-- 테이블 생성 명령어

CREATE TABLE EMP (

EMPNO NUMBER(4, 0),

ENAME VARCHAR2(10),

HIREDATE DATE,

SAL NUMBER(7, 2));

insert into emp values (7369, ‘SMITH’, ‘1980-12-17, 800);

drop table emp;

-- 사원들의 직급을 중복되지 않도록 검색

select distinct job

from emp;

-- 10번이나 30번에 소속된 사원명 검색

select ename, deptno

from emp

where deptno in (10, 30); -- is not in

-- &: 사용자로부터 매개변수 값을 입력 받기

select ename

from emp

where deptno = &부서번호;

-- 관리자가 널인 사원명 검색

select ename

from emp

where mgr is null; -- is not null

-- 사원명, 입사일을 검색 (가장 오래전에 입사한 사원이 먼저 출력되도록)

select ename, hiredate

from emp

order by hiredate asc;

-- 1981년대 입사한 사원명 검색

select ename, hireadte

from emp

where hiredate like ‘81%’; -- 문자열 연산자(%, \_)

-- 문자의 길이, 문자의 길이의 바이트 값

select ename, length(‘김동현’), lengthb(‘김동현’)

from dual;

-- 문자열 결합

select concat(ename, sal), ename || sal

from emp;

-- 이름의 1번째 값부터 3개, hiredate 4번째 값부터 2개

select ename, substr(ename, 1, 3), substr(hiredate, 4, 2) year

from emp;

-- mod 나머지, ceil 가장가까운큰정수, floor 가장가까운작은정수, power 제곱

select mod(11, 2) , ceil(123.456), floor(123.456), power(2, 8)

from dual;

-- 오늘 날짜를 기준으로 27개월 뒤, 달의 마지막에 해당하는 날짜, 다음으로 오늘 날짜

select add\_months(sysdate, 27), last\_day(sysdate), next\_day(sysdate, ‘금’)

from dual;

-- 사원들의 이름, 년봉(봉급 \* 12)-천단위구분기호표시/$기호표시, 세금(년봉의 3.3%)

select ename 사원명, to\_char(sal\*12, ‘$999,999’) 년봉, to\_char(round(sal\*12\*0.033), ‘$999,999’) 세금

from emp;

-- 입사일이 1,2,3월인 사원들의 입사일을 기준으로 현재까지 사원들의 사원명, 입사일, 근무일수, 근무개월수, 근무년수 출력

select ename 사원명, hiredate 입사일, trunc(sysdate-hiredate, 0) 근무일수,

trunc(months\_between(sysdate, hiredate), 0) 근무개월수, trunc((sysdate-hiredate)/365, 0) 근무년수

from emp

where to\_char(hiredatee, ‘mm’) between ‘01’ and ‘03’;

-- 사원명의 마지막 글자를 ‘\*’ 문자로 치환, 입사일의 일(day)를 ‘\*’으로 변경해서 출력하시오.

select replace(ename, substr(ename, -1, 1), ‘\*’) 사원명, replace(hiredate, substr(hiredate, -2, 2), ‘\*’) 입사일

from emp;

-- 사원번호가 짝수인 사원들의 사원명을 10자리로 출력하되 왼쪽 빈자리는 ‘\*’기호로 채워 출력하고, 마지막 문자’S는 제거

select lpad(ename, 10, ‘\*’) 사원명1, rtrim(ename, ‘S’) 사원명2

from emp

where mod(empno, 2) = 0;

-- 사원명, 봉급, 커미션, 비고 검색, 비고열은 comm이 널인 경우 ‘NO’, 널이 아닌 경우 ‘YES’로 출력

select ename, sal, comm, nvl2(comm, ‘YES’, ‘NO’) 비고

from emp;

-- 사원명, 직급, 보너스 검색 / 보너스는 job이 CLERK이면 봉급의 5% SALESMAN이면 봉급의 10%, 그 외는 봉급의 3% 지급

select ename 사원명, job 직급, decode(job, ‘CLERK’, sal\*0.05, ‘SALESMAN’, sal\*0.1, sal\*0.03) 보너스

from emp;

case 문이면 case when job=’CLERK’ then sal\*0.05 … else sal\*0.03 end 보너스

-- 정규식

^ : 해당 문자로 시작하는 // $ : 해당 문자로 끄타는 // . : 한 문자 // \* : 모든 문자 // [ ] : 해당 문자에 해당하는 한 문자

[^ ] : 해당 문자에 해당하지 않는 한 문자 // [[:alpha:]] : [a-zA-Z] // [[:digit:]] : [0-9] // [[:space:]] : ‘ ‘

-- 이름이 A,E,I,O,U 중 하나로 시작하는 사원명 검색 (regexp\_lik() 사용)

select ename from emp

where regexp\_like(ename, ‘&[A|E|I|O|U]’);

-- 이름의 마지막 문자를 ‘\*’ 문자로 치환, 입사일의 ‘/’ 문자를 ‘-‘ 문자로 치환하여 출력

select regexp\_replace(ename, ‘[A-Z]$’, ‘\*’) 사원명, regexp\_replace(hiredate, ‘/’, ‘-‘) 입사일

from emp;

-- 사원명, 봉급, 이름문자열 내에 ‘S’문자 개수, 봉급값 중 0의 개수 출력

select ename 사원명, sal 봉급, regexp\_count(ename, ‘S’) “문자 개수”, regexp\_count(sal, ‘0’) “0의 개수”

form emp;

-- count, sum, max, min, avg, stddev 표준편차, variance 분산

-- 직급별 사원이 3명 이상인 직급에 한하여 직급별 최대급여, 인원수 검색

select job, max(sal) 최대급여, count(\*) 인원수

from emp

group by job

having count(job) >= 3;

-- 직급별 평균봉급, 부서번호별 평균봉급, 전체 평균봉급 검색

select job 직급, deptno 부서번호, round(avg(sal)) 평균봉급

from emp

group by grouping sets(job, deptno, ());

-- 직급별 부서별 평균봉급, 직급별 평균봉급, 부서번호별 평균봉급, 전체 평균봉급 검색

select job 직급, deptno 부서번호, round(avg(sal)) 평균봉급

from emp

group by cube(job, deptno)

order by job, deptno;

