Database - P54

PostgreSQL metaphone : algorithm for creating string representation of word sound
can define how many characters in the output string
如 metaphone('Robin Williams', 7) → 'RBNWLMS'
other functions contained in fuzzystrmatch module
dmetaphone() : double metaphone
dmetaphone_alt() : alternate name pronunciation
soundex() : algorithm developed in the 1880s
by the U.S. Census to compare American surname
如 dmetaphone('Robin Williams') → 'RPNL'
dmetaphone_alt('Robin Williams') → 'RPNL'
soundex('Robin Williams') → 'R154'

hypercube search : cube datatype, for multidimensional vector
package cube method to find the closest point
within boundary of hypercube
cube_ur_coord(vector, dimension)
用于从 vector 中提取指定维度的值.
如 cube_ur_coord('(1,3,5,7,11)', 6) → 11
cube_distance(point1, point2)
用于计算两个向量之间的距离(欧几里得距离)
如 cube_distance('(0,0,0,0)','(1,1,1,1)') → 2
focus on likely candidate by using bounding cube
rather than computing distance at every point
cube_enlarge(cube, radius, dimensions)
radus : 对于每个维度的 value, 扩展范围为 value ± radius
dimensions : 新生成的向量的维度.
如果大于原向量维度, 则先对原向量补充值为 0
如 cube_enlarge('(1,2,3)', 2, 4)
→ '(-1,0,1,2),(3,4,5,2)'
运算符 @> : contained in bounding hypercube
<bounding_hypercube> @> <hypercube>
如 cube_enlarge('(1,2,3)', 2, 3) @> '(-2, 0, 1)'
→ 'f' (false)

Database - P55

MongoDB

designed to scale, main design golals : performance, ease of data access
document database, allowing for storage of object nested to any depth
not enforce schema, document can contain field / type unique
great choice for from small local project to huge production deployment

JSON document database
BSON : stored in binary form of JSON
_id : unique identifier assigned to each document stored in MongoDB
subdocument : attribute contained in document
store all data together, instead of creating and joining separate tables

sweet spot between flexible queryability and distributed nature
relational database : 3角decode数据 -> 使用数据
document database : 使用数据 > 3角decode数据

collection : 类似于 table, 可以在添加数据时动态地添加
当使用 insert() 把数据向不存在的 collection 插入数据时
自动地在当前的 database 中创建 collection
如 db.collectionname.insert()

when add new document, will be providing JSON to do

_id : unique field appended to every created document automatically in MongoDB

ObjectId (12 byte)	time	mid	pid	inc
	4 byte	3 byte	2 byte	3 byte

time : timestamp (4-byte)
mid : 3-byte client machine ID
pid : 2-byte client process ID
inc : 3-byte increment counter

using JavaScript as native language and to do complex thing

Database - P56

MongoDB

find() : MongoDB 中以 db.collection.find() 函数实现数据查询

基本调用形式为 db.collection.find([<query>][, <projection>])

当不传入参数调用时返回全部数据

即 db.collection.find() 等价于 SELECT * FROM table

<query> ::= "{" ("{"<field> : <selector>"}")* "}" 以逗号分隔

其中 field 可以是文件的属性名，如 _id

也可以内部文件 (embedded document) 的属性名

以双引号包括，即 "<embedded_document>.<attr>"

selector 是对于指定 field 的筛选条件

基本结构为 {(<operator> : <value>)*}，以逗号分隔

比较运算符：

$lt (<), $lte (<=), $eq (==), $ne (!=), $gte (>=), $gt (>)

以及用于匹配数组中元素的 $in / $nin

如 {$nin : [<value1>, ..., <valueN>]}

逻辑运算符：

$and (逻辑与), $or (逻辑或), $not (逻辑非), $nor (逻辑或非)

注意 {$nor : [<expr1>, ..., <exprN>]} 为真

当且仅当所有 <expr> 为假或其中的 <field> 不存在

元素运算符：

$exists : 用于判断指定 <field> 存在/不存在，{$exists : true/false}

$type : 用于判断指定 <field> 是否是指定类型 (或之一)

