



Explore the Azure Cosmos DB with .NET Core 2.0



Jeremy Likness
Cloud Developer Advocate





Jeremy Likness

Cloud Dev Advocate @ Microsoft
20 Year Professional Dev

 <https://blog.jeremylikness.com/>

 @JeremyLikness

 Jeremy.Likness@microsoft.com



Aaron Wislang
@as_w
Linux



Abel Wang
@AbelSquidHead
DevOps



Alena Hall
@lenadroid
High Scale / Big Data



Anthony Chu
@nthonychu
.NET



Ashley McNamara
@ashleymcnamara
Linux



Asim Hussain
@jawache
JavaScript / Node.js / Python



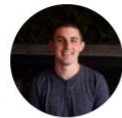
Bernd Verst
@BerndVerst
Linux / Containers / Python



Brandon Minnick
@BrandonXamarin
Xamarin / C# / iOS / Android



Brian Benz
@bbenz
Java



Brian Clark
@_clarkio
JavaScript / Node.js / Python



Brian Ketelsen
@bketelsen
Linux



Brian Peek
@BrianPeek
Emerging / Gaming



Bridget Kromhout
@bridgetkromhout



Bryan Liston
@listonb
Linux



Burke Holland
@burkeholland
JavaScript / Node.js / Python



Cecil Phillip
@cecilphillip
.NET / Xamarin



Damian Brady
@damovisa
DevOps



David Smith
@revodavid
Data Science / R



Donovan Brown
@DonovanBrown
DevOps



Erik St. Martin
@erikstmartin



Jasmine Greenaway
@paladique
.NET / Xamarin



Jeremy Likness
@JeremyLikness
.NET



Jessica Frazelle
@jessfraz
Linux / Open Source



Jim Bennett
@jimbobbennett
Xamarin / .NET



John Papa
@john_papa
JavaScript / Node.js / Python



Jonathan Giles
@JonathanGiles
Java



Laurent Bugnion
@LBugnion
.NET / Xamarin



Matthew Soucoup
@codemillmatt
.NET / Xamarin



Maxime Rouiller
@maximrouiller
.NET



Paige Bailey
@DynamicWebPaige
Data / AI / ML



Prashant Sridharan
@CoolAssPuppy
Chief Herder



Ruth Yakubu
@ruthiyakubu
Data / AI



Sarah Drasner
@sarah_edo
JavaScript / Node.js / Python



Scott Cate
@ScottCate
.NET



Seth Juarez
@sethjuarez
Emerging / ML / AI / Channel 9



Shayne Boyer
@spboyer
.NET



Simona Cotin
@simona_cotin
JavaScript / Node.js / Python



Steven Murawski
@stevenmurawski
DevOps



Tim Heuer
@timheuer
.NET / Xamarin



Vadim Karpusenko
@vadi
HPC / AI / ML / DL



Zachary Deptawa
@zdeptawa
Linux

 <https://aka.ms/advocates>

 [@azureAdvocates](https://twitter.com/azureAdvocates)

 <https://aka.ms/cosmos-docs>

Today's journey ...



- The SQL legacy
- A different way: NoSQL
- Introducing Azure Cosmos DB
- Demos
- Conclusion

“We are a way for the cosmos
to know itself.”

-Carl Sagan

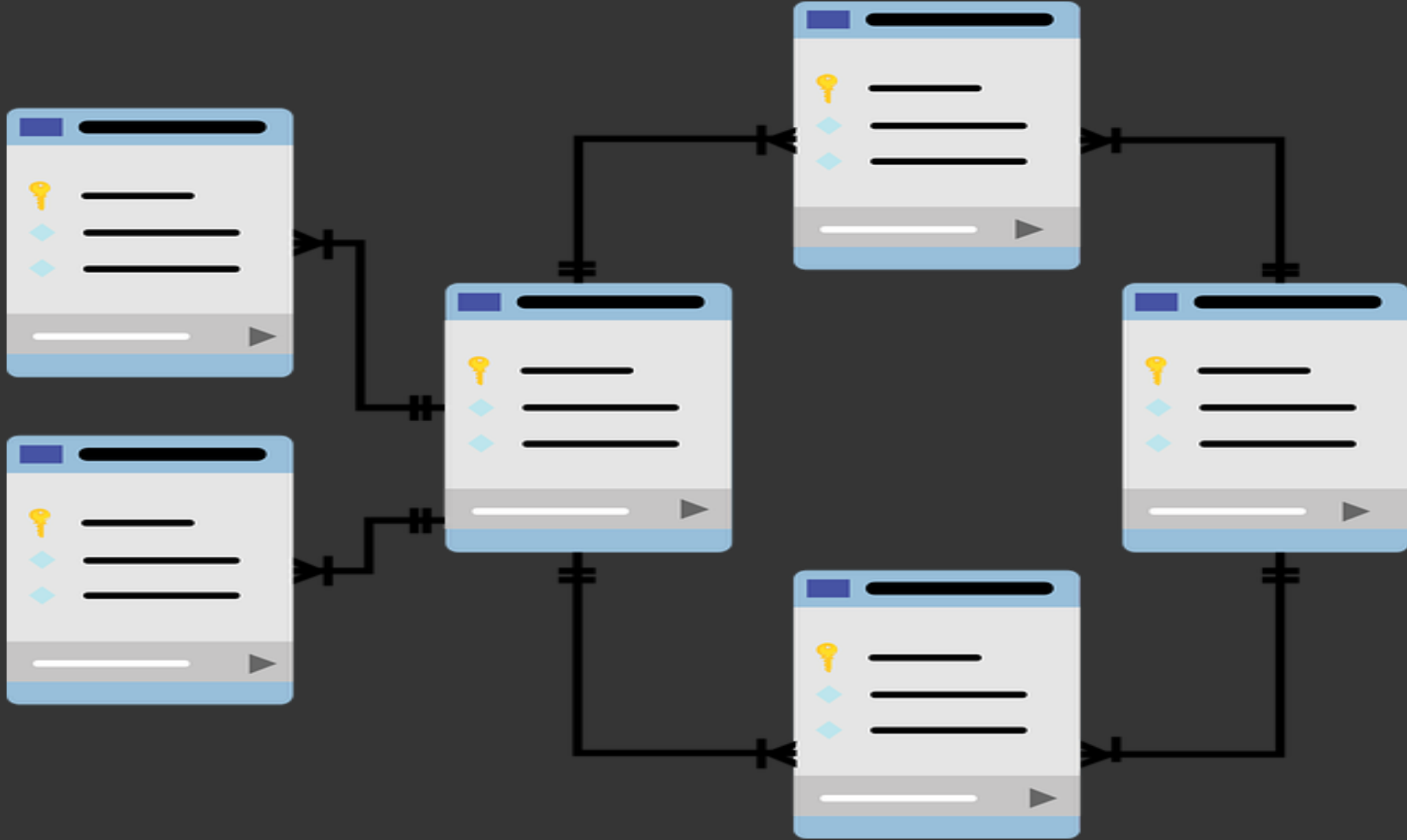
“But this is how we’ve always done it.”

