이인범

Backend Developer

설계부터 배포까지, 시스템을 안정적으로 지탱하는 백엔드 개발자입니다

병목은 추측이 아니라 계측으로 원인 계층을 특정합니다

일 수백만~수천만 건 규모 로그 집계에서 EXPLAIN ANALYZE로 병목을 짚어, GROUP BY 집계를 윈도우 함수로 재작성하고 work_mem 설정과 ES 인덱스 템플릿을 조정해 집계 성능을 30~50% 개선했습니다. 15만 부서 조직도 조회는 수만 회 반복되던 서브쿼리를 CTE로 분리하고 재귀 쿼리 구조를 정리해 1분에서 5초 이내로 단축했습니다. 개발 서버 커넥션 고갈은 pgBouncer로 서버 커넥션을 20개 이내로 낮춰 해소했고, MSA 환경의 연쇄 장애 대응은 대학원 연구로 이어가고 있습니다.

변경 비용은 모듈 경계와 테스트로 관리합니다

정제 로직이 4개 모듈에 흩어져 한 곳을 고치면 다른 곳에서 사이드 이펙트가 나던 구조를, 데이터 흐름 중심의 단일 모듈로 재설계했습니다. 화면마다 개별 구현되던 엑셀 다운로드는 인터페이스와 팩토리 패턴으로 공통화해 신규 화면은 기존 코드 수정 없이 확장만으로 대응하게 했고, JUnit 테스트 환경을 구축해 커버리지를 0에서 70%로 올리고 이슈 발생률을 30% 낮췄습니다.

About

- WEEDS KOREA · 2023. 06 ~ 재직 중
- 서강대학교 AI·SW대학원
- robor082373@gmail.com
- https://inbeom.tistory.com
- https://github.com/Lib0823

Skills

- Java, Python
- Spring Boot, Spring MVC, MyBatis, JPA, JUnit
- Vue.js, JSP
- PostgreSQL, pgBouncer, SQLite
- Elasticsearch, OpenSearch
- Docker, K3s, Jenkins, Prometheus, Grafana

개인정보 사용기록 감사 · 이상행위 탐지 시스템

2023. 06 ~ 재직 중

개요

고객사 정보시스템에 설치되어 개인정보 접근·사용 행위를 수집·분석하고, 이상행위 탐지와 소명을 지원하는 온프레미스 보안 솔루션입니다.
접속기록 법정 보관 의무(최소 2년)로 하루 수백만~수천만 건의 로그가 삭제 없이 누적됩니다.
규모가 커지며 드러난 조회 성능, 데이터 구조, 커넥션 자원, 코드 유지보수성 문제를 각 층위에서 해결했습니다.

