

선택함수

여러 가지 경우에 따라 알맞은 결과를 선택 할 수 있음

```
DECODE
-- DECODE(계산식 | 컬럼명, 조건값1, 선택값1, 조건값2, 선택값2..... , 아무것도 일치하지 않을 때 (Default))
-- 비교하고자 하는 값 또는 컬럼이 조건식과 같으면 결과 값 반환
-- 일치하는 값을 확인(자바의 SWITCH와 비슷함)
```

```
CASE WHEN
-- CASE WHEN 조건식 THEN 결과값
--      WHEN 조건식 THEN 결과값
--      ELSE 결과값
-- END

-- 비교하고자 하는 값 또는 컬럼이 조건식과 같으면 결과 값 반환
-- 조건은 범위 값 가능
```

정렬 기준이 되는 컬럼은 반드시 SELECT절에 포함 될 필요가 없다

```
/* 그룹 함수 */
--하나 이상의 행을 그룹으로 묶어 연산하여 총합,평균,하나의 결과 행으로 반환하는 함수.

--SUM (숫자가 기록된 컬럼명) : 합계
--AVG (숫자가 기록된 컬럼명) :평균
--MIN (컬럼명) : 최소
--MAX (컬럼명) : 최대
--MIN / MAX는 숫자가 아니어도 사용 가능하다 (숫자 :대소 , 날짜 :과거 미래 , 문자:ㄱ 나 드 ㄹ/ABCD
```

COUNT

```
-- * COUNT(* | 컬럼명) : 행 개수를 헤아려서 리턴
-- COUNT([DISTINCT] 컬럼명) : 중복을 제거한 행 개수를 헤아려서 리턴
-- COUNT(*) : NULL을 포함한 전체 행 개수를 리턴
-- COUNT(컬럼명) : NULL을 제외한 실제 값이 기록된 행 개수를 리턴한다
```


SUM,AVG,MIN,MAX,COUNT

 03_GROUP BY_HAVING.pdf

492 kB

GROUP BY를 통해 그룹함수의 값이 하나만 나와 값이 여러 개인 칼럼과 총돌이 나는 것을 제어한다.

HOW : 같은 값들이 여러 개 기록된 칼럼을 하나의 그룹으로 묶음

 04_JOIN.pdf

455 kB

선택 함수

여러 가지 경우에 따라 알맞은 결과를 선택 할 수 있음

선택함수 DECODE (계산식 | 조건값1,선택값1,조건값2,선택값2)

자바의 스위치같은 느낌

```
SELECT EMP_NAME AS 이름,DECODE ( SUBSTR(EMP_NO,8,1),1,'남자',2,'여자' )AS 성별
FROM EMPLOYEE;
```

EMP_NO의 8번째 자리 부터 1칸째 자리가 '1'일 경우 남자를 출력.'2'일 경우 여자를 출력한다.

```
CASE WHEN 조건식 THEN 결과값
      WHEN 조건식 THEN 결과값
      ELSE 결과값
END
```

CASE로 시작 END로 끝난다. (시작 할 때 바로 CASE랑 END부터 쓰고 시작)
비교하고자 하는 값 또는 컬럼이 조건식과 같으면 결과 값 반환
조건은 범위 값 가능

```
SELECT EMP_NAME,
       CASE
         WHEN SUBSTR(EMP_NO,8,1) = 1 THEN '남자'
         WHEN SUBSTR(EMP_NO,8,1) = 2 THEN '여자'
       END 성별
FROM EMPLOYEE;
```

만약 EMP_NO 의 8번째 자리 1의 수가 1일 경우 '남자'
만약 EMP_NO 의 8번째 자리 1의 수가 2일 경우 '여자'

```
SELECT EMP_ID,EMP_NAME,SALARY,
       CASE
         WHEN SALARY >= 50000000 THEN '고급'
         WHEN SALARY >=30000000 THEN '중급' --넘어온 값이 500백이 아닌건 이미 검증 되었기 때문에
         ELSE '초급'
       END
FROM EMPLOYEE
```

만약 SALARY가 500만 이상일 경우에는 '고급'
만약 SALARY가 300이상일 때에는 '중급' 이때 이미 500만 이상이 아닌 것이 이미 검증 됐기 때문에
500만 >= SALARY >= 300만이 증명된다

그룹 함수

하나 이상의 행을 그룹으로 묶어 연산하여 총합,평균,하나의 결과 행으로 반환하는 함수.

SUM (숫자가 기록된 컬럼명): 합계

```
SELECT SUM(SALARY)FROM EMPLOYEE;
```

EMPLOYEE에서 SALARY를 전부 조회해 SUM에 누적해 보여준다.

```
SELECT SUM(SALARY) FROM EMPLOYEE
```

```
WHERE DEPT_CODE = 'D9';
```

DEPT_CODE가 D9인 컬럼들의 SALARY를 SUM에 누적해 보여준다.

