VBA: Uma Ferramenta para Criação de Visões de Objeto de Dados Relacionais

Fernando Lemos, Lineu Santos, Valdiana Araújo, Wamberg Oliveira, Vânia Vidal

Departamento de Computação – Universidade Federal do Ceará (UFC)

e-mail: {fernandocl, lineu, valdiana, wamberg, vvidal}@lia.ufc.br

Resumo. *Visões são usadas para publicar dados de uma base de dados em uma nova estrutura ou esquema. A definição de visões de objeto sobre dados relacionais permite que esses dados sejam disponibilizados em uma estrutura orientada a objetos, sem que seja necessário mudar o esquema da base de dados. Assim, as visões de objeto constituem interfaces através das quais aplicações orientadas a objeto podem consultar e atualizar a base relacional de forma transparente. Neste trabalho apresentamos VBA (View-By-Assertions), uma ferramenta para facilitar a criação de visões de objeto sobre bases de dados relacionais. No nosso enfoque, a visão de objeto é definida por um conjunto de Assertivas de Correspondência, as quais especificam formalmente o mapeamento entre a base de dados e esquema da visão. A partir das assertivas de correspondência da visão, VBA gera automaticamente: (i) a consulta SQL, que sintetiza os objetos da visão a partir dos dados da base relacional, e (ii) os tradutores, que transformam atualizações sobre a visão em atualizações sobre a base relacional.*

1. Introdução

Na arquitetura de três-níveis de esquemas, as visões constituem as interfaces através das quais os usuários consultam e atualizam a base de dados. Além das vantagens já conhecidas das visões relacionais, as visões de objeto [8,9] permitem a coexistência e o compartilhamento de dados armazenados em base de dados relacional com diversas aplicações usando tecnologia de objeto. Assim, visões de objeto definidas sobre dados relacionais provêem uma técnica poderosa para se impor visões lógicas, ricamente estruturadas, sobre bases de dados já existentes. Dessa forma, dados relacionais podem ser disponibilizados através de um esquema orientado a objetos, sem que seja necessário mudar o esquema da base de dados. Outra vantagem é que os dados resultantes de uma consulta a uma visão de objeto podem ser automaticamente publicados como uma visão XML [12] ou um documento XML dinâmico [2,8], permitindo a integração de aplicações *Web* com base de dados relacionais.

Como as visões de objetos são apenas interfaces, temos que consultas e atualizações especificadas sobre as visões devem ser traduzidas em consultas e atualizações sobre a base de dados. Uma definição de visão de objeto consiste de dois mapeadores: o *mapeador de instâncias* e o *mapeador de atualizações*. O *mapeador de instâncias* consiste de uma consulta SQL que mapeia um estado da base no estado da visão correspondente. O *mapeador de atualizações* especifica como atualizações sobre a visão são traduzidas em uma seqüência de atualizações sobre a base de dados. O mapeador de atualizações consiste de um conjunto de tradutores, um para cada uma das operações de atualização que forem permitidas sobre a visão [3,7]. Um tradutor é uma função que recebe como entrada um pedido de atualização e gera uma tradução para esta atualização. Os tradutores podem ser implementados através de *triggers* no SGBD.

Definir uma consulta que realize o mapeamento de tuplas de tabelas relacionais em objetos com estrutura complexa é tarefa que exige conhecimentos avançados de SQL:1999 [4]. Da mesma forma, gerar tradutores para mapear objetos de estrutura complexa em tuplas de tabelas relacionais não é tarefa fácil [1,6]. No caso de modificações no esquema da base de dados, a consulta SQL e os tradutores precisam ser redefinidos. Atualmente, a criação e a manutenção das visões de objeto são feitas manualmente, e não temos conhecimento de nenhum trabalho ou ferramenta que auxilie na geração de visões de objeto.

Neste trabalho apresentamos VBA (View-By-Assertions), uma ferramenta para facilitar a criação de visões de objeto sobre bases de dados relacionais. Na sua forma atual, VBA só lida com visões preservadoras de objetos, ou seja, cada objeto da visão é semanticamente equivalente a uma tupla de alguma tabela da base relacional (tabela pivô) [11]. O restante deste artigo é dividido como segue. Na Seção 2 apresentamos o processo de geração de visões. Na Seção 3, apresentamos um exemplo de geração de uma visão com VBA. Em seguida, na Seção 4, apresentamos nossas conclusões e discutimos alguns melhoramentos que estão sendo desenvolvidos em VBA.

