KOSTA 학생관리

배성윤
심재용

목차

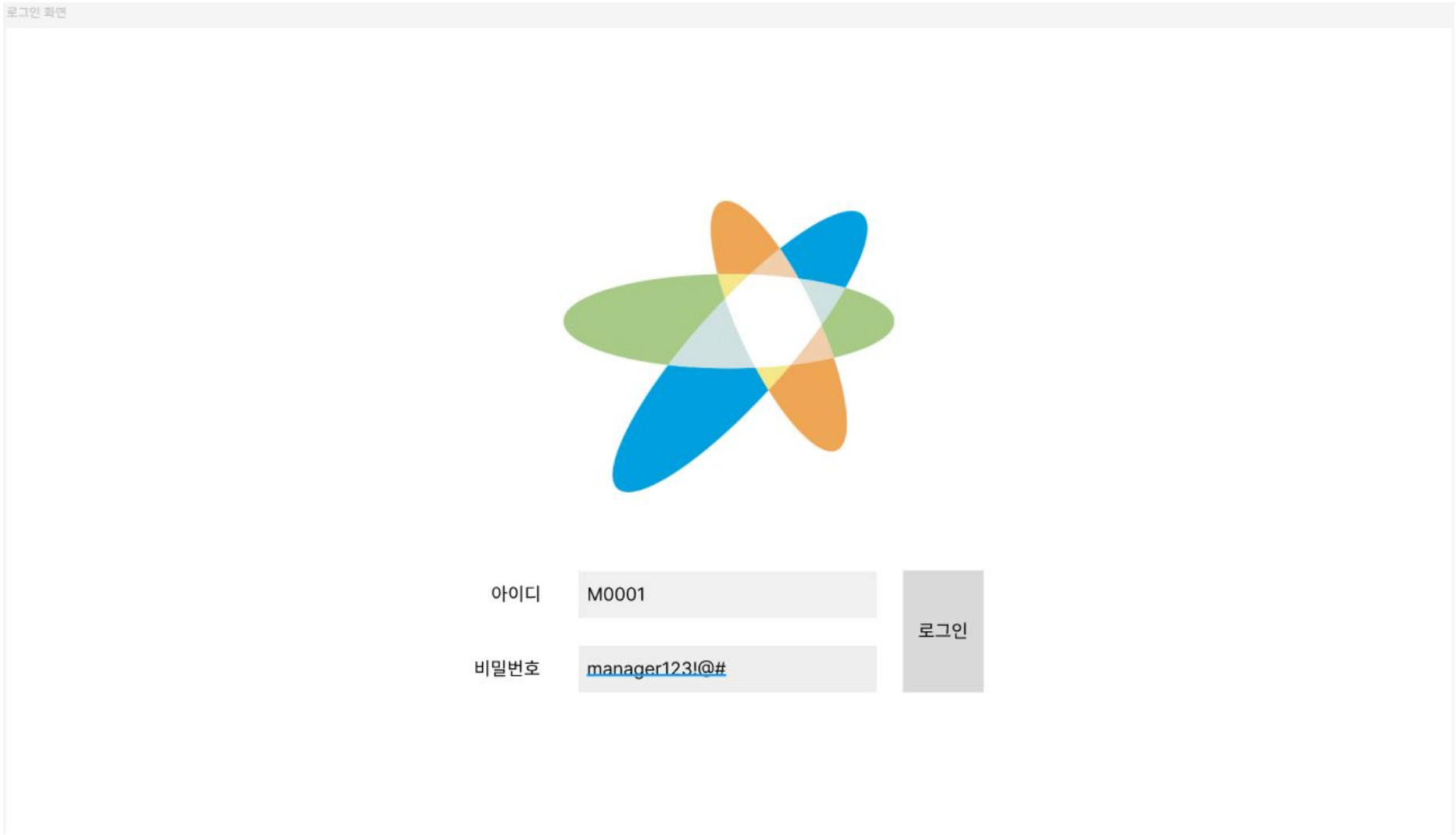
- 개요
- UI 명세서
- 논리 모델링
- 물리 모델링
- 업무 리스트
- DAO 구현
- Class Diagram
- 회고

KOSTA – 개요

- 관리자 로그인
  - 관리자만 접근 가능하도록 로그인 기능 제공
- 학생 관리
  - 기능: 학생 등록, 조회, 수정, 삭제(CRUD)
- 성적 조회
  - 기능: 교육과정 선택 → 학생 목록 + 성적 조회
- 출결 조회
  - 기능: 학생 목록 조회, 출결 정보 조회, 날짜 선택(달력, 페이지 선택)

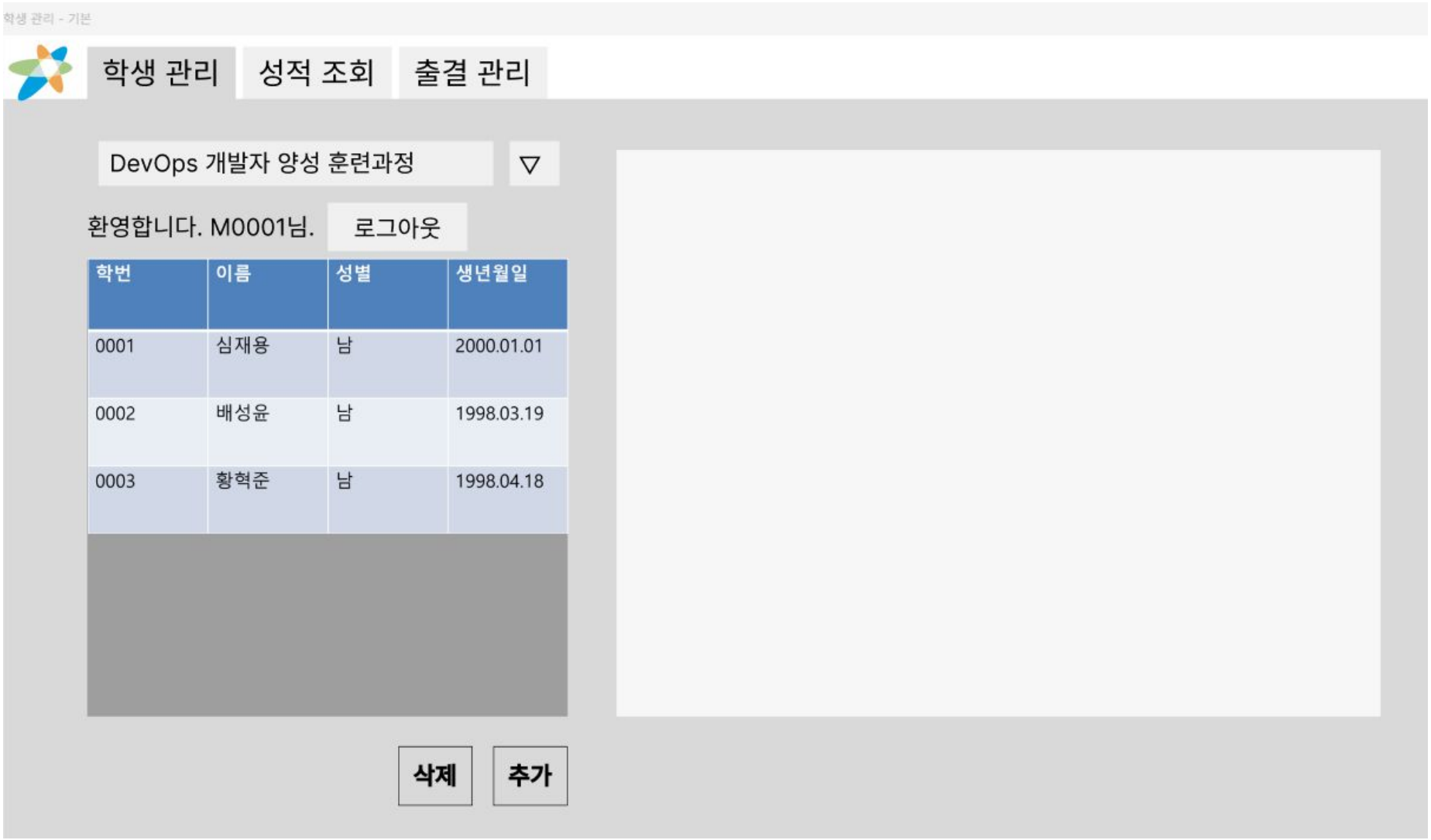
- 일정 관리
  - 데이터: 날짜 +시간 정보, 이벤트(일정) 정보
  - 기능: 달력표 출력, 이벤트(일정) 조회, 이벤트(일정) CRUD교육 과정 관리

