

[8장] URL 단축기 설계

가상 면접 사례로 배우는 대규모 시스템 설계 기초

이민석 / unchaptered

<https://inblog.ai/monthly-cs>
<https://github.com/monthly-cs/2024-03-system-design-interview-1>

문제 이해 및 설계 범위 확정


기능적 요구사항

- URL 단축
- URL 리디렉션
- 고가용성(High Availability), 고확장성(High Scalability), 장애 감내(Failure Tolerance)

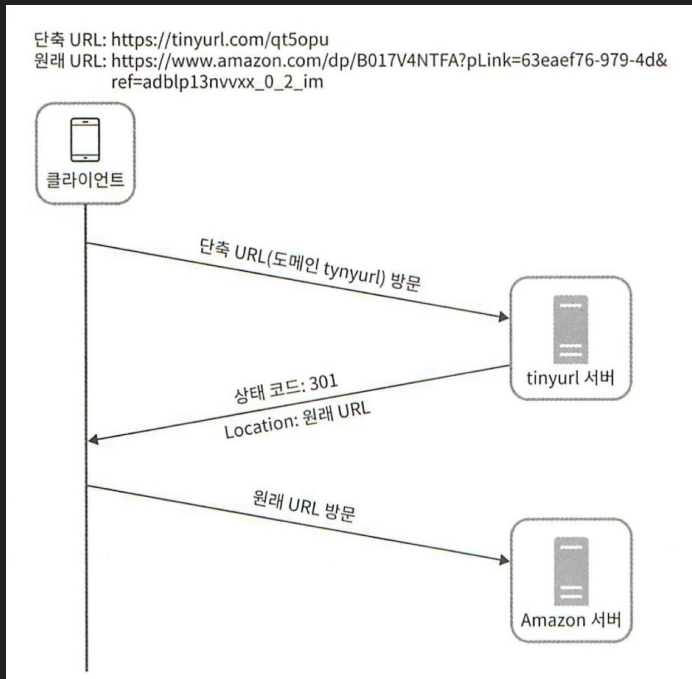
성능 요구사항

- 쓰기 연산 : 매일 1억개의 단축 URL 생성, 초당 1160
- 읽기 연산 : 쓰기:읽기=1:10일 경우 매일 1000만 개의 조회, 초당 11600
- 저장 공간 : 연간 365억 개의 레코드 저장 * URL 평균 길이 100은 100byte = 36.5 TiB

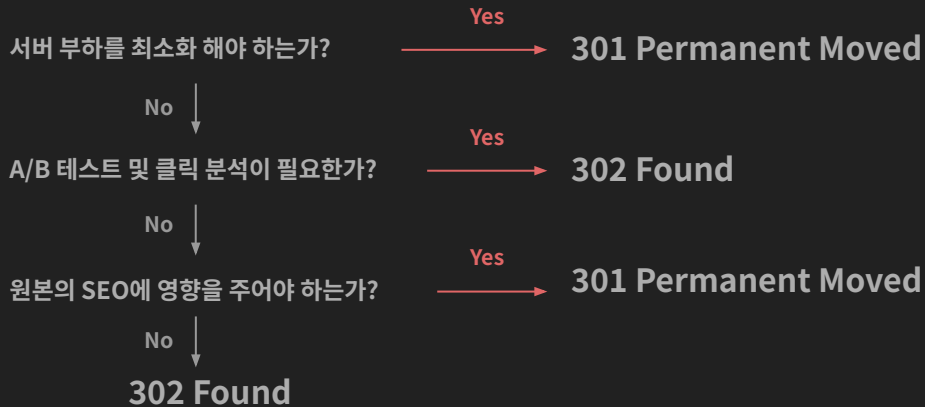
API 엔드포인트

	Request Body	Response
 /api/v1/data/shorten	{ "longUrl": "longUrlString" }	shortenUrl
/api/v1/shortUrl		lognUrlString

URL 리디렉션



<https://donghyeon.dev/%EC%9D%B8%ED%94%84%EB%9D%BC/2022/04/06/URL-%EB%8B%A8%EC%B6%95%EA%B8%B0-%EC%84%A4%EA%B3%84/>



301 vs 302 Redirect: What are the Differences Between Them?

