Licenciatura em Engenharia Informática
Departamento de Eletrónica, Telecomunicações e Informática
Universidade de Aveiro
Compiladores
2020/2021

Linguagem Bdex

Sophie Jane Pousinho, 97814
Andreia de Sá Portela, 97953
Raquel da Silva Ferreira, 98323
Rodrigo França Lima, 98475
Vicente Samuel Costa, 98515
Patrícia Matias Dias, 98546

Índice

1. Linguagem Secundária

1.1 - Introdução

A linguagem secundária é encarregue de adicionar a possibilidade de se dar load ou upload de tabelas de ou para ficheiros do tipo csv. Também há a possibilidade de dar upload de tabelas para ficheiros txt.

1.2 - Load de uma tabela através de um ficheiro .csv

As tabelas nas quais se vai aplicar load são compostas por uma linha, representante do nome das colunas, e múltiplas linhas seguintes representando a informação (Data) da tabela.

A instrução encarregada de fazer load é a seguinte:

```
Table newTable = READ(filename)
```

- A variável “newTable” é uma **TABLE** e representa a variável table na qual se dará load da data presente no ficheiro .csv que se pretende dar load.
- A variável “filename” é uma **string** e representa a variável correspondente ao nome do ficheiro .csv que se pretende dar load.

1.3 - Upload de uma tabela através de um ficheiro .csv

A instrução encarregada de fazer save da table num ficheiro .csv é a seguinte:

```
SAVE(filename, table_name)
```

- A variável “filename” é uma String e representa o nome do file .csv ou .txt onde será guardada a informação da tabela pretendida.
- A variável “table_name” é uma String e representa o nome da tabela na qual se quer guardar a data no respectivo file.

2. Linguagem Principal

2.1 - Introdução

A linguagem desenvolvida, denominada por **Bdex** tem como objetivo a possibilidade de criação e de interação com tabelas de informação. A interação com tais tabelas é possível através da criação de variáveis, de colunas ou linhas compostas por variáveis, da possibilidade de juntar tabelas e iterar sobre elas tendo em conta regras bem definidas.

2.2 - Estrutura da Linguagem

A linguagem desenvolvida é case insensitive, tem-se então que, por exemplo, LINE e Line e line representam a mesma ação. A linguagem possui métodos de iteração pelas tabelas, esta iteração permite executar código proveniente de cada elemento iterado, código este delimitado por { e }.

As Tabelas são compostas por colunas e linhas, uma tabela pode também não conter nenhuma linha ou coluna desde que tenha um nome.

As tabelas podem ser representadas no terminal através da função PRINT.

Para inserção de colunas ou de linhas é possível adicionar ou uma variável linha ou coluna ou dar join a uma coluna. Para adicionar colunas ou linhas estas devem cumprir certos requerimentos. Para variáveis linhas, estas devem ter os elementos suficientes para preencher as colunas da tabela à qual se pretende inserir a linha, os elementos da linha também devem cumprir o tipo de dados que a coluna possui. Para variáveis do tipo coluna esta deve possuir um nome único comparado com as outras colunas da mesma tabela, esta também deve possuir o mesmo número de elementos que as linhas atuais da tabela na qual se pretende aderir a tabela. Para se poder executar um join entre tabelas estas devem ser compatíveis devendo estas possuir as colunas necessárias ou as linhas necessárias para tal.

As Tabelas aplicam a noção de templates, a variável do tipo Template será o género de Table e poderá ser usada em múltiplas tabelas, esta variável irá definir que tipo de colunas e data é que a Table que aplica tal template irá possuir.

Operações aritméticas podem ser usadas para definições de variáveis ou condições booleanas, podendo-se usar a operação soma, diferença, multiplicação, divisão e resto da divisão.

As condições booleanas podem ser usadas tanto como condições para iteração sobre tabelas ou como condições para extração de dados e sub-tabelas das tabelas podendo-se usar: <,>,<=,>=,!=,==, disjunção (&&) e conjunção (||).