UI명세서-로그인

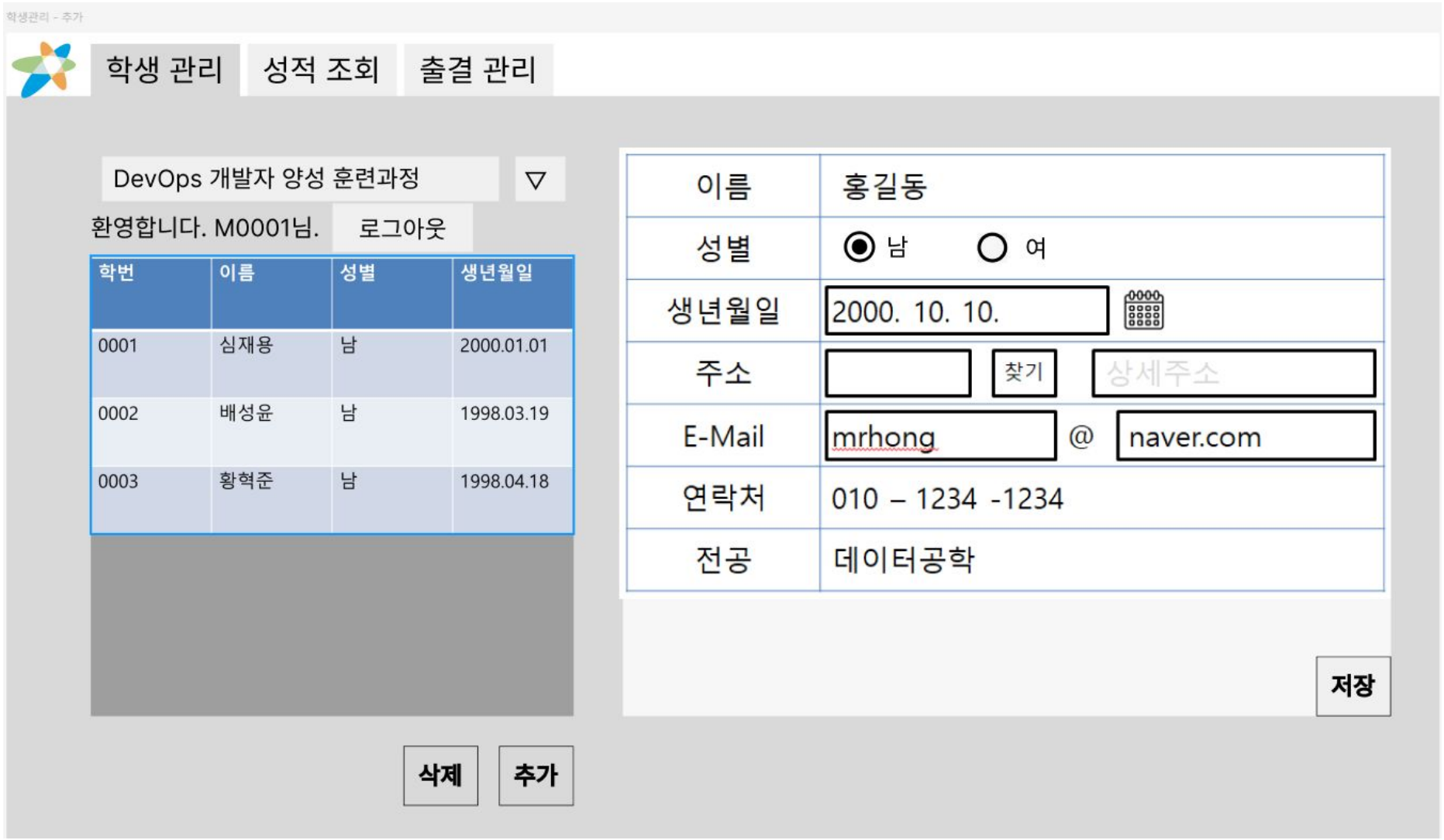


로그인-실패

학생 관리 – 기본


학생 관리 - 기본
학생 관리
성적 조회
출결 관리
DevOps 개발자 양성 훈련과정
환영합니다. M0001님.
로그아웃
학번
이름
성별
생년월일
0001
심재용
남
2000.01.01
0002
배성윤
남
1998.03.19
0003
황혁준
남
1998.04.18
삭제
추가