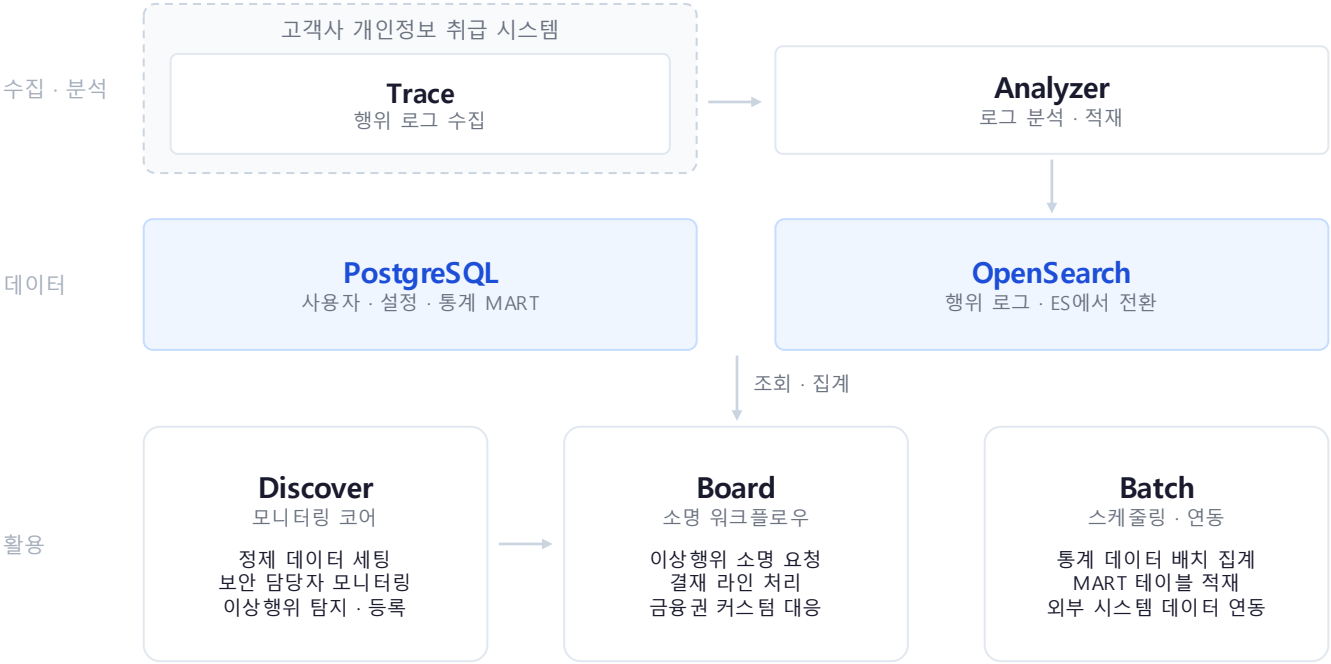
역할

- 대용량 집계 쿼리 튜닝 — EXPLAIN ANALYZE 병목 진단, 윈도우 함수 재작성, 양 스택 조정
- 조직도 트리 조회 재설계 — CTE 분리, 재귀 쿼리 개선, 응답 압축까지 전 구간 개선
- 집계 데이터 저장 구조 재설계 — MART 선집계·파티셔닝, OpenSearch 전환 및 매핑 재설계
- 개발 서버 커넥션 고갈 해소 — pgBouncer transaction 모드 도입, prepared statement 대응
- 정제 모듈 통합 리팩토링 — 4개 모듈 단일화, 팩토리 패턴 공통화, JUnit 테스트 환경 구축

Skills

- Java · Spring Boot · Spring MVC · MyBatis · JUnit · PostgreSQL · OpenSearch
- Docker · K3s · Jenkins · pgBouncer · EXPLAIN ANALYZE · Table Partitioning

시스템 아키텍처



Discover에서 탐지·등록한 이상행위를 Board로 넘겨 소명을 처리하고, Batch는 통계 집계와 외부 연동을 스케줄로 실행합니다.

1. 15만 부서 조직도 트리 조회 타임아웃

AS-IS

모니터링 화면의 조직도 트리 조회가 1분 이상 지연되어 타임아웃 발생. 부서 단위로 수만 회 반복 실행되는 서브쿼리, 불필요한 역직렬화, 그리고 15만 부서 트리를 통째로 내리며 커진 응답 페이로드가 겹친 문제.

TO-BE

수만 회 반복되던 서브쿼리를 CTE로 분리해 단일 실행 구조로 정리하고, 재귀 쿼리 구조 개선과 불필요한 역직렬화 제거를 함께 적용. 트리 JSON 응답 크기가 커 전송도 병목으로 남아, 실제 응답 크기를 측정해 해당 요청만 압축되도록 임계값을 정하고 Tomcat gzip 적용. 조회·직렬화·전송을 함께 개선해 1분 → 5초 이내(약 92% 단축).

핵심 키워드: 반복 서브쿼리 제거, CTE 분리, 조회·직렬화·전송 구간 개선

2. 집계 데이터 저장 구조 재설계 — RDB · NoSQL

AS-IS

통계 화면이 조회 시점마다 원본 로그를 직접 집계해 응답이 지연되고, 검색 스택은 nested 타입의 복합 집계 제약과 라이선스·확장성 한계를 함께 안고 있었음.

TO-BE

RDB는 스케줄링 배치로 선집계한 MART 테이블을 두고 조회 경로를 분리, 대규모 구간에는 테이블 파티셔닝 적용. NoSQL은 OpenSearch로 전환하고 주요 필드를 1depth keyword로 추가 매핑해 복합 집계 제약을 해소. JUnit 배치 테스트로 정합성을 사전 검증.

핵심 키워드: 선집계 MART · 파티셔닝, 검색엔진 전환 · 매핑 재설계

3. 개발 서버 커넥션 고갈

AS-IS

서비스 분리와 신규 개발이 겹치며 앱들이 요구하는 커넥션 합계가 PostgreSQL 서버 상한을 넘어섬. 피크·배치 병렬 실행 구간에서 초과분의 연결이 거부되며 개발 서버 전반의 작업이 중단되고 CPU·Memory 부하가 지속됨.

TO-BE

유휴 커넥션이 서버 연결을 점유하지 않도록 transaction 모드로 도입해, 실제 서버 커넥션이 pgBouncer 풀 크기(20) 이내로만 유지되도록 변경. 서버 커넥션이 매번 바뀌며 발생한 prepared statement 오류는 JDBC 서버측 prepare 비활성화로 우선 해소한 뒤, pgBouncer의 prepared statement 추적 기능으로 전환해 실행 계획 캐싱을 회복.

