Ferb: um framework para casamento de esquemas

Sérgio Luis Sardi Mergen , Carlos Alberto Heuser

[1]Instituto de Informática – UFRGS

mergen@inf.ufrgs.br, heuser@inf.ufrgs.br

Resumo. *A tarefa de casamento de esquemas pode ser desempenhada de forma mais precisa se diversas técnicas de casamento forem usadas de forma combinada. Neste artigo é apresentado um framework de casamento cujo objetivo é permitir que técnicas de casamentos de esquemas sejam combinadas de forma expressiva. O artigo apresenta comparações com um framework de casamento bastante conhecido, salientando os pontos em que a abordagem proposta apresenta melhorias.*

1. Introdução

O problema de casamento de esquema já é bastante conhecido na literatura [Rahm and Bernstein, 2001], sendo que diversas técnicas de casamento (ou casadores) foram desenvolvidas com o intuito de encontrar correspondências entre elementos de dois esquemas[Melnik et al., 2002, Xu and Embley, 2003, Mergen and Heuser, 2004a]. Na maioria dos casos, o uso de um único casador não é suficiente para que a qualidade dos casamentos obtidos seja satisfatória, sendo que o uso combinado de diversos casadores é mais aconselhável. O emprego de mais de um casador para resolver um único problema de casamento pode ser caracterizado de duas formas: casadores híbridos que integram múltiplas técnicas de casamento em um único algoritmo e casadores compostos que combinam o resultado de técnicas de casamento executadas de forma independente.

Uma breve comparação entre estes casadores mostra que os híbridos geralmente são mais eficientes, uma vez que eles são escritos de forma a explorar da melhor forma a combinação das técnicas de casamento. Por outro lado, os casadores compostos oferecem uma solução mais flexível, pois trata-se de uma solução modularizada [Rahm and Bernstein, 2001] .

Em torno do problema que envolve casadores compostos, diversas abordagens foram propostas[Madhavan et al., 2003, Aumueller et al., 2005] . Uma delas em particular[Aumueller et al., 2005], apresenta um *framework* de casamento (**COMA** - COmbining Match Algorithms) que facilita com que as técnicas de casamento sejam combinadas. O mecanismo de execução de casadores adotado em *COMA* ocorre em iterações, onde casa iteração compreende 3 fases distintas: *Identificação*, que determina os elementos do esquema relevantes para o casamento, *Execução*, que aplica os casadores selecionados para obter valores de similaridade entre os elementos e *Combinação*, que agrega os valores individuais de similaridade para derivar um valor único que indica a correspondência entre os elementos. Cada iteração pode ter uma configuração diferente, e a saída de uma iteração serve como entrada para a iteração posterior.

Esta solução conta ainda com uma série de recursos bastante úteis para a tarefa de casamento, como suporte a diferentes formatos de esquema, inclusive ontologias. No entanto, a forma com que a combinação das similaridades ocorre é um tanto limitada,

uma vez que não se consegue determinar de forma expressiva como e em que situações os casadores devem ser combinados.

Visando atender a limitação previamente mencionada, neste artigo é apresentada uma nova abordagem de *framework* para o casamento de esquemas, chamada *Ferb*. Este *framework* conta com uma arquitetura que foi especialmente projetada para prover ao usuário maior poder de expressão na combinação das técnicas de casamento, de forma a permitir que tarefas de casamento seja desempenhadas de forma mais inteligente.

Este artigo está dividido da seguinte forma: A seção 2 aborda o *framework* proposto para o casamento de esquemas. Na seção 3 é apresentada a arquitetura dos módulos de casamento, que são os componentes responsáveis por executar as técnicas de casamento propriamente ditas. A seção 4 apresenta o comportamento do *framework* perante uma tarefa real de casamento. Considerações finais são apresentadas na seção 5

2. Framework Proposto

O *framework* tem como objetivo principal permitir que projetos de casamento de esquemas sejam criados e gerenciados, através da adição ou remoção de casadores e da configuração de suas propriedades. Ao todo, *Ferb* é composto por três componentes básicos: *wrappers*, *módulo agrupador* e *módulos de casamento*. A interação entre estes componentes está ilustrada na figura 1. Cada um destes componentes é descrito a seguir:

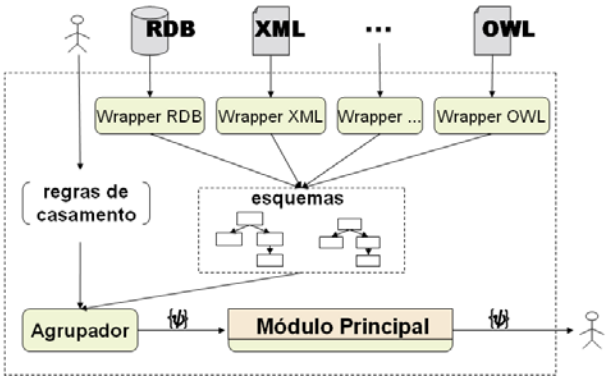

Figura 1: Interações entre os componentes do *framework*

Wrappers: O *framework* utiliza uma representação própria dos esquemas que se deseja comparar. Assim, são usados *wrappers* para transformar o esquema de entrada na representação aceita em *Ferb*. O formato de representação interno, ou esquema canônico, pode ser descrito conforme a definição 1.

Definição 1 *(Esquema Canônico) Considere que um esquema canônico E seja um conjunto de elementos $\{e_i\}$. Além disso, $parent(e_i)$ é uma função que retorna o elemento pai de e_i. Tem-se também que $E = C \cup A$, sendo que, dado um elemento $e_i \in E$, então:*

- $e_i \in C$, se $\exists\, e_j \in E$, de tal forma que $e_i = parent(e_j)$.

- $e_i \in A$, se $\nexists\, e_j \in E$, de tal forma que $e_i = parent(e_j)$.

De acordo com a definição acima, um esquema canônico é uma árvore (ou conjunto de árvores) composta por elementos, sendo que o conjunto de elementos complexos é representado por C e o conjunto de elementos atômicos é representado por A. Além disso, cada elemento é formado por demais características(tipo de dado, cardinalidade, etc) que não foram descritas neste artigo devido a falta de espaço.

Como a tarefa de casamento envolve dois esquemas, usa-se a notação e_o para indicar um elemento do esquema de origem E_o e e_d para indicar um elemento do esquema

destino E_d. Semanticamente os esquemas de origem e destino não possuem diferença alguma. Usa-se estes nomes apenas para salientar que trata-se de dois esquemas diferentes.

Módulo Agrupador: O módulo agrupador recebe os esquemas canônicos e forma pares de elementos $<e_o, e_d>$ de acordo com regras que indicam os tipos de elementos que podem ser casados uns com os outros. Estas regras se enquadram em 4 categorias, conforme descrito na tabela 1:

Regra	Descrição
<atômico, atômico>	Permite casamentos $<e_o, e_d>$ caso $(e_o \in A_o)$ e $(e_d \in A_d)$
<atômico, complexo>	Permite casamentos $<e_o, e_d>$ caso $(e_o \in A_o)$ e $(e_d \in C_d)$
<complexo, atômico>	Permite casamentos $<e_o, e_d>$ caso $(e_o \in C_o)$ e $(e_d \in A_d)$
<complexo, complexo>	Permite casamentos $<e_o, e_d>$ caso $(e_o \in C_o)$ e $(e_d \in C_d)$

Tabela 1: Regras para formação de pares

As regras são fornecidas pelo usuário em uma etapa anterior ao casamento. É importante ressaltar que mais de um regra podem ser adicionadas a um projeto de casamento. Caso as 4 regras sejam fornecidas, o conjunto inicial de casamentos equivale a $|E_o|X|E_d|$.

Usar estas regras para restringir casamentos pode ser útil, já que podem existir casos onde alguns tipos de casamentos são supérfluos. Por exemplo, em [Mergen and Heuser, 2004b] dados relacionais são transformados em XML usando para isso apenas casamentos entre os elementos atômicos.

Uma vez que as regras foram aplicadas, os pares resultantes são transformados em casamentos do tipo $\psi =< e_o, e_d, sim >$, onde sim é a similaridade entre os elementos e_o e e_d. Como nenhum casador foi processado ainda, a similaridade para todos os casamentos inicia com o valor zero. O resultado desta fase é um conjunto de casamentos $\{\psi\}$.