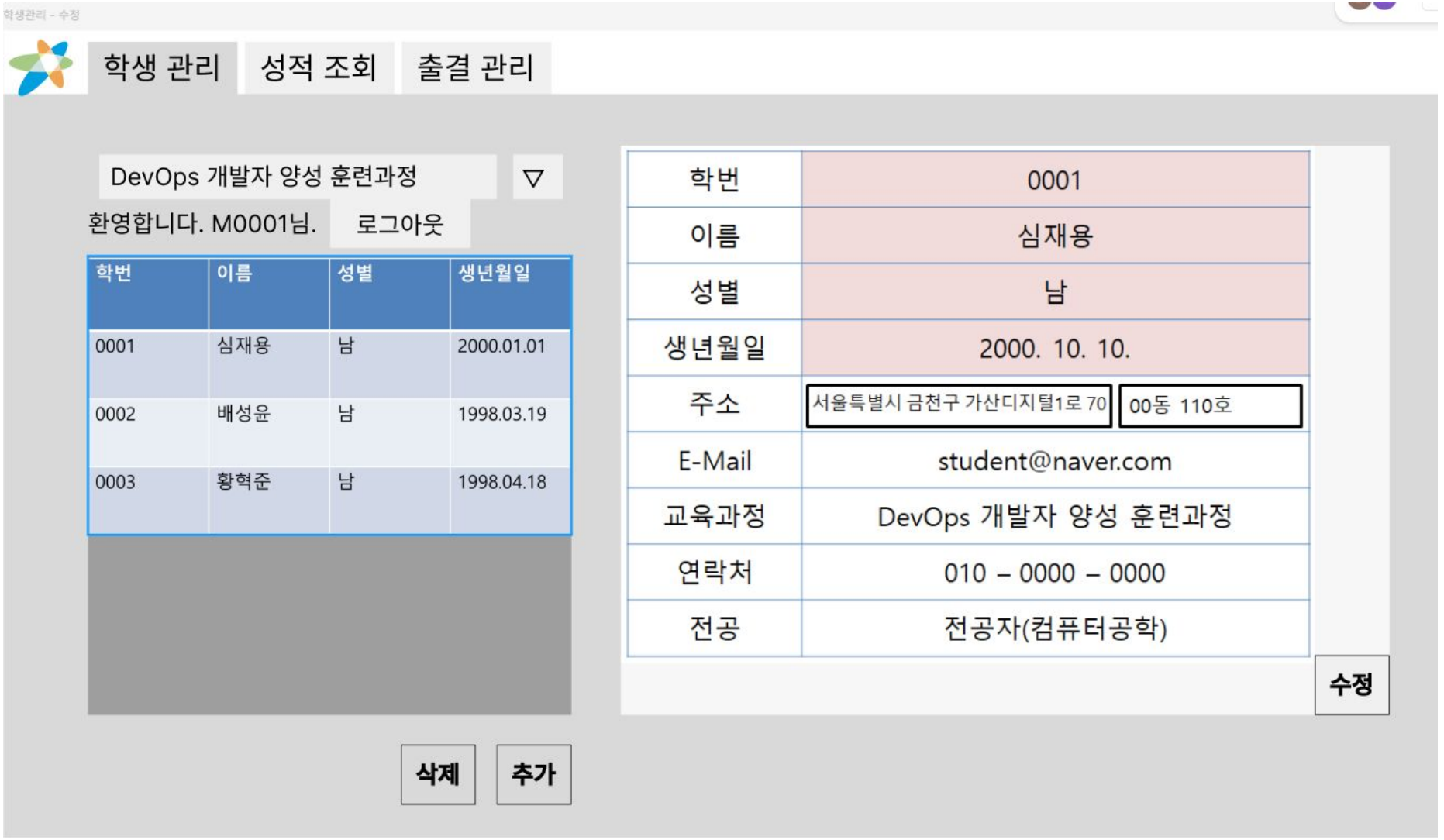
학생관리–추가


학생관리 - 추가
학생 관리
성적 조회
출결 관리
DevOps 개발자 양성 훈련과정
환영합니다. M0001님.
로그아웃
학번
이름
성별
생년월일
0001
심재용
남
2000.01.01
0002
배성윤
남
1998.03.19
0003
황혁준
남
1998.04.18
삭제
추가
이름
홍길동
성별
남
여
생년월일
2000. 10. 10.
주소
찾기
상세주소
E-Mail
mrhong
@
naver.com
연락처
010 – 1234 -1234
전공
데이터공학
저장

학생관리-조회


학생 관리-조회
학생 관리
성적 조회
출결 관리
DevOps 개발자 양성 훈련과정
환영합니다. M0001님.
로그아웃
학번 이름 성별 생년월일
0001 심재용 남 2000.01.01
0002 배성윤 남 1998.03.19
0003 황혁준 남 1998.04.18
학번 0001
이름 심재용
성별 남
생년월일 2000. 10. 10.
주소 서울특별시 금천구 가산디지털1로 70
E-Mail student@kosta.or.kr
교육과정 DevOps 개발자 양성 훈련과정
연락처 010 – 0000 – 0000
전공 전공자(컴퓨터공학)
수정
삭제
추가

학생관리–수정


학생관리 - 수정
학생 관리
성적 조회
출결 관리
DevOps 개발자 양성 훈련과정
환영합니다. M0001님.
로그아웃
학번 이름 성별 생년월일
0001 심재용 남 2000.01.01
0002 배성윤 남 1998.03.19
0003 황혁준 남 1998.04.18
삭제
추가
학번 0001
이름 심재용
성별 남
생년월일 2000. 10. 10.
주소 서울특별시 금천구 가산디지털1로 70 00동 110호
E-Mail student@naver.com
교육과정 DevOps 개발자 양성 훈련과정
연락처 010 – 0000 – 0000
전공 전공자(컴퓨터공학)
수정

학생 관리 – 삭제


학생 관리-삭제
학생 관리
성적 조회
출결 관리
DevOps 개발자 양성 훈련과정
환영합니다. M0001님.
로그아웃
학번
이름
성별
생년월일
0001
심재용
남
0002
배성윤
남
0003
황혁준
남
※ Warning ※
정말로 삭제하시겠습니까?
취소
확인
학번
0001
이름
심재용
남
2000. 10. 10.
student@kosta.or.kr
010 – 0000 – 0000
전공
전공자(컴퓨터공학)
수정
삭제
추가

성적 조회

학번	이름	시험1	프로젝트1
0001	심재용	23	76
0002	배성윤	54	45
0003	황혁준	54	43
0004	표수정	32	76
0005	지연우	56	46
0006	성민서	76	42
0007	박세희	79	54

성적 조회–시험 항목 추가


학생 관리
성적 조회
출결 관리
DevOps 개발자 양성 훈련과정
+ 시험 항목 추가
학번
이름
시험1
프로젝트1
0001
심재용
23
76
0002
0003
0004
0005
0006
성민서
76
42
0007
박세희
79
54
과목명을 입력하세요
: 프로젝트 2

성적 조회–시험 항목 추가 완료

학번	이름	시험1	프로젝트1	프로젝트2
0001	심재용	23	76	
0002	배성윤	54	45	
0003	황혁준	54	43	
0004	표수정	32	76	
0005	지연우	56	46	
0006	성민서	76	42	
0007	박세희	79	54	

출결조회

학번	이름	출입	외출	복귀	퇴실	비고
0001	심재용	08:50			17:53	출석
0002	배성윤	08:52			17:53	출석
0003	황혁준	08:53	11:32	13:22	17:53	외출
0004	표수정	08:54			15:53	조퇴
0005	지연우					출석(예비군)
0006	성민서					결석
0007	박세희	09:42			17:53	지각

출결조회-선택조회

학번	이름	출입	외출	복귀	퇴실	비고
0006	성민서					결석

관리자 테이블

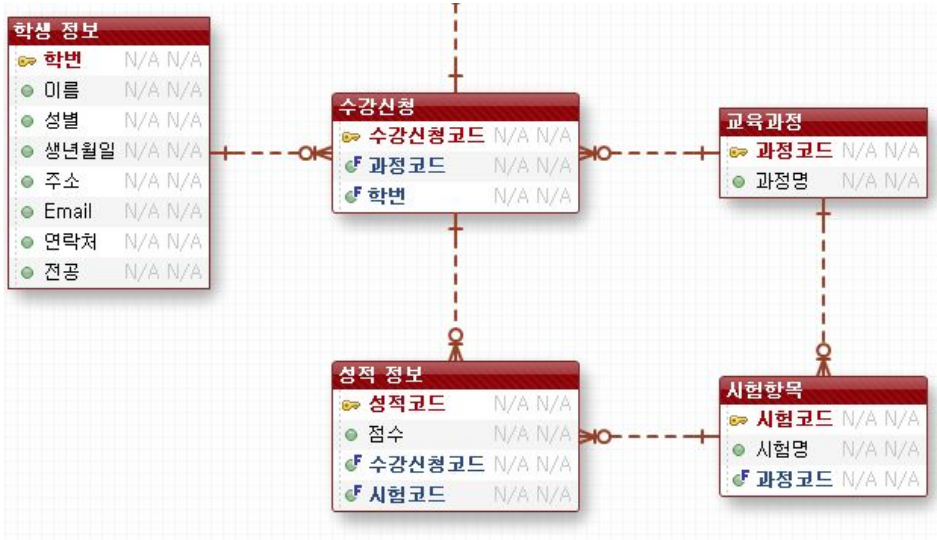
학생정보 테이블


학생 관리
성적 조회
출결 관리
DevOps 개발자 양성 훈련과정
환영합니다. M0001님.
로그아웃
학번 이름 성별 생년월일
0001 심재용 남 2000.01.01
0002 배성윤 남 1998.03.19
0003 황혁준 남 1998.04.18
학번 0001
이름 심재용
성별 남
생년월일 2000. 10. 10.
주소 서울특별시 금천구 가산디지털1로 70
E-Mail student@kosta.or.kr
교육과정 DevOps 개발자 양성 훈련과정
연락처 010 – 0000 – 0000
전공 전공자(컴퓨터공학)
수정
삭제
추가


