대학 학사 관리 및 수강신청 시스템 요구사항 분석 명세서

1. 시스템 개요 (System Overview)

본 시스템은 대학 학사 행정 전반(입학, 수강신청, 성적, 학적, 졸업)을 통합 관리하기 위한 학사 정보 시스템(Academic Information System)이다.
시스템의 주요 목표는 다음과 같다:

학생, 교수, 학과, 과목, 개설강좌, 수강 등의 학사 데이터를 통합·일관되게 관리

수강신청 과정에서의 실시간 검증(정원, 시간표, 선수·동시수강, 재수강)

외국인 학생의 식별 정보(여권번호) 처리를 포함한 유연한 신원 구조 지원

성적 입력 및 확정, 학점·평점 계산, 졸업 요건 검증 기능 제공

데이터 무결성과 학사 규정 준수 보장

학사 운영의 효율성과 사용자 경험(학생/교수)의 향상

※ 본 명세서는 분석·설계 단계 산출물이며,

UI 설계, 트랜잭션 코드, 운영환경 구성 등은 포함하지 않는다.

2. 데이터 요구사항 (Data Requirements)

각 엔터티는 실제 ERD 및 DDL 과 동일한 구조로 정의된다.
명명 규칙은 snake_case 기준(SQL 기준)이며, 문서 내 표기는 PascalCase 사용 가능.

2.1 학생 (Student)

식별자

student_id (PK)

관리 항목

name_kr : 한글 이름

name_en : 영문 이름 (선택)

rrn : 주민등록번호(내국인), 여권번호와 상호 배타적·필수

passport_no : 여권번호(외국인)

nationality : 국적

dept_code (FK → Department)

status : 학적 상태 (재학, 휴학, 복학, 졸업, 자퇴 등)

admission_date : 입학일

email

phone

address

특기사항

CHECK (rrn IS NOT NULL OR passport_no IS NOT NULL)

외국인 처리 규칙은 ERD 및 DDL 에 동일 적용.

2.2 교수 (Professor)

식별자

professor_id (PK)

관리 항목

name_kr, name_en

rrn 또는 passport_no

dept_code (FK)

position : 직급

office_location

office_phone

email

hire_date

2.3 학과 (Department)

식별자

dept_code (PK)

관리 항목

dept_name

college_name

office_location

office_phone

2.4 과목 (Course)

식별자

course_code (PK)

관리 항목

course_name_kr, course_name_en

credit

course_type : 전공필수·전공선택·교양 등

recommended_year : 권장 이수 학년

is_deleted : 폐지 여부

2.5 선수·동시수강 관계 (Prerequisite)

(ERD 의 구조 그대로 반영)

식별자

(course_code, prereq_course_code) 복합 PK

관리 항목

coreq_flag (BOOLEAN):

FALSE → 선수과목 (선이수 필수)

TRUE → 동시수강 허용

특기사항

CourseRelation 과 분리하지 않고 Prerequisite 단일 테이블로 통일

ERD 와 명세서의 일관성 확보

2.6 개설강좌 (OpenCourse)

식별자

open_course_id (PK)

관리 항목

year

term

course_code (FK)

section : 분반

professor_id (FK)

room : 강의실

capacity : 수강정원

enrolled_count : 현재 신청 인원

is_canceled : 폐강 여부

제약

UNIQUE (year, term, course_code, section)

정원과 현재 인원은 정합성 검증 대상

2.7 강의시간 (LectureSchedule)

식별자

schedule_id (PK)

관리 항목

open_course_id (FK)

day_of_week

start_period

end_period

is_consecutive : 연강 여부

2.8 수강 (Enrollment)

식별자

enrollment_id (PK)

관리 항목

student_id (FK)

open_course_id (FK)

requested_at

status : 신청, 취소, 대기 등

is_retake : 재수강 여부

제약

UNIQUE (student_id, open_course_id)

2.9 성적 (Grade)

(ERD 구조와 완전히 동일하도록 복원됨)

식별자

grade_id (PK)

관리 항목

enrollment_id (FK)

midterm_score

final_score

final_grade : A+, B0, C+, F 등

grade_point : 평점(4.5 기준 등)

grade_confirmed

confirmed_at

confirmed_by (FK → Professor)

특기사항

Enrollment 와 완전히 분리되어 ERD 와 명세서 일치 보장

2.10 감사로그 (AuditLog)

(ERD 에 존재하는 경우 포함)

식별자

log_id (PK)

관리 항목

table_name

record_id

operation (INSERT/UPDATE/DELETE)

changed_by

changed_at

remarks

3. 기능적 요구사항 (Functional Requirements)

본 시스템은 관리자, 교수, 학생의 역할(Role-Based) 기반으로 기능을 제공한다.

