데이터 모델과 성능

001. 정규화

1. 정규화(Normalization)

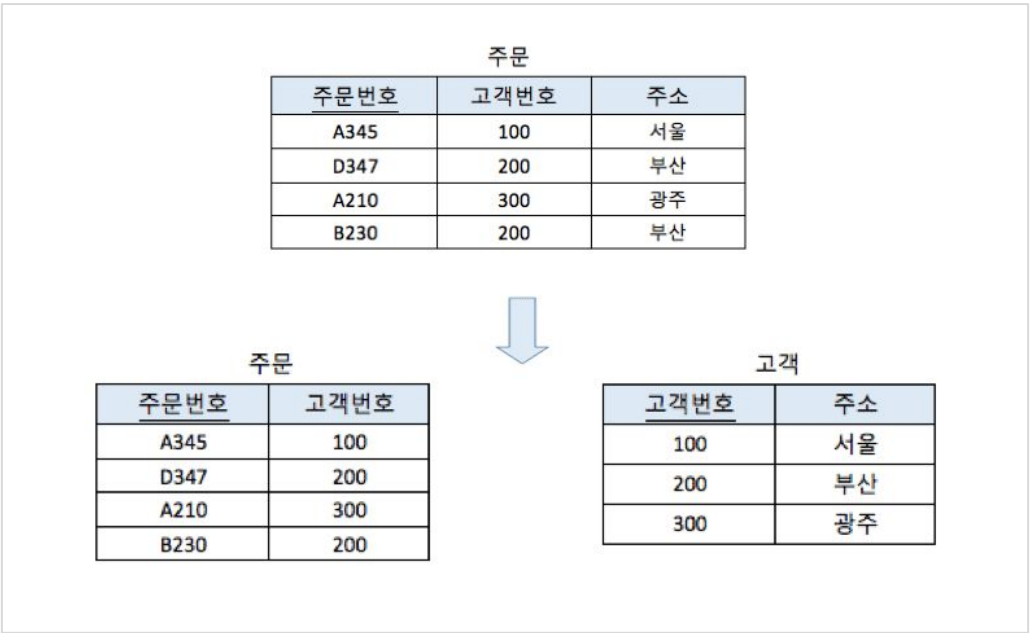
> 정규화는 데이터의 일관성, 최소한의 데이터 중복, 최대한의 데이터 유연성을 위한 방법이며 데이터를 분해하는 과정이다.

> 정규화는 데이터 중복을 제거하고 데이터 모델의 독립성을 확보하기 위한 방법이다.

> 정규화를 수행하면 비즈니스에 변화가 발생하여도 데이터 모델의 변경을 최소화할 수 있다.

> 정규화는 제1정규화부터 제5정규화까지 있지만, 실질적으로는 제3정규화까지만 수행한다.

정규화의 예시


주문
주문번호 고객번호 주소
A345 100 서울
D347 200 부산
A210 300 광주
B230 200 부산
주문
주문번호 고객번호
A345 100
D347 200
A210 300
B230 200
고객
고객번호 주소
100 서울
200 부산
300 광주

정규화 절차

정규화 절차	설명
제1정규화	속성(Attribute)의 원자성을 확보한다. / 기본키(Primary)를 설정한다.
제2정규화	기본키가 2개 이상의 속성으로 이루어진 경우, **부분함수 종속성**을 제거한다.
제3정규화	기본키를 제외한 칼럼 간에 종속성을 제외한다. / 즉, **이행함수 종속성**을 제거한다.
BCNF	기본키를 제외하고 후보키가 있는 경우, 후보키가 기본키를 종속시키면 분해한다.
제4정규화	여러 칼럼들이 하나의 칼럼을 종속시키는 경우 분해하여 다중값 종속성을 제거한다.
제5정규화	조인에 의해서 종속성이 발생되는 경우 분해한다.

2. 함수적 종속성(Functional Dependency)

1) 제1정규화

> 정규화는 함수적 종속성을 근거로 한다. 함수적 종속성이란 X → Y 이면 Y는 X에 함수적으로 종속된다고 말한다.

> 함수적 종속성은 X가 변화하면 Y도 변화하는지 확인한다.

ex) 회원ID가 변화하면 이름도 변경될 것이다. 이런 경우는 회원ID가 기본키가 되고, 회원ID가 이름을 함수적으로 종속한다고 한다.

