Smart-Settlement

대용량 트랜잭션 기반 비동기 정산 및 통계 플랫폼

Tech Stack

Category	Technology Stack
Backend	Java17, Spring Boot3.3, Spring Batch5.x, QueryDSL
Database	PostgreSQL 15, Redis (Redisson)
Infra/DevOps	Docker, GitHub Actions, JUnit5

프로젝트 기간 202601~202605

작성자 조수진
7년 자 백엔드 엔지니어
GitHub https://github.com/SUJINCHO
Email publiccho98@gmail.com

Smart-Settlement	Document	작성자	작성 날짜	수정자	수정 날짜
	개정 이력	조수진	26.01.01	조수진	26.01.10

버전	개정 일자	대상 시트	개정 내용 요약	개정자
v1.0	26.01.02	요구사항	요구사항 정의	조수진
v1.1	26.01.03	시스템 아키텍처	초안 시스템 아키텍처 설계 수립	조수진
v1.2	26.01.04	시스템 아키텍처	컴포넌트 간 데이터 흐름(Data Flow)을 구체화	조수진
v1.3	26.01.06	기술 의사 결정서	Redis(Redisson) 분산 락 전략 구체화 및 기술 의사결정 보강	조수진
v1.4	26.01.07	데이터 모델링 (ERD) 테이블 상세설계	데이터 모델링(ERD) 초안 작성 및 테이블 리스트 정의	조수진
v1.5	26.01.08	데이터 모델링 (ERD) 테이블 상세설계	테이블 상세설계 인덱스(Index) 전략 및 Default 제약조건 추가	조수진
v1.6	26.01.10	배치 프로세스	배치 프로세스 상세 흐름도(Step-by-Step) 및 예외 처리 로직 확정	조수진

Smart-Settlement	Document	작성자	작성 날짜	수정자	수정 날짜
	요구사항	조수진	26.01.02		

1. 개요

본 프로젝트는 매일 발생하는 수백만 건의 주문 데이터를 분석하여 판매자별 정산금을 산출하고, 대시보드용 통계 데이터를 생성하는 백엔드 중심의 엔터프라이즈 시스템입니다.
기존의 동기식 처리 방식이 대규모 트랜잭션 발생 시 시스템 부하를 초래하고 정합성 문제가 생기는 한계를 극복하기 위해, Spring Batch를 통한 배치 처리와 Redis를 활용한 분산 환경 동시성 제어를 구현하는 것을 목표로 합니다.

2. 기능 요구사항

사용자가 시스템을 통해 얻고자 하는 핵심 비즈니스 로직입니다.

ID	요구사항	상세	우선순위
F-001	일일 정산 자동화	매일 정해진 시간에 전일 주문 완료(PAID) 건을 수집하여 정산 데이터를 자동 생성	상
F-002	가변 수수료 계산	상품 마스터(product)에 정의된 수수료율(fee_rate)을 실시간 참조하여 주문별 수수료 및 실지급액을 산출	상
F-003	정산 상태 관리	정산이 완료된 주문 건의 상태를 PAID에서 SETTLED로 변경하여 중복 처리를 방지	상
F-004	수동 정산 실행 API	배치 장애나 데이터 보정 필요 시, 관리자가 특정 날짜를 지정하여 정산을 재수행할 수 있는 인터페이스 제공	중
F-005	배치 실행 이력 관리	각 배치 작업의 시작/종료 시간, 처리 건수, 성공/실패 여부를 settlement_log에 기록	상

