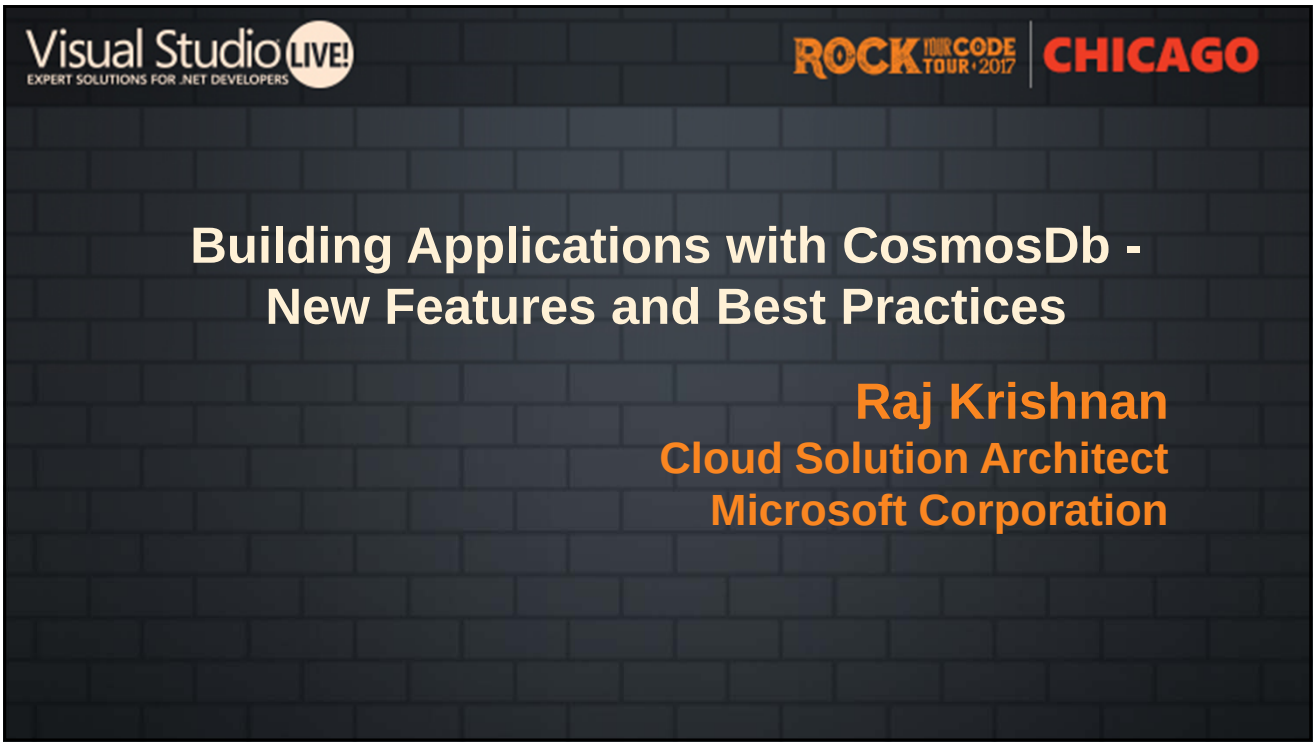


Session Objectives

- CosmosDb – The Microsoft NoSQL solution
  - Overview: What and When?
  - What's new?
- Building application with CosmosDb
  - Best Practices – Design, Develop and Monitor
- Tools and Techniques

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

“If all you have is a hammer, everything looks like a nail“

-Abraham Maslow

A Quick Primer on NoSQL

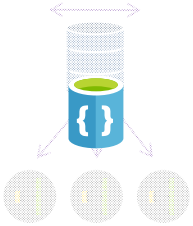
Database architectures

Relational

Scale up

Read optimized

Local index + query processing

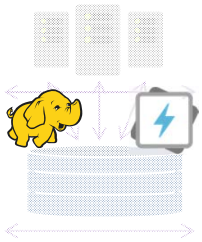


NoSQL

Scale out collocated

Write + read optimized

Local index + query processing



Data Lakes

Scale out disaggregated

Write optimized

No real-time index + query processing

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

NoSQL in a nutshell

NoSQL is varied

- Flexible Schema
- Key-value
- Wide-column
- Document-oriented
- Graph
- Multi-Model

CAP theorem

- In a distributed system you cannot achieve all three of Consistency, Availability and Partition tolerance. You can pick only two of:
- Consistency:
- Availability:
- Partition Tolerance:

