



AZ-203 Developer Training

Develop for the cloud and for Azure storage



Agenda

- ★ NoSQL & NoSQL Database
- ★ Cosmos DB
- ★ Local Emulator
- ★ Multiple APIs and Data Models
- ★ Request Units
- ★ Exceeding Reserved Throughput Limits
- ★ Whiteboarding the Cost
- ★ Request Unit Calculator
- ★ Achieving Elastic Scale
- ★ Partition Key
- ★ Cross Partition Queries
- ★ Replication-Why
- ★ Turnkey Global Distribution
- ★ NoSQL & NoSQL Database
- ★ Replication and Consistency
- ★ Consistency Levels
- ★ Document Database
- ★ Data Modeling: Relational vs. Document
- ★ Cosmos DB Resource Model
- ★ Resource Properties, Self-Links, and URIs
- ★ Data Migration Tool
- ★ Rich Query with SQL
- ★ SQL Operators and Functions
- ★ Client Development
- ★ Introducing the .NET SDK for the SQL API
- ★ Indexing Policies

Agenda

- ★ Users, Permissions, and Resource Tokens
- ★ Server-side Programming Model
- ★ Writing Server-side Code
- ★ Working with Triggers
- ★ Creating User-Defined Functions (UDFs)
- ★ Table API
- ★ Cosmos DB Graph Database
- ★ Graph Database Scenarios
- ★ Vertices and Edges
- ★ Populating the Graph
- ★ Bi-Directional Relationships
- ★ Writing Gremlin Queries
- ★ Demo: Busy World Traveler
- ★ Multi-Model Comic Book Catalog
- ★ Azure Table Storage



NoSQL & NoSQL Database

What Is NoSQL?

IntelliPaat

IntelliPaat



IntelliPaat
Microsoft

IntelliPaat

mongoDB.

IntelliPaat

IntelliPaat

A stylized blue eye with a sun-like iris, representing the Cassandra database system.

cassandra

IntelliPaat

IntelliPaat

IntelliPaat

APACHE
HBASE The Apache HBase logo, featuring the word "APACHE" in a small font above "HBASE" in large red letters, with a black orca icon to the right.

The Amazon DynamoDB logo, showing three yellow cubes stacked vertically next to the word "amazon" and "DynamoDB".

DynamoDB

IntelliPaat

The OrientDB logo, featuring a stylized orange and grey flame-like graphic next to the word "OrientDB".

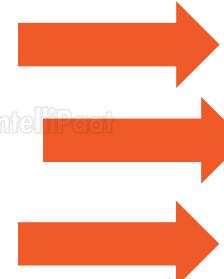
OrientDB

The ArangoDB logo, featuring a green avocado icon next to the word "ArangoDB".

ArangoDB

What Is NoSQL?

3Vs



Volume
Variety
Velocity

Every 60 seconds

- 98,000+ tweets
- 695,000 status updates
- 11 million instant messages
- 698,445 Google searches
- 168 million+ emails sent
- 1,820TB of data created
- 217 new mobile web users



What Is a NoSQL Database?



Distributed

Replicas ensure high throughput/availability, and low latency

Scale-out

Horizontal partitioning enables virtually limitless storage and throughput

Schema-free

Document, table, graph, and columnar data models



Cosmos DB

What Is Cosmos DB?



Evolution of DocumentDB

Scalable NoSQL document database Low latency (single-digit millisecond)

Virtually unlimited scale

Scale storage with server-side partitioning Scale throughput with variable request units

Turnkey global distribution

Point-and-click control over where your data gets geo-replicated

Multi-model / Multi-API

No longer exclusively a document database Also supports tables, graph, and columnar

Cosmos DB Evolution



2010 - 2014

Internal Microsoft DocumentDB service
Office, OneNote, Xbox

Aug 21, 2014

Azure DocumentDB preview
SQL grammar over schema-free JSON Tunable
throughput, indexing, consistency Server-side ACID
transactions
Runs on Azure (fully managed PaaS)

Apr 8, 2015

Azure DocumentDB GA (General Availability)

Apr - Aug
2015

New features
ORDER BY and string range queries
Geospatial support
SDK partitioning support

Mar 31, 2016

New features
Partitioned collections
Geo-replication
Wire support for MongoDB (BSON)

May 10, 2017

Azure Cosmos DB Turnkey global distribution Horizontal
partitioning

Low latency (single-digit millisecond)

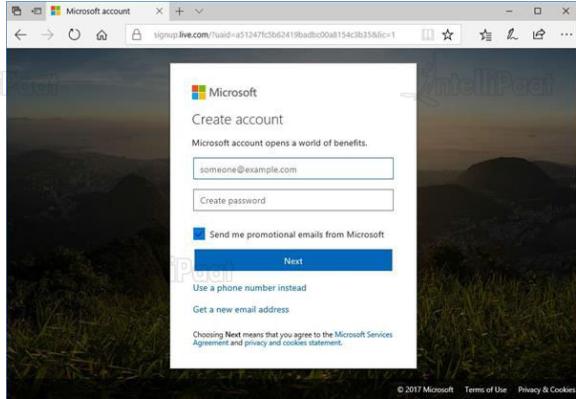
SLAs for guarantees on 99.99% availability, throughput,
latency, and consistency

Local emulator

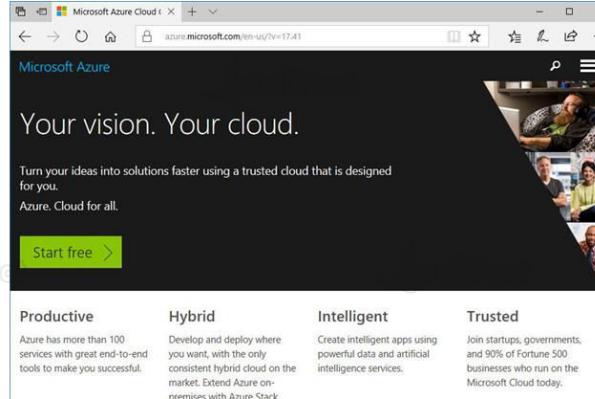
Multi-model, multi-API database

JSON (SQL API) BSON (MongoDB API)
Graph (Gremlin API) Key-value (Table API)
Columnar (Cassandra API)

Getting Started



Get a Microsoft account
<http://signup.live.com>



Get a Microsoft Azure subscription
<http://azure.microsoft.com>

Or,



Download the local emulator
<https://aka.ms/cosmosdb-emulator>



Local Emulator

Introducing the Local Emulator



- ★ Emulate Cosmos DB in a local development environment
- ★ Supports identical functionality as Azure Cosmos DB in the cloud



- ★ No need for:
 - ★ Azure subscription
 - ★ Cosmos DB account
 - ★ Internet connection



- ★ Develop and test locally
 - ★ Incur no costs
 - ★ Deploy to the cloud when ready

Hands-On

- ★ Creating a Cosmos DB account
- ★ Installing the Cosmos DB local emulator
- ★ Creating a collection
- ★ Creating documents





Multiple APIs and Data Models

Multiple APIs and Data Models



Atom Record Sequence

It's all ARS under the covers, projected as any desired data model

ARS

Document



SQL API (JSON) MongoDB API (BSON)

Key-Value



Table API

(replaces Azure Table Storage)

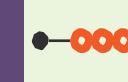
Graph



Gremlin API

(graph traversal language)

Columnar

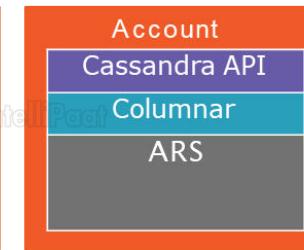
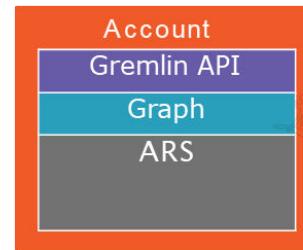
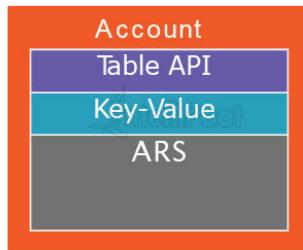
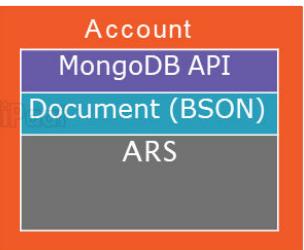
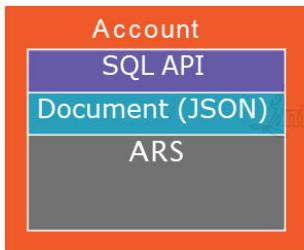


Cassandra API

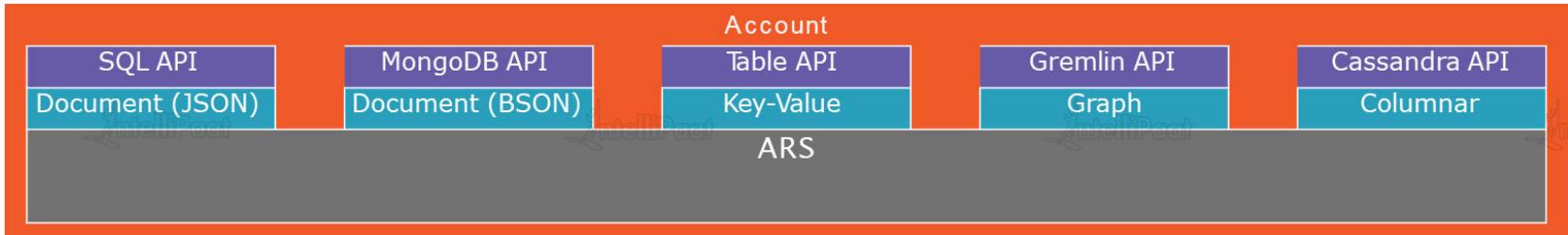
Multiple APIs and Data Models



Today



In the future



Course Structure



Concepts

- ★ Throughput
- ★ Partitioning
- ★ Global distribution

How-to

- ★ SQL API
- ★ Table API
- ★ Gremlin API

Measuring Performance

Performance is measured by the time taken to process a request.



Latency

How fast is the response for a given request?



Throughput
How many requests can be served within a specific period of time?





Request Units

Introducing Request Units



Throughput Currency

Blended measure of computational cost (CPU, memory, disk I/O, network I/O)

All Requests are Not Equal

Every Cosmos DB response header shows the RU charge for the request

Request Units are Deterministic

The same request will always require the same number of request units

Reserving Request Units



Provision request units per second (RU/s)
How many request units (not requests) per second are available to your application

Exceeding reserved throughput limits
Requests are “throttled”

Monitoring Request Unit Consumption



The image shows two side-by-side screenshots of the Azure Cosmos DB Emulator interface. Both screenshots have a header bar with tabs for 'Data Explorer - Microsoft A' and 'Azure Cosmos DB Emul'. Below the header is a URL bar showing 'localhost:8081/_explorer/index.html'. The main area is titled 'Azure Cosmos DB Emulator'.

Left Screenshot:

- Left Panel:** Shows 'Quickstart', 'Explorer' (selected), and 'Feedback'.
- Middle Panel:** Shows 'COLLECTIONS' with a 'Families' collection selected. Under 'Documents', there is a 'Query 1' tab with the following query:

```
1 SELECT * FROM c  
2 WHERE c.address.zipCode = '60601'
```
- Bottom Panel:** Shows the results of the query:

```
1 {  
    "familyName": "Smith",  
    "address": {  
        "addressLine": "123 Main Street",  
        "city": "Chicago",  
        "state": "IL",  
        "zipCode": "60601"  
    },  
    "parents": [  
        "Peter",  
        "Alice"  
    ],  
    "kids": [  
        "Adam",  
        "Jacqueline"  
    ]  
}
```

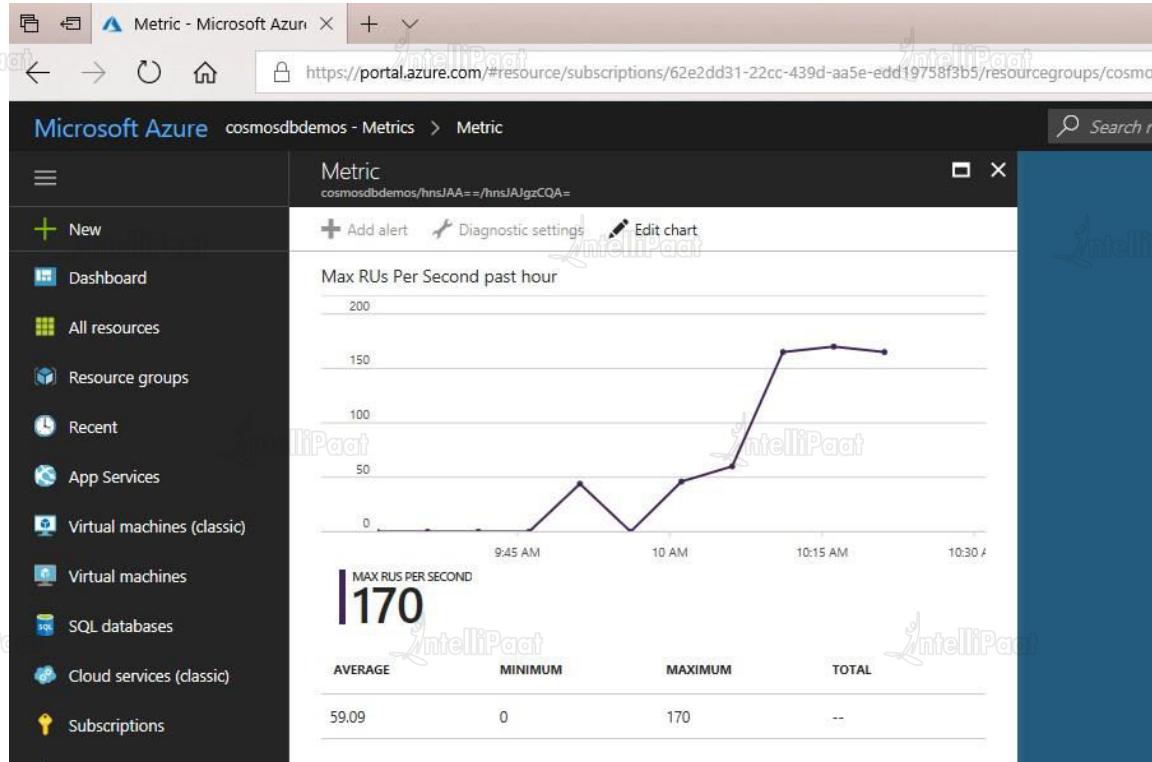
Right Screenshot:

- Left Panel:** Shows 'Quickstart', 'Explorer' (selected), and 'Feedback'.
- Middle Panel:** Shows 'COLLECTIONS' with a 'Families' collection selected. Under 'Documents', there is a 'Query 1' tab with the following query:

```
1 SELECT * FROM c  
2 WHERE c.address.city = 'Chicago'
```
- Bottom Panel:** Shows the results of the query:

```
1 {  
    "familyName": "Jones",  
    "address": {  
        "addressLine": "567 Harbor Boulevard",  
        "city": "Chicago",  
        "state": "IL",  
        "zipCode": "60603"  
    },  
    "parents": [  
        "David",  
        "Diana"  
    ],  
    "kids": [  
        "Evan"  
    ]  
}
```

Monitoring Request Unit Consumption



Monitoring Request Unit Consumption



The screenshot shows a Microsoft Visual Studio code editor window titled "Program.cs". The code is written in C# and demonstrates how to create a document collection and monitor the Request Units (RUs) consumed by the operation.

```
53 var collection = UriFactory.CreateDocumentCollectionUri("Families", "Families");
54
55 dynamic documentDefinition = new
56 {
57     familyName = "Smith",
58     address = new
59     {
60         addressLine = "123 Main Street",
61         city = "Chicago",
62         state = "IL",
63         zipCode = "60601"
64     },
65     parents = new string[]
66     {
67         "Peter",
68         "Alice"
69     },
70     kids = new string[]
71     {
72         "Adam",
73         "Jacqueline",
74         "Joshua"
75     }
76 };
77
78 var result = await client.CreateDocumentAsync(collection, documentDefinition);
79 var consumedRUs = result.RequestCharge;
80
81 Console.WriteLine($"Cost to create document: {consumedRUs} RUs"); < 194ms elapsed
82
83
84
85
```

A tooltip appears over the line of code `Console.WriteLine(\$"Cost to create document: {consumedRUs} RUs");` with the text "Cost to create document: 8.95 RUs" and a timestamp "194ms elapsed".

Exceeding Reserved Throughput Limits

Exceeding Reserved Throughput Limits



The screenshot shows a Microsoft Visual Studio interface during debugging. The main window displays a C# file named `Program.cs` with the following code:

```
86
87
88
89
90     var result = await client.CreateDocumentAsync(collection, documentDefinition);
91     Console.WriteLine("Document created successfully");
92 }
93 catch (DocumentClientException ex)
94 {
95     if (ex.StatusCode.HasValue && (int)ex.StatusCode.Value == 429)
96     {
97         Console.WriteLine($"Can't create document; request was throttled");
98     }
99     else
100    {
101        throw ex;
102    }
103
104
105
106
107
108
109
110
111
112
113 }
```

The code uses the `Azure SDK for .NET` to interact with Azure DocumentDB. A `try`-`catch` block handles a `DocumentClientException`. The exception is caught when the status code is `429`, which corresponds to "Request rate is large".

In the bottom right corner of the code editor, there is an `Exception Thrown` dialog box. It shows the following details:

- Exception Thrown**: `Microsoft.Azure.Documents.DocumentClientException`: Message: "Errors": ["Request rate is large"]
- ActivityId**: bf8825ff-ced9-464b-bcfb-0dc37f052c9a, Request URI: /apps/00465038-5ef7-41d8-be97-d498c6d8a08a/services/e443be7b-9d76-40c1-9487-5c414ef73dee/partitions/e136f4dc-e326-4619-9d8e-ba1873d08820replicas/13155400065082641p/, RequestStats:, SDK: Microsoft.Azure.Documents.Common/1.17.101.1'
- View Details | Copy Details**
- Exception Settings**
- Break when this exception type is thrown
Except when thrown from:
 ConsoleApp2.exe
- [Open Exception Settings](#) | [Edit Conditions](#)

Exceeding Reserved Throughput Limits



Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Process: [19640] ConsoleApp2.exe Lifecycle Events Thread: [22800] Worker Thread Stack Frame: ConsoleApp2.Program.RunDemo_CreateDocs()

Program.cs

```
86
87
88 try
89 {
90     var result = await client.CreateDocumentAsync(collection, documentDefinition);
91     Console.WriteLine("Document created successfully");
92 }
93 catch (DocumentClientException ex)
94 {
95     if (ex.StatusCode.HasValue && (int)ex.StatusCode.Value == 429) < 13ms elapsed
96     {
97         Console.WriteLine($"Can't create document; request was throttled");
98     }
99     else
100    {
101        throw ex;
102    }
103 }
```

SQL Server Object Explorer

Diagnostic Tools Solution Explorer Team Explorer

Autos

Name	Value	Type
(int)ex.StatusCode.Value	429	int
ex	{"Message: \\"Errors\\":\\\"Request ra Microsoft.Azure.Documents.DocumentClientException\\\""} "Message: (\\"Errors\\":\\\"Request ra Microsoft.Azure.Documents.DocumentClientException\\\")"	System.Net.HttpStatusCode?
ex.StatusCode	429	System.Net.HttpStatusCode
ex.StatusCode.HasValue	true	bool
ex.StatusCode.Value	429	System.Net.HttpStatusCode

Call Stack Breakpoints Exception Settings Command Window Immediate Window Output

Ready Ln 95 Col 25 Ch 7 INS Add to Source Control

Exceeding Reserved Throughput Limits



Program.cs

```
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103 } throw ex;
```

ActivityId: 36263b8f-7a3f-419e-a253-07f21a2cbefa
Message: {"Errors": ["Request rate is large"]}
InnerException: null
RequestCharge: 0.38
ResponseHeaders: {System.Collections.Specialized.NameValueCollection} {00:00:00.50000000}
RetryAfter: "Microsoft.Azure.Documents.Client"
Source: " at Microsoft.Azure.Documents.Client.ClientExtensions.<ParseResponseAsync>d__4.MoveNext()> End..."
StackTrace: 429
StatusCode: Void MoveNext()

Autos

Name	Value	Type
(int)ex.StatusCode.Value	429	int
ex	{"Message: {"Errors": ["Request rate is large"]}"}	Microsoft.Azure.Documents.DocumentClientException
ex.StatusCode	429	System.Net.HttpStatusCode?
ex.StatusCode.HasValue	true	bool
ex.StatusCode.Value	429	System.Net.HttpStatusCode

Call Stack Breakpoints Exception Settings Command Window Immediate Window Output

Ready Ln 89 Col 22 Ch 7 INS ↑ Add to Source Control ▾

Exceeding Reserved Throughput Limits



Progress Telerik Fiddler Web Debugger

File Edit Rules Tools View Help GET /book GeoEdge

WinConfig Replay Go Stream Decode | Keep All sessions Any Process Find Save Browse Clear Cache TextWizard

#	Result	Protocol	Host	URL
2643	201	HTTPS	cosmosdbdemos-westus.d...	/ dbs/Families/colls/Families/docs
2644	429	HTTPS	cosmosdbdemos-westus.d...	/ dbs/Families/colls/Families/docs
2645	429	HTTPS	cosmosdbdemos-westus.d...	/ dbs/Families/colls/Families/docs
2647	429	HTTPS	cosmosdbdemos-westus.d...	/ dbs/Families/colls/Families/docs
2649	200	HTTPS	outlook.office365.com	/ EWS/Exchange.asmx
2653	200	HTTPS	outlook.office365.com	/api/emsmdb/?MailboxId=1b9a...
2654	200	HTTPS	outlook.office365.com	/api/emsmdb/?MailboxId=0001...
2660	200	HTTPS	management.azure.com	/batch/api-version=2015-11-01
2666	200	HTTPS	outlook.office365.com	/api/emsmdb/?MailboxId=cab1...
2667	200	HTTPS	outlook.office365.com	/api/emsmdb/?MailboxId=cab1...
2668	200	HTTPS	outlook.office365.com	/ EWS/Exchange.asmx
2674	200	HTTPS	outlook.office365.com	/api/emsmdb/?MailboxId=cab1...
2676	200	HTTPS	outlook.office365.com	/ EWS/Exchange.asmx
2682	200	HTTPS	outlook.office365.com	/ EWS/Exchange.asmx
2683	200	HTTPS	outlook.office365.com	/ EWS/Exchange.asmx
2684	200	HTTPS	outlook.office365.com	/api/emsmdb/?MailboxId=cab1...
2688	200	HTTPS	management.azure.com	/batch/api-version=2015-11-01
2695	200	HTTPS	outlook.office365.com	/api/emsmdb/?MailboxId=cab1...
2696	200	HTTPS	outlook.office365.com	/api/emsmdb/?MailboxId=0001...
2698	200	HTTPS	outlook.office365.com	/api/emsmdb/?MailboxId=1b9a...
2702	200	HTTPS	outlook.office365.com	/api/nspi/?MailboxId=cab11a...
2705	200	HTTPS	outlook.office365.com	/api/emsmdb/?MailboxId=cab1...

