Conceito de Bancos de Dados Orientados à Objetos

Sistemas de banco de dados são ferramentas estratégicas para muitas organizações espalhadas pelo mundo. Nestes sistemas, o armazenamento de dados é orientado para satisfazer as necessidades de persistência e distribuição de informações que são manipuladas por programas do usuário.

A evolução recente da tecnologia de banco de dados está intimamente ligada com a história dos bancos de dados Relacionais. O modelo de dados Relacional é simples, eficiente e formal, o que permitiu a sua disseminação por diferentes fornecedores e a sua contínua aplicação trinta anos após o seu surgimento. Em um banco de dados Relacional, os dados são armazenados em tabelas, as quais internamente definem colunas associadas a tipos de dados simples (strings, inteiros, booleanos, entre outros).

O modelo Relacional é muito bem sucedido: mesmo usuários finais de computadores conhecem os produtos e fornecedores principais disponíveis no mercado, tais como Oracle, Sybase, Microsoft e IBM, e mesmo a linguagem de consulta SQL é conhecida por muitos. Entretanto, o modelo Relacional começou a ser questionado a partir do surgimento do paradigma de Orientação a Objetos (OO) e da crescente demanda dos usuários para utilizar sistemas de banco de dados para armazenar dados “não-convencionais”.

O modelo OO está em um nível de abstração superior, mais próximo do domínio da aplicação: enquanto que no modelo Relacional o desenvolvedor é forçado a capturar a realidade da aplicação sendo construída através de tabelas, onde o relacionamento e integridade das entidades é feito através de valores, a abordagem dos BDOO está no mesmo nível de abstração fornecido pelas linguagens de programação OO.

Como os modelos de dados usados tanto no programa quanto no banco de dados são consistentes, a adoção de BDOO na construção de sistemas OO constitui a situação ideal, visto que não há necessidade de transformar o modelo de objetos do programa para um modelo que diz respeito somente ao gerenciador de banco de dados.

Enquanto aplicações convencionais possuem uniformidade e estruturação dos dados, permitindo aplicações orientadas a registros e com campos atômicos (isto é, que estejam na Primeira Forma Normal), as aplicações ditas “não-convencionais” armazenam dados complexos tais como áudio, imagens e vídeo, onde a diversidade na estruturação da informação não segue os critérios definidos pelas Formas Normais do modelo Relacional.

Os modelos de dados com objetos complexos também são caracterizados pela grande quantidade de associações muitos-para-muitos (N:N) que, no modelo Relacional, levam a um grande número de junções para acessar os dados das tabelas de interseção.

O objetivo deste artigo é apresentar uma introdução aos bancos de dados Orientados a Objetos (BDOO). Inicialmente será apresentado o histórico da área, que serve de motivação para em seguida serem apresentados os elementos básicos que constituem os produtos comerciais disponíveis.

É importante observar que, embora o tema seja amplo, podendo estudar os produtos segundo diferentes pontos de vista, o objetivo principal aqui é restrito à apresentação da tecnologia de BDOO para o desenvolvimento de aplicações. Assim, embora o exemplo seja apresentado em um produto específico - o CA-Jasmine - os conceitos usados são válidos para a grande maioria dos produtos atuais.

Histórico

Antes de constituírem os produtos comerciais que estão disponíveis atualmente, a evolução do modelo de dados OO passou pela constatação das limitações dos modelos de banco de dados existentes (notadamente, o modelo Relacional), o que culminou em um conjunto de propostas que definem extensões ou alternativas completamente novas.

A discussão no tema está originada na divulgação, por parte de importantes grupos de pesquisa na área, de documentos que atualmente são conhecidos como Manifestos para bancos de dados avançados. Em 1989, Atkinson et al foram responsáveis pela divulgação do “The Object-Oriented Database System Manifesto”, documento que que foi motivado pela grande diversidade de produtos comerciais e experimentos acadêmicos em banco de dados que utilizavam a expressão Orientação a Objetos na sua denominação.

