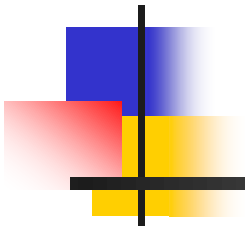
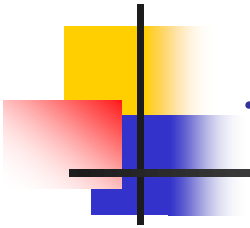


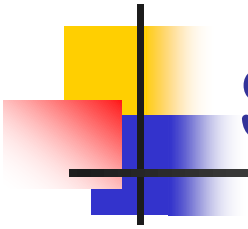
Oracle 3강 – SQL 복수 행 함수(그룹함수)

양 명 숙
[now4ever7@gmail.com]



목차

- Group 함수의 종류
- 특정 조건으로 세부적인 그룹화 하기(Group By절 사용)
- 조건을 주고 검색하기(Having 절 사용)
- 자동으로 소계/합계를 구해주는 함수
- 다른 그룹핑 관련 함수들



SQL 복수행 함수(그룹 함수)

- Group 함수의 종류
  - SQL 복수행 함수 – 단일 행 함수와 달리 한꺼번에 여러 건의 데이터가 함수로 입력됨
  - 모든 그룹함수에서 중요한 부분은 null 값의 포함여부임
    - 거의 대부분의 그룹함수는 함수에 * 를 사용하면 null을 포함
    - 칼럼 이름을 쓰면 해당 칼럼에 데이터가 있는 경우만 작업을 해서 출력하게 됨 (null 제외)

주로 사용되는 그룹함수

함수이름	의미	사용 예
COUNT	입력되는 데이터들의 건수를 출력	COUNT(sal)
SUM	입력되는 데이터들의 합계 값을 출력	SUM(sal)
AVG	입력되는 데이터들의 평균 값을 출력	AVG(sal)
MAX	입력되는 데이터들 중 최고 값을 출력	MAX(sal)
MIN	입력되는 데이터들 중 최저 값을 출력	MIN(sal)
STDDEV	입력되는 데이터들의 표준편차 값을 출력	STDDEV(sal)
VARIANCE	입력되는 데이터들의 분산 값을 출력	VARIANCE(sal)
ROLLUP	입력되는 데이터들의 소계 값을 자동으로 계산해서 출력	
CUBE	입력되는 데이터들의 소계 및 전체 총계를 자동 계산 후 출력	

주로 사용되는 그룹함수

함수이름	의미	사용 예
GROUPING	해당 칼럼이 그룹에 사용되었는지 여부를 1 또는 0으로 반환	
GROUPINGSET	한번의 질의로 여러 개의 그룹화 가능	

Group 함수의 종류

- (1) COUNT 함수
  - 입력되는 데이터의 총 건수를 반환

```
SQL> select count(*), count(hpage)
  2  from professor;

  COUNT(*) COUNT(HPAGE)
---------- ------------
        16            4
```

• Count(*) 의 결과 : null 값을 포함한 결과
• Count(hpage)의 결과 : null 값을 제외한 결과

- (2) SUM 함수
  - 입력된 데이터들의 합계 값을 구하는 함수

```
SQL> set null (null);
SQL> select name, bonus from professor;

NAME           BONUS
---------- ----------
조인형            100
박승고             60
송도권      (null)
양선희      (null)
김영조             80
주재승             90
김도형            110
나한열             50
김현정      (null)
심슨              130
최슬기      (null)

NAME           BONUS
---------- ----------
박원범             50
차범철      (null)
바비                80
전민희      (null)
허                  30
```

```
SQL> select count(bonus), sum(bonus)
  2  from professor;

COUNT(BONUS) SUM(BONUS)
------------ ----------
          10        780
```

null값은 제외하고 연산함
null값과의 연산은 결과가 null

• 데이터가 문자 또는 날짜일때는 sum, avg 사용불가 (연산 불가)
select sum(name) from professor;불가

Group 함수의 종류

- (3) AVG 함수
  - 입력된 값들의 평균값을 구해주는 함수

```
SQL> select count(bonus), sum(bonus), avg(bonus)
  2  from professor;

COUNT(BONUS) SUM(BONUS) AVG(BONUS)
------------ ---------- ----------
          10        780         78
```

