

2

SQL_실습

DDL (데이터 정의어) - CREATE, ALTER, DROP

- 식별자 기술할 때 규칙
 - 따옴표 없는 식별자 : 숫자, \$, #는 첫 글자에 올 수 없다
 - 따옴표 있는 식별자 : 어떤 문자 사용 가능 → 큰 따옴표는 사용 불가능
 - 하나의 네임스페이스 안에 서로 다른 두 객체가 동일한 이름 가질 수 없음
- VIEW
 - 뷰는 실제 데이터에 포함되지 않는다.
 - 테이블과 같은 네임스페이스를 사용하므로 스키마 내 다른 이름과 중복되면 안된다.
 - WITH CHECK OPTION
 - 기본 테이블에서 정보가 추출하는 것이므로 조건에 사용 되어진 컬럼값은 뷰를 통해서는 변경이 불가능
 - 만약 뷰를 생성할 때, 부서 번호가 20일 사원 정보만 추출했다면 해당 뷰로 부서 번호를 40번으로 변경할 수 없게됨
- SEQUENCE
 - 기본키 또는 유일키에 값을 넣을 때 사용
 - WHERE문 : CACHE → 미리 할당(DEFAULT=20)
 - 원래 있는 SEQUENCE : USERS_SEQUENCES
 - NEXTVAL, CURRVAL : NEXTVAL은 시퀀스증가, CURRVAL은 현재시퀀스로 증가
 - PRIMARY KEY는 CURRVAL는 쓸 수 없음
- CREATE INDEX : null, 테이블내에서 여러 개 생성 가능, index(object)

↔ primary key : not null, 하나만 생성 가능, constraint(제약조건) == 중복될 수 없는 유일값

- System privilege : db관리자는 사용자에게 시스템 권한을 부여하여 사용자는 특정 작업 수행 가능
 - Create session : 데이터베이스 연결 허용
 - Create table : 테이블, 인덱스 생성, connect, dml, drop, alter, truncate 가능
 - Select any table : 모든 스키마에 모든 테이블,뷰 쿼리 사용 가능
- Grant system privilege
 - 권한 부여할 새로운 사용자 생성

```
CREATE USER scott  
IDENTIFIED by tiberio;
```

→ 새로운 user scott생성 user password tiberio

```
GRANT CREATE SESSION, CREATE TABLE TO scott;
```

→ scott 사용자의 스키마에 테이블을 생성할 수 있는 권한 부여

```
GRANT ALTER ANY TABLE TO scott;
```

→ scott 사용자에게 스키마의 테이블을 변경할 수 있는 권한 부여

- 시스템 권한 확인

```
DESCRIBE dba_sys_privs;
```

- Object privileges

→ db관리자가 사용자에게 부여할 수 있는 객체 권한 (table, view, sequence, procedure)

- Grant object privilege : db 관리자는 사용자 및 역할 권한 부여 및 취소 가능

```
GRANT SELECT ON s_emp TO scott;
```

→ scott에게 s_emp 테이블 조회할 수 있는 권한 부여

```
GRANT INSERT(empno, ename, deptno),  
UPDATE(sal)  
ON s_emp TO scott;
```

→ scott에게 s_emp 테이블에 직원 번호, 이름, 부서 번호를 삽입할 수 있는 권한과 월급 수정할 수 있는 권한 부여

```
GRANT SELECT, INSERT  
ON s_dept  
TO scott  
WITH GRANT OPTION;
```

→ scott에게 s_dept 테이블 조회할 수 있는 권한과 다른 사람에게 동일한 권한을 부여할 수 있는 권한 부여

```
conn scott; #scott으로 사용자 변경
```

```
GRANT SELECT  
ON sys.s_dept  
TO PUBLIC;
```

→ scott이 시스템의 모든 사용자에게 sys의 s_dept 테이블을 조회할 수 있는 권한 부여

```
DESCRIBE user_tab_privs_made  
#grantee : 권한 가진 사용자  
#table_name : 객체 이름  
#grantor : 권한 준 사람  
#privilege : 가능한 권한  
#grantable : 다른 사용자에게 나눠줄 수 있는지
```

```
SELECT *  
FROM user_tab_privs_made  
WHERE GRANTEE='SCOTT';
```

→ 현재 사용자가 scott 에게 부여된 객체 조회

```
conn scott;  
  
SELECT *  
FROM user_tab_privs_recd;
```

→ 사용자 scott이 자신에 부여된 객체 조회

- CREATE SYNONYMS. : 사용자가 소유한 스키마나 다른 스키마에 속하는 동의어 생성

```
GRANT CREATE SYNONYM TO scott; #scott에게 권한 부여
```

```
conn scott;
CREATE SYNONYM s_dept
FOR sys.s_dept; . #synonym 생성
```

- REMOVE OBJECT PRIVILEGES : 다른 사용자에게 부여된 권한 제거

```
REVOKE SELECT, INSERT
ON s_dept
FROM scott;
```

→ scott에게 select, insert 할 수 있는 권한 취소

- Privilege grouped by ROLE

→ 역할은 사용자가 소유하지 않고, 스키마에도 존재하지 않음

→ role을 부여하면 간단히 한번에 권한 가짐

- CREATE A ROLE

```
CREATE ROLE acct_rec; #역할 이름을 acct_rec으로 생성

CREATE ROLE acct_pay
IDENTIFIED BY bicentennial; #패스워드 bicentennial, 역할 이름 acct_pay

GRANT CREATE TABLE, CREATE SYNONYM
TO manager; #동의어와 테이블 생성 권한이 있는 manager 역할 생성

CREATE USER kevin
IDENTIFIED BY tiberio; #신규 사용자 생성

GRANT manager to kevin; #신규 사용자 kevin에게 manager 권한 부여
```

