COSE371-03

DATABASE

Prof. Yondohn Chung

Term Project Report

The Course Evaluation System

1. **개요**
2. **Entity Relationship Model & Relation Schema**
3. **페이지 구성 및 기능 설명**

**Inho Kim 2012130888**

**Korea University**

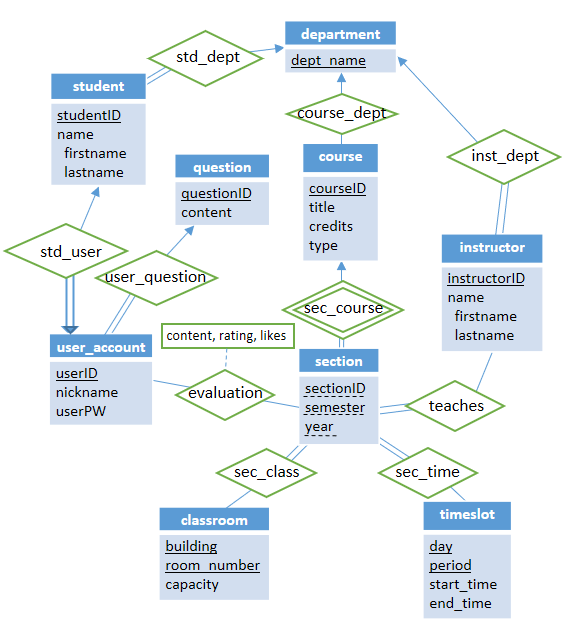
**English Language and Literature / College of Information and Communication**

1. **개요**

본 강의평가 시스템은 기본적으로 다음과 같은 가정 하에 만들어졌다.

* 회원제로 운영이 되며, 학생들을 위한 서비스이기 때문에 학생 회원만 고려하고, 교수 회원은 배제한다.
* 학생 한 명당 하나의 계정만 만들 수 있다. 여기서 학생이란, 학교의 학생 전체가 아니라 해당 강의평가 시스템을 사용하는 학생으로 한정한다.
* 강의평가는 한 계정당 하나의 강의에 대해 최대 한 번만 작성할 수 있다.
* 추천제도가 존재하며, 이는 작성된 강의평가에 대하여 회원들이 추천을 할 수 있는 시스템이다. Facebook의 ‘좋아요’ 기능과 유사하다. 여러 회원들로부터 받은 추천 수는 합산되어 강의평가를 열람할 때 정렬순서에 반영된다. 그런데, 한 계정당 하나의 강의평가에 대하여 여러 번 추천을 할 수 있다. 단, 자신이 작성한 강의평가에 대해서는 추천을 할 수 없도록 제한한다.
* 학교 데이터베이스 자체가 아니라 단순 강의평가 서비스이기 때문에 교수 정보, 강의 정보, 강의실에 따른 수용 인원 등의 학교 관련 data들은 클라이언트 서비스단에서 생성/변경되는 것이 아니라, 서버단에서 미리 주어진다고 가정한다. 단, 관리자 페이지를 두어서 필요에 따라 data에 대한 부분적인 수정이나 테스트용 data를 제한적으로 생성할 수 있도록 한다.

1. **Entity Relationship Model & Relation Schema**

앞서 살펴본 여러 조건들을 충족하면서, 강의평가 시스템에 필요한 전체적인 database 구조를 형성하기 위하여 우측의 Entity Relationship Model을 구성하였다. 강의평가 시스템 자체에 필요하지 않은 항목들은 최대한 배제하였지만, entity, relationship set들을 표현하는데 도움을 주는 항목들은 일부 남겨두었다.

User\_account는 하나의 회원 계정을 의미하며, 학생 한 명당 하나의 계정을 생성할 수 있다는 조건을 만족하기 위하여 student와 One-to-one의 relationship을 설정하였다. 그리고 교수 회원은 배제하였기 때문에, user\_account는 반드시 하나의 학생 정보를 보유해야 한다.

Figure 2-1 E-R Diagram

question이라는 entity set은 회원이 비밀번호를 잊어서 비밀번호를 찾기 위해 필요한 정보를 포함한다. 이는 처음 회원가입을 할 때 각 계정마다 하나씩 설정해두도록 한다. 따라서 one-to-one 관계가 성립된다.

