SQL,PL/SQL(ORACLE)
5.인덱스_뷰_시퀀스

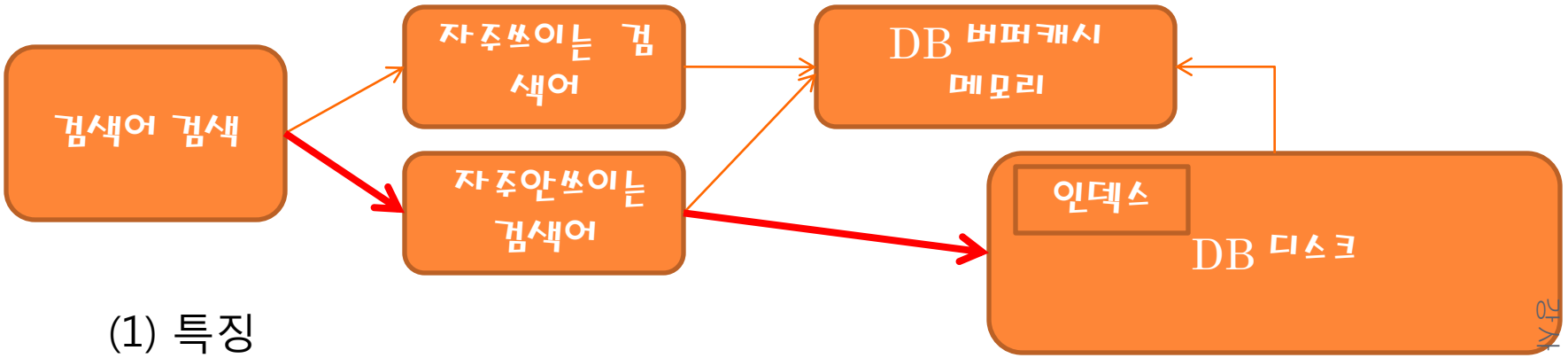
강사 양성호

1.INDEX (인덱스)

강사 양성호

1. INDEX (인덱스)

-인덱스란 포인터를 사용하여 행의 검색을 촉진시킬 수 있는 객체



(1) 특징

a. 인덱스는 테이블의 값을 빠르게 액세스 하도록 하는 DB객체
이진 검색 TREE를 이용하여 빠르게 디스크 입출력 횟수를 줄인다.

b. Oracle Server 가 인덱스를 자동적으로 사용하고 유지보수한다.
인덱스는 논리적으로도 물리적으로도 테이블과는 독립적이다.

c. 생성,삭제가 자유로우며 테이블이나 다른 인덱스에 영향을 주지 않는다.

d. 인덱스 테이블에는 INDEX KEY 와 ROWID(데이터주소,위치)에 대한 정보만을 갖고 있다.

(2) 생성

a. 자동생성 : 테이블에 생성시 PRIMARY KEY 또는 UNIQUE 제약 조건을 정의할 때 UNIQUE INDEX 자동 생성.

b. 사용자 생성
형식 : CREATE INDEX 인텍스명 ON 테이블명(컬럼명)

(3) 인덱스 생성시 주의사항

a. 생성을 많이 하는 것이 좋은 것만은 아니다
인덱스보유 테이블에 대한 DML이후 인덱스도 갱신해야하는 번거로움.

b. 생성이 필요한 경우
열이 WHERE절 또는 조인조건에서 자주 사용되는 경우
열이 광범위한 값을 포함하는 경우
열이 많은 수의 NULL 을 포함하는 경우
테이블은 대형이고 적게 읽어들일 것으로 예상되는 경우(2~4%)

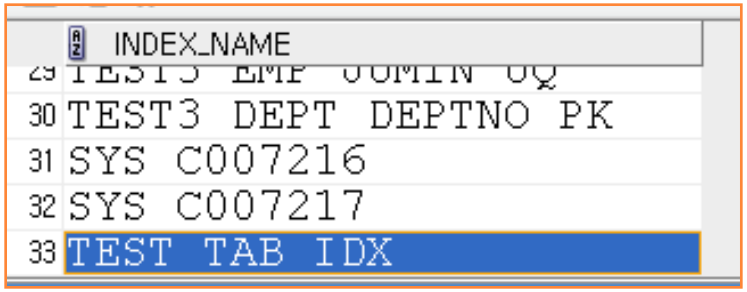
c. 생성해서 안 좋은 경우
테이블이 작다.
열이 질의의 조건으로 자주 사용되지 않는다.
대부분의 질의들이 2~4% 이상을 읽어들일것으로 예상되다.
테이블이 자주 갱신된다.(테이블에 하나 이상 인덱스를 가지고 있다면 테이블을 액세스하는데 DML문장은 인덱스의 유지 때문에 상대적으로 더 많은 시간이 걸림)

예)
```
drop table test_tab;
create table test_tab(
        num NUMBER(4) ,
        fname VARCHAR2(10),
        loc VARCHAR2(10),
        jumin CHAR(13),
        deptno NUMBER
);

drop sequence test_tab_seq;
create sequence test_tab_seq;

//여러 번 입력해주자.
insert into test_tab
values(test_tab_seq.nextval,'yangssem','서울','1234561234567',100);

select rowid,num,fname,loc,jumin,deptno from test_tab;
```

