RESIGeT On-Line Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica Uni-FACEF CENTRO UNIVERSITÁRIO DE FRANCA
05 Nr. 01 - 2015

BANCO DE DADOS OBJETO-RELACIONAL PARA APLICAÇÕES WEB

Eduardo Galvani Massino[1]
sargento.elfico@gmail.com
Carlos Eduardo de França Roland[2]
roland@facef.br

Resumo: o objetivo deste artigo é discutir a possibilidade de utilização de um Sistema Gerenciador de Bancos de Dados Objeto-Relacional para armazenar e consultar dados complexos do catálogo de produtos de um sistema de *e-Commerce*, no caso o Prestashop, que foi originalmente desenvolvido no modelo Relacional. Para tanto, o procedimento metodológico empregado foi isolar o escopo do projeto que trata do cadastro das características dos Produtos, refazendo a modelagem sob o modelo Objeto-Relacional, para comparar o tempo de execução das consultas SQL nos dois modelos, verificando a performance de cada implementação. Os testes de consultas feitas no protótipo com modelo Relacional, com o código SQL mais complexo envolvendo mais de 9 tabelas, duraram mais de 1 segundo cada, chegando ao máximo de 3,5 segundos. Já no modelo Objeto-Relacional, a escala de tempo permaneceu em milissegundos, chegando no máximo a 13,5 milissegundos para a mesma consulta, o que corresponde a 0,013 segundos. Pode-se, a partir dos resultados dos testes deste trabalho, considerar que o modelo Objeto-Relacional, principalmente em um contexto específico, apresenta vantagens em relação ao Relacional.

Palavras-chave: modelo Objeto-Relacional. BDOR. SGBD. e-*Commerce*. SQL.

Introdução

Atualmente, a maioria dos sistemas de *e-Commerce* utilizam Sistemas Gerenciadores de Banco de Dados (SGBD) relacionais, e em estruturas desses sistemas o nível de complexidade e relacionamento entre as entidades é muito alto. Tal complexidade resulta em uma alta carga de dados armazenados e na consequente lentidão na consulta e retorno dos dados filtrados.

A tecnologia de Banco de Dados Objeto-Relacional (BDOR) é pensada para permitir o armazenamento de dados complexos de forma mais simples. O objetivo deste trabalho é verificar o quanto se consegue melhorar a performance de consultas em sistemas web que processam atributos complexos de dados. Optou-se neste estudo, por verificar que resultados se obtém quando se tratar entidades Produtos como objetos em um gerenciador de bancos de dados que implemente esta tecnologia.

A metodologia adotada nesse trabalho foi o de isolar, em um sistema de *e-Commerce*, as tabelas e as relações da entidade Produto, especificadamente o atributo que armazena suas características, para transformar e, a partir de seu

[1] Discente do Curso de Pós Graduação em Gestão e Desenvolvimento de Software para Web do Uni-FACEF
[2] Docente do Curso de Pós Graduação em Gestão e Desenvolvimento de Software para Web do Uni-FACEF

modelo relacional, criar uma base de dados Objeto-Relacional com os mesmos dados, mas uma estrutura diferente de persistência. Dessa forma foi possível fazer uma comparação com a estrutura originalmente relacional, a fim de verificar as vantagens ou desvantagens em termos de desempenho quantitativo de tempo das consultas aos produtos.

1 Paradigmas de banco de dados

A utilização de banco de dados já é inerente ao cotidiano nas atividades da sociedade moderna. Sua utilização é essencial na coleta, alteração e consulta de informações pertinentes a todos os âmbitos de negócios, desde agricultura, indústria, comércio, serviços, etc.

Para que os dados possam ser mantidos são utilizados os SGBD nos quais os dados são armazenados de forma organizada, ficando disponíveis para serem consultados.

Há dois conceitos distintos envolvidos nesse processo: dados e informações. A informação é o uso do resultado do processamento de dados, como contagens, estatísticas, produtos, clientes, etc. Por exemplo: a partir de um conjunto de dados sobre compradores de um produto e suas respectivas idades, pode-se extrair a informação de que a maioria dos compradores possuem entre 18 e 25 anos.

