InterSystems
Health | Business | Government

Deal with Dynamic (Schema-less) Data

Chris Carmichael
Sales Engineer

Introduction – Document database

No SQL Databases

- Not Relational and don’t require SQL
- Goal is speed and scalability
- No schema or table joins
- Discount consistency for availability and partition tolerance

CAP Theorem

It is impossible for a distributed data store to simultaneously provide more than two out of the following three guarantees:
-Eric Brewer

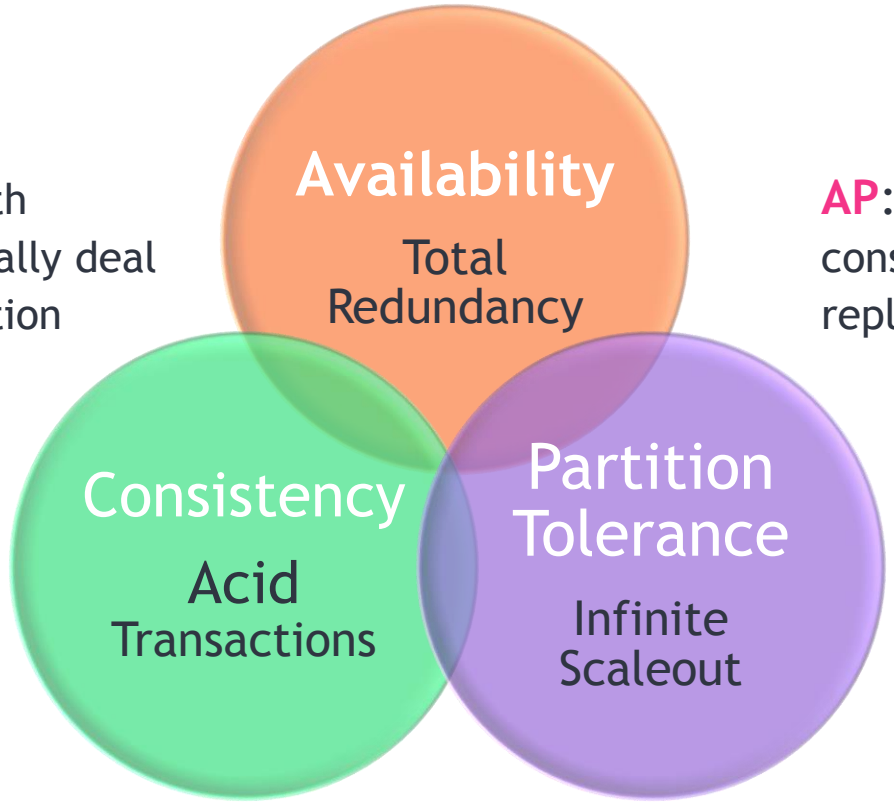
Consistency	Availability	Partition Tolerance
All the servers in the system will have the same data so users will get the same copy regardless of which server answers their request.	The system will always respond to a request, even if it's not the latest data or consistent across the system.	The system continues to operate as a whole even if individual servers fail or can't be reached.

CAP Theorem

One of the primary goals of NoSQL systems is horizontal scalability.

To scale horizontally, you need strong network partition tolerance which requires giving up either consistency or availability.

AC: Has trouble with partitions and typically deal with it with replication



AP: Achieves "eventual consistency" through replication and verification.

CP: Has trouble with availability while keeping data consistent across partitioned nodes

NoSQL Data Model Examples

Data Model	Example Requirements	Example Uses
Key-value Store	Availability, scale and speed	Big data capture, caching
Document	Indexing ability and model flexibility	Agile (mobile) app development, mixed data analytics
Column	Consistency	Random, realtime read/write access for big data
Graph	Articulated relationships	Analyzing and supporting social networks, knowledge intensive environments

Relational Databases

- Useful when the data is discrete and structured
- Structured data has a predefined format
- Can be stored in row and column tables
- Most Business data is unstructured
- Not easily stored in a tabular fashion
- Unstructured data refers to text extensive data
- May have discrete data as part of the text

