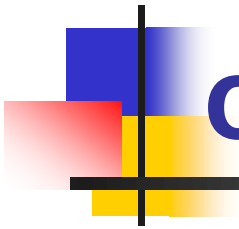
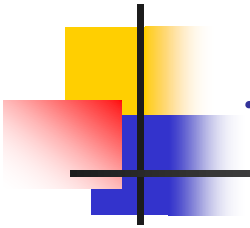


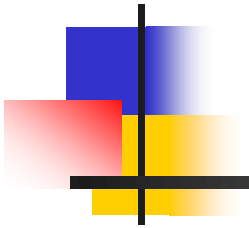
Oracle 6강 – insert, update, delete

양 명 숙
[now4ever7@gmail.com]

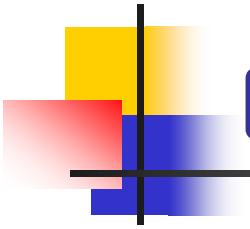


목차

- DML
  - INSERT/ UPDATE/ DELETE
  - TRANSACTION 관리하기



DML



DML

- 오라클 명령어 분류
  - DML(Data Manipulation Language)
    - select(조회), Insert(입력), update(변경), delete(삭제)
  - DDL(Data Definition Language)
    - Create(생성), alter(수정), drop(삭제), truncate(삭제)
  - DCL(Data Control Language)
    - Grant(권한주기), revoke(권한 뺏기)
  - TCL(Transaction Control Language)
    - Commit(확정), rollback(취소)

insert

- Insert
  - 테이블에 새로운 데이터를 입력할 때 사용하는 명령어
  - 데이터를 입력할 때 숫자 값 이외에는 데이터를 ‘(홑따옴표)로 감싸야 함
- [1] insert를 사용하여 단일 행 입력하기
  - 문법

```
INSERT INTO table [(column1, column2, ...)]
VALUES (value1, value2, ....)
```

  - 예1) dept2 테이블에 아래의 새로운 부서 정보를 입력하시오
  - 부서번호 : 9000, 부서명: 특판1팀
  - 상위부서 : 1006 (영업부), 지역 : 임시지역

```
INSERT INTO dept2 (dcode, dname, pdept , area)
VALUES (9000, ‘특판1팀’, '1006', ’임시지역’);
```

• 모든 칼럼에 데이터를 넣을 경우

```
INSERT INTO dept2
VALUES (9001, ‘특판2팀’, ’1006’, ’임시지역’);
```

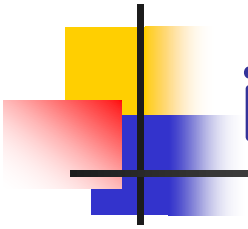
insert

- 예2) 특정 칼럼만 입력하기
  - 부서번호와 부서명, 상위부서 값만 아래의 값으로 입력하시오
  - 부서번호 : 9002, 부서명: 특판3팀
  - 상위부서 : 1006 (영업부)

```
INSERT INTO dept2 (dcode, dname, pdept)
VALUES (9002, '특판3팀', '1006');
```

- 예3) 날짜 데이터 입력하기
  - 아래 정보를 professor 테이블에 입력하시오
  - 교수번호 : 5001, 교수이름: 김설희
  - ID : kimsh, Position : 정교수
  - Pay : 510, 입사일 : 2013년 2월 19일

```
INSERT INTO professor(profno, name, id, position,pay, hiredate)
VALUES (5001, '김설희', 'kimsh', '정교수', 510, '2013-02-19');
```



insert

- 날짜형식
  - 유닉스(리눅스)용 오라클 : DD-MON-YY
    - 날짜 형식을 미리 변경한 후 '2013-02-19' 로 입력
    - 아니면 날짜 부분에 to_date()함수를 이용하여 to_date('2013-02-19', 'YYYY-MM-DD')형식으로 입력해야 함
    - 날짜 형식을 변경하는 명령어

```
Alter session set NLS_DATE_FORMAT = 'YYYY-MM-DD HH24:MI:SS'
```

  - 윈도용 오라클 : YYYY-MM-DD
- 자동으로 현재 날짜를 입력하려면 SYSDATE 사용

insert

- 예4) null 값 입력하기
  - 자동 null 값 입력하기
    - 데이터를 입력할 때 칼럼에 값을 안 주면 자동으로 null값이 들어감
  - 수동 null 값 입력하기
    - 데이터 부분에 null 값을 적어주면 됨

