02. 관계형 데이터베이스

목차

1. 관계형 데이터 모델

- 관계형 데이터 모델(Relational Data Model)
  - 테이블 형식을 이용하여 데이터들을 정의하고 설명한 모델
  - 실세계의 데이터를 누구나 직관적으로 이해할 수 있는 형태로 기술할 수 있는 간단한 방식을 제공
  - 테이블을 릴레이션(Relation)이라고 부른다.

이 름	전화번호	주소	생일
홍길동	010-1234-5678	서울	3월 15일
이건우	010-2132-2345	서울	8월 23일
이몽룡	010-3245-4368	부산	12월 14일

2. 릴레이션(Relation)(1)

- 릴레이션(Relation)
  - 수학적으로, 두 개 이상의 집합으로부터 각 집합을 구성하는 원소들의 순서쌍에 대한 집합을 의미

이름 = {홍길동, 김광식, 박철수, 최용만};

주소 = {서울, 대전, 대구, 부산};

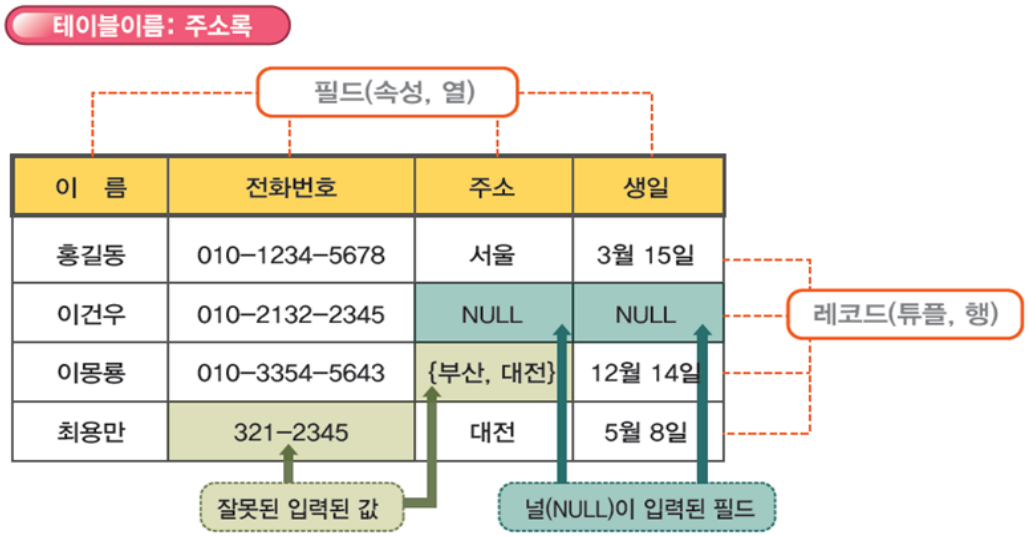
↓

순서쌍 : {<홍길동, 서울>,<김광식, 대전>,<박철수, 대구>,<최용만, 부산>}

이 름	주소
홍길동	서울
김광식	대구
박철수	서울
최용만	광주

2. 릴레이션(Relation)(2)

- 속성(Attribute) – 필드, 컬럼
  - 릴레이션을 구성하는 각 열(Column)의 이름
  - ex) 주소록 릴레이션을 구성하는 속성
    - ✓ 이름, 전화번호, 주소, 생일
- 튜플(Tuple) – 레코드, 행
  - 릴레이션의 각 행
  - ex) 주소록 릴레이션의 한 튜플
    - ✓ <홍길동, 010-1234-1234, 서울, 80/03/15>



2. 릴레이션(Relation)(3)

- 도메인(Domain)
  - 각 필드에 입력 가능한 값들의 범위, 즉 각 필드가 가질 수 있는 모든 값들의 집합
  - 원자값(Atomic Value, 더 이상 분리되지 않는 값)이어야 한다.
  - ex) 주소록
    - ✓ 이름 : 개인의 이름들로 구성된 문자열 집합
    - ✓ 전화번호 : 'ddd-dddd-dddd'의 형식으로 구성된 문자열의 집합
    - ✓ 주소 : 도시를 나타내는 문자열의 집합
    - ✓ 생일 : 'dd월dd일'로 구성된 문자열의 집합

- 널(NULL)
  - 특정 필드에 대한 값을 알지 못하거나 아직 정해지지 않아 입력하지 못한 경우의 필드 값
  - 0이나 공백 문자와는 다르다.

