

Luciano G. S. Ramalho

**O Modelo de Dados Semiestruturado
em Bases Bibliográficas:
do CDS/ISIS ao Apache CouchDB**

São Paulo, SP – Brasil
22 de novembro de 2010

DEPARTAMENTO DE BIBLIOTECONOMIA E DOCUMENTAÇÃO
ESCOLA DE COMUNICAÇÕES E ARTES
UNIVERSIDADE DE SÃO PAULO

Luciano G. S. Ramalho

**O Modelo de Dados Semiestruturado
em Bases Bibliográficas:
do CDS/ISIS ao Apache CouchDB**

Monografia apresentada para obtenção
do grau de Bacharel em Biblioteconomia
pela Universidade de São Paulo.

Orientador:

Prof. Dr. Marcos Luiz Mucheroni

São Paulo, SP – Brasil
22 de novembro de 2010

Resumo

O padrão MARC e bases de dados bibliográficas como LILACS e SciELO são exemplos de uso do modelo de dados semiestruturado em biblioteconomia. Esta monografia apresenta as bases teóricas do modelo semiestruturado, comparando-o ao modelo relacional, e descreve seu uso na base de dados LILACS e sua implementação nos sistemas de bancos de dados CDS/ISIS e Apache CouchDB. Em seguida, apresenta um método e ferramentas para a conversão de registros do CDS/ISIS para o CouchDB. Finalmente, é descrita a migração de cerca de 80.000 registros da base bibliográfica LILACS do CDS/ISIS para o CouchDB, e são implementados relatórios usando as facilidades de indexação deste gerenciador de banco de dados.

Descritores: bancos de dados, registros bibliográficos, ISO-2709, conversão de dados, modelos de dados, modelo semiestruturado, ISIS, JSON

Abstract

The MARC standard and bibliographic databases such as LILACS and SciELO are examples of the use of the semistructured data model in library science. This monograph presents the theoretical basis for the semistructured data model, comparing it to the relational model, and describes its use in the LILACS database and its implementation in the CDS/ISIS and the Apache CouchDB database systems. Next, a method and tools for the conversion of records from CDS/ISIS to CouchDB are shown. Finally, the migration of about 80.000 records of the LILACS bibliographic database, from CDS/ISIS to CouchDB, is described, and reports are implemented using the indexing facilities of this database management system.

Keywords: databases, bibliographic records, ISO-2709, data conversion, data models, semistructured model, ISIS, JSON

Para Marta,
por tudo.

Agradecimentos:

Colegas da BIREME/OPAS/OMS

Marcos Mucheroni

Imre Simon

Jairo e Maria Lucia

“Não é na certeza, na segurança e na tranquilidade que se fazem descobertas.”

C. G. Jung

Sumário

1	Introdução.....	10
1.1	Cenário atual e motivação.....	10
1.1.1	Exemplo de uso: ISIS na catalogação cooperativa.....	10
1.1.2	Contexto tecnológico.....	11
1.1.3	Crise e oportunidade.....	12
1.2	Objetivo deste trabalho.....	13
1.2.1	Objetivo geral.....	13
1.2.2	Objetivos específicos.....	13
1.3	Descrição do problema de pesquisa.....	13
1.4	Sumário da metodologia.....	13
2	Revisão da Literatura.....	14
2.1	Bancos de dados, bases de dados e termos relacionados.....	14
2.1.1	Resumo da terminologia adotada.....	16
2.2	Modelo de dados relacional.....	17
2.3	Modelo de dados semiestruturado.....	19
2.3.1	O livro vermelho de Hellerstein e Stonebraker.....	19
2.3.2	Referência específica: Data on The Web.....	22
2.3.3	Referência Específica: Semistructured Database Design.....	24
2.4	Família ISIS.....	25
2.4.1	CISIS.....	26
2.4.2	ISIS-NBP: ISIS Network Based Platform.....	26
2.4.3	WinISIS, J-ISIS e outras referências.....	27
2.5	Modelo de dados ISIS.....	28
2.5.1	Níveis de organização de uma base ISIS.....	29
2.5.2	ISIS Formatting Language: a linguagem de extração de dados.....	32
2.6	XML e JSON.....	33
2.6.1	XML narrativo versus XML de dados.....	34
2.6.2	Comparação entre JSON e XML.....	35
2.6.3	Modelo de dados JSON.....	37

2.7	Sistemas de banco de dados não-relacionais.....	39
2.7.1	Sistemas de bancos de dados orientados a documento.....	40
2.8	Metodologia LILACS.....	42
3	Metodologia.....	43
3.1	Seleção do sistema de banco de dados.....	43
3.2	Escolha da representação de registros ISIS em JSON.....	44
3.2.1	ISIS-JSON: padrões para a representação de registros ISIS em JSON.....	47
3.2.2	Formatos ISIS-JSON baseados em lista associativa.....	48
3.2.3	Formatos ISIS-JSON baseados em dicionário.....	51
3.2.4	Resumo dos tipos de ISIS-JSON.....	53
3.3	Seleção e obtenção do conjunto de dados.....	54
3.3.1	Procedimento de download da amostra.....	54
3.3.2	Ferramenta de conversão de ISO-2709 para JSON.....	57
3.3.3	Conversão da amostra de LILACS100K para JSON.....	59
3.4	Carga dos dados no CouchDB.....	61
3.5	Análise dos registros no CouchDB.....	62
3.5.1	Frequência de tipos de registro na amostra LILACS100K.....	64
3.5.2	Investigação: uso de subcampos repetidos em LILACS100K.....	71
4	Resultados.....	76
4.1	Levantamento de uma base teórica para estudar o modelo de dados ISIS.....	76
4.2	Identificação de SGBDs compatíveis com o modelo de dados ISIS.....	76
4.3	Catologação das variantes de ISIS-JSON.....	76
4.4	Ferramentas de conversão.....	77
4.5	Ferramentas de análise.....	77
4.6	Identificação de inconsistências na base LILACS.....	77
5	Conclusão.....	78
5.1	Limitações da pesquisa.....	78
5.2	Indicações para futuras pesquisas e indicações práticas.....	80
5.2.1	APIs para especificação de esquemas.....	80
5.2.2	Atualização automática de dados duplicados.....	81
	Glossário.....	83
	Bibliografia.....	86

1 Introdução

1.1 Cenário atual e motivação

Ao estudar o uso da informática em bibliotecas no Brasil, chama a atenção o fato de que muitas utilizam sistemas de bancos de dados da família ISIS¹, que teve origem na Unesco. Ao contrário dos principais sistemas de bancos de dados do mercado atual, a família ISIS não segue o Modelo Relacional Normalizado (MRN)². Enquanto no mercado de informática a ideia de banco de dados está fortemente ligada às características típicas de um sistema relacional, como a estrutura de tabelas e sua manipulação via linguagem SQL, num sistema ISIS nada disso existe.

À primeira vista, pode parecer que a opção pela família ISIS se deve apenas à falta de recursos das bibliotecas para investir em informática, já que os aplicativos ISIS têm sido distribuídos sem custo pela Unesco há décadas, muito antes de existirem sistemas de bancos de dados relacionais Open Source. Porém, um exame mais detido das características dos sistemas ISIS, e de como estas características são utilizadas em grandes projetos, revela que se trata de uma tecnologia muito bem adaptada à manipulação de registros bibliográficos, oferecendo vantagens importantes em relação a sistemas MRN neste tipo de aplicação.

1.1.1 Exemplo de uso: ISIS na catalogação cooperativa

Uma das principais fontes de informação coordenadas pela BIREME/OPAS/OMS³ é a base LILACS – Literatura Latino-Americana e do Caribe em Ciências da Saúde – um índice bibliográfico construído através de catalogação cooperativa, envolvendo centenas de centros de informação espalhados pelas Américas. Um registro bibliográfico de LILACS passa por várias etapas em seu ciclo de produção, desde a digitação inicial até a publicação, passando por revisão, certificação, indexação. Estas etapas são realizadas de forma distribuída na rede de centros cooperantes, o que significa que um registro pode ser transferido de um centro para outro antes ser publicado. E uma vez publicado, uma das principais utilidades de um índice

1 A família ISIS compreende diversos produtos, desde o aplicativo WinISIS, para microcomputadores de mesa, até sistemas voltados para a publicação de bases de dados na Web, e utilitários para o processamento de grandes volumes de registros em lote. Todos estes sistemas têm em comum o modelo de dados ISIS e uma linguagem para definição de índices e extração de dados.

2 SETZER, 2005, p. 13.

3 BIREME/OPAS/OMS: Centro Latino-Americano e do Caribe de Informação em Ciências da Saúde, um centro especializado da Organização Pan-Americana da Saúde/Organização Mundial da Saúde.

bibliográfico digital é a reutilização dos dados em outros sistemas, o que novamente implica na transmissão de registros.

Neste contexto, a principal vantagem da família ISIS em relação ao modelo relacional é oferecer uma representação mais rica de registro, com a possibilidade de campos repetitivos e subcampos, características comuns aos principais esquemas de registros bibliográficos, tais como MARC⁴, UNISIST⁵ e LILACS. Em um banco de dados relacional, as regras de normalização forçam o desmembramento dos dados de um registro bibliográfico em vários registros em diferentes tabelas. Esse espalhamento dificulta a gestão distribuída dos registros, e tornaria mais complexa a catalogação cooperativa nos moldes em que a rede LILACS opera.

1.1.2 Contexto tecnológico

Embora os sistemas de bancos de dados relacionais continuem sendo os mais utilizados no mercado, nos últimos anos o número de sistemas não-relacionais está aumentando, e não diminuindo. A natureza semiestruturada e muitas vezes hierárquica da organização de informações na Web, bem como a demanda por escalabilidade horizontal, esbarram em limitações práticas do modelo relacional (EURE, 2009). Isso tem motivado maior interesse do mercado por sistemas de banco de dados não relacionais, evidenciado pelo lançamento de novos produtos, publicações e conferências focadas nesta categoria de software que, em 2009, ganhou o rótulo NoSQL⁶. Apache Cassandra, Redis, Apache CouchDB e MongoDB são alguns exemplos de sistemas de bancos de dados NoSQL Open Source lançados nos últimos 5 anos. Dois exemplos de sistemas NoSQL proprietários com implantações de larga escala são Bigtable, da Google Inc. (CHANG, 2007), e Dynamo, da Amazon.com (DECANDIA, 2007).

Os proponentes de NoSQL não advogam que o modelo relacional esteja superado, ou o abandono da linguagem SQL. O termo NoSQL hoje é apresentado com abreviatura de “Not only SQL” (não apenas SQL). A ideia é que determinadas aplicações podem se beneficiar de modelos de dados alternativos que possuem características complementares em relação ao modelo relacional normalizado.

4 MARC: Machine Readable Cataloging, formato de registro bibliográfico criado pela Library of Congress, a biblioteca do Congresso dos Estados Unidos da América.

5 UNISIST: UNESCO's World Scientific Information Programme, projeto de disseminação de informação científica e tecnológica que estabeleceu a identificação de periódicos por ISSN (International Standard Serial Number), e definiu uma metodologia para descrição de documentos que influenciou a metodologia LILACS.

6 A origem do termo é relatada no artigo **NoSQL** na Wikipedia em inglês, disponível em <<http://en.wikipedia.org/wiki/NoSQL>>. Acesso em 14 nov. 2010.

1.1.3 Crise e oportunidade

Embora o modelo de dados ISIS atenda muito bem às necessidades de aplicações em biblioteconomia e ciência da informação, a falta de interesse do mercado tem limitado a inovação na família de sistemas ISIS. Mesmo aqueles produtos ISIS que se tornaram Open Source^{7 8} têm avançado lentamente por não terem reunido a massa crítica de desenvolvedores externos necessária para sua evolução sustentável, resultando em uma defasagem tecnológica.

Porém, o surgimento de novos sistemas de bancos de dados não-relacionais traz uma oportunidade de renovação tecnológica para aplicações que dependem do modelo de dados do ISIS. Em particular, entre os sistemas NoSQL Open Source mencionados, dois se classificam como sistemas de bancos de dados orientados a documentos (document oriented databases): Apache CouchDB e MongoDB (LENNON, 2009; PLUGGE, 2010). Nestes sistemas, um registro é um documento formado por uma quantidade variável de campos identificados por chaves alfanuméricas, e o valor de cada campo pode ser simples ou composto, como listas ordenadas de valores ou dicionários, que são associações entre chaves e valores.

Assim, no CouchDB e no MongoDB um registro ISIS completo pode ser armazenado como um único documento: campos repetitivos podem ser representados como listas de ocorrências, e subcampos podem ser armazenados em dicionários, associando cada código de subcampo ao seu valor⁹.

Isso torna viável migrar bases de dados que hoje utilizam a plataforma ISIS para CouchDB ou MongoDB sem alterar a estrutura dos registros dessas bases. Concretamente, temos a oportunidade de renovar a plataforma tecnológica das bases de dados LILACS sem perder os 25 anos de experiência e boas práticas em catalogação cooperativa que estão codificadas nos manuais de descrição bibliográfica da Metodologia LILACS.

7 A maioria dos sistemas da BIREME/OPAS/OMS baseados em ISIS têm repositórios de código públicos e são distribuídos como software livre pela licença LGPL. O acesso aos repositórios se dá pela URL <<http://reddes.bvsalud.org/php/index.php>>. Acesso em 14 nov. 2010.

8 O J-ISIS da Unesco também é Open Source, mas seu repositório público registra contribuições de um único usuário jcd, presumivelmente Jean-Claude Dauphin, que em 2008 era colaborador da Unesco. A alteração mais recente no repositório de código em 20 nov. de 2010 é a revisão #60 de 3 mai. 2009, conforme registra a página <<http://kenai.com/projects/j-isis/sources/subversion/show>>. Acesso em 20 nov. 2010.

9 Precisamente como representar um registro ISIS genérico em um banco de dados deste tipo é um dos temas principais deste trabalho.

1.2 Objetivo deste trabalho

1.2.1 Objetivo geral

Estudar a viabilidade de migração de dados de uma base de dados ISIS para um novo sistema banco de dados compatível com o modelo de dados ISIS.

1.2.2 Objetivos específicos

- Avaliar formas alternativas de representação de registros ISIS.
- Selecionar um sistema de bancos de dados apropriado para a armazenagem de bases ISIS.
- Desenvolver métodos e ferramentas de migração de dados.
- Migrar uma massa de dados substancial da base LILACS para um novo sistema de banco de dados.
- Realizar análises sobre a massa de dados utilizando os recursos do novo sistema banco de dados, para testar suas possibilidades.

1.3 Descrição do problema de pesquisa

A família de sistemas ISIS mostrou-se muito bem adaptada à operação de bases bibliográficas nos últimos 25 anos, mas com o tempo vieram também maiores dificuldades para evoluir. Com o surgimento de novos sistemas com modelos de dados mais flexíveis e dinâmicos, será que existe hoje uma alternativa para a migração de bases ISIS que evite uma reestruturação dos dados com impacto sobre as próprias metodologias de catalogação?

1.4 Sumário da metodologia

Após selecionar o sistema de banco de dados CouchDB como alvo da migração, avaliamos diferentes maneiras para representar registros ISIS de forma compatível com este sistema. Em seguida selecionamos uma amostra de registros da base LILACS, criamos ferramentas para conversão da amostra e a carregamos em uma instância de CouchDB. Finalmente, implementamos um relatório básico e um avançado para experimentar os mecanismos de indexação de dados do CouchDB.

2 Revisão da Literatura

2.1 Bancos de dados, bases de dados e termos relacionados

Antes de mais nada, precisamos superar uma dificuldade terminológica inicial. O termo “bancos de dados” é empregado com sentidos diferentes, mesmo em textos acadêmicos, e, às vezes, até em uma mesma oração. Uma contribuição pretendida por este trabalho é introduzir alternativas ao termo “banco de dados”, de acordo com suas diferentes acepções. Neste primeiro momento, buscaram-se as definições básicas, e, por isso, a escolha de começar pelas entradas 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).

“Banco de dados”, portanto, pode ser sinônimo de base de dados, no sentido de “coleção de dados”, ou, um programa que gerencia tal coleção. São conceitos muito distintos. É como confundir a água com o copo¹⁰. No mesmo dicionário, o verbete “base de dados” tem apenas uma definição:

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

Note que o dicionário remete à definição 1 de “banco de dados”. Isso significa que, para os lexicógrafos do Aurélio, “base de dados” é sempre uma coleção de dados.

Seguindo essa premissa, conseguimos evitar ambiguidades se usarmos sempre o “base de dados” em vez de “banco de dados” quando queremos nos referir a um conjunto de dados, e não a um programa de computador. Por exemplo: “LILACS é uma base de dados que inclui parte da produção científica em saúde da América Latina e do Caribe”.

Existe ainda um terceiro significado para “banco de dados” na terminologia técnica de administração de sistemas: na configuração de computadores, também se denomina “banco de dados” um conjunto de tabelas inter-relacionadas, identificado por um nome, e configurado com regras específicas de controle de acesso.¹¹ Por exemplo, em um computador rodando uma instância do software MySQL podem ser definidos vários bancos de dados, e cada banco de

¹⁰ Em linguagem coloquial, quando pedimos “um copo d'água” estamos interessados no líquido, e não no recipiente apropriado para servir uma porção individual de água.

¹¹ Veja por exemplo a seção “3.3.1. Criando e Selecionando um Banco de Dados” do **Manual de Referência do MySQL 4.1** em português, disponível em <<http://dev.mysql.com/doc/refman/4.1/pt/creating-database.html>>. Acesso em 4 abr. 2010.

dados é formado por uma ou mais tabelas.

A fim de diferenciar os três sentidos de “banco de dados”, adotaremos neste trabalho os seguintes termos:

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.

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 SQLite5 é um motor de banco de dados relacional e CISIS é um motor de banco de dados semiestruturado.

Finalmente, para evitar ambiguidade não usaremos o termo “banco de dados” sem uma das qualificações acima.

2.1.1 Resumo da terminologia adotada

Termo	Definição	Exemplo de uso
base de dados	coleção organizada de dados	<i>LILACS é uma base de dados de referências bibliográficas.</i>
sistema de banco de dados	termo genérico para qualquer software usado para manipular bases de dados	<i>PostgreSQL e WinISIS são dois sistemas de bancos de dados bem distintos.</i>
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	<i>O PostgreSQL é SGBD relacional, e o Apache CouchDB é um SGBD semiestruturado, mas o WinISIS não é um SGBD por ser um aplicativo monousuário.</i>
objeto banco de dados	conjunto nomeado de tabelas ou registros, comumente armazenado em um único arquivo no sistema de arquivos do computador	<i>No PostgreSQL um objeto banco de dados contém tabelas, mas no WinISIS um objeto banco de dados contém apenas registros.</i>
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	<i>O SQLite5 é um motor de banco de dados relacional e o CISIS é um motor de banco de dados da família ISIS.</i>

Tabela 1: Termos adotados nesta monografia para distinguir as diferentes acepções do termo "banco de dados".

2.2 Modelo de dados relacional

Informalmente, a observação de que “banco de dados” é sinônimo de “banco de dados relacional” se aplica inclusive ao mercado editorial acadêmico. A abordagem de livros-texto importantes como *Fundamentals of Database Systems* (ELMASRI, 2007) – adotado na disciplina MAC 426/5760 – Introdução aos Sistemas de Bancos de Dados no IME/USP – e *Sistema de Banco de Dados* (SILBERSCHATZ, 2006) enfatiza o modelo relacional. Modelos de dados não-relacionais têm uma abordagem superficial, motivada pela discussão de XML como forma de representação, bem como sistemas de bancos de dados orientados a objeto.

Na bibliografia de bancos de dados brasileira destaca-se a obra *Bancos de dados: aprenda o que são, melhore seu conhecimento, construa os seus* (SETZER, 2005). Além de alternar seções de didática informal, e, até bem humorada, com definições formais rigorosas, Setzer apresenta uma visão crítica do modelo relacional normalizado (MRN), como motivação para sua abordagem do modelo relacional não-normalizado (MRNN). Uma passagem em particular é extremamente relevante para este trabalho:

Para motivar o MRNN, que será abordado no cap. 6, seria interessante notar o absurdo do padrão do MRN: 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 – isso na solução com duplicações [...]. Na solução sem duplicação, a situação ainda piora mais (calcule como exercício quantas linhas seriam no total). Mas o que se vê e pega-se na mão no mundo real é um livro só, e não um picadinho de livro! Veremos que no MRNN tudo isso pode ser representado em uma só linha, que é o que se esperaria de um modelo de dados decente em forma de tabela: uma linha para cada livro.¹²

O problema descrito por Setzer advém da chamada 1ª Forma Normal (1FN ou 1NF, na sigla em inglês), um fundamento do modelo relacional:

No modelo relacional, formalizamos essa ideia de que atributos não possuem qualquer subestrutura. Um domínio é atômico se os elementos do domínio são considerados unidades indivisíveis. Dizemos que um esquema de relação R está na primeira forma normal (1FN) se os domínios de todos os atributos de R são atômicos.¹³

Esta definição claramente exclui subcampos e campos multivalorados (ou campos repetitivos, para usar a terminologia mais comum em biblioteconomia). Portanto, em um sistema de banco de dados aderente à 1FN, não podemos registrar os 3 autores em um mesmo registro.

¹² SETZER, 2005, p. 135.

¹³ SILBERSCHATZ, 2006, p. 178.

C. J. Date defende a opinião de que o requisito de atomicidade da 1FN, expresso na citação de Silberschatz et. al. precedente, é impreciso¹⁴ e que “em nenhum lugar o modelo relacional prescreve quais devem ser tais tipos [dos atributos das relações] e de fato eles podem ser tão complexos quanto se queira”¹⁵ Date vai além, afirmando na mesma página que, já que os atributos podem ser de qualquer tipo, toda e qualquer relação está na 1FN, por definição.

Setzer também defende que a 1FN não é um pré-requisito indispensável para a 2ª e a 3ª Forma Normal (2FN, 3FN)¹⁶. Mas Setzer, Silberschatz, Elmasri discordam de Date em sua caracterização de que o modelo relacional normalizado aceita campos com valores complexos, porque o requisito de atomicidade foi expresso por E. F. Codd – proponente original do modelo relacional – exatamente no trecho do seu artigo seminal em que o conceito de normalização foi definido:

For this reason (and others to be cited below) the possibility of eliminating nonsimple domains appears worth investigating. There is, in fact, a very simple elimination procedure, which we shall call normalization.¹⁷

Elmasri e outros autores referem-se ao modelo defendido por Date como “nested relational model” (modelo relacional aninhado) ou NF² como abreviatura para NFNF – Non-First Normal Form, e introduz o termo “flat relational model” (modelo relacional plano) para se referir ao modelo relacional normalizado como definido por Codd¹⁸. Estes são os modelos que Setzer chama de MRNN e MRN. No entanto, ainda segundo Elmasri e Setzer, o MRNN não é plenamente implementado em nenhum sistema de banco de dados amplamente distribuído e os recursos que existem neste sentido, em SGBDs como Oracle e PostgreSQL, são pouco utilizados na prática.