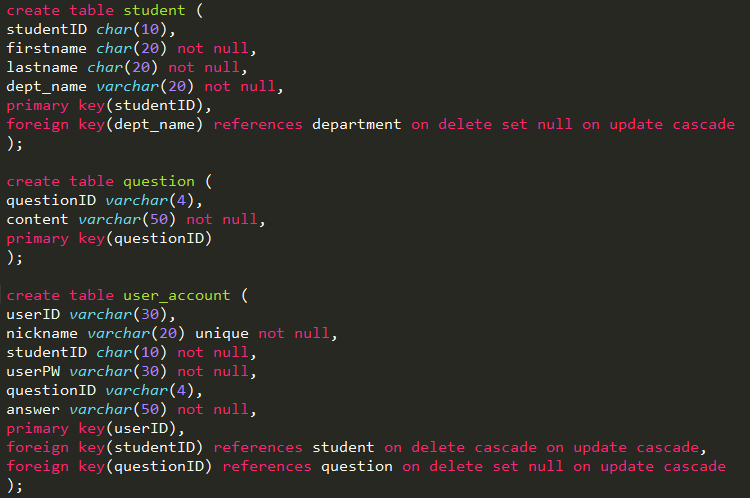
이 학생 회원 계정에 대한 내용을 relation으로 생성할 때, not null 조건을 적극활용하여 사용자가 빈 값을 집어넣으므로 인해 쓸데 없는 데이터들이 생성되는 것을 예방했다. 그리고 각각의 foreign key들에 cascading action들을 설정해 두어서 참조되는 relation의 값이 변경되거나 삭제될 경우 참조하는 쪽에서도 그 내용이 반영되도록 하였다.

Figure 2-2 학생 회원 계정 테이블 생성

예를 들어, 학과가 사라졌다고 해서 소속된 학생들이 모두 사라진다거나, 비밀번호 찾기 질문이 삭제되었다고 해서 계정 자체가 사라지는 경우가 있어서는 안되기 때문에 이런 경우에는 on delete set null을 사용하였다. 따라서 user\_account의 attribute인 questionID는 not null로 설정되어 있지 않다. 이는 비밀번호 찾기 질문에 대한 삭제와 수정 권한이 전적으로 database 관리자에게 있다는 가정하에 내려진 결론이다. E-R model에서는 user\_account가 question을 반드시 하나 포함해야 하는 total의 관계였지만, database 관리자가 주의하여 이 값이 null이 되지 않도록 해야 하는 것이다. 웹페이지 상에서는 회원 가입시에 반드시 질문 중 하나를 선택해야 하기 때문에 사용자에 의해 null이 될 일은 없으므로 문제가 database 관리자가 부주의하지 않은 이상 문제가 발생할 일은 없다.

그런데, user account는 반드시 하나의 student 정보를 포함해야 한다는 조건을 만족시키기 위하여 이 경우에는 on delete cascade라는 조건을 설정하였다.

evaluation이라는 relationship set은 강의평가를 표현한 것이다. 한 강의가 여러 회원에 의하여 평가될 수 있고, 한 회원은 여러 강의를 평가할 수 있기 때문에 many-to-many인 relationship이다. 이 evaluation을 별도의 entity로 표현하지 않은 이유는 개요에서 언급한 조건때문이다. 한 계정은 하나의 강의에 대하여 최대 한 개의 강의평가만 작성할 수 있기 때문에 굳이 별도의 primary key로 evaluation을 참조할 필요없이 user\_account와 section의 primary key를 합한 형태를 primary key로 사용하면 된다. 대신 강의평가라는 것은 내부적으로 내용과 평점, 즉, content와 rating 등의 정보를 포함하고 있기 때문에 relationship set 자체에 attribute로서 이를 표현하였다.

Likes 라는 것은 추천을 의미하는데, 원래는 회원이 특정 강의평가에 대해 추천을 하는 것이므로 별도의 relationship set으로 표현해 줄 수 있지만, 본 강의평가 시스템에서는 누가 어떤 강의를 추천하였는지는 고려대상에서 제외하고 단순히 어떤 강의평가가 받은 추천 수를 합산한 결과만 필요하다고 판단하였다. 따라서 likes는 별도의 relationship set으로 생성하지 않고, evaluation의 attribute로 포함시켰다.

