

# Advanced Database Project Report

Cloud Databases and Microsoft Azure

INFO-H-415

18.12.2017



Marie Elisabeth Heinrich – 000457502

Jayanthi Kambayatughar – 000457113

# Table of Contents

- 1 Introduction to Microsoft Azure and Azure SQL Cloud Database ..... 5
  - 1.1 Motivation ..... 5
  - 1.2 MS Azure Platform ..... 6
    - 1.2.1 Azure Cloud Computing Concept and Platform Architecture ..... 6
  - 1.3 Cloud Databases on MS Azure ..... 9
    - 1.3.1 General concept of Cloud Databases ..... 9
    - 1.3.2 Database as a Service (DBaaS) on MS Azure ..... 10
    - 1.3.3 Available Cloud Databases on MS Azure ..... 12
  - 1.4 MS Azure SQL Cloud DB ..... 13
    - 1.4.1 General introduction ..... 13
    - 1.4.2 Typical Use Cases ..... 15
    - 1.4.3 Evaluation of Azure SQL DB ..... 15
  - 1.5 Summarized Advantages and Disadvantages of Azure SQL ..... 16
- 2 Performance benchmark for a real-life application example of a retail sales management process 17
  - 2.1 Introduction to Use Case & Benchmark ..... 17
  - 2.2 Generation of test data ..... 17
  - 2.3 Specifications for test environment ..... 17
  - 2.4 Database creation ..... 18
    - 2.4.1 Create Database in Azure SQL ..... 18
  - 2.5 Querying Azure SQL database ..... 23
    - 2.5.1 Select sales data ..... 23
    - 2.5.2 Update product prices during promotions ..... 23
    - 2.5.3 Calculate total sales amount ..... 23
    - 2.5.4 Insert new Sales Records ..... 23
  - 2.6 Performance Benchmark ..... 25
    - 2.6.1 Performance ..... 25
    - 2.6.2 Database Cost Comparison ..... 33
    - 2.6.3 Further considerations for database evaluation ..... 35
- 3 Final conclusion ..... 35
- 4 Sources ..... 36

## List of figures

- Figure 1: Magic Quadrant for Cloud Infrastructure as a Service<sup>11</sup> ..... 5
- Figure 2: Architectural overview IaaS, PaaS and SaaS<sup>11</sup> ..... 6
- Figure 3: Microsoft Azure Platform Architecture ..... 8
- Figure 3: Overview service tiers ..... 11
- Figure 5: Azure Pricing Options<sup>7</sup> ..... 11
- Figure 5: Azure Pricing Calculator<sup>12</sup> ..... 12
- Figure 7: Relational database ranking statistics as of December 2017<sup>8</sup> ..... 13
- Figure 8: Azure Data Migration and Connection Workflow<sup>9</sup> ..... 14
- Figure 9: Physical distribution and application hierarchies..... 14
- Figure 10: Create an Azure SQL database on the MS Azure platform..... 18
- Figure 11: Selection of Service Tier for the Azure SQL database ..... 19
- Figure 12: Set Server firewall menu ..... 19
- Figure 13: Add Client IP to the Server Firewall..... 19
- Figure 14: Setting up a connection from SSMS to Azure..... 20
- Figure 15: T-SQL statement to create database schema..... 21
- Figure 16: bcp command-line statement to load data into the database..... 22
- Figure 17: Stored Procedure to insert sales records ..... 24
- Figure 18: Performance Test Results Select query 1.000.000 rows ..... 26
- Figure 19: Estimated memory allocation of SQL Azure per service tier<sup>10</sup> ..... 27
- Figure 20: Performance Test Results Join query 1.000.000 rows ..... 28
- Figure 21: Performance Test Results Select and Update query 1.000.000 rows..... 29
- Figure 22: Azure SQL Query Performance Insights ..... 30
- Figure 23: Write Rate Comparison (MB/min) of different Azure SQL Service Tiers<sup>10</sup> ..... 31
- Figure 24: Summary of average performance test results ..... 32
- Figure 25: Difference between Azure SQL and SQL Server Performance Test Results in % ..... 32

## List of tables

Table 1: Advantages and Disadvantages of Cloud Databases ..... 9

Table 2: Overview Cloud Databases on Azure..... 12

Table 3: Summarized Advantages and Disadvantages of Microsoft Azure ..... 16

Table 4: Monthly cost estimation on premise SQL Server ..... 33

Table 5: Monthly cost estimation Azure SQL Database ..... 34

# 1 Introduction to Microsoft Azure and Azure SQL Cloud Database

Cloud databases and Microsoft Azure are both themselves large topics. To summarize and examine the most important factors to consider, this report shall first give a top-down introduction from Cloud Computing, to Microsoft Azure and then to the Azure SQL cloud database in section 1. Second, a real-life OLTP application was implemented to conduct a performance benchmark study between a cloud database (Azure SQL) and a standard on premise solution (SQL Server) in section 2. The obtained results are then compared and discussed in section 2.6.

## 1.1 Motivation

Cloud Computing has gained increasing attractiveness in recent years. Research says that 24% of the total addressable IT market will be in the cloud by 2020 and almost one out of five virtual machines are already running in the cloud with a drastic market and offering growth.<sup>1</sup> According to Gartner’s Magic Quadrant for Cloud Infrastructure as a Service (IaaS) that was published in June 2017 (refer to figure 1), there are currently two leaders in the market: Amazon Web Services (AWS) and Microsoft with Microsoft Azure (MS Azure). But what are the reasons for this trend, what makes these cloud infrastructures so outstanding and why should a cloud database be considered?



Figure 1: Magic Quadrant for Cloud Infrastructure as a Service<sup>11</sup>

When it comes to traditional databases they have some major restrictions in common: their underlying hardware restrictions and limited scalability in terms of CPU and memory size. As a steadily growing amount of computing power is needed to tackle the challenges that come along with massive amounts of data, cloud databases quickly raised more interest due to the fact that they can be scaled to the required computing power on demand. This trend does affect most of today’s industries, from banking and assurance that have always dealt with huge datasets but also engineering or healthcare industries that see themselves confronted with the Internet of Things or advanced image recognition technologies, which are disrupting their current business with new business models and technologies based on data.

Furthermore, Cloud Computing is not only beneficial for companies that have large variations in the workload of their database but also for those that are growing rapidly and need systems that can cope with the growth. Additionally, many cloud databases are offered “as a Service” which shifts considerations about database maintenance and set up away from the user and thus can support businesses in bridging human resource shortages and eliminates the need for huge upfront investments in IT-infrastructure.

However, various factors need to be taken into account when evaluating the use of cloud solutions. Therefore, we developed a use case and conducted a small benchmarking study in order to get familiar with the cloud environment and explore highlights and drawbacks of a cloud database.

## 1.2 MS Azure Platform

This chapter shall give a brief general introduction to Cloud Computing and highlight characteristics of cloud platforms and databases, in particular of MS Azure and Azure SQL.

### 1.2.1 Azure Cloud Computing Concept and Platform Architecture

MS Azure is not a single cloud database itself. Rather, it can be seen as an enterprise-ready platform with a comprehensive set of several cloud services that can be managed over a network of globally distributed data centers, stretching from US and Canada all over China and South-Asia as well as several locations in Europe.

#### 1.2.1.1 Cloud infrastructure models

To understand the cloud service concept in more detail we will hereunder shortly introduce the most common cloud infrastructure models that can be deployed depending on the level of ownership that shall be kept:

- **Public Cloud:** All offered services – including hardware, software and infrastructure – can be accessed over a public network (e.g. internet) and are hosted and maintained by the respective cloud provider in a multi-tenant way.
- **Private Cloud:** The infrastructure is solely operated for a single organization in contrast to the multi-tenant offering in a public cloud. Thus, it offers more control over the client’s resources but still can leverage the advantages of cloud services.
- **Hybrid Cloud:** A secure, private connection between on premise solutions or a private cloud and the public cloud is set up in order to strategically extend capabilities with cloud services.<sup>2</sup>

Azure itself is set up as a public cloud, but also added private and hybrid capabilities and interfaces over the past years.

Other common terms used to differentiate cloud services are “Infrastructure as a Service” (IaaS), “Platform as a Service” (PaaS) and “Software as a Service” (SaaS). They also differ in the ownership of data, software and infrastructure as shown in figure 2.

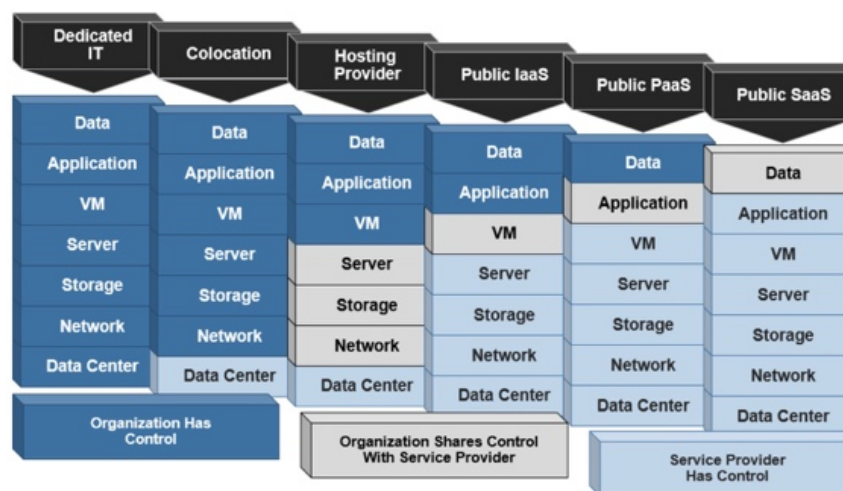


Figure 2: Architectural overview IaaS, PaaS and SaaS<sup>11</sup>

A typical Microsoft example for SaaS could be Office 365 that offers extends the software to the cloud. Azure itself focusses on PaaS and SaaS capabilities. Generally speaking, PaaS is targeting developers who don't want to care about underlying hardware or operating systems of their applications. An example for PaaS is Azure App Services. IaaS in contrast focusses on system administrators and offers a higher degree of the company's ownership but a lower degree of agility compared to PaaS.