FiddlerScript Log Filters AutoResponder Composer

Statistics Inspectors Headers TextView SyntaxView WebForms HexView Auth Cookies Raw

JSON XML

POST https://cosmosdbdemos-westus.documents.azure.com/dbs/Families/C...
x-ms-documentdb-partitionkey: []
x-ms-date: Sun, 19 Nov 2017 15:58:52 GMT
authorization: type%3dmaster%26ver%3d1.0%26sig%3dTdTnQeY3PzmG6vNqHb0t
x-ms-session-token: 0:862
Cache-Control: no-cache
x-ms-consistency-level: Session

Find... (press Ctrl+Enter to highlight all) View in Notepad

Transformer Headers TextView SyntaxView ImageView HexView WebView

Auth Caching Cookies Raw JSON XML

HTTP/1.1 429 Too Many Requests
Content-Type: application/json
Server: Microsoft-HTTPAPI/2.0
x-ms-retry-after-ms: 506
x-ms-schemaversion: 1.3
x-ms-quorum-acked-lsn: 881
x-ms-substatus: 3200
x-ms-current-write-quorum: 3
x-ms-current-replica-set-size: 4
x-ms-xp-role: 1
x-ms-global-committed-lsn: 881
x-ms-number-of-read-regions: 0
x-ms-request-charge: 0.38
x-ms-service-revision: version=1.17.101.1

Find... (press Ctrl+Enter to highlight all) View in Notepad

All Processes 1 / 2,022 https://cosmosdbdemos-westus.documents.azure.com/dbs/Families/colls/Families/docs



Whiteboarding the Cost

Whiteboarding the Cost



Application checklist

1. What does a typical item look like?
2. What are the typical queries that users will run?
3. How many writes per second are required?
4. How many queries per second are required?
5. What is the acceptable consistency level?
6. What is the indexing policy?

Estimating throughput needs

Item size	Reads/second	Writes/second	Request units
1 KB	500	100	$(500 * 1) + (100 * 5) = 1,000 \text{ RU/s}$
1 KB	500	500	$(500 * 1) + (500 * 5) = 3,000 \text{ RU/s}$
4 KB	500	100	$(500 * 1.3) + (100 * 7) = 1,350 \text{ RU/s}$
4 KB	500	500	$(500 * 1.3) + (500 * 7) = 4,150 \text{ RU/s}$
64 KB	500	100	$(500 * 10) + (100 * 48) = 9,800 \text{ RU/s}$
64 KB	500	500	$(500 * 10) + (500 * 48) = 29,000 \text{ RU/s}$

* Based on Session consistency indexing policy set to None.



Request Unit Calculator

Request Unit Calculator



"Waiting for response fr x + https://www.documentdb.com/capacityplanner

Azure Cosmos DB

Estimate Request Units and Data Storage

Azure Cosmos DB is offered in units of solid-state drive (SSD) backed storage and throughput. Request units measure Azure Cosmos DB throughput per second, and request unit consumption varies by operation and JSON document. Use this calculator to determine the number of request units per second (RU/s) and the amount of data storage needed by your application. Read the [Request Units in Azure Cosmos DB](#) article for more information.

Add one or more JSON documents that are each representative of one type of document used by your application.

Sample Document 1 X

Sample JSON document: [Upload Document](#)

Number of documents: ⓘ

Create / second: ⓘ

Read / second: ⓘ

Update / second: ⓘ

Delete / second: ⓘ

+ Add an additional sample document.

Estimated Total

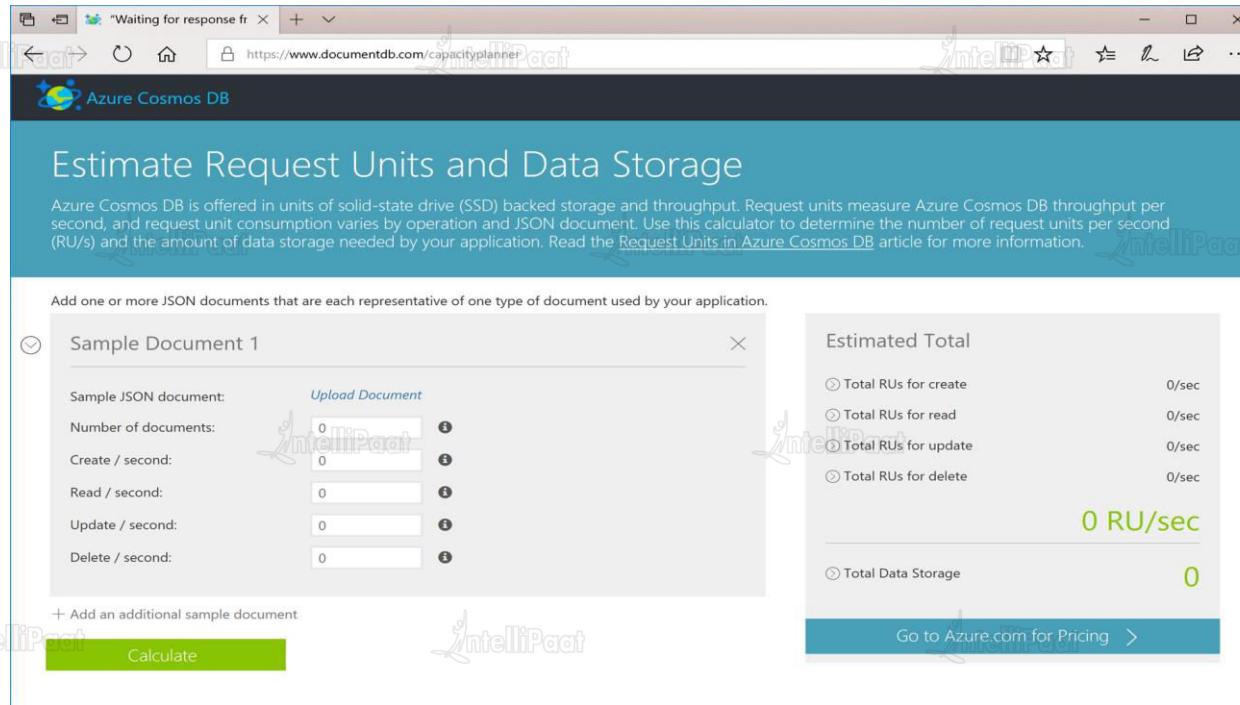
<input type="radio"/> Total RUs for create	0/sec
<input type="radio"/> Total RUs for read	0/sec
<input checked="" type="radio"/> Total RUs for update	0/sec
<input type="radio"/> Total RUs for delete	0/sec

0 RU/sec

Total Data Storage **0**

[Go to Azure.com for Pricing >](#)

Calculate



Request Unit Calculator



The screenshot shows the Azure Cosmos DB Request Unit Calculator. The URL in the browser is <https://www.documentdb.com/capacityplanner#>. The page title is "Estimate Request Units and Data Storage".

On the left, there is a sample document configuration section for "SmithFamily.json". It includes fields for "Number of documents" (1400000), "Create / second" (200), "Read / second" (1000), "Update / second" (0), and "Delete / second" (0).

On the right, the "Estimated Total" section displays the following values:

- Total RUs for create: 0/sec
- Total RUs for read: 0/sec
- Total RUs for update: 0/sec**
- Total RUs for delete: 0/sec

0 RU/sec

Total Data Storage: 0

[Go to Azure.com for Pricing >](#)

Calculate

Request Unit Calculator



The screenshot shows a Microsoft Edge browser window displaying the Azure Cosmos DB Capacity Planner calculator at <https://www.documentdb.com/capacityplanner#>. The page title is "Estimate Request Units and Data Storage".

On the left, there is a sample document configuration section for "SmithFamily.json" with 1,400,000 documents. It includes fields for Create / second (200), Read / second (1000), Update / second (0), and Delete / second (0). A "Calculate" button is present.

On the right, the "Estimated Total" section displays the following results:

Category	Value
Total RU/s for create	1790/sec
Total RU/s for read	1000/sec
Total RU/s for update	0/sec
Total RU/s for delete	0/sec
Total RU/s	2790 RUs/sec
Total Data Storage	0.45 GB

A link to "Go to Azure.com for Pricing" is also visible.

Pricing



SSD Storage

	A	B	C
1	1 GB	10 GB	
2	\$ 0.25	\$ 2.50	/month

Throughput

	A	B	C
1	100 RU/s	400 RU/s	
2	\$ 0.008	\$ 0.032	/hour
3	\$ 0.192	\$ 0.768	/day
4	\$ 5.856	\$ 23.424	/month



Achieving Elastic Scale

Achieving Elastic Scale



What Is Partitioning?

- Massive scale-out within a container

Unlimited Containers

- Logical resource composed of multiple partitions

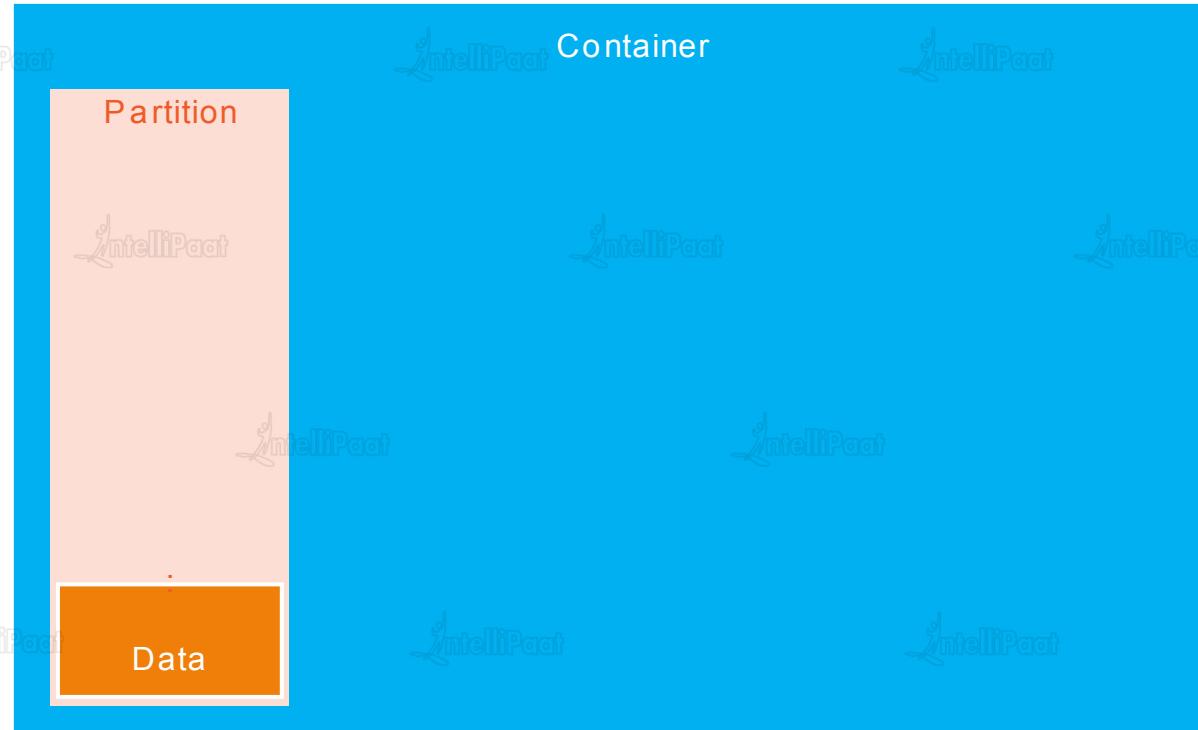
Partitions

- Physical fixed-capacity data buckets

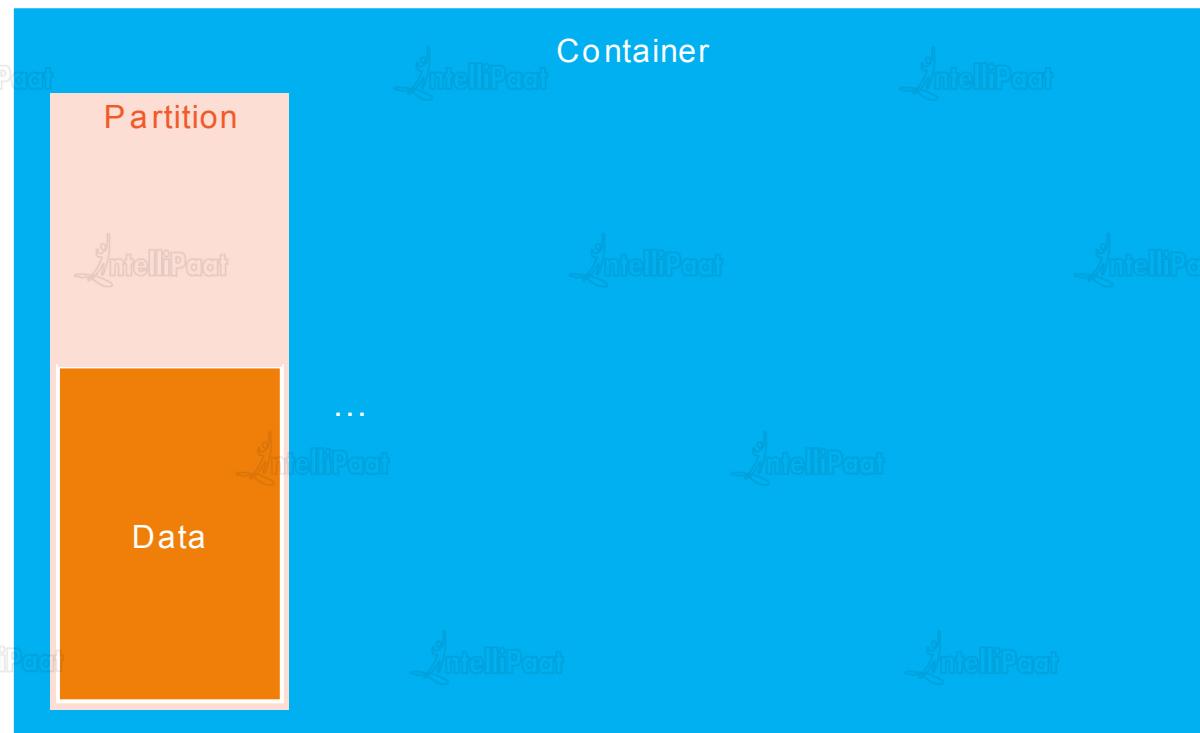
Automated Scale-out

- Cosmos DB transparently splits partitions to manage growth

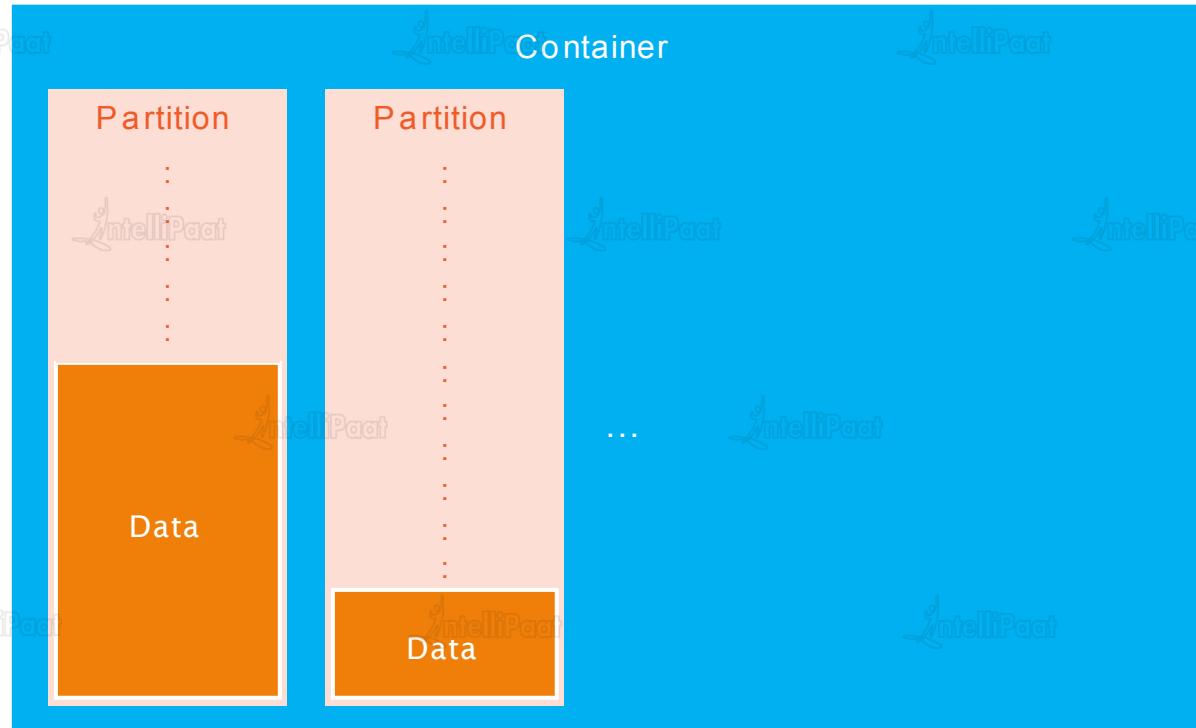
Achieving Elastic Scale



Achieving Elastic Scale



Achieving Elastic Scale





Partition Key

Selecting a Partition Key



Choosing the best partition key

- The right choice will deliver massive scale

Partition key values are hashed

- Hashed value determines the physical partition for storing each item

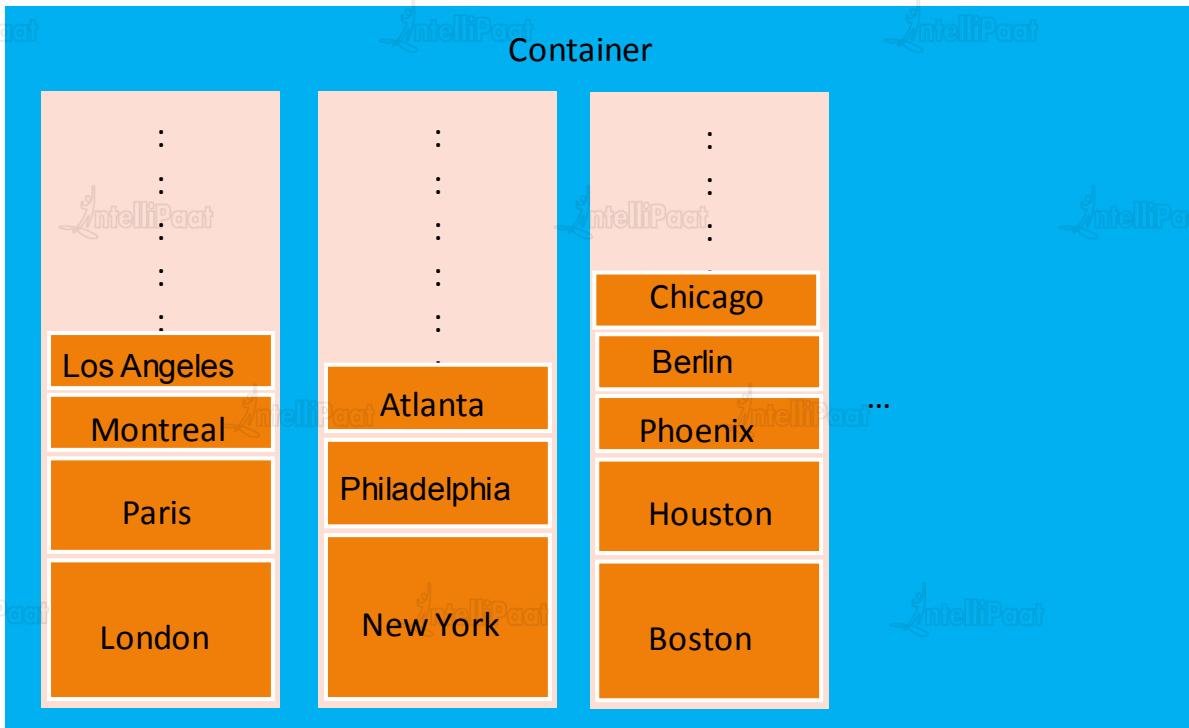
Partitions host multiple partition keys

- Items with the same partition key value are physically stored together on the same partition

