

앵커링 신규 시스템 설계

커스텀 Key 기반 전면 재설계 — 기존 모듈 대체 · 2026-08-06 v2 · feature/negodata

- 이 문서는 기존 앵커링 시스템의 확장이 아니라 대체 설계다.

 - ① 앵커링 값을 쪼개는 **key(축)** 구성을 회사마다 커스텀할 수 있다 — A사는 (협력사유형×가격대), B사는 (가격대만), C사는 (유형×가격대×카테고리).
 - ② 저장은 **테이블 2개**: 칸별 현재값(`anchor_policies`) + 변경 로그(`anchor_events`). 폴백 체인·파일·뷰 없음.
 - ③ 학습은 격주 배치가 아니라 **협상 종료 시점의 카운터 누적 + 즉시 조정**. 스케줄러 컨테이너·Redis·세션 스캔·소비 마킹 전부 소멸.
 - ④ 기존 anchoring 스키마(`adjustments`·뷰 2종)·파일 2벌·`schedules/anchoring` 모듈은 이관 후 폐기한다.

1. 핵심 개념 — key가 데이터다

기존 시스템은 칸 키(회사, 협력사유형, 가격구간)가 **스키마에 고정**돼 있다. 신규 시스템은 키를 JSONB 값으로 저장해, **회사가 어떤 축으로 값을 쪼갤지 스스로 정한다**.

회 사	축 구성(회사 설정)	칸 key 예시 (<code>cell_key</code> 실제 저장값)
A 사	<code>["supplier_type", "price_band"]</code> (기본값)	<code>{"supplier_type":1,"price_band":2}</code>
B 사	<code>["price_band"]</code>	<code>{"price_band":2}</code>
C 사	<code>["supplier_type", "price_band", "category"]</code>	<code>{"supplier_type":1,"price_band":2,"category":"MR0"}</code>

축의 **종류**는 코드의 축 레지스트리가 제공하고(각 축마다 "협상 컨텍스트에서 값을 뽑는 추출기" 구현), 회사는 그 조합만 고른다. 축 구성은 기존 회사별 커스터마이징 시스템(`companies.settings` JSONB)에 `settings.anchoring.axes` 로 저장한다.

축	값 추출	값 예
<code>price_band</code>	목표가 → 코드의 경계 배열 이진탐색. 기본 <code>[1만, 10만, 100만, 1천만, 1억]</code> → 6칸	<code>0~5</code> 정수
<code>supplier_type</code>	(상품,협력사) 매핑의 <code>supply_type</code>	<code>1/2/3</code>
<code>category</code>	<code>items.category</code>	문자열 코드
<code>qt_type</code>	견적 유형(신규/재협상)	정수 코드

축 추출이 불가능한 협상(예: `supplier_type` 축을 쓰는데 매핑이 없음)은 앵커는 걸되(기본값) 학습 카운트에서 제외한다. 유형 축을 안 쓰는 회사는 매핑이 없어도 학습에 아무 지장이 없다 — 커스텀 key가 기존의 "매핑 없으면 표본 제외" 문제를 회사 선택으로 우회하게 해준다.

2. DB 설계 — 테이블 2개

```

-- ① 칸별 현재 상태 (제자리 UPDATE - 이 테이블이 정보)
CREATE TABLE anchoring.anchor_policies (
  policy_id          BIGSERIAL PRIMARY KEY,
  company_id         UUID      NOT NULL,
  cell_key           JSONB     NOT NULL,    -- 회사 측 구성대로 조립된 key
  value_permille     SMALLINT NOT NULL CHECK (value_permille BETWEEN 10 AND 200), -- %(1~20%)
  window_success     INTEGER NOT NULL DEFAULT 0,    -- 다음 조정을 위한 진행 카운터
  window_total       INTEGER NOT NULL DEFAULT 0,
  last_adjusted_at   TIMESTAMPTZ,                -- 자동조정 최소간격 가드 기준
  created_at         TIMESTAMPTZ NOT NULL DEFAULT now(),
  updated_at         TIMESTAMPTZ NOT NULL DEFAULT now(),
  UNIQUE (company_id, cell_key)    -- JSONB 동등비교는 키 순서 무관 → 칸 유일성 보장
);

-- ② 변경 로그 (append-only - 이력·감사·화면용)
CREATE TABLE anchoring.anchor_events (
  event_id          BIGSERIAL PRIMARY KEY,
  company_id         UUID      NOT NULL,
  cell_key           JSONB     NOT NULL,
  kind              SMALLINT NOT NULL,    -- 1 SEED / 2 AUTO_ADJUST / 3 MANUAL_SET / 4 RESET
  value_before       SMALLINT,
  value_after        SMALLINT NOT NULL,
  sample_count       INTEGER,             -- AUTO_ADJUST 때만: 평가 표본 수
  success_count      INTEGER,             -- " : 성공 수
  actor_user_id      UUID,               -- MANUAL_SET/RESET 때만: 수정자
  created_at         TIMESTAMPTZ NOT NULL DEFAULT now()
);
CREATE INDEX idx_anchor_events_cell ON anchoring.anchor_events (company_id, created_at DESC);

-- ③ 세션 박제(기존 컬럼 유지 + key 박제 추가)
ALTER TABLE negotiation.sessions ADD COLUMN anchoring_cell_key JSONB; -- 견적 생성 시 조립된 key

