Sub Query

Rosa Ariani Sukamto
Email: rosa_if_itb_01@yahoo.com
Blog: http://udinrosa.wordpress.com
Website: http://www.gangsir.com

Apaan Ya????

- **Query di dalam query**

- **Subquery diletakkan pada klausa WHERE atau klausa HAVING atau bisa juga di klausa FROM**

Sub Query

SELECT *select_list*
FROM *table*
WHERE *exp_operator*

Inner Query akan di kerjakan terlebih dahulu

```
SELECT Nama FROM pacar
WHERE jumlah_kartu_kredit >
    (
        SELECT jumlah_kartu_kredit
        FROM pacar WHERE Nama='Tomi'
    );
```

```
SQL> SELECT Nama FROM pacar WHERE jumlah_kartu_kredit > ( SELECT jumlah_kartu_kredit FROM pacar WHER
E Nama='Tomi');

NAMA
--------------------------------------------------
Sami
Boy
Bay
```

Sub Query untuk Multi Kolom

Setiap row pada main query dibandingkan nilai dari multiple-row (multi baris) dan multiple-column (multi-kolom) pada subquery.

```
SELECT column, column . . .
FROM table
WHERE (columnA,columnB)
IN
    (
        SELECT column1,column2
        FROM table
        WHERE condition
    );
```

Perbandingan Kolom

Kolom pembanding dalam multiple-column (multi kolom) subquery ada dua jenis

- **Pairwise comparisons**
- **Nonpairwise comparisons – beberapa sub query yang digabung dengan operator logika AND**

Pairwise

```
SELECT jumlah_kartu_kredit, nama, alamat
FROM pacar WHERE (nama, alamat)
IN(
   SELECT nama, alamat FROM pacar
   WHERE pekerjaan_babe='Dokter'
);
```

```
SQL> SELECT jumlah_kartu_kredit, nama, alamat FROM pacar WHERE (nama, alamat) IN(SELECT nama, alamat
 FROM pacar WHERE pekerjaan_babe='Dokter');

no rows selected
```

Non Pairwise

```
SELECT nama, alamat, jumlah_kartu_kredit
FROM pacar
WHERE pekerjaan_babe IN
    (
        SELECT pekerjaan_babe
        FROM pacar
        WHERE nama IN('Sami', 'Tomi')
    )
    AND jumlah_kartu_kredit IN
    (
        SELECT jumlah_kartu_kredit
        FROM pacar
        WHERE pekerjaan_babe='Dokter'
    );
```

Sub Query pada Klausa FROM

- **Subquery yang diletakkan pada klausa FROM disebut INLINE VIEW**
- **Subquery yang diletakkan pada klausa FROM digunakan sebagai sumber data (berperan sebagai tabel)**

```
SELECT nama, alamat
FROM
    (
        SELECT nama, alamat
        FROM pacar
        WHERE jumlah_kartu_kredit=1
    ) ;
```

Ekspresi Sub Query Skalar

- **Subquery yang hanya menghasilkan satu kolom pada satu baris (tunggal).**
- **Scalar subquery bisa digunakan pada DECODE dan CASE.**

```
SELECT employee_id, last_name, DEPARTMENT_ID,
    (
        CASE
        WHEN department_id =
        (
            SELECT department_id
            FROM Departments
            WHERE location_id = 1800
        )
        THEN 'Canada'
        ELSE 'USA'
        END
    ) AS Location
FROM employees;
```

Sub Query Skalar pada Klausa ORDER BY

- **Kondisi seperti ini disebut correlated subquery.**
- **Sub Query dikerjakan belakangan**

```
SELECT employee_id,last_name
FROM employees e
ORDER BY
(
        SELECT department_name
        FROM departments d
        WHERE e.department_id = d.department_id
);
```

Sub Query Skalar pada Klausa WHERE

```
SELECT last_name, salary, department_id
FROM employees outer
WHERE salary >
(
    SELECT AVG(salary)
    FROM employees
    WHERE department_id = outer.department_id
);


SELECT e.employee_id, last_name, e.job_id
FROM employees e
WHERE 2 <=
(
    SELECT COUNT(*)
    FROM job_history
    WHERE employee_id = e.employee_id
);
```

Operator EXISTS

- **Digunakan untuk mengecek, apakah existensi dari hasil subquery.**
- **Jika hasil subquery ada :**
  - **Proses pencarian selesai**
  - **Kondisi di set TRUE**
- **Jika hasil subquery tidak ada :**
  - **Proses pencarian diteruskan sampai selesai**
  - **Kondisi di set FALSE**

Operator EXISTS

```
SELECT nama, alamat FROM pacar p
WHERE EXISTS
(
    SELECT 'd' FROM pacar WHERE
    jumlah_kartu_kredit=p.jumlah_kartu_kredit
);
```

```
SQL> SELECT nama, alamat FROM pacar p WHERE EXISTS( SELECT 'd' FROM pacar WHERE jumlah_kartu_kredit=
p.jumlah_kartu_kredit);

NAMA
--------------------------------------------------
ALAMAT
--------------------------------------------------------------------------------
Sami


Boy


Bay


NAMA
--------------------------------------------------
ALAMAT
--------------------------------------------------------------------------------
Tomi


Jay
```

Operator NOT EXISTS

- Kebalikan dari EXISTS operator

Contoh:
Mencari semua department yang tidak ada pegawainya.

```
SELECT department_id, department_name
FROM departments d
WHERE NOT EXISTS
(
    SELECT 'X'
    FROM employees
    WHERE department_id = d.department_id
);
```

- Selain untuk Query, correlated subquery juga bisa digunakan untuk proses UPDATE dan DELETE

Correlated Update

```
UPDATE employees e
SET department_name =
(
        SELECT department_name
        FROM departments d
        WHERE e.department_id = d.department_id
);
```

Correlated Delete

```
DELETE FROM employees_dummy E
WHERE employee_id =
(
        SELECT employee_id
        FROM emp_history
        WHERE employee_id = E.employee_id
);
```

Klausa WITH

- Digunakan untuk mendefinisikan block subquery, sehingga subquery yang sama tidak perlu ditulis ulang.
- WITH clause akan menyimpan hasil subquery di dalam temporary tablespacenya user.
- Dengan WITH clause akan meningkatkan performance.

```
WITH dept_costs AS
(
    SELECT department_name, SUM(salary) AS dept_total
    FROM employees,departments
    WHERE employees.department_id = Departments.department_id
    GROUP BY department_name
),
Avg_cost AS
(
    SELECT SUM(dept_total)/count(*) AS dept_avg
    FROM dept_costs
)
SELECT * FROM dept_costs
WHERE dept_total >
(
    SELECT * FROM avg_cost
)
ORDER BY department_name;
```

Penggunaan Tanda Petik Satu

- Character Literal (Penanda Karakter)

  …. WHERE name='Joni''s Baker'
- Date Literal

  …. WHERE tanggal='01-JAN-00'
- Column Heading

  COLUMN last_name HEADING 'LastName'
- Date Format Model

  TO_CHAR(sysdate, 'MM/DD/YY')

Penggunaan Tanda Petik Dua

- Alias

  SELECT last_name “employee” ……

- Date Format Literal

  TO_CHAR(sysdate, ‘DdTH”of”Month’)