CENTRO DE ENSINO SUPERIOR E DESENVOLVIMENTO – CESED

CENTRO UNIVERSITÁRIO – UNIFACISA

CURSO: SISTEMAS DE INFORMAÇÃO

PROFESSOR: JOSÉ ANDERSON RODRIGUES DE SOUZA

COMPONENTE CURRICULAR: CONECTAR BANCO DE DADOS COM P-O-O

EQUIPE: WÂNDERSON PIO, AYRTON VILLENEUVE, IVO MIRANDA,

PEDRO MELO, ÍTALO DANTAS.

ETAPA 02 – LISTA DE EXERCÍCIOS

1. O que você entende por Persistência de Dados?

A persistência de dados, na computação, refere-se ao armazenamento não-volátil de dados, por exemplo, o armazenamento em um dispositivo físico como um disco rígido, é a garantia de que um determinado dado foi salvo corretamente, sem modificações.

A persistência dos dados tem o objetivo de garantir que as informações serão armazenadas em um meio em que possam ser recuperadas de forma consistente. Ou seja, são registros permanentes e que não são perdidos quando há o encerramento da sessão.

2. Quais as diferenças entre objetos transientes e objetos persistentes? Explique.

Objetos transientes são objetos que existem apenas enquanto a aplicação que os criou continuar executando. Após o término da aplicação eles deixam de existir. Geralmente, os objetos transitórios são criados na memória durante a execução do programa, mas não estão vinculados a nenhum registro ou entrada em um sistema de armazenamento de dados. Esses objetos não foram persistidos (ou seja, não foram salvos) em nenhum banco de dados ou sistema de armazenamento permanente.

Um objeto persistente está associado a um armazenamento permanente, como um banco de dados. Esses objetos têm um ciclo de vida que vai além da execução do programa. Eles podem ser criados, lidos, atualizados e excluídos em relação ao armazenamento permanente. Os objetos persistentes são persistidos em um banco de dados ou sistema de armazenamento

permanente. Suas alterações refletem-se no armazenamento, e eles podem ser recuperados posteriormente a partir do armazenamento.

3. Qual a principal diferença entre Banco de Dados Relacional e Banco de Dados Orientada a Objetos?

Bancos de dados relacionais usam como base a relação entre informações e diferentes fatores. Na prática, você pode visualizá-lo como uma tabela, em que cada coluna representa um tipo de informação associada a uma linha, os dados são organizados em tabelas relacionadas, onde cada tabela possui linhas (registros) e colunas (atributos). As relações entre as tabelas são definidas por chaves estrangeiras e chaves primárias.

Um banco de dados orientado a objetos é organizado na forma de diferentes objetos, cada objeto pode conter atributos e métodos, os quais contém arquivos e informações agrupados, além dos procedimentos para sua leitura e processamento, ele não segue uma “lógica” rígida preestabelecida, criando apenas blocos de informação, cada um com um Identificador de objeto associado para identificação.

Um banco de dados relacional, cada informação individual é armazenada como um arquivo separado e independente dos demais, não havendo suporte para o armazenamento persistente de objetos com estrutura complexa. O foco fica em dados mais diretos.

Já os bancos de dados orientados a objetos utilizam identificadores para rotular cada objeto, junto com técnicas de indexação para localização de páginas em disco. Como resultado, o sistema consegue dar suporte para objetos de estruturação mais complexa.

Um banco orientado a objetos, por ter menos restrições estruturais, pode conter muito mais informação e em maior variedade. Isso resulta em um sistema de dados mais flexível diante de múltiplos formatos de arquivo, por exemplo. Já um banco de dados relacional contém um nível menor de complexidade, o que resulta em um volume relativamente menor de informação.

Um banco de dados orientado a objetos, como já mencionamos, utiliza Identificadores de objetos, que são como rótulos. Ao fazer uma busca por um desses rótulos, o sistema traz para você o conjunto de arquivos e informações contidos nele. E, dentro desse objeto, pode haver outros tipos de hierarquias e relações internas.

Já o modelo relacional, por sua vez, faz uso de duas chaves relacionais. A primária, como já mencionamos, é a base, normalmente representada na primeira coluna da tabela e identifica cada linha. Em seguida, é usada uma chave externa, que é relacionada com a primária. Juntas, ambas servem como coordenada para a localização da informação buscada.

Da mesma forma, pode haver chaves secundárias ou terciárias, que atuam como filtros de busca. Dessa forma, é possível usar outras chaves e separar aquelas que têm uma informação específica em determinada área, ou aquelas que têm uma área vazia.

