

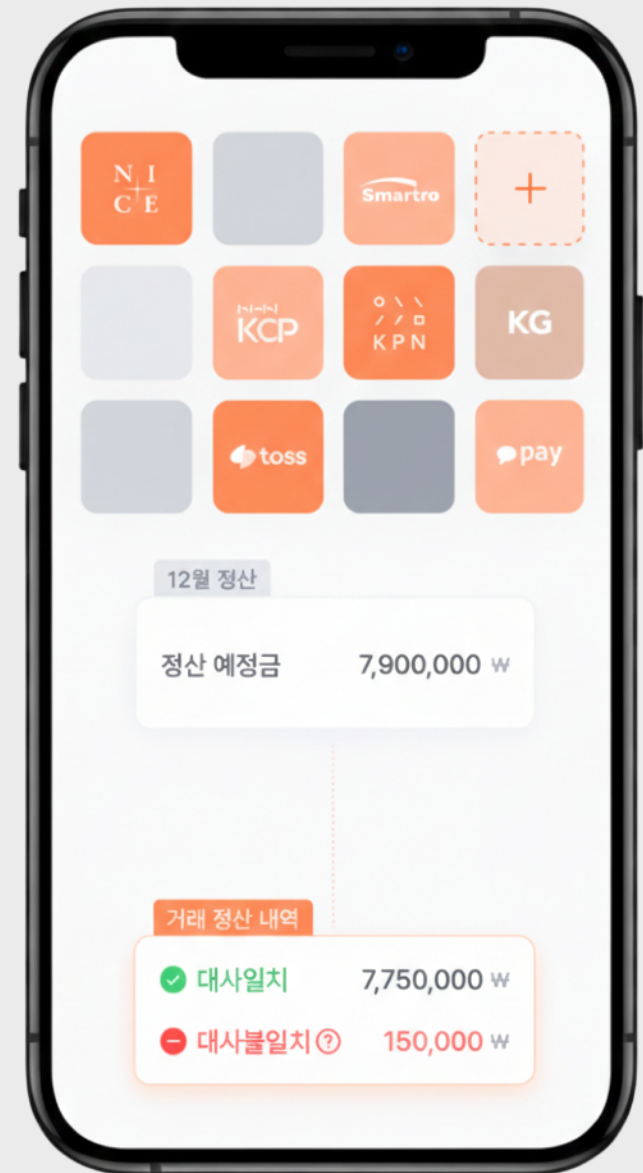
# settle-up

자산의 투명함을 세우다

🛡️ Safe-Deposit (자금 관리)







⚡ Auto-Settle (자동 정산)

👁️ Real-Check (실시간 대조)



# 팀원소개

## flow 팀원 소개

김진혁	박하얀	양준석	이진	한규진	황희수
					
GITHUB	GITHUB	GITHUB	GITHUB	GITHUB	GITHUB

# 배경

경제 경제일반

## ‘티메프 사태’ 14개월, 미정산에 빛 떠안아...“누가 죽어야 눈길 줄지”

위메프 기업회생절차 폐지로  
피해자들 구제받을 길 사라져  
티몬 매각 채권변제율 0.75%  
그마저 은행이 선순위로 받아  
피해자들, 불안·우울증 시달려

