IDERA® | Azure SQL Database Whitepaper

AZURE SQL DATABASE ARCHITECTURE

BY **HERVE ROGGERO**

SUMMARY

Although most developers and administrators can use Azure SQL Database (SQL Database) without understanding its underpinnings, it can be useful to study some of its underlying architecture to better understand how to manage your databases and how to adopt this technology properly in your custom application development initiatives. This white paper gives you a glimpse into the internal architecture of SQL Database and brings to light to some of the key concepts that SQL Database is built on.

INTRODUCTION

This white paper introduces you to Azure SQL Database, the Platform as a Service database service offered on the Microsoft Azure cloud. Originally called SQL Azure, Azure SQL Database has evolved significantly over the years and has become a robust and highly available database platform wrapped as a service offering with specialized security features.

If you have ever used SQL Database you already know how simple it is to use this relational database platform. Creating a database instance in SQL Database is as simple as pushing a few buttons, and within minutes you can have a fully supported, production-ready database infrastructure at your fingertips. Ser However you may wonder how this technology is put together; you may even ask yourself how Microsoft is able to guarantee high availability, or what the master database really is in a SQL Database, or even how the built-in connection firewall is implemented behind the scenes, both at the server and database level.

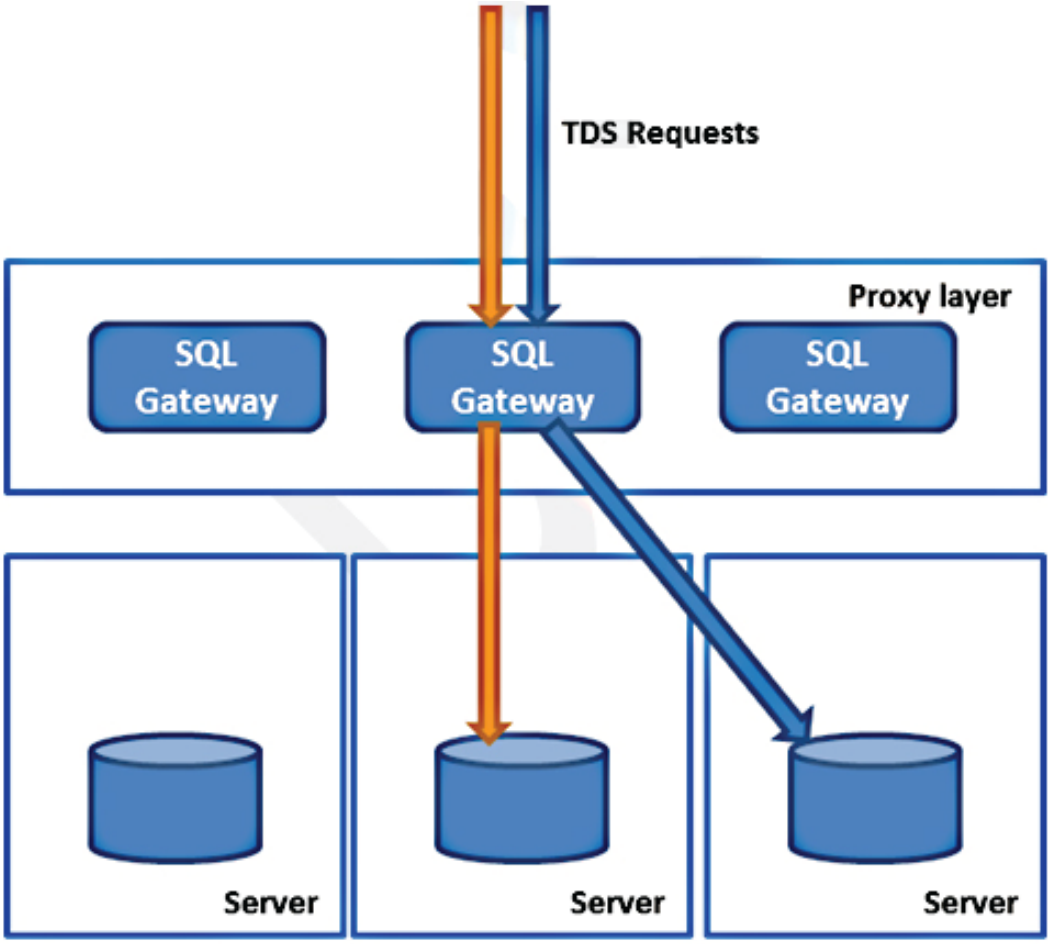
In order to gain a better understanding on how this infrastructure works, this white paper will give you a tour of SQL Database from an architecture point of view and provides a summary of the key features of the database platform.