2.3 - Variáveis e Tipos

A criação das variáveis é feita através de assignment do tipo: a = 1, indicando sempre o tipo de dados da variável a. Pode-se dar assignment através de métodos mais complexos como a = 1 + 1 ou como:

Int x = INPUT("Insira um número: ");

Na linguagem foram utilizados diversos tipos de dados, tipos estes que foram Numeric Real, Numeric Int, Text, Boolean, Table, Template, Column, Line e Expr.

Para variáveis Boolean, Text, Numeric Int, Numeric Real e Expr basta dar assign ao valor:

Boolean:
BOOL var = True/False;

Text:
STRING var = “string”;

Numeric Int:
INT var = numInteiro;

Numeric Real:
REAL var = numReal;

Expr:
Expr var = Where predicado (ex: 10 < 5);

Para variáveis do tipo Table, a tabela terá que ser criada ou através do nome desta ou através do nome e as respectivas colunas com os seus respectivos tipos de data, pode-se também dar load de um file .csv que a contenha ou retirar a tabela através de outra ou fazer um join. E ainda por uma operação aritmética de uma coluna de uma tabela com um valor numérico.

Table newTable = READ(fileName);
Table newTable = GET table_name attribute attribute ... WHERE conditions;
Table newTable = table1 JOIN table2;
Table newTable = table1 JOIN table2 ON conditions;
Table newTable = table1 WHERE condition;
Table newTable = table1.colunaNumerica operação valor; (ex: table1.ano + 1)

Para variáveis do tipo Template, pode-se criar através do nome e do tipo das colunas pertencentes a esse template. Como por exemplo:

TEMPLATE departamento [nome:String][nmec:Int] ;

2.4 - Métodos

A linguagem Bdex aceita diversos tipos de métodos, aplicáveis tanto para tabelas como linhas e colunas. Os métodos possíveis são CREATE, ADD, DELETE, INPUT, PRINT, FOR, GET, WHERE, JOIN, JOIN ON.

CREATE:
CREATE template [attribute1:data_type] [attribute2:data_type] ...;

ADD:
ADD tablename LINE (value1, value2,...) ;

DELETE:
DELETE template_name/table;
DELETE tablename LINE WHERE condition;

INPUT:
Int x = INPUT(message);

PRINT:
PRINT(tablename) ;

FOR:
FOR (Line var) tablename { code_to_be_executed }
FOR (type var) tableName.colName { code_to_be_executed }

GET:
STRING var = GET tablename colname WHERE condition;

WHERE:
STRING var = GET tablename colname WHERE condition;
TABLE t = tablename WHERE condition;
BOOL b = tablename where val;

JOIN:
TABLE tablename = tablename1 JOIN tablename2;

JOIN ON:
TABLE tablename = tablename 1 JOIN tablename2 ON column1 = column2;

2.5 - Operações aritméticas

As operações aritméticas são entre dois operandos, podendo estas operações estarem nested noutras operações aritméticas.
Para as operações utilizou-se a seguinte sintaxe:

- != para diferença
- == para igualdade
- <,>,<=,>= para verificação de menor, maior, menor ou igual, maior ou igual
- && para disjunções
- || para conjunções

2.6 - Input e Output

É possível a utilização de input de texto através do método INPUT, sendo possível assim assignment de valores a variáveis:

Int x = INPUT(message);
Também é possível, através da utilização do método PRINT, mostrar as tabelas no terminal como output:

PRINT(tablename) ;

2.7 - Estruturas Condicionais

A linguagem permite o uso de uma estrutura condicional, o método WHERE. Este avalia a condição presente para devolver um boolean. Esta condição será usada para filtering de valores que cumpram a condição, como por exemplo:

STRING var = GET tablename colname WHERE condition;
DELETE tablename LINE WHERE condition;
Table t = tablename WHERE condition;

2.8 - Ciclos

Existe também a possibilidade de uso de ciclos de código, exemplo é

FOR (Line var) tablename { code_to_be_executed }
FOR (type var) tableName.colName { code_to_be_executed }

2.9 - Comentários

A linguagem permite a noção de comentários, os comentários podem ser-se feitos através de duas maneiras:

Inline:

```
# Comentário de uma linha
```

Multi-line:

```
/*
    Comentário de múltiplas linhas
*/
```