IMPLEMENTAÇÃO DE UM BANCO DE DADOS ORIENTADO A OBJETOS A PARTIR DE UM MODELO DIMENSIONAL

Moisés Henrique Ramos Pereira[1]
Antônio da Mota Moura Junior [2] **– Orientador**
[1]moiseshrp@gmail.com; [2]amoura@acad.unibh.br
Curso de Ciência da Computação – Uni-BH (www.unibh.br)

Resumo *– Este trabalho aplica parte da metodologia proposta por Borba (2006) para a implementação de modelos dimensionais e implantação de Data Warehouses (DW) em um ambiente de desenvolvimento orientado a objetos. A metodologia em que esse trabalho se baseia apresenta técnicas para definir o modelo de negócio da organização, gerar o modelo dimensional, representar este modelo em um diagrama UML e, por fim, efetivar o seu mapeamento em um banco orientado a objetos pelo padrão ODMG. Conforme as adaptações realizadas para o estudo de caso, este trabalho desenvolve o mapeamento de um modelo de negócio real em diagrama de classes UML, implementando o banco diretamente em um framework de persistência de objetos integrado a uma linguagem OO comercialmente difundida.*

Palavras-chave *– Data warehouse. Modelo dimensional UML Modelagem. Banco de dados orientado a objetos. Metodologia.*

Abstract *– This work apply the methodology proposed by Borba (2006) for implementation of dimensional models and deployment of Data Warehouses (DW) in an oriented to objects development environment. The methodology on which this work is based presents techniques to define the business model of the organization, creating the dimensional model, representing it in UML diagram, and finally accomplish its mapping in a database oriented to objects by ODMG standard. As the adjustments made to the case study, this paper develops the mapping of an existing real business model in UML class diagram, implementing the database directly in a persistence framework for objects integrated into a language oriented to objects commercially distributed.*

Keywords *– Data warehouse. Dimensional model. UML. Modeling. Database oriented to objects. Methodology.*

1 INTRODUÇÃO

Neste trabalho são descritas e aplicadas algumas etapas da metodologia proposta no ano de 2006 por Sueli de Fátima Poppi Borba, na Universidade Federal de Santa Catarina, em tese de doutorado referente à modelagem e implementação de ambientes de *Data Warehouse* (DW) sob os conceitos do paradigma da Orientação a Objetos (OO), adaptando-a em um estudo de caso para um modelo dimensional conhecido.

Na era da sociedade do conhecimento, o sucesso das organizações está diretamente ligado ao acesso privilegiado à informação em tempo e em quantidade satisfatórios dentro da dinâmica de negócio (BORBA, 2006). Com a crescente necessidade de analisar o desempenho organizacional, favorecendo o processo de tomada de decisão, percebem-se no mercado diversas soluções de *Data Warehouses* (DW) que disponibilizam, de forma mais interessante para a empresa, as informações contidas em grandes volumes de dados distribuídos em diferentes sistemas.

Da mesma forma que as organizações precisam de agilidade para obter informações estratégicas existentes nos sistemas, é necessário que o processo de implementação do DW seja bem definido, rápido e barato, o que pode ser determinante dentro da conjuntura comercial. Esse é o problema apontado por Klein, Campos & Tanaka (1999), pois existe carência por uma metodologia bem definida para a implantação de um DW. Assim, alguns aspectos devem ser discutidos como o suporte à manipulação de dados e custos de implementação, pois em um DW é necessário modelar todos os agrupamentos de informações da organização, gerando grande quantidade de dados muito complexos para o armazenamento físico.

Partindo do pressuposto que a orientação a objetos poderia promover agilidade no desenvolvimento de bancos de dados como ocorre atualmente no mercado de

desenvolvimento de software, seria possível implementar soluções de DW seguindo uma metodologia de desenvolvimento que embarcasse este paradigma, aproveitando-se das tecnologias oferecidas pelas aplicações de modelagem OO para melhoria da documentação do DW como, por exemplo, a abstração oferecida pela UML (*Unified Modeling Language*) frente ao cliente.

O conceito de DW foi proposto em meados dos anos 80 pelos idealizadores Willian Inmon (INMON, 1992) e Ralph Kimball (KIMBALL, 1998) que descrevem os processos de análise de negócio para suprir as necessidades empresariais através de sistemas de apoio à tomada de decisão, permitindo que informações estratégicas sejam obtidas dos próprios dados existentes nas empresas. Já a teoria OO criada na década de 60 por Kristen Nygaard e Ole-Johan Dahl tem como objetivo oferecer uma nova visão de modelagem e programação, pois era notável que as pessoas pensavam em entidades e responsabilidades de negócio como um todo encapsulado, inferindo a partir daí o conceito de objeto.

Dessa forma, durante toda a história do pensamento científico para a resolução de problemas, diversos modelos foram definidos para documentar sistemas e metodologias de desenvolvimento, considerando as incertezas do pensamento humano e as abstrações de uma realidade para consenso de todos os envolvidos. Sobre as necessidades de diferencial competitivo promovido pelas soluções de DW e as facilidades geradas pelo paradigma OO no desenvolvimento de aplicações, percebe-se a relevância em estudar formas de agregar os dois conceitos a fim de extrair os resultados mais eficazes frente às necessidades de melhor custo-benefício nas análises de negócio.

O interesse deste trabalho é aplicar a metodologia de Borba (2006) para implementar um determinado modelo dimensional com o uso de tecnologias altamente difundidas no mercado que utilizam o paradigma OO, bem como apresentar a possibilidade do uso comercial de bancos de dados OO em soluções de DW.

Este trabalho está dividido em cinco seções, incluindo a introdução. A segunda seção aborda uma revisão da literatura, refletindo alguns dos problemas e soluções existentes quanto à modelagem e implementação de banco de dados, mais especificamente de bancos DW, e um breve histórico do paradigma OO.

A terceira seção apresenta, de maneira mais abrangente e conceitual, alguns modelos de dados conhecidos, incluindo os conceitos de DW e modelagem dimensional. Além disso, algumas técnicas são descritas para exemplificar as metodologias propostas para o desenvolvimento de DW. Já a seção 4 é dedicada à descrição mais detalhada da metodologia de Borba (2006) aplicada neste trabalho, aprofundando as etapas sobre o mapeamento do modelo dimensional utilizado para UML e sua implementação física por uma linguagem de programação OO. Este capítulo apresenta as ferramentas utilizadas no trabalho e os conceitos definidos no próprio modelo dimensional usado no estudo de caso.

Na seção final existem algumas reflexões acerca do uso da modelagem OO em ambientes DW, sugerindo algumas linhas de pesquisa para trabalhos futuros.

2 REVISÃO DE LITERATURA

Nas décadas de 80 e 90, o banco de dados era definido como uma única fonte de dados para todo o processamento das aplicações existentes no meio científico. Com a evolução tecnológica, o computador tornou-se extremamente aplicável no mercado comercial, suprindo as instituições empresariais com dados que favoreciam o ambiente de negócio, transformando os recursos computacionais de simples instrumentos de cálculo para verdadeiras ferramentas de análise de negócio com alto poder competitivo (VIDOTTI, 2001). Neste momento, a comunidade científica e as organizações passam a perceber a necessidade por alguma metodologia bem definida para a modelagem de bancos de dados e padronizar o

processo de desenvolvimento.

