

실전! MSA 개발 가이드

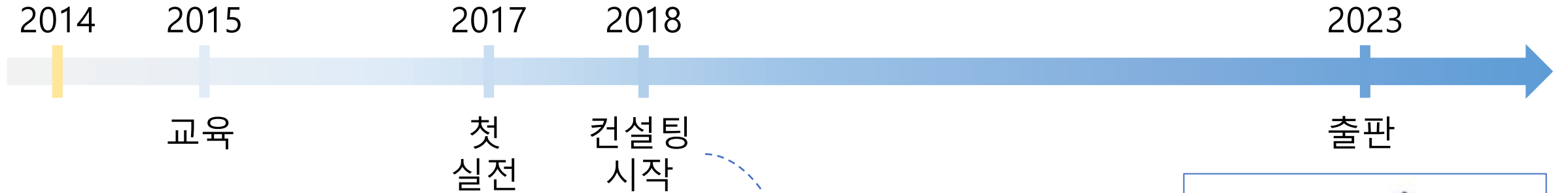
DB를 분리해서 개발한다고?

API로?

트랜잭션은?


소개

마이크로서비스를 적용할 때 가이드하고 함께 개발하고 있습니다.

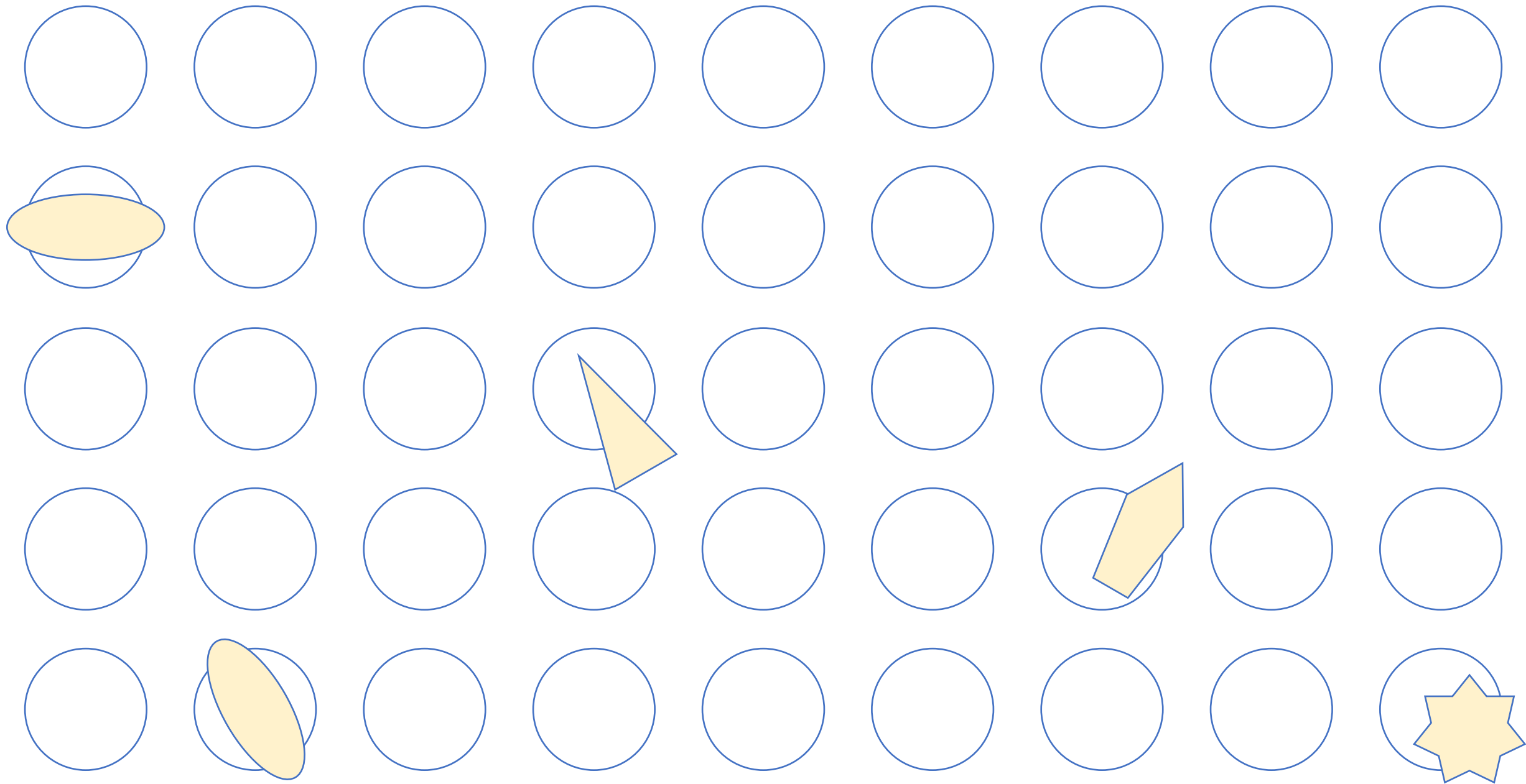


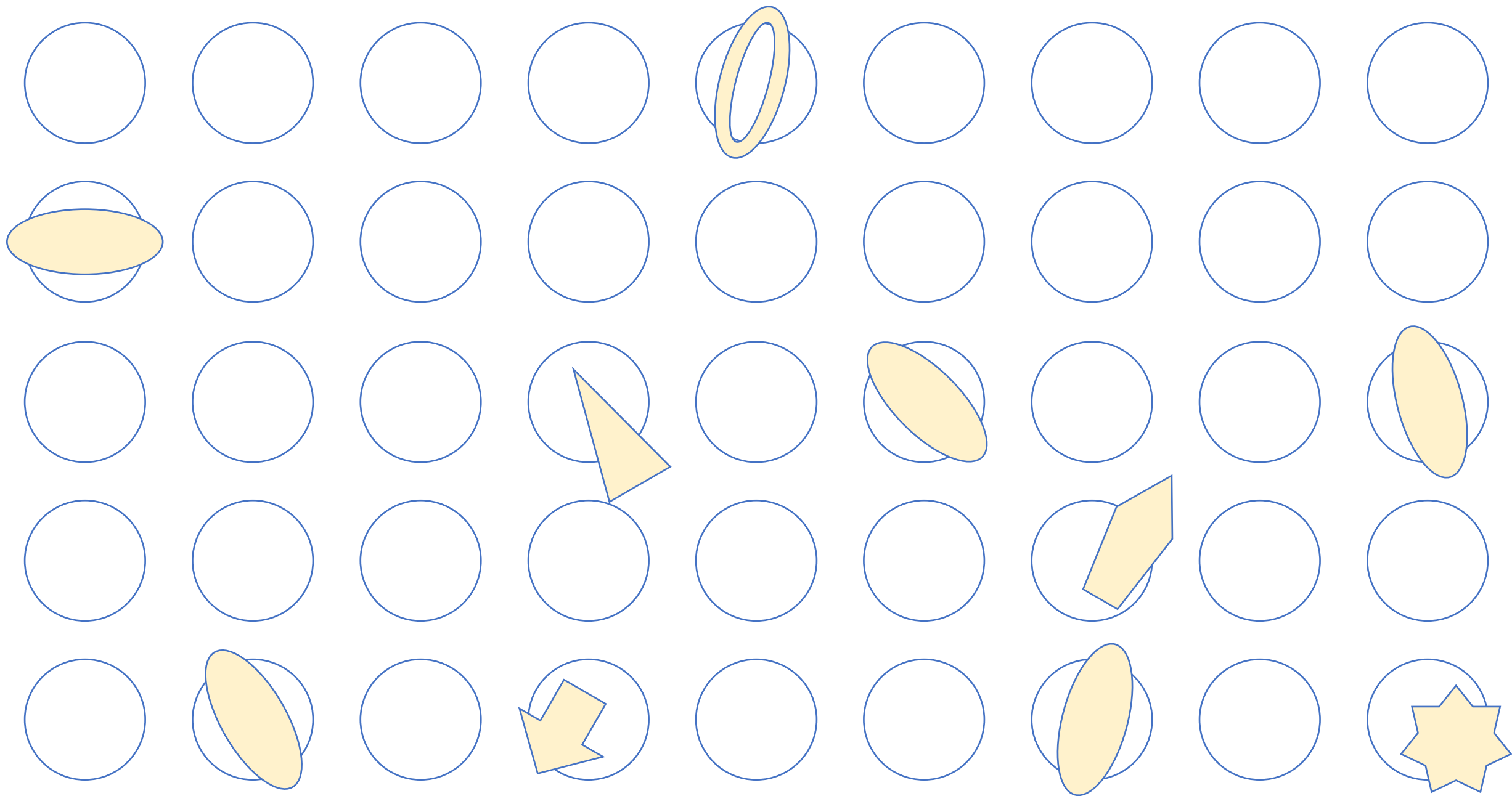
- 시스템에 적합한지 판단
- 적용 목표에 따라 서비스 설계
- 개발 가이드 및 함께 개발