- [2] insert를 사용하여 여러 행 입력하기
  - 먼저 professor 테이블을 복사하여 professor2 테이블을 생성

```
Create table professor2
As select * from professor;
```

subquery의 select 문을 이용해서 하나 이상의 다른 테이블로부터 가져온 여러 데이터를 대상 테이블에 여러 건의 데이터를 입력할 수 있다

```
INSERT INTO table명(컬럼1, 컬럼2, ...컬럼N)
 select 문
```

```
Insert into professor2
select * from professor;
```

여러 행을 입력할 때 두 테이블의 칼럼의 개수와 데이터 형이 동일해야 하며,
where 조건을 사용하여 원하는 조건을 주어도 되고,
서브쿼리를 사용해도 됨

insert

- [3] 테이블을 생성하면서 데이터 insert

```
Create table 신규 테이블명
As
select  선택컬럼1, ... n  from 기존 테이블명
```

  - 신규 테이블을 만들고 동시에 다른 테이블에서 select 된 컬럼과 결과 데이터를 insert 시킴
  - select 문의 테이블과 컬럼의 제약조건은 복제되지 않기 때문에 신규 테이블에 대해 테이블과 컬럼 제약 조건을 정의해야 함
    - PK 값도 생성하지 않음

```
create table imsi_tbl
as
select e.employee_id, e.first_name || ' ' || e.last_name as "name",
    nvl2(e.commission_pct, e.salary+e.salary*e.commission_pct, e.salary) as "pay",
    d.department_id, d.department_name
from employees e left join departments d
on e.department_id=d.department_id;
```

insert

```
create table imsi_tbl2(emp_id, name, pay, deptno, dname)
as
select e.employee_id, e.first_name || ' ' || e.last_name,
    nvl2(e.commission_pct, e.salary+e.salary*e.commission_pct, e.salary),
    d.department_id, d.department_name
from employees e left join departments d
on e.department_id=d.department_id;
```

- 두번째 예제는 신규 테이블에 컬럼 리스트가 정의되면서 select 문을 통해 필요한 데이터가 insert 됨
- 첫번째 예제의 경우 , insert문에 컬럼 리스트가 없는 상태에서 select 문 컬럼 리스트에 함수가 적용됐다면 Alias를 써서 insert 되는 데이터의 컬럼명을 지정해줘야 함
  - 그렇지 않으면 에러 처리됨

update

- Update
  - 기존 데이터를 다른 데이터로 변경할 때 사용하는 방법
  - 문법

```
UPDATE  table
SET column = value
WHERE 조건;
```

  - 예1) Professor 테이블에서 직급이 조교수인 교수들의 bonus를 100만원으로 인상하시오

```
UPDATE professor
SET bonus = 100
WHERE position='조교수';
```

update

- 예2) student 테이블에서 4학년 '김재수' 학생의 2전공(deptno2)을 101로 수정하시오.

```
update student set deptno2 = 101
where name='김재수' and grade=4;
```

- 예3) Professor 테이블에서 차범철교수의 직급과 동일한 직급을 가진 교수들 중 현재 급여가 250만원이 안 되는 교수들의 급여를 15% 인상하시오.

```
UPDATE professor
SET pay = pay * 1.15
WHERE position= ( SELECT position FROM  professor
                                WHERE name = '차범철')
AND pay < 250;
```

서브쿼리를 이용한 UPDATE

- 다중건의 update
  - 서브쿼리를 사용하면 한 번의 UPDATE명령으로 여러 개의 COLUMN을 수정할 수 있음
  - 여러 컬럼을 서브쿼리의 결과로 UPDATE하면 된다.

1) EMP01 테이블의 사원번호가 7844인 사원의 부서번호와 직무(JOB)를 사원번호가 7782인 사원과 같은 직무와 같은 부서로 배정하라

```
CREATE TABLE EMP01
 AS
SELECT * FROM EMP;

UPDATE EMP01
SET (JOB, DEPTNO) = (SELECT JOB, DEPTNO
                     FROM EMP01
                     WHERE EMPNO = 7782)
WHERE EMPNO = 7844;
```

다중건의 update를 하기 위해서는 기본적인 update 문의 폼을 사용하고 subquery로 추출한 데이터를 setting 하려는 컬럼의 데이터값으로 사용함

• cf. 다중 컬럼 서브쿼리

```
select grade, name, height
from student
where (grade, height) in ( select grade, max(height)
                           from student group by grade)
order by 1;
```

