

## Pertemuan 13

### JOIN dan GROUP

**Tujuan Intruksional:**

Pokok bahasan ini mempelajari tentang fungsi JOIN dan GROUP dalam APEX

**Kompetensi Yang Diharapkan:**

Mahasiswa diharapkan memahami tentang fungsi JOIN dan GROUP dalam APEX

**Waktu Pertemuan :** 100 Menit

#### 9.1 JOIN

JOIN adalah salah satu fungsi untuk menghubungkan beberapa atribut dari tabel (Entitas) yang berbeda yang memiliki relasi.

Contoh :

Ingin menghubungkan entitas Departments dan Location.

Tabel Departments :

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting	-	1700

Tabel Locations :

LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	COUNTRY_ID
1800	460 Bloor St. W.	ON M5S 1X8	Toronto	Ontario	CA
2500	Magdalen Centre, The Oxford Science Park	OX9 9ZB	Oxford	Oxford	UK
1400	2014 Jabbenwocky Rd	26192	Southlake	Texas	US
1500	2011 Interiors Blvd	99236	South San Francisco	California	US
1700	2004 Charade Rd	98199	Seattle	Washington	US

#### A. NATURAL JOIN

Tidak memerlukan sesuatu yang secara spesifik untuk menggabungkan 2 tabel, jika tabel itu memiliki relasi maka akan dapat dihubungkan.

SQL :

```
SELECT department_name, city  
FROM departments NATURAL JOIN locations ;
```

Hasil :

DEPARTMENT_NAME	CITY
IT	Southlake
Shipping	South San Francisco
Administration	Seattle
Executive	Seattle
Accounting	Seattle
Contracting	Seattle
Marketing	Toronto
Sales	Oxford

## B. USING

USING digunakan untuk menentukan atribut apa yang akan digunakan untuk menggabungkan beberapa tabel.

Contoh :

```
SELECT last_name, job_title
```

```
FROM employees JOIN jobs USING (job_id)
```

Hasil :

LAST_NAME	JOB_TITLE
Gietz	Public Accountant
Higgins	Accounting Manager
Whalen	Administration Assistant
King	President
Kochhar	Administration Vice President
De Haan	Administration Vice President
Hunold	Programmer
Ernst	Programmer
Lorentz	Programmer
Hartstein	Marketing Manager
More than 10 rows available. Increase rows selector to view more rows.	

## C. JOIN ON dan PENGGUNAAN ALIAS

Contoh PENGGUNAAN ALIAS :

```
SELECT e.last_name, e.email FROM employees e ;
```

Penggunaan huruf "e" diatas adalah contoh meng-alias-kan entitas.

Contoh JOIN ON :

```
SELECT last_name, job_title  
FROM employees e JOIN jobs j  
ON (e.job_id = j.job_id);
```

Hasil :

LAST_NAME	JOB_TITLE
Gietz	Public Accountant
Higgins	Accounting Manager
Whalen	Administration Assistant
King	President
Kochhar	Administration Vice President
De Haan	Administration Vice President
Hunold	Programmer
Ernst	Programmer
Lorentz	Programmer
Hartstein	Marketing Manager
More than 10 rows available. Increase rows selector to view more rows.	

### C. LEFT OUTER JOIN

bagian paling kiri dari tabel yang akan menjadi referensi.

Contoh :

SQL :

```
SELECT e.last_name, d.department_id, d.department_name  
FROM employees e  
LEFT OUTER JOIN departments d ON (e.department_id = d.department_id);
```

Hasil :

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
Hartstein	20	Marketing
Vargas	50	Shipping
Matos	50	Shipping
Davies	50	Shipping
Rajs	50	Shipping
Mourgos	50	Shipping
Lorentz	60	IT
Ernst	60	IT
Hunold	60	IT
Taylor	80	Sales
Abel	80	Sales
Zlotkey	80	Sales
De Haan	90	Executive
Kochhar	90	Executive
King	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
Grant	-	-

#### D. RIGHT OUTER JOIN

Bagian kanan tabel yang akan menjadi referensi atau acuan.

Contoh :

SQL :

```
SELECT e.last_name, d.department_id, d.department_name
```

```
FROM employees e
```

```
RIGHT OUTER JOIN departments d ON (e.department_id = d.department_id);
```

Hasil :

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
Hartstein	20	Marketing
Rajs	50	Shipping
Vargas	50	Shipping
Mourgos	50	Shipping
Matos	50	Shipping
Davies	50	Shipping
Ernst	60	IT
Hunold	60	IT
Lorentz	60	IT
Taylor	80	Sales
Abel	80	Sales
Zlotkey	80	Sales
De Haan	90	Executive
King	90	Executive
Kochhar	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
-	190	Contracting

20 rows returned in 0.01 seconds [Download](#)

E. FULL OUTER JOIN :

FULL OUTER JOIN adalah gabungan dari LEFT OUTER JOIN dan RIGHT OUTER JOIN.

