embarcadero

CodeRage® X

*Develop Anything, Anytime, Anywhere*

# FireDAC & MongoDB Introduction

**Dmitry Arefiev & Jim McKeeth**

# The Plan

- MongoDB overview
- New JSON / BSON RTL
- New FireDAC MongoDB API wrapping classes
- New FireDAC MongoDB datasets
- Questions & answers

# MongoDB. Introduction.

- MongoDB is a leading NoSQL, JSON-document oriented, highly scalable, simple setup, open source, free database.
- Runs on Windows, OS X, Linux and Solaris.
- Comes in both 32-bit and 64-bit architecture.

| RDBMS | MongoDB |
|---|---|
| Catalog / Database | Database |
| Table | Collection |
| Record | Document |
| SQL | CRUD = Insert, Find, Update, Delete, etc. With JSON-encoded arguments. |
| SELECT (joins, nested SELECT's, etc) | Single collection Find (no joins) |
| Transaction / ACID | Single document / ACID |
| Foreign key | References |
| SQL console app | Mongo.exe JavaScript console |

# Why MongoDB?

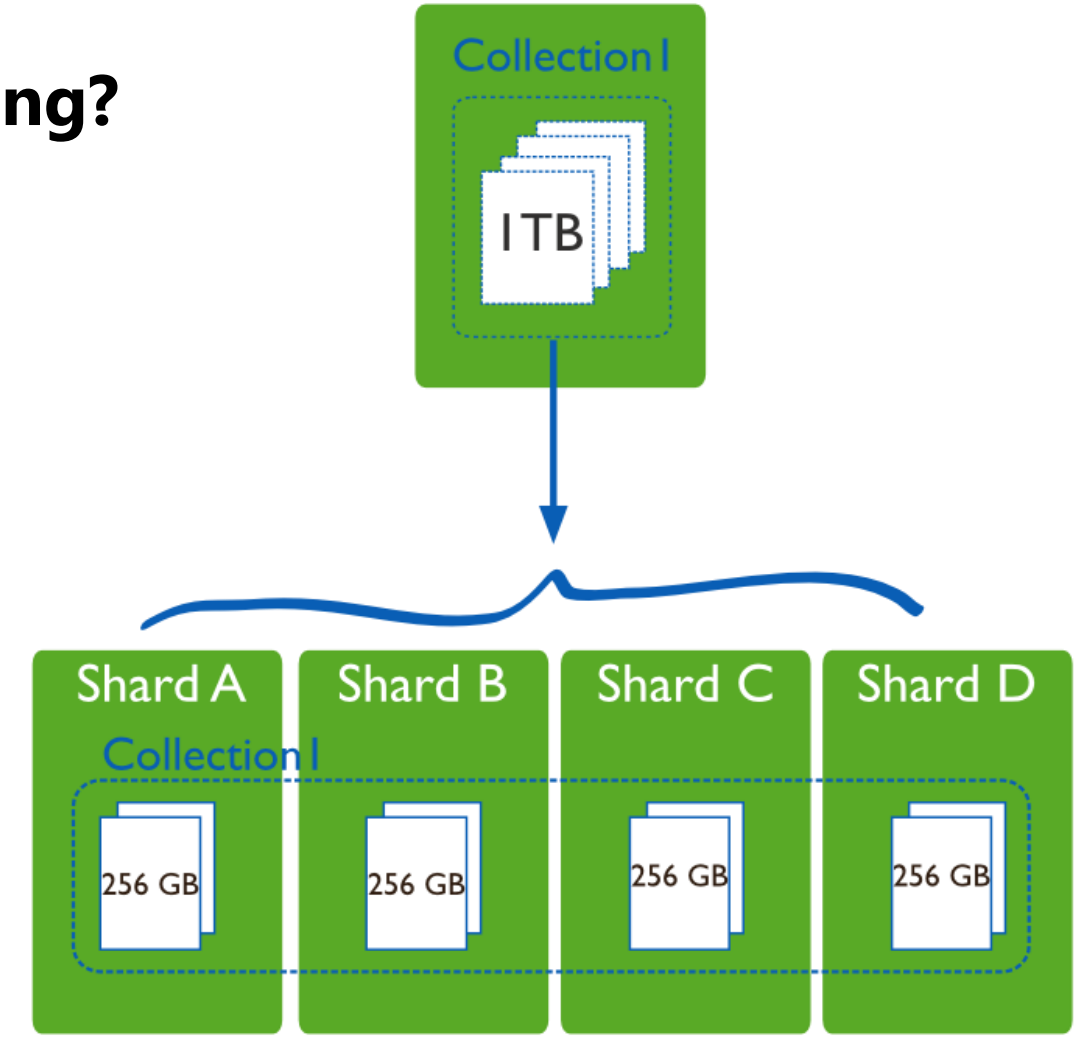
- Horizontally scalable (easily runs across many computers – cluster friendly)
- High availability for write heavy operations (no transactions)
- Supports very large data – Built in auto-sharding
- Location based query support (latitude and longitude distances)
- Schema-less (no enforced schema)
  - Great for Irregular data – vs “impedance mismatch” of normalization
  - Doesn’t require DBA to add column
  - No change management for schema changes
- Non-relational – Probably the most significant element