3. 테이블(1)

- 테이블 스키마(Table Schema)
  - 테이블 정의에 따라 만들어진 데이터 구조
  - $R(A_1, A_2, \ldots A_n)$
    - ✓ R : 테이블의 이름
    - ✓ $A_1, A_2, \ldots A_n$: 필드들의 이름

- 차수(Degree)
  - 테이블 스키마에 정의된 필드의 수
    - ✓ 차수 = 1 : 단항 테이블(Unary Relation)
    - ✓ 차수 = 2 : 이항 테이블(Binary Relation)
    - ✓ 차수 = n : n항 테이블(n-ary Relation)

3. 테이블(2)

- 테이블 인스턴스 (Table Instance)
  - 테이블 스키마에 현실 세계의 데이터를 레코드로 저장한 형태
  - 스키마는 한번 정의하면 거의 변함이 없지만 인스턴스는 수시로 바뀔 수 있다.
    – 레코드의 삽입, 삭제, 수정 등

- 기수(Cardinality)
  - 테이블 인스턴스의 레코드 수

학번	주민등록번호	이름	주소	학과명
1292001	900424-1825409	김광식	서울	컴퓨터공학과
1292002	900305-1730021	김정현	서울	컴퓨터공학과
1292003	891021-2308302	김현정	대전	컴퓨터공학과
1292301	890902-2704012	김현정	대구	산업공학과
1292303	910715-1524390	박광수	광주	산업공학과
1292305	921011-1809003	김우주	부산	산업공학과
1292501	900825-1506390	박철수	대전	전자공학과
1292502	911011-1809003	백태성	서울	전자공학과

3. 테이블(3)

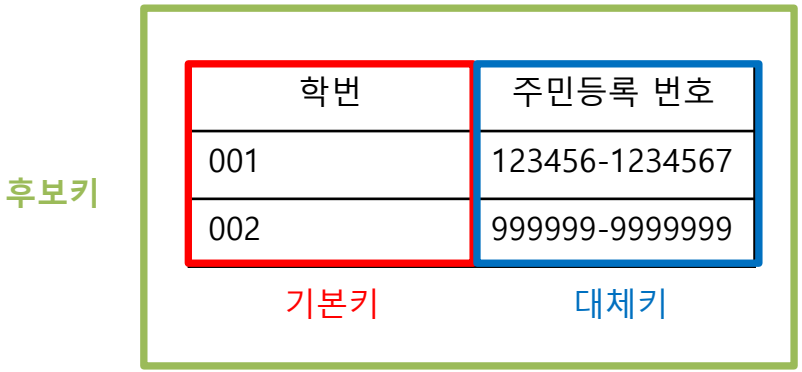
- 중복된 레코드가 존재하지 않는다.
  - 테이블 인스턴스는 레코드들의 '집합'이다.
- 레코드간의 순서는 의미가 없다.
  - 첫번째 레코드, 두번째 레코드 란 표현은 의미가 없다.
- 레코드 내에서 필드의 순서는 의미가 없다.
  - 테이블 스키마는 필드들의 집합으로 표현된다.
  - 첫번째 필드, 두번째 필드 란 표현은 의미가 없다.
- 모든 필드는 원자값을 가진다.

