Oracle SQL

DML – INSERT, UPDATE, DELETE

Data Manipulation Language

- 종류
  - Add New Row(s)
    - INSERT INTO 테이블명 [(컬럼 리스트)] VALUES (값 리스트);
  - Modify Exsiting Row(s)
    - UPDATE 테이블명 SET 변경내용 [WHERE 조건];
  - Remove Existing Row(s)
    - DELETE FROM 테이블명 [WHERE 조건];

- 트랜잭션의 대상
  - 트랜잭션은 DML의 집합으로 이루어짐
  - ALL or NOTHING

INSERT

- 테이블에 새로운 튜플을 삽입할 때 사용하는 명령

- Syntax

```
INSERT INTO table_name [{column [, column ...]}]
{VALUES (value[, value...])|Subquery};
```

  - 대응하는 colum과 value는 개수와 타입이 일치해야 한다
  - 테이블 내 모든 컬럼의 내용을 삽입할 때는 column 명을 생략할 수 있지만, 이때는 CREATE TABLE 문에 기술된 컬럼 순으로 value 값들을 지정해야 한다
  - Subquery를 이용하여 다른 테이블의 검색 결과를 삽입할 수 있다

INSERT

- 묵시적 방법: 컬럼 이름, 순서 지정하지 않음. 테이블 생성시 정의한 순서에 따라 값 지정

```
INSERT INTO author
VALUES (1, '박경리', '토지 작가 ' );
```

- 명시적 방법: 컬럼 이름 명시적 사용. 지정되지 않은 컬럼은 NULL 자동 입력

```
INSERT INTO author( author_id, author_name )
VALUES (2, '이문열' );
```

- Subquery 이용: 타 테이블로부터 데이터 복사 (테이블은 미리 존재해야 함)

```
  INSERT INTO department_usa
   SELECT department_id, department_name
     FROM deptments
    WHEREd department_anme = 'IT';
```

  - 참고로 Subquery 결과를 없는 테이블을 생성하고 데이터를 복사하고자 할 때는 CREATE TABLE AS SELECT 이용

UPDATE

- 테이블에 있는 튜플들 중에서 특정 튜플의 내용을 갱신할 때 사용하는 명령

- Syntax

```
UPDATE table_name
SET column=value[, column=value ...]
[WHERE condition];
```

  - WHERE 절이 생략된 UPDATE 문장은 해당 테이블 내의 모든 Row를 변경하므로 주의해서 사용해야 한다

UPDATE

- 조건을 만족하는 레코드를 변경
  - 10번 부서원의 금여를 100 인상 & 수수료를 0으로 변경

```
UPDATE emp
SET sal = sal + 100, comm = 0
WHERE deptno = 10;
```

- WHERE 절이 생략되면 모든 레코드에 적용
  - 모든 직원의 급여를 10% 인상

```
UPDATE emp SET sal = sal * 1.1
```

- Subquery를 이용한 변경
  - 담당 업무가 'SCOTT'과 같은 사람들의 월급을 부서 최고액으로 변경

```
UPDATE emp SET sal = (SELECT MAX(sal) FROM emp)
WHERE job =
          (SELECT job FROM emp WHERE ename='SCOTT');
```

DELETE

- 테이블에 있는 튜플들 중에서 특정 튜플을 삭제할 때 사용하는 명령

- Syntax

```
DELETE FROM table_name
[WHERE condition];
```

  - WHERE 절이 생략된 DELETE 문장은 해당 테이블 내의 모든 Row를 삭제하므로 주의해서 사용해야 한다

DELETE

- 조건을 만족하는 레코드 삭제
  - 이름이 'SCOTT'인 사원 삭제

```
DELETE FROM emp
WHERE ename = 'SCOTT';
```

- 조건이 없으면 모든 레코드 삭제 (주의!)
  - 모든 직원 정보 삭제

```
DELETE FROM emp;
```

- Subquery를 이용한 DELETE
  - 'SALES' 부서의 직원 모두 삭제

```
DELETE FROM emp
WHERE deptno = (SELECT deptno FROM dept
                WHERE dname = 'SALES');
```