서혜미, 이주빈 기자

수정 2025-09-17 10:18    등록 2025-09-17 05:00

아직 끝나지 않은 티메프 사태, 그리고 피해 뿐인 셀러들

# 배경

조선경제 > 경제 일반

## [단독] 이커머스, 판매대금 20일 내 주고 50%는 은행에 넣어야

티메프 사태 재발 방지책 마련

권순완 기자

업데이트 2024.10.10. 09:49 ∨

가

티메프 사태 이후 플랫폼/PG사 지급 정산 대금 별도 분리 법제화 추진

# 배경

## 중개 플랫폼 판매 대금 규제 강화안

	현행	강화안
정산 기한	의무 규정 없음	구매 확정일로 부터 20일
대금 별도 보관	의무 규정 없음	금융기관 등에 50% 예치

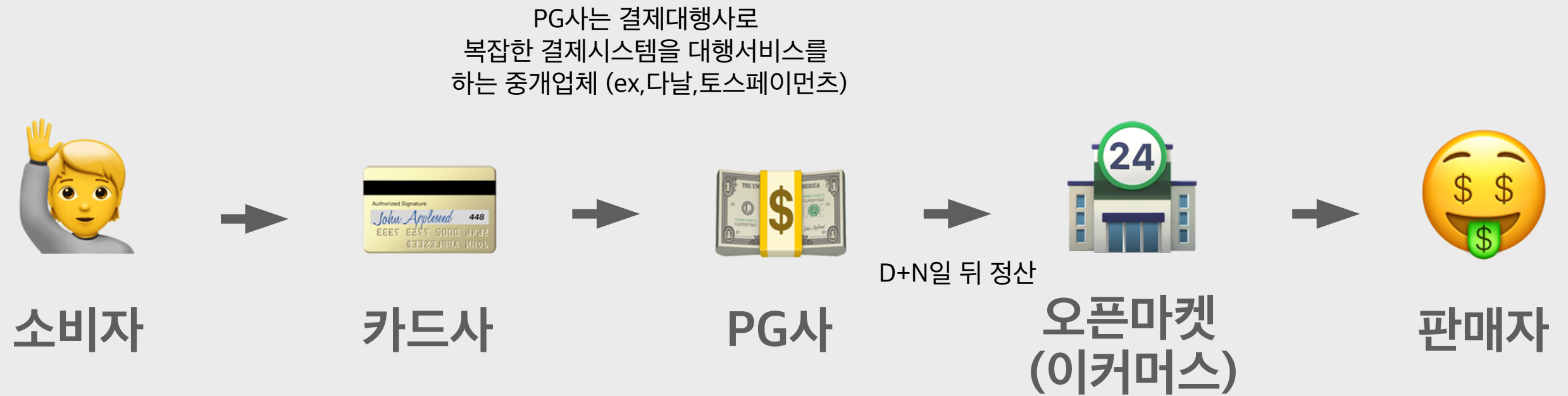
## 티몬·위메프 미정산 사태 피해 규모

미정산 피해 업체 수	4만8124곳
피해 금액 1억원 이상 업체	981곳
미정산 총규모	1조2789억원

자료=금융감독원

- ✓ 현행과 다르게 정산 기한 및 대금 별도 보관이라는 추가 규제 생성
- ✓ 자금을 더 투명하게 관리하고 이를 확인해야 할 서비스의 필요성을 느낌

# 서비스 소개



우리가 오픈마켓에서 구매한 대금은  
바로 판매자들에게 가지 않는다

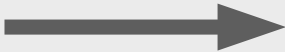
단순하지 않은 결제-지급 구조와  
헷갈리는 개념들이 있다.

# 서비스 소개



PG사

대금지급



오픈마켓

대금지급



판매자

예치 의무화



은행

대금지급



하지만 셀러(판매자) 입장에서는  
정산되서 온 금액만 받을 뿐  
어떻게 계산이되어서 언제 받는지는 모른다.