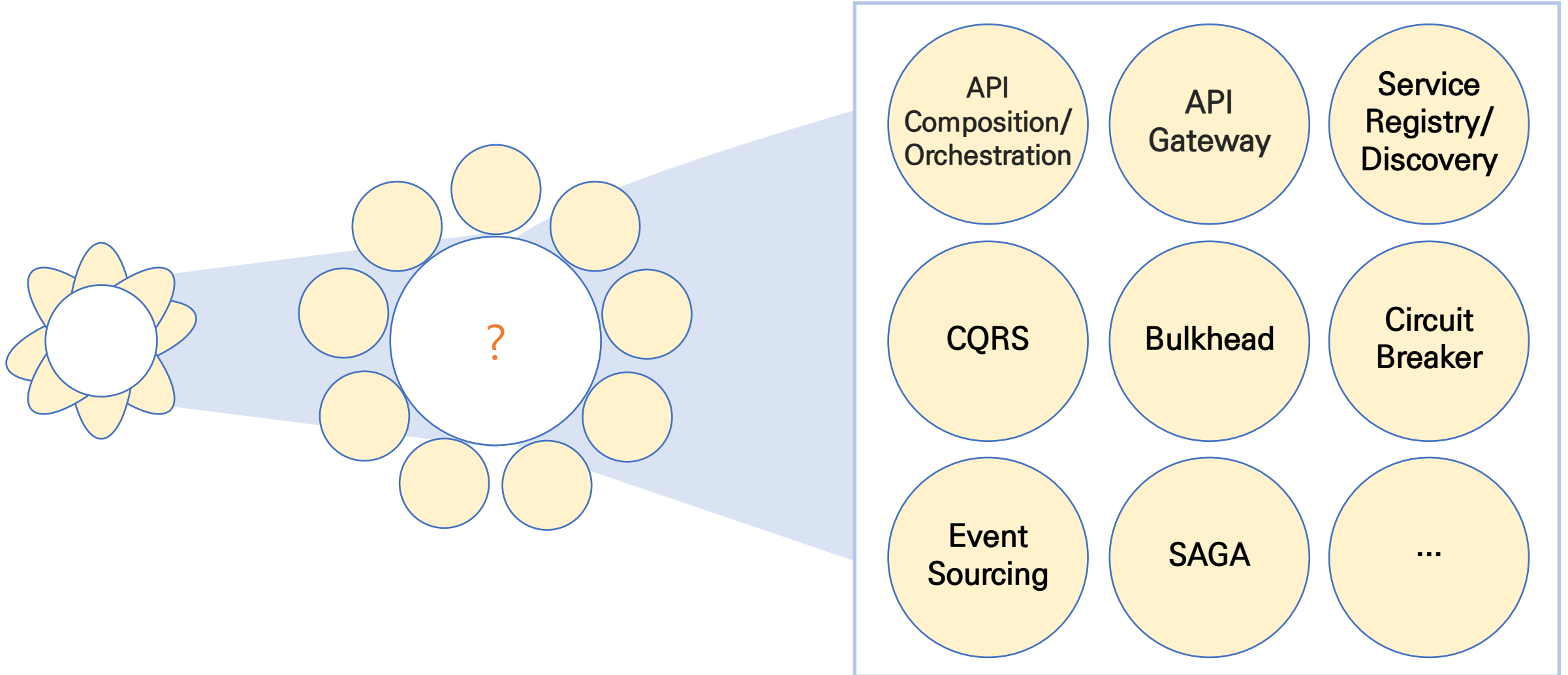


일반적인
시스템



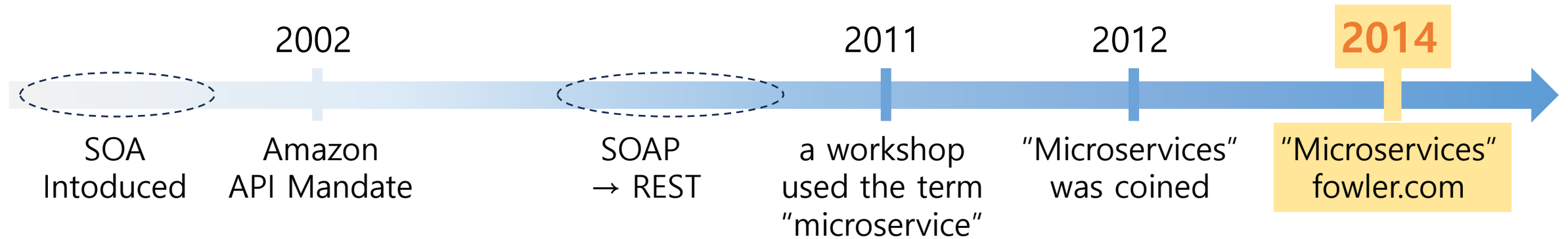


새로운 시스템들의 특징과 기술을 모아 “마이크로서비스”라 소개



Brief History

2014년에 파울러 닷컴에 소개되면서 더 널리 퍼지기 시작



Fred George



Adrian Cockcroft



James Lewis

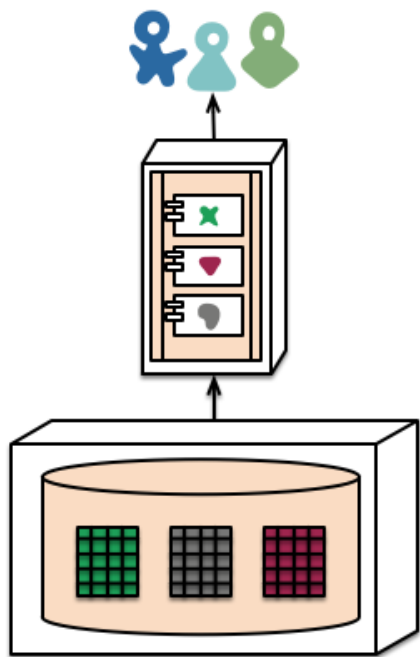


Martin Fowler

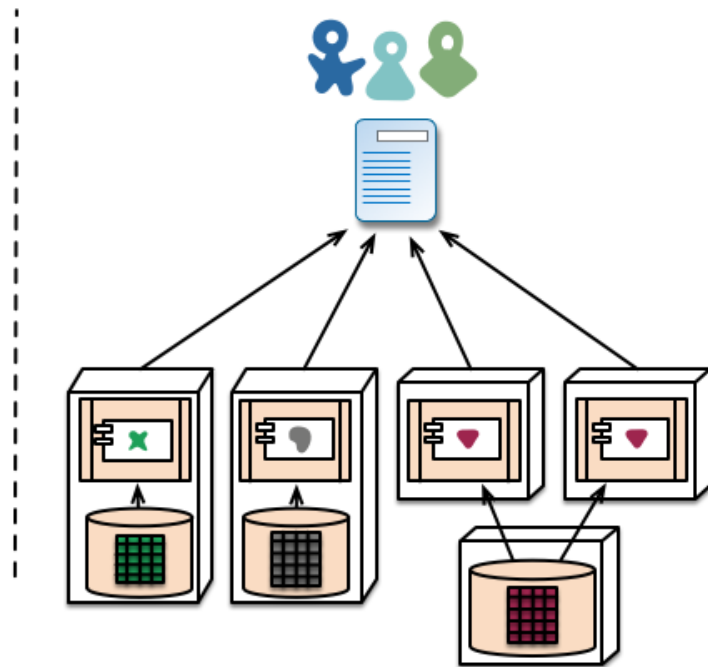
마이크로서비스 아키텍처의 정의

잘 정리되었지만

충분히 구체적이진 않은 정의



monolith - single database



microservices - application databases

마이크로서비스 아키텍처의 특징

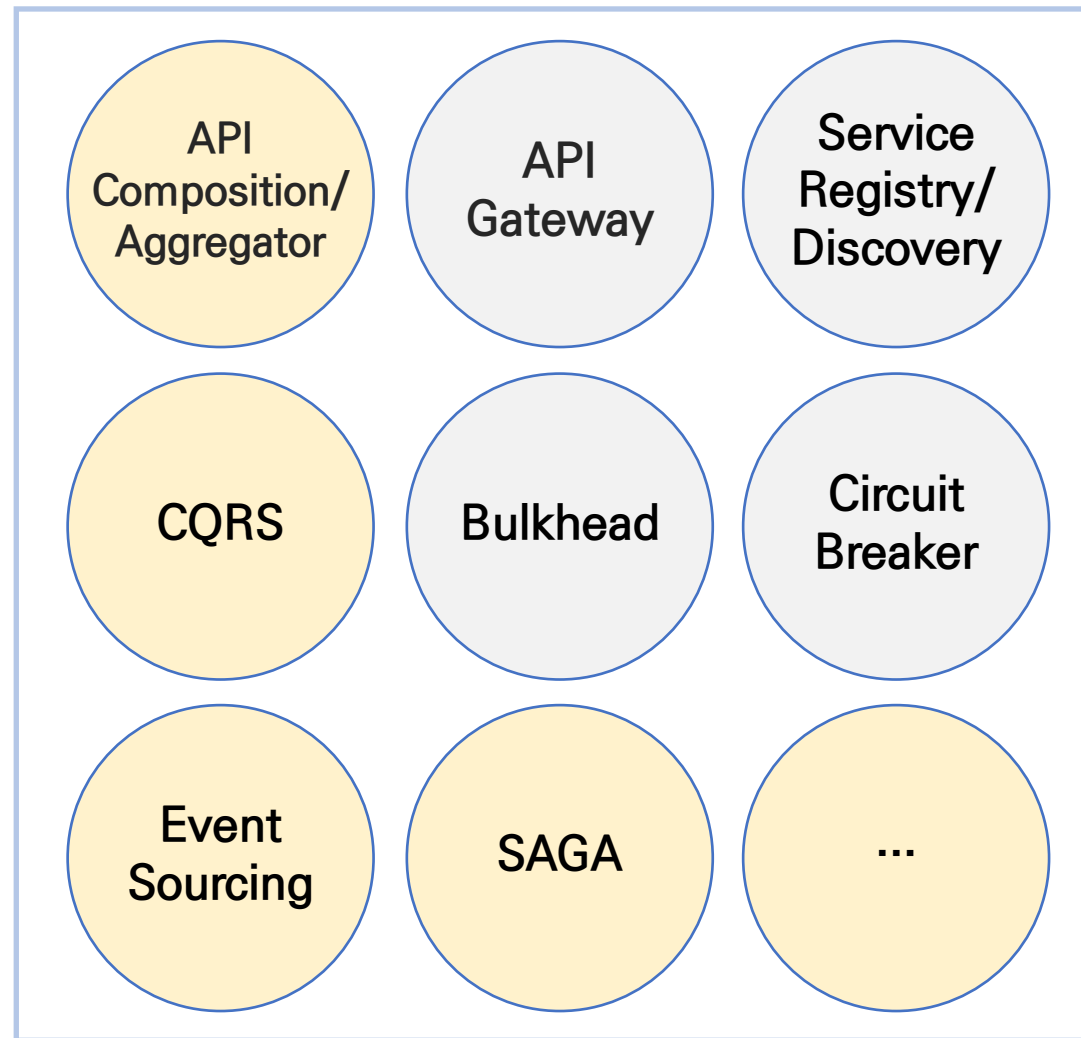
- 서비스 단위의 컴포넌트
- 비즈니스 기능 단위로 구성
- 제품 모델 (Not 프로젝트 모델)
- 단순한 통신 프로토콜
- 분산된 관리 체계
- 분산된 데이터 관리
- 인프라 자동화
- 실패를 고려한 설계
- 진화하는 설계

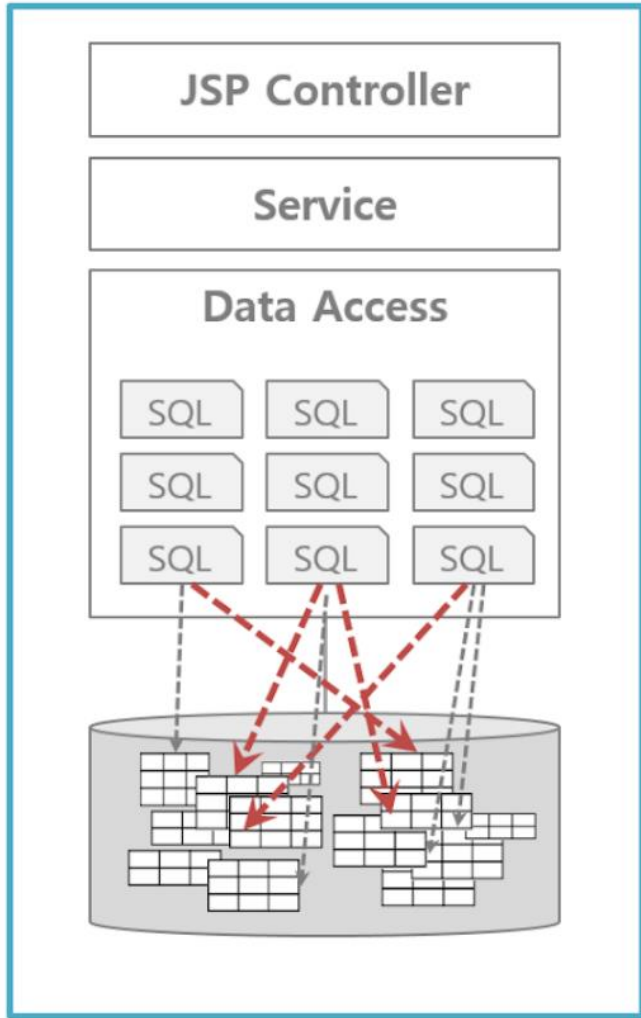
DB를 분리해서 개발한다고?

1. API로 속도가 나오겠어?

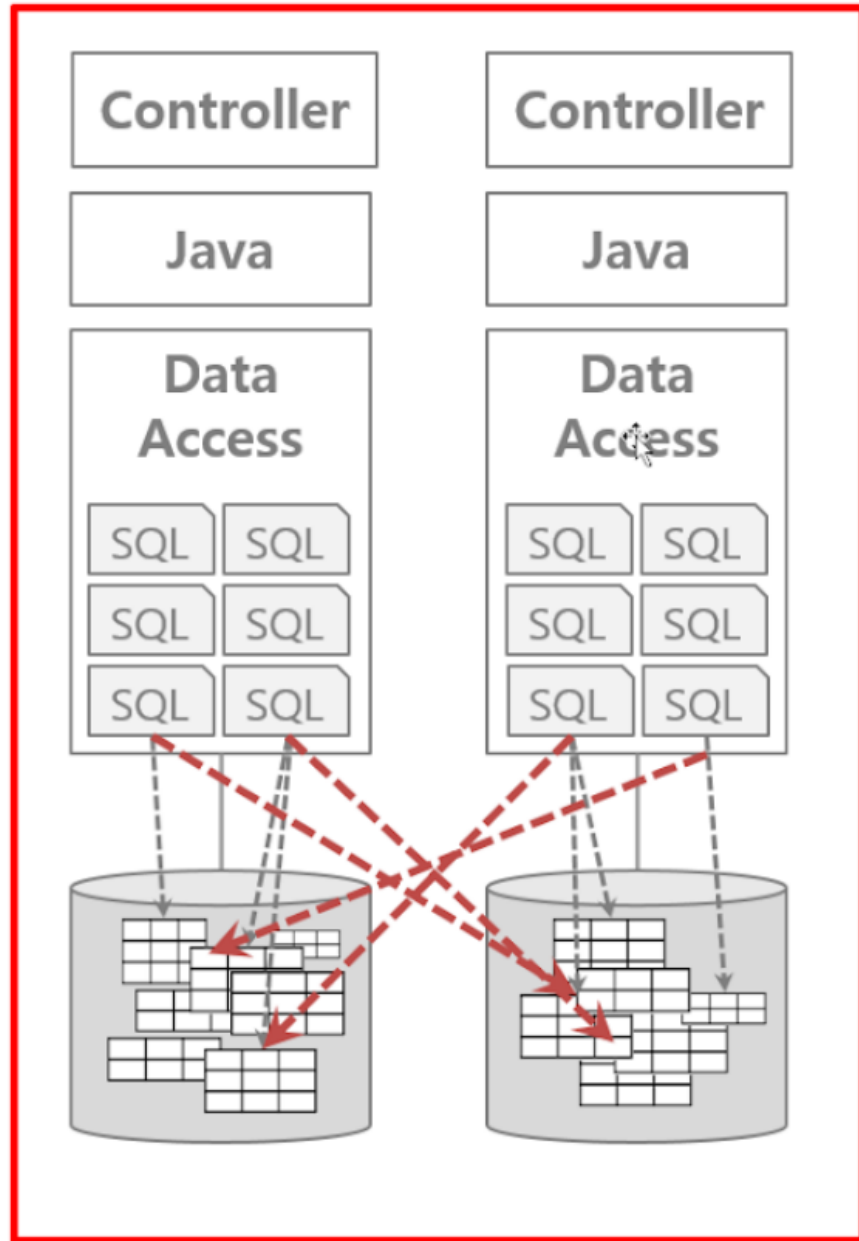
2. 트랜잭션 보장 없이
정합성이 맞겠어?

