URL 단축기 설계

URL 단축기 설계

1. 문제 이해 및 설계 범위 확정

2. 개략적 설계안 제시 및 동의 구하기

- API 엔드포인트
- URL 단축
- URL 리디렉션

3. 상세 설계

- 데이터 모델
  - 해시함수
  - URL 단축키 상세 설계
  - URL 리디렉션 상세 설계

4. 마무리

문제 이해 및 설계 범위 확정

✔️URL 단축: 주어진 긴 URL을 훨씬 짧게 줄인다
✔️URL 리디렉션: 축약된 URL 로 HTTP 요청이 오면 원래 URL로 안내한다.
✔️높은 가용성과 규모확장성, 그리고 장애 감내가 요구된다

개략적 추정

- 쓰기 연산: 매일 1억 개의 단축 URL 생성
- 초당 쓰기 연산: 1억(100million)/24/3600=1160
- 읽기 연산: 읽기 연산과 쓰기 연산 비율은 10:1이라고 하자. 그 경우 읽기 연산은 초당 11,600회 발생한다(1160×10=11,600).
- URL 단축 서비스를 10년간 운영한다고 가정하면 1억(100million)×365×10=3650억(365billion) 개의 레코드를 보관해야 한다.
- 축약 전 URL의 평균 길이는 100이라고 하자.
- 따라서 10년 동안 필요한 저장 용량은 3650억(365billion)×100바이트=36.5TB이다.

개략적 설계안 제시 및 동의 구하기

API 엔드포인트

📌 URL 단축용 엔드포인트

- 단축 URL을 반환

POST /api/data/shorten
{ longId: longURLString }

📌 URL 리디렉션용 엔드포인트

- 원래 URL을 반환

GET /api/v1/shortUrl

개략적 설계안 제시 및 동의 구하기

API 리디렉션

301	302
- HTTP 요청의 처리 책임이 영구적으로 이전 - 브라우저는 이 응답을 캐싱함	- HTTP 요청의 처리 책임이 일시적으로 이전 - 언제나 단축 url 서버에 먼저 보내진 후 원래 url 로 리디렉션
- 서버 부하를 줄일 때	- 트래픽 분석이 중요할 때

구현하는 가장 간단한 방법

- 해시 테이블에 <단축URL, 원래 URL> 쌍을 저장
  - 원래 URL = hashtable.get(단축 URL)
  - 301/302를 응답 location 헤더에 원래 URL을 넣은 후 전송

