CS585: Big Data Management

Project4 (MongoDB)

Xiaosong Wen (xwen2) & Tianyu Li (tli)

Install MongoDB v4.2.1 on Ubuntu 18.04:

sudo apt install mongodb-org

Start service:

sudo systemctl enable mongod
sudo systemctl start mongod

Question 1:

Setup:

Create collection 'test' and insert data:
http://docs.mongodb.org/manual/reference/bios-example-collection/

1) Write a CRUD operation(s) that inserts the following new records into the collection:

Code:

```
db.test.insertMany([
{
    "_id" : 20,
    "name" : {
        "first" : "Alex",
        "last" : "Chen"
        },
    "birth" : ISODate("1933-08-27T04:00:00Z"),
    "death" : ISODate("1984-11-07T04:00:00Z"),
    "contribs" : [
        "C++",
        "Simula"
    ],
    "awards" : [{
            "award" : "WPI Award",
            "year" : 1977,
            "by" : "WPI"
        }]
},
{
    "_id" : 30,
    "name" : {
        "first" : "David",
        "last" : "Mark"
    },
    "birth" : ISODate("1911-04-12T04:00:00Z"),
    "death" : ISODate("2000-11-07T04:00:00Z"),
    "contribs" : [
        "C++",
        "FP",
        "Lisp",
    ],
    "awards" : [
        {
            "award" : "WPI Award",
            "year" : 1963,
            "by" : "WPI"
        },
        {
            "award" : "Turing Award",
            "year" : 1966,
            "by" : "ACM"
        }
    ]
    }
])
```

Result:

{ “acknowledged” : true, “insertedIds” : [20, 30] }

2)Report all documents of people who got less than 3 awards or have contribution in “FP”

Code:

```
db.test.find(
    {
        $or: [
            { $where: "this.awards ? this.awards.length < 3 : false" },
            { contribs: "FP" }
        ]
    }
)
```

Result:

```
{ "_id" : 1, "name" : { "first" : "John", "last" : "Backus" }, "birth" : ISODate("1924-12-03T05:0
0:00Z"), "death" : ISODate("2007-03-17T04:00:00Z"), "contribs" : [ "Fortran", "ALGOL", "Backus-Na
ur Form", "FP" ], "awards" : [ { "award" : "W.W. McDowell Award", "year" : 1967, "by" : "IEEE Com
puter Society" }, { "award" : "National Medal of Science", "year" : 1975, "by" : "National Scienc
e Foundation" }, { "award" : "Turing Award", "year" : 1977, "by" : "ACM" }, { "award" : "Draper P
rize", "year" : 1993, "by" : "National Academy of Engineering" } ] }
{ "_id" : 6, "name" : { "first" : "Guido", "last" : "van Rossum" }, "birth" : ISODate("1956-01-31
T05:00:00Z"), "contribs" : [ "Python" ], "awards" : [ { "award" : "Award for the Advancement of F
ree Software", "year" : 2001, "by" : "Free Software Foundation" }, { "award" : "NLUUG Award", "ye
ar" : 2003, "by" : "NLUUG" } ] }
{ "_id" : 8, "name" : { "first" : "Yukihiro", "aka" : "Matz", "last" : "Matsumoto" }, "birth" : I
SODate("1965-04-14T04:00:00Z"), "contribs" : [ "Ruby" ], "awards" : [ { "award" : "Award for the
Advancement of Free Software", "year" : "2011", "by" : "Free Software Foundation" } ] }
{ "_id" : 9, "name" : { "first" : "James", "last" : "Gosling" }, "birth" : ISODate("1955-05-19T04
:00:00Z"), "contribs" : [ "Java" ], "awards" : [ { "award" : "The Economist Innovation Award", "y
ear" : 2002, "by" : "The Economist" }, { "award" : "Officer of the Order of Canada", "year" : 200
7, "by" : "Canada" } ] }
{ "_id" : 20, "name" : { "first" : "Alex", "last" : "Chen" }, "birth" : ISODate("1933-08-27T04:00
:00Z"), "death" : ISODate("1984-11-07T04:00:00Z"), "contribs" : [ "C++", "Simula" ], "awards" : [
 { "award" : "WPI Award", "year" : 1977, "by" : "WPI" } ] }
{ "_id" : 30, "name" : { "first" : "David", "last" : "Mark" }, "birth" : ISODate("1911-04-12T04:0
0:00Z"), "death" : ISODate("2000-11-07T04:00:00Z"), "contribs" : [ "C++", "FP", "Lisp" ], "awards
" : [ { "award" : "WPI Award", "year" : 1963, "by" : "WPI" }, { "award" : "Turing Award", "year"
: 1966, "by" : "ACM" } ] }
```

3) Update the document of “Guido van Rossum” to add “OOP” to the contribution list.

