제6장 병행제어

1. 충돌(conflict)

먼저, 병행제어(동시성제어)에 대해서 정리한다.

병행제어 정의

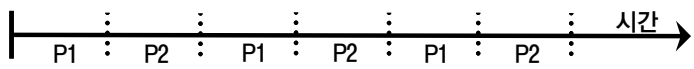
- 다중사용자 환경에서 하나의 데이터베이스에 여러 트랜잭션이 성공적으로 동시에 실행될 수 있도록 지원하는 기능이다.(데이터 무결성 및 일관성 보장)
- 트랜잭션들이 동시에 접근할 때 발행할 수 있는 문제점을 예방하고 해결하는 기능이다.

// 병행제어 목적

- 데이터베이스 공유 및 시스템의 활용도를 극대화한다.
- 데이터베이스 무결성, 일관성을 유지한다.
- 사용자에 대한 **응답시간**을 최소화한다.
- 트랜잭션의 직렬가능성 보장한다.

// 인터리빙(interleaving, 끼어들기)

다음은 2개의 프로그램 P1과 P2가 **인터리빙** 방식으로 수행되는 것을 보여준다.



- 다중사용자 환경에서 시스템에 **하나의 중앙처리장치(CPU)**만 있을 때, **병행처리** 모습이다.
- 데이터베이스에서 병행제어는 **인터리빙** 방식의 실행을 기초로 하여 발전되었다.
- **인터리빙** 방식은 접근속도(응답시간) 향상에 널리 쓰이는 기법이다.
- **전산7급 시험**에서도 기본적으로 인터리빙 방식의 실행을 전제로 해서 출제되고 있다.
- **본 교재**에서도 데이터베이스의 병행제어는 인터리빙 방식의 실행을 기초로 설명한다.

2 http://cafe.daum.net/pass365(홍재연)

// 충돌(conflict)

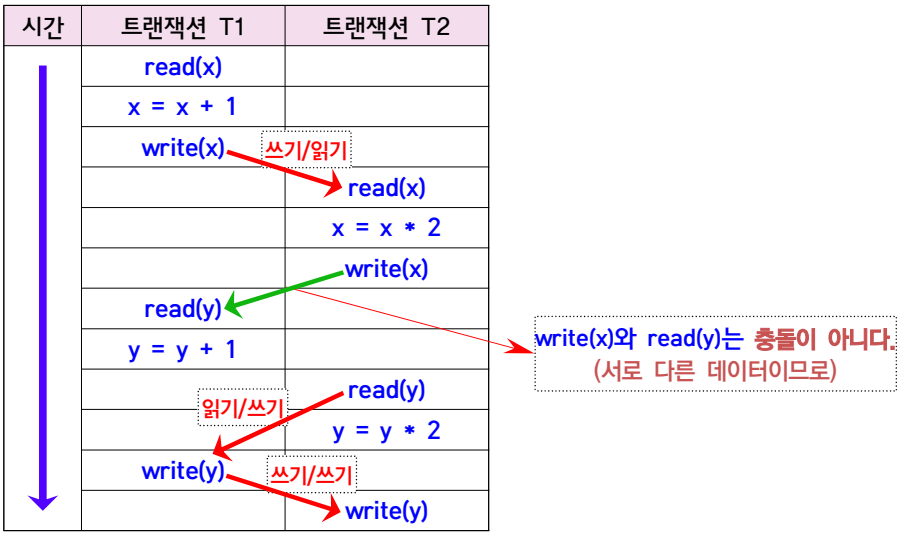
충돌은 데이터베이스 운영에서 다수의 트랜잭션이 병행 실행되면서 서로 간섭으로 발생된다.

