

# 데이터베이스 기말프로젝트

## 영어학원 출결 관리 시스템

컴퓨터과학과

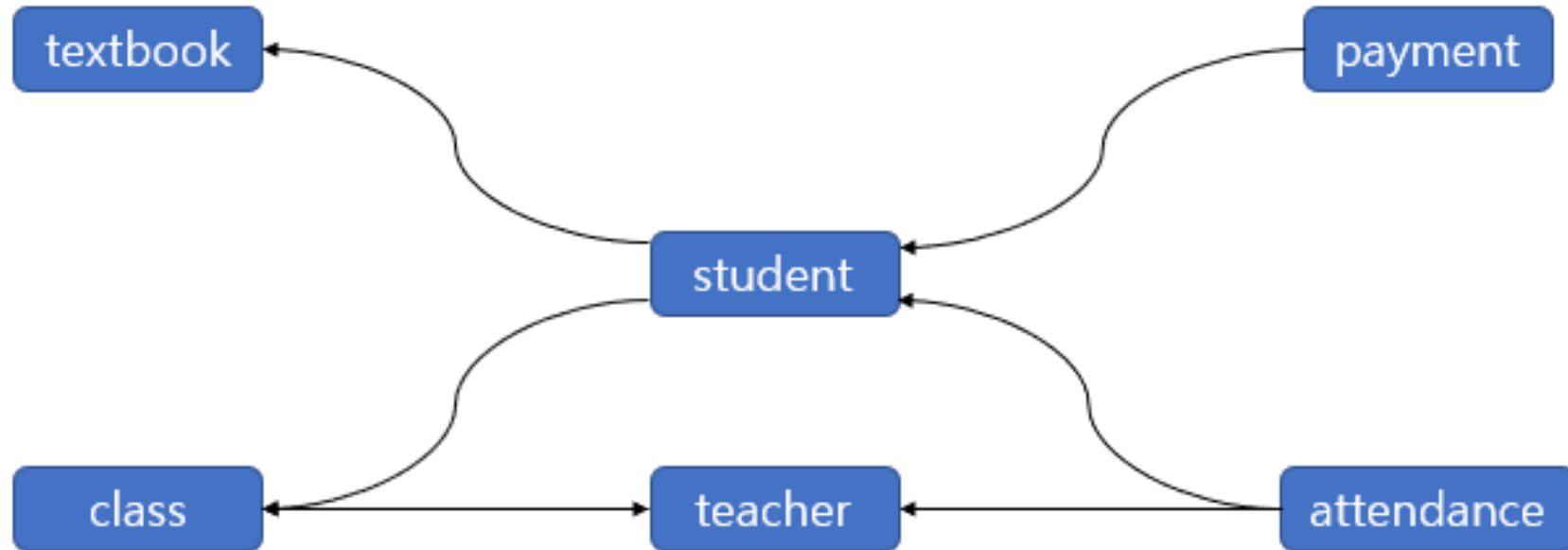
201733005

김신영

# 목차

- 데이터베이스 설계
- 전체 기능
- 구현한 기능
- 기능을 구현하기 위해 사용한 SQL문

# 데이터베이스 설계 - 테이블 구성과 참조 관계



# 학생 테이블 -> 반, 교재

```
mysql> desc student;
```

Field	Type	Null	Key	Default	Extra
student_ID	int	NO	PRI	NULL	
student_name	varchar(10)	NO		NULL	
age	int	NO		NULL	
textbook	varchar(20)	YES	MUL	NULL	
class_ID	varchar(10)	YES	MUL	NULL	

반 테이블

```
mysql> desc class;
```

Field	Type	Null	Key	Default	Extra
class_ID	varchar(10)	NO	PRI	NULL	
roomnumber	int	NO		NULL	
teacher_ID	int	NO	MUL	NULL	

교재 테이블

```
mysql> desc textbook;
```

Field	Type	Null	Key	Default	Extra
bookname	varchar(20)	NO	PRI	NULL	
bookstock	int	YES		NULL	
bookprice	int	NO		NULL	

반, 교재와 학생은 일반적인 일대다 관계이므로, 1측 개체인 반과 교재의 기본키들을 n측 개체인 학생 테이블에 포함시켰다.

