ORACLE

11.트랜잭션

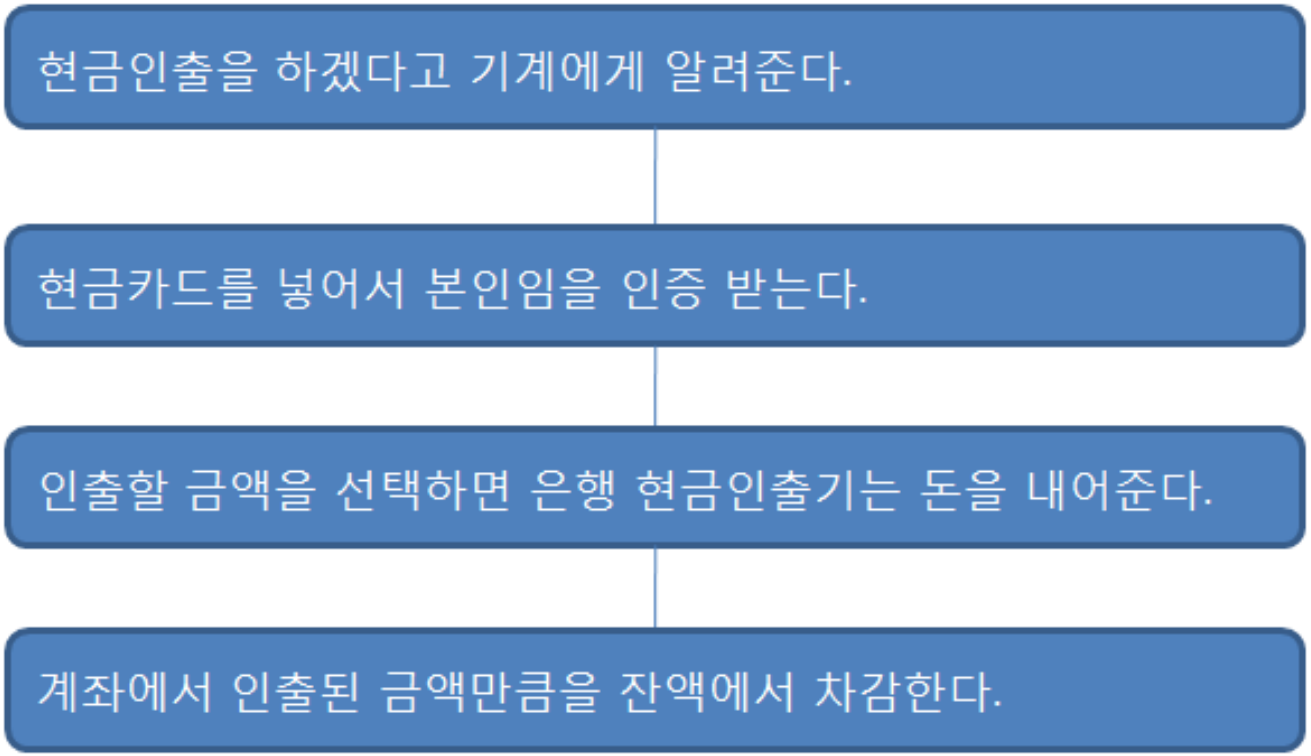
목차

1. 트랜잭션(1)

- 트랜잭션(Transaction)
  - 데이터 처리의 한 단위.
  - 오라클에서 발생하는 여러 개의 SQL 명령문들을 하나의 논리적인 작업 단위로 처리하는 과정
  - 하나의 트랜잭션은 All-or-Nothing 방식으로 처리
  - 여러 개의 명령어의 집합이 정상적으로 처리되면 정상 종료하도록 하고, 여러 개의 명령어 중에서 하나의 명령어라도 잘못되었다면 전체를 취소한다.
  - 데이터베이스에서 작업의 단위로 트랜잭션이란 개념을 도입한 이유는 데이터의 일관성을 유지하면서 안정적으로 데이터를 복구시키기 위해서이다.

1. 트랜잭션(2)

- 은행 현금 인출기(ATM) 에서 돈을 인출하는 과정으로 트랜잭션을 설명해 보자.



