

제2장



관계 데이터 모델과 제약조건

- 2.1 관계 데이터 모델의 개념
- 2.2 릴레이션의 특성
- 2.3 릴레이션의 키
- 2.4 무결성 제약조건
 - 연습 문제

2장. 관계 데이터 모델과 제약조건

↙ 이것 사용하는 DBMS를 '관계 DBMS'라 부른다.

- 관계 데이터 모델은 지금까지 제안된 데이터 모델들 중에서 가장 개념이 단순한 구현 (표현) 데이터 모델의 하나
- IBM 연구소의 E.F. Codd가 1970년에 관계 데이터 모델을 제안함
- 관계 데이터 모델을 최초로 구현한 가장 중요한 관계 DBMS 시제품은 1970년 대에 IBM 연구소에서 개발된 System R
- 1980년대 후반부터 여러 가지 데이터 모델들이 새로 등장했지만 관계 DBMS는 여전히 가장 널리 사용되는 DBMS

2장. 관계 데이터 모델과 제약조건(계속)

관계 데이터 모델이 큰 성공을 거둔 요인

- ✓ 바탕이 되는 데이터 구조로서 간단한 테이블(릴레이션)을 사용
- ✓ 중첩된 복잡한 구조가 없음
- ✓ 집합 위주로 데이터를 처리
- ✓ 다른 데이터 모델에 비해 이론이 잘 정립되었음
- ✓ 숙련되지 않은 사용자도 쉽게 이해할 수 있음
- ✓ 표준 데이터베이스 응용에 대해 좋은 성능을 보임
- ✓ 관계 데이터베이스 설계와 효율적인 질의 처리 면에서 뛰어난 장점을 가짐

↑
'테이블 형식'의
설계

← '행·열' 구분

← 효율적인 질의 처리

2.1 관계 데이터 모델의 개념

□ 관계 데이터 모델

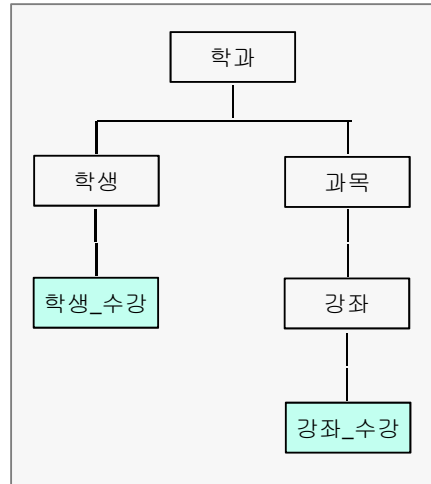
- 모든 데이터를 논리적으로 동일한 구조(릴레이션)의 관점에서 구성 ↖ 핵심!!!
- ✓ 논리적으로 연관된 데이터를 연결하기 위해서 링크나 포인터를 사용하지 않음 ⇒ '키'를 사용하여 데이터 항목들의 관계를 나타냄.
- ✓ 응용 프로그램들은 데이터베이스 내의 레코드들의 어떠한 순서와도 무관하게 작성됨 ↖ '키' ↗ ↗ 데이터들의 순서 상관X
- ✓ 선언적인 질의어를 통한 데이터 접근을 제공
- 사용자는 원하는 데이터(what)만 명시하고, 어떻게 이 데이터를 찾을 것인가(how)는 명시할 필요가 없음 ↖ '데이터 추상화'로 연결됨.

↖ 핵심!!!

참고: 다른 데이터 모델과 비교

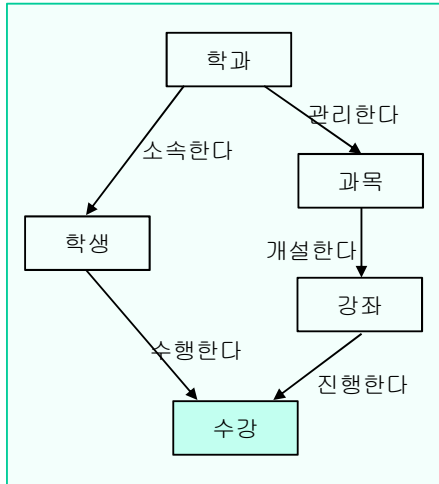
계층 데이터 모델

데이터베이스의 논리적 구조를 트리(tree) 형태로 표현



네트워크 데이터 모델

데이터베이스의 논리적 구조를 그래프(graph) 또는 네트워크(network) 형태로 표현



관계 데이터 모델

데이터베이스의 논리적 구조를 표(table) 형태로 표현



2.1 관계 데이터 모델의 개념(계속)

