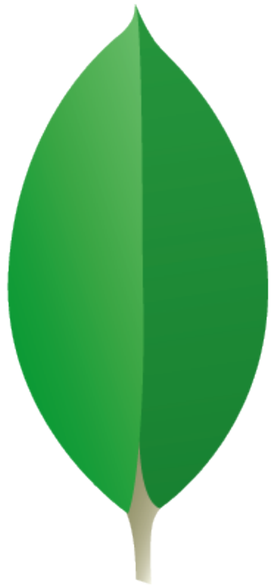
MongoDB for C# Developers

Simon Elliston Ball
Head of Big Data

@sireb

redgate
ingeniously simple



mongoDB

http://www.mongodb.org/

Document Database

Document Database

id	full_name	address
1	John Smith	3a Test Street
2	Jane Doe	1b Fake Street
3	...	...

Document Database

id	full_name	address
1	John Smith	3a Test Street
2	Jane Doe	1b Fake Street
3	...	...

id	customer_i		order_date
1	1	...	2013-10-1
2	1	...	...
3	...	...	...

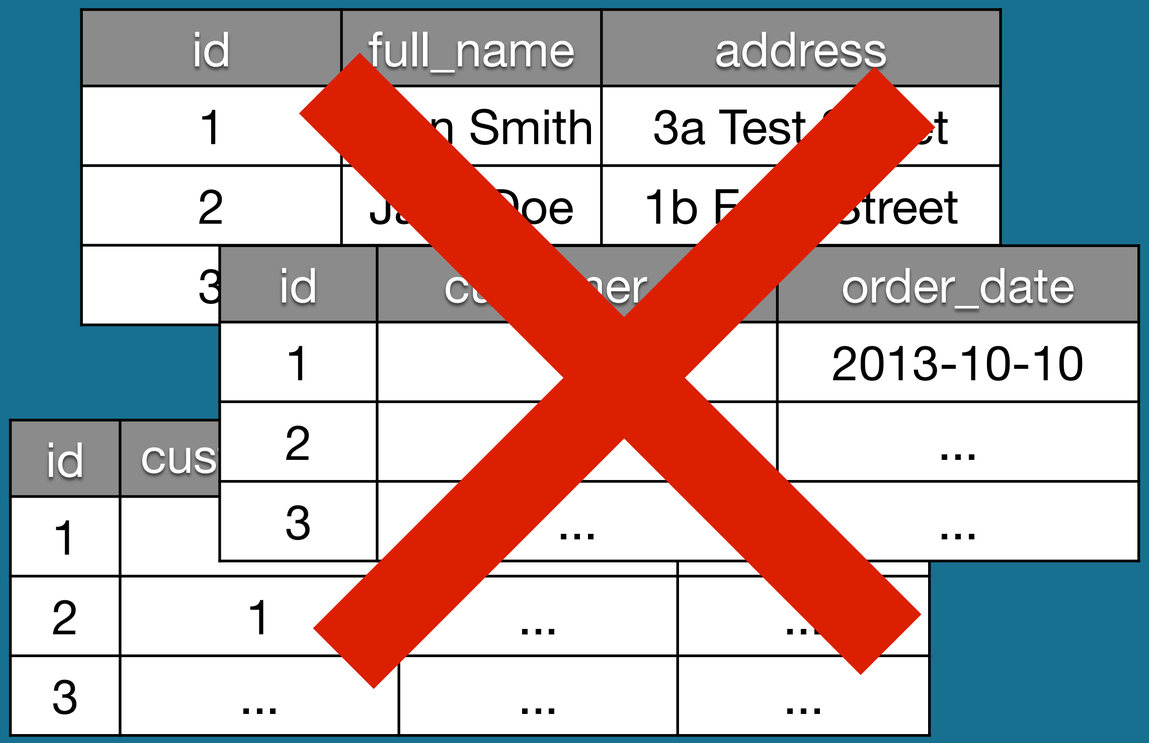
Document Database

id	full_name	address
1	John Smith	3a Test Street
2	Jane Doe	1b Fake Street
3		

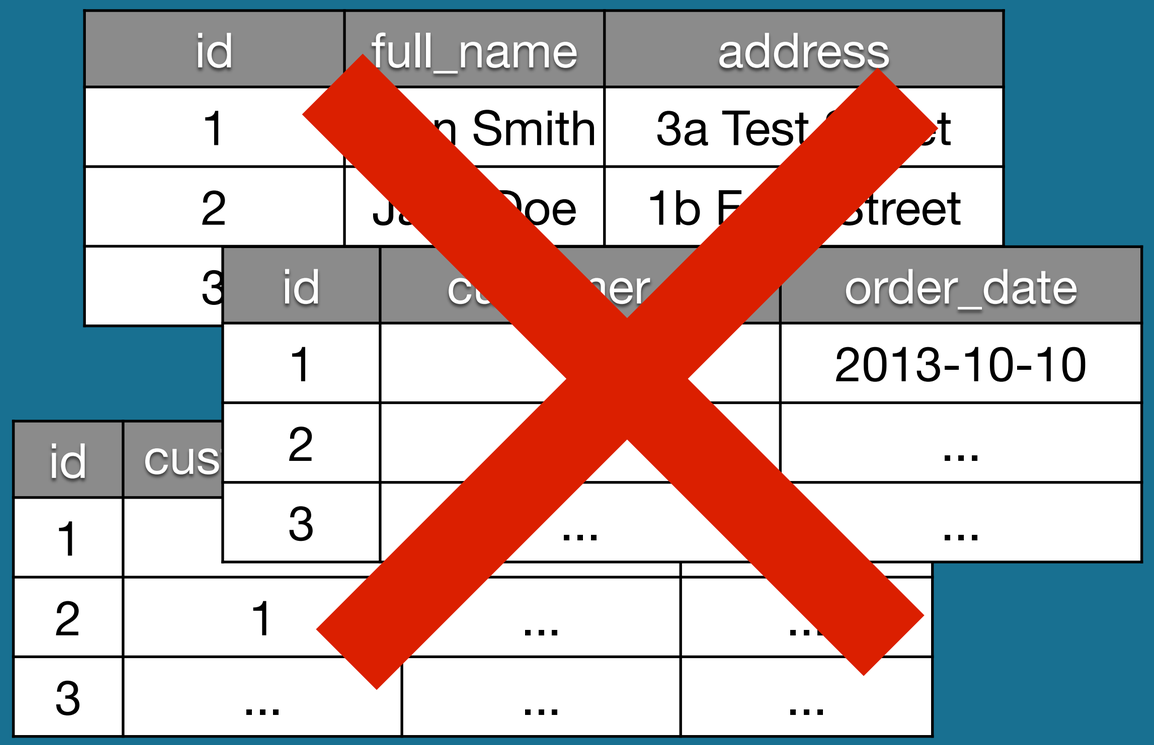
id	order_id	product_id
1	1	103
2	1	...
3	...	...

id	cus		
1			
2	1	...	...
3	...	...	...

Document Database



Document Database




```
customers = [
   {
      "_id" : ObjectId("5256b399ac46b80084974d9a"),
      "name" : "John Smith",
      "address" : "3a Test Street",
      "orders" [ {
         "order_date": "2013-10-10",
         "order_item": [
            { "product": "Widget"...}
            ...
         ]
         ...
      }]
   },
   {
      "_id" : ObjectId("5256b3a8ac46b80084974d9b"),
      "name" : "Jane Doe",
      "address" : "1b Fake Street"
   }
]
```

Key -> JSON

Document Database - Why?

Flexible Data Model - No Schema

Rapid Development

Scalability

Document Database - When?

Blogs

Document Database - When?

- Content management
- When read patterns are fixed
- Scaling requirements are unknown
