SQL 활용

LifeisGoodandHappy

초기 계정 설정

1. 계정 생성 : 12c 부터 c##_계정이름으로 변경, 이 방식을 원하지 않으면

- alter session set "_ORACLE_SCRIPT"=true;

2. CREATE USER db7 IDENTIFIED BY 1234
   default tablespace USERS; --3번 안 해도 됨.

3. 권한 부여 : 접속, tablespace 할당

grant connect, resource to db7;

4. Tablespace에 공간 할당 명령

→ grant unlimited tablespace to db7;

초기 계정 설정

- 자신의 tablespace 확인

  select username, default_tablespace
  from dba_users;
  SELECT * FROM DBA_USERS
  ORDER BY USERNAME;
- select * from tab; 사용자의 테이블 보기

데이터 백업과 복원

- 콘솔에서 프롬프트에서 작업
- 1) import

imp db7/1234 file=c:/db7.dmp ignore=y full=y

- 2) export

exp userid=db7/1234 owner=db7 file=d:₩db7.dmp

※ 권한이 없는 경우

grant exp_full_database, imp_full_database to계정명

SQL PLUS 명령어

- conn : 계정으로 접속할 때
- show user : 접속자(사용자) 확인
- cl scr : 화면 깨끗이 지울 때
- column 컬럼명 format 종류:

An	문자크기	ex)column employer format a5 -> ‘Adam ’
9	숫자형식	ex)column salary format 99999 -> ‘ 12’
0	숫자형식	ex)column salary format 00000 -> ‘00012’
$	달러삽입	ex)column salary format $99999 -> ‘$ 12’
L	지역화폐	ex)column salary format L99999 -> ‘₩ 12’
.	소수표시	ex)column salary format 999.99 -> ‘ 12.12’
,	천단위	ex)column salary format 99,999 -> ‘ 1,234’

SQL PLUS 명령어

- linesize : 화면의 라인 길이를 조정

ex) set linesize 130; select * from tab;

- pagesize : 페이지의 크기를 조정

ex) set pagesize 50; select * from tab;

- l[ist] : 이전 명령어 출력
- / : 버퍼에 이전 쿼리문을 출력
- r : l+/ 합쳐 놓은 명령어

SQL PLUS 명령어

- save 파일명.ext [append | replace]

ex) save query.ext (이전 명령 저장)

- host : dos모드로 빠져나가기
- exit : sqlplus모드로 돌아옴
- ed query.ext : 저장된 명령어 편집하기
- @query.ext : 저장된 명령어 일괄 실행
- spool query.lst : 명령어 갈무리 시작
- spool off : 명령어 갈무리 끝

spool test ~ spool off

host로 도스에서 test.lst 파일을 열어서 확인

SQL

DDL	create, alter, drop, truncate
DQL	select - 함수, GROUP BY, JOIN, SUB-QUERY
DML	insert, update, delete
TCL	commit, rollback
DCL	grant, revoke

select – 구조, alias

[select 불러오는 목록 from 테이블 where 조건]

```
desc employee;
select * from employee where ename = 'KING';
select eno, salary, salary*12 as 연봉, commission from employee;
select eno, salary, salary*12 "1년 연봉", commission from employee;
select dno, count(dno) 인원수 from employee group by dno order by dno;
```

select – 조건절

- 숫자

select * from emp where salary > 1000;

- 문자

select * from emp where ename='KING';

- 날짜

select * from emp where hiredate > '81/01/01'

※ 날짜양식 '1981/01/01' , '19810101' 가능

select - 조건절

- 비교연산자, 논리연산자 적용 – 숫자 검색

select eno, ename, job from employee where dno=10 or dno=20;

select eno, ename, job from employee where dno in(10,20);

select eno, ename, salary from employee where salary >=2000 and salary<=3000;

select eno, ename, salary from employee where salary between 2000 and 3000;

select - 조건절

- 비교연산자, 논리연산자 적용 – 문자 검색

select eno, ename from employee where ename > 'A' and ename <'B';

※ like 연산자 - %, _ : 와일드카드(대치문자)

select eno, ename from employee where ename like '%A%';

select eno, ename from employee where ename like '__A%';

select - 조건절

- 비교연산자, 논리연산자 적용 – 날짜 검색

```
select eno,ename,hiredate from employee
where hiredate>='81/01/01' and hiredate<='81/12/31';
select eno,ename,hiredate from employee
where hiredate between '81/01/01' and '81/12/31';
```

select - 조건절

• not 연산자

```
select eno,ename,salary, dno from employee
where (dno=10 or dno=20);
select eno,ename,salary, dno from employee
where dno <> 30;
select eno,ename,salary, dno from employee
where not(dno=30);
select eno,ename,salary, dno from employee
where not salary not between 1000 and 2000;
select * from employee where not commission is not null;
```

select - 조건절

- 다중 조건절

```
select eno,ename,hiredate, dno from employee
where hiredate between '81/01/01' and '81/12/31' and
(dno=10 or dno=20);
select eno,ename,hiredate, dno from employee
where hiredate between '81/01/01' and '81/12/31' and dno
in(10,20);
```

select – order by

select ename, salary, commission from employee where commission is not null

order by salary asc, commission desc;

※ asc는 기본 적용,

오라클에서는 order by절보다 hint 사용을

추천함 → 속도 문제

SELECT /*+ INDEX(PK_EMP)*/* FROM EMPLOYEE;

SQL - null

- null 에 대한 처리

select eno, nvl(commission, 0),nvl2(commission,0,1) 연봉 from employee; //nvl : null value

```
SELECT DECODE(NULL, NULL, 1, 0) AS DECODE_NULL,
    CASE NULL WHEN NULL THEN 1 ELSE 0
    END AS CASE_NULL_1,
    CASE WHEN NULL IS NULL THEN 1 ELSE 0
    END AS CASE_NULL_2
FROM DUAL;
```

SQL - null

- null 에 대한 처리

```
select eno, nvl(commission, 0),nvl2(commission,100,
commission) 연봉 from employee; //nvl : null value


SELECT DECODE(NULL, NULL, 1, 0) AS DECODE_NULL,
    CASE NULL WHEN NULL THEN 1 ELSE 0
    END AS CASE_NULL_1,
    CASE WHEN NULL IS NULL THEN 1 ELSE 0
    END AS CASE_NULL_2
FROM DUAL;
```

column type

- char(size) 고정 길이 문자 데이터1~2000byte
- varchar2(size) 가변 길이 문자 데이터(1~4000byte)
- number(p,s) p:전체 자릿수, s:소수점 자릿수
- date 날짜 형식을 저장하기 위한 데이터 타입
- timestamp date타입의 확장된 형태
- blob 대용량의 바이너리 데이터 타입(~4gb)
- clob 대용량의 텍스트 데이터 타입(~4gb)

SQL – 문자 함수

```
select 'Oracle Database' from dual;
select upper('Oracle Database') from dual;
select lower('Oracle Database') from dual;
select initcap('dont''t stop me') from dual;
select length('hello world') from dual;
select lengthb('hello world') from dual;
select length('헬로우 월드') from dual;
select lengthb('헬로우 월드') from dual;
```

SQL – 문자 함수

```
select concat('Oracle', 'Database') from dual;
select concat('Oracle', ' '||'Database') from dual;
select cust_first_name||' '||cust_last_name from demo_customers;
select substr('Hello World',1,3) from dual;
select substr('Hello World',-3,3) from dual;
select ename from employee where substr(ename,-2,2)='NG'; ※ ename like '%NG';
```

