

* MERGE

- 한번에 조건에 따라 INSERT, UPDATE 가 가능
- 해당 ROW가 있으면 UPDATE, 없으면 INSERT 문장이 실행
- syntax

MERGE INTO target_table_name

USING (table|view|subquery) ON (join condition)

WHEN MATCHED THEN

UPDATE SET col1 = val1[, col2 = val2...]

WHEN NOT MATCHED THEN

INSERT(...) VALUES(...)

-- scott유저로 접속, 테스트를 위한 테이블 생성

SQL>CREATE TABLE emp_test AS SELECT * FROM emp WHERE deptno = 10;

SQL>SELECT empno, ename, sal FROM emp_test; -- 간단하게 emp_test테이블에 데이터가 있으면 급여를 10%인상하고 없으면 새로 INSERT하는 예제.

SQL>MERGE INTO emp_test et

USING emp e

ON(et.empno = e.empno)

WHEN MATCHED THEN

UPDATE SET et.sal = e.sal*1.1

WHEN NOT MATCHED THEN

INSERT VALUES

(e.empno, e.ename, e.job, e.mgr, e.hiredate, e.sal, e.comm, e.deptno)

SQL>SELECT empno, ename, sal FROM emp_test

* 데이터베이스 TRANSACTION

◆ 트랜잭션은 데이터 처리의 한 단위 이다.

◆ 오라클 서버에서 발생하는 SQL문들을 하나의 논리적인 작업단위로써 성공하거나 실패하는 일련의 SQL문을 트랜잭션이라 할수 있다.

◆ ORACLE SERVER는 TRANSACTION을 근거로 데이터의 일관성을 보증.

◆ TRANSACTION은 데이터를 일관되게 변경하는 DML문장으로 구성 (COMMIT, ROLLBACK, SAVEPOINT)

① TRANSACTION의 시작

- 실행 가능한 SQL문장이 제일 처음 실행될 때

② TRANSACTION의 종료

- COMMIT이나 ROLLBACK

- DDL이나 DCL문장의 실행(자동 COMMIT)

- 기계 장애 또는 시스템 충돌(crash)

- deadlock 발생

- 사용자가 정상 종료

③ 자동 COMMIT은 다음의 경우 발생 합니다.

- DDL, DCL문장이 완료 될때

- 명시적인 COMMIT이나 ROLLBACK없이 SQL*Plus를 정상 종료 했을 경우

④ 자동 ROLLBACK은 다음의 경우 발생 합니다.

- SQL*Plus를 비정상 종료 했을 경우

- 비정상적인 종료, system failure

* SAVEPOINT 와 ROLLBACK TO

- SAVEPOINT는 사용자가 트랜잭션의 작업을 여러개의 세그먼트로 분할할 수 있도록 하는 특별한 작업이다.

- SAVEPOINT는 부분적인 롤백을 가능하게 하기 위해 트랜잭션에 대한 중간점을 정의 한다.

SQL>INSERT INTO emp(empno, ename, hiredate) VALUES(10000, 'test2', sysdate);

SQL>SAVEPOINT A;

저장점이 생성되었습니다. (여기서 SAVEPOINT를 생성)

SQL>INSERT INTO emp(empno, ename, hiredate) VALUES(10001, 'test3', sysdate);

SQL>INSERT INTO emp(empno, ename, hiredate) VALUES(10002, 'test4', sysdate);

SQL>DELETE FROM emp WHERE empno IN(10000, 10001, 10002);

SQL>SELECT empno, ename FROM emp WHERE empno IN(10000, 10001, 10002);

선택된 행이 없습니다.

SQL>ROLLBACK TO A;

롤백이 완료되었습니다. (SAVEPOINT까지만 롤백이 시행됩니다.)

SQL>SELECT empno , ename FROM emp WHERE empno IN(10000, 10001, 10002);

* COMMIT과 ROLLBACK

COMMIT : 변경사항 저장

ROLLBACK : 변경사항 취소

① COMMIT과 ROLLBACK의 장점

- 데이터의 일관성을 제공.

- 데이터를 영구적으로 변경하기 전에 데이터 변경을 확인하게 한다.

- 관련된 작업을 논리적으로 그룹화 할 수 있다.

