헬스케어 데이터베이스 실습

SQL 기반 데이터 분석 · 정규화 · 성능 최적화

GOAL

- 실제 **헬스케어 데이터를 RDBMS에 저장**
- **정규화(Normalization)** 로 이상현상 제거
- **인덱스(Index)** 로 성능 향상
- SQL과 Python으로 **통계 분석 및 위험도 탐지**

데이터 구성 예시

patient_id	name	age	gender	weight	height	date	BMI	blood_pressure	glucos
P001	김철수	45	M	72	175	2025-10-01	23.5	138/90	150
P001	김철수	45	M	72	175	2025-10-15	23.5	142/92	180

정규화란? (Normalization)

정규화는 데이터 중복을 최소화하고,
데이터의 무결성(일관성) 을 유지하기 위한 데이터베이스 설계 과정

정규화를 하지 않으면?

1. 삽입 이상 (Insertion Anomaly)

- 새 환자 등록 시, 건강 지표가 없어도 등록 해야함
  → NULL이 넘쳐나거나 입력 불가

2. 갱신 이상 (Update Anomaly)

- 환자 이름 변경 시, **모든 행을 수정**해야 함
  → '김철수'를 '김철식'으로 바꾸려면 모든 행 수정

3. 삭제 이상 (Deletion Anomaly)

- 환자의 마지막 진료를 삭제하면
  → 환자 정보까지 사라짐

> 하나의 테이블이 모든 걸 품으면 결국 **데이터 무결성 붕괴**

제1정규형 (1NF: First Normal Form)

조건

- 모든 컬럼이 원자값(Atomic Value) 을 가져야 한다.

원자값이란?

- 하나의 컬럼에 오직 한 개의 값만 들어가는 것을 의미

잘못된 예	올바른 예
혈압 = `"138/90"`	수축기(systolic)=138, 이완기(diastolic)=90
약물명 = `"메트포르민, 아스피린"`	약물명 = `"메트포르민"` / `"아스피린"` (행 분리)

1NF 적용 후

patient_id	name	age	gender	weight	height	date	BMI	systolic	diastolic	glu
P001	김철수	45	M	72	175	2025-10-01	23.5	138	90	150
P001	김철수	45	M	72	175	2025-10-01	23.5	138	90	150
P001	김철수	45	M	72	175	2025-10-15	23.5	142	92	180

비원자값(복합 데이터) 제거, 여전히 중복 데이터(이름, 성별 등)가 존재

제2정규형 (2NF: Second Normal Form)

조건

- 1NF를 만족하고,
- 부분 종속(Partial Dependency) 을 제거해야 한다.

부분 종속이란?

- 기본키가 여러 컬럼으로 구성된 경우,
  그 중 일부 컬럼에만 종속된 속성이 존재하는 것을 의미

제2정규형 (2NF: Second Normal Form)

컬럼	종속성
name, gender, height	patient_id 에만 종속
systolic, diastolic, glucose	(patient_id, date) 에 종속

2NF 적용: 테이블 분리

환자 테이블(patients)

patient_id	name	age	gender	weight	height
P001	김철수	45	M	72	175

건강지표 테이블(health_metrics)

patient_id	date	BMI	systolic	diastolic	glucose	heart_rate
P001	2025-10-01	23.5	138	90	150	82
P001	2025-10-15	23.5	142	92	180	85

2NF 적용: 테이블 분리

약물 테이블(treatments)

patient_id	date	drug_name	dose	prescription	drug_company	company_phone
P001	2025-10-01	메트포르민	500mg	식후 2회	한빛제약	02-1234-5678
P001	2025-10-15	메트포르민	500mg	식후 2회	한빛제약	02-1234-5678

제3정규형 (3NF: Third Normal Form)

조건

- 2NF를 만족
- 이행 종속(Transitive Dependency) 을 제거

이행 종속이란?

