데이터베이스 실습 13주차

! 기말고사 ! -> 정규화가 무엇인지, 트랜잭션이 무엇인지

정규화 : 함수적 종속성을 이용해, 릴레이션을 연관성이 있는 속성들로만 구성되도록 분해하여 이상 현상이 발생하지 않는 올바른 릴레이션으로 만들어 나가는 과정

데이터 안에는 중복과 종속성이 생길 수 밖에 없다. (이상 현상)

이상현상 – 삽입, 삭제, 갱신에서 발생

1. 필요하지 않은 필드는 빼기
2. 데이터베이스를 설계할 때 하나의 필드로 primary key 로 설정 (복합으로 만들지 말기)

정규형 (NF) : 릴레이션이 정규화된 정도

각 정규형마다 제약조건이 존재 (6가지의 제약조건)

현재 테이블에서 발생하는 정규형 무엇이고 어떤 이상이 있는지 설명 -> PPT 참고

제1정규형 (1NF) : 테이블의 모든 속성이 더는 분해되지 않는 원자 값만 가지면 제 1정규형을 만족

제1정규형을 만족하는 릴레이션

고객아이디	이벤트번호	당첨여부	등급	할인율
apple	E001	Y	gold	10%
apple	E005	N	gold	10%
apple	E010	Y	gold	10%
banana	E002	N	vip	20%
banana	E005	Y	vip	20%
carrot	E003	Y	gold	10%
carrot	E007	Y	gold	10%
orange	E004	N	silver	5%

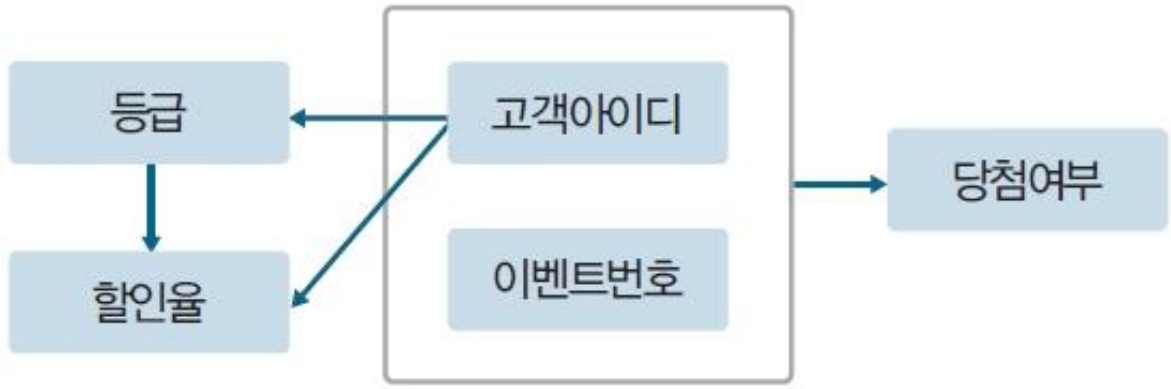
함수종속자 표현

고객아이디 → 등급

고객아이디 → 할인율

등급 → 할인율

{고객아이디, 이벤트번호} → 당첨여부



이상 현상 발생 이유? 기본키에 완전 함수 종속되지 못한 등급과 할인율 때문

고객아이디	이벤트번호	당첨여부	등급	할인율	
apple	E001	Y	vip	10%	
apple	E005	N	vip	10%	← 데이터 불일치로 인한 갱신 이상
apple	E010	Y	gold	10%	
banana	E002	N	vip	20%	
banana	E005	Y	vip	20%	
carrot	E003	Y	gold	10%	
carrot	E007	Y	gold	10%	
~~orange~~	~~E004~~	~~N~~	~~silver~~	~~5%~~	← 데이터 손실로 인한 삭제 이상
grape	NULL	NULL	silver	5%	← 삽입 불가로 인한 삽입 이상

