PRISCILLA APARECIDA VALE SILVA

BANCO DE DADOS ORIENTADO A OBJETOS: ESTUDO COMPARATIVO

Trabalho de conclusão de curso apresentado ao Curso de Ciência da Computação.

UNIVERSIDADE PRESIDENTE ANTÔNIO CARLOS

Orientador: Prof. Eduardo Macedo Bhering

BARBACENA
2004

PRISCILLA APARECIDA VALE SILVA

BANCO DE DADOS ORIENTADO A OBJETOS: ESTUDO COMPARATIVO

Este trabalho de conclusão de curso foi julgado adequado à obtenção do grau de Bacharel em Ciência da Computação e aprovado em sua forma final pelo Curso de Ciência da Computação da Universidade Presidente Antônio Carlos.

Barbacena – MG, 21 de junho de 2004.

_____

Orientador: Prof. Eduardo Macedo Bhering

_____

Prof. Luís Augusto Mattos Mendes - Membro da Comissão Examinadora

_____

Profª. Lorena Sophia Campos de Oliveira - Membro da Comissão Examinadora

AGRADECIMENTOS

Agradeço a Deus, que me iluminou e concedeu forças para persistir, a minha mãe que sempre me apoiou, ao meu Orientador que com sua competência me ajudou a concluir este importante trabalho e aos professores da UNIPAC pelo aprendizado.

RESUMO

Este trabalho consiste em desenvolver um estudo comparativo entre Sistemas Gerenciadores de Bancos de Dados Orientados a Objetos (SGBD-OO). Para isto foram apresentados conceitos de orientação a objetos aplicados à bancos de dados e as principais características dos SGBD-OO estudados.

SUMÁRIO

1 INTRODUÇÃO

Os Bancos de Dados surgiram aproximadamente em meados dos anos 60, decorrentes da possibilidade dos computadores armazenarem e gerenciarem grandes quantidades de dados em meios de armazenamento permanente de acesso direto e eficiente a cada dado (em geral discos magnéticos) e de necessidade de estruturar esses dados e prover rotinas padronizadas de acesso a eles. Um Sistema Gerenciador de Banco de Dados (SGBD) é um programa, ou um conjunto de programas destinado a controlar todos os aspectos de um Banco de Dados, tais como a declaração de sua estrutura de dados, gravação e leitura dos dados, recuperação de falhas na comunicação ou no meio de gravação, controles de concorrência, de acesso e de segurança dos dados, entre outros (NASSU, 1999).

Um modelo de dados é uma coleção de ferramentas conceituais utilizada para descrever a estrutura dos dados de um SGBD. Os primeiros modelos de dados utilizados foram o modelo hierárquico e o modelo de redes. Em 1970 Codd apresentou o modelo relacional (MR), que se impôs como padrão dos Sistemas Gerenciadores de Banco de Dados (SGBD) atuais. Todos esse modelos foram concebidos visando a sua utilização em ambientes comerciais onde, em geral, existe uma grande quantidade de registros de mesma estrutura (por exemplo, um cadastro de clientes de um mesmo banco) (NAVATHE, 1999).

Modelos e sistemas de dados convencionais vêm sendo muito bem sucedidos no que diz respeito ao desenvolvimento da tecnologia de banco de dados. Entretanto, eles apresentam algumas deficiências quando aplicações de banco de dados mais complexas precisam ser projetadas e implementadas. Como exemplo, o CAD/CAM (banco de dados para projetos de engenharia e arquitetura), o CASE (desenvolvimento de sistema de computação auxiliado por computador), o SIG (sistemas de informação geográfica), aplicações em medicina, telecomunicações, sistemas multimídia, etc. Estas aplicações mais novas diferem das comerciais por apresentarem estruturas mais complexas para objetos, transações de duração mais longa, novos tipos de dados para armazenar imagens ou grandes itens de texto, bem

como a necessidade de definir operações específicas de aplicações não padronizadas. Os banco de dados orientados a objetos foram desenvolvidos a fim de atender as necessidades destas aplicações mais complexas (NAVATHE, 1999).

Uma importante característica dos bancos de dados orientados a objetos é o poder que eles dão ao projetista para especificar tanto a estrutura de objetos complexos como as operações que podem ser aplicadas a esses objetos. Outra razão para o uso de banco de dados orientados a objetos é o uso crescente de linguagens de programação orientada a objetos para o desenvolvimento de aplicações de software (NAVATHE, 1999).

No capítulo 2 serão apresentados os conceitos fundamentais de Orientação a Objetos aplicados à Bancos de Dados. No capítulo 3 é feito um levantamento dos principais Sistemas de Gerenciamento de Banco de Dados Orientado a Objetos (SGBD-OO) e suas principais características. No capítulo 4 é feita a comparação entre as ferramentas apresentadas no capítulo anterior. Finalmente, no capítulo 5 são apresentadas as conclusões do trabalho.

2 CONCEITOS DE ORIENTAÇÃO A OBJETOS APLICADOS À BANCOS DE DADOS

O desenvolvimento dos Sistemas de Gerenciamento de Banco de Dados Orientado a Objetos (SGBD-OO), originou-se da combinação de idéias dos modelos de dados tradicionais e de linguagens de programação orientada a objetos. Os SGBD-OOs, possuem características não encontradas nas linguagens de programação tradicionais, como operadores de manipulação de estruturas, gerenciamento de armazenamento, tratamento de integridade e persistência dos dados (Sampaio, 2004).

