8장

URL 단축키 설계

1단계 문제 이해 및 설계 범위 확정

- 쓰기 연산, 초당 쓰기 연산, 읽기 연산 계산을 통해 설계 범위를 추정한다.

2단계 개략적 설계안 제시 및 동의 구하기

API 엔드포인트

1. URL 단축용 엔드포인트 : 새 단축 URL을 생성하고자 하는 클라이언트가 단축할 URL을 인자로 실어 POST 요청을 보내는 곳
2. URL 리디렉션용 엔드포인트 : 단축 URL에 대해 HTTP 요청이 오면 원래 URL로 보내주기 위한 용도의 엔드포인트

URL 리디렉션

1. 301 Permanently Moved : 주어진 URL에 대한 HTTP 요청의 처리 책임이 영구적으로 Location 헤더에 반환된 URL로 이전되었다는 응답.
    - 서버 부하를 줄일 수 있다.
2. 302 Found : 주어진 URL로의 요청이 일시적으로 Location 헤더가 지정하는 URL에 의해 처리되어야 함. 따라서 언제나 단축 URL 서버에 먼저 보내진 후 원래 URL로 리디렉션 되어야 함.
    - 트래픽 분석이 요구될 때 사용하는 것이 좋음.

URL 단축

해시 함수를 사용한다.

1. 주어지는 긴 URL이 다른 값이면 해시 값도 달라야 한다.

2. 계산된 해시 값은 원래 입력으로 주어졌던 긴 URL로 복원될 수 있어야 한다.

3단계 상세 설계

데이터 모델

- 해시 테이블
  - 초기 전략으로는 나쁘지 않지만, 메모리를 사용해야 한다는 단점이 있기 때문에 데이터베이스를 활용하는 것이 좋다.

해시 함수

- 해시 값 길이
  - 시스템이 서비스 되는 동안 만들어야 하는 URL의 개수에 맞춰 최소 길이를 설정한다.
- 해시 충돌 해소
  - 해시 함수
    - 충돌이 발생하는 경우 데이터를 덧붙여 해결할 수 있다. 하지만 URL을 생성할 때 한 번 이상 데이터베이스 질의를 해야 하는 단점이 있다.

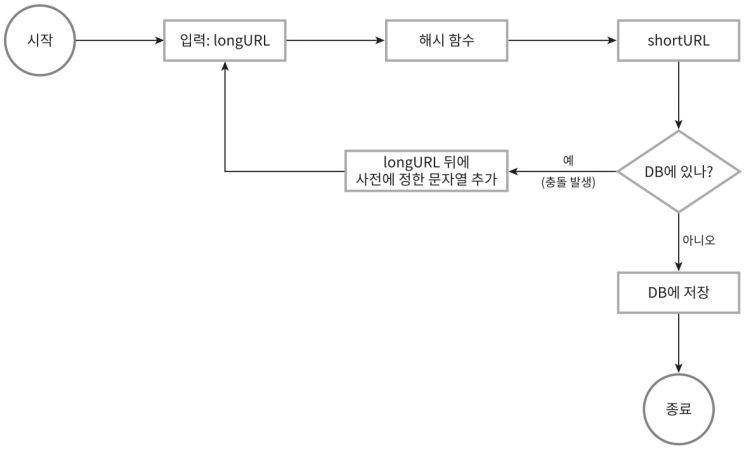

그림 8-5

  - base-62 변환
    - URL을 62진법으로 생성한다.

- $11157_{10} = 2 \times 62^2 + 55 \times 62^1 + 59 \times 62^0 = [2, 55, 59] \Rightarrow [2, T, X] \Rightarrow 2TX_{62}$이다. 그림 8-6에 이 변환 과정을 요약하여 보았다.

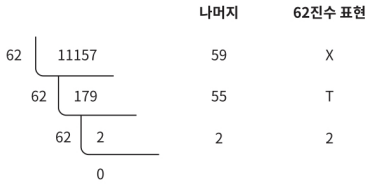

그림 8-6

- 따라서 단축 URL은 *https://tinyurl.com/2TX*가 된다.

