO: Mini-micro CDS-ISIS Reference Manual, version 2.3, p. 33

Porém, a partir do ano 2000 grandes sites usuários de sistemas bancos de dados proprietários, como Amazon.com, Google e Facebook, vêm revelando que, para obter maior desempenho e escalabilidade horizontal, estão utilizando sistemas não-relacionais em sua infraestrutura. Em particular, tanto o Google quanto a Amazon.com oferecem hoje gerenciadores de bancos de dados semiestruturados hospedados “na nuvem” (ou seja, em infraestruturas distribuídas com provisionamento sob demanda): são eles o Google Datastore, parte do serviço de hospedagem Google App Engine (Google, 2010), e o Amazon SimpleDB, um componente da plataforma AWS (Amazon.com, 2010).

Mais recentemente, projetos de gerenciadores de bancos de dados semiestruturados Open Source vêm lançando versões estáveis, prontas para uso em produção. Dois exemplos são o Apache CouchDB (Anderson, 2010), projeto incubado pela Apache Foundation, e o MongoDB, criado pela empresa 10Gen (10gen, 2010).

O modelo de dados do CouchDB é o mesmo do padrão JSON (Javascript Object Notation), e o MongoDB utiliza um modelo de dados semelhante, denominado BSON (ou Binary JSON). Ambos representam cada registro como um dicionário: uma coleção de pares chave-valor. Os valores dos campos podem ser dados simples ou coleções aninhadas, como listas e dicionários. Estas estruturas, combinadas com algumas convenções, permitem representar qualquer registro ISIS (figura 2).

```
{"10":
    [  {"_":"Lewis Carroll",
        "1":"USP",
        "2":"ECA",
        "p":"Brasil",
        "c":"São Paulo",
        "r":Editor"},
       {"_":"John Tenniel"
        "1":"USP",
        "2":"IME",
        "p":"Brasil",
        "c":"São Paulo",
        "r":"Ilustrador"}],
 "12":
    [  {"_":"Jaguadarte"
        "s":"Estudo comparado de traduções"}],
}
```

Figura 2: *Os mesmos dados da figura 1, representados no formato JSON. O registro é um dicionário, delimitado por chaves, neste caso contendo dois tags, correspondendo aos campos 10 e 12. Todas as ocorrências de cada campo aparecem como itens de uma lista, delimitada por colchetes. Cada ocorrência é também um dicionário, com marcadores correspondentes a cada subcampo, e um marcador especial "_" (caractere de sublinha) para representar o primeiro subcampo. Vale notar que o campo 12, embora neste exemplo não apareça repetido, é assim mesmo representado como uma lista contendo apenas um item, o dicionário com as chaves "_" e "s". A representação de todos os campos como listas é necessária na ausência de um esquema de dados predefinido, porque no modelo ISIS qualquer campo em princípio pode ser repetitivo.*

Uma representação JSON mais simples, e ao mesmo tempo mais próxima do formato nativo das bases ISIS é apresentada na Figura 3. Nesta representação os subcampos são armazenados concatenados juntos. É assim que eles são armazenados fisicamente nos arquivos em formato MST do ISIS e nos arquivos em formato ISO-2709.

Uma desvantagem do modelo semiestruturado em relação ao modelo relacional é que, para manter a unidade dos registros, existe a tendência de se duplicar dados. Por exemplo, em um registro bibliográfico semiestruturado os nomes dos autores aparecem embutidos no próprio registro. Isso significa que autores prolíficos terão seus nomes repetidos em muitos registros bibliográficos. No modelo relacional normalizado, os múltiplos registros conteriam apenas uma referência (uma chave estrangeira) apontando para o registro do autor. Ao evitar a duplicação dos dados, o modelo relacional

normalizado reduz a possibilidade de haverem inconsistência nos dados em caso de uma alteração. Por exemplo, se um autor muda de nome, esta alteração teria que ser feita várias vezes numa base semiestruturada, mas apenas uma vez em uma base relacional.