-Developer

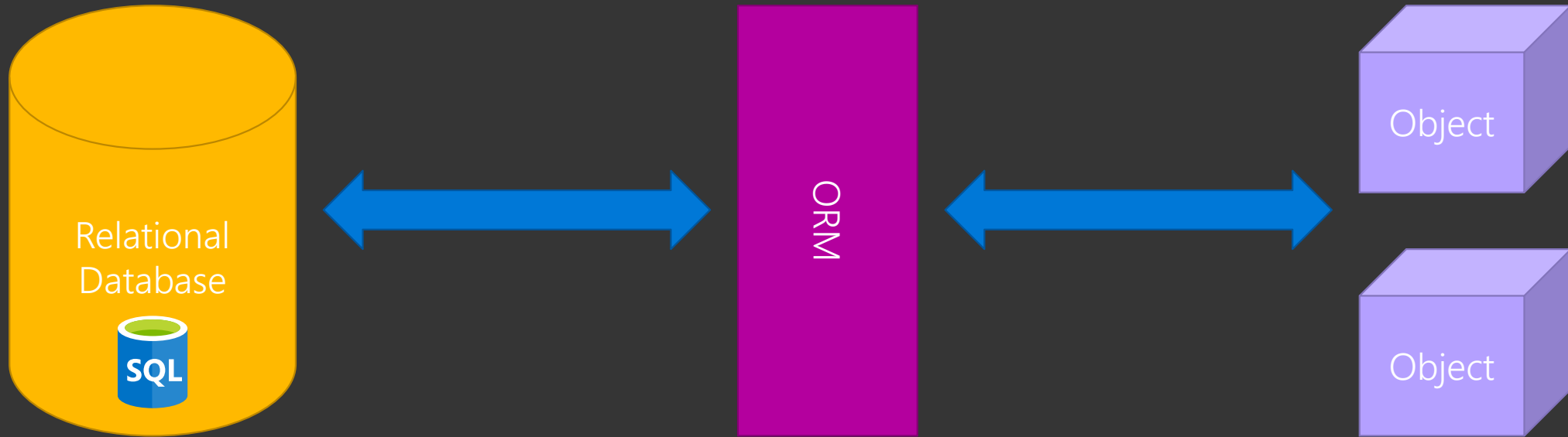




```
***** COMMODORE 64 BASIC V2 *****  
64K RAM SYSTEM 38911 BASIC BYTES FREE  
READY.  
10 OPEN 15,8,15  
20 OPEN 1,8,2,"COSMOSDB,L,"+CHR$(50)  
30 PRINT #15,"P"+CHR$(98)+CHR$(10)+CHR$(  
0)+CHR$(1)  
40 PRINT #1,"HOW WE ROLLED IN THE 80S"  
50 CLOSE 1
```



Disconnect: Relational vs. Domain



Introducing Not Only SQL: NoSQL

Key/Value

Column

Document

Graph

Key/Value



Key	Value		
Key1	\$3.05		
Key2		9/22/1974	
Key3	\$4.99		Profile1.jpg

Column



```
[
  { "id": 1, "name": "Jeremy Likness", "twitter": "@JeremyLikness" },
  { "id": 2, "name": "Scott Cate", "twitter": "@ScottCate", "isManager": true }
]

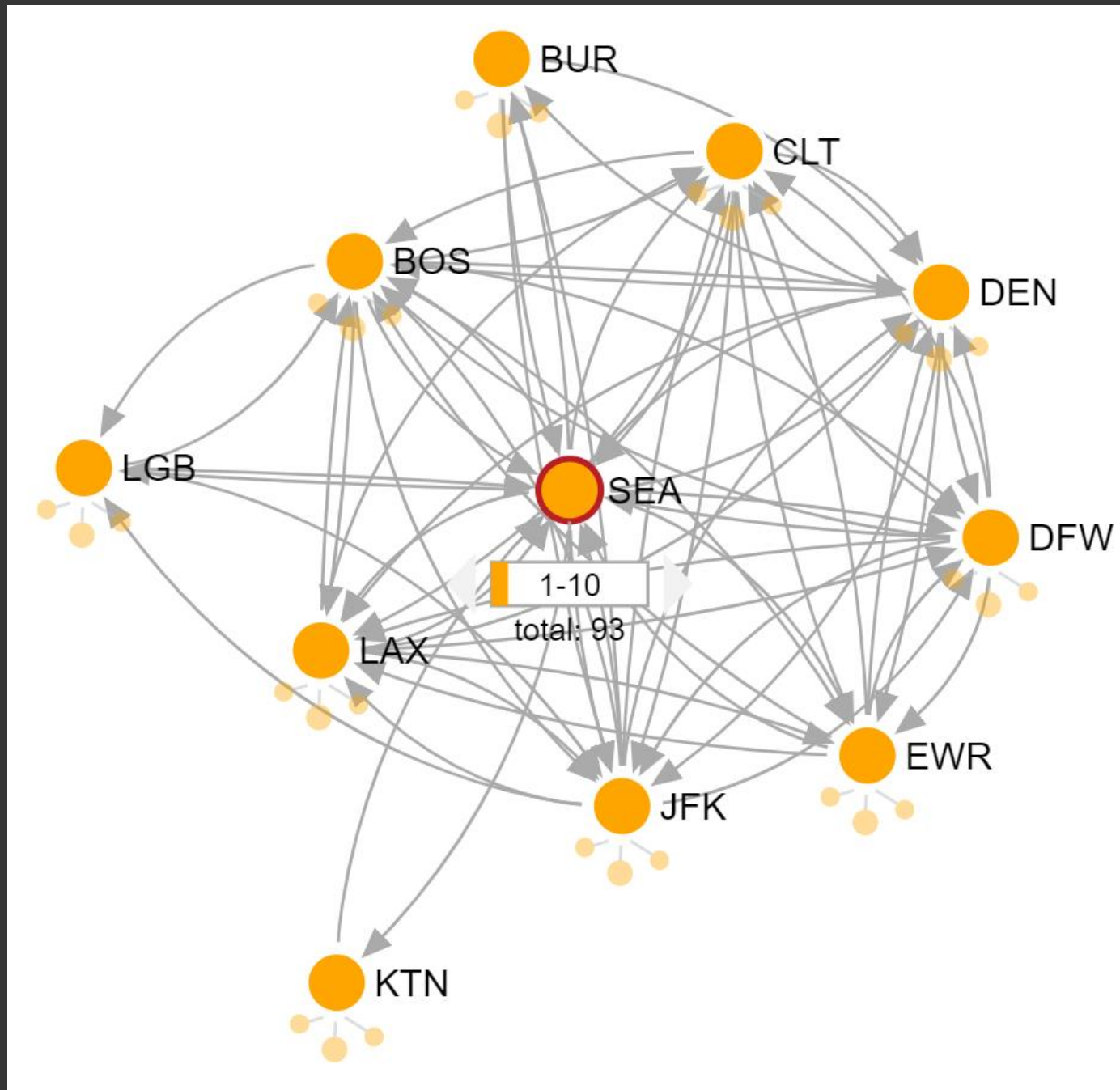
{
  "isManager" : { "2": true },
  "name": { "1": "Jeremy Likness", "2": "Scott Cate" },
  "twitter": { "1": "@JeremyLikness", "2": "Scott Cate" }
}
```

Document



```
{
  "_id": "204",
  "UnitOfMeasure": "g",
  "TagName": "FAT",
  "Description": "Total lipid (fat)",
  "SortOrder": 800
}
```

Graph



Advantages



Independent

Flexible
Schema

Indexing
Support



Easy
Manipulation

JSON Ready

Fast



Large
Datasets

Version
Proof

ORM Free



Azure Cosmos DB Handles all of your NoSQL Needs

- Column: Cassandra API
- Document: SQL API, MongoDB API
- Graph: Gremlin API
- Key/Value: Table API





Demo: Create Azure Cosmos DB





Introducing Serverless NoSQL with Azure Cosmos DB.



