Oracle Database

Oracle SQL

Oracle SQL

Basic Query - SELECT 문의 기초

SELECT

- 데이터베이스에서 원하는 데이터를 검색, 추출
- Syntax

```
SELECT [ALL|DISTINCT] 열_리스트
     FROM  테이블_리스트
     [WHERE 조건]
     [GROUP BY 열_리스트 [HAVING 그룹 조건]]
     [ORDER BY 열_리스트 [ASC | DESC]];
```

- 기능
  - Projection : 원하는 컬럼 선택
  - Selection : 원하는 튜플 선택
  - Join : 두 개의 테이블 결합
  - 기타 : 각종 계산, 정렬, 요약(Aggregation)

SELECT의 기능

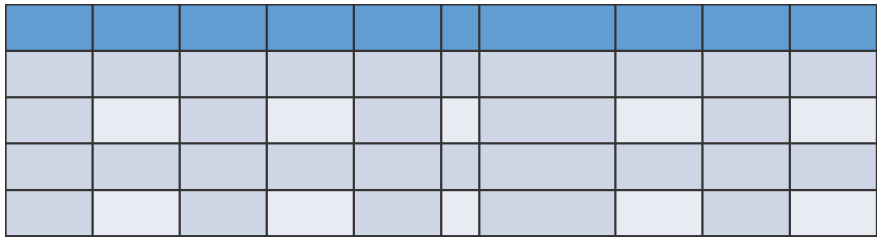
Projection

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80/12/17	880		20
7499	ALLEN	SALESMAN	7698	81/02/20	1760	300	30
7521	WARD	SALESMAN	7698	81/02/22	1375	500	30
7566	JONES	MANAGER	7839	81/04/02	2975		20
7654	MARTIN	SALESMAN	7698	81/09/28	1375	1400	30
7698	BLAKE	MANAGER	7839	81/05/01	2850		30
7782	CLARK	MANAGER	7839	81/06/09	2450		10
7788	SCOTT	ANALYST	7566	87/04/19	3000		20
7839	KING	PRESIDENT		81/11/17	5000		10
7844	TURNER	SALESMAN	7698	81/09/08	1650	0	30
7876	ADAMS	CLERK	7788	87/05/23	1210		20
7900	JAMES	CLERK	7698	81/12/03	1045		30
7902	FORD	ANALYST	7566	81/12/03	3000		20
7934	MILLER	CLERK	7782	82/01/23	1430		10

Selection

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Join



기본 SELECT 문

- 형식

```
SELECT * | {[DISTINCT] column|expression [alias], …}
     FROM  table
```

- 내용 설명
  - * : 모든 컬럼 반환
  - DISTINCT : 중복된 결과 제거
  - SELECT 컬럼명 : Projection
  - FROM 대상 테이블
  - ALIAS : 컬럼 이름 변경(표시용)
  - Expression : 기본적인 연산 및 함수 사용 가능