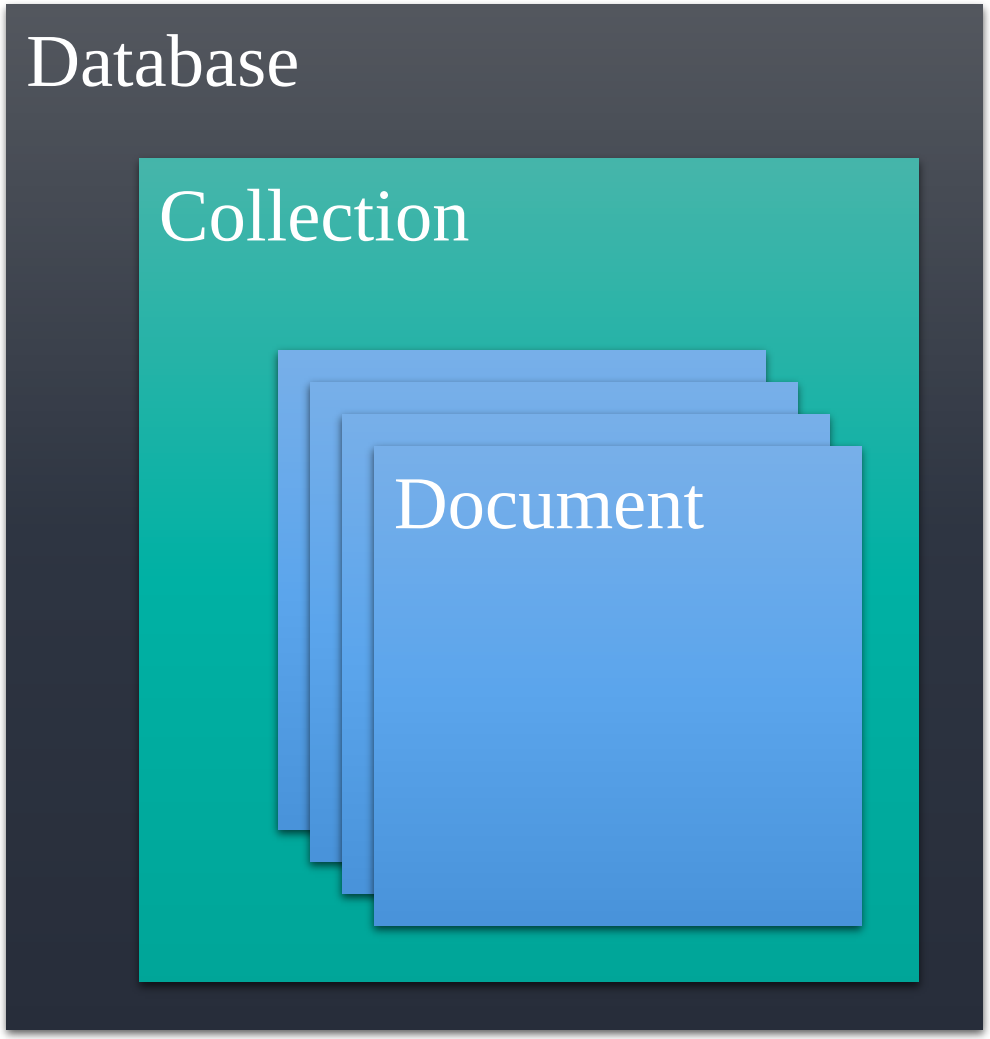
Student_Name	ID_Number	Office	GPA
Knuth	328012098	022	4.00
Von Neuman	481080220	555	3.78
Russell	238082388	022	3.85
Einstein	238001920	022	2.11
Newton	1727017	333	3.61
Karp	348882811	022	3.98
Bernoulli	2921938	022	3.21

During the second quarter of FY '07, Cherry Valley National Park purchased one ADA compliant bus in the amount of $200,000.

