Programação Avançada - AULA 11

Matheus Moresco

Análise e Desenvolvimento de Sistemas- 5º Período

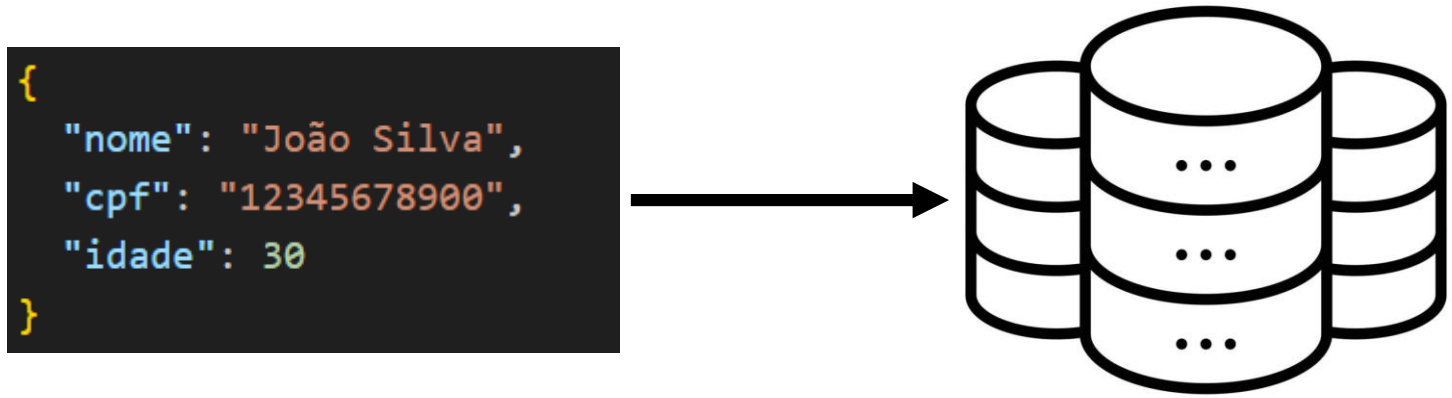
2025/01

Introdução

- Introdução a persistência de dados
  - O que é?
  - Importância
  - Vantagens
- JDBC
  - O que é?
  - Como usar?

Persistência de dados

- A **persistência de dados** é a capacidade de armazenar informações de forma permanente, garantindo que elas permaneçam disponíveis mesmo após o encerramento de um programa ou desligamento do sistema.



Por que a persistência de dados é importante?

- ✓Recuperação de dados após falhas
- ✓Compartilhamento de informações entre usuários
- ✓Manutenção do estado de uma aplicação
- ✓Segurança e integridade dos dados

Formas de Persistência de Dados

- **Arquivos**: Armazenamento simples em formatos como TXT, JSON e XML
- **Bancos de Dados Relacionais (SQL):** Estruturados em tabelas, usam SQL para manipulação
- **Bancos de Dados NoSQL**: Flexíveis e escaláveis, armazenam dados em documentos, chave-valor, colunas ou grafos
- **ORM (Object-Relational Mapping):** Frameworks como Hibernate e JPA facilitam a persistência em bancos relacionais

Persistência de dados com Arquivos

- Armazenamento em arquivos
  - Simples e fácil de implementar
  - Formatos comuns: TXT, CSV, JSON, XML
  - Pouco eficiente para grandes volumes de dados

- **Limitações:**
  - Dificuldade em buscas e modificações
  - Falta de estrutura para grandes volumes de dados

