

8장 : URL 단축기 설계

| | |
|--------|--------------|
| 👤 난이도 | ★★ |
| 📅 학습날짜 | @2026년 1월 2일 |

1. URL 단축의 정의

단축 URL 사용 이유

2. 설계를 위한 개략적 추정

기본 API 엔드포인트

3. URL 단축 방법

1. 해시 함수

2. base-n(62) 변환

상세 설계

1. URL 단축의 정의

단축 URL 서비스는 긴 원본 URL (Original URL)을 짧고 유일한 식별자 (Short Key)로 맵핑하여 사용자에게 제공하는 서비스이다.

기본적으로 다음과 같은 **기능**을 제공한다.

- URL 단축 : 원본 URL을 짧은 URL로 훨씬 짧게 줄인다
- URL 리디렉션 (redirection) : 축약된 URL로 HTTP 요청이 오면 원래 URL로 안내한다
- 높은 가용성과 규모 확장성, 그리고 장애 감내가 요구된다.

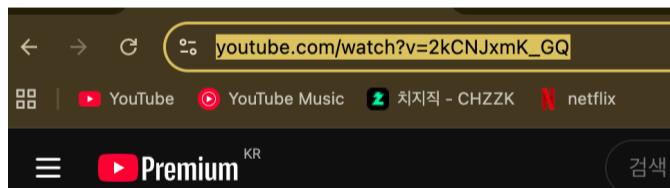
// 원본 URL

<https://www.systeminterview.com/q=chatsystem&c=loggedin&v=v3&l=long>

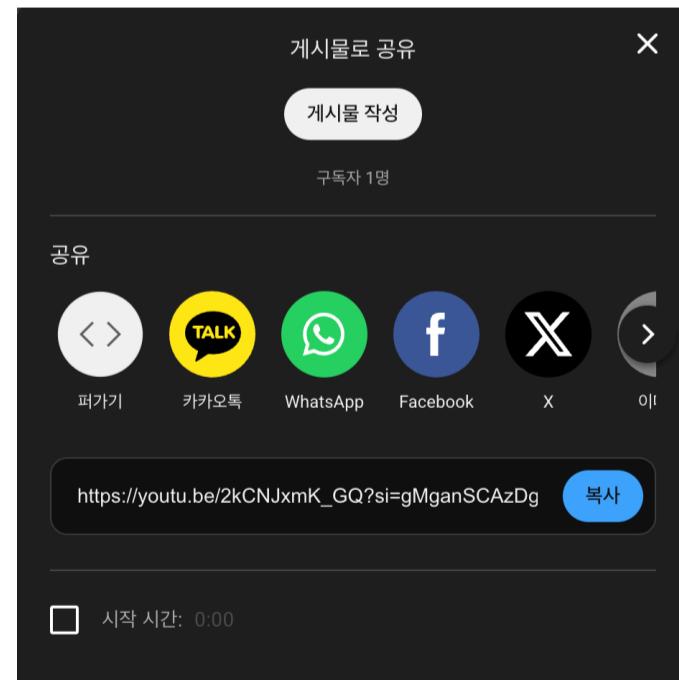
// 단축 URL

<https://tinyUrl.com/y7ke-ocwj>

유튜브의 원본 URL



유튜브의 단축 URL



그렇다면 이러한 단축 URL을 **사용하는 시점**은 언제이며, **왜 사용해야 할까?**



단축 URL 사용 이유

1. URL 주소 은닉

- 내부 식별자 보호:

site.com/orders/10293과 같이 URL에 내부 로직의 ID를 포함하는 경우가 있다.

이는, 공격자가 숫자를 바꿔가며 데이터를 탐색하는 ID Enumeration (ID 열거 공격)의 단서가 된다. 단축 URL은 이에 대처할 수 있는 좋은 방법이다.

2. 데이터 분석

- 관심사 분리:

URL 접속 로그 수집 로직을 메인 애플리케이션 서버와 분리하여, 메인 서버의 부하를 줄일 수 있습니다.

