Azure SQL Database

SQL Database Service Tier Selection

During tier selection, we try to identify the amount of services we require while trying to balance the cost.

As of now, there are 4 tiers available:

Basic

Standard

Premium

Premium RS

A given tier supplies X amount of resources such as max. database size, max. storage elastic pool, max. databases per pool, backup retention.

Max. storage in an elastic pool is where you can combine multiple databases in a flexible resource pool. So resources of a database with low usage can be allocated to a database with high usage.

A given tier supplies Y amount of performance such as Max DTUs, Max concurrent workers, Max. concurrent logins, Max. concurrent sessions.

Common Service Tier Use Cases

Use cases for each tier available in Azure SQL Database:

Basic Service tier –

Small database

Single active operation

Ideal for development and testing

Standard Service tier –

Support for multiple queries

Low – medium IO requirements

Ideal for cloud applications

Ideal for web applications

If the application starts underperforming on the DB part then move up a tier.

Premium tier –

High transaction vol & high performance

Supporting multiple users

Ideal for mission-critical operations where uptime needs to be maximized

Premium RS –

High transaction volume

Used where highest availability is not critical

Ideal for high-performance & analytical workloads

Because with analytical workloads, the thinking is that the sources of data can be reconstructed or re-accessed even if the data is lost as they are derived from other sources and fed into the data warehouse. We use Premium RS if we aren’t concerned about high availability but still want high performance.

Elastic pools

If we are implementing multiple DBs we have the option to use elastic pools to:

Manage & scale multiple DBs

Useful when demand on DBs are unpredictable

Cost-effective solution

DBs reside on a single server & share fixed resources. The costs are fixed as well & each DB configured in the pool are shared with other DBs & when DB A is over-utilized it can borrow from DB B. This allows us to level the peaks off.

To create elastic pools:

Azure portal

PowerShell

Transact-SQL

Resources of a database are measure in DTUs – Database Throughput Units

Resources of an elastic pool are measured in eDTUs – elastic Database Transaction Units

Features of elastic pools:

Autoscaling flexibility

Consumption of eDTUs adjusts to meet demand automatically

No database downtime required to scale up/down

Databases can be actively added or removed from an elastic pool

Database suitability (whether or not it should be in the pool) depends on activity/inactivity patterns. However a good rule of thumb is to put DBs that exhibit peaks & ebbs in activity levels.

Shared resources simplify management tasks. The resources are allocated dynamically.

Expenditures are predictable. Since we allocate a particular amount of resources to an elastic pool. Even if there are multiple DBs in the elastic pool, the expenditures are still predictable.

Scaling out with Elastic database tools

You might need to scale at some point to deal with increased workload, different configs. Etc.

We can do so using elastic database tools.

Some tools are:

Elastic database client library – Allows to create & maintain sharded databases.

Sharded database are another term for horizontal partitioning i.e., dividing the DB in some manner

Elastic database split merge tool – Moves data between sharded DBs

Elastic database jobs – Manage large numbers of Azure SQL databases, do administration etc.

Elastic database query – allows you to run tSQL queries that span multiple databases. This is then used to report using tools like PowerBI, Excel etc.

Elastic transactions – Allow you to run transactions that spans over several Azure SQL databases

Scaling

Horizontal a.k.a sharding – Division of a database into multiple different copies and different schemas, different servers, etc. it’s a way of ensuring data is isolated in some way, shape or form.

Vertical – Taking existing DB & changing edition up/down

Use cases for sharding

Large amount of data for a single database

Excessive workload transaction throughput

Physical isolation – Primary reasons. If there are multiple tenants in a single DB config. We need to ensure that each tenant is isolated from the others.

If you require varied geographic location for DBs

Custom DB organization

Single & multi-tenant sharding – Create a single shard for a single tenant & multiple shards for multi-tenants

Tempdb files

Optimal tempdb performance recommendations

Implementing SIMPLE recovery model – Helps to improve overall performance because it automatically reclaims log space. Keeps the overall space requirements to a min.

Automatic file growth

Be mindful of FILEGROWTH increment & original size of files. Overall size depends on what’s happening in the environment. The growth increment should remaining fairly consistent to upto 10% of original file size.

Do space preallocation – So that it doesn’t have to autogrow every time the service starts

Maximize bandwidth – Disk bandwidth. General recommendation is to allocate 1 file per processor. Not a bad idea to keep multiple files on multiple disks especially your best performing disks.

Standard file size – Keeps things consistent & reduces contention by reading from one location & writing to another

Tempdb database location

Connecting to Azure SQL database

There are limited options available in Azure SQL dashboard and most SQL operators love using SQL Server studio. We can connect Azure SQL to SQL Server Studio.

Adding a client

To add a new client or range of servers to Azure SQL:

Click on set server firewall option

Add the IP address of the client to the list of firewall rules

To add a range of servers, add the start & end IP to the list of firewall rules.

Now open SQL Server Studio, Connect > Database Engine. Enter server name as listed in Azure SQL dashboard (in overview page). Enter login & password > Click connect button.

Now the DB gets connected to SQL Server Studio & we can now easily manage the DBs.