ERD 피드백 적용

▼ ERD 데이터 흐름 분석

📍 시나리오 1: 사용자 회원가입 + 위치 설정

1단계: 회원가입 (카카오 로그인)

```
[사용자] → [카카오 OAuth] → [백엔드]
```

데이터 흐름:

```
-- 1. 카카오에서 받은 정보로 USER 테이블 생성
INSERT INTO USER (
    email,
    nickname,
    profile_image,
    provider,           -- 'KAKAO'
    provider_id,        -- 카카오 고유 ID
    created_at
) VALUES (...);

-- 2. USER_REPUTATION 자동 생성 (초기값)
INSERT INTO USER_REPUTATION (
    user_id,
    manner_score,       -- 0.00
    trust_score,        -- 0
    total_meetings,     -- 0
    created_at
) VALUES (1, 0.00, 0, 0, NOW());
```

2단계: 위치 정보 입력 (카카오맵 API)

```
[사용자] → [카카오맵 주소 검색] → [좌표 변환] → [USER 테이블 업데이트]
```

프론트엔드 → 백엔드 요청:

```
// 카카오맵 API 응답 예시
{
  "address": "서울특별시 강남구 테헤란로 152",
  "latitude": 37.5012767241426,
  "longitude": 127.039600248343
}
```

데이터 흐름:

```
-- USER 테이블 업데이트
UPDATE USER
SET
    location = '서울특별시 강남구 테헤란로 152',
    latitude = 37.5012767241426,
    longitude = 127.039600248343,
    location_updated_at = NOW(),
    updated_at = NOW()
WHERE user_id = 1;
```

3단계: 관심사 선택

```
[사용자] → [관심사 선택 UI] → [USER_INTEREST 테이블]
```

데이터 흐름:

```
-- 사용자가 "운동", "맛집탐방", "독서" 선택
INSERT INTO USER_INTEREST (user_id, interest_id, created_at)
VALUES
    (1, 3, NOW()),  -- 운동
    (1, 7, NOW()),  -- 맛집탐방
    (1, 12, NOW()); -- 독서
```

현재 상태:

```
USER 테이블:
- user_id: 1
- location: "서울특별시 강남구 테헤란로 152"
```

- latitude: 37.5012767241426
- longitude: 127.039600248343

USER_INTEREST:
- (1, 3) 운동
- (1, 7) 맛집탐방
- (1, 12) 독서

USER_REPUTATION:
- manner_score: 0.00
- trust_score: 0

🤖 시나리오 2: AI 알고리즘 모임 추천

AI 추천 쿼리 로직

1단계: 위치 기반 필터링 (1km 이내)

```
SELECT
    m.meeting_id,
    m.title,
    m.location,
    m.meeting_date,
    m.max_members,
    m.current_members,
    m.status,
    -- Haversine 공식으로 거리 계산
    (6371 * acos(
        cos(radians(37.5012767241426)) * cos(radians(m.latitude)) *
        cos(radians(m.longitude) - radians(127.039600248343)) +
        sin(radians(37.5012767241426)) * sin(radians(m.latitude))
    )) AS distance_km
FROM MEETING m
WHERE
    m.status = 'RECRUITING'
    AND m.meeting_date > NOW()
```

```
HAVING distance_km <= 1  -- 1km 이내
ORDER BY distance_km ASC;
```

2단계: 관심사 매칭

```
SELECT
    m.meeting_id,
    m.title,
    COUNT(DISTINCT mt.interest_id) AS matched_interests,
    distance_km
FROM MEETING m
JOIN MEETING_TAG mt ON m.meeting_id = mt.meeting_id
JOIN (
    -- 사용자 관심사
    SELECT interest_id FROM USER_INTEREST WHERE user_id = 1
) ui ON mt.interest_id = ui.interest_id
WHERE distance_km <= 1
GROUP BY m.meeting_id
ORDER BY matched_interests DESC, distance_km ASC;
```

3단계: 이미 스와이프한 모임 제외

```
SELECT
    m.meeting_id,
    m.title,
    matched_interests,
    distance_km
FROM (위의 쿼리 결과) AS recommended
WHERE m.meeting_id NOT IN (
    SELECT meeting_id
    FROM SWIPE_HISTORY
    WHERE user_id = 1
)
LIMIT 10;
```

4단계: AI 가중치 점수 계산

```
SELECT
    m.meeting_id,
    m.title,
    -- AI 추천 점수
    (
        (matched_interests * 40) +              -- 관심사 일치 40%
        ((1 - distance_km) * 30) +              -- 거리 근접 30%
        (m.max_members - m.current_members) * 2 + -- 여유 인원 20%
        (creator_reputation.manner_score * 10)  -- 방장 평판 10%
    ) AS recommendation_score,
    distance_km,
    matched_interests
FROM MEETING m
LEFT JOIN USER_REPUTATION creator_reputation
    ON m.creator_id = creator_reputation.user_id
ORDER BY recommendation_score DESC
LIMIT 20;
```

스와이프 액션 저장

```
-- 사용자가 모임을 "좋아요" 했을 때
INSERT INTO SWIPE_HISTORY (
    user_id,
    meeting_id,
    action,
    swiped_at
) VALUES (1, 42, 'LIKE', NOW());

-- 사용자가 모임을 "싫어요" 했을 때
INSERT INTO SWIPE_HISTORY (
    user_id,
    meeting_id,
    action,
    swiped_at
) VALUES (1, 43, 'DISLIKE', NOW());
```

👥 시나리오 3: 모임장이 모임 생성

1단계: 모임 기본 정보 입력

[모임장] → [모임 제목/설명/인원] → [카카오맵 장소 선택] → [MEETING 테이블]

