



NoSQL Data Stores

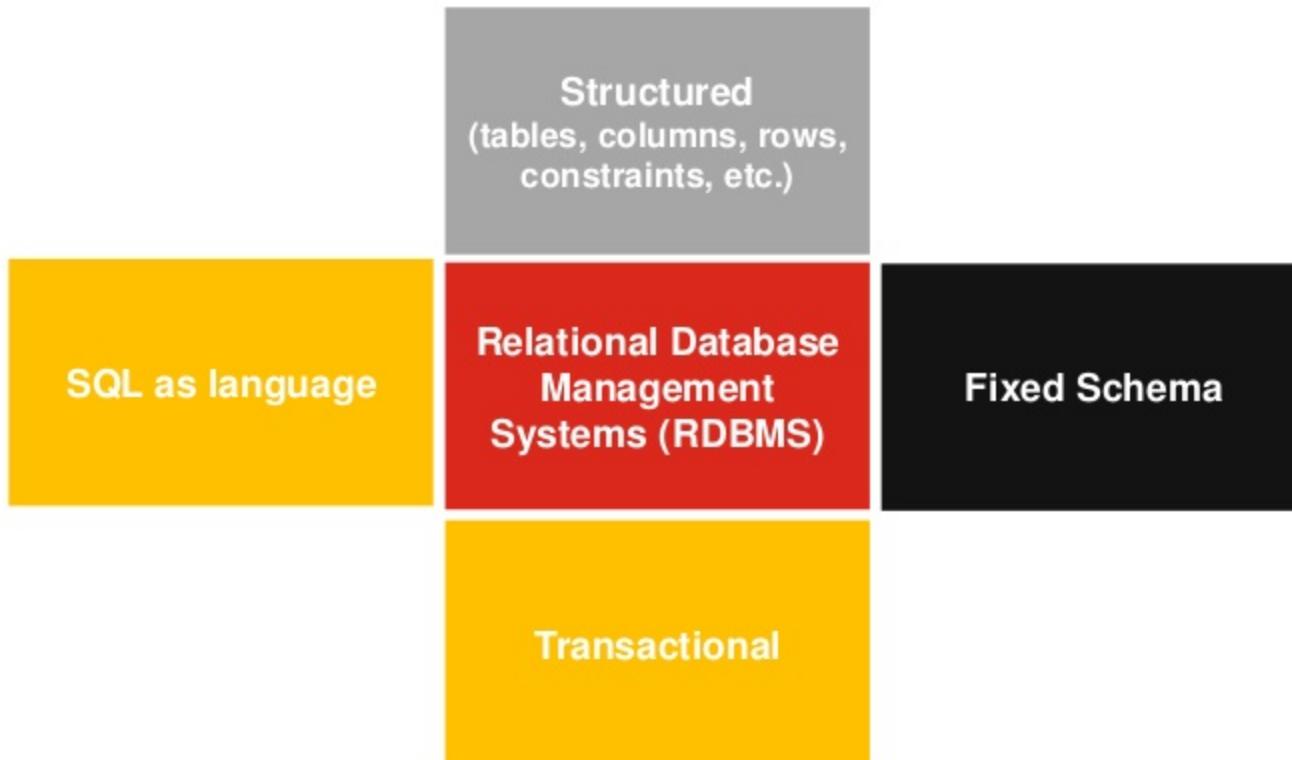
with Microsoft Azure

Outline

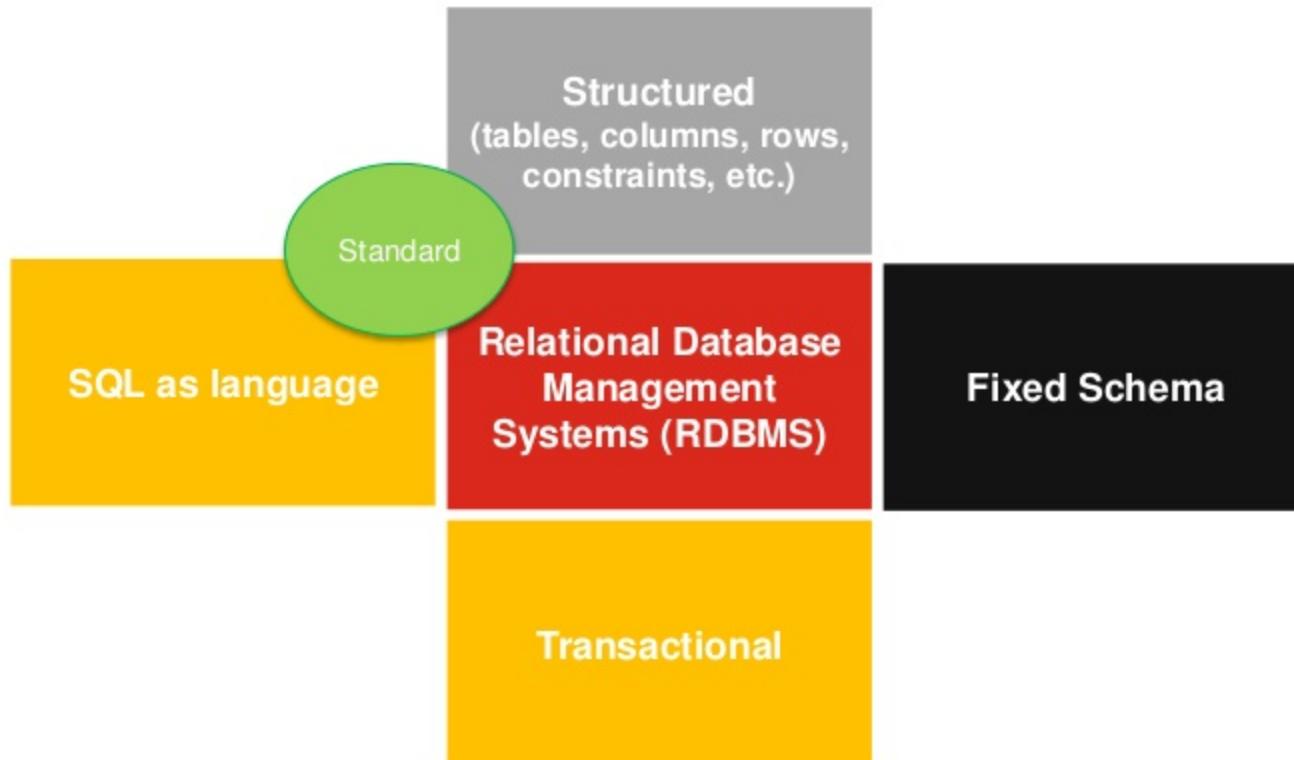
- What is NoSQL, and why?
- NoSQL Data Stores, tools & Technologies
- Solution Architecture Patterns
- Introducing Azure Redis Cache
- Introducing Azure Table Storage
- Introducing Azure DocumentDB
- Introducing HBase on Azure HDInsight
- How to Get Started with NoSQL
- Appendix: how to play with neo4j

Fundamentals

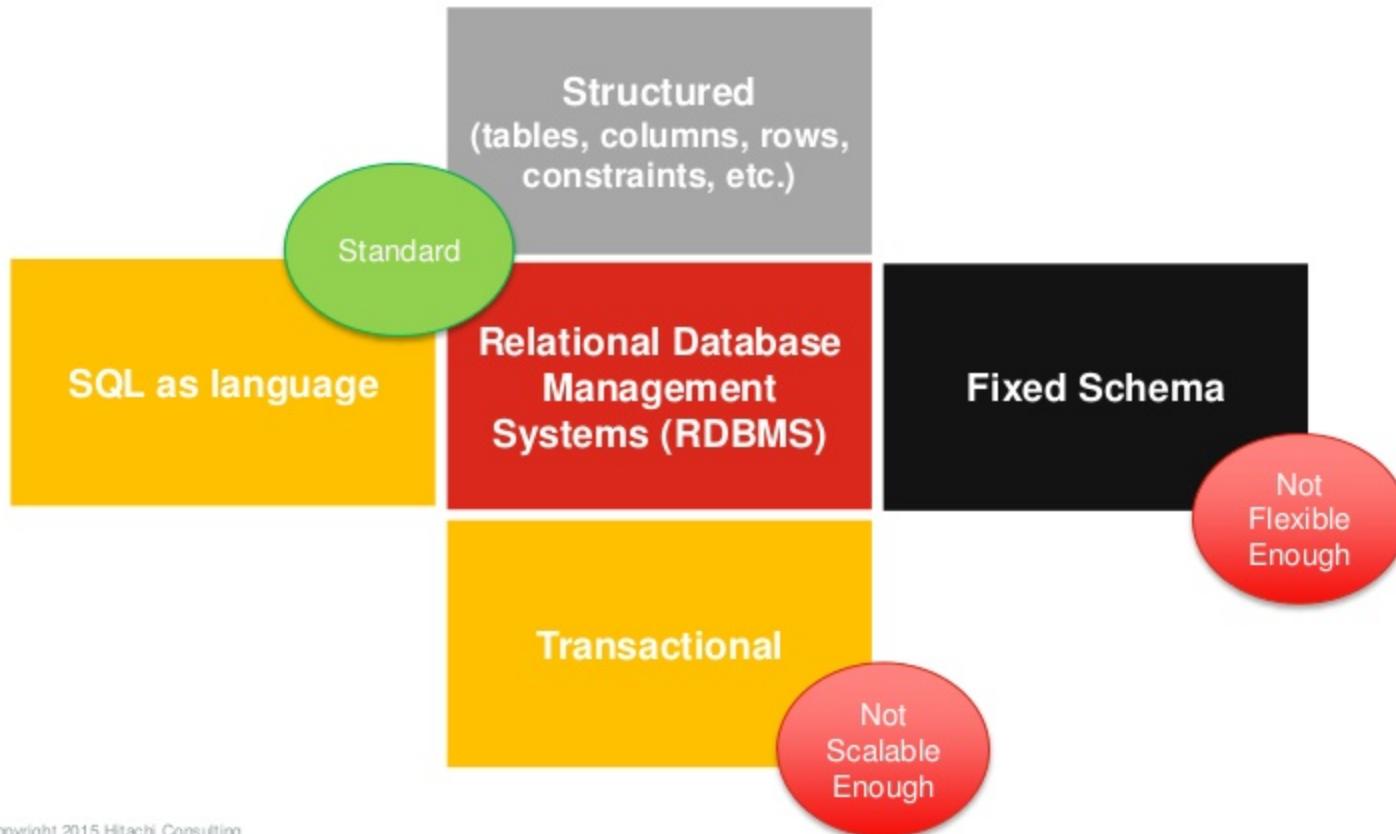
What is SQL?



What is SQL?



What is SQL?



What is NoSQL?

key attributes..



Non-relational

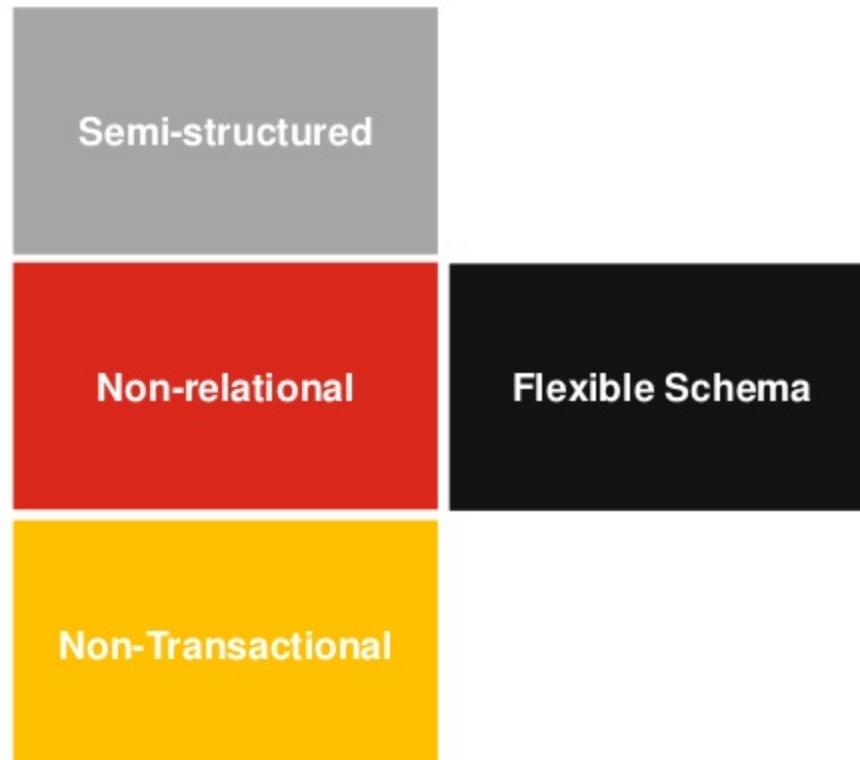
What is NoSQL?

key attributes..



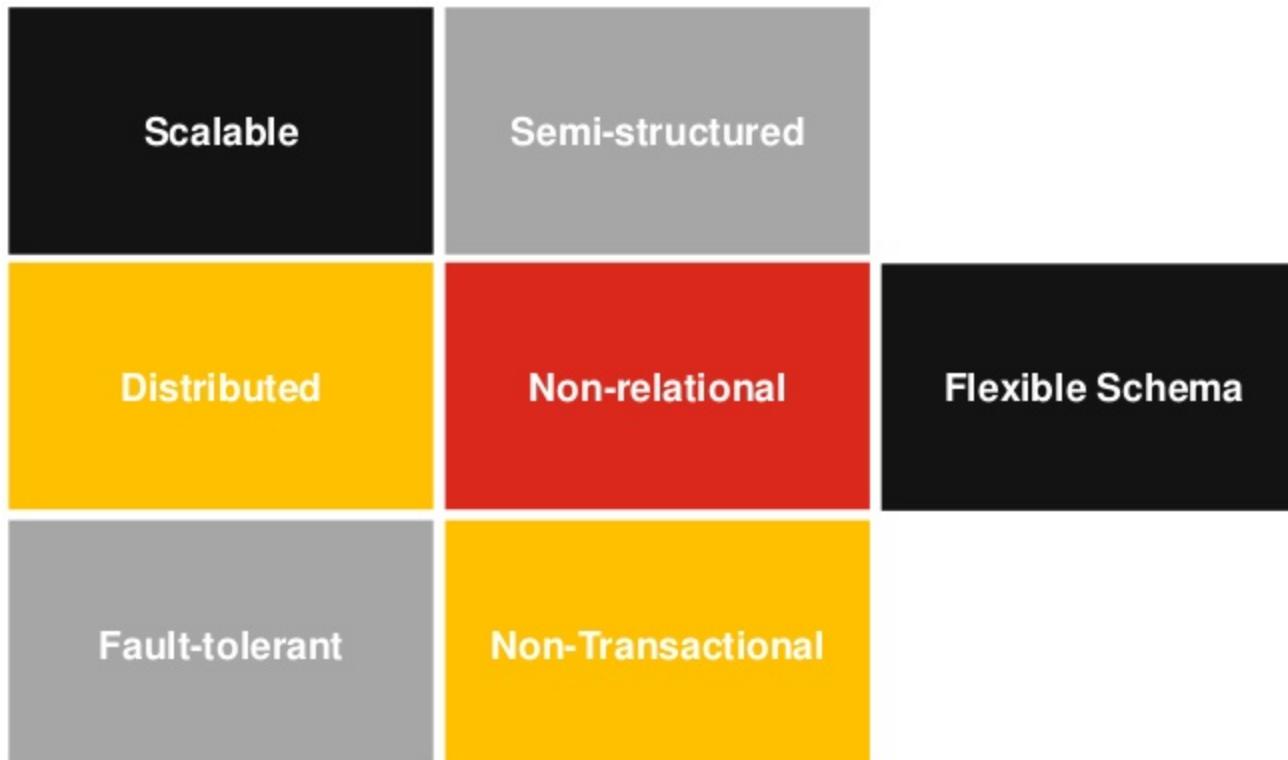
What is NoSQL?

key attributes..



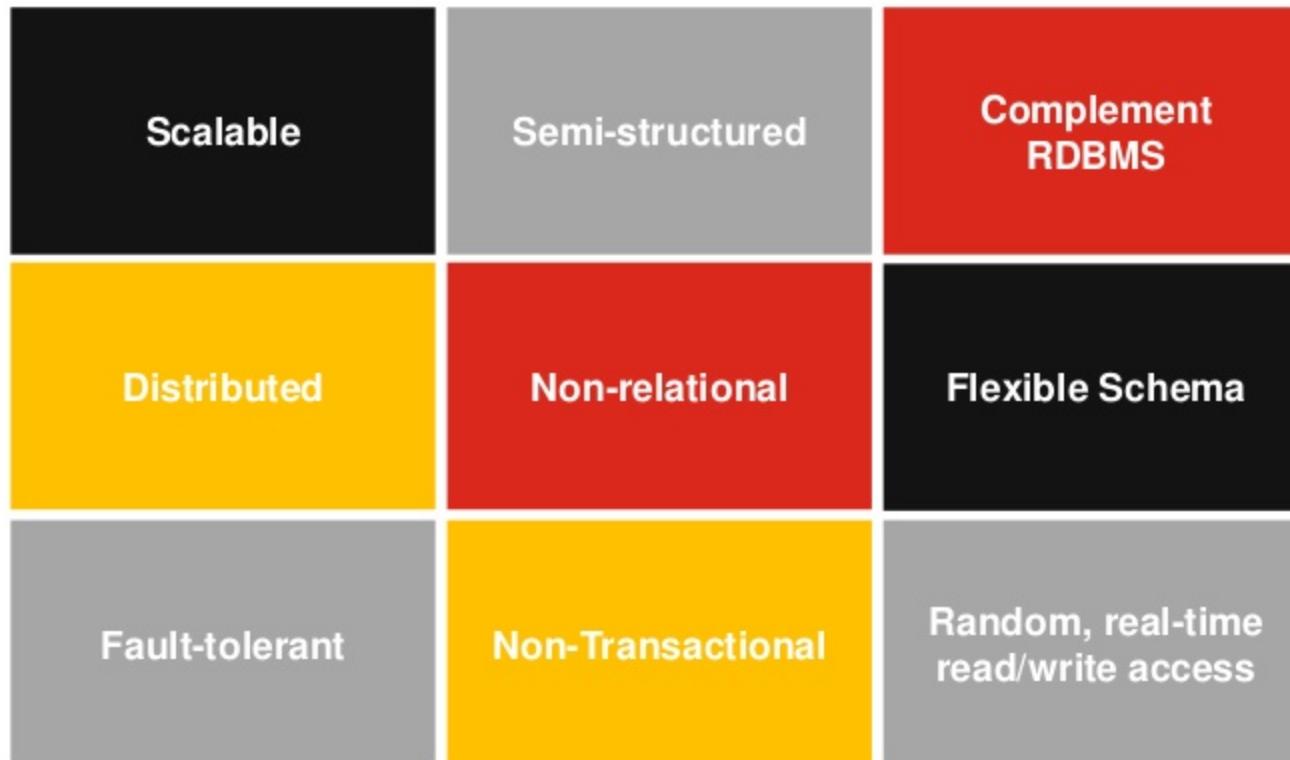
What is NoSQL?

key attributes..



What is NoSQL?

key attributes..



Why NoSQL?

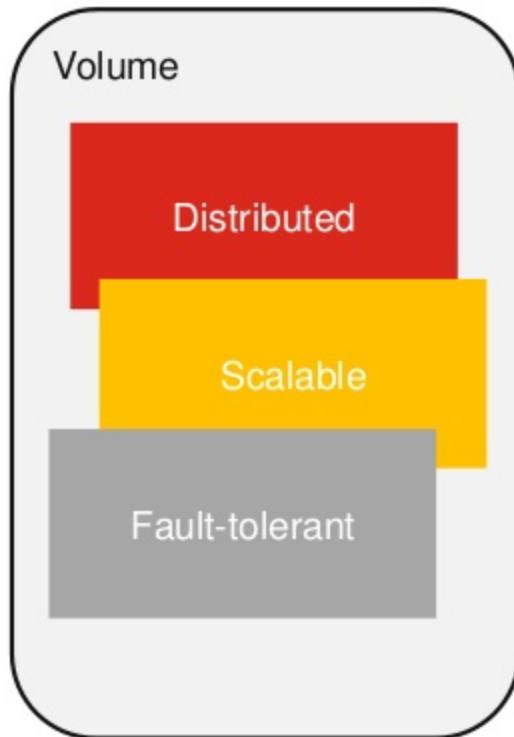
It is all about Big Data...

NoSQL data stores help overcoming Big Data challenges in real-time operational systems

Why NoSQL?

It is all about Big Data...

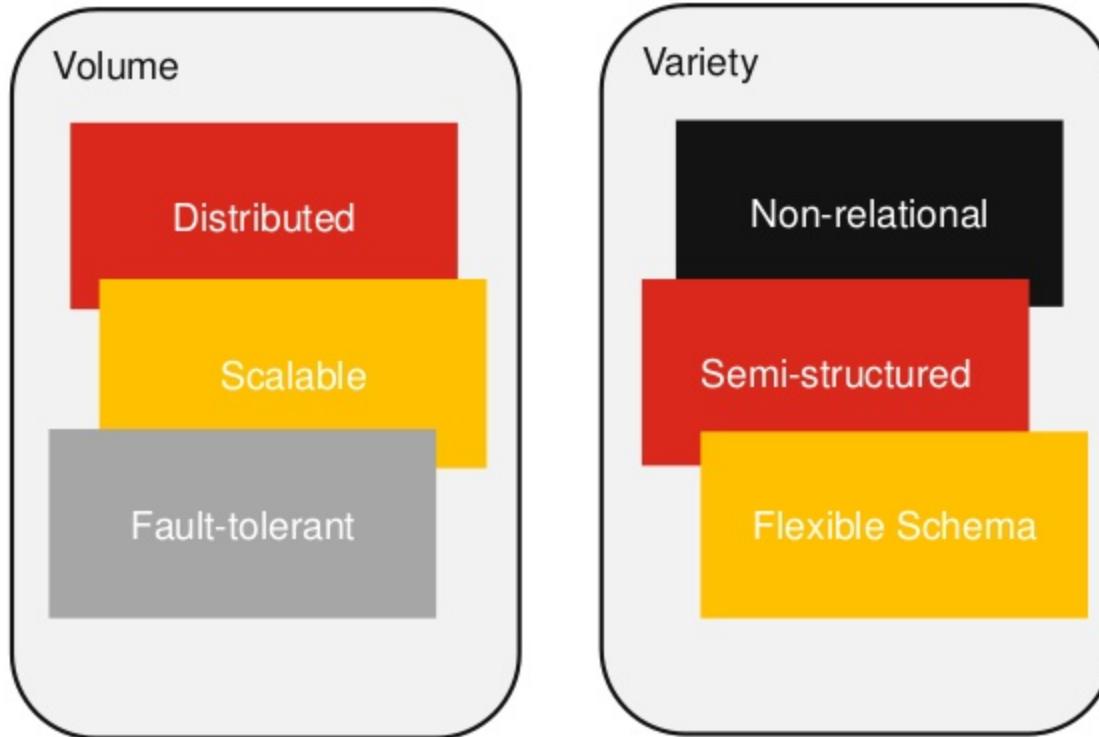
NoSQL data stores help overcoming Big Data challenges in real-time operational systems



Why NoSQL?

