180430
데이터베이스에서 데이터를 불러오는 언어 SQL - 오라클

---

**◆ 오라클 & 오렌지**

수업에서 사용하는 오라클 버전
오라클 g11r2 32bit / 왜냐하면 오렌지가 32bit랑만 호환!

※ 오라클 재설치시 삭제해도 깔끔하게 제거되지 않음. 완전히 삭제하고 같은 곳에 설치보다,
충돌 위험이 있으므로 새로 설치할 때 다른 폴더에 저장하는 형태로 재설치!

※ 오렌지 : 60일 기간 제한. 재설치하면 바로 trial 기간 종료
컴퓨터 날짜를 조정하는 식으로 사용하는 방법을..!

---

**◆ 오라클 설치 과정 in 수업**

- "서버클래스" 선택
- 4th "단일 인스턴스" 선택 / RAC (Real Application Clusters) - 여러 PC에 데이터 분산(저장,접근)
- 5th "고급설치"
- 7th 옵션선택 모두 해지 / 저장공간 부족시
- 11th 샘플스키마 체크 선택 / 가볍게 테스트할 sample data set

+비밀번호 : oracle

※ 설치 완료 후 뜬 창 닫지 말고!
'비밀번호 관리' 에서 <SCOTT, HR> 계정 잠금 해제!
/ 그렇지 않을 경우 시작할 때마다 명령어 입력해서 잠금 해제 해야 함

◆ **계정잠금 해제, 패스워드 관련 명령어**

① 시작 - cmd
② sqlplus / as sysdba;
#최상의 권한을 가진 슈퍼유저 : sysdba
패스워드변경, 권한해제 등
③ alter user scott account unlock;
유저명
alter user scott identified by oracle;
패스워드
④ alter user hr account unlock;
alter user hr identified by oracle;
⑤ exit

◆ **계정명 scott 접속 / in sqlplus (다이렉트접속)**
sqlplus scott/oracle
계정명/패스워드
# 접속 이후 db의 데이터 관리 가능

그런데 sqlplus로 보이는 데이터가 예쁘지 않음 -> 보기 쉽게 표현해주는 개발툴 사용

◆ **계정명 scott 접속 / in 오렌지 (원격접속)**
① Oracle home
OraDb11g_home1 선택
② TNS Name ; DB이름
ORCL (설치 때 정한 이름)
③ username, password
scott, oracle (접속하고자 하는 계정이름과 비밀번호)
④ Connect 버튼 클릭

**Chapter 1 SELECT : 데이터의 조회**

1장. 데이터의 조회

1. DESC

---

**DESC 테이블명 ;**

---

-테이블 : 데이터를 구분짓기 위한 최소단위

-테이블의 포맷을 확인하는 명령어

-해당 테이블의 컬럼이름, 컬럼의 Null여부, 컬럼의 데이터 타입을 조회 가능

cf. db설계 - 모델러

-> 테이블은 설계될 때 이미 포맷 정의가 되어있음. (vs 엑셀)

= 테이블 이름, column이 이미 정의됨 (분류, 이름, 개수 등), 데이터 유형 미리 정의.

for 틀린 데이터가 못 들어 오게! null값의 여부 조건도!

유형

숫자 - NUMBER

문자 - VACHAR2, CHAR

날짜 - DATE

* 고정형 문자 DB - 일정한 고정 SIZE ex. CHAR(13) : 어떤 데이터가 들어와도 무조건 13자리로 입력됨. 13으로 설계하고 6자리를 입력하면 나머지를 공백으로 채워서 무조건 맞춤

=> trim( column ) 사용

* 가변형 문자 DB - SIZE가 달라질 수 있는 형태 ex.VARCHAR2(20) : 최대 20byte, 자유로움

1. SELECT - 데이터 조회하기

[SELECT 기본 문법]

