Transactions

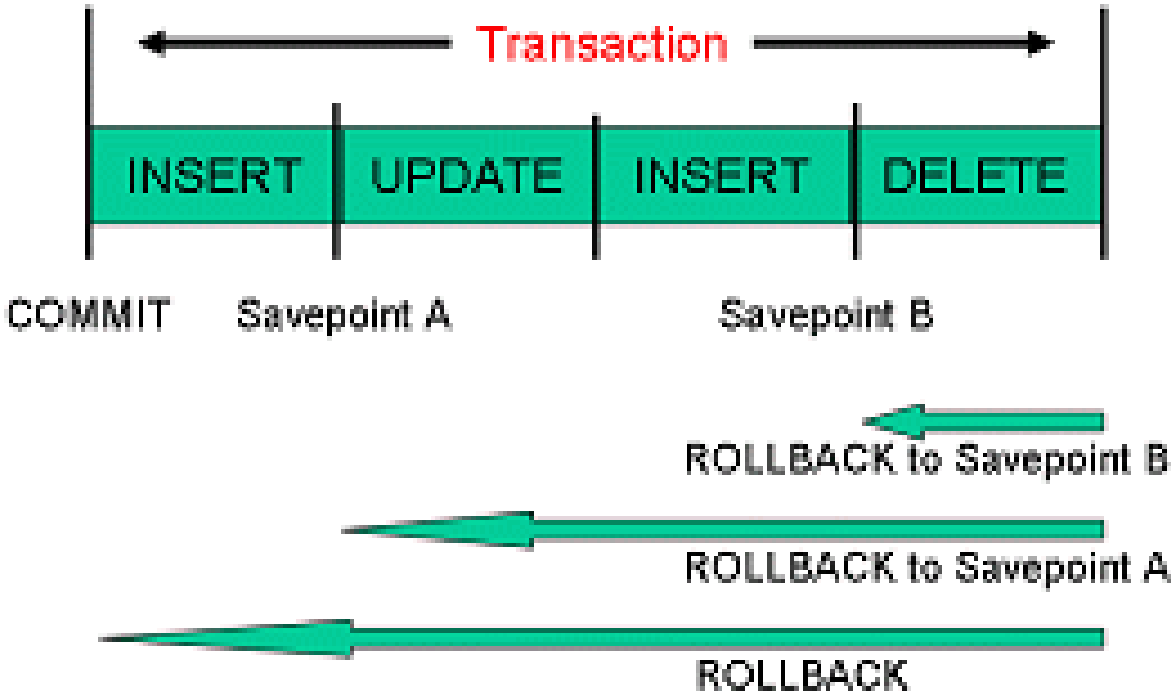
데이터베이스 트랜잭션

- 데이터 베이스 트랜잭션은 다음 중의 하나를 포함한다.
  - DML 문
  - 하나의 DDL (Data Definition Language) 문
  - 하나의 DCL (Data Control Language) 문
- 최초의 DML SQL문이 실행되면 시작한다.
- 다음 사건 중의 하나가 발생하면 종료한다.
  - COMMIT이나 ROLLBACK 문
  - DDL이나 DCL문이 실행 (자동적으로 commit)
  - 사용자가 iSQL*Plus 종료
  - System crashes (비정상적인 종료)

COMMIT 및 ROLLBACK문의 장점

- 데이터의 일관성을 제공할 수 있다.
- 영구적인 변경 전에 데이터의 변경사항을 확인할 수 있다.
- 관련된 작업을 논리적으로 그룹화 할 수 있다.

트랜잭션 제어


Transaction
INSERT
UPDATE
INSERT
DELETE
COMMIT
Savepoint A
Savepoint B
ROLLBACK to Savepoint B
ROLLBACK to Savepoint A
ROLLBACK

표시자까지 변경 내용 롤백

- SAVEPOINT 문을 사용하여 현재의 트랜잭션에 표시를 할 수 있다.
- ROLLBACK TO SAVEPOINT 문을 사용하여 표시점까지 Roll back 할 수 있다.

```
UPDATE …
        SAVEPOINT update_done;


        Savepoint created.


        INSERT …
        ROLLBACK TO update_done;


        Rollback complete.
```

암시적(Implicit) 트랜잭션 처리

- 자동 commit은 다음과 같은 상황하에서 발생한다.
  - DDL 문 실행
  - DCL 문 실행
  - 명시적으로 COMMIT이나 ROLLBACK 문을 실행하지 않고 정상적으로 iSQL*Plus 종료
- 자동 rollback은 iSQL*Plus에서 비정상적인 종료를 하거나 시스템 실패시 발생한다.

