신일제약 실적관리 시스템 데이터 구조 상세 (v3.1)

문서 버전: 3.1 (2025-07-30)

1. 개요

본 문서는 신일제약 실적관리 시스템의 핵심 데이터 구조, 테이블 간의 관계, 그리고 데이터 흐름의 중심이 되는 뷰(`View`)에 대해 상세히 기술합니다. 시스템의 모든 데이터 처리는 이 구조를 기반으로 이루어지므로, 유지보수 및 기능 확장 시 반드시 본 문서를 참고해야 합니다. 데이터 모델은 **원본 데이터의 불변성**을 유지하고, **관리자의 모든 변경 행위를 추적** 가능하게 하는 것에 중점을 두고 설계되었습니다.

---

2. 데이터 흐름도 (Lifecycle of Data)

하나의 실적 데이터가 생성되어 최종 정산되기까지의 여정은 다음과 같습니다.

```
(사용자)                          (관리자)
   |                                 |
[1. 실적 입력] ------------> [2. 데이터 로드 & 동기화] -------> [3. 검수 진행 (수정/추가/삭제)] ----
   | (performance_records)       | (performance_records_absorption) | (performance_records_abs
   | - review_status: '대기'     | - pr.status -> '검수중'          | - review_action 기록
   |                             | - pra.status -> '검수중'         | - _modify, _add 필드에 값 저
   |                             | - _add 필드 -> NULL로 초기화     |
```

---

3. 테이블 관계도 (Text-based ERD)

```
[auth.users] 1--* [performance_records] 1--1 [performance_records_absorption] *--1 [companies
   |                                             |                          |
   *--1 [companies] (user_id)                    *--1 [clients]             *--1 [clients]
                                                 *--1 [products]
```

- `1--*` : 일대다 (One-to-Many)
- `1--1` : 일대일 (One-to-One)

---

4. 주요 테이블 상세 정의

4.1. `performance_records` (사용자 실적 원본 테이블)

- **역할**: 사용자가 입력하는 모든 실적 데이터의 **원본(Raw Data)** 을 저장하는 가장 기초적인 테이블입니다.

- **특징**: 이 테이블의 데이터는 **절대 직접 수정되거나 삭제되지 않습니다.**
- **컬럼 정의**:
    - `id` (bigint, PK): 레코드 고유 ID.
    - `company_id` (uuid, FK -> `companies.id`): 실적의 주체인 업체 정보.
    - `settlement_month` (varchar): 정산월 정보.
    - `prescription_month` (varchar): 처방월 정보.
    - `client_id` (bigint, FK -> `clients.id`): 관련 거래처 정보.
    - `product_id` (uuid, FK -> `products.id`): 관련 제품 정보.
    - `prescription_qty` (numeric): 처방 수량.
    - `prescription_type` (varchar): 처방 유형 ('EDI' 기본값).
    - `remarks` (text): 비고.
    - `registered_by` (uuid, FK -> `auth.users.id`): 실적을 입력한 사용자 정보.
    - `review_status` (text): 레코드의 검수 상태 ('대기', '검수중', '완료'). 데이터 흐름을 제어하는 핵심 필드.
    - `review_action` (text): 검수 작업 유형.
    - `commission_rate` (numeric): 수수료율.
    - `updated_by` (uuid, FK -> `auth.users.id`): 수정자 정보.
    - `created_at`, `updated_at` (timestamptz): 생성/수정 시각.

4.2. `performance_records_absorption` (관리자 검수 및 분석용 테이블)

- **역할**: 관리자의 모든 검수 작업(수정, 추가, 삭제)의 **결과와 이력을 기록**하는 시스템의 핵심 운영 테이블입니다.
- **컬럼 정의**:
    - `id` (bigint, PK): 검수 레코드 고유 ID.
    - `settlement_month` (varchar): 정산월.
    - `company_id` (uuid, FK -> `companies.id`): 업체 ID.
    - `client_id` (bigint, FK -> `clients.id`): 거래처 ID.
    - `product_id` (uuid, FK -> `products.id`): 제품 ID.
    - `prescription_month` (varchar): 처방월.
    - `prescription_qty` (numeric): 처방 수량.
    - `prescription_type` (varchar): 처방 유형.
    - `commission_rate` (numeric): 수수료율.
    - `remarks` (text): 비고.
    - `registered_by` (uuid, FK -> `auth.users.id`): 등록자.
    - `review_status` (text): 관리자의 검수 상태 ('검수중', '완료').
    - `review_action` (text): 관리자의 작업 유형 (`'수정'`, `'추가'`, `'삭제'`).
    - `wholesale_revenue` (numeric): 도매 매출액.
    - `direct_revenue` (numeric): 직거래 매출액.
    - `total_revenue` (numeric): 총 매출액.
    - `absorption_rate` (numeric): 흡수율.
    - `updated_by` (uuid, FK -> `auth.users.id`): 수정자.
    - `created_at`, `updated_at` (timestamptz): 생성/수정 시각.

