5
Estudo de Caso

Para validar o Framework MsA, realizamos um estudo de caso para a materialização e manutenção de ligações entre dados de professores cadastrados no sistema RPA@PUC e publicados na Web pela DBLP. O framework desenvolvido monitora alterações em ambas as bases e realiza os procedimentos de identificação, materialização e manutenção necessários. Para reproduzir um uso real do framework, também publicamos os dados do RPA@PUC em RDF com suas ligações owl:sameAs materializadas.

5.1 Bases de Dados utilizadas

5.1.1. RPA@PUC

RPA@PUC é a sigla para a Rede de Perfis Acadêmicos da PUC-Rio, um sistema interno da universidade no qual são cadastrados dados relativos aos programas de pós-graduação Stricto Sensu. De acordo com o escopo desse trabalho, foram utilizados apenas os dados referentes ao Departamento de Informática.

5.1.2. DBLP

DBLP é uma sigla para Digital Bibliography & Library Project (http://dblp.uni-trier.de/). Trata-se de um repositório bibliográfico de ciência da computação hospedado na Universidade de Trier na Alemanha.

O banco de dados da DBLP pode ser obtido na Internet através do endereço http://dblp.uni-trier.de/xml/ (Ley, 2009). Os registros bibliográficos estão contidos em um gigantesco arquivo XML. No ano de 2009, essa base havia listado mais de um milhão e duzentos mil artigos.

Em sua versão on-line, as páginas dos autores são arquivos HTML estáticos gerados diariamente.

A DBLP disponibiliza URIs locais ou globais. Uma URI global é uma URI padrão da Internet (HTTP, FTP, etc.). Se não começar com o nome do protocolo seguido de dois pontos, então trata-se de uma URI local que aponta para dentro do Web site.

Para obter uma URI válida, simplesmente adiciona-se uma URI base de um servidor DBLP como prefixo. As bases dos servidores mais estáveis estão listadas a seguir.

- Servidor original da Universidade de Trier, Alemanha: http://www.informatik.uni-trier.de/~ley/db/;
- Servidor alternativo em Trier: http://dblp.uni-trier.de/db/. A seção de author pages deste servidor não inclui, intencionalmente, a facilidade de busca completa;
- Servidor "espelho" hospedado pela Association for Computing Machinery – Special Interest Group on Management of Data (ACM SIGMOD): http://www.sigmod.org/dblp/db/.

5.2 Montagem

Para execução do estudo de caso, obtivemos uma cópia das tabelas e dados do RPA@PUC referentes aos professores do Departamento de Informática. Através dessa cópia, criamos uma base de dados espelho na qual o estudo foi realizado. Com a base montada (Figura 7), executamos o script para instalação do Framework MsA.

