SQL 활용

트랜잭션

한국기술교육대학교
온라인평생교육원

학습내용

- 트랜잭션
- 트랜잭션 제어문(TCL)

학습목표

- 트랜잭션을 이해하고 설명할 수 있다.
- 트랜잭션 제어문(TCL)을 이용하여 데이터를 원상태로 돌릴 수 있다.

트랜잭션

1. 트랜잭션의 개념

◆ 트랜잭션이란?

- 트랜잭션(Transaction) : 논리적인 일의 단위
- 기본 설정

하나의 SQL은 하나의 트랜잭션임

  - 여러 개의 SQL문들이 합쳐져서 하나의 트랜잭션이 될 수도 있음

◆ 트랜잭션의 활용

대규모 데이터베이스를 수백, 수천 명 이상의 사용자들이 동시에 접근함

많은 사용자들이 동시에 데이터베이스의 서로 다른 부분 또는 동일한 부분을 접근하면서 데이터베이스를 사용함

⇨ 동시성

◆ 트랜잭션의 활용

- 동시성 제어
  - 동시에 수행되는 트랜잭션들이 데이터베이스에 미치는 영향 = 이들을 순차적으로 수행하였을 때 데이터베이스에 미치는 영향과 같도록 보장함
  - 다수 사용자가 데이터베이스를 동시에 접근하도록 허용하면서 데이터베이스의 일관성을 유지함
  - 여러 사용자나 여러 응용 프로그램들이 동시에 수행되어도 서로 간섭하지 못하도록 보장함

  ⇨ 트랜잭션 단위, 동시성 제어

- 회복
  - 데이터베이스를 갱신하는 도중에 시스템 고장 시에도 데이터베이스의 일관성을 유지함

  ⇨ 트랙잭션 단위 회복

트랜잭션

1. 트랜잭션의 개념

◆ 트랜잭션이 없다면?

① 은행 계좌 이자 증가

- 전체 계좌들에 대한 이자가 모두 계산되어야 함
- 만약 일부 계좌 이자만 증가되고 컴퓨터가 다운되었다가 재가동 된다면?

⇨ 처음부터 다시 계산하면 이중 이자 계산이 됨

② 다양한 예약 시스템

- 항공권, 극장 등의 예약 시스템
- 좌석을 선점하고 돈을 내기 전에 시스템이 다운됨
- 돈은 내지 않았지만 좌석을 잡았기 때문에 해당 좌석은 절대 다시 잡을 수 없어짐

③ 은행 계좌 이체 : A계좌에서 100원을 빼서 B계좌에 넣기

```
UPDATE ACCOUNT
SET BALANCE = BALANCE - 100
WHERE ID = A
```

```
UPDATE ACCOUNT
SET BALANCE = BALANCE + 100
WHERE ID = B
```

● 트랜잭션

1. 트랜잭션의 개념

◆ 트랜잭션이 없다면?

③ 은행 계좌 이체 : A계좌에서 100원을 빼서 B계좌에 넣기

● 계좌 이체 시 장애발생

```
UPDATE ACCOUNT
SET BALANCE = BALANCE - 100
WHERE ID = A
```

⬇ 장애

```
UPDATE ACCOUNT
SET BALANCE = BALANCE + 100
WHERE ID = B
```

▪ A 통장에서 돈만 빠져 나가고 B통장에 돈이 안 들어옴

⇨ 은행이 고객의 돈을 횡령한 것이 됨

● 횡령을 피하기 위해 먼저 B 계좌에 돈을 입금…

```
UPDATE ACCOUNT
SET BALANCE = BALANCE + 100
WHERE ID = B
```

⬇ 장애

```
UPDATE ACCOUNT
SET BALANCE = BALANCE - 100
WHERE ID = A
```

▪ B 통장에서 돈이 들어 왔는데 A 통장에 돈이 안 빠짐

⇨ 은행에 막대한 손실이 발생함

● 두 개의 SQL을 모아서 하나의 트랜잭션(계좌이체 업무)으로 관리함

두 DML문은 하나의 업무에 속한 작업들임

트랜잭션

2. 트랜잭션의 특성

◆ ACID

① Atomicity : 원자성

- 한 트랜잭션 내의 모든 연산들이 완전히 수행되거나 전혀 수행되지 않음(All or Nothing)을 의미함
- DBMS의 회복 모듈은 시스템이 다운되는 경우에, 부분적으로 데이터베이스를 갱신한 트랜잭션의 영향을 취소함으로써
- 트랜잭션의 원자성을 보장함
- 완료된 트랜잭션이 갱신한 사항은 트랜잭션의 영향을 재수행함으로써 트랜잭션의 원자성을 보장함

