Davide Mauri

SQL Server / Azure Data MVP for 12 Years
Worked in consulting services for 20 years
Joined Azure SQL group on mid 2019
- Still a developer at heart!
- Now Azure SQL PM

Focus on Azure SQL & Developers
Very active in the Community, Conference Speaker
- Website: http://davidemauri.it/
- DevBlogs: https://devblogs.microsoft.com/azure-sql/
- Twitter: @mauridb

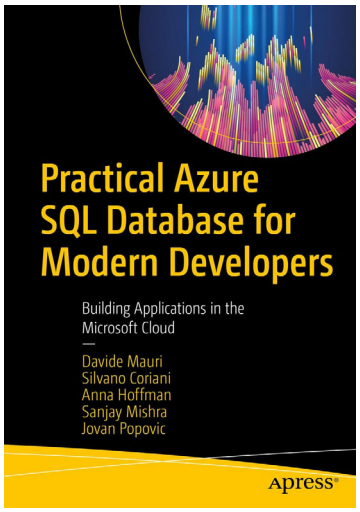
Visual Studio LIVE!

A book for the modern developer

A developer-focused book, to help you leverage all the relational and post-relational features that Azure SQL has, to easily create fast, scalable and secure applications

Lots of samples and discussions taking into account different languages:

- Python, .NET, Java
- …and more!



Visual Studio LIVE!

Session Goal

Stimulate discussion around new architectures options made available by native cloud database

- Review well-established beliefs / best practices / patterns
- Lateral Thinking

Visual Studio LIVE!

Application Scaleoutability

Modern applications are all based on the idea that they can/should scale-out easily

- Azure Functions / Web Apps / Containers

Creating a scale-out application is quite easy

- Stateless is the way
- Just move state management outside the app boundary
  - In a database usually

Visual Studio LIVE!

Database Scaleoutability

Scaling out a database is much more complex

- You cannot delegate something else to manage state/data

A distributed database is *way* more complex than a single-instance database

- Yeah, CAP theorem and all that jazz.
- But the CAP theorem is just the tip of the iceberg!

Visual Studio LIVE!

Challenges of distributed databases

Partitioning / Sharding
Data Distribution / Duplication
Conflict Resolution
Data Availability
Latency
Data Movement
Traffic Management
Etc..

Is really sharding the only option to scale out?

A new approach?

An application is built to be abstracted from the underlying database

- Well…not always. You want to take advantage of the features that the database you're using is providing you, right?
- Yeah, loose coupling and abstraction are good things in general. And we DO NOT WANT to lose them!

But, in general, an application is built assuming that there are exactly "n" database behind the scenes. Usually, n=1.

- More in general and application doesn't know if it is using a distributed database or not.

A new approach?

What if we change that idea?

- What if the application is AWARE that the database is might be distributed?
- Maybe it can take advance of this knowledge
  - App will still be perfectly able to work with a single database

A set of controlled nodes

Usually, distributed database provides one common entry point

- Shields the number of nodes actually used to serve and process data
- Usually there is a control node and ”n” worker nodes

A set of independent nodes

Let's try to change this: let the application know that there are many different nodes

Each one freely usable

Each one independent from the others

A happy set of nodes

How to know which node contains the data the app is interested in?

- Solve the problem at the root: all nodes will be able to access all the data

A happy set of diverse nodes

What if a distributed database can have nodes of different sizes?

- Maybe they can be used for different purposes?

Power to the App!

What if an application can decide where a query should be executed?

- Not always, but maybe for some queries it can express some preference.
  - For example, Read queries should be separated by Write queries
  - Or Reporting queries should be executed on a different node than those used for running OLTP queries

Query Tagging

If a query can be "tagged" it can be routed to the best (set of) nodes

- If they are available
- Idea conceptually similar to the "Data Dependent routing" technique…but based on metadata instead of data.

Let's simplify the problem for now, by avoiding write conflicts and having writes done only in one node

- OLTP is still 80% read and 20% anyway, so this should not be really a problem

Visual Studio LIVE!