Persistência de dados com Arquivos

```java
import java.io.FileWriter;
import java.io.IOException;

public class PersistenciaArquivo {
    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        try (FileWriter escritor = new FileWriter(fileName:"./dados.txt", append:true)) {
            escritor.write(str:"Usuário: João\n");
            System.out.println(x:"Dados salvos no arquivo.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Bancos de Dados Relacionais (SQL)

- **O que são bancos relacionais?**
  - Dados organizados em tabelas com relações definidas
  - Utilização da **linguagem SQL**

- **Principais conceitos:**
  - **Tabelas, colunas, chaves primárias e estrangeiras**
  - **Normalização** (evitar redundância)
  - **ACID** (Atomicidade, Consistência, Isolamento, Durabilidade)

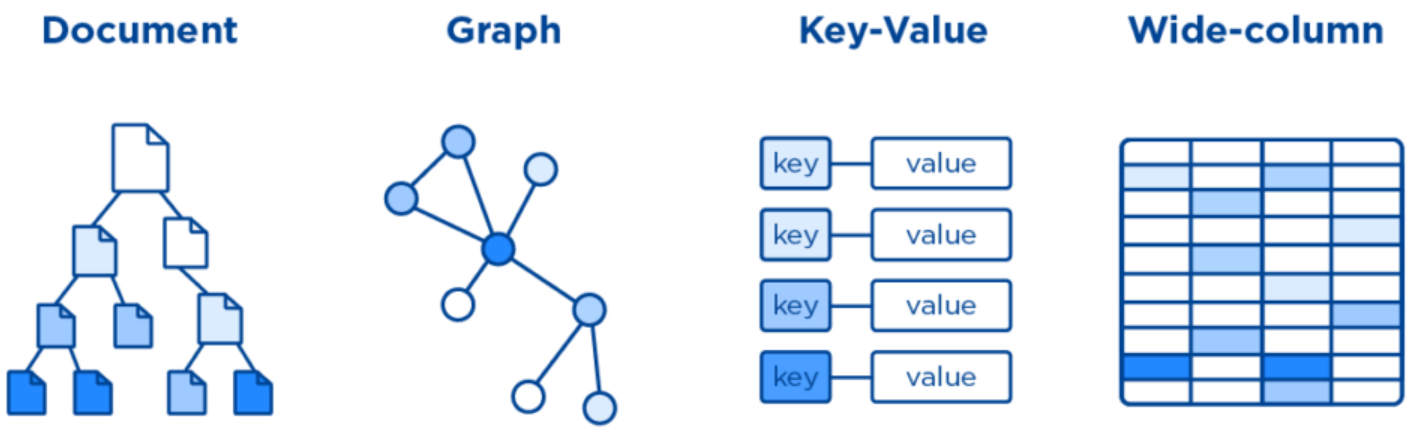
Bancos de Dados Relacionais (SQL)

```sql
CREATE TABLE usuarios (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(100),
    email VARCHAR(100) UNIQUE
);

UPDATE `usuarios`
    SET `email` = 'novoemail@email.com'
    WHERE (`id` = '3');

INSERT INTO `usuarios` (`nome`, `email`)
    VALUES ('user4', 'user4@email.com');

SELECT * FROM `usuarios`;
```

Id	Nome	email
1	User1	user1@email.com
2	User2	user2@email.com
3	User3	user3@email.com

Bancos de Dados NoSQL

- O que é NoSQL?
  - Alternativa flexível aos bancos relacionais
  - Estrutura de armazenamento não tabular (documentos, chave-valor, colunas, grafos)



Bancos de Dados NoSQL

- Exemplos de bancos NoSQL
  - Documentos: MongoDB
  - Chave-valor: Redis
  - Colunas: Cassandra
  - Grafos: Neo4j

Bancos de Dados NoSQL

- **Vantagens:**
  - Eficiência em consultas complexas
  - Suporte para transações seguras
  - Controle de integridade dos dados

- **Desvantagens:**
  - Pode ter desempenho inferior em grandes volumes de dados não estruturados
  - Requer modelagem e manutenção adequadas

Frameworks e Ferramentas para Persistência

- **JDBC** – Interface Java para Bancos Relacionais
  - Comunicação direta com bancos SQL
  - Uso de PreparedStatement para evitar SQL Injection

- **ORM** (Object-Relational Mapping)
  - Abstração para interação com banco de dados usando objetos Java
  - Principais ferramentas: JPA, Hibernate, Spring Data JPA

Frameworks e Ferramentas para Persistência

- **Vantagens do ORM:**
  - Redução de código SQL manual
  - Melhor manutenção e escalabilidade

- **Desvantagens do ORM:**
  - Pode ter desempenho inferior em consultas complexas
  - Curva de aprendizado inicial

Boas Praticas

Boas práticas essenciais:

- ✓Uso de PreparedStatements para evitar SQL Injection
- ✓Implementação de camadas de persistência separadas do código de lógica de negócio
- ✓Uso de pools de conexão para otimização (ex: HikariCP)
- ✓Monitoramento e otimização de consultas SQL

Erros comuns e como evitá-los:

- X Esquecer de fechar conexões de banco
- X Não tratar exceções corretamente
- X Criar consultas SQL pouco eficientes

Exemplo Prático

- Testar a Persistência de dados em Arquivos;