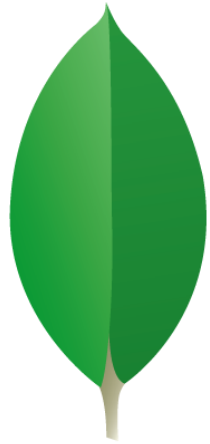


NoSQL and MongoDB



Eng. Hany Saad
SD Dept.
ITI – Assiut Branch



mongoDB

A decorative green geometric pattern consisting of various shades of green rectangles and lines, located in the bottom right corner of the slide.


What's MongoDB?

- MongoDB is a **cross-platform, document oriented** database that provides, high performance, high availability, and easy scalability.
- MongoDB works on concept of collection and document.
- Designed with both scalability and developer agility.
- Built for Speed
- Rich Document based queries for **Easy readability**.
- Full Index Support for **High Performance**.
- Replication and Failover for **High Availability**.
- Auto Sharding for **Easy Scalability**.
 - A **database shard** is a horizontal partition of data in a database or search engine. Each individual partition is referred to as a **shard** or **database shard**. Each shard is held on a separate database server instance, to spread load.
- MapReduce for **Aggregation**.
 - **MapReduce** is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a cluster


What's MongoDB (Cont.)?

- Document-oriented database
 - Uses JSON (BSON actually) •
- Schema-free
- Performance
 - Written in C++
 - Full index support
 - No transactions (has atomic operations)
 - Memory-mapped files (delayed writes)
- Scalability
 - Replication
 - Auto-Sharding
- Commercially supported
 - Lots of documentation
- Supported Platforms:
 - OSX - Linux - Solaris - Windows – FreeBSD.
- MongoDB Drivers:
 - C, C++, Java, Javascript, .NET (C# F#, PowerShell, etc), Node.js, Perl, PHP, Python, Ruby, Scala.

MongoDB is NOT RDBMS

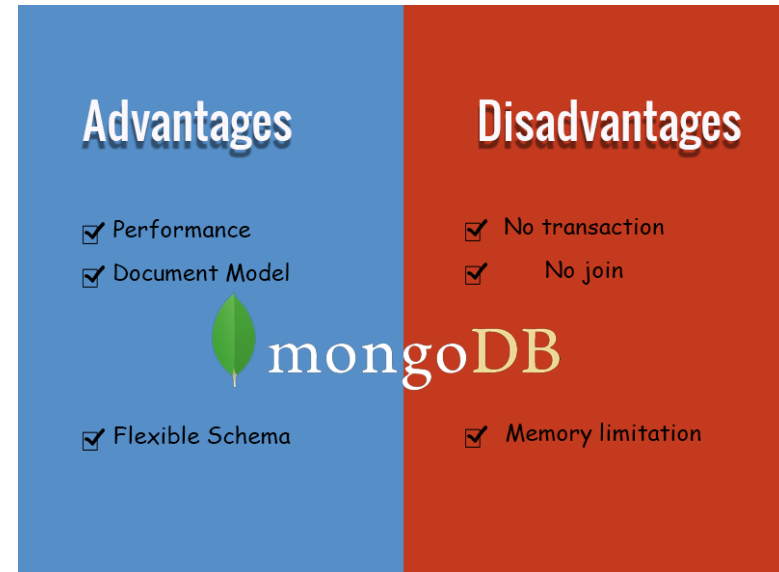
- Mongo is not a relational database like SQL Server
 - No transactions
 - No referential integrity
 - No joins
 - No schema, so no columns or rows (no more ALTER TABLE)
 - NoSQL
- 

Why use MongoDB?


- SQL was invented in the 70's to store data.
 - MongoDB stores documents (or) objects.
 - Now-a-days, everyone works with objects (Python/Ruby/Java/etc.)
 - And we need Databases to persist our objects. Then why not store objects directly ?
 - Embedded documents and arrays reduce need for joins. No Joins and No-multi document transactions.
- 

Great for / Not great for ...

- **MongoDB is great for:**
 - RDBMS replacement for Web Applications.
 - Semi-structured Content Management.
 - Real-time Analytics & High-Speed Logging.
 - Caching and High Scalability
 - Web 2.0, Media, SAAS, Gaming
 - HealthCare, Finance, Telecom, Government
- **Isn't great for:**
 - Highly Transactional Applications.
 - Problems requiring SQL




How it works?

- Clients connect to the MongoDB process, optionally authenticate themselves if security is turned on, and perform a sequence of actions, such as inserts, queries and updates.
 - MongoDB stores its data in files (default location is /data/db/), and uses memory mapped files for data management for efficiency.
- 

How it works (Cont.)?

| RDBMS | MongoDB |
|-------------|--|
| Database | Database |
| Table | Collection |
| Tuple/Row | Document |
| column | Field |
| Table Join | Embedded Documents |
| Primary Key | Primary Key (Default key _id provided by mongodb itself) |

How it works (Cont.)?

- **Database** is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.
 - **Collection** is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.
 - **Document** is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.
- 

How it works (Cont.)?

When we say
Database



Think
Database

- Made up of Multiple **Collections**.
- Created **on-the-fly** when referenced for the first time.

How it works (Cont.)?

When we say
Collection



Think
Table

- Schema-less, and contains **Documents**.
- **Indexable** by one/more keys.
- Created **on-the-fly** when referenced for the first time.

How it works (Cont.)?


