L13: Correlated Subqueries

Add on Lecture 11

Dr Kieran T. Herley

Semester One, 2023-24

School of Computer Science & Information Technology
University College Cork

Summary

Correlated subqueries.

Correlated subqueries

“Normal” subquery

```
SELECT title
FROM movies
WHERE score = (SELECT MAX(score) FROM movies);
```

Inner query “independent” of outer.

Correlated subqueries

"Normal" subquery

```
SELECT title
FROM movies
WHERE score = (SELECT MAX(score) FROM movies);
```

Inner query "independent" of outer.

Correlated subquery

```
SELECT yr, title,  score
FROM movies as m1
WHERE score =
   (SELECT MAX(score) FROM movies AS m2
       WHERE m2.yr = m1.yr);
```

Why correlated? Inner refers to m1.yr, so inner and outer not independent.

Correlated subqueries cont'd

```
SELECT yr, title, score
FROM movies as m1
WHERE score =
   (/* Best score for year m1.yr */);
```

- Outer SELECT evaluates `score =(...)` for every row
- Inner `(...)` subquery re-executed sepaerately for each
- (Inverts inner-towards-outer reasoning of straightforward uncorrelated subqueries.)

Correlated subqueries cont'd

```
SELECT yr, title,  score
FROM movies as m1
WHERE score =
   (SELECT MAX(score) FROM movies AS m2
       WHERE m2.yr = m1.yr);
```

"Entanglement" of constituent subqueries make these harder to reason about.

Finding remakes

Task List all movie titles that occur more than once i.e. where there were one or more remakes.

Finding remakes

Task List all movie titles that occur more than once i.e. where there were one or more remakes.

Solution

```
SELECT DISTINCT m1.title
FROM movies AS m1
WHERE 2 <=
  (SELECT COUNT(*) from movies as m2
     WHERE m1.title = m2.title
  );
```

Why correlated? Inner refers to m1.title.

Finding remakes cont'd

Solution

```
SELECT DISTINCT m1.title
FROM movies AS m1
WHERE 2 <=
   ( /* Number of films with  title  m1.title  */);
```

Reasoning

- Outer query checks `2 <= ( . .. )` condition for each row
- Inner `( . .. )` re-executed for each

Earliest appearances

Task List for each actor the first movie they ever made.

Earliest appearances

Task List for each actor the first movie they ever made.

Solution

```
SELECT a1.name, m1.title, m1.yr
FROM actors AS a1 JOIN castings AS c1 JOIN movies AS m1
   ON a1.id = c1.actorid  AND c1.movieid = m1.id
WHERE m1.yr =
   (SELECT MIN(m2.yr)
       FROM castings as c2 JOIN movies AS m2
           ON c2.movieid = m2.id
       WHERE c2.actorid = a1.id
    );
```

Why correlated? Inner refers to a1.id.

Earliest appearances cont'd

Solution

```
SELECT a1.name, m1.title, m1.yr
FROM actors AS a1 JOIN castings AS c1 JOIN movies AS m1
   ON a1.id = c1.actorid  AND c1.movieid = m1.id
WHERE m1.yr
    = ( /* Min year among all appearances for  actor  a1.id */);
```

Reasoning

- Outer query considers all actor-movie appearances
- Condition `m1.yr = (...)` checked for each
- Inner `(...)` re-executed for each for relevant actor's id

Actors active in 1920s

Task List all the actors who made a film in the 1920s.

Actors active in 1920s

Task List all the actors who made a film in the 1920s.

Solution

```
SELECT name
FROM actors
WHERE EXISTS
    (SELECT movieid FROM castings
         WHERE actorid = id
     INTERSECT
     SELECT id FROM movies
         WHERE yr BETWEEN 1920 AND 1929
    );
```

Note: `EXISTS (...)` if True is subquery `(...)` returns one or more rows and False otherwise.

Inner subquery

```
(SELECT movieid FROM castings
     WHERE actorid = id
 INTERSECT
 SELECT id FROM movies
     WHERE yr BETWEEN 1920 AND 1929
);
```

Yields of all films made by actor with number `id` that were made during the 1920s.

Note:

- first reference to id, refers to id from actors (outer query)
- second reference to id refers to id from movies