2. Processo de geração de uma visão de objeto com VBA

O processo de geração de uma visão de objeto com VBA consiste de quatro passos:

(i) Primeiro, o usuário define o Esquema da Visão, o qual consiste do tipo dos objetos da visão e demais tipos usados na definição do tipo da visão;

(ii) Em seguida, o usuário define as Assertivas de Correspondência (ACs) que especificam formalmente os relacionamentos entre o esquema da base de dados e o esquema da visão. Em [11] são definidas as condições formais sobre as quais um conjunto de ACs especifica completamente uma visão de objetos em termos de um esquema relacional e, neste caso, é mostrado que o mapeamento definido pelas ACs pode ser traduzido diretamente em uma consulta SQL. É importante salientar que outros formalismos de mapeamentos propostos ou são ambíguos [5] ou complexos [13], os quais não podem ser definidos graficamente. Como mostrado em [11], VBA lida com o problema da heterogeneidade semântica [10], permitindo que mapeamentos complexos possam ser especificados de forma simples;

(iii) No terceiro passo, VBA gera automaticamente a consulta SQL, que mapeia um estado da base de dados no estado da visão correspondente. Em [11] é apresentado um algoritmo que gera a consulta SQL da visão baseado nas ACs da visão;

(iv) Por último, VBA gera automaticamente os *triggers* para as operações de atualização que são permitidas sobre a visão. Em [3,7] são apresentados os algoritmos de geração dos *triggers* para os tipos básicos de operações de atualização de visões de objetos: inserção e remoção de objetos, modificação dos valores de atributos monovalorados e inserção e remoção de objetos em coleções aninhadas. Nosso formalismo de mapeamento permite identificar as situações em que as ambigüidades podem ser resolvidas em tempo de definição da visão [3,7] e assim definir a seqüência de atualizações na base de dados necessária para realizar uma atualização solicitada sobre a visão. Quando o usuário requisita uma atualização em uma visão, o tradutor correspondente realiza a tradução em atualizações sobre a base de dados, sem que seja necessário qualquer diálogo adicional com o usuário para eliminar possíveis ambigüidades [1,6].

3. Exemplo de criação de uma visão de objeto com VBA

Nessa seção, como estudo de caso, considere o esquema relacional ER mostrado na Figura 1. Suponha que o usuário deseje definir sobre ER as visões de objeto Pedidos_v e Clientes_v, cujos objetos são do tipo Tcliente e Tpedido, respectivamente, mostrados na Figura 2.

Para definir o esquema da visão no VBA, o usuário deve, através de uma interface gráfica, definir o nome e os atributos do tipo da visão. Para definir um atributo, o usuário deve definir o nome, o tipo do atributo, sua cardinalidade e se o mesmo é de valor ou de referência. No caso de atributo de tipo estruturado, um novo tipo deve ser definido (nome e atributos). No caso de atributo de referência, deve-se definir seu escopo (visão de objeto referenciada). Por exemplo, na visão Pedidos_v, o escopo do atributo cliente_ref do tipo Tcliente é a visão Clientes_v. Figura 3 mostra os *scripts* de criação dos tipos das visões Pedidos_v e Clientes_v gerados pela ferramenta.

Após definir o tipo dos objetos da visão o usuário seleciona, de uma lista de tabelas e visões relacionais da base de dados, a *tabela pivô* ou *visão pivô*, de forma que existe um mapeamento 1-1 entre tuplas da tabela/visão pivô e os objetos da visão. No nosso estudo de caso, suponha que para a visão Pedidos_v, a tabela Pedidos_rel é selecionada como tabela pivô. Após a escolha da tabela pivô, o usuário deve especificar as ACs que definem os relacionamentos entre atributos do tipo da visão com atributos/caminhos da tabela pivô. Para isso, o VBA exibe uma tela contendo, em formato de árvore de diretório, o esquema da visão e o esquema relacional, conforme mostrado na Figura 4. Observe que o tipo de cada atributo é especificado entre parênteses, e no esquema relacional, além dos atributos da tabela, são mostradas também as chaves estrangeiras e inversas de chave estrangeira, juntamente com a

