CSE 485/585 Mini-project #1: Logical Query Optimization

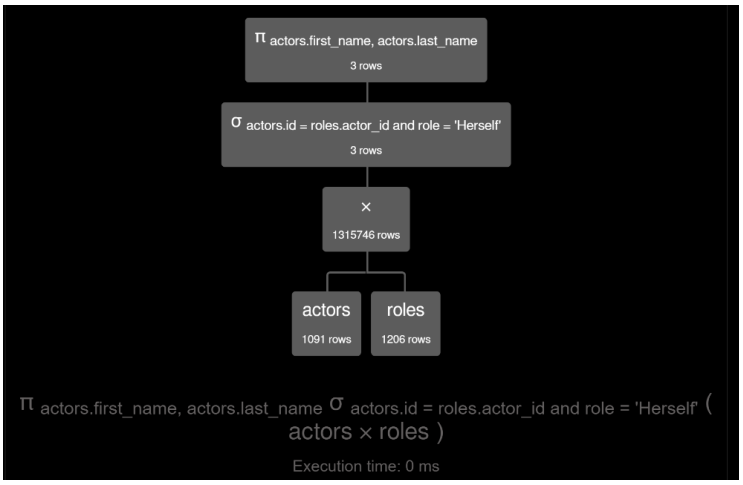
1. Create a shared Google Doc to do your work in.

2. Use the IMDB sample database (see the schema below for reference) in RelaX.

3. For each SQL query given below, create a canonical relational algebra tree in RelaX (directly executing the SQL may not work). Do not reorder the tables (use the same order in your RA query as shown in the SQL query).

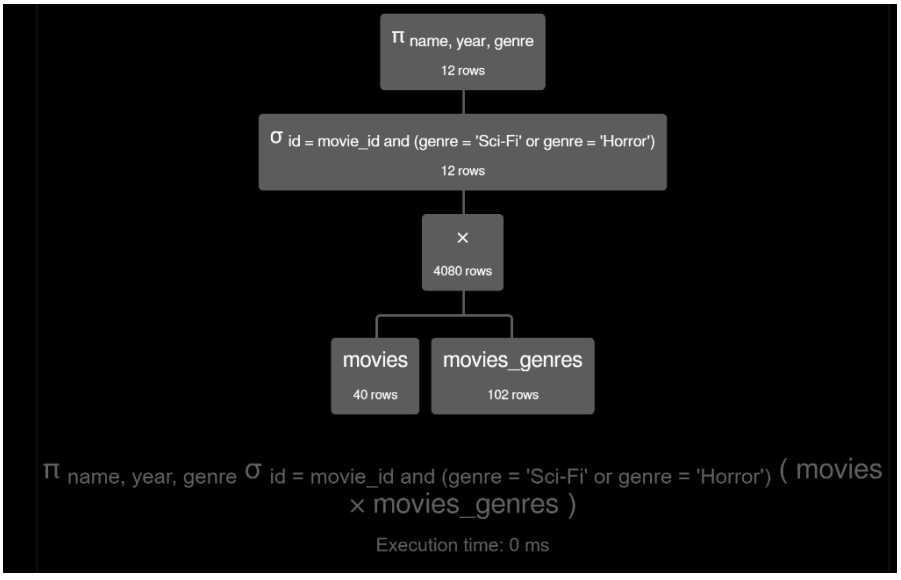
    - Give a screenshot of the RA tree and expression, unless otherwise indicated.
    - Compute the number of tuples represented by each node in the tree and give the final sum (cost of the whole tree) in the table below.

A.



