BCC – PI5
Professor: Heitor Murilo Gomes

Entrega

O que: Código fonte + modelo da base de dados + testes (scripts de bancos de dados usados para teste)
Quando: 5 de Junho de 2013 até as 21:30hs (defesa das 18:00 até as 21:30)
Onde: Lab 06
Como: Via pendrive ou para o e-mail heitor_murilo_gomes@yaho.com.br.
Defesa: O trabalho deve ser apresentado no dia da entrega. O aluno será indagado sobre questões pertinentes a sua solução.
Valor: até 10,0 pontos no semestre.
Grupo: grupos de até 2 alunos (pode ser individual).

Projeto Interdisciplinar 5

Introdução
O projeto é divido em duas partes. A primeira parte contempla o **desenvolvimento** de uma interface para a manipulação de um SGBD. A segunda parte exige a codificação de uma função que **estende** o SGBD, isto é, adiciona uma nova funcionalidade ao SGBD.
Todas as partes do projeto devem ser entregues e defendidas no mesmo dia (especificado acima).

Linguagem de programação: C++
SGBD: SQLite

Parte I.

Desenvolver o código em C++ que permite ao usuário realizar tarefas administrativas em um ou mais bancos de dados. Para manipular mais de um banco de dados por vez use o comando **ATTACH** no SQLite (veja anexo B).

O seu código deve ter alguma interface com o usuário. Nas especificações (a, b e c) são apresentados exemplos de interface, sendo um deles baseado em parâmetros e o segundo em opções de um menu. Você tem liberdade para escolher a interface com o programa, porém ela deve contemplar as funcionalidades exigidas.

a) Desenvolver o código referente as funções de **adicionar (insert)**, **remover (delete)**, **modificar (update)** elementos de uma tabela. O(s) banco(s) de dados devem ser parametrizados para o programa. Para testar, você pode utilizar a tabela "planets" e o banco de dados "banco_exemplo.sqlite", porém a sua solução não deve estar atrelada a um banco específico e/ou a uma tabela específica. Um comando pode envolver mais de um banco de dados, por exemplo, os dados a serem inseridos em uma tabela podem ser provenientes de uma tabela de outro banco de dados (**bulk insert**).

Exemplo 1 (parâmetros):

Adiciona um registro
programa.exe -b "banco_exemplo.sqlite" -c "INSERT INTO planets VALUES (7, 'Uranus', 'pl', 0, 1, 0, 0, 1, 1, 0, 0, 1)"

Remove o registro de object_id = 7
programa.exe -b "banco_exemplo.sqlite" -c "DELETE FROM planets WHERE object_id = 7"

Exemplo 2 (menu):

Menu inicial
"Qual banco?"
banco_exemplo.sqlite
"Qual o comando?"
(a) adicionar, (r) remover, (m) modificar, (v) visualizar."
a
"Qual tabela?"
planets
"Dado para o campo object_id: "
7
"Dado para o campo object: "
Uranus
...

b) A **visualização (select)** deve receber como entrada uma consulta e apresentar o resultado desta de forma paginada. Utilize **LIMIT** e **OFFSET** para navegar entre as páginas. Deve ser possível especificar a quantidade de resultados por página e navegar entre as páginas, por exemplo, o botão → avança a página, o ← retrocede e o **Q** finaliza.

Exemplo 3 (parâmetros – consulta no mesmo banco):

Consulta entre tabelas do mesmo banco
programa.exe -b "banco_exemplo.sqlite" -c "SELECT object_id, object FROM planets" -p 3
Tal que **-p** a quantidade de elementos por página.

*(veja o **anexo A** para um exemplo do formato da saída desse exemplo)*

Exemplo 4 (parâmetros – consulta entre bancos diferentes):

Consulta entre tabelas do mesmo banco
programa.exe -b "bd1.sqlite" –b2 "bd2.sqlite" -c "SELECT A.nome, B.nome FROM main.funcionario A, database2.cliente B WHERE a.cpf = b.cpf" -p 3
Tal que -b indica o arquivo do primeiro banco e –b2 o arquivo do segundo banco.

Parte II.

Desenvolver uma função de *approximate string matching* (“comparação” aproximada de strings) para o SQLite. Uma função de *approximate string matching* é o nome dado para as técnicas usadas para encontrar strings que são similares a um padrão (outra string). Existem diversos algoritmos quem implementam essa funcionalidade, você deverá implementar dois algoritmos de similaridade (n-gram e *edit distance*) e mais um auxiliar de pré-processamento. As funções criadas, com exceção da função de pré-processamento, devem ser acessíveis a partir do SQLite. Para isso use a função **sqlite3_create_function(...)** (veja anexo B).

Associe quatro variações das funções desenvolvidas ao banco de dados. Sendo duas que invocam diretamente os algoritmos n-gram e edit distance e outras duas que fazem o mesmo porém executam o pré-processamento da string antes de executar os algoritmos.

a) Pré-processamento de uma string. Uma função em C/C++ que recebe uma string remove caracteres especiais, transforma todos os caracteres em seus correspondentes minúsculos e ordena lexicograficamente as suas substrings (separadas por espaço ‘ ’). Por exemplo: a string “Javanir, Clonoviskis.” após o processamento irá remover os caracteres especiais ‘,’ e ‘.’, transformar ‘J’ para ‘j’ e ‘C’ para ‘c’ e por último inverter as ordem da primeira com a segunda substring. Resultado: “clonoviskis javanir”.

b) Algoritmo n-gram. Os parâmetros para a função devem ser S_1 (primeira string), S_2 (segunda string) e N (tamanho das substrings). O retorno da função é dado pela fórmula abaixo:

$$Conf(S_1, S_2) = \frac{2 * hits}{total}$$

Tal que “hits” representa a quantidade de substrings de tamanho N iguais e “total” a soma do número de substrings de tamanho N obtidas a partir de S_1 e da S_2. O emprego dessa fórmula torna a função reflexiva, isto é, $S_1 \approx S_2 \leftrightarrow S_2 \approx S_1$.
O retorno da função pode ser interpretado como o grau de confiança da similaridade entre S_1 e S_2.

c) Algoritmo *edit distance* (distância de levenshtein). Os parâmetros para a função devem ser S_1 (primeira string) e S_2 (segunda string). O retorno da função é o número mínimo de edições necessárias para que $S_1 = S_2$.

Anexo A – Resultado da consulta do exemplo 3

****
Pagina: 1 de 3
Query: SELECT object_id, object FROM planets

object_id	*object*
1	*Mercury*
2	*Venus*
3	*Earth*

****

****
Pagina: 2 de 3
Query: SELECT object_id, object FROM planets

object_id	*object*
4	*Mars*
5	*Jupiter*
6	*Saturn*

****

****
Pagina: 3 de 3
Query: SELECT object_id, object FROM planets

object_id	*object*
7	*Uranus*
8	*Neptune*
9	*Pluto*

****

Anexo B – Links

Introdução a interface de programação em C/C++ do sqlite3:
http://www.sqlite.org/cintro.html

lista de funções do sqlite3 (Em C): http://www.sqlite.org/c3ref/funclist.html

sqlite3_create_function(): http://www.sqlite.org/c3ref/create_function.html

Comando ATTACH DATABASE: http://www.sqlite.org/lang_attach.html

Arquitetura interna do sqlite3 (avançado): http://www.sqlite.org/arch.html

Funções do sqlite (SQL): http://www.sqlite.org/lang_corefunc.html