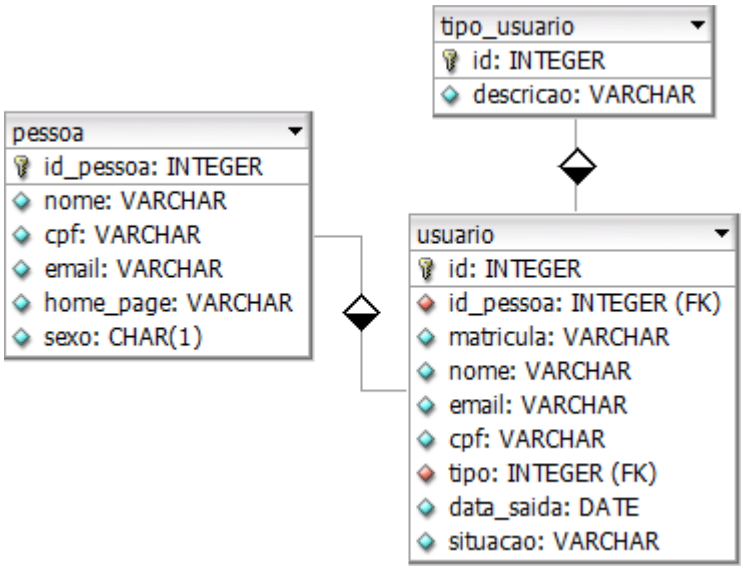

Figura 7: Tabelas do RPA@PUC referentes a dados de professores

Ao instalá-lo (Figura 8), o script criou as tabelas de controle para a materialização e manutenção das ligações externas (*sameas*, *sameas_uri* e *resolver*), e a função principal (MsA_tg_main) responsável por executar as manutenções necessárias para cada operação ocorrida nas tabelas do RPA@PUC que terão ligações externas publicadas na Web.

```
> ./install.sh
Carregando as configuracoes do banco...
Testando a conexao com o banco...
OK!
Verificando se o framework ja foi instalado...
NAO!

INSTALANDO...
psql:sql/framework-install.sql:17: NOTA:  CREATE TABLE criará sequênc
ia implícita "msa_resolver_id_resolver_seq" para coluna serial "msa_r
esolver.id_resolver"
psql:sql/framework-install.sql:17: NOTA:  CREATE TABLE / PRIMARY KEY
criará índice implícito "msa_resolver_pkey" na tabela "msa_resolver"
psql:sql/framework-install.sql:17: NOTA:  CREATE TABLE / UNIQUE criar
á índice implícito "msa_resolver_name_key" na tabela "msa_resolver"
psql:sql/framework-install.sql:23: NOTA:  CREATE TABLE criará sequênc
ia implícita "msa_sameas_id_sameas_seq" para coluna serial "msa_samea
s.id_sameas"
psql:sql/framework-install.sql:23: NOTA:  CREATE TABLE / PRIMARY KEY
criará índice implícito "msa_sameas_pkey" na tabela "msa_sameas"
psql:sql/framework-install.sql:35: NOTA:  CREATE TABLE criará sequênc
ia implícita "msa_sameas_uri_id_resolver_seq" para coluna serial "msa
_sameas_uri.id_resolver"
psql:sql/framework-install.sql:35: NOTA:  CREATE TABLE / PRIMARY KEY
criará índice implícito "msa_sameas_uri_pkey" na tabela "msa_sameas_u
ri"

Fim.
```

Figura 8: Execução do script de instalação do Framework MsA

Para realizar a busca por essas ligações, é necessário pelo menos um módulo de resolução de entidades. Para este estudo de caso, implementamos um módulo que obtém uma URI da DBLP a partir do nome de um professor. Ele recebe uma string com um nome próprio e retorna uma string com a URI da DBLP correspondente a esse nome.

Com o módulo implementado, o mapeamos dentro da tabela de controle *resolver*, juntamente com a tabela que irá acioná-lo e a coluna da qual ele obterá o dado a ser consultado na DBLP. A Figura 9 apresenta a criação desse mapeamento através do script add_mapping.sh. Ele recebe como parâmetro o nome do resolvedor (*dblp*), a tabela a ser mapeada (pessoa), a coluna dessa tabela que terá seu valor consultado na DBLP (nome), e o comando que executará o módulo de resolução de entidades (“java –cp src dblp_sameas”).

```
> ./add-mapping.sh dblp pessoa nome "java -cp src dblp_sameas"
Carregando as configuracoes do banco...
Testando a conexao com o banco...
OK!
Verificando se os parametros sao validos...
OK!
Sucesso!
Fim.
```

Figura 9: Criação do mapeamento entre a tabela e o módulo de resolução de entidades

Conforme adiantado no mapeamento anterior, escolhemos a tabela pessoa (com dados pessoais dos professores e alunos do Departamento de Informática da PUC-Rio) para ser monitorada e ter suas ligações externas materializadas. Para realizar esse mapeamento, o framework adiciona a coluna *id_sameas* a essa tabela (Figura 10), juntamente com dois gatilhos para cobrir as possíveis operações a serem identificadas – “before” para operações do tipo insert e update, e “after” para operações do tipo delete. Esses gatilhos fazem uso da função principal criada na instalação.