② Consistency : 일관성

- 어떤 트랜잭션이 수행되기 전에 데이터베이스가 일관된 상태를 가졌다면 트랜잭션이 수행된 후에 데이터베이스는 또 다른 일관된 상태를 가짐
- 트랜잭션이 수행되는 도중에는 데이터베이스가 일시적으로 일관된 상태를 갖지 않을 수 있음

③ Isolation : 격리성

- 고립성이라고도 함
- 한 트랜잭션이 데이터를 갱신하는 동안 이 트랜잭션이 완료되기 전에는 갱신 중인 데이터를 다른 트랜잭션들이 접근하지 못하도록 해야 함
- 다수의 트랜잭션들이 동시에 수행되더라도 그 결과는 어떤 순서에 따라 트랜잭션들을 하나씩 차례대로 수행한 결과와 같아야 함
- DBMS의 동시성 제어 모듈이 트랜잭션의 고립성을 보장함
- DBMS는 응용들의 요구사항에 따라 다양한 고립 수준(Isolation Level)을 제공함

④ Durability : 영속성

- 일단 한 트랜잭션이 완료되면 이 트랜잭션이 갱신한 것은 그 후에 시스템에 고장이 발생하더라도 손실되지 않음
- 완료된 트랜잭션의 효과는 시스템이 고장 난 경우에도 데이터베이스에 반영됨
- DBMS의 회복 모듈은 시스템이 다운되는 경우에도 트랜잭션의 지속성을 보장함

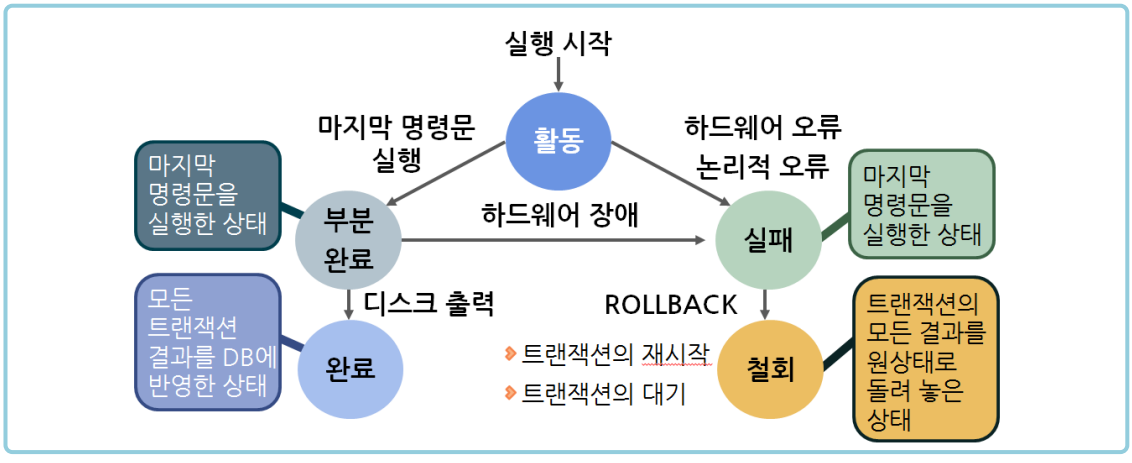
2. 트랜잭션의 특성

◆ ACID와 DB 기능

- ACID 특성과 DB의 기능은 모두 다 연관이 있음(연관성이 높은 순서)
  - DB 회복 기능 : 원자성과 지속성에 연관됨
  - DB 동시성 제어 : 일관성과 고립성에 연관
  - 무결성 제약 조건 : 일관성과 관련

3. 트랜잭션의 상태

◆ 트랜잭션의 상태 변화



트랜잭션 제어문(TCL)

1. 트랜잭션 제어문(TCL)

◆ COMMIT

- 트랜잭션의 마지막 명령어가 수행되었음을 나타냄
- 트랜잭션에 의한 변경을 확정
- COMMIT된 트랜잭션은 철회가 불가능함
- COMMIT 명령문 실행하기 전에 하나의 트랜잭션 변경한 결과를 다른 트랜잭션에서 접근할 수 없도록 방지하여 일관성을 유지함

◆ ROLLBACK

- 트랜잭션의 변경을 취소하고 트랜잭션 종료

◆ SAVEPOINT

- 현재 트랜잭션에서 ROLLBACK 시킬 위치 지정
- 대규모 트랜잭션(복수개의 명령어들로 이루진 트랜잭션)에서 오류 발생이 전체 트랜잭션을 취소 시키는 것이 큰 부담이 될 수 있음
- 실패한 일정부분만 취소 시키도록 함

