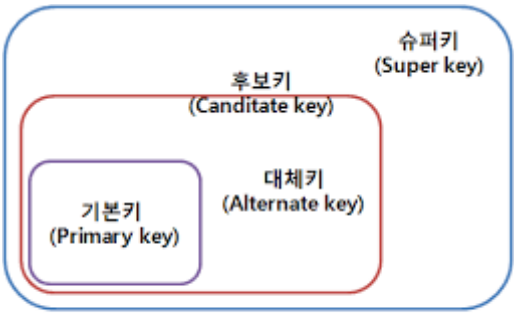
키(Key)

키는 데이터베이스에서 조건에 만족하는 튜플을 찾거나 순서대로 정렬할 때 튜플들을 서로 구분할 수 있는 기준이 되는 애트리뷰트를 말한다.

키의 종류	설명
슈퍼키	- 슈퍼키를 릴레이션을 구성하는 모든 튜플에 대해 유일성을 만족시키지만, 최소성은 만족시키지 못한다.
후보키	- 후보키는 릴레이션을 구성하는 속성들 중에서 튜플을 유일하게 식별하기 위해 사용하는 속성들의 부분집합, 즉 기본키로 사용할 수 있는 속성들을 말한다. - 유일성과 최소성을 만족한다.
대체키	- 후보키가 둘 이상일 때 기본키를 제외한 나머지 후보키들을 말한다. - 보조키라도고 한다.
기본키	- 기본키는 후보키 중에서 선택한 주키(Main Key)이다. - Null값을 가질 수 없고, 동일한 값이 중복되어 저장될 수 없다.
외래키	- 관계를 맺고 있는 릴레이션 R1, R2에서 릴레이션 R1이 참조하고 있는 릴레이션 R2의 기본키와 같은 R1 릴레이션의 속성을 외래키라고 한다.



후보키=기본키+대체키
기본키=후보키-대체키
슈퍼키 > 후보키 > 대체키 > 기본키

NATURAL JOIN(자연 조인)

- 두 테이블간의 JOIN 조건에서 동일한 이름을 갖는 모든 칼럼들에 대해 EQUI JOIN을 수행함.

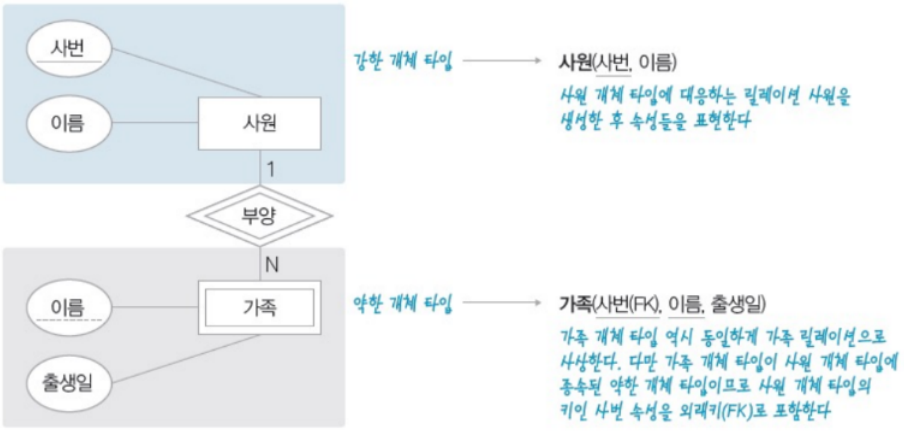
INNER JOIN(내부 조인)

- JOIN 조건에서 동일한 값이 있는 행만 반환

OUTER JOIN(외부 조인)

- Outer Join은 공통된 부분만 결합하는 Inner Join과 다르게 공통되지 않은 row도 유지한다.
- 이 때, 왼쪽 테이블의 row를 유지하면 Left Outer Join
- 오른쪽 테이블의 row를 유지하면 Right Outer Join
- 양쪽 테이블의 row를 모두 유지하면 Full Outer Join

개체 타입의 사상



1단계 : 강한 개체 타입

- 사원 개체 타입에 대응하는 릴레이션 사원을 생성한 후 속성들을 표현한다.
- 기본키와 외래키는 PK, FK로 표시

사원(사번, 이름)

2단계 : 약한 개체 타입

- 자신의 키와 함께, 강한 개체 타입의 키를 외래키로 사상해 자신의 기본키를 구성한다.

