**데이터 파이프라인 및 DB 스키마 설계서**

**1) 전체 데이터 파이프라인 (E2E)**

**1.1 핵심 전제(문서 근거 요약)**

* **데이터 원천**: 서울시 ‘실시간 도시데이터’(도시 통합/인구/상권/교통/환경/문화행사 등), 장소 단위(120 POI), 5~15분 단위 갱신·단일 장소 호출 패턴. 샘플키는 ‘광화문·덕수궁’만 허용(운영키에서 코드 호출 권장)
* **아키텍처 스캐폴드**: ECS Fargate(x86\_64), **RDS(PostgreSQL16+PostGIS)**, **ElastiCache Redis7**, ECR/ALB/ACM/Route53, GitHub Actions CI/CD로 무중단 배포 · 시크릿은 Secrets Manager 주입 권장
* **학습/일정 계획**: 9~12월 단계별로 요구사항→데이터 수집·파이프라인·DB 스키마→모델링/서비스화 순으로 진행

**1.2 레이어별 설계**

**(A) 수집/호출(Ingestion)**

* **수집 대상**: 120 POI 코드(예: POI104=어린이대공원), 통합 citydata + 필요 시 citydata\_ppltn 등 서브셋. **1회 1장소** 원칙(필수).
* **호출 주기**: 5분 라운드(권장). **rate-limit**: 0.7~1.0초/호출(120개 ≈ 2~3분/라운드, 5분 SLA 충족).
* **실행체계**:
  + **옵션1(간단·즉시)**: ECS Fargate **Scheduled Task** 2~3개 워커(코드 라운드 로빈).
  + **옵션2(확장)**: EventBridge → SQS(Work Queue) → Fargate Consumer N개(오토스케일).
* **신뢰성**: RESULT.CODE 파싱(정상 INFO-000)·재시도·서킷브레이커(5xx/타임아웃), 요청-응답 로그 표준화. **샘플/운영키 분기**(샘플=장소명만) 포함.
* **참조 스크립트**: 제공된 Bash/Python 일괄수집 스크립트(Postman 컬렉션 포함)는 즉시 사용 가능(경로/키만 세팅).

**(B) 적재(Staging → Curated)**

* **원본 보존**: raw.citydata에 **원문 JSONB** + 메타(요청시간/소스/HTTP/RESULT) **WORM 방식** 30일 보관.
* **표준화(정규화)**: 카테고리별 **스키마 매핑**(인구/상권/교통/대중교통/환경/문화/주차/전기차충전/따릉이 등) → **주기별 스냅샷 팩트**로 **정규화 적재**.
* **업서트/멱등성**: (poi\_code, observed\_at, category)를 **자연키**로 **UPSERT**. 업스트림 지연/중복에 안전.
* **품질검사**: 필드 유효성(타입/범위), 변동 스키마는 extras JSONB로 흡수, 파서 버전 태깅.

**(C) 가공/집계(Serving/Mart)**

* **뷰·MV**: latest\_snapshot\_per\_poi, trend\_12h, top\_changing\_poi(증감률), 혼잡도/상권 단계 산출 MV.
* **예측결과 저장(선택)**: ml\_population\_forecast\_5m(향후 12시간 예측; 모델 버전/학습일자 태깅). 매뉴얼의 혼잡도·예측 개념 반영(과거 평균·밀집도, 5분 단위)
* **API/RECO 제공**: API(Spring) 프록시에서 **표준화 응답** 제공, RECO(FastAPI)는 Redis 캐시 + DB 최신 스냅샷 결합(콜드 시 원본 프록시 호출).

**(D) 캐시/레이트/장애 내성**

* **Redis**: 60~120초 TTL 스냅샷/추천결과 캐시(키: citydata:POI104:ts / reco:nearby:<lat>:<lng>:purpose 등). 네임스페이스·TTL 규칙 권고.
* **백오프/Fail-Open**: 서울시 API 실패 시 최근 스냅샷 + “데이터 시각/출처” 명시 제공.
* **비용/성능**: S3/CloudWatch VPC 엔드포인트로 NAT 비용 절감(선택). ECS 오토스케일: CPU 60% or P95 레이턴시 기준.