제2정규형 (2NF) : 어떤 테이블이 제1정규형에 속하고, 기본키가 아닌 모든 속성이 기본키에 완전 함수 종속되면 제2정규형을 만족

제2정규형을 만족하는 릴레이션

분해 전의 이벤트참여 릴레이션

고객아이디	이벤트번호	당첨여부	등급	할인율
apple	E001	Y	gold	10%
apple	E005	N	gold	10%
apple	E010	Y	gold	10%
banana	E002	N	vip	20%
banana	E005	Y	vip	20%
carrot	E003	Y	gold	10%
carrot	E007	Y	gold	10%
orange	E004	N	silver	5%

부분 함수 종속을 제거하려고 분해

고객 릴레이션

고객아이디	등급	할인율
apple	gold	10%
banana	vip	20%
carrot	gold	10%
orange	silver	5%

이벤트참여 릴레이션

고객아이디	이벤트번호	당첨여부
apple	E001	Y
apple	E005	N
apple	E010	Y
banana	E002	N
banana	E005	Y
carrot	E003	Y
carrot	E007	Y
orange	E004	N

이상 현상 발생 이유? -> 이행적 함수 종속이 존재하기 때문

이행적 함수 종속 : 테이블을 구성하는 3개의 속성 집합 X, Y, Z에 대해 함수 종속 관계 X → Y와 Y → Z가 존재하면 논리적으로 X → Z가 성립되는데, 이때 Z가 X에 이행적으로 함수 종속되었다고 함

고객아이디	등급	할인율
apple	gold	15%
banana	vip	20%
carrot	gold	10%
orange	silver	5%
NULL	bronze	1%

← 데이터 불일치로 인한 갱신 이상

← 데이터 손실로 인한 삭제 이상

← 삽입 불가로 인한 삽입 이상

제3정규형 (3NF) : 릴레이션이 제2정규형에 속하고, 기본키가 아닌 모든 속성이 기본키에 이행적 함수 종속이 되지 않으면 제3정규형을 만족

제3저규형을 만족한는 릴레이션

분해 전의 고객 릴레이션

고객아이디	등급	할인율
apple	gold	10%
banana	vip	20%
carrot	gold	10%
orange	silver	5%

이행적 함수 종속을 제거하려고 분해

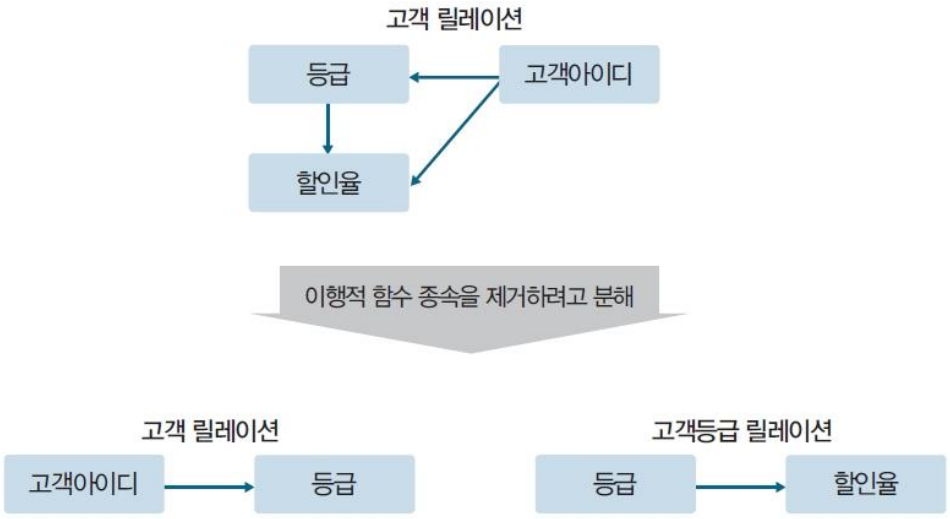
고객 릴레이션

고객아이디	등급
apple	gold
banana	vip
carrot	gold
orange	silver

고객등급 릴레이션

등급	할인율
gold	10%
vip	20%
silver	5%

테이블의 함수 종속 관계 (다이어그램)



