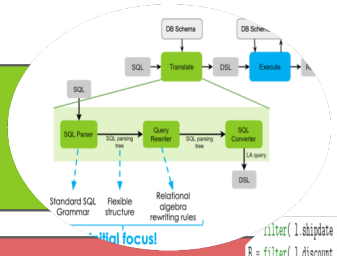
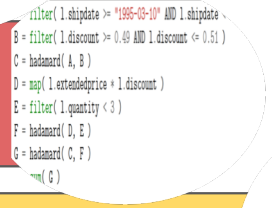
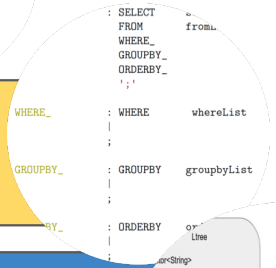
Converting SQL into a Linear Algebra DSL

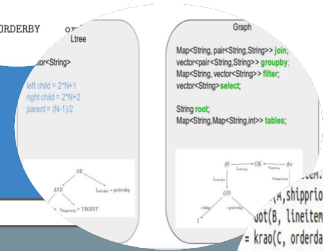
Luís Albuquerque, Rafael Fernandes
2017/2018

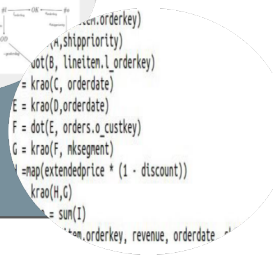
Índice



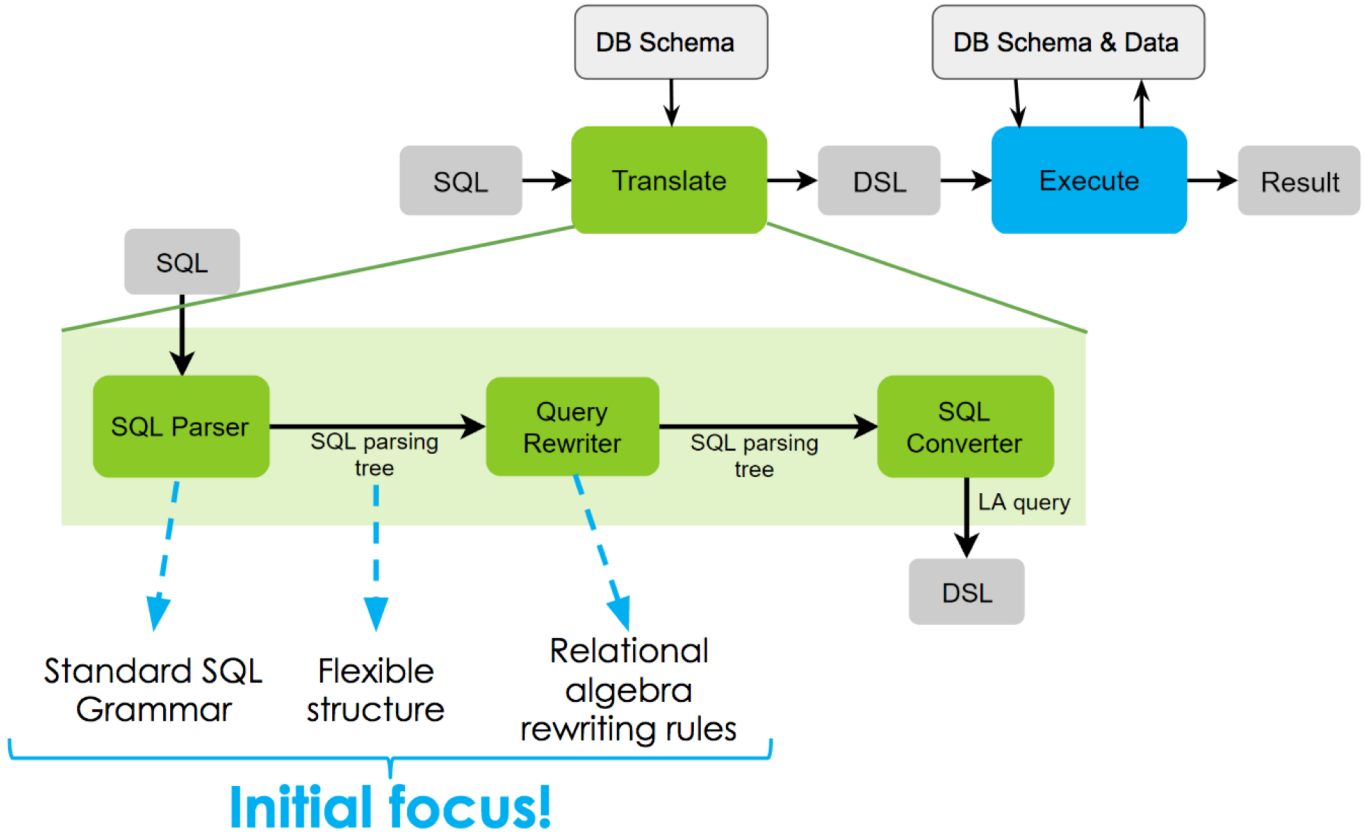








Descrição do Trabalho



Requisitos

```
select
    sum(l_extendedprice * l_discount) as revenue
from
    lineitem
where
    l_shipdate >= date '1995-03-10'
    and l_shipdate < date '1995-03-10' + interval '1' year
    and l_discount between 0.50 - 0.01 and 0.50 + 0.01
    and l_quantity < 3;
```

Desafios:

- não tem ordem específica
- a tradução não é *on-the-fly* (não é diretamente mapeável)
- produzir DSL otimizada

```
A = filter( l.shipdate >= "1995-03-10" AND l.shipdate < "1996-03-10" )
B = filter( l.discount >= 0.49 AND l.discount <= 0.51 )
C = hadamard( A, B )
D = map( l.extendedprice * l.discount )
E = filter( l.quantity < 3 )
F = hadamard( D, E )
G = hadamard( C, F )
H = sum( G )
return( H )
```

GIC

```
SelectBlock     : SELECT     selectList
                  FROM       fromList
                  WHERE_
                  GROUPBY_
                  ORDERBY_
                  ';'

WHERE_          : WHERE       whereList
                |
                ;

GROUPBY_        : GROUPBY    groupbyList
                |
                ;

ORDERBY_        : ORDERBY    orderbyList
                |
                ;

selectList      : selectListN