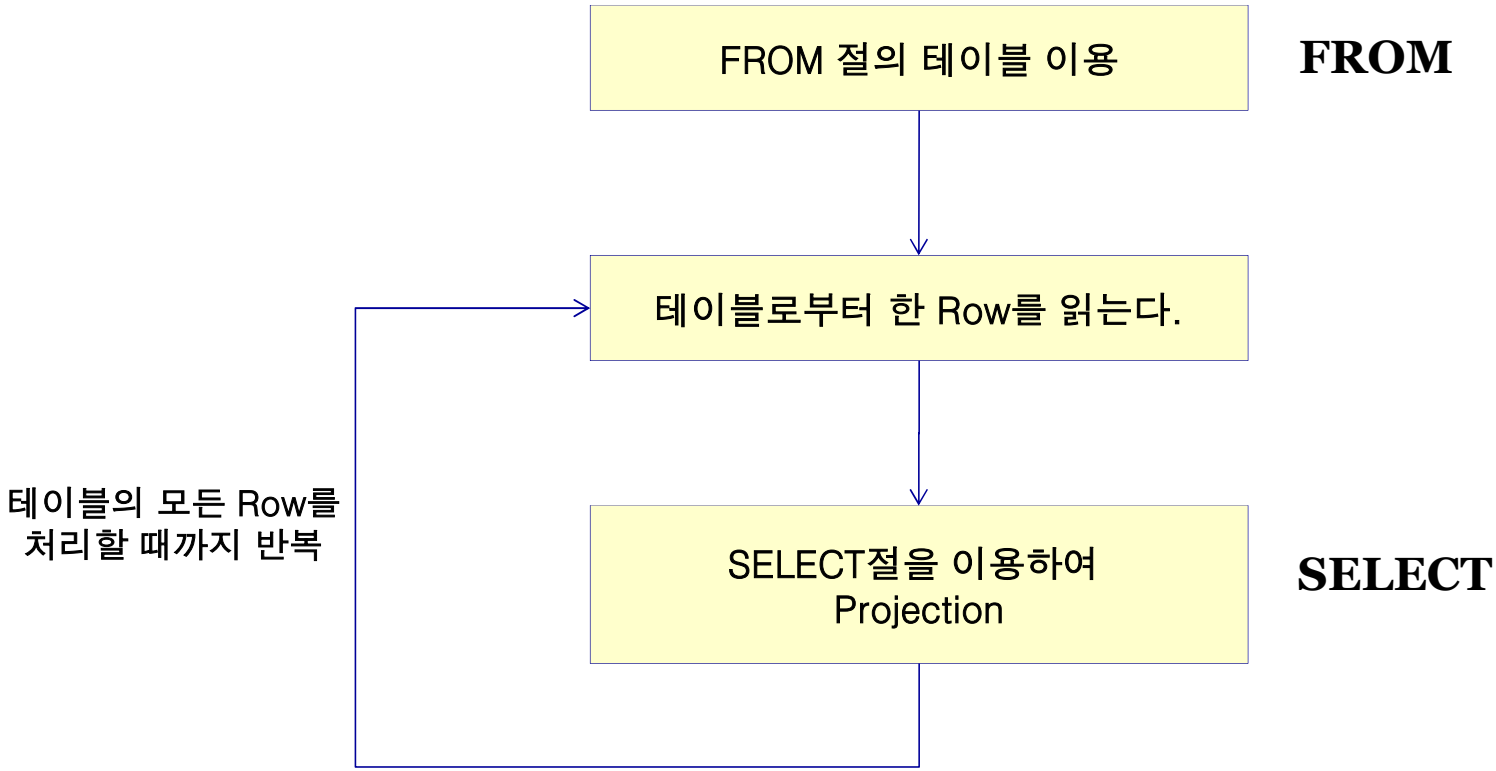
기본 SELECT 문의 예

- SELECT * FROM emp;
- SELECT ename FROM emp;
- SELECT ename, job FROM emp;
- SELECT ename 이름 FROM emp;

```
SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- -------- ---------- ---------- ----------
      7369 SMITH      CLERK            7902 80/12/17        800                    20
      7499 ALLEN      SALESMAN         7698 81/02/20       1600        300         30
      7521 WARD       SALESMAN         7698 81/02/22       1250        500         30
      7566 JONES      MANAGER          7839 81/04/02       2975                    20
      7654 MARTIN     SALESMAN         7698 81/09/28       1250       1400         30
      7698 BLAKE      MANAGER          7839 81/05/01       2850                    30
      7782 CLARK      MANAGER          7839 81/06/09       2450                    10
      7788 SCOTT      ANALYST          7566 87/04/19       3000                    20
      7839 KING       PRESIDENT             81/11/17       5000                    10
      7844 TURNER     SALESMAN         7698 81/09/08       1500          0         30
      7876 ADAMS      CLERK            7788 87/05/23       1100                    20
      7900 JAMES      CLERK            7698 81/12/03        950                    30
      7902 FORD       ANALYST          7566 81/12/03       3000                    20
      7934 MILLER     CLERK            7782 82/01/23       1300                    10
```

SELECT … FROM 절의 처리
: Flow



SELECT / FROM 절

- 모든 컬럼의 조회
  - 컬럼 목록에 *를 표시하면 테이블 내 모든 컬럼을 Projection 한다

```
SELECT * FROM {TABLE명};
```

  - SQL Recap:
    - 마지막은 세미콜론(;)
    - 대소문자 구분하지 않음

  - 테이블 employees의 모든 튜플을 불러와 모든 컬럼을 Projection 한다.

```
SELECT * FROM employees;
```

  - 테이블 departments의 모든 튜플을 불러와 모든 컬럼을 Projection 한다

```
SELECT * FROM departments;
```

SELECT / FROM 절

- 원하는 컬럼의 조회 (Projection을 원하는 컬럼명을 지정)
  - 컬럼 목록에 *를 표시하면 테이블 내 모든 컬럼을 Projection 한다

```
SELECT {컬럼명1}, {컬럼명2}, …
FROM {TABLE명};
```

  - 예: 다음 쿼리문을 보고 출력 결과를 예측해 봅시다

```
SELECT employee_id, first_name, last_name
FROM employees;
```

- [연습] hr.employees
  - 사원의 이름(first_name)과 전화번호, 입사일, 급여를 출력해 봅시다
  - 사원의 이름(first_name)과 성(last_name), 급여, 전화번호, 입사일을 출력해 봅시다

산술연산(Arithmetic Operation)

- 기본적인 산술연산 사용 가능
  - +, -, *, /, 부호, 괄호 등
  - 우선순위 : 부호 -> 괄호 -> *, / -> +, -
  - 컬럼명, 숫자
  - 예
    - `SELECT ename, (sal + 200) * 12 FROM emp;`

```
SQL> SELECT ename, (sal+200) * 12 FROM emp;

ENAME      (SAL+200)*12
---------- ------------
SMITH             12000
ALLEN             21600
WARD              17400
JONES             38100
MARTIN            17400
BLAKE             36600
```

SELECT / FROM 절
: 산술연산 연습

- 단순 수식 계산하기

```
SELECT {산술식} FROM dual;
```

  - dual : Oracle의 Pseudo Table. 특정 테이블이 아닌 오라클 시스템으로부터 값을 가져올 때 사용
  - 컬럼명 대신 산술식을 부여한다

- 필드 값의 산술연산

```
SELECT first_name, salary
FROM employees;
SELECT first_name, salary, salary*12
FROM employees;
SELECT first_name, salary, salary*12, (salary+300)*12
FROM employees;
```

- [예제] 다음을 실행해보고 오류의 원인이 무엇인지 알아봅시다

```
SELECT job_id*12
FROM employees;
```

NULL

- 아무런 값도 정해지지 않았음을 의미
- 어떠한 데이터타입에도 사용 가능
- NULL은 0이나 space 등과는 다르다
- NOT NULL 컬럼이나 Primary Key 속성의 컬럼에는 사용할 수 없음
- NULL을 포함한 산술식은 NULL
  - `SELECT sal, comm, (sal + comm) * 12 FROM emp;`

- `NVL(expr1, expr2)`
  - `expr1`이 `NULL`이면 `expr2`를 출력
  - 데이터타입이 호환 가능해야 함
  - `SELECT sal, comm, (sal + NVL(comm, 0)) * 12 FROM emp`