-- 직급별 사원명을 입사순서대로 가로로 출력

select job 직급, listagg(ename, ‘, ‘) within group (order by hiredate) 사원명

from emp

group by job;

-- 급여+커미션 값의 최대/최소/평균값 검색 / 커미셜이 널인 경우 0으로 계산, 출력 금액은 소수 첫째자리까지 출력

select round(max(sal + nvl(comm, 0)), 1) 최대, round(min(sal + nvl(comm, 0)), 1) 최소, round(sal + nvl(comm, 0)), 1) 평균

from emp;

-- 부서별/년도별 입사한 사원수 검색 (group by절)

select deptno, extract(year from hiredate) 년도, count(\*) 사원수

from emp

group by deptno, extract(year from hiredate)

order by deptno;

-- 부서별/년도별 입사한 사원수 검색 (가로로 출력)

select \*

from (select deptno, to\_char(hiredate, ‘yyyy’) 연도 form emp)

pivot (count(\*) for 연도 in (‘1980’, ‘1981’, ‘1982’, ‘1987’))

order by deptno;

-- 사원명, 봉급, 입사일, 소속부서평균봉급, 입사순서, 봉급비율, 먼저입사한사원명 검색

select ename, sal, hiredate, round(avg(sal) over(partition by deptno)) 소속부서평균봉급,

round(avg(sal) over (order by hiredate asc) 입사순서,

ltrim(round(ratio\_to\_report(sal) over(), 3)) 봉급비율,

lag(ename, 1) over (order by hiredate) 선임사원명

from emp;

-- 이름순으로 정렬 후 SMITH의 위치

select rank(‘SMITH’) within group (order by ename)

from emp;

-- ntile()

select ename, sal, ntile(4) over (order by sal)

from emp;

-- 부서번호별 최대급여, 최대급여수급자

select deptno, max(sal) 최대급여, min(ename) keep (dense\_rank first over by sal desc) 최대급여수급자

from emp

group by deptno;

-- 최대봉급, 최대봉급수급자, 최대봉급수급자의직급, 부서내최대봉급, 부서내최대봉급수급자

select ename, sal, deptno, max(sal) over() 최대봉급, first\_value(ename) over (order by sal desc) 최대봉급수급자,

last\_value(job) over(order by sal rows between current row and unbounded following) 최대봉급수급자의직급

max(sal) over(partition by deptno) 부서내최대봉급,

first\_value(ename) over(partition by deptno order by sal desc) 부서내최대봉급수급자

from emp

order by deptno;

-- 관리자명이 ‘KING’인 사원들의 이름, 직급 검색 / 셀프 조인

select e.name 사원명, m.ename 관리자명

from emp e, emp m

on e.mgr = m.empno

where m.ename=’KING’;

-- 관리자명이 ‘KING’인 사람들의 이름, 직급 (null도 출력) / 외부 조인

select e.ename 사원명, m.ename 관리자명, e,job

from emp e, emp m

where e.mgr = m.empno(+) and m.ename = ‘KING’;

-- 다중 데이터 조인(‘SALESMAN’에 한하여 사원명, 관리자명, 소속부서명)

select e.ename 사원명, m.ename 관리자명, dname 소속부서명

from emp e join emp m on e.mgr = m.empno

join dept on e.deptno = dept.deptno

where e.job = ‘SALESMAN’;

-- 외부조인

select e.ename 사원명, m.ename 관리자명, dname 소속부서명

from emp e, emp m, dept

where emp e left outer join emp m on e.mgr = m.empno

right outer join dept on e.deptno = dept.deptno;

-- 소속부서가 없는 사원정보 추가 행

INSERT INTO emp(empno, ename, job, mgr, hiredate, sal)

VALUES(7777, ‘MKCHOI’, ‘ANALYST’, 7782, ‘88/10/10’, 3000);

-- 모든 사원명, 모든 부서명 검색(소속부서가 없는 사원명, 소속사원이 없는 부서도 검색)

select e.ename 사원명, m.dname 관리자명, dname 소속부서명

from emp e left outer join emp m on e.mgr = m.empno

full outer join dept on e.deptno = dept.deptno;

-- 서브쿼리: 하나의 SELECT 문장의 절 안에 포함된 또 하나의 SELECT 문장

-- (메인쿼리-outer 쿼리, 서브쿼리-inner 쿼리)

-- 전체 사원들의 평균 봉급보다 많은 봉급을 받는 사원명, 봉급 검색

select ename, sal from emp

where sal > (select avg(sal) from emp);

-- ‘SCOTT’과 동일한 직급을 가진 사원명, 봉급

select ename, sal from emp

where job = (select job from emp where ename=’SCOTT’);

-- IN 연산자

-- ‘CHICAGO’, ‘DALLAS’에 위치한 부서에 소속된 사원명 검색

select ename from emp

where deptno in (select dept from dept where loc in (‘CHICAGO’, ‘DALLAS’));

-- ANY 연산자

-- 직급이 SALESMAN인 사원들의 어떠한 급여보다 많은 급여를 받는 사원명 검색

select ename, sal from emp

where sal >= any (select sal from emp where job = ‘SALESMAN’);

-- ALL 연산자

-- 직급이 SALESMAN인 사원들의 모든 급여(1600, 1500, 1250)보다 많은 급여를 받는 사원명 검색

select ename, sal from emp

where sal >= all (select sal from emp where job = ‘SALESMAN’);

-- EXISTS 연산자 (상호연관쿼리)

select ename, job from emp o

where exists (select \* from emp where mgr = o.empno);

-- 관리사원(부하직원)이 있는 사원이름(관리자명), 직급 검색

select ename, job from emp

where empno in (select distinct mgr from emp);

-- 관리자 번호 검색

select distinct mgr

from emp;

-- 소속사원이 있는 부서명 검색

select dname from dept

where exists (select \* from emp where deptno = dept.deptno);

-- 소속사원이 없는 부서명 검색

select dname frm dept

where deptno not in (select deptno from emp where deptno is not null);

-- ‘SCOTT’과 동일한 부서에서 근무하고 직급이 동일한 사원명 검색

select ename from emp

where (deptno, job = (select deptno, job from emp where ename = ‘SCOTT’) and ename <> ‘SCOTT’;