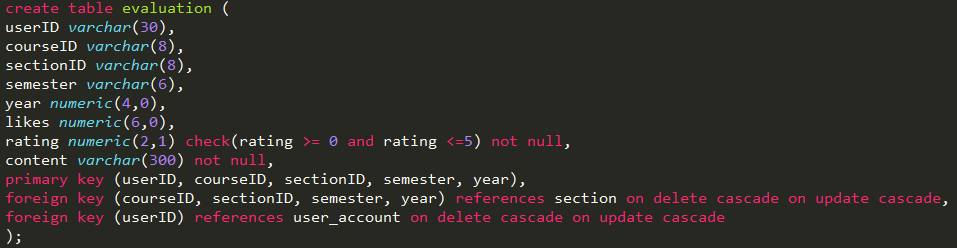


Figure 2-3 evaluation 테이블 생성

이 강의평가는 작성자의 계정 또는 강의가 삭제될 경우에 존재의 의미가 없어지게 된다. 따라서 relation으로 변환할 때 user\_account와 section의 primary key들을 foreign key로 참조하고, 삭제 및 수정되는 경우에 대하여 cascading action을 설정하였다. 또한 평점은 [0,5] 범위의 값을 가지며 null이 되지 않도록 조건을 설정하였다. 내용 (content)도 역시 아무 값도 없으면 의미가 없기 때문에 not null 조건을 설정하여 의미없는 빈 강의평가가 생성되지 않도록 하였다.

Course\_dept relationship set의 경우 특이하게 partial-partial 관계인데 이는 course entity set의 type이라는 attribute를 고려한 결과이다. Type은 해당 course가 전공과목인지 비전공과목인지를 구분해주는 attribute이다. 전공과목인 경우 어떤 학과의 전공인지를 명시할 수 있지만, 교양수업과 같은 비전공과목은 따로 학과 정보를 포함하지 않을 수도 있다. 따라서 이를 partial한 관계라고 판단하였고 relation으로 변환할 때는 다음과 같이 not null 조건을 설정하지 않았다.

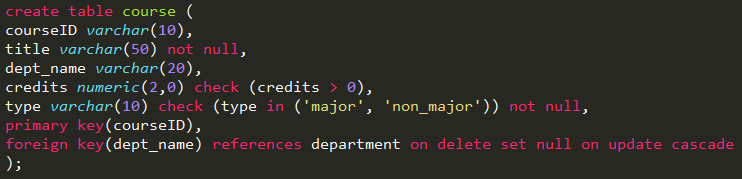


Figure -4 course 테이블 생성

department relation 쪽에서 값이 수정될 시에 course relation에도 반영해주기 위해 on update cascade라는 조건을 설정하였는데, 삭제의 경우에는 on delete set null로 설정하였다. 그 이유는 전공과목이었던 과목이 교양과목으로 재설정되는 경우가 발생할 수 있다고 고려했기 때문이다.

Sec\_time과 sec\_class는 section 하나가 개설되는 시간과 장소 정보를 포함하는 relationship set이다. 하나의 강의는 여러 시간 혹은 요일 및 여러 장소에서 에서 진행될 수 있다. 다시 말해서, 어떤 강의는 요일 별로 다른 강의실에서 진행될 수도 있다. 즉, 하나의 section에 대하여 시간 정보와 공간 정보는 multi value로 처리해야 한다. 따라서 section과 timeslot, classroom 이라는 entity set 세 개를 sec\_time\_class라는 하나의 relation으로 만들었다. 이 sec\_time\_class에서 foreign key로 timeslot과 classroom에 대한 참조를 통해 data를 저장할 수 있도록 구성하여 하나의 section이 여러 timeslot과 여러 classroom에 개설될 수 있다는 것을 표현하였다. 이때, timeslot 또는 classroom relation에서 수정 혹은 삭제가 일어날 경우 sec\_time\_class relation에도 반영이 되어야 하기 때문에 on delete set null과 on update cascade 두 가지의 cascading action을 설정하였다. timeslot이나 classroom에 대한 foreign key와는 달리 section에 대한 foreign key에는 on delete cascade 조건을 설정하였는데, 이는 section 자체가 사라질 경우 해당 section에 대한 시간과 공간 정보가 유명무실해진다고 판단했기 때문이다.

