

# Um Banco de Dados Orientado a Objeto para Java com Suporte a Transações e Controle de Concorrência

Epifânio  
Giovani Facchini  
João Marcelo  
Juliano Valentini  
Renato Costa

Outubro de 2004

## Resumo

Assim como tem acontecido com as linguagens de programação, essencialmente o JAVA, o paradigma orientado a objetos tende a se difundir no mercado de banco de dados. Isso devido ao fato de que um modelo de banco de dados orientado a objetos possui maior capacidade de especificação e maior nível de abstração, o que reduz o *gap* semântico, em relação aos bancos de dados relacionais. Nessa perspectiva, esse trabalho propõe o desenvolvimento de um protótipo desse tipo de banco de dados implementado em JAVA, com suporte a transações, controle de concorrência e de código fonte aberto.

**Palavras-chave:** banco de dados, orientado a objeto

## 1 Motivação

Os bancos de dados relacionais que atualmente dominam o mercado possuem uma série de deficiências[1]. As estruturas de dados, hoje em dia, são mais complexas e com maior ênfase em inteligência de negócios, integração de dados, suporte a XML, entre outros. Uma das alternativas aos bancos de dados tradicionais são os bancos de dados orientados a objetos.

Com esse tipo banco, na mesma perspectiva do paradigma de programação orientado a objetos, torna-se possível desenvolver um modelo de dados de forma mais completa e muito mais simples. Dessa forma, um modelo dados OO fica mais próximo do mundo real, ou seja, diminui-se o *gap* semântico.

A *Sun Microsystems*[2] especifica a *JDO(Java Data Objects)*[3]. A JDO é uma API Java para abstração da persistência de dados. A maioria dos bancos de dados orientados a objetos que são implantados em Java baseiam-se nesse modelo. O presente trabalho será baseado num subconjunto da JDO que forma a viabilizar o término no tempo determinado.

## 2 Objetivos

O grupo propõe o desenvolvimento de um banco de dados orientado a objetos com suporte a transações e concorrência. Para garantir o término do trabalho em tempo hábil, será deixado para trabalhos futuros alguns recursos.

Primeiramente, o objetivo dessa implementação é validar o funcionamento básico desse tipo de banco de dados. Para tanto será garantido os seguintes recursos:

- Criação do arquivo de dados;
- Exclusão do arquivo de dados;
- Exclusão de objetos;
- Execução de métodos (aka UPDATE, por exemplo);
- Inserção de objetos;
- Seleção de objetos.

Após essa etapa, o foco do trabalho será voltado à implementação de controles que garantam as propriedades ACID de uma transação:

- *Atomicity* – garante que tudo (*commit*) ou nada (*rollback*) que estiver numa transação seja executado;
- *Consistency* – estabelece regras configuráveis para as informações e que garante que somente dados válidos serão aceitos;
- *Isolation* – previne que duas ou mais transações concorrentes façam alteração no mesmo dado;
- *Durability* – Uma vez feito *commit* na transação, as mudanças feitas jamais podem ser perdidas.

Os recursos supracitados serão disponibilizados através de pacotes. O programador deverá instanciar objetos e invocar métodos. A documentação da API dos pacotes será disponibilizada no formato *JAVADOC*[4].

### 3 Metodologia

Inicialmente, para a definição do trabalho, especificou-se uma pseudo-linguagem de acesso ao banco de dados similar ao *OQL(Object Query Language)*[5]. Segue alguns exemplos de comandos suportados pela linguagem:

```
CREATE <DatabaseFileName>
```

Para criar um novo arquivo de dados.

```
DELETE (<ClassName>) FROM (<DatabaseFileName>) [WHERE (...)];
```

Para excluir objetos do banco.

```
DROP (<DatabaseFileName>)
```

Para excluir um banco de dados.

```
EXECUTE (method, method) AT (<ClassName>) FROM (<DatabaseFileName>) [WHERE (...)];
```

Para executar métodos.

```
INSERT INTO (<DatabaseFileName>) NEW <ClassName> VALUES (param1, param2, param3)
```

Para inserir objetos.

```
SELECT (<ClassName>) FROM (<DatabaseFileName>) [WHERE (...)];
```

Para selecionar objetos.

A partir dessa pseudo-linguagem definiu-se as operações de baixo nível necessárias para dar suporte a execução desses comandos. As operações de baixo nível estão relacionadas basicamente com a leitura e escrita no disco, união e intersecção de objetos (dados) do banco e seleção de objetos.

O controle de transações e concorrência será implementado, na medida do possível, de acordo com as especificações da interface *Transaction* da JDO que é uma sub-interface da *PersistenceManager*.

## 4 Cronograma

Dentro do possível, as atividades serão realizadas da seguinte maneira:

Atividade	10/Set	24/Set	8/Out	22/Out	5/Nov	19/Nov	3/Dez
1	•						
2		•	•				
3			•	•			
4				•			
5					•	•	•
6							★

Descrição das atividades:

- 1 Estudo e pesquisa das atuais implementações de SGDBOs;
- 2 Definição das classes abstratas e implementação inicial da classe que vai manipular o arquivo do banco;
- 3 Implementação das interfaces;
- 4 Testes e ajustes finais para o funcionamento do banco
- 5 Implementação do suporte a concorrência e transações
- 6 Apresentação do trabalho

## Referências

- [1] **Addressing the Semantic Gap in Databases: Lazy Software and the Associative Model of Data**, Carl W. Olofson, IDC Bulletin 27774 - Aug 2002.
- [2] **Sun Microsystems**. <http://www.sun.com>.
- [3] **Java Data Objects 2.0 - JSR243**, Early Draft 1, June 10, 2004. Java Data Objects Expert Group, Sun.
- [4] **JAVADOC**. <http://java.sun.com/javadoc/> Acessado em outubro de 2004.
- [5] **Object Data Standard: ODMG 3.0**, R. G. G. Cattell, et al. (Morgan Kaufmann Publishers, 2000; ISBN: 1558606475).