Document Database

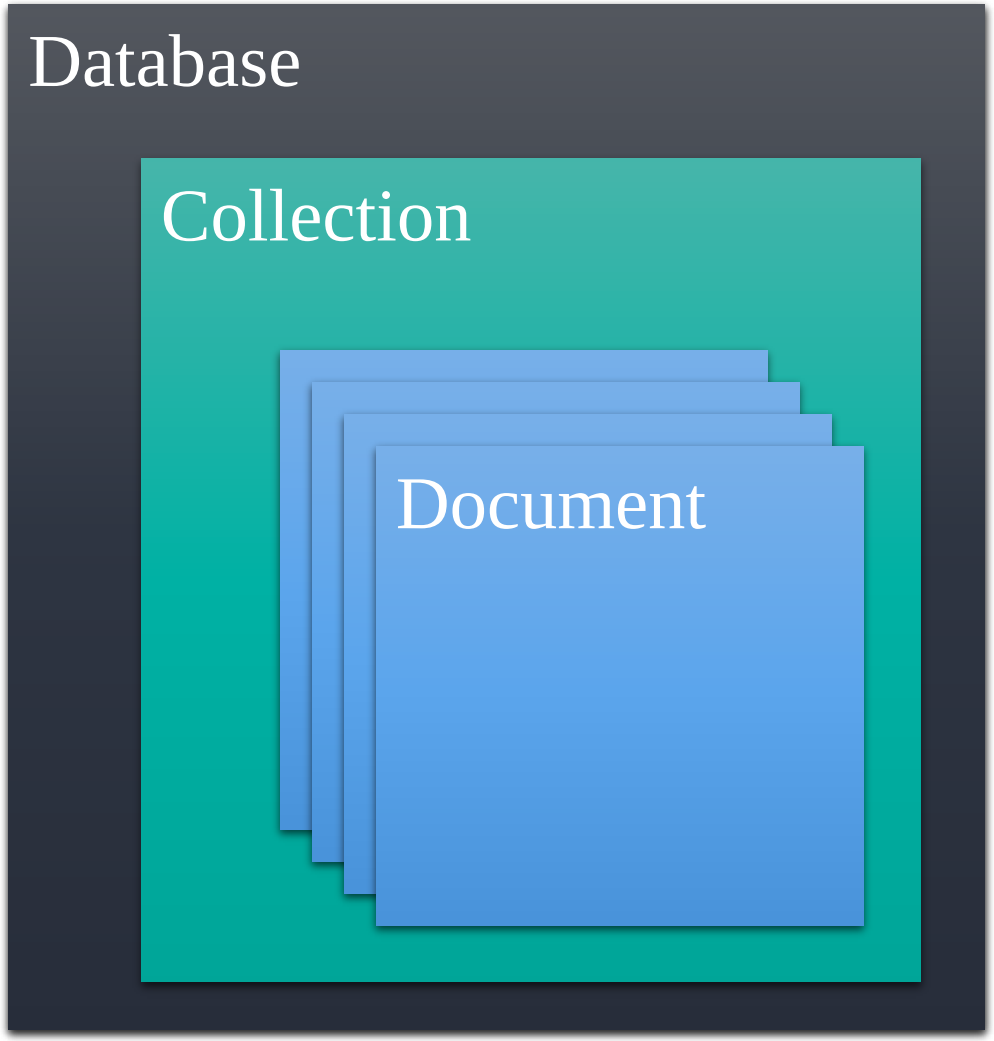
- Depending on database vendor may use JSON, BSON or XML as storage structure

RDBMS	Document DB
Table	Collection
Column	Key
Value	Value
Row	Document



Document Database

- Each record in the database is a document
- Each document is an autonomous set of data
- Documents contain:
  - Name value pairs
  - Arrays
  - Objects
- Nesting of arrays and objects can create complex structures
- Documents are replaced not updated



 | |

Document Example

An example using JSON notation:

```
{
        "Name" : "John Smith",
        "address" : {
                        "streetAddress": "21 2nd Street",
                        "city" : "New York",
                        "state" : "NY",
                        "postalCode" : 10021
                    },
        "phoneNumber" :
                    [
                        { "type" : "home", "number": "212 555-1234" },
                        { "type" : "fax", "number": "646 555-4567" }
                    ]
 }
```

