CT230 DATABASE SYSTEMS

PROBLEM SHEET 6

FOR LABS ON:

MON 7TH AND 14TH; TUE 8TH AND 15TH; THUR 10THAND 17TH NOVEMBER 2022

Learning Outcomes: To become familiar with Relational Algebra, Canonical Query Trees and Comparing efficiency of different trees.

Goal: This assignment involves a new database (imdb-sample) and the **querying of that database using the ReLaX calculator and relational algebra:**

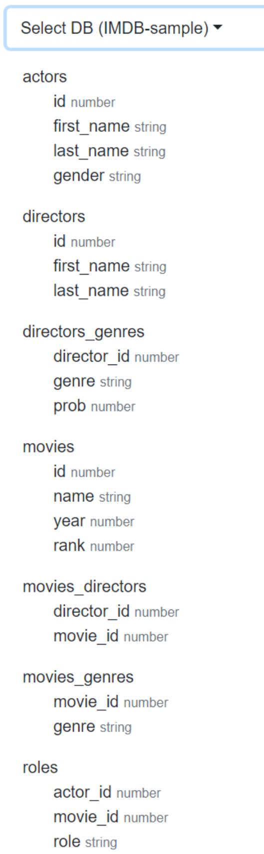
https://dbis-uibk.github.io/relax/calc/local/uibk/local/0

Examining & Marking: The material will be examined for marks via the final Blackboard quiz – Test 5 – which will be available on Friday 26th November.

SCHEMA:

The "imdb-sample" movie database, from the University of Saarland, comprises the following 7 tables which holds information on actors, movies and directors:

actors(id, first_name, last_name, gender)
directors(id, first_name, last_name)
directors_genres(director_id , genre, prob)
movies(id, name, year, rank)
movies_directors(director_id, movie_id)
movies_genres(movie_id, genre)
roles(actor_id, movie_id, role)



The table **movies** holds details on each movie (with id the primary key): the name, the year of release of the film, and the rank (similar to rating).
The **actors** table holds details on actors (with primary key id) and includes actor name and gender (single character).
The **directors** table holds details on directors (with primary key id) and includes the first name and surname of the director.

The **roles** table holds details on the actors in each movie and the tole they played; with movie_id and actor_id as the primary key. movie_id is a foreign key to id in the table movies. actor_id is a foreign key to id in the table actors.
The **movies_directors** table holds details on the directors of each movie with movie_id and director_id as the primary key. movie_id is a foreign key to id in the table movies. director_id is a foreign key to id in the table directors.
The **directors_genres** table holds details on the film genre usually associated with directors with director_id and genre as the primary key. Also stored is the probability that the director will direct a film of this genre. Note that one film can have a number of genres (e.g., drama, thriller).
movie_id is a foreign key to id in the table movies and director_id is a foreign key to id in the table directors. The **movies_genres** table holds details on the film genre with movie_id and genre as the primary key. Note that one film can have a number of genres (e.g., drama, thriller). movie_id is a foreign key to id in the table movies.

TASKS:

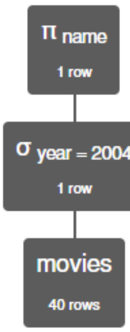
* Before you begin, load the dataset from the ReLaX calculator.

Week 10 lab: 7th, 8th and 10th November

Write relational algebra expressions to satisfy the following information needs, drawing/noting the query tree that represents your relational algebra expression:

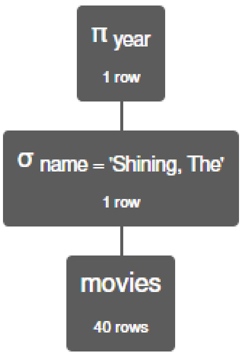
1. **List the names of all movies released in 2004.**

