Sistema Gerenciador de Banco de Dados Orientado a Objetos

Kamilla Taiwhscki Barros Silva[1], **Henrique Luís Santos Barata**[1], **Eurialdo Mendes Ferreira**[1]

[1]Faculdade de Engenharia de Computação – Universidade Federal do Pará (UFPA)
Rua Itaipú, 38 – Vila Permanente – Tucuruí – PA – Brazil

{kamillataiwhscki@gmail.com, henriquelbarat@gmail.com, eurialdogado@gmail.com

Resumo. Introdução: *Descreve detalhes sobre Sistemas Gerenciadores de Banco de Dados Orientado a Objetos e sua aplicação no desenvolvimento de banco de dados.* ***Referencial Teórico:*** *Nessa seção é detalhada a base teórica utilizada para as análises, fazendo uma breve síntese sobre os temas abordados.* ***Estudo de caso:*** *Foi utilizado uma aplicação desenvolvida pelos autores para fazer um estudo sobre o que são Sistemas Gerenciadores de Banco de Dados Orientados a Objetos e como eles fazem a comunicação com a base de dados e a aplicação citada. São explicitados os passos que são fundamentais para sintetizar o objetivo deste artigo.* ***Conclusões:*** *O estudo demonstrou como Sistemas Gerenciadores de Banco de Dados Orientado a Objetos (SGBDOOs) podem ser de grande utilidade para aplicações que utilizam Programação Orienta a Objetos e como isso pode facilitar o uso do usuário final assim como o trabalho dos desenvolvedores, já que a abstração e a menor complexidade dos problemas é algo que os desenvolvedores buscam cada vez mais, mostrando que SGDBOOs podem e devem ser mais explorados e difundidos no meio computacional.*

Abstract. Introduction: *Describes details about Object-Oriented Database Management Systems and their application in database development.* ***Theoretical Reference:*** *In this section the theoretical base used for the analyzes is detailed, giving a brief synthesis on the topics covered.* ***Case Study:*** *An application developed by the authors was used to make a study about what Object-Oriented Database Management Systems are and how they communicate with the database and the application. The steps that are fundamental to synthesize the objective of this article are explained.* ***Conclusions:*** *The study demonstrated how Object-Oriented Database Management Systems (OODMS) can be useful for applications that use Object-Oriented Programming and how this can facilitate the end-user's use as well as the work of the developers since the abstraction and the less complexity of the problems is something that the developers seek more and more, showing that OODMS can and should be more explored and diffused in the computational environment.*

1. Introdução

Sistema de modelo de dados tradicionais, como relacionais, de rede e hierárquicos, tem tido muito sucesso no desenvolvimento das tecnologias de banco de dados exigidas para muitas aplicações de banco de dados de negócio tradicionais. Porém, eles tem certas deficiências quando aplicações de um banco de dados mais

complexas precisam ser projetadas e implementadas [...]. Os bancos de dados de objeto foram propostos para atender a algumas necessidades dessas aplicações mais complexas [Elmasri and Navathe 1989]. É sabido que banco de dados e sistemas gerenciadores de banco de dados (SGBD) são fundamentais para diversos sistemas que exigem a manipulação de uma grande quantidade de dados. Da mesma forma, a orientação a objetos é uma técnica que está presente em diversas áreas da computação além de ser muito útil no desenvolvimento de sistemas de software.

Os bancos de dados orientados a objeto adotaram muitos dos conceitos que foram desenvolvidos originalmente para as linguagens de programação orientadas a objeto [Elmasri and Navathe 1989]. SGBDOO possuem integração a diversas linguagens de programação e a abstração de objetos assim como é feita usando os paradigmas da programação orientada a objetos. Diferentemente dos SGBDRs que fazem leituras lineares e bidirecionais nas tabelas, os SGBDOOs fazem um tipo especial de Objeto que é capaz de abstrair o mundo real para o banco de dados, deixando a aplicação menos complexa do ponto de vista do programador. Entretanto o SGBDOO ainda não é bem difundido no meio das aplicações devido a alguns pontos fracos de sua aplicação, por exemplo, a lentidão na busca de dados porém ele vem ganhando um espaço maior nos meios de aplicações robustas devido a sua capacidade de abstração e de facilitação no entendimento de cenários complexos. O objetivo deste artigo é abordar de forma conceitual o uso de um SGBD orientado a objetos utilizando como exemplo prático uma ferramenta desenvolvida pelos autores, projetada com o intuito de demonstrar programação orientada a objetos e o uso de um banco de dados orientado a objetos.