Essa extração de informações é o maior objetivo dos bancos de dados, pois apenas armazená-los sem nenhum propósito definido seria apenas um desperdício de recursos computacionais, tempo e dinheiro.

1.1 Banco de Dados Relacionais no Contexto Web

Os SGBD mais utilizados comercialmente são os que utilizam o modelo relacional. Neste modelo o banco de dados é representado como uma coleção de relações.

> Uma relação é pensada como uma tabela de valores, cada linha na tabela representa uma coleção de valores de dados relacionados. [...] representa um fato que corresponde a uma entidade ou relacionamento do mundo real. O nome da tabela e os nomes das colunas são usados para ajudar na interpretação do significado dos valores em cada linha [...]. Todos os valores em uma coluna são do mesmo tipo de dado (ELMASRI e NAVATHE, 2005, p. 90).

Na prática, por exemplo em um sistema web de *e-Commerce*, toda a lógica da aplicação é definida e a partir dela é gerada uma estrutura de tabelas relacionadas representando as entidades e relações de operações de comércio. As entidades são representadas pelos clientes e os produtos que estão à venda, e nesse caso, a compra, que basicamente é o registro dos produtos comprados por determinado cliente, é uma relação entre as entidades clientes e produtos, com dados adicionais como: hora da compra, forma de pagamento, custos de entrega, etc.

1.2 Uso Prático do Modelo Relacional em *e-Commerce*

Em um sistema complexo, que define rigorosamente todos os detalhes inerentes a uma operação comercial, a quantidade de entidades e relações é muito

grande, o que implica, de uma forma simplificada, na existência de muitas tabelas, cada tabela com muitas colunas, e obviamente muitas linhas nessas tabelas, de acordo com a popularidade do *e-Commerce* em questão.

Em um sistema web a velocidade da aplicação no tempo de resposta da execução das operações dos clientes é ponto crucial para o sucesso de sua utilização. Os clientes querem que a experiência de compra seja rápida e eficiente e não esperam justificativas para uma possível lentidão por motivos diversos como: muitos clientes fazendo compras ao mesmo tempo; muitos produtos para serem buscados por certos critérios numa pesquisa por palavras-chave; dentre outras.

Portanto, a consulta aos dados deve ser planejada e feita de uma forma que seja ao mesmo tempo simples e eficiente (que traga todas as informações úteis possíveis para a escolha do cliente). Essa consulta, ou *query* no jargão da programação, é feita, no caso dos modelos relacionais, tradicionalmente por meio de código escrito em Structured Query Language (SQL), e que de acordo com a linguagem ou *framework* utilizados no sistema pode ser vista diretamente no código, ou por meio de bibliotecas de mapeamento que encapsulam o código SQL em comandos próprios da linguagem de programação utilizada.

Nesses dois casos, a resposta do código SQL quando utilizado, sendo aparente ou não e de acordo com seu nível de complexidade, será mais ou menos rápida, sendo esse o tempo que é levado em conta quando um sistema é classificado como lento ou rápido, chamado de tempo de resposta do servidor. Por isso, códigos SQL eficientes são o primeiro passo para se ter um sistema web eficiente.

Essa eficiência da consulta tem a ver com a carga de dados, ou seja, com a quantidade de linhas de registro que essa consulta terá que tratar e devolver ao sistema. O que quer dizer que, um código que é eficiente para uma pequena quantidade de dados, pode não ser para uma grande quantidade, e então deverá ser revisado e até mesmo refeito.

1.3 Banco de dados orientado a objetos - BDOO

De acordo com Corbellini, Oliveira e Scherer (2012, p. 1), os primeiros BDOO foram criados numa tentativa de atender diretamente do SGBD o tratamento a estruturas de dados mais complexas e a uma melhor integração ao crescente número de linguagens de programação orientadas a objetos.

> Uma característica-chave dos bancos de dados orientados a objetos é o poder dado ao projetista para especificar tanto a estrutura de objetos complexos quanto as operações que podem ser aplicadas a esses objetos (ELMASRI e NAVATHE, 2005, p. 459).

