|  |  |
| --- | --- |
| **진기태**  **Backend Engineer** | Phone | 010-5570-9143  Email | msjinkitea@gmail.com  Github | <https://github.com/sensesis>  Blog | <https://kittae.tistory.com/> |

Summary

**-** 저는 문제를 단순히 해결하는 개발자보다는, **구조적인 접근과 설계 개선에 집중하는 백엔드 엔지니어**입니다.

**-** 맡은 일을 끝까지 책임지는 성격이며, 협업에서의 **문서화와 커뮤니케이션**을 중요하게 생각합니다.

**-** 특히 **데이터 흐름과 안정성**에 큰 관심을 갖고 있어, 대규모 데이터를 다루는 프로젝트에서 **Spring Batch 기반으로 270만 건의 데이터를 정제·적재**하고, PostgreSQL의 **쿼리 최적화 및 인덱스 설계**를 통해 스캔비용을 감소시킨 경험이 있습니다.

**-** 3개월 이상 소요될 수 있는 프로젝트를 **5주** 만에 기획부터 배포까지 완수하며, **우선순위를 조율**하고 **팀의 의사결정**과 **커뮤니케이션**을 주도한 경험이 있습니다.

Experience

**Rebo** 2025.03 ~ 2025.06 [GitHub](https://github.com/project-Rebo)

블루오션 · 레드오션을 판단하는 상권 분석 웹 서비스

**Backend | Spring, PostgreSQL**

**Infra | Docker,** **On-premise(Linux), Prometheus, Grafana**

**기여 내용 |**

Spring Batch 기반 대용량 데이터 정제 및 적재 파이프라인 구축

**문제:**

- 총 270만 건에 이르는 CSV 데이터를 로컬 파티셔닝 방식으로 적재하는 과정에서, 고유 키 중복 및 병렬 처리로 인한 동시성 충돌 문제가 발생함.

- JPA 기반 처리 방식은 Flush 및 Dirty Checking 오버헤드로 인해 성능 저하와 트랜잭션 실패가 빈번히 발생했으며, 이로 인해 배치 파이프라인의 신뢰성 및 속도에 문제가 생김. [(Blog)](https://kittae.tistory.com/45)

**해결:**

**- 임시 테이블(Staging Table) 도입:** 메인 테이블 대신 임시 테이블에 데이터를 우선 적재하여 트랜잭션 경합을 방지하고, 이후 정제된 데이터만 메인 테이블로 마이그레이션하는 이중 단계 구조로 설계함.

**-** **JPA -> JDBC 전환: 대량 데이터 삽입 시 발생하는 JPA의 퍼포먼스 한계를 극복하기 위해, JdbcBatchItemWriter를 사용한 JDBC 기반 대량 일괄 처리 방식으로 전환함.**

**-** **ID 기반 병렬 파티셔닝: Spring Batch의 Partitioner를 활용하여 CSV 데이터의 고유 ID 범위를 기준으로 병렬 파티션을 구성, 각 스레드가 독립적으로 데이터를 처리하도록 설정함.**

**결과:**

- **처리 성능 개선: 기존 JPA 기반 방식 대비 데이터 적재시간을 50% 개선함. (270만건 기준 약4분 30초 -> 2분 30초)**

- **신뢰성 향상: 동시성 충돌로 인한 배치 실패율을 0%로 감소시켜, 모든 데이터가 안정적으로 적재되는 파이프라인을 구현함.**

- **아이텍처 재사용성 확보: Staging -> 정제 -> Main 적재 구조를 모듈화하여, 향후 유사한 데이터 정제 요구사항에 즉시 대응 가능한 파이프라인을 구축함**상권분석 검색 기능 성능 개선 및 리소스 최적화

**문제:**

- 사용자에게 유연한 검색 경험을 제공하기 위해 대분류, 중분류, 소분류, 이름 조건을 동적으로 조합하는 구조(CASE, OR, %LIKE%)로 쿼리를 설계함.

- 이로 인해 PostgreSQL이 Seq Scan을 선택하면서, 불필요한 테이블 스캔으로 이어져 **검색 응답 속도 지연 및 스캔 비용 증가** 문제가 생김. [(Notion)](https://www.notion.so/21845f92baf980c2b78dd10c02078a2b?source=copy_link)

**해결:**

**- 쿼리 최적화**: 초기 쿼리에서 **DISTINCT** 및 **SELECT \*** 사용으로 불필요한 스캔 비용이 발생했으나, **DISTINCT제거** 및 **필수 컬럼 추출 방식**을 통해 데이터 접근 범위를 최소화함.

- **인덱스 도입:** 기존 쿼리는**CASE, OR, %LIKE%** 조건으로 인해 **Seq Scan**을 유발 했으나, 검색 조건 컬럼(category, name)에 복합 인덱스를 설계하여 쿼리 실행 계획을 **Index Scan**으로 전환함.

- **캐싱 도입:** Caffeine 라이브러리를 활용하여 **Time To Live(TTL)** 5분의 로컬 캐시 전략을 구현함. 동일한 검색 쿼리가 반복적으로 발생할 경우 데이터베이스에 직접 접근하지 않고 캐시된 결과를 즉시 반환하도록 시스템을 구성함.

**결과:**

- **응답 속도 개선: DISTINCT 제거 및 필수 컬럼 추출, 복합 인덱스 설계를 통해 불필요한 스캔을 줄이고 실행 계획을 최적화 하여, 사용자 요청에 대한 평균 응답 시간을 단축함.**

- **자원 사용 최적화:** 동일한 검색 쿼리가 반복될 경우, Caffeine 기반의 TTL 캐시를 통해 DB 조회를 우회함으로써 쿼리 실행 횟수를 감소시키고, 서버 연산 자원(CPU, 메모리) 소모를 절감함.