기초 데이터 관리 기능 (Administrator / Academic Affairs)

3.1.1 학과/학생/교수/과목 기본 정보 관리

학사관리자는 다음 항목을 등록·수정·조회·비활성화할 수 있다:

학과(Department), 학생(Student), 교수(Professor), 과목(Course)

삭제는 연관 데이터가 없는 경우에 한하여 허용

예: 학생이 수강 기록이 있으면 삭제 불가 → 비활성화 처리

3.1.2 외국인 식별 정보 처리

학생 정보는 다음 제약을 만족해야 한다:

(rrn IS NOT NULL) OR (passport_no IS NOT NULL)

두 값이 모두 NULL 일 수 없다.

3.2 강좌 개설 기능 (Professor / Academic Affairs)

1) 개설강좌 생성

강의 개설 시 필수 정보:

학년도(year)

학기(term)

과목(course_code)

분반(section)

담당교수(professor_id)

강의실(room)

강의시간(1:N 구조)

2) 중복 개설 방지 규칙

DB 레벨에서 다음 UNIQUE 제약을 사용하여 보장한다:

UNIQUE(year, term, course_code, section)

3.3 수강신청 기능 (Student)

수강 신청은 시스템의 핵심이며, 다음 검증 절차를 반드시 통과해야 한다.

1) 수강신청 기능

학생은 다음 기능을 수행할 수 있다:

개설강좌 목록 조회(검색, 필터 포함)

수강 신청

수강 취소

시간표 조회

2) 수강신청 검증 로직 (핵심)

신청 시 다음 모든 조건을 만족해야 한다.

1. 정원 초과 여부 확인

IF enrolled_count >= capacity → 신청 불가

동시성 제어는 트랜잭션 수준에서 처리

2. 시간표 중복 여부 확인

학생과 교수 모두 중복 금지.

학생 예시:

EXISTS schedule

WHERE (day_of_week, period_range) overlaps

    with student's existing courses

→ 신청 불가

교수 예시:

EXISTS schedule

WHERE professor teaches another class at same time

→ 개설 불가

3. 선수과목 이수 여부 확인

coreq_flag = FALSE 인 경우 선이수 필수

이수 인정 조건:

final_grade != 'F' (D 이상이면 인정)

4. 동시수강 허용 여부 확인

coreq_flag = TRUE 인 관계만 동시수강 허용

그 외는 선수과목을 반드시 이수해야 함

5. 학기별 최대 신청 학점 초과 여부 확인

학칙 기준 ex)

정규학기: 최대 18 학점

직전학기 성적 우수자: 21 학점 (정책 케이스별로 확장 가능)

시스템은 신청 전 학생의 누적 신청 학점을 계산해야 한다.

6. 재수강 여부에 따른 제한

재수강 인정 기준:

최종학점 C+ 이하 (grade_point < 2.5)

기존 성적이 B0 이상이면 재수강 불가

7. 중복 수강 금지

DB 제약으로 보장:

UNIQUE(student_id, open_course_id)

8. 학적상태 검증

‘재학’, ‘복학’ 상태인 경우만 신청 가능

휴학/제적/졸업/자퇴 상태는 신청 불가

3.4 성적 관리 기능 (Professor)

1) 성적 입력

교수는 자신의 강좌의 수강생에 대해 다음을 입력한다:

중간 점수(midterm_score)

기말 점수(final_score)

최종 학점(final_grade)

평점(grade_point)

2) 성적 확정

점수 입력 후 교수는 성적을 확정할 수 있다:

grade_confirmed = TRUE

confirmed_at 설정

confirmed_by = 교수 ID

성적 확정 이후에는 수정 불가(관리자 예외 기능 제외).

3) 학생 성적 조회

성적 확정 이후에만 학생이 조회 가능

3.5 학적 및 졸업 요건 기능

1) 학적 상태 관리

관리자는 학생의 상태를 관리할 수 있다:

재학, 휴학, 복학, 졸업, 자퇴 등

2) 졸업 요건 검증 기능

총 이수학점

전공필수 이수 여부

F 학점 미이수 처리

재수강 반영 규칙 적용

시스템은 이를 자동 계산하여 졸업 가능 여부를 안내해야 한다.