Para Borba (2006), desde os anos 90 que a necessidade de obtenção de dados relevantes, principalmente em tempo satisfatório imposto pela dinâmica do mercado, vem se apresentando cada vez mais determinante para o sucesso das organizações. Com a crescente necessidade de gerenciar o desempenho organizacional, favorecendo o processo de tomada de decisão, percebem-se no mercado diversas soluções de Data Warehouses (DW) que disponibilizam, de forma mais interessante para a empresa, as informações contidas em grandes volumes de dados (BORBA, 2006).

Em 1992, Inmon definia um DW como uma coleção de dados orientada por assunto, integrada, variável no tempo e não-volátil, usada no apoio aos processos de tomada de decisão gerenciais. Para ele, o DW é uma tecnologia voltada para o estudo de técnicas e ferramentas utilizadas em aplicações que envolvam análise intensiva de dados e integração de fontes de dados heterogêneas, de forma a prover agilidade e flexibilidade na gerência, manutenção e acesso a estes dados (INMON, 2005).

Como qualquer solução tecnológica adotada, inicialmente não se dava uma importância emergencial à definição de um processo bem fundamentado para o desenvolvimento de DW. Conforme Klein, Campos & Tanaka (1999), não existe uma metodologia sólida para implantação de ambientes DW, tendo somente guias que direcionam os arquitetos para obtenção do melhor resultado alcançável, indicando então um dos seus principais problemas de implantação. Para facilitar a implementação deste tipo de abordagem, Kimball (1998), como poucos autores que aprofundam em um modelo lógico, propõe nove passos simples para nortear a implementação de um DW, tendo como foco o modelo estrela. Ele descreve que a construção efetiva de um DW se inicia pela modelagem dimensional, uma técnica de projeto lógico de banco de dados que busca apresentar os dados em um formato que seja intuitivo e com alto desempenho para o usuário final.

Esse enfoque para a OO, paradigma de programação criada nos anos 60, baseia-se no fato de que os objetos oferecem maior abstração para a modelagem de dados muito complexos, pois eles representam melhor as entidades do mundo real, incorporando seus atributos e operações (BOSCARIOLI *et al*, 2006). Como ocorre na modelagem dimensional, é necessário encontrar todos os objetos que compõem o DW, pois os objetos podem ser estendidos para atender às novas demandas de mercado.

Outra vantagem do modelo orientado a objetos percebe-se na possibilidade de utilizar o mesmo modelo no projeto conceitual, onde os objetos são construídos; no projeto lógico, estendendo-se esses objetos conforme a tecnologia; e no físico, quando detalhes da ferramenta utilizada na implementação são incorporados ao modelo (BEZERRA, 2003).

Como os objetos representam entidades do mundo real mais fielmente, os valores de seus atributos podem ser muito complexos, podendo representar outros objetos. Já nos bancos de dados relacionais ocorrem incompatibilidades referentes a esse tipo de complexidade, pois existe uma grande separação entre os dados e suas operações (BOSCARIOLI *et al*, 2006).

Dessa forma, apesar do sucesso dos sistemas de gerenciamento de dados relacionais, Klein, Campos & Tanaka (1999) afirmam que a manipulação de informações complexas proporcionada pelas diversas aplicações, inclusive a tecnologia de DW, exigiram novas soluções de gerenciamento de dados como os sistemas de gerenciamento de banco de dados orientados a objetos (SGBDOO) e objeto-relacionais (SGBDOR). Dentro da modelagem de um DW, o esquema estrela se projeta como a principal forma de representação da dimensionalidade de negócio e, ao introduzir as abstrações OO, esses sistemas de gerenciamento permitem que as tabelas sejam melhor representadas. Neste

contexto surge a proposta de modelagem OO por Borba (2006) para bancos DW que aplica integralmente os conceitos da OO sem nenhum tipo de tecnologia intermediária.

Nos últimos anos ocorreram alguns usos de BDOO, como relata Rosemberg (2008). Um deles é o db4o. A Indra Sistemas, grupo espanhol líder em desenvolvimento de softwares contratada recentemente para desenvolver um centro de controle do sistema de trens bala da Espanha, utilizou o db4o como base de dados de tempo real para controlar o tráfego, gerando capacidade de processamento em torno de 200 mil objetos por segundo com pouco uso de memória (ROSEMBERG, 2008). De acordo com José Miguel Rubio Sánchez, gerente técnico da Indra, o maior benefício observado se mostrou na facilidade em trabalhar diretamente com objetos em consultas sem transformar os dados.

3 MODELOS DE DADOS E A ORIENTAÇÃO A OBJETOS

Nas últimas décadas, a computação vem evoluindo no que se refere às metodologias e tecnologias de modelagem e armazenamento de dados. Uma das tecnologias que contribuíram para essa evolução são os diversos tipos de bancos de dados que armazenam grande quantidade de dados em um curto espaço de tempo. Esta realidade passou a dificultar aos envolvidos identificar e analisar informações relevantes ali presentes para o negócio da organização. Dessa forma, o modelo de dados deve fornecer uma visão precisa e objetiva de como os dados serão armazenados para favorecer o entendimento de conceitos, especificações e regras durante o projeto de banco de dados (BORBA, 2006).

Um modelo de dados é uma coleção de conceitos que podem ser usados para descrever um conjunto de dados e as operações para manipulá-los. Para Elmasri & Navathe (2002), os modelos de dados podem ser classificados em modelo conceitual, lógico e físico, conforme a etapa de desenvolvimento do projeto do banco em que o modelo é utilizado.

O modelo conceitual representa as entidades e seus relacionamentos conforme observadas e expostas no mundo real durante a fase de análise entre os envolvidos da organização, desconsiderando detalhes impostos pela tecnologia, metodologias ou dispositivos físicos. No modelo lógico, construído a partir do modelo conceitual, as entidades mapeadas são representadas conforme um padrão mais técnico e formal, considerando limitações tecnológicas e decisões de projeto conforme a visão do usuário do SGBD, mas ainda se abstém do ambiente físico onde os dados serão armazenados no computador. Já no modelo físico, detalha-se os tipos de campos, o acesso à memória e demais itens para o SGBD, pois os dados são representados conforme o ambiente físico.

3.1 DATA WAREHOUSE E O MODELO DIMENSIONAL

Nesta seção, serão apresentadas algumas definições referentes à *Data Warehouse* (DW) e alguns elementos importantes do modelo dimensional para uma discussão das vantagens de um mapeamento orientado a objetos.

Encontra-se na literatura a caracterização de DW como "*(...) uma coleção de dados orientada a assuntos, integrada, não-volátil e variável no tempo em suporte às decisões gerenciais*" (INMON, 2005: 29). Borba (2006) complementa que um DW é uma importante ferramenta para análise e acesso mais global aos dados advindos de diversas bases de dados autônomas e heterogêneas, pois o DW possui uma cópia dos dados extraídos de um ou mais sistemas de produção na fase de carregamento (KIMBALL, 1998). O conceito de *Data Mart* é parecido com as definições acerca de um DW, mas focado sobre um determinado assunto, ou seja, *Data Mart* é um DW departamental que possui um assunto definido. Na abordagem de Kimball (2002), *Data Marts* são criados e, posteriormente, o DW é construído a partir da correlação entre eles, mas para Inmon (2005), o DW é construído para depois os assuntos serem

identificados como *Data Marts*. Como não é foco de análise deste trabalho, o detalhamento dessas implementações não será abordado.

