---

title: "Quiz\_MongoDB"

author: "Prashant B. Bhuyan"

date: "October 28, 2014"

output: pdf\_document

---

Question 1:

# Start Mongo DB and Connect

Microstructure-Research-MacBook-Pro:bin MicrostrRes$ cd /usr/local/Cellar/mongodb/2.6.5

Microstructure-Research-MacBook-Pro:2.6.5 MicrostrRes$ cd bin/

Microstructure-Research-MacBook-Pro:bin MicrostrRes$ mkdir bin/db

mkdir: bin: No such file or directory

Microstructure-Research-MacBook-Pro:bin MicrostrRes$ mkdir /usr/local/Cellar/mongodb/2.6.5/bin/db

mkdir: /usr/local/Cellar/mongodb/2.6.5/bin/db: File exists

Microstructure-Research-MacBook-Pro:bin MicrostrRes$ cd . .

Microstructure-Research-MacBook-Pro:bin MicrostrRes$ mongod --dbpath /usr/local/Cellar/mongodb/2.6.5/bin/db

# Output

2014-10-28T19:05:12.799-0400 [initandlisten] MongoDB starting : pid=2063 port=27017 dbpath=/usr/local/Cellar/mongodb/2.6.5/bin/db 64-bit host=Microstructure-Research-MacBook-Pro.local

2014-10-28T19:05:12.799-0400 [initandlisten]

2014-10-28T19:05:12.799-0400 [initandlisten] \*\* WARNING: soft rlimits too low. Number of files is 256, should be at least 1000

2014-10-28T19:05:12.799-0400 [initandlisten] db version v2.6.5

2014-10-28T19:05:12.799-0400 [initandlisten] git version: nogitversion

2014-10-28T19:05:12.799-0400 [initandlisten] build info: Darwin minimavericks.local 13.4.0 Darwin Kernel Version 13.4.0: Sun Aug 17 19:50:11 PDT 2014; root:xnu-2422.115.4~1/RELEASE\_X86\_64 x86\_64 BOOST\_LIB\_VERSION=1\_49

2014-10-28T19:05:12.799-0400 [initandlisten] allocator: tcmalloc

2014-10-28T19:05:12.799-0400 [initandlisten] options: { storage: { dbPath: "/usr/local/Cellar/mongodb/2.6.5/bin/db" } }

2014-10-28T19:05:12.808-0400 [initandlisten] journal dir=/usr/local/Cellar/mongodb/2.6.5/bin/db/journal

2014-10-28T19:05:12.808-0400 [initandlisten] recover : no journal files present, no recovery needed

2014-10-28T19:05:12.874-0400 [initandlisten] waiting for connections on port 27017

2014-10-28T19:06:12.882-0400 [clientcursormon] mem (MB) res:49 virt:2796

2014-10-28T19:06:12.882-0400 [clientcursormon] mapped (incl journal view):320

2014-10-28T19:06:12.882-0400 [clientcursormon] connections:0

2014-10-28T19:06:32.512-0400 [initandlisten] connection accepted from 127.0.0.1:51501 #1 (1 connection now open)

# Create Employment Database

> use employment

> show collections

> db.employment.count()

# Output

switched to db employment

> show collections

> 0

Question 2:

# Create employees collection

> use employment

switched to db employment

> db.createCollection("employees")

{ "ok" : 1 }

> show collections

employees

employment

system.indexes

# Add record for Wendy into an employee collection- \*\*\*Collection is implicitly created this way as well\*\*\*.

> db.employees.insert({Name: "Wendy Yasquez", Title: "Assistant Professor", Salary: "$86,000", Department: "Computer Science", Hire\_Year: "1998"})

> db.employees.find({Name: "Wendy Yasquez"}).pretty()

{

"\_id" : ObjectId("54502fdcbb19e246e561cb23"),

"Name" : "Wendy Yasquez",

"Title" : "Assistant Professor",

"Salary" : "$86,000",

"Department" : "Computer Science",

"Hire\_Year" : "1998"

}

Question 3:

# Function to insert new preforessors into employee collection

> function insert\_newProf(Name, Title, Salary, Department, Hire\_Year){ db.employees.insert({Name:Name, Title: Title, Salary: Salary, Department: Department, Hire\_Year: Hire\_Year})}

# Output:

> insert\_newProf("Wendy Yasquez", "Assistant Professor", 86000, "Computer Science", 1998)

> db.employees.find().pretty()