가족(사번(FK), 이름, 출생일)

관계 타입의 사상

3단계 : 이진 1:1 관계 타입

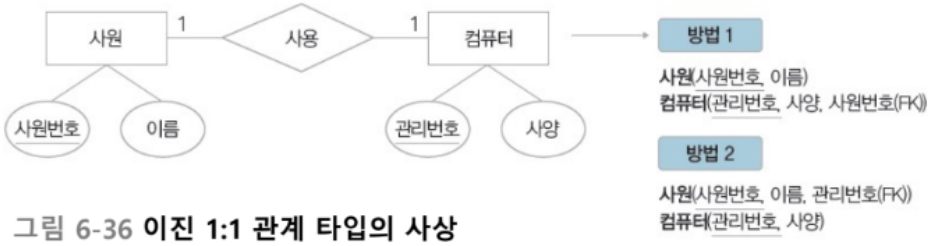


그림 6-36 이진 1:1 관계 타입의 사상

- 개체가 가진 정보 유형에 따라 판단
- 외래키에 NULL이 덜 발생하는 방법을 사용한다.

4단계 : 이진 1:N 관계 타입

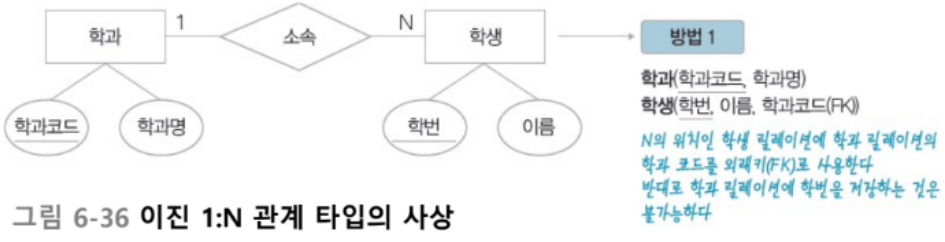


그림 6-36 이진 1:N 관계 타입의 사상

- N의 위치에 따라 방법1, 방법2 유형으로 사상

5단계 : 이진 M:N 관계 타입

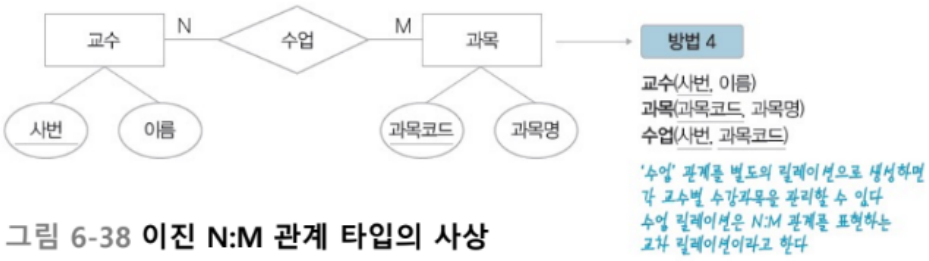
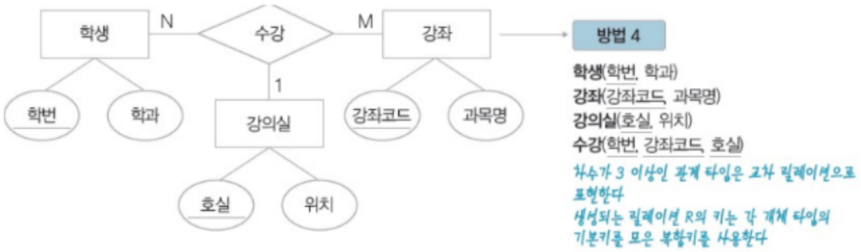


그림 6-38 이진 N:M 관계 타입의 사상

- 방법 4로 사상된다.
- 교수, 과목을 분류하고 R(관계)에는 두 개체의 키를 속성으로 만들면 된다.

