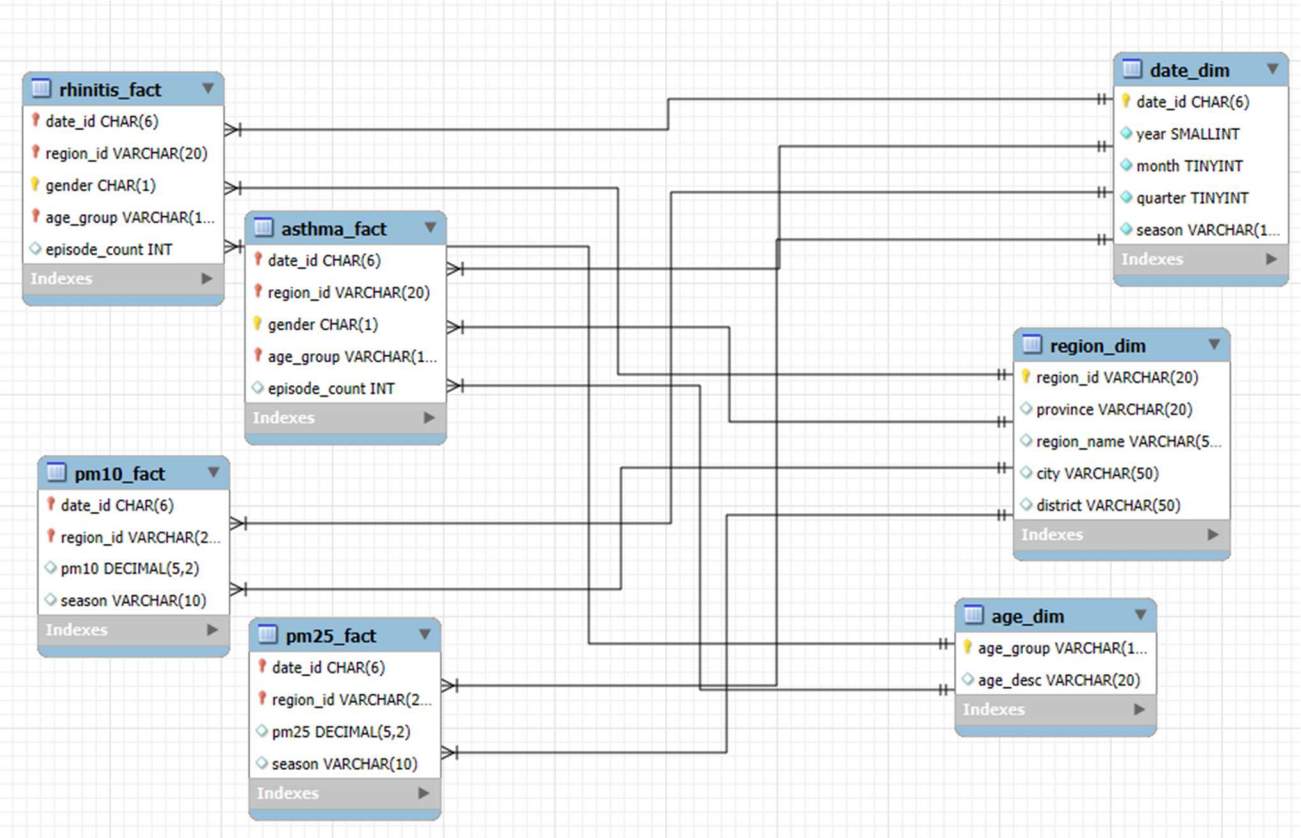
1. ERD 전체 구조 요약



주요 관계(Constraints)

1. pm10_fact.date_id → date_dim.date_id (FK)
2. pm25_fact.date_id → date_dim.date_id (FK)
3. pm10_asthma_fact.ym → (사실상) date_dim.date_id ("YYYYMM" 형태로 매핑)
4. pm25_asthma_fact.ym → date_dim.date_id (FK)
5. pm10_rhinitis_fact.ym → date_dim.date_id (FK)
6. pm25_rhinitis_fact.ym → date_dim.date_id (FK)
7. 네 개의 "천식/비염" 페이크트 테이블(pm10_asthma_fact, pm25_asthma_fact, pm10_rhinitis_fact, pm25_rhinitis_fact) 의 age_group 컬럼 → age_group_dim.age_group (FK)
8. 모든 사실(fact) 테이블의 region(또는 region name) 컬럼 → region_dim.region_id (FK)

2. 테이블별 정의

1) date_dim (날짜 차원 테이블)

