Creating your own Tinyurl style uid

Asked 16 years, 2 months ago Modified 10 years, 9 months ago Viewed 13k times

17

I'm writing a small article on humanly readable alternatives to Guids/UIDs, for example those used on TinyURL for the url hashes (which are often printed in magazines, so need to be short).

The simple uid I'm generating is - 6 characters: either a lowercase letter (a-z) or 0-9.

"According to my calculations captain", that's 6 mutually exclusive events, although calculating the probability of a clash gets a little harder than P(A or B) = P(A) + P(B), as obviously it includes numbers and from the code below, you can see it works out whether to use a number or letter using 50/50.

I'm interested in the clash rate and if the code below is a realistic simulation of anticipated clash rate you'd get from generating a hash. On average I get 40-50 clashes per million, however bare in mind the uid wouldn't be generated a million times at once, but probably only around 10-1000 times a minute.

What is the probability of a clash each time, and can anyone suggest a better way of doing it?

```
static Random _random = new Random();

public static void main()
{
    // Size of the key, 6
    HashSet<string> set = new HashSet<string>();
    int clashes = 0;
    for (int n=0;n < 1000000;n++)
    {
        StringBuilder builder = new StringBuilder();

        for (int i =0;i < 7;i++)
        {
            if (_random.NextDouble() > 0.5)
            {
                builder.Append((char)_random.Next(97,123));
            }
            else
            {
                builder.Append(_random.Next(0,9).ToString());
            }
        }

        if (set.Contains(builder.ToString()))
        {
```

```
            clashes++;
            Console.WriteLine("clash: (" +n+ ")" +builder.ToString());
        }

        set.Add(builder.ToString());
        _random.Next();
        //Console.Write(builder.ToString());
    }

    Console.WriteLine("Clashes: " +clashes);
    Console.ReadLine();
}
```

UPDATE: Here's the resulting article from this question

I really asked two questions here so I was cheating. The answer I was after was rcar's, however Sklivvz's is also the answer to the 2nd part (an alternative). Is it possible to make a custom unique id generator in a database, or would it be client side (which would be 2 possible reads first)?

The general idea I was after was using Ids in databases or other stores that can be used by phone or printed material, not a giant 16 byte guid.

UPDATE 2: I put the formula for two mutually exclusive events above instead of 2 independent ones (as getting an 'a' the first time doesn't mean you can't get an 'a' the second time). Should've been P(A and B) = P(A) x P(B)

c# algorithm probability

Share
Improve this question
Follow

edited Aug 14, 2012 at 17:10

asked Oct 10, 2008 at 10:11

Chris S
65.4k ● 53 ● 224 ● 240

8 Answers

Sorted by: Highest score (default)

31

Why do you want to use a random function? I always assumed that tinyurl used a base 62 (0-9A-Za-z) representation of a sequential Id. No clashes and the urls are always as short as possible.

You would have a DB table like

```
Id  URL
 1  http://google.com
 2  ...
... ...
```

```
156 ...
... ...
```

and the corresponding URLs would be:

```
http://example.com/1
http://example.com/2
...
http://example.com/2W
...
```

Share

Improve this answer

Follow

edited Mar 20, 2014 at 8:18

answered Oct 10, 2008 at 10:20

Sklivvz

31.1k ● 24 ● 118 ● 174

I think the point is this just simulates it but is not the actual hashing algorithm. Its so he can determine metrics, like the clash rate. – mattlant Oct 10, 2008 at 10:24

1 I hadn't heard of base62 until now! Seems like the exact way they do it, probably decoding from base62 instead of storing the base62 version of the key, as you put above. – Chris S Oct 11, 2008 at 8:58

2 I suspect TinyURL uses base 36 or at least base N where N < 62 and N >= 36. I don't think they let you use both lower case and upper case chars. If you want URLs to be easy to enter when dictated by someone over the phone, you don't want case-sensitive sequences. – Pavel Repin Jun 9, 2009 at 23:31

1 @Skliwz: How would you handle 9999999th id the same day? and probably 999....9th id at the end of the month? Meaning why would I ever want my url to look like: `http://site.com/999999999...999999w`? – Kamran Khan Aug 23, 2010 at 10:22

6

Look up the Birthday Paradox, it's the exact problem that you're running into.

The question is: How many people do you need to get together in a room, so that you have a 50% chance of any two people having the same birthdate? The answer may surprise you.

Share Improve this answer Follow

answered Oct 10, 2008 at 10:18

Greg Hewgill

990k ● 191 ● 1.2k ● 1.3k

4

The probability of a collision against one specific ID is:

```
p = ( 0.5 * ( (0.5*1/10) + (0.5*1/26) ) )^6
```

which is around 1.7×10^-9.

The probability of a collision after generating n IDs is 1-p^n, so you'll have roughly a 0.17% chance of a collision for each new insertion after 1 million IDs have been inserted, around 1.7% after 10 million IDs, and around 16% after 100 million.

1000 IDs/minute works out to about 43 million/month, so as Sklivvz pointed out, using some incrementing ID is probably going to be a better way to go in this case.

EDIT:

To explain the math, he's essentially flipping a coin and then picking a number or letter 6 times. There's a 0.5 probability that the coin flip matches, and then 50% of the time there's a 1/10 chance of matching and a 50% chance of a 1/26 chance of matching. That happens 6 times independently, so you multiply those probabilities together.