Turnkey Global Distribution

Learn more:

➡ <https://jlik.me/ckw>

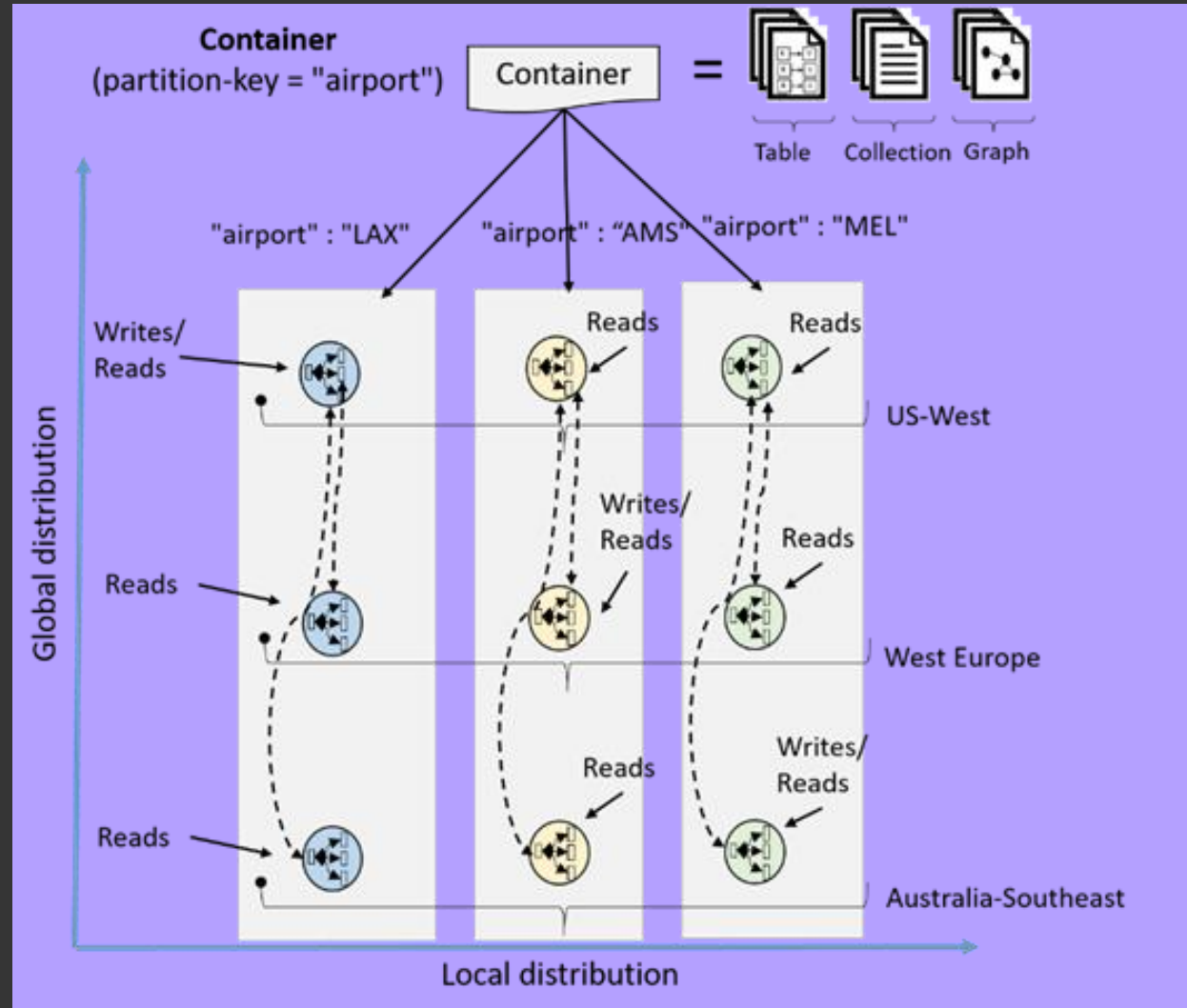




Learn more:

➡ <https://jlik.me/ckw>

Turnkey Global Distribution





Turnkey Global Distribution

Elastic scale out of storage and throughput

No single-
machine
bottleneck

Server-side
partition
management

GB to PB Storage,
selectable
throughput

Automatic
document
expiration

Pay by the hour,
customize
throughput cost

Support for
requests per
second or minute



Turnkey Global Distribution

Elastic scale out of storage and throughput

Guaranteed 99th percentile low latency

	Strong	Bounded Staleness	Session	Prefix	Eventual
Indexed Writes (1 KB)	<10ms + 2RTT	<10ms	<10ms	<10ms	<10ms
Reads (1 KB)	<10ms	<10ms	<10ms	<10ms	<10ms



Turnkey Global Distribution

Elastic scale out of storage and throughput

Guaranteed 99th percentile low latency

Five well-defined consistency models

Learn more:

➡ <https://jlik.me/ckx>



Strong



Bounded-stateless



Session

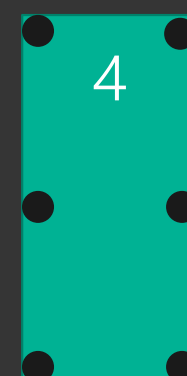
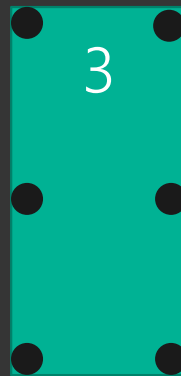
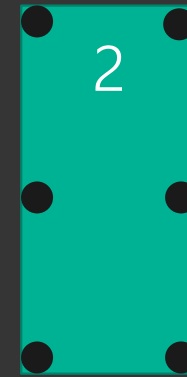
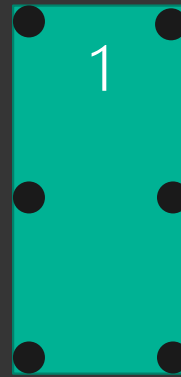


Consistent prefix



Eventual

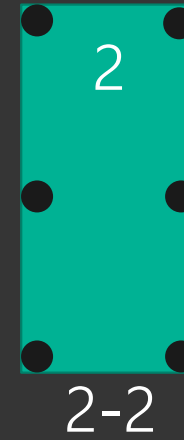
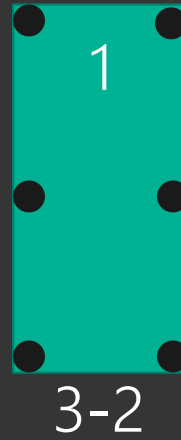




A	-	B
0	-	0
1	-	0
1	-	1
2	-	1
2	-	2
3	-	2

Learn more:

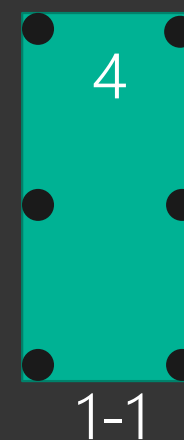
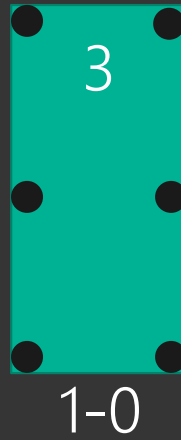
➡ <https://jlik.me/ckx>



Write 1:A +1
Write 1:B +1
Write 2:A +1
Write 1:A +1
...