• 다음은 트랜잭션 병행처리에서 충돌 발생 유무를 나타내고 있다.(충돌 종류는 3가지)

	트랜잭션 1	트랜잭션 2	설명
읽기-쓰기 충돌	read(a)	write(a)	2개의 연산이 각각 다른 트랜잭션에 속하면서 동일 항목에 대해 write(a) 연산이 적어도 1번 존재
쓰기-읽기 충돌	write(a)	read(a)	
쓰기-쓰기 충돌	write(a)	write(a)	
충돌 발생 안함	read(a)	read(a)	쓰기(write) 연산이 없으므로 충돌 발생 안함
충돌 발생 안함	read(a)	write(b)	쓰기(write)가 있지만, 서로 다른 항목이므로(a, b)

• 충돌은 서로 다른 트랜잭션이 동일한 데이터 항목을 연산할 때 발생하는데
• 충돌은 적어도 하나의 연산은 **쓰기(write)** 연산이 있을 때 발생한다.
• 충돌은 서로 다른 트랜잭션들 사이에 간섭으로 발생된다.(동일한 데이터 항목을 연산)
• 충돌은 데이터베이스 운영에 문제점을 유발시킬 수 있다.
• 병행제어를 통해 충돌이 발생해도 데이터베이스 운영에 문제가 없도록 제어할 수 있다.

• 다음은 트랜잭션 병행처리에서 충돌 발생을 보여준다.**(충돌 종류는 3가지)**



• 트랜잭션 병행처리에서 충돌 발생은 데이터베이스 운영에 문제점을 유발시킬 수 있다.
• 병행제어의 핵심은 충돌이 발생해도 데이터베이스 운영에 문제가 없도록 제어하는 것이다.

// 트랜잭션이 동시에 실행되면 발생될 수 있는 문제

다수의 트랜잭션이 동시에 실행될 때, 동시성제어를 하지 않으면 문제가 발생될 수 있다.
다음은 동시성제어를 하지 않는 경우(무제어)에 발생될 수 있는 문제점 들이다.

문제점	설 명
갱신분실 **(lost update)**	• 트랜잭션들이 **하나의 데이터를 동시에 갱신**할 때 발생하는 문제이다. • 어떤 트랜잭션이 갱신한 내용을 다른 트랜잭션이 덮어 쓴 경우 • 결과적으로, 이전 트랜잭션의 연산 결과가 분실된다.(갱신 무효) • 갱신분실은 **탐지 불가능**(undetectable) 문제라고도 한다.
오손판독 **(dirty read)**	• **미완료** 트랜잭션의 **중간 결과를 다른 트랜잭션이 참조**함으로써 발생 • 미완료 트랜잭션 T1이 쓴(write) 데이터를 다른 트랜잭션 T2가 읽고, 갱신 • 만약, 미완료 트랜잭션 T1이 실행 성공하면 : 문제점 없음(복귀가 불필요) • 만약, 미완료 트랜잭션 T1이 실행 실패하면 : 문제점 발생(복귀가 필요) • 트랜잭션 T2는 실행 실패한 트랜잭션 T1이 쓴 데이터를 참조한 문제점 발생 • 트랜잭션 T2는 잘못된 데이터를 참조했으므로 당연히 복귀되어야 함**(연쇄복귀)** • 그런데, T2가 이미 실행으로 완료하고 시스템을 떠나면 복귀 불가능 문제 발생 • 오손판독은 연쇄복귀(cascading rollback), 비완료 의존성 문제(uncommitted dependency problem), 임시 갱신 문제라고도 한다.
모순성 **(inconsistency)**	• 트랜잭션 수행에서 데이터를 읽는 시점에 따라 다른 값을 읽게 되는 현상 • 어떤 트랜잭션 수행 중에 다른 트랜잭션이 **끼어들어 값을 갱신**시킬 때 발생 • 그런데, **끼어들어 값을 갱신**시킨다고, 무조건 **모순**이 발생되는 것은 아니다. • 데이터베이스가 **일관성이 없는 모순된 상태**가 존재하게 되는 문제 • 데이터 모순성이 발생하면 데이터 불일치로 요약을 해도 정확하지 않게 된다. • 모순성(inconsistency)은 비반복 판독(non-repeatable read), 반복할 수 없는 읽기(unrepeatable read), 부정확한 요약(incorrect summary), 불일치 분석(inconsistent analysis) 문제라고도 한다.
유령판독 **(phantom read)**	• 특정 범위 값을 읽고 나중에 다시 읽었을 때 이전에 없었던 새로운 값이 있음 • 한 트랜잭션에서 같은 쿼리를 두 번 수행했는데, • 첫 번째 쿼리에서 없던 유령레코드가 두 번째 쿼리에서 나타나는 현상 • 트랜잭션 T1이 특정 범위의 조건으로 데이터를 검색하여 어떤 결과를 얻었다. • 이때, 트랜잭션 T2가 접근하여 해당 조건의 데이터를 일부 추가 또는 삭제함 • 그 후, 트랜잭션 T1이 같은 조건으로 데이터를 검색하면 다른 결과가 나온다. • T2가 추가/삭제한 데이터만큼 다른 결과가 나온다.**(유령처럼 보였다, 안 보였다)**