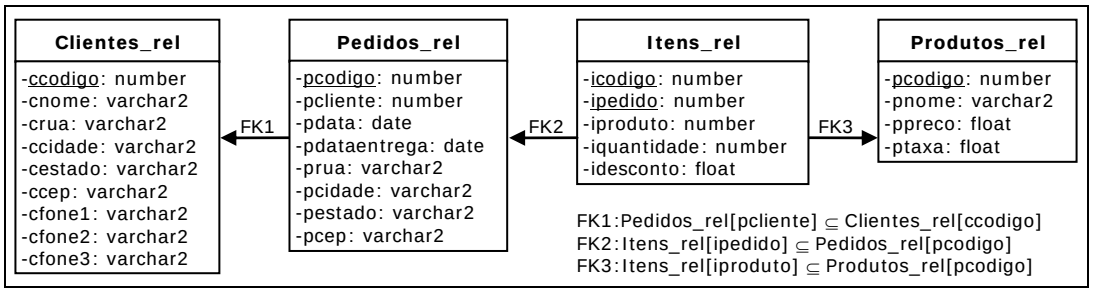

Figura 1. Esquema Relacional ER

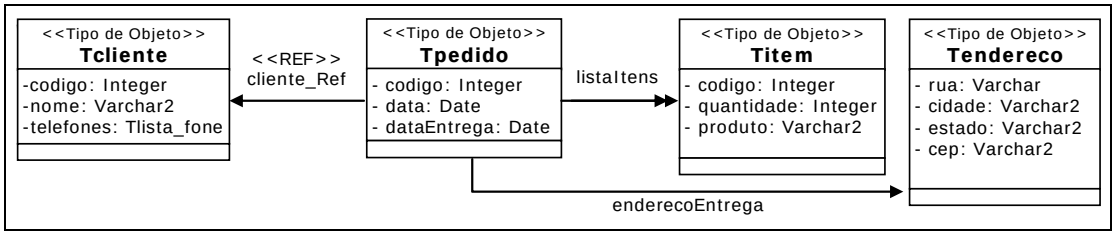

Figura 2. Tipos das Visões Pedidos_v **e** Clientes_v

```
CREATE TYPE Tlista_fone AS VARRAY(10) OF VARCHAR(15);
CREATE TYPE Titem AS OBJECT(
  codigo INTEGER,
  quantidade INTEGER,
  produto VARCHAR2(30));
CREATE TYPE Tlista_item AS TABLE OF Titem;
CREATE TYPE Tendereco AS OBJECT(
  rua VARCHAR2(30),
  cidade VARCHAR2(15),
  estado VARCHAR2(2),
  cep VARCHAR2(8) );
CREATE TYPE Tcliente AS OBJECT(
  codigo INTEGER,
  nome VARCHAR2(50),
  telefones Tlista_fone);
CREATE TYPE Tpedido AS OBJECT(
  codigo INTEGER,
  data DATE,
  dataEntrega DATE,
  enderecoEntrega Tendereco,
  cliente_ref REF Tcliente,
  listaItens Tlista_item);
```