Exists를 이용한 다중건의 update

- exists
  - subquery의 컬럼값이 존재하는지 여부를 체크함
  - 존재여부만 체크하기 때문에 존재하면 true, 존재하지 않으면 false를 리턴함
  - true가 리턴되면 set 컬럼의 update를 실행시키고 false가 리턴되면 update는 진행되지 않음

```
select a.*
from panmae a
where exists
(select 1 from product b
    where del_yn='Y' and a.p_code=p_code);

update panmae a
set a.p_code=
    (select p_code_new from product b
        where del_yn='Y' and a.p_code=p_code)
where exists
(select 1 from product b
    where del_yn='Y' and a.p_code=p_code);
```

DELETE

- DELETE
  - 데이터를 삭제하는 구문
  - 문법

```
DELETE FROM  table
WHERE 조건;
```

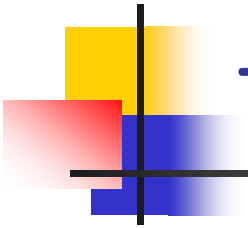
  - 예1) dept2 테이블에서 부서번호(dcode)가 9000번에서 9100번 사이인 매장들을 삭제하시오

```
DELETE FROM dept2
WHERE dcode between 9000 and 9100;
```

• Delete 문은 해당 데이터가 사용하고 있던 파일의 저장공간(extent)은 반납하지 않고 데이터만 삭제하는 구문
• 데이터가 삭제되더라도 저장공간을 반납하지 않기 때문에 용량이 줄어들지 않음

트랜잭션(Transaction) 이란?

- 은행에서 계좌 입,출금(송금)과 같은 개념
  - 은행에서는 송금 자체를 하나의 트랜잭션(거래)로 보고 A통장에서 출금한 돈이 B통장에 정확히 입금이 확인되면 그 때 거래를 성사시킴(commit)
  - 네트워크 장애로 인해 출금만 발생하고 입금이 되지 않았을 경우에는 이를 모두 취소(rollback)하게 됨
- 복불복 개념 – 모두 처리(Commit)하거나 모두 취소(Rollback)
- 반드시 트랜잭션 처리해야 함
- Savepoint를 설정하여, 설정한 savepoint까지 트랜잭션 관리 가능



Transaction 관리하기

- Transaction
  - 논리적인 작업 단위
  - 여러 가지 DML 작업들을 하나의 단위로 묶어 둔 것
  - 해당 트랜잭션 내에 있는 모든 DML이 성공해야 해당 트랜잭션이 성공하는 것이고 만약 1개의 DML이라도 실패하면 전체가 실패하게 됨
  - 모든 트랜잭션은 크기가 다를 수 있음

- TCL(Transaction Control Language)
  - commit – 트랜잭션 내의 작업의 결과를 확정하는 명령어
  - Rollback – 트랜잭션 내의 모든 명령어들을 취소하는 명령어

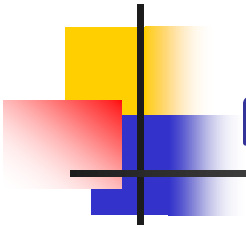
COMMIT

- 메모리 상에서 변경된 내용을 데이터 파일에 반영.
- COMMIT 을 수행하지 않고, 세션이 끊어지면 반영 안됨.

- [구문]

  COMMIT [WORK] [TO SAVEPOINT savepoint_name];

• Insert, update, delete 문을 실행하더라도 바로 데이터가 변경되지 않음
• 실제 데이터의 변경은 데이터 파일에 변경사항이 반영될 때 발생하게 됨
• 그 전에는 변경된 데이터들은 오직 오라클 메모리 상에만 존재함
• 최종적으로 데이터 파일에 적용하는 시점 => COMMIT 문장을 실행했을 때



ROLLBACK

- 메모리 상에서 변경된 내용을 데이터 파일에 반영하지 않고 종료.
  - 변경된 데이터들을 변경 전 상태로 되돌리는 역할
  - 작업한 내용을 되돌리는 '취소'의 개념

- [구문]

  ROLLBACK [WORK] [TO SAVEPOINT savepoint_name];

- COMMIT, ROLLBACK 모두 TRANSACTION 처리를 위해 존재.

TCL 명령어의 메모리 관점 동작

명령어	내용	메모리 조작
COMMIT	메모리의 내용을 하드디스크에 저장한다.	영구히 저장
ROLLBACK	메모리의 내용을 하드디스크에 저장하지 않고 버린다.	메모리 내용 무효화
CHECKPOINT	ROLLBACK 범위 설정을 위해 메모리상에 경계를 설정한다.	상태 기억

트랜잭션 조작 확인

- 연속된 SQL문으로 이루어진 트랜잭션이 현재 6번째 위치까지 실행되고 7번째 순서에서 ROLLBACK을 하려고 한다.