SQL – 문자 함수

```
select instr('Hello World~!','W') from dual;
select instr('Hello World~!','l',4,2) from dual;
select substrb('헬로우 월드',4,3) from dual;
select instrb('헬로우 월드','로') from dual;
select instrb('헬로우 월드','월',7,1)from dual;
select rpad(ename,7,' ') || lpad(salary,4,' ') from employee;
select '100101-1234567', rpad(substr('100101-
1234567',1,8),14,'*') from  dual;
```

SQL – 문자 함수

```
select ltrim(' life is good') from dual;
select rtrim(' life is good ') from dual;
select trim('A' from 'AquaA') from dual;
select trim(' hello world  ') from dual;
select regexp_replace(' hello world  ', ' ', '') from dual;
select regexp_replace(' hello world  ', 'll', 'LL') from dual;
```

SQL – 문자 함수

문자열의 일부를 '*'로 대체하라.

```
SELECT ENAME,RPAD(SUBSTR(ENAME,1,1),LENGTH(ENAME)-1,'*')||SUBSTR(ENAME,-1,1) FROM EMPLOYEE;
```

```
SELECT ENAME,'*'||SUBSTR(RPAD(SUBSTR(ENAME,1,1),LENGTH(ENAME),'*'),2,LENGTH(ENAME)) FROM EMPLOYEE;
```

```
SELECT ENAME,SALARY,'*'||SUBSTR( RPAD(SUBSTR(SALARY,1,1),LENGTH(SALARY),'*'),2,LENGTH(SALARY)) FROM EMPLOYEE;
```

SQL – 문자 함수

```
select ENAME,regexp_replace(ename, '[A-Z]', '*') from employee;
SELECT ENAME, SALARY, regexp_replace(SALARY, '[0-9]', '*')
FROM EMPLOYEE;
select ENAME,salary,regexp_replace(salary, '[A-Za-z0-9]',
'*') from employee;
select ename, salary, TRANSLATE(salary,
'1234567890','**********') from employee; 갯수동일
select ename, translate(ename,
'ABCDEFGHIJKMNLOPQRSTUVWXYZ', '**************************')
FROM EMPLOYEE;
```

SQL – 문자 함수

```
--한글일 때 RPAD를 쓸경우는 +1을 해준다.
SELECT '이규',
CASE WHEN LENGTH('이규')=2 THEN SUBSTR('이규',1,1) || '*'
WHEN LENGTH('이규')=3 THEN SUBSTR('이규',1,1) || '*' || SUBSTR('이
규',-1,1)
ELSE RPAD(SUBSTR('이규',1,1),LENGTH('이규'),'*') || SUBSTR('이규',-1,1)
END AS "이름가리기“ FROM DUAL;

SELECT '이규훈',RPAD(SUBSTR('이규훈',1,1),LENGTH('이규훈
'),'*')||SUBSTR('이규훈',-1,1) FROM dual;
```

SQL – 문자 함수

```
SELECT RPAD( SUBSTR('123456-1234567', 1, INSTR('123456-1234567','-')), LENGTH('123456-1234567'),'*') FROM DUAL;


SELECT ENAME, CASE WHEN LENGTH(ENAME)=2 THEN RPAD(SUBSTR(ENAME,1,1),LENGTH(ENAME)-1,'*') ELSE RPAD(SUBSTR(ENAME,1,1), LENGTH(ENAME)-1,'*') || SUBSTR(ENAME,-1,1)
END AS "이름가리기“ FROM EMPLOYEE;
```

SQL – 숫자 함수

```
select 98.7654   from dual;
--반올림
select round(98.7654)   from dual;
select round(98.7654, 2)   from dual;
select round(98.7654, -1)   from dual;
--절삭
select trunc(98.7654)   from dual;
select trunc(98.7654, 2)   from dual;
select trunc(98.7654, -1)   from dual;
select floor(98.7654) from dual;
```

SQL – 숫자 함수

```
--절상
select ceil(98.765) from dual;
select ceil(12.345) from dual;
-- Quiz) 소수2째자리에서 절상하시오
select ceil(98.7654*power(10,3))/power(10,3) from dual;
```

SQL – 숫자 함수

```
select sum(salary)+sum(nvl(commission,0)) from employee;
select trunc(avg(salary),2) from employee;
```

SQL – 숫자 함수

```
--나머지연산자
select mod(31, 2) from dual;
select mod(31, 8) from dual;
--기타
select power(10,3) from dual;
select power(2,10) from dual;
select sqrt(16) from dual;
select sign(0) from dual;
```

SQL – 날짜 함수

```
select sysdate from dual;
select sysdate+1 from dual;
select sysdate-1 from dual;
SELECT SYSDATE-1 어제, SYSDATE 오늘, SYSDATE+1 내일 FROM DUAL;
SELECT SYSDATE-(SYSDATE-1) FROM DUAL;
select sysdate+100/24 from dual;
select localtimestamp, sysdate from dual;
```

SQL – 날짜 함수

포맷모델	단위
CC, SCC	4자리 연도 끝 두 글자
SYYYY, YYYY, YYY, YY, Y, YEAR	7월 1일 기준
DDD, J, DD	일 (시간을 기준)
HH, H12, HH24, AM	시 (분을 기준)
Q	한 분기의 두 번째 분기
MONTH, MON,MM,RM	월 16일을 기준
DAY, DY, D	요일 기준 반
MI	분 (초sec 29를 기준)

SQL – 날짜 함수

```
SELECT TO_CHAR(ROUND(TO_DATE(20510101,
'YYYYMMDD'),'CC'),'YYYY/MM/DD') FROM DUAL;
SELECT TO_CHAR(ROUND(TO_DATE(20191218,
'YYYYMMDD'),'Y'),'YYYY/MM/DD') FROM DUAL;
SELECT TO_CHAR(ROUND(TO_DATE(201906101,
'YYYYMMDD'),'Q'),'YYYY/MM/DD') FROM DUAL;
SELECT TO_CHAR(ROUND(TO_DATE(20191216,
'YYYYMMDD'),'MM'),'YYYY/MM/DD') FROM DUAL;
SELECT TO_CHAR(ROUND(TO_DATE(20191219,
'YYYYMMDD'),'D'),'YYYY/MM/DD') FROM DUAL;
```

SQL – 날짜 함수

```
SELECT TO_CHAR(ROUND(TO_DATE('20510813
130101','YYYYMMDD HH24:MI:SS'),'DD'), 'YYYY/MM/DD
HH24:MI:SS') FROM DUAL;

SELECT TO_CHAR(ROUND(TO_DATE('20510813
133101','YYYYMMDD HH24:MI:SS'),'HH'), 'YYYY/MM/DD
HH24:MI:SS') FROM DUAL;

SELECT TO_CHAR(ROUND(TO_DATE('20510813
133130','YYYYMMDD HH24:MI:SS'),'MI'), 'YYYY/MM/DD
HH24:MI:SS') FROM DUAL;
```

SQL – 날짜 함수

```
SELECT TO_CHAR(TRUNC(TO_DATE(20510101,
'YYYYMMDD'),'CC'),'YYYY/MM/DD') FROM DUAL;
SELECT TO_CHAR(TRUNC(TO_DATE(20191218,
'YYYYMMDD'),'Y'),'YYYY/MM/DD') FROM DUAL;
SELECT TO_CHAR(TRUNC(TO_DATE(201906101,
'YYYYMMDD'),'Q'),'YYYY/MM/DD') FROM DUAL;
SELECT TO_CHAR(TRUNC(TO_DATE(20191216,
'YYYYMMDD'),'MM'),'YYYY/MM/DD') FROM DUAL;
SELECT TO_CHAR(TRUNC(TO_DATE(20191219,
'YYYYMMDD'),'D'),'YYYY/MM/DD') FROM DUAL;
```

