

1 REM Author : Henry  
2 REM Date : 2024.06.12  
3 REM Objective :  
4 REM Environment : Ubuntu Server 22.04 LTS, MySQL Workbench 8.0 CE, MySQL Community  
Server 8.0.37-0ubuntu0.22.04.3 (ubuntu)

5

6 REM Transaction

7

## 8 1. Transaction 이란?

- 9 1)한 개 이상의 실행될 SQL 문장의 집합
- 10 2)Database의 논리적 연산단위.
- 11 3)밀접히 관련되어 분리될 수 없는 한 개 이상의 Database 조작.
- 12 4)하나의 Transaction에는 하나 이상의 SQL 문장이 포함된다.
- 13 5)분할할 수 없는 최소의 단위.
- 14 6)데이터의 일관성을 보장
- 15 7)행의 LOCK 처리를 기본으로 함
- 16 8)데이터의 변경시 유효성을 제공하고, 사용자의 프로세스가 예기치 않게 중단되거나, 시스템  
장애가 발생하여 데이터의 일관성이 어렵게 된 경우에도 일관성을 보장하기 위한 시스템적  
응용
- 17 9)전부 적용하거나 전부 취소한다.
- 18 10)즉 ALL OR NOTHING의 개념.
- 19 11)Oracle에서는 Transaction의 대상이 되는 SQL 문장을 실행하면 자동으로 시작되고,  
COMMIT또는 ROLLBACK을 실행한 시점에서 종료된다.

20

21

## 22 2. Transaction의 특성

- 23 1)원자성(Atomicity) : Transaction에서 정의된 연산들은 모두 성공적으로 실행되었는지 아니면  
전혀 실행되지 않은 상태로 남아 있어야 한다.(all or nothing)
- 24 2)일관성(Conststency) : Transaction이 실행되기 전의 Database 내용이 잘못 되어 있지  
않다면 Transaction이 실행된 이후에도 Database의 내용에 잘못이 없으면 안 된다.
- 25 3)고립성(Isolation) : Transaction이 실행되는 도중에 다른 Transaction의 영향을 받아  
잘못된 결과를 만들어서는 안된다.
- 26 4)지속성(Durability) : Transaction이 성공적으로 수행되면 그 Transaction이 갱신한  
Database의 내용은 영구적으로 저장된다.

27

28

## 29 3.TCL(Tracnsaction Control Language)

- 30 1)COMMIT : 보류중인 데이터를 영구적인 데이터베이스로 변경사항을 저장하고 현재의  
TRANSACTION 을 종료한다.
- 31 2)ROLLBACK [TO SAVEPOINT name]: 보류중인 데이터의 변경사항을 모두 되돌리고 현재의  
트랜잭션을 종료한다. 만일 특정지점을 지정하지 않으면 모든 트랜잭션 취소한다.
- 32 3)SAVEPOINT name :현재의 트랜잭션의 저장점을 표시한다.

33

34

## 35 4.Transaction 의 범위

36 -아래의 사항들은 자동으로 TRANSACTION 이 적용된다.

37 1)DDL 문이 실행된 경우

38 2)DCL 문이 실행된 경우

39 3)명시적으로 commit 이나 rollback 이 실행되지 않은 상태에서 exit 를 해서 sqlplus 를  
종료할 때

40 4)강제적으로 Transaction 이 적용되는 경우

41 5)시스템 장애가 발생할 경우

42

43

44 5.트랜잭션 암시적 처리

45 1)자동 COMMIT은 아래의 사항에 발생

46 -DDL 문이 실행된 경우

47 -DCL 문이 실행된 경우

48 -명시적으로 COMMIT 또는 ROLLBACK 이 실행되지 않은 채 SQL\*Plus 에서 정상 종료(exit)한 경우

49 2)-자동 ROLLBACK

50 -SQL\*Plus 의 비정상 종료 시 또는 시스템 장애 발생시

51

52

53 6. COMMIT이나 ROLLBACK 이전의 Data의 상태

54 1)단지 Memory Buffer에만 영향을 받았기 때문에 Data의 변경 이전 상태로 복구 가능하다.