DML(데이터 조작용어) - SELECT, INSERT, UPDATE, DELETE

- SELECT

```
##Display all data in a table
select * from s_dept; #s_dept테이블에 모든 칼럼 출력

select dname from s_dept; #s_dept(부서)테이블의 부서이름 조회

desc s_emp; #s_emp테이블의 칼럼과 구조 조회
select ename, sal from s_emp; #s_emp테이블에서 사원 이름, 급여 출력

##Display unique columns of data
select distinct job "Title" #칼럼명 지정 큰따옴표(대소문자 구분, 공백, 특수문자)
from s_emp; #s_emp테이블에서 직무의 칼럼명을 Title로 지정하여 중복되지 않게 출력

##Display specific columns names
select ename EMPLOYEES
from s_emp; #ename의 칼럼명 EMPLOYEE로 지정

##Display specific rows of data - where절을 사용해 조건에 맞는 행을 조회
select ename
from s_emp
where sal > 2000; #s_emp테이블에서 급여가 2000초과인 사원의 이름 출력
```

```

select deptno, ename, job
from s_emp
where deptno=20; #s_emp테이블에서 부서번호가 20번인 사원의 부서번호, 이름, 직무 출력

select empno, ename, job, sal
from s_emp
where ename='BLAKE'; #Blake의 직원번호, 이름, 직무, 급여 조회 -> BLAKE로 찾아야됨

select ename, deptno, hiredate
from s_emp
where hiredate='82/12/22'; # 입사 날짜가 82/12/22일인 사원의 이름, 부서번호, 입사날짜 조회

select ename, job, sal
from s_emp
where job <> 'MANAGER'; #직무가 MANAGER가 아닌 사원의 이름, 직무, 급여 조회

select ename, hiredate
from s_emp
where hiredate>'82/11/01'; #입사 날짜가 82/11/01보다 큰 사원의 이름, 입사 날짜 조회

select ename, sal
from s_emp
where sal >= 2000; #급여 2000이상인 사원의 이름과 급여 조회

select deptno, ename
from s_emp
where deptno>=30; #부서번호가 30이상인 사원의 부서번호와 이름 조회

select ename, job
from s_emp
where ename >= 'SCOTT'; #이름의 알파벳순이 SCOTT보다 뒤에 있는 사원의 이름과 직무 조회

select ename, deptno, sal, hiredate
from s_emp
where hiredate<'81/05/01'; #입사 날짜가 81/05/01보다 작은 사원의 이름, 부서 번호, 월급, 입사날짜 조회

select ename, deptno, job
from s_emp
where deptno<=10; #부서 번호가 10이하인 사원의 이름, 부서번호, 직무 조회

#입사날짜가 80/12/01과 81/03/01사이에 있는 사원의 이름과 입사날짜 조회(이상, 이하)
select ename, hiredate
from s_emp
where hiredate between '80/12/01' and '81/03/01';

select ename, sal
from s_emp
where sal between 1500 and 2000; #급여가 1500와 2000사이인 사원의 이름과 급여 조회

select deptno
from s_dept
where deptno not between 10 and 20; #부서 번호가 10과 20사이에 있지 않은 부서번호 조회(10,20 포함x)

select ename, deptno
from s_emp
where ename not between 'ALLEN' and 'SMITH'; #직원 이름이 ALLEN과 SMITH 사이에 있지 않은 사원의 이름과 부

select ename, job, deptno

```

```

from s_emp
where job in ('MANAGER','SALESMAN'); #직무가 MANAGER, SALESMAN인 사원 이름, 부서 번호 조회

select empno, ename, deptno
from s_emp
where deptno not in (10,20); #부서 번호가 10,20이 아닌 사원번호,이름,부서 번호 조회

select ename
from s_emp
where ename like 'M%'; #사원 이름이 'M'으로 시작하는 사원 이름 조회

select ename, hiredate
from s_emp
where hiredate like '%01'; #입사 날짜가 '01'로 끝나는 사원 이름, 입사 날짜 조회

select ename
from s_emp
where ename like '%A%';. # 사원 이름에 'A'를 포함하는 사원 조회

select ename
from s_emp
where ename like 'B____F'; # 사원 이름이 'B'로 시작해서 'F'로 끝나는 다섯 글자의 사원 이름 조회

select ename, job, sal
from s_emp
where comm is null;. # 커미션 null값인 사원의 이름, 직무, 월급 조회

select ename, job, comm
from s_emp
where comm is not null;. # 커미션이 null값이 아닌 사원의 이름, 직무, 커미션 조회

##Display rows with complex conditions
select ename, sal, deptno
from s_emp
where deptno=20 and sal > 2000; # 부서 번호가 20이고, 급여가 2000보다 큰 사원의 이름, 부서 번호 조회

select ename, sal, deptno, job
from s_emp
where deptno=10 and job='MANAGER'; #부서 번호가 10이고,직무가 MANAGER인 사원의 이름, 월급, 부서번호, 직무조회

select ename, sal, deptno, job
from s_emp
where deptno=20 or job='SALESMAN';. # 부서 번호가 20이거나, 직무가 SALESMAN인 사원의 이름, 급여, 부서 번호 조회

select ename, sal, deptno
from s_emp
where sal >= 1000 and (deptno=10 or deptno=20); # 급여가 1000이상이고, 직무 번호가 10또는 20인 사원의 이름
#우선 순위 괄호로 지정

##Order the rows displayed
select ename, sal
from s_emp
where deptno=30
order by sal; # 부서번호가 30인 직원들의 급여를 오름차순 정렬

select ename, sal
from s_emp
where deptno=30

