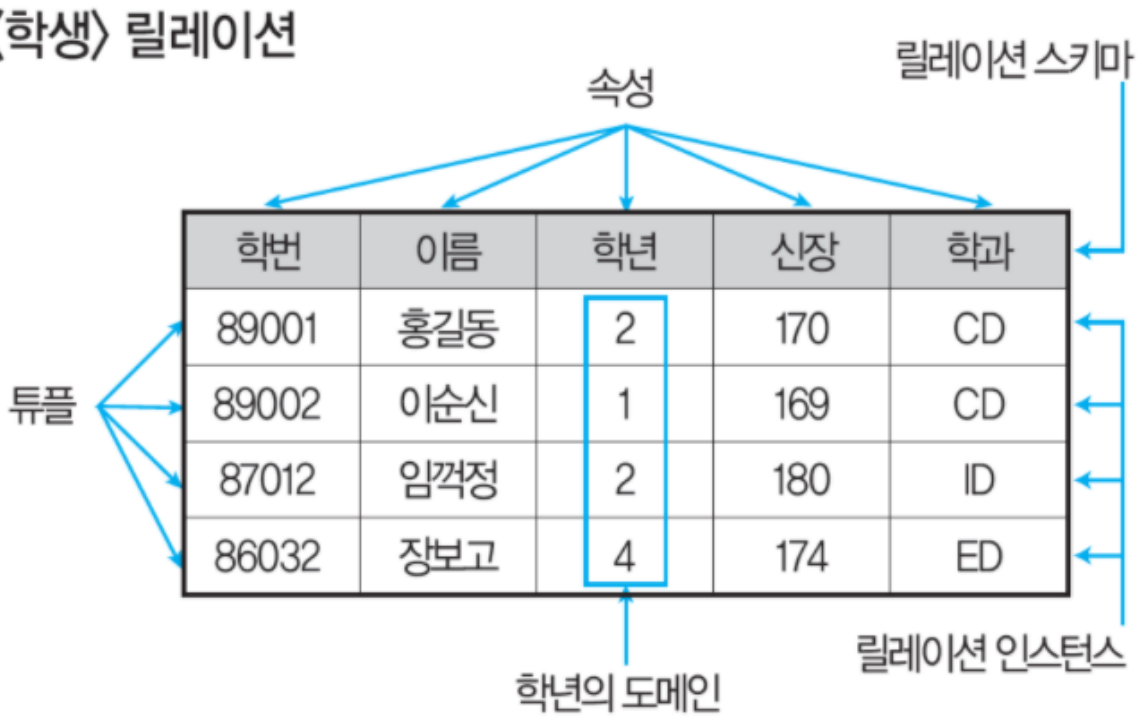
02-01 [DB]

관계형 데이터베이스 용어

아래 그림 및 용어는 E.F.Codd가 정의한 것으로 오늘 날 일반적으로 사용하는 용어와는 차이가 있음



1. 릴레이션(relation)

   우리가 테이블이라고 지칭했던 것.
   관계형 데이터베이스에서 정보를 구분하여 저장하는 기본 단위, 릴레이션들은 서로를 구분할 수 있는 이름을 가지며 동일한 데이터베이스 내에 같은 이름을 가진 릴레이션이 존재할 수 없다.

2. 속성(attribute)

   하나의 릴레이션은 현실세계의 어떤 개체(entity)를 표현하고 저장하는데 사용 되는데, 표현할 개체의 구체적인 정보 항목에 해당하는 것이 속성.
   현실세계의 개체들은 많은 속성을 가지는데 그 중에서 관리해야 할 필요가 있는 속성들만을 선택하여 릴레이션에 포함.
   속성역시 고유한 이름을 가지며 동일 릴레이션 내에 하나만 존재.

3. 튜플(tuple)

릴레이션이 현실세계의 어떤 개체를 표현한다면, 튜플은 그 개체에 속한 구성원들 개개의 정보를 표현.
Ex. 학생 : 개체 / 김철수 : '학생' 개체의 구성원

개체의 각 구성원에 대해 관리해야 할 정보의 항목은 한 릴레이션 내에서 동일하나, 그 내용은 서로 다르므로 튜플이라는 형태로 릴레이션 안에 표현하는 것.

릴레이션내에 하나의 튜플과 동일한(중복된) 튜플은 존재하면 안된다 (PK와 관련.)

4. 도메인(domain)

   릴레이션에 포함된 각각의 속성들이 가질 수 있는 값들의 집합
   도메인의 개념이 필요한 이유 ⇒ 릴레이션에 저장되는 데이터 값들이 본래 의도했던 값들만 저장되고 관리되도록 하는데 있음.
   Ex.) 성별 : 남 / 여만 들어갈 수 있음.
   ⇒ 이처럼 도메인에 포함해야 할 값을 미리 알 수 있고 선언해놓을 수 있는 속성도 존재하지만,
   Ex.) 이름 : 김종수 / 산다라박 / 마이클조던
   ⇒ 이처럼 도메인에 포함될 값의 경우의 수가 너무 많아서 미리 선언할 수 없는 속성도 존재
   때문에, 도메인만으로 특정 속성에 대해 그 속성과 관련 없는 값이 들어오는 것을 완벽하게 막을 수 있는 방법은 없음.
   DBMS에서는 각 속성에대해 데이터 타입과 길이를 미리 지정하여 그에 맞는 값들만 들어오도록 하는 방법을 사용.

