Battle of Time Series: SQL Server 2022 vs. ADX vs. InfluxDB vs. TimescaleDB

Sponsors


CONSORZIO
UNIVERSITARIO
DI PORDENONE
MOLTIPLICATORE DI VALORE


altitudo


beanTech
IT moves your business


[stesi]
Powered by Innovation


DATA
SATURDAYS


community

Gianluca Sartori

Founder of

QUANTUMDATIS

Independent SQL Server consultant

Data platform MVP

Works with SQL Server since 7.0

DBA @ Scuderia Ferrari

@spaghettidba

spaghettidba.com

quantumdatis.com

DATA SATURDAYS

AGENDA

1. What is Time Series data?
2. Options on the market
3. The contestants
4. The challenges
5. The winner

What is Time Series data?

Definitions

What is **time series data**?

> *"Time series data (or time-stamped data) is a collection of observations for a single object (entity) at different time intervals."*

What is a **time series database**?

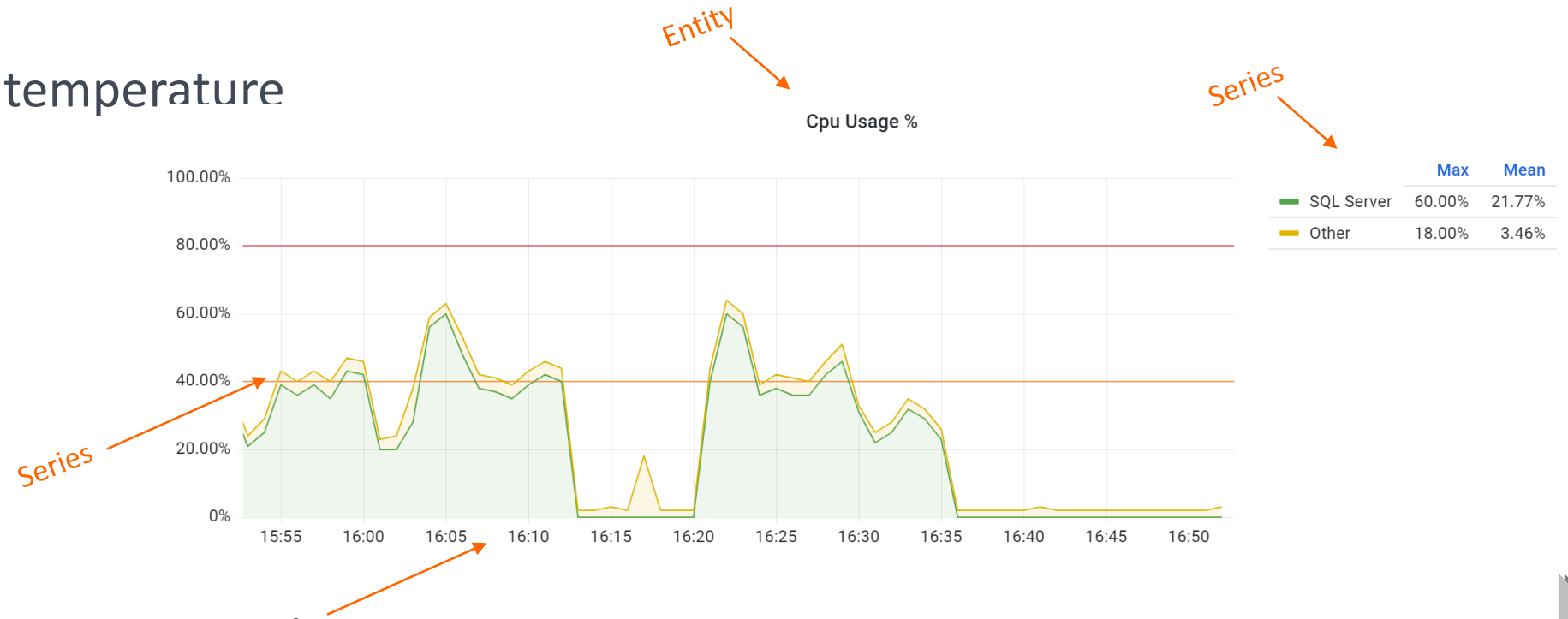
> *"A time series database (TSDB) is a database optimized for **time series data** and for measuring change over time."*

Time Series Data

Time series data is obtained by performing repeated measurements over time

Examples:

- Atmospheric temperature
- Stock prices
- CPU usage %
- Emails/sec
- Sensor data



Time Series Data

How is this different from relational data?

- Continuous Stream of data in time order
- Bulk uploads of large sets of data
- High volume data
- Data is append-only – no updates
- Delete large volumes of data when it goes out of scope
- Downsampling and aggregating high resolution data to save space

DATA SATURDAYS

Why does it matter?

Aircraft Engine

10 terabytes every 30 min.