Não é coincidência que Setzer utilize justamente um registro bibliográfico como exemplo motivador para apresentar o modelo relacional não-normalizado (MRNN). Evidentemente o MRN pode ser utilizado perfeitamente para lidar com este tipo de registro, mas é inegável que ele introduz complexidade, e os benefícios que acompanham esta complexidade não são tão evidentes, ao menos no caso de registros bibliográficos que, por sua própria natureza, tendem a ser registros estáticos.

14 DATE, 2005, p. 29.

15 DATE, 2005, p. 37. Nossa tradução.

16 SETZER, 2005, p. 299.

17 CODD 1970, p. 381 “Por este motivo (e outros a serem citados adiante) a possibilidade de eliminar domínios não-simples merece investigação. Há, de fato, um procedimento muito simples de eliminação, que chamaremos de normalização.” Nossa tradução.

18 ELMASRI, 2007, p. 788.

2.3 Modelo de dados semiestruturado

No plano teórico, o primeiro desafio deste trabalho foi encontrar a literatura para estudar o modelo de dados ISIS. Assim como outros modelos de dados anteriores ao relacional, o modelo de dados ISIS não se beneficiou de uma formalização teórica antes de sua implementação. As únicas fontes específicas sobre ISIS são os manuais técnicos dos diversos aplicativos da família ISIS, e neles não há referência a qualquer modelo teórico. Entretanto, o modelo semiestruturado, formalizado nos anos 90, é suficientemente próximo para possibilitar a extrapolação de muitos resultados de pesquisa. A melhor definição breve para o modelo de dados semiestruturado que encontramos foi esta:

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 or 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.¹⁹

Como veremos em 2.5 Modelo de dados ISIS (p. 28), o modelo de dados ISIS se encaixa nesta definição. Esta descoberta abriu as portas para muita literatura relativa ao modelo semiestruturado. Aqui mencionaremos apenas alguns trabalhos de maior relevância como ponto de partida para uma revisão mais aprofundada da literatura em um trabalho futuro.

2.3.1 O livro vermelho de Hellerstein e Stonebraker

A referência inicial mais importante para esta pesquisa foi *Readings in Database Systems* (HELLERSTEIN, 2005), conhecido como “the red book” (o livro vermelho), uma coletânea de artigos utilizada como livro-texto em disciplinas sobre bancos de dados na University of California at Berkeley²⁰.

O valor deste livro está na diversidade de temas e na abordagem voltada a fundamentos, muitas vezes independente de modelos de dados específicos, como se pode ver na lista de

¹⁹ SUCIU, 2009, p. 2601. 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 pode ser outro registro ou coleção. Além disso, as coleções podem ser heterogêneas, ou seja, podem 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.”

²⁰ Comentários de um dos autores (Hellerstein) sobre uso deste livro em Berkeley: <<http://redbook.cs.berkeley.edu/faq.html>>. Acesso em 19 nov. 2010.

artigos distribuídos entre seus nove capítulos²¹.

No artigo do capítulo introdutório, *What Goes Around Comes Around*, os editores da coletânea, Michael Stonebraker e Joseph M. Hellerstein deixam evidente sua inclinação para uma abordagem mais abrangente. Este artigo é um panorama bastante opinativo sobre a história dos modelos de dados, dividida em nove “eras”, a saber:

- Hierarchical (IMS): late 1960's and 1970's
- Network (CODASYL): 1970's
- Relational: 1970's and early 1980's
- Entity-Relationship: 1970's
- Extended Relational: 1980's
- Semantic: late 1970's and 1980's
- Object-oriented: late 1980's and early 1990's
- Object-relational: late 1980's and early 1990's
- Semi-structured (XML): late 1990's to the present²²

O resumo deste artigo termina com uma provocação:

Unfortunately, the main proposal in the current XML era bears a striking resemblance to the CODASYL proposal from the early 1970's, which failed because of its complexity. Hence, the current era is replaying history, and “what goes around comes around”. Hopefully the next era will be smarter.²³

No mesmo texto, são introduzidos os termos “schema first” e “schema last”, uma distinção fundamental:

In a “schema first” system the schema is specified, and instances of data records that conform to this schema can be subsequently loaded. Hence, the data base is always consistent with the pre-existing schema, because the DBMS rejects any records that are not consistent with the schema.

In this class of [semistructured] proposals the schema does not need to be specified in advance. It can be specified last, or even not at all. In a “schema last” system, data instances must be self-describing, because there is not necessarily a schema to give meaning to incoming records.²⁴

21 Sumário do livro na Web: **Contents: Readings in Database Systems, 4th Edition**. Disponível em <<http://redbook.cs.berkeley.edu/bib4.html>>. Acesso em 19 nov. 2010.

22 HELLERSTEIN, 2005, p. 2.

23 idem. “Infelizmente, a principal proposta para a atual era do XML tem grande semelhança com a proposta CODASYL do início dos anos 1970, que fracassou devido à sua complexidade. Assim, a era atual está repetindo a história, e 'o que vai, volta'. Oxalá a próxima era será mais inteligente”. Nossa tradução.

24 HELLERSTEIN, 2005, p. 30. “Em um sistema “esquema primeiro”, o esquema é especificado, e instâncias de registros de dados que estejam em conformidade com este esquema podem ser posteriormente carregadas. Assim, a base de dados está sempre consistente com o esquema preexistente, porque o SGBD rejeita

Conforme veremos, a família ISIS, bem como os bancos de dados orientados a documento CouchDB e MongoDB, são “schema last”, em contraste com todos os sistemas relacionais que são “schema first”.

Ao longo do artigo, Stonebraker e Hellerstein apresentam as lições aprendidas de cada “era”. Para a era *Semi Structured Data*, uma das lições é:

Lesson 16: Schema-last is probably a niche market .²⁵

Certamente, o ISIS é um sistema “esquema por último” que encontra-se restrito ao nicho das aplicações em bibliotecas e centros de documentação. O futuro dirá se o mesmo vai acontecer com outras implementações modernas do modelo semiestruturado.

As outras lições desta “era” são relativas a questões específicas de XML. Os autores têm uma visão bastante crítica do modelo semiestruturado, mas suas principais objeções estão focadas em aspectos relativos a XML e tecnologias associadas. Por exemplo, em relação à definição formal de esquemas de dados em XML, as previsões dos autores são pessimistas, com uma exceção – o único cenário otimista é:

Scenario 2: A “data-oriented” subset of XMLSchema will be proposed that is vastly simpler.²⁶

A qualificação “data-oriented” toca em um ponto central da complexidade de XML como modelo de dados, a dualidade “narrativa versus dados”, que abordaremos na seção 2.6 XML e JSON. Aparentemente, o mercado ouviu as advertências de Stonebraker e Hellerstein, porque, de forma geral, os sistemas de banco de dados semiestruturados que estão sendo lançados recentemente não pretendem lidar com XML nativamente, já que adotaram um modelo de dados parecido com JSON, que evita muito da complexidade associada ao XML. Ou, então, optaram por modelos de dados ainda mais simples.

No capítulo sobre *Web Services and Databases*, além de um artigo sobre um motor de busca de larga escala, de autoria dos fundadores do Google, encontramos estes dois (como citados no índice de HELLERSTEIN, 2005):

- Serge Abiteboul. Querying Semi-Structured Data. Proc. ICDT, 1997, 1-18.

quaisquer registros que não são compatíveis com o esquema. Nesta classe de propostas [semiestruturadas] o esquema não precisa ser especificado com antecedência. Pode ser especificado depois, ou mesmo não ser especificado. Em um sistema “esquema por último”, instâncias de dados devem ser autodescritivas, porque não existe necessariamente um esquema para dar sentido aos registros inseridos.” Nossa tradução.

25 HELLERSTEIN, 2005, p. 37. “Lição 16: Esquema-por-último provavelmente é um nicho de mercado.”

26 HELLERSTEIN, 2005, p. 35.

- Roy Goldman Jennifer Widom. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. Proc. VLDB, 1997, 436-445.

Foi no livro vermelho que encontramos a palavra-chave mais importante para o restante da pesquisa: *semi-structured* ou *semistructured*. Infelizmente, o termo aparece grafado das duas formas nos artigos, tanto em inglês (como se vê nos títulos acima), quanto em português. Pelo acordo ortográfico da língua portuguesa, que entrou em vigor no Brasil em 2009, o correto é “semiestruturado” e esta é a grafia que adotamos. Nas citações, manteremos a grafia da fonte.

2.3.2 Referência específica: Data on The Web

A partir da bibliografia do livro vermelho, encontramos a primeira obra inteiramente dedicada ao tema do modelo de dados semiestruturado, *Data on the Web*:

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

Data on the Web reúne resultados de pesquisas realizadas na década de 1990 no University of Pennsylvania Database Research Group (UPENN, 2010), no AT&T Laboratories e pelas equipes dos projetos Lore, na Stanford University (INFOLAB, 2010), e Verso no INRIA. Todas estas pesquisas começaram antes do formato XML ser anunciado em novembro de 1996. A relação com XML foi feita depois. No modelo de dados descrito, não há sinal dos conceitos de atributos e conteúdo misto, que XML herdou de SGML, conforme discutiremos em 2.6.2 Comparação entre JSON e XML (p. 35). O sucesso do XML abafou um pouco estes resultados, mas é evidente, até pela notação empregada em *Data on the Web* para representar dados semiestruturados, que os autores imaginavam algo muito parecido com JSON. Eis um exemplo de dado semiestruturado utilizando a sintaxe de *ssd-expressions* apresentada no livro²⁷:

```
{name: {first: "Alan", last: "Black"},
  tel: 2157786,
  email: "agb@abc.com"
}
```

Neste exemplo, a *ssd-expression* por coincidência tem exatamente a mesma sintaxe de um objeto ou dicionário em JavaScript. O equivalente em JSON teria aspas duplas em volta do nome dos atributos ("name", "first", "tel" etc.).

²⁷ ABITEBOUL, 1999, exemplo na p. 11, definição de *ssd-expressions* na p. 18.

Porém, a sintaxe de *ssd-expressions* tem uma diferença importante em relação a JSON: uma maneira de definir explicitamente identificadores de objetos, permitindo que o valor de um campo seja uma referência a um objeto definido na mesma expressão. Por exemplo²⁸:

```
{ person: &o1{ name: "Mary",
               age: 45,
               child: &o2,
               child: &o3
             },
  person: &o2{ name: "John",
               age: 17,
               relatives: { mother: &o1,
                           sister: &o3
                         }
             },
  person: &o3{ name: "Jane",
               country: "Canada"age: 17,
               mother: &o1
             }
}
```

Nesta estrutura, cada objeto pessoa é declarado com um identificador (*&o1*, *&o2* e *&o3*).

Quando estes identificadores aparecem como valores de atributos, como em *sister: &o3*, isso denota uma referência ao objeto, ou seja, o valor de *sister* neste caso é o próprio objeto *&o3*.

Vale também notar que esta estrutura possui três atributos *person*, e na primeira instância de *person* o atributo *child* aparece duas vezes, exemplificando o fato de que em uma *ssd-expression* os atributos podem ser repetidos.

Data on the Web divide-se em quatro partes. A primeira apresenta o modelo de dados semiestruturado, compara-o ao modelo relacional e discute suas semelhanças e diferenças em relação ao modelo de dados implícito no formato XML. A segunda parte apresenta os conceitos comuns às linguagens de consulta para sistemas semiestruturados, e apresenta algumas delas em particular, como Lorel e UnQL, e faz conexão entre estas ideias e as notações XSL e XML-QL. A terceira parte apresenta avanços na aplicação de tipos de dados formais ao modelo semiestruturado. A parte final discute questões de implementação e apresenta dois sistemas: Lore, um SGBD semiestruturado, e Strudel, um CMS (content management system, sistema de gestão de conteúdo para Web sites).

²⁸ ABITEBOUL, 1999, exemplo na p. 16.

2.3.3 Referência Específica: Semistructured Database Design

O segundo livro encontrado que é totalmente focado no presente tema foi:

TOK, Wang Ling; LEE Mong Li; DOBBIE, Gillian. **Semistructured Database Design**. Boston: Springer Science, 2005.

As principais contribuições desta obra, para citar os próprios autores, são:

- uma comparação entre modelos de dados para projetar a organização persistente de dados semiestruturados;
- a introdução de um modelo de dados, chamado ORA-SS – Object Relationship Attribute Data Model for SemiStructured Data (Modelo de Dados de Relacionamento de Atributos para Dados Semiestruturados), que acreditamos representar a semântica necessária para o projeto de organizações para a armazenagem de dados semiestruturados;
- um algoritmo para a extração de um esquema de dados a partir de uma instância de dados semiestruturada, tal como um documento XML;
- um algoritmo para a normalização de esquemas semiestruturados;
- um conjunto de regras para a validação de visões (*views*) criadas sobre uma instância semiestruturada subjacente;
- um algoritmo para a desnormalização de esquemas semiestruturados.²⁹

O modelo ORA-SS proposto pelos autores inclui uma notação para diagramas.

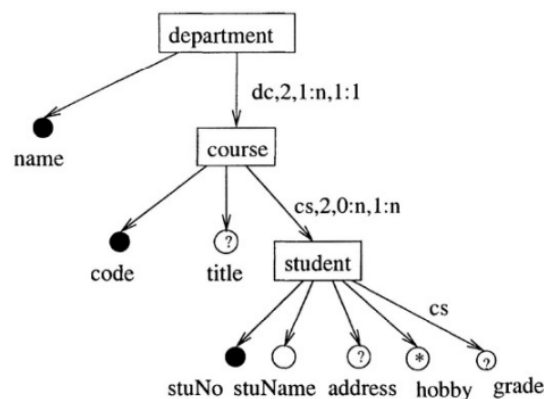


Figura 1: Exemplo de diagrama ORA-SS, reprodução de TOK, 2005, p. 31, figura 2.14

²⁹ TOK, 2005, p. xvi. Tradução nossa.

2.4 Família ISIS

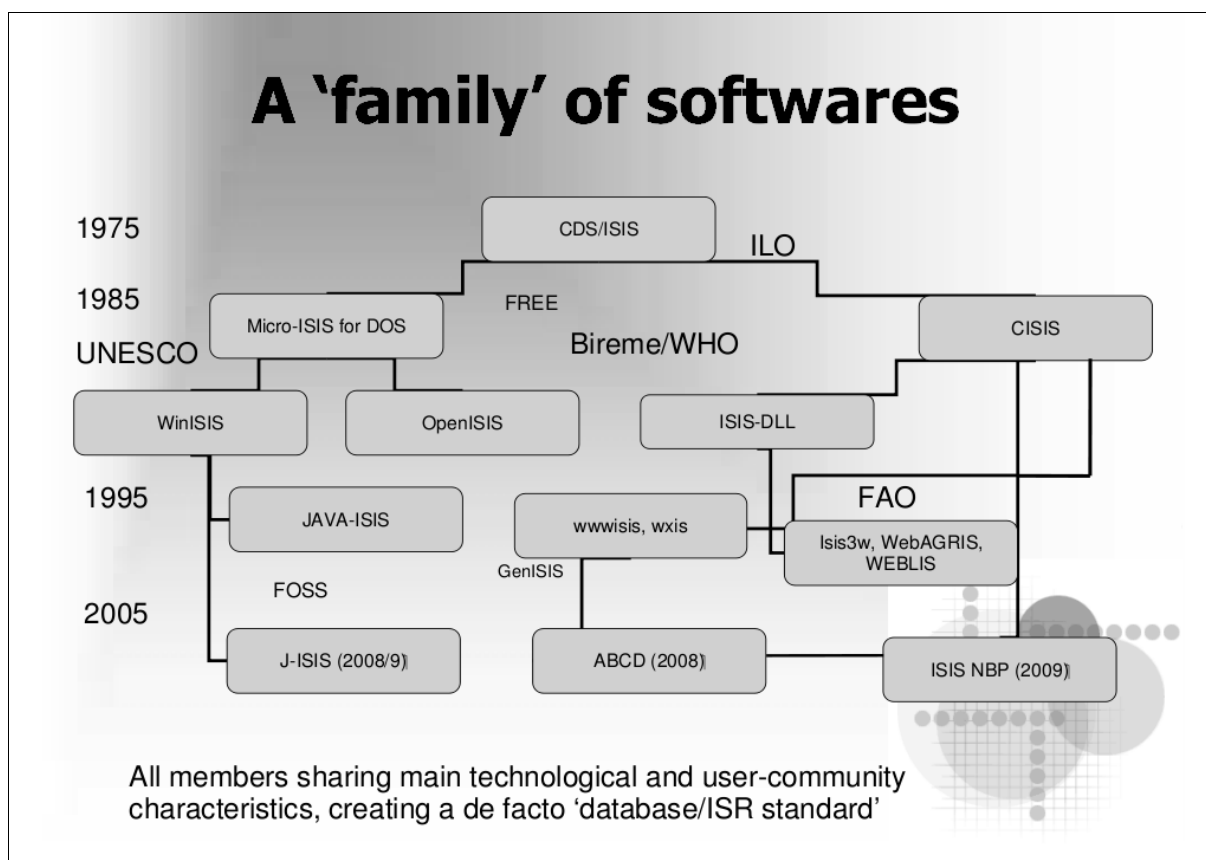


Figura 2: Slide apresentado por Egbert de Smet por ocasião de sua palestra de abertura no III Congresso Internacional de Usuários de CDS/ISIS, Rio de Janeiro, set. de 2008 (DE SMET 2008)

A gênese da família de sistemas ISIS está no Integrated Set of Information Systems (conjunto integrado de sistemas de informação) para computadores de grande porte, desenvolvido para uso interno na Organização Internacional do Trabalho (OIT, ou ILO na sigla em inglês) em 1965, posteriormente adaptado e ampliado por Giampaolo Del Bigio para uso no Computerized Documentation System (sistema de documentação computadorizado) da UNESCO. Esta é a origem da sigla CDS/ISIS.³⁰

Porém, sua popularização se deu a partir do MicroISIS, a transposição do CDS/ISIS para microcomputadores, feita por Giampaolo Del Bigio, ainda na UNESCO. O MicroISIS opera sobre o sistema operacional DOS, e continua sendo utilizado para catalogação em muitas bibliotecas, graças à compatibilidade do console do sistema Windows com o DOS.

Os principais documentos técnicos sobre o CDS/ISIS a que tivemos acesso foram o *Mini-*

30 LOPES, Francisco. **Historia_ISIS.doc**. Memorando interno da BIREME/OPAS/OMS.

micro CDS/ISIS Reference Manual (Version 2.3), de março de 1989 (UNESCO 1989), o *Suplement do CDS/ISIS Reference Manual – CDS/ISIS Version 3.0* (ORNAGER 1993) e *The CDS/ISIS Handbook* (BUXTON 1994).

2.4.1 CISIS

Em 1995, no I Congresso Mundial de Usuários de CDS/ISIS, a BIREME/OPAS/OMS anunciou o CISIS, um motor de banco de dados escrito em linguagem C, compatível com os formatos de arquivos do MicroISIS, porém otimizado para as necessidades específicas da BIREME, especialmente a publicação de bases de dados como Medline e LILACS em CD-ROM, posteriormente na Web. Sobre CISIS, a BIREME/OPAS/OMS oferece os manuais: *Conceitos Básicos de Bases de Dados CDS/ISIS: Iniciando o Uso do CISIS* (BIREME 2005), *Linguagem de Formato CISIS* (BIREME 2006a) e *Utilitários CISIS - Manual de Referência* (BIREME 2006b).

2.4.2 ISIS-NBP: ISIS Network Based Platform

Desenvolvimentos recentes sobre a plataforma ISIS, bem como propostas de evolução dela no contexto da BIREME/OPAS/OMS, têm sido realizados no wiki e no repositório público de código do projeto ISIS-NBP – ISIS Network Based Platform, ou Plataforma ISIS Baseada em Rede (BIREME, 2010a)³¹. Entre estes desenvolvimentos, destacamos alguns que foram fruto da presente pesquisa:

- *isis2json.py*: utilitário de conversão de bases ISIS para o formato JSON³²
- ISIS-JSON: proposta de padronização da representação de registros ISIS em formato JSON³³
- ISIS Data Model API (ISIS-DM): prova de conceito de interface de programação (API) para definição de esquemas de dados ISIS independente de mecanismo de persistência³⁴
- *schematize.py*: utilitário gerador de esquema de dados ISIS-DM a partir da análise de uma massa de dados existente³⁵

31 Disponível em: <<http://reddes.bvsaude.org/projects/isisnbp>>. Acesso em 22 nov. 2010.

32 Disponível em: <<http://reddes.bvsalud.org/projects/isisnbp/wiki/Tools>>. Acesso em 22 nov. 2010.

33 Disponível em: <<http://reddes.bvsalud.org/projects/isisnbp/wiki/ISIS-JSON>>. Acesso em 22 nov. 2010.

34 Disponível em: <<http://reddes.bvsalud.org/projects/isisnbp/wiki/ISIS-DM>>. Acesso em 22 nov. 2010.

35 Disponível em: <<http://reddes.bvsalud.org/projects/isisnbp/wiki/Schematizer>>. Acesso em 22 nov. 2010.

2.4.3 WinISIS, J-ISIS e outras referências

Paralelamente à criação do CISIS pela BIREME/OPAS/OMS, Del Bigio liderou o desenvolvimento do WinISIS, ou CDS/ISIS para Windows, a respeito do qual a melhor documentação que encontramos é *The CDS/ISIS for Windows Handbook* (BUXTON, 2001). Com a aposentadoria de Del Bigio em 1998³⁶, o desenvolvimento do WinISIS perdeu impulso, mas um outro projeto da UNESCO ganhou fôlego: o J-ISIS (DAUPHIN, 2010). Inicialmente criado como uma transposição multiplataforma do WinISIS em linguagem Java, o J-ISIS é hoje um sistema mais versátil, que inclui não só um aplicativo desktop nos moldes do WinISIS, mas também uma RIA (Rich Internet Application) chamada Web-JISIS (J-ISIS, 2010).

Além dos materiais sobre ISIS-NBP e J-ISIS, muito pouco se escreveu sobre ISIS nos últimos cinco anos. Duas exceções são o livro *AACR2, MARC21 and WINISIS: A Compatibility Study* (SHEWALE, 2009), e o wiki *Oráculo*³⁷.

36 Mensagem de Giampaolo Del Bigio aos participantes do encontro de MicroISIS em Montevidéu, 1998, disponível em <<http://www.axp.mdx.ac.uk/~alan2/bigio.htm>>. Acesso em 22 nov. 2010.