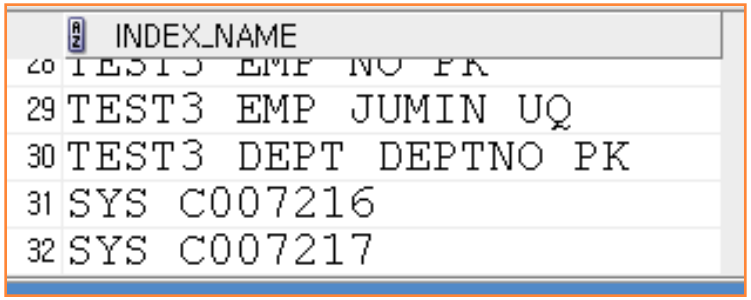
	ROWID	NUM	FNAME	LOC	JUMIN	DEPTNO
1	AAAFAKAAEAAAB9kAAA	1	yangssem	서울	1234561234567	100
2	AAAFAKAAEAAAB9kAAB	2	yangssem	서울	1234561234567	100
3	AAAFAKAAEAAAB9kAAC	3	yangssem	서울	1234561234567	100
4	AAAFAKAAEAAAB9kAAD	4	yangssem	서울	1234561234567	100
5	AAAFAKAAEAAAB9kAAE	5	yangssem	서울	1234561234567	100
6	AAAFAKAAEAAAB9kAAF	6	yangssem	서울	1234561234567	100
7	AAAFAKAAEAAAB9kAAG	7	yangssem	서울	1234561234567	100

//INDEX 생성
create index test_tab_idx on test_tab(fname);

//INDEX 검색
select index_name from user_indexes;



//INDEX 삭제
drop index test_tab_idx;



실습)다른 이름으로 인덱스를 생성하고 확인하시오.

예)
```
drop table test_tab;
create table test_tab(
        num NUMBER(4) ,
        fname VARCHAR2(10),
        loc VARCHAR2(10),
        jumin CHAR(13),
        deptno NUMBER
);
drop sequence test_tab_seq;
create sequence test_tab_seq;

begin
  for i in 1..1000 loop
    insert into test_tab
        values(test_tab_seq.nextval,
        SYS.dbms_random.string('A',9),
        '서울',
        '1234561234567',
        100);
  end loop;
  commit;
End;
```

인덱스 활용 1.
정렬시 느린
Order by 안쓰고
Index 로 정렬가능

	NUM	FNAME
1	1	ysOhCYcOM
2	2	AkTGKmQel
3	3	kwCdCkaoy
4	4	PLCLIQkdD
5	5	SXZWPkpdm
6	6	DEusxZUZA
7	7	VZgMxObnW
8	8	clOXZiLnt
9	9	VZZogGeTl
10	10	YkkDgiwxa

	NUM	FNAME
1	2	AkTGKmQel
2	6	DEusxZUZA
3	4	PLCLIQkdD
4	5	SXZWPkpdm
5	9	VZZogGeTl
6	7	VZgMxObnW
7	10	YkkDgiwxa
8	8	clOXZiLnt
9	3	kwCdCkaoy
10	1	ysOhCYcOM

```
//생성전
select num,fname from test_tab where fname>'0'; //인덱스 사용하라는 의미
//INDEX 생성
create index test_tab_idx on test_tab(fname);
```

활용1편에 이어서 설명.

```
select num, fname
from test_tab
where fname>'0'
And rownum=1;
```

인덱스 활용 2.
최대값,최소값
MAX,MIN대신
Index 로 획득가능

	NUM	FNAME
1	2	AkTGKmQel
2	6	DEusxZUZA
3	4	PLCLIQkdD
4	5	SXZWPkpdm
5	9	VZZogGeTl
6	7	VZgMxObnW
7	10	YkkDgiwxa
8	8	clOXZiLnt
9	3	kwCdCkaoy
10	1	ysOhCYcOM

//최상단 한행만 출력하면 MIN

	NUM	FNAME
1	2	AkTGKmQel

//오라클 튜닝에 사용하는 hint를 써서 index를 역스캔한다.
(단.num은 빼야함.)

```
select num,  /*+ index_desc(test_tab test_tab_idx)*/ fname
from test_tab
where fname >'0';
```

	FNAME
1	ysOhCYcOM
2	kwCdCkaoy
3	clOXZiLnt
4	YkkDgiwxa
5	VZgMxObnW
6	VZZogGeTl
7	SXZWPkpdm
8	PLCLIQkdD
9	DEusxZUZA
10	AkTGKmQel

//최상단 한행만 출력하면 MAX

```
select /*+ index_desc(test_tab test_tab_idx)*/ fname
from test_tab
where fname >'0' and rownum=1;
```

```
select /*+ index_desc(test_tab test_tab_idx)*/ max(fname)
from test_tab
where fname >'0';  //Index에 문제가 발생시 대비가능
```

	FNAME
1	ysOhCYcOM

2.VIEW (뷰)

강사 양성호

2. VIEW (뷰)

(1) 설명

VIEW 는 테이블에 대한 가상테이블.
VIEW 는 테이블이 반드시 있어야 존재함.
VIEW 는 테이블처럼 데이터를 직접 소유하지 않음.
VIEW 는 SELECT시에만 정의된 VIEW 형식에 맞춰 보여줌.