Column Alias

- 컬럼의 출력 제목을 변경
- alias 내에 공백이다 특수문자를 포함하고자 한다면 큰따옴표(" ")를 사용
- 형태
  - SELECT ename name FROM emp;
  - SELECT ename as name FROM emp;
  - SELECT ename "name" FROM emp;
  - SELECT (sal + comm) "Annual Salary" From emp;

```
SQL> select empno no, ename as name, job "to do" from emp;

        NO NAME       to do
---------- ---------- ---------
      7369 SMITH      CLERK
      7499 ALLEN      SALESMAN
      7521 WARD       SALESMAN
      7566 JONES      MANAGER
      7654 MARTIN     SALESMAN
      7698 BLAKE      MANAGER
      7782 CLARK      MANAGER
      7788 SCOTT      ANALYST
      7839 KING       PRESIDENT
      7844 TURNER     SALESMAN
      7876 ADAMS      CLERK
```

Literal

- SELECT 절에 사용되는 문자, 숫자, Date 타입 등의 상수
- Date 타입이나 문자열은 작은따옴표(' ')로 둘러싸야 함
- 문자열 결합(Concatenation) 연산자 : ||
- 예
  - SELECT ename, 1000, SYSDATE FROM emp;
  - SELECT 'Name is ' || ename || 'and no is ' || empno FROM emp;

```
SQL> SELECT 'Name is ' || ename || ' and no is ' || empno FROM emp;

'NAMEIS'||ENAME||'ANDNOIS'||EMPNO
--------------------------------------------------------------------------------
Name is SMITH and no is 7369
Name is ALLEN and no is 7499
Name is WARD and no is 7521
Name is JONES and no is 7566
Name is MARTIN and no is 7654
Name is BLAKE and no is 7698
Name is CLARK and no is 7782
Name is SCOTT and no is 7788
```

SELECT / FROM 절
: Alias 연습

- Projection 컬럼에 Alias 사용 예

```
SELECT employee_id as empNO, first_name "E-name", salary "급 여"
FROM employees;
```

- [예제] hr.employees 전체 튜플에 다음과 같이 Column Alias를 붙여 출력해 봅니다
  - 이름 : first_name last_name
  - 입사일: hire_date
  - 전화번호 : phone_number
  - 급여 : salary
  - 연봉 : salary * 12

SELECT / FROM 절
: NULL과 NVL 연습

- 다음 쿼리를 살펴봅시다

```
SELECT first_name, salary, commission_pct,
       salary + salary * commission_pct
    FROM employees;
```

  - SQL Recap:
    - NULL 이 포함된 산술계산은 NULL
      -> commission_pct가 NULL인 사원은 최종 급여가 계산되지 않고 NULL로 출력

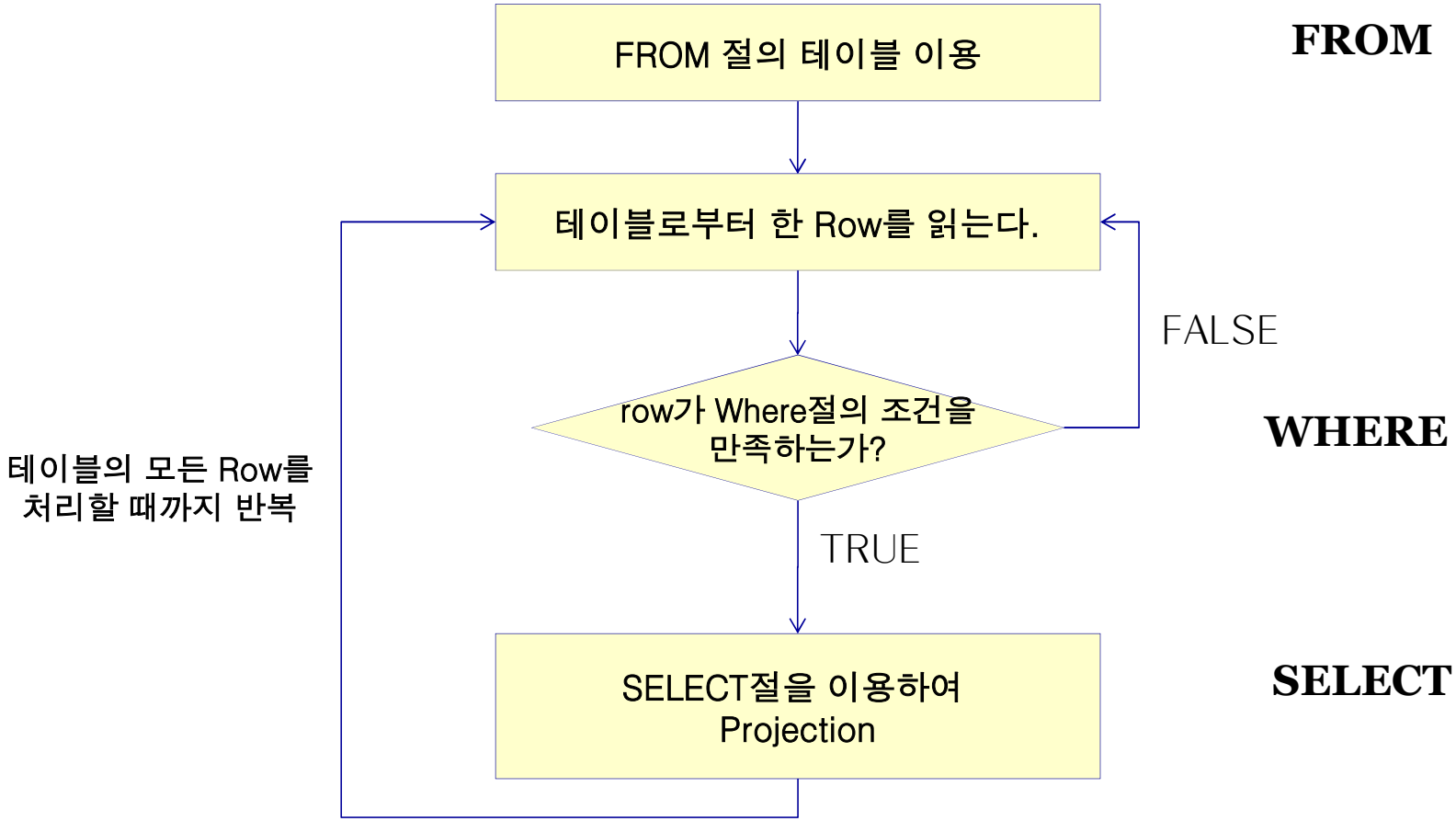
- NVL을 이용, 수정해 봅시다

```
SELECT first_name, salary, nvl(commission_pct, 0),
       salary + salary * nvl(commission_pct, 0)
    FROM employees;
```