14 million hours of flight in 2018

2 engines

= 560 million TB

2020s

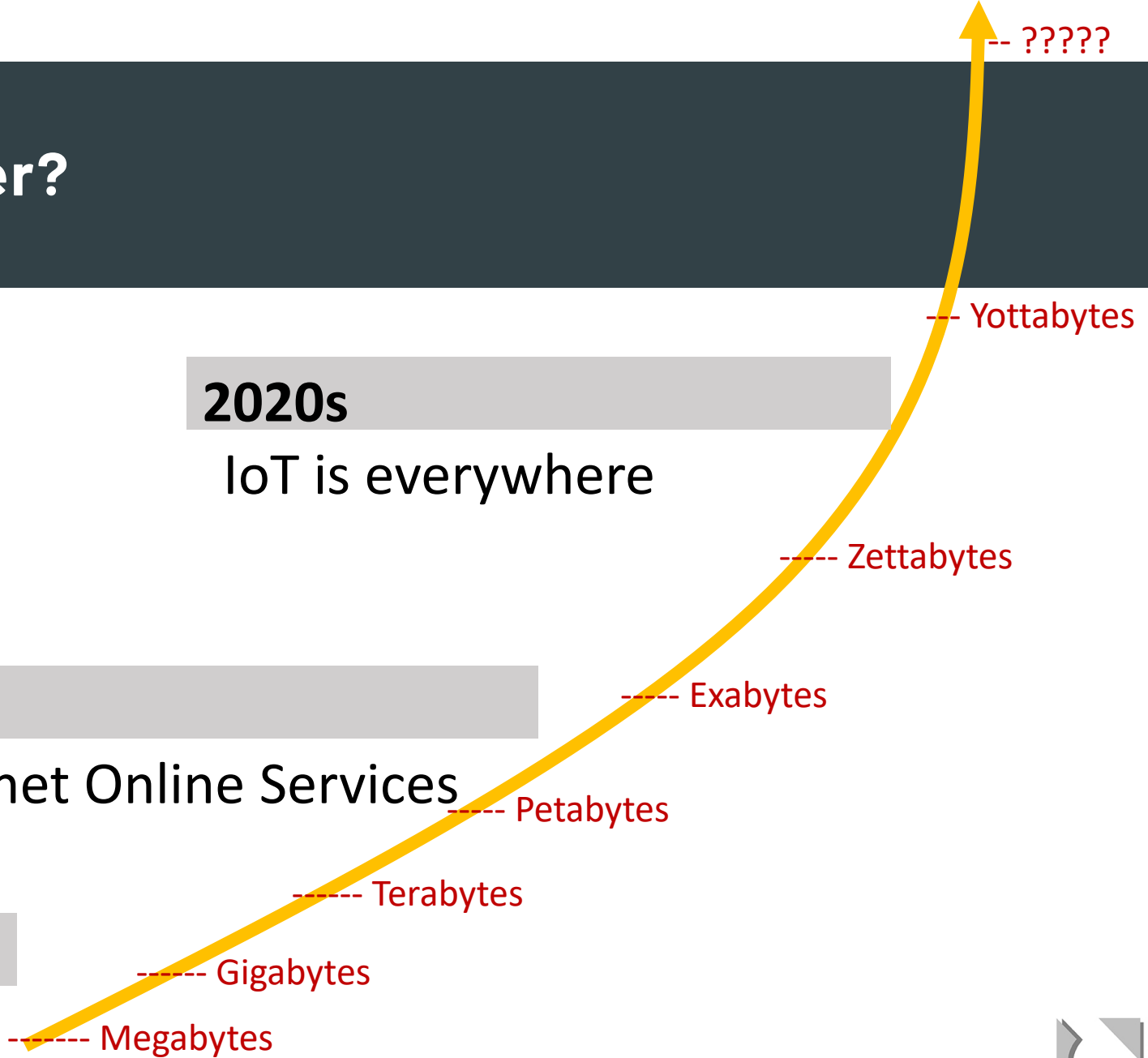
IoT is everywhere

2000s

Internet Online Services

1970s

Early Relational Databases



DATA SATURDAYS

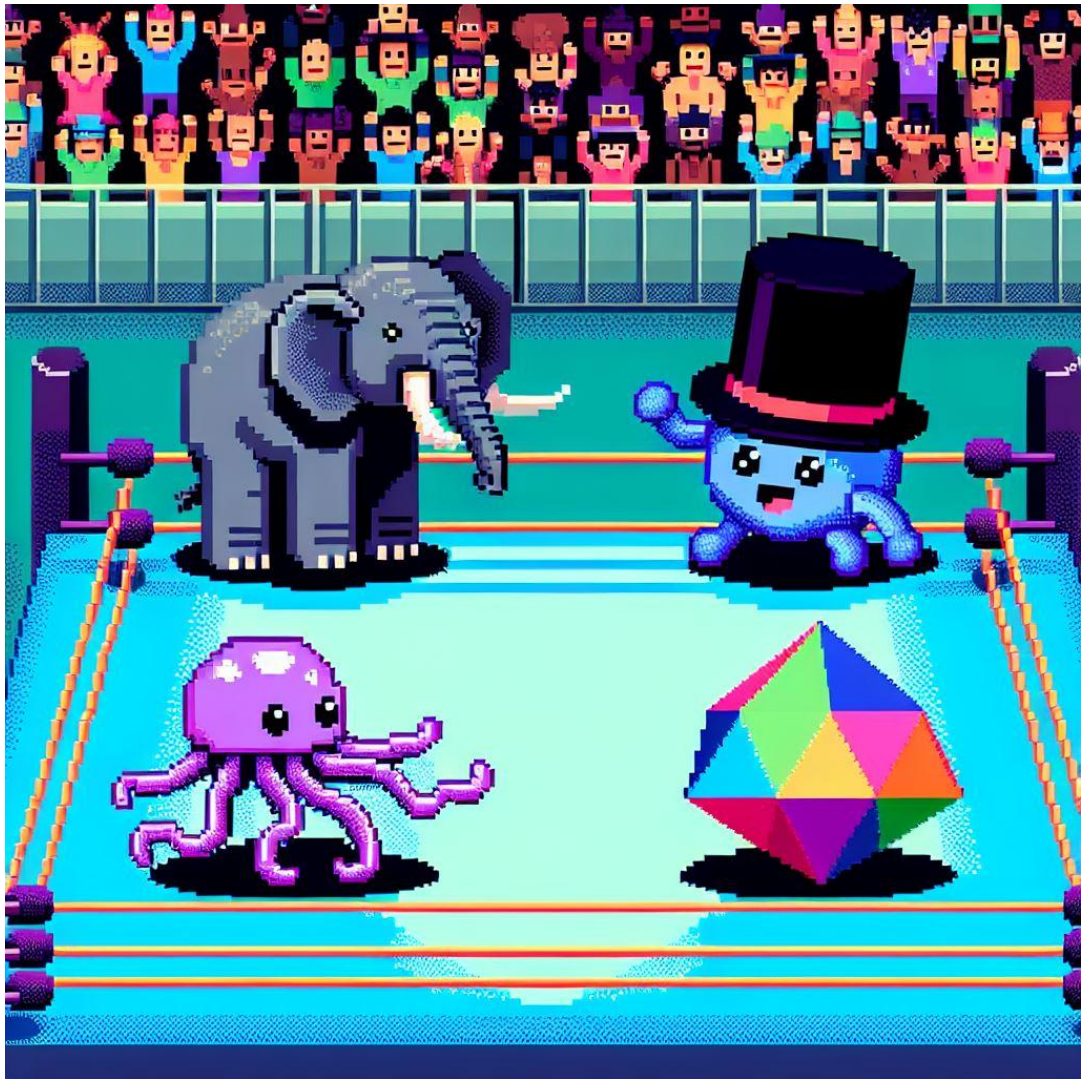
Time Series Database Options

RANK	DBMS	SCORE	
NOV 2024		NOV 2024	24 MONTH
1	InfluxDB	21.47	-8.55
2	KDB+	7.06	-1.72
3	Prometheus	6.92	+0.61
4	Graphite	4.91	-0.45
5	TimescaleDB	3.68	-0.88
6	QuestDB	2.91	+2.01
7	Apache Druid	2.70	-0.48
8	DolphinDB	2.60	+0.98
9	TDEngine	2.27	+1.08
?????	Azure Data Explorer	????	?????

Microsoft SQL Server 2022

Battle of TimeSeries!!!!

Timescale



Microsoft SQL Server

Azure Data Explorer

influxdb

DATA SATURDAYS

SQL Server 2022

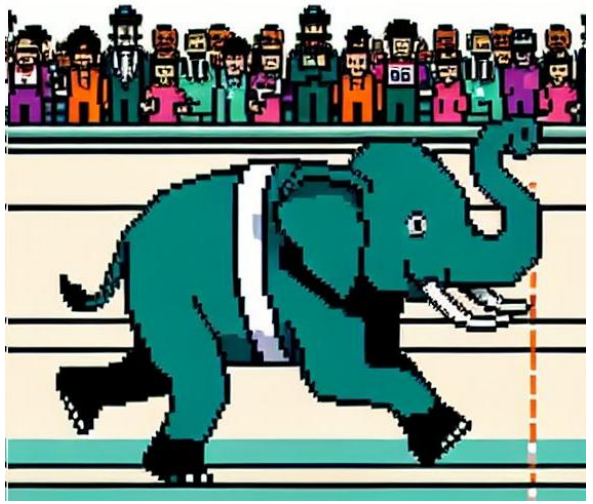
Microsoft SQL Server

- All-round, enterprise-ready relational database
- Not a time-series database
- Has some of the ingredients
- Has a specialized edition for time-series data called Azure SQL Edge

DATA SATURDAYS

Timescale

- Open Source Time-Series database
- Built on top of PostgreSQL as an extension
- Optimized to ingest, store and query time-series data
- Stores data in «HyperTables», special tables composed of «chunks»
- Time buckets partition data by timestamp
- Allows compression for
- Can also store relational data and all types of data supported by PostgreSQL



Azure Data Explorer

- Fully managed, high performance, big data analytics platform
- Can store and analyze structured, semi-structured and unstructured data
- Uses a «relational» model (tables, columns and rows) with fixed strongly-typed schemas
- Tables are stored in databases
- Clusters contain databases
- Log analytics, IoT, time series data

➔ not strictly a time-series database!