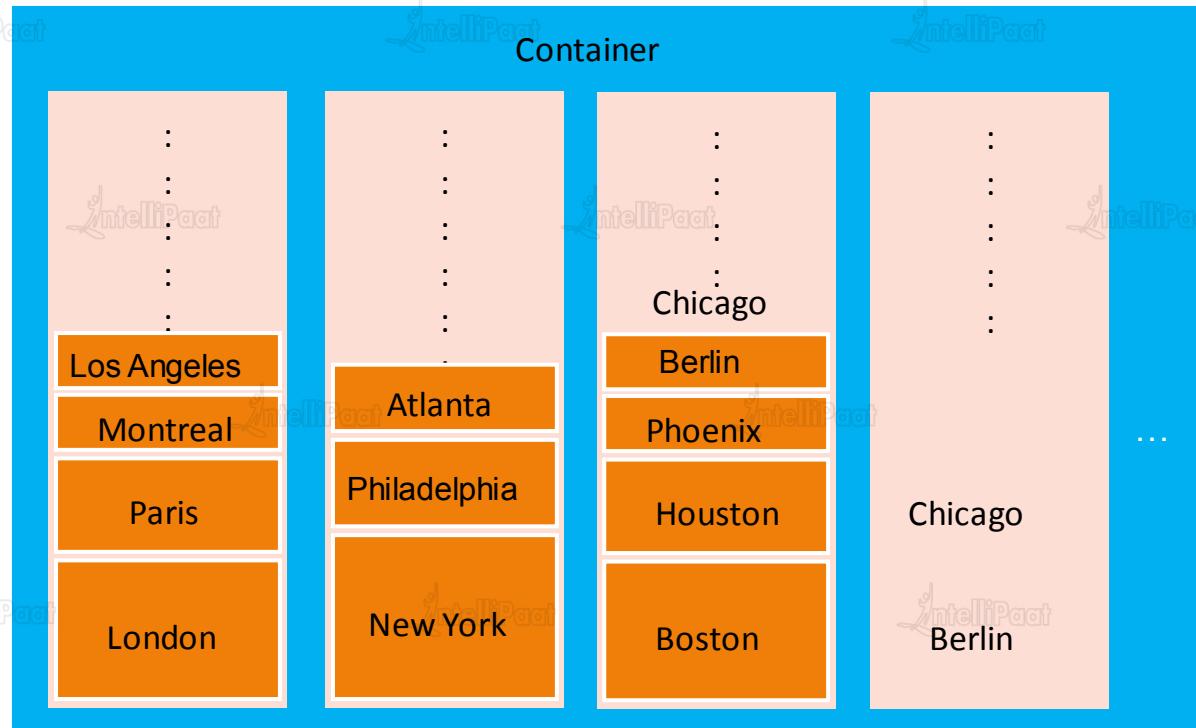
Two primary considerations

- Boundary for query and transactions
- No storage or performance bottlenecks

Selecting a Partition Key



Selecting a Partition Key



Choosing the Right Partition Key



Driven by data access patterns

Choose a property that groups commonly queried/updated items together

- User Profile Data
- User ID

- IoT
(e.g., device state)
Device ID

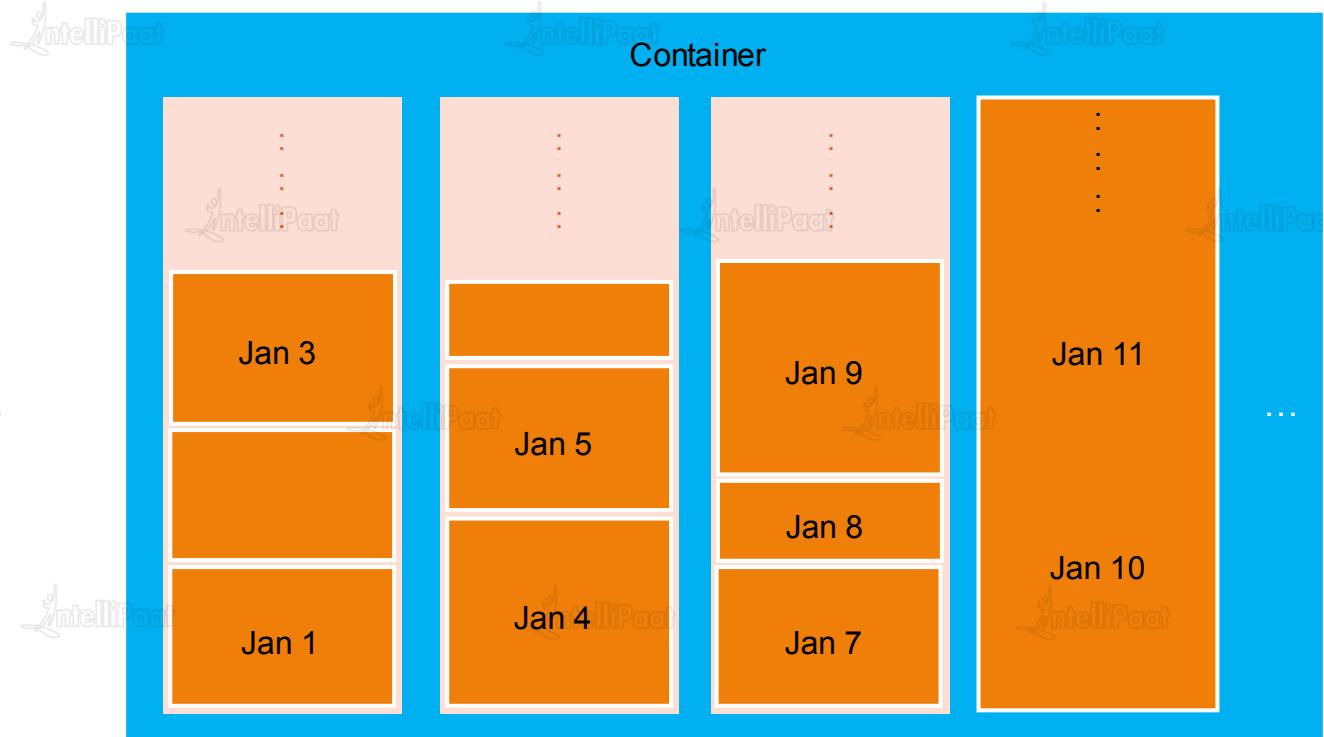
- Multi-tenant
Architecture
- Tenant ID

Choosing the Right Partition Key

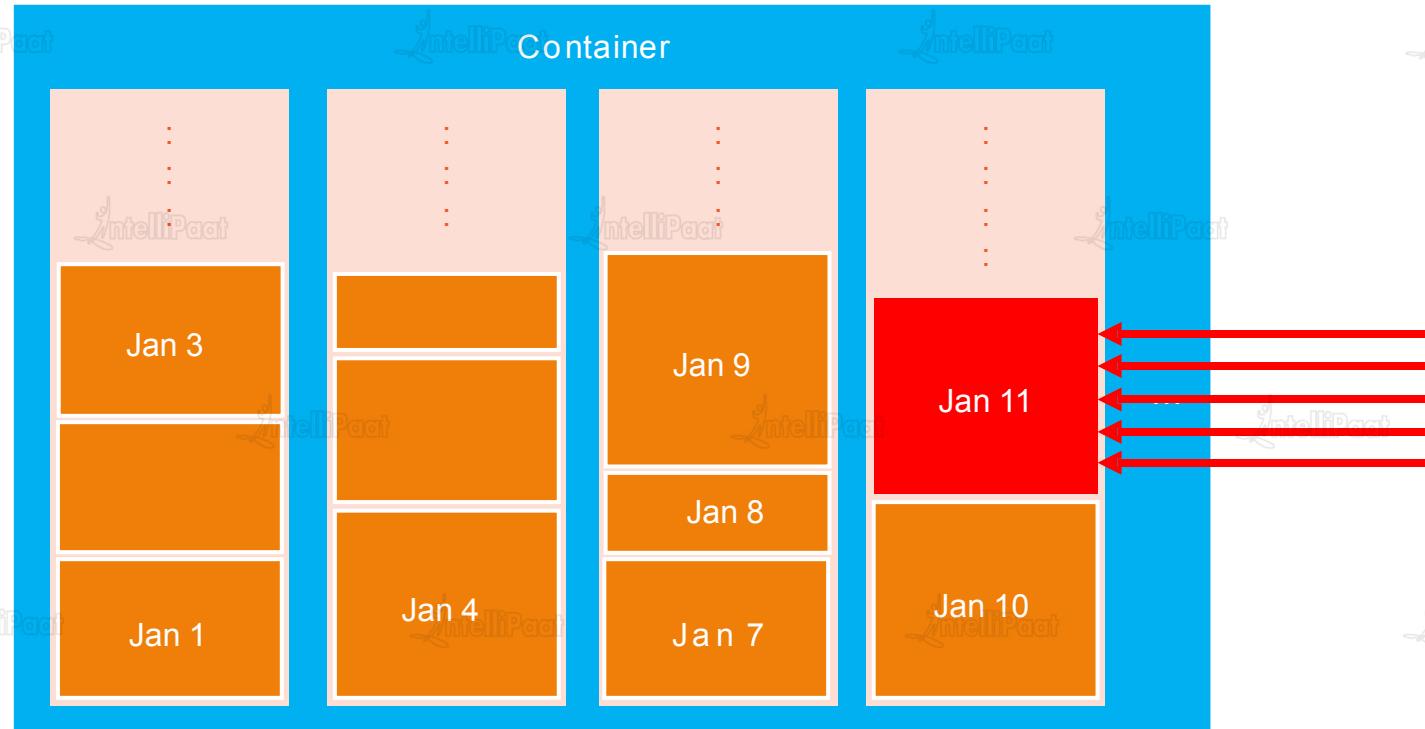


- ★ Generally, writes should be distributed uniformly across partitions
- ★ For example, user profile data with a user ID and creation date
 - ★ Partitioning by creation date
 - ★ Bad idea! All writes of the day are directed to the same partition

Choosing the Right Partition Key



Choosing the Right Partition Key

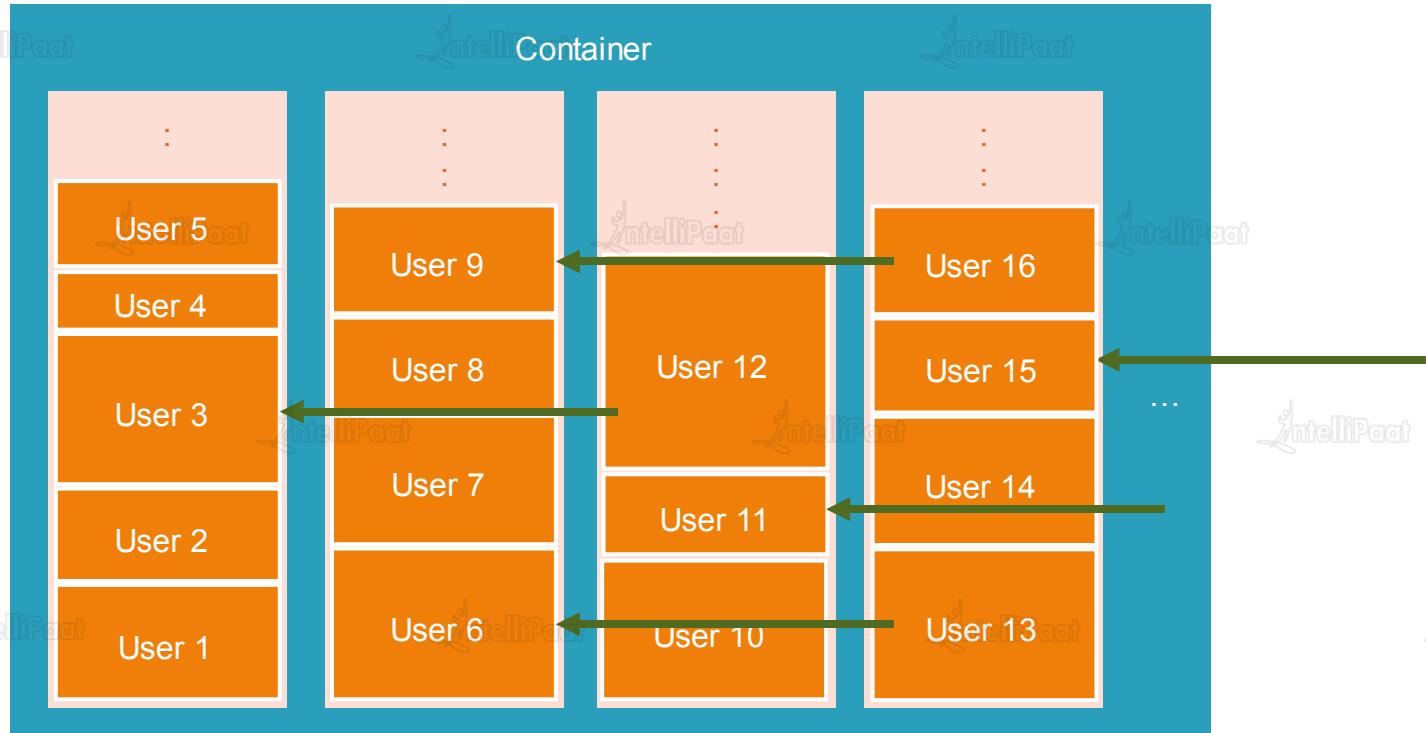


Choosing the Right Partition Key



- ★ Generally, writes should be distributed uniformly across partitions
- ★ For example, user profile data with a user ID and creation date
 - ★ Partitioning by creation date
 - ★ Bad idea! All writes of the day are directed to the same partition
 - ★ Partitioning by user ID
 - ★ Much better! Writes are directed to different partitions per user

Choosing the Right Partition Key

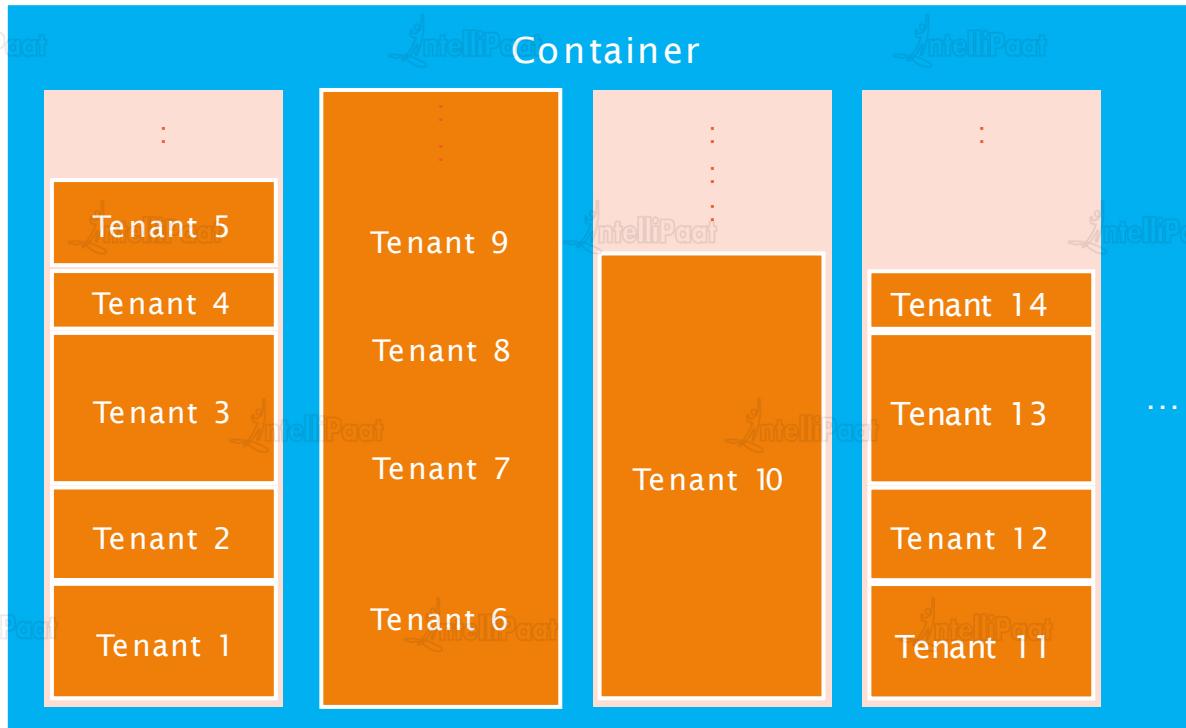


Choosing the Right Partition Key

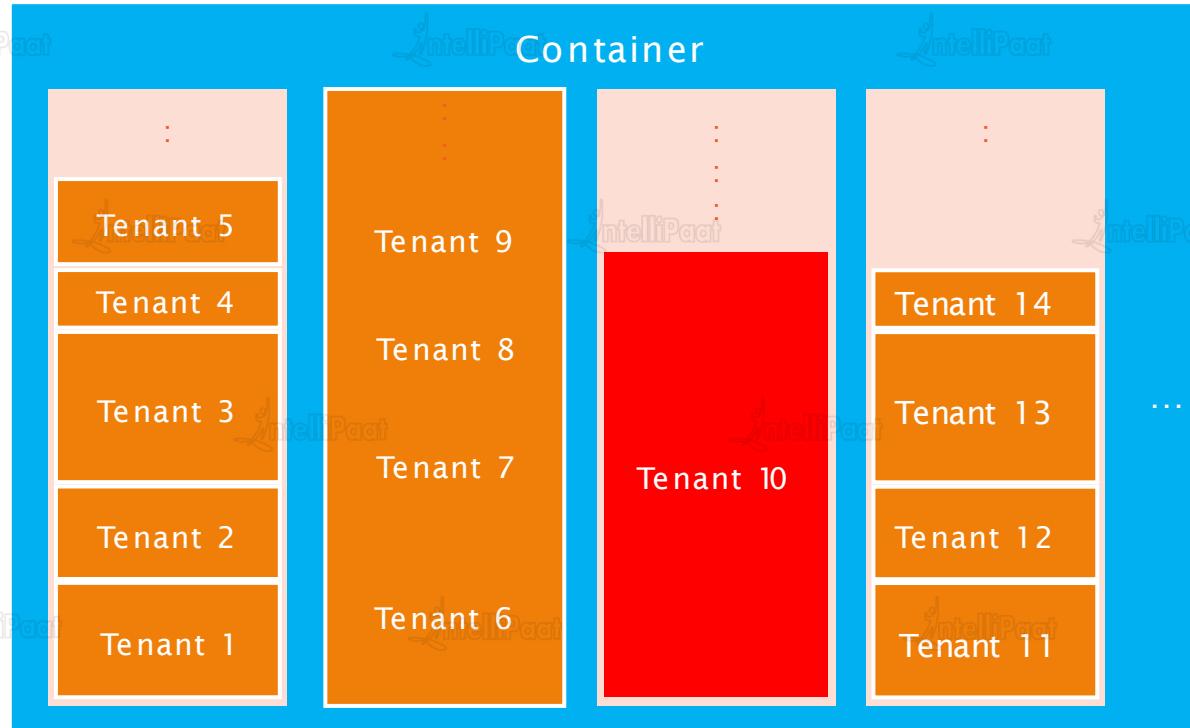


- ★ Generally, writes should be distributed uniformly across partitions
- ★ For example, user profile data with a user ID and creation date
 - ★ Partitioning by creation date
 - ★ Bad idea! All writes of the day are directed to the same partition
 - ★ Partitioning by user ID
 - ★ Much better! Writes are directed to different partitions per user
- ★ Create multiple containers for varying throughput needs
 - ★ Throughput is purchased at the container level

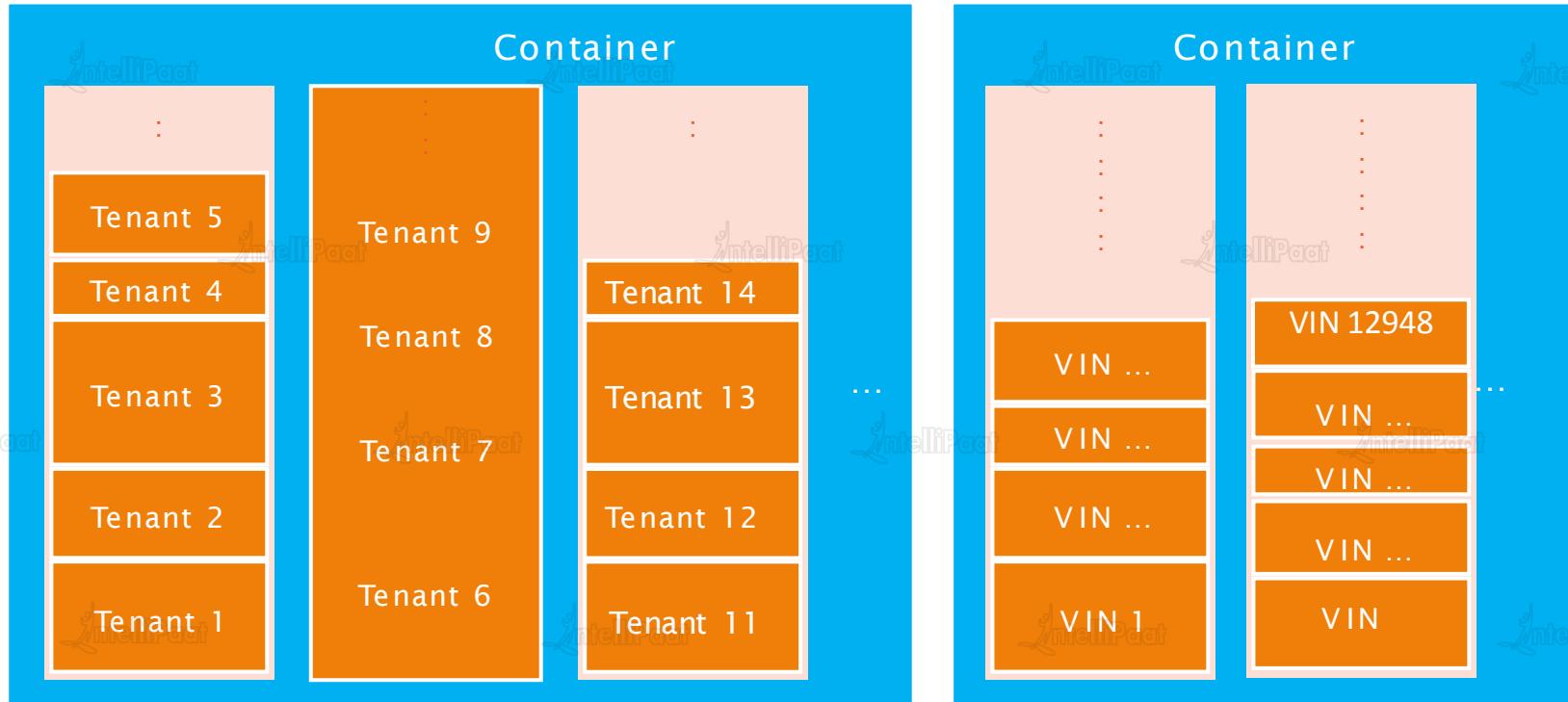
Choosing the Right Partition Key



Choosing the Right Partition Key



Choosing the Right Partition Key





Cross Partition Queries

Cross Partition Queries



Stored Procedures

- Always scoped to a single partition key

Queries

- Typically scoped to a single partition key

Cross-Partition Queries

- Span multiple partition keys
- Fan-out execution

Cross Partition Queries



```
using (var client = new DocumentClient(new Uri(endpoint), masterKey))
{
    var db = new Database { Id = "families" };
    await client.CreateDatabaseAsync(db);

    var coll = new DocumentCollection { Id = "families" };
    coll.PartitionKey.Paths.Add("/address/zipCode");

    await client.CreateDocumentCollectionAsync(dbUri, coll, new RequestOptions { OfferThroughput = 20000 });

    var sql = "SELECT * FROM c WHERE c.address.zipCode = '60603'";
    var query = client.CreateDocumentQuery<Document>(collUri, sql);
    var result = query.ToList();

    sql = "SELECT * FROM c WHERE c.address.city = 'Chicago'";
    query = client.CreateDocumentQuery<Document>(collUri, sql);
    try
    {
        result = query.ToList();
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }

    sql = "SELECT * FROM c WHERE c.address.city = 'Chicago'";
    var options = new FeedOptions { EnableCrossPartitionQuery = true, MaxDegreeOfParallelism = -1 };
    query = client.CreateDocumentQuery<Document>(collUri, sql, options);
    result = query.ToList();

    await client.DeleteDatabaseAsync(dbUri);
}
```