Note that Azure SQL Database was previously called SQL Azure. What was previously known as a SQL Azure database is now called a SQL Database instance.

SQL DATABASE TOPOLOGY

Let's start with the underlying infrastructure. SQL Database is made up up of a layer of routers, firewalls, servers and services that together provide a unique database service. When a connection request is sent to a SQL Database, it is actually handled by a proxy layer through a series of gateways that perform login validation, enforce security constraints such as firewall rules and denial of service attacks, and additional services like billing and provisioning.

Once the TDS (Tabular Data Stream) request is validated by the proxy layer, it is forwarded to the server that contains the database initially requested. So, the client sending the original request is not communicating directly with the server holding the database; it is going through the proxy layer that determines dynamically where the database is actually located at the time of the request by looking up an internal mapping table. This dynamic database routing mechanism allows the SQL Database infrastructure to move the database instances on other servers at any time to account for hardware failures and load distribution.



Simplified SQL Database Topology

ABOUT MASTER

Because your SQL Database instances are not stored on the same machine, you may wonder where your master database resides and what you can do with it. The master database is in fact a virtual database; it is handled by the Gateway layer where the general SQL Database Server security settings are defined. This explains why the master database is read-only in SQL Database. This also explains why you need to connect to the master database (i.e. the Gateway layer) in order to create additional databases.

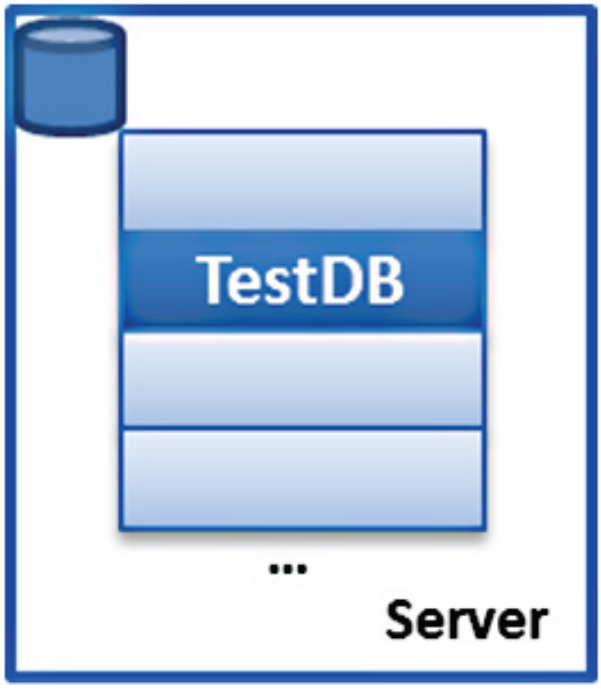
The server-wide firewall settings of SQL Database are also stored in master, at the Gateway layer. This also explains why changing firewall settings takes effect relatively quickly, because the Gateway layer is responsible for enforcing the firewall rules as well.

Last but not least, the Gateway layer also handles database provisioning, stores audit logs, and keeps summary information of bandwidth usage. As a result, the billing details are also provided by the Gateway layer and can be visible by querying the master database.

ABOUT SQL DATABASE INSTANCES

The current version of SQL Database is V12, which is a special build of SQL Server 2016. When you create a new server, a DNS entry is made so that the server is available from the Internet (once the firewall is configured), and new databases can be created. Each SQL Database, referred to as a database instance, is a contained SQL Server database and created on a server. This means that you can create users at the database level directly, making them more portable. These instances are running in a shared hosting model, with databases from other Azure tenants; while you are essentially sharing the hardware with other tenants, each database instance is strongly isolated from a security standpoint.