It is all about Big Data...

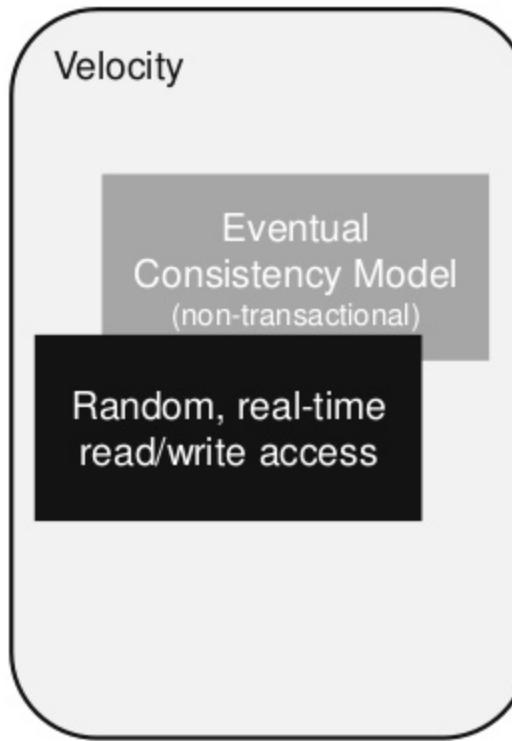
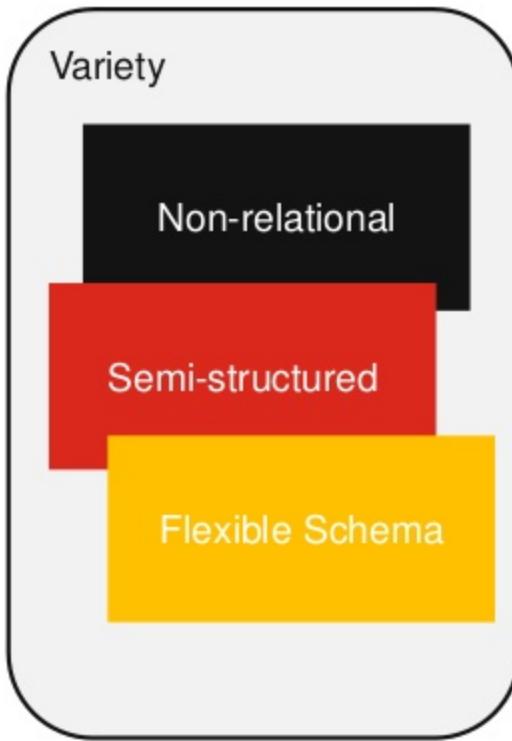
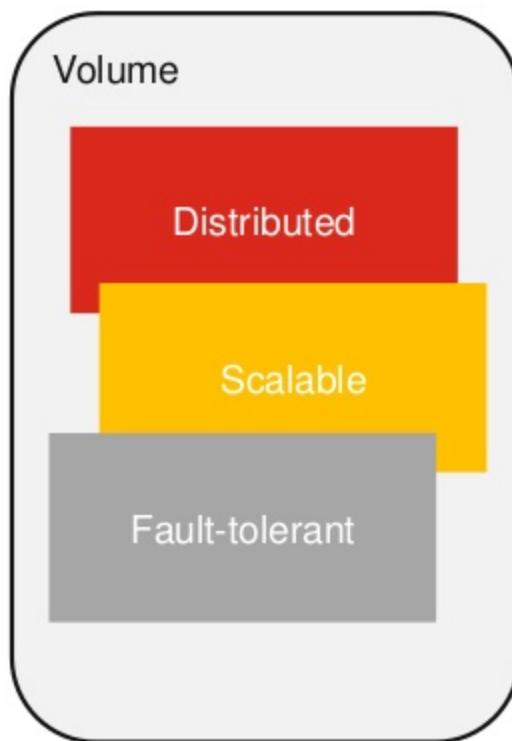
NoSQL data stores help overcoming Big Data challenges in real-time operational systems



Why NoSQL?

It is all about Big Data...

NoSQL data stores help overcoming Big Data challenges in real-time operational systems



NoSQL Usage Patterns

suitability...

Suitable for

Random, real-time read/write access

Reference Data

Variable Data
Structures

Singleton Select/ Insert/ update

NoSQL Usage Patterns

suitability...

Suitable for

Random, real-time read/write access

Reference Data

Variable Data Structures

Singleton Select/ Insert/ update

Not Suitable for

Batch Processing

Complex Analytical Queries

Joins

Complex Transactions

CAP Theorem

NoSQL & CAP Theorem

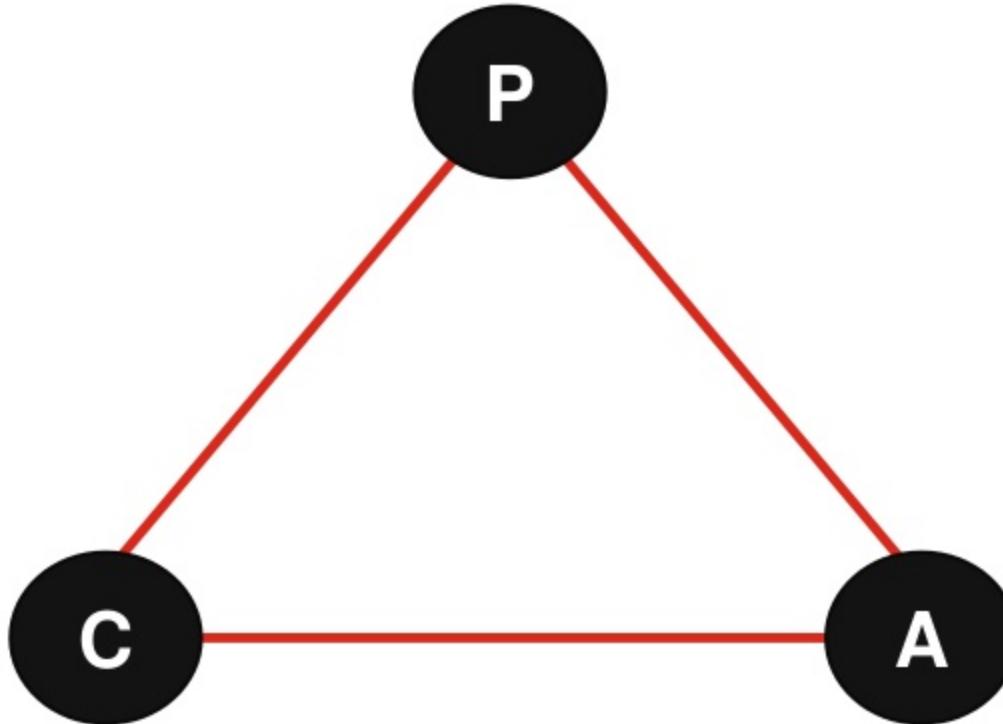
The trade-off...

- In order to handle large volume of data processing efficiently, we need to **scale out**, i.e. **partition** the data and **distribute** the computation
- Now we face a trade-off between **Consistency**, **Availability**, and **Partition Tolerance**
 - **Consistency:** Data is in a consistent state across all the nodes.
That is, **all** the reads would get you the same, **most recent** write.
 - **Availability:** Every request to the system **gets a response** (i.e., executed) on success/failure.
That is, system **responsiveness** (latency).
 - **Partition Tolerance:** The system **continuous to work** despite of message loss or **partition (node) failure**. That is, the system can sustain partial network failures.
- **CAP Theorem:** only two out of three properties can be satisfied in a distributed data system. In facet, it is **consistency vs availability**, wrt partition tolerance!

NoSQL & CAP Theorem

The trade-off...

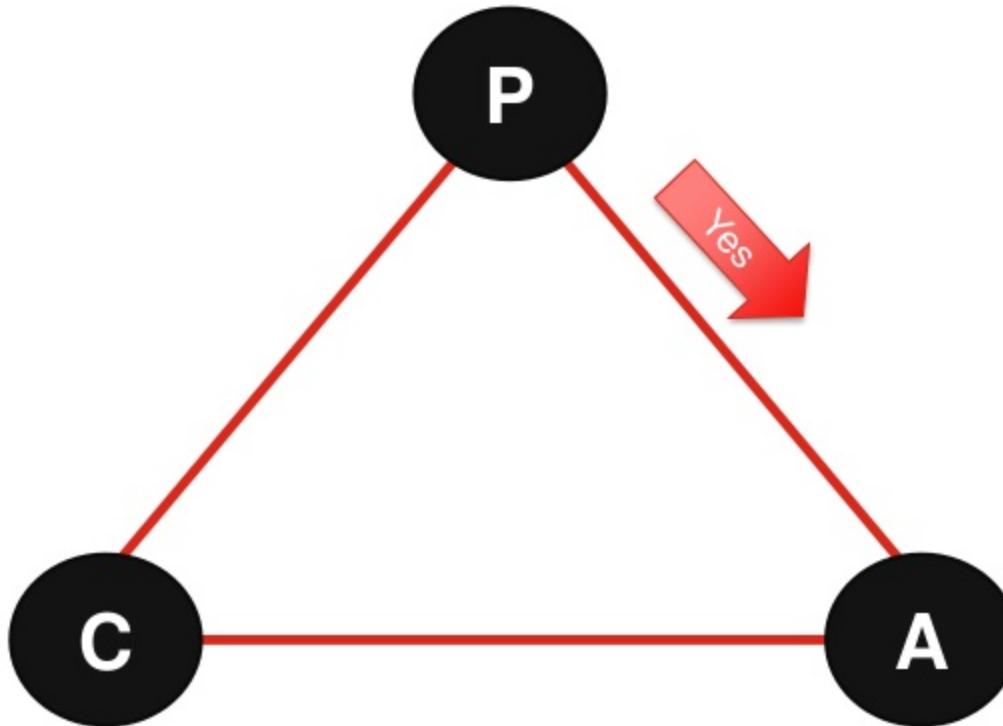
Continue working if a partition is not
reachable by the system?



NoSQL & CAP Theorem

The trade-off...

Continue working if a partition is not
reachable by the system?



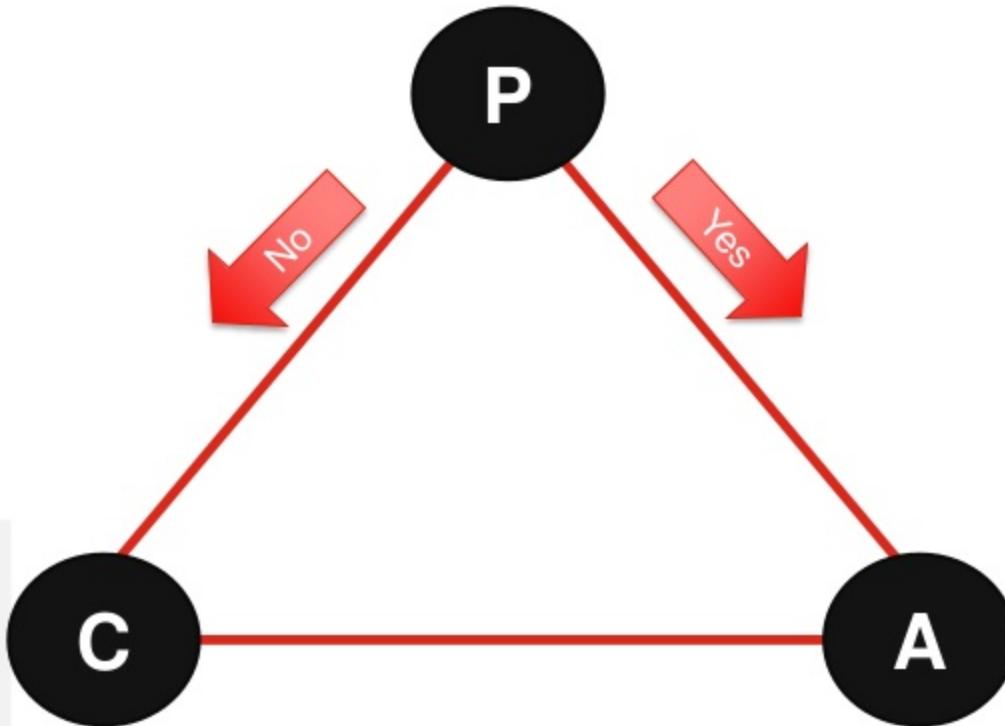
Big Data Systems

- BASE Mode – Eventually Consistency
- Remains available (operational & responsive)
- partition tolerant, i.e., **sacrifices consistency**

NoSQL & CAP Theorem

The trade-off...

Continue working if a partition is not
reachable by the system?



Transactional RDBMS

- ACID Mode – Strong Consistency
- Commits are atomic across the entire system
- Not partition tolerant, i.e., **sacrifices availability**

Big Data Systems

- BASE Mode – Eventually Consistency
- Remains available (operational & responsive)
- partition tolerant, i.e., **sacrifices consistency**

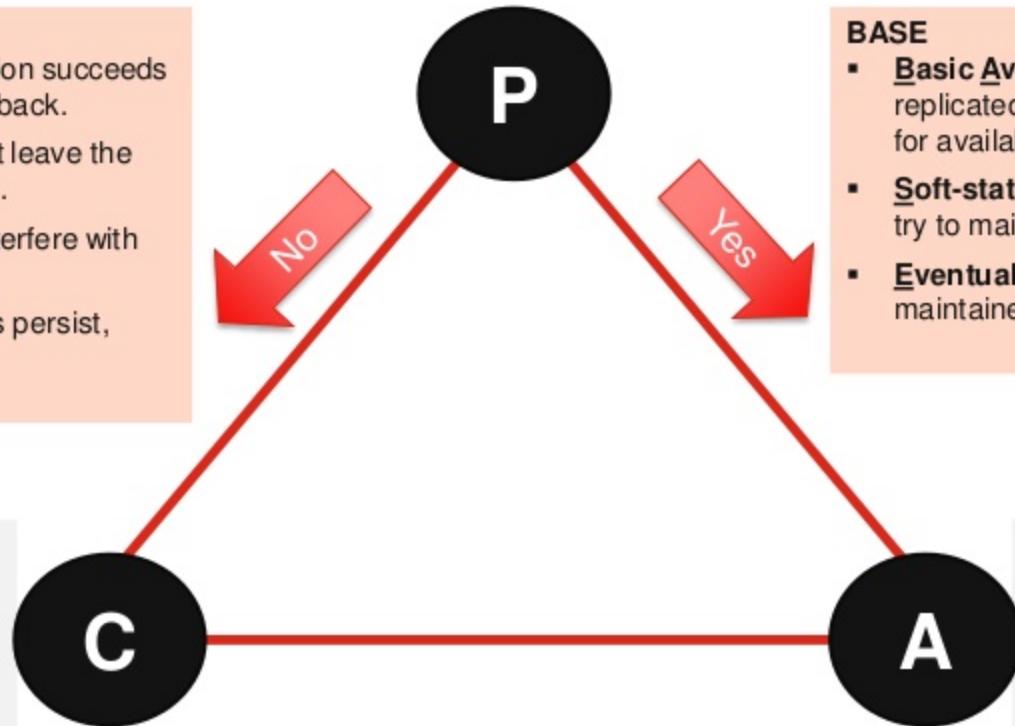
NoSQL & CAP Theorem

The trade-off...

Continue working if a partition is not
reachable by the system?

ACID

- **Atomic**: Everything in a transaction succeeds or the entire transaction is rolled back.
- **Consistent**: A transaction cannot leave the database in an inconsistent state.
- **Isolated**: Transactions cannot interfere with each other.
- **Durable**: Completed transactions persist, even when servers restart etc.



BASE

- **Basic Availability**: data is sharded and replicated, and consistency is compromised for availability
- **Soft-state**: Allow data to be inconsistent and try to maintain consistency later.
- **Eventual consistency**: Consistency is maintained later.

Transactional RDBMS

- ACID Mode – Strong Consistency
- Commits are atomic across the entire system
- Not partition tolerant, i.e., **sacrifices availability**

Big Data Systems

- BASE Mode – Eventually Consistency
- Remains available (operational & responsive)
- partition tolerant, i.e., **sacrifices consistency**

NoSQL & CAP Theorem

The trade-off...

Continue working if a partition is not
reachable by the system?

ACID

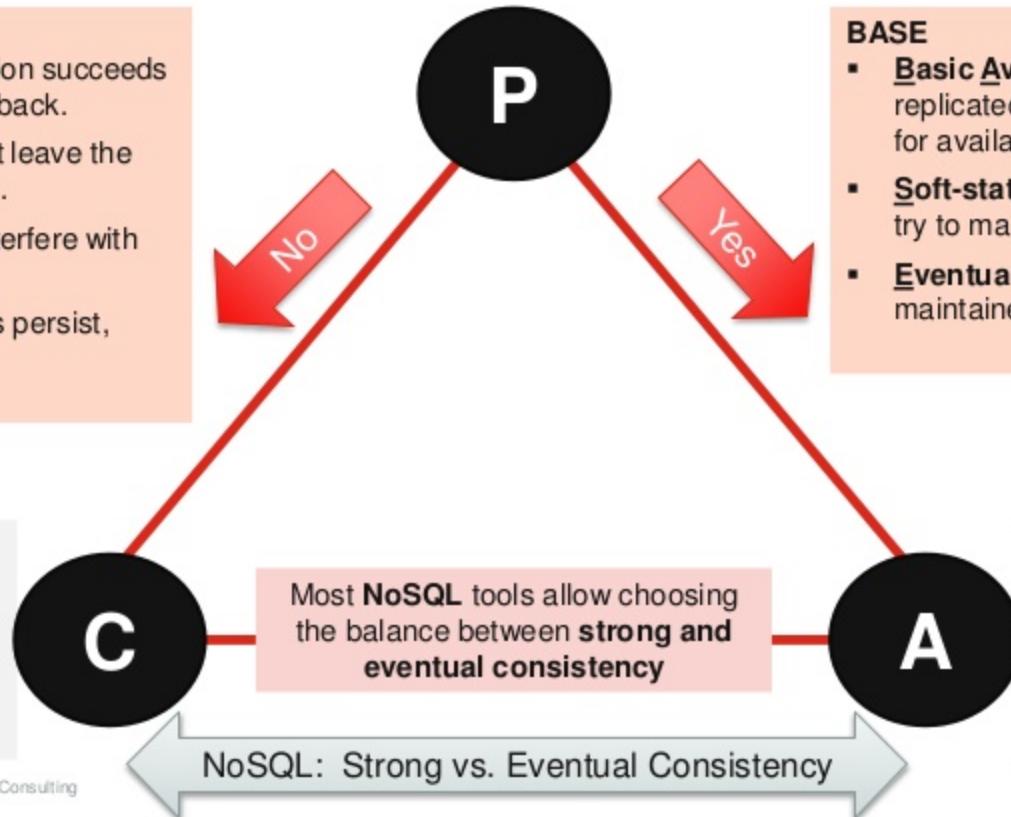
- **Atomic**: Everything in a transaction succeeds or the entire transaction is rolled back.
- **Consistent**: A transaction cannot leave the database in an inconsistent state.
- **Isolated**: Transactions cannot interfere with each other.
- **Durable**: Completed transactions persist, even when servers restart etc.

BASE

- **Basic Availability**: data is sharded and replicated, and consistency is compromised for availability
- **Soft-state**: Allow data to be inconsistent and try to maintain consistency later.
- **Eventual consistency**: Consistency is maintained later.

Transactional RDBMS

- ACID Mode – Strong Consistency
- Commits are atomic across the entire system
- Not partition tolerant, i.e., **sacrifices availability**



Big Data Systems

- BASE Mode – Eventually Consistency
- Remains available (operational & responsive)
- partition tolerant, i.e., **sacrifices consistency**

NoSQL Data Stores

NoSQL Data Stores

Categories and Breads

**Key/Value
Store**

**Column Family
Store**

**Document
Store**

**Graph
Store**

**Memory Cache
Store**

Others

NoSQL – Key/Value Stores

The agile...

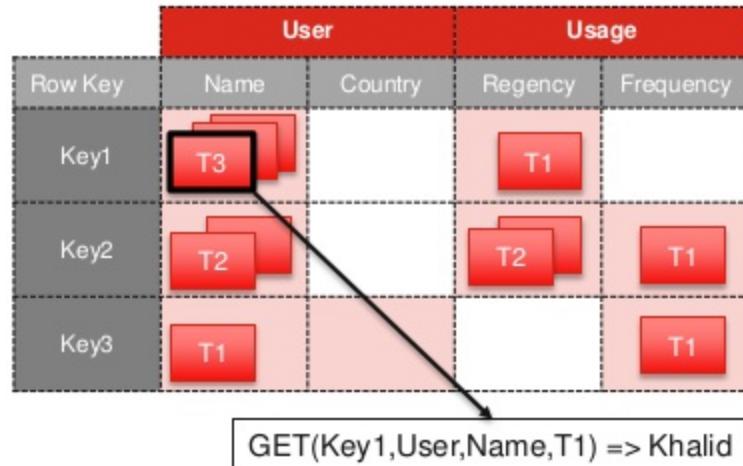
- Based on a **Hash Table** (Dictionary);
- **Unique key** and a pointer to a data (Value)
- Data Item can be **anything** (Blob).
- A key can only have one value.
- Simple APIs (GET, PUT, DELETE).
- Yet complex implementation (**no querying language**).
- Optimized for singleton operations.
- Example tools:
 - Amazon Dynamo (pioneer)
 - Apache Accumulo
 - Riak KV
 - Redis (memory cache)
 - Memcached (memory cache)

	Key	Value
Image name	image-12345.jpg	Binary image file
Web page URL	http://www.example.com/my-web-page.html	HTML of a web page
File path name	N:/folder/subfolder/myfile.pdf	PDF document
MD5 hash	9e107d9d372bb6826bd81d3542a419d6	The quick brown fox jumps over the lazy dog
REST web service call	view-person?person-id=12345&format=xml	<Person><id>12345</id></Person>
SQL query	SELECT PERSON FROM PEOPLE WHERE PID="12345"	<Person><id>12345</id></Person>

NoSQL – Column Family Stores

The big...

