

Students: Luke Ryssel, Richard Deming, Doug Barnes

Course Project: Project 2

Class: CS 360

This program expects all input to be in relation algebra and will not run properly when given other input. We recommend using the given symbols below.

Symbols:

Π	Project
σ	Select
\wedge	AND
\vee	OR
\bowtie	Natural Join
$\bowtie\rightarrow$	Natural Right Outer Join
$\rightarrow\bowtie$	Natural Left Outer Join
$\bowtie\llcorner$	Outer Join
\cap	Intersect
\cup	Union
$-$	Set Difference

Programs capabilities:	Incapable of:
Project	Select inside select statement
Select	Rename operation
Equivalent of SQL 'from'	Assignment operations for temporary tables
Union operations	Generalized projection
Set difference	Aggregate functions
Cartesian products	multiset relational algebra
Intersect	
Natural join	
Left Outer join	
Right outer join	
Full outer join	

Project

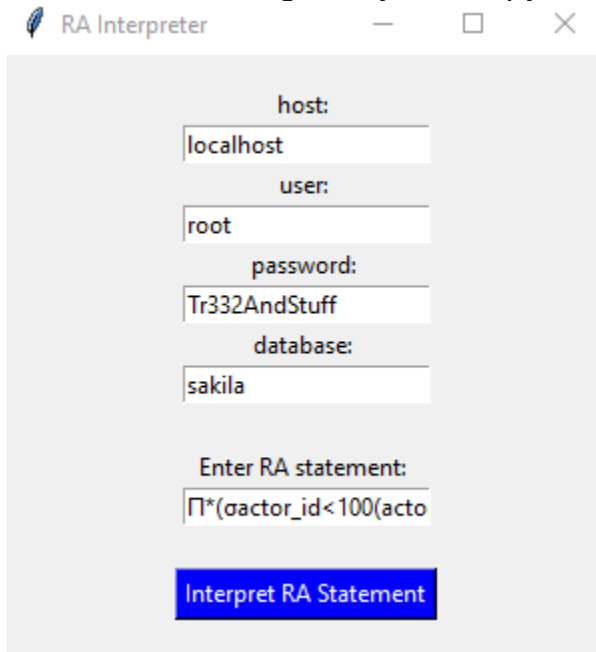
This project required us to design a RA (relational algebra) interpreter for SQL. In short, this program takes an RA expression from an SQL database and changes it to its logical equivalence in SQL.

The interpreter is written in python with the **MySQL connector** imported (used to import MySQL data) and **Tkinter** (used to design GUI in python). Although the program is written to run in Windows CMD it can be ran from other python platforms, for example, Visual Studio Code or a Linux terminal.

User Interface:

The User interface opens with the default database settings, but allows you to change them to your database settings before running an interpret command.

The UI is defined globally inside python, so there is not a specific function for it.

A screenshot of a web-based user interface titled "RA Interpreter". It features five input fields for database configuration: "host:" with "localhost", "user:" with "root", "password:" with "Tr332AndStuff", and "database:" with "sakila". Below these is a larger text area labeled "Enter RA statement:" containing the query $\Pi^*(actor_id < 100)(acto$. At the bottom is a blue button labeled "Interpret RA Statement".

host:
localhost

user:
root

password:
Tr332AndStuff

database:
sakila

Enter RA statement:
 $\Pi^*(actor_id < 100)(acto$

Interpret RA Statement

Python Functions:

interpretRA() – gets the relational algebra string from the UI code, and parses and transforms it into a MySQL statement. It then calls runscript(sqlStatement)

```
def interpretRA():  
    relationalStatement = entry.get()  
  
    table1 = ""  
    table2 = ""  
    setDiff = False  
    intersect = False  
    findPredicate = True  
    findIntPredicate = True  
  
    sqlStatement = "("  
    selectStatement = ""  
    fromStatement = ""  
    whereStatement = ""
```

```

predicate = ""
predicateIntersect = ""

for x in range( len(relationalStatement) ):
    if relationalStatement[x] == "U":
        sqlStatement = "(" + selectStatement + fromStatement + whereStatement
+ ") UNION ("
        selectStatement = ""
        fromStatement = ""
        whereStatement = ""
    if relationalStatement[x] == "∩":
        table1 = "(" + selectStatement + fromStatement + whereStatement + ")"
        selectStatement = ""
        fromStatement = ""
        whereStatement = ""
        intersect = True
    if relationalStatement[x] == "-":
        table1 = "(" + selectStatement + fromStatement + whereStatement + ")"
        selectStatement = ""
        fromStatement = ""
        whereStatement = ""
        setDiff = True
    if relationalStatement[x] == "Π":#Project symbol found, start writing sel
ect statement
        selectStatement = selectStatement + "SELECT "
        i = 1
        n = relationalStatement[x + i]
        if findIntPredicate == True:
            while n != "(":
                predicateIntersect = predicateIntersect + n
                i = i + 1
                n = relationalStatement[x+i]
            findIntPredicate = False
        i = 1
        n = relationalStatement[x + 1]
        while n != "(":
            selectStatement = selectStatement + n
            i = i + 1
            n = relationalStatement[x + i]
        selectStatement = selectStatement + " "

    if relationalStatement[x] == "(":# First paranthese found.
        if relationalStatement[x+1] == "σ":#Select symbol found, start writin
g where statement
            whereStatement = whereStatement + "WHERE "

```

```

i = 2
n = relationalStatement[x + 2]
if findPredicate == True:
    while n != "<" and n != ">" and n != "=":
        predicate = predicate + n
        i = i + 1
        n = relationalStatement[x+i]
    findPredicate = False
i = 2
n = relationalStatement[x + 2]
while n != "(":

    if n == "^":
        whereStatement = whereStatement + " and "
    elif n == "v":
        whereStatement = whereStatement + " or "
    else:
        whereStatement = whereStatement + n
    i = i + 1
    n = relationalStatement[x+i]
    whereStatement = whereStatement + " "
else:#There was no select sybmol after a forward facing paranthese, o
r there was another open paranthese so start the from statement
    fromStatement = fromStatement + "FROM "
    i = 1
    n = relationalStatement[x + i]
    while n != ")":
        if n == "⋈":
            fromStatement = fromStatement + " natural join "
        elif n == "⋈⋈":
            fromStatement = fromStatement + " natural right outer join "
n "
        elif n == "⋈⋈⋈":
            fromStatement = fromStatement + " natural left outer join "
"

        elif n == "X":
            fromStatement = fromStatement + ", "
        elif n == "⋈⋈⋈":
            fromStatement = fromStatement + " full join "
        else:
            fromStatement = fromStatement + n
        i = i +1
        n = relationalStatement[x+i]
    fromStatement = fromStatement + " "
if setDiff == True:

```

```

        table2 = "(" + selectStatement + fromStatement + whereStatement + ")"
        sqlStatement = selectStatement + "from " + table1 + "as t1 " + "natural 1
left join " + table2 + "as t2 " + "where t2." + predicate + " IS NULL;"
        if intersect == True:
            table2 = "(" + selectStatement + fromStatement + whereStatement + ")"
            sqlStatement = selectStatement + "from " + table1 + "as t1 " + "where " +
predicateIntersect + " in " + table2
        else:
            sqlStatement = sqlStatement + selectStatement + fromStatement + whereStat
ement + ")"
        runScript(sqlStatement)
        popupmsg(sqlStatement)

```

Runscript(sqlStatement) – runscript creates the connection to the mySQL database and sends the sql string. It then sends any output or errors from the mySQL server to the terminal window.

```

def runScript(myScript):
    db = mysql.connector.connect(
        host = hostEntry.get(),
        user = userEntry.get(),
        passwd = passwdEntry.get(),
        database = databaseEntry.get()
    )
    mycursor = db.cursor()
    mycursor.execute(myScript)
    for x in mycursor:
        print(x)

```