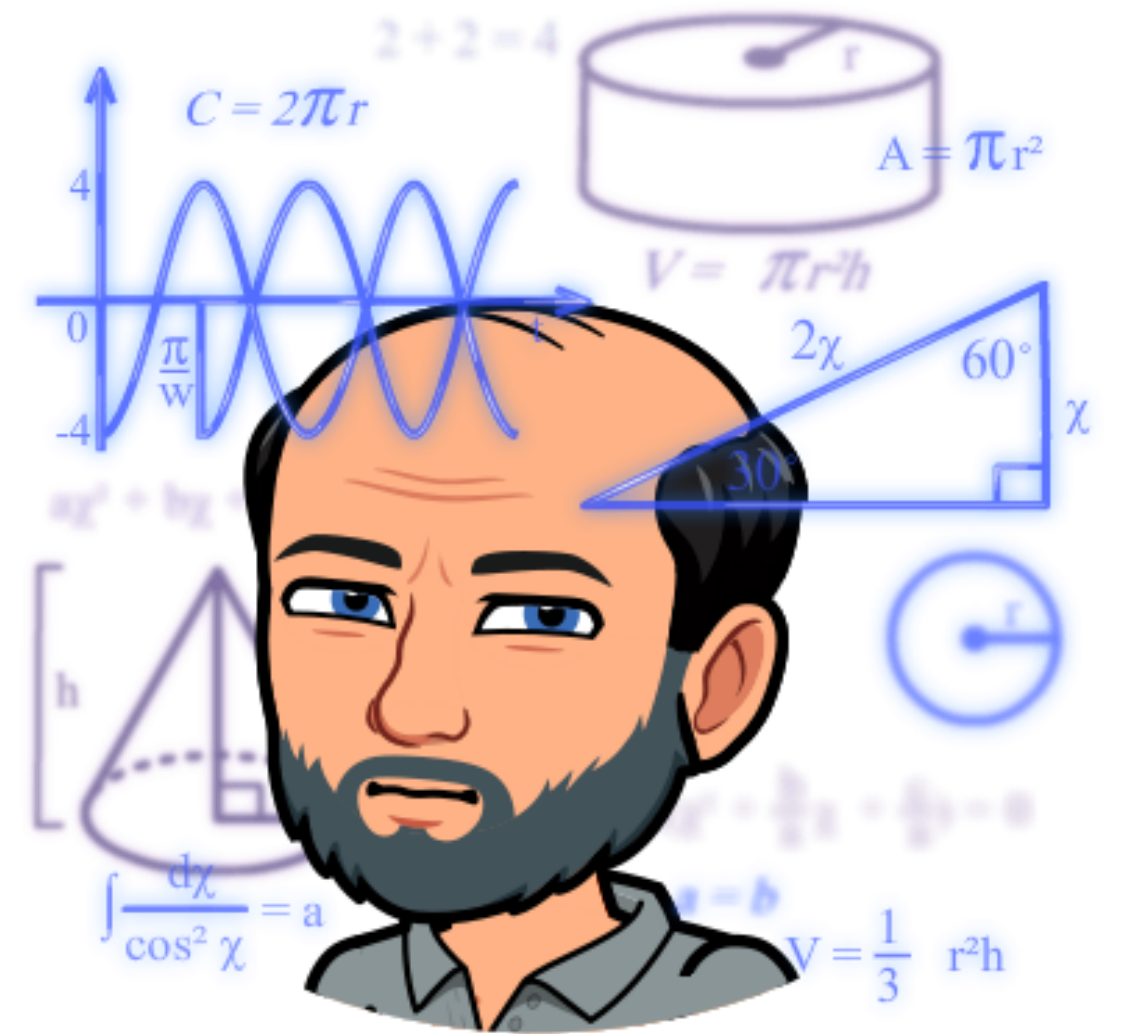


So, what about JSON in my database?



Thomas Hütter

Update Conference Praha 2021

What about JSON in my database?

Thomas Hütter, Diplom-Betriebswirt

- Application developer, consultant, accidental DBA, author
- Worked at consultancies, ISVs, end user companies
- SQL Server > 6.5, former „Navision“ > 3.0, R > 3.1.2
- Speaker at Data&Dev events around Europe

 @DerFredo <https://twitter.com/DerFredo>

 de.linkedin.com/in/derfredo

 www.xing.com/profile/Thomas_Huetter



sqlbits



SQLdays
konferenz



Agenda

- JSON...
 - where it comes from
 - what is it used for
 - what it looks like
- How SQL Server handles JSON
- Comparing with Azure Cosmos DB and MongoDB
- Conclusions
- Credits & resources

JSON... why bother?

- Original motivation: pre-study for a project
- Receive JSON data regularly and automated from an external source
- Data amount: x.000 records/documents per day, each < 10 kB
- Store and later retrieve/analyze the data
- Mainly on-premises environment, but cloud an option
- Project eventually didn't materialize („maybe later...”)
- Therefor no real proof of concept, no cost comparisons
- Author trashed the original slide deck 🙄

JSON... where it comes from

- In the early 2000s, a stateless, real-time server-to-browser data-interchange communication protocol was needed
- First specified by Douglas Crockford as a lightweight alternative to XML
- JSON = JavaScript Object Notation („hipster's XML")
- Based on a subset of JavaScript, but language-independent data format
- Relatively easy to read and write, as it is a plain-text format
- Easy to parse and generate by any programming language
- Current specifications: RFC 8259, IETF Standard 90, ECMA-404
- „The software should be used for good, not evil" is part of the license

JSON... what is it used for

- „Why JSON? - JSON is everywhere!“ (Jovan Popovic, PM at Microsoft)
- transferring data between web servers and applications
- IoT devices send semi-structured data in JSON format
- log and telemetry data may come JSON-formatted
- cloud services use JSON format internally (Azure Stream Analytics)
- user settings are stored in json files (e.g. VS Code or Azure Data Studio)
- popular data format for *NoSQL* databases
- ...

