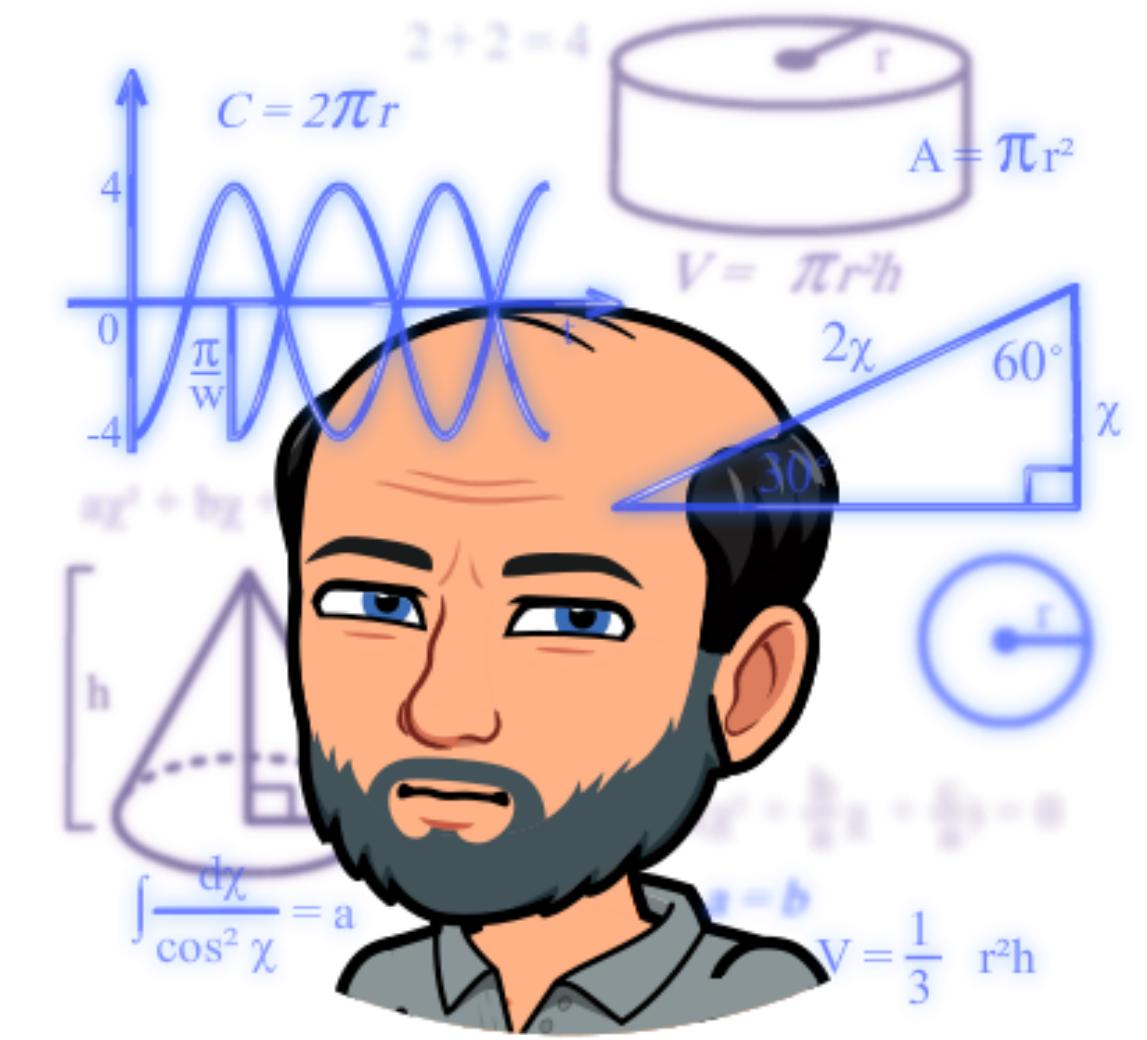


# So, what about JSON in my database?



Thomas Hütter

Data Saturday Rheinland  
St Augustin 2023

# What about JSON in my database?

Thomas Hütter, Diplom-Betriebswirt

- Developer for ERP apps, SQL scripts and BI stuff
- Worked at consultancies, ISVs, end user companies
- 1995: SQL Server, 2001: Nav/BC, 2014: R, 2020: Power\*
- Speaker at SQL / data / dev events around Europe