De fato, o final da década de 1980 foi marcado pela grande diversidade na terminologia de Orientação a Objetos, o que ocorria simultaneamente com os debates acerca da viabilidade deste novo paradigma no desenvolvimento de software. No contexto de banco de dados, o cenário era marcado pela falta de um modelo formal único que norteasse as diferentes implementações de BDOO. Assim, a grande maioria das soluções eram experimentais, possuindo diferenças profundas no que dizia respeito à conformidade de conceitos da Orientação a Objetos. O Manifesto de Atkinson foi importante por determinar um elenco de características obrigatórias para que um sistema pudesse ser considerado um BDOO.

Deste modo, um BDOO deveria, entre outras coisas, ser capaz de armazenar as informações como objetos de classes, sendo que tais classes poderiam ser organizadas na forma de uma hierarquia de generalização-especialização (termo conhecido popularmente como Herança). O conjunto de características obrigatórias é complementado por um conjunto de características opcionais, as quais incluíam, por exemplo, a Herança Múltipla. A essência do primeiro manifesto era a de propor as características para um novo modelo de dados que contemplasse todos as características da OO.

Um grupo que promove os bancos de dados Relacionais publicou, em 1990, o “Third Generation Data Base System Manifesto”. O documento, também conhecido como Segundo Manifesto, foi uma reação ao Primeiro Manifesto, argumentando que o modelo de dados Relacional é simples, eficiente e que poderia ser estendido para acomodar as mudanças requeridas pelas novas aplicações de computadores.

Embora uma análise criteriosa da influência destes Manifestos no desenvolvimento de sistemas de banco de dados atuais esteja fora do escopo deste texto, é inegável que a indústria se voltou a atender as necessidades levantadas. Em linhas gerais, pode-se afirmar que os sistemas de bancos de dados que seguem o modelo atualmente denominado Objeto-Relacional ou Relacional-Estendido foram influenciados decisivamente pelo Segundo Manifesto, enquanto que as opções Orientadas a Objetos seguem as propostas do Manifesto de Atkinson.

Principais Características dos Bancos de Dados Orientados a Objetos

Um BDOO possui, em geral, o objetivo de armazenar dados segundo um modelo muito parecido com o adotado pelas linguagens de programação OO. Em um BDOO, os dados são armazenados em objetos, sendo que tais objetos são regidos pelas características determinadas por suas respectivas classes. Assim, os conceitos essenciais do paradigma de objetos, tais como o encapsulamento, polimorfismo e troca de mensagens entre objetos são presentes em nível de banco de dados.

A Figura 1 apresenta um modelo de classes na notação UML (Unified Modeling Language) que é utilizado em um exemplo ilustrativo do uso de BDOO no armazenamento de dados. O diagrama da figura 1 contém seis classes, denominadas Pessoa, PapelPessoaUniversidade, Turma, Professor, Aluno e UniversidadeRaiz. As classes UniversidadeRaiz e Pessoa são abstratas (i.e., são utilizadas somente para reutilização da sua estrutura, não permitindo a criação de objetos). Neste modelo proposto, uma Pessoa pode assumir simultaneamente vários papéis no sistema, onde os papéis são definidos como sub-classes de PapelPessoaUniversidade. É importante observar que UniversidadeRaiz é uma classe que não existe no domínio do problema, tendo sido criada somente com o objetivo de definir uma superclasse única que possa ser eventualmente utilizada para acomodar mudanças necessárias para todas as classes do sistema.

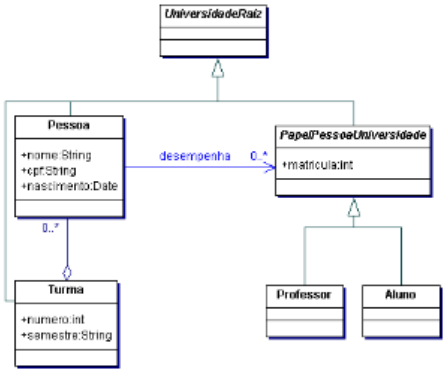

Figura 1. Modelo de Classes parcial usado para exemplificar o uso de BDOO

Identificador de Objetos - OID

A principal característica da OO em sistemas de banco de dados consiste na utilização de um identificador único gerado automaticamente pelo sistema, denominado OID (Object Id). Para ilustrar o uso do OID, será utilizado o modelo apresentado na figura 1 no banco de dados CA-Jasmine.

