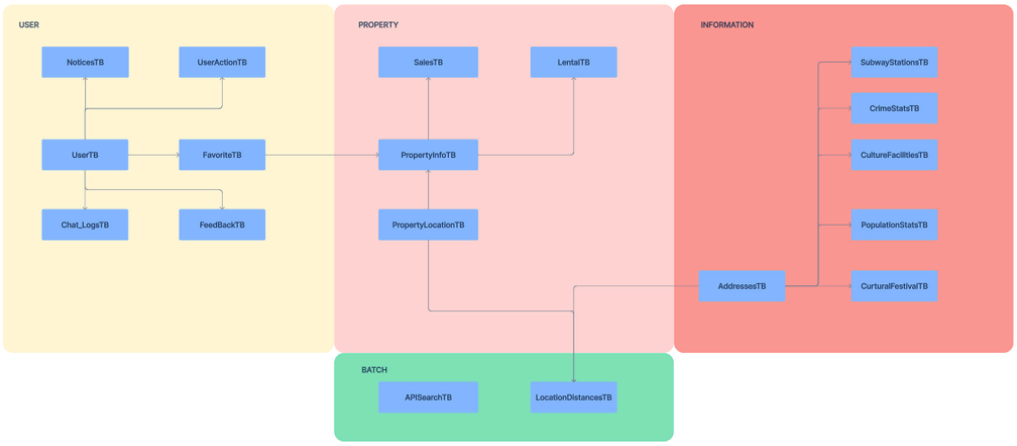
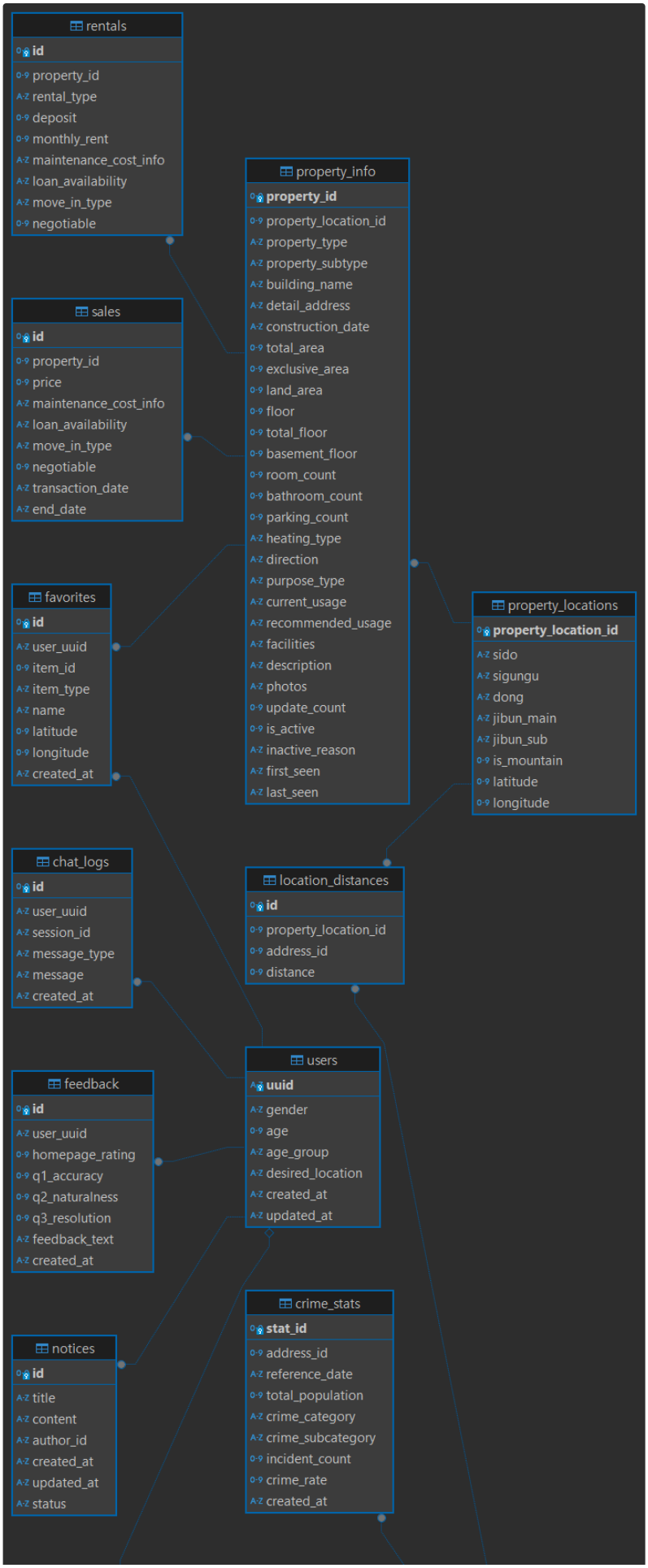
데이터베이스 설계 문서

