**Relational Algebra 2 SQL (RA2SQL)**

**Operadores en las condiciones:**

**Menor que:** <

**Mayor que:** >

**Igual:** =

**Distinto:** !=, <>

**Menor o igual:** <=

**Mayor o igual:** >=

**Y:** and, &

**O:** or, |

**No:** not, !

**Parentesis:** ()

**Aritmeticos:** + - \* / %

**Select:**

**Sintaxis:**

select[condicion](relacion)

select(relacion)

**Ejemplos:**

select[nota>90](estudiantes);

select[salario>350000 and dept=3](empleado);

select(productos);

**Project:**

**Sintaxis:**

project[lista\_campos](relacion)

**Ejemplos:**

project[nombre, apellido, identificacion](estudiantes);

project[id, nombre, salario](select[dept != 1](empleado));

**Union:**

**Sintaxis:**

union(relacion1, relacion2)

**Ejemplos:**

union(estudiantes\_progra1, estudiantes\_progra2);

union(select[dept=4](empleados),select[dept=2](empleados));

**intersect:**

**Sintaxis:**

intersect(relacion1,relacion2)

**Ejemplos:**

intersect(vendedores, encargados\_inventario);

intersect(select[])

**Except:**

**Sintaxis:**

except(relacion1,relacion2)

**Ejemplos:**

except(vendedores, encargados\_inventario);

except(select[])

**thetajoin:**

**Sintaxis:**

thetajoin[condicion](relacion1,relacion2)

thetajoin[lista\_campos](relacion1,relacion2)

**Ejemplos:**

thetajoin[emp.dept=dept.id](emp,dept)

thetajoin[empid](empleados, jefes);

**Variables para guardar relaciones:**

**Ejemplos:**

relacion = select[escuela='ecci' and creditos\_matriculados > 10](estudiantes);

project[nombre, apellido, carnet, promedio\_total](relacion)

e = empleado; d = departamento;

thetajoin[e.dept=d.num](e,d);

attribs=id,nombre,telefono,ciudad; //si alcanza el tiempo se implementa esta forma

project[attribs](estudiantes\_bases1);

project[attribs](estudiantes\_automatas);

**Ejemplos del resultado SQL esperado luego de la traduccion**

|  |  |
| --- | --- |
| **Algebra ralacional** | **SQL equivalente** |
| select(empleado); | Select \* from empleado; |
| project[carnet,carrera,nombre](estudiante) | Select carnet,carrera,nombre from estudiante; |
| e = empleado; d = departamento;  thetajoin[e.dept=d.num](e,d); | Select \* from empleado e inner join departamento d on e.dept=d.num; |
| e = empleado; d = departamento;  project[e.nombre,d.titulo](]thetajoin[e.dept=d.num](e,d)); | Select e.nombre,d.titulo from empleado e inner join departamento d on e.dept=d.num; |

**Archivos .RA2SQL**

Un archivo de texto con una o multiples expresiones de algebra relacional terminadas en ;

ejemplo:

**select[puesto='ventas' and ciudad!='san jose'](empleados);**

**r1 = select[total>250000 and excento\_iv=false](facturas);**

**r2 = select[productoid='e-222-1006' and cantidad < 10](detalle\_factura);**

**thetajoin[idfactura](r1,r2);**

se le pasa al compilador asi:

python RA2SQL.py -s -r ejemplo.RA2SQL

supongamos que le agregamos opciones al compilador si da tiempo, por ejemplo -s es separar cada consulta en archivos diferentes y -r reemplazar las variables por lo que significan en el archivo en lugar de escribirlas en la consulta, eso se implementa si da chance.

**Sobre el lenguaje**

**Lista de tokens validos**

|  |  |
| --- | --- |
| **Nombre del token** | **Expresion regular** |
| OP\_COMPARACION | <|<=|>|>=|!=|<> |
| OP\_LOGICO | \||&|!|and|or|not |
| LPAREN | \( |
| RPAREN | \) |
| OP\_ARITMETICO | \+|-|\\*|/|% |
| LPARENC | \[ |
| RPARENC | \] |
| IGUAL | = |
| COMA | , |
| ID | [a-z\_][a-z0-9\_]+ |
| COMILLA | ' |
| SELECT | select |
| PROJECT | project |
| TJOIN | thetajoin |
| BINARIO | union|intersect|except |
| FIN | ; |

**Gramatica**

**S → exp FIN**

**exp → unexp | biexp | asign**

**asign → ID IGUAL campos | ID IGUAL unexp | ID IGUAL biexp**

**unexp → op\_select LPAREN arg RPAREN**

**| PROJECT LPARENC campos RPARENC LPAREN arg RPAREN**

**campos → ID | ID COMA campos**

**arg → ID | unexp | biexp**

**op\_select → SELECT op\_select\_opc**

**op\_select\_opc → LPARENC condicion RPARENC | ε**

**condicion → condicion OP\_LOGICO condicion | op\_condicion | LPAREN condicion RPAREN**

**op\_condicion → ID OP\_LOGICO cadena | ID OP\_ARITMETICO ID | ID IGUAL cadena**

**cadena → COMILLA ID COMILLA | ID**

**biexp → op\_binario LPAREN arg COMA arg RPAREN**

**op\_binario → op\_join | BINARIO**

**op\_join → TJOIN LPARENC join\_args RPARENC**

**join\_args → campos | condicion**