DM505 Database Design and Programming
DM576 Database Systems

Panagiotis Tampakis

ptampakis@imada.sdu.dk

Subqueries

- A query that is part of another query.
- There are several ways that subqueries can be used:
  - Subqueries can **appear in FROM clauses**, followed by a tuple.
  - Subqueries can return **a single constant**, which can be compared to another value in a WHERE clause.
  - Subqueries can return **relations**, which can be compared to another value in WHERE clauses.

Subqueries That Return One Tuple

- If a subquery is guaranteed to produce one tuple, then the subquery can be used as a value
  - Usually, the tuple has one component
  - A run-time error occurs if there is no tuple or more than one tuple

Example: Single-Tuple Subquery

- Using Sells(<u>bar</u>, <u>beer</u>, price), find the bars that serve Pilsener for the same price Cafe Chino charges for Od.Cl.
- Two queries would surely work:
  1. Find the price Cafe Chino charges for Od.Cl.
  2. Find the bars that serve Pilsener at that price

Query + Subquery Solution

```
SELECT bar
FROM Sells
WHERE beer = ’Pilsener’ AND
    price = (SELECT price
             FROM Sells
             WHERE bar = ’Cafe Chino’
               AND beer = ’Od.Cl.’);
```

The price at Which C.Ch. sells Od.Cl.

Subqueries That Return Relations

- There are a number of SQL operators that we can apply to a relation *R* and produce a boolean result.
  - IN
  - EXISTS
  - ALL
  - ANY

The IN Operator

- <tuple> IN (<subquery>) is true if and only if the tuple is a member of the relation produced by the subquery
  - Opposite: <tuple> NOT IN (<subquery>)
- IN-expressions can appear in WHERE clauses

Example: IN

- Using Beers(name, manf) and Likes(drinker, beer), find the name and manufacturer of each beer that Peter likes

```
SELECT *
FROM Beers
WHERE name IN (SELECT beer
               FROM Likes
               WHERE drinker= ’Peter’);
```

The set of Beers Peter likes

What is the difference?

```
R(a,b); S(b,c)

SELECT a
FROM R, S
WHERE R.b = S.b;

SELECT a
FROM R
WHERE b IN (SELECT b FROM S);
```

IN is a Predicate About R's Tuples

```
SELECT a
FROM R
WHERE b IN (SELECT b FROM S);
```

Two 2's

One loop, over the tuples of R

a	b
1	2
3	4

R

b	c
2	5
2	6

S

(1,2) satisfies the condition; 1 is output once

This Query Pairs Tuples from R, S

```
SELECT a
FROM R, S
WHERE R.b = S.b;
```

Double loop, over the tuples of R and S

a	b
1	2
3	4

R

b	c
2	5
2	6

S

(1,2) with (2,5) and (1,2) with (2,6) both satisfy the condition;
1 is output twice

The Exists Operator

- EXISTS(<subquery>) is true if and only if the subquery result is not empty
- Example: From Beers(name, manf), find those beers that are the unique beer by their manufacturer

Example: EXISTS

```
SELECT name
FROM Beers b1
WHERE NOT EXISTS (
    SELECT *
    FROM Beers
    WHERE manf = b1.manf AND
        name <> b1.name);
```

Set of beers with the same manf as b1, but not the same beer

Notice scope rule: manf refers to closest nested FROM with a relation having that attribute

Notice the SQL "not equals" operator

The Operator ANY

- x = ANY(<subquery>) is a boolean condition that is true iff x equals at least one tuple in the subquery result
  - = could be any comparison operator.
- Example: x >= ANY(<subquery>) means x is not the uniquely smallest tuple produced by the subquery
  - Note tuples must have one component only

The Operator ALL

- *x* <> ALL(<subquery>) is true iff for every tuple *t* in the relation, *x* is not equal to *t*
  - That is, *x* is not in the subquery result
- <> can be any comparison operator
- Example: *x* >= ALL(<subquery>) means there is no tuple larger than *x* in the subquery result

Example: ALL

- From Sells(bar, beer, price), find the beer(s) sold for the highest price

```
SELECT beer
FROM Sells
WHERE price >= ALL(
    SELECT price
    FROM Sells);
```

price from the outer Sells must not be less than any price.

Example: Subquery in FROM

- Find the beers liked by at least one person who frequents Cafe Chino

Drinkers who frequent C.Ch.

```
SELECT beer
FROM Likes, (SELECT drinker
    FROM Frequents
    WHERE bar = 'C.Ch.')CCD
WHERE Likes.drinker = CCD.drinker;
```

Exercise

Exercise 6.3.2: Write the following queries, based on the database schema

```
Classes(class, type, country, numGuns, bore, displacement)
Ships(name, class, launched)
Battles(name, date)
Outcomes(ship, battle, result)
```

of Exercise 2.4.3. You should use at least one subquery in each of your answers and write each query in two significantly different ways (e.g., using different sets of the operators `EXISTS`, `IN`, `ALL`, and `ANY`).

a) Find the countries whose ships had the largest number of guns.

