L8: More on Joins

CS1106/CS6503: Intro to Relational Databases

Dr Kieran T. Herley

Semester One, 2023-24

School of Computer Science & Information Technology
University College Cork

Summary

More on joins.

Movies Database

cs1106 Goes to the Movies

Today we will get more practice with multi-table queris using the following simple film database

```
movies(id, title, yr, score, votes, director)
actors(id, name)
castings(movieid, actorid)
```

Movies and Actors tables

```
movies(id, title, yr, score, votes, director)
actors(id, name)
castings(movieid, actorid)
```

movies

- **id** unique id number for each movie
- **title** the name of the movie
- **yr** the year the movie was released
- **score** viewers rating (real number)
- **director** the name of the director

actors

- **id** unique id number for each actor
- **name** the actor's name

Castings Table

```
movies(id, title, yr, score, votes, director)
actors(id, name)
castings(movieid, actorid)
```

role

- "Bridges" movies and actors tables
- Models who appeared in what films

attributes

movieid id number of some movie

actorid id number of some actor

Signifies that the actor appeared in that movie

Target Practice

Query 1

Task List the maximum score obtained by any film(s) released during the 1960s.

Query 1

Task List the maximum score obtained by any film(s) released during the 1960s.

Solution

```
SELECT MAX(score)
FROM movies
WHERE yr BETWEEN 1960 AND 1969;
```

Query 2

Task List for each year the total number of films released that year and the maximum, minimum and average score obtained.

Query 2

Task List for each year the total number of films released that year and the maximum, minimum and average score obtained.

Solution

```
SELECT yr, COUNT(*), MIN(score), AVG(score), MAX(score)
FROM movies
GROUP BY yr;
```

Query 3

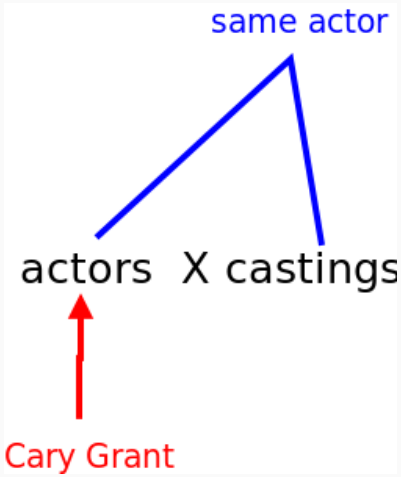
Task List the ids of all films starring Cary Grant.

Query 3

Task List the ids of all films starring Cary Grant.

Issues

- Need both `actors` and `castings` tables



Query 3 cont'd

Task List the ids of all films starring Cary Grant.

Solution

```
SELECT movieid
FROM
    actors JOIN castings
    ON id = actorid
WHERE name = 'Cary Grant';
```

Query 4

Task List the titles of all the films made by the director of "Vertigo".

Query 4

Task List the titles of all the films made by the director of "Vertigo".

Issues

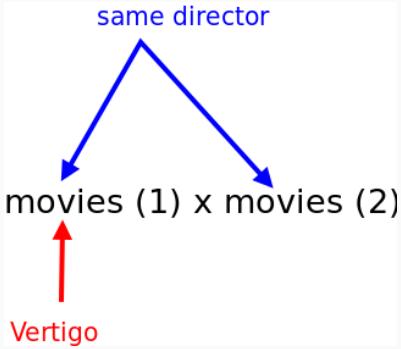
- Need pairs of movies: self-join of `movies` with itself

Query 4

Task List the titles of all the films made by the director of "Vertigo".

Issues

- Need pairs of movies: self-join of `movies` with itself



Query 4 cont'd

Task List the titles of all the films made by the director of "Vertigo".

Solution

```
SELECT m2.title
FROM
    movies AS m1
    JOIN movies as m2
    ON m1.director = m2.director
WHERE m1.title = 'Vertigo';
```

Query 5

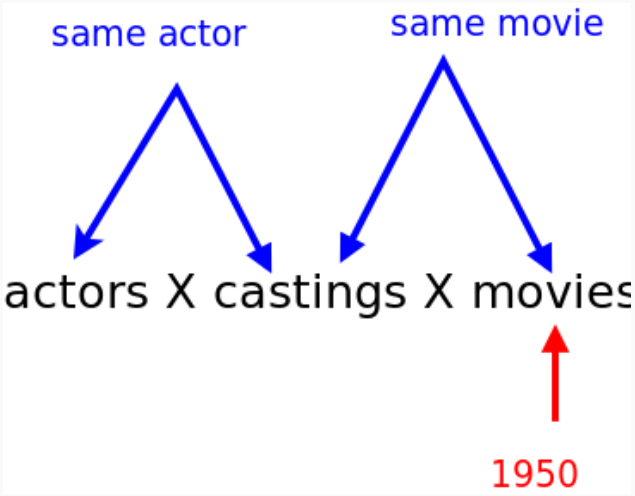
Task List alphabetically the names of all the actors who appeared in any film released in 1950.