마이크로서비스 기술

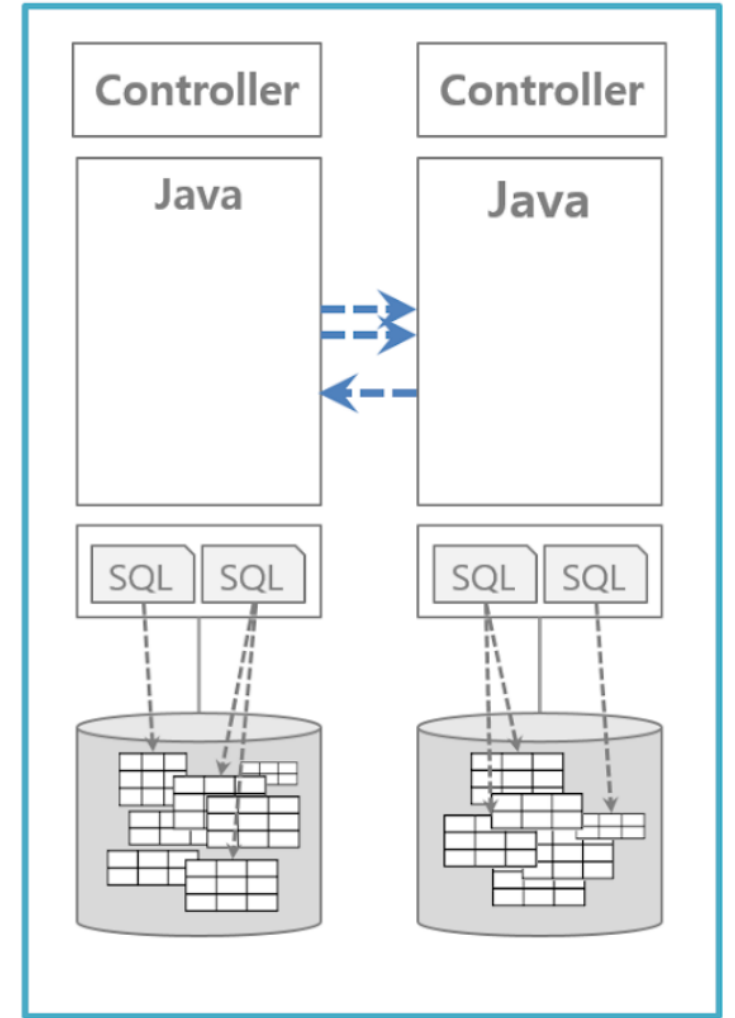




모노리식 아키텍처



껍데기만 MSA

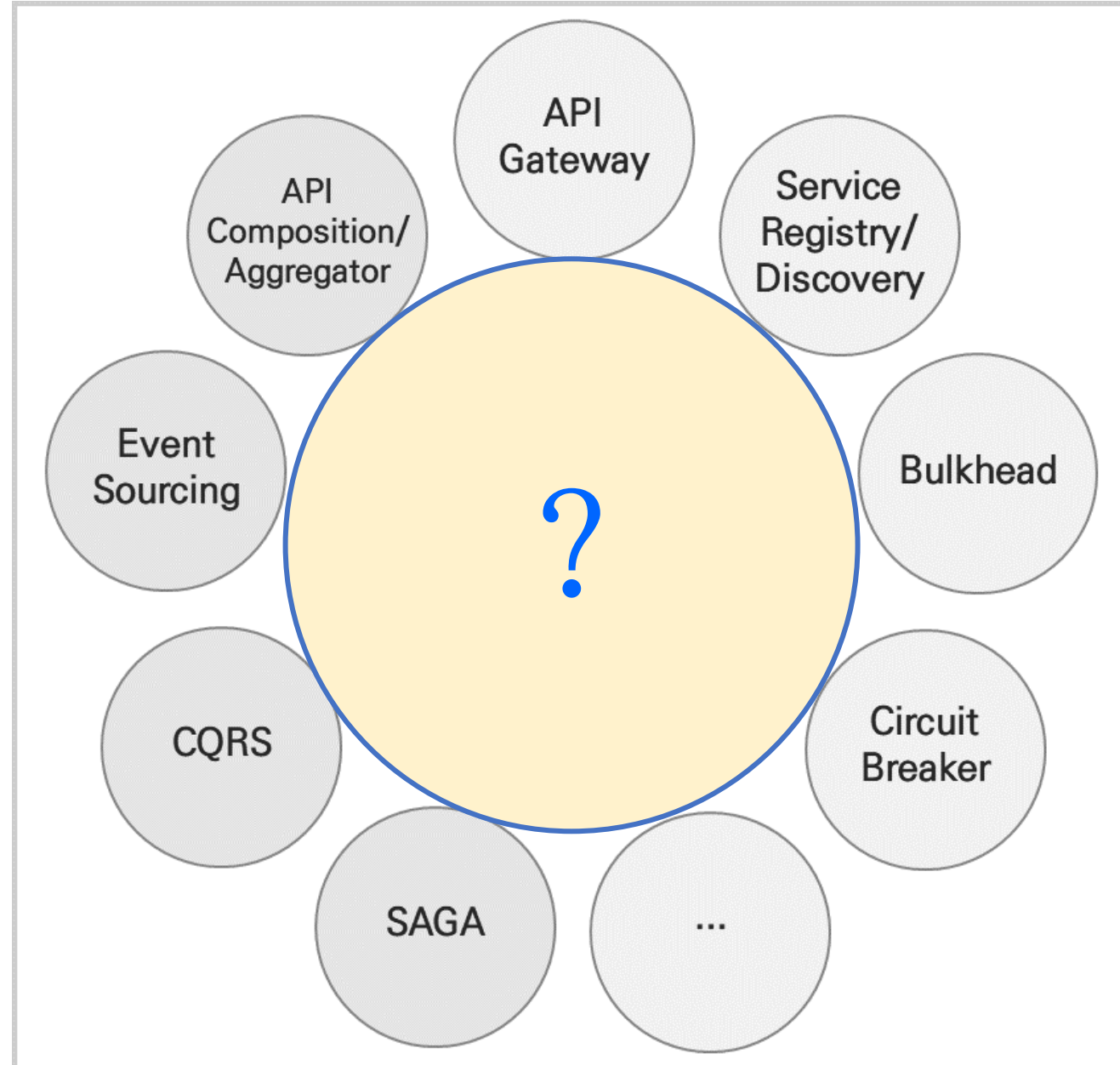


마이크로서비스 아키텍처

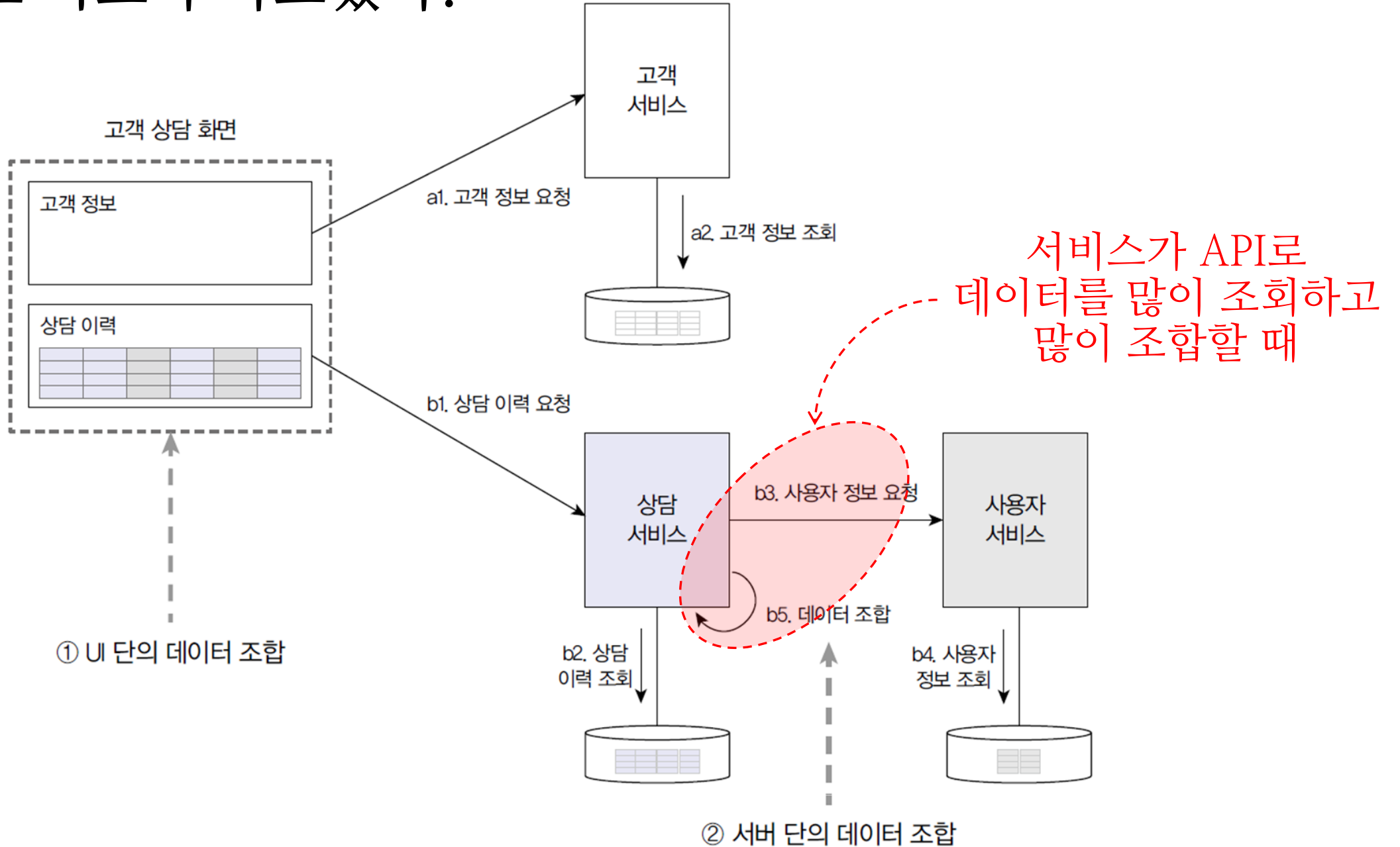
DB를 분리해서 개발한다고?

1. API로 속도가 나오겠어?

2. 트랜잭션 보장 없이
정합성이 맞겠어?



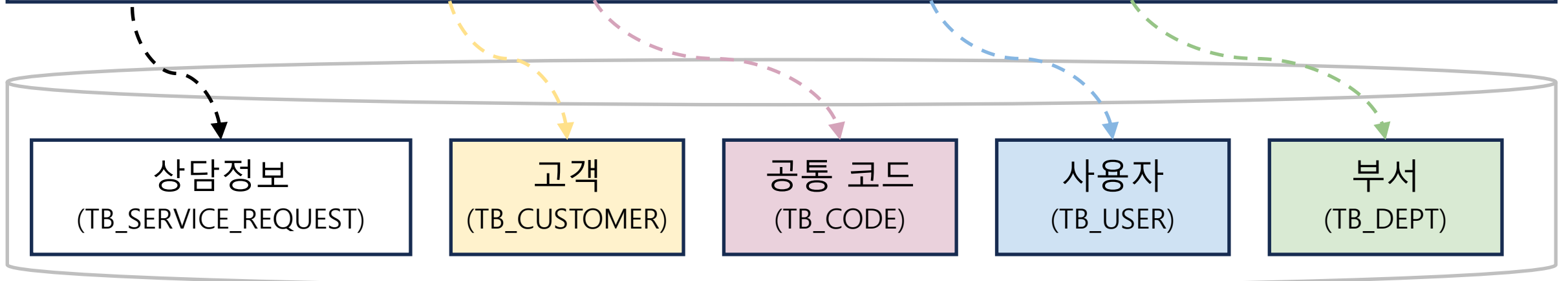
1. API로 속도가 나오겠어?



전체 상담 정보 화면

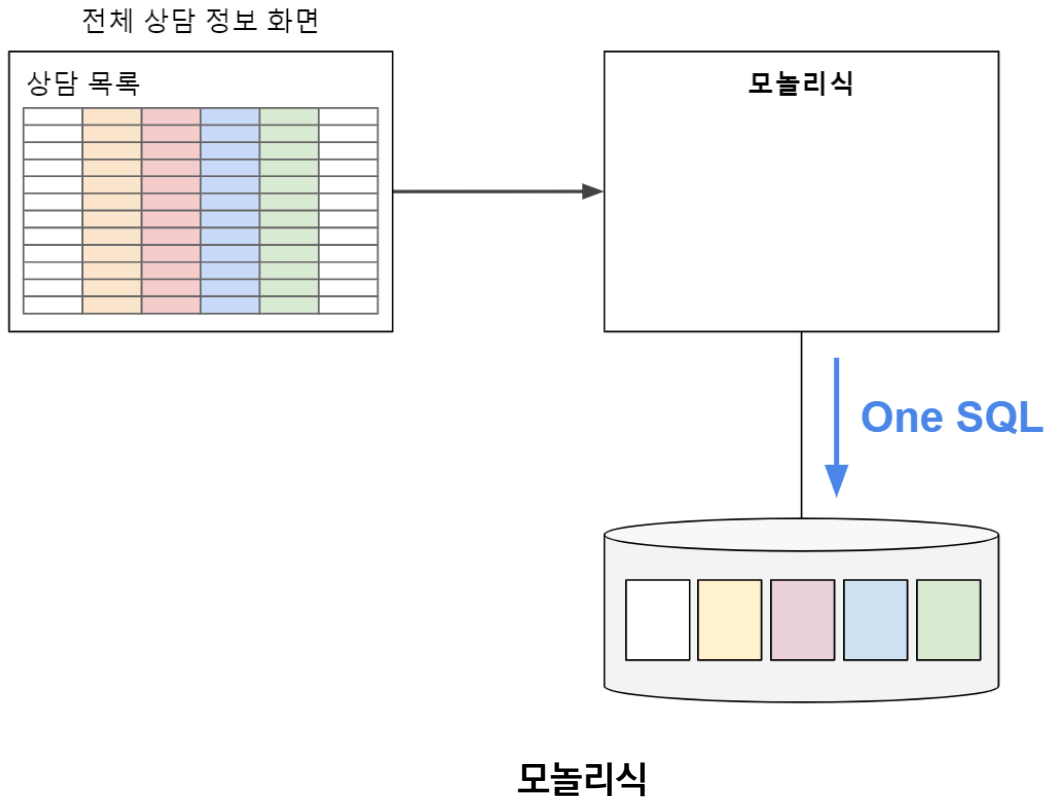
상담 목록

ID	상담제목	고객이름	유형	상세내용	상담원	VOC담당자	담당부서	생성일	변경일
0000000145158	...	임 지후	서비스요청	...	서 서연	오 지민	마케팅	2022-08-10	2022-08-10
0000000145159	...	강 서현	불만	...	송 윤서	한 서윤	마케팅	2022-08-10	2022-08-10
0000000145160	...	홍 건우	서비스요청	...	황 지우	송 서연	영업팀	2022-08-10	2022-08-10
0000000145161	...	송 서윤	불만	...	서 서연	송 서윤	영업팀	2022-08-10	2022-08-10
0000000145162	...	장 서윤	서비스요청	...	최 지우	한 서윤	마케팅	2022-08-10	2022-08-10



예시 - 모놀리식 시스템의 동작

One SQL

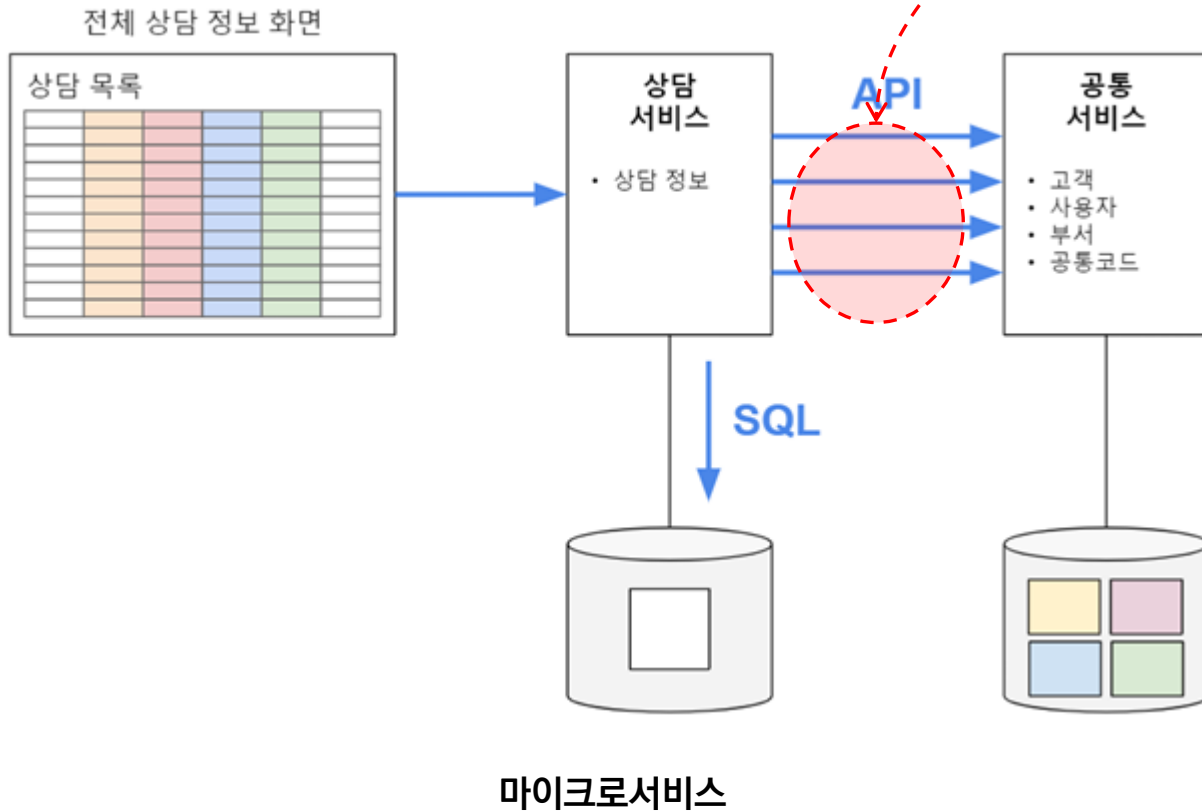


```
SELECT
  TSR.ID,
  TSR.TITLE,
  TSR.CUSTOMER_ID CUSTOMERID,
  TC.NAME CUSTOMERNAME,
  TSR."TYPE",
  (SELECT VALUE
   FROM TB_CODE TCO
   WHERE TCO.CODE = TSR."TYPE"
   AND TCO.CODE_TYPE='SR_TYPE') TYPENAME,
  TSR.DETAIL,
  TSR.STATUS,
  (SELECT VALUE
   FROM TB_CODE TCO
   WHERE TCO.CODE = TSR.STATUS
   AND TCO.CODE_TYPE='SR_STATUS') STATUSNAME,
  TSR.CALL_AGENT_ID CALLAGENTID,
  TUC."NAME" CALLAGENTNAME,
  TSR.VOC_ASSGNEE_ID VOCASSGNEEID,
  TUA."NAME" VOCASSGNEENAME,
  TSR.VOC_ASSGNEE_DEPT_ID VOCASSGNEEDEPTID,
  TD."NAME" VOCASSGNEEDEPTNAME,
  TSR.CREATED,
  TSR.UPDATED
FROM
  TB_SERVICE_REQUEST TSR
  LEFT JOIN TB_CUSTOMER TC ON TSR.CUSTOMER_ID = TC.ID
  LEFT JOIN TB_DEPT TD ON TSR.VOC_ASSGNEE_DEPT_ID = TD.ID
  LEFT JOIN TB_USER TUA ON TSR.VOC_ASSGNEE_ID = TUA.ID
  LEFT JOIN TB_USER TUC ON TSR.CALL_AGENT_ID = TUC.ID

ORDER BY TSR.UPDATED DESC
```

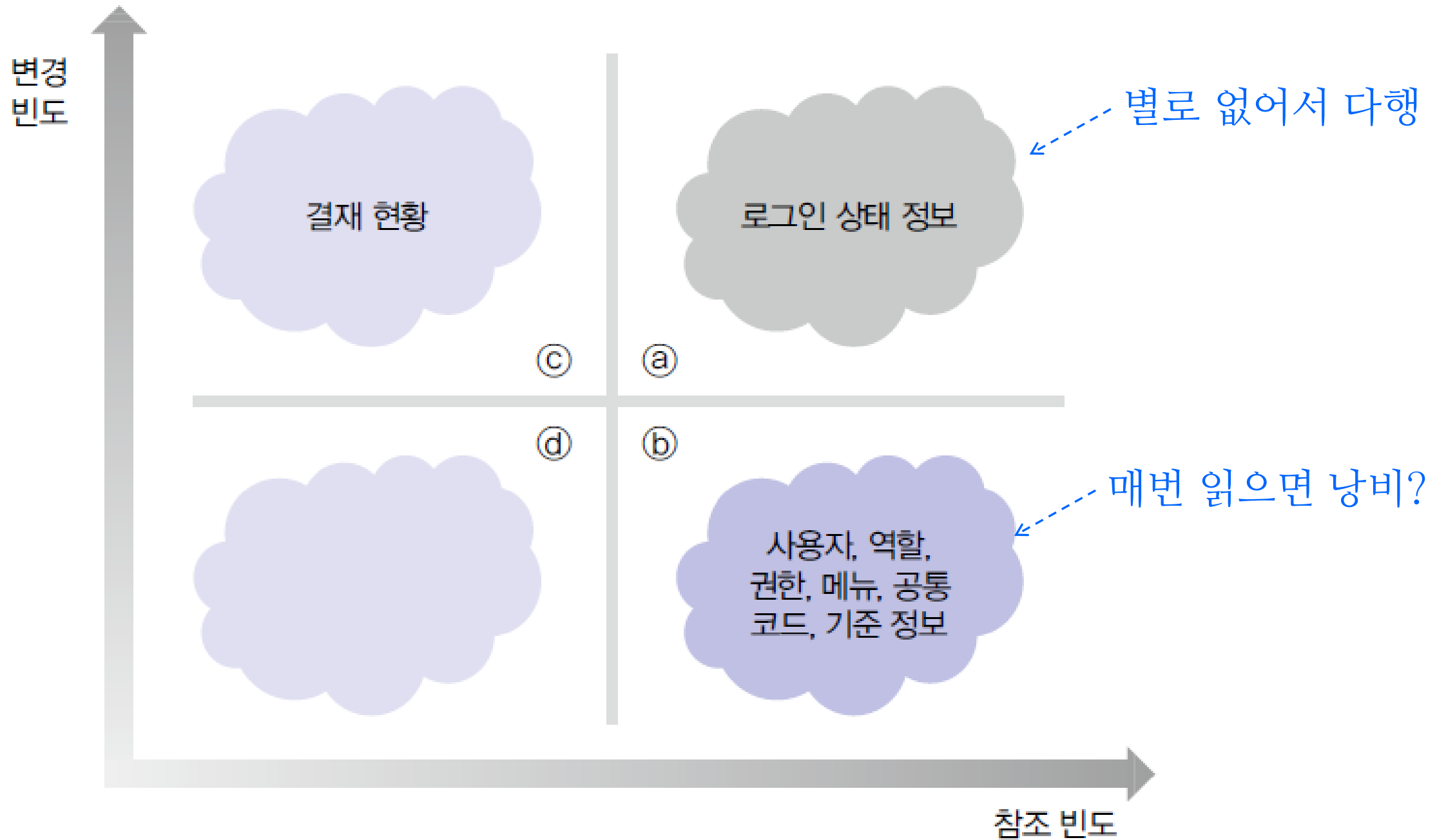
예시 - 마이크로서비스의 동작

조회하는 데이터가 많을 수록
많이 느려질 것이라고..