Este artigo foi divido em três partes. A primeira apresenta uma breve síntese sobre os conceitos de banco de dados e sistemas gerenciadores de banco de dados, baseada em pesquisas relacionadas ao assunto. A segunda parte explicita como foi feito o estudo de caso em cima da aplicação e a sua relação com SGBDOO. E por fim, a terceira parte é uma breve discussão sobre as conclusões obtidas após o estudo de caso.

2. Referencial Teórico

2.1. Banco de Dados

A tecnologia de banco de dados tem como fundamento básico permitir que os dados possam ser definidos e mantidos, independente dos sistemas de aplicação que venham a utilizá-los (independência DADO x PROCESSO). [Machado and Abreu 2004]

Os sistemas de Banco de Dados Orientado a Objetos (BDOO) permitem acesso direto aos dados de uma linguagem de programação orientada a objetos usando o sistema de tipo nativo da linguagem [Silberschatz et al. 2016]. Os BDOOs podem ser caracterizados como a junção dos conceitos de Orientação a Objetos com os conceitos de SGBDs, ou seja, eles tem todos os paradigmas de Orientação a Objetos, que são o encapsulamento, herança, polimorfismo e abstração, assim como são gerenciados por um SGBD, que receberá o nome de Sistema Gerenciador de Banco de Dados Orientado a Objetos.

2.2. Sistema Gerenciador de Banco de Dados

Um Sistema Gerenciador de Banco de Dados (SGBD) é uma ferramenta usada por desenvolvedores que os auxilia o gerenciamento, manipulação e organização dos dados de um banco de dados, permitindo assim que a aplicação não precise manipular esses dados diretamente.

Vários critérios são normalmente utilizados para classificar os SGBDs. O primeiro é o modelo de dados no qual o SGBD é baseado. O principal modelo de dados usado atualmente em muitos SGBDs comercial é o **modelo de dados relacional**. O **modelo de dados de objeto** foi implementado em alguns sistemas comerciais, mas não tem seu uso generalizado [Elmasri and Navathe 1989]. Esses SGBDs são chamados de SGBDR e SGBDOO, respectivamente.

Os SGBDs relacionais estão evoluindo continuamente e, em particular, tem incorporado muitos dos conceitos que foram desenvolvidos nos bancos de dados de objeto. Isso tem levado a uma nova classe de SGBDs, chamada **SGBD objeto-relacionais**. Assim, podemos categorizar os SGBDs com base no modelo de dados: relacionais, objeto, objeto-relacional, hierárquico, rede, entre outros [Elmasri and Navathe 1989].

O foco deste trabalho é no SGBDOO, que tem como principal característica a modelagem de estruturas complexas, assim como o comportamento do dado além de sua estrutura. Para isso é necessário que haja um banco de dados orientado a objetos, já que o SGBD é caracterizado em base do tipo de banco de dados, como foi citado anteriormente.

3. Estudo de caso

O ambiente de teste foi uma aplicação orientada a objetos desenvolvida em java com integração do SGBD SQlite e o objetivo do software é simular localmente como é feita a autenticação de um usuário em um servidor de autenticação e mostrar a comunicação de troca de pacotes entre o servidor e o cliente, através de um log de status. As informações do usuário são armazenadas em um banco de dados desenvolvido para esta finalidade.

3.1. Tecnologias utilizadas

Mecanismos de banco de dados SQL cliente/servidor se esforçam para implementar um repositório compartilhado de dados corporativos. Eles enfatizam escalabilidade, concorrência, centralização e controle. O SQLite se esforça para fornecer armazenamento de dados locais para aplicativos e dispositivos individuais. O SQLite enfatiza economia, eficiência, confiabilidade, independência e simplicidade.[Hipp and Wyrick & Company 2000]. O SQlite foi escolhido como SGBD do projeto, pois é uma biblioteca de código aberto que dispõe de um banco de dados na própria aplicação, o que faz com que o usuário final não tenha a necessidade de instalar programas adicionais, além de ser recomendado para aplicações de pequeno porte. Esse SGBD apesar de ter sido escrito em linguagem C, é um SGBD escrito no estilo Orientado a Objetos

A implementação da interface gráfica foi desenvolvida no Netbeans 8.2, um ambiente de desenvolvimento integrado gratuito e de código aberto, sendo muito utilizado para o desenvolvimento de softwares cliente-servidor, desktop e web. E a linguagem de programação escolhida para o desenvolvimento da aplicação foi o Java, que é uma das linguagens mais utilizadas do mundo, é simples, multiplataforma, além de ser uma linguagem orientada a objetos. Todo o projeto foi desenvolvido em base dos paradigmas de Programação Orientada a Obejtos.