핵심 키워드: transaction 모드 풀링, prepared statement 대응, 풀 사이징

4. 정제 로직 파편화로 인한 사이드 이펙트

AS-IS

데이터 가공·정제 기능이 4개 모듈에 분산돼 공통 로직이 중복되고 처리 흐름이 어긋남. 신규 기능 추가 시 4개 모듈을 모두 파악해야 했고, 한 곳을 고치면 다른 모듈에서 사이드 이펙트가 발생.

TO-BE

데이터 흐름 중심의 단일 통합 모듈로 재설계하고, 별도 브랜치로 분리해 기존 코드에 영향 없이 진행. 화면마다 개별 구현되던 엑셀 다운로드는 인터페이스·팩토리 패턴으로 공통화해 신규 화면은 확장만으로 대응. 공통 응답 포맷과 @ControllerAdvice 예외 처리도 표준화.

핵심 키워드: 모듈 통합 재설계, 팩토리 패턴 공통화, 응답·예외 표준화

성과 요약

- 집계 파이프라인 성능 30~50% 개선 — PostgreSQL · OpenSearch 양 스택 동시 튜닝
- 15만 부서 조직도 트리 조회 1분 → 5초 이내 (약 92% 단축)
- 개발 서버 커넥션 고갈 장애 해소 및 CPU · Memory 부하 안정화
- JUnit 테스트 환경 구축 — 커버리지 0 → 70%, 이슈 발생률 약 30% 감소
- 신규 화면 엑셀 다운로드는 기존 코드 수정 없이 확장만으로 구현 가능

회고

같은 문제라도 풀어야 할 레이어가 다르다는 것을 배웠습니다. 트리 조회는 쿼리 구조를 고쳐 해결했지만, 통계 조회는 쿼리를 다듬는 것만으로는 한계가 있어 실시간 집계를 포기하고 미리 계산해 두는 구조로 바꿔야 했고, 커넥션 고갈은 코드가 아니라 자원 계층의 문제였습니다. 정제 로직은 성능이 아니라 변경 비용의 문제라 모듈 경계를 다시 그어야 했습니다.
아쉬운 점은 개선 전후 측정 기준을 처음부터 정해두지 않아 일부 구간이 체감 수치로만 남았다는 것입니다. 이후 작업부터는 대표 쿼리의 실행 계획과 응답 시간을 먼저 기록하고 시작합니다.

Other Experiences

WEEDS KOREA · 2023. 06 ~ 재직 중

배포 체계 설계 · 리드

- 배포 TF 리드 — 4개 팀 제품 10여 종의 연동 지점과 잔여 기능·이슈를 전수 정리, 공수 산정 후 일감 배분
- 수동으로 브랜치·태그를 만들어 관리하던 버전 체계를 CI 기반 major·minor·patch 자동 채번으로 전환
- Jenkins 단독 구조를 GitLab CI → Nexus → Jenkins CD로 분리 — 재빌드 없이 적재된 산출물만 배포
- 제품과 서드파티 구성 요소의 버전을 함께 고정해, 고객사에 나간 릴리스를 동일하게 재현·재배포

인프라 현대화

- 개발 서버 8대 중 부하가 집중된 4대를 K3s 클러스터로 묶어 리소스 분배가 가능하도록 전환 설계·구축 리드
- 파드 직접 접근이 필요해 Ingress 대신 MetalLB로 사설 IP 부여, 장애 대비 Longhorn 레플리카
- 테스트 환경 간 간섭을 막기 위해 Namespace로 격리하고, Prometheus · Grafana · Loki 관측 체계 설계
- Kubernetes API를 감싼 MCP 서버를 개발해 조회·제한된 배포를 도구화 — 쉘 접속 없이 개발 환경 운영

민감정보 처리

- 민감정보 ARIA-256 암호화·복호화와 마스킹, HMAC 위변조 방지 구현 — 삼성카드 등 금융권으로 확대 적용
- 금융권 보안 취약점 점검과 GS 인증 심사에 대응해 API 보안 체계 보완 — 제품 GS 인증 1등급 획득

AI 기반 주식 데이터 분석 및 자동매매 시스템

GitHub

2026. 01 ~ 개발 진행 중

개요

주식·채권·코인을 아우르는 자산관리 서비스 FMA(Finance Manage Agent)로, 주식 영역을 우선 완성했고 채권·코인은 확장을 준비하고 있습니다.
매매 판단은 서비스 본체와 분리된 자동매매 에이전트가 담당해, 관리와 실행의 책임을 나눴습니다.
에이전트는 현재 정해진 시각에 파이프라인을 실행하는 스케줄 기반이며, 스스로 판단하도록 고도화하고 있습니다.