- When write speed matters


mongoDB


mongoDB


RAVENDB


mongoDB


RAVENDB

- Transactions per document
- Master-slave replication
- Many many languages: C#,

  JavaScript, Java, PHP, Python, Ruby,

  Scala, Erlang, Go, C, C++, Perl (and those are just the official ones)

- ACID, multi-document
- But… Background Indexing
- Master-master replication
- .NET Only

mongoDB

- Main interface: CLI
- Transactions per document
- Master-slave replication
- Many many languages: C#,

  JavaScript, Java, PHP, Python, Ruby,

  Scala, Erlang, Go, C, C++, Perl (and those are just the official ones)

Couchbase

- GUI Client
- Native JSON
- Geo-aware replication
- REST Interface, so anything
- View based queries

Getting started with MongoDB

Download from http://www.mongodb.org/

mongoDB

{name: "mongo", type:"DB"}

MONGO DOCS
MongoDB Documentation »

TRY IT OUT
Try The Online Shell »

DOWNLOADS
Download MongoDB »

DRIVERS
Get The Latest Drivers »

Agile and Scalable

MongoDB (from "humongous") is an open-source document database, and the leading NoSQL database. Written in C++, MongoDB features:

- **Document-Oriented Storage »**
  JSON-style documents with dynamic schemas offer simplicity and power.
- **Full Index Support »**
  Index on any attribute, just like you're used to.
- **Replication & High Availability »**
  Mirror across LANs and WANs for scale and peace of mind.
- **Auto-Sharding »**
  Scale horizontally without compromising functionality.
- **Querying »**
  Rich, document-based queries.