Azure SQL DB Hyperscale

Thanks to its cloud-native architecture (it's really a set of distributed services), Azure SQL DB Hyperscale is perfect for this idea

Thanks to Named Replicas we can have up to 30 Named Replicas each one independent from each other and from the primary

- But all will be seeing the SAME data!

Visual Studio LIVE!

Idea Pre-Conditions

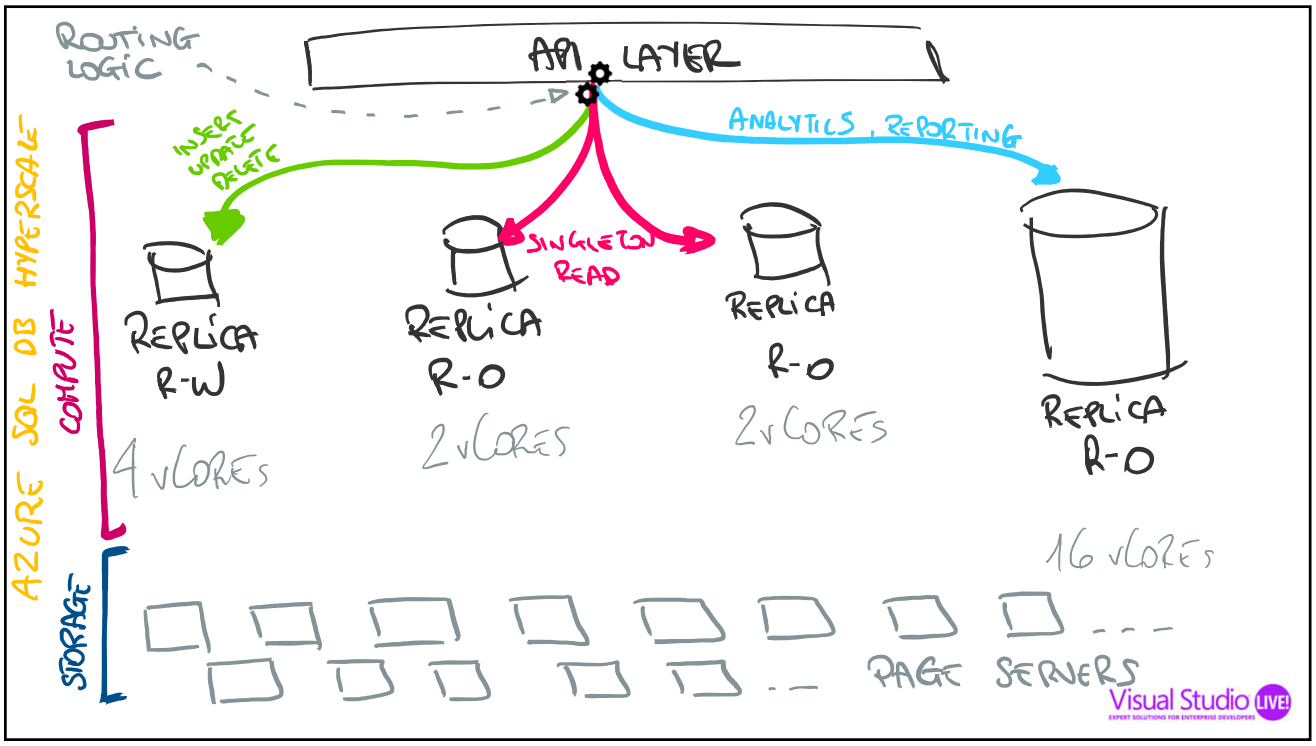
Borrowing the idea from microservices architecture, access to the database is governed by the API layer.