--AVG (숫자가 기록된 컬럼명):평균

```
SELECT AVG(SALARY) FROM EMPLOYEE;
```

모든 월급을 누적한후 평균 값을 보여준다

MIN (컬럼명): 최소 / MAX (컬럼명): 최대

MIN / MAX는 숫자가 아니여도 사용가능하다

(숫자:대소 , 날짜 :과거미래 , 문자:ㄱ ㄴ ㄷ ㄹ/ABCD

```
SELECT MIN(SALARY),MIN(HIRE_DATE),MIN(EMAIL)
FROM EMPLOYEE;
```

EMPLOYEE의 모든 SALARY,HIRE_DATE,EMAIL 에 접근하면서 최소값, 가장 빠른 입사일 , 가장 앞쪽 문자로 시작하는 이메일들 을 보여준다

COUNT(* | 컬럼명): 행 개수를 확인해 리턴

COUNT([DISTINCT] 컬럼명): 중복을 제거한 행 개수를 헤아려서 리턴

- DISTINCT : 중복을 제외하는 함수.

COUNT(*): NULL을 포함한 전체 행 개수를 리턴

COUNT(컬럼명): NULL을 제외한 실제 값이 기록된 행 개수를 리턴함

```
--EMPLOYEE 테이블의 전체 행의 개수 == 전체 직원수
SELECT COUNT(*) FROM EMPLOYEE;

--DEPT_CODE가 NULL이 아닌 행의 개수.
SELECT COUNT(*) FROM EMPLOYEE
WHERE DEPT_CODE IS NOT NULL;

SELECT COUNT(DEPT_CODE) FROM EMPLOYEE;
--EMPLOYEE 테이블에 있는 부서 개수

SELECT COUNT(DISTINCT DEPT_CODE) FROM EMPLOYEE;
--DISTINCT로 중복을 제거하고 카운트

--남자직원들만 조회
SELECT COUNT(*) FROM EMPLOYEE
WHERE SUBSTR(EMP_NO,8,1)=1;
```

```
SELECT문 해석 순서
5 : SELECT 컬럼명 AS 별칭, 계산식, 함수식
1 : FROM 참조할 테이블명
2 : WHERE 컬럼명 | 함수식 비교연산자 비교값
3 : GROUP BY 그룹을 묶을 컬럼명
4 : HAVING 그룹함수식 비교연산자 비교값
6 : ORDER BY 컬럼명 | 별칭 | 컬럼순번 정렬방식 [NULLS FIRST | LAST];
```

GROUP BY

GROUP BY절 : 같은 값들이 여러개 기록된 컬럼을 가지고 같은 값들을 하나의 그룹으로 묶음

GROUP BY 컬럼명 | 함수식

여러개의 값을 묶어서 하나로 처리할 목적으로 사용함

그룹으로 묶은 값에 대해서 SELECT절에서 그룹함수를 사용함

그룹 함수는 단 한개의 결과 값만 산출하기 때문에 그룹이 여러 개일 경우 오류 발생

여러 개의 결과 값을 산출하기 위해 그룹 함수가 적용된 그룹의 기준을 ORDER BY절에 기술하여 사용

```
SELECT DEPT_CODE, SALARY SUM(SALARY)
FROM EMPLOYEE;
```

이 때
DEPT_CODE는 23행이 나오는데 SUM(SALARY)은 1행만 나오므로 오류가 일어난다

이때 GROUP BY로 DEPT_CODE를 묶으면 그룹 별 SALARY의 합이 나오게 된다

```
-- EMPLOYEE 테이블에서
-- 부서코드, 부서 별 급여의 합계, 부서 별 급여의 평균(정수처리), 인원 수를 조회하고
-- 부서 코드 순으로 정렬
SELECT DEPT_CODE, SUM(SALARY) 인건비, ROUND(AVG(SALARY)) 평균, COUNT(*) 인원수
FROM EMPLOYEE
GROUP BY DEPT_CODE
ORDER BY DEPT_CODE;
```

```
-- EMPLOYEE 테이블에서
-- 성별과 성별 별 급여 평균(정수처리), 급여 합계, 인원 수 조회하고
-- 인원수로 내림차순 정렬

SELECT
    DECODE(SUBSTR(EMP_NO,8,1), '1', '남', '2', '여') 성별,
    FLOOR(AVG(SALARY)) " 급여 평균 ",
    SUM(SALARY) " 급여 합계 ",
    COUNT(*) "인원 수"
FROM EMPLOYEE
GROUP BY DECODE(SUBSTR(EMP_NO,8,1), '1', '남', '2', '여') --별칭을 못쓴다.
ORDER BY "인원 수";
```