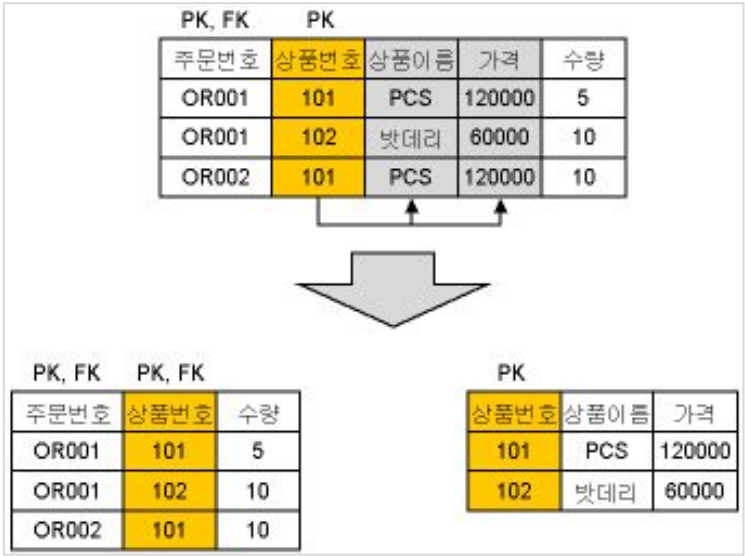
> 기본키를 잡는 것이 제1정규화이다.

2. 함수적 종속성(Functional Dependency)

2) 제2정규화

> 부분 함수 종속성이란, 기본키가 2개 이상의 칼럼으로 이루어진 경우에만 발생한다.

> 기본키가 하나의 칼럼으로 이루어지면 제2정규화는 생략한다.



2. 함수적 종속성(Functional Dependency)

3) 제3정규화

> 제3정규화는 이행 함수 종속성을 제거한다. 이행 함수 종속성이란, 기본키를 제외하고 칼럼간에 종속성이 발생하는 것이다.

> 제3정규화는 제1정규화와 제2정규화를 수행한 다음에 해야 한다.

칼럼 명	주문ID	상품ID	회원ID	회원 명	전화번호	회원등급	수량	단가
키 형태	PK							
데이터	1	A1	Lid	LLL	010-0000-0000	우수	3	3,000
	2	B3	Sid	SSS	010-0000-0001	일반	2	6,000
	3	A2	Jid	JJJ	010-0000-0002	일반	4	8,000
	4	B3	Lid	LLL	010-0000-0000	우수	1	3,000

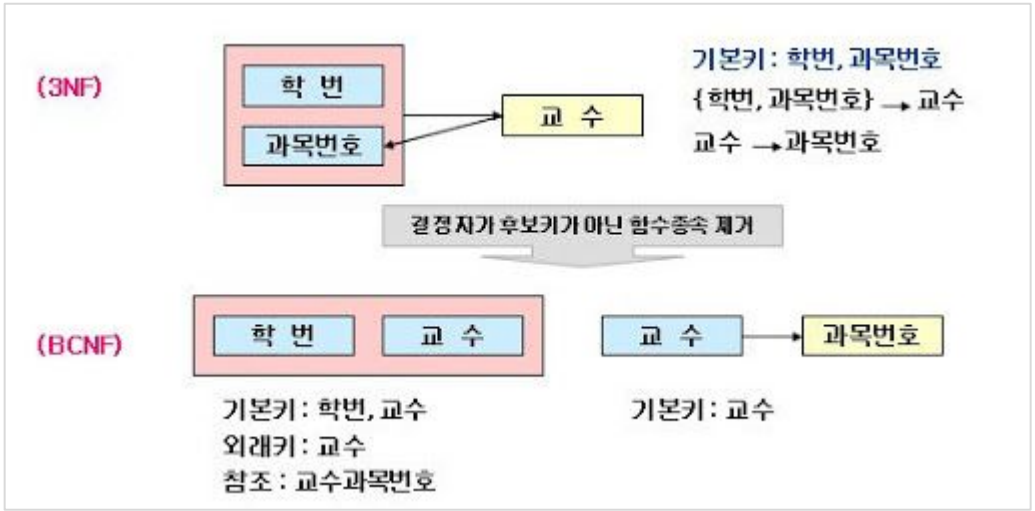
칼럼 명	주문ID	상품ID	회원ID	수량	단가
키 형태	PK		FK		
데이터	1	A1	Lid	3	3,000
	2	B3	Sid	2	6,000
	3	A2	Jid	4	8,000
	4	B3	Lid	1	3,000

칼럼 명	회원ID	회원 명	전화번호	회원등급
키 형태	PK			
데이터	Lid	LLL	010-0000-0000	우수
	Sid	SSS	010-0000-0001	일반
	Jid	JJJ	010-0000-0002	일반

2. 함수적 종속성(Functional Dependency)

4) BCNF(Boyce-Codd Normal Form)

> BCNF는 복수의 후보키가 있고, 후보키들이 복합 속성이어야 하며, 서로 중첩되어야 한다.