Enquanto os bancos relacionais utilizam-se do conceito de tabelas, colunas e linhas, os BDOO utilizam os conceitos já conhecidos da programação orientada a objetos: classes, objetos, herança, polimorfismo e encapsulamento.

Apesar de já existir diversas ferramentas disponíveis no mercado, como o ObjectDB, Jasmine, GemStone/S, entre outras, ainda há uma certa hesitação na cultura empresarial. Segundo Boscarioli, Bezerra, Benedicto e Delmiro (2006, p. 10), “devido ao tempo necessário e custos envolvidos, muitos hesitam na migração para um banco de dados orientado a objetos”, contribuindo para a permanência no modelo relacional, que além disso, já possui na bagagem anos de desenvolvimento, investimentos e aperfeiçoamentos.

1.4 Uma nova abordagem: Banco de Dados Objeto-Relacional (BDOR)

Numa tentativa de unir as vantagens de ambas as tecnologias anteriores, os BDOR modelam objetos armazenados em tabelas, ou seja, utilizam as tabelas do modelo relacional, mas nelas são armazenados objetos, com seus atributos e comportamentos, unindo assim os paradigmas.

> Utiliza os conceitos de supertabelas, supertipos, herança, reutilização de código, encapsulamento, controle de identidade de objetos (OID), referência a objetos, consultas avançadas e alta proteção dos dados (VASCONCELOS, 2013).

Ou seja, os dados são armazenados em relações, mas pode-se armazenar dados complexos abstraindo seu comportamento da mesma forma como é feito na orientação a objetos.

Na prática, cria-se um tipo X (que pode ser visto como objeto nesse contexto) que tem n atributos (ou colunas); e depois ao se criar uma tabela A, uma das suas colunas será do tipo X, e nessa coluna, ao invés de se armazenar um tipo comum de dados (*int*, *varchar*, *decimal*, etc.) se armazena o tipo X que por sua vez poderá ter várias colunas de tipos comuns ou mesmo outros tipos. Esse quadro geral adquire assim uma aparência de objetos e suas características, ou ao menos, de dados complexos, o que já é um avanço em relação ao tradicional modelo relacional.

Como exemplo de aplicações que se utilizam dessa tecnologia destacam-se: softwares de armazenamento de imagens obtidas por satélite, projetos de arquitetura, mapas geoespaciais e de relevo, dentre outros.

1.5 PostgreSQL

O PostgreSQL é um SGBD no modelo Objeto-Relacional, que está sob a licença *PostgreSQL License*, similar ao BSD e MIT, que permite o livre uso, modificação e distribuição. Ele tem vários recursos, com destaque ao suporte a operações assíncronas, busca de textos avançadas, dentre outros. Mas o mais interessante é seu compromisso em manter o padrão ANSI-SQL em sua linguagem de consulta.

Nele as tabelas, relacionamentos, *triggers*, e demais entidades de banco são considerados objetos. Mas como já dito no item 1.4, não segue à risca o conceito de objeto de uma linguagem de programação orientada a objetos. Apesar de se poder usar recursos como herança e polimorfismo, seu foco é na manipulação de dados complexos como na criação de tipos que são usados nas tabelas.

Portanto não há a notação comum de objetos, como a seguir:

SELECT NOW().to_char('HH12:MI:SS')

É mantida a notação comum de funções:

SELECT to_char(NOW(), 'HH12:MI:SS')

Porém essa função é polimórfica, ou seja, aceita outros parâmetros de acordo com sua definição, podendo ser usada de outras formas. Conclui-se que o modelo possui recursos de objetos, ao invés de usar o conceito de orientação a objetos propriamente dito.

Para exemplificar o conceito de tipos, considere o seguinte cenário, em linguagem SQL:

```
CREATE TYPE categoria AS (
        nome        text,
        código      integer,
        imposto     numeric
);
CREATE TABLE produto (
        nome        text,
        preco_base  numeric,
        categoria   categoria
);
```

Uma inserção na tabela Produto, com o tipo complexo Categoria, podendo ser feito de uma das duas formas a seguir:

INSERT INTO produto VALUES ('Lorem', 1.99, ROW('Categoria 1', 1, 0.35));

ou

INSERT INTO produto VALUES ('Lorem', 1.99, ROW('Categoria 1', 1, 0.35)::categoria);

A única diferença é a conversão realizada (::categoria) para o caso de existir na mesma estrutura outros tipos que tivessem colunas com as mesmas características que as do tipo Categoria.