1. 트랜잭션(3)

- 이러한 거래에 있어서 지켜야 할 중요한 것이 있다.
- 기계의 오동작 등으로 인하여 전산 상으로는 돈을 인출한 것으로 입력이 되었는데 돈은 안나온다거나, 돈은 나왔는데 일련의 에러나 문제로 인하여서 돈을 인출한 것이 전산 상으로 입력이 안되면 상당히 심각한 문제가 발생할 수 있다.
- 이 때문에 전산 상으로도 입력이 정상적으로 잘 되고, 돈도 인출이 정상적으로 잘 됨을 확인하고 나서야, 인출하는 하나의 과정이 정상적으로 처리되었음을 확인할 수 있다.
- 여기서 돈을 인출하는 일련의 과정이 하나의 묶음으로 처리되어야 한다는 것을 알 수 있다.
- 혹시 처리 도중에 문제가 발생한다면 진행되던 인출 과정 전체를 취소하고 다시 처음부터 시작해야 되는데
- 이것을 트랜잭션이라 한다.

1. 트랜잭션(4)

- 트랜잭션 제어를 위한 명령어(Transaction Control Language)에는 다음과 같은 것들이 있다.

```
COMMIT
SAVEPOINT
ROLLBACK
```

2. COMMIT 과 ROLLBACK(1)

- 앞장에서 데이터를 추가, 수정, 삭제하는 작업들을 학습했는데, 이러한 데이터를 조작하는 명령어인 DML(Data Manipulation Language)은 이들이 실행됨과 동시에 트랜잭션이 진행된다.
- DML 작업이 성공적으로 처리되도록 하기 위해서 COMMIT명령을, 작업을 취소하기 위해서는 ROLLBACK 명령으로 종료해야 한다.
- COMMIT은 모든 작업들을 정상적으로 처리하겠다고 확정하는 명령어로 트랜잭션의 처리 과정을 데이터베이스에 모두 반영하기 위해서 변경된 내용을 모두 영구 저장한다.
- COMMIT 명령어를 수행하게 되면 하나의 트랜잭션 과정을 종료한다.

2. COMMIT 과 ROLLBACK(2)

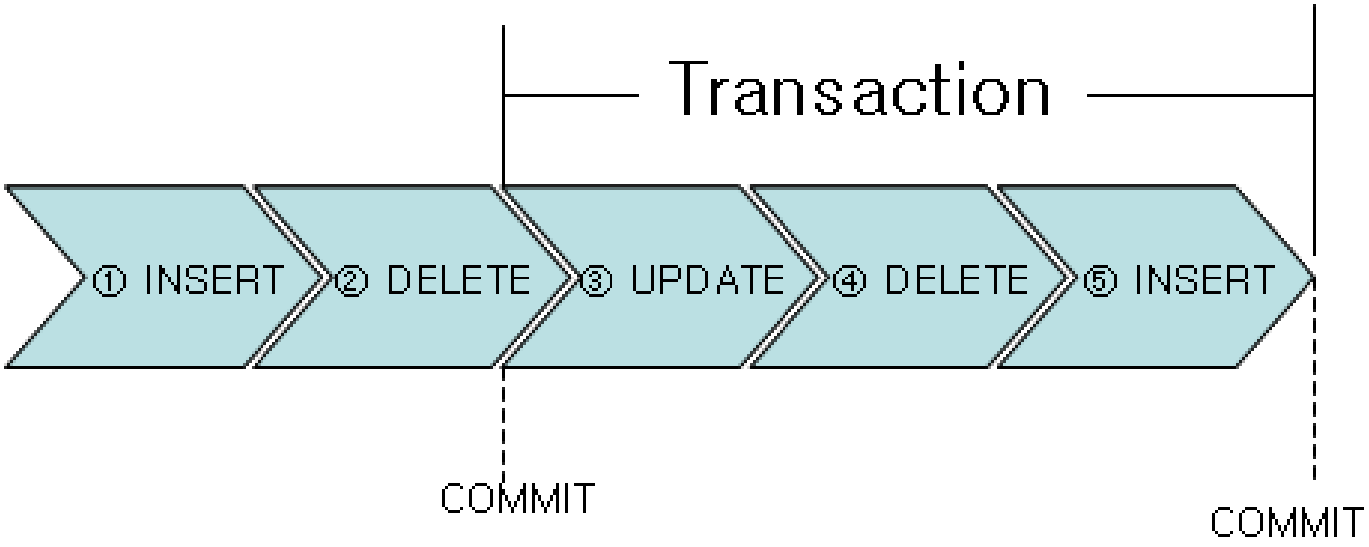
- ROLLBACK은 작업 중 문제가 발생되어서 트랜잭션의 처리 과정에서 발생한 변경사항을 취소하는 명령어이다.
- ROLLBACK 명령어 역시 트랜잭션 과정을 종료하게 된다.
- ROLLBACK은 트랜잭션으로 인한 하나의 묶음 처리가 시작되기 이전의 상태로 되돌린다.
- 트랜잭션은 여러 개의 물리적인 작업(DML 명령어)들이 모여서 이루어지는데 이러한 과정에서 하나의 물리적인 작업이라도 문제가 발생하게 되면 모든 작업을 취소해야 하므로 이들을 하나의 논리적인 작업 단위(트랜잭션)로 구성해 놓는다.
- 문제가 발생하게 되면 이 논리적인 작업 단위를 취소해 버리면 되기 때문이다.