{$type : <BSON type>} / {$type : [<type1>, ..., <typeN>]}

数组运算符：

$all : 用于判断 <field> 是否包含指定的所有值

$in : 类似于 any，用于判断 <field> 是否至少包含一个指定的值

$size : 用于判断 <field> 是否包含指定数量的元素

字符串运算符：

$text : 用于在 <field> 中搜索文本，默认为 false

$search : <string> 用于指定文本，$caseSensitive : <boolean> 表示大小写敏感

$regex : 用于匹配正则表达式，

MongoDB 使用 PCRE (Perl-compatible regular expression)

如 'Perl' ~= /^P.*e.*$l/i

Database - P57

MongoDB find(): PCRE (Perl-compatible regular expression) 的基本构造

use // delimiter, 可以在后边加 modifier

/i : case-insensitive.

metacharacter	purpose	where
\	escaped character	always
^	match the beginning	not in []
$	match the end	not in []
.	match any single character, except newline under /s, include newline	not in []
|	alternation	not in []
( )	grouping	not in []
[ ]	bracketd character class	not in []
-	indicate a range	only in []
^	complement the [] class	at beginning of []

quantifier	description
*	match 0 or more times
+	match 1 or more times
?	match 0 or 1 time
{n}	match exactly n times
{n,}	match at least n times
{n,m}	match at least n times and at most m times

$elemMach 运算符: specified document (or nested document)
match all of criteria
can also utilize advanced operator

$mod 运算符: 对于指定的整数类型的field, 计算其模 n 取余是否匹配指定值
{field: {$mod: [<divisor>, <remainder>]}}

<projection>:: = {<field1>: <value>; <field2>: <value>, ...}
value 可以取 0 表示不显示(exclusion), 取 1 表示显示 field (inclusion)
find() 中 _id 是默认显示的, 除非显式地指定 _id: 0
不允许同时出现 exclusion 和 inclusion
除非是 _id 的 exclusion

Database - P58

MongoDB

update() : 在MongoDB中, db.collection.update() 用于更新 collection 中的数据

基本调用形式为: db.collection.update(
<query>, <update>,
{ upsert : <boolean>,
multi : <boolean>,
... })

query 与 find() 函数中的查询条件一样 (query selector)
用于筛选需要更新数据的 document

update : 需进行的更新操作
基本定义结构为 { <operator1> : {<field1> : <value1>, ... },
<operator2> : {<field1> : <value1>, ... }, ... }

field 运算符:
$currentDate : 将指定 field 的值更新为当前时间
{$currentDate : { <field1> : <typeSpecification1>, ... }}
typeSpecification : 定义存储时间的类型
{$type : "timestamp"} / {$type : "date"}
如果传入布尔值 true, 则使用 {$type : "date"} (Date 类型)
$inc : 对指定的 field 的值增加指定的 value
{$inc : {<field1> : <value1>, <field1> : <value2>, ... }}
$mul : 对指定的 field 的值乘以指定的 value
{$mul : {<field1> : <value1>, <field2> : <value2>, ... }}
$min / $max : 只在指定 value 小于/大于 指定 field 的值时更新
{ $min | $max : {<field1> : <value1>, ... }}
$rename : 重命名指定的 field
{ $rename : {<field1> : <newname1>, ... }}
$set : 更新指定 field 的值
{ $set : { <field1> : <value1>, <field2> : <value2>, ... }}
特别注意: MongoDB 是 document-oriented 而非 attribute-oriented
所以如果不使用 $set, 则会替换原有的 document
$unset : 删除指定的 field
{$unset : {<field1> : "", <field2> : "", ... }}
$setOnInsert : 仅在更新结果为插入新的 document 时生效
当更新当前已存在的 document 时不生效