□ 기본적인 용어

- ✓ 릴레이션(relation): 2차원의 테이블(스프레드 시트와 유사)
- ✓ 레코드(record): 릴레이션의 각 행 (=row)
- ✓ 튜플(tuple): 레코드를 좀더 공식적으로 부르는 용어
- ✓ 애트리뷰트(attribute): 릴레이션에서 이름을 가진 하나의 열

2.1 관계 데이터 모델의 개념(계속)



[그림 2.1] 릴레이션의 예

ER 모델의 '차수'와는
다른 것이다.

2.1 관계 데이터 모델의 개념(계속)

□ 도메인(domain)

① 한 애트리뷰트에 나타날 수 있는 유효 값들의 집합

② 한 애트리뷰트의 도메인의 값들은 원자값

✓ 프로그래밍 언어의 데이터 타입과 유사함

✓ 동일한 도메인이 여러 애트리뷰트에서 사용될 수 있음

✓ 복합 애트리뷰트나 다치 애트리뷰트는 허용되지 않음

✓ 도메인 정의

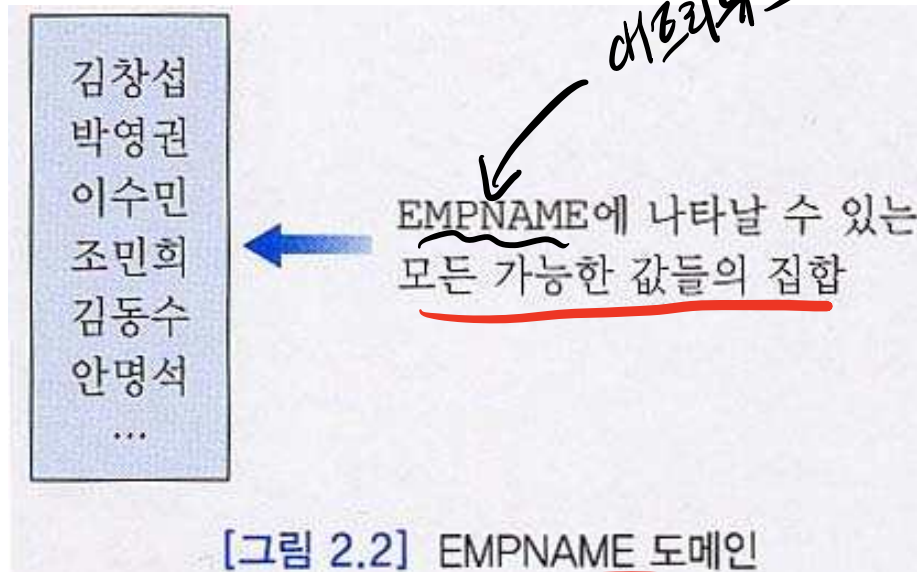
```
CREATE DOMAIN EMPNAME CHAR(10)
CREATE DOMAIN EMPNO INTEGER
CREATE DOMAIN DNO INTEGER
```

하위 애트리뷰트들을
값만 있는
애트리뷰트.

↳ Mult-Value-Attribute.

여를 표시하기 위해서, 'MVA' 테이블을
구분 마다.

2.1 관계 데이터 모델의 개념(계속)



2.1 관계 데이터 모델의 개념(계속)

□ 차수(degree)와 카디날리티(cardinality)

✓ 차수: 한 릴레이션에 들어 있는 애트리뷰트들의 수

- 유효한 릴레이션의 최소 차수는 1

- 릴레이션의 차수는 자주 바뀌지 않음

✓ 카디날리티: 릴레이션의 튜플 수

- 유효한 릴레이션은 카디날리티 0을 가질 수 있음

- 릴레이션의 카디날리티는 시간이 지남에 따라 계속해서 변함

cause, 테이블 스키마는
'내포'이기 때문에

↑ cause, 테이블 인스턴스는
'외연'이기 때문에

주키이는 'null' 값이 존재할 수 없음!!!

2.1 관계 데이터 모델의 개념(계속)

□ 널값(null value)

← '알려지지 않음' 또는 '적용할 수 없음'을 나타내기 위해 널값을 사용 >

해상!!

