



가상면접 사례로 배우는 대규모 시스템 설계 기초



PPT by 김주혁

목차

- 01 문제 이해 및 설계 범위 확정
- 02 단일 서버 키-값 저장소
- 03 분산 키-값 저장소
- 04 DynamoDB로 살펴보는 실 사례
- 05 요약

6장
키-값 저장소 설계

캬-값 저장소...?

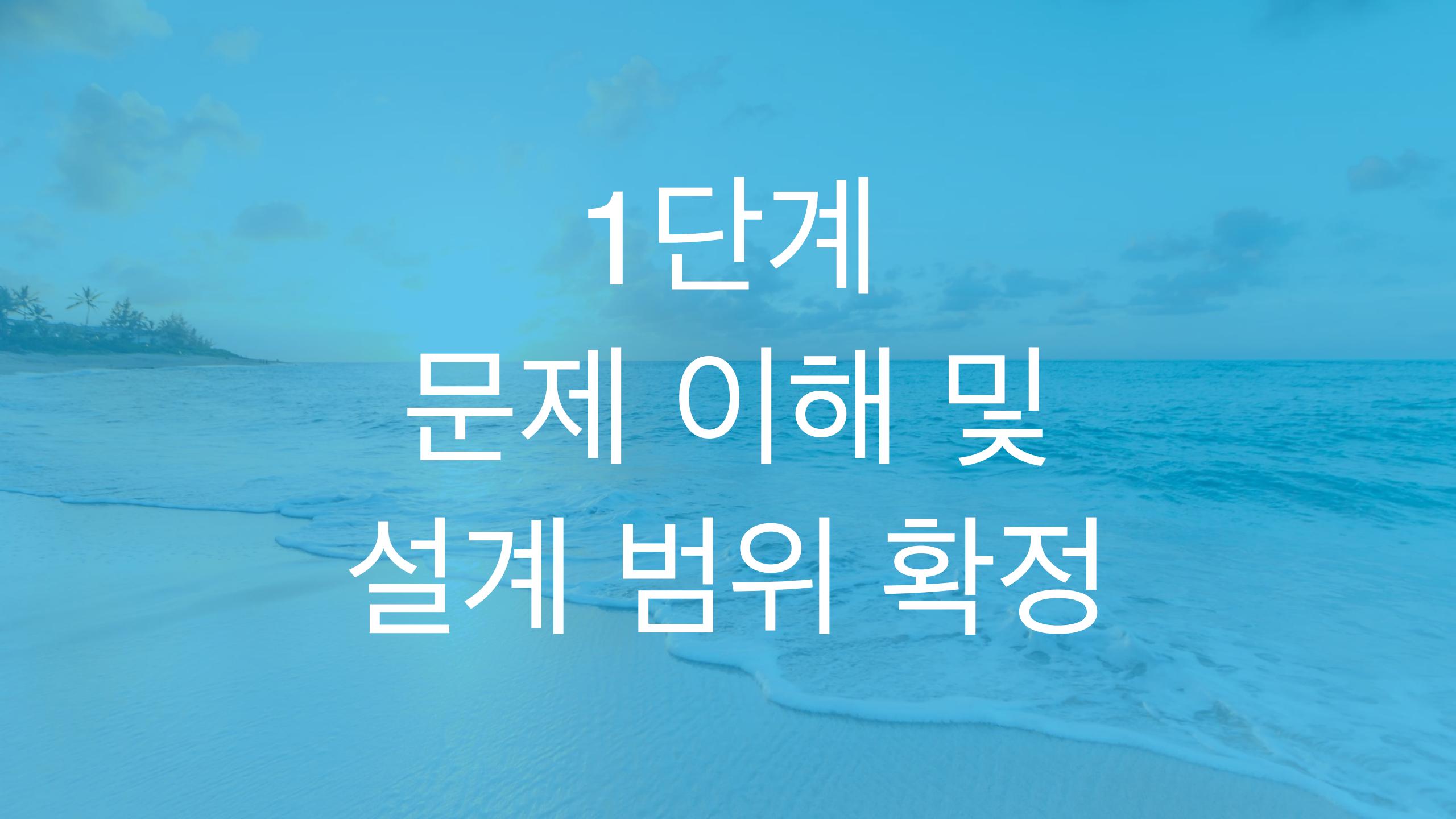




“

No SQL

”

A photograph of a tropical beach at sunset or sunrise. The sky is filled with soft, warm clouds. In the distance, a line of palm trees stands on the shore. The ocean waves are calm and blue. The overall atmosphere is peaceful and scenic.

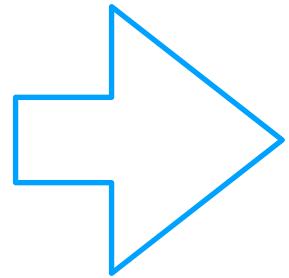
1단계 문제 이해 및 설계 범위 확정

데이터 일관성과 가용성 사이의 타협점

- 키-값 쌍의 크기는 10kb 이하다.
- 큰 데이터를 저장할 수 있어야 한다.
- 높은 가용성을 제공해야 한다.
- 높은 규모 확장성을 제공해야 한다.
- 데이터 일관성 수준은 조정이 가능해야 한다.
- 응답 지연시간(Latency)이 짧아야 한다.

A photograph of a tropical beach. The foreground shows light blue ocean water with small white waves. In the middle ground, a sandy beach meets the water. On the left side, there's a cluster of palm trees. The background is a bright, clear blue sky with a few wispy clouds.

2단계
단일 서버 키-값 저장소



A photograph of a tropical beach at sunset or sunrise. The sky is filled with soft, pastel-colored clouds. In the distance, a line of palm trees stands on the shore. The ocean waves are calm and blue. The overall atmosphere is peaceful and vacation-like.