DATA SATURDAYS

influxdb

Most popular time series database

Designed and optimized for time series data

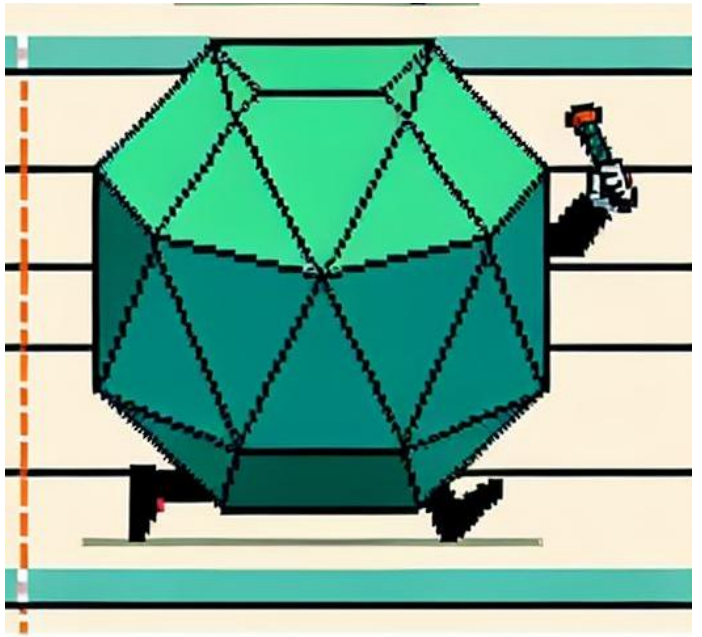
- Uses optimized TSM storage engine with columnar compression
- Built-in retention policies

Open Source with permissive licensing (MIT)

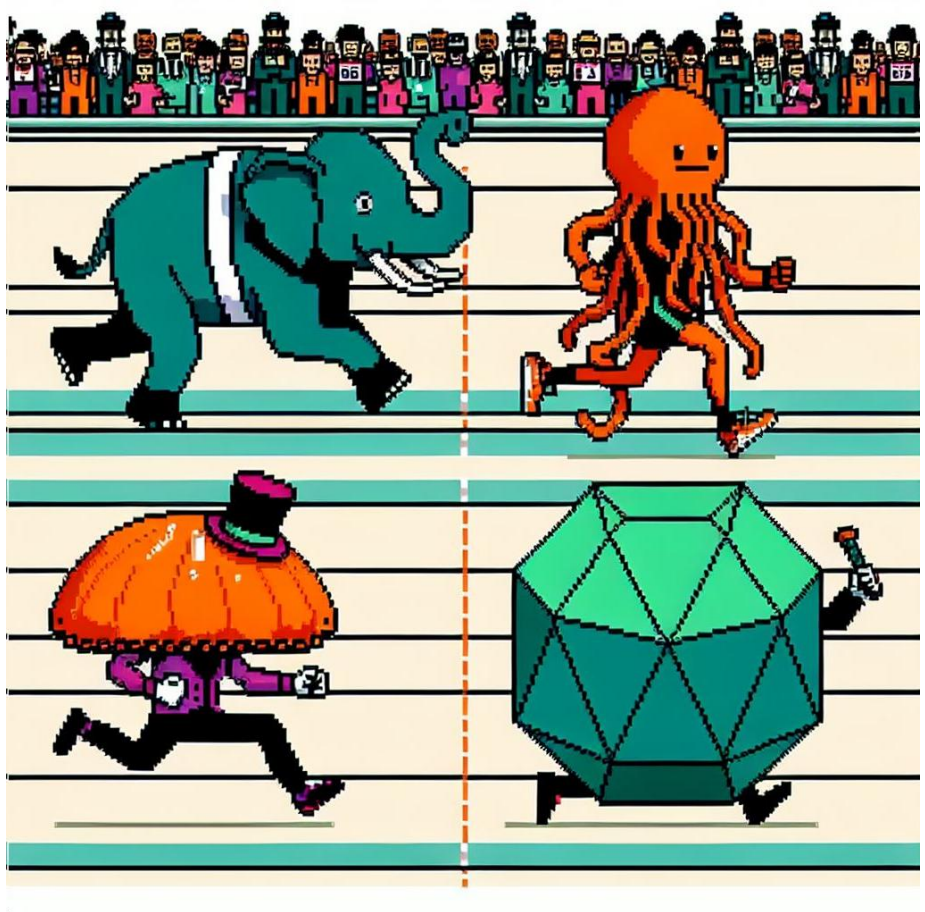
Written in GO

Cross platform:

windows, linux, macOS, docker, kubernetes, raspberryPI



The challenges



1. Licensing
2. Installation
3. Tooling
4. Storage Efficiency
5. Retention Policies and Downsampling
6. Query Capabilities

The rules:

- 0 to 3 points awarded
- I am the sole judge :)

DATA SATURDAYS

Licensing

DATA SATURDAYS

Licensing

SQL Server

- Commercial Product – public pricing
- Closed Source
- Has cloud option(s) – public pricing
- Has free editions with limitations

2

InfluxDB

- Commercial Product – secret pricing
- Open Source
- Has cloud option – public pricing
- Has extremely limited community edition
- Different versions with different limitations

0

TimescaleDB

- Commercial Product – secret pricing
- Open Source
- Has cloud option – public pricing
- Has free editions with **no** limitations

3

Azure Data Explorer

- Commercial Product – public pricing
- Closed Source
- Only available in the cloud
- Offers a limited free tier

1

Installation

DATA SATURDAYS

Installation and setup

SQL Server

- Cross-platform
- Works in containers
- GUI setup – Unattended setup
- Extensive and accurate documentation

3

InfluxDB

- Cross-platform
- Works in containers
- No setup required – single executable
- Nearly non-existent documentation

2

TimescaleDB

- Cross-platform
- Works in containers
- GUI setup – Unattended setup
- Extensive and accurate documentation *with OSS factor*

1

Azure Data Explorer

- No installation required
- Extensive and accurate documentation

1

Tooling

DATA SATURDAYS

Tooling

SQL Server

- SSMS
- Azure Data Studio
- Code extension
- Countless alternatives (some OSS)

3

InfluxDB

- Web UI - Chronograf
- CLI (influx.exe)
- Unofficial – unsupported OSS tools

0

TimescaleDB

- pg_admin
- Code extension
- Countless alternatives (some OSS)

2

Azure Data Explorer

- Web UI - dataexplorer.azure.com
- Kusto Explorer
- Azure Data Studio

1

DATA SATURDAYS

Storage Efficiency

SQL Server 2022 – Storage Engine

Big Heaps or Clustered Indexes are problematic

- Huge in size
- Hard to query
- Hard to maintain

Alternatives?