                | '*'
                ;

selectListN     : selectListNSub
                | selectListN ',' selectListNSub
                ;
```

```
selectListNSub : Term
               | Term AS NAME
               ;

fromList       : subfromList
               | fromList ',' subfromList
               ;

subfromList    : NAME
               | Join NAME
               | Join NAME ON Literal '=' Literal
               | Join NAME AS NAME
               | Join NAME AS NAME ON Literal '=' Literal
               | '{' SelectBlock '}'
```

```
Factor         : Literal
               | NAME '(' Args ')'
               | NOT Factor
               ;

Args           : Args1
               |
               ;

Args1          : Term
               | Args1 ',' Term
               | Args1 ' ' Term
```

```
               | '{' SelectBlock '}' AS NAME
               ;

Join           : JOIN
               | INNER JOIN
               | LEFT JOIN
               | RIGHT JOIN
               | FULL JOIN
               ;

whereList      : Exp
               ;

ExpR           : Exp AND Exp
               | Exp OR  Exp
               ;

Exp            : Term
               | ExpR
               | '(' ExpR ')'
               | NAME IN Inlist
               | NAME BETWEEN Literal AND Literal
               | EXISTS '(' SelectBlock ')' ';'
               | Literal LIKE REGEX
               ;

Term           : Factor
               | Term BBOP Term
               | Term IBOP Term
               ;
```

