

TinyURL/Bitly

Functional Requirements

CreateShortURL(String longURL)

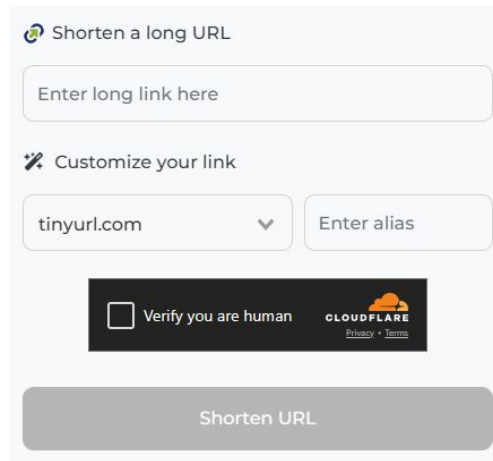
- Returns String shortURL*

FetchShortURL(String shortURL)

- Redirects user to corresponding longURL

*Short URL is in the format tinyurl.com/xxxxxxx

**We'll be leaving "PasteBin" and click counts for subsequent videos



The screenshot shows the TinyURL website interface. At the top, it says "Shorten a long URL". Below this is a text input field labeled "Enter long link here". Underneath that is a section titled "Customize your link" with a wrench icon. It features a dropdown menu currently showing "tinyurl.com" and a button labeled "Enter alias". Below the customization section is a dark box containing a checkbox labeled "Verify you are human" and the Cloudflare logo with links for "Privacy" and "Terms". At the bottom of the interface is a large grey button labeled "Shorten URL".

Supported Scale

URL Creations: 600 million per month = 228 per second

URL Retrievals: 10 billion per month = 3805 per second

Short URL Format

How many characters should we use for our URL suffix?

600 million creations/mo = 7.2 billion creations/y = 720 billion creations/century

- If each URL can contain a-z and 0-9, we have 36 choices per character
- If using 8 characters, we have $36^8 = 3$ trillion possibilities

URL Mapping Table

Short URL Suffix	Long URL	CreatorID	Expiry Time
hlyu76tt	JordansOnlyFans.com	23	Jan 1 2026, 00:00:00
09iomn43	JordanIsSmelly.com	42	Jan 1 2025 00:00:00
45647hvc			Jan 1 1970, 00:00:00

- 8 bytes for suffix
- 50 bytes (on average) for long URL
- 8 bytes for CreatorID
- 8 bytes for expiry time

74 bytes per row!

