MONGODB

15

15

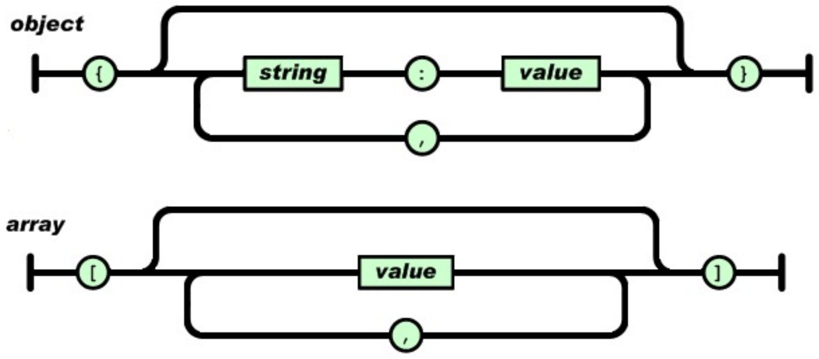
MongoDB vs Relational

- Scale out vs scale up
  - Automatic rebalancing
- Document (JSON) vs row
- Schema-free
- Stored Javascript vs stored procedures
- MapReduce vs SQL
- Files vs tables
- Secondary indexing

16

16

- JavaScript Object Notation (JSON) specify: literal values for types in JavaScript.



value can be any string, number, object, array, or the literal values true, false, or null.

17

Examples

```
var cities = [”Boston",
             ”New York",
             ”Washington D.C."];

var employee =
    {
        "Name": ”Joe",
        "Salary": 50000,
        "Skills": [”Azure",
                  ”Cassandra",
                  ”Hadoop”]
     };
```

18

18

MONGODB ESSENTIALS

19

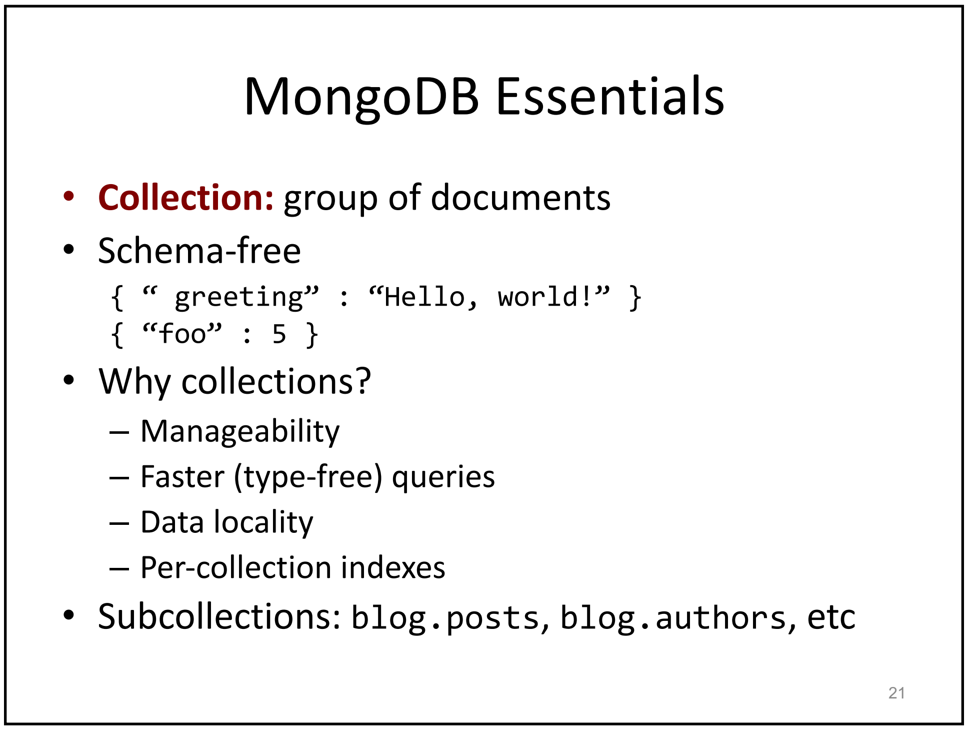
19

MongoDB Essentials

- **Document:** JSON syntax
  - But more types (BSON)!
- Keys are ordered
- No duplicates, names case-sensitive
- Fields have types
- Reserved chars: . And $
- Special key: “_id”