Figura 3. Definição dos tipos das visões Pedidos_v **e** Clientes_v

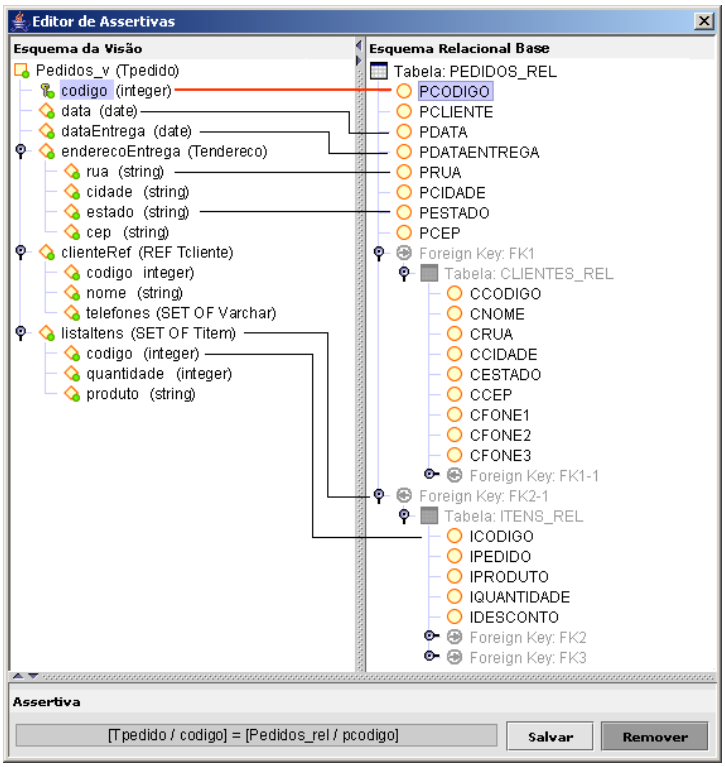

Figura 4. Editor de Assertivas de Correspondência do VBA

estrutura das tabelas relacionadas. Deste modo, o usuário pode definir caminhos que naveguem por essas chaves estrangeiras para as tabelas relacionadas. Para definir uma AC, o usuário relaciona graficamente o atributo da estrutura do esquema da visão com um atributo ou caminho de uma tabela base como indicado na Figura 4.

O processo de definição ACs com VBA é *top down* e consiste de dois passos: (i) Primeiro o usuário define as ACs para os atributos do tipo da visão; (ii) em seguida, define, de forma recursiva, as ACs para os atributos dos tipos estruturados. Assim, na definição das ACs de Pedidos_v, o usuário primeiro define as correspondências para os atributos de Tpedido. Assim:

- Para definir a AC do atributo codigo, o usuário seleciona codigo no esquema da visão e pcodigo no esquema da base de dados. Ao clicar no botão Salvar, VBA mostra a assertiva gerada Ψ_1: [Tpedido/ codigo] ≡ [Pedidos_rel/ pcodigo];
- Para definir a AC do atributo listaItens, o usuário seleciona listaItens no esquema da visão e fk_2^{-1} no esquema da base de dados. A assertiva gerada é Ψ_6:[Tpedido / listaItens] ≡ [Pedidos_rel / fk_2^{-1}];
- Para definir a AC do atributo enderecoEntrega, o qual não tem correspondência direta com nenhum atributo de Pedidos_v, o usuário seleciona o atributo enderecoEntrega, marca a opção Sem Correspondência, e salva. A assertiva gerada é Ψ_4:[Tpedido / enderecoEntrega] ≡ [Pedidos_rel / NULL].

Em seguida, o usuário deve definir as ACs para os atributos dos tipos estruturados TItem, Tendereco e Tcliente. É importante notar que como o atributo cliente_ref é de referência para a visão Clientes_v do tipo Tcliente, assim, caso as assertivas de Clientes_v já tenham sido definidas, temos que as ACs de Tcliente já estão definidas, e serão incorporadas às ACs de Pedidos_v. Figura 5 mostra as ACs entre o tipo da visão Pedidos_v e Clientes_v e a base relacional.

ACs de T_{pedido} & Pedidos_rel
ψ_1: [T_{pedido}/codigo] ≡ [Pedidos_rel/pcodigo]
ψ_2: [T_{pedido}/data] ≡ [Pedidos_rel/pdata]
ψ_3: [T_{pedido}/dataEntrega] ≡ [Pedidos_rel/pdataEntrega]
ψ_4: [T_{pedido}/enderecoEntrega] ≡ [Pedidos_rel/NULL]
ψ_5: [T_{pedido}/cliente_ref] ≡ [Pedidos_rel/fk_1]
ψ_6: [T_{pedido}/listaItens] ≡ [Pedidos_rel/fk_2^{-1}]
ACs de T_{item} & Itens_rel
ψ_{11}: [T_{item}/codigo] ≡ [Itens_rel/icodigo]
ψ_{12}: [T_{item}/quantidade] ≡ [Itens_rel/quantidade]
ψ_{13}: [T_{item}/produto] ≡ [Itens_rel/fk_3.pnome]