개략적 설계안 제시 및 동의 구하기

URL 단축

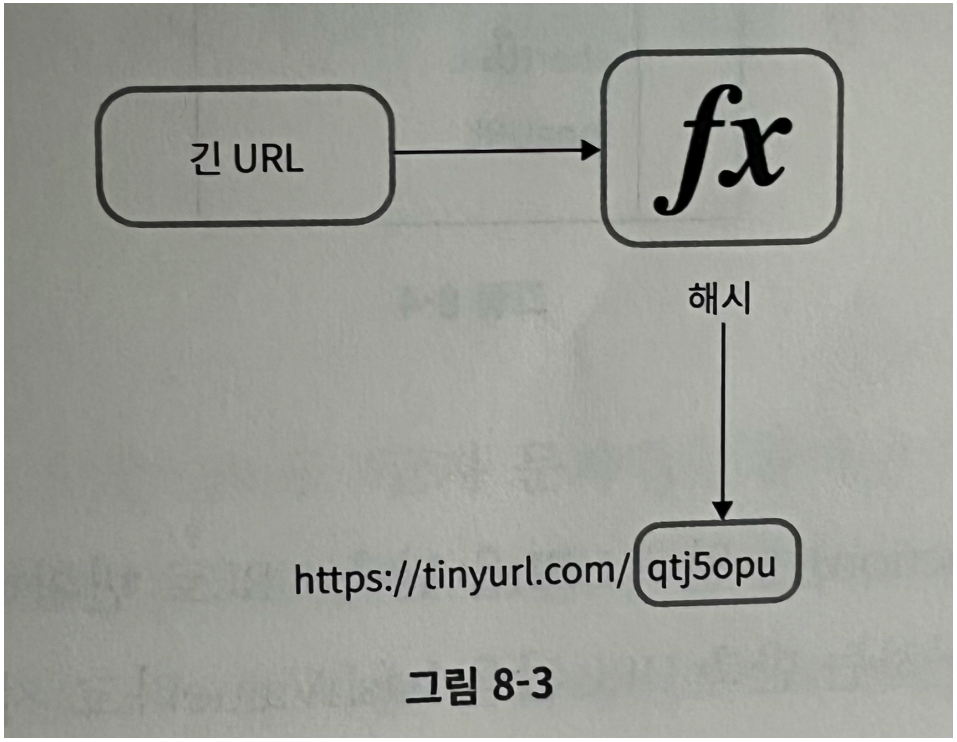

그림 8-3

- 입력으로 주어지는 긴 URL이 다른 값이면 해시 값도 달라야 한다

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

상세설계

데이터 모델

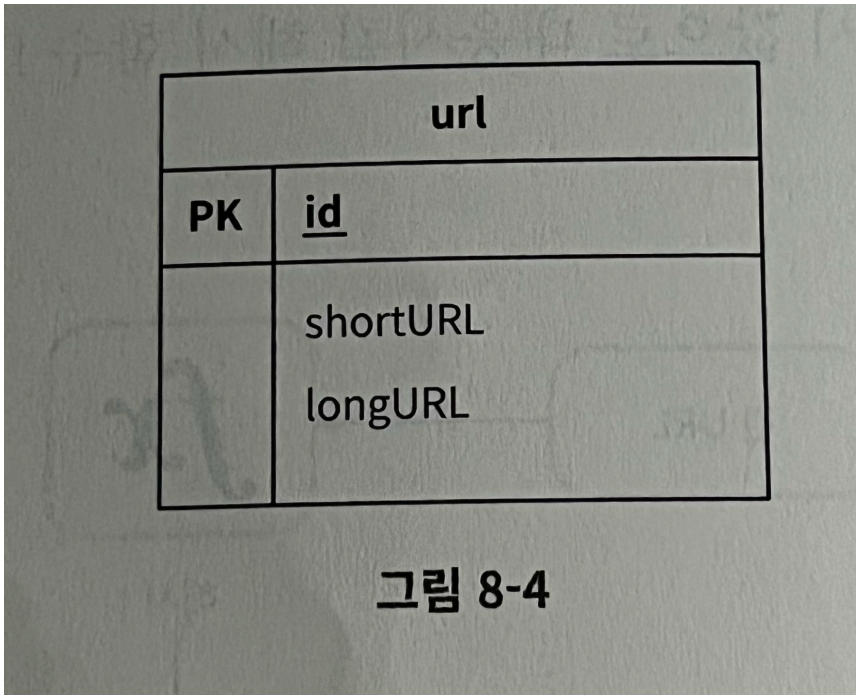

그림 8-4

상세설계

해시함수

단축 URL = hash value

n	URL 개수
1	$62^1 = 62$
2	$62^2 = 3,844$
3	$62^3 = 238,328$
4	$62^4 = 14,776,336$
5	$62^5 = 916,132,832$
6	$62^6 = 56,800,235,584$
7	$62^7 = 3,521,614,606,208 = \sim 3.5$조(trillion)
8	$62^8 = 218,340,105,584,896$

표 8-1

- hash value = 0-9, a-z, A-Z —> 62개

- n=7일때 3.5 조로 요구사항을 만족시키기 충분한 값

상세설계

해시함수

해시 후 충돌 해소

- 일반적으로 사용하는 해시함수는 길이가 너무 김
- 원하는 길이만큼 자른 후, 충돌하는지 확인
- 충돌이 해소될 때 까지 사전에 정한 문자열을 해시값에 덧붙임

