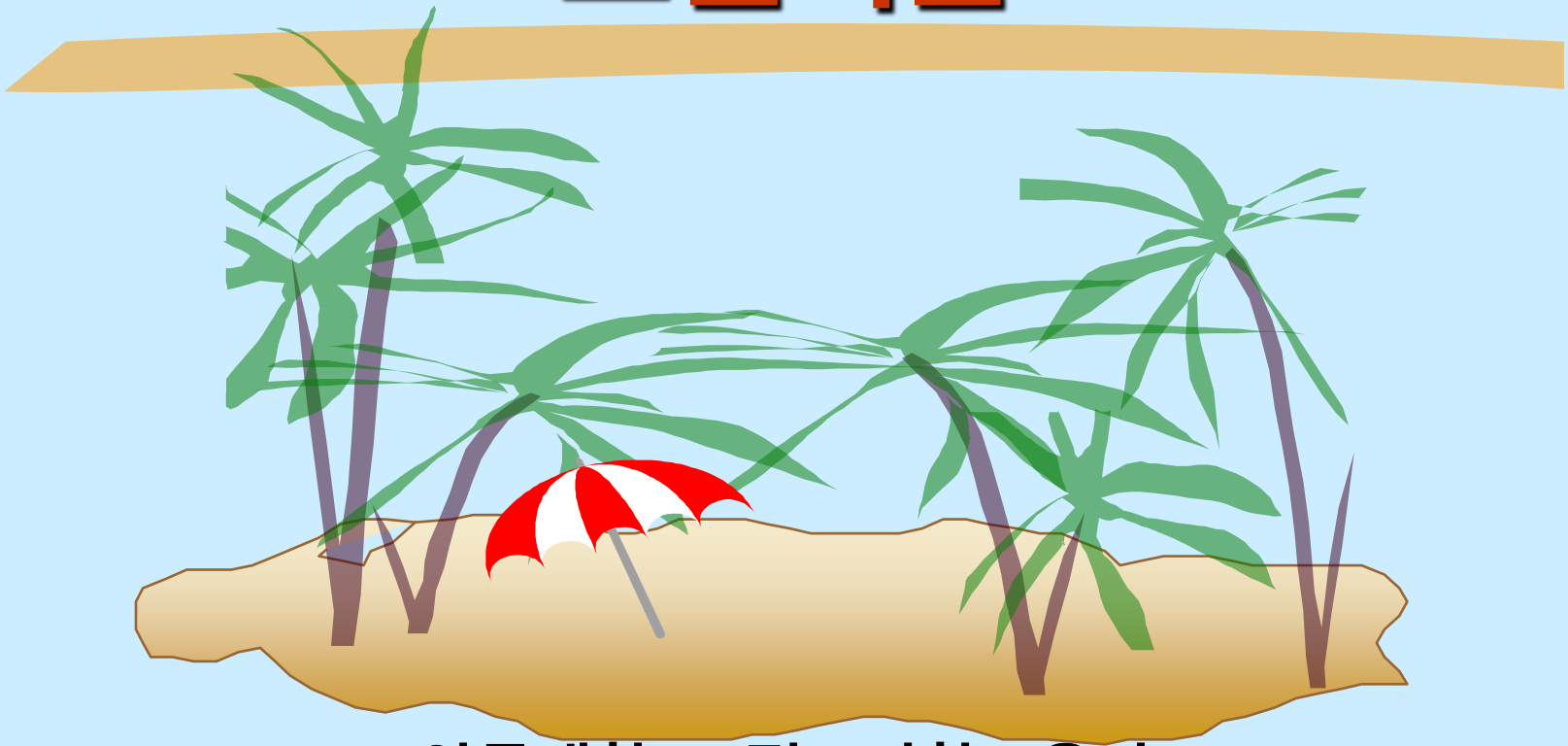
트랜잭션



안동대학교 정보과학교육과

배울 내용

- 트랜잭션의 개념, 상태
- 원자성과 지속성의 구현
- 동시 실행
- 직렬성
- 회복성
- 고립성의 구현
- SQL에서의 트랜잭션의 정의
- 직렬성 검사