```

```
order by sal desc;. #부서번호가 30인 직원들의 급여를 내림차순 정렬

select ename, deptno, sal
from s_emp
where sal >= 1500
order by deptno, sal desc;. #급여가 1500인 직원의 이름, 부서 번호, 급여를 조회/부서 번호 오름차순, 급여로 내림차순
```

• INSERT

```
insert into s_dept
values(50, 'HR', 'SEOUL'); #s_dept(deptno, dname, loc) : column_name
# 'HR' 정보 입력

insert into s_emp(empno, ename, job, mgr, hiredate, sal, comm, deptno)
values(1234, 'ALEX', 'DESIGNER', 7839, SYSDATE, 3000, NULL, 20);

select * from s_emp where ename='ALEX'; #사원이름이 ALEX인 정보 조회

#새로운 직원 henry에 대한 정보를 입력
insert into emp
values (7633, 'CHRIS', '', '', '', 2500, '', 50); #null을 명시적으로 삽입

insert into s_emp(empno, ename, sal, deptno)
values(7634, 'HENRY', 1300, 50); #총 8칼럼중 4칼럼만 값 지정 -> 나머지는 null(null을 암시적으로 삽입)

select *
from s_emp
where empno in (7633, 7634); #위에 만든 두 칼럼 출력

insert into s_emp(empno, ename, hiredate, sal, deptno)
values(7636, user, sysdate, 2000, 10) #db사용자의 이름으로 정보 입력
```

• UPDATE

```
update s_emp
set deptno=20 #변경 내용
where empno=7636;

update s_emp
set deptno=20, sal=4000
where ename='ALEX'; # alex의 부서번호와 월급 변경

update s_emp
set comm=1000
where deptno=10; #회사에서 부서 번호 10인 직원들에게 커미션을 1000씩 지급

select ename, comm, deptno
from s_emp
where deptno=10;
```

• Delete - 행 삭제

```
delete from s_emp
where ename='ALEX'; # 삭제

select ename from s_emp where ename='ALEX' # 출력 안되는 것 확인
```

```
delete from s_emp
where deptno=50;

select ename from s_emp where deptno=50;
```

→ 조건이 없이 테이블 이름만 입력하는 경우, 테이블 내 모든 데이터가 삭제된다. 전체 테이블을 삭제하는 것이 아니면 where절 생략하면 안 됨.

- Control Transactions

- transaction : insert, update, delete
- commit : 보류중인 모든 변경 내용을 영구적으로 만든다
 - 영향받은 행은 lock이 해제되고 다른 사용자가 행을 변경
- rollback : 데이터변경이 취소되고, 데이터 이전 상태가 복원
 - 영향을 받은 행은 lock이 해제되고, 다른 사용자가 행을 변경 가능
- savepoint : 현재 트랜잭션에 저장 지점 만든다
 - 부분 롤백을 수행하기 위해 저장 지점 미리 만듦
 - 동일한 이름의 저장 지점 설정하면 이전의 저장 지점은 삭제

```
insert into s_emp(empno, ename, hiredate, sal)
values(3790, 'GOODMAN', SYSDATE, 2000);

savepoint a; #savepoint a 저장

insert into s_emp(empno, ename, hiredate, sal)
values(3791, 'BADMAN', SYSDATE, 1000);

savepoint b; #savepoint b 저장

insert into s_emp(empno, ename, hiredate, sal)
values(3792, 'YESMAN', SYSDATE, 3000);

select empno, ename from s_emp #위에 3개의 insert문을 통해 출력됨

rollback to savepoint a; #savepoint a로 rollback복원
commit; #영구 저장

select empno, ename
from s_emp; #goodman만 삽입된 table 출력됨 -> savepoint를 이용해 영구적으로 변경
```

- commit이나 rollback 암묵적으로 실행하는 상황 주의 **
 - 자동 commit : create table과 같은 DDL 명령어 실행
 - 자동 commit : 명시적 commit 또는 rollback을 하지 않고 db 정상 종료
 - 자동 rollback : 비정상적인 종료 또는 시스템 오류

```
##perform computations with numbers
select ename, sal, sal+500 "NEW SALARY"
from s_emp
where job='MANAGER'
order by sal+500; #직무가 MANAGER인 사원의 $500 급여 인상 계산 후 사원이름, 급여, 인상급여(NEW SALARY) 출력

select ename, sal, sal*0.1 "BONUS"
from s_emp
where deptno=20
```

```

order by sal*0.1; #부서 번호가 20인 사원에 대해 급여의 10% 보너스를 지급 후, 사원이름, 급여, 보너스(BONUS)출

#rules of precedence
select ename, sal, (sal+100)*12 "ANNUAL COMPENSATION"
from s_emp
where deptno=10; #연간보상 급여에 $100상여 후 12곱함

select ename, sal, round(sal/22,2) "PERFORMACE PAY"
from s_emp
where job='CLERK'; #성과금 급여 나누기 근무일수(22일) -소수점 셋째자리에서 반올림

select ename, sal, trunc(sal/30,2) "PERFORMACE PAY"
from s_emp
where job='CLERK'; #성과금 급여 나누기 근무일수(22일) -소수점 둘째자리 밑으로 버림

select *
from s_emp
where mod(empno,2)=0; #사원 번호가 짝수인 직원의 정보 모두 조회

select ename, sal, comm, sal+comm
from s_emp
where deptno=30; #추가 금액(월급에 커미션 더한 값)

select ename, sal, comm, nvl(sal+comm,)
from s_emp
where deptno=30; #nvl(a,b) a의 값이 null이 아니면 a, null이면 빈칸 대신 b출력

select ename, hiredate, hiredate+90
from s_emp
where sal >= 2000; # 날짜형도 산술 가능

select ename, SYSDATE-hiredate DAYS, (SYSDATE-hiredate)/30 MONTHS
from s_emp
where deptno=20; #부서번호가 20인 사원의 근무기간 일(DAYS), 월(MONTHS)-SYSDATE(현재 시스템 날짜)

select ename, add_months(hiredate,6)
from s_emp
where job='MANAGER'; #직무가 manager인 사원의 입사날짜에 6개월을 더한 값-add_months

select last_day(sysdate)
from dual; #현재 달의 마지막 일자를 출력

#next_day(date, str) -> date와 가장 가까운 다음 요일 str 날짜 반환하는 함수

select ename, hiredate, months_between(sysdate, hiredate)
from s_emp
where job='SALESMAN'; #직무가 salesman인 사원의 이름, 입사날짜, 근무개월 출력 -months_between()

##reformat dates
select ename, to_char(hiredate, 'MM/YY')
from s_emp
where job='MANAGER'; #'YY'년 'MM'월 'DD'일

insert into s_emp(empno, ename, hiredate, deptno)
values(8371, 'MARU', to_date('19900708', 'YYYYMMDD'),40); #신입사원 날짜형(YYYY/MM/DD)으로 출력

##manipulate character strings
select empno||' '|| ename 'ID AND EMPLOYEE'

