Trabalho de Implementação II

Banco de Dados – 2016.1 Prof. Luiz André P. Paes Leme

Objetivo

Neste trabalho, cada grupo de até quatro alunos deverá implementar, em C ou C++, parte da funcionalidade do protótipo de SGBD, chamado BDUFF, especificado a seguir.

Protótipo de SGBD BDUFF

BDUFF é um o protótipo bastante simplificado de um sistema gerenciador de bancos de dados. Nesse sistema, um banco de dados é composto por um conjunto de relações e os comandos de definição e manipulação de dados são comandos SQL simplificados. A funcionalidade do BDUFF está especificada nas Seções 1 a 8 deste documento.

O enunciado deste trabalho encontra-se na Seção 9. Basicamente, deverão ser implementadas as operações da álgebra relacional especificadas na Seção 7

1. Relações

Cada relação é armazenada em dois arquivos do tipo texto (ascii). No primeiro arquivo, com extensão .dad, ficam armazenadas as tuplas da relação. No segundo, com extensão .ctl, estão armazenadas as informações de catálogo.

O formato do arquivo .ctl é o seguinte:

N,M
A_1,TIPO[,nn][,ord][,chv][,fk:T:restr|casc]
A_2,TIPO[,nn][,ord][,chv][,fk:T:restr|casc]
...
A_N,TIPO[,nn][ord][chv][,fk:T:restr|casc]
$T_1,T_2,\ldots,T_m$

Onde:
- N representa o grau da relação e M, a cardinalidade da relação,
- Ai é o nome do i-ésimo atributo, que deve estar em maiúsculas,
- TIPO indica o tipo do atributo: C ou I (cadeia de caracteres ou inteiro),
- nn representa que o atributo não admite o valor nulo,
- ord indica que a relação está ordenada crescentemente pelo atributo em questão (ord só pode aparecer associado a um atributo),

- chv indica que o atributo em questão é a chave primária da relação
  (chv só pode aparecer associado a um atributo),
- se chv ou ord for especificado, então nn também deve ser,
- fk indica que o atributo em questão é uma chave estrangeira da relação que referencia a
  tabela de nome T (uma tabela pode ter várias chaves estrangeiras),
- uma das opções restr ou casc deve estar especificada e indica que o comportamento da
  chave estrangeira em caso de remoção e atualização (“restrict” ou “cascade”),
- não há “brancos” antes ou depois das vírgulas,
- $T_1,T_2,\ldots,T_m$ são os nomes das tabelas que referenciam a relação.

O formato do arquivo .dad é o seguinte:

$V_{11},V_{12},...,V_{1N}$
$V_{21},V_{22},...,V_{2N}$
...
$V_{m1},V_{m2},...,V_{mN}$

Onde:
- Vij representa o valor do j-ésimo atributo da i-ésima tupla,
- m indica a cardinalidade da relação. Vale observar que a relação pode ser vazia e,
  neste caso, m=0,
- não há “brancos” antes ou depois das vírgulas,
- Se Vij for uma cadeia de caracteres, deverá ser armazenada entre aspas simples (’),
- Se Vij for o valor nulo, deverá ser representado pela palavra NULO, sem aspas.

Exemplo do conteúdo dos arquivos TAB.ctl e TAB.dad que armazenam a relação TAB composta por três atributos e quatro tuplas (TAB está ordenada crescentemente pelo atributo CYZ, o atributo A é uma chave de TAB, o atributo BX é uma chave estrangeira que referencia a tabela TTT com opção “cascade” e as tabelas TA e TB referenciam a tabela TAB):

- Arquivo TAB.ctl:

```
4,4
A,I,nn,chv
BX,I,fk:TTT:casc
CYZ,C,nn,ord
XZ,C
TA,TB
```

- Arquivo TAB.dad:

```
50,33,'Banco de Dados',"
777,NULO,'Cálculo I',' '
2,43,'Compiladores',NULO
51,1,'Estrutura de Dados','NULO'
```

Vale observar que a segunda tupla tem o valor nulo no atributo BX e a única tupla que tem o valor nulo no atributo XZ é a terceira.

