3주차

상태	시작 전
주제	데이터베이스

3주차(2024.07.05)

- 9 릴레이션 정규화 뷰와 시스템 카탈로그
- 10 뷰와 시스템 카탈로그

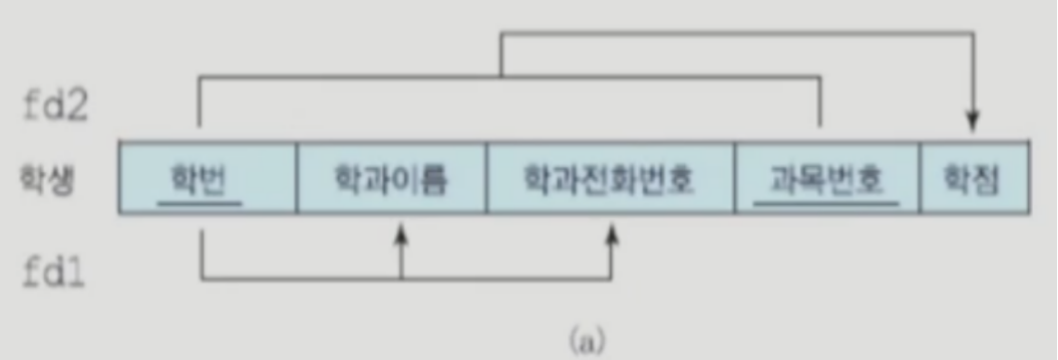
정규화와 갱신이상

> 💡 정규화에서 이상현상
>
> 중복 튜플 중 일부만 변경하여 데이터가 불일치하게 되는 문제

정규화는 이상 현상을 제거하기 위해서 릴레이션을 의미 있는 속성들로만 구성하기 위해 릴레이션을 분해하는 과정이다. 함수적 종속성을 판단하여 정규화를 진행한다.

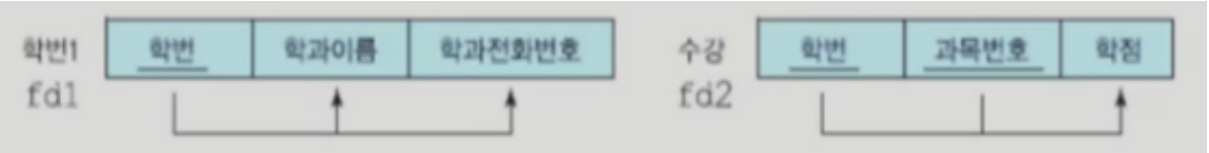
제 1정규형의 갱신이상

> 💡 제 1정규형 : 릴레이션에 속한 모든 속성의 도메인이 원자 값으로만 구성되어 있는 것



부분 함수적 종속성 존재

학번	지도교수	학과	과목번호	성적
100	P1	컴퓨터	C123	A
100	P1	컴퓨터	C234	B
200	P2	컴퓨터	C123	B
300	P3	전자	C400	A
400	P4	수학	C500	C



제 2정규형으로 분리

제 2정규형 갱신이상

💡 제 2정규형 : 제1정규형에 속하면서, 기본키가 아닌 모든 속성이 기본키에 완전 함수 종속되면 제2정규형

기본키가 복합키 일때 이것을 분리하는 과정

학생1

학번	학과이름	학과전화번호
11002	컴퓨터과학	210-2261
24036	정보통신	210-2585
11048	컴퓨터과학	210-2261

발생하는 갱신이상

- 삽입이상 : 학과를 새로 추가하려면 소속 학생이 반드시 있어야함 (학번 널 값 불가)
- 수정이상 : 학과의 전화번호가 변경되면 해당 학과에 속한 학생들의 전화번호를 같이 수정해야함
- 삭제이상 : 학과에서 마지막 학생이 삭제되면 학과의 정보도 삭제됨