37 Disponível em: <<http://www.oraculo.inf.br>>. Acesso em 22 nov. 2010.

2.5 Modelo de dados ISIS

A família ISIS foi criada especificamente para a catalogação de documentos³⁸, portanto é natural que seja bem adaptada a diversos contextos de uso em biblioteconomia e ciência da informação. Informalmente, podemos observar que o “modelo de dados ISIS” se aproxima dos modelos de dados dos registros MARC e da norma ISO-2709³⁹. Entre as características destes modelos, merecem destaque:

- aceitar a repetitividade de campos de dados;
- permitir a utilização de subcampos⁴⁰;

Em princípio, todos os campos são opcionais⁴¹, e, no limite, cada registro pode ter uma estrutura de campos e subcampos diferente de todos os demais. Entretanto, tipicamente todos os registros de um objeto banco de dados têm em comum pelo menos um mesmo subconjunto de campos. Existe um alto grau de compatibilidade, no nível lógico, entre registros ISIS e registros ISO-2709, a ponto de que muitas ferramentas e aplicativos ISIS importam ou exportam registros ISO-2709, e este formato é muito utilizado para transmissão de dados entre bases ISIS.

Em contraste, nos sistemas de banco de dados relacionais os registros são armazenados em tabelas e cada tabela tem um esquema de dados que define uma estrutura única para todos os registros nela contidos.

Como em outros sistemas de banco de dados semiestruturados, a falta de definições globais de esquema é suprida pela inclusão, junto com os dados de cada registro, de marcadores ou tags que identificam os campos. No CISIS e no WinISIS, o tag pode ser um número de 1 a 32767. Por exemplo, na metodologia LILACS, o tag 10 identifica um campo de Autor Pessoal, em um registro de nível analítico. Campos repetitivos são criados pelo uso repetido de um mesmo tag, da mesma forma que um elemento repetitivo em XML é representado pela repetição de um tag. Em um registro LILACS com três autores pessoais, o tag 10 aparece três vezes.

Campos opcionais podem ser omitidos; conseqüentemente, é desnecessário o uso de

38 BUXTON, 200, p. 3.

39 ISO2709, 2008.

40 BIREME, 2005 (Conceitos Básicos de Bases de Dados CDS/ISIS: Iniciando o Uso do CISIS Versão 3.x), p. 8.

41 Alguns aplicativos ISIS, como o WinISIS e o LILDBI-Web, permitem definir campos obrigatórios, mas esta restrição é implementada apenas no aplicativo, pois não existe modo de especificar diretamente no objeto banco de dados esse tipo de restrição.

marcadores nulos (como o NULL da linguagem SQL) para indicar a ausência de valores.

No CISIS o tag pode ser um número de 1 a 32767, mas na norma ISO-2709 os tags são alfanuméricos com 3 posições⁴². Isso significa que registros ISIS com campos de quatro dígitos não podem ser convertidos para ISO-2709, e, por outro lado, registros ISO-2709 com tags alfabéticos também não podem ser convertidos para ISIS. Isso explica porque muitas bases ISIS, na prática, não tenham tags maiores que 999, e, pelo mesmo motivo, porque a metodologia LILACS utiliza tags de no máximo 3 dígitos.

É interessante notar que a combinação de três dígitos ou letras possibilita 46656 combinações de tags, de acordo com a norma ISO-2709. Portanto, com algum tipo de codificação seria possível representar todos os 32767 tags numéricos dos registros ISIS. Porém, desconhecemos qualquer ferramenta que faça a conversão desta maneira.

2.5.1 Níveis de organização de uma base ISIS

Em todos os sistemas da família ISIS, um objeto banco de dados, do ponto de vista lógico⁴³, é uma coleção de registros, possivelmente heterogêneos. Ou seja, entre o objeto banco de dados e os registros não existe o nível intermediário da relação (ou tabela), como há nos sistemas relacionais. Por outro lado, os registros de um objeto banco de dados podem ter estruturas diferentes, ou seja, podem existir registros de diversos tipos (com esquemas de dados distintos), ao contrário do que acontece no sistema relacional, onde todos os registros de uma tabela têm a mesma estrutura.

Embora não exista o conceito de tabela para agrupar registros semelhantes em uma base ISIS, na prática as metodologias costumam indicar o uso de um ou mais campos obrigatórios para identificar o tipo de cada registro. Na metodologia LILACS, a combinação dos campos 5 e 6 define o tipo do registro, e esta informação é utilizada pelas aplicações para saber quais campos podem ser esperados em cada instância de registro. Existe inclusive uma base auxiliar em LILACS (a xLILACS), na qual a própria semântica dos demais campos varia conforme o tipo informado no tag 5 do registro.

Finalmente, no modelo de dados do ISIS existe o conceito de subcampo: um campo pode opcionalmente ser subdividido em subcampos, através do uso de marcadores especiais. Na

42 ISO2709, 2008, p. 5.

43 Fisicamente, a maioria dos sistemas ISIS armazena os dados de um objeto banco de dados em dois arquivos com extensões .MST e .XRF (BIREME 2005).

sintaxe padrão, os marcadores são formados pelo caractere ^ (circunflexo) seguido de uma letra da tabela ASCII (portanto, sem acento) ou de um dígito de 0 a 9. Por exemplo, ^r e ^3 são marcadores de subcampos utilizados no campo 10 da metodologia LILACS. A linguagem usada para definir índices e formatar resultados permite acessar cada subcampo de forma independente. Eis um exemplo de campo com subcampos em formato ISIS⁴⁴:

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

Os delimitadores « e » não fazem parte da sintaxe, mas são exibidos pelas ferramentas do CISIS para indicar o início e o fim do conteúdo do campo propriamente dito. Note que no exemplo acima há 5 subcampos, com códigos **1**, **2**, **p**, **c**, e **r**. O subcampo **1** contém o texto “USP”. Porém o conteúdo principal do campo, que é o nome do autor “Lewis Carroll” não está em nenhum subcampo.

Esta posição privilegiada não tem um nome na documentação do CISIS, mas no *Diccionario de Datos del Modelo LILACS Version 1.6a* (BIREME 2008a) é usado o símbolo * para denotar a parte do campo que não pertence a nenhum subcampo. Este símbolo está ligado à sintaxe da linguagem de formato do ISIS, e seu uso neste contexto é ambíguo, pois nesta linguagem o operador * devolve o primeiro subcampo, que pode ou não ter uma marca de subcampo, ou seja, em «alfa^1beta», o * corresponde a “alfa” mas em «^1beta^2gama» o operador * devolve “beta”.

O *Mini-micro CDS/ISIS Reference Manual (Version 2.3)*, explica esta questão assim:

Note that the first subfield of a subfielded field need not have a subfield delimiter, *provided that it is always present*. For example, if in a title field you wanted to use a subfield for the subtitle, the title part of the field, which will obviously always be present, need not have an explicit delimiter. Thus the following entry for this field would be possible⁴⁵: «Il nome della rosa^bNaturalmente, un manoscritto»

Adotaremos neste trabalho o termo **subcampo principal** para nos referirmos a esta parte de um campo.

44 Este exemplo didático não é aderente à metodologia LILACS, que exige o nome em ordem inversa e todos os subcampos de afiliação escritos por extenso, entre outras regras.

45 UNESCO, 1989, p. 33 “Note que o primeiro subcampo de um campo com subcampos não precisa ter um delimitador de subcampo, *desde que esteja sempre presente*. Por exemplo, se em um campo de título for desejável usar um subcampo para o subtítulo, a parte título do campo, que obviamente sempre estará presente, não precisa ter um delimitador explícito. Assim, a seguinte entrada para este campo seria possível.”. Nossa tradução.

A sintaxe de marcação de subcampos descrita tem algumas consequências:

- Sem repetir marcadores, um campo pode ter até 37 subcampos, contando o subcampo principal, 26 subcampos com marcadores alfabéticos e 10 com marcadores numéricos.
- É possível, em tese, existirem subcampos repetitivos, bastando, para isso, repetir um mesmo marcador de subcampo (porém, a linguagem de formato só consegue acessar a primeira ocorrência de cada subcampo).
- Não é possível aninhar subcampos, ou seja, não existe o conceito de sub-subcampo em um registro ISIS. Em outras palavras, os campos são divisíveis mas os subcampos são “atômicos”.

Vale notar que existe uma articulação interessante entre as ideias de campo repetitivo e subcampo: a existência de subcampos dá maior utilidade aos campos repetitivos. Por exemplo, se não existissem subcampos, mas apenas campos repetitivos, o registro de vários autores e seus países seria mais complicado. Uma alternativa seria usar um campo, digamos, campo 10, para os nomes, e outro campo, digamos, 210, para os países dos autores, de modo que cada ocorrência do campo 10 seria associada a uma ocorrência do campo 210.

Obviamente, este esquema é frágil, pois se um usuário remover um dos autores e não remover o país correspondente, a associação entre os campos fica arruinada. Portanto, a existência de subcampos no modelo ISIS aumenta bastante a utilidade e praticidade dos campos repetitivos.

2.5.2 ISIS Formatting Language: a linguagem de extração de dados

É importante notar que no formato interno de armazenagem do ISIS não existe uma separação entre os subcampos: o conteúdo inteiro de um campo é guardado como uma única string, com os marcadores embutidos. O suporte a subcampos na plataforma ISIS se dá na sua linguagem de formato, ISIS Formatting Language ou IFL (BIREME 2006a). Vejamos alguns exemplos de sua sintaxe, através de exemplos com o comando `v` (seletor de campo), considerando um registro com este campo 10:

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

expressão IFL	resultado	descrição
<code>v10[1]</code>	Lewis Carroll^1USP^2ECA^pBrasil^cSão Paulo^rEditor	ocorrência 1 do campo 10 no registro
<code>v10^p[1]</code>	Brasil	subcampo p da ocorrência 1 do campo 10 no registro
<code>v10^p[1]*0.3</code>	Bra	substring iniciando no deslocamento 0 (zero) com tamanho 3 do subcampo p da ocorrência 1 do campo 10 no registro

Tabela 2: Pequena amostra da sintaxe da linguagem de formato ISIS. Os três exemplos demonstram usos do comando `v`, o seletor de campo. Sua sintaxe é rica, mas não permite recuperar subcampos repetitivos

A linguagem de formato tem duas utilidades principais⁴⁶:

1. formatação de campos e subcampos para exibição, impressão ou exportação;
2. extração de campos e subcampos para geração de índices de busca e ordenação.

Em um SGBD relacional, as funções da IFL são desempenhadas pela linguagem SQL. No CouchDB, o outro sistema semiestruturado que utilizamos neste trabalho, um interpretador JavaScript vem integrado ao SGBD e desempenha estas funções, entre outras.

A IFL é bastante flexível mas sua sintaxe é lacônica. Sua legibilidade também é prejudicada pelo fato de que os tags de campos são sempre numéricos, marcadores de subcampos são limitados a um caractere alfanumérico, e a linguagem não possui mecanismos de abstração que permitam ao usuário definir funções ou atribuir identificadores mais amigáveis aos resultados das expressões.⁴⁷

46 UNESCO, 1989, p. 41

47 RAMALHO, 2010, p. 45. Tradução do autor.

2.6 XML e JSON

Nosso tratamento de XML (Extensible Markup Language) e JSON (JavaScript Object Notation) é inspirado por esta colocação em *Data on the Web*:

To summarize, we can think of XML as a physical representation – a data format – or as a logical representation. In this book we will promote de second view, for the logical representation is that of semistructured data. Indeed, much of this book will be concerned with the development of query languages for this representation, just as relational query languages have been developed for another highly successful logical representation, relational databases.⁴⁸

Portanto, este é o enfoque que daremos a XML e JSON. Por isso, falaremos às vezes sobre “o modelo de dados JSON”, em referência a uma representação lógica parecida, porém mais simples que “o modelo de dados XML”, que é o modelo de dados semiestruturado mais conhecido atualmente. E quando nos referirmos ao modelo de dados ISIS, estaremos pensando em um modelo de dados ainda mais simples que o JSON, mas ainda assim um exemplo de modelo de dados semiestruturado.

Tanto XML quanto JSON são padrões internacionais com fundamentos sólidos. XML é formalizado em recomendações do W3C⁴⁹, e é baseado na SGML (Standard Generalized Markup Language), que é normatizada como ISO-8879⁵⁰. JSON é formalizado em no RFC-4627 (CROCKFORD 2006b) e é baseado em um subconjunto da sintaxe de JavaScript, que é normatizada como ECMA-262 (ECMA 2009) e como ISO/IEC 16262.

XML é um padrão mais antigo, tem maior aceitação no mercado e na academia, e é também mais ambicioso, pretendendo ser um formato de intercâmbio que atenda as necessidades de troca de documentos semiestruturados complexos e variados, bem como o intercâmbio de dados em forma de registros relativamente mais simples, estruturados e uniformes.

JSON tem um objetivo mais modesto, de servir apenas como padrão para a transmissão de registros. Seu uso se disseminou com a arquitetura de aplicações AJAX, como formato de transporte de dados entre servidores Web e navegadores (ironicamente, o X de AJAX se refere

48 ABITEBOUL, 1999, p. 7. “Para resumir, podemos pensar em XML como uma representação física – um formato de dados – ou como uma representação lógica. Neste livro, promoveremos a segunda visão, porque a representação lógica é a representação de dados semiestruturada. De fato, muito deste livro se ocupará do desenvolvimento de linguagens de consulta para esta representação, assim como linguagens de consulta relacionais foram desenvolvidas para uma outra representação lógica altamente bem sucedida, as bases de dados relacionais.”. Nossa tradução.

49 Recomendações disponíveis em <<http://www.w3.org/XML/Core/>>. Acesso em 22 nov. 2010.

50 ISO 8879:1986 Information processing — Text and office systems — Standard Generalized Markup Language (SGML).

a XML, mas hoje o formato JSON é o preferido neste tipo de aplicação, por ser mais compacto e de interpretação mais fácil no navegador, graças ao uso da sintaxe de JavaScript). O desafio desta parte do levantamento não foi encontrar documentação sobre XML ou JSON, mas sim encontrar fundamentação para este crescente interesse por JSON, que se reflete não só na arquitetura AJAX, mas também no suporte a este formato por bancos de dados semiestruturados modernos, como veremos na seção 2.7.1 (p. 40).

2.6.1 XML narrativo versus XML de dados

Parte da complexidade de XML advém de suas origens e seu uso em um espectro muito largo de aplicações. Assim como HTML, a linguagem XML teve como ponto de partida a SGML, uma linguagem para marcação de documentos como artigos e monografias.

XML was designed for narrative documents meant to be read by humans: books, novels, plays, poems, technical manuals, and most specifically web pages. Its use for record-oriented data was a happy accident.⁵¹

Os usos narrativo e orientado a registros de XML podem ser comparados assim:

XML narrativo	XML de dados
Hierarquia de conteúdos muito flexível	Hierarquia de conteúdos muito rígida
Documentos podem ser extensos	Documentos são comumente curtos, mas podem ser extensos
Documentos são transformados para exibição como texto legível	Documentos podem não ser exibidos como texto legível
O ator primário é um ser humano	O ator primário é um processo computacional

Tabela 3: Características que, tipicamente, distinguem documentos XML narrativos de documentos XML de dados. (Tabela adaptada de MAUGET, L. E. Choosing an appropriate XML technology⁵²)

51 HAROLD, E. R. **Effective XML**. p. 75, item 13. “XML foi projetada para documentos narrativos a serem lidos por humanos: livros, romances, peças, poemas, manuais técnicos e, especificamente, páginas web. Seu uso para dados estruturados como registros foi um acidente feliz.”

52 MAUGET, L. E. **Choosing an appropriate XML technology** in IBM DeveloperWorks XML and Related Technologies certification prep, Part 5: XML testing and tuning. Disponível em <<http://www.ibm.com/developerworks/xml/tutorials/x-cert1425/section2.html#N101F2>> ou <<http://bit.ly/akcqIA>>. Acesso em 19 nov. 2010.

2.6.2 Comparação entre JSON e XML

JSON nasceu como uma alternativa mais simples para substituir o uso de XML, mas não em qualquer tipo de aplicação, e sim num subconjunto delas. Antes de apresentar mais formalmente o formato JSON, vamos apresentá-lo informalmente por meio de comparações com XML, a fim de deixar claro os problemas que JSON tenta solucionar.

Por ter um objetivo mais limitado, JSON é mais simples do que XML. Por exemplo, apenas a codificação de caracteres UTF-8 é aceita, e esta é a codificação recomendada pelo W3C para uso geral⁵³. Em XML é possível especificar uma codificação de caracteres arbitrária.

Do ponto de vista estrutural, JSON evita duas complexidades de XML que são úteis na marcação de documentos narrativos, mas não na transmissão de registros de bases de dados: a dualidade entre conteúdos e atributos, e o conteúdo misto. Vejamos cada uma destas questões.

Para exemplificar a dualidade entre conteúdos e atributos, suponha que temos um registro de aluno com campos “nome” e “número de matrícula”. Duas variantes possíveis em XML são:

```
<aluno>
  <matricula>123456</matricula>
  <nome>Fulano de Tal</nome>
</aluno>
```

ou então:

```
<aluno matricula="123456">
  <nome>Fulano de Tal</nome>
</aluno>
```

Há outras possibilidades.

Em JSON, a representação natural é uma só:

```
{ "aluno" : { "matricula" : "123456",
             "nome" : "Fulano de Tal" } }
```

A ideia de atributos em XML foi herdada de SGML, uma linguagem criada para a marcação de documentos para publicação eletrônica. No contexto de publicação, um atributo é útil para associar ao texto metadados ou parâmetros que não fazem parte do fluxo principal de texto, como por exemplo, associar um estilo visual a um parágrafo, ou uma URI a uma referência,

53 “The examples above show declarations for UTF-8 encoded content. This is likely to be the best choice of encoding for most purposes, but it is not the only possibility.” Disponível em: <<http://www.w3.org/International/O-charset>>, acesso em 15 nov. 2010. Ver também **Recommended list of Doctype declarations** <<http://www.w3.org/QA/2002/04/valid-dtd-list.html>>, acesso em 15 nov. 2010.

enfim, informações que não aparecerão para o ser humano que lerá do documento.

Porém, ao usar XML como formato para representação de registros, a escolha entre representar um dado como conteúdo ou atributo provoca dúvidas e discussões recorrentes:

There's a recurring mild flame war on the xml-dev mailing list about when one should use attributes and when one should use elements. There's a slightly hotter one about whether one should ever use attributes at all.⁵⁴

JSON não implementa o conceito de atributos, evitando esse dilema.

Outra complexidade estrutural do XML que JSON evita completamente é o chamado conteúdo misto. Eis um exemplo de XML com conteúdo misto:

```
<p>O uso de XML para dados organizados em <strong>registros</strong>
é um mero acidente.</p>
```

No trecho acima, temos um nodo `<p>` com três nodos filhos:

1. um nodo texto com o conteúdo “O uso de XML para dados organizados em ”;
2. um elemento `` com o conteúdo “registros”;
3. um nodo texto com o conteúdo “ é um mero acidente.”

No formato JSON simplesmente não existe conteúdo misto. Caso fosse necessário representar o texto acima em JSON, teríamos que recorrer à criação de novos itens, seguindo alguma convenção. Mas o caso é que JSON foi explicitamente concebido para representar registros que são fundamentalmente associações entre nomes e valores, e é por isso que não possui uma forma de representar valores anônimos desassociados, como os nodos texto do XML.

Portanto, JSON não é tão conveniente quanto XML para representar documentos narrativos, mas para expressar estruturas de dados ele é consideravelmente mais prático:

[XML features] The ability to represent the most general computer science data structures: records, lists and trees.⁵⁵

This is the most significant difference. While there are transformations which allow XML to express, JSON expresses them directly. JSON's simple values are the same as

54 “Existe um debate recorrente na lista xml-dev sobre quando se deve usar atributos, e quando se deve usar elementos. Existe um debate um pouco mais quente sobre se atributos deveriam ser usados em qualquer caso.” HAROLD, 2004, p. 69

55 **XML: Extensible Markup Language** in: Network Dictionary. Disponível em <http://www.networkdictionary.com/software/xml.php>. Acesso em 15 nov. 2010. “XML oferece a possibilidade de representar as estruturas de dados mais gerais da ciência da computação: registros, listas e árvores.”. Nossa tradução.

used in programming languages. JSON's structures look like conventional programming language structures. No restructuring is necessary. JSON's object is record, struct, object, dictionary, hash, or associate array. JSON's array is array, vector, sequence, or list.⁵⁶

2.6.3 Modelo de dados JSON

```
object
  {
    { members }
members
  pair
  pair , members
pair
  string : value
array
  [
    [ elements ]
elements
  value
  value , elements
value
  string
  number
  object
  array
  true
  false
  null
```

Figura 3: Sintaxe de alto nível do formato JSON. Os valores básicos podem ser strings, números, objects, arrays ou as constantes “true”, “false” e “null”. Objects são dicionários que associam strings a valores, e arrays são listas de valores. Exceto pelos detalhes sintáticos de strings e números, esta é a definição completa do formato. (Figura reproduzida do site <http://www.json.org>)

Uma boa explicação, em português, do modelo de dados JSON encontra-se no site JSON.org:

JSON está constituído em duas estruturas:

- Uma coleção de pares nome/valor. Em várias linguagens, isto é caracterizado como um *object*, *record*, *struct*, dicionário, *hash table*, *keyed list*, ou *arrays* associativos.
- Uma lista ordenada de valores. Na maioria das linguagens, isto é caracterizado como uma *array*, vetor, lista ou sequência.⁵⁷

56 CROCKFORD, 2006. “Esta é a diferença mais significativa. Apesar de que existem transformações que permitem a XML exprimir [tais estruturas], JSON as exprime diretamente. Os valores simples de JSON são os mesmos usados em linguagens de programação. As estruturas de JSON se parecem com estruturas de linguagens de programação. Nenhuma reestruturação é necessária. O object to JSON é registro, struct, object, dicionário, hash ou array associativo. O array do JSON é array, vetor, sequência ou lista.”. Nossa tradução.

57 **Introdução ao JSON**. Disponível em: <<http://www.json.org/json-pt.html>>. Acesso em 15 nov. 2010.

O primeiro elemento sintático de um documento JSON pode ser o caractere `{` para denotar o início de um dicionário, ou o caractere `[` para indicar o início de uma lista.⁵⁸

Estas duas estruturas podem ser aninhadas arbitrariamente, e, assim, árvores ou hierarquias podem ser representadas. Além de poderem ser dicionários ou listas, os valores internos podem ser tipos simples como números, strings ou as constantes `true`, `false` e `null`.

Os números seguem a sintaxe de JavaScript, podendo representar inteiros ou ponto flutuante.

As strings são cadeias de caracteres Unicode representando textos de tamanho arbitrário em qualquer idioma, codificadas em UTF-8 ou ASCII puro. Em ASCII, uma sequência como `\u6c23` pode ser usada para representar o caractere chinês 氣 – o ideograma do qi.

O modelo de dados JSON resume-se a isto.