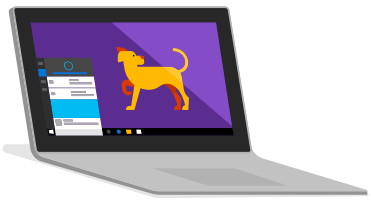
Let's build a Product Catalog
(comparing traditional vs schema-agnostic databases)

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

Let's start with a traditional schema-strict model…

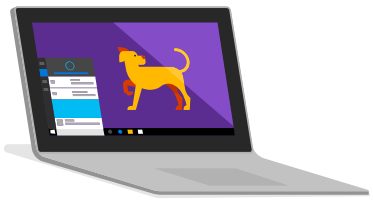
Item	Color	Microwave Safe	Liquid Capacity
Geek Mug	Graphite	Yes	16oz
Coffee Bean Mug	Tan	No	12oz

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS



Item	Color	Microwave Safe	Liquid Capacity
Geek Mug	Graphite	Yes	16oz
Coffee Bean Mug	Tan	No	12oz
Surface Book	Gray	???	???

!=




Item	Color	Microwave Safe	Liquid Capacity	CPU	Memory	Storage
Geek Mug	Graphite	Yes	16oz	???	???	???
Coffee Bean Mug	Tan	No	12oz	???	???	???
Surface Book	Gray	???	???	3.4 GHz Intel Skylake Core i7-6600U	16GB	1 TB SSD

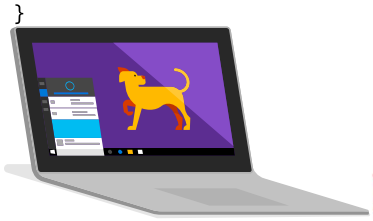
Not very efficient...

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

Schema-agnostic databases are a beautiful thing

```
{
  "ItemType": "Coffee Mug",
  "Name": "Geek Mug",
  "Color": "Graphite",
  "Capacity": {
    "value": 16,
    "units": "oz"
  },
  "Microwave-Safe": "yes",
  "Dishwasher-Safe": yes"
}
```

```
{
  "ItemType": "Laptop",
  "Name": "Surface Book",
  "Processor": [
    "2.4 GHz Core i5-6300U",
    "3.4 GHz Core i7-6600U"
  ],
  "Memory": [ "8GB", "16GB" ],
  "Storage": {
    "Type": "SSD"
    "Capacity": [ 128, 256, 512, 1024 ]
  }
}
```

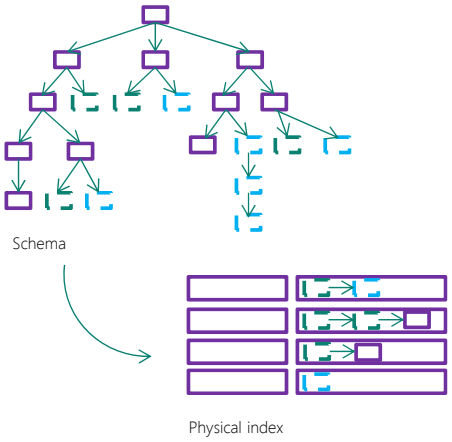



Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

An end to end application with DocumentDb

DEMO

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS



Schema agnostic indexing

- No schemas or secondary indices
- Highly write-optimized database engine
- Automatic and synchronous indexing
- Fully resource governed
- Hash, range, geo-spatial, time-series (+ columnar)
- Online and in-situ index transformations

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

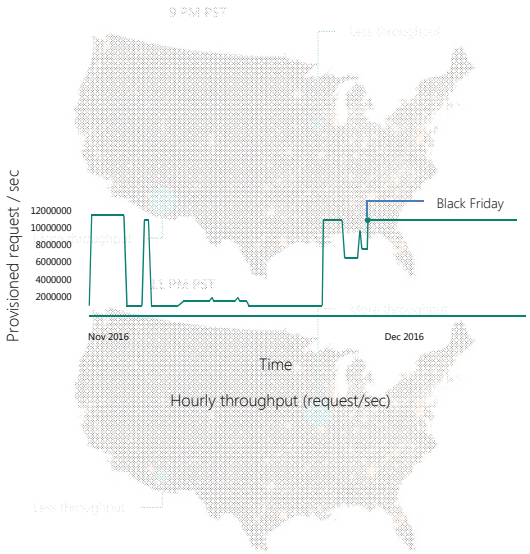
Rich SQL and JavaScript queries

No impedance mismatch

Query with SQL and JavaScript

Write business logic entirely in JavaScript with stored procedures and triggers

Multi-item ACID transactions with snapshot isolation

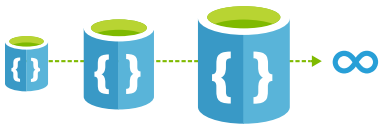


Elastically scalable throughput

Scale from 100 to 10s of millions of requests/sec across multiple regions

Pay by the hour

Automatic partition management



Elastically scalable storage

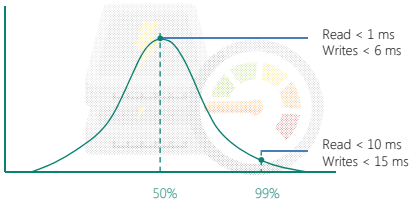
Independently scale storage and throughput

