7️⃣

231004 7. 분산 시스템을 위한 유일 ID 생성기 설계

7. 분산 시스템을 위한 유일 ID 생성기 설계

▼ 1. 문제 이해 및 설계 범위 확정

적절란 질문을 통해 모호함을 없애고 설계 방향을 정해야 함

```
-- 요구사항

1. ID는 유일해야 한다. (unique)
2. ID는 숫자로만 구성되어있어야 한다.
3. ID는 발급 날짜에 따라 정렬 가능해야한다.
4. ID는 64비트로 표현될 수 있는 값이어야한다.
5. ID는 발급 날짜에 따라 정렬 가능해야 한다.
6. 초당 10,000개의 ID를 만들 수 있어야 한다.
```

▼ 2. 개략적 설계안 제시 및 동의 구하기

▼ 다중 마스터 복제

데이터베이스의 **auto_increment** 기능 활용

ID의 값을 구할 때 1만큼 증가시키는 것이 아니라 **k(데이터베이스 서버 수)만큼 증가**

장점

- 규모 확장성 문제를 어느정도 해결할 수 있음
- 데이터베이스 수를 늘리면 초당 생산 가능 ID수도 늘릴 수 있음

단점

- 여러 데이터 센터에 걸쳐 규모를 늘리기 어려움
- ID의 유일성은 보장되지만 그 값이 시간 흐름에 맞추어 커지도록 보장할 수 없음

- 서버를 추가하거나 삭제할 때도 잘 동작하도록 만들기 어려움

▼ UUID

컴퓨터 시스템에 저장되는 정보를 유일하게 식별하기 위한 128비트 수

```
09c93e62-50b4-468d-bf8a-c07e1040bfb2
```

장점

- 충돌 가능성이 지극히 낮음
- 만들기 단순함
- 서버 사이의 조율이 필요 없으므로 동기화 이슈도 없음
- 각 서버가 자기가 쓸 ID를 알아서 만드는 구조이므로 확장도 쉬움

단점

- ID가 128비트
- ID를 시간순으로 정렬할 수 없음
- ID에 숫자가 아닌 값이 포함될 수 있음
- 이번 문제에서 사용 불가능

▼ 티켓 서버

티켓 서버(auto_increment 기능을 갖춘 데이터베이스 서버)를 중앙 집중형으로 하나만 사용하는 것

장점

- 숫자로만 구성된 ID를 쉽게 만들 수 있음
- 구현하기 쉬움
- 중소 규모 애플리케이션에 적합

단점

- 티켓 서버가 SPOF가 됨

- SPOF 이슈를 해결하기 위해 티켓 서버를 여러 대 준비하면 데이터 동기화같은 새로운 문제가 발생

▼ 트위터 스노플레이크

각개격파 전략/분할정복(divide and conquer)을 사용한 독창적인 ID 생성 기법

- sign bit
  - 1비트
  - 음수와 양수를 구별하는데 사용
- timestamp
  - 41비트
  - 기원 시각(epoch) 이후로 몇 밀리초가 경과했는지를 나타내는 값
- 데이터센터 ID
  - 5비트
  - 2^5(32)개의 데이터센터 지원 가능
- 서버 ID
  - 5비트
  - 데이터센터당 32개 서버 사용 가능
- 일련번호(sequence)
  - 12비트
  - 각 서버에서는 ID를 생성할 때마다 이 일련번호를 1만큼 증가시킴
  - 해당 값은 1밀리초가 경과할 때마다 0으로 초기화됨