2. Definição de Dados

As relações da base de dados são criadas através de uma versão simplificada do comando SQL CREATE TABLE. Este comando deverá ser escrito em arquivo texto (ascii) com extensão .sql.

O comando CREATE TABLE possui a seguinte sintaxe:

CREATE TABLE <nome-tabela> (
<nome-atrib-1> <tipo-atrib-1> [NN] [KEY] [ORD] [FK <tab-ref> [CASC]],
<nome-atrib-2> <tipo-atrib-2> [NN] [KEY] [ORD] [FK <tab-ref> [CASC]],
...
<nome-atrib-n> <tipo-atrib-n> [NN] [KEY] [ORD] [FK <tab-ref> [CASC]])
Onde:

- <nome-tabela> é o nome da relação,
- <nome-atrib-i>, $1 \leq i \leq n$ é o nome do i-ésimo atributo da relação,
- <tipo-atrib-i>, $1 \leq i \leq n$ é o tipo do i-ésimo atributo, podendo ser STRING ou INTEGER (cadeia de caracteres ou inteiro, respectivamente),
- NN, cláusula opcional, indica que o respectivo atributo não admite o valor nulo,
- KEY, cláusula opcional, indica que o respectivo atributo forma a chave primária da relação e só pode estar especificado em um atributo,
- ORD, cláusula opcional, indica que a relação estará fisicamente ordenado pelos valores deste atributo e também só pode estar especificado em um atributo,
- NN deve ser especificado se KEY ou ORD for,
- FK, cláusula opcional, indica que o respectivo atributo forma uma chave estrangeira que referencia a tabela <tab-ref> (esta tabela tem que existeir e tem que possuir chave primária); o atributo que forma a chave estrangeira e o que forma a chave primária da tabela referenciada devem ter o memso tipo,
- CASC, cláusula opcional da definição de chave estrangeira, indica que a chave estrangeira em questão está definida com a opção “cascade” para remoção e atualização, se CASC não for especificada, a opção para remoção e atualização é RESTRICT (default).

Ao ser executado, este comando cria os arquivos <nome-tabela>.dad e <nome-tabela>.ctl, que representam a relação definida. O arquivo .dad deverá estar vazio e o .ctl deverá conter as informações especificadas no comando, seguindo as definições da seção anterior.

3. Inserção de Dados

A inserção de dados em uma relação é realizada através de uma versão simplificada do comando SQL INSERT. Este comando deverá ser escrito em arquivo texto (ascii) com extensão .sql.

O comando INSERT possui a seguinte sintaxe:

INSERT
INTO <nome-tabela>
VALUES (<constante-1>,<constante-2>,...,<constante-n>);

Onde:
- <constante-i> é um inteiro, uma cadeia de caracteres (entre aspas duplas) ou a palavra NULO (sem aspas),
- a relação <nome-tabela> deve ter grau n,
- a i-ésima constante deve ter o mesmo tipo do i-ésimo atributo da relação <nome-tabela>,
- se a relação possui chave primária, a restrição de chave deve ser preservada,
- se <constante-i> for NULO então o respectivo atributo deve admitir valores nulos.

ATENÇÂO:

- se a cláusula ORD foi especificada na criação da respectiva relação, então o critério de ordenação deverá ser respeitado,
- se a relação possui chaves estrangeiras, a integridade referencial deverá ser respeitada.

4. Remoção de Dados

A remoção de dados em uma relação será realizada através de uma versão simplificada do comando SQL DELETE. Este comando deverá ser escrito em arquivo texto (ascii) com extensão .sql.

O comando DELETE terá a seguinte sintaxe:

DELETE
FROM <nome-tabela>
[WHERE <condição-sel>];

Onde:
- <nome-tabela> é o nome de uma tabela,
- <condição-sel> é uma condição de seleção da forma T=v, onde:
    - T é o nome de um atributo de <nome-tabela>,
    - v é um valor do tipo do atributo T
        (se v for uma cadeia de caracteres deverá vir entre aspas duplas).

