

SQL: The Query Language

Part III

Nested Queries

R&G - Chapter 5



Announcement

- **Assignment 4: SQL**

- Available in Canvas
- Due on Nov 2 (after midterm exam)

- **Midterm exam**

- Held in class on Nov 1.
- The exam will be taken on Canvas
- Cover ER diagram design, translating ER diagram to relational models, relational algebra and SQL
- 1 single-side A4 cheat sheet is allowed
- Reminder: take “Relational algebra operators exercise” before Midterm exam (available in Canvas under “Quizzes”)

Simple SQL

- **The form:**

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

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

- **Set operations**

Subquery 1

UNION/INTERSECT/EXCEPT

Subquery 2

Today's mission

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

Definition

- A Subquery (or Inner query, or Nested query) is a query within another SQL query and embedded within the WHERE clause.

```
Outer/main
query      SELECT
            FROM
            WHERE EXPRESION IN/EXISTS/op [ANY/ALL]
                (SELECT
                 subquery FROM
                 WHERE ... ) ;
```

- The subquery can be nested too
- A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Three Rules of Subqueries

1. In most cases, subqueries are included in the WHERE clause of the main query
 - Rarely subqueries are included in FROM clause
2. Subqueries must be enclosed within parentheses.
3. In most cases, a subquery has only one attribute in the SELECT clause
 - Rarely more than one attribute is included in the SELECT clause of the subquery for comparison with main query.

Connect Subquery with Main Query

- Statements that include a subquery usually take one of these formats
 - WHERE *attribute_name* **[NOT] IN** (*subquery*)
 - WHERE [NOT] **EXISTS** (*subquery*)
 - WHERE *expression op* **[ANY | ALL]** (*subquery*)

Connect Subquery with Main Query

Format 1:

In WHERE clause of the main query:

WHERE *attribute(s)* IN (*subquery*)

- IN: a set operation that checks *set membership*
- Subquery returns a set of values S
- WHERE ... IN returns *true* if there is any value of *attribute* is in S; otherwise, return false.
- *attribute* can contain multiple attributes
- Negation: WHERE *attribute* NOT IN (*subquery*)

Example of IN Operator

- **Schema**

- Boats (bid, bname, color)
- Sailors(sid, sname, rating, age)
- Reserves(sid, bid, day)

Query: 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.

Set Consistency for IN Operator

- The attributes before IN operator must be THE SAME as the attributes in SELECT clause of the subquery.

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

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

subquery

- Is this query correct?
 - NOT correct, as the subquery output pairs (sid, day), while the main query checks the membership of sid.

Evaluation of IN Queries

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

- **Evaluation of nested query**

- Similar to *nested loop* evaluation: *For each Sailors tuple, examine whether it satisfies the subquery.*
- The output of the subquery in this example is the same for each sailor tuple
 - The evaluation of subquery does NOT depend on the sailor tuple that is currently being examined by the outer query

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*)

Connec Subquery with Main Query

- **Format 2:**

In WHERE clause of the main query:

WHERE EXISTS (*subquery*)

- EXISTS: a Boolean operator that checks value existence
- WHERE EXISTS returns *true* if the result of subquery is not empty; otherwise returns *false*
- Negation: WHERE *expression* NOT EXISTS (*subquery*)

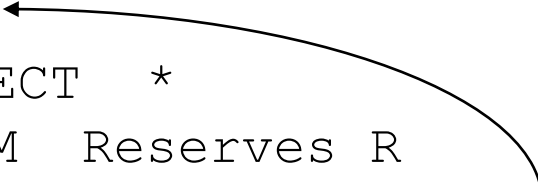
Example of EXISTS Operator

- **Schema**

- Boats (bid, bname, color)
- Sailors(sid, sname, rating, age)
- Reserves(sid, bid, day)

Query: 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 output of the subquery in this example is NOT the same for each sailor tuple
 - The subquery depends on the sailor record currently being examined by the outer query
- Can also specify negation **NOT EXISTS**

Example of EXISTS Operator



- **Schema**

- Boats (bid, bname, color)
- Sailors(sid, sname, rating, age)
- Reserves(sid, bid, day)

Query: find names of sailors who've reserved boat #103.

```
SELECT S.sname
FROM   Sailors S
WHERE  EXISTS (SELECT *
                FROM   Reserves R, Sailors S1
                WHERE  R.bid=103 AND S1.sid=R.sid);
```

- Is this query correct?
 - What's the output of this query if there is a reservation of boat #103?
- How to fix the query?

Rewrite INTERSECT Queries Using IN

Query: Find ID of sailors who've reserved both a red and a green boat

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

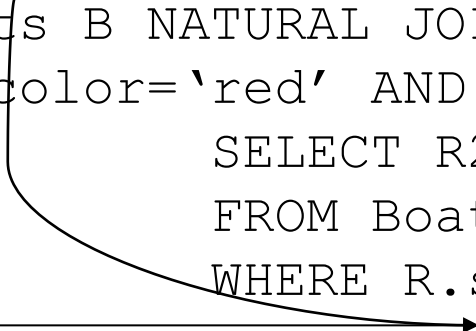
- How to write the SQL statement for the query: "*Find ID of sailors who've reserved a red boat, but never reserved a green boat*" (i.e., for *EXCEPT* queries)?
 - *EXCEPT* queries use NOT IN.