2 Plataforma de Comércio Eletrônico - Prestashop

O Prestashop é uma plataforma *open source* de comércio eletrônico. Pode ser obtido gratuitamente pelo seu endereço oficial http://www.prestashop.com/ e usado diretamente por comerciantes ou adaptado por uma equipe/empresa de desenvolvimento, com o objetivo, na maioria das vezes, de uma personalização da interface do *website* (*layout*).

É um projeto de grande escala tendo sua primeira versão lançada em maio de 2007. Seu escopo de projeto é bem amplo de forma a possibilitar a compra e venda dos mais variados tipos de produtos, cada um com suas particularidades como por exemplo: produtos eletrônicos, roupas, calçados, jogos e até mesmo conteúdo digital sob a forma de download, atendendo assim ao maior número possível de áreas comerciais.

Dessa forma o modelo relacional e por consequência a modelagem de classes do projeto foram produzidos de forma a permitir o cadastro e gerenciamento de todos esses diferentes tipos de produto, ou seja, a modelagem não é específica a qualquer área do comércio, mas genérica.

3 Análise de Desempenho – Modelo Relacional x Objeto-Relacional

Para essa análise foi usado um serviço de hospedagem que disponibiliza em ambiente *Linux*, o servidor *Apache*, e com ele a linguagem *PHP*, e os bancos de dados *MySQL* e *PostgreSQL*.

Foi instalado nesse ambiente a plataforma *Prestashop* em sua versão 1.5.3.1, que tem como requisitos os recursos mencionados, exceto o banco *PostgreSQL* que não é usado pela plataforma e não possui compatibilidade oficial (a compatibilidade e uso do *Prestashop* é apenas com o *MySQL*).

Para efeito da análise foram cadastrados três produtos do tipo Aliança, cuja particularidade é a opção do cliente ter que escolher sempre dois tamanhos diferentes de aro (diâmetro correspondente ao de seus dedos), já que o objetivo principal é a compra para dois parceiros, seja casamento, noivado, etc.

Para tanto, primeiramente foram cadastradas essas opções no cadastro de características (atributos e valores): Aro 1 e Aro 2, e em cada um as diferentes opções disponíveis para venda que estarão disponíveis para todos os produtos.

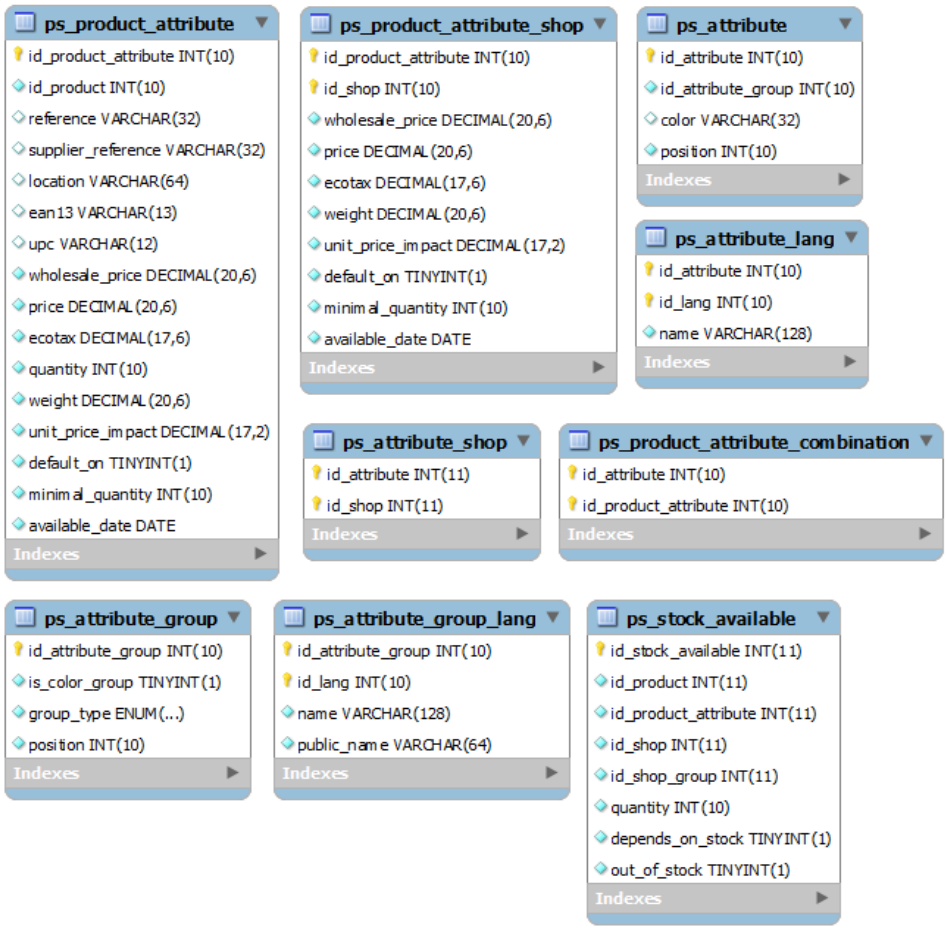
Ao todo foram cadastrados trinta valores em cada característica, ou seja, 30 diferentes opções para cada Aro. Em seguida foram cadastradas as três alianças que, para este teste, possuem a seguinte configuração:

- A primeira possui 30 opções para cada Aro, a diversidade máxima nesse caso, o que gera ao todo 900 combinações;
- A segunda possui 15 opções para cada Aro, metade das opções disponíveis, gerando 225 combinações.
- A terceira possui uma opção para cada Aro, apenas uma combinação.

O número de combinações diz respeito ao número de registros na tabela responsável no *Prestashop* em salvar essas opções dos produtos. Na realidade, por se tratar do modelo relacional, e como já dito, genérico a diferentes tipos de produtos, trata-se de um conjunto de nove tabelas, cuja modelagem é mostrada na figura 1, em formato de Diagrama Entidade-Relacionamento (DER).

Tal estrutura foi adaptada para o modelo Objeto-Relacional que é específica aos produtos do tipo Aliança que serão usados para a comparação do desempenho da consulta aos dados armazenados nos dois distintos modelos.

Figura 1 – MODELO RELACIONAL DAS CARACTERÍSTICAS



Fonte: AUTOR

O novo modelo foi implementado no banco de dados PostgreSQL, e apresenta como entidade de características do Produto apenas uma tabela, gerada a partir do comando SQL a seguir:

```
CREATE TABLE ps_product_alianca (
        id_atributo serial NOT NULL,
        id_product integer NOT NULL,
        tipo_produto tipo_produto NOT NULL,
        aro_1 aro[],
        aro_2 aro[],
        group_type group_type NOT NULL,
        price money,
        CONSTRAINT ps_product_alianca_pkey PRIMARY KEY (id_atributo)
)
WITH(OIDS=FALSE);
```

Além dos atributos mais pertinentes ao produto, que já existiam no modelo relacional, como *id_product* (responsável pelo relacionamento com a tabela de produto), *group_type* e *price*, foram acrescentados outros três.

O primeiro é o *tipo_produto*, que define se o tipo do produto em questão será uma Aliança (os dois aros serão usados) ou Anel (apenas o primeiro aro será usado) e outros de acordo com a necessidade, que é do tipo **tipo_produto**,

um tipo ENUM (um tipo que é definido com valores constantes), criado pelo código SQL a seguir:

```
CREATE TYPE tipo_produto AS ENUM (
        'aliança',
        'anel',
");
```

O segundo e o terceiro tipos criados são *aro_1* e *aro_2*, que serão usados para armazenar os aros e suas respectivas quantidades em estoque das alianças, sob o tipo aro[], onde os colchetes indicam que os campos armazenarão *arrays* do tipo de dados aro constituído dos atributos: aro e estoque, que foi criado a partir do comando SQL a seguir:

```
CREATE TYPE aro AS (
        aro integer,
        estoque integer);
```

