Introduction to Data Management CSE 344

Section 3

Yi-Shu Wei

Subqueries

- A parenthesized SELECT-FROM-WHERE statement (*subquery*) can be used as a value or expression in another query
- Usage for subquries returning single or many values

  SELECT: Single Value

  FROM: Many Tuples

  WHERE: Many Values (IN, EXISTS, ANY, ALL) or
  Single Value (other operators)

Subqueries: Exercise

Consider a photo sharing Website with database schema

Users(uid, name)

Comment(uid, pid, score, txt) *score is an int between 1 and 10

Picture(pid, authorid, img)

A picture is **highly rated** if it received at least one score of 10 from a user other than its author.

A user is **cautious** if she commented only on highly rated pictures (including the one who did not comment at all.)

Write a SQL query that finds all cautious users.

Subqueries: Exercise

A database with the following schema stores a collection of Webpages and the words they contain, and a dictionary in several languages.

Occurs(url, word)

Dictionary(language, word)

Write a SQL query that computes, for each Webpage, the language with the largest number of words on that page.

More Exercises: Joins and Subqueries

Consider a social network database with the following schema:

Person(pid, name)

Relationship(pid1, pid2, type) *type is either 'enemy' or 'friend'

(a) A second degree friend is a friend's friend. Write a SQL query that computes for each person the total number of their second degree friends.

NOTE: Not every person has friends, but you have to count everyone's second degree friends.

More Exercises: Joins and Subqueries

Consider a social network database with the following schema:

Person(pid, name)

Relationship(pid1, pid2, type) *type is either 'enemy' or 'friend'

(b) Write a SQL query that returns all persons who have at least 12 common friends with 'Mary'. (The query finds all pair pid1, pid2 of two persons where the name of pid1 is 'Mary' and there are at least 12 persons p such that pid1 and p are friends, and pid2 and p are also friends.)

More Exercises: Joins and Subqueries

Consider a social network database with the following schema:

Person(pid, name)

Relationship(pid1, pid2, type) *type is either 'enemy' or 'friend'

(c) Fred says: “My enemies' enemies are my friends.”

Prove that Fred is wrong by writing a SQL query that returns all Fred's enemies' enemies that are not his friends.