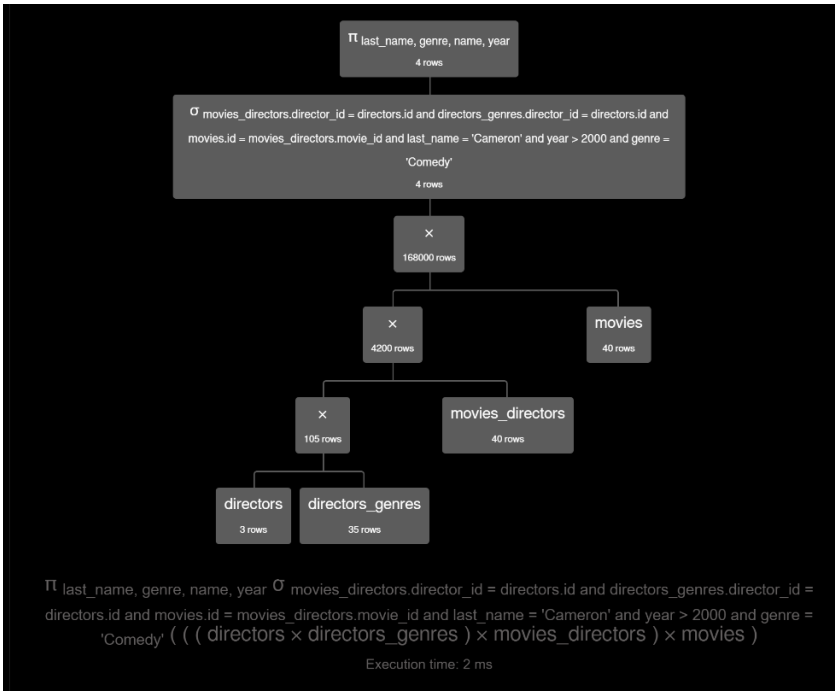
Cost = 1,318,049

B.



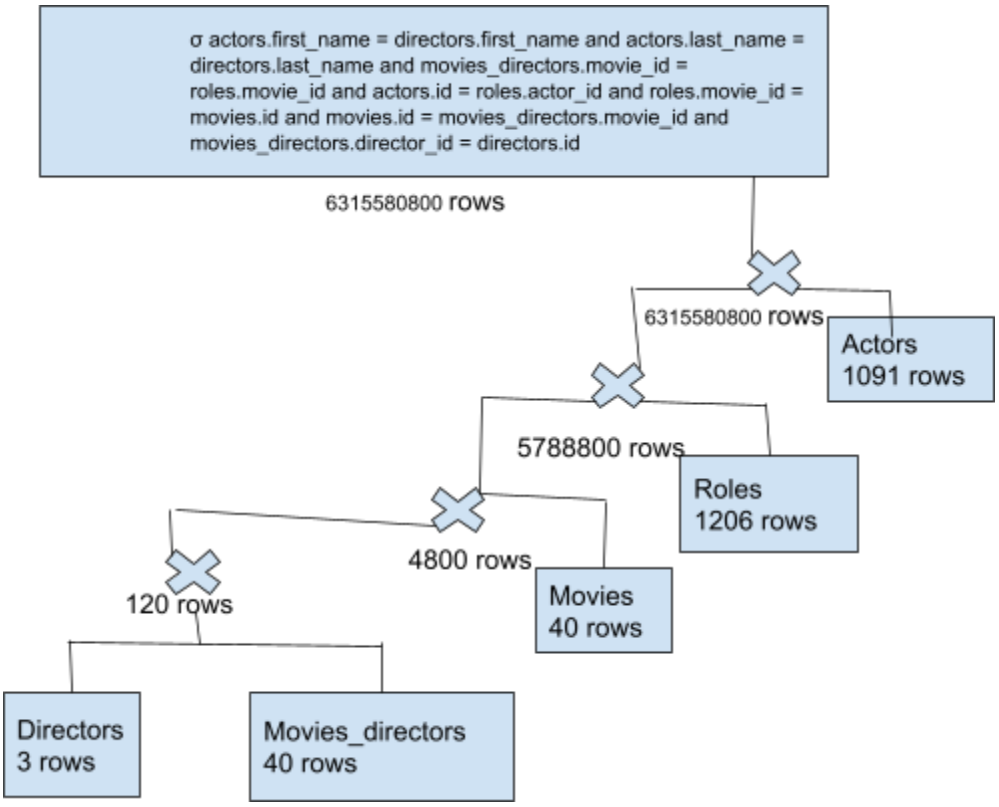
Cost = 4,246

C.