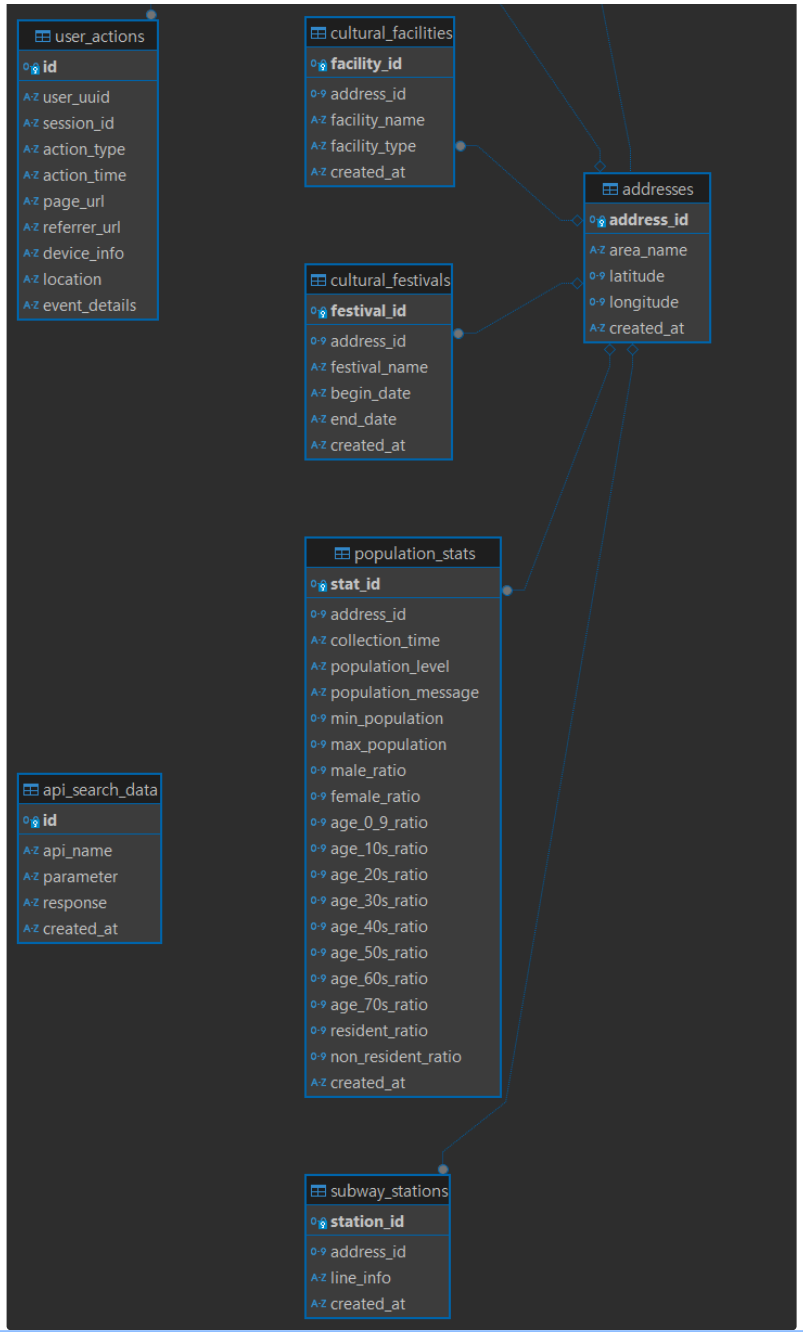
데이터베이스의 개념적 설계와 논리적 설계

- **개념적 설계**
  서집사의 개념적 설계는 요구사항 정의서에 의해 작성되었다.