WHERE

- 조건을 부여하여 만족하는 ROW를 선택(Selection)
- 연산자
  - =, !=, >, <, <=, >=
  - IN : 집합에 포함되는가?
  - BETWEEN a AND b : a와 b 사이인가?
  - IS NULL, IS NOT NULL : NULL 여부 검사
  - AND, OR : 둘 다 만족? 둘 중 하나만 만족?
  - NOT : 만족하지 않음?
  - ANY, ALL : 집합 중 어느 한 열, 집합 중 모든 열 (다른 비교 연산자와 함께 사용)
  - EXIST : 결과 Row가 한 개 이상 있는가? (Subquery에서 사용)

WHERE 절의 처리
: Flow



비교 연산자

```
SELECT dname
      FROM dept
      WHERE deptno = 30;
```

```
SELECT ename, sal
      FROM emp
      WHERE sal >= 1500;
```

Operator	Purpose
=	Equal to
<>	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

비교 연산자

```
SELECT dname
     FROM dept
     WHERE deptno IN (10, 20);
```

```
SELECT ename, sal
     FROM emp
     WHERE job = ANY('ANALYST', 'CLERK);
```

```
SELECT ename, sal
     FROM emp
     WHERE sal > ALL(2000, 3000);
```

Operator	Purpose
IN (*list*)	Equal to any member of *list*.
ANY (*list*)	Compares a value to each value in a *list*. Must be preceded by =, !=, >, <, <=, >=.
ALL (*list*)	Compares a value to every value in a *list*. Must be preceded by =, !=, >, <, <=, >=.
EXISTS (*subquery*)	TRUE if a *subquery* returns at least one row.

- ANY, ALL 연산자의 앞에는 비교연산자가 와야 함
- IN 연산자는 = ANY 연산자와 같은 결과
- NOT IN 연산자는 <>ALL 연산자와 같은 결과

비교 연산자

```
SELECT ename
     FROM emp
     WHERE hiredate BETWEEN
           '82/01/01' AND
           '82/12/31';
```

```
SELECT dname
     FROM dept
     WHERE dname LIKE 'A%';
```

```
SELECT ename
     FROM emp
     WHERE comm IS NULL;
```

Operator	Purpose
BETWEEN *a* AND *b*	greater than or equal to *a* and less than or equal to *b*.
a LIKE *b* [ESCAPE '*c*']	TRUE if *a* does match the pattern *b*. '%' and '_' are wildcard characters within *b*. A wildcard character is treated as a literal if preceded by the escape character '*c*'
IS NULL	Null test

WHERE 절
: 비교 연산자 연습

- [예제] hr.employees
  - 급여가 15000 이상인 사원들의 이름과 연봉을 출력하십시오
  - 07/01/01일 이후 입사자들의 이름과 입사일을 출력하십시오.
  - 이름이 'Lex'인 사원의 연봉과 입사일, 부서 ID를 출력하십시오.
  - 부서 ID가 10인 사원의 명단이 필요합니다.

LIKE 연산

- Wildcard를 이용한 문자열 부분 매칭
- Wildcard
  - % : 임의의 길이의 문자열(공백 문자 포함)
  - _ : 한 글자
- Escape
  - ESCAPE 뒤의 문자열로 시작하는 문자는 Wildcard 가 아닌 것으로 해석
- 예
  - ename LIKE 'KOR%' : KOR로 시작하는 모든 문자열 (KOR 포함)
  - ename LIKE 'KOR_' : KOR 뒤에 하나의 문자가 오는 모든 문자열
  - ename LIKE 'KOR/%%' ESCAPE '/' : 'KOR%'로 시작하는 모든 문자열

논리 연산자

- 논리 연산자를 이용하여 복잡한 질의 조건을 줄 수 있으며, 전체 조건식의 연산 결과가 TRUE인 Row만 선택된다

1	Arithmetic operators
2	Comparison operators
3	NOT
4	AND
5	OR

논리 연산자의 결과값

- NULL을 주의
  - NULL이 있으면 기본적으로 NULL, 확실히 답이 나오는 경우만 계산 가능

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

WHERE 절
: 논리 연산자 연습

- 조건이 두 개 이상일 때 한꺼번에 조회하기

```
select first_name, salary
from employees
where salary >= 14000
and salary <= 17000;
```

- 급여가 14000 이상 17000이하인 사원의 이름과 급여를 출력

- [예제] hr.employees
  - 급여가 14000 이하이거나 17000 이상인 사원의 이름과 급여를 출력하십시오.
  - 부서 ID가 90인 사원 중, 급여가 20000 이상인 사원은 누구입니까?