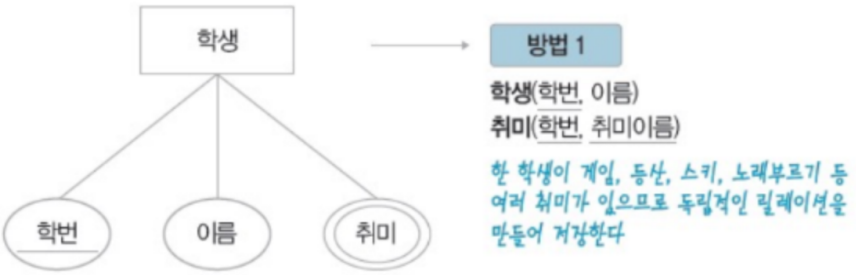
교수(사번,이름)
과목(과목코드, 과목명)
수업(사번, 과목코드)

6단계 : N진 관계 타입



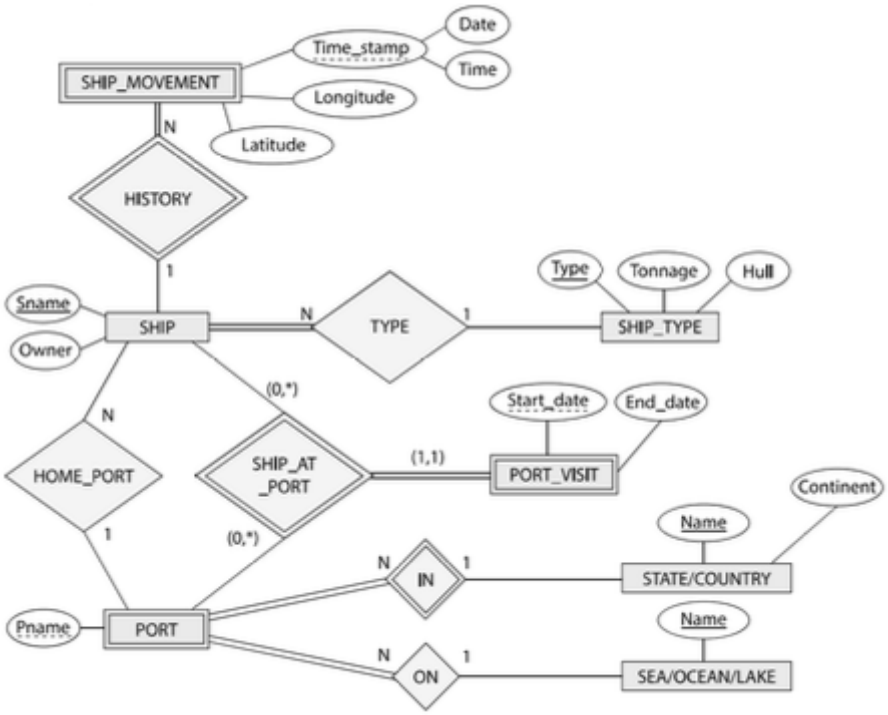
- 차수 3 이상인 다진 관계 타입일 때, 방법 4로 사상
강좌(강좌코드, 과목명)
학생(학번, 학과)
강의실(호실, 위치)
수강(학번, 강좌코드, 호실)

7단계 : 다중값 속성



- 속성의 개수를 알 수 없을 때 방법1
- 속성의 개수가 제한적으로 정해질 때 방법2
학생(학번, 이름)
취미(학번, 취미이름)

문제 : 아래 스키마를 릴레이션으로 사상하시오.



