

제 5 장

데이터 갱신과 트랜잭션 제어



목차

5.1 데이터 갱신(Insert, Update, Delete)

5.2 TCL(Transaction Control Language)

5.1 데이터 갱신(Insert, Update, Delete)

●SQL의 데이터 조작성(DML)

- 데이터의 갱신 → UPDATE
- 데이터의 삽입 → INSERT
- 데이터의 삭제 → DELETE

●트랜잭션 제어를 위한 TCL(Transaction Control Language)

- 트랜잭션 종료 → COMMIT
- 트랜잭션 작업 철회 → ROLLBACK

5.1 데이터 갱신(Insert, Update, Delete)

●INSERT

➤ 새로운 데이터를 테이블에 삽입하는 연산

INSERT

```
INTO table-name [(col-name, [, col-name] ... )]  
VALUES (constant [, constant] ... );
```

또는 다른 테이블로부터 데이터를 선택하여 삽입한다.

INSERT

```
INTO table-name [(col-name, [, col-name] ... )]  
SELECT ... FROM ... WHERE ...;
```

INSERT

INTO **table-name** VALUES (constant [, constant] ...);

- 주의할 점 : 속성의 순서대로 데이터를 넣어주어야 합니다.

5.1 데이터 갱신(Insert, Update, Delete)

● 단일 튜플 삽입

➢ 실습을 위해 a_enrol 테이블 생성

```
SQL> create table a_enrol  
2   as select *  
3   from enrol  
4   where stu_no < 20150000;
```

5.1 데이터 갱신(Insert, Update, Delete)

- a_enrol 테이블의 구조(desc) 및 데이터 확인(select)

SQL> desc a_enrol;

이름	널	유형
SUB_NO	NOT NULL	CHAR(3)
STU_NO		NUMBER(9)
ENR_GRADE		NUMBER(3)

SQL> select * from a_enrol;

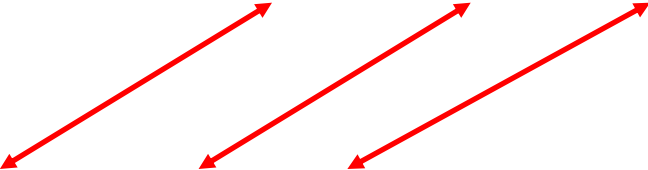
SUB_NO	STU_NO	ENR_GRADE
101	20131001	80
104	20131001	56
106	20132003	72
101	20131025	65
104	20131025	65
107	20143054	41

5.1 데이터 갱신(Insert, Update, Delete)

➤ a_enroll 테이블에 데이터 삽입

```
SQL> insert into a_enrol(sub_no, stu_no, enr_grade)
```

```
2 values ( 108, 20151062, 92 );
```



1개의 행이 만들어 졌습니다.

```
SQL> insert into a_enrol 생략가능  
2 values ( 109, 20152088, 85 );
```

1개의 행이 만들어 졌습니다.

5.1 데이터 갱신(Insert, Update, Delete)

- 일부 컬럼만 값이 존재하는 경우 생략 불가능

```
SQL> insert into a_enrol(sub_no, stu_no)
2 values ( 110, 20152088 );
```

1개의 행이 만들어 졌습니다.

- 테이블의 데이터 확인

```
SQL> select * from a_enrol;
```

SUB_NO	STU_NO	ENR_GRADE
101	20131001	80
104	20131001	56
106	20132003	72
101	20131025	65
104	20131025	65
107	20143054	41
108	20151062	92
109	20152088	85
110	20152088	

5.1 데이터 갱신(Insert, Update, Delete)

➤ NULL값의 명시적 표현

SQL> insert into a_enrol values(111, 20153075, null);
1개의 행이 만들어 졌습니다.