Number Of Database Nodes

3805 database reads per second

- Simple reads, can probably be handled on a single node

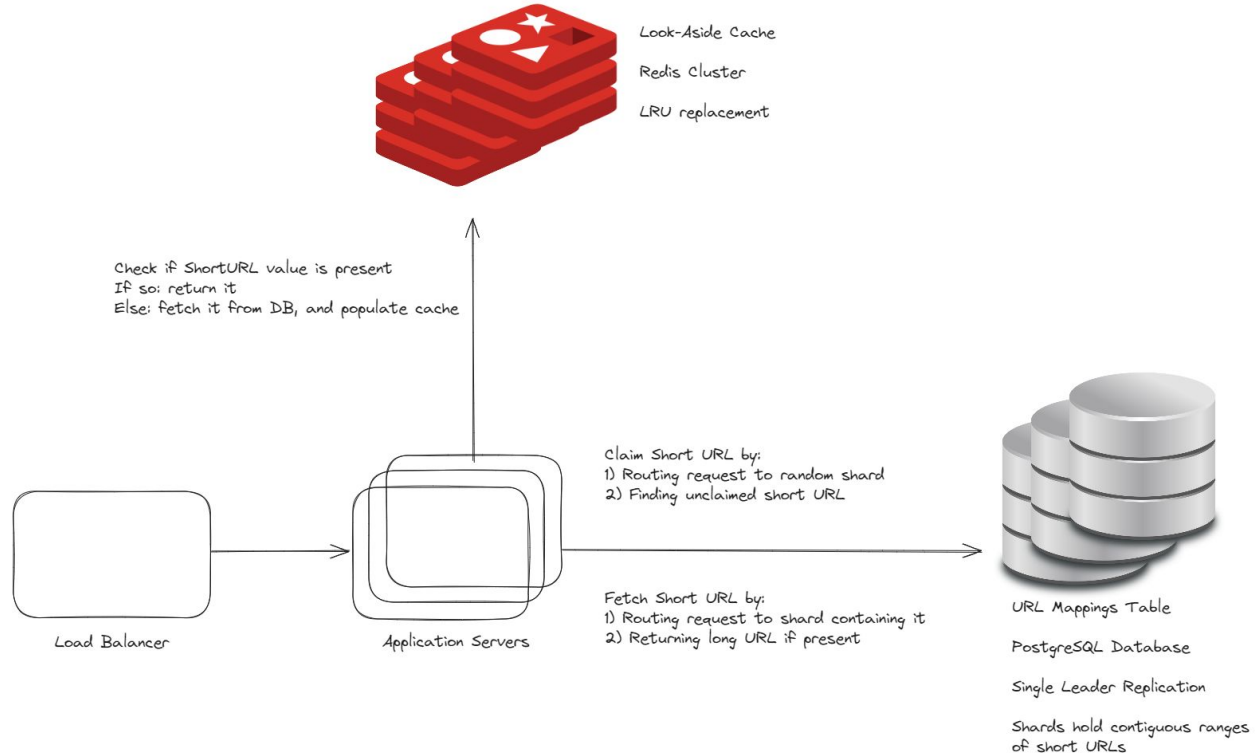
228 writes per second

- Can most likely be handled on a single node, depending on write complexity

74 bytes * 720 billion = 48TB of data

- Can be handled on a beefy node, but likely worth horizontally scaling

High Level Design



Key Generation Service Overview



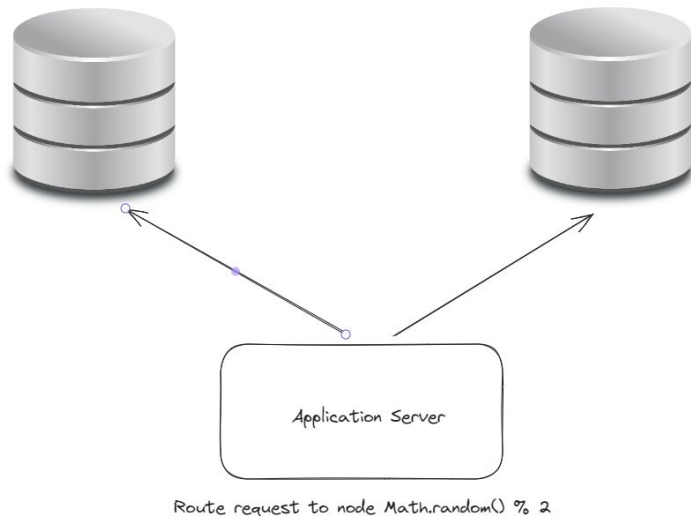
shortURL	long URL	clientId	expiration
0000000			0
0000001			0
0000002			0
.			.
.			.
.			.
hzzzzzzz			0



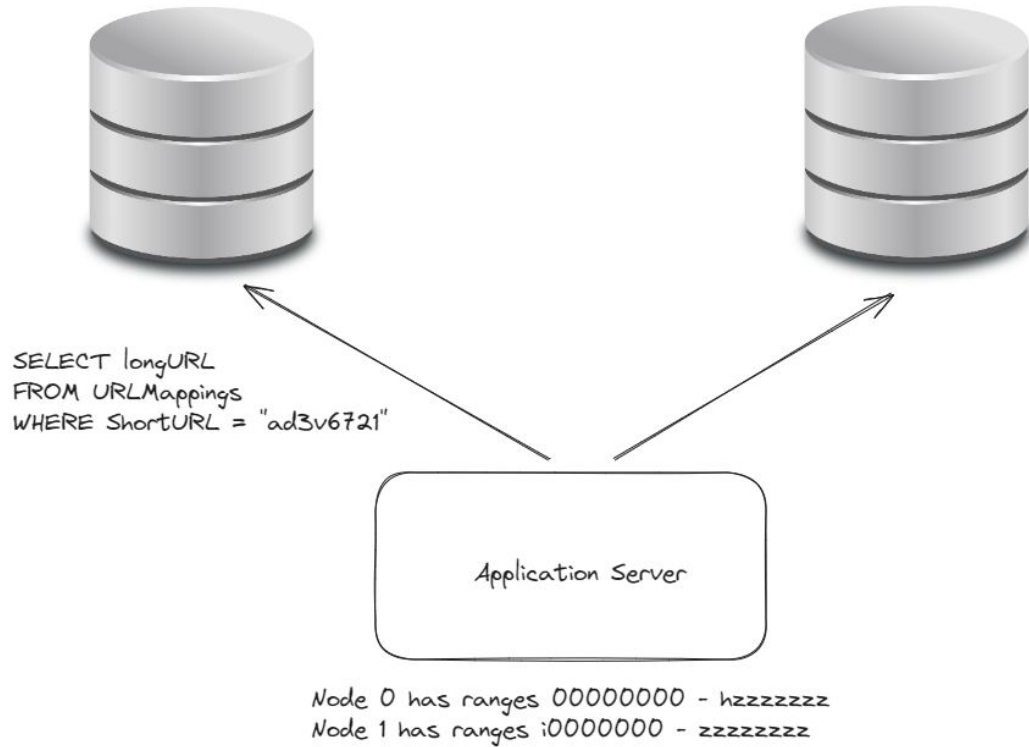
shortURL	long URL	clientId	expiration
i000000			0
i000001			0
i000002			0
.			.
.			.
.			.
zzzzzzzz			0