Os modelos de dados orientados a objetos tem um papel importante nos SGBDs, porque, em primeiro lugar, são mais adequados para o tratamento de dados complexos e que representam fielmente a realidade das aplicações gráficas, de hipertexto (como internet), de vídeo, imagem e som, programas e simulações. Depois, por possuírem maior naturalidade conceitual e, finalmente, por estarem em consonância com fortes tendências em linguagens de programação e engenharia de software de forma mais adequada no contexto de orientação a objetos.

No SGBD-OO, 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 (Sampaio, 2004).

2.1OBJETO COMPLEXO

Os objetos complexos são formados por construtores: listas, tuplas, registros são alguns exemplos. Um sistema deve ter no mínimo os construtores: lista e tupla Nos modelos orientados a objetos, os construtores são em geral ortogonais, isto é, qualquer construtor pode

ser aplicado a qualquer objeto. No modelo relacional este não é o caso, visto que só é possível aplicar o construtor de conjuntos as tuplas e o construtor de registro a valores atômicos (Sampaio, 2004).

2.2IDENTIDADE DE OBJETO

Um SGBD-OO fornece uma identidade única para cada objeto independente armazenado no banco de dados. Esta identidade única é geralmente implementada através de um identificador de objeto (OID) único, gerado pelo sistema. O valor de um OID não é visível para o usuário externo, mas é utilizado internamente pelo sistema para identificar cada objeto univocamente e para criar e gerenciar referências interobjeto (NAVATHE, 1999).

Sua principal propriedade é ser imutável, isto é, o valor de um OID para um determinado objeto não deve se alterar. Desta forma a identidade do objeto será preservada no mundo real, que está sendo representado. Portanto um SGBD-OO deve possuir algum mecanismo para gerar OIDs e preservar a propriedade de imutabilidade. Cada OID deve ser utilizado apenas uma vez, ou seja, mesmo se o objeto for removido do banco de dados, seu OID não deve ser atribuído a outro objeto. Essas duas propriedades significam que o OID não deva depender de quaisquer outros valores de atributo de objeto, uma vez que o valor do atributo pode ser modificado ou corrigido. Também pode-se basear o OID no endereço físico do objeto armazenado, desde que o endereço físico pode ser modificado após uma reorganização física do banco de dados. No entanto alguns sistemas utilizam o endereço físico como OID, com o propósito de aumentar a eficiência da recuperação do objeto. Se o endereço físico do objeto se modifica, um ponteiro indireto pode ser posicionado no endereço anterior, o que vem a fornecer a nova localização física do objeto (NAVATHE, 1999).

2.3ENCAPSULAMENTO

O conceito de encapsulamento está intimamente ligado com o conceito de abstração. Num dado objeto somente interessa ao cliente as funções que ele executa e não a

implementação da mesma. Na parte de interface é apresentada descrição das operações que o objeto executa, o que é acessível ao cliente, a implementação de tais operações fica encapsulada e só é visível ao próprio objeto (Rubinei, 2004).

Se precisar mudar o valor de alguma variável de instância de um objeto, deve-se enviar uma mensagem que recebe um valor como argumento e faz a alteração. Não é possível acessar variáveis de instância diretamente. Desta forma, os campos estarão escondidos para os usuários, o que previne alterações acidentais. Dizemos então que as variáveis de instância e seus métodos estão encapsulados em uma única entidade (Rubinei, 2004).

Se alguma modificação ocorrer em variáveis de instância de um certo objeto, é possível saber exatamente quais métodos interagiram com elas: são os métodos desse objeto. Nenhum outro método pode acessar esses dados. Isso simplifica a escrita, manutenção e alteração de programas (Rubinei, 2004).

2.4TIPOS E CLASSES

Um conjunto de objetos que possuem o mesmo tipo (atributos, relacionamentos, operações) podem ser agrupados para formar uma classe. A noção de classe é associada ao tempo de execução, podendo ser vista como uma representação por extensão, enquanto o tipo é uma representação intencional. Cada classe tem um tipo associado, o qual especifica a estrutura e o comportamento de seus objetos (Sampaio, 2004).

2.5OBJETOS PERSISTENTES E TRANSIENTES

Objetos persistentes são armazenados no banco de dados e persistem após o termino do programa. Os mecanismos usuais para tornar objetos persistentes são a nomeação e a acessibilidade. Objetos transientes são objetos que existem apenas enquanto a aplicação que os criou continuar executando, após o término da aplicação eles deixam de existir (NAVATHE, 1999).

2.6HIERARQUIA DE TIPOS E HERANÇA

Os sistemas OO também devem permitir hierarquias de tipo e herança. Em bancos de dados hierarquias implicam uma restrição nas extensões correspondentes aos tipos de hierarquia. A maioria das aplicações em banco de dados apresentam vários objetos do mesmo tipo ou classe. Desta forma banco de dados OO devem oferecer a capacidade de classificar objetos com base em seus tipos, assim como outros sistemas de banco de dados o fazem. Uma outra exigência apresentada por BDOO é que o sistema permita a definição de novos tipos baseados em outros tipos predefinidos conduzindo a uma hierarquia de tipo (NAVATHE, 1999).

