Formal Relational Query Parser

Documentation

Prerequisites

Before reading this documentation, you should have the knowledge of Relational Algebra and how the queries are written in Relational Algebra. A knowledge of fundamental operations of Relational Algebra is sufficient.

Basics

The relational algebra is a procedural query language. It consists of a set of operations that take one or two relations as input and produce a new relation as their result. The fundamental operations in the relational algebra are select,project,union,set difference,Cartesian product, and rename.

Select (table , args)

Select operation is used to select the tuples from a particular relation which satisfy some conditions. It will take the table and the args as input, the args will be the condition as a string.

In Formal Relational Query Parser, the args should be in *principal disjunctive normal form*, without any parantheses and the table name should be an existing relation in the db dump.

SELECT(table) condition1|condition2|......

Project (table , args)

Project operation is used to project the contents of particular columns from a relation. It will take the table and the args as input, the args will be the names of the columns.

In Formal Relational Query Parser, the args should be string containing the names of the columns separated by comma and the table name should be an existing relation in the db dump.

PROJECT(table) attribute1,attribute2,.....

Rename (table , args)

Rename operation is used to rename a particular relation. It will take the table and the args as input, the args will be the condition as a string.

In Formal Relational Query Parser, the args should be the new name of the table, or the new name of the table with the new names of the columns of the table and the table name should be an existing relation in the db dump.

RENAME(table) new_name

OR, RENAME(table_name) new_name{new names of all attributes separated by comma}

Union (table , table)

Union operation is used to select the tuples from two relations which are the common tuples in both relations. It will take two tables as input.

In Formal Relational Query Parser, the first table should be enclosed in '( )' and the second table in '[ ]' and the table names should be existing relations in the db dump.

UNION(table1) [table2]

SetDifference (table , table)

SetDifference operation is used to select the tuples a relation which are not present in the other relation. It will take two tables as input.

In Formal Relational Query Parser, the first table should be enclosed in '( )' and the second table in '[ ]' and the table names should be existing relations in the db dump.

SETDIFFERENCE(table1) [table2]

CartesianProduct (table , table)

CartesianProduct operation is used to get the Cartesian product of two tables/realtions. It will take two tables as input.

In Formal Relational Query Parser, the first table should be enclosed in '( )' and the second table in '[ ]' and the table names should be existing relations in the db dump.

CARTESIANPRODUCT(table1) [table2]

All the other RA operations can be done with the help of these fundamental operations.

Simple Queries

The simple queries carry out just one RA operation. They will follow the following syntax:

operation_name(table_name) args

For examples,

Query >> SELECT(Student) Name="Bab"&Dept="CSE"|Roll>5

Name	Dept	Roll
Bab	CSE	5
Ayush	CSE	14
Atishay	EE	10
Ayush	MNC	12

Query >> PROJECT(Student) Name,Dept

Name	Dept
Anant	ECE
Atishay	EE
Ayush	CSE
Ayush	MNC
Bab	CSE

Query >> RENAME(Faculty) Prof{ProfId,ProfName,Department,Sex}

ProfId	ProfName	Department	Sex
101	Ayush	CSE	M
102	Anant	CSE	M
201	Bab	CSE	M
202	Atishay	EE	M

Query >> UNION(Scholar) [Student]

Name	Dept	Roll
Anant	ECE	5
Atishay	EE	10
Ayush	CSE	14
Ayush	MNC	12
Bab	CSE	5

Query >> SETDIFFERENCE(Student) [Scholar]

Name	Dept	Roll
Anant	ECE	5
Ayush	MNC	12

Query >> CARTESIANPRODUCT(Student) [Department]

Name	Dept	Roll	Id	DeptName
Bab	CSE	5	1	CSE

Bab	CSE	5	2	MNC
Bab	CSE	5	3	ECE
Bab	CSE	5	4	EE
Ayush	CSE	14	1	CSE
Ayush	CSE	14	2	MNC
Ayush	CSE	14	3	ECE
Ayush	CSE	14	4	EE
Anant	ECE	5	1	CSE
Anant	ECE	5	2	MNC
Anant	ECE	5	3	ECE
Anant	ECE	5	4	EE
Atishay	EE	10	1	CSE
Atishay	EE	10	2	MNC
Atishay	EE	10	3	ECE
Atishay	EE	10	4	EE
Ayush	MNC	12	1	CSE
Ayush	MNC	12	2	MNC
Ayush	MNC	12	3	ECE
Ayush	MNC	12	4	EE

Nested Queries

The nested queries carry out multiple RA operations. They will generally follow the following syntax:

operation1(operation2(......(table_name) ....) args2) args1

For examples,

Query >>SELECT(UNION(PROJECT(Student) Name) [PROJECT(Faculty) Name]) *

```
Name
Anant
Atishay
Ayush
Bab
```

Query >> PROJECT(PROJECT(SELECT(Student) *) Name,Dept) Dept

```
Dept
CSE
ECE
EE
MNC
```

Query >> SETDIFFERENCE(Student) [SETDIFFERENCE(Student) [Scholar]]

```
Name        Dept        Roll
Bab         CSE         5
Ayush       CSE         14
Atishay     EE          10
```

Query >> RENAME(UNION(PROJECT(Student) Name,Dept) [PROJECT(Scholar) Name,Dept]) PersonalDetails

```
Name        Dept
Anant       ECE
Atishay     EE
Ayush       CSE
Ayush       MNC
Bab         CSE
```

Query >> UNION(UNION(PROJECT(Student) Name) [PROJECT(Faculty) Name]) [PROJECT(Scholar) Name]

```
Name
Anant
Atishay
Ayush
Bab
```

Some other operations that can be done by nesting the operations include *Intersect, Divide operator* etc.

INTERSECT(table1,table2):
SETDIFFERENCE(table1) [SETDIFFERENCE(table1) [table2]]

DIVIDE(relation,table1):

```
PROJECT(SETDIFFERENCE(CARTESIANPRODUCT(table1) [table2]) [relation]) args
```

For exit, simply type the EXIT in the query.