**Midterm Assignment: A university database system**

**12181465 김재욱**

**1. 요구사항 분석하기**

과제에서 해석한 요구사항은 다음과 같다.

* 교수는 id, name age를 가진다.
* 프로젝트는 number ,sponsor name , starting date, budget를 가진다.
* 대학원생은 id, name, age, degree program을 가진다.
* 각 프로젝트는 한 명의 교수(advisor))에게 관리(manage)된다.
* 각 프로젝트는 한 명의 교수(investigator)가 맡는다(work).
* 교수는 여러 개의 프로젝트를 관리할 수 있다.
* 대학원생은 여러 개의 프로젝트에 참여할 수 있으며 각 프로젝트마다 다른 교수가 관리(manage)해야한다.
* 각 프로젝트는 대학원생들이 조교로서 맡는다(work).
* 부서는 department number, department name, main office를 가진다.
* 부서는 교수에 의해 관리된다.
* 대학원생은 하나의 전공을 가진다.

이 요구사항에 대해 여러 관계를 포함하여 요악한다.

하나의 프로젝트에는 교수가 관리(manage)하며 다른교수와 함께 대학원생들이 일(work)한다

이때 교수는 한번에 여러 프로젝트를 관리(manage)할 수 있다

대학원생들은 여러 프로젝트에서 일할 수 있지만 관리(manage)하는 교수가 같은 프로젝트에 일할 수 없다.

각 부서는 한 교수에 의해 관리(manage)되며 대학원생들은 하나의 부서를 전공했다.

**2. 개념적 설계로 E-R 다이어그램 만들기**

요구사항을 기반으로 데이터 베이스를 구성하는데 필요한 개체와 속성, 개체간의 관계를 확립하여 ERD를 생성한다.

**2-1. 개체 및 속성 추출**

* 개체 - 실제 세계에서 독립적으로 존재하는 실체이며 실제 존재와 개념적 존재로 나뉜다.
* 속성 - 개체를 기술하는 속성

|  |  |
| --- | --- |
| 개체 | 속성 |
| Professor | id, name, age |
| Project | number, sponsor name, starting date, budget. |
| Graduate student | id, name, age, degree program |
| Department | department number, department name, main office |

**2-2. 개체 간의 관계 추출**

* 개체 간의 관계를 동사로 나타낸다.
* 1:1 ,1:N ,N:M
* 전체참여, 부분참여

|  |  |  |
| --- | --- | --- |
| 관계 | 관계에 참여하는 개체 | 유형 |
| Manage\_of | Professor : 교수는 여러 프로젝트를 관리 할 수 있다(부분참여)  Project : 모든 프로젝트는 한 교수에 의해 관리 되어야한다. (전체참여) | 1:N |
| Work\_on\_P | Professor : 교수는 한 프로젝트를 진행한다(부분참여)  Project : 모든 프로젝트는 한 교수에 의해 진행되야 한다.(전체참여) | 1:1 |
| Work\_on\_S | Graduate student : 교수가 겹치지 않는 한 여러 프로젝트를 참여할 수 있다. (부분참여)  Project : 모든 프로젝트에 대학원생이 있어야 한다.(전체참여) | M:N |
| Represent | Department : 한 부서는 한 교수님이 관리한다.(전체참여)  Professor : 교수는 한 부서의 관리자가 될 수 있다.(부분참여) | 1:1 |
| Belong\_to | Graduate student : 대학원생은 반드시 전공을 가진다(.전체참여)  Department : 한 부서는 같은 전공의 대학원생을 소속시킨다.(부분참여) | 1:N |

**2-3 속성 분류과 key 분류**

Atomic attribute : 더 이상 독립적인 의미로 나눌수 없는 속성

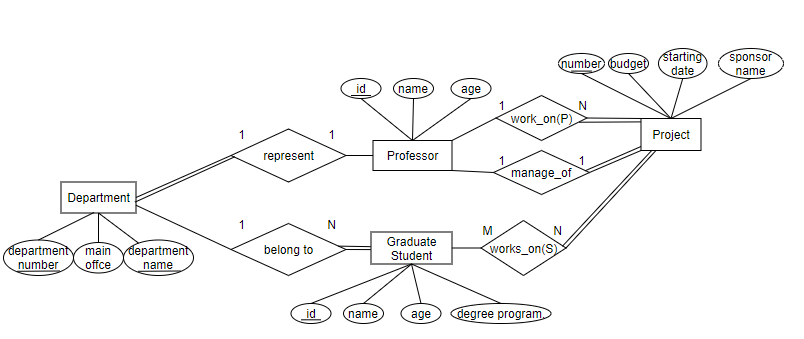
Composite attribute : 여러가지 독립적인 의미를 가진 구성요소로 나눌 수 있는 속성

Key attribute : 개체마다 다른 속성값을 가져 개체를 식별 할 수 있는 속성

|  |  |  |  |
| --- | --- | --- | --- |
| entity | Atomic attribute | Composite attribute | Key attribute |
| Professor | id, age | Name(first name, last name) | id |
| Project | number, budget | Sponsor name(first name, last name),  starting date(year,month,date), | number |
| Graduate student | id, age | Name(first name, last name) | id |
| Department | department number, department name, main office |  | department number, department name, |

(composite attribute에 대한 언급이 없었기 때문에 임의로 나누지 않고 독립적인 의미를 가진다고 간주하였다.)

**2-4 ER 다이어그램 생성하기**



**3. 릴레이션 스키마**

**3-1 릴레이션 스키마로 표현하기**

ERD를 통해 데이터베이스를 mysql로 구현하기 위해 릴레이션 스키마로 변환 헤야 한다.

다음과 같은 과정을 통해 ERD로부터 릴레이션 스키마로 변환한다.