✓ 예: 사원 릴레이션에 새로운 사원에 관한 튜플을 입력하는데, 신입 사원의 DNO(부서번호)가 결정되지 않았을 수 있음

널값은 숫자 도메인의 0이나 문자열 도메인의 공백 문자 또는 공백 문자열과 다름

✓ DBMS들마다 널값을 나타내기 위해 서로 다른 기호를 사용함

주키이는 'null' 값이 있을 수 없음!!!

2.1 관계 데이터 모델의 개념(계속)

□ 릴레이션 스키마(relation schema)

✓ 릴레이션을 구성하는 틀(template)로서 릴레이션의 이름과

② 릴레이션의 애트리뷰트들의 집합으로 표현

✓ 표기법

릴레이션이름(애트리뷰트1, 애트리뷰트2, ... 애트리뷰트N)

✓ 기본 키 애트리뷰트에는 밑줄 표시

✓ 내포(intension)라고 함

· 잘 변하지 않는다.

2.1 관계 데이터 모델의 개념(계속)

□ 릴레이션 인스턴스(relation instance)

- ✓ 어느 시점에 릴레이션에 포함되어 있는 튜플들의 집합
- ✓ 시간의 흐름에 따라 계속 변함
- ✓ 일반적으로 릴레이션에는 현재의 인스턴스만 저장됨
- ✓ 외연(extension)이라고 함

특정 시점의
데이터베이스
내용!!

2.1 관계 데이터 모델의 개념(계속)

EMPLOYEE	EMPNO	EMPNAME	TITLE	DNO	SALARY	내포
	2106	김창섭	대리	2	2000000	외연
	3426	박영권	과장	3	2500000	
	3011	이수민	부장	1	3000000	
	1003	조민희	대리	1	2000000	

[그림 2.3] 릴레이션의 내포와 외연

2.1 관계 데이터 모델의 개념(계속)

- 관계 데이터베이스(relational database) 스키마 ← DDL'로 작성
 - ✓ 하나 이상의 릴레이션 스키마들로 이루어짐
- 관계 데이터베이스 인스턴스 ← DML'로 작성.
 - ✓ 릴레이션 인스턴스들의 모임으로 구성됨

2.1 관계 데이터 모델의 개념(계속)

DEPARTMENT (DEPTNO, DEPTNAME, FLOOR)
EMPLOYEE (EMPNO, EMPNAME, TITLE, DNO, SALARY)

[그림 2.4] 관계 데이터베이스 스키마 (nhl)

DEPARTMENT	DEPTNO	DEPTNAME	FLOOR
	1	영업	8
	2	기획	10
	3	개발	9

EMPLOYEE	EMPNO	EMPNAME	TITLE	DNO	SALARY
	2106	김창섭	대리	2	2000000
	3426	박영권	과장	3	2500000
	3011	이수민	부장	1	3000000
	1003	조민희	대리	1	2000000
	3427	최종철	사원	3	1500000

[그림 2.5] 관계 데이터베이스 인스턴스

2.2 릴레이션의 특성

□ 릴레이션

- ✓ 튜플들의 집합

□ 릴레이션의 특성

- ✓ 각 릴레이션은 오직 하나의 레코드 타입만 포함

- ✓ 한 애트리뷰트 내의 값들은

모두 같은 도메인의 값

- ✓ 애트리뷰트들의 순서는

중요하지 않음

튜플 순서도 중요하지 않음!

<세 개의 어트리뷰트를 가진 테이블에
→ 네 개의 어트리뷰트를 가진 튜플이
들어갈 수 없다.>

DEPARTMENT

DEPTNO	DEPTNAME	FLOOR
1	영업	8
2	기획	10
3	개발	9

==

DEPARTMENT

FLOOR	DEPTNO	DEPTNAME
8	1	영업
10	2	기획
9	3	개발

[그림 2.6] 애트리뷰트들의 순서가 달라도 동일한 릴레이션

2.2 릴레이션의 특성(계속)

□ 릴레이션의 특성(계속)

✓ 동일한 튜플이 두 개 이상 존재하지 않음

⇒ 키가 존재함

한 튜플의 각 애트리뷰트는 원자값을 가짐

→ '중복 최소화'

⇒ 키 어트리뷰트가 존재함

즉, '기본 자료형'만 가능함.