학생 정보
학번 N/A N/A
이름 N/A N/A
성별 N/A N/A
생년월일 N/A N/A
주소 N/A N/A
Email N/A N/A
연락처 N/A N/A
전공 N/A N/A
수강신청
수강신청코드 N/A N/A
과정코드 N/A N/A
학번 N/A N/A
교육과정
과정코드 N/A N/A
과정명 N/A N/A

성적 테이블

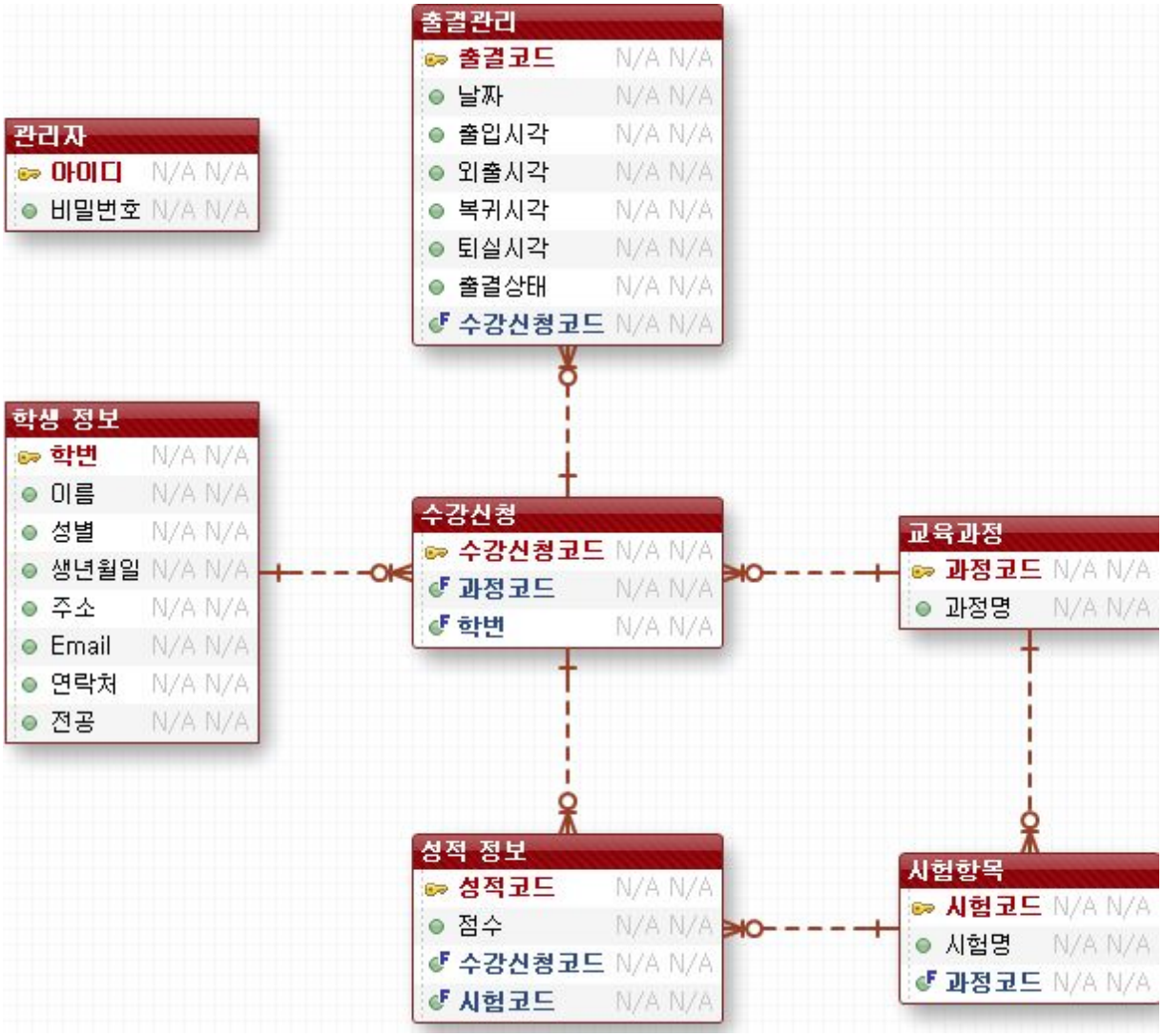
학번	이름	시험1	프로젝트1
0001	심재용	23	76
0002	배성윤	54	45
0003	황혁준	54	43
0004	표수정	32	76
0005	지연우	56	46
0006	성민서	76	42
0007	박세희	79	54