WHERE 절
: BETWEEN 연습

- BETWEEN 연산자를 이용한 특정 구간의 값 출력

```
select first_name, salary
from employees
where salary between 14000 and 17000;
```

• 급여가 14000 이상 17000이하인 사원의 이름과 급여를 출력하시오

  - 앞서 비교 연산자를 이용한 쿼리문과 비교해 봅시다
  - 작은 값을 앞쪽에, 큰 값을 뒤쪽에 부여
  - 유용하지만 느린 연산자에 속함

- [예제] hr.employees
  - 입사일이 07/01/01 ~ 07/12/31 구간에 있는 사원의 목록을 출력해 봅시다

WHERE 절
: IN 연습

- 여러 조건을 검사하기 위해 IN 연산자를 활용합니다

```
SELECT first_name, last_name, salary
FROM employees
WHERE first_name IN ('Neena', 'Lex', 'John');
```

- [예제]
  - 부서 ID가 10, 20, 40인 사원의 명단을 출력해 봅시다
  - MANAGER ID가 100, 120, 147 인 사원의 명단을 출력해 봅시다
    - 비교연산자+논리연산자를 이용하여 출력해 보고
    - IN 연산자를 이용해 출력해 봅시다
    - 두 쿼리를 비교해 보고 IN 연산자의 유용한 점을 생각해 봅시다

WHERE 절
: Like 연습

- SQL Recap:
  - % : 임의의 길이의 문자열(공백 문자 가능)
  - _ : 임의의 한 글자

- [예제] hr.employees
  - 이름에 am을 포함한 사원의 이름과 급여를 출력해 봅시다
  - 이름의 두 번째 글자가 a인 사람의 이름과 급여를 출력해 봅시다
  - 이름의 네 번째 글자가 a인 사원의 이름을 출력해 봅시다
  - 이름이 4글자인 사원 중 끝에서 두 번째 글자가 a 인 사원의 이름을 출력해 봅시다

연산자 우선 순위

1. Arithmetic Operators
2. Concatenation Operators
3. Comparison Conditions
4. IS [NOT] NULL, LIKE, [NOT] IN
5. [NOT] BETWEEN
6. NOT Logical condition
7. AND Logical condition
8. OR logical condition

ORDER BY

- 주어진 컬럼 리스트의 순서로 결과를 정렬
- 결과 정렬 방법
  - ASC : 오름차순 (작은값 -> 큰값) - default
  - DESC : 내림차순 (큰 값 -> 작은 값)
- 정렬 기준은 여러 컬럼에 지정 가능
- 컬럼 이름 대신 Alias, expr, SELECT 절 상의 순서(1, 2, 3…)도 사용 가능
  - 예) SELECT * FROM emp ORDER BY deptno, sal DESC
    : 의미 해석 = 부서 번호 순으로 정렬한 후, sal이 높은 사람부터 출력

ORDER BY

- ORDER BY 절을 이용하여 데이터를 사용하기 편리하게 정렬하기

```
SELECT first_name, salary
FROM employees
ORDER BY salary DESC;
```

```
SELECT first_name, salary
FROM employees
WHERE salary >= 9000
ORDER BY salary DESC;
```

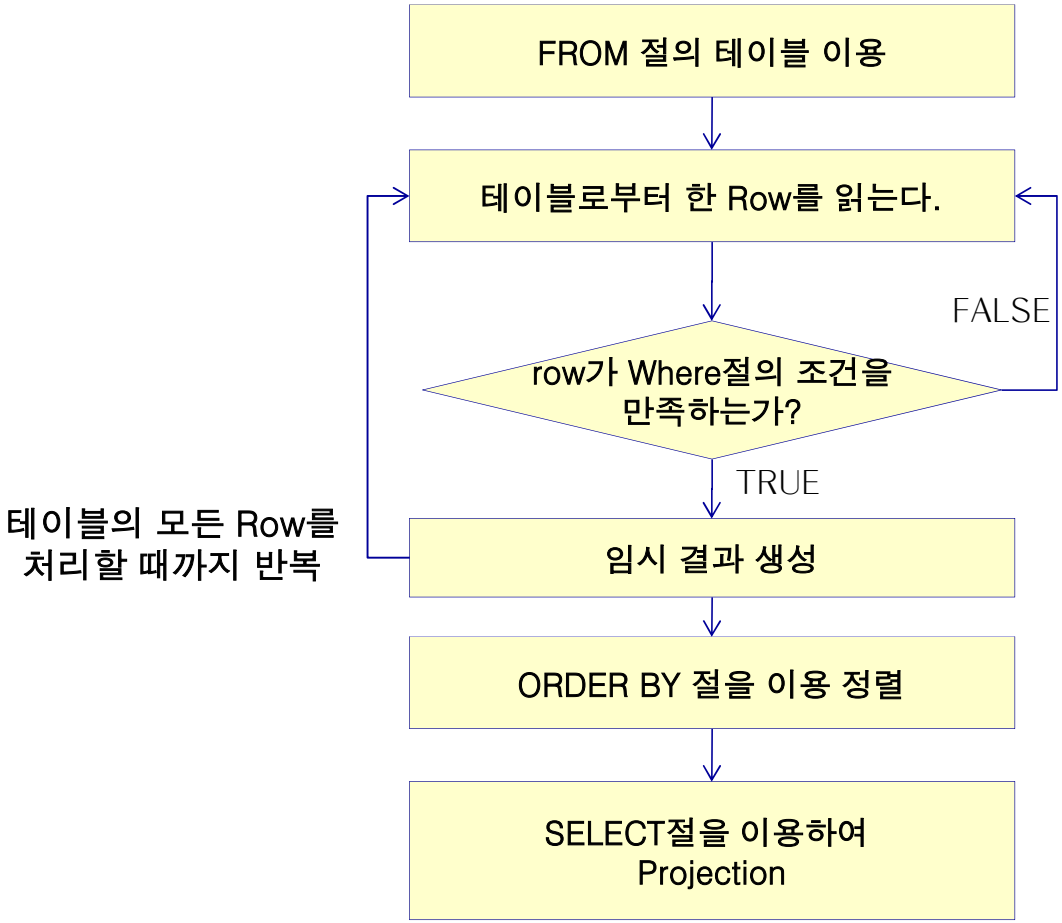
- 기본값: 오름차순 (ASC: 작은 값 -> 큰 값 순) <-> 내림차순(DESC)은 반대
  - 한글: 가, 나, 다, 라 …
  - 영어: A, B, C, D
  - 숫자: 1, 2, 3, 4
  - 날짜 : 오래된 날짜 -> 최근 날짜 순