- **Extensible Record stores - Wide Column Store**
- **Tabular Data Structure**
- Columns can be **extended** (organized in column family)
- Resulting in **Sparse Matrix**
- Millions of rows and Hundred of thousands of Columns
- APIs (GET, PUT, SCAN, DELETE, COUNT).
- A data item is accessed via (row key, column, timestamp)
- Example tools:
 - Google BigTable (pioneer)
 - Apache HBase
 - Apache Cassandra



Real-world Scenarios

- Website usage information in Google Analytics
- Geographic information in Google Maps
- Social Media Apps (Twitter, Facebook, etc..)
- Search Engine Web Crawling Results

NoSQL – Document Stores

The popular...

Not a MS Word doc!

- Same as Key/Value store, where the value is a **document** with a specific format. E.g. **JSON**, **BSON**, **XML**, etc.
- A document is retrieved by a **key**, or
- Since the format of the documents is known (e.g. JSON), a **query language** can be used to retrieve documents.
- E.g. retrieve all the document that has attribute “country” and its value equals “Algeria”.
- Attributes in side the document are **indexed**. Documents are often **versioned**.
- Supports wide range of **OLTP/Real-time** applications
- Example Tools
 - IBM lotus Nots (pioneer)
 - MongoDB
 - CouchDB - Apache Couchbase
 - Microsoft Azure DocumentDB

Key	Document
John	<pre>{ "name": "John", "country": "Canada", "age": 43, "lastUse": "March 4, 2014" }</pre>
Eva	<pre>{ "name": "Eva", "country": "Germany", "age": 25 }</pre>
Lou	<pre>{ "name": "Lou", "country": "Australia", "age": 51, "firstUse": "May 8, 2013" }</pre>
Total	<pre>{ "docCount": 3, "last": "May 1, 2014" }</pre>

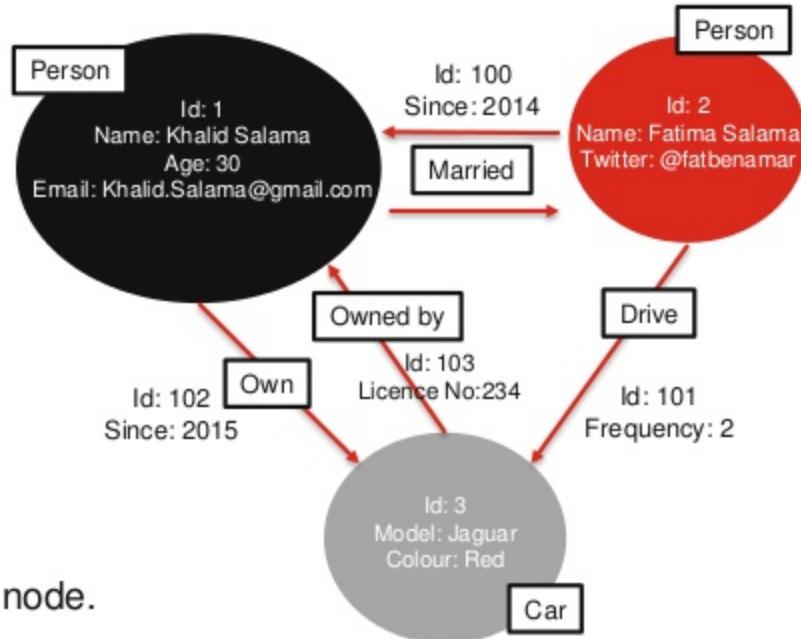
Real-world Scenarios

- Content Management and Personalization
- User Data Management
- Reference Data Management
- Lookups for Real-time Operational Intelligence

NoSQL – Graph Stores

The clever...

- Represent data in **graphical structures**: Nodes and Edges.
- **Nodes** represent **entities**, **Edges** represent **relationships** between entities.
- Relationships are **directed**, semantics of the direction is up to the application. E.g. “Married” is reflexive, “Owns” is not.
- Each Node/Edge has a set of **Key/Value properties**
- Each Node/Edge has a **label** (type of entity/relationship)
- **Optimized** to process **graph-related queries**,
E.g. the number of steps needed to get from one node to another node.
- Example Tools
 - Neo4j
 - OrientDB
 - Titan
 - Apache Giraph
 - Microsoft Graph Engine (Trinity)



Real-world Scenarios

- Social Networks
- Network and IT Operations
- Fraud Detection
- Digital Assets Management

NoSQL – Graph Stores

The clever...

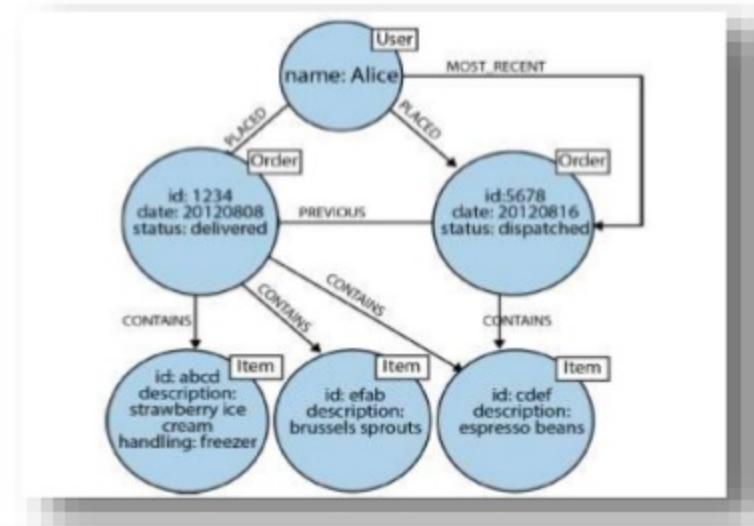
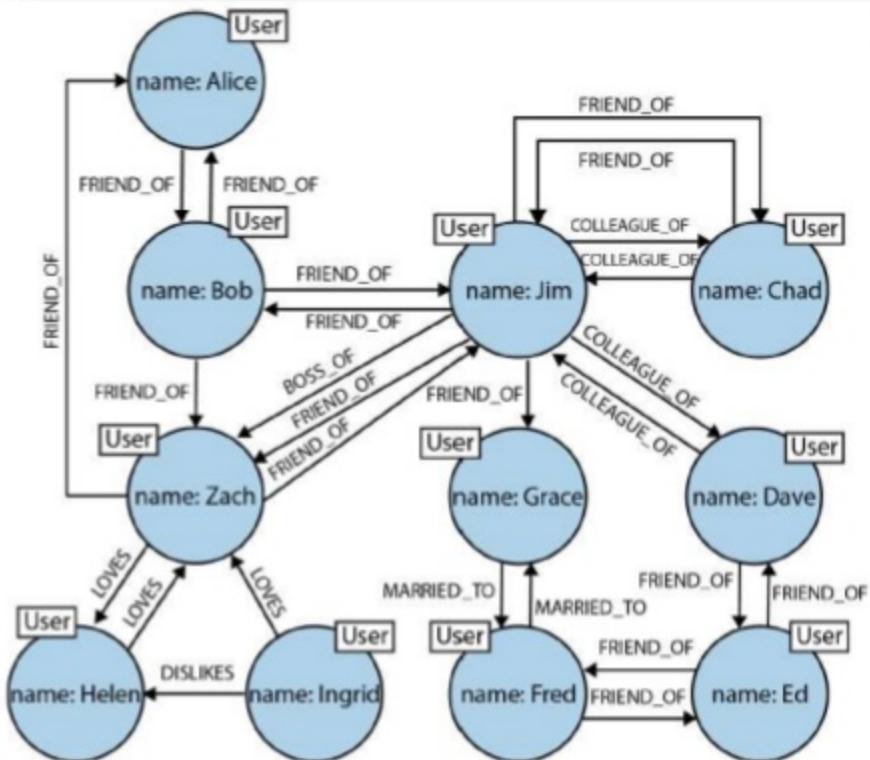


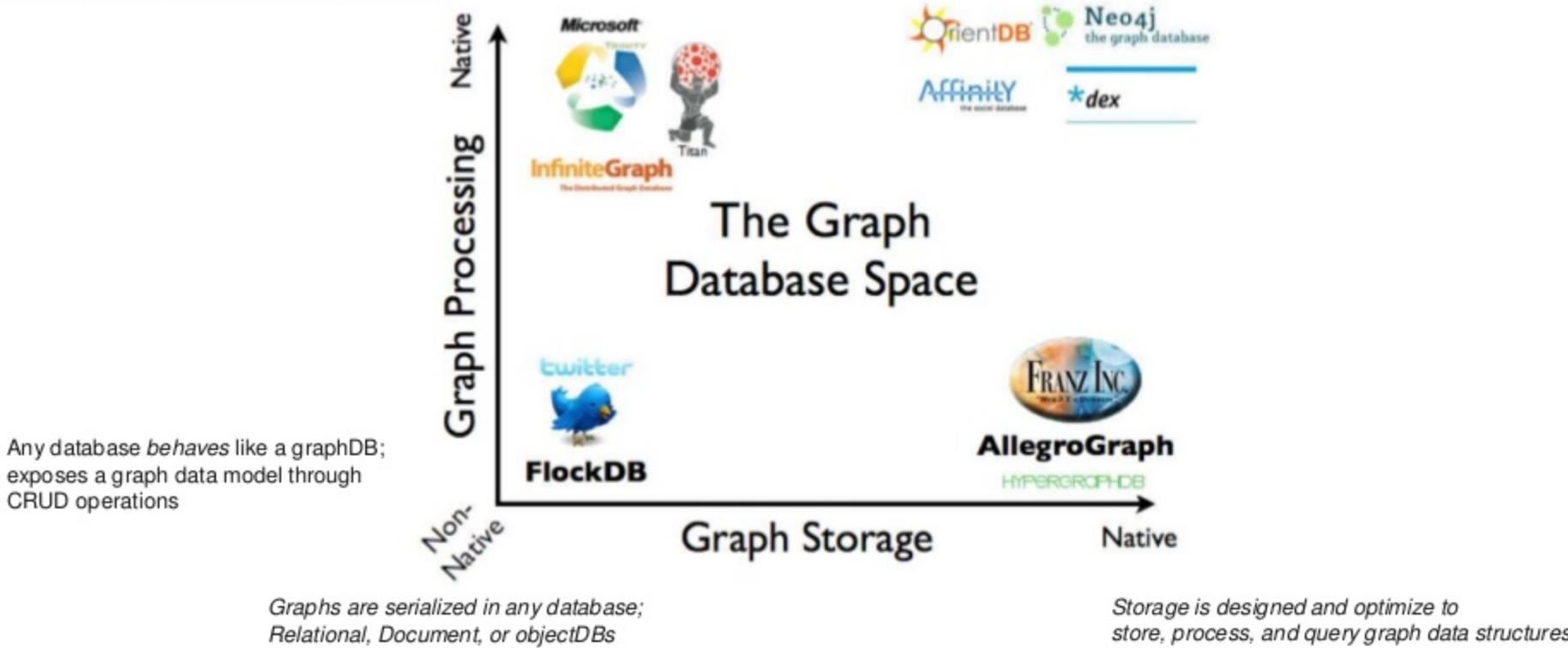
Table 2-1. Finding extended friends in a relational database versus efficient finding in Neo4j

Depth	RDBMS execution time(s)	Neo4j execution time(s)	Records returned
2	0.016	0.01	~2500
3	30.267	0.168	~110,000
4	1543.505	1.359	~600,000
5	Unfinished	2.132	~800,000

NoSQL – Graph Stores

The clever...

index-free adjacency; connected nodes physically “point” to each other in the database



NoSQL – Memory Cache Stores

The fast....

- Usually a **simple data structure** (Key/Value)
- The value can be a simple data type (String) or complex data objects.
- A **memory-based** store, usually, with **persistence** option.
- Used to optimize “hot” data access.
- Manages distributed web application session states.
- Can help to survive service downtime.
- Evolves **read/write strategies** (write-through, write-behind, etc.),
data expiration, and **conflict resolution** techniques.
- Example Tools:
 - Memcached (pioneer)
 - Redis

NoSQL – Other Data Stores

Storage for other data structures

Object-oriented databases

- Developed in the 1980s motivated by the common use of **object-oriented programming**.
- Simply store the objects in a database in a way that corresponds to their representation in the application, without the need of conversion or decomposition.
- The relationships between the objects, e.g. inheritance should also be maintained in the database.
- **Examples:** Caché, Db4o, Versant Object Database

Resource Description Framework (RDF) Data stores – Triple Data Stores

- Originally developed for describing metadata of IT resources.
- Used in connection with the semantic web, and other applications.
- The RDF model represents information as **triples** in the form of **subject-predicate-object**.
- **Examples:** MarkLogic, Virtuoso, Jena, Sesame, Algebraix

Multi-dimensional databases, Multi-value databases, Time-series databases, Event Sourcing databases, Multi-modal data stores, etc.

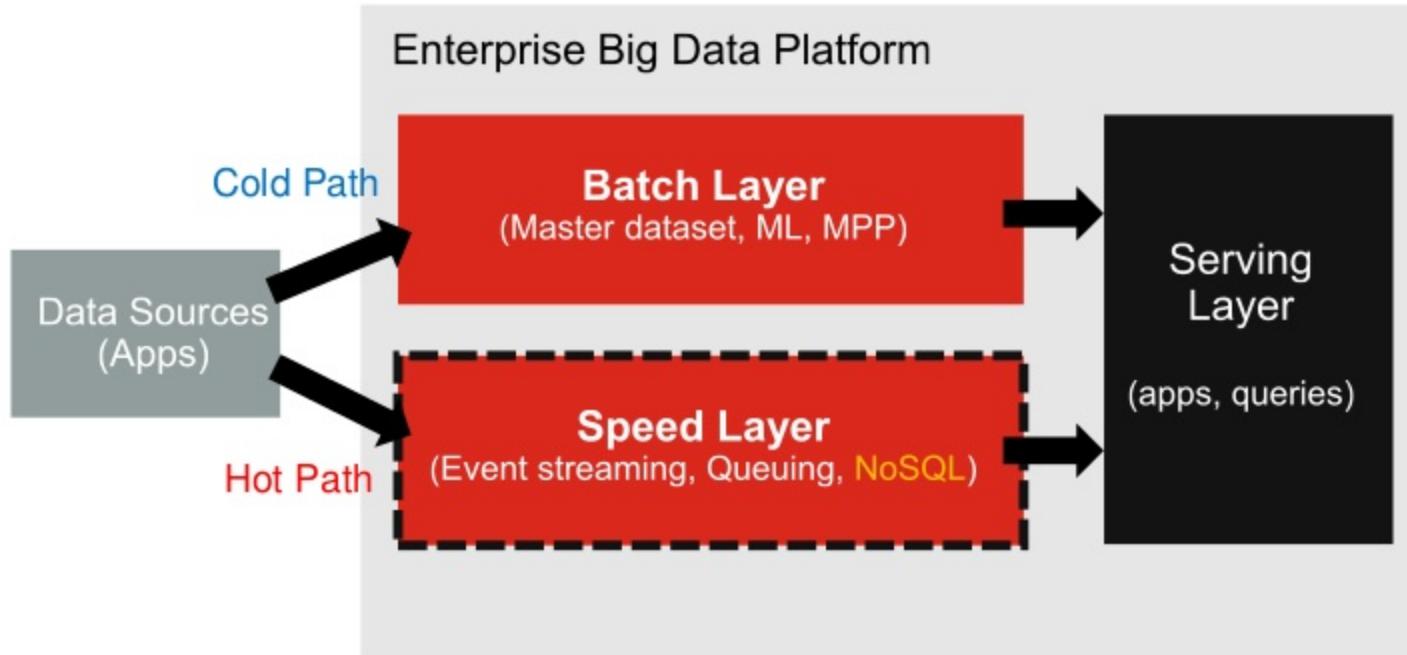
NoSQL Tools & Technologies

	Typical Usage	Lineage	Tools
Key/Value	<ul style="list-style-type: none"> ▪ Flexible data structure ▪ Dictionary/ lookup ▪ Value can be anything 	Amazon's Dynamo	<ul style="list-style-type: none"> ▪ Apache Accumulo ▪ Riak KV ▪ Redis
Column Family (Tabular/wide column)	<ul style="list-style-type: none"> ▪ Column-oriented access ▪ BigData with real-time read/write random access ▪ Extensible 	Google's BigTable	<ul style="list-style-type: none"> ▪ Apache Cassandra ▪ Apache HBase ▪ HBase on Microsoft HDInsight
Document	<ul style="list-style-type: none"> ▪ Query-able data ▪ Objects (complex structure) in JSON, BSON, XML, etc. ▪ CRUD apps 	IBM's Lotus Notes	<ul style="list-style-type: none"> ▪ MongoDB ▪ CouchDB ▪ Apache CouchBase ▪ Microsoft Azure DocumentDB
Graph	<ul style="list-style-type: none"> ▪ Social networks ▪ Fraud detection ▪ Relationship-heavy data 	Graph Theory	<ul style="list-style-type: none"> ▪ Neo4j ▪ OrientDB ▪ Titan ▪ Apache Giraph ▪ Microsoft Graph Engine
Memory Cache	<ul style="list-style-type: none"> ▪ Non-durable data ▪ Fast access 	LiveJournal's Memcached	<ul style="list-style-type: none"> ▪ Redis ▪ Microsoft Azure Redis ▪ Microsoft Azure Memcached (Preview)

Solution Architecture Patterns

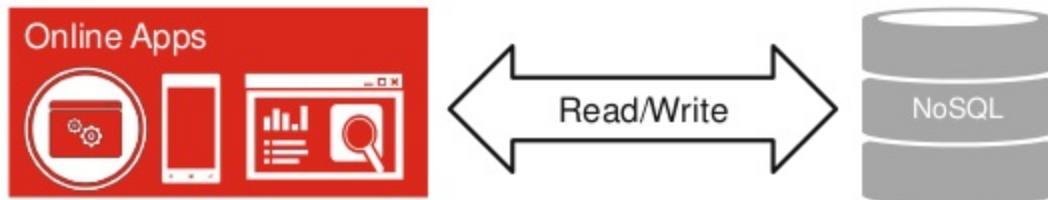
Lambda Architecture

NoSQL and Speed Layer



NoSQL Usage Patterns

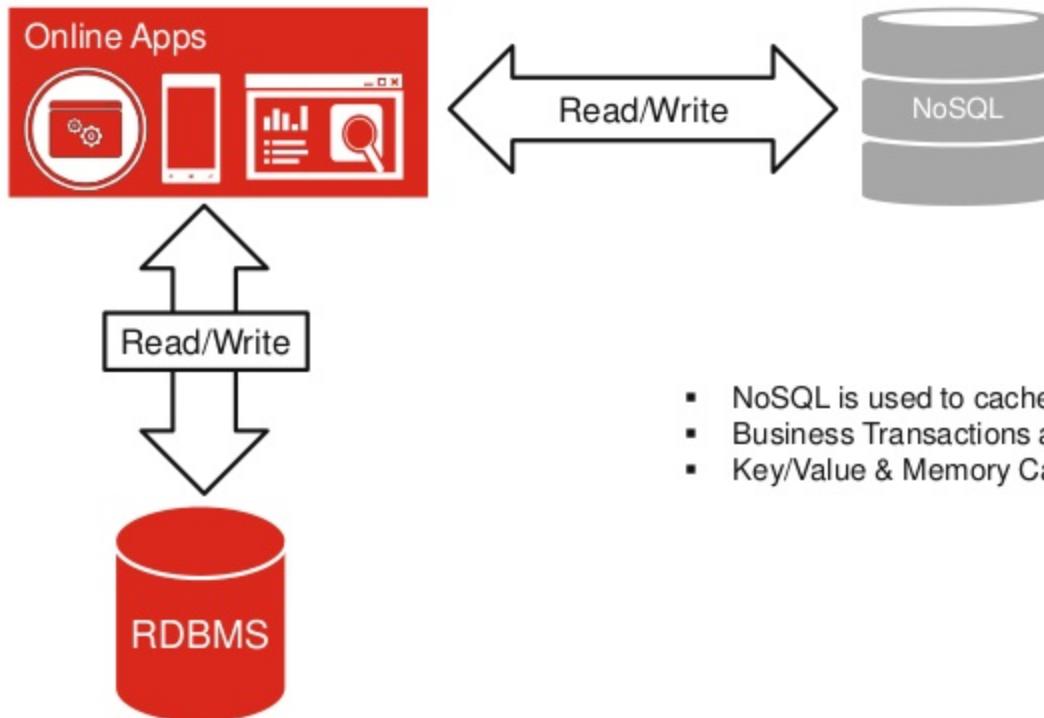
Common Scenarios



- Apps with NoSQL Backend data store
- High Throughputs
- Scalability and Availability
- Column Family, Graph & Document stores

