/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

뷰 (View)

- 하나 또는 여러 테이블로 부터 데이터의 부분 집합을 논리적으로 표현하는 것.

- 실제 데이터를 가지고 있는 것이 아니라 해당 데이터의 조회 SELECT 쿼리를 가지고 있는 것.

- 뷰를 이용해 조회뿐만 아니라 데이터 수정(insert/update/delete)도 가능하다. 원본테이블의 값이 바뀜

- 테이블처럼 사용가능

- 목적

- 복잡한 select문을 간결하게 처리 가능

- 사용자의 데이터 접근을 제한 (관리자, 사원 등 각 볼 수 있는 데이터를 달리할 수 있다.)

- 뷰의 종류

- 단순뷰

- 하나의 테이블에서 데이터를 조회하며 함수를 사용하지 않는다.

- 복합뷰

- 여러 테이블에서 데이터를 조회한다.

- 함수나 group by를 이용해 조회한다.

- 뷰를 통한 변경이 안될 수 있다.

- 뷰를 이용한 DML(INSERT/DELETE/UPDATE) 작업이 안되는 경우

- 다음 항목이 포함되 있는 뷰는 insert/delete/update 할 수 없다.

- 그룹함수

- group by 절

- distinct

- rownum

- SELECT 절에 표현식이 있는 경우

- View와 연결된 행에 NOT NULL 열이 있는 경우

- 구문

CREATE [OR REPLACE] VIEW 뷰이름

AS

SELECT 문

[WITH CHECK OPTION]

[WITH READ ONLY]

- OR REPLACE

- 같은 이름의 뷰가 있을 경우 삭제하고 새로 생성한다.

- WITH CHECK OPTION

- View에서 조회될 수 있는 값만 insert또는 update할 수 있다.

- WITH READ ONLY

- 읽기 전용 View로 설정. INSERT/DELETE/UPDATE를 할 수 없다. select만 가능

View 제거

DROP VIEW VIEW이름;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

create view emp\_view

as

select \* from emp where dept\_id = 60;

select \* from emp\_view;

select \* from emp;

select \*

from (select \* from emp where dept\_id = 60); --inline view, view와의 차이는 inline view는 일회용 view는 여러번 사용가능

select e.emp\_name, d.dept\_name

from emp\_view e, dept d

where e.dept\_id = d.dept\_id;

update emp\_view

set comm\_pct = 0.5

where emp\_id = 104; --이 때 emp테이블의 104번의 comm\_pct가 바뀐다. (실제 테이블의 값이 바뀜)

-- =

update emp

set comm\_pct = 0.5

where emp\_id = 104;

create or replace view emp\_view --똑같은 이름의 view가 있을 시 덮어쓰기함(지우고 만듦)

as

select emp\_id, emp\_name, dept\_id

from emp;

insert into emp\_view values (5000, '새이름', 60); --emp 테이블의 hire\_date가 not null이기 때문에 insert 불가

create view dept\_view

as

select \* from dept where loc = 'New York';

insert into dept\_view values(300,'새부서','서울');

insert into dept\_view values(301, '뉴욕부서', 'New York');

select \* from dept\_view; --dept\_view는 loc가 New York인 것만 보기 때문에 위에서 '서울'로 넣은 값은 뷰를 조회해도 나오지 않는다.

select \* from dept; --dept 테이블에서 조회 가능

create view dept\_view2

as

select \* from dept where loc = 'New York'

with check option; --View에서 조회될 수 있는 값만 insert또는 update할 수 있다. view의 where절의 조건을 만족하는 값(행)들만 변경 가능.

insert into dept\_view2 values(303, '서울부서', '서울'); --view에서 loc를 New York으로 정했기 때문에 insert가 불가능하다.

insert into dept\_view2 values(303, '서울부서', 'New York'); --정상 실행

update dept\_view2 set dept\_name = 'aaa' where dept\_id = 10; --view에 없는 데이터므로 update가 안됨(0행 변경)

create view dept\_view3

as

select \* from dept where loc = 'New York'

with read only;

select \* from dept\_view3;

insert into dept\_view3 values(400, 'a', 'b'); --read only view이기 때문에 insert나 update, delete(DML)는 불가능하다.