55 2)현재 사용자는 SELECT 문장으로 결과를 확인 가능하다.

56 3)다른 사용자는 현재 사용자가 수행한 명령의 결과를 볼 수 없다.

57 4)변경된 행은 잠금(LOCKING)이 설정되어서 다른 사용자가 변경할 수 없다.

58

59

60

61 REM COMMIT

62 1. 변경 사항을 영구히 저장

63

64 UPDATE emp SET deptno = 10 WHERE empno = 7782;

65 COMMIT;

66

67 2. COMMIT 이후의 Data의 상태

68 1)Data에 대한 변경 사항이 Database에 반영된다.

69 2)이전 Data는 영원히 잃어버리게 된다.

70 3)모든 사용자는 결과를 볼 수 있다.

71 4)관련된 행에 대한 잠금(LOCKING)이 풀리고, 다른 사용자들이 행을 조작할 수 있게 된다.

72

73

74 REM ROLLBACK

75 1. 보류 중인 변경 내용을 모두 되돌림

76 -Data 변경 사항이 취소되어 Data의 이전 상태로 복구되며, 관련된 행에 대한 잠금(LOCKING)이 풀리고 다른 사용자들이 Data의 변경을 할 수 있게 된다.

77

78 2. ROLLBACK이후의 상태

79 1)Data에 대한 변경 사항은 취소된다.

80 2)이전 Data는 다시 재저장된다.

81 3)관련된 행에 대한 잠금(LOCKING)이 풀리고, 다른 사용자들이 행을 조작할 수 있게 된다.

82

83 DELETE FROM emp;

84 ROLLBACK;

85

86

87 REM COMMIT과 ROLLBACK의 효과

88 1. Data 무결성 보장

89 2. 영구적인 변경을 하기 전에 Data이 변경 사항 확인 가능

90 3. 논리적으로 연관된 작업을 그룹핑하여 처리 가능.  
91  
92  
93 REM **SAVEPOINT**  
94 1. 현재 시점에서 **SAVEPOINT**까지 **Transaction**의 일부만 **ROLLBACK** 할 수 있다.  
95 2. 따라서 복잡한 대규모 **Transaction**에서 Error가 발생했을 때 **SAVEPOINT**까지의 **Transaction**  
만 **Rollback**하고 실패한 부분에 대해서만 다시 실행할 수 있다.  
96 3. 복수의 저장점을 정의할 수 있으며, 동일한 이름으로 저장점을 정의했을 때는 나중에 정의한  
저장점이 유효하다.  
97 4. Syntax  
98 **SAVEPOINT** savepoint\_name;  
99 5. 저장점까지 **rollback**할 때는 **ROLLBACK** 뒤에 저장점 명을 지정한다.  
100 **ROLLBACK TO** savepoint\_name;  
101 6. **Rollback**에 **SAVEPOINT** 명을 부여하여 실행하면 저장점 설정 이후에 있었던 **Data** 변경에  
대해서만 원래 **Data** 상태로 되돌아가게 된다.  
102 7. 저장점 지정없이 **ROLLBACK**을 실행했을 경우에는 반영안된 모든 변경 사항을 취소하고  
**Transaction** 시작 위치로 되돌아간다.  
103  
104  
105 REM 저장점까지 변경 내용 **ROLLBACK**  
106 1. **SAVEPOINT** 사용하여 현재 트랜잭션에 표시자를 생성하여 트랜잭션을 더 작은 부분으로  
나눈 후 **ROLLBACK TO SAVEPOINT** 문을 사용해 오류 중인 변경 내용을 해당 표시자까지  
되돌림.  
107  
108 **START TRANSACTION;**  
109  
110 **UPDATE** emp **SET** deptno = 10 **WHERE** empno = 7782;  
111 **SAVEPOINT** a;  
112  
113 **INSERT INTO** emp(empno, ename, job, mgr, hiredate, sal, comm, deptno)  
114 **VALUES** (7999, 'TOM', 'SALESMAN', 7782, CURDATE(), 2000, 2000, 10);  
115  
116 **ROLLBACK TO** a;