

ANO
2023



UNINTER

**CADERNO DE RESPOSTAS DA ATIVIDADE
PRÁTICA DE:**

BANCO DE DADOS

**ALUNO: (GUILHERME BRANDÃO MURUSSI RU: RU:
4159376)**

**Caderno de Resposta Elaborado por:
Prof. MSc. Guilherme Ditzel Patriota**

Prática 01 – Banco de Dados.

I. Introdução

Nesse trabalho foi utilizado um software que funciona como um cofrinho para moedas nas cotações de Euro, Dólar e Real. Com esse programa é possível armazenar esses 3 tipos de moeda, removê-las, converter seus valores para a cotação do real e lista-las. Esse software teve uma integração com Banco de Dados para guardar essas informações de forma mais confiável, para que não houvesse perda de dados. No Banco de Dados houve a criação de triggers para caso as moedas inseridas fossem removidas, assim como triggers para caso adicionadas e alocadas na tabela de sua respectiva cotação (Dólar, Euro, Real). O principal dado para essas tabelas serão as moedas e seus valores agregados.

II. Modelo Conceitual (MER):

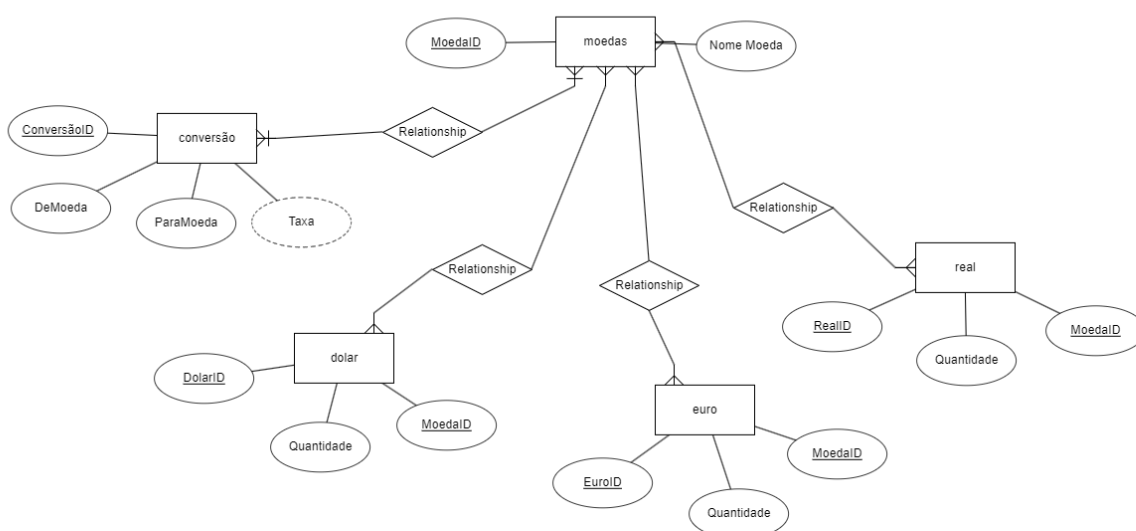


Figura 1: Diagrama MER com 5 entidades e seus relacionamentos. Usado no início da modelagem do banco de dados. Todas entidades possuem seus atributos e especificações de cada atributo.