프론트엔드 요청:

```
{
  "title": "강남역 런닝 크루 모집",
  "description": "매주 토요일 아침 10km 런닝",
  "max_members": 10,
  "meeting_date": "2026-01-15 09:00:00",
  "is_approval_required": true,  // 승인제

  // 카카오맵에서 선택한 위치
  "location": "서울특별시 강남구 강남대로 396",
  "latitude": 37.4979518,
  "longitude": 127.0276368,
  "location_detail": "강남역 2번 출구"
}
```

데이터 흐름:

```
-- 1. MEETING 테이블 생성
INSERT INTO MEETING (
    title,
    description,
    location,
    latitude,
    longitude,
    location_detail,
    max_members,
    current_members,
    meeting_date,
    status,
    is_approval_required,
```

```
    creator_id,
    created_at
) VALUES (
    '강남역 런닝 크루 모집',
    '매주 토요일 아침 10km 런닝',
    '서울특별시 강남구 강남대로 396',
    37.4979518,
    127.0276368,
    '강남역 2번 출구',
    10,
    1,  -- 모임장 자동 포함
    '2026-01-15 09:00:00',
    'RECRUITING',
    true,
    1,  -- 모임장 user_id
    NOW()
);

-- 2. 모임장 자동 참가 (MEETING_MEMBER)
INSERT INTO MEETING_MEMBER (
    meeting_id,
    user_id,
    role,
    status,
    joined_at,
    created_at
) VALUES (
    LAST_INSERT_ID(),  -- 방금 생성된 meeting_id
    1,                 -- 모임장 user_id
    'HOST',
    'APPROVED',
    NOW(),
    NOW()
);

-- 3. 관심사 태그 추가
INSERT INTO MEETING_TAG (meeting_id, interest_id, created_at)
VALUES
```

```
    (LAST_INSERT_ID(), 3, NOW()),  -- 운동
    (LAST_INSERT_ID(), 5, NOW());  -- 러닝
```

🔔 시나리오 4: 일반 사용자가 모임 참가 신청 (승인제)

1단계: 참가 신청

```
-- MEETING_MEMBER에 대기 상태로 추가
INSERT INTO MEETING_MEMBER (
    meeting_id,
    user_id,
    role,
    status,             -- 'PENDING' (대기중)
    request_message,
    created_at,
    joined_at
) VALUES (
    42,
    2,                  -- 신청자 user_id
    'MEMBER',
    'PENDING',
    '런닝 경험 3년차입니다. 잘 부탁드립니다!',
    NOW(),
    NOW()
);
```

2단계: 모임장에게 알림 전송

```
-- NOTIFICATION 테이블에 알림 생성
INSERT INTO NOTIFICATION (
    user_id,            -- 모임장 ID
    type,
    title,
    message,
    related_type,
    related_id,         -- meeting_id
    is_read,
```

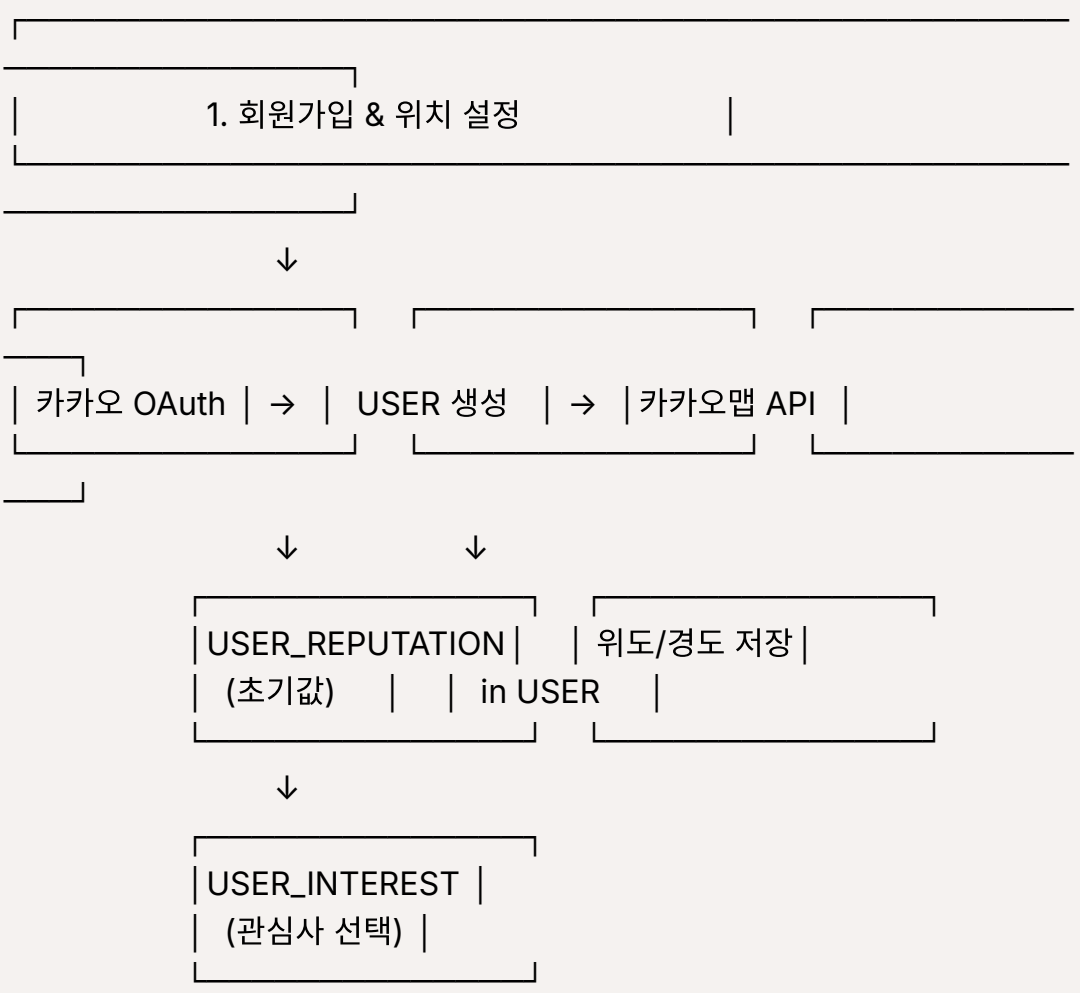
```
    created_at
) VALUES (
    1,                  -- 모임장 user_id
    'MEMBER_JOINED',
    '새로운 참가 신청이 있습니다',
    '"닉네임"님이 "강남역 런닝 크루" 모임 참가를 신청했습니다.',
    'MEETING',
    42,
    false,
    NOW()
);
```