Learn more:

➡ <https://jlik.me/ckx>





Turnkey Global Distribution

Elastic scale out of storage and throughput

Guaranteed 99th percentile low latency

Five well-defined consistency models

- Guaranteed latest version of reads
- Highest consistency
- Lowest performance
- Lowest availability
- 2x cost of Session -> Eventual



Strong



Bounded-staleness



Session



Consistent prefix

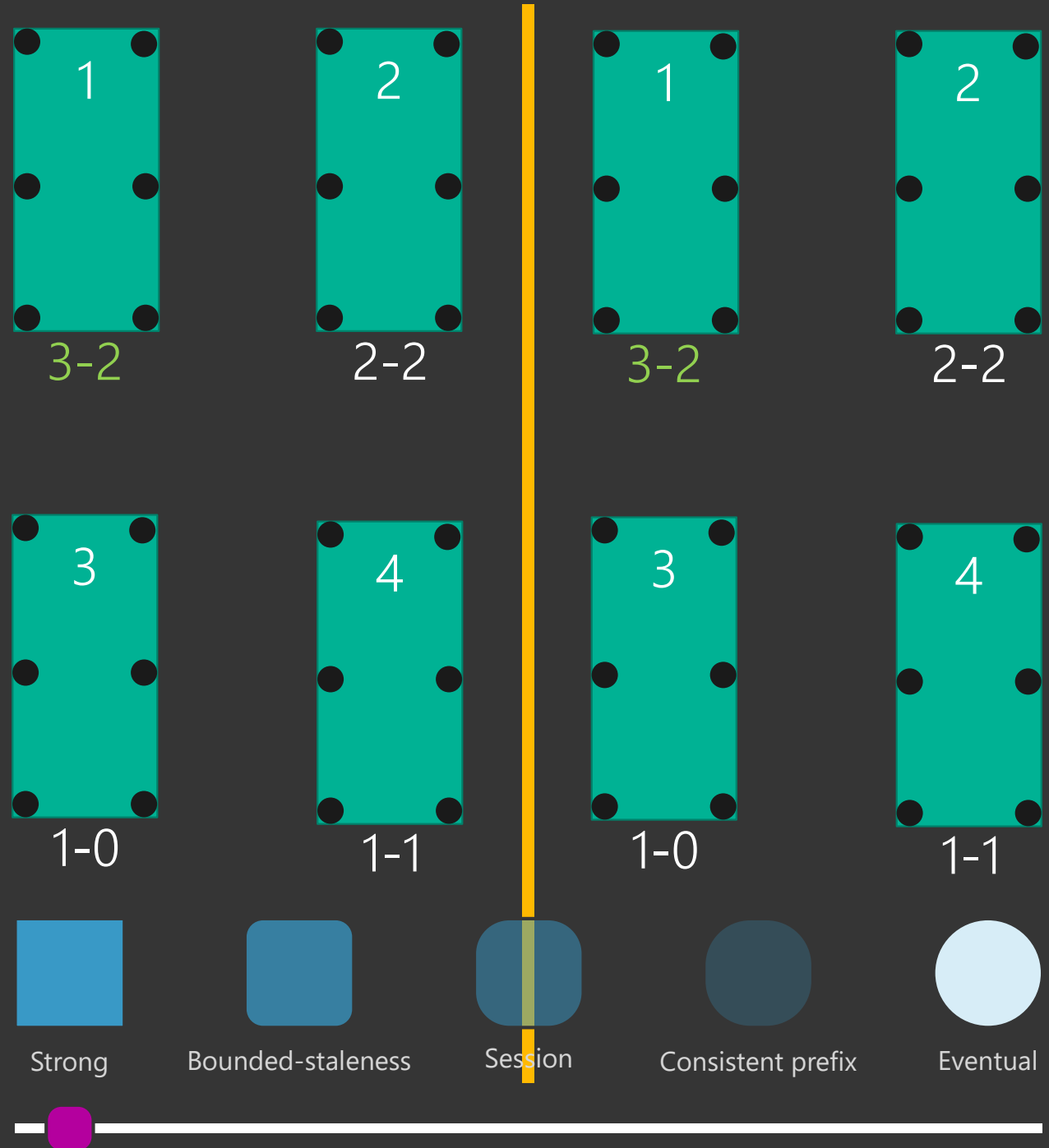


Eventual





- Guaranteed latest version of reads
- Highest consistency
- Lowest performance
- Lowest availability
- 2x cost of Session -> Eventual





Turnkey Global Distribution

Elastic scale out of storage and throughput

Guaranteed 99th percentile low latency

Five well-defined consistency models

- Consistent within a time frame (i.e. 5 minutes old)
- Better performance
- **Low availability**
- **2x cost of Session -> Eventual**