**2) 논리 데이터 모델(요약 ERD)**

[core.poi] 1 ────< [fact\_\*\_5m] >──── 1 [dim.category]

\ \

\ > [fact\_events\_daily]

>──── [dim.time\_5m]

**2.1 기준/공통 테이블**

* **core.poi**(PK: code)
  + code(PK, text), name(ko), type(enum: palace/special/commercial/park/station…), ward/gu,  
    centroid(point, geography), boundary(geometry, Polygon/MultiPolygon, SRID 4326), area\_m2(numeric), created\_at
  + 비고: **PostGIS** 필수. 120개 장소 메타 관리(공식 엑셀 원본 참조).
* **dim.category**(PK: code)
  + code(enum): population, commerce, road, transit, parking, ev\_charger, bike, weather, events
  + description
* **dim.time\_5m**(PK: ts\_utc)
  + ts\_utc(timestamptz), ts\_kst(timestamptz), yyyymmdd, dow, hour, minute, is\_weekend 등
* **raw.citydata**(PK: id bigserial)
  + poi\_code, fetched\_at, observed\_at, category, http\_status, result\_code, result\_message, payload(jsonb), hash(dedup), parser\_version, source(‘Seoul citydata’), inserted\_at
  + **인덱스**: (poi\_code, observed\_at, category), GIN(payload)

**2.2 팩트(스냅샷) 테이블 — 5분/10분 단위**

※ 실제 필드는 매뉴얼 응답 필드군을 정규화해 설계. 일부 복잡 항목은 extras JSONB로 유연성 확보.

* **fact\_population\_5m** (혼잡도/성·연령 분포 포함)
  + **PK**: (poi\_code, observed\_at)
  + poi\_code FK, observed\_at timestamptz, total bigint, congestion(enum: easy/normal/busy/peak),  
    male\_pct, female\_pct, age\_10s\_pct … age\_60s\_plus\_pct, area\_density\_level(enum), extras jsonb
  + **인덱스**: (observed\_at), (poi\_code, observed\_at desc)
* **fact\_commerce\_10m** (결제 금액/건수 + 업종별 breakdown)
  + PK(poi\_code, observed\_at), total\_amount bigint, total\_txn bigint,  
    bucket(enum: quiet/normal/busy/peak), **sector\_breakdown jsonb**(15대 업종 금액·건수 맵)
* **fact\_road\_5m** (도로소통/사고·돌발 요약)
  + PK(poi\_code, observed\_at), avg\_speed\_kmh, congestion\_index int, incidents\_cnt int, extras jsonb
  + (사고/돌발 상세는 별도 테이블 road\_incident\_5m로 풀어 저장 가능)
* **fact\_transit\_5m** (버스·지하철 도착/승하차 요약)
  + PK, bus\_arrivals\_cnt, subway\_arrivals\_cnt, onboard\_cnt, offboard\_cnt, extras jsonb
* **fact\_parking\_5m** (주차장 가용)
  + PK, lots\_total, lots\_available, utilization\_pct
* **fact\_ev\_5m** (전기차 충전소)
  + PK, chargers\_total, chargers\_available, type\_breakdown jsonb
* **fact\_bike\_5m** (따릉이)
  + PK, docks\_total, bikes\_available, rack\_utilization\_pct
* **fact\_weather\_10m**
  + PK, temp\_c, humidity\_pct, wind\_ms, rainfall\_mm, pm10, pm2\_5, uv\_index, alert\_level, extras
* **fact\_events\_daily**
  + **PK**: (poi\_code, event\_id)
  + title, start\_date, end\_date, venue, url, category, last\_seen\_at, extras

**2.3 파생/서빙 계층**

* **mv\_latest\_snapshot\_per\_poi** (MV): 각 카테고리 최신 1레코드
* **mv\_trend\_12h\_population / mv\_trend\_12h\_commerce**: 12시간 이동 시계열
* **ml\_population\_forecast\_5m**: poi\_code, target\_ts, pred\_population, model\_version, mae, created\_at

**3) 물리 스키마/DDL 초안 (PostgreSQL 16 + PostGIS)**

**네임스페이스**: core(기준), raw(원본), fact(정규화), mart(서빙/MV), ml(예측).  
**파티셔닝**: 대량 테이블(fact\_\*)은 **날짜 파티션(일 단위)**   
**인덱스**: (poi\_code, observed\_at) 필수, 시계열 역순 탐색/범위 조회 최적화.

-- 확장

CREATE EXTENSION IF NOT EXISTS postgis;

-- 1) 기준

CREATE SCHEMA IF NOT EXISTS core;

CREATE TABLE core.poi (

code TEXT PRIMARY KEY,

name\_ko TEXT NOT NULL,

type TEXT NOT NULL, -- ENUM 대체 가능

ward TEXT,

centroid GEOGRAPHY(POINT, 4326),

boundary GEOMETRY(MULTIPOLYGON, 4326),

area\_m2 NUMERIC,

created\_at TIMESTAMPTZ DEFAULT now()

);

CREATE INDEX ON core.poi USING GIST(boundary);

CREATE SCHEMA IF NOT EXISTS raw;