갱신이상 이유? 이행적 함수 종속성 발생

💡 이행적 함수 종속성

특정 속성(컬럼) A의 값이 바뀌면 다른 속성(컬럼) B의 값도 바뀌거나

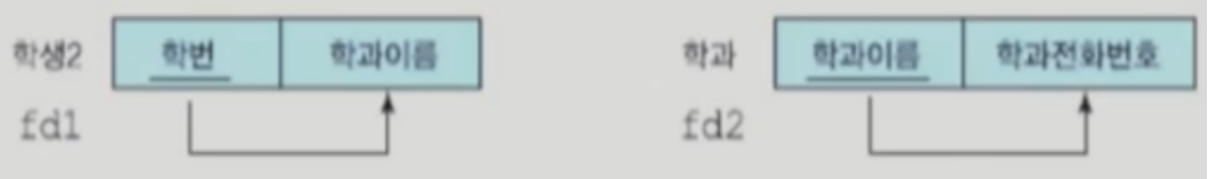
특정 속성(컬럼) A의 값을 알면 다른 속성(컬럼) B의 값도 유일하게 정해지는 종속 관계

위의 예시로 보면
학번 → 학과 이름
학과 이름 → 학과 전화번호

학번 → 학과 전화번호

근데 학번에 따라 학과 전화번호가 결정되는게 아니고 학과 이름으로 결정되는 것이니 이행적 함수 종속이 있다고 볼 수 있음

이행적 함수 종속성을 제거해주기 위해 제 3정규형으로 분리해줌



제 3정규형으로 분리

제 3정규형 갱신이상

💡 제 3정규형 : 제 2 정규형에 속하면서, 기본키가 아닌 모든 속성이 기본키에 이행적 함수 종속이 되지 않으면

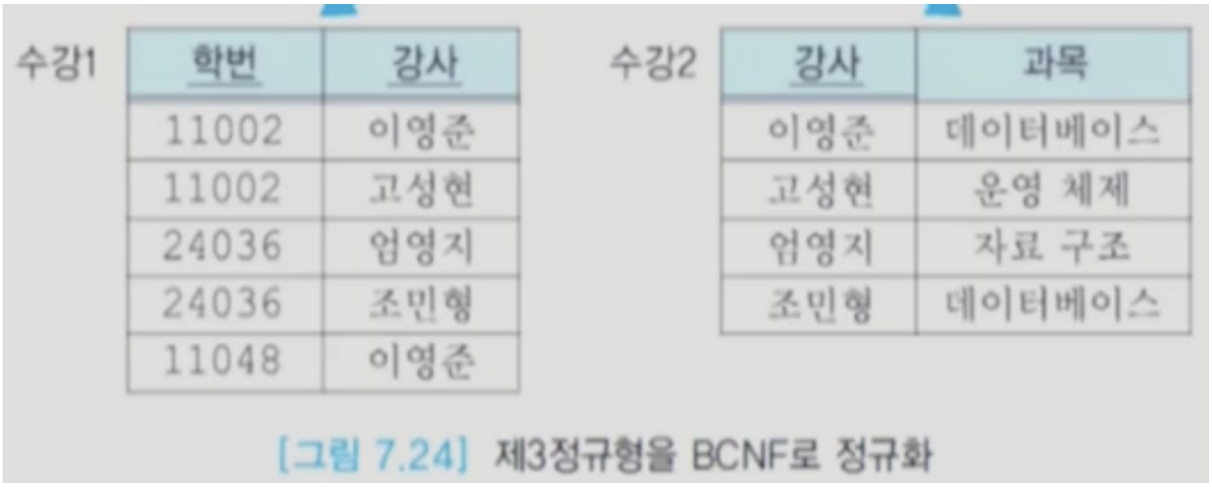
제 3 정규형

수강

학번	과목	강사
11002	데이터베이스	이영준
11002	운영 체제	고성현
24036	자료 구조	엄영지
24036	데이터베이스	조민형
11048	데이터베이스	이영준