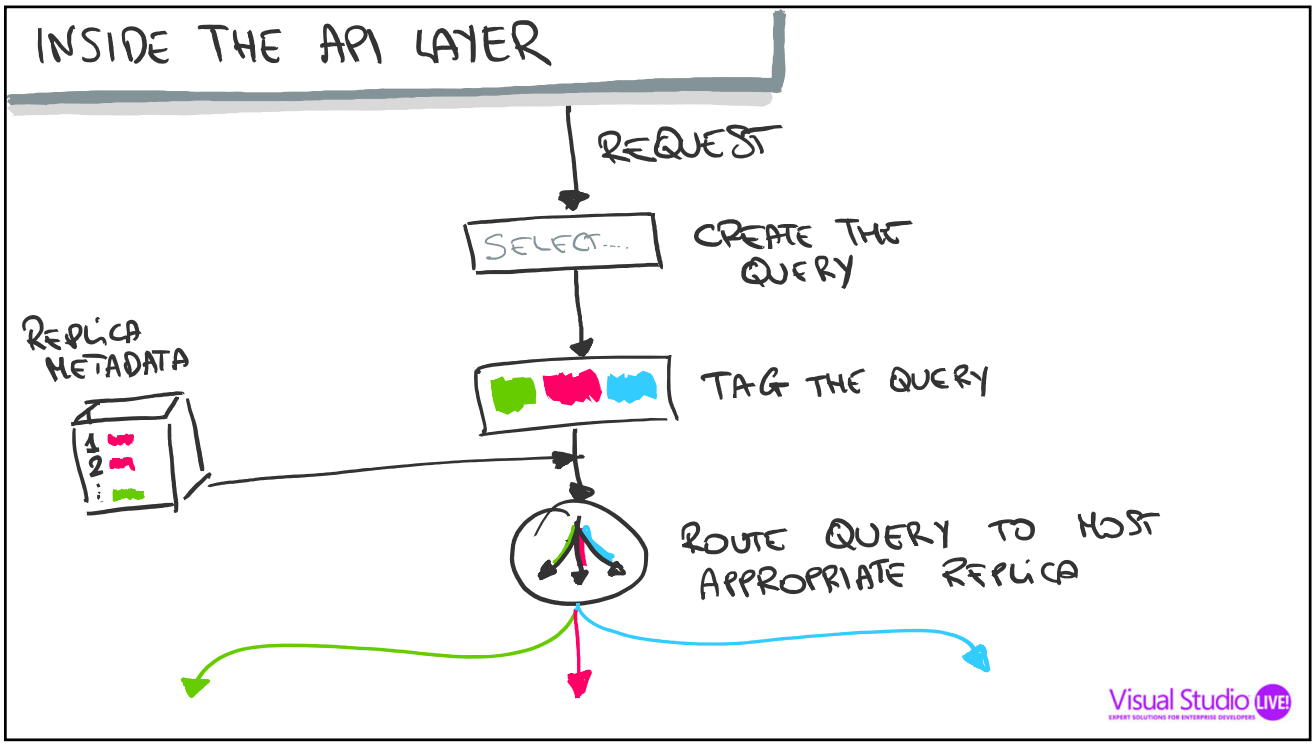
No other applications (other than ETL/Analytics) are allowed to access the database directly

- Directly = bypassing the API layer

In a microservice architecture the API layer can be a service itself!

- There could be a “routing” service





Database Scaleoutability in action!

DEMO

Visual Studio LIVE!

Results?

Pretty easy idea that actually works very well!

- Extremely easy to scale out the database

You can start from just one primary replica and add more as soon as more resources are needed

- Each replica can be set to handle one (or more) specific tag
- Workload will be distributed among all replicas able to handle a tag (for example: Reporting)

Benefits?

Adding or removing a replica doesn't require any disconnection

Adding and removing a replica is something that can be done in a dozen of seconds

- So the solution is quite elastic

Costs?

Secondary replicas (so any Named Replicas) are priced at a discounted rate!

Hyperscale

Hyperscale provides rapid, dynamic storage scaling up to 100 TB to help you optimize database resources for your workload's needs. Each additional read replica is billed at the Azure Hybrid Benefit pricing, which can be combined with discounted reservation pricing for greater savings when committing upfront to a 1 or 3-year term. The below pricing is for the primary instance.

vCORE	Memory (GB)	Pay as you go	Azure Hybrid Benefit[1] Price	1 year reserved capacity [2]	3 year reserved capacity [2]	3 years reserved with Azure Hybrid Benefit [2]
2	10.2	$412.64/month	$266.69/month ~35% savings	$319.29/month ~23% savings	$265.95/month ~36% savings	$120.00/month ~71% savings
4	20.4	$825.27/month	$533.37/month ~35% savings	$638.57/month ~23% savings	$531.90/month ~36% savings	$240.00/month ~71% savings
6	30.6	$1,237.91/month	$800.06/month ~35% savings	$957.85/month ~23% savings	$797.85/month ~36% savings	$360.00/month ~71% savings
8	40.8	$1,650.54/month	$1,066.74/month ~35% savings	$1,277.13/month ~23% savings	$1,063.80/month ~36% savings	$479.99/month ~71% savings

Visual Studio LIVE!