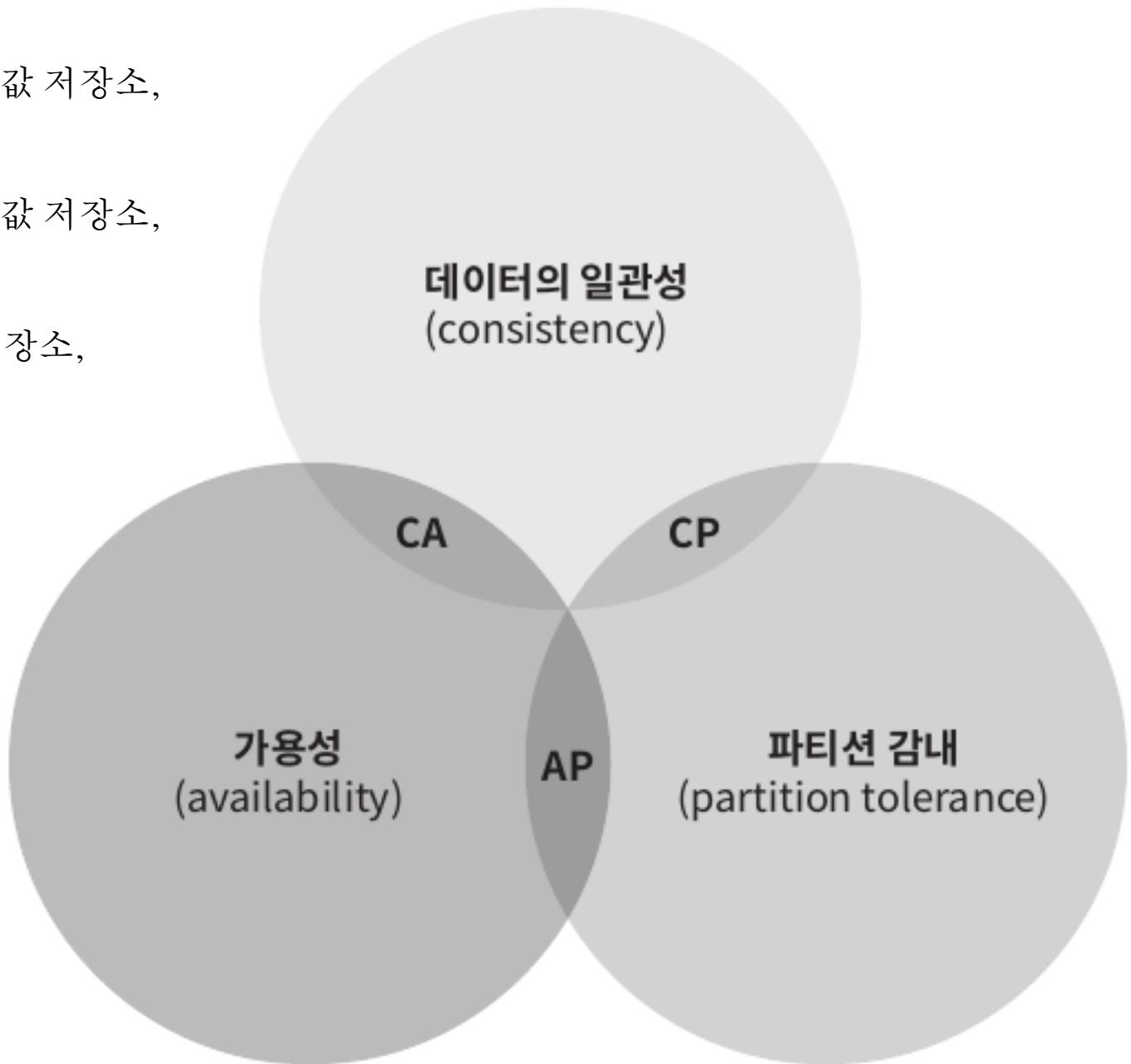
3단계 분산 키-값 저장소



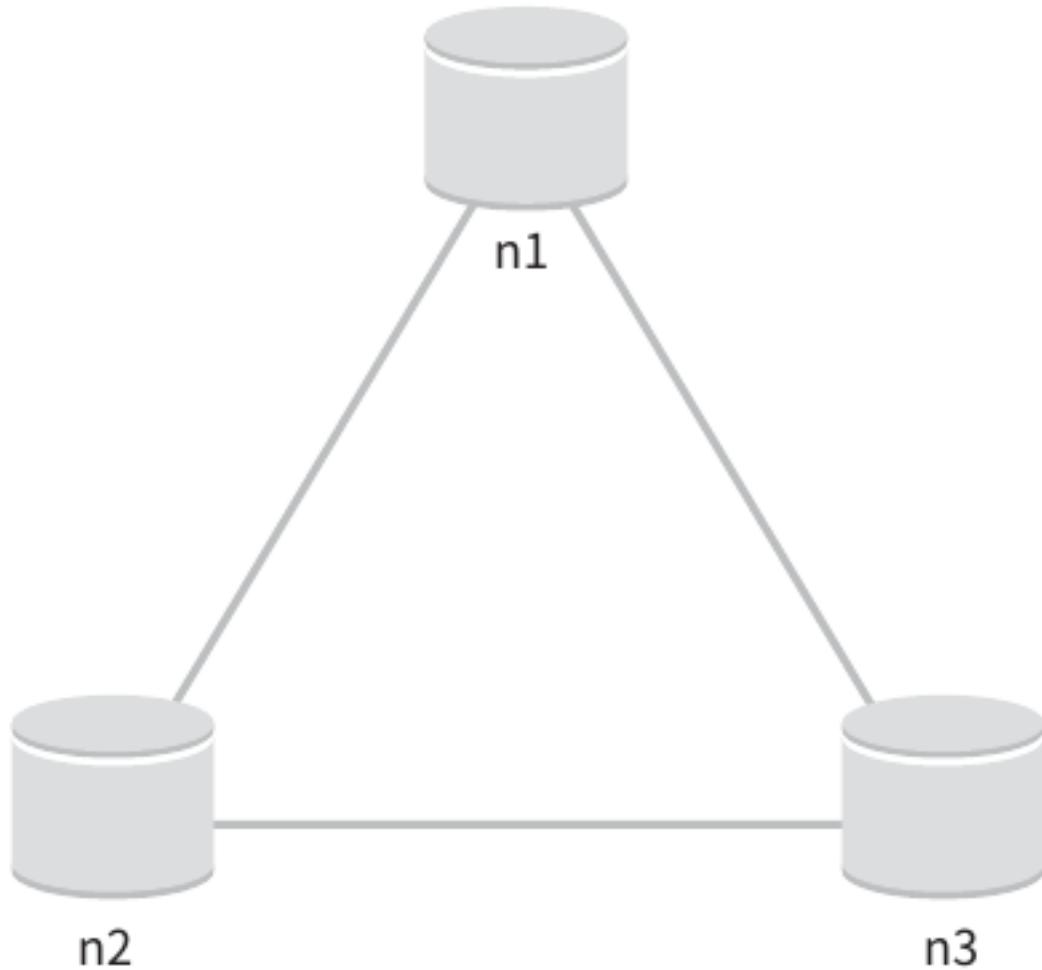
CAP



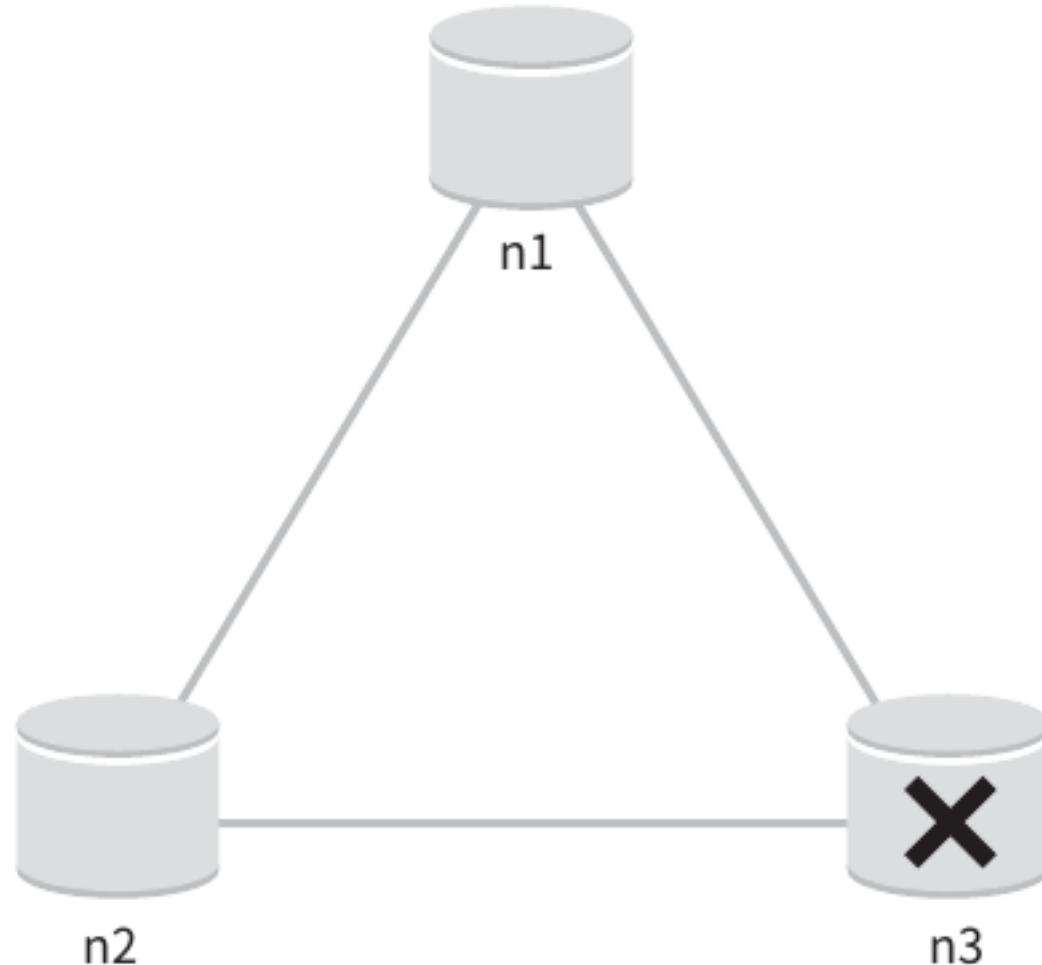
- **CP 시스템** : 일관성과 파티션 감내를 지원하는 키-값 저장소, 가용성을 희생한다.
- **AP 시스템** : 가용성과 파티션 감내를 지원하는 키-값 저장소, 데이터 일관성을 희생한다.
- **CA 시스템** : 일관성과 가용성을 지원하는 키-값 저장소, 파티션 감내는 지원하지 않는다.