SQL – 날짜 함수

```
SELECT TO_CHAR(TRUNC(TO_DATE('20510813
130101','YYYYMMDD HH24:MI:SS'),'DD'), 'YYYY/MM/DD
HH24:MI:SS') FROM DUAL;

SELECT TO_CHAR(TRUNC(TO_DATE('20510813
133101','YYYYMMDD HH24:MI:SS'),'HH'), 'YYYY/MM/DD
HH24:MI:SS') FROM DUAL;

SELECT TO_CHAR(TRUNC(TO_DATE('20510813
133130','YYYYMMDD HH24:MI:SS'),'MI'), 'YYYY/MM/DD
HH24:MI:SS') FROM DUAL;
```

SQL – 날짜 함수

```
select ename, round(sysdate-hiredate)
근무일수 from employee;
select ename, round(months_between(sysdate,hiredate)) 근무월수 from employee;
select ename, round( months_between(sysdate,hiredate)/12) 근무년수 from employee;
```

SQL – 날짜 함수

```
SELECT ENAME,HIREDATE, SYSDATE,
TRUNC(MONTHS_BETWEEN(SYSDATE, hiredate)/12)||'년 ' ||
MOD(ROUND(MONTHS_BETWEEN( SYSDATE,hiredate)),12) || '월
' AS 근속년수
FROM EMPLOYEE;
SELECT ENAME,HIREDATE, ADD_MONTHS( HIREDATE, 6) FROM
EMP;
```

SQL – 날짜 함수

```
SELECT SYSDATE, NEXT_DAY( TRUNC(SYSDATE ), '월요일')
FROM DUAL;
```

```
SELECT TO_CHAR(LAST_DAY(TO_DATE('20200201',
'YYYYMMDD')), 'YYYY/MM/DD') FROM DUAL;
SELECT TO_CHAR((TO_DATE('20200301', 'YYYYMMDD')-1),
'YYYY/MM/DD') FROM DUAL;
```

SQL – 형 변환 함수

```
SELECT TO_CHAR(SYSDATE, 'YYYY-MM-DD AM HH24-MI-SS DAY') FROM DUAL;

SELECT ENAME, TO_CHAR (SALARY, ‘L9,999.00') FROM EMPLOYEE;

SELECT ENAME, TO_CHAR(HIREDATE,'DAY') FROM EMPLOYEE WHERE TO_CHAR(HIREDATE,'DY') IN('월','화','수');

SELECT ENAME, TO_CHAR(HIREDATE,'DY') FROM EMPLOYEE WHERE TO_CHAR(HIREDATE,'d') between 2 and 4;
```

SQL – 형 변환 함수

```
SELECT ENAME,HIREDATE FROM EMPLOYEE WHERE HIREDATE
= TO_DATE( 19810220,'YYYYMMDD');

SELECT TO_NUMBER('100,000', '999,000') FROM DUAL;
```

SQL - 일반함수

```
SELECT ENAME, SALARY,commission,
NVL(COMMISSION,0),NVL2(COMMISSION,SALARY*12+NVL(CO
MMISSION,0),SALARY*12) FROM EMPLOYEE; //nvl : not value


SELECT NULLIF('A','A'),NULLIF('A','B') FROM DUAL;
```

SQL - 일반함수

```
select ename, salary, commission, coalesce(commission, salary, 0) from employee; //인자의 수는 제한 없음

select ename, dno, decode(dno,

10, '회계부서', 20, '연구부서', 30, '영업부서',

40, '경영부서', '기타부서') as 부서명 from employee;

select ename, dno, case when dno=10 then '회계' when dno=20 then '연구' when dno=30 then '영업' when dno=40 then '경영' else '기타' end as dname from employee;
```

SQL - 일반함수

```
select ename, dno, salary, job,
   decode( job, 'ANALYST', salary+200,
       'SALESMAN', salary+180,
       'MANAGER', salary+150,
       'CLERK', salary+100,
       salary ) as 부서명
from employee;
```

SQL - 일반함수

```
select ename, dno,
case when job='ANALYST' then salary+200
     when job='SALESMAN' then salary+180
     when job='MANAGER' then salary+150
     when job='CLERK' then salary+100
     else salary
end as dname
from employee;
```

SQL - 그룹함수

```
select
 to_char(sum(salary),'$9,999,999') as 총급여,
 to_char(avg(salary),'$9,999.0') as 평균급여,
 to_char(max(salary),'$9,999.0') as 최대급여,
 to_char(min(salary),'$9,999.0') as 최소급여,
 count(nvl(commission,0)) as 총인원
from employee;
select count(distinct job) from employee;
select count(*) from employee;
select count(commission) from employee;
--null은 카운트에 포함되지 않는다.
```

SQL - 그룹함수

```
select sum(salary) from employee where dno=10;
select dno, sum(salary), floor(avg(salary)) from employee where dno=10 group by dno;
select dno, sum(salary), floor(avg(salary)) from employee group by dno order by dno;
select dno, sum(salary), floor(avg(salary)) from employee where dno in(10,20) group by dno order by dno;
select dno, count(eno), max(salary), min(salary) from employee group by dno order by dno;
```

SQL - 그룹함수

- 개별성과 군집성이 만날 때 에러발생

select dno, sum(salary) from employee group by dno; //군집성은 군집성끼리

Quiz) manager가 관리하는 사원의 수가 2명 이상인 manager를 출력하시오?

select manager, count(manager) from employee group by manager having count(manager) >= 2;

SQL – 순위 함수

select ename,salary,rank() over(order by salary desc) from employee;

select hiredate,ename,salary, dense_rank() over(order by salary desc) from employee;

select hiredate,ename,salary, dense_rank() over(order by salary desc, hiredate asc) from employee;

select dno,ename,salary, rank() over(partition by dno order by salary desc) from employee;

SQL – column 병기

부서번호가 10번인 직원들 목록을 출력하되 조회한 목록 중에 제일 낮은 직원을 병기하시오.

```
select a.eno,ename,salary,dno,
    FIRST_VALUE(salary) over(order by salary rows unbounded preceding) as 최저,
    salary - FIRST_VALUE(salary) over(order by salary rows unbounded preceding)
as 차이
from (select * from employee where dno=10) a;
```

부서번호가 10번인 직원들 목록을 출력하되 조회한 목록 중에 제일 높은 직원을 병기하시오.

```
select eno,ename,salary,dno, LAST_VALUE(salary) over(order by salary rows
between unbounded preceding and UNBOUNDED following) as 최고 from
(select * from employee where dno=10) order by eno;
```

SQL – TopN

1. TopN

```
select * from (select ename,salary,rank() over(order by salary desc) as 랭킹 from employee) where rownum <= 5;
--넘버링하려면 아래처럼
select * from
(select rownum rn, eno,ename,salary from
(select eno,ename,salary from employee order by salary)) where rn<=5;
```

※ 12c 이후 부터 적용 가능

```
select * from employee order by salary desc fetch first 5 rows only;
```

SQL –범위지정

1. 범위지정(paging 방식)

```
select * from(select rownum rnum, A.* from (select ename,salary,dense_rank() over(order by salary desc) as 랭킹 from employee)A)where rnum >= 11 and rnum<16;

select * from(select rownum rn, A.* from (select eno, ename,salary from employee order by salary desc)A)where rn between 1 and 5;
```

※ 12c 이후 부터 적용 가능

```
select * from employee order by salary desc offset 0 rows fetch next 5 rows only;
```

SQL – join

- Cross join
- Equi join
- Non Equi join
- Self join
- Outer join

SQL – join

- Cross join = Cartesian products

※조인되는 테이블로 표현한는 경우의 수

행의 개수=행X행, 열의 개수=열+열

행*행 by 열+열 로 연산됨.

```
select * from employee, department;
select * from employee cross join department;
```

SQL – join

- Equi join – ※inner 생략해도 상관 없음

select ename,e.dno,dname from employee e, department d where e.dno=d.dno;

select ename,dno,dname from employee natural inner join department;

select ename,dno,dname from employee inner join department using(dno);

select ename,d.dno,dname from employee e inner join department d on(e.dno=d.dno);

SQL – join

- Non Equi join

select eno,ename,salary,grade from employee e, salgrade s where salary between losal and hisal;

select eno,ename,salary,grade from employee e join salgrade s

on(salary between losal and hisal);

select eno,ename, salary,grade from employee natural join salgrade

where salary between losal and hisal;

Natural join은 적용 안됨.