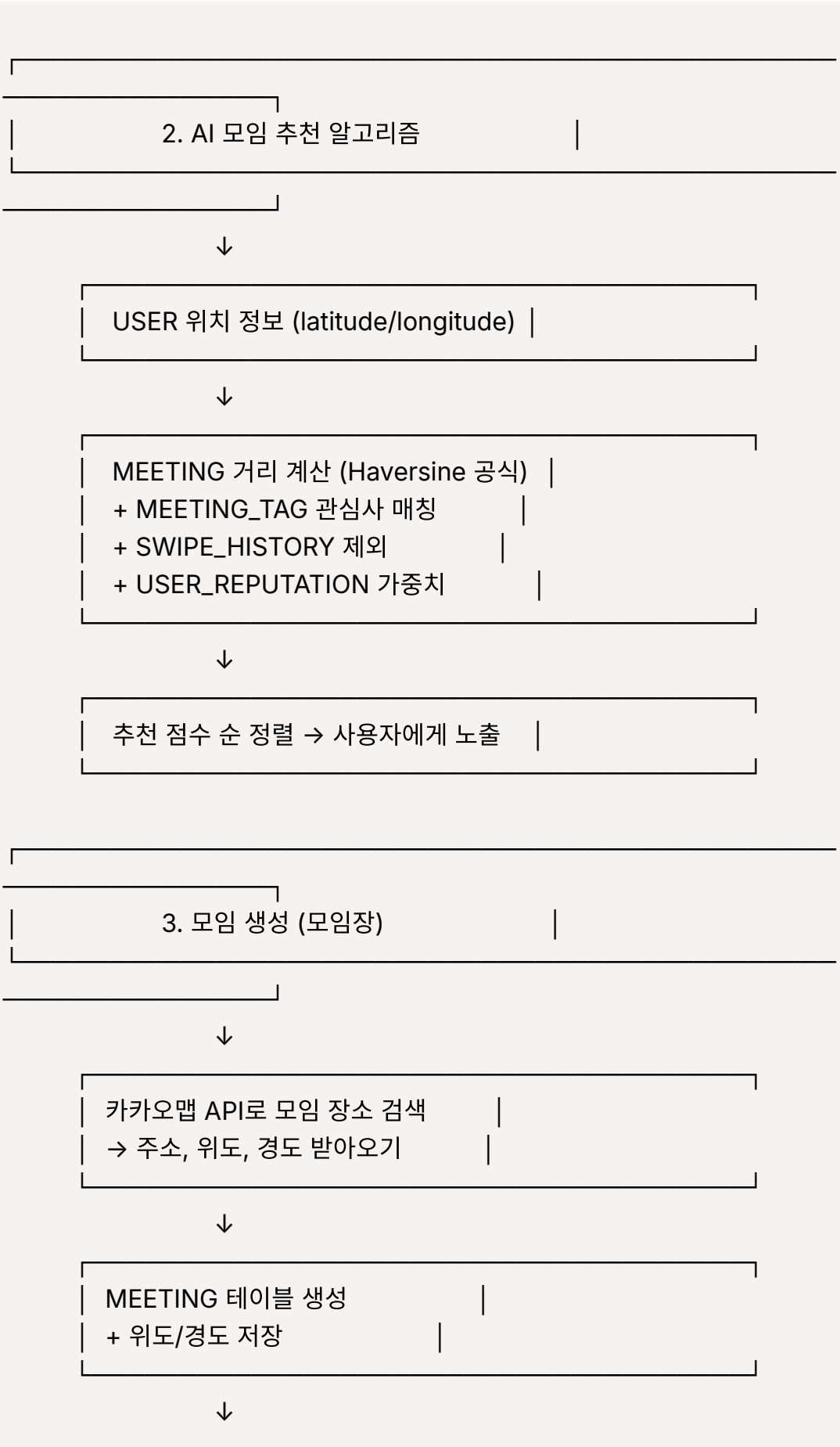
3단계: 모임장이 승인

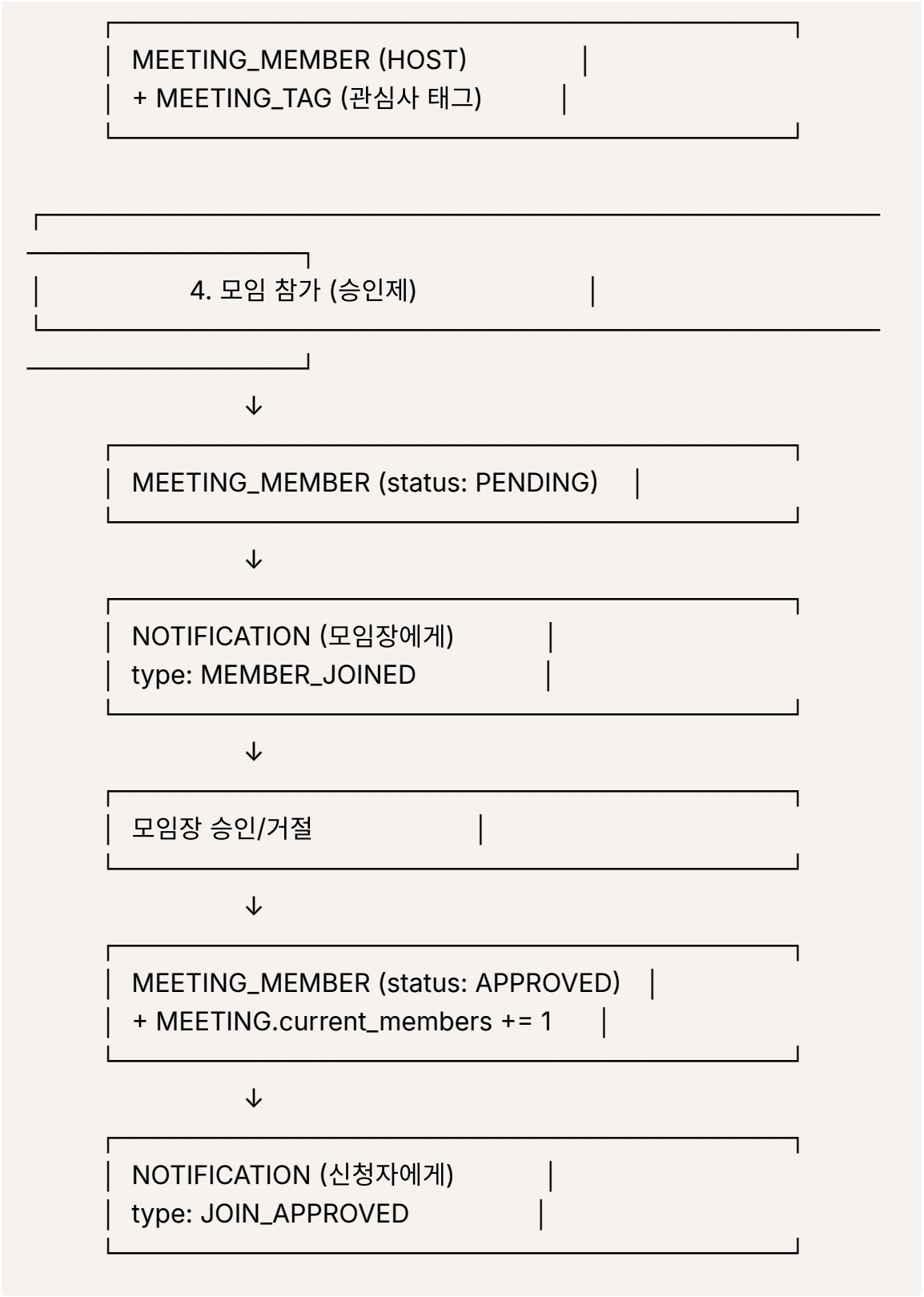
```
-- MEETING_MEMBER 상태 업데이트
UPDATE MEETING_MEMBER
SET
    status = 'APPROVED',
    response_message = '환영합니다! 토요일에 뵙겠습니다.',
    responded_at = NOW(),
    updated_at = NOW()
WHERE meeting_id = 42 AND user_id = 2;

-- MEETING 현재 인원 증가
UPDATE MEETING
SET
    current_members = current_members + 1,
    updated_at = NOW()
WHERE meeting_id = 42;

-- 신청자에게 승인 알림
INSERT INTO NOTIFICATION (
    user_id,
    type,
    title,
    message,
    related_type,
```

```
    related_id,
    is_read,
    created_at
) VALUES (
    2,                  -- 신청자 user_id
    'JOIN_APPROVED',
    '참가가 승인되었습니다',
    '"강남역 런닝 크루" 모임 참가가 승인되었습니다.',
    'MEETING',
    42,
    false,
    NOW()
);
```

📊 전체 데이터 흐름 다이어그램




2. AI 모임 추천 알고리즘
USER 위치 정보 (latitude/longitude)