트랜잭션 제어문(TCL)

2. 트랜잭션 모드

◆ MS-SQL은 3가지의 트랙잭션 모드를 지원함

① 자동 커밋 트랜잭션
- 하나의 명령문이 하나의 트랙잭션이 됨
- MS-SQL에서 기본 모드임

② 명시적 트랜잭션
- 명시적으로 사용자가 트랙잭션을 정의하는 형태
- BEGIN TRAN ~ COMMIT TRAN(또는 ROLLBACK TRAN)으로 이루어짐

③ 묵시적 트랜잭션
- 자동 커밋 트랜잭션의 반대되는 개념
- 사용자가 COMMIT TRAN(또는 ROLLBACK TRAN)을 입력하기 전까지 복수 개의 명령문을 하나의 트랜잭션으로 간주함
- BEGIN TRAN이 필요 없음
- 묵시적 트랜잭션의 설정

```
SET IMPLICIT TRANSACTIONS {ON|OFF}
```

- 트랜잭션 종료마다 사용자가 반드시 COMMIT / ROLLBAK을 명령문을 실행시켜야 함
- 고급 사용자가 아닌 이상 가능한 사용하지 않는 것이 좋음

3. 트랜잭션 제어문(TCL) 활용

◆ 간단한 트랜잭션 철회

Q 실습을 위하여 DEPARTMENT 테이블 내용을 DEPT01로 복사하기

```
USE MagicCorp
GO

SELECT * INTO DEPT01 FROM DEPARTMENT
SELECT * FROM DEPT01
```

결과 | 메시지

	DNO	DNAME	LOC
1	10	Accounting	Seoul
2	20	Human	Incheon
3	30	Sales	Yungin
4	40	Computing	Suwon

Q 트랜잭션을 시작한 후 DEPT01 테이블의 내용을 모두 지우고 트랜잭션 취소를 시켜보기

① 트랜잭션을 시작

② DEPT01 테이블 내용 지우기 ⇨ DEPT01 내용 보기

③ ROLLBACK ⇨ DEPT01 내용 보기

```
USE MagicCorp
GO

BEGIN TRAN

DELETE DEPT01

SELECT * FROM DEPT01

ROLLBACK TRAN

SELECT * FROM DEPT01
```

결과 | 메시지

	DNO	DNAME	LOC

	DNO	DNAME	LOC
1	10	Accounting	Seoul
2	20	Human	Incheon
3	30	Sales	Yungin
4	40	Computing	Suwon

3. 트랜잭션 제어문(TCL) 활용

◆ 오류발생에 따른 트랜잭션 철회

● 트랜잭션을 구성하는 명령문들 중 : 오류 발생 ⇨ 트랜잭션 철회

오류 발생하지 않음 ⇨ 완료

- MS-SQL에서 명령문의 오류는 @@ERROR라는 변수에 저장됨
- T-SQL에서 IF ~ ELSE ~ 및 GOTO 같은 구문을 사용할 수 있음

● DEPT01 테이블의 DNO는 NULL값이 올 수 없음

- 테이블 구조정보 보는 법 : EXEC sp_help 테이블명

	Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenN
1	DNO	int	no	4	10	0	no	(n/a)	(n/a)
2	DNAME	varchar	no	20			yes	no	yes
3	LOC	varchar	no	20			yes	no	yes

● 하나의 트랙잭션을 이용함

① DEPT01 테이블에서 부서번호 10번 튜플을 삭제함

② (NULL, 'PRODUCT', 'Seoul') 튜플을 삽입 ⇨ 오류 발생

③ 오류가 발생하면 해당 트랜잭션을 ROLLBACK함

```
USE MagicCorp
GO

BEGIN TRAN

DELETE FROM DEPT01 WHERE DNO = 10
SELECT * FROM DEPT01

INSERT INTO DEPT01 VALUES (NULL, 'PRODUCT', 'Seoul')

IF @@ERROR <> 0 GOTO ERROR_ROLLBACK

COMMIT TRAN
RETURN

ERROR_ROLLBACK:
ROLLBACK TRAN
GO

SELECT * FROM DEPT01
GO
```

결과 메시지

	DNO	DNAME	LOC
1	20	Human	Incheon
2	30	Sales	Yungin
3	40	Computing	Suwon

	DNO	DNAME	LOC
1	10	Accounting	Seoul
2	20	Human	Incheon
3	30	Sales	Yungin
4	40	Computing	Suwon

