Implementação
Armazenamento em Disco

André Luis Schwerz
Rafael Liberato Roberto

Descrição

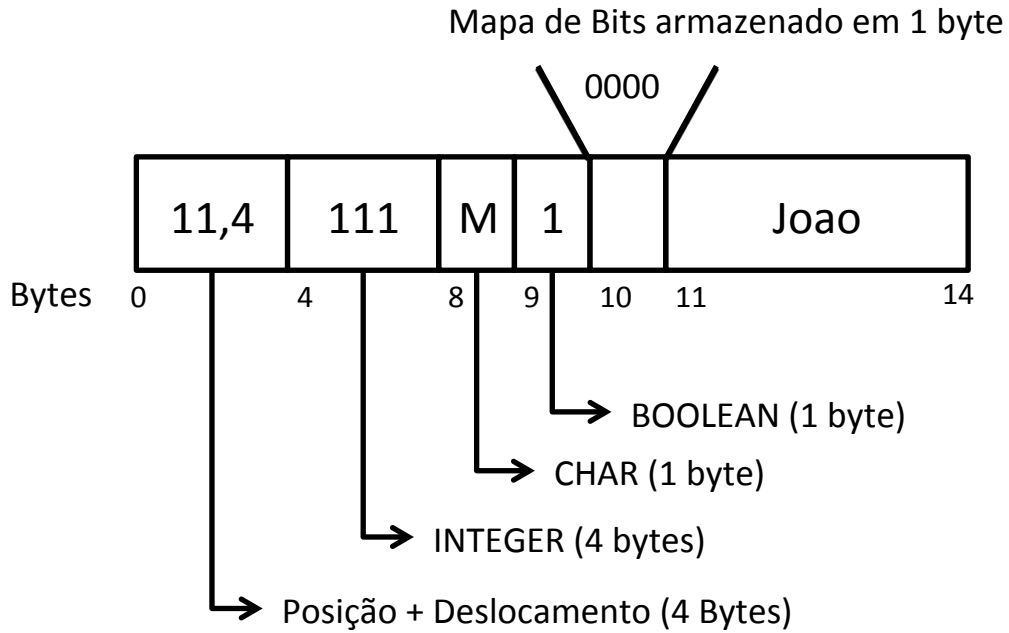
- Desenvolver um programa que **manipule registros** em **arquivos binários** com registros de tamanho variáveis.

- Menu de funcionalidades:
  1 – Criar arquivo
  2 – Inserir registros
  3 – Listar registros
  4 – Excluir registros

- Tipos de dados suportados:
  - **INTEGER (4 bytes)**
  - **BOOLEAN**
  - **VARCHAR(n)** onde n é o tamanho máximo da cadeia de caracteres.
  - **CHAR(n)** onde n é o tamanho da cadeia de caracteres.