# 출석 테이블

-> 학생, 선생님

```
mysql> desc attendance;
```

Field	Type	Null	Key	Default	Extra
a_time	datetime	NO	PRI	CURRENT_TIMESTAMP	DEFAULT_GENERATED
student_ID	int	NO	PRI	NULL	
temperature	decimal(3,1)	YES		NULL	
check_teacher	int	NO	MUL	NULL	

학생의 출석을 승인한 선생님을 확인하기 위해 매 출석마다 승인한 선생님의 id를 참조하여 함께 저장한다.

이로서 출석에 선생님의 책임을 부여하였다.

일대다 관계인 학생과 출석의 관계에서, 출석이 학생 없이 존재할 수는 없으므로 출석은 학생과의 관계에서 의존적이라고 할 수 있다.

따라서 강한 개체인 학생 개체의 기본키를 외래키로서 출석 테이블에 포함하여 기본키를 구성하였다.

```
mysql> desc teacher;
```

Field	Type	Null	Key	Default	Extra
teacher_ID	int	NO	PRI	NULL	
teacher_name	varchar(10)	NO		NULL	
t_class	int	YES		NULL	

```
mysql> desc student;
```

Field	Type	Null	Key	Default	Extra
student_ID	int	NO	PRI	NULL	
student_name	varchar(10)	NO		NULL	
age	int	NO		NULL	
textbook	varchar(20)	YES	MUL	NULL	
class_ID	varchar(10)	YES	MUL	NULL	

# 결제 내역 -> 학생

```
mysql> desc payment;
```

Field	Type	Null	Key	Default	Extra
pay_time	datetime	NO	PRI	CURRENT_TIMESTAMP	DEFAULT_GENERATED
student_ID	int	NO	PRI	NULL	
pay_amount	int	NO		NULL	

결제 테이블은 학생 테이블을 참조하는 방식에서 출석 테이블과 비슷하다.

출석 테이블과 같은 이유로(출석은 학생에 의존적이다) 기본키에 학생 아이디를 참조하는 외래키를 포함하였다.

```
mysql> desc student;
```

Field	Type	Null	Key	Default	Extra
student_ID	int	NO	PRI	NULL	
student_name	varchar(10)	NO		NULL	
age	int	NO		NULL	
textbook	varchar(20)	YES	MUL	NULL	
class_ID	varchar(10)	YES	MUL	NULL	

매 결제마다 결제한 학생의 정보를 기본키로서 포함하여 저장한다.

# 반 테이블 -> 선생님

```
mysql> desc class;
```

Field	Type	Null	Key	Default	Extra
class_ID	varchar(10)	NO	PRI	NULL	
roomnumber	int	NO		NULL	
teacher_ID	int	NO	MUL	NULL	

1대1 관계인 반과 교사 테이블은  
서로 외래키를 주고받도록 했다.

```
mysql> desc teacher;
```

Field	Type	Null	Key	Default	Extra
teacher_ID	int	NO	PRI	NULL	
teacher_name	varchar(10)	NO		NULL	
class_ID	varchar(10)	NO	MUL	NULL	

# 전체 기능

- 교재 재고량, 가격 파악
- 반 별 담당 교사와 학생 정보 파악

## 구현한 기능

- 학생 정보(이름, 나이, 반, 사용교재) 등록, 읽어 오기, 수정, 삭제
- 출석 기록 저장(날짜와 시간, 체온, 책임 선생님 정보 포함)
- 결제 정보 저장



# UI로 구현한 기능 - 학생 등록



2001	김신영	11	None
2002	이준혁	8	None
2003	정소화	11	None
2004	성원용	12	None
2005	이성규	9	Disney Reading 1
2006	이원용	11	ABC eggs 2
2007	김갑생	11	Disney Reading 2

학생 등록 창을 통해서 학생 정보를 다룰 수 있다.