3.2. Descrição da aplicação

A aplicação consiste em um Servidor de Autenticação onde um usuário final poderá se cadastrar e logo após realizar a autenticação no sistema com essas informações fornecidas, tudo isso através de uma interface amigável que foi desenvolvida com o propósito de ser de fácil uso. Como a aplicação foi pensada e desenvolvida nos paradigmas de Programação Orientada a Objetos, o SGBDOO SQLite foi o ideal para funcionar em conjunto com a aplicação, trazendo a tona os conceitos que são desejados nesse trabalho.

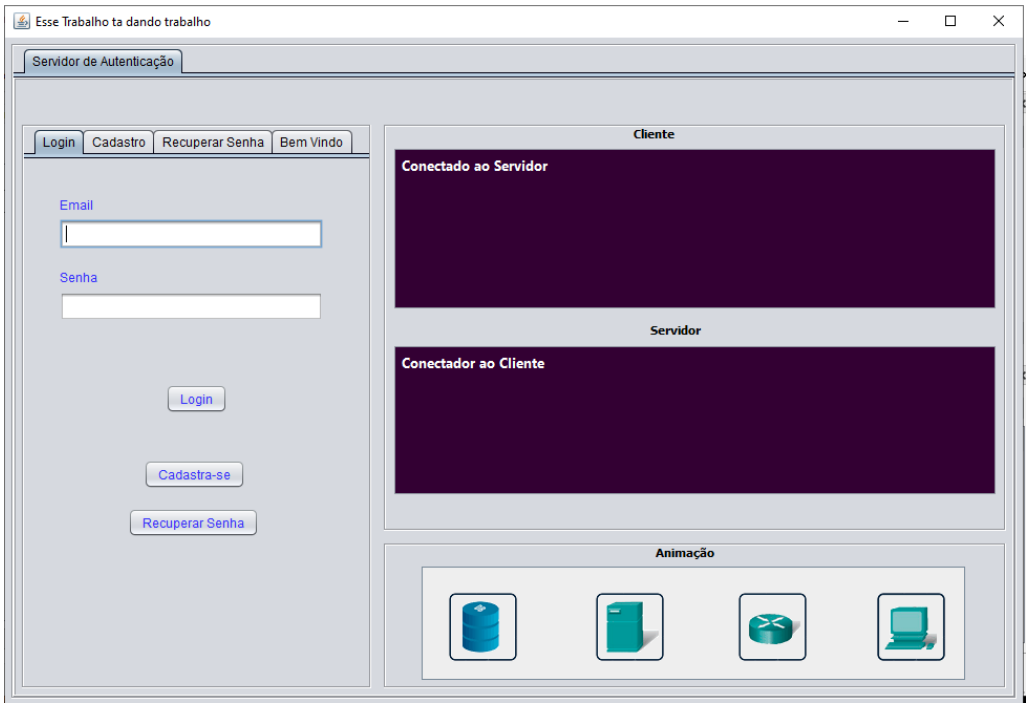

Figure 1. Tela de Login da aplicação

Os dados inseridos pelo usuário ao se cadastrar são armazenados no banco de dados previamente criado, essas informações poderão ser alteradas pelo próprio usuário, assim como poderão ser buscadas através da aplicação dependendo de qual operação o usuário deseja fazer. A figura 2 mostra a estrutura desse banco de dados:

```
CREATE TABLE "Usuario" (
    "id"    INTEGER PRIMARY KEY AUTOINCREMENT,
    "nome"  TEXT NOT NULL,
    "email" INTEGER NOT NULL UNIQUE,
    "senha" TEXT NOT NULL
);
```