↑ 더 이상 쪼갤 수 없는 값

DEPARTMENT	DEPTNO	DEPTNAME	FLOOR
	1	영업	{8, 9}
	2	기획	10
	3	개발	{7, 9}

[그림 2.7] 튜플의 각 애트리뷰트는 원자값만 가져야 함

← 잘못된 릴레이션!
⇒ 원자적인 값이 아니기 때문!

집합 ⇒ 원자값이

아닐!

☒ 221

- (= 피아블)

- 세리뷰로 순서로 상관 없음!

[그림 2.8] 튜플들의 순서가 달라도 동일한 릴레이션

ex) 학생 | — 학번, 학과, 이름, 주번, 주2, 현-본 번호, 출석, ...

2.3 릴레이션의 키

□ 릴레이션의 키

✓ 각 튜플을 고유하게 식별할 수 있는 하나 이상의 애트리뷰트들의 모임

✓ **수퍼 키**(superkey), **후보 키**(candidate key), **기본 키**(primary key), **대체 키**(alternate key), **외래 키**(foreign key)

□ 수퍼 키

✓ 한 릴레이션 내의 특정 튜플을 고유하게 식별하는 하나의 애트리뷰트 또는 애트리뷰트들의 집합

✓ 예: 신용카드 회사의 고객 릴레이션에서 (신용카드번호, 주소) 또는 (주민등록번호, 이름) 또는 (주민등록번호)

✓ 튜플들을 고유하게 식별하는데 꼭 필요하지 않은 애트리뷰트들을 포함할 수 있음

- ① 유일성
- ② 최소성
- ③ 불변성: 시간이 변화하여도, 키 값이 변하면 안된다.

① 유일성
을 만족하는 키

② 최소성은 기본 키와 같다!
① 유일성
② 최소성

DB 설계자가 후보키 중 선택한 키!

즉, 튜플들을 고유하게 식별하지 못하는 애트리뷰트도 포함될 수도 있다.

해상!!!

2.3 릴레이션의 키(계속)

□ 후보 키

- ✓ 각 튜플을 고유하게 식별하는 최소한의 애트리뷰트들의 모임

예: (신용카드번호, 주소)는 신용카드 회사의 고객 릴레이션의

후보 키가 아니지만 (신용카드번호)는 후보 키

- ✓ 모든 릴레이션에는 최소한 한 개 이상의 후보 키가 있음
- ✓ 후보 키도 두 개 이상의 애트리뷰트로 이루어질 수 있으며 이런 경우에

복합 키 (composite key)라고 부름

예: (학번, 과목번호)가 후보 키

수강

학번	과목번호	학점
11002	CS310	A0
11002	CS313	B+
24036	CS345	B0
24036	CS310	A+

[그림 2.9] 수강 릴레이션

2.3 릴레이션의 키(계속)

그림 2.10의 학생 릴레이션에서 이름이 후보 키가 될 수 있는가?

그림 2.10의 학생 릴레이션에서 이메일이 후보 키가 될 수 있는가?

학생	학번	이름	이메일
	11002	이홍근	sea@hanmail.net
	24036	김순미	smkim@venus.uos.ac.kr
	13427	박상웅	blue@hanmir.com

[그림 2.10] 학생 릴레이션

2.3 릴레이션의 키(계속)

□ 기본 키 ← 후보키 중 하나를 '기본키'로 지정함.

- ✓ 한 릴레이션에 후보 키가 두 개 이상 있으면, 설계자 또는 데이터베이스 관리자가 이들 중에서 하나를 기본 키로 선정함

예: 신용카드 회사의 고객 릴레이션에서 신용카드번호와 주민등록번호가 후보 키가 될 수 있음. 이 중에서 신용카드 번호를 기본 키로 선정

- ✓ 자연스러운 기본 키를 찾을 수 없는 경우에는 레코드 번호와 같이 종종 인위적인 키 애트리뷰트를 릴레이션에 추가할 수 있음

기본키에
대한 기준:
채택하기 쉬운
변수형을 원함!!
↑ 핵심!!!

ex) 문자열 보란 필수.