Code:

```
db.test.update(
    {
        name: {
            "first": "Guido",
            "last": "van Rossum"
        }
    },
    {
        $push: { contribs: "OOP" }
      }
)
```

Result:

WriteResult({ “nMatched” : 1, “nUpserted” : 0, “nModified” : 1 })

4) Insert a new filed of type array, called “comments”, into the document of “Alex Chen” storing the following comments: “He taught in 3 universities”, “died from cancer”, “lived in CA”

Code:

```
db.test.update(
    {
        name: {
            "first": "Alex",
            "last": "Chen"
        }
    },
    {
        $set: {
            comments: [
                "He taught in 3 universities",
                "died from cancer",
                "lived in CA"
            ]
        }
    }
)
```

Result:

WriteResult({ “nMatched” : 1, “nUpserted” : 0, “nModified” : 1 })

5) For each contribution by “Alex Chen”, say X, list the peoples’ names (first and last) who have contribution X.

Code:

```
db.test.find(
    {
        name: {
            "first": "Alex",
            "last": "Chen"
        }
    }
).forEach(
    function(u){
        contributions=u.contribs
    }
);
db.test.aggregate(
        [
            {$unwind:"$contribs"},
            {$match:{'contribs':{$in: contributions}}},
            {$group:{_id: "$contribs",people:{$push:"$name"}}}
        ]
)
```

Result:

{ “_id” : “Simula”, “people” : [{ “first” : “Kristen”, “last” : “Nygaard” }, { “first” : “Ole-Johan”, “last” : “Dahl” }, { “first” : “Alex”, “last” : “Chen” }] }
{ “_id” : “C++”, “people” : [{ “first” : “Alex”, “last” : “Chen” }, { “first” : “David”, “last” : “Mark” }] }

6) Report the distinct organization that gave awards. This information can be found in the “by” field inside the “awards” array.

Code:

```
db.test.distinct(
    'awards.by'
)
```

Result:

```
[
    "ACM",
    "IEEE Computer Society",
    "National Academy of Engineering",
    "National Science Foundation",
    "Inamori Foundation",
    " British Computer Society",
    "Data Processing Management Association",
    "United States",
    "IEEE",
    "Norwegian Data Association",
    "Free Software Foundation",
    "NLUUG",
    "The Japan Prize Foundation",
    "Canada",
    "The Economist",
    "WPI"
]
```

7) Delete from all documents any award given on 2011.

Code:

```
db.test.update(
    {
        "awards.year": 2011
    },
    {
        $pull: {
            awards: {
                year: 2011
            }
        }
    },
    {
        multi: true
    }
)
```

Result:

WriteResult({ “nMatched” : 1, “nUpserted” : 0, “nModified” : 1 })

8) Report only the names (first and last) of those individuals who won at least two awards in 2001.

Code:

```
db.test.aggregate(
    [
        {$unwind:"$awards"},
        {$match:{'awards.year':2001}},
        {$group:{_id: "$name",count:{$sum:1}}},
                {$match:{count:{$gte: 2}}},
                {$project:{name:1}}
    ]
)
```

Result:

{ “_id” : { “first” : “Ole-Johan”, “last” : “Dahl” } }
{ “_id” : { “first” : “Kristen”, “last” : “Nygaard” } }

9) Report the document with the largest id.

Code:

```
var cursor = db.test.find().sort({ _id: -1 }).limit(1);
var max_id = cursor.next()._id
var doc = db.test.findOne(
    {
        _id: max_id
    }
)
doc
```

Result:

```
{
    "_id" : ObjectId("51e062189c6ae665454e301d"),
    "name" : {
        "first" : "Dennis",
        "last" : "Ritchie"
    },
    "birth" : ISODate("1941-09-09T04:00:00Z"),
    "death" : ISODate("2011-10-12T04:00:00Z"),
    "contribs" : [
        "UNIX",
        "C"
    ],
    "awards" : [
        {
            "award" : "Turing Award",
            "year" : 1983,
            "by" : "ACM"
        },
        {
            "award" : "National Medal of Technology",
            "year" : 1998,
            "by" : "United States"
        },
        {
            "award" : "Japan Prize",
            "year" : 2011,
            "by" : "The Japan Prize Foundation"
        }
    ]
}
```

10) Report only one document where one of the awards is given by “ACM”.