Creating A Short URL

```
WITH candidate AS (  
  SELECT *  
  FROM URLMappings  
  WHERE expiryTime < NOW()  
  LIMIT 1  
  FOR UPDATE SKIP LOCKED  
)  
UPDATE URLMappings  
SET expiryTime = NOW() + INTERVAL '1 year',  
    userID = 4  
    longURL = "toes-are-my-passion.com"  
FROM candidate  
WHERE URLMappings.id = candidate.id  
RETURNING URLMappings.*;
```



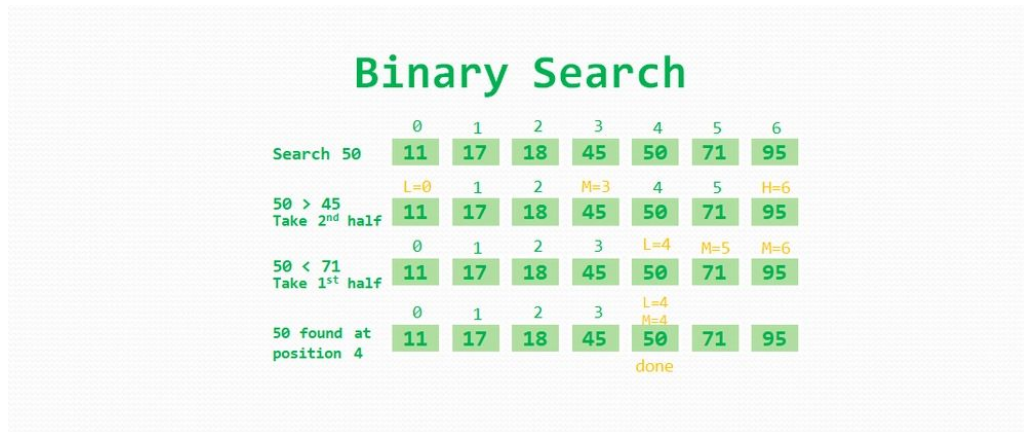
Retrieving A Short URL



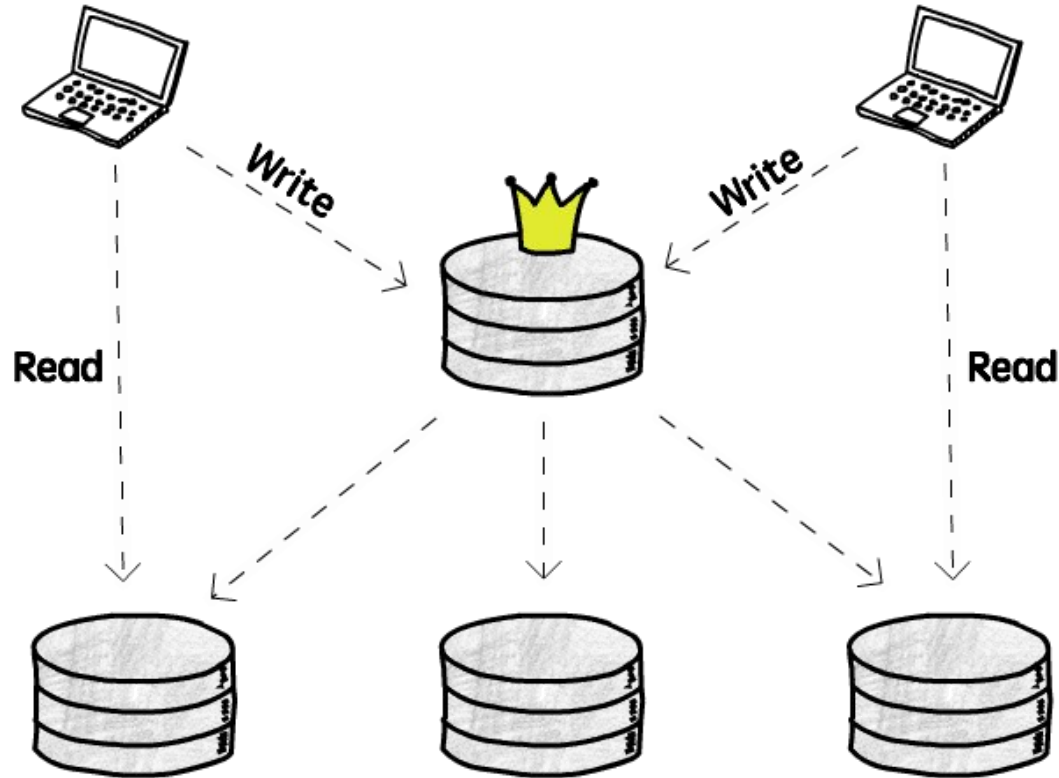
Database Indexing

Example Query:


```
SELECT longURL FROM URLMappings WHERE ShortURL = "ad3v6721"
```



