6. Exercise

Timo Bergerbusch 344408
Thomas Näveke 311392
Shu Xing 381176

16.01.2018

Exercise 6.1

6.1.1

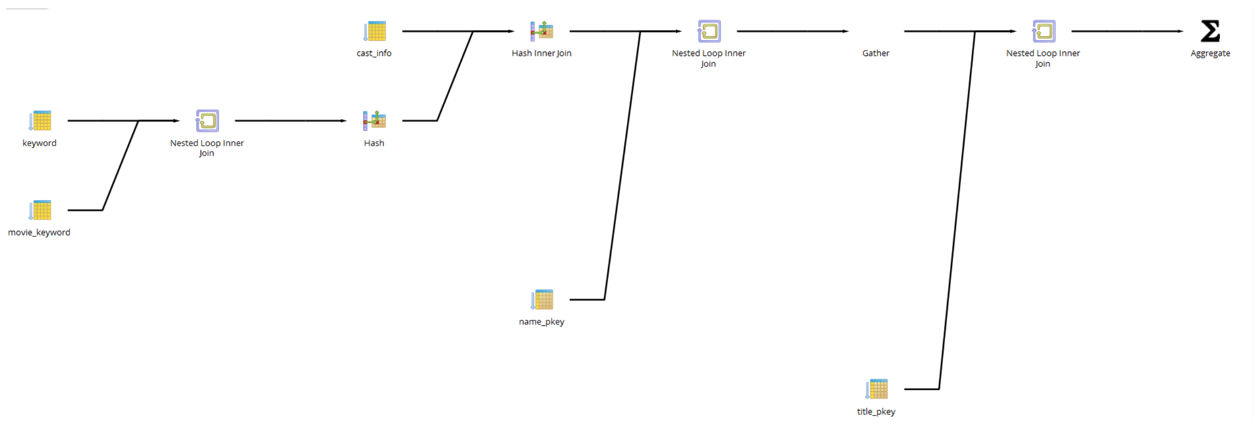
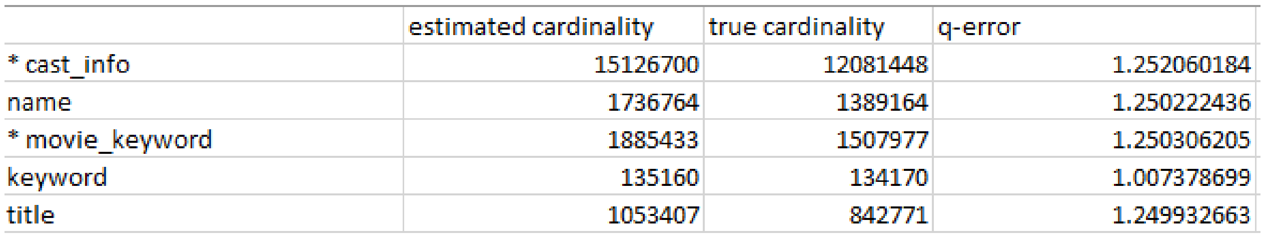


Figure 1: Execution Plan as calculated by PostgreSQL

The most expensive operation will be scanning through $cast\_info$ due to its sheer size.

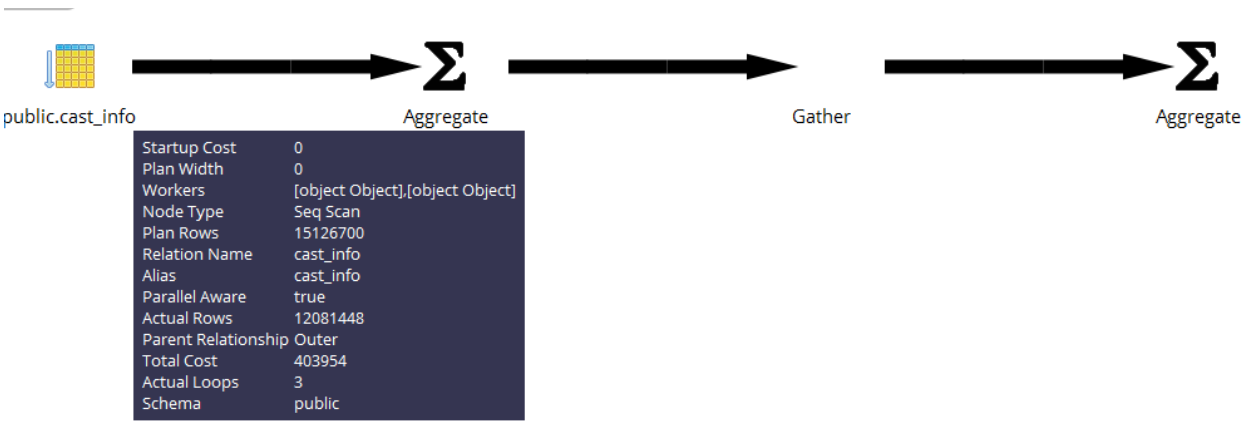
6.1.2

All cardinalities were determined with the query
"EXPLAIN (FORMAT JSON, ANALYZE ON, VERBOSE ON, COSTS ON,
BUFFERS OFF, TIMING OFF) SELECT COUNT(*) FROM table_name"