트랜잭션의 개념

- 트랜잭션
  - 다양한 데이터 항목에 접근하고 변경하는 연산들의 논리적 집합체
- 트랜잭션은
  - 일관성 있는 데이터베이스를 보여야 함
  - 실행 중에 데이터베이스는 불일치 상태에 있을 수 있으나 완료 후에는 일관성을 유지해야 함
- 다루어야 할 두 가지 주요사항
  - 하드웨어 고장 및 시스템 이상과 같은 다양한 종류의 고장
  - 여러 트랜잭션의 동시 실행

ACID 속성

데이터 무결성을 보존하기 위해 데이터베이스 시스템은 다음을 보장해야 함

- 원자성
  - 트랜잭션의 모든 연산이 데이터베이스에 적절히 반영되든지 또는 반영이 되지 않든지 해야 함
- 일관성
  - 고립 상태의 트랜잭션 실행은 데이터베이스의 일관성을 보존함
- 고립성
  - 여러 트랜잭션이 동시에 실행되더라도 각 트랜잭션은 다른 트랜잭션들이 동시에 실행되고 있다는 사실을 인지해서는 안 됨
- 지속성
  - 트랜잭션이 성공적으로 완료한 후에는 시스템 고장이 발생하더라도 데이터베이스에 이루어진 변경은 영속적

자금 이체의 예제

- 계좌 A에서 B로 50불을 이체하는 트랜잭션
  1. read(A)
  2. A := A - 50
  3. write(A)
  4. read(B)
  5. B := B + 50
  6. write(B)

- 일관성 요구 사항
  - A와 B의 합계는 트랜잭션의 실행으로 변하지 않음
- 원자성 요구 사항
  - 3번 절차 이후 6번 절차 이전에 트랜잭션이 실패하면, 시스템은 갱신이 데이터베이스에 반영되지 않도록 해야 함
  - 그렇지 않으면 불일치가 발생함

자금 이체의 예제(계속)

- 지속성 요구사항
  - 트랜잭션이 완료되었다(즉, 50불의 이체가 이루어짐)고 사용자가 통지를 받으면, 트랜잭션이 수행한 데이터베이스 갱신은 고장에도 불구하고 영속적이어야 함

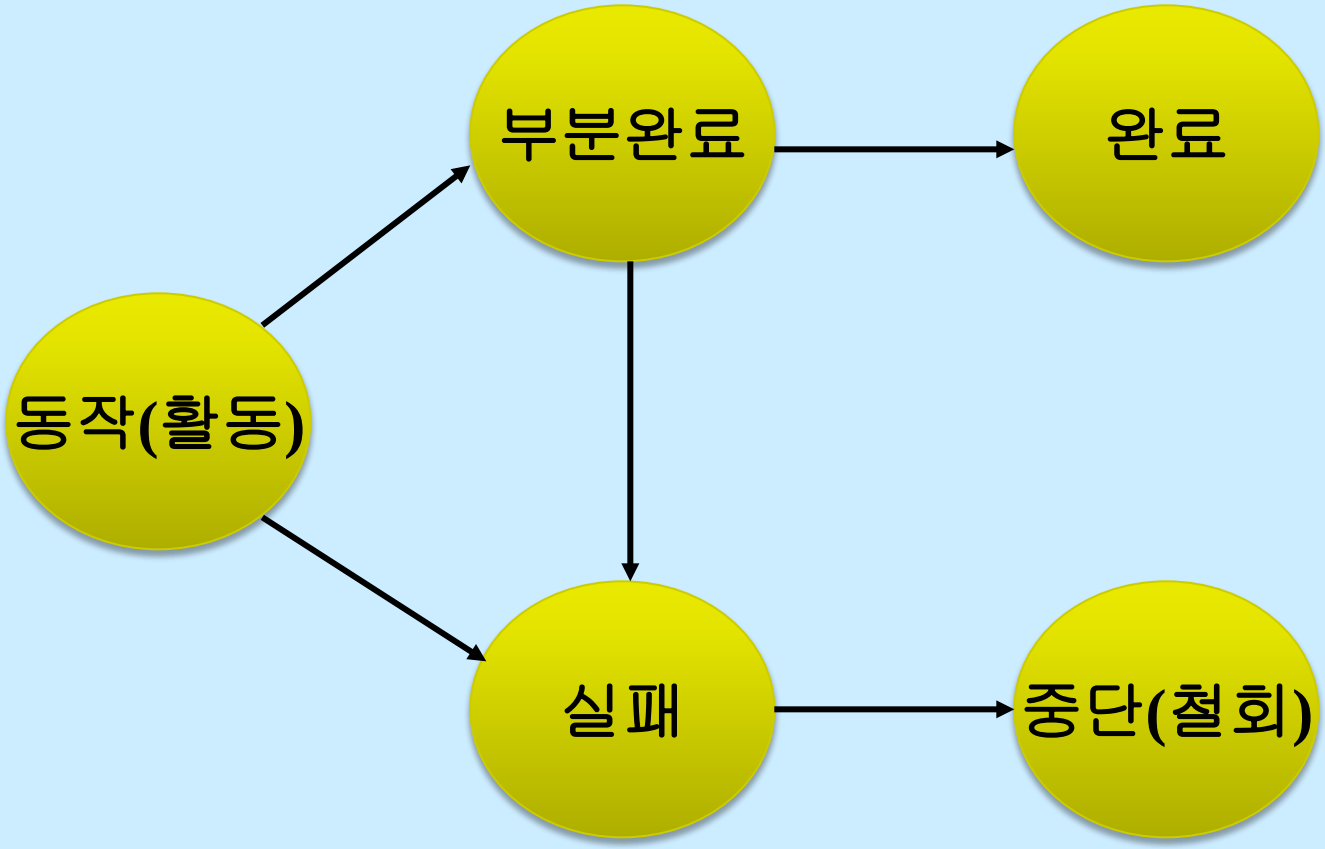
- 고립성 요구사항
  - 절차 3과 6사이에서 부분적으로 갱신된 데이터베이스에 다른 트랜잭션의 액세스를 허용하면, 데이터베이스에 불일치가 발생할 수 있음(A+B가 원래보다 적게 될 것)

트랜잭션을 순서대로(하나 끝난 후 다음) 실행시키면 당연히 보장될 수 있다. 그러나, 여러 트랜잭션을 동시에 실행시키면 상당한 이점을 가지게 됨

트랜잭션의 상태

- 동작(active)
  - 초기 상태; 트랜잭션이 실행중인 동안은 이 상태에 머뭄
- 부분완료(partially committed)
  - 마지막 명령문을 실행한 직후의 상태
- 실패(failed)
  - 정상적 실행을 더 이상 진행할 수 없음을 발견한 후 중단한 상태
- 중단(aborted)
  - 트랜잭션이 복귀되고 트랜잭션이 시작하기 이전의 데이터베이스 상태로 복구한 후, 중단된 후 두 가지 선택이 가능
    - 트랜잭션의 재 시작 – 내부 논리 오류가 없는 경우에만
    - 트랜잭션의 완전 종료
- 완료(committed)
  - 트랜잭션이 성공적으로 종료한 후

트랜잭션의 상태(계속)

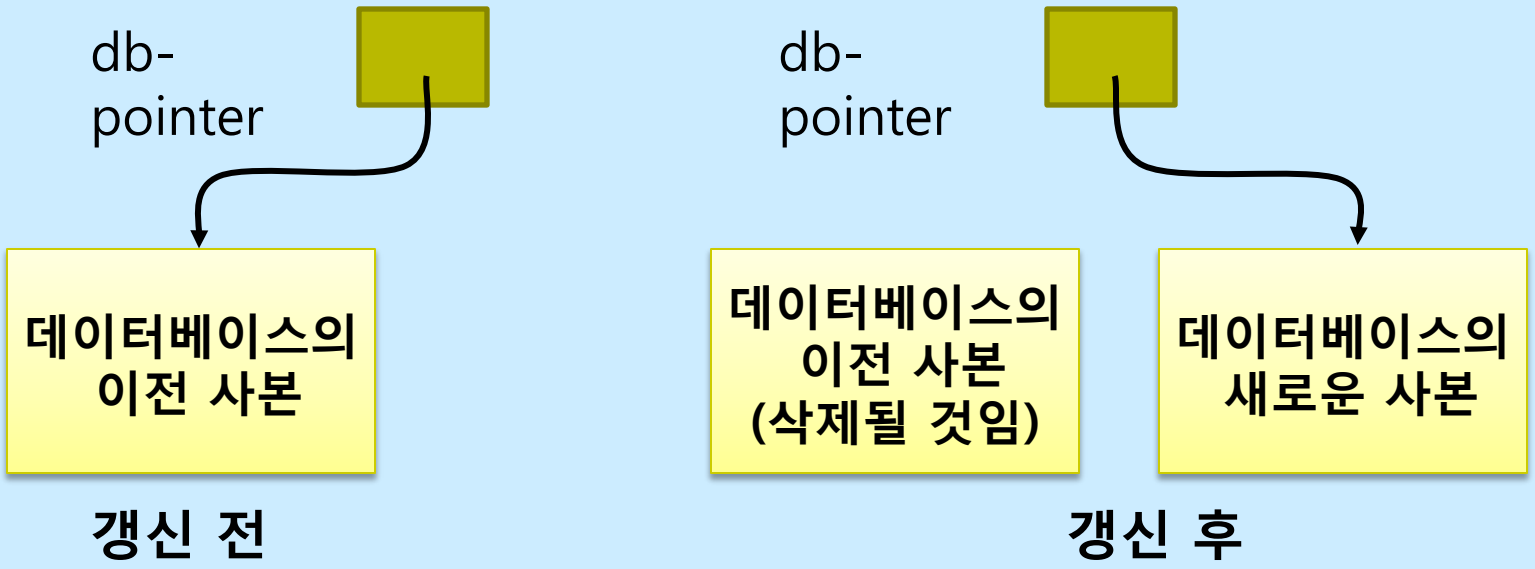


원자성과 지속성의 구현

- 데이터베이스 시스템의 회복 관리 구성 요소가 원자성과 지속성을 지원한다.
- 그림자-데이터베이스 기법
  - 한번에 하나의 트랜잭션만이 동작한다고 가정
  - db-pointer라는 포인터는 항상 데이터베이스의 현재 일관성 있는 사본을 가리킨다.
  - 모든 갱신은 데이터베이스의 그림자 사본에 이루어지며, db-pointer는 트랜잭션이 부분 완료에 도달하고 갱신된 모든 페이지가 디스크에 출력된 후에만 갱신된 그림자 사본을 가리키도록 한다.
  - 트랜잭션이 실패하는 경우, db-pointer가 가리키던 이전의 일관성 있는 사본이 사용될 수 있으며 그림자 사본은 삭제될 수 있다.

원자성과 지속성의 구현(계속)

- 그림자-데이터베이스 기법:



- 디스크는 고장이 없다고 가정
- 텍스트 에디터에는 효율적이지만, 대형 데이터베이스에는 극히 비효율적 : 단일 트랜잭션을 실행할 때도 데이터베이스 전체의 사본을 요구

동시 실행

- 여러 트랜잭션이 시스템 내에서 동시에 실행되도록 한다.

- 장점
  - 처리기와 디스크의 이용률을 증가시켜 보다 좋은 트랜잭션 처리율을 얻는다: 한 트랜잭션이 디스크 입출력을 수행하는 동안 다른 트랜잭션은 CPU를 사용할 수 있음
  - 트랜잭션에 대해 평균 응답 시간을 줄인다
    - 짧은 트랜잭션들이 긴 트랜잭션들 뒤에 기다릴 필요가 없음

- 동시성 제어 기법 : 데이터베이스의 일관성을 파괴하지 못하도록 동시 실행 트랜잭션들 간의 상호 작용을 제어하는 기법
  - 동시 실행의 정확성의 개념을 이해할 필요 있음

스케줄

- 스케줄 : 동시 실행 트랜잭션들의 명령들이 어떤 순서로 실행되는 가를 나타내는 순서
  - 트랜잭션들의 집합에 대한 스케줄에는 그들 트랜잭션의 모든 명령을 포함해야 함
  - 각 트랜잭션에 나타나는 명령들의 순서를 보존해야 함

예제 스케줄

- T_1은 계좌 A에서 B로 50불을 이체하고, T_2는 계좌 A의 잔고의 10%를 B로 이체한다 하자. 다음은 T_1이 끝난 후 T_2가 시작하는 직렬 스케쥴이다.

T_1	T_2
read(A) A := A - 50 write(A) read(B) B := B + 50 write(B)	
	read(A) temp := A * 0.1; A := A - temp write(A) read(B) B := B + temp write(B)

예제 스케줄(계속)

- T_1과 T_2는 앞의 예와 같다 하자. 다음 스케줄은 직렬 스케줄은 아니지만 스케줄1과 동등하다.

T_1	T_2
read(A)	
A := A - 50	
write(A)	
	read(A)
	temp := A * 0.1;
	A := A - temp
	write(A)
read(B)	
B := B + 50	
write(B)	
	read(B)
	B := B + temp
	write(B)

- 스케쥴1과 2에서 모두 A + B의 합계가 보존된다.

예제 스케줄(계속)

- 다음 동시 스케줄은 A + B 합계의 값을 보존하지 않는다.

T_1	T_2
read(A)	
A := A - 50	
	read(A)
	temp := A * 0.1;
	A := A - temp
	write(A)
	read(B)
write(A)	
read(B)	
B := B + 50	
write(B)	
	B := B + temp
	write(B)

직렬성

- 기본 가정 — 각 트랜잭션은 데이터베이스의 일관성을 보존
- 따라서, 트랜잭션 집합의 직렬 실행은 데이터베이스의 일관성을 보존
- 직렬 스케줄과 동등한 스케줄(동시 실행 가능)을 직렬 가능하다고 함. 서로 다른 유형의 스케줄 동등성이 다음과 같은 개념으로 발생함 :
  - 1. 충돌 직렬성
  - 2. 뷰 직렬성
- read와 write 명령 이외의 연산은 무시하며 트랜잭션은 읽고 쓰는 사이에 지역 버퍼내의 데이터에 임의의 계산을 수행한다고 가정. 예제의 단순한 스케줄은 read와 write 명령만으로 구성

충돌 직렬성

- 트랜잭션 T_i와 T_j 각각의 명령 I_i와 I_j가 충돌일 필요 충분 조건은 I_i와 I_j 모두 액세스하는 어떤 데이터 항목 Q가 존재하고 이들 명령중 적어도 하나가 Q를 쓰는 경우
  - 1. I_i = read(Q), I_j = read(Q). I_i와 I_j는 충돌이 아니다.
  - 2. I_i = read(Q), I_j = write(Q). 그들은 충돌이다.
  - 3. I_i = write(Q), I_j = read(Q). 그들은 충돌이다.
  - 4. I_i = write(Q), I_j = write(Q). 그들은 충돌이다.

- 직관적으로, I_i와 I_j간의 충돌은 그들간의 (논리적) 임시 순서를 강요한다. I_i와 I_j가 스케쥴 내에 연속해 있고 충돌이 아니면, 스케줄 내에서 그들을 교환한다 하더라도 결과는 같게 된다.

충돌 직렬성(계속)

- 스케줄 S가 비충돌 명령들의 일련의 교환으로 스케줄 S′로 변환될 수 있으면 S와 S′는 충돌 동등하다고 말한다.
- 스케줄 S가 직렬 스케쥴과 충돌 동등이면 충돌 동등 가능하다고 말한다.
- 충돌 동등 가능하지 않은 스케줄의 예

T_3	T_4
read(Q)	
	write(Q)
write(Q)	

- 상기 스케줄 내 명령들을 맞바꾸어 직렬 스케줄 $<T_3, T_4>$ 또는 $<T_4, T_3>$을 얻을 수 없다.

충돌 직렬성(계속)

- 아래의 스케줄 3은 일련의 비충돌 명령들의 맞바꿈으로 T_1 다음에 T_2가 수행되는 직렬 스케쥴 1로 변환될 수 있다. 그러므로, 스케줄 3은 충돌 직렬 가능하다.

T_1	T_2
read(A)	
write(A)	
	read(A)
	write(A)
read(B)	
write(B)	
	read(B)
	write(B)

뷰 직렬성

- S와 S′ 를 동일한 트랜잭션의 집합을 가진 스케줄이라 하자. S와 S′ 은 다음 세가지 조건을 만족하면 뷰 동등함
  1. 각 데이터 항목 Q에 대해 트랜잭션 T_i가 스케쥴 S내에서 Q의 초기값을 읽으면, 트랜잭션 T_i는 스케쥴 S′ 내에서 또한 Q의 초기값을 읽어야 함
  2. 각 데이터 항목 Q에 대해, 스케쥴 S내어서 T_i가 read(Q)를 실행하고 그 값은 트랜잭션 T_j가 생성한 것(있다면) 이라면, 스케쥴 S′내에서 트랜잭션 T_i 또한 트랜잭션 T_j가 생성한 Q의 값을 읽어야 함
  3. 각 데이터 항목 Q에 대해, 스케쥴 S내에서 마지막 write(Q) 연산을 수행한 트랜잭션(있다면)은 스케줄 S′ 내에서도 마지막 write(Q) 연산을 수행해야 함
- 뷰 동등성 또한 순순히 read와 write에만 근거

뷰 직렬성

- 스케줄 S가 직렬 스케쥴과 뷰 동등하면 뷰 직렬 가능
- 모든 충돌 직렬 가능 스케줄은 또한 뷰 직렬 가능
- 스케쥴9 — 뷰 직렬 가능하지만 충돌 직렬 가능하지 않은 스케줄

T_3	T_4	T_6
read(Q)		
	write(Q)	
write(Q)		
		write(Q)

- 충돌 직렬 가능하지 않은 모든 뷰 직렬 가능 스케줄은 맹목적 출력을 갖는다.

직렬성의 기타 개념

- 아래의 스케줄 8은 직렬 스케줄 $<T_1, T_5>$ 와 같은 결과를 생성하지만, 충돌 동등이거나 뷰 동등이지 않다.

T_1	T_5
read(A) A := A - 50 write(A)	
	read(B) B := B - 10 write(B)
read(B) B := B + 50 write(B)	
	read(A) A := A + 10 write(A)

- 그러한 동등성을 결정할 때는 read와 write 이외의 연산을 분석할 필요가 있다.

회복성

- 동시에 실행되는 트랜잭션들에서 트랜잭션 실패의 영향을 언급할 필요가 있음
  - 회복 가능 스케줄 — 트랜잭션 T_j가 먼저 쓴 데이터 항목을 트랜잭션 T_i가 읽는다면, T_i의 완료 연산이 T_j의 완료 연산 전에 나타난다.
  - 다음 스케줄(스케줄 11)은 T_9가 읽은 직후 완료하면 회복 가능하지 않다.

T_8	T_9
read(A)	
write(A)	
	read(A)
read(B)	

  - T_8이 중단되면, T_9는 불일치 상태의 데이터베이스를 읽게 되는 것이다. 그러므로, 데이터베이스는 스케줄이 회복 가능하도록 보장해야 한다.

회복성(계속)

- 연쇄 복귀 — 한 트랜잭션의 실패가 일련의 트랜잭션 복귀를 야기함. 어떤 트랜잭션도 현재 완료되지 않은 다음 스케줄을 고려해 보자(그래서 스케줄은 회복 가능)

T_{10}	T_{11}	T_{12}
read(A) read(B) write(A)		
	read(A) write(A)	
		read(A)

T_{10}이 실패하면, T_{11}과 T_{12}또한 복귀되어야 함

- 막대한 작업량의 취소를 야기할 수 있음

회복성(계속)

- 연쇄 복귀 없는 스케줄 — 연쇄 복귀는 발생할 수 없다;
- T_i가 먼저 쓴 데이터 항목을 T_j가 읽는 그러한 각 트랜잭션의 쌍 T_i와 T_j에 대해, T_i의 완료 연산은 T_j의 읽기 연산 전에 나타난다.
- 모든 연쇄 복귀 없는 스케줄은 또한 회복 가능하다.
- 연쇄 복귀 없는 스케줄로 제한하는 것이 바람직하다.

고립성의 구현

- 데이터베이스의 일관성과 또한 연쇄 복귀가 없도록 하기 위해 스케줄은 충돌 또는 뷰 직렬 가능하고 회복 가능해야 한다.
- 한번에 한 트랜잭션만이 실행할 수 있는 정책은 직렬 스케줄을 생성하지만, 동시성 정도는 낮다.
- 동시성 제어 기법들에는 허용하는 동시성의 양과 발생하는 비용 간에 손익 관계가 존재한다.
- 어떤 기법들은 충돌 직렬 가능 스케줄 만이 생성되도록 하는 반면, 다른 기법들은 충돌 직렬 가능하지 않은 뷰 직렬 가능 스케줄이 생성되도록 한다.

SQL에서의 트랜잭션 정의

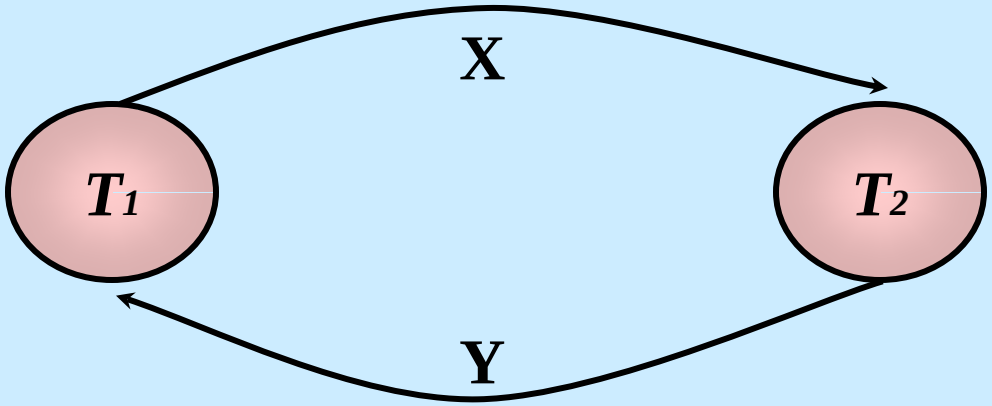
- 데이터 조작어에는 트랜잭션을 구성하는 행위 집합을 지정하기 위한 구조체를 내포해야 한다.
- SQL에서 트랜잭션은 묵시적으로 시작한다.
- SQL에서 트랜잭션은 다음 중 하나로 끝난다:
  - Commit work 현재 트랜잭션을 완료하고 새로운 트랜잭션을 시작한다.
  - Rollback work 현재 트랜잭션이 중단되도록 한다.
- SQL-92로 지정하는 동시성 단계는 다음과 같다:
  - Serializable — 기본값
  - Repeatable read
  - Read committed
  - Read uncommitted

SQL-92에서의 일관성 단계

- Serializable — 기본값
- Repeatable read — 완료된 레코드만을 읽을 수 있으며, 같은 레코드를 반복해 읽어도 같은 값을 반환해야 한다. 그러나, 트랜잭션이 직렬 가능하지 않을 수 있다 - 어떤 트랜잭션이 삽입한 레코드는 찾을 수 있지만 다른 레코드는 찾을 수 없다. 완료된 레코드만을 읽을 수 있지만, 어떤 레코드를 연속해 읽으면 다른 값을 (완료되었지만) 반환한다.
- Read uncommitted — 미완료 레코드 또한 읽을 수 있다.
- 데이터베이스에 관한 대략적인 정보(질의 최적기를 위한 통계 정보 등)를 수집하는 경우에는 동시성의 정도가 낮은 것이 유용하다.