π name σ year = 2004 movies



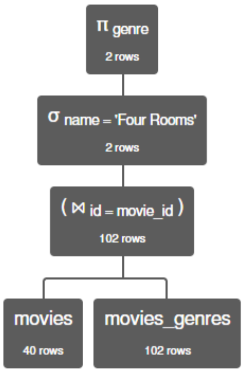
2. **find the release year of the movie with name ‘Shining, The’.**

π year σ name = 'Shining, The' movies



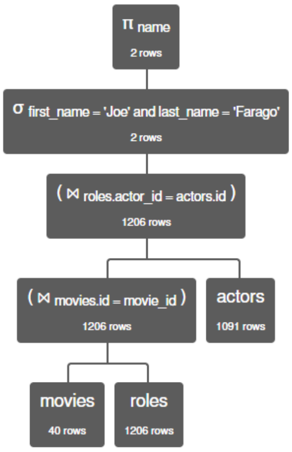
3. find the genres of the movie with the name ‘Four Rooms’.

π genre σ name = 'Four Rooms' (movies ⋈ id = movie_id movies_genres)



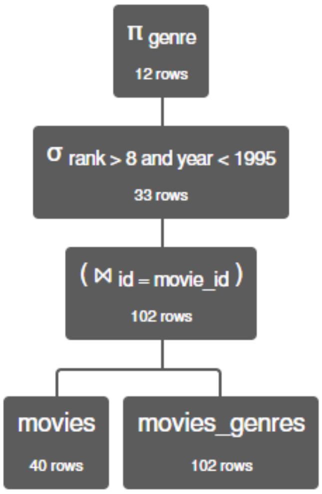
4. Find the names of movies that star the actor with name 'Joe Farago'.

π name σ first_name = 'Joe' AND last_name = 'Farago' ((movies ⋈ movies.id = movie_id roles) ⋈ roles.actor_id = actors.id actors)



5. List the genres of movies released before 1995 and with a ranking of more than 8.

π genre σ rank > 8 AND year < 1995 (movies ⋈ id = movie_id movies_genres)



Week 11 lab: 14th, 15th and 17th November

For each of the following: Write SQL statements to satisfy the following information needs, draw the canonical query tree (relational algebra) representation of your SQL solution and, using heuristic-based optimisation, produce an efficient relational algebra tree, writing out the relational algebra expression at the end.

(Note: The cartesian product won't work for large tables on the calculator as the temporary tables are too big)