Database - P59

MongoDB

update() : array运算符 : 当指定的field不是array时抛出错误

$addToSet : 当指定的value不在指定的field中时插入field

{$addToSet : {<field1> : <value1>, ···}}

$push : 向指定的field压入新元素

{$push : {<field1> : <value1>, ···}}

$each : 是用于$addToSet 和 $push 的 modifier

用于在指定value时, 如果value是一个array

$each将array中每个元素单独操作

否则将array视为一个完整的参数文

{$each : [<value1>, <value2>, ···]}

$pop : 用于从指定field中弹出第一个/最后一个元素

{$pop : {<field1> : <-1 | 1>, ···}}

其中 -1表示第1个元素, 而1表示最后一个元素

$pull : 用于从指定field中移除一个指定值或符合条件的全部[redacted]元素

{$pull : {<field1> : <value1 | condition1>, ···}}

$pullAll : 用于从指定field中移除指定列表中的所有元素

{$pullAll : {<field1> : [<value1>, <value2>, ···]}}

upsert : 传入可选的参数, [redacted]布尔值 <boolean>, 默认为false

true : 如果query没有匹配到任何[redacted]document

则插入新document并进行update操作

multi : 传入可选的布尔值参数 <boolean>, 默认为false

true : 如果query匹配到多个document, 则更新全部document

特别注意, 由于默认值为false,

于是MongoDB中的update默认只更新符合query的第一个document

reference : join in MongoDB inefficient, due to distributed nature

occasionally useful for document to reference each other

{"$ref" : <value>, "$id" : <value>, "$db" : <value>}

$ref : hold name of collection where referenced document reside

$id : contain value of _id field in the referenced document

$db : optional, contain name of database where referenced document

only some drivers support $db

Database - P60

MongoDB

remove() : 在MongoDB中, db.collection.remove() 用于从collection中删除数据

基本调用形式为: db.collection.remove(<query>,
{ justOne : <boolean>,
... })

query 与 find() 函数中的查询条件(query selector)一样
用于筛选需要删除的 document

justOne : 传入可选的布尔值参数 <boolean>, 默认值为 false
true : 如果 query 匹配到多个 document
则删除所有匹配到的 document

特别注意: 会删除整个 document (entire document will be deleted)

$where : request MongoDB run decision function across all documents
always last resort : run slowly / not use index / cannot be optimized
{ $where : <function | expression> }
通常用于 query 的 function 的返回值和 expression 的计算值应是 <boolean>
如 { $where : function() { return this.score >= 60; } }
或 { $where : "this.score >= 60" }
MongoDB have to blindly run the decision function against each document
despite no guarantee the field exists

indexing : useful feature of MongoDB in the name of enhanced query performance
provide several data structure for indexing
B-tree / two-dimensional / spherical GeoSpatial index
db.collection.getIndexes() 函数用于列出 collection 中定义的 Index
db.collection.ensureIndex() 函数用于在 collection 创建 Index
也可以使用 db.collection.createIndex() 函数
调用形式为 db.collection.ensureIndex(<keys>, <option>)
keys : 指定定义索引的 field, 以及 排序
1表示升序, -1表示降序
option : 用于定义 Index 的选项
background : 可选 <boolean>, true : build in background so operation does not block other database activity
unique : 可选 <boolean>, true : unique index
not accept insertion/update match existing value of index
name : 可选的 指定的 Index 名字

Database - P61

MongoDB

indexing: more costly to create than relational database
due to the schemaless nature of MongoDB
should be created in the background, done during non-peak time
running index creation manually / automatically
more control on when and how index rolled out

explain(): method to output the detail of given operation
如 db.collection.find().explain("executionStats").executionStats

system.profile: store information about query
timestamp the query take place
performance information, such as executionTimeMillisEstimate
system.profile 可以像一般 collection 进行 find()
如 db.system.profile.find().pretty()
可以通过 db.setProfilingLevel() 函数调整 profiling level
level 1: only store slow query greater than 100 milliseconds
level 2: store all queries

count(): provide the number of document included in result set
基本调用形式为 db.collection.count(<query>, <options>)
query 与 find() 函数中的查询条件 (query selector) 一样
option: 可选地包含一些可选属性
limit: maximum number of documents to count
skip: number of documents to skip before counting
特别地对于相同的查询条件 query
db.collection.count(query) 与 db.collection.find(query).count() 等价

distinct(): collect the result set into array of unique result
基本调用形式为 db.collection.distinct(<field>, <query>)
field: 返回的不重复结果所在的 field,
query 与 find() 函数中的查询条件相同
注意返回的是指定 field 中不重复结果的数组，而非 document
当 field 中的值为数组时，则其中元素会被视为独立的元素
当有可用的 Index 时，distinct() 可以使用 Index