  - AVG 함수 주의할 점
    - 위의 예) 총 인원 16명, 보너스를 받는 사람 10명
    - 전체 직원의 평균 보너스 : 총금액(780)/전체인원(16)
    - 그러나 AVG 함수는 자동으로 null 값을 제외하므로 10으로 나누어 잘못된 답이 나옴
  - 수정 후

```
SQL> select count(bonus), sum(bonus), avg(NVL(bonus,0))
  2  from professor;

COUNT(BONUS) SUM(BONUS) AVG(NVL(BONUS,0))
------------ ---------- ------------------
          10        780              48.75
```

Group 함수의 종류

문자열, 날짜도 비교는 가능
Max, min 함수도 사용 가능
select max(ename), min(ename)
from emp; --가능

- (4) MAX/MIN 함수
  - MAX – 주어진 데이터 중에서 가장 큰 값을 돌려줌
  - MIN – 가장 작은 값을 돌려줌
    - 원리 : MAX/MIN 함수는 여러 건의 데이터를 입력 받아서 순서대로 정렬을 함
    - 그리고 그 중에서 최대값/최소값을 추출함
    - => 시간이 오래 걸리는 함수 중 한가지

```
SQL> select max(sal), min(sal)
  2  from emp;

  MAX(SAL)   MIN(SAL)
---------- ----------
      5000        800
```

```
SQL> select max(hiredate), min(hiredate)
  2  from emp;

MAX(HIRE MIN(HIRE
-------- --------
87/05/23 80/12/17
```

distinct

- count([distinct] expr)
  - count 인자로 컬럼이 오게 될 경우 컬럼명 앞에 distinct가 올 수도 있음
  - 중복값 제외한 개수

```
select count(grade), count(distinct grade) from student;
```

COUNT(GRADE)	COUNT(DISTINCTGRADE)
20	4

- sum([distinct] expr)
- avg([distinct] expr)
- max([distinct] expr)
- min([distinct] expr)

Group 함수의 종류

- (5) STDDEV / VARIANCE 함수
  - STDDEV : 표준편차를 구하는 함수
  - VARIANCE 함수 : 분산을 구하는 함수

```
SQL> select stddev(pay), variance(pay)
  2  from professor;

STDDEV(PAY) VARIANCE(PAY)
----------- -------------
 117.869985    13893.3333
```

특정 조건으로 세부적인 그룹화하기

- GROUP BY 절 사용
  - 테이블 전체에 대한 집계를 구하는 것이 아니라, **특정 범위에서의 집계** 데이터를 구함
  - Professor 테이블에서 학과별로 교수들의 평균 급여를 출력하시오.

```
SQL> SELECT deptno, AVG(NVL(pay, 0)) "평균급여"
  2  from professor
  3  GROUP BY deptno;

    DEPTNO   평균급여
---------- ----------
       102 363.333333
       201        450
       301        255
       101        400
       202        285
       203        500
       103 383.333333

7 개의 행이 선택되었습니다.
```

- Professor 테이블에서 전체 교수들의 평균 급여를 출력하시오.

```
select avg(nvl(pay, 0)) from professor;
```

AVG(NVL(PAY,0))
370

```
select deptno, pay from professor
order by deptno;
```

DEPTNO	PAY
101	550
101	380
101	270
102	250
102	350
102	490
103	530
103	330
103	290
201	570
201	330
202	310
202	260
203	500
301	220
301	290

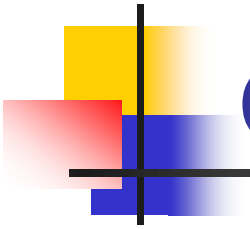
- group by절이 추가되면 group by 절에 사용된 항목별로 데이터의 정렬이 일어남
- DBMS 내부적으로 professor 테이블의 데이터들을 deptno 기준으로 정렬

특정 조건으로 세부적인 그룹화하기

- Professor 테이블에서 학과별, 직급별로 교수들의 평균 급여를 출력하시오.

```
select deptno, position, avg(nvl(pay,0)) as "평균급여"
from professor
group by deptno, position
order by deptno, position;
```

DEPTNO	POSITION	평균급여
101	전임강사	270
101	정교수	550
101	조교수	380
102	전임강사	250
102	정교수	490
102	조교수	350
103	전임강사	290
103	정교수	530
103	조교수	330
201	정교수	570
201	조교수	330
202	전임강사	260
202	조교수	310
203	정교수	500
301	전임강사	225
301	조교수	292.5