➤ 테이블의 데이터 확인

SQL> select * from a_enrol;

SUB_NO	STU_NO	ENR_GRADE
101	20131001	80
104	20131001	56
106	20132003	72
101	20131025	65
104	20131025	65
107	20143054	41
108	20151062	92
109	20152088	85
110	20152088	
111	20153075	

5.1 데이터 갱신(Insert, Update, Delete)

➤ 복수 행 삽입

- 부질의의 결과를 테이블에 입력
- 부질의 결과의 열의 위치 및 수가 테이블과 일치하여야 함

```
SQL> insert into a_enrol  
2  select * from enrol  
3  where stu_no like '2015%';
```

6개의 행이 만들어 졌습니다.

```
SQL> select * from a_enrol
```

SUB_NO	STU_NO	ENR_GRADE
101	20131001	80
104	20131001	56
106	20132003	72
101	20131025	65
104	20131025	65
107	20143054	41
108	20151062	92
109	20152088	85
110	20152088	
111	20153075	
103	20152088	45
108	20151062	81
102	20153075	66
105	20153075	56
102	20153088	61
105	20153088	78

5.1 데이터 갱신(Insert, Update, Delete)

●UPDATE

- 데이터 값을 변경함
 - 조건절을 만족하는 테이블내의 모든 데이터 변경
 - 조건절이 없는 경우 테이블의 모든 데이터 변경
 - WHERE절에 부질의 가능

UPDATE table-name

SET col-name = expression, [col-name = expression]

[WHERE predicate]

5.1 데이터 갱신(Insert, Update, Delete)

➤ 전체 데이터에 대한 변경

```
SQL> select * from a_enrol;
```

```
SQL> update a_enrol  
2   set enr_grade = enr_grade + 5;
```

16행이 갱신되었습니다.

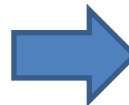
```
SQL> select * from a_enrol;
```

5.1 데이터 갱신(Insert, Update, Delete)

➤ 갱신 전후에 a_enrol 테이블의 데이터 확인

SQL> select * from a_enrol;

SUB_NO	STU_NO	ENR_GRADE
101	20131001	80
104	20131001	56
106	20132003	72
101	20131025	65
104	20131025	65
107	20143054	41
108	20151062	92
109	20152088	85
110	20152088	
111	20153075	
103	20152088	45
108	20151062	81
102	20153075	66
105	20153075	56
102	20153088	61
105	20153088	78



SUB_NO	STU_NO	ENR_GRADE
101	20131001	85
104	20131001	61
106	20132003	77
101	20131025	70
104	20131025	70
107	20143054	46
108	20151062	97
109	20152088	90
110	20152088	
111	20153075	
103	20152088	50
108	20151062	86
102	20153075	71
105	20153075	61
102	20153088	66
105	20153088	83

5.1 데이터 갱신(Insert, Update, Delete)

➤ 조건에 맞는 데이터 변경

```
SQL> select * from a_enrol;
```

```
SQL> update a_enrol  
2   set enr_grade = enr_grade + 10  
3   where sub_no = 104 ;
```

2행이 갱신되었습니다.

```
SQL> select * from a_enrol;
```

5.1 데이터 갱신(Insert, Update, Delete)

➤ 부질의를 갖는 UPDATE문

```
SQL> update a_enrol  
2   set enr_grade = enr_grade + 10  
3   where sub_no = (select sub_no  
4                   from subject  
5                   where sub_name = '시스템 분석설계');
```

2행이 갱신되었습니다.

```
SQL> select * from a_enrol;
```


5.1 데이터 갱신(Insert, Update, Delete)

●DELETE

➤데이터를 삭제하는 연산

- 조건절을 만족하는 테이블 내의 모든 데이터 삭제
- 조건절이 없는 경우 테이블의 모든 데이터 삭제
- WHERE절에 부질의 가능

DELETE

FROM table-name

[WHERE predicate]

5.1 데이터 갱신(Insert, Update, Delete)

➤ 특정 튜플 삭제

```
SQL> delete from a_enrol  
2  where stu_no = 20131001;
```

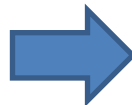
2행이 삭제되었습니다.