Ao se avaliar esta desvantagem, entretanto, é importante refletir sobre a prevalência da situação descrita acima. Na prática, registros bibliográficos raramente sofrem atualizações, diferentemente dos registros de estoques, contas, recursos etc. que caracterizam as aplicações comerciais para as quais os bancos de dados relacionais foram projetados.

```
{"10":
   ["Lewis Carroll^1USP^2ECA^pBrasil^São Paulo^rEditor",
    "John Tenniel^1USP^2IME^pBrasi^cSão Paulo^rIlustrador"],
 "12":
   ["Jaguadarte^sEstudo comparado de traduções"],
}
```

Figura 3: *Os dados da figura 2, representados no formato ISIS-JSON[8]. Neste esquema cada ocorrência de campo é um string, com os marcadores de subcampo preservados no formato original das bases ISIS, e não separados como itens em um dicionário. Este esquema é mais simples e compacto, mas tem um maior custo computacional quando se precisa extrair os subcampos.*

6. O projeto ISIS-NBP e o ISIS Data Model

Buscando incorporar à plataforma ISIS novas arquiteturas de sistemas baseados em redes, a BIREME/OPAS/OMS vem desenvolvendo desde 2007 o projeto ISIS-NBP: ISIS Network Based Platform (BIREME, 2010).

No âmbito do ISIS-NBP, um desenvolvimento recente é o ISIS-DM, ISIS Data Model (Ramalho, 2010). Tomando como ponto de partida as lições aprendidas com o uso de frameworks Web orientados a objetos, como Django e Symfony, o projeto ISIS-DM busca desenvolver ferramentas e metodologias para a conversão de bases de dados ISIS para

[8] ISIS-JSON é o esquema de dados JSON adotado para exportação de bases ISIS no projeto ISIS-NBP da BIREME, documentado em: <http://reddes.bvsaude.org/projects/isisnbp/wiki/Tools>. O programa isis2json.py, documentado na mesma página, pode ser utilizado para converter bases nos formatos ISIS ou ISO-2709 para JSON com subcampos saparados ou não, conforme as figuras 2 e 3 acima.

SGBDs semiestruturados Open Source, como o CouchDB e o MongoDB, preservando as características do modelo de dados ISIS que têm se mostrado úteis ao longo das décadas.

Concretamente, o projeto ISIS-DM está produzindo uma API[9], uma biblioteca de programação e conjunto de ferramentas que permitem:

1. definir esquemas de dados semiestruturados com regras de validação, tais como aquelas encontradas no Dicionário de Dados LILACS, através de classes em uma linguagem de programação orientada a objetos (Python, inicialmente), semelhante ao que permitem os atuais frameworks de persistência de dados;
2. extrair de dados para exibição e indexação utilizando APIs modernas para manipular strings, listas e dicionários;
3. gerar uma definição de esquema de dados a partir da análise automática de uma base de dados concreta;

As facilidades acima já se encontram implementadas[10] como prova de conceito. Os próximos passos são:

4. geração automática de formulários de entrada de dados a partir das definições de esquemas;
5. implementação de interfaces para persistência de dados em dois ou mais SGBDs semiestruturados, como Apache CouchDB, MongoDB, Amazon SimpleDB ou Google Datastore.

O objetivo é avaliar a viabilidade de migrar grandes bases de dados bibliográficas como LILACS e SciELO[11] para uma nova plataforma, mantendo intactos sua metodologia e o modelo de dados ISIS, mas ganhando independência em relação a qualquer mecanismo de persistência ou SGBD específico, para melhor sustentabilidade a longo prazo.

---

[9] API: Application Programming Interface, interface de programação de aplicações

[10] Código-fonte disponível no repositório público RedDes da BIREME/OPAS/OMS: <http://reddes.bvsalud.org/projects/isisnbp/browser/isisdm/properties>

[11] Scientific Electronic Library Online: base de dados de texto completo formada por periódicos científicos da América Latina, Caribe e outros países: <http://www.scielo.org>

7. Definição de esquemas de dados no ISIS-DM

A notação para a definição de um esquema de dados no ISIS-DM foi guiada por:

- o modelo de dados ISIS como expresso em sua documentação, em particular no que se refere à Field Definition Table (FDT), a notação utilizada para definir planilhas de entrada de dados e regras de validação de campos e subcampos (Unesco, 1989);
- o esquema de dados da base LILACS descrito em seu dicionário de dados e manual da metodologia (BIREME, 2008a; BIREME, 2008b);
- APIs de persistência de dados como o ORM[12] do framework Django (Django, 2010) e a Python Datastore API (Google, 2010).

O conceito fundamental do ISIS-DM é definir, para cada tipo de registro de uma base de dados, uma classe em uma linguagem de programação orientada a objetos. Por exemplo, na base LILACS a combinação dos campos 5 (tipo de literatura) e 6 (nível de tratamento) definem o tipo de registro, e consequentemente seu esquema de dados: campos permitidos, campos obrigatórios, campos opcionais etc.