NoSQL Usage Patterns

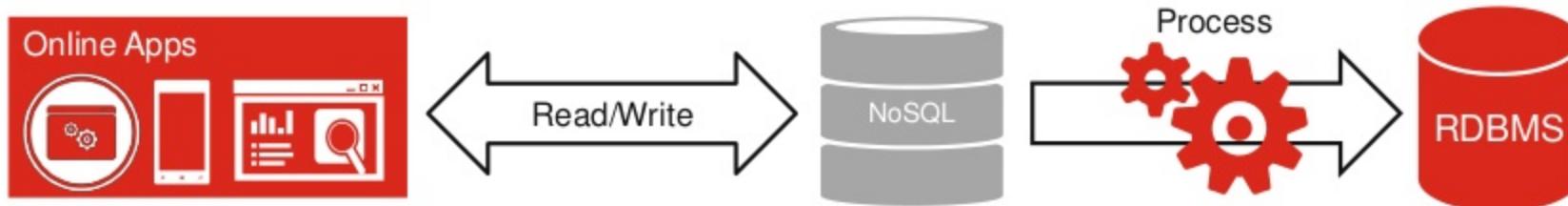
Common Scenarios



- NoSQL is used to cache web content, personalization, reference data
- Business Transactions are invoked and stored into RDBMS
- Key/Value & Memory Cache stores

NoSQL Usage Patterns

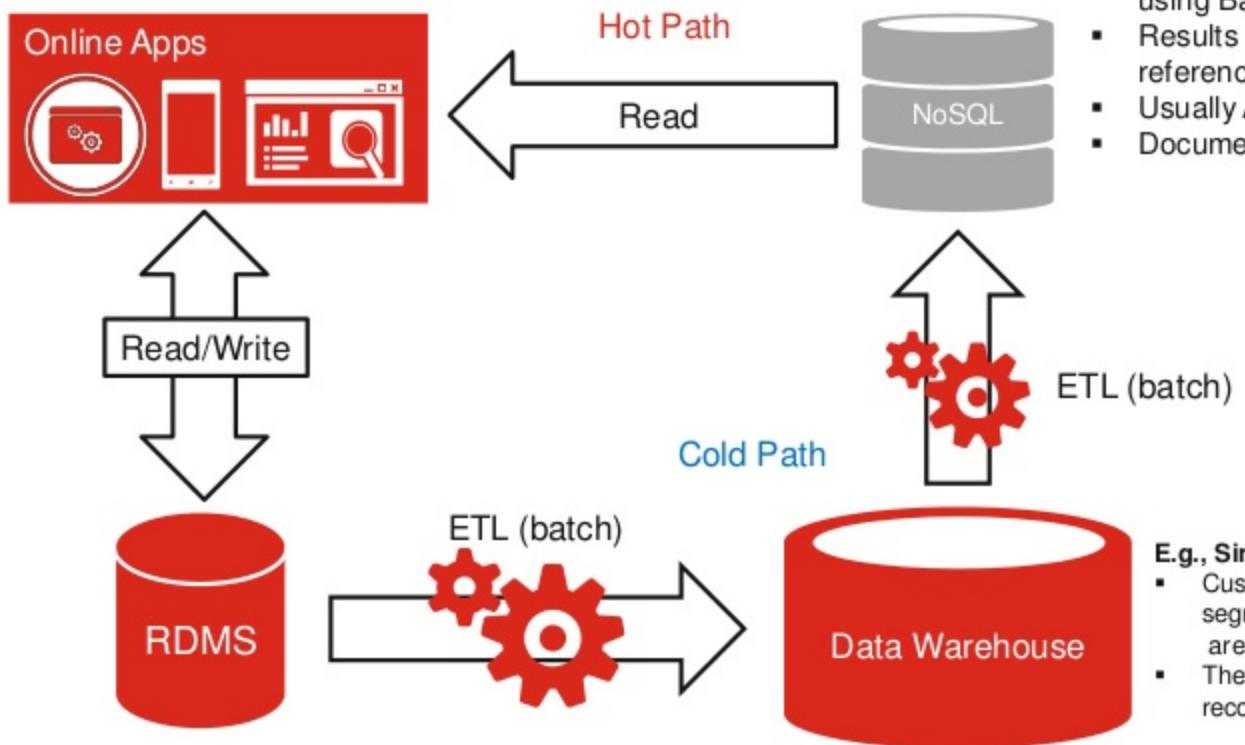
Common Scenarios



- NoSQL stores in-progress transactions/activities (i.e., purchase basket, forms, user session, etc.)
- When transactions are submitted, they are processed into a RDBMS
- Document Stores

NoSQL Usage Patterns

Common Scenarios



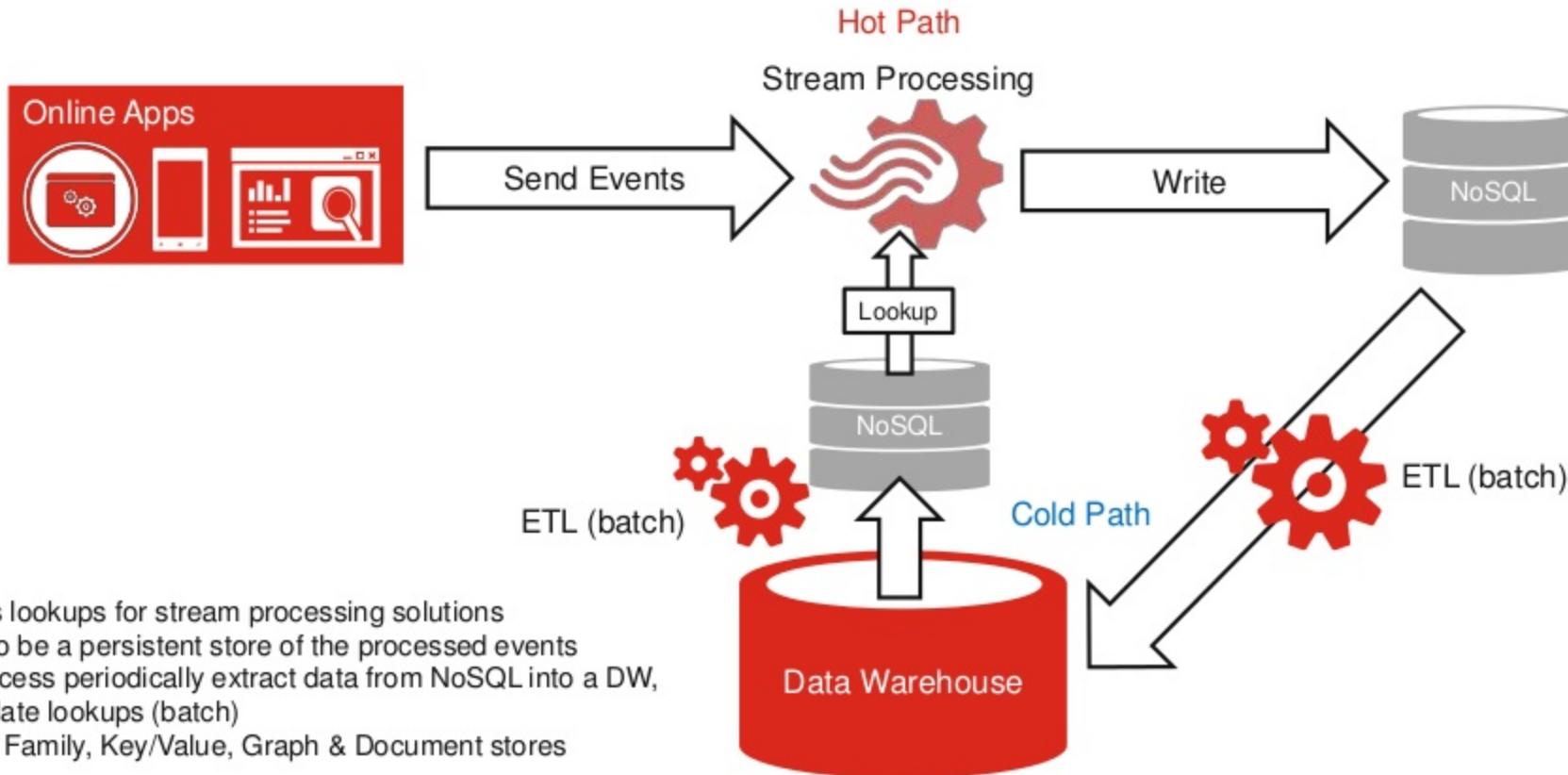
- Data is Extracted, Transformed, loaded from OLTPs to a DW
- Aggregations, KPIs, and scores are computed in using Batch Processing
- Results are populated to a NoSQL data store for reference use in apps
- Usually App hot read and ETL batch Write
- Document & Graph stores

E.g., Single Customer View:

- Customer Matching, customer KPIs, segment assignment, and propensity scoring are performed as batch processing in DW
- The output goes to NoSQL to be used for real-time recommendation, campaigning, targeted advertising, etc.

NoSQL Usage Patterns

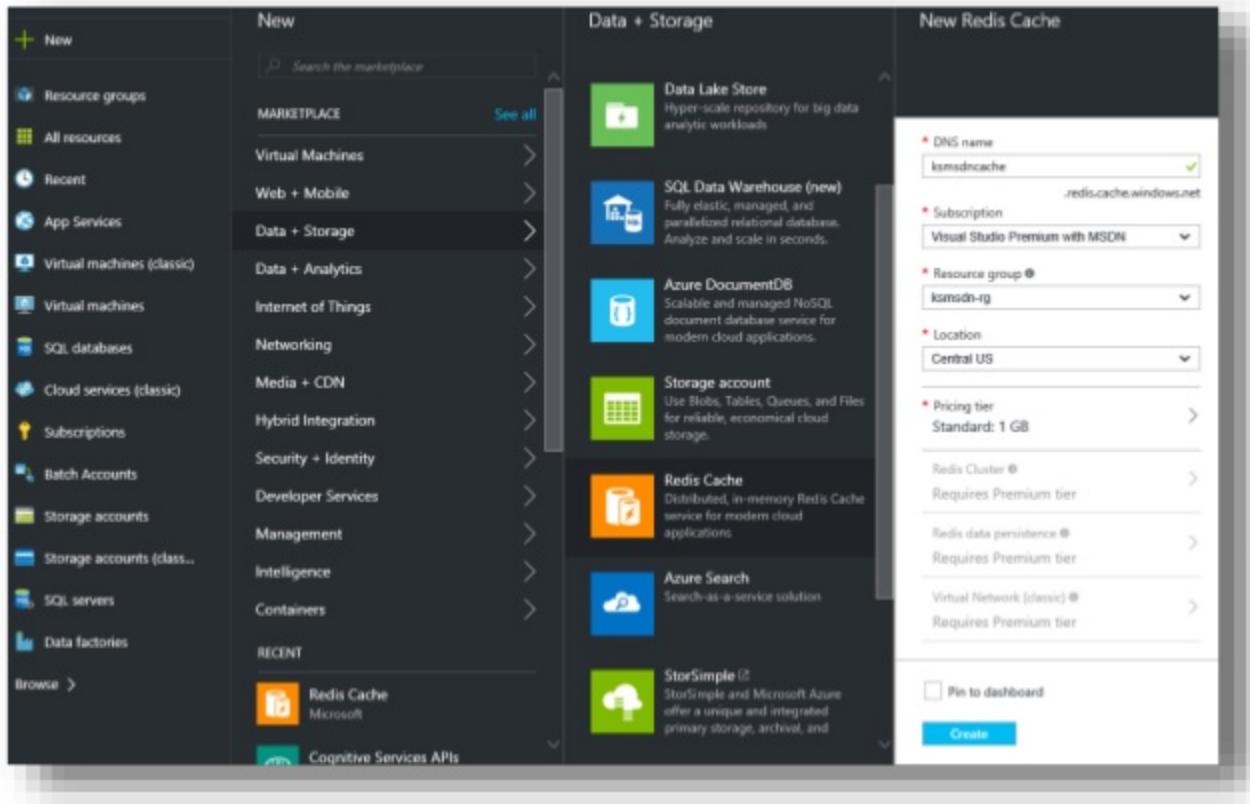
Common Scenarios



Azure Redis Cache

Azure Redis Cache

A Key/Value in memory store – a.k.a Data Structures Store

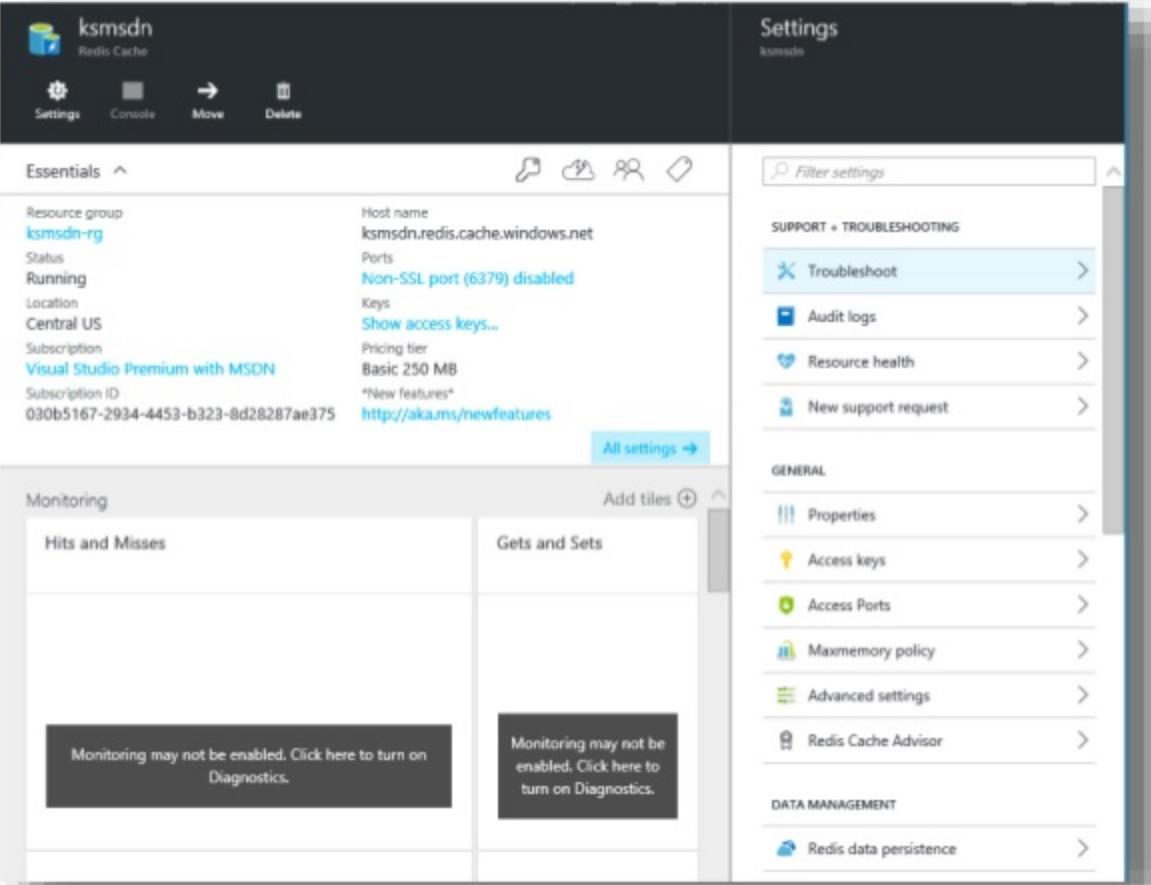


The screenshot shows the Azure portal's 'New' blade. In the 'Data + Storage' category, the 'Redis Cache' service is selected. The 'New Redis Cache' configuration page is displayed, prompting for a DNS name (ksmsdncache), subscription (Visual Studio Premium with MSDN), resource group (ksmsdn-rg), location (Central US), and pricing tier (Standard: 1 GB). Other options like Redis Cluster and Persistence are listed as requiring the Premium tier. A 'Create' button is at the bottom.

C0 Basic	C1 Standard	P1 Premium
250 MB Cache	1 GB Cache	6 GB Cache
Low network bandwidth	Low network bandwidth	Moderate network...
Shared infrastructure	Dedicated service	All Standard features
SSL	SSL	Data Persistence
Up to 256 connections	Up to 1,000 connections	Virtual Network
Configure Redis (key-value)	Configure Redis (key-value)	Redis Cluster
99.9% SLA	99.9% SLA	99.9% SLA
10.00 GBP/MONTH (ESTIMATED)	62.72 GBP/MONTH (ESTIMATED)	252.25 GBP/MONTH (ESTIMATED) PER SH...

Azure Redis Cache

A Key/Value *in memory store* – a.k.a *Data Structures Store*



The screenshot shows the Azure Redis Cache management interface. On the left, the 'Essentials' blade displays basic resource information:

Resource group	ksmsdn-rg	Host name	ksmsdn.redis.cache.windows.net
Status	Running	Ports	Non-SSL port (6379) disabled
Location	Central US	Keys	Show access keys...
Subscription	Visual Studio Premium with MSDN	Pricing tier	Basic 250 MB
Subscription ID	030b5167-2934-4453-b323-8d28287ae375	*New features*	http://aka.ms/newfeatures

On the right, the 'Settings' blade provides detailed configuration options:

- SUPPORT + TROUBLESHOOTING**: Troubleshoot, Audit logs, Resource health, New support request.
- GENERAL**: Properties, Access keys, Access Ports, Maxmemory policy, Advanced settings, Redis Cache Advisor.
- DATA MANAGEMENT**: Redis data persistence.

Both blades include sections for Monitoring (Hits and Misses, Gets and Sets) with a note: "Monitoring may not be enabled. Click here to turn on Diagnostics."

Azure Redis Cache

A Key/Value in memory store – a.k.a Data Structures Store

In the example, Radis caches sql query (key), and query xml result (value)

```
using StackExchange.Redis;
```

```
private static Lazy<ConnectionMultiplexer> lazyConnection = new Lazy<ConnectionMultiplexer>(() =>
{
    return ConnectionMultiplexer.Connect(RedisConnectionString);
});

public static ConnectionMultiplexer Connection
{
    get
    {
        return lazyConnection.Value;
    }
}
```

```
private DataTable GetDataFromCache(string sqlQuery)
{
    DataSet ds = new DataSet();
    DataTable tbResults = null;

    IDatabase cache = Connection.GetDatabase();

    string xmlContent = null;
    if (!cache.KeyExists(sqlQuery))
    {
        tbResults = this.GetDataFromDatabase(sqlQuery);

        using (StringWriter sw = new StringWriter())
        {
            tbResults.WriteXml(sw);
            xmlContent = sw.ToString();
        }

        cache.StringSet(sqlQuery, xmlContent);
    }

    xmlContent = cache.StringGet(sqlQuery);

    using(StringReader sr= new StringReader(xmlContent))
    {
        ds.ReadXml(sr);
        tbResults = ds.Tables[0];
    }

    return tbResults;
}
```

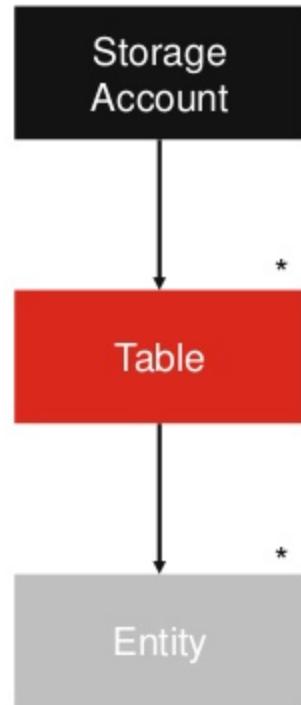
You can also store hash tables, lists, and sorted lists

Azure Table Storage

Azure Table Storage

A Key/Value NoSQL Store

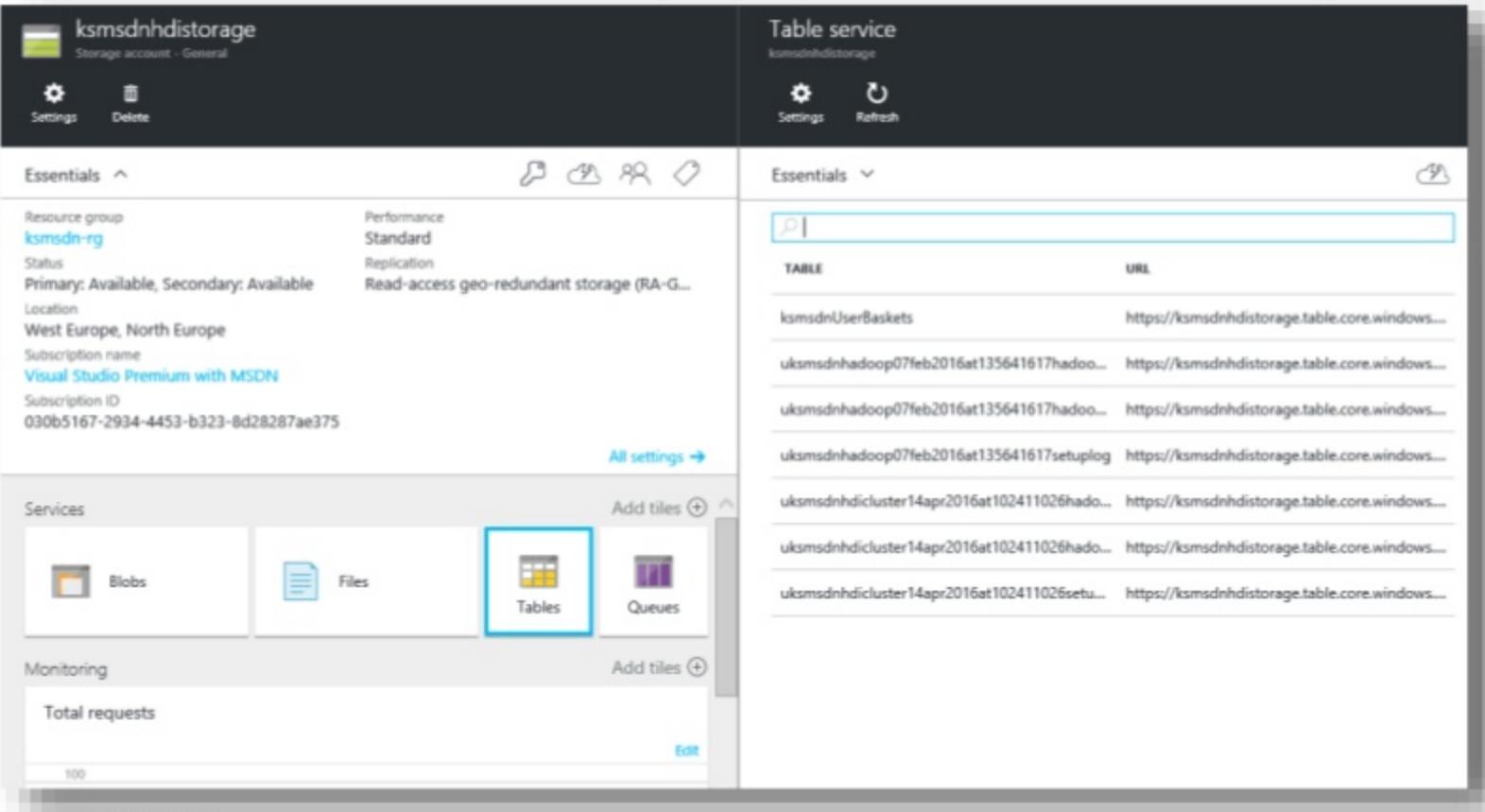
- Key/attribute store with a schema-less design.
- Adapts your data as the needs of your application evolve.
- Access to data is fast and cost-effective for all kinds of applications.
- Significantly lower in cost than traditional SQL for similar volumes of data.
- Web applications, address books, device information, metadata, etc.
- Row Key and Partition Key must be defined for the entity
- No complex joins, foreign keys, or stored procedures.
- Supports OData protocol and LINQ with WCF Data Service .NET Libraries.