(2) 사용목적

a. 기본 테이블에 대한 '보안 기능'을 설정해야 하는 경우
예) 특정컬럼 정보는 보여주고 싶지않을 때
b. 복잡하며 자주 사용되는 질의문을 보다 쉽고
'간단'하게 사용해야 하는 경우
c. create table ~ as ~ 를 써서 복사해서 생성한 테이블의 경우
원본테이블의 데이터가 변경되면 반영되지 않는다.
view로 생성하면 사용자가 view에 접근하는 순간 원본 테이블에서 데이터를 가져 오기 때문에 데이터 정확성이 높다.

강사 양성호

(3) 특징

a. VIEW는 데이터의 물리적인 저장공간을 따로 가지지 않는다.
   -> 이유 : 테이블 안의 데이터를 참조만 하고 있기 때문

b. 테이블을 기초로 하는 가상(논리) 테이블이다.

c. 하나 이상의 테이블로 만들어진다.

d. 뷰를 access 하면 관련된 테이블도 간접적으로 access 된다.

e. 테이블에서 선택된 컬럼정보만 참조 할 수 있다.(보안)

f. 복잡한 조인(JOIN)질의를 간단, 명료하게 실행할 수 있다.

g. 미리 튜닝된 SQL문을 사용하여 성능을 향상시킬 수 있다.

h. 비지니스 로직 변경 -> DB 테이블 변경 될 때
   응용프로그램에 대한 수정이 용이하다(유지보수가 좋다.)
      -> Language Part 에서 code 수정이 필요 없다.

i. 다른 세션에서도 동일하게 적용된다.(DB에 저장된다.)

j. view를 정의하는 질의어에는 order by 절을 쓸 수 없다.

원본 사본테이블 : select * from employees;

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
1	100	Steven	King	SKING	515.123.4567	03/06/17	AD_PRES	24000	(null)	(null)	90
2	101	Neena	Kochhar	NKOCHHAR	515.123.4568	05/09/21	AD_VP	17000	(null)	100	90
3	102	Lex	De Haan	LDEHAAN	515.123.4569	01/01/13	AD_VP	17000	(null)	100	90
4	103	Alexa...	Hunold	AHUNOLD	590.423.4567	06/01/03	IT_PROG	9000	(null)	102	60
5	104	Bruce	Ernst	BERNST	590.423.4568	07/05/21	IT_PROG	6000	(null)	103	60
6	105	David	Austin	DAUSTIN	590.423.4569	05/06/25	IT_PROG	4800	(null)	103	60
7	106	Valli	Patab...	VPATABAL	590.423.4560	06/02/05	IT_PROG	4800	(null)	103	60

(4) 일반 VIEW : 단일(SIMPLE) 뷰, 하나의 테이블로 생성되는 뷰 .

- 데이터 그룹 또는 함수를 포함하지 않는다.
- view를 통해 DML 수행 가능

```
//system or /as sysdba 사용자에서 권한 확인.
//grant create view to hr;

create or replace noforce view test_view_emp
as
select employee_id, first_name, salary from employees;

select * from test_view_emp;
```

	EMPLOYEE_ID	FIRST_NAME	SALARY
1	100	Steven	24000
2	101	Neena	17000
3	102	Lex	17000
4	103	Alexander	9000
5	104	Bruce	6000
6	105	David	4800
7	106	Valli	4800
8	107	Diana	4200
9	108	Nancy	12008
10	109	Daniel	9000
11	110	John	8200
12	111	Ismael	7700

//모든 테이블과 뷰 검색
select * from tab;

	TNAME	TABTYPE	CLUSTERID
46	TEST_EMP_TAB	TABLE	(null)
47	TEST_MARGE	TABLE	(null)
48	TEST_TAB	TABLE	(null)
49	TEST_TCL	TABLE	(null)
50	TEST_VIEW_EMP	VIEW	(null)

//생성된 테이블또는 뷰의 컬럼 확인
desc test_view_emp;

```
desc test_view_emp
이름                  널       유형
--------------------- -------- ------------
EMPLOYEE_ID NOT NULL NUMBER(6)
FIRST_NAME           VARCHAR2(20)
SALARY               NUMBER(8,2)
```


//생성된 뷰 목록 검색
select view_name from user_views;

	VIEW_NAME
1	EMP_DETAILS_VIEW
2	TEST_VIEW_EMP

//뷰 삭제
drop view test_view_emp;
view TEST_VIEW_EMP이(가) 삭제되었습니다.

	VIEW_NAME
1	EMP_DETAILS_VIEW

(5) JOIN VIEW 생성

< 일반 JOIN >

// 사원이름과 부서명을 검색!

```
select e.first_name, d.department_name
from employees e join departments d
on d.department_id = e.department_id;
```

	FIRST_NAME	DEPARTMENT_NAME
1	Jennifer	Administration
2	Pat	Marketing
3	Michael	Marketing
4	Sigal	Purchasing
5	Karen	Purchasing
6	Shelli	Purchasing
7	Den	Purchasing
8	Alexander	Purchasing
9	Guy	Purchasing
10	Susan	Human Resources
11	Kevin	Shipping
12	Jean	Shipping
13	Adam	Shipping

< JOIN을 이용한 VIEW 생성 >

```
CREATE OR REPLACE VIEW test_join_view AS
select e.first_name, d.department_name
from employees e join departments d
on d.department_id = e.department_id;

select * from test_join_view;
```

똑같다.