6. Find the names and roles of actors in the film named 'Four Rooms'.

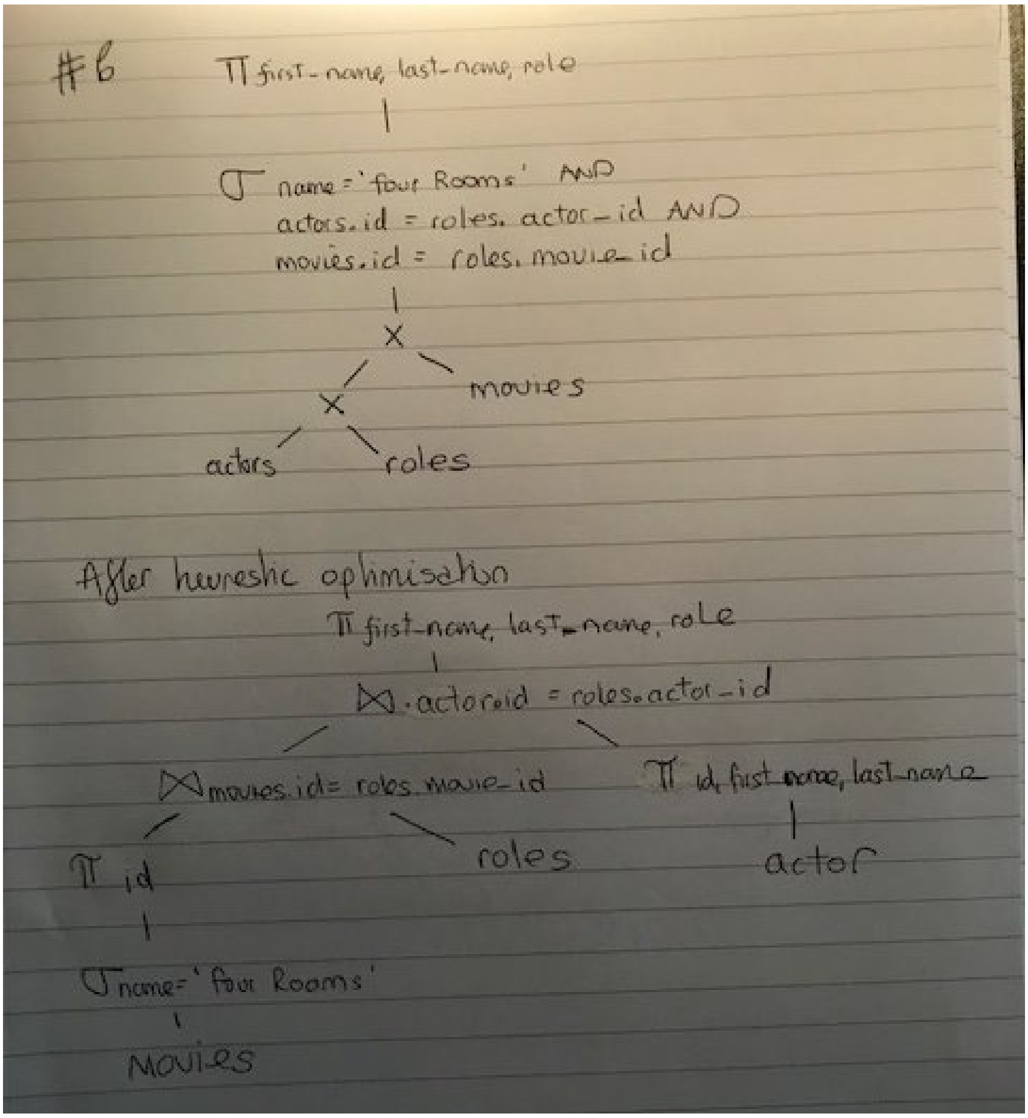
SELECT first_name, last_name, role

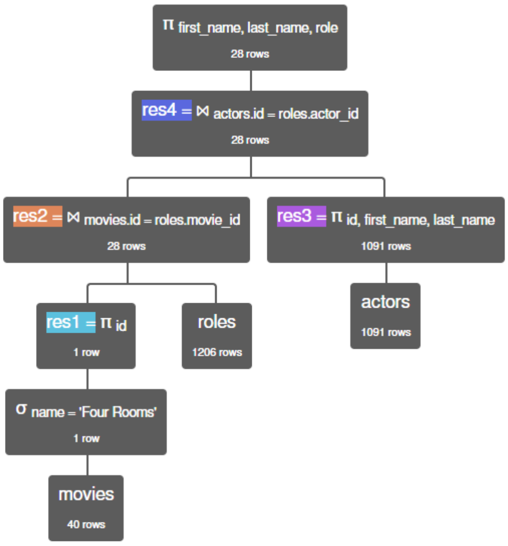
FROM   actors INNER JOIN roles ON actors.id = roles.actor_id

INNER JOIN movies ON movies.id = roles.movie_id

WHERE name = 'Four Rooms';

Canonical Query Tree and Efficient Query tree





$\pi_{\text{first\_name, last\_name, role}} \left(\left(\pi_{\text{id}}\ \sigma_{\text{name = 'Four Rooms'}}\ \text{movies} \bowtie_{\text{movies.id = roles.movie\_id}} \text{roles} \right) \bowtie_{\text{actors.id = roles.actor\_id}} \pi_{\text{id, first\_name, last\_name}}\ \text{actors} \right)$