⁵⁸ CROCKFORD 2006b. “A JSON text is a serialized object or array.”. Tradução: “Um texto JSON é um objeto ou um array serializado”

2.7 Sistemas de banco de dados não-relacionais

Boa parte do conteúdo da Web é organizado de forma hierárquica, encontra-se em documentos semiestruturados HTML ou XML, e é multimídia por natureza. Esse conteúdo é, portanto, formado por componentes distintos agregados. Modelar este tipo de informação em um banco de dados relacional normalizado não é trivial, e pode resultar em problemas de desempenho, em virtude do custo computacional das inevitáveis junções entre dados de tabelas distintas. Desnormalizar e escalar horizontalmente ajudam a enfrentar a dinâmica do tráfego na Web, mas não são fáceis de implementar em bancos de dados relacionais, que foram projetados para manter a consistência em tempo real a qualquer custo (EURE, 2009).

Estas questões motivaram grandes sites como Google, Amazon.com e Facebook a desenvolver e implantar, em larga escala, sistemas de bancos de dados não-relacionais, iniciando a tendência que ganhou o apelido de NoSQL (Not only SQL: não apenas SQL). O Apache Cassandra foi criado pelo Facebook e é também utilizado por outros sites de alto tráfego, como Twitter e Digg, e grandes empresas, como Cisco e Rackspace (APACHE, 2010). Embora continuem sendo proprietários, os sistemas não-relacionais do Google – Bigtable (CHANG, 2006) – e da Amazon – Dynamo (DECANDIA, 2007) – agora estão expostos ao público na forma de serviços de armazenagem em nuvem. O Bigtable é base do Google Datastore, o SGBD oferecido como parte do serviço de hospedagem AppEngine. Já a Amazon.com oferece serviços baseados no Dynamo, sob a marca Amazon Web Services⁵⁹.

Porém, o rótulo NoSQL é amplo demais (como tendem a ser as definições pela negativa). Uma grande variedade de mecanismos de persistência de dados com objetivos, arquiteturas e recursos muito variados são não-relacionais, e não apenas sistemas novos. CDS/ISIS, MUMPS⁶⁰, Adabas⁶¹ e Berkeley DB⁶² são sistemas NoSQL com mais de 20 anos de idade. Cassandra, Redis, CouchDB, Hypertable, Riak, ThruDB, Hadoop Hbase, MongoDB⁶³ são produtos novos, vários deles com menos de 5 anos de maturação. Cada um deles representa uma combinação peculiar de características, otimizada para fins bem específicos.

59 Serviços descritos em <<http://aws.amazon.com/>>. Acesso em 22 nov. 2010.

60 Não existe uma página oficial sobre MUMPS, mas há diversas variantes e fornecedores referenciados na página sobre MUMPS na Wikipedia: <<http://en.wikipedia.org/wiki/MUMPS>>. Acesso em 22 nov. 2010.

61 Produto descrito em <http://www.softwareag.com/corporate/products/adabas_2010/default.asp>. Acesso em 22 nov. 2010.

62 Produto descrito em <<http://www.oracle.com/technetwork/database/berkeleydb/overview/index.html>>. Acesso em 22 nov. 2010.

63 Um excelente ponto de partida para conhecer sistemas NoSQL é o site <<http://nosql-database.org/>>. Acesso em 22 nov. 2010.

Um critério de seleção entre as várias possibilidades NoSQL é o modelo de dados. Muitos dos produtos que acabamos de mencionar são “key-value stores” (armazéns de chave-valor). É o caso do BerkeleyDB: essencialmente, é um produto otimizado para recuperar rapidamente um BLOB (Binary Large Object), ou mais precisamente “um monte de bytes”, porque nativamente estes produtos não são capazes de criar índices a partir de uma parte do conteúdo do registro. Na verdade, é até estranho falar de registro quando não se têm campos. Em um key-value store o registro é simplesmente um valor opaco, e não há campos distinguíveis no nível do sistema de banco de dados. Fica a cargo da aplicação que está lendo um destes registros decifrar sua estrutura interna.

Citamos este tipo de sistema como um exemplo extremo de algo que não seria a melhor opção para armazenar registros ISIS. Afinal, um registro ISIS não só tem campos, como tem até subcampos, e aplicações ISIS dependem de poder acessar estes subcampos de forma eficiente por meio de índices.

2.7.1 Sistemas de bancos de dados orientados a documento

Entre os diversos sistemas NoSQL recentes, uma subcategoria nos parece a mais apropriada para lidar com registros de dados semiestruturados, incluindo registros ISIS: são os sistemas de bancos de dados orientados a documento.

Few databases identify themselves as document databases. As of this writing, the only well-known document database apart from MongoDB is Apache's CouchDB. CouchDB's document model is similar, although data is stored in plain text as JSON, whereas MongoDB uses the BSON binary format. Like MongoDB, CouchDB supports secondary indexes; the difference is that the indexes are defined by writing map reduce functions, which is a bit more involved than the declarative syntax using by MySQL and MongoDB.⁶⁴

É importante observar que tanto o MongoDB quanto o CouchDB são software livre, multiplataforma (GNU/Linux, MS Windows, Mac OS X) e ambos tiveram suas versões 1.0 lançadas recentemente: MongoDB em 27 ago. 2009, CouchDB em 14 jul. 2010.

⁶⁴ BANKER, 2010, p. 23. Livro em pre-print. Capítulo 1 disponível gratuitamente em <<http://www.manning.com/banker/>>. Acesso em 19 nov. 2010. Nossa tradução: “Poucos sistemas de bancos de dados se identificam como sistema de bancos de dados orientados a documento. Quando estas palavras foram escritas, o único sistema orientado a documento além do MongoDB era o CouchDB da Apache. O modelo de documentos do CouchDB é semelhante, embora os dados sejam armazenados como texto puro como JSON, enquanto o MongoDB utiliza o formato binário BSON. Como o MongoDB, o CouchDB suporta índices secundários; a diferença é que os índices definem-se escrevendo funções de mapeamento e redução (map reduce), que é um pouco mais complicado do que a sintaxe declarativa usada no MySQL e no MongoDB.”

O formato BSON utilizado pelo MongoDB é um formato binário (não legível por seres humanos, porém eficiente para ser processado pelo computador). Conceitualmente, o BSON é muito parecido com o JSON, sendo a principal diferença a grande variedade de tipos primitivos⁶⁵, entre eles um tipo ObjectID, que o torna até mais parecido com o modelo semiestruturado das *ssd-expressions* de Abiteboul et. al. (1999).

Tanto CouchDB quanto MongoDB são SGBD (Sistemas de Gerenciamento de Banco de Dados⁶⁶), pois foram projetados para permitir que várias aplicações, cada uma delas com um ou mais usuários, acessem a base de dados simultaneamente, de forma segura e controlada, para fazer consultas, inclusões, edições e exclusões sem correr o risco de corromper os dados, e sem expor os dados a pessoas não autorizadas, graças ao controle de acesso por senhas e permissões. O CISIS, em comparação, não oferece este tipo de funcionalidade, pois ele é apenas um motor de banco de dados⁶⁷. Em sistemas baseados no CISIS, por exemplo, é a aplicação que controla o acesso dos usuários, gerencia senhas e verifica permissões.

Uma excelente comparação entre o CouchDB e o MongoDB se encontra no próprio site do CouchDB. A análise foi escrita por Dwight Merriman, fundador e CEO da 10gen⁶⁸, a empresa que criou o MongoDB. Por brevidade, vamos citar apenas os parágrafos finais:

Casos de Uso

Pode ser útil examinar alguns problemas particulares e considerar como os resolveríamos.

- Se estivéssemos construindo o Lotus Notes, usaríamos Couch porque seu modelo MVCC de versionamento e reconciliação se encaixa perfeitamente. Qualquer cenário em que os dados fiquem offline por horas e depois voltem a estar online combina com ele. Em geral, se tivermos várias instâncias de bancos de dados replicadas, eventualmente conciliadas, geograficamente distribuídas, frequentemente fora do ar, usaremos Couch.
- Se tivermos requisitos de desempenho muito altos, usaremos Mongo. Por exemplo, armazenagem de perfis de usuário de Web sites e cacheamento de dados de outras fontes.
- Para um cenário com taxas de atualização muito altas, usaríamos Mongo porque ele é bom nisso. Por exemplo, atualizar em tempo real contadores analíticos de sites (exibições de páginas, visitas, etc.).⁶⁹

65 Especificação do formato BSON, disponível em <<http://bsonspec.org/#/specification>>. Acesso em 19 nov. 2010.

66 Ver definição de **sistema gerenciador de banco de dados** no Glossário, p. 83.

67 Ver definição de **motor de banco de dados** no Glossário, p. 83.

68 Página com biografias da equipe da 10gen <<http://www.10gen.com/team>>. Acesso em 22 nov. 2010.

69 MERRIMAN, 2010. Nossa tradução.

2.8 Metodologia LILACS

A metodologia LILACS – Literatura Latino-Americana e do Caribe em Ciências da Saúde – “surgiu diante da necessidade de uma metodologia comum para o tratamento descentralizado da literatura científica-técnica em saúde produzida na América Latina e Caribe” (BIREME, 2010b). Concretamente, a metodologia vem sendo aplicada há 25 anos na construção coletiva da base LILACS, com registros bibliográficos fornecidos por centros cooperantes localizados em 19 países das Américas.

A pesquisa na base LILACS pode ser feita diretamente no *Portal LILACS*⁷⁰, mas também na rede BVS – Biblioteca Virtual em Saúde – e em outras bases internacionais como MEDLINE, Web of Science e OCLC WorldCat. Em 14 nov. 2010, a base LILACS contém 532.695 registros bibliográficos, sendo 426.703 artigos de 813 periódicos diferentes, além de 74.765 monografias e 25.101 teses. Do total de registros, atualmente 29.9% incluem a referência para o texto completo disponível em acesso aberto.

A metodologia é usada não só na operação da base LILACS, mas também BBO – Bibliografia Brasileira de Odontologia, BDENF – Base de Dados de Enfermagem, MEDACARIB – Literatura do Caribe em Ciências da Saúde e bases de dados nacionais de países da América Latina.

O lançamento de LILACS em 1985 marcou também a primeira aplicação da tecnologia ISIS na BIREME/OPAS/OMS. Enquanto o *Portal LILACS* é voltado para o usuário final da base bibliográfica, os bibliotecários e técnicos dos centros cooperantes da rede LILACS encontram documentação e aplicativos na página *Modelo da Biblioteca Virtual em Saúde – Metodologias e aplicativos – LILACS*⁷¹.

Para este estudo sobre o modelo de dados ISIS e sua aplicação na LILACS, os dois manuais mais importantes foram:

BIREME (2008a). **Diccionario de datos del modelo LILACS Versión 1.6a**. São Paulo, SP, 2008.

BIREME (2008b). **Metodologia LILACS: Manual de Descrição Bibliográfica, 7ª ed.** São Paulo, SP, 2008.

70 BIREME, 2010c. Disponível em <<http://lilacs.bvsalud.org/>>. Acesso em 22 nov. 2010.

71 BIREME, 2010b. Disponível em <<http://bvsmodelo.bvsalud.org/php/level.php?lang=pt&component=27&item=6>> ou <<http://bit.ly/dIW8Pt>>. Acesso em 22 nov. 2010.

3 Metodologia

Tendo em mente que o objetivo geral deste trabalho foi estudar a viabilidade de migração dos registros de uma base de dados ISIS para um sistema de banco de dados orientado a documentos, o primeiro passo foi escolher qual produto utilizar. Embora o processo como um todo pudesse ser feito com qualquer um dos dois produtos que identificamos (CouchDB ou MongoDB), a definição do produto em particular afetou todas as etapas subsequentes.

3.1 Seleção do sistema de banco de dados

Como mencionado, encontramos no mercado dois SGBD semiestruturados orientados a documento, livres e disponíveis em versões estáveis: O MongoDB e o CouchDB.

No caso das bases de dados mais importantes construídas pela BIREME/OPAS/OMS, a análise de Merriman⁷² sugere que:

- para bases de atualização distribuída, como a LILACS, o CouchDB é a opção que melhor atende;
- para bases publicadas de forma centralizada e com alto tráfego, como Scielo, o MongoDB pode oferecer vantagens no ambiente de produção;

Entre os dois sistemas, considerando o cenário de uso da catalogação cooperativa característico da rede LILACS, optamos pelo CouchDB por ter um melhor suporte a replicação master-master (onde várias instâncias do sistema de banco de dados recebem inserções e atualizações simultaneamente, inclusive offline, para posterior sincronização).

Uma possível arquitetura seria a combinação de ambos, aproveitando a forte semelhança entre seus modelos de dados: usar o CouchDB para atividades processamento técnico de registro, como catalogação e indexação, e o MongoDB nos ambientes de acesso público que exigem maior desempenho.

72 MERRIMAN, 2010. Disponível em <<http://www.mongodb.org/display/DOCS/Comparing+Mongo+DB+and+Couch+DB>>. Acesso em 19 nov. 2010.

3.2 Escolha da representação de registros ISIS em JSON

Existem várias formas de representar registros ISIS genéricos em formato JSON.

A forma mais simples e direta seria construir uma lista associativa⁷³, ou seja, uma simples enumeração de pares formados por tags e conteúdos de campos. Por exemplo, este registro LILACS (parcial, alguns campos abreviados ou omitidos):

```
1 «BR1.1»
2 «538886»
4 «LILACS»
4 «LLXPEDT»
5 «S»
6 «as»
8 «Internet^ihttp://www.demneuropsy.com.br/imageBank/PDF/v3n3a04.pdf?aid2=168&...»
10 «Kanda, Paulo Afonso de Medeiros^1University of São Paulo ^2School of Medicine^3Cognitive Disorders of Clinicas Hospital Reference Center^pBrasil ^cSão Paulo^rorg»
10 «Anghinah, Renato^1University of São Paulo^2School of Medicine^3Cognitive Disorders of Clinicas Hospital Reference Center^pBrasil^cSão Paulo^rorg»
12 «The Clinical use of quantitative EEG in cognitive disorders»
12 «A utilização clínica do EEG quantitativo nos transtornos cognitivos»
30 «Dement. neuropsychol»
31 «3»
32 «3»
35 «1980-5764»
```

Na conversão para JSON como lista associativa, o registro acima fica assim:

```
[["1", "BR1.1"],
["2", "538886"],
["4", "LILACS"],
["4", "LLXPEDT"],
["5", "S"],
["6", "as"],
["8", "Internet^ihttp://www.demneuropsy.com.br/imageBank/PDF/v3n3a04.pdf?aid2=168&..."],
["10", "Kanda, Paulo Afonso de Medeiros^1University of São Paulo ^2School of Medicine^3Cognitive Disorders of Clinicas Hospital Reference Center^pBrasil ^cSão Paulo^rorg"],
["10", "Anghinah, Renato^1University of São Paulo^2School of Medicine^3Cognitive Disorders of Clinicas Hospital Reference Center^pBrasil^cSão Paulo^rorg"],
["12", "The Clinical use of quantitative EEG in cognitive disorders"],
["12", "A utilização clínica do EEG quantitativo nos transtornos cognitivos"],
["30", "Dement. neuropsychol"],
["31", "3"],
["32", "3"],
["35", "1980-5764"]
]
```

Repare a estrutura da lista associativa: o registro completo é uma lista, e os itens da lista são também listas. Porém as listas internas têm sempre exatamente dois itens: o tag e o valor do campo. Embora esta forma preserve toda a estrutura e informações do registro ISIS original, esta representação JSON, em particular, tem um custo computacional relativamente alto, por dois motivos:

73 Ver definição de **lista associativa** no Glossário, p. 83.

1. para localizar um campo pelo tag, o sistema de banco de dados precisa percorrer sequencialmente itens do registro⁷⁴;
2. para encontrar um subcampo, é preciso interpretar o conteúdo do campo, percorrendo-o caractere a caractere;

Considere a vida útil de um registro bibliográfico típico: ele será inserido na base de dados apenas uma vez, mas será consultado muitas vezes. Então, vale a pena explorar representações que tornem mais eficiente o acesso aos campos e subcampos. Um exemplo de tal representação JSON, ainda para o mesmo registro, seria esta:

```
{
  "1": [{"_": "BR1.1"}],
  "2": [{"_": "538886"}],
  "5": [{"_": "S"}],
  "4": [{"_": "LILACS"},
        {"_": "LLXPEDT"}],
  "6": [{"_": "as"}],
  "8": [{"i": "http://www.demneuropsych.com.br/imageBank/PDF/v3n3a04.pdf?id2=168&...",
        "l": "en.pdf",
        "_": "Internet"}],
  "10": [{"c": "São Paulo",
          "1": "University of São Paulo ",
          "p": "Brasil ",
          "3": "Cognitive Disorders of Clinicas Hospital Reference Center",
          "2": "School of Medicine",
          "r": "org",
          "_": "Kanda, Paulo Afonso de Medeiros"},
        {"c": "São Paulo",
          "1": "University of São Paulo",
          "p": "Brasil",
          "3": "Cognitive Disorders of Clinicas Hospital Reference Center",
          "2": "School of Medicine",
          "r": "org",
          "_": "Anghinah, Renato"}],
  "12": [{"_": "The Clinical use of quantitative EEG in cognitive disorders"},
        {"_": "A utilização clínica do EEG quantitativo nos transtornos cognitivos"}],
  "32": [{"_": "3"}],
  "31": [{"_": "3"}],
  "30": [{"_": "Dement. neuropsychol"}],
  "35": [{"_": "1980-5764"}]
}
```

Neste formato, o registro inteiro é representado como um dicionário⁷⁵, uma forma de associação entre chaves e valores mais eficiente que a lista associativa. Nota-se, por exemplo, que o identificador de cada tag aparece apenas uma vez nesta representação. O acesso ao campo, a partir do tag, também pode ser mais rápido, especialmente se o registro tiver muitos campos. Isso se dá graças ao uso de uma tabela de dispersão⁷⁶ na representação interna de

74 Esta é uma consequência do uso de lista associativa.

75 Ver definição de **dicionário** no Glossário, p. 83.

76 TABELA DE DISPERSÃO. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2010. Disponível em: <http://pt.wikipedia.org/w/index.php?title=Tabela_de_dispers%C3%A3o&oldid=22567329>. Acesso em 20 nov. 2010.

dicionários.

Porém, a diferença mais visível é que os próprios valores dos campos se tornam listas de dicionários nesta representação. Cada ocorrência de um campo é um dicionário, onde as chaves são os códigos dos subcampos, e os valores são os conteúdos destes subcampos. Por exemplo, a primeira ocorrência do campo 10 (Autor institucional, nível analítico) neste registro contém:

código do subcampo	descrição	valor
–	nome	Kanda, Paulo Afonso de Medeiros
1	afiliação (nível 1)	University of São Paulo
2	afiliação (nível 2)	School of Medicine
3	afiliação (nível 3)	Cognitive Disorders of Clinicas Hospital Reference Center
p	afiliação (país)	Brasil
c	afiliação (cidade)	São Paulo
r	grau de responsabilidade	org

Tabela 4: Definição e conteúdos dos subcampos do campo 10 do exemplo de registro exibido na página anterior.

O uso da tabela de dispersão tem uma consequência prática importante: a ordem dos itens em um dicionário JavaScript é indefinida, ou seja, não é garantido que as chaves aparecem em ordem ascendente ou em qualquer ordem previsível, e diferentes implementações ou mesmo versões da linguagem podem exibir as chaves de em diferentes ordens. O RFC-4627, que define o formato JSON é bem claro neste aspecto. Os dicionários (ou objects) que formam documentos JSON são desordenados:

An object is an unordered collection of zero or more name/value pairs, where a name is a string and a value is a string, number, boolean, null, object, or array.⁷⁷

Em discussões internas na BIREME/OPAS/OMS houve consenso de que a ordem dos tags não é importante, desde que sempre seja garantida a ordem relativa das ocorrências de um mesmo tag porque, por exemplo, nos campos de autor, a ordem dos autores creditados é relevante. A representação acima garante isso (veja os conteúdos do tag 10). A relevância da ordem dos subcampos já não é tão consensual.

⁷⁷ CROCKFORD 2006b. Disponível em <<http://tools.ietf.org/html/rfc4627>>. Acesso em 15 nov. 2010. “Um objeto é uma coleção desordenada de zero ou mais pares nome/valor, onde o nome é uma string e o valor é uma string, número, booleano, null, object ou array.” Nossa tradução.

3.2.1 ISIS-JSON: padrões para a representação de registros ISIS em JSON

Ao constatar que várias formas diferentes poderiam ser empregadas para representar registros ISIS em JSON, resolvemos catalogar alguns padrões de modo sistemático, conforme a tabela a seguir.

ISIS-JSON	Representação do registro completo	Representação de cada ocorrência
tipo 0	lista associativa com tags repetidos [tag, ocorrência] [["10", «conteúdo», ...], ...]	string única " ^aAlfa^bBeta"
tipo 1	dicionário com listas de ocorrências { "10" : [«conteúdo», ...], ... }	string única " ^aAlfa^bBeta"
tipo 2	dicionário com listas de ocorrências { tag : [«conteúdo», ...], ... }	lista associativa [("a", "Alfa"), ("b", "Beta")...]
tipo 3	dicionário com listas de ocorrências { tag : [«conteúdo», ...], ... }	dicionário de strings { "a" : "Alfa", "b" : "Beta"... }
tipo 4	dicionário com listas de ocorrências { tag : [«conteúdo», ...], ... }	dicionário de listas de strings { "a" : ["Alfa", ...], "b" : ["Beta", ...], ... }
tipo 5	lista associativa sem tags repetidos [tag, lista de ocorrências] [["10", [«conteúdo», ...]], ...]	lista associativa [("a", "Alfa"), ("b", "Beta")...]
tipo 6	lista associativa sem tags repetidos [tag, lista de ocorrências] [["10", [«conteúdo», ...]], ...]	string única " ^aAlfa^bBeta"

Tabela 5: Algumas das possibilidades mais simples de representação de registros ISIS no formato JSON. Todas elas podem ser utilizadas em um sistema de banco de dados semiestruturado como o CouchDB, porém as representações que não usam um dicionário no primeiro nível (tipo 0, 5 e 6) precisam ser embutidas em outra estrutura porque o registro no CouchDB é sempre um dicionário no primeiro nível.

Nas próximas seções apresentaremos em detalhes cada um destes tipos de representação ISIS-JSON.

3.2.2 Formatos ISIS-JSON baseados em lista associativa

O ISIS-JSON tipo 0 é a forma mais direta de conversão de um registro ISIS para o formato JSON. Cada registro ISIS é representado por uma lista associativa de ocorrências de campos, e o valor de cada ocorrência de um campo é representado por uma string. Veja este exemplo abreviado:

```
[
  ["2", "538886"],
  ["10", "Kanda, Paulo Afonso^1USP^2FMUSP^3CRDC^pBrasil^cSão Paulo^rorg"],
  ["10", "Smidth, Magali Taino^1USP^2FMUSP^3CRDC^pBrasil^cSão Paulo^rorg"]
]
```

Note que o formato ISIS-JSON tipo 0:

- preserva a ordem absoluta dos tags no registro;
- aceita subcampos repetitivos
- preserva a ordem dos subcampos
- não pode ser inserido diretamente no CouchDB

O último ponto foi mencionado anteriormente, mas merece um esclarecimento mais detalhado: cada registro no CouchDB precisa ser um dicionário. Isto porque o CouchDB utiliza dois metadados para manipular os registros: um campo `_id`, que é o identificador único do registro no objeto banco de dados, e um campo `_rev` que é a revisão do registro, necessária para o funcionamento do mecanismo de controle de atualizações concorrentes MVCC⁷⁸ do CouchDB.

