

6. JOINING TABLES

Obyektif :

Setelah menyelesaikan bahasan ini, diharapkan dapat melakukan hal berikut:

1. Menuliskan statement SELECT yang melibatkan beberapa tabel sekaligus
 2. Menyebutkan dan menjelaskan tipe-tipe JOIN dalam SQL Server
 3. Memahami relasi field antar table
 4. Menampilkan data yang secara umum tidak diperbolehkan pada kondisi join menggunakan Outer Join
 5. Memahami penggunaan sub query
 6. Memahami perbedaan berbagai jenis sub query
 7. Membuat sintaks sub query yang tepat untuk menampilkan data yang dibutuhkan
-

6.1. Menampilkan Data dari Beberapa Table

Dalam proses Normalisasi database, tabel yang besar akan dipecah menjadi tabel-tabel yang kecil dengan tujuan menghilangkan pengulangan data (duplikasi data), memperkecil ukuran database, mempercepat pemrosesan data (select, insert, delete, update), dan meningkatkan integritas data. Clausa 'JOIN' dalam statement T-SQL dapat digunakan untuk menggabungkan data – data yang saling berkaitan antar tabel-tabel yang telah dipecah tadi.

Ada 4 macam JOIN dalam SQL server 2008 yaitu :

1. INNER JOIN.
2. OUTER JOIN.
3. FULL JOIN.
4. CROSS JOIN.

6.2. INNER JOIN

Inner join adalah operasi yang sering digunakan. Inner Join akan menampilkan data dari dua tabel yang memiliki nilai field (data) yang sama pada kedua tabelnya seperti yang disebutkan dalam klausa INNER JOIN. Jadi yang akan ditampilkan dari operasi ini adalah baris record yang memiliki pasangan nilai yang sama.

Sintaks :

```
SELECT <select list>
FROM <first_table>
INNER JOIN <secon table> [ON <join_condition>] [OTHER OPTION]
```

Dalam SQL 2008 - penggunaanya secara default – klausa ‘JOIN’ adalah sama dengan ‘INNER JOIN’.

Contoh:

```
SELECT a. DEPTNO, a. ENAME, b. DNAME
FROM EMP AS a INNER JOIN DEPT AS b
ON a. DEPTNO = b. DEPTNO
ORDER BY a. DEPTNO DESC
```

Setelah dua atau lebih direlasikan berdasarkan field yang sama, data dapat diambil dari setiap tabel yang ada dengan perintah SELECT, maupun menyeleksi berdasarkan kondisi tertentu. Contoh perintah yang melibatkan 3 tabel :

```
SELECT a. DEPTNO, a. ENAME, b. DNAME, c. GRADE
FROM EMP a, DEPT b, SALGRADE c
ON b. DEPTNO = a. DEPTNO
AND a. SAL BETWEEN c. LOSAL AND c. HISAL
```

	DEPTNO	ENAME	DNAME	GRADE
1	20	SMITH	RESEARCH	1
2	30	ALLEN	SALES	3
3	30	WARD	SALES	2
4	20	JONES	RESEARCH	4
5	30	MARTIN	SALES	2
6	30	BLAKE	SALES	4
7	10	CLARK	ACCOUNTING	4
8	20	SCOTT	RESEARCH	4
9	10	KING	ACCOUNTING	5
10	30	TURN...	SALES	3
11	20	ADAMS	RESEARCH	1
12	30	JAMES	SALES	1
13	20	FORD	RESEARCH	4
14	10	MILLER	ACCOUNTING	2

Query executed successfully.

6.3. OUTER JOIN

Perintah OUTER JOIN hampir sama dengan INNER JOIN, bedanya adalah baris record yang tidak memiliki pasangan akan tetap di tampilkan. Dengan kata lain, perintah ini menghasilkan data dari kedua tabel dimana :

- Akan menampilkan data yang memiliki kesamaan nilai pada kedua tabel.
- Akan menampilkan data yang tidak memiliki kesamaan nilai pada kedua nilai dengan memberikan nilai NULL pada 'JOIN CONDITION'.