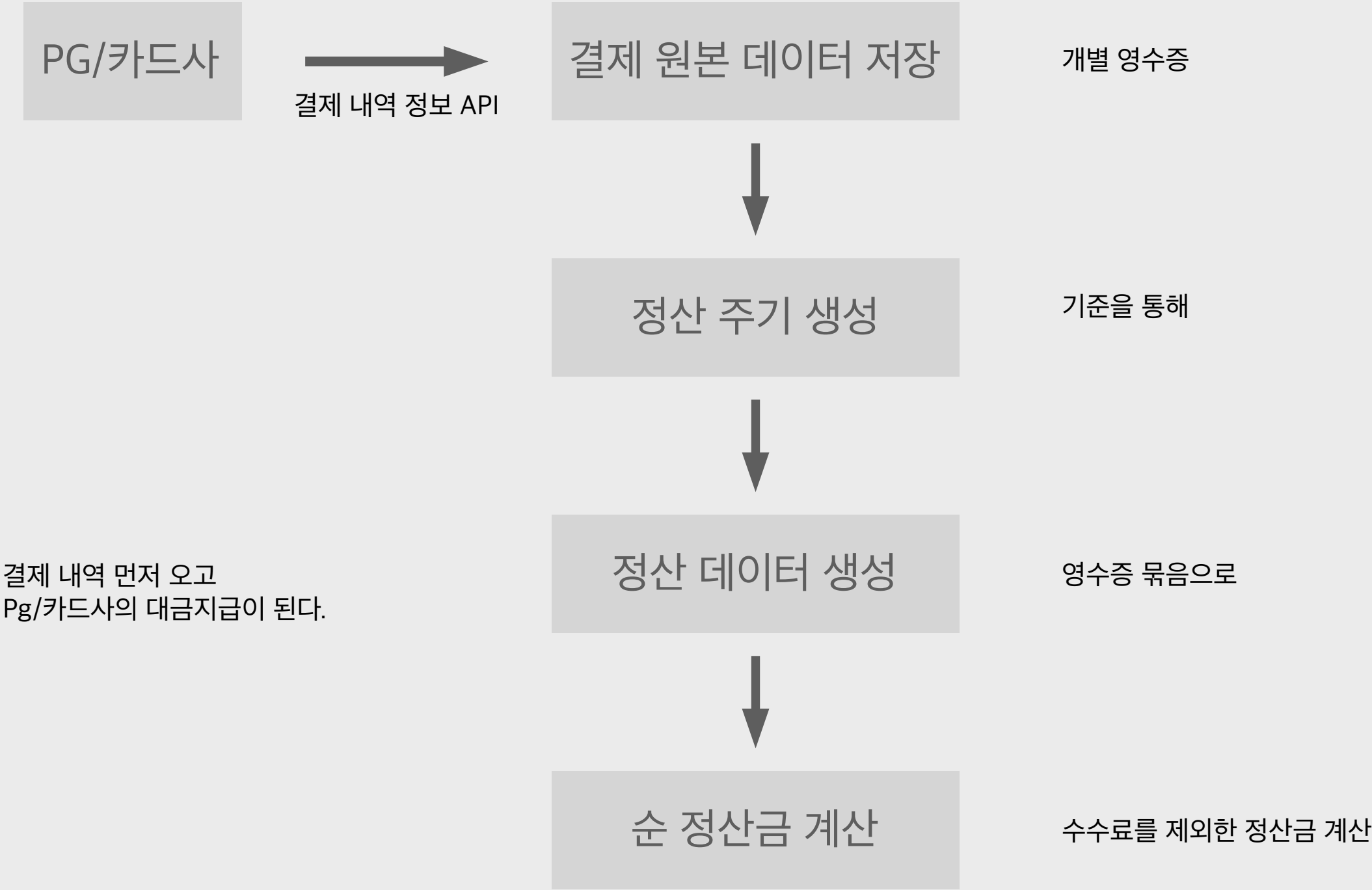
# 서비스 소개

## ◎ 서비스 목표 : 자금 분리와 투명한 기록

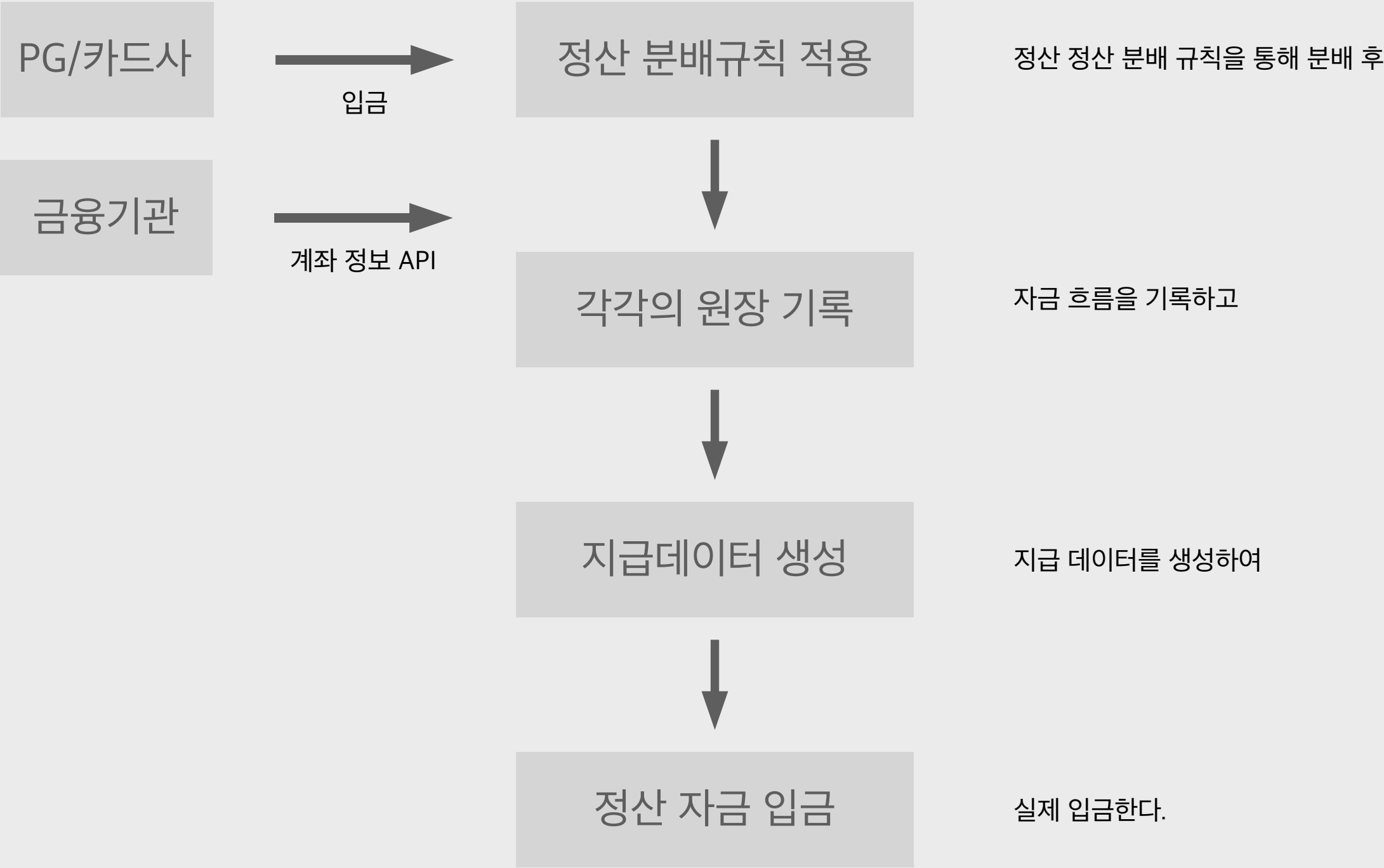
정산 대금이 운영 자금과 분리되어 안전하게 관리 될 수 있는 정산 아키텍처를 구축하고,  
결제부터 입금까지의 전 과정을 데이터로 투명하게 기록, 검증할 수 있는 B2B 정산 관리 환경 제공