Módulo de casamento: Os casamentos formados pelo módulo agrupador são submetidos ao módulo de casamento principal. Todo o processamento referente a obtenção de valores de similaridade entre os pares casados é executado dentro deste módulo. Na próxima seção são dados mais detalhes sobre como a arquitetura de um módulo é formada, e como ocorre o processamento dentro de um módulo.

3. Arquitetura dos Módulos

O processamento dos casadores é embutido dentro de uma estrutura chamada de módulo. O fluxo de dados na entrada e saída dos módulos é um conjunto de casamentos $\{\psi\}$, sendo que a saída representa o resultado do processamento interno do casador sobre os casamentos fornecidos na entrada. Os módulos podem ser classificados em módulos simples e módulos compostos.

3.1. Módulos Simples

Um módulo simples é constituído por um algoritmo de casamento (casador), uma função de filtro e uma função de combinação de similaridade. O fluxo de execução dentro de um módulo simples começa com o algoritmo de casamento. O conjunto de casamentos de entrada é processado, sendo que, para cada ψ, o algoritmo de casamento é invocado de modo a fornecer um novo valor de similaridade. Após a fase de cômputo da nova similaridade, são executadas duas etapas, nesta ordem: i) combinação de similaridades e ii) filtro de casamentos. Os detalhes referentes a estas etapas são pormenorizados a seguir.

Combinação das similaridades: Para cada casamento ψ, é realizada a combinação de dois valores de similaridades, sendo que o primeiro valor corresponde a similaridade do casamento antes da execução do casador, enquanto o segundo valor corresponde a similaridade computada pelo casador. Para diferenciar ambos, sim_o é usado para referenciar o primeiro e s_c para referenciar o segundo.

Para a combinação, pode-se usar 4 métricas distintas: a métrica max retorna o maior valor dentre s_o e s_c. Como esta métrica presume que o maior valor realmente indique o valor de similaridade correto, esta técnica é considerada otimista. Já a métrica min, aceita o menor valor como correto, e é considerada pessimista. A métrica $mArit$ considera que ambos scores possuem igual relevância, sendo o resultado final a média aritmética entre s_o e s_c. A última métrica $mPond$ é uma extensão de $mArit$ que permite pesos diferentes para cada valor. Como resultado na combinação, os casamentos são atualizados com o valor de similaridade combinado.

Além disso, é possível restringir a combinação, de modo que esta operação só possa ser efetuada se o valor final de similaridade for superior a um ponto de corte. Este ponto de corte pode ser tanto um valor previamente estabelecido, bem como o próprio valor original de similaridade s_o. Caso o valor final seja inferior, o valor combinado é descartado, prevalecendo o valor da similaridade original s_o.

Filtro: A última etapa desempenhada dentro de um módulo simples envolve a eliminação de casamentos incorretos. Esta eliminação segue os mesmo moldes do *framework COMA*, sendo permitidas 3 formas distintas de filtro. Para facilitar a compreensão dos filtros, usa-se a definição abaixo:

Definição 2 *(Lista ordenada de casamentos) Dado um elemento e_χ, considere que $<\psi_1, \psi_2, \psi_i, ..., \psi_n>$ seja uma lista ordenada de casamentos, sendo que $\psi_i.e_o = e_\chi$ ou $\psi_i.e_d = e_\chi$. Ademais, dados os casamentos ψ_i e ψ_j, então $i > j$ caso $\psi_i.sim > \psi_j.sim$.*

Dito isso, para cada elemento e_χ, pode-se computar o filtro utilizando os seguintes métodos:

MaxN(N) Elimina todos os casamentos ψ_i onde $i < N$.
MaxDelta(Δ) Elimina todos os casamentos ψ_i onde $\psi_1.sim - \psi_i.sim > \Delta(delta)$.
Threshold(θ) Elimina todos os casamentos ψ_i onde $\psi_i.sim < \theta$.