Isso significa que um registro ISIS-JSON tipo 0 precisa ser embutido em um dicionário para ser inserido no CouchDB, por exemplo desta forma:

```
{
  "_id": "ed0411ccb319212cf40524856200091f",
  "_rev": "1-ced0bc24378341196cad97f9f8da560c",
  "fields": [
    [ "2", "538886" ],
    [ "10", "Kanda, Paulo Afonso^1USP^2FMUSP^3CRDC^pBrasil^cSão Paulo^rorg" ],
    [ "10", "Smidth, Magali Taino^1USP^2FMUSP^3CRDC^pBrasil^cSão Paulo^rorg" ]
  ]
}
```

⁷⁸ MVCC é Multi-Version Concurrency Control (controle de concorrência por múltiplas versões). Uma explicação de seu uso no CouchDB pode ser encontrada em ANDERSON, 2010, disponível em <<http://guide.couchdb.org/draft/consistency.html>>. Acesso em 15 nov. 2010.

O formato ISIS-JSON tipo 6 pode ser exemplificado assim:

```
[
  ["2",
    ["538886"],
  ],
  ["10",
    ["Kanda, Paulo Afonso^1USP^2FMUSP^3CRDC^pBrasil^cSão Paulo^rorg",
     "Smidth, Magali Taino^1USP^2FMUSP^3CRDC^pBrasil^cSão Paulo^rorg"]
  ]
]
```

Este formato é também uma lista associativa, mas neste caso cada tag aparece apenas uma vez, e não várias vezes como no ISIS-JSON tipo 0. Para acomodar a repetição de ocorrências então cada valor associado a um tag é uma lista de strings. No exemplo acima, o tag 2 está associado a uma lista com apenas um valor, "538886", e o tag 10 está associado a uma lista com dois valores, "Kanda, Paulo Afonso..." e "Smidth, Magali Taino...".

Outro tipo de registro baseado em lista associativa do nosso catálogo é o tipo 5:

```
[
  ["2",
    [
      [
        ["_", "538886"]
      ]
    ]
  ],
  ["10",
    [
      [
        ["_", "Kanda, Paulo Afonso"],
        ["1", "USP"],
        ["2", "FMUSP"],
        ["3", "CRDC"],
        ["p", "Brasil"],
        ["c", "São Paulo"],
        ["r", "org"]
      ],
      [
        ["_", "Smidth, Magali Taino"],
        ["1", "USP"],
        ["2", "FMUSP"],
        ["3", "CRDC"],
        ["p", "Brasil"],
        ["c", "São Paulo"],
        ["r", "org"]
      ]
    ]
  ]
]
```

Neste caso, cada ocorrência de cada campo é expandida na forma de uma lista associativa onde as chaves são os códigos dos subcampos e os valores são seus conteúdos respectivos. Como mencionado antes, adotamos a convenção de associar o código “_” ao subcampo

principal.

Assim como os registros do tipo 0, os registros do tipo 5 e 6 precisam ser embutidos em um dicionário para serem carregados no CouchDB.

Estas três representações têm em comum as seguintes características:

- Necessitam ser embutidas em um dicionário para serem carregadas no CouchDB. Esta é uma limitação comum a qualquer representação de registro que não use um dicionário no primeiro nível.
- Preservam a ordem absoluta dos tags e dos subcampos. Esta é uma vantagem que não pode ser emulada quando se usam dicionários em lugar de listas associativas, ao menos no contexto do JSON onde dicionários são por definição desordenados.

3.2.3 Formatos ISIS-JSON baseados em dicionário

O formato mais simples baseado em dicionário é o ISIS-JSON tipo 1:

```
{ "10":
  [ "Kanda, Paulo Afonso^1USP^2FMUSP^3CRDC^pBrasil^cSão Paulo^rorg",
    "Smidth, Magali Taino^1USP^2FMUSP^3CRDC^pBrasil^cSão Paulo^rorg" ],
  "2":
    [ "538886" ]
}
```

Em qualquer formato baseado em dicionário no primeiro nível, cada tag só aparecerá uma vez, porque esta é uma característica essencial do dicionário: as chaves têm que ser únicas. Outra característica já mencionada é que a ordem das chaves é indefinida porque os dicionários em JSON são desordenados. Para acomodar a repetição dos campos, os valores do dicionário são listas. No caso do tipo 1, são listas de strings.

O ISIS-JSON tipo 2 expande os subcampos em lista associativas, como vimos anteriormente no ISIS-JSON tipo 5.

```
{ "10":
  [
    [
      [ "_", "Kanda, Paulo Afonso" ],
      [ "1", "USP" ],
      [ "2", "FMUSP" ],
      [ "3", "CRDC" ],
      [ "p", "Brasil" ],
      [ "c", "São Paulo" ],
      [ "r", "org" ]
    ],
    [
      [ "_", "Smidth, Magali Taino" ],
      [ "1", "USP" ],
      [ "2", "FMUSP" ],
      [ "3", "CRDC" ],
      [ "p", "Brasil" ],
      [ "c", "São Paulo" ],
      [ "r", "org" ]
    ]
  ],
  "2":
  [
    [
      [ "_", "538886" ]
    ]
  ]
}
```

O formato ISIS-JSON tipo 3 foi o segundo exemplo na seção 3.2.1 (p. 47). Ele utiliza dicionários para representar os subcampos, com a consequência de que a ordem dos subcampos não é preservada, e também não é possível ter subcampos repetidos.

Exemplo em formato ISIS-JSON tipo 3:

```
{ "10":
  [
    {
      "_": "Kanda, Paulo Afonso",
      "1": "USP",
      "2": "FMUSP",
      "3": "CRDC",
      "c": "São Paulo",
      "p": "Brasil",
      "r": "org"
    },
    {
      "_": "Smidth, Magali Taino",
      "1": "USP",
      "2": "FMUSP",
      "3": "CRDC",
      "c": "São Paulo",
      "p": "Brasil",
      "r": "org"
    }
  ],
  "2":
  [
    {
      "_": "538886"
    }
  ]
}
```

Finalmente, o formato ISIS-JSON tipo 4 é uma expansão do tipo 3 para permitir subcampos repetidos. Para atingir este objetivo, o valor de cada subcampo (exceto o principal) deixa de ser uma simples string, e passa a ser uma lista de strings, podendo assim acomodar os valores de várias ocorrências de um determinado subcampo.

```
{ "10":
  [
    {
      "_": "Kanda, Paulo Afonso",
      "1": ["USP"],
      "2": ["FMUSP"],
      "3": ["CRDC"],
      "c": ["São Paulo"],
      "p": ["Brasil"],
      "r": ["org"]
    },
    {
      "_": "Smidth, Magali Taino",
      "1": ["USP"],
      "2": ["FMUSP"],
      "3": ["CRDC"],
      "c": ["São Paulo"],
      "p": ["Brasil"],
      "r": ["org"]
    }
  ],
  "2":
  [
    {
      "_": "538886"
    }
  ]
}
```

3.2.4 Resumo dos tipos de ISIS-JSON

A tabela a seguir resume as principais características dos padrões de representação ISIS-JSON que estudamos.

	tipo 0	tipo 1	tipo 2	tipo 3	tipo 4	tipo 5	tipo 6
aceita campos repetitivos	■	■	■	■	■	■	■
contêiner dos campos	alist	dict	dict	dict	dict	alist	alist
preserva ordem absoluta dos tags no registro	■					■	■
preserva ordem relativa das ocorrências de cada tag	■	■	■	■	■	■	■
aceita subcampos repetitivos	■	■	■		■	■	■
contêiner dos subcampos	string	string	alist	dict	dict	alist	string
tipo dos subcampos	string	string	string	string	list	string	string
preserva ordem dos subcampos	■	■	■			■	■
importação direta no CouchDB		■	■	■	■		

Tabela 6: Características funcionais dos diferentes tipos de representação ISIS-JSON. As estruturas são: alist = lista associativa (acesso sequencial); dict = dicionário (acesso direto por hash); string = texto (recuperação das marcas de subcampo através da análise – parsing – do conteúdo, por exemplo, por meio de expressões regulares); list = lista simples (acesso pelo índice da ocorrência do subcampo). A importação direta no CouchDB só é possível para os registros onde os campos são representados como um dicionário (tipos 1 a 4). Os registros tipo 0, 5 e 6 precisam ser embutidos em um item de dicionário porque no CouchDB o primeiro nível de um registro precisa ser um dicionário

3.3 Seleção e obtenção do conjunto de dados

Como funcionário da BIREME/OPAS/OMS, o autor solicitou, em março de 2010, ao então diretor da instituição, Dr. Abel Packer, autorização para uso de uma massa de dados da LILACS. Foi autorizada a cópia de 20% da base para este trabalho. Na ocasião havia cerca de 520.000 registros na base, então optamos por utilizar uma amostra de 100.000 registros neste trabalho.

O download da amostra foi feito uma vez, a título de teste, em março de 2010, e a segunda vez no dia 17 out. 2010, para melhor documentação do procedimento adotado. A massa de dados que usamos nos resultados finais foi a do dia 17 out. 2010. Para referência futura, adotamos o nome LILACS100K para nos referirmos a esta amostra.

3.3.1 Procedimento de *download* da amostra

Os registros da base LILACS estão disponíveis para acesso público tanto para visualização na interface Web do site da BVS Regional, usando um navegador qualquer, quanto para download em lotes, através do mesmo site. Para fazer o download, adotamos o procedimento descrito pelas figuras a seguir.



Passo 1: Página principal da BVS regional em <<http://regional.bvsalud.org>> em 17 out. 2010. Na ocasião, o link LILACS remetia para uma página de pesquisa específica para esta base de dados. Depois que estas telas foram capturadas este link passou a remeter para o novo portal LILACS, mas a busca original continua disponível como ilustrada nos passos seguintes, desde que se utilize o link Pesquisa via formulário iAH no novo portal LILACS em <<http://lilacs.bvsalud.org/>>.

bvs biblioteca virtual em saúde Pesquisa em bases de dados [español](#) | [english](#)
 Base de dados : LILACS Formulário livre
 Pesquisar por : [Formulário básico](#) [Formulário avançado](#)
 Entre uma ou mais palavras
 \$
 Todas as palavras (AND) Qualquer palavra (OR)
[CONFIG](#) [PESQUISAR](#)
Notas :

- Esta opção recupera palavras do título do artigo, palavras do resumo, nome de substâncias, nome de pessoas como assunto, e descritores de assunto.
- Idioma da pesquisa:

Passo 2: Página de busca da LILACS, usando o sistema iAH desenvolvido pela BIREME/OPAS/OMS para pesquisa Web sobre bases ISIS. Digitamos \$ no campo de busca, porque este é o sinal de “coringa”. Assim encontramos todos os registros existentes. O formulário retratado acima é o básico. Atualmente o formulário avançado é exibido por padrão, mas o mesmo procedimento pode ser feito digitando \$ no primeiro campo.

bvs biblioteca virtual em saúde Pesquisa em bases de dados
 Base de dados : LILACS
 Pesquisa : \$
 Referências encontradas : 529493 [\[refinar\]](#)
 Mostrando: 1.. 10 no formato [\[Detalhado\]](#)
 página 1 de 52950 ir para página 1 2 3 4 5 6 7 8 9 10
 1 / 529493 LILACS
 seleciona **Id:** 560470
 para imprimir **Autor:** Villamil J., Luis Carlos; Romero P., Jaime Ricardo; Cediel B., Natalia.
 Fotocópia **Título:** La salud animal y la globalización: el desafío de políticas sostenibles y equitativas en el contexto de los países en desarrollo / Animal health and globalization: the challenge of equitable and sustainable policies in the context of developing countries
 Documentos relacionados **Fonte:** Rev. med. vet. (Bogotá);(15):79-94, ene.-jun. 2008.

Passo 3: O sistema encontrou 529.493 registros na base LILACS. Os resultados aparecem em ordem cronológica decrescente (registros mais novos primeiro), como se pode observar pelo número identificador do registro acima: 560470. Os números identificadores são atribuídos aos registros em sequência ascendente, mas por questões operacionais alguns números de identificação não são usados ou são descartados. O botão >>ENVIAR RESULTADO (segundo botão azul da barra superior) remete para a próxima tela, onde se pode iniciar o download dos registros encontrados.

Passo 4: Nesta tela podemos selecionar quais registros baixar e o formato do arquivo para download. Baixamos primeiro um lote com o intervalo de 1 a 10.000, depois um lote com o intervalo de 1 a 100.000. Entre os formatos disponíveis, escolhemos ISO-2709 pois é este o formato que os centros cooperantes da LILACS utilizam para transmitir registros.

Fizemos o download duas vezes: primeiro um lote de 10.000 registros, depois um lote de 100.000 registros, a partir dos resultados da mesma busca, de tal modo que o segundo lote inclui o primeiro. Os arquivos gerados foram:

- lilacs10k.iso (34.632.617 bytes ou ~34 MB);
- lilacs100k.iso (323.402.872 bytes ou ~309 MB);

Abaixo, listagens do diretório documentando com precisão os arquivos da amostra:

```
luciano@meng:~/Dropbox/tcc/lilacs/data/lilacs2010-10-17$ ls -la
total 349668
drwxr-xr-x 3 luciano luciano      4096 2010-10-31 13:36 .
drwxr-xr-x 4 luciano luciano      4096 2010-11-02 09:55 ..
-rw-r--r-- 1 luciano luciano 323402872 2010-10-17 13:59 lilacs100k.iso
-rw-r--r-- 1 luciano luciano  34632617 2010-10-17 13:37 lilacs10k.iso
```

```
luciano@meng:~/Dropbox/tcc/lilacs/data/lilacs2010-10-17$ ls -lah
total 342M
drwxr-xr-x 3 luciano luciano 4.0K 2010-10-31 13:36 .
drwxr-xr-x 4 luciano luciano 4.0K 2010-11-02 09:55 ..
-rw-r--r-- 1 luciano luciano 309M 2010-10-17 13:59 lilacs100k.iso
-rw-r--r-- 1 luciano luciano  34M 2010-10-17 13:37 lilacs10k.iso
```

3.3.2 Ferramenta de conversão de ISO-2709 para JSON

A ferramenta `isis2json.py`⁷⁹ foi criada por este autor no decorrer desta pesquisa como um utilitário genérico para conversão entre bases do formato ISIS ou ISO-2709 para JSON.

Se o arquivo de entrada está no formato ISO-2709, o script `isis2json.py` pode ser executado simplesmente com o interpretador Python incluído por padrão em praticamente qualquer sistema GNU/Linux ou Mac OS X, e disponível também no sistema Windows.

O script suporta também arquivos de entrada no formato ISIS (um par de arquivos, extensões `.mst` e `.xrf`), mas neste caso `isis2json.py` precisa ser invocado pelo interpretador Python, pois necessita da biblioteca de programação ZeusIII desenvolvida por Heitor Barbieri na BIREME/OPAS/OMS em linguagem Java. Em 22 nov. 2010 tal biblioteca ainda não está disponível para download fora da rede interna da BIREME/OPAS/OMS.

O programa `isis2json.py` oferece várias opções para controlar seu funcionamento. Por exemplo, os comandos abaixo convertem parte da base CDS que vem como exemplo no WinISIS do formato ISO-2709 para o formato JSON utilizando alguns parâmetros opcionais:

```
$ ./isis2json.py cds.iso -c -f -q 1 > cds1.json
```

Os parâmetros usados neste exemplos são:

- **-c** gera um arquivo em formato JSON compatível com o formato de inclusão em lote do CouchDB
- **-f** expande os campos na forma de dicionários, com marcadores de subcampos como chaves (este formato corresponde ao ISIS-JSON tipo 4, como veremos na seção 3.2, p. 44)
- **-q 1** gera apenas um registro na saída (a opção **q** controla o número de registros a serem gerados, e é útil para produzir lotes menores a partir de uma entrada volumosa);

⁷⁹ Documentação e código-fonte podem ser encontrados no repositório do projeto ISIS-NBP, disponível em: <http://reddes.bvsalud.org/projects/isisnbp/wiki/Tools>. Acesso em 20 nov. 2010.

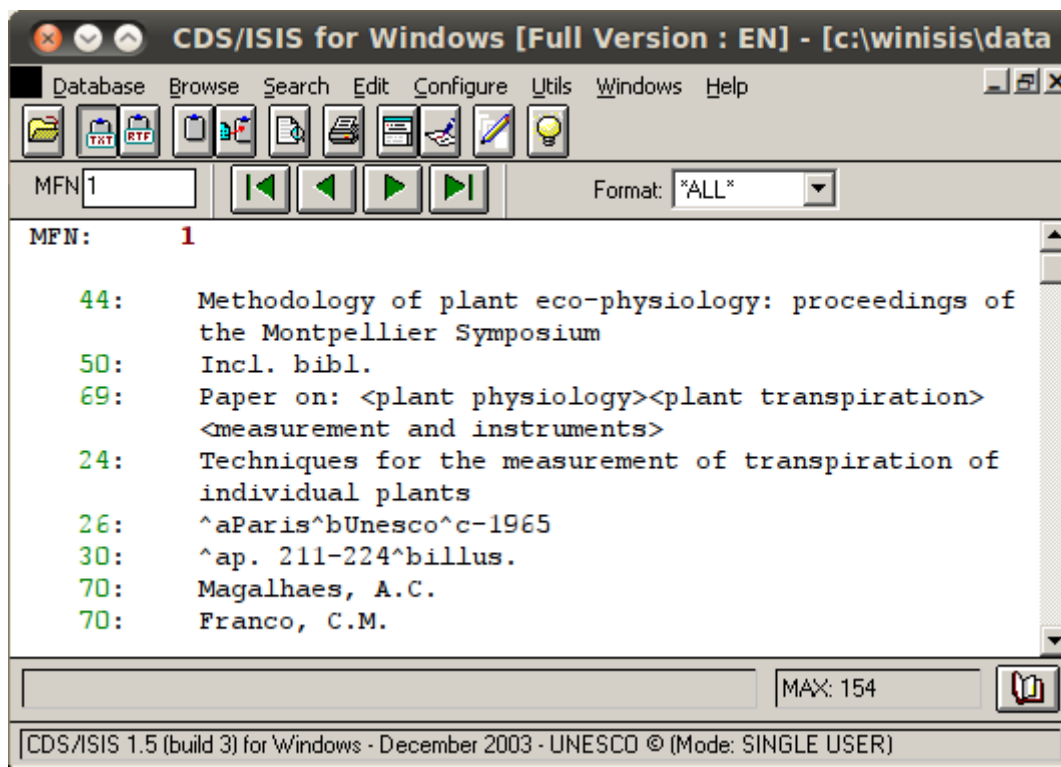


Figura 4: Registro #1 da base de dados CDS que acompanha o pacote WinISIS (tela do CDS/ISIS 1.5 for Windows rodando sobre Ubuntu Linux 10.04 com a ajuda do sistema WINE)

O registro exportado pelo exemplo anterior, cuja representação no WinISIS podemos ver na tela acima, apresenta-se como na listagem abaixo uma vez convertido para o formato JSON (endentaç o e quebras de linha foram acrescentadas para melhor visualizaç o):

```
{ "docs" :
  [
    {"070": [{"_": "Magalhaes, A.C."}, {"_": "Franco, C.M."}],
     "069": [{"_": "Paper on: <plant physiology><plant transpiration>\
              <measurement and instruments>"}],
     "024": [{"_": "Techniques for the measurement of transpiration of individual plants "}],
     "026": [{"a": "Paris", "c": "-1965", "b": "Unesco"}],
     "030": [{"a": "p. 211-224", "b": "illus."}],
     "044": [{"_": "Methodology of plant eco-physiology: proceedings of the Montpellier\
              Symposium "}],
     "050": [{"_": "Incl. bibl."}]
  ]
}
```

O dicion rio com a chave “docs”   uma esp cie de envelope que envolve os registros no formato de importa o em lote do CouchDB. O envelope cont m uma lista de registros (no exemplo, apenas um registro), e cada registro   um dicion rio associando tags  s ocorr ncias dos campos. Neste formato, que denominamos ISIS-JSON tipo 3, os campos s o representados como listas de ocorr ncias, e cada ocorr ncia   um dicion rio onde o subcampo

principal é indicado pela chave “_” (caractere de sublinha), e as chaves dos demais subcampos são os respectivos códigos.

3.3.3 Conversão da amostra de LILACS100K para JSON

Uma vez criada a ferramenta `isis2json.py`, a conversão da amostra LILACS100K para JSON foi feita em 10 partes, porque testes anteriores revelaram duas peculiaridades sobre o processo de importação em lote do CouchDB:

1. o processo de importação utiliza apenas um núcleo da CPU para cada requisição, então em equipamentos com dois núcleos é praticamente duas vezes mais rápido importar dois lotes de 10.000 registros em paralelo do 20.000 registros em um único lote;
2. para massas de dados maiores do que 50MB, às vezes o processo de importação era abortado sem sucesso (não investigamos a causa disso, porque a observação 1 já era motivo suficiente para subdividir a massa de dados);

Para gerar 10 lotes de 10.000 registros a partir do arquivo `lilacs100k.iso`, utilizamos um script shell, acionando `isis2json.py` com as seguintes opções:

- **-q 10000** para indicar a quantidade de registros a serem exportados
- **-s N** para indicar quantos registros saltar a partir do início do arquivo de entrada, onde N foi sendo sucessivamente alterado para 10000, 20000, 30000 etc.
- **-c** para gerar arquivos no formato do CouchDB
- **-t v** para prefixar os tags com a letra “v”, facilitando seu acesso via JavaScript
- **-i 2** para utilizar o conteúdo do tag 2 como identificador do registro.

O script completo tomou esta forma:

```
#!/bin/bash
./isis2json.py -c -t v -i 2 -q 10000 lilacs100k.iso > lilacs100k-0.json &
./isis2json.py -c -t v -i 2 -q 10000 -s 10000 lilacs100k.iso > lilacs100k-1.json &
./isis2json.py -c -t v -i 2 -q 10000 -s 20000 lilacs100k.iso > lilacs100k-2.json &
./isis2json.py -c -t v -i 2 -q 10000 -s 30000 lilacs100k.iso > lilacs100k-3.json &
./isis2json.py -c -t v -i 2 -q 10000 -s 40000 lilacs100k.iso > lilacs100k-4.json &
./isis2json.py -c -t v -i 2 -q 10000 -s 50000 lilacs100k.iso > lilacs100k-5.json &
./isis2json.py -c -t v -i 2 -q 10000 -s 60000 lilacs100k.iso > lilacs100k-6.json &
./isis2json.py -c -t v -i 2 -q 10000 -s 70000 lilacs100k.iso > lilacs100k-7.json &
./isis2json.py -c -t v -i 2 -q 10000 -s 80000 lilacs100k.iso > lilacs100k-8.json &
./isis2json.py -c -t v -i 2 -q 10000 -s 90000 lilacs100k.iso > lilacs100k-9.json
```

Note que no script acima não foi utilizada a opção **-f** que discutimos anteriormente. Portanto os subcampos não foram expandidos e os arquivos gerados contém registros ISIS-JSON tipo 1. Um outro lote de arquivos foi gerado utilizando procedimento semelhante porém com a opção **-f** para gerar registros ISIS-JSON tipo 4.