Um exemplo simples de definição de esquema de dados no ISIS-DM em linguagem Python pode ser visto na Figura 4.

b.

```
class Monografia(CheckedModel):
    titulo = SingularProperty(required=True)
    autores = PluralProperty(required=True, subfields='pr')
    paginas = NumberProperty(validator=gt_zero)
```

Figura 4: *Definição de uma classe de esquema de dados no ISIS-DM. A classe Monografia é subclasse de CheckedModel, que implementa diversos métodos para criação, acesso e validação dos campos de um registro. Os campos são instâncias de diferentes classes: SingularProperty gera campos de texto não repetitivos, PluralProperty gera campos texto repetitivos, e NumberProperty gera campos numéricos não repetitivos. Os argumentos passados para os construtores dos diferentes campos definem restrições para validação de campos obrigatórios e subcampos permitidos.*

[12] Object-relational Mapper: Mapeador objeto-relacional, uma API orientada a objetos para a manipulação de registros relacionais

Dada uma definição de esquema de dados como a da Figura 4, a representação dos registros em JSON pode ser ainda mais simplificada, como na Figura 5:

```
{"titulo": "Jaguadarte: Estudo comparado de traduções",
 "autores": ["Lewis Carroll^pBrasil^rEditor",
             "John Tenniel^pBrasilr^Ilustrador"],
 "paginas": 128
}
```

Figura 5: *Este exemplo ilustra um registro JSON gerado a partir do esquema de dados da figura 4, onde se define que "titulo" é um campo não repetitivo, portanto não é necessário representar este valor como uma lista de ocorrência, mas apenas com um simples texto. Da mesma forma, o campo numérico "páginas" pode ser expresso por um inteiro. Apenas para o campo autores, que é repetitivo, a estrutura de lista precisa ser utilizada. A predefinição do esquema de dados possibilita ainda que o uso de subcampos seja controlado. Apenas os marcadores "p" e "r" podem ser usados no campo "autores", como definido na figura 4.*

8. Geração de um esquema a partir de uma massa de dados

Entre as bases de dados operadas pela BIREME/OPAS/OMS, LILACS é a que tem o modelo de dados mais complexo. As principais causas desta complexidade são:

- longo tempo de operação: a metodologia LILACS existe desde 1982[13], e neste período o esquema de dados já passou por sete edições; em uma amostra contendo 100.000 registros recentes da base LILACS, encontramos 58 campos distintos representados entre os registros;
- variedade de tipos de registro: a base LILACS contém registros descrevendo coleções, periódicos e monografias individuais ou como parte de periódicos ou capítulos de livros, totalizando 14 tipos diferentes de registros básicos, que podem ter campos complementares quando descrevem documentos ligados a conferências ou projetos.

Qualquer solução futura precisa levar em conta esta complexidade, e para lidar com ela desenvolvemos um programa que gera um modelo de dados ISIS-DM a partir da análise de uma massa de dados. O programa, schematize.py[14] lê uma base em formato

---

[13] BIREME/OPAS/OMS. Metodologia LILACS: Manual de Descrição Bibliográfica , 7ª ed. (2008), p. 3

[14] documentação em <http://reddes.bvsaude.org/projects/isisnbp/wiki/Schematizer>, código-fonte no repositório RedDes da BIREME/OPAS/OMS em:

ISIS-JSON e produz o código-fonte de uma subclasse de isisdm.CheckedModel como na Figura 6.

Em linhas gerais, o programa schematize.py varre a base de entrada do primeiro ao último registro, fazendo o seguinte para cada campo encontrado:

- se nunca existe mais de uma ocorrência, gerar uma SingularProperty, do contrário, PluralProperty ;
- se o tag aparece em todos os registros, gerar um atributo required=True ;
- se ocorrem subcampos, gerar um atributo subfields com as marcas encontradas ;
- se todas as ocorrências de um campo são do mesmo tamanho N, gerar um atributo fixed_len=N ;
- se todas as ocorrências de um campo são numéricas, gerar um atributo numeric=True ;

