5주차(04/07)

무결성 제약 조건

- 엔티티 무결성 제약조건 : 기본키
- 참조 무결성 제약조건 : 외래키
  - 참조하는 테이블의 외래키 값은 참조되는 테이블의 기본키 값에 반드시 존재해야 함
  - 외래키를 통해 두 테이블간의 데이터 무결성을 유지하는 것
  - 제한(restrict) : `ON DELETE RESTRICT`
    - 자식 테이블에서 부모 테이블을 참조하고 있는 값이 있다면 삭제 연산 거절
  - 연쇄(cascade) : `ON DELETE SET CASCADE`
    - 부모 테이블의 데이터가 삭제되면 자식 테이블의 데이터도 함께 삭제
  - 널 값으로 대체(nullify) : `ON DELETE SET NULL`
    - 부모 테이블의 데이터가 삭제되면 자식 테이블의 데이터를 NULL로 설정
  - default값으로 대체 : `ON DELETE SET DEFAULT`

제약 조건	부모 테이블	자식 테이블
입력	제약 없음	부모 테이블에 데이터가 존재하는지 검증
수정	수정하려는 데이터를 자식 테이블에서 참조하고 있는지 검증	부모 테이블에 존재하는 다른 데이터로 변경 가능
삭제	삭제하려는 데이터를 자식 테이블에서 참조하고 있는지를 검증	제약 없음

키의 종류

- 수퍼키(super key) : 유일성
- 후보키(candidate key) : 유일성, 최소성

- 기본키(primary key)
- 대체키(alternate key)
- 외래키(foreign key)

역공학(Reverse Engineer)

- 이미 존재하는 데이터베이스 구조를 바탕으로 다이어그램(ERD)이나 모델을 자동으로 생성하는 것
- 데이터베이스 → 시각화된 설계도(ERD)로 변환하는 과정

SQL

- 데이터 정의어
  - 테이블 생성 : `CREATE TABLE`
  - 테이블 수정 : `ALTER TABLE`
  - 테이블 삭제 : `DROP TABLE`
- 데이터 조작어
  - 데이터 조회 : `SELECT`
  - 데이터 입력 : `INSERT`
  - 데이터 수정 : `UPDATE`
  - 데이터 삭제 : `DELETE`
- 데이터 제어어

개념적 데이터 모델링(개념적 설계)

- 데이터 모델링 3단계 :
  1. 개념적 설계 : ERD
  2. 논리적 설계 : 스키마
  3. 물리적 설계
- 개념적 설계 - ERD(Entity Relationship Diagram : 개체 관계 다이어그램)

    - 1976년 P.Chen 제안
- 개체(엔티티 : entity) ⇒ 테이블
    - 업무과 관여하는 어떤 것(thing)
    - 실세계에 존재하는 유, 무형의 객체
    - 정규 엔티티(strong entity / regular entity) ⇒ 한 겹 사각형
        - 자신의 키 속성을 사용하여 고유하게 엔티티들을 식별할 수 있는 엔티티
        - 독립적으로 존재하면서 고유하게 식별(사람, 사물, 사건, 장소)
    - 약 엔티티(weak entity) : 약한 엔티티 ⇒ 두 겹 사각형(특수 케이스)
        - 자신의 키 속성을 갖기에 충분한 속성을 갖지 못한 엔티티
        - 자체적으로 키를 보유하지 못하는 엔티티

      ⇒ 부분키 사용(기본키 == 소유엔티티의 기본키 + 부분키)

      > 💡 **부분키(Partial key)**
      > - 키와 비슷하지만 완벽하게 키라고 할 수 없음
      > - 약 엔티티에서 사용되는 키(점선 밑줄)

        - 식별 관계로 연결(실선 연결) : 주인의 기본키와 연결
        - 다른 엔티티에 종속 : 소유 엔티티 ⇒ 소유 엔티티 타입이 있어야 함
        - 의존 종속성 : 한 개체의 존재가 다른 개체에 의해 영향 받음
        - 존재 종속 : 해당 엔티티가 없다면 존재하지 않는 종속성
- 속성(애트리뷰트 : attribute) ⇒ 필드
    - 어떤 것이 가지는 속성(attribute)
    - 엔티티 또는 관계가 갖는 성질이나 특성
    - 단순 속성 : 더 이상 다른 속성으로 나눌 수 없는 속성
    - 키 속성(주 식별자) : 엔티티들을 식별할 수 있는 유일한 제약 조건을 갖는 속성(밑줄 표시)
    - 복합 속성 : 두 개 이상의 속성으로 이루어진 속성