	FIRST_NAME	DEPARTMENT_NAME
1	Jennifer	Administration
2	Pat	Marketing
3	Michael	Marketing
4	Sigal	Purchasing
5	Karen	Purchasing
6	Shelli	Purchasing
7	Den	Purchasing
8	Alexander	Purchasing
9	Guy	Purchasing
10	Susan	Human Resources
11	Kevin	Shipping
12	Jean	Shipping
13	Adam	Shipping

(6) 핸들링(handling)

1) 형식

```
CREATE OR REPLACE VIEW [FORCE|NOFORCE] view명[(별칭1, 별칭2,...,n)]
AS
[SUB-QUERY]
[WITH CHECK OPTION [CONSTRAINT 제약조건명]]
[WITH READ ONLY];
```

<2> 옵션(option) 설명

- REPLACE : 동일한 뷰이름으로 생성시 덮어쓴다.
- FORCE : 기본 테이블의 존재여부에 관계없이 뷰 생성
- NOFORCE : 기본 테이블이 존재 할 때만 생성 가능
- WITH CHECK OPTION : view에 의해 액세스 될 수 있는 행만이 입력, 갱신 될 수 있다.
- WITH READ ONLY : DML작업을 할 수 없다.

7) WITH CHECK OPTION

```
drop table test_board;
CREATE TABLE TEST_BOARD(   WNUM NUMBER(4),   WRITER VARCHAR2(10),
   TITLE VARCHAR2(50), CON VARCHAR2(10), WDATE CHAR(13), VCOUNT NUMBER);
insert into test_board values(10,'aaa','bbb','ccc','ddd',10);
insert into test_board values(20,'aaa','bbb','ccc','ddd',10);
insert into test_board values(30,'aaa','bbb','ccc','ddd',10);
select * from TEST_BOARD;

drop view test_view;
CREATE OR REPLACE VIEW test_view(WNUM,WRITER,TITLE) AS
SELECT WNUM,WRITER,TITLE
FROM TEST_BOARD
WHERE WNUM IN(20,30)
WITH CHECK OPTION;

// 20,30번데이터만 뷰로 생성하겠다는 조건절 의미는
// WITH CHECK OPTION 선언 유무에 상관없이 같으나
// 뷰에 아래와 같이 10번정보가 입력되면 원본에 못넣는다고 체크가 걸리는 것이다.
insert into test_view(wnum,writer,title) values(10,'xxx','yyy');// TEST_BOARD에 입력안됨
insert into test_view(wnum,writer,title) values(20,'xxx','yyy');
insert into test_view(wnum,writer,title) values(30,'xxx','yyy');
select * from test_view;
```

```
[아래와 같은 테이블 생성]
drop TABLE test4_dept CASCADE CONSTRAINTS PURGE;// 제약조건 무시하고 삭제
CREATE TABLE test4_dept(
   deptno NUMBER(2),
   dname VARCHAR2(15) default '개발부',
   loc_id CHAR(1),
   CONSTRAINT test4_dept_deptno_pk PRIMARY KEY (deptno),
   CONSTRAINT test4_dept_loc_ck CHECK(loc_id IN('1', '2'))
);
drop TABLE test4_emp CASCADE CONSTRAINTS PURGE;
CREATE TABLE test4_emp(
   empno NUMBER(4),
   ename VARCHAR2(10) CONSTRAINT test4_emp_ename_nn NOT NULL,
   loc_name VARCHAR2(6),
   jumin CHAR(13),
   deptno NUMBER(2),
   sal NUMBER,
   CONSTRAINT test4_emp_no_pk PRIMARY KEY (empno),
   CONSTRAINT test4_emp_jumin_uq UNIQUE (jumin),
   CONSTRAINT test4_emp_deptno_fk FOREIGN KEY (deptno) REFERENCES test4_dept(deptno)
);
INSERT INTO test4_dept  VALUES(10, '영업부', '1');
INSERT INTO test4_dept  VALUES(20, '기획부', '1');
INSERT INTO test4_dept  VALUES(30, '홍보부', '2');
INSERT INTO test4_dept  VALUES(40, '관리부', '2');
INSERT INTO test4_emp  VALUES(1001, '홍길동', '서울', '1234561234567', 10,3000);
INSERT INTO test4_emp  VALUES(1002, '최길동', '서울', '1234561234568', 10,4000);
INSERT INTO test4_emp  VALUES(1003, '박길동', '경기', '1234561234569', 20,5000);
INSERT INTO test4_emp  VALUES(1004, '양길동', '경기', '1234561234571', 30,6000);
INSERT INTO test4_emp  VALUES(1005, '한길동', '서울', '1234561234572', 40,7000);
INSERT INTO test4_emp  VALUES(1006, '강길동', '서울', '1234561234573', 40,8000);
Commit;
```

[실습1]

문1) 부서ID가 10번인 사원데이터를 갖는 이름이 test4_emp_view 인 뷰를 생성시 empno, ename 두개 컬럼만으로 생성후, 뷰검색, 뷰구조 확인하라.

```
create or replace view test4_emp_view as
select empno,ename from test4_emp where deptno=10;

select * from test4_emp_view;

desc test4_emp_view;
```