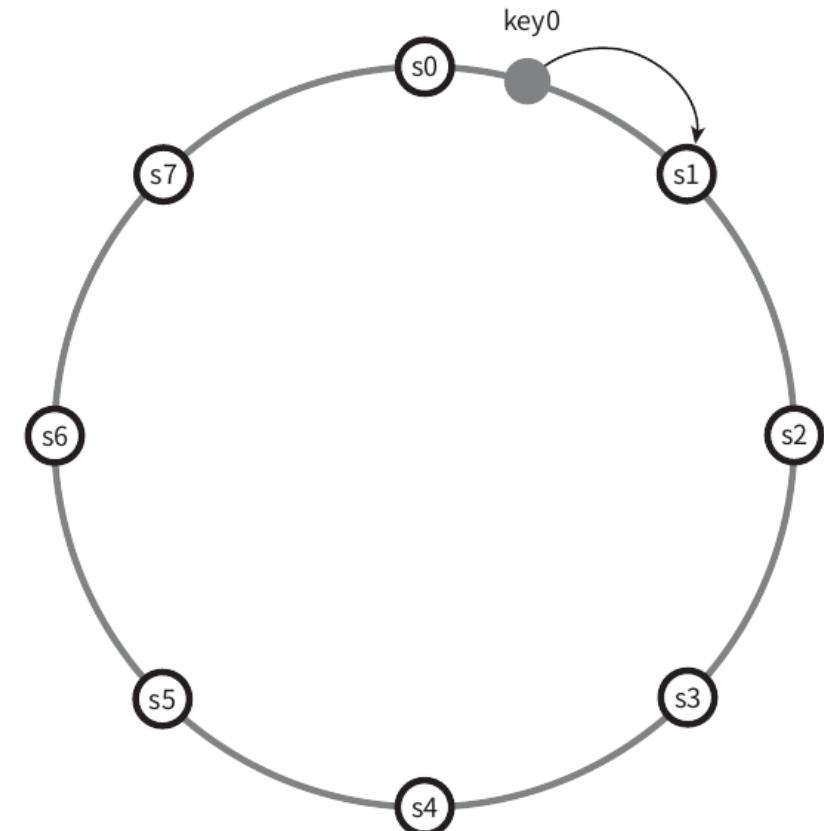
이상적 상태



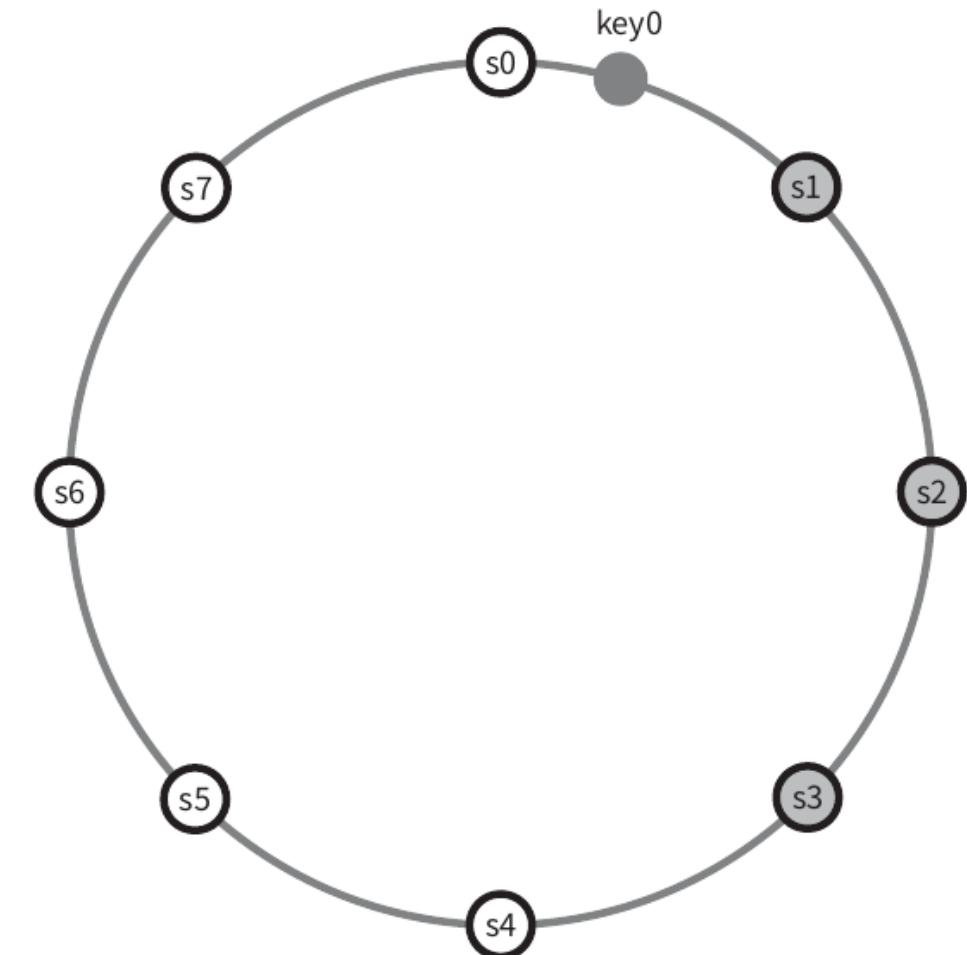
실세계의 분산 시스템



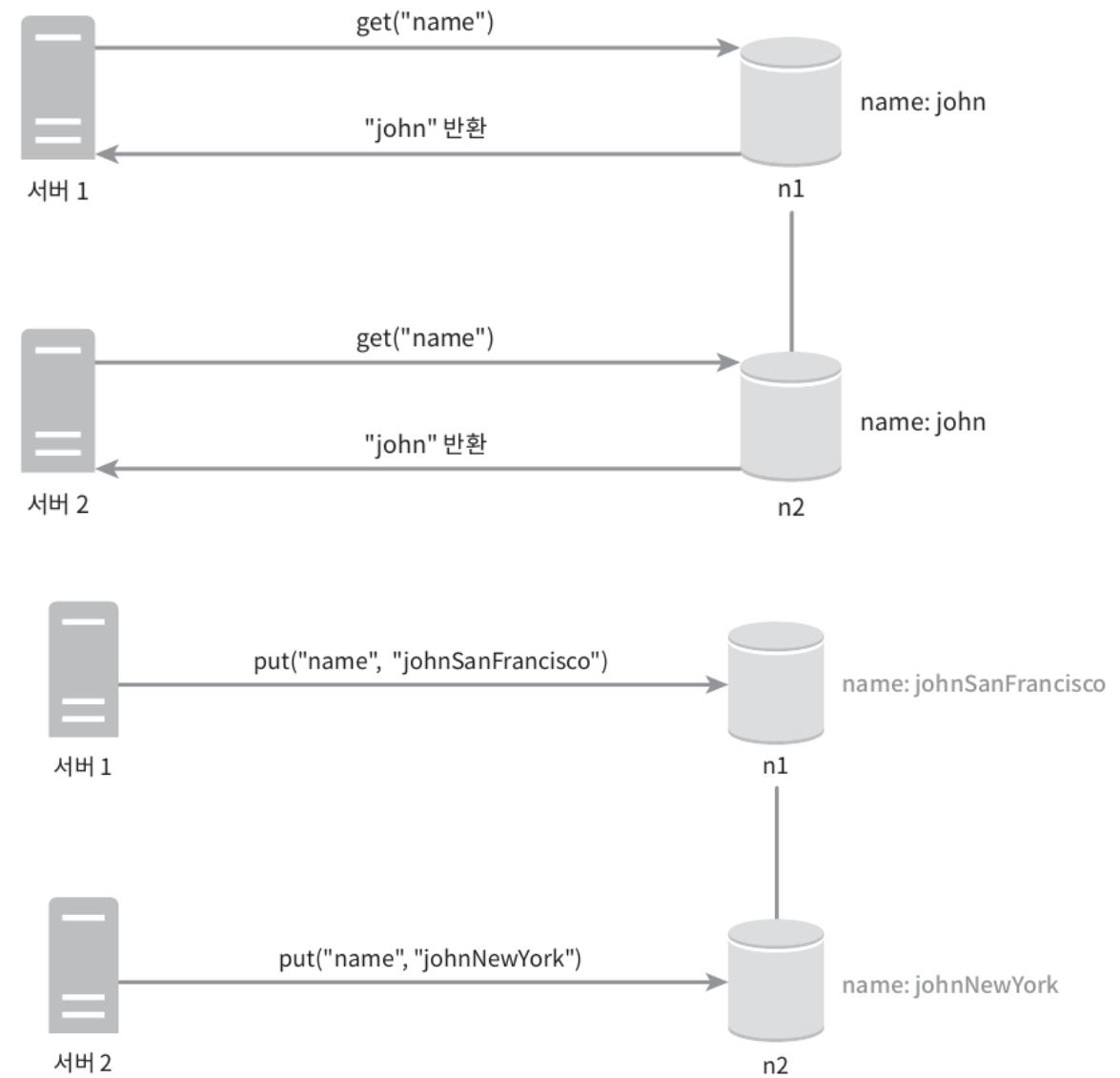
시스템 컴포넌트 데이터 파티션