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

 [de.linkedin.com/in/derfredo](https://de.linkedin.com/in/derfredo)

 <https://techhub.social/@DerFredo>



# sqlbits

{ } NDC  
Conferences



Pure expertise at the  
SQL Server Konferenz 2018  
FEB 26th - 28th, 2018 | DARMSTADT

Sign up now at [sqlkonferenz.de](http://sqlkonferenz.de)



# 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](http://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

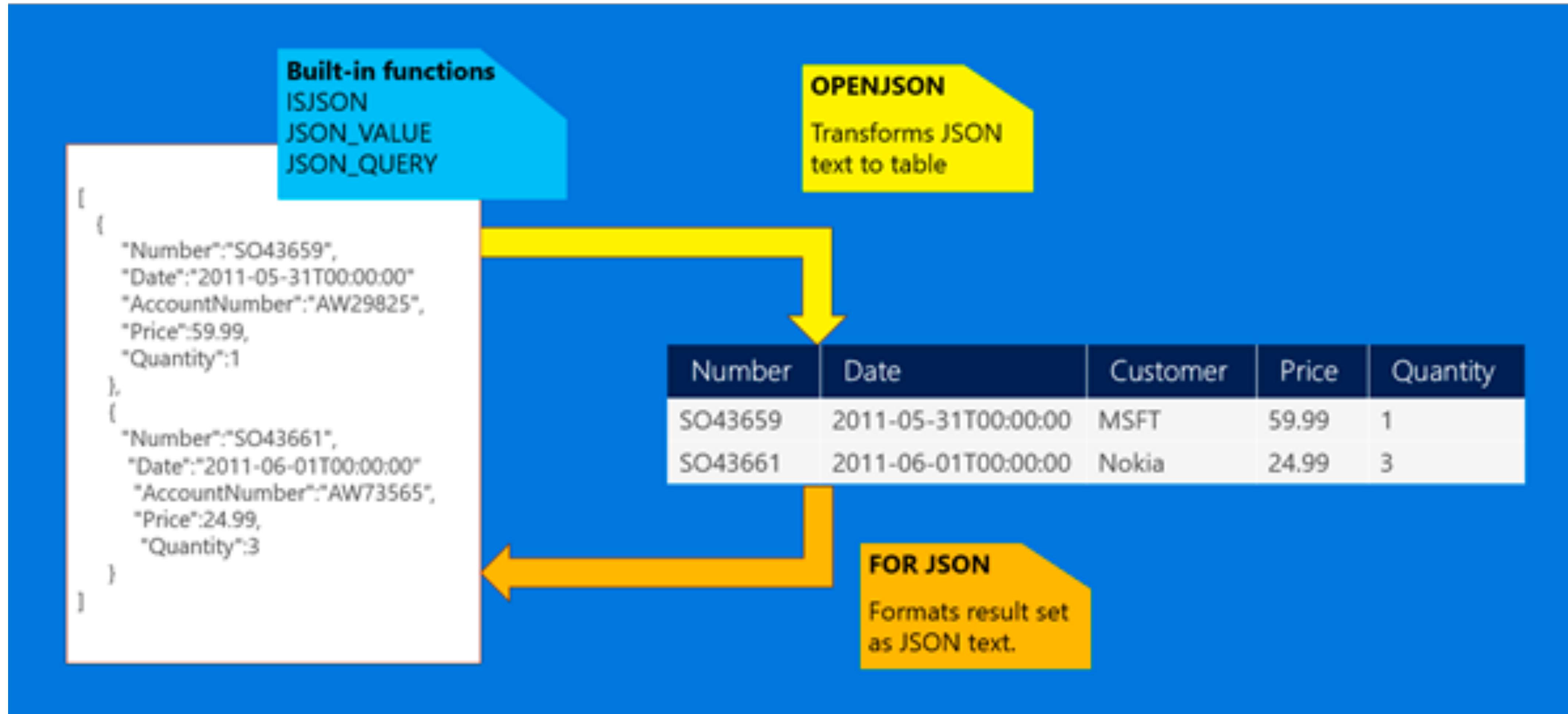
# JSON... what it looks like

- { "event": "Sat Rheinland 2023", "location": "St Augustin" }
- {  
  "event": "Sat Rheinland 2023",  
  "location": "St Augustin"  
}
- {"event": "Sat Rheinland 2023"; "location": "St Augustin"}
- {event: "Sat Rheinland 2023", location: "St Augustin"}

