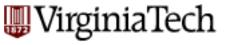


CS 4604: Introduction to Database Management Systems

B. Aditya Prakash

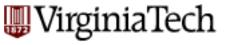
Lecture #4: Subqueries in SQL



Announcements

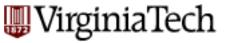
Project assignment 1 due today

- Homework 1 released today
 - due next Friday 2/8
 - SQL and Relational Algebra



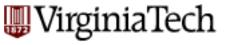


- Most popular embedded db in the world
 - Iphone (iOS), Android, Chrome....
- (Very) Easy to use: no need to set it up
- Self-contained: data+schema
- DB on your laptop: useful for testing, understanding....



Linear Notation for Relational Algebra

- Relational algebra expressions can become very long.
- Use linear notation to store results of intermediate expressions.
 - A relation name and a parenthesised list of attributes for that relation
 - Use Answer as the conventional name for the final result
 - The assignment symbol :=
 - Any expression in relational algebra on the right



Example of Linear Notation

- Name pairs of students who live at the same address.
- Normal expression:

```
\pi_{S1.Name,S2.Name}(
\sigma_{S1.Address=S2.Address}(\rho_{S1}(Students) \times \rho_{S2}(Students)))
```



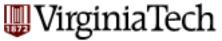
Example of Linear Notation

Normal expression:

```
\pi_{S1.Name,S2.Name}(
\sigma_{S1.Address=S2.Address}(\rho_{S1}(Students) \times \rho_{S2}(Students)))
```

Linear Notation:

```
Pairs(P1, N1, A1, P2, N2, A2) := \rho_{S1}(Students) \times \rho_{S2}(Students)
Matched(P1, N1, A1, P2, N2, A2) := \sigma_{A1=A2}(Pairs(P1, N1, A1, P2, N2, A2))
Answer(Name1, Name2) := \pi_{N1,N2}(Matched(P1, N1, A1, P2, N2, A2))
```



Interpreting Queries Involving Multiple Relations

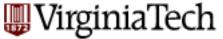
- SELECT A, B FROM R, S WHERE C;
- Nested loops:

for each tuple t1 in R

for each tuple t2 in S

if the attributes in t1 and t2 satisfy C

output the tuples involving attributes A and B



Interpreting Queries Involving Multiple Relations

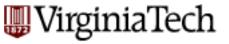
- SELECT A, B FROM R, S WHERE C;
- Conversion to relational algebra:

$$\pi_{A,B}(\sigma_C(R\times S))$$

Compute R X S

Apply selection operator $\sigma()$ to R X S

Project the result tuples to attributes A and B

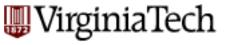


Motivation for Subqueries

Find the name of the professor who teaches "CS 4604."

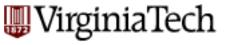
```
SELECT Name
FROM Professors, Teach
WHERE (PID = ProfessorPID) AND (Number = '4604') AND (DeptName = 'CS');
```

- Do we need to take the natural join of two big relations just to get a relation with one tuple?
- Can we rewrite the query without using a join?



Nesting

- A query can be put inside another query
- Most commonly in the WHERE clause
- Sometimes in the FROM clause (depending on the software)
- This subquery is executed first (if possible)

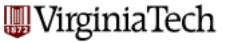


Subquery Example

Find the name of the professor who teaches "CS 4604."

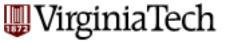
```
SELECT Name
FROM Professors
WHERE PID =
(SELECT ProfessorPID
FROM Teach
WHERE (Number = 4604) AND (DeptName = 'CS')
);
```

 When using =, the subquery must return a single tuple



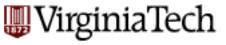
Conditions Involving Relations

- SQL includes a number of operators that apply to a relation and produce a boolean result.
- These operators are very useful to apply on results of sub-queries.



Conditions Involving Relations

- Let R be a relation and t be a tuple with the same set of attributes.
 - EXISTS R is true if and only if R contains at least one tuple.
 - t IN R is true if and only if t equals a tuple in R.
 - t > ALL R is true if and only if R is unary (has one attribute)
 and t is greater than every value in R.
 - Can use any of the other five comparison operators.
 - If we use <>, R need not be unary.
 - t > ANY R (which is unary) is true if and only if t is greater than at least one value in R.
- We can use NOT to negate EXISTS, ALL, and ANY.

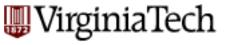


Subqueries Using Conditions

 Find the departments of the courses taken by the student with name 'Suri'.

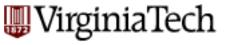
```
SELECT DeptName
FROM Take
WHERE StudentPID IN

( SELECT PID
FROM Students
WHERE (Name = 'Suri')
);
```



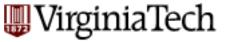
Correlated vs Uncorrelated

- The previous subqueries did not depend on anything outside the subquery
 - ...and thus need to be executed just once.
 - These are called <u>uncorrelated</u>.
- A <u>correlated</u> subquery depends on data from the outer query
 - ... and thus has to be executed for each row of the outer table(s)



Correlated Subqueries

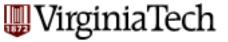
 Find course names that have been used for two or more courses.