COMMIT 또는 ROLLBACK 실행이전의 데이터 상태

- 데이터의 이전 상태는 복구될 수 있다.
- 현재 사용자는 SELECT 문을 사용하여 DML 문의 결과를 확인할 수 있다.
- 다른 사용자는 현재 사용자가 수행한 DML 문의 결과를 볼 수 없다.
- 변경된 내용은 lock이 설정되어 다른 사용자가 변경할 수 없다.

COMMIT 실행 이후의 데이터 상태

- 데이터 베이스에 영구적으로 데이터를 변경한다.
- 데이터의 이전 상태는 완전히 상실된다.
- 모든 사용자가 결과를 볼 수 있다.
- 관련된 행에 대한 lock은 풀리고 다른 사용자들이 행을 조작할 수 있게 된다.
- 모든 savepoint가 제거된다.

데이터 커밋

- Make the changes.

```
DELETE FROM employees
 WHERE employee_id = 99999;

1 row deleted.

INSERT INTO departments
VALUES (290, 'Corporate Tax', NULL, 1700);

1 row inserted.
```

- Commit the changes.

```
COMMIT;

Commit complete.
```

ROLLBACK 실행이후의 데이터 상태

ROLLBACK 문을 사용하여 모든 결정되지 않은 변경 들을 버린다.

- 데이터의 변경은 취소된다.
- 데이터의 이전 상태로 복구된다.
- 관련된 행에 대한 lock이 풀린다.

```
DELETE FROM copy_emp;

        22 rows deleted.

        ROLLBACK;

        Rollback complete.
```

명령문 레벨 롤백

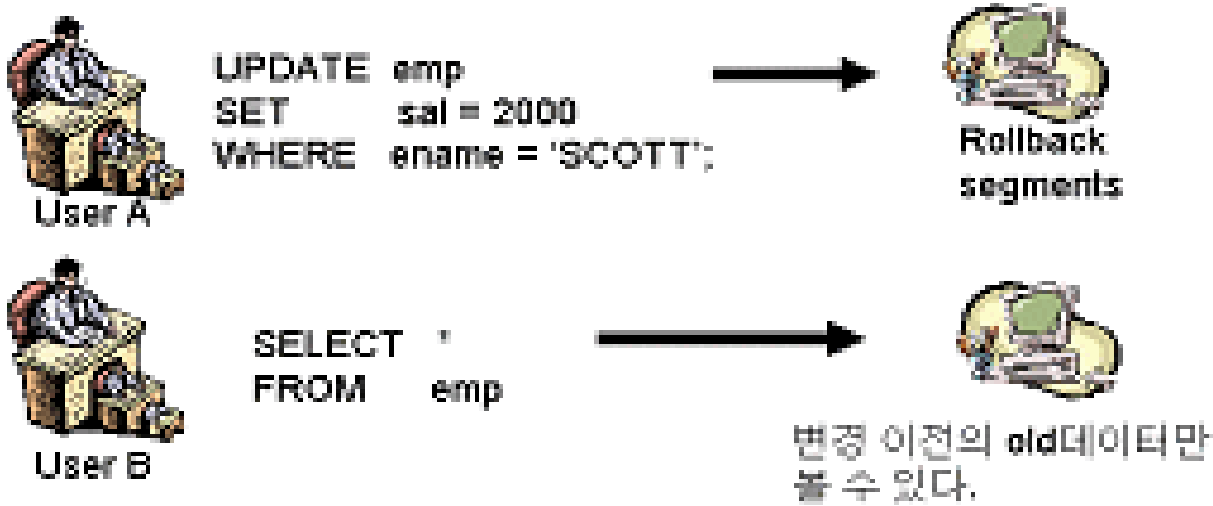
- 만약 단일 DML 문이 실행 중에 실패하면, 실패한 문장만이 roll back 된다.
- 오라클 서버는 묵시적인 savepoint를 만들어 현재의 트랜잭션을 commit하려 한다.
- 다른 모든 변경사항은 그대로 유지된다.
- COMMIT이나 ROLLBACK 문을 실행하여 명시적으로 트랜잭션을 종료해야 한다.