```



```

from s_emp
where deptno=20; #사원번호와 사원이름 열 결합

select initcap(ename), lower(job)
from s_emp
where job='CLERK' #사원이름의 첫글자는 대문자-initcap, 직무는 모두 소문자-lower

select upper(initcap(ename)), job
from s_emp
where job='CLERK'; #첫글자만 대문자인 사원이름을 모두 대문자로 출력

select ename, substr(ename, -1, 2)
from s_emp
where deptno=10; #사원이름의 뒤에 두 글자만 출력- substr(char,m,n) -> char을 m번째 n글자까지 출력(n안하면 !

select count(*) from s_emp; #s_emp테이블에 있는 모든 사원의 수 출력

##group rows together
select job, count(*) "Number"
from s_emp
group by job; #직무별로 그룹화하여 사원의 수 출력

select deptno, count(*) "Head Count"
from s_Emp
group by deptno; #부서 번호에 따른 사원 수 출력

####group by없이 정규 열과 그룹 함수(count) 쓰면 안됨####

select deptno, job, count(*)
from s_emp
where deptno=30
group by deptno, job #부서번호가 30인 사원의 직무별 사원수

select deptno, avg(sal) SAL, count(*)
from s_emp
group by deptno
having count(*)>=3 #직원이 세명이상인 부서의 급여 평균과 직원수 출력

select job, sum(sal)
from s_emp
group by job
having sum(sal)>5000
order by sum(sal) desc; #직무별 급여의 합이 5000보다 큰 직무와 급여 합 출력(급여 합은 내림차순)

##display data from related tables
select e.last_name, e.dept_id, d.dept_name
from e_emp e, e_dept d
where e.dept_id=d.dept_id; #사원들의 성, 부서 id, 부서 이름 출력

##create a view of multiple tables
select문 위에 create view empdeptvu as 쓰면 empdeptvu view 생성

##nest queries the return one row
select last_name, title
from e_emp
where title in
      (select title
       from e_emp

```

```

        where last_name='SMITH'); #smith와 동일한 직무를 가진 사원의 성 출력

select last_name, title, dept_id
from e_emp
where dept_id in
        (select dept_id
        from e_emp
        where upper(last_name)='BIRI'); #biri와 동일한 부서에 있는 사람의 성, 직무, 부서 출력

select id, last_name, salary, dept_id
from e_emp
where dept_id in
        (select id
        from e_dept
        where region_id=2); #지역2에 있는 부서에서 일하는 직원

select last_name, id, dept_id
from e_emp
where dept_id in (select id
                                from e_dept
                                where name='Sales')

or dept_id in (select id
                                from e_dept
                                where region_id=2); #영업부 또는 지역2에 있는 모든 직원의 성, id,

```

PL/SQL

- 생성과 실행
 - 반복 처리(Loop)
 - 비교 처리(If)
 - error 처리(예외 처리)
 - 변수 선언
 - 네트워크 트래픽 감소

Stored procedure

```

##1.os편집기로 파일 생성
vi salary_cal.sql
#procedure문 생성
create procedure salary_cal
begin
    select문~~
end;

##2.데이터베이스 접속
tbsql sys/tibero
@salary_cal.sql

##3.execute 명령어를 사용해 procedure 호출
execute salary_cal

```

- EXAMPLE

```
tbsql sys/tibero
```

```

##빈 테이블에 insert하는 procedure 생성##

#로그 정보를 저장하기 위한 테이블 생성
create table log_table
(userid varchar2(10),
log_date date);

#procedure 파일 생성
vi log_execution.sql
create or replace procedure log_execution
is
begin
insert into log_table(userid, log_date)
values(user, sysdate);
end log_execution;

#procedure 생성
@log_execution.sql
show errors #에러 출력
execute log_execution

#로그 정보가 기록된 테이블 조회
select * from log_table;

##사원 삭제하기##

#테이블 생성
vi fire_emp.sql
create or replace procedure fire_emp
(v_emp_no in s_emp.empno%type) #s_emp table에 empno의 type을 쓰겠다 선언
is
begin
delete from s_emp      #삭제
where empno=v_emp_no;  #테이블 사원 procedure와 연결
end fire_emp;

#procedure 생성
@fire_emp.sql #생성만 아직 호출(execute)안해서 안쓰임

#출력확인
select ename, empno
from s_emp
where empno=7654; #1row

#precedure empno=7654에 호출
execute fire_emp(7654);

select ename, empno
from s_emp
where empno=7654; #0row

##사원번호 입력하고 사원에 대한 정보 검색##

vi query_emp.sql
create or replace procedure query_emp
(v_emp_no in s_emp.empno%type,
v_emp_name out s_emp.ename%type, #외부 참조(입력 받을)
v_emp_sal out s_emp.sal%type,