4.3. 마스터 테이블

- **companies**:
  - id (uuid, PK), user_id (uuid, FK -> auth.users.id)
  - company_name, business_registration_number, representative_name
  - business_address, landline_phone, contact_person_name, mobile_phone, mobile_phone_2
  - email, default_commission_grade (text, 'A' 기본값), remarks
  - approval_status (text, 'pending' 기본값), status (text, 'active' 기본값)
  - user_type (text, 'user' 기본값), company_group (text)
  - assigned_pharmacist_contact, receive_email (text)
  - created_by, approved_at, updated_by (uuid, FK -> auth.users.id)
  - created_at, updated_at (timestamptz)
- **clients**:
  - id (bigint, PK), client_code (text)
  - name, business_registration_number, owner_name, address, remarks
  - status (text, 'active' 기본값)
  - created_at, updated_at (timestamptz)
- **products**:
  - id (uuid, PK), product_name, insurance_code, price (integer)
  - commission_rate_a, commission_rate_b, commission_rate_c (numeric)
  - standard_code, unit_packaging_desc, unit_quantity (integer)
  - base_month (text), status (text, 'active' 기본값)
  - created_at, updated_at (timestamptz)
- **pharmacies**:
  - id (bigint, PK), pharmacy_code (text)
  - name, business_registration_number, address, remarks
  - status (text, 'active' 기본값)
  - created_at, updated_at (timestamptz)
- **settlement_months**:
  - id (bigint, PK), settlement_month (varchar)
  - start_date, end_date (date), notice (text)
  - status (varchar, 'active' 기본값), remarks (text)
  - created_at (timestamptz)
- **settlement_share**:
  - id (bigint, PK), settlement_month (text)
  - company_id (uuid, FK -> companies.id)

- `share_enabled` (boolean), `created_at` (timestamptz)

4.4. 매핑 테이블

- **`client_company_assignments`**:
  - `id` (bigint, PK), `client_id` (bigint, FK -> `clients.id`)
  - `company_id` (uuid, FK -> `companies.id`)
  - `company_default_commission_grade` (text, 'A' 기본값)
  - `modified_commission_grade` (text)
  - `created_at` (timestamptz)
- **`client_pharmacy_assignments`**:
  - `id` (bigint, PK), `client_id` (bigint, FK -> `clients.id`)
  - `pharmacy_id` (bigint, FK -> `pharmacies.id`)
  - `created_at` (timestamptz)

4.5. 매출 데이터 테이블

- **`wholesale_sales`**:
  - `id` (bigint, PK), `pharmacy_code`, `pharmacy_name`
  - `business_registration_number`, `address`, `standard_code`
  - `product_name`, `sales_amount` (numeric), `sales_date` (date)
  - `created_at`, `updated_at` (timestamptz)
  - `created_by`, `updated_by` (text)
- **`direct_sales`**:
  - `id` (bigint, PK), `pharmacy_code`, `pharmacy_name`
  - `business_registration_number`, `address`, `standard_code`
  - `product_name`, `sales_amount` (numeric), `sales_date` (date)
  - `created_at`, `updated_at` (timestamptz)
  - `created_by`, `updated_by` (text)

4.6. 기타 테이블

- **`notices`**:
  - `id` (uuid, PK), `title`, `content`
  - `is_pinned` (boolean, false 기본값), `view_count` (integer, 0 기본값)
  - `author_id` (uuid, FK -> `auth.users.id`)
  - `file_url`, `links` (text)
  - `created_at`, `updated_at` (timestamptz)

- **`performance_evidence_files`**:
  - `id` (uuid, PK), `company_id` (uuid, FK -> `companies.id`)
  - `client_id` (bigint, FK -> `clients.id`)
  - `settlement_month` (text), `file_name`, `file_path`
  - `file_size` (bigint), `uploaded_by` (uuid, FK -> `auth.users.id`)
  - `uploaded_at`, `created_at` (timestamp)