보이스/코드 정규형 (BCNF) : 하나의 테이블에 후보키를 여러 개 가지고 있는 경우 이상 현상이 발생할 수 있다. -> 해결하기 위해 제3정규형보다 좀 더 엄격한 제약조건을 제시

제3정규형을 만족하지만 보이스/코드 정규형을 만족하지 않는 릴레이션 -> 함수 종속 관계에서 모든 결정자가 후보키가 아니기 때문

고객아이디	인터넷강좌	담당강사번호
apple	영어회화	P001
banana	기초토익	P002
carrot	영어회화	P001
carrot	기초토익	P004
orange	영어회화	P003
orange	기초토익	P004

BCNF를 만족하는 릴레이션

강좌신청 릴레이션

고객아이디	인터넷강좌	담당강사번호
apple	영어회화	P001
banana	기초토익	P002
carrot	영어회화	P001
carrot	기초토익	P004
orange	영어회화	P003
orange	기초토익	P004

후보키가 아닌 결정자를 제거하려고 분해

고객담당강사 릴레이션

고객아이디	담당강사번호
apple	P001
banana	P002
carrot	P001
carrot	P004
orange	P003
orange	P004

강좌담당 릴레이션

담당강사번호	인터넷강좌
P001	영어회화
P002	기초토익
P003	영어회화
P004	기초토익

이상 현상의 발생 이유는? 담당강사번호가 후보키가 아님에도 인터넷강좌 속성을 결정하기 때문

고객아이디	인터넷강좌	담당강사번호
apple	영어회화	P001
~~banana~~	~~기초토익~~	~~P002~~
carrot	영어회화	P001
carrot	중급토익	P004
orange	영어회화	P003
orange	기초토익	P004
NULL	중급토익	P005

제4정규형, 제5정규형은 정의만 외우기

제4정규형 : 릴레이션이 보이스/코드 정규형을 만족하면서, 함수 종속이 아닌 다치 종속을 제거하면 제4정규형에 속함

제5정규형 : 릴레이션이 제4정규형을 만족하면서, 후보키를 통하지 않는 조인 종속 (JD; Join Dependency)을 제거하면 제5정규형에 속함

정규화 과정

! 시험 ! 트랜잭션의 특징, 알고리즘, 연산, 상태에 대해 설명

트랜잭션 : 하나의 작업을 수행하는데 필요한 데이터베이스 연산들을 모아놓는 것

트랜잭션 특성

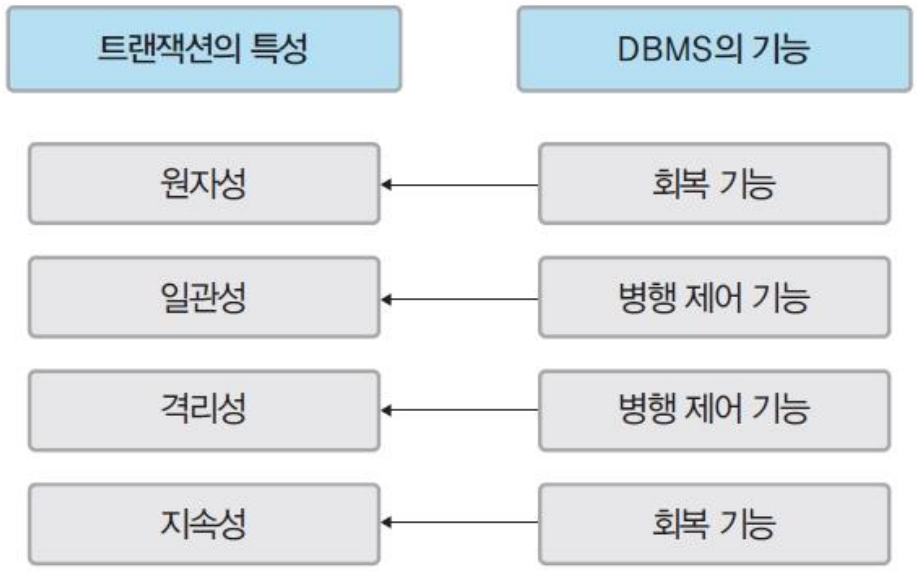
원자성 : 트랜잭션의 연산들이 모두 정상적으로 실행되거나 하나도 실행되지 않아야 하는 all-or-nothing 방식을 의미