- **목적**: "YYYYMM" 형태의 연·월 정보를 분리하여, 분기나 계절 같은 추가적인 날짜 속성을 제공
- **컬럼**
    1. date_id CHAR(6) – PK
        - 예: "200601", "200602", ..., "202410"
    2. year SMALLINT – 연도 (예: 2006, 2007, ...)
    3. month TINYINT – 월 (1~12)
    4. quarter TINYINT – 분기 (1~4)
    5. season VARCHAR(10) – 계절 (예: "winter", "spring", "summer", "autumn")
- **제약**
    - PRIMARY KEY (date_id)
    - date_id 값은 중복 불가
    - year : NOT NULL
    - month : NOT NULL
    - quarter : NOT NULL
    - season : NOT NULL

---

2) region_dim (지역 차원 테이블)

- **목적**: "시도" 단위 지역 정보를 관리한다. 사실 테이블에서는 region_id 로 FK 연결
- **컬럼**
    1. region_id VARCHAR(20) – PK
        - 예: "서울특별시", "부산광역시", "대구광역시" ... "제주특별자치도"
        - (만약 내부 코드(숫자식별자)가 있으면, 그 값을 region_id 로 두어야 함)
    2. province VARCHAR(20) – (예시) 대분류 명칭, 예: "서울특별시"
    3. region_name VARCHAR(50) – (예시) "서울 특별시 강남구"처럼 세부 명칭이 필요할 때 사용할 수 있음

4. city VARCHAR(50) – (예시) "서울" 등 시급 단위 정보
5. district VARCHAR(50) – (예: "강남구", "중구" 등의 구단위 정보

- **제약**
  - PRIMARY KEY (region_id)
  - 모든 컬럼 중 region_id 가 유일해야 함.
  - 나머지 컬럼은 상황에 따라 NULL 허용(필요하다면 NOT NULL 제약 추가 가능)

---

3) age_group_dim (연령대 차원 테이블)

- **목적**: "천식/비염" 환자 테이블에서 사용되는 연령대 코드를 관리
- **컬럼**
  1. age_group VARCHAR(10) – PK
     - 예: "0–5 세", "6–17 세", "18–34 세", "35–49 세", "50–64 세", "65+세" 등
  2. age_range VARCHAR(20) – 해당 연령대의 구체적인 범위(예: "0–5 세" → "0~5 세")
- **제약**
  - PRIMARY KEY (age_group)
  - age_group 값 중복 불가
  - age_range 는 NULL 허용 가능(필요 시 NOT NULL 로 설정)

---

4) pm10_fact (PM10 농도 사실 테이블)

- **목적**: "시도별·월별 PM10 평균 농도"를 저장
- **컬럼**
  1. id INT AUTO_INCREMENT – PK
  2. date_id CHAR(6) – FK → date_dim.date_id
  3. region_name VARCHAR(50) – FK → region_dim.region_id (실제로는 region_id 라는 이름으로 같음)
  4. pm10 DECIMAL(5,2) – 해당 월·해당 시도의 PM10 평균 농도(㎍/㎥)
  5. season VARCHAR(10) – (중복) 계절 정보(예: "winter", "spring" 등)

    - 참고: 사실 테이블에서 season 을 굳이 중복 저장할 필요 없이, date_dim.season 을 JOIN 해서 사용할 수 있다.
      → "정규화" 관점에서는 중복 저장을 피하는 편이 좋다.
- **제약**
    - PRIMARY KEY (id)
    - FOREIGN KEY (date_id) REFERENCES date_dim(date_id)
    - FOREIGN KEY (region_name) REFERENCES region_dim(region_id)
    - pm10: NULL 허용 여부는 사용 상황에 따라 결정. 실제로 전처리된 데이터에는 NULL 이 없도록 가공되어 있어야 함.

---

5) pm25_fact (PM2.5 농도 사실 테이블)

- **목적**: "시도별·월별 PM2.5 평균 농도"를 저장
- **컬럼**
    1. id INT AUTO_INCREMENT – PK
    2. date_id CHAR(6) – FK → date_dim.date_id
    3. region_name VARCHAR(50) – FK → region_dim.region_id
    4. pm25 DECIMAL(5,2) – 해당 월·해당 시도의 PM2.5 평균 농도(㎍/㎥)
    5. season VARCHAR(10) – (중복) 계절 정보
- **제약**
    - PRIMARY KEY (id)
    - FOREIGN KEY (date_id) REFERENCES date_dim(date_id)
    - FOREIGN KEY (region_name) REFERENCES region_dim(region_id)

---

6) pm10_asthma_fact (천식 + PM10 사실 테이블)

- **목적**: "시도별·월별·성별·연령대별 천식 외래 환자 수"를 저장
- **컬럼**
    1. id INT AUTO_INCREMENT – PK
    2. ym CHAR(6) – "YYYYMM" 형태의 연월. FK 로 보려면 date_dim.date_id 로 매핑 가능