2. COMMIT 과 ROLLBACK(3)

- 여러 개의 DML 명령어들을 어떻게 하나의 논리적인 단위인 트랜잭션으로 묶을 수 있을까?
- 트랜잭션은 마지막으로 실행한 COMMIT(혹은 ROLLBACK) 명령 이후부터 새로운 COMMIT(혹은 ROLLBACK) 명령을 실행하는 시점까지 수행된 모든 DML 명령들을 의미한다.

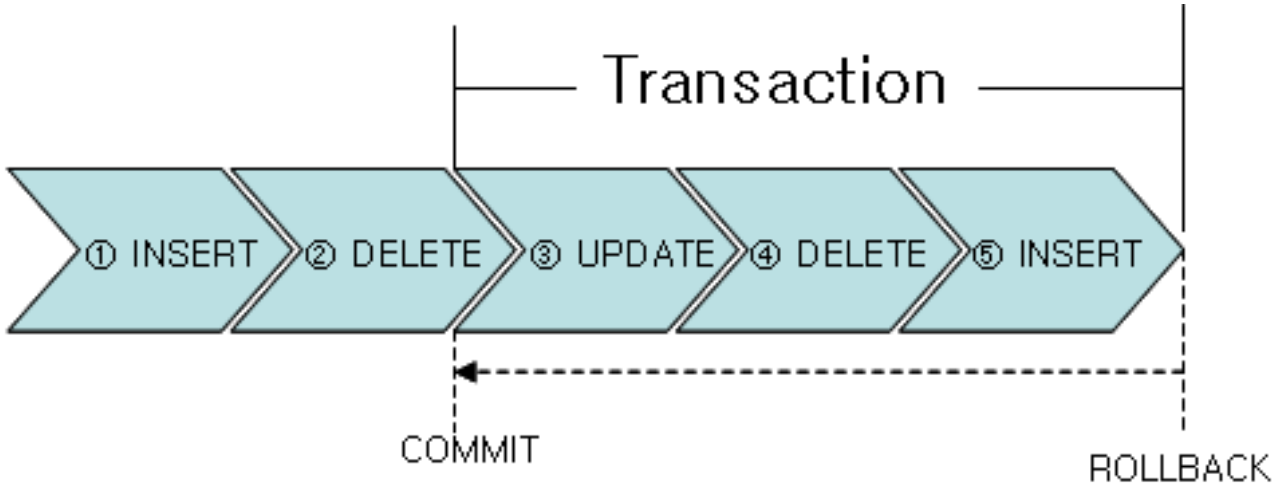
2. COMMIT 과 ROLLBACK(4)

- 아래 그림에서 UPDATE 문으로 데이터를 갱신하고(③), DELETE 문으로 데이터를 삭제하고(④), INSERT 문을 사용해 데이터를 삽입(⑤)한다.
- 만약 이 모든 과정이 오류 없이 수행되었다면 지금까지 실행한 모든 작업(③, ④, ⑤)을 "데이터베이스에 영구 저장하라"는 명령으로 COMMIT을 수행한다.