JSON... what it looks like (see also json.org)

- A JSON object is an unordered collection of name-value pairs, enclosed in curly brackets { and } aka braces
- Name-value pairs are separated by a comma ,
- Names are strings surrounded by double quotes " " and followed by a colon :
- For the values there is a limited choice of data types:
 - numbers (signed, decimal fractions, exponential notation)
 - strings (Unicode, double-quote delimited, backslash escaping supported)
 - boolean values (`true` or `false`)
 - empty/unknown value using `NULL`
 - array = ordered list of elements, comma-separated within []
 - another object, so nesting is possible
- Throw in whitespace (incl. linefeeds) almost everywhere

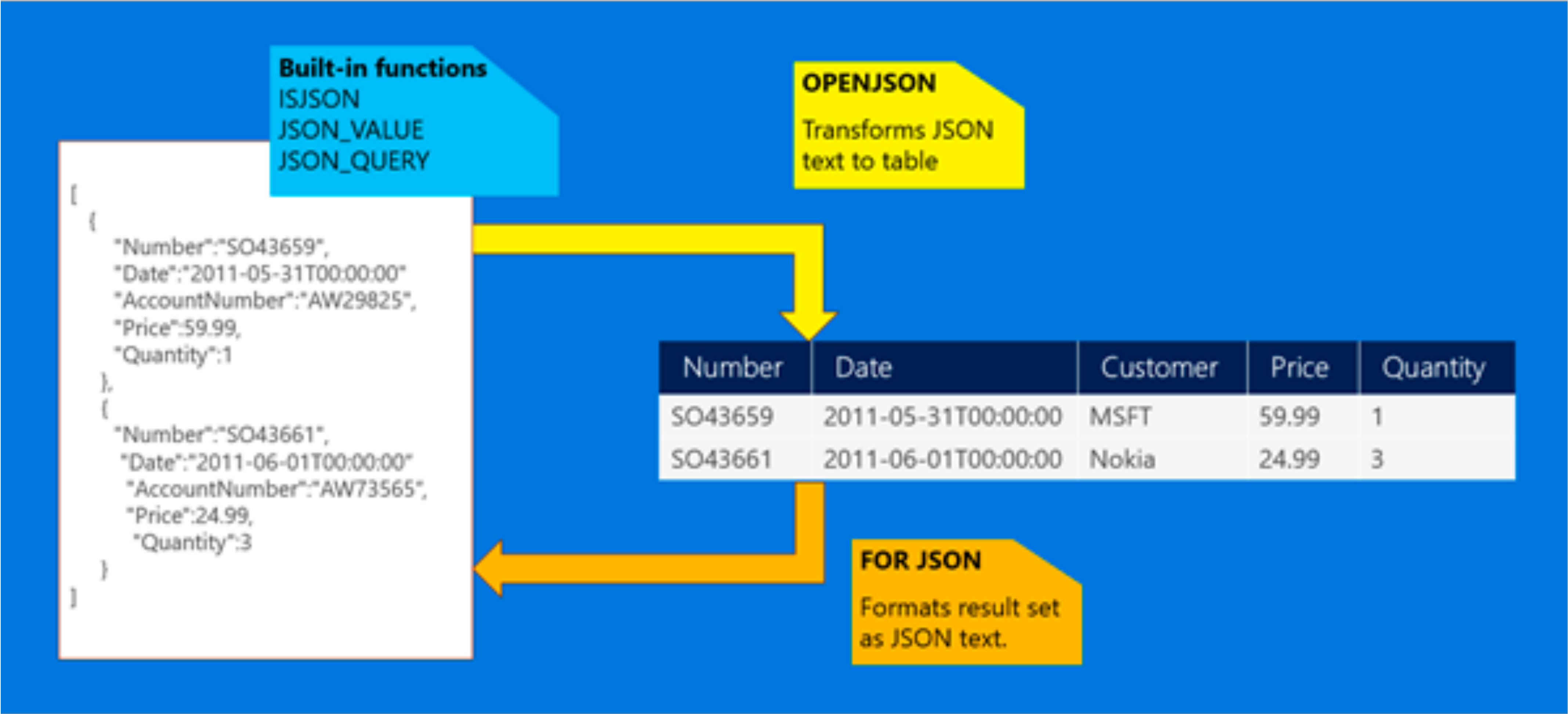
JSON... what it looks like

- `{"event": "Update Conference", "location": "Prague"}`
- `{
 "event": "Update Conference",
 "location": "Prague"
}`
- `{"event": "Update Conference"; "location": "Prague"}`
- `{event: "Update Conference", location: "Prague"}`

JSON... what it looks like

- {
 "ID": 1,
 "CustomerName": "Tailspin Toys (Head Office)",
 "CustomerCategoryName": "Novelty Shop",
 "Contact": {
 "Phone": "(308) 555-0100",
 "Fax": "(308) 555-0101"
 },
 "CityName": "Lisco"
}
- {
 "PersonID": 10,
 "FullName": "Stella Rosenhain",
 "OtherLanguages": ["Dutch", "Finnish", "Lithuanian"]
}