Cross Partition Queries



```
using (var client = new DocumentClient(new Uri(endpoint), masterKey))
{
    var db = new Database { Id = "families" };
    await client.CreateDatabaseAsync(db);

    var coll = new DocumentCollection { Id = "families" };
    coll.PartitionKey.Paths.Add("/address/zipCode");

    await client.CreateDocumentCollectionAsync(dbUri, coll, new RequestOptions { OfferThroughput = 20000 });

    var sql = "SELECT * FROM c WHERE c.address.zipCode = '60603'";
    var query = client.CreateDocumentQuery<Document>(collUri, sql);
    var result = query.ToList(); ≤ 7ms elapsed

    sql = "SELECT * FROM c WHERE c.address.city = 'Chicago'";
    query = client.CreateDocumentQuery<Document>(collUri, sql);
    try
    {
        result = query.ToList();
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }

    sql = "SELECT * FROM c WHERE c.address.city = 'Chicago'";
    var options = new FeedOptions { EnableCrossPartitionQuery = true, MaxDegreeOfParallelism = -1 };
    query = client.CreateDocumentQuery<Document>(collUri, sql, options);
    result = query.ToList();

    await client.DeleteDatabaseAsync(dbUri);
}
```

Cross Partition Queries



```
using (var client = new DocumentClient(new Uri(endpoint), masterKey))
{
    var db = new Database { Id = "families" };
    await client.CreateDatabaseAsync(db);

    var coll = new DocumentCollection { Id = "families" };
    coll.PartitionKey.Paths.Add("/address/zipCode");

    await client.CreateDocumentCollectionAsync(dbUri, coll, new RequestOptions { OfferThroughput = 20000 });

    var sql = "SELECT * FROM c WHERE c.address.zipCode = '60603'";
    var query = client.CreateDocumentQuery<Document>(collUri, sql);
    var result = query.ToList();

    sql = "SELECT * FROM c WHERE c.address.city = 'Chicago'";
    query = client.CreateDocumentQuery<Document>(collUri, sql);
    try
    {
        result = query.ToList();
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }

    sql = "SELECT * FROM c WHERE c.address.city = 'Chicago'";
    var options = new FeedOptions { EnableCrossPartitionQuery = true, MaxDegreeOfParallelism = -1 };
    query = client.CreateDocumentQuery<Document>(collUri, sql, options);
    result = query.ToList();

    await client.DeleteDatabaseAsync(dbUri);
}
```

Cross Partition Queries



```
using (var client = new DocumentClient(new Uri(endpoint), masterKey))
{
    var db = new Database { Id = "families" };
    await client.CreateDatabaseAsync(db);

    var coll = new DocumentCollection { Id = "families" };
    coll.PartitionKey.Paths.Add("/address/zipCode");

    await client.CreateDocumentCollectionAsync(dbUri, coll, new RequestOptions { OfferThroughput = 20000 });

    var sql = "SELECT * FROM c WHERE c.address.zipCode = '60603'";
    var query = client.CreateDocumentQuery<Document>(collUri, sql);
    var result = query.ToList();

    sql = "SELECT * FROM c WHERE c.address.city = 'Chicago'";
    query = client.CreateDocumentQuery<Document>(collUri, sql);
    try
    {
        result = query.ToList();
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message); ≤ 78ms elapsed
    }

    sql = "SELECT * FROM c WHERE c.address.city = 'Chicago'";
    var options = new FeedOptions { EnableCrossPartitionQuery = true, MaxDegreeOfParallelism = -1 };
    query = client.CreateDocumentQuery<Document>(collUri, sql, options);
    result = query.ToList();

    await client.DeleteDatabaseAsync(dbUri);
}
```

Cross Partition Queries

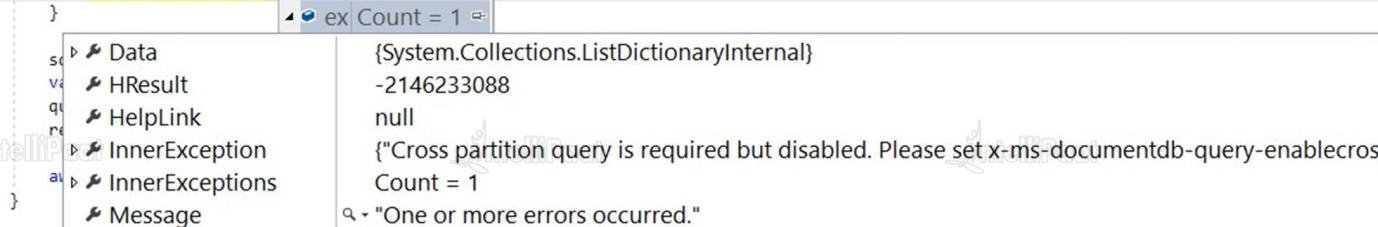
```
using (var client = new DocumentClient(new Uri(endpoint), masterKey))
{
    var db = new Database { Id = "families" };
    await client.CreateDatabaseAsync(db);

    var coll = new DocumentCollection { Id = "families" };
    coll.PartitionKey.Paths.Add("/address/zipCode");

    await client.CreateDocumentCollectionAsync(dbUri, coll, new RequestOptions { OfferThroughput = 20000 });

    var sql = "SELECT * FROM c WHERE c.address.zipCode = '60603'";
    var query = client.CreateDocumentQuery<Document>(collUri, sql);
    var result = query.ToList();

    sql = "SELECT * FROM c WHERE c.address.city = 'Chicago'";
    query = client.CreateDocumentQuery<Document>(collUri, sql);
    try
    {
        result = query.ToList();
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message); ≤ 78ms elapsed
    }
}
```



The screenshot shows a debugger window with an exception object named 'ex'. The 'Message' property is highlighted with a yellow background. A tooltip provides the message: {"Cross partition query is required but disabled. Please set x-ms-documentdb-query-enablecrosspartition header."}. Other properties shown include 'Count = 1' and 'InnerExceptions'.

Cross Partition Queries

```
using (var client = new DocumentClient(new Uri(endpoint), masterKey))
{
    var db = new Database { Id = "families" };
    await client.CreateDatabaseAsync(db);

    var coll = new DocumentCollection { Id = "families" };
    coll.PartitionKey.Paths.Add("/address/zipCode");

    await client.CreateDocumentCollectionAsync(dbUri, coll, new RequestOptions { OfferThroughput = 20000 });

    var sql = "SELECT * FROM c WHERE c.address.zipCode = '60603'";
    var query = client.CreateDocumentQuery<Document>(collUri, sql);
    var result = query.ToList();

    sql = "SELECT * FROM c WHERE c.address.city = 'Chicago'";
    query = client.CreateDocumentQuery<Document>(collUri, sql);
    try
    {
        result = query.ToList();
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);  ≤ 78ms elapsed
    }

    sql = "SELECT * FROM c WHERE c.address.city = 'Chicago'";
    var options = new FeedOptions { EnableCrossPartitionQuery = true, MaxDegreeOfParallelism = -1 };
    query = client.CreateDocumentQuery<Document>(collUri, sql, options);
    result = query.ToList();

    await client.DeleteDatabaseAsync(dbUri);
}
```

Cross Partition Queries



```
using (var client = new DocumentClient(new Uri(endpoint), masterKey))
{
    var db = new Database { Id = "families" };
    await client.CreateDatabaseAsync(db);

    var coll = new DocumentCollection { Id = "families" };
    coll.PartitionKey.Paths.Add("/address/zipCode");

    await client.CreateDocumentCollectionAsync(dbUri, coll, new RequestOptions { OfferThroughput = 20000 });

    var sql = "SELECT * FROM c WHERE c.address.zipCode = '60603'";
    var query = client.CreateDocumentQuery<Document>(collUri, sql);
    var result = query.ToList();

    sql = "SELECT * FROM c WHERE c.address.city = 'Chicago'";
    query = client.CreateDocumentQuery<Document>(collUri, sql);
    try
    {
        result = query.ToList();
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }

    sql = "SELECT * FROM c WHERE c.address.city = 'Chicago'";
    var options = new FeedOptions { EnableCrossPartitionQuery = true, MaxDegreeOfParallelism = -1 };
    query = client.CreateDocumentQuery<Document>(collUri, sql, options);
    result = query.ToList(); ≤3ms elapsed

    await client.DeleteDatabaseAsync(dbUri);
}
```

Cross Partition Queries



```
using (var client = new DocumentClient(new Uri(endpoint), masterKey))
{
    var db = new Database { Id = "families" };
    await client.CreateDatabaseAsync(db);

    var coll = new DocumentCollection { Id = "families" };
    coll.PartitionKey.Paths.Add("/address/zipCode");

    await client.CreateDocumentCollectionAsync(dbUri, coll, new RequestOptions { OfferThroughput = 20000 });

    var sql = "SELECT * FROM c WHERE c.address.zipCode = '60603'";
    var query = client.CreateDocumentQuery<Document>(collUri, sql);
    var result = query.ToList();

    sql = "SELECT * FROM c WHERE c.address.city = 'Chicago'";
    query = client.CreateDocumentQuery<Document>(collUri, sql);
    try
    {
        result = query.ToList();
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }

    sql = "SELECT * FROM c WHERE c.address.city = 'Chicago'";
    var options = new FeedOptions { EnableCrossPartitionQuery = true, MaxDegreeOfParallelism = -1 };
    query = client.CreateDocumentQuery<Document>(collUri, sql, options);
    result = query.ToList();

    await client.DeleteDatabaseAsync(dbUri); ≤ 936ms elapsed
}
```



Replication-Why



Replication-Why?



- Performance**
- Within a region, ensures SLA on RUs purchased
 - Across regions, brings data closer to the consumer

- Business continuity**
- In the event of major failure or natural disaster



Turnkey Global Distribution

Turnkey Global Distribution



Turnkey Global Distribution



- ★ Associate any number of regions with your Cosmos DB account
 - ★ Limited to geo-fencing policies
- ★ Dynamically add/remove regions
 - ★ Associate (and disassociate) regions with the click of a mouse
- ★ Failover priorities
 - ★ Choose preferred region, followed by regions for failover in order of preference

Replicate data globally andr1

Save Discard Manual Failover Failover Priorities

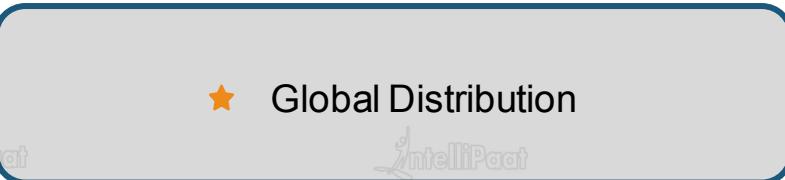
Click on a location to add or remove regions from your Azure Cosmos DB account.
* Each region is billable based on the throughput and storage for the account. Learn more

A world map with numerous blue diamond markers scattered across it, representing the locations of Azure Cosmos DB regions. The markers are concentrated in North America, Europe, and Asia, with smaller clusters in South America, Australia, and Africa.



Hands-On

Hands-On





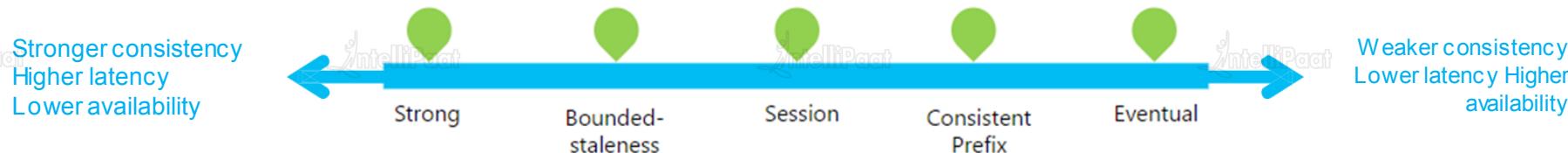
Replication and Consistency

Replication and Consistency



How do you ensure consistent reads across replicas?

Define a consistency level



Replication within a region

Data moves extremely fast (typically, within 1ms) between neighboring racks

Global replication

It takes hundreds of milliseconds to move data across continents



Consistency Levels

Five Consistency Levels



Strong

- No dirty reads

Session

- No dirty reads for writers
(read your own writes)
- Dirty reads possible for other users

Bounded Staleness

- Dirty reads possible
- Bounded by time and updates

Consistent Prefix

- Dirty reads possible
- Reads never see out-of-order writes

Eventual

- Stale reads possible
- No guaranteed order

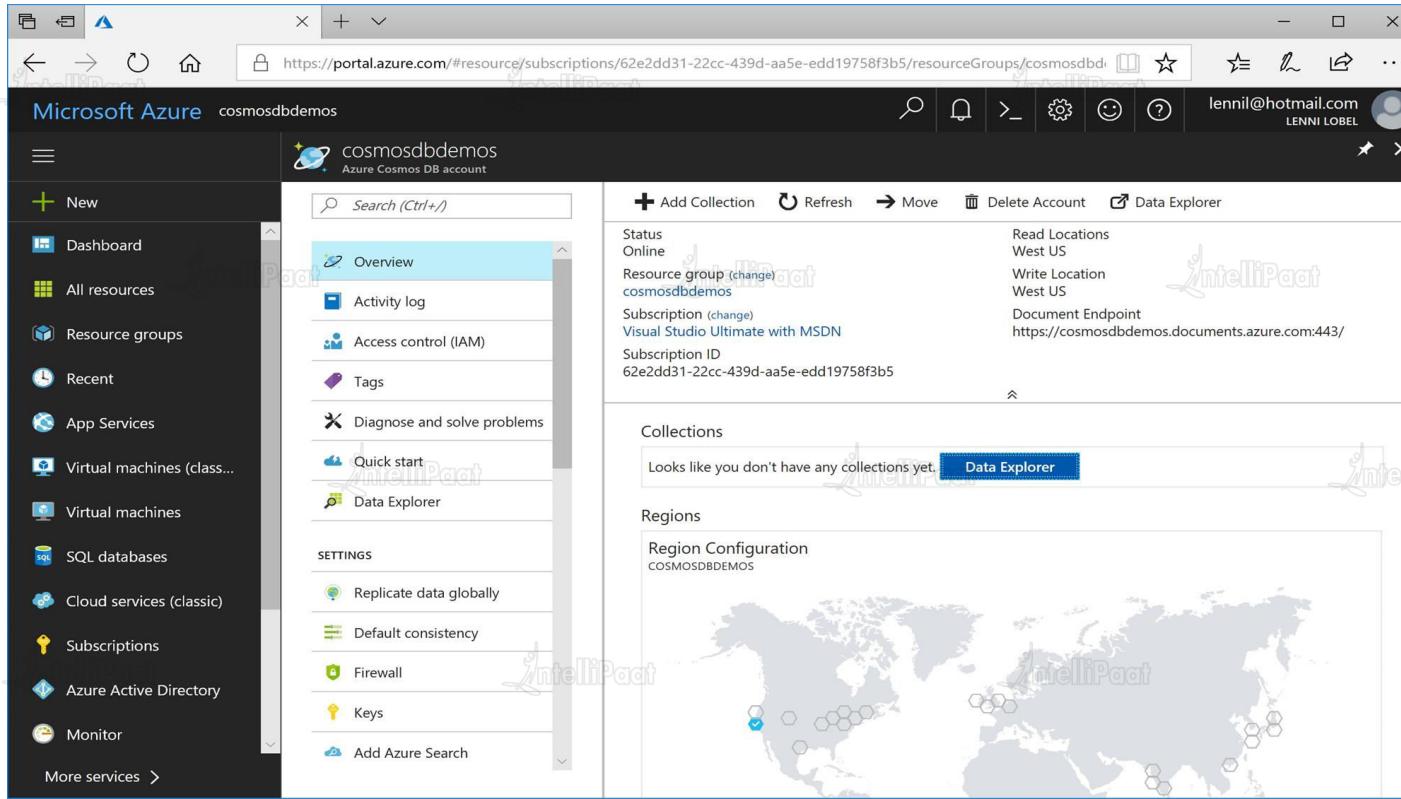
Setting the Consistency Level



Set default for entire account

Can be changed at any time

Setting the Consistency Level



The screenshot shows the Microsoft Azure portal interface for managing an Azure Cosmos DB account named "cosmosdbdemos".

Left Sidebar: Lists various Azure services and resources.

Top Bar: Shows the URL <https://portal.azure.com/#resource/subscriptions/62e2dd31-22cc-439d-aa5e-edd19758f3b5/resourceGroups/cosmosdbdemos/providers/Microsoft.DocumentDB/databaseAccounts/cosmosdbdemos>, the user "lennil@hotmail.com LENNI LOBEL", and navigation icons.

Account Overview: Displays the account name "cosmosdbdemos" and "Azure Cosmos DB account".

Actions: Includes "Add Collection", "Refresh", "Move", "Delete Account", and "Data Explorer".

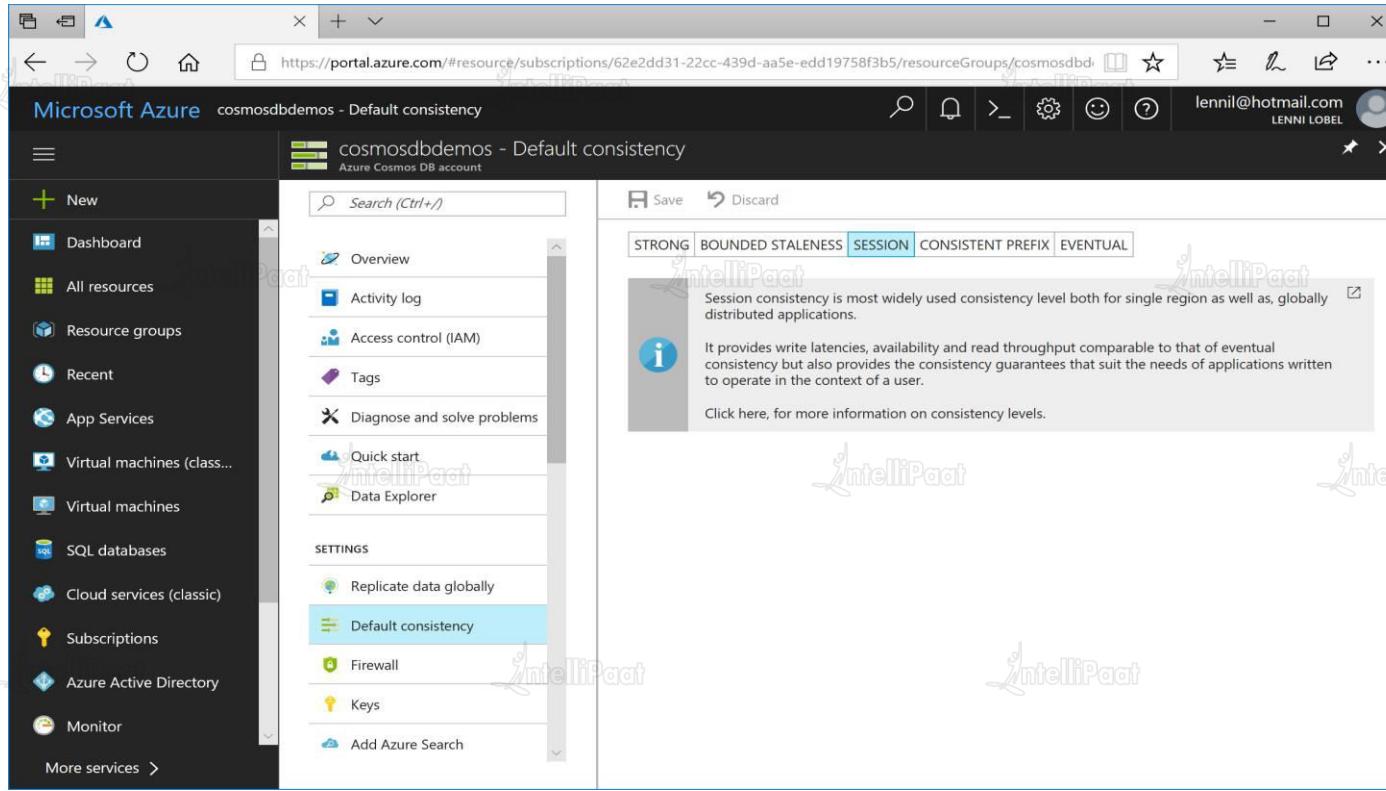
Status: Online. Resource group: "cosmosdbdemos". Subscription: "Visual Studio Ultimate with MSDN". Subscription ID: "62e2dd31-22cc-439d-aa5e-edd19758f3b5".