E.F. Codd의 용어	File 시스템의 용어	자주 사용되는 언어
릴레이션(relation)	파일(file)	테이블(table)
속성(attribute)	필드(field)	열(column), 컬럼
튜플(tuple)	레코드(record)	행(row)

기본키와 외래키

데이터베이스 설계를 위해 반드시 이해해야 할 개념 중의 하나(기본키, 외래키)

현재 피겨 교본의 도서번호, 도서이름, 출판사, 가격은 완전히 동일한데 이는 중복된 두 개의 튜플.
이 때 두 개의 중복된 튜플은 DB상에서 문제가 될 수 있음(카운트, 식별의 문제)
이를 위해,
중복된 튜플의 삽입을 막아야 할 필요가 있음.

- 중복 여부를 효과적으로 확인하는 방법
  ⇒ 후보키(candidate key) : 테이블에서 각 튜플을 구별하는 데 기준이 되는 하나 혹은 그 이상의 컬럼들의 집합.
  후보키는 테이블에 있는 각 튜플을 고유하게 식별할 수 있어야한다.
- 위 예시에서 도서를 구분할 수 있는 방법은 도서이름, 출판사 책을 표현하는 어떤 속성으로도 불가능.
  때문에, 도서를 구분하기 위해 '도서번호'로 구분하며 이를 릴레이션(테이블)에 대한 '후보키'라고 부른다.

DBMS는 새로운 튜플이 릴레이션에 삽입될 때 새로운 튜플의 후보키 값이 기존의 튜플들의 후보키 값과 동일한지 여부를 비교하여 중복여부를 확인

* 후보키(candidate key)는 기본키(primary key), 대체키(alternate key)로 구분된다.

학생 테이블이 존재할 때, '학생번호'는 각 튜플을 구분하는 기준으로 사용될 수 있지만,
마찬가지로 '주민번호' 역시 개인별로 유일하므로 튜플을 구분하는 기준이 될 수 있음.
이 때
한 릴레이션 내에 학생번호, 주민번호 둘 다 존재한다면 한 릴레이션(테이블)은 한 개 이상의 후보키를 가질 수 있다.

이 때 설계자는 어떤 값을 튜플을 구분하는 값으로 할 지 선택해야하며 이 중 선택된 것이 '기본 키', 선택되지 않은 것이 '대체 키' 라고 한다.
Ex.)위의 경우 주민번호보다 짧고, 릴레이션이 나타내는 공간에서 더 주로 사용되는 '학생번

호’가 기본키에 더 적합.
* 만약 한 릴레이션 내에 후보키가 한개라면 자연스럽게 후보 키 = 기본 키
** ‘복합 키’ : 한 릴레이션 내에서 두 개 이상의 컬럼이 합쳐져야만 후보키 역할을 할 수 있으면 두 개 이상의 컬럼을 묶어 기본 키로 지정가능. 이런 경우를 ‘복합 키’ 라고 명시
***
기본 키에 대해서는 인덱스를 생성하여 중복성 여부를 빠르게 검사하나, 대체 키로 지정된 컬럼에 대해서는 중복성 여부는 체크하되, 자동으로 인덱스가 만들어지지는 않음

**** 후보 키는(기본 키가 될 수 있는) 최소한의 컬럼 혹은 컬럼들의 집합으로 후보키를 구성해야 한다는 것.
그렇지 않다면 무결성이 깨질 수 있음. (학번이 동일하고 이름만 다른 튜플이 중복, but 현실 세계에서 학번이 동일한 것은 불가능)

외래키 : 테이블 간의 데이터 일치 및 무결성 보증 수단

만약 사원 정보 테이블에서 사원이 속한 부서번호에 해당하는 부서명을 찾고 싶다면, 부서 테이블을 조회해야 하는데, 이 때 부서 테이블을 조회할 수 있는 부서번호가 사원 테이블에 있다면.
’사원 테이블이 부서 테이블을 참조한다.’
⇒ 테이블 A가 테이블 B의 기본키에 해당하는 컬럼을 가지고 있을 때, 테이블 A가 테이블 B를 참조한다고 말하며,
A 테이블에 있는 B 테이블의 기본 키를 ‘외래 키(foreign key)’ 라고 한다.

- 외래 키가 무결성을 지킬 수 있는 이유

  1. Ex.) 만약 부서 테이블에 존재하지 않는 부서 번호를 가진 튜플이 삽입된다면?
     ⇒ 튜플의 삽입을 거절. (부서 테이블에 해당 값이 존재하지 않으므로.)
  2. Ex.) 만약 부서 테이블에 특정 부서가 사라진다면 ?
     ⇒ 사원 테이블의 일부 튜플들이 부서번호를 잃은 채 존재.