Além disso, pode-se efetuar o filtro levando em consideração os elementos de um único esquema, ao invés dos elementos de ambos esquemas casados. Assim, a lista ordenada de casamentos passaria a conter apenas os elementos ψ onde $\psi.e_o = e_\chi$, ou apenas os elementos ψ onde $\psi.e_d = e_\chi$. Em alguns casos, pode existir diferença se o filtro é computado utilizando apenas os elementos do esquema origem, apenas os elementos do esquema destino, ou ambos, como é demonstrado em [Aumueller et al., 2005].

A arquitetura apresentada neste artigo permite que a fase de filtro seja aplicada após a execução de cada casador, acarretando na possível eliminação antecipada de casamentos incorretos. Esta eliminação antecipada serve como forma de otimizar o processamento dos casadores posteriores, visto que os mesmos não precisam computar valores de similaridade para casamentos incorretos.

3.2. Módulos Compostos

Os módulos compostos diferenciam-se dos módulos simples por não conterem um casador associado. Ao invés disto, um módulo composto é formado por uma combinação de módulos, que pode incluir tanto módulos simples como outros módulos compostos. Cabe salientar que o módulo principal do sistema trata-se de um módulo composto.

Dentro de um módulo composto, o fluxo de processamento pode ser serial ou paralelo. No fluxo serial, os módulos internos são processados em uma seqüência ordenada, o que se assemelha a forma de processamento da abordagem *COMA*. A inovação consiste na definição de fluxo paralelo, onde os módulos internos são processados de forma concomitante.

Assim como os módulos simples, os módulos compostos possuem uma função de filtro e uma função de combinação de scores, que são executados após o término da execução de todos os seus módulos internos. A combinação das similaridades do módulo composto serial segue o mesmo raciocínio do módulo simples, sendo que a similaridade s_o equivale a similaridade usada na entrada do módulo composto e a similaridade s_c equivale a similaridade computada pelo último módulo interno.

Já no módulo composto com fluxo paralelo, não existe um único valor de similaridade computada s_c, já que o módulo é composto por uma série de módulos internos que executam em paralelo. Com isso, a combinação dos resultados ocorre de forma distinta, sendo que o resultado final corresponde a combinação das similaridades resultantes de cada módulo interno, ao invés da combinação da similaridade original s_o com um único valor de similaridade s_c.

4. Estudo de Caso

Para exemplificar o uso do *framework Ferb*, nesta seção é apresentado um projeto de casamento cujo objetivo é casar dois esquemas relacionais. Para tanto, dispõe-se de três casadores distintos. Um deles, chamado s, analisa a estrutura dos elementos para inferir um valor de similaridade. O casador e utiliza uma métrica lingüística para atribuir um valor de similaridade para cada par de elementos. Já o casador t utiliza um dicionário de palavras para verificar se os nomes dos elementos possuem alguma relação de parentesco.

Os três casadores s, e e t estão respectivamente associados aos módulos simples $MS1$, $MS2$ e $MS3$. A figura 2 permite analisar como estes módulos estão dispostos. Observa-se o uso de dois módulos compostos($MC1$ e $MC2$). O módulo $MC1$ é composto pelos módulos $MS1$ e $MC2$, ligados em série. Já o módulo $MC2$ é composto pelos módulos simples $MS2$ e $MS3$, ligados em paralelo.