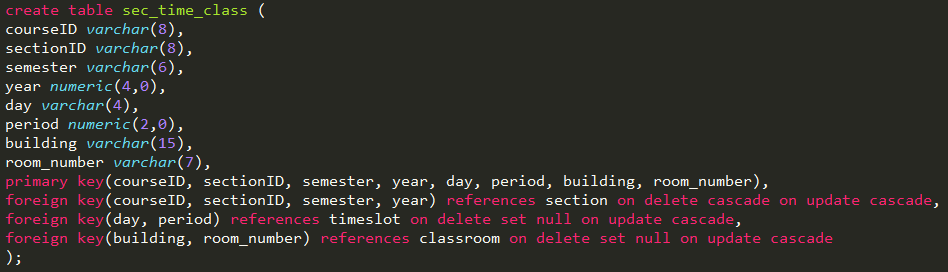


Figure 2-5 sec\_time\_class 테이블 생성

Timeslot과 classroom은 별도의 relation으로 생성하였으며, 특히 timeslot같은 경우 tuple 하나가 특정 요일의 특정 교시 하나를 표현하도록 하였다. 이렇게 구성함으로써 하나의 section이 여러 개의 timeslot을 multi value로 포함한다는 개념이 성립되는 것이다. Classroom같은 경우도 특정 건물의 특정 강의실 하나를 tuple 하나가 표현하도록 하여서, section하나가 multi value로서 여러 tuple들을 포함할 수 있도록 하였다.

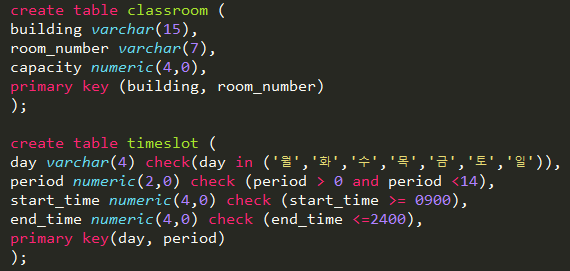


Figure 2-6 timeslot 및 classroom 테이블 생성

앞서 논의한 바와 같이 Entity Relation Model의 전체적인 구조와 여러 세부적인 조건들을 만족하기 위하여 relation들에 최대한 다양한 조건들을 설정하였다. 언급되지 않은 relation들은 수업시간에 다뤘던 학교 database와 구조적으로 큰 차이가 거의 없기 때문에 별도로 설명하지 않았다. 예를 들어, instructor나 teaches, 또는 department같은 경우의 relation들은 학교 database와 유사하게 구성하였다. 최종적으로 생성한 relation은 총 12개이며 전체적인 Relation Schema는 다음과 같다.

**department = (dept\_name)**

**student = (studentID, firstname, lastname, dept\_name)**

**question = (questionID, content)**

**user\_account = (userID, nickname, studentID, userPW, questionID, answer)**

**instructor = (instructorID, firstname, lastname, dept\_name)**

**course = (courseID, title, dept\_name, credits, type)**

**classroom = (building, room\_number, capacity)**

**timeslot = (day, period, start\_time, end\_time)**

**section = (courseID, sectionID, semester, year)**

**sec\_time\_class = (courseID, sectionID, semester, year, day, period, building, room\_number)**

**teaches = (instructorID, courseID, sectionID, semester, year)**

**evaluation = (userID, courseID, sectionID, semester, year, likes, rating, content)**

Figure 2-7 Relation Schema

1. **페이지 구성 및 기능 설명**
2. **회원 계정 관련**

회원 계정과 관련해서는 php의 Session을 활용하여 다양한 기능을 구현하였다. 가장 기본적인 로그인 화면 (/login.php)에서는 사용자로부터 아이디와 비밀번호를 입력받고, 이 정보를 login\_submit.php로 전달한다. login\_submit.php에서는 database 내에서 계정정보를 확인하여 존재하지 않는 아이디를 입력했는지, 혹은 아이디는 맞게 입력했지만 비밀번호를 잘못 입력했는지를 체크하여 사용자에게 feedback한다. Database 내에 알맞은 아이디와 비밀번호의 set이 존재한다면 로그인을 허용하고 session을 활성화시킨다.

Figure 3-(1)-1 login.php

비밀번호 찾기 기능 (/findpw.php)은 회원가입시에 설정해놓은 아이디, 학번 및 비밀번호 확인용 질문과 그에 대한 답변을 입력받아서 계정의 비밀번호를 알려주는 서비스이다. 사용자로부터 입력받은 정보를 findpw\_submit.php로 넘겨서 database에의 쿼리를 통해 비밀번호 값을 찾아서 사용자에게 알려준다.