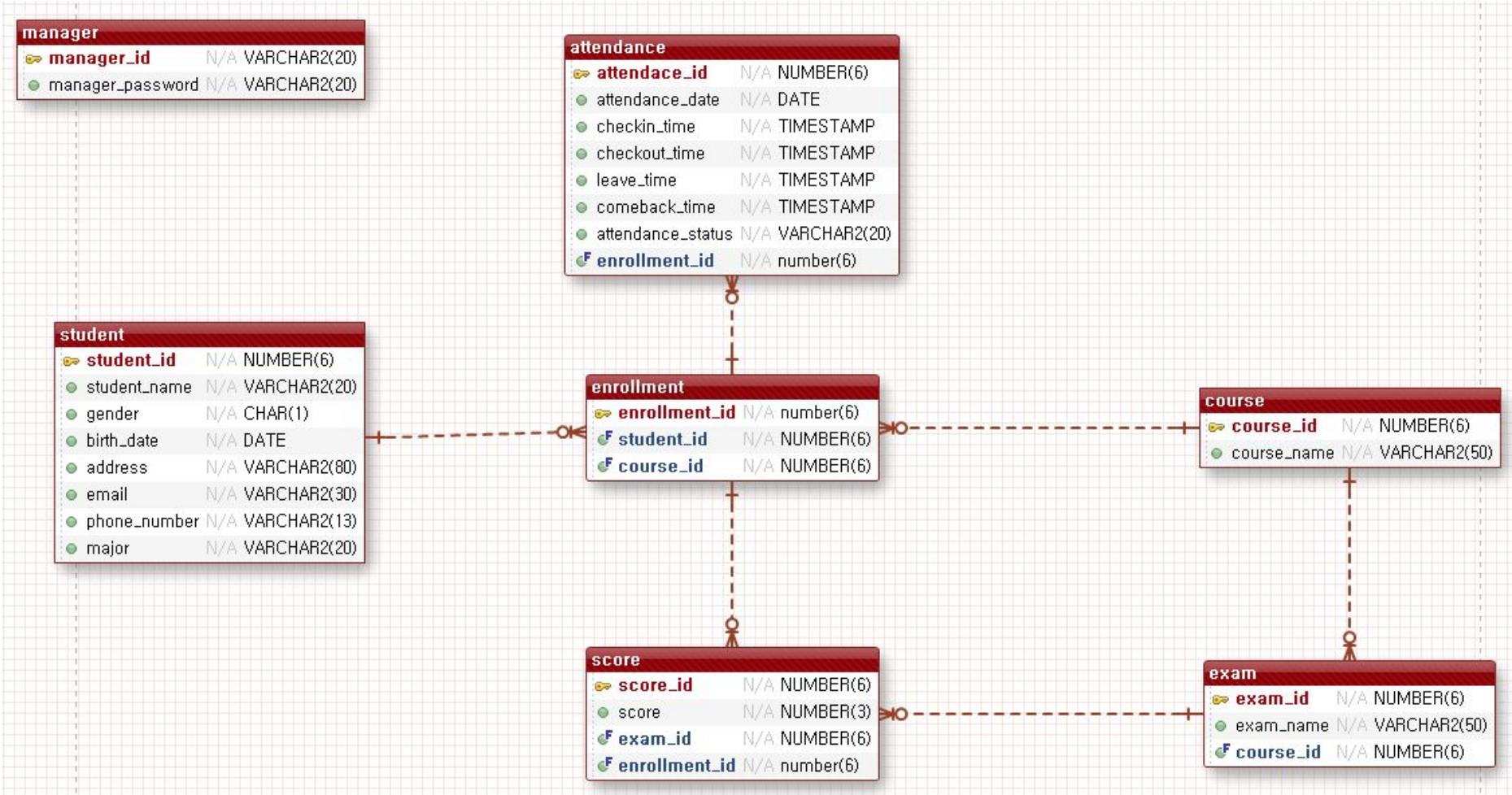
출결 테이블

학번	이름	출입	외출	복귀	퇴실	비고
0001	심재용	08:50			17:53	출석
0002	배성윤	08:52			17:53	출석
0003	황혁준	08:53	11:32	13:22	17:53	외출
0004	표수정	08:54			15:53	조퇴
0005	지연우					출석(예비군)
0006	성민서					결석
0007	박세희	09:42			17:53	지각

최종 테이블


출결관리
출결코드 N/A N/A
날짜 N/A N/A
출입시각 N/A N/A
외출시각 N/A N/A
복귀시각 N/A N/A
퇴실시각 N/A N/A
출결상태 N/A N/A
수강신청코드 N/A N/A
관리자
아이디 N/A N/A
비밀번호 N/A N/A
학생 정보
학번 N/A N/A
이름 N/A N/A
성별 N/A N/A
생년월일 N/A N/A
주소 N/A N/A
Email N/A N/A
연락처 N/A N/A
전공 N/A N/A
수강신청
수강신청코드 N/A N/A
과정코드 N/A N/A
학번 N/A N/A
교육과정
과정코드 N/A N/A
과정명 N/A N/A
성적 정보
성적코드 N/A N/A
점수 N/A N/A
수강신청코드 N/A N/A
시험코드 N/A N/A
시험항목
시험코드 N/A N/A
시험명 N/A N/A
과정코드 N/A N/A

물리 모델링


manager
manager_id N/A VARCHAR2(20)
manager_password N/A VARCHAR2(20)
attendance
attendace_id N/A NUMBER(6)
attendance_date N/A DATE
checkin_time N/A TIMESTAMP
checkout_time N/A TIMESTAMP
leave_time N/A TIMESTAMP
comeback_time N/A TIMESTAMP
attendance_status N/A VARCHAR2(20)
enrollment_id N/A number(6)
student
student_id N/A NUMBER(6)
student_name N/A VARCHAR2(20)
gender N/A CHAR(1)
birth_date N/A DATE
address N/A VARCHAR2(80)
email N/A VARCHAR2(30)
phone_number N/A VARCHAR2(13)
major N/A VARCHAR2(20)
enrollment
enrollment_id N/A number(6)
student_id N/A NUMBER(6)
course_id N/A NUMBER(6)
course
course_id N/A NUMBER(6)
course_name N/A VARCHAR2(50)
score
score_id N/A NUMBER(6)
score N/A NUMBER(3)
exam_id N/A NUMBER(6)
enrollment_id N/A number(6)
exam
exam_id N/A NUMBER(6)
exam_name N/A VARCHAR2(50)
course_id N/A NUMBER(6)

입력 데이터1

	STUDENT_ID	STUDENT_NAME	GENDER	BIRTH_DATE	ADDRESS	EMAIL	PHONE_NUMBER	MAJOR
1	1	심재용	남	98/08/10	서울특별시 금천구 독산동	wodyd0810@naver.com	010-1111-1111	물리학과
2	2	배성윤	남	00/10/01	경기도과천시 갈현동	tjddbs@naver.com	010-2222-2222	보안학과
3	3	표수정	여	95/03/16	서울특별시 서초구 내곡동	tnwjd@daum.net	010-3333-3333	재활학과
4	4	지연우	여	01/11/30	서울특별시 금천구 가산동	dusdn@gmail.com	010-4444-4444	정보통신학과
5	5	황혁준	남	98/04/18	서울특별시 서초구 양재동	gurwns@gmail.com	010-5555-5555	경영학과
6	6	성민서	여	00/05/23	서울특별시 용산구 한남동	alstj@naver.com	010-6666-6666	수학과
7	7	박세희	여	96/09/07	포항시 북구 흥해읍	tpgml@naver.com	010-7777-7777	기계공학과
8	8	최정문	여	97/10/28	부산광역시 남구 대연동	wjdans@naver.com	010-8888-8888	디자인학과
9	9	고은별	여	93/09/01	부산광역시 수영구 광안동	silverstar@daum.net	010-9999-9999	우주항공학과
10	10	김채운	남	99/03/27	부산광역시 진구 전포동	codns@naver.com	010-1234-5678	체육학과

	COURSE_ID	COURSE_NAME
1	1	풀스택 과정
2	2	DevOps 과정
3	3	Java 과정

	ENROLLMENT_ID	STUDENT_ID	COURSE_ID
1	1	1	1
2	2	2	1
3	3	3	1
4	4	4	1
5	5	5	2
6	6	6	2
7	7	7	2
8	8	8	2
9	9	9	3
10	10	10	3
11	11	1	2
12	12	2	2
13	13	3	3
14	14	1	3

	MANAGER_ID	MANAGER_PASSWORD
1	M001	qwer123!
2	M002	asdf456!
3	M003	zxcv789!