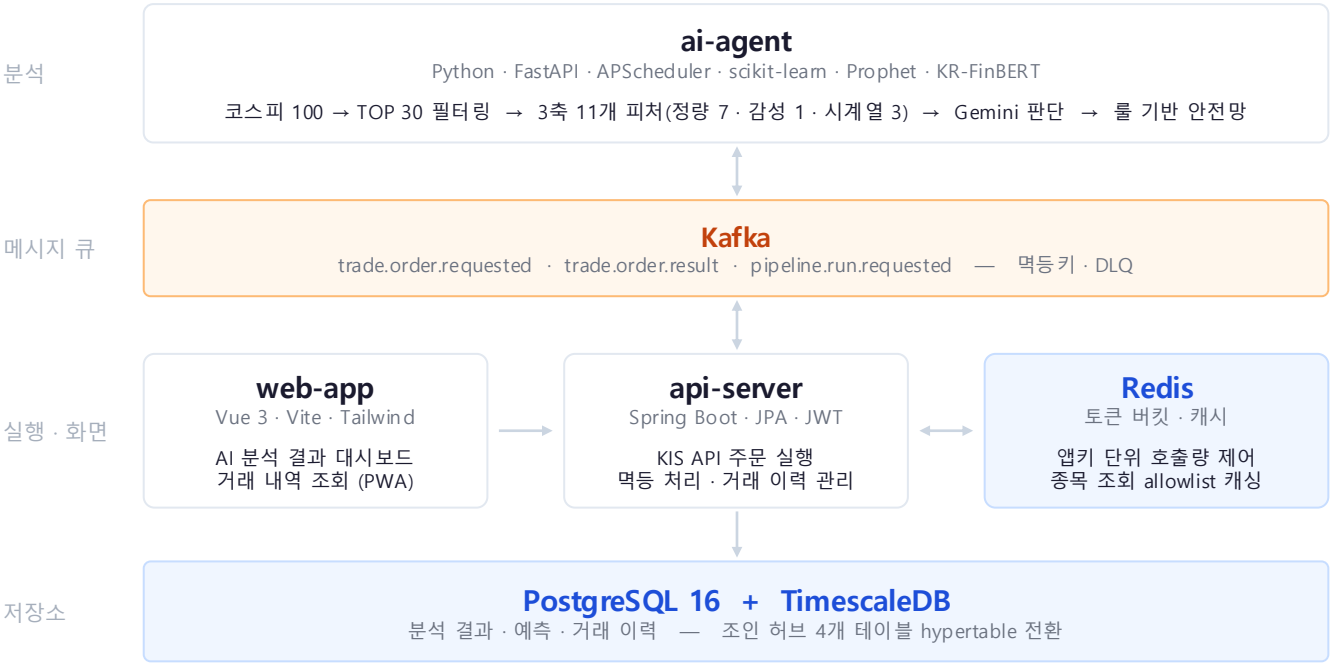
역할

- 전체 시스템 설계 — 3개 모듈 구성, 모듈 간 메시지 계약, 데이터 파이프라인 설계
- ai-agent — 3축 분석 파이프라인 구현(정량 7 · 감성 1 · 시계열 3), Gemini 연동, 룰 기반 안전망 설계
- api-server — Spring Boot 인증·주문 실행 서버, KIS API 연동, 멱등 처리, Liquibase 스키마 버전 관리
- web-app — Vue 3 PWA 대시보드, 3축 분석 결과 · 판단 근거 · 거래 내역 화면 구성

Skills

- Python · FastAPI · Spring Boot · JPA · JWT · Liquibase · Vue 3
- PostgreSQL · TimescaleDB · Kafka · Redis · Docker · Gemini API · KIS API

시스템 아키텍처



ai-agent는 APScheduler로 매일 정해진 시각에 파이프라인을 실행하고, DB 스키마는 api-server의 Liquibase로 버전 관리합니다.

1. 누적되는 시계열 데이터의 조인 비용 증가

GitHub

AS-IS

화면 데이터를 조립하기 위해 stock_filter_score를 허브로 5개 테이블을 LEFT JOIN하는 구조. 데이터가 연 단위로 누적될수록 조인 비용이 계속 불어남.