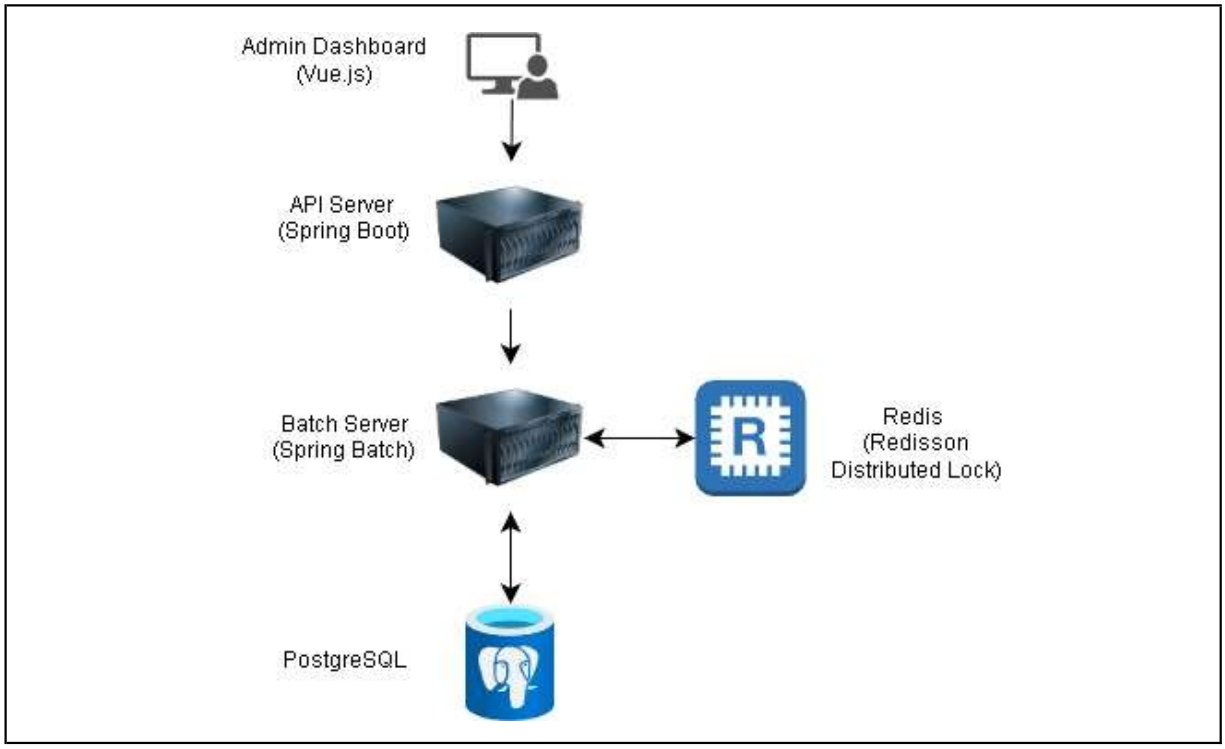
3. 비기능 요구사항

시스템의 품질과 안정성을 결정합니다.

ID	요구사항	상세	우선순위
N-001	데이터 정합성 보장	분산 서버 환경에서 동일 날짜/주문에 대한 중복 정산이 발생하지 않도록 Redis 분산 락(Redisson)을 적용	상
N-002	대용량 처리 성능	일일 100만 건 이상의 트랜잭션을 30분 이내에 처리할 수 있도록 Chunk 기반 페이징 처리를 수행	중
N-003	장애 복구 (Restart)	배치 중단 시 전체 롤백이 아닌, 실패한 지점(Chunk 단위)부터 이어서 처리할 수 있는 Restartability를 보장	상
N-004	정밀도 유지	소수점 연산 오차 방지를 위해 모든 금액 및 수수료 계산에는 Numeric 타입을 사용하고 부동 소수점 오차를 원천 차단	상
N-005	가용성	API 서버와 Batch 서버를 분리하여, 대량 정산 작업 중에도 관리자 대시보드 조회 성능 확보	중

Smart-Settlement	Document	작성자	작성 날짜	수정자	수정 날짜
	시스템 아키텍처	조수진	26.01.03	조수진	26.01.04

1. 아키텍처 다이어그램



2. 시스템 구성 개요

구성 요소	기술 스택	주요 역할 및 설계 의도
Admin API	Spring Boot 3.3	분산 환경에서 동일 정산 대상에 대한 중복 처리 방지 및 정합성 보장
Batch Server	Spring Batch 5.x	대량 주문 데이터의 정산 로직 수행. API 서버와 분리하여 성능 간섭 최소화
Distributed Lock	Redis (Redisson)	분산 환경에서 동일 정산 대상에 대한 중복 처리 방지 및 정합성 보장
Database	PostgreSQL 15	주문 원장 및 정산 결과 저장. 대량 조회를 위한 인덱스 최적화 적용

