

MSA 전환을 위한 데이터베이스 현황 분석 및 분리 전략 가이드

1. 개요

마이크로서비스 아키텍처(MSA) 전환에서 데이터베이스 분리는 가장 중요하면서도 복잡한 과정 중 하나입니다. 기존 모놀리식 시스템의 통합 데이터베이스를 서비스별로 분리하는 것은 단순한 기술적 작업이 아니라 비즈니스 도메인 이해와 데이터 의존성 분석을 기반으로 한 전략적 접근이 필요합니다.

본 가이드는 MSA 전환을 위한 데이터베이스 현황 분석 방법론과 실무에 적용 가능한 분리 전략을 제공합니다. Database per Service 패턴을 중심으로 한 체계적인 접근 방법과 실제 구현 시 고려해야 할 사항들을 다룹니다.

2. 데이터베이스 현황 분석 프레임워크

2.1 분석 목표 설정

데이터베이스 현황 분석은 다음과 같은 목표를 달성하기 위해 수행됩니다:

- 현재 데이터베이스 구조와 서비스 간 의존성 파악
- MSA 분리 시 데이터 경계 및 소유권 정의
- 분리 과정에서 발생할 수 있는 리스크 식별
- 점진적 분리 전략 수립을 위한 우선순위 결정

2.2 데이터베이스 현황 분석 템플릿

데이터베이스 현황 분석 기본 템플릿			
분석 항목	상세 내용	수집 방법	비고
데이터물 정보	데이터명, 종도, 데이터 불용, 상황물	DB 스키마 분석, 시스템 문서 검토	비즈니스 중요도 표시
데이터물 간 관계	FK 관계, 조인 제약, 트랜잭션 범위	ERD 분석, 쿼리 로그 분석	의존성 강도 평가
접근 패턴	CRUD 비율, 동시성 요구사항, 성능 특성	APM 도구, 쿼리 성능 분석	서비스별 분류
데이터 일관성	트랜잭션 경계, 일관성 요구사항	비즈니스 프로세스 분석	ACID vs BASE 판단
기술적 제약	스토리지 용량, 인덱스, 뷰, 프로시저	DB 자체 분석	분리 시 영향도 평가

2.3 도메인별 데이터 분류

DDD(Domain-Driven Design) 원칙에 따라 데이터를 비즈니스 도메인별로 분류하는 것이 MSA 분리의 첫 번째 단계입니다.

도메인별 데이터 분류 템플릿				
도메인	핵심 엔티티	관련 테이블	비즈니스 기능	외부 의존성
사용자 관리	User, Role, Permission	users, roles, permissions, user_roles	인증, 인가, 프로필 관리	없음
주문 관리	Order, OrderItem	orders, order_items, payment_info	주문 생성, 결제 처리	사용자, 상품, 결제
상품 관리	Product, Category	products, categories, inventory	상품 등록, 재고 관리	없음
결제 관리	Payment, Invoice	payments, invoices, refunds	결제 처리, 환불 관리	주문, 사용자

3. 데이터베이스 분리 전략

3.1 Database per Service 패턴

Database per Service 패턴은 각 마이크로서비스가 자체 데이터베이스를 소유하고 관리하는 MSA의 핵심 원칙입니다. 이 패턴은 다음과 같은 이점을 제공합니다:

- 느슨한 결합:** 서비스 간 데이터베이스 의존성 제거
- 기술적 자유도:** 서비스별 최적화된 데이터베이스 선택 가능
- 독립적 확장:** 서비스별 개별 확장 및 성능 최적화
- 장애 격리:** 한 서비스의 DB 장애가 다른 서비스에 미치는 영향 최소화

주의사항: Database per Service 패턴 적용 시 분산 트랜잭션, 데이터 일관성, 복잡한 조인 쿼리 등의 문제가 발생할 수 있습니다. 이를 해결하기 위해 SAGA 패턴, CQRS, API Composition 등의 패턴을 함께 고려해야 합니다.

3.2 분리 접근 방법

데이터베이스 분리는 다음 세 가지 접근 방법 중 하나를 선택할 수 있습니다:

3.2.1 데이터베이스 우선 분리

애플리케이션 코드는 유지한 채 데이터베이스를 먼저 분리하는 방식입니다.

장점	단점	적용 시나리오
<ul style="list-style-type: none">서비스 영향 최소화변경 통제 용이성성능 영향 사전 파악	<ul style="list-style-type: none">단기간 이익 제한적복잡한 데이터 처리 필요개발 기간 연장	<ul style="list-style-type: none">성능 민감성 미수 우려정신적 집중 필요재가시 시스템 유지

3.2.2 애플리케이션 코드 우선 분리

데이터베이스는 공유한 채 애플리케이션을 먼저 분리하는 방식입니다.