Contoh :

SQL :

SELECT e.last\_name, d.department\_id, d.department\_name

FROM employees e

FULL OUTER JOIN departments d ON (e.department\_id = d.department\_id);

Hasil :

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
King	90	Executive
Kochhar	90	Executive
De Haan	90	Executive
Whalen	10	Administration
Higgins	110	Accounting
Gietz	110	Accounting
Zlotkey	80	Sales
Abel	80	Sales
Taylor	80	Sales
Grant	-	-
Mourgos	50	Shipping
Rajs	50	Shipping
Davies	50	Shipping
Matos	50	Shipping
Vargas	50	Shipping
Hunold	60	IT
Ernst	60	IT
Lorentz	60	IT
Hartstein	20	Marketing
Fay	20	Marketing
-	190	Contracting

21 rows returned in 0.00 seconds [Download](#)

## F. INNER JOIN

INNER JOIN adalah penggabungan data yang dimunculkan hanya data yang terdapat pada tabel - tabel yang dihubungkan.

Contoh :

SQL :

```
SELECT e.last_name, d.department_id, d.department_name
```

```
FROM employees e
```

```
INNER JOIN departments d ON (e.department_id = d.department_id);
```

Hasil :

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Hartstein	20	Marketing
Fay	20	Marketing
Mourgos	50	Shipping
Rajs	50	Shipping
Davies	50	Shipping
Matos	50	Shipping
Vargas	50	Shipping
Hunold	60	IT
Ernst	60	IT
Lorentz	60	IT
Zlotkey	80	Sales
Abel	80	Sales
Taylor	80	Sales
King	90	Executive
Kochhar	90	Executive
De Haan	90	Executive
Higgins	110	Accounting
Gietz	110	Accounting

19 rows returned in 0.00 seconds   [Download](#)

## 9.2 GROUP

- **Group By Clause**

Group By merupakan kalusa yang digunakan untuk menampilkan sekumpulan data pada tabel berdasarkan kelompok tertentu.

Query:

```
SELECT department_id, AVG(salary)
FROM employees
GROUP BY department_id
ORDER BY department_id;
```



Output:

[illegible]

Query pada contoh diatas menampilkan data 'department\_id', dan rata-rata 'salary' dari tabel 'employees', dimana baris data dikelompokkan berdasarkan 'department\_id' kemudian dihitung rata-rata dari 'salary' dan diurutkan berdasarkan 'department\_id'. Sehingga pada tiap baris misalnya baris pertama menampilkan karyawan yang bekerja pada department 10 memiliki rata-rata gaji sebesar 4400.

- **Group Withn Group**

Group Withn Group merupakan istilah yang digunakan untuk pengelompokan bersarang.

Query:

```
SELECT department_id, job_id, count(*)
FROM employees
WHERE department_id > 40
GROUP BY department_id, job_id
ORDER BY department_id;
```

Output:

DEPARTMENT_ID	JOB_ID	COUNT(*)
50	ST_CLERK	4
50	ST_MAN	1
60	IT_PROG	3
80	SA_MAN	1
80	SA_REP	2
90	AD PRES	1
90	AD VP	2
110	AC_ACCOUNT	1
110	AC MGR	1

Query pada contoh diatas menampilkan data karyawan yang dikelompokkan berdasarkan 'department\_id', lalu di dalam masing-masing departemen, dilakukan pengelompokan lagi berdasarkan 'job\_id' kemudian menampilkan jumlah karyawan yang bekerja pada setiap pekerjaan di setiap departemen.

- **Having Clause**

Klausula Having digunakan untuk menyeleksi data berdasarkan kriteria tertentu, kriteria tersebut menentukan kondisi bagi Group By. Kelompok data yang memenuhi klausula Having saja yang akan ditampilkan.

Query:

```
SELECT department_id, AVG(salary), Count(department_id)
FROM employees
HAVING AVG(salary)>7000
GROUP BY department_id
ORDER BY department_id;
```

Output:

DEPARTMENT_ID	AVG(SALARY)	COUNT(DEPARTMENT_ID)
20	9500	2
80	10033.3333333333333333333333333333	3
90	19333.3333333333333333333333333333	3
110	10150	2

Query pada contoh diatas menampilkan hasil pengelompokan data berdasarkan 'department\_id' kemudian menampilkan hasil pengelompokan data yang memiliki rata-rata 'salary' lebih dari 7000.