SQL

```
SELECT
  TSR.ID,
  TSR.TITLE,
  TSR.CUSTOMER_ID CUSTOMERID,
  TSR."TYPE",
  TSR.DETAIL,
  TSR.STATUS,
  TSR.CALL_AGENT_ID CALLAGENTID,
  TSR.VOC_ASSGNEE_ID VOCASSGNEEID,
  TSR.VOC_ASSGNEE_DEPT_ID VOCASSGNEEDEPTID,
  TSR.CREATED,
  TSR.UPDATED
FROM
  TB_SERVICE_REQUEST TSR
ORDER BY TSR.UPDATED DESC
LIMIT $1 OFFSET $2
```



① 데이터 복제

사용자 테이블

칼럼명	비고
ID	ID
CREATED	생성 일시
CREATED_BY	생성자 ID
UPDATED	변경 일시
UPDATED_BY	변경자 ID
LOGIN_ID	로그인 ID
PASSWORD	암호
FIRSTNAME	이름
LASTNAME	성
GENDER	성별
SSN	주민번호
NATIONALITY	국적
DOB	생일
EMAIL	이메일
PHONE_NO	전화번호
LANGUAGE	언어
TIMEZONE	시간대
EMP_NUMBER	사번
GRADE	직급
JOB_TITLE	직무
DEPARTMENT_CODE	부서 코드
POSTAL_CODE	우편번호
ADDRESS	주소
HIRE_DATE	입사일

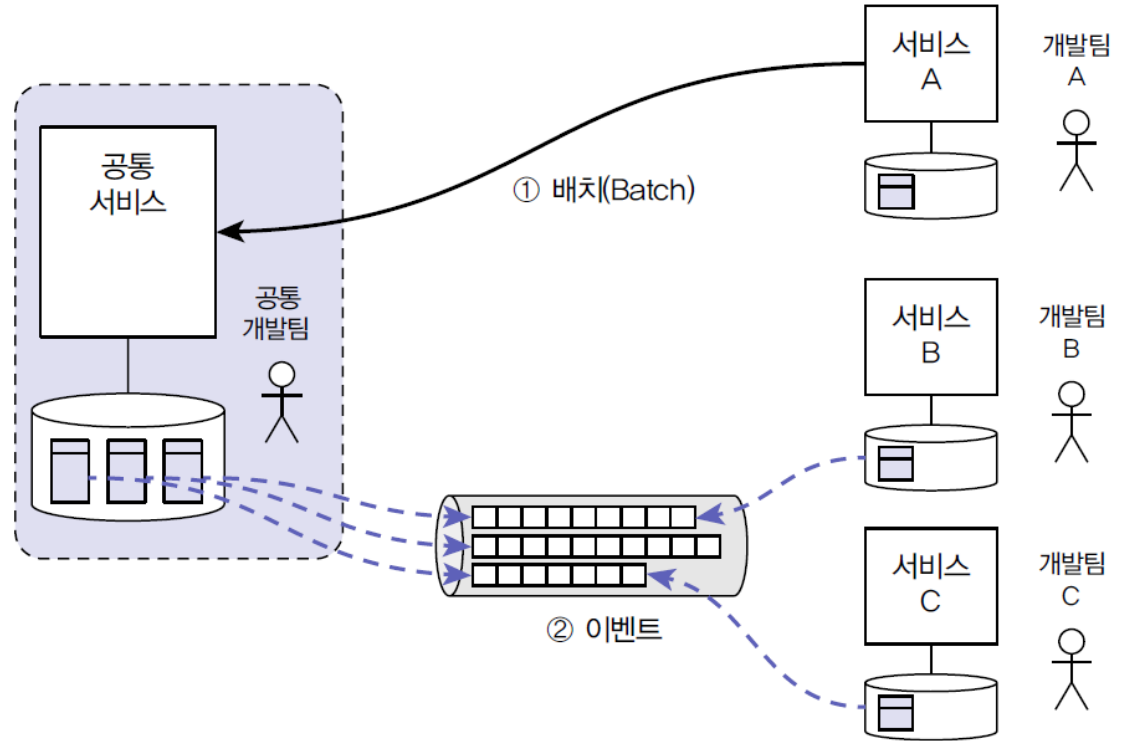
공통 서비스

사용자 테이블

칼럼명	비고
ID	ID
UPDATED	변경 일시
LOGIN_ID	로그인 ID
FIRSTNAME	이름
LASTNAME	성
EMAIL	이메일
PHONE_NO	전화번호

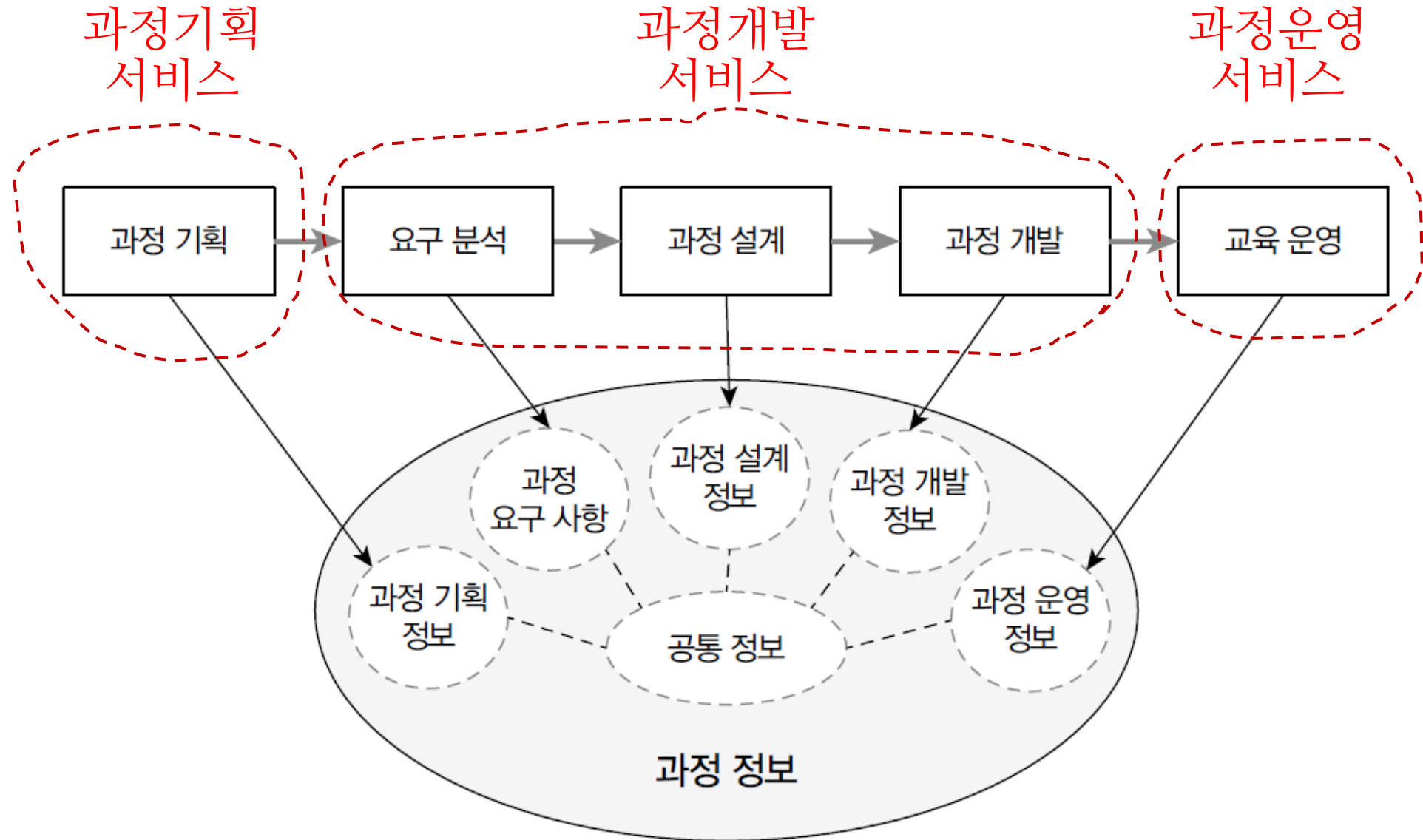
업무 서비스

필요한 속성만 복사!



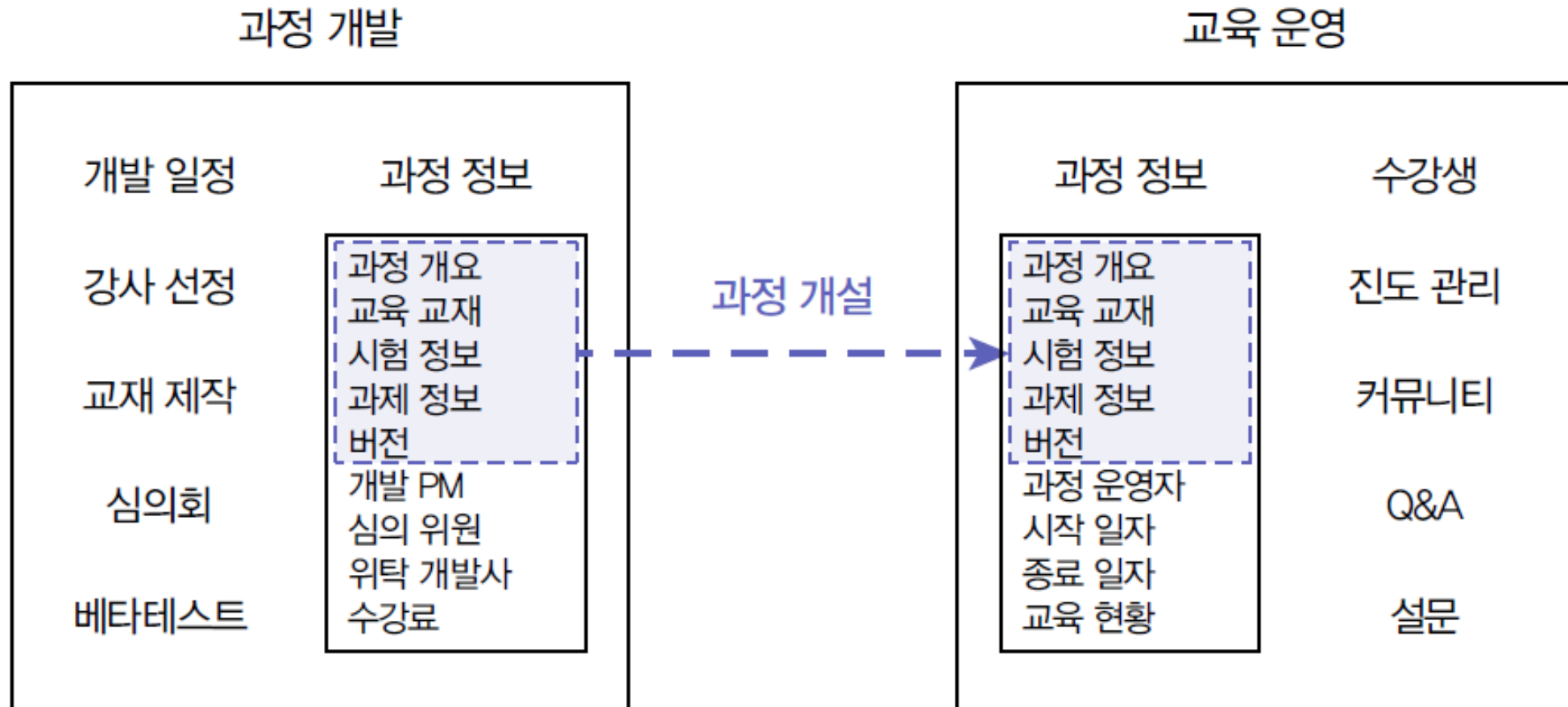
필요한 것만 끌여가는 구조
(NO 중앙에서 일괄 동기화)

② 모델링 변경



② 모델링 변경 (cont.)

