Class #12

04. Software Architecture Patterns

Software Architectures

Master in Informatics Engineering

Cláudio Teixeira (claudio@ua.pt)

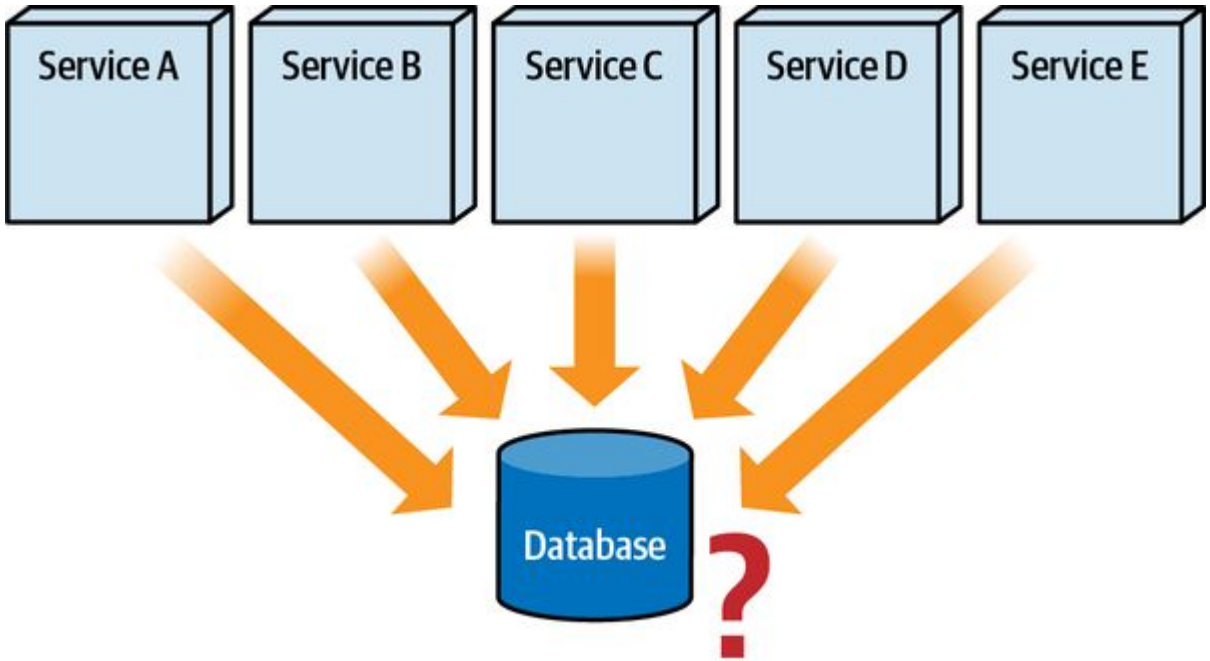
Agenda

- Data Domains
- Design patterns And decisions

