






Azure, as a cloud platform, has a lot of storage services, including various SQL databases. This document contains a comparison table of performance results for multi-model database service CosmosDB, using its SQL API, and few most popular SQL databases.

	benchmark (sql1server1benchmark/benchmark)
	mysqlbenchmark
	postgresql1benchmark
	sql1server1benchmark
	ucosqlcosmos

Code in the repository contains Jupiter notebooks to populate databases with test data.

PostgreSQL, SQL Server showed near the same result, however, MySQL was much faster.

Test Data	CosmosDB (SQL API)	PosgreSQL	SQL Server	MySQL (InnoDB)
Dataset: file with 48439 rows, 21 columns, 4.4 MB For benchmark purposes, dataframe with 1000 rows was created basing on the original CSV. https://globaldatalab.org/assets/2019/09/SHDI%20Complete%203.0.csv	Iterative insert of 1000 rows - 335.3 s Read data by key ~ - 0.23168847960 s basing on 500 selects by id	Iterative insert of 1000 rows - 525.8 s Read data by key ~ 0.52203638100 s basing on 500 selects by id	Iterative insert of 1000 rows - 477.8 s Read data by key ~ 0.293667940399 s basing on 500 selects by id	Iterative insert of 1000 rows - 182.1 s Read data by key ~ 0.14684908499999 s basing on 500 selects by id -fastest read by key -fastest insert

<p>Example of database structures</p>	<pre>{ "id": "0", "iso_code": "AFG", "country": "Afghanistan", "year": "2002", "GDLCODE": "Ar101", "level": "Subnat", "region": "Central", "shi": ".454", "healthindex": "58", "incindex": ".394", "edindex": ".409", "lifexp": "57.7", "lgnic": "7.213", "esch": "9.1", "msch": "4.7", "pop": "3443.952", "_rid": "", "_selfAA==/", "_etag": "\"b800871e-0000-0700-0000-6128ef410000\"", "_attachments": "attachments/", "_ts": "1630072641" }</pre>	<pre>CREATE TABLE public.benchmark (id text, iso_code text, country text, year text, gdlcode text, level text, region text, shdi text, healthindex text, incindex text, edindex text, lifexp text, lgnic text, esch text, msch text)</pre>	<pre>CREATE TABLE [dbo].[Benchmark]([id] [varchar](max) NULL, [iso_code] [varchar](max) NULL, [country] [varchar](max) NULL, [year] [varchar](max) NULL, [GDLCODE] [varchar](max) NULL, [level] [varchar](max) NULL, [region] [varchar](max) NULL, [shdi] [varchar](max) NULL, [healthindex] [varchar](max) NULL, [incindex] [varchar](max) NULL, [edindex] [varchar](max) NULL, [lifexp] [varchar](max) NULL, [lgnic] [varchar](max) NULL, [esch] [varchar](max) NULL, [msch] [varchar](max) NULL) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY] GO</pre>	<pre>CREATE TABLE Benchmark('id' Longtext NULL, 'iso_code' Longtext NULL, 'country' Longtext NULL, 'year' Longtext NULL, 'GDLCODE' Longtext NULL, 'level' Longtext NULL, 'region' Longtext NULL, 'shdi' Longtext NULL, 'healthindex' Longtext NULL, 'incindex' Longtext NULL, 'edindex' Longtext NULL, 'lifexp' Longtext NULL, 'lgnic' Longtext NULL, 'esch' Longtext NULL, 'msch' Longtext NULL);</pre>
---------------------------------------	---	--	--	--

SQL Dbs settings:

To reach similar condition for all SQL Dbs, I used tables with max string column types for each database:

text	for PostgreSQL,
[varchar](max)	for SQL Server,
Longtext	for MySQL (InnoDB)

Each result was calculated basing on 3 executions.

Each database was created using the lowest possible performance capabilities in the cloud.

Data was inserted without using bufferisation, however buffesition is possible to use for MySQL and PostgreSQL.

CosmosDB (SQL API) settings:

- 1) Cosmos has default consistency level named as session -

Home > ucusqlcosmos | Default consistency

ucusqlcosmos | Default consistency

Search (Ctrl+V) Save Discard

Overview Activity log Access control (IAM) Tags Diagnose and solve problems Quick start Notifications Data Explorer

Settings Features Replicate data globally Default consistency Backup & Restore Firewall and virtual networks Private Endpoint Connections CORS Dedicated Gateway Keys

STRONG BOUNDED STALENESS **SESSION** CONSISTENT PREFIX EVENTUAL

Session consistency is most widely used consistency level both for single region as well as, globally distributed applications.

Understand Session consistency

It provides write latencies, availability and read throughput comparable to that of eventual consistency but also provides the consistency guarantees that suit the needs of applications written to operate in the context of a user.

Check Microsoft animation for more details
<https://docs.microsoft.com/en-us/azure/cosmos-db/consistency-levels>

2) The lowest possible capacity was used for CosmosDB (Throughput (RU/s) - 400).

Containers		
ID	Database	Throughput (RU/s)
UCUcontainer	ucudatabase	400

3) No replication was enabled for benchmark

ucusqlcosmos | Replicate data globally

Search (Ctrl+V) Save Discard Manual Failover Automatic Failover

Overview Activity log Access control (IAM) Tags Diagnose and solve problems Quick start Notifications Data Explorer

Settings Features Replicate data globally Default consistency Backup & Restore Firewall and virtual networks Private Endpoint Connections CORS Dedicated Gateway

Click on a location to add or remove regions from your Azure Cosmos DB account.
 * Each region is billable based on the throughput and storage for the account. [Learn more](#)

Configure regions
 Multi-region writes: **Disable** **Enable**

Configure the regions for reads, writes and availability zone (supported in selected regions and can only be configured when a new region is added). [+ Add region](#)

Write Region	Availability Zone
West US	

Read Regions	Availability Zone	Action
The account has no read regions.		