Gramática tradutora

```
SelectBlock     : SELECT      selectList
                  FROM        fromList
                  WHERE_
                  GROUPBY_
                  ORDERBY_
                  ';'                                   { final();}

WHERE_          : WHERE       whereList                 { ; }
                |                                       { ; }
                ;

GROUPBY_        : GROUPBY     groupbyList               { ; }
                |                                       { ; }
                ;

ORDERBY_        : ORDERBY     orderbyList               { ; }
                |                                       { ; }
                ;

selectList      : selectListN                           { ; }
                | '*'                                   { select_asterisco();}
                ;

selectListN     : selectListNSub                        { ; }
                | selectListN ',' selectListNSub        { ; }
                ;

selectListNSub : Term                                   { select_term($1);}
                | Term AS NAME                          { select_term_name($1,*$3);}
                ;
```

```
fromList        : subfromList                           { ; }
                | fromList ',' subfromList              { ; }
                ;

subfromList     : NAME                                  { subfromlist_name(*$1);}
                | Join NAME                             { ; }
                | Join NAME ON Literal '=' Literal      { subfromlist_join_literal_literal($6,$4);}
                | Join NAME AS NAME                     { subfromlist_join_nameAS(*$4,*$2);}
                | Join NAME AS NAME ON Literal '=' Literal   { subfromlist_join_as_literal_literal(*$2,*$4,$6,$8);}
                | '{' SelectBlock '}'                   { ; }
                | '{' SelectBlock '}' AS NAME           { ; }
                ;

Join            : JOIN                                  {/*$$ = "Normal"*/;}
                | INNER JOIN                            {/*$$ = "Inner"*/;}
                | LEFT JOIN                             {/*$$ = "Left"*/;}
                | RIGHT JOIN                            {/*$$ = "Rigth"*/;}
                | FULL JOIN                             {/*$$ = "Full"*/;}
                ;

whereList       : Exp                                   { ; }
                ;

ExpR            : Exp AND Exp                          {$$ = $1; expr_and($1,$3); }
                | Exp OR  Exp                          {$$ = $1; expr_or($1,$3);}
                ;

Exp             : Term                                  {$$ = itr; exp_term($1);}
                | ExpR                                  {$$ = $1 ; }
                | '(' ExpR ')'                          {$$ = $2 ; }
                ;
```

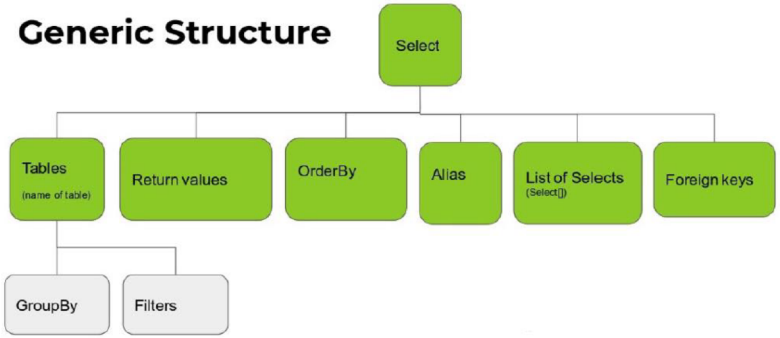
Gramática tradutora

```
groupbyList     : groupbyListSub                          { ; }
                | groupbyList ',' groupbyListSub          { ; }
                ;

groupbyListSub : HAVING NAME '(' Literal ')' BBOP Literal  { ; }
                | Literal                                  { groupbyListSub_literal($1);}
                ;

orderbyList     : orderbyListSub                          { ; }
                | orderbyList ',' orderbyListSub          { ; }

                ;

orderbyListSub : NAME                                      { ; }
                | NAME order                               { ; }
                ;

order           : ASC                                      { ; }
                | DESC                                     { ; }
```