3.4 Carga dos dados no CouchDB

Para carregar os arquivos JSON gerados no procedimento anterior, criamos duas bases: **lilimp** para os registros ISIS-JSON tipo 1, e **lilacs** para registros tipo 4.

Para carregar cada um dos lotes em **lilimp**, utilizamos o comando curl do GNU/Linux, um cliente do protocolo HTTP em linha de comando. A chamada do curl utilizada foi a seguinte:

```
$ curl -d @lilacs100k-0.json -X POST http://127.0.0.1:5984/lilimp/_bulk_docs \
-H"Content-Type: application/json"
```

Os argumentos passados para o comando curl foram:

- **-d** para informar o nome do arquivo a ser carregado
- **-X** para definir o verbo HTTP a ser utilizado (no caso, POST)
- **-H** para especificar um cabeçalho “Content-Type”

O caminho da URL alvo da requisição, `/lilimp/_bulk_docs`, é documentado na API (Application Programming Interface) do CouchDB⁸⁰.

Procedimento semelhante foi utilizado para carregar a base **lilacs** com registros do tipo 4.

Key	Value
"538905" ID: 538905	{rev: "1-783e02601860caba002792e4baffe734"}
"538904" ID: 538904	{rev: "1-62871b374fa07f51a11305be8c08d73c"}
"538903" ID: 538903	{rev: "1-821784a3cfac7f429d18970ba8d97159"}
"538902" ID: 538902	{rev: "1-e3570e906f02d596659f36d2389e76f7"}
"538901" ID: 538901	{rev: "1-63279b2ae1874a5534b9679ed7da30a4"}
"538900" ID: 538900	{rev: "1-207bead98ad14794dc289acc871ab43a"}
"538899" ID: 538899	{rev: "1-89745bef779123b3a5f24493e7932040"}
"538898" ID: 538898	{rev: "1-debdb0c3433809a208423d7c8363932a"}
"538897" ID: 538897	{rev: "1-ee0929a4965a75ec496143b6c09e8d04"}
"538896" ID: 538896	{rev: "1-0649d1b6f121b04f5f6dbc6cb046f59e"}

Figura 5: Tela do CouchDB, acessada via navegador Firefox, mostrando os primeiros dos 100.000 registros da amostra LILACS100K no objeto banco de dados **lilacs**.

80 O uso de `_bulk_docs` é documentado em http://wiki.apache.org/couchdb/HTTP_Bulk_Document_API. Acesso em 21 nov. 2019.

3.5 Análise dos registros no CouchDB

Field	Value
<code>_id</code>	<code>"538900"</code>
<code>_rev</code>	<code>"1-edbb0d1e2e06a04494673bc5c9b19dfb"</code>
<code>v1</code>	<code>0 "BRL.1"</code>
<code>v10</code>	<code>0 "Guimarães, Henrique Cerqueira^1Federal University of Minas Gerais^2Faculty of Medicine^3Department of Internal Medicine. Behavioral and Cogni..."</code> <code>1 "Fialho, Patricia Paes Araujo^1Federal University of Minas Gerais^2Faculty of Medicine^3Department of Internal Medicine. Behavioral and Cogni..."</code> <code>2 "Carvalho, Viviane Amaral^1Federal University of Minas Gerais^2Faculty of Medicine^3Department of Internal Medicine. Behavioral and Cognitive..."</code> <code>3 "Santos, Etelvina Lucas dos^1Federal University of Minas Gerais^2Faculty of Medicine^3Department of Internal Medicine. Behavioral and Cogniti..."</code> <code>4 "Caramelli, Paulo^1Federal University of Minas Gerais^2Faculty of Medicine^3Department of Internal Medicine. Behavioral and Cognitive Neurolo..."</code>
<code>v113</code>	<code>0 "p"</code>
<code>v12</code>	<code>0 "Brazilian caregiver version of the Apathy Scale"</code> <code>1 "Versão brasileira direcionada ao cuidador da Escala de Apatia"</code>
<code>v2</code>	<code>0 "538900"</code>

Figura 6: Registro LILACS #538900 em formato ISIS-JSON tipo 1 na base *lilimp*, visualizado através da interface Web do CouchDB, denominada Futon.

O CouchDB possui uma interface Web, chamada Futon, que pode ser acessada com qualquer navegador moderno. O Futon é voltado para desenvolvedores e administradores de sistemas, não para usuários finais. Interfaces mais amigáveis precisam ser desenvolvidas sob medida para exibir os registros e permitir que eles sejam pesquisados, criados e editados por usuários não técnicos. Na tela acima, podemos ver duas colunas: na esquerda, as chaves que identificam os campos do registro, e à direita, seus valores.

Os dois primeiros campos são `_id` e `_rev`: ambos existem em qualquer registro no CouchDB. O valor de `_id` pode ser fornecido pelo usuário no momento da criação do registro, como fizemos ao usar o script `isis2json.py` com a opção `-i 2` (indicando que o valor do tag 2 deveria ser usado como identificador). Se o registro for submetido ao CouchDB sem um `_id`, o próprio SGBD, atribui um `_id` com um valor gerado por uma função `UUID`⁸¹, com a esta aparência: `"ed0411ccb319212cf40524856200091f"`.

81 UUID é Universally Unique Identifier, um mecanismo padrão para geração de identificadores universalmente únicos, definido no RFC-4122 <<http://tools.ietf.org/html/rfc4122>>. Acesso em 21 nov. 2010.

O campo `_rev` contém um identificador de revisão, formado por um número sequencial e um *hash*, uma espécie de assinatura digital do conteúdo do registro, gerada por um algoritmo de dispersão criptográfico. Por exemplo, a terceira revisão de um registro pode ter o seguinte valor de `_rev`: "3-ced0bc24378341196cad97f9f8da560c".

Os demais campos são exatamente os campos do registro ISIS-JSON. No exemplo da Figura 6, temos um registro no formato ISIS-JSON tipo 1, com os tags prefixados com a letra "v". Na Figura 7 temos os mesmos dados em outra base, em formato ISIS-JSON tipo 4.

Field	Value
<code>_id</code>	538900
<code>_rev</code>	1-207bead98ad14794dc289acc871ab43a
<code>1</code>	0
<code>10</code>	0 <ul style="list-style-type: none">r "Guimarães, Henrique Cerqueira"c "Belo Horizonte"1 "Federal University of Minas Gerais"2 "Faculty of Medicine"p "Brasil"3 "Department of Internal Medicine. Behavioral and Cognitive Neurology Unit"1234
<code>113</code>	0 <ul style="list-style-type: none">0 "Brazilian caregiver version of the Apathy Scale"
<code>12</code>	0 <ul style="list-style-type: none">0 "Brazilian caregiver version of the Apathy Scale"

Figura 7: O mesmo registro LILACS #538900 no Futon do CouchDB, agora em formato ISIS-JSON tipo 4 na base *lilacs*.

Note que neste caso os itens da coluna da direita são mais complexos: onde antes tínhamos apenas strings associadas a cada subcampo, agora somente os subcampos principais (chave "_") têm apenas uma string, os demais subcampos têm listas de strings. Observe que as listas são numeradas a partir de zero, então na Figura 7 sempre que aparece uma chave 0, trata-se do primeiro item de uma lista.

A aba **Source**, no canto superior esquerdo da janela do Futon, dá acesso à visualização e edição direta do registro em formato JSON.

3.5.1 Frequência de tipos de registro na amostra LILACS100K

Uma análise básica a se fazer em qualquer massa de dados LILACS é a contagem de registros por tipo.

O *Manual de Descrição Bibliográfica da Metodologia LILACS*⁸² estabelece que os campos 5 e 6 definem o tipo do registro.

O campo 5 (Tipo de Literatura) pode ter os valores:

- **S:** Série;
- **M:** Monografia;
- **T:** Tese, Dissertação;
- **N:** Não Convencional;
- **P:** Projeto (categoria complementar, deve ser combinada com S, M, T ou N)
- **C:** Conferência (categoria complementar, deve ser combinada com S, M, T ou N)

código	descrição
S	Documento publicado em uma série periódica
SC	Documento de conferência em uma série periódica
SCP	Documento de projeto e conferência em uma série periódica
SP	Documento de projeto em uma série periódica
M	Documento publicado em uma Monografia
MC	Documento de conferência em uma monografia
MCP	Documento de projeto e conferência em uma monografia
MP	Documento de projeto em uma monografia
MS	Documento publicado em uma série monográfica
MSC	Documento de conferência em uma série monográfica
MSP	Documento de projeto em uma série monográfica
T	Tese, Dissertação (publicado ou não)
TS	Tese, Dissertação pertencente a uma série monográfica
N	Documento não convencional
NC	Documento de conferência em forma não convencional
NP	Documento de projeto em forma não convencional

Tabela 7: Código ou combinação de códigos previstos para a categorização do Tipo de Literatura, tag 5 (tabela extraída do Manual de Descrição Bibliográfica, BIREME 2008b, p. 17)

82 BIREME 2008b, p. 15.

O campo 6 (Nível de Tratamento), pode ter os seguintes valores:

- **m**: nível monográfico;
- **mc**: nível monográfico de coleção;
- **ms**: nível monográfico de série;
- **am**: nível analítico monográfico;
- **amc**: nível analítico monográfico de coleção;
- **ams**: nível analítico monográfico de série;
- **as**: nível analítico de série;
- **c**: nível coleção;

A combinação entre os campos 5 e 6 define o tipo do registro, e conseqüentemente, as regras para preenchimento dos demais campos. Por exemplo, existem três tags usados para identificar o Autor Pessoal: 10, 16 e 23. O tag 10 é usado para identificar o Autor Pessoal em registros de nível analítico, ou seja, na descrição de um artigo de periódico ou capítulo de monografia. Já o tag 16 está associado ao Autor Pessoal em registros de nível monográfico, ou seja, registros que descrevem uma monografia como um todo. Finalmente, o tag 23 é para Autor Pessoal em nível de coleção, usado para identificar o organizador ou editor da coleção.

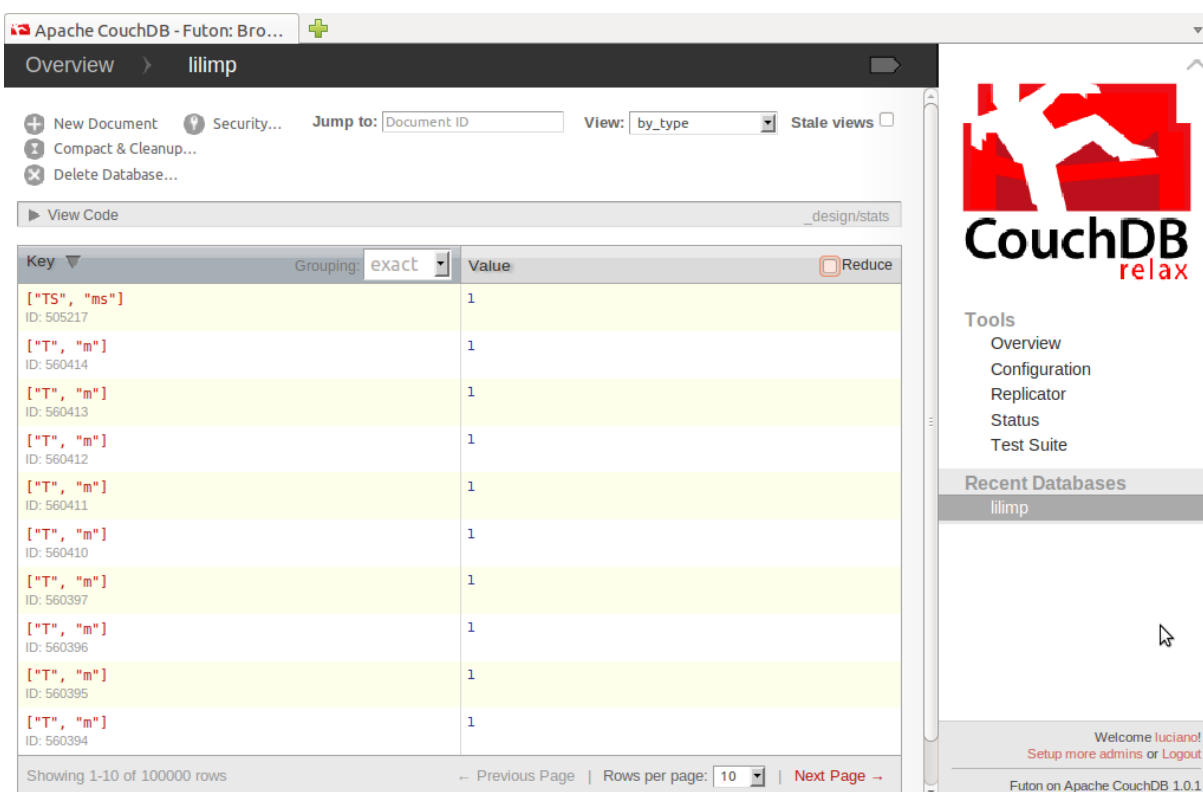
Em um registro com tag 5 = **M** e tag 6 = **amc** (abreviadamente, tipo **M:amc**) os campos 10, 16 e 23 são essenciais, ou seja, devem ser preenchidos caso a informação exista. Já num registro tipo **M:m**, somente o campo 16 deve ser preenchido.

Para fazer uma estatística de tipos de registros na amostra LILACS100K, utilizando o CouchDB, criamos uma *view*, que é um índice secundário, equivalente ao um arquivo invertido utilizado no ISIS para realizar buscas com alto desempenho. Enquanto no ISIS a definição de um índice é feita utilizando expressões na linguagem de formato, codificadas em um arquivo chamado FST (Field Select Table), no CouchDB a *view* é implementada pela criação de funções em JavaScript (ou em outras linguagens, como Python ou Erlang, atualmente suportadas para esta finalidade). Uma *view* é formada por uma função chamada **map**, e outra (opcional) chamada **reduce**. A função **map** tem o objetivo de extrair dados dos registros para construir o índice. A função **reduce** serve para agrupar estes dados, produzindo

resultados agregados tais como contagens e somatórias. Para fazer uma estatística por tipos de registro criamos uma view chamada **by_type**, com a seguinte função **map**:

```
function(doc) {
  var key1 = (doc.hasOwnProperty('v5') && (doc.v5.length > 0)) ? doc.v5[0] : '?';
  var key2 = (doc.hasOwnProperty('v6') && (doc.v6.length > 0)) ? doc.v6[0] : '?';
  emit([key1, key2], 1);
};
```

Essencialmente o que esta função faz é verificar se o registro tem os tag 5 e 6 (chamados de **v5** e **v6** na base **lilimp**), e gerar uma entrada no índice na forma de um par [v5, v6], com o valor 1, para efeito de contagem. A view, ou índice, resultante desta função é o seguinte:



Key	Value
["TS", "ms"] ID: 505217	1
["T", "m"] ID: 560414	1
["T", "m"] ID: 560413	1
["T", "m"] ID: 560412	1
["T", "m"] ID: 560411	1
["T", "m"] ID: 560410	1
["T", "m"] ID: 560397	1
["T", "m"] ID: 560396	1
["T", "m"] ID: 560395	1
["T", "m"] ID: 560394	1

*Figura 8: Resultado da função **map** descrita nesta página são 100.000 entradas no índice, onde a chave da entrada é a combinação dos tags 5 e 6, e o valor é o número 1. Isto não tem grande utilidade por si só, até que implementamos a função **reduce** para somar as quantidades por tipo.*

A função **reduce** para este caso é muito simples, apenas somar os valores (todos aqueles números 1 gerados pela função **map**):

```
function(keys, values) {
  return sum(values);
}
```

Porém o resultado da função **reduce** é significativo, como veremos na figura a seguir.

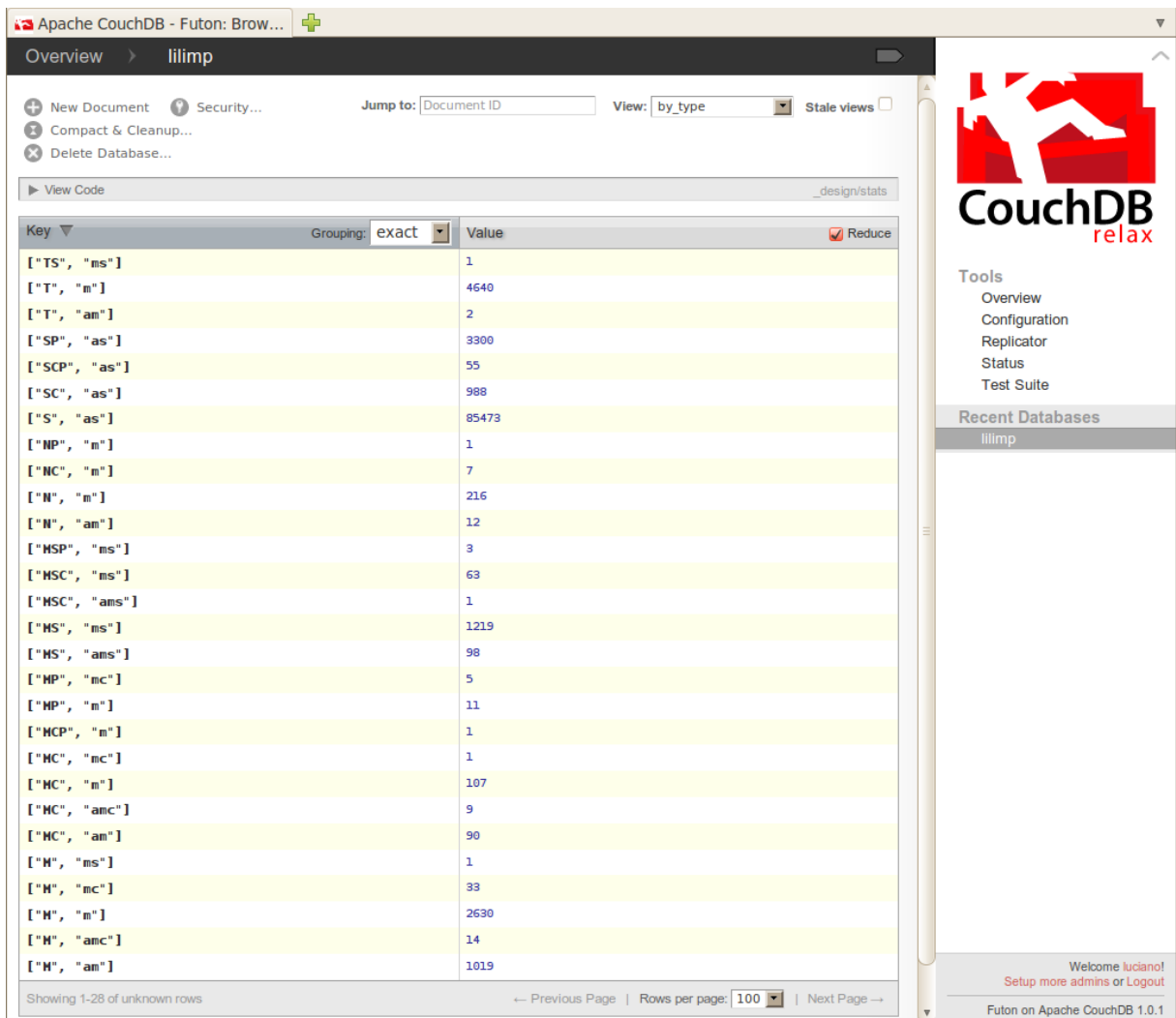


Figura 9: A aplicação da função **reduce** no índice gerado pela função **map** da página anterior é a redução de 100.000 linhas para 28, pelo agrupamento das chaves idênticas. Na coluna da direita, o resultado da função **sum**, utilizada na função **reduce**, é a contagem da quantidade e registros de cada tipo. O tipo mais numeroso, portanto, é o tipo **S:as**, com 85.473 registros, ou seja, 85.5% da amostra LILACS100K.

O mais interessante da arquitetura do CouchDB é que seu único protocolo de entrada e saída é HTTP. Isso significa que todos os resultados de view são imediatamente acessíveis via URLs convencionais, independente do uso da interface amigável do Futon. Por exemplo, a view que definimos nestas páginas pode ser acessada diretamente por uma URL como esta:

`http://192.168.1.9:5984/lilimp/_design/stats/_view/by_type?group=true`

Repare o nome da base **lilimp** e da view, **by_type**, e o parâmetro **group=true**. O resultado de acessar esta URL é um documento JSON contendo os dados da view, como na listagem:

```

{"rows": [
  {"key": ["M", "am"], "value": 1019},
  {"key": ["M", "amc"], "value": 14},
  {"key": ["M", "m"], "value": 2630},
  {"key": ["M", "mc"], "value": 33},
  {"key": ["M", "ms"], "value": 1},
  {"key": ["MC", "am"], "value": 90},
  {"key": ["MC", "amc"], "value": 9},
  {"key": ["MC", "m"], "value": 107},
  {"key": ["MC", "mc"], "value": 1},
  {"key": ["MCP", "m"], "value": 1},
  {"key": ["MP", "m"], "value": 11},
  {"key": ["MP", "mc"], "value": 5},
  {"key": ["MS", "ams"], "value": 98},
  {"key": ["MS", "ms"], "value": 1219},
  {"key": ["MSC", "ams"], "value": 1},
  {"key": ["MSC", "ms"], "value": 63},
  {"key": ["MSP", "ms"], "value": 3},
  {"key": ["N", "am"], "value": 12},
  {"key": ["N", "m"], "value": 216},
  {"key": ["NC", "m"], "value": 7},
  {"key": ["NP", "m"], "value": 1},
  {"key": ["S", "as"], "value": 85473},
  {"key": ["SC", "as"], "value": 988},
  {"key": ["SCP", "as"], "value": 55},
  {"key": ["SP", "as"], "value": 3300},
  {"key": ["T", "am"], "value": 2},
  {"key": ["T", "m"], "value": 4640},
  {"key": ["TS", "ms"], "value": 1}
]}

```

Ou seja, com a construção de uma view formada por duas pequenas funções JavaScript, agora temos um verdadeiro Web Service que informa, sob demanda, quantos registros de cada tipo existem na base **lilimp**. Trata-se de um Web Service praticamente em tempo real porque a atualização dos índices é feita de forma incremental: sempre que um novo registro é inserido na base, o CouchDB registra as views que ficaram desatualizadas, e quando a URL correspondente a uma view “vencida” é acessada, a função **map** correspondente a esta view é executada para cada registro novo ou alterado, e o resultado do **reduce** é recalculado.

