**DESIGN A URL SHORTENER**

### Understanding Problem and Establishing Design Scope:

Assuming the generated tinyurl will start with https://tinyurl.com/

1. What type of characters should we expect from the tinyUrl generator?

-> [0-9,a-z,A-Z]

1. What would be the max length of the unique ID generated for the tinyUrl?

-> As short as possible

1. What would be the traffic for this system?

-> 100 million URLs are generated per day.

1. Can shortened urls be deleted or updated?

-> For simplicity, let us assume it can not be deleted or updated.

### Back of the Envelope Estimation:

100 million = 100 X 10^6 = 10^8 requests per day

= 10^8/ 60\*60\*24

= 10^8/86400

= 10^8/100000 = 10^8/10^5

= 10^3 requests per second

**Request/sec: 10^3 requests**

**Read to Write ration Requests: 10:1**

**Read Requests per second : 10^3(approx)**

**Write Requests per second : 10(approx)**

So total characters we can use is 10+26+26 = 62

Assuming that the system will run for 10 years

So, 10^9 \* 365 = 10^11(approx) urls would be generated in 10 years. So to support this the tinyUrl should **generate a length of minimum 7 characters**.  
As 62^7 > 10^11

Assuming the average length of the URL is 100.

So data over 10 years is 10^11\*10\* 100 = 10^14 = **100 TB data storage is required**

### High Level Design Propositions and approaches:

**API endpoints:**

1. Url Shortening: To create a new short URL, a client sends a POST request, which contains one parameter: the original URL

**Request:**

POST api/v1/data/shorten

{

“longUrl”: “https://mysamplewebsite.com/my\_profile”

}

**Response:**  
{

“shortUrl”: “https://tinyurl.com/abc123A”  
}

1. URL redirecting: To redirect a short URL to the corresponding URL, a client sends a GET request. The API looks like this:

**Request:**  
GET /api/v1/shortURL?domain=tinyurl.com?uniqueId=abc123A

**Response:**

Status code: 301

Response header:  
location: <https://mysamplewebsite.com/my_profile>

301 redirect: A 301 redirect shows that the requested URL is “permanently” moved to the long URL. Since it is permanently redirected, the browser caches the response and subsequent requests for the same URL will not be sent to the URL shortening service. Instead, requests are redirected to the long URL server directly.

302 redirect: A 302 redirect means that the URL is “temporarily” moved to a long URL, meaning that subsequent requests for the same URL will be sent to the URL shortening service first. Then , they are redirected to the long URL server.  
  
**If the priority is to reduce server load,** using **301 redirect** makes sense as **only the first request of the same URL is sent to URL shortening servers**.   
However, **if analytics is more important, 302 redirect** is a better choice **as it can track click rate and source of the click more easily**.

**Intuitive Approach:**The most intuitive approach is to implement URL shortening using HashMap tables. Assuming the hash table stores <shortURL, longURL> pairs, URL redirecting can be implemented by the following:  
- Get longURL: longURL = hashTable.get(shortURL)

- Once you get the longURL, perform a URL redirect.

To support the URL shortening use case, we must find a hash function fx that maps a long URL to hashValue.

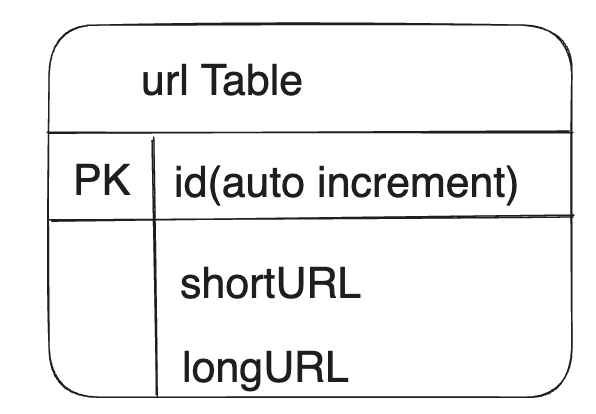
The hash function must satisfy the following requirements:

- Each longURL must be hashed to one hashValue.

- Each hashValue can be mapped back to longURL.

### Design Deep Dive:

**Data Model:**Storing in a hash table is not feasible for the real-world as memory resources are limited and expensive. A better option is to store <shortURL, longURL> mapping in a relational database.

