DBMS PROJECT-2

Database System Development

Devang Kulshreshtha
14075021
B.Tech CSE

Compilation and Execution :

To compile the program, move on to the PROJECT2 directory and type the command in the terminal (ubuntu) -

```
g++ -std=c++11 code.cpp -o code1
```

To execute the program, run -

```
./code1
```

When program starts, user can enter queries in the formats specified in the later section. The programs exits when user gives “exit()” as the input.

Files Description:

- code.cpp : The main .cpp file of the system. Run this file in an IDE or compile and execute ot from the terminal to start the program.
- code : Compiled file which can be directly executed from the ubuntu terminal.
- Inputf.in : text file containing a list of queries that can be directly executed by copy-paste in the program.
- DBMetadata.txt : text file containing metadata(table name, schema, primary key) information about existing tables.
- {table_name}.txt : text file for each of the existing tables containing records in text format.

When new table is created(CREATE query) or a table is updated(INSERT query) , new files are formed as well as existing files(e.g. “DBMetadata.txt”) are updated. At the start, 2 tables(Aquarium and Museum) are present in the database.

Queries Format :

- CREATE

  CREATE *Table_name* {*column_name1,column_name2,...*} {*column_name1,...*}

  The *Table_name* parameter specifies the name of the relation.
  The first *column_name* parameters specify the names of the columns of the table.
  The second *column_name* parameters specify the names of columns which act as the primary key of the table.

Example:

- ❏ CREATE Student {Roll,Name,Contact,Age} {Roll,Age}
- ❏ CREATE Employee {EmpID,Name,Contact,Age} {EmpID,Name}

- INSERT

  INSERT *Table_name* {*value1,value2,value3,...*}

  The *Table_name* parameter specifies the name of the relation.
  The *value* parameters specify the values to be inserted in the relation. The values given must be in the order of relation schema.

  Example:

  - ❏ INSERT Student {14075021,Devang,9621633895,20}
  - ❏ INSERT Employee {39,Meera,9456462693,40}

- SELECT

  SELECT *Table_name*
  {column_name:operator:value,column_name:operator:column_name,...}
  {column_name,column_name,...} (for simple query)

  SELECT [*Table*]
  {column_name:operator:value,column_name:operator:column_name,...}
  {column_name,column_name,...} (for nested query)

  The *Table_name* specifies the name of the relation on which SELECT operation is to be performed.
  The *Table* specifies the table on which SELECT operation is to be performed.
  The parameters in the 1st {} braces specify the conditions to be satisfied by the relation records. Similar to WHERE conditions in SQL.
  The parameters in the 2nd {} braces specify the columns to be projected finally.

  Example:

  - ❏ SELECT Employee {} {EmpID,Name}
  - ❏ SELECT Student {Age:=:20,Roll:=:14075020} {Roll,Age}
  - ❏ SELECT [SELECT [SELECT Employee {} {EmpID,Name,Contact,Age}] {EmpID:>:39} {Contact,Age,Name}] {} {Age}

  Here the 3rd query is doubly nested.

- PROJECT

PROJECT *Table_name* {column_name1,column_name2,...} (for simple query)

PROJECT [*Table*] {column_name1,column_name2,...} (for nested query)

Parameters are same as in SELECT query.

Example:

- ❏ PROJECT Student {Age,Name}
- ❏ PROJECT [SELECT Student {} {Age,Name,Contact}] {Age,Name}

Here the 2nd query is nested.

- <u>RENAME</u>

RENAME *Table_name* {*New_Table_name*} {col1:col11,col2:col22,...} (for simple query)

RENAME [*Table*] {*New_Table_name*} {col1:col11,col2:col22,...} (for nested query)

New_Table_name is the name of the new table. If left blank table name is not changed. Col1,col2,.. are initial column names and col11,col22,... are names after renaming.

Example:

- ❏ RENAME Student {NewStudent} {Name:Naam,Contact:Telephone}
- ❏ RENAME [SELECT Student {} {Name,Contact,Rollno}] {NewStudent} {Name:Naam,Contact:Telephone}

Here the 2nd query is nested.

- <u>UNION / INTERSECTION / DIFFERENCE / CARTESIAN PRODUCT</u>

TYPE *Table1_name Table2_name*
TYPE [*Table1*] [*Table2*]
TYPE *Table1_name* [*Table2*]
TYPE [*Table1*] *Table2_name*

Where TYPE = “UNION” or “INTERSECT” or “DIFFERENCE” or “CROSS”
If *table1* and *table2* are different, then the result is renamed to *table1*.name+*table2*.name

Example:

- ❏ UNION [SELECT Student {} {Contact,Age}] [SELECT Employee {} {Contact,Age}]
- ❏ SELECT [CROSS Student Employee] {Roll:<=:EmpID} {Student.Name,Student.Age,Employee.Age}

- INTERSECT [SELECT Student {} {Roll,Contact,Age}] [SELECT Student {} {Roll,Contact,Age}]