▼ 3. 상세 설계: 트위터 스노플레이크

데이터센터 ID와 서버 ID는 시스템이 시작할 때 결정

일반적으로 시스템 운영 중에는 바뀌지 않음

데이터센터 ID나 서버 ID를 잘못 변경하게 되면 ID 충돌이 발생할 수도 있음

타임스탬프

앞서 살펴본 ID 구조에서 가장 중요한 41비트를 차지하고 있음

시간이 흐름에 따라 점점 큰 값을 갖게 되기 때문에 시간순으로 정렬 가능하게 됨

이진 표현 형태로부터 UTC 시각을 추출하는 예제

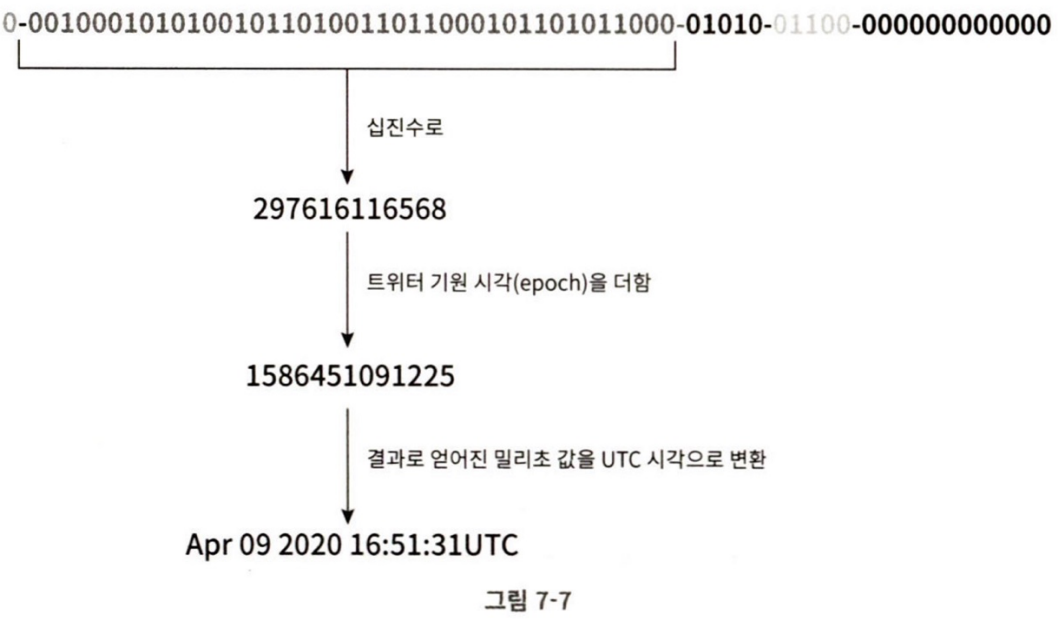

그림 7-7

- 이 방법을 역으로 적용하면 어떤 UTC 시각도 상술한 타임스탬프 값으로 변환 가능
- 41비트로 표현할 수 있는 타임스탬프의 최댓값 → 2^41 – 1 = 2199023255551밀리초(69년)
- 기원시각을 현재랑 가깝게 하면 오버플로가 발생하는 시점을 늦춰놓는 것이 됨
- 69년이 지나면 기원 시각을 바꾸거나 ID 체계를 다른 것으로 이전해야 함

일련변호

12비트이므로 2^12(4096)개의 값을 가질 수 있음

어떤 서버가 같은 밀리초 동안 하나 이상의 ID를 만들어 낸 경우에만 0보다 큰 값을 갖게 됨

▼ 4. 마무리

설계 이후에 추가로 논의할 수 있는 주제

- 시계 동기화(clock synchronization)
  - 이번 설계에서는 ID 생성 서버들이 전부 같은 시계를 사용한다고 가정하였지만 이런 가정은 하나의 서버가 여러 코어에서 실행될 경우 유효하지 않을 수 있음
  - NTP(Network Time Protocol)는 이 문제를 해결하는 가장 보편적 수단

- 각 섹션의 길이 최적화
    - 동시성이 낮고 수명이 긴 애플리케이션이라면 일련번호 절의 길이를 줄이고 타임스탬프 절의 길이를 늘리는 것이 효과적일 수 있음
- 고가용성(high availability)
    - ID 생성기는 필수 불가결(mission critical) 컴포넌트 이므로 아주 높은 가용성을 제공해야 할 것

▼ 토론

유일 ID를 생성할 때 auto_increment를 사용하지 않을 경우 UUID와 스노우플레이크를 많이 사용하는 것 같은데 각각 어떤 경우에 사용하는 것이 좋을까

UUID

- 국제적으로 고유한 식별자가 필요한 경우.
- 다양한 플랫폼 및 언어 간 호환성이 필요한 경우.
- ID의 순서나 시간과 무관하게 고유성이 중요한 경우.
- UUID 버전 4를 사용하면 충돌 가능성이 매우 낮으므로 무작위한 ID가 요구되는 경우.

Snowflake

- 순차적인 ID가 필요한 경우, 예를 들어 시간 순서대로 정렬해야 하는 데이터베이스 테이블에서.
- 분산 환경에서 노드 간 충돌 없이 고유한 ID가 필요한 경우.
- 상대적으로 작은 크기의 ID가 필요한 경우.