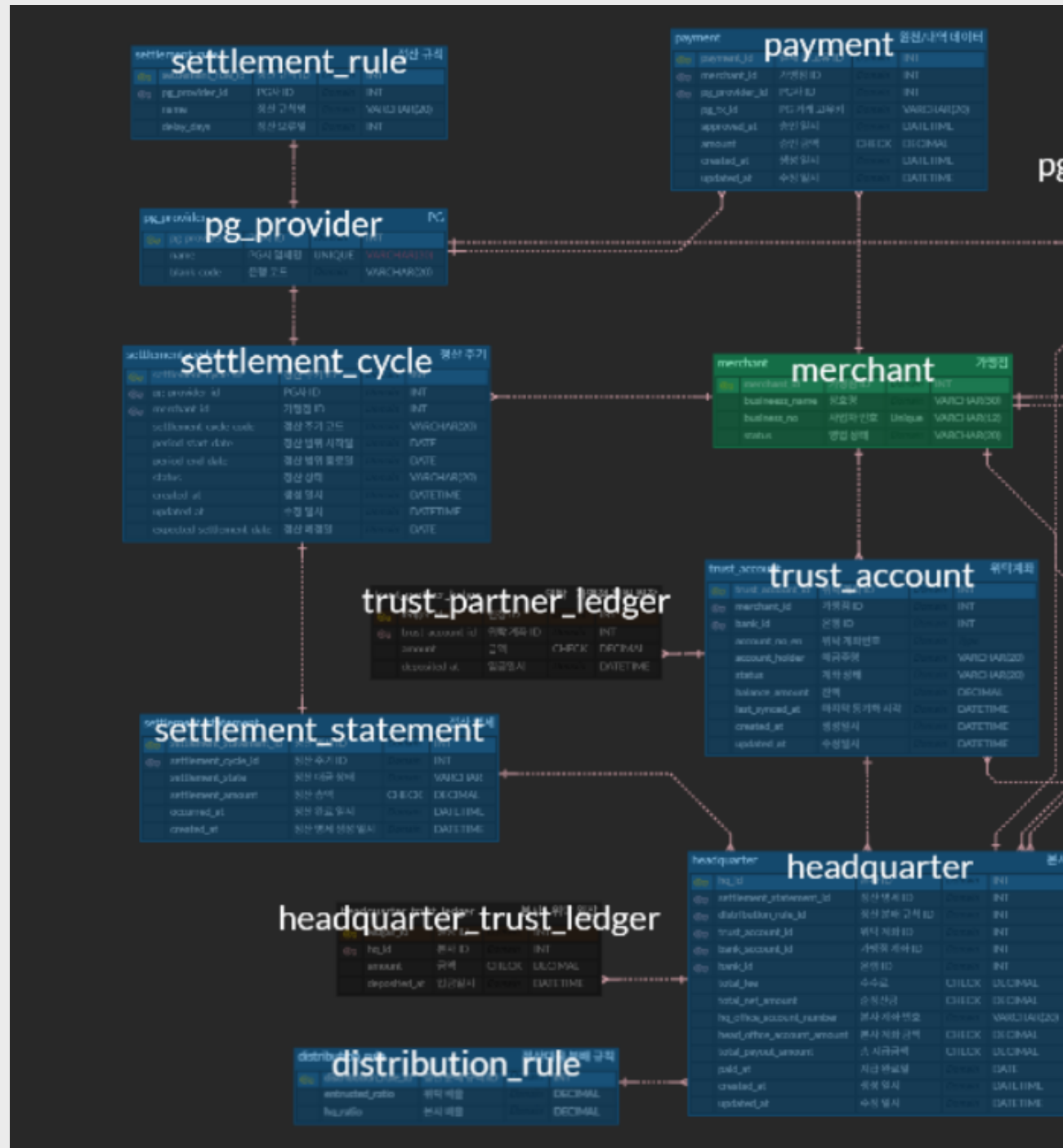
# 흐름도



# 흐름도



# 주요 테이블

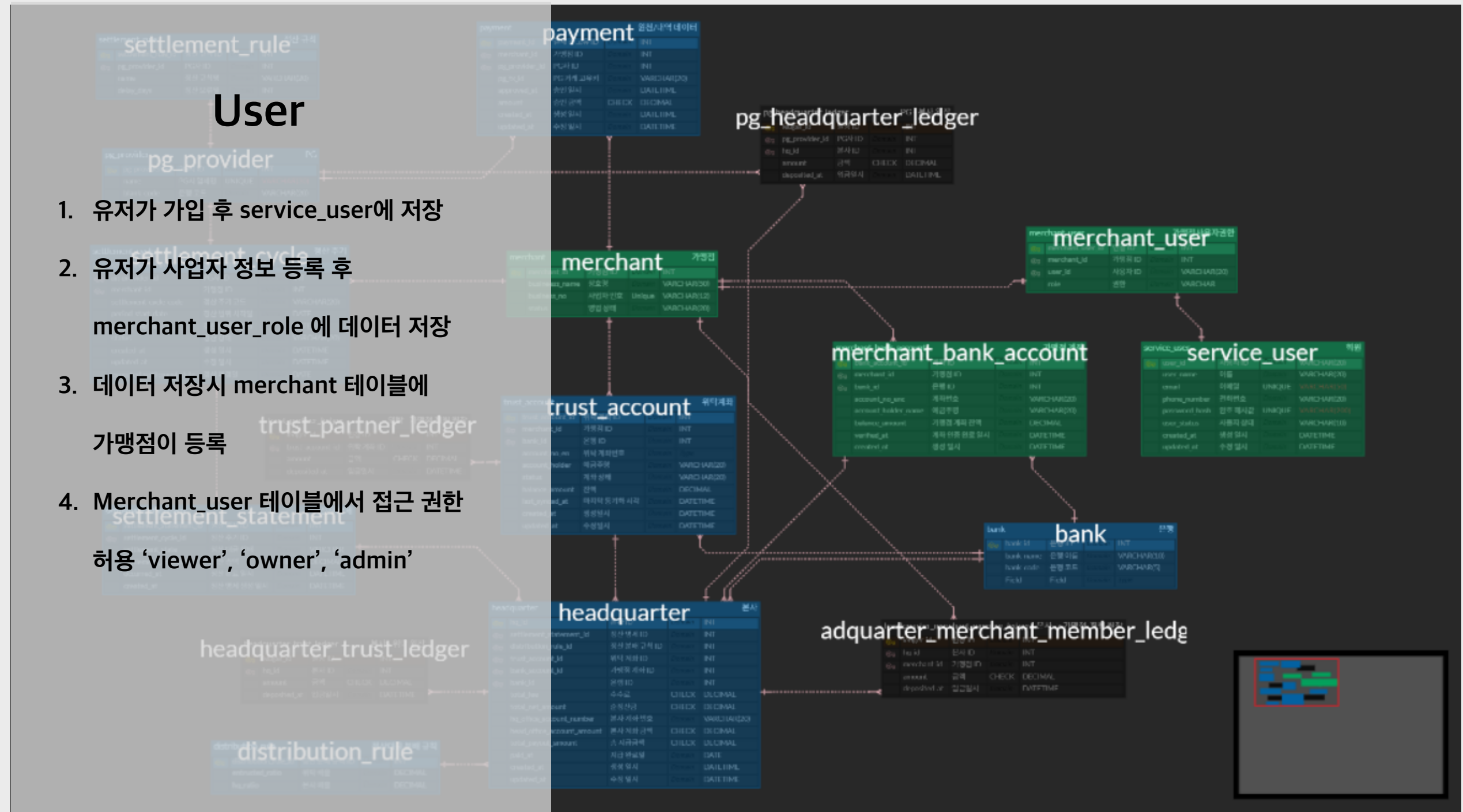


## System

1. 결제로우데이터(개별영수증) payment에 저장
2. Pg\_id와 merchant\_id를 통해 정산 대상을 특정
3. settlement\_cycle\_id 생성
4. 생성된 주기를 통해 statement 명세 작성(영수증 묶음)
5. Headquarter에서 distribution\_rule를 통해 지급데이터 생성
6. 위탁계좌로 입금
7. hq\_trust\_ledger(원장) 거래 흐름 기록
8. 이후 각각 merchant\_account 로 데이터 기록
9. trust\_partner\_ledger(원장) 거래 흐름 기록
10. \* 금액 타입 decimal(19,4) / 원장 기록 bigint