좋은 DB 스키마 설계 목적
- 정보의 중복과 갱신 이상 방지
- 정보의 손실 방지
- 실세계 표현
- 애트리뷰트들 간의 관계 명확하게 표현
- 무결성 제약조건의 시행을 간단하게 표현
- 갱신 이상이 발생하지 않도록 노력 후 효율성 고려

갱신 이상

종류	설명
삽입 이상	불필요한 정보를 함께 저장하지 않고 원하는 정보도 저장 불가능
삭제 이상	유용한 정보를 함께 삭제하지 않고는 어떤 정보도 삭제가 불가능
수정 이상	반복된 데이터 중 일부만 수정시 데이터의 불일치 발생

나쁜 설계 예시

사원

사원이름	사원번호	주소	전화번호	부서번호1	부서이름1	부서번호2	부서이름2
김창섭	2106	우이동	726-5869	1	영업	2	기획
박영권	3426	사당동	842-4538	3	개발	∧	∧
이수민	3011	역삼동	579-4685	2	기획	3	개발

[그림 7.1] 사원 릴레이션

- 널(Null)값 존재

사원

사원이름	사원번호	주소	전화번호	부서번호1	부서이름1
김창섭	2106	우이동	726-5869	1	영업
김창섭	2106	우이동	726-5869	2	기획
박영권	3426	사당동	842-4538	3	개발
이수민	3011	역삼동	579-4685	2	기획
이수민	3011	역삼동	579-4685	3	개발

[그림 7.2] 사원 릴레이션

- 부서 수 제한이 필요 없다.
- 첫 번째 레코드 삭제 시 1번 영업부의 정보가 삭제 된다.(삭제이상)
- 부서 이름 변경시 일부 사원의 투플만 부서이름을 변경하는 경우 데이터의 불일치가 발생한다.(수정이상)
- 4번 홍보부 추가 시 나머지 애트리뷰트가 전부 NULL을 가지게 된다. 사원번호가 NULL이기에 무결성 제약조건으로 인해 삽입불가(삽입이상)
- 정보의 중복 발생

이러한 문제들을 릴레이션 분해를 통해 해결한다.

관계 데이터베이스 설계의 비공식적인 지침
- 이해하기 쉽고 명확한 스키마 생성
- NULL값 피하기
- 가짜 튜플 생성 방지
- 스키마 정제

함수적 종속성
- 릴레이션의 애트리뷰트들의 의미로부터 결정된다.
- 스키마에 대한 주장(NOT 인스턴스)
- 가능한 모든 인스턴스들이 만족
- 제2 정규형부터 BCNF 적용

결정자
- 어떤 애트리뷰트의 값은 다른 애트리뷰트의 값을 고유하게 결정 가능
- 사원번호는 사원이름을 고유하게 결정
- 주소는 사원이름 결정 X
- A->B , A가 B를 결정한다.

사원

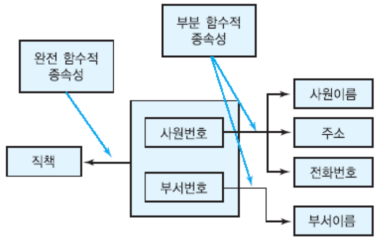
사원번호	사원이름	주소	전화번호	직책	부서번호	부서이름
4257	정미림	홍제동	731-3497	팀장	1	홍보
1324	이범수	양재동	653-7412	프로그래머	2	개발
1324	이범수	양재동	653-7412	웹 디자이너	1	홍보
3609	안명석	양재동	425-8520	팀장	3	홍보

[그림 7.4] 사원 릴레이션

- 사원번호 -> 사원이름
- 사원번호 -> 주소
- 사원번호 -> 전화번호
- 부서번호 -> 부서이름

함수적 종속성
- A->B이면 B가 A에 함수적으로 종속한다.
- 각 A의 값에 대해 반드시 한 개의 B값이 대응된다.

완전 함수 종속성 (FFD)
- 릴레이션 R에서 애트리뷰트 B가 애트리뷰트 A[복합 애트리뷰트]에 함수적으로 종속(A->B)이면서 A의 어떠한 진 부분 집합에도 함수적 종속이 안될 시 A 그대로 함수적 종속.