CREATE TABLE raw.citydata (

id BIGSERIAL PRIMARY KEY,

poi\_code TEXT NOT NULL REFERENCES core.poi(code),

category TEXT NOT NULL,

fetched\_at TIMESTAMPTZ NOT NULL,

observed\_at TIMESTAMPTZ NOT NULL,

http\_status INT,

result\_code TEXT,

result\_message TEXT,

payload JSONB NOT NULL,

hash TEXT, -- 중복 방지

parser\_version TEXT,

source TEXT,

inserted\_at TIMESTAMPTZ DEFAULT now()

);

CREATE INDEX ON raw.citydata(poi\_code, observed\_at, category);

CREATE INDEX ON raw.citydata USING GIN(payload);

-- 2) 팩트(예: 인구)

CREATE SCHEMA IF NOT EXISTS fact;

CREATE TABLE fact.population\_5m (

poi\_code TEXT NOT NULL REFERENCES core.poi(code),

observed\_at TIMESTAMPTZ NOT NULL,

total BIGINT,

congestion TEXT, -- easy/normal/busy/peak

male\_pct NUMERIC, female\_pct NUMERIC,

age\_10s\_pct NUMERIC, age\_20s\_pct NUMERIC, age\_30s\_pct NUMERIC,

age\_40s\_pct NUMERIC, age\_50s\_pct NUMERIC, age\_60p\_pct NUMERIC,

density\_lvl TEXT, -- 면적대비 밀집도 단계

extras JSONB,

PRIMARY KEY(poi\_code, observed\_at)

) PARTITION BY RANGE (observed\_at);

-- 일 단위 파티션 예시

CREATE TABLE fact.population\_5m\_2025\_09\_20

PARTITION OF fact.population\_5m FOR VALUES FROM ('2025-09-20') TO ('2025-09-21');

-- 상권(10분)

CREATE TABLE fact.commerce\_10m (

poi\_code TEXT NOT NULL REFERENCES core.poi(code),

observed\_at TIMESTAMPTZ NOT NULL,

total\_amount BIGINT,

total\_txn BIGINT,

bucket TEXT, -- quiet/normal/busy/peak

sectors JSONB, -- 업종별 금액/건수 맵

extras JSONB,

PRIMARY KEY(poi\_code, observed\_at)

) PARTITION BY RANGE (observed\_at);

-- 도로/교통/환경/문화행사 등은 동일 패턴으로 생성

-- ...

-- 3) 서빙/예측

CREATE SCHEMA IF NOT EXISTS mart;

-- 최신 스냅샷 MV

CREATE MATERIALIZED VIEW mart.latest\_population AS

SELECT DISTINCT ON (p.poi\_code)

p.poi\_code, p.observed\_at, p.total, p.congestion, p.extras

FROM fact.population\_5m p

ORDER BY p.poi\_code, p.observed\_at DESC;

CREATE INDEX ON mart.latest\_population(poi\_code);

CREATE SCHEMA IF NOT EXISTS ml;

CREATE TABLE ml.population\_forecast\_5m (

poi\_code TEXT NOT NULL REFERENCES core.poi(code),

target\_ts TIMESTAMPTZ NOT NULL,

pred\_total BIGINT NOT NULL,

model\_ver TEXT NOT NULL,

train\_date DATE,

mae NUMERIC,

created\_at TIMESTAMPTZ DEFAULT now(),

PRIMARY KEY (poi\_code, target\_ts, model\_ver)

);

**4) ETL(파서) 매핑 규칙**

**정석**: “원본(JSONB) ⇒ 정규화 팩트(스냅샷) + extras 보조”  
매뉴얼의 **카테고리/항목**(인구 25개, 상권 24개, 도로/대중교통/환경/문화행사 등)을 표준 키로 매핑. **필드 추가/변경**은 extras에 흡수 후, 릴리즈로 정규 필드 승격.

* **인구(population)**: total, 성·연령 분포, **혼잡도 단계**(과거 28일 대비 + 면적 밀집도 보정). 면적 보정은 core.poi.area\_m2 또는 boundary 기반 계산으로 일관성 확보.
* **상권(commerce)**: total\_amount/txn, **소비 단계(bucket)**(4주 동요일/동시간대 평균 대비), 업종 15분류 breakdown(JSONB).
* **교통(road/transit)**: 평균 속도/정체도, 돌발/사고 건, 버스·지하철 도착·승하차(지연 시간대 예외 처리).
* **환경(weather)**: 기온/강수/미세먼지/특보 등. **관측소 기준**이면 POI와 **최단거리 매칭**(PostGIS KNN) 가능.
* **문화행사(events)**: 일 단위 업데이트 → fact\_events\_daily 반영.

**5) 접근 패턴/인덱스/성능**

* **핫 쿼리**
  + 최신 값: SELECT \* FROM mart.latest\_population WHERE poi\_code=?
  + 기간 추이: SELECT ... FROM fact.population\_5m WHERE poi\_code=? AND observed\_at >= now()-interval '12 hours' ORDER BY observed\_at;
  + 랭킹/증감: GROUP BY poi\_code ORDER BY (last.total - avg\_24h.total) DESC LIMIT 10;
* **인덱스**
  + 모든 팩트: (poi\_code, observed\_at DESC) 커버링 인덱스 + 파티션 프루닝.
  + 공간 탐색(주변 추천): core.poi.boundary/centroid에 **GIST**.
* **보존/아카이브**
  + raw 30일, fact 365일(요약 테이블로 집계 후 구간별 파티션 드롭), 로그 30일(스캐폴드 권고).

**6) 운영·보안·거버넌스**

* **시크릿/권한**: 앱 Role ↔ Secrets Manager 최소 권한(실서비스는 리소스 범위 제한), Execution/App Role 분리.
* **CI/CD**: deploy-api.yml, deploy-reco.yml로 **ECR 빌드/푸시 → ECS 무중단 업데이트**(이미지 태그 latest + SHA, 서비스는 **SHA 고정** 배포).
* **SLO**: p80 ≤ 500ms, p95 ≤ 800ms, 에러율 < 1%, 캐시 히트율 ≥ 70%. 알람: ALB 5xx, TG Unhealthy, ECS 자원 사용량, 캐시 히트율.
* **출처 고지**: 공공누리 제1유형(출처표시) 문구를 응답/화면에 **항상 노출**. 프록시에서 자동 삽입 권장.

**7) RECO 연동 설계(요점)**

* **데이터 소스**: (1) DB 최신 스냅샷(MV/팩트) + (2) API 프록시 라이브 콜(캐시 미스 시) 하이브리드.
* **피처 예시**: 현재 혼잡도/증감률, 업종별 소비 활성도, 대중교통 승하차 지수, 날씨/특보, 시간대(야간/주말) 가중치.
* **캐시 키**: reco:nearby:<lat>:<lng>:<purpose>:<time\_bucket>, TTL 1~5분. **Rate-limit**는 .env에서(이미 운영 적용).
* **엔드포인트**: POST /reco/v1/recommend(현재 구현 완료 상태에서 DB 피처 추가만 하면 됨).

**8) 단계별 실행 체크리스트**

1. **DB 스키마 적용**: 위 DDL → Flyway/Liquibase 마이그레이션 생성·배포.
2. **수집 워커 배치**:
   * 초안: Fargate Scheduled Task 2개(POI 분할), 라운드 5분.
   * 실패/재시도/로그: RESULT.CODE, HTTP, 소요시간 기록.
3. **파서 구현**: 카테고리별 매핑 + UPSERT. extras로 필드변경 흡수.
4. **MV/뷰 생성**: 최신 스냅샷/트렌드/랭킹.
5. **API 통합**: Spring 프록시 → DB 경유 응답(선캐시) + 원본 콜 폴백.
6. **RECO 피처 조립**: DB 스냅샷 피처 결합, Redis 캐시, A/B 파라미터 노출.
7. **모니터링/알람**: 지표·로그·대시보드 구성(CloudWatch + RDS Performance Insights).
8. **보안/컴플라이언스**: 출처표기, 키 비공개, CORS 최소 허용, TLS 강제, WAF(봇/SQLi/XSS) 권장.

**부록 A. 설계 근거**

* **단일 장소/단일 호출** 제약과 **5분 단위** 갱신 주기를 **라운드 로빙 수집** 설계로 충족(쉬운 수평 확장).
* **혼잡도/상권 단계**는 **과거 평균 대비 + 면적/밀집 보정** 정의를 충실히 재현하도록 **정규화 팩트 + MV** 이원화.
* **PostGIS** 도입으로 **POI 경계/면적 기반 지표**(밀집도, 주변 추천, 근접 매칭) 제공 가능. 스캐폴드가 RDS/PostGIS를 권장.
* **Redis**는 캐시/버스트 제어를 공식 문서 템플릿 수준에서 권고(네임스페이스/TTL 규칙).
* **프록시/API 표준화**는 키 보호·로깅·캐시·업무에러 승격(RESULT→HTTP) 등 안정운영을 위해 필수.

**부록 B. 운영 규정(요약)**

* **라운드 주기**: 5분(기본), 비정상 다발 시 10분.
* **재시도**: 429/5xx/타임아웃 시 지수백오프(0.8, 최대3).
* **유지보수**: 신규 필드는 extras에 먼저 적재→월간 릴리즈로 정규 필드 승격.
* **보존**: raw 30일, fact 12개월, 월 집계 36개월.
* **릴리즈 관리**: parser\_version, schema\_version 커밋-SHA 태깅.