GROUP BY 절 사용

- 세부적인 그루핑 조건을 추가하고 싶으면 group by 절에 적어주면 됨
- 전체의 평균 급여를 구하는 것이 아니라 각 학과별로 평균급여를 구하는 것
  - 학과 번호로 모아서 (그룹핑 해서) 평균 급여를 구하는 것이므로 학과번호 별로 group by를 한 것

GROUP BY 절 사용

- group by 절을 사용할 경우 주의사항
  - 1. select 절에 사용된 그룹함수 이외의 칼럼이나 표현식은 반드시 group by 절에 사용되어야 함
    - 그렇지 않을 경우 에러 발생

```
SQL> SELECT deptno, position, AVG(NVL(pay, 0)) "평균급여"
  2  from professor
  3  GROUP BY deptno;
SELECT deptno, position, AVG(NVL(pay, 0)) "평균급여"
               *
1행에 오류:
ORA-00979: GROUP BY 표현식이 아닙니다.
```

  - 2. group by 절에 사용된 칼럼은 select 절에 사용되지 않아도 됨

```
SQL> SELECT deptno,  AVG(NVL(pay, 0)) "평균급여"
  2  from professor
  3  GROUP BY deptno, position;

    DEPTNO   평균급여
---------- ----------
       201        330
       202        310
       301        220
       301        290
```

  - 3. group by 절에는 반드시 칼럼명이 사용되어야 하며 칼럼 Alias(별칭)는 사용하면 안됨

```
SQL> SELECT deptno dno, AVG(NVL(pay, 0)) "평균급여"
  2  from professor
  3  GROUP BY dno;
GROUP BY dno
         *
3행에 오류:
ORA-00904: "DNO": 부적합한 식별자
```

조건을 주고 검색하기

Having절 – **GROUP BY**절에 의해 출력된 결과에 대한 조건을 정의한다.
-조건에 집계함수의 결과가 필요한 경우에 한해 이를 having절에 명시함
- 집계함수를 위한 조건절
- having 절은 group by된 결과를 제한하고자 할 때 사용

- Having 절 사용
  - 부서별 평균급여를 구한 후, 평균 급여가 450 초과인 부서의 부서번호와 평균 급여를 구하시오.

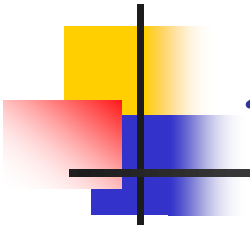
```
SQL> select deptno, AVG(NVL(pay, 0)) from professor
  2  where AVG(pay) > 450
  3  group by deptno;
where AVG(pay) > 450
      *
2행에 오류:
ORA-00934: 그룹 함수는 허가되지 않습니다
```

DEPTNO	평균 급여
101	400
102	363
103	383
201	450
202	285
203	500
301	255

  - Where 절은 그룹 함수를 비교 조건으로 쓸 수 없음
  - => where 대신 having절 사용
  - 그룹함수를 비교 조건으로 사용하려면 반드시 having 절을 사용해야 함
  - Having 절은 group by 절 다음에 와야 함

```
SQL> select deptno, AVG(NVL(pay, 0)) from professor
  2  group by deptno
  3  having AVG(pay) > 450;

    DEPTNO AVG(NVL(PAY,0))
---------- ---------------
       203             500
```



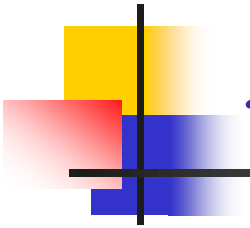
실습

1. emp테이블의 부서별 급여의 총합 구하기.

2. emp 테이블의 job별로 급여의 합계 구하기.

