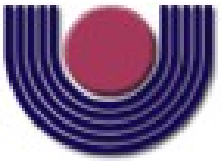

UNIOESTE – Universidade Estadual do Oeste do Paraná

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

Colegiado de Informática

Curso de Bacharelado em Informática

Bancos de Dados Orientados a Objetos e Objeto/Relacional

Jefferson A. do Rosario, Hemerson L. S. Carlin e Alexandre S. Cardoso

CASCAVEL

2009

JEFFERSON A. DO ROSARIO

HEMERSON L. S. CARLIN

ALEXANDRE S. CARDOSO

BANCOS DE DADOS ORIENTADOS A OBJETOS E OBJETO/RELACIONAL

Monografia apresentada como nota parcial da disciplina de Banco de Dados I do curso de Bacharelado em Informática, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel

Orientador: Prof. Carlos José Maria Olguín

CASCAVEL

2009

DEDICATÓRIA

Dedicamos a todos os colegas e professores que dedicaram seu tempo na solução de duvidas oriundas no decorrer do trabalho

Lista de Tabelas

Sumário

Resumo

Este trabalho relata uma pequena definição e alguns exemplos dos banco de dados orientado a objetos e objeto-relacional. As ferramentas que nos dão auxilio são utilizadas nos exemplos, podendo assim, ser explicadas na pratica. A visão do usuário é um grande incentivo para que novas ferramentas possam tornar flexível a integração de objetos e base de dados.

Palavras-chave: Banco de Dados Orientado a Objetos, Banco de Dados Objeto-Relacional, ferramentas.

Capítulo 1

Introdução

Banco de dados são softwares que tem como funcionalidade principal armazenar dados específicos onde serão obtidas informações para determinada ação. Hoje os Bancos de dados mais utilizados são os relacionais, no qual todos os dados são armazenados em tabelas que se relacionam umas com as outras, porém, a necessidade de se trabalhar com aplicações mais complexas, levou a evolução dos BDOO's (Bancos de Dados Orientados a Objeto), onde os dados são armazenados na forma de objetos. Mostraremos algumas das funcionalidades de um Banco de Dados Orientado a Objeto, suas principais características, a facilidade de se trabalhar com as LPOO's (Linguagens de Programação Orientadas a Objeto), vantagens e desvantagens dessa nova tecnologia.

Os paradigmas de persistência mais utilizados são o banco de dados relacional e o banco de dados orientados a objetos. Os dois apresentam algumas vantagens e desvantagens, pensando nisso, foi desenvolvido o banco de dados objeto relacional, que visa aproveitar as características boas de cada um dos sistemas de persistência.

Capítulo 2

Banco de Dados Orientado a Objetos

2.1 Como surgiram os BDOO's

O desenvolvimento dos Sistemas de Gerenciamento de Banco de Dados Orientado a Objetos (SGBDOO) teve origem na combinação de ideias dos modelos de dados tradicionais e de linguagens de programação orientada a objetos.

No SGBDOO, a noção de objeto é usada no nível lógico e possui características não encontradas nas linguagens de programação tradicionais, como operadores de manipulação de estruturas, gerenciamento de armazenamento, tratamento de integridade e persistência dos dados.

Os modelos de dados orientados a objetos tem um papel importante nos SGBD´s porque são mais adequados para o tratamento de objetos complexos (textos, gráficos, imagens) e dinâmicos (programas, simulações), por possuírem maior naturalidade conceitual e, finalmente, por estarem em harmonia com fortes tendências em linguagens de programação e engenharia de software. A junção entre as linguagens de programação e banco de dados é um dos problemas que estão sendo tratados de forma mais adequada no contexto de orientação a objetos[8].

2.2. Em que são utilizados os BDOO´s

2.2.1. Objetos complexos

Os objetos complexos são formados por construtores (conjuntos, listas, tuplas, registros, coleções, arrays) aplicados a objetos simples (inteiros, booleanos, strings). Nos modelos

orientados a objetos, os construtores são em geral ortogonais, isto é, qualquer construtor pode ser aplicado a qualquer objeto. Em SGBDOO, também podemos utilizar estes tipos de dados estruturados, assim sendo, a consulta ao banco de dados precisa ser mais complexa, pois ao invés de acesso a tabelas e registros, é necessário o acesso a listas, tuplas, arrays, entre outros[7].

2.2.2. Exemplos de aplicações complexas

- Projetos de engenharia e arquitetura.
- Experiências cientificas.
- Telecomunicações.
- Sistemas de informações geográficas.
- Multimídia.

