

# Implementation of Databases

## BONUS TASK

Due on January 31, 2017

*Prof. Dr. rer. pol. Matthias Jarke*

*Dr. rer. nat. Christoph Quix*

**Submitted by:**

Sanchit Alekh, Idil Esen Zulfikar

*MSc. Software Systems Engineering*

## Problem 1

Draw the estimated execution plan for the above query. Which one is the most expensive step?

## Solution

The Query Plan is illustrated by the following diagram:

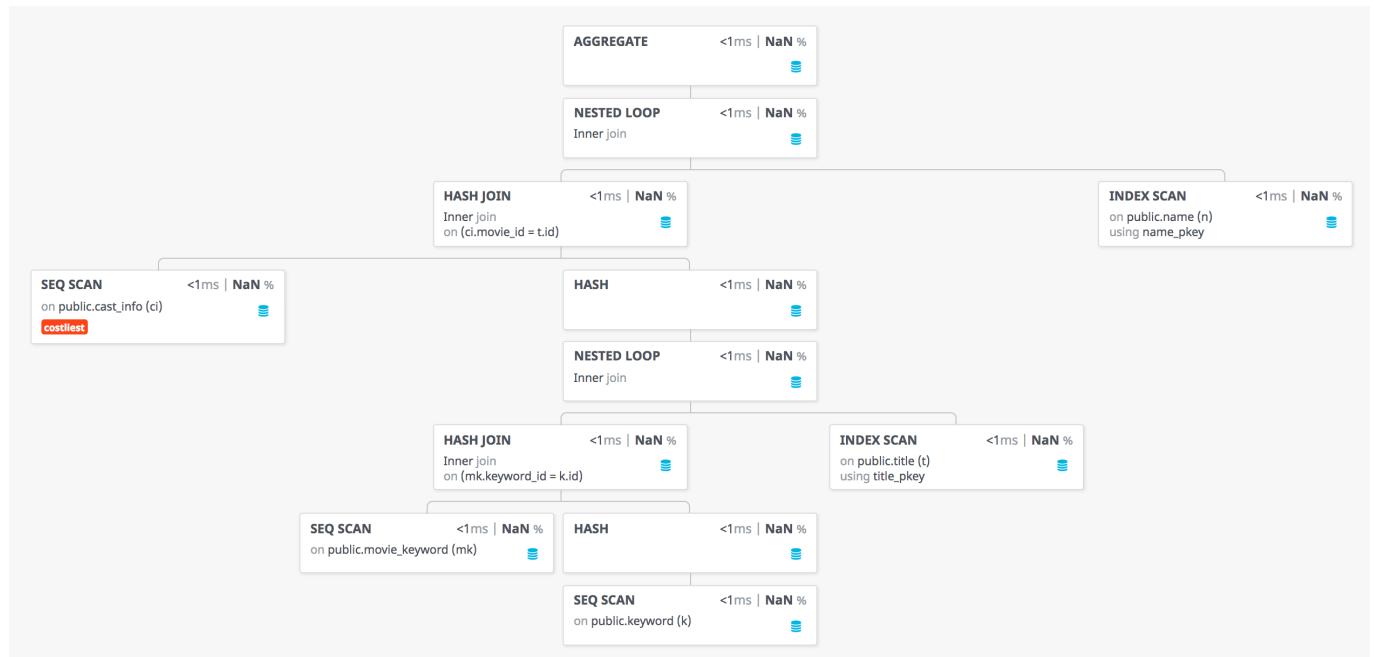


Figure 1: Query Plan for the Query

The most expensive step according to the **Explain** functionality of PostgreSQL is the **Sequential Scan on cast\_info**.

It has a cost of **615339.92**

Further details about the same is given in *Figure 2*.

## Problem 2

Fill in Table 2 with the estimated cardinality, true cardinality, and q-error of each base table of the given query.

## Solution

The filled up *Table 2* and the required screenshots are attached below:

SEQ SCAN

<1ms | NaN %

on public.cast\_info (ci)

costliest

Seq Scan Node

finds relevant records by sequentially scanning the input record set. When reading from a table, Seq Scans (unlike Index Scans) perform a single read operation (only the table is read).