How SQL Server handles JSON



How SQL Server handles JSON

Constructing JSON from relational table data

- `SELECT ... FROM ... FOR JSON AUTO | PATH`
AUTO formats the JSON output automatically,
PATH allows for nesting of complex objects
- `ROOT (...)` adds a single top-level element to the output
- `INCLUDE_NULL_VALUES`
specify this to include JSON properties for NULL values in your output
- `WITHOUT_ARRAY_WRAPPER`
removes the square brackets surrounding the JSON output by default

How SQL Server handles JSON

Storing JSON data in SQL Server

- No dedicated data type for JSON data in SQL Server
- Recommended ways of storing:
 - `NVarChar(4000)` fits on one data page (performance benefit) or
 - `NVarChar(Max)` Unicode data with a size of up to 2GB
- Since this is a text data type that is supported in all subsystems of SQL Server, you can take advantage of column store indexes, memory optimized tables, external files, Polybase, ...

How SQL Server handles JSON

Retrieving JSON data from SQL Server

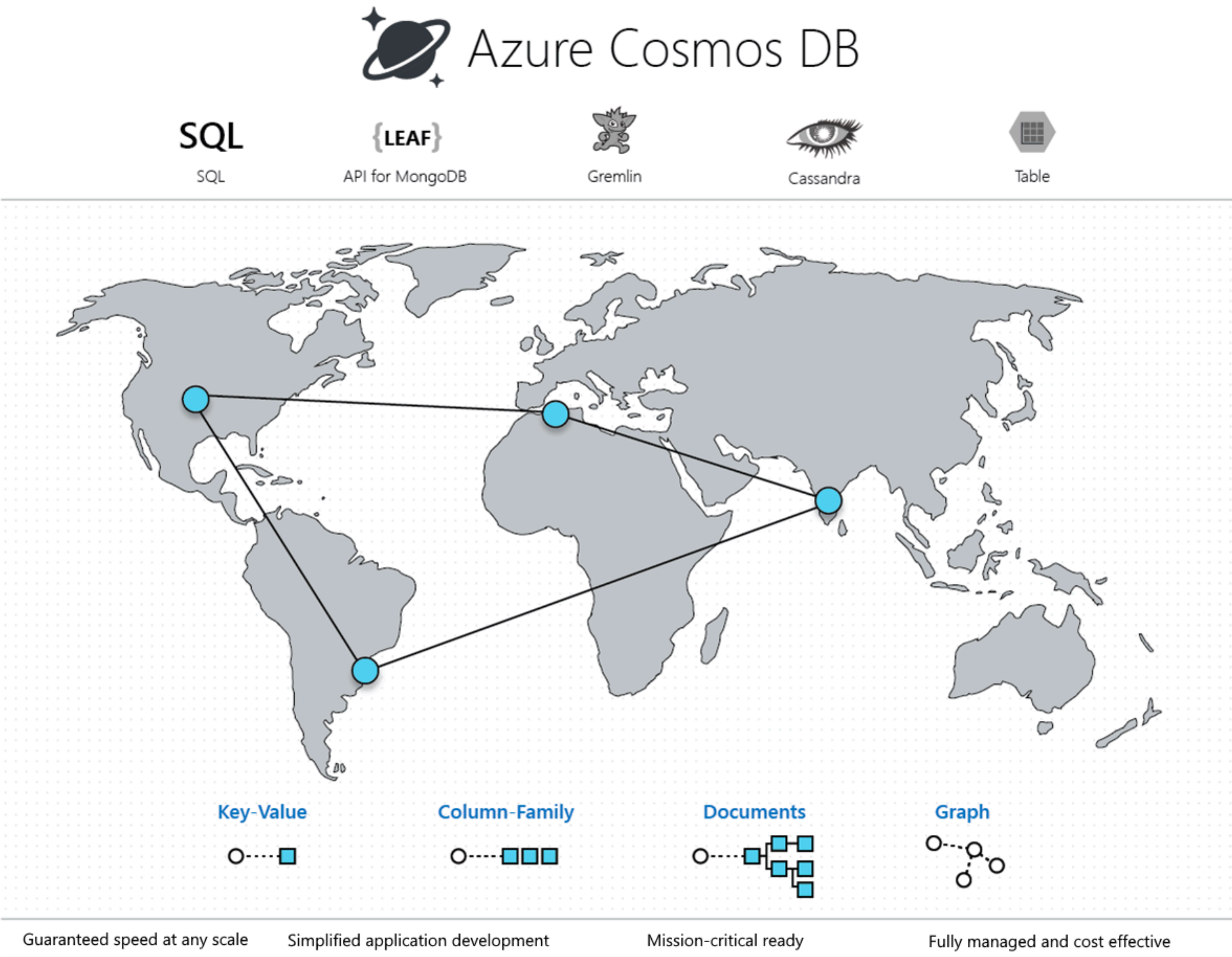
- `ISJSON()` checks for valid JSON syntax
- `OPENJSON()` to transform (simple) JSON to relational format, by default returns key, value and type of your JSON data
- `JSON_VALUE()` extracts one scalar value from a JSON object/text, returns a single text value of type `NVarChar(4000)`
- `JSON_QUERY()` to extract an array or an object from a JSON string
- Results can be further processed just like T-SQL