3. 핵심 데이터 흐름 (Data Flow)

Trigger: 스케줄러(Cron) 또는 관리자 API 호출을 통해 DailySettlementJob 시작
Locking: Redis를 사용하여 해당 날짜/작업에 대한 분산 락 획득 (중복 실행 방지)
Read: PostgreSQL에서 정산 대상 주문(PAID 상태) 데이터를 Chunk 단위로 조회
Process: 상품별 수수료율을 적용하여 정산 금액 산출 및 검증
Write: 정산 결과 저장 및 주문 상태 변경(SETTLED), 실행 로그 기록
Unlock: 작업 완료 후 Redis 락 해제

Smart-Settlement	Document	작성자	작성 날짜	수정자	수정 날짜
	기술 의사결정서	조수진	26.01.06	조수진	26.01.07

1. 개요

본 섹션은 프로젝트 설계 과정에서 직면한 주요 기술적 과제와 이를 해결하기 위한 의사결정 과정을 기록합니다. 시니어 엔지니어로서 기술 선택의 근거와 트레이드오프(Trade-off)를 명확히 하는 데 목적이 있습니다.

2. 기술 의사결정 상세 (Decision Log)

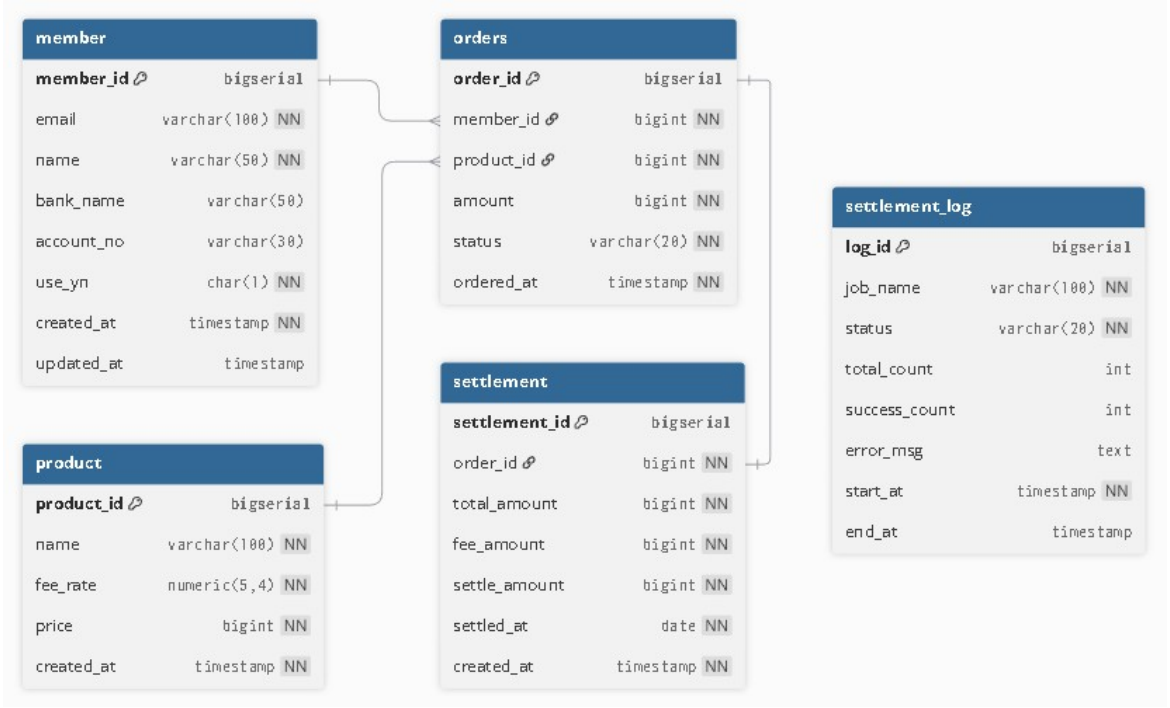
ID	항목	기술 선택	비교 대안	결정 사유 및 기대 효과
TD-01	Batch Framework	Spring Batch 5.x	Quartz, Custom Scheduler	- 대용량 데이터 처리에 최적화된 Chunk 지향 프로세싱 지원 - 실패 지점부터 재시작 가능한 Restartability 및 상태 관리 강점 - 트랜잭션 관리 및 다양한 ItemReader/Writer 제공으로 개발 생산성 향상 -Chunk 프로세싱 도입을 통해 단일 트랜잭션 부하를 방지하고, 일일 100만 건 기준 30분 이내 처리 성능 확보 목적
TD-02	Concurrency Control	Redis (Redisson)	DB Pessimistic Lock (Select for update)	-분산 서버 환경에서 가벼운 락 메커니즘 제공으로 DB 부하 경감 - Lettuce 대비 Redisson은 스핀 락(Spin Lock)을 사용하지 않고 Pub/Sub 기반으로 락 획득을 대기하므로 Redis 부하 경감 - Redisson의 Pub/Sub 기반 락 획득 방식으로 Spin Lock 대비 네트워크 자원 절약 - 정산 중복 실행 방지를 위한 높은 응답성 확보
TD-03	Data Access	QueryDSL	Mybatis, Native JPA	- 컴파일 시점에 쿼리 오류를 발견할 수 있는 타입 안정성 확보 - 정산 통계 등 복잡한 동적 쿼리를 자바 코드로 가독성 있게 구현 가능 - IDE 지원을 통한 쿼리 작성 속도 및 유지보수성 향상
TD-04	Database	PostgreSQL	MySQL (MariaDB)	- 복잡한 분석 쿼리 및 대용량 데이터 처리에 최적화된 Query Optimizer 기능 - 정산 금액 연산의 정확도를 위한 풍부한 수치형 데이터 타입(Numeric) 지원 - MVCC 모델을 통한 동시성 제어 효율성 및 확장성 고려