---

**SELECT [DISTINCT] {* | 컬럼명 | 표현식}**

**FROM 테이블명 또는 뷰명**

---

- ***** : 전부
- **컬럼명** (from desc) : 여러개인 경우 **,** 로 구분
- **표현식** : 단독컬럼으로 구성되어있지 않은 모든 것 - 변형[컬럼끼리 결합, 연결 등]
  for 원본 데이터를 가공

#데이터베이스 내에 저장되어있는 데이터를 조회할 수 있는 언어

#ctrl+enter : line별실행

#전체실행 : F5

#오른쪽마우스 - show grid as a excel file

#정형화된 데이터는 반드시 db에서 화면에 출력할 수 있게 sql 언어를 통해서 불러와야.
비정형 ex. 그림, 소리, 영상

#select는 절대로 원데이터 훼손 X

**1) 모든 컬럼 조회하기 ( * )**

---

```
select *
  from emp;
```

---

#emp : 샘플스키마의 emp 테이블, scott 계정 안에 있는 데이터 - 타 계정에 있는 데이터는 불가

**2) 원하는 칼럼만 조회하기 (원하는 컬럼 이름들을 SELECT 절에 명시)**

---

```
select EMPNO, ENAME, SAL
  from emp;
```

---

-조회하기를 원하는 칼럼이 여러 개일 경우 콤마(,)로 구분

**3) 표현식을 사용하여 출력하기**

---

```
select EMPNO, ENAME, SAL, 1, sal+100, 'very good~~'
  from emp;
```

---

-문자열(=기존 db상에서 정의된 단위가 아닌 형식! 일반 텍스트)은 ' '를 통해 표현!

ex. EMPNO vs 'EMPNO'

-표현식 역시 콤마(,)로 구분

#SELECT : ***가 아닌 이상 다 칼럼으로 인식! 칼럼만 들어갈 수 있는** 자리.

칼럼이 있으면 불러옴, 칼럼으로 없으면 전체 행에 대해 모두 동일하게 표현됨 like 칼럼.

**4) 칼럼 별칭 사용하여 출력하기**

---

```
select EMPNO, ENAME 이름, SAL, sal*1.1 as "10% 인상연봉"
  from emp;
```

---

**컬럼 별칭(Alias)**의 정의

-컬럼의 원래 이름이 아닌 별칭을 의미

-주로 연산이 수행된 컬럼에 사용됨

-컬럼명 다음에 as를 쓴 뒤 그 뒤에 별칭 명시(as는 생략 가능)

-**특수문자**나 **공백**을 컬럼의 별칭으로 사용하려면, 별칭에 반드시 큰 따옴표(" ") 사용 필요

-> 특수문자 없을 때 그냥 써도 되긴 함!

-select의 alias는 오직 order by에서만 사용 가능(순서상)

cf. from절의 알리아스는 "" 사용 불가

#원래 컬럼 이름을 변경하는 것이 아닌, 출력될 때 임시로 바꾸어서 보여주는 것 뿐!

**5) 연결(합성) 연산자(Concatenation)로 컬럼들을 붙여서 출력하기**

---

```
select empno || '/' ||ename|| '/' ||sal
  from emp;
```

---

-서로 다른 컬럼을 마치 하나의 컬럼인 것처럼 연결해서 출력해야 할 경우

연결 연산자 (||)를 이용해서 표현

-함수(+R,Python)로는 두 개만 가능, 따라서 DB 작업시 미리 합치는 걸 더 추천

-ex. concat('aa', 'bb') => 여러 개 사용시 중첩해서 사용하므로 불편.

**6) DISTINCT - 중복된 값을 제거하고 출력하기**

select **distinct** {* / 컬럼명 / 표현식}
from 테이블명 또는 뷰명

-distinct: 중복된 행의 값을 제거하는 것
#distinct는 항상 select 뒤에, column 앞에 써야 함
#SQL에서 중복 처리한 후 R/Python 처리가 훨씬 빠름! 따라서 sql 안에서 하도록!

