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

커서(Cursor)

- 커서는 컴퓨터에서 현재 위치를 가리키는 것을 말한다.

- 오라클에서 커서

- 임의의 SQL문이 처리된 결과를 가리키는 것으로 커서를 이용해 처리된 결과 집합의 정보를 얻을 수 있다.

- 조회결과가 한행 이상인 경우 반드시 커서를 사용해야 한다.

- select into 로는 한행의 결과만 조회가능하다.

- 커서의 진행 단계

- 커서 선언

- 커서 열기(open)

- 패치 (fetch)

- 커서 닫기 (close)

- 묵시적 커서, 명시적 커서

- 묵시적 커서

- 오라클 내부에서 자동으로 생성되어 사용되는 커서로 PL/SQL 블록에서

실행하는 문장(INSERT,UPDATE,DELETE, SELECT INTO)이 실행될 때마다 자동으로 생성됨.

- 위의 진행 단계가 자동으로 처리된다.

- 명시적 커서

- 사용자가 직접 정의해서 사용하는 커서로 select 결과를 조회할 때 사용한다.

- 위의 세가지 진행 단계를 명시적으로 처리해야 한다.

- 커서 속성

- 커서가 제공하는 결과에대한 정보들.

- 커서이름%FOUND : 결과 집합에서 조회가능한 행이 있는지 여부 반환. 1행 이상 있으면 참(True) 없으면 거짓(False) 반환

- 커서이름%NOTFOUND : %FOUND의 반대. Cursor 로 부터 마지막으로 얻은 커서의 결과 set에 레코드가 없으면 참(True)

- 커서이름%ROWCOUNT : SQL문에 영향 받은 행의 수를 반환

- 커서이름%ISOPEN : 현재 커서가 OPEN 상태이면 참(True), CLOSE 상태이면 거짓(False)

- 커서이름

- 묵시적 커서 : SQL 지정. (예: SQL%FOUND)

- 명시적 커서 : 커서이름 지정 (예: my\_cursor%FOUND)

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

SET SERVEROUTPUT ON;

-- 묵시적 커서의 예 (update 개수 출력)

begin

update emp

set emp\_name = emp\_name;

dbms\_output.put\_line(sql%rowcount);

end;

/

-- 묵시적 커서의 예(select 조회결과 행수)

declare

v\_name emp.emp\_name%type;

begin

select emp\_name

into v\_name

from emp

where emp\_id = 100;

if sql%found then --조회가능한 행이 남아있으면

dbms\_output.put\_line(sql%rowcount);

end if;

end;

/

--TODO

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

--Database에서 DML이 실행되면 그 기록을 저장할 테이블

- log\_no: 번호

- category : 어떤 구문이 실행되었는지 구분

- 'I' - insert, 'U' - update, 'D' - delete

- exec\_datetime : DML구문 실행 일시

- table\_name : 실행된 테이블

- row\_count : 구문이 적용된 행의 개수

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

drop sequence dml\_log\_no\_seq;

create sequence dml\_log\_no\_seq;

drop table dml\_log;

create table dml\_log(

log\_no number primary key,

category char not null constraint ck\_dml\_log\_category check(category in ('I', 'D', 'U')),

exec\_datetime date default sysdate not null,

table\_name varchar2(100) not null,

row\_count number not null

);

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

dml\_log\_no\_seq 테이블의 log\_no 컬럼의 값을 생성할 sequence.

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

drop table emp\_copy;

create table emp\_copy as select \* from emp;

-- TODO: 파라미터로 dept\_id를 받아서 그 부서에 소속된 직원들을 삭제하고 dml\_log에 기록을 남기는 stored procedure를 생성

--emp\_copy

create or replace procedure delete\_emp\_by\_dept\_id\_sp (p\_dept\_id dept.dept\_id%type)

is

v\_del\_cnt binary\_integer; --삭제 행수를 저장할 변수

begin

delete from emp\_copy

where dept\_id = p\_dept\_id;

v\_del\_cnt:= sql%rowcount;

insert into dml\_log values(dml\_log\_no\_seq.nextval,'D',sysdate,'emp\_copy',v\_del\_cnt);

commit;

end;

/

exec delete\_emp\_by\_dept\_id\_sp(110);

select \* from dml\_log;

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

- 명시적 커서 사용 문법

1. 커서 정의

- CURSOR 커서이름 [(파라미터이름 datatype[:=기본값], ...)] IS

select 구문;

- 파라미터

- 커서 열때 전달받을 값이 있는 경우 선언(없으면 생략)

- 변수명 데이터타입 [:= 기본값]

- 기본값이 있는 경우 호출시 값 전달을 생략할 수 있다.