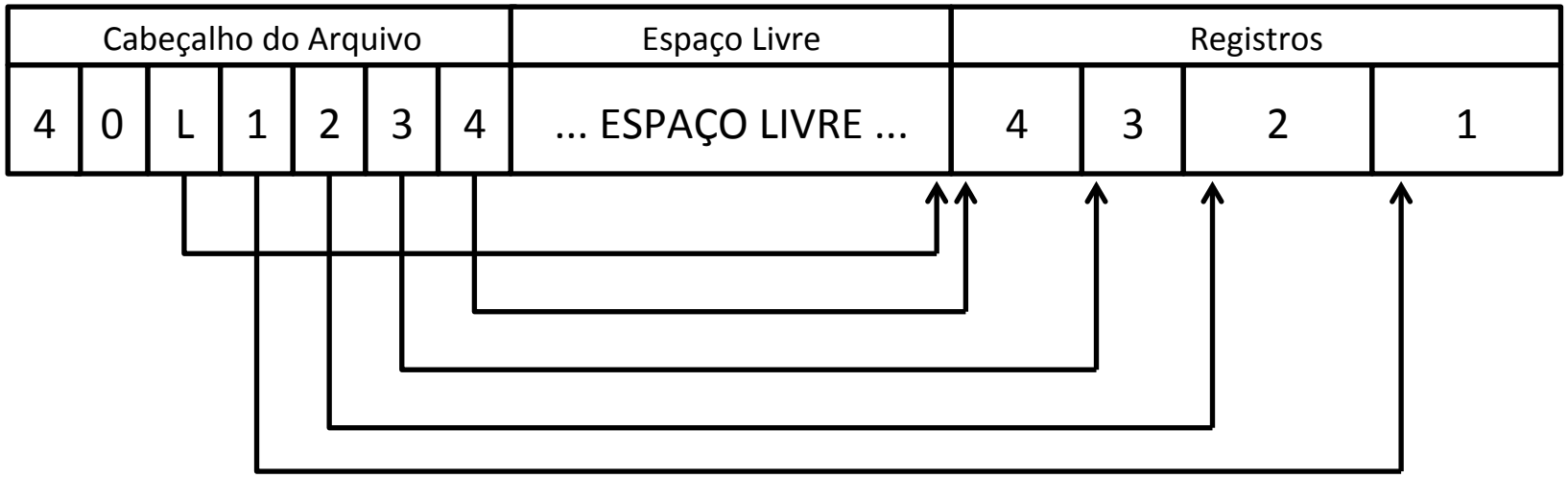
Descrição

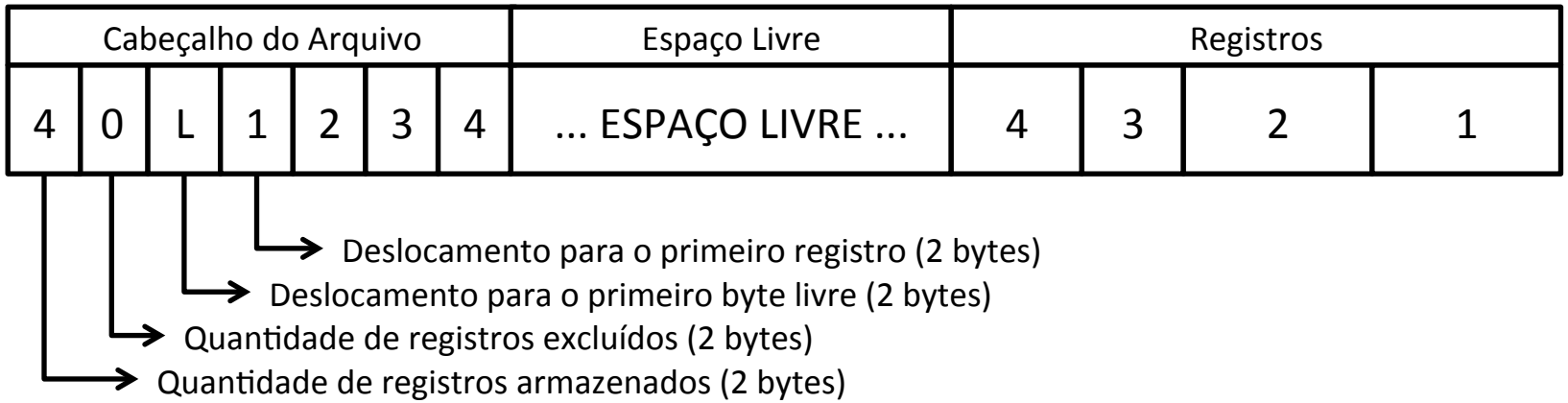
- Na representação de um registro considerando os atributos `ID` (`INTEGER`), `sex` (CHAR), `status` (BOOLEAN) como atributo fixo e `name` (VARCHAR) como atributo variável, temos:



Descrição

- Representação um arquivo considerando registros de tamanho variável.