```

```

v_emp_comm out s_emp.comm%type)
is
begin
    select ename, sal, comm
    into v_emp_name, v_emp_sal, v_emp_comm
    from s_emp
    where empno=v_emp_no;
end query_emp;
/

@query_emp.sql

variable emp_name varchar2(15) #세션이 살아있는 동안 사용할 수 있는 변수를 선언할 때 사용
variable emp_sal number
variable emp_comm number

execute query_emp(7900, :emp_name, :emp_sal, :emp_comm)
print emp_name

##비교문##
#1) if~then~end if
set serveroutput on #써줘야 출력됨
declare
v_condition number :=1; #declare문은 선언할때 :=
begin #조건문
if v_condition=1 then
dbms_output.put_line('데이터값은 1입니다!!'); #print와 같은
end if #if문 end
end;
/

#2)if~then~else~end if
set serveroutput on
declare
v_condition number:=2;
begin
if v_condition=1 then
dbms_output.put_line('데이터값은 1입니다!!');
else
dbms_output.put_line('데이터값은 1이 아닙니다!!');
end if;
end;
/

#3)if~then~elsif~else~end if
set serveroutput on
declare
if v_condition number :=2;
begin
if v_condition>1 then
bms_output.put_line('데이터값은 1보다 큼니다!!');
elsif v_condition=1 then
bms_output.put_line('데이터값은 1입니다!!');
else
bms_output.put_line('데이터값은 1보다 작습니다!!');
end if;
end;
/

```

```

##반복문##
#1)Loop문
set serveroutput on
declare
    cnt  number:=0; #cnt 숫자형 변수 선언
begin
    loop
        cnt:=cnt+1;
        dbms_output.put_line(cnt);
        exit when cnt=10; #(반복 중단) -> 1~10까지 반복
    end loop; #loop end
end;
/

#2)for_loop문 (반복되는 횟수 정확히 아는 경우, 최대최소내에서 반복)
set serveroutput on
declare
    i  number;
begin
    for i in 1..10 loop #1~10까지 반복
        if (mod(i,2)=1) then #홀수일때
            dbms_output.put_line(i); #i출력
        end if;
    end loop;
end;
/

#3) while~end loop문(while조건문이 만족할때까지 반복)
set serveroutput on
declare
    v_cnt  number:=1;
    v_str  varchar2(10):=null; #변수 초기값 선언
begin
    while v_cnt<=10 loop #10이하까지 반복
        v_str:=v_str||'#'; #'#'문자형 지정
        dbms_output.put_line(v_str);
        v_cnt:=v_cnt+1; #'#'문자 1~10까지 출력
    end loop;
end;
/

```

Stored function

- stored procedure와 동일한 개념, 기능
- stored procedure(로직 처리하고 끝냄) ↔ stored function(처리 결과를 사용자에게 돌려줌-return)

```

##1.os편집기로 파일 생성
vi test_check.sql
#function 생성
create function test_check
(v_test  number;)
return number; #사용자에게 돌려줌

##2.데이터베이스 접속
tbsql sys/tibero
@test_check.sql

```

```
##3.execute 명령어를 사용해 procedure 호출
variable v_test_number
execute :test_check:=test_check(7900)
```

- Example

```
##입력값에 세금 매겨주는 함수 생성##
vi tax.sql
create or replace function tax
(v_value in number)
return number;
is
begin
return(v_value*0.07);
end tax;

tbsql sys/tibero
@tax.sql

variable x.number;
execute :x :=tax(100);  #100*0.07을 출력

##DML(update, insert, select문)에서도 실행 가능##
select sal, tax(sal)
from s_emp
where empno=7900; #위에 선언한 tax function 사용
```

Cursor and Exception

Cursor

- 행 단위로 처리를 해야할 때 사용하는 방식
- implicit cursor(암시적 커서)
 - 한 번 실행에 하나의 결과를 리턴하는 sql문
 - 사용되는 스칼라 변수는 한번에 하나의 값만 저장

```
select empno, ename
into :v_no, :v_name
from s_Emp
where deptno=10;
```

- explicit cursor(명시적 커서) : sql문 실행시켰을 때 결과 여러 개 → 암시적 커서는 에러발생
 - 1) cursor(실행하고자 하는 select문 작성)
 - 2) open(cursor선언에서 선언문 select문 실행 의미)
 - 3) fetch(open된 select문에 의해 검색된 하나의 행 정보 읽어옴/결과 여러개면 반복문 실행)
 - 4) close(선언된 select 해제)

Exception

- 정상적으로 실행되지 못할 때 에러 발생
- exception에 의해 처리가능
- 자주 발생하지 않는 에러는 사용자가 직접 정의 가능

- 시스템 정의 예외

ex) no_data_found(select into 결과에서 로우가 하나도 없는 경우)

too_many_rows(select into 결과에서 로우가 둘 이상인 경우)