발생하는 갱신이상

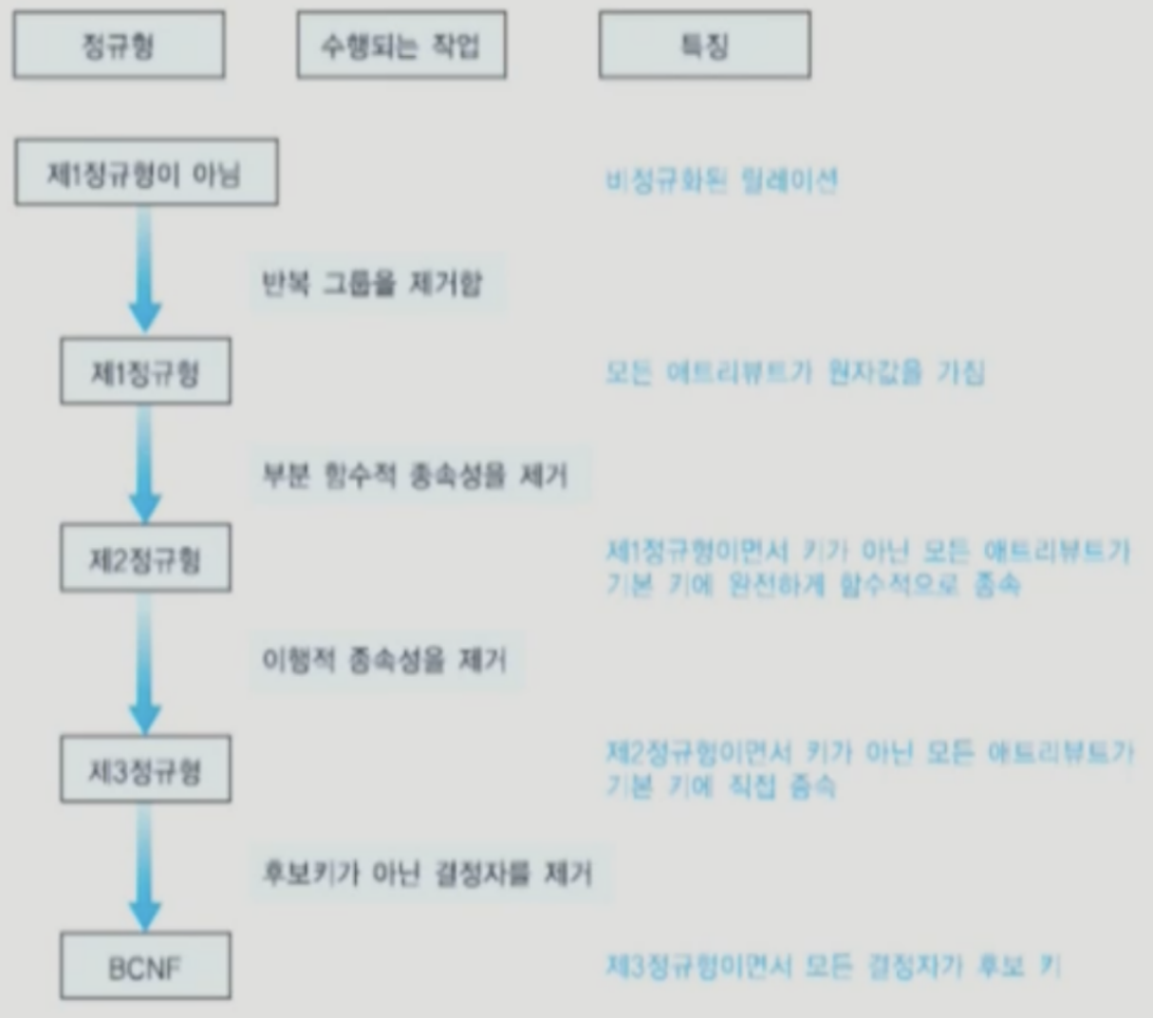
- 삽입이상 : 강사가 과목을 추가하려면 학생의 정보가 반드시 필요 (널 값 불가)
- 삭제 이상 : 학생의 정보를 삭제하면 강사도 같이 삭제됨
- 수정 이상 : 강사의 과목이 변경되는 경우 모두 찾아서 변경시켜주어야 한다.