# MongoDB/NoSQL is a *different* type of Database Better for *certain usages*, and worse for others

- MongoDB
  - Schema-less
  - Programmer controlled
  - Auto-Sharding
  - Location based queries
  - Native JavaScript interactions
  - Horizontally scalable
- RDBMS like InterBase
  - Ridged schema
  - DBA controlled
  - Complex transactions
  - Referential integrity
  - Standard SQL support
  - Embeddable

# What is Sharding / Horizontal Scaling?

- Divides the data set and distributes the data over multiple servers, or shards. Each shard is an independent database, and collectively, the shards make up a *single logical database*.
- Used to support deployments with very large data sets and high throughput operations.



# MongoDB. Installation.

- Straight forward install
  - https://www.mongodb.org/downloads (32-bit & 64-bit)
- Setup environment – folder for database
  - Default is c:\data\db
- Start MongoDB: mongod.exe *[options]*
  - Or setup as a service . . .
- Tutorial
  - http://embt.co/install-mongodb-windows [MongoDB.org]
  - http://embt.co/connect-mongodb [DocWiki]

# MongoDB. Documents.

- NoSQL = No Schema:
  - Documents in a collection may have any / dynamic structure
  - Documents in a collection still normally have some common elements
- All around BSON = Binary JSON:
  - More data types (OID, Binary, Long numbers, Dates)
  - Faster reading / writing
  - Faster navigation
  - 1-to-1 mapping between BSON and Extended JSON
- A Document:
  - Any valid BSON object
  - Each document has unique "_id: <value>" pair
  - Any document keys may be indexed
  - Any document keys may be used to find a document
  - Document may be replaced in full or updated partially
  - Each document operation is atomic

# RTL. JSON / BSON support.

- New JSON/BSON RTL:
  - **TJsonTextWriter** / **TBsonWriter** – write streams
  - **TJsonTextReader** / **TBsonReader** – read stream
  - **TJsonObjectBuilder** – “fluent” style JSON objects builder
  - **TJsonIterator** – fast forward-only JSON iterator
- New MongoDB specific classes:
  - **TMongoDocument** – represents a MongoDB document, provides simplified construction API, provides document builder and iterator

# FireDAC. MongoDB. Overview.

- New FireDAC driver.

```
DriverID=Mongo
Server=127.0.0.1
```

- No
  - SQL = No **TFDCommand**, **TFDQuery**, **TFDTable**, **TFDStoredProc**, **TFDTableAdapter**, **TFDSchemaAdapter**, **TFDScript**, **TFDMetaInfoQuery**
  - TX = No **TFDTransaction**
- Yes
  - **TFDConnection**
  - **FireDAC.Phys.MongoDBWrapper.pas** – API wrapping and command builder classes
  - **FireDAC.Phys.MongoDBDataSet.pas** – MongoDB specific dataset components
  - **TFDEventAlerter** (coming), **TFDLocalSQL**, **TFDBatchMove**

# MongoDB. Documents. Creating.

```
db.Restaurants.Insert({name: "Vella", address: {street: "2 Avenue",
building: "1480", coord: [-73.95, 40.77]}, …})
```