Database Replication



Caching Overview




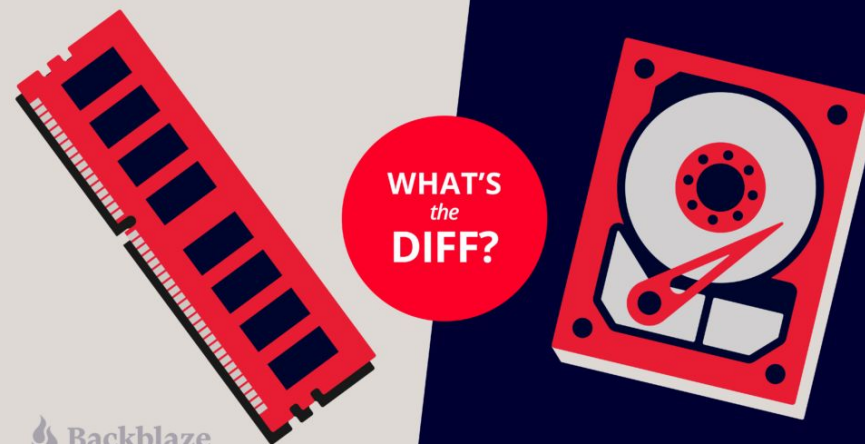
The illustration shows a large group of small, light-blue human figures on the left, and a smaller group of larger, dark-blue human figures on the right. Below the figures are several stacks of coins, with the tallest stack positioned under the larger group of figures, visually representing the 80/20 rule of the Pareto Principle.

Pareto Principle

[pə-ˈrā-(l) tō ˈprin(t)-s(ə-)pəl]


The observation that, although there are often many causes for any observed phenomenon, it is often the case that a small subset of those causes are responsible for most of the observation.

 Investopedia



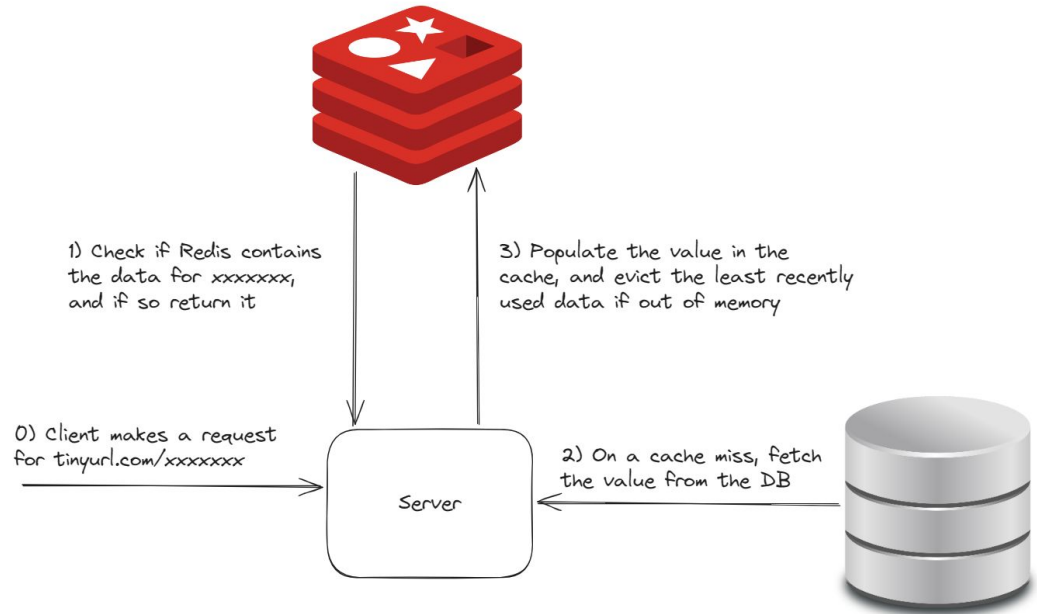
The illustration compares two types of storage. On the left, a red SSD (Solid State Drive) is shown at an angle. On the right, a red HDD (Hard Disk Drive) is shown, with its internal platters and read/write head visible. A red circle with white text is placed between them.

**WHAT'S
the
DIFF?**

 Backblaze

Redis - Look Aside Caching

- 1% of keys account for 50% of load
- We need 480gb of memory to serve 2000 reads per second



Deep Dives

- Database Contention
- Database Sharding
- Citus
- Hashing vs. Key Generation Service
- Choosing A Database
 - Index
 - Replication
- Caching

Deep Dives - Database Contention

```
WITH candidate AS (  
  SELECT *  
  FROM URLMappings  
  WHERE expiryTime < NOW()  
  LIMIT 1  
  FOR UPDATE SKIP LOCKED  
)  
UPDATE URLMappings  
SET expiryTime = NOW() + INTERVAL '1 year',  
    userID = 4  
    longURL = "toes-are-my-passion.com"  
FROM candidate  
WHERE URLMappings.id = candidate.id  
RETURNING URLMappings.*;
```

Key1 - claimed

Key2 - claimed

Key3 - claimed

Key4 - locked

Key5 - unclaimed (don't block, skip to here!!)