- Data compression
- Clustered Columnstore Indexes

DATA SATURDAYS

SQL Server 2022 – Columnstore Indexes

- Each column is stored independently
- Compresses data very efficiently
- Suitable when lots of repeating data exists
- Good for bulk operations
- Don’t play well with UPDATEs and DELETEs

DEMO!

InfluxDB - Series

SQL Server CPU usage percent from `sys.dm_os_ring_buffers`

time	SQL Server CPU	Other CPU
08:00	10	3
08:01	15	2
08:02	18	6
08:03	73	1

InfluxDB Series – An Example

This is saved in a **measurement** (let's call it **CPUHistory)** with **two series:**

time	SQL Server CPU	Other CPU
08:00	10	3
08:01	15	2
08:02	18	6
08:03	73	1

[CPUHistory, SQL Server CPU]

time	SQL Server CPU
08:00	10
08:01	15
08:02	18
08:03	73

[CPUHistory, Other CPU]

time	Other CPU
08:00	3
08:01	2
08:02	6
08:03	1

Series – Tags

If we add **tags**, we will get **one series for each tag value:**

time	**ServerName**	SQL Server CPU	Other CPU
08:00	ACCOUNTING	10	3
08:00	CRM	37	7
08:01	ACCOUNTING	15	2
08:01	CRM	48	5
08:02	ACCOUNTING	18	6
08:02	CRM	41	9
08:03	ACCOUNTING	73	1
08:03	CRM	28	7

[CPUHistory, **ACCOUNTING**, SQLServerCPU]

time	SQL Server CPU
08:00	10
08:01	15
08:02	18
08:03	73

[CPUHistory, **ACCOUNTING**, OtherCPU]

time	Other CPU
08:00	3
08:01	2
08:02	6
08:03	1

[CPUHistory, **CRM**, SQLServerCPU]

time	SQL Server CPU
08:00	37
08:01	48
08:02	41
08:03	28

[CPUHistory, **CRM**, OtherCPU]

time	Other CPU
08:00	7
08:01	5
08:02	9
08:03	7

InfluxDB – Storage Engine

Series are stored independently

We get one series for each combination of tag values → **cardinality**

v1 & v2 Storage engine is called **TSM** (Time series Structured Merge tree)

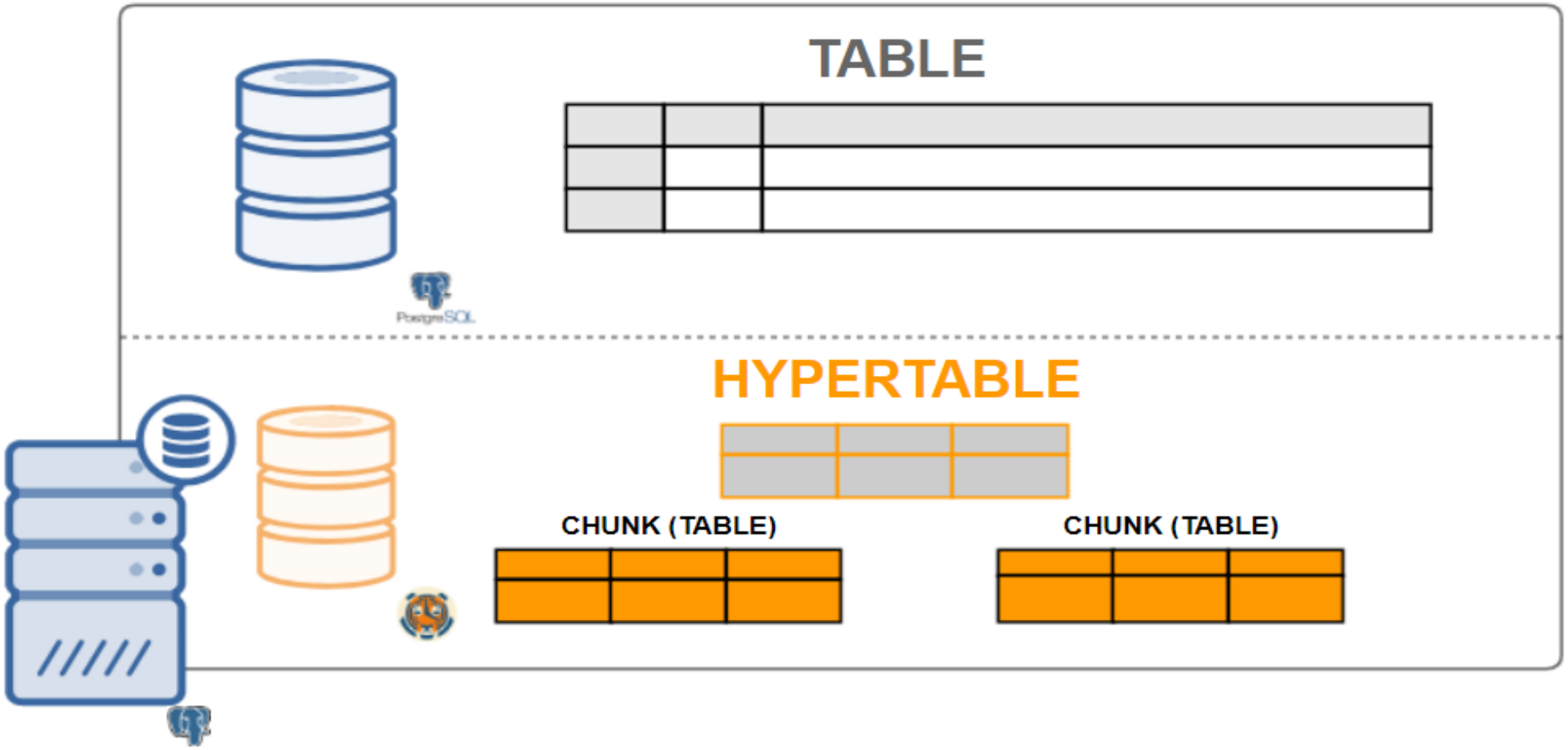
derived from **LSM** (Log-Structured Merge tree) → **Cassandra**