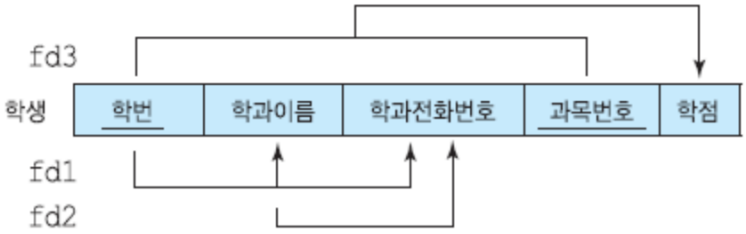
[그림 7.6] 완전 함수적 종속성과 부분 함수적 종속성

- (사원번호, 부서번호) -> 직책, 사원번호 ->직책:불가(완전 함수적 종속성)

부분적 함수 종속성
- (사원번호, 부서번호) -> 부서이름 O
- 부서번호 -> 부서이름 : 부분 함수적 종속성 O
- 사원번호 -> 부서이름 : 부분 함수적 종속성 X

이행적 함수 종속성

A, B, C 애트리뷰트가 있을 때 애트리뷰트 C가 이행적으로 A에 종속한다는 것.



[그림 7.7] 이행적 함수적 종속성

- 학번 -> 학과이름 ∧ 학과이름 -> 전화번호
- 학번 -> 전화번호 : 이행적으로 종속

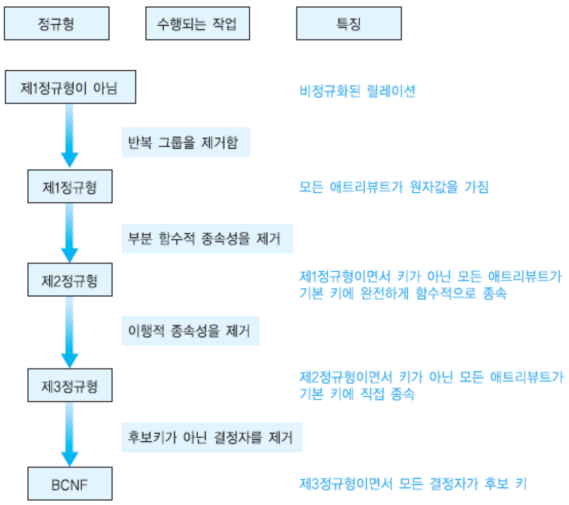
정규화

정규화는 데이터의 일관성, 최소한 데이터 중복, 최대한의 데이터 유연성을 위한 방법이며 데이터를 분해하는 과정이다.

정규화 절차	설명
제1 정규화	- 속성(Attribute)의 원자성을 확보한다. - 기본키(Primary)를 설정한다.
제2 정규화	- 기본키가 2개 이상의 속성으로 이루어진 경우, 부분 함수 종속성을 제거(분해)한다. - A->B, A->C
제3 정규화	- 기본키를 제외한 칼럼 간에 종속성을 제거한다. - 즉, 이행 함수 종속성을 제거한다. A->B->C
BCNF	- 기본키를 제외하고 후보키가 있는 경우, 후보키가 기본키를 종속시키면 분해한다.
제4 정규화	- 여러 칼럼들이 하나의 칼럼을 종속시키는 경우 분해하여 다중값 종속성을 제거한다.
제5 정규화	- 조인에 의해서 종속성이 발생되는 경우 분해한다.

정규화의 장점 : 테이블을 분해해서 데이터의 중복성을 제거하기 때문에 데이터 모델의 유연성을 높인다.

정규화의 단점 : 조회시 조인을 유발하기 때문에 cpu와 메모리를 많이 사용한다.