2. 커서열기

- OPEN 커서이름 [(전달값1, 전달값2,..);

- 커서에 파라미터가 선언된 경우 전달할 값을 ( ) 에 지정. 없으면 () 생략

3. 커서로부터 조회 결과 데이터 읽기

- FETCH 커서이름 INTO 변수 [,..]

- 변수 : 커서가 반환하는 조회 결과값을 받을 변수. select 절의 컬럼 개수만큼의 변수를 선언한다.

4. 커서 닫기

CLOSE 커서이름;

- 커서 정의는 선언부에서 작성

- 커서 열기 ~ 닫기는 구현부에서 작성

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

-- DEPT\_ID 가 100 인 직원의 정보 조회

declare

rec\_emp emp%rowtype;

--1. 커서를 정의: 어떤 sql문의 실행 결과와 연결될 커서인지 정의.

cursor emp\_cur is

select \*

from emp

where dept\_id = 100;

begin

--2. 커서 열기(open) :1에서 정의한 sql문이 실행되고 그 실행 결과의 커서와 연결된다.

open emp\_cur;

--3. 조회결과 가져오기

fetch emp\_cur into rec\_emp; --한 행을 조회

dbms\_output.put\_line(rec\_emp.emp\_id||'-'||rec\_emp.emp\_name);

--4. 커서 닫기

close emp\_cur;

end;

/

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 파라미터가 있는 CURSOR \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

declare

rec\_emp emp%rowtype;

--1. 커서를 정의: 어떤 sql문의 실행 결과와 연결될 커서인지 정의.

cursor emp\_cur (cp\_dept\_id dept.dept\_id%type) is

select \*

from emp

where dept\_id = cp\_dept\_id;

begin

--2. 커서 열기(open) :1에서 정의한 sql문이 실행되고 그 실행 결과의 커서와 연결된다.

open emp\_cur(100);

--3. 조회결과 가져오기

fetch emp\_cur into rec\_emp; --한 행을 조회

dbms\_output.put\_line(rec\_emp.emp\_id||'-'||rec\_emp.emp\_name);

fetch emp\_cur into rec\_emp; --한 행을 조회

dbms\_output.put\_line(rec\_emp.emp\_id||'-'||rec\_emp.emp\_name);

--4. 커서 닫기

close emp\_cur;

open emp\_cur(110);

fetch emp\_cur into rec\_emp;

dbms\_output.put\_line(rec\_emp.emp\_id||'-'||rec\_emp.emp\_name);

fetch emp\_cur into rec\_emp;

dbms\_output.put\_line(rec\_emp.emp\_id||'-'||rec\_emp.emp\_name);

close emp\_cur;

end;

/

select \* from emp where dept\_id in (110, 100);

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

- LOOP 반복문을 이용해 조회한 모든 결과 조회

- cursor 의 fetch 구문을 loop 내에서 반복한다.

- 조회 가능한 행이 없으면 LOOP 를 빠져 나온다.

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

declare

cursor dept\_cur(cp\_loc dept.loc%type) is

select \*

from dept

where loc = cp\_loc;

rec\_dept dept%rowtype;

begin

open dept\_cur('New York');

--loop문을 이용해 fetch

loop

fetch dept\_cur into rec\_dept;

-- fetch dept\_cur into rec\_dept.dept\_id;

-- fetch dept\_cur into v\_id, v\_name, v\_loc;

exit when dept\_cur%notfound;

dbms\_output.put\_line(rec\_dept.dept\_id||'-'||rec\_dept.dept\_name||'-'||rec\_dept.loc);

end loop;

close dept\_cur;

end;

/

-- TODO: EMP테이블에서 커미션(comm\_pct)이 NULL이 직원의 ID(emp\_id), 이름(emp\_name) , 급여(salary), 커미션(comm\_pct)를

--이름 오름차순으로 정렬한 결과를 출력

declare

rec\_emp emp%rowtype;

cursor emp\_cur is

select emp\_id, emp\_name, salary, comm\_pct

from emp

where comm\_pct is null

order by emp\_name;

begin

open emp\_cur;

loop

fetch emp\_cur into rec\_emp.emp\_id, rec\_emp.emp\_name, rec\_emp.salary, rec\_emp.comm\_pct;

exit when emp\_cur%notfound;

dbms\_output.put\_line(rec\_emp.emp\_id||'-'||rec\_emp.emp\_name||'-'||rec\_emp.salary||'-'||rec\_emp.comm\_pct);

end loop;

close emp\_cur;

end;