-- ‘SCOTT’과 동일한 직급을 가진 사원명 검색

select ename from emp o

where exists (select \* from emp I where i.ename = ‘SCOTT’ and i.job = o.job);

-- ‘SCOTT’과 동일한 부서에 속한 사원명 검색

select ename frm emp o

where exists (select \* from emp I where i.ename = ‘SCOTT’ and i.deptno = o.deptno);

-- 다중 컬럼 서브쿼리

-- 부서별 최대봉급을 받는 사원명

select deptno, ename from emp

where (deptno, sal) in (select deptno, max(sal) from emp group by deptno);

-- jones보다 많은 봉급을 받는 사원명, 직급 검색

select ename, sal from emp

where sal > any (select sal from emp where ename = ‘JONES’);

-- 직속상사가 ‘jones’인 사원의 사원명, 입사일, 소속부서명 검색

select ename, hiredate, dname

from emp, dept

where mgr = (select empno from emp where ename = ‘JONES’) and emp.deptno = dept.deptno;

-- 부서명 RESEARCH에 소속된 사원명, 봉급 검색(서브쿼리, exists 사용)

select ename, sal

from emp e

where exists (select \* from dept d where e.deptno = d.deptno and d.dname = ‘RESAECH);

-- 30번에 근무하는 사원들의 모든 급여보다 많은 급여를 받는 사원명

select ename from emp

where sal > all (select sal from emp where deptno = 30);

-- 직급별 최소봉급을 받는 사원명, 봉급, 직급 검색

select ename, sal, job

from emp

where (job, sal) in (select job, min(sal) from emp group by job);

-- 서브쿼리

-- 1) 스칼라 서브쿼리

-- 2) 인라인 뷰(동적뷰) – FROM 부속질의

-- 3) 중첩질의: WHERE 부속질의 (상관 또는 비상관 형태)

-- 모든 사원의 이름, 봉급, 그 사원이 속한 부서의 평균봉급

select ename, sal, (select avg(sal) from emp where deptno = o.deptno)

from emp o;

-- 분석함수로

select ename, sal, avg(sal) over(partition by deptno)

from emp;

-- 부서명별 사원수

select dname, count(\*)

from emp join dept

on emp.deptno (+) = dept.deptno

group by ename;

-- 부서번호별 사원수

select deptno, (select dname from dept where deptno = emp.deptno), count(\*)

from emp

group by deptno;

-- 가장 많은 봉급을 받는 사원 5명 검색

select rownum, ename, sal

from (select ename, sal from emp order by sal desc)

where rownum <= 5;

-- 소속부서 평균봉급보다 많은 봉급을 받는 사원명, 봉급, 평균봉급

-- 부서번호별 평균봉급 inline view

select ename, sal, asal

from emp a, (select deptno, avg(sal) asal from emp group by deptno) b

where a.deptno = b.deptno and sal > asal;

-- View: 가상 테이블

-- 단순 뷰 복합 뷰

-- 하나의 테이블로 생성 여러 개의 테이블로 생성

-- 그룹 함수 사용 불가능 그룹 함수 사용 가능

-- distinct 사용 불가능 사용 가능

-- dml (데이터 조작어) 사용 가능 사용 불가능

-- 뷰 생성 시 권한 불충분한 경우 권한 부여

-- SQL> conn / as sysdba;

-- GRANT CREATE VIEW to scott;

-- [or replace] 옵션: 뷰 수정 가능

-- [force] 옵션: 기본 테이블 존재 여부에 상관없이 뷰 생성 / 강제적으로 뷰를 만들어라

-- [noforce] 옵션: 기본 테이블이 존재할 경우에만 생성 (기본값)

-- [with check option] 옵션: 주어진 제약 조건에 맞는 데이터만 입력 및 수정 허용

-- [with read only] 옵션: select만 가능한 읽기 전용 뷰 생성

-- 뷰 생성

CREATE or replace VIEW emp\_dno30

as

SELECT ename, job, sal, deptno

FROM emp

WHERE deptno = 30;

-- 뷰에 데이터 삽입

insert into emp\_dno30 values(‘MKCHOI’, ‘AN’, 1000, 20);

-- update

update emp\_dno30

set deptno = 10

where sal > 1000;

-- 뷰 조회

select \* from emp\_dno30;

-- 뷰 정보 조회

desc user\_views;

select view\_name, text, read\_only

from user\_views;

-- 데이터언어:

-- 데이터정의어: 객체 생성(create), 수정(alter), 삭제(drop)

-- 데이터조작어: 삽입(insert), 삭제(delete), 갱신(update)

-- 데이터제어어:

-- 데이터검색어: select

-- 사원명, 소속부서명, 직속상사명, 봉급, 입사일 검색 (복합 뷰라서 수정은 불가능)

create or replace view emd(ename, dname, mname, sal, hiredate)

as

select e.ename, dname, m.ename, e.sal, e.hiredate

from emp e, emp m, dept

where e.mgr = m.empno (+) and e.deptno = dept.deptno;

-- Materialized View (MVIEW) – 구체화된 뷰 / 실체화된 뷰

SCOTT> CONN / AS SYSDBA;

SYS> GRANT query rewrite to scott;

SYS> GRANT create materialized view to scott;

-- MView 생성

CREATE MATERIALIZED VIEW m\_emp

build immediate

refresh on demand

-- enable query rewrite

as

select deptno, count(\*)

from emp

group by deptno;

-- MView 삭제

drop materialized view m\_emp;

-- 사원명, 소속부서명, 직급, 봉급을 검색하는 view(view\_ed) 생성

create or replace VIEW view\_ed

as

select ename, dname, job, sal

from emp, dept

where emp.deptno = dept.deptno;

-- 생성된 뷰를 활용하여 3명 이상 소속사원이 있는 부서들의 부서명, 사원수, 평균급여 검색

select dname, count(\*), round(avg(sal), 1)

from view\_ed

group by dname

having count(dname) >= 3;

-- 사용자가 생성한 뷰 정보 조회

desc user\_views;

select view\_name, text, read\_only

from user\_views;

-- 생성된 뷰 삭제

drop view view\_ed;

-- 생성된 뷰를 통해 새로운 사원 정보(이름, 봉급, 직급)를 입력해보자, 삽입가능한가?