---
select distinct job
  from emp;
---
select distinct job, deptno
  from emp;
---
#행의 데이터 차원에서만 중복제거 할 수 있다.
#여러 개의 column을 선택한 경우
job도 중복제거, deptno도 각각 중복제거 된 값을 한 행에 쓸 수는 없음.
 여러 개는 결합한 column의, 즉 둘 다(and)를 보아서 중복을 따짐

# distinct 뒤에 오는 column의 완벽하게 구별되는 경우만 나타내줌
# column 순서가 달라도 같은 결과! 순서는 중요하지 않음

◆ **실습예제 문제**

**1. DEPT 테이블을 사용해서 deptno를 부서#, dname을 부서명, loc를 위치로 별명을 설정하여 출력하세요. (Alias)**

**SOL)**

```
select deptno as 부서#, dname as 부서명, loc as 위치
  from dept;
```

# ""를 쓰는 특수문자가 따로 있음. 기능이 정의된 특수문자만(ex, *, 공백 등) ""사용
따라서 부서#는 ""를 사용하지 않아도 에러 안 뜸!

**2. EMP테이블에서 부서별(deptno)로 담당하는 업무(job)가 하나씩 출력하도록 하여라. (distinct)**

**SOL)**

```
select distinct deptno, job
  from emp;
```

#desc 테이블명;
처음 순서에 하면 칼럼명 살펴보기 편함 !

**3. 학생 테이블(student)을 사용하여 모든 학생들이 '서진수 의 키는 180 cm, 몸무게는 55 kg 입니다' 와 같은 형식으로 출력되도록 하고, 칼럼이름은 "학생의 키와 몸무게"라는 별명으로 출력해보세요. (연결 연산자)**

**SOL)**

```
select name|| '의 키는 ' ||height|| ' cm, 몸무게는 ' ||weight|| ' kg 입니다' as "학생의 키와 몸무게"
    from student;
```

**4. 교수 테이블(professor)을 사용하여 교수의 이름과 직급이 아래와 같이 홍길동 (교수), 홍길동 '교수' 이렇게 나오도록 출력해보세요. 출력된 컬럼 이름은 교수님입니다.**

**SOL)**

```
  select name||'('||position||'), '||name|| '''' ||position|| '''' as "교수님"
    from professor;
```

★ '을 문자열로 쓰려면 ''''으로 표현
# '에 대해서 '''로 하면 '를 문자열처럼 나타낼 수 없음 /싱글따옴표의 짝 때문..

**7) WHERE절을 활용하여 원하는 조건만 조회하기**

**SELECT [Column or Expression]**
**FROM [Table or View]**
**WHERE 조건식;**

- 역할 : 조건을 만족하는 **데이터**만 추려서 조회하고 싶을 때

```
------------------------------
select name, height
   from student
 where height >= 170;
------------------------------
```

**- 문자, 날짜 타입의 칼럼의 형식을 조회할 경우 꼭 홑따옴표(')를 사용해야 함.**

- 숫자에 홑따옴표를 해도 되지만 치명적인 실수가 있을 수 있음.
시스템에서 숫자로 바꿔줘서 나온 결과. (과정이 복잡하므로 성능적으로 안 좋은 영향을 미침)
-> 따라서 잘 구분할 것 !

**◆ 문자**

```
-------------------------
 select ename, sal
   from EMP
 where ename = 'SMITH';
---------------------------
```

**- 문자 in data : 유일하게 대소문자 구분되는 지점**
(not column)

```
----------------------------
 select ename, sal
    from EMP
   where upper(ename) = upper('smith');
----------------------------
```

★★★
문자의 경우 대소문자를 구분함. **upper, lower로 치환**하면 테이블을 직접 조회하지 않아도 가능

◆ **날짜**

```
------------------------------
select *
  from EMP
 where hiredate > '1982/01/01';
------------------------------
```