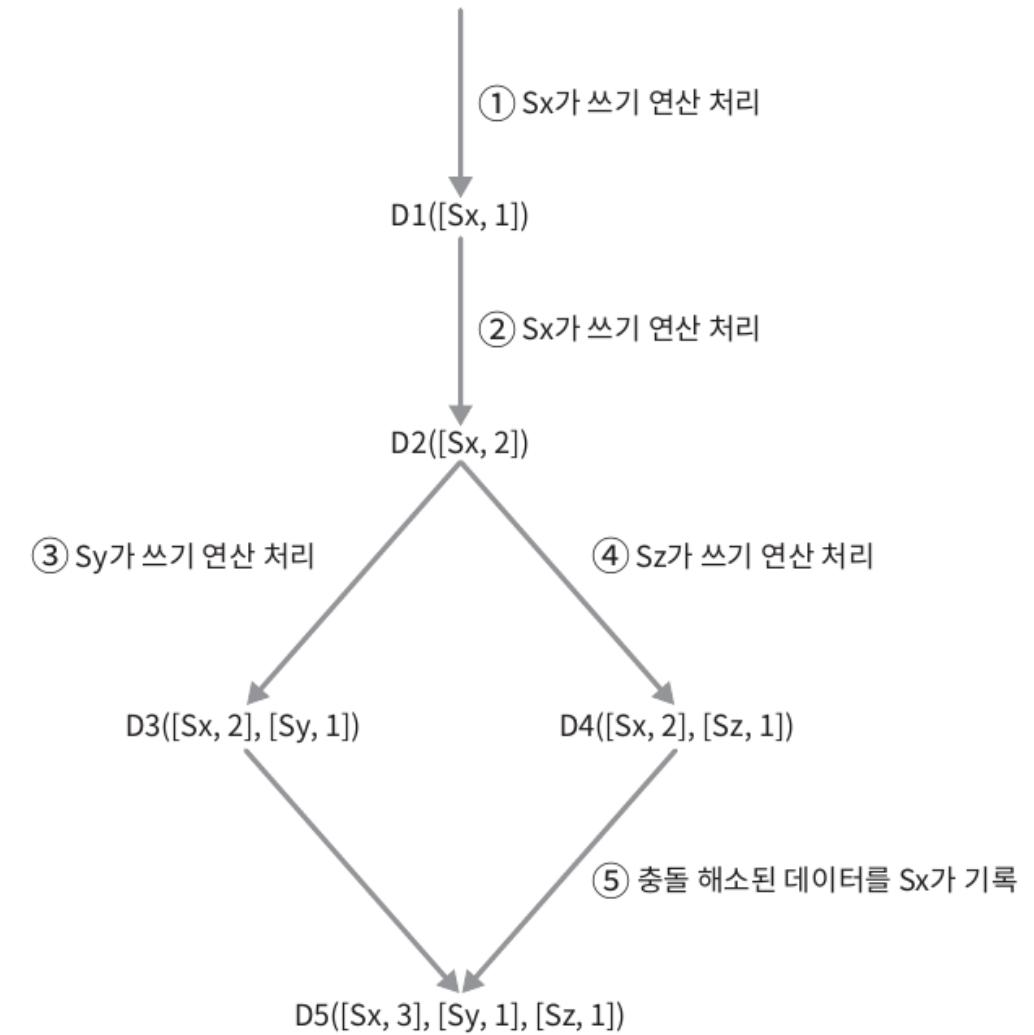
시스템 컴포넌트 데이터 일관성



시스템 컴포넌트 일관성 불일치 해소 데이터 버저닝



시스템 컴포넌트 일관성 불일치 해소 벡터 시계



장애 처리



- 장애 처리

장애 감지 기법을 통해, 장애 처리 기법을 살펴볼 수 있다.