- Partition Key
- Row Key

Azure Table Storage

A Key/Value NoSQL Store



The image shows two side-by-side screenshots of the Azure portal interface.

Left Screenshot: Storage Account Overview

Storage account: ksmsdnhdstorage

General Settings:

- Resource group:** ksmsdn-rg
- Status:** Primary: Available, Secondary: Available
- Location:** West Europe, North Europe
- Subscription name:** Visual Studio Premium with MSDN
- Subscription ID:** 030b5167-2934-4453-b323-8d28287ae375

Services: Blobs, Files, **Tables** (selected), Queues

Monitoring: Total requests: 100

Right Screenshot: Table Service Overview

Table service: ksmsdnhdstorage

Essentials:

- Performance:** Standard
- Replication:** Read-access geo-redundant storage (RA-GRS)

Tables:

TABLE	URL
ksmsdnUserBaskets	https://ksmsdnhdstorage.table.core.windows.net/ksmsdnUserBaskets
uksmsdnhadoop07feb2016at135641617hadoop...	https://ksmsdnhdstorage.table.core.windows.net/uksmsdnhadoop07feb2016at135641617hadoop...
uksmsdnhadoop07feb2016at135641617hadoop...	https://ksmsdnhdstorage.table.core.windows.net/uksmsdnhadoop07feb2016at135641617hadoop...
uksmsdnhadoop07feb2016at135641617setuplog	https://ksmsdnhdstorage.table.core.windows.net/uksmsdnhadoop07feb2016at135641617setuplog
uksmsdnhdcluster14apr2016at102411026hado...	https://ksmsdnhdstorage.table.core.windows.net/uksmsdnhdcluster14apr2016at102411026hado...
uksmsdnhdcluster14apr2016at102411026hado...	https://ksmsdnhdstorage.table.core.windows.net/uksmsdnhdcluster14apr2016at102411026hado...
uksmsdnhdcluster14apr2016at102411026setu...	https://ksmsdnhdstorage.table.core.windows.net/uksmsdnhdcluster14apr2016at102411026setu...

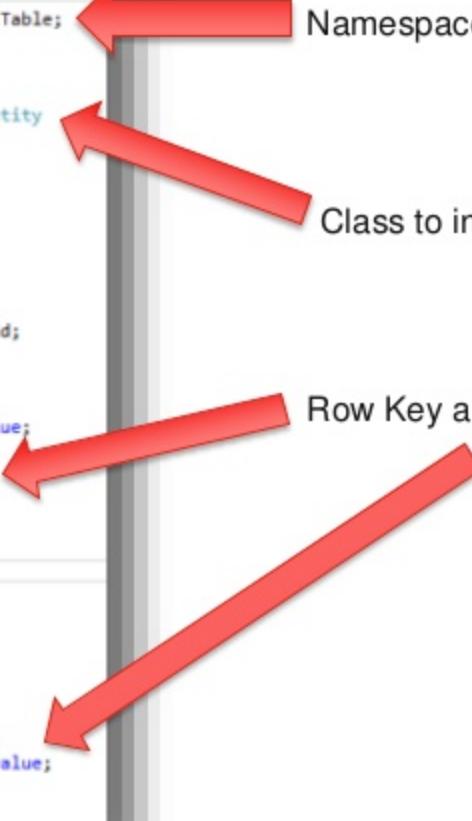
Azure Table Storage

A Key/Value NoSQL Store

```
using Microsoft.WindowsAzure.Storage.Table;           ← Namespace to use
namespace HCUK.AzureNoSQL.Model
{
    public class UserBasket : TableEntity
    {
        private string _accountId;
        private string _region;

        public string AccountId
        {
            get
            {
                return this._accountId;
            }
            set
            {
                this._accountId = value;
                this.RowKey = value;
            }
        }

        public string Region
        {
            get
            {
                return this._region;
            }
            set
            {
                this._region = value;
                this.PartitionKey = value;
            }
        }
    }
}
```



Azure Table Storage

A Key/Value NoSQL Store

If a Table Entity has an attribute that is a collection or object, Table Storage will ignore it.
Thus, needs to serialized/ de-serialized

```
public List<Item> Items
{ get; set; }

[Newtonsoft.Json.JsonIgnore]
public string ItemString
{
    get
    {
        return String.Join("|", this.Items.Select(t => t.ToString()).ToArray());
    }
    set
    {
        this.Items = new List<Item>();
        foreach (var itemString in value.Split('|'))
        {
            var elements = itemString.Trim('(', ')').Split(',');
            this.Items.Add(new Item() { ItemId = elements[0], Quantity = int.Parse(elements[1]), Value = double.Parse(elements[2]) });
        }
    }
}
```

Azure Table Storage

A Key/Value NoSQL Store

```

CloudTable _azureTable;

public AzureTableStorageServices(string accountName, string keyValue, string tableName)
{
    var credentials = new Microsoft.WindowsAzure.Storage.Auth.StorageCredentials(accountName, keyValue);
    CloudStorageAccount storageAccount = new CloudStorageAccount(credentials, false);
    CloudTableClient azureTableclient = storageAccount.CreateCloudTableClient();

    this.TableName = tableName;
    this._azureTable = azureTableclient.GetTableReference(this.TableName);
}

```

```

public T RetrieveEntity(string partitionKey, string rowKey)
{
    TableOperation retrieveOperation = TableOperation.Retrieve<T>(partitionKey, rowKey);
    var result = _azureTable.Execute(retrieveOperation);
    return result.Result as T;
}

```

```

public void RemoveEntity(T entity)
{
    var deleteOperation = TableOperation.Delete(entity);
    this._azureTable.ExecuteAsync(deleteOperation);
}

```

```

public void RemoveEntity(string partitionKey, string rowKey)
{
    var entity = RetrieveEntity(partitionKey, rowKey);

    if (entity != null)
    {
        var deleteOperation = TableOperation.Delete(entity);
        this._azureTable.ExecuteAsync(deleteOperation);
    }
}

```

```

public List<T> RetrieveEntitiesByPartition(string partitionKey)
{
    List<T> results = new List<T>();

    TableQuery<T> query = new TableQuery<T>()
        .Where(TableQuery.GenerateFilterCondition("PartitionKey", QueryComparisons.Equal, partitionKey));

    return results;
}

```

```

public void AddEntity(T entity, InsetBehaviour insertBehavior = InsetBehaviour.Normal)
{
    TableOperation insertOperation = null;

    switch (insertBehavior)
    {
        case InsetBehaviour.Normal: insertOperation = TableOperation.Insert(entity); break;
        case InsetBehaviour.Replace: insertOperation = TableOperation.InsertOrReplace(entity); break;
        case InsetBehaviour.Merge: insertOperation = TableOperation.InsertOrMerge(entity); break;
    }

    _azureTable.Execute(insertOperation);
}

```

```

public void AddEntities(IEnumerable<T> entities)
{
    var partitions = entities.Select(e => e.PartitionKey).Distinct();

    foreach (var partition in partitions)
    {
        TableBatchOperation batch = new TableBatchOperation();
        foreach (var entity in entities.Where(e => e.PartitionKey == partition))
            batch.Add(TableOperation.InsertOrReplace(entity));
        _azureTable.ExecuteBatchAsync(batch).Wait();
    }
}

```

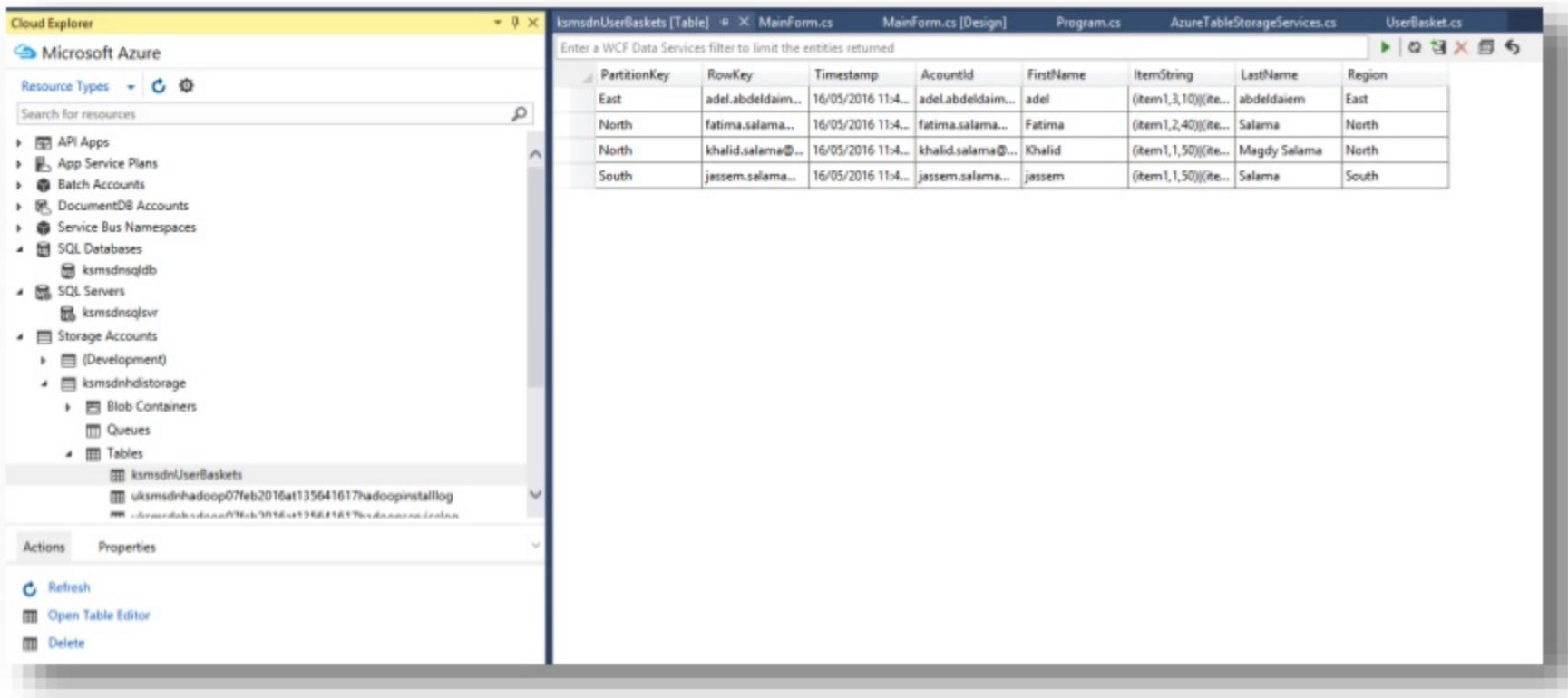
```

public static List<UserBasket> GetUserBasketByCondition
(CloudTable azureTable, string filterCondition)
{
    TableQuery<UserBasket> query = new TableQuery<UserBasket>()
        .Where(filterCondition);
    return azureTable.ExecuteQuery<UserBasket>(query).ToList();
}

```

Azure Table Storage

A Key/Value NoSQL Store



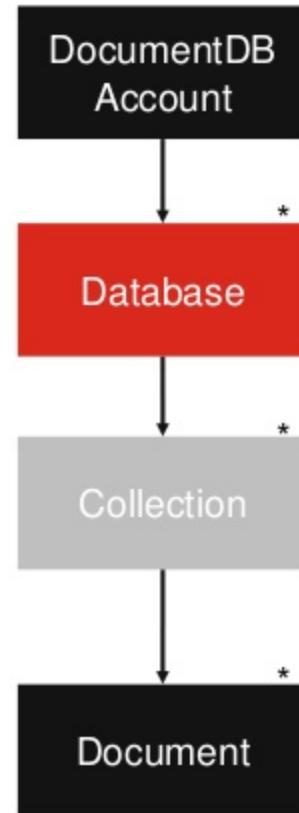
PartitionKey	RowKey	Timestamp	AcountId	FirstName	ItemString	LastName	Region
East	adel.abdeldaim...	16/05/2016 11:4...	adel.abdeldaim...	adel	(item1,3,100)(ite...	abdeldaiem	East
North	fatima.salama...	16/05/2016 11:4...	fatima.salama...	Fatima	(item1,2,40)(ite...	Salama	North
North	khalid.salama@...	16/05/2016 11:4...	khalid.salama@...	Khalid	(item1,1,50)(ite...	Magdy Salama	North
South	jassem.salama...	16/05/2016 11:4...	jassem.salama...	jassem	(item1,1,50)(ite...	Salama	South

Azure DocumentDB

Azure DocumentDB

A cloud-based Document store

- JSON Database
- Elastically scalable throughput and storage
- Ad hoc queries with familiar SQL syntax
- JavaScript execution within the database
- Tunable consistency levels
- Fully managed
- Open by design



Azure DocumentDB

Azure DocumentDB Structure



Azure DocumentDB

Getting Started

Marketplace

Data + Storage

Everything

Virtual Machines

Web + Mobile

Data + Storage

Data + Analytics

Internet of Things

Networking

Media + CDN

Hybrid Integration

Security + Identity

Developer Services

Management

Intelligence

Containers

Azure DocumentDB

Results

NAME	PUBLISHER	CATEGORY
Azure DocumentDB	Microsoft	Storage
DocumentDB - Protocol Support for MongoDB (preview)	Microsoft	Storage

Related to your search ▾

 SQL Database Microsoft

 Website + SQL + Redis Cache (preview) Microsoft

 Memcached Cloud (preview) Redis Labs

DocumentDB account

New DocumentDB account

* ID
ksmsdn documents.azure.com

* Subscription
Visual Studio Premium with MSDN

* Resource Group
ksmsdn-rg

* Location
West Europe

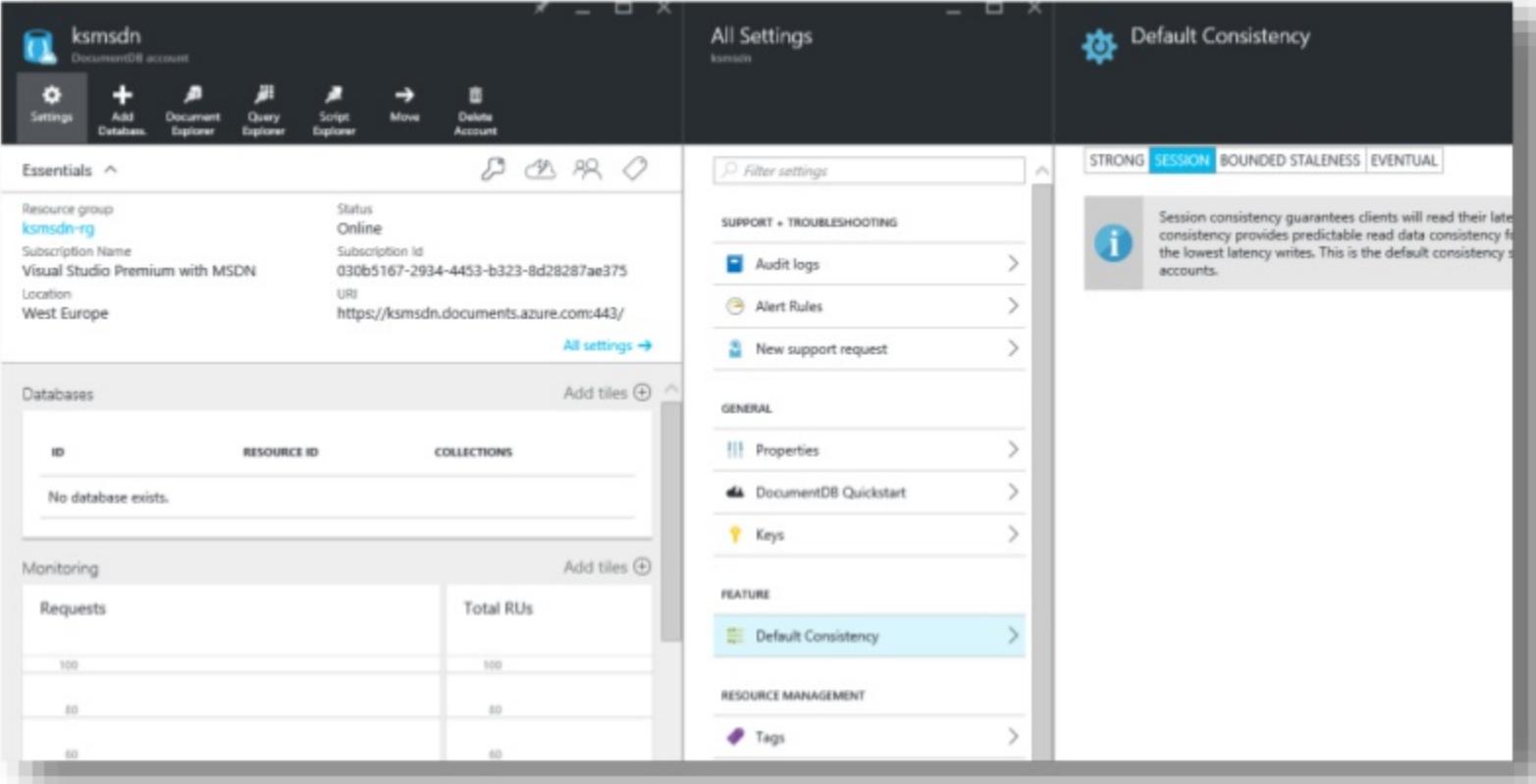
 Provisioning a DocumentDB account will take less than 5 minutes.

Pin to dashboard

Create

Azure DocumentDB

Getting Started



The screenshot shows the Azure DocumentDB portal interface. On the left, the 'Essentials' blade displays account details: Resource group (ksmsdn-rg), Subscription Name (Visual Studio Premium with MSDN), Location (West Europe), and Status (Online). It also shows monitoring data for Requests and Total RUs over time. The 'Databases' blade indicates no database exists. On the right, the 'All Settings' blade is open, specifically showing the 'Default Consistency' section. This section includes tabs for STRONG, SESSION (which is selected), BOUNDED STALENESS, and EVENTUAL. A note states: 'Session consistency guarantees clients will read their latest writes. This is the default consistency for accounts.' Other settings listed include Audit logs, Alert Rules, New support request, Properties, DocumentDB Quickstart, and Keys.

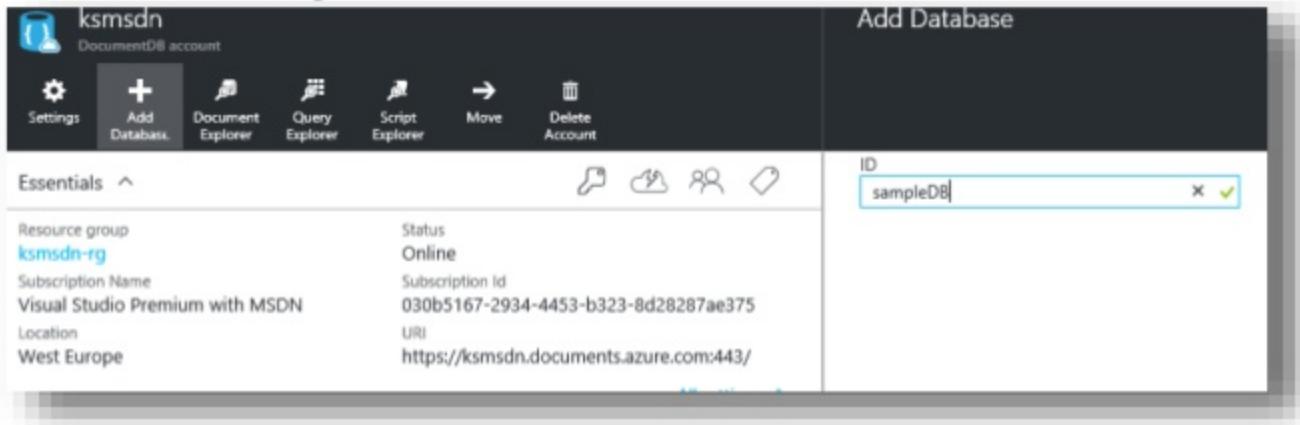
Azure DocumentDB

DocumentDB Consistency Model

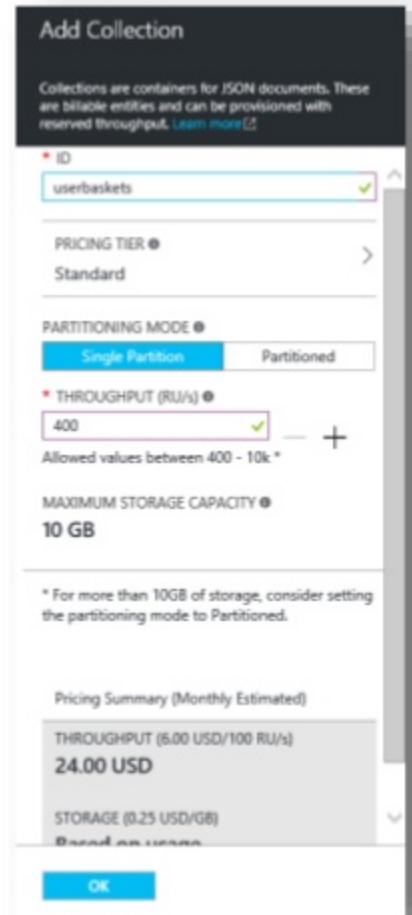
Description	
Strong	The slowest of the four, but is guaranteed to always return correct data.
Session	Ensures that an application always sees its own writes correctly, but allows access to potentially out-of-date or out-of-order data written by other applications.
Bound Staleness	Ensures that an application will see changes in the order in which they were made. This option does allow an application to see out-of-date data, but only within a specified window, e.g., 500 milliseconds.
Eventual	Provides the fastest access, but also has the highest chance of returning out-of-date data.