입력 데이터2

	EXAM_ID	EXAM_NAME	COURSE_ID
1	1	웹 프로그래밍 기초	1
2	2	데이터베이스 모델링	1
3	3	자바 객체지향 프로그래밍	2
4	4	알고리즘 자료구조	2
5	5	풀스택 개발 프로젝트	3

	SCORE_ID	SCORE	EXAM_ID	ENROLLMENT_ID
1	1	60	1	1
2	2	89	1	2
3	3	83	1	3
4	4	79	1	4
5	5	93	2	1
6	6	84	2	2
7	7	75	2	3
8	8	85	2	4
9	9	76	3	5
10	10	96	3	6
11	11	92	3	7
12	12	89	3	8
13	13	96	4	5
14	14	76	4	6

	ATTENDANCE_ID	ATTENDANCE_DATE	CHECKIN_TIME	CHECKOUT_TIME	LEAVE_TIME	COMEBACK_TIME	ATTENDANCE_STATUS	ENROLLMENT_ID
1	1	25/09/03	25/09/03 08:46:00.000000000	25/09/03 17:59:00.000000000	(null)	(null)	출석	1
2	2	25/09/03	(null)	(null)	(null)	(null)	출석(예비군)	2
3	3	25/09/03	25/09/03 08:23:00.000000000	25/09/03 14:59:00.000000000	(null)	(null)	조퇴	3
4	4	25/09/03	25/09/03 08:24:00.000000000	25/09/03 17:50:00.000000000	(null)	(null)	출석	4
5	5	25/09/03	25/09/03 09:26:00.000000000	25/09/03 17:54:00.000000000	(null)	(null)	지각	5
6	6	25/09/03	25/09/03 08:16:00.000000000	25/09/03 17:15:00.000000000	(null)	(null)	출석	6
7	7	25/09/03	25/09/03 08:39:00.000000000	25/09/03 17:57:00.000000000	25/09/03 10:00:00.000000000	25/09/03 12:23:00.000000000	외출	7
8	8	25/09/03	25/09/03 08:23:00.000000000	25/09/03 15:50:00.000000000	(null)	(null)	조퇴	8
9	9	25/09/03	25/09/03 08:19:00.000000000	25/09/03 17:55:00.000000000	(null)	(null)	출석	9
10	10	25/09/03	25/09/03 08:20:00.000000000	25/09/03 17:53:00.000000000	25/09/03 11:00:00.000000000	25/09/03 14:00:00.000000000	외출	10
11	11	25/09/03	25/09/03 08:05:00.000000000	25/09/03 17:51:00.000000000	(null)	(null)	출석	11
12	12	25/09/03	25/09/03 08:47:00.000000000	25/09/03 17:50:00.000000000	(null)	(null)	출석	12
13	13	25/09/03	25/09/03 08:23:00.000000000	25/09/03 17:50:00.000000000	(null)	(null)	출석	13

업무 리스트

학생관리 (교육과정을 수강 중인 학생의 정보를 관리)	학생정보 조회
	학생정보 추가(이름, 성별, 생년월일, 주소, email, 연락처, 전공)
	학생정보 수정(주소, email, 연락처, 전공)
	학생정보 삭제
출결조회 (교육과정을 수강 중인 학생의 출결정보를 조회)	날짜 선택 기능
	날짜에 따른 출결 정보 출력
	출결상태(출석, 결석, 조퇴 등)에 따른 필터링 기능
성적조회 (교육과정을 수강 중인 학생의 성적 정보를 조회)	성적 조회 기능
	정렬 기능

SQL – 학생관리

학생관리

```
-- 학생 정보 조회
select s.student_id, s.student_name, s.gender, s.birth_date, s.address, s.email, s.phone_number,
s.major
from student s
join enrollment e on s.student_id = e.student_id
join course c on c.course_id = e.course_id
where c.course_id = '1';

-- 학생 정보 추가
insert into student(student_id, student_name, gender, birth_date, address, email, phone_number,
major) values(student_seq.nextval, '김채운', '남', to_date('1999-03-27'), '부산광역시 진구
전포동', 'codns@naver.com', '010-1234-5678', '체육학과');

-- 학생 정보 수정
update student set address = '서울시 강남구 개포동', email = 'test@kosta.or.kr', phone_number =
'010-1234-1234', major = '묵찌빠' where student_id = '1';

-- 학생 정보 삭제
update student set student_name = '알수없음', gender = '무', birth_date = to_date('1900-01-01'),
address = '알수없음', email='알수없음', phone_number = '알수없음', major='알수없음' where
student_id = 1;
```

SQL – 출결조회

```sql
--날짜별 출석 조회
select student.student_id, student.student_name, a.attendance_date, a.checkin_time,
a.leave_time, a.comeback_time, a.checkout_time, a.attendance_status
from attendance a
join enrollment on a.enrollment_id = enrollment.enrollment_id
join student on student.student_id = enrollment.student_id
where a.attendance_date = '2025-09-03' and c.course_id = 1
order by student.student_id asc;

--출결 상태 필터링
select student.student_id, student.student_name, a.attendance_date, a.checkin_time,
a.leave_time, a.comeback_time, a.checkout_time, a.attendance_status
from attendance a
join enrollment on a.enrollment_id = enrollment.enrollment_id
join student on student.student_id = enrollment.student_id
where a.attendance_date = '2025-09-03' and a.attendance_status = '지각'
order by student.student_id asc;
```

SQL – 성적조회

```sql
--성적 조회
select student.student_id, student.student_name, exam.exam_name, score.score
from score
join exam on exam.exam_id = score.exam_id
join enrollment on score.enrollment_id = enrollment.enrollment_id
join student on enrollment.student_id = student.student_id
join course on course.course_id = enrollment.course_id
where course.course_id= 1;

--성적 정렬
select student.student_id, student.student_name, exam.exam_name, score.score
from score
join exam on exam.exam_id = score.exam_id
join enrollment on score.enrollment_id = enrollment.enrollment_id
join student on enrollment.student_id = student.student_id
join course on course.course_id = enrollment.course_id
where course.course_id= 1
order by score.score desc;
```

인터페이스

```
public interface StudentManageDAO {
    public Collection<StudentVO> getStudents(String courseName);
    public boolean addStudent(StudentVO studentVO);
    public boolean updateStudent(int studentId, StudentVO studentVO);
    public boolean deleteStudent(int studentId);

    public Collection<AttendanceVO> getAttendances(int courseId, String date);
    public Collection<AttendanceVO> getAttendances(int courseId, String date, String attendanceStatus);

    public Collection<ScoreVO> getScores(int courseId);
}
```

구현1

```java
/** 학생 정보 조회 */
@Override
public Collection<StudentVO> getStudents(String courseName) {
    List<StudentVO> list = new ArrayList<>();

    String sql =
            "select s.student_id, s.student_name, s.gender, s.birth_date, s.address, s.email, s.phone_number, s.major "
            + "from student s "
            + "join enrollment e on s.student_id = e.student_id "
            + "join course c on c.course_id = e.course_id "
            + "where c.course_name = ?";

    try {
        PreparedStatement preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setString(1, "풀스택 과정");
        ResultSet resultSet = preparedStatement.executeQuery();
        while(resultSet.next()){
            list.add(
                    new StudentVO(
                            resultSet.getInt(1),
                            resultSet.getString(2),
                            resultSet.getString(3),
                            resultSet.getString(4),
                            resultSet.getString(5),
                            resultSet.getString(6),
                            resultSet.getString(7),
                            resultSet.getString(8)
                    )
                );
        }

        resultSet.close();
        preparedStatement.close();

    } catch (SQLException e) {
        e.printStackTrace();
    }
    return list;
}
```

구현2

```java
/** 학생 정보 추가 */
@Override
public boolean addStudent(StudentVO studentVO) {
    boolean result = false;
    String sql = "insert into student "
            + "(student_id, student_name, gender, birth_date, address, email, phone_number, major) "
            + "values "
            + "(student_seq.nextval, ?, ?, to_date(?), ?, ?, ?, ?)";
    try {
        PreparedStatement preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setString(1, studentVO.getStudentName());
        preparedStatement.setString(2, studentVO.getGender());
        preparedStatement.setString(3, studentVO.getBirthDate());
        preparedStatement.setString(4, studentVO.getAddress());
        preparedStatement.setString(5, studentVO.getEmail());
        preparedStatement.setString(6, studentVO.getPhoneNumber());
        preparedStatement.setString(7, studentVO.getMajor());

        result = preparedStatement.executeUpdate() == 1;

        preparedStatement.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return result;
}
```