- **논리적 설계**
  팀원들과 상의를 한 요구사항 정의서를 가지고 'ERD 클라우드'를 사용하여 논리적 설계를 진행하였다.
  SQLite 성능 최적화를 위해서 5가지 PRAGMA를 설정하였다
  - Write-Ahead Logging 모드
    변경 사항을 실제 데이터베이스 파일에 적용하기 전에 로그 파일에 기록하는 방식
  - 동기화 수준 조정
    성능과 데이터 무결성 간의 균형을 맞춤.
  - 캐시 크기 증가
    데이터베이스 성능이 향상되지만, 시스템의 메모리 자원을 더 많이 사용
  - 임시 저장소를 메모리로
    임시 데이터를 디스크가 아닌 빠른 메모리에 저장하고 접근
  - 메모리 매핑 크기 증가
    디스크에서 데이터를 읽을 때 발생하는 I/O가 줄어들어 성능을 최적화
- **데이터베이스의 물리적 설계**


rentals
id
property_id
rental_type
deposit
monthly_rent
maintenance_cost_info
loan_availability
move_in_type
negotiable
property_info
property_id
property_location_id
property_type
property_subtype
building_name
detail_address
construction_date
total_area
exclusive_area
land_area
floor
total_floor
basement_floor
room_count
bathroom_count
parking_count
heating_type
direction
purpose_type
current_usage
recommended_usage
facilities
description
photos
update_count
is_active
inactive_reason
first_seen
last_seen
sales
id
property_id
price
maintenance_cost_info
loan_availability
move_in_type
negotiable
transaction_date
end_date
favorites
id
user_uuid
item_id
item_type
name
latitude
longitude
created_at
property_locations
property_location_id
sido
sigungu
dong
jibun_main
jibun_sub
is_mountain
latitude
longitude
chat_logs
id
user_uuid
session_id
message_type
message
created_at
location_distances
id
property_location_id
address_id
distance
users
uuid
gender
age
age_group
desired_location
created_at
updated_at
feedback
id
user_uuid
homepage_rating
q1_accuracy
q2_naturalness
q3_resolution
feedback_text
created_at
crime_stats
stat_id
address_id
reference_date
total_population
crime_category
crime_subcategory
incident_count
crime_rate
created_at
notices
id
title
content
author_id
created_at
updated_at
status



데이터베이스 모델

```python
class Gender(enum.Enum):
    M = "M"
    F = "F"
    O = "O"

class MessageType(enum.Enum):
    USER = "user"
    BOT = "bot"

class NoticeStatus(enum.Enum):
    ACTIVE = "active"
    INACTIVE = "inactive"

class MoveInType(enum.Enum):
    """입주가능유형 코드"""
    IMMEDIATE = "30063INSTANT"  # 즉시입주
```

```
    NEGOTIATE = "30063DISCUSS"  # 협의가능
    AFTER_REPAIR = "30063REPAIR"  # 리모델링후입주
    SCHEDULED = "30063SCHEDULE"  # 날짜협의
    AFTER = "30063AFTER"  # 이후입주가능

# Property 관련 Enum 클래스들 추가
class PropertyType(enum.Enum):
    """매물 유형 코드"""
    APARTMENT = "30000C01"     # 아파트
    VILLA = "30000C04"         # 빌라/다세대
    OFFICETEL = "30000C02"     # 오피스텔
    HOUSE = "30000C03"         # 단독/다가구
    HOUSING = "30000C13"       # 주택
    COMMERCIAL = "30000C08"    # 상가
    OFFICE = "30000C05"        # 사무실
    BUILDING = "30000C06"      # 건물
    LAND = "30000C07"          # 토지
    FACTORY = "30000C10"       # 공장
    BUILDING_UNIT = "30000C11" # 건물유닛
    SHOPPING = "30000C12"      # 쇼핑몰
    OTHER = "30000C09"         # 기타

class PropertySubType(enum.Enum):
    """매물 서브타입 코드"""
    NEW_APT_SALE = "30001SC011"    # 신축분양
    NEW_011_SALE = "NEW_011_SALE"  # 신축분양(아파트)
    NEW_013_SALE = "NEW_013_SALE"  # 신축분양(아파트 기타)
    NEW_021_SALE = "30001SC021"    # 신축분양(오피스텔)
    NEW_031_SALE = "30001SC031"    # 신축분양(단독/다가구)
    NEW_032_SALE = "30001SC032"    # 신축분양(단독/다가구2)
    NEW_041_SALE = "30001SC041"    # 신축분양(빌라)
    NEW_042_SALE = "NEW_042_SALE"  # 신축분양(다세대)
    NEW_043_SALE = "NEW_043_SALE"  # 신축분양(다세대2)
    NEW_044_SALE = "NEW_044_SALE"  # 신축분양(주택)
    NEW_045_SALE = "30001SC045"    # 신축분양(주택2)
    NEW_051_SALE = "30001SC051"    # 신축분양(사무실)
    NEW_061_SALE = "NEW_061_SALE"  # 신축분양(건물)
    NEW_071_SALE = "30001SC071"    # 신축분양(상가)
    NEW_081_SALE = "NEW_081_SALE"  # 신축분양(상가주택)