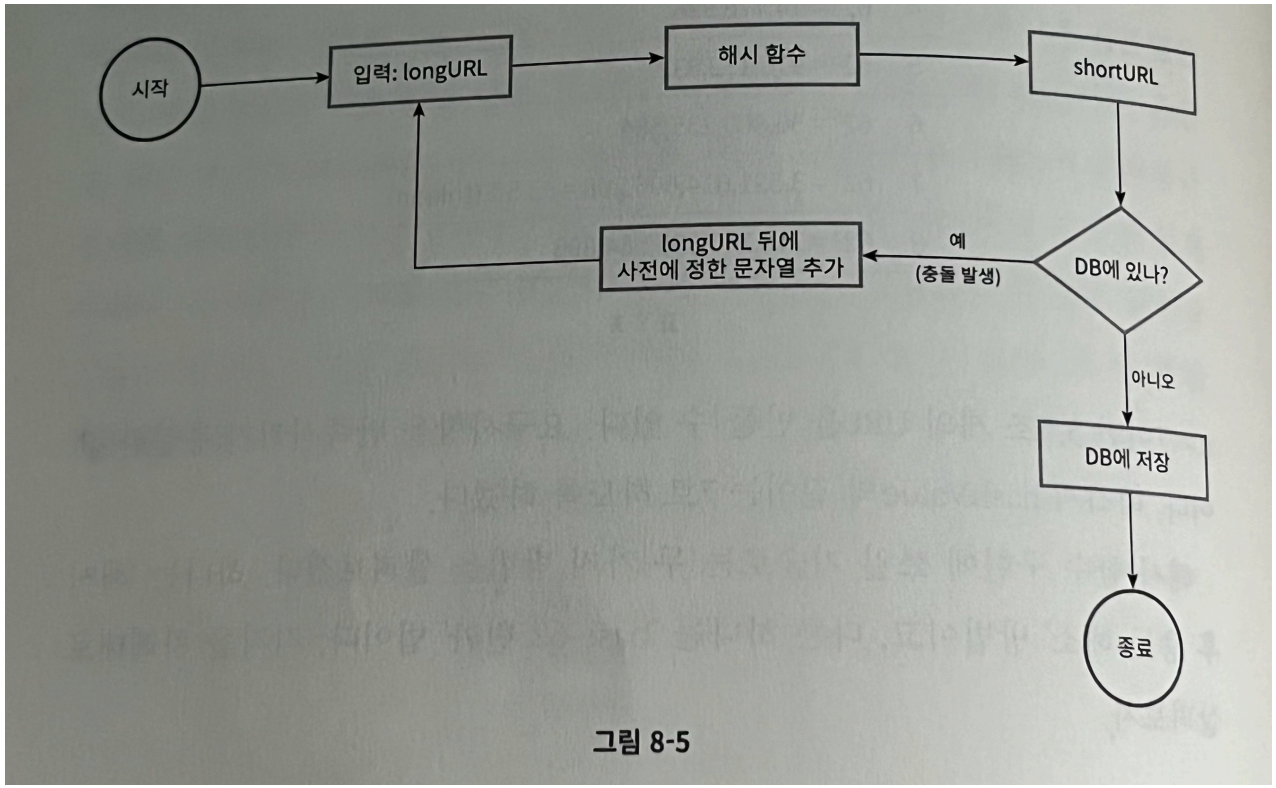

그림 8-5

상세설계

해시함수

base-62

- 10진수를 62진법으로 변환
- 10진수 -> DB에 넣기 위해 생성한 유일한 id

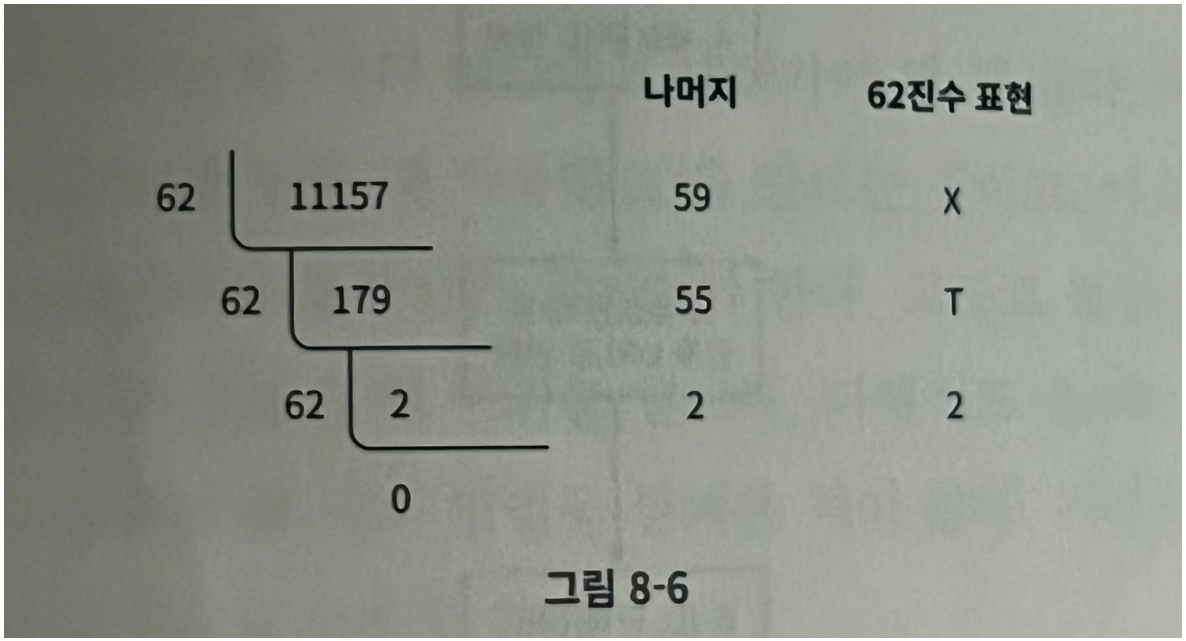

그림 8-6

상세설계

해시함수

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