3.6 감사 로그 기능 (선택/확장)

1) 감사 로그 기록

학생·교수·과목·개설강좌 등 주요 데이터 수정 시:

누가

언제

어떤 값에서 어떤 값으로 바꿨는지

이를 AuditLog 에 적재할 수 있다.

4. 비즈니스 규칙 (Business Rules) & 검증 로직 (Validation Logic)

본 섹션은 수강신청·성적·재수강·선수과목·정원 관리 등 시스템의 핵심 정책을 명확하게 정의한다.

4.1 정원 관리 규칙 (Capacity Management)

규칙 1 — 정원 제한

enrolled_count >= capacity
→ 신청 불가

규칙 2 — 동시성 제어 (Transaction Level Enforcement)

수강신청 시 “정원 경합”을 방지하기 위해 DB 트랜잭션에서 다음 절차 수행:

절차

강좌 row 잠금:

```
SELECT capacity, enrolled_count
FROM open_course
WHERE open_course_id = :id
FOR UPDATE;
```

정원 확인

수강 신청 INSERT

enrolled_count = enrolled_count + 1 업데이트

COMMIT

규칙 3 — 취소 시 인원 감소

```
UPDATE open_course
SET enrolled_count = enrolled_count - 1
WHERE open_course_id = :id
  AND enrolled_count > 0;
```

규칙 4 — 정합성 점검

정원과 실제 count 가 맞는지 야간 배치로 점검:

```
SELECT oc.open_course_id
FROM open_course oc
LEFT JOIN enrollment e
ON oc.open_course_id = e.open_course_id
AND e.status = 'APPROVED'
GROUP BY oc.open_course_id
HAVING COUNT(e.enrollment_id) != oc.enrolled_count;
```

2) 정원 관리 동시성 제어 및 업데이트 책임

정원(enrolled_count)의 증감 처리는 다음 두 가지 방식 중 하나로 수행할 수 있다.
본 시스템에서는 안정성과 정합성을 위해 데이터베이스 트리거 기반 처리를 공식 정책으로 채택한다.

방법 1: 애플리케이션 로직에서 직접 업데이트

애플리케이션 코드에서 수강 신청 승인 시 정원을 증가시키는 방식이다.

```
UPDATE open_course
SET enrolled_count = enrolled_count + 1
WHERE open_course_id = :course_id;
```

장점: 구조 단순

단점: 동시 신청 시 Race Condition 위험이 높음 → 부정확한 정원 값 발생 가능

방법 2: 데이터베이스 트리거 기반 자동 업데이트 (권장)

수강 상태가 APPROVED 로 변경되는 순간 정원이 자동 반영되도록 트리거를 둔다.

예시:

```
CREATE TRIGGER trg_enrollment_count
AFTER INSERT OR UPDATE OR DELETE ON enrollment
FOR EACH ROW
EXECUTE FUNCTION fn_increment_enrolled_count();
```

트리거 내부 로직은 다음을 수행한다.

INSERT 또는 UPDATE 이벤트에서 status='APPROVED'가 된 경우 → 정원 +1

UPDATE 또는 DELETE 이벤트에서 status='APPROVED'가 해제된 경우 → 정원 -1

enrolled_count 음수 방지 처리 포함

공식 정책

본 명세서는 방법 2(트리거 기반 정원 관리) 를 채택함을 명시한다.
애플리케이션 레벨에서의 정원 직접 수정은 허용하지 않는다.

4.2 시간표 중복 검증 (Schedule Conflict Policy)

규칙 1 — 동일 요일·교시 중복 금지

학생은 같은 요일, 겹치는 시간대의 강의 신청 불가

교수도 동일 시간대에 두 강좌 배정 불가

검사 로직:

```
SELECT 1
FROM lecture_schedule ls1
JOIN lecture_schedule ls2
ON ls1.day_of_week = ls2.day_of_week
AND ls1.start_period <= ls2.end_period
AND ls2.start_period <= ls1.end_period
WHERE ls2.open_course_id = :new_course
AND ls1.open_course_id IN (
SELECT open_course_id
FROM enrollment
WHERE student_id = :student
 );
```

4.3 선수과목 검증 (Prerequisite Policy)

규칙 1 — 선이수 필수 조건

coreq_flag = FALSE 인 경우 반드시 선수과목을 이미 이수해야 한다.

규칙 2 — 이수 인정 기준

final_grade != 'F'

즉, D0 이상은 모두 이수로 인정.