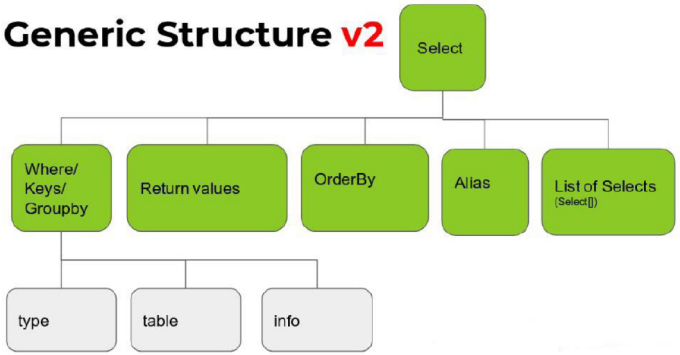
```
                Literal         : NAME                              { literal_name(*$1); $$ = itr2;}


                                | NAME '.' NAME                     {$$ = itr2; literal_name_name(*$1,*$3);}

                                | DATE                              {$$ = itr2; literal_date(*$1);}
                                | CONSTANT                          {$$ = itr2; literal_constant(*$1);}
                                | BOOL                              {$$ = itr2; literal_bool(*$1);}
                                | ANY                               {$$ = itr2; literal_any(*$1);}
                                | ALL                               {$$ = itr2; literal_all(*$1);}
                                ;
```

Estruturas usadas

Generic Structure



Generic Structure v2

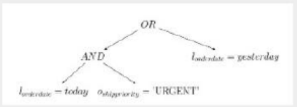


Generic Structure v3 (C++)

Ltree

Vector<String>

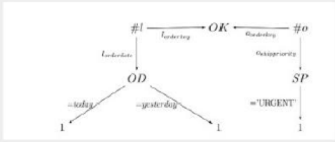
left child = 2*N+1
right child = 2*N+2
parent = (N-1)/2



Graph

Map<String, vector<vector<String>> >

aux:
Map<String, String> ← attribute table
Map<String,int> ← attribute type

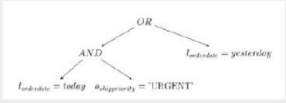


Generic Structure v4 (C++)

Ltree

Vector<String>

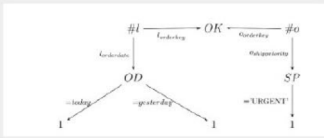
left child = 2*N+1
right child = 2*N+2
parent = (N-1)/2



Graph

Map<String, pair<String,String>> **join**;
vector<pair<String,String>> **groupby**;
Map<String, vector<String>> **filter**;
vector<String> **select**;

String **root**;
Map<String,Map<String,int>> **tables**;



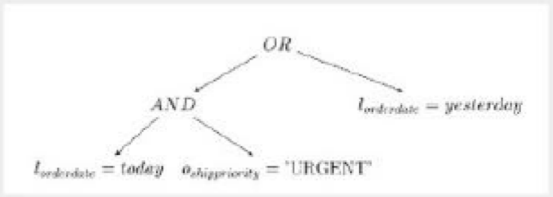
A estrutura final

Generic Structure v4 (C++)

Ltree

Vector<String>

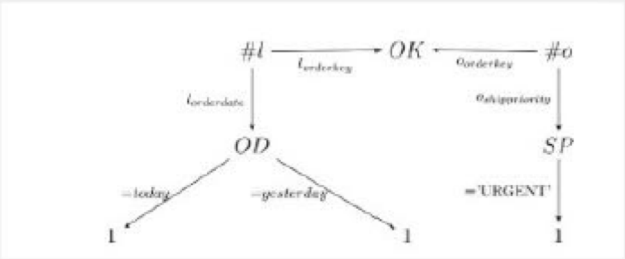
left child = 2*N+1
right child = 2*N+2
parent = (N-1)/2