Conclusions

Taking advantage of database scaleoutability *is* a win-win

- More scalability and performances, less costs

Azure SQL DB Hyperscale is a cloud-optimized database that will enable you to enjoy database Scaleoutability

Visual Studio LIVE!

Conclusions

At least: Architect your application so that they can manage more than one database connection string

- One for reading & writing
- One for read only

Better: create an API layer that knows which nodes are available and orchestrate access to nodes

Best: tag queries and spread the workload to most appropriate nodes

Ultra: use ML to create and destroy replicas at the right time to optimize costs

Visual Studio LIVE!

Wait, where's the code?

Azure SQL Database Hyperscale Named Replicas OLTP Scale-Out Sample

09/10/2021 •

Browse code

license MIT

This sample shows how you can use Azure SQL Database Hyperscale Named Replicas to take advantage of ability to scale-out the database in an OLTP solution.

Scenario

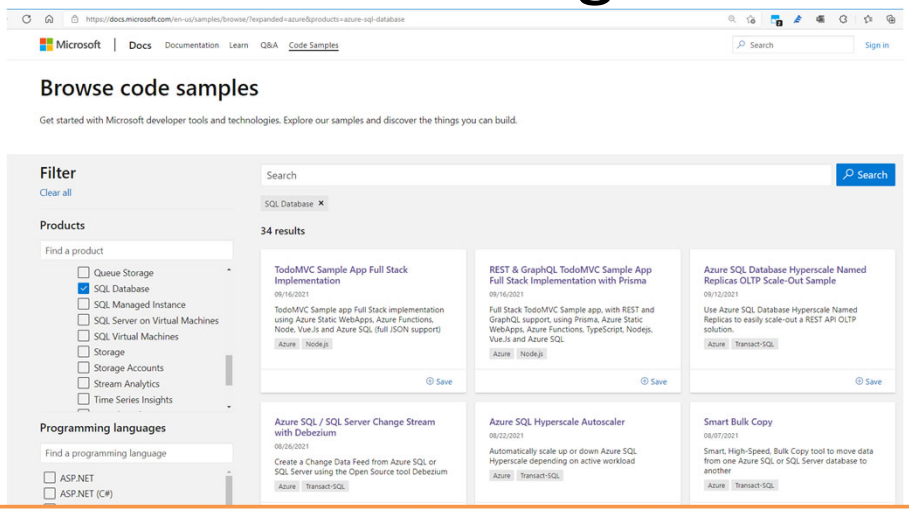
The code in the `./app` folder provide a REST endpoint that implements a basic shopping cart API. The REST endpoint has three methods:

- `GET /{id}`: return a JSON document containing a user shopping cart
- `GET /package/{id}`: return all the shopping carts that contains a package with the specified `id` value
- `PUT /`: store the received JSON document containing a user shopping cart

https://docs.microsoft.com/en-us/samples/azure-samples/azure-sql-db-named-replica-oltp-scaleout/azure-sql-db-named-replica-oltp-scaleout/

tudio LIVE!

And for even more goodness



https://docs.microsoft.com/en-us/samples/browse/?expanded=azure&products=azure-sql-database

References

GitHub Repo

https://github.com/Azure-Samples/azure-sql-db-named-replica-oltp-scaleout

Practical Azure SQL Database for Modern Developers: Building Applications in the Microsoft Cloud

https://www.amazon.com/Practical-Azure-Database-Modern-Developers-ebook/dp/B08MVYSLWL

Awesome Azure SQL

https://github.com/yorek/awesome-azure-sql

Azure SQL Samples & Best Practices

https://github.com/yorek/azure-sql-db-samples

Thanks!

Visual Studio LIVE!
EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS