Grouping_id 추가 실습

7. 부서번호와 job번호로 그룹을 지어 salary의 최대값을 구하라. Job별 salary의 최대값, 부서별 salary의 최대값, 전체 salary의 최대값도 출력되어야 한다. 원래 NULL인 경우와 grouping을 해서 NULL로 표기되는 경우를 구별하기 위하여 grouping_id(컬럼이름)컬럼을 두개 추가하라. 즉, 총 5개의 컬럼이 출력되어야 한다.

```
select grouping_id(department_id), grouping_id(Job_id), department_id, job_id, Max(Salary) "Max_Salary"
from employees
group by Cube(department_id, job_id)
order by 1, 2, 3;
```

이처럼 작성을 다시 해보게 되면 grouping_ID에 관련해서 그 값이 원래 있거나 원래 Null인 경우에는 0을 출력하고, 컬럼별 통계를 내는 경우에는 1를 출력하게 되는 것이다. 그래서 실질적으로는 다음과 같이 참조할 수 있다.

(Grouping_id(department_id)), (Grouping_id(job_id))

0, 0 : 부서 내의 직급 별 봉급 최댓값을 출력해댄다. (물론 부서가 null인 새끼도 포함이 된다.

0, 1 : 부서별 봉급 최댓값을 출력한다.

1, 0 : 직급 별 봉급 최댓값을 출력한다.

1, 1 : 전체 봉급 최댓값을 출력한다.

8. 위의 문제를 rollup을 이용하여 질의를 다시 작성하라. 질의 결과에 어떤 차이가 있나?

```
select grouping_id(department_id), grouping_id(Job_id), department_id, job_id, Max(Salary) "Max_Salary"
from employees
group by Rollup(department_id, job_id)
order by 1, 2, 3;
```

이처럼 작성하게 된다면 방금 cube 함수를 이용해서 모든 컬럼별로 4가지의 경우대로 출력을 했지만, 애석하게도 Rollup 함수는 job_id에 있는 통계에 관련되어서는 씹고 넘어간다. 그래서 결국 출력되는 녀석들은 다음과 같다.

(Grouping_id(department_id)), (Grouping_id(job_id))

0, 0 : 부서 내의 직급 별 봉급 최댓값을 출력해댄다. (물론 부서가 null인 새끼도 포함이 된다.

0, 1 : 부서별 봉급 최댓값을 출력한다.

~~1, 0 : 어건 그냥 씹고 넘긴다.~~

1, 1 : 전체 봉급 최댓값을 출력한다.

9. 위의 문제를 grouping sets를 이용하여 질의를 다시 작성하라. 질의 결과에 어떤 차이가 있나?

```
select grouping_id(department_id), grouping_id(Job_id), department_id, job_id, Max(Salary) "Max_Salary"
from employees
group by Grouping Sets(department_id, job_id)
order by 1, 2, 3;
```

이처럼 작성하게 된다면 각각 컬럼별 통계만 내는 역할을 한다. 그래서 다음과 같은 결과가 출력이 된다.

(Grouping_id(department_id)), (Grouping_id(job_id))

~~0, 0 : 어건 그냥 씹고 넘긴다.~~

0, 1 : 부서별 봉급 최댓값을 출력한다.

1, 0 : 직급별 봉급 최댓값을 출력한다.

~~1, 1 : 어것도 그냥 씹고 넘긴다.~~

75페이지 문장 실습해보기

1. Cube를 이용하면 다음과 같은 결과로서 나오게 된다.

```
select grouping_id(department_id), grouping_id(job_id), department_id, job_id, count(*), Sum(Salary)
from employees
Group by Cube(department_id, job_id)
order by 1, 2, 3, 4;
```

grouping_id(department_id), grouping_id(job_id)

0, 0 : 부서 내 직급 별 인원수와 봉급 총합이 각각 나오게 된다. 물론 부서에 속하지 않은 녀석도 나오니 참고하자.

0, 1 : 부서별 봉급 총합과 인원 수가 나오게 된다. 부서에 속하지 않은 녀석은 1, 7000이 나오는 걸 알 수 있을 것이다.

1, 0 : 직급별 봉급 총합과 인원 수가 나오게 된다.

1, 1 : 총 인원수와 봉급의 총합이 나오게 된다.

2. Rollup 함수를 이용해보기

```
select grouping_id(department_id), grouping_id(job_id), department_id, job_id, count(*), Sum(Salary)
from employees
Group by Rollup(department_id, job_id)
order by 1, 2, 3, 4;
```

방금 전처럼 직급별 인원과 봉급 총합에 대해서는 제외하고 출력하게 된다.

3. Grouping Sets 함수를 이용해 보기

```
select grouping_id(department_id), grouping_id(job_id), department_id, job_id, count(*), Sum(Salary)
```

```
from employees
Group by Rollup(department_id, job_id)
order by 1, 2, 3, 4;
```
이 문장에 대해서는 직책 별, 부서 별만 출력이 된다.

<참조> Group By가 3개 이상인 경우에는 어떻게 결과가 나오게 될까요?
이번에는 부서별, 직책별, 봉급 만원 별(3개의 컬럼을 기준)로 직원들의 수에 대해 출력을 해 볼 것이다. 다음 문장들을 참조하자.
1) Cube 함수
이처럼 작성하게 되면 8가지의 경우로 나뉘어서 출력이 된다. 그래서 출력 되는 경우는 모두 출력된다고 생각하면 된다.
2) Rollup 함수
이처럼 작성하게 되면 부서별로 중점이 된다면, 부서 내 직책 내 봉급 별(000), 부서 내 직책 별(001), 부서 별(011), 전체(111)를 참조하게 된다. 뒤에서부터 1로서 말아 올린다고 생각하면 쉽게 이해 된다.
3) Grouping Sets 함수
이는 각 컬럼 별로만 출력하게 되어 3개의 컬럼 각각만 참조하게 된다.