Geralmente, um tipo é definido designando a ele um nome e estabelecendo então um número de atributos e operações para o tipo. Em alguns casos, os atributos e operações são conjuntamente denominados como funções, uma vez que atributos se assemelhem a funções que não contem nenhum argumento. Um nome de função pode ser utilizado para fazer referência ao valor de um atributo ou para fazer referência ao valor resultante de uma operação (NAVATHE, 1999).

2.7HERANÇA MÚLTIPLA

A herança múltipla ocorre quando um subtipo é subtipo de dois ou mais tipos e, assim sendo, herdam as funções (atributos e métodos) de ambos os supertipos. Um problema que pode ocorrer com a herança múltipla é que os supertipos dos quais o subtipo tipo tem herança podem ter funções distintas com o mesmo nome, criando uma ambigüidade (NAVATHE, 1999).

Há várias técnicas para lidar com ambigüidade na herança múltipla (NAVATHE, 1999):

- Fazer com que o sistema verifique se há ambigüidade quando o subtipo for criado e deixar o usuário escolher qual a função a ser herdada nesse momento;

- Utilizar o default do sistema;

- Desautorizar totalmente a herança múltipla se ocorrer uma ambigüidade de nome, forçando, por outro lado, o usuário mudar o nome de uma das funções em um dos supertipos.

2.8POLIMORFISMO

O sentido da palavra polimorfismo é o uso de um único nome para definir várias formas distintas. Em linguagens de programação, polimorfismo é uma facilidade que permite que dois ou mais objetos diferentes respondam a mesma mensagem (Rubinei, 2004).

O objeto emissor não precisa saber como o objeto receptor implementa a mensagem. Apenas os objetos receptores devem se preocupar com isso. Uma analogia ao polimorfismo é o sinal dos colégios. Embora seja um único sinal, ele significa coisas distintas para cada aluno: uns vão para casa, outros para biblioteca e terceiros voltam para sala de aula, todos respondem ao sinal, mas cada um do seu jeito (Rubinei, 2004).

O polimorfismo permite um objeto emissor comunicar-se com um objeto receptor sem ter que entender que tipo de objeto ele é, apenas o objeto receptor deve reconhecer a mensagem (Rubinei, 2004).

3 SISTEMAS GERENCIADORE DE BANCOS DE DADOS ORIENTADOS A OBJETOS

Nos últimos anos foram produzidas muitas implementações de sistemas de BDOO, entre protótipos de empresas comerciais, universidades e produtos comerciais (NASSU, 1999).

Neste capítulo será feito um levantamento dos principais sistemas disponíveis, incluindo para cada um deles as principais características no que diz respeito aos conceitos de orientação a objetos. A escolha das ferramentas estudadas foi feita com base em citações nos materiais utilizados, como fonte de referência e pesquisa em sites especializados.

A seguir serão apresentadas as principais ferramentas estudadas.

3.1CACHÉ

Fundada em 1978 e sediada em Cambridge, Massachusetts, a InterSystems sempre atuou no segmento de Tecnologia da Informação (Intersystems, 2004).

Nos EUA, mais de 80% dos hospitais e laboratórios têm suas aplicações rodando nos sistemas InterSystems. Estes sistemas são confiáveis para os segmentos complexos do mercado como: áreas financeiras, telecomunicações, órgãos governamentais e até mesmo empresas de internet (Intersystems, 2004).

3.1.1 CARACTERÍSTICAS PRINCIPAIS

Tecnologia Avançada de Objetos: Caché suporta um modelo completo de objeto, incluindo encapsulamento, polimorfismo, heranças múltiplas, coleções e relacionamentos.

Objetos Caché podem ser projetados como Java, EJB, COM, C++, XML e SOAP para uma compatibilidade mais fácil com outras ferramentas e tecnologias (Intersystems, 2004c).

Arquitetura Unificada de Dados: Caché apresenta uma Arquitetura Unificada de Dados que automaticamente torna os dados disponíveis tanto como objetos e tabelas, eliminando a necessidade de sincronizar definições de dados ou mapear de uma forma para outra. Arquitetura Unificada de Dados reduz grandemente o tempo de desenvolvimento e custos de processamento então as aplicações são mais rápidas para desenvolvimento, e executam mais rapidamente (Intersystems, 2004c).

Acesso a Dados Objetos: Os dados podem ser modelados como objetos. Proporcionando rápido desenvolvimento de aplicações e modelagem intuitiva de dados complexos (Intersystems, 2004a).

Fácil Projeção de Classes: Com poucos cliques, as classes Caché podem ser projetadas como classes Java, COM, ou C++. Tornando o desenvolvimento mais rápido e fornecendo conectividade a outras tecnologias e ferramentas (Intersystems, 2004b).

Acesso a Dados SQL: Permite acesso relacional à base de dados Caché. Suporta tanto ODBC como JDBC. Aumentando a performance de aplicações relacionais legadas e fornecendo conectividade SQL a queries padrão, relatórios e ferramentas de análise (Intersystems, 2004a).

Acesso a Dados Pós-Relacionais: Fornece controle direto de estruturas pós-relacionais no banco de dados Caché. Permitindo conectividade com sistemas legados e alta performance (Intersystems, 2004a).