BCNF로 분리

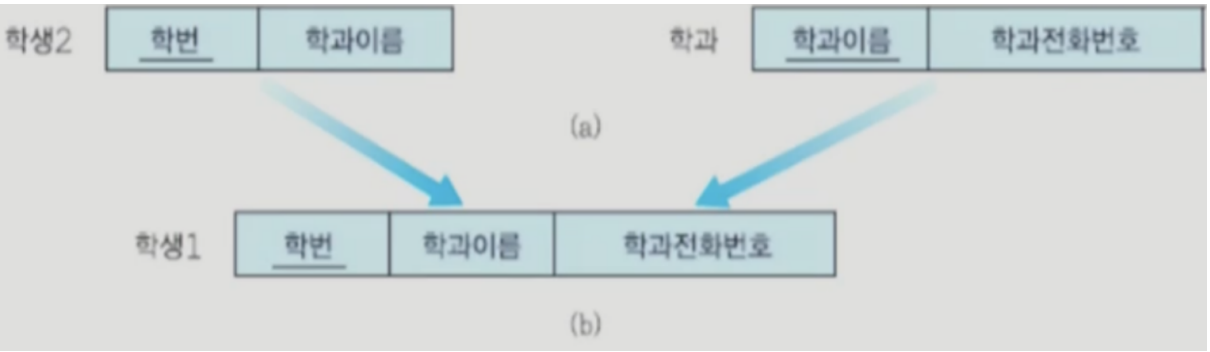
보이스 코드 정규화

💡 BCNF 정규화 : 제 3정규형을 만족해야 하고, 모든 결정자가 후보키 집합에 속해야 한다.



반 정규화

- 분해된 릴레이션을 합치는 것
- 정규화 단계가 감소함
- 오히려 정규화전의 과정의 쿼리 처리가 효율이 좋다면 역정규화를 하면됨
- 중복과 갱신 이상의 단점 보다 효율적의 장점이 더 클 때를 뜻함
- 역정규화의 종류로 JOIN줄이기, 컬럼 추가로 그룹화 줄이기



뷰와 시스템 카탈로그

💡 뷰

- 원본 릴레이션에서 유도된 릴레이션으로써 외부 뷰와 다름
- 보안, 복잡한 쿼리를 단순화, 데이터 독립성을 위해 사용

시스템 카탈로그

- DB의 여러가지 다양한 객체의 정보를 가짐
- 시스템 카탈로그를 통해 원하는 릴레이션을 DB에서 찾고, 그 릴레이션에 어떤 애트리뷰트들이 있는지, 데이터 타입은 무엇인지 쉽게 파악 가능

뷰

- 외부 뷰: ANSI/SPARC 3단계 아키텍처에서 사용자가 보는 DB의 구조
- 외부 뷰와는 다르게 여기서 뷰는 관계 DB에서 한 사용자의 외부 뷰가 아닌, 기존의 원본 릴레이션에 대한 SELECT문으로 선택한 유도된 릴레이션을 의미
- 즉, 뷰는 조회만 가능한 외부 뷰 + 데이터 검색, 갱신까지 가능

뷰의 정의

CREATE VIEW 뷰이름 [(속성)]

AS

SELECT문[WITH CHECK OPTION];

```
CREATE VIEW   EMP_DNO3 (ENO, ENAME, TITLE)
AS SELECT     EMPNO, EMPNAME, TITLE
   FROM       EMPLOYEE
   WHERE      DNO=3;
```