3. 사용자 경험 개선 (+사용성)

- 유연한 URL 제어:

이미 배포된 링크라 하더라도, URL 단축 서버 단에서 목적지 주소를 변경하여, 이벤트 종료 후 다른 페이지로 유도하거나, 서버 장애 시 긴급하게 점검 페이지로 리다이렉션하는 등의 대응이 가능하다.

- 인코딩 이슈 방지:

한글이나 특수문자가 포함된 URL은 브라우저 인코딩 시 `%EB%B8%94%EB%A1%9C...` 와 같이 매우 길어지며, 복사/붙여넣기 과정에서 링크가 깨질 위험이 크다. 단축 URL을 사용하여 영문과 숫자의 조합으로 단순화할 수 있다.

- (Etc) 광고 홍보 시 간단한 URL 제공

인쇄물 광고에 링크를 두거나, SMS(80Byte 제한), 트위터 등 글자 수 제한이 있는 플랫폼에서 메세지 비용을 절감할 수 있다.

2. 설계를 위한 개략적 추정

단축 URL을 설계하기 전에, 시스템의 개략적인 추정을 진행해야 한다.

[기본 스펙]

- 트래픽 규모 : 매일 1억 개의 단축 URL을 생성해 낼 수 있어야 한다.
- 단축 URL의 길이 : 짧으면 짧을수록 좋다
- 단축 URL의 문자 제한 : 숫자 (0-9)와 영문자 (a-z, A-Z)만 사용 가능
- 단축 URL 삭제, 갱신 여부 : 불가능

[단축 URL 서버 부하]

- 쓰기 연산 : 매일 1억 개의 단축 URL 생성해야 함
 - 초당 쓰기 연산 : $1\text{억} / 24 / 60 / 60 = 1160$
- 읽기 연산 : 읽기와 쓰기 연산의 비율을 10:1로 가정하면, 읽기 연산은 초당 11,600회 발생한다

[단축 URL DB 용량]

- URL 단축 서비스를 10년간 운영한다고 가정하면, $1\text{억} * 365 * 10 = 3650\text{억 개의 레코드를 보관}$
- 축약 전 URL의 평균 길이가 100이면, $3650\text{억} * 10\text{Byte} = 36.5\text{TB 이상의 저장 용량이 필요}$

기본 API 엔드포인트

1. URL 단축용 엔드포인트: POST [/api/v1/data/shorten](http://api/v1/data/shorten)

- 목적 : 새 단축 URL을 생성하기
- 사용 용도 : 공유하기 or 알림톡 발송 등 원본 URL을 토대로 단축 URL을 생성하고자 하는 경우

2. URL 리디렉션용 엔드포인트: GET [/api/v1/shortUrl](#)

- 목적 : HTTP 리디렉션 목적지가 될 원래 URL
- 사용 용도 : 단축 URL로 접속을 시도하는 유저에게 원본 URL로 리디렉션해주는 기능
- 301 혹은 302 응답을 보내주면, 클라이언트에서 리디렉션한다.



301 / 302 리디렉션의 차이

- **301 (Moved Permanently):**

해당 URL에 대한 HTTP 요청의 처리 책임이 영구적으로 Location 헤더에 반환된 URL로 이전되었다는 응답이다.

영구적으로 이전되었으므로, 브라우저는 이 응답을 캐시한다.

→ 추후 같은 단축 URL에 요청을 보낼 필요가 있을 때 브라우저는 캐시된 원래 URL로 요청을 보내게 된다.

◦ 단, **캐시가 되면 트래픽이 발생하지 않아**, 트래픽 분석에 용이하지 않을 수 있어서 상황에 따라 적절히 사용해야 함

- **302 (Found):**

주어진 URL로의 요청이 '일시적으로' Location 헤더가 지정하는 URL에 의해 처리되어야 한다는 응답이다.

일시적으로 이전되었으므로, 브라우저는 이 응답을 캐시하지 않는다.

→ 클라이언트는 언제나 단축 URL 서버를 거쳐서 원본 URL을 받아와야 한다

+ 하나의 단축기 서비스 안에서, 301을 쓸 수도 있고 302를 쓸 수도 있고 복합으로 사용해도 된다.

3. URL 단축 방법

원본 URL을 단축 URL로 변환하기 위해서는 해시 함수를 사용하는 방식이 가장 대중적이다.