Como visto no capítulo 2, os bancos de dados operacionais são baseados nas aplicações da empresa, tendo desempenho analisado conforme o sistema ao qual está associado. Já um DW é baseado em assunto, ou seja, os dados são organizados conforme os principais assuntos do negócio da empresa.

Um DW também é não-volátil, ou seja, os dados não são deletados e nem atualizados livremente como nas bases operacionais. Essa propriedade permite melhor performance do DW, pois o ambiente executará as transações analíticas sem perder tempo de processamento no controle de concorrência como nos bancos de dados operacionais acessados por diversos usuários que podem executar diferentes tarefas sobre os mesmos dados (KIMBALL, 1998). Em determinados períodos, os dados dos aplicativos em produção são inseridos e, a partir daí, serão apenas consultados durante as demandas de negócio. Conforme Kimball (1998), o primeiro estágio do DW onde estes dados são carregados para serem tratados em processos ETL (*extract-transformation-load*) é conhecido como *data staging area* e esse controle de carga é feito fora do expediente das consultas analíticas.

A integridade dos dados é outra importante característica de um ambiente DW, pois os dados advindos das diversas fontes operacionais podem possuir diferentes formatos e valores, em conformidade às suas respectivas aplicações. Para Inmon (2005), a integração é o processo mais importante na manutenção de um DW e como os dados são alimentados a partir de múltiplas fontes, eles devem ser manipulados a fim de terem uma única imagem corporativa.

O DW é variante no tempo, ou seja, os dados nesse ambiente representam uma faixa extensa de tempo, podendo armazenar, por exemplo, as realidades e instâncias de um determinado dado concebido há vários anos. Conforme Vidotti (2001), essa característica do DW é garantida pela implementação de uma dimensão de tempo que, no momento em que os dados são inseridos no DW, receberá em seu atributo-chave o valor da data desta atualização, permitindo assim a redundância dos dados que estão diferenciados pela data de alteração no ambiente operacional.

Voltando-se para a implementação de ambientes DW depois de apresentar suas características fundamentais, são descritas, a seguir, duas metodologias que possuem muitos elementos em comum, mas seguem paradigmas diferentes em diversas fases: o modelo de Kimball e o modelo proposto por Borba (2006).

Frente à realidade do mercado no gerenciamento de informações estratégicas, Kimball *et al* (1998) promoveu, desde meados dos anos 80, um modelo contendo as principais diretrizes de desenvolvimento do DW, distribuídas em 9 fases. Esse modelo afirma que os projetos do DW devem incidir sobre as necessidades do negócio e que os dados devem ser unidimensionais quando apresentados aos clientes. Cada projeto no DW deverá ter um ciclo finito com início e fim bem definidos. A sua primeira fase consiste do Planejamento de Projeto, seguida pelas fases de Definição dos Requisitos de Negócio, *Design* Técnico de Arquitetura, Seleção e Instalação de Produtos, Modelagem Dimensional, *Design* Físico, *Design* e Desenvolvimento da *Data Staging Area*, Especificação e Implementação da Aplicação Analítica, Implantação, Manutenção e Crescimento (KIMBALL, 2002). Este modelo foca o desenvolvimento do DW para um ambiente de bancos de dados relacionais.

A Fig. 1 apresenta o diagrama de ciclo de vida do modelo de Kimball & Ross (2002), indicando a distribuição das fases seqüenciais, dependências e fases concorrentes em um projeto de desenvolvimento de DW. Kimball & Ross (2002) alertam que o diagrama não reflete uma cronologia absoluta entre as fases e nem o verdadeiro tempo de duração de cada uma delas. É um modelo que pode ser considerado como um simples roteiro que facilita a equipe de desenvolvimento no projeto

e que serve tanto para os projetos-assunto de Data Mart como para todo o ambiente DW.

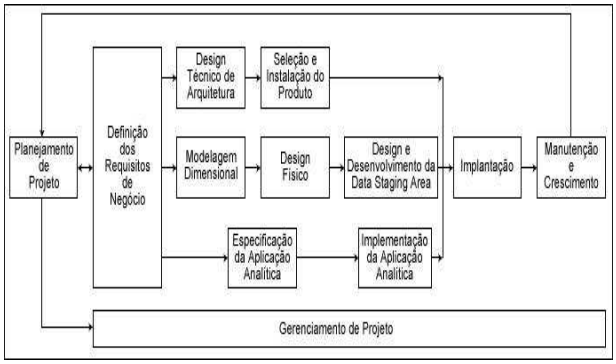

Figura 1 – Diagrama de ciclo de vida dimensional de negócio – Modelo de Kimball. Fonte: KIMBALL & ROSS, 2002.

O modelo proposto por Borba (2006) promove a implementação do DW em BDOO, trabalhando com os conceitos do paradigma OO em várias de suas fases. Estes conceitos, quando representados pela UML, oferecem uma modelagem adequada das características de um DW, desde a fase de levantamento de requisitos até a implementação. Conforme Borba & Morales (2006), as 5 etapas que compõe este modelo são, nesta ordem: a Definição do Modelo do Negócio; Geração do Modelo Dimensional; Geração do Modelo Dimensional Representado pela UML; Mapeamento do Modelo UML para BDOO. Este modelo será devidamente discutido na próxima seção, pois trata-se de uma parte fundamental para a realização deste trabalho.

Uma das técnicas de modelagem mais utilizadas atualmente para a implementação de um DW é a modelagem dimensional, um tipo de modelo lógico que reflete os requisitos de negócio em uma estrutura de cubo de dados. Conforme Kimball & Ross (2002), para construir o modelo dimensional é necessário identificar o processo de negócio a ser modelado, descrever o nível de granularidade e mapear as dimensões e fatos do negócio. A partir disso, é modelada uma tabela central, a tabela fato, conectada a outras tabelas que armazenarão as dimensões. Geralmente, somente a tabela fato é normalizada, não necessariamente todas as tabelas como no modelo ER.

Barbieri (2001) complementa que a modelagem dimensional representa a informação como interseções das várias dimensões do negócio para que o usuário veja os dados conforme o entendimento que ele tem dessa realidade. O modelo dimensional possui uma estrutura mais assimétrica, diferentemente do modelo ER, mapeando uma tabela central chamada de tabela fato com diversos *joins* com outras tabelas secundárias, as dimensões (KIMBALL, 1998). No estudo da modelagem dimensional, mostra-se necessário entender os conceitos de tabelas fatos e dimensões.

As tabelas fatos, também conhecidas como cubos de decisão ou tabelas de fatos, são representantes das transações ou ocorrências existentes no negócio. De forma mais específica, Kimball (1998) afirma que as tabelas de fatos armazenam as medições numéricas de desempenho extraídas dos fatos relevantes do negócio e as combinações entre as dimensões ali presentes. Para Borba (2006), em muitos casos, os fatos não são conhecidos e, geralmente, representam valores numéricos, também chamados de métricas, quantificados em medidas conforme uma realidade que é determinada pelos níveis de dimensão.

As dimensões representam os diversos tipos de visões que os usuários poderão ter do negócio. Este tipo de tabela armazena as descrições das dimensões de negócio, de preferência que sejam modeladas em campos textuais, e são utilizadas como elementos condicionais que restringem ou formam os cabeçalhos das consultas analíticas do usuário (KIMBALL, 1998).