MEETING 거리 계산 (Haversine 공식)
+ MEETING_TAG 관심사 매칭
+ SWIPE_HISTORY 제외
+ USER_REPUTATION 가중치
추천 점수 순 정렬 → 사용자에게 노출
3. 모임 생성 (모임장)
카카오맵 API로 모임 장소 검색
→ 주소, 위도, 경도 받아오기
MEETING 테이블 생성
+ 위도/경도 저장



🔍 핵심 쿼리 정리

1️⃣ 사용자 주변 1km 모임 찾기

```
SELECT
    m.*,
    (6371 * acos(
        cos(radians(?)) * cos(radians(m.latitude)) *
        cos(radians(m.longitude) - radians(?)) +
        sin(radians(?)) * sin(radians(m.latitude))
    )) AS distance_km
FROM MEETING m
WHERE m.status = 'RECRUITING'
HAVING distance_km <= 1
ORDER BY distance_km;
```

2️⃣ 관심사 기반 추천

```
SELECT
    m.meeting_id,
    m.title,
    COUNT(mt.interest_id) AS matched_count
FROM MEETING m
JOIN MEETING_TAG mt ON m.meeting_id = mt.meeting_id
WHERE mt.interest_id IN (
    SELECT interest_id FROM USER_INTEREST WHERE user_id = ?
)
GROUP BY m.meeting_id
ORDER BY matched_count DESC;
```