2.2.3.Características das aplicações complexas

- Transações de duração mais longa;
- Novos tipos de dados para armazenar imagens ou grandes itens de texto;
- Necessidade de definir operações específicas de aplicações não-padronizadas.

2.3. Surgimento dos BDOO'S

Cada objeto possui um identificador de objeto ou OID (object identifier), que o torna único, não usa a linguagem sql, por isso não há querys, na verdade você busca por seus objetos através de metodologias predefinidas. Chamamos estas metodologias de Native Query's[6].

Na diferenciação do modelo relacional e do orientado a objeto, representado na Tabela 1.

Modelo Relacional	Modelo OO
Tabela (entidades)	Objetos
Linhas (registros)	Tuplas
Query's (consultas,etc)	Native Query's
Sql Ansci	Métodos, construtores

Tabela 1: Como o dado é representado tanto no modelo relacional como no orientado a objetos

A forma de acesso aos dados no banco é remodelada porque os SGBD´s orientados a objetos sugerem novos tipos de dados como sequências de bits, ponteiros, linhas, números complexos e elementos de dados do tipo array. Para acessar uma array, um modo especial de consulta teria que ser construído, por exemplo:

select carro *from* **vendas** *where* vendidos > 200;

A consulta acima relacionada retorna os carros vendidos com quantidade acima de 200 unidades.

2.4. Exemplos de Sistemas de Gerência Banco de Dados Orientado a Objetos

2.4.1. O SGBD Órion

Existem vários tipos de SGBDOO, vários deles de suma importância para determinadas funções. Dentre eles existe o Órion que é muito utilizado em perícias. O Órion conta com 1103 veículos de carga e 4121 veículos de passeio e comerciais leves cadastrados em seu banco de dados, alem de ser o mais barato do mercado. Presente em mais de 640 oficinas, o Órion possibilitou a realização de mais de130 mil perícias, no ano de 2006, e mais de 58 mil, até maio deste ano, pelo processo de imagem.

Com o objetivo de atuar cada vez mais na melhoria do software, foi oferecida uma nova versão do Órion. As oficinas e seguradoras contam com as seguintes novas funcionalidades:

Comparativo de revisões: Possibilita a oficina a total gestão do processo de peritagem;

Laudo em extensão XML: Possibilita a integração com o sistema de gestão interna da oficina;

Novo layout da agenda de visitas: Possui todas as informações necessárias para o trâmite de realização de orçamento e comunicação direta com o perito da seguradora;

Novo layout de fotos: Possibilita a inserção de mais de 30 fotos por processo;

Consulta eletrônica de peças: Permite a consulta eletrônica de peças, tanto por descrição como por part-number.

2.4.2. DB4O

Existe também o DB4O, um poderoso SGBDOO para manipulação de objetos como base de dados, fácil maneira de se armazenar objetos nativamente em JAVA ou. NET (Próxima

Release da suporte a PHP), possui uma performance ate 40X maior que qualquer banco de dados relacional, processa aproximadamente 200.000 objetos por segundo, seu código é aberto e seu custo é muito baixo(praticamente zero).

2.4.3. O Caché

O SGBD Caché realiza a união entre as características dos modelos relacionais e orientados a objeto, pois possui suporte a tabelas e objetos. O Caché é um banco de dados pós-relacional, com estrutura de dados multidimensional, que combina as tecnologias de orientação a objetos e relacional. A arquitetura unificada de dados constrói uma camada de descrição para objetos e tabelas relacionais que são mapeados diretamente em sua estrutura multidimensional. Assim, aplicações relacionais podem coexistir com novos componentes de negócios construídos com a tecnologia de objetos. Todos os SGBD´s desenvolvidos após o modelo relacional podem ser chamados de pós-relacional. Cada uma das diversas tecnologias contidas no Caché é aplicada a determinadas tarefas. A tecnologia de objetos é ideal para a modelagem de dados complexos e a relacional favorece a análise de dados e a geração de relatórios. O SGBD Caché, destaca-se com sua arquitetura unificada de dados e a integração com tecnologias e o desenvolvimento de aplicações. Além de seu desempenho ele permite a integração entre a linguagem padrão de banco de dados, que é a SQL (*Structured Query Language* – Linguagem de Consultas Estruturada), e Objetos, assim trabalhando com SQL e OQL (*Object Query Language* – Linguagem de Consultas a Objetos). A ferramenta Studio,nativa do Caché, é um grande facilitador na criação e manipulação das classes que constituem a base de dados.

