7장. 분산 시스템을 위한 유일 ID 생성기 설계

Created	@October 27, 2022 11:27 AM
Progress	In Progress

학습 TODO list

- [ ] 질의응답을 통한 요구사항에서, ID는 64비트로 표현될 수 있는 값이어야 한다는 근거

> *auto_increment 속성이 설정된 관계형 데이터 베이스의 기본 키는 분산 환경에서 사용할 수 없다. 데이터베이스 서버 한 대로는 그 요구를 감당할 수 없고, 여러 데이터베이스 서버 한 대로는 지연시간(delay)을 낮추기가 어렵다.*

👩‍💻 `유일성`이 보장되는 `ID 생성기` 구현에 쓰일 수 있는 다양한 전략

- `다중 마스터 복제`
- `UUID`
- `티켓 서버`
- `트위터 스노플레이크` → 모든 요구사항을 만족하면서도 분산 환경에서 규모 확장이 가능함

7.1. 1단계: 문제 이해 및 설계 범위 확정

> *적절한 질문을 통해 모호함을 없애고 설계 방향을 정한다.*

👩‍💻 ID는 어떤 특성을 갖나요?

👨‍💻 **ID는 유일해야 하고, 정렬 가능해야 합니다.**

👩‍💻 새로운 레코드에 붙일 ID는 항상 1만큼 큰 값이어야 하나요?

👨‍💻 ID의 값은 시간이 흐름에 따라 커질 테지만 언제나 1씩 증가한다고 할 수는 없습니다. 다만 확실한 것은, **아침에 만든 ID보다는 저녁에 만든 ID가 큰 값을 갖는다**는 점입니다.

👩‍💻 **ID는 숫자로만 구성되나요?**

👨‍💻 **그렇습니다.**

👩‍💻 시스템 규모는 어느 정도입니까?

👨‍💻 **초당 10,000개의 ID를 생성할 수 있어야 합니다.**

> *질의 응답을 통해 파악한 요구사항*

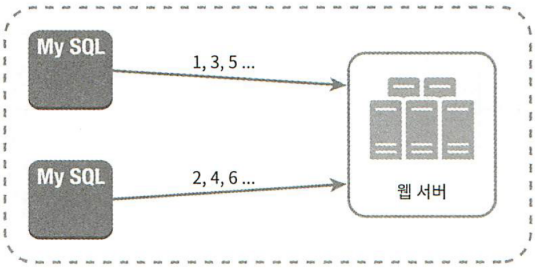
- ID는 유일해야 한다.
- ID는 숫자로만 구성되어야 한다.
- ID는 64비트로 표현될 수 있는 값이어야 한다.
- ID는 발급 날짜에 따라 정렬 가능해야 한다.
- 초당 10,000개의 ID를 만들 수 있어야 한다.

7.2. 2단계: 개략적 설계안 제시 및 동의 구하기

👩‍💻 분산 시스템에서 유일성이 보장되는 ID를 만드는 방법

- 다중 마스터 복제(multi-master replication)
- UUID(Universally Unique Identifier)
- 티켓 서버(ticket server)
- 트위터 스노플레이크(twitter snowflake) 접근법

7.2.1. 다중 마스터 복제



다중 마스터 복제 구성

- `auto_increment` 기능을 활용하여 `k`만큼 증가시킨다.
  - `k`: 현재 사용 중인 데이터베이스 서버의 수

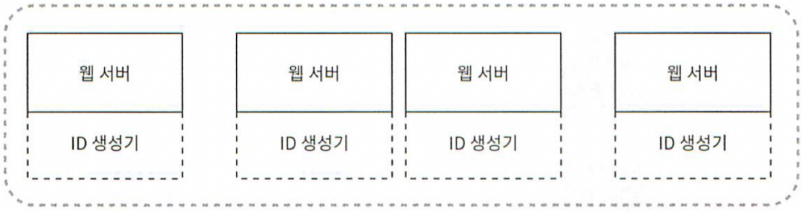
👩‍💻 `다중 마스터 복제` 방법의 장점

- 규모 확장성 문제를 어느 정도 해결할 수 있다.
  - 데이터베이스 수를 늘리면 초당 생산 가능한 ID 수도 늘릴 수 있다.