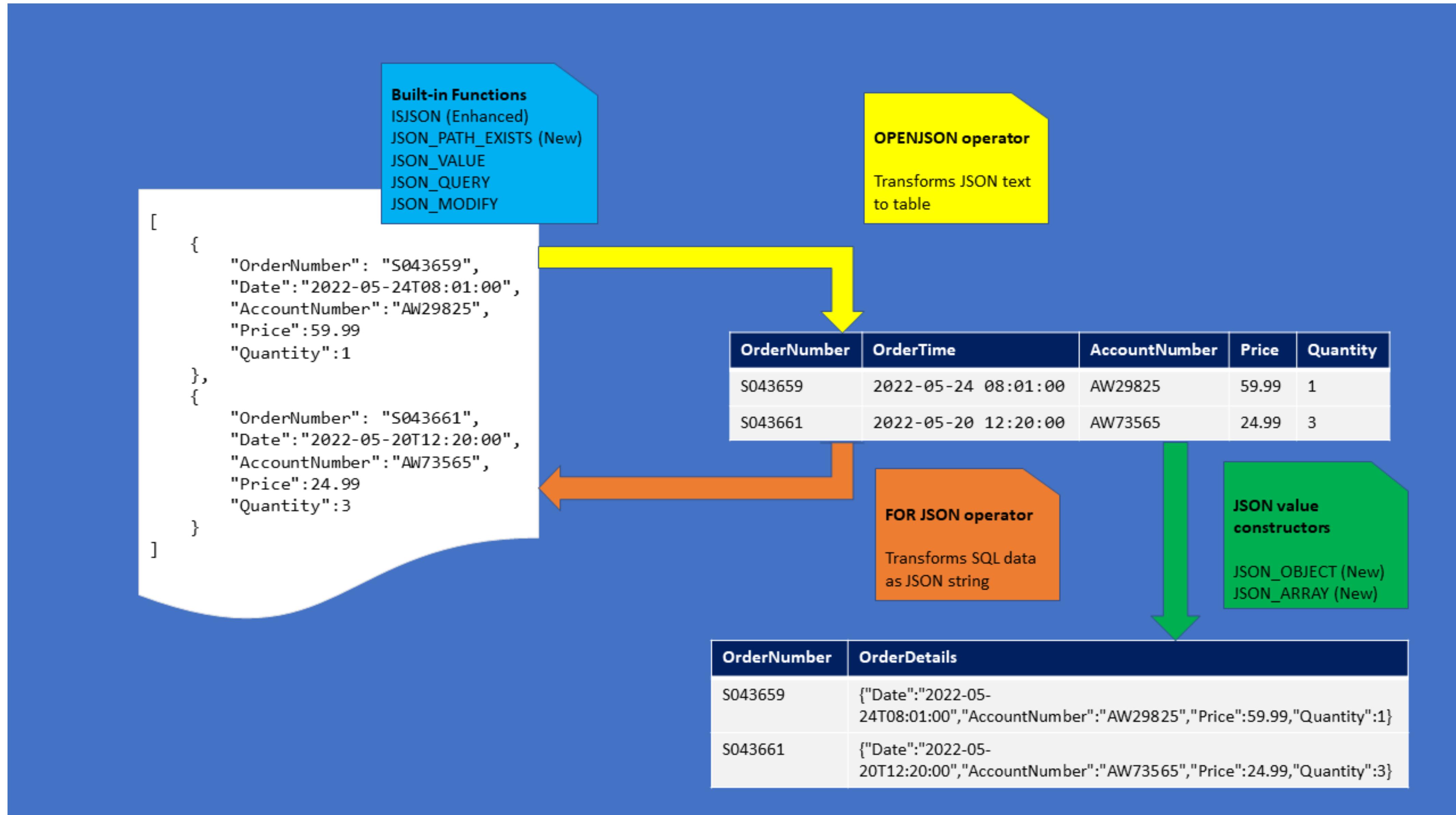
# JSON... what it looks like

- {  
  "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 < 2022 handles JSON



# How SQL Server 2022 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
- `JSON_OBJECT()` and `JSON_ARRAY()`  
new functions in [SQL Server 2022](#) to generate JSON output

