PES UNIVERSITY ONLINE

MongoDB

Vinay Joshi
Department of
Computer Science and Engineering

MongoDB
File Based Systems

- Program defines and manages it’s own data

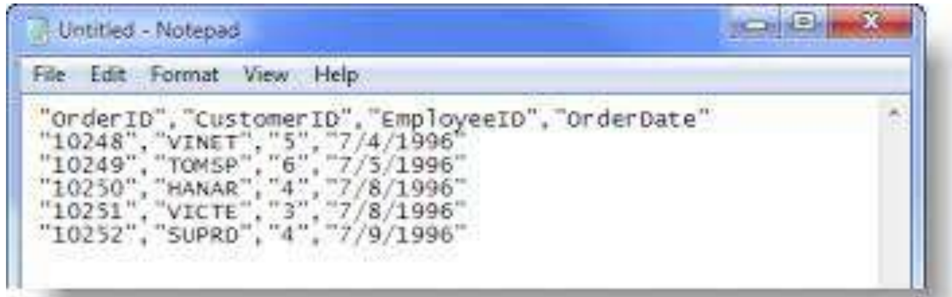
Limitations:

  - Separation and isolation
  - Duplication
  - Program & data dependence
  - Fixed queries
  - Proliferation of application programs

MongoDB

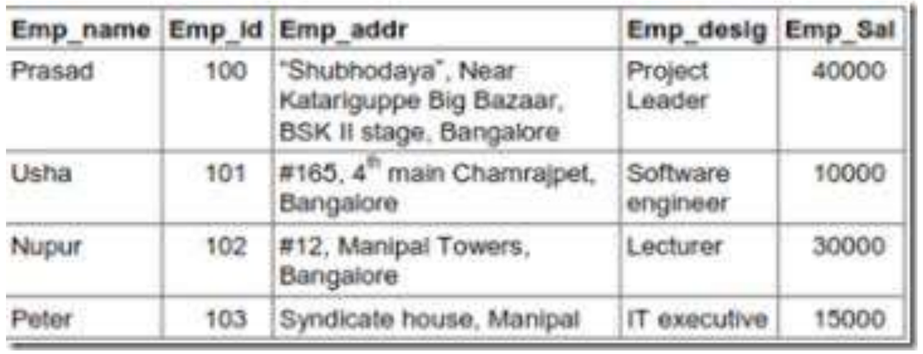
File Based Systems

File (Typically a CSV file)



```
"OrderID","CustomerID","EmployeeID","OrderDate"
"10248","VINET","5","7/4/1996"
"10249","TOMSP","6","7/5/1996"
"10250","HANAR","4","7/8/1996"
"10251","VICTE","3","7/8/1996"
"10252","SUPRD","4","7/9/1996"
```

Database



Emp_name	Emp_id	Emp_addr	Emp_desig	Emp_Sal
Prasad	100	"Shubhodaya", Near Katariguppe Big Bazaar, BSK II stage, Bangalore	Project Leader	40000
Usha	101	#165, 4th main Chamrajpet, Bangalore	Software engineer	10000
Nupur	102	#12, Manipal Towers, Bangalore	Lecturer	30000
Peter	103	Syndicate house, Manipal	IT executive	15000

MongoDB
Introduction

- Name comes from “Humongous” & huge data
- Written in C++, developed in 2009
- Creator: 10gen, former doublick

- Definition: MongoDB is an open source, document-oriented database designed with both scalability and developer agility in mind
- Instead of storing your data in tables and rows as you would with a relational database, in MongoDB you store JSON-like documents with dynamic schemas (schema-free, schemaless)

