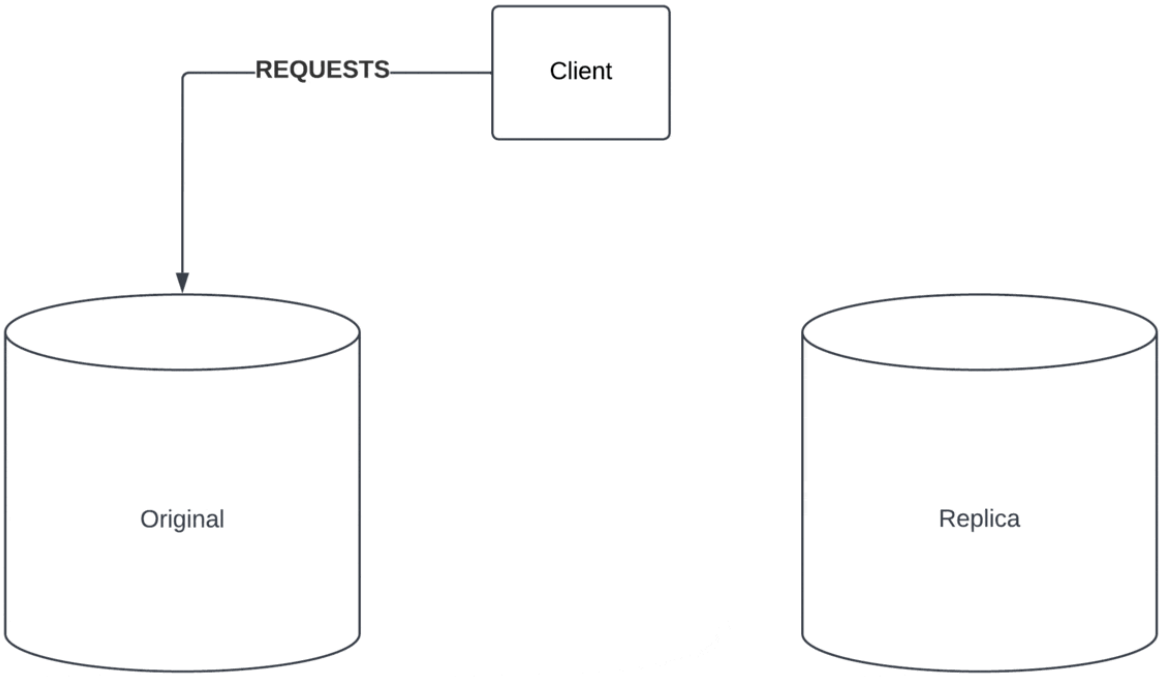
So you found a flashy new database technology. You run some tests, check the performance, and are convinced!

Time to speak to your manager!
But wait…

How do you move millions of records, each entry representing a user's aspirations, into a new database? Without any downtime? AND without any mistakes…

After sweating profusely, you come up with a plan:

Brute Force Approach

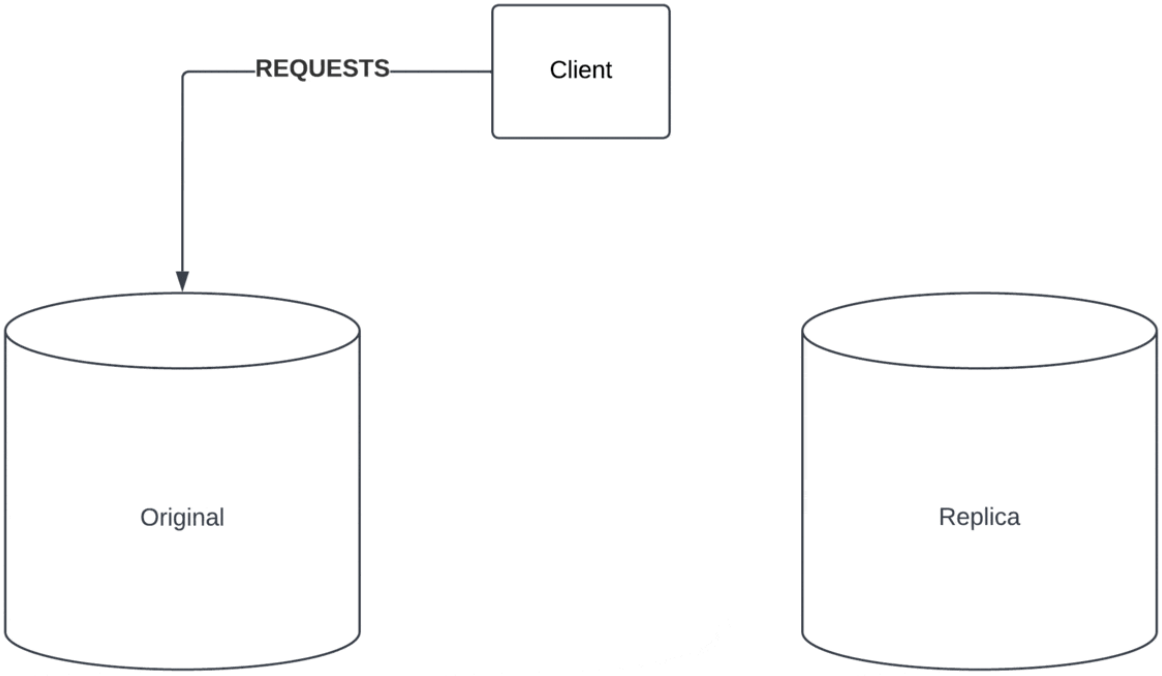


1. Stop the existing DB.
2. This fails all incoming requests to the DB (You should probably have some email communication setup beforehand to notify users of a planned outage).

3. Take a Data Dump of the current DB into your shiny new DB.
4. When the process completes, point the existing servers to the new DB. This will take a deployment or a server restart.

What did you say? You can't affort downtime at all? No worries, we have your back!

Optimised approach



This is going to be more complex engineering, but worth it the savings in downtime.

1. Set up a change data capture solution (Similar to a SQL trigger).
2. Take a database copy of all older records.
3. Setup a view or proxy to serve existing clients through the old database.

4. Set the clients to point to the proxy.
5. Now point the proxy to the new database.
6. Point clients to the new database directly (this requires a deployment or server restart).
7. Delete the proxy.

'

Yey! Remember, despite our best efforts, there is always a chance of a data miss. It's best to check for correctness after deployment, and keep your clients in the loop!