직렬성 검사

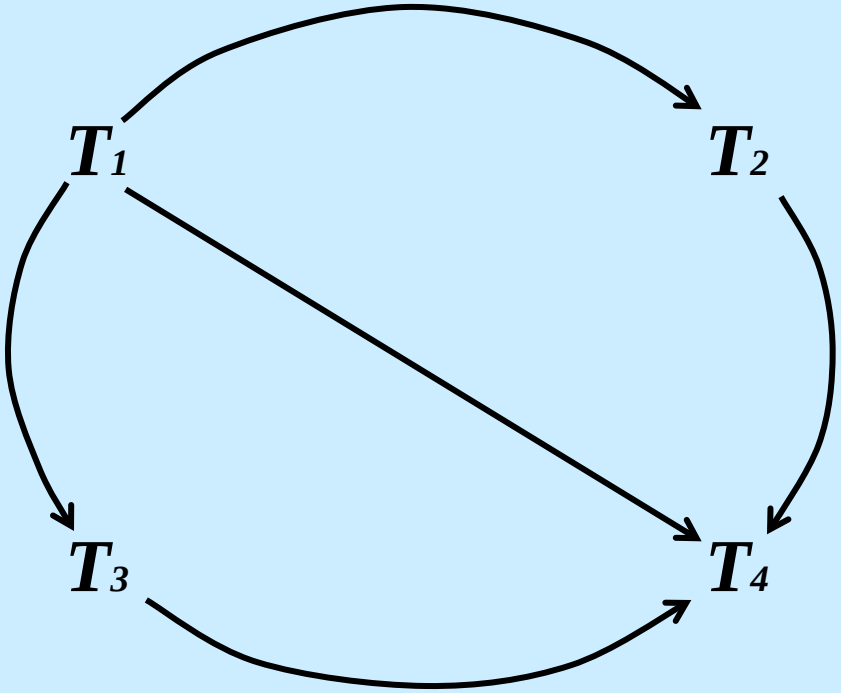
- 트랜잭션의 집합 T_1, T_2, ..., T_n의 어떤 스케쥴을 고려해 보자.
- 선행 그래프 — 정점이 트랜잭션(명)인 방향성 그래프
- 두 트랜잭션이 충돌이고 충돌이 앞에서 발생한 데이터 항목에 T_i가 액세스하면 T_i에서 T_j로 호를 그린다.
- 호 위에 액세스한 항목 이름을 붙일 수 있다.
- 예 1



예제 스케줄(스케줄 A)

T_1	T_2	T_3	T_4	T_5
	read(X)			
read(Y)				
read(Z)				
				read(V)
				read(W)
				write(W)
	read(Y)			
	write(Y)			
		write(Z)		
read(U)				
			read(Y)	
			write(Y)	
			read(Z)	
			write(Z)	
read(U)				
write(U)				

스케줄 A에 대한 선행 그래프



T5

충돌 직렬성 검사

- 스케줄이 충돌 직렬 가능인 필요 충분 조건은 그의 선행 그래프가 비순환 일 때이다.
- n^2 차수의 시간이 걸리는 순환 탐지 알고리즘(n은 그래프 내 정점의 수)이 존재한다(보다 나은 알고리즘은 n + e 시간이 걸린다. e는 간선의 수).
- 선행 그래프가 비순환이면, 그래프의 위상 정렬로 직렬성 순서를 얻을 수 있다. 이것은 그래프의 부분 순서와 일치하는 선형 순서이다. 예를 들어, A에 대한 직렬성 순서는 $T_5 \rightarrow T_1 \rightarrow T_3 \rightarrow T_2 \rightarrow T_4$ 이다.

뷰 직렬성 검사

- 충돌 직렬성 검사를 위한 선행 그래프는 뷰 직렬성 검사를 위해 수정해야 한다.

- 라벨 선행 그래프를 구축한다. 0이 아닌 같은 라벨을 가진 모든 간선의 쌍으로부터 하나의 간선을 선택해 라벨 선행 그래프에서 추출한 비순환 그래프를 찾는다. 스케줄이 뷰 직렬 가능한 필요충분 조건은 그러한 비순환 그래프가 있는 경우이다.

- 그러한 비순환 그래프를 찾는 문제는 NP-complete 문제에 속한다. 따라서, 효율적인 알고리즘은 존재하지 않는다.

- 그러나, 뷰 직렬성을 위한 어떤 충분을 조건만을 검사하는 실질적인 알고리즘은 여전히 사용될 수 있다.

동시성 제어와 직렬성 검사

- 스케줄이 실행된 후 직렬성을 검사하는 것은 너무 늦다.
- 목표 — 직렬성을 보장할 동시성 제어 규약을 개발하는 것. 규약은 일반적으로 선행 그래프가 생성될 때 검사하지 않고, 비직렬 가능 스케줄을 회피하는 원칙을 부과한다.
- 직렬성 검사는 동시성 제어 규약이 왜 정확한지의 이해를 돕는다.

요약

- 트랜잭션의 개념, 상태
- 원자성과 지속성의 구현
- 동시 실행
- 직렬성
- 회복성
- 고립성의 구현
- SQL에서의 트랜잭션의 정의
- 직렬성 검사

다음 배울 내용 : 동시성 제어 기법