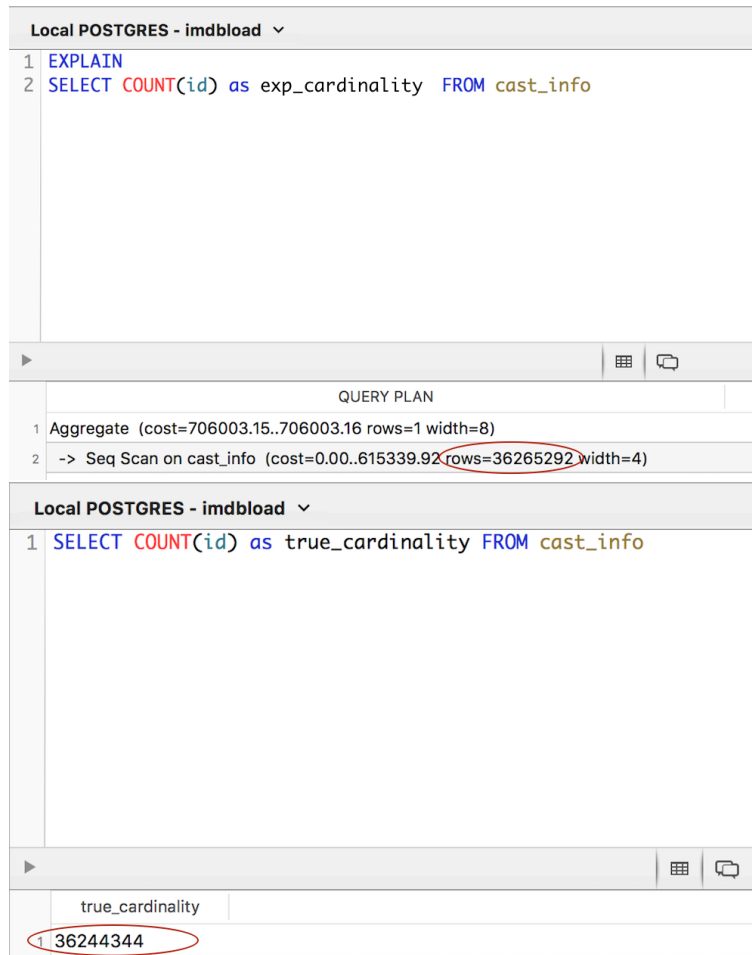
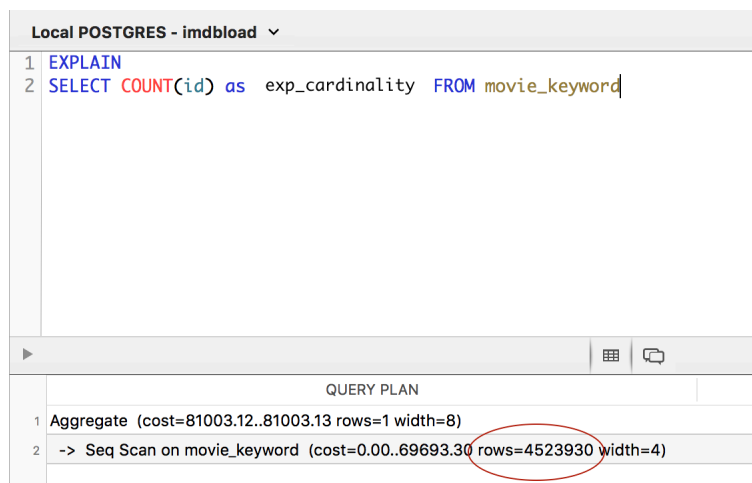
Startup Cost	0
Plan Width	8
Node Type	Seq Scan
Plan Rows	36265292
Relation Name	cast_info
Alias	ci
Parallel Aware	false
Output	ci.id,ci.person_id,ci.movie_id,ci.person_role_id,ci.note,ci.nr_order,ci.role_id
Parent Relationship	Outer
Total Cost	615339.92
Schema	public
*Planner Row Estimate Factor	
*Planner Row Estimate Direction	1
*Actual Duration	
*Actual Cost	615339.92
*Slowest Node (by duration)	false
*Largest Node (by rows)	false
*Costliest Node (by cost)	true