Read Locations: West US. Write Location: West US. Document Endpoint: <https://cosmosdbdemos.documents.azure.com:443/>.

Collections: A message states "Looks like you don't have any collections yet." with a "Data Explorer" button.

Regions: A world map showing region configuration for "COSMOSDBDEMO".

Setting the Consistency Level



Microsoft Azure cosmosdbdemos - Default consistency

cosmosdbdemos - Default consistency

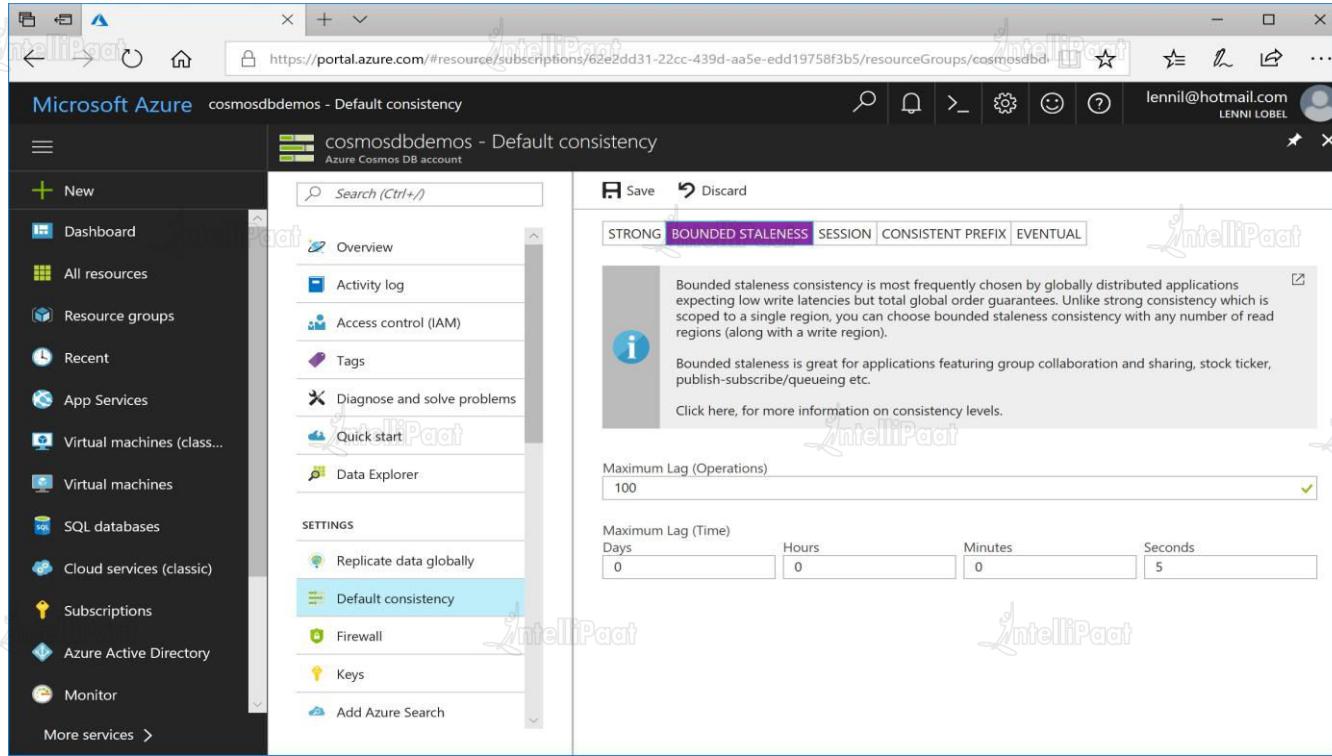
STRONG BOUNDED STALENESS SESSION CONSISTENT PREFIX EVENTUAL

Session consistency is most widely used consistency level both for single region as well as, globally distributed applications.

It provides write latencies, availability and read throughput comparable to that of eventual consistency but also provides the consistency guarantees that suit the needs of applications written to operate in the context of a user.

Click here, for more information on consistency levels.

Setting the Consistency Level



The screenshot shows the Microsoft Azure portal interface for managing an Azure Cosmos DB account named "cosmosdbdemos". The left sidebar lists various services like Dashboard, All resources, Resource groups, Recent, App Services, Virtual machines, SQL databases, Cloud services, Subscriptions, Azure Active Directory, Monitor, and More services. The main content area is titled "cosmosdbdemos - Default consistency" and displays the "Default consistency" settings. The "STRONG" tab is selected, followed by "BOUNDED STALENESS", "SESSION", "CONSISTENT PREFIX", and "EVENTUAL". A detailed description of Bounded Staleness is provided, stating it's great for applications like group collaboration, stock tickers, and publish-subscribe/queueing. Below this, there are input fields for "Maximum Lag (Operations)" set to 100 and "Maximum Lag (Time)" set to 0 days, 0 hours, 0 minutes, and 5 seconds.

Setting the Consistency Level

Set default for entire account

- Can be changed at any time

Override at the request level

- Any request can weaken the default consistency level

```
new DocumentClient(new Uri(endpoint), masterKey, connectionPolicy, ConsistencyLevel.)
```

- BoundedStaleness
- ConsistentPrefix
- Eventual
- Session
- Strong



Document Database



What Is a Document Database?



Object graph
Semi-structured JSON
(JavaScript Object
Notation) documents

```
{  
    "id": 10,  
    "name": "Clementine's",  
    "username": "McDonalds",  
    "email": "Rey.McDonalds@McDonalds.com",  
    "open": true,  
    "hours": {},  
    "categories": [  
        "Burgers",  
        "Fast Food",  
        "Restaurants"  
    ],  
    "city": "Homestead",  
    "review_count": 5,  
    "name": "McDonald's",  
    "neighborhoods": [  
        "Homestead"  
    ],  
    "longitude": -79.91008,  
    "state": "PA",  
    "stars": 2,  
    "latitude": 40.412086,  
    "attributes": {  
        "Take-out": true,  
        "Wi-Fi": "free",  
        "Drive-Thru": true,  
        "Good For": {  
            "dessert": false,  
            "latenight": false,  
            "lunch": false,  
            "dinner": false,  
            "breakfast": false,  
            "brunch": false  
        },  
        "Caters": false,  
        "Noise Level": "average",  
        "Takes Reservations": false,  
        "Delivery": false  
    }  
},  
{  
    "id": "124216000000072038",  
    "description": "3",  
    "website": "3",  
    "numberOfEmployees": "3",  
    "phone": "3",  
    "name": "account3",  
    "shippingAddress": {  
        "country": "3",  
        "stateOrProvince": "3",  
        "city": "3",  
        "postalCode": "3",  
        "street1": "3"  
    },  
    "billingAddress": {  
        "country": "3",  
        "stateOrProvince": "3",  
        "city": "3",  
        "postalCode": "3",  
        "street1": "3"  
    },  
    "train": {  
        "date": "07/04/2016",  
        "time": "09:30",  
        "from": "New York",  
        "to": "Chicago",  
        "seat": "57B"  
    },  
    "passenger": {  
        "name": "John Smith"  
    },  
    "price": 1234.25,  
    "comments": ["Lunch & dinner incl.", "\"Have a nice day!\""]  
}
```

What Is a Document Database?



Object graph
Semi-structured JSON (JavaScript Object
Notation) documents

Two API choices

1. SQL API(fka DocumentDB API)
2. MongoDB API



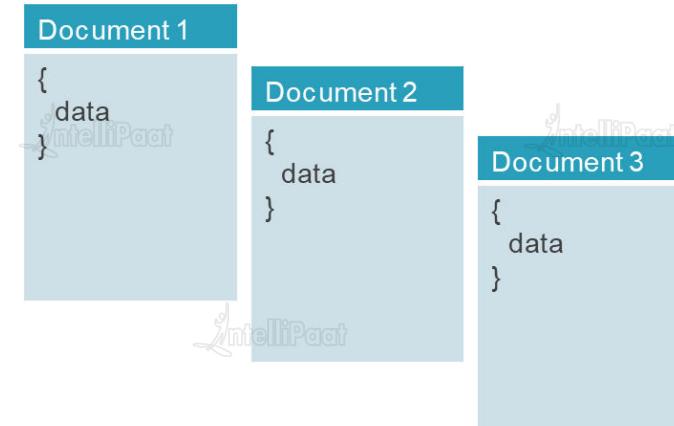
Data Modeling: Relational vs. Document

Data Modeling: Relational vs. Document



Relational Store	Document Store
Rows	Documents

Table
Row 1
Data
Row 2
Data
Row 3
Data
...



Data Modeling: Relational vs. Document



Relational Store	Document Store
Rows	Documents
Columns	Properties

Col1	Col2	Col3	Col4
Data	Data	Data	Data
Data	Data	Data	Data
Data	Data	Data	Data

Document 1

```
{  
  "prop1": data,  
  "prop2": data,  
  "prop3": data,  
  "prop4": data  
}
```

Document 2

```
{  
  "prop1": data,  
  "prop2": data,  
  "prop3": data,  
  "prop4": data  
}
```

Document 3

```
{  
  "prop1": data,  
  "prop2": data,  
  "prop3": data,  
  "prop4": data  
}
```

Data Modeling: Relational vs. Document



Relational Store			
Rows		Columns	
		Strongly-typed schemas	

Document Store			
Documents		Properties	
		Schema-free	

ID	Name	IsActive	Dob
1	John Smith	True	8/30/1964
2	Sarah Jones	False	2/18/2002
3	Adam Stark	True	7/13/1987

Document 1

```
{  
  "id": 1,  
  "name": "John Smith",  
  "isActive": true, "dob":  
  "1964-08-30"  
}
```

Document 2

```
{  
  "id": 2,  
  "fullName": "Sarah J ones",  
  "dob": "2002-02-18"  
}
```

Document 3

```
{  
  "id": 3,  
  "fullName":  
  {  
    "first": "Adam",  
    "last": "Stark"  
  },  
  "isActive": true,  
  "dob": "2015-04-19"  
}
```

Data Modeling: Relational vs. Document



Relational Store	Document Store
Rows	Documents
Columns	Properties
Strongly-typed schemas	Schema-free
Highly normalized	Typically denormalized

The diagram illustrates a relationship between two tables: a User Table and a Holdings Table. A green bracket on the left side of the slide groups the User Table and the Holdings Table. An arrow points from the User Table to the Holdings Table, indicating a one-to-many relationship where multiple holdings are associated with a single user.

User Table		
UserID	Name	Dob
1	John Smith	8/30/1964

Holdings Table			
StockID	UserID	Qty	Symbol
1	1	100	MSFT
2	1	75	WMT

Document

```
{"id": 1,  
 "name": "John Smith",  
 "dob": "1964-30-08",  
 "holdings": [  
     { "qty": 100, "symbol": "MSFT" },  
     { "qty": 75, "symbol": "WMT" }  
 ]}
```

Data Modeling: Relational vs. Document



Relational Store	Document Store
Rows	Documents
Columns	Properties
Strongly-typed schemas	Schema-free
Highly normalized	Typically denormalized

Document

```
{  
  "postid": "1",  
  "title": "My blog post",  
  "body": "Post content...",  
  "comments": [  
    "comment #1",  
    "comment #2",  
    "comment #3",  
    "comment #4",  
    :  
    "comment #1598873",  
    :  
  ]  
}
```



Data Modeling: Relational vs. Document



Relational Store	Document Store
Rows	Documents
Columns	Properties
Strongly-typed schemas	Schema-free
Highly normalized	Typically denormalized

Document

```
{  
  "postid": "1",  
  "title": "My blog post",  
  "body": "Post content..."  
}
```

Document

```
{  
  "postid": "1",  
  "comment": "comment #1"  
}  
  "postid": "1",  
  "comment": "comment #3"  
}
```

Data Modeling: Relational vs. Document



Relational Store	Document Store
Rows	Documents
Columns	Properties
Strongly-typed schemas	Schema-free
Highly normalized	Typically denormalized

Document

```
{  
  "postid": "1",  
  "title": "My blog post",  
  "body": "Post content...",  
  "comments": [  
    "comment #1",  
    "comment #2",  
    :  
    "comment #100"  
  ]  
}
```

Document

```
"postid": "1",  
"comments": [  
  "comment #301",  
  "comment #302",  
  :  
  "comment #400"  
]
```

Data Modeling: Relational vs. Document



Relational Store	Document Store
Rows	Documents
Columns	Properties
Strongly-typed schemas	Schema-free
Highly normalized	Typically denormalized

Document

```
{  
    "itemid": "1",  
    "part": "1",  
    "prop1": "Typically read",  
    "prop2": "Typically read",  
    "prop3": "Typically read",  
    "prop4": "Typically read",  
    :  
    "prop200": "Typically read",  
}
```

Document

```
{  
    "itemid": "1",  
    "part": "2",  
    "prop1": "Typically updated",  
    "prop2": "Typically updated",  
    :  
    "prop10": "Typically updated",  
}
```

Data Modeling: Relational vs. Document



Relational Store	Document Store
Rows	Documents
Columns	Properties
Strongly-typed schemas	Schema-free
Highly normalized	Typically denormalized

Document

```
{  
  "name": "John Smith",  
  "orderDate": "2015-30-03",  
  "details": [  
    { "qty": 5, "code": "BC", "desc": "Black chair" },  
    { "qty": 1, "code": "RT", "desc": "Red table" },  
    { "qty": 2, "code": "YC", "desc": "Yellow clock" }  
  ]  
}
```

Document

```
{  
  "name": "Sarah J ones,  
  "orderDate": "2015-16-04",  
  "details": [  
    { "qty": 1, "code": "PL", "desc": "Purple lamp" },  
    { "qty": 3, "code": "YC", "desc": "Yellow clock" }  
  ]  
}
```

Document

```
{  
  "name": "Adam Stark",  
  "orderDate": "2015-06-05",  
  "details": [  
    { "qty": 2, "code": "BC", "desc": "Black chair" },  
    { "qty": 1, "code": "YC", "desc": "Yellow clock" }  
  ]  
}
```

Data Modeling: Relational vs. Document



Relational Store	Document Store
Rows	Documents
Columns	Properties
Strongly-typed schemas	Schema-free
Highly normalized	Typically denormalized

Document

```
{  
  "name": "John Smith",  
  "orderDate": "2015-03-30",  
  "details": [  
    { "qty": 5, "code": "BC" },  
    { "qty": 1, "code": "RT" },  
    { "qty": 2, "code": "YC" }  
  ]  
}
```

Document

```
{  
  "name": "Sarah J Jones",  
  "orderDate": "2015-06-04",  
  "details": [  
    { "qty": 1, "code": "PL" },  
    { "qty": 3, "code": "YC" }  
  ]  
}
```

Document

```
{  
  "name": "Adam Stark",  
  "orderDate": "2015-06-05",  
  "details": [  
    { "qty": 2, "code": "BC" },  
    { "qty": 1, "code": "YC" }  
  ]  
}
```

Document

```
{  
  "code": "BC", "desc":  
  "Black chair"  
}
```

Document

```
{  
  "code": "RT",  
  "desc": "Red table"  
}
```

Document

```
{  
  "code": "YC",  
  "desc": "Yellow clock"  
}
```

Data Modeling: Relational vs. Document



Relational Store	Document Store
Rows	Documents
Columns	Properties
Strongly-typed schemas	Schema-free
Highly normalized	Typically denormalized

Document

```
{  
  "id": "1",  
  "name": "John Smith",  
  "dob": "1964-30-08",  
  "holdings": [  
    { "qty": 100, "symbol": "MSFT" },  
    { "qty": 75, "symbol": "WMT" }  
  ]  
}
```



Cosmos DB Resource Model

Cosmos DB Resource Model



Azure Cosmos DB
account

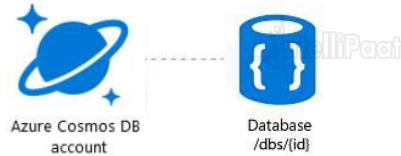


IntelliPaat

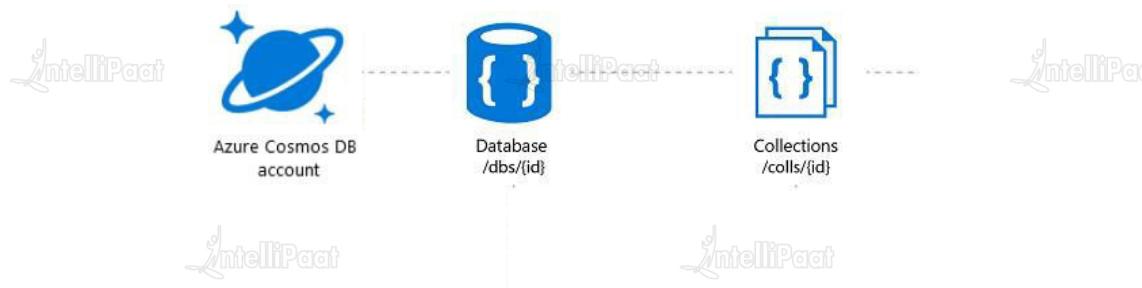
IntelliPaat



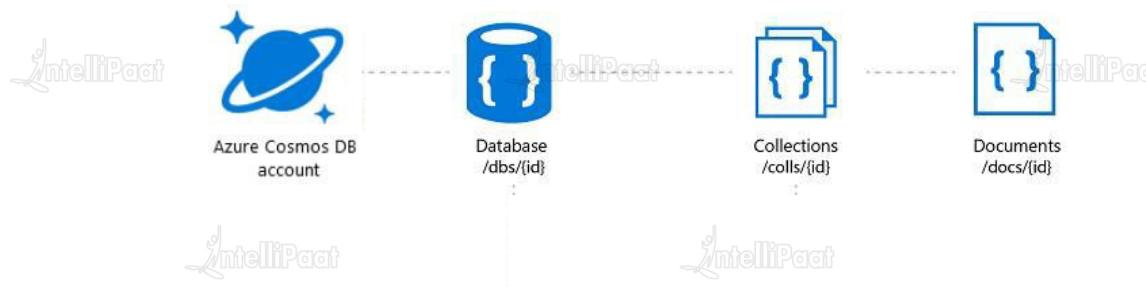
Cosmos DB Resource Model



Cosmos DB Resource Model



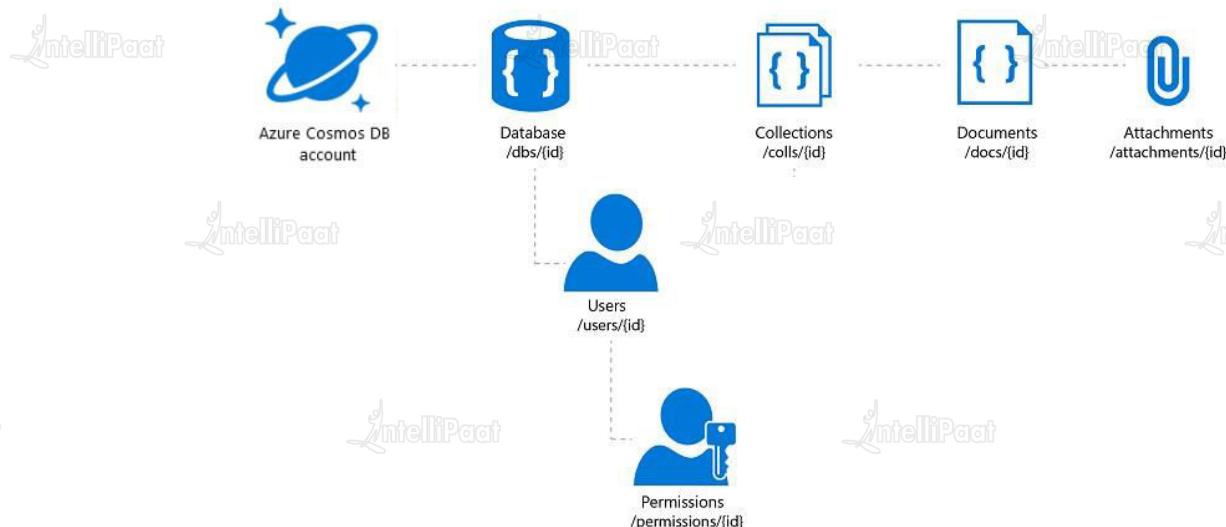
Cosmos DB Resource Model



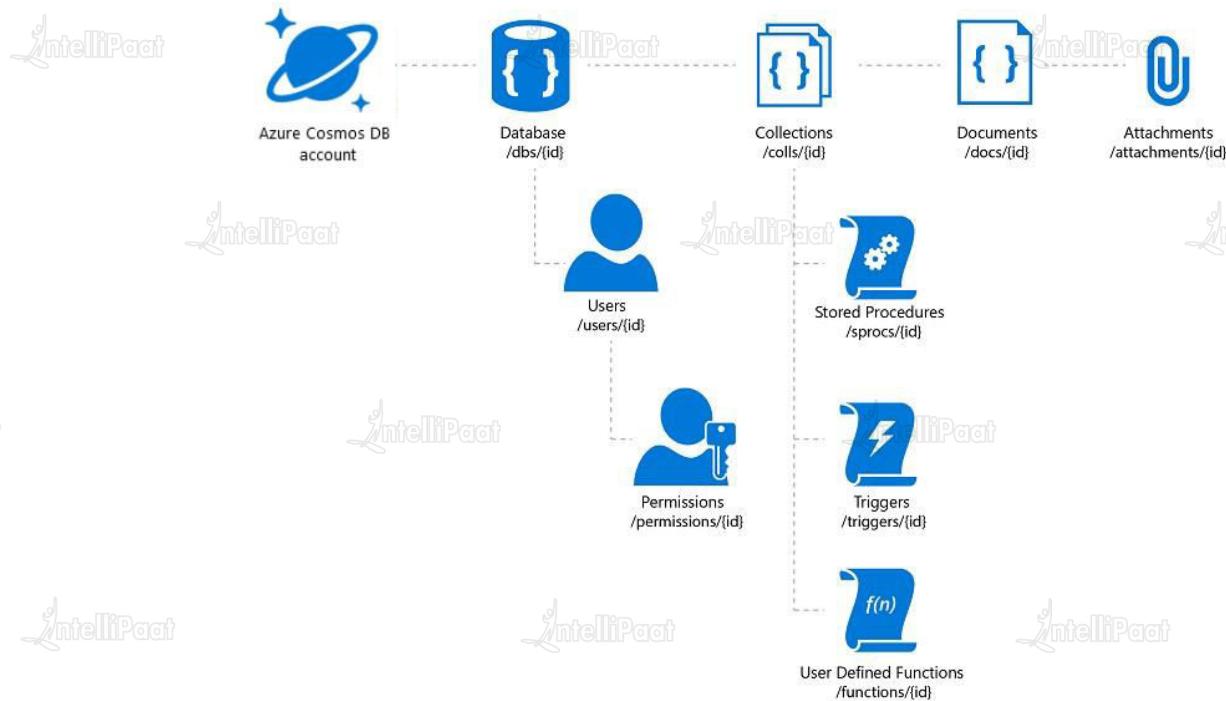
Cosmos DB Resource Model



Cosmos DB Resource Model



Cosmos DB Resource Model





Resource Properties, Self-Links, and URIs

Resource Properties, Self-Links, and URIs



Every resource contains these properties

Property	Value
id	User-defined unique ID (string)