- 장애 감지

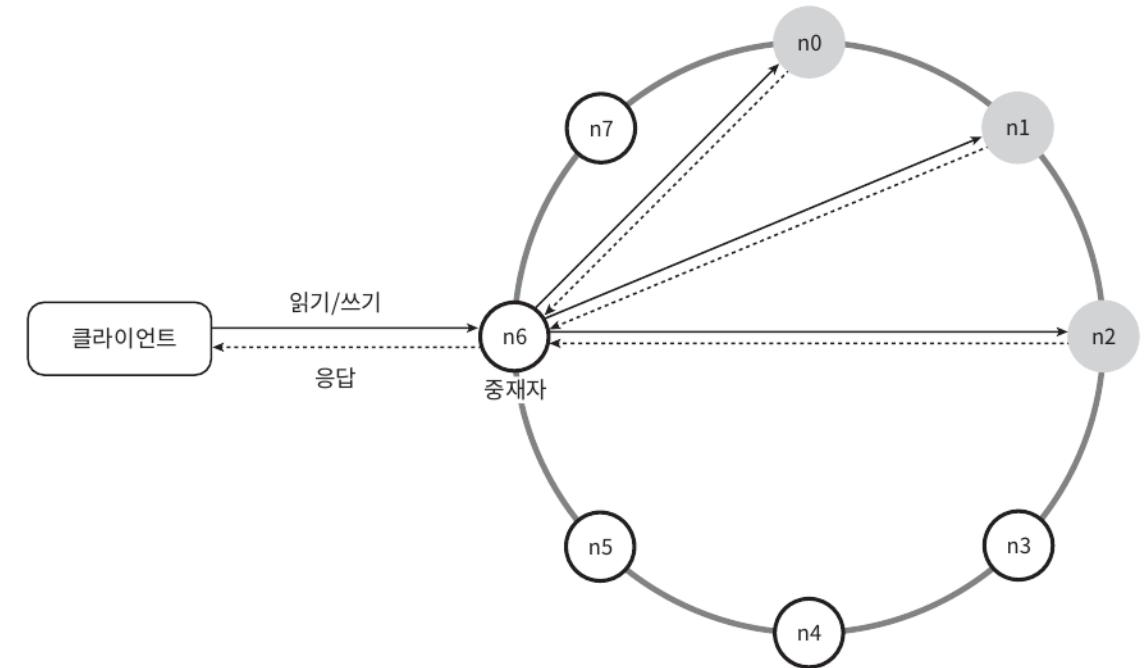
- 모든 노드 사이의 멀티 캐스팅 구축 : 손쉽지만, 서버가 많다면 비효율적
- 가십 프로토콜 같은 분산형 장애 감지 : 보다 더 효율적