A Figura 2 apresenta, assim, na parte superior o Class Browser do Jasmine. Class Browser é a ferramenta que permite manipular as classes, objetos, consultas e métodos para o esquema de um banco de dados. No retângulo rotulado como Class Family são apresentadas todas as classes do sistema (subclasses de UniversidadeRaiz). O Class Browser exibe no lado direito os objetos criados a partir da classe Pessoa: o que está listado como univCF01::1026:1025 e univCF01::1026:1026 corresponde exatamente aos OIDs correspondentes para cada objeto da classe.

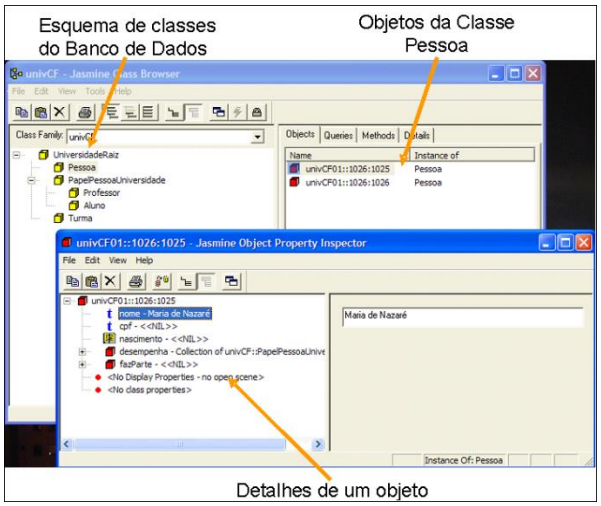

Figura 2. Exemplo de Modelo de Classes no BDOO Jasmine

Referências Para Outros Objetos

Conforme o modelo de classes UML apresentado na figura 1, as associações são elementos importantes na construção de um modelo de dados que influencia, em última instância, o programa associado. Como a referência a objetos é realizada sempre em função dos OIDs, as associações são implementadas na forma de variáveis (atributos) do objeto que contém os OIDs dos objetos associados.

Além dos atributos da classe (nome, cpf e nascimento) são incluídas as referências para 'desempenha' e 'fazParte' correspondendo às associações desempenha papel e composição de turma, respectivamente. Assim, no Jasmine é utilizado o tipo Collection para armazenar associações do tipo um para muitos existentes no esquema de banco de dados.

Linguagens de Definição e Manipulação de Dados

Todos os sistemas de banco de dados possuem linguagens para definição e manipulação de dados que são essenciais no seu uso diário, tanto por parte dos usuários-finais quanto dos desenvolvedores de aplicações. Enquanto que nos Bancos de Dados Relacionais a linguagem SQL é um verdadeiro padrão, sendo utilizado em praticamente todos os produtos disponíveis, no mundo OO isto não ocorre, pois grande parte dos produtos adotam linguagens proprietárias. Assim, esta seção utiliza o exemplo de Universidade apresentado na seção anterior para descrever a linguagem ODQL do Jasmine.

O Jasmine fornece uma linguagem proprietária que serve tanto para a criação quanto para manipulação de banco de dados. A figura 3 mostra parcialmente o script ODQL para criação do modelo de classes do sistema Universidade. A linha 7 descreve a família de classes (ClassFamily [1]) padrão a ser utilizada (systemCF), a qual será utilizada para derivar a família do sistema, denominada UniversidadeRaiz.