2.3 릴레이션의 키(계속)

□ 대체 키

- ✓ 기본 키가 아닌 후보 키

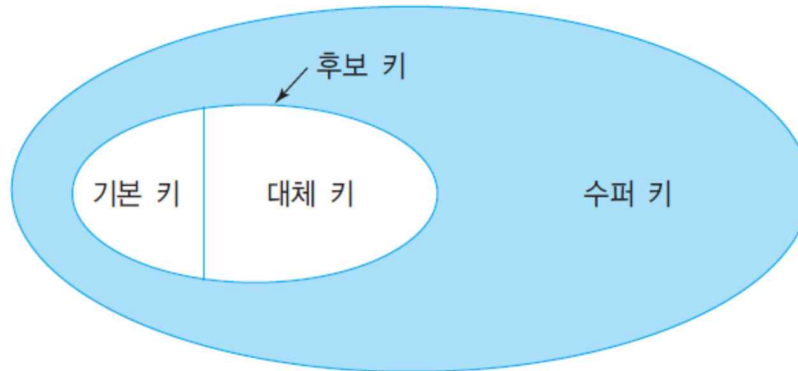
예: 신용카드 회사의 고객 릴레이션에서 신용카드번호를 기본 키로
선정하면 주민등록번호는 대체 키

□ 외래 키

- < 어떤 릴레이션의 기본 키를 참조하는 애트리뷰트 >
- ✓ 관계 데이터베이스에서 릴레이션들 간의 관계를 나타내기 위해서 사용됨
- ✓ 외래 키 애트리뷰트는 참조되는 릴레이션의 기본 키와 동일한 도메인을 가져야 함
- ✓ 자신이 속한 릴레이션의 기본 키의 구성요소가 되거나 되지 않을 수 있음

다른 릴레이션의
기본키를
수입해서
쓰는 키!

2.3 릴레이션의 키(계속)

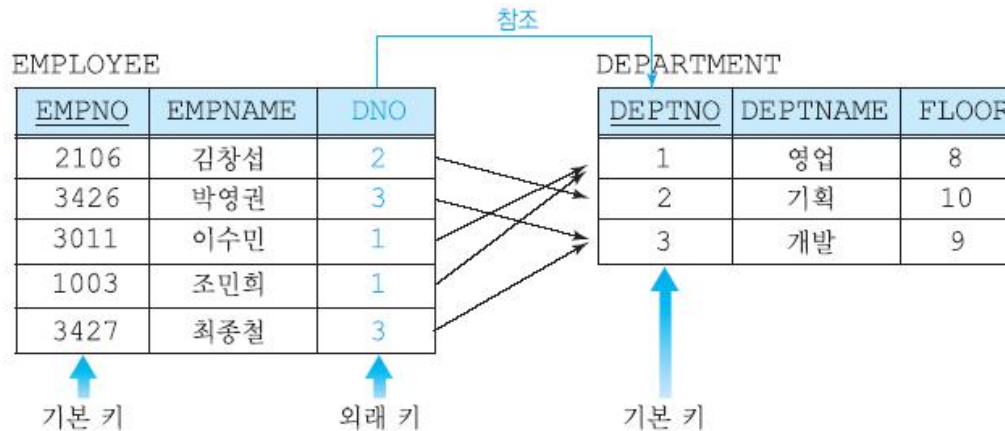


[그림 2.11] 키들의 포함 관계

2.3 릴레이션의 키(계속)

□ 외래 키의 유형 **← 가장 흔한 유형!!**

✓ 다른 릴레이션의 기본 키를 참조하는 외래 키



[그림 2.12] 다른 릴레이션을 참조하는 외래 키

2.3 릴레이션의 키(계속)

□ 외래 키의 유형(계속)

- ✓ 자체 릴레이션의 기본 키를 참조하는 외래 키

참조

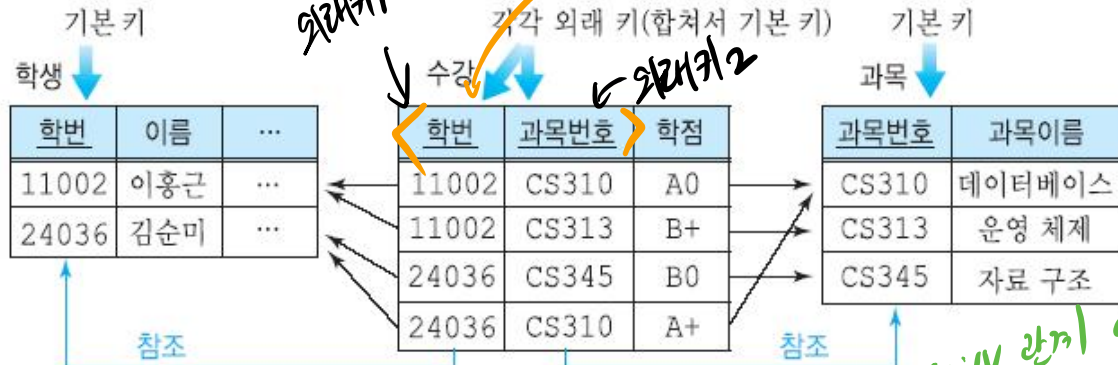
<u>EMPNO</u>	EMPNAME	MANAGER	DNO
2106	김창섭	3426	2
3426	박영권	3011	3
3011	이수민	^	1
1003	조민희	3011	1
3427	최종철	2106	3