    NEW_091_SALE = "NEW_091_SALE"  # 신축분양(토지)
    NEW_092_SALE = "NEW_092_SALE"  # 신축분양(토지2)
    NEW_101_SALE = "30001SC101"    # 신축분양(공장)
    NEW_111_SALE = "30001SC111"    # 신축분양(기타)
    NEW_112_SALE = "30001SC112"    # 신축분양(기타2)
    NEW_113_SALE = "30001SC113"    # 신축분양(기타3)
    NEW_114_SALE = "30001SC114"    # 신축분양(기타 상가주택)
    PRESALE_APT = "30001SC012"     # 분양권
    LEASABLE_APT = "30001SC013"    # 임대아파트
    RECONSTRUCTED = "30001SC014"    # 재건축
    REDEVELOPED = "30001SC015"     # 재개발
    VILLA = "30001SC041"           # 빌라
    MULTI_FAMILY = "30001SC042"    # 다세대
    MULTI_UNIT = "30001SC043"      # 다가구
    HOUSING = "30001SC044"         # 주택
    COMMERCIAL = "30001SC061"      # 상가
    OFFICE = "30001SC081"          # 사무실
    LAND = "30001SC091"            # 토지
    LAND2 = "30001SC092"           # 토지2
```

```
    OTHER = "30001SC999"           # 기타

class HeatingType(enum.Enum):
    """난방 유형 코드"""
    CENTRAL = "30081CENTRAL"       # 중앙난방
    INDIVIDUAL = "30081INDIVIDUAL" # 개별난방
    DISTRICT = "30081DISTRICT"     # 지역난방
    NONE = "30081NONE"             # 난방없음
    SEPARATE = "30081SEPARATE"     # 개별난방
    CENTER = "30081CENTER"         # 중앙난방
    DISTRICT_CENTRAL = "30081DISTRICT_CENTRAL"  # 지역중앙난방
    DISTRICT_LOCAL = "DISTRICT_LOCAL"  # 지역난방
    OTHER = "30081OTHER"           # 기타

class RentalType(enum.Enum):
    """임대 유형"""
    MONTHLY = "월세"
    YEARLY = "전세"
    HALF_YEARLY = "반전세"

class LoanAvailability(enum.Enum):
    """대출가능여부"""
    NO_LOAN = "30053C010"     # 융자금 없음
    LOAN_UNDER_30 = "30053C011"    # 융자금 시세대비 30%미만
    LOAN_OVER_30 = "30053C012" # 시세대비 30%이상

class Address(Base):
    __tablename__ = 'addresses'
    __table_args__ = (
        Index('idx_area_name', 'area_name'),  # area_name에 인덱스 추가
    )

    address_id = Column(Integer, primary_key=True)
    area_name = Column(String(100), nullable=False)  # 지역명
    latitude = Column(Float)  # 위도
    longitude = Column(Float)  # 경도
    created_at = Column(DateTime, default=lambda: datetime.now(timezone.utc))

    # 관계 설정
    cultural_facilities = relationship("CulturalFacility", back_populates="address")
    cultural_festivals = relationship("CulturalFestival", back_populates="address")
    subway_stations = relationship("SubwayStation", back_populates="address")
    crime_stats = relationship("CrimeStats", back_populates="address")
    population_stats = relationship("PopulationStats", back_populates="address")
    distances = relationship("LocationDistance", back_populates="address")

class CulturalFacility(Base):
    __tablename__ = 'cultural_facilities'

    facility_id = Column(Integer, primary_key=True)
    address_id = Column(Integer, ForeignKey('addresses.address_id'))
    facility_name = Column(String(100), nullable=False)
    facility_type = Column(String(50), nullable=False)  # 문화시설 유형
    created_at = Column(DateTime, default=lambda: datetime.now(timezone.utc))

    address = relationship("Address", back_populates="cultural_facilities")

class CulturalFestival(Base):
```

```
    __tablename__ = 'cultural_festivals'

    festival_id = Column(Integer, primary_key=True)
    address_id = Column(Integer, ForeignKey('addresses.address_id'))
    festival_name = Column(String(100), nullable=False)
    begin_date = Column(String(10))  # YYYY-MM-DD
    end_date = Column(String(10))    # YYYY-MM-DD
    created_at = Column(DateTime, default=lambda: datetime.now(timezone.utc))

    address = relationship("Address", back_populates="cultural_festivals")

class SubwayStation(Base):
    __tablename__ = 'subway_stations'