Name-value pair

Object

Array

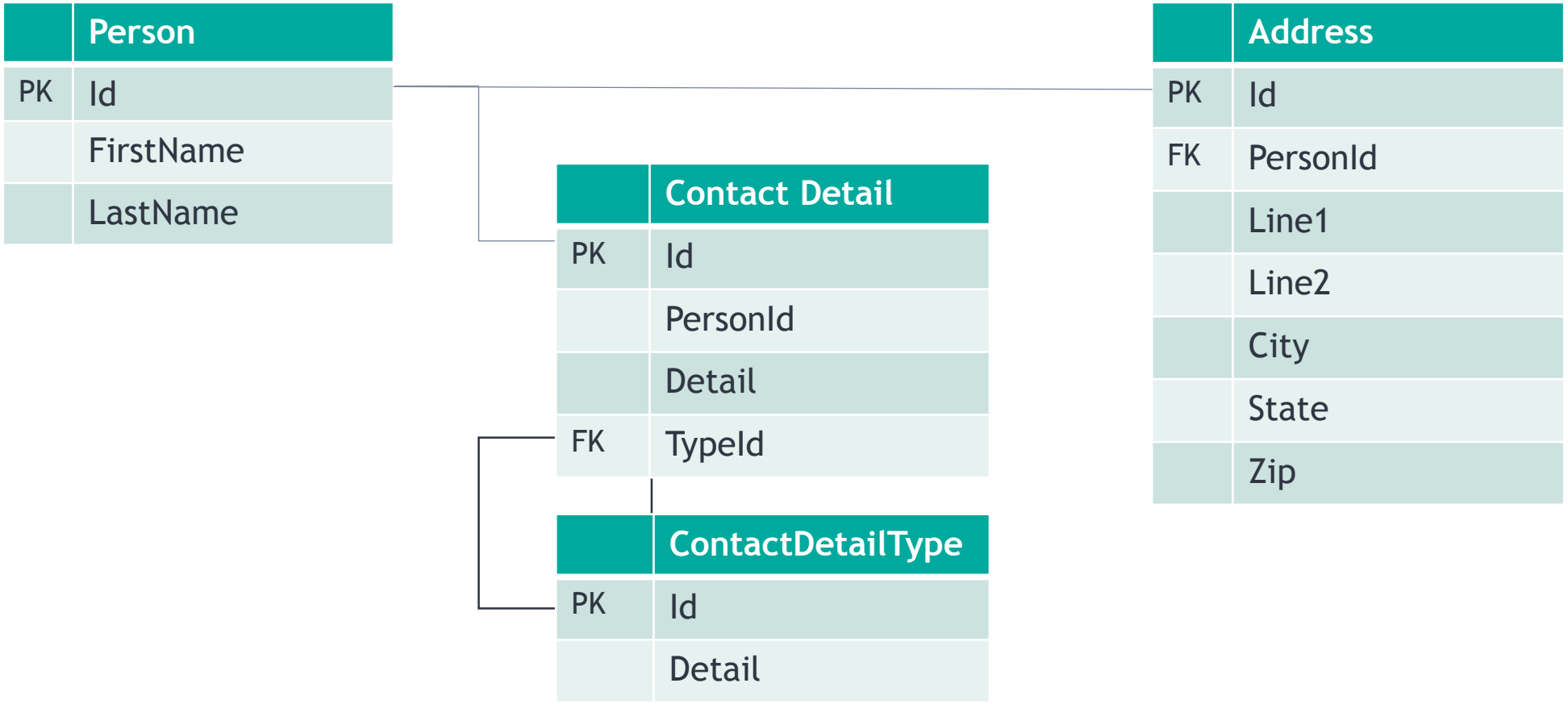
Document Database

The document model has benefits suited for specific use cases:

- **Flexibility**
  - Schemas don't have to be defined upfront (or ever) and therefore application developers can quickly setup work environments for their data and easily adapt to changes in data structure.
- **Sparseness**
  - Attributes with a particular key may appear in some documents within a collection, but not in others.
- **Hierarchical Data**
  - Structured types may be arrays or objects, nested arbitrarily deep. Data is stored de-normalized in contrast to relational systems where data is stored normalized.
- **Dynamic Typing**
  - Values for a particular key have no fixed data type and may vary from document to document.

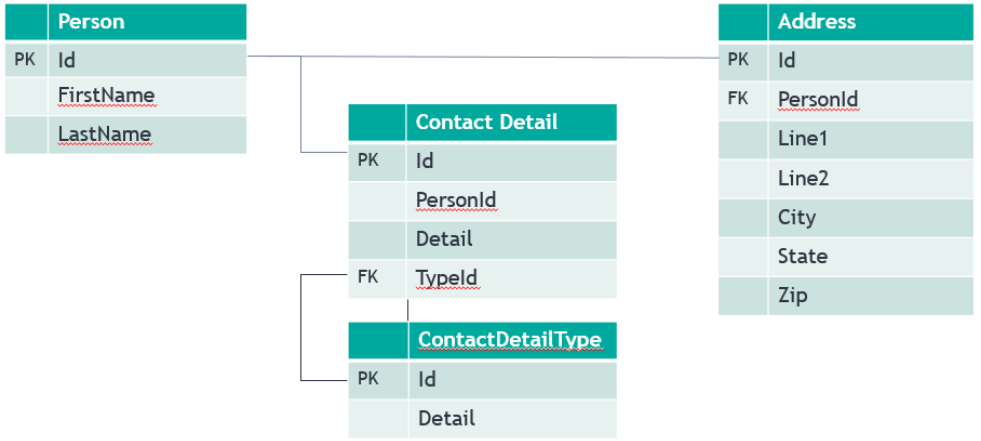
Document vs Relational

Relational data should be normalized, avoid storing redundant data, instead refer to it