3. region VARCHAR(50) – 시도명. FK → region_dim.region_id
4. gender VARCHAR(10) – 성별 (예: "남", "여")
5. age_group VARCHAR(10) – 연령대 (예: "0–5 세", "65+세" 등). FK → age_group_dim.age_group
6. visit_count INT – 해당 월·시도·성별·연령대별 천식 외래 방문자 수

- **제약**
  - PRIMARY KEY (id)
  - FOREIGN KEY (ym) REFERENCES date_dim(date_id) 〈또는 ym 을 직접 date_id 로 두면 FK 설정 가능〉
  - FOREIGN KEY (region) REFERENCES region_dim(region_id)
  - FOREIGN KEY (age_group) REFERENCES age_group_dim(age_group)
  - gender 컬럼은 단순 분류용이므로 제약 없이 NULL 허용 가능(필요 시 NOT NULL)

---

7) pm10_rhinitis_fact (비염 + PM10 사실 테이블)

- **목적**: "시도별·월별·성별·연령대별 비염 외래 환자 수"를 저장
- **컬럼**
  1. id INT AUTO_INCREMENT – PK
  2. ym CHAR(6) – "YYYYMM" 형태. FK → date_dim.date_id
  3. region VARCHAR(50) – 시도명. FK → region_dim.region_id
  4. gender VARCHAR(10) – 성별 (예: "남", "여")
  5. age_group VARCHAR(10) – 연령대. FK → age_group_dim.age_group
  6. visit_count INT – 해당 월·시도·성별·연령대별 비염 외래 방문자 수
- **제약**
  - PRIMARY KEY (id)
  - FOREIGN KEY (ym) REFERENCES date_dim(date_id)
  - FOREIGN KEY (region) REFERENCES region_dim(region_id)
  - FOREIGN KEY (age_group) REFERENCES age_group_dim(age_group)

---

8) pm25_asthma_fact (천식 + PM2.5 사실 테이블)

- **목적**: "시도별·월별·성별·연령대별 천식 외래 환자 수" (PM2.5 기준)
- **컬럼 구조 및 제약은 pm10_asthma_fact와 동일**
    1. id INT AUTO_INCREMENT – PK
    2. ym CHAR(6) – FK → date_dim.date_id
    3. region VARCHAR(50) – FK → region_dim.region_id
    4. gender VARCHAR(10)
    5. age_group VARCHAR(10) – FK → age_group_dim.age_group
    6. visit_count INT

---

9) pm25_rhinitis_fact (비염 + PM2.5 사실 테이블)

- **목적**: "시도별·월별·성별·연령대별 비염 외래 환자 수" (PM2.5 기준)
- **컬럼 구조 및 제약은 pm10_rhinitis_fact와 동일**
    1. id INT AUTO_INCREMENT – PK
    2. ym CHAR(6) – FK → date_dim.date_id
    3. region VARCHAR(50) – FK → region_dim.region_id
    4. gender VARCHAR(10)
    5. age_group VARCHAR(10) – FK → age_group_dim.age_group
    6. visit_count INT

3. python

3.1 테이블 생성

```
# date_dim
  """
  CREATE TABLE IF NOT EXISTS date_dim (
      date_id CHAR(6) PRIMARY KEY,
      year SMALLINT NOT NULL,
      month TINYINT NOT NULL,
      quarter TINYINT NOT NULL,
      season VARCHAR(10) NOT NULL
  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

```
    """,

    # age_group_dim
    """
    CREATE TABLE IF NOT EXISTS age_group_dim (
        age_group VARCHAR(10) PRIMARY KEY,
        age_range VARCHAR(20)
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
    """,

    # pm10_asthma_fact (천식+PM10)
    """
    CREATE TABLE IF NOT EXISTS pm10_asthma_fact (
        id INT AUTO_INCREMENT PRIMARY KEY,
        ym CHAR(6),
        region VARCHAR(50),
        gender VARCHAR(10),
        age_group VARCHAR(10),
        visit_count INT
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
    """,

    # pm10_rhinitis_fact (비염+PM10)
    """
    CREATE TABLE IF NOT EXISTS pm10_rhinitis_fact (
        id INT AUTO_INCREMENT PRIMARY KEY,
        ym CHAR(6),
        region VARCHAR(50),
        gender VARCHAR(10),
        age_group VARCHAR(10),
        visit_count INT
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
    """,

    # pm10_fact (PM10 농도)
    """
    CREATE TABLE IF NOT EXISTS pm10_fact (
        id INT AUTO_INCREMENT PRIMARY KEY,
        date_id CHAR(6),
        region_name VARCHAR(50),
        pm10 DECIMAL(5,2),
        FOREIGN KEY(date_id) REFERENCES date_dim(date_id)
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
    """,

    # pm25_asthma_fact (천식+PM2.5)
    """
    CREATE TABLE IF NOT EXISTS pm25_asthma_fact (
```

```
        id INT AUTO_INCREMENT PRIMARY KEY,
        ym CHAR(6),
        region VARCHAR(50),
        gender VARCHAR(10),
        age_group VARCHAR(10),
        visit_count INT
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
    """,

    # pm25_rhinitis_fact (비염+PM2.5)
    """
    CREATE TABLE IF NOT EXISTS pm25_rhinitis_fact (
        id INT AUTO_INCREMENT PRIMARY KEY,
        ym CHAR(6),
        region VARCHAR(50),
        gender VARCHAR(10),
        age_group VARCHAR(10),
        visit_count INT
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
    """,

    # pm25_fact (PM2.5 농도)
    """
    CREATE TABLE IF NOT EXISTS pm25_fact (
        id INT AUTO_INCREMENT PRIMARY KEY,
        date_id CHAR(6),
        region_name VARCHAR(50),
        pm25 DECIMAL(5,2),
        FOREIGN KEY(date_id) REFERENCES date_dim(date_id)
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
    """
```

3.2 데이터 삽입

```
# date_dim 삽입 (date_df_map 에는
['year_month','date_id','year','month','quarter','season'] 포함)
date_df_map["date_id"] = date_df_map["date_id"].astype(str)
with engine.begin() as conn:
    conn.execute(text("DELETE FROM date_dim;"))
# dtype 인자 제거: MySQL 이 자동으로 타입을 생성하도록 함
date_df_map.to_sql(
    "date_dim",
    engine,
    if_exists="append",
    index=False
)
```

```python
# age_group_dim 삽입
age_group_df.columns = ["age_group", "age_range"]
with engine.begin() as conn:
    conn.execute(text("DELETE FROM age_group_dim;"))
age_group_df.to_sql(
    "age_group_dim",
    engine,
    if_exists="append",
    index=False
)

# pm10_fact (wide → long)
pm10_long = pm10_df.melt(
    id_vars="year_month", var_name="region_name", value_name="pm10"
)
pm10_long["date_id"] = pm10_long["year_month"].str.replace("-", "")
pm10_long = pm10_long[["date_id", "region_name", "pm10"]].dropna()
with engine.begin() as conn:
    conn.execute(text("DELETE FROM pm10_fact;"))
pm10_long.to_sql(
    "pm10_fact",
    engine,
    if_exists="append",
    index=False
)

# pm25_fact (wide → long)
pm25_long = pm25_df.melt(
    id_vars="year_month", var_name="region_name", value_name="pm25"
)
pm25_long["date_id"] = pm25_long["year_month"].str.replace("-", "")
pm25_long = pm25_long[["date_id", "region_name", "pm25"]].dropna()
with engine.begin() as conn:
    conn.execute(text("DELETE FROM pm25_fact;"))
pm25_long.to_sql(
    "pm25_fact",
    engine,
    if_exists="append",
    index=False
)

# pm10_asthma_fact 삽입 (ym: 'YYYYMM' 형식으로 변경)
pm10_asthma_df = pm10_asthma_df.rename(columns={"year_month": "ym"})
pm10_asthma_df["ym"] = pm10_asthma_df["ym"].str.replace("-", "")
with engine.begin() as conn:
    conn.execute(text("DELETE FROM pm10_asthma_fact;"))
pm10_asthma_df.to_sql(
    "pm10_asthma_fact",
```

```
    engine,
    if_exists="append",
    index=False
)

# pm25_asthma_fact 삽입
pm25_asthma_df = pm25_asthma_df.rename(columns={"year_month": "ym"})
pm25_asthma_df["ym"] = pm25_asthma_df["ym"].str.replace("-", "")
with engine.begin() as conn:
    conn.execute(text("DELETE FROM pm25_asthma_fact;"))
pm25_asthma_df.to_sql(
    "pm25_asthma_fact",
    engine,
    if_exists="append",
    index=False
)

# pm10_rhinitis_fact 삽입
pm10_rhinitis_df = pm10_rhinitis_df.rename(columns={"year_month": "ym"})
pm10_rhinitis_df["ym"] = pm10_rhinitis_df["ym"].str.replace("-", "")
with engine.begin() as conn:
    conn.execute(text("DELETE FROM pm10_rhinitis_fact;"))
pm10_rhinitis_df.to_sql(
    "pm10_rhinitis_fact",
    engine,
    if_exists="append",
    index=False
)

# pm25_rhinitis_fact 삽입
pm25_rhinitis_df = pm25_rhinitis_df.rename(columns={"year_month": "ym"})
pm25_rhinitis_df["ym"] = pm25_rhinitis_df["ym"].str.replace("-", "")
with engine.begin() as conn:
    conn.execute(text("DELETE FROM pm25_rhinitis_fact;"))
pm25_rhinitis_df.to_sql(
    "pm25_rhinitis_fact",
    engine,
    if_exists="append",
    index=False
)
```