- **`absorption_analysis`**:
  - `id` (bigint, PK), `performance_record_id` (bigint, FK -> `performance_records.id`)
  - `review_status` (text), `review_action` (text)
  - `user_edit_status` (text), `update_by` (uuid, FK -> `auth.users.id`)
  - `company_id_add`, `client_id_add`, `product_id_modify`, `prescription_month_modify`
  - `prescription_qty_modify`, `prescription_type_modify`, `remarks_modify`
  - `commission_rate_modify`, `created_add_at`, `settlement_month_add`
  - `registered_add_by` (uuid, FK -> `auth.users.id`)
  - `wholesale_revenue`, `direct_revenue`, `total_revenue`, `absorption_rate`
  - `created_at`, `updated_at` (timestamptz)

---

5. 핵심 데이터 뷰 (View): `review_details_view`

`review_details_view`는 시스템의 **데이터 조회 로직을 극적으로 단순화**시키는 가장 중요한 가상 테이블입니다.

5.1. View의 SQL 구조 (실제 코드)

```
CREATE OR REPLACE VIEW public.review_details_view AS
WITH analysis_base AS (
    SELECT
        aa.id AS absorption_analysis_id,
        aa.performance_record_id,
        aa.review_status,
        aa.review_action,
        pr.user_edit_status,
        -- 등록자 ID 결정 (관리자가 추가했으면 관리자 ID, 아니면 원본 등록자 ID)
        COALESCE(aa.registered_add_by, pr.registered_by) AS final_registered_by_id,
        aa.update_by, -- 수정자 ID
        COALESCE(aa.company_id_add, pr.company_id) AS final_company_id,
        COALESCE(aa.client_id_add, pr.client_id) AS final_client_id,
        COALESCE(aa.product_id_modify, pr.product_id) AS final_product_id,
        COALESCE(aa.prescription_month_modify, pr.prescription_month) AS prescription_month,
        COALESCE(aa.prescription_qty_modify, pr.prescription_qty) AS prescription_qty,
        COALESCE(aa.prescription_type_modify, (pr.prescription_type)::text) AS prescription_t
        COALESCE(aa.remarks_modify, pr.remarks) AS remarks,
        aa.commission_rate_modify,
        -- 날짜 결정 (관리자 추가일 우선)
```

```
        COALESCE(aa.created_add_at, pr.created_at) AS final_created_at,
        COALESCE(aa.settlement_month_add, pr.settlement_month) AS final_settlement_month,
        aa.updated_at
    FROM absorption_analysis aa
    LEFT JOIN performance_records pr ON aa.performance_record_id = pr.id
)
SELECT
    ab.absorption_analysis_id,
    ab.performance_record_id,
    ab.review_status,
    ab.review_action,
    ab.user_edit_status,
    ab.final_company_id AS company_id,
    ab.final_client_id AS client_id,
    ab.final_product_id AS product_id,
    ab.prescription_month,
    ab.prescription_qty,
    ab.prescription_type,
    ab.remarks,
    ab.final_created_at AS created_at,
    ab.final_settlement_month AS settlement_month,
    ab.updated_at,
    -- 최종 등록자 이름 결정 로직 수정
    CASE
        WHEN registrant.role = 'admin' THEN registrant.contact_person_name
        ELSE registrant.company_name
    END AS registered_by_name,
    updater.contact_person_name AS updated_by_name, -- 수정자 이름
    c.company_name,
    c.company_group,
    c.business_registration_number,
    c.representative_name,
    c.assigned_pharmacist_contact AS manager_name,
    cl.name AS client_name,
    p.product_name AS product_name_display,
    p.insurance_code,
    p.price,
    (ab.prescription_qty * p.price) AS prescription_amount,
    COALESCE(
        ab.commission_rate_modify,
        CASE
            WHEN c.default_commission_grade = 'B' THEN p.commission_rate_b
            ELSE p.commission_rate_a
        END
    ) AS commission_rate
FROM analysis_base ab
-- 이름 조인을 위한 조인
LEFT JOIN companies registrant ON ab.final_registered_by_id = registrant.user_id
LEFT JOIN companies updater ON ab.update_by = updater.user_id
-- 나머지 데이터 조인
LEFT JOIN companies c ON ab.final_company_id = c.id
LEFT JOIN clients cl ON ab.final_client_id = cl.id
LEFT JOIN products p ON ab.final_product_id = p.id;
```

5.2. `COALESCE` 로직의 중요성

`COALESCE` 는 "NULL이 아닌 첫 번째 값을 반환하라"는 의미의 SQL 함수입니다. 이 View는 `COALESCE` 를 적극적으로 사용하여, 관리자의 수정/추가 이력을 종합하고 **최종적으로 사용해야 할 데이터를 동적으로 결정**합니다.