Resource Properties, Self-Links, and URIs



Every resource contains these properties

Property	Value
id	User-defined unique ID (string)
_rid	Resource ID

Resource Properties, Self-Links, and URIs



Every resource contains these properties

Property	Value
<code>id</code>	User-defined unique ID (string)
<code>_rid</code>	Resource ID
<code>_ts</code>	Timestamp (last updated) epoch value

Resource Properties, Self-Links, and URIs



Every resource contains these properties

Property	Value
<code>id</code>	User-defined unique ID (string)
<code>_rid</code>	Resource ID
<code>_ts</code>	Timestamp (last updated) epoch value
<code>_etag</code>	GUID used for optimistic concurrency

Resource Properties, Self-Links, and URIs



Every resource contains these properties

Property	Value
<code>id</code>	User-defined unique ID (string)
<code>_rid</code>	Resource ID
<code>_ts</code>	Timestamp (last updated) epoch value
<code>_etag</code>	GUID used for optimistic concurrency
<code>_self</code>	URI path to the resource

Resource Properties, Self-Links, and URIs



Every resource contains these properties

Property	Value
<code>id</code>	User-defined unique ID (string)
<code>_rid</code>	Resource ID
<code>_ts</code>	Timestamp (last updated) epoch value
<code>_etag</code>	GUID used for optimistic concurrency
<code>_self</code>	URI path to the resource
<code>_attachments</code>	URI path suffix to the resource attachments

Resource Properties, Self-Links, and URIs



```
{  
    "familyName": "Smith",  
    "address": {  
        "addressLine": "123 Main Street",  
        "city": "Chicago",  
        "state": "IL",  
        "zipCode": "60601"  
    },  
    "parents": [  
        "Peter",  
        "Alice"  
    ],  
    "kids": [  
        "Adam",  
        "Jacqueline",  
        "Joshua"  
    ]  
}
```

Resource Properties, Self-Links, and URIs



```
{  
    "familyName": "Smith",  
    "address": {  
        "addressLine": "123 Main Street",  
        "city": "Chicago",  
        "state": "IL",  
        "zipCode": "60601"  
    },  
    "parents": [  
        "Peter",  
        "Alice"  
    ],  
    "kids": [  
        "Adam",  
        "Jacqueline",  
        "Joshua"  
    ],  
    "id": "ab7aaafc4-6a6e-f19e-bf5f-f7985b2f156d"  
}
```

Resource Properties, Self-Links, and URIs



```
{  
    "familyName": "Smith",  
    "address": {  
        "addressLine": "123 Main Street",  
        "city": "Chicago",  
        "state": "IL",  
        "zipCode": "60601"  
    },  
    "parents": [  
        "Peter",  
        "Alice"  
    ],  
    "kids": [  
        "Adam",  
        "Jacqueline",  
        "Joshua"  
    ],  
    "id": "ab7aafc4-6a6e-f19e-bf5f-f7985b2f156d",  
    "_rid": "v84pA07nPQABAAAAAAAABg==",  
    "_self": "dbs/v84pAA==/colls/v84pA07nPQA=/docs/v84pA07nPQABAAAAAAAABg==/",  
    "_etag": "\"00004700-0000-0000-0000-5a3e991a0000\"",  
    "_attachments": "attachments/",  
    "_ts": 1514051866  
}
```

```
var updatedAt = new DateTime(1970, 1, 1, 0, 0, 0, DateTimeKind.Utc)  
    .AddSeconds(1514051866)  
    .ToLocalTime()  
    .ToString("ddd M/d/yyyy h:m:s tt");
```

Resource Properties, Self-Links, and URIs



```
{  
  "familyName": "Smith",  
  "address": {  
    "addressLine": "123 Main Street",  
    "city": "Chicago",  
    "state": "IL",  
    "zipCode": "60601"  
  },  
  "parents": [  
    "Peter",  
    "Alice"  
  ],  
  "kids": [  
    "Adam",  
    "Jacqueline",  
    "Joshua"  
  ],  
  "id": "ab7aafc4-6a6e-f19e-bf5f-f7985b2f156d",  
  "_rid": "v84pA07nPQABAAAAAAAABg==",  
  "_self": "dbs/v84pAA=/colls/v84pA07nPQA=/docs/v84pA07nPQABAAAAAAAABg==/",  
  "_etag": "\"00004700-0000-0000-0000-5a3e991a0000\"",  
  "_attachments": "attachments/",  
  "_ts": 1514051866  
}
```

```
var updatedAt = new DateTime(1970, 1, 1, 0, 0, 0, DateTimeKind.Utc)  
  .A updatedAt  ToString("ddd M/d/yyyy h:m:s tt");  
  .ToLocalTime()  
  .ToString("ddd M/d/yyyy h:m:s tt");
```

Resource Properties, Self-Links, and URIs



```
{  
    "familyName": "Smith",  
    "address": {  
        "addressLine": "123 Main Street",  
        "city": "Chicago",  
        "state": "IL",  
        "zipCode": "60601"  
    },  
    "parents": [  
        "Peter",  
        "Alice"  
    ],  
    "kids": [  
        "Adam",  
        "Jacqueline",  
        "Joshua"  
    ],  
    "id": "ab7aafc4-6a6e-f19e-bf5f-f7985b2f156d",  
    "_rid": "v84pA07nPQABAAAAAAAABg==",  
    "_self": " dbs/v84pAA==/colls/v84pA07nPQA=/docs/v84pA07nPQABAAAAAAAABg==/",  
    "_etag": "\"00004700-0000-0000-0000-5a3e991a0000\"",  
    "_attachments": "attachments/",  
    "_ts": 1514051866  
}
```

```
string id1 = " dbs/v84pAA==/colls/v84pA07nPQA=/docs/v84pA07nPQABAAAAAAAABg==/";  
string id2 = " dbs/Families/colls/Families/docs/ab7aafc4-6a6e-f19e-bf5f-f7985b2f156d";  
Uri id3 = UriFactory.CreateDocumentUri("Families", "Families", "ab7aafc4-6a6e-f19e-bf5f-f7985b2f156d");
```

Resource Properties, Self-Links, and URIs



```
{  
    "familyName": "Smith",  
    "address": {  
        "addressLine": "123 Main Street",  
        "city": "Chicago",  
        "state": "IL",  
        "zipCode": "60601"  
    },  
    "parents": [  
        "Peter",  
        "Alice"  
    ],  
    "kids": [  
        "Adam",  
        "Jacqueline",  
        "Joshua"  
    ],  
    "id": "ab7aafc4-6a6e-f19e-bf5f-f7985b2f156d",  
    "_rid": "v84pAO7nPQABAAAAAAAABg==",  
    "_self": "dbs/v84pAA==/colls/v84pAO7nPQA=/docs/v84pAO7nPQABAAAAAAAABg==/",  
    "_etag": "\"00004700-0000-0000-0000-5a3e991a0000\"",  
    "_attachments": "attachments/",  
    "_ts": 1514051866  
}
```

Resource Properties, Self-Links, and URIs



```
{  
    "familyName": "Smith",  
    "address": {  
        "addressLine": "123 Main Street",  
        "city": "Chicago",  
        "state": "IL",  
        "zipCode": "60601"  
    },  
    "parents": [  
        "Peter",  
        "Alice"  
    ],  
    "kids": [  
        "Adam",  
        "Jacqueline",  
        "Joshua"  
    ],  
    "id": "SmithFamily"  
}
```

Resource Properties, Self-Links, and URIs



```
{  
    "familyName": "Smith",  
    "address": {  
        "addressLine": "123 Main Street",  
        "city": "Chicago",  
        "state": "IL",  
        "zipCode": "60601"  
    },  
    "parents": [  
        "Peter",  
        "Alice"  
    ],  
    "kids": [  
        "Adam",  
        "Jacqueline",  
        "Joshua"  
    ],  
    "id": "SmithFamily",  
    "_rid": "v84pA07nPQACAAAAAAABg==",  
    "_self": "dbs/v84pAA==/colls/v84pA07nPQA=/docs/v84pA07nPQACAAAAAAABg==/",  
    "_etag": "\"00004800-0000-0000-0000-5a3e9a210000\"",  
    "_attachments": "attachments/",  
    "_ts": 1514052129  
}
```

Resource Properties, Self-Links, and URIs



```
{  
    "familyName": "Smith",  
    "address": {  
        "addressLine": "123 Main Street",  
        "city": "Chicago",  
        "state": "IL",  
        "zipCode": "60601"  
    },  
    "parents": [  
        "Peter",  
        "Alice"  
    ],  
    "kids": [  
        "Adam",  
        "Jacqueline",  
        "Joshua"  
    ],  
    "id": "SmithFamily",  
    "_rid": "v84pA07nPQACAAAAAAAABg==",  
    "_self": "dbs/v84pAA==/colls/v84pA07nPQA=/docs/v84pA07nPQACAAAAAAAABg==/",  
    "_etag": "\\"00004800-0000-0000-0000-5a3e9a210000\\\"",  
    "_attachments": "attachments/",  
    "_ts": 1514052129  
}
```

Resource Properties, Self-Links, and URIs



```
{  
    "familyName": "Smith",  
    "address": {  
        "addressLine": "123 Main Street",  
        "city": "Chicago",  
        "state": "IL",  
        "zipCode": "60601" // Identify collection by...  
    },  
    "parents": [  
        "Peter", // ...self link  
        "Alice" string id1 = "dbs/v84pAA==/colls/v84pA07nPQA=/";  
    ],  
    "kids": [  
        "Adam", // ...ID-based routing  
        "Jacqueline", string id2 = "dbs/Families/colls/Families";  
        "Joshua"  
    ],  
    "id": "SmithFamily",  
    "_rid": "v84pA07nPQACAAAAAAAABg==",  
    "_self": "dbs/v84pAA==/colls/v84pA07nPQA=/docs/v84pA07nPQACAAAAAAAABg==/",  
    "_etag": "\"00004800-0000-0000-0000-5a3e9a210000\"",  
    "_attachments": "attachments/",  
    "_ts": 1514052129  
}
```

Resource Properties, Self-Links, and URIs



```
{  
    "familyName": "Smith",  
    "address": {  
        "addressLine": "123 Main Street",  
        "city": "Chicago",  
        "state": "IL",  
        "zipCode": "60601"  
    },  
    "parents": [  
        "Peter",  
        "Alice"  
    ],  
    "kids": [  
        "Adam",  
        "Jacqueline",  
        "Joshua"  
    ],  
    "id": "SmithFamily",  
    "_rid": "v84pA07nPQACAAAAAAAABg==",  
    "_self": "dbs/v84pAA==/colls/v84pA07nPQA=/docs/v84pA07nPQACAAAAAAAABg==/",  
    "_etag": "\"00004800-0000-0000-0000-5a3e9a210000\"",  
    "_attachments": "attachments/",  
    "_ts": 1514052129  
}
```

```
string id1 = "dbs/v84pAA==/colls/v84pA07nPQA=/";  
string id2 = "dbs/Families/colls/Families";  
  
Uri id3 = UriFactory.CreateDocumentCollectionUri("Families", "Families");  
► id3 {dbs/Families/colls/Families} ▾
```



Data Migration Tool

Data Migration Tool



Open source project

Microsoft Download Center

<http://www.microsoft.com/en-us/download/details.aspx?id=46436>

Documentation and How-To

<https://azure.microsoft.com/en-us/documentation/articles/documentdb-import-data/>

Source code on GitHub

<https://github.com/azure/azure-documentdb-datamigrationtool>

Import from

- SQL Server
- JSON
- CSV
- MongoDB
- Azure Table Storage
- and more...

Hands-On

★ Importing data from SQL Server





IntelliPaat

Rich Query with SQL

Rich Query with SQL



Cosmos DB SQL

Special version of SQL designed for JSON
.NET SDK also includes a LINQ provider

Familiar keywords

SELECT, FROM, WHERE, JOIN, IN, BETWEEN,
ORDER BY

Work with JSON

Descend into any subsection of a document
Iterate nested arrays for intra-document join
Return documents as-is, or as custom shape

Example

```
SELECT ch.name, ch.birthdate  
FROM Families AS f  
JOIN ch IN f.children  
WHERE f.address.state = 'CA'
```



SQL Operators and Functions

SQL Operators and Functions

Common operators

Math	+ - * / %
Bitwise	& ^ <>> >>>
Logical	AND, OR
Comparison	= <> > >= < <=
Ternary & Coalesce	? : ??
String	(concatenate)

Built-in functions

Math	ABS, CEILING, EXP, FLOOR, LOG, LOG10, POWER, ROUND, SIGN, SQRT, SQUARE, TRUNC, ACOS, ASIN, ATAN, ATN2, COS, COT, DEGREES, PI, RADIANS, SIN, TAN
Type Checking	IS_ARRAY, IS_BOOL, IS_NULL, IS_NUMBER, IS_OBJECT, IS_STRING, IS_DEFINED, IS_PRIMITIVE
String	CONCAT, CONTAINS, ENDSWITH, INDEX_OF, LEFT, LENGTH, LOWER, LTRIM, REPLACE, REPPLICATE, REVERSE, RIGHT, RTRIM, STARTSWITH, SUBSTRING, UPPER
Array	ARRAY_CONCAT, ARRAY_CONTAINS, ARRAY_LENGTH, ARRAY_SLICE
Aggregate	COUNT, SUM, MIN, MAX, AVG
Spatial	ST_DISTANCE, ST_WITHIN, ST_INTERSECTS, ST_ISVALID, ST_ISVALIDDETAILED

Hands-On

- ★ Simple SQL Query
- ★ Scalar Expression Queries
- ★ Query Operators and Built-in Functions
- ★ Querying Documents in a Collection
- ★ More SQL Queries
- ★ Aggregation Queries
- ★ Spatial Queries





Client Development

Client Development

Build web-scale applications
Uses REST/HTTP

Platform SDKs

- .NET / .NET
- Core Java
- Node.js
- Python



Introducing the .NET SDK for the SQL API

Introducing the .NET SDK for the SQL API

Create a DocumentClient instance

- Supply connection information (endpoint and key)

Invoke methods to access resources

- Create, modify, and delete resources
- Use POCOs or dynamics for document objects

Task Parallel Library (TPL)

- Simplified asynchronous programming
- Use async/await keywords with Task objects

Introducing the .NET SDK for the SQL API



Synchronous code

```
private void Main()
{
    DoSomething();
}

private void DoSomething()
{
    // do some work
}
```

Asynchronous code

```
private async void Main()
{
    await DoSomething();
}

private async Task DoSomething()
{
    // do some asynchronous work
}
```

Access resources
Update resources
Document objects

Task Parallel Library (TPL)

- Simplified asynchronous programming
- Use `async/await` keywords with `Task` objects

Introducing the .NET SDK for the SQL API



Synchronous code

```
private void Main()
{
    var result = GetSomething();
}

private string GetSomething()
{
    // do some work
    return "Hello";
}
```

Asynchronous code

```
private async void Main()
{
    var result = await GetSomething();
}

private async Task<string> GetSomething()
{
    // do some asynchronous work
    return "Hello";
}
```

Access resources
Update resources
Document objects

Task Parallel Library (TPL)

- Simplified asynchronous programming
- Use `async/await` keywords with `Task` objects

Introducing the .NET SDK for the SQL API



Create a DocumentClient instance

- Supply connection information (endpoint and key)

Invoke methods to access resources

- Create, modify, and delete resources
- Use POCOs or dynamics for document objects

Task Parallel Library (TPL)

- Simplified asynchronous programming
- Use async/await keywords with Task objects

LINQ provider

- Automatically translates LINQ queries to SQL

Hands-On

- ★ Getting started with the .NET SDK
- ★ Working with databases
- ★ Working with collections
- ★ Creating documents
- ★ Querying for documents
- ★ Replacing and deleting documents





Indexing Policies



iiPaat

iPaat

Indexing Policies



Hash index

Equality queries Strings and numbers

Range index

Equality, range, ORDER BY Strings and
numbers

Spatial index

Distance and intersection Points, polygons,
line strings

Indexing Policies



Collection-wide policy

Established when creating a collection
Can be changed after
collection is created

Indexing Policies

The screenshot displays two windows side-by-side. On the left is the Microsoft Azure Data Explorer interface for the 'cdbwus - Data Explorer' account. The 'Scale & Settings' section is selected under 'mydb'. On the right is the 'DocumentDB Data Migration Tool' showing 'Target Information'. Under 'Indexing Policy', the following JSON code is displayed:

```
1 {
2   "indexingMode": "consistent",
3   "automatic": true,
4   "includedPaths": [
5     {
6       "path": "/",
7       "indexes": [
8         {
9           "kind": "Range"
10          }
11        ]
12      }
13    ]
14 }
```

Indexing Policies



Collection-wide policy

Established when creating a collection
Can be changed after collection is created

Automatic indexing

Can switch to manual

Can override on a per-document basis

Selective indexing

Include/exclude selected property paths

Individual properties

(?)/name/?

/address/state/?

Recursive properties

(*)/address/*

/item/colors[]/*

Indexing Policies



Collection-wide policy

Established when creating a collection can be changed after collection is created

Automatic indexing

Can switch to manual

Can override on a per-document basis

Selective indexing

Include/exclude selected property paths

Indexing modes

Consistent (synchronous)

Lazy (asynchronous)



Users, Permissions, and Resource Tokens

Users, Permissions, and Resource Tokens



Resource tokens vs. master key

Provides granular control over security

Create database users

Then create permissions for each user

Get resource token from permission

Read or All access to a single resource

(collection, document, stored procedures, triggers,
user-defined functions)

Connect using resource tokens

Access will be granted based on all supplied
resource tokens

Hands-On

- ★ Custom indexing
- ★ Working with users and permissions





Server-side Programming Model

Server-side Programming Model



Run code inside Cosmos DB

- Stored procedures
- Triggers
- User-defined functions

Scoped by partition key

- Read/write documents with that partition key
- Fully transactional updates (ACID guarantee)

Bounded execution

- Implement continuation model for long running processes

Handle throttling (code 429)

- Implement retry logic when exceeding reserved throughput



Writing Server-side Code

Writing Server-side Code



Write a JavaScript function

- Obtain context object

Context exposes

- Collection
- Request
- Response



IntelliPaat

Working with Triggers

Working with Triggers



Attach custom behavior

- Access document being created or replaced

Two trigger types

- Pre-trigger
- Post-trigger

Not fired automatically

- Must be explicitly requested with an operation

Creating User-Defined Functions (UDFs)

Creating User-Defined Functions (UDFs)



Extend the query language

- Write custom business logic that can be called from inside queries

Use with care

- Querying on a UDF requires a scan (cannot use index)

No access to context

- Compute-only JavaScript

Hands-On

- ★ Creating triggers
- ★ Pre-triggers
- ★ Post-triggers
- ★ User-Defined Functions (UDFs)





Table API

What Is the Table API?

Replaces Azure Table storage

- Implement a key-value data model

Key-value data model

- Key = PartitionKey + RowKey
- Value = key-value pairs!

Leverage Cosmos DB back end

- Predictable throughput
- Global distribution
- Automatic indexing

SQL API vs. Table API



Cosmos DB

SQL API

Table API



SQL API vs. Table API



Cosmos DB	SQL API	Table API
Container	Collection	Table

SQL API vs. Table API



Cosmos DB	SQL API	Table API
Container	Collection	Table
Item	Document	Row

Document (SQL API)

```
{  
    "genre": "sci-fi",  
    "id": "Star Trek II",  
    "year": 1982,  
    "length": "1h, 53m",  
    "description": "Khan is back!"  
}
```

Row (Table API)

PartitionKey	RowKey	Year	Length	Description
sci-fi	Star Trek II	1982	1h, 53m	Khan is back!