3. 트랜잭션 제어문(TCL) 활용

◆ SAVEPOINT를 이용한 트랜잭션 부분 철회

● 트랜잭션 내에서 SAVEPOINT의 지정

```
SAVE TRAN 저장점명
```

▪ 트랜잭션 내에 저장점명을 다르게 하면 여러 개의 SAVEPOINT를 지정할 수 있음

● 저장점 위치로 취소

```
ROLLBACK TRAN 저장점명
```

● 하나의 트랙잭션을 이용함

① DEPT01 테이블에서 부서번호 10번 튜플을 삭제함

② 저장점 설정함

③ (50, 'PRODUCT', 'Seoul') 추가함

④ 저장점으로 ROLLBACK

⑤ (60, DESIGN, 'Jeju') 추가함

⇨ SAVEPOINT 에 의하여 (50, 'PRODUCT', 'Seoul')을 삽입되지 않을 것을 알 수 있음

⇨ (60, 'DESING', 'Jeju')는 삽입되었음

```
USE MagicCorp
GO

BEGIN TRAN

DELETE FROM DEPT01 WHERE DNO = 10

SAVE TRAN svpoint1

INSERT INTO DEPT01 VALUES (50, 'PRODUCT', 'Seoul')

ROLLBACK TRAN svpoint1

INSERT INTO DEPT01 VALUES(60, 'DESIGN', 'Jeju')

COMMIT TRAN
GO

SELECT * FROM DEPT01
```

결과 | 메시지

	DNO	DNAME	LOC
1	60	DESIGN	Jeju
2	20	Human	Incheon
3	30	Sales	Yungin
4	40	Computing	Suwon

핵심요약

1. 트랜잭션

■ 트랜잭션의 개념

- 트랜잭션(Transaction)
  - 논리적인 일의 단위
- 기본 설정 : 하나의 SQL은 하나의 트랜잭션임
- 여러 개의 SQL문들이 합쳐서 하나의 트랜잭션이 될 수도 있음
- 트랜잭션의 활용
  - 동시성 제어 : 여러 사용자나 여러 응용 프로그램들이 동시에 수행되어도 서로 간섭하지 못하도록 보장함
  - 회복 : 데이터베이스를 갱신하는 도중에 시스템 고장 시에도 데이터베이스의 일관성을 유지함

■ 트랜잭션의 특성

- 원자성(Atomicity)
  - 한 트랜잭션 내의 모든 연산들이 완전히 수행되거나 전혀 수행되지 않음
- 일관성(Consistency)
  - 어떤 트랜잭션이 수행되기 전에 데이터베이스가 일관된 상태를 가졌다면 트랜잭션이 수행된 후에 데이터베이스는 또 다른 일관된 상태를 가짐
- 격리성(Isolation)
  - 한 트랜잭션이 데이터를 갱신하는 동안 이 트랜잭션이 완료되기 전에는 갱신 중인 데이터를 다른 트랜잭션들이 접근하지 못하도록 해야 함
- 영속성(Durability)
  - 일단 한 트랜잭션이 완료되면 이 트랜잭션이 갱신한 것은 그 후에 시스템에 고장이 발생하더라도 손실되지 않음

핵심요약

1. 트랜잭션

■ 트랜잭션의 상태

- 부분완료 : 마지막 명령문을 실행한 상태
- 완료 : 모든 트랜잭션 결과를 DB 에 반영한 상태
- 실패 : 트랜잭션의 실패
- 철회 : 트랜잭션의 모든 결과를 원상태로 돌려 놓은 상태

핵심요약

2. 트랜잭션 제어문(TCL)

■ 트랜잭션 제어문

- COMMIT

  - 트랜잭션의 마지막 명령어가 수행되었음을 나타냄

- ROLLBACK

  - 트랜잭션의 변경을 취소하고 트랜잭션 종료

- SAVEPOINT

  - 현재 트랜잭션에서 ROLLBACK 시킬 위치 지정

■ 트랜잭션 제어문(TCL) 활용

- 간단한 트랜잭션 철회
- 오류발생에 따른 트랜잭션 철회

  - 트랜잭션을 구성하는 명령문들 중에서 오류가 발생되면 트랜잭션을 철회하고 그렇지 않으면 완료하는 것이 필요

- SAVEPOINT를 이용한 트랜잭션 부분 철회

  - 트랜잭션 내에서 SAVEPOINT의 지정

```
SAVE TRAN 저장점명
```

  - 저장점 위치로 취소

```
ROLLBACK TRAN 저장점명
```