v3 Storage is based on **Parquet**

Available in the InfluxDB cloud

(Soon?) available on-prem

TimescaleDB – Storage Engine

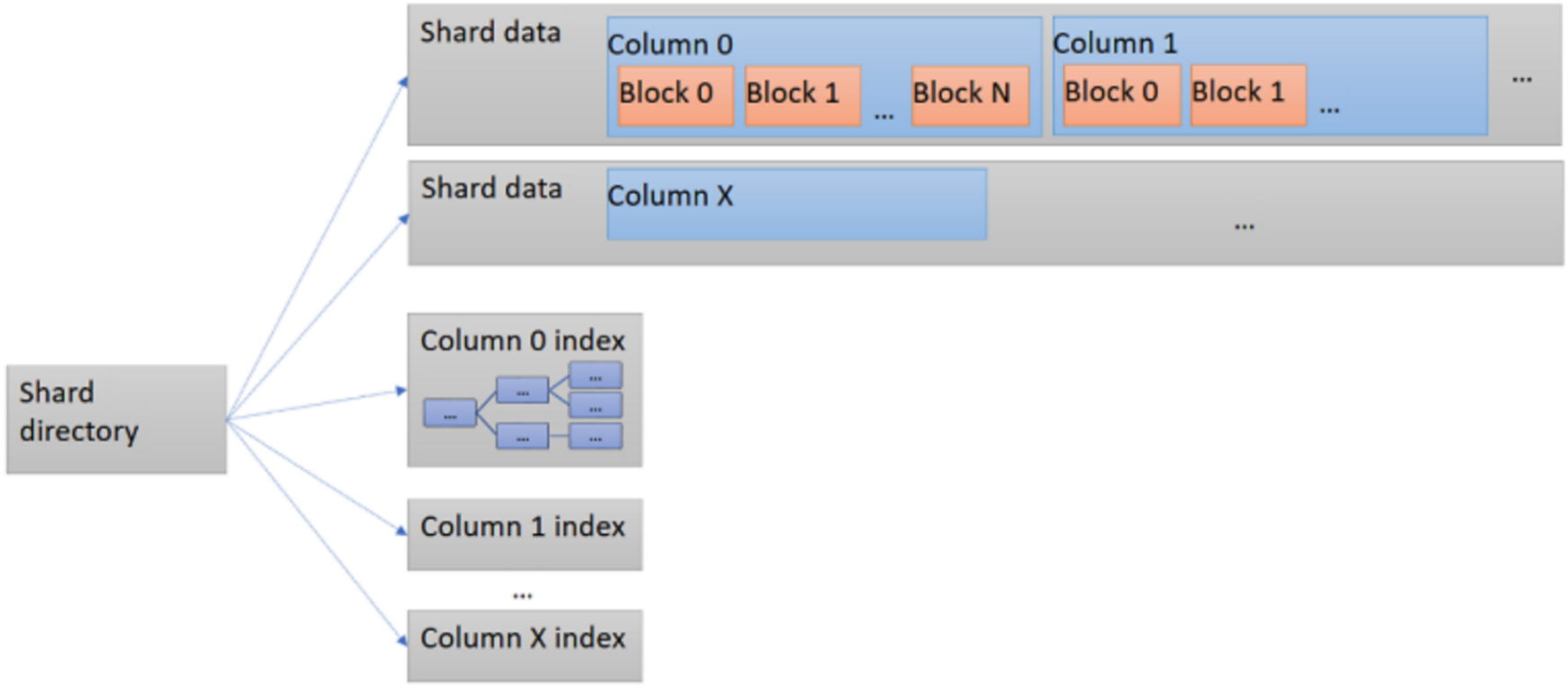


Azure Data Explorer - Internals

- Separates storage and compute resources
- Persistent data resides in Azure Blob Storage
  - Data is stored in **extents** (shards)
  - Extents are spread across cluster nodes
  - Extents are cached in SSD and memory
  - Data is compressed with columnar compression
- Compute uses a cache for persistent storage
- Row store is used when ingesting data (streaming ingestion)