Newsletter Signup

Keep up to date with MongoDB!

E-Mail Address

Sign Up

MongoDB 2.4 is now available

Download MongoDB 2.4

Upcoming Events

Oct 10	Webinar: Schema Design
Oct 11	MongoDB Paris
Oct 14	MongoDB Munich

Getting started with the C# client

```
PM> Install-Package mongocsharpdriver
```

File Edit View Project Build Debug Team SQL SQL Prompt Tools VMware ANTS Test .NET Reflector SQL Connect Analyze Window Help

Start Debug Synchronize...

Performance Explorer Server Explorer

Program.cs*

MongoSamples.Program Main(string[] args)

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace MongoSamples
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13         }
14     }
15 }
16
```

100 %

Output Find Symbol Results Code Coverage Results Data Tools Operations

Solution Explorer

Search Solution Explorer (Ctrl+;)

- Solution 'MongoSamples' (1 project)
  - MongoSamples
    - Properties
    - References
      - Microsoft.CSharp
      - System
      - System.Core
      - System.Data
      - System.Data.DataSetExtensions
      - System.Xml
      - System.Xml.Linq
    - App.config
    - Program.cs

Team Explorer Solution Explorer Properties

Wooah there.

I thought you said JSON...

BSON Binary JSON

BSON Binary JSON

```
[Serializable]
public enum BsonType
{
    Double = 0x01,
    String = 0x02,
    Document = 0x03,
    Array = 0x04,
    Binary = 0x05,
    Undefined = 0x06,
    ObjectId = 0x07,
    Boolean = 0x08,
    DateTime = 0x09,
    Null = 0x0a,
    RegularExpression = 0x0b,
    JavaScript = 0x0d,
    Symbol = 0x0e,
    JavaScriptWithScope = 0x0f,
    Int32 = 0x10,
    Timestamp = 0x11,
    Int64 = 0x12,
    MinKey = 0xff,
    MaxKey = 0x7f
}
```

Typed

BSON Binary JSON

```
[Serializable]
public enum BsonType
{
    Double = 0x01,
    String = 0x02,
    Document = 0x03,
    Array = 0x04,
    Binary = 0x05,
    Undefined = 0x06,
    ObjectId = 0x07,
    Boolean = 0x08,
    DateTime = 0x09,
    Null = 0x0a,
    RegularExpression = 0x0b,
    JavaScript = 0x0d,
    Symbol = 0x0e,
    JavaScriptWithScope = 0x0f,
    Int32 = 0x10,
    Timestamp = 0x11,
    Int64 = 0x12,
    MinKey = 0xff,
    MaxKey = 0x7f
}
```

Typed

Serialisation library

BSON Binary JSON

```
[Serializable]
public enum BsonType
{
    Double = 0x01,
    String = 0x02,
    Document = 0x03,
    Array = 0x04,
    Binary = 0x05,
    Undefined = 0x06,
    ObjectId = 0x07,
    Boolean = 0x08,
    DateTime = 0x09,
    Null = 0x0a,
    RegularExpression = 0x0b,
    JavaScript = 0x0d,
    Symbol = 0x0e,
    JavaScriptWithScope = 0x0f,
    Int32 = 0x10,
    Timestamp = 0x11,
    Int64 = 0x12,
    MinKey = 0xff,
    MaxKey = 0x7f
}
```

Typed

Serialisation library

Annotate POCOs to control mapping

or write config code if you must

Connecting...

```
var connectionString = "mongodb://localhost";
var client = new MongoClient(connectionString);
var server = client.GetServer();
```

Connecting...