  - 이를 위해 DBMS는 foreign key가 삭제 될 때 몇 가지 조치가 가능.
    1. 제한(restricted)
    삭제하려는 튜플의 부서 번호 값을 사원 테이블에서 가지고 있는 튜플이 있다면 삭제 연산을 거절.
    2. 연쇄(cascade)
    삭제 연산을 수행한 뒤 삭제된 부서번호 값을 갖는 사원 테이블의 튜플도 함께 삭제.

    참조 관계를 따라가며 연쇄적으로 관련된 튜플들을 삭제.

    3. 널값으로 대체(nullify)
    삭제 연산을 수행한 뒤 삭제된 부서번호 값을 갖는 사원 테이블의 튜플에서 부서번

호를 null값으로 변경
* 기본은 1(제한)이며, 현업에서는 제한, 연쇄 등의 이유로 foreign key지정을 잘 하지 않음.

- 외래 키를 통해 두 테이블 간의 데이터 무결성을 유지하는 것을 ‘참조 무결성’ 이라고 한다.

뷰(View)

하나의 테이블, 혹은 여러 테이블에 대하여 특정 사용자나 조직의 관점에서 데이터를 바라볼 수 있도록 해주는 수단으로, ‘가상 테이블’이라고도 부른다.

Ex.) 사원 테이블이 존재할 때 사원 정보를 필요로 하는 부서가 많으며, 이들 모두 사원 테이블의 필요한 속성이 각각 다름.
1. 인사팀 : 급여 지급을 위한 {사번, 이름, 입사일자, 급여액}
2. 기획실 : 인력 배치 {사번, 이름, 근무부서, 담당 업무}
3. 사내복지 : 생일 선물 {사번, 이름, 생년월일, 주소}

같은 사원에 대한 정보를 부서별로 팔요로 하는 세부항목이 다를 수 있음.

이 때 사원 테이블에 있는 모든 사원 정보를 보여주게 되면, 기획실에 보여서는 안되는 ‘급여액’같은 항목이 같이 노출되게 됨.

뷰는 이런 경우 각 부서의 필요애 맞는 데이터를 제공 가능.

사용자 관점에서 뷰는 일반 테이블과 거의 구분되지 않음.
⇒ 질의가 가능하기 때문에,
일반 테이블과 뷰의 가장 큰 차이점은 실제로 물리적인 데이터를 가지고 있느냐(일반테이블) / 없느냐(뷰)
⇒
뷰는 일반 테이블로부터 데이터를 가져다 보여주는 것 이므로, 일반 테이블이 있어야 정의 가능.

▼ 예시

```
CREATE VIEW view_emp1
AS SELECT empid, ename, hired_date, salary
FROM emp;

CREATE VIEW high_salary
AS SELECT empid, ename, dept, salary
```

```
FROM emp
WHERE salary >= 350
```

> 뷰의 사용목족
> 1. 하나의 테이블에 대하여 여러 부서에서 서로 다른 관점으로 보길 원할 때.
> 2. 테이블에 급여와 같이 일반 사용자에게는 감추어야 할 컬럼이 있을 때, 그것을 제외하고 뷰를 만들어 제공함으로써 보안을 유지
> 3. 자주 사용하는 복잡한 질의문을 미리 뷰로 정의하여 두고 간편하게 사용하고자 할 때.

- 뷰에 대해서 튜플의 삽입 삭제는 가능한 경우가 있고, 불가능한 경우가 있다.

SQL 언어

관계형 DB는 SQL이라는 표준을 지키며 명령문을 제공.
SQL은 비절차적 언어(non-procedural language)이며, 사용자는 자신이 원하는 것만을 명시하며, 원하는 것을 DBMS가 어떻게 처리할지는 명시할 필요가 없음.
* DBMS는 실행 시 대소문자를 구분하지 않음.(테이블 이름, 컬럼의 이름 등)
편의상 DBMS의 예약어는 '대문자' 그 외(테이블 이름, 컬럼 이름)등은 '소문자'로 명시

- DML

1. SELECT
   테이블에 저장된 정보를 조회하는 데 사용
   DB의 목적이 정보를 한곳에 모아 관리하면서 다수가 공유하는 것임을 생각할 때 SQL에서 가장 빈번히 사용되는 명령어.
2. INSERT

   릴레이션(테이블)에 튜플(행)을 삽입할 때 사용.
3. UPDATE
   릴레이션(테이블)에 저장되어 있는 튜플(행)의 값을 변경할 때 사용
4. DELETE
   릴레이션(테이블)에 저장되어 있는 튜플(행)의 값을 삭제할 때 사용

- DDL

5. CREATE
   릴레이션(테이블), 뷰, 사용자 등 데이터베이스 내의 객체들을 생성하는 데 사용.

DML은 일반 DB사용자 들이 사용하나, DDL은 데이터베이스 구조와 관련 있으므로 DBA나 설계자들이 주로 사용.

6. ALTER
   CREATE로 생성한 데이터베이스 내의 객체를 변경할 때 사용
7. DROP

   CREATE의 반대 명령어로 데이터베이스내의 객체를 제거하는 역할.