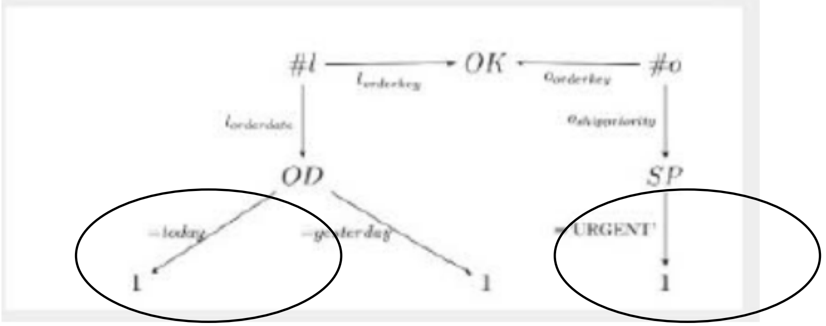
Graph

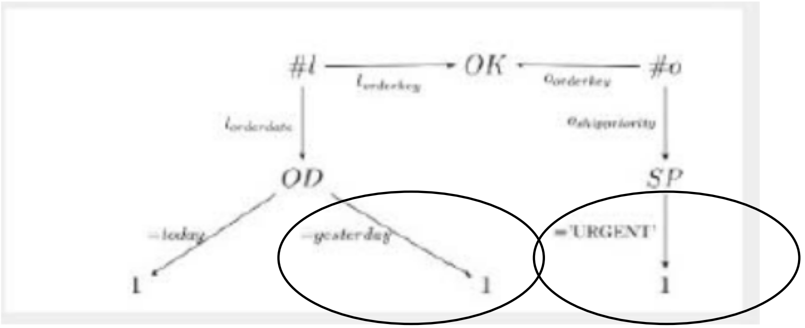
Map<String, pair<String,String>> **join**;
vector<pair<String,String>> **groupby**;
Map<String, vector<String>> **filter**;
vector<String> **select**;

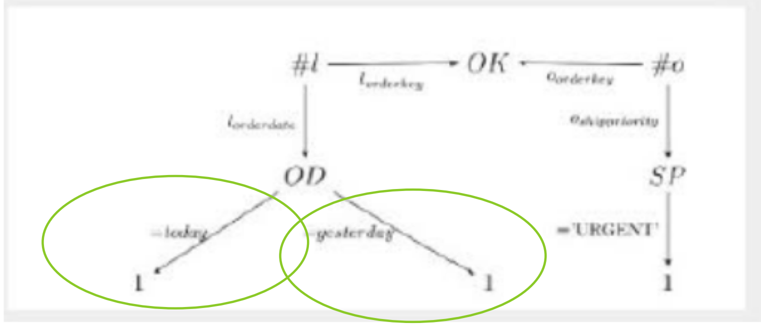
String **root**;
Map<String,Map<String,int>> **tables**;



Eficiência computacional







Testes e resultados

```sql
select
  sum(l_extendedprice * l_discount) as revenue
from
  lineitem_1
where
  l_shipdate >= '1994-01-01'
  and l_shipdate < '1994-01-01' + interval '1' year
  and l_discount between 0.06 - 0.01 and 0.06 + 0.01
  and l_quantity < 24;
```

Testes e resultados

```
select
  sum(l_extendedprice * l_discount) as revenue
from
  lineitem_1
where
  l_shipdate >= '1994-01-01'
  and l_shipdate < '1994-01-01' + interval '1' year
  and l_discount between 0.06 - 0.01 and 0.06 + 0.01
  and l_quantity < 24;
```

```
mainGraph.add_table("lineitem_1","lineitem_1.extendedprice","measure");
mainGraph.add_table("lineitem_1","lineitem_1.discount","measure");
mainGraph.add_table("lineitem_1","lineitem_1.shipdate","dimension");
mainGraph.add_table("lineitem_1","lineitem_1.quantity","measure");
mainGraph.add_select("sum(lineitem_1.extendedprice * lineitem_1.discount)","revenue");
mainGraph.newRoot("lineitem_1");
mainGraph.add_map_filter("lineitem_1.shipdate","lineitem_1.shipdate >= '1994-01-01'");
trees.push_back(create_tree("lineitem_1.shipdate >= '1994-01-01'","lineitem_1.shipdate"));
mainGraph.add_map_filter("lineitem_1.shipdate","lineitem_1.shipdate < '1994-01-01' + interval '1' year");
trees.push_back(create_tree("lineitem_1.shipdate < '1994-01-01' + interval '1' year","lineitem_1.shipdate"));
change_trees(join_trees(trees[0],trees[1],"AND"),0);
mainGraph.add_map_filter("lineitem_1.discount","lineitem_1.discount >= 0.06 - 0.01");
trees.push_back(create_tree("lineitem_1.discount >= 0.06 - 0.01","lineitem_1.discount"));
change_trees(join_trees(trees[0],trees[2],"AND"),0);
mainGraph.add_map_filter("lineitem_1.discount","lineitem_1.discount <= 0.06 + 0.01");
trees.push_back(create_tree("lineitem_1.discount <= 0.06 + 0.01","lineitem_1.discount"));
change_trees(join_trees(trees[0],trees[3],"AND"),0);
mainGraph.add_map_filter("lineitem_1.quantity","lineitem_1.quantity < 24");
trees.push_back(create_tree("lineitem_1.quantity < 24","lineitem_1.quantity"));
change_trees(join_trees(trees[0],trees[4],"AND"),0);
copy_tree(trees[0]);
print_tree();
resolve(0);
returnf();
```

