07. 분산 시스템을 위한 유일 ID 생성기 설계

- 분산 환경에서 auto_incremet 로는 요구를 감당할 수 없음

1단계. 문제 이해 및 설계 범위 확정

- ID는 유일해야
- ID는 숫자로만 구성되어야
- ID는 64비트로 표현할 수 있는 값이어야
- ID는 발급 날짜에 따라 정렬 가능해야
- 초당 10000개 생성이 가능해야

2단계: 개략적인 설계안 제시 및 동의 구하기

다중 마스터 복제

- auto_increment 활용. k(= db 서버 수)만큼 증가
- 단점
  - 여러 데이터센터에 걸쳐 확장이 어려움
  - 시간 흐름에 맞추어 커지도록 보장이 어려움
  - 서버 추가, 삭제시 잘 동작하도록 만들기 어려움

UUID

- 장점
  - 단순. 동기화 이슈도 없음
  - 규모 확장이 쉬움
- 단점
  - 128비트. 길다
  - 시간순 정렬이 불가능
  - 숫자 아닌 값도 포함

티켓 서버

- 데이터 서버를 중앙 집중형으로 하나만 사용
- 장점
  - 유일성 보장
  - 구현이 쉬움. 중소 규모에 적합
- 단점
  - SPOF가 됨.

트위커 스노우플레이크 접근법

- 64비트를 분할
  - 사인비트: 1비트. 나중을 위해 유보.
  - 타임스탬프: 41비트. epoch
  - 데이터센터 ID: 5비트. 32개 데이터센터 지원
  - 서버 ID: 5비트. 데이터센터당 32개 서버 사용 가능
  - 일련번호: 12비트. 각 서버에서 ID 생성시마다 일련번호 1씩 증가

3단계: 상세 설계

- 스노우플레이크 채택하여 설계
- 데이터센터 ID, 서버 ID: 시스템 시작시 결정.
- 타임스탬프, 일련번호: ID 생성시 동작중 생성
- 타임스탬프
  - 시간이 흐르면서 증가. ID는 결국 시간순 정렬
  - 41자리는 69년까지만 동작 가능.
- 일련번호
  - 12비트. 4096개의 값을 가질 수 있음.

4단계: 마무리

- 추가 논의 사항
  - 시계 동기화: ID 생성기들이 전부 같은 시계를 가진다고 가정. 이 가정은 유효하지 않을 수도.
  - 각 절의 길이 최적화: 동시성이 낮고 수명이 긴 애플리케이션이라면, 일련번호 절의 길이를 줄이고 타임스탬프 절을 늘릴 수도.
  - 고가용성: ID 생성기는 필수불가결. 높은 가용성을 제공해야