No modelo dimensional, o mapeamento dos relacionamentos é feito sob uma lógica diferente da aplicada ao modelo ER. As dimensões fazem relacionamentos de 1:n com a tabela de fatos, cujo lado 1 está na dimensão e o lado n na tabela fato, representando que uma instância de uma determinada dimensão pode estar em uma ou mais instâncias dos fatos ocorridos no negócio. Em muitos modelos dimensionais o lado n de relacionamentos é descrito por um asterisco, notação muito encontrada também em modelos orientados a

objetos. Conforme Borba (2006), enquanto os relacionamentos entre as entidades são mapeados de forma explícita no modelo ER, no modelo dimensional os relacionamentos representam, implicitamente, as interseções entre a tabela de fatos e suas dimensões, dentro do ideal imposto pelo cubo de dados. A Fig. 2 abaixo apresenta um esboço de um modelo dimensional com alguns dos elementos até agora descritos.

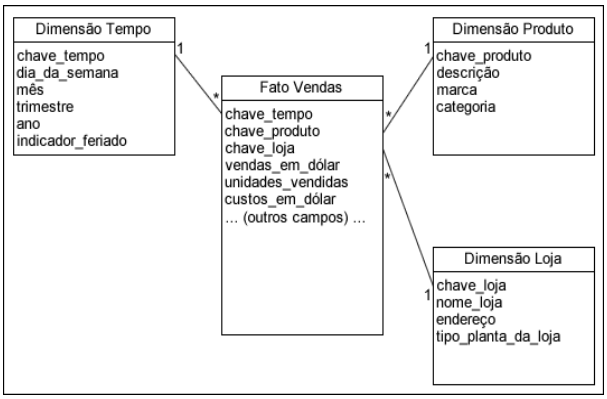

Figura 2 – Esboço de um modelo dimensional. Fonte: Adaptado de KIMBALL, 1998: 10.

Os campos das tabelas desenhadas para o modelo dimensional são extremamente importantes, pois, dependendo de como são classificados no modelo, eles refletirão alguma característica e informação importantes para os processos analíticos de tomada de decisão, seja para armazenar um fato do negócio ou uma descrição textual de uma dimensão. De acordo com Kimball (1998), essa classificação é denominada granularidade da tabela. Toda tabela de fatos possui como atributo-chave a combinação das chaves primárias das tabelas dimensionais as quais está ligada.

Existem diversos modelos dimensionais, mas os mais utilizados academicamente e comercialmente são o modelo Estrela e o modelo Floco de Neve, conhecidos na literatura, respectivamente, por *Star Schema* e *Snowflake Schema* (BORBA, 2006). A metodologia discutida na seção 4 aborda detalhes do mapeamento para cada um destes modelos, enfatizando mais o modelo Estrela, pois trata-se do formato do modelo dimensional utilizado.

4 MODELO DIMENSIONAL EM AMBIENTE DE PERSISTÊNCIA DE OBJETOS

Este capítulo descreve a metodologia proposta por Borba (2006), aprofundando nas etapas utilizadas para o modelo de desenvolvimento deste trabalho. Além disso, apresenta o modelo referente ao Crédito Sem Barreiras (CSB) da Vivo, pois o foco é a modelagem para um ambiente de persistência de objetos e não as áreas da análise de requisitos e gestão de negócio, pré-requisitos da modelagem dimensional. E, por fim, esta seção termina descrevendo o mapeamento do modelo dimensional CSB em diagrama de classes UML e a sua implementação em um ambiente OO.

Partindo da análise entre as diversas metodologias divulgadas ao longo dos anos para implementar o modelo dimensional na construção de um DW, Borba (2006) visa atender a todos os requisitos analisados em seu modelo. Neste trabalho, a implementação foi realizada, utilizando a linguagem orientada a objetos Java, associada ao framework de persistência de objetos db4o.

A primeira etapa proposta por Borba (2006) consiste na definição do modelo de negócio com análise de requisitos junto ao cliente, a fim de contemplar as decisões das áreas a serem atendidas pela solução de DW modelada. Conforme Borba & Morales (2006), neste momento, os detalhes do negócio que serão base no desenvolvimento do projeto são analisados e traduzidos para o ambiente operacional adotado através de um modelo ER, documentando a finalidade do projeto e os recursos técnicos que serão utilizados.

Logo a seguir, é realizada a geração do modelo multidimensional, ou seja, o modelo de negócio identificado. Nesta etapa são identificados os sistemas e bases de dados operacionais, inclusive suas funcionalidades dentro deste ambiente, para mapear o modelo dimensional. Borba (2006) enfatiza que o modelo ER desenhado na etapa anterior deve ser minuciosamente analisado para que possam ser extraídas as dimensões que fazem parte do

negócio, podendo indicar dimensões implícitas a serem modeladas. A granularidade das entidades também merece atenção, pois a partir desse estudo serão mapeados os atributos das dimensões e os fatos do negócio a fim de definir tabelas de fatos e métricas de derivação. É necessário, durante a análise, definir uma dimensão Tempo, pois o DW promove a equivalência de tempo sob a visão do negócio, agregando dados por um determinado tipo de período, conforme a necessidade da disposição dos dados (BORBA, 2006).

As próximas etapas da metodologia abordam a representação do modelo dimensional em diagrama de classes UML, o mapeamento deste diagrama para um BDOO, e, finalmente, a sua implementação. A Fig. 3 demonstra a ordem de execução destas etapas e um esboço de toda a metodologia de Borba (2006).

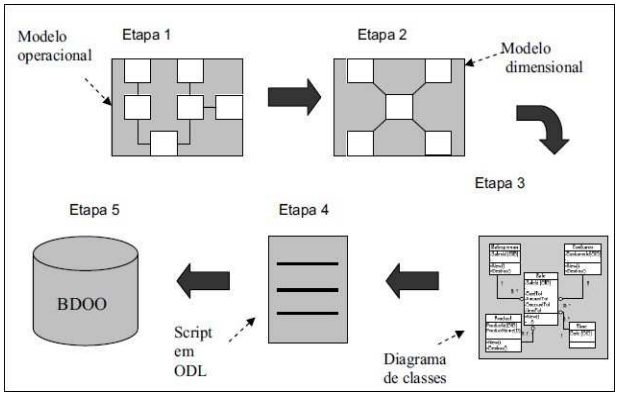

Figura 3. Resumo da metodologia de Borba.
Fonte: BORBA & MORALES (2006)

Estas últimas etapas são descritas nas próximas seções, utilizando um modelo dimensional real já definido. Segue, antes disso, uma breve descrição das ferramentas utilizadas para a implementação do modelo em BDOO para, além de fundamentar, exemplificar a aplicação da metodologia de Borba (2006) sobre modelos de dados reais.

4.1 AS FERRAMENTAS UTILIZADAS

Para representar o modelo dimensional em diagrama de classes UML foi utilizado o software Jude Community 3.2.1, disponibilizado para download no site oficial http://jude.change-vision.com. A codificação das classes mapeadas nesta fase foi realizada em linguagem de programação Java, utilizando-se o ambiente de desenvolvimento Eclipse, versão 3.2, integrado ao framework de persistência de objetos db4o na versão 7.4, disponíveis para download, respectivamente, em seus endereços http://www.eclipse.org e http://www.db4o.com. É necessário instalar plug-ins do bd4o no Eclipse para a integração. Estes podem ser baixados no site oficial citado do db4o. Para implementar o teste de execução realizado e descrito na última seção, foi utilizado o Oracle Express 10g para simular um ambiente operacional.