    station_id = Column(Integer, primary_key=True)
    address_id = Column(Integer, ForeignKey('addresses.address_id'))
    line_info = Column(String(20), nullable=False)  # 호선 정보
    created_at = Column(DateTime, default=lambda: datetime.now(timezone.utc))

    address = relationship("Address", back_populates="subway_stations")

class CrimeStats(Base):
    __tablename__ = 'crime_stats'

    stat_id = Column(Integer, primary_key=True)
    address_id = Column(Integer, ForeignKey('addresses.address_id'))
    reference_date = Column(String(8))  # STDR_DE_ID (YYYYMMDD 형식)
    total_population = Column(Float)    # TOT_LVPOP_CO
    crime_category = Column(String(50), nullable=False)     # 범죄 대분류
    crime_subcategory = Column(String(50), nullable=False)  # 범죄 소분류
    incident_count = Column(Integer, nullable=False)        # 발생 건수
    crime_rate = Column(Float)          # 인구 10만명당 범죄 발생률
    created_at = Column(DateTime, default=lambda: datetime.now(timezone.utc))

    # 관계 설정
    address = relationship("Address", back_populates="crime_stats")

class PopulationStats(Base):
    __tablename__ = 'population_stats'

    stat_id = Column(Integer, primary_key=True)
    address_id = Column(Integer, ForeignKey('addresses.address_id'))
    collection_time = Column(DateTime, nullable=False)  # 데이터 수집 시간
    population_level = Column(String(50))  # 실시간 인구 수준 (약간 붐빔, 보통 등)
    population_message = Column(Text)  # 실시간 인구 메시지
    min_population = Column(Integer)  # 인구 최소값
    max_population = Column(Integer)  # 인구 최대값
    male_ratio = Column(Float)  # 남성 비율
    female_ratio = Column(Float)  # 여성 비율
    age_0_9_ratio = Column(Float)  # 0-9세 비율
    age_10s_ratio = Column(Float)  # 10대 비율
    age_20s_ratio = Column(Float)  # 20대 비율
    age_30s_ratio = Column(Float)  # 30대 비율
    age_40s_ratio = Column(Float)  # 40대 비율
    age_50s_ratio = Column(Float)  # 50대 비율
    age_60s_ratio = Column(Float)  # 60대 비율
    age_70s_ratio = Column(Float)  # 70대 이상 비율
    resident_ratio = Column(Float)  # 거주인구 비율
```

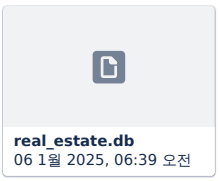
```
    non_resident_ratio = Column(Float)  # 비거주인구 비율
    created_at = Column(DateTime, default=lambda: datetime.now(timezone.utc))

    # 관계 설정
    address = relationship("Address", back_populates="population_stats")

class LocationDistance(Base):
    __tablename__ = 'location_distances'

    id = Column(Integer, primary_key=True)
    property_location_id = Column(Integer, ForeignKey('property_locations.property_location_id'), nullable=
    address_id = Column(Integer, ForeignKey('addresses.address_id'), nullable=False)
    distance = Column(Float, nullable=False)  # 미터 단위

    property_location = relationship("Location", back_populates="distances")
    address = relationship("Address", back_populates="distances")

class Location(Base):
    __tablename__ = "property_locations"

    property_location_id = Column(Integer, primary_key=True)
    sido = Column(String(20), nullable=False)
    sigungu = Column(String(20), nullable=False)
    dong = Column(String(20), nullable=True)
    jibun_main = Column(String(20), nullable=True)
    jibun_sub = Column(String(20), nullable=True)
    is_mountain = Column(Boolean, nullable=False)
    latitude = Column(DECIMAL(10, 7), nullable=False)
    longitude = Column(DECIMAL(10, 7), nullable=False)

    # 관계 설정
    properties = relationship("Property", back_populates="location")
    distances = relationship("LocationDistance", back_populates="property_location")

class User(Base):
    __tablename__ = "users"

    uuid = Column(CHAR(36), primary_key=True, default=lambda: str(uuid.uuid4()))