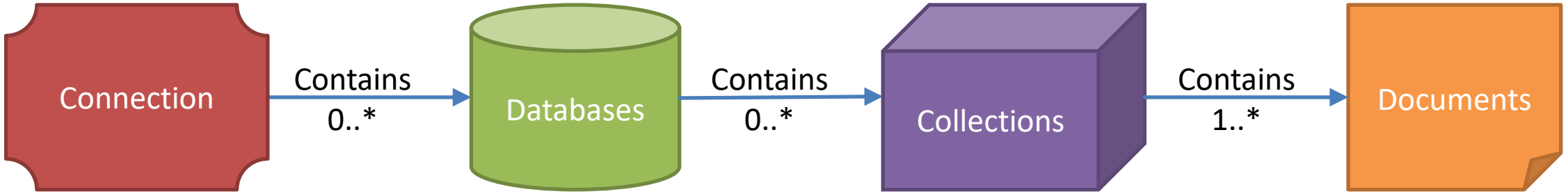
- **_id** pair will be added automatically
- WriteConcern may specify what MongoDB will “guarantee” at end of write operation
  - http://docs.mongodb.org/manual/core/write-concern/
  - WriteConcern property on Database, Connection and Collection.
- Batch insertion can be used. Similar to FireDAC ArrayDML.

# MongoDB. Documents. Reading.

```
db.Restaurants.Find({})
db.Restaurants.Find({"address.street": "2 Avenue"})
```

- Flexible selection criteria “language”, which is a JSON document, including:
  - Projection – similar to SELECT list
  - Match – similar to WHERE
  - Sort – similar to ORDER BY
- No joins
- Returns a cursor with JSON documents

# Architecture




- Collections exist while they contain documents
- Documents within collection don’t need a consistent schema, but typically are similar
- Documents are made of fields with types and values

# FireDAC. MongoDB. Wrapping classes.

- Main way to MongoDB. Major classes:
  - TMongoEnv – “root” utility class
  - TMongoConnection – connection API
  - TMongoDatabase – database API
  - TMongoCollection – collection API (all CRUD operations)
  - TMongoDocument – document API
  - TMongoInsert, TMongoUpdate, TMongoQuery, TMongoPipeline, etc – “fluent” style command builders

```
uses
  FireDAC.Phys.MongoDBWrapping;
…
FDConnection1.Connected := True;
FCon := TMongoConnection(FDConnection1.CliObj);
FEnv := FCon.Env;
```

# FireDAC. MongoDB. Inserting. Non-fluent.

- Useful when document / command builder is used in different code places, subroutines. IOW, we need an explicit reference to object.

```
oDoc := FEnv.NewDoc; // TMongoDocument
try
  oDoc
    .Add('name', 'Vella')
    .BeginObject('address')
      .Add('street', '2 Avenue')
      .BeginArray('coord')
        .Add('0', -73.9557413)
        .Add('1', 40.7720266)
      .EndArray
    .EndObject;
  FCon['test']['restaurants'].Insert(oDoc);
finally
  oDoc.Free;
end;
```

# FireDAC. MongoDB. Inserting. Fluent.

- Useful when document / command builder is used in single code place. IOW, we do not need an explicit reference to object.

```
FCon['test']['restaurants'].Insert() // returns TMongoInsert builder
  .Values()
    .Add('name', 'Vella')
    .BeginObject('address')
      .Add('street', '2 Avenue')
      .BeginArray('coord')
        .Add('0', -73.9557413)
        .Add('1', 40.7720266)
      .EndArray
    .EndObject
  .&End
.Exec;
```

# FireDAC. MongoDB. Querying.

- IMongoCursor interface – represent MongoDB cursor
- Get all documents:

```
oCrs := FCon['test']['restaurants'].Find();
while oCrs.Next do
  Memo1.Lines.Add(oCrs.Doc.AsJSON);
```

- Get filtered and sorted documents, fluent style:

```
oCrs := FCon['test']['restaurants'].Find() // returns TMongoQuery builder
  .Match
    .Add('address.street', '2 Avenue')
  .&End
  .Sort
    .Field('name', True)
  .&End;
while oCrs.Next do
  Memo1.Lines.Add(oCrs.Doc.AsJSON);
```