ATENÇÂO:

- se a tabela em questão é referenciada por uma chave estrangeira, a integridade referencial deverá ser respeitada, considerando a opção “restrict” ou “cascade” especificada. Caso haja mais de uma chave estrangeira referenciando a tabela, verificar primeiro as que são especificadas com a opção “restrict” para evitar propagar uma remoção que posteriormente terá que ser bloqueada.

5. Alteração de Dados

A alteração de dados em uma relação será realizada através de uma versão simplificada do comando SQL UPDATE. Este comando deverá ser escrito em arquivo texto (ascii) com extensão .sql.

O comando UPDATE terá a seguinte sintaxe:

UPDATE <nome-tabela>
SET <atribuição>
[WHERE <condição-sel>];

Onde:
- <nome-tabela> é o nome de uma tabela,
- <atribuição> é uma atribuição da forma T=v, onde:
    - T é o nome de um atributo de <nome-tabela>,
    - v é um valor do tipo do atributo T ou a palavra NULO
        (se v for uma cadeia de caracteres deverá vir entre aspas duplas),
    - se T for chave primária, a restrição da chave deve ser preservada,
    - se T não admitir nulos, essa restrição deve ser preservada.

ATENÇÂO:

- se T for chave primária e a tabela em questão for referenciada por uma chave estrangeira, a integridade referencial deverá ser respeitada, considerando a opção "restrict" ou "cascade" especificada. Caso haja mais de uma chave estrangeira referenciando a tabela, verificar primeiro as que são especificadas com a opção "restrict" para evitar propagar uma remoção que posteriormente terá que ser bloqueada,
- se a cláusula ORD foi especificada na definição de T, então o critério de ordenação deverá ser respeitado,
- se T for uma chave estrangeira, a integridade referencial deverá ser respeitada**.**

- <condição-sel> é uma condição de seleção da forma T=v, onde:
    - T é o nome de um atributo de <nome-tabela>,
    - v é um valor do tipo do atributo T
        (se v for uma cadeia de caracteres deverá vir entre aspas duplas),

6. Consulta aos Dados

Uma consulta realizada no BDUFF será especificada através de uma versão simplificada do comando SQL SELECT. Neste caso, para que seja executado, este comando deverá ser transformado em uma seqüência de operações algébricas equivalentes. Uma consulta SQL deverá ser escrito em arquivo texto (ascii) com extensão .sql.

O comando SELECT SQL poderá ter as seguintes formas:

A) SELECT [DISTINCT] <lista-atributos>
FROM <relação>
[WHERE <condição-sel>]
[ORDER BY <lista-atributos-ordem>];

Onde:
- <relação> é o nome de uma relação,
- <lista-atributos> é uma lista, separada por vírgulas, de nomes de atributos de <relação> (não há "brancos" antes ou depois das vírgulas), os nomes dos atributos podem estar qualificados com o nome da relação à qual pertencem,

- <condição-sel> é uma condição de seleção de uma das seguintes formas: T=v, T<v, T<=v, T>v, T>=v ou T<>v, onde:
  - T é o nome de um atributo de <relação>,
  - v é um valor do tipo do atributo T
    (se v for uma cadeia de caracteres deverá vir entre aspas duplas),

- <lista-atributos-ordem> é uma lista, separada por vírgulas, de nomes de atributos de <relação> que aparecem em <lista-atributos>, que define o critério de ordenação da relação resultante (não há "brancos" antes ou depois das vírgulas), os nomes dos atributos podem estar qualificados com o nome da relação à qual pertencem.