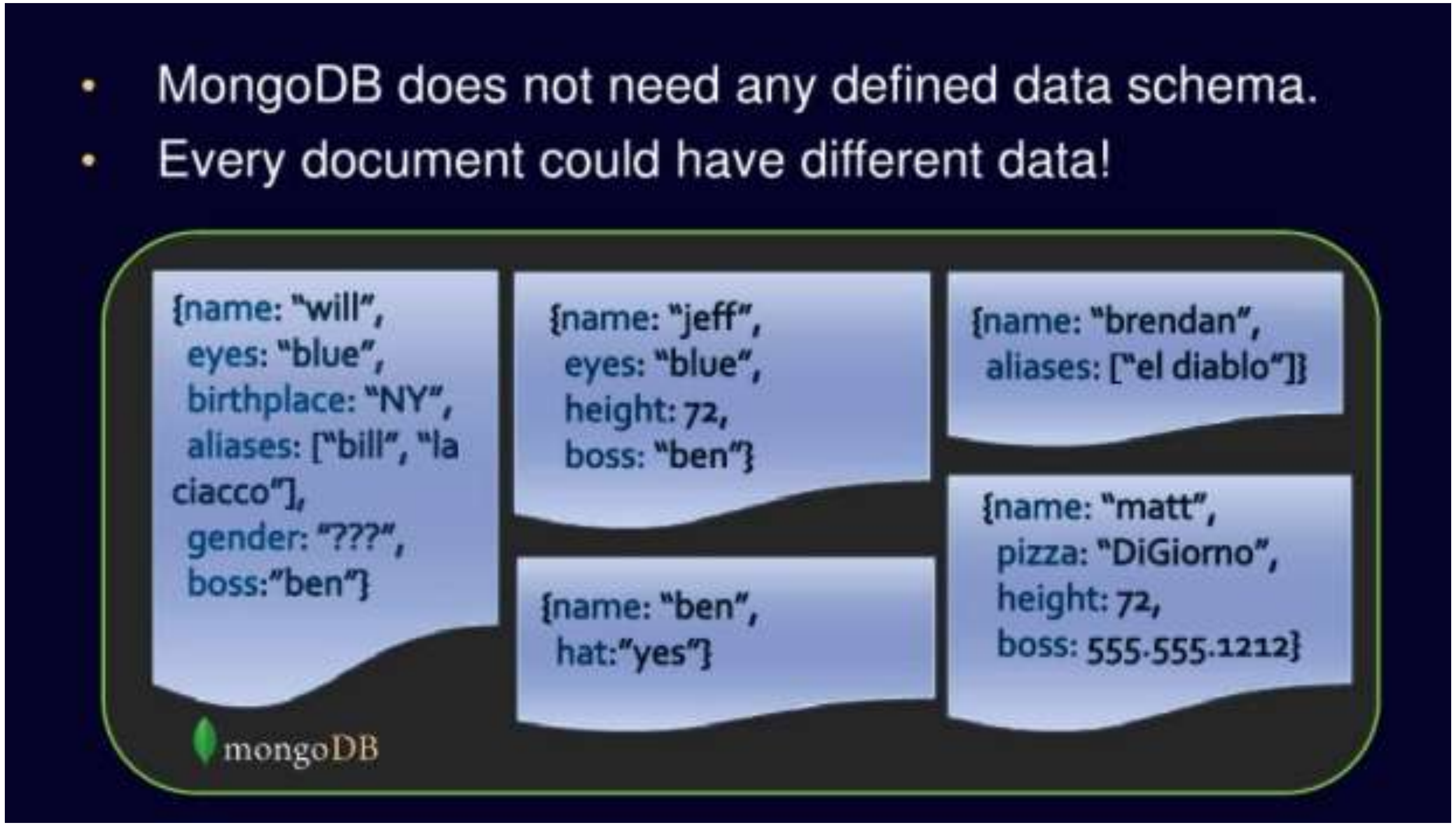
MongoDB
NoSQL Database

- Stands for **N**ot **O**nly **SQL??**
- Class of non-relational data storage systems
- Usually do not require a fixed table schema nor do they use the concept of joins to derive data from different tables

MongoDB

MongoDB – a NoSQL Database

- No Defined Schema (Schema Free or Schema Less)