- **서비스 안정성 확보:** 정제된 쿼리와 캐싱 전략을 통해, 고동시 요청 환경에서도 검색 서비스의 안정성과 처리 성능을 유지할 수 있는 구조를 완성함.

**DevSketch** 2024.12 ~ 2025.02 [Medium](https://medium.com/@qkrb8019/%ED%94%84%EB%A1%9C%EC%A0%9D%ED%8A%B8-%EC%84%A4%EA%B3%84%EB%B6%80%ED%84%B0-%EC%83%9D%EC%84%B1%EA%B9%8C%EC%A7%80-ai%EA%B0%80-%EB%8F%84%EC%99%80%EC%A3%BC%EB%8A%94-devsketch-2024-siliconvalley-winter-bootcamp-8f69595ec7a1) [GitHub](https://github.com/sensesis/DevSketch)

사용자 요청에 따라 설계부터 배포까지 모든 과정을 자동화 해주는 AI 기반 웹 서비스

**Backend | Django, RabbitMQ, Celery, MySQL**

**Infra | Docker, Traefik,** **GCP(GCE, Google Cloud), Prometheus, Grafana, Jenkins**

**기여 내용 |**

AI 설계 기능 구현

**문제:****-** 사용자가 요청한 ERD, 시스템 아키텍처, API 문서 등을 자동 생성하는 기능은 단일 요청당 3개의 복합 문서를 생성해야 했고, 동기 방식에서는 평균 62초 이상 소요되는 병목이 발생함.

**해결:**

- 작업 부하를 비동기화하기 위해 **Celery + RabbitMQ** 기반의 작업 큐 구조를 설계하고, **문서 단위로 태스크를 분할**하여 병렬 처리되도록 조정함.

- 대량 요청 시에도 API가 즉시 응답하도록 개선되었고, 병렬 문서 처리로 평균 응답 시간이 **62초 → 34초**로 약 45% 감소함.[(Blog)](https://kittae.tistory.com/39)

가상 환경 자동화(Docker-in-Docker, DinD)

**문제:**

- 프로젝트 초기에는 개발자마다 로컬 환경 구성이 달라 테스트 환경 불일치 문제가 빈번히 발생했고, 운영 환경과의 차이로 인해 배포 시 의도치 않은 오류가 발생했음.

**해결:**

**-** 컨테이너 내에 또 다른 Docker 엔진을 구동하는 **Docker-in-Docker(DinD)** 기술을 도입하여, 사용자 요청에 따라 독립적인 가상 실행 환경을 생성하도록 설계함.

**-** 각 컨테이너는 요청된 코드나 스크립트를 동일한 환경에서 실행할 수 있도록 미리 프로비저닝된 이미지를 기반으로 시작되며, 실행 후 자원 낭비를 막기 위한 **컨테이너 타임아웃 정책(30분**)을 병행 적용함.

**결과:**

**-** 사용자별 환경 구성 시간이 **0초로 수렴**하며, 운영 환경과 일치하는 테스트 환경에서의 사전 검증이 가능해졌고, 배포 후 오류 발생률이 현저히 감소함.

CI/CD 파이프라인 구축

**문제:**

**-** 수동 빌드 및 배포로 인한 시간 낭비, 휴먼 에러 발생, 긴급 배포 시 서비스 중단 위험이 존재함.

- 컨테이너 리소스 사용량에 대한 실시간 모니터링 체계가 없어, 시스템 부하에 대한 대응이 어려운 상태였음.

**해결:**

**-** GitHub → Jenkins → Docker로 이어지는 **CI/CD 파이프라인**을 구축하고, 푸시 이벤트 기반으로 자동 빌드 및 배포가 수행되도록 구성함.

**-** **cAdvisor + Grafana + Prometheus** 스택을 통해 Docker 컨테이너별 리소스(CPU, Memory) 사용량을 실시간으로 시각화하고, 과부하 발생 시 알림을 받을 수 있도록 설정함.

**- Traefik을 활용해 서비스간 라우팅을 자동화하고, Jenkins + Docker 기반의 자동 빌드/배포 환경을 구축하여 서비스 중단 시간을 최소화한 배포 시스템을 구현함.**

**결과:**

- 배포 자동화로 반복적인 수작업이 제거되어, 빌드-배포까지 소요 시간이 줄어들어 **팀의 운영 및 관리 효율성이 크게 향상됨**.

- 운영 상태에 대한 가시성을 확보하여 장애 조기 대응이 가능해졌고, 시스템 안정성이 강화됨.

**Skills**

**Language |** Java, Python

**Library / Framework |** Spring Boot, Django

**Database |** MySQL, PostgreSQL

**Monitoring |** Prometheus, Grafana

**Development Tools** **|** Docker, Nginx, Traefik, Jenkins

**ETC |** RabbitMQ, Celery

**Activities**

**Techeer [테커]** 9기 | 2025.02 ~ 현재

- 실리콘밸리 개발자의 SW 개발자 커리어 그룹

- 프로젝트, 스터디, 네트워킹 행사 등 다양한 활동 참여

**부트캠프**

- [티타임즈 x Techeer] 2024 하계 실리콘밸리 SW 부트캠프 2024.06 ~ 2024.08

- [티타임즈 x Techeer] 2024 동계 실리콘밸리 SW 부트캠프(1등 수상 🥇) 2024.12 ~ 2025.02

**한성대학교**

-[한성SW중심대학 페스티벌 캡스톤 전시회] PPURISAM(우수상) 2024.09 ~ 2024.11

-[한성대학교 캡스톤디자인 작품발표회] Adcanvas(장려상) 2025.03 ~ 2024.06

**Education**

한성대학교 컴퓨터공학과 | 2020.03 - 2026.02(졸업예정)