delete from dept\_view3; --역시 불가능

create view emp\_name\_view

as

select emp\_name, length(emp\_name) name\_length --함수가 들어갔으므로 insert등은 불가능하다.

from emp;

select \* from emp\_name\_view;

create view emp\_view2

as

select dept\_id, max(salary) 최대급여, min(salary) 최소급여

from emp

group by dept\_id;

select \* from emp\_view2;

update emp set salary = 30000 where emp\_id =108; --emp테이블의 값이 바뀐 것이 emp\_view2에 바로 적용되어 그대로 보여짐

create view emp\_dept\_view

as

select e.emp\_id, e.emp\_name, e.salary, e.job\_id, e.hire\_date, e.comm\_pct, d.dept\_id, d.dept\_name, d.loc from emp e left join dept d on e.dept\_id = d.dept\_id;

select \* from emp\_dept\_view --join하지 않아도 join한 결과를 볼 수 있음

where loc = 'Seattle';

--TODO: 급여(salary)가 10000 이상인 직원들의 모든 컬럼들을 조회하는 View 생성

create or replace view emp\_salary

as

select \* from emp where salary>=10000 order by salary;

select \* from emp\_salary;

--TODO: 부서위치(dept.loc) 가 'Seattle'인 부서의 모든 컬럼들을 조회하는 View 생성

create view dept\_loc\_view

as

select \* from dept where loc = 'Seattle';

select \* from dept\_loc\_view;

--TODO: JOB\_ID가 'FI\_ACCOUNT', 'FI\_MGR' 인 직원들의 직원\_ID(emp.emp\_id), 직원이름(emp.emp\_name), 업무\_ID(emp.job\_id),

-- 업무명(job.job\_title), 업무최대급여(job.max\_salary), 최소급여(job.min\_salary)를 조회하는 View를 생성

create view job\_emp\_view

as

select e.emp\_id, e.emp\_name, e.job\_id, j.job\_title, j.max\_salary, j.min\_salary

from emp e left join job j on e.job\_id = j.job\_id where e.job\_id in( 'FI\_ACCOUNT','FI\_MGR' );

select \* from job\_emp\_view;

--TODO: 직원들의 정보와 직원의 급여 등급(salary\_grade.grade)을 조회하는 View를 생성

create or replace view emp\_grade\_view

as

select e.\*, s.\* from emp e join salary\_grade s on e.salary between s.low\_sal and s.high\_sal;

select \* from emp\_grade\_view;

--TODO: 직원의 id(emp.emp\_id), 이름(emp.emp\_name), 업무명(job.job\_title), 급여(emp.salary), 입사일(emp.hire\_date),

-- 상사이름(emp.emp\_name), 상사의입사일(emp.hire\_date), 소속부서이름(dept.dept\_name), 부서위치(dept.loc)를 조회하는 View를 생성

-- 상사가 없는 직원의 경우 상사이름, 상사의입사일을 null로 출력.

-- 부서가 없는 직원의 경우 '미배치'로 출력

-- 업무가 없는 직원의 경우 '없무없음' 으로 출력

create view e\_d\_view

as

select e.emp\_id, e.emp\_name, nvl(j.job\_title,'업무없음') job\_title, e.salary, e.hire\_date, m.emp\_name 상사id,

m.hire\_date 상사입사일, nvl(d.dept\_name,'미배치') dept\_name, nvl(d.loc,'미배치') loc

from emp e left join emp m on e.mgr\_id = m.emp\_id

left join job j on e.job\_id = j.job\_id

left join dept d on e.dept\_id = d.dept\_id;

select \* from e\_d\_view;

--TODO: 업무별 급여의 통계값을 조회하는 View 생성. 출력 컬럼 업무명, 급여의 합계, 평균, 최대, 최소값을 조회하는 View를 생성

create or replace view j\_s

as

select j.job\_title, sum(salary) 급여합계, round(avg(salary)) 평균급여, max(salary) 최대급여, min(salary) 최소급여

from emp e left join job j on e.job\_id = j.job\_id

group by j.job\_title;

select \* from j\_s;