Query 5

Task List alphabetically the names of all the actors who appeared in any film released in 1950.

Issues

- Need actor-casting-movies triples :



Query 5 cont'd

Task List alphabetically the names of all the actors who appeared in any film released in 1950.

Solution

```
SELECT actors.name, movies.title, movies.yr
FROM
    actors JOIN castings JOIN movies
    ON actors.id = castings.actorid
        AND castings.movieid = movies.id
WHERE movies.yr = 1950
ORDER BY actors.name;
```

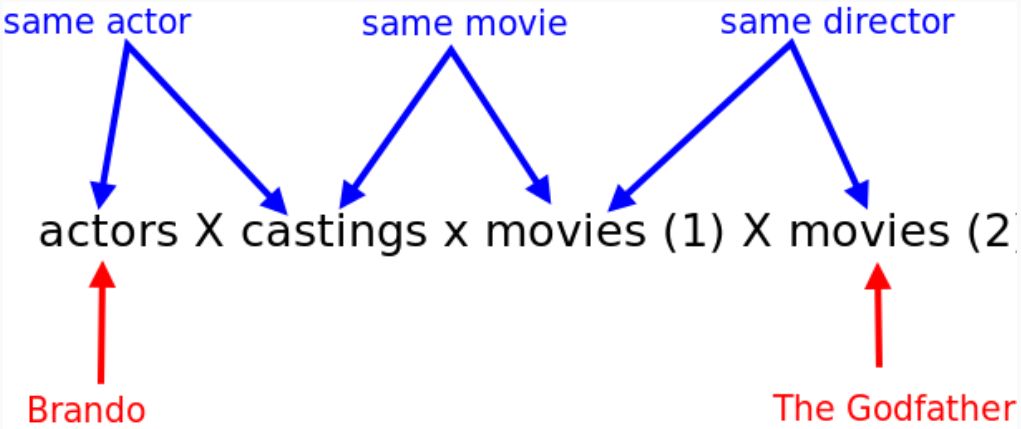
Query 6

Task List all the films in which Marlon Brando stars that were directed by the director of "The Godfather"; list the titles and years of the films concerned, arranged in increasing order by year.

Plan

- Can use (3-way) join to generate all appearances (actor-casting-movie) by Brando
- Use 4-way join ((actor-casting-movie)-movie) to generate appearance-movie combinations and filter for Coppola

Query 6 cont'd



Query 6 cont'd

Task List all the films in which Marlon Brando stars that were directed by the director of "The Godfather"; list the titles and years of the films concerned, arranged in increasing order by year.

Solution

```
SELECT m1.title, m1.yr
FROM
    actors AS a1
    JOIN castings AS c1
    JOIN movies AS m1
    JOIN movies AS m2
    ON a1.id = c1.actorid
        AND c1.movieid = m1.id
        AND m1.director = m2.director
WHERE
  m2.title = 'Godfather, The'
  AND a1.name = 'Marlon Brando'
ORDER BY m1.yr;
```

Query 7

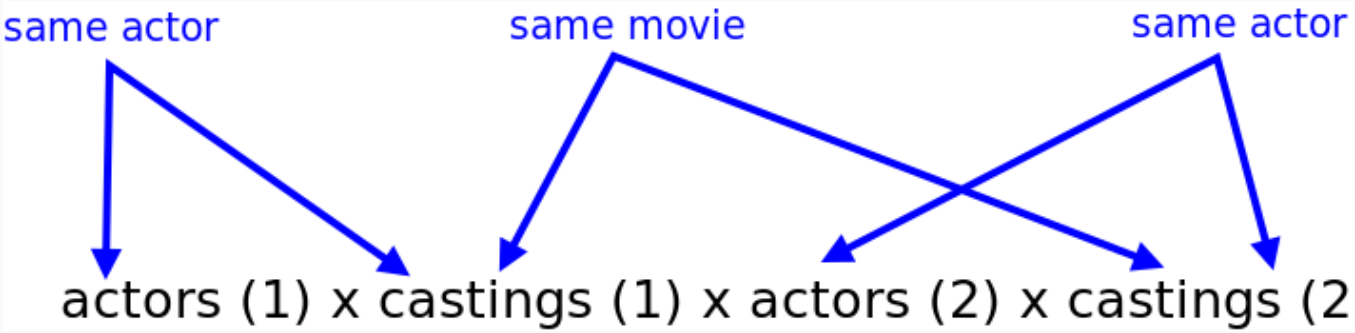
Task List the names of all pairs of actors that have appeared together in more than four films.