Sintaks :

```
SELECT <SELECT list>
FROM <the table you want to be the 'LEFT' table>
<LEFT | RIGHT> [OUTER]
JOIN < the table you want to be the 'RIGHT' table>
ON <JOIN CONDITION> [OTHER OPTION]
```

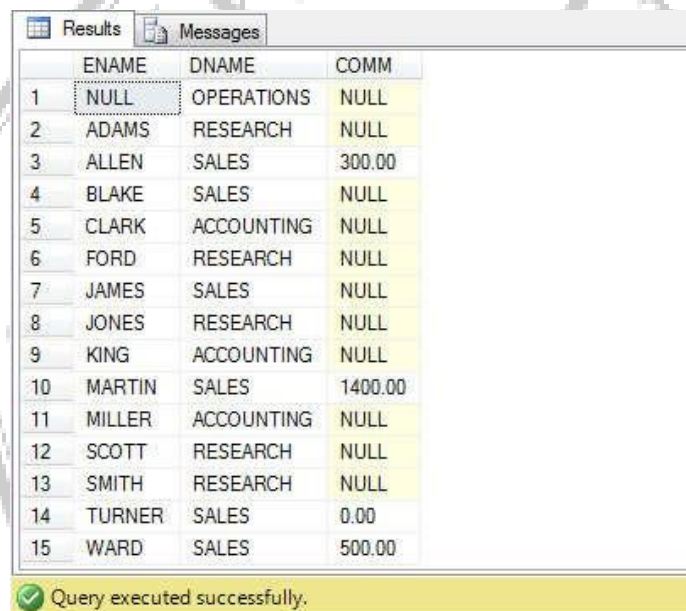
Jenis – jenis Outer Join :

1. Left Outer Join

Menampilkan seluruh tabel kiri dan tabel sebelah kanan yang mempunyai sepasang nilai dan jika table disebelah kanan tidak memiliki nilai akan ditampilkan NULL.

Contoh:

```
SELECT EMP. ENAME, DEPT. DNAME, EMP. COMM
FROM DEPT LEFT JOIN EMP
ON EMP. DEPTNO = DEPT. DEPTNO
ORDER BY ENAME ASC
```



	ENAME	DNAME	COMM
1	NULL	OPERATIONS	NULL
2	ADAMS	RESEARCH	NULL
3	ALLEN	SALES	300.00
4	BLAKE	SALES	NULL
5	CLARK	ACCOUNTING	NULL
6	FORD	RESEARCH	NULL
7	JAMES	SALES	NULL
8	JONES	RESEARCH	NULL
9	KING	ACCOUNTING	NULL
10	MARTIN	SALES	1400.00
11	MILLER	ACCOUNTING	NULL
12	SCOTT	RESEARCH	NULL
13	SMITH	RESEARCH	NULL
14	TURNER	SALES	0.00
15	WARD	SALES	500.00

Query executed successfully.

2. Right Outer Join

Menampilkan seluruh table disebelah kanan dan table disebelah kiri yang mempunyai pasangan nilai dan jika table disebelah kiri tidak memiliki nilai akan ditampilkan null; Contoh:

```
SELECT EMP. ENAME, DEPT. DNAME, EMP. COMM
FROM EMP RIGHT JOIN DEPT
ON EMP. DEPTNO = DEPT. DEPTNO
ORDER BY ENAME ASC
```

Hasil Join table akan sama seperti pada Left Outer Join dimana data yang tidak terdefinisi pada table sebelah kiri akan memunculkan isi data berupa NULL.

6.4 . FULL JOIN

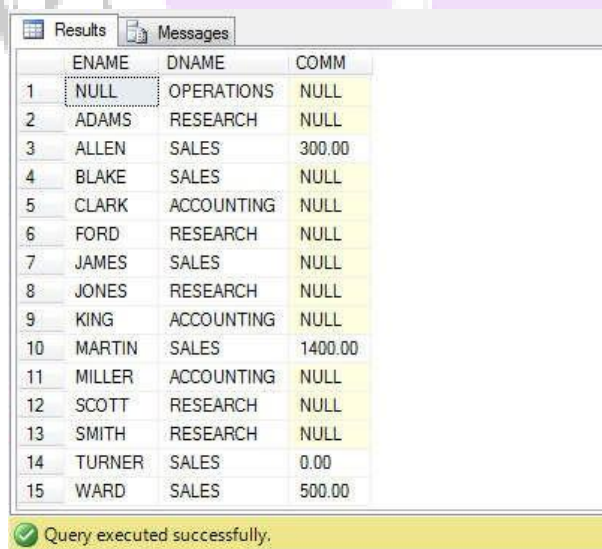
FULL JOIN sama dengan FULL OUTER JOIN akan mencocokkan dari kedua tabel baik dari tabel kiri ke tabel kanan maupun dari tabel kanan ke tabel kiri, baik yang mempunyai pasangan nilai maupun tidak.