2. COMMIT 과 ROLLBACK(5)

- 롤백 명령은 마지막으로 수행한 COMMIT 명령까지만 정상 처리(①, ②)된 상태로 유지하고 그 이후에 수행했던 모든 DML 명령어 작업(③, ④, ⑤)들을 취소시켜 이전 상태로 원상 복귀시킨다.



- 트랜잭션은 이렇듯 All-or-Nothing 방식으로 DML 명령어들을 처리한다.

2. COMMIT 과 ROLLBACK(6)

- COMMIT과 ROLLBACK의 장점
  - 데이터 무결성이 보장된다.
  - 영구적인 변경 전에 데이터의 변경 사항을 확인할 수 있다.
  - 논리적으로 연관된 작업을 그룹화 할 수 있다.

- COMMIT
  - Transaction(INSERT, UPDATE, DELETE) 작업 내용을 실제 DB에 저장한다.
  - 이전 데이터가 완전히 UPDATE 된다.
  - 모든 사용자가 변경된 데이터의 결과를 볼 수 있다.

- ROLLBACK
  - Transaction(INSERT, UPDATE, DELETE) 작업 내용을 취소한다.
  - 이전 COMMIT한 곳/이전 트랜잭션이 종료된 지점 까지만 복구된다.

3. AUTO COMMIT(1)

- 데이터베이스 사용자가 COMMIT이나 ROLLBACK 명령어를 명시적으로 수행시키지 않더라도 다음과 같은 경우에 자동 커밋 혹은 자동 롤백이 발생한다.
- 자동 커밋과 자동 롤백 명령이 되는 경우
  - SQL*PLUS가 정상 종료되었다면 자동으로 COMMIT 되지만, 비정상 종료되었다면 자동으로 ROLLBACK 된다.
  - DDL과 DCL 명령문이 수행된 경우 자동으로 COMMIT 된다.
  - 정전이 발생했거나 컴퓨터 Down시(전원 끊김) 자동으로 ROLLBACK된다.

3. AUTO COMMIT(2)

- ex) CREATE문에 의한 자동 커밋에 의해서 이전에 수행했던 DML 명령어가 자동으로 커밋됨을 확인한다.
- 1. 부서 번호가 40번인 부서를 삭제한다.

예

```
DELETE * FROM DEPT02
WHERE DEPTNO=40;
```

- 2. 삭제 후 부서 테이블(DEPT)과 동일한 내용을 갖는 새로운 테이블(DEPT03)을 생성한다.

예

```
CREATE TABLE DEPT03
AS
SELECT * FROM DEPT;
```

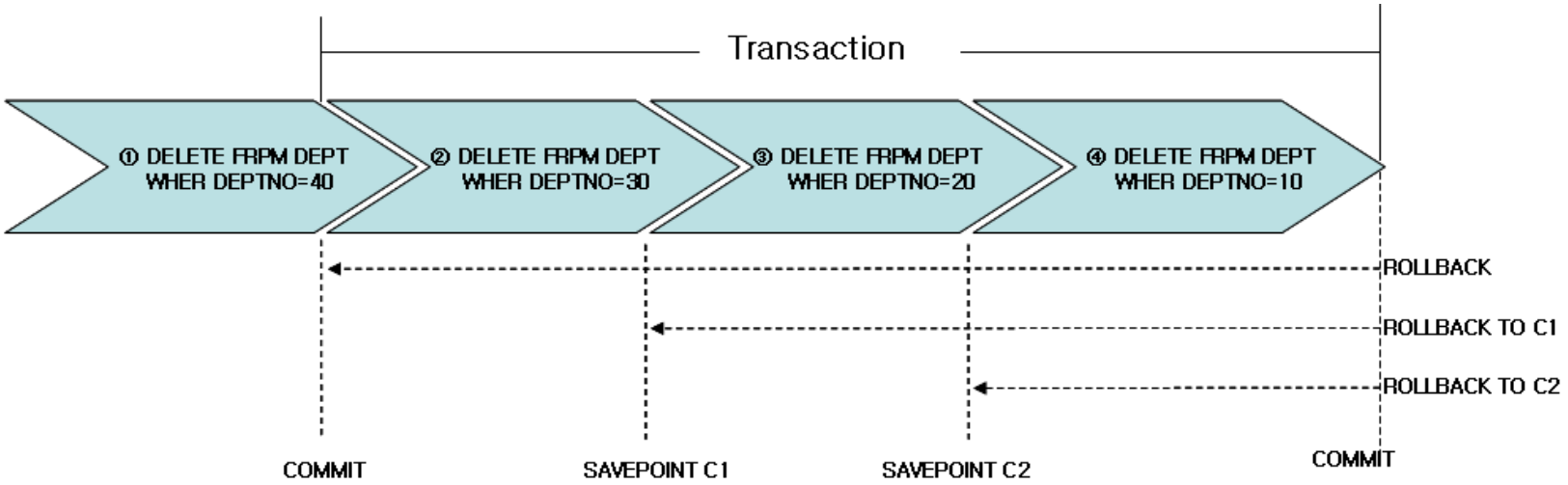
- 3. DEPT02 테이블의 부서번호가 40번인 부서를 다시 되살리기 위해서 ROLLBACK 명령문을 수행하여도 이미 수행한 CREATE문 때문에 자동으로 커밋이 발생하였으므로 되살릴 수 없다.