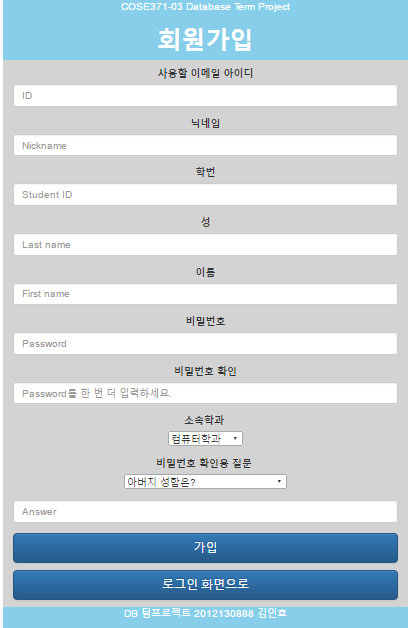
회원가입 기능 (/signup.php)은 위의 두 기능과 기본적으로 같은 구조를 갖는다. 사용자로부터 입력받은 정보를 실제 데이터를 핸들링하는 signup\_submit.php로 넘겨서 새로운 계정을 생성해준다. 이때, 학생 한 명당 최대 하나의 계정만을 만들 수 있다는 조건을 충족시키기 위해서 학번을 입력하도록 하여 이미 database내에 존재하는 학번일 경우에 가입을 허가하지 않도록 한다.

Figure 3-(1)-2 findpw.php

이외에 이메일 아이디와 닉네임에 대하여도 중복체크를 하여 중복된 아이디 혹은 중복된 닉네임의 계정이 생기지 않도록 unique 조건을 만족시켜주었다. 그리고 소속학과 및 비밀번호 확인용 질문은 drop down list로 제공하여 사용자가 null값을 입력할 수 없도록 방지하였다. 그리고 다른 항목들에 대해서도 기본적으로 input 값이 비어있는지를 확인하여 하나라도 비어있을 경우 가입이 허가되지 않도록 하였다.

Figure 3-(1)-3 signup.php

로그인이 정상적으로 완료되었다면 /mainpage.php로 이동한다. 기본적으로 화면에 간편검색 및 상세검색창을 띄워주며, 개인정보 수정, 로그아웃, 회원탈퇴 등 계정 정보에 관련된 기능들도 포함한다. 좌측상단의 링크는 관리자페이지로 이동하기 위한 것인데, Session 정보를 파악해서 로그인된 닉네임이 admin일 경우에만 이 링크를 렌더링하고, admin이 아닌 다른 계정이 로그인 한 경우 아예 보여주지 않는다.

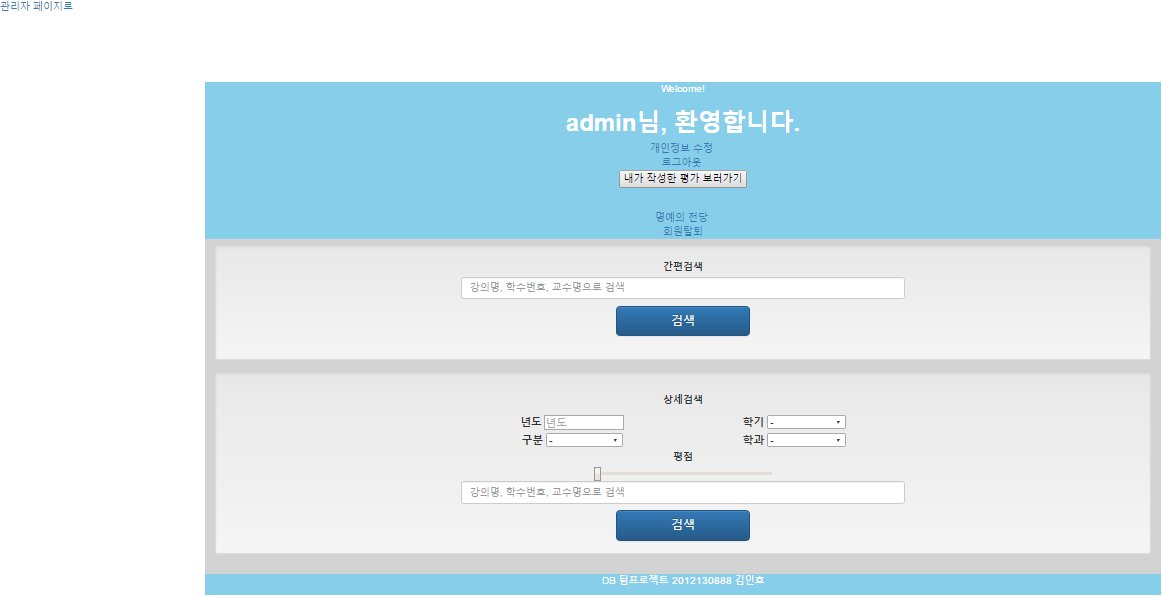
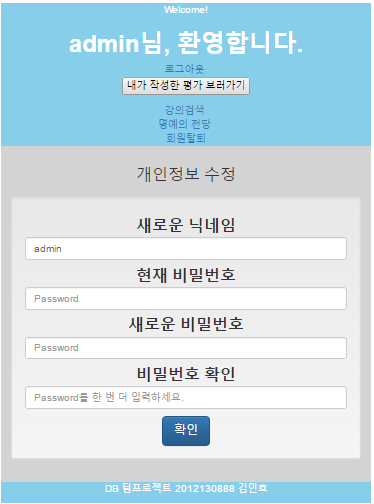