👩‍💻 `다중 마스터 복제` 방법의 단점

- 여러 데이터 센터에 걸쳐 규모를 늘리기 어렵다.
- ID의 유일성은 보장되지만 그 값이 시간 흐름에 맞추어 커지도록 보장할 수 없다.
- 서버를 추가하거나 삭제할 때도 잘 동작하도록 만들기 어렵다.

7.2.2. UUID



UUID를 사용하는 시스템의 구조

- `UUID`: 컴퓨터 시스템에 저장되는 정보를 유일하게 식별하기 위한 128비트짜리 수
  - ex. `09c93e62-50b4-468d-bf8a-c07e1040bfb2`
  - 충돌 가능성이 낮다.
  - 서버 간 조율 없이 독립적으로 생성 가능하다.

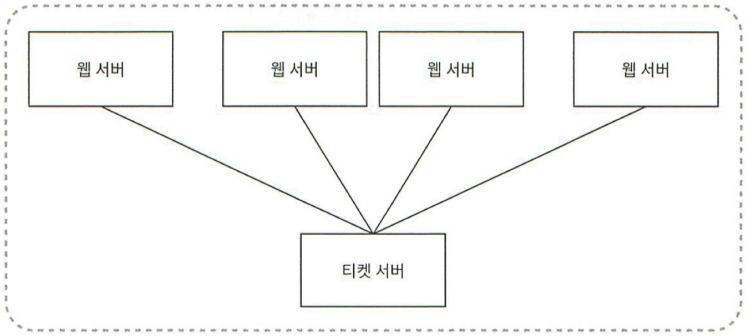
👩‍💻 `UUID` 방법의 장점

- `UUID`를 만드는 것은 단순하다. 서버 사이의 조율이 필요 없으므로 `동기화` 이슈도 없다.
- 각 서버가 자기가 쓸 ID를 알아서 만드는 구조이므로 규모 확장도 쉽다.

👩‍💻 `UUID` 방법의 단점

- ID가 `128비트`로 길다.
  - 문제의 요구사항은 `64비트`이다.
- ID를 시간순으로 정렬할 수 없다.
- ID에 숫자(numeric)가 아닌 값이 포함될 수 있다.

7.2.3. 티켓 서버



티켓 서버 구성

- `auto_increment` 기능을 갖춘 데이터베이스 서버(`티켓 서버`)를 중앙 집중형으로 하나만 사용한다.
- Flickr에서 분산 기본 키(distributed primary key)를 만드는 방법으로 `티켓 서버`를 사용한다.

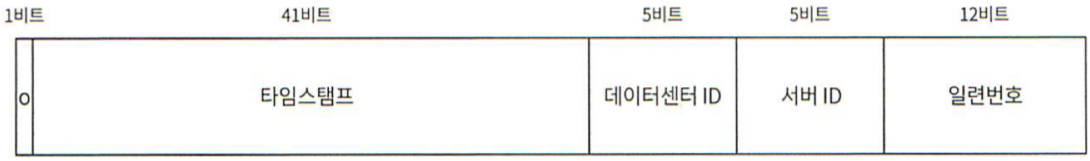
`티켓 서버` 방법의 장점

- 유일성이 보장되는 오직 숫자로만 구성된 ID를 쉽게 만들 수 있다.
- 구현하기 쉽고, 중소 규모 애플리케이션에 적합하다.

`티켓 서버` 방법의 단점

- `티켓 서버`가 `SPOF(Single-Point-of-Failure)`가 된다.
  - 이 서버에 장애가 발생하면, 해당 서버를 이용하는 모든 시스템이 영향을 받는다.
  - 이 이슈를 피하려면 티켓 서버를 여러 대 준비해야하는데, 그러면 `데이터 동기화` 같은 새로운 문제가 발생한다.