SQL API vs. Table API



Cosmos DB	SQL API	Table API
Container	Collection	Table
Item	Document	Row
Partition Key	Any property	PartitionKey

SQL API vs. Table API



Cosmos DB	SQL API	Table API
Container	Collection	Table
Item	Document	Row
Partition Key	Any property	PartitionKey
ID	id	RowKey

Why Use the Table API?



Migrate existing applications

- Just change the connection string

Upgrade SDK

- Cosmos DB Table SDK uses native protocol

No advantage for new applications

- Table API is a layer over SQL API

Hands-On

- ★ Simple Azure Table Storage application





Cosmos DB Graph Database

Cosmos DB Graph Database



Graph container

Horizontal partitioning, provisioned throughput,
global distribution, indexing

Vertex and Edge objects

Entities and relationships

Both can hold arbitrary key-value pairs

Create/Retrieve/Update/Delete

GraphSON and Gremlin

Apache TinkerPop

<http://tinkerpop.apache.org>

Focus on relationships

Chain relationship queries with Gremlin

Graph Database Scenarios

Graph Database Scenarios



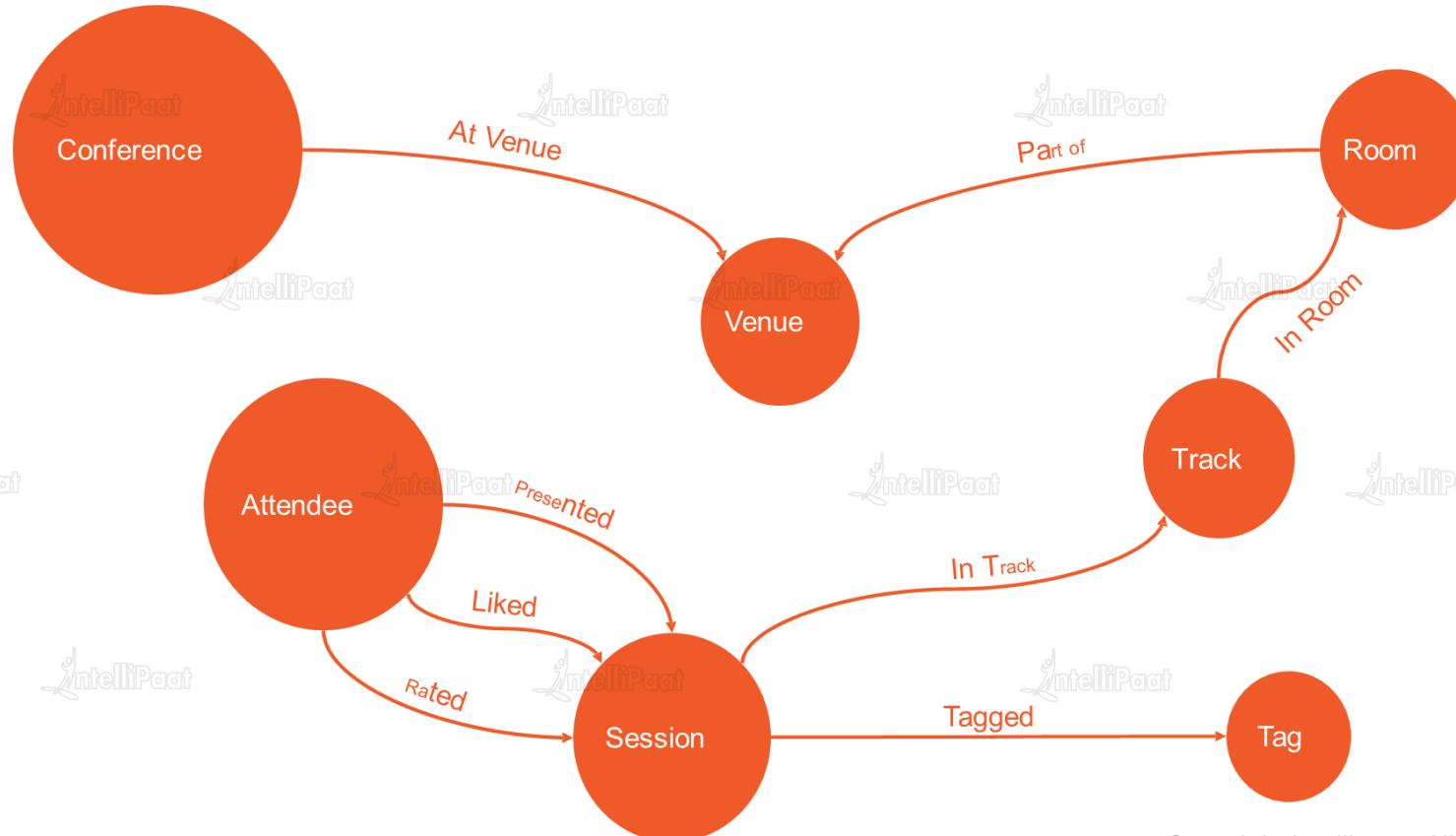
Complex relationships

- Many “many-to-many” relationships
- Excessive JOINs
- Analyze interconnected data and relationships

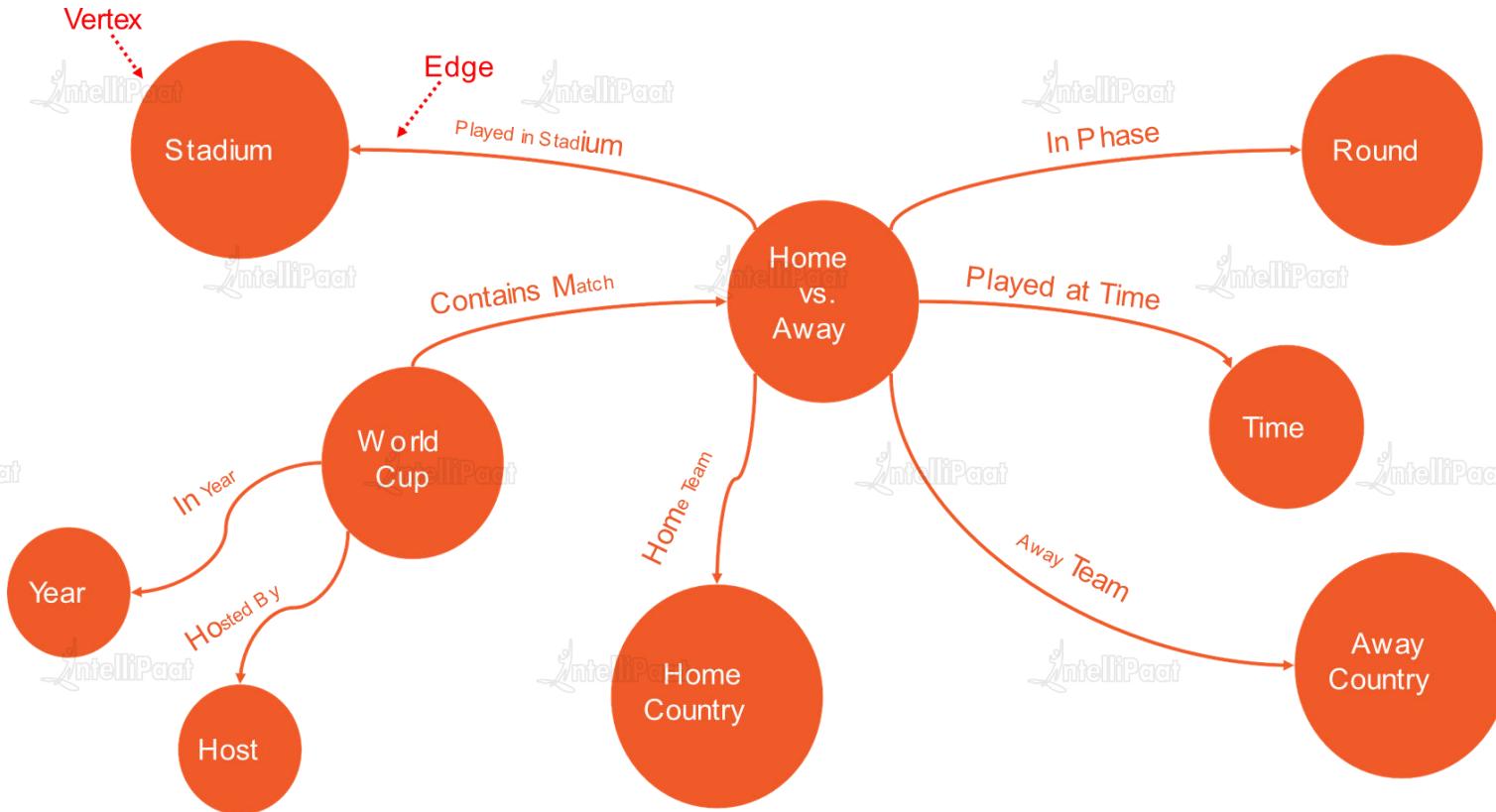
Typical graph applications

- Social networks
- Recommendation engines
- Knowledge graphs
- Many more...

Graph Database Scenarios



Graph Database Scenarios





Vertices and Edges



Vertices and Edges



Vertex and Edge properties

- id (within partition key) label (type)
- Additional arbitrary properties (including the partition key)

Additional Edge properties

- Cardinality(in-and-out vertices)
- Create two edges for bi-directionality

Vertices and Edges

IntelliPaat



Vertex
label: person
id: John
age: 25
likes: pizza



Vertex
label: company
id: Acme
founded: 2001
location: NY



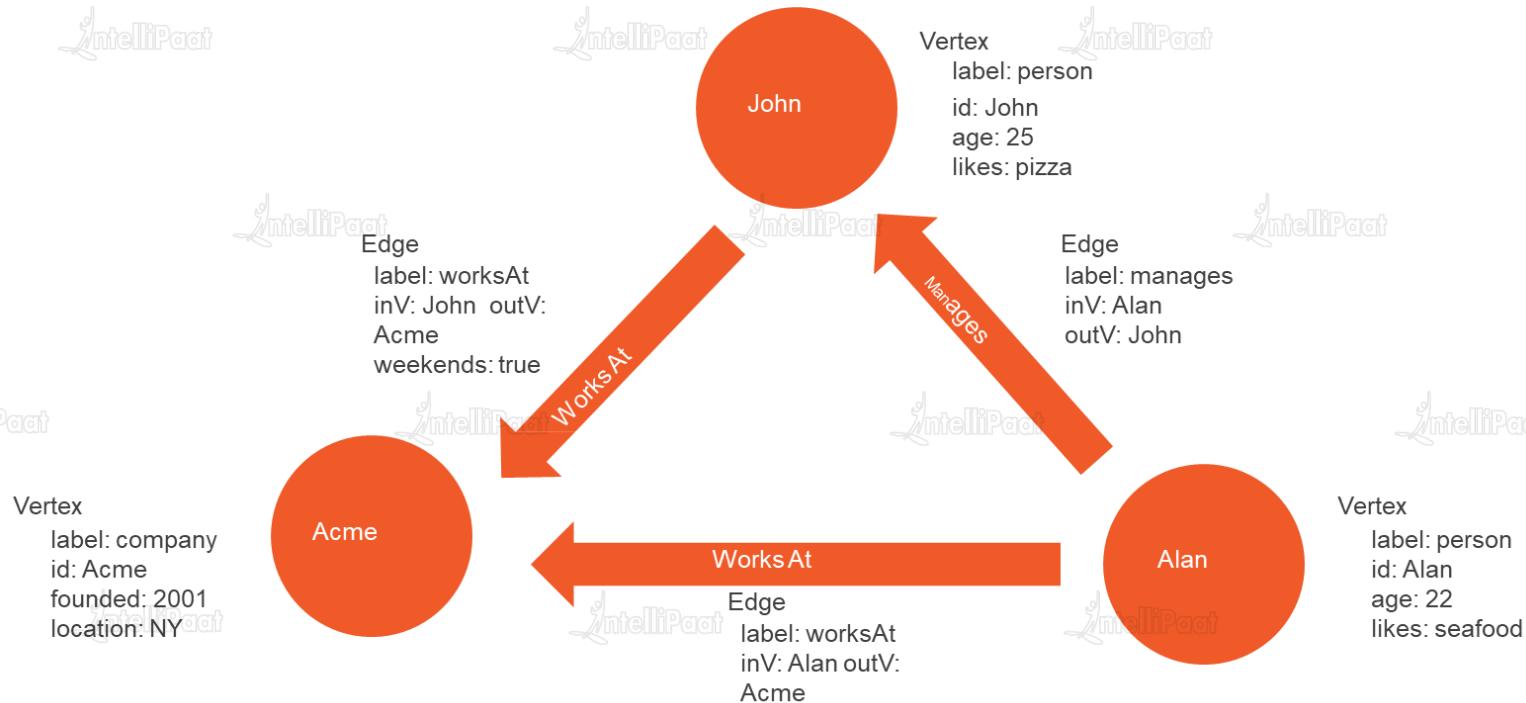
Vertex
label: person
id: Alan
age: 22
likes: seafood



Vertices and Edges

IntelliPaat

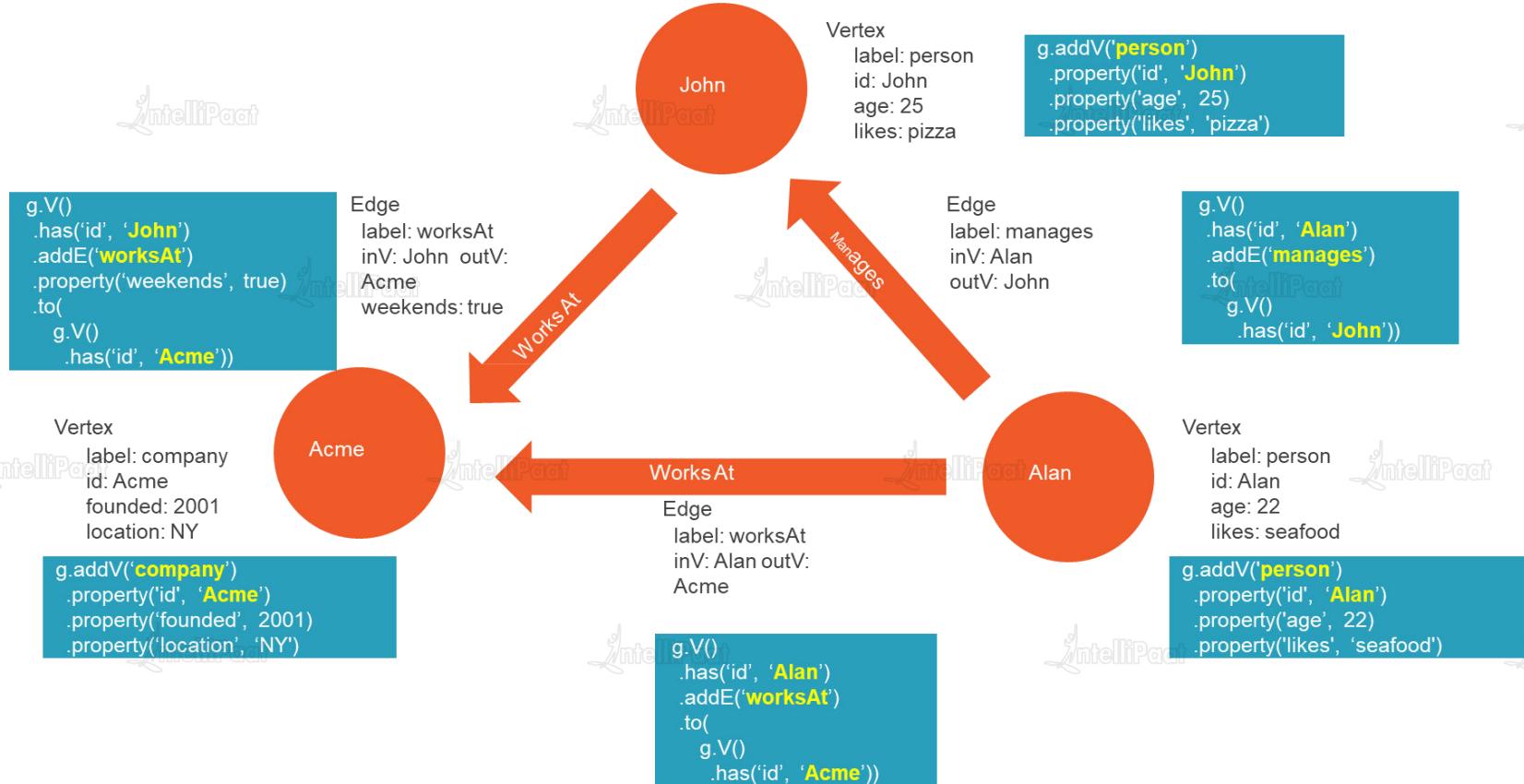
IntelliPaat



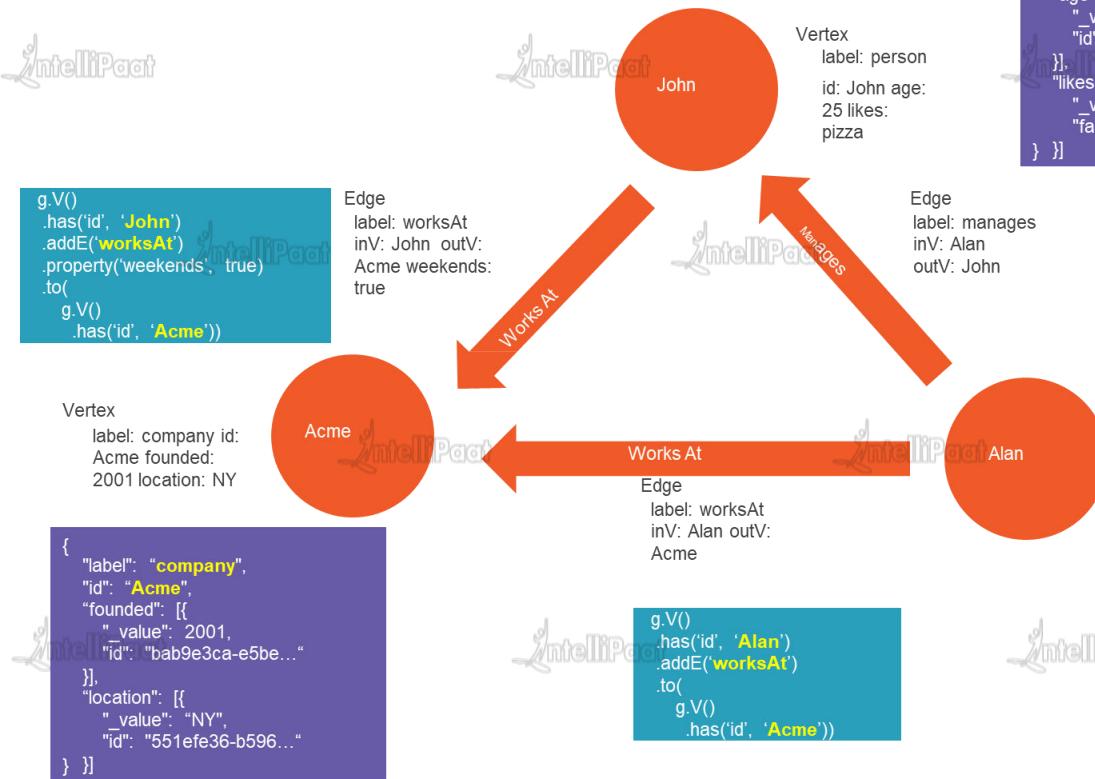


Populating the Graph

Populating the Graph



Populating the Graph



```
{  
    "label": "person",  
    "id": "John",  
    "age": [{  
        "value": 25,  
        "id": "2006891f-e73d..."  
    }],  
    "likes": [{  
        "value": "pizza", "id": "fa8deae0-f4c9..."  
    } ]}
```

```
g.V()  
.has('id', 'Alan')  
.addE('manages')  
.to(  
    g.V()  
.has('id', 'John'))
```

```
Vertex  
label: person  
id: Alan  
age: 22  
likes: seafood
```

```
{  
    "label": "person",  
    "id": "Alan",  
    "age": [{  
        "value": 25,  
        "id": "cfaeb335-371a..."  
    }],  
    "likes": [{  
        "value": "seafood", "id": "79c14511-c80f..."  
    } ]}
```

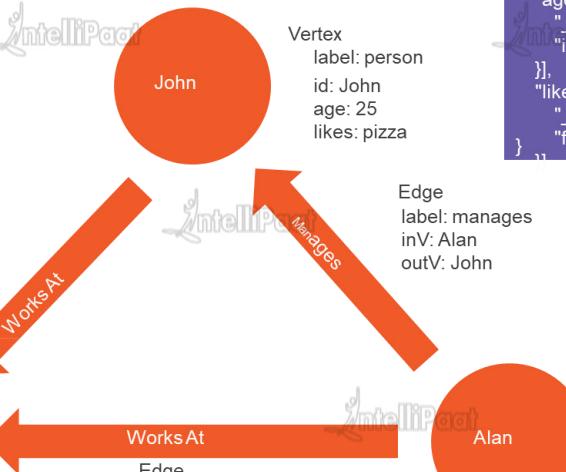
Populating the Graph

```
{  
  "id": "c837bf96-fc2d...",  
  "_vertexId": "John",  
  "_vertexLabel": "person",  
  "label": "worksAt",  
  "_sink": "Acme",  
  "_sinkLabel": "company",  
  "weekends": true,  
  "_isEdge": true  
}
```

Vertex
label: company
id: Acme
founded: 2001
location: NY

```
{  
  "label": "company",  
  "id": "Acme",  
  "founded": [  
    {"_value": 2001, "id": "bab9e3ca-e5be..."}  
  ],  
  "location": [  
    {"_value": "NY", "id": "551efe36-b596..."}  
  ]  
}
```

Edge
label: worksAt
inV: John outV:
Acme
weekends: true



```
{  
  "label": "person",  
  "id": "John",  
  "age": [{"_value": 25, "id": "2006891f-e73d..."}],  
  "likes": [{"_value": "pizza", "id": "fa8deae0-f4c9..."}]  
}
```

```
{  
  "id": "59609f8f-c925...",  
  "_vertexId": "Alan",  
  "_vertexLabel": "person",  
  "label": "manages",  
  "_sink": "John",  
  "_sinkLabel": "person",  
  "_isEdge": true  
}
```