III. Modelo Lógico (EER) após Normalização

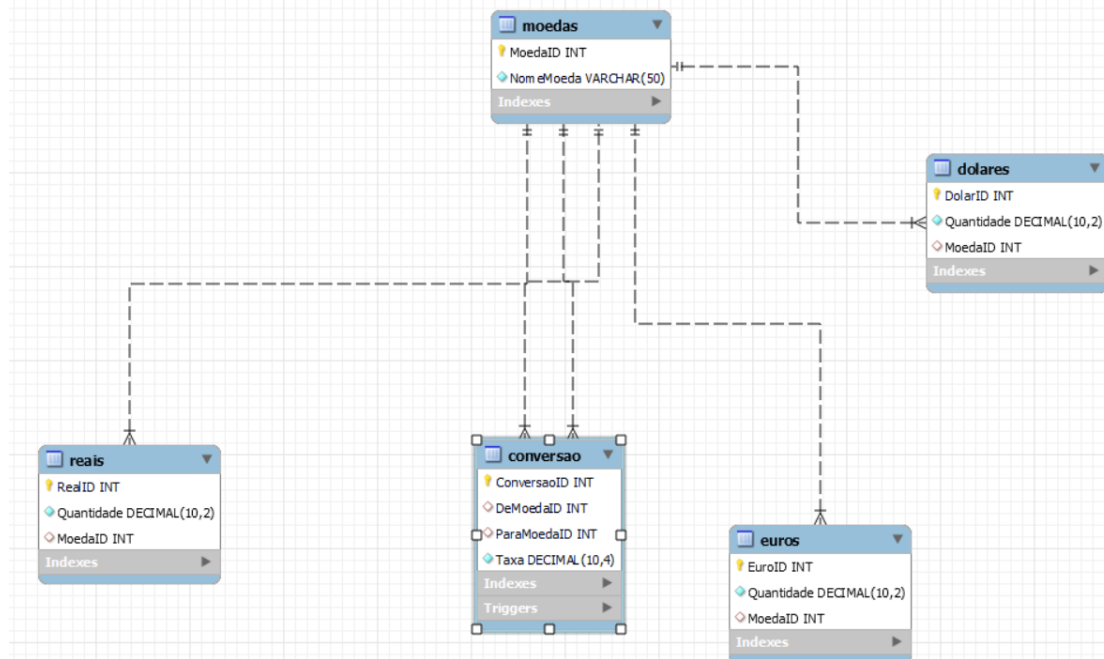


Figura 2: Modelo lógico com as 5 tabelas, uma sendo para moedas, outras 3 para as cotações e uma de conversão. Cada moeda possui um ID que é uma chave estrangeira nas cotações para guardar informação no Banco de Dados.

IV. Modelo Físico (SCHEMA)

```

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

```

```

-- Schema mydb

```

```

-- Schema cofrinho2.db

```

```

-- Schema cofrinho2.db

```

```

CREATE SCHEMA IF NOT EXISTS `cofrinho2.db` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci ;
USE `cofrinho2.db` ;

```

```

-- Table `cofrinho2.db`.`moedas`

```

```

CREATE TABLE IF NOT EXISTS `cofrinho2.db`.`moedas` (
  `MoedaID` INT NOT NULL,
  `NomeMoeda` VARCHAR(50) NOT NULL,
  PRIMARY KEY (`MoedaID`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```

```

-- Table `cofrinho2.db`.`conversao`

```

```

CREATE TABLE IF NOT EXISTS `cofrinho2.db`.`conversao` (
  `ConversaoID` INT NOT NULL,

```



```

`DeMoedaID` INT NULL DEFAULT NULL,
`ParaMoedaID` INT NULL DEFAULT NULL,
`Taxa` DECIMAL(10,4) NOT NULL,
PRIMARY KEY (`ConversaoID`),
INDEX `DeMoedaID` (`DeMoedaID` ASC) VISIBLE,
INDEX `ParaMoedaID` (`ParaMoedaID` ASC) VISIBLE,
CONSTRAINT `conversao_ibfk_1`
FOREIGN KEY (`DeMoedaID`)
REFERENCES `cofrinho2.db`.`moedas` (`MoedaID`),
CONSTRAINT `conversao_ibfk_2`
FOREIGN KEY (`ParaMoedaID`)
REFERENCES `cofrinho2.db`.`moedas` (`MoedaID`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

-----
-- Table `cofrinho2.db`.`dolares`
-----
CREATE TABLE IF NOT EXISTS `cofrinho2.db`.`dolares` (
  `DolarID` INT NOT NULL,
  `Quantidade` DECIMAL(10,2) NOT NULL,
  `MoedaID` INT NULL DEFAULT NULL,
  PRIMARY KEY (`DolarID`),
  INDEX `MoedaID` (`MoedaID` ASC) VISIBLE,
  CONSTRAINT `dolares_ibfk_1`
  FOREIGN KEY (`MoedaID`)
  REFERENCES `cofrinho2.db`.`moedas` (`MoedaID`))
  ENGINE = InnoDB
  DEFAULT CHARACTER SET = utf8mb4
  COLLATE = utf8mb4_0900_ai_ci;

-----
-- Table `cofrinho2.db`.`euros`
-----
CREATE TABLE IF NOT EXISTS `cofrinho2.db`.`euros` (
  `EuroID` INT NOT NULL,
  `Quantidade` DECIMAL(10,2) NOT NULL,
  `MoedaID` INT NULL DEFAULT NULL,
  PRIMARY KEY (`EuroID`),
  INDEX `MoedaID` (`MoedaID` ASC) VISIBLE,
  CONSTRAINT `euros_ibfk_1`
  FOREIGN KEY (`MoedaID`)
  REFERENCES `cofrinho2.db`.`moedas` (`MoedaID`))
  ENGINE = InnoDB
  DEFAULT CHARACTER SET = utf8mb4
  COLLATE = utf8mb4_0900_ai_ci;

