SQL: The Query Language
Part III
Nested Queries

R&G - Chapter 5

Simple SQL

- **The form:**

  SELECT $A_1, A_2, \ldots, A_n$
  FROM $r_1, r_2, \ldots, r_m$
  WHERE P

  - A_i represents an attribute
  - r_i represents a relation
  - P is a predicate

- **This query is equivalent to the relational algebra expression:**

$$\Pi_{A_1, A_2, \ldots, A_n}(\sigma_P(r_1 \times r_2 \times \ldots \times r_m))$$

Today's mission

- **Nested queries**
  - One of the most powerful features of SQL
  - Loved and hated

Definition

- **A nested query is a select-from-where query that has another select-from-where query embedded within it.**
  - The embedded query is called a *subquery*
  - The subquery appears within the WHERE clause
    - Can sometimes appear in the FROM clause
  - The subquery can be nested too
- **A common use of subqueries is to perform tests for set membership, set comparisons, and set cardinality.**

Components of sub-queries

- A subquery has the following components:
  - A regular SELECT query including the regular select list components.
  - A regular FROM clause including one or more table or view names.
  - An *optional* WHERE clause.
  - An *optional* GROUP BY clause.
  - An *optional* HAVING clause.
- The SELECT query of a subquery is always enclosed in parentheses

Operators in Nested Queries

- **Allowed set-comparison operators in nested queries**
  - IN, EXISTS, NOT IN, NOT EXISTS, UNIQUE
    - IN: *returns true if the value is in the set*
    - EXISTS: *returns true if the set is not empty*.
    - UNIQUE: *returns true if there are no duplicates*.
  - Also available: *op* ANY, *op* ALL
    - *op* : arithmetic operators (<, >, <=, >=, =, <>)
    - ANY: some systems use SOME

How to Connect Subquery with Outer Query

- Statements that include a subquery usually take one of these formats:
  - WHERE *expression* [NOT] IN (*subquery*)
  - WHERE [NOT] EXISTS (*subquery*)
  - WHERE *expression op* [ANY | ALL] (*subquery*)
    - *op* : arithmetic operators (<, >, <=, >=, =, <>)

Example of [NOT] IN Operator

- IN operator: *returns true if the value is in the set*

  Example: Find names of sailors who've reserved boat #103:

Use nested queries

```
SELECT  S.sname
FROM  Sailors S
WHERE  S.sid IN (SELECT R.sid
                 FROM  Reserves R
                 WHERE  R.bid=103)
```

subquery

- **To find sailors who've *not* reserved #103, use NOT IN.**

Evaluation of Nested Queries

```
SELECT  S.sname
FROM  Sailors S
WHERE  S.sid IN (SELECT R.sid
                  FROM  Reserves R
                  WHERE  R.bid=103)
```

- **To understand semantics of nested queries:**
  - think of a *nested loop* evaluation: *For each Sailors tuple,* examine whether it satisfies the subquery.

Example of [NOT] EXIST Operator

Find names of sailors who've reserved boat #103:

```
SELECT  S.sname
FROM  Sailors S
WHERE EXISTS (SELECT  *
              FROM  Reserves R
              WHERE R.bid=103 AND S.sid=R.sid)
```

- **EXISTS**: *returns true if the set is not empty*.
- **The inner subquery depends on the row currently being examined by the outer query**
- **Can also specify NOT EXISTS**

Rewrite INTERSECT Queries Using IN

Find sid's of sailors who've reserved both a red and a green boat

```
SELECT  R.sid
FROM Boats B, Reserves R
WHERE R.bid=B.bid
      AND B.color='red'
      AND R.sid IN (SELECT R2.sid
                    FROM  Boats B2, Reserves R2
                    WHERE  R2.bid=B2.bid
                    AND  B2.color='green')
```

- **What about *Find sid's of sailors who've reserved a red boat, but never reserved a green boat?***
  - EXCEPT queries re-written using NOT IN.

