분산 시스템에서 데이터 일관성 유지 기법 비교하기

분산 시스템에서 **데이터 일관성**은 쉽지 않은 문제로, 다양한 기법이 있습니다. 각 방식은 트레이드오프가 명확하므로 목적과 현장에 따라 선택해야 합니다. 아래에 대표적인 일관성 유지 기법들을 **비교 표(특징/장단점)**로 정리합니다.

기법	설명	장점	단점/제한	적용 예시
강한 일관성 (Strong Consistency / Linearizability)	모든 노드에서 데이터가 즉시 일치. 요청 순서 보장 (쓰기 직후 즉시 읽으면 변경 반영).	데이터 신뢰도/정확성 최상, 은행/거래 등 적합	응답 지연(네트워크 비용↑), 장애 시 가용성 저하	RDBMS+Replication, Spanner, Zookeeper
최종적 일관성 (Eventual Consistency)	일정 시간이 지나면 노드간 데이터가 결국 동일해짐. 즉시 일치 보장 X	높은 가용성/성능, 일부 장애에도 서비스 가능	데이터 최신성 보장 X, 쓰기-읽기 사이 충돌 가능	DynamoDB, Amazon S3, Cassandra
읽기 일관성 (Read-Your-Writes, Monotonic Read)	한 클라이언트가 쓴 데이터는 바로 읽을 수 있음. 다른 클라이언트는 지연될 수 있음	사용자 단위 일관성, UX보장	모든 사용자에 대해 강한 일관성 보장 X	일반 웹/모바일 앱
제한적 일관성 (Session Consistency)	한 세션 사용자에게만 강한 일관성 보장	UX 효과적, 지연 적음	세션 끊기면 보장 X	Redis(세션), 채팅서비스
쿼럼 기반 일관성 (Quorum/Consensus: Paxos, Raft 등)	읽기/쓰기에 과반수 동의 필요. 일관성/가용성 타협점	일관성/가용성 균형, 장애 시 일부 노드 복구 가능	쓰기/읽기 지연 발생(연산과정 필요)	CockroachDB, MongoDB, Cassandra
분산 트랜잭션/2PC(2-Phase Commit), 3PC (Three-Phase Commit)	여러 노드에 걸친 트랜잭션 조율. 실패 시 롤백	일관성 높음, 중요 트랜잭션에 적합	성능 저하(락, 네트워크 트래픽), 장애 시 중단	글로벌 결제, ERP 시스템

핵심 이해 포인트

- **CAP 이론:**
  - 데이터 일관성(Consistency), 가용성(Availability), 네트워크 분할 허용(Partition Tolerance) 중 두 가지만 완벽히 달성 가능.
  - 실시간 거래/은행은 **강한 일관성**에 치중, SNS/로그분석 등은 **최종적 일관성**을 선호함.[1]

- **Consensus 알고리즘(예: Raft, Paxos):**
  - 데이터 갱신 시 여러 노드가 합의하여 일관성 보장. 장애/네트워크 분할 시에도 부분 기능 유지.[1]
- **분산 트랜잭션(2PC, 3PC):**
  - 복수 노드에 걸친 원자적 Commit/Abort. 네트워크 지연, 장애 시 오랜 대기 가능성 있음.

각 기법의 선택은 **서비스 특성, 실제 트래픽, 장애 허용 정도, 데이터 최신성 요구**에 따라 달라집니다. 필요한 경우 추가적인 상세 설명이나 샘플 코드, 적용 구조를 문의해 주세요!

⁂

1. https://systemdesign.one/consistency-patterns/