Code:

```
db.test.findOne(
    {'awards.by': "ACM"}
)
```

Result:

```
{
    "_id" : 1,
    "name" : {
        "first" : "John",
        "last" : "Backus"
    },
    "birth" : ISODate("1924-12-03T05:00:00Z"),
    "death" : ISODate("2007-03-17T04:00:00Z"),
    "contribs" : [
        "Fortran",
        "ALGOL",
        "Backus-Naur Form",
        "FP"
    ],
    "awards" : [
        {
            "award" : "W.W. McDowell Award",
            "year" : 1967,
            "by" : "IEEE Computer Society"
        },
        {
            "award" : "National Medal of Science",
            "year" : 1975,
            "by" : "National Science Foundation"
        },
        {
            "award" : "Turing Award",
            "year" : 1977,
            "by" : "ACM"
        },
        {
            "award" : "Draper Prize",
            "year" : 1993,
            "by" : "National Academy of Engineering"
        }
    ]
}
```

Question 2:

Using the database in **Question 1**

1). Write an aggregation query that group by the award name

Code

```
db.test.mapReduce(
    function(){
        if(this.awards!=null)
         for(var i=0;i<this.awards.length;i++)
                emit(this.awards[i].award,1);},
    function(key,values){return Array.sum(values)},
    {out:"award"})
db.award.find()
```

Result

```
#council output for mapReduce
{
    "result" : "award",
    "timeMillis" : 274,
    "counts" : {
        "input" : 12,
        "emit" : 28,
        "reduce" : 7,
        "output" : 17
    },
    "ok" : 1
}

#output for .find()
{ "_id" : "Award for the Advancement of Free Software", "value" : 2 }
{ "_id" : "Computer Sciences Man of the Year", "value" : 1 }
{ "_id" : "Distinguished Fellow", "value" : 1 }
{ "_id" : "Draper Prize", "value" : 1 }
{ "_id" : "IEEE John von Neumann Medal", "value" : 2 }
{ "_id" : "Japan Prize", "value" : 1 }
{ "_id" : "Kyoto Prize", "value" : 1 }
{ "_id" : "NLUUG Award", "value" : 1 }
{ "_id" : "National Medal of Science", "value" : 2 }
{ "_id" : "National Medal of Technology", "value" : 2 }
{ "_id" : "Officer of the Order of Canada", "value" : 1 }
{ "_id" : "Rosing Prize", "value" : 2 }
{ "_id" : "The Economist Innovation Award", "value" : 1 }
{ "_id" : "Turing Award", "value" : 6 }
{ "_id" : "W. W. McDowell Award", "value" : 1 }
{ "_id" : "W.W. McDowell Award", "value" : 1 }
{ "_id" : "WPI Award", "value" : 2 }
```

2). Write an aggregation query that groups by the birth year

Code:

```
db.test.aggregate(
    {$match:{birth:{$exists:true}}},
    {$group:{_id:{$year:"$birth"},
    idarray:{$addToSet:"$_id"}}
})
```

Result:

```
{ "_id" : 1924, "idarray" : [ 1 ] }
{ "_id" : 1955, "idarray" : [ 9 ] }
{ "_id" : 1933, "idarray" : [ 20 ] }
{ "_id" : 1965, "idarray" : [ 8 ] }
{ "_id" : 1927, "idarray" : [ ObjectId("51df07b094c6acd67e492f41") ] }
{ "_id" : 1906, "idarray" : [ 3 ] }
{ "_id" : 1926, "idarray" : [ 4 ] }
{ "_id" : 1931, "idarray" : [ 5 ] }
{ "_id" : 1956, "idarray" : [ 6 ] }
{ "_id" : 1941, "idarray" : [ ObjectId("51e062189c6ae665454e301d") ] }
{ "_id" : 1911, "idarray" : [ 30 ] }
```

3). Report the document with the smallest and largest _ids.