SQL – join

• Non Equi join

Quiz) salgrade와 랭킹을 구하라

```
select ename, salary, grade,
    DENSE_RANK() over (order by salary desc) as 랭킹
from employee e, salgrade s
where e.salary between s.losal and s.hisal;
```

SQL – join

- Self join

```
select eno, ename,manager
--(select ename from employee where eno = e.manager) 매니저명 from employee e;
select e.eno, e.ename, m.ename 매니저명 from employee e, employee m
where e.manager = m.eno;
```

SQL – join

• Outer join => null 때문에 발생

```
select e.eno, e.ename, e.manager, m.ename
from employee e, employee m
where e.manager = m.eno(+);
```

※(+)가 반대일 경우는 관리자 아닌 사람 나옴

```
select e.eno, e.ename, e.manager, m.ename
from employee e left outer join employee m
on(e.manager = m.eno);
```

SQL – join

- Outer join

select e.eno, e.ename, e.manager, m.ename

from employee e right outer join employee m on(e.manager = m.eno);

※ manager 컬럼에서 manager가 아닌 사람

select eno, ename from employee

where eno not in (select distinct manager from employee where manager is not null);

SQL – join

- 집합

select ename,dno from employee where dno in(20,30)

union/union all/intersect/minus(except)

select ename,dno from employee where dno in(10,30);

※ 첫번째 쿼리문을 기준,출력열 동일해야함.

*는 적용됨, order by는 마지막에 추가

SQL - 서브쿼리

서브쿼리란? 쿼리에 삽입된 쿼리이다.

- select field

select eno,ename,(select dname from department where dno=e.dno) 부서명 from employee e;

Quiz)서브쿼리로 상관명 불러오시오.

select eno, ename, manager,(select ename from employee where eno=e.manager) 상관명 from employee e;

SQL - 서브쿼리

- 조건절

```
select ename, dno from employee
where salary = (select min(salary) from employee where dno=10);

select * from employee
where manager=(select manager from employee where ename='SCOTT');
```

SQL - 서브쿼리

- table

```
select rownum rn, A.* from
(select eno,ename,salary from employee order by salary desc)
A;

select * from (select rownum rn, A.* from
  (select * from employee order by salary
   desc) A) B where rn between 1 and 5;
```

SQL - 서브쿼리

- having scott의 부서외의 부서 최소 급여자 목록을 구하시오

select dno, min(salary) from employee group by dno having dno <> (select dno from employee where ename='SCOTT');

Quiz) 부서별 급여가 제일 작은 사원의 사번, 이름, 부서 번호, 급여를 출력 하시오.

select eno, ename, dno, salary from employee where salary in (select min(salary) from employee group by dno);

SQL - 서브쿼리

- 다중 행 서브쿼리

```
select ename, dno from employee
where dno in (select distinct dno from employee);
```

```
select eno,ename,salary,dno from employee where salary in
(select min(salary) from employee group by dno);
```

SQL - 서브쿼리

SELECT COLUMNS FROM TABLES WHERE EXISTS (subquery);

1)EXISTS는 오직 서브쿼리만 쓸 수 있다.

2)EXISTS는 유효한 **select,insert,update,delete** 문장에서만 사용 가능.

3)서브쿼리의 row가 하나라도 존재할 경우 SELECT 실행.

- **EXISTS A : 서브쿼리가 하나라도 존재하면 select문 실행**

SELECT ENO, ENAME, DNO FROM EMPLOYEE E WHERE EXISTS (SELECT DNO,DNAME FROM DEPARTMENT WHERE E.DNO=10);

- **NOT EXISTS A : 서브쿼리가 하나도 없으면 select문 실행**

SELECT * **FROM** suppliers **WHERE NOT EXISTS** (**SELECT** * **FROM** orders **WHERE** suppliers.supplier_id = orders.supplier_id);

Exists vs in

-EXISTS: **해당 row가 존재 하는지** 만 확인하고, 더 이상 수행되지 않음.

-IN: 실제 존재하는 데이터들의 모든 값까지 확인.

SQL - 서브쿼리

문제: 부서 번호가 30, 20, 10인 직원 출력하기

(1) IN 연산자 활용 시

```
select dno,eno from employee where dno in(30,20,10) order by 1,2;
```

(2) EXISTS 연산자 활용 시

```
select dno,eno from employee where exists(select dno from department where dno in (30, 20, 10) and dno=dno) order by 1, 2;
```

(3) EXISTS 연산자 활용 시

```
select dno,eno from employee where exists(select 1 from department where dno in(30,20,10) and dno=dno) order by 1, 2;
```

SQL - 서브쿼리

- 다중 행 서브쿼리

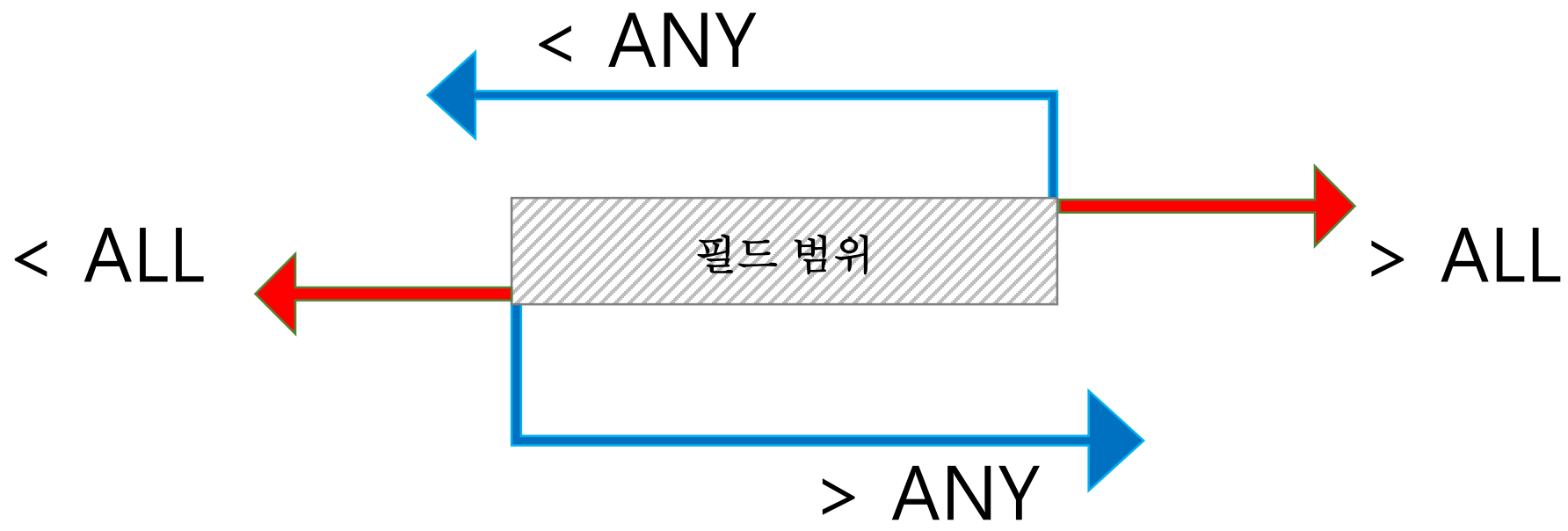
```
select eno,ename,job,salary from employee
where salary < any(select salary from employee where job='SALESMAN') and job<>'SALESMAN';
select eno,ename,job,salary from employee
where salary > any(select salary from employee where job='SALESMAN') and job<>'SALESMAN';
```