No partition management

Automatically indexed SSD storage

Global distribution across Azure regions

Automatic expiration via TTL



Guaranteed low latency

Globally distributed with reads and writes served from local region

Write optimized, latch-free database engine designed for SSDs

Synchronous and automatic indexing at sustained ingestion rates

Global distribution from the ground-up

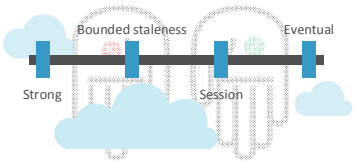
Worldwide presence

Automatic multi-region replication

Multi-homing APIs

Manual and automatic failovers

Latency, throughput, consistency, and availability guarantees



Well defined consistency models

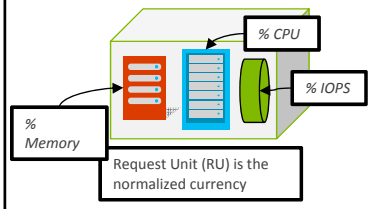
Global distribution forces us to navigate the CAP theorem

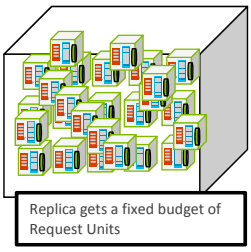
Four well-defined consistency levels to choose from

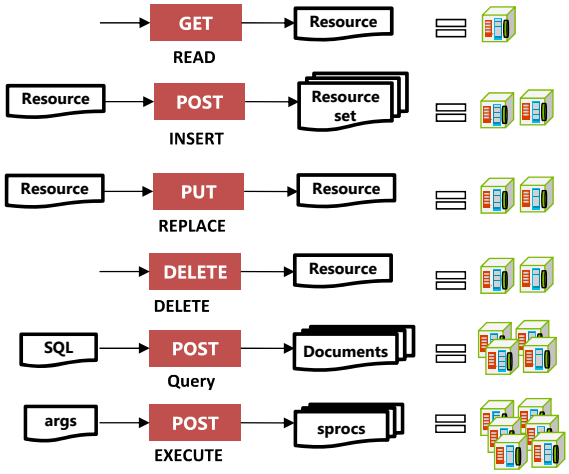
Intuitive with clear PACELC tradeoffs

Can be overridden on a per-request basis

Request units







Predictable Performance

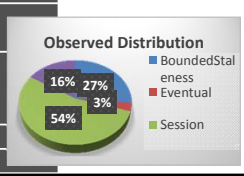
Most import metric in DocumentDB!

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

Consistency Models

LEFT TO RIGHT → *Relaxed consistency => better performance and availability*

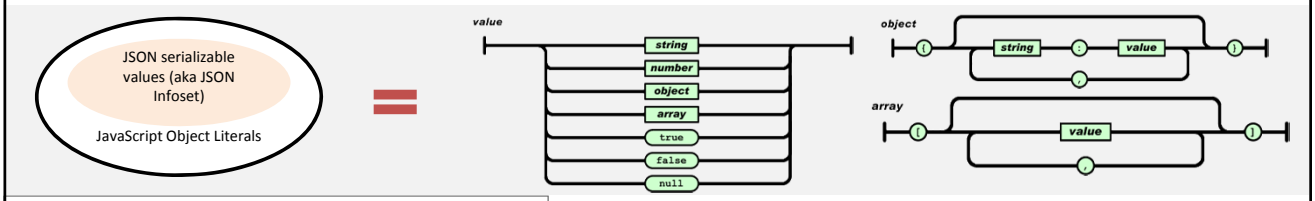
Consistency Level	Strong	Bounded Staleness	Session	Eventual
Total global order	Yes	Yes, outside of the "staleness window"	No, partial "session" order	No
Consistent prefix guarantee	Yes	Yes	Yes	Yes
Monotonic reads	Yes	Yes, across regions outside of the staleness window and within a region all the time	Yes, for the given session	No
Monotonic writes	Yes	Yes	Yes	Yes
Read your writes	Yes	Yes (in the write region)	Yes	No