표 8-3

상세설계

URL 단축기 상세 설계

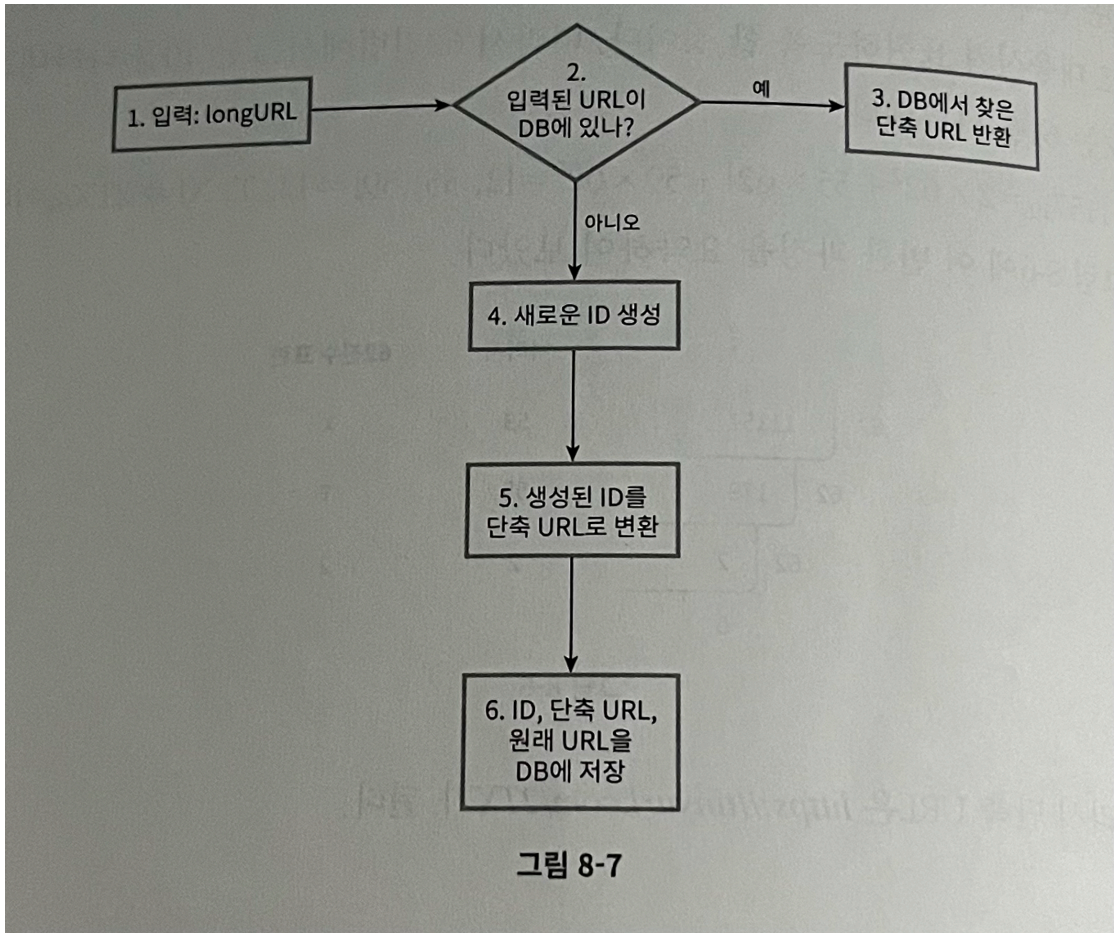

그림 8-7

1. 입력으로 긴 URL을 받는다.

2. 데이터베이스에 해당 URL이 있는지 검사한다.

3. 데이터베이스에 있다면 해당 URL에 대한 단축 URL을 만든 적이 있는 것이다.

   따라서 데이터베이스에서 해당 단축 URL을 가져와서 클라이언트에게 반환한다.