-- 복합 뷰이기 때문에 삽입이 불가능하다

insert into view\_ed(ename, sal, job) values(‘KIM’, 1000, ‘SALESMAN’);

-- 가장 최근에 입사한 사원 5명의 정보(사원명, 직급, 소속부서번호, 입사일) 검색

-- 인라인 뷰 사용

select rownum, ename, job, deptno, hiredate

from (select rownum, ename, job, deptno, hiredate from emp order by hiredate)

where rownum <= 5;

-- 1981년대에 입사한 사원들의 사원명, 봉급, 소속부서평균봉급 검색

-- 스칼라 서브쿼리 사용

select a.ename, a.sal, (select avg(sal) from emp b where a.deptno = b.deptno)

from emp a

where extract(year from hiredate) = 1981;

-- 소속사원이 있는 부서명 검색(다중행 서브쿼리, exists 서브쿼리 사용)

select d.dname

from dept d

where exists (select 1 from emp e where d.deptno = e.deptno);

-- 테이블 구조 생성, 변경 및 삭제하는 DDL(Data Definition Language)

-- 테이블 구조 생성

create table emp01(empno number(4), ename varchar(20), sal number(7, 2));

desc emp01;

select \* from emp01;

-- 서브 쿼리로 테이블 생성(제약 조건 없이 생성)

create table emp01

as

select \* from emp;

-- 레코드 없는 테이블 구조만 복사

create table emp01

as

select \* from emp

where 1 = 2;

-- 테이블 구조 삭제

drop table emp01;

-- 테이블 구조 변경(alter table)

-- 새로운 컬럼 추가(add)

alter tble emp01

add (memo varchar(20));

-- 기존 컬럼 수정(modify)

alter table emp01

modify (memo varchar2(30));

-- 기존 컬럼 삭제(drop)

alter table emp01

drop column memo;

-- 컬럼명 변경(rename)

alter table emp01

rename column memo to bigo;

-- 읽기 전용 테이블로 변경

alter table emp01

read only;

-- 읽고 쓰기

alter table emp01

read write;

-- 테이블의 내용을 추가, 수정, 삭제하는 DML(Data Manipulation Language)

-- 데이터 조작어

-- 새로운 행 삽입

create table emp01

as

select \* from emp where 1=2;

insert into emp01(empno, ename, sal, hiredate, deptno)

values (7777, ‘mkchoi’, 3000, ‘2021/12/31’, 10);

insert into emp01

values (8888, ‘mkchoi’, ‘top’, 7832, ‘2021/12/31’, 5000, null, 20);

-- 기존 테이블 데이터를 기반으로 삽입

-- 서브쿼리로 다중 행 삽입

insert into emp01(empno, ename)

select empno, ename

from emp

where job = ‘MANAGER’;

insert into emp01

select \*

from emp

where deptno in (10, 20);

-- 다중 테이블에 다중 행 입력

create table empmgr

as

select empno, ename, mgr, hiredate

from emp where 1=2;

create table emphire

as

select empno, ename, hiredate, sal

from emp where 1=2;

insert all

when hiredate >= ‘1985/1/1’ then

into empmgr values(empno, ename, mgr, hiredate)

when sal >= 2000 then

into emphire values(empno, ename, hiredate, sal)

select empno, ename, mgr, hiredate, sal

from emp;

-- 테이블 삭제

drop table empmgr;

-- 테이블 행 제거

truncate table empmgr;

-- insert all vs first

create table empsal1 – empsal2, empsal3 생성

as

select empno, ename, sal, job

from emp

where 1=2;

insert first

when sal >= 3000 then

into empsal1 values(empno, ename, sal, job)

when sal >= 2000 then

into empsal2 values(empno, ename, sal, job)

else

into empsal3 values(empno, ename, sal, job)

select empno, ename, sal, job

from emp;

-- 테이블 내용을 수정하기 위한 update문

-- 모든 행 변경

-- 모든 사원의 봉급을 10% 인상된 값으로 수정

update empsal1

set sal = sal \* 1.1;

-- 특정 행만 변경

-- 직급이 ‘MANAGER’인 사원의 봉급을 10% 인상된 값으로 수정

update empsal1

set sal = sal \* 1.1

where job = ‘MANAGER’;

-- ‘DALLAS’에 위치한 부서에 근무하는 사원들의 커미션을 50으로 변경

update emp01

set comm 50

where deptno = (select deptno from dept where loc = ‘DALLAS’);

-- 다중 컬럼값 변경

update emp01

set hiredate = sysdate, sal = 10000, comm = 500

where ename = ‘SCOTT’

-- 모든 사원의 급여, 입사일을 ‘KING’ 사원의 급여, 입사일과 동일한 값으로 변경

update emp01

set (sal, hiredate) = (select sal, hiredate from emp where ename = ‘KING’);

-- 테이블의 데이터를 삭제하기 위한 delete문

delete from emp01;

-- 조건에 맞는 특정행만 삭제

delete from emp01 where deptno = 10;

-- 서브쿼리 사용

-- 부서명이 ‘SALES’인 부서에서 근무하는 사원 정보 삭제

delete from emp01

where deptno = (select deptno from dept where dname = ‘SALES’);

-- 테이블 데이터 병합하기 위한 merge문 (update + insert)

create table empcp

as

select \* from emp;

create table emp20

as

select \* from emp

where deptno = 20;

-- empcp 테이블의 데이터를 원본으로 emp20에 데이터가 이미 있으면 sal값 10% 인상(update), 없으면 새로 insert

merge into emp20

using empcp

on (emp20.empno = empcp.empno)

when matched then

update set emp20.sal = empcp.sal\*1.1

when not matched then

insert (empno, ename, sal) values(empcp.empno, empcp.ename, empcp.sal);