{

"\_id" : ObjectId("54510528bb19e246e561cb38"),

"Name" : "Wendy Yasquez",

"Title" : "Assistant Professor",

"Salary" : 86000,

"Department" : "Computer Science",

"Hire\_Year" : 1998

}

Question 4:

# Inserting other professors into employees collection

> insert\_newProf("Raoul Dewan", "Assistant Professor", 78000, "Physics, Biology", 2009)

> insert\_newProf("Isabelle Winters", "Assistant Professor", 92000, "Physics", 1995)

> insert\_newProf("Jack McDunn", "Assistant Professor", 101000, "Physics", 1993)

# Output

> db.employees.find().pretty()

{

"\_id" : ObjectId("54510528bb19e246e561cb38"),

"Name" : "Wendy Yasquez",

"Title" : "Assistant Professor",

"Salary" : 86000,

"Department" : "Computer Science",

"Hire\_Year" : 1998

}

> insert\_newProf("Raoul Dewan", "Assistant Professor", 78000, "Physics, Biology", 2009)

> insert\_newProf("Isabelle Winters", "Assistant Professor", 92000, "Physics", 1995)

> insert\_newProf("Jack McDunn", "Assistant Professor", 101000, "Physics", 1993)

> db.employees.find().pretty()

{

"\_id" : ObjectId("54510528bb19e246e561cb38"),

"Name" : "Wendy Yasquez",

"Title" : "Assistant Professor",

"Salary" : 86000,

"Department" : "Computer Science",

"Hire\_Year" : 1998

}

{

"\_id" : ObjectId("5451057cbb19e246e561cb39"),

"Name" : "Raoul Dewan",

"Title" : "Assistant Professor",

"Salary" : 78000,

"Department" : "Physics, Biology",

"Hire\_Year" : 2009

}

{

"\_id" : ObjectId("54510598bb19e246e561cb3a"),

"Name" : "Isabelle Winters",

"Title" : "Assistant Professor",

"Salary" : 92000,

"Department" : "Physics",

"Hire\_Year" : 1995

}

{

"\_id" : ObjectId("545105acbb19e246e561cb3b"),

"Name" : "Jack McDunn",

"Title" : "Assistant Professor",

"Salary" : 101000,

"Department" : "Physics",

"Hire\_Year" : 1993

}

Question 5 and Question 6:

# Insert administrative employees into the employees collection using the insert\_newProf() function from above

> function insert\_AdminStaff(){

... insert\_newProf("Tonja Balndr", "Assistant to the Dean", 42000, "Arts and Sciences", 2001)

... insert\_newProf("Dennis Bohnet", "Vice President", 106500, "Academic Affairs", 1997)

... }

# Output:

> insert\_AdminStaff()

> db.employees.find().pretty()

{

"\_id" : ObjectId("54510528bb19e246e561cb38"),

"Name" : "Wendy Yasquez",

"Title" : "Assistant Professor",

"Salary" : 86000,

"Department" : "Computer Science",

"Hire\_Year" : 1998

}

{

"\_id" : ObjectId("5451057cbb19e246e561cb39"),

"Name" : "Raoul Dewan",

"Title" : "Assistant Professor",

"Salary" : 78000,

"Department" : "Physics, Biology",

"Hire\_Year" : 2009

}

{

"\_id" : ObjectId("54510598bb19e246e561cb3a"),

"Name" : "Isabelle Winters",

"Title" : "Assistant Professor",

"Salary" : 92000,

"Department" : "Physics",

"Hire\_Year" : 1995

}

{

"\_id" : ObjectId("545105acbb19e246e561cb3b"),

"Name" : "Jack McDunn",

"Title" : "Assistant Professor",

"Salary" : 101000,

"Department" : "Physics",

"Hire\_Year" : 1993

}

{

"\_id" : ObjectId("54511438bb19e246e561cb3c"),

"Name" : "Tonja Balndr",

"Title" : "Assistant to the Dean",

"Salary" : 42000,

"Department" : "Arts and Sciences",

"Hire\_Year" : 2001

}

{

"\_id" : ObjectId("54511438bb19e246e561cb3d"),

"Name" : "Dennis Bohnet",

"Title" : "Vice President",

"Salary" : 106500,

"Department" : "Academic Affairs",

"Hire\_Year" : 1997

}

Problem 6:

Please see Question 5 Above that contains the solution to Problem 6.