부서번호가 3인 사원의 (사원번호, 사원이름, 직급)을 나타내는 뷰를 정의

EMPLOYEE

EMPNO	EMPNAME	TITLE	MANAGER	SALARY	DNO
2106	김창섭	대리	1003	2500000	2
3426	박영권	과장	4377	3000000	1
3011	이수민	부장	4377	4000000	3
1003	조민희	과장	4377	3000000	2
3427	최종철	사원	3011	1500000	3
1365	김상원	사원	3426	1500000	1
4377	이성래	사장	∧	5000000	2

위의 릴레이션 상황에서 EMP_DNO3의 뷰는 파란색 부분을 뜻함

스냅샷

- 어느 시점에 SELECT문의 결과를 기본 릴레이션 형태로 저장해 놓은 것
- 스냅샷은 사진을 찍은 것과 같아서 스냅샷을 정의하는 시점의 기본 릴레이션의 내용이 스냅샷에 반영된다.
- 어떤 시점의 조직체의 현황, 예를 들어 몇년 몇월 시점에 근무하던 사원들의 정보, 재고 정보 등이 스냅샷으로 정의될 수 있다.

뷰의 쿼리연산

- 기존의 ANSI/SPARC 3단계에 해당하는 외부 뷰는 조회 기능만 제공
- 그러나, 뷰 또한 유도된 릴레이션이므로 쿼리처리가 가능

뷰의 장점

- 데이터 무결성 보장뷰를 통해 투플 추가 및 수정 시 뷰의 SELECT문의 조건과 맞지 않으면 거절대신 WITH CHECK OPTION이 있어야함
- SELECT문에서 WHERE DNO = 3과 맞지 않으므로 거절

```
CREATE VIEW  EMP_DNO3 (ENO, ENAME, TITLE)
AS  SELECT   EMPNO, EMPNAME, TITLE
    FROM     EMPLOYEE
    WHERE    DNO = 3
    WITH CHECK OPTION;
```

```
UPDATE   EMP_DNO3
SET      DNO = 2
WHERE    ENO = 3427;
```

- 복잡한 쿼리를 간단하게 표현할 수 있다.
- 데이터 무결성을 보장하는데 활용할 수 있다.
- 데이터 독립성을 제공한다.
  - 속성들이 바뀌어도 기존 질의를 바꾸는 번거로움을 줄임
- 데이터 보안 기능을 제공한다.
  - 원본 릴레이션에 접근하지 않고 뷰의 **유도된 릴레이션**으로 접근시킴으로써 보안 메커니즘으로 작동
- 동일한 데이터에 대한 여러 가지 뷰를 제공한다.
  - 원본 릴레이션은 모든 것을 다 보여주지만, 뷰를 통해 다르게 보여줄 수 있음

뷰의 갱신

뷰에 대한 갱신도 원본 릴레이션에 영향을 미침

```
INSERT INTO EMP_DNO3
VALUES (4293, '김정수', '사원');
```

```
INSERT INTO EMPLOYEE
VALUES(4293, '김정수', '사원', , , );
```

```
INSERT INTO EMP_PLANNING
VALUES ('박지선', '대리', 2500000);
```

```
INSERT INTO EMPLOYEE
VALUES( , '박지선', '대리', , 2500000, );
```

갱신이 불가능한 뷰

- 원본 릴레이션의 기본키가 포함되지 않은 뷰
- 뷰가 가지지 않은 애트리뷰트가 원본 릴레이션에선 NOT NULL일 때
- 집단 함수가 포함된 뷰
- 조인으로 정의된 뷰

시스템 카탈로그

- 시스템내의 객체(기본 릴레이션, 뷰, 인덱스, 사용자, 접근 권한 등)의 모든 테이터를 포함
- 이런 데이터를 **메타 데이터**라고 부름
- 표준화되어있지 않아서, DBMS마다 서로 다른 시스템 카탈로그로 제공
- **데이터 사전**또는 **시스템 테이블**이라고 부름