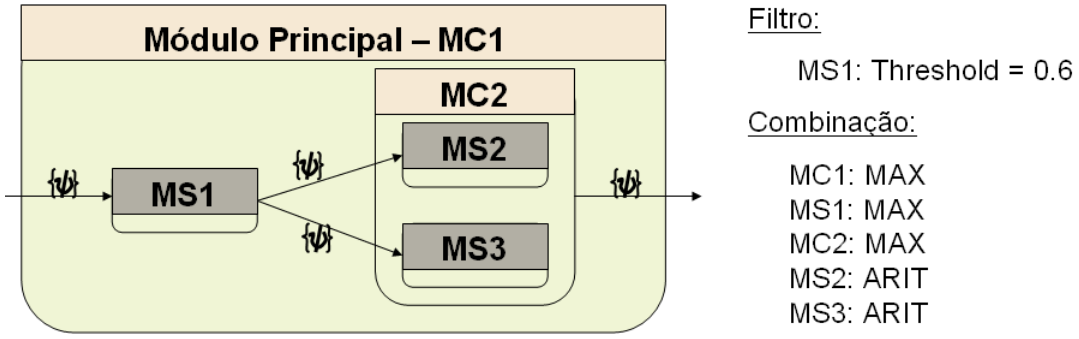

Figura 2: Configuração interna do módulo principal

A figura também apresenta informações adicionais sobre os módulos, mostrando como as funções de combinação e filtro de cada um foram configuradas. Com relação as regras de casamento, supõe-se que elas tenham sido configuradas de modo a permitir que apenas os elementos atômicos sejam casados.

Como o fluxo dentro de $MC1$ é serial, a entrada de $MS1$ é a mesma de $MC1$. Já no módulo $MC2$ o fluxo é paralelo, o que significa que tanto $MS2$ quanto $MS3$ recebem

uma cópia da saída de $MS1$. De forma resumida, para cada casamento, o resultado final de similaridade será equivalente a média aritmética entre a similaridade estrutural e o valor máximo dentre a similaridade lingüística atribuída por e e a similaridade atribuída por t. Além disso, caso a similaridade estrutural seja inferior a 0.6, o casamento é desconsiderado.

5. Conclusão

O *framework* apresentado neste artigo já está parcialmente desenvolvido, na linguagem Java. No seu estado atual, *Ferb* conta com dois *wrappers*, responsáveis por criar esquemas canônicos a partir de um esquema relacional e de um esquema DTD. Outra característica importante do *framework* é o seu caráter extensível, o que permite que novos casadores sejam adicionadas ao sistema na medida que eles tornarem-se disponíveis. Para isto, é necessário que os casadores sejam escritos em java, e que obedeçam a uma estrutura pré-definida, de forma que o *framework* saiba como processar o casador.

Além disso, foi desenvolvido um sistema que permite com que o usuário interaja com o *framework*. Este sistema já foi usado com sucesso no casamento entre esquemas relacionais e esquemas XML (DTDs). Em comparação com a abordagem manual de combinação dos casadores, o sistema traz como benefícios a agilidade na configuração de um projeto de casamento de esquemas, além da possibilidade de analisar os resultados obtidos para posterior refinamento das configurações.

Apesar do *framework COMA* possuir muito mais recursos, a forma como os casadores é combinada em *Ferb* é inédita, pois possibilita que o usuário defina com mais liberdade como os casadores interagem entre si. Todavia, existem casos que ainda não são atendido pelo *framework*. Por exemplo, considerando os casadores demonstrados anteriormente, não seria possível construir um projeto onde a similaridade final dos elementos fosse dada pelo valor máximo entre a média da combinação de pares de casadores, como em $\max((\frac{(s+t)}{2}), (\frac{(s+e)}{2}), (\frac{(e+t)}{2}))$. O estudo das modificações que a arquitetura deve sofrer para dar suporte a este tipo de equação é um tema pertinente para trabalhos futuros.

Referências

Aumueller, D., Do, H. H., Massmann, S., and Rahm, E. (2005). Schema and ontology matching with coma++. In *SIGMOD Conference*, pages 906–908.

Madhavan, J., Bernstein, P., Chen, K., Halevy, A., and Shenoy, P. (2003). Corpus-based schema matching. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'2003)*, Acapulco.

Melnik, S., Garcia-Molina, H., and Rahm, E. (2002). Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings of the 18th ICDE Conference*.

Mergen, S. and Heuser, C. A. (2004a). Matching of xml schemas and relational schemas. In *XIX Simpósio Brasileiro de Banco de Dados (SBBD'2004)*, Brasilia.

Mergen, S. and Heuser, C. A. (2004b). A software for exchanging data between xml and databases. XIX Simpósio Brasileiro de Banco de Dados (SBBD'2004). Brasilia.

Rahm, E. and Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *VLDB Journal: Very Large Data Bases*, 10(4):334–350.

Xu, L. and Embley, D. (2003). Discovering direct and indirect matches for schema elements. In *Eighth International Conference on Database Systems for Advanced Applications (DASFAA'03)*, Kyoto.