Sintaks :

```
SELECT <SELECT list>
FROM <the table you want to be the 'LEFT' table>
FULL [OUTER]
JOIN < the table you want to be the 'RIGHT' table>
ON <JOIN CONDITION> [OTHER OPTION]
```

Contoh:

```
SELECT EMP. ENAME, DEPT. DNAME, EMP. COMM
FROM EMP FULL JOIN DEPT
ON EMP. DEPTNO = DEPT. DEPTNO
ORDER BY ENAME ASC
```



	ENAME	DNAME	COMM
1	NULL	OPERATIONS	NULL
2	ADAMS	RESEARCH	NULL
3	ALLEN	SALES	300.00
4	BLAKE	SALES	NULL
5	CLARK	ACCOUNTING	NULL
6	FORD	RESEARCH	NULL
7	JAMES	SALES	NULL
8	JONES	RESEARCH	NULL
9	KING	ACCOUNTING	NULL
10	MARTIN	SALES	1400.00
11	MILLER	ACCOUNTING	NULL
12	SCOTT	RESEARCH	NULL
13	SMITH	RESEARCH	NULL
14	TURNER	SALES	0.00
15	WARD	SALES	500.00

Query executed successfully.

6.5. CROSS JOIN

CROSS JOIN sangat berbeda dengan JOIN yang lain. CROSS JOIN akan menggabungkan setiap record data pada tabel yang kiri dengan seluruh isi data pada tabel kanan.

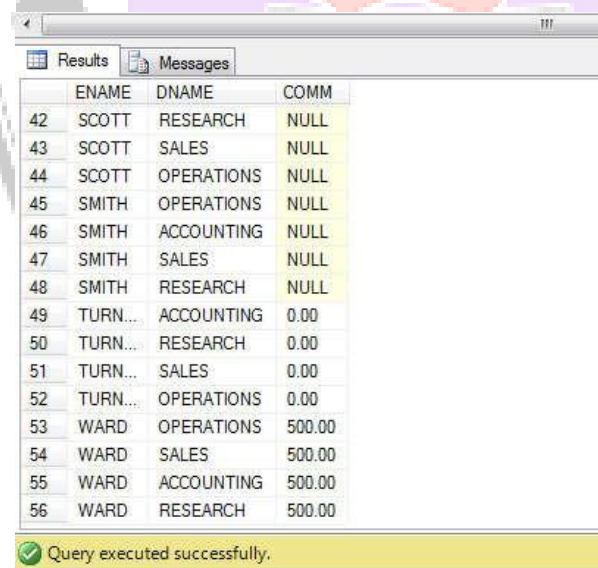
Sehingga nantinya dihasilkan setiap kemungkinan pasangan data dari tabel kiri dengan tabel kanan. Jika tabel di sebelah kiri berjumlah 10 record dan tabel kanan berjumlah 8 record, maka record yang dihasilkan dari perintah ini adalah 80 record. Pada clause 'CROSS JOIN' kita tidak menggunakan operator 'ON'.

Sintaks :

```
SELECT <select list>
FROM <the table you want to be the 'LEFT' table>
CROSS JOIN < the table you want to be the 'RIGHT' table>
```

Contoh:

```
SELECT EMP. ENAME, DEPT. DNAME, EMP. COMM
FROM EMP CROSS JOIN DEPT ORDER
BY EMP. ENAME ASC
```



	ENAME	DNAME	COMM
42	SCOTT	RESEARCH	NULL
43	SCOTT	SALES	NULL
44	SCOTT	OPERATIONS	NULL
45	SMITH	OPERATIONS	NULL
46	SMITH	ACCOUNTING	NULL
47	SMITH	SALES	NULL
48	SMITH	RESEARCH	NULL
49	TURN...	ACCOUNTING	0.00
50	TURN...	RESEARCH	0.00
51	TURN...	SALES	0.00
52	TURN...	OPERATIONS	0.00
53	WARD	OPERATIONS	500.00
54	WARD	SALES	500.00
55	WARD	ACCOUNTING	500.00
56	WARD	RESEARCH	500.00

6.6. SUBQUERY

Subquery adalah query biasa yang berada di dalam query lain. Jadi disini kita mengenal 2 istilah yaitu query luar dan query dalam yang dipisah dengan

tanda dalam kurung, dimana query dalam berfungsi sebagai basis kondisi sebagian data bagi query luar. Nilai yang dapat dikembalikan oleh query dalam bisa satu nilai (operator =) atau sekumpulan nilai (operator IN). Subquery juga dapat diartikan sebagai penggunaan perintah select didalam perintah select yang lain.