Conforme Paterson *et al* (2006), o db4o é um tipo de BDOO de código aberto que permite aos desenvolvedores de linguagens de programação OO reduzir consideravelmente os custos envolvidos no desenvolvimento. Segundo o site oficial (DB4OBJECTS, 2008), o db4o executa consultas até 44 vezes mais rápido que algumas soluções conhecidas no mercado como o hibernate que apenas oferece um mapeamento de aplicações OO para bancos de dados relacionais e não persistência direta de objetos.

A Fig. 4 abaixo ilustra um comparativo entre as técnicas em bancos de dados relacionais e o uso do db4o para a implementação dos objetos modelados. Observa-se que existem persistência e mapeamento diretos dos objetos no BDOO, facilitando a consulta a esses objetos, diferente de um SGBD relacional associado às técnicas de mapeamento objeto-relacional que implementam fisicamente os objetos modelados em diversas tabelas, projetando maior complexidade na manipulação dos mesmos.

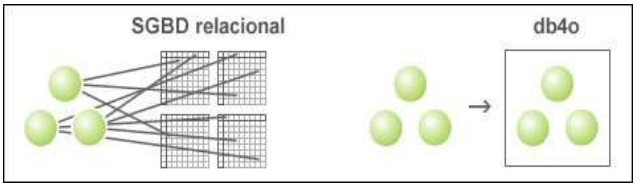

Figura 4. Forma de armazenamento de objetos de um SGBD relacional e no db4o.
Fonte: DB4OBJECTS, 2008.

4.2 MODELO DIMENSIONAL UTILIZADO

O modelo utilizado para a aplicação da metodologia de Borba (2006) trata-se do modelo dimensional do serviço Créditos Sem Barreiras (CSB), disponibilizado pela operadora Vivo em Minas Gerais, e encontra-se implementado no *Data Mart* de Vendas. Este *Data Mart* fornece informações referentes ao gerenciamento de vendas da Vivo - MG e armazena os dados das ativações de contratos de celulares de planos pré e pós-pagos, das vendas de produtos como aparelhos e chips, das ativações de cartões e o acompanhamento entre estas vendas e metas.

O serviço de Crédito sem Barreiras é uma forma de venda de recarga programada da Vivo - MG. O sistema origem permite que o cliente com um celular de plano pós-pago agende a compra de créditos pré-pagos para outro celular designado por ele. Os usuários do DW acompanham a movimentação da carteira, adesões e cancelamentos referentes ao serviço. Existem análises por clientes pós-pagos distintos que aderiram ao CSB e também por agendamentos. Informações mais pontuais como os números de celulares pós e pré-pagos, valores de crédito, canal de compra e funcionário que efetuou a venda são extraídas diretamente das dimensões do DW, onde os dados são tratados e manipulados conforme as demandas de negócio. O modelo dimensional referente ao CSB está representado na Fig. 5.

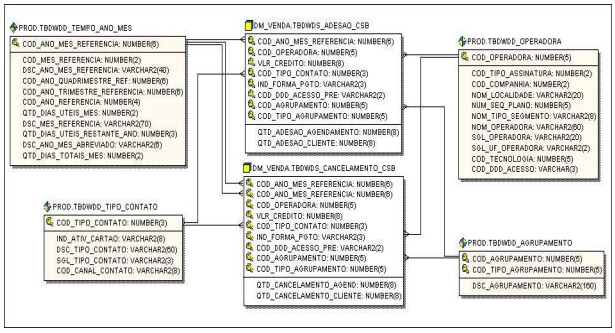

Figura 5. Modelo dimensional CSB.
Fonte: Adaptado de Core Synesis, 2008.

Conforme a Fig. 5 acima, o modelo dimensional CSB utilizado possui duas tabelas de fatos conectadas a outras quatro tabelas de dimensão. Este modelo dimensional agrupa dois modelos representados pelas tabelas de fatos ADESAO_CSB e CANCELAMENTO_CSB que armazenam informações sobre as adesões dos clientes ao serviço e os cancelamentos realizados, respectivamente. As outras tabelas TEMPO_ANO_MES, TIPO_CONTATO, OPERADORA e AGRUPAMENTO referem-se, assim, à dimensão de tempo no negócio, por qual tipo de contato o cliente solicitou sua adesão ou cancelamento do serviço CSB, a qual operadora o cliente pertence e os agrupamentos de clientes por alguma característica, serviço ou planos em comum mapeados na base de dados.

Os atributos-chave das tabelas de fatos são compostos por referências ou Foreign Keys das chaves primárias das tabelas dimensionais. Quando necessário, como neste modelo, outros atributos relevantes de um fato de negócio podem também fazer parte do grupo de atributos-chave das tabelas fato a fim de melhorar a qualidade de resposta nas consultas analíticas. Conforme Machado & Abreu (2004), os relacionamentos entre as tabelas estão representados logicamente por estas *Foreign Keys* e, como todo modelo que possui configuração baseada no ER, para cada valor atômico presente nestes campos das tabelas fatos existe correspondente de mesmo valor no atributo-chave da dimensão, permitindo assim buscar as devidas associações no conjunto de resposta solicitado pelo usuário.

De forma mais geral, este modelo dimensional agrupa as semânticas e fatos de negócio referentes às informações de adesões e cancelamentos do CSB (Crédito Sem Barreiras) solicitados por um cliente de uma determinada operadora através de um tipo de contato em um período de tempo.

4.3 MAPEAMENTO DO MODELO DIMENSIONAL EM UML

Com o modelo dimensional definido por meio de um esquema estrutural como, por exemplo, o *Star Schema* ou *Snowflake Schema*, basta classificar as entidades e suas hierarquias conforme as regras do negócio a fim de mapeá-

las sob a perspectiva OO da UML, permitindo que os dados representados sejam agregados posteriormente (BORBA, 2006). Dessa forma, como visto nas definições acerca de classes, as tabelas dimensionais são modeladas como classes dimensionais no respectivo diagrama da UML e as tabelas de fatos são representadas por classes de fatos. Além de mapear as classes e seus atributos para representar as tabelas do modelo, faz-se necessário também traçar os relacionamentos e as multiplicidades entre as classes fato e suas dimensões.

No mapeamento das tabelas do modelo dimensional em classes, além de seguir as notações de modelagem da UML, OMG e ODMG, deve-se aplicar os padrões definidos no DW pela organização. Para auxiliar as equipes envolvidas na administração e desenvolvimento do DW, é interessante que aja padronização para identificar, explicitamente, as tabelas de fatos e as dimensões em consultas OLAP (*On-line Analytical Processing*) e rotinas ETL.

Conforme Borba (2006), uma das formas mais apropriadas para representar os relacionamentos entre as classes de fatos e as classes das dimensões é por meio de agregações compartilhadas, pois, pela semântica dos conceitos de DW em modelos dimensionais, a tabela fato agrega as dimensões. A agregação representa que um fato de negócio é constituído de diversas partes, as dimensões, ou seja, as dimensões do negócio fazem parte de um fato de negócio. Somente para exemplificar, pois em ambientes DW não existe formalmente exclusão e alteração de dados, a exclusão de um fato não implica que as dimensões que o constituíam devam ser excluídas, demonstrando que as informações referentes às dimensões são independentes dos fatos ocorridos.