검증 쿼리

```
SELECT COUNT(*) AS unsatisfied
FROM prerequisite p
WHERE p.course_code = :target_course
AND p.coreq_flag = FALSE
AND NOT EXISTS (
SELECT 1
FROM open_course oc
JOIN enrollment e ON e.open_course_id = oc.open_course_id
JOIN grade g ON g.enrollment_id = e.enrollment_id
WHERE e.student_id = :student_id
AND oc.course_code = p.prereq_course_code
AND g.grade_confirmed = TRUE
AND g.final_grade != 'F'
 );
```

unsatisfied > 0 이면 미이수 선수과목 존재 → 신청 불가.

3) 다단계(Chain) 선수과목 처리 정책

선수과목 체계가 다음과 같이 연속될 수 있다.

C → B → A

(A 신청 시 B 필요, B 신청 시 C 필요)

본 시스템의 정책은 다음과 같다.

정책: “직접 선수과목만 검증” (Non-Recursive Checking)

A 과목의 직접 선수과목이 B 이면 → B 만 이수 여부를 확인

B 의 선수과목 C 까지는 검증하지 않음

재귀 탐색을 하지 않는 이유:

무한 재귀 위험 제거

학사 규정 대부분이 “직접 선수과목”만 요구

구현 복잡도 감소

따라서,
Prerequisite(course_code, prereq_course_code)
테이블에 명시된 직접 연결만 검증한다.

4.4 동시수강(Corequisite) 정책

규칙 1 — coreq_flag = TRUE

해당 과목은 선수과목을 이수하지 않아도 같은 학기에 동시 신청 가능

규칙 2 — 동시수강 과목도 시간표 중복 검증 대상

동시수강이라고 해서 시간표 중복을 허용하지 않는다.

4.5 재수강 규칙 (Retake Policy)

규칙 1 — 재수강 대상

동일 course_code 에 대해 이전 성적이 존재해야 한다.

규칙 2 — 재수강 허용 조건

가장 최근 이수 성적이:

grade_point < 2.5 → C+ 이하

규칙 3 — 재수강 불가 조건

이전 이수 성적이 B0(3.0) 이상 → 재수강 불가

검증 쿼리

```
SELECT g.final_grade, g.grade_point
FROM enrollment e
JOIN open_course oc ON e.open_course_id = oc.open_course_id
JOIN grade g ON g.enrollment_id = e.enrollment_id
WHERE e.student_id = :student
AND oc.course_code = :course
AND g.grade_confirmed = TRUE
ORDER BY e.requested_at DESC
LIMIT 1;
```

4.6 학점 제한 규칙 (Credit Limit Policy)

기본 규칙

1 학기 최대 신청 가능 학점 = 18 학점

확장 규칙

직전학기 평점이 4.0 이상 → 21 학점

검증 로직

```
SELECT SUM(c.credit)
FROM enrollment e
JOIN open_course oc ON oc.open_course_id = e.open_course_id
JOIN course c ON c.course_code = oc.course_code
WHERE e.student_id = :student
AND oc.year = :year
AND oc.term = :term;
```

4.7 학적 상태 규칙 (Academic Status Policy)

신청 가능 상태

재학

복학

신청 불가 상태

휴학

자퇴

제적

졸업

4.8 성적 확정(Grade Confirmation) 정책

규칙

교수만 성적 확정 가능

확정된 후 수정 불가 (관리자 override 만 가능)

확정 조건

midterm_score, final_score 존재

final_grade, grade_point 입력 완료

확정 처리

UPDATE grade

SET grade_confirmed = TRUE,

confirmed_at = NOW(),

confirmed_by = :professor_id

WHERE grade_id = :id;

4.9 감사 로그(Audit Log) 규칙

기록 조건

데이터 변경이 발생하는 경우:

student, professor, course, department

open_course, lecture_schedule

enrollment, grade

저장 컬럼:

table_name

record_id

operation (INSERT/UPDATE/DELETE)

changed_at

changed_by

remarks

데이터 사전 (Data Dictionary)

모든 테이블/칼럼명은 snake_case 를 사용. 타입 표기는 PostgreSQL 스타일(필요시 DBMS 에 맞게 조정).