SQL - 서브쿼리

- 다중 행 서브쿼리

```
select eno,ename,job,salary from employee
where salary < all(select salary from employee where job='SALESMAN') and job<>'SALESMAN';
select eno,ename,job,salary from employee
where salary > all(select salary from employee where job='SALESMAN') and job<>'SALESMAN';
```

SQL - 서브쿼리



조인과 서브쿼리를 잘 쓰기 위한 비법

1. 찾고자 하는 데이터가 있는 테이블을 나열한다.
2. 나열된 테이블의 연결 고리를 찾는다.
3. 조인으로 할지, 서브쿼리로 할지 결정하고 코드로 실행한다.

SQL - DDL

- char(size) 고정 길이 문자 데이터1~2000byte
- varchar2(size) 가변 길이 문자 데이터(1~4000byte)
- number(p,s) p:전체 자릿수, s:소수점 자릿수
- date 날짜 형식을 저장하기 위한 데이터 타입
- timestamp date타입의 확장된 형태
- blob 대용량의 바이너리 데이터 타입(~4gb)
- clob 대용량의 텍스트 데이터 타입(~4gb)

SQL - DDL

- create

```
create table dept(dno number(2,0), dname varchar2(14),loc varchar2(13));
create table dept2
as select * from department;
create table dept_third
as select * from department where 1=0;
```

※ 테이블의 제약조건은 서브쿼리를 이용해서 작성된 경우 복사 불가.

SQL - DDL

- create with comments

```
COMMENT ON TABLE MEMBER_TBL_02 IS '회원정보명세서';
COMMENT ON COLUMN MEMBER_TBL_02.CUSTNO IS '회원번호';
COMMENT ON COLUMN MEMBER_TBL_02.CUSTNAME IS '회원성명';
COMMENT ON COLUMN MEMBER_TBL_02.PHONE IS '회원전화';
COMMENT ON COLUMN MEMBER_TBL_02.ADDRESS IS '주소';
COMMENT ON COLUMN MEMBER_TBL_02.JOINDATE IS '가입일자';
COMMENT ON COLUMN MEMBER_TBL_02.GRADE IS '고객등급';
COMMENT ON COLUMN MEMBER_TBL_02.CITY IS '거주도시';
SELECT * FROM USER_TAB_COMMENTS WHERE TABLE_NAME='MEMBER_TBL_02';
select * from user_col_comments;
```

SQL - DDL

- alter

```
alter table dept2 add(reg date);
alter table dept2 modify reg varchar2(20);
alter table dept2 rename column reg to r1;
alter table dept2 drop column r1;
alter table dept2 set unused (dno);
alter table dept2 drop unused columns;
```

SQL - DDL

- rename

```
rename employee2 to emp3;
alter table dept1 rename to dept_first;
```

- truncate

```
truncate table dept2;
```

- drop

```
drop table dept2;
```

Data Dictionary

- 접두어 user_ , all_ , dba_ 다양한 응용가능

※ 데이터사전 : 데이터베이스자원을 효율적으로 관리하기 위해 다양한 정보를 저장하는
시스템 테이블의 집합.

Data Dictionary

```
SELECT * FROM TAB;
SELECT * FROM USER_TABLES;
SELECT * FROM USER_OBJECTS WHERE OBJECT_TYPE='TABLE';

SELECT * FROM USER_IND_COLUMNS WHERE TABLE_NAME
IN('MAKER');
SELECT * FROM USER_INDEXES;
SELECT * FROM USER_OBJECTS WHERE OBJECT_TYPE='INDEX';

SELECT * FROM USER_SEQUENCES;
SELECT * FROM USER_OBJECTS WHERE OBJECT_TYPE='SEQUENCE';
```

Data Dictionary

```
SELECT * FROM USER_PROCEDURES;
SELECT * FROM DBA_OBJECTS WHERE OBJECT_TYPE='PROCEDURE';

SELECT * FROM USER_VIEWS;
SELECT * FROM USER_OBJECTS WHERE OBJECT_TYPE='VIEW';

SELECT * FROM USER_CONSTRAINTS WHERE TABLE_NAME='DEPT2';
```

SQL – DML

- insert – insert into 테이블명( ) values(값...)

```
insert into dept(dname, dno) values('ACC',10); //null이 묵시적으로 입력됨
insert into dept values(20,'RESEARCH',30);
insert into dept values(30,'SALES',NULL);
insert into d1 select * from d2 where dno=40;
```

SQL – DML

• insert – insert into 테이블명( ) values(값...)

```
insert into dept_third values(50,'DEVELOPMENT',40,SYSDATE);
insert into dept_third
values(50,'DEVELOPMENT',40,'2019/12/24');
insert into dept_third values(50,'DEVELOPMENT',40,
    TO_DATE('20191225','YYYYMMDD'));
```

SQL – DML

• insert – insert all [into 테이블명 values]

```
INSERT ALL
INTO tbl_a VALUES('CD01','1', '1111' ,'TEST')
INTO tbl_a VALUES('CD01','1', '2222' ,'TEST')
INTO tbl_a VALUES('CD01','1', '3333' ,'TEST')
INTO tbl_a VALUES('CD01','1', '4444' ,'TEST')
SELECT * FROM DUAL;
```

※ 반드시 끝에’SELECT * FROM DUAL’ 추가

SQL – DML

• update –

update 테이블명 set 열=값, … where 조건

update dept_THIRD set loc=10, dname='MANAGER' WHERE DNO=30;

※ WHERE 없으면 모든 행 업데이트 됨!

update dept_third set dname=(select dname from department where dno=30), loc=(select loc from department where dno=30) where dno=30

SQL – DML

• delete – delete (from) 테이블 where 조건

```
delete dept_third where dno = 50;
delete dept_third where LOC = (select LOC from loc where cityname='NEWYORK');
```

※ INSERT 후에 DDL을 사용하게 되면
INSERT 된 내용이 자동 COMMIT이 된다.

SQL – TCL

Transaction : 하나의 데이터처리 작업 단위
목적 : 일관성 보장, DML에서만 사용,
All-OR-Nothing 적용.
commit; //하나 이상의 트랜잭션의 처리.
rollback; //가장 최근 commit으로 회귀.
Ex) EMPS테이블을
작성하고 표와
같이 insert 하시오.