/

-- TODO: DEPT 테이블에서 위치가 'New York' 인 부서들의 부서ID, 부서명 출력

declare

cursor dept\_cur(cp\_loc dept.loc%type) is

select \*

from dept

where loc = cp\_loc;

rec\_dept dept%rowtype;

begin

open dept\_cur('New York');

--loop문을 이용해 fetch

loop

fetch dept\_cur into rec\_dept;

exit when dept\_cur%notfound;

dbms\_output.put\_line(rec\_dept.dept\_id||'-'||rec\_dept.dept\_name||'-'||rec\_dept.loc);

end loop;

close dept\_cur;

end;

/

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

FOR 문을 이용한 CURSOR 결과 조회

FOR 레코드 IN 커서명 [(전달값1, 전달값2,..)]

LOOP

처리 구문

END LOOP;

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

declare

cursor dept\_cur(cp\_loc dept.loc%type) is

select \* from dept

where loc = cp\_loc;

begin

for dept\_row in dept\_cur('Seattle') -- for문에서 커서를 open

loop

dbms\_output.put\_line(dept\_row.dept\_name||'-'||dept\_row.loc);

end loop; --반복문 나오면서 자동 close

if dept\_cur%isopen then --현재 커서의 open상태 조회 - TRUE: open, FALSE: close

dbms\_output.put\_line('열려있음');

else

dbms\_output.put\_line('닫혀있음');

end if;

end;

/

-- TODO: EMP테이블에서 커미션(comm\_pct)이 NULL이 직원의 ID(emp\_id), 이름(emp\_name) , 급여(salary), 커미션(comm\_pct)를

--이름 오름차순으로 정렬한 결과를 출력

declare

cursor emp\_cur is

select emp\_id, emp\_name, salary, comm\_pct

from emp

where comm\_pct is null

order by emp\_name;

begin

for e\_row in emp\_cur

loop

dbms\_output.put\_line(e\_row.emp\_id||'-'||e\_row.emp\_name||'-'||e\_row.salary||'-'||e\_row.comm\_pct);

end loop;

end;

/

-- TODO: DEPT 테이블에서 위치가 'New York' 인 부서들의 부서ID, 부서명 출력

DROP TABLE dept\_copy;

CREATE TABLE dept\_copy AS select \* from dept;

-- TODO: EMP 테이블에서 salary가 13000 이상인 직원들이 소속된 부서(DEPT\_COPY테이블)의 loc을 '서울'으로 변경하시오.

declare

v\_dept\_id dept.dept\_id%type;

cursor dept\_id\_cur is

select dept\_id from emp

where salary >=13000;

begin

for d\_row in dept\_id\_cur

loop

-- dbms\_output.put\_line(d\_row.dept\_id);

update dept\_copy

set loc = '인천'

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

end loop;

commit;

-- open dept\_id\_cur;

-- loop

-- fetch dept\_id\_cur into v\_dept\_id;

-- exit when dept\_id\_cur%notfound;

-- update dept\_copy

-- set loc = '서울'

-- where dept\_id = v\_dept\_id;

-- end loop;

end;

/

select \* from dept\_copy;

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

FOR문에 직접 커서의 내용 정의 (단순조회)

FOR 레코드 IN (SELECT 문)

LOOP

처리구문

END LOOP;

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

begin

for d\_row in (select \* from dept)

loop

-- dbms\_output.put\_line(d\_row.dept\_id||'-'||d\_row.dept\_name||'-'||d\_row.loc);

if d\_row.loc = 'New York' then

insert into dept\_copy values d\_row;

end if;

end loop;

end;

/

select \* from dept\_copy;

-- TODO: 업무 ID (job.job\_id)와 급여 인상률(0~1)를 파라미터로 받아서 그 업무를 담당하는 직원들의 급여(emp.salary)를

-- 업무의 max\_salary(job.max\_salary) \* 인상률 만큼 인상처리하는 stored procdure를 구현.