1) department (학과)

dept_code — VARCHAR(10) — PK, NOT NULL
학과 식별자

name — VARCHAR(100) — NOT NULL
학과명

college — VARCHAR(100) — NULL
단과대학명

office — VARCHAR(50) — NULL
사무실 위치

phone — VARCHAR(20) — NULL
사무실 전화번호

2) professor (교수)

professor_id — VARCHAR(20) — PK, NOT NULL

name — VARCHAR(100) — NOT NULL

name_en — VARCHAR(100) — NULL

rrn — CHAR(13) — NULL

passport_no — VARCHAR(30) — NULL

dept_code — VARCHAR(10) — FK → department(dept_code), NOT NULL

position — VARCHAR(50) — NULL

office — VARCHAR(50) — NULL

office_phone — VARCHAR(20) — NULL

email — VARCHAR(100) — NULL

hire_date — DATE — NULL

CONSTRAINT professor_ident_ck — CHECK ((rrn IS NOT NULL) OR (passport_no IS NOT NULL))
rrn 또는 passport_no 중 적어도 하나 필수

3) student (학생)

student_id — VARCHAR(20) — PK, NOT NULL

name_kr — VARCHAR(100) — NOT NULL

name_en — VARCHAR(100) — NULL

rrn — CHAR(13) — NULL

passport_no — VARCHAR(30) — NULL

nationality — VARCHAR(50) — NULL

dept_code — VARCHAR(10) — FK → department(dept_code), NOT NULL

status — VARCHAR(20) — NOT NULL
예: 'enrolled','leave','withdrawn','graduated' 등

admission_date — DATE — NOT NULL

address — VARCHAR(200) — NULL

phone — VARCHAR(20) — NULL

email — VARCHAR(100) — NULL

CONSTRAINT student_ident_ck — CHECK ((rrn IS NOT NULL) OR (passport_no IS NOT NULL))

4) course (과목)

course_code — VARCHAR(20) — PK, NOT NULL

course_name_kr — VARCHAR(200) — NOT NULL

course_name_en — VARCHAR(200) — NULL

credit — DECIMAL(3,1) — NOT NULL

course_type — VARCHAR(50) — NULL
전공필수/전공선택/교양 등

recommended_year — INT — NULL

is_deleted — BOOLEAN — DEFAULT FALSE — NOT NULL
폐지 여부

5) open_course (개설강좌)

open_course_id — BIGSERIAL — PK, NOT NULL

year — INT — NOT NULL

term — VARCHAR(10) — NOT NULL
학사 규정에 맞게 값 통일 (예: '1','2' 또는 'Spring','Fall')

course_code — VARCHAR(20) — FK → course(course_code), NOT NULL

section — VARCHAR(10) — NOT NULL

professor_id — VARCHAR(20) — FK → professor(professor_id) — NULL 허용(미배정 가능)

room — VARCHAR(50) — NULL

capacity — INT — NOT NULL

enrolled_count — INT — DEFAULT 0, NOT NULL

is_canceled — BOOLEAN — DEFAULT FALSE, NOT NULL

CONSTRAINT open_course_uk — UNIQUE(year, term, course_code, section)
동일 조합 중복 개설 방지

6) lecture_schedule (강의시간)

schedule_id — BIGSERIAL — PK, NOT NULL

open_course_id — BIGINT — FK → open_course(open_course_id), NOT NULL

day_of_week — SMALLINT — NOT NULL
1=Mon ... 7=Sun

start_period — INT — NOT NULL

end_period — INT — NOT NULL

is_consecutive — BOOLEAN — DEFAULT FALSE, NOT NULL

CONSTRAINT lecture_slot_ck — CHECK (start_period <= end_period)

7) enrollment (수강)

enrollment_id — BIGSERIAL — PK, NOT NULL

student_id — VARCHAR(20) — FK → student(student_id), NOT NULL

open_course_id — BIGINT — FK → open_course(open_course_id), NOT NULL

requested_at — TIMESTAMP — DEFAULT now(), NOT NULL

status — VARCHAR(20) — NOT NULL
예: 'applied','waiting','cancelled','approved','rejected'

is_retake — BOOLEAN — DEFAULT FALSE, NOT NULL

created_by — VARCHAR(50) — NULL

created_at — TIMESTAMP — DEFAULT now(), NOT NULL

CONSTRAINT enrollment_uk — UNIQUE (student_id, open_course_id)
동일 학생의 중복 신청 방지

8) grade (성적)

grade_id — BIGSERIAL — PK, NOT NULL

enrollment_id — BIGINT — FK → enrollment(enrollment_id) ON DELETE CASCADE, NOT NULL

midterm_score — DECIMAL(5,2) — NULL

final_score — DECIMAL(5,2) — NULL

final_grade — VARCHAR(3) — NULL
예: 'A+', 'A0', 'B+', 'C0', 'F'

grade_point — DECIMAL(3,2) — NULL
예: 4.50, 3.50 등

grade_confirmed — BOOLEAN — DEFAULT FALSE, NOT NULL

confirmed_at — TIMESTAMP — NULL

confirmed_by — VARCHAR(20) — FK → professor(professor_id) — NULL

비고: grade_confirmed = TRUE 일 때만 final_grade·grade_point 을 신뢰하여 졸업·재수강 판정에 사용.