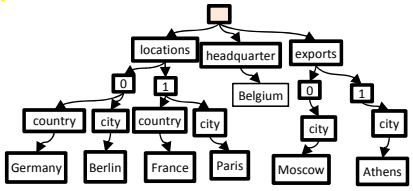
Query + Indexing

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

Documents as Trees



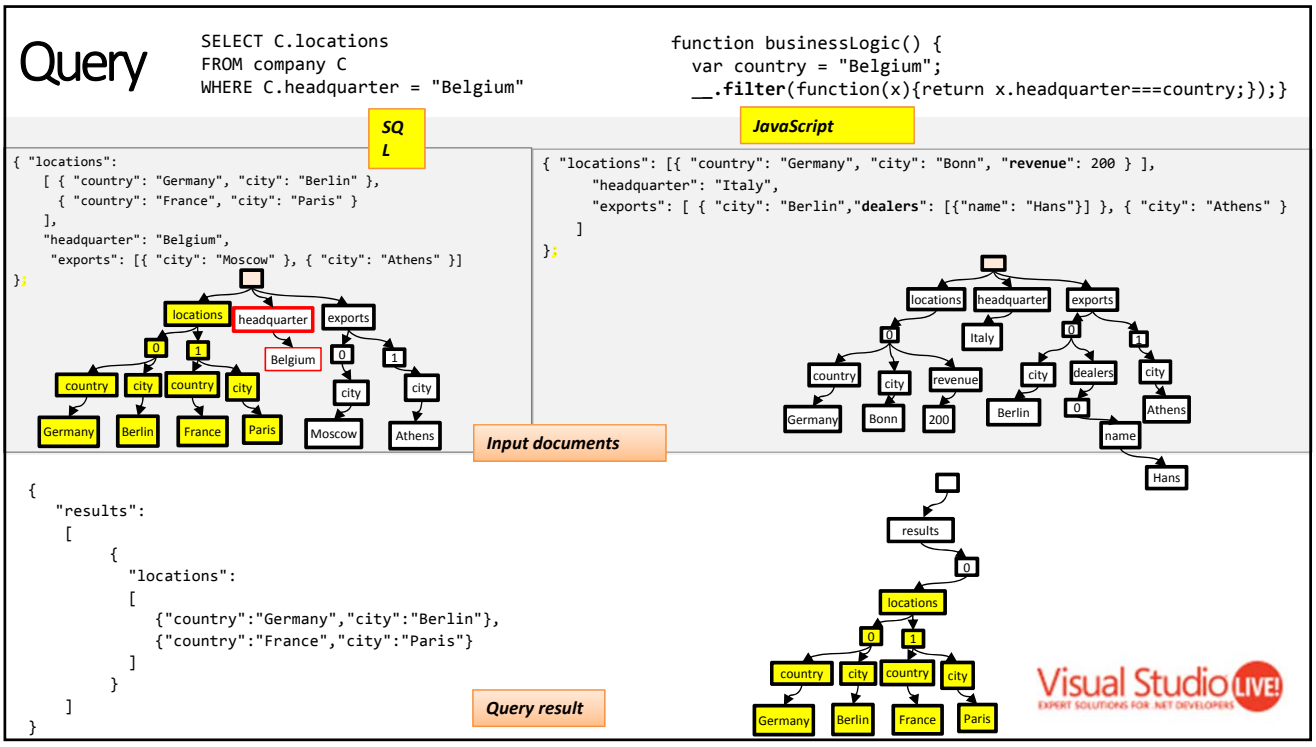

```
{
    "locations":
    [
      { "country": "Germany", "city": "Berlin" },
      { "country": "France", "city": "Paris" }
    ],
    "headquarter": "Belgium",
     "exports":[{ "city": "Moscow" },{ "city": "Athens"}]
};
```