# FireDAC. MongoDB. Reading.

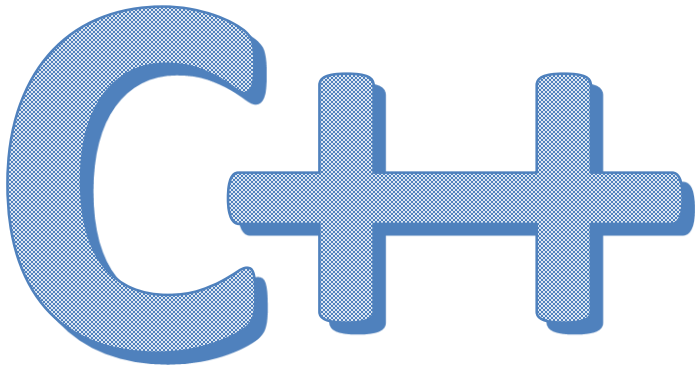
- **TJSONIterator** provides read-only forward-only iterator for JSON/BSON document content:
  - **Next** - to next element of the same level
  - **Recurse** - enter into nested object or array
  - **Return** - return to parent object or array
  - **Key**, **&Type**, **AsXxxx** –read content of element

```
oIter := oCrs.Doc.Iterator;
try
  if oIter.Find('address.coord') then begin
    oIter.Recurse;                    // "enter" into 'coord'
    oIter.Next;                       // go to 'coord[0]'
    P.lat := oIter.AsDouble;          // read 'coord[0]' value
    oIter.Next;                       // go to 'coord[1]'
    P.long := oIter.AsDouble;         // read 'coord[1]' value
  end;
finally
  oIter.Free;
end;
```

# FireDAC. MongoDB. Wrapping classes.

- Main way to MongoDB. Major classes:
  - TMongoEnv – “root” utility class
  - TMongoConnection – connection API
  - TMongoDatabase – database API
  - TMongoCollection – collection API (all CRUD operations)
  - TMongoDocument – document API
  - TMongoInsert, TMongoUpdate, TMongoQuery, TMongoPipeline, etc – “fluent” style command builders

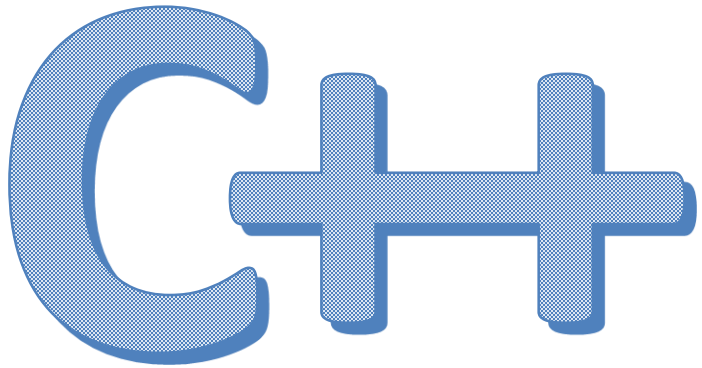
```
#include <FireDAC.Phys.MongoDBWrapper.hpp>
…
  FDConnection1->Open();
  FCon = (TMongoConnection*)FDConnection1->CliObj;
  FEnv = FCon->Env;
```



# FireDAC. MongoDB. Inserting. Non-fluent.

- Useful when document / command builder is used in different code places, subroutines. IOW, we need an explicit reference to object.

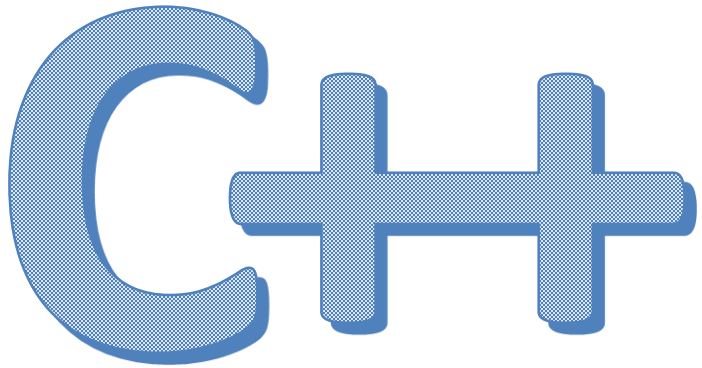
```
TMongoDocument *oDoc = FEnv->NewDoc();
 try {
   oDoc
   ->Add("name", "Vella")
   ->BeginObject("address")
     ->Add("street", "2 Avenue")
     ->BeginArray("coord")
       ->Add("0", -73.9557413)
       ->Add("1", 40.7720266)
     ->EndArray()
   ->EndObject();
   // Other interactions with oDoc
   FCon->GetDatabase("test")->GetCollection("restaurants")->Insert(oDoc);
 }
  __finally {
   oDoc->Free();
 }
```