```

규약 준수: 코드 SMALLINT · 시각 TIMESTAMPTZ · 값은 기존과 동일한 정수 천분율(%). 폐기 대상: `anchoring.adjustments` ·뷰 2종· `anchoring_base.json` 2벌·46구간 상수· `sessions.used_by_adjustment_id`.

예시 데이터

company_id	cell_key	value_permille	window (성공/전체)	last_adjusted_at
A사	<code>{"supplier_type":1,"price_band":2}</code>	30 (3%)	2 / 4	08-22
A사	<code>{"supplier_type":2,"price_band":2}</code>	10 (1%)	5 / 7	—
B사	<code>{"price_band":2}</code>	50 (5%) ← 수동 설정	0 / 0	—
C사	<code>{"supplier_type":1,"price_band":2,"category":"MR0"}</code>	10 (1%)	0 / 0	—

- 1행: 한 번 자동조정된 칸(1%→3%), 조정 후 표본 4건 다시 쌓이는 중.
- 3행: B사 OWNER가 화면에서 5% 입력 — 자동값·수동값 구분 컬럼이 없다. 같은 value를 고치는 것이고 출처는 events에 남는다.
- 행이 없는 칸 = 아직 협상이 없었던 칸. 첫 협상 때 기본 1%로 자동 생성(SEED).

3. 런타임 흐름

견적 생성 — 목표가 331,920원, A사(축: 유형×가격대)

① 회사 axes 로드 → 추출기 실행: `supplier_type=1(매핑)`, `price_band=2(경계 배열 이진탐색)` → `cell_key = {"supplier_type":1,"price_band":2}`

② `anchor_policies` 에서 (A사, `cell_key`) 조회 — 없으면 1%로 행 생성(SEED 이벤트)

③ 앵커가 = $331,920 \times (1-0.03) = 321,960$ 원(10원 반올림) → 세션에

`anchoring_price·anchoring_value·anchoring_cell_key` 박제

실패 격리: 이 조회가 어떤 이유로든 실패해도 기본 1%로 진행 — "견적 생성은 앵커 때문에 죽지 않는다" 규칙 유지(코드 폴백 상수 하나)

협상 종료 — 세션이 닫히는 트랜잭션 안에서

① 가격 흔적 판정(기존 기준 그대로): 가격을 한 번이라도 써냈으면 표본, 아니면 제외

② 성공 = 낙찰가 ≤ 박제된 앵커가

③ 박제된 `cell_key` 행에 원자적 증가: `UPDATE ... SET window_total = window_total + 1, window_success = window_success + (성공?1:0)`

기존 격주 배치의 "종료 세션 전체 스캔 → 파생 판정 → 소비 마킹 → 이중조정 방어"가 이 UPDATE 한 줄로 대체된다. 세션은 정확히 자기 종료 시점에 1회만 반영되므로 이중 집계가 구조적으로 불가능

자동 조정 — 카운터 증가 직후 같은 트랜잭션에서 검사

① `window_total ≥ 10` 이고 `last_adjusted_at` 이 14일 이전(또는 NULL)이면 평가

② 성공률 $\geq 60\%$ → $+\delta$ / $<30\%$ → $-\delta$ / 사이 → 유지. δ 는 `supplier_type` 축이 있으면 유형별(유통2%p·제조1%p·총판1.5%p), 없으면 기본 1%p. 결과는 1~20% 클램프