3. 리스크 관리

대상	잠재적 리스크	대응 방안
Redis	인프라 관리 포인트증가	금융 정합성 확보를 최우선 가치로 선정
QueryDSL	초기 설정 복잡 및 학습 곡선	런타임 에러 방지 및 유지보수 용이

Smart-Settlement	Document	작성자	작성 날짜	수정자	수정 날짜
	데이터 모델링 (ERD)	조수진	26.01.07	조수진	26.01.08

1. 엔티티 관계도 (ERD)



2. 엔티티 목록

물리명	논리명	상세
member	판매자/회원 정보	정산 대상을 식별
product	상품 정보	수수료율 등 기준 정보 포함
orders	주문 원장	정산의 대상이 되는 데이터
settlement	정산 결과	배치가 계산해서 저장하는 결과
settlement_log	정산 실행 로그	배치 작업 성공/실패 기록

3. 데이터 모델링 설계 주안점

항목	설계 전략	기대 효과
Data Integrity	금액 필드(amount)에 bigint 타입 적용	부동 소수점 오차 방지 및 대용량 연산 시 CPU 효율 극대화
Performance	orders.status 인덱스 설계 (예정)	정산 대상(PAID) 데이터 추출 시 Full Scan 방지 및 처리 속도 향상
Traceability	settlement_log 독립 설계	배치 작업 단위의 성공/실패 이력을 체계적으로 관리하여 장애 대응력 강화
Accuracy	fee_rate에 고정 소수점 타입 적용	소수점 4자리까지의 정밀한 수수료 산출로 정산 신뢰도 확보

Smart-Settlement	Document	작성자	작성 날짜	수정자	수정 날짜
	테이블 상세설계	조수진	26.01.07	조수진	26.01.08

1. member (판매자/회원 정보)

판매자 정보를 관리하며, 정산금 지급을 위한 결제 수단 정보를 포함합니다.

컬럼명	타입	Null 허용	제약 조건	설명
member_id	BIGSERIAL	NO	PK	회원 고유 번호 (자동 증가)
email	VARCHAR(100)	NO	UNIQUE	회원 이메일 (로그인 ID 겸용)
name	VARCHAR(50)	NO		이름 또는 상호명
bank_name	VARCHAR(50)	YES		정산 계좌 은행명
account_no	VARCHAR(30)	YES		정산 계좌 번호
use_yn	CHAR(1)	NO	DEFAULT 'Y'	활성 여부 (Y/N)
created_at	TIMESTAMP	NO	DEFAULT NOW()	등록 일시
updated_at	TIMESTAMP	YES		수정 일시

2. product (상품 정보)

정산 시 가장 중요한 수수료율을 결정하는 기준 정보 테이블입니다.