Strong



Bounded-staleness



Session



Consistent prefix

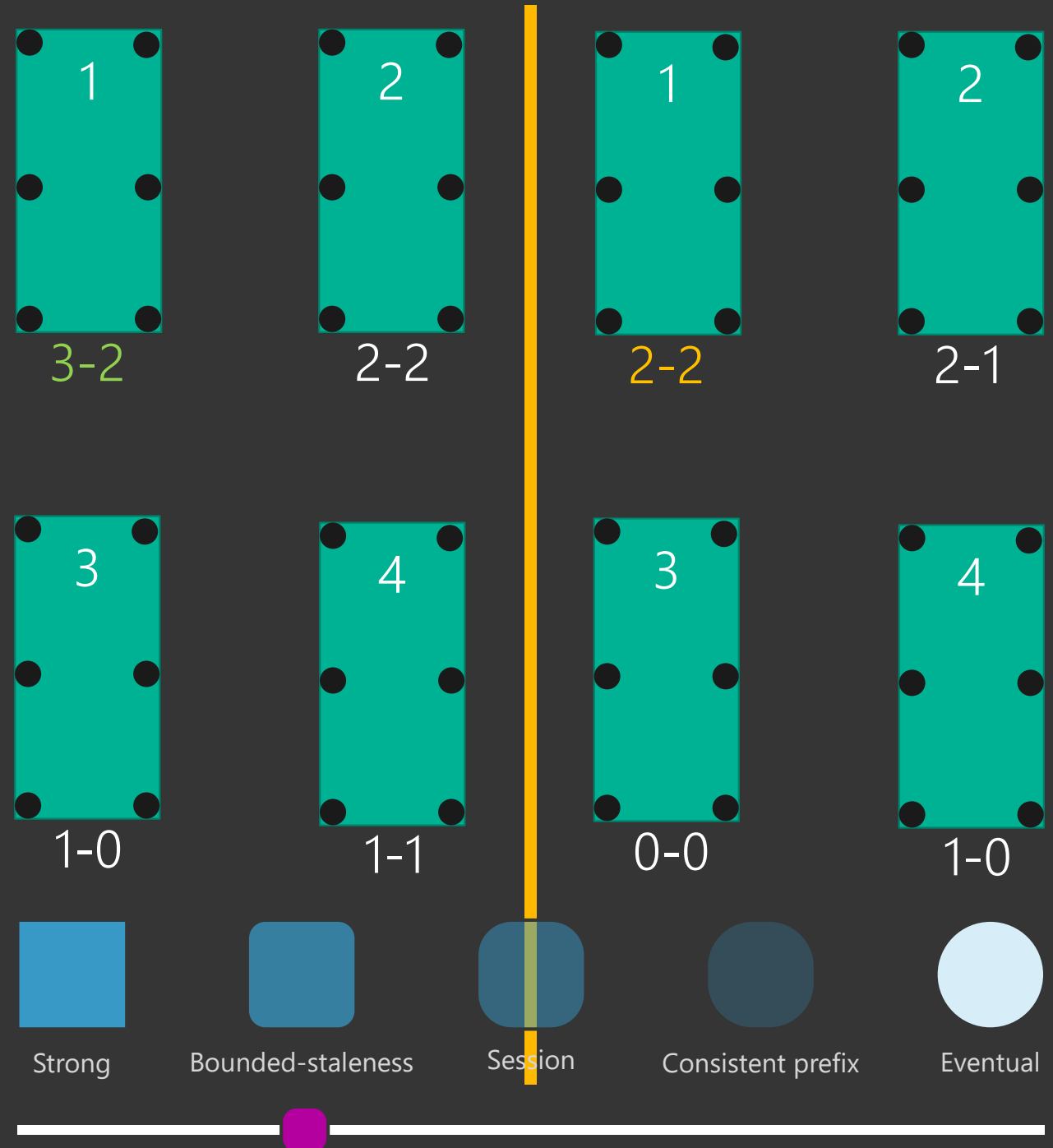


Eventual





- Consistent within a time frame (i.e. 5 minutes old)
- Better performance
- Low availability
- 2x cost of Session -> Eventual





Turnkey Global Distribution

Elastic scale out of storage and throughput

Guaranteed 99th percentile low latency

Five well-defined consistency models

- Consistent for the session
- Always read own writes
- Other data ordered but not guaranteed current
- Good performance
- Good availability



Strong



Bounded-staleness



Session



Consistent prefix

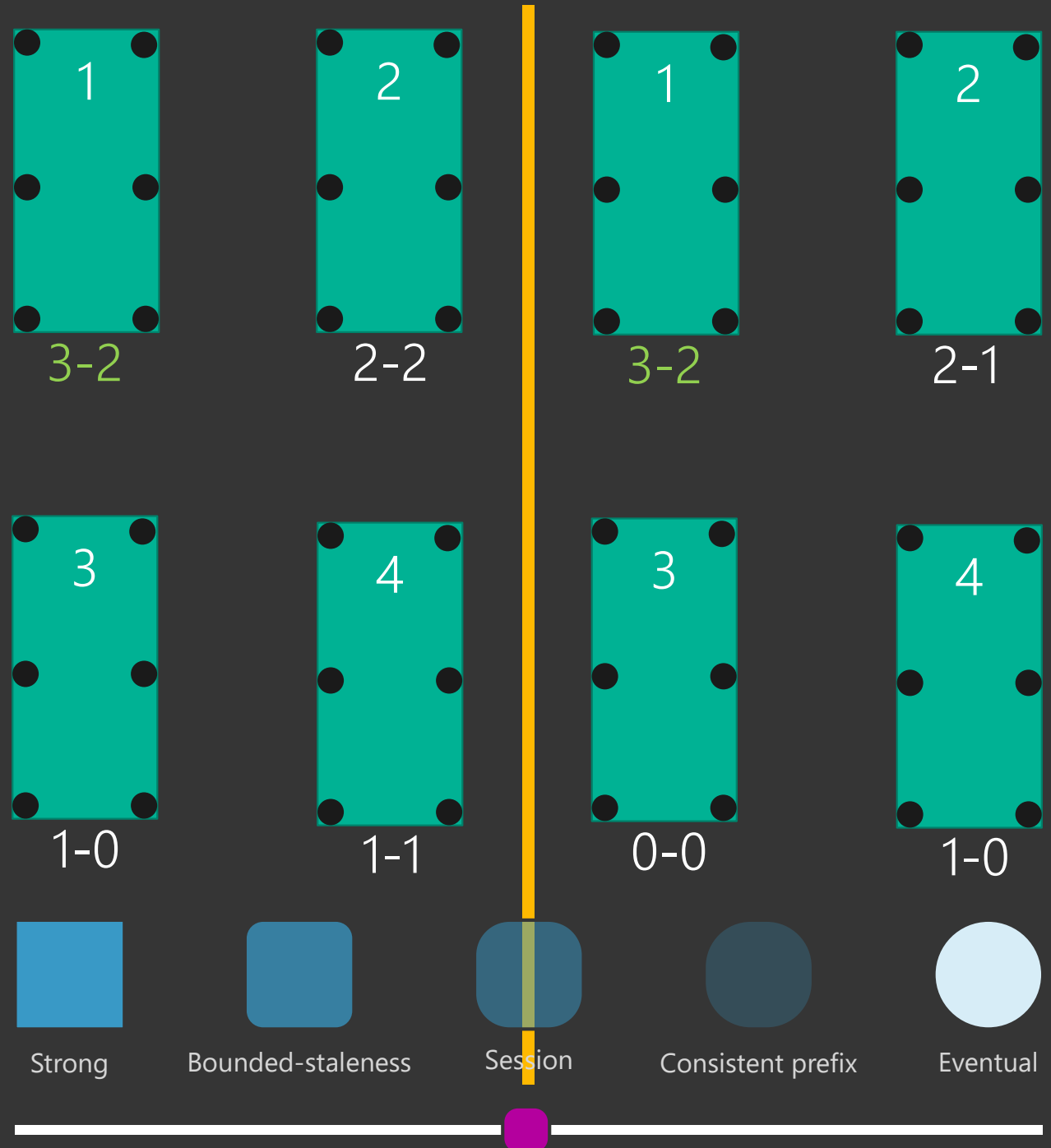


Eventual





- Consistent for the session
- Always read own writes
- Other data ordered but not guaranteed current
- Good performance
- Good availability





Turnkey Global Distribution

Elastic scale out of storage and throughput

Guaranteed 99th percentile low latency

Five well-defined consistency models

- Read is consistent to a point in time
- No guarantee how current
- Good performance
- Excellent availability