# How SQL Server handles JSON

## Storing JSON data in SQL Server

- No dedicated data type for JSON data in SQL Server (... yet 😊)
- 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

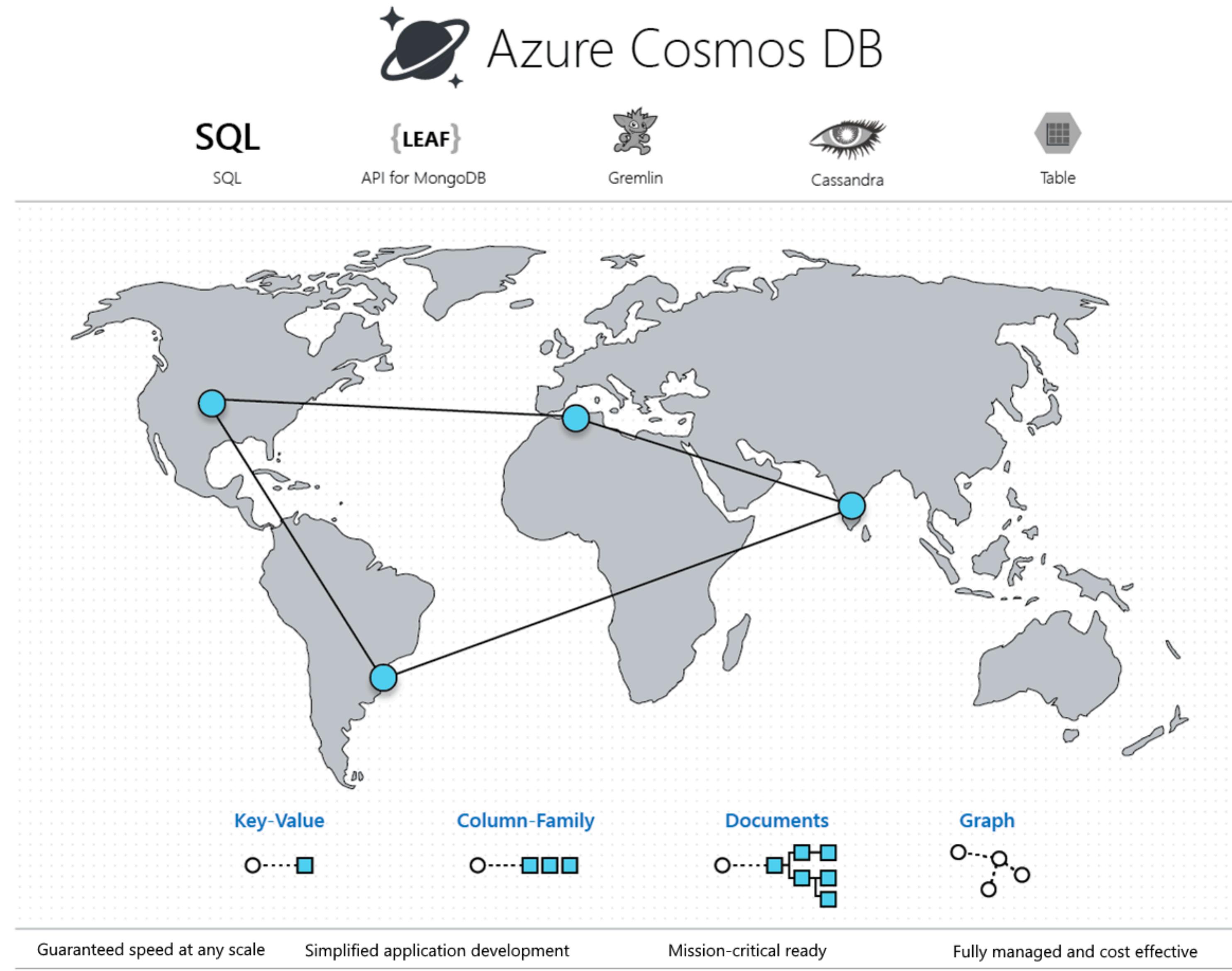
- `ISJSON()` checks for valid JSON syntax, additional parameters in [SQL 2022](#):  
`ISJSON(SomeVar, VALUE | OBJECT | ARRAY | SCALAR)`
- New in [SQL 2022](#): `JSON_PATH_EXISTS(SomeVar, '$.SomePath')`
- `OPENJSON()` to transform (simple) JSON to relational format,  
by default returns key, value and 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 June 2023)
- APIs available:
  - native Core(SQL) API - enables queries in SQL syntax
  - MongoDB API - wire-compatible to MongoDB Server version 4.2/[5.0] (as of June 2023)
  - 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

