#ASLI ENGINEERING

Should some microservices share a database?

SWIPE

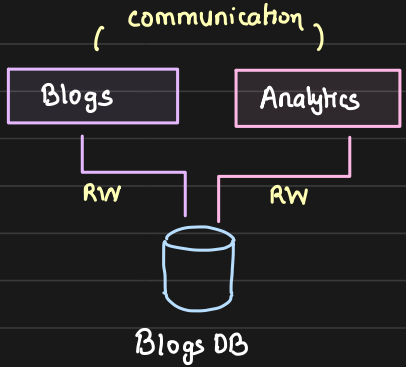
BY

ARPIT BHAYANI

Microservices sharing a database

One of the simplest ways to integrate a couple of microservices is to let them share a database.

eg: Blogs service stores the published blog in Blogs DB, while the analytics service also updates the total views a blog got in the same DB.




```
{
  "id": "sharing-a-db",
  "title": "_____",
  "total views": 1729
}
```

This is the simplest approach to integrate microservices. Anyone who wants to read or change anything can just read or update the database directly.

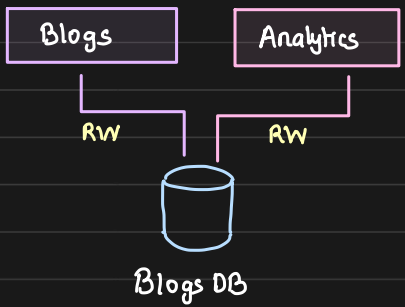
Advantages of this approach

- simplest way of integration
- no middlemen involved
- no latency overhead
- quick development time
- simpler operation
- better performance - no middlemen

Apart from all of these advantages, there are a few disadvantages to keep in mind...

Challenge 1: External parties are getting internal details

By sharing a database an external party, analytics, gets the internal schema, and other implementation details of Blogs service



- what's the schema
- design decisions
  - soft delete v/s hard deletes
  - normalization
  - redundancy

Given that an external service has access to the database,

- what if the Blogs service thinks of changing the schema?
  - better performance
  - better maintainability

  The analytics service would need to change its logic accordingly or the changes made by the Blogs service should always be Backward compatible

- what if the Blogs service wants to move from relational to non-relational?

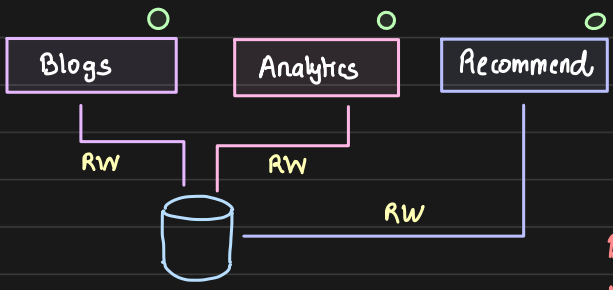
  Because of this tight coupling, Blogs service cannot take an independent call

  So, autonomy of Blogs team on their service is gone!

So, we now have tight coupling...

Challenge 2: Sharing DB = sharing Business logic

Say to render a particular data there are a few specific tables to fetch data from : T_1, T_2, T_3 and T_4



The logic to fetch the information is implemented by all the dependent services.

But, what if Blogs team changes the logic and now uses T_1, T_2, T_7, T_4

All the dependent services will have to change the logic at their end

So, we lose cohesion...

Core principle behind microservices are

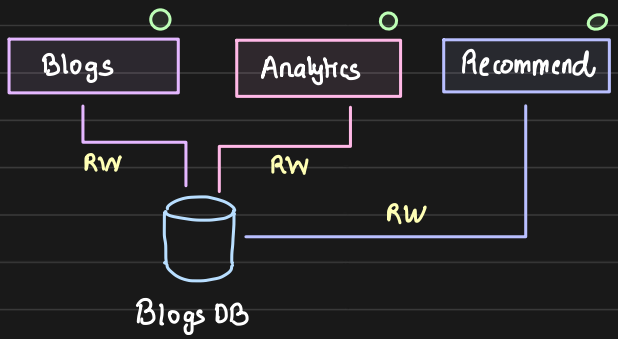
- loose coupling ✗
- High cohesion ✗

We are losing both by sharing the DB

Challeng 3 : Risk of data corruption and deletion

Now that all the dependent services have WRITE access to the same database, there are massive chances of someone

- corrupting the data
  - wrong script
  - limited knowledge
- accidentally deleting all the data



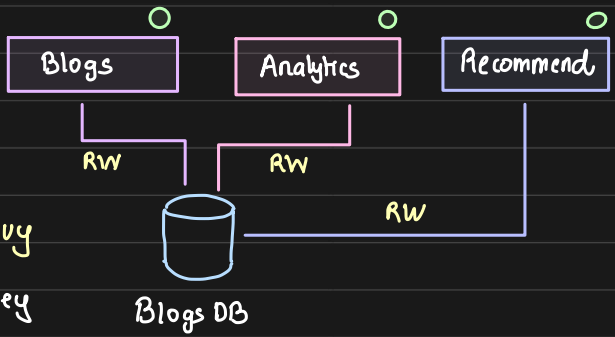
DB ACL has to be managed well so as to prevent this

Challenge 4: Abusing the shared DB

Say, analytis service wrok a few new queries to render some new user facing dashboards and these queries are super-heavy

This will affect other services as they depend on the same DB.

No way to automatically throttle DB queries



Given that there are a few challenges.

but does this mean we should never share a DB ?

No, there are places where it is beneficial to share a DB

1. Sharing a DB is a Quick solution, when crunched on time

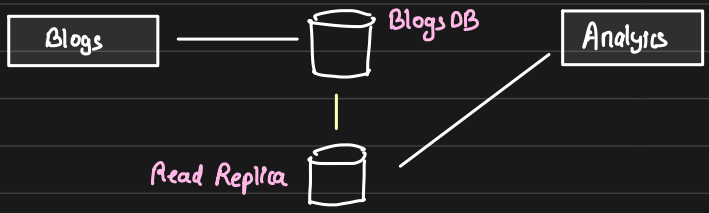
Doing this requires efforts and co-ordination from multiple teams

2. When schema does not change often

Schema or business logic do not change often, so why create that unnecessary dependency

3. Read load can be moved to a Replica

Heavy analytics query can run on a separe replica of the Blogs DB



This way dependent system does not put a high read load on the main DB