3.2JASMINE

O Jasmine é um sistema gerenciador de banco de dados orientado a objetos, desenvolvido pela Computer Associates em conjunto com a Fujitsu em 1996 e lançado em versão de avaliação em Dezembro de 1997 (Shikida, 2004). O Jasmine inclui, além do sistema

de gerenciamento da base, uma ferramenta de desenvolvimento de aplicações: Jasmine Studio.O Jasmine armazena estruturas de classes com suas instâncias, e é manipulado por uma linguagem de consulta a objetos chamada ODQL (Manual do Jasmine, 2004). O sistema suporta as seguintes plataformas: HP e3000, HP ProLiant Servers, HP Unix Servers, IBM, Intel-Based, SUN (Jasmine, 2004).

3.2.1 CARACTERÍSTICAS PRINCIPAIS

ODQL: é a linguagem de criação e manipulação de classes e objetos. A linguagem apresenta os seguintes tipos de dados (Manual do Jasmine, 2004):

 Entidades: existem no Banco de Dados e tem OID

 Literais: são valores simples, como: números e string. Os tipos de literais podem ser: Integer, Date, Decimal, Real, Boolean, String e ByteSequence.

 Tuplas: são valores compostos contendo um certo número de componentes. Estes componentes podem ser entidades, literais, coleções de entidades, coleções de literais (menos tuplas). As tuplas não podem ser usadas na definição de atributos de classe (Manual do Jasmine, 2004).

 Coleções: são tipos de dados complexos que podem ser: array, list, set e bag. Podendo possuir tamanho fixo (array) ou tamanho variável (list, set e bag) (Manual do Jasmine, 2004).

Interfaces com C e C++: O Jasmine oferece interfaces que permitem que se escrevam códigos em C e C++. Sentenças ODQL podem ser diretamente embutidas em C ou C++ , para a escrita de métodos (Manual do Jasmine, 2004).

Suporte para armazenamento estruturado de dados multimídia: A biblioteca de classes multimídia define classes para gerenciamento de dados multimídia. Ela contém classes com definições para diversos tipos de objetos multimídia comuns, tais como: vídeo, áudio, figuras

e diversas classes de suporte (Manual do Jasmine, 2004).

Arquitetura distribuída cliente/servidor: Jasmine usa uma arquitetura de software cliente/servidor. A implementação cliente/servidor permite a um servidor suportar múltiplas bases de dados, e cada cliente pode acessar bases múltiplas em vários servidores (Manual do Jasmine, 2004).

As aplicações Jasmine são executadas em *workstations* clientes, podendo ser em aplicações *stand-alone* ou aplicações *plug-ins* para browsers WEB, como o Netscape Navigator. Aplicações Jasmine comunicam-se com o servidor da base de dados, o qual executa a lógica de negócio da aplicação e provê o armazenamento para objetos multimídia e outros dados na aplicação (Manual do Jasmine, 2004).

Integração da base de dados: Jasmine provê integração completa e suporte para outros bancos de dados, incluindo bancos relacionais, tais como Openingres, Sybase e SQLServer (Manual do Jasmine, 2004).

3.2.2 CONCEITOS IMPORTANTES

Além dos conceitos básicos de orientação a objeto, outros conceitos importantes relativos ao Jasmine são necessários para sua utilização. Esses conceitos são listados a seguir (Manual do Jasmine, 2004):

Família de Classes: uma família de classes *(class famil*y) representa o local lógico onde serão armazenadas as classes. Uma família de classes é armazenada em um Store, sendo que um store pode conter várias famílias de classes (Manual do Jasmine, 2004).

Jasmine Studio: ambiente de desenvolvimento e gerenciamento de classes e objetos. Permite a criação e manipulação de classes e objetos, bem como a criação de aplicações (Manual do Jasmine, 2004).

Modo de Execução: o servidor Jasmine pode ser executado de dois modos: *modo de produção* e *modo de desenvolvimento*. O modo de produção não permite mudanças nos

metadados, ou seja, nas classes, família de classes e stores. Já o modo de desenvolvimento permite mudanças em todos os metadados (Manual do Jasmine, 2004).

Store: um *store* é o local físico onde serão armazenados as classes e os objetos criados no Jasmine. Um *store* pode compreender um ou mais arquivos e pode armazenar uma ou mais famílias de classes (Manual do Jasmine, 2004).

3.3OBJECTSTORE

O ObjectStore é um Sistema Gerenciador de Banco de Dados Orientado a Objetos (SGBD-OO), que trabalha com arquitetura Cliente/Servidor (Ali Adams, 2004). É uma solução de gerência de dados para o desenvolvimento em internet, telecomunicações e outros ambientes distribuídos, combinando características de gerenciamento de dados de objeto com Java e C++, e reduzindo o esforço de codificação em até 60%. O sistema oferece todas as características de SGBD tradicional (constituído por um conjunto de dados associados a um conjunto de programas para acesso a esses dados), incluindo persistência, gerenciamento de transações (controle de concorrência e recuperação), acesso distribuído, consultas associativas sobre grandes volumes de dados e utilitários para a administração do banco de dados (Cyberdyne, 2004).

O sistema está disponível para diversas plataformas, como Microsoft Windows XP, 2000 e NT; Solaris SPARC 32 e 64bit; Unix, Linux e OS/2 (Cyberdyne, 2004) . O sistema também suporta diversos ambientes de rede para o interoperabilidade entre estações de trabalho e PC’s, incluindo suporte para TCP/IP, Novell IPX/SPX, e outros protocolos de rede (Cyberdyne, 2004) .