3. emp 테이블의 job별로 최고 급여 구하기

4. emp 테이블의 job별로 최저 급여 구하기



실습

1. emp 테이블의 job별로 급여의 평균 구하기 – 소수이하 2자리만 표시

2. Student 테이블에서 grade별로 weight, height의 평균, 최대값 구하기
3. 2번의 결과에서 키의 평균이 170 이하인 경우 구하기

4. emp2 테이블에서 emp_type별로 pay의 평균을 구한 상태에서 평균 연봉이 3천만원 이상인 경우의 emp_type 과 평균 연봉을 읽어오기

5. emp2의 자료를 이용해서 직급(position)별로 사번(empno)이 제일 늦은 사람을 구하고 그 결과 내에서 사번이 1997로 시작하는 경우 구하기
   (사번의 최대값), like 이용

6. emp 테이블에서 hiredate가 1982년 이전인 사원들 중에서 deptno별, job별 sal의 합계를 구하되
   그 결과 내에서 합계가 2000 이상인 사원만 조회

select 문장의 실행 순서 알아보기

- 옵티마이저 – SQL 문장의 실행 계획을 세운다
- 옵티마이저가 SQL 문을 파싱하면서 SQL의 문법과 적절한 테이블, 컬럼 사용 여부를 검증한다
- 파싱된 SQL의 정보는 오라클의 경우 주로 SGA(System Global Area, 시스템 공유 영역)내의 Shared Pool에 저장되어 같은 SQL 수행시 파싱을 생략할 수 있게 함

select SQL문 실행 순서(오라클 SQL 기준)

5. select {*, 컬럼}
1. from 테이블
2. [where 조건]
3. [group by 컬럼]
4. [having 조건]
6. [order by 컬럼 [desc]]

- from 절이 가장 먼저 실행되고, 그 다음 where 절, group by절, having 절, select 절, 마지막으로 order by 절이 실행됨
- [1] from 절에 사용된 테이블을 인식하여 데이터 딕셔너리에서 관련된 정보들을 파악
  - 옵티마이저는 SGA(System Global Area)내의 Shared Pool의 Dictionary Cache에 관련된 테이블에 대한 정보가 있으면 활용함

select SQL문 실행 순서(오라클 SQL 기준)

- 다른 SQL문에서 이전에 사용된 테이블의 정보가 사용되면 Dictionary Cache에 저장된 정보를 다른 SQL문의 활용시 사용될 수 있음

- [2] where 절에서 조건에 맞는 데이터를 추출
  - from 절에서 사용된 테이블의 정보나 상수가 아닌 조건을 사용시 Syntax 에러 발생
- [3] group by절이 추가되면 group by 절에 사용된 항목별로 데이터의 정렬이 일어남
- [4] having 절은 group by 절로 정렬이 된 데이터를 대상으로 조건을 정의

select SQL문 실행 순서

- [5] 대부분의 RDBMS가 row(행)기준 저장 구조임
  - 그래서 select 이전까지 원하지 않는 컬럼까지도 데이터베이스의 메모리에 저장됨
- [6] order by 절이 가장 나중에 실행됨.
  - select 절에서 선택되지 않은 칼럼이나 연산도 데이터베이스의 메모리에 저장되어 있으므로 order by 절에서 사용 가능

```
select ename, sal from emp
where job='SALESMAN'
order by hiredate, (sal+comm);
```

- 이런 파싱 단계를 거쳐서 옵티마이저는 실행계획을 세우고 사용된 SQL을 SGA(System Global Area) 내의 Shared Pool의 Library Cache에 저장함
- 이후에 동일한 SQL이 사용되는 경우에 재활용이 되도록 함

자동으로 소계/합계를 구해주는 함수

- rollup
  - 소그룹간의 소계를 계산
  - 그룹핑 된 결과에 그룹별 합계 정보를 추가함
- cube
  - group by 항목들간의 다차원적인 소계를 계산할 수 있는 기능
  - 그룹핑 된 컬럼의 모든 가능한 조합에 대한 합계 정보를 추가함
- 소계. 중계, 합계, 총합계 등 여러 레벨의 결산 보고서를 만드는데 유용
- 하나의 sql로 테이블을 한번만 읽어서 보고서 작성
- grouping 함수와 decode를 이용하면 원하는 포맷의 보고서 작성 가능

자동으로 소계/합계를 구해주는 함수

- (1) ROLLUP 함수
  - ROLLUP 함수 – 주어진 데이터들의 소계를 구해줌
    - Group by 절에 주어진 조건으로 소계 값을 구해줌

```
select deptno, position, count(*), sum(pay)
from professor
group by rollup(deptno, position);
```