Figure 3-(1)-4 mainpage.php



개인정보 수정을 선택하였을 경우 changeuserinfo.php 페이지로 이동하여 회원이 닉네임이나 비밀번호를 변경할 수 있도록 한다. Update SQL query를 활용해 변경을 해주는데, 실제 query문을 실행하기 전에 변경하려는 닉네임이 이미 존재하는지를 판단하여 unique 조건을 만족해주었다. 대신 닉네임을 변경하고 싶지 않아서 원래의 닉네임을 그대로 입력하는 경우에도 중복 체크 조건에 걸리는 버그를 해결하기 위해 원래 설정되어 있던 닉네임을 입력한 경우에는 update query를 실행시키지 않도록 하였다.

Figure 3-(1)-5 changeuserinfo.php

그리고 회원탈퇴를 클릭할 경우에는 다시 한 번 탈퇴할 것인지를 사용자에게 물어보고, delete query를 이용해서 계정 정보를 삭제한다. 이때 Relation에 설정해 둔 on delete cascade 조건 덕에 student 정보만 삭제하면 자동으로 계정 정보도 삭제되기 때문에, 계정 정보에 포함되어 있는 studentID 값을 통해 student 정보를 삭제한다.

1. **검색 기능 및 강의 평가 관련**

Figure 3-(1)-4의 메인페이지에서 간편 검색 혹은 상세 검색을 통해 사용자가 데이터를 전달하면, database에 query를 수행하여 search\_result.php라는 페이지로 이동한 뒤 결과물로서 강의 목록을 보여준다. 사용자가 입력하지 않은 값에 대해서는 null로 처리하지 않고 검색의 조건에서 제외해줌으로써 검색의 유연성을 높였다. 그리고 data에 대한 접근성 및 테스트의 유용성을 위하여 사용자가 아무 정보도 input으로서 제공하지 않으면, 아무 조건도 고려하지 않고 모든 강의목록을 렌더링하도록 구성하였다.

그런데, Query문을 수행할 때는 여러가지 복잡한 조건들을 생각해야 한다. 예를 들어, 교수명 검색의 경우 성 혹은 이름이 두 글자 이상인 경우도 고려해야 하며, dept\_name이라는 동일한 이름으로 쓰인 attribute 때문에 교수의 소속학과와 전공 과목의 학과 등이 구분되어야 한다. 또한 강의목록을 평균 평점을 기준으로 우선적으로 정렬한 뒤, 같은 평균 평점을 보유할 경우 작성된 강의평가의 수를 기준으로 다시 정렬해야 한다. 또한 작성된 강의평가가 아예 없는 경우 해당 강의가 검색 결과에서 제외되는 버그를 예방해야 한다. 이러한 다양한 조건들을 고려하여 다음과 같은 다소 복잡한 Query문을 사용하였다.