Cost = 175,391

D. .

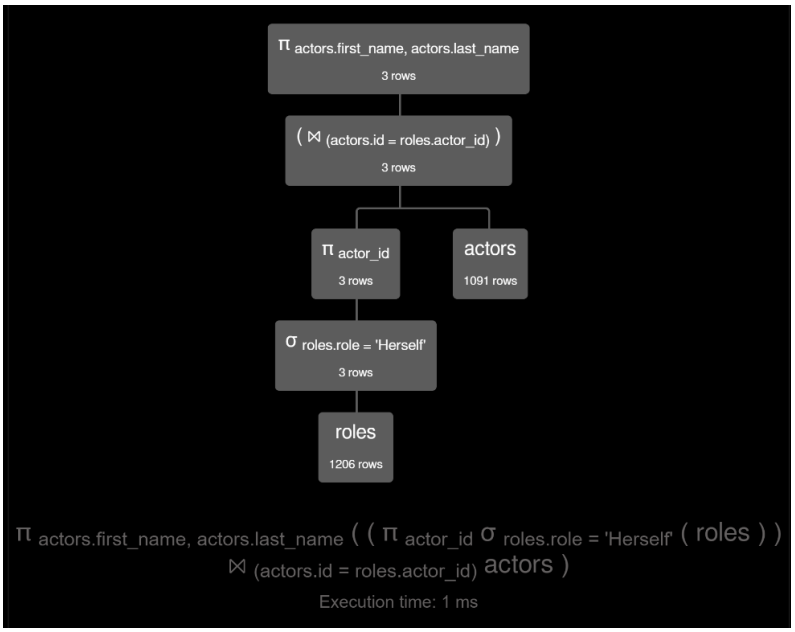


Cost= 12636957700

4. Apply the three optimization heuristics listed below <u>in order</u> and execute the final version of the RA tree in RelaX. Give the screenshot of the resulting RA tree and the tuples in the answer. Your optimized trees should be left-deep. Compute the total cost and fill in the table below.

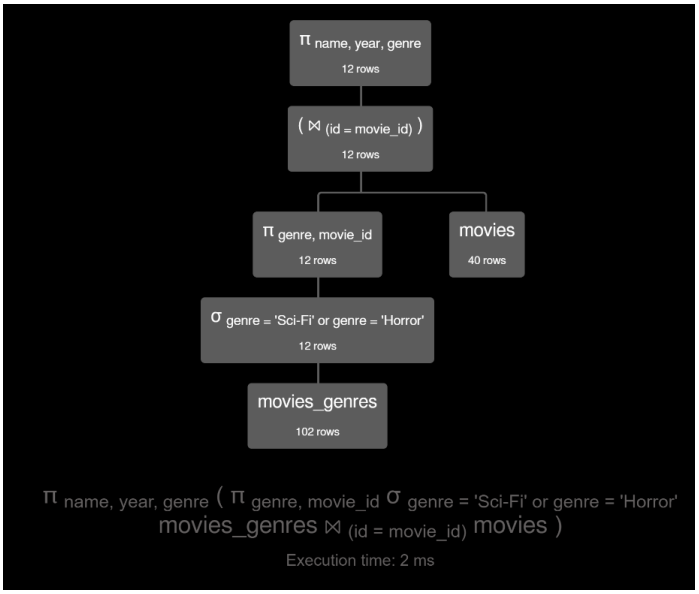
    a. push selects down the tree as far as possible,
    b. convert select and cross product to join where possible, and
    c. perform projection injection between a join and a selection where possible. If there is no selection below a join, skip the projection injection step.

A.