7.2.4. 트위터 스노플레이크 접근법



트위터 스노플레이크 접근법의 64비트 ID 구조

- ID를 바로 생성하는 대신, `분할정복(divide and conquer)`을 적용한다.
  - 생성해야 하는 ID의 구조를 여러 절(section)로 분할한다.
    - 사인(sign) 비트: `1비트`를 할당한다. 음수와 양수를 구별하는데 사용할 수 있다.
    - 타임스탬프(timestamp): `41비트`를 할당한다. 기원 시각(epoch) 이후로 몇 밀리초(millisecond)가 경과했는지 나타낸다.(`ID 생성기`가 돌고 있는 중에 만들어지는 값)
    - 데이터센터 ID: `5비트`를 할당한다.(시스템이 시작할 때 결정되는 값)
      - 2^5=32개 데이터센터를 지원할 수 있다.
    - 서버 ID: `5비트`를 할당한다.(시스템이 시작할 때 결정되는 값)

    - 데이터센터 당 32개의 서버를 사용할 수 있다.
- 일련번호: `12비트`를 할당한다. 각 서버에서 ID를 설정할 때마다 일련번호를 1만큼 증가시킨다. 이 값은 1밀리초가 경과할 때마다 0으로 초기화(reset)된다.(`ID 생성기`가 돌고 있는 중에 만들어지는 값)

> *요구사항을 모두 만족한다.*

👩‍💻 `트위터 스노플레이크 접근법` 방법의 유의점

- 데이터센터 ID와 서버 ID는 시스템이 시작할 때 결정되며, 일반적으로 시스템 운영 중에는 바뀌지 않는다.
    - 데이터센터 ID나 서버 ID를 잘못 변경하게 되면 ID 충돌이 발생할 수 있다.

7.3. 3단계: 상세 설계

> *트위터 스노플레이크 접근법을 사용하여 상세한 설계를 진행해보자.*

7.3.1. 타임스탬프

트위터 스노플레이크 접근법에서 UTC 시각을 추출하는 방법(역으로 적용하면 UTC 시각을 타임스탬프 값으로 변환할 수 있다.)

- 타임스탬프는 `트위터 스노플레이크 접근법`의 ID 구조에서 가장 중요한 `41비트`를 차지하고 있다.
- 타임스탬프는 시간이 흐름에 따라 점점 큰 값을 가지므로 ID는 시간순으로 정렬 가능하다.
- 41비트로 표현할 수 있는 타임스탬프의 최댓값은 2^{41} `-1=2199023255551밀리초=약 69년`이다.
  - `트위터 스노플레이크 접근법`의 ID 생성기는 69년 동안만 정상 동작한다.
  - 기원 시각을 현재에 가깝게 맞춰서 `오버플로(overflow)`가 발생하는 시점을 늦춘다.
  - 69년이 지나면 기원 시각을 바꾸거나 ID 체계를 다른 것으로 이전(migration)해야 한다.

7.3.2. 일련번호

- 일련번호는 `12비트`이므로 2^{12}=4096개의 값을 가질 수 있다.
  - 어떤 서버가 밀리초 동안 하나 이상의 ID를 만들어 낸 경우만 0보다 큰 값을 갖는다.

7.4. 4단계: 마무리

추가 논의 사항

- `시계 동기화(clock synchronization)` : `ID 생성` 서버들이 전부 같은 시계를 사용한다는 가정은 하나의 서버가 여러 코어에서 실행될 경우 유효하지 않을 수 있다. 여러 서버가 물리적으로 독립된 여러 장비에서 실행되는 경우도 마찬가지다.
  - `NTP(Network Time Protocol)`로 문제를 해결할 수 있다.
- `각 절(section)의 길이 최적화` : `동시성(concurrency)`이 낮고 수명이 긴 애플리케이션이라면 `일련번호` 절의 길이를 줄이고 `타임스탬프` 절의 길이를 늘리는 것이 효과적일 수 있다.
- `고 가용성(high availability)` : `ID 생성기`는 `필수 불가결(mission critical) 컴포넌트`이므로 아주 높은 가용성을 제공해야 할 것이다.