TO-BE

기존 SQL·JOIN·JPA를 그대로 유지할 수 있는 TimescaleDB를 택해 전환하고, 조인 허브 4개 테이블을 hypertable로 변환해 시간 기반 자동 파티셔닝. 30일 청크 압축 정책과 갱신 쿼리의 score_date 조건 추가로 대상 청크를 좁힘. 단일 날짜 조회 4.2배, 데이터 정리 28배(무중단), 저장 공간 33% 절감.

핵심 키워드: Hypertable 전환, 시간 기반 파티셔닝, 압축 정책

2. 주문 유실과 파이프라인 중복 실행

GitHub

AS-IS

ai-agent가 api-server를 동기 HTTP로 호출해 실패 시 재시도 없이 주문이 유실되고 감사 로그도 남지 않음. 스케줄과 수동 트리거가 락 없이 동시 실행돼 중복 매수가 나갈 수 있었음.

TO-BE

Kafka 토픽 3개로 주문 요청·실행 결과·파이프라인 트리거를 분리하고 단일 컨슈머로 순차 처리. 멱등키(userId:stockCode:tradeDate:side)로 동일 메시지가 3회 발행돼도 KIS 호출 1회를 보장. 주문이 나갔을 리 없는 KIS 미접촉 오류만 지수 백오프로 재시도하고, 호출 이후 실패는 중복 주문 위험 때문에 DLQ로 격리. 동기 대기 수십 분 → 수 ms.

핵심 키워드: 멱등키 설계, 재시도 가능 오류 분리, DLQ 격리

3. 외부 API Rate Limit 초과와 중복 호출

GitHub

AS-IS

다른 사용자가 같은 종목을 조회해도 매번 KIS를 다시 호출. rate limit 초과로 거부되면 컨슈머가 'KIS 호출 단계 실패'로 분류해 재시도 없이 DLQ로 보냄.

TO-BE

앱키 단위 Redis 토큰 버킷으로 소켓을 열기 전에 거부해 'KIS 미접촉'을 보장하고, 전용 예외 타입으로 분리해 재시도 가능한 단계로 되돌림. 조회성 TR_ID 5개만 allowlist 캐싱(현재가·호가 30초, 재무 24시간). 종목 상세 30명 조회 기준 150회 → 4회(97.3% 절감).

핵심 키워드: 앱키 단위 토큰 버킷, allowlist 캐싱, 예외 분리 재시도

회고

세 이슈 모두 '기능은 되는데 규모나 동시성이 끼면 깨진다'는 공통점이 있었습니다. 동기 HTTP 호출은 한 건씩 테스트할 때 문제가 없었고, 조인 쿼리도 데이터가 적을 땐 빨랐습니다. 정상 경로만 확인하고 넘어가면 실패·중복·누적 조건에서 뒤늦게 드러난다는 것을 배웠습니다.
아쉬운 점은 rate limit 기본값과 캐시 TTL을 KIS 공식 수치가 아니라 추정치로 잡았다는 것입니다. 실제 한도를 확인해 근거 있는 값으로 교체하는 것이 다음 과제입니다.

외래관광객의 비수도권 방문에 미치는 영향요인 분석

GitHub

2026. 03 · 서강대 AI·SW대학원 통계기반데이터분석 과제 · 투고 준비 중

5인 공저 · 「외래관광객조사」 2018 · 2019 · 2023 · 2024 원자료 (표본 64,957명)

개요

코로나19 이후 방한 관광은 회복됐지만 서울 방문율은 76.4% → 80.3%로 늘었습니다.
수도권만 방문한 경우와 비수도권 방문이 포함된 경우를 나눠 이항 로지스틱 회귀로 분석했습니다.
인구통계(국적·성별·연령)와 여행행태(여행유형·단독여행·체류기간)를 함께 투입한 다변량 모형입니다.

역할

- 변수 정의 — 코드북과 실제 데이터를 대조해 4개년의 상이한 코드 체계를 단일 변수 정의로 매핑
- 전처리 — 연도별 수집 방식·가중치 구조 차이 조정, 결측 처리 기준 수립 (표본 25% 감소 구간 검토)
- 재현성 검증 — 샘플 데이터셋으로 각자 독립 분석한 뒤 산출값을 교차 대조해 결과 편차 확인

주요 결과

- 체류기간이 최대 영향 요인 (OR 2.02) — 여행유형 1.58 · 단독여행 0.62 · 연령 1.04
- 국가군 부분 유의 (서구권 1.12 ↑ / 일본 0.88 ↓) · McFadden R^2 0.051, AUC 0.657
- 인과가 아닌 연관으로 해석 — 표현도 '결정요인'에서 '영향요인'으로 수정