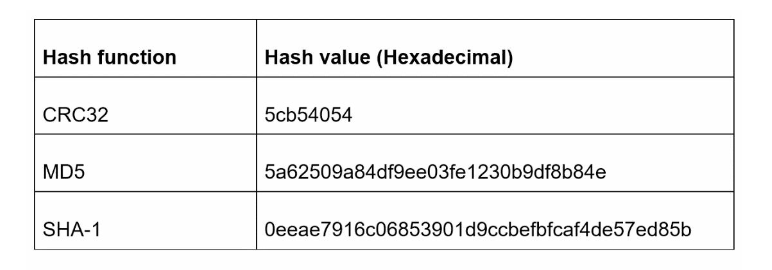
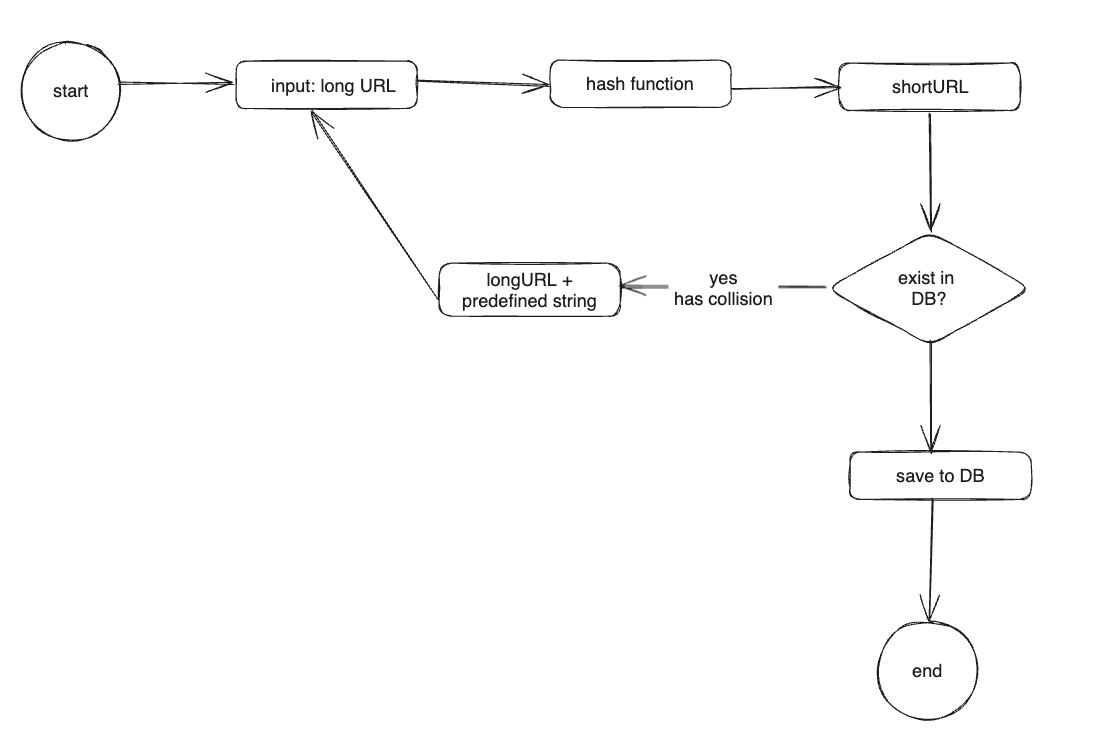


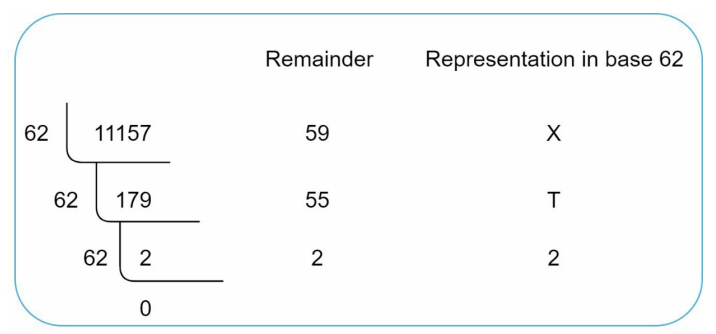
**Hash function**

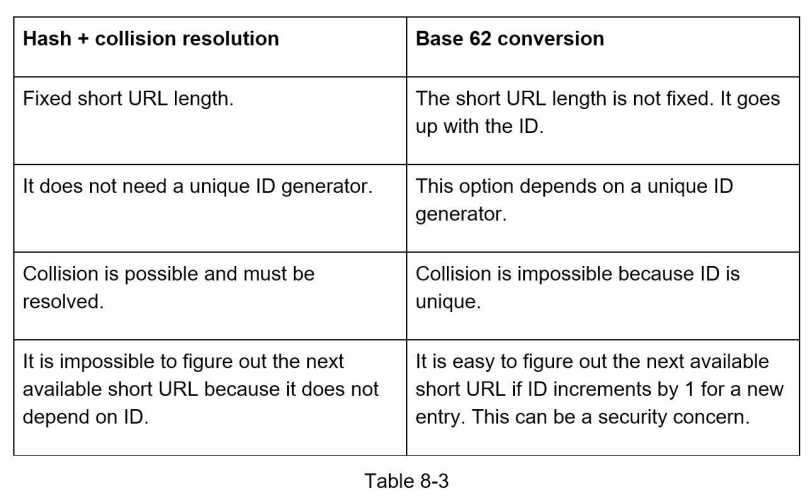
Hash function is used to hash a long URL to a short URL, also known as hashValue.