Key6 - unclaimed

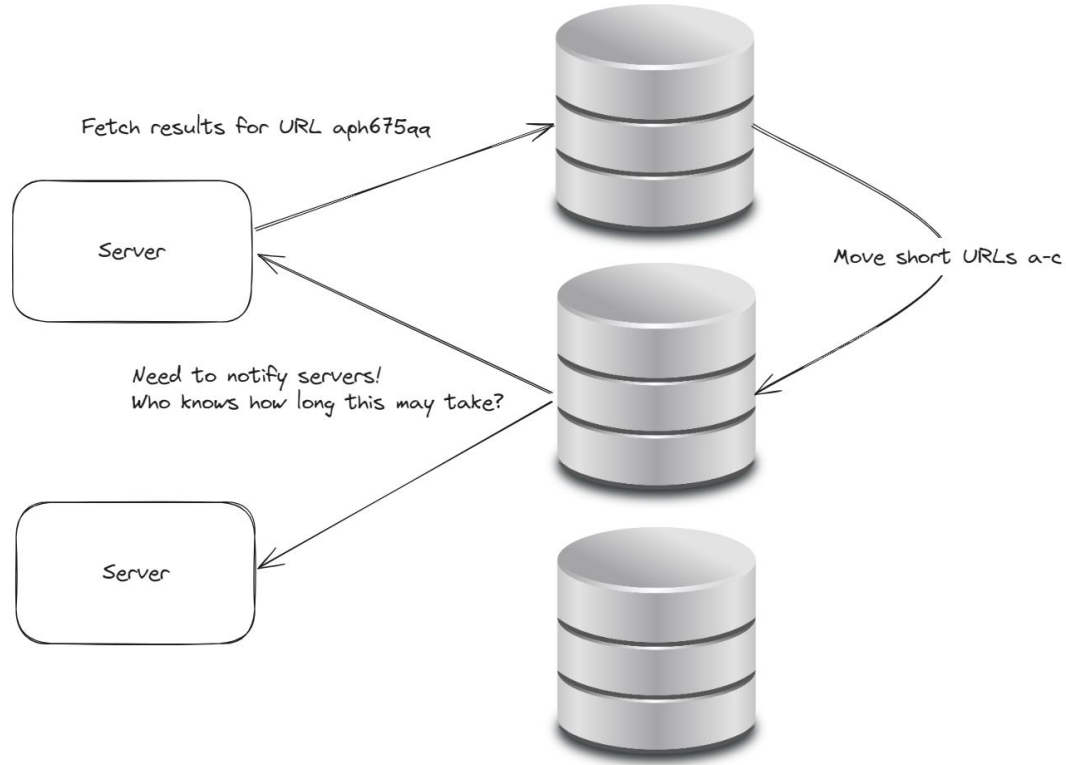
Key7 - unclaimed

Key8 - unclaimed

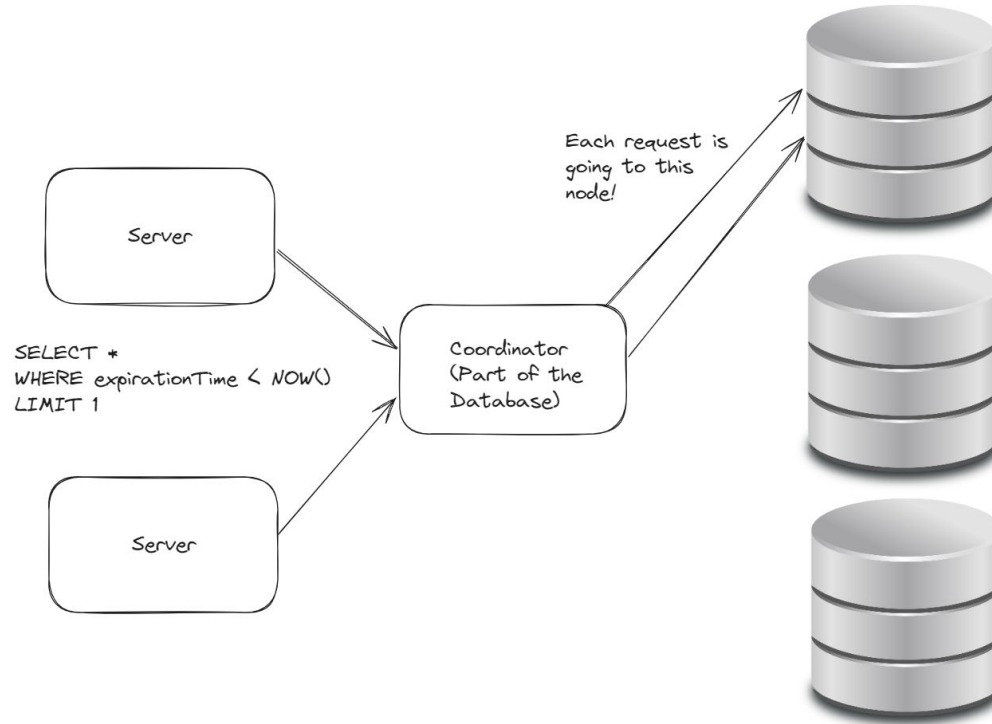
Key9 - unclaimed

Key10 - unclaimed

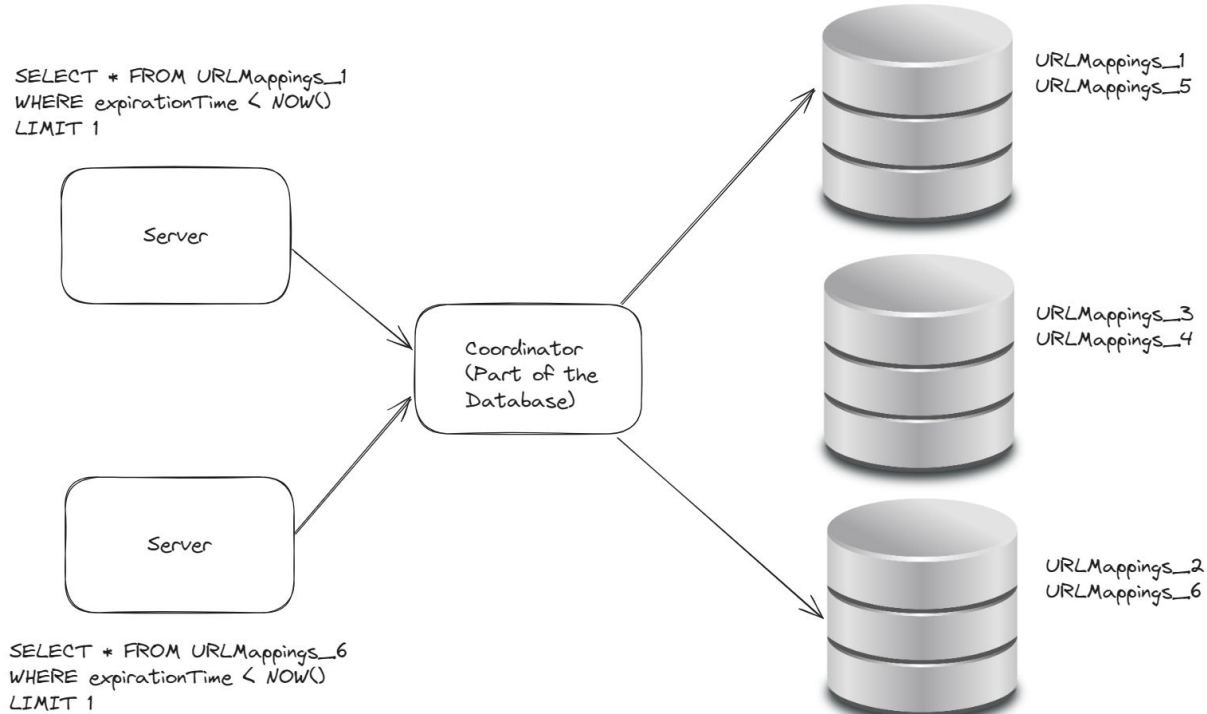