# FireDAC. MongoDB. Inserting. Fluent.

- Useful when document / command builder is used in single code place. IOW, we do not need an explicit reference to object.

```
TMongoDatabase *db = FCon->GetDatabase("test");
TMongoCollection *col = db->GetCollection("restaurants");
col->Insert() // returns TMongoInsert builder
  ->Values()
    ->Add("name", "Vella")
    ->BeginObject("address")
      ->Add("street", "2 Avenue")
      ->BeginArray("coord")
        ->Add("0", -73.9557413)
        ->Add("1", 40.7720266)
      ->EndArray()
    ->EndObject()
  ->End()
->Exec();
```



# FireDAC. MongoDB. Reading.

- **TJSONIterator** provides read-only forward-only iterator for JSON/BSON document content:
  - **Next** - to next element of the same level
  - **Recurse** - enter into nested object or array
  - **Return** - return to parent object or array
  - **Key**, **&Type**, **AsXxxx** –read content of element

# FireDAC. MongoDB. More.

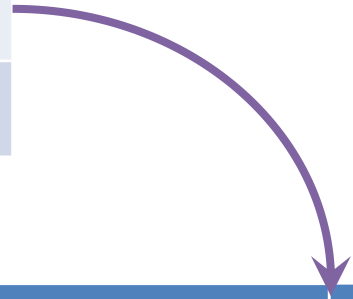
- **TMongoCollection.Update**, **TMongoUpdate**
  - Documents updating method and builder
- **TMongoCollection.Delete**, **TMongoSelector**
  - Documents deleting method and builder
- **TMongoCollection.Aggregate**, **TMongoPipeline**
  - Documents aggregating method and builder
- And more …

# FireDAC. MongoDB. Datasets. Overview.

- Conflict:
  - MongoDB documents - no schema
  - TDataSet - schema is mandatory
- Expectations:
  - Optimistically about an average collection:
    - The same named document keys – the same data semantic
    - The same named document keys – the same data type family
- Solution:
  - Scan first N documents and build common schema
  - This works for nested objects and arrays too

# FireDAC. MongoDB. Datasets. Example.

| JSON |
| --- |
| {name: ‘Dmitry’, kind: ‘Human, job: ‘Programmer’} |
| {name: ‘Orange’, kind: ‘Fruit’, fat: ‘Low’} |
| {name: ‘Audi’, kind: ’Car’, engine: ‘3tdi’} |



| name | kind | job | fat | engine |
| --- | --- | --- | --- | --- |
| Dmitry | Human | Programmer | <null> | <null> |
| Orange | Fruit | <null> | Low | <null> |
| Audi | Car | <null> | <null> | 3tdi |

# FireDAC. MongoDB. Datasets. Data Types.

- JSON nested object -> **ftADT**
- JSON nested array -> **ftDataSet**
- Unlimited nesting level

```
{ name: “Vella”,
  address: {
    street: “2 Avenue”,
    building: “1480”,
    coord: [
      -73.95,
      40.77
    ] } }
```

```
ftWideString, ‘name’
ftADT, ‘address’
   ftWideString, ‘street’
   ftWideString, ‘building’
   ftDataSet, ‘coord’
      ftDouble, ‘Elem’
```

# FireDAC. MongoDB. Datasets. Classes.

- **TFDMongoDataSet** – attaches to a MongoDB cursor
  - **Connection** - MongoDB TFDConnection
  - **DatabaseName**, **CollectionName** – collection path
  - Scans first 2 * FetchOptions.RecordsetSize
  - Automatic dataset editing
- **TFDMongoQuery** – uses Find to produce cursor
  - **QProject** – JSON string, similar to SELECT
  - **QMatch** – JSON string, similar to WHERE
  - **QSort** – JSON string, similar to ORDER BY
  - … or **Query** – TMongoQuery builder, only at run-time
- **TFDMongoPipeline** – uses Aggregate to produce cursor

# Learning Resources

- MongoDB.org
  - http://embt.co/install-mongodb-windows
  - https://docs.mongodb.org/manual/core/crud-introduction/
- DocWiki
  - http://embt.co/connect-mongodb
- Samples
  - Object Pascal\Database\FireDAC\Samples\DBMS Specific\MongoDB
- Books
  - *Instant MongoDB* by Amol Nayak
  - *MongoDB: The Definitive Guide* by Kristina Chodorow
- **CodeRage session Part 2, coming up next!**