Figure 2. Tabela do banco de dados

3.2.1. DBconnect

A comunicação entre a aplicação e o banco de dados é feita através de uma classe chamada DBconnect, essa classe é instanciada pelas outras classes que desejam fazer uma conexão com o banco de dados e através da Interface de Programação de Aplicativos JDBC é realizada a conexão com o SGBD. Essa classe faz uma tentativa de conexão com o banco de dados, ao ser estabelecida essa conexão as operações desejadas são realizadas.A seguir está o código fonte de como é feita essas conexão:

```
1
2  package Banco_de_Dados;
3
4  import static Interface_Servidor.Tela_Inicial.Texto_Log_Cliente;
5  import static Interface_Servidor.Tela_Inicial.Texto_Log_Servidor
       ;
6  import java.sql.Connection;
7  import java.sql.DriverManager;
8  import java.sql.SQLException;
9
10   public class DbConnect {
11
12     public static String MensagemCliente = "";
13     public static String MensagemServidor = "";
14
15     public static DbConnect getInstace() {
16         return new DbConnect();
17     }
18
19     public Connection getConnection() {
20         String connect_string = "jdbc:sqlite:Autenticacao.db";
21
22         Connection connection = null;
23         try {
24             MensagemCliente = MensagemCliente.concat("
                   INFORMA ES: Enviando dados ao servidor\n");
25             MensagemServidor = MensagemServidor.concat("
                   INFORMA ES: Enviando requisi es ao Banco de
                    Dados\n");
26              Texto_Log_Cliente.setText(MensagemCliente);
```

```
            Texto_Log_Servidor.setText(MensagemServidor);
            Class.forName("org.sqlite.JDBC");
            connection = DriverManager.getConnection(
               connect_string);
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
        return connection;
    }
}
```

Através da própria aplicação é possível realizar as operações de modificações do banco de dados, fazendo com que não fosse necesário inserir as queries toda vez que fosse necessário modificar o banco de dados, já que a proposta da aplicação é justamente demonstrar a modificação do banco de dados através da aplicação desenvolvida. A seguir está o código da operação de recuperação de dados de um usuário, operação esta que é realizada quando este usuário deseja realizar Login, já que é necessário verificar se este usuário está cadastrado no sistema:

```
  Connection connection = DbConnect.getInstace().getConnection()
     ;
  Statement statement = connection.createStatement();
                    ResultSet resultSet = statement.executeQuery
                       ("Select * from Usuario where email"
                            + " = '" + email + "'and senha='" +
                               senha + "'");
```

Todas as classes da aplicação fazem consultas no banco de dados através da classe DBconnect, sendo ela a mais importante das classes desenvolvidas.

4. Conclusões

Este artigo apresentou uma abordagem teórica sobre SGBDOO. Apesar da grande variedade de SGBDOOs no mercado, o uso dessa tecnologia é muito menor que o uso de um SGBD relacional, por exemplo. Foram mostradas as características de um SGBD e suas funcionalidade, sendo que a mais adequada para a proposta da aplicação era o SQLite, a sua escolha se deu por ser um SGBD que possui seu banco de dados dentro da própria aplicação, é de fácil manuseio e é recomendado para sistemas de pequeno porte, que foi o caso da aplicação na qual foi apresentada na seção 3.1, além de ser um SGBD criado especificanemten em paradigmas de Orientação a Objetos.

A aplicação criada tem como propósito ser um simulador, por isso não havia a necessidade de uma conexão de rede, que simulasse o cadastro e a autenticação de um usuário no sistema. Todas essas operações trabalharam diretamente no banco de dados, fazendo operações de busca, atualização e inserção dos dados desse banco de dados. A

classe DBconnect fazia com que essas operações pudessem ser realizadas, trabalhando como um intermediário entre a aplicação e o SGBD.

Com o desenvolvimento da aplicação foi verificado que as necessidades propostas foram atendidas, algumas das operações principais do SQlite foram utilizadas, como o insert e o update. O uso de uma linguagem orientada a objetos, que nesse contexto foi o Java, foi eficiente ao ser usada ao lado de um SGBDOO, foi possível concluir que isso pode facilitar tanto o uso do usuário final, quanto dos desenvolvedores. É importante notar que sem um banco de dados orientado a objetos, não é possível ter um SGBDOO, sendo que o que define o SGBDOO é o próprio banco de dados orientado a objetos. A abstração sempre foi uma peça chave para a orientação a objetos, aumentando a produtividade e reduzindo a complexidade no desenvolvimento do software. Apesar de SGBDOOs permitirem a abstração dos problemas e a diminuição de sua complexidade, o que caracterizam a sua vantagem em cima dos outros SGBDs, eles tem a desvantagem de serem mais lentos, já que a demora par buscar dados é algo que os desenvolvedores querem evitar.

5. Referências

References

Elmasri, R. and Navathe, S. B. (1989). *Sistemas de banco de dados*. Pearson Education do Brasil, 6th edition.

Hipp and Wyrick & Company, I. (2000). *Appropriate Uses For SQLite*.

Machado, F. and Abreu, M. (2004). *Projeto de Banco de Dados: Uma Visão Prática*. Editora Érica Ltda, 8th edition.

Silberschatz, A., Korth, H. F., and Sudarshan, S. (2016). *Sistema de banco de dados*. Elsevier, 5th edition.