How SQL Server handles JSON

Modifying JSON data within SQL Server

- `JSON_MODIFY()` updates the value of a property in a JSON string and returns the updated JSON string
- can be used to modify existing values or append new key-value pairs
- in *strict* mode, the referenced property must exist
- in *lax* mode (default):
 - if the property does not exist, `JSON_Modify` tries to insert the value on the specified path
 - the specified key will be deleted if the new value is NULL

Comparing with Azure Cosmos DB and MongoDB



Comparing with Azure Cosmos DB and MongoDB

- Cosmos DB is a distributed multi-model NoSQL database service, exposing the data model (items in containers) through several APIs
- Developed by *Microsoft Corporation*, Redmond, Washington/USA
- Initial release in May 2017
- License: Proprietary; free tier 1000 RU/s and 25 GB of storage (as of Nov 2021)
- APIs available:
 - native Core(SQL) API - enables queries in SQL syntax
 - MongoDB API - wire-compatible to MongoDB Server version 4.0 (as of Nov 2021)
 - Cassandra API - enables Apache Cassandra apps
 - Gremlin API - graph database
 - Table API - overcoming limitations of Azure Table Storage

Comparing with Azure Cosmos DB and MongoDB

●Managing your Azure CosmosDB

Home > Azure Cosmos DB >

cosmos

Azure Cosmos DB API for MongoDB account

Search (Cmd+/)

«

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Cost Management

Quick start

Notifications

Data Explorer

Settings

Connection String

Features

Replicate data globally

Default consistency

Backup & Restore

Firewall and virtual networks

Private Endpoint Connections

Data Migration

+ Add Collection

↺ Refresh

→ Move

↻ Data Migration

🗑 Delete Account

🔗 Data Explorer

🌐 Enable geo-redundancy

📘 Welcome to your Azure Cosmos DB Free Tier account! Your first 1000 RU/s and 25 GB of storage will be free for the lifetime of this account. [Click here to learn more.](#)

^ Essentials

Status : Online

Resource group (Move) : RGCosmos

Subscription (Move) : Azure subscription 1

Subscription ID :