∘ 비교

해시 후 충돌 해소 전략	base-62 변환
단축 URL의 길이가 고정됨	단축 URL의 길이가 가변적. ID 값이 커지면 같이 길어짐
유일성이 보장되는 ID 생성기가 필요치 않음	유일성 보장 ID 생성기가 필요
충돌이 가능해서 해소 전략이 필요	ID의 유일성이 보장된 후에야 적용 가능한 전략이라 충돌은 아예 불가능
ID로부터 단축 URL을 계산하는 방식이 아니라서 다음에 쓸 수 있는 URL을 알아내는 것이 불가능	ID가 1씩 증가하는 값이라고 가정하면 다음에 쓸 수 있는 단축 URL이 무엇인지 쉽게 알아낼 수 있어서 보안상 문제가 될 소지가 있음

URL 단축기 상세 설계

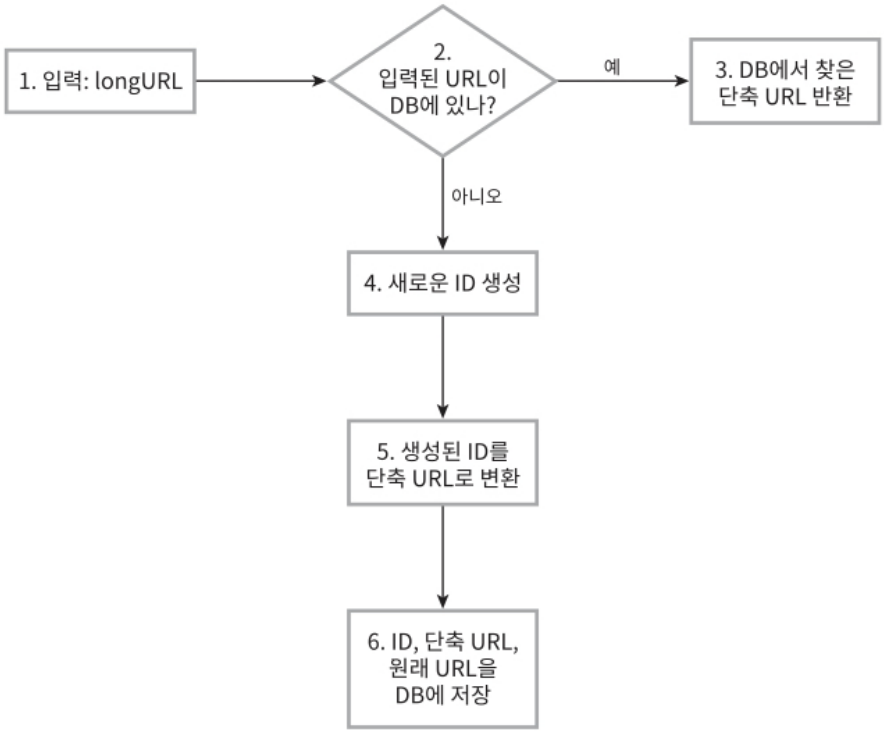

그림 8-7

- ID 생성기
  - 단축 URL과 긴 URL의 기본키로 사용하기 위한 ID를 생성해야 한다.

URL 리디렉션 상세 설계

1. 로드밸런서가 발생한 요청을 웹 서버에 전달.
2. 단축 URL이 캐시에 있는 경우
   a. 원래 URL을 꺼내서 클라이언트에게 전달.
3. 단축 URL이 캐시에 없는 경우
   a. 데이터베이스에 있다면 꺼내서 전달한다.
   b. 데이터베이스에 없다면 잘못된 단축 URL을 입력한 경우이다.
4. 데이터베이스에서 꺼낸 URL을 캐시에 넣은 후 사용자에게 반환한다.

4단계 마무리

- 논의하면 좋은 것

- 처리율 제한 장치
- 웹 서버의 규모 확장
- 데이터베이스의 규모 확장
- 데이터 분석 솔루션
- 가용성, 데이터 일관성, 안정성