Azure DocumentDB

Getting Started



The screenshot shows the Azure DocumentDB account management interface. On the left, there's a sidebar with options like Settings, Add Database, Document Explorer, Query Explorer, Script Explorer, Move, and Delete Account. Below that is the 'Essentials' section with details about the resource group (ksmsdn-rg), subscription name (Visual Studio Premium with MSDN), location (West Europe), and URI (https://ksmsdn.documents.azure.com:443/). A central modal window titled 'Add Database' has an 'ID' field containing 'sampleDB'. There are also icons for settings, cloud storage, users, and a clipboard.



This screenshot shows the 'Add Collection' dialog. It includes instructions: 'Collections are containers for JSON documents. These are billable entities and can be provisioned with reserved throughput.' Below are fields for 'ID' (set to 'userbaskets'), 'PRICING TIER' (set to 'Standard'), 'PARTITIONING MODE' (set to 'Single Partition'), 'THROUGHPUT (RU/s)' (set to '400'), and 'MAXIMUM STORAGE CAPACITY' (set to '10 GB'). A note at the bottom states: '* For more than 10GB of storage, consider setting the partitioning mode to Partitioned.' At the bottom right is an 'OK' button.

Azure DocumentDB

Databases and Collections

ksmsdn
DocumentDB account

Settings Add Database Document Explorer Query Explorer Script Explorer Move Delete Account

Essentials ^

Resource group ksmcdn-rg Status

Subscription Name Visual Studio Premium with MSDN

Location West Europe

Add Database

ID sampleDB

sampleDB Database

Add Collection Document Explorer Query Explorer Script Explorer Delete Databases

Collections Add tiles +

ID	PRICING TIER	THROUGHPUT	STATUS
userbaskets	Standard	400	Up to date

Monitoring Add tiles +

Requests	Total RU/s
100	100
80	80
60	60
40	40
20	20
0	0

If storage, consider setting to Partitioned.

Capacity 400 - 10k *
RU/s (0.25 USD/Gb monthly Estimated)
Storage (0.25 USD/Gb Based on usage)

OK

Collections are containers for JSON documents. These are billable entities and can be provisioned with reserved throughput. [Learn more](#)

* ID

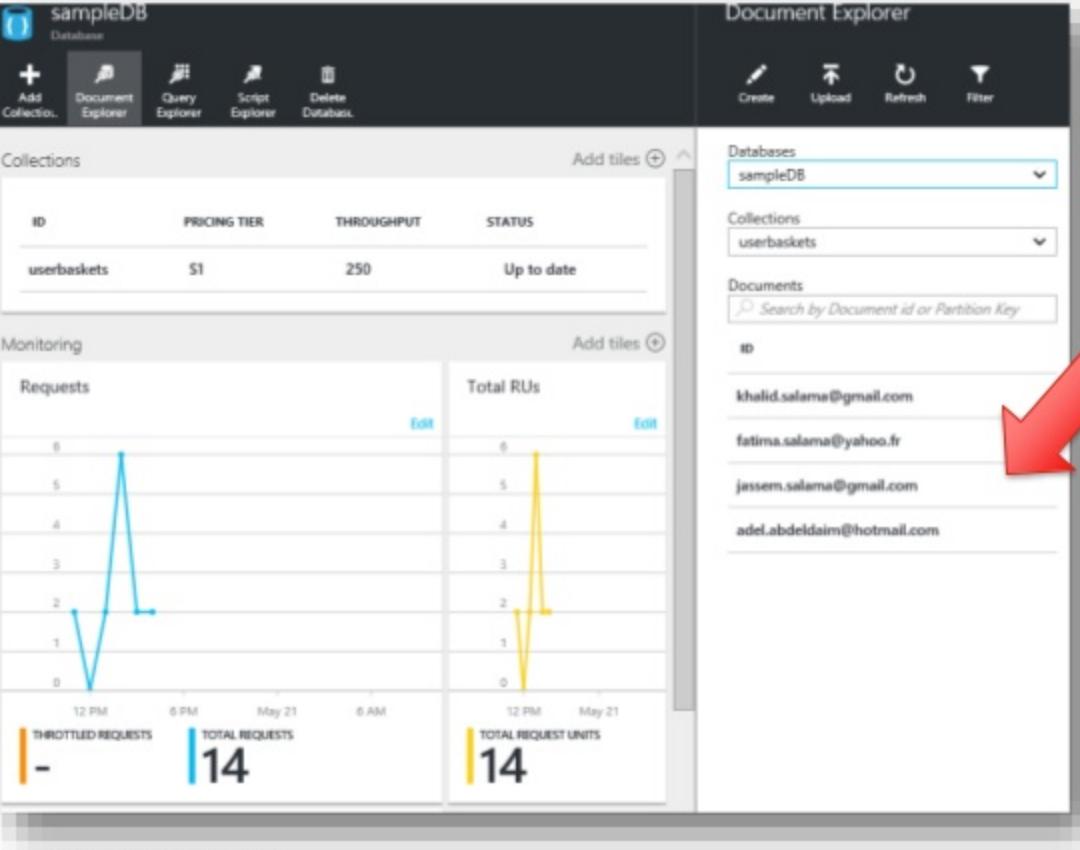
userbaskets

PRICING TIER Standard

Partitioned

Azure DocumentDB

Document Explorer



The screenshot shows the Azure DocumentDB portal interface. On the left, the 'sampleDB' database blade is open, displaying the 'Collections' section with a table showing one collection named 'userbaskets'. Below it is the 'Monitoring' section with two charts: 'THROTTLED REQUESTS' and 'TOTAL REQUESTS' both showing a value of 14, and 'TOTAL REQUEST UNITS' showing a value of 14. On the right, the 'Document Explorer' blade is open, showing the 'Databases' dropdown set to 'sampleDB' and the 'Collections' dropdown set to 'userbaskets'. The 'Documents' section lists several email addresses: khalid.salama@gmail.com, fatima.salama@yahoo.fr, jassem.salama@gmail.com, and adel.abdeldaim@hotmail.com.

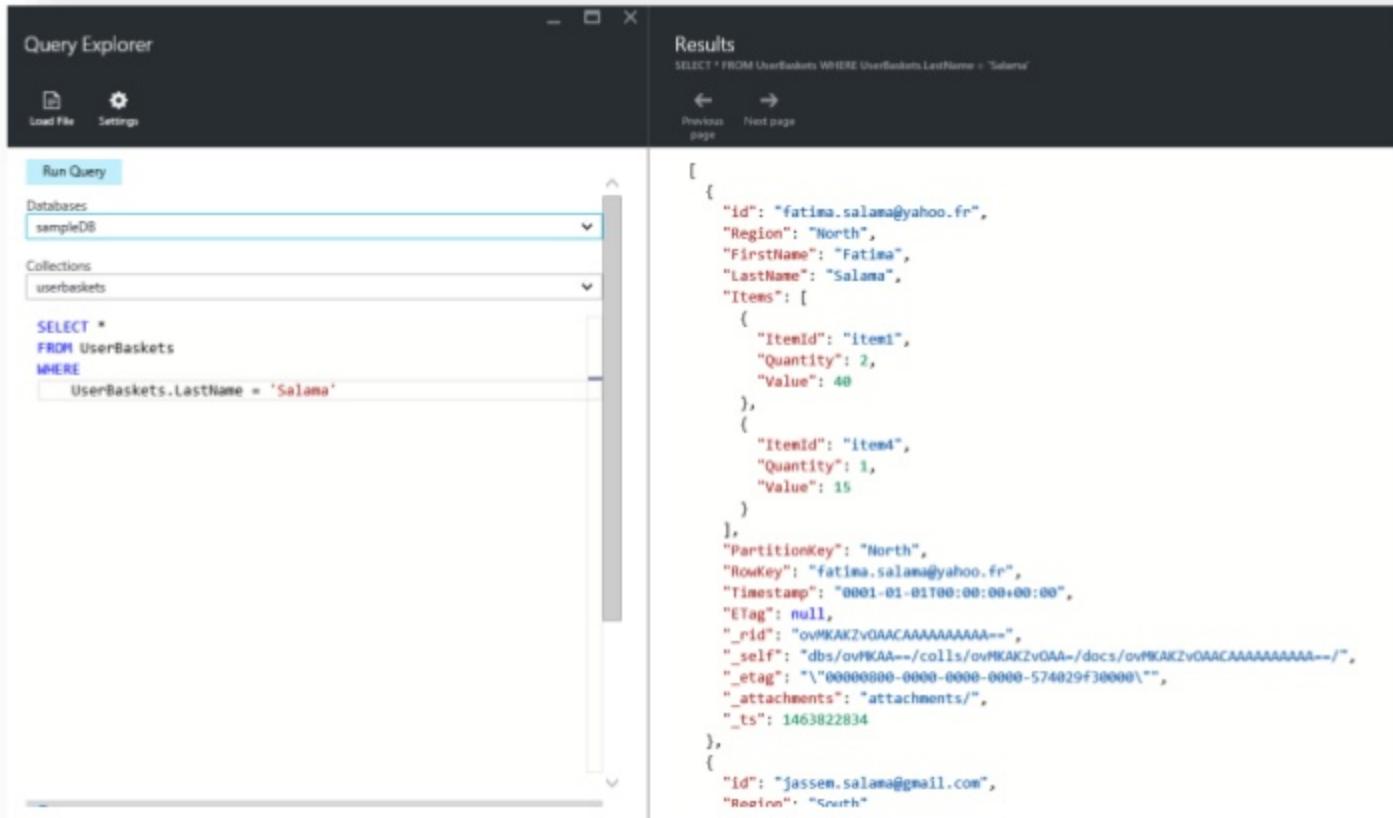
```
public class UserBasket : TableEntity
{
    private string _accountId;
    private string _region;

    [JsonProperty(PropertyName = "id")]
    public string AcountId
    {
        get
        {
            return this._accountId;
        }
        set
        {
            this._accountId = value;
            this.RowKey = value;
        }
    }

    public string Region
    {
        get
        {
            return this._region;
        }
        set
        {
            this._region = value;
            this.PartitionKey = value;
        }
    }
}
```

Azure DocumentDB

Query Explorer



The screenshot shows the Azure DocumentDB Query Explorer interface. On the left, the 'Run Query' button is highlighted. Below it, the 'sampledDB' database is selected in the 'Databases' dropdown, and the 'userbaskets' collection is selected in the 'Collections' dropdown. The query editor contains the following T-SQL code:

```
SELECT *
FROM UserBaskets
WHERE
    UserBaskets.LastName = 'Salama'
```

The results pane displays the query results as JSON documents. One document is shown in full:

```
[{"id": "fatima.salama@yahoo.fr", "Region": "North", "FirstName": "Fatima", "LastName": "Salama", "Items": [{"ItemId": "item1", "Quantity": 2, "Value": 40}, {"ItemId": "item4", "Quantity": 3, "Value": 15}], "PartitionKey": "North", "RowKey": "fatima.salama@yahoo.fr", "Timestamp": "2001-01-01T00:00:00+00:00", "ETag": null, "_rid": "ovMKAKZv0AACAAAAAA==", "self": "dbs/ovMKAA==/colls/ovMKAKZv0AA=/docs/ovMKAKZv0AACAAAAAA==/", "_etag": "\\"00000800-0000-0000-574029f30000\\\"", "attachments": "attachments/", "_ts": 1463822834}, {"id": "jassem.salama@gmail.com", "Region": "South"}]
```

Azure DocumentDB

.NET Code

Microsoft.Azure.Documents.Client.DocumentClient

Includes the following operations,

- Create/Delete
- Read
- Replace/Upsert,

for the following objects,

- Database
- Collection
- Document
- Attachment
- User
- Permission
- USerDefinedFunction
- StoredProcedure
- Trigger

Azure DocumentDB

.NET Code

```
using Microsoft.Azure.Documents;
using Microsoft.Azure.Documents.Client;
using Microsoft.Azure.Documents.Linq;

public string AddItem(T item)
{
    return CreateItemAsync(item).Result.Id;
}

public Document GetDocument(string id)
{
    return Client.CreateDocumentQuery(Collection.DocumentsLink)
        .Where(d => d.Id == id)
        .AsEnumerable()
        .FirstOrDefault();
}

public IEnumerable<T> GetItems(string sqlQuery)
{
    FeedOptions queryOptions = new FeedOptions { MaxItemCount = -1 };

    IQueryable<T> query = this.Client.CreateDocumentQuery<T>(
        UriFactory.CreateDocumentCollectionUri(Database.Id, Collection.Id),
        sqlQuery,
        queryOptions);

    var result = query.ToList();
    return result;
}

public class DocumentDBServices<T>
{
    public Database Database{...}
    public DocumentCollection Collection{...}
    private DocumentClient Client{...}

    public DocumentDBServices(string endPoint, string accountKey, string databaseId, string collectionId, bool dropCollection=false)
    {
        this.Client = new DocumentClient(new Uri(endPoint), accountKey);
        this.Collection = this.ReadOrCreateCollection(databaseId, collectionId, dropCollection);
    }

    public T GetItem(Expression<Func<T, bool>> predicate)
    {
        return Client.CreateDocumentQuery<T>(Collection.DocumentsLink)
            .Where(predicate)
            .AsEnumerable()
            .FirstOrDefault();
    }

    public IEnumerable<T> GetItems(Expression<Func<T, bool>> predicate)
    {
        return Client.CreateDocumentQuery<T>(Collection.DocumentsLink)
            .Where(predicate)
            .AsEnumerable();
    }

    public async Task<Document> UpdateItemAsync(string id, T item)
    {
        Document doc = GetDocument(id);
        return await Client.ReplaceDocumentAsync(doc.SelfLink, item);
    }
}
```

HBase on Azure



Introducing Apache HBase

HBase & Hadoop Big Data Ecosystem

Applications

Acquisition

- Sqoop
- Flume
- HDFS (APIS)

Batch

- Map Reduce
- Pig

In-Memory

- Spark

Stream

- Storm
- Flume
- Kafka
- Spark Streaming

SQL

- Hive
- Phoenix
- Spark-SQL

NoSQL

- Accumulo
- HBase
- CouchBase

Machine Learning

- Mahout
- Spark-Mllib

Search

- Solr
- Elastic-search

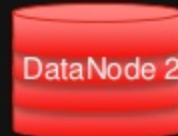
Orchest. Mgmt

- Oozie
- Ambari
- Zoo-Keeper

Named Node

Yet Another Resource Negotiator (YARN)

Hadoop Distributed File System (HDFS)



...



Introducing Apache HBase

HBase & Hadoop Big Data Ecosystem

HBase Cluster

APIs: Java Client, Thrift, Avro, REST

HBase Region Server

Region 1

Mem Store HFile

Write-ahead log

Region 2

MemStore HFile

Write-ahead log

Region N

Mem Store HFile

Write-ahead log

Zookeeper Services

HBase Master

Named Node

Yet Another Resource Negotiator (YARN)

Hadoop Distributed File System (HDFS)

DataNode 1

DataNode 2

DataNode 3

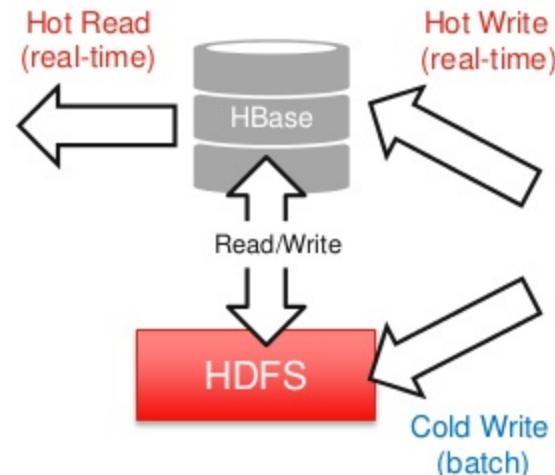
....

DataNode N

HBase on Azure HDInsight

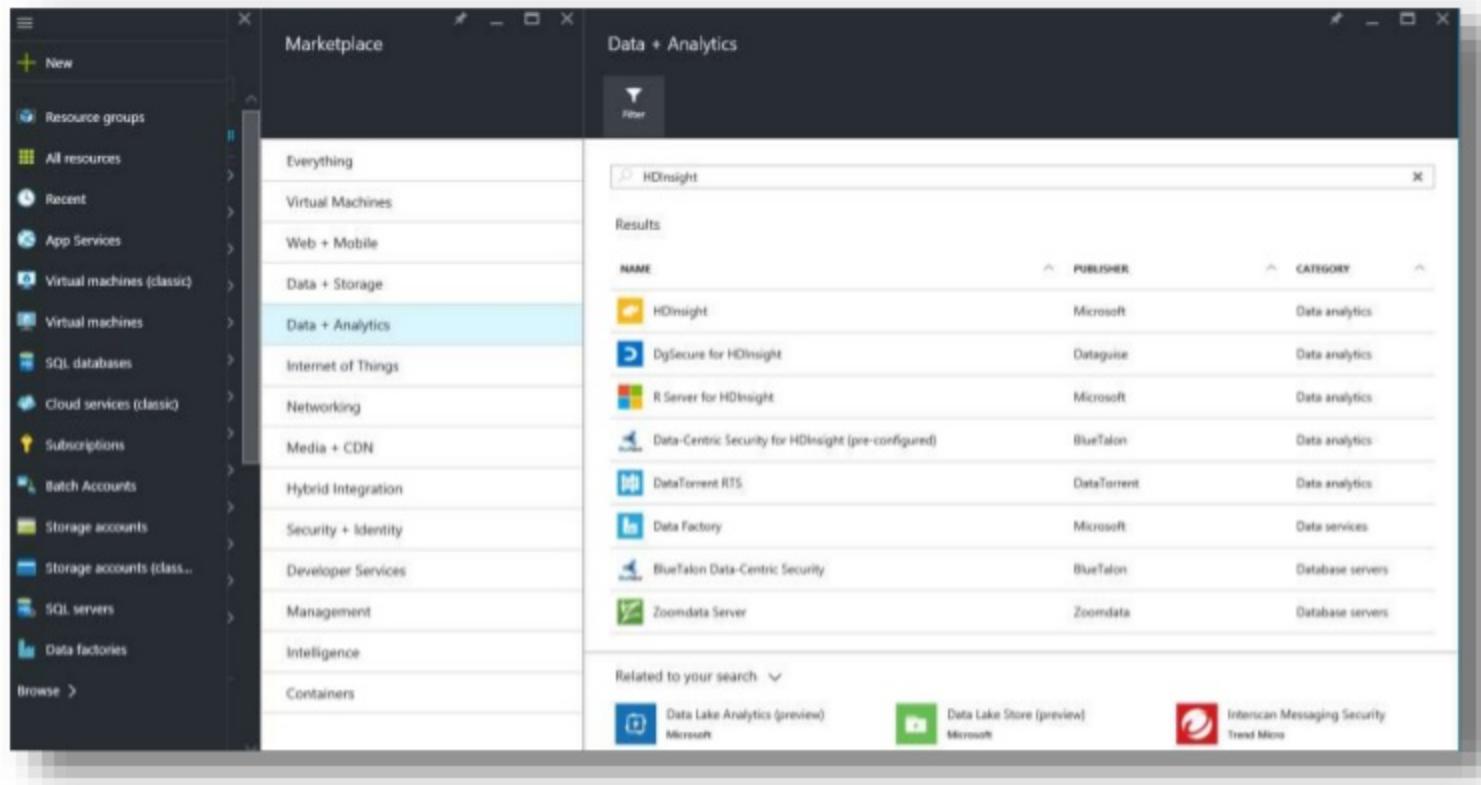
a brief introduction to HBase

- HDFS is suitable for batch processing (i.e., scan over big data files)
- HBase is optimized for **fast record lookups**, and singleton operations
- HDFS is usually the file system for HBase
- Rows maintained in **sorted lexicographical order** for efficient rows scan
- Row ranges are **partitioned** into tablets
- Column are grouped **into column families for locality indication**
- Simple commandlet: create, alter, drop, list, describe, get, put, incr, scan, count, delete, truncate, etc. <https://learnhbase.wordpress.com/2013/03/02/hbase-shell-commands/>
- APIs support batch operations
- A common choice with stream processing solutions



HBase on Azure HDInsight

Getting Started



The screenshot shows the Azure Marketplace interface. On the left, there's a sidebar with various service categories like Resource groups, All resources, Recent, App Services, etc. The 'Data + Analytics' category is selected. In the main pane, a search bar at the top has 'HDInsight' typed into it. Below the search bar, there's a 'Results' section with a table. The table has columns for NAME, PUBLISHER, and CATEGORY. The results listed are:

NAME	PUBLISHER	CATEGORY
HDInsight	Microsoft	Data analytics
DgSecure for HDInsight	Dataguiise	Data analytics
R Server for HDInsight	Microsoft	Data analytics
Data-Centric Security for HDInsight (pre-configured)	BlueTalon	Data analytics
DataTorrent RTS	DataTorrent	Data analytics
Data Factory	Microsoft	Data services
BlueTalon Data-Centric Security	BlueTalon	Database servers
Zoomdata Server	Zoomdata	Database servers

At the bottom of the results pane, there are three recommended items: Data Lake Analytics (preview) by Microsoft, Data Lake Store (preview) by Microsoft, and Interscan Messaging Security by Trend Micro.