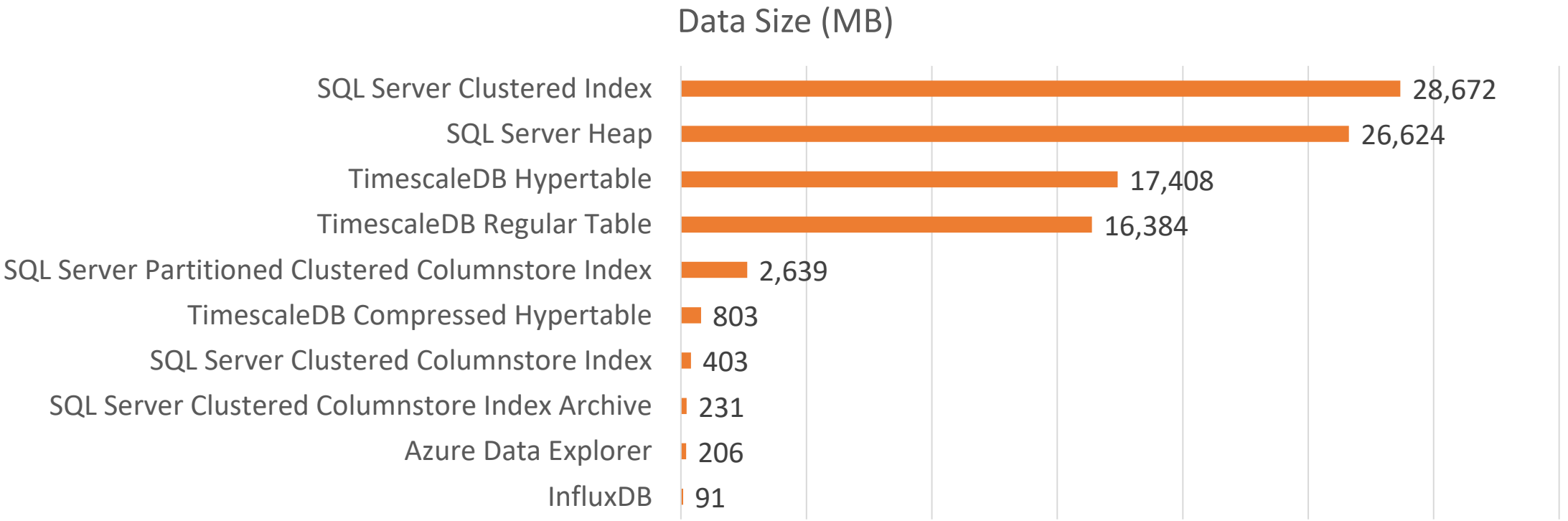
Azure Data Explorer – Shard Format

Compressed column store with free text support and full-text inverted index



Storage Efficiency

Data size for SQL Server performance counters, 2 instances,
15 days of data, 82 million rows



DATA SATURDAYS

Storage Efficiency

SQL Server

- Terrible with regular tables
- Better with clustered columnstore
- Partitioning affects efficiency

1

InfluxDB

- Stellar compression ratios
- Unaffected by sharding

3

TimescaleDB

- Terrible with regular tables
- Regular hypertables are just as inefficient
- Compressed hypertables are fine

2

Azure Data Explorer

- Uses an undocumented columnar compression algorithm
- Stores data efficiently

2

DATA SATURDAYS

Retention Policies and Downsampling

DATA SATURDAYS

Storage Efficiency

SQL Server

- Data needs to be deleted manually
- Partitioning makes it nearly-instantaneous
- Requires scheduling jobs

0

InfluxDB

- Retention policies are built-in
- Downsampling can be achieved with continuous queries in v1 and with jobs in v2 and v3

3

TimescaleDB

- Retention policies are built-in
- Downsampling is implemented with materialized views

3

Azure Data Explorer

- Retention policies are built-in
- Data gets soft-deleted when expired
- Downsampling is implemented with materialized views or with Power Automate
- Not really a time-series database

1

DATA SATURDAYS

Query Capabilities

InfluxDB – Query Capabilities

InfluxDB supports 2 query languages:

InfluxQL → similar to SQL, born with version 1, still available in v2

```
SELECT MEAN(SQLServerCPU) AS avg_cpu
FROM CPUHistory
WHERE time > now() -1d
GROUP BY server_name
```

InfluxDB – Query Capabilities

Flux → doesn't look like SQL (more like KQL), default in v2, also available in v1

Supports «advanced» query features like joins…

```
from(bucket:"demo/autogen" )
    |> range(start: -1d)
    |> filter(fn: (r) => r._measurement == "CPUHistory")
    |> filter(fn: (r) => r._field == "SQLServerCPU")
    |> mean()
    |> group(columns: ["server_name"])
    |> yield(name: "avg_cpu")
```

InfluxDB – Query Capabilities

InfluxDB v3 supports SQL

InfluxDB Cloud Serverless uses the **Apache Arrow DataFusion** implementation of SQL

Supports JOINs
Supports CTEs
Supports windowing functions
… Supports everything you would expect from a decent database