Quiz) 다음 중 제3정규화와 관련이 있는 것은?

ㄱ. 속성의 원자성

ㄴ. 부분 함수 종속성

ㄷ. 이행 함수 종속성

ㄹ. 다치 함수 종속성

002. 정규화와 성능

1. 정규화의 문제점

> 정규화는 테이블을 분해해서 데이터 중복을 제거하기 때문에 데이터 모델의 유연성을 높인다.

>정규화는 데이터 조회(SELECT) 시에 조인(Join)을 유발하기 때문에 CPU와 메모리를 많이 사용한다.

> 조인시 중첩된 루프(Nested Loop)를 사용해야 한다. → 마치 for문과 같은 형태가 됨

> 결론적으로 조인이 부하를 유발한다.

> 정규화의 문제점을 해결하기 위해서 반정규화를 하여 하나의 테이블에 저장한다면 조인을 통한 성능 저하는 해결된 것이다. → 반정규화의 필요성

2. 정규화를 사용한 성능 튜닝

> 조인으로 인하여 성능이 저하되는 문제를 반정규화로 해결할 수 있다.

> 반정규화는 데이터를 중복시키기 때문에 또 다른 문제점을 발생시킨다.

ex) 너무 많은 컬럼이 한 테이블에 주가되면 한 개 행의 크기가 데이터베이스 관리 시스템의 입출력 단위인 블록의 크기(Block Size)를 넘어서게 된다. → 이 경우 한 개의 행을 읽기 위해서 여러개의 블록을 읽어야 한다. → 성능저하가 옴

> 예제와 같은 문제가 일어나면 테이블을 분해하는 방법밖에 없고, 따라서 정규화는 입출력 데이터의 양을 줄여서 성능을 향상시킬 수 있는 것이다.

3. 반정규화(De-Normalization)

1) 반정규화

> 데이터베이스의 성능 향상을 위하여, 데이터 중복을 허용하고 조인을 줄이는 데이터베이스 성능 향상 방법이다.

> 반정규화는 조회(SELECT) 속도를 향상하지만, 데이터 모델의 유연성은 낮아진다.

3. 반정규화(De-Normalization)

2) 반정규화를 수행하는 경우

> 정규화에 충실하면 종속성, 활용성은 향상되지만 수행 속도가 느려지는 경우

> 다량의 범위를 자주 처리해야 하는 경우

> 특정 범위에 데이터만 자주 처리하는 경우

>요약/집계 정보가 자주 요구되는 경우

3. 반정규화(De-Normalization)

3) 반정규화 기법

- 계산된 컬럼 추가: 배치 프로그램으로 총 판매액, 평균잔고, 계좌평가 등을 미리 계산하고, 그 결과를 특정 칼럼에 추가한다.
- 테이블 수직분할: 하나의 테이블을 두 개 이상의 테이블로 분할한다. 즉, 칼럼을 분할하여 새로운 테이블을 만드는 것이다.
- 테이블 수평분할: 하나의 테이블에 있는 값을 기준으로 테이블을 분할하는 방법이다.
- 테이블 병합: 1대1 관계, 1대N관계(많은 데이터 중복이 발생) 테이블 병합 혹은 슈퍼타입과 서브 타입 관계가 발생했을 시에 테이블을 통합하여 성능을 향상시킨다.

▲ 슈퍼타입과 서브타입 → 고객 > 개인고객/법인고객 → 고객은 슈퍼타입, 개인/법인고객은 서브타입

슈퍼타입 및 서브 타입 변환 방법

변환 방법	설명
OneToOne Type	- 슈퍼 타입과 서브 타입을 개별 테이블로 도출한다. - 테이블의 수가 많아서 조인이 많이 발생하고 관리가 어렵다.
Plus Type	- 슈퍼타입과 서브 타입 테이블로 도출한다. - 조인이 발생하고 관리가 어렵다.
Single Type	- 슈퍼 타입과 서브 타입을 하나의 테이블로 도출한다. - 조인 성능이 좋고 관리가 편리하지만, 입출력 성능이 나쁘다.

중요) 파티션(Partition 기법)

> 데이터베이스에서 파티션을 사용하여 테이블을 분할할 수 있다. 파티션을 사용하면 논리적으로는 하나의 테이블이지만 여러 개의 데이터 파일에 분산되어서 저장된다.