Kegunaan Subquery :

1. Untuk memecah perintah logika tunggal menjadi beberapa langkah logika yang berhubungan.
2. Untuk menyediakan daftar sebagai target dari Clausa WHERE yang digunakan bersama dengan [IN | EXIST | ANY | ALL].

Tingkat kerumitan Subquery tergantung pada kerumitan hubungan antara query dalam dengan query luar.

6.6.1. Jenis Sub Query

Jenis – Jenis Subquery :

- a. Nested Query / Query bersarang.
- b. Correlated Query.
- c. Derived Tables.

a. NESTED QUERY / QUERY BERSARANG.

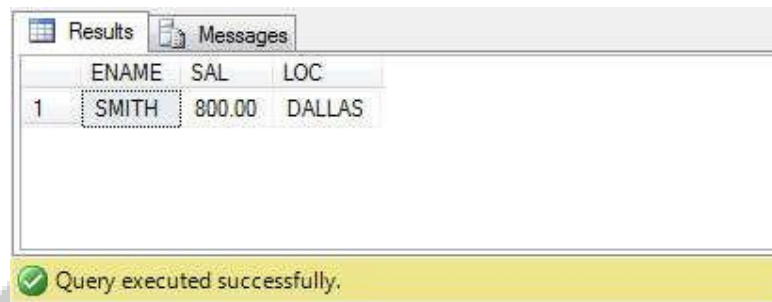
Adalah subquery dimana data hanya berjalan satu arah yaitu dari query dalam ke query luar saja.

Sintaks :

```
SELECT <select_list> FROM <sometable>
WHERE <some_column> = (
    SELECT <single_column> FROM <some_table>
    WHERE <condition that results in only one row returned>) atau :
SELECT <select_list> FROM <sometable>
WHERE <some_column> IN (
    SELECT <single_column> FROM <some_table>
    WHERE <condition>)
```

Contoh:

- ```
SELECT DISTINCT a. ENAME, a. SAL, b. LOC
FROM EMP a JOIN DEPT b
ON a. DEPTNO = b. DEPTNO
WHERE a. SAL = (SELECT MIN(SAL) from EMP)
```

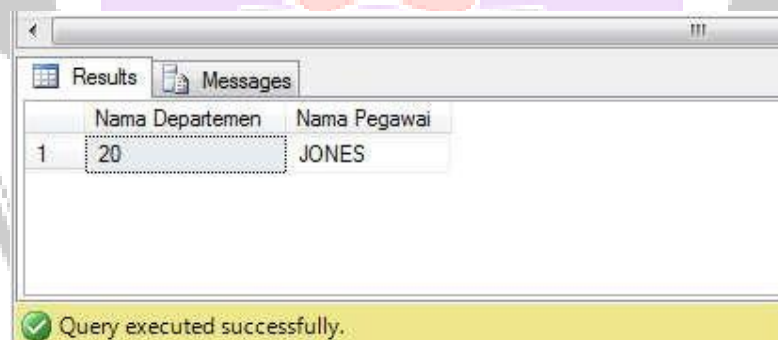


Results Messages

|   | ENAME | SAL    | LOC    |
|---|-------|--------|--------|
| 1 | SMITH | 800.00 | DALLAS |

Query executed successfully.

- ```
SELECT DEPTNO AS "Nama Departemen", ENAME AS "Nama
Pegawai"
FROM EMP
WHERE SAL = (SELECT SAL FROM EMP
WHERE EMPNO = 7566)
```



Results Messages

	Nama Departemen	Nama Pegawai
1	20	JONES

Query executed successfully.

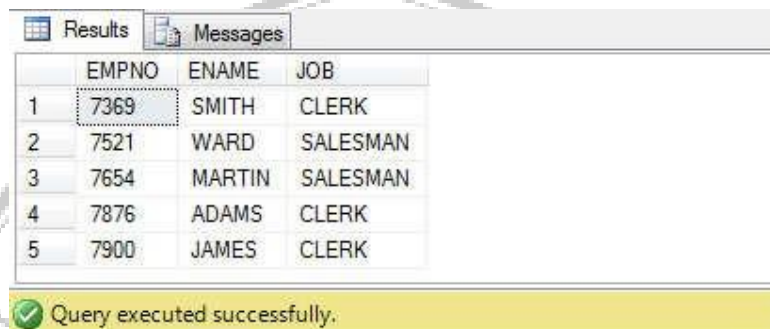
OPERATOR NOT, ANY, SOME, ALL.

