Non-Scalar Subqueries

Database Development 2022
Cody Henrichsen

1

Definition

▷ Multi-row subqueries are any subquery that returns more than one row as the value.
▷ Multi-column subqueries have more than one column (duh)
▷ As such the equality and relational operators (on their own) are not going to work to provide a comparison
▷ Instead we will use the IN, ANY, and ALL keywords in conjunction with the equality/relational operators

2

Usage

▷ These are more common subqueries as they generate more information that can be compared with multiple values.
▷ Most questions that need SQL to answer do not resolve to a scalar subquery
▷ Rules to process the WHERE clause are different depending on type of query

3

Multiple columns and/or rows

▷ Looking for a matching **SET** of values
▷ The column count, type, and order must match on both sides of the comparison
▷ Use the IN or NOT IN keywords for doing an equality comparison against the values returned by they subquery

4

ANY and ALL

▷ One column, multiple rows
▷ Use the ANY keyword with a equality or relational operator to compare values against at least one of the subquery values
  ▷ Think logical OR
▷ Use the ALL keyword with a relational operator ONLY to compare EVERY value in the subquery
  ▷ Think logical AND

5

Columns Example

```
SELECT
    stu.first_name,
    stu.zip
FROM
    student stu
WHERE
    (stu.student_id, stu.created_by) IN -- Use IN/NOT IN for a match of multiple values
    (
        SELECT
            sec.section_id,
            sec.modified_by
        FROM
            section sec
        WHERE
            sec.location LIKE 'L%'
    )
;
```

6

Rows: ANY / ALL

Logical OR

```
SELECT
    stu.first_name,
    stu.zip
FROM
    student stu
WHERE
    stu.student_id <= ANY
    (
        SELECT
            sec.section_id
        FROM
            section sec
        WHERE
            sec.location LIKE 'L%'
    )
;
```

Logical AND

```
SELECT
    stu.first_name,
    stu.zip
FROM
    student stu
WHERE
    stu.student_id > ALL
    (
        SELECT
            sec.section_id
        FROM
            section sec
        WHERE
            sec.location LIKE 'L%'
    )
;
```

7

_____

_____

_____

_____

_____

_____

_____