문2) 부서ID가 20번인 부서데이터를 갖는 이름이 test4_dept_view 인 뷰를 생성시 deptno, dname 두개 컬럼만으로 생성 후, 뷰검색, 뷰구조 확인하라.

```
create or replace view test4_dept_view as
select deptno,dname from test4_dept where deptno=20;

select * from test4_dept_view;

desc test4_dept_view;
```

[실습2]

문1) 부서ID가 10번인 사원데이터를 갖는 이름이 test4_emp_view 인
뷰를 생성시, 컬럼 이름을 empno를 employee_id,
ename을 employee_name으로 별칭 설정하라.

```
create or replace view test4_emp_view(employees_id, employee_name) as
select empno,ename from test4_emp where deptno=10;
```


문2) 부서ID가 20번인 부서데이터를 갖는 이름이 test4_dept_view 인
뷰를 생성시, 컬럼 이름을 deptno 를 department_id,
dname 을 department_name 으로 별칭설정하라.

```
create or replace view test4_dept_view(department_id,department_name) as
select deptno,dname from test4_dept where deptno=20;
```

[실습3]

문1) test4_emp 테이블과 test4_dept 테이블을 조인하여
empno를 사원번호로,
ename을 사원명으로,
dname을 부서명으로,
loc_name 을 지역명으로
바꾸는 test4_emp_join_dept_view 를 생성하시오.

```
create or replace view
test4_emp_join_dept_view("사원번호","사원명","부서명","지역명") as
select e.empno, e.ename, d.dname, e.loc_name
from test4_emp e,test4_dept d
where e.deptno=d.deptno;
```


[실습4]

문1) test4_emp 테이블과 test4_dept 테이블을 조인하여
dname를 부서명 으로,
min(e.sal)을 최저급여 으로,
max(e.sal)을 최고급여 으로,
avg(e.sal) 을 평균급여 으로
바꾸고 부서이름별로 test4_emp_join_dept_view 를 생성하시오.

```
create or replace view
test4_emp_join_dept_view("부서명","최저급여","최고급여","평균급여") as
select d.dname, min(e.sal), max(e.sal), avg(e.sal)
from test4_emp e, test4_dept d
where e.deptno=d.deptno
group by d.dname;
```

강사 양성호

3.SEQUENCE(시퀀스)

강사 양성호

3. SEQUENCE (시퀀스)

(1) 설명

연속적인 숫자값을 자동으로 증감시키는 숫자를
발생시키는 객체
즉, 시퀀스를 생성한 후, 호출만하면 연속적으로 번호를
(oracle에서) 증가/감소시켜 제공해 준다.

(2) 문법

CREATE SEQUENCE [시퀀스명]
[INCREMENT BY n] 0을 제외한 증가단위 28자리까지 ,양수,음수가능
ex) 2일 경우 2씩 증가,10일경우 10씩 증가한다.
[START WITH n] 시퀀스생성 시작되는값.기본값 minvalue n 과 동일.
[MAXVALUE n | NOMAXVALUE] 끝값
[MINVALUE n | NOMINVALUE] 시작 값 (끝값-시작값>increment by n)
[CYCLE | NOCYCLE] 기본NOCYCLE 최대 또는 최소값일때 그만~~^^
[CACHE | NOCACHE] 기본캐시됨.값 미리 할당됨.속도 빠름.
[ORDER | NOORDER] 기본NOORDER

```
CREATE SEQUENCE test_seq;
    DESC seq;  // 또는 DESC user_sequences;
    SELECT * FROM seq;
    SELECT * FROM user_sequences;
    SELECT sequence_name FROM seq;
    SELECT sequence_name, max_value, min_value, cycle_flag  FROM seq;
```

(3) 시퀀스 함수

주의: 시퀀스 생성 후 NEXTVAL 을 호출해야
시퀀스에 초기값이 설정된다.

```
CREATE TABLE TEST_BOARD(
   WNUM NUMBER(4),
   WRITER VARCHAR2(10) CONSTRAINT test_b_writer_nn NOT NULL,
   TITLE VARCHAR2(50) CONSTRAINT test_b_title_nn NOT NULL,
   CON VARCHAR2(4) CONSTRAINT test_b_con_nn NOT NULL,
   WDATE CHAR(13) CONSTRAINT test_b_wdate_nn NOT NULL,
   VCOUNT NUMBER,
   CONSTRAINT test_b_wnum_pk PRIMARY KEY (WNUM)
);

create sequence test_board_seq;
```

1) NEXTVAL

```
SELECT test_board_seq.nextval FROM dual;
```

2) CURRVAL

```
SELECT test_board_seq.currval FROM dual;

Insert into test_board  values( test_board_seq.nextval,'t1','t2','t3','t4',10 );
```

예1)

```
create sequence test_board_seq;
select * from seq;
```

	SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	CYCLE_FLAG	ORDER_FLAG	CACHE_SIZE	LAST_NUMBER
1	DEPARTMENTS_SEQ	1	9990	10	N	N	0	280
2	EMPLOYEES_SEQ	1	999999999999999999999999999	1	N	N	0	207
3	LOCATIONS_SEQ	1	9900	100	N	N	0	3300
4	SAMPLE1_SEQ	1	999999999999999999999999999	1	N	N	20	21
5	TEST_BOARD_SEQ	1	999999999999999999999999999	1	N	N	20	1

```
SELECT test_board_seq.currval FROM dual;
```