DEPTNO	POSITION	COUNT(*)	SUM(PAY)
101	정교수	1	550
101	조교수	1	380
101	전임강사	1	270
101		3	1200
102	정교수	1	490
102	조교수	1	350
102	전임강사	1	250
102		3	1090
103	정교수	1	530
103	조교수	1	330
103	전임강사	1	290
103		3	1150
201	정교수	1	570
201	조교수	1	330
201		2	900
202	조교수	1	310
202	전임강사	1	260
202		2	570
203	정교수	1	500
203		1	500
301	조교수	1	290
301	전임강사	1	220
301		2	510
		16	5920

deptno별 모든 position의 subtotal - 소계
grand total - 합계

```
select city, department_name, job_id, count(*) "인원수", sum(salary) "급여합계"
from emp_views
group by rollup(city, department_name, job_id)
order by city, department_name, job_id;
```

칼럼개수+1

city1, city+department2, city+de+job3, 전체4 일케 4개

(a,b,c)=>(a,b,c), (a, b), (a), ()

CITY	DEPARTMENT_NAME	JOB_ID	인원수	급여합계
London	Human Resources	HR_REP	1	6500
London	Human Resources		1	6500
London			1	6500
Munich	Public Relations	PR_REP	1	10000
Munich	Public Relations		1	10000
Munich			1	10000
Oxford	Sales	SA_MAN	5	61000
Oxford	Sales	SA_REP	29	243500
Oxford	Sales		34	304500
Oxford			34	304500
Seattle	Accounting	AC_ACCOU	1	8300
Seattle	Accounting	AC_MGR	1	12000
Seattle	Accounting		2	20300
Seattle	Administration	AD_ASST	1	4400
Seattle	Administration		1	4400
Seattle	Executive	AD_PRES	1	24000
Seattle	Executive	AD_VP	2	34000
Seattle	Executive		3	58000
Seattle	Finance	FI_ACCOUI	5	39600
Seattle	Finance	FI_MGR	1	12000
Seattle	Finance		6	51600
Seattle	Purchasing	PU_CLERK	5	13900
Seattle	Purchasing	PU_MAN	1	11000
Seattle	Purchasing		6	24900
Seattle			18	159200
South Sar	Shipping	SH_CLERK	20	64300
South Sar	Shipping	ST_CLERK	20	55700
South Sar	Shipping	ST_MAN	5	36400
South Sar	Shipping		45	156400
South Sar			45	156400
Southlake	IT	IT_PROG	5	28800
Southlake	IT		5	28800
Southlake			5	28800
Toronto	Marketing	MK_MAN	1	13000
Toronto	Marketing	MK_REP	1	6000
Toronto	Marketing		2	19000
Toronto			2	19000
			106	684400

각 도시와 부서별 소계, 도시별 전체 소계, 전체 합계를 나타내줌

자동으로 소계/합계를 구해주는 함수

- (2) CUBE 함수
  - group by 항목들간의 다차원적인 소계를 계산할 수 있는 기능
  - 그룹핑 된 컬럼의 모든 가능한 조합에 대한 합계 정보를 추가함
  - 모든 경우의 수에 대한 소계를 산출함

  - 예) cube 함수에 도시명, 부서명, 직급 세 가지 컬럼이 들어오게 되면, 이 세가지 가능한 조합의 수는 8가지(도시별, 부서별, 직급별, 도시와 부서별, 도시와 직급별, 부서와 직급별, 도시와 부서와 직급별, 그리고 전체)이 소계 정보를 보여줌

CUBE 함수

```
select deptno, position, count(*), sum(pay)
from professor
group by cube(deptno, position)
order by deptno, position;
```

DEPTNO	POSITION	COUNT(*)	SUM(PAY)
101	전임강사	1	270
101	정교수	1	550
101	조교수	1	380
101		3	1200
102	전임강사	1	250
102	정교수	1	490
102	조교수	1	350
102		3	1090
103	전임강사	1	290
103	정교수	1	530
103	조교수	1	330
103		3	1150
201	정교수	1	570
201	조교수	1	330
201		2	900
202	전임강사	1	260
202	조교수	1	310
202		2	570
203	정교수	1	500
203		1	500
301	전임강사	1	220
301	조교수	1	290
301		2	510
	전임강사	5	1290
	정교수	5	2640
	조교수	6	1990
		16	5920

```
select city, department_name, job_id, count(*) "인원수", sum(salary) "급여합계"
from emp_views
where city='London'
group by cube(city, department_name, job_id)
order by city, department_name, job_id;
```