③ `value_per mille` UPDATE + AUTO_ADJUST 이벤트(before/after/표본수/성공수) + 카운터 리셋

격주 토요일을 기다리지 않는다. 표본이 차는 즉시 조정되고, 최소 간격 14일이 과속을 막는다. 배치 유실(8/1 같은) 문제 자체가 소멸

수동 설정 — 설정 화면(OWNER)

값 입력 → 같은 행 `value_per mille` UPDATE + MANUAL_SET 이벤트(actor 기록). 자동조정과 동일한 경로라 우선순위 개념이 없다. 이후 학습은 그 값에서 이어간다. "되돌리기" = 원하는 시점 값으로 다시 UPDATE(RESET 이벤트)

축 구성 변경(회사 설정 변경)

축을 바꾸면 기존 행들과 key 모양이 달라지므로 **그 회사 학습은 새로 시작한다**(기존 행은 남지만 새 key와 매칭되지 않음 — 어떤 설계로도 "유통×30만대에서 배운 값"을 "카테고리 칸"으로 옮길 수는 없다). 화면에서 축 변경 시 이 사실을 경고하고 확인받는다. 진행 중인 협상은 박제된 key로 끝까지 정상 집계된다.

4. 케이스별 조회·기록 정리 (실데이터)

회사 = dev 실존 a35152d2, 축 = 기본 ["supplier_type", "price_band"], 가격대 경계 = [1만, 10만, 100만, 1천만, 1억]. 세션 3건은 전부 dev DB 실데이터다.

케이스 1 — 견적 생성: 칸에 값이 있음

```
-- 목표가 331,920원 → band = 2 ([10만~100만]), 매핑 supply_type = 1(유통)
SELECT value_permille
FROM anchoring.anchor_policies
WHERE company_id = 'a35152d2-db61-4760-9f1e-beb9736d957f'
  AND cell_key = '{"supplier_type":1,"price_band":2}':::jsonb;
-- → 30 (3%) → 앵커가 = 331,920 × 0.97 = 321,960원(10원 반올림)
-- 세션에 박제: anchoring_price=321960, anchoring_value=30, anchoring_cell_key={...}
```

케이스 2 — 견적 생성: 칸에 값이 없음 (그 조건의 첫 협상)

```
-- 조회 결과 0행 → 기본 1%로 행을 만들면서 그 값을 쓴다
INSERT INTO anchoring.anchor_policies (company_id, cell_key, value_permille)
VALUES ('a35152d2-...', '{"supplier_type":1,"price_band":2}', 10)
ON CONFLICT (company_id, cell_key) DO NOTHING; -- 동시에 견적 2개가 와도 안전
INSERT INTO anchoring.anchor_events (company_id, cell_key, kind, value_after)
VALUES ('a35152d2-...', '{"supplier_type":1,"price_band":2}', 1 /*SEED*/, 10);
-- 앵커가 = 331,920 × 0.99 = 328,600원 (dev 실제 세션이 정확히 이 값)
```

케이스 3 — 축 값을 못 뱌음 (매핑 없음 — dev 세션 42건 중 38건이 이 경우)

supplier_type 축을 쓰는 회사인데 (상품,협력사) 매핑이 없으면 cell_key를 조립할 수 없다 → 행 조회·생성 없이 기본 1%로 앵커만 걸고, 세션의 anchoring_cell_key = NULL 로 박제한다. 종료 시 key가 없으므로 학습 카운트에서 자동 제외. 협상 자체는 아무 지장 없다. (유형 축을 안 쓰는 회사는 이 케이스가 아예 발생하지 않는다.)

케이스 4 — DB 조회 자체가 실패 (장애·스키마 미적용)

예외를 밖으로 던지지 않고 코드 폴백 상수 1%로 진행 + WARN 로그. "견적 생성은 앵커 때문에 죽지 않는다" — 기존 규칙 그대로.