> A → B, B → C 관계처럼
> 기본키가 아닌 컬럼이 또 다른 컬럼을 결정하는 경우

제3정규형 (3NF: Third Normal Form)

컬럼	관계
(weight, height) → BMI	BMI는 체중과 키로 계산 가능
birth_date → age	나이는 생년월일로 계산 가능
drug_name → drug_company, company_phone	약물 이름으로 제약회사 이름, 정보를 유추 가능

3NF 적용:계산 컬럼 제거

patients

patient_id	name	gender	birth_date
P001	김철수	M	1980-03-05

health_metrics

patient_id	date	weight	height	systolic	diastolic	glucose	heart_rate
P001	2025-10-01	72	175	138	90	150	82
P001	2025-10-15	72	175	142	92	180	85

3NF 적용:계산 컬럼 제거

drugs

drug_id	drug_name	drug_company	company_phone
D001	메트포르민	한빛제약	02-1234-5678

treatments

patient_id	date	drug_id	dose	prescription
P001	2025-10-01	D001	500mg	식후 2회
P001	2025-10-15	D001	500mg	식후 2회

요약

구분	정규화 전	정규화 후 (3NF)
중복 데이터	약물명, 제약사, 전화번호, BMI, 나이 등 반복	코드/계산으로 대체
함수 종속	`drug_name → drug_company`, `birth_date → age`, `(weight,height)→BMI`	비키 종속 제거
데이터 일관성	갱신 시 여러 행 수정 필요	한 곳만 수정
조회 성능	테이블 크기 큼 (중복 많음)	조인 필요하지만 무결성↑
정규화 수준	2NF 이하	3NF (이행 종속 제거)

인덱스(Index)의 필요성

문제

- 환자 1만 명, 건강 지표 100만 건이라면?
- `WHERE patient_id='P001'` 이 매우 느려짐

해결: 인덱스

- 책의 “목차(index)”처럼 **데이터 검색 속도를 빠르게** 함
- 특정 컬럼(patient_id, date 등)에 인덱스 생성

```
CREATE INDEX idx_patient_date
ON health_metrics (patient_id, date);
```

MySQL 테이블 스키마 생성

patients

```
CREATE TABLE patients (
  id          INT AUTO_INCREMENT PRIMARY KEY,
  name        VARCHAR(100) NOT NULL,
  gender      CHAR(1)      NOT NULL,
  birth_date DATE
);
```

health_metrics

```sql
CREATE TABLE health_metrics (
  id INT AUTO_INCREMENT PRIMARY KEY,
  patient_id INT NOT NULL,
  date DATE NOT NULL,
  weight DECIMAL(5,2),
  height DECIMAL(5,2),
  systolic SMALLINT,
  diastolic SMALLINT,
  glucose SMALLINT,
  heart_rate SMALLINT,
  FOREIGN KEY (patient_id) REFERENCES patients(id)
);
```

drugs

```sql
CREATE TABLE drugs (
  id              INT AUTO_INCREMENT PRIMARY KEY,
  drug_code       VARCHAR(20) UNIQUE NOT NULL,
  drug_name       VARCHAR(200) NOT NULL,
  drug_company    VARCHAR(200),
  company_phone   VARCHAR(30)
);
```

treatments

```sql
CREATE TABLE treatments (
  id INT AUTO_INCREMENT PRIMARY KEY,
  patient_id INT NOT NULL,
  drug_id INT NOT NULL,
  date DATE NOT NULL,
  dose VARCHAR(50),
  prescription VARCHAR(200),
  FOREIGN KEY (patient_id) REFERENCES patients(id),
  FOREIGN KEY (drug_id) REFERENCES drugs(id)
);
```

환자별 최신 건강 상태 보기

```sql
SELECT
    p.id AS patient_id,
    p.name,
    h.date,
    h.systolic,
    h.diastolic,
    h.glucose,
    h.heart_rate
FROM patients p
INNER JOIN health_metrics h
    ON p.id = h.patient_id
WHERE p.id = 1
ORDER BY h.date DESC
LIMIT 3;
```

환자 처방 내역 + 약물 정보 함께 조회

```sql
SELECT
    t.id AS treatment_id,
    p.name AS patient_name,
    t.date,
    d.drug_name,
    d.drug_company,
    t.dose,
    t.prescription
FROM treatments t
JOIN patients p
    ON p.id = t.patient_id
JOIN drugs d
    ON d.id = t.drug_id
ORDER BY t.date DESC;
```