Total throughput limit : No total throughput limit

Read Locations : West Europe

Write Locations : West Europe

URI : https://.azure.com:443/

Server Version : 4.0

Free Tier Discount : Opted In

See more

Collections

ID	Database	Throughput (RU/s)
Sats	Data	400 (Shared)
JSONDemo	Data	400

Monitoring

Show data for last

1 hour24 hours7 days30 days

Number of requests per 5 minutes ⓘ

2,4

2,2

Find

Request Charge ⓘ

18

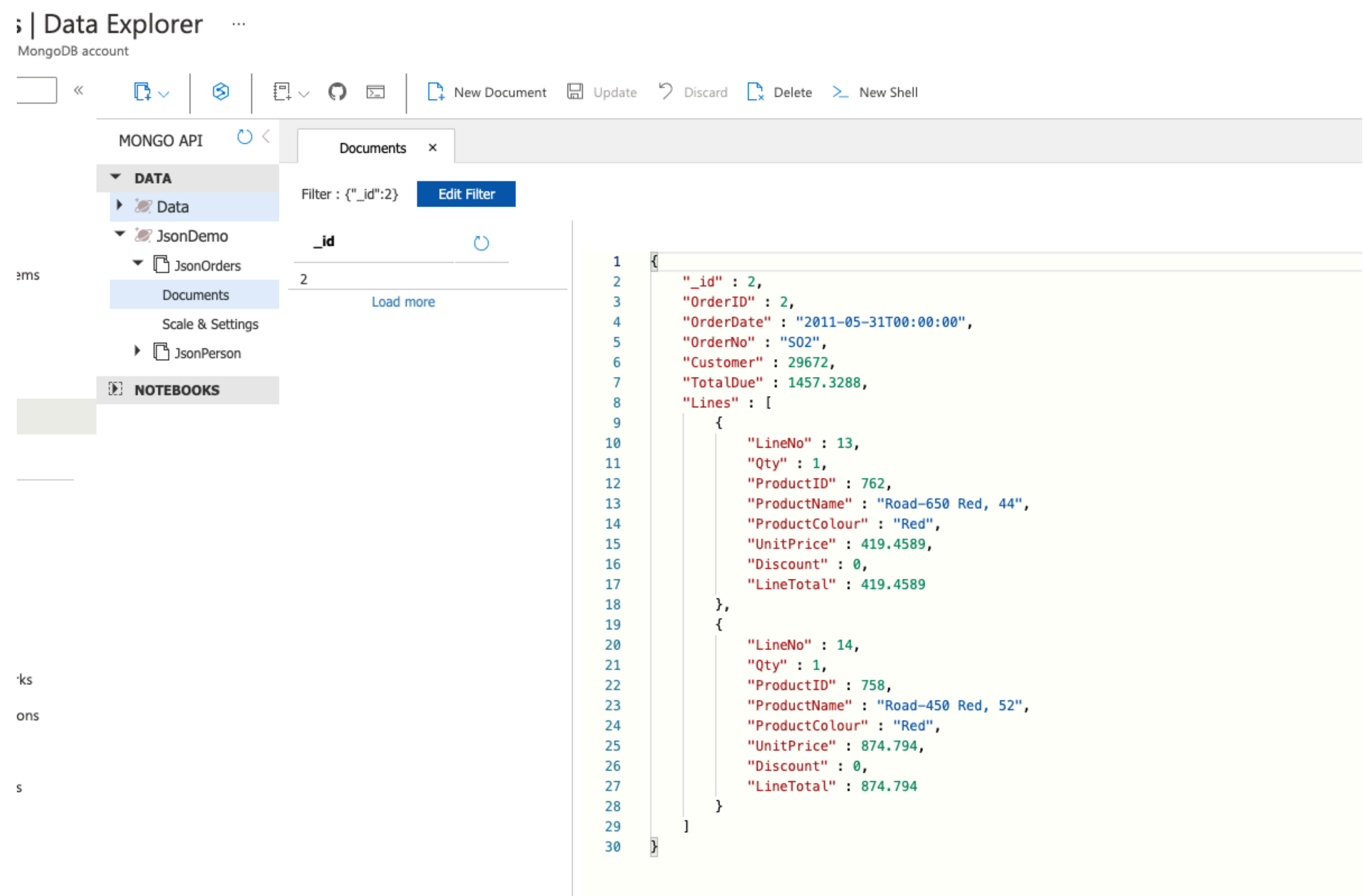
16

Find

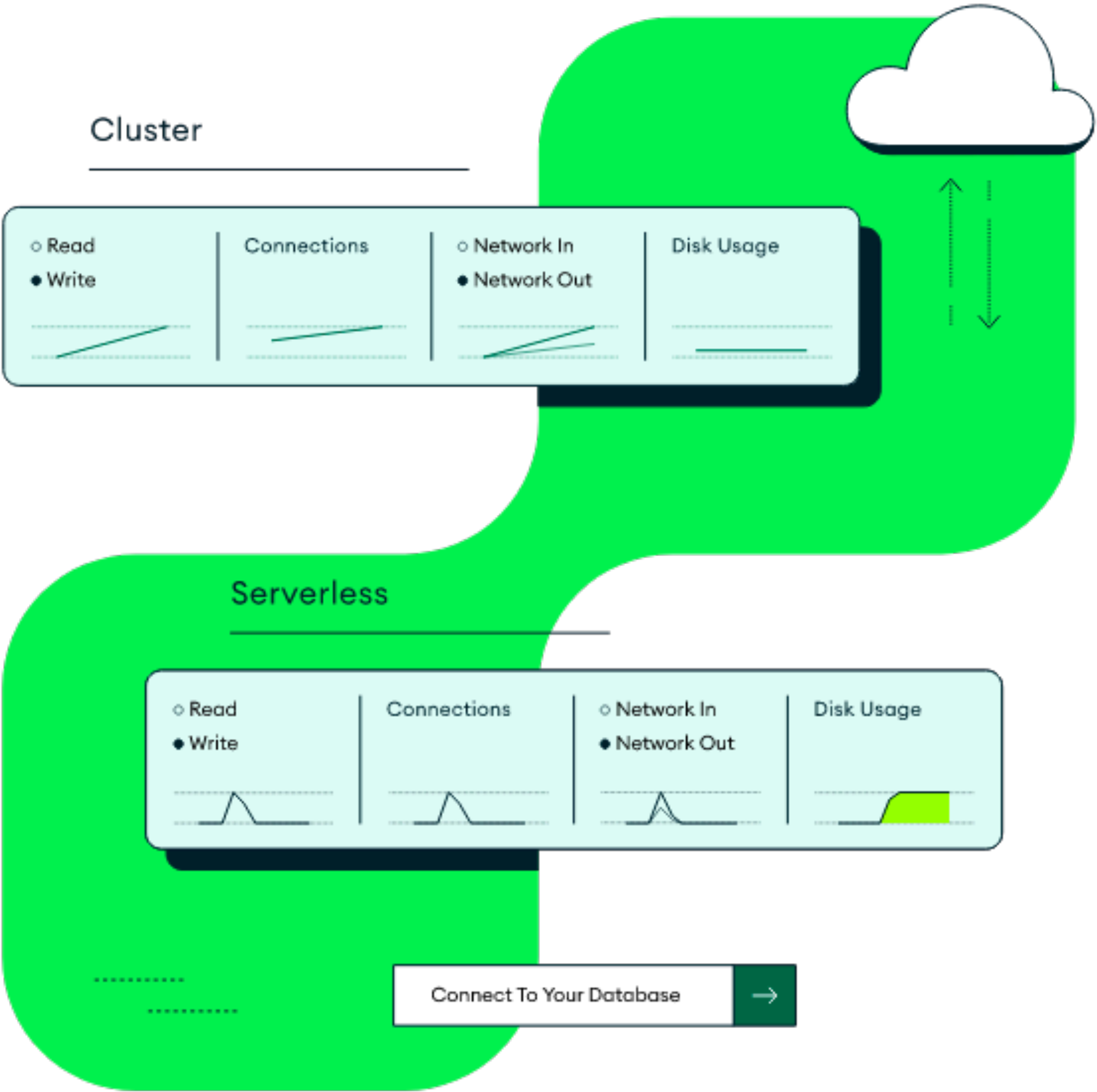
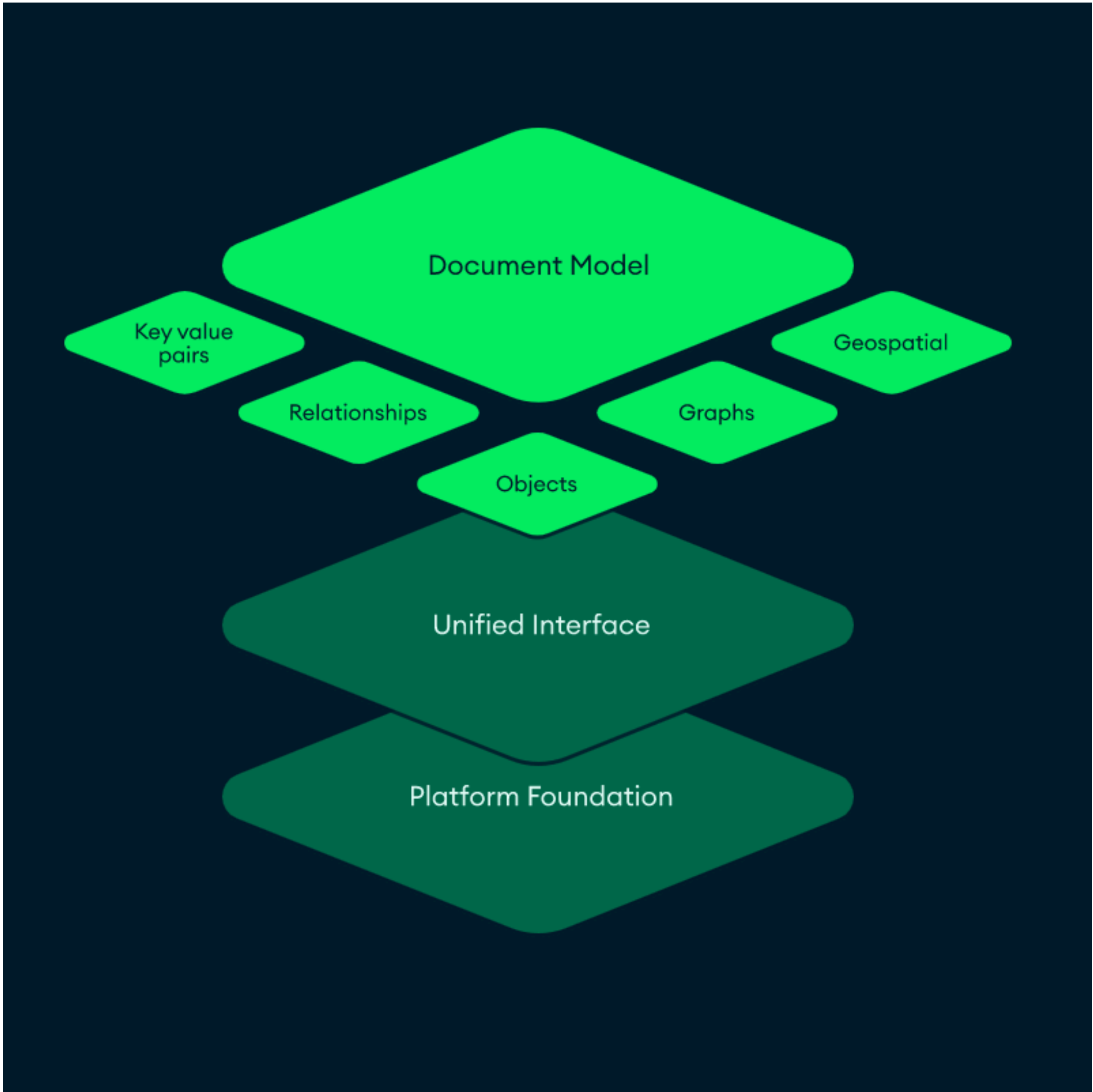
Estimated Cost (hourly) ⓘ