3.3.1 CARACTERÍSTICAS PRINCIPAIS

ObjectStore é um SGBD-OO passivo, isto é, ele armazena no BD somente o estado dos objetos, e não a implementação do comportamento dos objetos. A definição das classes (interface) e a implementação do comportamento dos objetos faz parte do esquema da

aplicação e somente a interface é que faz parte do esquema do banco de dados, para descrever os tipos de objetos que serão armazenados (Ali Adams, 2004).

Essa abordagem traz um armazenamento eficiente dos objetos, embora acarrete sobrecarga da rede (devido à transferência de dados para processamento na máquina cliente) e não possibilite a existência de procedimentos armazenados (Ali Adams, 2004).

O ObjetctStore possui recursos que tornam possível a modelagem de dados de maneira flexível, possibilitando assim, seu uso em aplicações que demandam estruturas de dados complexos (Cyberdyne, 2004) . Dentre esses recursos, destaca-se a possibilidade de criação de novos tipos dinâmicos, para estender a definição de classes existentes durante a execução do programa, e a existência de bibliotecas de classes para gerenciamento de coleções de objetos, gerenciamento e controle de versões e evolução do esquema (Cyberdyne, 2004).

O sistema ObjectStore possui diferentes pacotes que podem ser adquiridos em separado. Um desses pacotes oferece armazenamento persistente para a linguagem de programação Java e um outro para a linguagem de programação C++. O pacote C++ é estreitamente integrado com a linguagem C++ e oferece o armazenamento persistente para objetos C++, utilizando declarações de classes em C++ como linguagem de definição de dados, através de uma sintaxe de C++ estendida, que inclui construtores adicionais especificamente úteis nas aplicações de banco de dados (NAVATHE, 1999).

O ObjectStore suporta diversos tipos de coleções (*Sets, Bags, Lists, Arrays e Dictionaries*) (Intersystems, 2004)e possui funções aplicadas aos mesmos, incluindo as funções que podem ser utilizadas, respectivamente, para incluir, remover e criar um elemento em uma coleção (NAVATHE, 1999).

As referências entre objetos são implementadas através da utilização do tipo *Reference*, que é o encapsulamento de um ponteiro para um objeto persistente, incluindo a descrição do BD onde se encontra o objeto. Dessa forma, é possível realizar referências a objetos em outras bases de dados. A desvantagem desse mecanismo é o *overhead* causado para armazenar a referência, que pode chegar a 16 bytes por referência (Ali Adams, 2004).

3.4O2

O sistema O2 é um projeto iniciado na França em 1988. Até 1991 o projeto era experimental e, com o fim do convênio que o financiava, tornou-se um produto comercial. Foi criada uma empresa, a O2 Technology, que o passou a comercializar e desenvolver. O O2 pode ser executado em estações Sun Sparc, HP9000, IBM RISC System 6000, Bull DPX/20, Silicon Graphics, SNI RM, Dec Alpha e INTEL. O sistema fornece um ambiente gráfico para criação de telas, O2Loock, e um "browser" para percorrer os objetos no banco de dados. É fornecido também um ambiente integrado para a programação, com um compilador e depurador integrados. Nas versões mais recentes foi incorporado uma ferramenta de ligação do banco de dados à WWW (World Wide Web), que permite que objetos armazenados no banco de dados sejam exibidos em páginas HTML (O2 Web) . O sistema permite integração com as linguagens C e C++, e mais recentemente Java (NASSU, 1999).

3.4.1 CARACTERÍSTICAS PRINCIPAIS

A declaração dos dados é feita através de uma linguagem que é uma extensão da linguagem C, o O2C Java (NASSU, 1999). Os tipos primitivos do O2 são boolean, character, integer, real, string e bit. São disponíveis também construtores de tipos complexos de dados, que podem ser aplicados recursivamente: Tuple, List e Set. A linguagem possui herança múltipla (NAVATHE, 1999).

Não há diferença na declaração dos dados persistentes e não persistentes, o que mantém a ortogonalidade dos tipos e a persistência Java (NASSU, 1999). O SGBDOO O2 suporta persistência através da acessibilidade, que simplifica a programação e impõe a integridade referencial. Quando um objeto ou valor se torna persistente, todos os seus subobjetos fazem o mesmo, liberando o programador de explicitamente realizar esta tarefa. A qualquer tempo, um objeto pode se alterar entre persistente e transiente. Durante a criação do objeto, o programador não precisa decidir se o objeto é persistente (NAVATHE, 1999).

A parte essencial do sistema chamada O2Engine é responsável por grande parte do

funcionamento do O2. Isto inclui o fornecimento de suporte para armazenamento, recuperação e atualização de objetos persistentes armazenados, que podem ser compartilhados por inúmeros programas. O O2Engine implementa o controle de concorrência, recuperação e mecanismo de segurança que são típicos em sistemas de banco de dados. Além disso, o O2Engine implementa um modelo de gerenciamento de transações e mecanismos de evolução do esquema, versionamento, bem como um mecanismo de replicação. A implementação do O2Engine baseia-se na arquitetura cliente/servidor, para acomodar a tendência atual na direção de sistemas informatizados de rede e distribuídos. O componente de Servidor, que pode ser uma máquina servidora de arquivos, é responsável por recuperar dados eficientes e por manter o controle de concorrência e recuperação de informações. O O2Engine apresenta três componentes principais (NAVATHE, 1999):

- Componente de armazenamento: encontra-se no nível mais baixo, sua implementação divide-se entre cliente e servidor.