    gender = Column(SQLEnum(Gender), nullable=False)
    age = Column(Integer, nullable=False)
    age_group = Column(String(10), nullable=False)
    desired_location = Column(String(255), nullable=False)
    created_at = Column(DateTime, nullable=False, server_default=func.now())
    updated_at = Column(DateTime, nullable=False, server_default=func.now(), onupdate=func.now())

    # 관계 설정
    feedbacks = relationship("Feedback", back_populates="user")
    chat_logs = relationship("ChatLog", back_populates="user")
    favorites = relationship("Favorite", back_populates="user")
    user_actions = relationship("UserAction", back_populates="user")

class Feedback(Base):
    __tablename__ = "feedback"

    id = Column(Integer, primary_key=True)
    user_uuid = Column(CHAR(36), ForeignKey('users.uuid'), nullable=False)
    homepage_rating = Column(Integer, nullable=False)
    q1_accuracy = Column(Integer, nullable=False)
```

```
    q2_naturalness = Column(Integer, nullable=False)
    q3_resolution = Column(Integer, nullable=False)
    feedback_text = Column(Text)
    created_at = Column(DateTime, nullable=False, server_default=func.now())

    # 관계 설정
    user = relationship("User", back_populates="feedbacks")

class Notice(Base):
    __tablename__ = "notices"

    id = Column(Integer, primary_key=True)
    title = Column(String(255), nullable=False)
    content = Column(Text, nullable=False)
    author_id = Column(CHAR(36), ForeignKey('users.uuid'), nullable=False)
    created_at = Column(DateTime, nullable=False, server_default=func.now())
    updated_at = Column(DateTime)
    status = Column(SQLEnum(NoticeStatus), nullable=False, default=NoticeStatus.ACTIVE)

class ChatLog(Base):
    __tablename__ = "chat_logs"

    id = Column(Integer, primary_key=True)
    user_uuid = Column(CHAR(36), ForeignKey('users.uuid'), nullable=False)
    session_id = Column(CHAR(36), nullable=False)
    message_type = Column(SQLEnum(MessageType), nullable=False)
    message = Column(Text, nullable=False)
    created_at = Column(DateTime, nullable=False, server_default=func.now())

    # 관계 설정
    user = relationship("User", back_populates="chat_logs")

class UserAction(Base):
    __tablename__ = "user_actions"

    id = Column(Integer, primary_key=True)
    user_uuid = Column(CHAR(36), ForeignKey('users.uuid'))
    session_id = Column(CHAR(36), nullable=False)
    action_type = Column(String(50), nullable=False)
    action_time = Column(DateTime, nullable=False, server_default=func.now())
    page_url = Column(String(255), nullable=False)
    referrer_url = Column(String(255))
    device_info = Column(String(255), nullable=False)
    location = Column(String(255))
    event_details = Column(JSON)

    # 관계 설정
    user = relationship("User", back_populates="user_actions")

class Property(Base):
    __tablename__ = "property_info"

    property_id = Column(Integer, primary_key=True)
    property_location_id = Column(Integer, ForeignKey("property_locations.property_location_id"), nullable=
    property_type = Column(SQLEnum(PropertyType))
    property_subtype = Column(SQLEnum(PropertySubType), nullable=True)
    building_name = Column(String(100), nullable=True)
    detail_address = Column(String(200), nullable=True)
```

```
    construction_date = Column(String(10))  # YYYY-MM-DD 형식으로 저장
    total_area = Column(DECIMAL(10,2), nullable=True)
    exclusive_area = Column(DECIMAL(10,2), nullable=True)
    land_area = Column(DECIMAL(10,2), nullable=True)
    floor = Column(Integer, nullable=True)