7. Find the director of the movie with the name 'Four Rooms'.

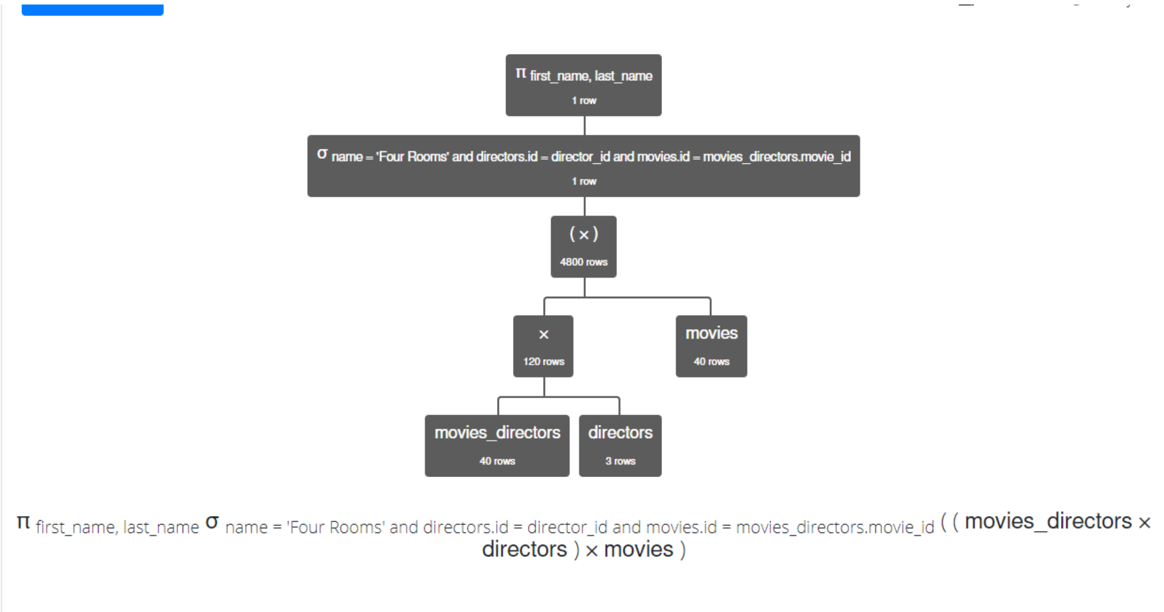
SELECT first_name, last_name

FROM movies_directors INNER JOIN directors ON id = director_id

INNER JOIN movies ON movies.id = movie_directors.movie_id

WHERE name = 'Four Rooms'

Canonical Query Tree:



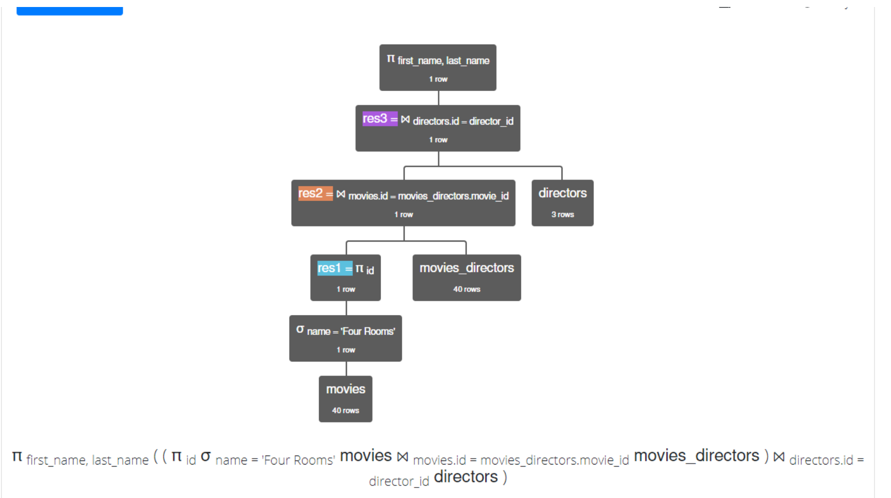
π first_name, last_name σ name = 'Four Rooms' AND directors.id = director_id AND movies.id = movies_directors.movie_id (movies_directors x directors x movies)