4. 키(Key)

- 키(Key)
  - 필드들의 일부로 각 레코드들을 유일하게 식별해낼 수 있는 식별자(Identifier)
  - 일반적으로 하나의 필드를 지정하여 키로 지정하나, 여러 개의 필드들로 키를 구성할 수도 있다.
    - ✓ 두 개 이상의 레코드로 구성된 키를 복합키(Composite key)라고 한다.
  - 레코드간의 순서가 의미가 없으므로 레코드를 구분하기 위해서는 각 레코드의 값을 이용한다.
  - 예를 들어 신입생 테이블의 학번 또는 주민등록번호 필드는 유일하므로 키가 될 수 있다.
    - ✓ 학과명은 키가 될 수 없다.
  - 관계형 데이터 모델에서 특정 레코드를 구별하거나 탐색하기 위한 유일한 방법.

4.1 키의 종류(1)

- 슈퍼키(Super Key)
  - 아무런 제약 조건 없이 레코드들을 식별할 수 있는 필드의 집합
  - 모든 튜플에 대해 유일성을 만족하지만, 최소성은 만족하지 못한다.
    - ✓ 유일성 : 하나의 키 값으로 하나의 튜플을 유일하게 식별할 수 있어야한다.
    - ✓ 최소성 : 키를 구성하는 속성 하나를 제거하면 유일하게 식별할 수 없도록 꼭 필요한 최소의 속성으로 구성되어야 한다.
- 후보키(Candidate Key)
  - 각 튜플들을 구별하는 데 기준이 되는 하나 혹은 그 이상의 필드 집합
  - 학생 테이블에서 '학번'과 '주민등록 번호'가 후보키가 될 수 있으며, '학번'을 기본키로 선택시 기본키가 되지못한 '주민등록 번호'를 대체키라고 부른다.



4.1 키의 종류(2)

- 기본키(Primary Key)
  - 후보키 중에서 식별자로 정의한 하나의 키
  - 되도록 하나의 필드로 구성된 후보키를 설정하는 것이 유리하다.

- 기본키의 특징
  - 속성이 항상 고유한 값을 가져야한다.
  - 값이 변경될 가능성이 높은 속성은 기본키로 선정하지 않는것이 좋다.
  - 테이블에 기본키는 하나만 만들 수 있다.
  - 널 값을 가질 수 없다.
    - ✓ 기본키는 식별자 기능을 하기때문에 기본키로 정의된 필드가 널값을 갖게 되면 식별을 할 수 없어진다.

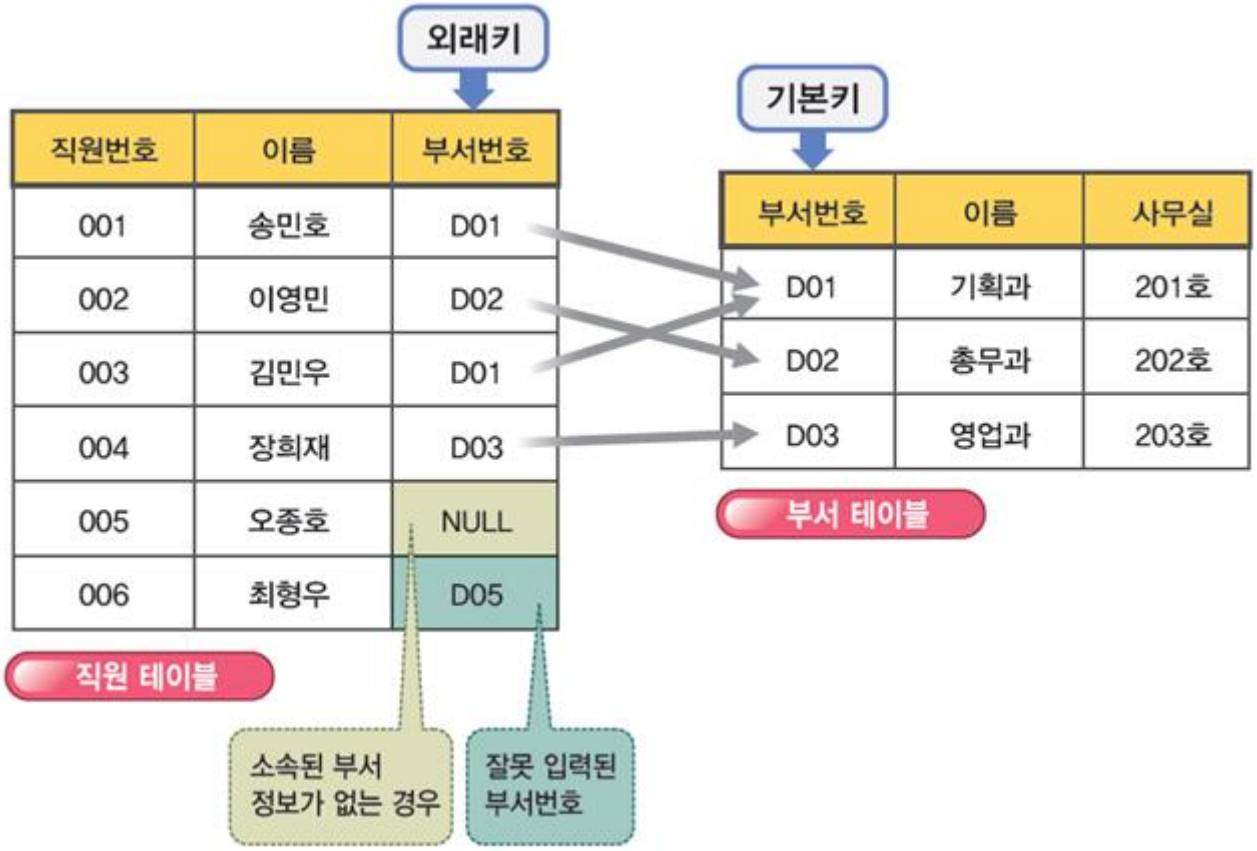
4.1 키의 종류(3)