Comparing with Azure Cosmos DB and MongoDB

- Accessing data in Azure CosmosDB via Data Explorer



Comparing with Azure Cosmos DB and MongoDB





Comparing with Azure Cosmos DB and MongoDB

- MongoDB is a multi-platform document-oriented NoSQL database program using JSON style documents with optional schemas
- Developed by *MongoDB Inc.* (former *10gen Inc.*), New York City/USA
- Initial release in Feb 2009, current version (as of Nov 2021) 5.0
- License: SSPL (Server-side public license), source available
- Editions available:
 - Community Server - free (for Linux, MacOS, Windows)
 - Enterprise Server - commercial
 - MongoDB Atlas - on AWS, Microsoft Azure, Google Cloud Platform

Comparing with Azure Cosmos DB and MongoDB

- Provisioning a MongoDB environment

MDBU



Access Manager

Billing

All ClustersGet HelpThomas

ORGANIZATION

Projects

Alerts 0

Activity Feed

Settings

Access Manager

Billing

Support

Live Migration

MDBU

Projects


New Project

Find a project...

Project Name	Clusters	Users	Teams	Alerts	Actions
JsonDemo	1 Cluster	2 Users	0 Teams	0 Alerts	<div>...</div> <div></div>
M001	1 Cluster	2 Users	0 Teams	0 Alerts	<div>...</div> <div></div>
SQLSat	1 Cluster	1 User	0 Teams	0 Alerts	<div>...</div> <div></div>

System Status: All Good Last Login: 84.133.194.141

©2021 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales



Comparing with Azure Cosmos DB and MongoDB

- Creates a 3-node cluster

MDBU

Access Manager

Billing

All Clusters

Get Help

Thomas

Praha21

Atlas

Realm

Charts

DEPLOYMENT

Databases

Triggers

Data Lake

SECURITY

Database Access

Network Access

Advanced

MDBU > PRAHA21 > DATABASES

ClusterFra

VERSION 4.4.10

REGION AWS Frankfurt (eu-central-1)

CLUSTER TIER M0 Sandbox (General)

OverviewReal TimeMetricsCollectionsSearchProfilerPerformance AdvisorOnline ArchiveCmd Line Tools

SANDBOXNODESREPLICA SETCONNECTCONFIGURATION

REGION Frankfurt (eu-central-1)

clusterfra-shard-00-00.3ol...SECONDARY

clusterfra-shard-00-01.3ol...PRIMARY

clusterfra-shard-00-02.3ol...SECONDARY

This is a Shared Tier Cluster

If you need a database that's better for high-performance production applications, upgrade to a dedicated cluster.