While a 301 redirect indicates that a page has permanently moved to a new location, a 302 redirect means that the move is only temporary. Keep reading to learn more.



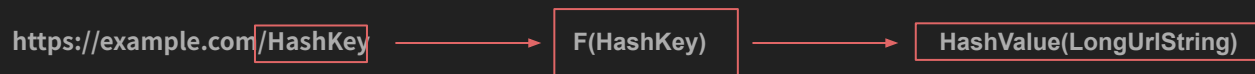
Rock Content Writer
Content writer

✓ Human crafted content

Jan 5, 22 | 6 min read

<https://rockcontent.com/blog/301-vs-302-redirect/>

URL 단축



데이터 모델

url	
PK	<u>id</u>
	shortURL longURL

함수 값 길이

[0-9a-zA-Z]의 경우의 수 = $10 + 26 + 26 = 62$ 개

성능 요구사항에 따르면 10년에 3650억개 필요하고, 62^7 은 3.5조에 해당한다.
따라서 **7자리수의 값(Value)**으로 대다수 요구사항을 충족할 수 있다.

기능적 요구사항

- URL 단축
- URL 리디렉션
- 고가용성(High Availability), 고확장성(High Scalability), 장애 감내(Failure Tolerance)

성능 요구사항

- 쓰기 연산 : 매일 1억개의 단축 URL 생성, **초당 1160**
- 읽기 연산 : 쓰기:읽기=1:10일 경우 매일 1000만 개의 조회, **초당 11600**
- 저장 공간 : **연간 365억 개의 레코드 저장** * URL 평균 길이 100은 100byte = 36.5 TiB

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	$62^7 = 3,521,614,606,208$
8	$62^8 = 218,340,105,584,896$


알고리즘 1안 - 해시 후 충돌 해소

해시 함수	해시 결과 (16진수)
CRC32	5cb54054
MD5	5a62509a84df9ee03fe1230b9dfb84e
SHA-1	0eeae7916c06853901d9ccbefbfcaf4de57ed85b

<https://donghyeon.dev/%EC%9D%B8%ED%94%84%EB%9D%BC/2022/04/06/URL-%EB%8B%A8%EC%B6%95%EA%B8%B0-%EC%84%A4%EA%B3%84/>


Bloom Filters | Algorithms You Should Know #2 | Real-world Examples

조회수 15만회 · 1년 전

 ByteByteGo

Animation tools: Illustrator and After Effects ABOUT US: Covering topics and trends in large-scale system design, from the authors ...

자막

 Bloom Filter | What Are Bloom Filters | Hash Functions

장면 3개

https://youtu.be/V3pzngeLqw?si=hctGrMGh8fN_4gXg

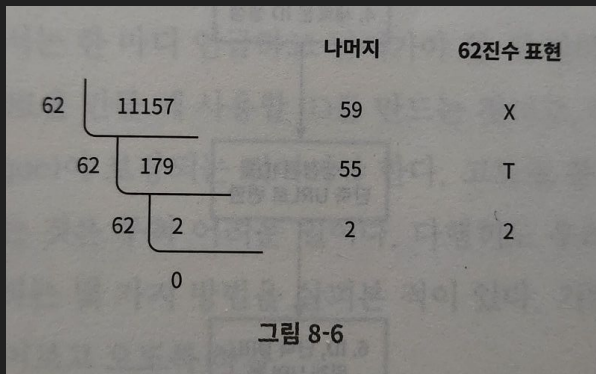
기본적인 해쉬 알고리즘 사용

CRC32 알고리즘도 8자리에 해당
→ 요구사항인 7자리보다 길다.
→ 이를 7자리로 자르고 충돌 해결을 위한
프로세스를 추가할 수 있으나 **오버헤드가 큼**