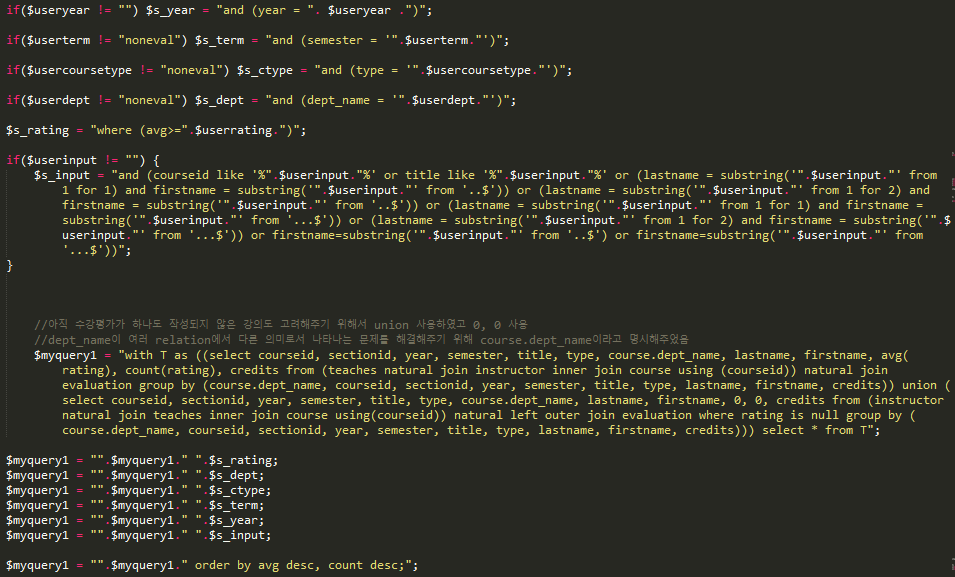


Figure 3-(2)-1 검색 Query

이러한 일련의 조건들을 거쳐서 나타나는 강의들은 클릭이 가능한 component로 만들었다. Javascript를 이용하여 사용자가 클릭을 할 경우 해당 강의에 대한 상세정보와 작성된 강의평가들의 목록이 슬라이드되면서 화면에 나타나도록 하였다.

 렌더링되는 강의목록의 순서는 평점과 작성된 강의평가 수에 따라 정렬이 된 상태이고, 클릭을 했을 때의 과목 상세정보에서는 해당 section에 정의되어 있는 정보들을 표현한다. Figure 3-(2)-3과 Figure 3-(2)-4에서 볼 수 있듯이, 전공과목의 경우 학과와 전공이라는 문자열을 표기하고, 비전공과목의 경우 학과정보를 표시하지 않고 교양이라는 문자열을 보여준다. 그리고 sec\_time\_class에 추가되어 있는 시간과 장소에 대한 multivalue들을 반복문을 통해 모두 보여주고, 강의평가가 작성되지 않아서 평균평점 및 수강평가 수가 0인 강의도 나타나도록 하였다.

Figure 3-(2)-4 비전공과목

Figure 3-(2)-3 전공과목

Figure 3-(2)-2 search\_result.php

 수강평가 버튼을 누르면 해당 강의에 작성된 강의평가 목록을 받은 추천 수를 기준으로 정렬하여 나열해주고, 가장 하단에는 강의평가를 남길 수 있는 란을 제시한다. 목록에 제시된 강의평가 들은 각각 추천하기 링크를 달아두어 사용자가 추천을 할 수 있도록 하였다. Abusing을 막기 위해서 ajax를 통해 페이지를 refresh하기 전까지는 추천이 최대 한 번 까지만 되도록 하였고, php의 session 기능을 통해 자신이 작성한 강의평가에 대해서는 추천을 할 수 없도록 막아두었다. 그리고 또한 php의 session 기능을 활용해 자신이 작성한 강의평가에는 수정, 삭제버튼이 나타나도록 하였다. 수정버튼을 클릭하면, 수정을 할 수 있는 창이 슬라이드되면서 나타나며 사용자가 수정사항을 입력할 수 있도록 하고, 삭제 버튼을 클릭할 경우 해당 강의평가를 database로부터 삭제한다.

Figure 3-(2)-5 강의평가 목록

그리고 사용자가 강의평가 작성을 시도했을 때, 우선적으로 search query를 수행하여 이미 해당 계정이 해당 강의에 대해 작성된 평가가 있는지 검사한다. 만약 이미 해당 강의에 대한 강의평가를 작성한 계정이 접근을 했다면, 작성을 할 수 없도록 하여 unique 조건을 만족시켰다. 또한 사용자가 아무 내용도 입력하지 않았을 시에도 강의평가 작성을 할 수 없게 만들어서 not null 조건을 만족시켰다.