구현3

```java
/** 학생 정보 수정 */
@Override
public boolean updateStudent(int studentId, StudentVO studentVO) {
    boolean result = false;
    String sql = "update student "
            + "set address = ?, email = ?, phone_number = ?, major = ?"
            + "where student_id = ?";
    try {
        PreparedStatement preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setString(1, studentVO.getAddress());
        preparedStatement.setString(2, studentVO.getEmail());
        preparedStatement.setString(3, studentVO.getPhoneNumber());
        preparedStatement.setString(4, studentVO.getMajor());
        preparedStatement.setInt(5, studentId);

        result = preparedStatement.executeUpdate() == 1;

        preparedStatement.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return result;
}
```

구현4

```java
/** 학생 정보 수정 */
@Override
public boolean deleteStudent(int studentId) {
    boolean result = false;
    String sql = "update student set "
            + "student_name = '알수없음', gender = '무', birth_date = to_date('1980-01-01'), address = '알수없음', email='알수없음', phone_number = '알수없음', major='알수없음' "
            + "where student_id = ?";
    try {
        PreparedStatement preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setInt(1, studentId);

        result = preparedStatement.executeUpdate() == 1;

        preparedStatement.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return result;
}
```

구현5

```java
/** 출석 정보 조회 */
@Override
public Collection<AttendanceVO> getAttendances(int courseId, String date) {
    return getAttendances(courseId, date, "");
}

@Override
public Collection<AttendanceVO> getAttendances(int courseId, String date, String attendanceStatus) {
    List<AttendanceVO> list = new ArrayList<>();

    String sql =
            "select s.student_id, s.student_name, a.attendance_date, a.checkin_time, a.leave_time, a.comeback_time, a.checkout_time, a.attendance_status "
            + "from attendance a "
            + "join enrollment e on a.enrollment_id = e.enrollment_id "
            + "join student s on s.student_id = e.student_id "
            + "where e.course_id = ? and a.attendance_date = ? and a.attendance_status like ?"
            + "order by s.student_id asc";

    try {
        PreparedStatement preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setInt(1, courseId);
        preparedStatement.setString(2, date);
        preparedStatement.setString(3, "%"+attendanceStatus+"%");

        ResultSet resultSet = preparedStatement.executeQuery();

        while(resultSet.next()){
            list.add(
                    new AttendanceVO(
                            resultSet.getInt(1),
                            resultSet.getString(2),
                            resultSet.getString(3),
                            resultSet.getString(4),
                            resultSet.getString(5),
                            resultSet.getString(6),
                            resultSet.getString(7),
                            resultSet.getString(8)
                    )
                );
        }

        resultSet.close();
        preparedStatement.close();

    } catch (SQLException e) {
        e.printStackTrace();
    }
    return list;
}
```

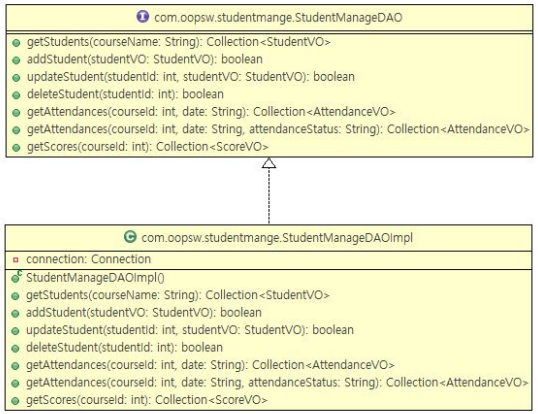
구현6

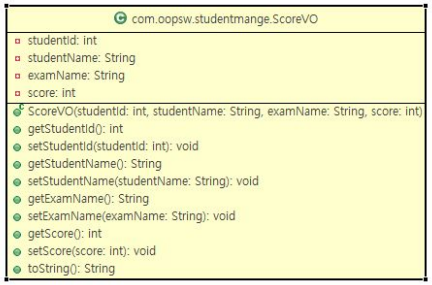
```java
/** 성적 정보 조회 */
@Override
public Collection<ScoreVO> getScores(int courseId) {
    List<ScoreVO> list = new ArrayList<>();

    String sql = "select st.student_id, st.student_name, e.exam_name, s.score "
            + "from score s "
            + "join exam e on e.exam_id = s.exam_id "
            + "join enrollment en on s.enrollment_id = en.enrollment_id "
            + "join student st on en.student_id = st.student_id "
            + "join course c on c.course_id = en.course_id "
            + "where c.course_id= ?";
    try {
        PreparedStatement preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setInt(1, courseId);

        ResultSet resultSet = preparedStatement.executeQuery();

        while(resultSet.next()){
            list.add(
                    new ScoreVO(
                            resultSet.getInt(1),
                            resultSet.getString(2),
                            resultSet.getString(3),
                            resultSet.getInt(4)
                    )
                );
        }

        resultSet.close();
        preparedStatement.close();

    } catch (SQLException e) {
        e.printStackTrace();
    }

    return list;
}
```