- 날짜 : 포맷이 까다로움. **저장되어 있는 형태와 일치해야 가능**! 그런데 어떤 포맷인지 데이터를 일일이 조회하기 어려움

--> **to_char를 통해 확실하게 함**

**8) SQL에서 다양한 연산자 사용하기**

| 연산자 종류 | 설명 |
|---|---|
| = | 같은 조건을 검색 |
| !=, <> | 같지 않은 조건을 검색 |
| > | 큰 조건을 검색 |
| >= | 크거나 같은 조건을 검색 |
| < | 작은 조건을 검색 |
| <= | 작거나 같은 조건을 검색 |
| BETWEEN a AND b | A와 B사이에 있는 범위 값을 모두 검색 (A,B 포함) |
| IN(a,b,c) | a이거나 b이거나 c인 조건을 검색(OR) (A,B,C 포함) |
| Like | 특정 패턴을 가지고 있는 조건을 검색 |
| Is Null / Is Not Null | Null값을 검색 / Null이 아닌 값을 검색 |
| A AND B | A조건과 B조건을 모두 만족하는 값만 검색 |
| A OR B | A조건이나 B조건 중 한가지라도 만족하는 값을 검색 |
| NOT A | A가 아닌 모든 조건을 검색 |

ex) not between a and b

not like ~~

not in(a,b,c,)...

**+ BETWEEN, IN : 문자, 날짜 사용 가능 -> ' ' 홑따옴표 꼭 붙여서 넣어야 함**

**+ 문자 대소문자 변경**

180501

◆ **단축키**

**ctrl + shift + ₩ : 뒷부분 드래그 단축키**

**ctrl + u : 소문자 변경**

**ctrl + shift + u : 대문자 변경**

**ctrl + shift + f : sql 문장 포맷 변경**

**ctrl + - : 주석처리**

**ctrl + shift + - : 주석해제**

(1) 산술연산자 사용하기

*문자 : 데이터는 대/소문자 구분 => upper/lower 함수로 치환하여 사용

*날짜 포맷 : not 공용. 기업마다 선호하는 날짜 유형 다름 yy mm dd 왔다갔다.. ~ 일일이 확인할 수 없어서 하나로 맞추는 함수 살필 예정

----------성능 관련 참조----------

```
 select *
   from EMP
 where sal > 5000/1.1;
```

---

★더 성능이 좋음★

★column을 가공하지 않는 편이 출력 관련 성능이 좋다★

비교 - 같은 결과지만 더 느림

---

```
 select *
   from EMP
 where sal*1.1 > 5000;
```

---

(2) BETWEEN 연산자를 사용하기

---

```
select *
 from EMP
 where sal between 1000 and 1600;
```

---

- BETWEEN : 숫자 데이터에 많이쓰임
- A, B을 포함한 결과 값이 출력됨

# select : 열을 추리려는 목적
# where : 행을 추리려는 목적

(3) IN 연산자로 여러 조건을 간편하게 검색하기

---

```
 select *
 from EMP
 where job in ('CLERK', 'SALESMAN');
```

---

- IN : 문자, 범주형 데이터에 많이 쓰임

-문자 : '홑따옴표'로 표시!!!.
-원데이터 대/소문자-> 치환(upper, lower)으로도 데이터를 살펴보지 않고도 가능!

# where job = 'CLERK' or job = 'SALESMAN' 과 같은 in 연산자,
하지만 조건이 더 많아지는 경우 계속 반복해서 써야 하는 불편함.. 따라서 in 연산자 사용!

(4) LIKE 연산자로 비슷한 것들 모두 찾기

-**패턴**만으로도 검색할 수 있는 기호

- **%** : 글자수 제한이 없다(0개 포함).
- _ (underscore) : _ 하나에 한 글자만 올 수 있음

◆ **like 예제**

1) 이름이 A로 시작하는 직원 모두 출력
: ename **like 'A%'**
2) 이름이 A로 시작하는 4자리인 직원 모두 출력
: ename **like 'A___'**
3) 이름에 O가 포함된 직원 모두 출력
: ename **like '%O%'**
4) 이름이 E로 끝나는 직원 모두 출력
: ename **like '%E'**
5) 두 번째 이름이 E인 직원 모두 출력
: ename **like '_E%'**

* **싱글따옴표** 붙이는 것 잊지 말기 문자형이니까!
* **대문자/소문자구분** 역

(5) **IS** NULL / **IS NOT** NULL 연산자를 활용하기

- NULL : 사용할 수 없고, 지정되지 않고, 적용할 수 없는 값. 0과 space와는 또 다른 개념.
- **is** 형태로 연산자 작성하는 것 기억하기!