### ***1.2.1.2 Cloud Computing characteristics***

On the basis of these concepts there are a few common basic characteristics of Cloud Computing offerings, defined by the National Institute of Standards and Technology, that are incorporated by MS Azure and highlight potential advantages of using a cloud architecture:

- **Elasticity & self-service**
  - Dynamic adjustment of services “on the fly” to suit actual demands and save costs
  - Supports short-term demand peaks, e.g. in case of product launches or sales promotions or utilizing more VM's for testing purposes
- **Scalability**
  - Vertical and horizontal scalability of all services (e.g. database performance, number of utilized VM's etc.), supporting a long-term strategy by allowing growth of the IT infrastructure along with the growth of the corporation
- **Pooling**
  - Computing (CPU, RAM), storage and network services are pooled for different stakeholders within a company or amongst several corporates and can be scaled on tap without requiring every corporate to build their own infrastructure for each service.
- **Pay-per-Use concept**
  - Costs are calculated on a pay-per-use basis which allows maximal efficiency of resource utilization

### ***1.2.1.3 Azure platform architecture***

As stated in the beginning of the chapter, Azure is a combination of different service offerings. This implicates that there is no fixed architecture that applies for every use case. Instead one may think of it as a service-oriented architecture that depends on the use case and which may incorporate the following components:

- Identity as a Service to manage directories, e.g. Azure AD (Active Directory)
- Container as a Service to host and manage window server containers, Docker containers or orchestration engines using Azure Container Services
- Virtual Machines as a Service with multitenant storage options
- Application Insights as a Service: Telemetry systems that can be attached to cloud applications to monitor and get insights of the application or to perform analytics (e.g. Azure Machine Learning, IoT Suite)
- Storage as Service in form of a managed disks model, traditional storage accounts or Object Storage Offerings (Blob storage)
- Database as a Service: Azure offers its own databases like Azure SQL Database or the NoSQL DocumentDB as well as other database services like inApp MySQL, MonoDB or HDInsights
- Recovery as a Service: Azure Site Recovery provides an orchestration tool for disaster recovery<sup>3</sup>

## Advanced Database Project Report – Cloud Databases and Microsoft Azure

Please note that this list should not be seen as an exhaustive list. The full list of services can be found in the Product directory of Azure [here](#). As we are focusing on Azure SQL in this report, this shall only serve as a first introductory overview and only the database architecture of Azure SQL will be discussed further in the subsequent chapters. An overview of the most prominent provided services mentioned earlier is shown in figure 3:

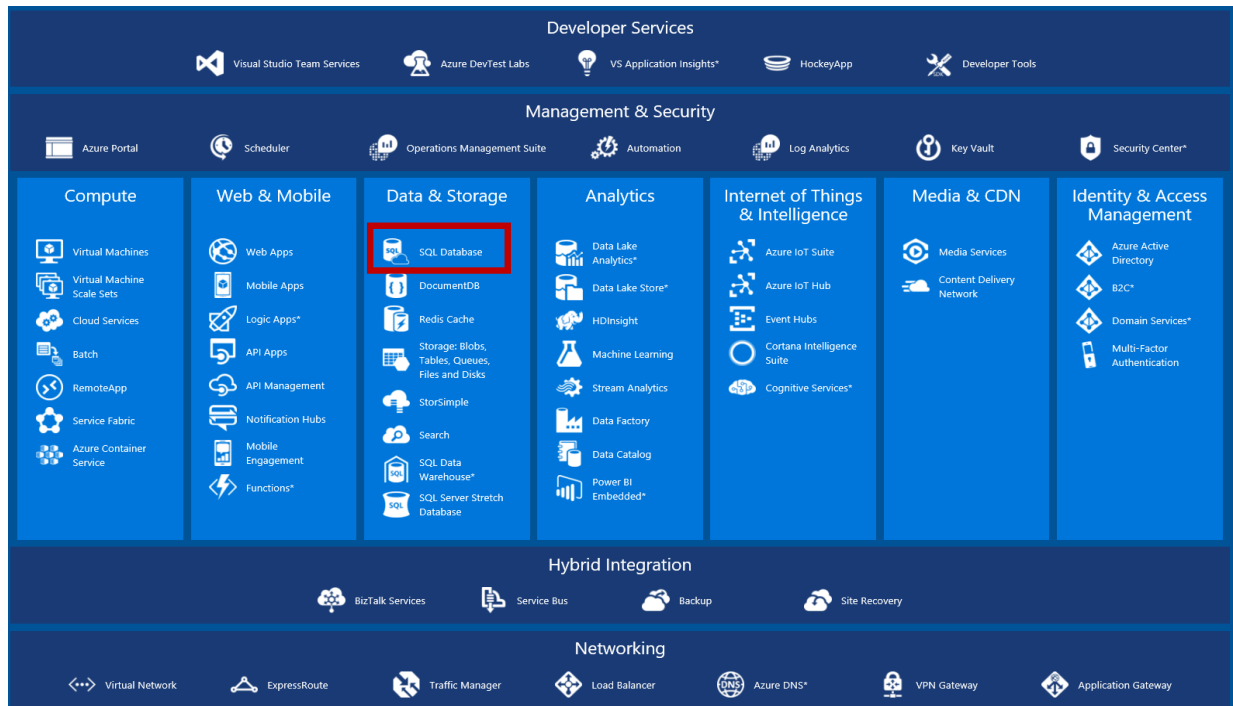


Figure 3: Microsoft Azure Platform Architecture



### 1.3 Cloud Databases on MS Azure

As we’ve seen in the last section, Cloud Computing comprises many different topics. Therefore, the next two sections shall guide us from the bigger scale, the overall platform, to one of its services (Cloud Databases, chapter 1.3.1ff) and then provide an in-depth analysis of one specific database: MS Azure SQL Database.

#### 1.3.1 General concept of Cloud Databases

A cloud database in general is a database that can be built on and accessed through a cloud computing infrastructure platform as discussed in chapter 1.2. This can be realized either through direct queries (e.g. SQL statements) or via API calls. From a structural and design perspective cloud databases are similar to the ones that are deployed on the corporate on premise servers and the database behavior to queries should be the same. The key differences are in the scalable performance and the location of residence of the database:

A cloud database can either be deployed on the company’s IT infrastructure via a virtual machine or as a Database as a Service model, where it is running on the Cloud provider’s infrastructure. This implicates especially for cloud databases of a larger scale that they can usually referred to as distributed databases. Even if the database may appear to the user as a single, coherent database, it is actually constructed out of a set of databases that can be distributed over several server locations. The database of a cloud provider can be accessed solely via the internet through a web interface or vendor-provided application interface (API) in contrast to an on premise database that is connected to local users through an internal local area network (LAN).<sup>4</sup>

The following table highlights the key advantages and disadvantages of cloud databases:

Advantages	Disadvantages
<ul style="list-style-type: none"> <li>• Simultaneous access and modification of data from multiple users using a network</li> <li>• No need to buy dedicated hardware (if database is offered as a service)</li> <li>• Can support SQL and NoSQL databases that</li> <li>• Reduced risk of data loss in case of a power outage as the data is distributed and mirrored over multiple locations</li> </ul>	<ul style="list-style-type: none"> <li>• May have a slightly slower response time than an on premise deployment due to the routing through the internet instead of an internal LAN</li> <li>• Computer Security is always a major concern and risk of Cloud Computing, thus, only trusted providers should be chosen</li> </ul>

*Table 1: Advantages and Disadvantages of Cloud Databases*

### 1.3.2 Database as a Service (DBaaS) on MS Azure

Based upon the above mentioned characteristics of cloud databases, Azure offers several services and pricing options for their databases that shall be discussed in the subsequent sections.

#### 1.3.2.1 Offered cloud database services

Using the services of a cloud database provider allows several benefits that are summarized hereinafter:

- Elimination of physical infrastructure: In a DBaaS environment, the service provider (Azure) is responsible for maintaining, setting up and operating the database software, leaving the DBaaS users' responsibility and attention only on their data
- Cost Savings due to reduced capital expenditures, decreased operating costs and less needed physical space. In addition, cloud database technology is becoming more mature and thus leads to a price decline.
- Instantaneous scalability "on the fly" without downtime: Azure can quickly offer additional fee-based capacity, throughput and access bandwidth in case of demand peaks or highly volatile demands. Furthermore, Azure provides so called elastic pools: Using a single database that is fully isolated and optimized for workloads is beneficial when performance is more or less predictable. Contrary, elastic pools can be used when facing volatile demands. Databases scraped together in one pool automatically scale up and down based on the demand. Then, the collective performance of a pool is managed rather than the one of a single database.
- Performance and availability guarantees: Through a service level agreement (SLA) Azure is obligated to provide performance guarantees (typically including quantifying minimum uptime availability and transaction response times). For the Azure SQL database, Microsoft ensures 99.99% availability in the SLA.
- Ease of access: Clients can access cloud databases from virtually everywhere through the internet and are not tied to a LAN anymore
- Latest technology & Security: To remain competitive, Azure is continuously keeping its infrastructure up to date with security and other feature updates
- Failover support & availability: Failover support typically encompasses the operation of multiple mirror image server and data storage facilities. If handled properly, data is kept secure through backups on remote servers in case of natural disasters, equipment failure or power outages. Additionally, Microsoft ensures 99.99% availability in their SQL database SLA
- Built-in intelligence: Integrated performance monitoring, alerting and tuning like automatic index management, adaptive query processing or intelligent threat detection to detect potentially harmful attempts to access sensitive data
- Security and Compliance: Starting in May 2017, all newly created Azure SQL databases are automatically protected with transparent data encryption, dynamic data masking and row-level security<sup>5</sup>

### 1.3.2.2 MS Azure Pricing Model

If using an Azure Database, a user only pays for what he or she needs. As the pricing is different for different databases, we shall focus on the pricing of Azure SQL used as a single database.

Pricing on Azure depends on the service tier, which again has different pricing levels. Billing is carried out per hourly usage of the database. There are four major services tiers, namely Basic, Standard, Premium and Premium RS, that differ in the number of Database Transaction Units (DTUs), included storage and maximum storage. Depending on the business application, utilization and resource restrictions, one service tier has to be chosen preliminary, but can be changed anytime at a later stage. Another factor is the availability that Azure commits to in the Service Level Agreement (SLA)<sup>6</sup>. An overview of included capacities and features of the pricing models is shown in figure 3.

#### Single databases

	Basic	Standard	Premium	Premium RS
Maximum storage size*	2 GB	1 TB	4 TB	1 TB
Maximum DTUs	5	3000	4000	1000

#### Elastic pools

	Basic	Standard	Premium	Premium RS
Maximum storage size per database*	2 GB	1 TB	1 TB	1 TB
Maximum storage size per pool*	156 GB	4 TB	4 TB	1 TB
Maximum eDTUs per database	5	3000	4000	1000
Maximum eDTUs per pool	1600	3000	4000	1000
Maximum number of databases per pool	500	500	100	100

Figure 4: Overview service tiers

	Basic	Standard	Premium	Premium RS
Target workload	Development and production	Development and production	Development and production	Workload that can tolerate data loss up to 5-minutes due to service failures
Uptime SLA	99.99%	99.99%	99.99%	N/A while in preview
Backup retention	7 days	35 days	35 days	35 days
CPU	Low	Low, Medium, High	Medium, High	Medium
IO throughput	Low	Medium	Order of magnitude higher than Standard	Same as Premium
IO latency	Higher than Premium	Higher than Premium	Lower than Basic and Standard	Same as Premium
Columnstore indexing and in-memory OLTP	N/A	N/A	Supported	Supported

Figure 5: Azure Pricing Options<sup>7</sup>

The measure for resource utilization is called Database Transaction Units or DTUs, and is a blended measure of CPU, memory and I/O (data and transaction log I/O), whose ratio was originally determined by an OLTP benchmark workload. If the workload exceeds the amount of any of these resources, the throughput is throttled. DTUs can be calculated using the [Azure DTU calculator](#).

As prices for single databases can range from 0.0057€/hour up to 18.14€/hour and for elastic pool from 0.0851€/hour up to 25.30€/hour, they won't be covered exhaustively here. Instead, please refer to the [this](#) website for pricing details of all service tiers.

Alternatively, Azure provides a pricing calculator for all its products to calculate the total cost of the cloud service set-up upfront (refer to figure 5) [here](#).

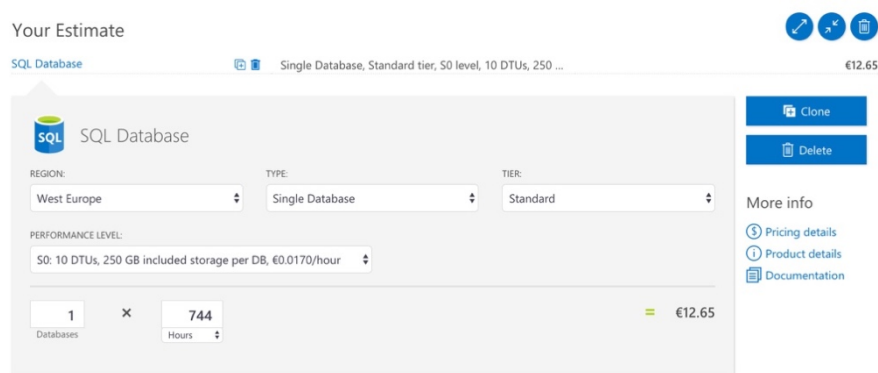


Figure 6: Azure Pricing Calculator<sup>12</sup>

### 1.3.3 Available Cloud Databases on MS Azure

Before going into in-depth analysis of the Azure SQL Azure database, the following table provides a brief overview of the currently offered cloud database options on MS Azure. In addition to that, Azure also provides individualized services for time series data (Time Series Insights), search engine databases (Azure Search), or object storage (Blob Storage) and offers integration of third-party solutions like MongoDB or open source frameworks like Hadoop through HDInsights.