3️⃣ 승인 대기 중인 참가 신청 조회

```
SELECT
    mm.member_id,
    u.nickname,
    u.profile_image,
    ur.manner_score,
    mm.request_message,
    mm.created_at
FROM MEETING_MEMBER mm
JOIN USER u ON mm.user_id = u.user_id
```

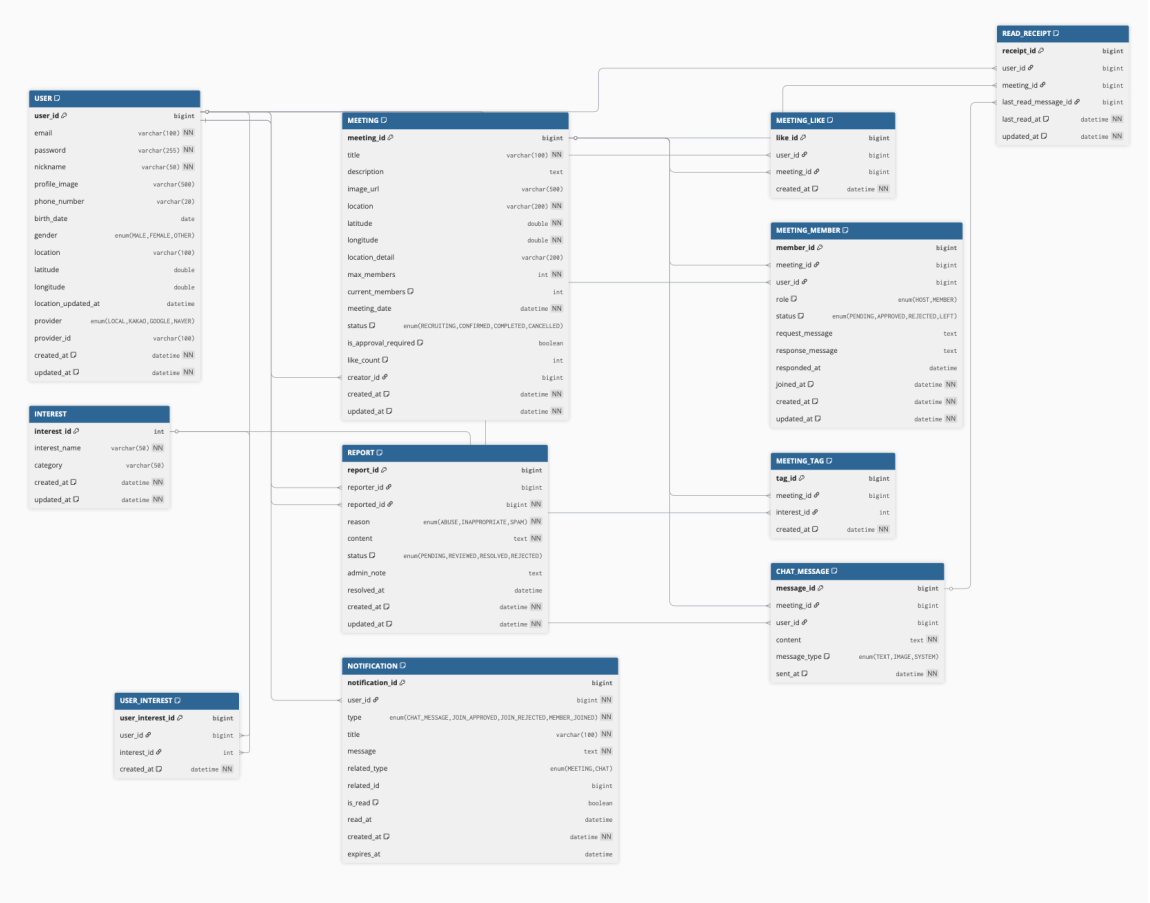
```
LEFT JOIN USER_REPUTATION ur ON u.user_id = ur.user_id
WHERE mm.meeting_id = ?
  AND mm.status = 'PENDING'
ORDER BY mm.created_at DESC;
```

✅ ERD가 잘 설계된 이유

1. **위치 기반 검색:** `latitude`, `longitude` 컬럼으로 정확한 거리 계산
2. **AI 추천 시스템:** 관심사, 위치, 평판 데이터를 모두 JOIN 가능
3. **승인제 모임:** `MEETING_MEMBER.status`로 대기/승인 상태 관리
4. **알림 시스템:** `NOTIFICATION` 테이블로 모든 이벤트 추적
5. **스와이프 이력:** `SWIPE_HISTORY`로 중복 추천 방지

이 구조로 카카오맵 연동과 AI 추천 알고리즘을 완벽하게 구현할 수 있습니다! 🚀

▼ ERD



📋 전체 테이블 명세

1️⃣ USER (사용자) - ✨

```
USER
├─ user_id (BIGINT, PK, AUTO_INCREMENT)
├─ email (VARCHAR(100), UNIQUE, NOT NULL)
├─ password (VARCHAR(255), NOT NULL)
├─ nickname (VARCHAR(50), NOT NULL)
├─ profile_image (VARCHAR(500), NULL)
├─ location (VARCHAR(100), NULL)
├─ latitude (DOUBLE, NULL)
├─ longitude (DOUBLE, NULL)
├─ location_updated_at (DATETIME, NULL)
├─ provider (ENUM: 'LOCAL', 'KAKAO', 'GOOGLE', 'NAVER', NULL)
├─ provider_id (VARCHAR(100), NULL)
├─ created_at (DATETIME, NOT NULL, DEFAULT CURRENT_TIMESTAM
P)
└─ updated_at (DATETIME, NOT NULL, DEFAULT CURRENT_TIMESTAM
P ON UPDATE CURRENT_TIMESTAMP)

INDEX idx_email (email)
INDEX idx_user_location (latitude, longitude)
```