[그림 7.25] 각 정규형의 특징과 정규화 과정

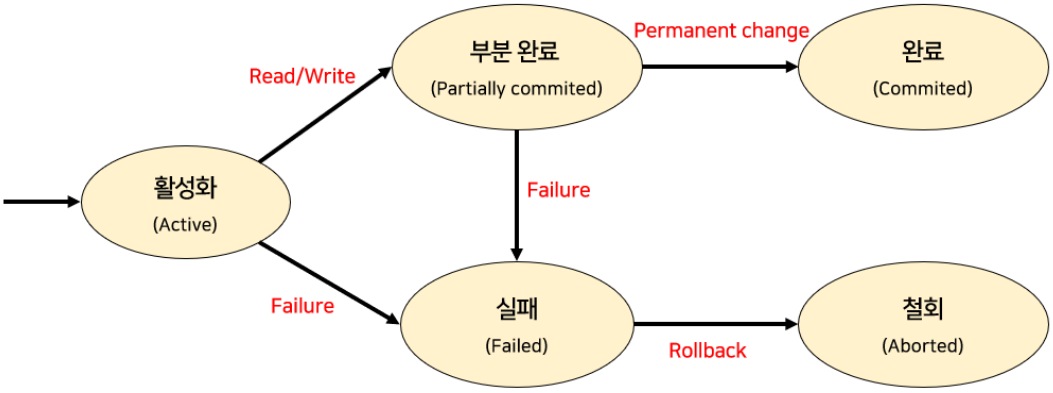
트랜잭션

인가받지 않은 사용자로부터 데이터를 보장하기 위해 DBMS가 가져야하는 특성이자, 데이터베이스 시스템에서 하나의 논리적인 기능을 정상적으로 수행하기 위한 작업의 기본 단위이다.

트랜잭션의 특성(ACID)

특성	설명	주요기법
원자성 (Atomicity)	- 트랜잭션을 구성하는 연산은 반드시 모두 실행이 되거나 혹은 아예 실행되지 않아야 한다.	- Commit/Rollback - 회복성 보장
일관성 (Consistency)	- 트랜잭션이 완료된 상태에서도 트랜잭션 이전의 상황과 동일하게 데이터의 일관성이 있어야 한다.	- 무결성 제약조건 - 동시성 제어
독립성 (Isolation)	- 트랜잭션은 다른 트랜잭션에 간섭을 주거나 받지 않고 독립적으로 실행되어야 한다.	- Read Uncommited - Read Commited - Repeatable Read - Serializable
영속성 (Durability)	- 트랜잭션이 성공적으로 완료되었을 때 그 결과는 영구적으로 반영되어야 한다.	- 회복기법

트랜잭션의 상태 변화



트랜잭션의 상태 설명(활부완신철)

상태	설명
활동 상태(Active)	- 초기 상태, 트랜잭션이 실행 중일 때 가지는 상태
부분 완료 상태 (Partially Committed)	- 마지막 명령문이 실행된 후에 가지는 상태
완료 상태(Committed)	- 트랜잭션이 성공적으로 완료된 후 가지는 상태
실패 상태(Failed)	- 정상적인 실행이 더 이상 진행될 수 없을 때 가지는 상태
철회 상태(Aborted)	- 트랜잭션이 취소되고 데이터베이스가 트랜잭션 시작 전 상태로 환원된 상태

트랜잭션 제어

트랜잭션 제어언어는 TCL(Transaction Control Language)이라고 하며, 트랜잭션의 결과를 허용하거나 취소하는 목적으로 사용되는 언어를 지칭한다.

명령어	핵심	설명
커밋(COMMIT)	트랜잭션 확정	- 트랜잭션을 메모리에 영구적으로 저장하는 명령어
롤백(ROLLBACK)	트랜잭션 취소	- 트랜잭션 내역을 저장 무효화시키는 명령어
체크포인트(CHECKPOINT)	저장 시기 설정	- ROLLBACK을 위한 시점을 지정하는 명령어

병행 제어(일관성 주요 기법) = 동시성 제어

1) 병행 제어(Concurrency Control)

병행 제어는 다수 사용자 환경에서 여러 트랜잭션을 수행할 때, 데이터베이스 일관성 유지를 위해 상호작용을 제어하는 기법이다.

2) 병행 제어의 목적

- 데이터베이스의 공유를 최대화한다.
- 시스템의 활용도를 최대화한다.
- 데이터베이스의 일관성을 유지한다.
- 사용자에 대한 응답시간을 최소화한다.