Strong



Bounded-staleness



Session



Consistent prefix

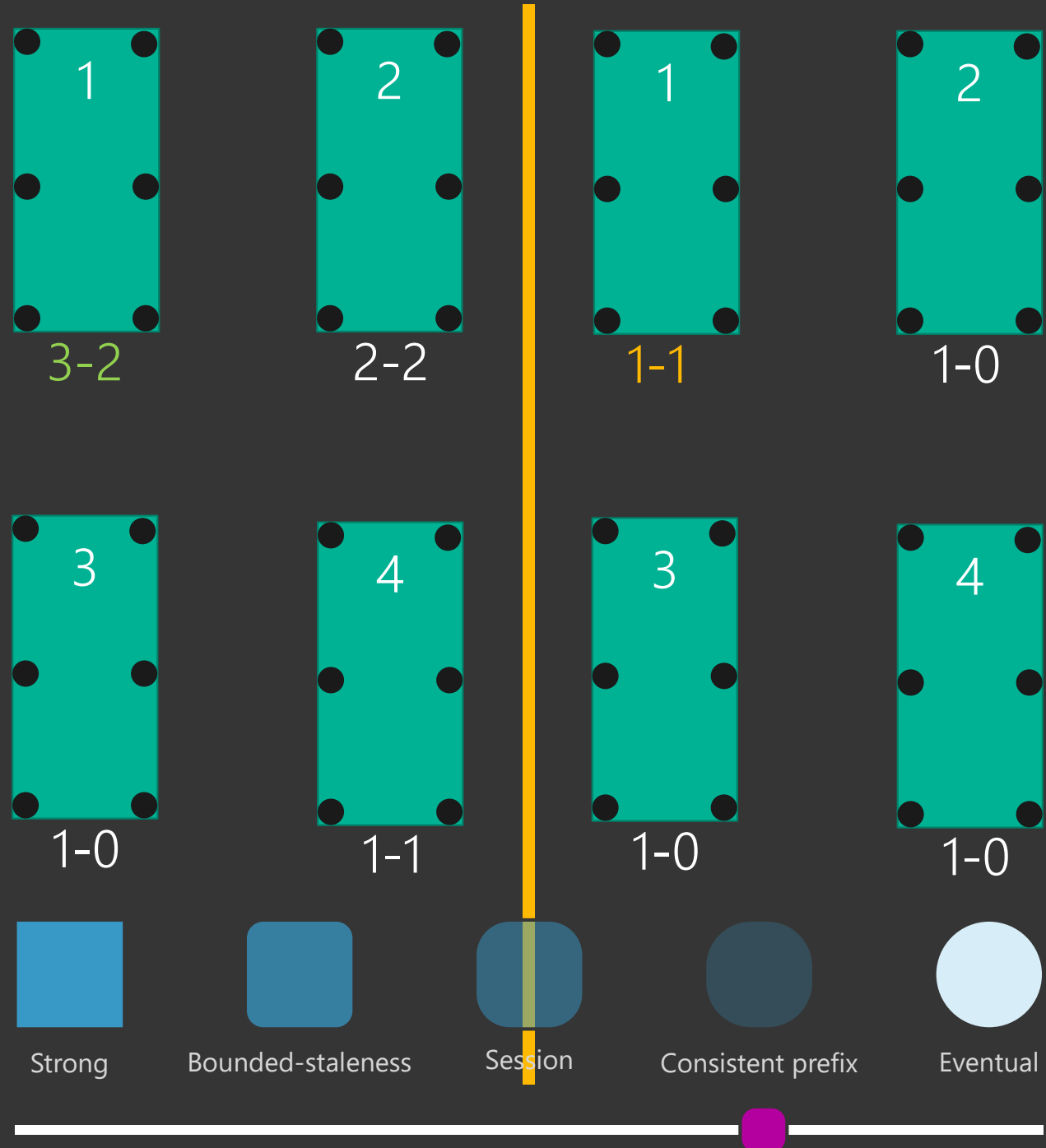


Eventual





- Read is consistent to a point in time
- No guarantee how current
- Good performance
- Excellent availability





Turnkey Global Distribution

Elastic scale out of storage and throughput

Guaranteed 99th percentile low latency

Five well-defined consistency models

- No guaranteed ordering/freshness of data
- Eventually “catches up”
- Highest performance
- Highest availability



Strong



Bounded-staleness



Session



Consistent prefix

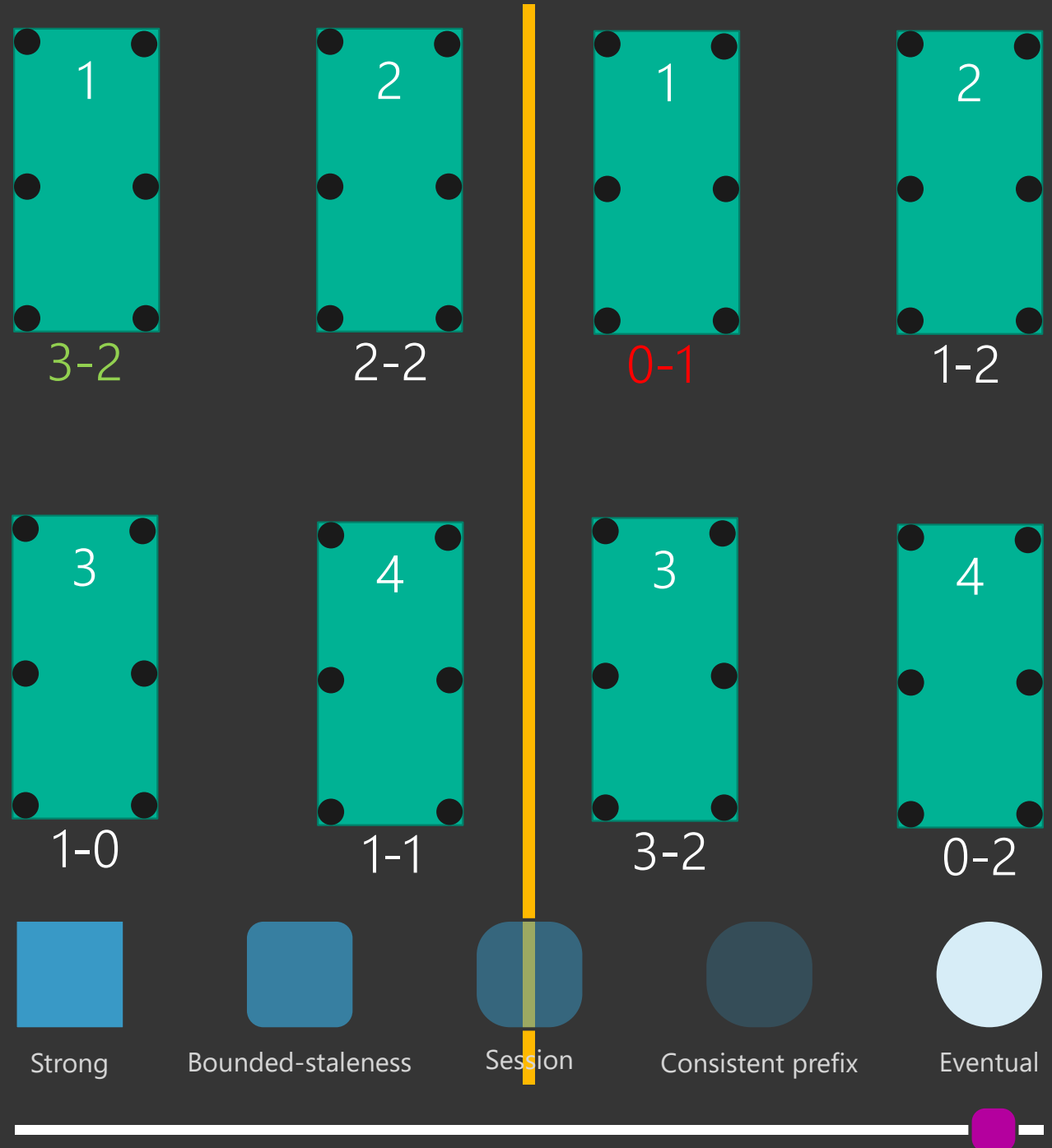


Eventual





- No guaranteed ordering/freshness of data
- Eventually “catches up”
- Highest performance
- Highest availability