<http://reddes.bvsaude.org/projects/isisnbp/browser/tools/schematize.py>

```
class SampleType(CheckedModel):
    v12 = SingularProperty(tag='12', subfields=u'dnp')
    v24 = SingularProperty(tag='24', required=True)
    v25 = PluralProperty(tag='25')
    v26 = SingularProperty(tag='26', subfields=u'abc')
    v30 = SingularProperty(tag='30', subfields=u'abc')
    v44 = SingularProperty(tag='44', subfields=u'v')
    v50 = SingularProperty(tag='50')
    v69 = SingularProperty(tag='69')
    v70 = PluralProperty(tag='70')
    v71 = PluralProperty(tag='71')
    v72 = SingularProperty(tag='72', subfields=u'dp')
    v74 = PluralProperty(tag='74')
    v76 = PluralProperty(tag='76', subfields=u'z')

# 150 records analyzed out of 150 total records
# 13 distinct tags found
# 0 unique fields detected
```

Figura 6: *Classe de registro gerada automaticamente pelo programa schematize.py a partir da análise dos registros da base CDS[15]. Os subcampos listados são os que ocorrem nos registros analisados. O campo 24 ganhou o atributo "required" porque o programa verificou que todos os registros da amostra possuem este campo. É apenas uma suposição: o programa não tem a pretensão de gerar esquemas definitivos, mas apenas auxiliar o programador na definição de um esquema. Os comentários assinalados com # ao final do código-fonte dão estatísticas sobre a base analisada: 150 registros analisados, 13 tags (campos) distintos encontrados, e nenhum campo de conteúdo único que poderia ser usado como chave primária para identificação dos registros.*

Além disso, é possível filtrar os registros de entrada por tipo, para gerar um esquema específico para cada tipo de registro. Por exemplo, na base xLILACS[16] existem registros dos tipos C, R, S e E, com estruturas e significados bastante distintos, como descrito no dicionário de dados (BIREME, 2008a). Não faria sentido, no caso da base xLILACS, gerar um esquema a partir de todos os registros, pois a semântica dos campos varia conforme o tipo do registro. Dado que o tipo de registro é definido pelo valor do tag 5, o programa schematize.py pode ser invocado de forma a selecionar apenas os registros com valor R no tag 5, da seguinte maneira: "./schematize.py xlilacs.json -f5:R". O resultado é o esquema ilustrado na Figura 7.

```
class TypeR(CheckedModel):
    v01 = SingularProperty(tag='1', required=True,
                  subfields=u'eip')
    v02 = SingularProperty(tag='2', required=True)
    v03 = SingularProperty(tag='3', required=True,
                  subfields=u'af')
    v04 = PluralProperty(tag='4', subfields=u'eipt',
                  fixed_len=31)

# 8 records analyzed out of 117 total records
# 5 distinct tags found
```

[15] a base CDS é um dos exemplos incluídos no pacote do aplicativo CDS/ISIS for Windows (WinISIS) distribuído pela UNESCO; para realizar este teste, convertemos a base CDS do formato MST nativo do CDS/ISIS para ISIS-JSON usando o isis2json.py.

[16] xLILACS é uma base auxiliar incluída no pacote **LILDBI-Web, o atual aplicativo padrão de entrada de dados da metodologia LILACS. A base xLILACS contém a descrição do modelo de dados LILACS e parâmetros de operação para o LILDBI-WEB.**

Figura 7: *Esquema de dados gerado a partir da base xLILACS, filtrando apenas os registros do tipo R. O campo 4 recebeu o atributo "fixed_len=31" porque o programa verificou que este é o tamanho de todos as ocorrências deste campo na amostra. Pode ser mera coincidência, mas o objetivo é fornecer dados para as decisão do programador que vai utilizar o código-fonte gerado como ponto de partida para a definição de um esquema.*

9. Pesquisas futuras

Os componentes de software do projeto ISIS-DM são pré-requisito para uma migração experimental da base LILACS para um gerenciador de banco de dados semiestruturado capaz de importar dados JSON, como ocorre com CouchDB e MongoDB. Uma vez que uma amostra significativa da base LILACS seja carregada nestes gerenciadores de banco de dados, estaremos em condições de medir o desempenho da carga e acesso aos registros e avaliar as facilidades de busca, os protocolos de comunicação com aplicações, os mecanismos de replicação e outros aspectos relevantes para o desenvolvimento de aplicativos de catalogação, pesquisa bibliográfica e bibliometria, essencialmente os tipos de aplicativos que a BIREME/OPAS/OMS já opera sobre a plataforma ISIS tradicional. Também está nos planos a avaliação de frameworks web existentes quando à sua flexibilidade para lidar com dados semiestruturados.

