Database System Lab (CSE 3104)

Session 8

Nazmus Sakib, Assistant Professor, Dept. of CSE, AUST

Nowshin Nawar Arony, Adjunct Lecturer, Dept. of CSE, AUST

Subquery

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

- Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

Rules that Subqueries must follow

- Subqueries must be enclosed within parentheses.
- Subqueries that return more than one row can only be used with multiple value operators such as the IN operator.
- A subquery can have only one column in the SELECT clause.
- An ORDER BY command cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY command can be used to perform the same function as the ORDER BY in a subquery.

Subqueries with the SELECT Statement

Syntax:

```
SELECT column_name(s)
FROM table WHERE column_name OPERATOR
(SELECT column_name(s) FROM table [WHERE] )
```

Subqueries that return only one value

Example:

```
SELECT * FROM CUSTOMER WHERE
CUSTOMERID = (SELECT CUSTOMERID
FROM CUSTOMERORDER WHERE OrderDate='2019-01-13');
```

```
SELECT * FROM CUSTOMER WHERE
CUSTOMERID <= (SELECT CUSTOMERID
FROM CUSTOMERORDER WHERE OrderDate='2019-01-13');
```

Using Between

Example:

```
SELECT * FROM CUSTOMER WHERE
CUSTOMERID BETWEEN (SELECT CUSTOMERID
FROM CUSTOMERORDER
WHERE OrderDate='2019-01-13')
AND 5;
```

```
SELECT * FROM CUSTOMER WHERE
CUSTOMERID BETWEEN
(SELECT CUSTOMERID FROM CUSTOMERORDER
WHERE OrderDate='2019-01-13')
AND
(SELECT CUSTOMERID FROM CUSTOMERORDER
WHERE OrderDate= '2019-01-12')
```

Subqueries that return multiple values

Syntax:

```
SELECT column_name(s)
FROM table WHERE column_name IN
(SELECT column_name(s) FROM table [WHERE] )
```

Example:

```
SELECT * FROM CUSTOMER WHERE
CUSTOMERID IN (SELECT CUSTOMERID
FROM CUSTOMERORDER WHERE Bill > 10);
```

Any/SOME Keyword

The ANY OR SOME keyword denotes that the search condition is TRUE if the comparison is TRUE for **at least one** of the values that is returned.

If the subquery returns no value, the search condition is FALSE.

Syntax:

```
SELECT column_name(s)
FROM table WHERE column_name OPERATOR
ANY(SELECT column_name(s) FROM table [WHERE] )
```

Example:

```
SELECT * FROM CUSTOMER WHERE
CUSTOMERID > ANY (SELECT CUSTOMERID
FROM CUSTOMERORDER WHERE Bill > 10)
```

Syntax:

```
SELECT column_name(s)
FROM table WHERE column_name OPERATOR
SOME(SELECT column_name(s) FROM table [WHERE] )
```

Example:

```
SELECT * FROM CUSTOMER WHERE
CUSTOMERID > SOME(SELECT CUSTOMERID
FROM CUSTOMERORDER WHERE Bill > 10)
```

ALL Keyword

The ALL keyword specifies that the search condition is TRUE if the comparison is TRUE for **every** value that the subquery returns.

If the subquery returns no value, the condition is FALSE.

Syntax:

```
SELECT column_name(s)
FROM table WHERE column_name OPERATOR
ALL(SELECT column_name(s) FROM table [WHERE] )
```

Example:

```
SELECT * FROM CUSTOMER WHERE
CUSTOMERID > ALL (SELECT CUSTOMERID
FROM CUSTOMERORDER WHERE Bill > 10)
```

ORDER BY IN SUBQUERY GIVES ERROR

ERROR:

```
SELECT * FROM CUSTOMER WHERE CUSTOMERID
IN (SELECT CUSTOMERID FROM CUSTOMERORDER
WHERE Bill > 10 ORDER BY CUSTOMERID)
```

CORRECT VERSION:

```
SELECT * FROM CUSTOMER WHERE CUSTOMERID
IN (SELECT CUSTOMERID FROM CUSTOMERORDER
WHERE Bill > 10 GROUP BY CUSTOMERID)
```

USING Aggregate FUNCTIONS

- **SHOW THOSE CUSTOMER IDs WHOSE AVERAGE BILLS ARE GREATER THAN THE AVERAGE OF ALL THE ORDERS**

```
SELECT CUSTOMERID FROM CUSTOMERORDER
GROUP BY CUSTOMERID HAVING
AVG(BILL) > (SELECT AVG(BILL) FROM CUSTOMERORDER)
```

Nested subqueries

- A subquery can be nested inside other subqueries.