**Hash value length**

The hash value consists of characters from [0-9, a-z, A-Z] consisting of 62 possible characters.As calculated in Back of the envelope estimation length of the hashValue is 7.  
  
Will explore two types of hash function for URL shortener(to convert a long URL to a 7 character string):  
1. **Hash + collision resolution**

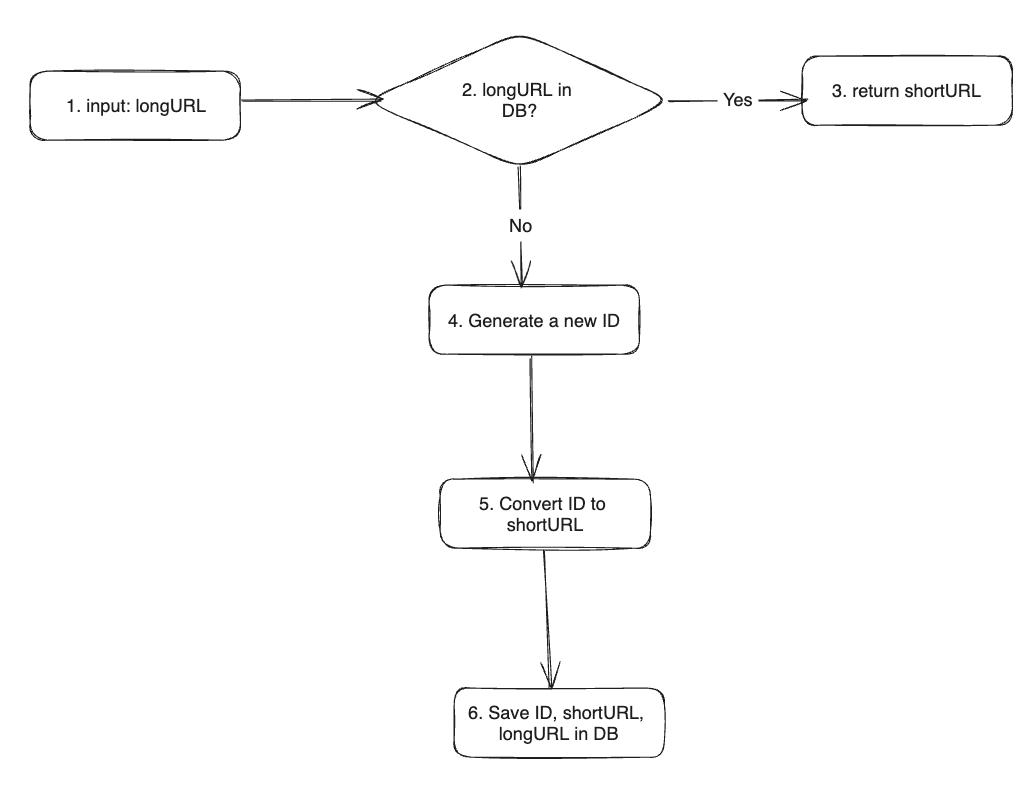
A straightforward solution is to use well known hash functions like CRC32, MD5 or SHA-1.  
As shown in the table even the shortest value CRC32 is too large(>7). We would need to make it shorter.  
To fix this we can take the first 7 characters. But this would result in hash collisions. So we can recursively append a new predefined string until no more collisions are discovered.  
This method will remove collisions but it is expensive to query the database to check if a shortURL exists for every request. A technique called bloom filter can improve performance. A bloom filter is a space efficient probabilistic technique to test if an element is a member of a set.  
2. **Base 62 conversion**

Base conversion helps to convert the same number between its different number representation systems. Base 62 conversion is used as there are 62 possible characters for hash value.

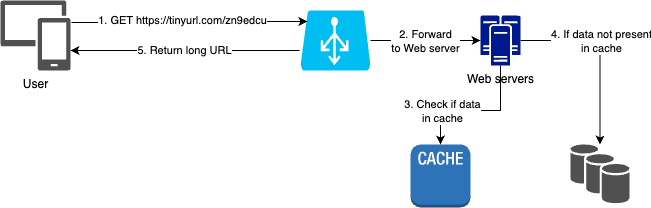


**URL shortening Deep Dive**

Using Base 62 conversion



**URL redirecting deep dive**

****