##1)미리 정의된 에러 처리 방법

#procedure 생성

create or replace procedure test

(v_sal in s_emp.sal%type) #인수 선언

is

v_ename s_emp.ename%type; #변수 선언

begin

select ename

into v_ename

from s_emp

where sal=v.sal;

dbms_output.put_line('해당 사원' || v_ename || '입니다.');

exception

when no_data_found then

raise_application_error(-20002, 'DATA NOT FOUND');

when too_many_rows then

raise_application_error(-20003, 'TOO MANY ROWS');

when no_data_found then

raise_application_error(-20004, 'OTHERS ERROR'); #시스템 정의 예외 선언

end;

/

##2)미리 정의되지 않은 에러 처리 방법

vi test2.sql

create or replace procedure hire_emp

(v_emp_name in s_emp.ename%type,

v_emp_job in s_emp.job%type,

v_mgr_no in s_emp.mgr%type,

v_emp_hiredate in s_emp.hiredate%type,

v_emp_sal in s_emp.sal%type,

v_emp_comm in s_emp.comm%type,

v_dept_no in s_emp.deptno%type)

is

e_invalid_mgr exception; #정의되지 않은 예외 선언

pragma exception_init(e_invalud_mgr, -10008); #특정 에러코드와 예외 메시지를 연결해주는 역할

begin

insert into s_emp(empno,ename,job,mgr,hiredate,sal,comm,deptno)

values(emp_id.nextval,v_emp_name,v_emp_job,v_mgr_no,v_emp_hiredate,

v_emp_sal,v_emp_comm,v_dept_no);

commit work;

exception

when e_invalid_mgr then

raise_application_error(-20201, 'deptno is not a valid department.');

end hire_emp;

/

@test2.sql

execute hire_emp('STONE', 'CLERK', 9000, SYSDATE, 950, 300, 44);

##3)사용자가 정의하는 에러 처리 방법

#s_emp테이블로부터 어떤 조건을 만족하는 행의 count를 계산하여 0값이면 'There is no employee salary'
#메세지를 출력하고, 0값이 아니면 'There are rows employee' 메시지 출력

```
create or replace procedure test3
(v_sal in s_emp.sal%type)
is
    v_low_sal s_emp.sal%type:=v_sal-100;
    v_high_sal s_emp.sal%type:=v_sal+100;
    v_no_emp number(7);

    e_no_emp_returned exception;
    e_more_than_one_emp exception;

begin
    select count(ename)
    into v_no_emp #변수 선언
    from s_emp
    where sal between(v_sal-100)and(v_sal+100);
    if v_no_emp=0 then
        raise e_no_emp_returned; #0이면
    elsif v_no_emp>0 then
        raise e_more_than_one_emp; #0보다 크면
    end if;

    exception
        when e_no_emp_returned then
            dbms_output.put_line('There is no employee salary');
        when e_more_than_one_emp then
            dbms_output.put_line('There are rows employee');
        when others then
            dbms_output.put_line('Some other error occurred');

end;
/

@test3.sql

execute test3(9000) #no
execute test3(3000) #rows
```

##4)예외 트래핑 함수: 개발자가 직접 참조하여 처리/sqlcode함수 참조하면 sql문 실행 결과
##sqlcode값이 0이면 에러없이 실행/1이면 사용자가 정의한 에러/100이면 no_data_found
##sqlcode(해당에러코드), sqlerrm(해당 에러 메시지)

```
exception when 밑에
v_err_code:=SQLCODE; #에러코드
v_err_message:=SQLERRM; #에러메시지 출력
```

PL/SQL

```
ACCEPT 변수 이름 PROMPT ' 문자열
' #사용자가 직접입력
EX )
ACCEPT P_NUM PROMPT 'number'
BEGIN
IF MOD(&P_NUM, 2) = 0 THEN
```



```

DBMS_OUTPUT.PUT_LINE('짝수');
ELSE
DBMS_OUTPUT.PUT_LINE('홀수');
END IF;
END; /

```

```

accept p_deptno prompt '부서 번호 입력
' #사용자가 직접 입력
declare
v_ename s_emp.ename%type;
v_sal s_emp.sal%type;
v_deptno s_emp.deptno%type;
cursor emp_cursor is #1)cursor 선언
select ename, sal, deptno
from s_emp
where deptno = &p_deptno; #부서번호 테이블에서 가져옴
begin
open emp_cursor; #2)cursor열기
loop #반복문
fetch emp_cursor into v_ename, v_sal, v_deptno; #3)cursor할 변수지정
exit when emp_cursor%notfound; #loop 끝낼 조건 선언
dbms_output.put_line(v_ename|| ' ' ||v_sal|| ' ' ||v_deptno); #출력
end loop; #loop끝냄
close emp_cursor; #4)cursor 닫기
end; /

```

1. (인수없는)프로시저 생성방법

```

CREATE OR REPLACE PROCEDURE 프로시저이름
IS

[

변수이름 데이터타입;  -- 프로시저내에서 사용할 변수선언

변수이름 데이터타입;

변수이름 데이터타입;

..

]

BEGIN

기능 구현;

END;

```

2. (인수있는) 프로시저 생성방법

```

CREATE OR REPLACE PROCEDURE 프로시저이름(
변수이름 IN 데이터타입, 변수이름 IN 데이터타입, .... --IN 생략가능

)

IS

