220928 수업 -2

작성일시	@2022년 9월 28일 오전 9:56
키워드	
유형	
자료	
복습	☐

일반함수

- 프로그램 성격이 강한 함수

- nvl(), nvl2() : null 처리 함수
- decode() : java if 문

```
select id, decode(id, 100, '아이티',
                      200, '영업부',
                      300, '관리팀',
                           '기타부서') as 부서이름
from t_emp;
```

- case() : java switch 문

> 💡 case 조건식 when 결과1 then 출력1
> when 결과2 then 출력2
> when 결과3 then 출력3
> else 출력4
> end "컬럼명"

```
select '0' || to_char(zipcode), case zipcode when 2 then '서울'
                                             when 31 then '경기'
                                             when 41 then '제주'
                                             else '기타'
```

```
                                    end "지역이름"
from t_zip;
```

```
select case when sal <= 1000 then '4급'
            when sal between 1001 and 2000 then '3급'
            when sal between 2001 and 3000 then '2급'
            when sal between 3001 and 4000 then '2급'
            else '특급'
        end "급수", empno, ename
from emp;
```

- 오라클 SQL (변수, 제어문 개념이 없음)

집계함수

1. 집계함수는 group by 절과 같이 사용
2. 모든 집계함수는 null 무시
3. select 절에 집계함수 이외에 다른 컬럼이 오면 반드시 그 컬럼은 group by 절에 명시

count(*) : row 수

```
select count(*) from emp;    --14
```

- count(컬럼명) : 데이터 건수
    - null 포함x

```
select count(comm) from emp;    -- 6
select count(nvl(comm, 0)) from emp;    -- 14
```

sum() : 합계

```
select sum(sal) as 급여의합 from emp;   --29025
```

avg() : 평균

```
select avg(sal) as 급여의평균 from emp;  --2073.21428571428571428571428571428571428571

select round(avg(sal), 0) as 급여평균 from emp; --2073
```

max() : 최대

```
select max(sal) from emp;
```

min() : 최대

```
select min(sal) from emp;
```

```
select empno, count(empno) from emp;
```

- not a single-group group function
- empno와 count(empno)의 열 수가 다름 14건 1건

```
select sum(sal), avg(sal), max(sal), min(sal), count(sal) from emp;
```

- 열 수 같아 동작 가능

그루핑 - group by

부서별 평균 급여

```
select deptno, avg(sal)
from emp
group by deptno;
```

- 실행 순서 : from → goup by → select

직종별 평균 급여

```
select job, round(avg(sal))
from emp
group by job;
```

여러번 묶기

부서별, 직종별 급여의 합

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

	DEPTNO	JOB	SUM(SAL)
1	20	CLERK	1900
2	30	SALESMAN	5600
3	20	MANAGER	2975
4	30	CLERK	950
5	10	PRESIDENT	5000
6	30	MANAGER	2850
7	10	CLERK	1300
8	10	MANAGER	2450
9	20	ANALYST	6000

순서

from → where → group by → select → order by

Having 절

- goup by의 조건절
- select의 가명칭을 사용할 수 없음 (실행 순서 때문!!)

직종별 평균급여가 3000달러 이상인 사원의 직종과 평균급여를 출력

```
select job, avg(sal) as avgsal
from emp
group by job
having avg(sal) >= 3000;
```

순서

from → where → group by → having → select → order by

1. from 절의 조건절은 where
2. group by 절의 조건절은 having 절

JOIN

Join 방법	설 명
Cartesian Product	모든 가능한 행들의 Join
Equijoin	Join 조건이 정확히 일치하는 경우 사용(일반적으로 PK 와 FK 사용)
Non-Equijoin	Join 조건이 정확히 일치하지 않는 경우에 사용(등급,학점)
Outer Join	Join 조건이 정확히 일치하지 않는 경우에도 모든 행들을 출력
Self Join	하나의 테이블에서 행들을 Join 하고자 할 경우에 사용
Set Operators	여러 개의 SELECT 문장을 연결하여 작성한다.