Document vs Relational

- Retrieving a persons record requires joins to gather the name, address and contact details.
- Creating or updating requires write operations across many individual tables




```
SELECT p.FirstName, p.LastName, a.City, cd.Detail
FROM Person p
JOIN ContactDetail cd ON cd.PersonId = p.Id
JOIN ContactDetailType on cdt ON cdt.Id = cd.TypeId
JOIN Address a ON a.PersonId = p.Id
```


Document vs Relational

- This is how we would model the same data as a document
- The data is de-normalized
- All of the data is embedded
- Retrieving the data is now a single operation
- Creating or updating the document is a single operation

```
{
   "id":"1",
   "firstName":"Thomas",
   "lastName":"Andersen",
   "addresses":[
      {
         "line1":"100 Some Street",
         "line2":"Unit 1",
         "city":"Seattle",
         "state":"WA",
         "zip":98012
      }
   ],
   "contactDetails":[
      {"email":"thomas@andersen.com"},
      {"phone":"+1 555 555-5555","extension":5555}
   ]
}
```

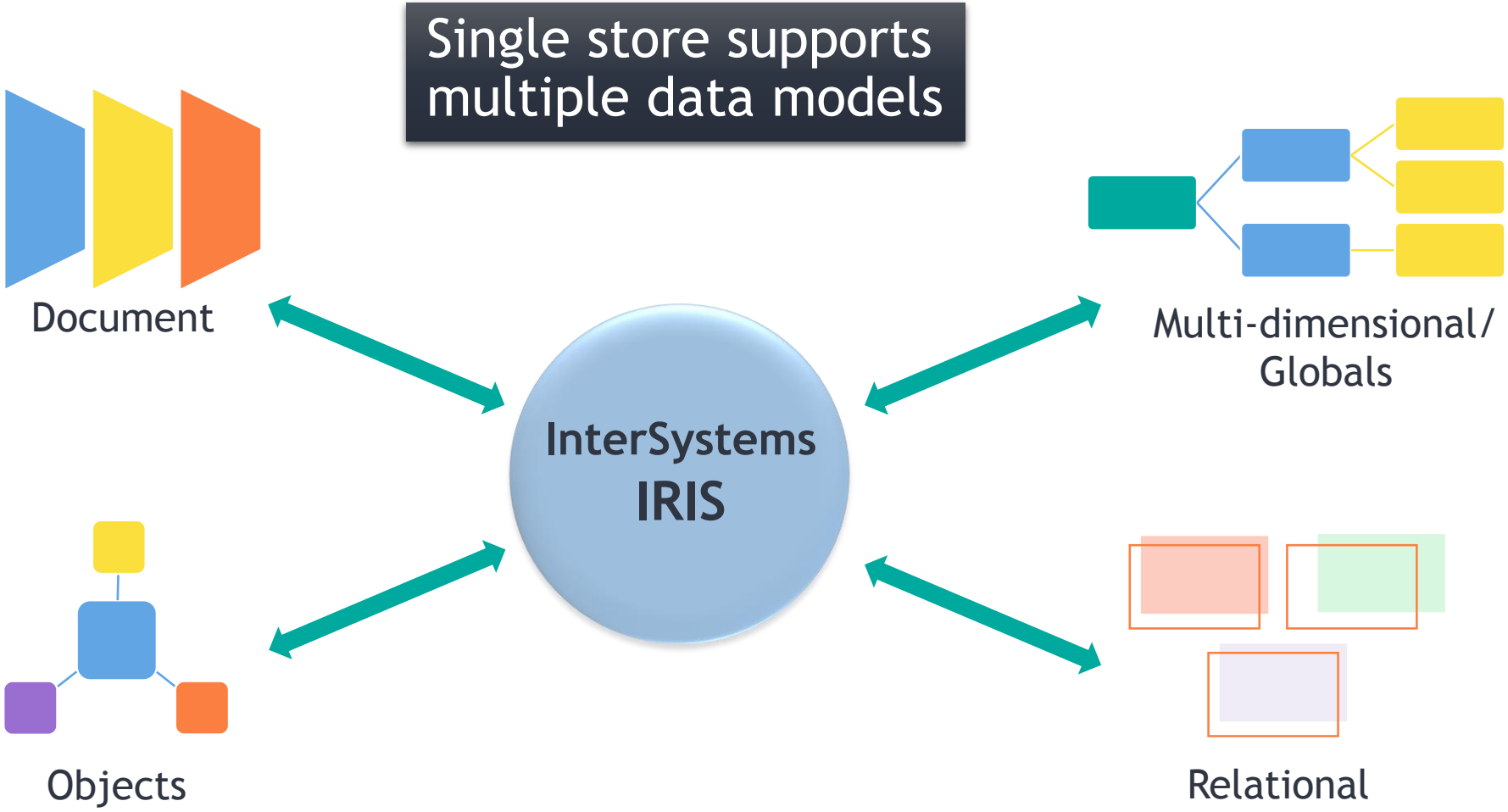

Document vs Relational

- Use relational when:
  - Application needs query flexibility
  - Requires sort order
  - Doesn’t have large data or performance requirements