9) prerequisite (선수/동시수강 관계)

course_code — VARCHAR(20) — FK → course(course_code), NOT NULL

prereq_course_code — VARCHAR(20) — FK → course(course_code), NOT NULL

coreq_flag — BOOLEAN — DEFAULT FALSE, NOT NULL
TRUE 면 동시수강 허용

PRIMARY KEY (course_code, prereq_course_code)

10) audit_log (감사로그)

log_id — BIGSERIAL — PK, NOT NULL

table_name — VARCHAR(50) — NOT NULL

record_id — VARCHAR(100) — NOT NULL

operation — VARCHAR(10) — NOT NULL
INSERT / UPDATE / DELETE

changed_by — VARCHAR(50) — NULL

changed_at — TIMESTAMP — DEFAULT now(), NOT NULL

remarks — TEXT — NULL

11) 추가 보조 테이블(권장)

system_parameter — 시스템 운영 변수(예: max_credits_per_semester, retake_grade_threshold 등) 보관

param_key VARCHAR, param_value VARCHAR 등

user_account — 로그인·권한 관리(외부 ID 제공 시 연결)

데이터 타입·정책 요약 (참고)

정수형: INT/SMALLINT/BIGSERIAL 사용(자동증가: BIGSERIAL)

문자열: VARCHAR(n)로 길이 제한 권장

날짜/시간: DATE, TIMESTAMP (tz 필요하면 TIMESTAMP WITH TIME ZONE)

소수: 평점·학점 등은 DECIMAL(3,2) 또는 DECIMAL(3,1)로 정밀도 설정

민감정보: rrn 등은 실제 DB 적용시 암호화/접근제어 필요

제약조건 (Constraints) & Naming Rule

이 섹션은 전체 스키마가 일관성 있게 동작하도록 하기 위한 식별자 규칙, 외래키 규칙, Unique, Check, Naming 규칙 등을 정의한다.
ERD, 명세서, DDL 전부 이 규칙을 기준으로 만들어졌다는 것을 보장한다.

6.1 명명 규칙(Naming Rules)

1) 테이블(Table) 규칙

snake_case 소문자

단수형 사용

예:

student, course, open_course, lecture_schedule, grade

2) 컬럼(Column) 규칙

snake_case

PK 는 일반적으로 {table}_id

ex) open_course_id, enrollment_id, grade_id

3) 제약조건(Naming) 규칙

제약명의 통일성을 위해 다음 형식을 강제한다:

제약 종류 규칙

PK	{table}_pk
FK	{table}_{column}_fk
UNIQUE	{table}_{column(s)}_uk
CHECK	{table}_{column}_ck

예시

student_ident_ck

open_course_uk (year, term, course_code, section)

prerequisite_pk

4) 인덱스(Index) 규칙

성능 최적화를 위해 다음 유형의 인덱스를 생성 권장:

FK 컬럼: 자동 생성 or idx_{table}_{column}

자주 조회되는 컬럼:

course_code, professor_id, dept_code

수강신청 시 자주 참조되는 조합 인덱스:

(student_id, open_course_id) ← UNIQUE

(open_course_id, student_id) ← 조회용 보조 인덱스 가능

6.2 기본 제약조건(Primary Constraints)

1) Primary Key

모든 테이블은 단일 PK 또는 복합 PK 를 반드시 가진다.

open_course.open_course_id

enrollment.enrollment_id

grade.grade_id

prerequisite(course_code, prereq_course_code) ← 복합 PK

6.3 외래키 제약(Foreign Key Constraints)

From 테이블	**FK 컬럼**	**To 테이블**	**비고**
student	dept_code	department	필수
professor	dept_code	department	필수
open_course	course_code	course	필수

From 테이블	FK 컬럼	To 테이블	비고
open_course	professor_id	professor	교수 미배정 시 NULL 허용
lecture_schedule	open_course_id	open_course	강좌 삭제 시 CASCADE 금지
enrollment	student_id	student	필수
enrollment	open_course_id	open_course	필수
grade	enrollment_id	enrollment	CASCADE 삭제
grade	confirmed_by	professor	NULL 가능

6.4 UNIQUE 제약조건

핵심 정책 기반으로 반드시 존재해야 하는 제약.