케이스 5 — 협상 종료: 카운터 반영

```
-- 실세션 ①: 목표가 331,920 / 박제 앵커 328,600 / 낙찰 320,000 → 320,000 ≤ 328,600 = 성공
UPDATE anchoring.anchor_policies
SET window_total = window_total + 1,
    window_success = window_success + 1, -- 실패면 이 줄 없음
    updated_at = now()
WHERE company_id = 'a35152d2-...'
  AND cell_key = (세션에 박제된 anchoring_cell_key);

-- 실세션 ②: 목표가 109,718 / 앵커 108,620 / 120,000 제시하다 거절(status 5)
-- → 가격 흔적 있음 = 표본, 성공 아님 → window_total만 +1
-- 실세션 ③: 목표가 29,000 / 가격 제시 없이 미참여(status 4)
-- → 가격 흔적 없음 → 아무것도 안 함 (표본 제외)
```

케이스 6 — 카운터가 10을 채움: 즉시 자동 조정

```
-- 위 UPDATE 직후 같은 트랜잭션에서 (행 잠금 후) 검사
-- window = 7/10 (70%) 이고 last_adjusted_at이 14일 이전(또는 NULL)이면:
-- 70% ≥ 60% → +6(유통 20%) → value 10 → 30
UPDATE anchoring.anchor_policies
SET value_permille = 30, window_success = 0, window_total = 0, last_adjusted_at = now()
WHERE policy_id = ...;
INSERT INTO anchoring.anchor_events (... , kind, value_before, value_after, sample_count, success_count)
VALUES (... , 2 /*AUTO_ADJUST*/, 10, 30, 10, 7);
-- 14일이 안 지났으면 조정하지 않고 카운터만 계속 쌓다가, 지난 뒤 첫 종료 때 누적분으로 평가
```

케이스 7 — 화면에서 수동 설정 (값이 있든 없든)

```
-- OWNER가 그리드에서 5.0% 입력 — 행이 없던 칸이면 이게 곧 "시작값 커스텀"
INSERT INTO anchoring.anchor_policies (company_id, cell_key, value_permille)
VALUES ('a35152d2-...', '{"supplier_type":2,"price_band":2}', 50)
ON CONFLICT (company_id, cell_key)
DO UPDATE SET value_permille = 50, window_success = 0, window_total = 0, updated_at = now();
-- 카운터 리셋: 새 값 기준으로 표본을 다시 센다
INSERT INTO anchoring.anchor_events (... , kind, value_before, value_after, actor_user_id)
VALUES (... , 3 /*MANUAL_SET*/, 10, 50, '수정자 uuid');
```

케이스 8 — 설정 화면 조회

```
SELECT cell_key, value_permille, window_success, window_total, last_adjusted_at, updated_at
FROM anchoring.anchor_policies
WHERE company_id = 'a35152d2-...'; -- 그리드는 이 결과를 축 구성대로 배치만 하면 끝
-- 이력 모달:
SELECT * FROM anchoring.anchor_events
WHERE company_id = 'a35152d2-...' ORDER BY created_at DESC LIMIT 50;
```

5. negodata 설정 화면

/settings 앵커링 탭, OWNER 전용. 화면은 회사의 축 구성을 읽어 **동적으로** 그린다 — 축 1개면 리스트, 2개면 표(행×열), 3개면 축 하나를 필터로 빼고 표.

앵커링

협상 시작가를 목표가에서 얼마나 낮춰 부를지 정합니다. 값은 직접 수정할 수 있고, 협상 결과가 쌓이면 자동으로 조정됩니다.

축 구성: 협력사유형 가격대 + 카테고리 + 견적유형

3건

수동 설정

1건

자동 조정

표본 15건

다음 조정까지 누적 중

변경 이력

가격대	유통	제조	총판
~1만원	1.0%	1.0%	1.0%
1만~10만	1.0% 5/7	1.0%	1.0%
10만~100만	3.0% 자동조정 08-22	5.0% 수동	1.0%
100만~1천만	1.0%	1.0%	칸 없음
1천만~1억	1.0%	칸 없음	칸 없음

저장

축 구성 변경

- 셀 = 그 칸의 현재값. 인라인 편집(1.0~20.0%, 0.1%p 스텝 — 검증은 프론트 전용). 배치로 마지막 변경 출처(자동/수동)와 일자 표시, 작은 회색 숫자는 진행 중 표본 카운터.
- "칸 없음" = 아직 협상이 없어 행이 미생성. 클릭해 값을 입력하면 행 생성(= 시작값 커스텀).
- 변경 이력 모달 = `anchor_events` 리스트(일시 · 칸 · 종류 · before→after · 표본/성공 · 수정자).
- 축 구성 변경 = 레지스트리의 축 중 조합 선택. 저장 전 "학습이 새로 시작됩니다" 경고.