The screenshot shows the Azure Cosmos DB API for MongoDB account overview page. The left sidebar contains navigation links for 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, and Data Migration. The main content area displays the following information:

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**

| Setting                | Value                     | Setting            | Value   |
|------------------------|---------------------------|--------------------|---|
| Status                 | Online                    | Read Locations     | West Europe   |
| Resource group (Move)  | RGCosmos                  | Write Locations    | West Europe   |
| Subscription (Move)    | Azure subscription 1      | URI                | <a href="https://.azure.com:443/">https://.azure.com:443/</a> |
| Subscription ID        | (redacted)                | Server Version     | 4.0   |
| Total throughput limit | No total throughput limit | Free Tier Discount | Opted In  |

**Collections**

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

**Monitoring**

Show data for last [1 hour](#) **24 hours** [7 days](#) [30 days](#)

**Number of requests per 5 minutes** ⓘ

| Time | Requests |
|------|----------|
| 2,4  | Find     |
| 2,2  |          |

**Request Charge** ⓘ

| Time | Charge |
|------|--------|
| 18   | Find   |
| 16   |        |

**Estimated Cost (hourly)** ⓘ

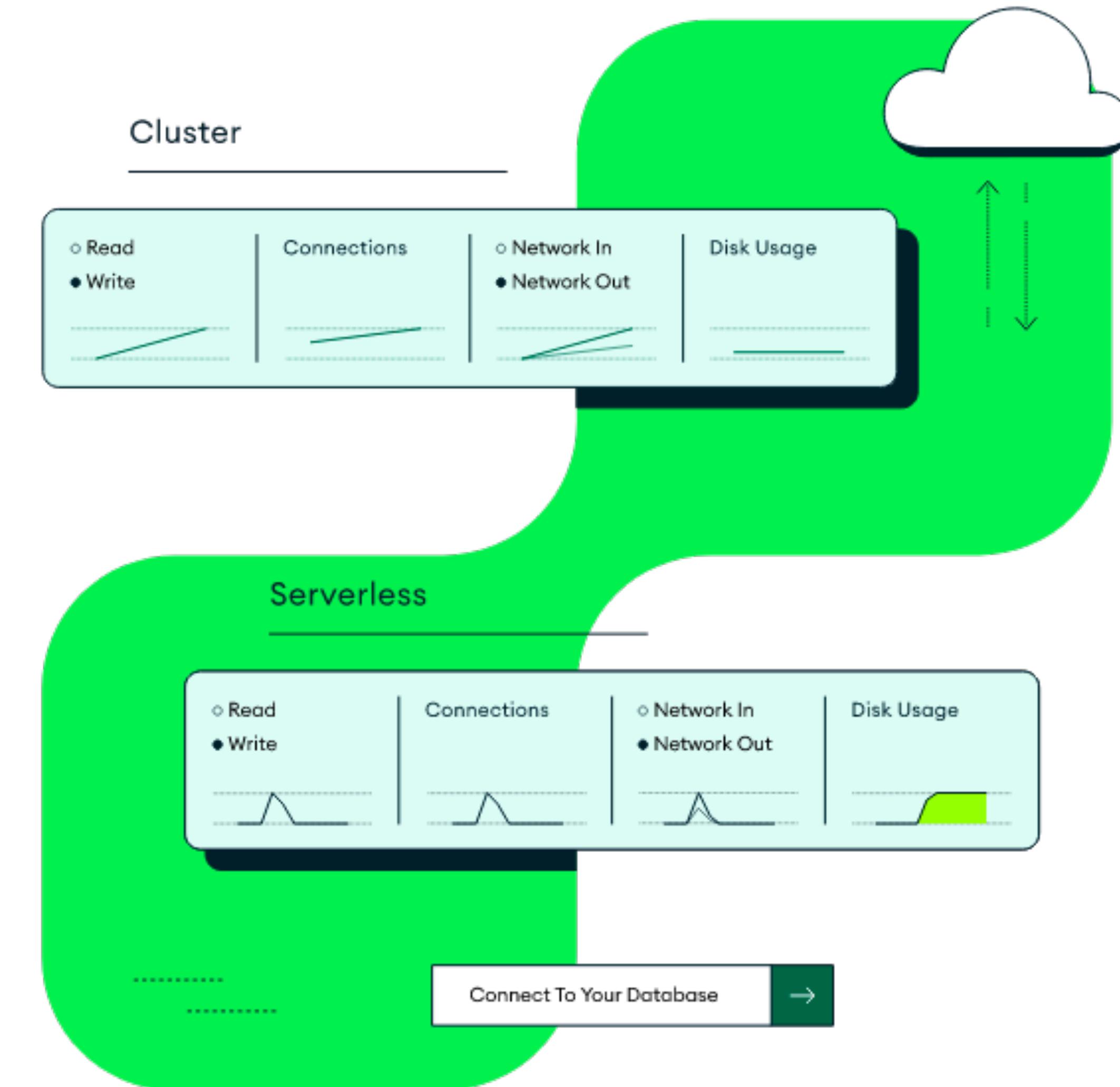
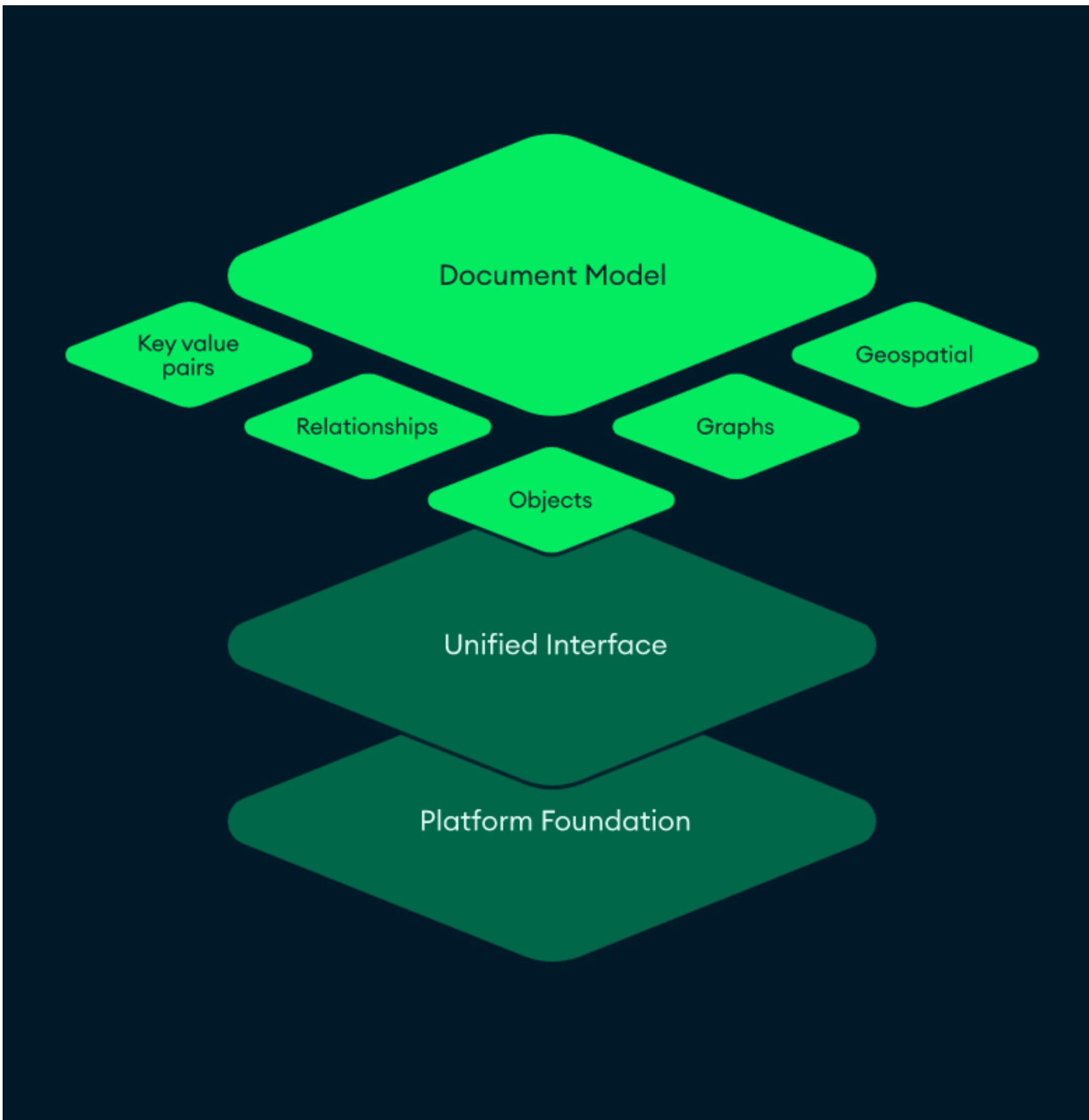
# Comparing with Azure Cosmos DB and MongoDB