Problem 7:

# Find all employees with salaries less than $90,000.

db.employees.find('this.Salary < 90000').pretty()

# Output

{

"\_id" : ObjectId("54510528bb19e246e561cb38"),

"Name" : "Wendy Yasquez",

"Title" : "Assistant Professor",

"Salary" : 86000,

"Department" : "Computer Science",

"Hire\_Year" : 1998

}

{

"\_id" : ObjectId("5451057cbb19e246e561cb39"),

"Name" : "Raoul Dewan",

"Title" : "Assistant Professor",

"Salary" : 78000,

"Department" : "Physics, Biology",

"Hire\_Year" : 2009

}

{

"\_id" : ObjectId("54511438bb19e246e561cb3c"),

"Name" : "Tonja Balndr",

"Title" : "Assistant to the Dean",

"Salary" : 42000,

"Department" : "Arts and Sciences",

"Hire\_Year" : 2001

}

Problem 8:

# Find professors with salaries less than $90,000.

> db.employees.find('this.Salary < 90000 && this.Title == "Assistant Professor"').pretty()

# Output:

> db.employees.find('this.Salary < 90000 && this.Title == "Assistant Professor"').pretty()

{

"\_id" : ObjectId("54510528bb19e246e561cb38"),

"Name" : "Wendy Yasquez",

"Title" : "Assistant Professor",

"Salary" : 86000,

"Department" : "Computer Science",

"Hire\_Year" : 1998

}

{

"\_id" : ObjectId("5451057cbb19e246e561cb39"),

"Name" : "Raoul Dewan",

"Title" : "Assistant Professor",

"Salary" : 78000,

"Department" : "Physics, Biology",

"Hire\_Year" : 2009

}

Problem 9:

# find physics professors hired before 2001

> db.employees.find('this.Hire\_Year < 2001 && this.Department == "Physics"').pretty()

# Output:

{

"\_id" : ObjectId("54510598bb19e246e561cb3a"),

"Name" : "Isabelle Winters",

"Title" : "Assistant Professor",

"Salary" : 92000,

"Department" : "Physics",

"Hire\_Year" : 1995

}

{

"\_id" : ObjectId("545105acbb19e246e561cb3b"),

"Name" : "Jack McDunn",

"Title" : "Assistant Professor",

"Salary" : 101000,

"Department" : "Physics",

"Hire\_Year" : 1993

}

Problem 10:

# Return staff members that are not associated exclusively with the physics department.

# Find people who may be associated with Physics and some other Dept and copy

# that information to a variable called re.

> var re = db.employees.find({Department:{$regex:/Physics,/i}}).pretty()

# Find all people that are not at all associated with the Physics department and copy #that informatino to a variable called re2

> var re2 = db.employees.find({Department:{$not:/Physics/i}}).pretty()

# Insert the contents of re into a new collection called re3.

> while(re.hasNext()){

... db.re3.insert(re.next())

... }

# Insert the contents of re2 into re3 as well.

> while(re2.hasNext()){

... db.re3.insert(re2.next())

... }

# Output

> db.re3.find().pretty()

{

"\_id" : ObjectId("5451057cbb19e246e561cb39"),

"Name" : "Raoul Dewan",

"Title" : "Assistant Professor",

"Salary" : 78000,

"Department" : "Physics, Biology",

"Hire\_Year" : 2009

}

{

"\_id" : ObjectId("54510528bb19e246e561cb38"),

"Name" : "Wendy Yasquez",

"Title" : "Assistant Professor",

"Salary" : 86000,

"Department" : "Computer Science",

"Hire\_Year" : 1998

}

{

"\_id" : ObjectId("54511438bb19e246e561cb3c"),

"Name" : "Tonja Balndr",

"Title" : "Assistant to the Dean",

"Salary" : 42000,

"Department" : "Arts and Sciences",

"Hire\_Year" : 2001

}

{

"\_id" : ObjectId("54511438bb19e246e561cb3d"),

"Name" : "Dennis Bohnet",

"Title" : "Vice President",

"Salary" : 106500,

"Department" : "Academic Affairs",

"Hire\_Year" : 1997

}

Problem 11:

# Find employees with salaries > 100k or hired before 1997

> db.employees.find({$or: [ {Hire\_Year: {$lt: 1997}},{Salary:{$gt:100000}}]}).pretty()

# Output:

> db.employees.find({$or: [ {Hire\_Year: {$lt: 1997}},{Salary:{$gt:100000}}]}).pretty()

{

"\_id" : ObjectId("54510598bb19e246e561cb3a"),

"Name" : "Isabelle Winters",

"Title" : "Assistant Professor",

"Salary" : 92000,

"Department" : "Physics",

"Hire\_Year" : 1995

}

{

"\_id" : ObjectId("545105acbb19e246e561cb3b"),

"Name" : "Jack McDunn",

"Title" : "Assistant Professor",

"Salary" : 101000,

"Department" : "Physics",

"Hire\_Year" : 1993

}

{

"\_id" : ObjectId("54511438bb19e246e561cb3d"),

"Name" : "Dennis Bohnet",

"Title" : "Vice President",

"Salary" : 106500,

"Department" : "Academic Affairs",

"Hire\_Year" : 1997

}

Problem 12:

# Update Tonja's salary to be $46,200 to reflect a 10% raise.

db.employees.update({Name: "Tonja Balndr"},{'$set':{Salary:(46200)}})

# Output:

> db.employees.find().pretty()

{

"\_id" : ObjectId("54510528bb19e246e561cb38"),

"Name" : "Wendy Yasquez",

"Title" : "Assistant Professor",

"Salary" : 86000,

"Department" : "Computer Science",

"Hire\_Year" : 1998

}

{

"\_id" : ObjectId("5451057cbb19e246e561cb39"),

"Name" : "Raoul Dewan",

"Title" : "Assistant Professor",

"Salary" : 78000,

"Department" : "Physics, Biology",

"Hire\_Year" : 2009

}

{

"\_id" : ObjectId("54510598bb19e246e561cb3a"),

"Name" : "Isabelle Winters",

"Title" : "Assistant Professor",

"Salary" : 92000,

"Department" : "Physics",

"Hire\_Year" : 1995

}

{

"\_id" : ObjectId("545105acbb19e246e561cb3b"),

"Name" : "Jack McDunn",

"Title" : "Assistant Professor",

"Salary" : 101000,

"Department" : "Physics",

"Hire\_Year" : 1993

}

{

"\_id" : ObjectId("54511438bb19e246e561cb3c"),

"Name" : "Tonja Balndr",

"Title" : "Assistant to the Dean",

"Salary" : 46200,

"Department" : "Arts and Sciences",

"Hire\_Year" : 2001

}

{

"\_id" : ObjectId("54511438bb19e246e561cb3d"),

"Name" : "Dennis Bohnet",

"Title" : "Vice President",

"Salary" : 106500,

"Department" : "Academic Affairs",

"Hire\_Year" : 1997

}

Problem 13:

# Remove Raoul Dewan

> db.employees.remove({Name: "Raoul Dewan"})

# Output

> db.employees.find().pretty()

{

"\_id" : ObjectId("54510528bb19e246e561cb38"),

"Name" : "Wendy Yasquez",

"Title" : "Assistant Professor",

"Salary" : 86000,

"Department" : "Computer Science",

"Hire\_Year" : 1998

}

{

"\_id" : ObjectId("54510598bb19e246e561cb3a"),

"Name" : "Isabelle Winters",

"Title" : "Assistant Professor",

"Salary" : 92000,

"Department" : "Physics",

"Hire\_Year" : 1995

}

{

"\_id" : ObjectId("545105acbb19e246e561cb3b"),

"Name" : "Jack McDunn",

"Title" : "Assistant Professor",

"Salary" : 101000,

"Department" : "Physics",

"Hire\_Year" : 1993

}

{

"\_id" : ObjectId("54511438bb19e246e561cb3c"),

"Name" : "Tonja Balndr",

"Title" : "Assistant to the Dean",

"Salary" : 46200,

"Department" : "Arts and Sciences",

"Hire\_Year" : 2001,

"salary" : 46200

}

{

"\_id" : ObjectId("54511438bb19e246e561cb3d"),

"Name" : "Dennis Bohnet",

"Title" : "Vice President",

"Salary" : 106500,

"Department" : "Academic Affairs",

"Hire\_Year" : 1997

}

WriteResult({ "nRemoved" : 1 })

Problem 14:

# Move Raoul Dewan's record to pasteemployees collection and add depart year value of #2014.

> use paste\_employees

> db.paste\_employees.insert({Name: "Raoul Dewan",

... Title: "Assistant Professor",

... Salary: 78000,

... Department: "Physics, Biology",

... Hire\_Year: 2009,

... Depart\_Year: 2014

... })

# Output:

> WriteResult({ "nInserted" : 1 })

> use paste\_employees

switched to db paste\_employees

> db.paste\_employees.find().pretty()

{

"\_id" : ObjectId("545159fd7ab10fc05d9b2ceb"),

"Name" : "Raoul Dewan",

"Title" : "Assistant Professor",

"Salary" : 78000,

"Department" : "Physics, Biology",

"Hire\_Year" : 2009,

"Depart\_Year" : 2014

}