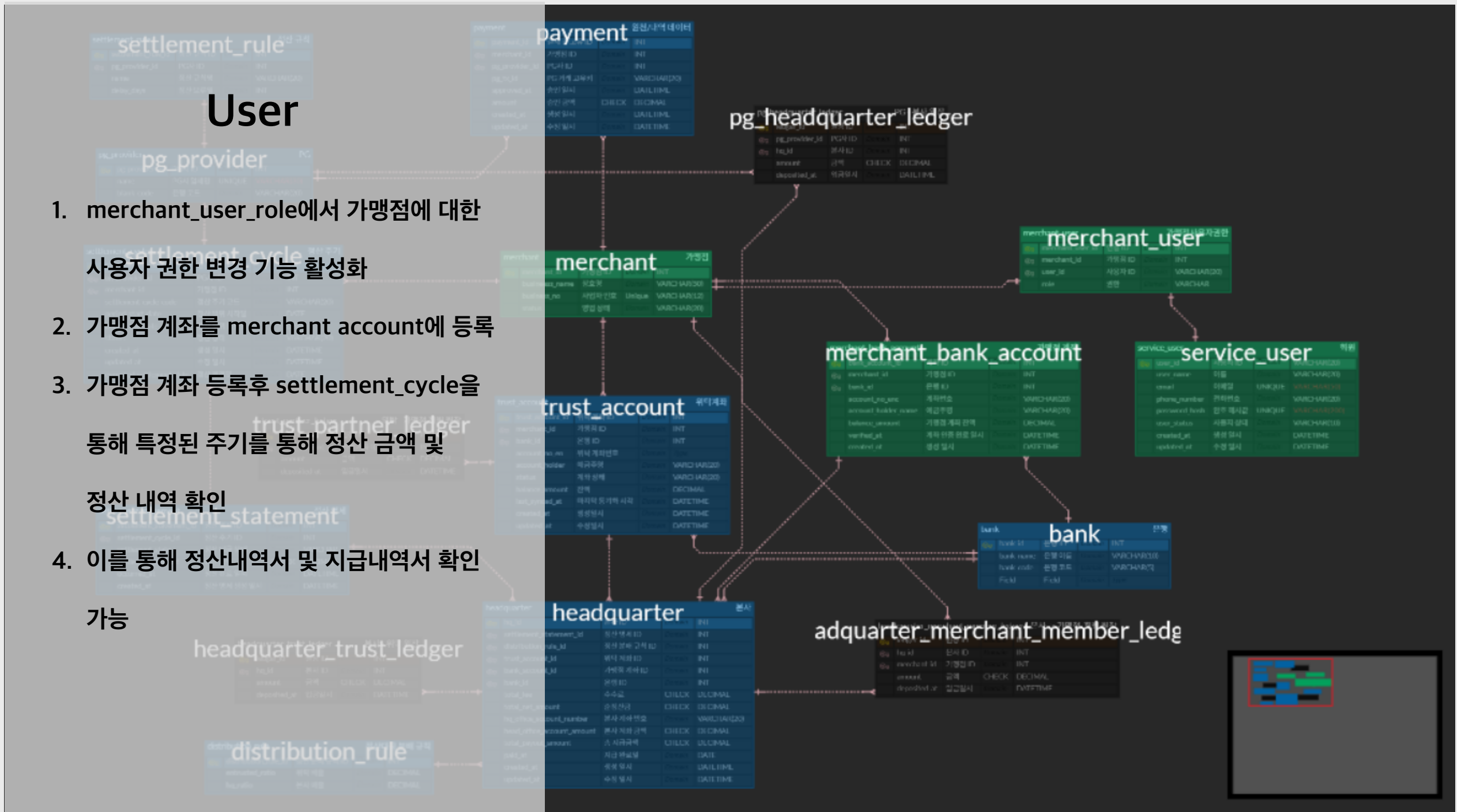
# 주요 테이블

1. 유저가 가입 후 service\_user에 저장
2. 유저가 사업자 정보 등록 후 merchant\_user\_role 에 데이터 저장
3. 데이터 저장시 merchant 테이블에 가맹점이 등록
4. Merchant\_user 테이블에서 접근 권한 허용 'viewer', 'owner', 'admin'



# 주요 테이블

1. merchant\_user\_role에서 가맹점에 대한 사용자 권한 변경 기능 활성화
2. 가맹점 계좌를 merchant account에 등록
3. 가맹점 계좌 등록후 settlement\_cycle을 통해 특정된 주기를 통해 정산 금액 및 정산 내역 확인
4. 이를 통해 정산내역서 및 지급내역서 확인 가능



# 주요 요구명세 설명

FR-02-04

정산 명세 생성

```

1  -- 4. FR-02-04 정산 명세 일괄 생성 (영수증 다발 처리)
2  INSERT INTO settlement_statement (
3      settlement_cycle_id,
4      settlement_state,
5      settlement_amount,
6      occurred_at,
7      created_at
8  )
9  SELECT
10     sc.settlement_cycle_id,
11     'REQUESTED',
12     SUM(p.amount),    -- 각 주기 별로 속한
13     NOW(),
14     NOW()
15 FROM settlement_cycle sc
16 JOIN payment p ON p.merchant_id = sc.merchant_id
17                AND p.pg_provider_id = sc.pg_provider_id
18 WHERE sc.status = 'PENDING' -- 아직 정산되지 않은 모든 주기들을 대상으로 함
19       AND p.approved_at >= sc.period_start_date
20       AND p.approved_at < DATE_ADD(sc.period_end_date, INTERVAL 1 DAY)
21 GROUP BY sc.settlement_cycle_id;
22 -- settlement_cycle 에서 상태값이 PENDING 인 영수증 다발로 만들어 settlement_statement 에 상태값 REQUESTED로 변경 후 명세서의 기재
23 SELECT * FROM settlement_statement;

```

1. 돈이 지급되지 않은 pending 상태 정산 주기 특정
2. 결제 날짜와 정산기간 확인
3. 해당 주기에 해당하는 p.amount 합산후 settlement\_statement내 status requested(신청)으로 insert

settlement_statement (14r x 6c)						
#	1 settlement_statement_id	2 settlement_cycle_id	3 settlement_state	4 settlement_amount	5 occurred_at	6 created_at
11	11	1	REQUESTED	10,000.0	2026-01-20 12:30:23	2026-01-20 12:30:23
12	12	1	REQUESTED	10,000.0	2026-01-20 12:36:44	2026-01-20 12:36:44
13	13	1	REQUESTED	10,000.0	2026-01-20 12:39:24	2026-01-20 12:39:24
14	14	1	REQUESTED	10,000.0	2026-01-20 14:27:53	2026-01-20 14:27:53