참고사항

- 데이터 입력, 수정시 자주 사용되는 Pseudo 컬럼
  - USER : 현재 사용자명
  - SYSDATE : 현재 날짜와 시간
  - ROWID : 열의 위치 정보

```
INSERT INTO emp(eno, hiredate) VALUES (200, SYSDATE);
```

- DEFAULT : default 값이 정의된 컬럼에 기본 값을 입력할 경우 사용

```
INSERT INTO book VALUES (200,'Gems', DEFAULT);
```

- DELETE와 TRUNCATE의 차이
  - Delete는 Rollback 가능. 그러나 대량의 log 등을 유발하므로 Truncate보다 느림

- 모든 DML 문은 Integrity Constraint를 어길 경우 에러 발생

Oracle SQL

Transaction

Transaction

- 정의
  - DB에서 하나의 작업으로 처리되는 논리적 작업 단위
  - DBMS의 Concurrency Control과 Recovery에서 중요한 역할을 수행

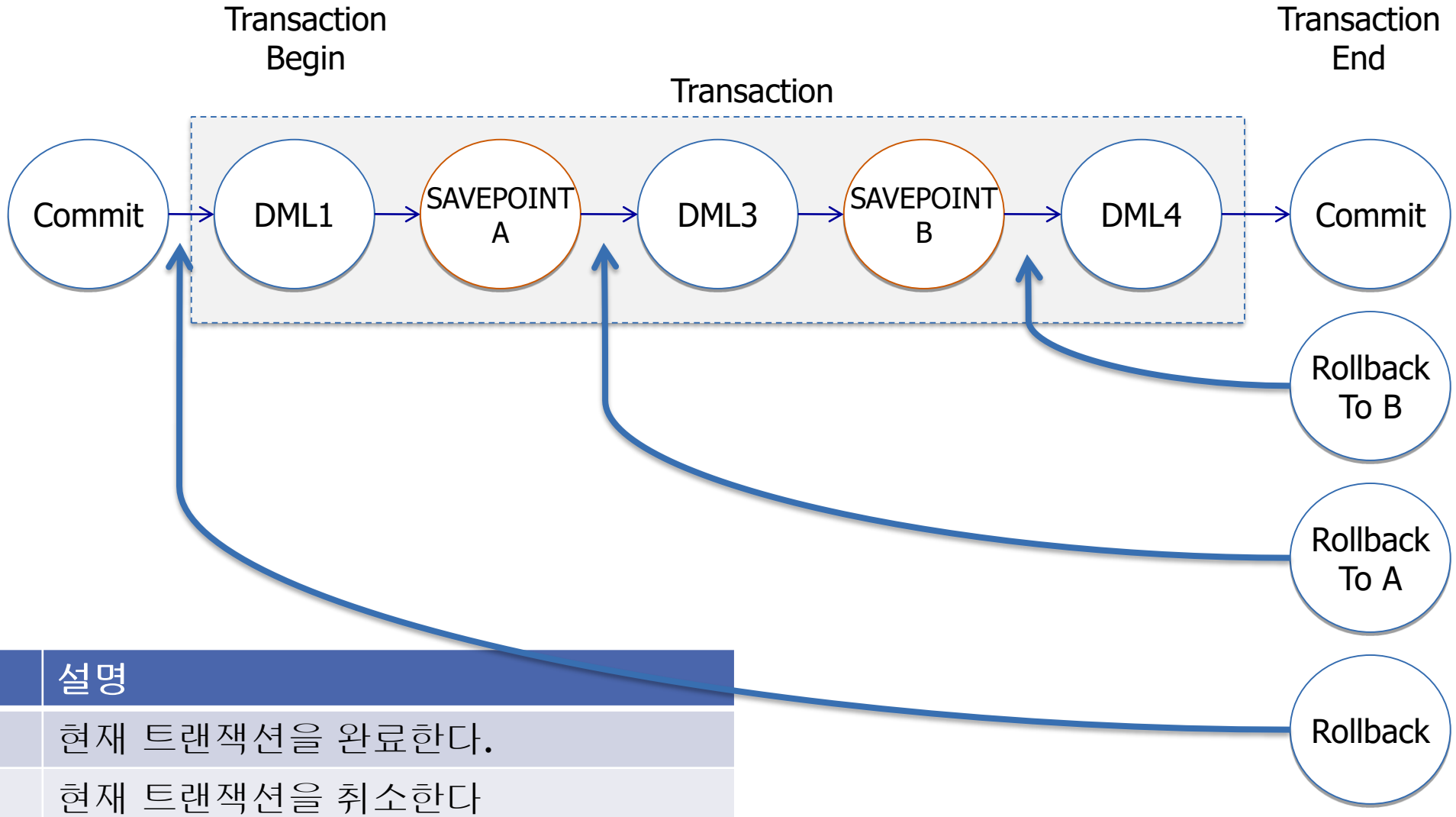
- ACID Property
  - 원자성(Atomicity): All or Nothing. 하나의 단위로 처리되어야 함 (중간까지만 처리됨은 불가)
  - 일관성(Consistency) : 데이터베이스의 일관성(무결성)을 깨지 않아야 함
  - 격리성(Isolation): 다른 Transaction과 동시에 수행되더라도 독립적으로 영향을 받지 않아야 햠
  - 영속성(Durability) : 한번 수행 완료(commit) 되면 영원히 반영되어 있어야 함(시스템 Crash 상황에서라도)