- Document Database is good for:
  - High availability
  - Scalability
  - Flexibility

Document oriented database using InterSystems IRIS

InterSytems multi-model architecture



DocDB Classes

Class	Purpose
%DocDB.Database	API to manage document databases
%DocDB.Document	Instance Container for a document
%DocDB.REST	REST APIs for DocDB
%Library.DynamicAbstractObject	Object class which contains JSON structures. Used as the container for documents.

Working with DocDB – create a document database

- Create a new document database

```
set personDB = ##class(%DocDB.Database).CreateDatabase("Person")
```

- Create new properties, these allow you to search on the property and index it

```
do personDB.%CreateProperty("FirstName","%String","firstName",0)
do personDB.%CreateProperty("LastName","%String","lastName",1)
```


Working with DocDB – insert a document into the database

- Sample dynamic abstract object using JSON notation

```
set thisPerson = {"id":"1","firstName":"Thomas","lastName":"Andersen","addresses":[{"line1":"100 Some Street",
                  "line2":"Unit 1","city":"Seattle","state":"WA","zip":98012}],"contactDetails":[
                  {"email":"thomas@andersen.com"},{"phone":"+1 555 555-5555","extension":5555}]}
```

- Transform dynamic abstract object to JSON string

```
set thisPersonJSON = thisPerson.%ToJSON()
```

- Accepts a JSON string and inserts it into the document database

```
do personDB.%FromJSON(thisPersonJSON)
```


Working with DocDB – insert a document into the database

This is a persisted class so the usual methods also work:

- Instantiate an instance of the Document class

```
set personDB=##class(ISC.DM.Person).%New()
```

- Set the JSON structure to the %Doc property of the document class

```
set personDB.%Doc={"id":"1","firstName":"Thomas","lastName":"Andersen","addresses":[{"line1":"100 Some Street",
        "line2":"Unit 1","city":"Seattle","state":"WA","zip":98012}],"contactDetails":[
        {"email":"thomas@andersen.com"},{"phone":"+1 555 555-5555","extension":5555}]}
```

- Set the JSON structure to the %Doc property of the document class

```
write personDB.%Save()
```


Working with DocDB – find a document

- InstantiateDocument object

```
set db=$system.DocDB.GetDatabase("Person")
```

- Search for document format is [searchProperty,comaprisonValue,comparisonOperater]
  Search without anyprojection request returns entire document

```
set result=db.%FindDocuments(["firstName","Bill","="])
w result.%ToJSON()
```

{"sqlcode":100,"message":null,"content":[{"%Doc":"{\"id\":\"1\",\"firstName\":\"Bill\",
\"lastName\":\"Smith\",\"addresses\":[{\"line1\":\"100 Any Place\",\"line2\":\"Apt A\",
\"city\":\"Seattle\",\"state\":\"WA\",\"zip\":98012}],\"contactDetails\“:[{\"email\":\"Bill@smith.com\"},
{\"phone\":\"+1 816 555-1234\",\"extension\":3}]}","%DocumentId":"2",
"%LastModified":"2017-08-14 21:39:35.895"}]}

Working with DocDB – find a document

- Search with a projection request returns only those properties

```
set result=db.%FindDocuments(["firstName","Bill","="],["%DocumentId","firstName"])
write result.%ToJSON()
```

{"sqlcode":100,"message":null,"content":[{"%DocumentId":"2","firstName":"Bill"}]}

- Search with a return limit

```
set result=db.%FindDocuments(["firstName","Bill","="],["%DocumentId","firstName"] ,{"limit":5})
```


Working with DocDB – delete a document

- Delete a document using the documentId, see also %DeleteDocumentByKey

```
set sc = db.%DeleteDocument(2)
```