특히 textbook과 class는 각각 교재 테이블과 반 테이블에서 값을 가져와 **학생 테이블이 바로 참조**할 수 있도록 하였다.

# 사용한 SQL문 - 학생등록

각 버튼에 사용된 SQL문들

```
cursor.execute("insert into student values \n\n('"+ student_ID +"', '"+ student_name + "', \n\n '"+ age + "', '"+ textbook + "', '"+ class_ID + "'"))
```

```
cursor.execute("delete from student \n\nwhere student_ID = '"+ e_id.get() + "'")
```

```
cursor.execute("update student set student_name = \n\n '"+ student_name + "', age = '"+ age + "', \n\n textbook = '"+ textbook + "', class_ID = '"+ class_ID + "' \n\nwhere student_ID = '"+ student_ID + "'")
```

```
cursor.execute("select * from student \n\nwhere student_ID = '"+ e_id.get() + "'")
```

리스트박스에 학생 정보를  
보이기 위한 SQL문

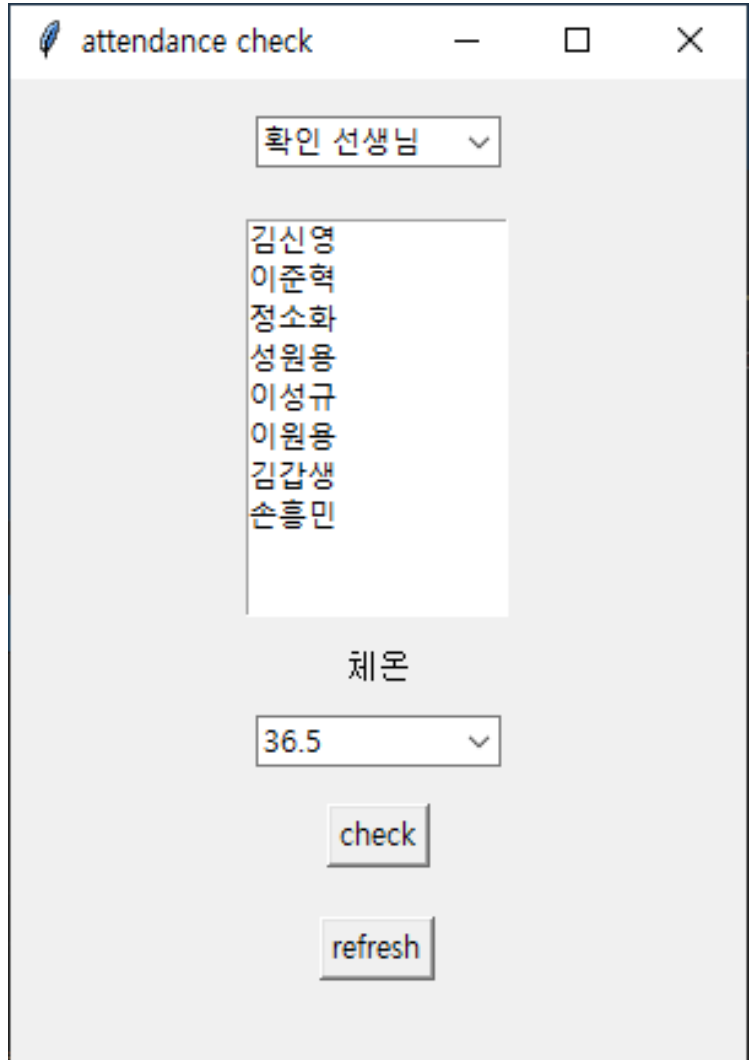
```
cursor.execute("select * from student")
```

교재, 반 선택지(콤보 박스)를 위한 SQL문

```
cursor.execute("select bookname from textbook")\ntextbooknames = [item[0] for item in cursor.fetchall()]
```

```
cursor.execute("select class_ID from class")\nclassnumbers = [item[0] for item in cursor.fetchall()]
```

# UI로 구현한 기능 - 출석 체크



attendance check

확인 선생님 ▾

김신영  
이준혁  
정소화  
성원용  
이성규  
이원용  
김갑생  
손흥민

체온

36.5 ▾

check

refresh

출석 체크 창에서는 현재 학생 테이블에 존재하는 학생들의 출석을 체크할 수 있다.