res1 = π id σ name = 'Four Rooms' movies

res2 = res1 ⋈ movies.id = movies_directors.movie_id movies_directors

res3 = res2 ⋈ directors.id = director_id directors

π first_name, last_name res3

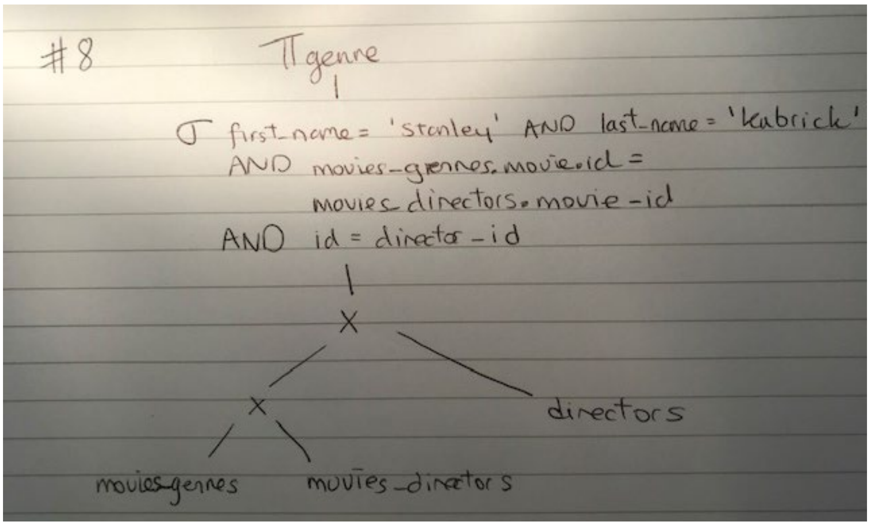


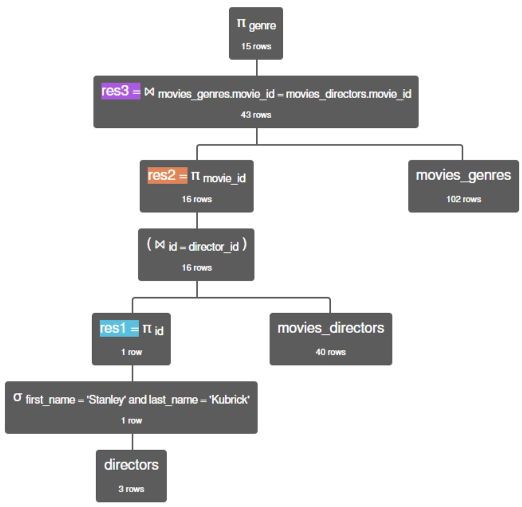
8. Find the genres of movies directed by the director 'Stanley Kubrick'.

SELECT genre

FROM movies_genres INNER JOIN movies_directors ON movies_genres.movie_id = movies_directors.movie_id INNER JOIN directors ON id = director_id

WHERE first_name = 'Stanley' AND last_name = 'Kubrick'





π_{genre} (π_{movie_id} (π_{id} $\sigma_{first\_name = 'Stanley' \text{ and } last\_name = 'Kubrick'}$ directors $\bowtie_{id = director\_id}$ movies_directors) $\bowtie_{movies\_genres.movie\_id = movies\_directors.movie\_id}$ movies_genres)

res1 = π id σ first_name = 'Stanley' AND last_name = 'Kubrick' directors

res2 = π movie_id (res1 ⋈ id = director_id movies_directors)

res3 = res2 ⋈ movies_genres.movie_id = movies_directors.movie_id movies_genres

π genre res3