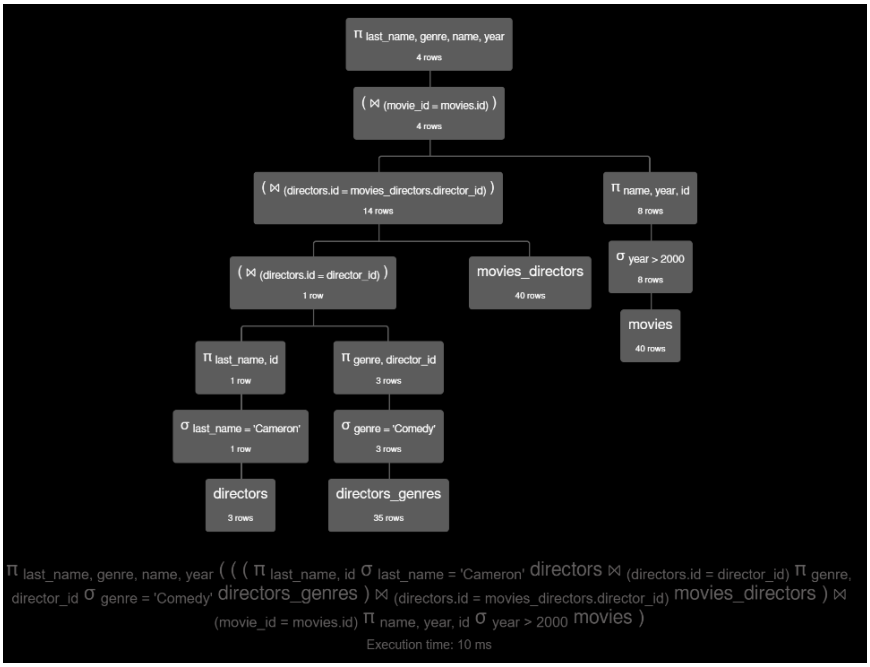
π actors.first_name, actors.last_name (
(π actor_id σ roles.role = 'Herself' (roles)) ⋈ (actors.id = roles.actor_id) actors)
Cost = 2,309

B.



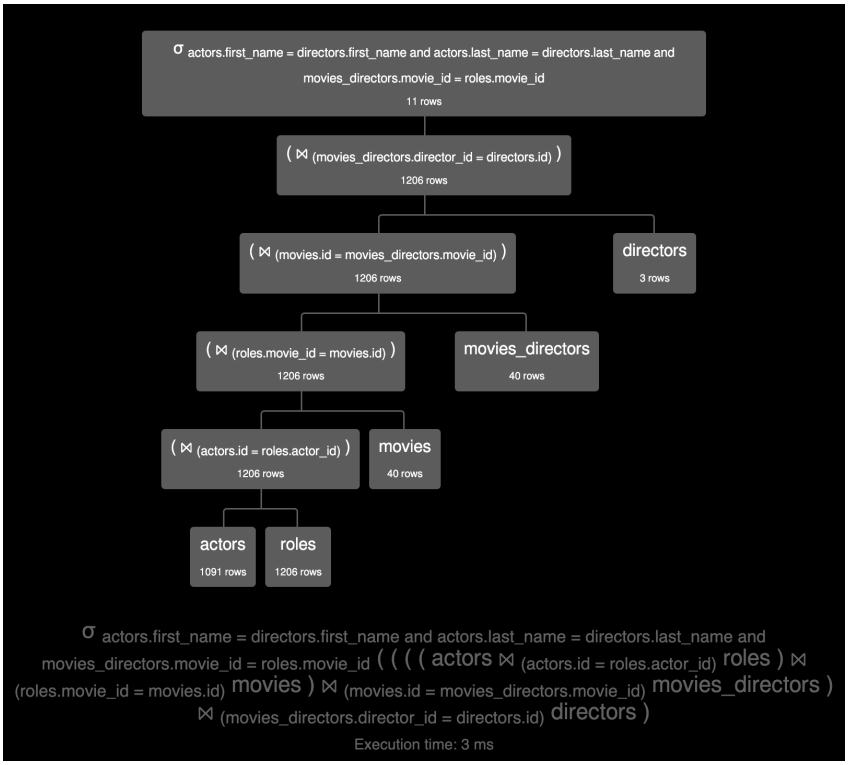
π name, year, genre
(π genre, movie_id σ genre = 'Sci-Fi' ∨ genre = 'Horror' movies_genres ⋈ (id = movie_id) movies)
Cost = 190

C.