선생님과 학생의 이름을 선택하고, 체온을 선택한 뒤 체크 버튼을 누르면 **출석 테이블에 저장**되는 방식이다.

# 사용한 SQL문 - 출석체크

학생과 선생님의 **이름** 가져와서  
콤보 박스와 리스트 박스에 넣기

```
cursor.execute("select teacher_name from teacher")  
teachernames = [item[0] for item in cursor.fetchall()]
```

```
cursor.execute("select student_name from student")
```

가져왔던 **이름들을** 활용해서 ID값  
가져오기

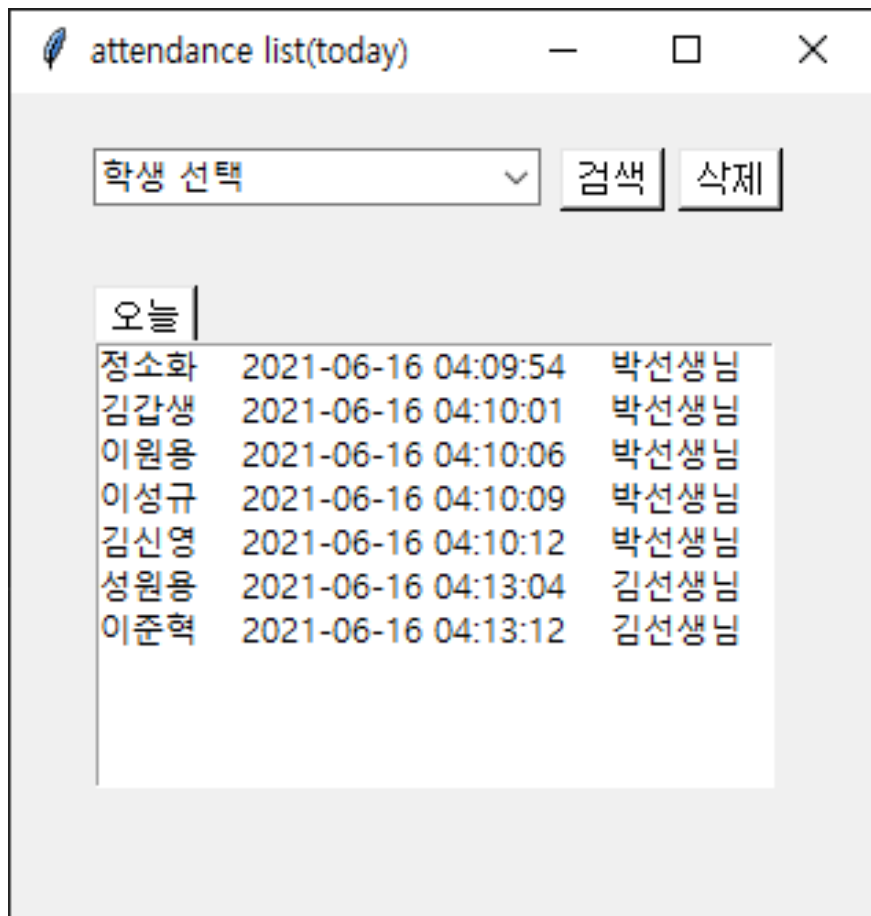
```
cursor.execute("select student_ID from student \\  
| where student_name = '" + s_list.get(ANCHOR) + "'")
```

```
cursor.execute("select teacher_ID from teacher \\  
| where teacher_name = '" + teacher.get() + "'")
```

ID값들과 온도를 출석 테이블에  
저장하기

```
cursor.execute("insert into attendance\  
| (student_ID, temperature, check_teacher)\  
| values('" + s_id2 + "', '" + temperature2 + "', '" + t_id2 + "')")  
cursor.execute("commit")
```