**※ 산술연산에서 연산하고자 하는 값 중 하나라도 NULL값을 가지면 무조건 결과로 NULL이 출력**

```
--------------
 select ename, sal, comm, sal + comm as 총연봉
   from EMP;
----------------
ename   sal      comm      총연봉
SMITH   800
ALLEN   1600     300       1900
WARD    1250     500       1750
JONES   2975
MARTIN  1250     1400      2650
BLAKE   2850
```

=> comm이 null인 경우, sal이 있음에도 총연봉은 null이 출력됨

---
```
select ename, sal, comm, sal + comm as 총연봉
   from EMP
where comm is not null;
```
---

(6) 논리연산자 활용하기

**- 우선순위 NOT > AND > OR**

- AND나 OR을 먼저 사용하게 하고 싶으면 ( ) 괄호로 먼저 묶기!
- where절의 조건을 여러 개 지정할 때 사용

**NOT + IN / BETWEEN / LIKE**

**◆ where절 실습문제**

1. EMP테이블에서 급여(sal)가 3000 이상인 사원의 사원번호, 이름, 업무, 급여를 출력하라.

**sol)**

```
 select empno, ename, job, sal
   from emp
 where sal >= 3000;
```

2. emp테이블에서 담당업무가 manager인 사원의 정보를 사원번호, 성명, 업무, 급여, 부서번호를 출력하라.

**sol)**

```
select empno, ename, job, sal, deptno
 from emp
where job = 'MANAGER' ;
```

3. emp테이블에서 1982년 1월 1일 이후에 입사한 사원의 사원번호, 성명, 업무, 급여, 입사일자, 부서번호를 출력하여라.

**sol)**

```
select empno, ename, job, sal, hiredate, deptno
 from emp
where hiredate >= '1982/01/01' ;
```

◆ **연산자 실습문제 - emp 테이블**

1. 급여가 1300에서 1700 사이인 사원의 성명, 업무, 급여, 부서번호를 출력하라.

**sol)**

```
select ename, job, sal, deptno
 from emp
where sal between 1300 and 1700 ;
```

2. 사원번호가 7902, 7788, 7566인 사원의 사원번호, 성명, 업무, 급여, 입사일자를 출력하라.

**sol)**

```
select empno, ename, job, sal, hiredate
 from emp
where empno in (7902, 7788, 7566) ;
```

**3. 입사일자가 82년도인 사원의 사원번호, 성명, 업무, 급여, 입사일자, 부서번호를 출력하라.**

**sol)**

```
select empno, ename, job, sal, hiredate, deptno
 from emp
where hiredate between '1982/01/01' and '1982/12/31';
```

* where hiredate **like** '1982%' => 안나옴.. 정답은 where hiredate **like '82%'**

--- **like : 원데이터 형식과 정확하게 일치해야 함** yyyy-mm-dd가 아닌 yy-mm-dd의 data 원본.

sqlplus 에서 보면 **원본 데이터의 날짜 형태**가 나타남. 오렌지에선 변경된 날짜 형식.

**cf)** 1월에 입사한 사람을 like로 찾으려면? -> where hiredate **like '__/01/__'**

/ like로 날짜를 접근하는 건 정확한 방법이 아님을 염두하기!

4. 이름의 첫 글자가 'M'인 사원의 이름, 급여를 조회하라.