-- 외래키를 만들기 위해선 기본키로 설정되어 있어야 함

CREATE TABLE empcp(

EMPNO NUMBER(4, 0) primary key, -- not null, unique

ENAME VARCHAR2(10) unique,

JOB VARCHAR2(9) constraint empcp\_job\_nn not null,

MGR NUMBER(4, 0),

HIREDATE DATE,

SAL NUMBER(7, 2) check (sal > 500),

COMM NUMBER(7, 2),

DEPTNO NUMBER(2, 0) references dept(deptno));

insert into empcp values (8888, ‘SMITH2’, ‘CLERK’, 7902, ‘1980-12-17’, 300, null, 20);

-- pk 제약 조건 추가

alter table dept

add constraint dept\_deptno\_pk primary key(deptno);

-- 제약 조건 조회

select constraint\_name, constraint\_type, table\_name, R\_CONSTRAINT\_NAME

from user\_constraints;

-- 어떤 컬럼에 어떤 제약 조건이 지정되었는지 조회

select \*

from user\_cons\_columns;

-- 외래키 제약 조건

ALTER TABLE empcp

ADD CONSTRAINT empcp\_mgr\_fk FOREIGN KEY(mgr) REFERENCES empcp(empno)

on delete set null;

-- 기존 제약 조건 삭제

alter table empcp

drop constraint empcp\_mgr\_fk

-- 제약 조건 비활성화/활성화 (enable/disable)

alter table empcp

disable constraint EMPCP\_MGR\_FK;

alter table empcp

enable constraint EMPCP\_MGR\_FK;

insert into empcp values (7788, ‘SM’, null, 9999, ‘1980-12-17’, 600, null, 30);

-- validate/novalidate (exceptions 테이블 활용)

-- 사용자 테이블 목록

select ‘drop table ‘ || table\_name || ‘ cascade constraints;’

from user\_tables;

-- 제약 조건 조회

select table\_name, constraint\_name, constraint\_type

from user\_constraints;

desc user\_constraints;

-- 제약 조건 추가 (not null)

alter table emp

modify ename constraint emp\_empname\_nn not null;

-- 제약 조건 추가 (pk)

alter table emp

add constraint emp\_empno\_pk primary key(empno);

insert into emp values (7782, ‘mkchoi’, ‘CLERK’, 7902, ‘1980-12-17’, 800, null, 20);

-- 제약 조건 비활성화

alter table emp

disable constraint emp\_empname\_nn;

alter table emp

disable constraint emp\_empno\_pk;

-- 제약 조건 활성화

alter table emp

enable novalidate constraint emp\_empname\_nn; -- 기존 데이터의 제약 조건 만족 여부 체크 X

alter table emp

enable validate constraint emp\_empname\_nn; -- 만족하는지 검사

-- exceptions 테이블 만들기

SQL> conn

Enter user-name: scott

Enter password:

Connected.

SQL> @?/rdbms/admin/utlexcpt.sql

-- exceptions에 입력

alter table emp

enable validate constraint emp\_empname\_nn

exceptions into exceptions;

-- exceptions에서 확인

select rowed, ename

from emp

where rowed in (select row\_id from exceptions);

-- 제약 조건 위배되는 행 수정

update emp

set ename = ‘KIM’

where rowid = ‘AAAF8fAABAAALCxAAN’

-- emp 테이블의 rowed

select rowed, ename

from emp;

-- exceptions 테이블의 행 비우기

truncate table exceptions;

-- 1. emp 테이블과 동일한 구조로 된 테이블 복사(empcp)

-- dept 테이블과 동일한 구조로 된 테이블 복사(deptcp)

create table empcp

as

select \* from emp where 1=2;

create table deptcp

as

select \* from dept where 1=2;

-- 2. emp 테이블에서 dept가 10, 20인 데이터 empcp 테이블에 삽입

-- 모든 dept 데이터 deptcp 테이블에 삽입

insert into empcp

select \* from emp

where deptno in (10, 20);

insert into deptcp

select \* from dept;

-- 3. empcp 테이블 empno에 기본키 제약 조건, deptno에 deptcp 테이블의 deptno를 참조하는 외래키 제약 조건 추가

-- (오류 발생 시 원인 이유 설명 및 해결

alter table empcp

add constraint empcp\_empno\_pk primary key(empno);

alter table empcp

add constraint empcp\_deptno\_fk foreign key(deptno) references deptcp(deptno)

on delete cascade;

-- 외래키를 설정하기 위해 참조하는 키는 반드시 고유키 또는 기본키여야 하는데 그렇지 않아 오류가 발생함

-- deptcp 테이블의 deptno에 기본키 제약 조건을 설정하여 해결한다

alter table deptcp

add constraint deptcp\_deptno\_pk primary key(deptno);

-- 4. empcp 테이블의 ename에 not null 제약 조건 지정

alter table empcp

modify ename constraint emp\_ename\_nn not null;

-- not null 제약 조건은 add 대신 modify 명령문을 사용

-- 5. 사용자 소유의 테이블 목록 조회

select \*

from user\_tables;

-- 6. 생성된 제약 조건 정보 조회

select constraint\_name, constraint\_type, table\_name, R\_CONSTRAINT\_NAME

from user\_constraints;

-- 7. empcp 테이블에 새로운 데이터 삽입

-- (7777, ‘hyupsung’, ‘CLERK’, 7782, ‘2000-01-23’, 1300, 500, 50);

-- 삽입 가능한가? 오류가 발생했다면 오류 원인은?

insert into empcp values (7777, ‘hyupsung’, ‘CLERK’, 7782, ‘2000-01-23’, 1300, 500, 50);

-- 삽입이 불가능, deptno에 값을 넣으려고 했지만 deptno가 참조하는 parent key에 50이 없어서 오류가 발생한다

-- 8. exceptions 테이블 생성(@?/rdbms/admin/utlexcpt.sql)

-- sql command line에서 실행

SQL> conn

Enter user-name: scott

Enter password:

Connected.

SQL> @?/rdbms/admin/utlexcpt.sql

Table created.