이때, 해시 함수는 다음 특징을 지녀야 한다.

1. 입력으로 주어진 URL이 다르면 항상 다른 해시 값이 도출되어야 한다.

→ 해시 충돌 **✗**

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

이때, 개략적 추정 단계에서 약 3650억 개의 레코드를 보관해야 한다고 언급하였다.

해시 값이 [0-9, a-z, A-Z]로만 구성되어 있다면, 총 62개의 문자를 사용할 수 있다.

따라서, $62^n \geq 3650\text{억}$ 이 되는 최소값은 $n = 7$ 이다.

→ **해시 값의 길이는 7**로 설정하면 시스템에 알맞는 설계이다.

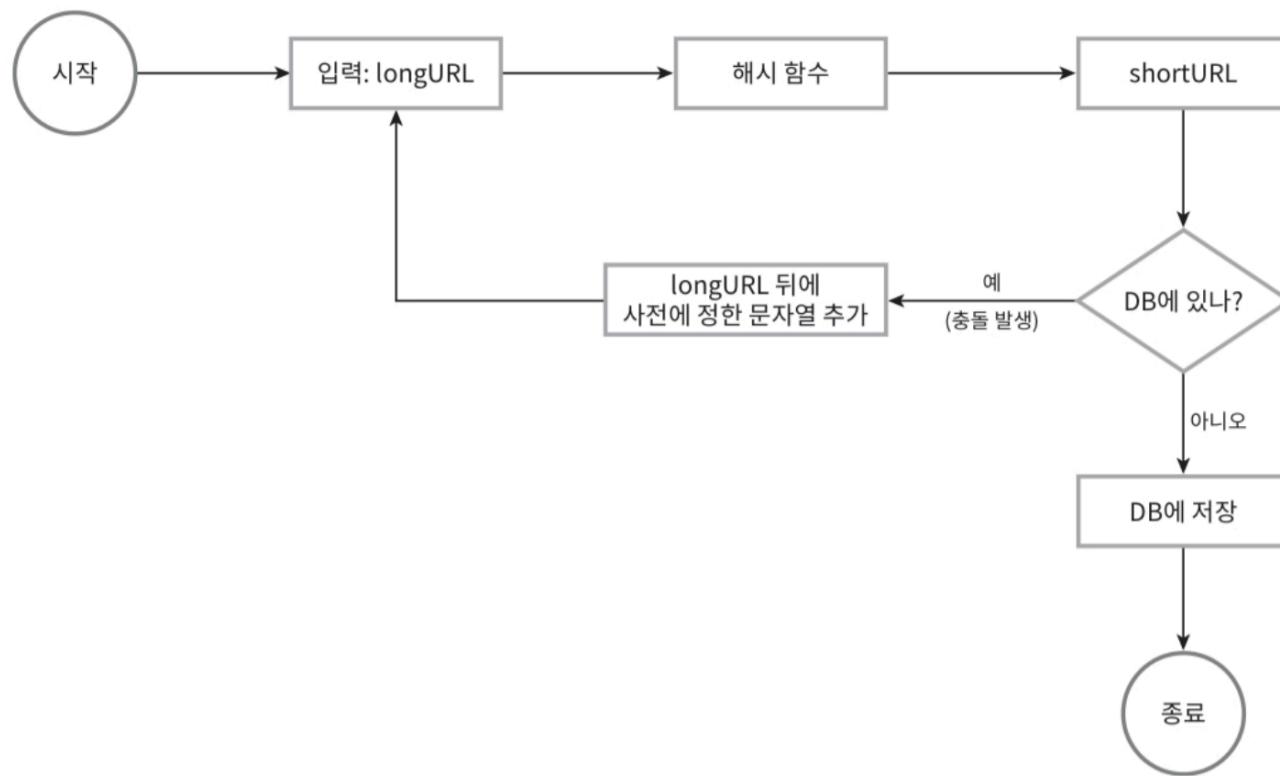
1. 해시 함수

잘 알려진 해시 함수로는 아래 해시 함수들을 사용하면 된다.

- CRC32, MD5, SHA-1

다만, 이러한 해시함수를 사용하더라도 길이 7이하로 설정하지는 못한다.