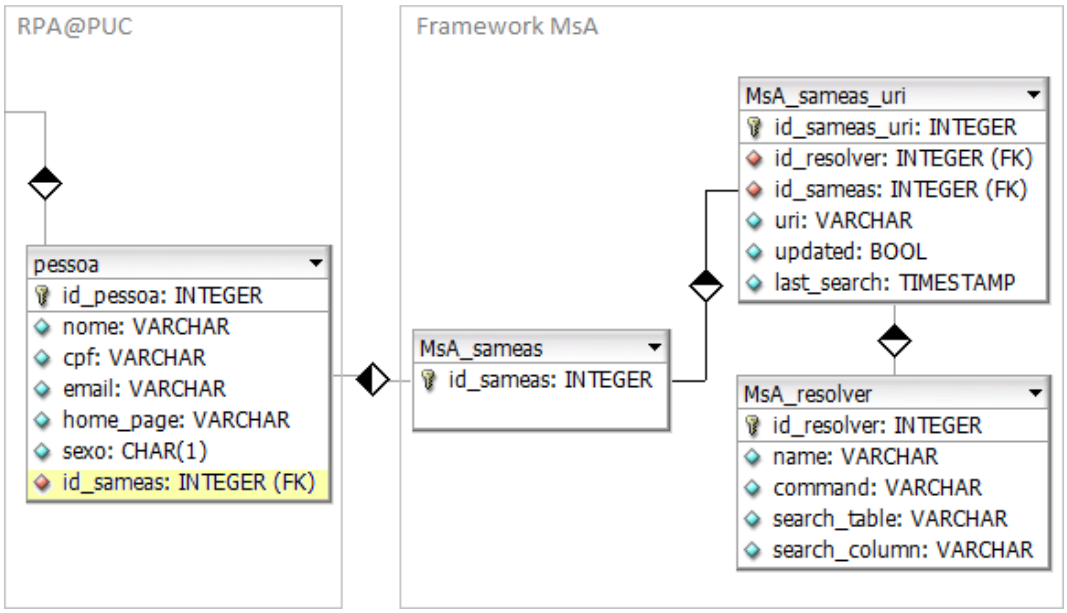

Figura 10: Modelo do banco de dados após a instalação e o mapeamento do Framework MsA

A tabela pessoa, ao receber alguma operação em seus registros, acionará o gatilho correspondente. O mesmo realizará as devidas materializações e manutenções, de acordo com a operação identificada.

5.3 Inicialização

A comunicação entre a fonte de dados externa - no caso a DBLP - e a base de dados local é realizada após a inicialização do componente Monitor.

Ao inicializá-lo (Figura 11), consultas periódicas são realizadas na tabela *sameas_uri*, buscando por registros da tabela pessoa que estejam desatualizados, ou seja, que precisam de uma nova busca externa por sua URI equivalente. Ao identificar algum registro desatualizado, o Monitor aciona o módulo de resolução de entidades da DBLP, obtém o resultado da execução, e o atualiza com a nova URI encontrada.

```
> ./start.sh
Verificando se o processo ja esta sendo executado...
Aparentemente nao.

Carregando as configuracoes do banco...

Testando a conexao com o banco...
OK!

Inicializando...
PID: 17224

Fim.
```

Figura 11: Inicialização do Monitor

Para este estudo de caso, definimos uma periodicidade diária para a realização de buscas na base de dados externa. Com isso, caso nenhuma URI seja encontrada na primeira busca, outras são realizadas diariamente, até obter algum resultado.

No momento da execução inicial do Monitor, a tabela pessoa possuía 990 registros. Com a inicialização, cada um desses registros ganhou: um identificador *id_sameas*; um registro na tabela *sameas* com esse identificador; e um outro registro na tabela *sameas_uri*, com a coluna *updated* com o valor *false*, significando que a *uri* precisa ser consultada na Web.

O Monitor identificou os 990 registros pendentes e executou o módulo de resolução de entidades “dblp” para cada um deles. Dentre todas as execuções, 266 retornaram uma URI da DBLP como resultado e tiveram a ligação owl:sameAs correspondente materializada na base de dados local. Essas consultas e materializações foram realizadas em uma total de 14 minutos e 50 segundos. Os demais registros permaneceram pendentes para serem consultados novamente no dia seguinte.

5.4 Execução

Simulamos um caso de teste para cada tipo de modificação possível na tabela pessoa do RPA@PUC (monitorada pelo Monitor) e na base de dados da DBLP. Os passos de execução de cada um deles estão descritos nas seções a seguir.