- Gerência de objetos: lida com estruturação de objetos e valores, com o agrupamento de objetos relacionados em página de disco, indexação de objetos, manutenção da identidade de objetos, realização de operações em objetos, e assim por diante.

- Gerente de esquema: o mais alto nível funcional. Acomoda definições de classe, tipo e métodos; oferece mecanismos de herança; verifica a consistência de declarações de classe e viabiliza a evolução de esquema, que inclui a criação, a modificação e a exclusão de declarações de classe, de modo incremental. Quando uma aplicação aceita um objeto cuja classe tenha sido alterada, o gerente de objeto automaticamente adapta sua estrutura à definição corrente da classe.

3.5GEMSTONE

Introduzido em 1987, o GemStone é o SGBD-OO a mais tempo disponível comercialmente (NASSU, 1999). O sistema foi desenvolvido pela Sérvio Logic, com idéia inicial de transformar a linguagem SmallTalk em uma linguagem para banco de dados, de

onde surgiu a linguagem de declaração e manipulação dos dados, a linguagem OPAL. O sistema é composto basicamente por dois processos principais, o Gem e o Stone. O servidor Gem executa os métodos, além da avaliação das consultas. O monitor Stone é responsável pela alocação de OID's, gerenciamento os objetos persistentes, controle de concorrência, autorização, transações, e serviço de recuperação. O Stone em geral reside na máquina servidora, enquanto o Gem pode ser executado no servidor ou em uma estação cliente. As servidoras podem ser máquinas VAX, SUN 3 e SUN 4, enquanto as máquinas clientes podem ser IBM PC, Masc, SUN3 e 4, ou Tektronix (Cyberdyne, 2004a).

3.5.1 CARACTERÍSTICAS PRINCIPAIS

Servidor SmallTalk: GemStone permite a criação de classes e métodos que são armazenados e executados diretamente no banco de dados. Esses métodos podem ser acessados internamente ou através de aplicações externas, reduzindo significativamente o tráfego na rede e permitindo às aplicações aproveitar o poder de processamento do servidor. Isto também elimina a necessidade de reconstruir ou re-desenvolver aplicações quando as regras de processamento (regras de negócio) forem modificadas (NASSU, 1999).

Acesso concorrente à múltiplas linguagens: GemStone suporta acesso concorrente para aplicações desenvolvidas em Smalltalk, C++ ou C. Todas as aplicações, indepedente da linguagem utilizada, podem ter acesso simultâneo aos mesmos objetos armazenados no banco de dados (NASSU, 1999).

Segurança em nível de objetos: O controle de autorização pode ser aplicado a qualquer objeto do banco de dados, permitindo um "ajuste fino" da segurança de objetos (NASSU, 1999).

Evolução dinâmica do esquema: GemStone suporta modificações do esquema através de versões de classes, o que permite total migração de objetos entre versões de suas classes com uma simples mensagem (NASSU, 1999).

Oferece suporte para estações de trabalho e servidores operando com sistemas UNIX em diversas arquiteturas (Sun, IBM, HP, etc.). GemStone é um dos membros do ODMG, e

suporta este padrão, além do padrão ANSI C++ (NASSU, 1999).

4 COMPARAÇÃO DE BANCOS DE DADOS ORIENTADOS A OBJETOS

A seguir será apresentado um quadro comparativo entre os bancos de dados orientados a objetos citados no capítulo anterior.

SGBDOO / **Característica**	**CACHÉ**	**JASMINE**	**OBJECTSTORE**	**O2**	**GEMSTONE**
1- Suporte à tipos de dados definidos pelo usuário.	SIM	SIM	SIM	SIM	SIM
2- Suporte à mecanismo de herança.	SIM	SIM	SIM	SIM	SIM
3- Suporte à relacionamento de agregação.	-	SIM	SIM	SIM	SIM
4- Suporte à herança múltipla.	SIM	SIM	SIM	SIM	Smalltalk -NÃO Java - SIM
5- Suporte ao conceito de versões.	-	NÃO	SIM	SIM	SIM
6- Checar à cardinalidade entre os objetos.	SIM	SIM	SIM	SIM	NÃO
7- Suporte à longas transações.	-	NÃO	SIM	NÃO	NÃO
8- Suporte à criptografia de dados.	-	NÃO	NÃO	SIM	SIM
9- Linguagem de definição de atributos de objetos.	-	C C++ ODQL Java	C C++ Java Smalltalk	C++ Smalltalk Java	Java Smalltalk
10- Linguagem de definição de métodos	SIM	SIM	SIM	SIM	SIM