Database - P62

MongoDB aggregate() : return document according to user-defined logic

基本调用形式为 db.collection.aggregate([{<stage>}, ...])

aggregation pipeline : sequence of data aggregation operation or stage

$group : group by specified identifier and apply accumulator expression

{$group : {_id : <expression>, <field1> : {<accumulator1> : <expr>}, ...}}

_id : 表示group所使用的tuple，也可以是null

可以是 {<name> : <expression>, ...}

如 _id : {month : {$month : "$date"}, day : {$dayOfMonth : "date"}

如果 _id : null 则将整个collection视为一个group

field : 返回document中的属性名

accumulator operator :

$addToSet : array of unique expression value for each group

order of element undefined

$avg : average of numerical value, ignore non-numerical value

$first / $last : value of first/last document for each group

only order if document in defined order

$max / $min : highest / lowest expression value for each group

$sum : sum of numerical value, ignore non-numerical value

当使用 {$sum : 1} 时等同于 count

$mergeObjects : document created by combining input document for each group

$push : array of expression for each group

$stdDevPop / $stdDevSamp : population / sample standard deviation of input value

$match : filter document to pass only match specified condition

基本形式为 {$match : {<query>}}

其中 query 与 find() 函数中的 read operation query syntax 相同

通常写在 pipeline 的最开始

$sort : 基本形式为 {$sort : {<field1> : <sort order>, ...}}

field : 用于排序的属性，优先级以从高到低传入

sort order : 1 / -1 表示从升序 / 降序

或者 {$meta : "textScore"}, by computed "textScore" metadata in descending order

Database - P63

MongoDB aggregate() : $project: pass along document with requested field to next stage
specified field can be existing field or newly computed field
基本形式为：{$project : {<specification>}}
specification : <field> : <1 | true> : inclusion of field
_id : <0 | false> : exclusion of _id
<field> : <0 | false> : exclusion of field

$limit : limit number of document passed to next stage
基本形式为：{$limit : <positive integer>}
注意 $limit 在 pipeline 中所处的位置不同会对结果产生影响

Mongo provide large level of flexibility in aggregation pipeline
warn : aggregation can be very slow
if large number of stages / perform against large collection
schemaless Mongo can only optimize query so far
powerful and performant if structure not require large transformation to be useful

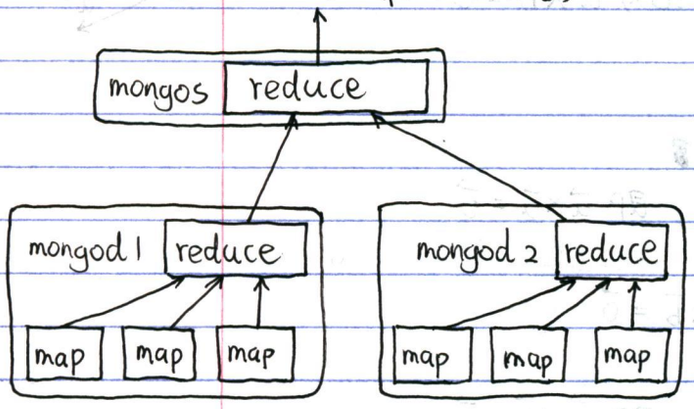
server-side command : several pre-built command in Mongo in addition to JavaScript function
most executed on the server
some restricted to running only on admin database (use admin)
如 db.runCommand("top")
will output access detail about all collection on the server
db.listCommands() , list the commands
注意 db.runCommand({"find" : "collection"}) 和 db.collection.find() 输出不同
db.runCommand() : operation executed on the server
db.collection.find() : wrapper function
created for convenience by shell's JavaScript interface
db.system.js.save({_id : <identifier> , value : <expression>})
用于将函数保存在 db.system
allow load and execute function stored on the server
db.system : normal collection like other in Mongo
如 _id : name of function , value : definition of function
db.loadServerScripts() : load server-side function

Database - P64

MongoDB MapReduce : designed for performing computation over large dataset ideally using distributed system