일관성 : 트랜잭션이 성공적으로 수행된 후에도 데이터베이스가 일관된 상태를 유지해야 함을 의미

격리성 : 수행 중인 트랜잭션이 완료될 때까지 중간 연산 결과에 다른 트랜잭션들이 접근할 수 없음을 의미

지속성 : 트랜잭션이 성공적으로 완료된 후 데이터베이스에 반영한 수행 결과는 영구적이어야 함을 의미

트랜잭션의 4가지 특성을 보장하기 위해 필요한 기능



트랜잭션의 연산

commit 연산 : 트랜잭션의 수행이 성공적으로 완료되었음을 선언하는 연산

rollback 연산 : 트랜잭션의 수행이 실패했음을 선언하는 연산

트랜잭션의 상태

활동(active) 상태 : 트랜잭션이 수행되기 시작하여 현재 수행 중인 상태

부분 완료(partially committed) 상태 : 트랜잭션의 마지막 연산이 실행을 끝낸 직후의 상태

완료(committed) 상태 : 트랜잭션이 성공적으로 완료되어 commit 연산을 실행한 상태

실패(failed) 상태 : 장애가 발생하여 트랜잭션의 수행이 중단된 상태

철회(aborted) 상태 : 트랜잭션의 수행 실패로 rollback 연산을 실행한 상태

장애 : 시스템이 제대로 동작하지 않는 상태

유형	설명	
트랜잭션 장애	의미	트랜잭션 수행 중 오류가 발생하여 정상적으로 수행을 계속할 수 없는 상태
	원인	트랜잭션의 논리적 오류, 잘못된 데이터 입력, 시스템 자원의 과다 사용 요구, 처리 대상 데이터의 부재 등
시스템 장애	의미	하드웨어의 결함으로 정상적으로 수행을 계속할 수 없는 상태
	원인	하드웨어 이상으로 메인 메모리에 저장된 정보가 손실되거나 교착 상태가 발생한 경우 등
미디어 장애	의미	디스크 장치의 결함으로 디스크에 저장된 데이터베이스의 일부 혹은 전체가 손상된 상태
	원인	디스크 헤드의 손상이나 고장 등

디스크와 메인 메모리 간의 데이터 이동 연산 : input / output

메인 메모리와 프로그램 변수 간의 데이터 이동 연산 : read / write

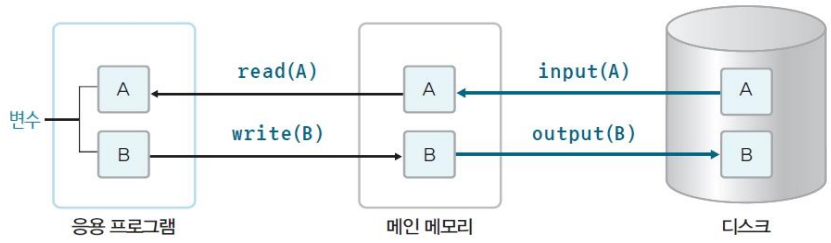


그림 10-18 응용 프로그램이 실행한 트랜잭션의 수행을 위해 필요한 데이터 이동 연산

회복 : 장애가 발생했을 때 데이터베이스를 장애가 발생하기 전의 일관된 상태로 복구시키는 것

회복을 위해 데이터베이스 복사본을 만드는 방법

덤프 (dump)	데이터베이스 전체를 다른 저장 장치에 주기적으로 복사하는 방법
로그 (log)	데이터베이스에서 변경 연산이 실행될 때마다 데이터를 변경하기 이전 값과 변경한 이후의 값을 별도의 파일에 기록하는 방법

회복을 위한 기본 연산

redo (재실행)	가장 최근에 저장한 데이터베이스 복사본을 가져온 후 로그를 이용해 복사본이 만들어진 이후에 실행된 모든 변경 연산을 재실행하여 장애가 발생하기 직전의 데이터베이스 상태로 복구 (전반적으로 손상된 경우에 주로 사용)
undo (취소)	로그를 이용해 지금까지 실행된 모든 변경 연산을 취소하여 데이터베이스를 원래의 상태로 복구 (변경 중이었거나 이미 변경된 내용만 신뢰성을 잃은 경우에 주로 사용)

로그 파일 : 데이터를 변경하기 이전의 값과 변경한 이후의 값을 기록한 파일

병행 수행(concurrency) : 여러 사용자가 데이터베이스를 동시 공유할 수 있도록 여러 개의 트랜잭션을 동시에 수행하는 것을 의미

병행 제어(concurrency control) 또는 동시성 제어 : 병행 수행 시 같은 데이터에 접근하여 연산을 실행해도 문제가 발생하지 않고 정확한 수행 결과를 얻을 수 있도록 트랜잭션의 수행을 제어하는 것을 의미

병행 수행 시 발생할 수 있는 문제점

갱신 분실(lost update) : 하나의 트랜잭션이 수행한 데이터 변경 연산의 결과를 다른 트랜잭션이 덮어써 변경 연산이 무효화되는 것

모순성(inconsistency) : 하나의 트랜잭션이 여러 개 데이터 변경 연산을 실행할 때 일관성 없는 상태의 데이터베이스에서 데이터를 가져와 연산함으로써 모순된 결과가 발생하는 것

연쇄 복귀(cascading rollback) : 트랜잭션이 완료되기 전 장애 발생으로 rollback 연산을 수행하면, 장애 발생 전에 이 트랜잭션이 변경한 데이터를 가져가 변경 연산을 실행한 다른 트랜잭션에도 rollback 연산을 연쇄적으로 실행해야 한다는 것

트랜잭션 스케줄 : 트랜잭션에 포함되어 있는 연산들을 수행하는 순서

트랜잭션 스케줄	의미
직렬 스케줄	인터리빙 방식을 이용하지 않고 트랜잭션별로 연산들을 순차적으로 실행시키는 것
비직렬 스케줄	인터리빙 방식을 이용하여 트랜잭션들을 병행해서 수행시키는 것
직렬 가능 스케줄	직렬 스케줄과 같이 정확한 결과를 생성하는 비직렬 스케줄

비직렬 스케줄에 따라 병행 수행하면 갱신 분실, 모순성, 연쇄 복귀 등의 문제가 발생할 수 있어 결과의 정확성을 보장할 수 없음

병행 제어 기법 : 병행 수행하면서도 직렬 가능성을 보장하기 위한 기법

로킹 기법 : 한 트랜잭션이 먼저 접근한 데이터에 대한 연산을 끝낼 때까지는 다른 트랜잭션이 그 데이터에 접근하지 못하도록 상호 배제(mutual exclusion)함

공용, 전용 lock

연산	설명
공용shared lock	트랜잭션이 데이터에 대해 공용 lock 연산을 실행하면, 해당 데이터에 read 연산을 실행할 수 있지만 write 연산은 실행할 수 없다. 그리고 해당 데이터에 다른 트랜잭션도 공용 lock 연산을 동시에 실행할 수 있다. (데이터에 대한 사용권을 여러 트랜잭션이 함께 가질 수 있음)
전용exclusive lock	트랜잭션이 데이터에 전용 lock 연산을 실행하면 해당 데이터에 read 연산과 write 연산을 모두 실행할 수 있다. 그러나 해당 데이터에 다른 트랜잭션은 공용이든 전용이든 어떤 lock 연산도 실행할 수 없다. (전용 lock 연산을 실행한 트랜잭션만 해당 데이터에 대한 독점권을 가질 수 있음)