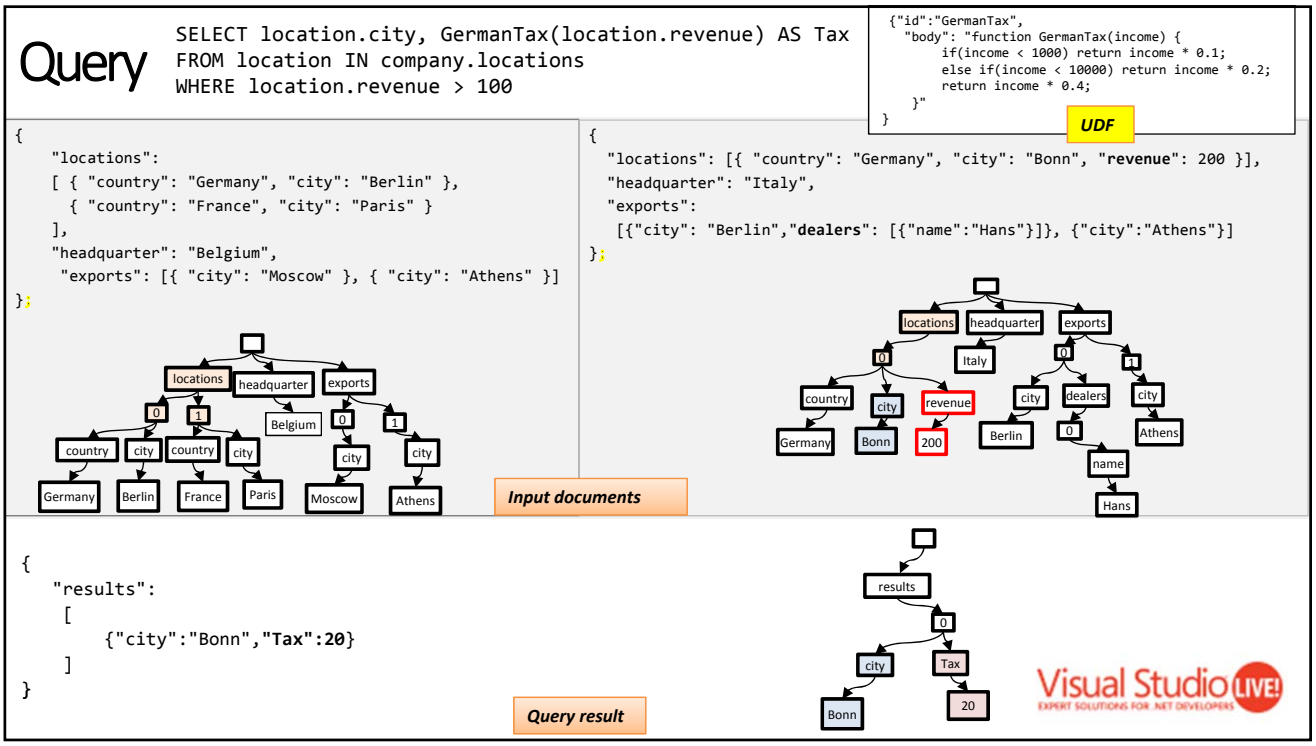


JSON document as tree

- Databases vary in their commitment to JSON data model
  - Relational stores – Require schema, indices & OR mapping; great query
  - Key-Value stores – Schema-free; values are opaque; poor query
- DocumentDB - Deep commitment to the JSON data model & JavaScript directly within the database engine
  - Automatic indexing of documents without requiring schema or secondary indices
  - SQL query dialect rooted in JSON; extensible via JavaScript
  - Efficient execution of application logic with (JavaScript) language integrated database transactions
  - Minimal impedance mismatch between the programming languages and the database type systems