4. SAVEPOINT(1)

- SAVEPOINT 명령을 써서 현재의 트랜잭션을 작게 분할할 수 있다.
- 저장된 SAVEPOINT는 ROLLBACK TO SAVEPOINT 문을 사용하여 표시한 곳까지 ROLLBACK할 수 있다.
- 여러 개의 SQL문의 실행을 수반하는 트랜잭션의 경우, 사용자가 트랜잭션 중간 단계에서 SAVEPOINT를 지정할 수 있다.
- 이 SAVEPOINT는 차후 ROLLBACK과 함께 사용해서 현재 트랜잭션 내의 특정 SAVEPOINT까지 ROLLBACK 할 수 있게 된다.
- 뒤의 예제에서 ROLLBACK TO C1하면 SAVEPOINT도 지워지고 C2라는 SAVEPOINT도 지워진다.

4. SAVEPOINT(3)

- 아래 그림을 보면 COMMIT 명령이 내려진 후 다음 COMMIT 명령이 나타날 때까지가 하나의 트랜잭션으로 구성되므로 ②번에서 ④번까지가 하나의 트랜잭션이 된다.
- 이렇게 트랜잭션을 구성할 때 중간 중간 SAVEPOINT 명령으로 위치를 지정해 놓으면(예를 들어 C) 하나의 트랜잭션 내에서도 ROLLBACK TO C(SAVEPOINT 문을 사용하여 표시한 곳)까지 ROLLBACK할 수 있다.



4. SAVEPOINT(4)

- ex) SAVEPOINT로 특정 위치를 지정하기 위한 사용 형식이다.