Database Type	Azure Offerings
Relational Database Management Systems	Azure SQL Database Azure Database for MySQL and PostgreSQL SQL Server Stretch Database
Key/Value stores	Azure Cosmos DB Azure Redis Cache
Document & Graph Database	Azure Cosmos DB
Data Analytics	Azure Data Lake

Table 2: Overview Cloud Databases on Azure

## 1.4 MS Azure SQL Cloud DB

Coming from the overall Azure platform, this chapter shall now focus on one database available on the Azure platform: Azure SQL database.

### 1.4.1 General introduction

MS Azure SQL Database is a Relational Cloud Database provided as a DBaaS (Database as a Service) by Microsoft Azure. It was released in 2010 by Microsoft and runs on the Microsoft Azure cloud computing platform. It is a managed Database Service, which means that it covers scalability, backup and high availability of the database and can be queried using T-SQL or .net. It also includes built-in intelligence features which enables the database to learn query and workload patterns and tunes the database to maximize performance, reliability, and data protection. It currently holds rank 15 in the relational DB-engine ranking and rank 26 in the overall DB-engine ranking.

138 systems in ranking, December 2017

Rank			DBMS	Database Model	Score		
Dec 2017	Nov 2017	Dec 2016			Dec 2017	Nov 2017	Dec 2016
1.	1.	1.	Oracle +	Relational DBMS	1341.54	-18.51	-62.86
2.	2.	2.	MySQL +	Relational DBMS	1318.07	-3.96	-56.34
3.	3.	3.	Microsoft SQL Server +	Relational DBMS	1172.48	-42.59	-54.17
4.	4.	4.	PostgreSQL +	Relational DBMS	385.43	+5.51	+55.41
5.	5.	5.	DB2 +	Relational DBMS	189.58	-4.48	+5.24
6.	6.	6.	Microsoft Access	Relational DBMS	125.88	-7.43	+1.18
7.	7.	7.	SQLite +	Relational DBMS	115.19	+2.44	+4.36
8.	8.	8.	Teradata	Relational DBMS	74.74	-3.49	+1.37
9.	9.	9.	SAP Adaptive Server	Relational DBMS	65.68	-1.35	-4.74
10.	↑ 11.	↑ 13.	MariaDB +	Relational DBMS	56.73	+1.44	+12.64
11.	↓ 10.	↓ 10.	FileMaker	Relational DBMS	55.20	-3.64	+1.08
12.	12.	12.	Hive +	Relational DBMS	54.67	+1.42	+5.27
13.	13.	↓ 11.	SAP HANA +	Relational DBMS	46.49	-2.69	-5.28
14.	14.	14.	Informix	Relational DBMS	27.45	-0.26	+0.38
15.	15.	↑ 16.	Microsoft Azure SQL Database	Relational DBMS	21.85	-0.26	+1.02
16.	16.	↓ 15.	Vertica +	Relational DBMS	20.81	-0.84	-0.28
17.	↑ 18.	↑ 18.	Firebird	Relational DBMS	17.25	-0.02	+1.57
18.	↓ 17.	↓ 17.	Netezza	Relational DBMS	17.24	-1.29	-2.19
19.	19.	↑ 20.	Impala	Relational DBMS	14.46	+0.61	+4.77
20.	20.	↓ 19.	Amazon Redshift +	Relational DBMS	12.58	+0.02	+0.12

Figure 7: Relational database ranking statistics as of December 2017<sup>8</sup>

MS Azure SQL Database shares a common code base with SQL Server. At the database level, it supports most of the features of SQL Server and is compatible with SQL Server 2014 and 2016. The major feature differences between Azure SQL Database and SQL Server are at the instance level. Since 2015 it can not only be configured as a single database, but also within an elastic pool. This elastic pool logically groups several databases into one pool, in which they can share resources. This allows to make use of currently unused resources of other databases in the pool via an exchange of eDTUs that have a similar concept to the normal DTUs.

Figure 8 shows a typical workflow from the source to the consumption of Azure SQL. Structured, semi-structured or unstructured data can be stored in the database in the cloud and from there consumed by Business Intelligence Apps like Microsoft PowerBI or other web applications.

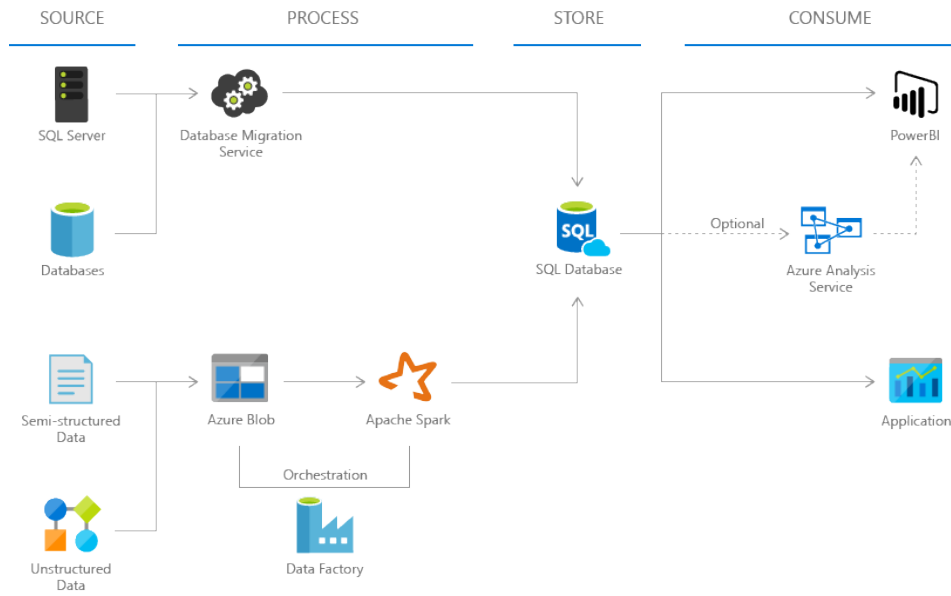


Figure 8: Azure Data Migration and Connection Workflow<sup>9</sup>

Coming from an application perspective, the hierarchy of Azure SQL can be described as shown in the following figure:

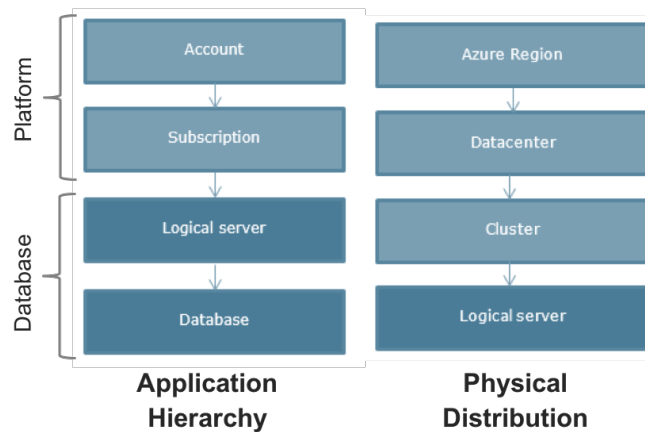


Figure 9: Physical distribution and application hierarchies

Via the Azure platform, a database user creates an account and adds a subscription for one of the service tiers. All created SQL databases are deployed on one or several servers. The Azure servers actually aren't physical servers or machines. Instead, they serve as a logical collection of databases that are sharing the same user directories, firewall settings and diverse other general settings which can be investigated in the system "master" database. When setting up a server and deploying the database, the user can only choose a region in which the physical servers are placed and which should ideally be as close as possible to the users' location to avoid latency issues. However, the exact geo-graphical distribution of the physical database servers and their replicates is not visible to the user.

### 1.4.2 Typical Use Cases

The following are typical use cases of MS Azure SQL Database:

- Relational data storage for websites and cloud based applications
- Business and consumer web and mobile applications, especially for multi-tenant applications
- OLTP processes with relational data across manifold industries, i.e. used by retail or logistics, often combined with Data Warehousing services
- Outsourcing of the database management in order to let the company focus on value-added services in case of the companies' resource limitations and storing data in the cloud for effective security and isolation

### 1.4.3 Evaluation of Azure SQL DB

We determined 4 factors that are important to consider when the performance of the database as a whole shall be evaluated.

- **Performance**  
Recall that the performance of the database is calculated in Database Transaction Units, which is a blended measure of CPU, memory and I/O. Microsoft guarantees a certain level of resources for a database and providing a predictable level of performance. Microsoft also provides a “DTU Calculator” to estimate the number of DTUs needed. Certainly, this flexibility is a major advantage of Azure SQL but also makes it difficult to compare it to other database systems as the performance of the database depends on the service tier chosen or the amount of money spent. Furthermore, Azure does not provide the exact Memory and CPU capacities or read/write rates equivalent to the DTUs, which may not be an issue if the database is used in day-to-day business but makes benchmark comparisons more challenging.
- **Included Services**  
As mentioned in the previous sections, Azure SQL is offered as a “Database as a Service”. Thus, all usual concerns about maintenance, security or replication are handled by Microsoft and not the user anymore. Several log statistics, alerts and performance optimizers complete the service package.
- **Functionality & Complexity**  
As Azure SQL is code based on SQL Server, it follows most of SQL Servers design features, integrates well with existing systems and offers a familiar development environment for database administrators. The intuitive user interface on the Azure portal makes it easy to use by eliminating almost any need for hard-coding during the set-up. As it is embedded in the Azure cloud service platform, it can be utilized for many of the offered services and thus integrated into data flows of web applications, Business Intelligence or Big Data Frameworks.
- **Cost**  
The cost for usage of azure is based on different factors. If we only focus on the cost of Azure SQL, it depends on the geographic region, type and size of database, the type of subscription and the DTU's consumed. Microsoft also provides a “Pricing calculator” to get an estimation of the price. In general, Cloud databases have the great cost advantage of eliminating the need for an expensive server set-up (hardware and software), maintenance and replacement after a certain period of time. The exact cost advantage over an on premise set-up depends on the desired performance, database size and workloads, but overall seems to be less costly, especially if all

included services and future scalability perspectives are considered. In chapter 2.6.2. we made a rough cost estimation to give a first impression of the price range of Azure.