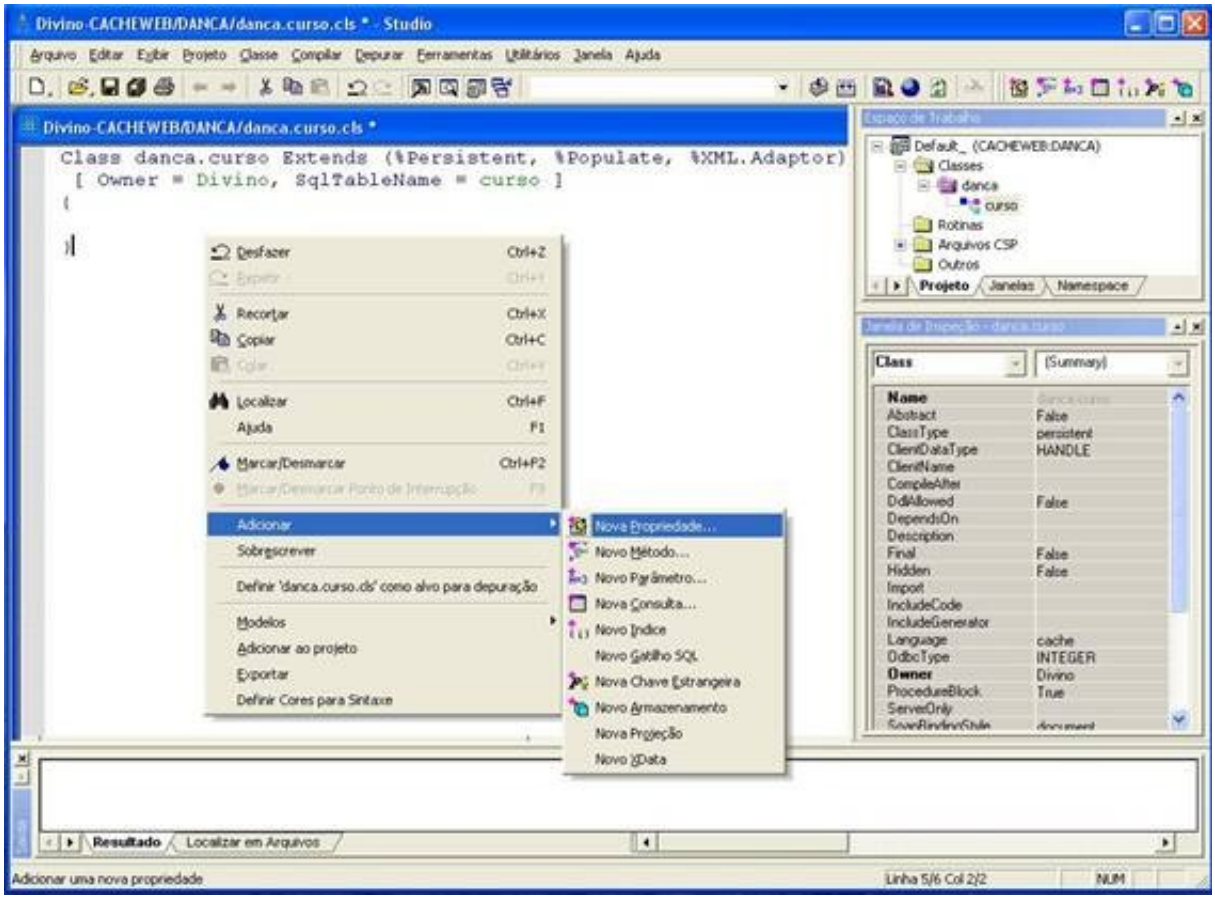

Figura 1: Interface gráfica da ferramenta Studio do Caché

2.4.4. Banco de Dados de Objetos ZODB

O Banco de Dados de Objetos do Zope (ZODB) fornece um banco de dados orientado a objeto para Python que provê um alto grau de transparência. Aplicações podem tirar vantagem dos recursos do banco de dados objeto com pouca, ou nenhuma, alteração na lógica da aplicação. Com a exceção de objetos "root" (raiz), não é necessário consultar ou atualizar objetos dentro de interações do banco de dados. Objetos são obtidos e atualizados através de interações normais de objetos. Uma interface de armazenamento conectável fornece uma gama incrível de flexibilidade para gerenciar dados. Transações podem ser desfeitas de uma maneira que mantem a integridade da transação. Um cache de objetos fornece alto desempenho, uso eficiente de memória, e proteção contra buracos na memória impostos por referências circulares. O ZODB é um software livre, mantido sob a licença ZPL.