--TODO: 직원수, 부서개수, 업무의 개수를 조회하는 View를 생성

create view e\_d\_j

as

select count(distinct e.emp\_id) 직원수, count(distinct d.dept\_id) 부서수, count(distinct j.job\_id) 업무수

from emp e, dept d, job j;

--union이용

create view ex

as

select '직원수' label , count(\*) cnt from emp

union all

select '부서수' , count(\*) from dept

union all

select '업무수' , count(\*) from job;

select \* from e\_d\_j;

select \* from ex;

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

시퀀스 : SEQUENCE

- 자동증가하는 숫자를 제공하는 오라클 객체

- 테이블 컬럼이 자동증가하는 고유번호를 가질때 사용한다.

- 하나의 시퀀스를 여러 테이블이 공유하면 중간이 빈 값들이 들어갈 수 있다.

생성 구문

CREATE SEQUENCE sequence이름

[INCREMENT BY n] n:숫자

[START WITH n]

[MAXVALUE n | NOMAXVALUE]

[MINVALUE n | NOMINVALUE]

[CYCLE | NOCYCLE(기본)]

[CACHE n | NOCACHE]

- INCREMENT BY n: 증가치 설정. 생략시 1

- START WITH n: 시작 값 설정. 생략시 0

- 시작값 설정시

- 증가: MINVALUE 보다 크커나 같은 값이어야 한다.

- 감소: MAXVALUE 보다 작거나 같은 값이어야 한다.

- MAXVALUE n: 시퀀스가 생성할 수 있는 최대값을 지정

- NOMAXVALUE : 시퀀스가 생성할 수 있는 최대값을 오름차순의 경우 10^27 의 값. 내림차순의 경우 -1을 자동으로 설정.

- MINVALUE n :최소 시퀀스 값을 지정

- NOMINVALUE :시퀀스가 생성하는 최소값을 오름차순의 경우 1, 내림차순의 경우 -(10^26)으로 설정

- CYCLE 또는 NOCYCLE : 최대/최소값까지 갔을때 순환할 지 여부. NOCYCLE이 기본값(순환반복하지 않는다.)

- CACHE|NOCACHE : 캐쉬 사용여부 지정.(오라클 서버가 시퀀스가 제공할 값을 미리 조회해 메모리에 저장) NOCACHE가 기본값(CACHE를 사용하지 않는다.)

시퀀스 자동증가값 조회

- sequence이름.nextval : 다음 증감치 조회

- sequence이름.currval : 현재 시퀀스값 조회

시퀀스 수정 (start with는 수정 의미가 없다)

ALTER SEQUENCE 수정할 시퀀스이름

[INCREMENT BY n]

[MAXVALUE n | NOMAXVALUE]

[MINVALUE n | NOMINVALUE]

[CYCLE | NOCYCLE(기본)]

[CACHE n | NOCACHE]

수정후 생성되는 값들이 영향을 받는다. (그래서 start with 절은 수정대상이 아니다.)

시퀀스 제거

DROP SEQUENCE sequence이름

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

-- 1부터 1씩 자동증가하는 시퀀스

create sequence ex01\_seq;

select \* from user\_sequences; --만든 시퀀스에 대한 정보 조회

select ex01\_seq.nextval from dual;

-- 1부터 50까지 10씩 자동증가 하는 시퀀스

create sequence ex02\_seq

increment by 10

maxvalue 50;

alter sequence ex02\_seq

cycle

nocache;

select ex02\_seq.nextval from dual; --1,11,21,31,41 max에 다다르면 에러발생

--cycle지정시 max에 다다르면 다시 처음으로 돌아감

-- 100 부터 150까지 10씩 자동증가하는 시퀀스

create sequence ex03\_seq

increment by 10

start with 100

maxvalue 150;

select ex03\_seq.nextval from dual; --다음값

select ex03\_seq.currval from dual; --현재값

-- 100 부터 150까지 2씩 자동증가하되 최대값에 다다르면 순환하는 시퀀스

create sequence ex04\_seq

increment by 2

start with 100

maxvalue 150

cycle; --cycle은 시퀀스가 생성하는 숫자의 개수가 cache size보다 커야한다.