5.4.1. Alterações na Base Local

- **Inserção de um professor que possui URI na DBLP**

1) Inserimos o professor ‘Alan Mathison Turing’ na tabela *pessoa*;
2) A operação disparou o gatilho associado à tabela:
    a. Inseriu um novo valor (sequencial) na tabela *id_sameas*;
    b. Inseriu esse *id_sameas*, juntamente ao *id_resolver* mapeado para a tabela *pessoa*, na tabela *sameas_uri*, mantendo a coluna *updated* com o valor padrão (*false*);
    c. Por fim, acrescentou o valor do novo *id_sameas* na coluna *id_sameas* do novo registro do professor;
3) O Monitor encontrou o registro pendente na tabela *sameas_uri*:
    a. Executou o módulo de resolução de entidades *dblp*;
    b. Obteve uma URI como resultado;
    c. Atualizou a coluna *uri* da tabela *sameas_uri* com o resultado encontrado, a coluna *updated* com o valor *true*, e a coluna *last_search* com a data da consulta.
4) A ligação foi materializada com sucesso.

- **Inserção de um aluno que não possui URI na DBLP**

1) Inserimos a aluna ‘Carla Gonçalves Ourofino’ na tabela *pessoa*;
2) A operação disparou o gatilho associado à tabela (ver caso anterior);
3) O Monitor encontrou o registro pendente na tabela *sameas_uri*:
    a. Executou o módulo de resolução de entidades *dblp*;
    b. Não obteve uma URI como resultado;
    c. Atualizou apenas a coluna *last_search* da tabela *sameas_uri* com o resultado encontrado, mantendo a coluna *updated* com o valor padrão *false*;
4) O passo 3 está sendo executado diariamente, e só realizará a materialização quando o Monitor obtiver algum resultado, ou seja, quando o aluno que foi inserido ganhar alguma URI equivalente na DBLP.

- **Atualização de um professor que tem ligação owl:sameAs**

1) Atualizamos o nome do professor ‘Alan Mathison Turing’ para ‘Alan Manuel Tavares’ na tabela *pessoa*;
2) A operação disparou o gatilho associado à tabela:
    a. Obteve o *id_sameas* do registro atualizado;
    b. Identificou que houve alteração no valor da coluna utilizado anteriormente na execução do módulo de resolução de entidades;
    c. Atualizou a coluna *updated* da tabela *sameas_uri* com o valor *false*, a coluna *uri* com o valor nulo e a coluna *last_search* com a data da consulta;
3) O Monitor encontrou o registro pendente na tabela *sameas_uri*:
    a. Executou o módulo de resolução de entidades *dblp*;
    b. Não obteve uma URI como resultado;
    c. Atualizou apenas a coluna *last_search* da tabela *sameas_uri* com o resultado encontrado, mantendo a coluna *updated* com o valor padrão *false*;
4) A ligação anteriormente existente foi “desmaterializada” com sucesso;
5) O passo 3 está sendo executado diariamente, e só realizará uma nova materialização quando o Monitor obtiver algum resultado, ou seja, quando houver alguma URI na DBLP equivalente ao professor do novo nome atribuído ao registro.

- **Remoção de um professor**

1) Removemos o registro do professor ‘Alan Manuel Tavares’ da tabela *pessoa*;
2) A operação disparou o gatilho associado à tabela:
    a. Obteve o *id_sameas* do registro removido;
    b. Removeu o registro da tabela *sameas* correspondente a esse *id_sameas*;
3) O SGBD preenche com o valor nulo as colunas *id*_sameas da tabela *pessoa* com valores correspondentes ao *id_sameas* removido (propriedade “on delete set null”);

4) O SGBD remove os registros da tabela *sameas_uri* correspondentes ao *id_sameas* removido (propriedade “on delete cascade”);
5) A ligação foi desmaterializada com sucesso.

5.4.2. Alterações na Base Externa

Mensalmente, todos os registros da tabela *pessoa* que tinham ligações materializadas tiveram a coluna *updated* da tabela *sameas_uri* atualizadas para *false*.

O Monitor encontrou esses registros como pendentes na tabela *sameas_uri* e executou o módulo de resolução de entidades para cada um deles.

Quando obteve alguma URI como resultado, atualizou a coluna *uri* para o valor obtido, a coluna *updated* para *true* e a coluna *last_search* para a data da consulta.