### 1.5 Summarized Advantages and Disadvantages of Azure SQL

The following table provides a summarized overview of the advantages and disadvantages of the database:

Advantages	Disadvantages
<b>Performance</b>	
<ul style="list-style-type: none"> <li>• Scalable performance up to high levels</li> <li>• Adjustable performance anytime</li> <li>• Lower down-time</li> </ul>	<ul style="list-style-type: none"> <li>• Low throughput and high execution time for lower service tiers</li> <li>• Though supporting most of the features of SQL Servers, some are still not implemented</li> </ul>
<b>Services</b>	
<ul style="list-style-type: none"> <li>• Development of data-driven applications and websites in any programming language, without need for managing the infrastructure</li> <li>• Services like hardware maintenance, security, active directory, updates etc. are covered by Microsoft</li> <li>• Build-in intelligence and performance monitoring tools support users to design optimal database and application architectures</li> </ul>	
<b>Functionality &amp; Complexity</b>	
<ul style="list-style-type: none"> <li>• Flexible service plans meet the needs of both - large and small businesses</li> <li>• Easy connections using username and password, human friendly user interface</li> </ul>	<ul style="list-style-type: none"> <li>• The only way to send data to Azure is to upload or sync it across the network, which rises the need for an appropriate bandwidth</li> <li>• SQL Azure has limitations on usage and will throttle database connections under heavy loads</li> </ul>
<b>Cost</b>	
<ul style="list-style-type: none"> <li>• The higher the service tier, the more the cost, but performance and resources scale as well</li> <li>• No upfront-cost for server set-up</li> <li>• Cheap, low performance service tier offerings for test deployments</li> </ul>	<ul style="list-style-type: none"> <li>• Cost increase to quite expensive levels for higher service tiers</li> </ul>

Table 3: Summarized Advantages and Disadvantages of Microsoft Azure



## 2 Performance benchmark for a real-life application example of a retail sales management process

In order to get a better understanding of the database performance and usage in real-life applications, we developed a retail sales management process, which is described in section 2.1. We then created the according datasets and set up two identical databases on SQL Server and Microsoft Azure, with a step-by-step guide for Azure SQL DB. In section 2.6 we conduct a performance benchmark between these databases, based on typical queries of the application.

### 2.1 Introduction to Use Case & Benchmark

In this application, a small-sized company that owns 3 stores to sell its products would like to store its sales and product data. As of now, all transactional data is stored in small, local relational databases which makes it impossible for the management to query the overall sales data or conduct generic updates of the database, i.e. price changes of products due to promotions. Additionally, the number of orders for his products usually vary time to time. For example, on the festivals the sales are very high and in the normal days they are comparatively low, which rises the need for on-demand scalability of the database performance. Therefore, the management decided to invest in one central database for all sales data, but is yet unsure if a cloud or on premise set-up is the right choice. By investigating the performance of both systems, we shall analyze which one is more suitable for this use case from both, a technical and a business perspective.

### 2.2 Generation of test data

To perform the benchmarking, two datasets were randomly generated, which only differ in the number of rows, but have the same schemas. Both sets contain data entries for two tables, a sales table and a product table. The schemas for the relations are as follows:

- Sales {salesID, sdate, storeName, products, quantity}
- Products {productID, product\_name, price}

The small dataset contains 10.000 rows, whereas the large data set contains 1.000.000 rows in each relation. *SalesID* and *productID* are unique primary keys and *products* in the sales relation is a foreign key of *productID* to identify the products sold in a sales transaction. For detailed attribute types and constrains, please refer to the code snippet in section 2.4.1.4.

### 2.3 Specifications for test environment

- SQL Server: Microsoft SQL Server Management Studio 2014
- Azure SQL Database, Subscription: Free Trial, Service Tier: Standard S1 service tier (250GB storage, 20 DTU)
- MacBook Pro, MacOS High Sierra v.10.13.1, Processor 2,7 GHz Intel Core i5, 250GB SSD, Memory 8 GB 1867 MHz DDR3, Running a parallels Desktop v. 13.0.1 Virtual Machine with Microsoft 10 and unlimited scalable resource sharing upon demand

## 2.4 Database creation

This section provides a How-To Guide to set up a SQL database on the Microsoft Azure platform.

### 2.4.1 Create Database in Azure SQL

The Azure SQL database can be deployed in three different ways:

- using SQL Server Management Studio (SSMS),
- using .NET or
- migrate an existing SQL Server database

In the following steps we are showing how to create the database using SSMS.

#### 2.4.1.1 Create new SQL database on the Azure platform

The first step is to create the database at the Azure portal. Therefore, we have to sign in to our Microsoft account and navigate to the SQL database repository, where we can add a new database as shown in figure 10. As a name we chose *SalesApplication* and the subscription can be selected from the subscribed service level of the Azure account, which in our case is the free trial. Azure offers to group different databases in a resource group, which share the general database settings. This is in particular useful if the intention is to set up more than one database as common settings for e.g. firewalls only have to be set up once. In our case, we created a new resource group and called it *PerformanceTesting*.

In the select source field, the user can choose between migrating an existing database from SSMS or create a new one. If the user hasn't created a server on Azure before, a new one needs to be created, following the server set-up wizard. In our case we chose a server location in West Europe and named it *peritest*. The server configuration is important to be able to connect and interact with the database later on. As we don't have many databases, we don't configure elastic pools as of now.

The screenshot shows the 'SQL Database' creation interface in the Azure portal. It features several configuration fields: 'Database name' set to 'SalesApplication', 'Subscription' set to 'Free Trial', 'Resource group' set to 'PerformanceTesting' (with 'Create new' selected), 'Select source' set to 'Blank database', and 'Server' set to 'peritest (West Europe)'. There is a checkbox for 'Want to use SQL elastic pool?' which is set to 'Not now'. The 'Pricing tier' is set to 'Standard S3: 100 DTU, 250 GB'. At the bottom, there is a 'Create' button and a 'Pin to dashboard' checkbox.

Figure 10: Create an Azure SQL database on the MS Azure platform

The pricing tier is the most important part of the configuration as it directly influences the performance of the database. The maximum service tier support in the free account is the Standard S1 service tier with 250GB storage and 20 DTUs as shown in figure 11.

After entering all required information press 'Create' to let Azure create the database. This process might take a few minutes, but can be monitored in the notification center on the Azure portal status bar in the upper right corner of the screen at all time.

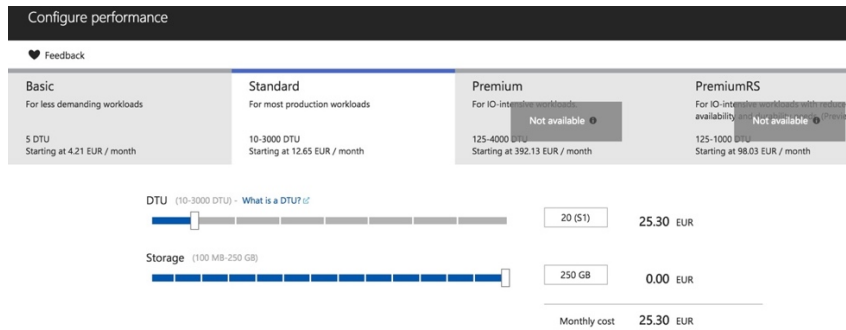


Figure 11: Selection of Service Tier for the Azure SQL database

### 2.4.1.2 Set firewall settings

After creating the database, the firewall has to be configured in order to allow inbound and outbound connections to the database. To do so, open the newly created database on the Azure platform and navigate to *Set Server Firewall* in the menu bar at the top as shown in figure 12.

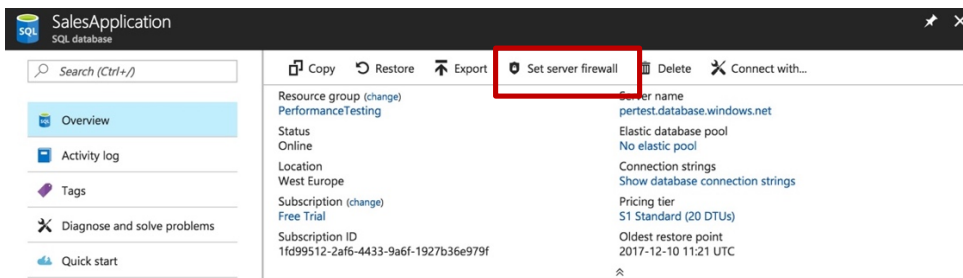


Figure 12: Set Server firewall menu

Press the 'Add Client IP' and enter your IP address to allow connections to your clients as shown in figure 13. If this step is skipped, SSMS will raise an access denied error when connecting to the database.

*Please keep in mind that these setting have to be updated, whenever the network over which the client connects to the database has changed, as the respective DNS server assigns a new IP address that has to be registered.*

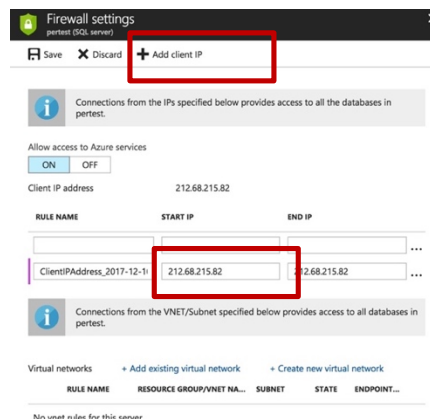


Figure 13: Add Client IP to the Server Firewall

### 2.4.1.3 Connecting to Azure via SSMS

The next step is to connect to the Azure SQL database from SSMS. Open the connection wizard in SSMS and type in the credentials showed in figure 14. The server name has to be of the type <servername>.database.windows.net. Azure is only accepting SQL Server Authentication. Login and Password are the ones chosen when creating the server on Azure. Navigate to the tab 'Connection Properties' and enter the SQL database name (here: *SalesApplication*) in the database field. Then press connect. The Azure server and database should now show up in the connection manager in the same way as local repositories do.

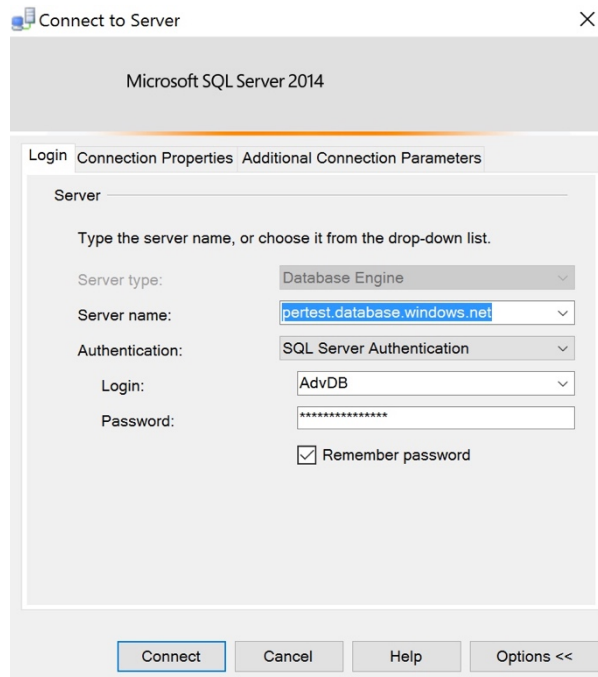


Figure 14: Setting up a connection from SSMS to Azure

### 2.4.1.4 Create database schema

So far the database is still empty. To create the schema specified in section 2.2 open a new query and run the following T-SQL statement, which sets the relation schema, attribute types, primary and foreign key constraints as well as the index. As the database will be mainly queried to lookup sales for a specific store or product, we decided to introduce clustered indexes on the primary keys of both relations.

```
USE SalesApplicationMac
GO
/*Object: Table [dbo].[products]*/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[products](
    [productID] [int] NOT NULL,--
    [product_name] [varchar](40) NOT NULL,
    [price] [int] NOT NULL,
    CONSTRAINT [PK_products_master] PRIMARY KEY CLUSTERED
(
    [productID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO

/* Object: Table [dbo].[sales]*/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[sales](
    [salesID] [int] NOT NULL,
    [sdate] [varchar](40) NOT NULL,
    [storeName] [varchar](40) NOT NULL,
    [product] [int] NOT NULL,
    [quantity] [int] NOT NULL,
    PRIMARY KEY CLUSTERED
([salesID] ASC)
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO

/* Object: ForeignKey [FK_products_sales] */
ALTER TABLE [dbo].[sales] WITH CHECK ADD CONSTRAINT [FK_sales_products] FOREIGN KEY([product])
REFERENCES [dbo].[products] ([productID])
GO
ALTER TABLE [dbo].[sales] CHECK CONSTRAINT [FK_sales_products]
GO
```

Figure 15: T-SQL statement to create database schema

After running these statements, the tables `dbo.products` and `dbo.sales` should appear in the *tables* folder of the database.