- NOT
Operator ini berarti 'tidak' dan biasanya digunakan dengan kata IN.
- ANY, SOME.
ANY dan SOME adalah sama, perbedaannya adalah SOME adalah ANSICompliant sedang ANY tidak. Penggunaanya hampir sama dengan

Operator IN hanya saja kita dapat menggunakan beberapa kondisi dan operator lain seperti (\geq , \leq , $<$, $>$, $!$, etc).

Contoh :

```
SELECT EMPNO, ENAME, JOB
FROM EMP
WHERE SAL < ANY
(SELECT SAL FROM EMP WHERE JOB = 'CLERK')
```



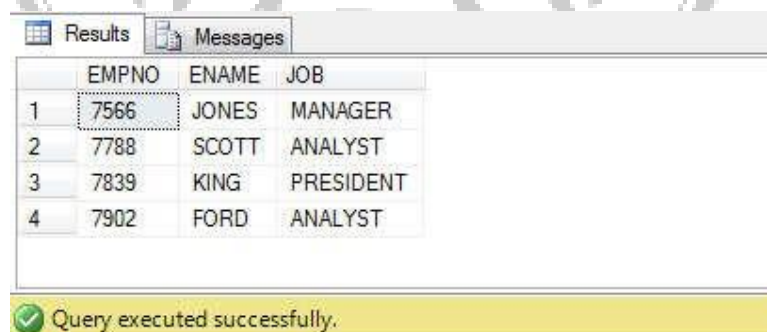
The screenshot shows a SQL query result window with two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with 5 rows and 3 columns: EMPNO, ENAME, and JOB. The first row is highlighted. Below the table, a yellow message bar indicates 'Query executed successfully.'

	EMPNO	ENAME	JOB
1	7369	SMITH	CLERK
2	7521	WARD	SALESMAN
3	7654	MARTIN	SALESMAN
4	7876	ADAMS	CLERK
5	7900	JAMES	CLERK

- ALL
Penggunaannya sama dengan SOME dan ANY.

Contoh :

```
SELECT EMPNO, ENAME, JOB
FROM EMP
WHERE SAL > ALL
(SELECT AVG(SAL) FROM EMP
GROUP BY DEPTNO)
```



The screenshot shows a SQL query result window with two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with 4 rows and 3 columns: EMPNO, ENAME, and JOB. The first row is highlighted. Below the table, a yellow message bar indicates 'Query executed successfully.'

	EMPNO	ENAME	JOB
1	7566	JONES	MANAGER
2	7788	SCOTT	ANALYST
3	7839	KING	PRESIDENT
4	7902	FORD	ANALYST

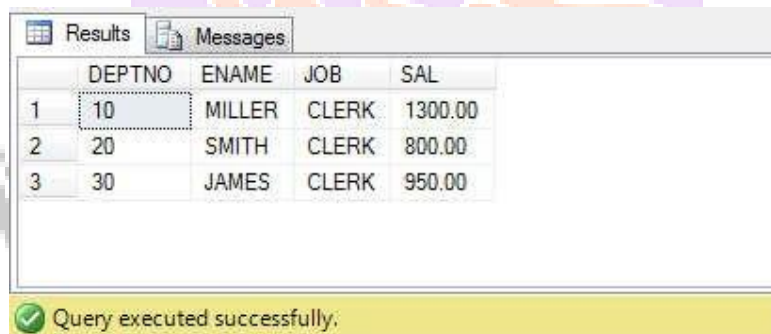
b. CORRELATED QUERY.

Adalah subquery dimana data berjalan dua arah yaitu dari query luar ke query dalam dan dari query dalam ke query luar. Pada Correlated query terjadi 3 langkah yaitu :

1. Query luar melewati parameter record yang berupa referensi ke query dalam.
2. Query dalam mengeksekusi berdasarkan nilai parameter record tadi.
3. Query dalam mengembalikan hasil ke query luar yang kemudian digunakan untuk menyelesaikan perintah.

