

# **Oracle Academic Initiative**

## **Oracle9i Introduction to SQL**



**Oleh:**

**Tessy Badriyah, SKom.MT**

**Politeknik Elektronika Negeri Surabaya  
Institut Teknologi Sepuluh Nopember  
Surabaya**

## BAB 17 : Perbaikan dengan Klausa GROUP BY

### 17.1. Sasaran

- Menggunakan operasi ROLLUP untuk menghasilkan nilai sub total
- Menggunakan operasi CUBE untuk menghasilkan nilai *cross-tabulation*
- Menggunakan fungsi GROUPING untuk mengidentifikasi nilai baris yang dibuat oleh ROLLUP atau CUBE
- Menggunakan GROUPING SETS untuk menghasikan himpunan hasil tunggal

### 17.2. Review : Fungsi Group

Fungsi GROUP beroperasi pada himpunan yang baris-barisnya memberi satu hasil per group.

```
SELECT      [column,] group_function(column) . . .
FROM        table
[WHERE      condition]
[GROUP BY  group_by_expression]
[ORDER BY  column];
```

Contoh :

```
SELECT AVG(salary), STDDEV(salary),
        COUNT(commission_pct), MAX(hire_date)
FROM    employees
WHERE   job_id LIKE 'SA%';
```

### 17.3. Review : Klausa GROUP BY

Sintak dari klausa GROUP BY :

```
SELECT      [column,] group_function(column) . . .
FROM        table
[WHERE      condition]
[GROUP BY  group_by_expression]
[ORDER BY  column];
```

Contoh :

```
SELECT  department_id, job_id, SUM(salary),
        COUNT(employee_id)
FROM    employees
GROUP BY department_id, job_id ;
```

### 17.4. Review : Klausa HAVING

Klausa HAVING digunakan untuk menentukan group mana yang akan ditampilkan. Lebih lanjut kita bisa membatasi group berdasarkan suatu kondisi tertentu.

```
SELECT      [column,] group_function(column) . . .
FROM        table
[WHERE      condition]
[GROUP BY  group_by_expression]
[HAVING   having_expression]
[ORDER BY  column];
```

### 17.5. GROUP BY dengan Operator ROLLUP dan CUBE

ROLLUP atau CUBE digunakan bersama dengan GROUP BY untuk menghasilkan baris superaggregate melalui kolom referensi silang (*cross-referencing columns*).

Pengelompokkan ROLLUP menghasilkan himpunan hasil yang berisi pengelompokan baris dan nilai sub total. Pengelompokan CUBE menghasilkan himpunan hasil yang berisi baris-baris dari ROLLUP dan baris cross-tabulation.

### 17.6. Operator ROLLUP

ROLLUP adalah perluasan terhadap klausa GROUP BY. Operasi ini digunakan untuk menghasilkan aggregate terakumulasi, semisal sub total.

Sintaknya :

```
SELECT      [column,] group_function(column) . . .
FROM        table
[WHERE      condition]
[GROUP BY  [ROLLUP] group_by_expression]
[HAVING    having_expression];
[ORDER BY  column];
```

### 17.7. Contoh Operator ROLLUP

```
SELECT  department_id, job_id, SUM(salary)
FROM    employees
WHERE   department_id < 60
GROUP BY ROLLUP (department_id, job_id);
```

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
10		4400
20	MK_MAN	13000
20	MK_REP	6000
20		19000
50	ST_CLERK	11700
50	ST_MAN	5800
50		17500
		40900

9 rows selected.

### 17.8. Operator Cube

CUBE adalah perluasan dari klausa GROUP BY. Operator CUBE digunakan untuk menghasilkan nilai *cross-tabulation* dengan statement SELECT tunggal.

```
SELECT      [column,] group_function(column)...
FROM        table
[WHERE      condition]
[GROUP BY  [CUBE] group_by_expression]
[HAVING    having_expression]
[ORDER BY  column];
```

17.9. Contoh Operator Cube

```
SELECT department_id, job_id, SUM(salary)
FROM employees
WHERE department_id < 60
GROUP BY CUBE (department_id, job_id) ;
```

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
10		4400
20	MK_MAN	13000
20	MK_REP	6000
20		19000
50	ST_CLERK	11700
50	ST_MAN	5800
50		17500
	AD_ASST	4400
	MK_MAN	13000
	MK_REP	6000
	ST_CLERK	11700
	ST_MAN	5800
		40900

14 rows selected.