Deep Dives - Key Rebalancing



Deep Dives - Database Managed Request Routing



Deep Dives - Citus



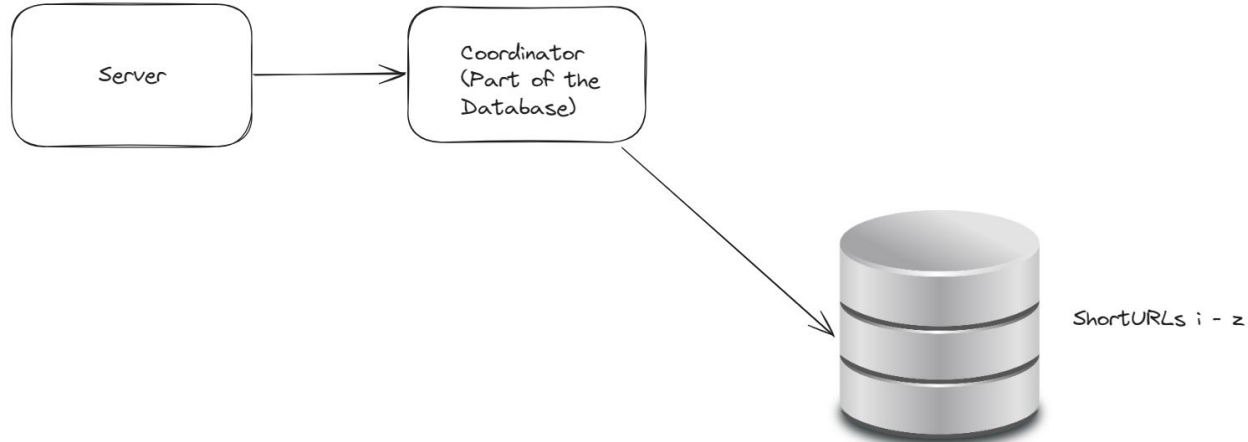
Deep Dives - Hashing Vs. Key Generation Service