    total_floor = Column(Integer, nullable=True)
    basement_floor = Column(Integer, nullable=True)
    room_count = Column(Integer, nullable=True)
    bathroom_count = Column(Integer, nullable=True)
    parking_count = Column(Integer, nullable=True)
    heating_type = Column(SQLEnum(HeatingType), nullable=True)
    direction = Column(String(20), nullable=True)
    purpose_type = Column(String(20), nullable=True)
    current_usage = Column(String(100), nullable=True)
    recommended_usage = Column(String(100), nullable=True)
    facilities = Column(JSON, nullable=True)
    description = Column(Text, nullable=True)
    photos = Column(JSON, nullable=True)  # 사진 URL 리스트 저장

    update_count = Column(Integer, default=0)  # 업데이트 횟수
    is_active = Column(Boolean, default=True)  # 활성화 상태
    inactive_reason = Column(String(200))  # 비활성화 사유
    first_seen = Column(DateTime, default=lambda: datetime.now(timezone.utc))  # 최초 발견 시간
    last_seen = Column(DateTime, default=lambda: datetime.now(timezone.utc), onupdate=lambda: datetime.now(

    # 관계 설정
    location = relationship("Location", back_populates="properties")
    rentals = relationship("Rental", back_populates="property")
    sales = relationship("Sale", back_populates="property")
    favorites = relationship("Favorite", back_populates="property")

class Favorite(Base):
    __tablename__ = "favorites"

    id = Column(Integer, primary_key=True)
    user_uuid = Column(CHAR(36), ForeignKey('users.uuid'), nullable=False)
    item_id = Column(Integer, ForeignKey('property_info.property_id'), nullable=False)
    item_type = Column(String(50), nullable=False)
    name = Column(String(255), nullable=False)
    latitude = Column(DECIMAL(10, 7))
    longitude = Column(DECIMAL(10, 7))
    created_at = Column(DateTime, nullable=False, server_default=func.now())

    # 관계 설정
    user = relationship("User", back_populates="favorites")
    property = relationship("Property", back_populates="favorites")

class Rental(Base):
    __tablename__ = "rentals"

    id = Column(Integer, primary_key=True)
    property_id = Column(Integer, ForeignKey('property_info.property_id'), nullable=False)
    rental_type = Column(SQLEnum(RentalType), nullable=False)
    deposit = Column(DECIMAL(10,2), nullable=False)
    monthly_rent = Column(DECIMAL(10,2), nullable=False)
    maintenance_cost_info = Column(JSON)
    loan_availability = Column(SQLEnum(LoanAvailability))
    move_in_type = Column(SQLEnum(MoveInType))
```

```
    negotiable = Column(Boolean, default=False)

    # 관계 설정
    property = relationship("Property", back_populates="rentals")

class Sale(Base):
    __tablename__ = "sales"

    id = Column(Integer, primary_key=True)
    property_id = Column(Integer, ForeignKey('property_info.property_id'), nullable=False)
    price = Column(DECIMAL(10,2), nullable=False)
    maintenance_cost_info = Column(JSON)
    loan_availability = Column(SQLEnum(LoanAvailability))
    move_in_type = Column(SQLEnum(MoveInType))
    negotiable = Column(Boolean, default=False)
    transaction_date = Column(DateTime)
    end_date = Column(DateTime)

    # 관계 설정
    property = relationship("Property", back_populates="sales")

class APISearchData(Base):
    __tablename__ = 'api_search_data'

    id = Column(Integer, primary_key=True)
    api_name = Column(String(100), nullable=False)  # API 이름
    parameter = Column(String(500))  # API 파라미터
    response = Column(String)  # API 응답 데이터 (TEXT 타입)
    created_at = Column(DateTime, default=lambda: datetime.now(timezone.utc))
```

※ 데이터베이스



real_estate.db
06 1월 2025, 06:39 오전