- Accessing data in Azure CosmosDB via Data Explorer

The screenshot shows the Azure Cosmos DB Data Explorer interface. The left sidebar has sections for 'DATA' (selected), 'NOTBOOKS', and 'SCALE & SETTINGS'. Under 'DATA', there's a 'MongoDB account' section with 'Data', 'JsonDemo' (selected), and 'JsonOrders' (selected). 'JsonOrders' contains 'Documents' and 'Scale & Settings'. The main area shows a 'Documents' tab with a filter bar set to '\_id: 2'. The document details pane shows the following JSON:

```
1  {
2    "_id": 2,
3    "OrderID": 2,
4    "OrderDate": "2011-05-31T00:00:00",
5    "OrderNo": "SO2",
6    "Customer": 29672,
7    "TotalDue": 1457.3288,
8    "Lines": [
9      {
10        "LineNo": 13,
11        "Qty": 1,
12        "ProductID": 762,
13        "ProductName": "Road-650 Red, 44",
14        "ProductColour": "Red",
15        "UnitPrice": 419.4589,
16        "Discount": 0,
17        "LineTotal": 419.4589
18      },
19      {
20        "LineNo": 14,
21        "Qty": 1,
22        "ProductID": 758,
23        "ProductName": "Road-450 Red, 52",
24        "ProductColour": "Red",
25        "UnitPrice": 874.794,
26        "Discount": 0,
27        "LineTotal": 874.794
28      }
29    ]
30  }
```

# 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 stable version (as of June 2023) 6.0.6
- 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

The screenshot shows the MongoDB Atlas interface for managing projects. The top navigation bar includes 'MDBU' (selected), 'Access Manager', 'Billing', 'All Clusters', 'Get Help', and a user profile for 'Thomas'. The left sidebar has links for 'ORGANIZATION' (Projects selected), 'Alerts (0)', 'Activity Feed', 'Settings', 'Access Manager', 'Billing', 'Support', and 'Live Migration'. The main 'Projects' section displays a table with three rows:

| Project Name | Clusters  | Users   | Teams   | Alerts   | Actions |
|--------------|-----------|---------|---------|----------|---------|
| JsonDemo     | 1 Cluster | 2 Users | 0 Teams | 0 Alerts | ...     |
| M001         | 1 Cluster | 2 Users | 0 Teams | 0 Alerts | ...     |
| SQLSat       | 1 Cluster | 1 User  | 0 Teams | 0 Alerts | ...     |

At the bottom, there's a footer with 'System Status: All Good Last Login: 84.133.194.141' and links for 'Status', 'Terms', 'Privacy', 'Atlas Blog', and 'Contact Sales'. A circular icon with a person icon is also present.

# Comparing with Azure Cosmos DB and MongoDB

- Creates a 3-node cluster

The screenshot shows the MongoDB Atlas Cluster Overview page for a cluster named 'ClusterFra'. The top navigation bar includes 'MDBU', 'Access Manager', 'Billing', 'All Clusters', 'Get Help', and a user profile for 'Thomas'. The left sidebar has sections for 'DEPLOYMENT' (selected), 'Databases' (selected), 'Triggers', 'Data Lake', 'SECURITY' (Database Access, Network Access, Advanced), and 'MongoDB Cloud Shell'. The main content area shows the cluster details: Version 4.4.10, Region AWS Frankfurt (eu-central-1), and Cluster Tier M0 Sandbox (General). It lists three nodes: 'clusterfra-shard-00-00.30...', 'clusterfra-shard-00-01.30...', and 'clusterfra-shard-00-02.30...', all marked as Secondary. A note states 'This is a Shared Tier Cluster' and suggests upgrading to a dedicated cluster. Below this are charts for 'Operations' (R: 0.01 W: 0, 723.1/s) and 'Connections' (7, 500 max). A 'Logical Size' chart shows 329.4 MB over the last 6 hours, with a peak of 512.0 MB. The bottom footer includes 'System Status: All Good', copyright information (©2021 MongoDB, Inc.), and links for Status, Terms, Privacy, Atlas Blog, Contact Sales.