### 2.4.1.5 *Populate the database with the application data*

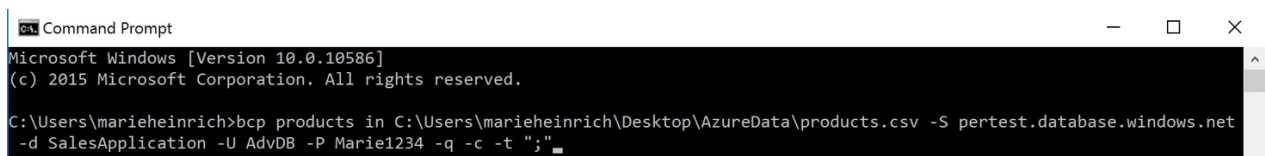
To populate the database, we use the bcp command-line utility.

*Note that for the following steps the bcp and sqlcmd command-line utility have to be installed on the client. Both are available for free download in the Microsoft store.*

Open the command prompt window on your client and execute a command following this structure:

```
bcp <TableName> in <path of file> -S <ServerName> -d <DatabaseName> -U <Username> -P <Password> -q -c -t “;”
```

Replace all <placeholders> with the respective information of your database and adjust “;” to the special character that separates the entries in the .csv or .txt file containing the data that shall be inserted. For our application the command is the following:



```
Command Prompt
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\marieheinrich>bcp products in C:\Users\marieheinrich\Desktop\AzureData\products.csv -S pertest.database.windows.net
-d SalesApplication -U AdvDB -P Marie1234 -q -c -t ";"
```

Figure 16: bcp command-line statement to load data into the database

After uploading the product data, the sales data has to be uploaded, following the same procedure, but replacing the table name *products* with *sales* and adjusting the file path respectively.

### 2.4.1.6 *Additional notes & SQL Server set-up*

As we’re planning to conduct the benchmarking with two dataset sizes (10.000 and 1.000.000 rows respectively), we decided to set up two databases to avoid deleting and uploading of data between the tests. Both databases are identical except the number of rows. Therefore, we repeated all steps of the previous sections to create a database for the smaller data set, which is referred to *SalesApplicationSmall* hereinafter.

Furthermore, we assume that the reader is familiar with the database set up in SQL server, so we won’t explain detailed steps here. For the benchmarking on SQL Server we used the same scheme as shown in figure 15 and used the Import Wizard from SQL Server to import the datasets from the respective flat files to the databases. A detailed documentation can be found [here](#).

## 2.5 Querying Azure SQL database

We identified the following typical use cases for our application that require the subsequent different queries types, which were executed during our performance test.

### 2.5.1 Select sales data

To start we first performed a full table scan by executing the following SQL statement:

```
SELECT COUNT(*)  
FROM dbo.sales  
WHERE quantity between 0 AND 10000
```

In a real-life application, it is quite unlikely that a full table scan of all sales transaction is requested. Instead, a more specific where clause would be added in order to display sales transactions for a certain store or time period. However, as we want to conduct our benchmark on the full set of rows, we decided to set the where clause to a range that covers the full table for testing purposes.

### 2.5.2 Update product prices during promotions

A common event in our use case is setting promotions for the products. In this case, the price in the product relation needs to be adjusted, executing the following SQL statement:

```
UPDATE dbo.products  
SET price = 120
```

Following the same logic as above, we update all product records in our benchmark to avoid affecting only a few rows with our query.

### 2.5.3 Calculate total sales amount

Another important query is the calculation of the total sales amount. For this, the sales and product relations have to be joined based on the productID and the sales quantity has to be multiplied with the product price:

```
SELECT quantity * price AS SalesAmount  
FROM dbo.products p, dbo.sales s  
WHERE p.productID = s.product
```

### 2.5.4 Insert new Sales Records

To simulate the insertion of 10.000 new records (1.000.000 for *SalesApplication* respectively), we used the following set up:

First, a new table called *isales* with the schema {salesID, sdate, storeName, products, quantity} was created in the database. To simulate the insertions, we copied the entries of the *sales* relation to the *isales* relation, using the following procedure:

```
/* Create table type */
CREATE TYPE insertionTable AS TABLE
(salesID int, sdate varchar(40), storeName varchar(40), product int, quantity int);
GO

/* Create procedure to insert data into new table. */
CREATE PROCEDURE dbo.usp_InsertSales
@IT insertionTable READONLY
AS
SET NOCOUNT ON
INSERT INTO dbo.isales
(salesID
,sdate, storeName
,product, quantity)
SELECT salesID
,sdate, storeName
,product, quantity
FROM @IT;
GO

/* Declare variable that references the table type. */
DECLARE @SalesIT AS insertionTable;

/* Add data to the variable from other table in the db. */
INSERT INTO @SalesIT (salesID
,sdate, storeName
,product, quantity)
SELECT *
FROM dbo.sales;

/* Pass variable data to stored procedure and execute. */
EXEC usp_InsertProducts @SalesIT;
GO
```

Figure 17: Stored Procedure to insert sales records



## **2.6 Performance Benchmark**

The benchmark is following the performance evaluation factors outlined in section 1.4.3. The results for the technical performance testing are shown and discussed in section 2.6.1. In a second step we evaluate the cost of both databases by putting the obtained performance results into perspective with the arising costs according to the Azure SQL cost calculator in section 2.6.2. In section 2.6.3 we lastly add a few qualitative considerations about the included services, functionality and flexibility of the database to round up our comprehensive performance evaluation.

### **2.6.1 Performance**

The goal of the performance testing is to determine if SQL Server Azure SQL is more performant for the selected use case queries on the chosen service tier. To evaluate the database performance, we executed all queries listed in section 2.5 eleven times for each of the four databases (two databases each on SQL Server and Azure, one containing the large and one the small dataset). We ignored the first execution as this usually takes significantly longer due to the caching and took an average of the response time of the remaining ten executions (derived from the SQL Server client statistics).

After the final consolidation of performance test results we observed that the results for the large and the small dataset did not differ significantly in their overall trend, but only in their scale. Therefore, we will hereinafter only discuss the results per query executed on the large data set and shall discuss the differences to the small dataset in a subsequent section.

### 2.6.1.1 Performance Test Results: Select query

The following charts show the Execution times of all 10 executions for the select query.

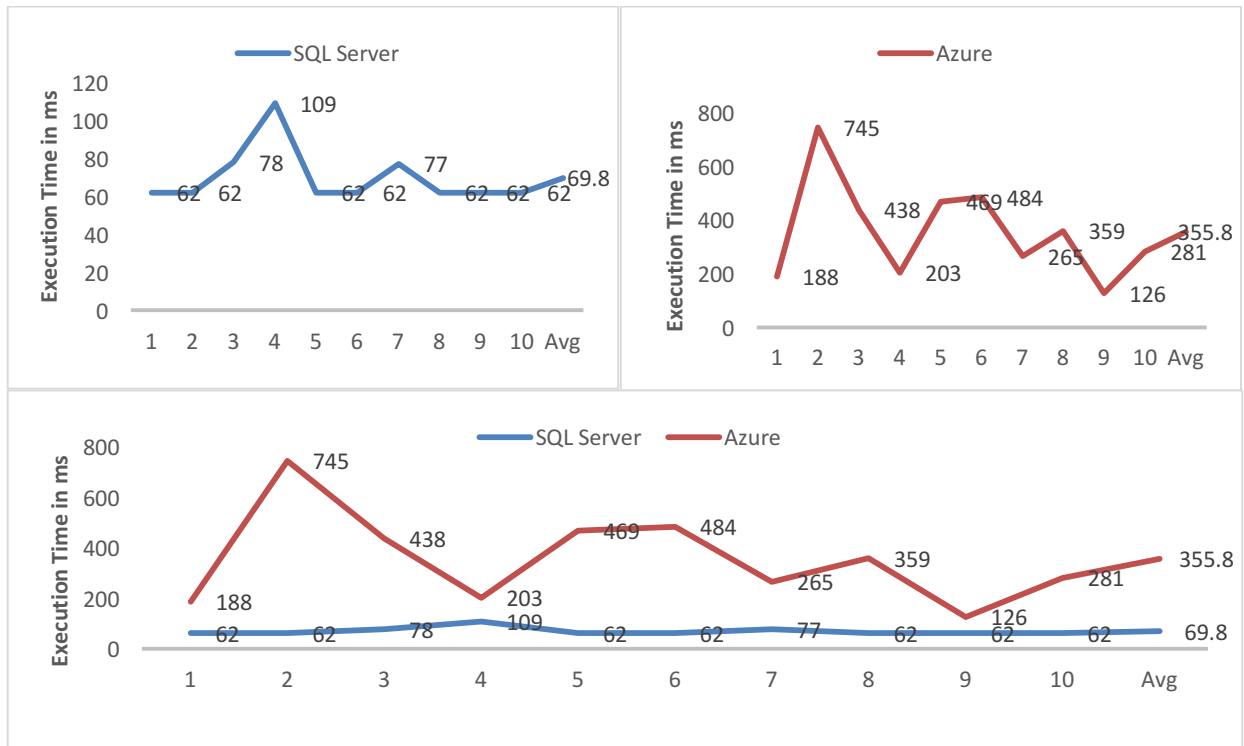


Figure 18: Performance Test Results Select query 1,000,000 rows

We observe that SQL Server is outperforming Azure by running the query on average 5x faster. Surprised by that clear results, we tried to figure out the reason for the slow performance of Azure and took a closer look on the query. The query that was used actually tests the read rates of the databases. By using a COUNT(\*) we ensure that the results are not strongly affected by network or I/O issues as only one integer needs to be outputted, whereas the WHERE clause forces the database management system to scan through the whole table as there is no index on the quantity column. Thus, as we don't send much data through the network, the volatility in both graphs might show the effect of choosing the best query plan to execute the query, whereon SQL Server performs more stable.

However, for the relative comparison we should keep in mind that we still compare hardware and network performance rather than DBMS performance, as the code base for both databases is SQL Server. SQL Server in this case was run on a notebook that has 8GB of RAM and a 2.7Ghz processor whereas Azure is only "equipped" with the resources reflecting 20 DTUs.

An Azure SQL benchmark that was conducted in a more professional environment over longer time periods of time with heavier workloads was able to calculate the memory underlying memory of different service tiers. Despite the fact that the benchmark is already 2 years old and service tiers might have changed, it still shows impressively, how less memory is allocated to the database in the standard tiers and how it scales up with the premium tiers.

In the graph the green bars show results for the tests conducted in 2015 whereas the blue ones for tests conducted in 2014 and thus reflect, how Microsoft is shifting and adjusting resources under the surface of its platform. By investigating this chart, it is obvious why the processing time of Azure is generally slower, as the low resource assignment has also been shown for CPU or read/write rates in the benchmark study.