Quando não obteve nenhum resultado, atualizou a coluna *last_search* para a data da consulta, e a coluna *uri* para nulo, pois a materialização anterior deixou de ser válida. Como nos casos das alterações na base local, o Monitor encontra esses registros pendentes diariamente e só realizará novas materializações quando o módulo retornar algum resultado.

5.5 Publicação do Resultado

Com o Framework MsA executando em segundo plano para materializar as ligações owl:sameAs, realizamos a publicação dos dados em RDF, utilizando as URIs que forem sendo materializadas. Para isso, utilizamos o D2R Server – uma ferramenta para a publicação de bancos de dados relacionais non-RDF na Web, seguindo os princípios de Linked Data. O D2R Server permite que o publicador de dados defina um mapeamento entre um esquema relacional de banco de dados e vocabulários RDF. Baseado neste mapeamento, publica os dados na Web e permite consultas através do protocolo SPARQL (Cyganiak, Bizer, Garbers, Maresch, & Becker, 2012).

Em resumo, cada URI publicada pelo D2R Server, quando acessada, monta e executa uma consulta no banco de dados para obter as informações correspondentes. Isto é feito através da elaboração de uma regra relativa à

consulta no arquivo de mapeamento utilizado. Para a publicação de dados do RPA@PUC, descrevemos as regras de mapeamento de cada tabela. Dentre essas regras, implementamos a responsável por publicar as ligações owl:sameAs materializadas, realizada a partir de consultas na tabela *sameas_uri* do framework desenvolvido.

A Figura 12 apresenta a regra utilizada para publicar as ligações owl:sameAs da tabela pessoa do sistema RPA@PUC. Ela especifica a relação entre a tabela do usuário (tabela pessoa) e a tabela do Framework MsA (*sameas_uri*) que contém a URI equivalente a ser publicada como objeto da tripla.

```
map:pessoa_sameas a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:pessoa;
    d2rq:property owl:sameAs;
    d2rq:join "sameas_uri.id_sameas <= rpa.pessoa.id_sameas";
    d2rq:condition "sameas_uri.updated is true";
    d2rq:uriColumn "sameas_uri.uri";
    .
```

Figura 12: Mapeamento do D2R Server para publicação de ligações owl:sameAs

Vale ressaltar que o D2R Server foi utilizado como exemplo concreto, mas o modelo poderia ser utilizado com qualquer outra ferramenta de publicação de dados em formato RDF na Web.

A Figura 13 demonstra a solução utilizada neste estudo de caso: o uso do D2R Server para publicação dos dados, juntamente com o Framework MsA desenvolvido neste trabalho. Para publicar as ligações owl:sameAs junto aos demais dados da base, basta utilizar as URIs materializadas na tabela *sameas_uri* do framework.

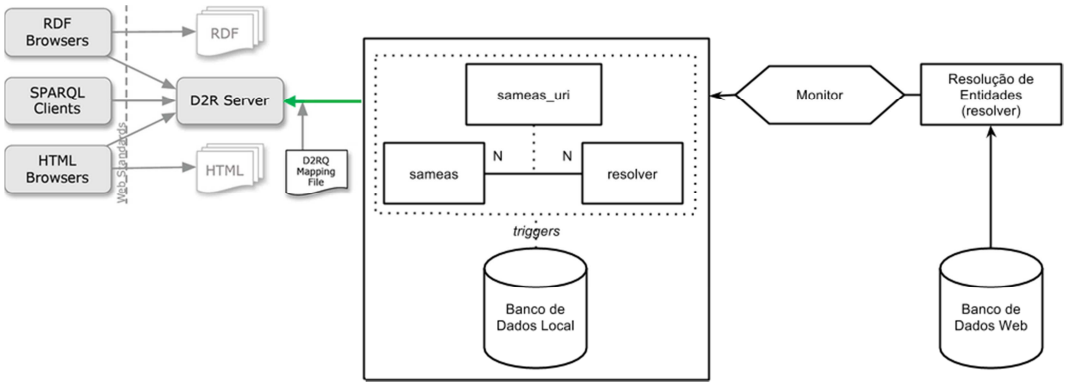

Figura 13: Uso conjunto do D2R Server com o Framework MsA