Testes e resultados

```
select
  sum(l_extendedprice * l_discount) as revenue
from
  lineitem_1
where
  l_shipdate >= '1994-01-01'
  and l_shipdate < '1994-01-01' + interval '1' year
  and l_discount between 0.06 - 0.01 and 0.06 + 0.01
  and l_quantity < 24;
```

```
mainGraph.add_table("lineitem_1","lineitem_1.extendedprice","measure");
mainGraph.add_table("lineitem_1","lineitem_1.discount","measure");
mainGraph.add_table("lineitem_1","lineitem_1.shipdate","dimension");
mainGraph.add_table("lineitem_1","lineitem_1.quantity","measure");
mainGraph.add_select("sum(lineitem_1.extendedprice * lineitem_1.discount)","revenue");
mainGraph.newRoot("lineitem_1");
mainGraph.add_map_filter("lineitem_1.shipdate","lineitem_1.shipdate >= '1994-01-01'");
trees.push_back(create_tree("lineitem_1.shipdate >= '1994-01-01'","lineitem_1.shipdate"));
mainGraph.add_map_filter("lineitem_1.shipdate","lineitem_1.shipdate < '1994-01-01' + interval '1' year");
trees.push_back(create_tree("lineitem_1.shipdate < '1994-01-01' + interval '1' year","lineitem_1.shipdate"));
change_trees(join_trees(trees[0],trees[1],"AND"),0);
mainGraph.add_map_filter("lineitem_1.discount","lineitem_1.discount >= 0.06 - 0.01");
trees.push_back(create_tree("lineitem_1.discount >= 0.06 - 0.01","lineitem_1.discount"));
change_trees(join_trees(trees[0],trees[2],"AND"),0);
mainGraph.add_map_filter("lineitem_1.discount","lineitem_1.discount <= 0.06 + 0.01");
trees.push_back(create_tree("lineitem_1.discount <= 0.06 + 0.01","lineitem_1.discount"));
change_trees(join_trees(trees[0],trees[3],"AND"),0);
mainGraph.add_map_filter("lineitem_1.quantity","lineitem_1.quantity < 24");
trees.push_back(create_tree("lineitem_1.quantity < 24","lineitem_1.quantity"));
change_trees(join_trees(trees[0],trees[4],"AND"),0);
copy_tree(trees[0]);
print_tree();
resolve(0);
returnf();
```

```
A = filter(lineitem_1.shipdate >= '1994-01-01' AND lineitem_1.shipdate < '1994-01-01' + interval '1' year)
B = filter(lineitem_1.discount >= 0.06 - 0.01 AND lineitem_1.discount <= 0.06 + 0.01 AND lineitem_1.quantity < 24)
C = krao(B, A)
D =map(lineitem_1.extendedprice * lineitem_1.discount)
E = krao(D,C)
revenue = sum(E)
return(revenue)
```

