Ron Ben Shimol

Answers:

1. We can enhance the backend to have faster performance for frequently accessed short URLs by using cache in the server side. For example, saving the most frequently used short URLs. In addition, we can also use cookies in the client side to reduce repeating requests to the server for each client.

2. To enhance the implementation to have one “short URL” redirect to a few “long URLs”, based on redirect percentage, we can add an additional column to the DB named “percentage” in the “urlLinks” table, as well as adding few rows to the table with the same “short URL” and different “long URLs”.
   When the client makes requests with this “short URL” we will get all the relative rows from the DB and flip a biased coin according to the percentage.

3. Below is an example for proper sql query (lastTimeStr is a datetime of the beginning of the last hour):
   “SELECT linkId, count(id) AS Amount FROM urlRequests WHERE linkId IS NOT NULL AND date BETWEEN '"+lastTimeStr+"' AND datetime('now', 'localtime') GROUP BY linkId ORDER BY Amount DESC LIMIT 5;”

4. We can prevent an attack, in which a malicious client fills the database with millions of URL redirections by using cache in the server side, when a client makes a request for URL redirection, we will save a new entry (for a short duration), containing the client’s IP, counter and the request time in the cache. When the client initiates additional requests in short period of time, the counter will be incremented. Hence we can check if the counter for the current IP is higher then some predefined limit, and block requests from this IP address.

5. After the user presses the “Create short URL” button, the client side makes AJAX request (http GET) to the server side api, to a method called resolveShortUrl. the client sends as a request argument the string from the text input (using encodeURIComponent to encode the string before sending it). the server side API gets the request and checks if the longUrl already exists in the DB.
   Iff it exists in the DB, the server simply sends the relative shortUrl retrieved from the DB back to the client. if the longUrl doesn’t exist in the DB, the server will generate a short random string, that doesn’t exist in the DB (as shortURL).
   A new row will be added to the DB with the values of the longUrl and shortUrl. after adding the row to the DB, the server will send the shortUrl to the client side. When the client receives the short URL code, it will concate the domain URL to the shortUrl and display it to the user.
   Hooray! :)