[그림 2.13] 자체 릴레이션을 참조하는 외래 키

2.3 릴레이션의 키(계속)

□ 외래 키의 유형(계속)

- ✓ 기본 키의 구성요소가 되는 외래 키



[그림 2.14] 기본 키의 구성요소가 되는 외래 키

m:N 관계 테이블 만들기
 'm:N'의 관계 테이블을 테이블로 나타낸

2.4 무결성 제약조건

❑ 데이터 무결성 제약조건(data integrity constraint)

- ✓ 데이터의 정확성 또는 유효성을 의미
- ✓ 일관된 데이터베이스 상태를 정의하는 규칙들을 묵시적으로 또는 명시적으로 정의함
- ✓ 데이터베이스가 갱신될 때 DBMS가 자동적으로 일관성 조건을 검사하므로 응용 프로그램들은 일관성 조건을 검사할 필요가 없음

2.4 무결성 제약조건(계속)

❑ 도메인 제약조건(domain constraint)

- ✓ 각 애트리뷰트 값이 반드시 원자값이어야 함
- ✓ 애트리뷰트 값의 디폴트 값, 가능한 값들의 범위 등을 지정할 수 있음
- ✓ 데이터 형식을 통해 값들의 유형을 제한하고, CHECK 제약 조건을 통해 값들의 범위를 제한할 수 있음
- ✓ SQL2는 도메인을 명시적으로 정의하는 것을 허용하지만, 오라클은 지원하지 않음

집합 자료형이 될 수 없다.

❑ 키 제약조건(key constraint)

- ✓ 키 애트리뷰트에 중복된 값이 존재해서는 안됨

2.4 무결성 제약조건(계속)

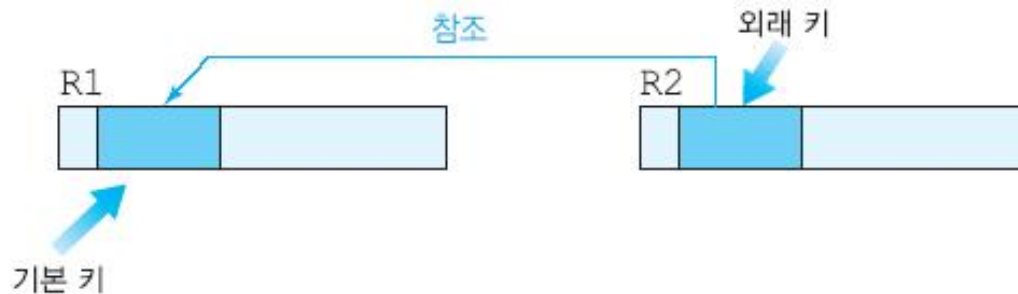
- ❑ 기본 키와 엔티티 무결성 제약조건(entity integrity constraint)
 - ✓ 릴레이션의 기본 키를 구성하는 어떤 애트리뷰트도 널값을 가질 수 없음
 - ✓ 대체 키에는 적용되지 않음
 - ✓ 사용자는 릴레이션을 생성하는 데이터 정의문에서 어떤 애트리뷰트가 릴레이션의 기본 키의 구성요소인가를 DBMS에게 알려줌

2.4 무결성 제약조건(계속)

□ 외래 키와 참조 무결성 제약조건(referential integrity constraint)

- ✓ 참조 무결성 제약조건은 두 릴레이션의 연관된 튜플들 사이의 일관성을 유지하는데 사용됨
- ✓ 관계 데이터베이스가 릴레이션들로만 이루어지고, 릴레이션 사이의 관계들이 다른 릴레이션의 기본 키를 참조하는 것을 기반으로 하여 묵시적으로 표현되기 때문에 외래 키의 개념이 중요
- ✓ 릴레이션 R2의 외래 키가 릴레이션 R1의 기본 키를 참조할 때 참조 무결성 제약조건은 아래의 두 조건 중 하나가 성립되면 만족됨
 - 외래 키의 값은 R1의 어떤 튜플의 기본 키 값과 같다
 - 외래 키가 자신을 포함하고 있는 릴레이션의 기본 키를 구성하고 있지 않으면 널값을 가진다