5.1 데이터 갱신(Insert, Update, Delete)

➤ 삭제 전후에 a_enrol 테이블의 데이터 확인

SQL> select * from a_enrol;

SUB_NO	STU_NO	ENR_GRADE
101	20131001	80
104	20131001	56
106	20132003	72
101	20131025	65
104	20131025	65
107	20143054	41
108	20151062	92
109	20152088	85
110	20152088	
111	20153075	
103	20152088	45
108	20151062	81
102	20153075	66
105	20153075	56
102	20153088	61
105	20153088	78



SUB_NO	STU_NO	ENR_GRADE
106	20132003	77
101	20131025	70
104	20131025	70
107	20143054	46
108	20151062	97
109	20152088	90
110	20152088	
111	20153075	
103	20152088	50
108	20151062	86
102	20153075	71
105	20153075	61
102	20153088	66
105	20153088	83

5.1 데이터 갱신(Insert, Update, Delete)

➤ 테이블내의 모든 데이터가 삭제

SQL> delete from a_enrol

14행이 삭제되었습니다.

-delete는 컬럼의 구조나 남기지만, 데이터는 삭제 됩니다.

-drop은 컬럼과 데이터를 모두 삭제 합니다.

❖ Drop table a_enrol;

5.1 데이터 갱신(Insert, Update, Delete)

●MERGE

➤데이터 이동 작성시 사용하는 연산

```
MERGE INTO table-name
  USING table-name | view-name | subquery
  ON join condition
  WHEN MATCHED THEN
    UPDATE SET
      field = expression, [col-name = expression]
  WHEN NOT MATCHED THEN
    INSERT [(col-name [, col-name] ... )]
    VALUES (constant [, constant] ... );
```

5.1 데이터 갱신(Insert, Update, Delete)

➤ 학생(student) 테이블의 내용을 a_student로 수정 또는 삽입하는 MERGE 연산

```
SQL> merge into a_student a
      2 using student s
      3 on (s.stu_no = a.stu_no)
      4 when matched then
      5     update set a.stu_weight = s.stu_weight
      6 when not matched then
      7     insert values(s.stu_no, s.stu_name, s.stu_dept, null,
      8                  null, null, null, s.stu_weight);
```

- a_student데이터 중 학생테이블에 데이터가 있는 학생은 체중을 변경하고, 없으면 학생테이블의 데이터를 a_student에 삽입합니다.

```
SQL> select * from a_student
```

--원래 student 테이블에 속성을 줄 당시에 학년, 반, 성별, 키 값에 대하여
-- not null을 지정한 관계로 위의 문장의 실행이 안됩니다.
--student 테이블의 null 처리에 따라서 결과가 다르게 나올 수 있습니다.
--만약에 실행하려면 학년, 반, 성별, 키, 몸무게가 null로 되어 있어야 가능합니다.

5.2 TCL(Transaction Control Language)

●트랜잭션

- 트랜잭션이란 사용자에게 의해 실행 된 SQL문의 집합
- 변경된 데이터는 TCL에 의해 데이터베이스에 반영됨
- 트랜잭션 처리는 데이터 무결성(Integrity) 유지
- DML문의 한 번 이상 실행이 하나의 트랜잭션이 되며,
- DDL문은 하나의 명령이 하나의 트랜잭션이 된다.

5.2 TCL(Transaction Control Language)

➤ 실습에 앞서 a_student, b_student 테이블 생성하고, 데이터 확인

```
SQL> create table a_student
```

```
as
```

```
2  select * from student
```

```
3  where stu_dept in ('기계', '전기전자');
```

```
SQL> select * from a_student;
```

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20132003	박희철	전기전자	3	B	M		63
20142021	심수정	전기전자	2	A	F	168	45
20152088	조민우	전기전자	1	C	M	188	90
20143054	유가인	기계	2	C	F	154	47
20153088	이태연	기계	1	C	F	162	50
20153075	옥한빛	기계	1	C	M	177	80