Rewrite INTERSECT Queries Using EXISTS

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

```
SELECT  R.sid
FROM Boats B NATURAL JOIN Reserves R
WHERE B.color='red' AND EXISTS (
    SELECT R2.sid
    FROM Boats B2 NATURAL JOIN Reserves R2
    WHERE R.sid=R2.sid AND B2.color='green');
```



- Add R.sid=R2.sid in subquery to connect main and sub queries
- Question: what will be the output of this query if there is no R.sid=R2.sid in WHERE clause of the subquery?
 - When there is a reservation of a green boat?
 - When there is no reservation of a green boat?

Connect Subquery with Main 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*)

Connect Subquery with Main Query

- **Format 3:**

In WHERE clause of the main query:

WHERE *expression* *op* [ANY | ALL] (*subquery*)

- *op* : arithmetic operators (<, >, <=, >=, =, <>)

ANY Operator

- ANY(EXISTENCE quantifier)
 - Syntax: $v \text{ op ANY } S$
 - v : a single value;
 - S : a set of values
 - op : $=$, $<>$, $<$, $>$, $<=$, $>=$
 - return true if **at least one** element t in S such that $v \text{ op } t$ is true

$(5 < \text{ANY } \begin{array}{|c|} \hline 0 \\ \hline 5 \\ \hline 6 \\ \hline \end{array}) = \text{true}$ (read: $5 <$ some tuple in the relation)

$(5 < \text{ANY } \begin{array}{|c|} \hline 0 \\ \hline 5 \\ \hline \end{array}) = \text{false}$

$(5 = \text{ANY } \begin{array}{|c|} \hline 0 \\ \hline 5 \\ \hline \end{array}) = \text{true}$

$(5 <> \text{ANY } \begin{array}{|c|} \hline 0 \\ \hline 5 \\ \hline \end{array}) = \text{true}$ (since $0 <> 5$)

ALL Operator

- ALL(UNIVERSAL quantifier)
 - Syntax: $v \text{ op } ALL \ S$
 - v : a single value;
 - S : a set of values
 - op : $=$, $<>$, $<$, $>$, $<=$, $>=$
 - returns true if **for each element** t in S , $v \text{ op } t$ is true.

$$(5 < \text{all } \begin{array}{|c|} \hline 0 \\ \hline 5 \\ \hline 6 \\ \hline \end{array}) = \text{false}$$

$$(5 < \text{all } \begin{array}{|c|} \hline 6 \\ \hline 10 \\ \hline \end{array}) = \text{true}$$

$$(5 = \text{all } \begin{array}{|c|} \hline 4 \\ \hline 5 \\ \hline \end{array}) = \text{false}$$

$$(5 <> \text{all } \begin{array}{|c|} \hline 4 \\ \hline 6 \\ \hline \end{array}) = \text{true (since } 5 <> 4 \text{ and } 5 <> 6)$$

Example of ANY Operator

- **Schema**

- Boats (bid, bname, color)
- Sailors(sid, sname, rating, age)
- Reserves(sid, bid, day)

Query: find sailors whose rating is greater than the rating of at least one sailor who are older than 20.

```
SELECT      *
FROM        Sailors S
WHERE       S.rating          (SELECT  S2.rating
                                FROM    Sailors S2
                                WHERE   S2.age>20) ;
```

Example of ANY Operator

- **Schema**

- Boats (bid, bname, color)
- Sailors(sid, sname, rating, age)
- Reserves(sid, bid, day)

Query: find sailors whose rating is greater than the rating of at least one sailor who are older than 20.

```
SELECT  *
FROM    Sailors S
WHERE   S.rating > ANY (SELECT  S2.rating
                        FROM    Sailors S2
                        WHERE   S2.age>20) ;
```

Example of ALL Operator

- **Schema**

- Boats (bid, bname, color)
- Sailors(sid, sname, rating, age)
- Reserves(sid, bid, day)

Query: find sailors whose rating is greater than the rating of all sailors who are older than 20.

```
SELECT      *
FROM        Sailors S
WHERE       S.rating          (SELECT  S2.rating
                                FROM    Sailors S2
                                WHERE   S2.age>20) ;
```


Example of ALL Operator

- **Schema**

- Boats (bid, bname, color)
- Sailors(sid, sname, rating, age)
- Reserves(sid, bid, day)

Query: find sailors whose rating is greater than the rating of all sailors who are older than 20.

```
SELECT      *
FROM        Sailors S
WHERE       S.rating > ALL (SELECT  S2.rating
                             FROM    Sailors S2
                             WHERE   S2.age>20) ;
```

Example of ALL Operator

- **Schema**
 - Boats (bid, bname, color)
 - Sailors(sid, sname, rating, age)
 - Reserves(sid, bid, day)

Query: find sailors who have the highest rating.

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

Example of ALL Operator

- **Schema**

- Boats (bid, bname, color)
- Sailors(sid, sname, rating, age)
- Reserves(sid, bid, day)

Query: find sailors who have the highest rating.

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

Question: what if we use S.rating >ALL?



Example of ALL Operator

- The previous solution is wrong as no sailor can have higher rating than himself/herself. Thus it will return empty answer.
- What if a condition `WHERE S.sid <> S2.sid` is added to subquery?

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

Question: What will be the output by this query for the following 2 cases?

- (1) There is only one sailor of the highest rating
- (1) There are multiple sailors of the highest rating