Share

Improve this answer

Follow

edited Oct 10, 2008 at 17:38

answered Oct 10, 2008 at 10:37

Randy

4,023 ● 21 ● 25

Bad idea to hash the ID - you need to get the unhashed ID back to lookup the row. See Sklivvz' answer. – MSalters Oct 10, 2008 at 13:07

I don't think your maths is quite right. The OP's data suggests ~50 collisions per million, while you predict 1700 (0.17% of 1000000). Perhaps I am missing something? – freespace Oct 10, 2008 at 15:55

I didn't mean an actual hash; I just meant to follow Sklivvz's answer. I'll edit my answer to clarify that. – Randy Oct 10, 2008 at 17:19

The probability of a collision on the next insert after 1,000,000 is 0.17%, not for every collision prior to that. As I said, the probability of a collision is not constant; it depends on the number of IDs already used. – Randy Oct 10, 2008 at 17:21

Looks like if you remove the 0-9 factor and remove the mutually exclusive part, the chance of a clash is a lot lower. Maybe just use the ascii code for 0-9 instead. This would be 1/26 x 1/26 x 1/26 x 1/26 x 1/26 x 1/26 = 0.038^6. What's the 0.5*1/26 for above? – Chris S Oct 11, 2008 at 8:53

4

Some time ago I did exactly this, and I followed the way Sklivvz mentioned. The whole logic was developed with a SQL server stored procedure and a couple of UDF (user defined functions). The steps were:

- say that you want to shorten this url: Creating your own Tinyurl style uid
- Insert the URL in a table

- Obtain the @@identity value of the last insert (a numeric id)
- Transform the id in a corresponding alphanumeric value, based on a "domain" of letters and numbers (I actually used this set: "0123456789abcdefghijklmnopqrstuvwxyz")
- Return that value back, something like 'cc0'

The conversion was realized thru a couple of very short UDF.

Two conversion called one after the other would return "sequential" values like these:

```
select dbo.FX_CONV (123456) -- returns "1f5n"

select dbo.FX_CONV (123457) -- returns "1f5o"
```

If you are interested I can share the UDF's code.

Share
Improve this answer
Follow

edited May 23, 2017 at 10:27
Community Bot
1 • 1

answered Oct 10, 2008 at 13:01
ila
4,724 • 7 • 38 • 41

2 This is why nobody should ever ask "should I share the code?" and they should just share the code. It's been over 5 years and KMan is still waiting. – Buns of Aluminum Dec 21, 2015 at 20:18

0

Why not just use a hashing algorithm? and use a hash of the url?

if you are using random numbers chances are you will get clashes because they are indeterminate.

hashes arent proovably unique but there is a fairly good chance that the hash of a string will be unique.

Correction

Actually wait you want them to be humanly readable... if you put them in hex they are technically humanly readable.

or you could use an algorithm that converted a hash into a humanly readable string. if the humanly readable string is a different representation of the hash it should also be as "unique" as the hash, ie base 36 of the original hash.

Share Improve this answer Follow

answered Oct 10, 2008 at 10:18

Omar Kooheji
55.6k ● 71 ● 186 ● 243

I think the point is this just simulates it but is not the actual hashing algorithm. Its so he can determine metrics, like the clash rate. – mattlant Oct 10, 2008 at 10:25

0

I would generate a random value representative of the data that you are going to hash, and then hash that and check clahses rather than trying to simulate with random manually made hashes. This will give you a better indicator. And you will have more randomness because you will have more to randomize (Assuming your data to be hashed is larger :)).

Share Improve this answer Follow

answered Oct 10, 2008 at 10:27

mattlant
15.4k ● 4 ● 36 ● 44

0

If you're using 6 characters, a-z and 0-9, thats a total of 36 characters. The number of permutations is thus 36^6 which is 2176782336.. so it should only clash 1/2176782336 times.

Share Improve this answer Follow

answered Oct 10, 2008 at 10:32

Ryan
4,901 ● 4 ● 25 ● 21

Yes, because the characters aren't distributed evenly. Consider a (really bad) algorithm: if you roll a D100 and the result is exactly 42, generate a very long GUID. Otherwise, result="Doh". Number of possible hashes: massive. Chance of clash? Huge! – Jon Skeet Oct 10, 2008 at 10:41

1 @Jon: any reasonable hash algorithm would have an even distribution. @Ryan: your calculation is correct if the question was "if I generated 2 hash values, what is the probability of them colliding?". However the question here retains previously generated values, so the maths is not so simple. – freespace Oct 10, 2008 at 14:22

0

from wikipedia:

> When printing fewer characters is desired, GUIDs are sometimes encoded into a base64 or Ascii85 string. Base64-encoded GUID consists of 22 to 24 characters (depending on padding), for instance:

```
7QDBkvCA1+B9K/U0vrQx1A
7QDBkvCA1+B9K/U0vrQx1A==
```

> and Ascii85 encoding gives only 20 characters, e. g.:

```
5:$Hj:Pf\4RLB9%kU\Lj
```

So if you are concerned with uniqueness, a base64 encoded GUID gets you somewhat closer to what you want, though its not 6 characters.

Its best to work in bytes first, then translate those bytes into hexadecimal for display, rather than working with characters directly.

Share Improve this answer Follow

answered Oct 10, 2008 at 10:34

cfeduke
23.2k 10 63 65