Funcionalidades

1 – CRIAR ARQUIVO

- Após o usuário selecionar a opção CRIAR ARQUIVO, o programa deve solicitar ao usuário um arquivo com extensão (.sql) que contenha 1 ou mais sentenças com SQL CREATE.
- O usuário pode selecionar a opção CRIAR ARQUIVO inúmeras vezes.
- O programa pode assumir que não haverá erros de sintaxe em cada sentença, ie., o programa não precisa verificar se as sentenças seguem a definição.
- Para cada SQL CREATE, o programa deve criar um arquivo de **2KB** de acordo com as estrutura apresentada em sala de aula.
  - DICA: Inicialize um arquivo ocupando os 2KB com "lixo".
- Além disso, você pode criar um arquivo auxiliar para armazenar os meta-dados da tabela.

Funcionalidades

1 – CRIAR ARQUIVO

– Arquivo texto (.sql) contendo o formato do arquivo a ser criado seguindo a seguinte sintaxe:

Sintaxe

```
{<sentence>}

<sentence>:=
        CREATE TABLE table_name (
                {column_name <data_type>}
        );

<data_type> := INTEGER |
               CHAR([1,2,...n]) |
               VARCHAR([1..n]) |
               BOOLEAN
```

Funcionalidades

1 – CRIAR ARQUIVO

- Exemplo:

```
CREATE TABLE employee (
   id INTEGER,
   name VARCHAR(255),
   sex CHAR(1)
   status BOOLEAN
);

CREATE TABLE department(
   id INTEGER,
   name VARCHAR(255)
)
```

Funcionalidades

2 – INSERIR REGISTROS

- Ao selecionar a opção INSERIR REGISTROS, o programa deve solicitar ao usuário um arquivo texto com extensão (.sql) que contenha 1 ou mais sentenças com SQL INSERT.
- O programa pode assumir que não haverá erros de sintaxe em cada sentença, ie., o programa não precisa verificar se as sentenças seguem a definição.
- Cada sentença INSERT SQL contem dados de pelo menos um atributo definido na sentença SQL CREATE. Os campos não informados devem ser considerados NULOS.
- O programa deve armazenar os registros de cada instrução SQL INSERT no arquivo binário especificado seguindo o algoritmo explicado em sala de aula.
- O programa deve informar na saída-padrão quantas sentenças foram executadas com sucesso.
- Caso o arquivo esteja cheio, uma mensagem deve informar o usuário na saída-padrão.
    - É opcional criar um novo arquivo, quando o primeiro arquivo ficar cheio.

Funcionalidades

2 – INSERIR REGISTROS

- Exemplo:

```
INSERT INTO employee (id, name, sex, status)
       values(2, 'Jose da Silva', 'M', true);
INSERT INTO employee (id, name, sex, status)
       values(1, 'Maria Ferreira', 'F', false);
INSERT INTO employee (id, name, sex)
       values(4, 'Joaquim Barbosa', 'M');
INSERT INTO employee (id, nome)
       values(5, 'João Souza');
INSERT INTO employee (id, status)
       values(6,false);
```

Funcionalidades

3 – LISTAR REGISTROS

- Após escolher a opção **Listar Registros**, o programa solicita ao usuário a informação de qual arquivo deverá ser listado.
- Os registros podem ser listados na saída padrão.
- Não é necessário fazer qualquer filtro nessa listagem.

Funcionalidades

4 – EXCLUIR REGISTROS

- Após escolher a opção **Excluir Registros**, o programa deve solicitar ao usuário um arquivo texto com extensão (.sql) que contenha 1 ou mais sentenças com SQL DELETE.
- O programa pode assumir que não haverá erros de sintaxe em cada sentença, ie., o programa não precisa verificar se as sentenças seguem a definição.
- Cada sentença SQL DELETE informará **apenas um único campo de busca** que resultará na exclusão de zero, um ou vários registros.
- Poderão ser utilizados os seguintes operadores de comparação:
  - =, !=, > e <.
- É necessário informar na saída-padrão quanto registros foram excluídos considerando todas as sentenças descritas no arquivo .sql
- O programa deverá eliminar o registro no arquivo binário especificado deixando-o fragmentado.
  - A implementação de qualquer técnica de desfragmentação é opcional.