Vertex
label: person
id: Alan
age: 22
likes: seafood

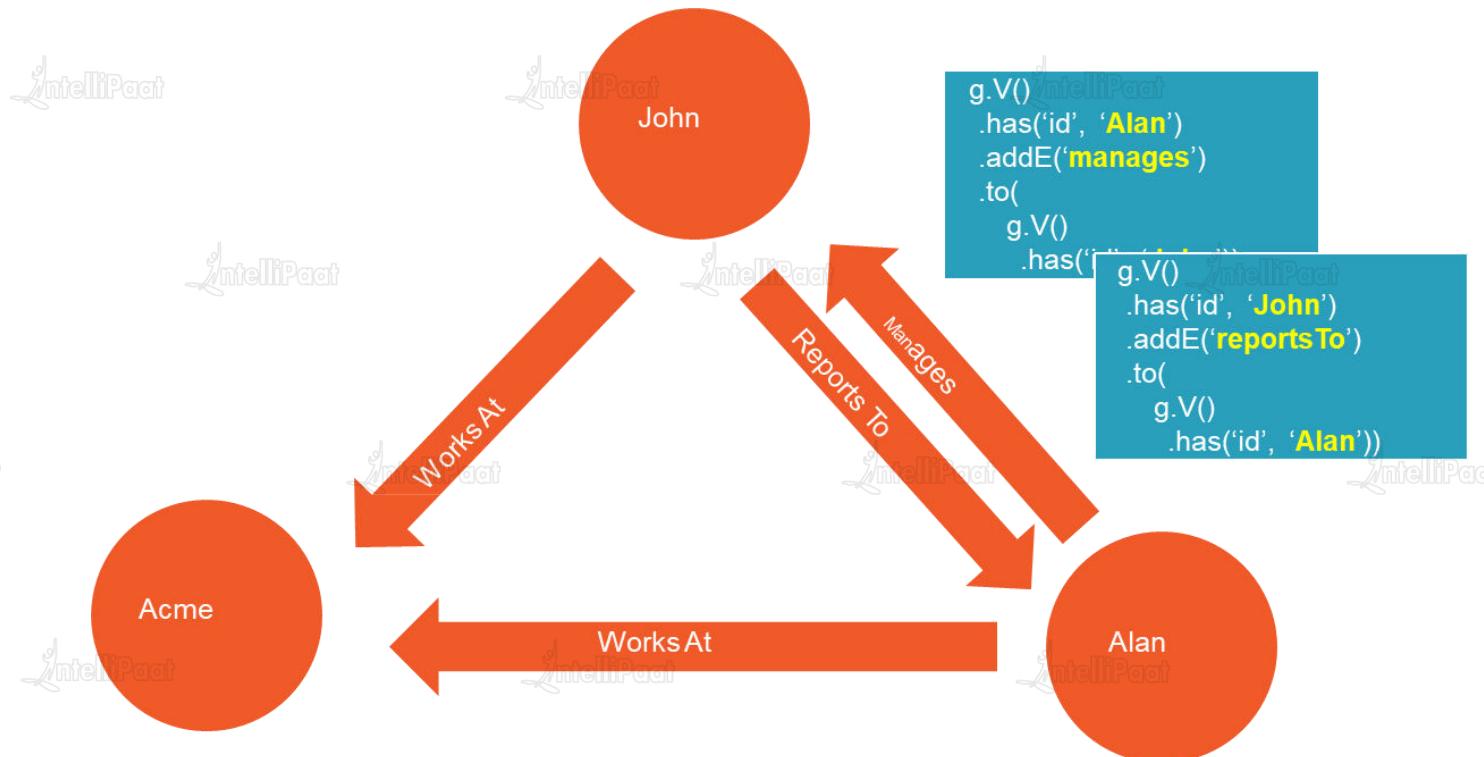
```
{  
  "label": "person",  
  "id": "Alan",  
  "age": [{"_value": 25, "id": "cfaeb335-371a..."}],  
  "likes": [{"_value": "seafood", "id": "79c14511-c80f..."}]  
}
```

```
{  
  "id": "fb3c578f-9ef7...",  
  "_vertexId": "Alan",  
  "_vertexLabel": "person",  
  "label": "worksAt",  
  "_sink": "Acme",  
  "_sinkLabel": "company",  
  "_isEdge": true  
}
```

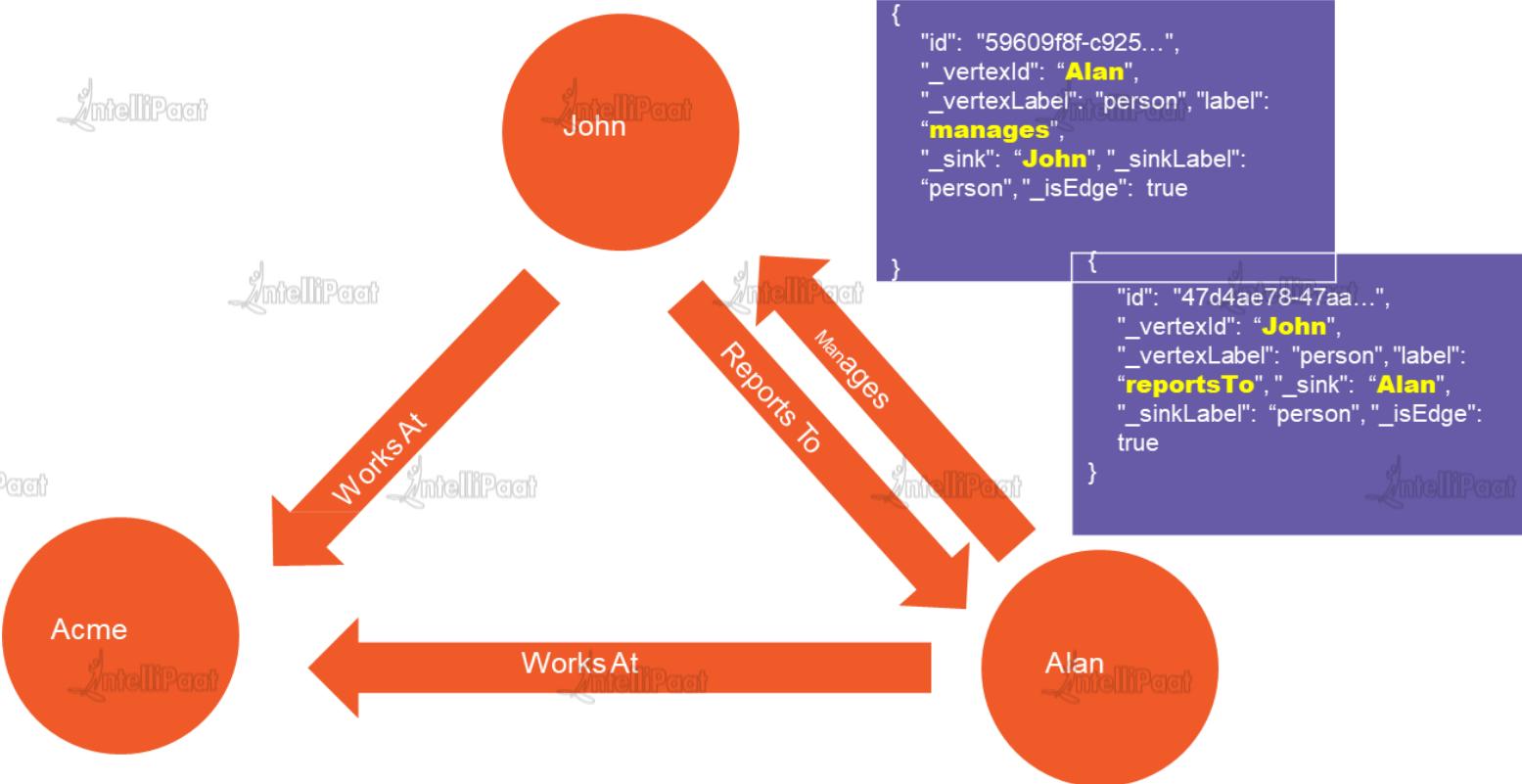


Bi-Directional Relationships

Bi-Directional Relationships



Bi-Directional Relationships



Hands-On

- ★ Creating a simple graph





IntelliPaat

Writing Gremlin Queries

Writing Gremlin Queries



Cosmos DB Gremlin support

<http://aka.ms/gremlin-support>

Functional language

Chain multiple steps together

Define Vertices and Edges

.addV('label')

.addE('label')

.property('key', 'value')

Query on filters and relationships

.V('label')

.out('label')

.has('property', condition)

Writing Gremlin Queries

The screenshot shows a Microsoft Edge browser window with the following details:

- Title Bar:** Azure Cosmos DB Gremlin
- Address Bar:** https://docs.microsoft.com/en-us/azure/cosmos-db/gremlin-support#gremlin-steps
- Left Sidebar (Navigation):**
 - Filter
 - > Samples
 - > Concepts
 - > How To Guides
 - > Develop
 - > SQL API
 - > MongoDB API
 - > Graph API
 - Gremlin support** (highlighted in blue)
 - > Table API
 - Change feed
 - Geospatial
 - Indexing
 - Connected Service in Visual Studio
 - > Manage
 - > Integrate
 - > Reference
 - > Resources
- Main Content Area:**

Gremlin steps

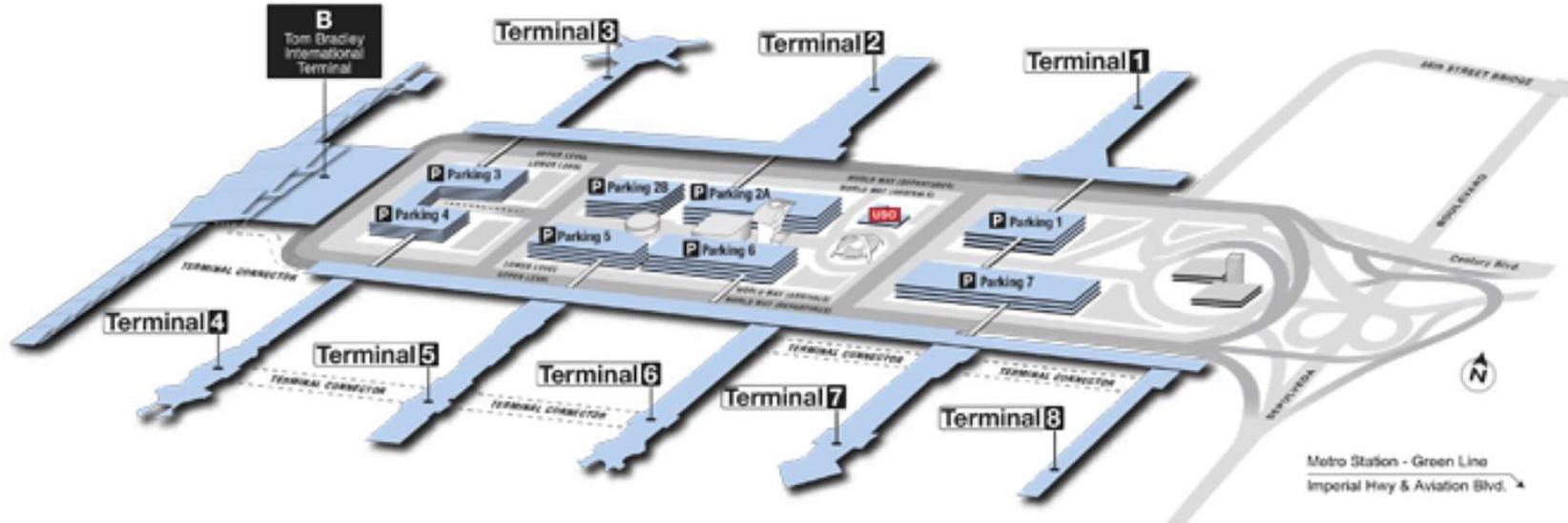
Now let's look at the Gremlin steps supported by Azure Cosmos DB. For a complete reference on Gremlin, see [TinkerPop reference](#).

step	Description	TinkerPop 3.2 Documentation	Notes
<code>addE</code>	Adds an edge between two vertices	addE step	
<code>addV</code>	Adds a vertex to the graph	addV step	
<code>and</code>	Ensures that all the traversals return a value	and step	
<code>as</code>	A step modulator to assign a variable to the output of a step	as step	
<code>by</code>	A step modulator used with <code>group</code> and <code>order</code>	by step	
<code>coalesce</code>	Returns the first traversal that returns a result	coalesce step	
<code>constant</code>	Returns a constant value. Used with <code>coalesce</code>	constant step	
<code>count</code>	Returns the count from the traversal	count step	
- Right Sidebar (Actions):**
 - Feedback
 - Edit
 - Share
 - Theme: Light
- In This Article:**
 - [Gremlin by example](#)
 - [Gremlin features](#)
 - [Gremlin wire format: GraphSON](#)
 - [Gremlin partitioning](#)
 - Gremlin steps** (highlighted in blue)
 - [Next Steps](#)
- Bottom Right Corner:** Is this page helpful? (YES / NO)



Demo: Busy World Traveler

Demo: Busy World Traveler

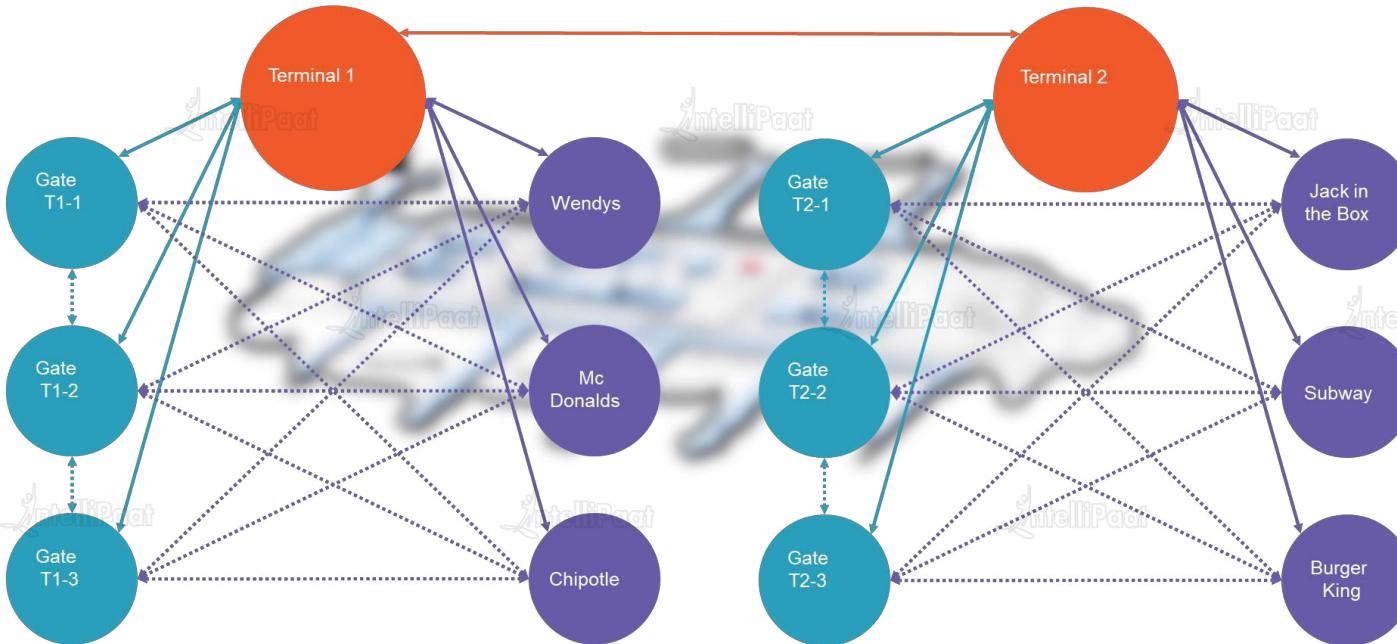


Hands-On

- ★ Creating a simple graph
- ★ Busy world traveler



Demo: Busy World Traveler



Hands-On

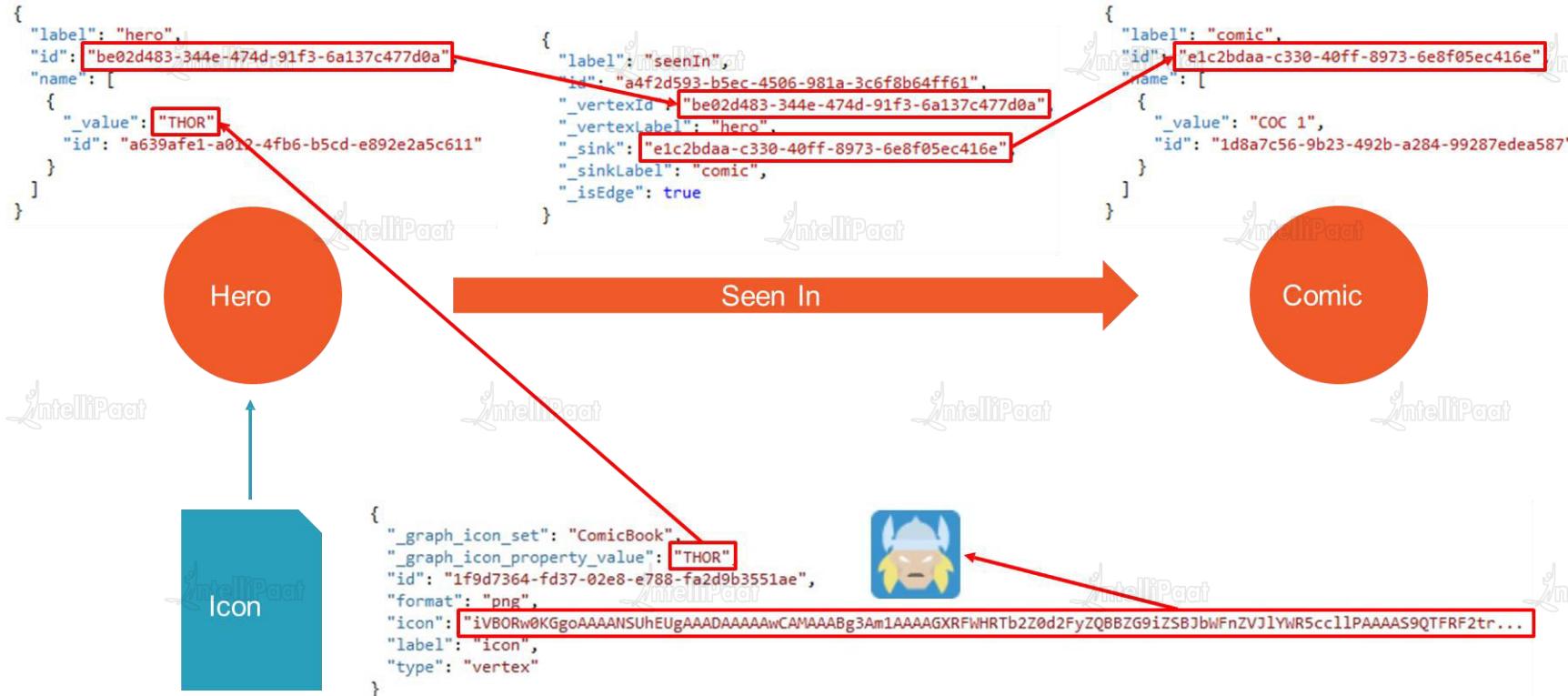
- ★ Populating the airport graph
- ★ Querying the airport graph
- ★ Multi-model comic book catalog





Multi-Model Comic Book Catalog

Multi-Model Comic Book Catalog



Following This Course



Introduction & Overview

Storage Account, Table Storage Overview,
Browsing and Creating Data

Integrating with Table Storage

Working with the Azure Table Storage from
.NET Applications to perform basic CRUD
operations



IntelliPaat

Azure Table Storage

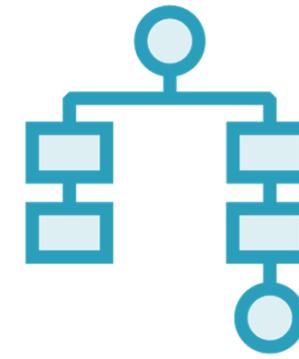
Why Choose the Azure Table Storage?



Low Latency



Scalable Apps & High
Availability



Flexible data structure

Azure Table Storage



- ★ NoSQL
- ★ Key-value storage
- ★ Uses JSON to serialize the data
- ★ Flexible schema changes
- ★ Scalable and highly available
- ★ Single region with optional secondary read-only region

Different Azure Table Storage Options



Azure Table Storage

- ★ No global distribution
- ★ Additional read-only region
- ★ Low latency
- ★ Can only index using a primary key
- ★ Eventual consistency using a second region
- ★ Price optimized for Storage

Azure Cosmos DB Table API

- ★ Globally distributed
- ★ Slightly higher latency, single digit latency
- ★ No need index management
- ★ Can index on any property (automatic)
- ★ Consistency can be tuned to the application specific needs
- ★ Price optimized for Throughput

Hands-On

- ★ Setting up Azure Table Storage from the Microsoft Azure Portal
- ★ Using the Table Service REST API
- ★ Setting up Azure Table Storage with the Azure Cosmos DB Table API
- ★ Migrating to Azure Cosmos DB Table API



Hands-On

- ★ Introducing Azure Table Storage in a .NET Application
- ★ Modifying the Structure of Entities
- ★ Migrating to Azure Cosmos DB Table API
- ★ Generating Shared Access Signatures from Code
- ★ Applying and Enforcing Table Policies
- ★ Exporting Metrics





India: +91-7847955955



US: 1-800-216-8930 (TOLL FREE)



sales@intellipaat.com

24/7 Chat with Our Course Advisor