2.4 무결성 제약조건(계속)



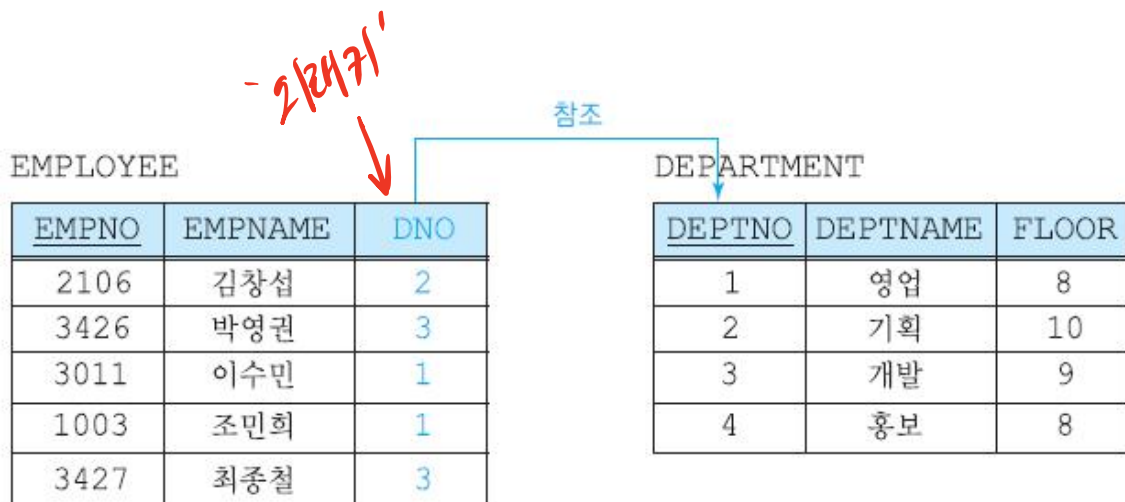
[그림 2.16] 참조 무결성 제약조건

2.4 무결성 제약조건(계속)

□ 무결성 제약조건의 유지

- ✓ 데이터베이스에 대한 갱신 연산은 삽입 연산, 삭제 연산, 수정 연산으로 구분함
- ✓ DBMS는 각각의 갱신 연산에 대하여 데이터베이스가 무결성 제약조건들을 만족하도록 필요한 조치를 취함
- ✓ DBMS는 외래 키가 갱신되거나, 참조된 기본 키가 갱신되었을 때 참조 무결성 제약조건이 위배되지 않도록 해야 함
- ✓ EMPLOYEE 릴레이션의 DNO 애트리뷰트가 DEPARTMENT 릴레이션의 기본 키인 DEPTNO를 참조하는 외래 키이므로, DEPARTMENT를 참조된 릴레이션, EMPLOYEE를 참조하는 릴레이션으로 부르기로 함

2.4 무결성 제약조건(계속)



[그림 2.17] 관계 데이터베이스 인스턴스

2.4 무결성 제약조건(계속)

□ 삽입

- ✓ 참조되는 릴레이션에 새로운 튜플이 삽입되면 참조 무결성 제약조건은 위배되지 않음
- ✓ DEPARTMENT에 새로 삽입되는 튜플의 기본 키 애트리뷰트의 값에 따라서는 도메인 제약조건, 키 제약조건, 엔티티 무결성 제약조건 등을 위배할 수 있음
- ✓ 참조하는 릴레이션에 새로운 튜플을 삽입할 때는 도메인 제약조건, 키 제약조건, 엔티티 무결성 제약조건 외에 참조 무결성 제약조건도 위배할 수 있음
 - 예: EMPLOYEE 릴레이션에 (4325, 오혜원, 6)이라는 튜플을 삽입하면 참조 무결성 제약조건을 위배하게 됨

2.4 무결성 제약조건(계속)

❑ 삭제

✓ **참조하는** 릴레이션에서 튜플이 삭제되면 도메인 제약조건, 키 제약조건, 엔티티 무결성 제약조건, 참조 무결성 제약조건 등 모든 제약조건을 위배하지 않음