장점	단점	적용 시나리오
<ul style="list-style-type: none">빠른 서비스 분리독립 배포 가능개발 생산성 향상	<ul style="list-style-type: none">데이터 결합도 유지긴밀한 독립성 부족장애 전파 위험	<ul style="list-style-type: none">빠른 서비스 분리 필요데이터 분리 복잡도 높음점진적 전환 선호

3.2.3 동시 분리

데이터베이스와 애플리케이션을 동시에 분리하는 방식입니다.

권장하지 않음: 동시 분리는 높은 리스크를 수반하며 예상보다 오랜 시간이 소요됩니다. 가능한 한 단계적 접근을 권장합니다.

3.3 분리 패턴별 구현 방법

3.3.1 논리적 분리 (Schema per Service)

단일 데이터베이스 인스턴스 내에서 서비스별로 스키마를 분리하는 방법입니다.

```
-- 서비스별 스키마 분리 예시
CREATE SCHEMA user_service;
CREATE SCHEMA order_service;
CREATE SCHEMA product_service;

-- 서비스를 사용자 지정 생성
CREATE USER 'user_service_user'@'%' IDENTIFIED BY 'password';
CREATE USER 'order_service_user'@'%' IDENTIFIED BY 'password';
CREATE USER 'product_service_user'@'%' IDENTIFIED BY 'password';

-- 스키마별 권한 부여
GRANT ALL PRIVILEGES ON user_service.* TO 'user_service_user'@'%';
GRANT ALL PRIVILEGES ON order_service.* TO 'order_service_user'@'%';
GRANT ALL PRIVILEGES ON product_service.* TO 'product_service_user'@'%';
```

3.3.2 물리적 분리 (Database per Service)

각 서비스가 독립된 데이터베이스 인스턴스를 갖는 방법입니다.

분리 레벨	구현 방법	장점	단점
데이터물 분리	서비스별 전용 테이블	낮은 오버헤드	소유권 불분명
스키마 분리	서비스별 전용 스키마	명확한 소유권	단일 장애영
인스턴스 분리	서비스별 DB 서버	완전한 독립성	높은 운영 비용

4. 데이터베이스 의존성 분석

4.1 의존성 분석 체크리스트

데이터물 간 의존성 분석 <ul style="list-style-type: none">□ 외래키(FK) 관계 매핑□ 조인 쿼리 패턴 분석□ 트랜잭션 경계 식별□ 데이터 참조 빈도 측정□ 순환 참조 여부 확인 비즈니스 프로세스 의존성 <ul style="list-style-type: none">□ 업무 흐름별 데이터 사용 패턴□ 트랜잭션 일관성 요구사항□ 실시간 처리 vs 배치 처리 구분□ 보고서 및 분석 쿼리 영향도 기술적 의존성 <ul style="list-style-type: none">□ 저장 프로시저 및 함수 의존성□ 뷰(View) 사용 현황□ 트리거 및 제약조건□ 인덱스 최적화 현황

4.2 의존성 해결 방안

4.2.1 강한 의존성 해결

서비스 간 강한 의존성이 있는 경우 다음과 같은 해결 방안을 고려할 수 있습니다:

의존성 유형	해결 방안	구현 방법	주의사항
트랜잭션 경계	SAGA 패턴	이벤트 기반 분산 트랜잭션	결구적 일관성 수용
실시간 데이터 조회	API Composition	애플리케이션 레벨 조인	성능 오버헤드 고려
복잡한 쿼리	CQRS	읽기 전용 뷰 구성	데이터 동기화 필요
침묵 데이터	데이터 복제	이벤트 기반 동기화	데이터 일관성 관리

4.2.2 약한 의존성 해결

약한 의존성은 다음과 같은 방법으로 해결할 수 있습니다:

- 데이터 비정규화:** 서비스 내부에서 필요한 데이터를 중복 저장
- 캐시 활용:** 자주 참조하는 데이터를 캐시로 관리
- 배치 동기화:** 비실시간 데이터는 배치로 동기화
- 이벤트 발행:** 데이터 변경 시 이벤트를 발행하여 관련 서비스에 알림

5. 실무 적용 가이드

5.1 분리 순서 결정

데이터베이스 분리 시 다음 기준에 따라 우선순위를 결정합니다:

분리 우선순위 평가 매트릭스			
평가 기준	높음 (H)	보통 (M)	낮음 (L)
비즈니스 중요도	핵심 업무 영역	중요 업무 영역	부가 업무 영역
기술적 복잡도	단순한 CRUD	중간 복잡도	복잡한 비즈니스 로직
데이터 의존성	독립적	약한 의존성	강한 의존성
변경 빈도	변경 빈번	보통	변경 드물음
팀 역량	숙련도 높음	보통	숙련도 낮음