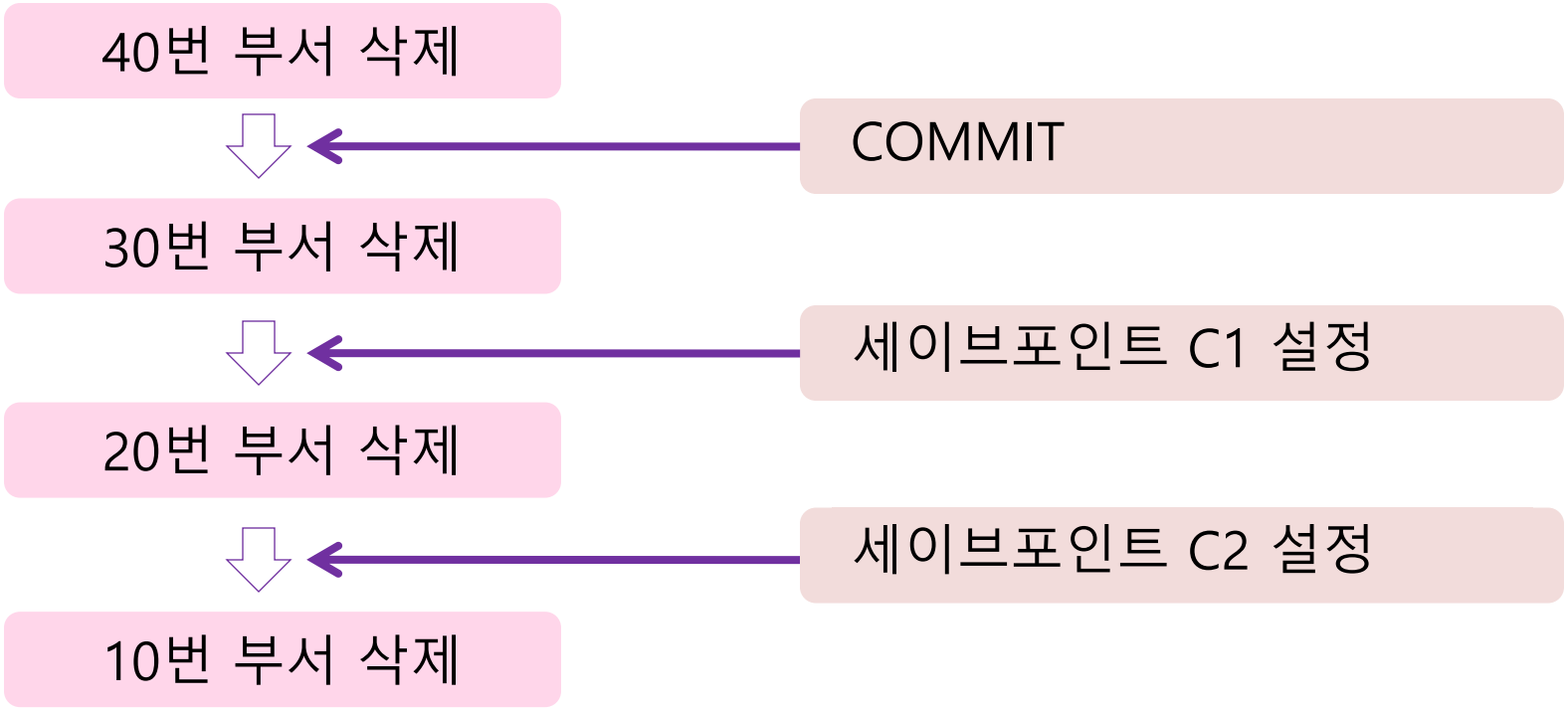
형식 **SAVEPOINT *LABEL_NAME;***

- ex) SAVEPOINT로 지정해 놓은 특정 위치로 되돌아가기 위한 사용형식이다.

형식 **ROLLBACK TO *LABEL_NAME;***

4. SAVEPOINT(5)

❖ ex) 다음과 같이 트랜잭션 중간 단계에서 세이브포인트를 지정해보자.



4. SAVEPOINT(6)

- 1. 부서 번호가 40번인 부서를 삭제한 후에 커밋을 수행하여 새롭게 트랜잭션을 시작한다.

형식	**DELETE FROM DEPT01 WHERE DEPTNO=40;** **COMMIT;**

- 2. 부서 번호가 30번인 부서를 삭제한다.

형식	**DELETE FROM DEPT01 WHERE DEPTNO=30;**

- 3. 세이브포인트 C1을 설정한 후, 부서 번호가 20번인 사원을 삭제한다.

형식	**SAVEPOINT C1;** **DELETE FROM DEPT01 WHERE DEPTNO=20;**

- 4. 세이브포인트 C2를 설정한 후, 부서 번호가 10번인 사원을 삭제한다.

형식	**SAVEPOINT C2;** **DELETE FROM DEPT01 WHERE DEPTNO=10;**

4. SAVEPOINT(7)

❖ ex) 부서번호가 10번인 사원을 삭제하기 바로 전으로 되돌리기 위해 세이브 포인트를 이용해서 트랜잭션 중간 단계로 돌려본다.

* ROLLBACK 명령을 내리게 된다면 이전 COMMIT 지점으로 되돌아가므로 10, 20, 30번 부서의 삭제가 모두 취소된다. 따라서 원했던 10번 부서의 삭제 이전까지만 되돌리려면 다시 30, 20번의 부서를 삭제해 주서야 한다.

형식 **ROLLBACK TO C2;**

❖ 위 결과 화면을 보면 세이브포인트 C2 지점으로 이동되어 10번 부서의 삭제 이전으로 되돌려진 것을 확인할 수 있다.

형식 **ROLLBACK TO C1;**

❖ 마지막으로 이전 트랜잭션까지 롤백한 후의 결과를 본다.

형식 **ROLLBACK;**