Because SQL Database instances are replicated internally for high availability, some of the partitions also hold replicated instances. In the following figure you see a representation of a SQL Database instance called TestDB; this instance is in fact a contained database.



SQL Database instance as a partition within a SQL Server database

This architecture has a few implications from a performance and management standpoint. The TestDB database instance shares system stored procedures and internal tables with other database instances on the same SQL Server database. As a result, SQL Database limits access to sensitive internal system objects for security reasons preventing administrators from executing certain maintenance commands. However, many system stored procedures and dynamic management views (DMV) are available, allowing you to troubleshoot certain performance problems on a SQL Database instance.

The following lists the DMVs currently available:

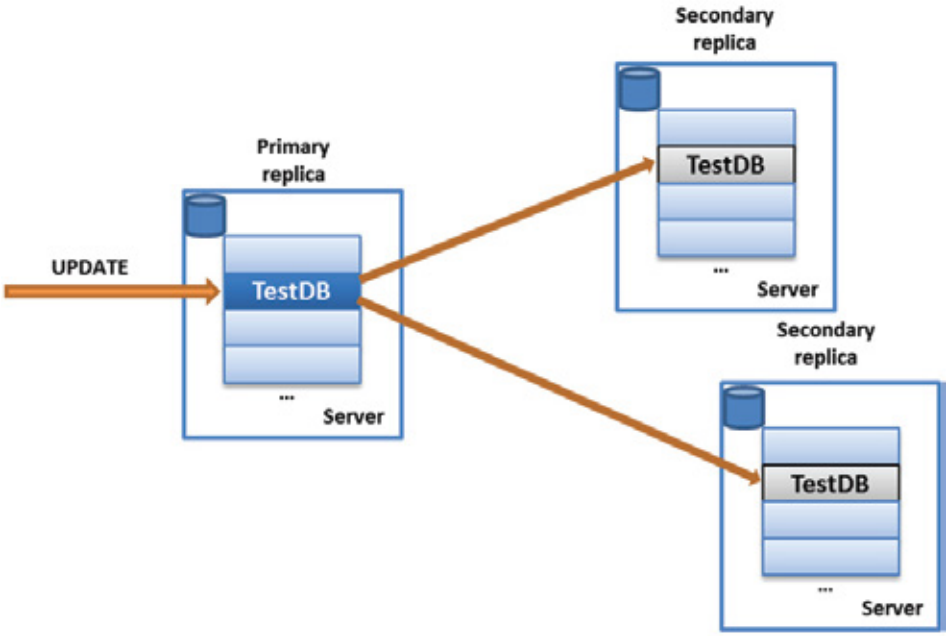
- sys.dm_tran_active_transactions
- sys.dm_tran_database_transactions
- sys.dm_tran_locks
- sys.dm_tran_session_transactions
- sys.dm_exec_connections
- sys.dm_exec_query_plan
- sys.dm_exec_query_stats
- sys.dm_exec_requests
- sys.dm_exec_sessions
- sys.dm_exec_sql_text
- sys.dm_exec_text_query_plan
- sys.dm_db_partition_stats

LOAD BALANCER

The Gateway service also uses a load balancing mechanism that periodically evaluates the current load on the primary replica. The Gateway has the authority to upgrade a secondary replica as the new primary replica. This continuous shift in primary replica promotion allows SQL Database to respond to increases in workloads quickly and to better distribute server resources across the pool of servers available.

HIGH AVAILABILITY

SQL Database instances are copied to other partitions for high redundancy and availability. Each SQL Database instance, including its copies, is called a replica. The primary replica is the database instance you connect to and execute statements against. There are two secondary replicas for redundancy. When a database commit is issued against the primary replica, at least one of the other two secondary replicas must also confirm the commit operation before the transaction is considered committed. This is referred to as a quorum commit.