Transaction in Oracle

- 구성
  - DML (INSERT, UPDATE, DELETE)의 집합
  - DDL이나 DCL은 한 문장이 트랜잭션으로 처리

- 트랜잭션의 정의
  - 시작
    - 명시적인 트랜잭션 시작 명령 없음(타 DBMS와의 차이점)
    - 첫 DML이 시작되면 트랜잭션 시작
  - 명시적 종료:
    - COMMIT
    - ROLLBACK
  - 묵시적 종료
    - DDL, DCL 등이 수행될 때 (atomic commit)
    - SQL*PLUS 등에서의 정상적 종료 (atomic commit)
    - 시스템 오류 (atomic rollback)

Transaction Control



Command	설명
COMMIT	현재 트랜잭션을 완료한다.
ROLLBACK	현재 트랜잭션을 취소한다
SAVEPOINT x	트랜잭션의 중간에 롤백 지점을 만든다
ROLLBACK TO x	지정한 롤백 지점으로 돌아간다

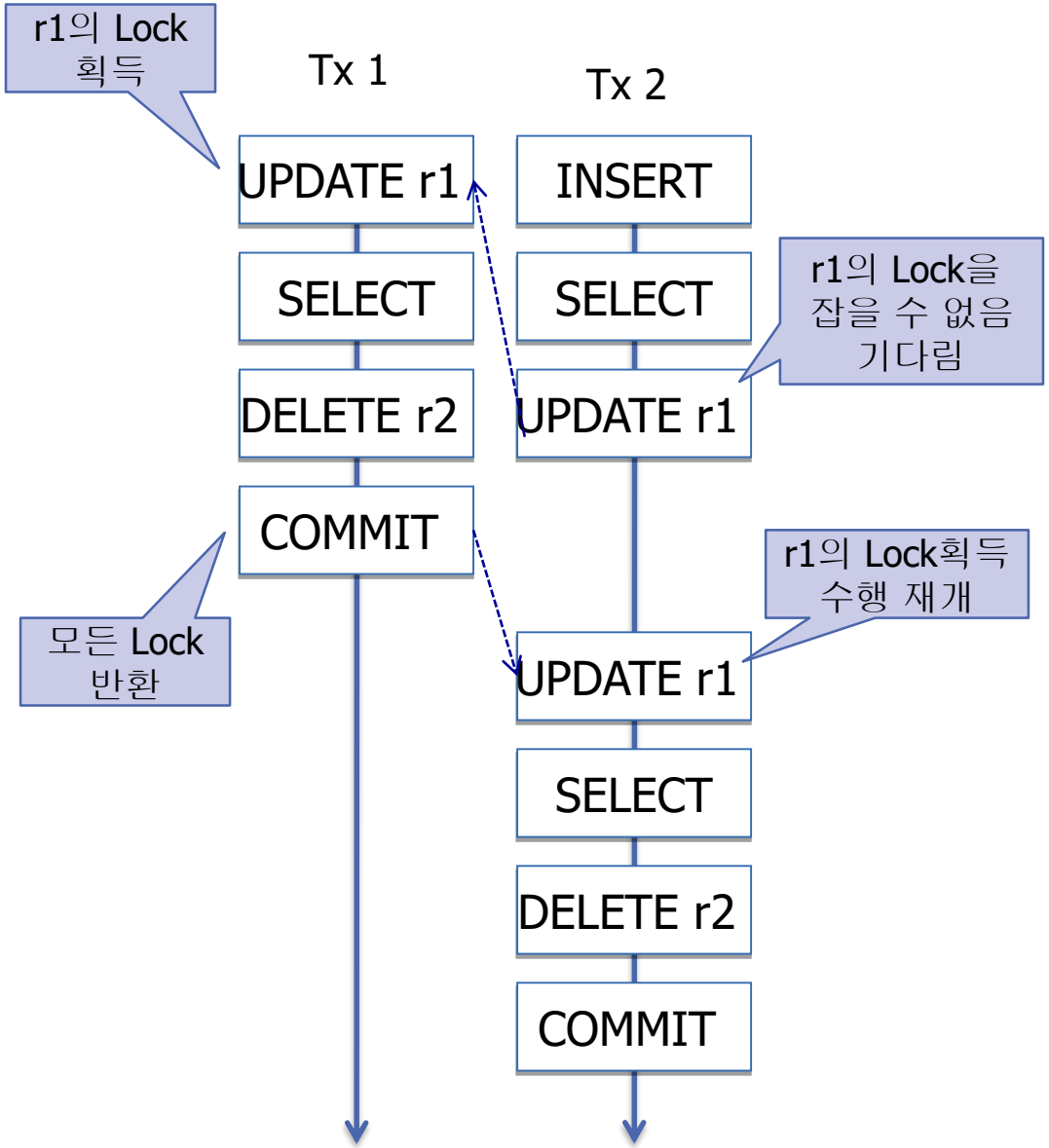
State of Data

- Before Commit/Rollback
  - 현재 사용자는 DML의 결과를 볼 수 있다
  - 다른 사용자는 현재 DML 결과를 볼 수 없다 (변경 이전 버전이 보임. Read Consistency)
  - DML에 의해 변경된 모든 Row는 Lock이 걸린다. (다른 트랜잭션에서 수정 불가)
- After Commit
  - 변경이 영속적으로 DB에 반영됨. 이전 상태는 사라짐 (더 이상 Rollback 불가)
  - 변경 결과를 모든 사용자가 볼 수 있음
  - 모든 Lock이 풀림. 모든 Savepoint 사라짐
- After Rollback
  - 모든 DML의 변경이 취소됨
  - 모든 Lock이 풀림. 모든 Savepoint 사라짐

Lock in Oracle