Após incluir as agregações, as multiplicidades devem ser mapeadas para detalhar mais o diagrama. Nos relacionamentos entre as dimensões e a fato, a extremidade referente às dimensões possui multiplicidade 0..1, pois estas armazenam os dados em mais baixo nível; e a extremidade junto à fato possui multiplicidade 1..*, pois as dimensões estão em diversas instâncias de fatos (BORBA, 2006).

Para aplicar esta etapa da metodologia, os mapeamentos acima serão descritos logo a seguir. Tendo como base o modelo dimensional CSB e os padrões considerados, foram modeladas as classes das dimensões TempoAnoMes, TipoContato, Operadora e Agrupamento que correspondem às tabelas dimensionais de mesmo nome. As classes fato AdesaoCSB e CancelamentoCSB foram modeladas a partir das tabelas de fato ADESAO_CSB e CANCELAMENTO_CSB do modelo dimensional CSB.

Na construção do diagrama, para implementar o último tipo de notação acima, ou seja, a identificação das classes de fatos e dimensionais, as abreviações CS e CD foram concatenadas aos nomes das classes para identificar, respectivamente, as classes de fatos sumarizadas e as dimensões na manipulação dos dados pelo desenvolvedor. As classes de fatos do diagrama dimensional, agora CSAdesaoCSB e CSCancelamentoCSB, foram generalizadas para uma classe CSCSB, pois elas apresentam atributos em comum, além de possuírem dados referentes ao mesmo tipo de serviço, no caso o Crédito Sem Barreiras.

Conforme Cattell *et al* (2000), todo objeto em uma base de dados possui um identificador único OID que deve ser modelado nos diagramas de dados. Borba (2006) confirma que os identificadores devem ser representados de forma explícita, mas somente para classes de dimensão, colocando-se a restrição {OID} no atributo definido para cada classe. Já os descritores são os atributos representativos de cada classe dimensional e devem ser textuais, preferencialmente, por serem campos de referência para o usuário, modelados com a restrição {D} no diagrama de classes UML (BORBA, 2006). Um padrão sugerido neste trabalho para nomear os OIDs é concatenar o termo OID ao nome da classe sem carregar a sigla CD. Conforme as análises e padrões descritos, o diagrama de classes CSB é apresentado na Fig. 6.

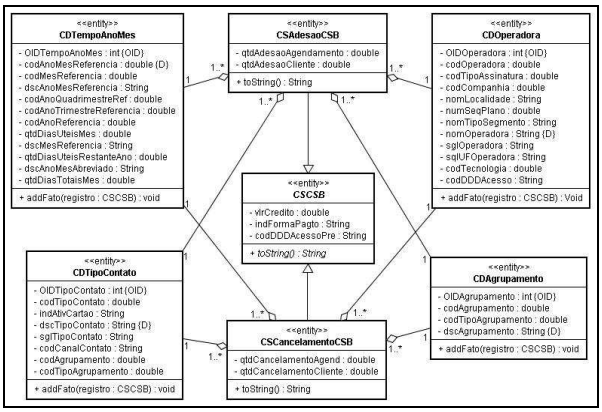

Figura 6 – Diagrama de classes CSB.

Percebe-se na Fig. 6 que as classes fato CSAdesaoCSB e CSCancelamentoCSB estão conectadas diretamente às classes de dimensões CDTipoContato, CDOperadora, CDTempoAnoMes e CDAgrupamento, assim representando os mesmos dois modelos dimensionais referentes às adesões e cancelamentos do serviço CSB. Os descritores escolhidos no diagrama são, justamente, os atributos mais significativos e que possuem o tipo *String* por serem atributos textuais.

Analisando Rumbaugh, Jacobson & Booch (2004), classe pode possuir métodos que retornam e alteram diretamente os dados de seus atributos. Estes métodos são representados pelas operações *gets* e *sets*, mas, por padrão, não é necessário mapeá-los explicitamente no diagrama. Foram representadas somente as operações *addFato* nas dimensões e *toString* para as classes fato que correspondem, respectivamente, aos métodos para adicionar os fatos que cada dimensão participa e a descrição dos valores internos dos fatos ocorridos, métodos dos quais serão implementados na próxima etapa da metodologia. A classe CSCSB criada para generalizar as classes fato foi modelada como classe abstrata, pois mesmo contendo atributos e operações próprios, não será instanciada para armazenar dados. Para implementar o polimorfismo no diagrama, a operação *toString* definida nas classes de fatos foi mapeada também para esta classe CSCSB, pois mesmo que esse diagrama seja muito usado no nível conceitual, pode-se adicionar detalhes de implementação (BEZERRA, 2003).

4.4 IMPLEMENTAÇÃO DO DIAGRAMA UML NO BDOO

Nesta seção está descrito o processo de implementação da etapa da metodologia de Borba (2006) que visa codificar o script para um determinado BDOO a partir do diagrama de classes obtido anteriormente. Para este trabalho, houve adaptações para a geração do código referente à persistência de objetos que, na verdade, utiliza uma linguagem de programação OO para desenvolver diretamente as classes UML modeladas. Borba & Morales (2006) comentam que o padrão ODMG oferece a linguagem OQL, mas ela não oferece interoperabilidade e trata-se de um padrão de codificação que será depois implementado, independente de linguagem de programação. Já neste trabalho a abordagem utilizada delega aos desenvolvedores de linguagens de programação OO, neste caso a linguagem Java (SUN MICROSYSTEMS, 2009), a codificação direta das classes e relacionamentos pelo padrão UML.

Conforme Cattell *et al* (2000), o modelo de objetos propõe pelo padrão ODMG que o estado de um objeto seja representado por seus atributos e relacionamentos. Este tipo de representação é muito importante, pois as consultas em BDOO são feitas de acordo com os estados fornecidos.

Na formalização do modelo de dados como também abordado no diagrama de classes UML, foi definido que uma classe também possui operações que devem ser representadas e, posteriormente, implementadas em métodos presentes no escopo desta classe (RUMBAUGH, JACOBSON & BOOCH, 2004). Como citado anteriormente, além das operações definidas explicitamente no diagrama, os métodos implícitos também devem ser codificados, com exceção das operações *sets* que não devem ser implementadas, pois não é usual que algum dado existente nas dimensões e nas classes de fatos seja deletado ou alterado. Na Fig. 7 abaixo a classe dimensional CDAgrupamento está definida em Java, conforme os mesmos

atributos definidos no diagrama de classes CSB e as operações implementadas pelos métodos *addFato* e *gets* destes atributos.

```
public class CDAgrupamento {

        private int codAgrupamento;
        private int codTipoAgrupamento;
        private String dscAgrupamento;
        private ArrayList listaFatos;

        public int getCodAgrupamento(){
                return codAgrupamento;
        }
        public int getCodTipoAgrupamento(){
                return codTipoAgrupamento;
        }
        public String getDscAgrupamento(){
                return dscAgrupamento;
        }
        public void addFato(CSCSB credito){
                this.listaFatos.add(credito);
        }
}
```

Figura 7 – Código parcial da classe de dimensão CDAgrupamento em Java.