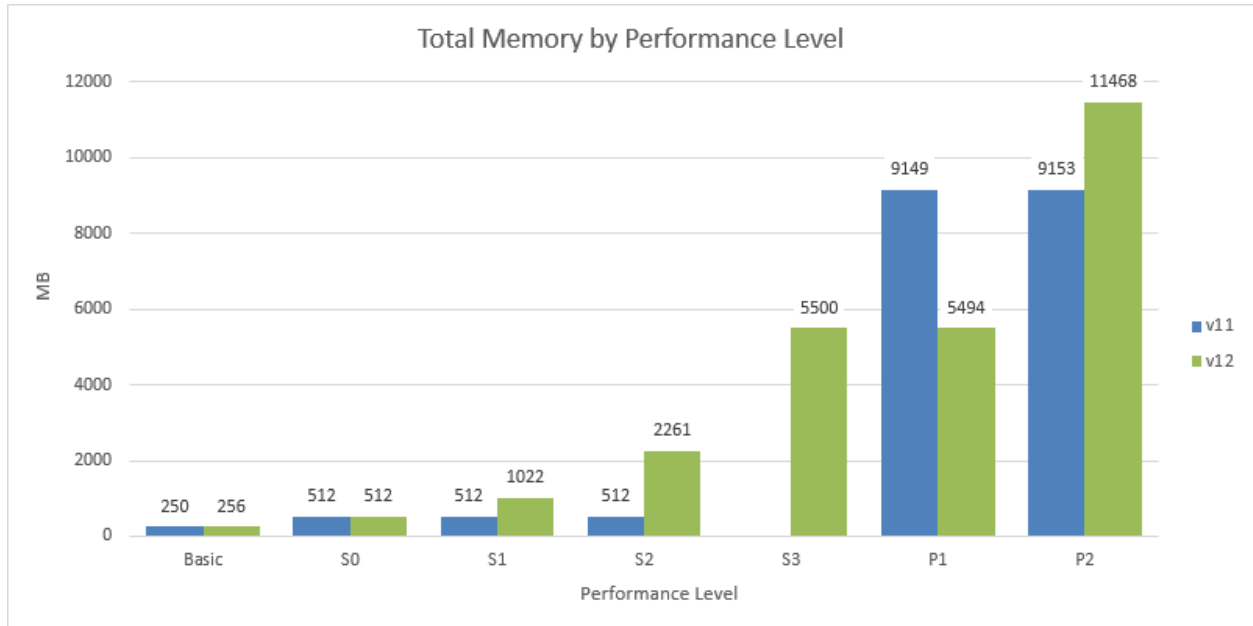


Figure 19: Estimated memory allocation of SQL Azure per service tier<sup>10</sup>

### 2.6.1.2 Performance Test Results: Join query

The following charts show the Execution times of all 10 executions for the join query.

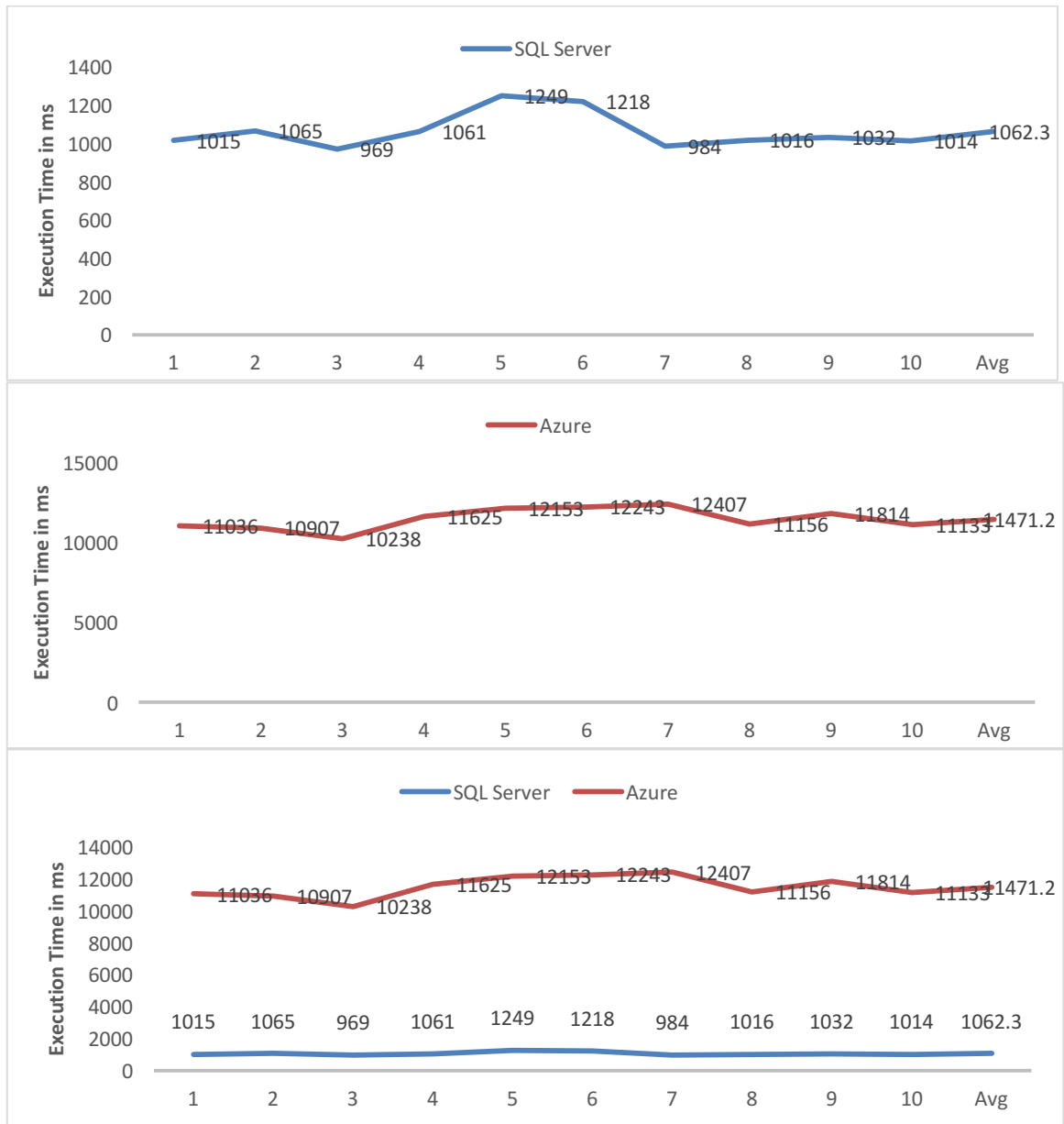


Figure 20: Performance Test Results Join query 1.000.000 rows

For this test results we first observe that the performance is much more stable than for the previous query. This might be due to the fact that the join is executed on the columns with clustered primary keys, which makes the execution straighter forward. But still, SQL Server is again on average executing 9x faster than Azure, similar to the results obtained above. We assume that the slower execution time of Azure might again be due to the low performance level assigned as well as due to network latencies between the notebook and the Azure server.

### 2.6.1.3 Performance Test Results: update and insert queries

The following charts show the Execution times of all 10 executions for the update and insert queries.

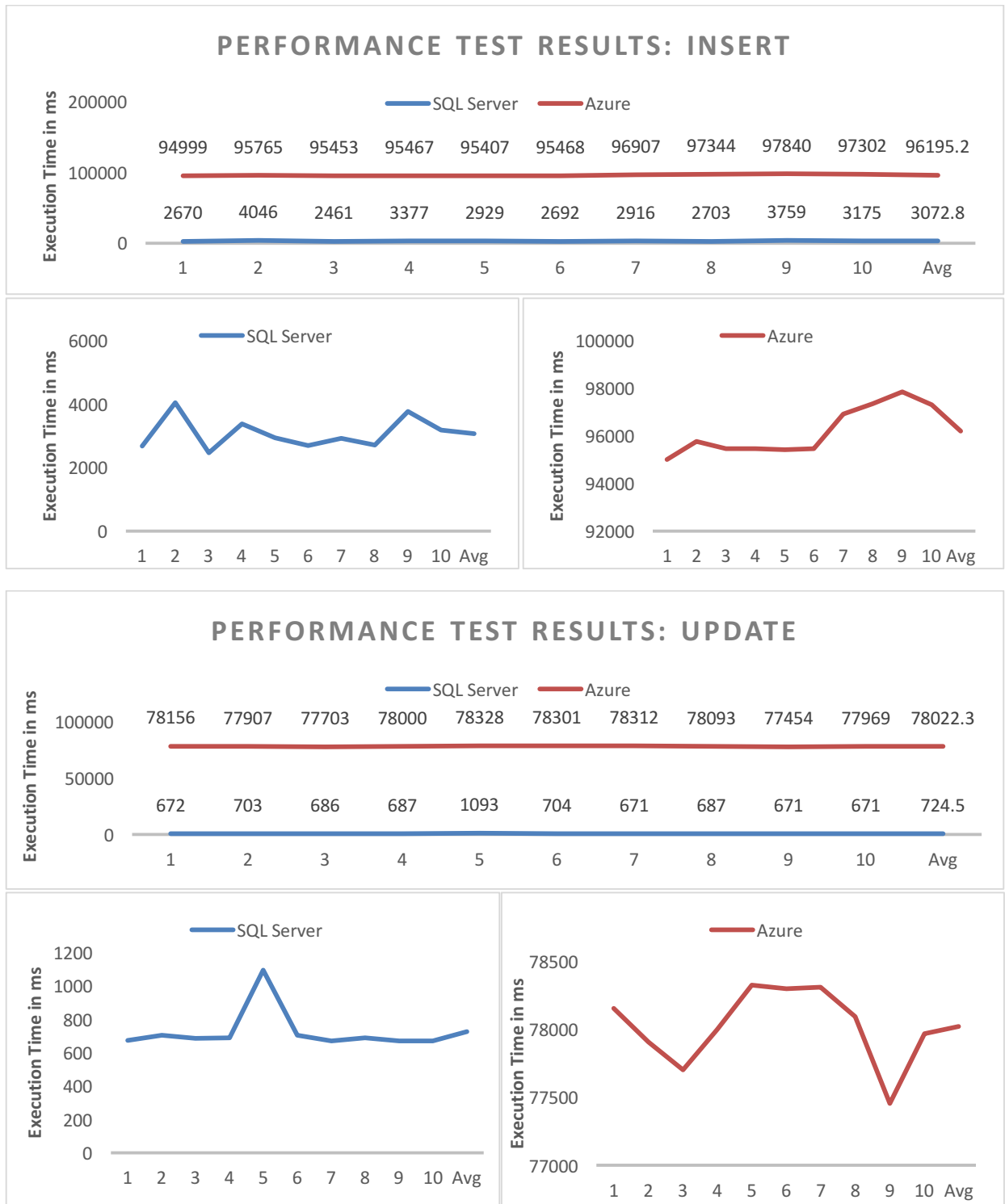


Figure 21: Performance Test Results Select and Update query 1.000.000 rows

When considering the two queries above, the difference between Azure SQL and SQL Server reaches a new level. SQL Server is outperforming Azure SQL by 107 times in the update query and 32 times in the insert query. We assume that two things could be an essential determinant to this: The DUTs assigned to the used service tier as well as the network.