3) 병행 제어 미보장시 문제점

상태	설명
갱신 손실 (Lost Update)	- 먼저 실행된 트랜잭션의 결과를 나중에 실행된 트랜잭션이 덮어쓸 때 발생하는 오류
현황 파악오류 (Dirty Read)	- 트랜잭션의 중간 수행 결과를 다른 트랜잭션이 참조하여 발생하는 오류
모순성 (Inconsistency)	- 두 트랜잭션이 동시에 실행되어 데이터베이스의 일관성이 결여되는 오류
연쇄복귀 (Cascading Rollback)	- 복수의 트랜잭션이 데이터 공유 시 특정 트랜잭션이 처리를 취소할 경우 트랜잭션이 처리한 곳의 부분을 취소하지 못하는 오류

4) 병행 제어 기법의 종류

상태	설명
로킹 (Locking)	- 같은 자원을 액세스하는 다중 트랜잭션 환경에서 DB의 일관성과 무결성을 유지하기 위해 트랜잭션의 순차적 진행을 보장하는 직렬화 기법 - 로킹의 특징은 다음과 같음 - 데이터베이스, 파일, 레코드 등은 로킹 단위가 될 수 있음. - 로킹 단위가 작아지면 데이터베이스 공유도가 증가 - 로킹 단위가 작아지면 로킹 오버헤드 증가 - 한꺼번에 로킹할 수 있는 객체의 크기를 로킹 단위라고 함

데이터베이스 고립화 수준(격리성 주요 기법)

1) 고립화 수준(Isolation) 개념

고립화 수준은 다른 트랜잭션이 현재의 데이터에 대한무결성을 해치지 않기 위해 잠금을 설정하는 정도이다.

2) 고립화 수준 종류

상태	설명
Read Uncommitted	- 한 트랜잭션에서 연산(갱신) 중인(아직 커밋되지 않은) 데이터를 다른 트랜잭션이 읽는 것을 허용하는 수준 - 연산(갱신) 중인 데이터에 대한 연산은 불허
Read Committed	- 한 트랜잭션에서 연산(갱신)을 수행할 때, 연산이 완료될 때까지 연산 대상 데이터에 대한 읽기를 제한하는 수준 - 연산이 완료되어 커밋된 데이터는 다른 트랜잭션이 읽는 것을 허용.
Repeatable Read	- 선행 트랜잭션이 특정 데이터를 읽을 때, 트랜잭션 종료 시까지 해당 데이터에 대한 갱신/삭제를 제한하는 수준
Serializable Read	- 선행 트랜잭션이 특정 데이터 영역을 순차적으로 읽을 때, 해당 데이터 영역 전체에 대한 접근을 제한하는 수준

회복 기법(영속성 주요 기법)

1) 회복 기법(Recovery) 개념

- 회복 기법은 트랜잭션을 수행하는 도중 장애로 인해 손상된 데이터베이스를 손상되기 이전의 정상적인 상태로 복구시키는 작업이다.

2) 회복 기법 종류

- 회복 기법 종류에는 로그 기반 회복 기법인 지연 갱신 회복 기법, 즉각(즉시) 갱신 회복 기법, 체크 포인트 회복 기법, 그림자 페이징 회복 기법이 있다.

<table>
<tr><th>상태</th><th>설명</th></tr>
<tr><td>로그 기반
회복 기법</td><td>- 지역 갱신 회복 기법과 즉각 갱신 회복 기법이 있음.
<table><tr><td>지연 갱신 회복 기법
(Deferred Update)</td><td>트랜잭션이 완료되기 전까지 데이터베이스에 기록하지 않는 기법</td></tr><tr><td>즉각 갱신 회복 기법
(Immediate Update)</td><td>트랜잭션 수행 중 갱신 결과를 바로 DB에 반영하는 기법</td></tr></table></td></tr>
<tr><td>미디어 회복 기법</td><td>- 디스크에 발생할 수 있는 장애에 대비한 회복 기법
- 덤프(복사본)을 이용하여 전체 데이터베이스의 내용을 일정 주기마다 다른 안전한 저장 장치에 복사</td></tr>
</table>

교착 상태(deadlock)

- 트랜잭션들이 상대가 독점하고 있는 데이터에 unlock 연산이 실행되기를 서로 기다리면서 트랜잭션의 수행을 중단하고 있는 상태
- 처음부터 교착 상태가 발생하지 않도록 예빵하거나, 발생 시 빨리 탐지하여 필요한 조치를 취해야 함.