5.2 TCL(Transaction Control Language)

➤ 실습에 앞서 b_student 테이블 생성하고, 데이터 확인

```
SQL> create table b_student  
2   as  
   select * from student  
3   where stu_dept in ('전기전자', '컴퓨터정보');
```

```
SQL> select * from b_student;
```

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20152088	조민우	전기전자	1	C	M	188	90
20142021	심수정	전기전자	2	A	F	168	45
20132003	박희철	전기전자	3	B	M		63
20151062	김인중	컴퓨터정보	1	B	M	166	67
20141007	진현무	컴퓨터정보	2	A	M	174	64
20131001	김종현	컴퓨터정보	3	C	M		72
20131025	육성우	컴퓨터정보	3	A	F	172	63

5.2 TCL(Transaction Control Language)

➤ b_student의 내용을 삭제

```
SQL> delete from b_student;
```

7행이 삭제되었습니다.

```
SQL> select * from b_student;
```

선택된 레코드가 없습니다.

```
SQL> rollback;
```

롤백이 완료되었습니다.

```
SQL> select * from b_student;
```

--데이터를 삭제후에 commit을 실행하면 rollback을 실행하여도 데이터는 존재하지 않습니다. 그러나, commit을 하지 않고 rollback을 하면 데이터를 살릴 수 있습니다.(15분 이내에)

5.2 TCL(Transaction Control Language)

●b_student 테이블의 내용을 검색

SQL> select * from b_student;

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20152088	조민우	전기전자	1	C	M	188	90
20142021	심수정	전기전자	2	A	F	168	45
20132003	박희철	전기전자	3	B	M		63
20151062	김인중	컴퓨터정보	1	B	M	166	67
20141007	진현무	컴퓨터정보	2	A	M	174	64
20131001	김종현	컴퓨터정보	3	C	M		72
20131025	육성우	컴퓨터정보	3	A	F	172	63

5.2 TCL(Transaction Control Language)

➤ DDL명령을 통한 자동 COMMIT

SQL> delete from b_student;

7행이 삭제되었습니다.

➤ 테이블을 생성하는 DDL명령어를 실행

SQL> create table c_student

2 (stu_no number,

3 stu_name char(10));

 자동commit 역할

테이블이 생성되었습니다.

SQL> rollback;

SQL> select * from b_student;

선택된 레코드가 없습니다.

5.2 TCL(Transaction Control Language)

- 비정상적인 작업의 종료
- a_student 테이블의 내용을 검색

SQL> select * from a_student;

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20132003	박희철	전기전자	3	B	M		63
20142021	심수정	전기전자	2	A	F	168	45
20152088	조민우	전기전자	1	C	M	188	90
20143054	유가인	기계	2	C	F	154	47
20153088	이태연	기계	1	C	F	162	50
20153075	옥한빛	기계	1	C	M	177	80

5.2 TCL(Transaction Control Language)

- a_student 테이블의 모든 데이터를 삭제

SQL> delete from a_student;

6행이 삭제되었습니다.

- a_student 테이블의 데이터가 삭제된 것을 검색

SQL> select * from a_student;

선택된 레코드가 없습니다.

5.2 TCL(Transaction Control Language)

➤ 작업관리자를 이용하여 SQL Plus 프로세스(developer)를 비정상적으로 종료

SQL> select * from a_student;