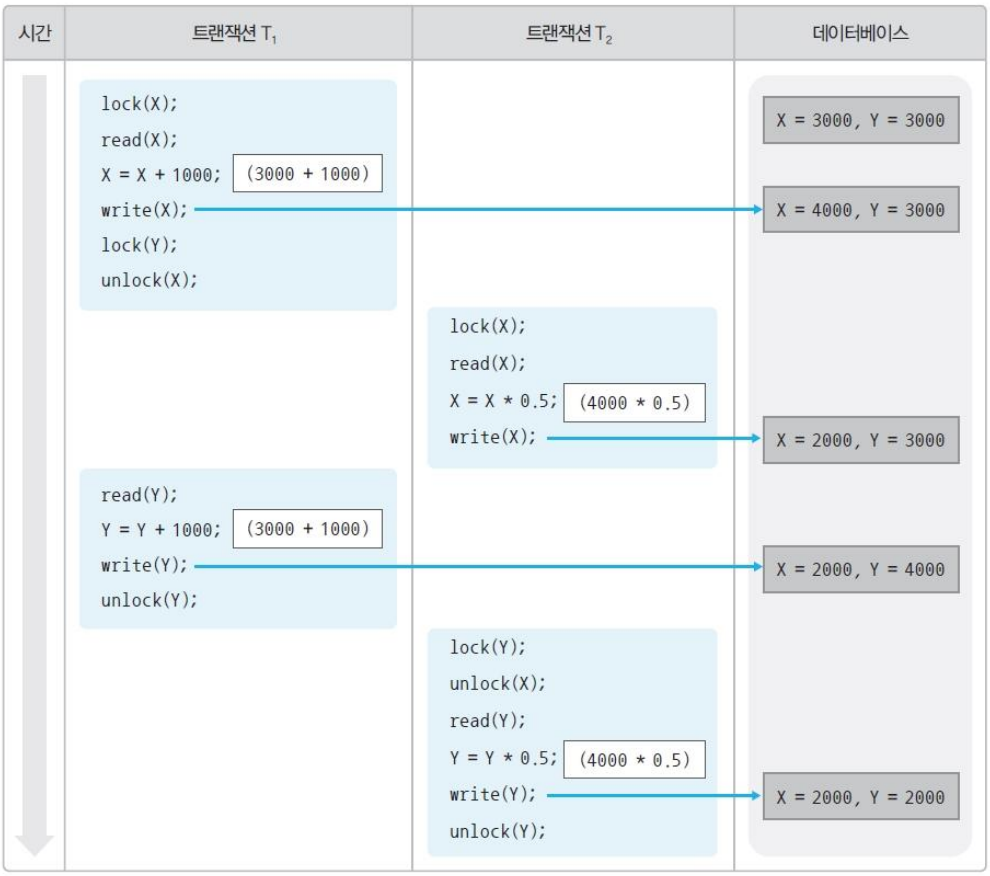
! 기말고사 ! 2단계 로킹 규약이 무엇인지

2단계 로킹 규약 : 기본 로킹 규약의 문제를 해결하고 트랜잭션의 직렬 가능성을 보장하기 위해 lock과 unlock 연산의 수행 시점에 대한 새로운 규약을 추가한 것