- 외래키(Foreign Key)
  - 다른 테이블의 기본키를 참조하는 필드 집합
  - 두 테이블 스키마 R_1, R_2에 대해 R_1의 어떤 필드 집합 FK가 다음 두 조건을 만족하면, FK는 R_2의 기본키인 PK를 참조하는 R_1의 외래키가 된다.
    - ✓ FK의 필드들은 테이블 스키마 R_2의 기본키 PK와 동일한 도메인을 갖는다.
    - ✓ R1의 각 레코드의 FK값은 R_2의 레코드 중 하나의 PK값과 일치하거나 널이 된다.
  - 여기서 R1 레코드의 FK값이 널이 된다는 것은 알지 못하거나 아직 결정되지 않았다는 것을 의미한다.

  - R1 : 참조하는 테이블
  - R2 : 참조되는 테이블

4.1 키의 종류(4)

- 외래키(Foreign Key)
  - 직원(직원번호, 이름, 부서번호)
  - 부서(부서번호, 부서명, 사무실)



5. 관계형 데이터베이스 관리 시스템(1)

- 관계형 데이터베이스 관리시스템(RDBMS: Relational Database Management System)
  - 가장 일반적인 형태의 DBMS
  - 관계형 데이터 모델에 기반하여 하나 이상의 테이블로 실세계를 표현한 데이터베이스
  - 테이블은 2차원 형태의 표처럼 볼 수 있도록 행(Row)과 열(Column)로 구성한다.

- 장점
  - 작성과 이용이 비교적 쉽고 확장이 용이하다.
  - 처음 데이터베이스를 만든 후 관련되는 응용 프로그램들을 변경하지 않고도, 새로운 데이터 항목을 데이터베이스에 추가할 수 있다.

5. 관계형 데이터베이스 관리 시스템(2)

- 데이터 딕셔너리(Data Dictionary: DD)
  - 관계형 데이터베이스에서 객체를 정의하게 되면 그 객체가 가진 메타 데이터(metadata)의 정보가 저장되는 곳.
  - 사용자에 의해서 추가, 삭제, 수정되지 못하며 오로지 오라클 시스템에 의해서만 가능.

- SQL(Structured Query Language)
  - 사용자와 관계형 데이터베이스를 연결시켜 주는 표준 검색 언어