사원번호	이름	부서
10	김기획	기획부
20	박인사	인사부
30	최재무	재무부
40	오영업	영업부

SQL – TCL

Ex1) 사원번호가 40인 사원을 삭제하고 commit 하시오.
DELETE DEPT WHERE 사원번호=40;
COMMIT;
Ex2) 사원번호가 30인 사원을 삭제하시오
DELETE DEPT WHERE 사원번호=30;
Ex3) 'SAVEPOINT S1'을 설정하고 사원번호 20인 사원을 삭제하시오.
SAVEPOINT S1;
DELETE DEPT WHERE 사원번호=20;
Ex4) 'SAVEPOINT S2'을 설정하고 사원번호 10인 사원을 삭제하시오.
SAVEPOINT S2;
DELETE DEPT WHERE 사원번호=10;
Ex5) 'SAVEPOINT S2'까지 ROLLBACK하시오. ROLLBACK TO S2;
Ex6) 'SAVEPOINT S1'까지 ROLLBACK하시오. ROLLBACK TO S1;
Ex7) SAVEPOINT 없이 ROLLBACK하시오. ROLLBACK;

SQL – 제약성(CONSTRAINTS)

- PRIMARY KEY (컬럼 레벨O, 테이블 레벨O)

```
CREATE TABLE DEPT_SECOND(
    DNO NUMBER(2) CONSTRAINT PK_SEC PRIMARY KEY,
    DNAME VARCHAR2(14),
    LOC VARCHAR2(13));
CREATE TABLE DEPT3(
    DNO NUMBER(2),
    DNAME VARCHAR2(14),
    LOC VARCHAR2(13), //테이블레벨에서 복수 적용 가능
    CONSTRAINT pk_dept_third primary key(dno, dname));
ALTER TABLE DEPT3  ADD PRIMARY KEY(dno, dname);
```

SQL – 제약성(CONSTRAINTS)

```
ALTER TABLE EMP_SECOND ADD CONSTRAINT PK_EMP_SEC PRIMARY KEY(ENO);  //제약성 추가

ALTER TABLE EMP_SECOND DROP CONSTRAINT PK_EMP_SEC; //제약성 삭제

SELECT * FROM USER_CONSTRAINTS;

SHOW RECYCLEBIN; //제약성 삭제 휴지통비움

PURGE RECYCLEBIN;

//제약성조회

SELECT * FROM USER_CONSTRAINTS WHERE TABLE_NAME='EMP_SECOND';
```

SQL – 제약성(CONSTRAINTS)

- FOREIGN KEY (컬럼 레벨O, 테이블 레벨O)

```
create table dept ( dno number(2), dname varchar2(14),
 loc varchar2(2) CONSTRAINT FK_DEPT_LOC REFERENCES
    LOC(LOC) 상대테이블의 PRIMARY KEY에 연결해야 한다.
create table dept ( dno number(2), dname varchar2(14),
 loc varchar2(2),
 CONSTRAINT Fk_DEPT_LOC FOREIGN KEY(LOC)
  REFERENCES LOC(LOC));
```

SQL – 제약성(CONSTRAINTS)

- 추가, 이름변경, 삭제

① ALTER TABLE DEPT ADD CONSTRAINT FK_DEPT_LOC FOREIGN KEY(LOC) REFERENCES LOC(LOC); --추가

② ALTER TABLE DEPT RENAME CONSTRAINT FK_DEPT_LOC TO FK_DEPT_LOC2; --이름 변경

③ ALTER TABLE DEPT DROP CONSTRAINT pk_d1; --삭제

④ ALTER TABLE DEPT MODIFY CONSTRAINT --작동안됨FK_DEPT_LOC REFERENCES DEPARTMENT(DNO);

- 2가지를 모두 조회할 것

```
select * from user_objects;

select * from user_constraints where table_name = 'DEPT';
```

SQL – 제약성(CONSTRAINTS)

- FOREIGN KEY 예제 1

```
DROP TABLE DEPT_SECOND; DROP TABLE EMP_SECOND;
SELECT * FROM USER_CONSTRAINTS WHERE TABLE_NAME = 'EMP_SECOND';
create table EMP_SECOND AS SELECT * FROM EMPLOYEE;
create table DEPT_SECOND AS SELECT * FROM DEPARTMENT;
ALTER TABLE DEPT_SECOND ADD CONSTRAINT PK_DNO PRIMARY KEY(DNO);
//FOREIGN KEY의 해당 테이블에 기본키가 없으면 외래키 생성 불가
ALTER TABLE emp_second ADD CONSTRAINT FK_DNO FOREIGN KEY (DNO) REFERENCES DEPT_SECOND(DNO);
INSERT INTO EMP_SECOND(ENO, ENAME, DNO) VALUES(500,'LGH',60);//위배
INSERT INTO DEPT_SECOND(DNO,DNAME,LOC) VALUES(60,'MARKETTING',30);
INSERT INTO EMP_SECOND(ENO, ENAME, DNO) VALUES(500,'LGH',60);//가능
```

SQL – 제약성(CONSTRAINTS)

- FOREIGN KEY 예제 1

```
DELETE DEPT_SECOND WHERE DNO=60; //위배 자식노드
ALTER TABLE EMP_SECOND drop CONSTRAINT FK_DNO;
ALTER TABLE EMP_SECOND ADD CONSTRAINT FK_DNO2
FOREIGN KEY (DNO) REFERENCES DEPT_SECOND (DNO)
ON DELETE CASCADE / ON DELETE SET NULL;
//오라클은 TRIGGER 이용 ON UPDATE CASCADE를 활용함.
```

RESTRICTED: 기본값. 참조되면, 변경,삭제 연산 취소.

CASCADE: 변경, 삭제 참조되는 개체도 변경,삭제된다.

SET NULL: 변경, 삭제 참조되는 개체의 값을 NULL로 설정.

SQL – 제약성(CONSTRAINTS)

- FOREIGN KEY – 추가 삭제

```
ALTER TABLE TABLEName drop CONSTRAINT FK_NAME;
ALTER TABLE TABLENAME ADD CONSTRAINT FK_NAME FOREIGN KEY (FId)
REFERENCES OTHERTABLE (Id)
ON DELETE CASCADE / ON DELETE SET NULL;

ALTER TABLE EMP_SECOND drop CONSTRAINT FK_DNO2;
ALTER TABLE EMP_SECOND ADD CONSTRAINT FK_DNO2
FOREIGN KEY (DNO) REFERENCES DEPT_SECOND (DNO)
ON DELETE CASCADE;
```

SQL – 제약성(CONSTRAINTS)

- NOT NULL (컬럼 레벨O, 테이블 레벨X)

```
create table customer(
    id varchar2(20) not null,
    pwd varchar2(20) not null,
    name varchar2(30) not null,
    phone varchar2(30),
    address varchar2(100));
insert into customer values(null, null, null, '010-1111-1111','seoul');    //->
에러 제약성 위배
```

SQL – 제약성(CONSTRAINTS)

- NOT NULL (컬럼 레벨O, 테이블 레벨X)

ALTER TABLE [테이블명] MODIFY [컬럼명] NOT NULL

```
CREATE TABLE EMP2(
ENO NUMBER(2) CONSTRAINT NN_EMP2ENO NOT NULL,
    ENAME VARCHAR2(14),
    SALARY NUMBER(6));
SELECT * FROM USER_CONSTRAINTS;
ALTER TABLE EMP2 MODIFY ENO NOT NULL;
--추가할때 MODIFY사용
ALTER TABLE EMP2 MODIFY ENAME CONSTRAINT NN_EMP2_ENAME
NOT NULL;
```

SQL – 제약성(CONSTRAINTS)

- UNIQUE (컬럼 레벨O, 테이블 레벨O)

```
create table customer(
    id varchar2(20) unique  //컬럼레벨,
    pwd varchar2(20) not null,
    name varchar2(30) not null,
    phone varchar2(30),
    address varchar2(100));
    //CONSTRAINT UQ_CUSTOMER_ID UNIQUE(ID) );테이블레벨
insert into customer values(null, 'aa', 'bb', '010-1111-1111','seoul');
//insert됨,중복적용가능.null은unique제약조건 위반 안됨
```

SQL – 제약성(CONSTRAINTS)

- CHECK - *범위*를 설정(컬럼 레벨O, 테이블 레벨O)

```
create table emp_second(eno number(4) constraint
  emp_sec_eno_pk primary key,ename
  varchar2(10),SALARY NUMBER(7) CONSTRAINT
  CK_SAL_M CHECK(SALARY>0));
//제약 위배 INSERT INTO emp_second VALUES(10,'LGH',0);
create table emp_second(
   eno number(4) constraint emp_sec_eno_pk primary key,
   ename varchar2(10),
   SALARY NUMBER(7),
   CONSTRAINT CK_SAL_M CHECK(SALARY>0));
```

SQL – 제약성(CONSTRAINTS)

- DEFAULT (컬럼 레벨O, 테이블 레벨X)

```
create table emp_second(eno number(4) constraint emp_sec_eno_pk
primary key, ename varchar2(10),
SALARY NUMBER, COMMISSION NUMBER DEFAULT 0);


INSERT INTO EMP_SECOND(ENO,ENAME,SALARY) VALUES(10,'LGH',1000);
//강제로 NULL값을 넣으면 입력됨.
```

SQL – 제약성(CONSTRAINTS)

- 제약조건 추가하기

※ 서브 쿼리문으로 테이블 생성시 제약조건은 복사 안됨.