Code:

```
db.test.find().sort({_id:1}).limit(1);
db.test.find().sort({_id:-1}).limit(1);
```

Result:

```
{ "_id" : 1, "name" : { "first" : "John", "last" : "Backus" }, "birth" : ISODate("1924-12-03T05:0
0:00Z"), "death" : ISODate("2007-03-17T04:00:00Z"), "contribs" : [ "Fortran", "ALGOL", "Backus-Na
ur Form", "FP" ], "awards" : [ { "award" : "W.W. McDowell Award", "year" : 1967, "by" : "IEEE Com
puter Society" }, { "award" : "National Medal of Science", "year" : 1975, "by" : "National Scienc
e Foundation" }, { "award" : "Turing Award", "year" : 1977, "by" : "ACM" }, { "award" : "Draper P
rize", "year" : 1993, "by" : "National Academy of Engineering" } ] }
{ "_id" : ObjectId("51e062189c6ae665454e301d"), "name" : { "first" : "Dennis", "last" : "Ritchie"
 }, "birth" : ISODate("1941-09-09T04:00:00Z"), "death" : ISODate("2011-10-12T04:00:00Z"), "contri
bs" : [ "UNIX", "C" ], "awards" : [ { "award" : "Turing Award", "year" : 1983, "by" : "ACM" }, {
"award" : "National Medal of Technology", "year" : 1998, "by" : "United States" }, { "award" : "J
apan Prize", "year" : 2011, "by" : "The Japan Prize Foundation" } ] }
```

Question 3:

Assume we model the records and relationships in Figure 1 using the Parent-Referencing model (Slide 4 inMongoDB-3).

```
new_categories = [
    { _id: "MongoDB", parent: "Databases" },
    { _id: "dbm", parent: "Databases" },
    { _id: "Databases", parent: "Programming" },
    { _id: "Languages", parent: "Programming" },
    { _id: "Programming", parent: "Books" },
    { _id: "Books", parent: null },
];
db.ParentRef.insert(new_categories);
```

1). Write a query to report the ancestors of “MongoDB”.
Code:

```
var stack = [], ancestors = [], level = 0;
var category = db.ParentRef.findOne({_id: "MongoDB"});
stack.push(category);
while (stack.length > 0) {
    level++;
    var current = stack.pop();
    var parent = db.ParentRef.findOne({_id: current.parent});
    if (parent) {
        ancestors.push({ Name: parent._id, Level: level });
        stack.push(parent);
    }
}
ancestors
```

Result:

```
 [
    {
        "Name" : "Databases",
        "Level" : 1
    },
    {
        "Name" : "Programming",
        "Level" : 2
    },
    {
        "Name" : "Books",
        "Level" : 3
    }
]
```

2).You are given only the root node, i.e., _id = “Books”, write a query that reports the height of the tree.
Code:

```
var stack = [], visitedIds = {}, level = 0;
var category = db.ParentRef.findOne({_id: "Books"});
stack.push(category);
while (stack.length > 0) {
    var current = stack.pop();
    var children = db.ParentRef.find({parent: current._id});
    if (!(current.parent in visitedIds)) {
        level++;
        visitedIds[current.parent] = 1;
    }

    while (children.hasNext()) {
        var child = children.next();
        stack.push(child);
    }
}
level
```

Result:

```
4
```

Assume we model the records and relationships in Figure 1 using the Child-Referencing model (Slide 9 in MongoDB-3)

```
new_categories2 = [
    { _id: "MongoDB", children: [] },
    { _id: "dbm", children: [] },
    { _id: "Databases", children: ["dbm", "MongoDB"] },
    { _id: "Languages", children: [] },
    { _id: "Programming", children: ["Databases", "Languages"] },
    { _id: "Books", children: ["Programming"] },
 ];
db.ChildRef.insert(new_categories2);
```

3). Write a query to report the parent of “dbm”.
Code:

```
parent = db.ChildRef.findOne(
    { children: { $in: ["dbm"] }}
);
parent
```

Result:

```
{ "_id" : "Databases", "children" : [ "dbm", "MongoDB" ] }
```

4).Write a query to report the descendants of “Books”.
Code:

```
var stack = [], descendants = [];
var category = db.ChildRef.findOne({_id: "Books"});
stack.push(category);
while (stack.length > 0) {
    var current = stack.pop();
    var children = db.ChildRef.find({_id: {$in: current.children}});

    while (children.hasNext()) {
        var child = children.next();
        descendants.push(child._id);
        stack.push(child);
    }
}
descendants
```

Result:

```
[ "Programming", "Databases", "Languages", "MongoDB", "dbm" ]
```