7_18_2022__Oracle

create 객체를 만들 때

drop 객체를 삭제할 때

alter 객체를 변경할 때

—user, table…

insert, update, delete

select(5)

- from(1) 어떤 대상으로 부터 (table name or select-sub query)
- where(2) 조건절 - 원하는 row를 추려내는 역할
- group by (3?) 특정 row 를 그룹으로 묶는다(그 그룹은 하나의 row로 묶인다.) 그룹을 대표하는 값만 select 가능
- having(4?) 묶은 그룹중에 정보를 추려내는 역할 avg(), sum(), max() …
- order by(last) select 해 온 정보를 정렬

```
ex) select empno, ename, job
from emp
where empno > 7800;
---
select deptno, count(*), max(sal), min(sal)
from emp
group ny deptno
having mas(sal)>=3000;
```

join

- 두개 이상의 table을 합쳐서 정보를 출력하기
- 정보를 가져올 곳, 어떤 정보를 가져올지 확실하게 알려줘야함
  - 정보를 가져올 곳을 확실하게 정하지 않으면 출력안됨
  - 어떤 정보를 확실하게 지정하지 않으면 모든 정보가 출력됨

```
select ename, emp.deptno, dept.deptno, dname, loc
from emp, dept
where emp.deptno = dept.deptno;
```

- ansi join
  - where에 join조건을 사용하면 가독성이 떨어질 수 있어서 사용한다.

```
select ename, dept.deptno, dept.deptno, dname, loc
from emp
join dept on emp.deptno = dept.deptno
```

- 같은 값을 불러온다면 using을 사용하면 더 간단해짐

```
select ename, deptno, deptno, dname, loc
from emp
join dept using(deptno)
```

- 여러개의 table을 한번에 join 할 상황도 있음

```
emp, dept, salgrade 모두에서 정보를 얻어와야 하는 상황

  select ename, dname, grade
  from emp, dept, salgrade
  where emp.deptno=dept.deptno
  and (sal between losal and hisal);

where절 and을 사용해 다음 조건을 걸어준다.

ansi join
  select ename, dname, grade
  from emp
  join dept using(deptno)
  join salgrade on sal between losal and hisal;
```

- self join

```
select e1.empno, e1.ename, e1.mgr, e2.ename
from emp e1, emp e2
where e1.mgr=e2.empno;

ansi join
  select e1.empno, e1.ename, e1.mgr, e2.ename
  from emp e1
  join emp e2 on e1.mgr = e2.empno
```

- outer join

```
select ename, dept. deptno, dname
from emp, dept
where emp.deptno(+)=dept.deptno;

ansi join
  select ename, deptno, dname
  from emp
  right outer join dept using(deptno);
```

여러 정보를 여러페이지에 나누어 출력할 때

1. 정렬

```
정렬된 테이블
  from(
    select empno, ename, sal
    from emp
    order by sal asc) result1
```

2. 행번호 부여

```
select result1.*,rownum
from(select empno, ename, sal
     from emp
     order by sal asc) result1
```

3. SELECT

```
select *
from
  (select result1.*,rownum as rnum
  from
    (select empno, ename, sal
    from emp
    order by sal asc) result1)
where rnum between 4 and 6
```

CREATE

제약조건을 만들 때 이름을 붙이지 않으면 system이 알아서 붙인다.

user_constraints로 제약조건에 대한 정보를 불러올 수 있음.

가독성이 좋지 않기 때문에 이름을 붙여주는게 좋음

not null - c primary key -p forign key -r

constraint 사용 이름 정해주기

```
create table member(
num number constraint member_num_pk primary key,
name varchar2(12) constraint member_name_nn not null,
addr varchar2(16));
```

제약조건 check

check 의 조건이 true 일때만 추가 가능.

```
create table dept2(
deptno number(2) constraint dept2_deptno_pk primary key,
dname varchar2(12) constraint dept2_dname_nn not null,
loc varchar2(16) constraint dept2_loc_ck check(loc in('seoul','busan')));

-----
  SQL> insert into dept2
  2  (deptno, dname, loc)
  3  values(10, 'sales', 'seoul');
    1 개의 행이 만들어졌습니다.

  SQL> insert into dept2
  2  (deptno, dname, loc)
  3  values(10, 'sales', 'hanyang');
    insert into dept2
    1행에 오류:
    ORA-02290: 체크 제약조건(SCOTT.DEPT2_LOC_CK)이 위배되었습니다

  SQL> insert into dept2
  2  (deptno, dname, loc)
  3  values(20, 'sales', 'seoul');
    1 개의 행이 만들어졌습니다.
```