※ PRIMARY KEY, FOREIGN KEY는 ADD를 이용하여 추가.

✓ALTER TABLE EMP MODIFY ENAME CONSTRAINT EMP_NN NOT NULL;

✓ALTER TABLE EMP MODIFY ENAME CONSTRAINT EMP_N NULL;

✓ALTER TABLE EMP MODIFY ENAME CONSTRAINT EMP_UQ UNIQUE;

✓ALTER TABLE EMP MODIFY SALARY CONSTRAINT emp_chk check(SALARY NOT BETWEEN 0 AND 100);

✓ALTER TABLE EMP MODIFY SALARY DEFAULT 0;

SQL – 제약성(CONSTRAINTS)

- 제약조건 삭제하기

※ 제약 조건 적용된 테이블을 삭제 시 제약 객체 전부 같이 삭제.

✓ALTER TABLE EMP DROP CONSTRAINT EMP_NN;

✓ALTER TABLE EMP DROP CONSTRAINT EMP_N;

✓ALTER TABLE EMP DROP CONSTRAINT EMP_UQ;

✓ALTER TABLE EMP DROP CONSTRAINT emp_chk;

✓ALTER TABLE EMP MODIFY DEFAULT NULL;

SQL – 제약성(CONSTRAINTS)

- 제약조건 활성화 및 비활성화

※ 일시적으로 적용 가능

```
ALTER TABLE EMP_SECOND disable CONSTRAINT fk_emp_sec;
ALTER TABLE EMP_SECOND enable CONSTRAINT fk_emp_sec;
```

- 제약조건 조회

```
select * from user_constraints where table_name = 'EMP';
```

SQL – VIEW

```
CREATE VIEW V_EMP_DEP AS
SELECT ENAME 사원명, DNAME 부서명
FROM EMPLOYEE E, DEPARTMENT D
WHERE E.DNO = D.DNO;
SELECT * FROM V_EMPNAME_DEPNAME WHERE 사원명 = 'SCOTT';
```

※ 뷰의 장점:보안성과 사용자 편의성.
뷰 생성 후 조건절과 같이 사용하면 좋다.
별칭이 주어졌을 때 별칭으로 조회 할 것

SQL – VIEW

```
SELECT * FROM USER_VIEWS;
CREATE OR REPLACE VIEW V_EMP_JOB AS SELECT * FROM EMP_SECOND;
INSERT INTO V_EMP_JOB(ENO, ENAME, DNO, JOB) VALUES(8000, 'LGH', 30, 'SALESMAN');
SELECT * FROM V_EMP_JOB WHERE ENAME='LGH';
```

SQL – VIEW

VIEW는 가상열, 또는 조인 뷰에 의하여 하나 이상의 기본 테이블인 경우 INSERT를 사용할 수 없다. 또한 Virtual Column (가상 컬럼)이 있는 경우는 insert가 되지 않는다.

SQL – VIEW

```
CREATE VIEW V_EMP AS
SELECT DNO, SUM(SALARY) HAP FROM EMP_SECOND GROUP BY DNO;
SELECT * FROM V_EMP;
```

※ VIEW FIELD가 집계 함수면 DML 안 됨.

```
DROP VIEW V_EMP;
```

SQL – VIEW

CREATE VIEW V_EMP AS SELECT DNO, SUM(SALARY) HAP FROM EMP_SECOND GROUP BY DNO;

※ 뷰의 FIELD가 집계되면 DML발생 안함.

DROP VIEW V_EMP;

CREATE OR REPLACE FORCE VIEW V_EMP AS SELECT ENO,ENAME,SALARY FROM EMP_FINISH; //없는 테이블도 강제 생성됨.

SQL – VIEW

```
CREATE OR REPLACE FORCE VIEW V_EMP AS
SELECT ENO,ENAME,SALARY,JOB
FROM EMP_SECOND WHERE JOB='MANAGER' WITH CHECK OPTION;

INSERT INTO V_EMP(ENO,ENAME,SALARY,JOB)
VALUES(200,'LGH',1000,'SALES'); //삽입 안됨.
```

SQL – VIEW

```
CREATE OR REPLACE FORCE VIEW V_EMP2
AS SELECT ENO, ENAME, DNO, JOB FROM EMPLOYEE WHERE JOB LIKE 'MANAGER' WITH READ ONLY;

INSERT INTO V_EMP2(ENO, ENAME, DNO, JOB) VALUES(9000, 'LGH',10,'CLERK');
```

SQL – SEQUENCE

- 생성

```
create sequence sample_seq increment by 10 start with 10;
CREATE SEQUENCE sample_seq NOCYCLE MAXVALUE 10;
```

- 조회

```
SELECT * FROM USER_SEQUENCES;
select * from user_objects where object_type ='SEQUENCE';
```

- 삭제 DROP SEQUENCE sample_seq

SQL – SEQUENCE

- 시퀀스를 사용하는 예

```
create sequence seq_sample increment by 10 start with 10;
create table dept_copy as select * from department where 1=0;
insert into dept_copy values(seq_sample.nextval, 'research',30);
select * from dept_copy;
select seq_sample.currval from dual;
```

- 시퀀스를 사용하지 않는 경우 :: 증감의 독립성 보장 안됨

```
INSERT INTO EMP_SECOND VALUES(
(SELECT MAX(ENO)+1 FROM EMP_SECOND), 'LGH', 'sales', 7839,
SYSDATE,1000, 500,10);
```

SQL – SEQUENCE

- 특징

1. sequence는 객체이기 때문에 특정 테이블에 종속되지 않는다.
2. sequence의 증감 중에 에러가 발생해도 시퀀스의 값은 자동으로 증감한다.
3. sequence의 현재 값을 알기 위해서는 sequence.nextval를 먼저 호출한 뒤에

   sequence.currval로 현재 값을 알 수 있다.

SQL – SEQUENCE

SELECT SQ_DEP_COPY.NEXTVAL FROM DUAL;

SELECT SQ_DEP_COPY.CURRVAL FROM DUAL;

※ NEXTVAL을 호출해야 CURRVAL을 호출!