4. 데이터베이스에 없는 경우에는 해당 URL은 새로 접수된 것이므로 유일한 ID를 생성한다.

   이 ID는 데이터베이스의 기본 키로 사용된다.

5. 62진법 변환을 적용, ID를 단축 URL로 만든다.

상세설계

URL 리디렉션 상세 설

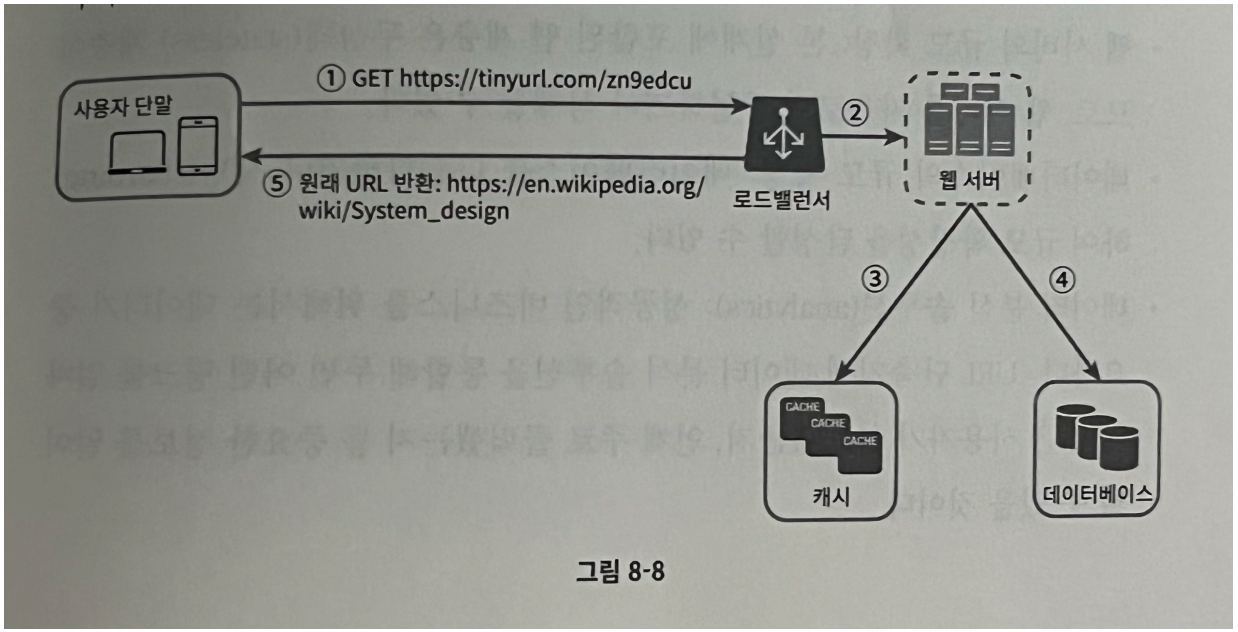

그림 8-8

1. 사용자가 단축 URL을 클릭한다.
2. 로드밸런서가 해당 클릭으로 발생한 요청을 웹 서버에 전달한다.
3. 단축 UIRL이 이미 캐시에 있는 경우에는 원래 URL을 바로 꺼내서 클라이언트에게 전달한다.
4. 캐시에 해당 단축 URL이 없는 경우에는 데이터베이스에서 꺼낸다.
   데이터 베이스에 없다면 아마 사용자가 잘못된 탄축 URL을 입력한 경우일 것이다.
5. 데이터베이스에서 꺼낸 URL을 캐시에 넣은 후 사용자에게 반환한다

마무리

처리율 제한 장치	처리율 제한 장치를 두면, IP 주소를 비롯한 필터링 규칙들을 이용해 요청을 걸러낼 수 있음
웹서버 규모 확장	웹 계층은 무상태(stateless) 계층이므로, 웹 서버를 자유로이 증설하거나 삭제할 수 있음
데이터베이스 규모 확장	데이터베이스를 다중화하거나 샤딩(sharding) 하여 규모 확장성을 달성
데이터 분석 솔루션	URL 단축기에 데이터 분석 솔루션을 통합해 두면 어떤 링크를 얼마나 많은 사용자가 클릭했는지, 언제 주로 클릭했는지 등 중요한 정보를 알아 낼 수 있음
가용성, 데이터 일관성	대규모 시스템이 성공적으로 운영되기 위해 반드시 갖추어야 할 속성