```

```
[
변수이름 데이터타입;  -- 프로시저내에서 사용할 변수선언

..

]

BEGIN

기능 구현;

END;
```

실습_1 : select문

사원 테이블 - S_EMP

부서 테이블 - S_DEPT

```
#1)사원 테이블의 모든 데이터를 출력하라
select *
from s_emp;

#2)사원 테이블에서 각 사원의 직무, 사원번호, 이름, 입사일을 출력하라.
select job, empno, ename, hiredate
from s_emp;

#3) 사원 테이블에서 직무를 출력하되, 각 항목이 중복되지 않게 출력하라.
select distinct job
from s_emp;

#4) 급여가 2,700이상인 사원의 이름 및 급여를 출력하라.
select ename, sal
from s_emp
where sal>=2700;

#5) 급여가 2,000 이상 - 4,000이하의 범위에 속하지 않는 모든 사원의 이름과 급여를 출력되,
#급여를 기준으로 내림차순으로 정렬하라.
select ename, sal
from s_emp
where sal not between 2000 and 4000
order by sal desc;

#6) 1981년 2월 20일 ~ 1981년 5월 1일에 입사한 사원의 이름, 직무, 입사일을 출력
#하되, 입사일을 기준으로 오름차순 정렬하라.
select ename, job, hiredate
from s_emp
where hiredate between '0081/02/20' and '0081/05/01'
order by hiredate;

#7) 10번 및 30번 부서에 속하는 모든 사원의 이름과 부서 번호를 출력하되, 알파벳
#순으로 정렬하라.
select ename, deptno
from s_emp
where deptno in (10,30)
order by ename;
```

#8) 10번 및 30번 부서에 속하는 모든 사원 중 급여가 1,500을 넘는 사원의 이름과
#급여를 출력하라. 단, 컬럼명을 각각 "Employee" 및 "Monthly Salary"로 지정하라.
select ename "Employee", sal "Monthly Salary"
from s_emp
where deptno in (10,30) and sal>1500;

#9) 관리자가 없는 모든 사원의 이름 및 직무를 출력하라.
select ename, job
from s_emp
where mgr is null;

#10) 커미션을 받는 모든 사원의 이름, 급여 및 커미션을 출력하되, 급여를 내림차순하
#여 정렬하라.
select ename, sal, comm
from s_emp
where comm is not null
order by sal desc;

#11) 이름의 세 번째 문자가 A인 모든 사원의 이름을 출력하라.
select ename
from s_emp
where ename like '__A%';

#12) 이름에 L이 두 번 들어가며 부서 30에 속해있는 사원의 이름을 출력하라.
select ename
from s_emp
where ename like '%L%L%' and deptno=30;

#13) 직업이 Clerk 또는 Analyst이면서 급여가 1000, 3000, 5000 이 아닌 모든 사원
#의 이름, 직업 및 급여를 출력하라.
select ename, job, sal
from s_emp
where job in ('CLERK','ANALYST') and sal not in (1000,3000,5000);

#14) 사원번호, 이름, 급여 그리고 15%인상된 급여를 정수로 표시하되 컬럼명을 "New
#Salary"로 지정하여 출력하라.
select empno, ename, sal, sal*1.15 "New Salary"
from s_emp;

#15) 14번 문제와 동일한 데이터에서 급여 인상분(새 급여에서 이전 급여를 뺀 값)을
#추가해서 출력하라. 단, 컬럼명은 Increase로 하라.
select empno, ename, sal, sal*1.15 "New Salary", sal*0.15 "Increase"
from s_emp;

#16) 사원의 이름과 커미션을 출력하되, 커미션이 책정되지 않은 사원의 커미션은 'no
#commission'으로 출력하라.
select ename, comm, nvl(to_char(comm), 'no commission')
from s_emp;

#17) 직무와 각 직무별 최고 급여와 최저 급여를 출력하라.
select job, max(sal), min(sal)
from s_emp
group by job;

#18) 직무와 인원수, 평균 급여를 출력하라. 단, 직무별 인원수가 3 이상인 경우에만 출력하라.
select job, count(*), avg(sal)
from s_emp

```
group by job
having count(*)>=3;
```

#19) 각 부서별 부서번호, 인원 수, 급여 합계를 출력하라.

```
select deptno, count(*), sum(sal)
from s_emp
group by deptno;
```

#20) 부서번호, 부서 별 인원 수, 급여 합계를 보이되, 급여 합계가 7,000 ~ 11,000만 출력하라.

#단 입사날짜가 1981-01-01 이전 사원은 제외하고, 급여 합계가 많은 순으로 출력하라.

```
select deptno, count(*), sum(sal)
from s_emp
where hiredate >'0081/01/01'
group by deptno
having sum(sal) between (7000,11000)
order by sum(sal) desc;
```

실습_2 : join문

#1) 부서별로 부서 이름, 부서 위치, 인원 수 및 평균 급여를 출력하라. 그리고 각각의 컬럼명

#을 department, location, employee, average로 표시해라.

```
select d.dname "department", d.loc "location", sum(e.empno) "employee", avg(e.sal) "average"
from s_emp e, s_dept d
where e.deptno=d.deptno
group by e.deptno, d.dname, d.loc;
```

#2) Smith와 동일한 부서에 속한 모든 사원의 이름 및 입사일을 출력해라. 단, Smith는 제외하고 출력해라.

```
select ename, hiredate
from s_emp
where deptno in (select deptno
                  from s_emp
                  where ename='SMITH') and ename <> 'SMITH';
```

#3) 이름에 T가 들어가는 사원이 속한 부서에서 근무하는 모든 사원의 직원번호 및 이름을 출력해라.

```
select empno, ename
from s_emp
where deptno in (select deptno
                  from s_emp
                  where ename like '%T%');
```

#4) 부서 위치가 Dallas인 모든 사원의 이름, 부서번호 및 직위를 출력하라.

```
select e.ename, e.deptno, e.job
from s_emp e, s_dept d
where e.deptno=d.deptno and d.loc='DALLAS';
```