Refletindo agora sobre alguns detalhes de implementação, toda classe deve possuir um método construtor para inicializar todos os seus atributos quando a mesma é instanciada. Este método construtor recebe os valores atômicos referentes a todos os seus atributos para as inicializações, exceto para a lista de fatos que deve ser apenas criada no construtor, pois terá elementos adicionados nela somente quando ocorrer um fato do qual a dimensão participe.

Conforme Bezerra (2003), para que o conceito estrutural de objeto seja implementado, os atributos das classes devem possuir visibilidade privada e os métodos, inclusive os utilizados para acessarem esses atributos, devem possuir visibilidade pública, modeladas no Java pelas palavras reservadas *private* e *public*.

Observa-se no código Java desenvolvido que os identificadores e descritores não estão assim explícitos, sendo que estes primeiros nem sequer encontram-se implementados. Segundo Cattell *et al* (2000), os OIDs são transparentes para os usuários e os desenvolvedores, ou seja, o OID deve ser representado nos diagramas, mas não precisa ser codificado explicitamente: o próprio sistema de gerenciamento de objetos do ambiente OO utilizado para persistir os objetos gera, automaticamente, um OID associado ao objeto no momento em que este é instanciado.

Já nas classes de fatos os atributos que correspondem às métricas foram implementados como variáveis estáticas, usando-se a palavra reservada *static*. Toda vez que um objeto de fato for criado, indicando que um novo tipo de adesão ou cancelamento foi realizado, a métrica é incrementada automaticamente no método construtor, conforme o tipo de serviço. Como são atributos estáticos, os valores destas métricas são visíveis por todos os objetos instanciados destas classes (SUN MICROSYSTEM, 2009). Este tipo de implementação facilita as consultas OLAP posteriores que não sobrecarregariam o banco com cálculos de campos derivados para obter as sumarizações, pois basta consultar o campo pelo método *get* correspondente para obter esse somatório armazenado no BDOO.

4.5 Implementação e Execução da Consultas

Depois da codificação das classes sugere-se realizar a carga do novo banco de dados OO criado ou parte dele com os dados existentes no ambiente operacional. É neste momento que os recursos do BDOO escolhido são plenamente utilizados, pois a definição das classes pode ser feita por qualquer linguagem de programação em ambientes de desenvolvimento diversos, inclusive por scripts para modelar as classes, mas é no processo de carga e realização de consultas que o BDOO tem seu desempenho e funcionalidades avaliadas.

De acordo com Paterson (2006), o bd4o trabalha com três formatos de consultas: o QBE, as *Native Queries* e o SODA. O QBE (*Query-By-Example*) é o mecanismo mais básico oferecido pelo db4o e fornece à consulta um novo objeto com o mesmo estado

do objeto desejado, ou seja, um objeto externo é criado como exemplo para a consulta retornar o objeto armazenado que possui os mesmos valores de alguns ou todos os atributos fornecidos.

As *Native Queries* são muito utilizadas para consultas mais complexas e são escritas com as expressões da linguagem de programação OO utilizada, ultimamente C# ou Java. Neste tipo de formato, informa-se ao banco um trecho de código da linguagem utilizada com a lógica requerida pela consulta através de um objeto *Query*, próprio de uma biblioteca db4o.

Já o formato SODA (*Simple Object Data Access*) é uma API que oferece diversos tipos de objetos e métodos para realizar as consultas sob uma perspectiva mais OO. Dessa forma, a partir de instancias de objetos próprios suportados pelo bd4o, métodos SODA são combinados para realizar consultas que podem retornar uma lista de objetos, o *ObjectSet* (PATERSON, 2006). Sobre esta lista aplicam-se polimorfismo, encapsulamento e diversos mecanismos da OO para processar as informações obtidas. Para realizar as cargas e consultas das classes criadas nesta trabalho, foi utilizado o formato SODA para manipular os dados.

O novo BDOO implementado a partir do diagrama CSB é um objeto do tipo *ObjectContainer* que gera um arquivo de extensão YAP através do método *openFile* do bd4o. Para inserir os dados no banco, basta invocar o método *store* deste objeto, passando como parâmetro os objetos a serem armazenados e, depois de utilizado, o método *close* é chamado para encerrar a sessão com o *ObjectContainer*.

O código implementado para os testes de carga e consulta possui os dois métodos *processoStagingProd*, que insere os dados nas dimensões; e *processoDMVendasAdesao*, que representa os dados do modelo dimensional ADESAO_CSB. Conforme o site da Sun (SUN MICROSYSTEMS, 2009), para aplicações relacionais, estas cargas são facilmente implementadas via JDBC, conjunto de métodos muito utilizados comercialmente oferecidos pela tecnologia Java para acessar bancos de dados transacionais e mapear seus dados em objetos para, depois, gravá-los no *ObjectContainer* criado do bd4o.

Os objetos referentes às dimensões são buscados do banco, alterados e novamente gravados. Como já discutido, é papel do BDOO utilizado associar um novo OID para cada novo objeto armazenado, mas o db4o gerencia esse tipo de situação automaticamente, verificando se o objeto a ser armazenado pelo método *store* é resultante de algum processamento sobre algum objeto consultado do mesmo tipo. Caso seja, ele somente atualiza o objeto consultado, senão um novo objeto é armazenado no *ObjectSet* instanciado (PATERSON, 2006).

Por fim, para ilustrar a codificação do teste, realizou-se uma busca por todas as ocorrências de objetos do tipo CSAdesaoCSB armazenados no BDOO, apresentando como resultado de execução os atributos descritores das dimensões participantes e as métricas da própria classe de fatos.

Incluindo a realização desta busca como todo o processamento anterior discutido, o db4o demonstrou bom desempenho de execução, semelhante aos custos de implementação e testes das aplicações OO pelo fato de basear suas consultas no próprio framework oferecido pela linguagem OO hospedeira. Conforme as informações oficiais do db4o (DB4OBJECTS, 2008), o uso deste banco de objetos permite facilmente o armazenamento de estruturas altamente complexas de dados, atingindo ótimo desempenho de performance em relação aos bancos de dados objeto-relacionais existentes no mercado: o db4o pode ser executado até 44 vezes mais rápido que o MySQL com Hibernate, pois economiza cerca de 90% do custo de desenvolvimento da camada de persistência e agiliza a comercialização em um período de tempo até 10% menor em relação a algumas tecnologias existentes.

5 CONCLUSÃO

A utilização das técnicas formuladas pela metodologia proposta por Borba em 2006 proporcionou bons resultados que refletiram alto desempenho e baixo custo de desenvolvimento frente às tecnologias utilizadas atualmente para bancos de dados, incluindo maior uso da abstração junto ao cliente e melhoria da semântica de negócio promovida pela união dos ideais da UML e dos conceitos de DW.

Existem hoje no mercado diversas soluções que convergem seus esforços sobre a tecnologia objeto-relacional a qual não suporta integralmente os conceitos da OO. Dependendo do negócio modelado, estes conceitos devem ser aplicados para apresentar aos envolvidos uma estrutura de armazenamento e processamento mais próxima da semântica desejada. Mesmo que seja comum, em um primeiro momento, que a tecnologia objeto-relacional seja utilizada para carregar os dados do ambiente operacional, geralmente relacional, as próximas manipulações dos dados no DW seriam somente sobre o BDOO modelado. Quando utilizado um modelo OO visando a sua implementação em ambientes de bancos de dados, os conceitos oferecidos pelo paradigma OO são preservados e aplicados de forma direta entre o banco e as aplicações, incluindo as cargas de dados ocorridas dentro do DW e as consultas OLAP realizadas pelos usuários nos sistemas gerenciais.