순서	명령문	비고
1	트랜잭션 시작	
2	INSERT	
3	SAVEPOINT A	A라는 이름으로 복구 지점 설정
4	UPDATE	
5	SAVEPOINT B	B라는 이름으로 복구 지점 설정
6	DELETE	
7	(현재 위치에서 ROLLBACK 예정)	복구 지점 사용에 따라 결과 달라짐.

트랜잭션 시나리오

트랜잭션 조작 확인

- 7번째 위치에서 ROLLBACK을 할 때 사용 명령에 따라 다음과 같이 결과가 달라진다

7번째 명령문	결과
ROLLBACK	1번째 명령까지 취소됨. 즉, 현재 위치 이전 트랜잭션 처리 내용이 모두 취소됨.
ROLLBACK TO A	1~3까지의 명령은 유효하게 남고, 4~6까지의 내용이 취소됨.
ROLLBACK TO B	1~5까지의 명령은 유효하게 남고, 6 내용이 취소됨.

실습

- dept => dept01 테이블 만들기
- emp => emp01 테이블 만들기
- --insert
- 1) dept01, emp01 테이블에 데이터 입력하기
  - dept01 테이블에는 모든 칼럼 입력, emp01 테이블에는 일부 칼럼만 입력

- --update
- 1) DEPT01 테이블의 부서번호가 30인 부서의 위치(LOC)를 '부산'으로 수정
- 2) DEPT01 테이블의 지역을 모두 '서울'로 수정
- 3) emp01 테이블에서 job이 'MANAGER' 인 사원의 급여(sal)를 10% 인상

- --서브쿼리를 이용한 update
- 1) 사원번호가 7934인 사원의 급여와, 직무를 사원번호가 7654인 사원의 직무와 급여로 수정(emp01 테이블 이용)

- --다른 테이블을 참조한 UPDATE
- 1) DEPT01 테이블에서 부서이름이 SALES인 데이터를 찾아 그 부서에 해당되는 EMP01 테이블의 사원업무(JOB)를 'SALSEMAN'으로 수정
- 2) DEPT01 테이블의 위치(loc)가 'DALLAS'인 데이터를 찾아 그 부서에 해당하는 EMP01 테이블의 사원들의 직무(JOB)을 'ANALYST'로 수정

실습

- -- DELETE
- 1) EMP01테이블에서 7782의 사원번호인 사원정보를 모두 삭제
- 2) EMP01테이블에서 직무(JOB)이 'CLERK'인 사원들의 정보를 삭제
- 3) EMP01테이블의 모든 정보를 삭제 후 rollback

- --서브쿼리를 이용한 데이터의 삭제
- 1) 'ACCOUNTING'부서명에 대한 부서코드를 DEPT01테이블에서 검색한 후 해당 부서코드를 가진 사원의 정보를 EMP01테이블에서 삭제
- 2) DEPT01테이블에서 부서의 위치가 'NEW YORK'인 부서를 찾아 EMP01테이블에서 그 부서에 해당하는 사원을 삭제

insert

```
SQL> select * from p_01;

        NO NAME
---------- ----------
      1001 조인형
      1002 박승관
      1003 송도권
```

```
SQL> select * from p_02;

        NO NAME
---------- ----------
      2001 양선희
      2002 김영조
      2003 주승재
```

- insert All을 이용한 여러 테이블에 여러 행 입력하기
  - 예1) 다른 테이블에 한꺼번에 데이터 입력하기

```
Insert  ALL  into p_01(no, name) Values(1, 'AAA')
             into p_02(no, name) Values(2, 'BBB')
SELECT * FROM DUAL;
```

p_01테이블과 p_02 테이블에 각각 서로 다른 데이터를 동시에 입력

  - 예2) 다른 테이블의 데이터를 가져와서 입력하기
    - Professor 테이블에서 교수번호가 1000번에서 1999번까지인 교수의 번호와 교수이름은 p_01 테이블에 입력하고, 교수번호가 2000번에서 2999번까지인 교수의 번호와 교수이름은 p_02 테이블에 입력하시오.

```
Insert  ALL
When profno between 1000 and 1999 then
   into p_01 Values(profno, name)
When profno between 2000 and 2999 then
   into p_02 Values(profno, name)
select  profno, name from professor;
```

insert

- 예3) 다른 테이블에 동시에 같은 데이터 입력하기
  - Professor 테이블에서 교수번호가 3000번에서 3999번까지인 교수들의 번호와 교수이름을 p_01 테이블과 p_02 테이블에 동시에 입력하시오.

```
Insert  ALL
            into p_01 Values(profno, name)
            into p_02 Values(profno, name)
select  profno, name from professor
Where profno between 3000 and 3999;
```