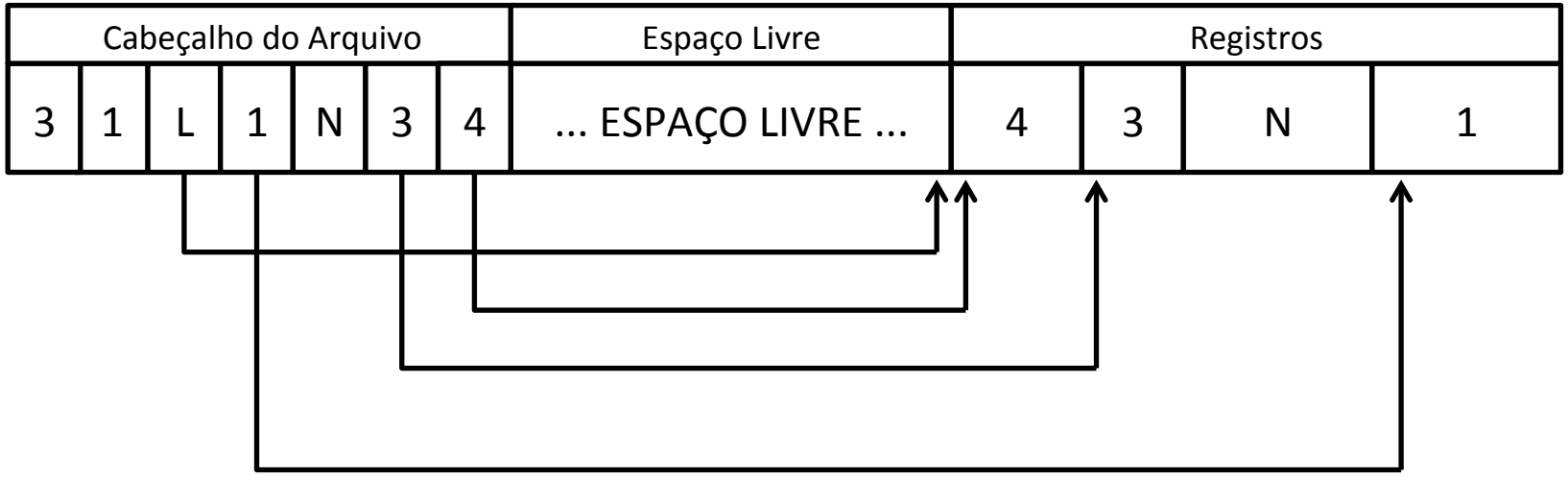
Funcionalidades

4 – EXCLUIR REGISTROS

- Exemplo:

```
DELETE FROM employee WHERE id = 2;
DELETE FROM employee WHERE sex = 'M';
DELETE FROM employee WHERE name = 'Joao da Silva'
```

- Estrutura do arquivo caso o registro 2 seja excluído.



Informações Gerais

- Deadline:
  - 13 de junho de 2016
  - Via Moodle.

- Grupos:
  - Duplas.

- Você pode utilizar na implementação as linguagens C ou JAVA, de acordo com a sua preferência.

- Além do código-fonte comentado, você deve descrever em um arquivo texto um passo-a-passo para execução do seu trabalho e todas as funcionalidades implementadas.

- Critérios de avaliação
  - Qualidade da solução proposta [4.0]
  - Efetividade nos testes de execução. [4.0]
  - Qualidade da informação descrita nos comentários e no readme.txt. [2.0]
- A cada dia de atraso na entrega, o trabalho será penalizado em 1.0 ponto.