쿼리 처리 활용과 쿼리 최적화

SELECT EMPNAME, SALARY, SALARY * 1.1

FROM EMPLOYEE

WHERE TITLE = '과장' **AND** DNO = 2;

1. SELECT구문 전체가 문법적으로 올바른지 확인
2. EMPLOYEE 릴레이션이 DB에 존재하는지 확인
3. SELECT절과 WHERE절의 애트리뷰트가 릴레이션에 존재하는지 확인
4. SALARY와 DNO가 숫자연산에 붙었으므로 정수형이고, TITLE이 문자열과 비교되므로 CHAR형인지 확인
5. 사용자가 EMPNAME, SALARY의 검색 권한이 있는지 확인
6. WHERE절의 애트리뷰트인 TITLE, DNO가 인덱스로 정의되어 있는지 확인
7. 인덱스가 존재한다면, 어느 인덱스로 조회하는 것이 빠른지 선택(최적화)

> 시스템 카탈로그의 예
>
> - SYS_RELATION: DB안에서 릴레이션들 목록과 그 릴레이션이 몇개의 튜플, 애트리뷰트를 가졌는지, 한 튜플의 크기는 얼만지 등을 저장
> - SYS_ATTIBUTE: 애트리뷰트들의 목록과 그 애트리뷰트가 어느 릴레이션에 속했는지, 타입은 뭔지 등을 저장

SYS_RELATION

RelId	RelOwner	RelTups	RelAtts	RelWidth
EMPLOYEE	KIM	7	6	36
DEPARTMENT	KIM	4	3	18
...	...	...	...	...

SYS_ATTRIBUTE

AttrelId	AttID	AttName	AttOff	AttType	AttLen	PkorFk
EMPLOYEE	1	EMPNO	0	int	4	Pk
EMPLOYEE	2	EMPNAME	4	char	10	
EMPLOYEE	3	TITLE	14	char	10	
EMPLOYEE	4	MANAGER	24	int	4	Fk
EMPLOYEE	5	SALARY	28	int	4	
EMPLOYEE	6	DNO	32	int	4	Fk
DEPARTMENT	1	DEPTNO	0	int	4	Pk
DEPARTMENT	2	DEPTNAME	4	char	10	
DEPARTMENT	3	FLOOR	14	int	4	
...	...	...	...	...	...	

시스템 카탈로그의 갱신

- 시스템 카탈로그는 갱신이 불가능
- DELETE, UPDATE 또는 INSERT 자체가 불가능

> KIM이 EMPLOYEE의 MANAGER 애트리뷰트를 지우기 위해
>
> **ALTER TABLE** EMPLOYEE **DROP COLUMN** MANAGER; > 가능
>
> **DELETE FROM** SYS_ATTRIBUTE
>
> **WHERE** AttRelId= 'EMPLOYEE' **AND** AttName= 'MANAGER'; > 거절

MS SQL Server의 시스템 카탈로그

- 통계 정보는 UPDATE STATISTICS문을 사용하여 수동으로 갱신할 수 있음
- SQL Server는 릴레이션의 데이터가 변경될 때 주기적으로 통계 정보를 자동으로 갱신함
- 통계 정보를 갱신할 필요성이 있는가를 확인하기 위해서 샘플링 방법을 사용함
- 통계 정보가 갱신되는 빈도는 애트리뷰트 또는 인덱스의 크기와 변경되는 데이터의 양에 따라 결정됨
- 모든 릴레이션들에 대한 구성을 정의하는 데이터를 시스템 테이블이라고 부르는 특수한 테이블 집합에 저장함
- 사용자나 응용 프로그램이 정보 스키마 뷰, Transact-SQL문 및 함수, 시스템 저장 프로시저 등을 사용하여 시스템 테이블에 저장된 정보를 검색함