10. Conclusões

Conforme observa Setzer (2005), o modelo relacional normalizado não é o mais conveniente para representar registros bibliográficos no contexto de sistemas de catalogação cooperativa, onde o intercâmbio de registros completos traz mais benefícios que a normalização. Por outro lado, o modelo de dados do ISIS, influenciado pelo padrão MARC e pela norma ISO-2709, nasceu exatamente para representar registros bibliográficos. Isso justifica a preferência por estes formatos na biblioteconomia.

Por muitos anos, tanto o mercado quanto a academia se concentraram no modelo relacional. Somente a partir de 1995, o modelo semiestruturado foi formalizado e ganhou uma base teórica. Mais recentemente, aumentou o interesse do mercado por SGBDs não-relacionais, e o modelo semiestruturado ganhou novas implementações e grandes usuários, como Google e Amazon.com. Esta tendência significa que, após um período em que "estado da arte" em bancos de dados era sinônimo de "modelo relacional", agora

podemos observar inovação e um número crescente de casos de uso de gerenciadores de bancos de dados não-relacionais. Consequentemente, os profissionais de biblioteconomia podem esperar o surgimento de sistemas capazes de atender às necessidades típicas de catalogação cooperativa tão bem quanto a tradicional família ISIS, mas desta feita incorporando a arquitetura robusta e escalável que se espera de verdadeiros sistemas gerenciadores de bancos de dados multiusuário.

Referências

ABITEBOUL, Serge; BUNEMAN, Peter e SUCIU, Don. Data on the Web: From Relations to Semistructured Data and XML. Morgan Kaufmann, San Francisco, 1999.

AMAZON.COM. Amazon SimpleDB. Disponível em <http://aws.amazon.com/simpledb/> acesso em: ago 2010.

ANDERSON, J. Chris; LEHNARDT, Jan; SLATER, Noah. CouchDB: The Definitive Guide, Sebastopol, O'Reilly Media, 2010.

BIREME/OPAS/OMS. Diccionario de datos del modelo LILACS Versión 1.6a, São Paulo, SP, 2008.

BIREME/OPAS/OMS. ISIS Network Based Platform. Disponível em: <http://reddes.bvsaude.org/projects/isisnbp>, acesso em: set 2010.

BIREME/OPAS/OMS. Metodologia LILACS: Manual de Descrição Bibliográfica , 7ª ed. São Paulo, SP, 2008.

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

CUNHA, Murilo Bastos da; CAVALCANTI, Cordélia Robalinho de Oliveira. Dicionário de Biblioteconomia e Arquivologia. Brasília, DF: Briquet de Lemos/Livros, 2008. 451p.

DJANGO SOFTWARE FOUNDATION. Django Project / Model Syntax. Disponível em <http://www.djangoproject.org>, acesso em: set 2010.

FERREIRA, Aurélio Buarque de Hollanda. Novo Dicionário da Língua Portuguesa. 4ª ed., Curitiba, PR: Positivo, 2009.

GOOGLE INC. The Python Datastore API. Disponível em : <http://code.google.com/appengine/docs/python/datastore/>, acesso em: ago, 2010.

HASSELBRING, W. Information System Integration, Communications of the ACM, Volume 43, Issue 6 , p. 32 - 38, 2000.

INFOLAB, Stanford University. Lore: a Database Management System for XML. Disponível em: <http://infolab.stanford.edu/lore/home/index.html>, acesso em: ago 2010.

LIU, L.; ÖZSU, M. T. Encyclopedia of database systems: Springer, 2009.

MARTINEZ USERO, J.A.; LARA NAVARRA, P. La interoperabilidad de la información. Barcelona: UOC, 2007.

RAMALHO, Luciano. Implementing a modern API for CDS/ISIS, a classic semistructured NoSQL database, 11º Fórum Internacional do Software Livre, XI Workshop Sobre Software Livre, p. 42 – 47, Porto Alegre, 2010.

SETZER, Valdemar; CORRÊA DA SILVA, Flávio. Bancos de dados: aprenda o que são, melhore seu conhecimento, construa os seus. 1ª ed. São Paulo: Edgard Blücher, 2005.

UNESCO. Mini-micro CDS-ISIS Reference Manual, version 2.3. Paris, 1989.

UNIVERSITY OF PENNSYLVANIA. Database Group, Semistructured data, and XML. Disponível em: <http://db.cis.upenn.edu/research/SS_XML.html>, acesso em: ago 2010.

10GEN, INC. MongoDB (web site). Disponível em: <http://www.mongodb.org/>, acesso em: ago 2010.