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

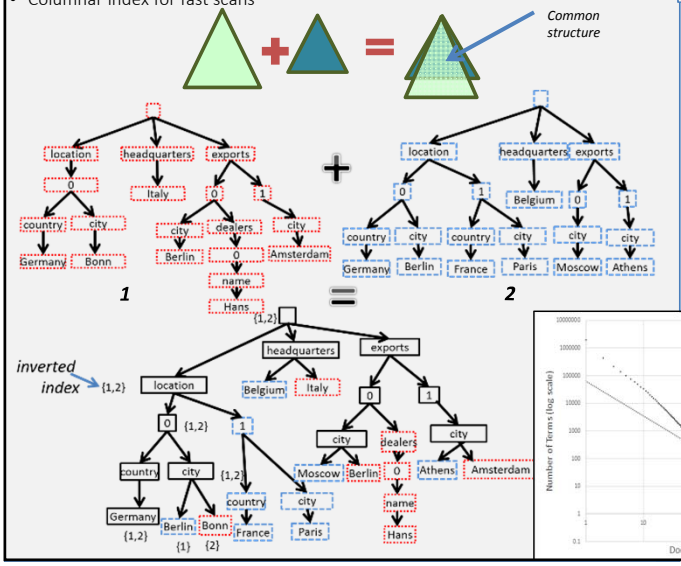

Query
SELECT C.locations
FROM company C
WHERE C.headquarter = "Belgium"
SQL
function businessLogic() {
var country = "Belgium";
__.filter(function(x){return x.headquarter===country;});}
JavaScript
{ "locations":
[{ "country": "Germany", "city": "Berlin" },
{ "country": "France", "city": "Paris" }
],
"headquarter": "Belgium",
"exports": [{ "city": "Moscow" }, { "city": "Athens" }]
};
{ "locations": [{ "country": "Germany", "city": "Bonn", "revenue": 200 }],
"headquarter": "Italy",
"exports": [{ "city": "Berlin","dealers": [{"name": "Hans"}] }, { "city": "Athens" }
]
};
Input documents
{
"results":
[
{
"locations":
[
{"country":"Germany","city":"Berlin"},
{"country":"France","city":"Paris"}
]
}
]
}
Query result
Visual Studio LIVE!

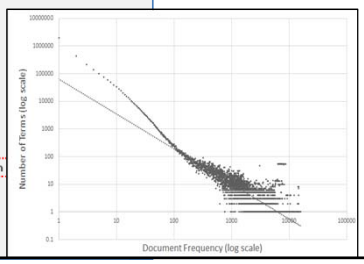

Query
SELECT location.city, GermanTax(location.revenue) AS Tax
FROM location IN company.locations
WHERE location.revenue > 100
{"id":"GermanTax",
"body": "function GermanTax(income) {
if(income < 1000) return income * 0.1;
else if(income < 10000) return income * 0.2;
return income * 0.4;
}"
}
UDF
{
"locations":
[{ "country": "Germany", "city": "Berlin" },
{ "country": "France", "city": "Paris" }
],
"headquarter": "Belgium",
"exports": [{ "city": "Moscow" }, { "city": "Athens" }]
};
{
"locations": [{ "country": "Germany", "city": "Bonn", "revenue": 200 }],
"headquarter": "Italy",
"exports":
[{"city": "Berlin","dealers": [{"name":"Hans"}]}, {"city":"Athens"}]
};
Input documents
{
"results":
[
{"city":"Bonn","Tax":20}
]
}
Query result
Visual Studio LIVE!

Schema Agnostic Indexing

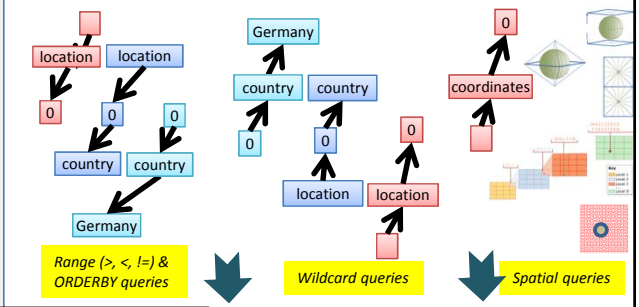
- Support for rich hierarchical, relational and analytical queries
- Different path encodings depending on index type
- Support for multi-tenancy requires fixed upper bound on index size

- Logically the index is a union of all the document trees
- Structure contributed by the interior nodes, instance values are the leaves
- Columnar index for fast scans





- Structural information and instance values are normalized into a unifying concept of JSON-Path



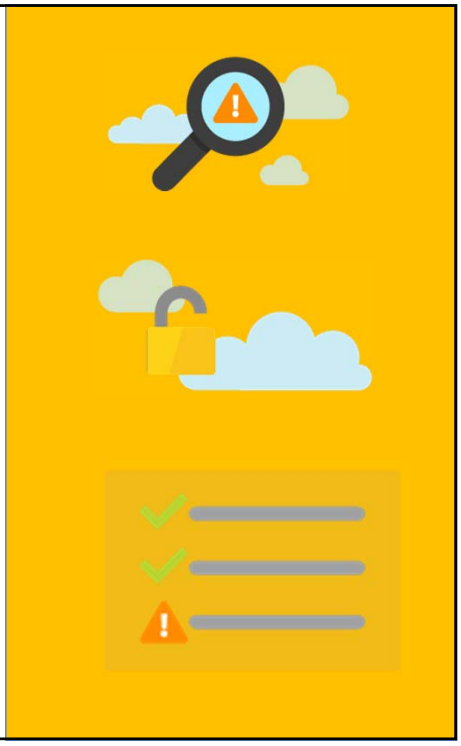
Terms	Postings List
$/location/0/	1, 2
location/0/country/	1, 2
location/0/city/	1, 2
0/country/Germany	1, 2
1/country/France	2
…	…
0/city/Moscow	2
0/dealers/0	2

Dynamic Encoding of Postings List (E-WAH/differential)