ALTER SEQUENCE SQ_EMP_COPY MAXVALUE 9999999 CYCLE CACHE 5;

DROP SEQUENCE SQ_EMP_COPY;

※ sequence 이상 증가 시

alter sequence DEMO_ORD_SEQ nocache;

SQL – INDEX

- 목적: 검색속도를 향상, 데이터 검색 용도

```
CREATE TABLE TEST( IDX NUMBER CONSTRAINT PK_IDX PRIMARY KEY,    NAME VARCHAR2(20));
INSERT INTO TEST VALUES(11, 'B1');
INSERT INTO TEST VALUES(13, 'Q1');
INSERT INTO TEST VALUES(12, 'E1');
INSERT INTO TEST VALUES(15, 'A1');
INSERT INTO TEST VALUES(14, 'M1');
select * from test;
select /*+ INDEX(TEST PK_IDX) */ * FROM TEST;
create index idx_test_name on test(name);
select /*+ INDEX(TEST idx_test_name) */ * FROM TEST where name >' ';
```

SQL – INDEX

- CREATE INDEX IDX_TEST ON TEST(NAME);
- ALTER INDEX IDX_TEST REBUILD;
- SELECT /*+ INDEX(TEST IDX_TEST) */* FROM TEST where name > '';
- DROP INDEX IDX_TEST;
- 오라클 힌트 사용

- Select /*+ Index(테이블명 인덱스명)*/ * from 테이블명; // 숫자 검색

- Select /*+ Index(테이블명 인덱스명)*/ * from 테이블명 where 컬럼 > ' '; // 문자 검색

SQL – INDEX

- 오라클 힌트 사용 없이 인덱스로만 조회

SELECT * FROM EMP_COPY ORDER BY ENAME; //내부적으로 정렬작업 후 보여줌

SELECT * FROM EMP_COPY WHERE ENAME > '0'; //인덱스(선정렬됨)를 보여줌

EX) 인덱스 전과 후

DROP INDEX IXENAME; //먼저 인덱스 없는 상태

SELECT * FROM EMP_COPY WHERE ENAME > '0';

CREATE INDEX IXENAME ON EMPLOYEE(ENAME);

SELECT * FROM EMPLOYEE WHERE ENAME > '0';

SQL – INDEX

```
ALTER INDEX IXENAME MONITORING USAGE;
ALTER INDEX IXENAME NOMONITORING USAGE;
SELECT * FROM V$OBJECT_USAGE WHERE INDEX_NAME='IXENAME';
SELECT * FROM USER_OBJECTS WHERE OBJECT_TYPE='INDEX';
SELECT * FROM USER_IND_COLUMNS WHERE TABLE_NAME IN('TEST');
ALTER INDEX IDX_EMP_ENAME REBUILD;
```

SQL – INDEX

- 고유/비고유

```
CREATE UNIQUE INDEX IDX_DNO ON DEP_COPY(DNO);
```

- 결합인덱스

```
CREATE INDEX IDX_DN ON DEP(DNO,LOC);
```

- 함수기반 인덱스

```
CREATE INDEX IDX_S ON EMP(SALALRY*12);
```

SQL – DCL

관리자 계정	사용자 계정
시스템 보안	데이터 보안
시스템 권한	객체 권한
DB 접근에 대한 권한	계정별 객체 소유에 대한 권한
DB 관리자에 대한 권한	DB 사용자에 대한 권한

SQL – DCL

• 계정 권한 주기 예제 1
system 계정으로 로그인 후
1)select * FROM ALL_USERS ORDER BY USERNAME;
--C## 안 붙이려고 하면
alter session set "_ORACLE_SCRIPT"=true;
2)create user user1 identified by 1234
  DEFAULT TABLESPACE USERS;
--아이디 입력 시 소문자가 대문자로 자동변경
3)GRANT CREATE SESSION TO USER1;
   GRANT CREATE TABLE TO USER1;

SQL – DCL

• 계정 권한 주기 예제 2

TABLESPACE는 디스크 공간에 사용자 객체를 저장하기 위한 장소

--반드시 DEFAULT_TABLESPACE를 확인할 것

SELECT * FROM DBA_USERS WHERE USERNAME='USER1';

ALTER USER USER1 DEFAULT TABLESPACE USERSPACE;

GRANT UNLIMITED TABLESPACE TO USER1;//SYSTEM이 알아서drop user "USER1" CASCADE;

SQL – DCL

- tablespace 생성과 조회

1. 조회 : SELECT * FROM USER_TABLESPACES;
2. 생성 : create tablespace USERSPACE datafile ‘user_space’ size 2048M autoextend on next 4M maxsize unlimited logging permanent extent management local autoallocate blocksize 8K segment space management manual flashback on;

SQL – DCL

• tablespace 사용자 할당 조회 및 할당,수정,삭제

1. 조회 : SELECT * FROM DBA_USERS WHERE USERNAME = 'USER1';

2. 할당 : create user user2 identified by 1234 default tablespace USERS quota 2m on userspace;

3. 수정 : ALTER USER USER1 DEFAULT TABLESPACE USERS;

4. 공간권한: grant unlimited tablespace to USER2;

5. 삭제 : drop tablespace user1;

SQL – DCL

//권한 조회 SELECT * FROM SESSION_PRIVS;

• 권한 부여 SYSTEM으로 로그인 후

CREATE USER USER2 IDENTIFIED BY 1234;

GRANT CREATE SESSION,CREATE SYNONYM TO USER2;

//USER1로 로그인 후

GRANT SELECT ON USER1.TEMP TO USER2;

※WITH GRANT OPTION 추가 시 권한부여가능

//USER2로 로그인 후

SELECT * FROM USER1.TEMP;

CREATE SYNONYM UTEST FOR USER1.TEST;

SELECT * FROM UTEST;

SQL – DCL

- 권한 제거

//SYSTEM으로 로그인 후

```
REVOKE CREATE SESSION FROM USER2;
```

//USER1로 로그인 후

```
REVOKE SELECT ON USER1.TEMP FROM USER2;
```

//USER2로 로그인 후

SELECT * FROM USER1.TEMP; -> 조회 안됨.

SQL – DCL

```
//USER1이 모두(PUBLIC)에게 접근 허용
GRANT SELECT ON USER1.TEMP TO PUBLIC;
//SYSTEM이 모두가 사용하는 동의어 작성
CREATE PUBLIC SYNONYM USERTEMP FOR USER1.TEMP;

DROP PUBLIC SYNONYM USERTEMP;
```

SQL – DCL

- ROLE : 관련된 권한끼리 묶어 놓은 객체

1) SYSTEM ROLE(TABLESPACE까지 자동할당)

ROLE 종류	ROLE에 부여된 권한
DBA	WITH ADMIN OPTION에 있는 모든 권한
CONNECT	CREATE SESSION
RESOURCE	CREATE TRIGGER,CREATE SEQUENCE CREATE TYPE,CREATE PROCEDURE CREATE CLUSTER,CREATE OPERATOR CREATE INDEXTYPE,CREATE TABLE

SQL – DCL

2) 사용자 ROLE 생성 및 할당

```
CREATE ROLE MYROLE;
GRANT CREATE SESSION, CREATE TABLE, CREATE VIEW, CREATE SEQUENCE, CREATE SYNONYM TO MYROLE;
CREATE USER USER5 IDENTIFIED BY 1234 DEFAULT TABLESPACE USERS;
GRANT MYROLE TO USER5;
GRANT SELECT ON DB7.EMPLOYEE TO MYROLE;
```

SQL – DCL

3) ROLE 조회

```
SELECT * FROM ROLE_SYS_PRIVS WHERE ROLE LIKE 'MY%';
SELECT * FROM USER_ROLE_PRIVS;
SELECT * FROM ROLE_TAB_PRIVS WHERE OWNER LIKE'%SYS%';
DROP ROLE MYROLE;
```

• USER1의 권한명 조회

```
SELECT * FROM DBA_ROLE_PRIVS WHERE GRANTEE = 'USER1';
```

• SYSTEM ROLE인 CONNECT에 대한 세부내역 조회

```
SELECT * FROM DBA_SYS_PRIVS WHERE GRANTEE = 'CONNECT' ;
```

• 사용자 권한의 세부내역 조회

```
SELECT * FROM ROLE_SYS_PRIVS WHERE ROLE LIKE 'MY%';
```

SQL – system 계정

※ system 계정 로그인 안 될 경우