Upgrade

Logical Size 329.4 MB

512.0 MB max

0.0 B

Last 6 Hours

Operations R: 0.01 W: 0

723.1/s

Last 6 Hours

Connections 7

500 max

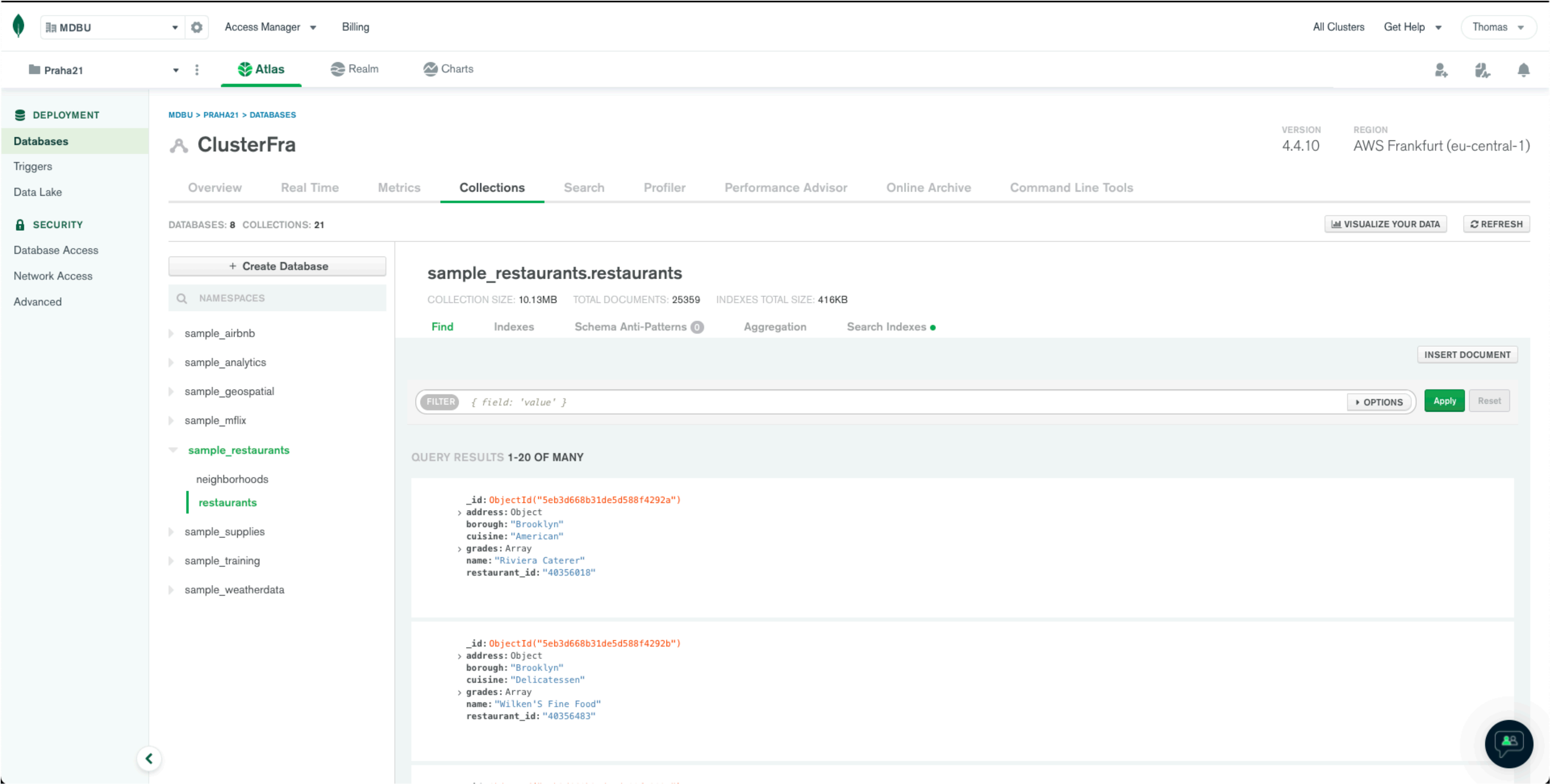
Last 6 Hours

System Status: All Good

©2021 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

Comparing with Azure Cosmos DB and MongoDB

- Installed the sample databases



Comparing with Azure Cosmos DB and MongoDB

● Using MongoDB Compass to access your data

The screenshot displays the MongoDB Compass application window titled "MongoDB Compass - Praha21/JsonDemo.JsonOrders". The left sidebar shows the database structure for "Praha21", including 12 DBS and 31 COLLECTIONS. The "JsonDemo" database is expanded, showing the "JsonOrders" collection. The main panel displays the "JsonDemo.JsonOrders" collection with 31.5k documents, a total size of 23.3MB, and an average size of 742B. The "Documents" tab is active, showing a filter bar with the filter "{OrderID: 2}" and a "FIND" button. Below the filter bar, there are buttons for "ADD DATA", "VIEW", and a toggle for document format (JSON, GridFS, etc.). The document content is displayed in a JSON format, showing a single document with fields like "_id", "OrderID", "OrderDate", "OrderNo", "Customer", "TotalDue", "Lines", "LineNo", "Qty", "ProductID", "ProductName", "ProductColour", "UnitPrice", "Discount", and "LineTotal".