- MongoDB does not need any defined data schema.
- Every document could have different data!

```
{name: "will",
 eyes: "blue",
 birthplace: "NY",
 aliases: ["bill", "la
ciacco"],
 gender: "???",
 boss:"ben"}
```

```
{name: "jeff",
 eyes: "blue",
 height: 72,
 boss: "ben"}
```

```
{name: "brendan",
 aliases: ["el diablo"]}
```

```
{name: "ben",
 hat:"yes"}
```

```
{name: "matt",
 pizza: "DiGiorno",
 height: 72,
 boss: 555.555.1212}
```

MongoDB

Terms Mapping (DB vs. MongoDB)

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

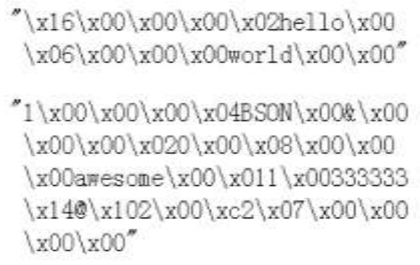
MongoDB

Data Model – BSON Format

- BSON format (binary JSON)
- Developers can easily map to modern object-oriented languages without a complicated ORM layer.

```
{ author: 'joe',
  created : new Date('03/28/2009'),
  title : 'Yet another blog post',
  text : 'Here is the text...',
  tags : [ 'example', 'joe' ],
  comments : [
              { author: 'jim',
                comment: 'I disagree'
              },
              { author: 'nancy',
                comment: 'Good post'
              }
  ]
}
```

Remember it is stored in binary formats

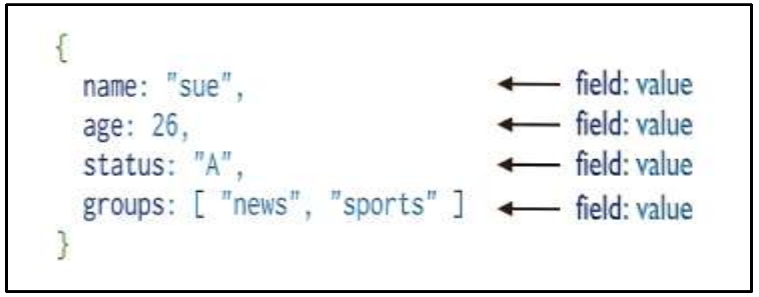


```
"\x16\x00\x00\x00\x02hello\x00
 \x06\x00\x00\x00world\x00\x00"

"1\x00\x00\x00\x04BSON\x00&\x00
 \x00\x00\x020\x00\x08\x00\x00
 \x00awesome\x00\x011\x00333333
 \x14@\x102\x00\xc2\x07\x00\x00
 \x00\x00"
```

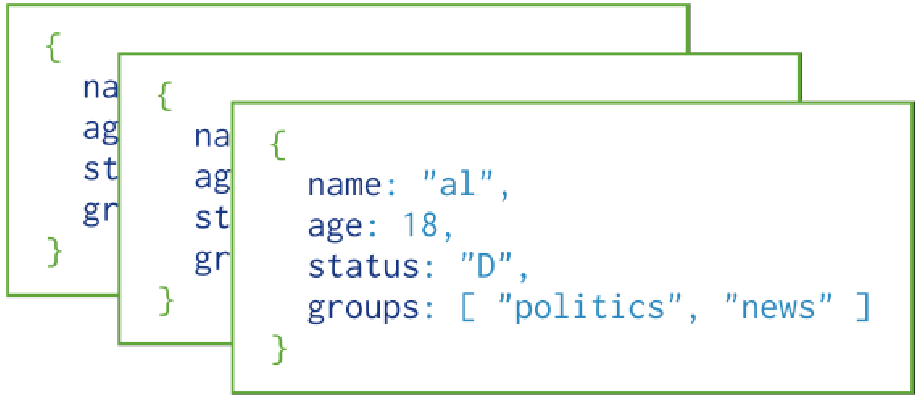
MongoDB
MongoDB Data Model

One *document* (e.g., one tuple in RDBMS)



```
{
   name: "sue",                          ← field: value
   age: 26,                              ← field: value
   status: "A",                          ← field: value
   groups: [ "news", "sports" ]          ← field: value
}
```

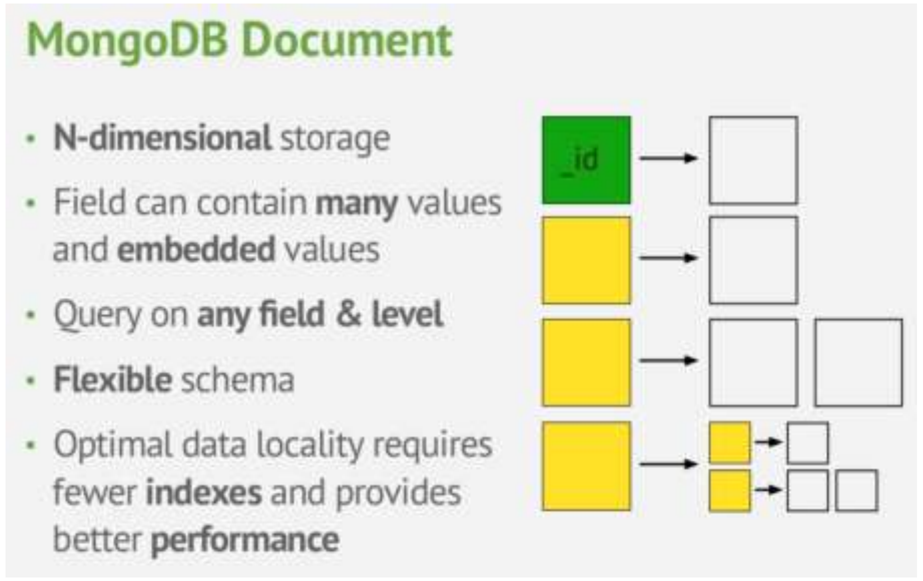
One *Collection* (e.g., one Table in RDBMS)



```
{
   name: "al",
   age: 18,
   status: "D",
   groups: [ "politics", "news" ]
}
```

Collection

- **Collection** is a group of similar documents
- Within a collection, each document must have a unique Id

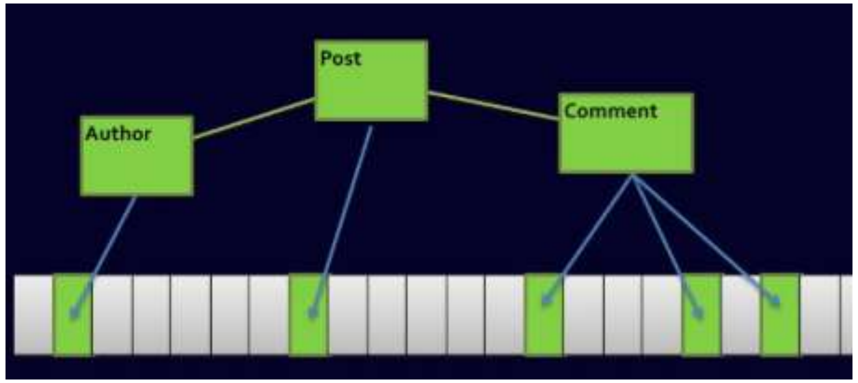


MongoDB Document

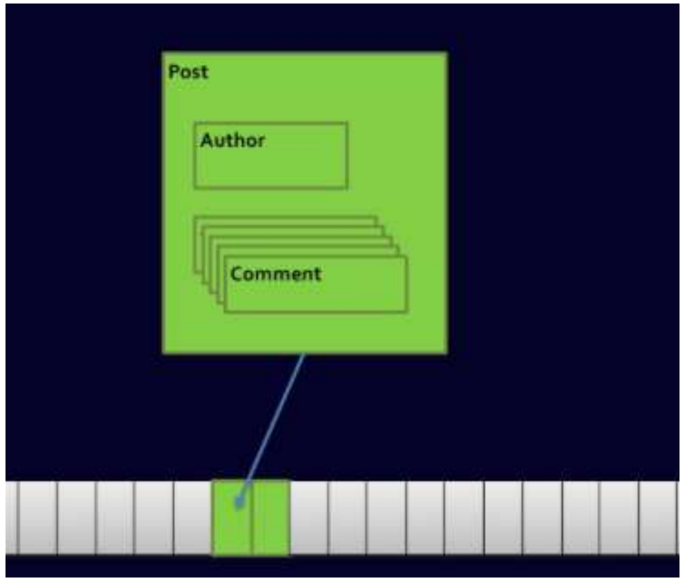
- **N-dimensional** storage
- Field can contain **many** values and **embedded** values
- Query on **any field & level**
- **Flexible** schema
- Optimal data locality requires fewer **indexes** and provides better **performance**

MongoDB
Complex Join Queries

Relational DBs



MongoDB



MongoDB

Getting Started…

Install it

Practice simple stuff

Move to complex stuff

Install it from here: http://www.mongodb.org

Manual: http://docs.mongodb.org/master/MongoDB-manual.pdf
(Focus on Ch. 3, 4 for now)

Dataset: http://docs.mongodb.org/manual/reference/bios-example-collection/

MongoDB
Data Operations

Create

    db.collection.insert(<document>)
    db.collection.save(<document>)
    db.collection.update(<query>, <update>, { upsert: true })

Read

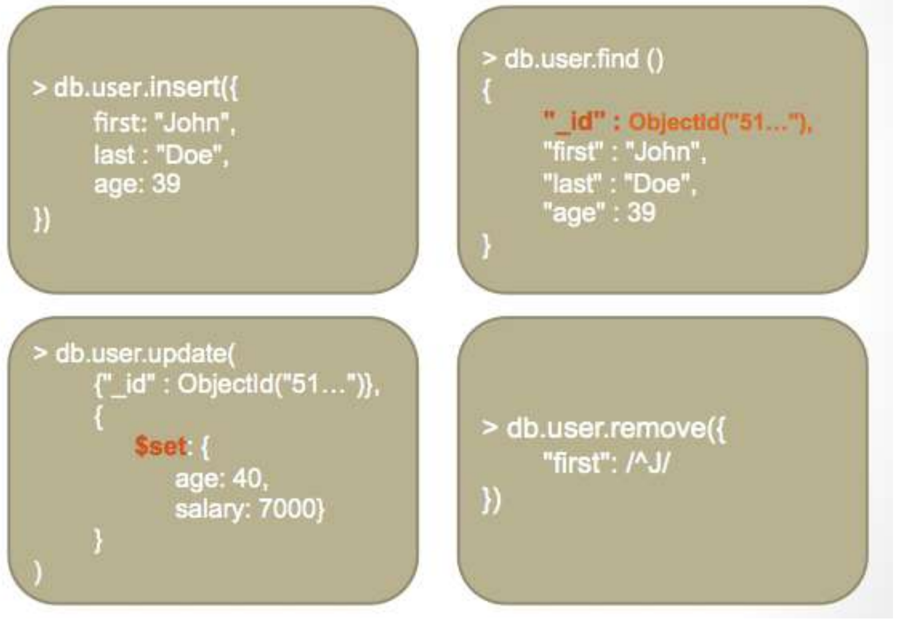
    db.collection.find(<query>, <projection>)
    db.collection.findOne(<query>, <projection>)

Update

    db.collection.update(<query>, <update>, <options>)

Delete

    db.collection.remove(<query>, <justOne>)



```
> db.user.insert({
        first: "John",
        last : "Doe",
        age: 39
})
```

```
> db.user.find ()
{
        "_id" : ObjectId("51..."),
        "first" : "John",
        "last" : "Doe",
        "age" : 39
}
```

```
> db.user.update(
        {"_id" : ObjectId("51...")},
        {
            $set: {
                age: 40,
                salary: 7000}
        }
)
```

```
> db.user.remove({
        "first": /^J/
})
```

MongoDB
Example Operations – Creation and Deletion

In RDBMS

```
CREATE TABLE users (
    id MEDIUMINT NOT NULL
        AUTO_INCREMENT,
    user_id Varchar(30),
    age Number,
    status char(1),
    PRIMARY KEY (id)
)
```

In MongoDB

Either insert the 1st docuement

```
db.users.insert( {
    user_id: "abc123",
    age: 55,
    status: "A"
 } )
```

Or create “Users” collection explicitly

```
db.createCollection("users")
```

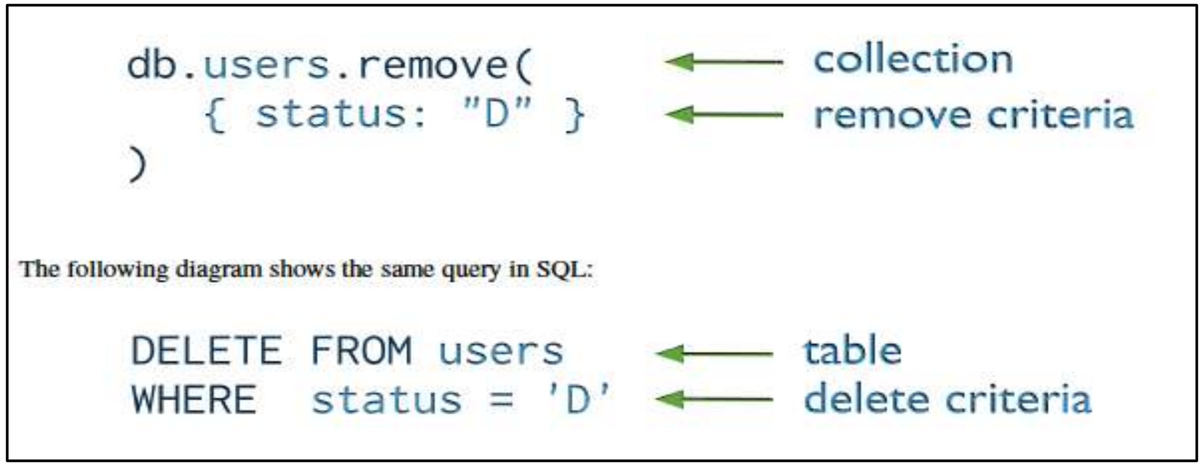
```
DROP TABLE users
```

```
db.users.drop()
```

MongoDB

Example Operations – Removal or Deletion

- You can put condition on any field in the document (even ***_id***)



```
db.users.remove(          ←—— collection
   { status: "D" }        ←—— remove criteria
)
```

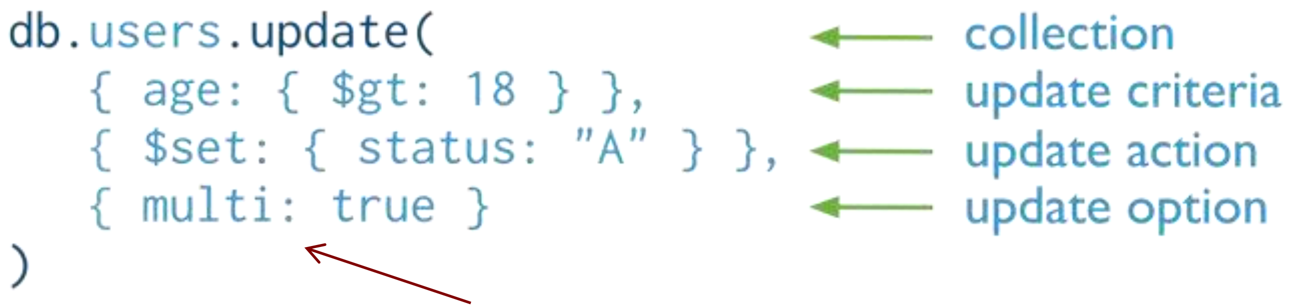
The following diagram shows the same query in SQL:

```
DELETE FROM users         ←—— table
WHERE  status = 'D'       ←—— delete criteria
```