방법 : 트랜잭션이 lock과 unlock 연산을 확장 단계와 축소 단계로 나누어 실행

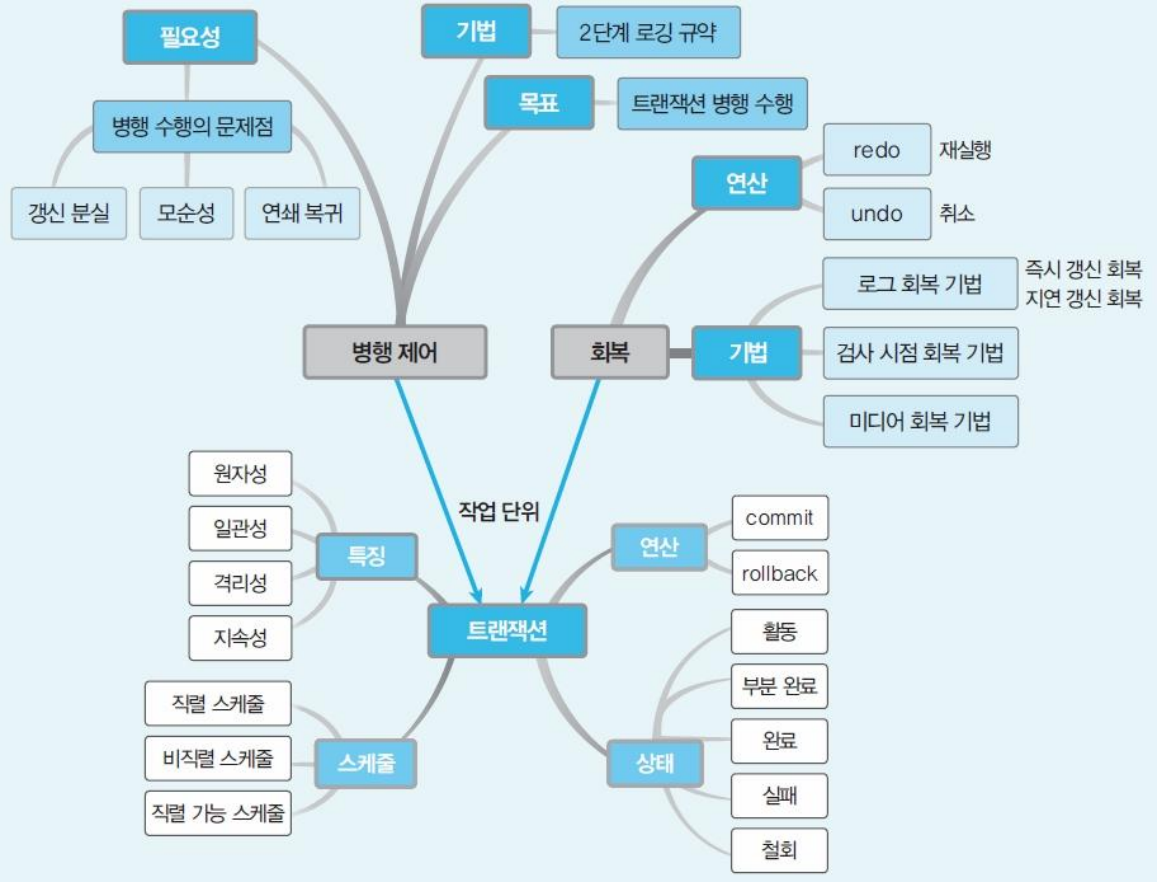
확장 단계	트랜잭션이 lock 연산만 실행할 수 있고, unlock 연산은 실행할 수 없는 단계
축소 단계	트랜잭션이 unlock 연산만 실행할 수 있고, lock 연산은 실행할 수 없는 단계

2단계 로킹 규약을 준수하여 직렬 가능성이 보장된 스케줄 예



교착 상태(deadlock) : 트랜잭션들이 상대가 독점하고 있는 데이터에 unlock 연산이 실행되기를 서로 기다리면서 수행을 중단하고 있는 상태

표위주로 시험 낼 예정!


필요성
병행 수행의 문제점
갱신 분실
모순성
연쇄 복귀
기법
2단계 로깅 규약
목표
트랜잭션 병행 수행
병행 제어
회복
연산
redo
재실행
undo
취소
기법
로그 회복 기법
즉시 갱신 회복
지연 갱신 회복
검사 시점 회복 기법
미디어 회복 기법
작업 단위
원자성
일관성
격리성
지속성
특징
트랜잭션
스케줄
직렬 스케줄
비직렬 스케줄
직렬 가능 스케줄
연산
commit
rollback
상태
활동
부분 완료
완료
실패
철회