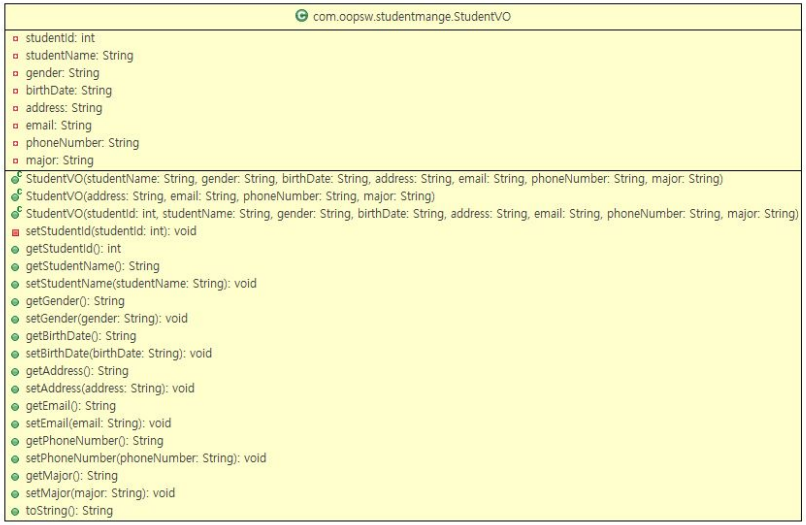
단위 테스트

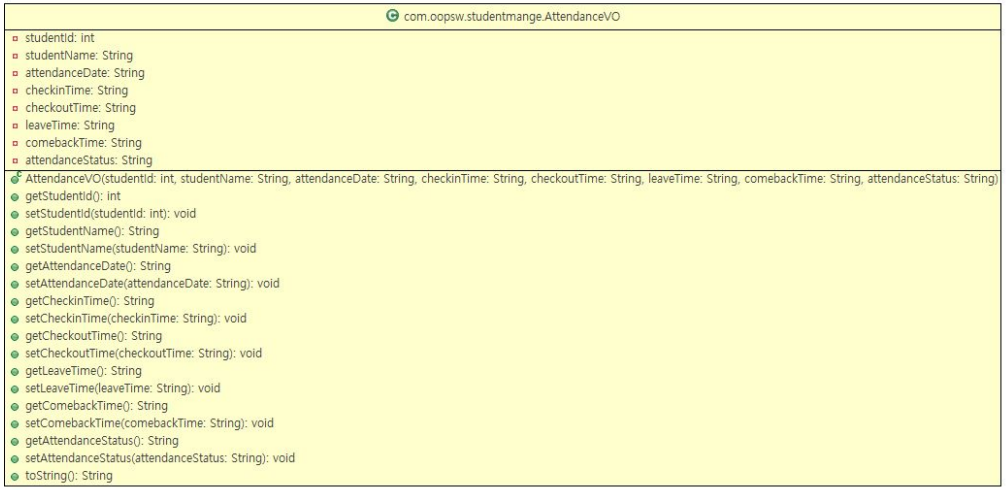
```java
@Test
public void getStudent() {
    Collection<StudentVO> list = dao.getStudents("풀스택 과정");
    for(StudentVO sVo : list){System.out.println(sVo);}
}

@Test
public void addStudent() {
    assertTrue(dao.addStudent(new StudentVO("홍길동", "남", "1700-01-01", "조선시 한양", "mrhong@chosun.go.kr", "010-8282-8282", "의적학")));
}

@Test
public void updateStudent() {
    assertTrue(dao.updateStudent(1, new StudentVO("조선시 평양", "mrhong@korea.go.kr", "010-8282-8282", "도적학")));
}
@Test
public void deleteStudent() {
    assertTrue(dao.deleteStudent(4));
}
@Test
public void getAttendance() {
    Collection<AttendanceVO> list = dao.getAttendances(3, "2025-09-04", "출석");
    for(AttendanceVO aVo: list) System.out.println(aVo);
}
```

구현


com.oopsw.studentmange.StudentManageDAO
getStudents(courseName: String): Collection<StudentVO>
addStudent(studentVO: StudentVO): boolean
updateStudent(studentId: int, studentVO: StudentVO): boolean
deleteStudent(studentId: int): boolean
getAttendances(courseId: int, date: String): Collection<AttendanceVO>
getAttendances(courseId: int, date: String, attendanceStatus: String): Collection<AttendanceVO>
getScores(courseId: int): Collection<ScoreVO>
com.oopsw.studentmange.StudentManageDAOImpl
connection: Connection
StudentManageDAOImpl()
getStudents(courseName: String): Collection<StudentVO>
addStudent(studentVO: StudentVO): boolean
updateStudent(studentId: int, studentVO: StudentVO): boolean
deleteStudent(studentId: int): boolean
getAttendances(courseId: int, date: String): Collection<AttendanceVO>
getAttendances(courseId: int, date: String, attendanceStatus: String): Collection<AttendanceVO>
getScores(courseId: int): Collection<ScoreVO>


com.oopsw.studentmange.ScoreVO
studentId: int
studentName: String
examName: String
score: int
ScoreVO(studentId: int, studentName: String, examName: String, score: int)
getStudentId(): int
setStudentId(studentId: int): void
getStudentName(): String
setStudentName(studentName: String): void
getExamName(): String
setExamName(examName: String): void
getScore(): int
setScore(score: int): void
toString(): String


com.oopsw.studentmange.StudentVO
studentId: int
studentName: String
gender: String
birthDate: String
address: String
email: String
phoneNumber: String
major: String
StudentVO(studentName: String, gender: String, birthDate: String, address: String, email: String, phoneNumber: String, major: String)
StudentVO(address: String, email: String, phoneNumber: String, major: String)
StudentVO(studentId: int, studentName: String, gender: String, birthDate: String, address: String, email: String, phoneNumber: String, major: String)
setStudentId(studentId: int): void
getStudentId(): int
getStudentName(): String
setStudentName(studentName: String): void
getGender(): String
setGender(gender: String): void
getBirthDate(): String
setBirthDate(birthDate: String): void
getAddress(): String
setAddress(address: String): void
getEmail(): String
setEmail(email: String): void
getPhoneNumber(): String
setPhoneNumber(phoneNumber: String): void
getMajor(): String
setMajor(major: String): void
toString(): String


com.oopsw.studentmange.AttendanceVO
studentId: int
studentName: String
attendanceDate: String
checkinTime: String
checkoutTime: String
leaveTime: String
comebackTime: String
attendanceStatus: String
AttendanceVO(studentId: int, studentName: String, attendanceDate: String, checkinTime: String, checkoutTime: String, leaveTime: String, comebackTime: String, attendanceStatus: String)
getStudentId(): int
setStudentId(studentId: int): void
getStudentName(): String
setStudentName(studentName: String): void
getAttendanceDate(): String
setAttendanceDate(attendanceDate: String): void
getCheckinTime(): String
setCheckinTime(checkinTime: String): void
getCheckoutTime(): String
setCheckoutTime(checkoutTime: String): void
getLeaveTime(): String
setLeaveTime(leaveTime: String): void
getComebackTime(): String
setComebackTime(comebackTime: String): void
getAttendanceStatus(): String
setAttendanceStatus(attendanceStatus: String): void
toString(): String

회고

배성윤 : 요구사항 분석부터 db모델링, dao 구현까지 하는것은 쉽게 하지 못할 경험이었습니다.
이 과정을 통해 단순히 개발만이 아닌 분석, 설계, 협동의 어려움을 알게 되었고 같이 단순히 공부만으로는 알 수 없는 경험을 했습니다.
그 중 가장 기억에 남는 것은 db모델링 중 데이터를 삭제하는 방법과 dao구현을 위해 vo를 어느 범위로 구현해야 하는지에 고민했을 때 입니다.

처음 학생의 정보를 삭제하는 것을 고민할 때 단순히 db에서 학생의 데이터를 삭제하면 될 줄 알았습니다.
하지만 db 구조 상 관계가 형성이 되었기에 학생의 정보를 삭제하기 위해서는 해당 데이터가 참조된 테이블들의 모든 데이터를 같이 지워야 했습니다.
여기서 어느 범위까지 데이터를 지울 지 선택을 해야했고 학생의 정보와 관련된 데이터는 유지하되, 학생의 개인정보의 값을 변경하여 식별할 수 없게 하는것으로 삭제 기능을 구현했습니다.

또한 vo의 경우 db에서 데이터를 가져오기 위해 범위를 어떻게 지정해야 할 지 고민했습니다. 단순히 db의 테이블을 기준으로 할지, ui상의 화면에 해당하는 데이터들을 기준으로 할지 고민했고 저는 후자를 선택했습니다.

이러한 고민은 저에게 문제를 해결 하는데 있어 중요한 것은 어떤 기준을 가지고 무엇을 선택하는지의 문제지, 정답은 없다는 것을 실감하게 되었습니다.

심재용 : db에서 사용자 정보 기본키를 삭제하려고 하니 곳곳에 참조가 돼서 삭제하기가 어려웠습니다.

이를 해결하기 위해 CASCADE라는 문법을 사용하려 했는데 기본키를 지우기 위해 관련 데이터를 모두 지우는 게 맞는가 라는 의문이 들었습니다. 알아보니 사용자의 기본키를 삭제하는 것만으로도 그 사용자와 관계 있는 많은 사람들의 데이터가 수정이 된다는 것을 알고 삭제를 말아야겠다는 결정을 내렸습니다.

이번 경험으로 제가 사용하는 서비스들을 볼 때마다 db가 어떻게 구성돼있을까 생각해보게 되는 계기가 되었습니다.