컬럼명	타입	Null 허용	제약 조건	설명
product_id	BIGSERIAL	NO	PK	상품 고유 번호
name	VARCHAR(100)	NO		상품 이름
fee_rate	NUMERIC(5,4)	NO	DEFAULT 0	수수료율 (예: 0.0300 = 3%)
price	BIGINT	NO		상품 기본 단가
created_at	TIMESTAMP	NO	DEFAULT NOW()	등록 일시

3. orders (주문 원장)

정산 배치가 읽어 들일 원천 데이터입니다.

컬럼명	타입	Null 허용	제약 조건	설명
order_id	BIGSERIAL	NO	PK	주문 고유 번호
member_id	BIGINT	NO	FK (member)	판매자(회원) 번호
product_id	BIGINT	NO	FK (product)	주문 상품 번호
amount	BIGINT	NO		최종 결제 금액
status	VARCHAR(20)	NO		주문 상태 (PAID, SETTLED)
ordered_at	TIMESTAMP	NO	INDEX (status, ordered_at)	주문 일시 (배치 조회 기준)

4. settlement (정산 결과)

배치가 실행된 후 최종 정산 금액을 저장하는 테이블입니다.

컬럼명	타입	Null 허용	제약 조건	설명
settlement_id	BIGSERIAL	NO	PK	정산 결과 번호
order_id	BIGINT	NO	FK (orders)	대상 주문 번호 (1:1 연결)
total_amount	BIGINT	NO		주문 총액
fee_amount	BIGINT	NO		계산된 수수료 총액
settle_amount	BIGINT	NO		실 지급액 (총액 - 수수료)
settle_date	DATE	NO	INDEX	정산 대상 일자
created_at	TIMESTAMP	NO	DEFAULT NOW()	정산 처리 일시

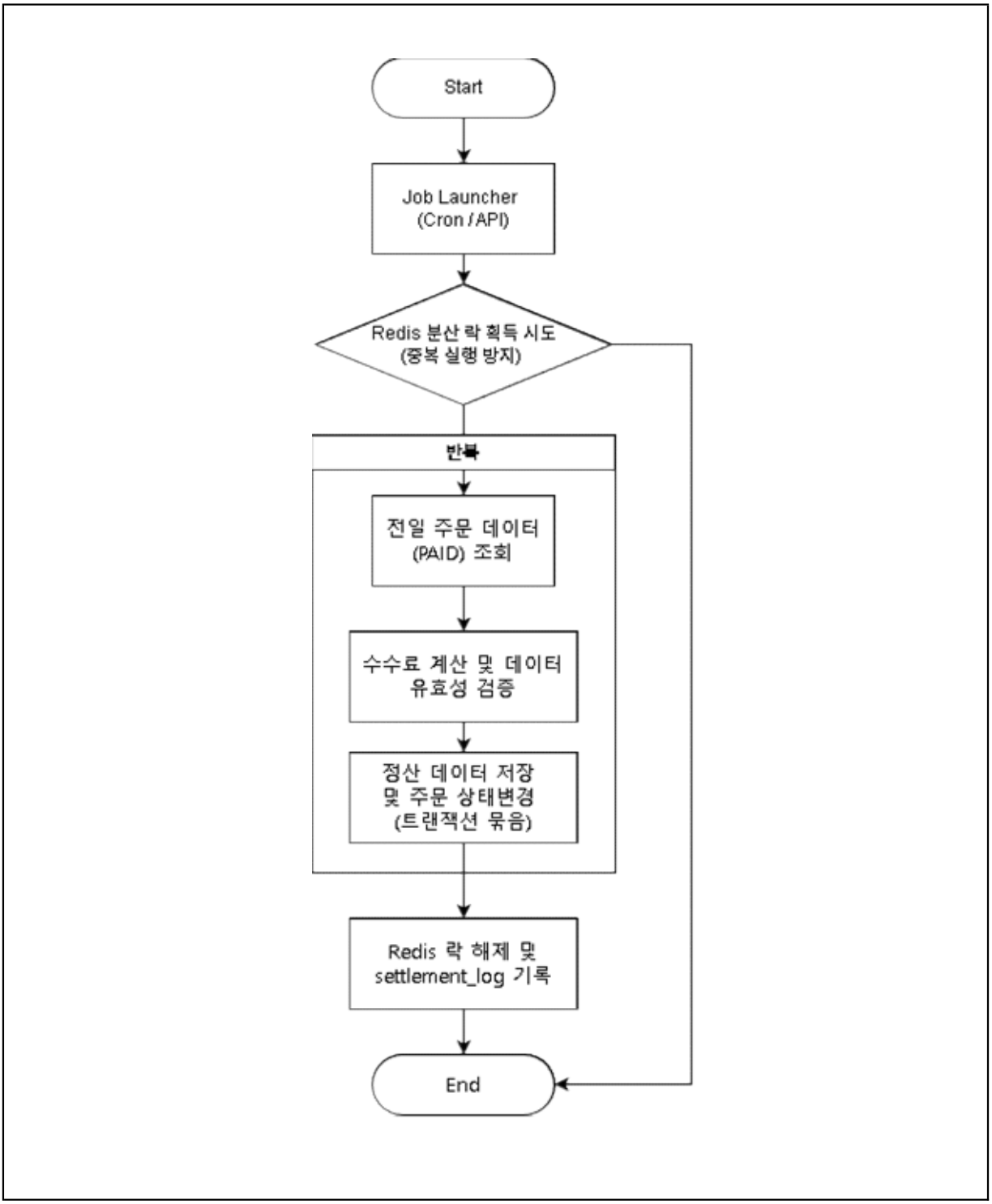
5. settlement_log (정산 실행 로그)
운영 단계에서 병목 구간 파악 및 로그 분석을 위한 테이블입니다.