To examine the first, we queried the Azure SQL system statistics of the database using the following SQL code:

```
SELECT
    AVG(avg_cpu_percent) AS 'Average CPU Utilization In Percent',
    MAX(avg_cpu_percent) AS 'Maximum CPU Utilization In Percent',
    AVG(avg_data_io_percent) AS 'Average Data IO In Percent',
    MAX(avg_data_io_percent) AS 'Maximum Data IO In Percent',
    AVG(avg_log_write_percent) AS 'Average Log Write Utilization In Percent',
    MAX(avg_log_write_percent) AS 'Maximum Log Write Utilization In Percent',
    AVG(avg_memory_usage_percent) AS 'Average Memory Usage In Percent',
    MAX(avg_memory_usage_percent) AS 'Maximum Memory Usage In Percent'
FROM sys.dm_db_resource_stats;
```

The outcome of this query can also be obtained by looking at the query performance insights statistics provided in the Azure portal, which revealed the following graph:

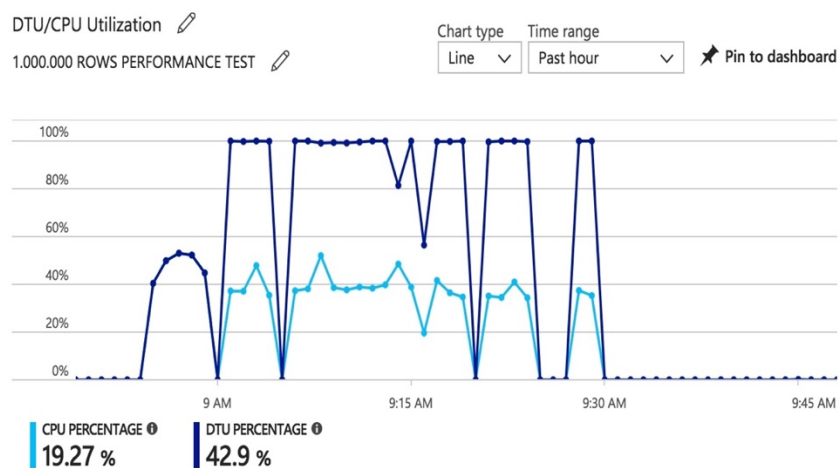


Figure 22: Azure SQL Query Performance Insights

From first sight we can see that we were right in our assumption of exceeding the DTU limits. Following the dark blue line, we can identify that the graph clearly hits the 100% mark while the query was run and is cut off at this point. Azure SQL is throttling the performance significantly if the DTU capacity is reached. Throughout the testing Azure also raised an alert that the workload is heavy and a higher service tier should be considered. Recall that DTU is a measure combining CPU, memory and I/O performance metrics. We can see in the graph that in this case the CPU was not the main issue. The database statistics query revealed that both, the memory and the log write percentages reached 100% during the query execution – a clear evidence that the service tier was utilized to the maximum.

In addition to that we wanted to retrieve the average write of Azure SQL rate during the insert. Therefore, we first queried the table size of the sales table with the following SQL statement:

```
EXEC sp_spaceused N'dbo.sales';
GO
```

The statement returned a total table size of 52 MB. Knowing from figure 21 that our execution time for the insert was 78,022ms, we can calculate that our write rate (insertion) was equivalent to 0.669 MB/s or 40 MB/min. When we compare this result with the service tier benchmark that we introduced earlier, we notice that the write rate from our small set-up complies with the write rate result for the S1 service tier that was derived during the service tier benchmark:

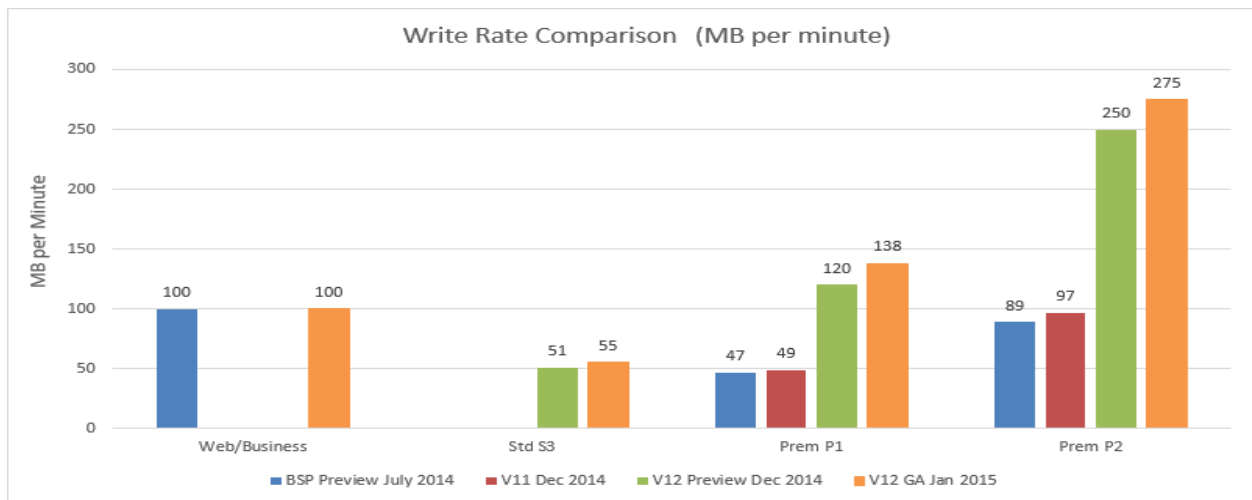
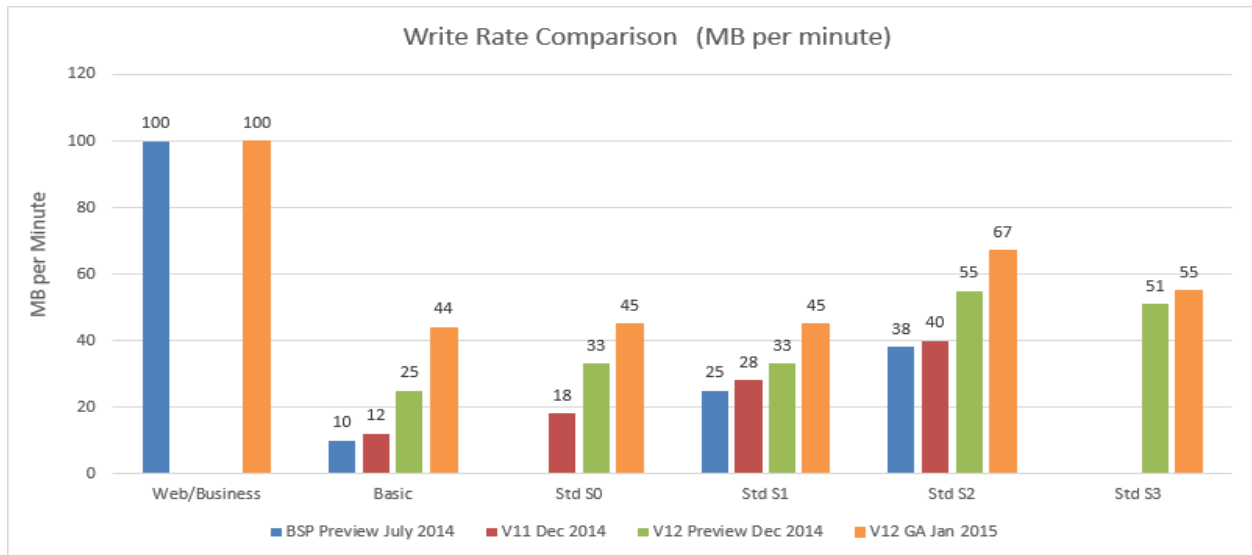


Figure 23: Write Rate Comparison (MB/min) of different Azure SQL Service Tiers<sup>10</sup>

We therefore conclude that our assumption that we reached the maximum of our service tier was correct and the main reason for the slow performance. Hence, if the performance should be increased, the service level need to be increased as it can be seen in figure 23.

### 2.6.1.4 Result comparison of large and small dataset

The following charts show the average execution time obtained per query for the large and the small dataset and the respective differentiation in percent.

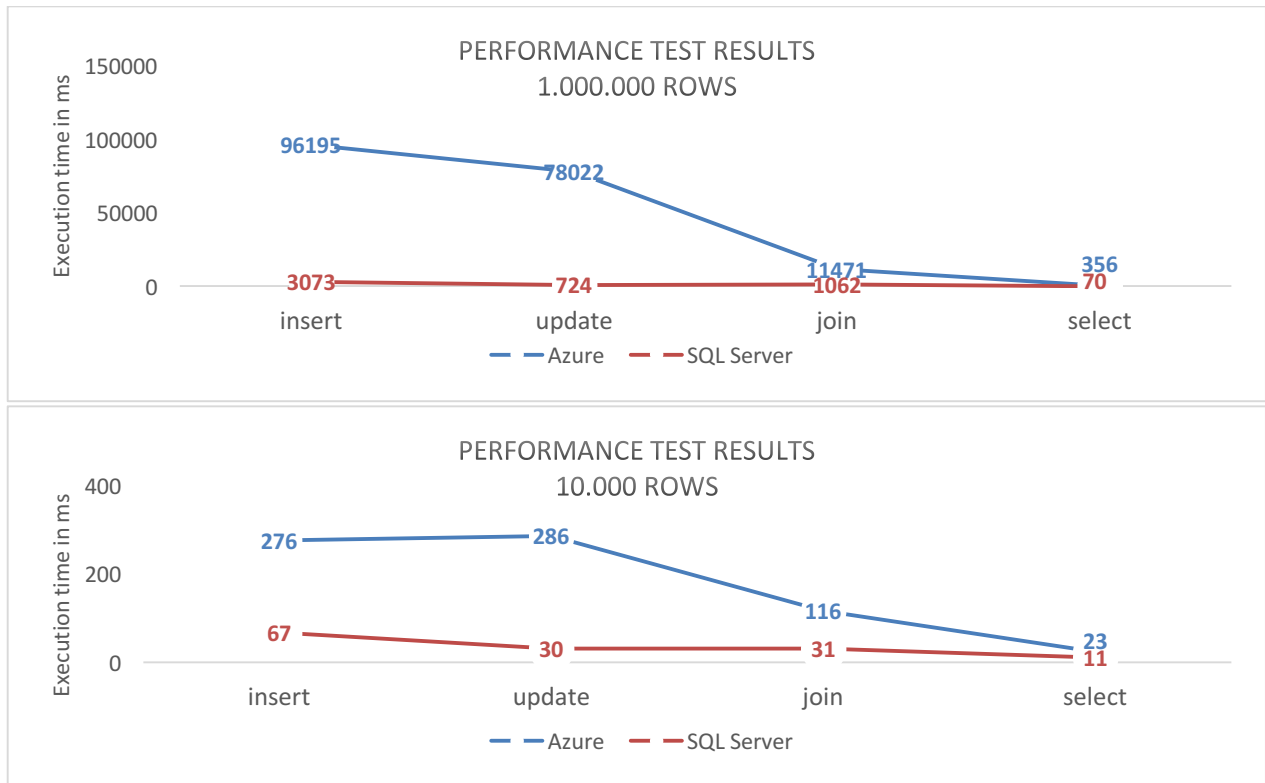


Figure 24: Summary of average performance test results

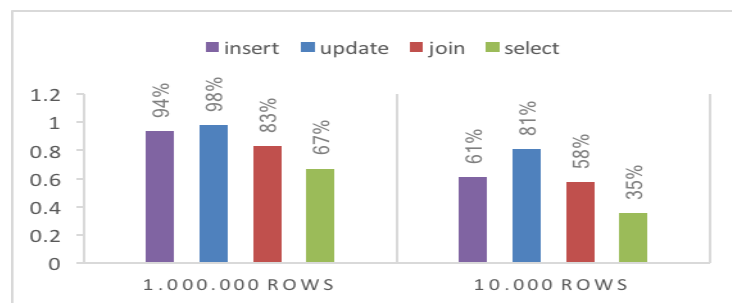


Figure 25: Difference between Azure SQL and SQL Server Performance Test Results in %

It can clearly be seen that the overall trends are the same for both datasets. However, considering the relative difference of the results obtained from the two databases, the results are much closer to each other for every query. This might mainly be due to the fact that the small dataset does not come close to the limits of CPU, I/O or memory capacities of Azure and neither scratches the network bandwidth restrictions. We observe that the update instead of the insert query has the longest execution time for the small dataset, which might as well due to its limited size, which is not affected by the network or DTU limitations respectively. However, Azure still shows significantly worse performance than SQL Server, due to the overall DTU capacity, which is very low with only 20 DTU's for S1. Nevertheless, the fact that Azure performs worse in this tests does not mean it is the wrong choice for our use case.