π last_name, genre, name, year
(((π last_name, id σ last_name = 'Cameron' directors ⋈ (directors.id = director_id) π genre, director_id σ genre = 'Comedy' directors_genres) ⋈ (directors.id = movies_directors.director_id) movies_directors) ⋈ (movie_id = movies.id) π name, year, id σ year > 2000 movies)
Cost = 165

D.



σ actors.first_name = directors.first_name and actors.last_name = directors.last_name and movies_directors.movie_id = roles.movie_id
((((actors ⋈ (actors.id = roles.actor_id) roles)
⋈ (roles.movie_id = movies.id) movies)
⋈ (movies.id = movies_directors.movie_id) movies_directors)
⋈ (movies_directors.director_id = directors.id) directors)

5. Fill in the table for the total number of rows in each tree (canonical and fully optimized).

	A	B	C	D
canonical	1,318,049	4,246	172,391	12636957700
optimized	2,309	190	165	7,215

6. Include a screenshot of your timelog. It should show the total number of hours working together and total for each individual doing solo work. Don't double count the hours together, just sum it once.

names	date	start time	end time	duration	location	activity description	tasks for next time
Lauren Bockelman	2/20/23	1:30	2:15	0.75	Home	Reviewed project document. Created a canonical relational algebra tree in RelaX and apllied the three optimization heuristics to query A and calculated the cost of both trees.	Meet with John and continue working on the project.
Lauren Bockelman; John Doll	2/20/23	5:15	6:30	1.25	Benton	Together we created all the optimized RA trees by rewriting the SQL as relational algebra. Did the first three canonical trees and calculated the costs.	Draw the last canonical tree and calculate its cost.
Lauren Bockelman; John Doll	2/20/23	10:30	11:00	0.5	Laurens Home	We worked together to draw out and calculate the cost of the last canonical tree.	Finished.

Queries:

A. select distinct actors.first_name, actors.last_name
from actors, roles
where actors.id = roles.actor_id
and role = 'Herself'

B. select distinct name, year, genre
from movies, movie_genres
where id = movie_id
and (genre = 'Sci-Fi' or genre = 'Horror')

C. select last_name, genre, name, year
from directors, directors_genres, movies_directors, movies
where movies_directors.director_id = directors.id
and directors_genres.director_id = directors.id
and movies.id = movies_directors.movie_id
and last_name = 'Cameron'
and year > 2000
and genre = 'Comedy'

D. Give all the attributes for the table where the actor has the same name as a director (first and last) and the actor acted in a movie that the director directed.

Give the SQL, canonical RA tree, and optimized RA tree. You may not be able to execute the canonical RA tree, in which case you will have to draw it yourself and compute the cost. Note: nothing is projected, so there is no need for the projection injection step.

```
select *
from actors, directors, roles, movies_directors, movies
where  actors.first_name = directors.first_name
        and actors.last_name = directors.last_name
        and actors.id = roles.actor_id
        and roles.movie_id = movies.id
        and movies.id = movies_directors.movie_id
```

and movies_directors.movie_id = directors.id
and movies_directors.movie_id = roles.movie_id

Select DB (IMDB-sample)

actors
- id number
- first_name string
- last_name string
- gender string

directors
- id number
- first_name string
- last_name string

directors_genres
- director_id number
- genre string
- prob number

movies
- id number
- name string
- year number
- rank number

movies_directors
- director_id number
- movie_id number

movies_genres
- movie_id number
- genre string

roles