Rewrite INTERSECT Queries Using EXIST

Find sid's of sailors who've reserved both a red and a green boat

```
SELECT  R.sid
FROM Boats B, Reserves R
WHERE R.bid=B.bid
      AND B.color='red'
      AND EXIST (SELECT R2.sid
                 FROM Boats B2, Reserves R2
                 WHERE R2.bid=B2.bid AND
                       R.sid=R2.sid AND
                       B2.color='green')
```

Connect Subquery with ANY/ALL Operator

- Statements that include a subquery usually take one of these formats:
  - WHERE *expression* [NOT] IN (*subquery*)
  - WHERE [NOT] EXISTS (*subquery*)
  - WHERE *expression op* [ANY | ALL] (*subquery*)
    - *op* : arithmetic operators (<, >, <=, >=, =, <>)

Definition of ANY Clause

- **F <comp> ANY *r* ⇔ exists *t* ∈ *r* *s.t.* (F <comp> *t*)**
  Where <comp> can be: <, >, <=, >=, =, <>

(5< **ANY** | 0 | 5 | 6 |) = true

(read: 5 < some tuple in the relation)

(5< **ANY** | 0 | 5 |) = false

(5 = **ANY** | 0 | 5 |) = true

(5 <> **ANY** | 0 | 5 |) = true (since 0 ≠ 5)

(= **ANY**) ≡ **IN operator**

However, (<> **ANY**) ≢ **NOT IN**

Definition of ALL Clause

- **F <comp> all *r* ⇔ for all *t* ∈ *r* (F <comp> *t*)**

(5< **all** $\boxed{0}$ $\boxed{5}$ $\boxed{6}$) = false

(5< **all** $\boxed{6}$ $\boxed{10}$) = true

(5 = **all** $\boxed{4}$ $\boxed{5}$) = false

(5 <> **all** $\boxed{4}$ $\boxed{6}$) = true (since 5 ≠ 4 and 5 ≠ 6)

(<> **ALL**) ≡ **not in**

However, (= **ALL**) ≢ **in**

Example of ANY Operator

- **Find sailors whose rating is greater than the rating of <u>some</u> sailors whose name is Lubber:**

```
SELECT  *
FROM  Sailors S
WHERE  S.rating > ANY (SELECT  S2.rating
                        FROM  Sailors S2
                       WHERE S2.sname='lubber')
```

Example of ALL Operator

- **Find sailors whose rating is greater than the rating of <u>all</u> sailors whose name is Lubber:**

```
SELECT  *
FROM  Sailors S
WHERE  S.rating > ALL (SELECT  S2.rating
                        FROM  Sailors S2
                       WHERE S2.sname='lubber')
```

Example of ALL Operator

- **Find sailors who have the highest rating.**

```
SELECT  *
FROM  Sailors S
WHERE  S.rating >= ALL (SELECT  S2.rating
                        FROM  Sailors S2);
```

Question: can we use "S.rating > ALL"

Division in SQL

Find the name of sailors who've reserved all boats.

$$\rho(Tempsids,(\pi_{sid,bid}\text{Re}serves)/(\pi_{bid}Boats))$$

$$\pi_{sname}(Tempsids \bowtie Sailors)$$

How can we write it in SQL?

```
SELECT * FROM USERS
WHERE CLUE > 0

(0 row(s) affected)
```

Division in SQL

Find sailors who've reserved all boats.

Rephrase: Find the sailors for whom there is no such boat that he/she has not reserved.

```
SELECT  S.sname
FROM  Sailors S
WHERE  NOT EXISTS  (SELECT  B.bid
                    FROM  Boats B
                    WHERE  NOT EXISTS  (SELECT  R.bid
                                        FROM  Reserves R
                                        WHERE  R.bid=B.bid
                                               AND R.sid=S.sid))
```

Sailors S such that ...

there is no boat B without ...

a Reserves tuple showing S reserved B

Basic SQL Queries - Summary

- SQL provides functionality close to that of the basic relational model.
  - some differences in duplicate handling, null values, set operators, etc.
- Typically, many ways to write a query
  - the system is responsible for figuring a fast way to actually execute a query regardless of how it is written.
- Lots more functionality beyond these basic features. Will be covered in subsequent lectures.