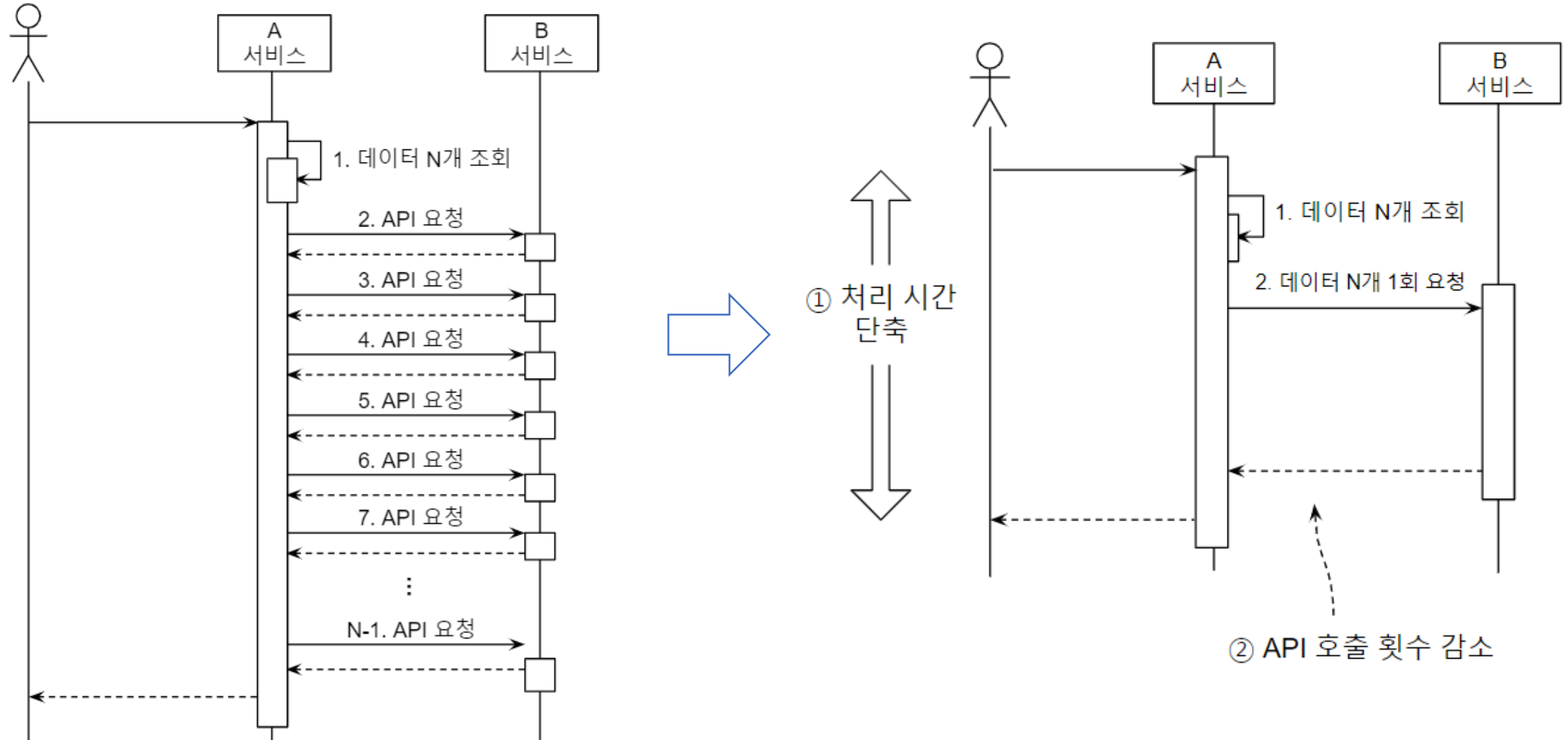
공통 속성은 각 서비스에 복제
특화 속성은 오너 서비스에 배치



③ 일괄 조회

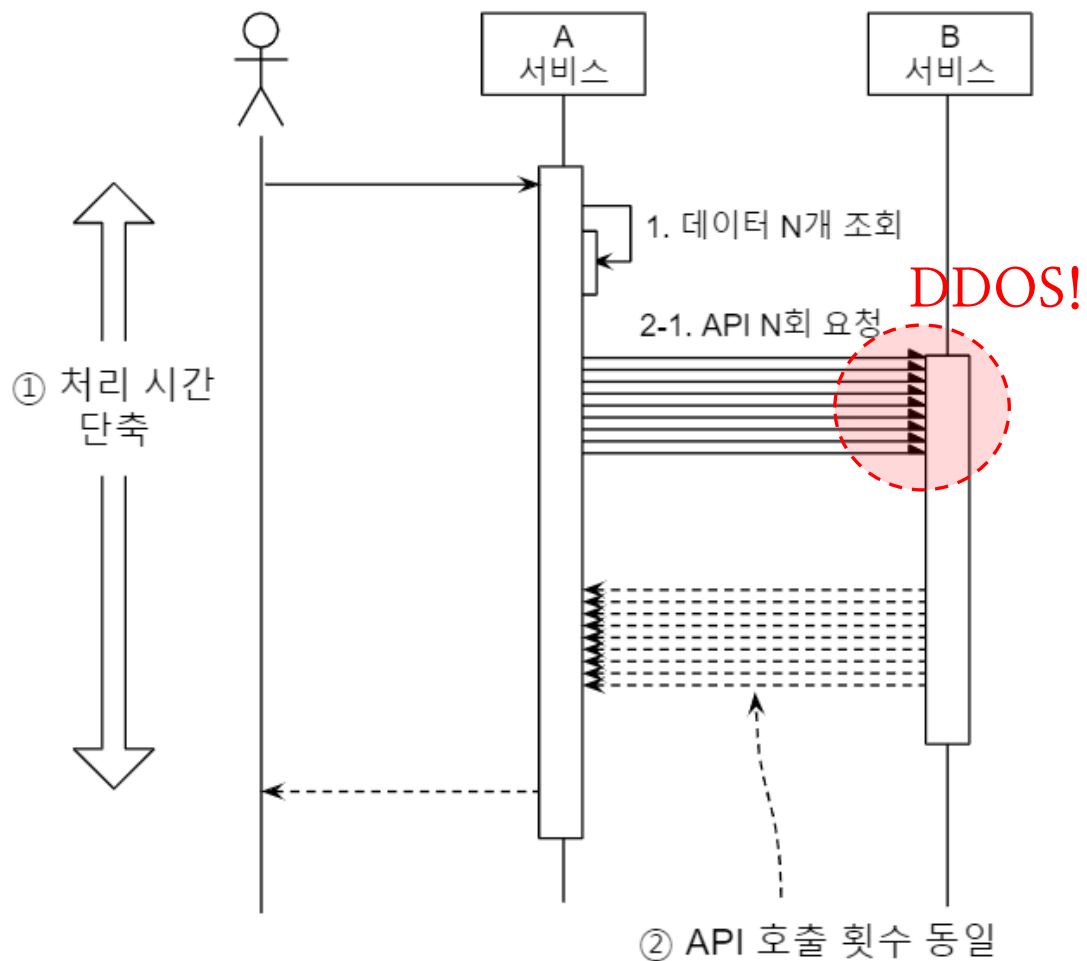
REST API의 N+1 Problem

모아서 한 번에 조회
== SELECT IN

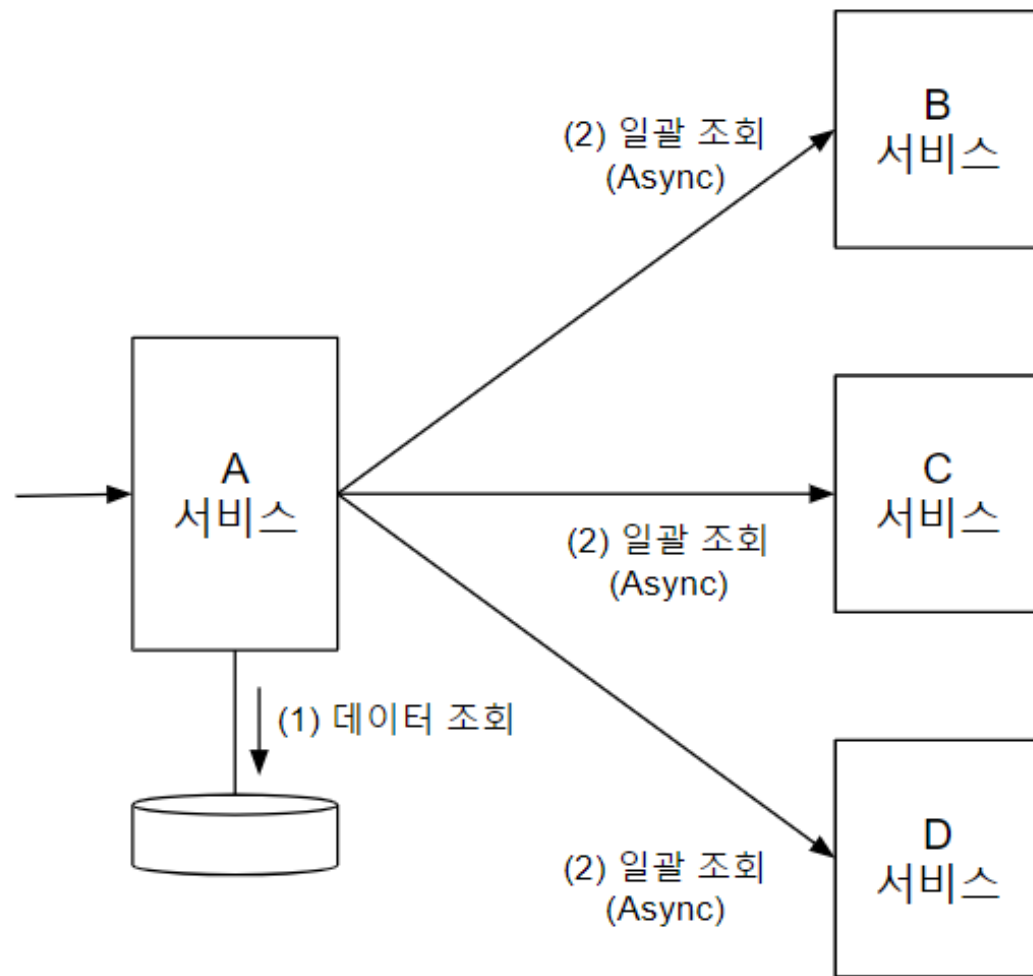


병렬 조회 ?

순간적으로 큰 부하가 생길 수 있음

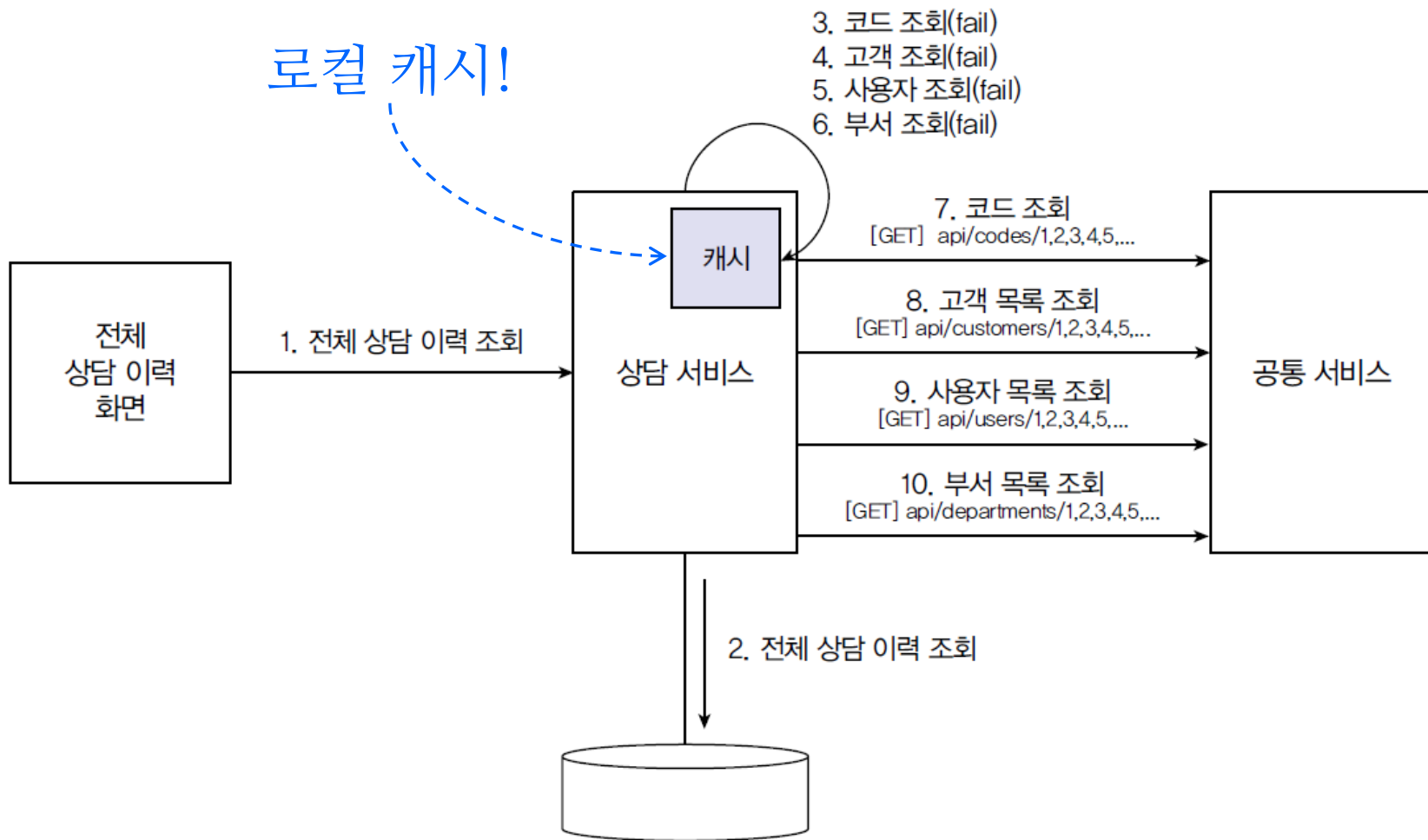


꼭 필요한 경우에만,
일괄 조회를 병렬로 실행

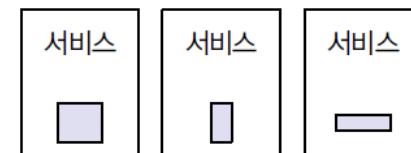


④ 로컬 캐시

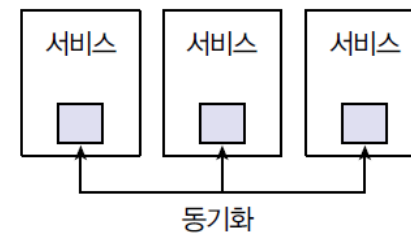
Redis는 느려요!



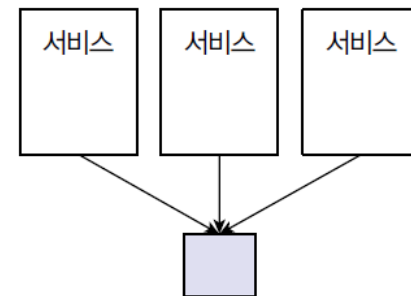
로컬 캐시



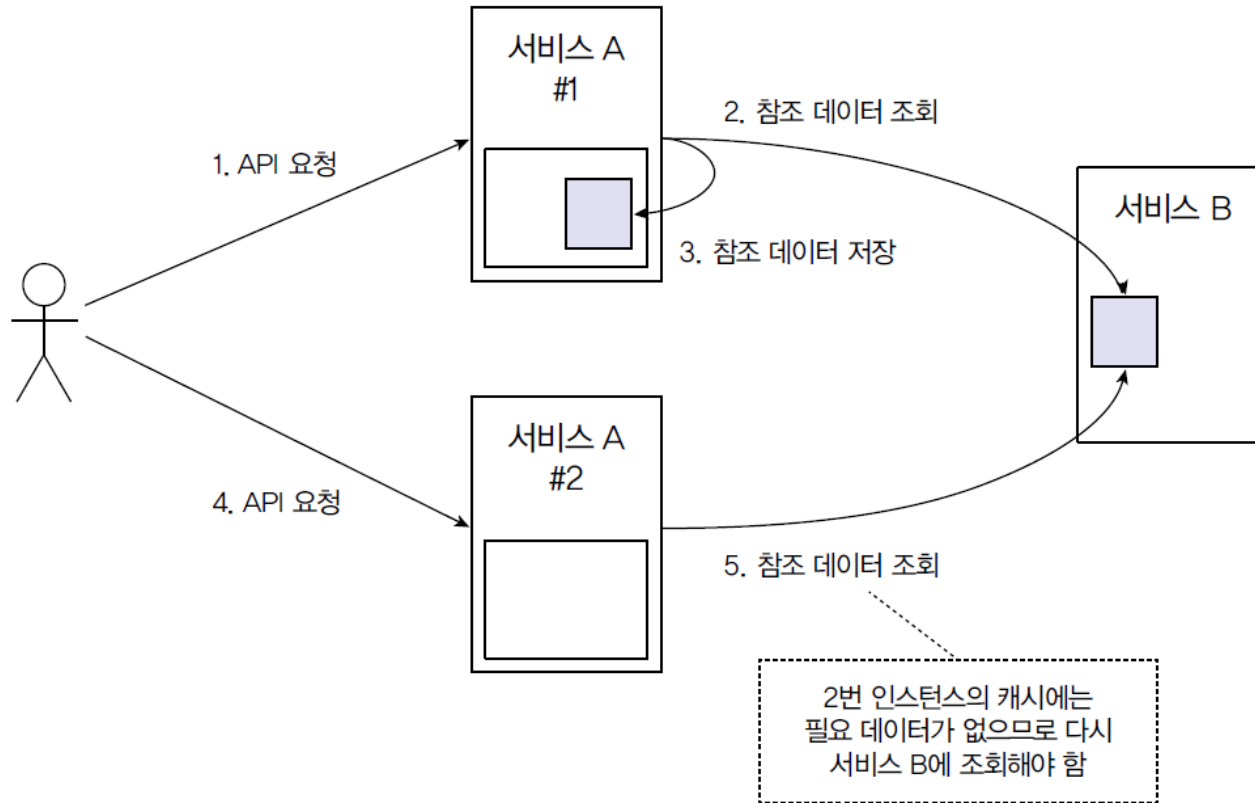
복제 캐시



분산 캐시



로컬 캐시는 동기화하지 않습니다.