4. Qual o objetivo do mapeamento objeto-relacional (ORM)?

ORM é uma ferramenta que utiliza mecanismos que possibilitam a manipulação dos objetos por meio do mapeamento entre sistemas orientados a objetos e banco de dados relacionais.

O ORM cria uma ligação entre a aplicação e o banco de dados, através deste processo os objetos podem ser persistidos em um banco de dados relacional sem a necessidade de conversão de objetos para um modelo relacional

5. Defina:

a) Dados estruturados;

Dados estruturados são dados em um formato padronizado, com estrutura bem definida e que obedecem a um modelo de dados e seguem uma ordem persistente e de fácil acesso por humanos e programas. Esse tipo de dados geralmente é armazenado em um banco de dados.

Formato consistente, relacionamentos claros, facilidade de consulta, compatibilidade com sistemas

b) Dados semiestruturados;

Dados semiestruturados são um tipo de dados que não possuem uma estrutura rígida e predefinida, como os dados estruturados em bancos de dados relacionais, mas ainda possuem alguma estruturação básica que permite a organização dos dados. A principal característica dos dados semiestruturados é que eles não seguem um esquema fixo, o que os torna mais flexíveis e adequados para lidar com informações que podem variar em formato e conteúdo.

Os dados semiestruturados podem ser representados em diversos formatos, como XML, JSON, YAML, HTML, texto simples ou até mesmo em formatos proprietários. Eles são frequentemente usados em documentos, páginas da web, logs de servidores, mensagens de rede e outras fontes de dados variáveis.

c) Dados não estruturados.

Os dados não estruturados possuem uma estrutura totalmente inversa e não podem ser organizados em tabelas.

Esse tipo de dados não tem uma estrutura bem definida, não tem um padrão pré-estabelecido. São dados flexíveis e dinâmicos, podendo ser compostos por diversos elementos diferentes dentro um todo.

Um exemplo simples de dados não estruturados são os dados de redes sociais. Esse é um típico exemplo, pelo grande número de textos, imagens, vídeos e diversos outros que são criados diariamente a partir do uso das tecnologias.

Na verdade, a maior parte dos dados gerados no mundo todo são dados não estruturados, tendo aproximadamente o percentual de 80%. Esses dados têm uma complexidade um pouco maior para análise, já que são informações de difícil processamento e recuperação, pois não contam com componentes necessários para a sua identificação. Porém, com a disponibilização de tecnologias que nos ajudam nesse quesito, essa complexidade da análise de dados tem diminuído.

Esse tipo de dados inclui vídeos, relatórios, pesquisas, documentos do Word, imagens e memorandos.

6. Qual o nome da biblioteca responsável pela extração/captura de dados disponíveis em arquivos HTML ou XML? Explique.

O Jsoup é uma biblioteca Java amplamente utilizada para web scraping e análise de documentos HTML. Ele fornece recursos poderosos para analisar páginas da web, extrair informações de HTML e manipular dados HTML de forma programática. O Jsoup é especialmente útil quando você deseja extrair dados de sites da web e processar esses dados em aplicativos Java.

7. Os arquivos do tipo XML (eXtensible Markup Language) surgiram como forma de estruturação e troca de dados pela internet. Dentre suas principais característica preencha os seguintes questionamentos:

a) Sintaxe inicial na primeira linha do arquivo.xml

A primeira linha de um documento XML é geralmente chamada de "prolog" e contém informações sobre a versão do XML e a codificação de caracteres usada.

<?xml version="1.0" encoding="UTF-8"?>

b) Os dados são organizados em formato hierárquico ou tabular?

Os dados em XML são organizados em formato hierárquico. O XML segue um modelo de dados hierárquico, onde os elementos (ou "tags") podem conter outros elementos aninhados. Isso permite a criação de estruturas de dados complexas, semelhantes a árvores, que representam relações hierárquicas entre os dados. Cada elemento pode ter atributos e pode conter texto ou outros elementos, criando uma estrutura de árvore que reflete a organização dos dados.

c) Quais são as formas de representação de um documento XML. Justifique.

A representação textual ocorre em forma de texto em uma sequência estabelecida e com regras a serem seguidas, já a estrutura de árvore é um modo mais visual mostrando as “dependências e/ou relações” entre os dados. Essa são algumas das regras utilizadas na estruturação de u documento XML: cada elemento possui um único pai (Exceção: O elemento raiz); cada elemento possui um número arbitrário de irmãos e filhos; um elemento sem filho é denominado folha; todo documento XML deve possuir uma raiz; todas as tags devem ser fechadas; as tags devem estar bem aninhadas; as tags XML são “case sensitive”; um elemento pode ter conteúdo vazio, <fone/>, <nada> </nada>.