-- 9. empcp 테이블 deptno에 지정된 외래키 제약 조건 비활성화

alter table empcp

disable constraint empcp\_deptno\_fk;

-- 10. 다시 empcp 테이블에 새로운 데이터 삽입

-- (7777, ‘hyupsung’, ‘CLERK’, 7782, ‘2000-01-23’, 1300, 500, 50);

-- 삽입 가능한가?

insert into empcp values (7777, ‘hyupsung’, ‘CLERK’, 7782, ‘2000-01-23’, 1300, 500, 50);

-- 삽입 가능하다

-- 11. empcp 테이블 deptno에 지정된 외래키 제약 조건 다시 활성화

-- (제약조건에 위배되는 데이터 행은 exceptions 테이블에 저장하도록 지정)

alter table empcp

enable validate constraint empcp\_deptno\_fk

exceptions into exceptions;

-- 12. exceptions 테이블 내용 조회

select \* from exceptions;

-- 13. empcp 테이블에서 exceptions 테이블에 기록된 행과 일치하는 데이터(rowed, empno, deptno)를 조회

select rowid, empno, deptno

from empcp

where rowid in (select row\_id from exceptions);

-- 14. 위에서 조회된 데이터를 참고하여 무결성 제약조건에 위배되는 데이터 수정

update empcp

set deptno = 20

where rowid = ‘AAAF8sAABAAALD5AAA’

-- 15. deptcp 테이블의 기본키 제약 조건 삭제 (참조하는 테이블이 존재할 경우 마지막에 cascade 옵션 사용)

alter table deptcp

drop constraint DEPTCP\_DEPTNO\_PK cascade;

-- 16. empcp, deptcp 테이블의 모든 데이터 삭제, exceptions 테이블 데이터도 삭제

truncate table empcp;

truncate table deptcp;

truncate table exceptions;

-- 17. empcp, deptcp 테이블 삭제

drop table empcp;

drop table deptcp;

-- 1) 인덱스를 활용하여 정렬한 효과를 내는 방법

-- 예제 테이블 생성

create table empindex

as

select empno, ename, sal

from emp

where deptno in (10, 20)’;

-- ename 열에 index 생성

create index idx\_empi\_ename

on empindex(ename);

-- 인덱스 정보 확인

select \* from user\_indexes;

-- 인덱스 삭제

drop index idx\_empi\_ename;

-- 인덱스 사용하지 않는 일반적인 SQL문

select ename from empindex;

-- 인덱스 사용 SQL문 – 빠르게 정렬된 데이터 검색 (오름차순)

select ename from empindex

where ename > ‘0’;

-- 2) 인덱스를 활용하여 최소값/최대값 구하는 방법

select min(ename) from empindex; -- 정렬 발생

select ename from empindex where ename > ‘0’ and rownum=1; -- stopkey

select max(ename) from empindex; -- 정렬 발생, 성능 떨어짐

-- 시퀀스 생성

create sequence dept\_deptno\_seq

increment by 10

start with 10

maxvalue 50

minvalue 10

cycle

cache 2;

-- 시퀀스 삭제

drop sequence dept\_deptno\_seq;

-- 시퀀스 관련 데이터 딕셔너리

desc user\_sequences;

-- 시퀀스 조회 및 수정

select sequence\_name, min\_value, max\_value, increment\_by, cycle\_flag, order\_flag, cache\_size, last\_number

from user\_sequences;

-- 시퀀스를 테이블의 기본 키에 적용

insert into dept01

values(dept\_deptno\_seq.nextval, ‘AAA’, ‘BBB’);

-- 시퀀스 수정

alter sequence dept\_deptno\_seq

maxvalue 1000;

-- 현재 시퀀스의 값 currval, 다음 값을 알아낼 때 nextval 사용

-- step 1. nextval로 새로운 값 생성

select dept\_deptno\_seq.nextval from dual;

-- step 2. 시퀀스의 현재값 알아보기

select dept\_deptno\_seq.currval from dual;

-- king 사원의 관리 사원들 조회

select level, lpad(ename, level\*4, ‘-‘), empno, mgr

from emp

start with ename = ‘KING’

connect by prior empno = mgr; -- 자식 쪽에 prior

select level, lpad(ename, level\*5, ‘ ‘), empno, mgr

frm emp

start with ename = ‘SMITH’

connect by empno = prior mgr; -- 부모 쪽에 prior

-- 사원명, 관리자명 : 조인 사용

-- prior로 관리자명 검색

select ename 사원명, prior ename 관리자명, level, lpad(‘ ‘, (level-1)\*2, ‘ ‘) || ename levelname, sys\_connect\_by\_path(ename, ‘/’) enamepath

from emp

start with mgr is null

connect by prior empno=mgr

order siblings by ename; -- 같은 레벨 안에서 순서를 지정

-- 계층구조에서 일부분만 계층화

select empno, ename 사원명, prior ename 관리자명, level, lpad(‘ ‘, (level-1)\*2, ‘ ‘) || ename level\_ename,

sys\_connect\_by\_path(ename, ‘/’) path\_ename

from emp

start with mgr is null

connect by prior empno = mgr and ename <> ‘JONES’ – 계층 구조 전개 조건

order siblings by ename;

-- 비고

-- where절 항상 마지막에 필터로 처리

select empno, ename 사원명, prior ename 관리자명, level, lpad(‘ ‘, (level-1)\*2, ‘ ‘) || ename level\_ename,

sys\_connect\_by\_path(ename, ‘/’) path\_ename

from emp

where ename <> ‘JONES’

start with mgr is null

connect by prior empno = mgr

order siblings by ename;

-- 루트사원, isleaf

select connect\_by\_root(ename) 루트사원, sys\_connect\_by\_path(ename, ‘\’) path, empno 사원번호, mgr 관리자,

connect\_by\_isleaf

from emp

start with mgr is null

connect by prior empno = mgr;

-- where절에 isleaf

select lpad(ename, level\*5, ‘\*’) ename, sys\_connect\_by\_path(ename, ‘/’) path

from emp

where connect\_by\_isleaf=0

start with empno = 7369

connect by empno = prior mgr;

-- level 별로 급여 합계와 사원수를 조회

select level, avg(sal) total, count(empno) c nt

from emp

start with mgr is null

connect by prior empno = mgr

group by level

order by level;

-- 1. 인덱스 생성

-- emp 테이블의 job 컬럼에 인덱스를 생성

create index idx\_empi\_job

on emp(job);

-- 생성된 인덱스 정보 확인

select \* from user\_indexes;

-- 위에서 생성된 인덱스 삭제

drop index idx\_empi\_job;