Create a short URL for:

-> Jordansjellybeans42.edu from userId 31 at time x

1) Take hash and truncate to 8 chars = zlmn9876

2) If already taken, retry with zlmn9877, repeat until success

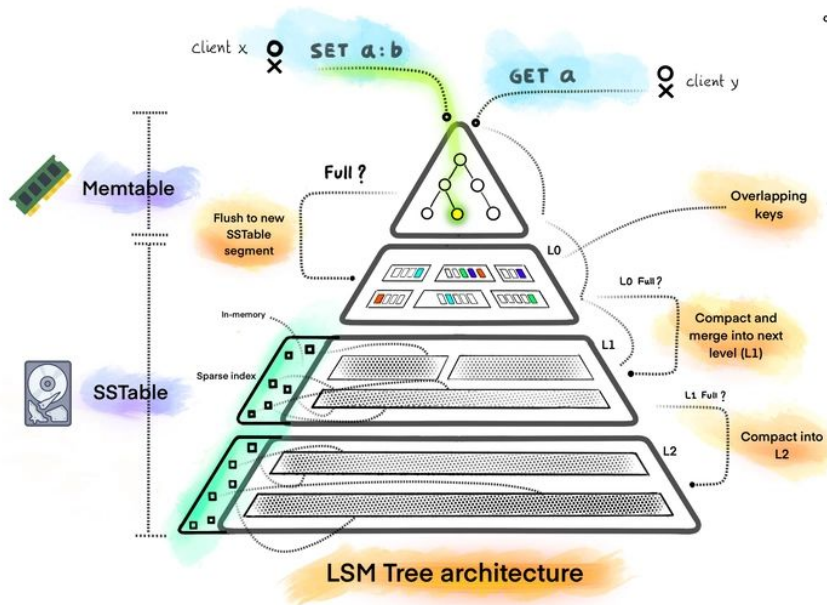


Deep Dives - Choosing A Database

- Index - PostgreSQL uses a B-Tree
- Replication - PostgreSQL uses single leader replication

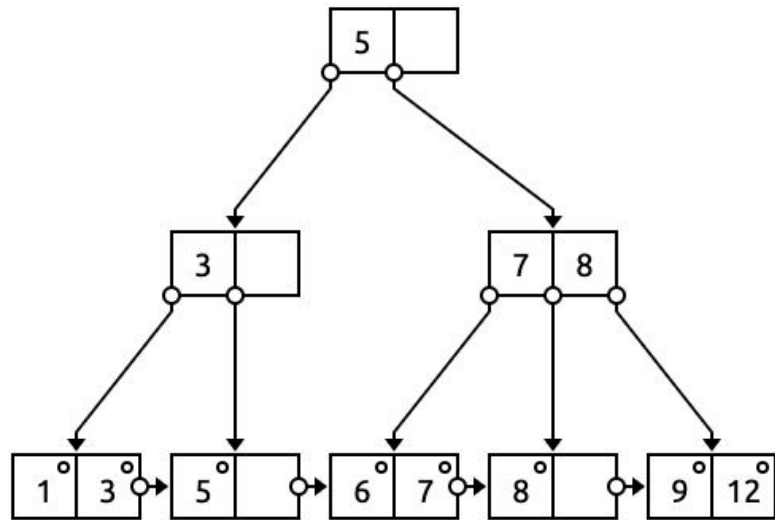
Maybe sharding techniques as well? Already covered before with Citus

Deep Dives - Database Index



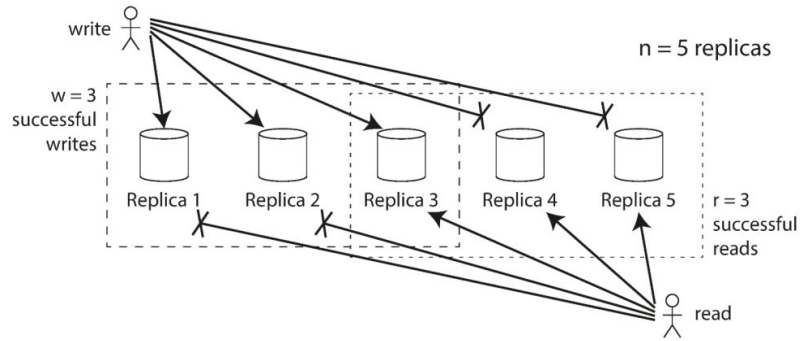
- Writes go to in-memory sorted memtable
- When memtable gets full, flush to SStable file on disk
- SStables can be compacted to reclaim space taken up by old values of keys
- Reads first check memtable, then SStables in order from newest to oldest