  - 정렬 조건이 복수일 경우는 ,로 구분하여 명시

ORDER BY

- [연습] hr.employees
  - 부서 번호를 오름차순으로 정렬하고 부서번호, 급여, 이름을 출력하십시오
  - 급여가 10000 이상인 직원의 이름을 급여 내림차순(높은 급여 -> 낮은 급여)으로 출력하십시오
  - 부서 번호, 급여, 이름 순으로 출력하되 부서번호 오름차순, 급여 내림차순으로 출력하십시오.

ORDER BY 절의 처리



Oracle SQL

단일행 함수 (Single Row Function)

SQL Functions

- Single-Row Function : 하나의 Row를 입력으로 받는 함수
  - 숫자 함수
  - 문자 함수
  - 날짜 함수
  - 변환 함수
  - 기타 함수
- Aggregation Function : 집합 함수
- Analytic Function : 분석 함수
- Regular Expression : 정규표현식 (Oracle 10g 이상)

단일행 함수
: 문자열 함수

Function	설명
CONCAT(s1,s2)	문자열 결함
INITCAP(s)	첫글자만 대문자로 변경
LOWER(s)	소문자로 변경
UPPER(s)	대문자로 변경
LPAD(s1,n,s2)	문자열의 왼쪽 채움 (길이:n, 채움문자 s2)
RPAD(s1,n,s2)	문자열 오른쪽 채움 (길이:n, 채움문자 s2)
LTRIM(s,c)	문자열 왼쪽 c문자열 제거
RTRIM(s,c)	문자열 오른쪽 c문자열 제거
CHR(n)	ASCII값이 n인 문자 반환
REPLACE(s,p,r)	문자열 치환, S속의 p문자열을 r로 치환
SUBSTR(s,m,n)	부분 문자열, m번째부터 길이 n인 문자열 반환
TRANSLATE(s,from,to)	s에서 from 문자열의 각 문자를 to문자열의 각 문자로 변환
ASCII(s)	ASCII값 반환
INSTR(s1,s2,m,n)	문자열 검색, s1의 m번째부터 s2 문자열이 나타나는 n번째 위치 반환
LENGTH(s)	문자열 길이 반환

단일행 함수
: 문자열 함수 사용 예

- 대소문자 변환

Function	Result
LOWER('Database system')	database system
UPPER('Database system')	DATABASE SYSTEM
INITCAP('Database system')	Database System

- 문자열 조작

함수	결과
CONCAT('Data', 'Base')	DataBase
SUBSTR('Database',2,4)	atab
LENGTH('database')	8
INSTR('Database', 'b')	5
LPAD(salary,10,'*')	*****24000
RPAD(salary, 10, '*')	24000*****
TRIM('#' FROM '##Database###')	Database

단일행 함수
: 주요 문자열 함수

- INITCAP(컬럼명)
  - 영어의 첫 글자만 대문자로 출력하고 나머지는 소문차로 출력하는 함수

```
SELECT email, INITCAP(email), department_id
     FROM employees
     WHERE department_id = 100;
```

- LOWER(컬럼명) / UPPER(컬럼명)
  - 문자열 값을 전부 소문자/대문자로 변경하는 함수

```
SELECT first_name, LOWER(first_name), UPPER(first_name)
     FROM employees
     WHERE department_id = 100;
```

  - 대소문자 구분 없이 문자열을 검색하고자 할 때 유용하게 사용

단일행 함수
: 주요 문자열 함수

- SUBSTR(컬럼명, 시작위치, 글자수)
  - 주어진 문자열에서 특정 길이의 문자열을 구하는 함수

```
SELECT first_name, SUBSTR(first_name,1,3), SUBSTR(first_name,-3,2)
     FROM employees
     WHERE department_id = 100;
```

  - 시작 위치가 양수인 경우 왼쪽부터 검색하여 글자수만큼 추출
  - 시작 위치가 음수인 경우 오른쪽 -> 왼쪽 검색을 한 후 글자수만큼 추출

단일행 함수
: 주요 문자열 함수

- LPAD(컬럼명, 자리수, '채울문자') / RPAD(컬럼명, 자리수, '채울문자')
  - LPAD() : 왼쪽 공백에 특별한 문자로 채우기
  - RPAD() : 오른쪽 공백에 특별한 문자로 채우기

```
SELECT first_name,
       LPAD(first_name,10,'*'),
       RPAD(first_name,10,'*')
FROM employees;
```

단일행 함수
: 주요 문자열 함수

- REPLACE(컬럼명, 문자열1, 문자열2)
  - 컬럼명에서 문자열1을 문자열2로 바꾸는 함수

```
SELECT first_name,
       REPLACE(first_name, 'a', '*')
FROM employees
WHERE department_id =100;
```

```
SELECT first_name,
       REPLACE(first_name, 'a', '*'),
       REPLACE(first_name, SUBSTR(first_name, 2, 3), '***')
FROM employees
WHERE department_id =100;
```