-- 2. 봉급 최소값을 검색하기 위한 질의문을 정렬을 사용하지 않고 인덱스를 활용하여 검색

select ename form emp where job > ‘0’ and rownum=1;

-- 3. 시퀀스 생성

-- emp 테이블의 emp열에 유일한 값을 생성해서 삽입하기 위한 시퀀스 생성

create sequence emp\_empno seq

increment by 1 – 증가값

start with 2107001 – 시작값

maxvalue 2107100 – 최대값

nocycle;

-- 시퀀스를 이용하여 emp 테이블에 새로운 행을 삽입

alter table emp

modify empno number(7);

insert into emp

values(emp\_empno\_seq.nextval, ‘DONG’, ‘MANAGER’, 9999, ‘98/11/25’, 1000, null, 20);

-- 현재까지 생성된 시퀀스 값 조회

select \*

frm user\_sequences;

-- 4. 계층형 질의

-- JONES 사원을 시작으로 부하 직원 목록 검색 (sys\_connect\_by\_path 활용)

select lpad(ename, level\*4, ‘ ‘) 사원명, sys\_connect\_by\_path(ename, ‘/’) 사원명경로

from emp

start with ename = ‘JONES’

connect by prior empno = mgr;

-- SMITH를 기준으로 상사 목록 검색

select lpad(ename, level\*4, ‘ ‘) 사원명, sys\_connect\_by\_path(ename, ‘/’) 사원명경로

from emp

start with ename = ‘SMITH’

connect by empno = prior mgr;

-- 계층 데이터에서 LEAF 노드(관리사원이 있는 사원)가 아닌 사원들의 사원명, 봉급 검색 (KING 기준 순방향)

select lpad(ename, level\*4, ‘ ‘) ename, sal sal

frm emp

where connect\_by\_isleaf = 0

start with ename = ‘KING’

connect by prior empno = mgr;

-- 계층형 쿼리를 활용하여 사원명, 관리자명을 검색(동일 레벨에서는 ename 순서로)

select lpad(ename, level\*4, ‘ ‘) 사원명, prior ename 관리자명

from emp

start with ename = ‘KING’

connect by prior empno = mgr

order siblings by ename;

-- 사원정보 출력 프로시저(%type 데이터 타입)

create or replace procedure Emp\_Info1(p\_empno in emp.empno%type)

is

v\_empno emp.empno%type;

v\_ename emp.ename%type;

v\_sal emp.sal%type;

begin

DBMS\_OUTPUT.enable;

select empno, ename, sal

into v\_empno, v\_ename, v\_sal

from emp

where empno = p\_empno;

-- 결과값 출력

DBMS\_OUTPUT.PUT\_LINE(‘사원번호 : ‘ || v\_empno);

DBMS\_OUTPUT.PUT\_LINE(‘사원이름 : ‘ || v\_ename);

DBMS\_OUTPUT.PUT\_LINE(‘사원급여 : ‘ || v\_sal);

end;

/

-- DBMS\_OUTPUT 결과값을 화면에 출력하기 위해

-- SET SERVEROUTPUT ON;

-- 사원정보 조회(복합데이터타입 %rowtype: 하나 이상의 데이터값을 갖는 데이터 타입으로 배열과 비슷한 역할이고 재사용 가능)

create or replace procedure Emp\_Info2(p\_empno in emp.empno%type)

is

v\_emp emp%ROWTYPE;

begin

DBMS\_OUTPUT.enable;

-- %ROWTYPE 변수 사용

select empno, ename, hiredate

into v\_emp.empno, v\_emp.ename, v\_emp.hiredate

from emp

where empno = p\_empno;

-- 결과값 출력

DBMS\_OUTPUT.PUT\_LINE('사원번호 : ' || v\_emp.empno);

DBMS\_OUTPUT.PUT\_LINE('사원이름 : ' || v\_emp.ename);

DBMS\_OUTPUT.PUT\_LINE('입 사 일 : ' || v\_emp.hiredate);

end;

/

-- 특정 부서 사원 정보 출력 프로시저 (PL/SQL 테이블 데이터 타입)

create or replace procedure Emp\_Info3(v\_deptno in emp.deptno%type)

is

type empno\_table is table of emp.empno%type

index by binary\_integer;

type ename\_table is table of emp.ename%type

index by binary\_integer;

type sal\_table is table of emp.sal%type

index by binary\_integer;

-- 테이블타입으로 변수를 선언해서 사용

empno\_tab empno\_table;

ename\_tab ename\_table;

sal\_tab sal\_table;

i binary\_integer := 0;

begin

DBMS\_OUTPUT.enable;

for emp\_list in(select empno, ename, sal from emp where deptno = v\_deptno) loop

I := I + 1;

empno\_tab(i) := emp\_list.empno;

ename\_tab(i) := emp\_list.ename;

sal\_tab(i) := emp\_list.sal;

end loop;

for cnt in 1..i loop

-- TABLE변수에 넣은 값을 뿌려줌

DBMS\_OUTPUT.PUT\_LINE('사원번호 : ' || empno\_tab(cnt));

DBMS\_OUTPUT.PUT\_LINE('사원이름 : ' || ename\_tab(cnt));

DBMS\_OUTPUT.PUT\_LINE('사원급여 : ' || sal\_tab(cnt));

DBMS\_OUTPUT.PUT\_LINE('------------------');

end loop;

end;

/

-- PL/SQL RECORD 데이터타입 : 여러 개의 데이터 타입을 갖는 변수들의 집합

create or replace procedure Emp\_Info4(p\_empno in emp.empno%type)

is

type emp\_record is record

(v\_empno number, v\_ename varchar2(30), v\_hiredate date);

emp\_rec emp\_record;

begin

DBMS\_OUTPUT.enable;

select empno, ename, hiredate

into emp\_rec.v\_empno, emp\_rec.v\_ename, emp\_rec.v\_hiredate

from emp

where empno = p\_empno;

DBMS\_OUTPUT.PUT\_LINE('사원번호 : ' || emp\_rec.v\_empno);

DBMS\_OUTPUT.PUT\_LINE('사원이름 : ' || emp\_rec.v\_ename);

DBMS\_OUTPUT.PUT\_LINE('입사일 : ' || emp\_rec.v\_hiredate);

end;