1) 개설강좌 중복 방지

UNIQUE(year, term, course_code, section)

2) 학생 중복 신청 방지

UNIQUE(student_id, open_course_id)

3) 선수과목 PK = UNIQUE

(course_code, prereq_course_code) → PRIMARY KEY

6.5 CHECK 제약조건

필수 정책을 데이터 레벨에서 보장.

1) 학생/교수 식별정보

CHECK ((rrn IS NOT NULL) OR (passport_no IS NOT NULL))

2) 학기 구분

(기관 표준에 맞게 필요 시 적용)

예: term IN ('1', '2', 'Summer', 'Winter')

3) 강의시간 범위

CHECK (start_period <= end_period)

6.6 ON DELETE / ON UPDATE 규칙

기본 원칙:

학생/교수/학과/과목은 물리 삭제 금지 → 비활성화 정책

수강(enrollment) 삭제 시 grade 는 CASCADE 로 삭제 가능

open_course 삭제 시 lecture_schedule, enrollment 는 CASCADE 금지
이유: 학기 기록 보존 필요

6.7 트리거(Trigger) 정책 (선택적 정의지만 권장)

다음 업무는 트리거로 처리 가능:

1. 수강신청 생성 시 enrolled_count 자동 증가

(또는 애플리케이션 로직으로 처리—둘 중 하나 반드시 문서화)

2. 수강취소 시 감소

3. 감사로그(Audit) 기록 자동 생성

트리거 예시(개념 정의):

AFTER UPDATE OR INSERT OR DELETE ON student

FOR EACH ROW

EXECUTE audit_log_trigger();

6.8 데이터 무결성 규칙 정리 (Integrity Rules Summary)

엔티티 무결성

모든 테이블의 PK 는 NULL 불가

참조 무결성

FK 는 반드시 유효한 PK 를 참조

미배정 허용 시 FK 에 NULL 허용

도메인 무결성

점수 범위, 학점 문자 규격, 학기 값 등 CHECK 제약으로 보장

비즈니스 무결성

선수과목/재수강/시간표 중복 등은 DB+애플리케이션 양측에서 보장

6.9 제약 충돌 방지 규칙

COLUMN 명 변경 시 ERD/DDL/명세서 3 곳을 동시에 수정

ENUM 처럼 보이는 문자열은 표준 문자열 사용

NOT NULL 컬럼 추가 시 기존 데이터 영향 고려

CASCADE 사용은 최소화 (학적 기록 보존 필요 때문)

ERD 논리 구조 설명 (Logical ERD Description)

7.1 ERD 전체 요약

본 시스템의 ERD 는 다음 10 개 주요 엔티티로 구성된다:

Department

Professor

Student

Course

Prerequisite

OpenCourse

LectureSchedule

Enrollment

Grade

AuditLog

이들은 학사 운영을 위한 기본 구조를 구성하며,
모든 관계는 PK-FK 기반의 논리적 일관성을 가진다.

7.2 엔티티별 상세 관계

1) Department — Professor / Student 의 상위 엔티티

Department (1) — (N) Professor

관계: 1:N

제약: professor.dept_code → FK → department.dept_code

의미: 한 학과에는 여러 교수 소속 가능

Department (1) — (N) Student

관계: 1:N

학생도 학과 소속 필수

2) Course — OpenCourse

Course (1) — (N) OpenCourse

하나의 과목(Course)은 여러 개설강좌(OpenCourse)를 가질 수 있음

제약: open_course.course_code FK

3) Course — Prerequisite (선수·동시수강 관계)

Course (1) — (N) Prerequisite

course_code 를 “본 과목”으로 보고

prereq_course_code 를 “선수과목”으로 본다.

특징:

복합 PK (course_code, prereq_course_code)

coreq_flag = TRUE 면 “동시수강 허용”

4) Professor — OpenCourse

Professor (1) — (N) OpenCourse

한 교수는 여러 강좌를 개설할 수 있음

professor_id 는 NULL 가능(임시 미배정)

5) OpenCourse — LectureSchedule

OpenCourse (1) — (N) LectureSchedule

하나의 개설강좌가 여러 강의시간 slot 가지는 구조

시간표 중복 검증을 위해 이 구조 고정

6) Student — Enrollment — OpenCourse

(수강 구조의 핵심)

Student (1) — (N) Enrollment

한 학생은 여러 강좌를 신청 가능

OpenCourse (1) — (N) Enrollment

한 개설강좌에 여러 학생 등록

결론:

Student ↔ OpenCourse 는 N:M

Enrollment 가 관계 해소 엔티티

제약:

UNIQUE(student_id, open_course_id)

7) Enrollment — Grade

Enrollment (1) — (1) Grade

1:1 관계

Enrollment 가 삭제되면 Grade 자동 삭제 (CASCADE)

성적이 Enrollment 와 분리된 이유:

재수강/평점/확정일자 관리 때문

ERD 와 명세서 불일치 해결의 핵심 포인트

8) AuditLog — 모든 테이블과 느슨한 관계

AuditLog 는 다음과 같은 구조를 갖는 독립 엔터티:

table_name: 문자열

record_id: 문자열

operation: INSERT/UPDATE/DELETE

데이터 변경 시 Trigger 또는 Application 에서 적재.