단일행 함수
: 숫자 함수

Function	설명	Example	Result
ABS(n)	절대값	ABS(-5)	5
CEIL(n)	n 보다 크거나 같은 최소 정수	CEIL(-2.4)	-2
FLOOR(n)	n 보다 작거나 같은 최대 정수	FLOOR(-2.4)	-3
MOD(m,n)	나머지	MOD(13,2)	1
POWER(m,n)	m의 n승	POWER(2,3)	8
ROUND(m,n)	소수점아래 n자리까지 반올림	ROUND(4.567,2)	4.57
TRUNC(m,n)	소수점아래 n자리미만 버림	TRUNC(4.567,2)	4.56
SIGN(n)	부호 (1, 0, -1)	SIGN(-10)	-1

단일행 함수
: 주요 숫자 함수

- ROUND(숫자, 출력을 원하는 자리수)
  - 주어진 숫자의 반올림을 하는 함수

```
SELECT ROUND(123.346, 2) "r2",
       ROUND(123.456, 0) "r0",
       ROUND(123.456, -1) "r-1"
FROM dual;
```

- TRUNC(숫자, 출력을 원하는 자리수)
  - 주어진 숫자의 버림을 하는 함수

```
SELECT TRUNC(123.346, 2) "r2",
       TRUNC(123.456, 0) "r0",
       TRUNC(123.456, -1) "r-1"
FROM dual;
```

Date 타입

- Oracle의 Date Type
  - century, year, month, day, hours, minutes, seconds 등을 포함한 내부 표현 (7bytes)
  - Date Format에 따라 입/출력됨
  - 기본 Date Format : 'RR/MM/DD' or 'DD-MON-RR'
    - RR은 Y2K를 고려, 2자리 년도 표기 (00 ~ 49: 2000년대 / 50~99: 1900년대)
  - 포맷 확인
    - SELECT value FROM nls_session_parameters WHERE parameter = 'NLS_DATE_FORMAT';

Oracle RR Format

현재 연도	처리 연도	YY	RR
1950 – 1999	00 – 49	1900 – 1949	2000 – 2049
	50 – 99	1950 – 1999	1950 – 1999
2000 – 2049	00 – 49	2000 – 2049	2000 – 2049
	50 – 99	2050 – 2099	1950 – 1999

단일행 함수
: Date 함수

Function	Purpose
ADD_MONTHS(d,n)	d날짜에 n달 더함
LAST_DAY(d)	d의 달의 마지막 날
MONTHS_BETWEEN(d1,d2)	d1, d2사이의 달 수
NEW_TIME(d,z1,z2)	z1타임존의 d에서 z2타임존의 날짜 생성
NEXT_DAY(d,day)	d날 후의 첫 day요일의 날짜
ROUND(d,fmt)	fmt에 따른 날짜 반올림
TRUNC(d,fmt)	fmt에 따른 날짜 버림
SYSDATE	현재 날짜 시간 반환

단일행 함수
: Date 함수 사용 예

FUNCTION	RESULT
MONTHS_BETWEEN ('01-SEP-95','11-JAN-94')	19.677419
ADD_MONTHS ('11-JAN-94',6)	'11-JUL-94‘
NEXT_DAY ('01-SEP-95','FRIDAY')	'08-SEP-95‘
LAST_DAY('01-FEB-95')	'28-FEB-95‘

현재날짜를 **'25-JUL-95'**가정	
ROUND(SYSDATE,'MONTH')	01-AUG-95
ROUND(SYSDATE ,'YEAR')	01-Jan-96
TRUNC(SYSDATE ,'MONTH')	01-JUL-95
TRUNC(SYSDATE ,'YEAR')	01-JAN-95

단일행 함수
: 주요 Date 함수

- SYSDATE
  - 현재 날짜와 시간을 출력해주는 함수

```
SELECT sysdate
FROM dual;
```

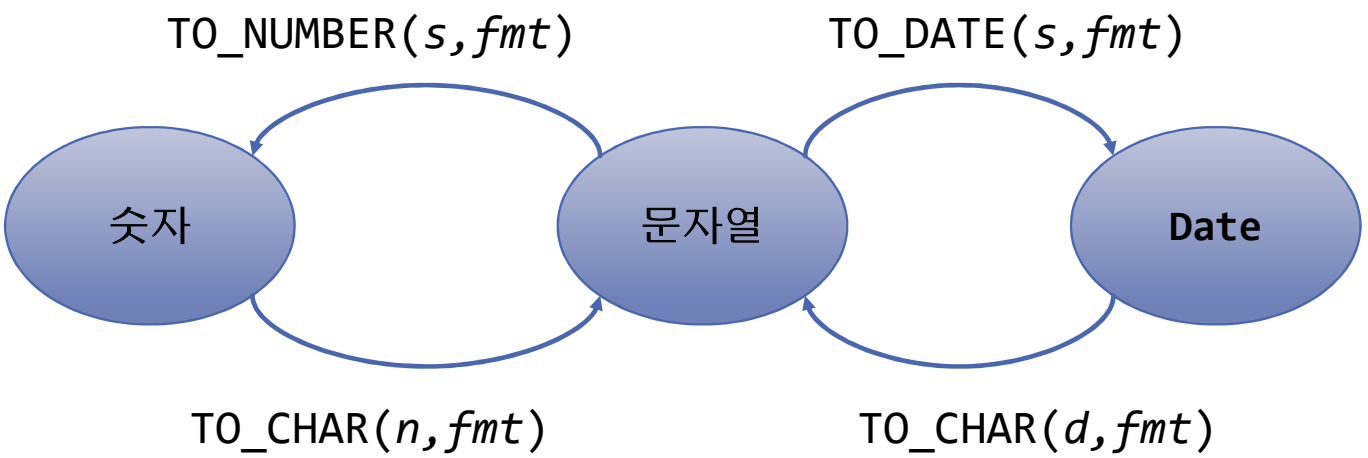
```
SELECT sysdate
FROM employees;
```

- MONTH_BETWEEN(d1, d2)
  - d1 날짜와 d2 날짜 사이의 개월수를 출력하는 함수

```
SELECT MONTH_BETWEEN(sysdate, hire_date)
FROM employees
WHERE department_id = 110;
```