WHERE 문으로 그룹화 시키고 싶은 조건만 구한 다음 그룹화 할 수 있다.

```
-- EMPLOYEE 테이블에서 부서코드가 'D5', 'D6'인 부서의 평균 급여 조회
SELECT DEPT_CODE, FLOOR(AVG(SALARY))
FROM EMPLOYEE
WHERE DEPT_CODE IN('D5', 'D6')
GROUP BY DEPT_CODE;

-- EMPLOYEE 테이블에서 직급 별 2000년도 이후 입사자들의 급여 합을 조회
SELECT JOB_CODE, SUM(SALARY)
FROM EMPLOYEE
WHERE HIRE_DATE >= '2001/01/01'
GROUP BY JOB_CODE;
```

* 여러 컬럼을 묶어서 그룹으로 지정 가능

*** GROUP BY 사용시 주의사항 ***

- SELECT문에 GROUP BY절을 사용할 경우
- SELECT절에 명시한 조회하려면 컬럼
- 그룹함수가 적용되지 않은 컬럼을
- 모두 GROUP BY절에 작성해야함.

1. 부서코드를 우선 그룹화 한 다음
2. 부서 별로 나뉜진 사람들의 JOB_CODE로 또 그룹화 한다.

3. 그 다음 같은 그룹에서 같은 JOB_CODE들만의 월급을 SUM으로 누적한다.

```
SELECT DEPT_CODE, JOB_CODE, SUM(SALARY)
FROM EMPLOYEE
GROUP BY DEPT_CODE, JOB_CODE
ORDER BY DEPT_CODE;
```

집계함수 (ROLLUP, CUBE)

그룹 별 산출한 결과 값의 집계를 계산하는 함수

GROUP BY 절에만 작성하는 함수

ROLLUP 함수 : 그룹별로 중간 집계 처리를 하는 함수

그룹별로 묶여진 값에 대한 '중간 집계'와 '총 집계'를 계산하여 자동으로 추가하는 함수

* 인자로 전달받은 그룹중에서 가장 먼저 지정한 그룹별 합계와 총 합계를 구하는 함수

쉽게 말에 한 그룹의 값을 각각 출력하고 그 그룹안에 있던 모든 값들을 합친 다음 보여주는 것

```
-- * SET OPERATION (집합 연산)
-- 여러 개의 SELECT 결과물을 하나의 쿼리로 만드는 연산자
-- 여러가지의 조건이 있을 때 그에 해당하는 여러개의 결과값을 결합시키고 싶을때 사용
-- 초보자들이 사용하기 쉽다. (조건들을 어떻게 엮어야 되는지 덜 생각해도 되니깐)
-- (주의) 집합 연산에 사용되는 SELECT문은 SELECT절이 동일해야함

-- UNION은 OR 같은 개념 (합집합) --> 중복 제거
-- INTERSECT는 AND 같은 개념 (교집합)
-- UNION ALL은 OR 결과 값에 AND 결과값이 더해진거 (합집합 + 교집합) --> 중복 미제거
-- MINUS는 차집합 개념
```

```
SELECT EMP_ID, EMP_NAME, DEPT_CODE, SALARY
FROM EMPLOYEE
WHERE DEPT_CODE = 'D5'

UNION/INTERSECT/UNION ALL/MINUS //연결점

SELECT EMP_ID, EMP_NAME, DEPT_CODE, SALARY
FROM EMPLOYEE
WHERE SALARY > 3000000;
```

JOIN

하나 이상의 테이블에서 데이터를 조회하기 위해 사용한다

수행 결과는 하나의 **RESULT SET**으로 나온다

- 관계형 데이터베이스에서 SQL을 이용해 테이블간 '관계'를 맺는 방법.

- 관계형 데이터베이스는 최소한의 데이터를 테이블에 담고 있어 원하는 정보를 테이블에서 조회하려면 한 개 이상의 테이블에서 데이터를 읽어와야 되는 경우가 많다.

이때, 테이블간 관계를 맺기 위한 **연결고리 역할**이 필요한데, 두 테이블에서 **같은 데이터를 저장하는 컬럼**이 연결고리가 됨.

1. 내부 조인(INNER JOIN) (== 등가 조인(EQUAL JOIN))

연결되는 컬럼의 값이 일치하는 행들만 조인됨. (== 일치하는 값이 없는 행은 조인에서 제외됨.)