- **ROLLUP**

ROLLUP digunakan untuk menghitung subtotal dan total dari setiap kelompok data.

Syarat penggunaannya adalah

- adanya tipe data numerik yang akan dihitung.
- kumpulan data yang sejenis dalam kumpulan data.

Query:

```
SELECT department_id, job_id, SUM(salary)
FROM employees
WHERE department_id < 50
GROUP BY ROLLUP (department_id, job_id);
```

Output:

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
10	-	4400
20	MK_MAN	13000
20	MK_REP	6000
20	-	19000
-	-	23400

Menampilkan total dari hasil pengelompokan departement\_id yang memiliki nilai 10 (subtotal dari department\_id)

Menampilkan total dari hasil pengelompokan department\_id yang memiliki nilai 20 (subtotal dari department\_id)

Menampilkan total dari hasil seluruh pengelompokan department\_id

- **CUBE**

CUBE memiliki fungsi yang hampir sama dengan ROLLUP yaitu menghitung total dan sub total, namun CUBE digunakan untuk membuat cross-tabulasi(sub-total lebih dari satu dimensi).

Query:

```
SELECT department_id, job_id, SUM(salary)
FROM employees
WHERE department_id < 50
GROUP BY CUBE (department_id, job_id);
```

Output:

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
-	-	23400
-	MK_MAN	13000
-	MK_REP	6000
-	AD_ASST	4400
10	-	4400
10	AD_ASST	4400
20	-	19000
20	MK_MAN	13000
20	MK_REP	6000

Total dari seluruh hasil pengelompokan

menampilkan subtotal dari dimensi job\_id dengan nilai 'MK\_MAN'

menampilkan subtotal dari dimensi job\_id dengan nilai 'MK\_REP'

menampilkan subtotal dari dimensi job\_id dengan nilai 'AD\_ASST'

menampilkan subtotal dari dimensi department\_id dengan nilai 10

menampilkan subtotal dari dimensi department\_id dengan nilai 20

- **Grouping Set**

Grouping Sets adalah perluasan dari klausa Group By. Grouping Sets digunakan untuk mendefinisikan pengelompokan lebih dari satu dalam query yang sama.

Query:

```
SELECT department_id, job_id, manager_id, SUM(salary)
```

```
FROM employees
```

```
WHERE department_id < 50
```

```
GROUP BY GROUPING SETS
```

```
((job_id, manager_id), (department_id, job_id), (department_id, manager_id));
```

Output:

DEPARTMENT_ID	JOB_ID	MANAGER_ID	SUM(SALARY)
-	MK_MAN	100	13000
-	AD_ASST	101	4400
-	MK_REP	201	6000
10	AD_ASST	-	4400
20	MK_MAN	-	13000
20	MK_REP	-	6000
10	-	101	4400
20	-	100	13000
20	-	201	6000

- GRUPING Function

Fungsi GRUPING digunakan untuk mengetahui baris mana yang menampilkan data dari hasil dari perhitungan, dan baris mana yang menampilkan data sebenarnya dari database, penggunaan fungsi GRUPING akan mengembalikan nilai 0 atau 1, di mana:

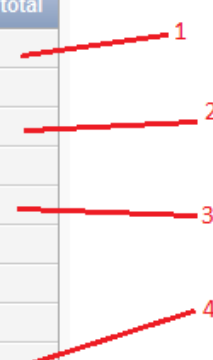
- Nilai 0 yang dihasilkan menandakan :
  - Atribut telah digunakan untuk menghitung nilai agregat
  - Nilai NULL yang ada pada kolom adalah *stored* NULL.
- Nilai 1 yang dihasilkan menandakan :
  - Atribut tidak pernah digunakan untuk menghitung nilai agregat
  - Nilai NULL yang ada pada kolom adalah nilai NULL yang dihasilkan dari penggunaan operator ROLLUP atau CUBE sebagai hasil dari pengelompokan data.