/

-- insert문

-- 사원 등록 프로시저

create or replace procedure Insert\_pro

(v\_empno in emp.empno%type,

v\_ename in emp.ename%type,

v\_deptno in emp.deptno%type)

is

begin

DBMS\_OUTPUT.enable;

insert into emp(empno, ename, hiredate, deptno)

values(v\_empno, v\_ename, sysdate, v\_deptno);

DBMS\_OUTPUT.PUT\_LINE('사원번호 : ' || v\_empno);

DBMS\_OUTPUT.PUT\_LINE('사원이름 : ' || v\_ename);

DBMS\_OUTPUT.PUT\_LINE('사원부서 : ' || v\_deptno);

DBMS\_OUTPUT.PUT\_LINE('데이터 삽입 완료');

end;

/

-- update

-- 수정: 특정 사원의 급여를 일정 % 인상/인하하는 프로시저

create or replace procedure Update\_Pro

(v\_empno in emp.empno%type,

v\_rate in number)

is

v\_emp emp%rowtype;

begin

DBMS\_OUTPUT.enable;

update emp

set sal = sal+(sal \* (v\_rate/100))

where empno = v\_empno;

DBMS\_OUTPUT.PUT\_LINE('데이터 수정 성공');

-- 수정된 데이터 확인하기 위해 검색

select empno, ename, sal

into v\_emp.empno, v\_emp.ename, v\_emp.sal

from emp

where empno = v\_empno;

DBMS\_OUTPUT.PUT\_LINE(' \*\*\*\* 수정 확인 \*\*\*\* ');

DBMS\_OUTPUT.PUT\_LINE('사원번호 : ' || v\_emp.empno);

DBMS\_OUTPUT.PUT\_LINE('사원이름 : ' || v\_emp.ename);

DBMS\_OUTPUT.PUT\_LINE('사원급여 : ' || v\_emp.sal);

end;

/

-- 함수

create or replace function Update\_fun

(v\_empno in emp.empno%type,

v\_rate in number)

return number

is

v\_emp emp%rowtype;

begin

DBMS\_OUTPUT.enable;

update emp

set sal = sal+(sal \* (v\_rate/100))

where empno = v\_empno;

DBMS\_OUTPUT.PUT\_LINE('데이터 수정 성공');

-- 수정된 데이터 확인하기 위해 검색

select empno, ename, sal

into v\_emp.empno, v\_emp.ename, v\_emp.sal

from emp

where empno = v\_empno;

DBMS\_OUTPUT.PUT\_LINE(' \*\*\*\* 수정 확인 \*\*\*\* ');

DBMS\_OUTPUT.PUT\_LINE('사원번호 : ' || v\_emp.empno);

DBMS\_OUTPUT.PUT\_LINE('사원이름 : ' || v\_emp.ename);

DBMS\_OUTPUT.PUT\_LINE('사원급여 : ' || v\_emp.sal);

return v\_emp.sal;

end;

/

-- 함수 실행

-- 1. 반환값 저장할 변수 선언

-- var sal number;

-- 2. 함수 실행

-- exec :sal := update\_fun(7839, 10);

-- 3. 변수값 확인

-- print sal;

-- 특정 부서의 평균급여와 사원수를 출력

create or replace procedure ExpCursor(v\_deptno dept.deptno%type)

is

-- 커서 정의

cursor dept\_avg

is

select dname, count(empno) cnt, round(avg(sal), 3) salary

from emp e, dept d

where e.deptno = d.deptno and e.deptno = v\_deptno

group by d.dname;

-- 커서를 패치하기 위한 변수 선언

v\_dname dept.dname%type;

emp\_cnt number;

sal\_avg number;

begin

-- 커서의 오픈

open dept\_avg;

-- 커서의 패치

fetch dept\_avg into v\_dname, emp\_cnt, sal\_avg;

DBMS\_OUTPUT.PUT\_LINE('부서명 : ' || v\_dname);

DBMS\_OUTPUT.PUT\_LINE('사원수 : ' || emp\_cnt);

DBMS\_OUTPUT.PUT\_LINE('평균급여 : ' || sal\_avg);

DBMS\_OUTPUT.PUT\_LINE('-----------------------');

-- 커서의 CLOSE

close dept\_avg;

end;

/

-- for문을 사용 : 커서의 OPEN, FRTCH, CLOSE가 자동 발생

-- 부서별 사원수와 급여 합계 출력

create or replace procedure ForCursor

is

cursor dept\_sum is

select dname, count(empno) cnt, sum(sal) salary

from emp e, dept d

where e.deptno = d.deptno

group by dname;

begin

-- cursor를 for문에서 실행시킨다

for emp\_list in dept\_sum loop

DBMS\_OUTPUT.PUT\_LINE('부서명 : ' || emp\_list.dname);

DBMS\_OUTPUT.PUT\_LINE('사원수 : ' || emp\_list.cnt);

DBMS\_OUTPUT.PUT\_LINE('급여합계 : ' || emp\_list.salary);

DBMS\_OUTPUT.PUT\_LINE('---------------------');

end loop;

end;

/

-- execute forcursor;

-- 커서의 속성 활용

create or replace procedure AttrCursor

is

v\_empno emp.empno%type;

v\_ename emp.ename%type;

v\_sal emp.sal%type;

cursor emp\_list is

select empno, ename, sal

from emp;

begin

DBMS\_OUTPUT.enable;

open emp\_list;

loop

fetch emp\_list into v\_empno, v\_ename, v\_sal;

-- 데이터를 찾지 못하면 빠져나갑니다

exit when emp\_list%notfound;

end loop;

DBMS\_OUTPUT.PUT\_LINE('전체 데이터 수 ' || emp\_list%rowcount);

close emp\_list;

end;

/

-- 트리거

create or replace trigger sum\_trigger

before

insert or update on emp

for each row

declare

avg\_sal number;

begin

select round(avg(sal), 3) into avg\_sal from emp;

DBMS\_OUTPUT.PUT\_LINE('급여 평균 : ' || avg\_sal);

end;

/

insert into emp(empno, ename, job, hiredate, sal)

values(1000, 'LION', 'SALES', SYSDATE, 5000);