1) 연결에 사용할 두 컬럼명이 다른 경우

EMPLOYEE 테이블, DEPARTMENT 테이블을 참조하여
사번, 이름, 부서코드, 부서명 조회
EMPLOYEE 테이블에 DEPT_CODE 컬럼과 DEPARTMENT 테이블에 DEPT_ID 컬럼은
서로 같은 부서 코드를 나타낸다.
이를 통해 두 테이블이 관계가 있음을 알고 조인을 통해 데이터 추출이 가능.
연결에 사용할 컬럼명이 다른 경우 ON()을 사용

```
SELECT EMP_ID, EMP_NAME, DEPT_CODE, DEPT_TITLE
FROM EMPLOYEE
JOIN DEPARTMENT ON (DEPT_CODE=DEPT_ID);
```

2) 연결에 사용할 두 컬럼명이 같은 경우

EMPLOYEE 테이블, JOB 테이블을 참조하여
사번, 이름, 직급코드, 직급명 조회
연결에 사용할 컬럼명이 같은 경우 USING(컬럼명)을 사용함

```
SELECT EMP_ID, EMP_NAME, JOB_CODE, JOB_NAME
FROM EMPLOYEE
JOIN JOB USING (JOB_CODE);
```

```
-- 오라클 사용법 : 별칭 사용
-- 테이블 별로 별칭을 등록할 수 있음.

SELECT EMP_ID, EMP_NAME, E.JOB_CODE, JOB_NAME
FROM EMPLOYEE E, JOB J
WHERE E.JOB_CODE = J.JOB_CODE;
```

내부조인에 문제점
연결에 사용되는 컬럼에 값이 NULL이면 조회 결과에 포함되지 않는다.

-- 2. 외부 조인(OUTER JOIN)

-- 두 테이블의 지정하는 컬럼값이 일치하지 않는 행도 조인에 포함을 시킴
--> *반드시 OUTER JOIN임을 명시해야 한다.

1) LEFT [OUTER] JOIN : 합치기에 사용한 두 테이블 중 왼쪽에 기술된 테이블의 컬럼 수를 기준으로 JOIN

ANSI 표준 방식

```
SELECT EMP_NAME, DEPT_TITLE
FROM EMPLOYEE LEFT JOIN DEPARTMENT ON (DEPT_CODE=DEPT_ID);
```

오라클 방식

```
SELECT EMP_NAME, DEPT_TITLE
FROM EMPLOYEE, DEPARTMENT
WHERE DEPT_CODE = DEPT_ID(+);
```

2) RIGHT [OUTER] JOIN : 합치기에 사용한 두 테이블 중 오른쪽에 기술된 테이블의 컬럼 수를 기준으로 JOIN

ANSI 표준 방식

```
SELECT EMP_NAME, DEPT_TITLE
FROM EMPLOYEE RIGHT JOIN DEPARTMENT ON (DEPT_CODE = DEPT_ID);
```

오라클 방식

```
SELECT EMP_NAME, DEPT_TITLE
FROM EMPLOYEE, DEPARTMENT
WHERE DEPT_CODE(+) = DEPT_ID;
```

3) 교차 조인 : 각 테이블의 모든 값을 중복으로 연결한다.



4) 비등가 조인(NON EQUAL JOIN) : '='(등호)를 사용하지 않는 조인문 지정한 컬럼 값이 일치하는 경우가 아닌, 값의 범위에 포함되는 행들을 연결하는 방식

```
SELECT EMP_NAME, SALARY, E.SAL_LEVEL, S.SAL_LEVEL
FROM EMPLOYEE E
JOIN SAL_GRADE S ON (SALARY BETWEEN MIN_SAL AND MAX_SAL);
```

-- 5. 자체 조인(SELF JOIN): 같은 테이블을 조인 / 자기 자신과 조인을 맺음

ANSI 표준

```
SELECT E.EMP_ID, E.EMP_NAME, NVL(E.MANAGER_ID, '없음'), NVL(M.EMP_NAME, '없음')
FROM EMPLOYEE E
LEFT JOIN EMPLOYEE M ON (E.MANAGER_ID=M.EMP_ID);
```

오라클 구문

```
SELECT E.EMP_ID, E.EMP_NAME, NVL(E.MANAGER_ID, '없음'), NVL(M.EMP_NAME, '없음')
FROM EMPLOYEE E, EMPLOYEE M
WHERE E.MANAGER_ID = M.EMP_ID;
```

6. 자연 조인(NATURAL JOIN)

동일한 타입과 이름을 가진 컬럼이 있는 테이블 간의 조인을 간단히 표현하는 방법
반드시 두 테이블 간의 동일한 컬럼명, 타입을 가진 컬럼이 필요
없을 경우 교차조인이 됨.

7. 다중 조인

N개의 테이블을 조회할 때 사용 (순서 중요!)
** JOIN은 순서대로 하나씩 진행된다!**

A와 B를 조인하고 B에 있는 칼럼을 조회하고 싶을 경우
B를 먼저 조인하고 C를 조인해야 참조 가능하다