Query:

```
SELECT department_id, job_id, SUM(salary),  
       GROUPING(department_id) AS "Dept sub total",  
       GROUPING(job_id) AS "Job sub total"  
FROM employees  
WHERE department_id < 50  
GROUP BY CUBE (department_id, job_id);
```

Output:

DEPARTMENT_ID	JOB_ID	SUM(SALARY)	Dept sub total	Job sub total
-	-	23400	1	1
-	MK_MAN	13000	1	0
-	MK_REP	6000	1	0
-	AD_ASST	4400	1	0
10	-	4400	0	1
10	AD_ASST	4400	0	0
20	-	19000	0	1
20	MK_MAN	13000	0	0
20	MK_REP	6000	0	0



1. Nilai 23400 menunjukkan total gaji dari seluruh karyawan yang memenuhi syarat klusa WHERE. Di sini sudah tidak digunakan apa jenis pekerjaannya atau kode departmentnya. Karena kedua kolom, yaitu 'department\_id' dan 'job\_id' tidak digunakan dan nilai NULL yang ada pada kolom 'department\_id' dan 'job\_id' merupakan hasil dari penggunaan operator CUBE, maka nilai yang dihasilkan dari fungsi GROUPING adalah 1 di kolom 'Dept sub total' dan 'Job sub total'.
2. Nilai 6000 pada kolom 'salary' menunjukkan total gaji seluruh karyawan yang bekerja sebagai 'MK\_REP' (tidak menggunakan department\_id). Kolom yang digunakan adalah kolom 'job\_id'. Karena itu, nilai kembalian yang dihasilkan pada kolom 'Job sub total' adalah 0, sedangkan kolom 'Dept sub total' berisi nilai 1, karena kolom 'department\_id' tidak digunakan untuk menghitung total gaji ini.
3. Nilai 4400 pada kolom 'salary' menunjukkan total gaji seluruh karyawan yang berada di departemen 10 (tidak menggunakan job\_id). Kolom yang digunakan adalah kolom 'department\_id'. Karena itu, nilai kembalian yang dihasilkan pada kolom 'Dept sub total' adalah 0, sedangkan kolom 'Job sub total' berisi nilai 1, karena kolom 'job\_id' tidak digunakan untuk menghitung total gaji ini.
4. Nilai 6000 pada kolom 'salary' menunjukkan total gaji karyawan yang bekerja sebagai 'MK\_REP' pada department 20. Kolom yang digunakan adalah kolom 'department\_id' dan 'job\_id'. Karena itu, nilai kembalian yang dihasilkan pada kolom 'Dept sub total' dan 'Job sub total' adalah 0.

## LATIHAN SOAL JOIN

1. Buatkan perintah SQL untuk memunculkan last\_name dari tabel (entitas) employees , city dari tabel (entitas) locations, menggunakan NATURAL JOIN !
2. Buatkan perintah SQL untuk tampilan dibawah ini menggunakan :

A. USING

B. JOIN ON

Clue :

- department\_name ada di entitas DEPARTMENTS

- salary ada di entitas EMPLOYEES

DEPARTMENT_NAME	SALARY
Administration	4400
Marketing	13000
Marketing	6000
Shipping	5800
Shipping	3500
Shipping	3100
Shipping	2600
Shipping	2500
IT	9000
IT	6000
More than 10 rows available. Increase rows selector to view more rows.	
10 rows returned in 0.01 seconds <a href="#">Download</a>	

3. Buatlah tampilan seperti dibawah ini dengan mengimplementasikan salah satu fungsi INNER JOIN, LEFT OUTER JOIN, FULL OUTER JOIN, RIGHT OUTER JOIN !

A.

LAST_NAME	SALARY	DEPARTMENT_NAME
King	24000	Executive
Kochhar	17000	Executive
De Haan	17000	Executive
Whalen	4400	Administration
Higgins	12000	Accounting
Gietz	8300	Accounting
Zlotkey	10500	Sales
Abel	11000	Sales
Taylor	8600	Sales
Grant	7000	-
Mourgos	5800	Shipping
Rajs	3500	Shipping
Davies	3100	Shipping
Matos	2600	Shipping
Vargas	2500	Shipping
Hunold	9000	IT
Ernst	6000	IT
Lorentz	4200	IT
Hartstein	13000	Marketing
Fay	6000	Marketing
-	-	Contracting
21 rows returned in 0.00 seconds <a href="#">Download</a>		



B.

LAST_NAME	SALARY	DEPARTMENT_NAME
Whalen	4400	Administration
Fay	6000	Marketing
Hartstein	13000	Marketing
Rajs	3500	Shipping
Vargas	2500	Shipping
Mourgos	5800	Shipping
Matos	2600	Shipping
Davies	3100	Shipping
Ernst	6000	IT
Hunold	9000	IT
Lorentz	4200	IT
Taylor	8600	Sales
Abel	11000	Sales
Zlotkey	10500	Sales
De Haan	17000	Executive
King	24000	Executive
Kochhar	17000	Executive
Gietz	8300	Accounting
Higgins	12000	Accounting
-	-	Contracting

20 rows returned in 0.00 seconds [Download](#)

C.