-----
-- Table `cofrinho2.db`.`reais`
-----
CREATE TABLE IF NOT EXISTS `cofrinho2.db`.`reais` (
  `RealID` INT NOT NULL,
  `Quantidade` DECIMAL(10,2) NOT NULL,
  `MoedaID` INT NULL DEFAULT NULL,
  PRIMARY KEY (`RealID`),
  INDEX `MoedaID` (`MoedaID` ASC) VISIBLE,
  CONSTRAINT `reais_ibfk_1`
  FOREIGN KEY (`MoedaID`)
  REFERENCES `cofrinho2.db`.`moedas` (`MoedaID`))
  ENGINE = InnoDB
  DEFAULT CHARACTER SET = utf8mb4
  COLLATE = utf8mb4_0900_ai_ci;

USE `cofrinho2.db`;

DELIMITER $$
USE `cofrinho2.db` $$
CREATE
DEFINER='root'@`localhost`
TRIGGER `cofrinho2.db`.`UpdateEurosOnConversionInsert`
AFTER INSERT ON `cofrinho2.db`.`conversao`

```



```
FOR EACH ROW  
BEGIN  
    DECLARE novoValor DECIMAL(10, 2);  
    SELECT NEW.Taxa INTO novoValor;  
    UPDATE Euros SET Quantidade = Quantidade * novoValor WHERE MoedaID = 1;  
END$$
```

```
DELIMITER ;
```

```
SET SQL_MODE=@OLD_SQL_MODE;  
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;  
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Código-fonte (COD)

```
package empresa;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.util.Scanner;  
  
public class CofrinhoApp {  
  
    private static Connection connection;  
  
    public static void main(String[] args) {  
        try {  
            // Configurar a conexão com o banco de dados MySQL  
            String url = "jdbc:mysql://localhost:3306/cofrinho2.db";  
            String user = "root";  
            String password = "G@nese@nimes1!";  
            connection = DriverManager.getConnection(url, user, password);  
  
            // Menu principal  
            Scanner scanner = new Scanner(System.in);  
            while (true) {  
                System.out.println("Menu do Cofrinho:");  
                System.out.println("1. Adicionar moedas");  
                System.out.println("2. Retirar moedas");  
                System.out.println("3. Listar moedas");  
                System.out.println("4. Calcular valor em Real");  
                System.out.println("5. Fechar programa");  
  
                int escolha = scanner.nextInt();  
                scanner.nextLine(); // Limpar o buffer de entrada  
  
                switch (escolha) {  
                    case 1:  
                        adicionarMoedas();  
                        break;  
                    case 2:  
                        retirarMoedas();  
                        break;  
                    case 3:  
                        listarMoedas();  
                        break;  
                    case 4:  
                        calcularValorEmReal();  
                        break;  
                    case 5:  
                        break;  
                }  
            }  
        } catch (SQLException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

```
        System.out.println("Programa encerrado.");
        return;
    default:
        System.out.println("Opção inválida. Tente novamente.");
        break;
    }
}

} catch (SQLException e) {
    e.printStackTrace();
} finally {
    if (connection != null) {
        try {
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

}

private static void adicionarMoedas() throws SQLException {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Digite o valor da moeda: ");
    double valor = scanner.nextDouble();
    scanner.nextLine(); // Limpar o buffer de entrada

    System.out.println("Digite o tipo de moeda (Euro, Dólar, Real): ");
    String tipoMoeda = scanner.nextLine();

    // Verificar se o tipo de moeda é válido
    if (!tipoMoeda.equals("Euro") && !tipoMoeda.equals("Dólar") &&
!tipoMoeda.equals("Real")) {
        System.out.println("Tipo de moeda inválido.");
        return;
    }

    // Inserir a moeda no banco de dados
    String sql = "INSERT INTO " + tipoMoeda + " (Quantidade) VALUES (?)";
    PreparedStatement preparedStatement = connection.prepareStatement(sql);
    preparedStatement.setDouble(1, valor);
    preparedStatement.executeUpdate();
    System.out.println("Moedas adicionadas com sucesso.");
}

private static void retirarMoedas() throws SQLException {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Digite o valor da moeda a ser retirada: ");
    double valor = scanner.nextDouble();
    scanner.nextLine(); // Limpar o buffer de entrada