Contoh:

```
SELECT a. DEPTNO, a. ENAME, a. JOB, a. SAL FROM EMP a
WHERE a. SAL = (SELECT MIN (b. SAL) FROM EMP b
WHERE a. DEPTNO = b. DEPTNO)
ORDER BY DEPTNO
```



	DEPTNO	ENAME	JOB	SAL
1	10	MILLER	CLERK	1300.00
2	20	SMITH	CLERK	800.00
3	30	JAMES	CLERK	950.00

c. DERRIVED TABLES.

Digunakan untuk melakukan query yang meliputi banyak tabel.

Sintaks :

```
SELECT <select_list>
FROM (<query that returns a regular result set>) AS <alias name> JOIN
<some other base or derived table>.
```

CONTOH:

```
SELECT DISTINCT c.CompanyName
FROM Customer AS c
```

```

JOIN (SELECT CustomerID FROM Orders o
      JOIN [Order Details] od ON o.OrderID = od.OrderID
      JOIN Products p ON od.ProductID = p.ProductID
      WHERE p.ProductName = 'Chocolade ') AS spen
ON c.CustomerID = spen.CustomerID

JOIN (SELECT CustomerID FROM Orders o
      JOIN [Order Details] od ON o.OrderID = od.OrderID
      JOIN Products p ON od.ProductID = p.ProductID
      WHERE p.ProductName = 'Vegie-spread ') AS spap
ON c.CustomerID = spap.CustomerID

```

Operator EXIST.

Fungsinya sama seperti keyword IN, tetapi ia tidak mengembalikan nilai data tetapi mengembalikan nilai TRUE/FALSE tergantung dari keberadaan record dalam list. Sehingga dalam hal performa ia lebih baik dibandingkan dengan operator IN.

Contoh:

```

SELECT CustomerID , CompanyName FROM Customer Cu
WHERE EXIST
      (SELECT OrderID FROM Orders o
       WHERE o.CustomerID = cu.CustomerID)

SELECT CustomerID , CompanyName FROM Customer Cu
WHERE NOT EXIST
      (SELECT OrderID FROM Orders o
       WHERE o.CustomerID = cu.CustomerID)

```

6.6.2. USER DEFINED FUNCTION

Adalah kumpulan statement T-SQL yang sudah dioptimasi dan dicompile dan dapat dipanggil sebagai satu unit tunggal yang dapat mengembalikan sebuah nilai atau tabel.

Sintaks :

```
CREATE FUNCTION [owner_name].function_name
([<@parameter name><scalar datatype> [=<default value>[,...N]]])
RETURN {<scalar type> | TABLE}
      WITH {ENCRYPTION | SCHEMABINDING}}
AS
BEGIN
[<function statements>]
{RETURN <type as defined in RETURN clause> | RETURN
  (<SELECT Statement>)}
```

Contoh:

```
CREATE FUNCTION EMP30 (@InputJob VARCHAR(50))
RETURNS table
AS
RETURN (SELECT ENAME, JOB, SAL FROM EMP
        WHERE JOB = @InputJob)
GO
```

Pada statement tersebut, dibuat sebuah function dengan nama EMP30, dimana ketika function dijalankan, akan menampilkan data yang diambil dari table EMP. Untuk menampilkan function dengan perintah :

```
SELECT * FROM EMP30 ('CLERK')
GO
```

Yang akan ditampilkan adalah data pegawai yang jenis pekerjaannya CLERK.

Results		Messages	
	ENAME	JOB	SAL
1	SMITH	CLERK	800.00
2	ADAMS	CLERK	1100.00
3	JAMES	CLERK	950.00
4	MILLER	CLERK	1300.00

Query executed successfully.

6.7. UNION

Operasi UNION dipergunakan untuk menggabungkan dua operasi selection menjadi satu Output. Perhatikan contoh berikut:

```


SELECT DEPTNO, ENAME, JOB, SAL
FROM EMP
WHERE DEPTNO = 10
UNION
SELECT DEPTNO, ENAME, JOB, SAL
FROM EMP
WHERE DEPTNO = 20

```

Results

Messages

	DEPTNO	ENAME	JOB	SAL
1	10	CLARK	MANAGER	2450.00
2	10	KING	PRESIDENT	5000.00
3	10	MILLER	CLERK	1300.00
4	20	ADAMS	CLERK	1100.00
5	20	FORD	ANALYST	3000.00
6	20	JONES	MANAGER	2975.00
7	20	SCOTT	ANALYST	3000.00
8	20	SMITH	CLERK	800.00

 Query executed successfully.