	NEXTVAL
1	1

	NEXTVAL
1	2

현재 값 확인

예2)

```
CREATE SEQUENCE test_board_seq2
 INCREMENT BY 1
 START WITH 1001
 MAXVALUE 1010
 MINVALUE 1001
 NOCACHE;
```

	SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	CYCLE_FLAG	ORDER_FLAG	CACHE_SIZE	LAST_NUMBER
1	DEPARTMENTS_SEQ	1	9990	10	N	N	0	280
2	EMPLOYEES_SEQ	1	999999999999999999999999999	1	N	N	0	207
3	LOCATIONS_SEQ	1	9900	100	N	N	0	3300
4	SAMPLE1_SEQ	1	999999999999999999999999999	1	N	N	20	21
5	TEST_BOARD_SEQ	1	999999999999999999999999999	1	N	N	20	41
6	TEST_BOARD_SEQ2	1001	1010	1	N	N	0	1001

	NEXTVAL
1	1001

→

	NEXTVAL
1	1002

→

	NEXTVAL
1	1010

→

ORA-08004: sequence TEST_BOARD_SEQ2.NEXTVAL exceeds MAXVALUE and cannot be instantiated
08004, 00000 - "sequence %s.NEXTVAL %s %sVALUE and cannot be instantiated"
*Cause: instantiating NEXTVAL would violate one of MAX/MINVALUE
*Action: alter the sequence so that a new value can be requested

예3)

```
drop sequence test_board_seq2;
CREATE SEQUENCE test_board_seq2
      INCREMENT BY 1
      START WITH 1001
      MAXVALUE 1010
      MINVALUE 1001
      CACHE 9
      CYCLE;  //주의 : 사이클 정의시 캐시 값은 MAX-MIN 보다 작게 설정할 것.
```

	SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	CYCLE_FLAG	ORDER_FLAG	CACHE_SIZE	LAST_NUMBER
1	DEPARTMENTS_SEQ	1	9990	10	N	N	0	280
2	EMPLOYEES_SEQ	1	9999999999999999999999999	1	N	N	0	207
3	LOCATIONS_SEQ	1	9900	100	N	N	0	3300
4	SAMPLE1_SEQ	1	9999999999999999999999999	1	N	N	20	21
5	TEST_BOARD_SEQ2	1001	1010	1	Y	N	9	1001

```
SELECT test_board_seq.currval FROM dual;
```

NEXTVAL
1001

→

NEXTVAL
1002

→

NEXTVAL
1010

→

NEXTVAL
1001

실습) test_board_seq3 이름으로 시퀀스를 생성하라.(시작값 -1,증가단위2, 최대값10,최소값-11,번호 자동 순환,캐시없음)

```
CREATE SEQUENCE test_board_seq3
   INCREMENT BY 2 START WITH -1
   MAXVALUE 10 MINVALUE -11
   NOCACHE   CYCLE;
```

예4)

```
//시퀀스 변경
ALTER SEQUENCE test_board_seq3 INCREMENT BY 2;
ALTER SEQUENCE test_board_seq3 MAXVALUE 5;

//변경불가
ALTER SEQUENCE test_board_seq3 START WITH 3;//변경불가옵션

//시퀀스 삭제
DROP SEQUENCE test_board_seq3;
```

강사 양성호

4.ROWID, ROWNUM

강사 양성호

4.ROWID , ROWNUM 컬럼

	ROWID	ROWNUM	DEPARTMENT_ID	DEPARTMENT_NAME
1	AAAEAOAAEAAAACtAAA	1	10	Administration
2	AAAEAOAAEAAAACtAAB	2	20	Marketing
3	AAAEAOAAEAAAACtAAC	3	30	Purchasing
4	AAAEAOAAEAAAACtAAD	4	40	Human Resources
5	AAAEAOAAEAAAACtAAE	5	50	Shipping
6	AAAEAOAAEAAAACtAAF	6	60	IT
7	AAAEAOAAEAAAACtAAG	7	70	Public Relations
8	AAAEAOAAEAAAACtAAH	8	80	Sales
9	AAAEAOAAEAAAACtAAI	9	90	Executive
10	AAAEAOAAEAAAACtAAJ	10	100	Finance
11	AAAEAOAAEAAAACtAAK	11	110	Accounting

1) 설명

oracle에서 테이블을 생성하면 기본적으로 제공되는 컬럼

2) 종류

ROWID

- ROW 의 고유 ID(중간에 row 수정/삭제시 불변)
- 검색접근이 빠르다.

```
update TEST_BOARD set TITLE ='오라클 SQL,PL/SQL'
where rowid in(select rowid from TEST_BOARD where WNUM =1);
```

ROWNUM

- 행의 INDEX (중간에 row 삭제시 변함)

```
SELECT ROWID, ROWNUM, department_id, department_name
        FROM departments;
SELECT count(*) FROM departments;//카운트 일반적
SELECT MAX(ROWNUM) FROM departments; //보다 효율적!!
```