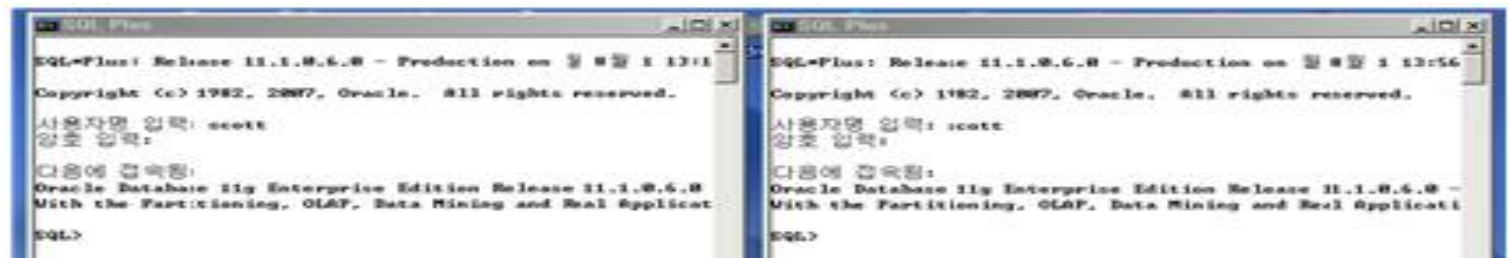
STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20132003	박희철	전기전자	3	B	M		63
20142021	심수정	전기전자	2	A	F	168	45
20152088	조민우	전기전자	1	C	M	188	90
20143054	유가인	기계	2	C	F	154	47
20153088	이태연	기계	1	C	F	162	50
20153075	옥한빛	기계	1	C	M	177	80

.비 정상종료시에는 트랜잭션이 자동으로 rollback되어서 데이터가 살아있음.

5.2 TCL(Transaction Control Language)

● 병행처리

➢ 실습을 위해 SQLPlus를 두 번 실행함.



```
Run SQL Command Line
SQL*Plus: Release 11.2.0.2.0 Production on 수 6월 28 09:35:25 2023
Copyright (c) 1982, 2014, Oracle. All rights reserved.

SQL> conn hyun;
Enter password:
Connected.
SQL> select * from tab;
```


5.2 TCL(Transaction Control Language)

- 각각의 화면에 다음과 같이 시간순서를 고려하여 입력하고 실행

시간	프로그램1	프로그램2
①	SQL> select * from a_sudent;	
②	SQL> insert 2 into a_student(stu_no, stu_name) 3 values(10,'홍');	
③	SQL> select * 2 from a_student;	
④		SQL> select * 2 from a_student; ❌
⑤	SQL> commit;	
⑥		SQL> select * 2 from a_student; 😊

insert into a_student values(20230628, '홍길동', '인공지능', 1, 'B', 'M',
195.23, 98.12);

5.2 TCL(Transaction Control Language)

●데이터 로킹(Lock)

- 병행처리의 문제점을 해결하기 위한 방법
- a_student 테이블에 기본키 설정

SQL> alter table a_student

2 add constraints pk_a_student primary key(stu_no);

5.2 TCL(Transaction Control Language)

●로킹(Locking)

시간	프로그램1	프로그램2
①	SQL> select * 2 from a_student; SQL> insert 2 into a_student(stu_no, stu_name) 3 values(20,'김'); SQL> select * 2 from a_student;	
②		SQL> select * 2 from a_student; SQL> insert 2 into a_student(stu_no, stu_name) 3 values(30,'0');
③	SQL> COMMIT or ROLLBACK;	
④		SQL>

5.2 TCL(Transaction Control Language)

●행 단위 로킹

시간	프로그램1	프로그램2
①	SQL> select * 2 from a_student; SQL> update a_student 2 set stu_class = 'A' 3 where stu_no = 20153088 SQL> select * 2 from a_student;	
②		SQL> select * 2 from a_student; SQL> delete a_student 2 where stu_no = 20153088;
③	SQL> commit or rollback;	
④		SQL>

5.2 TCL(Transaction Control Language)

●검색에 의한 로킹

시간	프로그램1	프로그램2
①	SQL> select * 2 from a_student 3 where stu_no = 20153088 4 for update;	
②		SQL> update a_student 2 set stu_weihht = 165 3 where stu_no = 20153088;
③	SQL> commit;	
④		SQL>

5.2 TCL(Transaction Control Language)

●테이블 로킹

시간	프로그램1	프로그램2
①	SQL> update a_student 2 set stu_weight = stu_weight - 5;	
②		SQL> update a_student 2 set stu_grade = 2 3 where stu_no = 20153075
③	SQL> commit;	
④		SQL>