20

20



MongoDB Essentials

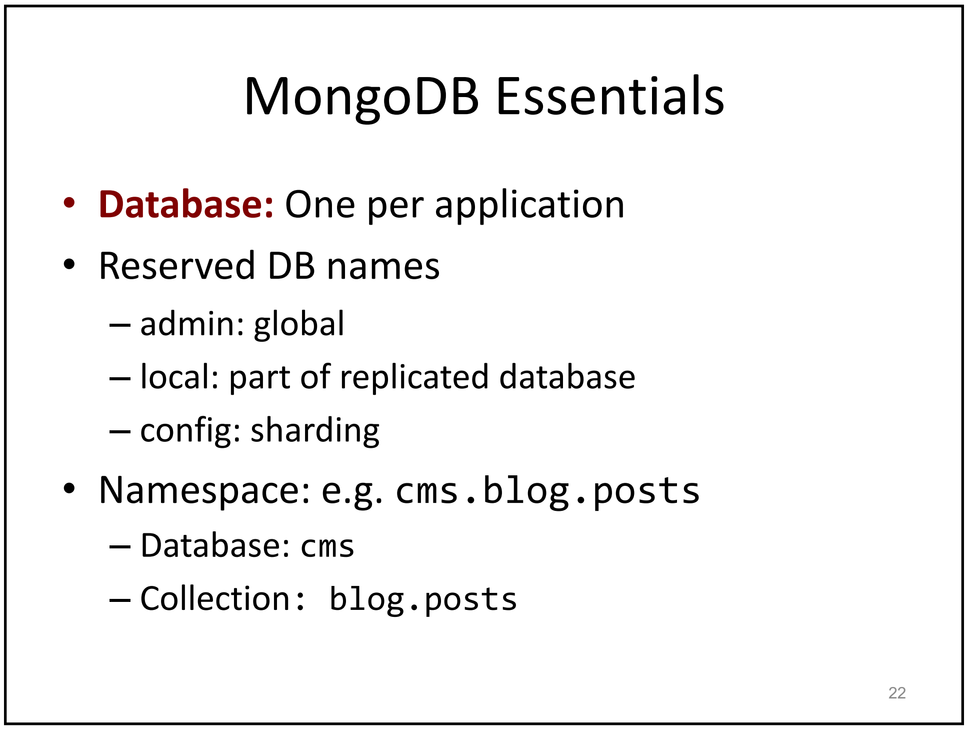
- **Collection:** group of documents
- Schema-free

```
{ " greeting" : "Hello, world!" }
{ "foo" : 5 }
```

- Why collections?
  - Manageability
  - Faster (type-free) queries
  - Data locality
  - Per-collection indexes
- Subcollections: `blog.posts`, `blog.authors`, etc

21

21



MongoDB Essentials

- **Database:** One per application
- Reserved DB names
  - admin: global
  - local: part of replicated database
  - config: sharding
- Namespace: e.g. `cms.blog.posts`
  - Database: `cms`
  - Collection: `blog.posts`

22

22

Running MongoDB

- Server: `mongod`
- Default directory
  - Unix: `/data/db`
  - Windows: `C:\data\db`
- Default port: 27017
  - http: MongoDB port + 1000