- 정렬시에도 변할 수 있음

```
SELECT department_name , ROWNUM
        FROM departments ORDER BY department_id;
SELECT department_name , ROWNUM
        FROM departments ORDER BY department_id DESC;
SELECT ROWNUM , department_id, department_name
        FROM departments WHERE ROWNUM <=5;
```

4.ROWID , ROWNUM 컬럼

실습1) 사원테이블(employees)에서 가장 최근에 입사한 사원 순으로 5명을 출력하자.

select first_name,hire_date from employees order by hire_date desc; //입사정렬

```
select first_name, hire_date
from (select * from employees order by hire_date desc)
where rownum<=5;
```

```
select job_id,SUM(SALARY) "IT_PROG" from employees WHERE JOB_ID='IT_PROG'
GROUP BY ROLLUP(JOB_ID) ORDER BY JOB_ID ASC;
select job_id,SUM(SALARY) "IT_PROG" from employees WHERE JOB_ID='IT_PROG'
GROUP BY JOB_ID ORDER BY JOB_ID ASC;
select SUM(decode(job_id,'IT_PROG',salary,0)) "IT_PROG"   from employees;
```

5.ROLLUP, CUBE

강사 양성호

5.ROLLUP, CUBE

예)부서번호가 40번이하인 사원들에 대해 부서별 직급별 월급이 얼마인지 조회하라.

```
select department_id, job_id,sum(salary)
from employees
where department_id <=40
group by department_id, job_id
order by department_id;
```

	DEPARTMENT_ID	JOB_ID	SUM(SALARY)
1	10	AD_ASST	4400
2	20	MK_MAN	13000
3	20	MK_REP	6000
4	30	PU_CLERK	13900
5	30	PU_MAN	11000
6	40	HR_REP	6500

1) ROLLUP

```
select department_id, job_id,sum(salary)
from employees
where department_id <=40
group by ROLLUP(department_id, job_id)
order by department_id;
```

부서별,직급별, 총합계 표시가능.

	DEPARTMENT_ID	JOB_ID	SUM(SALARY)	
1	10	AD_ASST	4400	
2	10	(null)	4400	소계
3	20	MK_MAN	13000	
4	20	MK_REP	6000	
5	20	(null)	19000	소계
6	30	PU_CLERK	13900	
7	30	PU_MAN	11000	
8	30	(null)	24900	소계
9	40	HR_REP	6500	
10	40	(null)	6500	소계
11	(null)	(null)	54800	총합계

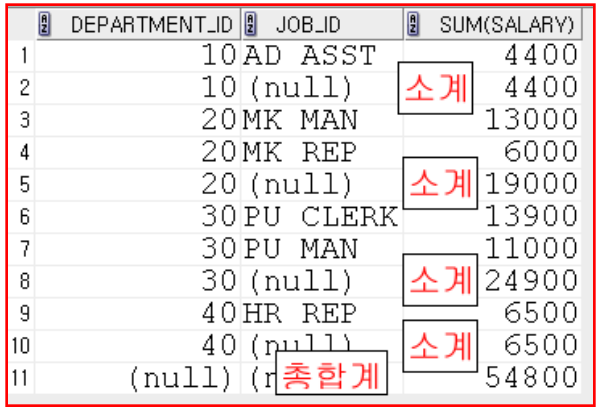
5.ROLLUP, CUBE

예)부서번호가 40번이하인 사원들에 대해 부서별 직급별 월급이 얼마인지 조회하라.

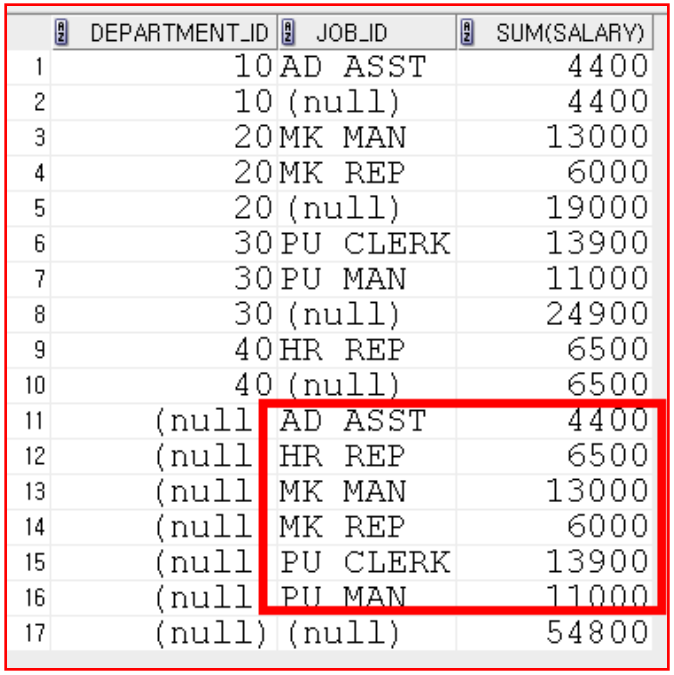
2) CUBE

```
select department_id, job_id,sum(salary)
from employees
where department_id <=40
group by CUBE(department_id, job_id)
order by department_id;
```

부서별,직급별, 총합계 표시가능.