CITY	DEPARTMENT_NAME	JOB_ID	인원수	급여합계
London	Human Resources	HR_REP	1	6500
London	Human Resources		1	6500
London		HR_REP	1	6500
London			1	6500
	Human Resources	HR_REP	1	6500
	Human Resources		1	6500
		HR_REP	1	6500
			1	6500

- 파라미터 개수가 n 개라 하면, rollup 함수는 n+1 개의 결과를 추출
- cube 함수는 2의 n 승 개의 결과를 추출

(a, b, c) => 모든 경우의 수만큼의 소계가 만들어짐, 2의 3승개 => 8개

(a,b,c), (a,b), (a,c), (b,c), (a), (b), (c) ,()

CITY	DEPARTMENT_NAME	JOB_ID	인원수	급여합계
Seattle	Accounting	AC_ACCOUNT	1	8300
Seattle	Accounting	AC_MGR	1	12008
Seattle	Accounting		2	20308
Seattle	Administration	AD_ASST	1	4400
Seattle	Administration		1	4400
Seattle	Executive	AD_PRES	1	24000
Seattle	Executive	AD_VP	2	34000
Seattle	Executive		3	58000
Seattle	Finance	FI_ACCOUNT	5	39600
Seattle	Finance	FI_MGR	1	12008
Seattle	Finance		6	51608
Seattle	Purchasing	PU_CLERK	5	13900
Seattle	Purchasing	PU_MAN	1	11000
Seattle	Purchasing		6	24900
Seattle		AC_ACCOUNT	1	8300
Seattle		AC_MGR	1	12008
Seattle		AD_ASST	1	4400
Seattle		AD_PRES	1	24000
Seattle		AD_VP	2	34000
Seattle		FI_ACCOUNT	5	39600
Seattle		FI_MGR	1	12008
Seattle		PU_CLERK	5	13900
Seattle		PU_MAN	1	11000
Seattle			18	159216

CITY	DEPARTMENT_NAME	JOB_ID	인원수	급여합계
	Accounting	AC_MGR	1	12008
	Accounting		2	20308
	Administration	AD_ASST	1	4400
	Administration		1	4400
	Executive	AD_PRES	1	24000
	Executive	AD_VP	2	34000
	Executive		3	58000
	Finance	FI_ACCOUNT	5	39600
	Finance	FI_MGR	1	12008
	Finance		6	51608
	Purchasing	PU_CLERK	5	13900
	Purchasing	PU_MAN	1	11000
	Purchasing		6	24900
		AC_ACCOUNT	1	8300
		AC_MGR	1	12008
		AD_ASST	1	4400
		AD_PRES	1	24000
		AD_VP	2	34000
		FI_ACCOUNT	5	39600
		FI_MGR	1	12008
		PU_CLERK	5	13900
		PU_MAN	1	11000
			18	159216

```
select city, department_name, job_id, count(*) "인원수",  sum(salary) "급여합계" from emp_views
where city='Seattle'
group by cube(city, department_name, job_id)
order by city, department_name, job_id;
```

다른 그룹핑 관련 함수들

G_position 칼럼은 각 부서별 소계값을 구할 때는 그루핑에 사용되지 않았음을 보여줌

- (1) GROUPING 함수
  - ROLLUP함수와 CUBE 함수와 함께 사용되는 함수로 어떤 칼럼이 해당 GROUPING 작업에 사용되었는지 아닌지를 구별해주는 역할을 함
  - 어떤 칼럼이 GROUPING 작업에 사용되었으면 0을 반환하고, 사용되지 않았으면 1을 반환

```
SQL> select deptno, sum(pay),
  2       GROUPING(deptno) g_deptno
  3  from professor
  4  group by ROLLUP(deptno);

    DEPTNO   SUM(PAY)   G_DEPTNO
---------- ---------- ----------
       101       1200          0
       102       1090          0
       103       1150          0
       201        900          0
       202        570          0
       203        500          0
       301        510          0
                 5920          1
```

(전체sum, grouping아님)