2.4.5. GemStone

O SGBDOO GemStone, que utiliza a técnica de objetos distribuídos, oferece uma arquitetura Cliente/Servidor escalável e robusta com o seu sistema de controle de objetos. A comunicação do GemStone se baseia no ORB (Object Request Broker), que por sua vez utiliza o CORBA (Common Object Request Broker Architecture) para organizar os padrões dos objetos que servirão para a troca de mensagens. A OMG (Object Management Group) garante então, os padrões de interoperabilidade entre os sistemas de objetos gerenciados pelo CORBA.

A OMG deliberou os seguintes padrões de trabalho para o Inter-ORB:

GIOP (General Inter-ORB Protocol) que especifica o formato das mensagens e a representação dos dados para toda a intercomunicação do ORB. O GIOP garante, portanto, que a ordenação dos bytes transmitidos não são problemas entre os ORB´s existentes.

IIOP (Internet Inter-ORB Protocol) que permite a ligação entre o GIOP e o protocolo de comunicação TCP/IP.

O CORBA 2.0 possui ambos os protocolos de comunicação: o GIOP e o IIOP. A arquitetura de objetos distribuídos do CORBA estende os limites do GemStone para as linguagens de objetos adicionais, interfaces de programação de aplicações e para serviços não objetos. A adoção da padronização CORBA garante a interoperabilidade entre os diversos sistemas de objetos nas empresas. A grande migração da indústria de sistemas monolíticos para sistemas distribuídos, fez com que a OMG focasse o CORBA como seu ponto de ação central, reunindo seus esforços para a padronizar os objetos componentes de sistemas de informações distribuídos. A plataforma Windows desenvolve seus aplicativos baseados na tecnologia de objetos distribuídos através do COM (Component Object Model) e o OLE (Object Linking and Embedding).

2.4.6 Oracle 10g

Os SGBD’s mostrados anteriormente são puramente orientados a objeto, porem existe também os Objeto relacionais que misturam banco de dados relacional com conceitos de orientação a objetos. Um exemplo é o Oracle 10g que usa SQL no sistema Objeto Relacional.

2.4.6.1.Como isso funciona?

Existe no Oracle o Object Type que um tipo definido pelo usuário na qual equivale ao de classe em POO (Programação Orientada a Objeto). O Object Type captura tanto a estrutura como o comportamento de um objeto. A sintaxe segue o exemplo abaixo:

CREATE TYPE <nome do tipo> AS OBJECT (<lista de atributos e métodos>);

Exemplo:

CREATE TYPE tipo_pessoa AS OBJECT (nome VARCHAR2(30), fone VARCHAR2(20));

2.5. Vantagens

Entre as Vantagens dos SGBD's OO, podemos destacar:

- Capacidade de Armazenamento de Objetos
- Poder de Processamento de Requisições
- Não possuem Chaves Primarias nem Estrangeiras, aumentando o desempenho das consultas e processos
- Os Objetos se comunicam entre si através de mensagens.

2.6. Desvantagens

Entre as Desvantagens dos SGBDOO's podemos destacar:

- Falta de Padronização das linguagens de manipulação dos dados;
- Alto custo de aquisição das novas tecnologias;
- Curva de aprendizagem e adaptação ao novo ambiente demorada.

Capítulo 3

Banco de Dados Objeto Relacional

3.1 O que é um BDOR

Os sistemas de persistência, a princípio, eram divididos em 2 categorias, sendo banco de dados relacional e banco de dados orientados a objetos. Os bancos de dados relacionais (BDR), nos dias de hoje estão mais consolidados no mercado de trabalho, tiveram anos de pesquisa e aprimoramento para poder chegar ao nível de eficiência que possui. Eles possuem uma ótima otimização em consultas e um ótimo gerenciamento de transições. Em contrapartida, são muito mais complicados de serem entendidos e implementados.

A partir desses dois modelos de persistência, surgiu a ideia de banco de dados objeto-relacional (BDOR), que combina características de ambos sistemas de persistência. O BDOR possui a riqueza do BDOO, banco de dados orientado a objetos, e a eficiência no gerenciamento de dados do BDR. Os BDOR são uma tecnologia relativamente nova no mercado de trabalho e vem ganhando seu espaço continuamente[2].

A ideia de um banco de dados objeto relacional, é basicamente armazenar um objeto de forma relacional, reduzindo assim a impedância da programação orientada a objetos utilizando o banco de dados relacional. As tabelas utilizadas na persistência, são as classes, e os registros são representados pelas instâncias das classes correspondentes.

Não é necessária uma correspondência direta entre as tabelas de dados e as classes do programa. A relação entre as tabelas onde originam os dados e o objecto que os disponibiliza é configurada pelo programador, isolando o código do programa das alterações à organização dos dados nas tabelas do banco de dados.