ROLLUP



CUBE

6.SUB QUERY 응용

강사 양성호

6.SUB QUERY 응용

from절에 sub query 와 별칭 주는 with

예1)부서번호가 50번인 사원들 중, 이 부서의
평균 급여액 보다 낮은 월급을 받는 사원명단을 추출하라.

```
select e.employee_id, e.last_name, e.salary
from
(select employee_id,manager_id, salary, last_name
  from employees
  where department_id=50) e,
(select avg(salary) avg_salary
  from employees
  where department_id=50) d
where e.salary < d.avg_salary;
```

```
with e as
        (select employee_id,manager_id, salary, last_name
         from employees
         where department_id=50),
      d as
        (select avg(salary) avg_salary
         from employees
         where department_id=50)
select e.employee_id, e.last_name, e.salary
from e,d
where e.salary < d.avg_salary;
```

결과는 같다

	EMPLOYEE_ID	LAST_NAME	SALARY
1	125	Nayer	3200
2	126	Mikkilineni	2700
3	127	Landry	2400
4	128	Markle	2200
5	129	Bissot	3300
6	130	Atkinson	2800
7	131	Marlow	2500
8	132	Olson	2100
9	133	Mallin	3300
10	134	Rogers	2900
11	135	Gee	2400
12	136	Philtanker	2200
13	138	Stiles	3200
14	139	Seo	2700
15	140	Patel	2500
16	142	Davies	3100
17	143	Matos	2600
18	144	Vargas	2500
19	180	Taylor	3200
20	181	Fleaur	3100
21	182	Sullivan	2500
22	183	Geoni	2800
23	186	Dellinger	3400
24	187	Cabrio	3000
25	190	Gates	2900
26	191	Perkins	2500
27	194	McCain	3200
28	195	Jones	2800
29	196	Walsh	3100
30	197	Feeney	3000
31	198	OConnell	2600
32	199	Grant	2600

예2)전체 부서별 평균 급여액보다 부서별 급여합계액이 큰 부서의 명단을 추출하라.

```
//1.부서별 급여합계
select department_name, sum(salary) sum_sal
from employees e, departments d
where e.department_id = d.department_id
group by department_name;
```

	DEPARTMENT_NAME	부서합계
1	Administration	4400
2	Accounting	20308
3	Purchasing	24900
4	Human Resources	6500
5	IT	28800
6	Public Relations	10000
7	Executive	58000
8	Shipping	156400
9	Sales	304500
10	Finance	51608
11	Marketing	19000


```
//2.부서별 전체평균 : 전사원 급여총계 나누기 부서수
select dt.sum_amt/dc.cnt 부서평균
from (select sum(e.salary) sum_amt from employees e, departments d
        where e.department_id = d.department_id) dt,
     (select count(*) cnt from departments d
        where d.department_id IN (select department_id from employees)) dc;
```

	부서평균
1	62219.636363636363636363636363636363636364

예2)전체 부서별 평균 급여액보다 부서별 급여합계액이 큰 부서의 명단을 추출하라.

```
//3-1.최종 일반 서브쿼리
select a.department_name, a.sum_sal
from (select department_name,sum(salary) sum_sal
        from employees e, departments d
        where e.department_id = d.department_id
        group by department_name) a,
      (select dt.sum_amt/dc.cnt avg_amt
        from (select sum(e.salary) sum_amt
                from employees e, departments d
                where e.department_id = d.department_id) dt,
             (select count(*) cnt
                from departments d
                where d.department_id
                        IN (select department_id from employees)) dc) b
where a.sum_sal > b.avg_amt;
```

	DEPARTMENT_NAME	SUM_SAL
1	Sales	304500
2	Shipping	156400

예2)전체 부서별 평균 급여액보다 부서별 급여합계액이 큰 부서의 명단을 추출하라.

```
select a.department_name, a.sum_sal
from (select department_name,sum(salary) sum_sal
            from employees e, departments d
            where e.department_id = d.department_id
            group by department_name) a,
      (select dt.sum_amt/dc.cnt avg_amt
            from (select sum(e.salary) sum_amt
                        from employees e, departments d
                        where e.department_id = d.department_id) dt,
                  (select count(*) cnt
                        from departments d
                        where d.department_id
                              IN (select department_id from employees)) dc) b
where a.sum_sal > b.avg_amt;
```

```
//3-2.최종 with 서브쿼리
with dept_costs as (select department_name,sum(salary) sum_sal
                        from employees e, departments d
                        where e.department_id = d.department_id
                        group by department_name),
      avg_cost as (select sum(sum_sal)/count(*) avg from dept_costs)
select dept_costs.*
from dept_costs, avg_cost
where dept_costs.sum_sal > avg_cost.avg;
```

7.PRIVILEGE(권한) 관리

강사 양성호

권한 부여 및 수정

예1) HR사용자에게 테이블생성,세션생성 권한 할당.
GRANT CREATE TABLE, CREATE SESSION TO HR;

예2) HR사용자에게 테이블 생성 권한 해제.
REVOKE CREATE TABLE FROM HR;

예3) 권한 조회
SELECT * FROM DBA_SYS_PRIVS WHERE GRANTEE='HR';
SELECT * FROM DBA_SYS_PRIVS WHERE GRANTEE='TEST';

예4) HR과 TEST 사용자에게 부서테이블의 부서명과 지역코드컬럼을
변경 할 수 있도록 권한을 부여하라.
GRANT UPDATE(department_name, location_id) ON departments TO scott, hr;