✓ **참조되는** 릴레이션에서 튜플이 삭제되면 참조 무결성 제약조건을 위배하는 경우가 생기거나 생기지 않을 수 있음 *← 핵심!!*

- 예1: DEPARTMENT 릴레이션에서 네 번째 튜플인 (4, 홍보, 8)을 삭제하더라도 참조 무결성 제약조건을 위배하지 않음
- 예2: DEPARTMENT 릴레이션에서 세 번째 튜플인 (3, 개발, 9)를 삭제하면 참조 무결성 제약조건을 위배하게 됨

2.4 무결성 제약조건(계속)

□ 참조 무결성 제약조건을 만족시키기 위해서 DBMS가 제공하는 옵션

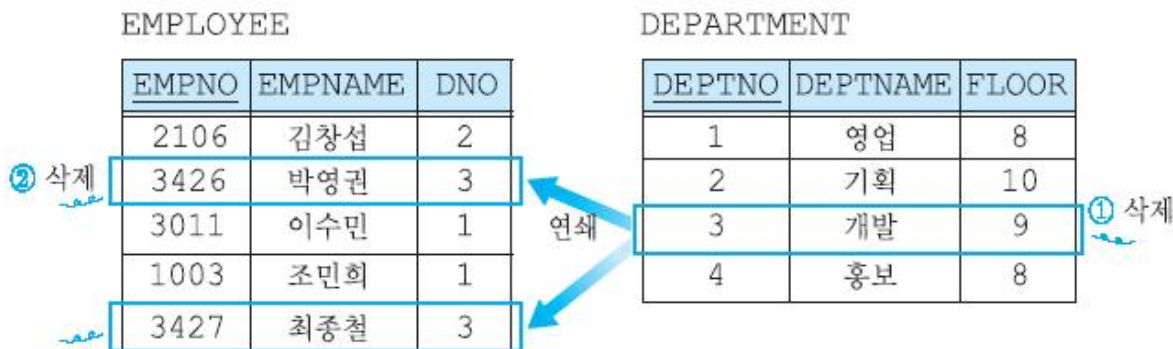
✓ 제한(restricted)

- 위배를 야기한 연산을 단순히 거절
- 예: DEPARTMENT 릴레이션에서 (3, 개발, 9)를 삭제하면 참조 무결성 제약조건을 위배하게 되므로 삭제 연산을 거절

✓ 연쇄(cascade)

- 참조되는 릴레이션에서 튜플을 삭제하고, 참조하는 릴레이션에서 이 튜플을 참조하는 튜플들도 함께 삭제
- 예: DEPARTMENT 릴레이션에서 (3, 개발, 9)를 삭제하면 EMPLOYEE 릴레이션에서 부서번호 3을 참조하는 두 번째 튜플과 다섯 번째 튜플도 함께 삭제

2.4 무결성 제약조건(계속)



[그림 2.18] 연쇄 삭제

2.4 무결성 제약조건(계속)

□ 참조 무결성 제약조건을 만족시키기 위해서 DBMS가 제공하는 옵션(계속)

✓ 널값(nullify)

- 참조되는 릴레이션에서 튜플을 삭제하고, 참조하는 릴레이션에서 이 튜플을 참조하는 튜플들의 외래 키에 널값을 삽입
- 예: DEPARTMENT 릴레이션에서 (3, 개발, 9)를 삭제하면 EMPLOYEE 릴레이션에서 부서번호 3을 참조하는 두 번째 튜플과 다섯 번째 튜플의 부서번호에 널값을 삽입

✓ 디폴트값

- 널값을 넣는 대신에 디폴트값을 넣는다는 것을 제외하고는 바로 위의 옵션과 비슷함

2.4 무결성 제약조건(계속)

□ 수정

- ✓ DBMS는 수정하는 애트리뷰트가 기본 키인지 외래 키인지 검사함
- ✓ 수정하려는 애트리뷰트가 기본 키도 아니고 외래 키도 아니면 수정 연산이 참조 무결성 제약조건을 위배하지 않음
- ✓ 기본 키나 외래 키를 수정하는 것은 하나의 튜플을 삭제하고 새로운 튜플을 그 자리에 삽입하는 것과 유사하므로, 삽입 및 삭제에서 설명한 제한, 연쇄, 널값, 디폴트값 규칙이 수정 연산에도 적용됨
- ✓ 오라클에서는 수정 연산에 대해 제한적으로 참조 무결성 제약조건을 유지