```
var connectionString = "mongodb://localhost";
var client = new MongoClient(connectionString);
var server = client.GetServer();
```

That’s it.

The driver will disconnect,
release objects and pool for you.

But...

Collections

Database

Collection

Document

_id

field…

CRUD creating new documents

With mapped entities:

```
var developerCollection = database.GetCollection<Developer>("team");

var Developer = new Developer(1,"Test", "Person");
developerCollection.Insert(Developer);
var Developer2 = new Developer(2,"Another", "Developer");
developerCollection.Insert(Developer2)
```

The BsonDocument way:

```
var documentCollection = database.GetCollection("team");

BsonDocument document = new BsonDocument();
document.Add(new BsonElement("name", "Testing"))
        .Add(new BsonElement("number", new BsonInt32(42)));

documentCollection.Insert(document);
```

CRUD creating new documents

The BsonDocument way:

```
var documentCollection = database.GetCollection("team");

BsonDocument document = new BsonDocument();
document.Add(new BsonElement("name", "Testing"))
        .Add(new BsonElement("number", new BsonInt32(42)));

documentCollection.Insert(document);
```

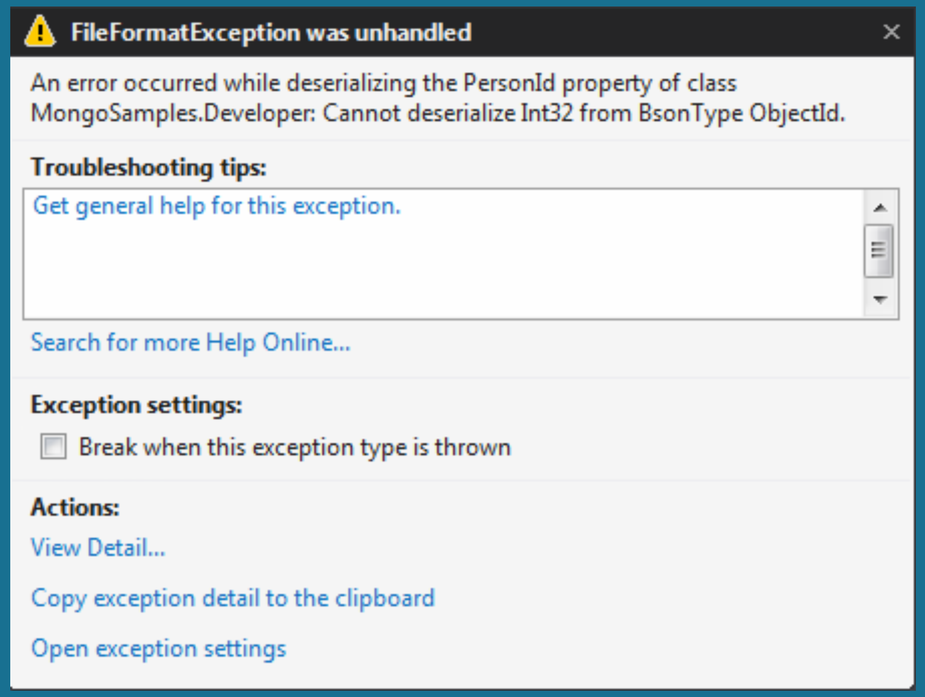
Beware of mixing your BSONs

```
List<Developer> allDevelopers = developerResults.ToList<Developer>();
```

CRUD creating new documents

Beware of mixing your BSONs

```
List<Developer> allDevelopers = developerResults.ToList<Developer>();
```



CRUD basic document reads

```
MongoCursor<BsonDocument> documentResults = documentCollection.FindAll();
MongoCursor<Developer> developerResults = developerCollection.FindAll();
```

CRUD basic document reads

```
MongoCursor<BsonDocument> documentResults = documentCollection.FindAll();
MongoCursor<Developer> developerResults = developerCollection.FindAll();
```

```
var cursor = collection.FindAll();
cursor.Skip = 100;
cursor.Limit = 10;

foreach (var developer in cursor) {
    ...
```

CRUD basic document reads

```
MongoCursor<BsonDocument> documentResults = documentCollection.FindAll();
MongoCursor<Developer> developerResults = developerCollection.FindAll();
```

```
var cursor = collection.FindAll();
cursor.Skip = 100;
cursor.Limit = 10;

foreach (var developer in cursor) {
    ...
}
```

```
var readQuery = Query<Developer>.EQ(n => n.PersonId, 2);
Developer developerRead = developerCollection.FindOne(readQuery);
```

CRUD basic document reads

```
MongoCursor<BsonDocument> documentResults = documentCollection.FindAll();
MongoCursor<Developer> developerResults = developerCollection.FindAll();
```

```
var cursor = collection.FindAll();
cursor.Skip = 100;
cursor.Limit = 10;

foreach (var developer in cursor) {
    ...
}
```

```
var readQuery = Query<Developer>.EQ(n => n.PersonId, 2);
Developer developerRead = developerCollection.FindOne(readQuery);
```

```
BsonDocument documentRead = documentCollection.FindOne(new QueryDocument {
    { "_id", documentId }
});
```

CRUD update