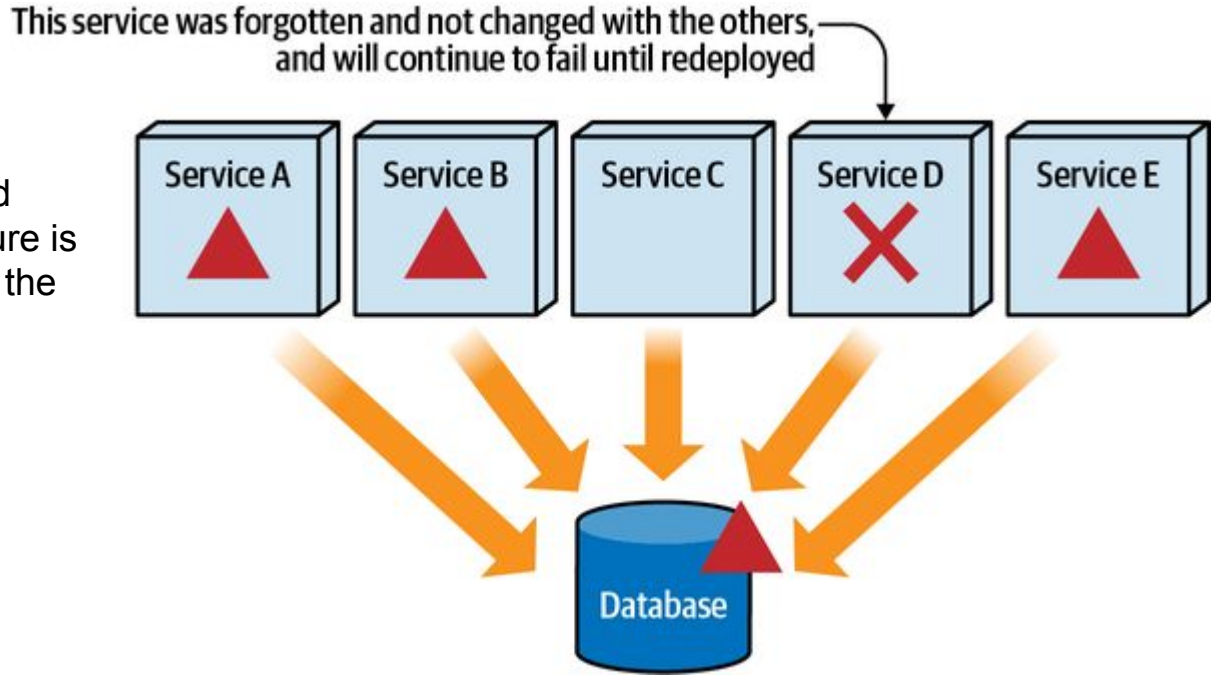
Moving data around

https://learning.oreilly.com/library/view/software-architecture-the/9781492086888/ch06.html#idm45978844975728

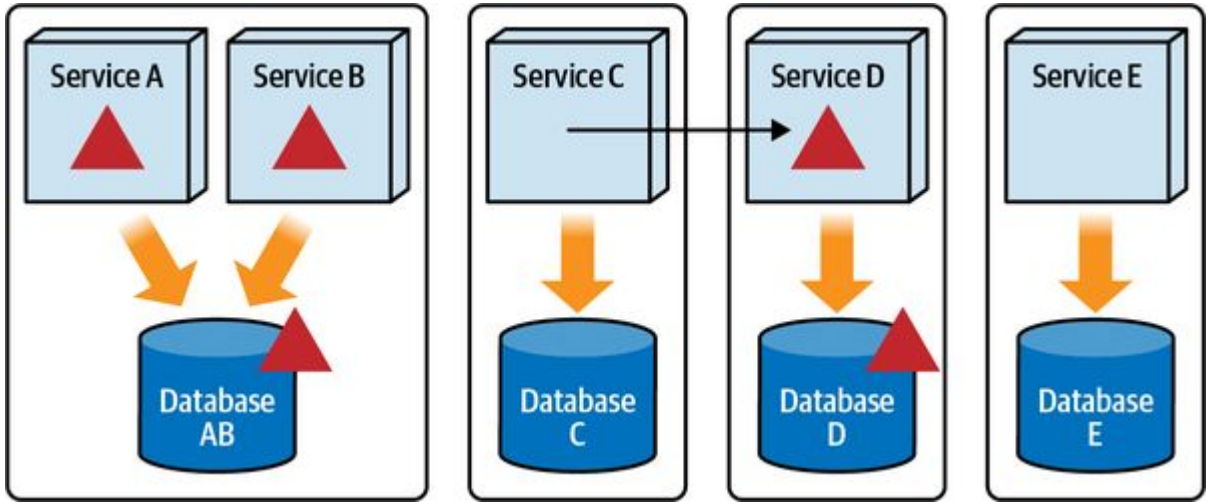
Why to move away from this?

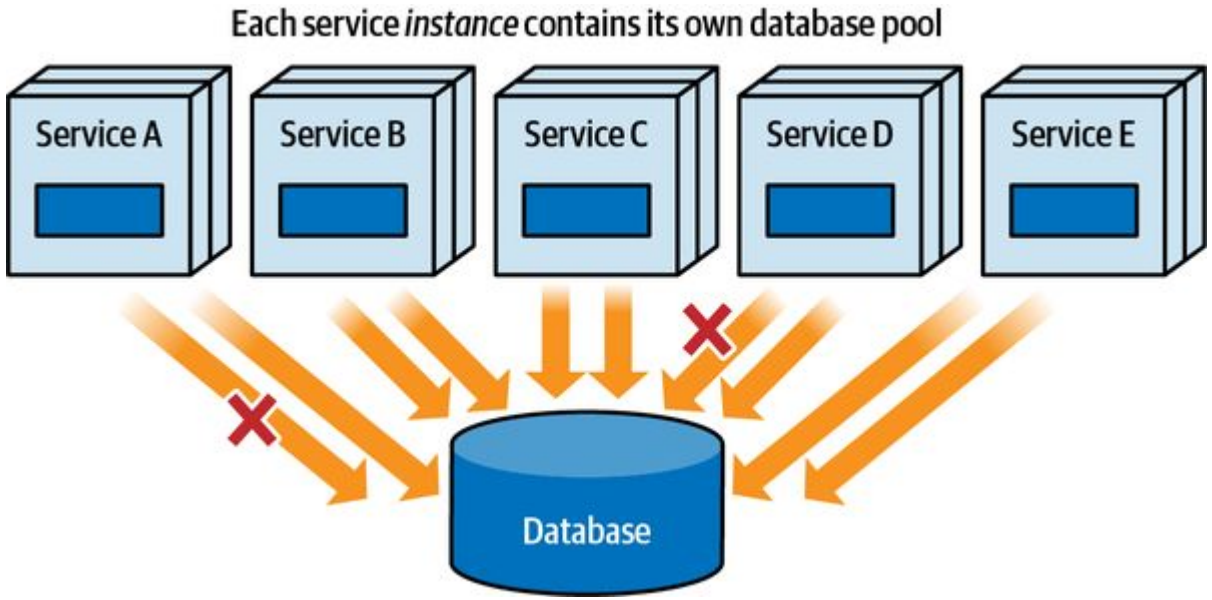


The real danger of changing a shared database in any distributed architecture is forgetting about services that access the table just changed.



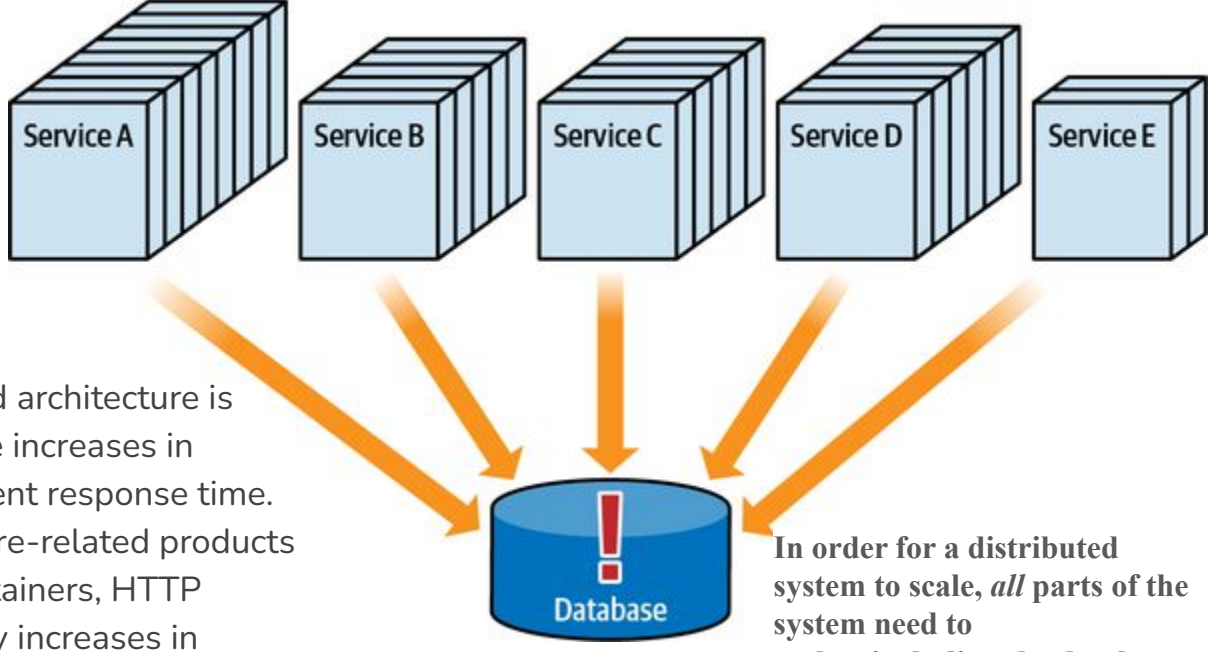
Breaking apart a database into well-defined bounded contexts significantly helps control breaking database changes. The bounded context concept comes from the seminal book Domain-Driven Design by Eric Evans (Addison-Wesley) and describes the source code, business logic, data structures, and data all bound together—encapsulated—within a specific context.





Establishing a connection to a database is an expensive operation. A database connection pool is often used not only to increase performance, but also to limit the number of concurrent connections an application is allowed to use. In monolithic applications, the database connection pool is usually owned by the application (or application server). However, in distributed architectures, each service—or more specifically, each service instance—typically has its own connection pool.

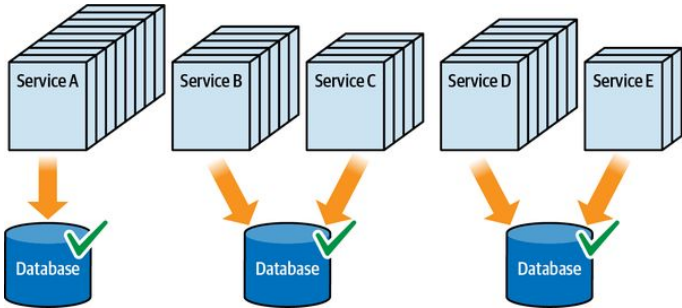
Original monolithic application	200 connections
Distributed services	50
Connections per service	10
Minimum service instances	2
Total service connections	1,000



One of the many advantages of a distributed architecture is scalability—the ability for services to handle increases in request volume while maintaining a consistent response time. Most cloud-based and on-prem infrastructure-related products do a good job at ensuring that services, containers, HTTP servers, and virtual machines scale to satisfy increases in demand. But what about the database?

In order for a distributed system to scale, *all* parts of the system need to scale—including the database.

Breaking data into separate data domains or even a database-per-service, requires fewer connections to each database, hence providing better database scalability and performance as the services scale.



Azure proposed Patterns

https://learn.microsoft.com/en-us/azure/architecture/browse/

Implementation plan & execution

- https://learn.microsoft.com/en-gb/azure/architecture/web-apps/guides/reliable-web-app/dotnet/plan-implementation
- https://github.com/Azure/reliable-web-app-pattern-dotnet

Prepping Implementation & Execution

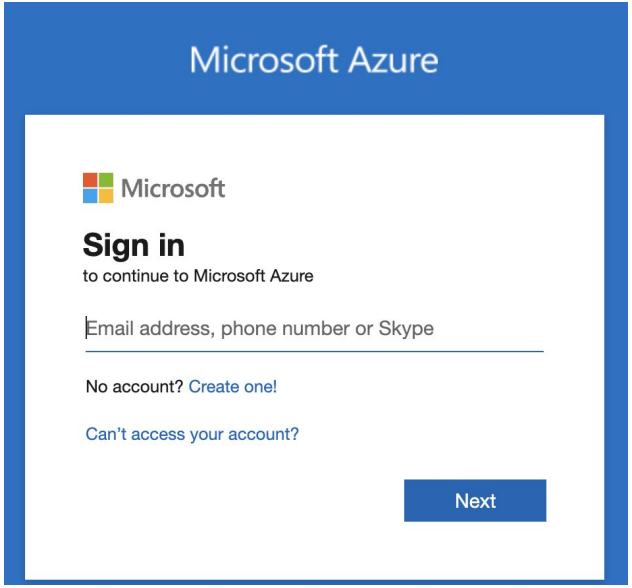
- Define business goals (as always)
- Choose the right managed services
  - Application platform (Autoscaling, security, pipelines & automation, etc.)
  - Identity management (Authentication and authorization, oAuth 2.0, etc..)
  - Database *(reliability, resiliency, performance, security, ...)*
  - Application performance monitoring (*Anomaly detection, troubleshooting, monitoring, etc.*)
  - Cache (*speed, volume, nonsticky sessions, ...)*
  - Load balancer (*Global load balancing, probes, DDos Protection, CDN, etc.)*
  - Web application firewall
  - Configuration storage (*Configuration flexibility, pipelines, etc*)
  - Secrets manager *(encryption, managed identities, monitoring, logging, ease of integration)*
  - Storage solution *(security, encryption, resiliency)*
  - Endpoint security (*Enhanced security communication, minimal effort*)
  - Network security (shared/isolated, segmented, etc)

Patterns, patterns, patterns

- https://learn.microsoft.com/en-us/azure/architecture/patterns/
- https://learn.microsoft.com/en-us/azure/well-architected/pillars

https://portal.azure.com/

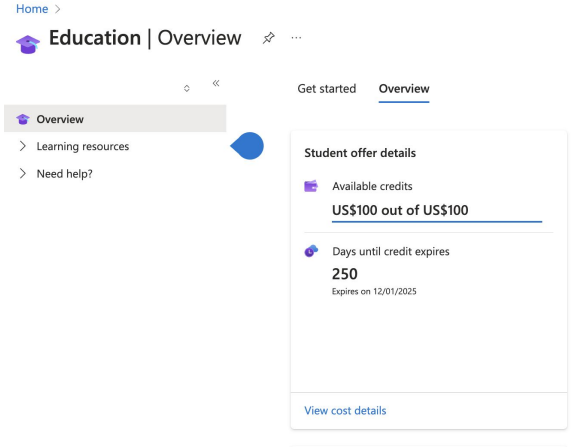
Sign in with @ua.pt login!

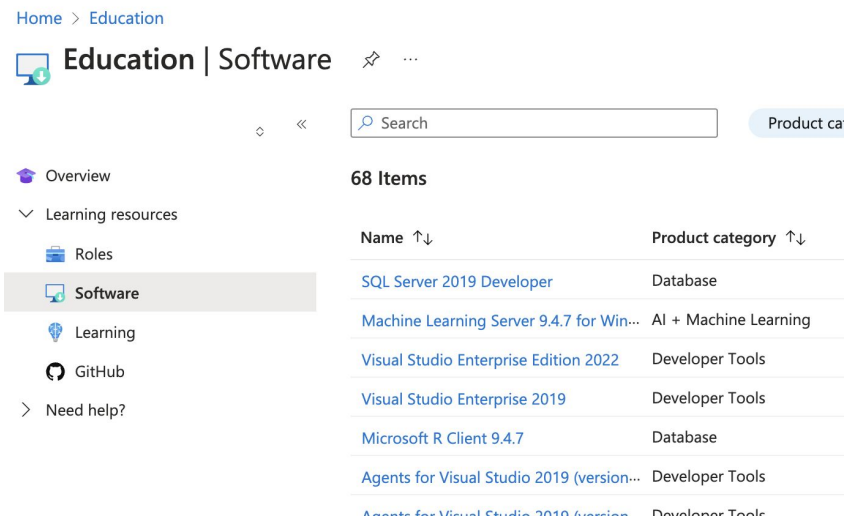


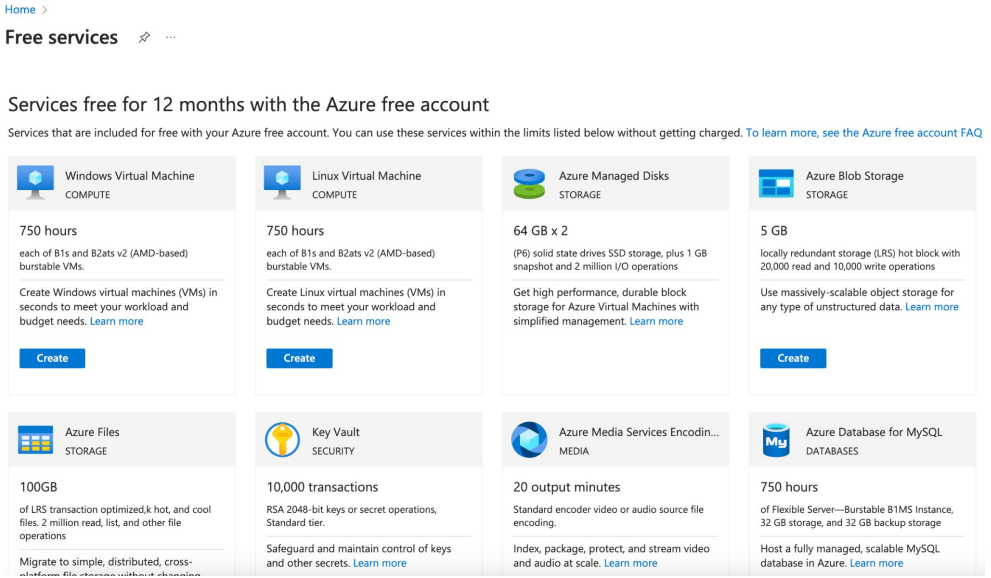
Access student benefits

Get free software, Azure credit, or access Azure Dev Tools for Teaching after you verify your academic status.

Explore Learn more







https://portal.azure.com/#view/Microsoft_Azure_Billing/FreeServicesBlade

Some of the perks of being student in Aveiro

Deploy Async Request example @Azure

Group assignment
30 min

https://learn.microsoft.com/en-us/azure/architecture/patterns/async-request-reply
https://github.com/mspnp/cloud-design-patterns/tree/main/async-request-reply

Circuit Breaker pattern @.net

- https://learn.microsoft.com/en-us/azure/architecture/patterns/circuit-breaker
- What are the main points for this pattern?
- spot how to use it in the example
- Health Endpoint Monitoring pattern

Retry Pattern

- https://learn.microsoft.com/en-us/azure/architecture/patterns/retry

Identify patterns for final assignment

Group assignment
120 min

Look into implementation details, try the examples!
prep a table accordingly with

Bibliography

- https://learn.microsoft.com/en-us/azure/architecture/browse/
- https://learning.oreilly.com/library/view/software-architecture-the/9781492086888/ch06.html
- https://learn.microsoft.com/en-us/azure/architecture/microservices/migrate-monolith