단일행 함수
: 변환 함수

- 묵시적 변환 : 변환 함수 없어도 어느 정도 자동으로 변환됨
- 자동으로 변환되지 않을 때는 명시적 변환 함수를 사용



단일행 함수
: Date 변환 포맷

- 예시는 American 포맷인 경우

fmt	Description	Example
SS	Second.	0 – 59
SSSSS	Seconds past midnight.	0 – 86399
MI	Minute.	0 – 59
HH, HH24	Hour of day.	0 – 12,23
AM,PM	Meridian indicator.	AM,PM
DD	Day of month.	1 – 31
DAY	Name of day.	SUNDAY – SATURDAY
DY	Abbreviated name of day.	SUN – SAT
D	Day of week.	1 –7
DDD	Day of year.	1 – 366

fmt	Description	Example
W	Week of month.	1 – 5
WW	Week of year.	1 – 53
MM	Two-digit numeric abbreviation of month.	1 – 12
MON	Abbreviated name of month.	JAN – DEC
MONTH	Name of month.	JANUARY – DECEMBER
Q	Quarter of year.	1 – 4
RM	Roman numeral month.	I – XII
AD,BC	AD, BC Indicator.	AD, BC
Y,YY,YYY	1,2,3-digit year.	
YYYY, SYYYY	4-digit year. "S" prefixes BC dates with "-".	

단일행 함수
: Date 변환 포맷 (계속)

- 예시는 American 포맷인 경우

fmt	Description	Example
YEAR, SYEAR	Year, spelled out. "S" prefixes BC dates with "-".	
RR	Given a year with 2 digits. *Returns a year in the next century if the year is <50 and the last 2 digits of the current year are >=50.* *Returns a year in the preceding century if the year is >=50 and the last 2 digits of the current year are <50.*	
RRRR	Round year.	
CC, SCC	One greater than the first two digits of a four-digit year; "S" prefixes BC dates with "-".	
J	Julian day. the number of days since January 1, 4712 BC.	
SP	Spelled number.	
TH	Ordinal number.	

단일행 함수
: 변환 연습

- TO_CHAR(날짜, '출력형식')

```
SELECT sysdate,
       TO_CHAR(sysdate, 'YYYY-MM-DD HH24:MI:SS')
FROM dual;
```

단일행 함수
: 숫자 변환 포맷

fmt	Description	Example
9	숫자	99999
0	강제로 0 출력	09999
,	지정된 위치에 ,	99,999
.	소수점	999.99
$	$ 마크	$99999
FM	앞부분의 채움 문자(공백) 없앰.	FM90.9
L	Local 화폐단위	L99,999
MI	음수에 – 부호	99999MI
PR	음수에 괄호	99999PR
RN	로마자 (대소문자 따라 다름)	RN rn
S	부호 기호	S99,999
X	16진수	XXX xxx

단일행 함수
: 숫자 변환 포맷 연습

- TO_CHAR(숫자, '출력형식')
  - 숫자형 -> 문자형으로 변환하기

```
SELECT first_name, to_char(salary*12, '$999,999.99') "SAL"
FROM employees
WHERE department_id = 110;
```

단일행 함수
: 기타 함수

- NULL 관련
  - NVL(expr1, expr2) : expr1이 NULL이면 expr2, 아니면 expr1
  - NVL2(expr1, expr2, expr3) : expr1이 NULL이면 expr3, 아니면 expr2
  - NULLIF(expr1, expr2) : 두 식이 같으면 NULL, 아니면 expr1
  - COALESCE(expr1, expr2, … exprN) : 첫 NOT NULL인 식 반환, 없으면 exprN
- 데이터 타입에 유의한다
- 예
  - ```
    SELECT ename, NVL(TO_CHAR(mgr), 'No Manager')
      FROM emp;
    ```
    ```

Conditional Expression

- CASE

```
CASE {expr1} WHEN {expr2} THEN {expr3}
               [WHEN {expr4} THEN {expr5}
                   ...
               ELSE {expr6}]
```

  - IF – THEN - ELSE와 비슷한 로직을 제공
  - expr1이 expr2와 일치할 때 -> expr3
  - expr1이 expr4와 일치할 때 -> expr5
  - 만족하는 조건이 없으면 -> expr6

- DECODE

```
DECODE( {value}, {if1}, {then1}
               [, {if2}, {then2}]
               , {else} )
```

Conditional Expression
: Example

- CASE
  - ```
    SELECT ename, job, sal, CASE job WHEN 'CLERK' THEN 1.10 * sal
                               WHEN 'MANAGER' THEN 1.15 * sal
                               WHEN 'PRESIDENT' THEN 1.20 * sal
                               ELSE sal END REVISED_SALARY
    FROM emp;
    ```

- DECODE
  - ```
    SELECT ename, job, sal, DECODE(job, 'CLERK', 1.10 * sal,
                                   'MANAGER', 1.15 * sal,
                                   'PRESIDENT', 1.20 * sal,
                                   sal) REVISED_SALARY
    FROM emp;
    ```

Conditional Expression
: Example

- [연습] hr.employees
  - 직원의 이름, 부서, 팀을 출력하십시오
  - 팀은 코드로 결정하며 다음과 같이 그룹 이름을 출력합니다
    - 부서 코드가 10 ~ 30이면: 'A-GROUP'
    - 부서 코드가 40 ~ 50이면: 'B-GROUP'
    - 부서 코드가 60 ~ 100이면 : 'C-GROUP'
    - 나머지 부서는 : 'REMAINDER'