LAST_NAME	SALARY	DEPARTMENT_NAME
Whalen	4400	Administration
Fay	6000	Marketing
Hartstein	13000	Marketing
Vargas	2500	Shipping
Matos	2600	Shipping
Davies	3100	Shipping
Rajs	3500	Shipping
Mourgos	5800	Shipping
Lorentz	4200	IT
Ernst	6000	IT
Hunold	9000	IT
Taylor	8600	Sales
Abel	11000	Sales
Zlotkey	10500	Sales
De Haan	17000	Executive
Kochhar	17000	Executive
King	24000	Executive
Gietz	8300	Accounting
Higgins	12000	Accounting
Grant	7000	-

20 rows returned in 0.01 seconds [Download](#)

D.

LAST_NAME	SALARY	DEPARTMENT_NAME
Whalen	4400	Administration
Hartstein	13000	Marketing
Fay	6000	Marketing
Mourgos	5800	Shipping
Rajs	3500	Shipping
Davies	3100	Shipping
Matos	2600	Shipping
Vargas	2500	Shipping
Hunold	9000	IT
Ernst	6000	IT
Lorentz	4200	IT
Zlotkey	10500	Sales
Abel	11000	Sales
Taylor	8600	Sales
King	24000	Executive
Kochhar	17000	Executive
De Haan	17000	Executive
Higgins	12000	Accounting
Gietz	8300	Accounting

19 rows returned in 0.01 seconds [Download](#)

4. Tulisan SQL untuk gambar dibawah ini menggunakan ON dan ALIAS

Clue :

- city dari entitas locations

19 rows returned in 0.02 seconds [Download](#)

## SOAL MATERI GROUPING

1. Tuliskan query yang dapat menampilkan manager\_id, jumlah karyawan dan rata-rata gaji yang diterima oleh suatu kelompok karyawan yang dikelompokkan berdasarkan manager\_id, dan diurutkan berdasarkan rata rata gaji yang terbesar ke yang terkecil.

MANAGER_ID	Number of Employees	AVG(SALARY)
-	1	24000
100	5	12660
102	1	9000
149	3	8866.6666666666666666666666666667
205	1	8300
101	2	8200
201	1	6000
103	2	5100
124	4	2925

- [illegible]

- | JOB_ID   | Number Of Employees | AVG(SALARY) |
|----------|---------------------|-------------|
| AD_ASST  | 1                   | 4400        |
| IT_PROG  | 3                   | 6400        |
| MK_REP   | 1                   | 6000        |
| ST_CLERK | 4                   | 2925        |
| ST_MAN   | 1                   | 5800        |

4. Tuliskan query yang dapat menampilkan data department\_id, manager\_id, dan total gaji yang dikelompokkan berdasarkan manager\_id yang berada dalam suatu kelompok department\_id dengan kondisi department\_id dibawah 60 dan tampilkan subtotal dan total dari gaji yang diterima pada tiap tiap kelompok data serta keterangan apakah baris tersebut menggunakan perhitungan dari kelompok manager\_id atau department\_id.

DEPARTMENT_ID	MANAGER_ID	total salary	GROUPING(DEPARTMENT_ID)	GROUPING(MANAGER_ID)
10	101	4400	0	0
10	-	4400	0	1
20	100	13000	0	0
20	201	6000	0	0
20	-	19000	0	1
50	100	5800	0	0
50	124	11700	0	0
50	-	17500	0	1
-	-	40900	1	1

5. Tuliskan query yang dapat menampilkan data department\_id, manager\_id, dan total gaji yang dikelompokkan berdasarkan manager\_id yang berada dalam suatu kelompok department\_id dengan kondisi department\_id dibawah 50 dan tampilkan subtotal dan total dari gaji berdasarkan dimensi kelompok department\_id dan dimensi kelompok manager\_id serta keterangan apakah baris tersebut menggunakan perhitungan dari kelompok manager\_id atau department\_id.

DEPARTMENT_ID	MANAGER_ID	total salary	GROUPING(DEPARTMENT_ID)	GROUPING(MANAGER_ID)
-	-	23400	1	1
-	100	13000	1	0
-	101	4400	1	0
-	201	6000	1	0
10	-	4400	0	1
10	101	4400	0	0
20	-	19000	0	1
20	100	13000	0	0
20	201	6000	0	0