Em contraste, construir um Web Service deste tipo sobre uma base ISIS é muito mais complicado, e na prática hoje a BIREME/OPAS/OMS somente publica este tipo de estatística das bases da BVS como resultado de processamentos feitos em lote semanalmente ou com frequência ainda menor.

Mas a view **by_type** que acabamos de construir oferece mais possibilidades. Como a chave utilizada neste índice é uma chave composta formada por duas partes [v5, v6], podemos solicitar que o agrupamento seja feito apenas pelo primeiro nível [v5]. A figura e a listagem a seguir mostram como.

Key	Value
["TS"]	1
["T"]	4642
["SP"]	3300
["SCP"]	55
["SC"]	988
["S"]	85473
["NP"]	1
["NC"]	7
["N"]	228
["MSP"]	3
["MSC"]	64
["MS"]	1317
["MP"]	16
["MCP"]	1
["MC"]	207
["M"]	3697

Figura 10: A mesma view *by_type*, agora com agrupamento apenas pelo nível 1 da chave, correspondente ao tag *v5*. Em vez de 28 tipos e subtipos, agora temos a contagem totalizada por 16 tipos.

Para acessar o resultado agrupado pelo nível 1 da chave, usamos o parâmetro **group_level=1**:

http://192.168.1.9:5984/lilimp/_design/stats/_view/by_type?group=true&group_level=1

O resultado agora é o seguinte documento JSON, uma lista com 16 itens em vez de 28:

```
{
  "rows": [
    {"key": ["M"], "value": 3697},
    {"key": ["MC"], "value": 207},
    {"key": ["MCP"], "value": 1},
    {"key": ["MP"], "value": 16},
    {"key": ["MS"], "value": 1317},
    {"key": ["MSC"], "value": 64},
    {"key": ["MSP"], "value": 3},
    {"key": ["N"], "value": 228},
    {"key": ["NC"], "value": 7},
    {"key": ["NP"], "value": 1},
    {"key": ["S"], "value": 85473},
    {"key": ["SC"], "value": 988},
    {"key": ["SCP"], "value": 55},
    {"key": ["SP"], "value": 3300},
    {"key": ["T"], "value": 4642},
    {"key": ["TS"], "value": 1}
  ]
}
```

Utilizando estes recursos do CouchDB, montamos o relatório detalhado a seguir, que cruza dados da amostra LILACS100K com informações extraídas do arquivo auxiliar xLILACS, uma base ISIS que contém parte das informações do dicionário de dados da LILACS em

formato digital. A base xLILACS é parte da distribuição do aplicativo LILDBI-Web⁸³.

	%	freq.	v5	v6	xv3	tipo de literatura	nível de tratamento	
1	1.02%	1019	M	am	^a	Documento publicado em uma Monografia	analítico monográfico	
2	0.01%	14	M	amc	^a		analítico monográfico de coleção	
3	2.63%	2630	M	m	^f		monográfico	
4	0.03%	33	M	mc	^f		monográfico de coleção	
5	0.00%	1	M	ms	?		monográfico de série	
6	0.09%	90	MC	am	?	Documento de conferência em uma monografia	analítico monográfico	
7	0.01%	9	MC	amc	?		analítico monográfico de coleção	
8	0.11%	107	MC	m	?		monográfico	
9	0.00%	1	MC	mc	?		monográfico de coleção	
10	0.00%	1	MCP	m	?	Documento de projeto e conferência em uma monografia	monográfico	
11	0.01%	11	MP	m	?	Documento de projeto em uma monografia	monográfico	
12	0.01%	5	MP	mc	?		monográfico de coleção	
13	0.10%	98	MS	ams	^a	Documento publicado em uma série monográfica	analítico monográfico de série	
14	1.22%	1219	MS	ms	^f		monográfico de série	
15	0.00%	1	MSC	ams	?	Documento de conferência em uma série monográfica	analítico monográfico de série	
16	0.06%	63	MSC	ms	?		monográfico de série	
17	0.00%	3	MSP	ms	?	Documento de projeto em uma série monográfica	monográfico de série	
18	0.01%	12	N	am	^a	Documento não convencional	analítico monográfico	
19	0.22%	216	N	m	^f		monográfico	
20	0.01%	7	NC	m	?	Documento de conferência em forma não convencional	monográfico	
21	0.00%	1	NP	m	?	Documento de projeto em forma não convencional	monográfico	
22	85.47%	85473	S	as	^f	Documento publicado em uma série periódica	analítico de série	
23	0.99%	988	SC	as	?	Documento de conferência em uma série periódica	analítico de série	
24	0.06%	55	SCP	as	?	Documento de projeto e conferência em uma série periódica	analítico de série	
25	3.30%	3300	SP	as	?	Documento de projeto em uma série periódica	analítico de série	
26	0.00%	2	T	am	^a	Tese, Dissertação (publicado ou não)	analítico monográfico	
27	4.64%	4640	T	m	^f		monográfico	
28	0.00%	1	TS	ms	^f	Tese, Dissertação pertencente a uma série monográfica	monográfico de série	
total:		100000	(registros na amostra LILACS-100k)					

Tabela 8: Frequência dos tipos indicados nos tags 5 e 6 dos 100.000 registros da amostra LILACS-100k.

Legenda

%	porcentagem (frequência/100.000×100)
frequência	número de registros na amostra LILACS-100k com cada combinação de valores nos campos 5 e 6
v5	código do tag 5 em LILACS: Tipo de Literatura
v6	código do tag 6 em LILACS: Nível de Tratamento
tipo de literatura	descrição do valor do tag 5 ⁸⁴
nível de tratamento	descrição do valor do tag 6 ⁸⁵
xv3	subcampo do campo 3 do registro tipo R correspondente em xLILACS ⁸⁶
	^f : fonte (primeiro registro)
	^a : analítica (demais registros)
	? : não consta em xLILACS registro R correspondente a esta combinação dos tags 5 e 6

83 Código-fonte disponível em <<http://reddes.bvsalud.org/projects/lildbi>>. Acesso em 20 nov. 2010.

84 BIREME 2008b, p. 17.

85 BIREME 2008b, p. 18.

86 BIREME 2008a, p. 24.

3.5.2 Investigação: uso de subcampos repetidos em LILACS100K

Uma dúvida comum entre usuários e mesmo desenvolvedores que trabalham com a tecnologia ISIS é o suporte a subcampos repetitivos. Um usuário pode escrever uma determinada marca de subcampo várias vezes em um mesmo campo, e algumas aplicações ISIS não fazem este tipo de validação. No caso específico da metodologia LILACS, o *Manual de Descrição Bibliográfica*⁸⁷ e o *Diccionario de Datos del Modelo LILACS*⁸⁸ não indicam o uso de subcampos repetitivos, mas também não o proibem explicitamente.

Na linguagem de formato do ISIS só é possível acessar a primeira ocorrência de um subcampo, portanto valores digitados em subcampos repetidos não são incluídos nos arquivos invertidos (índices), e conseqüentemente não serão recuperados nas buscas. O aplicativo LILDBI-Web, usado para entrada de dados em LILACS, não permite o uso de subcampos repetitivos. Mas ele não é o único aplicativo utilizado para entrada de dados.

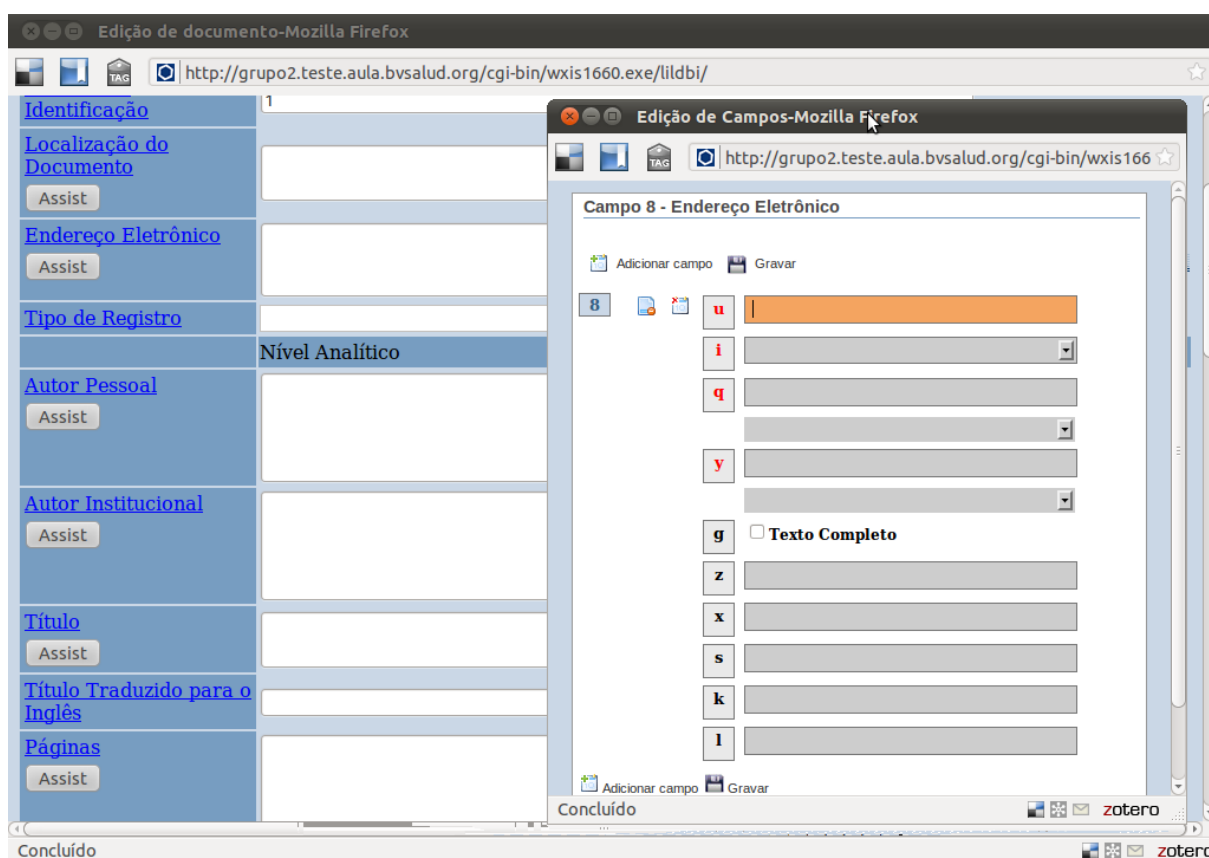


Figura 11: O aplicativo LILDBI-Web permite que o usuário digite subcampos diretamente no campo principal, ou utilize uma tela de pop-up chamada **assist** para facilitar a correta entrada dos valores de subcampos. A tela **assist** não permite a entrada de subcampos repetidos.

87 BIREME, 2008b.

88 BIREME, 2008a.

A questão a investigar então é: existem efetivamente casos de subcampos repetitivos na massa de dados LILACS100K? Especificamente, queremos determinar quantos subcampos repetitivos aparecem nos registros tipo S, os que descrevem artigos de periódicos, e o tipo mais comum (85% da amostra LILACS100K).

Para esta análise, foi criada uma outra view, chamada **rep_subs**, com uma função map bem mais complexa que a anterior:

```
function(doc) {
  // !code vendor/isis/_attachments/listsubfields.js
  var repsubs = '';
  var key, occs, occ;
  if (doc.v5[0] === 'S') {
    for (key in doc) {
      if (key.match(/^v\d+$/)) {
        occs = doc[key];
        for (var i=0; i < occs.length; i++) {
          occ = occs[i];
          repsubs = repeatingsubfields(occ);
          if (repsubs.length > 0) {
            emit(key, [i,repsubs, occ]);
          }
        }
      }
    }
  }
};
```

Não vamos explicar detalhadamente esta função. Um ponto importante está na quinta linha, onde é feito um teste para verificar se o campo **v5** tem valor “S”. Esta é uma prática comum em funções map: selecionar um tipo específico de registro para processar.

A segunda linha faz referência a um outro arquivo JavaScript chamado `listsubfields.js`, cujo código está no repositório do projeto ISIS-NBP da BIREME/OPAS/OMS⁸⁹.

Essencialmente o `listsubfields.js` implementa duas funções:

- `listsubfields`: faz a análise de subcampos em um campo JSON-ISIS tipo 0, 1 ou 6, devolvendo uma string contendo todos os códigos de subcampos encontrados;
- `repeatingsubfields`: devolve uma string contendo os códigos de subcampos repetidos;

⁸⁹ Código-fonte disponível em:

<http://reddes.bvsalud.org/projects/isisnbp/browser/sandbox/luciano.ramalho/couchdb/lilimp/stats/vendor/isis/_attachments/listsubfields.js> ou <<http://bit.ly/i5pxrE>>. Acesso em 20 nov. 2010.

Vejam os alguns casos de campos repetidos encontrados. Na Figura 7 uma das ocorrências é

```
"v82" , ID: 494553, [0, "b", "CLINICAS DENTÁRIAS^BARRETOS,  
SP^BRASIL" ]
```

Interpretando: tag v82 (Região não-DECS) do registro #494553, na ocorrência 0 deste tag, o marcador **b** aparece repetido. Porém examinando o conteúdo do campo vemos que o digitador não entendeu como funciona a notação de subcampos do ISIS. Ele aparentemente imaginou que o circunflexo ^ é um delimitador, como se fosse uma pontuação, e grudou o ^ nas palavras “Barretos” e “Brasil”, sem colocar um código de subcampo. Provavelmente muitos outros registros tem exatamente este tipo de problema, e apenas não foram encontrados nesta análise porque coincidentemente o sinal ^ não foi agregado a palavras com iniciais iguais, como “Barretos” e “Brasil” no exemplo acima. Na verdade, segundo o *Manual de Descrição Bibliográfica da Metodologia LILACS*, não é permitido o uso de subcampos no tag 82.

Porém, o maior grupo de ocorrências de subcampo repetitivo é no tag v14: são 26 casos entre 55. No tag v14 (Páginas, nível analítico), o *Manual de Descrição Bibliográfica da Metodologia LILACS* prevê o uso de dois subcampos:

- ^f (first): número da primeira página do documento;
- ^l (last): número da última página do documento.

Quando o documento é dividido em várias partes não sequenciais, o campo deve ser repetido. Mas alguns documentalistas optam por repetir os subcampos no mesmo campo, como neste exemplo:

```
"v14", ID: 491011, [0, "f1", "^f233^1238^f289^1292"]
```

Neste caso aparecem repetidos os subcampos **f** e **l**. Aparentemente o documento em questão foi publicado em duas partes, da página 233 a 238, e da 289 a 292. O documentalista produziu um campo assim:

```
14 «^f233^1238^f289^1292»
```

Mas deveria ter produzido isto:

```
14 «^f233^1238»  
14 «^f289^1292»
```

Este exemplo ajuda a demonstrar que, do ponto de vista semântico, com a repetição de

campos o uso de subcampos repetidos é desnecessário, ao menos na metodologia LILACS.

A lista completa dos casos de subcampos repetidos nos registros tipo S de LILACS100K pode ser encontrada no wiki do projeto ISIS-NBP⁹⁰.

Além de contribuir para o esclarecimento de dúvidas sobre o uso ou abuso de subcampos repetidos, esta pesquisa demonstra o poder do sistema de indexação por views, programadas em JavaScript, oferecido pelo CouchDB. Enquanto na linguagem de formato do ISIS não se pode sequer encontrar subcampos além da primeira ocorrência, pudemos gerar este relatório esclarecedor graças ao uso de uma linguagem moderna e altamente expressiva, com suporte a expressões regulares e outras operações avançadas com texto. E nossa confiança no resultado é reforçada pela possibilidade de definir e reutilizar funções verificadas por testes automatizados.

⁹⁰ Lista disponível em <<http://reddes.bvsalud.org/projects/isisnbp/wiki/RepeatingSubfieldsinLILACS100k>>. Acesso em 20 nov. 2010.

4 Resultados

4.1 Levantamento de uma base teórica para estudar o modelo de dados ISIS

Quando iniciamos esta pesquisa, não conhecíamos nenhuma literatura científica que desse subsídio para o estudo do modelo de dados ISIS. O termo NoSQL aparece com frequência crescente na Web, mas carece de uma definição rigorosa, e não está associado a uma produção científica. É muito mais um tag, no sentido Web 2.0 do termo, do que um descritor de assunto confiável para pesquisas aprofundadas.

O descritor “semiestruturado”, que encontramos em Hellerstein (2005) foi a chave para uma produção científica altamente relevante para o entendimento do potencial e limitações do modelo de dados ISIS, e dos caminhos para a sua evolução com novas bases tecnológicas.

4.2 Identificação de SGBDs compatíveis com o modelo de dados ISIS

Com base na conceituação, apoiada na literatura, pudemos identificar entre os novos SGBDs semiestruturados a categoria dos produtos orientados a documento, que inclui o CouchDB e o MongoDB. Graças ao suporte ao modelo de dados JSON, estes dois SGBDs podem lidar com registros ISIS, após um processo de conversão simples. Pelas suas características de suporte a replicação, o CouchDB configura-se como uma boa opção para redes de catalogação cooperativa, como é o caso da rede LILACS. O MongoDB, com maior foco em alta performance, pode ser utilizado em bibliotecas digitais públicas de grande tráfego, como é a SciELO.

4.3 Catalogação das variantes de ISIS-JSON

Ao converter registros ISIS para o formato JSON, nos deparamos com algumas escolhas que trazem vantagens e desvantagens. Identificamos alguns padrões de implementação de formatos, e iniciamos um catálogo com sete tipos de ISIS-JSON. Ao nomear estes tipos esperamos contribuir para futuras análises mais aprofundadas sobre os casos de uso mais indicados para cada um deles.

4.4 Ferramentas de conversão

Desenvolvemos algumas ferramentas de conversão, notadamente o script `isis2json.py` e o módulo `iso2709.py`, que serviram para fazer a carga de LILACS100K no CouchDB, mas já tem encontrado uso prático no dia-a-dia da BIREME/OPAS/OMS e do projeto SciELO, onde bancos de dados semiestruturados estão sendo avaliados neste momento. Estes scripts são software livre, e encontram-se no repositório do projeto ISIS-NBP⁹¹ da BIREME/OPAS/OMS.

4.5 Ferramentas de análise

Para lidar com registros ISIS-JSON no CouchDB, criamos diversas funções JavaScript, particularmente para fazer a expansão de campos com subcampos para dicionários ou listas associativas, e para acessar subcampos específicos para a geração de índices. Tais funções também se encontram no repositório do ISIS-NBP.

4.6 Identificação de inconsistências na base LILACS

O estudo sobre subcampos repetidos foi apenas uma entre várias análises que fizemos e revelaram inconsistências na base LILACS. Considerando-se que os 100.000 registros analisados são os mais novos, pode-se supor que existam ainda mais inconsistências no restante da base, até porque, independente de erros de digitação e processamento, é fato que a metodologia LILACS sofreu alterações em seus 25 anos de história, e normalmente os registros antigos não foram reeditados para se adequar às novas regras.

91 Repositório disponível em <<http://reddes.bvsalud.org/projects/isisnbp/>>. Acesso em 20 nov. 2010.

5 Conclusão

Embora o modelo de dados ISIS atenda muito bem às necessidades de aplicações em biblioteconomia e ciência da informação, outros sistemas de bancos de dados oferecem hoje o modelo de dados semiestruturado, compatível e mais flexível que o modelo ISIS, portanto capaz de acomodar bases ISIS atuais sem mudanças em seus dicionários de dados e metodologias.

A família ISIS não contou, em sua infância e adolescência, com um arcabouço teórico que norteasse sua evolução e sinalizasse os caminhos potencialmente infrutíferos. Mas desde os anos 90 este arcabouço começou a se organizar, sob a bandeira do modelo de dados semiestruturado. A pesquisa científica sobre este modelo de dados, suas linguagens, casos de uso e questões de implementação oferecem apoio para a evolução das aplicações de catalogação que hoje dependem de um modelo de dados, implementado nos sistemas ISIS, que foi projetado desde o início para apoiar registros bibliográficos.

Do ponto de vista prático, este trabalho provou que são simples as ferramentas e técnicas necessárias para a migração de bases ISIS para um SGBD orientado a documentos, graças à semelhança entre os modelos de dados. Mostrou também que o CouchDB oferece recursos tão ou mais sofisticados que o ISIS para a formulação de índices de pesquisa. Finalmente, deixou claro que, por sua natureza voltada para aplicações Web, o CouchDB pode agilizar o desenvolvimento de Web Services e aplicações AJAX que consomem tais serviços.

5.1 Limitações da pesquisa

A massa de dados escolhida, embora seja bastante sofisticada do ponto de vista de seu dicionário de dados e da diversidade de tipos de registro, não é suficientemente grande para permitir avaliações sobre o desempenho do CouchDB, mesmo informalmente. A BIREME/OPAS/OMS opera um espelho com cerca de 18 milhões de registros da base MEDLINE da National Library of Medicine dos EUA, ou seja, uma massa de dados quase 180 vezes maior do que a amostra utilizada neste estudo. É evidente que para lidar com uma massa de dados nesta escala precisaríamos de mais recursos computacionais (os testes para este trabalho foram feitos em um notebook básico de 2008 e em um desktop de 5 anos atrás). Outra limitação relevante foi o foco apenas no CouchDB. Repetir os experimentos com o MongoDB (ou outros sistemas com modelo de dados parecido) traria parâmetros de

comparação quanto a facilidade de uso, recursos e desempenho.

Finalmente, embora nosso levantamento bibliográfico tenha localizado muitos artigos de periódicos, nosso estudo ficou concentrado nos livros. Sendo este o nosso primeiro contato com o tema do modelo semiestruturado, nos beneficiamos da didática, da visão panorâmica e revisão da literatura que são características de livros texto. Por outro lado, perdemos o maior aprofundamento e os resultados mais recentes que só os artigos podem trazer.

5.2 Indicações para futuras pesquisas e indicações práticas

5.2.1 APIs para especificação de esquemas

O acúmulo de inconsistências ao longo do tempo é um dos maiores desafios dos sistemas de bancos de dados “schemaless”. Se o SGBD não tem mecanismos próprios para garantir a aderência a um dicionário de dados, então as aplicações precisam se encarregar disso de forma consistente e prática para o desenvolvedor. Uma saída para isso é o uso de APIs declarativas de esquemas de dados, nos moldes do ORM⁹² do framework Django (DJANGO 2010), ou de outros projetos mais recentes. Nos últimos dias desta pesquisa descobrimos que existe no site da IETF⁹³ um “draft” (rascunho) de RFC⁹⁴ chamado JSON-SCHEMA⁹⁵ que é uma forma de definir esquemas JSON em JSON, da mesma forma que XML Schema utiliza a notação do XML para definir a estrutura de documentos XML.