MongoDB Compass - Praha21/JsonDemo.JsonOrders

Praha21

12 DBS 31 COLLECTIONS

★ FAVORITE

HOSTS

clusterfra-shard-00-00.3ol7s.mongodb.net:27017
clusterfra-shard-00-01.3ol7s.mongodb.net:27017
clusterfra-shard-00-02.3ol7s.mongodb.net:27017

CLUSTER

Replica Set (atlas-68afbv-shard-0)
3 Nodes

EDITION

MongoDB 4.4.10 Enterprise

Filter your data

JsonDemo

JsonOrders

admin
config
local
sample_airbnb
sample_analytics
sample_geospatial
sample_mflix
sample_restaurants
sample_supplies

JsonDemo.JsonOrders

DOCUMENTS 31.5k TOTAL SIZE 23.3MB AVG. SIZE 742B INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER {OrderID: 2} OPTIONS FIND

ADD DATA VIEW

Displaying documents 1 - 1 of 1

```
{
  "_id": 2,
  "OrderID": 2,
  "OrderDate": "2011-05-31T00:00:00",
  "OrderNo": "S02",
  "Customer": 29672,
  "TotalDue": 1457.3288,
  "Lines": [
    {
      "LineNo": 13,
      "Qty": 1,
      "ProductID": 762,
      "ProductName": "Road-650 Red, 44",
      "ProductColour": "Red",
      "UnitPrice": 419.4589,
      "Discount": 0,
      "LineTotal": 419.4589
    },
    {
      "LineNo": 14,
      "Qty": 1,
      "ProductID": 758,
      "ProductName": "Road-450 Red, 52",
      "ProductColour": "Red",
      "UnitPrice": 874.794,
      "Discount": 0,
      "LineTotal": 874.794
    }
  ]
}
```

Comparing with Azure Cosmos DB and MongoDB

- Atlas - cloud version of the MongoDB database on AWS, Azure, Google
 - Realm - build mobiles apps, web sites, services to connect to MongoDB
 - Charts - build dashboards and charts on your data
 - Data Lake - „serverless“, scalable analyzing query engine
- Enterprise Server
 - commercial edition, includes additional capabilities
- Community Server
 - on-premises version of MongoDB Atlas
- MongoDB Compass
 - the GUI client/tool for local installation

Comparing with Azure Cosmos DB and MongoDB

	SQL Server	CosmosDB	MongoDB
Primary DB model	Relational	Document/Graph/ Key-value/Table	Document Store
Initial release, license	1989, commercial	2017, commercial	2009, open source
Cloud-based/on-prem	yes / yes	yes / no	yes / yes
Operating system	Windows, Linux, Docker	(hosted)	Windows, Linux, MacOS
Supported program. languages	~ 12	~ 6 + Mongo-driven	~ 30
DB-ranking (*) overall/ document stores	3 / -	26 / 3	5 / 1
Partitioning/replic ation	yes / yes	yes / yes	yes / yes
Consistency	immediate	immediate, eventual	immediate, eventual
Scalability	on-prem vs Azure	„instant, automatic“	vertical & horizontal

(*) according to db-engines.com, Nov 2021

Conclusions for handling JSON in your database

● **SQL Server**

- on-premises or in Azure Cloud, probably „already there“
- no dedicated (optimized) data type for handling JSON
- integrates well with relational data queries

● **Azure Cosmos DB** (with Mongo API)

- MS cloud only, no on-premises solution
- integrates well with other Azure services
- will scale mostly „without“ manual input

● **MongoDB**

- on-premises, or in one of the big 3 cloud environments
- „the original“ when it comes to JSON-style document store
- scalable from free to Enterprise Server, not automatically

Credits & resources

- General info: <https://www.json.org/json-en.html>, <https://en.wikipedia.org/wiki/JSON>
- SQL Server: <https://docs.microsoft.com/en-us/sql/relational-databases/json/json-data-sql-server>
- Cosmos DB: <https://docs.microsoft.com/en-us/azure/cosmos-db/introduction>,
<https://docs.microsoft.com/en-us/azure/cosmos-db/sql/sql-query-working-with-json>,
<https://docs.microsoft.com/en-us/azure/cosmos-db/mongodb/mongodb-introduction>
- MongoDB: <https://docs.mongodb.com/>
- Comparisons:
<https://db-engines.com/en/system/Microsoft+Azure+Cosmos+DB;Microsoft+SQL+Server;MongoDB>,
<https://sourceforge.net/software/compare/Azure-Cosmos-DB-vs-MongoDB-vs-SQL-Server/>,
<https://www.trustradius.com/compare-products/azure-cosmos-db-vs-mongodb-atlas>

What about JSON in my database?

Thank you for your time and interest & please keep in touch:

 @DerFredo <https://twitter.com/DerFredo>

 de.linkedin.com/in/derfredo

 www.xing.com/profile/Thomas_Huetter



This file and the demo script can be found at:

<https://j.mp/DerFredoUpd21>