HBase on Azure HDInsight

Getting Started

New HDInsight Cluster

Cluster Type configuration

Learn about HDInsight and cluster versions. [Learn more](#)

Cluster Type: HBase **Operating Systems**: Linux **Version**: HBase 1.1.2 (HDI 3.3)

Cluster Tier [View info](#)

STANDARD	PREMIUM (PREVIEW) *
Administration: Manage, monitor, connect	Administration: Manage, monitor, connect
Scalability: On-demand node scaling	Scalability: Available on Hadoop and Spark cluster
99.9% Uptime SLA	Microsoft R Services for HDInsight
Automatic patching	
+ 0.00 GBP/HOUR/HOUR	+ 0.01 GBP/HOUR/HOUR

Credentials
Configure required settings

Data Source
Configure required settings

Pricing
Please configure required settings

Optional Configuration

Please fix the errors on this page before continuing.

Pin to dashboard

Create

Select

Cluster Credentials

Create login and remote access credentials for the cluster and any applications.

Cluster Login Username: admin
Cluster Login Password: *****

Information: This is used for job submission, for adi

Enable Remote Desktop?: YES

Expires On: 2016-08-26

Remote Desktop Username: AdminName
Remote Desktop Password: *****

Information: This is used to remotely connect to yo

Select

Data Source

The cluster will use this data source as the primary location for most data access, such as job input and log output.

Selection Method: From all subscriptions

Select storage account: komadrihdstorage (West Europe)

Create new

Choose Default Container: komadrihbase

Location: West Europe

Pricing

To learn more, visit our pricing page. [Learn more](#)

Number of Region nodes: 1

Region node size: A3 (1 node, 4 cores)

Head node size: A3 (2 nodes, 8 cores)

Zookeeper node sizes: A2 (3 nodes, 6 cores)

REGION NODES: 0.15 x 1 = 0.15

HEAD NODES: 0.15 x 2 = 0.30

ZOOKEEPER NODES: 0.07 x 3 = 0.22

TOTAL COST: 0.66 GBP/HOUR (ESTIMATED)

18 of 60 cores would be used in West Europe.

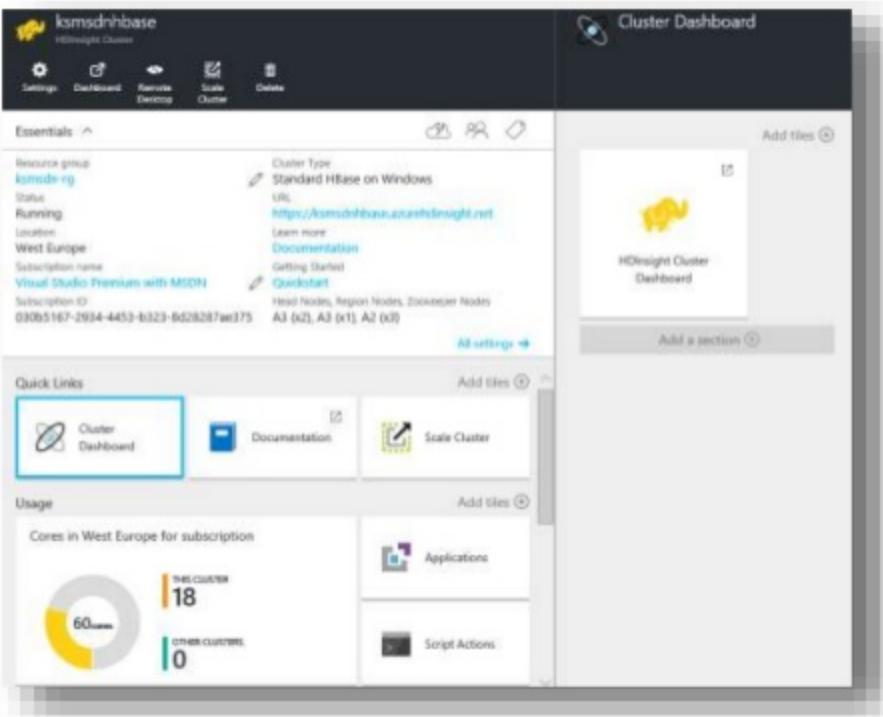
This price estimate does not include storage costs, network egress costs, or subscription discounts.

Questions? Contact billing support.

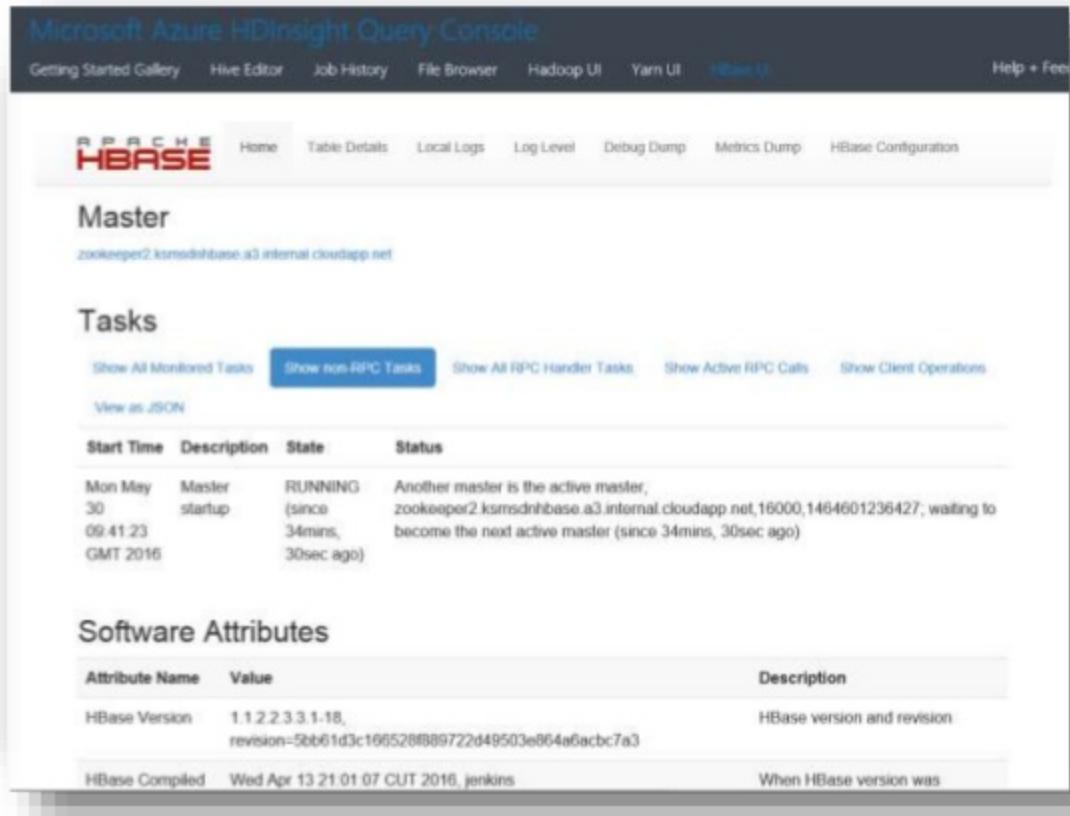
Select

HBase on Azure HDInsight

Exploring HBase



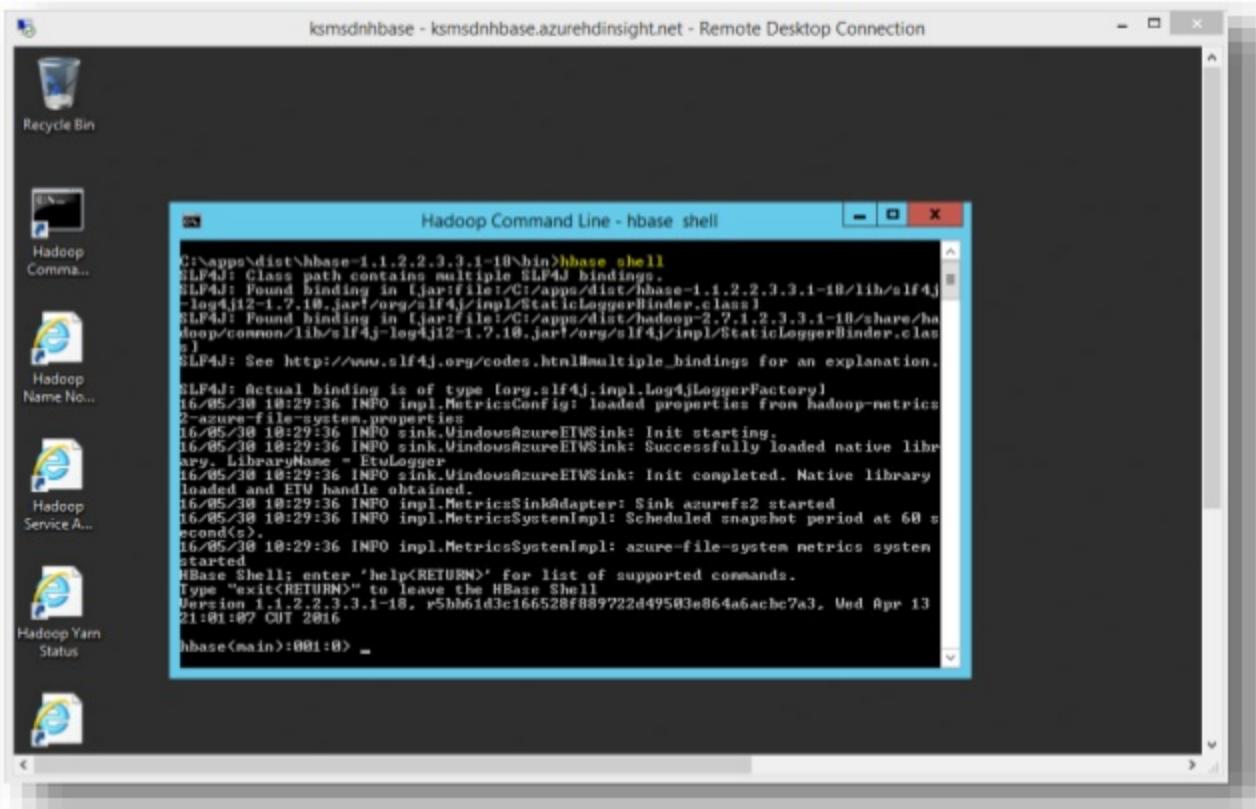
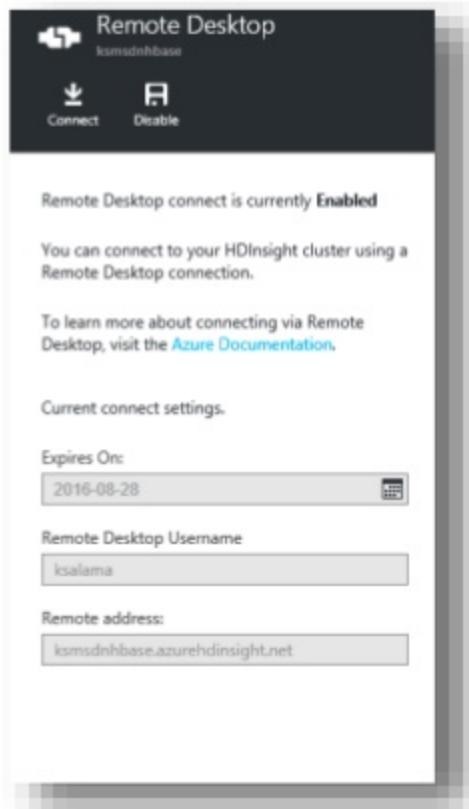
The screenshot shows the Azure portal interface for the 'ksmsdhbase' HDInsight cluster. On the left, there's a sidebar with 'Essentials' and 'Quick Links' sections. The 'Cluster Dashboard' section is highlighted. It displays the cluster type as 'Standard HBase on Windows', the URL as 'https://ksmsdhbase.azurehdinsight.net', and various status indicators like 'Status: Running'. Below this, there are sections for 'Usage' showing cores in West Europe and a 'Script Actions' section.



The screenshot shows the 'Apache HBase' master page in the Azure HDInsight Query Console. It features a 'Master' section with the URL 'zookeeper2.ksmsdhbase.a3.internal.cloudapp.net'. Below it is a 'Tasks' section with tabs for 'Show All Monitored Tasks' (selected), 'Show non-RPC Tasks', 'Show All RPC Handler Tasks', 'Show Active RPC Calls', and 'Show Client Operations'. A table lists a single task: 'Mon May 30 09:41:23 GMT 2016' (Master startup) in 'RUNNING' state since 34mins, 30sec ago. The 'Software Attributes' section shows the HBase version as '1.1.2.2.3.3.1-18, revision=5bb61d3c16652bf889722d49503e864a6acbc7a3'.

HBase on Azure HDInsight

Exploring HBase



HBase on Azure HDInsight

Exploring HBase

Basic Commands

```
create 'table_test', {NAME=>'f1'}, {NAME=>'f2'}
```

List

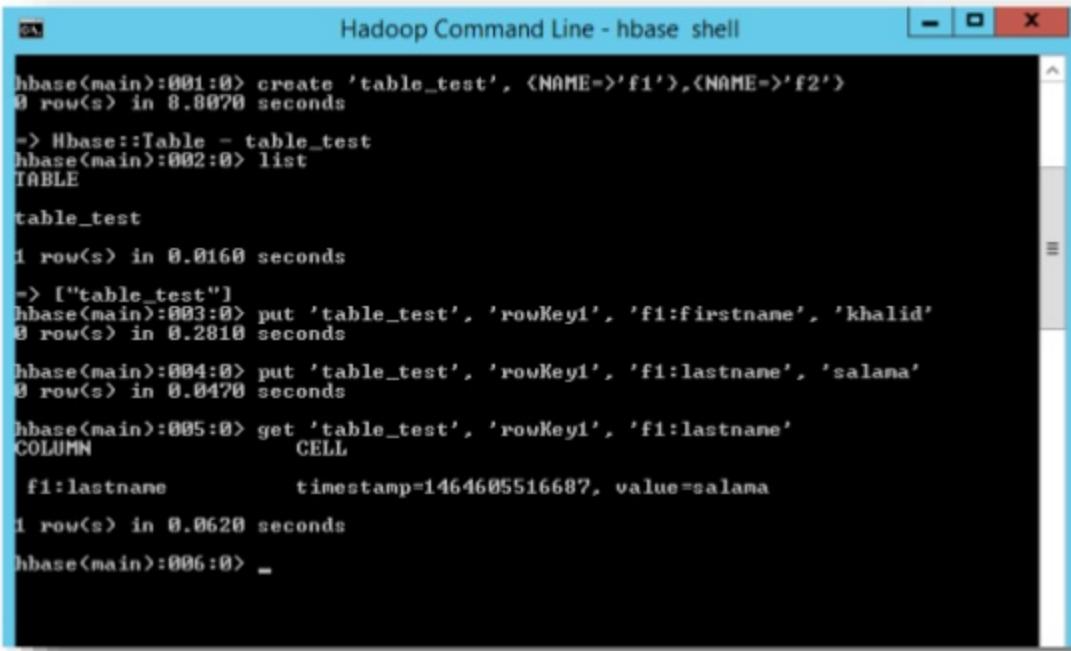
```
put 'table_test', 'rowKey1', 'f1:firstname', 'khalid'  
put 'table_test', 'rowKey1', 'f1:lastname', 'salama'  
put 'table_test', 'rowKey1', 'f2:level', '3'
```

```
put 'table_test', 'rowKey2', 'f1:firstname', 'paul'  
put 'table_test', 'rowKey2', 'f1:lastname', 'linehame'  
put 'table_test', 'rowKey2', 'f1:email', 'plinham@hitachi.com'
```

```
get 'table_test', 'rowKey1', 'f1:lastname'
```

```
disable 'table_test'
```

```
drop 'table_test'
```



The screenshot shows a terminal window titled "Hadoop Command Line - hbase shell". The session starts with creating a table:

```
hbase(main):001:0> create 'table_test', {NAME=>'f1'}, {NAME=>'f2'}  
0 row(s) in 8.8070 seconds
```

Then, it lists the table:

```
-> Hbase::Table - table_test  
hbase(main):002:0> list  
TABLE
```

It then shows the table definition:

```
table_test  
1 row(s) in 0.0160 seconds
```

Next, it performs a put operation:

```
-> ["table_test"]  
hbase(main):003:0> put 'table_test', 'rowKey1', 'f1:firstname', 'khalid'  
0 row(s) in 0.2810 seconds
```

Then, it performs another put operation:

```
hbase(main):004:0> put 'table_test', 'rowKey1', 'f1:lastname', 'salama'  
0 row(s) in 0.0470 seconds
```

Finally, it performs a get operation:

```
hbase(main):005:0> get 'table_test', 'rowKey1', 'f1:lastname'  
COLUMN CELL  
f1:lastname timestamp=1464605516687, value=salama  
1 row(s) in 0.0620 seconds
```

```
hbase(main):006:0>
```

HBase on Azure HDInsight

Exploring HBase

Other Commands

```
get 'table test' , 'rowkey1'
```

```
get 'table test', 'rowkey1' , {COLUMN => [f1:lastname]}
```

```
get 'table test', 'rowkey2', {TIMERANE => [0:1000]}
```

```
scan 'table test' {LIMIT =>100}
```

```
scan 'table test' {STARTROW=>'rowkey5' , STOPROW='rowkey10'}
```

HBase on Azure HDInsight

HBase Reader/Writer using .NET

```
using Microsoft.HBase.Client;
using org.apache.hadoop.hbase.rest.protobuf.generated;

public HBaseServices(Uri clusterUri, string userName, string password, string tableName, bool maintainRowCount=true)
{
    var credentials = new ClusterCredentials(clusterUri, userName, password);
    this.TableName = tableName;
    this._maintainRowCount = maintainRowCount;

    client = new HBaseClient(credentials);

    if (!client.ListTablesAsync().Result.name.Contains(this.TableName))
    {
        // Create the table
        var tableSchema = new TableSchema();
        tableSchema.name = this.TableName;
        if(maintainRowCount)
            tableSchema.columns.Add(new ColumnSchema { name = "d" });
        client.CreateTableAsync(tableSchema).Wait();
        Console.WriteLine("Table \'{0}\' created.", TableName);
    }
}
```

```
public class HBaseItem
{
    public string ColumnFamily
    { get; set; }

    public string ColumnName
    { get; set; }

    public string Value
    { get; set; }

    public long Timestamp
    { get; set; }
}
```

HBase on Azure HDInsight

HBase Reader/Writer using .NET

```

private async Task AddRow(string rowKey, IEnumerable<HBaseItem> items)
{
    var row = new CellSet.Row { key = Encoding.UTF8.GetBytes(rowKey) };

    foreach (var item in items)
    {
        var cell = new Cell
        {
            column = Encoding.UTF8.GetBytes(item.ColumnFamily + ":" + item.ColumnName),
            data = Encoding.UTF8.GetBytes(item.Value)
        };
        row.values.Add(cell);
    }

    await this.client.CheckAndPutAsync(this.TableName, row, null);

    if (_maintainRowCount)
    {
        long count = this.RowCount;
        count++;
        await this.SetRowCount(count);
    }
}

```

```

public async Task AddRows(Dictionary<string, List<HBaseItem>> hbaseRows)
{
    var set = new CellSet();

    foreach (string rowKey in hbaseRows.Keys)
    {
        var row = new CellSet.Row { key = Encoding.UTF8.GetBytes(rowKey) };
        var items = hbaseRows[rowKey];

        foreach (var item in items)
        {
            var cell = new Cell
            {
                column = Encoding.UTF8.GetBytes(item.ColumnFamily + ":" + item.ColumnName),
                data = Encoding.UTF8.GetBytes(item.Value)
            };
            row.values.Add(cell);
        }

        set.rows.Add(row);
    }

    await this.client.StoreCellsAsync(this.TableName, set);

    if (_maintainRowCount)
    {
        long count = this.RowCount;
        count+=set.rows.Count;
        await this.SetRowCount(count);
    }
}

```

HBase on Azure HDInsight

HBase Reader/Writer using .NET

```

public List<HBaseItem> GetRow(string rowKey)
{
    List<HBaseItem> items = new List<HBaseItem>();

    var cells = this.client.GetCellsAsync(this.TableName, rowKey).Result.rows[0].values;
    foreach (var cell in cells)
    {
        var parts = Encoding.UTF8.GetString(cell.column).Split(':');

        var item = new HBaseItem()
        {
            ColumnFamily = parts[0]
            , ColumnName = parts[1]
            , Value = Encoding.UTF8.GetString(cell.data)
            , Timestamp = cell.timestamp
        };

        items.Add(item);
    }

    return items;
}

public string GetValue(string rowKey, string columnFamily, string columnName)
{
    return this.GetRow(rowKey)
        .Where(i=> i.ColumnFamily==columnFamily && i.ColumnName==columnName)
        .First().Value;
}

```

```

public async Task<List<string>> GetRowsByColumnValues(IEnumerable<HBaseItem> items
    , string startRowKey = null, string endRowKey = null)
{
    var list = new List<string>();

    RequestOptions scanOptions = RequestOptions.GetDefaultOptions();
    scanOptions.AlternativeEndpoint = "hbaserest0/";
    var scanSettings = new Scanner
    {
        batch = 100000
        ,startRow = Encoding.UTF8.GetBytes(startRowKey)
        ,endRow = Encoding.UTF8.GetBytes(endRowKey)
    };
    ScannerInformation scannerInfo = null;
    try
    {
        scannerInfo = await client.CreateScannerAsync(this.TableName, scanSettings, scanOptions);

        CellSet next;
        while ((next = await client.ScannerGetNextAsync(scannerInfo, scanOptions)) != null)
            foreach (CellSet.Row row in next.rows)
                if (Match(row,items))
                    list.Add(Encoding.UTF8.GetString(row.key));
    }

    return list;
}
finally
{
    if (scannerInfo != null)
    {
        client.DeleteScannerAsync(this.TableName, scannerInfo, scanOptions).Wait();
    }
}

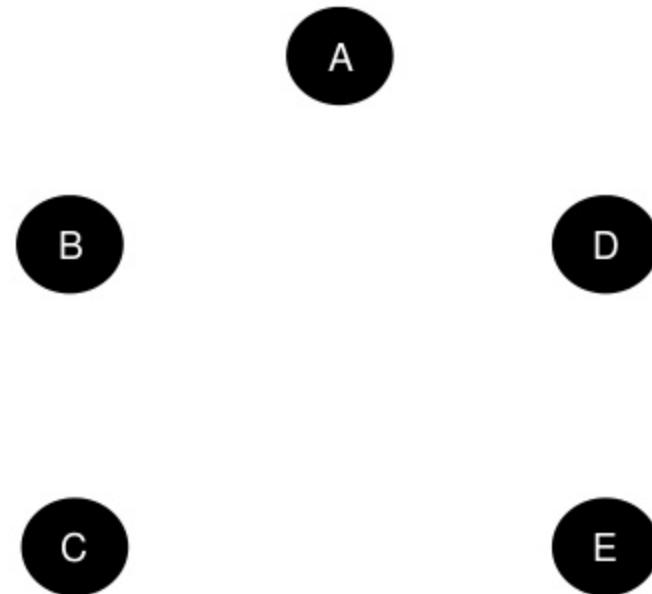
```

How to Get Started with NoSQL?