Testes e resultados

```
select
  sum(l_extendedprice * l_discount) as revenue
from
  lineitem_1
where
  l_shipdate >= '1994-01-01'
  and l_shipdate < '1994-01-01' + interval '1' year
  and l_discount between 0.06 - 0.01 and 0.06 + 0.01
  and l_quantity < 24;
```

```
mainGraph.add_table("lineitem_l","lineitem_l.extendedprice","measure");
mainGraph.add_table("lineitem_l","lineitem_l.discount","measure");
mainGraph.add_table("lineitem_l","lineitem_l.shipdate","dimension");
mainGraph.add_table("lineitem_l","lineitem_l.quantity","measure");
mainGraph.add_select("sum(lineitem_l.extendedprice * lineitem_l.discount)","revenue");
mainGraph.newRoot("lineitem_l");
mainGraph.add_map_filter("lineitem_l.shipdate","lineitem_l.shipdate >= '1994-01-01'");
trees.push_back(create_tree("lineitem_l.shipdate >= '1994-01-01'","lineitem_l.shipdate"));
mainGraph.add_map_filter("lineitem_l.shipdate","lineitem_l.shipdate < '1994-01-01' + interval '1' year");
trees.push_back(create_tree("lineitem_l.shipdate < '1994-01-01' + interval '1' year","lineitem_l.shipdate"));
change_trees(join_trees(trees[0],trees[1],"AND"),0);
mainGraph.add_map_filter("lineitem_l.discount","lineitem_l.discount >= 0.06 - 0.01");
trees.push_back(create_tree("lineitem_l.discount >= 0.06 - 0.01","lineitem_l.discount"));
change_trees(join_trees(trees[0],trees[2],"AND"),0);
mainGraph.add_map_filter("lineitem_l.discount","lineitem_l.discount <= 0.06 + 0.01");
trees.push_back(create_tree("lineitem_l.discount <= 0.06 + 0.01","lineitem_l.discount"));
change_trees(join_trees(trees[0],trees[3],"AND"),0);
mainGraph.add_map_filter("lineitem_l.quantity","lineitem_l.quantity < 24");
trees.push_back(create_tree("lineitem_l.quantity < 24","lineitem_l.quantity"));
change_trees(join_trees(trees[0],trees[4],"AND"),0);
copy_tree(trees[0]);
print_tree();
resolve(0);
returnf();
```

```
A = filter(lineitem_l.shipdate >= '1994-01-01' AND lineitem_l.shipdate < '1994-01-01' + interval '1' year)
B = filter(lineitem_l.discount >= 0.06 - 0.01 AND lineitem_l.discount <= 0.06 + 0.01 AND lineitem_l.quantity < 24)
C = krao(B, A)
D =map(lineitem_l.extendedprice * lineitem_l.discount)
E = krao(D,C)
revenue = sum(E)
return(revenue)
```

Testes e resultados

```sql
select
    lineitem.orderkey,
    sum(extendedprice * (1 - discount)) as revenue,
    orderdate,
    shippriority
from
    customer,
    orders,
    lineitem
where
    mktsegment = 'BUILDING'
    and customer.custkey = orders.custkey
    and lineitem.orderkey = orders.orderkey
    and orderdate < date '1995-03-15'
    and shipdate > date '1995-03-15'
group by
    lineitem.orderkey,
    orderdate,
    shippriority;
```

Testes e resultados

```sql
select
    lineitem.orderkey,
    sum(extendedprice * (1 - discount)) as revenue,
    orderdate,
    shippriority
from
    customer,
    orders,
    lineitem
where
    mktsegment = 'BUILDING'
    and customer.custkey = orders.custkey
    and lineitem.orderkey = orders.orderkey
    and orderdate < date '1995-03-15'
    and shipdate > date '1995-03-15'
group by
    lineitem.orderkey,
    orderdate,
    shippriority;
```

```
A = filter(mktsegment = 'BUILDING')
B = dot(A, orders.o_custkey)
C = filter(orders.orderdate < date '1995-03-15')
D = krao(B, C)
E = krao(orders.orderdate,D)
F = krao(orders.shippriority,E)
G = dot(F, lineitem.orderkey)
H = filter(lineitem.shipdate > date '1995-03-15')
I = krao(G, H)
J = krao(lineitem.orderkey,I)
K =map(lineitem.extendedprice * (1 - lineitem.discount))
L = krao(K,J)
revenue = sum(L)
return(lineitem.orderkey, revenue, orders.orderdate, orders.shippriority);
```

Conclusão