▼ 예시

- - 카카오로 가입한 사용자 예시
  user_id: 1
  email: "hong@gmail.com"
  password: NULL -- 소셜 로그인은 비번 없음
  nickname: "홍길동"
  profile_image: "https://k.kakaocdn.net/dn/profile/abc123.jpg"
  location: "서울특별시 강남구"
  latitude: 37.5172
  longitude: 127.0473
  provider: "KAKAO"
  provider_id: "2736482847"
  created_at: "2024-01-15 09:30:25"
  updated_at: "2024-01-15 09:30:25"

2️⃣ INTEREST (관심사)

```
INTEREST
├─ interest_id (INT, PK, AUTO_INCREMENT)
├─ interest_name (VARCHAR(50), UNIQUE, NOT NULL)
├─ category (VARCHAR(50), NULL)
├─ created_at (DATETIME, NOT NULL, DEFAULT CURRENT_TIMESTAM
P)
└─ updated_at (DATETIME, NOT NULL, DEFAULT CURRENT_TIMESTAM
P ON UPDATE CURRENT_TIMESTAMP)

UNIQUE KEY uk_interest_name (interest_name)
```

▼ 예시

- - 초기 데이터 예시
  interest_id: 1, interest_name: "러닝", category: "운동"
  interest_id: 2, interest_name: "등산", category: "운동"
  interest_id: 3, interest_name: "맛집탐방", category: "음식"
  interest_id: 4, interest_name: "영화감상", category: "문화"

3️⃣ USER_INTEREST (사용자 관심사)

```
USER_INTEREST
├─ user_interest_id (BIGINT, PK, AUTO_INCREMENT)
├─ user_id (BIGINT, FK → USER)
├─ interest_id (INT, FK → INTEREST)
├─ created_at (DATETIME, NOT NULL, DEFAULT CURRENT_TIMESTAM
P)
└─ UNIQUE KEY uk_user_interest (user_id, interest_id)

FOREIGN KEY fk_userinterest_user (user_id) REFERENCES USER(user_i
d) ON DELETE CASCADE
FOREIGN KEY fk_userinterest_interest (interest_id) REFERENCES INTER
EST(interest_id) ON DELETE CASCADE
```

▼ 예시

- - 홍길동(user_id=1)이 러닝, 맛집탐방을 좋아함
  user_interest_id: 1, user_id: 1, interest_id: 1 -- 러닝

user_interest_id: 2, user_id: 1, interest_id: 3 -- 맛집탐방

- - 김철수(user_id=2)가 등산, 영화를 좋아함

user_interest_id: 3, user_id: 2, interest_id: 2 -- 등산
user_interest_id: 4, user_id: 2, interest_id: 4 -- 영화감상

4️⃣ MEETING (모임)

```
MEETING
├─ meeting_id (BIGINT, PK, AUTO_INCREMENT)
├─ title (VARCHAR(100), NOT NULL)
├─ description (TEXT, NULL)
├─ image_url (VARCHAR(500), NULL)
├─ location (VARCHAR(200), NOT NULL)
├─ latitude (DOUBLE, NOT NULL)
├─ longitude (DOUBLE, NOT NULL)
├─ location_detail (VARCHAR(200), NULL)
├─ max_members (INT, NOT NULL)
├─ current_members (INT, DEFAULT 1)
├─ meeting_date (DATETIME, NOT NULL)
├─ status (ENUM: 'RECRUITING', 'CONFIRMED', 'COMPLETED', 'CANC
ELLED', DEFAULT 'RECRUITING')
├─ is_approval_required (BOOLEAN, DEFAULT false)
├─ like_count (INT, DEFAULT 0)
├─ creator_id (BIGINT, FK → USER)
├─ created_at (DATETIME, NOT NULL, DEFAULT CURRENT_TIMESTAM
P)
└─ updated_at (DATETIME, NOT NULL, DEFAULT CURRENT_TIMESTAM
P ON UPDATE CURRENT_TIMESTAMP)

INDEX idx_meeting_location (latitude, longitude)
INDEX idx_meeting_date_status (meeting_date, status)
INDEX idx_meeting_status (status)
INDEX idx_creator (creator_id)
INDEX idx_like_count_date (like_count, meeting_date)

FOREIGN KEY fk_meeting_creator (creator_id) REFERENCES USER(user
_id) ON DELETE CASCADE
```

▼ 예시

- - 강남역 러닝 모임 예시
  meeting_id: 1
  title: "강남역 저녁 러닝 크루"
  description: "매주 화요일 저녁 7시에 만나서 한강까지 달려요!"
  location: "서울 강남구 강남대로 396"
  latitude: 37.4979
  longitude: 127.0276
  location_detail: "강남역 2번 출구"
  max_members: 8
  current_members: 3
  meeting_date: "2024-01-23 19:00:00"
  status: "RECRUITING"
  is_approval_required: false
  like_count: 12
  creator_id: 1

5️⃣ MEETING_LIKE (모임 좋아요)

```
MEETING_LIKE
├─ like_id (BIGINT, PK, AUTO_INCREMENT)
├─ user_id (BIGINT, FK → USER)
├─ meeting_id (BIGINT, FK → MEETING)
├─ created_at (DATETIME, NOT NULL, DEFAULT CURRENT_TIMESTAM
P)
└─ UNIQUE KEY uk_user_meeting_like (user_id, meeting_id)

INDEX idx_meeting_like (meeting_id)
INDEX idx_user_like (user_id)

FOREIGN KEY fk_like_user (user_id) REFERENCES USER(user_id) ON D
ELETE CASCADE
FOREIGN KEY fk_like_meeting (meeting_id) REFERENCES MEETING(me
eting_id) ON DELETE CASCADE
```