23

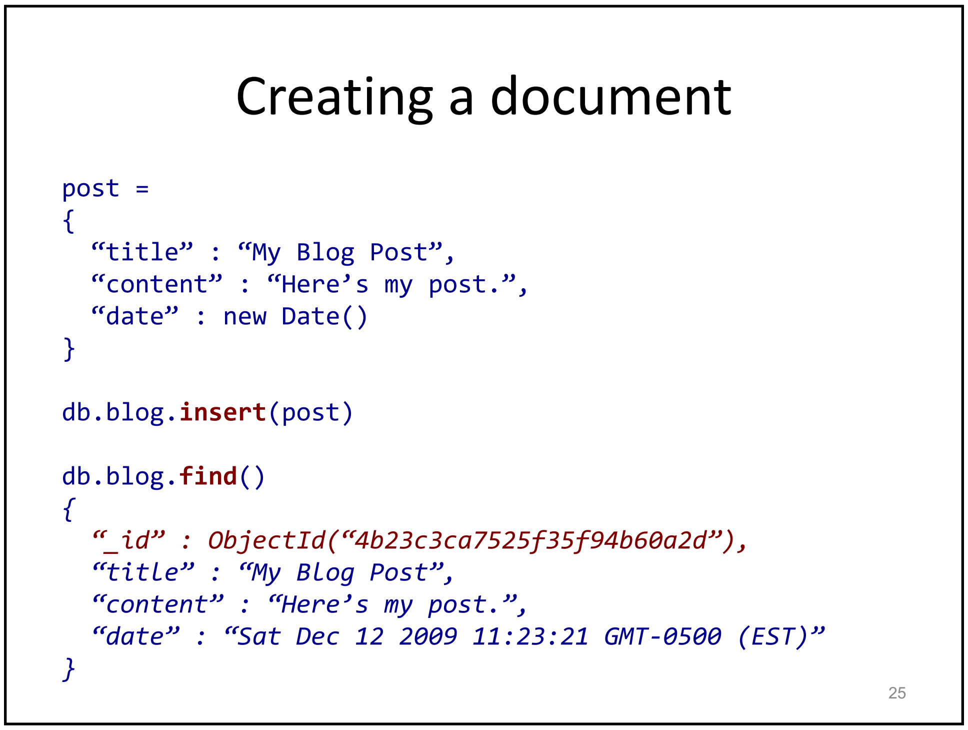
23

Running MongoDB

- Shell: `mongo`
- MongoDB client
- Javascript interpreter
- Default database: `test`
  - Switch: `use foobar`
- Bound variable: db

24

24



Creating a document

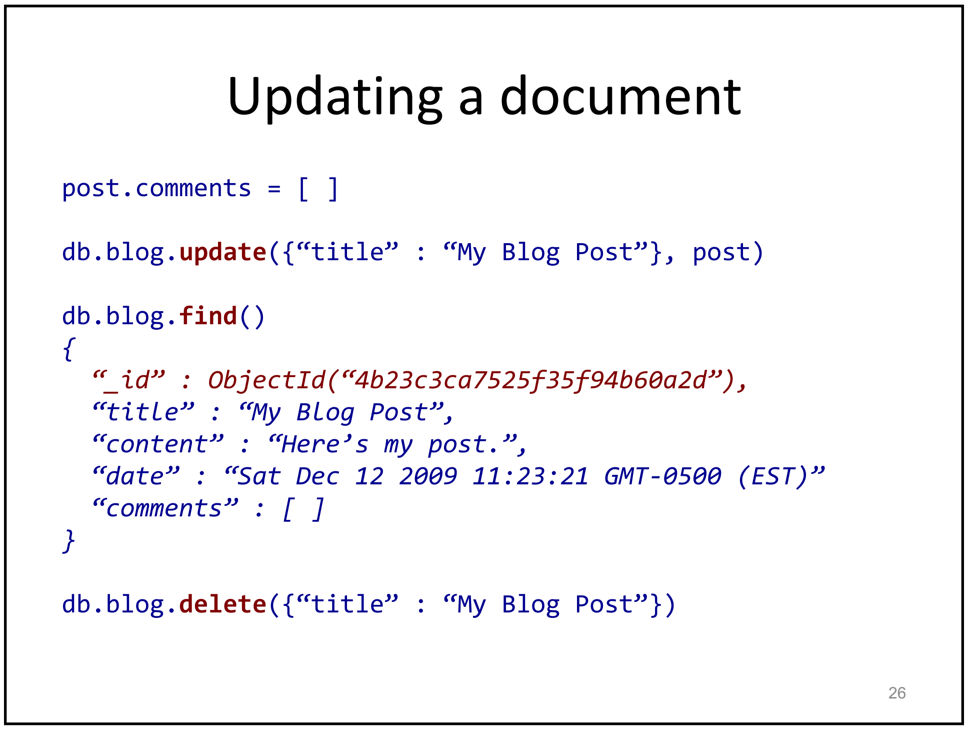
```
post =
{
  “title” : “My Blog Post”,
  “content” : “Here’s my post.”,
  “date” : new Date()
}

db.blog.insert(post)

db.blog.find()
{
  “_id” : ObjectId(“4b23c3ca7525f35f94b60a2d”),
  “title” : “My Blog Post”,
  “content” : “Here’s my post.”,
  “date” : “Sat Dec 12 2009 11:23:21 GMT-0500 (EST)”
}
```