테이블 수준의 제약조건

not null 은 테이블 레벨로는 불가

create table dept2(

```
create table dept2(
deptno number(2),
dname varchar(12) constraint dept2_dname_nn not null,
loc varchar2(12),
constraint dept2_deptno_pk primary key(deptno),
constraint dept2_loc_ck check(loc in('seoul','busan')));
```

```
create table emp2(
empno number(4),
ename varchar2(12) constraint emp2_ename_nn not null,
deptno number(2),
constraint emp2_empno_pk primary key(empno),
constraint emp2_deptno_fk foreign key(deptno) references dept2(deptno))
```

테이블에 칼럼 추가

```
alter table 테이블명 add(칼럼명 데이터타입);
```

제약조건 추가

```
alter table member
add constraint member_num_pk primary key(num);
```

데이터 수정

```
alter table 테이블명
modify(칼럼명 데이터타입(xx));
```

칼럼명 변경

```
alter table member
rename column addr to my_addr;
```

칼럼 삭제

```
alter table member
drop(my_addr);
```

Not Null 변경하기

```
alter table member
modify(name constraint member_name_ not null);
```

시퀀스 만들기(시퀀스 - 연속되는 숫자값을 자동으로 부여)

```
create sequence test_seq;

시퀀스 함수 사용
  select test_seq.nextval from dual;
현재 시퀀스 확인
  select test_seq.currval from dual;

시퀀스도 테이블로 관리가 됨
  desc user_sequences
```

시퀀스 초기설정이 가능

```
create sequence my_seq
increment by 10
start with 100
nocache;
```

만들어진 시퀀스는 수정이 불가

테이블과 시퀀스의 조화

```
1.테이블 만들기
  create table member(
  num number constraint member_num_pk primary key,
  name varchar2(16) constraint member_name_nn not null,
  addr varchar2(20))

  테이블이 생성되었습니다.
  SQL> desc member
   이름                                      널?      유형
   ----------------------------------------- -------- ----------------------------
   NUM                                       NOT NULL NUMBER
   NAME                                      NOT NULL VARCHAR2(16)
   ADDR                                               VARCHAR2(20)

2.시퀀스 만들기
  SQL> create sequence member_seq;
  시퀀스가 생성되었습니다.

3. 시퀀스를 밸류값으로 활용하기
  SQL> insert into member
```

```
    2  (num, name, addr)
    3  values(member_seq.nextval, '두부', '페어펙스');
  1 개의 행이 만들어졌습니다.

  SQL> insert into member
    2  (num,name,addr)
    3  values(member_seq.nextval, '유키', '하이데저트');
  1 개의 행이 만들어졌습니다.

  SQL> insert into member
    2  values(member_seq.nextval, '소주', '리치몬드');
    1 개의 행이 만들어졌습니다.

4. 결과
  SQL> select * from member;

         NUM NAME  ADDR
  ---------- ----- -------------
           1 두부  페어펙스

           2 유키  하이데저트

           3 소주  리치몬드
```

다른 테이블을 똑같이 복사하기

```
create table member2(
2  num number,
3  name varchar2(16),
4* addr varchar2(26))
insert into member2 select * from member;
```

테이블을 만듦과 동시에 똑같이 만들 수도 있음

```
create table membe3 as select * from member;
```

테이블의 구조만 복사(데이터는 없음)

```
create table emp2 as select * from emp where 1=2;
```

**제약조건은 복사가 안됨

스칼라 타입(오라클 데이터형

- char : 고정 길이의 문자, 최대 2000byte

- varchar2 : 가변 길이의 문자, 최대 4000byte
- number : 숫자값을 -38자리수 부터 +38자리수를 저장 가능
  - number(10) 정수 10자리
  - number(10,2) 전체 자리수 10 자리 소수점 이하 2자리
  - number(p,s) p는 전체자리수 s는 소수점 이하 자리수
- clob : 문자 데이터 최대 4gm까지 저장 가능
  - jdbc에서 읽어올 때 getString()으로 읽어올 수 없다(10g버전부터 가능) getClob() 으로 읽어와야함
  - date : 날짜(시강)
  - blob : 2진 데이터, 즉 바이너리 데이터를 저장할 때 사용한다.