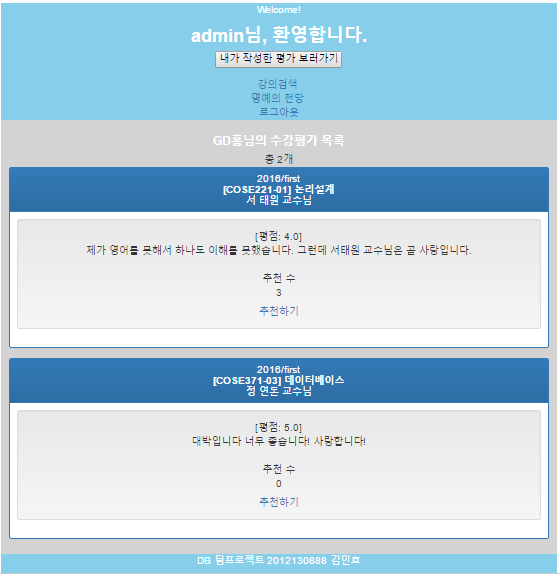
 각 강의평가 작성자의 닉네임은 버튼형태로 설정해두었는데, 이것을 클릭하면 해당 닉네임을 보유한 회원이 작성한 강의평가들을 모아볼 수 있는 eval\_list.php 페이지로 이동한다. 이는 아무 페이지 상단의 ‘내가 작성한 평가 보러가기’라는 버튼을 클릭함으로써 접근할 수도 있는데, 이 경우에는 로그인된 계정이 작성한 평가들을 모아서 보게 된다. 이때 해당 페이지에서도 php의 session 정보를 활용하여 자신이 쓴 강의평가일 경우 수정, 삭제버튼이 나타나도록 하였고, 추천하기 링크의 경우 자신의 강의평가를 제외한 다른 강의평가에만 추천하기가 적용되도록 search\_result.php 페이지와 동일하게 구성되었다.

Figure 3-(2)-6 강의평가 목록

 그리고 페이지 상단의 명예의 전당 링크를 클릭하면 작성한 강의평가들에 대해 가장 누적 추천수가 많은 계정과 가장 많은 강의평가를 작성한 계정들을 정렬하여 순위를 매겨 보여준다. 계정이 많아질 경우 페이지의 길이가 불필요하게 길어지는 것을 막기 위해 10위까지만 보여지도록 설정하였다. 순위표에 나타난 계정의 닉네임은 역시 버튼 형태로 제공하여 클릭할 경우 eval\_list.php로 이동해서 해당 계정이 작성한 강의평가들을 모아서 볼 수 있도록 하였다.

Figure 3-(2)-7 명예의 전당

1. **관리자 페이지**

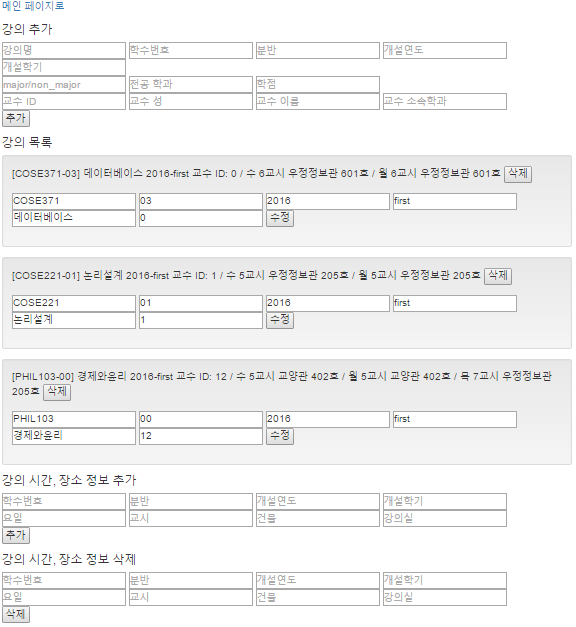
학교 강의나 교수에 대한 정보 등은 기본적으로 database 내에 미리 정의되어 있다는 가정이 있지만, 테스트용 혹은 관리용으로 강의에 대한 정보를 추가하거나 수정 및 삭제할 수 있도록 하는 것이 유용할 것이라고 판단했기 때문에 별도의 관리자 페이지를 만들었다. 해당 페이지는 admin이라는 닉네임을 가진 계정만 접근할 수 있으며, 페이지 내부에서 직접적으로 database 내에 강의 정보를 update할 수 있다. 강의를 추가하거나 수정할 때는 중복 조건을 체크하여 primary key의 unique 조건을 만족시켜주었다. 수정 시 수정을 원치 않아서 그대로 남겨두는 값에 대해서는 update query를 실행하지 않았다. 새로운 강의 정보를 생성할 때 기존에 없던 course나 instructor는 새로 생성되지만, course와 instructor가 삭제가 가능하게 되면 여러 문제가 발생할 수 있기 때문에 강의 정보 삭제 시에는 section만 삭제되도록 제한하였다.

Figure 3-(3)-1 관리자 페이지