• 이런 문제가 발생하는 이유는 **충돌된 데이터를 통해 트랜잭션 사이에 간섭**이 일어나기 때문이다.
• **병행제어**를 통해 충돌이 발생해도 데이터베이스 운영에 문제가 없도록 제어할 수 있다.

기출문제 분석

1. 하나의 데이터베이스시스템 내에서 적절한 제어 없이 트랜잭션들을 동시에 실행하였을 경우 여러 문제가 발생할 수 있다. 이를 해결하기 위한 동시성제어가 올바르게 동작하지 않을 경우 발생할 수 있는 문제점으로 옳지 않은 것은? [2016년 국가 7급]

① 갱신손실 문제 ② 부정확한 요약 문제
③ 반복할 수 없는 읽기 문제 ④ 지역적 오류 문제

♤ **동시성제어에서 발생할 수 있는 문제점**

• 지역적 오류 문제는 분산 데이터베이스에서 발생할 수 있는 문제 유형이다.

// 동시성제어 정의

• 다중 사용자 환경에서 데이터베이스에 여러 트랜잭션이 성공적으로 동시에 실행될 수 있도록 지원하는 기능이다.(데이터 무결성 및 일관성 보장)
→ 트랜잭션들이 동시에 접근할 때 발행할 수 있는 문제점을 예방하고 해결하는 기능이다.

// 동시성제어가 올바르게 동작하지 않을 경우 발생할 수 있는 문제점

① 갱신손실 문제(lost update problem)

• 트랜잭션들이 하나의 데이터를 동시에 갱신할 때 발생하는 문제이다.
• 어떤 트랜잭션이 갱신한 내용을 다른 트랜잭션이 덮어 쓴 경우(갱신 무효)
• 결과적으로, 특정 트랜잭션의 연산 결과가 분실된다.

② 불일치 분석 문제(inconsistent analysis problem)

• 데이터 모순성(data inconsistency), 부정확한 요약(incorrect summary), 반복할 수 없는 읽기(unrepeatable read) 문제라고도 한다.
• 두 트랜잭션이 동시에 실행할 때 데이터베이스가 일관성 없는 모순된 상태가 되는 문제
• 데이터 모순성이 발생하여 요약을 해도 정확하지 않고, 반복적으로 읽으면 안 된다.

③ 비완료 의존성 문제(uncommitted dependency problem)

• 연쇄복귀(cascading rollback), 오손판독(dirty read), 임시 갱신 문제라고도 한다.
• 트랜잭션들이 동시에 같은 데이터에 접근하여 갱신할 때
→ 한 트랜잭션은 실행 성공하여 완료된 상태(복귀가 필요 없음)
→ 다른 트랜잭션은 실행 실패(갱신을 취소하고 원래 상태로 복귀해야 함)
• 다른 트랜잭션이 처리한 부분에 대해서는 취소가 불가능하다.(회복불능)

정답 : ④

2. <보기>의 두 트랜잭션 T1, T2를 적절한 제어 없이 동시에 수행할 때 발생할 수 있는 상황에 대한 설명으로 가장 옳지 않은 것은? [2020년 서울 7급]

----<보기>----------

T1	T2
read_item(X);	read_item(X);
X = X - 5;	X = X * 2;
read_item(Y);	write_item(X);
write_item(X);	read_item(Y);
Y = Y + 5;	read_item(X);
write_item(Y);	