# 주요 요구명세 설명

FR-02-05

정산 완료 예정일 계산 및 갱신

```

1  -- FR-02-05 정산 완료 예정일 계산 및 갱신 "정산 주기 종료일에 D+N을 가산함으로써 가맹점에 대한 실제 지급일을 산출
2  -- 갱신된 정산 주기, 정산대금 상태 변경
3
4  UPDATE settlement_cycle sc
5  JOIN settlement_rule sr ON sc.pg_provider_id = sr.pg_provider_id
6  SET
7      sc.expected_settlement_date = DATE_ADD(sc.period_end_date, INTERVAL sr.delay_days DAY),
8      sc.status = 'REQUESTED',
9      sc.updated_at = NOW()
10 WHERE sc.status = 'CALCULATED';
11
12 SELECT * FROM
13 settlement_cycle;
14
15 -- PENDING -> CALCULATED -> REQUESTED -> SETTLED
16 -- 계산 완료된 정산 데이터를 찾아 settlement_rule 테이블에 따라 입금 예정일을 확정
17 -- CALCULATED -> REQUESTED 으로 상태 변경

```

1. cycle 테이블에서 지급 규칙에 따라 예상일 update
2. calculated된 상태를 requested로 변경

settlement\_cycle (10r × 10c)

#	1 settlement_cycle_id	2 settlement_cycle_code	3 merchant_id	4 pg_provider_id	5 period_start_date	6 period_end_date	7 status	8 created_at	9 updated_at	10 expected_settlement_d...
1	1	CYC-20260120-01	1	1	2026-01-19	2026-01-20	REQUESTED	2026-01-20 09:00:00	2026-01-20 14:41:13	2026-01-21
2	2	CYC-20260120-02	2	2	2026-01-19	2026-01-20	REQUESTED	2026-01-20 09:00:00	2026-01-20 14:41:13	2026-01-23
3	3	CYC-20260120-03	3	3	2026-01-19	2026-01-20	REQUESTED	2026-01-20 09:00:00	2026-01-20 14:41:13	2026-01-27
4	4	CYC-20260120-04	4	4	2026-01-19	2026-01-20	REQUESTED	2026-01-20 09:00:00	2026-01-20 14:41:13	2026-01-22



# 주요 요구명세 설명

FR-02-05

정산 완료 예정일 계산 및 갱신

```

1  -- 명세서 상태도 '지급대기'로 일괄 변경
2  UPDATE settlement_statement ss
3  JOIN settlement_cycle sc ON ss.settlement_cycle_id = sc.settlement_cycle_id
4  SET ss.settlement_state = 'READY'
5  WHERE sc.status = 'REQUESTED' AND ss.settlement_state = 'REQUESTED';
6
7  SELECT * FROM settlement_statement;
8
9  -- settlement_cycle 테이블의 REQUESTED 상태와 settlement_state 테이블의 REQUESTED 상태 컬럼들을 찾아
10 -- 명세서 상태를 '지급대기'로 변경

```

1. 최종적으로 cycle테이블의 Requested
2. state테이블에 requested된 상태값을 ready로 변경

settlement\_statement (14r × 6c)

#	1 settlement_statement_id	2 settlement_cycle_id	3 settlement_state	4 settlement_amount	5 occurred_at	6 created_at
7	7	7	COMPLETED	51,000.0	2026-01-20 09:30:00	2026-01-20 09:00:00
8	8	8	COMPLETED	12,500.0	2026-01-20 09:30:00	2026-01-20 09:00:00
9	9	9	COMPLETED	99,000.0	2026-01-20 09:30:00	2026-01-20 09:00:00
10	10	10	COMPLETED	67,000.0	2026-01-20 09:30:00	2026-01-20 09:00:00
11	11	1	READY	10,000.0	2026-01-20 12:36:25	2026-01-20 12:36:25
12	12	1	READY	10,000.0	2026-01-20 12:36:44	2026-01-20 12:36:44
13	13	1	READY	10,000.0	2026-01-20 12:39:24	2026-01-20 12:39:24
14	14	1	READY	10,000.0	2026-01-20 14:27:53	2026-01-20 14:27:53



# 주요 요구명세 설명

FR-02-05	정산 완료 예정일 계산 및 갱신
----------	-------------------

결제건을 모으고 날짜를 계산해  
실제 입금하기 까지의 전체 일정 관리

그 일정 안에서 정확히 얼마를 주고 받아야 하는지에  
대한 영수증

단계	Settlement Cycle (일정)	Settlement Statement (금액)
1. 합산	(생성) PENDING → CALCULATED	(생성) REQUESTED
2. 가산	CALCULATED → REQUESTED	REQUESTED
3. 승인	REQUESTED	REQUESTED → READY
4. 완료	REQUESTED → SETTLED	READY → COMPLETED

