COL 362 & COL 632

SQL – Subqueries & Recursion

25 Jan 2023

Announcements

- Assignment 1 will be released today evening
  - Deadline: **3rd February 2023 11:59pm (hard deadline)**
  - Lots of SQL writing
  - Large database and many relations
- Use PostgreSQL 8.4.22
- To be done individually, no collaborations
- If we spot “suspicious” activity, we will conduct a viva / quiz

- Project: a total of 49 teams have registered (**129 students**)
- Another form to fill soon – to select the project type and an initial proposal submission (**immediately after minor1**)

SQL - Subqueries

- SQL provides a mechanism for the nesting of subqueries. A **subquery** is a **select-from-where** expression that is nested within another query.
- The nesting can be done in the following SQL query

```
select A_1, A_2, ..., A_n
from r_1, r_2, ..., r_m
where P
```

as follows:

- **From clause:** r_i can be replaced by any valid subquery
- **Where clause:** P can be replaced with an expression of the form:

  `B <operation> (subquery)`

  B is an attribute and `<operation>` to be defined
- **Select clause:**

  A_i can be replaced be a subquery that generates a single value.

SQL – Subqueries

- Give all movies that are released in 2011 and have an actor who is older than 50 years
  - Intersect the set of movie titles from 2011 and set of movie titles which have an actor whose age > 50

```
Select Distinct Movies.Title
From Movies
Where Movies.Year = 2011 AND
Movies.Title in (Select Movies.Title
                 From Movies, Actors
                 Where Movies.Name = Actors.Name
                 And Actors.Age > 50;)
```

```
Select Distinct Movies.Title
From Movies, Actors
Where Movies.Year = 2011 AND
      Movies.ActorName = Actors.Name AND
      Actors.Age > 50;
```

Movies. Title = "Ragnarok"

in ("Ragnarok", "X-men", ...)

SQL - Subqueries

```
Movie (title, year, studioName, producerName)
StarsIn (movieTitle, movieYear, actorName)
MovieExec (name, netWorth)
```

Give the networth of all producers of movies in which “Benedict Cumberbatch” has starred in.

```
select sum(networth)
from Movie, StarsIn, MovieExec
where StarsIn.actorName = "Benny"
 and title = movieTitle
and producerName = name
~~Group By (ne~~
```

```
select netWorth
from MovieExec
where name in (
    select producerName distinct
    from Movie
    where (title, year) in (
      select movieTitle, movieYear
      from StarsIn
      where actorName = ‘Benedict Cumberbatch’
    )
 );
```

```
select netWorth
from MovieExec, Movie, StarsIn
where Movie.producerName = MovieExec.name
  and title = movieTitle
  and year = movieYear
  and actorName = ‘Benedict Cumberbatch’;
```

SQL – Subqueries

(Scoping rule)

- Find the average instructors' salaries of those departments where the average salary is greater than ~~$42,000~~ Rs. 42,000

***instructor*(ID, *name*, *dept_name*, *salary*)**

```
select dept_name, avg_salary
from ( select dept_name,
              avg (salary) as avg_salary
              from instructor
              group by dept_name)
where avg_salary > 42000;
```

as T

en (

(use T)

)

```
select dept_name, avg_salary
from ( select dept_name,
              avg (salary)
              from instructor
              group by dept_name )
as dept_avg (dept_name, avg_salary)
where avg_salary > 42000;
```

Correlated Subqueries

- Nested subquery evaluated many times, once for each assignment of value to some term in the subquery that comes outside

```
Movie (title, year, studioName, producerName)
```

for each movie find (if there is) a movie with same name but released **afterwards**.

```
select title
from Movie OLD
where year < ANY
      (    select year
           from Movie
           where title = Old.title
      );
```