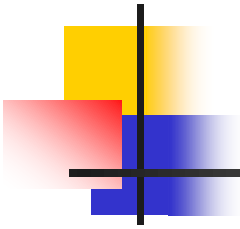
8 개의 행이 선택되었습니다.

```
SQL> select deptno, position, sum(pay),
  2       GROUPING(deptno) g_deptno,
  3       GROUPING(position) g_position
  4  from professor
  5  group by ROLLUP(deptno, position);

    DEPTNO POSITION      SUM(PAY)   G_DEPTNO G_POSITION
---------- ---------- ---------- ---------- ----------
       101 정교수            550          0          0
       101 조교수            380          0          0
       101 전임강사          270          0          0
       101                  1200          0          1
       102 정교수            490          0          0
       102 조교수            350          0          0
       102 전임강사          250          0          0
       102                  1090          0          1
       103 정교수            530          0          0
       103 조교수            330          0          0
       103 전임강사          290          0          0

    DEPTNO POSITION      SUM(PAY)   G_DEPTNO G_POSITION
---------- ---------- ---------- ---------- ----------
       103                  1150          0          1
       201 정교수            570          0          0
       201 조교수            330          0          0
       201                   900          0          1
       202 조교수            310          0          0
       202 전임강사          260          0          0
       202                   570          0          1
       203 정교수            500          0          0
       203                   500          0          1
       301 조교수            290          0          0
       301 전임강사          220          0          0

    DEPTNO POSITION      SUM(PAY)   G_DEPTNO G_POSITION
---------- ---------- ---------- ---------- ----------
       301                   510          0          1
                            5920          1          1
```



GROUPING 함수

- rollup이나 cube에 의한 소계가 계산된 결과
  - grouping(expr) = 1
  - GROUPING 작업에 사용되지 않았으면 1을 반환
  - expr 값이 null 일 경우에는 1을 반환
- 그 외의 결과
  - grouping(expr) = 0
  - GROUPING 작업에 사용되었으면 0을 반환
  - expr 값이 null 이 아닌 경우에는 0을 반환
- grouping 함수와 decode를 이용
  - 소계를 나타내는 필드에 원하는 문자열을 지정할 수 있음

GROUPING 함수 이용

- GROUPING 함수를 이용하여 좀 더 식별 가능하게 그룹핑된 결과를 추출해 낼 수 있음

```
select deptno,
decode(grouping(position),1,'[부서별 합계]', position) "직급",
sum(pay)
from professor
group by rollup(deptno, position)
order by deptno, position;
```

DEPTNO	직급	SUM(PAY)
101	전임강사	270
101	정교수	550
101	조교수	380
101	[부서별 합계]	1200
102	전임강사	250
102	정교수	490
102	조교수	350
102	[부서별 합계]	1090
103	전임강사	290
103	정교수	530
103	조교수	330
103	[부서별 합계]	1150
201	정교수	570
201	조교수	330
201	[부서별 합계]	900
202	전임강사	260
202	조교수	310
202	[부서별 합계]	570
203	정교수	500
203	[부서별 합계]	500
301	전임강사	220
301	조교수	290
301	[부서별 합계]	510
	[부서별 합계]	5920

GROUPING 함수 이용

```
select decode(grouping(deptno),1,'[전체 학과]', deptno) "학과",
decode(grouping(position),1,'[합계]', position) "직급", sum(pay)
from professor
group by cube(deptno, position)
order by deptno, position;
```

학과	직급	SUM(PAY)
101	전임강사	270
101	정교수	550
101	조교수	380
101	[합계]	1200
102	전임강사	250
102	정교수	490
102	조교수	350
102	[합계]	1090
103	전임강사	290
103	정교수	530
103	조교수	330
103	[합계]	1150
201	정교수	570
201	조교수	330
201	[합계]	900
202	전임강사	260
202	조교수	310
202	[합계]	570
203	정교수	500
203	[합계]	500
301	전임강사	220
301	조교수	290
301	[합계]	510
[전체 학과]	전임강사	1290
[전체 학과]	정교수	2640
[전체 학과]	조교수	1990
[전체 학과]	[합계]	5920

다른 그룹핑 관련 함수들

- (2) GROUPING SETS
  - 그루핑 조건이 여러 개일 경우 유용하게 사용
  - 예) STUDENT 테이블에서 학년별로 학생들의 인원수 합계와 학과별로 인원수의 합계를 구해야 하는 경우에 기존에는 학년별로 인원수 합계를 구하고 별도로 학과별로 인원수 합계를 구한 후 UNION 연산을 했음

기존 방법

```
select grade, count(*) from student
group by grade
union
select deptno1, count(*) from student
group by deptno1;
```

GRADE	COUNT(*)
1	5
2	5
3	5
4	5
101	4
102	4
103	2
201	6
202	2
301	2

GROUPING SETS 이용

```
select grade, deptno1, count(*) from student
group by grouping sets((grade), (deptno1))
order by grade, deptno1;
```