---

① 두 트랜잭션이 끝난 후 Y의 값에 오류가 발생할 가능성은 없다.
② 초기 주어진 X의 값이 10이라면 T1의 read_item(X)에서 읽어 들인 X의 값은 10일 수도 있고 20일 수도 있다.
③ 초기 주어진 X의 값이 10이라면 두 트랜잭션이 끝난 후 X의 값은 10 또는 15가 될 것이다.
④ 초기 주어진 Y의 값이 10이라면 T2의 read_item(Y)에서 읽어 들인 Y의 값은 10일 수도 있고 15일 수도 있다.

♤ **병행제어**

- 충돌은 서로 다른 트랜잭션이 동일한 데이터 항목을 연산할 때 발생하는데
- 충돌은 적어도 하나의 연산은 쓰기(write) 연산이 있을 때 발생한다.
- 충돌은 서로 다른 트랜잭션들 사이에 간섭으로 발생된다.(동일한 데이터 항목을 연산)
- 충돌은 데이터베이스 운영에 문제점을 유발시킬 수 있다.
- 병행제어를 통해 충돌이 발생해도 데이터베이스 운영에 문제가 없도록 제어할 수 있다.

- 다음은 트랜잭션 병행처리에서 충돌 발생 유무를 나타내고 있다.(충돌 종류는 3가지)

	트랜잭션 1	트랜잭션 2	설명
읽기-쓰기 충돌	read(a)	write(a)	2개의 연산이 각각 다른 트랜잭션에 속하면서 동일 항목에 대해 write(a) 연산이 적어도 1번 존재
쓰기-읽기 충돌	write(a)	read(a)	
쓰기-쓰기 충돌	write(a)	write(a)	
충돌 발생 안함	read(a)	read(a)	쓰기(write) 연산이 없으므로 충돌 발생 안함
충돌 발생 안함	read(a)	write(b)	쓰기(write)가 있지만, 서로 다른 항목이므로(a, b)

// 트랜잭션 T1, T2를 적절한 제어 없이 동시에 수행할 때 발생할 수 있는 상황

① 초기 X의 값이 10일 때 : 최종 결과 X = 5가 되는 경우

T1	T2	수행 과정
read_item(X);		X = 10
X = X - 5;		X = X - 5 = 10 - 5 = 5
read_item(Y);		
	read_item(X);	X = 10
	X = X * 2;	X = X * 2 = 10 * 2 = 20
	write_item(X);	X = 20 (기록)
	read_item(Y);	
	read_item(X);	
write_item(X);		X = 5 (최종 기록)
Y = Y + 5;		
write_item(Y);		

③ 최종 결과 X = 10이 되는 경우
T1.X = 10
X = X - 5 = 10 - 5 = 5
T2.X = 5
X = X * 2 = 5 * 2 = 10

④ 최종 결과 X = 15가 되는 경우
T2.X = 10
X = X * 2 = 10 * 2 = 20
T1.X = 5
X = X - 5 = 20 - 5 = 15

② 초기 X의 값이 10일 때 : 최종 결과 X = 20이 되는 경우

T1	T2	수행 과정
read_item(X);		X = 10
	read_item(X);	X = 10
X = X - 5;		X = X - 5 = 10 - 5 = 5
	X = X * 2;	X = X * 2 = 10 * 2 = 20
read_item(Y);		
write_item(X);		X = 5 (기록)
Y = Y + 5;		
write_item(Y);		
	write_item(X);	X = 20 (최종 기록)
	read_item(Y);	
	read_item(X);	

↓
↓분석 결과
↓

③ 초기 주어진 X의 값이 10이라면 두 트랜잭션이 끝난 후 **X의 값은 10 또는 15**가 **될 것이다**.(×)
→ **X의 값은 5 또는 10 또는 15 또는 20**이 **될 수 있다**.
→ **X의 값이 10 또는 15**가 되는 경우는 간단하게 분석하였다.

정답 : ③