Security + Compliance

Security

- Firewall support to restrict access to specific IP addresses
- Built-in RBAC support
- Highly scalable AuthZ model with built-in support for users and permissions
- Fine grained/row level AuthZ
- All external (and internal) communication over SSL
- Coming Soon: Encryption@Rest

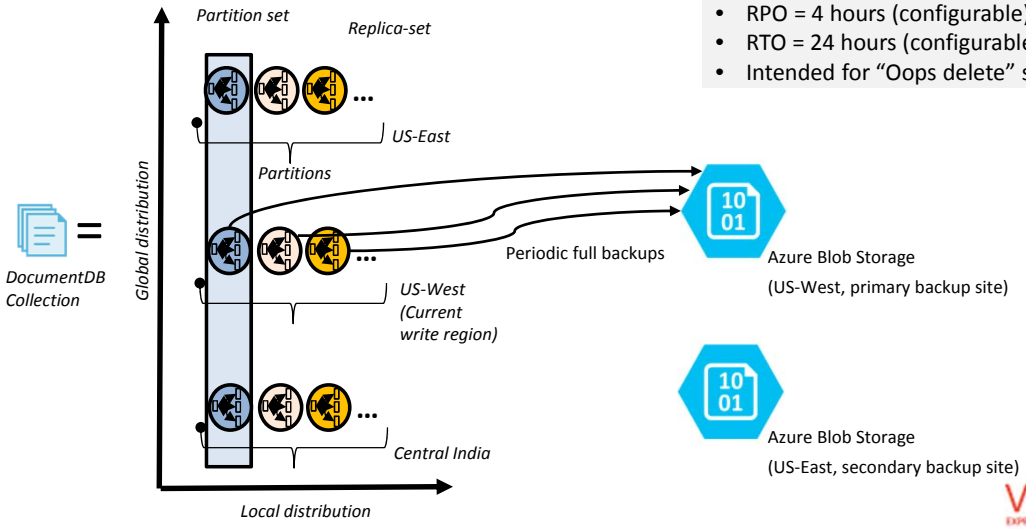


Compliance

Certification Details	Compliance Status
Strong Privacy and Security Commitments · No mining of customer data for advertising · No voluntary disclosure to law enforcement agencies	Achieved
Contractual commitment to meet US and EU data residency requirements	Achieved
ISO 27001	Achieved
ISO 27018	Achieved
EU Model Clauses (EUMC)	Achieved
HIPAA Business Associate Agreement	Achieved
PCI	Started (in progress)
SOC 1 & SOC 2	Started (in progress)
FedRAMP, IRS 1075, UK Official (IL2)	Started (in progress)
Health Information Trust Alliance (HITRUST)	Planned

Online Backups

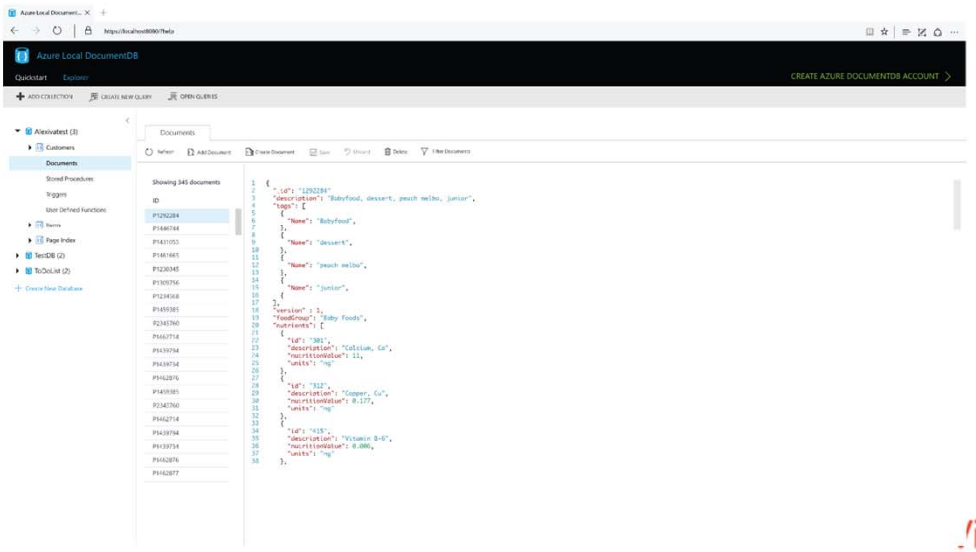
- Free, online, continuous, and full backups
- RPO = 4 hours (configurable)
- RTO = 24 hours (configurable)
- Intended for "Oops delete" scenarios (not for DR)