# UI로 구현한 기능 - 출석 기록 확인



오늘		
정소화	2021-06-16 04:09:54	박선생님
김갑생	2021-06-16 04:10:01	박선생님
이원용	2021-06-16 04:10:06	박선생님
이성규	2021-06-16 04:10:09	박선생님
김신영	2021-06-16 04:10:12	박선생님
성원용	2021-06-16 04:13:04	김선생님
이준혁	2021-06-16 04:13:12	김선생님

학생의 이름을 검색하여 이름 별 출석 내역을 확인할 수도 있고,

첫 화면은 오늘 출석 체크한 기록을 보이도록 하여 **혹시 잘못 체크한 경우** 그 출석 기록을 삭제할 수 있도록 하였다.

# 사용한 SQL문 - 출석 기록 확인

```
cursor.execute("select s.student_name, a.a_time, t.teacher_name \n                from student s, attendance a, teacher t \n                where s.student_id = a.student_id \n                  and t.teacher_id = a.check_teacher \n                  and date(a_time) = date(now()) \n                order by a_time asc")
```

삭제를 위해 **오늘의 전체 학생 출석정보**를 가져오는 SQL문이다. date() 함수를 사용하여 **테이블과 현재시간의 날짜만 비교**하여, **오늘의 출석만** 불러오도록 하였다. 빠른 날짜 순으로 나열하였다.

```
cursor.execute("select s.student_name, a.a_time, t.teacher_name \n                from student s, attendance a, teacher t \n                where s.student_id = a.student_id \n                  and t.teacher_id = a.check_teacher \n                  and s.student_name = ' ' + student_name + ' ' \n                order by a_time asc")
```

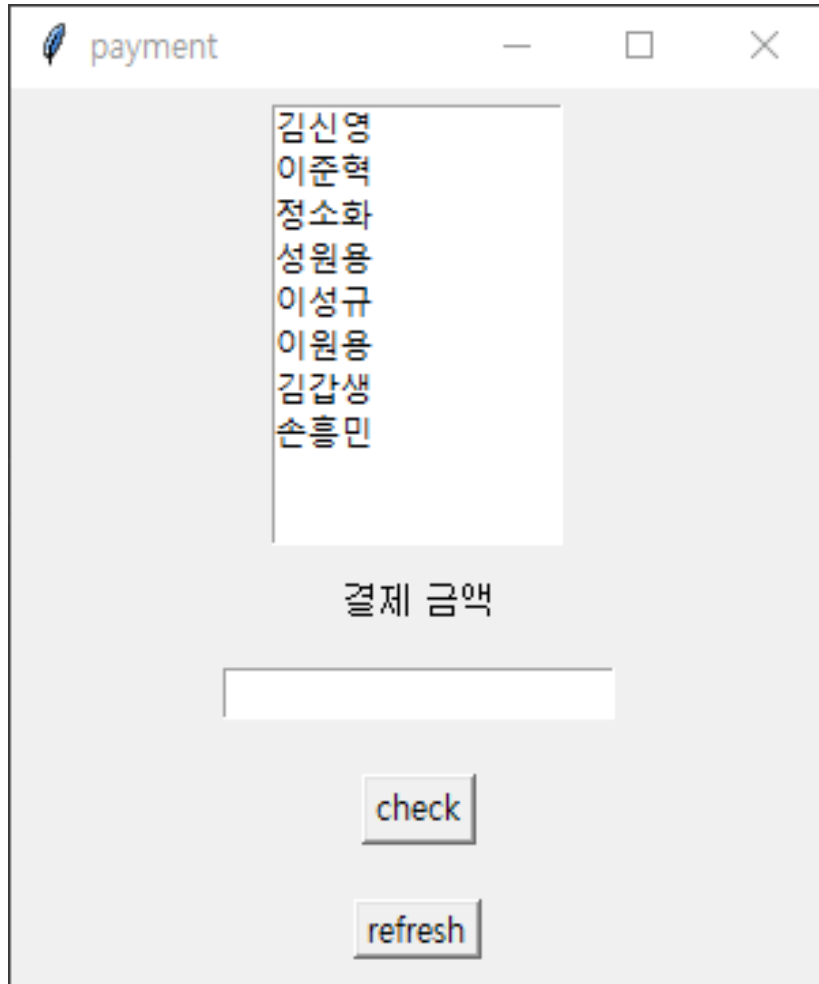
오늘의 전체 출석 정보가 아닌, **특정 학생의 전체 날짜 출석 정보**를 불러오는 SQL문이다. 위와 마찬가지로 학생과 출석 테이블을 조인하였다. 오늘 뿐만 아니라 해당 학생의 이전 출석 기록도 확인할 수 있다.

```
cursor.execute("select student_ID from student \n                where student_name = ' ' + delete_name + ' '")
```

```
cursor.execute("delete from attendance \n                where student_ID = ' ' + delete_id2 + ' '\n                  and date(a_time) = date(now())")
```

