Anotações - Aula

▼ Aula 01 (14/08/2023) (Apresentação da Disciplina)

- Objetivos Gerais:
  - Identificar soluções de persistências adequadas às necessidades de stakeholders e contexto tecnológico;
  - Desenvolver componentes de software voltados para persistência, usando os principais modelos de dados existentes.
- O termo persistência de dados em si refere-se a **qualquer forma de armazenamento de dados** que for necessária em nosso sistema. O mais usual é utilizar um banco de dados, mas nada impede que sejam utilizadas outras ferramentas, como uma planilha no Excel, um arquivo de texto, etc.
- Listas de exercícios todas as semanas.
- Toda terça-feira terá resolução das listas de atividades.
- A metodologia visa o desenvolvimento de habilidades pessoais e não somente técnicas como, organização, autonomia, trabalho em equipe, auto-disciplina, comunicação e responsabilidade.
- Avaliação de trabalhos práticos em dupla e apresentação (3 trabalhos práticos parciais e 1 trabalho final).
- Adoção do Moodle como sistema principal.
- Módulos:
  - Arquivos em formatos texto e binário.
  - Modelo relacional.
  - Bancos não relacionais, NoSQL e Big Data.
- Maioria dos editores de textos e slides que usamos no dia a dia utiliza XML como formato binário de documentos.
- Avaliação:

- Trabalhos Práticos: 80% da nota
- Listas de exercício individuais: 20% da nota

▼ Aula 01 (15/08/2023) (Continuação da Apresentação e início da Leitura de Arquivos)

- Bibliografia: Designing Data-Intensive Applications, KLEPPMANN, Martin.
- ***Escalabilidade:***
  - Escalabilidade Horizontal
    - A **escalabilidade horizontal** envolve a adição de mais nodes (ou seja, máquinas) à estrutura de um sistema já existente.
    - Vantagens:
      - Maior tolerância a falhas.
      - Redundância de disponibilidade.
      - Custo proporcional ao desempenho.
    - Desvantagens:
      - Complexidade de gerenciamento.
      - Necessidade de redimensionamento dinâmico.
      - Desempenho por máquina pode ser menor.
  - Escalabilidade Vertical
    - A **escalabilidade vertical** é baseada na expansão de uma rede a partir da adição de mais energia e memória à unidade de processamento principal do sistema.
    - Vantagens:
      - Aumento de desempenho individual.
      - Simplicidade de gerenciamento.
      - Compatibilidade com software existente.
    - Desvantagens:
      - Limitações físicas.
      - Pontos únicos de falha.

        - Custo elevado.

- ***Cluster***
    - Um cluster em banco de dados se refere a uma configuração na qual múltiplos servidores ou nós de processamento trabalham juntos para fornecer alta disponibilidade, escalabilidade e desempenho para um sistema de gerenciamento de banco de dados (DBMS). Em outras palavras, um cluster é um conjunto de computadores interconectados que trabalham em conjunto como uma única unidade, compartilhando recursos e tarefas.

- ***Replicação e Particionamento.***
    - A replicação e o particionamento são estratégias comuns usadas em sistemas de gerenciamento de banco de dados (DBMS) para melhorar a disponibilidade, o desempenho e a escalabilidade dos sistemas.
    - **Replicação:**
        - A replicação envolve a criação e a manutenção de cópias idênticas dos dados em vários nós ou servidores. Essas cópias são mantidas sincronizadas para garantir que os dados sejam consistentes em todos os lugares. A replicação é frequentemente usada para melhorar a disponibilidade dos dados, permitindo que os sistemas continuem funcionando mesmo se um nó falhar. Algumas características importantes da replicação incluem:
            - **Redundância:** Ter cópias rdundantes dos dados ajuda a garantir que, se um nó falhar, os dados ainda estejam acessíveis em outros nós.
            - **Leituras Distribuídas:** As consultas de leitura podem ser distribuídas entre as cópias para melhorar o desempenho e reduzir a carga em um único nó.
            - **Recuperação de Falhas:** Se um nó falhar, o sistema pode redirecionar as solicitações para outros nós ativos, minimizando o impacto na operação.
            - **Latência:** A replicação pode introduzir latência na replicação de dados entre os nós, especialmente em sistemas de replicação síncrona.
    - **Particionamento:**

- O particionamento envolve a divisão de uma tabela grande em partes menores, chamadas partições, com base em critérios específicos. Cada partição é tratada como uma entidade separada pelo SGBD (Sistema de Gerenciamento de Banco de Dados), permitindo melhor desempenho e gerenciamento dos dados. O particionamento é frequentemente usado para melhorar o desempenho e a escalabilidade das consultas em bancos de dados grandes. Algumas características importantes do particionamento incluem:
    - **Melhoram o desempenho:** Ao dividir uma tabela grande em partições menores, as consultas podem ser executadas mais rapidamente, pois o DBMS pode processar apenas as partições relevantes.
    - **Gerenciamento Eficiente:** O particionamento facilita o gerenciamento de dados, pois as partições individuais podem ser otimizadas ou mantidas separadamente.
    - **Escalabilidade:** O particionamento permite que um banco de dados cresça horizontalmente, adicionando mais servidores ou nós, à medida que as partições são distribuídas entre eles.
    - **Balanceamento de Carga:** Partições podem ser distribuídas entre os nós do cluster para distribuir a carga de trabalho de maneira uniforme.
    - **Critérios de Particionamento:** As partições podem ser baseadas em valores de coluna (como intervalos de datas ou valores numéricos) ou em funções de hash.