- Show the OrderIDs of the Orders Placed Under the Salesmen whose salary is more than the average salary of all salesmen.

```
select OrderID from CustomerOrder
where SalesmanId In ( Select SalesmanId
From Salesman Where Salary > ( Select AVG(Salary)
From Salesman))
```

- Show the FirstName, LastName of those Customers who are attended by Salesman whose salary is greater than 30,000

```
Select FirstName, LastName from CUSTOMER
Where CUSTOMERID IN (
Select CUSTOMERID From CustomerORDER
Where SalesmanID = ANY(
Select SalesmanID from Salesman where Salary > 30000))
```

EXISTS Keyword

The **EXISTS** operator checks if a subquery returns any rows.

WHERE EXISTS (subquery)

The expression EXISTS (subquery) returns TRUE if the subquery returns at least one row, otherwise it returns FALSE.

```
SELECT * FROM CUSTOMER WHERE
Exists (SELECT CUSTOMERID
FROM CUSTOMERORDER
WHERE Bill > 10)
```

NOTE:
EXISTS is different from other operators like IN,ANY etc., because it doesn’t compare values of columns, instead, it checks whether any row is retrieved from subquery or not.

NOT EXISTS Keyword

NOT operator with the EXISTS operator to inverse the meaning of the EXISTS operator.

WHERE NOT EXISTS (subquery)

The expression NOT EXISTS (subquery) returns TRUE if the subquery returns no row, otherwise it returns FALSE.

NOT EXISTS Keyword

```
SELECT * FROM
CUSTOMER WHERE
Not Exists
(SELECT CUSTOMERID
FROM CUSTOMERORDER
WHERE Bill > 10)
```

A subquery can also be found in the **Select clause.**

Example:

```
SELECT CUSTOMERID, Firstname,
(SELECT SUM(Bill) FROM CustomerOrder
where Customer.CustomerID = CustomerOrder.CustomerID
Group by CustomerID) AS TotalBill
FROM Customer
```

Renaming the table

You can also rename the table as table_name t1.

Example:

```
SELECT CUSTOMERID, Firstname,
(SELECT SUM(Bill) FROM CustomerOrder c2
where c1.CustomerID = c2.CustomerID
Group by CustomerID) AS TotalBill
FROM Customer c1
```

A subquery can also be found in the **FROM clause.**

Example:

```
select CUSTOMER.CUSTOMERID, subquery.total_amt
from CUSTOMER,
(select CUSTOMERORDER.CUSTOMERID,
Sum(CUSTOMERORDER.Bill) as total_amt
from CUSTOMERORDER
group by CUSTOMERID) subquery
WHERE subquery.CUSTOMERID = CUSTOMER.CUSTOMERID
```

Subquery in Insert Statement

- The INSERT statement uses the data returned from the subquery to insert into another table.

```
INSERT INTO CUSTOMER (FirstName, LastName)
SELECT FirstName, LastName FROM CUSTOMER
WHERE CustomerID IN (SELECT CustomerID
FROM CUSTOMEROrder
Group By CustomerID Having Avg(Bill)>10);
```

Subquery in Update Statement

- Either single or multiple columns in a table can be updated when using a subquery with the UPDATE statement.

```
UPDATE CUSTOMER
SET COUNTRY = 'USA'
WHERE FirstName IN
 (SELECT FirstName FROM CUSTOMER
WHERE CUSTOMERID >5);
```

Subquery in Delete Statement

- The subquery can be used in conjunction with the DELETE statement as well.

```
DELETE FROM CUSTOMER
Where CUSTOMERID Not IN
(SELECT CUSTOMERID FROM CUSTOMERORDER);
```