읽기 일관성

- 읽기 일관성은 항상 데이터의 검색이 일관되게 보증한다.
- 어떤 사용자에 의해 행해진 변경은 다른 사용자에 의해 행해진 변경과 충돌하지 않는다.
- 읽기 일관성은 데이터를 같게 보증한다.
  - Reader는 writer에 대하여 기다리지 않는다.
  - Writer는 reader에 대하여 기다리지 않는다.

읽기 일관성 구현

- 데이터베이스에의 두 가지 접근 형태
  - Read operation (SELECT)
  - Write operation (INSERT, UPDATE, DELETE)



잠금

- 동시 트랜잭션 사이의 상호 작용이 파괴되지 않도록 막아 준다.
- 사용자의 행동을 요구하지 않는다.
- 자동적으로 최소한의 제한이 적용된다.
- 묵시적인 잠금과 명시적인 잠금의 두 가지 형식이 있다.

암시적(Implicit) 잠금

- Two lock modes :
  - Exclusive : 다른 사용자를 들어오지 못하게 한다.
  - Share : 다른 사용자들의 접근을 허용한다.
- High level of data concurrency :
  - DML : 테이블 공유, 행 독점
  - Queries : 잠금을 요구하지 않는다.
  - DDL : 객체의 정의를 보호한다.
- Locks held until commit or rollback

실습

- STEP 1: Bell의 SALARY를 검색하는 질의를 작성하시오.
- STEP 2: Bell의 SALARY를 5,000으로 변경하시오.
- STEP 3: Bell의 SALARY가 제대로 갱신되었는지 확인하시오.
- STEP 4: 트랜잭션을 rollback 하시오.
- STEP 5: Bell의 SALARY가 원래 값으로 복구되었는지 확인하시오.
- STEP 6: Bell의 SALARY를 5,000으로 변경하시오.
- STEP 7: Salary5000 이라는 이름으로 세이브포인트를 설정하시오
- STEP 8: Bell의 SALARY를 7,000으로 변경하시오.
- STEP 9: Bell의 SALARY가 얼마인지 확인하시오.
- STEP 10: Salary5000위치로 rollback 하시오.
- STEP 11: Bell의 SALARY가 얼마인지 확인하시오.
- STEP 12: 트랜잭션을 commit 하시오.
- STEP 13: Bell의 SALARY가 얼마인지 확인하시오.

실습

- 두 개의 데이터베이스 접속을 만든다. 접속 A, 접속 B라고 부름.
  - 접속 A: tbAdmin의 접속창 (접속 안된 경우 File/New Connection 으로 접속)
  - 접속 B: cmd 창에서 tbsql 실행 후, connect hr/tibero 수행 (hr은 username, tibero는 암호임)
- STEP 1: 접속 A에서 아래 SQL 문 수행

```
UPDATE employees SET salary=7000 WHERE last_name = 'Bell';
```

- STEP 3: 접속 B에서 isolation level을 serializable로 설정

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

- STEP 4: 접속 B에서 아래 SELECT 문 수행

```
SELECT * FROM employees WHERE last_name = 'Bell';
```

- STEP 5: 접속 B에서 아래 UPDATE 문 수행 -> wait가 발생함

```
UPDATE employees SET department_id=100 WHERE last_name='Bell';
```

- STEP 6: 접속 A에서 롤백
- STEP 7: 접속 B가 진행됨을 확인하고 롤백
- STEP 4에서는 wait가 발생하지 않으나, STEP 5에서는 발생하는 이유는?