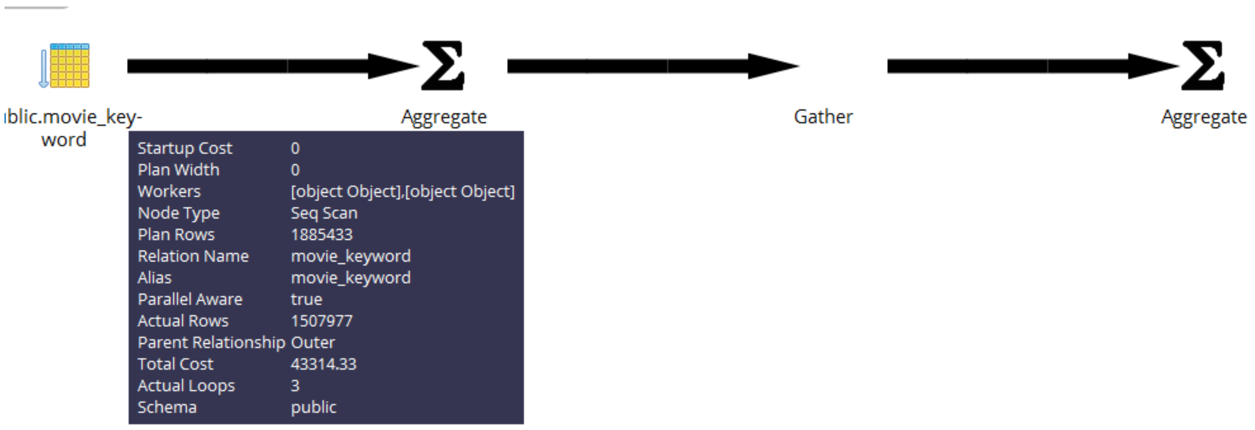


	estimated cardinality	true cardinality	q-error
* cast_info	15126700	12081448	1.252060184
name	1736764	1389164	1.250222436
* movie_keyword	1885433	1507977	1.250306205
keyword	135160	134170	1.007378699
title	1053407	842771	1.249932663

Figure 2: Cardinality and q-error



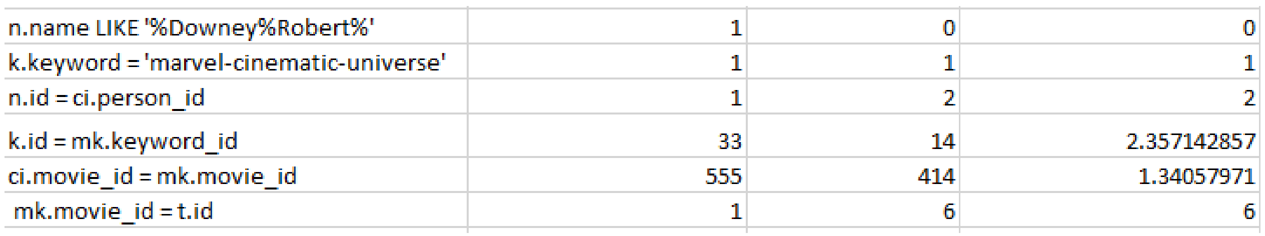
Startup Cost	0
Plan Width	0
Workers	[object Object],[object Object]
Node Type	Seq Scan
Plan Rows	15126700
Relation Name	cast_info
Alias	cast_info
Parallel Aware	true
Actual Rows	12081448
Parent Relationship	Outer
Total Cost	403954
Actual Loops	3
Schema	public



Startup Cost	0
Plan Width	0
Workers	[object Object],[object Object]
Node Type	Seq Scan
Plan Rows	1885433
Relation Name	movie_keyword
Alias	movie_keyword
Parallel Aware	true
Actual Rows	1507977
Parent Relationship	Outer
Total Cost	43314.33
Actual Loops	3
Schema	public

6.1.3

Since the proposed table differs from the query plan, we adjusted it to fit our query plan. For Screenshots see last page.



n.name LIKE '%Downey%Robert%'	1	0	0
k.keyword = 'marvel-cinematic-universe'	1	1	1
n.id = ci.person_id	1	2	2
k.id = mk.keyword_id	33	14	2.357142857
ci.movie_id = mk.movie_id	555	414	1.34057971
mk.movie_id = t.id	1	6	6

Figure 3: Cardinality and q-error

6.1.4

As we can read from the charts, the estimated and true cardinality often differ. This is due to the statistical sampling of several entries the optimizer uses to determine the estimated values. We could increase the accuracy by increasing the number of entries sampled. The closer we get to the number of entries, the closer the values will be to reality. However that makes the whole process of optimization very inefficient.

6.1.5

Exercise 6.2 (Datalog)

6.2.1

a) `parent(X,Y) :- child (Y,X).`

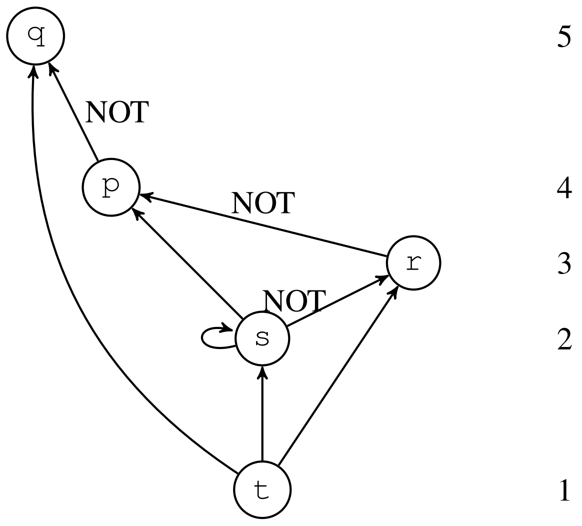
b) `married(X,Y) :- child(Z,X) , child(Z,Y).`

c) `sister(X,Y) :- child(X,Z), child(Y,Z), NOT male(X).`
Comment: assuming halfsisters are also considered sisters.

d) `halfbrother(X,Y) :- parent(Z,X), parent(Z,Y), parent(N,X), NOT parent(N,Y), male(X).`
Comment: assuming every person has only two parents.

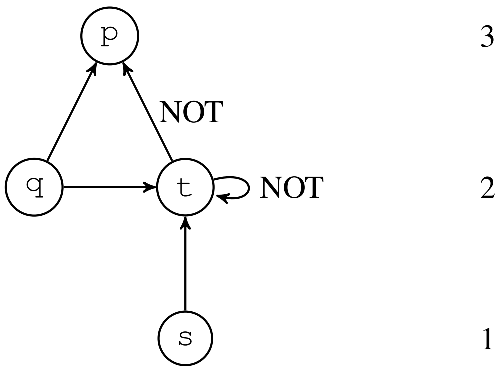
6.2.2

- Datalog program 1:

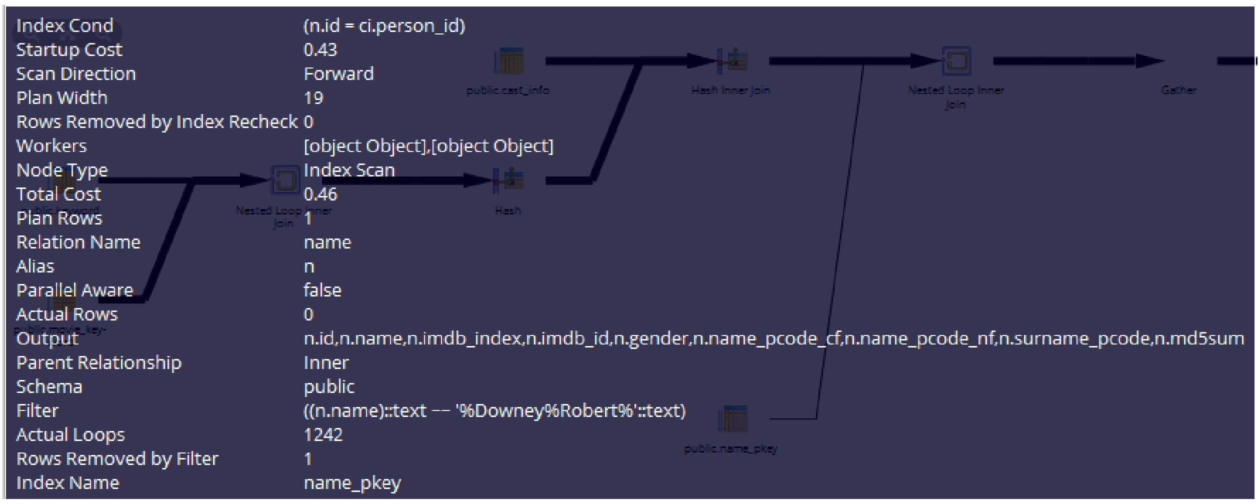


No two predicates in a layer depend negatively on each other, so the program is stratified.

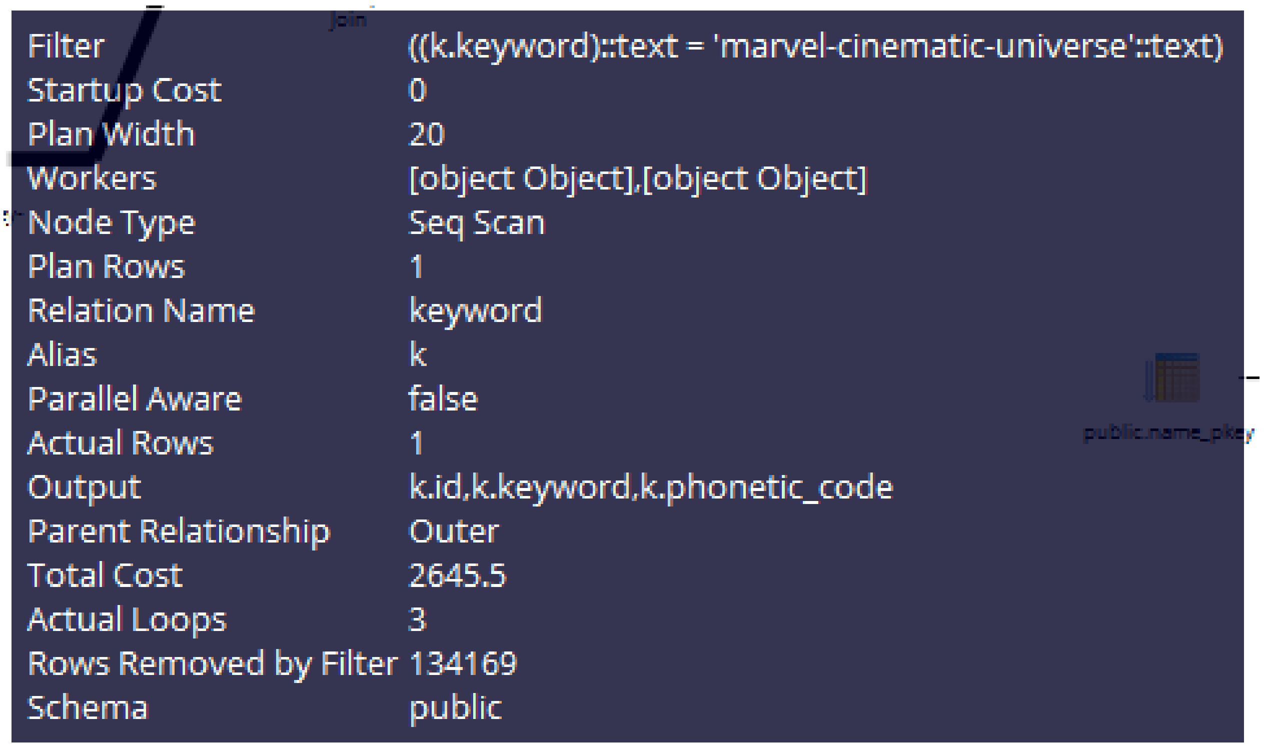
- Datalog program 2:



Since the predicate `t` depends negatively on itself program 2 is not stratified



Index Cond	(n.id = ci.person_id)
Startup Cost	0.43
Scan Direction	Forward
Plan Width	19
Rows Removed by Index Recheck	0
Workers	[object Object],[object Object]
Node Type	Index Scan
Total Cost	0.46
Plan Rows	1
Relation Name	name
Alias	n
Parallel Aware	false
Actual Rows	0
Output	n.id,n.name,n.imdb_index,n.imdb_id,n.gender,n.name_pcode_cf,n.name_pcode_nf,n.surname_pcode,n.md5sum
Parent Relationship	Inner
Schema	public
Filter	((n.name)::text ~~ '%Downey%Robert%'::text)
Actual Loops	1242
Rows Removed by Filter	1
Index Name	name_pkey



Filter	((k.keyword)::text = 'marvel-cinematic-universe'::text)
Startup Cost	0
Plan Width	20
Workers	[object Object],[object Object]
Node Type	Seq Scan
Plan Rows	1
Relation Name	keyword
Alias	k
Parallel Aware	false
Actual Rows	1
Output	k.id,k.keyword,k.phonetic_code
Parent Relationship	Outer
Total Cost	2645.5
Actual Loops	3
Rows Removed by Filter	134169
Schema	public