B) SELECT [DISTINCT] <lista-atributos>
FROM (<relaçãoA> JOIN <relaçãoB> ON <condição-junção>)
[WHERE <condição-sel>]
[ORDER BY <lista-atributos-ordem>];

Onde:
- <relaçãoA> e <relaçãoB> são nomes de relações distintas,
- <lista-atributos> é uma lista, separada por vírgulas, de nomes de atributos pertencentes, individualmente, a <relaçãoA> ou a <relaçãoB> (não há "brancos" antes ou depois das vírgulas); os nomes dos atributos podem estar qualificados com os nomes das relações às quais pertencem; se um atributo da lista ocorrer com o mesmo nome nas duas relações, então deverá estar qualificado com o nome de uma das relações.
- <condição-junção> é uma condição de junção da forma T=U, onde:
  - T é o nome de um atributo de <relaçãoA> e U é o nome de um atributo de <relaçãoB>, tendo ambos o mesmo tipo; T e U podem estar qualificados com os nomes das relações às quais pertencem,
- <condição-sel> é uma condição de seleção de uma das seguintes formas: T=v, T<v, T<=v, T>v, T>=v ou T<>v, onde:
  - T é o nome de um atributo de <relaçãoA> ou de <relaçãoB>, podendo estar qualificado com o nome da relação à qual pertence; T deverá obrigatoriamente estar qualificado caso existam nas duas tabelas atributos com o mesmo nome de T,
  - v é um valor do tipo do atributo T
    (se v for uma cadeia de caracteres deverá vir entre aspas duplas),
- <lista-atributos-ordem> é uma lista, separada por vírgulas, de nomes de atributos de <relação> que aparecem em <lista-atributos>, que define o critério de ordenação da relação resultante (não há "brancos" antes ou depois das vírgulas), os nomes dos atributos podem estar qualificados com o nome da relação à qual pertencem.

7. Álgebra Relacional

Uma consulta especificada através do comando SELECT deverá ser transformada em uma seqüência de operações algébricas. Estas operações deverão ser armazenadas, uma em cada linha, em um arquivo texto (ascii) com extensão .alg.

Neste trabalho, as operações algébricas terão a seguinte sintaxe:

Seleção com = :	S1(A,T,v,Z)
Seleção com < :	S2(A,T,v,Z)
Seleção com <= :	S3(A,T,v,Z)
Seleção com > :	S4(A,T,v,Z)
Seleção com >= :	S5(A,T,v,Z)
Seleção com <> :	S6(A,T,v,Z)
Seleção com = e a relação A ordenada por T:	O1(A,T,v,Z)
Seleção com < e a relação A ordenada por T:	O2(A,T,v,Z)
Seleção com <= e a relação A ordenada por T:	O3(A,T,v,Z)
Seleção com > e a relação A ordenada por T:	O4(A,T,v,Z)
Seleção com >= e a relação A ordenada por T:	O5(A,T,v,Z)
Seleção com <> e a relação A ordenada por T:	O6(A,T,v,Z)
Junção:	J1(A,B,<condição-jun>,Z)
Junção com relações ordenadas pelos atributos envolvidos na condição da junção:	J2(A,B,<condição-jun>,Z)
Projeção:	P1(A,<n>,<lista-de-atributos>,Z)
Projeção com ordenação:	P2(A,<n>,<lista-de-atributos>,Z,<ordem>)
Projeção com eliminação de duplicatas:	P3(A,<n>,<lista-de-atributos>,Z)
Projeção com ordenação e eliminação de duplicatas:	P4(A,<n>,<lista-de-atributos>,Z,<ordem>)

Onde:
- A e B são relações de entrada,
- T é o nome de um atributo da relação A,
- v é um valor do tipo do atributo T,
- Z é a relação de saída (Z deve ser diferente de A e de B):
    - após cada operação, a relação resultante será gravada nos arquivos Z.ctl e Z.dad,
- <condição-jun> é uma condição de junção da forma T=V, onde:
    - T é um atributo da relação A,
    - V é um atributo da relação B,
- <n> é o número de atributos da projeção,
- <lista-de-atributos> é uma lista de atributos da relação A, da forma $B_1,B_2,...,B_n$,
- <ordem> é um subconjunto da lista de atributos <lista-de-atributos>.