- ***Transações***
    - Uma transação em um banco de dados é uma sequência de uma ou mais operações que são tratadas como uma unidade indivisível de trabalho. O conceito de transação é fundamental para garantir a consistência, a integridade e a confiabilidade dos dados em um sistema de gerenciamento de banco de dados (DBMS). As propriedades ACID são frequentemente associadas a transações para garantir esses aspectos:
        - **Atomicidade (Atomicity)**: A propriedade atomicidade garante que uma transação seja tratada como uma operação única e indivisível.

Isso significa que todas as operações dentro de uma transação são executadas com sucesso ou todas são desfeitas em caso de erro. Não pode haver um estado intermediário onde algumas operações foram aplicadas e outras não.

- **Consistência (Consistency):** A propriedade de consistência garante que uma transação leva o banco de dados de um estado válido para outro estado válido. Isso implica que as transações não podem violar as restrições de integridade definidas no banco de dados.
- **Isolamento (Isolation):** A propriedade de isolamento garante que as operações de uma transação sejam isoladas de outras transações sejam isoladas de outras transações concorrentes. Isso significa que o resultado de uma transação não é visível para outras transações até que a transação seja concluída. O isolamento evita problemas como leituras sujas, leituras não repetíveis, escritas fantasmas e anomalias de atualização.
- **Durabilidade (Durability):** A propriedade de durabilidade garante que as mudanças feitas por uma transação sejam permanentes, mesmo em caso de falha do sistema. Uma vez que uma transação é confirmada (commit), suas alterações são armazenadas de forma segura e não podem ser desfeitas.

- ***Unicode***
  - Unicode é um padrão de codificação de caracteres que visa abranger praticamente todos os caracteres de todos os sistemas de escrita conhecidos, além de símbolos, pontuações e outros elementos gráficos utilizados em textos em todo o mundo. Ele fornece uma maneira de representar e processar caracteres de diferentes idiomas, scripts e símbolos em computadores e sistemas de software.
  - UTF-8, UTF-16 e UTF-32.
- ***ASCII***
  - ASCII (American Standard Code for Information Interchange) é um conjunto de codificações de caracteres que foi amplamente utilizado nas primeiras décadas da computação para representar caracteres em dispositivos eletrônicos e sistemas de comunicação. Ele é um dos conjuntos de caracteres mais simples e básicos, composto por 128

caracteres, incluindo letras maiúsculas e minúsculas, números, símbolos de pontuação e caracteres de controle.

- ***ISO-8859-1(Latin 1)***
  - ISO-8859-1, também conhecido como Latin 1, é uma codificação de caracteres que faz parte da família ISO/IEC 8859 de padrões de codificação de caracteres. Essa codificação foi desenvolvida para representar uma série de idiomas europeus ocidentais, incluindo inglês, francês, alemão, espanhol, italiano e muitos outros. Ela foi uma evolução do ASCII, mas com a adição de caracteres acentuados e especiais frequentemente usados em línguas além do inglês.
- ***Fluxos de entrada e saída***
  - Em programação, especialmente em linguagens orientadas a objetos, os conceitos de fluxo de entrada (input stream) e fluxo de saída (output stream) são usados para lidar com a entrada e saída de dados de maneira organizada e eficiente. Esses conceitos são frequentemente usados para manipular a entrada e saída de arquivos, redes, dispositivos de E/S (entrada/saída) e muito mais.
  - Fluxo de entrada (Input Stream):
    - Sequência de dados lida pelo programa.
    - Proveniente de um arquivo, teclado, conexão de rede ou qualquer outra fonte de dados.
    - Permite a leitura de dados de forma sequencial pelo programa.
  - Fluxo de saída (Output Stream):
    - Sequência de dados escrita pelo programa.
    - Gravados em arquivos, enviados por rede ou exibidos na tela.
    - ex: Imprimir mensagem na tela com print('') em Python.
- Exemplo de leitura de arquivo externo em Java:
  - O código abaixo é uma tentativa de ler o conteúdo de um arquivo de texto. (O código abaixo não funciona, está presente apenas para explicação do algoritmo de leitura).
  - `InputStream` é uma classe abstrata que recebe o nome do arquivo a ser aberto como parâmetro.

- `BufferedReader` é uma classe padrão Java utilizada para realizar a leitura eficiente de caracteres a partir de um fluxo de entrada.
- a função `readLine()` atribui o primeiro caractere da linha para a variável `s`, que ao final é impressa na tela.

```
public class TestaEntrada {
  public static void main(String args[]){
    InputStream is = new InputStream("arquivo.txt");
    BufferedReader br = new BufferedReader(is);
    String s = br.readLine();
    System.out.println(s);
  }
}
```

- Sobre o envio de atividades em código:
  - Exemplo de envio no Moodle: q1.java, q2.java, etc.
  - Zipar o código e enviar tudo de uma vez.