REST Interface - Databases

Base URL: http://localhost:57777/api/docdb/v1

API	HTTP Method	URI
GetAllDatabases	GET	/namespaceName
DropAllDatabases	DELETE	/namespaceName
CreateDatabase	POST	/namespaceName/db/databaseName
DropDatabase	DELETE	/namespaceName/db/databaseName
GetDatabase	GET	/namespaceName/db/databaseName

REST Interface – Database Properties

Base URL: http://localhost:57777/api/docdb/v1

API	HTTP Method	URI
CreateProperty	POST	/*namespaceName*/prop/*databaseName*/*propertyName*?type=*propertyType*&path=*propertyPath*&unique=*propertyUnique*
DropProperty	DELETE	/*namespaceName*/prop/*databaseName*/*propertyName*
GetProperty	GET	/*namespaceName*/prop/*databaseName*/*propertyName*

REST Interface - Documents

Base URL: http://localhost:57777/api/docdb/v1

API	HTTP Method	URI
SaveDocument	POST	/*namespaceName*/doc/*databaseName*/
SaveDocument	PUT	/*namespaceName*/doc/*databaseName*/*id*
SaveDocumentByKey	PUT	/*namespaceName*/doc/*databaseName*/ keyPropertyName/keyValue
DeleteDocument	DELETE	/*namespaceName*/doc/*databaseName*/*id*
DeleteDocumentByKey	DELETE	/*namespaceName*/doc/*databaseName*/ keyPropertyName/keyValue

REST Interface - Documents

Base URL: http://localhost:57777/api/docdb/v1

API	HTTP Method	URI
FindDocuments	POST	/*namespaceName*/find/*databaseName*
GetAllDocuments	POST	/*namespaceName*/find/*databaseName*
GetDocument	GET	/*namespaceName*/doc/*databaseName*/id
GetDocumentByKey	POST	/*namespaceName*/doc/*databaseName*/*keyPropertyName*/*keyValue*

CURL Example – get all documents

```
curl --user _SYSTEM:SYS -w "\n\n%{http_code}\n" -H "Accept: application/json" -H "Content-Type: application/json" -X POST http://localhost:57777/api/docdb/v1/samples/find/continents
```

{"content":{"sqlcode":100,"message":null,"content":[{"%Doc":"{\"code\":\"NA\",\"name\":\"North America\"}","%DocumentId":"1","%LastModified":"2017-08-14 19:42:16.943"},{"%Doc":"{\"code\":\"SA\",\"name\":\"South America\"}","%DocumentId":"2","%LastModified":"2017-08-14 19:42:16.944"},{"%Doc":"{\"code\":\"AF\",\"name\":\"Africa\"}","%DocumentId":"3","%LastModified":"2017-08-14 19:42:16.944"},{"%Doc":"{\"code\":\"AS\",\"name\":\"Asia\"}","%DocumentId":"4","%LastModified":"2017-08-14 19:42:16.944"},{"%Doc":"{\"code\":\"EU\",\"name\":\"Europe\"}","%DocumentId":"5","%LastModified":"2017-08-14 19:42:16.944"},{"%Doc":"{\"code\":\"OC\",\"name\":\"Oceana\"}","%DocumentId":"6","%LastModified":"2017-08-14 19:42:16.944"},{"%Doc":"{\"code\":\"AN\",\"name\":\"Antarctica\"}","%DocumentId":"7","%LastModified":"2017-08-14 19:42:16.944"}]}}

CURL Example – get all documents starting with A

```
curl --user _SYSTEM:SYS -w "\n\n%{http_code}\n" -H "Accept: application/json" -H
"Content-Type: application/json" -X POST
http://localhost:57772/api/docdb/v1/samples/find/continents -d
'{"restriction":["Name","A","%STARTSWITH"]}'
```

{"content":{"sqlcode":100,"message":null,"content":[{"%Doc":"{\"code\":\"AF\",\"name\":\"Africa\"}","%DocumentId":"3","%LastModified":"2017-08-15 21:33:03.64"},{"%Doc":"{\"code\":\"AS\",\"name\":\"Asia\"}","%DocumentId":"4","%LastModified":"2017-08-15 21:33:03.64"},{"%Doc":"{\"code\":\"AN\",\"name\":\"Antarctica\"}","%DocumentId":"7","%LastModified":"2017-08-15 21:33:03.64"}]}}