\*Pev calculated value

Figure 2: Details of the Most Expensive Step - Sequential Scan on *cast\_info*

	estimated cardinality	true cardinality	q-error
★ <i>cast_info</i>	36265292	36244344	1.00057
<i>name</i>	4168205	4167491	1.00017
★ <i>movie_keyword</i>	4523930	4523930	1.00
<i>keyword</i>	134170	134170	1.00
<i>title</i>	2528305	2528312	1.0000027

Figure 3: Calculation of Cardinalities and Error

Figure 4: Expected and True Cardinalities of *cast\_info*Figure 5: Expected Cardinality of *movie\_keyword*



The screenshot shows a PostgreSQL query window titled "Local POSTGRES - imdbload". The query entered is "SELECT COUNT(id) as true\_cardinality FROM movie\_keyword". The result is displayed in a table with one column, "true\_cardinality", and one row containing the value "4523930". The value "4523930" is circled in red.

	true_cardinality
1	4523930

Figure 6: True Cardinality of *movie\_keyword*

## Problem 3

Fill in Table 3 with the estimated cardinality  $c$ , true cardinality  $c_t$ , and q-error  $e_q$  of each intermediate steps (mainly joins) of the execution plan.

### Solution

The filled up Table 3 is given in Figure 7

	$c$	$c_t$	$e_q$
$\sigma_{n.name LIKE '%Downey\%Robert\%'}(name)$	414	2	207
$\sigma_{n.name LIKE '%Downey\%Robert\%'}(name) \bowtie cast\_info$	15013830888	72488688	207.12
$\sigma_{keyword='marvelcinematicuniverse'}(keyword)$	1	1	1
$\sigma_{keyword='marvelcinematicuniverse'}(keyword) \bowtie movie\_keyword$	4523930	4523930	1
$(\sigma_{n.name LIKE '%Downey\%Robert\%'}(name) \bowtie cast\_info)$ $\bowtie$ $(\sigma_{keyword='marvelcinematicuniverse'}(keyword) \bowtie movie\_keyword)$	could not execute completely in 120 mins	6792151996 9149840	-
$(\sigma_{n.name LIKE '%Downey\%Robert\%'}(name) \bowtie cast\_info)$ $\bowtie$ $(\sigma_{keyword='marvelcinematicuniverse'}(keyword) \bowtie movie\_keyword)$ $\bowtie$ $title$	could not execute completely in 120 mins	17172631854 56013886095 36	-

Figure 7: Calculation of Cardinalities and Error

## Problem 4

The database is heavily used with the query above and some additional queries. This generates a huge workload and the performance is going down. Your task as a database administrator is to increase the performance for the given workload.