Demo: Consistency Levels





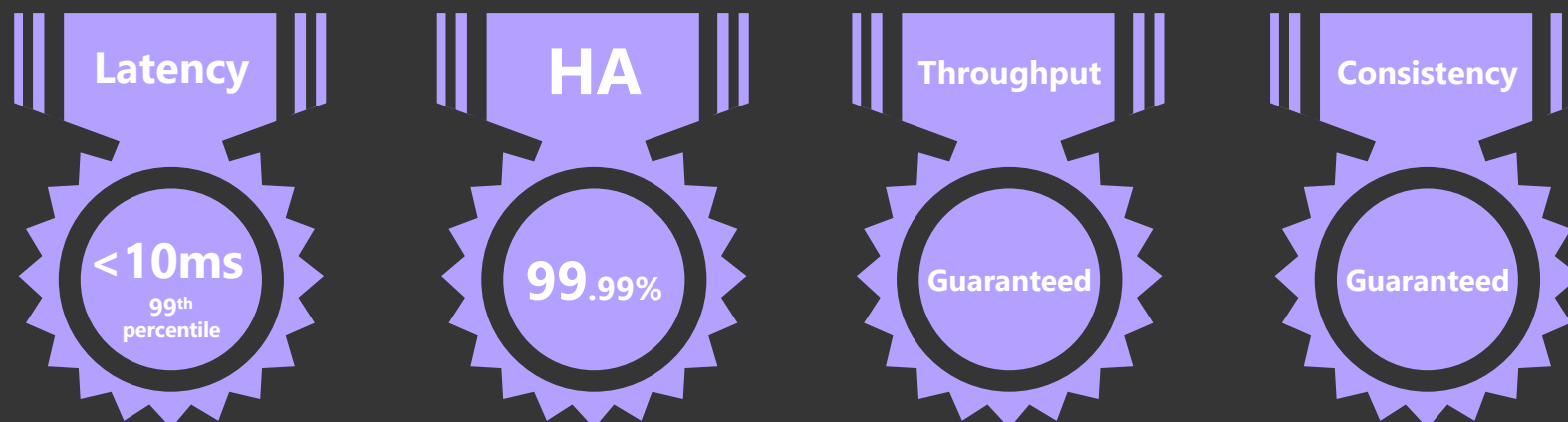
Turnkey Global Distribution

Elastic scale out of storage and throughput

Guaranteed 99th percentile low latency

Five well-defined consistency models

Comprehensive SLAs





1 RU = Read 1 KB Item with 10 Unique Properties



<https://jlik.me/ckv>

Item Size	Reads/second	Writes/second	Request units
1 KB	500	100	1,000 RU
1 KB	500	500	3,000 RU
4 KB	500	100	1,350 RU
4 KB	500	500	4,150 RU
64 KB	500	100	9,800 RU
64 KB	500	500	29,000 RU

Why Cosmos DB?



Automated
backups

Automatic
failover

ACID inside
partition

Change feed

Encryption
(rest & transit)

Geo-fencing

Geospatial
data

LINQ

Local emulator

ODBC

Recycle bin (30
days)

Security (ISO
27001, EUMC,
HIPAA)

Serverless
bindings

Sprocs,
triggers, UDFs



Table API

Got Interfaces?



SQL API



- Atoms – primitive types
- Records – structs
- Sequences – arrays of atoms, records, or other sequences

Atom-Record-Sequence (ARS)



Projection is used to map ARS to the API

Container

Atom-Record-Sequence (ARS)





Collection

Table

Graph

Projection

Container

Atom-Record-Sequence (ARS)





Doc



Key



Vertex

Projection

Item

Atom-Record-Sequence (ARS)