17.10. Fungsi GROUPING

Fungsi GROUPING dapat digunakan baik untuk operator CUBE maupun ROLLUP. Dengan menggunakan fungsi GROUPING, kita dapat mencari sub total dari tiap pengelompokan. Dengan perintah tersebut kita juga bisa membedakan nilai NULL yang disimpan dari atau dibuat oleh ROLLUP atau CUBE. Fungsi GROUPING akan mengembalikan 0 atau 1.

```
SELECT [column,] group_function(column) . ,
GROUPING(expr)
FROM table
[WHERE condition]
[GROUP BY [ROLLUP][CUBE] group_by_expression]
[HAVING having_expression]
[ORDER BY column];
```

17.11. Contoh Fungsi GROUPING

```
SELECT department_id DEPTID, job_id JOB,
SUM(salary),
GROUPING(department_id) GRP_DEPT,
GROUPING(job_id) GRP_JOB
FROM employees
WHERE department_id < 50
GROUP BY ROLLUP (department_id, job_id);
```

DEPTID	JOB	SUM(SALARY)	GRP_DEPT	GRP_JOB
10	AD_ASST	4400	0	0
10		4400	0	1
20	MK_MAN	13000	0	0
20	MK_REP	6000	0	0
20		19000	0	1
		23400	1	1

6 rows selected.

### 17.12. GROUPING SETS

GROUPING SETS adalah perluasan dari klausa GROUP BY. GROUPING SETS digunakan untuk mendefinisikan pengelompokan lebih dari satu dalam query yang sama. Oracle server melakukan komputasi untuk semua pengelompokan yang ditentukan dalam klausa GROUPING SETS dan menggabungkan hasilnya dari pengelompokan individual dengan operasi UNION ALL.

Himpunan pengelompokan memiliki efisiensi untuk :

- Hanya sekali melewati base table jika diperlukan
- Tidak perlu menulis complex UNION statements
- Ada banyak elemen yang dimiliki oleh GROUPING SETS, tapi yang paling utama adalah performansi atau unjuk kerjanya.

### 17.13. Contoh GROUPING SETS

```
SELECT  department_id, job_id,
        manager_id, avg (salary)
FROM    employees
GROUP BY GROUPING SETS
        ((department_id, job_id), (job_id, manager_id));
```

DEPARTMENT_ID	JOB_ID	MANAGER_ID	AVG(SALARY)
10	AD_ASST		4400
20	MK_MAN		13000
20	MK_REP		6000
50	ST_CLERK		2925
...			
	SA_MAN	100	10500
	SA_REP	149	8866.66667
	ST_CLERK	124	2925
	ST_MAN	100	5800

26 rows selected.

### 17.14. Kolom Komposit

Kolom komposit adalah kumpulan kolom yang berlaku sebagai unit.

ROLLUP (a, (b, c), d)

Untuk menentukan kolom komposit digunakan klausa GROUP BY untuk mengelompokkan kolom-kolom dengan tanda kurung sehingga Oracle Server memberlakukan mereka sebagai unit pada saat melakukan komputasi operasi ROLLUP atau CUBE.

### 17.15. Contoh Kolom Komposit

```
SELECT  department_id, job_id, manager_id,
        SUM(salary)
FROM    employees
GROUP BY ROLLUP ( department_id, (job_id, manager_id));
```

DEPARTMENT_ID	JOB_ID	MANAGER_ID	SUM(SALARY)
10	AD_ASST	101	4400
10			4400
20	MK_MAN	100	13000
20	MK_REP	201	6000
20			19000
50	ST_CLERK	124	11700
...			
			175500

23 rows selected.

### 17.16. Concatenated Grouping

*Concatenated groupings* menyediakan cara yang pasti untuk mengenerate kombinasi pengelompokan yang berguna. Untuk menentukan himpunan *concatenated grouping*, kita pisahkan beberapa himpunan pengelompokan, operasi ROLLUP dan CUBE dengan tanda koma sehingga Oracle Server mengkombinasikannya ke dalam klausa GROUP BY tunggal. Hasilnya adalah cross-product dari pengelompokan untuk setiap himpunan pengelompokan.

### 17.17. Contoh Concatenated Grouping

```
SELECT  department_id, job_id, manager_id,
        SUM(salary)
FROM    employees
GROUP BY department_id,
        ROLLUP (job_id),
        CUBE (manager_id);
```

DEPARTMENT_ID	JOB_ID	MANAGER_ID	SUM(SALARY)
10	AD_ASST	101	4400
20	MK_MAN	100	13000
...			
10		101	4400
20		100	13000
...			
10	AD_ASST		4400
10			4400
...			
	SA_REP		7000
			7000

49 rows selected.

### 17.18. Latihan