# 주요 요구명세 설명

FR-08-05	자금흐름 원장 감사
----------	------------

```

1 -- 가맹점 기준 전체 자금 흐름 타임라인 감사
2 SELECT
3   flow_type,
4   amount,
5   deposited_at
6 FROM (
7   /* 본사 -> 가맹점 */
8   SELECT
9     'HQ_TO_MERCHANT' AS flow_type,
10    amount,
11    deposited_at
12 FROM headquarter_merchant_member_ledger
13 WHERE merchant_id = 100
14
15 UNION ALL
16
17 /* 위탁 -> 가맹점 */
18 SELECT
19   'TRUST_TO_MERCHANT' AS flow_type,
20   amount,
21   deposited_at
22 FROM trust_partner_ledger
23 WHERE merchant_id = 100
24 ) t
25 ORDER BY deposited_at ASC;
26
27 /* 감사할 가맹점 id로 본사->가맹점 원장과 위탁->가맹점 원장을
28 UNION ALL로 합쳐서 비교 */

```

가맹점 기준 전체 흐름 감사  
merchant\_id를 통해 서로 다른 원장의  
입금 기록을 통합하여 hq - 위탁 / 위탁 - 가맹점  
자금흐름을 보여줌

#	1 trust_account_id	2 total_hq_to_trust	3 total_trust_to_merchant
1	1	4,750.0	4,750.0

# 주요 요구명세 설명

FR-08-05

자금흐름 원장 감사

```

1 -- 본사 기준 들어온 돈 vs 나간 돈 감사
2 SELECT
3   h.hq_id,
4
5   /* 들어온 돈 */
6   COALESCE(SUM(pgh.amount), 0) AS total_pg_in,
7
8   /* 나간 돈 (가맹점 지급) */
9   COALESCE(SUM(hm.amount), 0) AS total_hq_to_merchant
10
11 FROM headquarter h
12 LEFT JOIN pg_headquarter_ledger pgh
13   ON pgh.hq_id = h.hq_id
14 LEFT JOIN headquarter_merchant_member_ledger hm
15   ON hm.hq_id = h.hq_id
16 WHERE h.hq_id = 1
17 GROUP BY h.hq_id;
18
19 /* 감사할 본사를 본사id 기준 PG->본사 원장과 본사->가맹점 원장을 조회 */

```

본사 기준 들어온 입금액과  
위탁 계좌 출금액 확인

headquarter (1r x 3c)

#	1 hq_id	2 total_pg_in	3 total_hq_to_merchant
1	1	10,000.0	4,750.0

# 주요 요구명세 설명

FR-08-05

자금흐름 원장 감사

```

1 SELECT                                     -- 정산 1건 (hq_id) 단위 정합성 감사
2   h.hq_id                                AS 본사_정산_ID,
3   ss.settlement_statement_id             AS 정산_명세_ID,
4   ss.settlement_cycle_id                 AS 정산_주기_ID,
5   ss.settlement_amount                   AS 정산_기준_금액,
6   h.total_fee                            AS 수수료_금액,
7   h.total_net_amount                     AS 순정산_금액,
8   COALESCE(hm_sum.hq_paid, 0)            AS 본사_지급_금액,
9   COALESCE(tp_sum.trust_paid, 0)         AS 위탁_지급_금액,
10  COALESCE(hm_sum.hq_paid, 0) + COALESCE(tp_sum.trust_paid, 0) AS 실제_총_지급_금액,
11
12  ( h.total_net_amount
13    - (COALESCE(hm_sum.hq_paid, 0)
14      + COALESCE(tp_sum.trust_paid, 0))
15  )                                       AS 순정산_대비_지급_차이
16 FROM headquarter h
17 JOIN settlement_statement ss
18   ON ss.settlement_statement_id = h.settlement_statement_id
19 /* 본사 → 가맹점 지급 합계 */
20 LEFT JOIN (
21   SELECT hq_id, SUM(amount) AS hq_paid FROM headquarter_merchant_member_ledger
22   GROUP BY hq_id
23 ) hm_sum ON hm_sum.hq_id = h.hq_id
24 /* 위탁 → 가맹점 지급 합계 */
25 LEFT JOIN (
26   SELECT merchant_id, SUM(amount) AS trust_paid FROM trust_partner_ledger
27   GROUP BY merchant_id
28 ) tp_sum ON tp_sum.merchant_id = h.merchant_id
29 WHERE h.hq_id = 1;

```

정산 단일 건에 대해 전체 원장 통합 상세대조  
정산 기준액 - 수수료 - 본사지급액 - 위탁지급액 확인후  
미지급금이나 과다 지급 없이 데이터 무결성 증명

headquarter (1r x 10c)

#	1 본사_정산_ID	2 정산_명세_ID	3 정산_주기_ID	4 정산_기준_금액	5 수수료_금액	6 순정산_금액	7 본사_지급_금액	8 위탁_지급_금액	9 실제_총_지급_금액	10 순정산_대비_지급_차이
1	1	1	1	10,000.0	500.0	9,500.0	4,750.0	4,750.0	9,500.0	0.0