MDBU Access Manager Billing All Clusters Get Help Thomas

Praha21 Atlas Realm Charts

**DEPLOYMENT**

**Databases** (Selected)

Triggers Data Lake

**SECURITY**

Database Access Network Access Advanced

**ClusterFra**

VERSION 4.4.10 REGION AWS Frankfurt (eu-central-1) CLUSTER TIER M0 Sandbox (General)

SANDBOX NODES REPLICA SET CONNECT CONFIGURATION ...

REGION Frankfurt (eu-central-1)

clusterfra-shard-00-00.30... SECONDARY

clusterfra-shard-00-01.30... PRIMARY

clusterfra-shard-00-02.30... 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](#)

Operations R: 0.01 W: 0 723.1/s  
Last 6 Hours

Logical Size 329.4 MB 512.0 MB max  
0.0 B  
Last 6 Hours

Connections 7 500 max  
0  
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

The screenshot shows the MongoDB Atlas interface for the ClusterFra database. The left sidebar has sections for DEPLOYMENT, Databases (selected), Triggers, Data Lake, SECURITY, Database Access, Network Access, and Advanced. The top navigation bar includes Access Manager, Billing, All Clusters, Get Help, and Thomas. The main area shows the ClusterFra database with 8 databases and 21 collections. The restaurants collection is selected, displaying its details: Collection Size: 10.13MB, Total Documents: 25359, Indexes Total Size: 416KB. It includes tabs for Find, Indexes, Schema Anti-Patterns (0), Aggregation, and Search Indexes (1). A FILTER button is present. Below the collection details, the QUERY RESULTS 1-20 OF MANY section shows two documents:

```
_id: ObjectId("5eb3d668b31de5d588f4292a")
> address: Object
  borough: "Brooklyn"
  cuisine: "American"
> grades: Array
  name: "Riviera Caterer"
  restaurant_id: "40356018"

_id: ObjectId("5eb3d668b31de5d588f4292b")
> address: Object
  borough: "Brooklyn"
  cuisine: "Delicatessen"
> grades: Array
  name: "Wilken'S Fine Food"
  restaurant_id: "40356483"
```

# Comparing with Azure Cosmos DB and MongoDB

- Using MongoDB Compass to access your data

The screenshot shows the MongoDB Compass interface. The left sidebar lists databases (Praha21, 12 DBS, 31 Collections, FAVORITE) and collections (JsonDemo.JsonOrders, JsonPerson). The main window displays the JsonDemo.JsonOrders collection, which contains 31.5k documents. The total size is 23.3MB with an average size of 742B. The collection has indexes. A filter is applied: {OrderID: 2}. The results show two documents. The first document has an OrderID of 2, OrderDate of "2011-05-31T00:00:00", and a Lines array containing two items. The second document has an OrderID of 2, OrderDate of "2011-05-31T00:00:00", and a Lines array containing one item.

```
_id: 2
OrderID: 2
OrderDate: "2011-05-31T00:00:00"
OrderNo: "SO2"
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 mobile 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/<br>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.<br>languages            | ~ 12                   | ~ 6 + Mongo-driven                 | ~ 30                  |
| DB-ranking (*) overall/<br>document stores | 3 / -                  | 27 / 4                             | 5 / 1                 |
| Partitioning/replic<br>ation               | yes / yes              | yes / yes                          | yes / yes             |
| Consistency                                | immediate              | immediate, eventual                | immediate, eventual   |
| Scalability                                | on-prem vs Azure       | „instant, automatic“               | vertical & horizontal |

# 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 yet
- 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 & keep in touch:

-  @DerFredo <https://twitter.com/DerFredo>
-  [de.linkedin.com/in/derfredo](https://de.linkedin.com/in/derfredo)
-  <https://techhub.social/@DerFredo>



This file and the demo scripts can be found at:

<https://bit.ly/DerFredoRhein23>