Por último, foram inseridos os registros que se igualam à lógica relacional das três alianças já definidas, em que uma possui uma combinação de 30 x 30 aros, a outra de 15 x 15 aros, e a última 1 x 1 aro. Por exemplo, o registro de 15x15 é mostrada a seguir.

```
INSERT INTO "public"."ps_product_alianca" ("id_atributo","id_product","tipo_produto","aro_1","aro_2","group_type","price") VALUES (nextval('ps_product_alianca_id_atributo_seq'::regclass),'9','aliança', '{"(11,100)","(12,100)","(13,100)","(14,100)","(15,100)","(16,100)","(17,100)","(18,100)","(19,100)","(20,100)","(21,100)","(22,100)","(23,100)","(24,100)","(25,100)"}','{"(11,100)","(12,100)","(13,100)","(14,100)","(15,100)","(16,100)","(17,100)","(18,100)","(19,100)","(20,100)","(21,100)","(22,100)","(23,100)","(24,100)","(25,100)"}', 'select','1500');
```

3.1 Análise de desempenho das consultas SQL

Com a estrutura e dados pronta, basta executar as consultas e registrar a velocidade de cada uma. Foram feitas três consultas consecutivas de cada um dos 3 produtos, na estrutura relacional e Objeto-Relacional.

A consulta relacional, é a realizada normalmente pelo Prestashop, quando um usuário do *website* navega na página do produto. É chamada pela função *getAttributesGroups*() da classe *Product*, que por sua vez é chamada pela função *assignAttributesGroups*() da classe *ProductController*, o comando é integralmente mostrado na figura 2.

Figura 2 - CONSULTA RELACIONAL PRESTASHOP

```
SELECT ag.`id_attribute_group`, ag.`is_color_group`, agl.`name` AS group_name, agl.`public_name` AS public_group_name,
a.`id_attribute`, al.`name` AS attribute_name, a.`color` AS attribute_color, pa.`id_product_attribute`,
IFNULL(stock.quantity, 0) as quantity, product_attribute_shop.`price`, product_attribute_shop.`ecotax`,
pa.`weight`, product_attribute_shop.`default_on`, pa.`reference`, product_attribute_shop.`unit_price_impact`,
pa.`minimal_quantity`, pa.`available_date`, ag.`group_type`
FROM `ps_product_attribute` pa
INNER JOIN ps_product_attribute_shop product_attribute_shop
  ON (product_attribute_shop.id_product_attribute = pa.id_product_attribute AND product_attribute_shop.id_shop = 1)
LEFT JOIN ps_stock_available stock ON (stock.id_product = pa.id_product
  AND stock.id_product_attribute = IFNULL(`pa`.id_product_attribute, 0) AND stock.id_shop = 1 )
LEFT JOIN `ps_product_attribute_combination` pac ON pac.`id_product_attribute` = pa.`id_product_attribute`
LEFT JOIN `ps_attribute` a ON a.`id_attribute` = pac.`id_attribute`
LEFT JOIN `ps_attribute_group` ag ON ag.`id_attribute_group` = a.`id_attribute_group`
LEFT JOIN `ps_attribute_lang` al ON a.`id_attribute` = al.`id_attribute`
LEFT JOIN `ps_attribute_group_lang` agl ON ag.`id_attribute_group` = agl.`id_attribute_group`
INNER JOIN ps_attribute_shop attribute_shop
  ON (attribute_shop.id_attribute = a.id_attribute AND attribute_shop.id_shop = 1)
WHERE pa.`id_product` = 8 AND al.`id_lang` = 2 AND agl.`id_lang` = 2
GROUP BY id_attribute_group, id_product_attribute
ORDER BY ag.`position` ASC, a.`position` ASC
```

Fonte: AUTOR

Por sua vez, a consulta elaborada para a estrutura Objeto-Relacional, é mostrada na figura 3.