- **예시 시나리오**:
    1. **원본 데이터**: `performance_records` 에 `prescription_qty = 100` 인 데이터가 있음.
    2. **수정 발생**: 관리자가 이 데이터를 120으로 수정하면, `performance_records_absorption` 에 `prescription_qty_modify = 120` 이 저장됨.
    3. **View 조회**: `COALESCE(aa.prescription_qty_modify, pr.prescription_qty)` 는 `aa.prescription_qty_modify` 가 `NULL` 이 아니므로 **120**을 반환함.
    4. **수정 미발생**: 만약 수정되지 않았다면, `aa.prescription_qty_modify` 는 `NULL` 이므로, `COALESCE` 는 두 번째 값인 `pr.prescription_qty` 즉, **100**을 반환함.
- **기대 효과**: 이 패턴 덕분에 프론트엔드는 데이터의 수정 여부나 추가 여부를 전혀 신경 쓸 필요 없이, `review_details_view` 의 최종 필드(`prescription_qty` , `company_id` 등)만 조회하면 **항상 올바른 최신 데이터**를 얻을 수 있습니다. 이는 코드의 가독성과 유지보수성을 극대화합니다.

---

7. 최신 추가 테이블 및 기능

7.1. 실적 증빙 파일 관리 (`performance_evidence_files`)

목적: 실적 데이터의 증빙 자료를 안전하게 저장하고 관리

컬럼 정의:

- `id` (uuid, PK): 파일 고유 ID
- `company_id` (uuid, FK -> `companies.id`): 업체 ID
- `client_id` (bigint, FK -> `clients.id`): 거래처 ID
- `settlement_month` (text): 정산월
- `file_name` (text): 원본 파일명
- `file_path` (text): Supabase Storage 내 파일 경로
- `file_size` (bigint): 파일 크기 (bytes)
- `uploaded_by` (uuid, FK -> `auth.users.id`): 업로드한 사용자
- `uploaded_at` (timestamp): 업로드 시각
- `created_at` (timestamp): 생성 시각

특징:

- Supabase Storage를 활용한 안전한 파일 저장
- 업체별 접근 권한 제어 (RLS 정책)
- 파일 메타데이터 추적으로 관리 효율성 향상

6.2. 실적입력기간 관리 (`settlement_months` 확장)

기존 컬럼에 추가된 기능:

- `start_date` (date): 실적입력 시작일
- `end_date` (date): 실적입력 종료일
- `notice` (text): 전달 사항
- `status` (varchar, 'active' 기본값): 활성/비활성 상태

핵심 기능:

- **일반 사용자**: 실적입력기간 내에서만 실적 등록/수정 가능
- **관리자**: 실적입력기간과 무관하게 언제든지 실적 등록/수정 가능
- **UI 제어**: 기간 외에는 모든 입력 버튼 비활성화

6.3. 사용자 편집 상태 추적 (`performance_records` 확장)

추가된 컬럼:

- `user_edit_status` (text): 사용자 편집 상태
- `updated_by` (uuid, FK -> `auth.users.id`): 수정자 정보
- `updated_at` (timestamptz): 수정 시각

상태 값:

- `'editable'` : 편집 가능
- `'locked'` : 편집 불가 (검수중/완료)
- `'pending'` : 대기 중

6.4. 제품 정보 월별 관리 (`products` 확장)

핵심 기능:

- **월별 제품 정보**: `base_month` 필드로 월별 제품 정보 관리
- **보험코드 유니크**: 같은 보험코드의 제품은 월별로 하나만 유지
- **자동 업데이트**: 처방월 변경 시 제품 정보 자동 업데이트

데이터 무결성:

- 보험코드 기준 유니크 처리로 중복 방지
- 월별 제품 정보 캐싱으로 성능 최적화
- 제품 정보 변경 시 실적 데이터 자동 반영

6.5. 키보드 네비게이션 상태 관리

프론트엔드 상태 변수:

- `currentCell` (ref): 현재 포커스된 셀 정보 `{ row: number, col: string }`
- `fieldRefs` (ref): 각 입력 필드의 DOM 참조
- `productInputRefs` (ref): 제품명 입력 필드의 DOM 참조

키보드 이벤트 처리:

- **전역 이벤트**: document 레벨에서 키보드 이벤트 처리
- **조건부 처리**: 편집 가능한 상태에서만 키보드 기능 활성화
- **충돌 방지**: 제품 검색 중 특정 키 차단

6.6. 제품 검색 시스템 상태 관리

상태 변수:

- `productSearchForRow` (ref): 제품 검색 상태
  - `query` : 검색 쿼리
  - `results` : 검색 결과 배열
  - `selectedIndex` : 선택된 항목 인덱스
  - `show` : 드롭다운 표시 여부
  - `activeRowIndex` : 활성 행 인덱스

캐싱 시스템:

- `productsByMonth` (ref): 월별 제품 목록 캐시
- `productDropdownStyle` (ref): 드롭다운 위치 스타일

6.7. 실시간 검증 시스템

입력 검증 규칙:

- **제품명**: 제품 선택 필수 (자동완성 지원)
- **처방수량**: 숫자만 입력 가능, 0보다 큰 값
- **처방액**: 자동 계산, 수정 불가
- **필수 필드**: 제품명 + 처방수량 조합 필수

실시간 피드백:

- **처방액 계산**: 수량 입력 시 즉시 계산
- **제품 정보 업데이트**: 처방월 변경 시 자동 업데이트
- **상태 표시**: 검수상태별 색상 구분

6.8. 정렬 규칙 시스템

사용자별 정렬 규칙:

- **일반 사용자**: 처방건수 0건 → 1건 이상 (입력된 것이 아래로)
- **관리자**: 처방건수 0건 → 1건 이상 (입력된 것이 위로)

- **실적 검수**: 검수상태별 정렬 (신규 → 검수중 → 완료)

정렬 로직:

- **가나다순**: `localeCompare()` 메서드 사용
- **상태별 그룹화**: 조건에 따라 그룹별로 분리 후 정렬
- **실시간 정렬**: computed 속성으로 실시간 정렬

6.9. 상태별 편집 권한 시스템

권한 규칙:

- **일반 사용자**: '대기' 상태만 편집 가능
- **관리자**: 모든 상태 편집 가능
- **상태별 UI**: 편집 불가 상태의 시각적 표시

구현 방식:

- `isRowEditable()` 함수로 행별 편집 가능 여부 판단
- `isInputEnabled` ref로 전체 편집 상태 제어
- 키보드 이벤트 조건부 처리

6.10. 성능 최적화 시스템

제품 데이터 캐싱:

- **월별 캐시**: `productsByMonth` 객체로 월별 제품 목록 캐시
- **중복 제거**: 보험코드 기준 유니크 처리
- **지연 로딩**: 필요한 월의 제품만 로드

키보드 이벤트 최적화:

- **전역 이벤트**: document 레벨에서 키보드 이벤트 처리
- **조건부 처리**: 편집 가능한 상태에서만 키보드 기능 활성화
- **충돌 방지**: 제품 검색 중 특정 키 차단

---

7. 데이터 흐름 최적화

7.1. 제품 정보 자동 업데이트 흐름

```
1. 사용자가 처방월 변경
   ↓
2. 기존 제품의 보험코드로 새 처방월에서 검색
   ↓
3. 검색 결과에 따라 분기:
   - 결과 있음: 새 제품 정보로 자동 업데이트
```

```
   - 결과 없음: 제품 선택 해제
   ↓
4. 처방수량이 있으면 처방액 재계산
```

7.2. 키보드 네비게이션 흐름

```
1. 사용자가 키보드 입력
   ↓
2. 전역 키보드 이벤트 핸들러에서 처리
   ↓
3. 편집 가능 여부 확인
   ↓
4. 제품 검색 상태 확인
   ↓
5. 적절한 동작 실행:
   - 화살표 키: 셀 간 이동
   - Enter 키: 필드별 특별 동작
   - 특수 키: 행 관리 동작
```

7.3. 실시간 검증 흐름

```
1. 사용자가 데이터 입력
   ↓
2. 입력값 실시간 검증
   ↓
3. 검증 결과에 따라 피드백 제공:
   - 성공: 정상 처리
   - 실패: 오류 메시지 표시
   ↓
4. 관련 필드 자동 업데이트 (처방액 계산 등)
```

7.4. 상태별 편집 권한 흐름

```
1. 화면 로드 시 사용자 권한 확인
   ↓
2. 실적 데이터의 검수 상태 확인
   ↓
3. 권한과 상태에 따라 편집 가능 여부 결정
   ↓
4. UI 요소 활성화/비활성화
   ↓
5. 키보드 이벤트 조건부 처리
```

이러한 최신 기능들을 통해 신일 PMS는 더욱 효율적이고 사용자 친화적인 실적 관리 시스템으로 발전했습니다. 모든 기능은 데이터 무결성을 보장하면서 사용자 경험을 최적화하는 방향으로 설계되었습니다.