ACs de $T_{endereco}$ & Pedidos_rel
ψ_7: [$T_{endereco}$/rua] ≡ [Pedidos_rel/prua]
ψ_8: [$T_{endereco}$/cidade] ≡ [Pedidos_rel/pcidade]
ψ_9: [$T_{endereco}$/estado] ≡ [Pedidos_rel/pestado]
ψ_{10}: [$T_{endereco}$/cep] ≡ [Pedidos_rel/pcep]
ACs de $T_{cliente}$ & Clientes_rel
ψ_{14}: [$T_{cliente}$/codigo] ≡ [Clientes_rel/ccodigo]
ψ_{15}: [$T_{cliente}$/nome] ≡ [Clientes_rel/cnome]
ψ_{16}: [$T_{cliente}$/telefone] ≡ [Clientes_rel,{cfone1, cfone2, cfone3}]

Figura 5. Assertivas de Correspondência de Pedidos_v e Clientes_v

Após definir as ACs da visão, VBA gera automaticamente a definição SQL e os tradutores de atualização da visão. Figura 6 mostra a consulta SQL que define a visão Pedidos_v. Figura 6 mostra também as ACs que deram origem a cada pedaço do código SQL gerado. Para a visão Pedidos_v são gerados os tradutores (*triggers*) para os seguintes tipos de atualização: inserção, remoção e modificação em Pedidos_v e inserção e remoção na coleção aninhada listaItens. Figura 7 mostra o *trigger* para a operação de inserção na coleção aninhada listaItens da visão Pedidos_v.

É importante notar que a criação das visões Pedidos_v e Clientes_v permite o

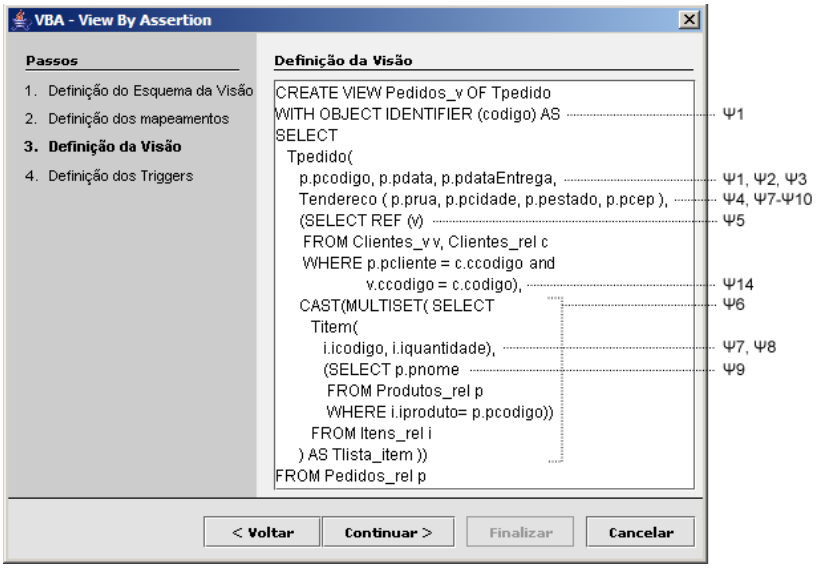

Figura 6. Consulta SQL da visão Pedidos_v

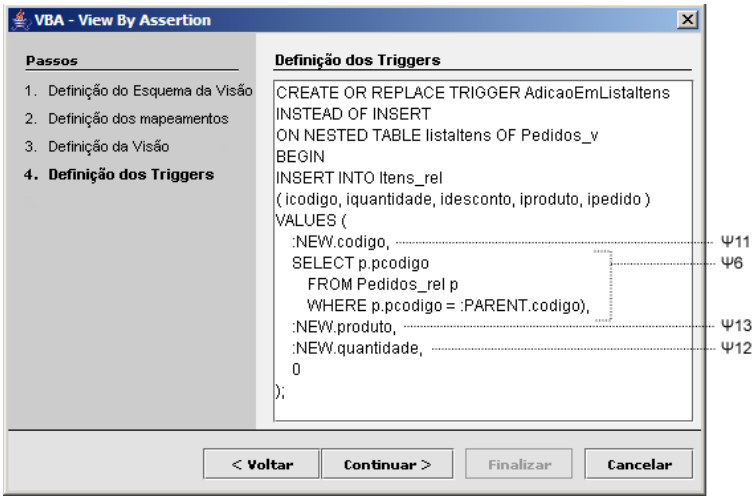

Figura 7. *Trigger* para a operação de inserção na coleção aninhada listaItens

desenvolvimento de aplicações orientadas a objeto que manipulem as visões como se fossem tabelas de objeto. Dessa forma, os dados armazenados na estrutura do esquema relacional ER são disponibilizados na estrutura dos tipos das visões sem que seja necessário mudar o esquema da base de dados.