Cada operação listada deverá ser implementada por uma rotina específica. A operação >(A,T,v,Z), por exemplo, deve realizar a seleção sobre a relação A, gerando a relação Z, onde a condição de seleção é T>v. Algoritmos para implementar as operações da álgebra relacional podem ser encontrados no Navathe.

Para evitar ambiguidade nos nomes dos atributos, no caso de a relação de saída Z ser resultante de uma junção, os nomes dos atributos de Z deverão estar concatenados com o nome da relação de entrada. Caso a junção seja entre a relação A e ela mesma, os nomes dos atributos deverão estar concatenados com os nomes A1 e A2.

8. Considerações Gerais

O SGBD BDUFF aceita (como entrada) um arquivo texto (ascii) com extensão .sql contendo um dos comandos SQL apresentados anteriormente.

O comando SELECT é traduzido para a respectiva seqüência de operações algébricas. Esta seqüência é armazenada em um arquivo. Em seguida, é lida e executada. A execução de cada operação algébrica consiste em ler as relações de entrada envolvidas, executar a respectiva operação, e gravar em disco a relação resultante.

De acordo com o que foi definido na Seção 6, o comando SELECT poderá ter duas formas. A primeira deverá ser traduzida em duas (ou uma) operações algébricas: seleção e projeção (ou somente projeção). E a segunda deverá ser traduzida em três (ou duas) operações algébricas: seleção, junção e projeção (ou junção e projeção). A rotina de projeção a ser utilizada vai depender da ocorrência ou não das cláusulas DISTINTCT e ORDER BY.

Não é necessário verificar a correção sintático-semântica das operações algébricas de entrada, pois serão geradas na tradução do SELECT em álgebra, porém, será necessário verificar a correção sintático-semântica dos comandos SQL.

Quando o comando SQL contiver um erro, uma mensagem informativa deve reportá-lo.

As palavras reservadas dos comandos SQL, assim como os nomes dos atributos e das relações, podem ser escritas em minúsculas ou maiúsculas. Porém o conteúdo das cadeias de caracteres será “case sensitive”.

O resultado de um comando SELECT deve ser não somente gravado em um arquivo, mas também exibido na tela.

As definições dos formatos dos arquivos .ctl e .dad e a sintaxe das operações algébricas e dos comandos SQL deverão ser rigorosamente seguidas.

O programa será ativado da seguinte maneira, sem interface gráfica:

BDUFF <comandoSQL> <enter>

onde <comandoSQL> é o nome do arquivo txt com extensão .sql que contém um comando SQL a ser executado. Os arquivos que compõem as relações do banco de dados devem estar armazenados no mesmo diretório que o executável BDUFF.

9. Tarefas para 2015.2

O trabalho deste período consistem em implementar todas as operações da álgebra relacional especificadas na Seção 7 deste documento. O programa desenvolvido deverá: (a) aceitar um arquivo texto (ascii) com extensão .alg, que deverá conter uma sequência de comandos algébricos, um em cada linha, e (b) executar essa sequência de operações algébricas.

Em uma data até 21/03, o monitor Bernardo deverá testar, com uma bateria de teste própria, cada programa junto com os alunos. Esse teste será realizado em uma data que os alunos deverão agendar. O monitor poderá repetir o teste até que não haja mais erros no programa. Em 22/03, o Bernardo me entregará um relatório informando o resultado dos testes realizados. O grupo que realizar o melhor trabalho, na avaliação do monitor, receberá (cada aluno) 0,5 na nota da P2.

Os fontes do trabalho deverão ser enviados a mim por email até o dia 21/03 e uma versão impressa deverá ser entregues também a mim na mesma data.

As apresentações para o monitor deverão ser feitas por todos os elementos do grupo. Não em dupla, como no primeiro trabalho.

Na aula do dia 23/03, chamarei (poderei escolher ou sortear) um aluno de cada grupo para me apresentar detalhes da implementação. Essa apresentação será importante para a definição de nota. Portanto, para não prejudicar os colegas, quem não participar do trabalho, não deverá colocar seu nome como integrante do grupo. Quem não comparecer nessa data para a apresentação não receberá a nota deste trabalho.