로컬 캐시의 업데이트

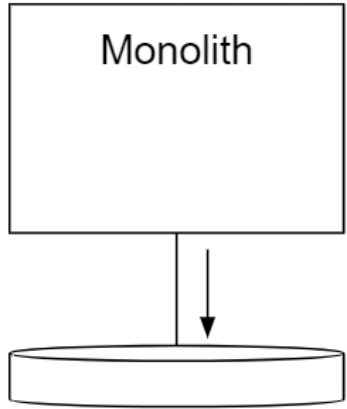
시연

상담 목록

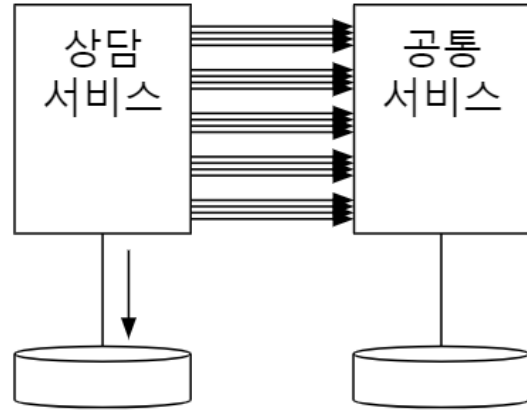
ID	상담제목	고객이름	유형	상세내용	상담원	VOC담당자	담당부서	생성일	변경일
0000000145158	...	임 지후	서비스요청	...	서 서연	오 지민	마케팅	2022-08-10	2022-08-10
0000000145159	...	강 서현	불만	...	송 윤서	한 서윤	마케팅	2022-08-10	2022-08-10
0000000145160	...	홍 건우	서비스요청	...	황 지우	송 서연	영업팀	2022-08-10	2022-08-10
0000000145161	...	송 서윤	불만	...	서 서연	송 서윤	영업팀	2022-08-10	2022-08-10
0000000145162	...	장 서윤	서비스요청	...	최 지우	한 서윤	마케팅	2022-08-10	2022-08-10

성능 비교 대상

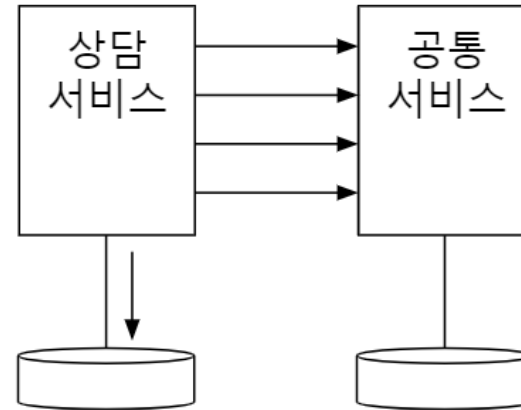
1. SQL JOIN



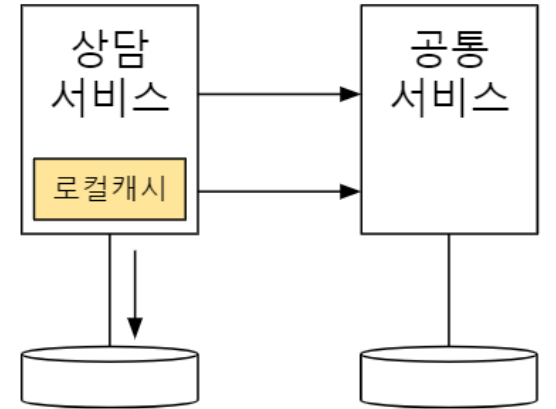
2. N+1



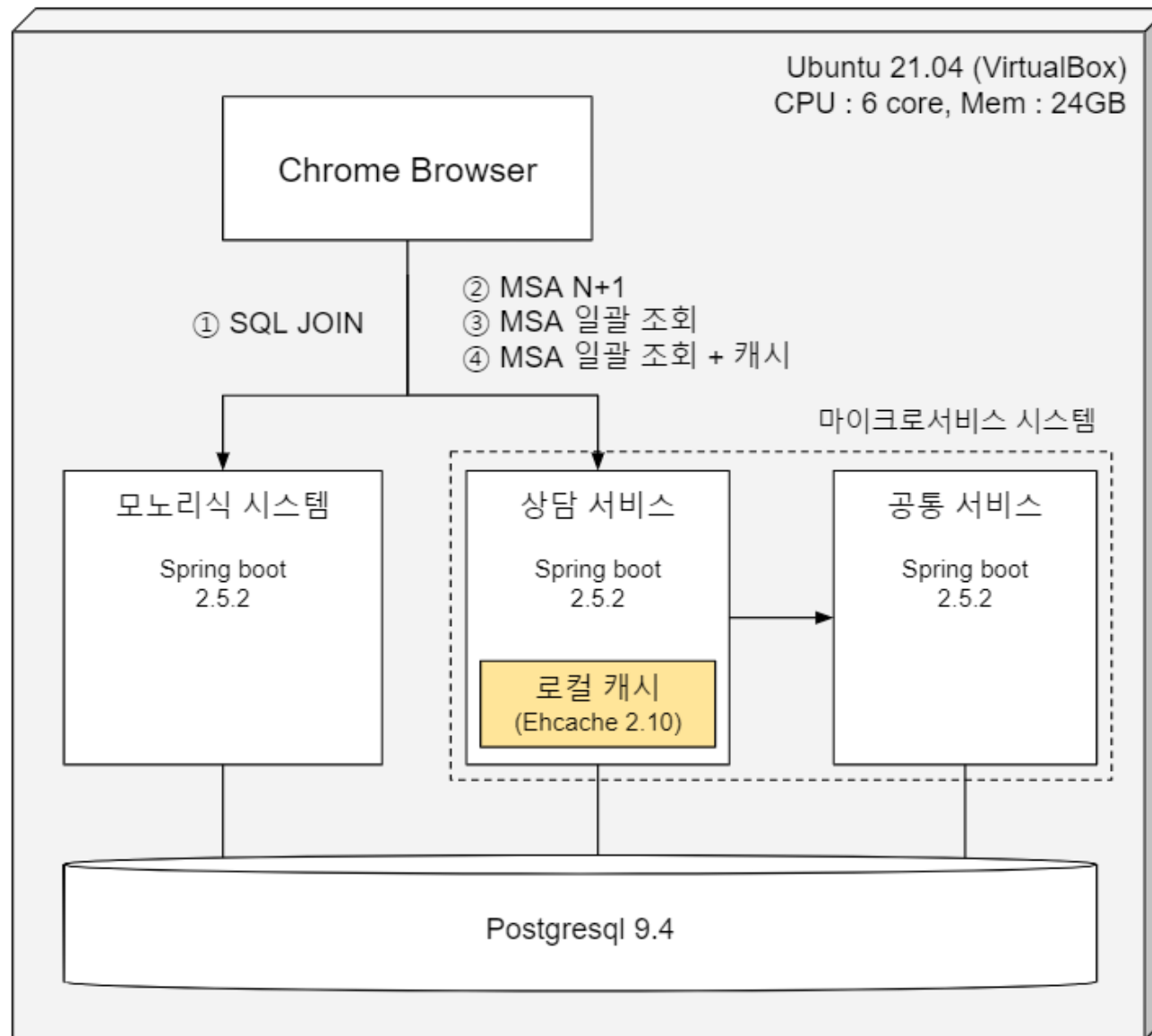
3. 일괄 조회



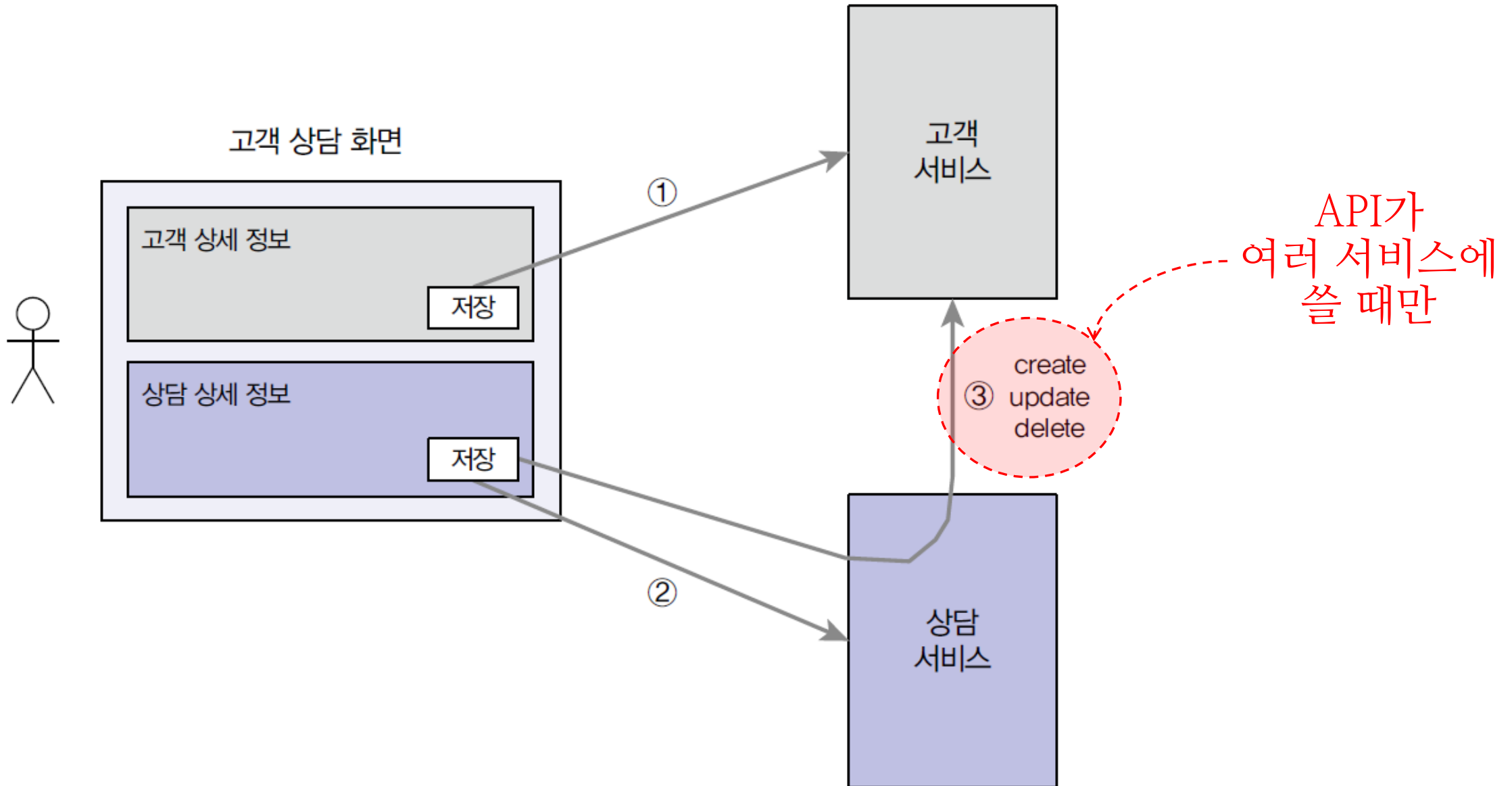
4. 일괄 조회 + 캐시



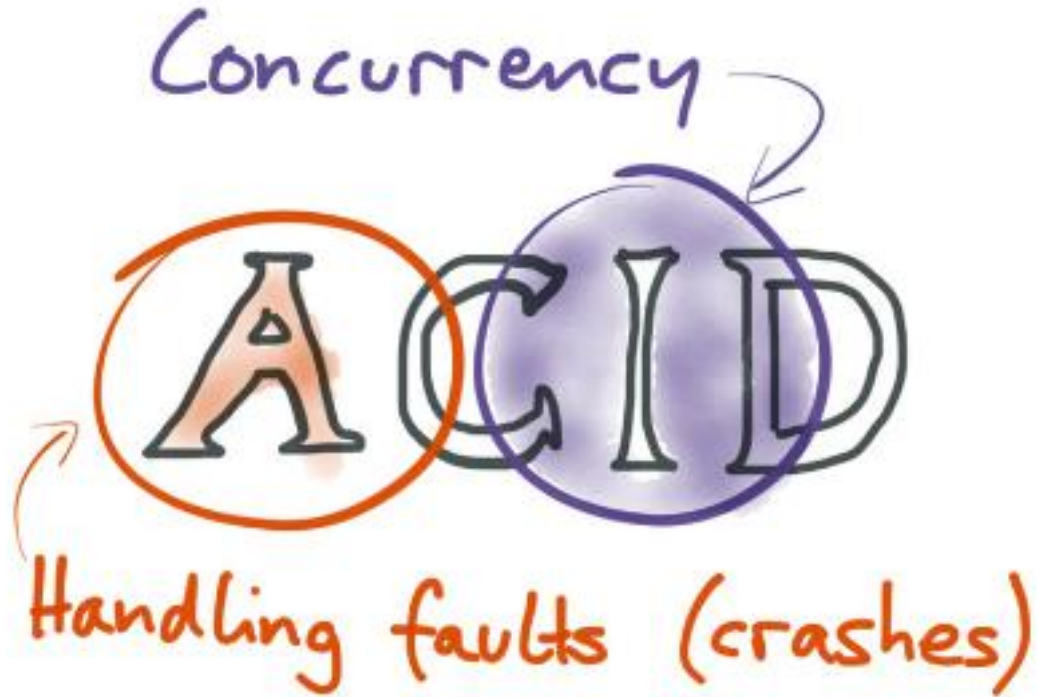
환경



2. 트랜잭션 보장 없이 정합성이 맞겠어?



ACID 트랜잭션



Atomicity (원자성) : DB Rollback

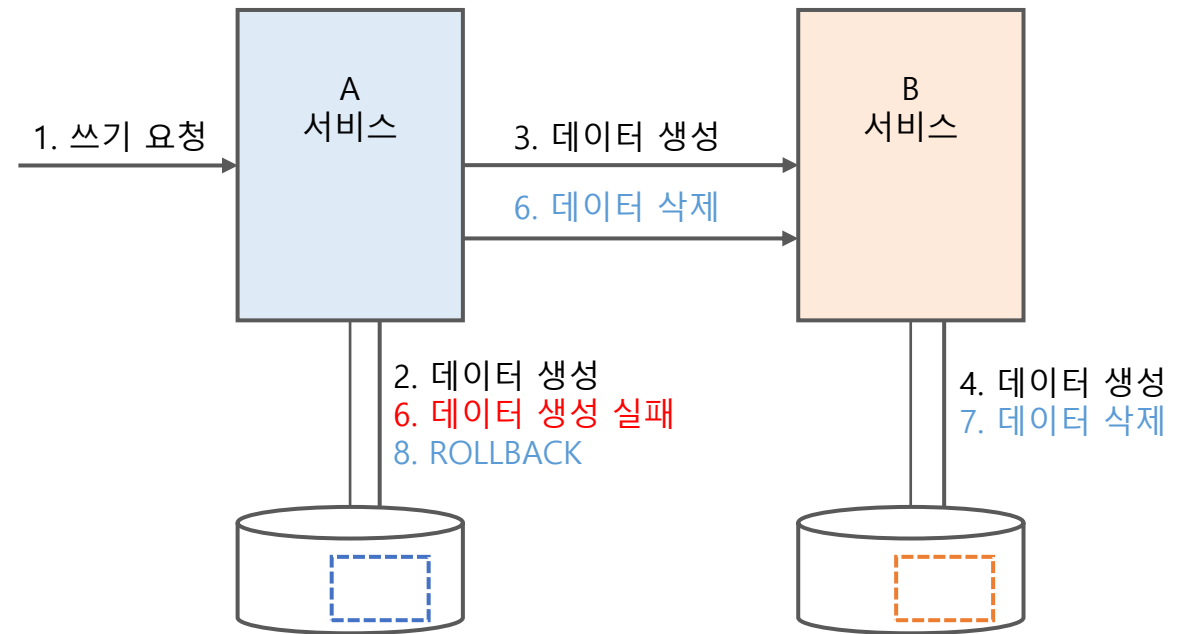
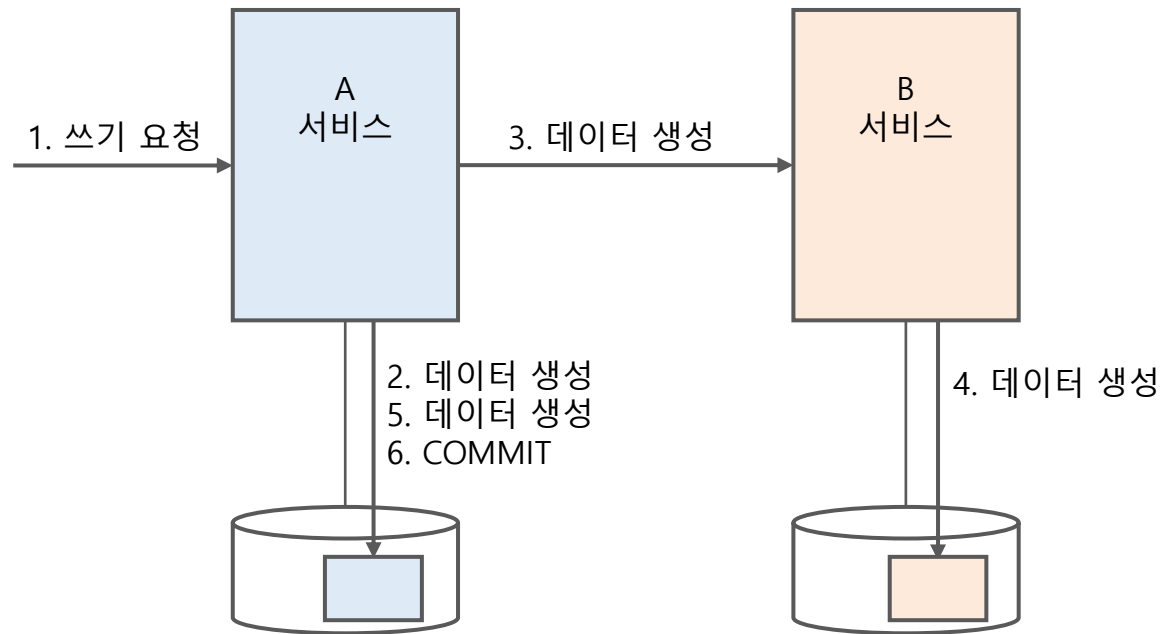
Consistency (일관성)