Figura 3 - CONSULTA OBJETO-RELACIONAL

```
SELECT id_atributo, id_product, tipo_produto, aro_1, aro_2, group_type, price
FROM ps_product_alianca
WHERE id_product = 9
```

Fonte: AUTOR

Os tempos de cada consulta foram registrados utilizando-se a ferramenta *SQLYog* para o banco MySQL e o *PhpPgAdmin* para o PostgreSQL. A escala usada é em milissegundos e os tempos estão relacionados na tabela 1.

Tabela 1 – Tempos registrados nas consultas testadas

	Combinações	*1ª. Consulta (ms)*	*2ª. Consulta (ms)*	*3ª. Consulta (ms)*
Aliança 1 *Relacional*	30 x 30	3.517,0	2.953,0	2.785,0
Objeto-relacional		13,6	2,7	2,5
Aliança 2 *Relacional*	15 x 15	3.065,0	2.809,0	2.773,0
Objeto-relacional		3,6	2,7	2,7
Aliança 3 *Relacional*	1 x 1	1.657,0	1.304,0	1.268,0

Objeto-relacional		3,1	3,0	2,3

Fonte: AUTOR

Considerações finais

Como pode ser observado, a diferença dos tempos é significativa. Todas as consultas feitas no modelo Relacional, com o código SQL bem mais complexo, envolvendo mais de 9 tabelas, duraram mais de 1 segundo cada, chegando ao máximo de 3,5 segundos na primeira consulta do produto com mais combinações. Já no modelo Objeto-Relacional, a escala de tempo permaneceu em milissegundos, chegando no máximo a 13,5 milissegundos, o que corresponde a 0,013 segundos.

Um ponto a ressaltar é que a cada execução consecutiva o servidor conseguiu reduzir o tempo das consultas nos dois modelos. Tal funcionalidade que normalmente é implementada nos SGBD e que é chamada de *cache de querys,* mantém os dados armazenados em um espaço de memória especial configurado na instalação do sistema, e que leva em conta a igualdade da consulta em relação às anteriores mantidas em *cache,* em vez de analisar e executar a sentença SQL novamente. O *cache de query* é compartilhado entre sessões, de tal forma que um resultado gerado por um cliente pode ser enviado como resposta a uma consulta idêntica gerada por outro cliente (MySQL, 2014).

Conclui-se que o modelo Objeto-Relacional, principalmente em um contexto específico, é vantajoso em relação ao Relacional.

Deve-se, contudo, ressaltar que a modelagem relacional original do Prestashop é genérica, e portanto, se fosse específica a um tipo apenas de produto poderia ter um melhor desempenho, o que não muda o fato de a tecnologia de modelagem Objeto-Relacional ser uma ótima alternativa ao padrão estabelecido de mercado, possuindo recursos interessantes relacionados ao paradigma de orientação a objetos, e sobretudo um ótimo desempenho quando tratando de estruturas de dados complexas.

Referências

BOSCARIOLI, Clodis; BEZERRA, Anderson; BENEDICTO, Marcos de; DELMIRO, Gilliard. *Uma reflexão sobre Banco de Dados Orientados a Objetos*. 2006. 12 f. Universidade Estadual do Oeste do Paraná, Toledo, 2006.

CORBELLINI, Anderson; OLIVEIRA, William Hart; SCHERER, Adriana Paula Zamin. *Banco de Dados Orientados a Objetos*. 2012. 12 f. Faculdade Dom Bosco, Porto Alegre, 2012.

ELMASRI; NAVATHE. *Sistemas de Banco de Dados*. 4. ed. São Paulo: Pearson Education, 2005. 730p.

MySQL. *The MySQL Query Cache.* Disponível em: http://dev.mysql.com/doc/refman/5.1/en/query-cache.html acesso em: 28 out. 2014.

VASCONCELOS, Rafael Oliveira*. Comparativo entre Banco de Dados Orientado a Objetos (BDOO) e Bancos de Dados Objeto Relacional (BDOR).* 2009. Disponível em: http://rafaeloliveirav.wordpress.com/2009/06/19/artigo-comparativo-entre-banco-de-dados-orientado-a-objetos-bdoo-e-bancos-de-dados-objeto-relacional-bdor-parte-2/ acesso em 19 out. 2013.