8. Elabore um documento xml sobre produtos disponíveis para venda em empresas do comércio eletrônico/móveis/imóveis/roupas, a partir das seguintes condições:

- O produto deve possuir 5 características;
- Cada produto deve ter um nome de identificação;
- No documento deverá ter pelo menos dois produtos preenchidos.

```
<?xml version="1.0" encoding="UTF-8"?>
<produtos>
  <produto>
    <nome>Camiseta Casual</nome>
    <caracteristicas>
      <cor>Azul</cor>
      <tamanho>M</tamanho>
      <material>Algodão</material>
      <estampa>Listrado</estampa>
      <preco>29.99</preco>
    </caracteristicas>
  </produto>
  <produto>
    <nome>Calça Jeans</nome>
    <caracteristicas>
      <cor>Preto</cor>
      <tamanho>36</tamanho>
      <material>Jeans</material>
      <estampa>Sem estampa</estampa>
      <preco>49.99</preco>
    </caracteristicas>
  </produto>
  <!-- Adicione mais produtos conforme necessário -->
</produtos>
```

9. Defina o que é um documento JSON e quais suas principais características.

Json é um arquivo que contém uma série de dados estruturados em formato texto e é utilizado para transferir informações entre sistemas. É importante dizer que, apesar de sua origem ser por meio da linguagem JavaScript, JSON não é uma linguagem de programação.

Em vez disso, é uma notação para a transferência de dados que segue um padrão específico. Por isso, pode ser amplamente utilizada em diferentes linguagens de programação e sistemas.

Os dados contidos em um arquivo no formato JSON devem ser estruturados por meio de uma coleção de pares com nome e valor ou ser uma lista ordenada de valores. são organizados em uma estrutura hierárquica, onde os objetos podem conter outros objetos (aninhamento). Isso permite a representação de dados complexos e relacionados. Seus elementos devem conter:

chave: corresponde ao identificador do conteúdo. Por isso, deve ser uma string delimitada por aspas;

valor: representa o conteúdo correspondente e pode conter os seguintes tipos de dados: string, array, object, number, boolean ou null.

10. O que significa o processo de serialização (JSON.stringify) e desserialização (JSON.parse) de documentos do tipo JSON?

A serialização é o processo de converter uma estrutura de dados, geralmente um objeto ou um valor em uma linguagem de programação, em uma sequência de caracteres no formato JSON.

```
const pessoa = { nome: "João", idade: 30 };

const pessoaJSON = JSON.stringify(pessoa);

console.log(pessoaJSON); // Saída: '{"nome":"João","idade":30}'
```

A desserialização é o processo inverso da serialização. Ela envolve a conversão de uma sequência de caracteres no formato JSON de volta para uma estrutura de dados nativa em uma linguagem de programação.

```
const pessoaJSON = '{"nome":"João","idade":30}';

const pessoa = JSON.parse(pessoaJSON);

console.log(pessoa); // Saída: { nome: 'João', idade: 30 }
```

11. Faça um exemplo de documento JSON a partir de dados sobre serviços de vendas online.

• Utilize dados do tipo, string, inteiro, array e objetos.

```
{
 "empresa": "Minha Loja Online",
 "website": "https://www.minhaloja.com",
 "fundacao": 2010,
 "categorias": ["Eletrônicos", "Moda", "Casa e Decoração"],
 "produtos": [
  {
   "id": 1,
   "nome": "Smartphone Modelo X",
   "preco": 699.99,
   "estoque": 50,
   "desconto": true
  },
  {
   "id": 2,
   "nome": "Camiseta Estampada",
   "preco": 29.99,
   "estoque": 100,
   "desconto": false
  }
 ],
 "endereco": {
  "rua": "Rua das Vendas, 123",
  "cidade": "Cidade Online",
  "estado": "Estado Virtual",
  "cep": "12345-678"
 }
}
```


Neste exemplo:

"empresa" é uma string que representa o nome da empresa de vendas online.

"website" é uma string que representa o URL do site da empresa.

"fundacao" é um número inteiro que representa o ano de fundação da empresa.

"categorias" é um array de strings que lista as categorias de produtos oferecidos pela empresa.

"produtos" é um array de objetos que representa produtos individuais. Cada produto possui um id, um nome, um preço, uma quantidade em estoque e um valor booleano desconto.