! b) Find the classes of ships, at least one of which was sunk in a battle.

c) Find the names of the ships with a 16-inch bore.

d) Find the battles in which ships of the Kongo class participated.

!! e) Find the names of the ships whose number of guns was the largest for those ships of the same bore.

Extended Relational Algebra

The Extended Algebra

δ = eliminate duplicates from bags

τ = sort tuples

γ = grouping and aggregation

Outerjoin: avoids "dangling tuples" = tuples that do not join with anything

Duplicate Elimination

- $R_1 := \delta(R_2)$
- R_1 consists of one copy of each tuple that appears in R_2 one or more times

Example: Duplicate Elimination

R = (

A	B
1	2
3	4
1	2

)

$\delta(R)$ =

A	B
1	2
3	4

Sorting

- $R_1 := \tau_L (R_2)$
  - L is a list of some of the attributes of R_2
- R_1 is the list of tuples of R_2 sorted lexicographically according to the attributes in L, i.e., first on the value of the first attribute on L, then on the second attribute of L, and so on
  - Break ties arbitrarily
- τ is the only operator whose result is neither a set nor a bag

Example: Sorting

R = (

A	B
1	2
3	4
5	2

)

τ_B (R) = [(5,2), (1,2), (3,4)]

Aggregation Operators

- Aggregation operators are not operators of relational algebra
- Rather, they apply to entire columns of a table and produce a single result
- The most important examples: SUM, AVG, COUNT, MIN, and MAX

Example: Aggregation

R = (

A	B
1	3
3	4
3	2

)

SUM(A) = 7
COUNT(A) = 3
MAX(B) = 4
AVG(B) = 3

Grouping Operator

- $R_1 := \gamma_L (R_2)$

  L is a list of elements that are either:

  1. Individual (*grouping*) attributes
  2. AGG(A), where AGG is one of the aggregation operators and A is an attribute
     - An arrow and a new attribute name renames the component

Applying $\gamma_L(R)$

- Group *R* according to all the grouping attributes on list *L*
  - That is: form one group for each distinct list of values for those attributes in *R*
- Within each group, compute AGG(*A*) for each aggregation on list *L*
- Result has one tuple for each group:
  1. The grouping attributes and
  2. Their group's aggregations

Example: Grouping/Aggregation

R = (

A	B	C
1	2	3
4	5	6
1	2	5

)

$\gamma_{A,B,AVG(C)\text{->}X}$ (R) = ??

First, group *R* by *A* and *B* :

A	B	C
1	2	3
1	2	5
4	5	6

Then, average *C* within groups:

A	B	X
1	2	4
4	5	6

Outerjoin

- Suppose we join $R \bowtie_C S$
- A tuple of R that has no tuple of S with which it joins is said to be *dangling*
  - Similarly for a tuple of S
- Outerjoin preserves dangling tuples by padding them NULL

Example: Outerjoin

R = (

A	B
1	2
4	5

)

S = (

B	C
2	3
6	7

)

(1,2) joins with (2,3), but the other two tuples are dangling

R OUTERJOIN S =

A	B	C
1	2	3
4	5	NULL
NULL	6	7

Exercise

Exercise 5.2.1 : Here are two relations:

$R(A, B)$: $\{(0,1),\ (2,3),\ (0,1),\ (2,4),\ (3,4)\}$

$S(B, C)$: $\{(0,1),\ (2,4),\ (2,5),\ (3,4),\ (0,2),\ (3,4)\}$

Compute the following: a) $\pi_{A+B,A^2,B^2}(R)$; b) $\pi_{B+1,C-1}(S)$; c) $\tau_{B,A}(R)$; d) $\tau_{B,C}(S)$; e) $\delta(R)$; f) $\delta(S)$; g) $\gamma_{A,\ \mathtt{SUM}(B)}(R)$; h) $\gamma_{B,\mathtt{AVG}(C)}(S)$; ! i) $\gamma_A(R)$; ! j) $\gamma_{A,\mathtt{MAX}(C)}(R \bowtie S)$; k) $R \overset{\circ}{\bowtie}_L S$; l) $R \overset{\circ}{\bowtie}_R S$; m) $R \overset{\circ}{\bowtie} S$; n) $R \overset{\circ}{\bowtie}_{R.B<S.B} S$.

Summary 4

More things you should know:

- Subqueries in SQL
- Duplicate Elimination
- Sorting
- Aggregation
- Grouping
- Outer Joins