▼ 예시

- - 여러 사용자들이 1번 모임에 좋아요
  like_id: 1, user_id: 2, meeting_id: 1 -- 김철수가 좋아요

like_id: 2, user_id: 3, meeting_id: 1 -- 박영희가 좋아요
like_id: 3, user_id: 4, meeting_id: 1 -- 이민수가 좋아요

6️⃣ MEETING_MEMBER (모임 참가자)

```
MEETING_MEMBER
├─ member_id (BIGINT, PK, AUTO_INCREMENT)
├─ meeting_id (BIGINT, FK → MEETING)
├─ user_id (BIGINT, FK → USER)
├─ role (ENUM: 'HOST', 'MEMBER', DEFAULT 'MEMBER')
├─ status (ENUM: 'PENDING', 'APPROVED', 'REJECTED', 'LEFT', DEFAU
LT 'APPROVED')
├─ request_message (TEXT, NULL)
├─ response_message (TEXT, NULL)
├─ responded_at (DATETIME, NULL)
├─ joined_at (DATETIME, NOT NULL, DEFAULT CURRENT_TIMESTAMP)
├─ created_at (DATETIME, NOT NULL, DEFAULT CURRENT_TIMESTAM
P)
└─ updated_at (DATETIME, NOT NULL, DEFAULT CURRENT_TIMESTAM
P ON UPDATE CURRENT_TIMESTAMP)

UNIQUE KEY uk_meeting_user (meeting_id, user_id)
INDEX idx_meeting_status (meeting_id, status)
INDEX idx_user_status (user_id, status)
INDEX idx_meeting_role (meeting_id, role)

FOREIGN KEY fk_member_meeting (meeting_id) REFERENCES MEETIN
G(meeting_id) ON DELETE CASCADE
FOREIGN KEY fk_member_user (user_id) REFERENCES USER(user_id)
ON DELETE CASCADE
```

▼ 예시

- - 1번 모임 참가자들
  member_id: 1, meeting_id: 1, user_id: 1, role: "HOST", status: "APPROVED" -- 모임장
  member_id: 2, meeting_id: 1, user_id: 2, role: "MEMBER", status: "APPROVED" -- 일반 참가자

member_id: 3, meeting_id: 1, user_id: 5, role: "MEMBER", status: "PENDING" -- 승인 대기

7️⃣ MEETING_TAG (모임 관심사 태그)

```
MEETING_TAG
├─ tag_id (BIGINT, PK, AUTO_INCREMENT)
├─ meeting_id (BIGINT, FK → MEETING)
├─ interest_id (INT, FK → INTEREST)
├─ created_at (DATETIME, NOT NULL, DEFAULT CURRENT_TIMESTAM
P)
└─ UNIQUE KEY uk_meeting_interest (meeting_id, interest_id)

INDEX idx_meeting_tag (meeting_id)
INDEX idx_interest_tag (interest_id)

FOREIGN KEY fk_tag_meeting (meeting_id) REFERENCES MEETING(me
eting_id) ON DELETE CASCADE
FOREIGN KEY fk_tag_interest (interest_id) REFERENCES INTEREST(inte
rest_id) ON DELETE CASCADE
```

▼ 예시

- - 1번 모임에 "러닝" 태그 연결
  tag_id: 1, meeting_id: 1, interest_id: 1 -- 러닝 모임임을 표시

8️⃣ CHAT_MESSAGE (채팅 메시지)

```
CHAT_MESSAGE
├─ message_id (BIGINT, PK, AUTO_INCREMENT)
├─ meeting_id (BIGINT, FK → MEETING)
├─ user_id (BIGINT, FK → USER)
├─ content (TEXT, NOT NULL)
├─ message_type (ENUM: 'TEXT', 'IMAGE', 'SYSTEM', DEFAULT 'TEX
T')
├─ sent_at (DATETIME, NOT NULL, DEFAULT CURRENT_TIMESTAMP)
└─ INDEX idx_meeting_sent (meeting_id, sent_at)

FOREIGN KEY fk_message_meeting (meeting_id) REFERENCES MEETIN
```

```
G(meeting_id) ON DELETE CASCADE
FOREIGN KEY fk_message_user (user_id) REFERENCES USER(user_id)
ON DELETE CASCADE
```

▼ 예시

- - 1번 모임 채팅방 메시지들
  message_id: 1, meeting_id: 1, user_id: 1, content: "안녕하세요! 오늘 날씨 좋네요", message_type: "TEXT"
  message_id: 2, meeting_id: 1, user_id: 2, content: "네! 기대돼요 ㅎㅎ", message_type: "TEXT"
  message_id: 3, meeting_id: 1, user_id: 1, content: "https://s3.../map.jpg", message_type: "IMAGE"

9️⃣ READ_RECEIPT (읽음 처리)

```
READ_RECEIPT
├─ receipt_id (BIGINT, PK, AUTO_INCREMENT)
├─ user_id (BIGINT, FK → USER)
├─ meeting_id (BIGINT, FK → MEETING)
├─ last_read_message_id (BIGINT, FK → CHAT_MESSAGE, NULL)
├─ last_read_at (DATETIME, NOT NULL, DEFAULT CURRENT_TIMESTAMP)
├─ updated_at (DATETIME, NOT NULL, DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP)
└─ UNIQUE KEY uk_user_meeting_receipt (user_id, meeting_id)

INDEX idx_user_meeting (user_id, meeting_id)

FOREIGN KEY fk_receipt_user (user_id) REFERENCES USER(user_id) ON DELETE CASCADE
FOREIGN KEY fk_receipt_meeting (meeting_id) REFERENCES MEETING(meeting_id) ON DELETE CASCADE
FOREIGN KEY fk_receipt_message (last_read_message_id) REFERENCES CHAT_MESSAGE(message_id) ON DELETE SET NULL
```