SGBDOO / Caracteristica	CACHÉ	JASMINE	OBJECTSTORE	O2	GEMSTONE
de objetos.					
11- Armazena os métodos dos objetos no BD.	-	C C++ ODQL Java	C C++ Java Smalltalk	C++ Java Smalltalk	Java Smalltalk
12- Suporte às linguagens.	-	C++ Java Smalltalk	C++ Java Smalltalk	C++ Java Smalltalk	C++ Java Smalltalk
PADRÕES					
13- Suporte à linguagem de definição de objetos (Object Definition Language - ODL).	-	NÃO	NÃO	NÃO	SIM
14- Suporte à linguagem de consulta à objetos (Object Query Language - OQL).	-	OQL - NÃO ODQL - SIM	NÃO	SIM	NÃO
15- Suporte à integração - ODMG C++.	-	SIM	SIM	SIM	NÃO
16- Suporte à integração - ODMG Smalltalk.	-	NÃO	NÃO	SIM	NÃO
17- Suporte à linguagem SQL padrão, em ambiente interativo.	-	SIM	SIM	NÃO	Smalltalk - SIM Java – NÃO
18- Suporte à linguagem SQL padrão, em ambiente integrado.	-	SIM	SIM	NÃO	Smalltalk - SIM Java – NÃO
19- Suporte à linguagem de consulta baseada em SQL.	-	SIM	SIM	SIM	Smalltalk - SIM Java - NÃO
CONSULTAS					
20- Suporte à consultas através de interface gráfica com	SIM	SIM	SIM	SIM	SIM

SGBDOO / Característica	CACHÉ	JASMINE	OBJECTSTORE	O2	GEMSTONE
usuário.					
21- Suporte à consultas através de linguagens de quarta geração (4GL).	-	NÃO	NÃO	SIM	NÃO
22- Suporte à consultas através de linguagens orientadas a objeto (ex. C++).	-	SIM	SIM	SIM	SIM
MODIFICAÇÃO DO ESQUEMA					
23- Suporte à modificações do esquema do BD através de interface gráfica com usuário (GUI).	SIM	SIM	SIM	SIM	SIM
24- Suporte à modificações do esquema através de linguagens de quarta geração (4GL).	-	NÃO	NÃO	SIM	NÃO
25- Suporte à modificações do esquema através de linguagens orientadas a objeto.	-	SIM	SIM	SIM	SIM Java Smalltalk
FERRAMENTAS CASE					
26- Suportado por ferramentas CASE.	-	SIM	NÃO	SIM	NÃO
ACESSO À OUTROS SGBD´s					
27- Uma aplicação rodando em um BD-OO pode ler dados de outro BD-OO.	-	NÃO	SIM	SIM	NÃO
28- Uma aplicação rodando em um BD-OO pode modificar	-	NÃO	SIM	SIM	NÃO

SGBDOO / Caracteristica	CACHÉ	JASMINE	OBJECTSTORE	O2	GEMSTONE
dados de outro BD-OO.					
29- Uma aplicação rodando em um BD-OO pode ler dados do SGBD-OR ORACLE.	-	SIM	SIM	SIM	SIM
30- Uma aplicação rodando em um BD-OO pode ler dados de outros SGBD´s.	-	SIM OpenIngres Sybase Informix MS-SQL CA-IDMS CA-Datacom DB2,VSAM,RMS	SIM	SIM	SIM Sybase Informix
ARQUITETURA					
31- Suporte à ambiente multi-tarefa mono-usuário.	SIM	SIM	SIM	SIM	SIM
32- Suporte à ambiente multi-usuário.	SIM	SIM	SIM	SIM	SIM
33- Suporte à ambiente cliente-servidor.	SIM	SIM	SIM	SIM	SIM
34- Aplicações podem rodar de forma autônoma (independente do servidor).	-	NÃO	SIM	SIM	NÃO
35- Suporte à transações aninhadas.	-	NÃO	SIM	NÃO	NÃO
36- Suporte à longas transações.	-	NÃO	SIM	NÃO	NÃO
SERVIDOR					
37- Apresenta suporte ao MS-Windows.	SIM	SIM NT	SIM NT e 95	SIM NT	SIM NT
38- Apresenta suporte ao OS/2.	-	NÃO	SIM	NÃO	NÃO
49- Apresenta	-	SIM	SIM	SIM	NÃO

SGBDOO / Caracteristica	CACHÉ	JASMINE	OBJECTSTORE	O2	GEMSTONE
suporte ao SUN OS.					
40- Apresenta suporte ao SUN SOLARIS.	-	SIM	SIM	SIM	SIM
41- Apresenta suporte ao AIX.	-	NÃO	SIM	SIM	SIM
42- Apresenta suporte ao VMS.	-	NÃO	NÃO	NÃO	NÃO
43- Suporte à dados distribuídos em diversos servidores.	-	SIM	SIM	SIM	SIM
CLIENTE					
44- Apresenta suporte ao MS-Windows.	-	SIM NT e 95	SIM NT e 95	SIM NT	SIM NT
45- Apresenta suporte ao Macintosh.	-	NÃO	NÃO	SIM	SIM
46- Apresenta suporte ao SUN OS.	-	SIM	SIM	SIM	NÃO

5 CONCLUSÃO

Este trabalho teve como objetivo principal realizar um estudo comparativo entre Bancos de Dados Orientados a Objetos. Para realizar a comparação foram necessários o estudo e pesquisa dos principais sistemas disponíveis.

Os sistemas gerenciadores de banco de dados orientados a objeto acrescentam persistência e outras características de bancos de dados às linguagens de programação orientadas a objeto. Ao contrário, os modelos de dados objeto-relacionais estendem o modelo de dados relacionais fornecendo um tipo de sistema mais rico, incluindo orientação a objeto e acrescentando estruturas a linguagens de consulta relacionais, como SQL, para tratar os tipos de dados acrescentados. Sistemas de dados de ambos os tipos estão no mercado e o projetista de banco de dados precisa escolher o tipo de sistema apropriado às necessidades da aplicação em que está trabalhando.