- Read the slides!
- Azure Documentation – Azure Table Storage
<https://azure.microsoft.com/en-gb/documentation/articles/storage-dotnet-how-to-use-tables/>
- Azure Documentation – Azure DocumentDB
<https://azure.microsoft.com/en-gb/documentation/services/documentdb/>
- Azure Documentation – Azure Redis Cache
<https://azure.microsoft.com/en-gb/documentation/services/redis-cache/>
- Azure Documentation – HBase on HDInsight
<https://azure.microsoft.com/en-gb/documentation/articles/hdinsight-hbase-overview/>
- GitHub – tweet-sentiment with HBase
<https://github.com/maxluk/tweet-sentiment>
- Azure Documentation – Understanding NoSQL on Azure
<https://azure.microsoft.com/en-gb/documentation/articles/fundamentals-data-management-nosql-chappell/>
- db-engines - Knowledge Base of Relational and NoSQL Database Management Systems
<http://db-engines.com/en/>
- NoSQL Databases
<http://nosql-database.org/>

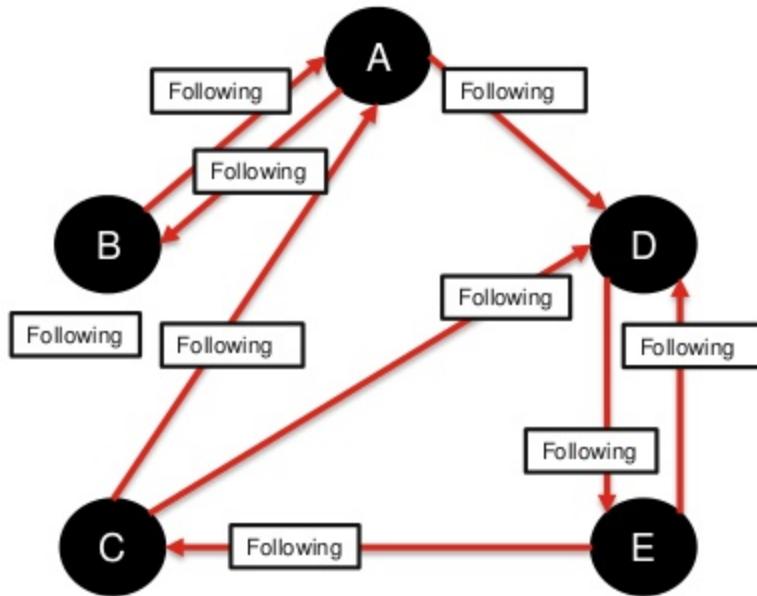
How to Play with neo4j

A widely-used GraphDB



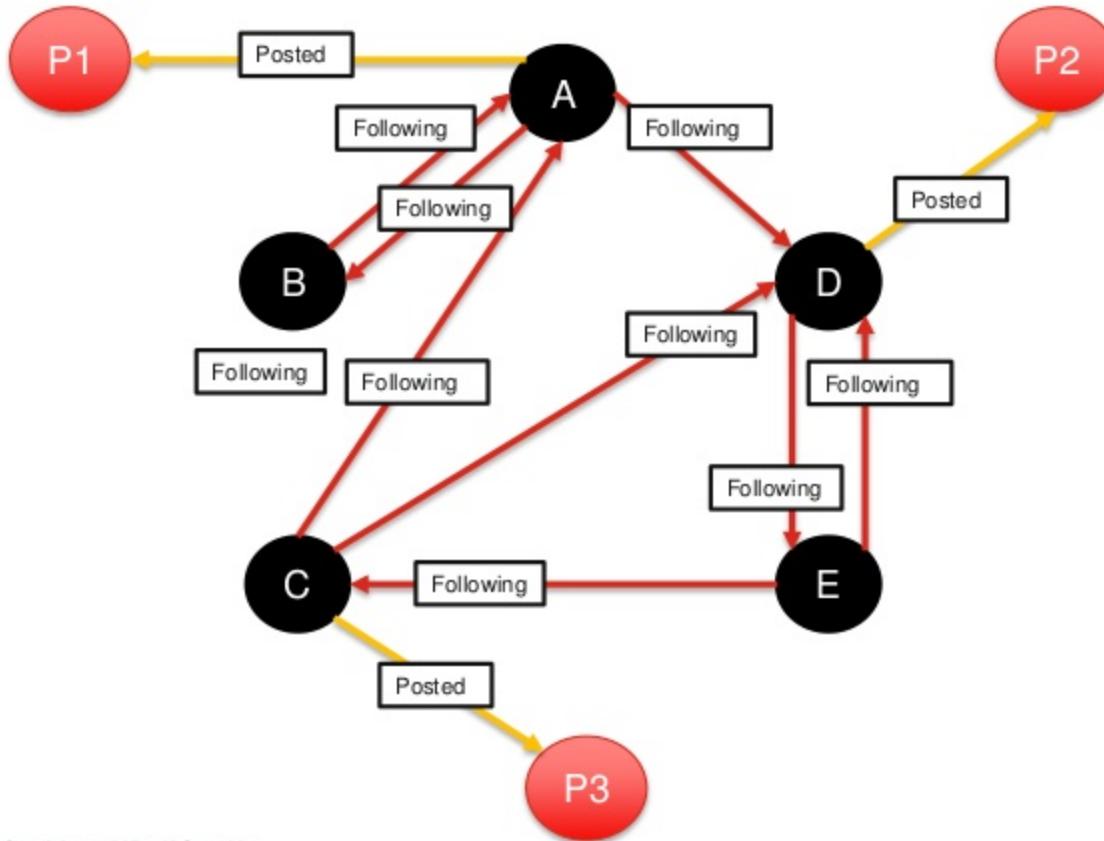
How to Play with neo4j

A widely-used GraphDB



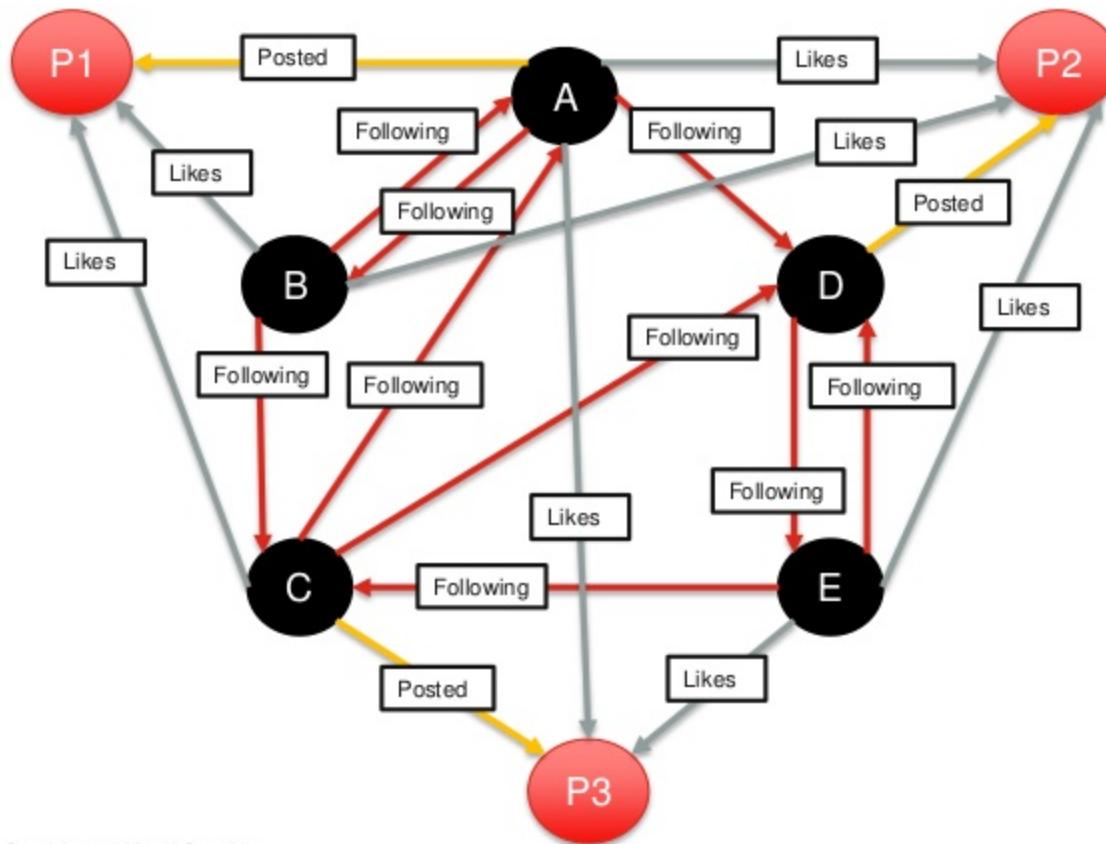
How to Play with neo4j

A widely-used GraphDB



How to Play with neo4j

A widely-used GraphDB



How to Play with neo4j

A widely-used GraphDB

CREATE

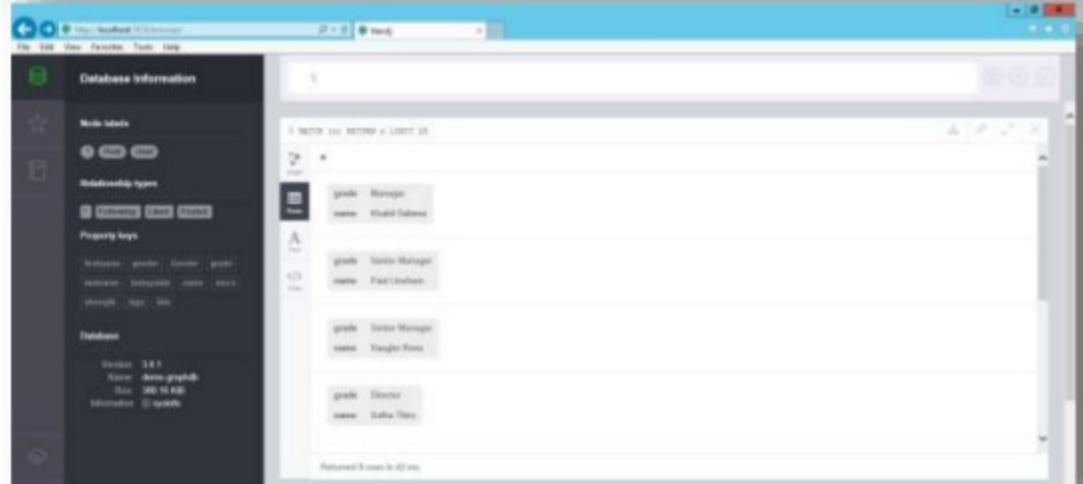
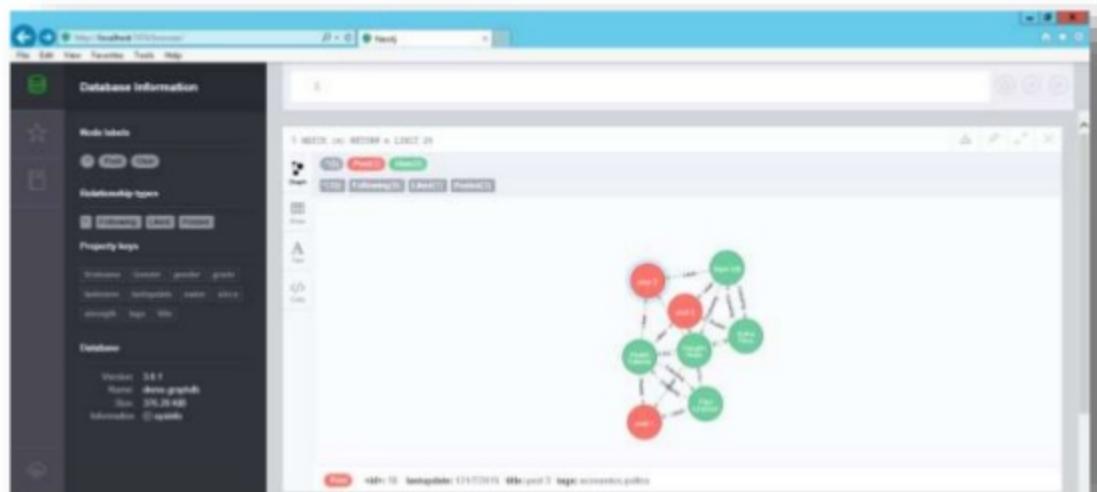
```
(a:User{name:"Khalid Salama", grade:"Manager"}),
(b:User{name:"Paul Lineham", grade:"Senior Manager"}),
(c:User{name:"Vaughn Rees", grade:"Senior Manager"}),
(d:User{name:"Sutha Thiru", grade:"Director"}),
(e:User{name:"Mark Hill", grade:"VP"}),
```

```
(a)-[:Following{since:'2014'}]->(d),
(a)-[:Following{since:'2014'}]->(b),
(b)-[:Following{since:'2010'}]->(a),
(d)-[:Following{since:'2011', strength:"high"}]->(e),
(e)-[:Following{since:'2014'}]->(d),
(e)-[:Following{since:'2015'}]->(c),
(c)-[:Following]->(d),
(c)-[:Following{since:'2013', strength:"low"}]->(a),
(b)-[:Following]->(c),
```

```
(p1:Post{title:"post 1", lastupdate:"01/01/2016", tags:['sports','life style']}),
(p2:Post{title:"post 2", lastupdate:"03/05/2015"}),
(p3:Post{title:"post 3", lastupdate:"121/7/2015", tags:['economics','politics']}),
```

```
(a)-[:Posted]->(p1),
(d)-[:Posted]->(p2),
(c)-[:Posted]->(p3),
```

```
(b)-[:Liked]->(p1),
(c)-[:Liked]->(p1),
(a)-[:Liked]->(p2),
(b)-[:Liked]->(p2),
(e)-[:Liked]->(p2),
(a)-[:Liked]->(p3),
(e)-[:Liked]->(p3)
```



How to Play with neo4j

A widely-used GraphDB

```
-- fetch one node
MATCH (u:User{name:"Khalid Salama"}) RETURN u

-- fetch an attribute of a node
MATCH (u:User{name:"Khalid Salama"}) RETURN u.grade

-- fetch nodes by conditions
MATCH (u:User{grade:"Senior Manager"}) RETURN u
--  
MATCH (u:User)
WHERE u.grade = 'Senior Manager'
RETURN u
--  
MATCH (u:User)
WHERE u.name =~ "Sutha.+"
-- START WITH, END WITH, CONTAIN, IN []
RETURN u
--  
MATCH ()-[r:Posted]->(p:Post)
WHERE 'sports' IN p.tags
RETURN p

-- Whom khalid is following?
MATCH (x:User{name:"Khalid Salama"})-[r:Following]->(y:User)
RETURN x,r,y

-- Who is Following Khalid
MATCH (x:User{name:"Khalid Salama"})<-[r:Following]-(y:User)
RETURN x,r,y

-- Update
MERGE (u:User { name:"Khalid Salama" })
SET u.practice = "Data Insights & Analytics"
RETURN u
```

```
- Get Count of Posts
MATCH (p:Post) RETURN COUNT(p)
-- Get User Count By Grade
MATCH (u:User) RETURN u.grade, COUNT(u)
-- Get User and Followers
MATCH (u:User)<-[:Following]-(f:User)
RETURN u.name AS User, COLLECT(f.name) AS follows,COUNT(f) AS Total

-- Constraint
CREATE CONSTRAINT ON (u:User) ASSERT u.name IS UNIQUE
-- Index
CREATE INDEX ON :User(grade)

-- Get users following each other
MATCH (u1:User)-[:Following]->(u2:User)-[:Following]->(u1)
RETURN u1.name,u2.name

-- Get Users likes a post posted by a follower
MATCH (u:User)-[:Liked]->(p:Post)<-[ :Posted]-(u2:User)-[:Following]->(u)
RETURN u,p,u2

-- Get Following of Following
MATCH (u:User)-[:Following]->()-[:Following]->(u2:User)
Return u.name,COLLECT(DISTINCT u2.name)

-- Get User with max 3 steps from Paul
MATCH (u:User)-[:Following*..3]->(us:User{name:"Paul Lineham"})
Return u

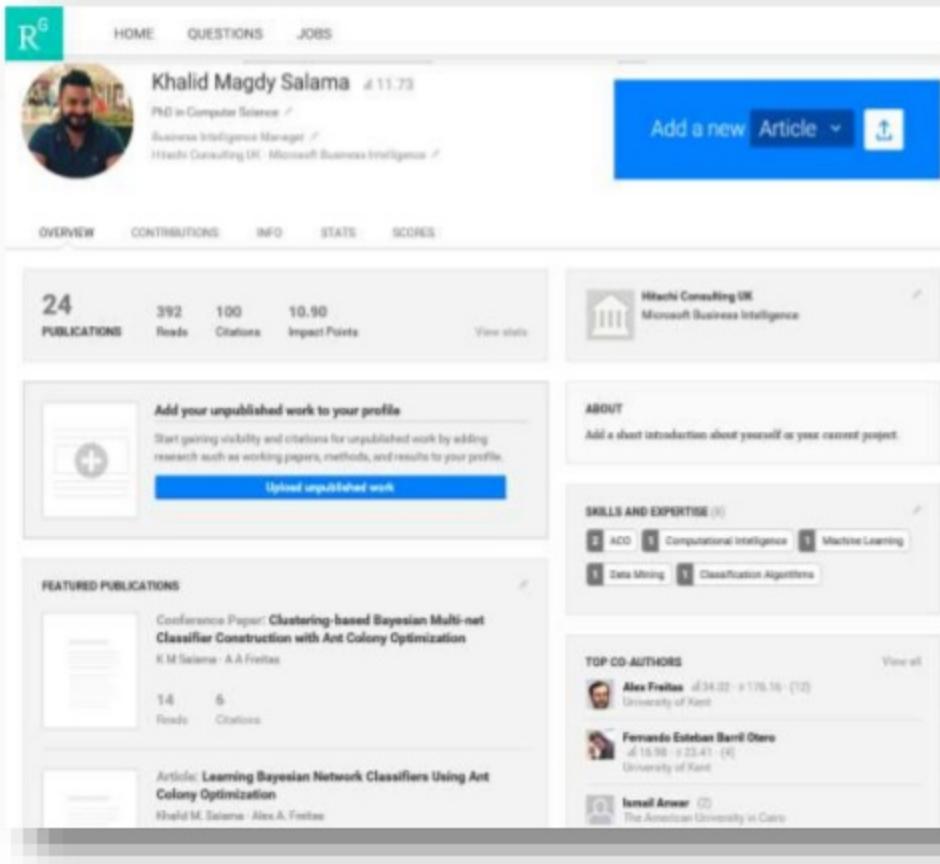
-- Shortest path
MATCH
    (u1:User{name:"Mark Hill"}),
    (u2:User{name:"Paul Lineham"}),
    p=SHORTESTPATH((u1)-[:Following*..10]->(u2))
RETURN p

-- Get nodes having a property
MATCH(p)
WHERE EXSITS(p.tags)
```

My Background

Applying Computational Intelligence in Data Mining

- Honorary Research Fellow, School of Computing , University of Kent.
- Ph.D. Computer Science, University of Kent, Canterbury, UK.
- M.Sc. Computer Science , The American University in Cairo, Egypt.
- 25+ published journal and conference papers, focusing on:
 - *classification rules induction,*
 - *decision trees construction,*
 - *Bayesian classification modelling,*
 - *data reduction,*
 - *instance-based learning,*
 - *evolving neural networks,* and
 - *data clustering*
- **Journals:** *Swarm Intelligence, Swarm & Evolutionary Computation, Applied Soft Computing, and Memetic Computing.*
- **Conferences:** ANTS, IEEE CEC, IEEE SIS, EvoBio, ECTA, IEEE WCCI and INNS-BigData.



The screenshot shows Khalid Magdy Salama's profile on ResearchGate. At the top, there is a green header with the letter 'R' and a small 'G'. Below the header, the profile information for Khalid Magdy Salama is displayed, including his name, title (PhD in Computer Science), and current role (Business Intelligence Manager at Hitachi Consulting UK). A blue button on the right says "Add a new Article". Below the profile, there are tabs for OVERVIEW, CONTRIBUTIONS, INFO, STATS, and SCORES. The OVERVIEW section shows 24 PUBLICATIONS, 392 Reads, 100 Citations, and 10.90 Impact Points. There is a "View stats" link. A large central box encourages users to "Add your unpublished work to your profile" and provides a "Upload unpublished work" button. Below this, there is a "FEATURED PUBLICATIONS" section showing two publications: "Conference Paper: Clustering-based Bayesian Multi-net Classifier Construction with Ant Colony Optimization" and "Article: Learning Bayesian Network Classifiers Using Ant Colony Optimization". Both publications have 14 Reads and 6 Citations. To the right, there are sections for "ABOUT" (with a placeholder for a short introduction), "SKILLS AND EXPERTISE" (listing ACO, Computational Intelligence, Machine Learning, Data Mining, and Classification Algorithms), and "TOP CO-AUTHORS" (listing Alex Freitas, Fernando Esteban Barril Otero, and Ismail Awad).



Thank you!