Evaluating Correlated Subqueries

```
SELECT CourseName
FROM Courses AS First
WHERE CourseName IN
     (SELECT CourseName
     FROM Courses
     WHERE (Number <> First.Number)
     AND (DeptName <> First.DeptName)
    );
```

- Evaluate query by looping over tuples of First, and for each tuple evaluate the subquery.
- Scoping rules: an attribute in a subquery belongs to one of the tuple variables in that subquery's FROM clause, or to the immediately surrounding subquery, and so on.



Subqueries in FROM clauses

- Can use a subquery as a relation in a FROM clause.
- We must give such a relation an alias using the AS keyword.
- Let us find different ways of writing the query "Find the names of Professors who have taught the student whose first name is 'Suri'."
- The old way:

```
SELECT Professors.Name
```

FROM Professors, Take, Teach, Students

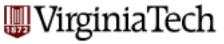
WHERE (Professors.PID = Teach.ProfessorPID)

AND (Teach.CourseNumber = Take.CourseNumber)

AND (Teach.DeptName = Take.DeptName)

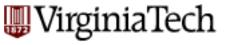
AND (Take.StudentPID = Student.PID)

AND (Student.Name = 'Suri %');



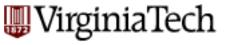
"Find the names of (Professors who have taught (courses taken by (student with first name 'Suri')))."

```
SELECT Name
FROM Professors
WHERE PID IN
   (SELECT ProfessorPID
     FROM Teach
     WHERE (Number, DeptName) IN
              SELECT Number, DeptName
              FROM Take, Students
              WHERE (StudentPID = PID) AND
                     (Students.Name = 'Suri%')));
```



Unrolling it further

```
SELECT Name
  FROM Professors
  WHERE PID IN
   (SELECT ProfessorPID
   FROM Teach
   WHERE (Number, DeptName) IN
     (SELECT Number, DeptName
     FROM Take
     WHERE StudentPID IN
        (SELECT PID
        FROM Students
        WHERE Name = 'Suri %')));
```



Aggregate Operators

- COUNT (*)
- COUNT ([DISTINCT] A)
 - A is a column
- SUM ([DISTINCT] A)
- AVG ([DISTINCT] A)
- MAX (A)
- MIN (A)
- Count the number of sailors
 SELECT COUNT (*)
 FROM Sailors S

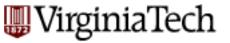


Find name and age of the oldest sailor(s)

SELECT S.sname, MAX (S.age)
FROM Sailors S

- This is illegal, but why?
 - Cannot combine a column with a value

SELECT S.sname, S.age FROM Sailors S WHERE S.age = (SELECT MAX (S2.age) FROM Sailors S2)



GROUP BY and HAVING

- So far, aggregate operators are applied to all (qualifying) tuples.
 - Can we apply them to each of several groups of tuples?
- Example: find the age of the youngest sailor for each rating level.
 - In general, we don't know how many rating levels exist, and what the rating values for these levels are!
 - Suppose we know that rating values go from 1 to 10; we can write 10 queries that look like this:

For
$$i = 1, 2, ..., 10$$
:

SELECT MIN (S.age) FROM Sailors S WHERE S.rating = i

VirginiaTech Find the age of the youngest sailor for each rating level

SELECT S.rating, MIN (S.age) as age
FROM Sailors S
GROUP BY S.rating

(1) The sailors tuples are put into "same rating" groups.

(2) Compute the Minimum age for each

rating group.

Prakash 2013

32 95	Andy Bob
ating	Age
	25.5
	63.5
	45.0
	55.5
	25.5

Sid

22

31

85

Sname

Dustin

Lubber

Art

Rating

8

Age

45.0

55.5

25.5

25.5

63.5

24

Rating	Age	——
3	25.5	(2)
7	45.0	
8	25.5	VT CS 4604

WirginiaTech

Find the age of the youngest sailor for each rating level that has at least 2 members

		Sid	Sname	Rating	Age
	SELECT S.rating, MIN (S.age) as minage	22	Dustin	7	45.0
	FROM Sailors S	31	Lubber	8	55.5
	GROUP BY S.rating	<i>85</i>	Art	3	25.5
	HAVING COUNT(*) > 1	<i>32</i>	Andy	8	25.5
. •	The sailors tuples are put into	95	Bob	3	63.5

2. Eliminate groups that have < 2 members.

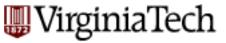
"same rating" groups.

3. Compute the Minimum age for each rating group.

3	ige for		
	Rating	Minage	
	3	25.5	
	8 T CS 4604	25.5	

_	
Rating	Age
3	25.5
3	63.5
7	45.0
8	55.5
8	25.5

Prakash 2013



Queries With GROUP BY and HAVING

SELECT [DISTINCT] target-list

FROM relation-list

WHERE qualification

GROUP BY grouping-list

HAVING group-qualification

SELECT S.rating, MIN (S.age) as age

FROM Sailors S

GROUP BY S.rating

HAVING S.rating > 5

- The target-list contains (i) attribute names (ii) terms with aggregate operations (e.g., AVG (S.age)).
- The attribute list (e.g., S.rating) in target-list must be in grouping-list.
- The attributes in group-qualification must be in grouping-list.