분산 데이터베이스

- 데이터베이스 시스템 구축 시에 한 대의 물리적 시스템에 데이터베이스 관리 시스템을 설치하고 여러 명의 사용자가 데이터베이스 관리 시스템에 접속하여 데이터베이스를 사용하는 구조를 중앙 집중형 데이터베이스라고 한다.
- 또한 물리적으로 떨어진 데이터베이스에 네트워크로 연결하여 단일 데이터베이스 이미지를 보여 주고 분산된 작업 처리를 수행하는 데이터베이스를 분산 데이터베이스라고 한다.
- 분산 데이터베이스를 사용하는 고객은 시스템이 네트워크로 분산되어 있는지의 여부를 인식 하지 못하면서, 자신만의 데이터베이스를 사용하는 것처럼 사용할 수 있다. 이처럼 데이터베이스는 투명성을 제공해야 한다.
- 투명성은 분산 데이터베이스에서 중요한 요소이며 투명성의 종류에는 분할, 위치, 지역사상, 중복, 장애 및 병행 투명성이 있다.

단편화(fragmentation)

하나의 릴레이션을 더 작은 조각(단편)으로 나누고 각 조각을 별개의 릴레이션으로 처리하는 것

장점

1) 각 조각(단편)이 전체 릴레이션의 일부가 되기 때문에 저장 공간을 적게 사용하고, 관리할 데이터도 줄어듦.
2) 데이터 중복의 장점은 그대로 취하면서 데이터 중복의 단점을 보완할 수 있음.

단편화 종류	설명
수평적 단편화	릴레이션을 수평적으로 단편화하는 것, 릴레이션을 튜플(행) 단위로 나눔
수직적 단편화	릴레이션을 수직적으로 단편화하는 것, 릴레이션을 속성(열) 단위로 나눔
혼합 단편화	수평적 단편화와 수직적 단편화를 모두 사용하여 릴레이션을 나눔

단편화 수행 조건

수행 조건	설명
완전성	전체 릴레이션의 모든 데이터는 어느 한 단편(조각)에는 꼭 속해야 한다.
회복성	단편화된 단편(조각)들로부터 원래의 전체 릴레이션을 회복할 수 있어야 한다.
분리성	전체 릴레이션의 모든 조각을 서로 중복되지 않게 분리해야 한다.

분산 데이터베이스의 투명성 종류

투명성	설명
분할 투명성	- 고객은 하나의 논리적 릴레이션이 여러 단편으로 분할되어 각 단편의 사본이 여러 시스템에 저장되어 있음을 인식할 필요가 없다.
위치 투명성	- 고객이 사용하려는 데이터의 저장 장소를 명시할 필요가 없다. - 고객은 데이터가 어느 위치에 있더라도 동일한 명령을 사용하여 데이터에 접근할 수 있어야 한다.
지역 사상 투명성	- 지역 DBMS와 물적 데이터베이스 사이의 사상이 보장됨에 따라 각 지역 시스템 이름과 무관한 이름이 사용 가능하다.
중복 투명성	- 데이터베이스 객체가 여러 시스템에 중복되어 존재함에도 고객과는 무관하게 데이터의 일관성이 유지된다.
장애 투명성	- 데이터베이스가 분산되어 있는 각 지역의 시스템이나 통신망에 이상이 발생해도, 데이터의 무결성은 보장된다.
병행 투명성	- 여러 고객의 응용 프로그램이 동시에 분산 데이터베이스에 대한 트랜잭션을 수행하는 경우에도 결과에 이상이 없다.

분산 데이터베이스의 장점과 단점

장점	단점
- 데이터베이스의 신뢰성과 가용성이 높다. - 분산 데이터베이스가 병렬 처리를 수행하기 때문에 빠른 응답이 가능하다. - 분산 데이터베이스를 추가하여 시스템 용량 확장이 쉽다.	- 데이터베이스가 여러 네트워크를 통해서 분리되어 있기 때문에 관리와 통제가 쉽다. - 보안관리가 어렵다. - 데이터 무결성 관리가 어렵다. - 데이터베이스 설계가 복잡하다.