- Lock
  - Concurrent Transaction의 올바른 실행을 보장
  - Oracle이 자동으로 Lock을 관리
- Lock의 종류
  - DML: Table Share, Row Exclusive
    - 오라클이 자동으로 Lock 획득 시도
  - SELECT: No Locks required
  - DDL : Protects object definitions

- Locked 된 Row는 COMMIT이나 ROLLBACK이 수행될 때까지 유지됨



Set Transaction Property

- Transaction Type
  - SET TRANSACTION READ ONLY;
    - 읽기 전용 (Lock 필요 없음)
    - Transaction-Level Read Consistency (트랜잭션 시작할 때 봤던 값이 끝까지 나옴)
  - SET TRANSACTION READ WRITE; (default)
    - Statement-Level Read Consistency (Statement가 시작할 때 값을 읽게 됨), WRITE 가능
- Isolation Level
  - SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
    - 나보다 늦게 시작한 트랜잭션이 먼저 수정 후 COMMIT한 row를 수정하려면 ERROR 발생
    - 모든 충돌연산의 순서가 트랜잭션 순서와 동일해짐
  - SET TRANSACTION ISOLATION LEVEL READ COMMITTED; (default)
    - 기본적인 Lock 기반 변경. Commit 된 데이터 변경 가능

DEADLOCK

- 둘 이상의 트랜잭션이 서로 상대방의 Lock을 순환 대기하여 어떤 트랜잭션도 더 이상 진행할 수 없는 상태
- Oracle이 주기적으로 자동 detect 하여 에러를 돌려준다