해결방법 : 처음 7개 글자만 사용하고, 해시 결과가 충돌이 발생할 경우 충돌이 해소될 때까지 사전에 정한 문자열을 해시값에 덧붙이는 방법을 사용한다.



단, 단축 URL을 생성할 때 한 번 이상 DB 질의를 해야 하므로 오버헤드 ↗

- 데이터베이스 대신 블룸 필터를 사용하면 성능을 높일 수는 있다.

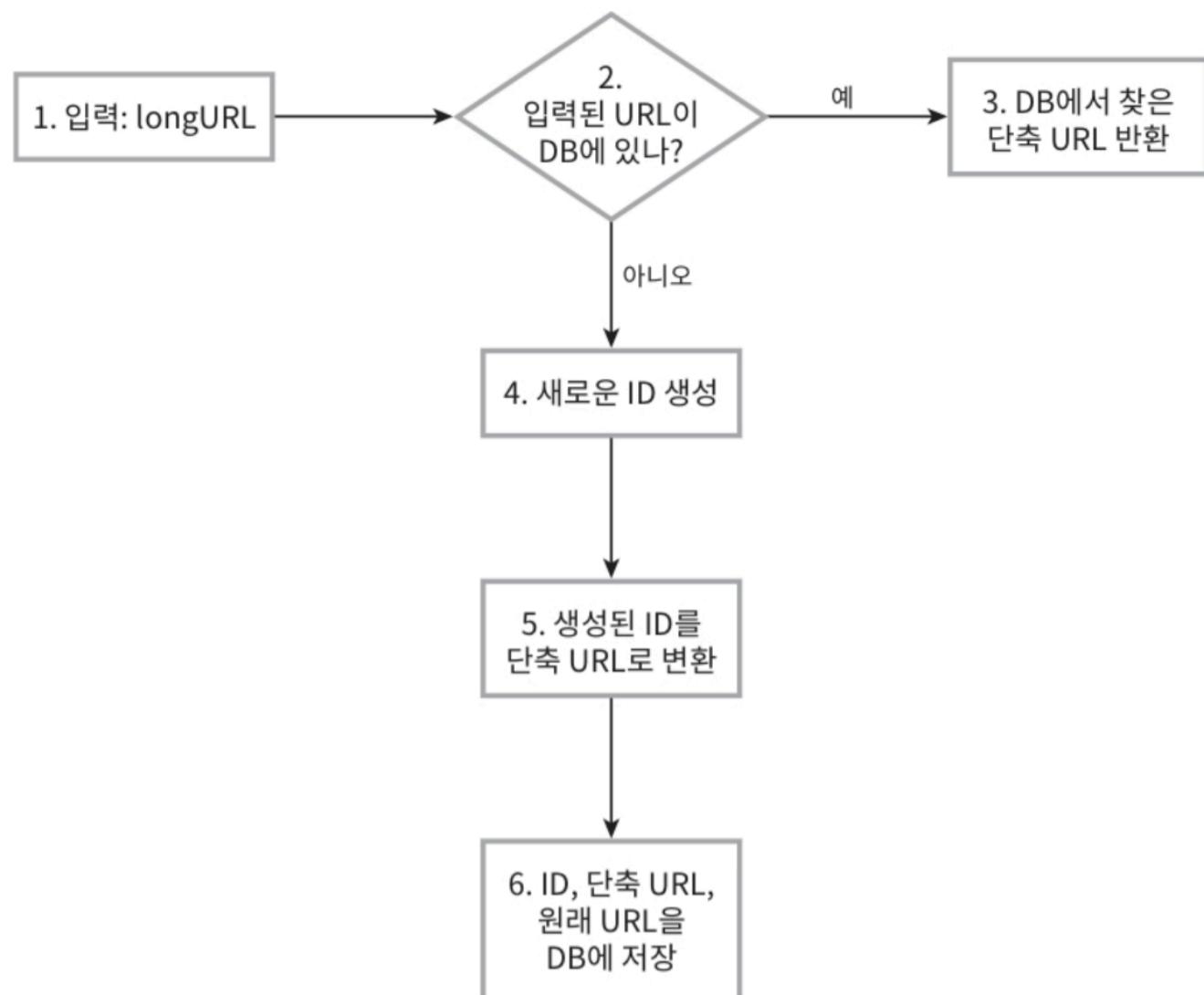
2. base-n(62) 변환

해시값의 구성 요소가 총 62개의 문자로 이루어진다 하였으니, 62진법을 사용하여 변환하면 7글자 제한의 단축 URL을 생성할 수 있다.