장애 처리

- 일시적 장애처리
- 영구적 장애처리
- 데이터 센터 장애 처리

시스템 아키텍처 다이어 그램

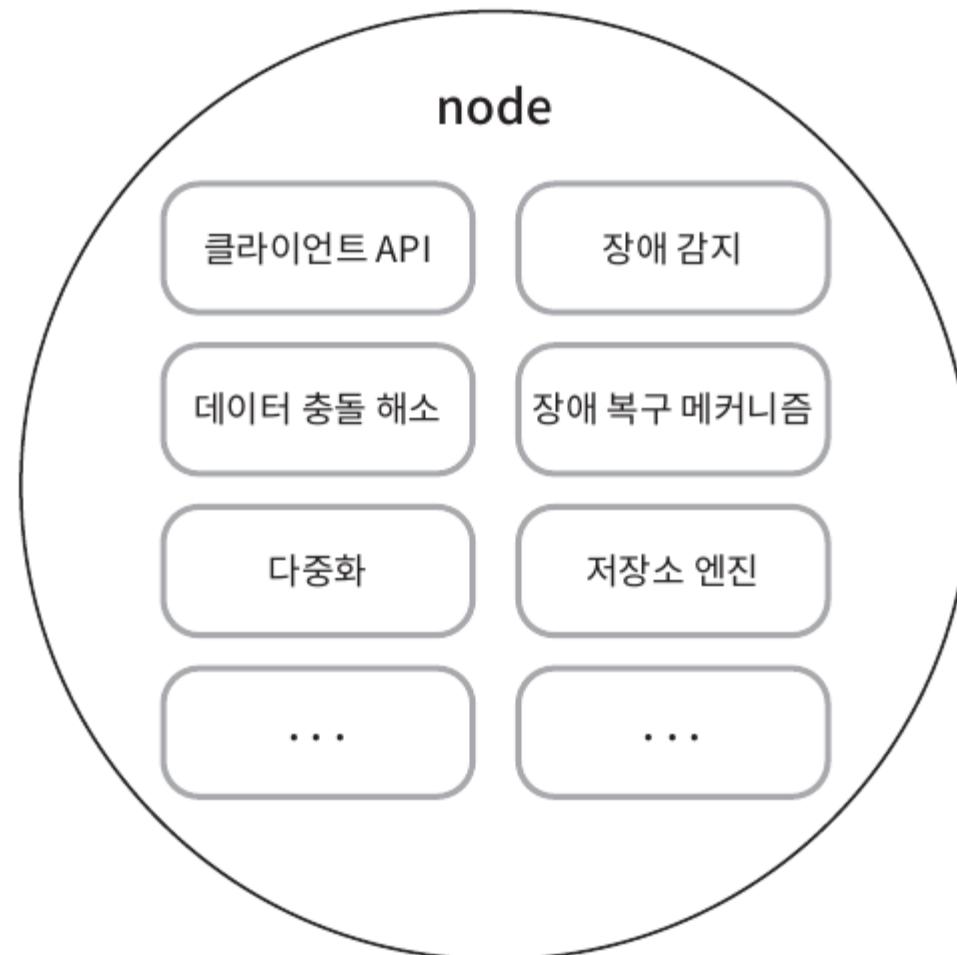


시스템 아키텍처 다이어 그램



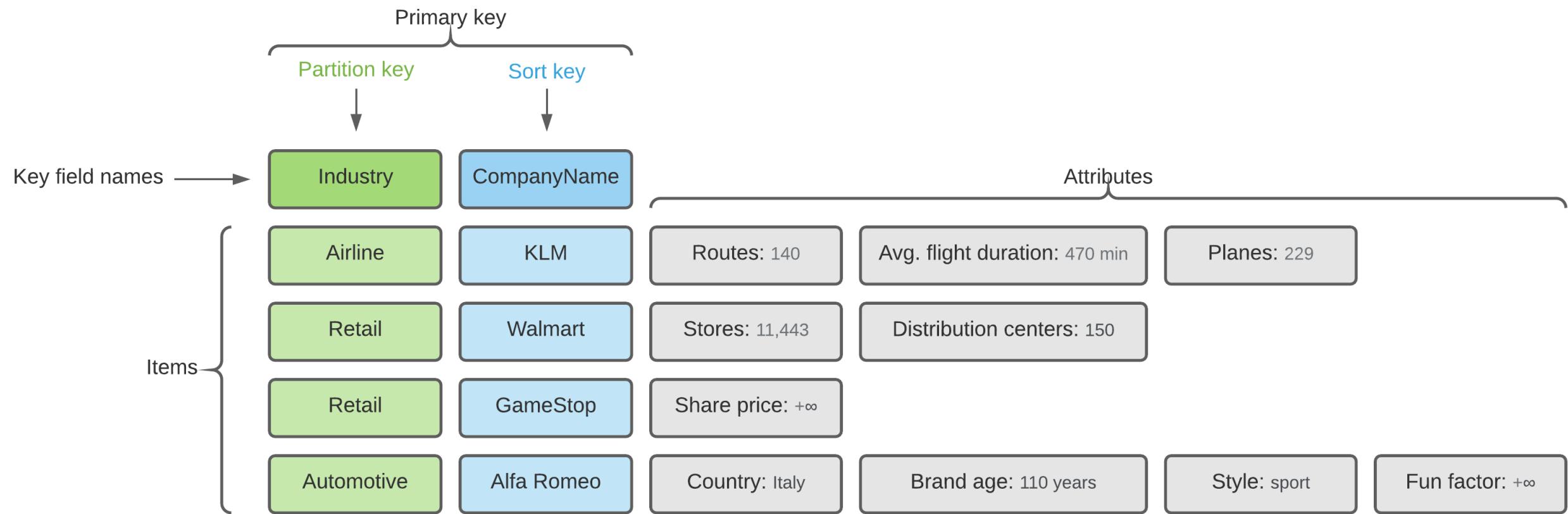
- 클라이언트는 키-값 저장소가 제공하는 두 가지 단순한 API, 즉 get과 통신한다.
- 중재자는 클라이언트에게 키-값 저장소에 대한 proxy 역할을 하는 노드다.
- 노드는 안정 해시의 해시 링위에 분포한다.
- 노드를 자동적으로 추가 또는 삭제할 수 있도록, 시스템은 완전히 분산된다.
- 데이터는 여러 노드에 다중화된다.
- 모든 노드는 같은 책임을 지므로, SPOF는 존재하지 않는다.