db.users.remove ( ) ➡ Removes all documents from *users* collection

MongoDB

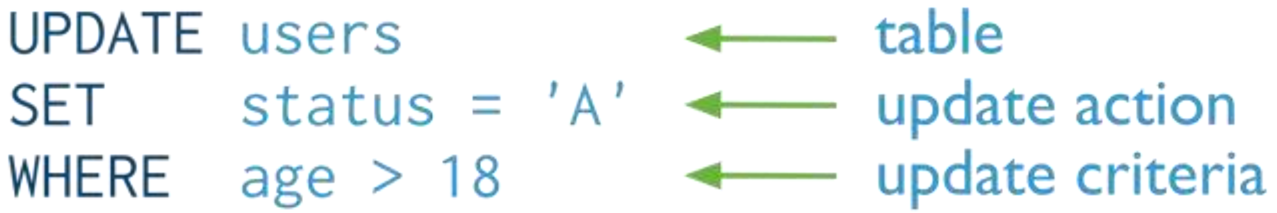
Example Operations – Update



```
db.users.update(
   { age: { $gt: 18 } },
   { $set: { status: "A" } },
   { multi: true }
)
```

Otherwise, it will update only the 1st matching document

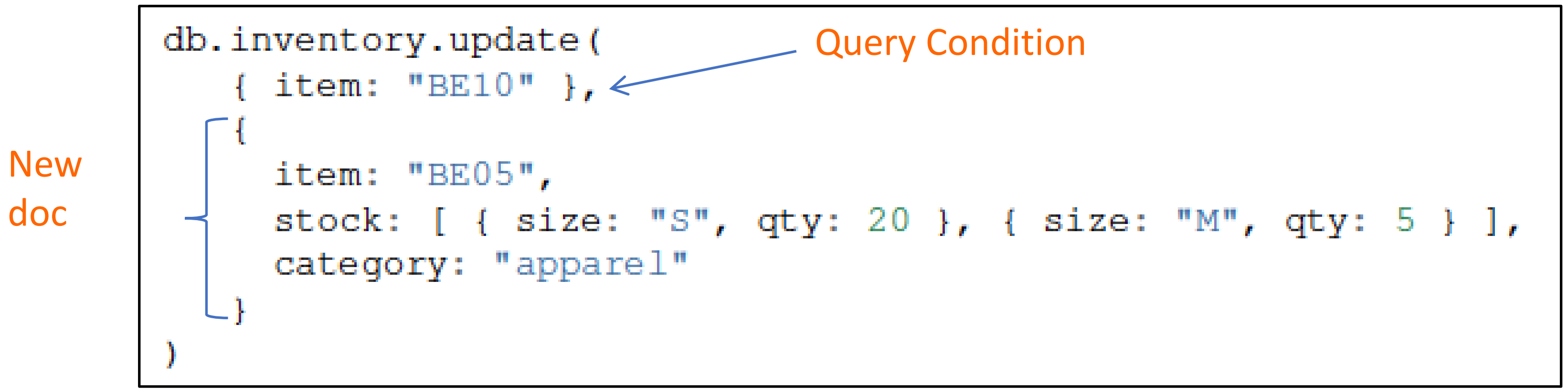
Equivalent to in SQL:



```
UPDATE users
SET    status = 'A'
WHERE  age > 18
```

MongoDB

Example Operations – Replace a document



```
db.inventory.update(
   { item: "BE10" },
   {
     item: "BE05",
     stock: [ { size: "S", qty: 20 }, { size: "M", qty: 5 } ],
     category: "apparel"
   }
)
```

For the document having item = “BE10”, replace it with the given document

MongoDB

Example Operations – Insert or Update?

```
db.inventory.update(
   { item: "TBD1" },
   {
     item: "TBD1",
     details: { "model" : "14Q4", "manufacturer" : "ABC Company" },
     stock: [ { "size" : "S", "qty" : 25 } ],
     category: "houseware"
   },
   { upsert: true }
)
```

The ***upsert*** option

If the document having item = “TBD1” is in the DB, it will be replaced
Otherwise, it will be inserted.

THANK YOU

Vinay Joshi
Department of
Computer Science and Engineering
vinayj@pes.edu