"endereço" é um objeto que representa o endereço da empresa, com campos para a rua, cidade, estado e cep.

12. Quais são as principais diferenças entre documentos do tipo JSON e XML.

Notação

A primeira diferença entre os dois modelos é a forma de fazer a notação dos dados. O JSON utiliza uma notação simples, enquanto o XML utiliza uma estrutura de tags personalizadas para representar os objetos. Além disso, elas devem conter o par, ou seja, a tag de abertura e a de fechamento.

Outra característica da notação XML é que o seu conteúdo não precisa ser delimitado com aspas, como acontece com os textos no formato JSON. Nele, o que indica o início e o fim das informações são as tags de abertura e fechamento.

Tipos de dados

O formato XML suporta diferentes tipos de dados, entre eles: imagens e gráficos, o que não é possível transmitir no formato JSON, pois ele só oferece suporte a números e textos. Em contrapartida, o XML não oferece suporte a arrays.

Codificação

A codificação representa as formas de conversão para o formato binário suportadas pelo modelo. O JSON utiliza o formato UTF-8, enquanto o XML oferece essa e outras opções. É importante dizer que o UTF-8 é o formato mais utilizado em sites na internet, como o WordPress e muitos outros.

Facilidade de leitura e comentários

Os arquivos JSON são fáceis de entender, pois sua estrutura e notação são bem simples. Já o XML é mais estruturado e, portanto, com a interpretação mais complexa. Outra diferença é com relação aos comentários no arquivo, que são permitidos apenas no modelo XML.

13. Para que serve utilizar JDBC com Sistemas de Gerenciamento de Banco de Dados.

O JDBC é uma API — Application Programming Interface — do Java que contém uma série de classes e interfaces para realizar a comunicação entre uma aplicação desenvolvida em Java e o banco de dados utilizado por ela. É um recurso antigo do Java, pois está disponível desde a ver o JDBC desempenha um papel crucial na conectividade de aplicativos Java a sistemas de gerenciamento de banco de dados, tornando possível o acesso e a manipulação de dados de maneira eficiente e segura. A API é composta pelos pacotes java.sql e javax.sql.

14. Quais são os principais componentes durante a implementação do JDBC? Explique.

A API JDBC é composta por dois componentes centrais. Em primeiro lugar, podemos falar dos pacotes (Java.sql e Javax.sql) que contêm as classes e interfaces que padronizam a comunicação da aplicação Java com uma base de dados

Outro item importante são os drivers, verdadeiros responsáveis pela conexão e interação com um banco específico. Um driver JDBC é uma classe que implementa a interface java.sql.Driver. Muitos drivers são totalmente desenvolvidos com o uso de Java, o que colabora para serem carregados de maneira dinâmica.

Tipos de drives:

Tipo 1: A JDBC-ODBC possibilita o acesso a drivers do tipo ODBC, um padrão já consolidado para o acesso a bases de dados. Tipo 2: Neste tipo de driver é implementado o protocolo do proprietário do banco de dados. Ele transforma as chamadas JDBC em chamadas do banco com o uso da API proprietária. Tipo 3: Este tipo de driver faz a conversão das chamadas JDBC em outras chamadas do banco de dados, que são direcionadas para uma camada intermediária de software, ou middleware. Assim, a chamada será convertida para o protocolo do banco. Tipo 4: São escritos puramente em Java e implementam o protocolo proprietário do banco de dados. No geral, têm desempenho superior, já que acessam diretamente o SGBD, sistema gerenciador de banco de dados.

15. Cites restrições sobre a utilização do JDBC para sistemas atuais.

O JDBC está presente em sua maioria em projetos mais antigos dentro do sistema, porém ainda ocorre sua implementação em projetos mais recentes sob a alegação de alguns aspectos que buscam justificá-lo, como por exemplo, sua performance, que devido a ser uma biblioteca de baixo nível proporciona uma interação otimizada com o banco, o que se torna muito relevante em um cenário de um ecommerce. A certeza que o desenvolvedor possui de que será executado no banco exatamente o que se escreve, sem ter a ocorrência de queries extras desnecessárias ou não otimizadas que um ORM mal implementado pode facilmente causar, também e uma das justificativas.

Embora existam essas características que apoiem o JDBC, também existem pontos que o desfavorecem, como por exemplo sua sintaxe que em muitos cenários pode exigir a escrita de muito código para tarefas rotineiras e simples, causando dificuldade de manutenção e pouca produtividade.