基本调用形式为 db.runCommand ({
mapReduce : <collection>,
map : <function>,
reduce : <function>,
finalize : <function>,
out : <output>,
query : <document>,
sort : <document>,
limit : <number>,
scope : <document>,
... })

db.runCommand ({'mapReduce'... })



mapReduce : 指定用于 MapReduce 运算的 collection
注意 view 不支持 MapReduce 运算

map : JavaScript function associate / map one value with one key and emit the key : value pair
responsible for transforming each input document into 0 or more output document
基本定义形式为 function ( ) {
...
emit (key, value); }
reference the current document as "this" within function
should not access the database for any reason
should be pure, have no side effect
single emit hold half of MongoDB's maximum BSON document size (16MB)
may optionally call emit(key, value) any number of time

query : 可选地用于筛选用于 map 函数的输入 document
与 find() 函数中的查询条件 (query selector) 相同

limit : 可选地指定用于 map 函数的输入 document 的最大数量

Database - P65

MongoDB MapReduce: sort: 可选地对输入document进行排序, useful for optimization
sort key must be in existing index for collection
specify sort key to be same as emit key for fewer reduce operation

reduce: JavaScript function "reduce" to single object
from all the values associated with a particular key
基本定义形式为: function(key, values){
...
return result;}
should not access the database, even perform read operation
should not affect the outside system
will not call reduce for the key has only one single value
values argument: array of value object mapped to the key
can invoke more than once for the same key
previous output from reduce function
will become one of input values to next reduce function
input must not be larger than half of maximum BSON document size
may be violated when larger document joined in subsequent step
type of return object must be identical to type of value emitted
must be associative(结合律), idempotent(幂等律), commutative(交换律)
reduce(key, [C, reduce(key, [A,B])]) == reduce(key, [C,A,B])
reduce(key, [reduce(key, values)]) == reduce(key, values)
reduce(key, [A,B]) == reduce(key, [B,A])

finalize: 可选地定义函数, follow the reduce method and modify the output
基本定义形式为: function(key, reducedValue){
...
return modifiedObject;}
should not access the database for any reason
should be pure, have no side effect

scope: 可选地定义全局变量(global variable), {identifier1: value1, ...}
map/reduce/finalize can access variable defined in scope parameter

Database - P66

MongoDB

MapReduce: out: 指定输出 MapReduce 运算的结果的位置
either output to collection or return result inline
有基本定义的形式有
out: <collectionName>
out: { <action>: <collectionName>
[, db: <dbName>]
[, sharded: <boolean>]
[, nonAtomic: <boolean>] }
out: {inline: 1}
action 指定对于 <collectionName> 的操作
replace: 如果 <collectionName> 存在则替换
merge: 存入已存在的 <collectionName>
overwrite existing document with same key
reduce: 与 merge 类似, 对于有相同 key 的已存在 document
调用 reduce function
db: name of database to write output
sharded: true: shard output if output database enabled sharding
nonAtomic: true: not locking database during post-processing

highly available (HA): enabling replication across servers
sharding collection into pieces
running query in parallel

replica set: Mongo support data consistency and partition tolerance (CP)
through sharding, but cost
address weakness in sharding model by duplicating data
mongod --smallfiles --replSet <dbName> --port <port>
docker-compose file bring up database instances
rs.initiate() 用于初始化 replica set
基本调用形式为: rs.initiate({
_id: <replicaSetName>
members: [{_id: <int>, host: <string>, ...}
...]
})

Database - P67

MongoDB replica set : access to new object rs (replica set) in addition to db (database)