성별별 평균 혈압/혈당 분석

```sql
SELECT
    p.gender,
    ROUND(AVG(h.systolic), 1) AS avg_systolic,
    ROUND(AVG(h.diastolic), 1) AS avg_diastolic,
    ROUND(AVG(h.glucose), 1) AS avg_glucose
FROM patients p
JOIN health_metrics h
    ON p.id = h.patient_id
GROUP BY p.gender;
```

약물별 처방 건수

```sql
SELECT
    d.drug_name,
    COUNT(t.id) AS n_prescriptions
FROM treatments t
JOIN drugs d
    ON d.id = t.drug_id
GROUP BY d.drug_id, d.drug_name
ORDER BY n_prescriptions DESC;
```

Python 으로도 실행해보기

- 모델 설정
- app 설정
- config 설정

Patients

```
class Patient(models.Model):
    id = fields.IntField(pk=True)
    name = fields.CharField(max_length=100, index=True)
    gender = fields.CharField(max_length=1)
    birth_date = fields.DateField(null=True)
```

Drugs

```
class Drug(models.Model):
    id = fields.IntField(pk=True)
    drug_code = fields.CharField(max_length=20, unique=True)
    drug_name = fields.CharField(max_length=200, index=True)
    drug_company = fields.CharField(max_length=200, null=True)
    company_phone = fields.CharField(max_length=30, null=True)
```

HealthMetrics

```
class HealthMetric(models.Model):
    id = fields.IntField(pk=True)
    patient = fields.ForeignKeyField("models.Patient", related_name="metrics",
                                     on_delete=fields.CASCADE)  # patient_id 컬럼 생성
    date = fields.DateField()
    weight = fields.DecimalField(max_digits=5, decimal_places=2, null=True)
    height = fields.DecimalField(max_digits=5, decimal_places=2, null=True)
    systolic = fields.SmallIntField(null=True)
    diastolic = fields.SmallIntField(null=True)
    glucose = fields.SmallIntField(null=True)
    heart_rate = fields.SmallIntField(null=True)
```

Treatments

```
class Treatment(models.Model):
    id = fields.IntField(pk=True)
    patient = fields.ForeignKeyField("models.Patient", related_name="treatments",
                                     on_delete=fields.CASCADE)  # patient_id
    drug = fields.ForeignKeyField("models.Drug", related_name="treatments",
                                  on_delete=fields.RESTRICT)   # drug_id
    date = fields.DateField()
    dose = fields.CharField(max_length=50, null=True)
    prescription = fields.CharField(max_length=200, null=True)
```

환자별 최신 건강 상태

```python
# q1_latest_metrics.py
from models import Patient, HealthMetric

async def latest_metrics(patient_id: int, limit: int = 3):
    rows = (HealthMetric
            .filter(patient_id=patient_id)
            .select_related("patient")            # JOIN patient 미리 로드
            .order_by("-date")
            .limit(limit))
    for r in await rows:
        print(r.patient.id, r.patient.name, r.date, r.systolic, r.diastolic, r.glucose, r.heart_rate)
```

처방 + 약물 정보 함께 조회

```python
# q2_treatments_join.py
from models import Treatment

async def treatments_with_drug_and_patient():
    rows = (Treatment
            .all()
            .select_related("patient", "drug")     # 두 FK 모두 JOIN
            .order_by("-date"))
    for t in await rows:
        print(t.id, t.date, t.patient.name, t.drug.drug_name, t.drug.drug_company, t.dose, t.prescription)
```

실습

- 환자별 최신 건강 데이터 N건
  목표: 한 환자의 최근 건강 측정 기록 3건 보기
- 처방 + 약물 정보 함께
  목표: 환자 처방 내역을 약물명/제약사까지 묶어서
- 위험 환자 탐지
  목표: 수축기>140 또는 혈당>200 환자 최신 기록

challenge!

- 건강 기록이 없어도 환자 목록에 나오게
  목표: 최근 30일 내 측정이 없는 환자도 “NULL”로 보이게