▼ 예시

- - 각 사용자별 마지막 읽은 메시지
  receipt_id: 1, user_id: 1, meeting_id: 1, last_read_message_id: 3 -- 1번

사용자는 3번 메시지까지 읽음
receipt_id: 2, user_id: 2, meeting_id: 1, last_read_message_id: 2 -- 2번 사용자는 2번 메시지까지 읽음

🔟 REPORT (신고) -

```
REPORT
├─ report_id (BIGINT, PK, AUTO_INCREMENT)
├─ reporter_id (BIGINT, FK → USER)
├─ reported_id (BIGINT, FK → USER, NULL)
├─ reported_meeting_id (BIGINT, FK → MEETING, NULL)
├─ report_type (ENUM: 'USER', 'MEETING', 'CHAT', NOT NULL)
├─ reason (ENUM: 'SPAM', 'ABUSE', 'INAPPROPRIATE', 'NO_SHOW', 'OTHER', NOT NULL)
├─ content (TEXT, NOT NULL)
├─ status (ENUM: 'PENDING', 'REVIEWED', 'RESOLVED', 'REJECTED', DEFAULT 'PENDING')
├─ admin_note (TEXT, NULL)
├─ resolved_at (DATETIME, NULL)
├─ created_at (DATETIME, NOT NULL, DEFAULT CURRENT_TIMESTAMP)
└─ updated_at (DATETIME, NOT NULL, DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP)

INDEX idx_status_created (status, created_at)
INDEX idx_reporter (reporter_id)
INDEX idx_reported_user (reported_id)

FOREIGN KEY fk_report_reporter (reporter_id) REFERENCES USER(user_id) ON DELETE SET NULL
FOREIGN KEY fk_report_reported (reported_id) REFERENCES USER(user_id) ON DELETE SET NULL
FOREIGN KEY fk_report_meeting (reported_meeting_id) REFERENCES MEETING(meeting_id) ON DELETE SET NULL
```

▼ 예시

- - 스팸 사용자 신고 예시
  report_id: 1

reporter_id: 2 -- 김철수가 신고함
reported_id: 7 -- 스팸러를 신고함
reason: "SPAM" -- 스팸으로 신고
content: "계속 광고 메시지 보내고 있습니다"
status: "PENDING" -- 검토 중

1️⃣1️⃣ NOTIFICATION (알림)

```
NOTIFICATION
├─ notification_id (BIGINT, PK, AUTO_INCREMENT)
├─ user_id (BIGINT, FK → USER, NOT NULL)
├─ type (ENUM: 'CHAT_MESSAGE', 'JOIN_APPROVED', 'JOIN_REJECTED', 'MEMBER_JOINED', NOT NULL)
├─ title (VARCHAR(100), NOT NULL)
├─ message (TEXT, NOT NULL)
├─ related_type (ENUM: 'MEETING', 'CHAT', NULL)
├─ related_id (BIGINT, NULL)
├─ is_read (BOOLEAN, DEFAULT false)
├─ read_at (DATETIME, NULL)
├─ created_at (DATETIME, NOT NULL, DEFAULT CURRENT_TIMESTAMP)
└─ expires_at (DATETIME, NULL)

INDEX idx_user_unread_created (user_id, is_read, created_at)
INDEX idx_expires (expires_at)

FOREIGN KEY fk_notification_user (user_id) REFERENCES USER(user_id) ON DELETE CASCADE
```

▼ 예시

- - 참가 승인 알림
  notification_id: 1
  user_id: 5
  type: "JOIN_APPROVED"
  title: "참가 승인됨"
  message: "강남역 저녁 러닝 크루 참가가 승인되었습니다"
  related_type: "MEETING"

related_id: 1
is_read: false

🎯 핵심 기능

- **사용자 관리**: 개인정보 포함 프로필 관리
- **모임 매칭**: AI 기반 위치/관심사 추천
- **실시간 채팅**: WebSocket 기반 그룹 채팅
- **좋아요 시스템**: 간단한 모임 선호도 표현
- **신고 시스템**: 사용자 대상 간소화된 신고

📊 테이블 구성 (총 11개)

1. **USER** - 사용자 기본정보 + 개인정보
2. **INTEREST** - 관심사 마스터 데이터
3. **USER_INTEREST** - 사용자별 관심사 매핑
4. **MEETING** - 모임 기본정보 + 좋아요 수
5. **MEETING_LIKE** - 모임 좋아요 기록
6. **MEETING_MEMBER** - 모임 참가자 관리
7. **MEETING_TAG** - 모임별 관심사 태그
8. **CHAT_MESSAGE** - 실시간 채팅 메시지
9. **READ_RECEIPT** - 채팅 읽음 처리
10. **REPORT** - 사용자 신고 기능
11. **NOTIFICATION** - 푸시 알림 관리

🚀 AI 추천 알고리즘

- **위치 기반** (30%): 사용자 GPS 위치와 모임 장소 거리
- **관심사 매칭** (30%): 사용자-모임 관심사 일치도
- **인기도** (40%): 모임별 좋아요 수 가중치

🔔 알림 시스템 (4가지)

- `CHAT_MESSAGE` : 새 메시지 알림

- `JOIN_APPROVED`: 모임 참가 승인 알림
- `JOIN_REJECTED`: 모임 참가 거절 알림
- `MEMBER_JOINED`: 새 멤버 참가 알림 (호스트용)

⚠️ 신고 카테고리 (3가지)

- `ABUSE`: 욕설/비방
- `INAPPROPRIATE`: 불건전한 내용
- `SPAM`: 스팸/도배