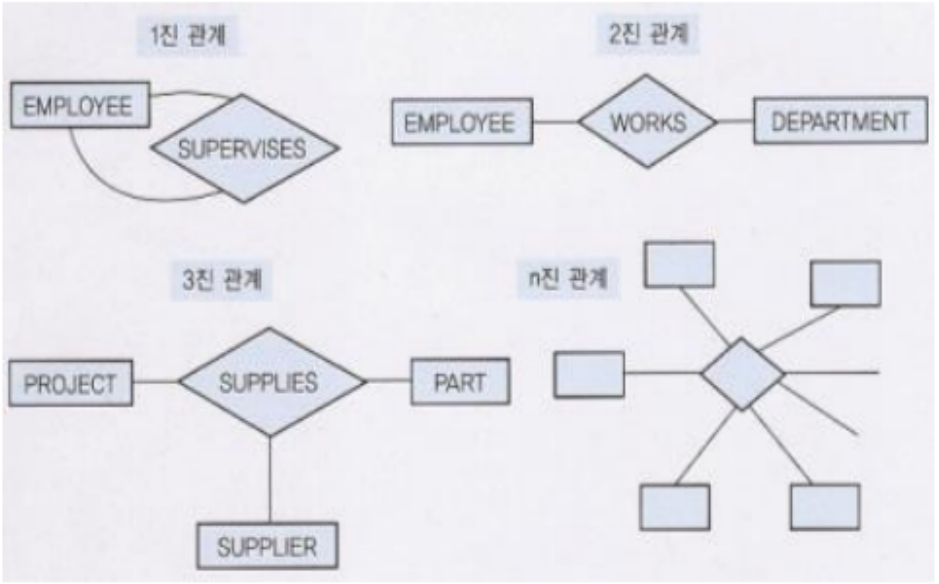
        - ex) 주소(시, 구, 동), 이름(성, 이름)
            - 방법 1 : 한 칸에 그대로 씀
            - 방법 2 : 칸칸이 씀
    - 다중 값(다치) 속성
        - 속성 하나에 여러 개의 값을 가질 수 있는 속성(별도의 엔티티로 분리)
        - 식별 관계로 연결
        - 콤마로 분리 가능

> 💡 **복합 속성 VS 다중값 속성**
>
> - 복합 속성
>   - 속성 안에 하위 속성이 들어 있는 구조(값이 여러 개인게 아님)
>   - ex) **주소**는 **시/도**, **구**, **도로명**, **건물번호** 등으로 속성을 또 나눌 수 있지만 값이 여러 개인 것은 아님
> - 다중 값 속성
>   - 속성 하나에 값이 여러 개 들어갈 수 있음
>   - ex) 한 사람이 여러 개의 **이메일**을 가질 수 있음 ⇒ 이메일 밑에 하위 속성이 있는게 아니라 말 그대로 값이 여러 개

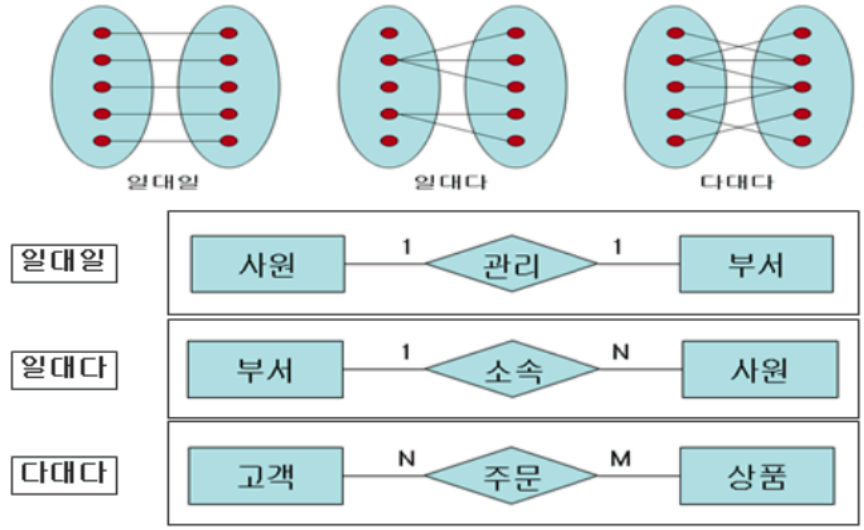
    - 유도 속성
        - 실제 값이 저장된 것이 아니라 저장된 값으로부터 계산해서 얻은 값을 사용하는 속성(점선 표시)
        - 대부분 논리적 설계에서는 제거

기호	의미
	개체 타입
	약한 개체 타입
	관계 타입
	식별 관계 타입
	전체 참여 개체 타입
	속성
	식별자 속성
	부분키 속성
	다중 값 속성
	복합 속성
	유도 속성

- 관계(relationship) : 업무가 관여하는 어떤 것 간의 관계(relationships) ⇒ 차수
  - 관계 차수 : 관계와 이어진 엔티티 수 N ⇒ N진 관계



  - 관계 카디날리티
    - 일대일 관계(1 : 1)
    - 일대다 관계(1 : N / N : 1) ⇒ N쪽으로 외교관이 감
    - 다대다 관계(N : M)



- 관계 종속성
  - 필수(Mandatory) ⇒ 전체 참여
  - 선택(Optional) ⇒ 부분 참여

💡 **스키마**

- 데이터베이스를 구성하는 데이터구조와 제약조건에 대한 명세를 구체적으로 기술한 것
  - 개체(Entity)
  - 속성(Attribute)
  - 관계(Relationship)
  - 제약조건

데이터 모델의 구성요소(D = <S , O, C>)

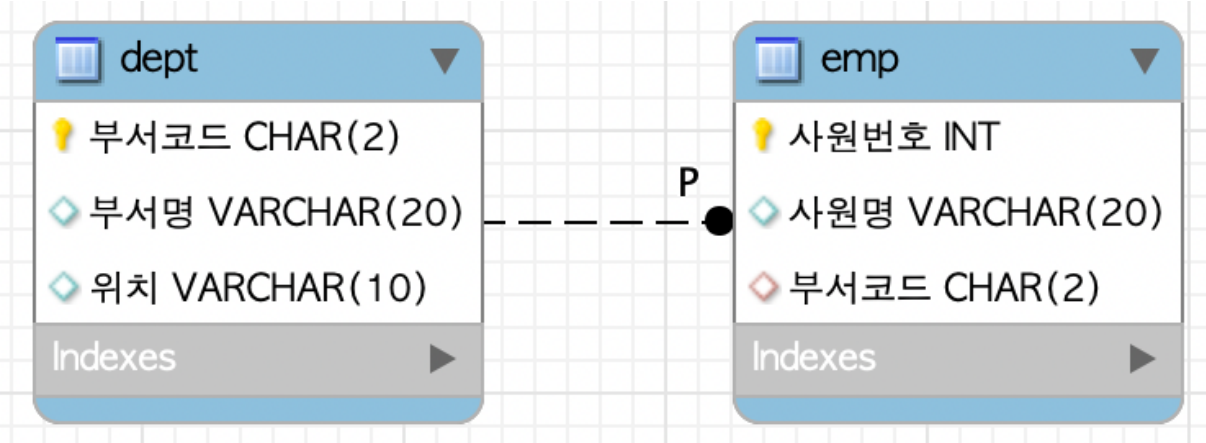
- 구조(Structure)
- 연산(Operation)
- 제약조건(Constraint)

[실습]

```
create schema W05 default character set utf8mb4;
use W05;

-- (MySQL) safe mode 해제
-- 0은 해제, 1은 설정
set SQL_SAFE_UPDATES = 0;

-- 테이블 목록 조회
show tables;

-- 테이블 구조 확인
desc emp;
desc dept;
desc 학생;
desc 학과;

-- 테이블 삭제
drop table if exists emp;
drop table if exists dept;
drop table IF EXISTS 학생;
drop table IF EXISTS 학과;
```

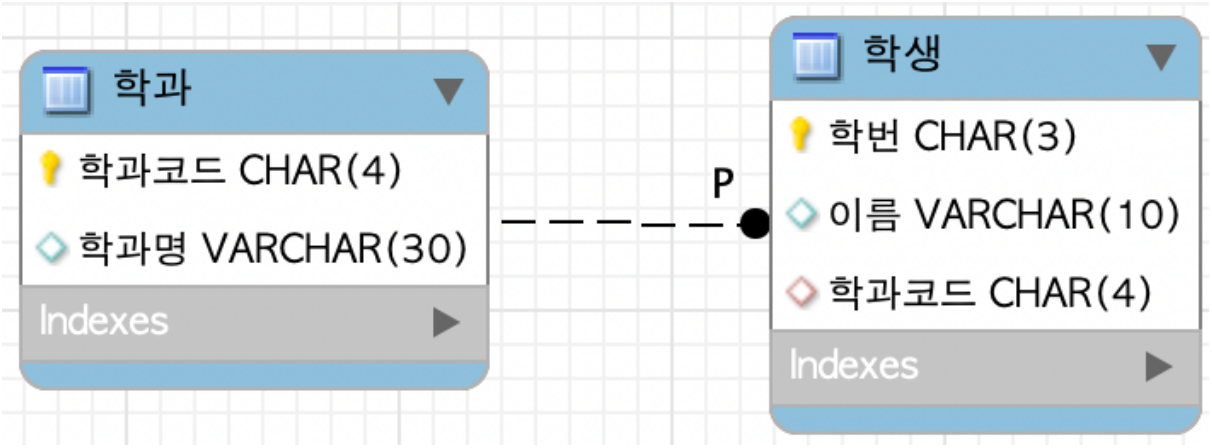
```
-- ********** dept, emp table **********
-- dept(부서코드, 부수명, 위치)
CREATE TABLE dept (
      부서코드 char(2) NOT NULL ,
      부서명 varchar(20) ,
      위치 varchar(10) ,
      PRIMARY KEY (부서코드)
);

insert into dept values('AA','총무부','서울');
insert into dept values('BB','영업부','대전');
insert into dept values('CC','기획부','서울');
```

```
select * from dept;

-- emp(사원번호,사원명, 부서코드)
CREATE TABLE emp (
      사원번호 int NOT NULL ,
      사원명 varchar(20) ,
      부서코드 char(2)      ,
      PRIMARY KEY (사원번호) ,
      FOREIGN KEY(부서코드)
          REFERENCES dept(부서코드)
);

insert into emp values(100,'신재영','BB');
insert into emp values(101,'오주원','AA');
insert into emp values(102,'이대은','BB');

select * from emp;
```




```
-- ********** 학과, 학생 table **********
-- 학과(학과코드, 학과명)
create table 학과 (
   학과코드 char(4) not null,
   학과명 varchar(30),
   constraint pk_학과
      primary key(학과코드)
);
```

```
insert into 학과 values('1001','컴퓨터학과');
insert into 학과 values('2001','체육학과');

select * from 학과;

-- 학생(학번, 이름, 학과코드)
create table 학생 (
    학번 char(3) not null,
    이름 varchar(10),
    학과코드 char(4),
    primary key(학번),
    foreign key(학과코드)
        references 학과(학과코드)
);

insert into 학생 values('501','박지성','1001');
insert into 학생 values('401','김연아','2001');
insert into 학생 values('402','장미란','2001');
insert into 학생 values('502','추신수','1001');

select * from 학생;

-- 학과코드 2001을 삭제하시오.
delete
from 학과
where 학과코드 = 2001;

-- 학과코드 1001을 A001로 수정하시오.
update 학과
set 학과코드 = 'A001'
where 학과코드 = '1001';
```


학과
학과코드 CHAR(4)
학과명 VARCHAR(30)
Indexes
P
학생
학번 CHAR(3)
이름 VARCHAR(10)
학과코드 CHAR(4)
Indexes