이때, 원본 URL에 DB ID를 생성한 이후 이를 토대로 62진법 변환을 하는 방법이다.

11157 (10) → 2TX (62)

- 원본 ID 값이 커지면, 같이 길어진다
- 분산 환경에서도 유일성을 보장할 수 있는 ID 생성기가 필요하다
- 유일성만 보장된다면 충돌이 절대 발생하지 않는다.
- ID가 1씩 증가하는 구조라면, 다음에 사용할 단축 URL이 무엇인지 쉽게 알아낼 수 있어서 보안상 문제가 될 소지가 있다.

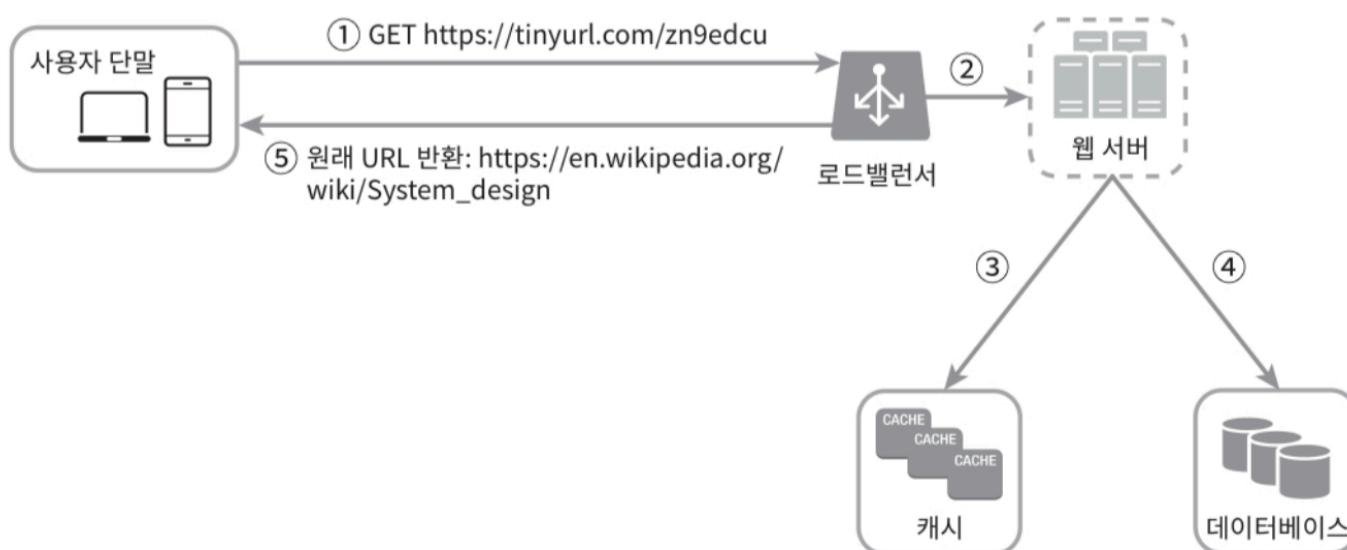


예시

1. 입력된 URL : https://en.wikipedia.org/wiki/Systems_design이다.
 2. 해당 URL에 대해 ID생성시가 반환한 ID는 200921567938이다
 3. 62진수로 반환하면 zn9edcu이다.
 4. 서버에서 다음 레코드를 만든다.
- | ID | shortURL | longURL |
|--------------|----------|---|
| 200921567938 | zn9edcu | https://en.wikipedia.org/wiki/Systems_design |

상세 설계

쓰기보다 읽기를 더 자주하는 시스템이라면, 로드밸런서 / 캐시를 활용하여 부하를 줄일 수 있다.



캐시가 붙은 설계