#5) KING에게 보고하는 모든 사원의 이름과 급여를 출력하라.

```
select ename, sal
from s_emp
where mgr in (select ename
               from s_emp
               where ename='KING');
```

#6) Sales 부서의 모든 사원에 대한 부서번호, 이름 및 직위를 출력하라.

```
select e.deptno, e.ename, e.job
from s_emp e, s_dept d
where e.deptno=d.deptno and d.dname='SALES';
```

#7) 커미션을 받는 사원과 급여가 일치하는 사원의 이름, 부서 번호, 급여를 출력해라.

```
select ename, deptno, sal
from s_emp
where sal in (select sal
              from s_emp
              where comm is not null);
```

#8) 이름에 A가 들어가는 사원과 같은 직업을 가진 사원의 이름과 월급, 부서번호를 출력하라

```
select ename, sal, deptno
from s_emp
where job in (select job
              from s_emp
              where ename like '%A%');
```

실습_3 : declare문(pl/sql)

#1) 밑변과 높이 빗변을 각각 직접 입력하고, 직각 삼각형이 맞는지 출력되게하는 PL/SQL 작성

#POWER(NUM1,NUM2)-NUM1을 NUM2제공한 결과를 반환

```
accept a prompt '밑변을 입력하시오:'
accept b prompt '높이를 입력하시오:'
accept c prompt '빗변을 입력하시오:'
```

```
begin
if power(&a,2)+power(&b,2)=power(&c,2) then dbms_output.put_line('직각 삼각형이 맞습니다. ');
else dbms_output.put_line('직각 삼각형이 아닙니다. ');
end if;
end;
/
```

#2) 두 수를 물어보게 입력 받아 두 수의 합이 결과로 출력되는 PL/SQL 작성 DECLARE로 변수 선언

```
accept a prompt '첫번째 수 입력:'
accept b prompt '두번째 수 입력:'
```

```
declare
    c number;
begin
    &a+&b:=c;
end;
```

#3) 사원 번호를 물어보고, 해당 값을 입력했을 때 해당 사원의 급여를 출력하는 PL/SQL을 작성

```
declare
v_sal number;

accept p_empno prompt '사원 번호:'

begin
select sal into v_sal
from s_emp
where empno=&p_empno;
dbms_output.put_line('해당 사원은' ||sal|| '입니다. ');
end;
```

#4) s_emp 테이블에서 이름을 입력 받아 해당 사원의 월급이

#3000이상이면 고소득자,

#2000이상이고 3000보다 작으면 중간 소득자,

#2000보다 작은 사원은 저소득자라는 메시지를 출력하는 PL/SQL 코드를 작성하시오

```
accept p_ename prompt '사원 이름 입력:'
```

```

declare
    v_ename s_emp.ename%type:=upper('&p_ename');
    v_sal s_emp.sal%type;
begin
    select sel into v_sal
    from s_emp
    where ename=v_ename;

    if v_sal>=3000 then dbms_output.put_line('고소득자');
    elsif v_sal>=2000 then dbms_output.put_line('중소득자');
    else dbms_output.put_line('저소득자');
end if;
end;
/

```

##답##

```

declare
    i number(10):=0;
    j number(10):=0;
begin
    for i in 2..9 loop
        for j in 1..9 loop
            dbms_output.put_line(i||'x' ||j||'=' ||i*j);
        end loop;
    end loop;
end;
##

```

#5) LOOP 문으로 구구단 2단을 출력하시오
set serveroutput on

```

declare
cnt number:=0;

begin
loop
cnt:=cnt+1;
dbms_output.put_line(2*cnt);
exit when cnt=9;

end loop;
end;
/

```

##답##

```

declare
    v_count number(10):=0;
begin
    loop
        v_count := v_count+1;
        dbms_output.put_line('2X' ||v_count||'=' ||2*v_count);
        exit when v_count=9;
    end loop;
end;
##

```

#6) WHILE LOOP 문으로 구구단 3단을 출력하시오

```
set serveroutput on
```

```
declare
```

```
cnt number:=0;
```

```
begin
```

```
while cnt<9 loop
```

```
cnt:=cnt+1;
```

```
dbms_output.put_line(3*cnt);
```

```
end loop;
```

```
end;
```

```
/
```

```
##답##
```

```
declare
```

```
    v_count number(10):=0;
```

```
begin
```

```
    while v_count<9 loop
```

```
        v_count := v_count+1;
```

```
        dbms_output.put_line('3X' || v_count || '=' || 3*v_count);
```

```
    end loop;
```

```
end;
```

```
/
```

```
##
```

#7) FOR LOOP 문으로declare 구구단 4단을 출력하시오

```
set serveroutput on
```

```
declare
```

```
I number;
```

```
begin
```

```
for I in 1..9 loop
```

```
dbms_output.put_line(4*I);
```

```
end loop;
```

```
end;
```

```
/
```

```
##답##
```

```
declare
```

```
    v_count number(10):=0;
```

```
begin
```

```
    for v_count in 1..9 loop
```

```
        dbms_output.put_line('4X' || v_count || '=' || 4*v_count);
```

```
    end loop;
```

```
end;
```

```
/
```

```
##
```

#8) FOR LOOP문을 '2번' 이용해 구구단2단부터 9 단까지 출력하시오

```
##답##
```

```
declare
```

```
    i number(10):=0;
```


```
    j number(10):=0;
```

```
begin
```

```
    for i in 2..9 loop
```

```
        for j in 1..9 loop
```

```
        dbms_output.put_line(i||'x'||j||'='||i*j);
    end loop;
end loop;
end;
##
```

 SQL_Exam