관계 (테이블과 테이블과의 관계)

- 1 : 1
- 1 : N -> 70%는 이 관계
- N : M

등가 조인 (equi join)

- 원테이블과 대응되는 테이블에 있는 컬럼의 데이터를 1:1 매핑

문법

- SQL JOIN 문법(오라클 문법) > 간단
- ANSI 문법 (표준) 권장

[SQL JOIN 문법]

```
select *
from m, s
where m.m1 = s.s1; --권장 사항 x
```

	M1	M2	S1	S2
1	A	1	A	X
2	B	1	B	Y

[ANSI 문법]

```
select *
from m inner join s
on m.m1 = s.s1;
```

```
select *
from m join s
on m.m1 = s.s1;      -- on 조인의 조건절
```

```
select e.empno, e.ename, d.deptno, d.dname
from emp e join dept d
on e.deptno = d.deptno;
```

- 가명칭 사용

테이블 설계

- CRUD 개념, 데이터 건수를 잘 생각해야함

SQL JOIN

```
select m.m1, m.m2, s.s2, x.x2
from m, s, x
where m.m1 = s.s1 and s.s1 = x.x1;
```

ANSI 문법

```
select *
from m join s on m.m1 = s.s1
       join x on s.s1 = x.x1;
```

비등가 조인 non-equi join

- 1:1로 비교할 컬럼이 없을 경우 사용
- 의미만 존재 → 등가 조인의 문법을 그대로 사용

```
select e.empno, e.ename, e.sal
```

```
from emp e join salgrade s
on e.sal between s.losal and s.hisal;
```

	EMPNO	ENAME	SAL
1	7369	SMITH	800
2	7900	JAMES	950
3	7876	ADAMS	1100
4	7521	WARD	1250
5	7654	MARTIN	1250
6	7934	MILLER	1300

---

outer join

- equi join 먼저 선행하고 나서 + 남아있는 데이터 가져오면 됨
- outer join : 주종관계를 파악
- 주인이 되는 테이블의 남은 데이터를 가져올 수 있음

- left outer join (왼쪽)
- right outer join (오른쪽)
- full outer join(left, right >> union)

```
select *
from m left join s
on m.m1 = s.s1
```

	M1	M2	S1	S2
1	A	1	A	X
2	B	1	B	Y
3	(null)	3	(null)	(null)
4	C	3	(null)	(null)

- 조인조건에 만족하지 않는 주인 조건의 데이터를 가져오고 남은 곳은 null로 둠

```
select *
from m full join s
on m.m1 = s.s1;
```

	M1	M2	S1	S2
1	A	1	A	X
2	B	1	B	Y
3	C	3	(null)	(null)
4	(null)	3	(null)	(null)
5	(null)	(null)	(null)	Z

- null 처리 때문에 쓰임

```
select e.employee_id, e.last_name, d.department_id, d.department_name
from employees e join departments d
on e.department_id = d.department_id;
```

- left가 없을 경우, null인 값을 join 불가하므로 누락되는 데이터가 발생

```
select e.employee_id, e.last_name, d.department_id, d.department_name
from employees e left join departments d
on e.department_id = d.department_id;
```

---

self join 자기 참조 → 문법(x) → 등가조인 문법 사용

- 하나의 테이블에 있는 컬럼이 자신의 테이블에 있는 다른 컬럼을 참조하는 경우

```
select e.empno, e.ename, m.empno, m.ename
from emp e left join emp m
on e.mgr = m.empno;
```

- Emp >> Mgr (관리자)
- king의 mgr이 null이므로 left 사용

Natural join

- equijoin과 동일하다 보면 됨
- 두 테이블의 [동일 이름]을 가지는 컬럼 모두 조인
- 조인하고 싶지 않아도 강제로 모두 조인

```
select empno, ename, deptno
from emp natural join dept;
```

using문을 사용하면 컬럼을 선택해 조인 가능

```
SELECT e.empno, e.ename, deptno , dname
FROM emp e JOIN dept d USING(deptno);
```

조인 조건이 없는 조인 결과

→ 행의 수 x 행의 수

```
select * from emp, dept;
select * from emp cross join dept;
```

ㅎ 과제 ㅎ…

교제 p.259 ~ p.282 예제