creativocoder.dev

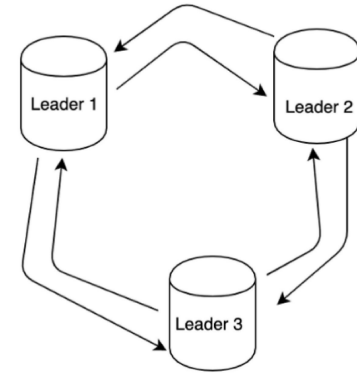


- Reads/writes traverse B-tree on disk and operate on data in place

Deep Dives - Replication

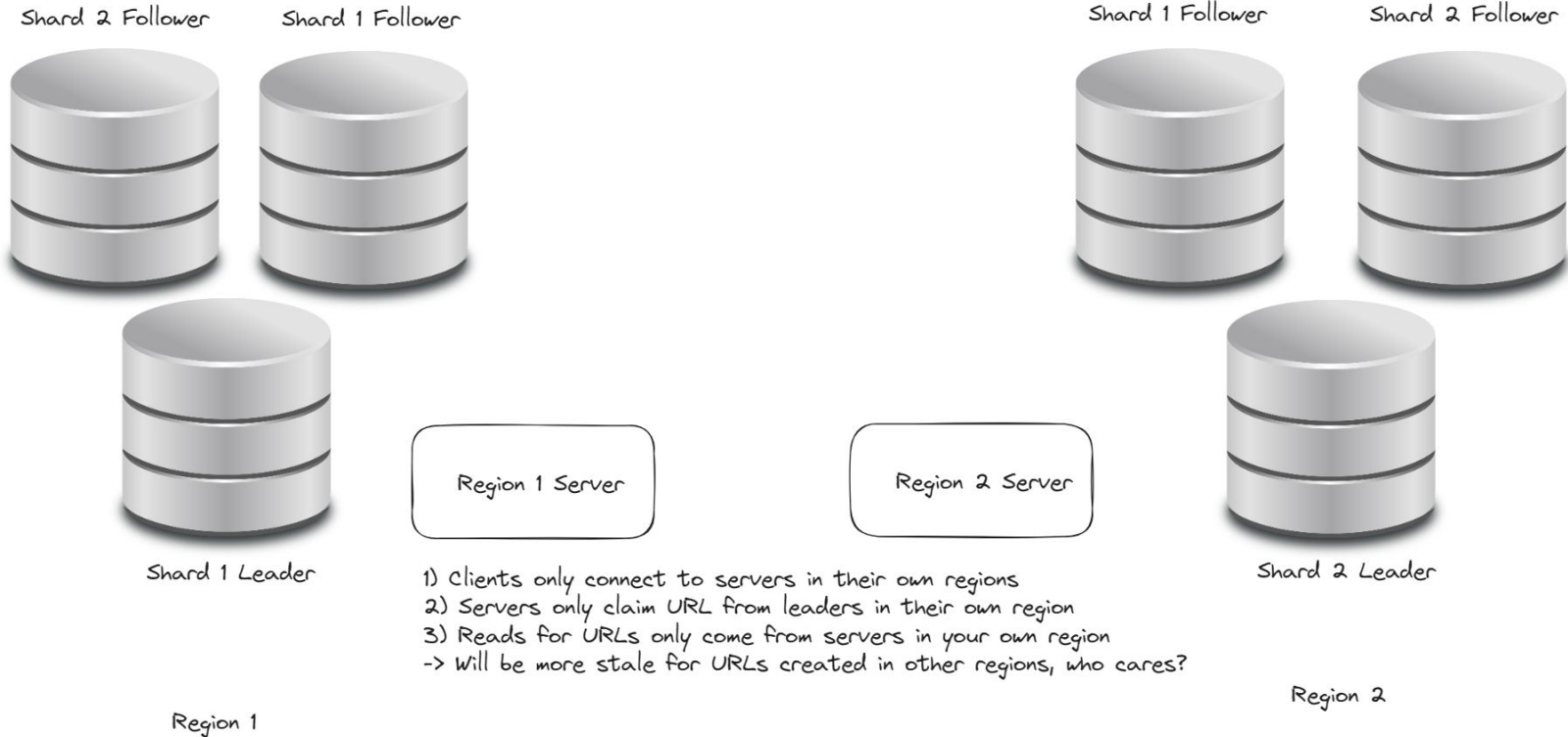


Leaderless Replication



Multi-Leader Replication

Deep Dives - Multiple Regions



Deep Dives - Read Through Caching

