DBMS Relational Algebra Project

Faizan Khan

17 October 2016

The Project

This readme file describes the details of the **DBMS Project II: Implementation of Relational Algebra functions**.

Aim of the Project

To write a program to implement Relational Algebra functions present in DBMS.

Functions implemented

The program implements the following Relational Algebra functions:

1. SELECT
2. PROJECT
3. RENAME
4. UNION
5. SET DIFFERENCE
6. CARTESIAN PRODUCT

Project Details

The program assumes the following:

- The database folder is present in the same directory as the main project file
- The database folder has a *db_info.txt* file containing the details of the database
- The database folder has a *tables* subdirectory which contains all table files in space separated form in *.txt* format
- The tables have their schema in the top three lines:
  - First line has table name
  - Second line has column names
  - Third line has column data types
- The data in the files is valid:

  - There is no newline
  - All data has same number of columns (except) table name
  - The data types corresponding to the data are correct
- The data types allowed are:
  - *int*
  - *float*
  - *varchar*
- Comparison accross data types is not allowed
- Queries are delimited by a **;**

Syntax

The basic syntax of the queries is as follows:

1. Show tables in database as specified in *db_info.txt*:

   ```
   SHOW_TABLES;
   ```

2. SELECT:

   ```
   SEL{predicate}(table_name);
   ```

3. PROJECT:

   ```
   PROJECT{column_name1:column_name2}(table_name);
   ```

4. RENAME:

   ```
   REN{new_table_name|column_name1:column_name2}(table_name);
   ```

5. UNION:

   ```
   UNI(table_name1, table_name2);
   ```

6. SET DIFFERENCE:

   ```
   DIF(table_name1, table_name2);
   ```

7. CARTESIAN PRODUCT:

   ```
   CRP(table_name1, table_name2);
   ```

8. EXIT:

   ```
   EXIT;
   ```

The predicate syntax for SEL:

```
SEL{(a > b | c = d) & e ! f}(table_name)
```

The available operators are:

1. +: Add two *ints* or *floats*
2. -: Subtract two *ints* or *floats*
3. *: Multiply two *ints* or *floats*
4. /: Divide two *ints* or *floats*
5. =: Compare two operands, `TRUE` if equal
6. **!**: Compare two operands, `TRUE` if unequal
7. >: Compare two operands, `TRUE` if former is greater than latter
8. <: Compare two operands, `TRUE` if former is lesser than latter
9. &: Logical `AND`
10. |: Logical `OR`

NOTES:

- Comparison between different data types is not allowed
- `varchar` (Strings) are enclosed in `'`
- `float` operands *must* have a `.`
- Invalid operands are not allowed

The predicate syntax for PRO:

```
PRO{a:b:c:d}(table_name)
```

NOTES:

- Column names must be separated by a `:`
- Column names must be present in table
- Column names are displayed in given order

The predicate syntax for REN:

```
REN{new_table_name|a:b:c:d}(table_name)
REN{new_table_name}(table_name)
```

NOTES:

- Table name and column names must be separated by a `|`
- Column names must be separated by a `:`
- Column names must be valid
- Column names are renamed in given order
- If no column names are provided, the colum names are retained
- Either all or none of the column names must be provided

Sample Queries

```
1. SEL{faculty.salary > 80000 & dept = 'CSE'}(faculty);
2. PRO{id:name:grade}(studuent);
3. UNI(PRO{name}(faculty), PRO{name}(student));
4. DIF(PRO{name}(faculty), PRO{name}(student));
5. CRP(student, course);
6. REN{stud|stud_id:name:cgpa}(student);
```

For more queries, see **Query** file

Comments:

- Syntax errors are not allowed
- SELECT takes exactly one predicate
- RENAME **cannot rename to a table already present**
- Column and table names must be alphanumeric
- Invalid comparisons between data types are not allowed
- UNI and DIF operations must **have same column names and data types**. Order need not be same
- Nested queries are allowed
- CRP operation needs **different column names**
- PRO should have **unique column names**
- All whitespaces are ignored