--cycle시 minvalue값으로 돌아간다.(위 경우 1)

select ex04\_seq.nextval from dual;

create sequence ex05\_seq

increment by 10

start with 100

maxvalue 150

minvalue 100

cycle

cache 3;

select ex05\_seq.nextval from dual;

-- -1부터 자동 감소하는 시퀀스

create sequence ex06\_seq

increment by -1;

select ex06\_seq.nextval from dual;

select \* from user\_sequences;

-- -1부터 -50까지 -10씩 자동 감소하는 시퀀스

create sequence ex07\_seq

increment by -10

minvalue -50;

select ex07\_seq.nextval from dual;

select ex07\_seq.currval from dual;

-- -10부터 -100까지 -10씩 감소하는 시퀀스

create sequence ex08\_seq

increment by -10

start with -10

minvalue -100

cycle --감소: 순환시 maxvalue부터 시작

nocache;

alter sequence ex08\_seq

maxvalue -10; --현재 값보다 큰값이어야 한다.

select ex08\_seq.nextval from dual;

-- 100 부터 -100까지 -100씩 자동 감소하는 시퀀스

-- 감소: 시퀀스가 만드는 값은 절대 maxvalue보다 클 수 없다.

create sequence ex09\_seq

increment by -100

start with 100

minvalue -100; --에러

create sequence ex09\_seq

increment by -100

start with 100

minvalue -100

maxvalue 100;

select ex09\_seq.nextval from dual;

-- 15에서 -15까지 1씩 감소하는 시퀀스 작성

create sequence ex10\_seq

increment by -1

start with 15

minvalue -15

maxvalue 15;

select ex10\_seq.nextval from dual;

-- -10 부터 1씩 증가하는 시퀀스 작성

create sequence ex11\_seq

start with -10

minvalue -10; --start with이하 값으로 설정

select ex11\_seq.nextval from dual;

--제거

drop sequence ex10\_seq;

select \* from user\_sequences;

-- Sequence를 이용한 값 insert

create table items(

no number primary key, --1씩 자동 증가

name varchar2(100) not null

);

drop sequence items\_no\_seq;

create sequence items\_no\_seq;

insert into items values(items\_no\_seq.nextval, 'item이름'||ex01\_seq.nextval);

select \* from items;

rollback; --sequence는 rollback대상이 아니다. rollback을 하더라도 이미 증가가 된 값은 유지된다.

drop table dept\_copy;

create table dept\_copy

as

select \* from dept where 1=0;

-- TODO: 부서ID(dept.dept\_id)의 값을 자동증가 시키는 sequence를 생성. 10 부터 10씩 증가하는 sequence

-- 위에서 생성한 sequence를 사용해서 dept\_copy에 5개의 행을 insert.

drop table dept\_copy;

create table dept\_copy

as

select \* from dept where 1=0;

drop sequence dept\_id\_seq;

create sequence dept\_id\_seq

increment by 10

start with 10;

insert into dept\_copy values(dept\_id\_seq.nextval,'부서이름'||dept\_id\_seq.nextval, '지역');

select \* from dept\_copy;

-- TODO: 직원ID(emp.emp\_id)의 값을 자동증가 시키는 sequence를 생성. 10 부터 1씩 증가하는 sequence

-- 위에서 생성한 sequence를 사용해 emp\_copy에 값을 5행 insert

drop table emp\_copy;

create table emp\_copy

as

select \* from emp where 1=0;

drop sequence emp\_id\_seq;

create sequence emp\_id\_seq

start with 10;

insert into emp\_copy values(emp\_id\_seq.nextval, '이름'||emp\_id\_seq.nextval, emp\_id\_seq.nextval, emp\_id\_seq.nextval, sysdate, 10000, null, 100);

select \* from emp\_copy;