완전한 분산 설계

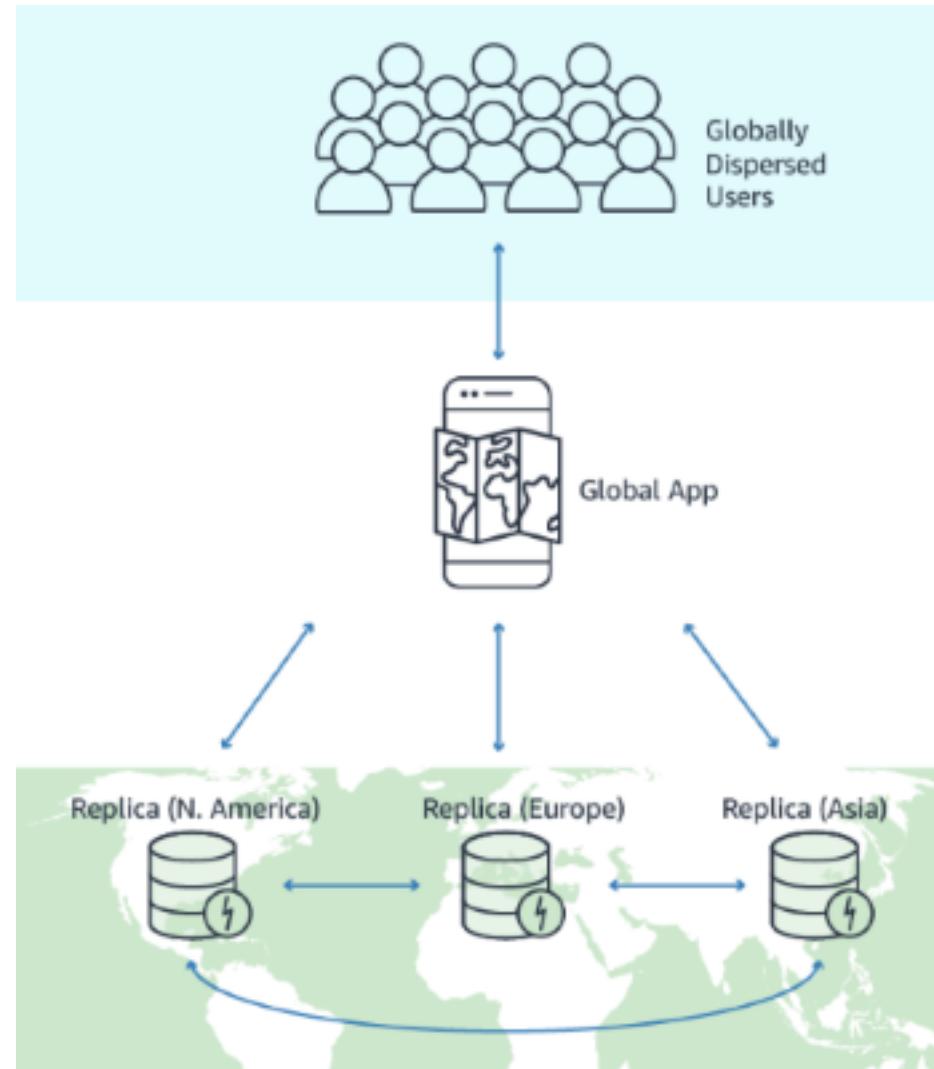


4단계 다이나모 디비로 알아보는 실 사례

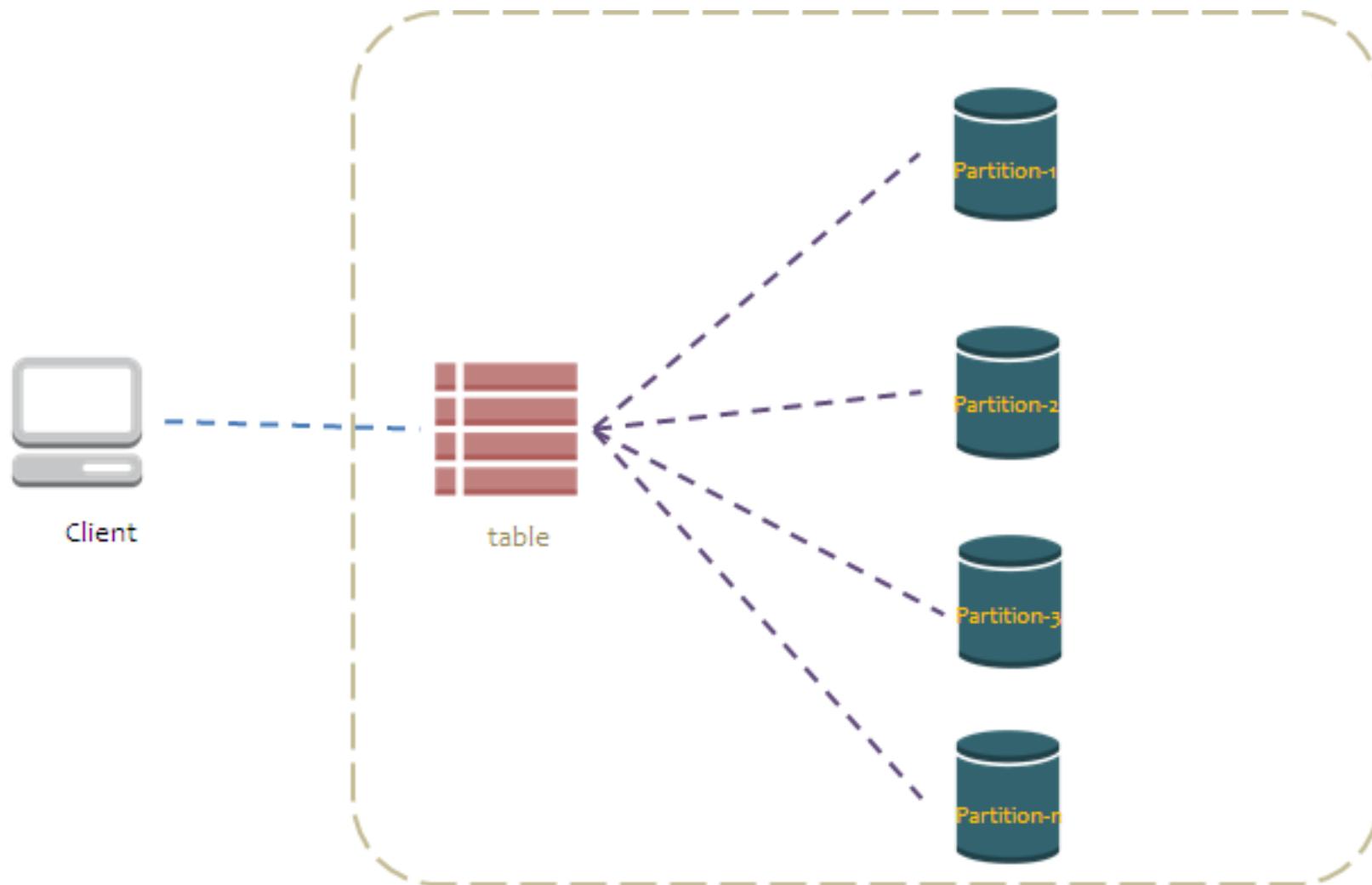
DynamoDB 데이터 파티션



DynamoDB 데이터 일관성



쓰기 경로 / 읽기 경로



5단계
마무리

1. 대규모 데이터 저장 ⇒ 안정 해시를 사용해 서버들에 부하 분산
2. 읽기 연산에 대한 높은 가용성 보장 ⇒ 데이터를 여러 데이터센터에 다중화
3. 쓰기 연산에 대한 높은 가용성 보장 ⇒ 버저닝 및 벡터 시계를 사용한 충돌 해소
4. 데이터 파티션 ⇒ 안정 해시
5. 점진적 규모 확장성 ⇒ 안정 해시
6. 다양성 ⇒ 안정 해시
7. 조절 가능한 데이터 일관성 ⇒ 정족수 합의
8. 일시적 장애 처리 ⇒ 느슨한 정족수 프로토콜과 단서 후 임시 위탁
9. 영구적 장애 처리 ⇒ 머클 트리
10. 데이터 센터 장애 대응 ⇒ 여러 데이터 센터에 걸친 데이터 다중화

출처

<https://www.google.com/search?>

q=%08dynamodb+data+partition&sca_esv=100a8d72724cfbcd&udm=2&biw=1920&bih=883&sxsrf=ACQVn08Aw7gwtt
UWz_LHYECEcS1Pweij4w%3A1711203883195&ei=K-b-
ZcS7C9z71e8PhPGd0AE&ved=0ahUKEwiE_eGcy4qFAxXcffUHHYR4BxoQ4dUDCBA&uact=5&oq=%08dynamodb+d
ata+partition&gs_lp=Egxnd3Mtd2I6LXNlcnAiGAhkeW5hbW9kYiBkYXRhlHBhcnRpdGlvbkiJIIAAWJUhcAB4AJABAJgBj
AGgAaEWqgEEMS4yNLgBA8gBAPgBAZgCD6ACpQ7CAgUQABiABMICCBAAGIAEGLEDwgIEEAYA8ICDhAAGIAE
GloFGLEDGIMBwgIEECMYJ8ICBBAAGB7CAgcQABiABBgTwglIEAAYBRgeGBPCAgcQABiABBgYwgIGEAAYHhgTm
AMAkgcEMC4xNaAHoFw&sclient=gws-wiz-serp#vhid=E-wqfuM6xoodIM&vssid=mosaic,

<https://aws.amazon.com/ko/dynamodb/global-tables/>,

<https://velog.io/@bbkyoo/%EA%B0%80%EC%83%81-%EB%A9%B4%EC%A0%91-%EC%82%AC%EB%A1%80%EB%A1%9C-%EB%B0%B0%EC%9A%B0%EB%8A%94-%EB%8C%80%EA%B7%9C%EB%AA%A8-%EC%8B%9C%EC%8A%A4%ED%85%9C-%EC%84%A4%EA%B3%84-%EA%B8%B0%EC%B4%88-6%EC%9E%A5-%ED%82%A4-%EA%B0%92-%EC%A0%80%EC%9E%A5%EC%86%8C-%EC%84%A4%EA%B3%84>