블룸 필터(Bloom Filter) 사용

확률에 근거하여 어떤 대상을 기록 및 탐색
→ 오차가 발생할 수 있으나, 그만큼 효율적
→ 블룸 필터의 길이를 제어하여, 오차
범위를 조절 가능

알고리즘 2안 - base-62 변환



62진법을 사용하여 암호화

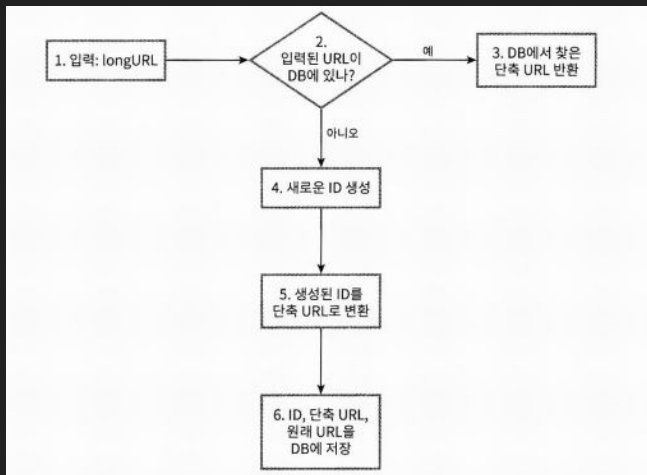
1부터 차례대로 숫자가 커지면서 3.5조까지 증가

알고리즘 선택 (해시 후 충돌 해소)

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	ID 유일성이 보장된 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

<https://donghyeon.dev/%EC%9D%B8%ED%94%84%EB%9D%BC/2022/04/06/URL-%EB%8B%A8%EC%B6%95%EA%B8%B0-%EC%84%A4%EA%B3%84/>

URL 단축키 상세 설계



입력된 URL이 https://~/System_design 이다.

이 URL에 대해 ID 생성기가 **2009215674938**을 반환

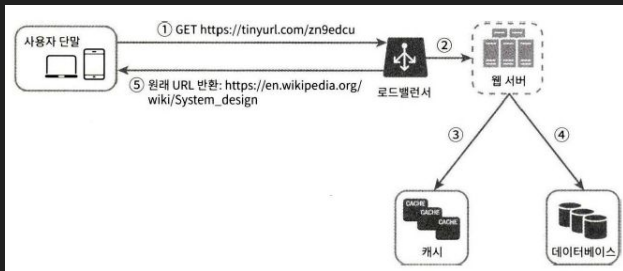
이 ID를 62진수로 표기하여 **zn9edcu**를 반환

아래와 같이 새로운 데이터베이스 레코드 생성

ID	shortURL	longURL
2009215674938	zn9edcu	https://en.wikipedia.org/wiki/Systems_design

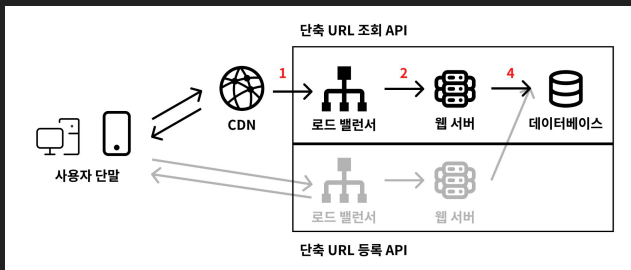
<https://velog.io/@bw1611/URL-%EB%8B%A8%EC%B6%95%ED%82%A4-%EC%84%A4%EA%B3%84>

URL 리디렉션 상세 설계



도서에서 추천한 구조

캐시를 읽기 전략으로 사용하여, 성능을 향상



개인적인 생각

CDN을 통해서 단축 URL을 캐싱하는 편이 효율적이지 않나?

→ 캐시를 쓸 때보다 **비즈니스 로직이 간결해짐**

→ 캐시를 쓸 때보다 운영, 재해복구 측면의 복잡성이 낮아짐

→ CDN이 로드밸런서 앞에 있어서, 로드 밸런서/웹 서비스 부하 감소

→ CDN 방화벽 설정을 통해서, 로드 밸런서/웹 서비스 보호 가능

→ CDN 캐시 정책 등을 수정함으로써, 최적화에 개발자 도움이 필요 없음

추가로 논의할 것들에 대하여

- 처리율 제한 장치(Rate Limiter)
- 웹 서버의 규모 확장
- 데이터베이스 규모 확장
- 데이터 분석 솔루션
- 가용성, 데이터 일관성, 안정성

감사합니다.