7.3 Cardinality 표 (한눈에 보는 관계 요약)

관계	**Cardinality**	**설명**
Department – Professor	1:N	학과 소속 교수
Department – Student	1:N	학과 소속 학생
Course – OpenCourse	1:N	한 과목의 여러 개설강좌
Course – Prerequisite	1:N	선수 과목 정의
Professor – OpenCourse	1:N	교수당 여러 강좌
OpenCourse – LectureSchedule	1:N	강좌당 여러 시간 슬롯
Student – Enrollment	1:N	학생의 수강기록
OpenCourse – Enrollment	1:N	강좌의 수강기록
Enrollment – Grade	1:1	수강기록의 성적

7.4 ERD 구조 시각적 기준

1) 엔티티 박스명은 PascalCase 로:

Student

Professor

Course

OpenCourse

LectureSchedule

Enrollment

Grade

Prerequisite

Department

AuditLog

(DB 내부는 snake_case 지만 ERD 박스는 PascalCase 사용)

2) 필드명은 snake_case 그대로 기입:

예:

student_id : PK

dept_code : FK

passport_no

admission_date

3) 관계선 표기 규칙

실선(Solid line): 필수 FK

점선: 선택 FK

까마귀발로 N 표현

막대기(|)로 1 표현

예:

Department |——< Professor

Enrollment |——| Grade (1:1)

Course |——< Prerequisite

Student |——< Enrollment >——| OpenCourse

4) Prerequisite 엔티티를 Course 옆에 위치

왼쪽이 course_code(본 과목),

오른쪽이 prereq_course_code(선수과목)

5) Grade 는 Enrollment 바로 오른쪽에 연결

(Enrollment – Grade = 1:1 명확히 표현)

6) LectureSchedule 은 OpenCourse 아래 배치

(1:N 구조 직관적으로 보이도록)

7.5 ERD 검증 체크리스트

다음 체크리스트를 통과하면 "ERD 완성본"이라고 봄:

[ ] Enrollment 에 성적 컬럼 없음

[ ] Grade 엔티티 존재

[ ] Prerequisite 단일 엔티티로 통일 (coreq_flag 포함)

[ ] Student/Professor 에 (rrn OR passport_no) CHECK 필요

[ ] OpenCourse 에 UNIQUE(year, term, course_code, section)

[ ] Enrollment UNIQUE(student_id, open_course_id)

[ ] Relationship 표기 1:N/N:M 모두 정확

[ ] 테이블명/컬럼명 모두 snake_case 유지

[ ] 모든 FK, PK 표기됨

부록 B. 학점 체계 (Grading Scale)

본 시스템에서 사용하는 학점–평점 매핑 기준은 다음과 같다.

학점 평점(Grade Point) 100 점 환산 비고

A+	4.5	95~100	최우수
A0	4.0	90~94	우수
B+	3.5	85~89	
B0	3.0	80~84	
C+	2.5	75~79	**재수강 허용 기준선**
C0	2.0	70~74	최소 이수 기준
D+	1.5	65~69	
D0	1.0	60~64	최소 이수
F	0.0	0~59	미이수

적용 규칙

재수강 허용 기준: grade_point < 2.5 (C+ 미만)

선수과목 이수 기준: final_grade != 'F' (D0 이상)

졸업 요건: 평균 평점 2.0 이상 유지

부록 C. 코드 값 표준 (Code Book)

본 시스템에서 사용하는 주요 코드 값은 다음과 같다.

1. 학기 코드 (term)

코드	의미
1	1 학기
2	2 학기
S	하계학기(Summer)
W	동계학기(Winter)

2. 학적 상태 (student.status)

코드	의미
ENROLLED	재학
LEAVE	휴학
WITHDRAWN	자퇴

코드	의미
GRADUATED	졸업

3. 수강 상태 (enrollment.status)

코드	의미
APPLIED	신청
APPROVED	승인
WAITING	대기
CANCELLED	취소