```
developerRead.LastName = "Something-Else";
developerCollection.Save(developerRead);
```

CRUD update

Write Concerns

Only relevant with Replication

The number of replicas which need to report successful writes

```
collection.Save(developerRead, new MongoInsertOptions
    {
        WriteConcern = WriteConcern.WMajority
    }
);
```

CRUD update

```
var update = new UpdateDocument {
    { "$set", new BsonDocument("LastName", "A new name") }
};

var query = new QueryDocument {
    { "LastName", "Developer" }
};

collection.Update(query, update);
```

NB. Only updates one document

CRUD update

```
var update = new UpdateDocument {
    { "$set", new BsonDocument("LastName", "A new name") }
};

var query = new QueryDocument {
    { "LastName", "Developer" }
};

collection.Update(query, update);
```

NB. Only updates one document

```
collection.Update(query, update, new MongoUpdateOptions
{
    Flags = UpdateFlags.Multi
});
```

Applies to all documents that match query

CRUD upsert

```
var update = new UpdateDocument {
    { "$set", new BsonDocument("LastName", "A new name") }
};
var query = Query<Developer>.EQ(d => d.PersonId, 10);

collection.Update(query, update, new MongoUpdateOptions
{
    Flags = UpdateFlags.Upsert
});
```

CRUD deleting

```
var query = new QueryDocument {
    { "LastName", "Person" }
};
collection.Remove(query);
```

CRUD deleting

```
var query = new QueryDocument {
    { "LastName", "Person" }
};
collection.Remove(query);
```

```
collection.RemoveAll();
```

```
collection.Drop();
```

LINQ integration

Just make the collection queryable

```
using System.Linq;
using MongoDB.Driver.Linq;

var query =
            from e in collection.AsQueryable()
            where e.LastName == "Person"
            select e;

foreach (var developer in query){
    ...
```

GridFS in C#

16MB document limit

GridFS used to break documents into chunks

```
using (var fs = new FileStream("largeVideo.m4v", FileMode.Open))
{
    database.GridFS.Upload(fs, "largeVideo.m4v");
}
```

```
database.GridFS.Download("test.m4v", "largeVideo.m4v");
```

MapReduce (JavaScript)

```
var map =
    "function() {" +
    "    for (var key in this) {" +
    "        emit(key, { count : 1 });" +
    "    }" +
    "}";

var reduce =
    "function(key, emits) {" +
    "    total = 0;" +
    "    for (var i in emits) {" +
    "        total += emits[i].count;" +
    "    }" +
    "    return { count : total };" +
    "}";

var mr = collection.MapReduce(map, reduce);
```

Yes, it’s a word count. Yes, it’s JavaScript.

Special Queries GeoNear

```
var query = Query.EQ("properties.amenity", new BsonString("pub"));

// here
double lon = 54.9117468;
double lat = -1.3737675;

var earthRadius = 6378.0; // km
var rangeInKm = 3000.0; // km

var options = GeoNearOptions
        .SetMaxDistance(rangeInKm / earthRadius /* to radians */)
        .SetSpherical(true);

var results = collection.GeoNear(query, lat, lon, 10, options);

foreach (var result in results.Hits)
    ...
```

Aggregation Framework

Aggregation Framework

Pipeline based

Aggregation Framework

Pipeline based

Chained **operators**

Aggregation Framework

Pipeline based

Chained **operators**

$project
$match
$limit
$skip
$unwind
$group
$sort
… http://docs.mongodb.org/manual/reference/operator/aggregation/

Aggregation Framework

Demo

Summary

Document databases are simple

BSON annotation makes POCO mapping is easy

CRUD is straight forward

You can use **LINQ**

Hard stuff is possible

Acknowledgements

Resources

The MongoDB C Sharp Language Center:

http://docs.mongodb.org/ecosystem/drivers/csharp/

A tutorial on the driver from MongoDB themselves:

http://docs.mongodb.org/ecosystem/tutorial/use-csharp-driver/#csharp-driver-tutorial

Sample code from this talk:

https://github.com/simonellistonball/MongoForCsharpSamples

A good walkthrough on MongoDB with ASP.NET MVC:

http://www.drdobbs.com/database/mongodb-with-c-deep-dive/240152181

Bonus extras

A Glimpse plugin to view mongo query details:

https://github.com/simonellistonball/Glimpse.MongoDB

redgate

ingeniously simple

Questions?

Simon Elliston Ball
simon.ellistonball@red-gate.com

@sireb

http://bit.ly/MongoForCsharp

redgate
ingeniously simple