Isolation (독립성) : Read Committed

Durability (지속성)

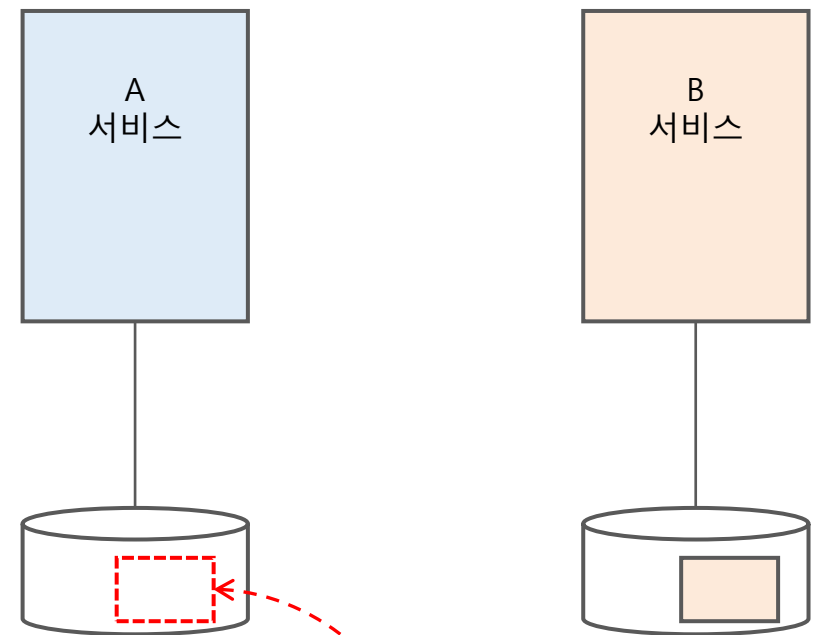
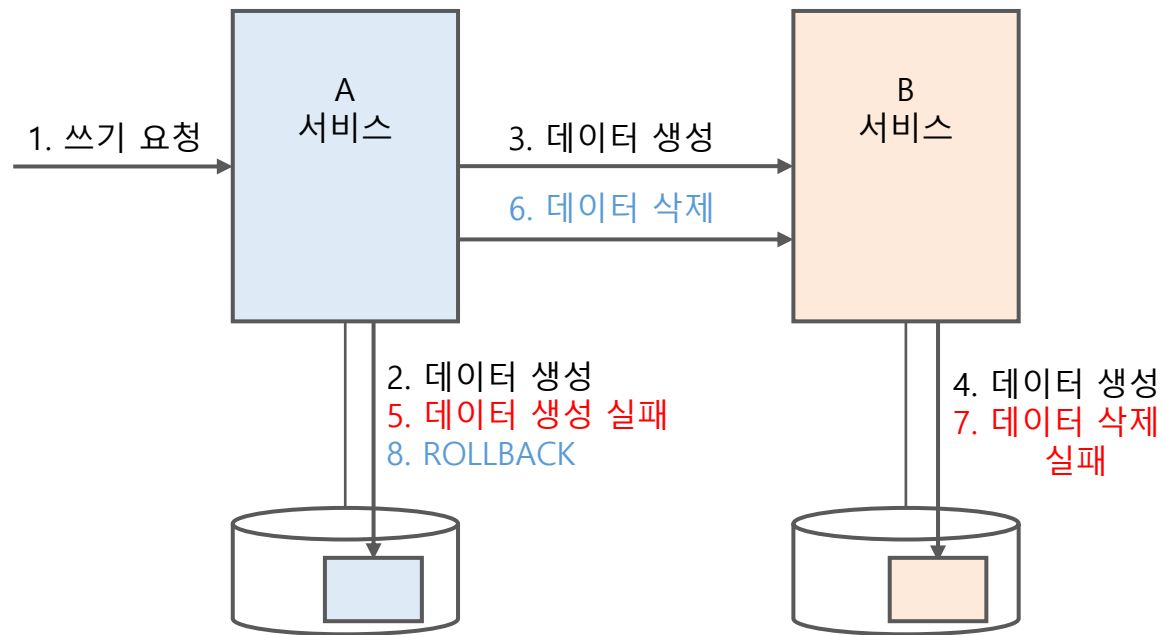
구현 이슈 - 원자성 보완

쓰기에 실패하면
B 서비스에 커밋된 데이터를 API로 삭제하고(6)
로컬에 생성한 데이터는 롤백합니다.



구현 이슈 - 원자성 보완 (cont.)

B 서비스의 데이터를 삭제하다 실패하면?



데이터가 짝이 안 맞게 남게 됩니다.

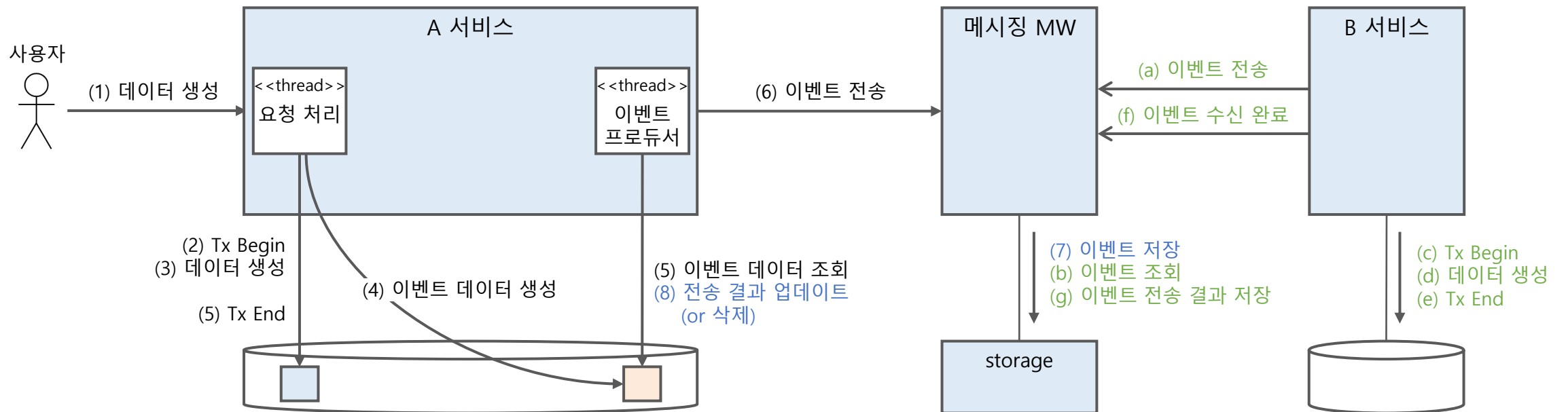
자동으로 재시도?

회복할 시간이 필요합니다.



이벤트의 재시도

이벤트는 무조건 전달됩니다.
실패하더라도 결국엔 안전하게 전달됩니다.
간혹 여러 번 전달될 수도 있습니다.



RabbitMQ, Kafka는 세부 동작 방식은 다르지만
둘 다 메시지를 디스크에 저장하는 기능을 제공함

실패했는데 성공했다고?

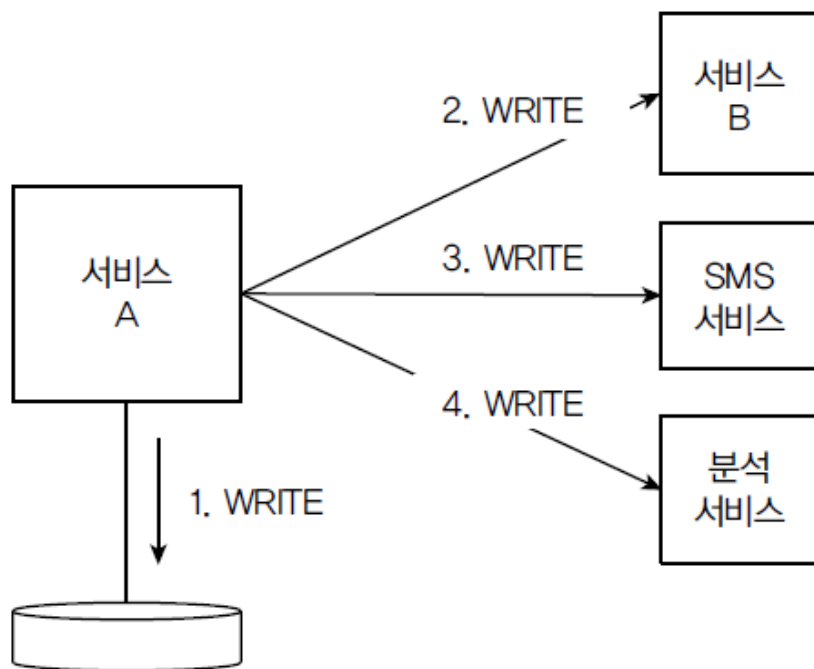


실패했다고 다시 하면
두 번 하는 것일 수도...

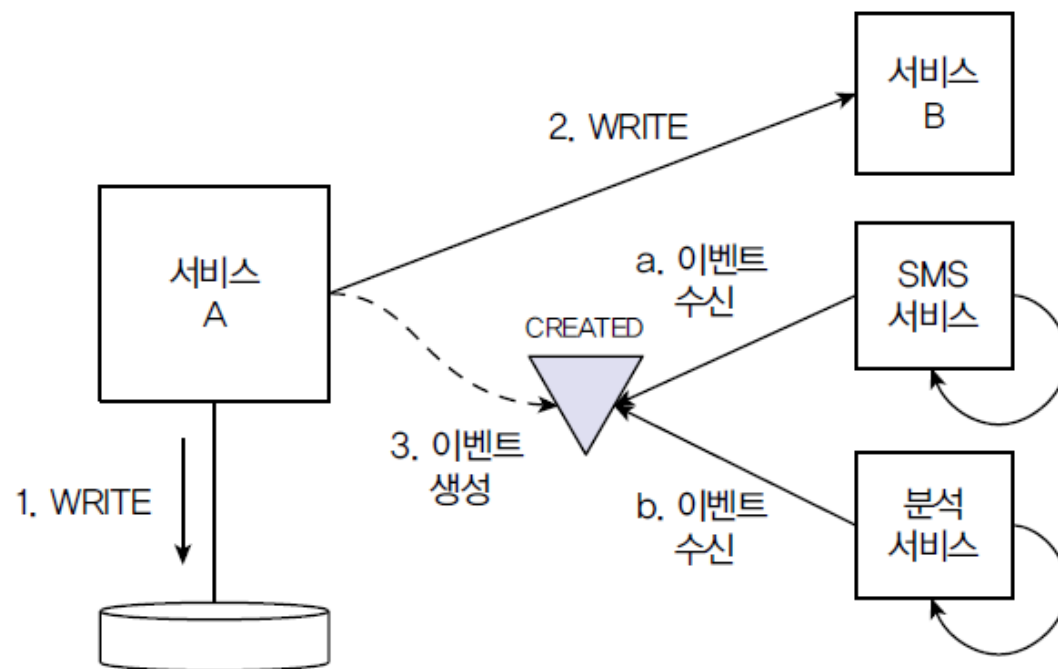
두 번 실행해도
동일한 결과가 나와야 함

① 긴 TX 나누기

- ① 실패해도 전체를 취소가 없다면 이벤트로 분리
- ② 취소할 수 없는 쓰기는 이벤트로 분리



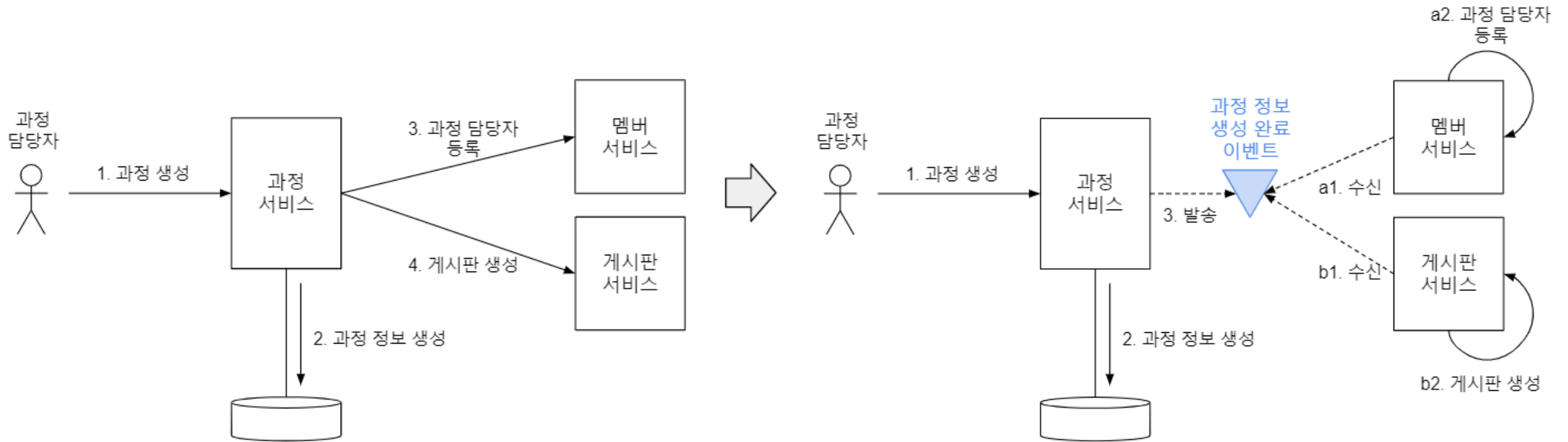
보상 트랜잭션 대상 : 2,3번



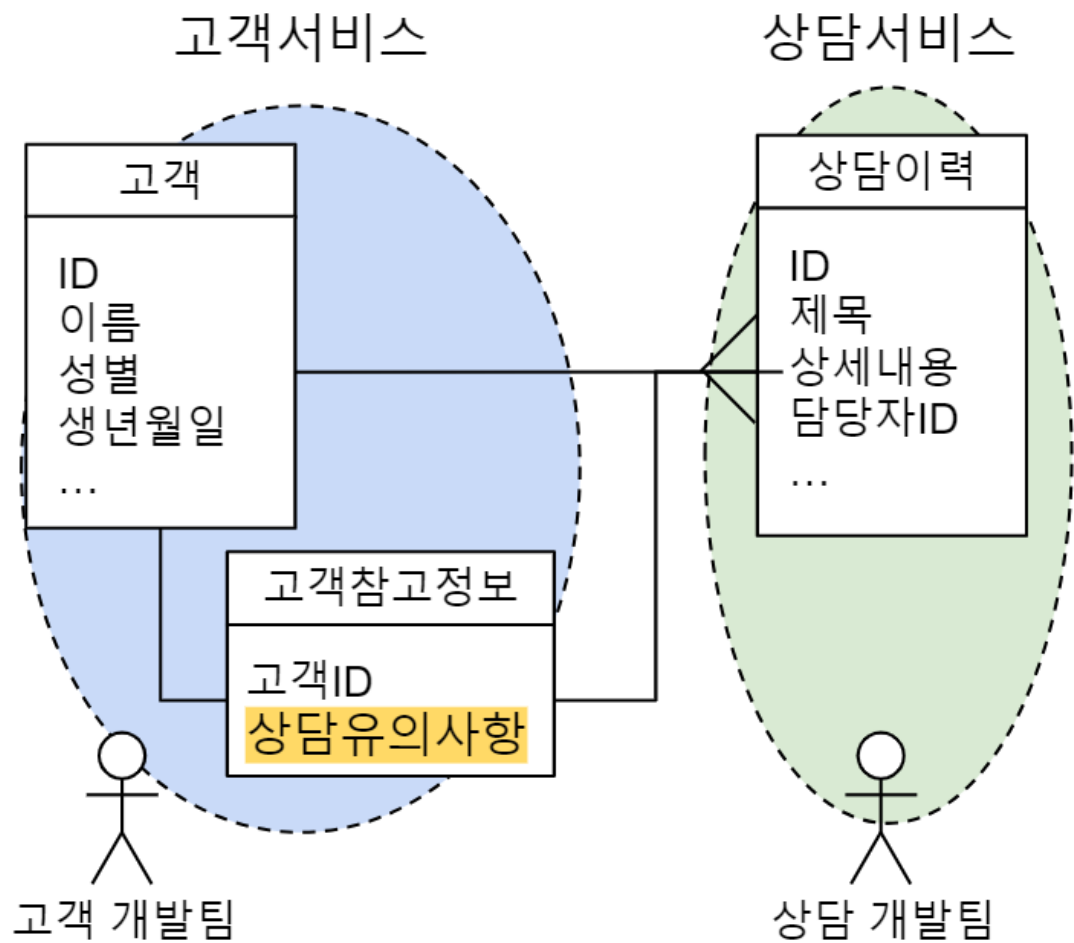
보상 트랜잭션 대상 : 2번 or 없음

② 역할 분리 → 각자 알아서

과정 서비스는 이벤트만 발송하고 끝
나머지는 다른 서비스가 각자 하는 게 좋은 설계일 수도..