컬럼명	타입	Null 허용	제약 조건	설명
log_id	BIGSERIAL	NO	PK	로그 고유 번호
job_name	VARCHAR(100)	NO		실행 작업 명칭
status	VARCHAR(20)	NO		결과 (SUCCESS, FAIL)
total_count	INT	YES		총 처리 대상 건수
success_count	INT	YES		성공 건수
error_msg	TEXT	YES		실패 시 에러 메시지 상세
start_at	TIMESTAMP	NO		배치 시작 일시
end_at	TIMESTAMP	YES		배치 종료 일시

Smart-Settlement	Document	작성자	작성 날짜	수정자	수정 날짜
	배치 프로세스	조수진	26.01.10		

1. 프로세스 개요 (Process Summary)
본 배치는 일 단위 주문 데이터를 수집하여 정산금을 산출하고, 결과 저장 및 실행 로그를 기록하는 **Chunk 기반 프로세스**로 설계되었습니다. 장애 발생 시 실패한 지점부터 재시작(Restartability)이 가능하도록 구성합니다.

2. 배치 프로세스 순서도



3. 단계별 상세 흐름 (Step-by-Step Flow)

단계	작업 명칭	상세 로직 및 처리 내용	비고
01. Start	Job 시작	스케줄러(Cron)에 의한 Job 인스턴스 생성 및 파라미터 검증	중복 실행 방지 락 체크
02. Read	주문 데이터 조회	Orders 테이블에서 status='PAID' 이고 전일 날짜인 데이터를 Chunk 단위(예: 1,000건)로 조회	페이징 처리 (PageSize 고정)
03. Process	정산금 산출	1. Product 테이블의 fee_rate 참조 2. 수수료 및 실지급액 계산 (BigInteger 연산) 3. 데이터 정합성 검증	계산 오류 시 해당 건 Skip/Log
04. Write	결과 저장	1. Settlement 테이블에 결과 Insert 2. Orders 테이블 상태 변경 (PAID → SETTLED)	하나의 트랜잭션으로 처리
05. End	Job 완료 및 로그	Settlement_log에 최종 처리 건수, 성공 여부, 소요 시간 기록 및 리소스 정리	실패 시 StackTrace 기록