    System.out.println("Digite o tipo de moeda a ser retirada (Euro, Dólar,
Real): ");
    String tipoMoeda = scanner.nextLine();

    // Verificar se o tipo de moeda é válido
    if (!tipoMoeda.equals("Euro") && !tipoMoeda.equals("Dólar") &&
!tipoMoeda.equals("Real")) {
        System.out.println("Tipo de moeda inválido.");
        return;
    }
}
```



```
// Verificar se há moedas suficientes para retirar
String sql = "SELECT Quantidade FROM " + tipoMoeda;
PreparedStatement preparedStatement = connection.prepareStatement(sql);
ResultSet resultSet = preparedStatement.executeQuery();
if (resultSet.next()) {
    double quantidadeAtual = resultSet.getDouble("Quantidade");
    if (valor > quantidadeAtual) {
        System.out.println("Quantidade insuficiente de moedas para
retirar.");
        return;
    }
}

// Atualizar a quantidade de moedas no banco de dados
sql = "UPDATE " + tipoMoeda + " SET Quantidade = Quantidade - ? WHERE
Quantidade >= ?";
preparedStatement = connection.prepareStatement(sql);
preparedStatement.setDouble(1, valor);
preparedStatement.setDouble(2, valor);
int rowsAffected = preparedStatement.executeUpdate();
if (rowsAffected > 0) {
    System.out.println("Moedas retiradas com sucesso.");
} else {
    System.out.println("Falha ao retirar moedas.");
}
}

private static void listarMoedas() throws SQLException {
    System.out.println("Moedas no cofrinho:");
    listarTipoMoeda("Euro");
    listarTipoMoeda("Dólar");
    listarTipoMoeda("Real");
}

private static void listarTipoMoeda(String tipoMoeda) throws SQLException {
    String sql = "SELECT Quantidade FROM " + tipoMoeda;
    PreparedStatement preparedStatement = connection.prepareStatement(sql);
    ResultSet resultSet = preparedStatement.executeQuery();
    if (resultSet.next()) {
        double quantidade = resultSet.getDouble("Quantidade");
        System.out.println(tipoMoeda + ": " + quantidade);
    }
}

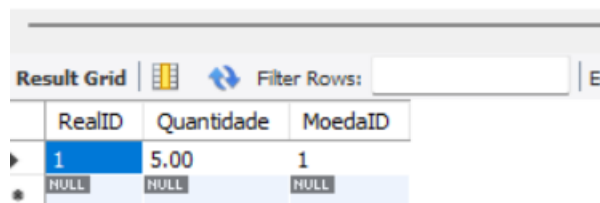
private static void calcularValorEmReal() throws SQLException {
    double valorTotal = 0.0;
    String[] tiposMoedas = {"Euro", "Dólar", "Real"};
    for (String tipoMoeda : tiposMoedas) {
        String sql = "SELECT SUM(Quantidade * (SELECT Taxa FROM Conversao
WHERE DeMoedaID = (SELECT MoedaID FROM Moedas WHERE NomeMoeda = ?) AND
ParaMoedaID = 3)) AS ValorTotal FROM " + tipoMoeda;
        PreparedStatement preparedStatement =
connection.prepareStatement(sql);
        preparedStatement.setString(1, tipoMoeda);
        ResultSet resultSet = preparedStatement.executeQuery();
        if (resultSet.next()) {
            double valor = resultSet.getDouble("ValorTotal");
            valorTotal += valor;
        }
    }
    System.out.println("Valor total em Real: " + valorTotal);
}
```

}

V.

VI. Prints de funcionamento do software.

```
Opções:
1. Adicionar moeda
2. Retirar moeda
3. Listar moedas
4. Calcular valor em Real
5. Fechar cofrinho
Escolha uma opção: 1
Digite o valor da moeda: 5
Digite o tipo de moeda (Real, Euro, Dólar, etc.): Real
Digite o país de origem da moeda: Brasil
1 • select * from reais;
```



RealID	Quantidade	MoedaID
1	5.00	1
NULL	NULL	NULL

(por algum motivo meu interpretador coloca esses caracteres em palavras com acento)

VII. Conclusão

O Objetivo do projeto foi a integração de um banco de dados em um projeto JAVA já existente, porém não imaginei que teria que mudar tanto as classes para que fosse possível a utilização do BD.