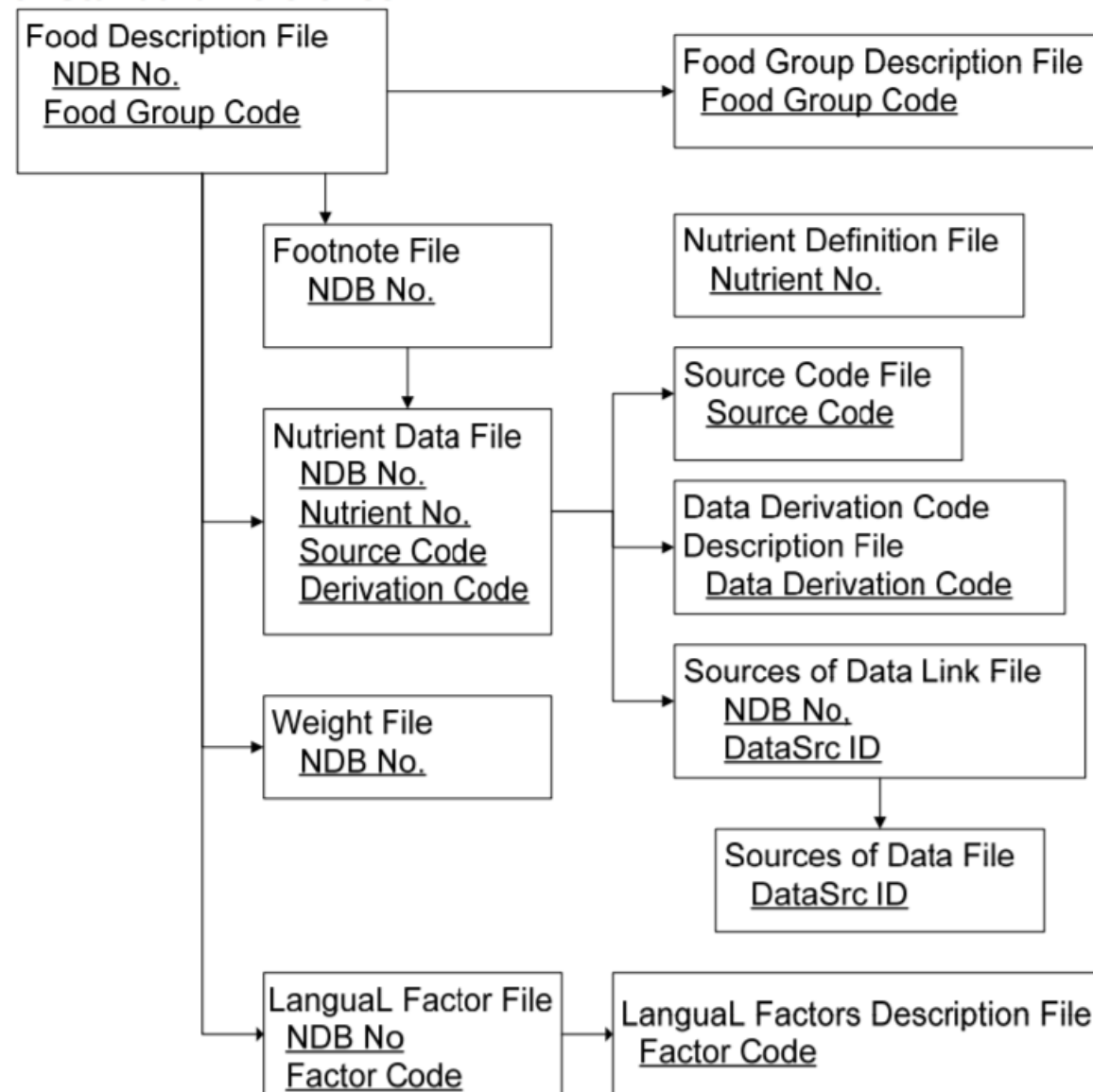
mongoDB®

USDA Database





Figure 1. Relationships among files in the USDA National Nutrient Database for Standard Reference *



* Underlined items denote key fields.

Source:

https://www.ars.usda.gov/ARSUserFiles/80400525/Data/SR/SR28/sr28_doc.pdf



Food Groups

Nutrient
Definitions

Food Items
Nutrient Data
Weights

- This could be implemented as a single collection with a type identifier



```
{  
  "foodId": 1,  
  "nutrients": [  
    {"name": "protein", "abbr": "pro"},  
    {"name": "carbohydrate", "abbr": "carb"}  
  ]  
}
```




mongoDB®



```
{  
  "foodId": 1,  
  "nutrients": {  
    "pro": {"name": "protein"},  
    "carb": {"name": "carbohydrate"}  
  }  
}
```



DEMO





Table API

Got Interfaces?



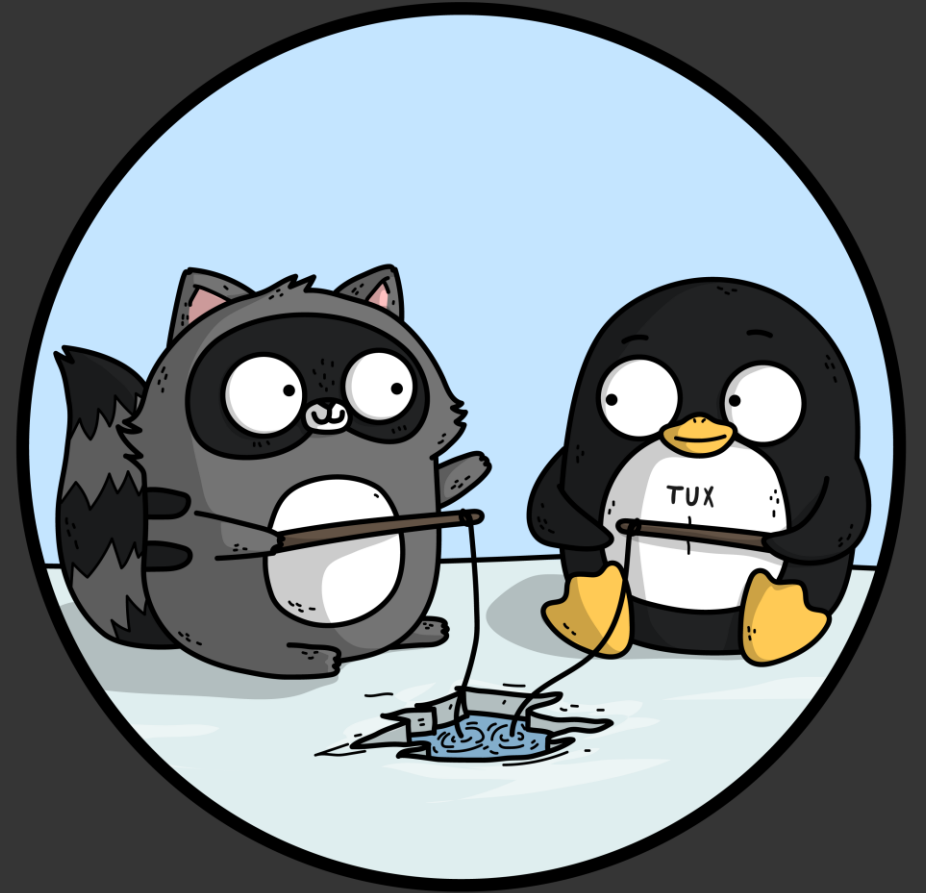
SQL API





Gremlin
 $G = (V, E)$

Flights





Gremlin
 $G = (V, E)$



```
[
  {"label": "airport",
    "id": "MAG",
    "latitude": [{"_value": -5.20707988739, "id": "2e0bfd38-95a3-4bbc-bac1-0e346fe9d3cc"}],
    "longitude": [{"_value": 145.789001465, "id": "58087297-6041-4d4b-ab4b-c938d310e3f6"}]
  },
  {"label": "flight",
    "distance": 229918.897192623,
    "id": "090cc119-b1aa-4f1f-96e5-545728191471",
    "_sink": "ADL",
    "_sinkLabel": "airport",
    "_vertexId": "WYA",
    "_vertexLabel": "airport",
    "_isEdge": true
  }
]
```



Table API

Got Interfaces?



SQL API





Table API



SQL API

Link Shortener





Table API

Got Interfaces?



SQL API



Cassandra API



- Public preview
- Details here: <https://aka.ms/cassandra-api>



```
// Connect to cassandra cluster (Cassandra API on Azure Cosmos DB supports only TLSv1.2)
var options = new Cassandra.SSLOptions(SslProtocols.Tls12, true, ValidateServerCertificate);
options.SetHostNameResolver((ipAddress) => CassandraContactPoint);
Cluster cluster = Cluster.Builder().WithCredentials(UserName, Password).WithPort(CassandraPort)
    .AddContactPoint(CassandraContactPoint).WithSSL(options).Build();
ISession session = cluster.Connect();
```

In Conclusion ...

- NoSQL: a different approach
- Azure Cosmos DB: one engine to rule them all ... but multiple APIs to bind them
- Scale, speed, and security – all turnkey, all serverless
- .NET-ready





 <https://blog.jeremylikness.com/>

 @JeremyLikness

 Jeremy.Likness@microsoft.com

USDA

<https://github.com/JeremyLikness/explore-cosmos-db>

Flights

<https://github.com/anthonychu/cosmosdb-gremlin-flights>

Link Shortener

<https://github.com/JeremyLikness/jlik.me>

Link Shortener Simplified

<https://github.com/JeremyLikness/ShortLink>

Try Cosmos DB Free:

<https://aka.ms/get-cosmos>

<https://aka.ms/cosmos-docs>

Global distribution: <https://jlik.me/ckw>

Consistency levels: <https://jlik.me/ckx>

Request Units calculator: <https://jlik.me/ckv>

Performance tips: <https://aka.ms/cosmos-perf>