3.2. Ferramentas de auxílio

Para poder privar o programador de códigos SQL, existem ferramentas que fazem a transição de um objeto para uma tabela, algumas ferramentas que fazem esta transição são o Hibernate, ficheiros XML, Doctrine, ORMer, Propel entre outras.

3.2.1. Hibernate

Considerando a Figura 2, os passos para utilização do hibernate são: criar as classes artista, cd, música e capa, essas classes são feitas normalmente em Java. Em seguida, deve-se mapear cada uma das classes feitas, esse mapeamento que permite ao Hibernate criar as tabelas para armazenar o objeto de forma relacional[1]. Após fazer o mapeamento, deve-se criar um arquivo de configuração do hibernate, com ele a ferramenta poderá fazer a transição corretamente. Após todos os passos cumpridos, o programa está pronto para funcionar, porém os arquivos de mapeamento (.hbm.xml), bem como o arquivo de configuração do Hibernate (hibernate.cfg.xml), devem estar no classpath da aplicação[3].

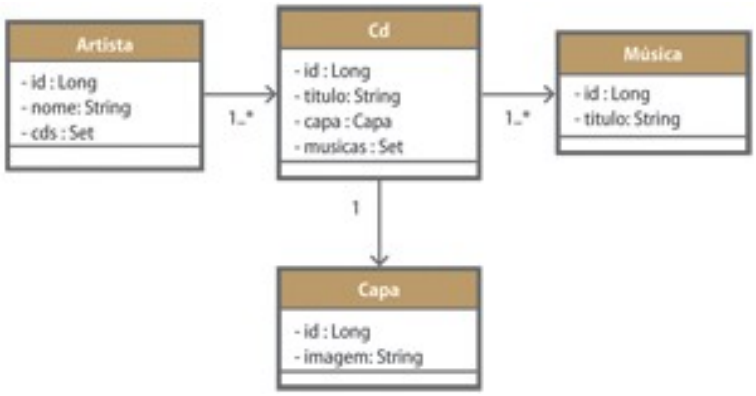

Figura 2: Diagrama de Classe

3.2.2. Doctrine

O Doctrine é um objeto relacional mapper ou ORM (mapeador de objeto-relacional). A ideia é acessar todos os seus dados mySQL ou de banco através de objetos PHP. Ele segue o design pattern criado pelo Martin Fowler de Active Record[4], que é a ideia de objeto ser a referência da tabela.

Capítulo 4

Conclusões

Os bancos de dados são, sem duvida nenhuma, uma peça fundamental no mercado da informação. A evolução dos SGBDOO's facilitará a forma de se trabalhar com dados mais complexos e com os objetos das LPOO's. O uso da orientação a objetos em sistemas de banco de dados é cada vez mais crescente.

É claro que isso não quer dizer que o modelo relacional será extinto, entretanto, daqui a alguns anos a tendência da tecnologia OO terá significativa superioridade com relação ao seu uso em aplicações corporativas.

Os bancos de dados orientados a objetos (BDOO), possui um pior desempenho, se comparado ao BDR, e é pouco utilizado em aplicativos comerciais. O BDOO tem uma maior simplicidade em sua implementação e heterogeneidade a nível de modelo, porém possui menor capacidade de consulta. É adequado para implementações não convencionais.

O paradigma de objeto relacional ainda não esta bem consolidado no mercado de trabalho, porém vem ganhando seu espaço rapidamente devido ao seu conjunto de características que facilitam a implementação da persistência. Considerando que o BDOO ainda não está completamente desenvolvido, pode ser que futuramente, o paradigma de objeto relacional venha a perder seu espaço para p BDOO. Porém ate la, é bem provável que o BDOR se torne o sistema de persistência mais utilizado.

Referências Bibliográficas

[1] http://www.br.re.dhat.com/pdf/jboss/Hibernate.pdf, acessado em 16 de setembro de 2009 ás 11:15.

[2] http://www.inf.ufsc.br/~ronaldo/bdnc/7-bdor.pdf, acessado em 16 de setembro de 2009 ás 11:03.

[3] http://www.mundojava.com.br/NovoSite/hibernate.shtml, acessado em 15 de setembro de 2009 ás 18:35.

[4] http://www.rafaelcunha.com/2009/08/07/doctrine-orm/, acessado em 16 de setembro de 2009 ás 14:34.

[5] Ramos, Ricardo. Banco de Dados Orientado Objeto

[6] Silberschatz, A., Korth, H. and Sudarshan, S. (1999) “Sistema de Banco de Dados”, Makron Books.

[7] Fontes, Attila. Nova Geração, a Tecnologia dos BDOO’s

[8] Rational, Inc – 004. Disponível em: http://www.malima.com.br/article_read.asp?id=40