README

Trabalho de Banco de Dados 2
Por Raphael Machado dos Santos e Eduardo Mendes Senger

01/06/2014 UPDATE: BM with data dictionary by CACMATTHI (Caciano – Mateus - Thiago)
Alterações:
buffer.h
– Estruturas e DEFINES que estavam no *buffer.c* foram levados para o *buffer.h.*
– Função *showbuffer* foi alterada para INT.
– Função *fillBuffer*: os parâmetros *fdata* e *fmeta* foram substituídos por *nomeTabela* e arquivo (Explicação de cada parâmetro abaixo).
– DEFINES de erros acrescentados.
buffer.c
– Funcao *fillBuffer()* - Nova estratégia: uso de dicionário de dados.
Funcao s*howbuffer()* - Modo de exibição dos dados.
main.c
– Verificação de erro.
Nova estratégia:
Nessa atualização, a aplicação deve receber como parâmetro (na hora da execução) um arquivo chamado
dicionario.dat, esse arquivo contem nomes de tabela e seus caminhos.
EXEMPLO:
fs_personagemfs_personagem.datfs_statusfs_status.dat

O dicionário de dados e passado como parâmetro da função *fillBuffer,* junto com o nome da tabela que a aplicação quer inserir no buffer.

Na função *fillBuffer,* o arquivo de dicionário de dados e aberto e é feita a leitura dele, onde a função compara o nome da tabela passada como parâmetro com o nome da tabela que está no arquivo. Se a comparação retornar 0, ou seja, foi encontrada a tabela, a função coleta o caminho da tabela e faz a leitura do arquivo da mesma.

Na leitura do arquivo da tabela, primeiramente e coletado o meta daquela tabela, onde são lidos a quantidade de campos. E é feita a leitura do nome, tipo e tamanho do(s) campo(s) n vezes, onde N e a quantidade de campos.
Apos a leitura do meta, no cabeçalho do arquivo, a leitura continua e coleta os dados.

Caso ocorrer algum erro que for sabido pela aplicação, ela fornece uma mensagem com o nome do erro.

O modo que os dados são apresentados também foi alterado, agora ha delimitadores dentro da tupla, podendo assim, mostrar de forma mais clara e limpa cada informação.

COMO EXECUTAR:

No modo que se encontra, se compilado e executado a aplicação ira ler os dados do arquivo *fs_personagem*. Porem há outro arquivo que pode ser lido, para escolher qual arquivo deve lido para preencher o *buffer* basta ir na *main.c* → função *fillbuffer* e trocar o penúltimo parâmetro, no momento possui *fs_personagem.* Mas o usuário pode trocá-lo por *fs_status.*
Apos executar aparecera um menu com 3 opções:
0 – Preencher o buffer
1 – Listar buffer
2 – Sair
Primeiramente o usuário deve Inserir os dados no buffer e em seguida ir na opção 1 e listar os dados.

Este trabalho utiliza a licença GNU General Public License.

---

void initBuffer(buffer *bPool,int lenght,field *fieldList, int fieldCount);

Descrição:
Esta função inicializa o buffer colocando os parâmetros standard em todas variáveis que serão utilizadas.

buffer *bPool Qual Buffer Pool deve ser iniciado.
int lenght Tamanho do Buffer.
field *fieldList Onde os campos vão ser armazenados.
int fieldCount Contagem de quantos campos terão neste BufferPool.
RETORNO Nenhum.

---

void findNextAvaliable(buffer *bPool);

Descrição:
Esta função procura a próxima página disponível.

buffer *bPool Buffer Pool a ser trabalhado
RETORNO NENHUM

---

void applyReplacementPolicies(buffer *bPool);

Descrição:
Esta função aplica as políticas de substituição, a política utilizada tende a remover a pagina do buffer que contem o menor pinCount.

buffer *bPool Buffer Pool a ser trabalhado
RETORNO NENHUM

---

void bufferInsert(buffer *bPool,char *tuple, int diskSeek, int tupleLenght);

Descrição:
Esta função insere um novo elemento no buffer.

buffer *bPool Buffer Pool a ser inserido
char *tuple String da tupla a ser inserida no Buffer Pool
int diskSeek Posicao no disco onde essa pagina se encontra
int tupleLenght Tamanho maximo da tupla a ser inserida
RETORNO NENHUM

---

int fillBuffer(buffer *bufferPool, field *fieldList,char *nomeTabela, char *arquivo);

Descrição:
Esta função que vai ler o arquivo de dicionário de dados e em seguida ler o arquivo que o usuário deseja e preencher o Buffer Pool com eles.

buffer *bufferPool Buffer Pool onde serão armazenados os dados.

field fieldList Estrutura que contem os campos que foram lidos do arquivo de metadados.
*nomeTabela Nome da tabela que o usuário quer inserir no buffer.
*arquivo Dicionário de dados que e passado como parâmetro na execução.
ERRO_ARQUIVO Retorno caso der erro na leitura do arquivo ou arquivo inexistente.
TABELA_NOTFOUND Retorno caso tabela não for encontrada ou inexistente.
BUFFER_PREENCHIDO Retorno caso tudo ocorrer com sucesso.

---

int showBuffer(buffer *bufferPool);

Descrição:
Imprime na tela o conteúdo de todas as paginas do BufferPool.

buffer *bufferPool Buffer Pool a ser imprimido na tela
BUFFER_VAZIO Retorno quando ocorre erro no buffer.