When we say
Document



Think
Record/Row

- Stored in a **Collection**.
- Can have **_id** key – works like Primary keys in SQL Server.
- Document storage in **BSON** (Binary form of JSON).
- By document, we mean that we store data that is a structured collection of keyvalue pairs, where keys are strings, and values are any of a rich set of data types, including arrays and documents.

BSON

- It is a computer data interchange format used mainly as a data storage and network transfer format in the MongoDB database.
 - It is a **binary form** for representing simple data structures .
 - BSON is **faster** to scan for specific fields than JSON.
 - BSON adds some additional types such as a date data type and a byte-array (bindata) datatype.
 - Client drivers serialize data to BSON, then transmit the data over the wire to the db.
 - Data is stored on disk in BSON format. Thus, on a retrieval, the database does very little translation to send an object out, allowing high efficiency.
 - The client driver unserialized a received BSON object to its native language format.
- 

BSON (Cont.)

```
var post = {  
  '_id': '3432',  
  'author': DBRef('User', 2),  
  'title': 'Introduction to MongoDB',  
  'body': 'MongoDB is an open sources.. ',  
  'timestamp': Date('01-04-12'),  
  'tags': ['MongoDB', 'NoSQL'],  
  'comments': [{ 'author': DBRef('User', 4),  
                  'date': Date('02-04-12'),  
                  'text': 'Did you see.. ',  
                  'upvotes': 7, ... }  
}
```

BSON(Cont.)

A document such as {"hello":"world"} will be stored as:

```
Bson:
\x16\x00\x00\x00      // total document size
\x02                   // 0x02 = type String
hello\x00              // field name
\x06\x00\x00\x00world\x00 // field value (size of value, value, null terminator)
\x00                   // 0x00 = type E00 ('end of object')
```

- What are the goals of BSON?
 - Fast scan-ability
 - Easy manipulation
 - Additional data types

Sample Document

USERS

| <i>ID</i> | <i>First</i> | <i>Last</i> |
|-----------|--------------|-------------|
| 1 | Shane | Johnson |

USER SKILLS

| <i>User ID</i> | <i>Skill Name</i> |
|----------------|-------------------|
| 1 | Big Data |
| 1 | Java |
| 1 | NoSQL |

USER EXPERIENCE

| <i>User ID</i> | <i>Role</i> | <i>Company</i> |
|----------------|----------------|----------------|
| 1 | Technical Mktg | Red Hat |
| 1 | Product Mktg | Couchbase |

RELATIONAL DATABASE

```
{
  "firstName": "Shane",
  "lastName": "Johnson",
  "skills": ["Big Data", "Java", "NoSQL"],
  "experience": [
    {
      "role": "Technical Marketing",
      "company": "Red Hat"
    },
    {
      "role": "Product Marketing",
      "company": "Couchbase"
    }
  ]
}
```

JSON DOCUMENT

Sample Document - 2

```
{
  _id: ObjectId(7df78ad8902c)
  title: 'MongoDB Overview',
  description: 'MongoDB is no sql database',
  by: 'tutorials point',
  url: 'http://www.tutorialspoint.com',
  tags: ['mongodb', 'database', 'NoSQL'],
  likes: 100,
  comments: [
    {
      user: 'user1',
      message: 'My first comment',
      dateCreated: new Date(2011,1,20,2,15),
      like: 0
    },
    {
      user: 'user2',
      message: 'My second comments',
      dateCreated: new Date(2011,1,25,7,45),
      like: 5
    }
  ]
}
```

- **_id** is a 12 bytes hexadecimal number which assures the uniqueness of every document.
- You can provide **_id** while inserting the document, If you didn't provide then MongoDB provide a unique id for every document.
- These 12 bytes first 4 bytes for the current timestamp, next 3 bytes for machine id, next 2 bytes for process id of mongodb server and remaining 3 bytes are simple incremental value.

Getting started...

- **Download MongoDB.exe version suitable for your device from here:**
<https://www.mongodb.com/download-center/community>
- **Create file to hold the data:**
 - Default: C:\data
 - Custom: any path, and specify the path when connecting to the DB
- **Run your mongo server**
 - Navigate to your extracted file > bin >
 - Run: `mongod.exe -dbpath`
 - `mongod` stands for “Mongo Daemon”. `mongod` is a background process used by MongoDB. The main purpose of `mongod` is to manage all the MongoDB server tasks.
- **Run your mongo shell**
 - Run: `mongo.exe` to run queries.
 - Mongo shell is a command line shell that can interact with the client (for example, system administrators and developers).

DEMO !


A green geometric pattern consisting of various shades of green rectangles and lines, located in the bottom right corner of the slide.

References

- **Textbook references:**

- MongoDB: the definitive guide
- The Little MongoDB

- **Online Tutorials:**

- <https://code.tutsplus.com/tutorials/getting-started-with-mongodb-part-1--net-22879>
 - <https://www.tutorialspoint.com/mongodb/>
 - <https://docs.mongodb.com/manual/tutorial/>
- 

A decorative horizontal band at the top of the slide featuring a complex, overlapping pattern of green and yellow geometric shapes, creating a sense of depth and movement.

Thanks...

A decorative horizontal band at the bottom of the slide, similar to the one at the top, featuring a complex, overlapping pattern of green and yellow geometric shapes.

Eng. Hany Saad
SD Dept.
ITI – Assiut Branch