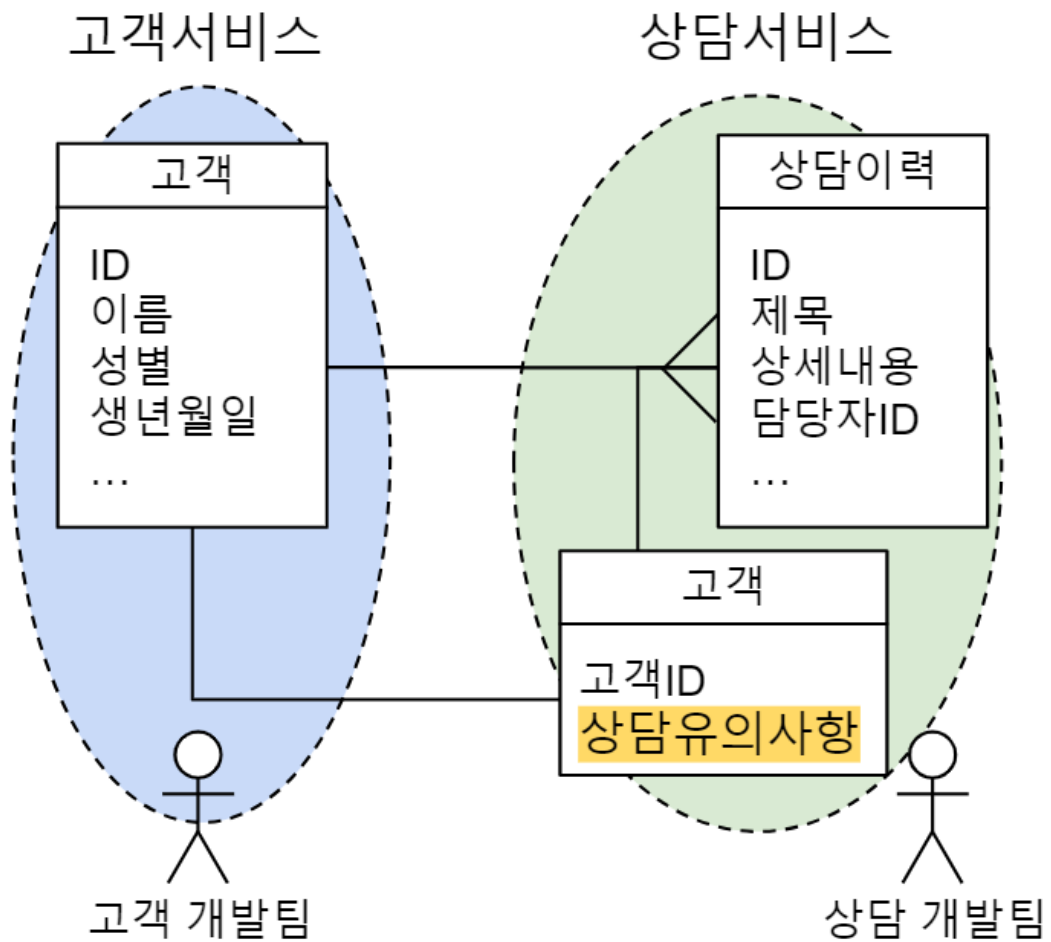


③ 모델링 변경



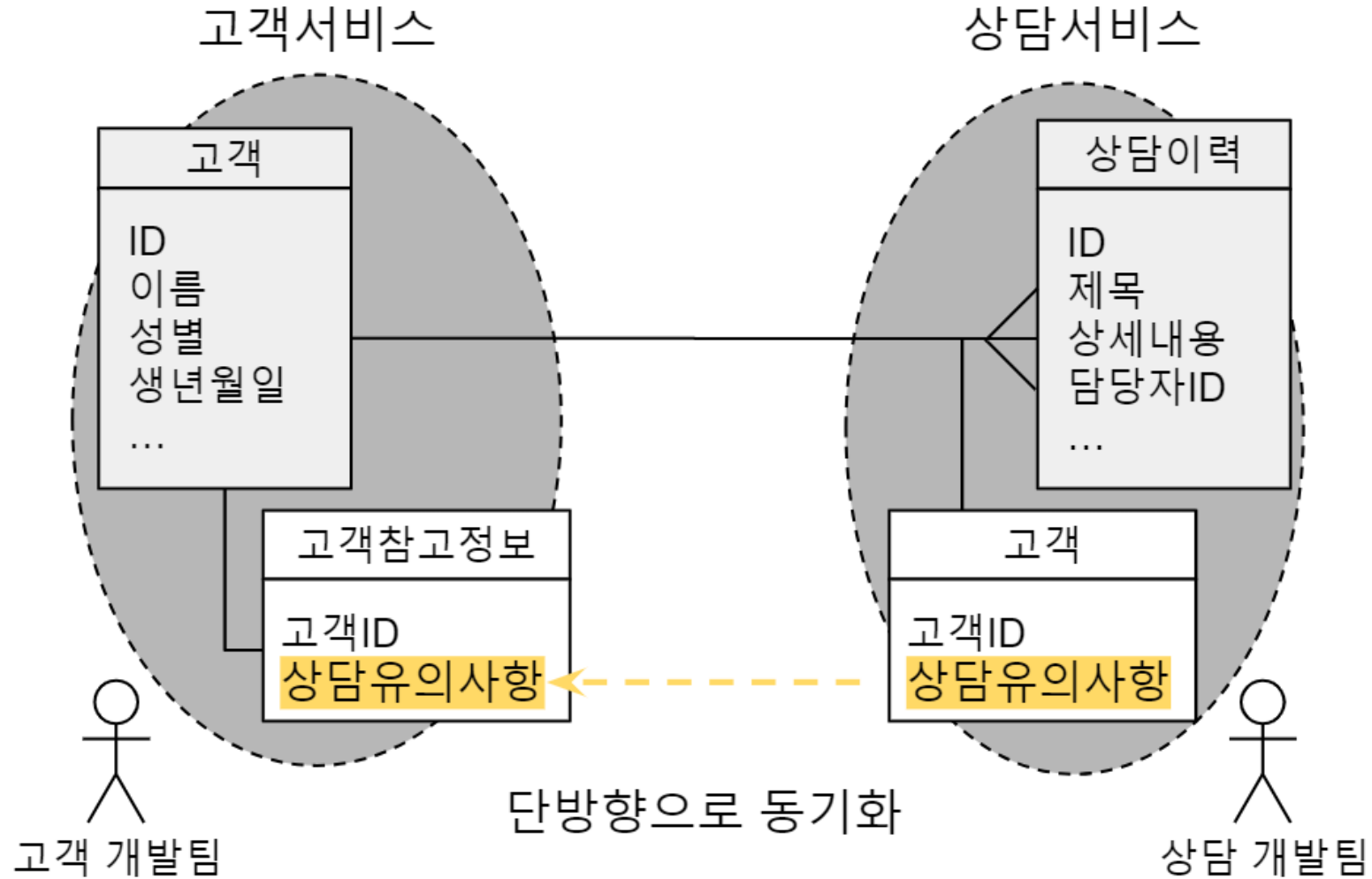
고객서비스에 배치

데이터는 오너십을 가진 서비스에 배치



상담서비스에 배치

③ 모델링 변경 (cont.)



④ 서비스 경계 변경

너무 구현하기 힘들면,
서비스를 합치거나 경계를 변경해도 괜찮습니다.

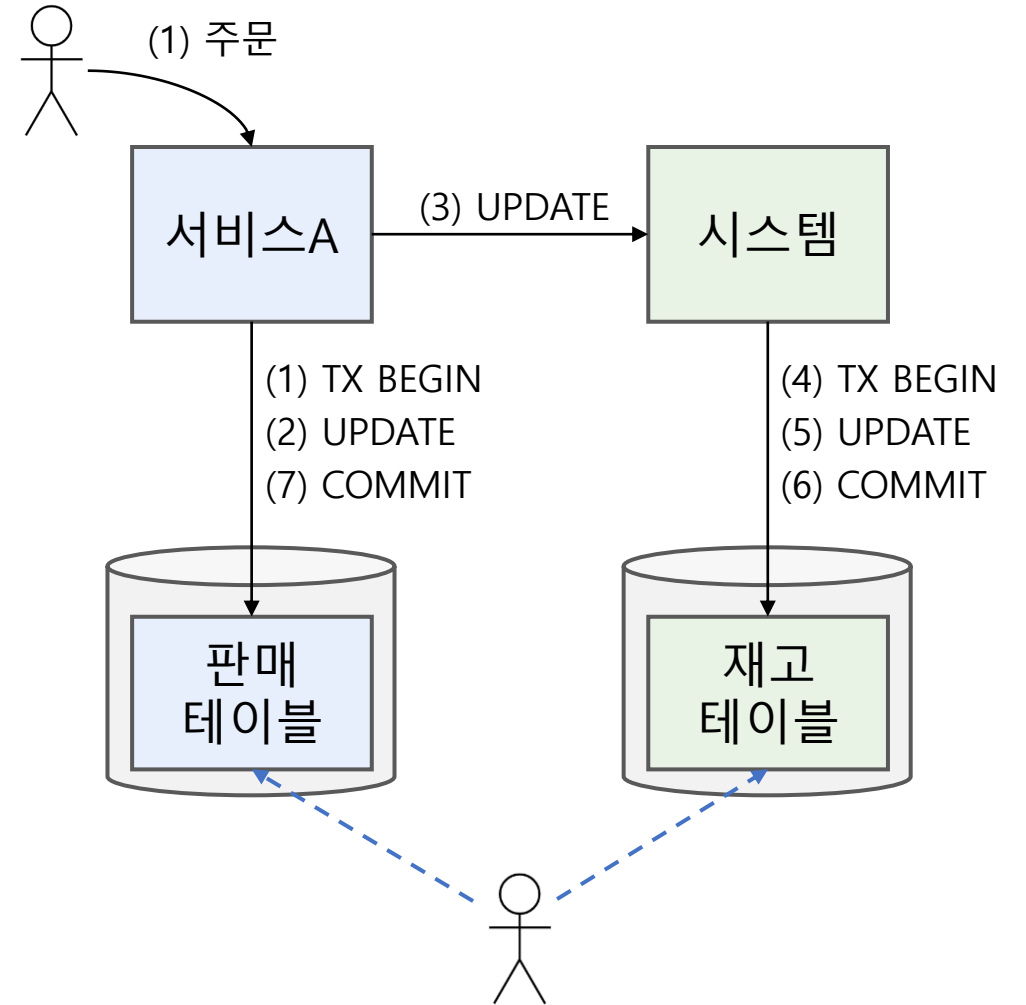
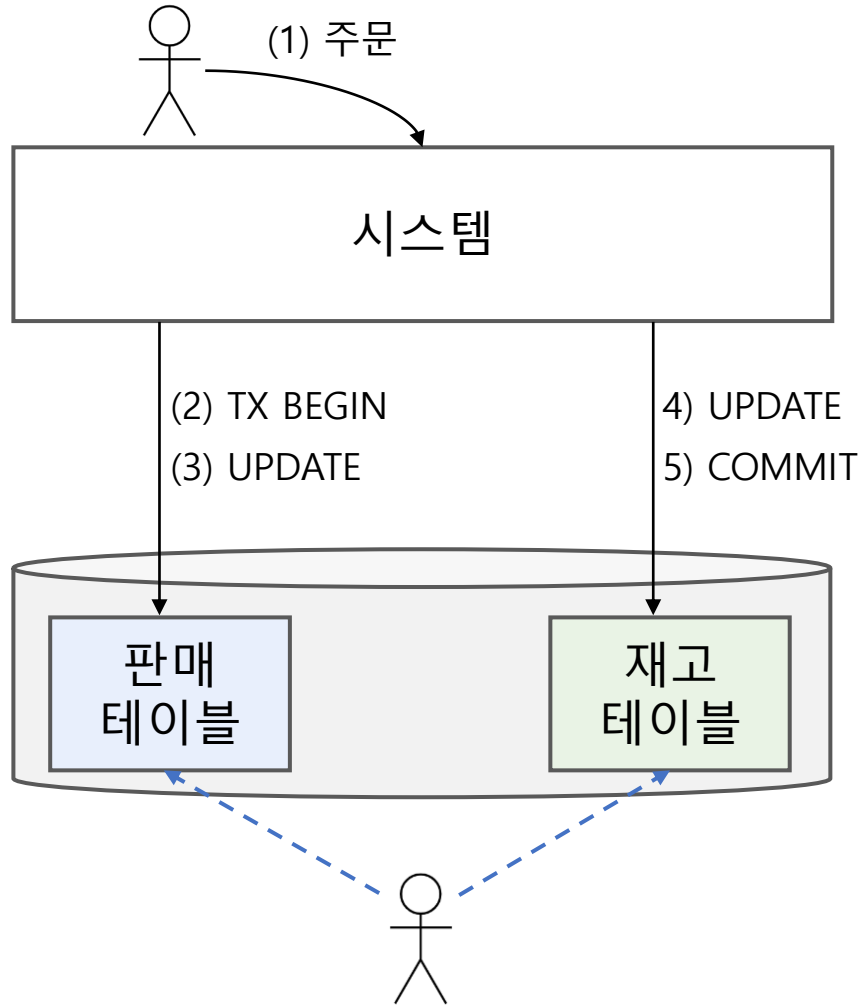
구현 이슈 - 독립성 보완

서비스 간의 트랜잭션의 Isolation Level은 Read Uncommitted

	Dirty Read	Non-repeatable Read	Phantom Read	Write Skew	Database Default Level
Read Uncommitted	발생 가능	발생 가능	발생 가능	발생 가능	
Read Committed	발생 안함	발생 가능	발생 가능	발생 가능	Oracle, MSSql, PostgreSQL
Repeatable Read	발생 안함	발생 안함	발생 가능	발생 가능	MySQL, Mariadb
Serializable	발생 안함	발생 안함	발생 안함	발생 안함	

구현 이슈 - 독립성 보완

① 서비스간의 데이터가 순간적으로 일치하지 않을 수 있음

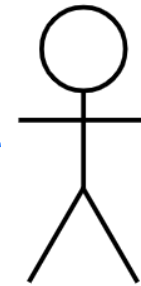


구현 이슈 - 독립성 보완 (cont.)

② 데이터베이스의 동기화 메커니즘을 사용할 수 없음

	Phantom Read	Write Skew
Read Uncommitted	발생 가능	발생 가능
Read Committed	발생 가능	발생 가능
Repeatable Read	발생 가능	발생 가능
Serializable	발생 안함	발생 안함

원래 DB가 해주지 않고
개발자가 해결해야 하는데



① ~~SELECT FOR UPDATE~~

② Application Lock

만 사용합니다

패턴

- 꼭 필요한 속성만 복사
- 일괄 조회
- 로컬 캐시
- 전체 취소 안 하면 이벤트로 쓰기
- 멱등성
- 분산 모델
- 데이터는 오너십 가진 서비스에

안티 패턴

- 테이블 스키마 복사
- 병렬 호출
- API 재시도
- 로컬 캐시 동기화

※ 시연 코드 : <https://github.com/wharup/microservices-example-sql-vs-api>

감사합니다.