GROUP BY TIME

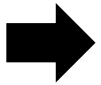
InfluxQL allows grouping by time:

```
SELECT MEAN(value) AS avg_lazy_writes_sec
FROM "sqlserver_performance"
WHERE counter = 'Lazy writes/sec'
    AND sql_instance = 'SQLCSRV04:SQL2017'
    AND time > now() - 1d
GROUP BY time(1h)
```

GROUP BY TIME

… GROUP BY time(1h)

time	lazy_writes_sec	
16-10-2022 10:00:00	398091	
…	…	10:00
16-10-2022 10:59:59	395644	
16-10-2022 11:00:00	432211	
…	…	11:00
16-10-2022 11:59:59	452325	
16-10-2022 12:00:00	456541	
…	…	12:00
16-10-2022 12:59:59	433505	
16-10-2022 13:00:00	456654	
…	…	13:00
16-10-2022 13:59:59	456788	
16-10-2022 14:00:00	417545	
…	…	…



time	avg_lazy_writes_sec
16-10-2022 10:00:00	398,091.00
16-10-2022 11:00:00	400,187.50
16-10-2022 12:00:00	406,095.00
16-10-2022 13:00:00	411,167.76
16-10-2022 14:00:00	417,963.40
16-10-2022 15:00:00	425,683.27
16-10-2022 16:00:00	433,001.40
16-10-2022 17:00:00	441,937.02
16-10-2022 18:00:00	449,937.73
16-10-2022 19:00:00	450,737.81
16-10-2022 20:00:00	453,467.40
16-10-2022 21:00:00	455,703.76
16-10-2022 22:00:00	457,270.19
16-10-2022 23:00:00	458,743.47
17-10-2022 00:00:00	461,533.64
…	…

FILL

InfluxQL allows filling intervals with data:

```
SELECT MEAN(value) AS avg_lazy_writes_sec
FROM "sqlserver_performance"
WHERE counter = 'Lazy writes/sec'
    AND sql_instance = 'SQLCSRV04:SQL2017'
    AND time > now() - 1d
GROUP BY time(1h) FILL(NULL)
```

FILL

… FILL(NULL|VALUE)

time	value
16-10-2022 10:35:45	398091
16-10-2022 10:36:01	398091
16-10-2022 12:16:15	406095
16-10-2022 12:16:30	406095
16-10-2022 13:06:45	411167
16-10-2022 13:07:01	411167

time	avg_lazy_writes_sec
16-10-2022 10:00:00	398,091.00
16-10-2022 11:00:00	NULL
16-10-2022 12:00:00	406,095.00
16-10-2022 13:00:00	411,167.76

SQL Server – Query capabilities

The new DATE_BUCKET function in SQL Server 2022 helps implement GROUP BY TIME

```
SELECT DATE_BUCKET(hour, 1, time) AS time, AVG(value) AS avg_lazy_writes_sec
FROM [dbo].[sqlserver_performance_cci_partitioned]
WHERE time >= '2022-10-01'
    AND time < '2022-10-02'
    AND counter = 'Lazy writes/sec'
    AND sql_instance = 'SQLCSRV04:SQL2017'
GROUP BY DATE_BUCKET(hour, 1, time)
ORDER BY 1
```

SQL Server – Query capabilities

The new GENERATE_SERIES function in SQL Server 2022 helps implement FILL

```
SELECT DATEADD(hour, value, '2022-10-01') AS time
FROM GENERATE_SERIES(0, DATEDIFF(hour, '2022-10-01', '2022-10-02') -1)
```

DEMO!

Azure Data Explorer – Query Capabilities

- ADX supports the Kusto Query Language (KQL)
- It's a brand new language, no similarities with SQL
- SQL is not supported but you can convert SQL to KQL with EXPLAIN
- KQL is simple and powerful
- … but it's not SQL!!!
- SQL support for KQL databases is coming in 2025!!!

```
Perf
| where TimeGenerated >= ago(10m)
| where CounterName == "% Free Space"
| project PerfComputer = Computer
        , CounterName
        , CounterValue
        , PerfTime=TimeGenerated
| join ( InsightsMetrics
        | where TimeGenerated >= ago(10m)
        | project IMComputer = Computer
                , Namespace
                , Name
                , Val
                , IMTime=TimeGenerated
      )
    on $left.PerfComputer == $right.IMComputer
```

DEMO!

TimescaleDB – Query capabilities

PostgreSQL has a rich SQL dialect called PL/pgSQL (or just pgSQL) which handles most queries with ease.

TimescaleDB adds **hyperfunctions** to handle common time-series use cases

GROUP BY TIME is achieved with **time_bucket()**

FILL is achieved with **time_bucket_gapfill()**

Carry over of the last value is implemented with **locf()**

DEMO!

DATA SATURDAYS

Query capabilities

SQL Server

- Supports T-SQL
- Has gained (some) time-series capabilities with SQL Server 2022

1

InfluxDB

- Has specific time-series query features
- InfluxQL
- Flux
- SQL

3

TimescaleDB

- Supports pgSQL
- Has specific time-series query features

3

Azure Data Explorer

- Has specific time-series query features
- Supports KQL
- Has a SQL endpoint

2

DATA SATURDAYS

And the winner is...

	10		11		14		8

TimescaleDB!

DATA SATURDAYS

Resources

Questions? Ask me!

spaghettidba@sqlconsulting.it

Get slides and demos from GitHub:

https://github.com/spaghettidba/CodeSamples