A linha 13 apresenta a definição da classe UniversidadeRaiz. A definição da classe Pessoa, por sua vez, ocupa das linhas 18 a 34 da figura 3: a linha 23 define a sua super-classe (UniversidadeRaiz) enquanto que os atributos da classe estão definidos nas linhas numeradas de 28 a 37. Deve-se ressal-

tar que, tal como definido no modelo UML apresentado na figura 1, a associação unidirecional 'desempenha' definida para a classe Pessoa e PapelPessoaUniversidade, corresponde à definição do atributo desempenha localizado na linha 31 do script ODQL. Por outro lado, a agregação bidirecional entre as classes Turma e Pessoa (turma é composta por pessoas, enquanto que uma pessoa sabe a turma a qual ela pertence) afeta a definição da classe Pessoa da seguinte forma: a linha fazParte (linhas 34-37 do script) define a lista de objetos da classe Turma que estão associados.

```
/*************************************************************
1. Arquivo: Univ.odql
2. Definição de Classes para Sistema Universidade
3. v1.00 Jan 2003
4. Rodrigo Quites Reis / Carla A. Lima Reis
5. *************************************************************/
6. defaultCF systemCF;
7. /* Class: UniversidadeRaiz
8. Super-classe fornecida no caso que seja necessária uma
9. alteração global no comportamento de todas as classes do
10. sistema. Abordagem similar a dos exemplos fornecidos pela CA */
11.
12. defineClass univCF::UniversidadeRaiz
13. super: systemCF::Composite
14. description: "Classe base p/ classes de Universidade"
15. {
16. };
17. /* Class: Pessoa
18. Classe generica para armazenar informacoes sobre uma Pessoa.
19. Contem um alista de papeis que pode desempenhar
20. */
21. defineClass univCF::Pessoa
22. super: univCF::UniversidadeRaiz
23. description: "Pessoa"
24. {
25. maxInstanceSize: 8;
26. instance:
27. String nome;
28. String cpf;
29. Date nascimento;
30. List desempenha
31. description: "Lista de papeis desempenhados pela Pessoa"
32. default: List{};
33. List fazParte
34. description: "Turmas em que a pessoa participa"
35. default: List{};
36. };
```

Figura 3. Script para criação da classe Pessoa no Jasmine

A Figura 4 apresenta um script ODQL usado para criar alguns objetos no modelo proposto. Um objeto rotulado como 'oPessoa' é criado na linha 7, e seus valores são definidos nas linhas 8, 9 e 10. As linhas 11 e 12 utilizam o conceito de polimorfismo para definir uma variável do tipo PapelPessoaUniversidade (linha 11) para em seguida instanciá-la com um objeto da sub-classe Professor. Finalmente, a linha 14 associa o objeto 'oPapel' à lista 'desempenha' do objeto 'oPessoa'.

```
1. /* Arquivo: EntradaDadosExemploUniversidade.odql

2. Script para entrada de dados quando não existem métodos
```

```
3. definidos
4. */
5. defaultCF universidadeCF;
6. Pessoa oPessoa;
7. oPessoa = Pessoa.new();
8. oPessoa.nome = "Maria de Nazare";
9. oPessoa.nascimento = Date.construct(1972,2,26);
10. oPessoa.cpf = “000000000-00”;
11. PapelPessoaUniversidade oPapel;
12. oPapel = Professor.new();
13. oPapel.matricula = "0000193";
14. oPessoa.directAdd("desempenha", oPapel);
```

Figura 4. Script para criação de instâncias das classes Pessoa e PapelUniversidadePessoa

ODQL também é utilizada para definir consultas no banco de dados, retornando valores que normalmente são atribuídos a uma variável, lembrando o cursor da SQL conforme [8]. Por exemplo, a figura 5 mostra uma consulta que obtém uma lista dos objetos cujo nome é Maria de Nazaré.

```
List pessoaList;
 pessoaList = Pessoa from Pessoa where Pessoa.Nome = “Maria de Nazaré”
```

Figura 5 Exemplo de consulta em ODQL

Existem outras abordagens adotadas por diferentes sistemas de BDOO no que diz respeito à manipulação e consulta em banco de dados. Por exemplo, o padrão ODMG [9] propõe a linguagem OQL (Object Query Language), específica para consulta em sistemas BDOO. Por outro lado, o Caché permite o uso do próprio SQL para manipulação de dados.