Sob uma visão analítica, é perceptível que a metodologia discutida abrange todas as etapas usuais para um processo de desenvolvimento de DW descritas por Kimball, mas seu principal destaque é a fundamentação no paradigma OO. Como a programação OO está cada vez mais presente no desenvolvimento de softwares, alcançando até as linguagens do ambiente web, e como nenhuma organização sobrevive sem a implantação e gerência de um banco de dados, é notável a compatibilidade da abordagem OO para a modelagem de bancos de objetos.

Estas mesmas organizações, com o rápido crescimento do seu volume de dados, vêm implantando soluções de DW em busca de diferencial competitivo. Dessa forma, pela dinâmica do mercado, é interessante que as empresas possuam uma metodologia bem definida e ágil para a implantação de soluções inteligentes no gerenciamento de dados do ambiente de DW. A metodologia discutida neste trabalho, bem como as adaptações realizadas, projeta um único processo de definição e implementação, utilizando uma única abordagem a baixos custos de desenvolvimento. A modelagem para BDOO apresenta-se como uma opção viável para a modelagem dimensional, permitindo o tratamento de objetos complexos, o aumento da reusabilidade de código e desempenho compatível com os modelos relacionais.

AGRADECIMENTOS

Agradeço a todos que me incentivaram nesta caminhada rumo ao conhecimento. Agradeço à minha mãe, alma inspiradora, e a toda minha família; aos meus amigos Adriano, Ana Paula, Breno e Francielly, que me acompanham em amizade sólida; à minha namorada Luanda, quem sempre me ajudou e me confortou; e ao meu orientador pelos conselhos. Muito obrigado a todos.

REFERÊNCIAS

BARBIERI, Carlos. “BI - Business Intelligence – Modelagem & Tecnologia”. Rio de Janeiro (RJ): Axcel Books, 2006. 424 p.

BEZERRA, Eduardo. “Princípios de Análise e Projeto de Sistemas com UML”. Rio de Janeiro (RJ): Elsevier: Campus, 2007. 2ª ed. 369 p.

BORBA, Sueli de Fátima Poppi. “Metodologia para implantação de modelos dimensionais em banco de dados orientado a objetos”. Florianópolis (SC), 2006. 228p. Doutorado em Engenharia de Produção - Universidade Federal de Santa Catarina - UFSC. Disponível em: http://www.tede.ufsc.br/teses/PEPS5029.pdf - Acesso em janeiro de 2009.

BORBA, Sueli de Fátima Poppi; MORALES, Aran Bey Tcholakian. “Aplicação de Banco de Dados Orientado a Objetos na Modelagem Multidimensional”. Florianópolis (SC), 2006. XXI Simpósio Brasileiro de Banco de Dados - SBBD. Disponível em: http://www.lbd.dcc.ufmg.br:8080/colecoes/sbbd/2006/010.pdf - Acesso em janeiro de 2009.

BOSCARIOLI, Clodis; BEZERRA, Anderson; BENEDICTO, Marcos de; DELMIRO, Gilliard. “Uma reflexão sobre Banco de Dados Orientados a Objetos”. Ponta Grossa (PR), 2006. IV Congresso de Tecnologias para Gestão de Dados e Metadados do Cone Sul - CONGED. Disponível em: http://conged.deinfo.uepg.br/artigo4.pdf - Acesso em janeiro de 2009.

CATTELL, Rick. G. G.; BARRY, Douglas K.; BERLER, Mark; EASTMAN, Jeff; JORDAN, David; RUSSELL, Craig; SCHADOW, Olaf; STANIENDA, Torsten; VELEZ, Fernando. “The Object Data Standard: ODMG 3.0”. San Francisco (USA), 2000. Grupo de Gestão de Objetos - Objects Management Group - OMG. Disponível em: http://www.omg.org/docs/omg/04-07-02.pdf - Acesso em janeiro de 2009.

CORE SYNESIS. Disponível em: http://www.coresynesis.com.br - Acesso em dezembro/2008.

DB4OBJECTS. “db4o - Banco de objetos de código aberto”. Disponível em: https://www.db4o.com/portugues/db4o%20Product%20Information%20V5.0(Portuguese).pdf - Acesso em dezembro de 2008.

ELMASRI, Ramez; NAVATHE, Sham. “Sistemas de banco de dados: fundamentos e aplicações”. Rio de Janeiro (RJ): LTC, 2006. 3ª ed. 837 p.

INMON, William H. “Building the Data Warehouse”. Indianapolis (USA): Wiley Computer Publishing, 2005. 4ª ed. 543 p.

KIMBALL, Ralph. “Data Warehouse Toolkit”. Tradução de Mônica Rosemberg. São Paulo (SP): Makron Books, 1998. 388 p.

KIMBALL, Ralph; ROSS, Margy. “The Data Warehouse Toolkit”. Indianapolis (USA): Wiley Computer Publishing, 2002. 2ª ed.421 p.

KIMBALL, Ralph; ROSS, Margy; REEVES, Laura; THORNTHWAITE, Warren. “Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing and Deploying Data Warehouses”. New York (USA): John Wiley & Sons, 1998. 2ª ed. 405 p.

KLEIN, Lawrence Zordam; CAMPOS, Maria Luiza Machado; TANAKA, Astério Kiyoshi. “A Tecnologia Objeto-Relacional em Ambientes de Data Warehouse: Uso de Séries Temporais como Tipo de Dado Não Convencional”. Florianópolis (SC), 1999. XIV Simpósio Brasileiro de Banco de Dados - SBBD. Disponível em: http://www.inf.ufsc.br/sbbd99/anais/SBBD-Completo/30.PDF - Acesso em janeiro de 2009.

MACHADO, Felipe Nery Rodrigues; ABREU, Maurício Pereira de. “Projeto de Banco de Dados - Uma visão prática”. São Paulo (SP): Érica, 2004. 11ª ed. 299 p.

PATERSON, Jim. EDILCH, Stefan. HÖRNING, Henrik. HÖRNING, Reidar - “The Definitive Guide to bd4o” - Nova Iorque (EUA): Apress, 2006. 484 p.

ROSEMBERG, Dave. “INDRA: Sistema de Missão Crítica para controle de trens de alta velocidade”. Tradução de Cássio R. Eskelsen. Disponível em: https://www.db4o.com/portugues/db4o%20Success%20Story%20-%20INDRA%20Sistemas(Portuguese).pdf- Acesso em dezembro de 2008.

RUMBAUGH, James; JACOBSON, Ivar; BOOCH, Grady. “The Unified Modeling Language Reference Manual”. Boston (EUA): Addison-Wesley, 2004. 2ª ed. 721 p.

SUN MICROSYSTEMS. Disponível em: http://www.sun.com/ - Acesso em janeiro/2009.

VIDOTTI, Julio Cesar. “Projeto de um Data Warehouse: Análise de Custo/Benefício”. Cuiabá (MT), 2001. 36 p. Universidade Federal do Mato Grosso - UFMT. Disponível em: http://www.ufmt.br/cacomp/Downloads/monografias/ProjetoDataWareHouse.pdf - Acesso em janeiro de 2009.