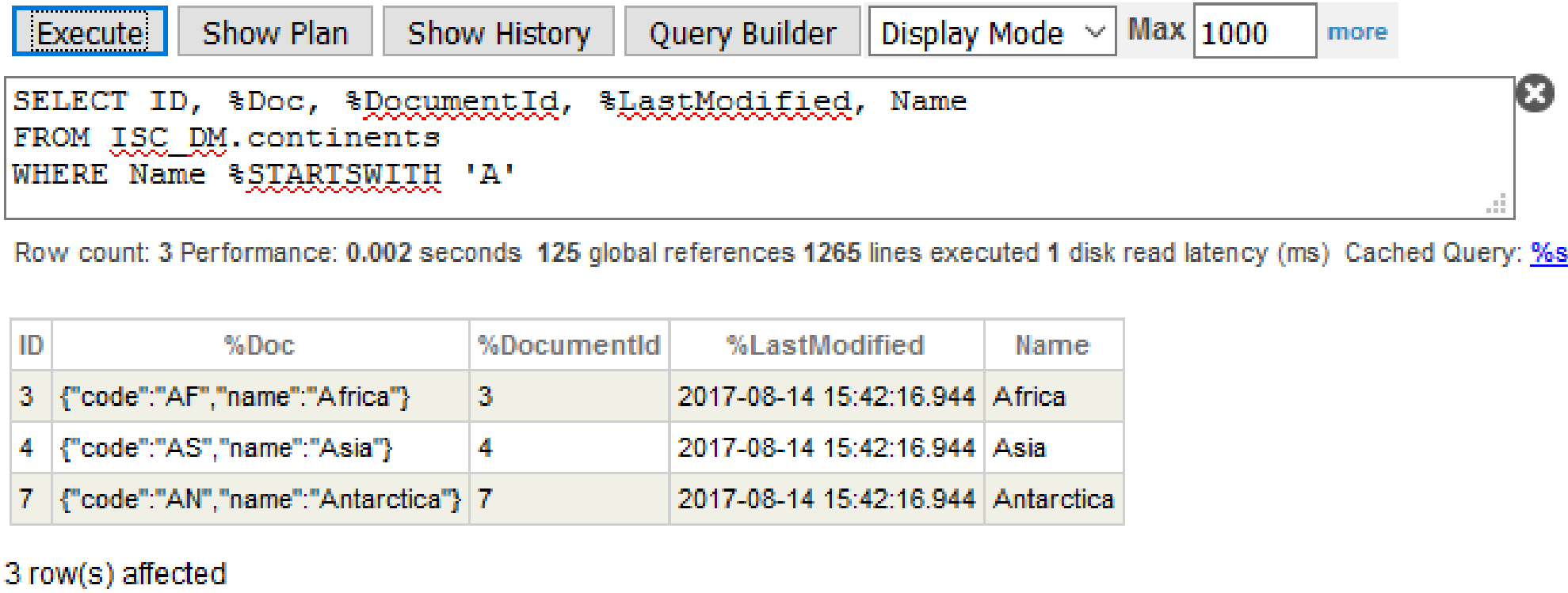
Document Database Properties

Add properties to use with a restriction or to use in an SQL query.

```
Class ISC.DM.Wx Extends %DocDB.Document [ Owner = {UnknownUser}, ProcedureBlock ]
{

Property city As %String [ SqlComputeCode = {
set {*}=$$%EvaluatePathOne^%DocDB.Document({%Doc},"query.results.channel.location.city")
}, SqlComputed, SqlComputeOnChange = %Doc ];

Property state As %String [ SqlComputeCode = {
set {*}=$$%EvaluatePathOne^%DocDB.Document({%Doc},"query.results.channel.location.region")
}, SqlComputed, SqlComputeOnChange = %Doc ];

Index city On city As SQLUPPER(120) [ Not Unique ];

Index state On state As SQLUPPER(120) [ Not Unique ];
```


SQL Queries

Use the properties set up in the document database to write queries:



Demo

In summary

- Document databases are:
  - Flexible
  - Scalable
  - Agile
  - Sparse and typeless

- Intersystems IRIS implementation
  - Uses JSON as the document structure
  - Provides document API
  - Provides REST API
  - Can use SQL to access documents via properties

Where to go from here…

While you are at the Solutions Developers Conference:

- Engage in detailed technical discussions by contacting *Dan Pasco* in the Tech Exchange
- Contact me Chris Carmichael or with the the PM *Stefan Whitman*

Learn how to use the technology showcased in this session by visiting:

- https://learning.intersystems.com/ (Online Learning)
- http://www.intersystems.com/services-support/learning-services/ (Classroom Training)
- http://docs.intersystems.com/ (Documentation)

Contact your Sales Engineer for additional post conference engagements

The power behind what matters.

InterSystems®
Health | Business | Government

Thank you.