As mentioned earlier, Azure allows easy performance scalability by choosing a higher service tier. Thus, we can conclude that the retail company of our use case should choose a higher service tier in order to achieve similar or better results to SQL server.

### 2.6.2 Database Cost Comparison

We discussed already that the performance of the database depends on the service tier chosen and can be scaled up significantly if one is willing to pay for it. In the previous section we observed that the S1 service tier obviously is not sufficient to cope with workloads based on the larger dataset. To solve our use case, it is important to know, what financial obligations both deployments (on premise and cloud) would bring along. We therefore estimated the cost for both options. These considerations however are very rough and might differ significantly based on assumptions and real-life database usage. However, the aim here is only to provide an idea of the price ranges needed, which then would be needed to calculate in detail upon deployment.

In general, we're assuming need for high availability and security. For the on premise solution we assume that the hardware for the server needs to be purchased but network infrastructure incl. firewalls, routers, wires etc. is already existing. Spare parts needed for maintenance as well as rental partial cost of the space for server rooms or depreciations are also not considered. We would need to set up two servers (one for the actual deployment, another for the replication), both equipped with Windows Server 2016 licenses and covered by a warranty and service agreement. The SQL server license only needs to be bought for one server, as the second one won't be queried by any of the users. Both Microsoft licenses are subjective to the cores covered and cost are in accordance with the prices shown on the respective Microsoft Store websites as of December 2017. The server costs were taken from a server online shop: [www.servershop24.de](http://www.servershop24.de). Cost were aggregated over a 3-year period and then broken down to monthly cost as we assume that after 3 years the servers need to be replaced due to performance, scale and security considerations. Cost for Azure SQL were calculated for different service tiers using the cost calculator on the Azure platform, assuming 1 single database, 720h/month availability, 750GB Storage, Region West Europe and standard support.

On premise SQL Server	
Hardware: 2x Server (HP ProLiant DL360 Gen9 Server, 2x Xeon E5-2628v3 8-core 2.5GHz, 32GB RAM, 2x300GB SAS 10K)	6,200 €
Hardware Care packet 3 year 24/7 warranty	1,320 €
Windows Server 2016 Datacenter Edition License (2 x 16 cores)	12,310 €
SQL Server 2016 License (min 4 cores per processor = 8 cores)	57,024€
Total	76,854€
<b>Per month (over 3yr period)</b>	<b>2,135€</b>

Table 4: Monthly cost estimation on premise SQL Server

Azure SQL Database monthly cost	
Service Tier S1: 20 DTU (used in this benchmark)	277€
Service Tier S4: 200DTU	411€
Service Tier P1: 125DTU	585€
Service Tier P2: 250DTU	888€
Service Tier P4: 500DTU	1,467€
Service Tier P6: 1000DTU	2,710€

*Table 5: Monthly cost estimation Azure SQL Database*

We observe the painful license cost for the on premise set-up that significantly determine the overall monthly cost. However, as we're talking about hardware, it is quite likely that these costs increase even further, if spare parts or other additional maintenance work is needed that is not covered by the general insurance. With respect to the Azure SQL DB cost we can see that prices within the premium service tiers (P1-6 in the table) are more or less scaling linearly with the amount of DTU provided. In contrast this seems not to apply for a change between the standard and premium service tier. From the table we can see that between the S4 and P1 service tier, the DTU amount decreased whereas the price increased. This results in a cost per performance jump of 130%. It provides an indication that DTU actually does not refer to the same underlying memory, CPU and read/write rates compared between two levels of service tiers and can only be used for comparisons in-between the same tier level. However, to determine the exact resources and scaling function further, in-depth benchmarking across all service tiers is needed. A first insight can be derived from figure 23, although this figure might have changed due to adjustments from Microsoft as well as additionally added service tiers.

Nevertheless, for our use case we could see that SQL Server which dealt pretty well with the workload was on average approx. 50x faster than Azure SQL. Transferring this to DTUs, we could estimate that we would need  $20\text{DTU} \times 50 = 1000\text{DTUs}$  in a standard service tier to achieve a good performance or 434DTUs in a premium tier, assuming that the 130% cost/per performance jump reflects the same increase in terms of resources. Thus, we would target service tier P4 as a base for the next performance benchmark for our use case.

From a cost perspective for the use case, using Azure SQL with a P4 Service Tier would still save cost of approx. 31% per month compared to the on premise solution. In addition to all the service, flexibility and functionality features of the DB, the cost estimation turns the considerations around to the favor of Azure SQL. However, this should just be seen as a first exploration into the analysis of how much resources are behind the DTU in every service tier and can be used as a basis for a new performance test but should not be considered as a fully reliable result of a performance benchmark.

### **2.6.3 Further considerations for database evaluation**

Most conclusions so far were related to hardware specifications. But there are other factors like the application design which may also affect the performance. Therefore, we also need to keep these things into mind while designing a database: Are indexes created for right set of attributes? Does the system avoid deadlocks and round-trips? For these cases, Azure provides a performance monitor that is able to suggest new or better index structures.

Another important factor for our consideration is the cloud service landscape of Azure that comes with the Azure SQL database. In case the retail store considers to develop an app or a website for its business, Azure would have all the necessary infrastructure already in place, whereas for SQL Server, all software and hardware for development, testing and production environment would need to be bought. Additionally, as security is becoming more important and the technology landscape is changing at a rapid speed, in the future it will be difficult for companies to keep up with the latest security patches, maintenance and technologies if the database is hosted on premise. In contrast, for a cloud database like Microsoft Azure, Microsoft takes care of all that and guarantees services and availability in the respective SLAs.

Last but not least, we mentioned in the introduction of the use case that the business plans to grow its business and thus will need to scale up its IT landscape accordingly, which is highly supported by on click for Azure SQL, but not for SQL Server.

## **3 Final conclusion**

Summarizing the results obtained during our performance evaluation, it could clearly be seen that the assigned resources (in terms of DTU) to the used Standard S1 Service Tier, were not sufficient enough to compete with our on premise set up of SQL Server. However, as this is related to hardware resources and network capabilities, we cannot conclude that Azure SQL database in general performs worse than the SQL Server database. Rather, Azure SQL needs to be run on a higher service level in order to catch up or even outperform SQL Server.

Nevertheless, taking into account the other benefits that come along with the SQL Azure database, such as flexibility, scalability, availability, integration with other cloud services on the Azure platform and maintenance services, Azure SQL still remains a very attractive package. So, if a company just wants to concentrate on its application instead of all the hardware, maintenance, security, license, updates, troubleshoots, storage etc., then shifting their database to the cloud is surely a good option. For our use case, we clearly suggest the retail store owner to opt for the cloud database and subscribe to a service tier which is suitable for his business. With this, he can dynamically adjust the service tiers whenever needed and scale his IT structure up according to his business growth.

Thinking even further, we're heading to a world which gets evermore connected. Thus, it is indispensable for all businesses to think about the structure and data usage of their future company in order to invest in the right and long-term suitable technologies from the beginning.

## 4 Sources

- (1) Cancila, M.; Toombs, D.; Waite, A. D.; Khnaser, E. 2017 Planning Guide for Cloud Computing. **2016**, No. October 2016.
- (2) IBM. What is a Cloud Database <https://www.techopedia.com/definition/133/cloud-provider> (accessed Dec 16, 2017).
- (3) Microsoft. Directory Azure Services <https://azure.microsoft.com/en-us/services/> (accessed Dec 16, 2017).
- (4) Tupper, C. D. Distributed Databases <http://linkinghub.elsevier.com/retrieve/pii/B978012385126000022X> (accessed Dec 16, 2017).
- (5) Microsoft. What is the Azure SQL Database service? | Microsoft Docs <https://docs.microsoft.com/en-gb/azure/sql-database/sql-database-technical-overview> (accessed Dec 16, 2017).
- (6) Microsoft. SLA for SQL Database [https://azure.microsoft.com/en-us/support/legal/sla/sql-database/v1\\_0/](https://azure.microsoft.com/en-us/support/legal/sla/sql-database/v1_0/) (accessed Dec 16, 2017).
- (7) Microsoft. Pricing - SQL Database <https://azure.microsoft.com/en-us/pricing/details/sql-database/> (accessed Dec 16, 2017).
- (8) DB-Engines. Relational DB - Engines Ranking <https://db-engines.com/en/ranking/relational+dbms> (accessed Dec 16, 2017).
- (9) Microsoft. SQL Database Interfaces <https://azure.microsoft.com/en-us/services/sql-database/> (accessed Dec 16, 2017).
- (10) Unknown. Azure SQL Database Performance Benchmark <https://cbailiss.wordpress.com/2015/01/31/azure-sql-database-v12-ga-performance-inc-cpu-benchmarking/> (accessed Dec 16, 2017).
- (11) Leong, L.; Toombs, D.; Gill, B.; Petri, G.; Haynes, T. Magic Quadrant for Cloud Infrastructure as a Service. **2013**, No. May, 25.
- (12) Microsoft. SQL Database: What is a DTU? <https://docs.microsoft.com/en-gb/azure/sql-database/sql-database-what-is-a-dtu%0A> (accessed Dec 16, 2017).

## 5 Appendix

Full result sets of the conducted performance benchmark tests in ms.

10000 ROWS										
AZURE SQL DATABASE						SQL DATABASE				
ROUND	INSERT	UPDATE	JOIN	SELECT	SELECT	INSERT	UPDATE	JOIN	SELECT	SELECT
1	453	796	62	687	15	93	30	47	31	0
2	281	484	78	1016	30	46	31	15	62	2
3	187	141	78	750	30	93	31	31	15	0
4	405	125	158	829	15	62	46	31	47	1
5	156	110	439	1126	15	61	15	31	63	0
6	453	187	93	1360	31	94	30	31	46	5
7	203	171	47	876	14	62	41	15	31	15
8	327	421	78	1111	30	62	15	31	31	15
9	140	310	63	439	15	62	46	31	62	10
10	157	111	61	609	32	61	15	15	31	0
AVERAGE	276.2	285.6	115.6	880.3	22.7	69.6	30	30.8	41.9	11

1 MILLION ROWS										
AZURE SQL DATABASE						SQL DATABASE				
ROUND	INSERT	UPDATE	JOIN	SELECT	SELECT	INSERT	UPDATE	JOIN	SELECT	SELECT
1	94999	78156	11036	51766	188	2670	672	1015	2596	62
2	95765	77907	10907	49509	745	4046	703	1065	2484	62
3	95453	77703	10238	55352	438	2461	686	969	2469	78
4	95467	78000	11625	51260	203	3377	687	1061	2083	109
5	95407	78328	12153	59648	469	2929	1093	1249	1953	62
6	95468	78301	12243	50511	469	2692	704	1218	2219	62
7	96907	78312	12407	53484	250	2916	671	984	3033	77
8	97344	78093	11156	53473	359	2703	687	1016	2192	62
9	97840	77454	11814	48164	126	3759	671	1032	2125	62
10	97302	77969	11133	48684	281	3175	671	1014	2019	62
AVERAGE	96195.2	78022.3	11471.2	52185.1	355.8	3072.8	724.5	1062.3	2317.3	69.8