SQL Database's quorum Commit

This replication architecture ensures 99.99% connection availability and is designed to fail over quickly to a secondary replica if the primary replica fails. The failover could take place for multiple reasons including: a failure on the primary replica, a failure on the server itself, or a failure on the rack that holds the server. The failover could also happen for other reasons that are not considered hard failures, such as during an upgrade of the SQL Database environment. The failover process could take a few seconds, which will disconnect clients with active sessions to the primary replica at the time of the failure and prevent new connections until the new replica becomes available.

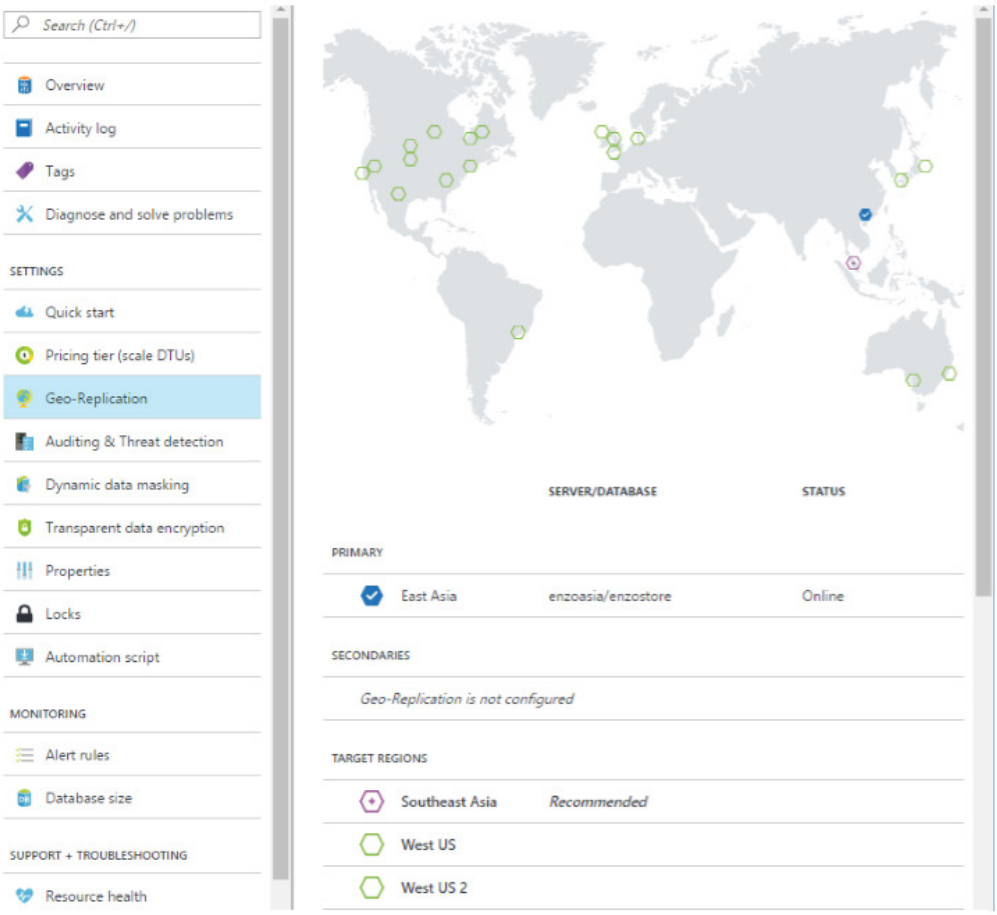
BACKUP AND GEO REPLICATION

SQL Database offers advanced backup and geo-replication capabilities; geo replication is available for both backup recovery and asynchronous transactional replication. Point in time restores allows you to restore a database automatically backed up by Azure within 7 to 35 days depending on your database tier (a database tier is a service level you choose when creating a database). You also have the option to use long-term retention for your backups for up to 10 years.

In addition to point in time restores, SQL Database provides the ability to geo replicate backup files and replicate transactions to multiple databases asynchronously. Geo-replicated backup files are managed by Microsoft directly; the backup files are stored as blob storage in the paired database center. Paired data centers are controlled by Microsoft as well; for example the US East is paired with US West.