create or replace procedure todo06\_01(p\_job\_id job.job\_id%type, p\_rate binary\_double)

is

v\_max\_salary job.max\_salary%type;--업무의 max\_salary를 저장할 변수

cursor emp\_cur is

select emp\_id

from emp

where job\_id = p\_job\_id;

begin

--max salary

select max\_salary

into v\_max\_salary

from job

where job\_id = p\_job\_id;

--salary update

for e\_row in emp\_cur

loop

update emp

set salary = salary + v\_max\_salary \* p\_rate

where emp\_id = e\_row.emp\_id;

end loop;

--commit;

end;

/

exec todo06\_01('AD\_PRES', 0.2);

select \* from emp where job\_id = 'AD\_PRES';

-- TODO: EMP테이블에서 커미션(comm\_pct)이 NULL이 직원의 ID(emp\_id), 이름(emp\_name) , 급여(salary), 커미션(comm\_pct)를 이름 오름차순으로 정렬한 결과를 출력하는 stored Procedure를 구현

-- TODO: DEPT 테이블에서 위치가 'New York' 인 부서들의 부서ID, 부서명 출력

-- TODO: 년도를 파라미터로 받아 그해 입사한 직원들의 ID, 이름, 급여, 입사일을 출력하는 Procedure 구현

create or replace procedure EX\_06\_01(p\_year varchar2)

is

begin

for e\_row in (select emp\_id, emp\_name, salary, hire\_date from emp

where to\_char(hire\_date,'yyyy') = p\_year)

loop

dbms\_output.put\_line(e\_row.emp\_id||'-'||e\_row.emp\_name||'-'||e\_row.salary||'-'||e\_row.hire\_date);

end loop;

end;

/

EXEC EX\_06\_01('2005');

EXEC EX\_06\_01('2006');

--TODO: 부서의 이름(dept.dept\_name)을 파라미터로 받아서 그 부서에 속한 직원의 ID(emp.emp\_id), 직원이름(emp.emp\_name), 급여(emup.salary)를 급여가 많은 순으로 출력하는 stored Procedure를 구현.

create or replace procedure EX\_06\_02(p\_dept\_name dept.dept\_name%type)

is

cursor emp\_dept\_cursor is

select e.emp\_id, e.emp\_name, e.salary

from emp e,dept d

where e.dept\_id = d.dept\_id(+)

and d.dept\_name = p\_dept\_name;

begin

for e\_row in emp\_dept\_cursor

loop

dbms\_output.put\_line(e\_row.emp\_id||'-'||e\_row.emp\_name||'-'||e\_row.salary);

end loop;

end;

/

EXEC EX\_06\_02('IT');

EXEC EX\_06\_02('Purchasing');

--TODO: 정수를 파라미터로 받아서 그 숫자 이상의 직원을 가지는 부서의 ID(dept.dept\_id), 부서이름(dept.dept\_name)을 출력하는 stored Procedure를 구현.

EXEC EX\_06\_03(3);

EXEC EX\_06\_03(10);

create or replace procedure EX\_06\_03(p\_num binary\_integer)

is

cursor dept\_id\_cur is

select dept\_id

from emp

group by dept\_id

having count(\*)>p\_num;

cursor dept\_cur(cp\_dept\_id dept.dept\_id%type) is

select dept\_id, dept\_name

from dept

where dept\_id = cp\_dept\_id;

begin

for d\_row in dept\_id\_cur

loop

for d\_row2 in dept\_cur(d\_row.dept\_id)

loop

dbms\_output.put\_line(d\_row2.dept\_id||'-'||d\_row2.dept\_name);

end loop;

end loop;

end;

/

select dept\_id, dept\_name

from dept

where dept\_id in( select dept\_id

from emp

group by dept\_id

having count(\*)>3);

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

커미션 내역 테이블

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

drop table commission\_trans;

create table commission\_trans(

emp\_id number(6),

commission number,

comm\_date date

);

-- TODO: EMP 테이블에서 COMM\_PCT 가 있는(not null) 직원의 커미션(salary \* comm\_pct)을 commission\_trans 테이블에 저장하는

--stored Procedure를 구현.

-- comm\_date는 실행 일시를 넣는다.

declare

--커서 정의

cursor emp\_cur is

select emp\_id, salary, comm\_pct

from emp

where comm\_pct is not null;

begin

for e\_row in emp\_cur

loop

insert into commission\_trans values (e\_row.emp\_id, e\_row.salary \* e\_row.comm\_pct, sysdate);

end loop;

commit;

end;

/

SELECT \* FROM commission\_trans;