4. Conclusão

Neste trabalho apresentamos VBA, uma ferramenta para facilitar a tarefa de criação de visões de objeto sob dados relacionais e seus *triggers* de manutenção. A novidade do enfoque proposto é que as visões de objeto são definidas através de ACs que especificam formalmente o relacionamento entre o esquema da visão e o esquema da base de dados. O formalismo das ACs permite especificar várias formas de correspondência, inclusive casos onde os esquemas possuem estruturas diferentes. A ferramenta oferece uma interface gráfica para apoiar a criação do tipo da visão e a edição de suas ACs, e gera automaticamente a consulta SQL, que realiza o mapeamento definido pelas assertivas da visão, e os *triggers* de manutenção da visão.

A ferramenta também suporta a manutenção das visões de objeto. Baseada nas ACs, VBA identifica automaticamente as visões de objeto afetadas pelas modificações no esquema da base de dados. Para cada visão afetada, deve-se redefinir as assertivas de acordo com o novo esquema da base de dados; e, baseado nas novas ACs, a ferramenta gera a nova consulta SQL e os novos tradutores de atualização da visão.

Futuramente, estenderemos a ferramenta de forma a suportar novos tipos de visão (união, interseção e diferença), e também permitir a criação de visões sobre base de dados objeto-relacional.

Referências

1. Barsalou, T., Siambela, N., Keller, A.M., Wiederhold, G., *Updating relational databases through object-based views*, In: Proceedings of the 1991 ACM SIGMOD, p.248-257, May 1991, Denver, US
2. Ceri, S., et al., *Designing Data-intensive Web Applications*, Morgan Kaufmann, Dec 2002.
3. Costa, J.P., *Atualização de Bancos de Dados Objeto Relacionais Através de Visões de Objetos*, Dissertação de Mestrado, Universidade Federal do Ceará, 2002.
4. Eisenberg, A., Melton, J., *SQL: 1999, formerly known as SQL3*, In: ACM SIGMOD Record, v.28 n.1, p.131-138, Mar 1999.
5. Hernández, M. A., Miller, R. J., Haas, L. M., *Clio: A Semi-Automatic Tool For Schema Mapping*, SIGMOD Conference 2001.
6. Keller, A.M., *The role of semantics in translating view updates*, IEEE Computer, vol.19, pp. 63-73, Jan 1986.
7. Oliveira, W.G.C., *Atualização de Bancos de Dados Objeto Relacionais Através de Visões XML*, Dissertação de Mestrado, Universidade Federal do Ceará, 2004.
8. Oracle Corporation. Disponível em: http://technet.oracle.com. (Acessado em 30 de junho 2005).
9. Takahashi, T., Keller, A. M., *Implementation of Object View Query on a Relational Database*, In: Data and Knowledge Systems for Manufacturing and Engineering, May 1994.
10. Vidal, V.M.P., Lóscio, B.F., *Solving the Problem of Semantic Heterogeneity in Defining Mediator Update Translators*, In: Proc. of 18th Intern. Conf. on Conceptual Modeling, p.293-308, France.
11. Vidal, V.M.P., Santos, L.A.L., Araújo, V.S., Lemos, F.C., *Geração Automática de Visões de Objeto de Dados Relacionais*, In: Simpósio Brasileiro de Banco de Dados, outubro 2005.
12. Vidal, V.M.P., Santos, L.A.L., Oliveira, W.G.C., Araújo, V.S., Lemos, F.C., Lima, D.R.C., *XML Publisher: Um Framework para Publicação de Dados Armazenados em Banco de Dados Relacional ou Objeto Relacional como XML*, In: I Sessão de Demos, SBBD, 2004, p. 07-12, Brasília.
13. Yu, C., Popa, L., *Constraint-Based XML Query Rewriting for Data Integration*, In: SIGMOD Records, pages 371–382, 2004.