Issues

- Use join to generate all "co-appearances": pairs of actor-castings relating to same film
- Use grouping and aggregation on join table for counting

Query 7 cont'd

Task List the names of all pairs of actors that have appeared together in more than four films.



Query 7 cont'd

Task List the names of all pairs of actors that have appeared together in more than four films.

Solution

```
SELECT a1.name, a2.name, COUNT(*)
FROM
    actors  AS a1
    JOIN castings AS c1
    JOIN actors AS a2
    JOIN castings as c2
    ON a1.id = c1.actorid
        AND a2.id = c2.actorid
        AND a1.id < a2.id
        AND c1.movieid = c2.movieid
GROUP BY a1.name, a2.name
HAVING COUNT(*) > 4
ORDER BY COUNT(*) DESC;
```

Query 8

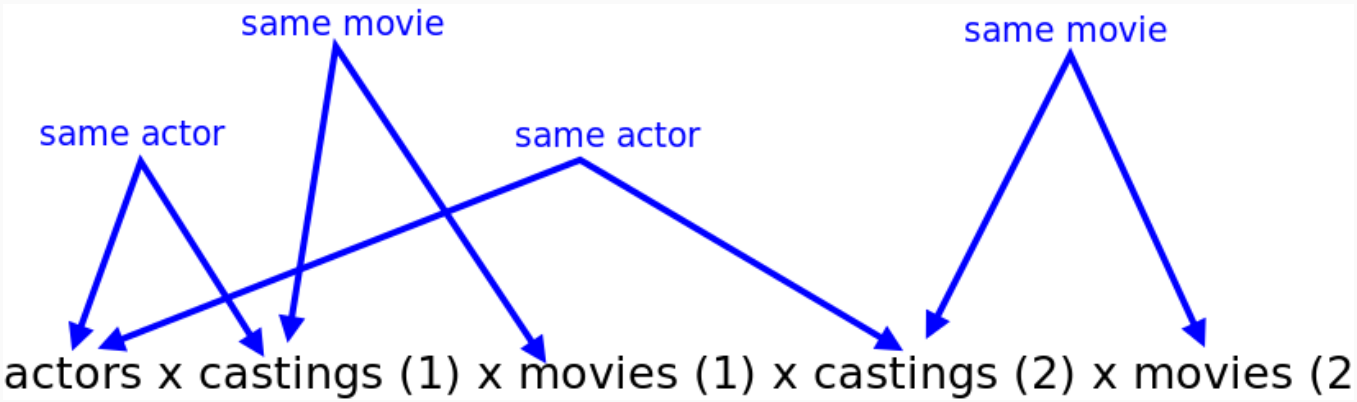
Task List alphabetically all the actors who have appeared in a film with a score below 3.0 and also in a film with a score above 8.5.

Issues

- Use join to genetate actor-appearance1-appearance2 tuples
- (where appearances1/appearances2 relate to actor)
- Filter to ensure appearance1 is a bad film (< 3.0) and
- Filter to ensure appearance2 is a good film (> 8.5) and

Query 8

List alphabetically all the actors who have appeared in a film with a score below 3.0 and also in a film with a score above 8.5.



Query 8 cont'd

Task List alphabetically all the actors who have appeared in a film with a score below 3.0 and also in a film with a score above 8.5.

Solution

```
SELECT DISTINCT a.name
    actors  AS a JOIN
    castings  AS c1 JOIN
    movies AS m1 JOIN
    castings  AS c2 JOIN
    movies AS m2 JOIN
    ON
        a.id  = c1.actorid  AND
        a.id  = c2.actorid  AND
        m1.id = c1.movieid AND
        m2.id = c2.movieid
WHERE
    m1.score < 3.0 AND m2.score > 8.5
ORDER BY a.name;
```

Notes and Acknowledgements

Reading

Code

Acknowledgements