Além de facilitar a manutenção da consistência dos dados, uma API para especificação de esquemas pode trazer outros benefícios práticos como:

- apoio à geração automática de formulários de entrada de dados, com validação no navegador;
- isolamento da aplicação em relação ao banco de dados específico;

Em relação a este último ponto, o paralelo com os ORMs é evidente: um dos principais benefícios de um ORM é permitir que uma aplicação acesse dados de SGBDs diferentes de forma transparente para o programador. Da mesma forma, é concebível que uma API para definição de esquemas semiestruturados possa permitir a armazenagem dos dados em CouchDB, MongoDB, Google Datastore, Amazon SimpleDB ou mesmo em versões recentes do PostgreSQL que incorporaram campo tipo dicionário.

Enquanto esta pesquisa foi realizada, implementamos como prova de conceito uma API para definições de esquema chamada ISIS-DM, que se encontra no repositório do ISIS-NBP e foi objeto de artigo publicado no XI Workshop de Software Livre do FISL 2010 (RAMALHO, 2010). Posteriormente, aprofundamos esta pesquisa paralela com o script `schematize.py`

92 ORM: Object Relational Mapper – camada de integração entre um modelo de dados orientado a objetos e um sistema de banco de dados relacional.

93 Internet Engineering Task Force, organismo que define padrões e normas técnicas para a Internet.

94 RFC: Request For Comments – modelo de documento e fluxo para aprovação de padrões e normas do IETF.

95 JSON-SCHEMA disponível em <<http://tools.ietf.org/html/draft-zyp-json-schema-03>>. Acesso em 22 nov. 2010.

para a geração automática de um esquema ISIS-DM a partir da análise de uma massa de dados ISIS-JSON. Este programa também está disponível no repositório do ISIS-NBP, e foi um dos temas de artigo aceito para apresentação oral no XI ENANCIB (MUCHERONI, 2010). O próximo passo para o ISIS-DM seria sua integração com o driver de acesso a um SGBD semiestruturado.

5.2.2 Atualização automática de dados duplicados

Durante nossos estudos sobre a operação da base LILACS aprendemos que existe muita redundância de dados entre os registros. Por exemplo, os registros analíticos que descrevem os capítulos de um livro embutem cópias de muitos dos campos do registro monográfico que descreve o tal livro. Na terminologia da BIREME/OPAS/OMS, o registro monográfico é um “registro fonte”, porque dele originam dados que vão para os registros analíticos. Essa redundância é desejável para assegurar que os registros sejam completos, facilitando seu uso e distribuição. Porém, como a redundância tende a gerar inconsistências, as chamadas “anomalias de atualização” (mesmo considerando que registros bibliográficos raramente são atualizados, erros acontecem e precisam ser corrigidos).

A redundância de dados, ou desnormalização, é também uma prática comum para aumentar o desempenho e a escalabilidade de aplicações Web. Em sua palestra “Não se preocupe mais, use App Engine” no evento The Developer's Conference 2010⁹⁶ em São Paulo, Rodolpho Eckhardt, engenheiro do Google relatou técnicas para lidar com anomalias de atualização em aplicações hospedadas no App Engine, que dependem do Google Datastore como sistema de banco de dados. O que falta no Datastore, como atualmente oferecido pelo Google App Engine, é uma API para especificar interdependências de dados entre registros que devem ter campos redundantes por algum motivo.

Para exemplificar um caso de uso de tal API, considere a base LILACS, onde há registros de nível monográfico, que descrevem livros, e registros de nível analítico, que descrevem cada capítulo separadamente. Para que o registro de nível analítico seja completo é necessário que ele descreva não apenas o próprio capítulo, com seu título, autores etc., mas também o livro onde o capítulo se insere, que também tem seu título, autores e outros metadados.

⁹⁶ Programa disponível em <<http://www.thedevelopersconference.com.br/tdc/2010/sp/trilha-python>>. Acesso em 21 nov. 2010.

Usando a sintaxe da API ISIS-DM, poderíamos declarar dois tipos de registros assim:

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

class Chapter(CheckedModel):
    title = SingularProperty(required=True, subfields='s')
    creators = PluralProperty(required=True, subfields='yr')
    book = ObjectReference(Book, required=True)
    book_title = MirrorProperty(book, 'title', deferred=True)
    book_creators = MirrorProperty(book, 'creators', deferred=True)
```

Neste exemplo, o registro de nível analítico é o `Chapter` (capítulo), e o registro de nível monográfico é `Book` (livro). A relação é estabelecida através do atributo `book` em `Chapter`, que é do tipo `ObjectReference`. Os campos `book_title` e `book_creators` do capítulo são `MirrorProperty`, espelhos dos campos correspondentes no `book` vinculado. O parâmetro `deferred=True` significa que a replicação dos dados será postergada, ou seja, quando uma instância de `Chapter` for criada e salva no SGBD, os campos espelho podem ser inicialmente nulos. Posteriormente, um processo assíncrono pode varrer as instâncias de `Chapter` atualizando os campos espelho com os valores dos registros fonte correspondentes. Se especificarmos `deferred=False`, ao salvar um registro `Chapter` a camada de persistência por trás da API deverá recuperar o livro relacionado para preencher os dados espelhados antes de salvar o registro do capítulo.

Em vez da normalização a todo custo, os desafios da Web deixam cada vez mais atraente a ideia de que...

Duplicated data is bad only insofar as the effort to keep it consistent is burdensome.⁹⁷

A implementação da API descrita acima pode vir a ser uma contribuição útil para a publicação de bases bibliográficas na Web, ou para outras aplicações que, por qualquer razão, necessitem trabalhar com dados desnormalizados.

97 Tradução: “Dados duplicados são ruins somente na medida em que o esforço para mantê-los consistentes é oneroso.” MONASH, C. A. Counterpunching in the DBMS2 Debate. In: Computerworld, 10 out. 2005. Disponível em: http://www.computerworld.com/s/article/105255/Counterpunching_in_the_DBMS2_Debate. Acesso em 21 nov. 2010.

Glossário

banco de dados: Termo com muitas definições, por isso evitamos usá-lo nesta monografia.

No contexto deste trabalho, adotamos os termos mais específicos **base de dados**, **objeto banco de dados**, sistema de banco de dados e sistema gerenciador de banco de dados.

base de dados: Coleção organizada e inter-relacionada de dados⁹⁸.

campo: Um dos elementos que compõe um registro. Em registros MARC ou ISIS, cada campo é identificado por um **tag** numérico, podem existir várias ocorrências do mesmo campo em um registro, e um campo pode ser dividido em **subcampos**.

dicionário: Tipo de dado composto, que representa uma associação não ordenada entre chaves únicas e valores⁹⁹. *Array associativo*, *hashmap* e *mapping* são alguns sinônimos de **dicionário** usados em diferentes linguagens de programação. Dada uma chave, o acesso ao um item correspondente é muito eficiente, e praticamente não depende do número de itens no dicionário¹⁰⁰. No formato JSON, o dicionário é denominado *object* (como na linguagem JavaScript), e as chaves são sempre **strings**. Um dicionário JSON tem a seguinte sintaxe:

```
{ "AC" : "Acre", "AL" : "Alagoas", "AP" : "Amapá" }
```

lista: Tipo de dado composto, que representa uma sequência ordenada de itens¹⁰¹. Em JSON a lista é chamada de *array*, como na linguagem JavaScript. Embora na maioria das linguagens de programação um *array* seja uma sequência de itens do mesmo tipo, em JavaScript e JSON não existe esta restrição, e os itens podem ser de tipos misturados. Por isso adotamos o termo lista. A sintaxe de uma lista em JSON é:

```
[ "Acre", "Alagoas", "Amapá" ]
```

lista associativa: Estrutura de dados que representa uma associação ordenada entre chaves (não necessariamente únicas) e valores¹⁰². Não confundir com *array associativo*, que é

98 FERREIRA, 2009.

99 BLACK, 2010.

100 Na maioria das linguagens de programação, os dicionários são implementados de forma eficiente usando-se uma tabela de dispersão. Ver: TABELA DE DISPERSÃO. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2010. Disponível em: <http://pt.wikipedia.org/w/index.php?title=Tabela_de_dispers%C3%A3o&oldid=22567329>. Acesso em 20 nov. 2010.

101 LIST (COMPUTING). In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2010. Disponível em: <[http://en.wikipedia.org/w/index.php?title=List_\(computing\)&oldid=376178627](http://en.wikipedia.org/w/index.php?title=List_(computing)&oldid=376178627)>. Acesso em 20 nov. 2010.

102 ASSOCIATION LIST. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2010. Disponível em: <http://en.wikipedia.org/w/index.php?title=Association_list&oldid=322436433>. Acesso em 20 nov. 2010.

sinônimo de **dicionário**. Uma lista associativa pode ser utilizada para representar o conteúdo de qualquer dicionário, sem perda de informação. O contrário não é verdadeiro, pois o dicionário não preserva a ordem dos itens, e não permite vários itens com a mesma chave. Por outro lado, dada uma chave, para se obter o valor correspondente em uma lista associativa é preciso percorrer todos os itens até encontrar (ou não) a chave correspondente. No caso geral, em que a lista associativa não está ordenada pelas chaves, o tempo médio de acesso é proporcional ao número de itens. No formato JSON não existe um tipo de dado primitivo para representar listas associativas, mas elas podem ser facilmente criadas pelo aninhamento de **listas** desde que cada lista interna tenha apenas dois itens. Por exemplo:

```
[["AC", "Acre"], ["AL", "Alagoas"], ["AP", "Amapá"]]
```

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 SQLite5 é um motor de banco de dados relacional e CISIS é um motor de banco de dados semiestruturado.

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

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

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 esta definição, todo SGBD é um **sistema de banco de dados**, mas nem todo sistema de banco de dados é um SGBD. Por exemplo, o PostgreSQL é um sistema gerenciador de banco de dados relacional, e o Apache CouchDB é um SGBD semiestruturado.

string: Em computação, uma string é uma cadeia de caracteres, em geral utilizada para representar textos legíveis. Em registros ISIS, todos os campos contêm strings. Em registros JSON, string é apenas um dos tipos de dados possíveis.

subcampo: Em registros MARC ou ISIS, cada campo pode ser subdividido em partes chamadas subcampos. Nos sistemas da família ISIS, os subcampos são delimitados por marcadores no formato ^x, onde x pode ser um dígito ou uma letra.

subcampo principal: Em registros ISIS, o subcampo principal é a parte do campo que precede qualquer delimitador de subcampo, ou o conteúdo inteiro do campo, caso não haja nenhum delimitador de subcampo. Por exemplo, o subcampo principal do campo «Lewis Carroll^1USP^2ECA^pBrasil» contém o nome “Lewis Carroll”.

tag: Literalmente, etiqueta ou identificação¹⁰³. Em registros ISIS, os tags são números entre 1 e 32767, que identificam os campos. O significado do tag depende da metodologia utilizada para a criação do registro. Por exemplo, na metodologia LILACS, o tag 10 identifica o campo Autor Pessoal (nível analítico). Em linguagens de marcação, como XML e HTML, um tag identifica um elemento, ou nodo, da estrutura do documento. Ao representar registros de bases de dados em XML é comum que os tags identifiquem o tipo do registro e seus campos. No exemplo abaixo temos os tags <work>, <title> e <author>:

```
<work><title>O Alienista</title><author>Machado de Assis</author></work>
```

Bibliografia

1. 10GEN, INC. **MongoDB** (Web site). Disponível em: <<http://www.mongodb.org/>>. Acesso em 10 ago. 2010.
2. ABITEBOUL, Serge; BUNEMAN, Peter; SUCIU, Dan. **Data on the Web: From Relations to Semistructured Data and XML**. San Francisco: Morgan Kaufmann, 1999.
3. AMAZON.COM. **Amazon SimpleDB**. Disponível em: <<http://aws.amazon.com/simpledb/>>. Acesso em 10 ago. 2010.
4. ANDERSON, J. Chris; LEHNARDT, Jan; SLATER, Noah. **CouchDB: The Definitive Guide**. Sebastopol: O'Reilly Media, 2010. Disponível em <<http://guide.couchdb.org/>>. Acesso em 10 nov. 2010.
5. APACHE FOUNDATION. **The Apache Cassandra Project**. Disponível em: <<http://cassandra.apache.org/>>. Acesso em 14 nov. 2010.
6. BANKER, Kyle. **MongoDB in Action (pre-print ebook)**. Greenwich: Manning, 2010. Capítulo 1 disponível gratuitamente em: <<http://www.manning.com/banker/>>. Acesso em 22 nov. 2010.
7. BIREME/OPAS/OMS (2005). **Conceitos Básicos de Bases de Dados CDS/ISIS: Iniciando o Uso do CISIS – Versão 3.x**. São Paulo, SP, 2005. Disponível em: <<http://bvsmodelo.bvsalud.org/download/cisis/CISIS-ConceitosBasicos-pt.pdf> >. Acesso em 22 nov. 2010.
8. BIREME/OPAS/OMS (2006a). **Linguagem de Formato CISIS – 4.x**. São Paulo, SP, 2006. Disponível em: <<http://bvsmodelo.bvsalud.org/download/cisis/CISIS-LinguagemFormato4-pt.pdf> >. Acesso em 22 nov. 2010.
9. BIREME/OPAS/OMS (2006b). **Utilitários CISIS – Manual de Referência – Versão 5.2**. São Paulo, SP, 2006. Disponível em: <<http://bvsmodelo.bvsalud.org/download/cisis/CISIS-ManualReferencia-pt-5.2.pdf> >. Acesso em 22 nov. 2010.

10. BIREME/OPAS/OMS (2008a). **Diccionario de Datos del Modelo LILACS Versión 1.6a**. São Paulo, SP, 2008.
11. BIREME/OPAS/OMS (2008b). **Metodologia LILACS: Manual de Descrição Bibliográfica 7 ed.** São Paulo, SP, 2008.
12. BIREME/OPAS/OMS (2010a). **ISIS Network Based Platform**. Disponível em: <<http://reddes.bvsaude.org/projects/isisnbp>>. Acesso em 10 set. 2010.
13. BIREME/OPAS/OMS (2010b). **Modelo da Biblioteca Virtual em Saúde – Metodologias e aplicativos – LILACS**. Disponível em: <<http://bvsmodelo.bvsalud.org/php/level.php?lang=pt&component=27&item=6> >. Acesso em 22 nov. 2010.
14. BIREME/OPAS/OMS (2010c). **Portal LILACS**. Disponível em: <<http://lilacs.bvsalud.org/>>. Acesso em 22 nov. 2010.
15. BLACK, Paul E. **dictionary**. In: **Dictionary of Algorithms and Data Structures** [online], BLACK, Paul E. (Ed.), U.S. National Institute of Standards and Technology. Disponível em: <<http://xw2k.nist.gov/dads/HTML/dictionary.html>>. Acesso em 15 nov. 2010.
16. BUXTON, Andrew; HOPKINSON, Alan. **The CDS/ISIS for Windows Handbook**. Paris: UNESCO/CI, 2001.
17. BUXTON, Andrew; HOPKINSON, Alan. **The CDS/ISIS Handbook**. London: Library Association Publishing, 1994.
18. CHANG, Fay; DEAN, Jeffrey; GHEMEWAT, Sanjay; et al. **Bigtable: A Distributed Storage System for Structured Data**. In: OSDI'06: Seventh Symposium on Operating System Design and Implementation, Seattle, WA, 2006. Disponível em: <<http://labs.google.com/papers/bigtable.html>>. Acesso em 10 set. 2010.
19. CODD, E. F. **A Relational Model of Data for Large Shared Data Banks**. Communications of the ACM, v. 13, n. 6, p. 377-387, jun. 1970. Disponível em:

- <<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.98.5286&rep=rep1&type=pdf>>. Acesso em 21 nov. 2010.
20. CROCKFORD, Douglas (2006a). **JSON: The Fat-Free Alternative to XML**. JSON.ORG, 2006. Disponível em: <<http://www.json.org/fatfree.html>>. Acesso em 15 nov. 2010.
 21. CROCKFORD, Douglas (2006b). **RFC-4627: The application/json Media Type for JavaScript Object Notation (JSON)**. The Internet Society, 2006. Disponível em: <<http://tools.ietf.org/html/rfc4627>>. Acesso em 15 nov. 2010.
 22. CUNHA, Murilo Bastos da; CAVALCANTI, Cordélia Robalinho de Oliveira. **Dicionário de Biblioteconomia e Arquivologia**. Brasília: Brique de Lemos/Livros, 2008.
 23. DATE, C. J. **Database in Depth: Relational Theory for Practitioners**. Sebastopol: O'Reilly Media, 2005.
 24. DAUPHIN, Jean-Claude. **J-ISIS Quick Tutorial**. Disponível em: <<http://kenai.com/projects/j-isis/downloads>>. Acesso em 15 nov. 2010.
 25. DE SMET, Egbert. **New Challenges for a New Future of ISIS** (slides em formato PDF). In: III Congresso Internacional de Usuários de CDS/ISIS (ISIS3WC), Rio de Janeiro, set. de 2008. Disponível em: <<http://www.eventos.bvsalud.org/agendas/isis3/program.php>>. Acesso em 21 nov. 2010.
 26. DECANDIA, Giuseppe; HASTORUN, Deniz; JAMPANI, Madan; et al. **Dynamo: Amazon's Highly Available Key-Value Store**. In: Proceedings of the 21st ACM Symposium on Operating Systems Principles, Stevenson, WA, 2007.
 27. DJANGO SOFTWARE FOUNDATION. **Django Project Model Syntax**. Disponível em: <<http://www.djangoproject.org>>. Acesso em 10 set. 2010.
 28. ECMA INTERNATIONAL. **Standard ECMA-262 - ECMAScript Language Specification - 5th edition (December 2009)**. Disponível em: <<http://www.ecma-international.org/publications/standards/Ecma-262.htm>>. Acesso em 21 nov. 2010.

29. EURE, I. **Looking to the Future with Cassandra**. In: Digg Technology Blog. Disponível em: <<http://about.digg.com/blog/looking-future-cassandra>>. Acesso em 9 set. 2010.
30. FERREIRA, Aurélio Buarque de Hollanda. **Novo Dicionário da Língua Portuguesa**. 4 ed. Curitiba: Positivo, 2009.
31. GOOGLE INC. **The Python Datastore API**. Disponível em: <<http://code.google.com/appengine/docs/python/datastore/>>. Acesso em ago. 2010.
32. HAROLD, Elliott R. **Effective XML: 50 Specific Ways to Improve Your XML**. Reading, MA: Addison-Wesley, 2004.
33. HELLERSTEIN, Joseph M.; STONEBRAKER, Michael (eds). **Readings in Database Systems**, 4th Edition. Cambridge, MA: MIT Press, 2005.
34. HASSELBRING, W. **Information System Integration**. In: Communications of the ACM, Volume 43, Issue 6, p. 32-38, 2000.
35. INFOLAB, Stanford University. **Lore: a Database Management System for XML**. Disponível em: <<http://infolab.stanford.edu/lore/home/index.html>>. Acesso em 24 ago. 2010.
36. **Introdução ao JSON**. Disponível em: <<http://www.json.org/json-pt.html>>. Acesso em 15 nov. 2010.
37. ISO2709. **ISO 2709:2008 – Information and documentation – Format for information exchange**. Disponível para compra em: <http://www.iso.org/iso/catalogue_detail.htm?csnumber=41319>. Acesso em 22 nov. 2010.
38. ISO8879. **ISO 8879:1986 – Information processing – Text and office systems – Standard Generalized Markup Language (SGML)**. Disponível para compra em: <http://www.iso.org/iso/catalogue_detail.htm?csnumber=16387>. Acesso em 22 nov. 2010.

39. ISO/IEC16262. **ISO/IEC 16262:2002 – Information technology – ECMAScript language specification** . Disponível para compra em:
<http://www.iso.org/iso/catalogue_detail.htm?csnumber=33835 >. Acesso em 22 nov. 2010.
40. J-ISIS PROJECT. **Web-JISIS Reference Manual**. Disponível em: <Web-JISIS Reference Manual <http://kenai.com/projects/j-isis/downloads>>. Acesso em 22 nov. 2010.
41. LENNON, Joe. **Beginning CouchDB**. Apress, 2009.
42. MARTELLI, Alex. **Python in a Nutshell, 2nd Edition**. Sebastopol: O'Reilly Media, 2006.
43. MARTINEZ USERO, J.A.; LARA NAVARRA, P. **La Interoperabilidad de la Información**. Barcelona: UOC, 2007.
44. MERRIMAN, Dwight. **Comparing Mongo DB and Couch DB**. MongoDB.org.
Disponível em:
<<http://www.mongodb.org/display/DOCS/Comparing+Mongo+DB+and+Couch+DB>>.
Acesso em 22 nov. 2010.
45. MUCHERONI, M. L.; RAMALHO, L. **Modelos de Dados para Bases Bibliográficas: Legado, Padrões e Alternativas**. In: XI ENANCIB – Encontro Nacional de Pesquisa em Ciência da Informação, 2010. Rio de Janeiro. Anais Eletrônicos. Disponível em:
<<http://congresso.ibict.br/index.php/enancib/xienancib/paper/view/529>>. Acesso em 20 nov. 2010.
46. **NoSQL**. In: Wikipédia: a enciclopédia livre. Disponível em:
<<http://en.wikipedia.org/wiki/NoSQL>>. Acesso em 14 nov. 2010.
47. ORNAGER, Susanne. **Suplement do CDS/ISIS Reference Manual (CDS/ISIS Version 3.0)**. Bonn: DSE – Deutsche Stiftung für Internationale Entwicklung, 1993.
48. PLUGGE, Eelco; MEMBREY, Peter; HAWKINS, Tim. **The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing**. Apress, 2010.

49. RAMALHO, L. **Implementing a Modern API for CDS/ISIS, a Classic Semistructured NoSQL Database**. In: COSTA, Cristiano A.; TODT, Eduardo. (Ed.) 11º Fórum Internacional do Software Livre, XI Workshop sobre Software Livre, Porto Alegre, 2010. p. 42-47.
50. SCHWARTZ, B. **A Gentle Introduction to CouchDB for Relational Practitioners**. Disponível em: <<http://www.couch.io/gentle-introduction>>. Acesso em 15 nov. 2010.
51. SETZER, Valdemar; CORRÊA DA SILVA, Flávio. **Bancos de Dados: Aprenda o que São, Melhore seu Conhecimento, Construa os Seus**. 1 ed. São Paulo: Edgard Blücher, 2005.
52. SUCIU, Dan. **Semi-Structured Data Model**. In: LIU, L.; ÖZSU, M. T. Encyclopedia of Database Systems: Springer, 2009. p. 2601-2605.
53. TOK, Wang Ling; LEE, Mong Li; DOBBIE, Gillian. **Semistructured Database Design**. Boston: Springer Science, 2005.
54. UNESCO. **Mini-micro CDS/ISIS Reference Manual (Version 2.3)**. Paris, 1989.
55. UPENN. **Database Group, Semistructured data, and XML**. University of Pennsylvania. Disponível em: <http://db.cis.upenn.edu/research/SS_XML.html>. Acesso em ago. 2010.
56. ZYP, Kris. **A JSON Media Type for Describing the Structure and Meaning of JSON Documents**. Disponível em: <<http://tools.ietf.org/html/draft-zyp-json-schema>>. Acesso em 15 nov. 2010.