> **Range Partition:** 데이터 *값의 범위*를 기준으로 파티션을 수행한다.

> **List Partition:** *특정한 값*을 지정하여 파티션을 수행한다.

> **Hash Partition:** *해시 함수*를 적용하여 파티션을 수행한다.

> **Composite Partition:** 범위와 해시를 *복합적*으로 사용하여 파티션을 수행한다.

Quiz) 파티션 기법 중에서 2개 이상의 기법을 사용하는 것은 무엇인가?

ㄱ. Range Partition

ㄴ. List Partition

ㄷ. Hash Partition

ㄹ. Compose Partition

4. 분산 데이터베이스

1) 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.

> 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 인식하지 못하면서, 자신만의 데이터베이스를 사용하는 것처럼 사용할 수 있다. → **투명성**

> 투명성은 분산 데이터베이스에서 중요한 요소이며 투명성의 종류에는 분할, 위치, 지역사상, 중복, 장애 및 병행 투명성이 있다.

4. 분산 데이터베이스

투 명 성	기 능	장 점	단 점
분할 (단편화)	·DB 분할 관리에 무관한 작업환경 제공	·Bottle neck 방지 ·시스템 성능 향상	·충분한 설계기술 필요
위치 투명성	·Data 저장장소에 무관한 접근 제공	·Application 단순화 ·Data의 자유로운 site 왕래	·이중처리로 속도저하 ·저장공간 낭비
지역사상	·각 지역 시스템과 무관한 이름사용	·상향식 점진적 확장 제공	이질형시스템 구현시 복잡
중복 투명성	·논리적Data객체의 site 중복 가능 ·Data일관성 유지에 무관한 사용제공	·질의응답 성능 개선	·갱신전파 overhead 추가 ·기억 공간
장애 투명성	·구성요소 장애에 무관한 Transaction 원자성 유지	·장애처리구현 단순 〈사용자〉	·장애원인 규명 복잡
병행 투명성	·동시수행시각 Transaction 결과 일관성 유지	·자원사용 극대화	·복잡한 locking

장 점	단 점
- 지역 자치성, 점증적 시스템 용량 확장 - 신뢰성과 가용성 - 효용성과 융통성 - 빠른 응답 속도와 통신비용 절감 - 데이터의 가용성과 신뢰성 증가 - 시스템 규모의 적절한 조절 - 각 지역 사용자의 요구 수용 증대	- 소프트웨어 개발 비용 - 오류의 잠재성 증대 - 처리 비용의 증대 - 설계, 관리의 복잡성과 비용 - 불규칙한 응답 속도 - 통제의 어려움 - 데이터 무결성에 대한 위협

4. 분산 데이터베이스

스키마란?

1. 스키마는 데이터베이스의 구조와 제약 조건에 관한 전반적인 명세를 기술한 메타데이터의 집합이다.
2. 스키마는 데이터베이스를 구성하는 데이터 개체(Entity), 속성(Attribute), 관계(Relationship) 및 데이터 조작 시 데이터 값들이 갖는 제약 조건 등에 관해 전반적으로 정의한다.
3. 스키마는 사용자의 관점에 따라 외부 스키마, 개념 스키마, 내부 스키마로 나눠진다.

2) 분산 데이터베이스 설계 방식

> 상향식 설계방식: 지역 스키마 작성 후 향후 전역 스키마를 작성하여 분산 데이터베이스를 구축한다.

> 하향식 설계방식: 전역 스키마 작성 후 해당 지역 사상 스키마를 작성하여 분산 데이터베이스를 구축한다.

> 분산 데이터베이스를 하향식 접근 방식으로 구축한다는 것은 기업 전체의 전사 데이터 모델을 수렴하여 전역 스키마를 생성하고, 그 다음 각 지역별로 지역 스키마를 생성하여 분산 데이터베이스를 구축하는 것이다. 상향식 접근 방식은 지역별로 데이터베이스를 구축한 후에 전역 스키마로 통합하는 것이다.

Quiz) 다음 중 분산 데이터베이스 장점으로 올바르지 않은 것은?

ㄱ. 데이터 처리를 병렬적으로 실행할 수가 있으므로 빠른 응답이 가능한다.

ㄴ. 논리적으로 통합되어 있으므로 데이터 무결성 관리가 쉽다.

ㄷ. 시스템 확장이 편리하다.

ㄹ. 분산 데이터베이스는 시스템에 대한 가용성이 우수하다.

감사합니다.