- COMMIT, SAVEPOINT, ROLLBACK 문장으로 TRANSACTION의 논리를 제어 할 수 있다.

② COMMIT이나 ROLLBACK 이전의 데이터 상태

- 데이터 이전의 상태로 복구가 가능.

- 현재 사용자는 SELECT문장으로 DML작업의 결과를 확인할 수 있다.

- 다른 사용자는 SELECT문장으로 현재 사용자 사용한 DML문장의 결과를 확인할 수 없다.

- 변경된 행은 LOCK이 설정되어서 다른 사용자가 변경할 수 없다.

③ COMMIT이후의 데이터 상태

- 데이터베이스에 데이터를 영구적으로 변경

- 데이터의 이전 상태는 완전히 상실

- 모든 사용자가 결과를 볼 수 있다.

- 변경된 행의 LOCK이 해제되고 다른 사용자가 변경할 수 있습니다.

- 모든 SAVEPOINT는 제거 됩니다.

☞ 데이터베이스 링크(Database Link)

데이터베이스 링크는 클라이언트 또는 현재의 데이터베이스에서 네트워크상의 다른 데이터베이스에 접속하기 위한 접속 설정을 정의하는 오라클 객체.

◆ 우선 고려되어야 사항은 **ORACLE INSTANCE**가 두개이상이고 각각의 **HOST NAME**과 **ORACLE_SID**는

다르고 **NLS_CHARACTER_SET**은 동일하게 되어 있어야 한다
- 만약 같은 **MECHINE**에서 **INSTANCE**의 **ORACLE_SID**가 같다면 **TNS ERROR**가 발생.

- 또한 미래를 위해 다른 **MECHINE**이라 할지라도 **ORACLE_SID**는 규칙에 의해 다르게 가져가는 것이 좋다

- 그리고 **NLS_CHARACTER_SET**이 동일하게 되어 있지 않으면 **DATA** 입출력시 **?????**로 나타납니다.

- 데이터베이스 링크로 연결되는 서버에 리스너가 꼭 띄어져 있어야 됩니다

[Syntax]

```
CREATE [PUBLIC] DATABASE LINK link_name  
CONNECT TO username IDENTIFIED BY password  
USING SERVICE_NAME
```

- **PUBLIC** : 오라클 시노님과 마찬가지로 **PUBLIC** 옵션을 사용하면 공용 데이터베이스 링크를 생성 할 수 있다. 다른 사용자도 함께 사용할 수 있다는 얘기다. **PUBLIC** 옵션을 사용하지 않으면 링크를 생성한 자신만 사용 할 수 있게된다.

[Syntax]

- **link_name** : 데이터베이스 링크의 이름을 지정.

- **service_name** : 네트워크 접속에 사용할 오라클 데이터베이스 네트워크 서비스명을 지정 합니다.

- **username, password** : 오라클 사용자명과 비밀번호를 지정

☞ 데이터베이스 링크의 사용

-- 데이터베이스 링크 생성 예제

```
SQL>CREATE DATABASE LINK test_server
```

```
CONNECT TO scott IDENTIFIED BY tiger USING 'testdb';
```

이 데이터베이스 링크 생성 문장에서 **USING**다음에 오는 **testdb**는 **tnsnames.ora**파일에 정의되어 있어야 한다.

```
===== tnsnames.ora =====
```

```
testdb =
```

```
(DESCRIPTION =  
  (ADDRESS_LIST =  
    (ADDRESS = (PROTOCOL = TCP)(HOST = 해당아이  
Ⅱ))(PORT = 1521))  
  )  
  (CONNECT_DATA =  
    (SERVICE_NAME = ora9i)  
  )  
)
```

데이터베이스 링크의 삭제..

```
SQL>DROP DATABASE LINK test_server;
```

```
=====
```

-- 데이터베이스 링크를 통한 데이터의 조회..

```
SQL>SELECT ename FROM emp@test_server;
```

--시노님을 생성해서 사용하면 더욱더 편리하게 사용 할 수 있다.

```
SQL> CREATE SYNONYM emplink FOR emp@test_server;
```

-- 시노님을 통한 조회

```
SQL>SELECT ename FROM emplink;
```