When using transactional replication, you can define up to four read-only destination databases in any of the Azure regions. In other words you can connect to any replicated database and issue SQL commands to access data. You can choose to fail over to a replicated database and make it the new primary database.

In this screenshot, you see a database created in East Asia, called enzostore, with the recommended target region for Geo-Replication being Southeast Asia. However you can choose other data centers to replicate the database closer to your customers in various geographic regions.



SQL DATABASE FIREWALL

As mentioned previously, SQL Database implements a firewall mechanism to help you limit access to your database instances. The firewall is designed to keep a list of allowed IP ranges for which database credentials can be further authorized by SQL Database. In other words, if a client connects outside of the allowed IP ranges, the connection request will be denied even if the credentials are valid.

You can manage two sets of firewall rules: server-level and database-level. Server-level rules are evaluated first; database rules are evaluated if the server-level rules fail. A few stored procedures and system views are available to manage the firewall rules. For example you use the sys.firewall_rules view to list the current rules. To manage the rules you use the sp_set_firewall_rule and sp_delete_firewall_rule stored procedures to add and delete firewall rules respectively. You may need to consider the following regarding setting up firewall rules with SQL Database:

- SQL Database is secured by default. You must enable the desired IP ranges before establishing a database connection.
- The Azure management portal allows you to configure the IP range rules.
- The IP ranges defined in SQL Database only work for public IP Addresses. If you are trying to connect from a machine that is connected to the Internet through NAT (Network Address Translation) you will need to first determine its public IP Address.

AUTHENTICATION AND AUTHORIZATION

SQL Database provides similar levels of user authentication and authorization to SQL Server 2016 by allowing you to manage logins and database-level users, either as traditional server-scoped logins or the more recent contained database login model. If you have used SQL Database before, you might be interested to learn that it now supports Azure Active Directory Authentication. When using a server-scoped login, you first need to create the login on the server (master) and the corresponding user in the desired database(s). When using the contained database login model, you only need to create the user in the desired database(s).

ENCRYPTION, AUDITING, DATA MASKING, COMPLIANCE

In addition to the aforementioned firewall configuration options and user authentication and authorization security measures, SQL Database offers optional disk-level encryption at rest, auditing, data masking, and compliance certifications for more sensitive applications

When enabled, the disk-level encryption feature (TDE) allows you to specify that all data should be encrypted at rest, including database backups. While TDE provides a significant security capability, you may still want to mask data access to certain fields. For example you may want to mask data for a Social Security Number. Data masking allows you to define masking rules for specific login accounts.

From an auditing standpoint, you can easily integrate with Microsoft Power BI to offer drill down analysis of your logs. You can also configure email alerts of unexpected activities, such as SQL Injection detection, that can be sent to one or more email addresses. You can obtain additional information on the portal, such as the source IP Address and the actual T-SQL statement that caused the alert.

RESOURCE LIMITS AND THROTTLING

Throttling is a term used to describe a usage boundary in SQL Database or other cloud resources in Azure that are enforced by delayed execution of queries or through a loss of connection. In SQL Database specifically, throttling is enforced on the database server to prevent heavy loads from affecting other database tenants on the same server. If your query consumes too many resources, such as CPU, I/O, or memory, your query may be delayed and executed when the resources become available.

If your system exceeds one of the limits for your database tier, for example you attempt to create more than 300 concurrent sessions on a basic service tier, throttling may kick in and your application may receive an error.

The set of rules used to determine whether throttling will take place and to which degree can be complex and may change over time. In addition throttling can have direct implications on coding practices and may affect how you design certain database components. For more information about throttling and general database connection management, check SQL Database Resource Limits.