* 모든 개체는 릴레이션으로 변환한다.

|  |
| --- |
| Professors |
| prof\_id | prof\_name | prof\_age |

|  |
| --- |
| Department |
| department\_number | department\_Name | main\_Office |

|  |
| --- |
| Graduate students |
| s\_id | s\_name | s\_age | s\_degree\_program |

|  |
| --- |
| Project |
| proj\_number | proj\_sponsor\_name | proj\_starting\_date | proj\_budget |

위에서 분석 한 것 처럼 각 튜플을 구분 할 수 있는 key attribute들 중 하나를 선택하여 primary key 로 설정한다. 이때 선택 되지 못한 다른 key들을 alternate key라고한다.

각 릴레이션의 id 와 number를 primary key로 설정한다.

* N:M 관계는 관계이름을 릴레이션으로 하고 참여하는 속성들을 릴레이션의 속성으로 변환한다.

프로젝트에서 일하는 대학원생은 여려명일 수 있기 때문에 다중값을 가진다. mysql에서는 여러 개의 속성값을 가질 수 없으므로 이런 경우 하나의 독립된 릴레이션을 만드므로써 표현할 수 있다.

work\_on 이라는 이름을 가진 릴레이션을 만들고 project\_num과 strudent\_id를 속성으로 삽입하면 하나의 프로젝트는 여러 대학원생을 표현 할 수 있다.

이때 primary key 는 다른 관계에 의해 추가 속성이 있으므로 나중에 이야기 한다.

|  |
| --- |
| Work\_on |
| Project\_num | Student\_id |

* 1:1 관계는 관계를 가지는 한 릴레이션의 속성으로 추가하고 이 속성은 관계를 가지는 다른 릴레이션의 primary key를 참조한다.

다음의 관계들은 위의 표에서 명시한대로 1:1 관계를 가지기 때문에 각 개체의 foreign key로 표현한다.

foreign key란 다른 릴레이션의 primary key를 가르키는 데이터의 참조무결성(referential integrity)을 확인하기 위해 사용된다. 참조하는 속성이 foreign key이다.

**Represent 관계**

각 부서는 교수가 관리하며(chairman) 보통 각 부서의 개수보다 교수의 수가 더 많으므로 속성은 professors의 릴레이션에 포함되면 null값이 많이 생기게 된다.

때문에 departments 릴레이션에 manage\_p\_id 라는 속성을 추가해 professors의 primary key를 참조한다.

|  |
| --- |
| departments |
| department\_number | department\_Name | main\_Office | manage\_p\_id |

**Work\_on\_P 관계**

Work\_on의 관계에서 각 프로젝트는 대학원생을 필수로 가지므로 projects에서 모든 속성값을 work\_on 릴레이션의 project\_num이 가진다. **Work\_on\_P** 표현하기 위해 work\_on 릴레이션에 모든 project\_num이 있으므로 professor\_id 라는 속성을 추가한다.

관계의 조건으로 프로젝트에 참여하는 대학원생은 같은 교수가 관리하는 프로젝트를 동시에 할 수 없다. 그러므로 프로젝트를 관리하는 교수의 기본키를 추가한다. 대학원생과 교수의 id를 primary key로 묶어서 사용한다면 한 대학원생이 같은 교수가 관리하는 프로젝트를 삽입 할 수 없다.

여러 primary key를 동시에 사용하는 것을 composite key라고 한다. 즉 하나의 속성으로만 튜플을 구분 할 수 없을 때 여러 속성을 사용하여 구분될 수 있도록 한다.

.

|  |
| --- |
| work\_on |
| project\_num | student\_id | professor\_id |

* 1:N 관계는 1에 해당하는 개체가 N에 해당하는 개체에 속해 foreign key로 표현한다.

**Manage\_of 관계**

1:N = professor : project 이므로 한 교수가 여러 프로젝트를 관리 할 수 있다.

이에 따라 N에 해당하는 개체인 projects에 manage\_p\_id라는 속성을 추가한다.

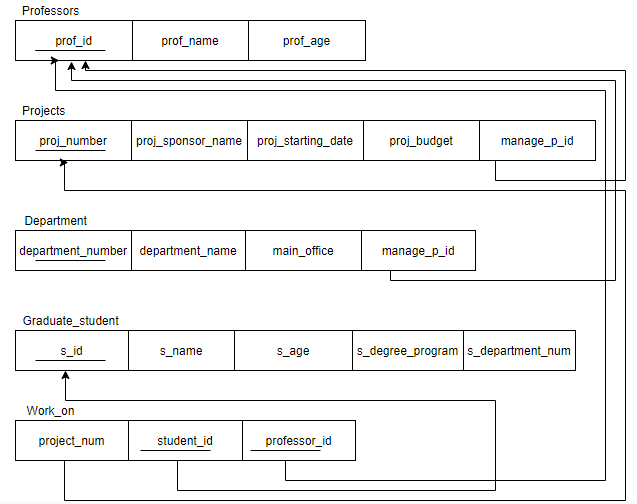
|  |
| --- |
| projects |
| proj\_number | proj\_sponsor\_name | proj\_starting\_date | proj\_budget | manage\_p\_id |

**Belong\_to 관계**

1:N = department : Graduate student 이므로 한 부서가 여러 대학원생을 소속 시킬 수 있다. 이에 따라 N에 해당하는 개체인 Graduate student에 s\_department\_num라는 속성을 추가시킨다.

|  |
| --- |
| Graduate students |
| s\_id | s\_name | s\_age | s\_degree\_program | s\_department\_num |

**3-2. 관계 데이터베이스 스키마**



**4. 물리적 스키마 구현**

**데이터베이스를 생성 스크립트 첨부(DB.sql)**

**튜플 추가 스크립트 첨부(university\_insert.sql)**