Extensões persistentes às linguagens de programação e sistemas objeto-relacionais estão direcionados à diferentes mercados. A natureza declaratória e o poder limitado da linguagem SQL (comparação com uma linguagem de programação) fornecem boa proteção de dados contra erros de programação e tornam otimizações de alto nível, com redução de operações de leitura e escrita, relativamente fáceis. Sistemas objeto-relacionais visam aumentar o poder de expressão do modelo de dados relacional, pelo uso de dados complexos.

O sucesso dos Sistemas de Bancos de Dados Relacionais (SGBD-R) não resulta apenas em um alto nível de independência de dados e um modelo de dados mais simples do que os Sistemas Gerenciadores de Bancos de Dados Orientados a Objetos (SGBD-OO). Seu sucesso se deve também à padronização que sofreram, o que difere dos SGBD-OO por apresentarem apenas os padrões: ODMG-93 e SQL3.

Um dos problemas dos SGBD-OO é a falta de padronização, e os problemas que isso acarreta. Na prática, cada ferramenta utiliza uma linguagem própria para definição do esquema e manipulação do banco de dados. A tentativa da ODMG é realizar a padronização dessa linguagem, através do padrão ODMG, que inclui as linguagens ODL e OQL. No entanto esse padrão não é aceito por todas as ferramentas, e em alguns casos aceito

parcialmente.

Outro grande problema dos SGBD-OO é a mudança de paradigma em relação ao modelo relacional. Considerando a ampla base instalada, número de aplicações e pessoal já treinado para o modelo relacional, a transição para o modelo OO torna-se complicada e envolve alto investimento principalmente no que diz respeito ao treinamento de pessoal.

Com a falta de material referente as características dos Bancos de Dados Orientados a Objetos, fica clara a ausência de informações no que diz respeito às características das ferramentas estudadas.

REFERÊNCIAS BIBLIOGRÁFICAS

(Ali Adams, 2004) Ali Adams - ObjectStore Features - ObjectStore disponível em : <http://www.geocities.com/aliadams/BillingSystem/OS.htm> Acesso em: 01/06/2004

(Cyberdyne, 2004) Cyberdyne - Object Orientation FAQ - Object Store disponível em: <http://www.cyberdyne-object-sys.com/ oofaq/oo-faq-S-8.13.0.5.html> Acesso em: 12/05/2004

(Cyberdyne, 2004a) Cyberdyne - Object Orientation FAQ - Object Store <http://www.cyberdyne-object-sys.com/oofaq/oo-faq-S-8.12.1.3.html> Acesso em: 13/05/2004

(Intersystems, 2004) Intersystems - disponível em: <http://www.intersystems.com.br/cgi-bin/nph-gwcgi?MGWLPN=ISC&wlapp=ISC&guid=4kg76dE6kXUVnKAWn6SEQgGsg51Agy> Acesso: em 01/05/2004

(Intersystems, 2004a) Intersystems - disponível em: <www.intersystems.com.br/cgi-bin/nph-mgwcgi?MGWLPN=ISC&wlapp=ISC&guid=CoThBYPHim7hmxkP9kFyXKf1SM7LsU> Acesso: em 01/05/2004

(Intersystems, 2004b) Intersystems - disponível em: <www.intersystems.com.br/cgi-bin/nph-mgwcgi?MGWLPN=ISC&wlapp=ISC&guid=q3P8KrTsX4Zf3nRppm6wisZr56LxLN> Aceso em: 01/05/2004

(Intersystems, 2004c) Intersystems - disponível em: <www.intersystems.com.br/cgi-bin/nph-mgwcgi?MGWLPN=ISC&wlapp=ISC&guid=MCGzUibkS1KrtJoSAx4rYmqMCpgLe2>

Acesso em: 01/05/2004

(Jasmine, 2004) Jasmine – disponível em:

<http://h21007.www2.hp.com/dspp/mop/mop_partner_product_detail_IDX/1,1331,1045,00.html> Acesso em: 25/05/2004

(Manual do Jasmine, 2004) Manual do Jasmine – disponível em: <http://www.recope.dc.ufscar.br/recope/download/bd/jasminemanual.pdf> Acesso em: 02/05/2004

(NASSU, 1999) Eugênio A. Nassu, Waldemar W. Setzer. **Bancos de Dados Orientados a Objetos**, Edgard Blücher LTDA, 1999

(NAVATHE, 1999) Ramez Elmasri, Shamkant Navathe. **Fundamentals of Database Systems,** Paperback, 1999

(Rubinei, 2004) Rubinei – Encapsulamento disponível em: <http://minerva.ufpel.edu.br/~rubinei/encapsulamento.html> Acesso em: 11/03/2004

(Sampaio, 2004) Sampaio - Banco de Dados Orientado a Objetos disponível em: <http://www.ufpa.br/sampaio/curso_de_sbd/bdoo/apostila/per1.html> Acesso: em 20/02/2004

(Shikida, 2004) Shikida - disponível em: <http://www.npdi.dcc.ufmg.br/publicacoes/shikida01.pdf> Acesso em: 25/05/2004