UNDERSTANDING BILLING

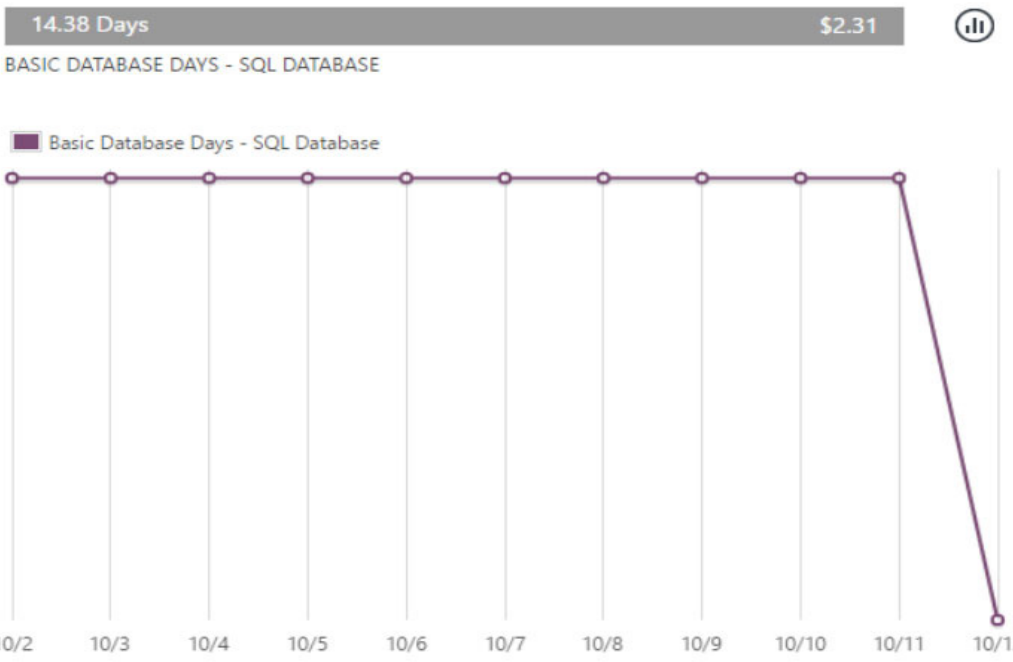
We discussed earlier that each database instance has two secondary replicas. You are charged for the primary replica only. The secondary replicas are invisible to you and, as such, are not included in your service fees. However you will be charged for databases used for geo-replication.

When you create a database server, you can choose to create a single database at a time, or an elastic pool. An elastic pool is a logical container that allows you to create many databases for which performance and billing is managed at the pool level. Elastic pools allow you to control the overall allocation of CPU/IO/Memory resources across your databases.

The charge for database usage is measured in Database Transaction Units (DTUs) for single databases or elastic DTUs (eDTUs) for elastic pools. Databases are created in a performance tier (Basic, Standard, Premium), allowing you to choose the number of DTUs or eDTUs available to your databases.

Your invoice contains a summary of the charges consumed by your Azure services, including databases and data transfer. The entry level cost (basic tier) for the first DTU is $0.0067 hourly at the time of this writing, which is roughly equivalent to $4.99 per month. You should note that pricing depends on the tier your database belongs to, and the geographic region.

The following diagram shows that my account is being charged about 0.16 DU per day and has an accumulated cost of $2.31 as of 10/12.



CONCLUSION

This whitepaper introduced you to important concepts related to Azure SQL Database and explains how some of the underlying services provide high availability, security and replication. Because SQL Database is built as a shared infrastructure service, in which your databases may reside alongside other customers' databases, and because SQL Database requires a flexible self-service architecture, some of the traditional database concepts are implemented differently than on SQL Server, such as the master database. In addition, SQL Database imposes performance constraints that vary based on your service level, such as throttling, to ensure a fair repartition of resources between tenants. Last but not least, it is important to note that SQL Database, like any other service offered on the Microsoft Azure cloud, evolves rapidly with frequent updates offering enhanced management, development and operational capabilities. As a result, check back often on the Azure portal to see how SQL Database evolves over time.

ABOUT IDERA

IDERA understands that IT doesn't run on the network – it runs on the data and databases that power your business. That's why we design our products with the database as the nucleus of your IT universe.

Our database lifecycle management solutions allow database and IT professionals to design, monitor and manage data systems with complete confidence, whether in the cloud or on-premises.

We offer a diverse portfolio of free tools and educational resources to help you do more with less while giving you the knowledge to deliver even more than you did yesterday.

Whatever your need, IDERA has a solution.