### Solution

Following is the SQL script used to optimize the workload. The script is also provided as an SQL file.

```
CREATE NONCLUSTERED INDEX CAST_INFO_MOVIE_ID
ON cast_info (movie_id);

CREATE NONCLUSTERED INDEX CAST_INFO_PERSON_ID
ON cast_info (person_id);
```

```
CREATE NONCLUSTERED INDEX CAST_INFO_PERSON_ROLE_ID
    ON cast_info (person_role_id);

CREATE NONCLUSTERED INDEX CAST_INFO_ROLE_ID
    ON cast_info (role_id);

CREATE NONCLUSTERED INDEX COMP_CAST_TYPE_KIND
    ON comp_cast_type (kind);

CREATE NONCLUSTERED INDEX CHAR_NAME_NAME
    ON char_name (name);

CREATE NONCLUSTERED INDEX COMPANY_NAME_COUNTRY_CODE
    ON company_name (country_code);

CREATE NONCLUSTERED INDEX CAST_INFO_MOVIE_ID
    ON cast_info (movie_id);

CREATE NONCLUSTERED INDEX COMPLETE_CAST_MOVIE_ID
    ON complete_cast (movie_id);

CREATE NONCLUSTERED INDEX COMPLETE_CAST_SUBJECT_ID
    ON complete_cast (subject_id);

CREATE NONCLUSTERED INDEX COMPLETE_CAST_STATUS_ID
    ON complete_cast (status_id);

CREATE NONCLUSTERED INDEX INFO_TYPE_INFO
    ON info_type (info);

CREATE NONCLUSTERED INDEX KEYWORD_KEYWORD
    ON keyword (keyword);

CREATE NONCLUSTERED INDEX KEYWORD_PHONETIC_TYPE
    ON keyword (phonetic_type);

CREATE NONCLUSTERED INDEX KIND_TYPE_KIND
    ON kind_type (kind);

CREATE NONCLUSTERED INDEX LINK_TYPE_LINK
    ON link_type (link);
```

```
CREATE NONCLUSTERED MOVIE_COMPANIES_MOVIE_ID
    ON movie_companies (movie_id);

CREATE NONCLUSTERED MOVIE_COMPANIES_COMPANY_ID
    ON movie_companies (company_id);

CREATE NONCLUSTERED MOVIE_COMPANIES_COMPANY_TYPE_ID
    ON movie_companies (company_type_id);

CREATE NONCLUSTERED MOVIE_COMPANIES_NOTE
    ON movie_companies (note);

CREATE NONCLUSTERED MOVIE_INFO_MOVIE_ID
    ON movie_info (movie_id);

CREATE NONCLUSTERED MOVIE_INFO_INFO_TYPE_ID
    ON movie_info (info_type_id);

CREATE NONCLUSTERED MOVIE_INFO_INFO
    ON movie_info (info);

CREATE NONCLUSTERED MOVIE_INFO_NOTE
    ON movie_info (note);

CREATE NONCLUSTERED MOVIE_INFO_IDX_MOVIE_ID
    ON movie_info_idx (movie_id);

CREATE NONCLUSTERED MOVIE_INFO_IDX_INFO_TYPE_ID
    ON movie_info_idx (info_type_id);

CREATE NONCLUSTERED MOVIE_INFO_IDX_INFO
    ON movie_info_idx (info);

CREATE NONCLUSTERED MOVIE_INFO_IDX_NOTE
    ON movie_info_idx (note);

CREATE NONCLUSTERED MOVIE_KEYWORD_MOVIE_ID
    ON movie_keyword (movie_id);

CREATE NONCLUSTERED MOVIE_KEYWORD_KEYWORD_ID
```



```
    ON movie_keyword (keyword_id);

CREATE NONCLUSTERED MOVIE_LINK_MOVIE_ID
    ON movie_link (movie_id);

CREATE NONCLUSTERED MOVIE_LINK_LINKED_MOVIE_ID
    ON movie_link (linked_movie_id);

CREATE NONCLUSTERED MOVIE_LINK_LINK_TYPE_ID
    ON movie_link (link_type_id);

CREATE NONCLUSTERED NAME_NAME
    ON name (name);

CREATE NONCLUSTERED NAME_GENDER
    ON name (gender);

CREATE NONCLUSTERED NAME_NAME
    ON name (name);

CREATE NONCLUSTERED NAME_NAME_CODE_CF
    ON name (name_pcode_cf);

CREATE NONCLUSTERED NAME_NAME_CODE_NF
    ON name (name_pcode_nf);

CREATE NONCLUSTERED PERSON_INFO_PERSON_ID
    ON person_info (person_id);

CREATE NONCLUSTERED PERSON_INFO_INFO_TYPE_ID
    ON person_info (info_type_id);

CREATE NONCLUSTERED PERSON_INFO_INFO
    ON person_info (info);

CREATE NONCLUSTERED PERSON_INFO_NOTE
    ON person_info (note);

CREATE NONCLUSTERED ROLE_TYPE_ROLE
    ON role_type (role);
```

```
CREATE NONCLUSTERED PERSON_INFO_PERSON_ID  
ON person_info (person_id);
```

```
CREATE NONCLUSTERED TITLE_TITLE  
ON title (title);
```

```
CREATE NONCLUSTERED TITLE_SEASON_NR  
ON title (season_nr);
```

```
CREATE NONCLUSTERED TITLE_EPISODE_NR  
ON title (episode_nr);
```

```
CREATE NONCLUSTERED TITLE_PRODUCTION_YEAR  
ON title (production_year);
```