Recent Updates

DocumentDB Local Emulator

Free, downloadable, and high fidelity version of the cloud service for offline dev/test



Change Feed

- Lambda pattern with significantly lower TCO
- Single scalable database solution for **both** ingestion and query

Native Graph Processing

Gremlin and SQL query languages

Independently scalable graph engine (using Tinkerpop framework)

Globally distributed, elastically scalable, low latency, auto-indexed NoSQL database

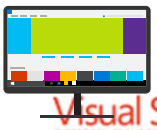
Common scenarios + use cases

Retail
- Product Catalog
- Ordering and Payment Pipelines
- Recommendations + Personalization
- Customer 360 View

IoT / Sensor Data
- Telemetry + Event Store
- Telematics
- Device Registry

Gaming
- Multiplayer Games
- Social Gameplay
- Leaderboards
- Game Analytics

Ad Technology + Social Analytics
- User behavior telemetry
- Personalization
- Customer 360 view



IoT / Sensor Data

Business Needs:

- Lots of sensors emitting telemetry
  => high rate of ingestion

  (Volume)

- React quickly to anomalies
  => low-latency queries

  (Velocity)

- Many different generations of devices
  => different schemas

  (Variety)

Microsoft Azure

Schneider Electric

Honeywell

TOYOTA

Ford

ECOLAB

Itron

Visual Studio LIVE!

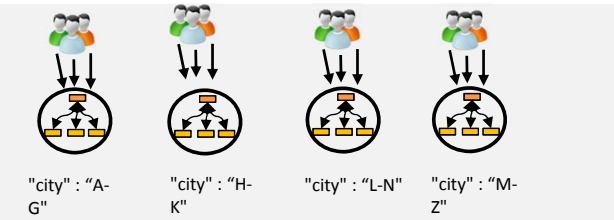
Learnings

- Design your schemas in the way you will use them in your application
- Be careful with the write-and-readback
- Go with Partitioned collections from the get-go
- Integration and migration tools from RDBMS require education
- Create a small wrapper for creation of collections, retrievals, etc.
- Use code to create smaller than 1100 RU Collections
- Hashing can reduce collection size

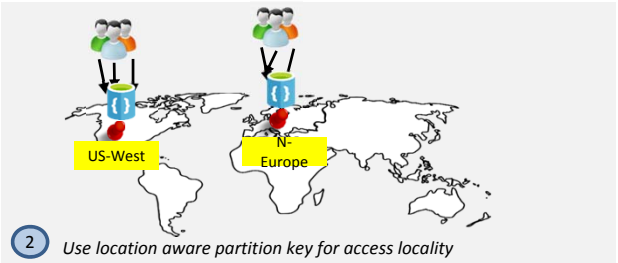
Document DB made it possible to have a highly performing globally distributed architecture that is flexible to adapt to business changes and easy to manage.

Visual Studio LIVE!

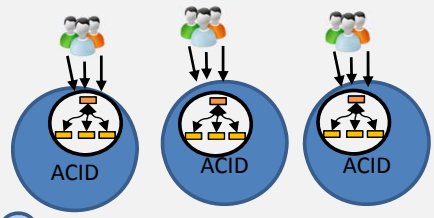
Best practices: Partitioning



1 *Select a partition key which provides even distribution*



2 *Use location aware partition key for access locality*



3 *Select a partition key which can be a transaction scope*

4 *Don't use timestamp for write heavy workloads. Use time ranges (hour, week, month, year etc) or for more even distribution*

Tips and Best Practices

- Controlling Cost
  - Reservation pricing for throughput;
  - Performance Level
    - https://www.documentdb.com/capacityplanner
    - https://azure.microsoft.com/en-us/pricing/calculator/?service=documentdb
    - https://azure.microsoft.com/en-us/documentation/articles/documentdb-request-units/#estimating-throughput-needs

Tips and Best Practices

- Indexing
  - customize the indexing policy for a collection and fine tune it based on performance and query consistency requirements. I
  - nclude / Exclude paths from indexing based on search requirements. https://azure.microsoft.com/en-us/documentation/articles/documentdb-indexing-policies

Tips and best practices

- **Collection Type and Partitioning:**
  - Use partitioned collections
  - **choose a partition key property** to distribute your workload evenly across partition key values.(10Gb / 10,000RU) https://azure.microsoft.com/en-us/documentation/articles/documentdb-partition-data/

Tips and best practices

- **Connection Policies:**
  - Use Direct Mode and TCP to setup connectivity to DocumentDB from client application

    https://azure.microsoft.com/en-us/documentation/articles/documentdb-performance-tips/#connection-protocol