GRADE	DEPTNO1	COUNT(*)
1		5
2		5
3		5
4		5
	101	4
	102	4
	103	2
	201	6
	202	2
	301	2

GROUPING SETS

- cube를 사용한 쿼리에서 원하는 집계만 수행할 수 있음

```
select grade, deptno1, count(*) from student
group by cube(grade, deptno1)
order by grade, deptno1;
```

```
select grade, deptno1, count(*) from student
group by grouping sets(grade, deptno1, ())
order by grade, deptno1;

select grade, count(*) from student
group by grouping sets(grade, ())
order by grade;
```

GRADE	DEPTNO1	COUNT(*)
1		5
2		5
3		5
4		5
	101	4
	102	4
	103	2
	201	6
	202	2
	301	2
		20

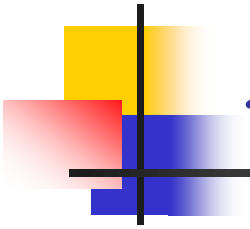
GRADE	COUNT(*)
1	5
2	5
3	5
4	5
	20

GROUPING SETS

```
select grade, deptno1, count(*) from student
group by grade, deptno1
order by grade, deptno1;
```

기존 group by와 비교

GRADE	DEPTNO1	COUNT(*)
1	101	1
1	102	1
1	103	1
1	201	2
2	101	1
2	102	1
2	201	2
2	301	1
3	101	1
3	102	1
3	201	1
3	202	1
3	301	1
4	101	1
4	102	1
4	103	1
4	201	1
4	202	1



실습

- panmae 테이블에서 수량(p_qty)이 3개 이상인 데이터에 대해 판매일(p_date)별, 판매점(p_store)별로 판매금액(p_total)의 합계 구하기
  - rollup, cube이용하여 소계 출력
  - 각각의 경우 grouping함수를 이용해서 요약정보 출력하기(decode()도 이용)

그룹함수 실습

- 1. professor 테이블을 사용하여 교수 중에서 급여(pay)와 보너스(bonus)를 합친 금액이 가장 많은 경우와 가장 적은 경우, 평균 금액을 구하시오.
  - 단, 보너스가 없을 경우는 보너스를 0으로 계산하고, 출력 금액은 모두 소수점 첫째 자리까지만 나오게 하시오
  - max, min, avg, round, nvl 함수 사용

decode 함수 이용-보고서 가로 세로 바꿔보기

emp 테이블에서 부서별로 각 직급별 sal의 합계가 몇인지 계산해서 출력하기

```
select deptno,
    sum(decode(job, 'CLERK', sal)) "CLERK",
    sum(decode(job, 'MANAGER', sal)) "MANAGER",
    sum(decode(job, 'PRESIDENT', sal)) "PRESIDENT",
    sum(decode(job, 'ANALYST', sal)) "ANALYST",
    sum(decode(job, 'SALESMAN', sal)) "SALESMAN"
    from emp
    group by deptno
    order by deptno;
```

DEPTNO	CLERK	MANAGER	PRESIDENT	ANALYST	SALESMAN
10	1300	2450	5000		
20	1900	2975		6000	
30	950	2850			5600
40		9500			

```
select deptno, job, sum(sal)
    from emp
    group by deptno,job
    order by deptno,job;
```

DEPTNO	JOB	SUM(SAL)
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
20	ANALYST	6000
20	CLERK	1900
20	MANAGER	2975
30	CLERK	950
30	MANAGER	2850
30	SALESMAN	5600
40	MANAGER	9500

실습

- student 테이블에서 deptno1(학과)별, grade(학년)별 키(height)의 평균구하기
  - [1] group by 이용
  - [2] group by, decode 이용-가로, 세로 바꿔서

DEPTNO1	GRADE	평균키
101	1	162
101	2	182
101	3	164
101	4	180
102	1	179
102	2	171
102	3	161
102	4	172
103	1	163
103	4	168
201	1	174
201	2	170.5
201	3	171
201	4	177
202	3	177
202	4	182
301	2	184
301	3	160

DEPTNO1	1학년	2학년	3학년	4학년
101	162	182	164	180
102	179	171	161	172
103	163			168
201	174	170.5	171	177
202			177	182
301		184	160	