25

25



Updating a document

```
post.comments = [ ]

db.blog.update({“title” : “My Blog Post”}, post)

db.blog.find()
{
  “_id” : ObjectId(“4b23c3ca7525f35f94b60a2d”),
  “title” : “My Blog Post”,
  “content” : “Here’s my post.”,
  “date” : “Sat Dec 12 2009 11:23:21 GMT-0500 (EST)”
  “comments” : [ ]
}

db.blog.delete({“title” : “My Blog Post”})
```

26

26

Shell Commands

```
show dbs
show collections
show users
show profile
use db-name
db.help()
db.foo.help()
db.foo.find()
db.foo.find( {title : "My Blog Post"} )
it
```

27

27

Iterating over Subcollections

```
var collections =
    ["posts", "comments", "authors"];

doStuff(db.blog.posts);
doStuff(db.blog.comments);
doStuff(db.blog.authors);

for (i in collections)
{
   doStuff(db.blog[collections[i]]);
}
```

db.blog["posts"]
Is same as
db.blog.posts

28

28

Data Types

- Null
- Boolean
- 32-bit integer
- 64-bit integer
- 64-bit floating point
- String
- Symbol
- Object id
- Date
- Regular expression
- Code
- Binary data
- Maximum value
- Minimum value
- Undefined
- Array
- Embedded document

29

29

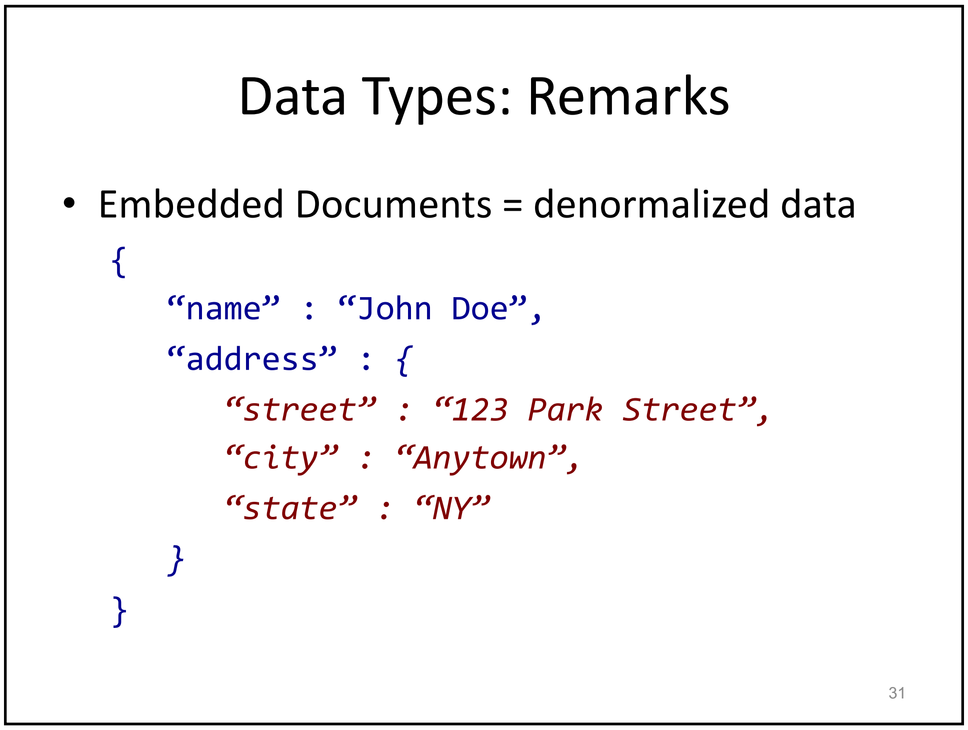
Data Types: Remarks

- Numbers
  - MongoDB: 4-byte int, 8-byte int, 8-byte float
  - Javascript: float
  - 8-byte int: approximate value in shell
- Dates

  `new Date(…)`
- _id and ObjectId
  - `ObjectId = (Timestamp || Machine || PID || Increment)`
  - Auto-generation of _id: client-side

30

30



Data Types: Remarks

- Embedded Documents = denormalized data

```
{
   “name” : “John Doe”,
   “address” : {
      “street” : “123 Park Street”,
      “city” : “Anytown”,
      “state” : “NY”
   }
}
```