5.2 단계별 분리 프로세스

1단계: 준비 및 분석

- 현황 조사:** 데이터베이스 스키마, 데이터 관계, 쿼리 패턴 분석
- 도메인 정의:** 비즈니스 도메인별 경계 설정
- 의존성 분석:** 데이터 및 서비스 의존성 매핑
- 분리 계획:** 우선순위 및 일정 수립

2단계: 파일럿 프로젝트

- 후보 선정:** 독립성이 높고 복잡도가 낮은 서비스 선택
- 프로토타입 개발:** 분리 패턴 검증
- 성능 테스트:** 분리 후 성능 영향 측정
- 교훈 도출:** 경험을 바탕으로 프로세스 개선

3단계: 점진적 분리

- 단계별 실행:** 우선순위에 따라 순차적 분리
- 검증 및 롤백:** 각 단계별 검증 후 진행
- 모니터링:** 성능 및 안정성 지속 모니터링
- 최적화:** 분리 후 성능 최적화 수행

5.3 분리 후 운영 고려사항

5.3.1 모니터링 체계

- 서비스별 성능 모니터링:** 응답시간, 처리량, 에러율
- 데이터베이스 모니터링:** 연결 수, 쿼리 성능, 리소스 사용률
- 분산 트랜잭션 모니터링:** SAGA 패턴 실행 상태
- 데이터 일관성 모니터링:** 서비스 간 데이터 동기화 상태

5.3.2 백업 및 복구

- 서비스별 백업 전략:** 각 서비스의 특성에 맞는 백업 정책
- Point-in-time 복구:** 분산 환경에서 일관성 있는 복구 방안
- 재해 복구 계획:** 서비스별 우선순위를 고려한 복구 순서

5.3.3 보안 관리

- 접근 제어:** 서비스별 데이터베이스 접근 권한 관리
- 암호화:** 데이터 전송 및 저장 시 암호화
- 감사:** 데이터 접근 및 변경 이력 추적

6. 위험 관리 및 대응 방안

6.1 주요 위험 요소

위험 유형	발생 가능성	영향도	대응 방안
데이터 일관성 문제	높음	높음	SAGA 패턴, 분산 트랜잭션
성능 저하	중간	높음	캐시 전략, 쿼리 최적화
분리 실패	낮음	높음	롤백 계획, 단계적 접근
운영 복잡도 증가	높음	중간	자동화 도구, 모니터링

6.2 대응 전략

위험 완화 체크리스트 <ul style="list-style-type: none">□ 롤백 계획 수립 및 테스트□ 단계별 검증 포인트 설정□ 성능 기준선 설정 및 모니터링□ 비상 연락망 구성□ 장애 대응 매뉴얼 작성□ 정기적인 복구 훈련 실시

7. 결론 및 권고사항

7.1 성공 요인

MSA 데이터베이스 분리 성공을 위한 핵심 요인들은 다음과 같습니다:

- 점진적 접근:** 빅뱅 방식이 아닌 단계적 분리
- 도메인 중심 설계:** 서비스 비즈니스 도메인을 기반으로 한 경계 설정
- 철저한 준비:** 현황 분석과 의존성 파악
- 팀 역량:** MSA 패턴에 대한 이해와 실무 경험
- 모니터링:** 분리 과정과 운영 상태의 지속적 모니터링

7.2 최종 권고사항

- 현실적 목표 설정:** 조직의 역량과 상황에 맞는 분리 범위 결정
- 파ilot 프로젝트:** 작은 규모로 시작하여 경험 축적
- 지속적 개선:** 분리 후에도 지속적인 최적화 수행
- 조직 문화:** 분산 시스템 운영에 적합한 조직 문화 구축
- 기술 투자:** 모니터링, 자동화 도구에 대한 투자

본 가이드에서 제시한 방법론을 활용하여 조직의 상황에 맞는 데이터베이스 분리 전략을 수립하고 실행하시길 바랍니다. MSA 전환은 단순한 기술적 변화가 아니라 조직과 프로세스의 전반적인 변화를 수반하므로, 충분한 준비와 단계적 접근을 통해 성공적인 전환을 이루시길 권합니다.

추가 참고 자료 <ul style="list-style-type: none">MSA 성숙도 체크리스트API 엔드포인트 현황 분석 가이드서비스 간 통신 패턴 가이드분산 트랜잭션 구현 방법데이터 동기화 전략 수립 가이드