**sol)**

```
select ename, sal
  from emp
 where ename like 'M%';
```

5. 이름의 두 번째 글자가 'L'인 사원의 이름, 업무를 조회하라

**sol)**

```
select ename, job
  from emp
 where ename like '_L%';
```

6. 보너스가 null인 사원의 사원번호, 이름, 업무, 급여, 입사일자, 부서번호를 출력하여라.

**sol)**

```
select empno, ename, job, sal, hiredate, deptno
  from emp
 where comm is null;
```

7. 급여가 1100 이상이고, job이 manager인 사원의 사원번호, 성명, 담당업무, 급여, 입사일자, 부서번호를 출력하라.

**sol)**

```
select empno, ename, job, sal, hiredate, deptno
  from emp
 where sal >= 1100
   and job = 'MANAGER';
```

8. 급여가 1100 이상이거나, 이름이 M으로 시작하지 않는 사원의 사원번호, 성명, 담당업무, 급여, 입사일자, 부서번호를 출력하여라.

**sol)**

```
select empno, ename, job, sal, hiredate, deptno
  from emp
 where sal >= 1100
    or ename not like 'M%';
```

9. job이 manager, clerk, analyst가 아닌 사원의 사원번호, 섬여, 업무, 급여, 부서번호를 출력하여라.

**sol)**

```
select empno, ename, job, sal, deptno
  from emp
 where job not in ('MANAGER','CLERK','ANALYST');
```

10. emp 테이블에서 job이 president이고, 급여가 1500이상이거나 업무가 salesman인 사원의 사원번호, 이름, 업무, 급여를 출력하여라.

**sol)**

```
select empno, ename, job, sal
  from emp
 where job = 'PRESIDENT'
   and sal >= 1500
    or JOB = 'SALESMAN';
```

**8) order by 절을 사용하여 출력 결과 정렬하기**

**ORDER BY 컬럼1 [ASC | DESC] , 컬럼2 [ASC | DESC] ...**

- ASC가 기본값, 생략 가능.
- ASC : 오름차순, DESC : 내림차순
- 컬럼1 순으로 정렬 -> 컬럼1이 동일한 경우 컬럼2 순으로 정렬

#cf ) desc 테이블명;
#메모리 많이 잡는 경우도, 필요할 때만 쓰기

[정렬 순서] (오름차순)
- 한글 : 가, 나, 다, 라, ...
- 영어 : A, B, C, D, ...
- 숫자 : 1, 2, 3, 4, ...
- 날짜 : 예전 날짜부터 최근 날짜로

---

```
select empno as 사원번호, ename, sal as salary
 from emp
 where deptno = 10
 order by 사원번호 desc , 2 ;
```

---

1. Alias **사용 가능**
2. **select 절에 작성한 column의 순서로 column 표시 대체 가능함**! - column 이름이 긴 경우 (원래는 COLUMN 이름 기술이 원칙)

* **select**의 **Alias는 오로지 order by**에서만 Alias사용 가능. where에서는 안됨..!!!
/오라클이 각 구문을 해석하는 순서 상 select절 이후는 오직 order by 만!

#쓰는 순서 뒤바뀌면 안 됨! select - from - where - order by

**◆ order by 실습문제 - emp 테이블**

1. 가장 최근에 입사한 순으로 사원번호, 이름, 업무, 급여, 입사일자, 부서번호를 출력하여라.

**sol)**

```
select empno, ename, job, sal, hiredate, deptno
   from EMP
 order by hiredate desc;
```

2. 부서번호로 정렬한 후, 부서번호가 같을 경우 급여가 많은 순으로 정렬하여 사원번호, 성명, 업무, 부서번호, 급여를 출력하여라.

**sol)**

```
select empno, ename, job, deptno, sal
  from emp
order by deptno, sal desc;
```

3. 부서별로 담당하는 업무를 한 번씩 조회하라. 단, 업무 기준으로 정렬해서 나오도록 한다.

**sol)**

```
select distinct deptno, job
  from emp
order by job;
```