API

엔드포인트	역할
GET <code>/v1/company/anchoring</code>	축 구성 + 칸 그리드(각 칸: <code>cell_key</code> , 값, 카운터, 마지막 변경 출처·일시) + 요약
PUT <code>/v1/company/anchoring/values</code>	칸 값 일괄 upsert — <code>[{cell_key, value_permille}]</code> (MANUAL_SET)
PUT <code>/v1/company/anchoring/axes</code>	축 구성 변경(<code>companies.settings</code> 갱신)
GET <code>/v1/company/anchoring/events</code>	변경 이력 페이지 조회

규약: `AnchoringProtocol(WebPacketProtocol)` 베이스 + `Req_/Res_` 쓰기는 `execute_lambda_run`. OWNER 게이팅 프론트+백엔드. 새 라우터 디렉토리 → `docker compose restart negodata-backend`. 프론트 orval 재생성.

6. 기존 시스템에서 갈아타기

현행 데이터 실측(2026-08-06): 조정 이력 0행, 세션 42건 전부 시작값 1% — **이관할 학습 자산이 없다. 지금이 교체 적기다.**

단계	작업	완료 기준
1	DDL 2테이블 + sessions.anchoring_cell_key 추가(신규 스크립트 + dev 보정 SQL, 맥등) → 축 레지스트리·key 조립·policies 읽기 (견적 생성 경로 교체, 파일 로드 제거)	pytest green + 견적 생성 실응답 앵커가 동일(전 칸 1%)
2	협상 종료 혹: 카운터 증가 + 즉시 조정 + events 기록	테스트 협상으로 카운터·조정·이벤트 실측 확인
3	설정 화면(조회 그리드 → 값 편집 → 축 구성 → 이력)	수정 → 견적 재생성 시 앵커가 즉시 반영(실응답)
4	구시스템 철거: schedules/anchoring 컨테이너·Redis, adjustments·뷰, 파일 2벌, negodata common/anchoring 구코드, used_by_adjustment_id	compose에서 anchoring 서비스 제거 후 전체 플로우 정상

기존 대비 요약

기존	신규
키 고정(회사×유형×46구간), 스키마에 박제	키 커스텀 — 회사가 축 조합 선택, JSONB로 저장
저장 4원화: 코드 상수 + 파일 2벌 + adjustments + 뷰 2, 읽기 4단 폴백	테이블 2개, 읽기 2단(행 없으면 1%)
격주 토 배치: 전체 스캔·파생판정·소비마킹·이중조정 방어, 유실 시 회차 증발	협상 종료 시 카운터 +1, 표본 차면 즉시 조정(최소 14일 간격). 배치·Redis·스케줄러 소멸
수동 설정 자리가 없음(append-only 유도 구조)	수동=자동과 같은 UPDATE, 출처는 events
46구간 과세분화로 표본 분산	기본 6칸(1만/10만/100만/1천만/1억 경계)으로 시작, 필요 시 세분화

증가량

테이블	증가 규칙
anchor_policies	칸당 1행 upsert — 협상 횟수와 무관. 상한 = 회사별 축 조합 곱(기본 축이면 3×6=18행/회사)
anchor_events	값이 실제로 바뀌거나 칸이 생길 때만 1행. 조정은 칸당 최소 14일 간격이라 상한 명확

7. 결정 필요 / 리스크

사안	내용
공유 sessions 컬럼 정리	used_by_adjustment_id 는 구시스템 소유 — 철거 4단계에서 함께 정리(루트 backend-schedules 참조 grep 필수)
동시 종료 경합	카운터 증가는 원자적 UPDATE라 안전. 조정 판정은 행 잠금(SELECT ... FOR UPDATE) 후 수행
축 레지스트리 초기 범위	1차: supplier_type · price_band. category · qt_type은 추출기만 준비하고 화면 노출은 2차
δ·임계값의 축 독립성	60/30 임계·표본 10·간격 14일은 전역 상수로 시작. 회사별 커스텀은 수요 확인 후
구시스템 병행 기간	1~3단계 동안 격주 배치는 꺼둔다(어차피 조정 0행) — 병행 운영 없음, 컷오버 1회