리스트 박스에서 가져온 학생 이름으로 ID를 불러온다. 그 후 그 아이디를 활용해 해당 출석 정보를 삭제하는 SQL문이다.

# UI로 구현한 기능 - 결제 기록




The screenshot shows a web application window titled "payment". Inside the window, there is a list of student names: 김신영, 이준혁, 정소화, 성원용, 이성규, 이원용, 김갑생, and 손흥민. Below the list, there is a label "결제 금액" (Payment Amount) and an input field. At the bottom, there are two buttons: "check" and "refresh".

결제 체크 창에서는 현재 학생 테이블에 존재하는 학생들의 결제 체크할 수 있다.

학생의 이름을 선택하고 결제 금액을 입력하여 버튼을 누르면 **결제 테이블에 저장**되는 방식이다.

# UI로 구현한 기능 - 결제 기록 확인

 payment list

학생 선택

검색

삭제

오늘

김갑생	2021-06-16 14:39:53	270000
이성규	2021-06-16 16:56:25	270000
성원용	2021-06-16 16:57:16	270000
성원용	2021-06-16 16:59:08	29000
성원용	2021-06-16 17:16:42	270000
이준혁	2021-06-16 17:16:53	290000

학생의 이름을 검색하여 이름 별 결제 내역을 확인할 수도 있고,

첫 화면은 오늘 결제 체크한 기록을 보이도록 하여 **혹시 잘못 결제한 경우** 그 출석 기록을 삭제할 수 있도록 하였다.



## 추가적으로 구현 가능한 기능

```
mysql> select c.class_id, t.teacher_name, count(s.student_ID)
-> from student s, class c, teacher t
-> where s.class_ID = c.class_ID
-> and c.teacher_ID = t.teacher_ID
-> group by c.class_ID;
```

class_id	teacher_name	count(s.student_ID)
1	김선생님	1
2	박선생님	5
3	윤선생님	2

반 별 담당 선생님과  
반에 속한 학생 수

## 추가적으로 구현 가능한 기능

```
mysql> select b.bookname, count(s.student_name)
-> from textbook b, student s
-> where s.textbook = b.bookname
-> group by b.bookname;
```

bookname	count(s.student_name)
Disney Reading 1	2
Disney Reading 3	2
Grammer practicing 1	1
Disney Reading 2	1
ABC eggs 3	1
ABC eggs 1	1

교재 별 사용 학생 수

# 보완할 점

외래키 제약조건에 **on\_delete\_cascade 제약사항**을 미리 넣지 못해서 학생정보 삭제 시도 시 다른 테이블에 학생의 정보가 존재한다면 외래키 제약조건에 위배되어 삭제가 되지 않음.

제약조건을 삭제 후 다시 설정해 주면 테이블 사이의 관계가 잘 유지될지 확신이 없어 기능시연과 제출에 큰 지장이 생길까 두려워 수정을 하지 못함. (다른 테이블의 튜플을 제거하면 아이디 삭제 가능)

on\_delete\_cascade 제약조건을 포함시키는 것이 구현 의도와 맞다고 생각해서 그 점이 아쉽다.