status() : check replica set is up and running

→ Member <name> is now is state PRIMARY/SECONDARY

PRIMARY server will be the master server

may be the first server started

once master node shutdown

one of other servers take over and be promoted to master

only one master node per replica set

cannot write to/read from SECONDARY node

each mongod service get one vote

one with freshest data elected the new master

when bring back up server shut down

node will go into recovery state

and try to resync data with new master node

one write in Mongo replica set not considered successful

until most nodes have a copy

data duplicated across other servers within replica set

network partition : network disconnect between nodes

the majority of nodes still communicate now constitute the network

MongoDB expect odd number of total nodes in replica set

if no partition have clear majority of network

no set of servers can service request, system is down

even the set with original master node

CouchDB built to allow multiple master nodes

MongoDB simply not allow multiple master nodes, to deal with conflict between

(CP) always know the most recent value

concern strong consistency on write

arbiter : voting but not replicating server in Mongo replica set

not always have odd number of replicating servers

or increase voting right on server

useful for breaking ties

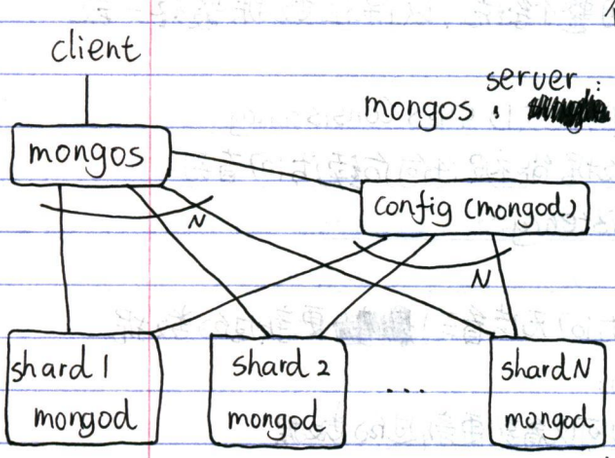
在 rs.initiate() 函數中初始化 arbiter server

rs.initiate({... arbiterOnly : true ...})

Database - P68

MongoDB

Sharding : core goal of Mongo : provide safe and quick handling of large dataset
range of values split (sharded) onto servers
mongod --smallfiles --shardsvr --port <port>
special parameter necessary to be considered shard server
mean the server capable of sharding
config server : regular mongod process in Mongo
keep track of which server own which data
mongod --smallfiles --configsvr --replSet <replName> --port <port>
in docker-compose file
create second replica set for cluster's configuration
在函数 rs.initiate() 中初始化 config server cluster
rs.initiate({... configsvr : true ... })



mongos : server : connect to mongoconfig config server
to keep track of sharding information stored
mongos : single point of entry for client
lightweight clone of mongod
support any command can run directly at mongod
mongos --configdb <configUrl> --port <port>
create process in docker-compose file
access to new object sh (sharding) in addition to db and rs
sh.addShard() : add shard replica set to sharded cluster
must be run on mongos instance
调用形式为 sh.addShard("<replica_set>/<hostname><:port>, ...")
db.runCommand({enablesharding : <db_name>})
configure database to shard
db.runCommand({shardcollection : "<db_name>.<collection>",
key : <shardkey>, ... })
configure collection and field to shard
shardkey : {field : 1}, ranged based sharding
{field : "hashed"}, specify hashed shard key
db.collection.getShardDistribution()
print data distribution statistics for sharded collection

Database - P69

MongoDB strength : ability to handle huge amount of data and request through replication and horizontal scaling
very flexible data model, no need to conform to schema
simply nest any value
built to be easy to use, command similar to SQL database concept different enough to solve problem RDBMS not good at

weakness : encourage denormalization of schema
typo and bug by allowing arbitrary data to be inserted into any collection difficult to debug or find a misspelling
not much advantage to flexibility of schemaless database if already have mature and locked-down schema
setting up clustered MongoDB instance require forethought database operate better as large cluster of server

Geospatial query : core of geospatial query lies in indexing
geospatial data indexed using special index : geohash
find not only values of specified value and range quickly but also nearby values in ad-hoc query quickly
legacy coordinate pair : store location use 2d index
有两种形式，通过数组 / embedded document
array 的基本形式为
<field> : [<longitude>, <latitude>]
embedded document 的基本形式为
<field> : {<field1> : <longitude>, <field2> : <latitude>}
注意必须 <longitude> 在前，<latitude> 在后
<logitude> 取值范围为 [-180, 180]
<latitude> 取值范围 为 [-90, 90]
geohash 有两种形式 : 2dsphere / 2d
2dsphere : calculate geometries on earth-like sphere
2d : calculate geometries on two-dimentional plane
注意 geospatial index 不可用于 shard key when sharding collection
但是依旧可以在 shardedcollection 中使用 geospatial index

Database - P70

Mongo DB

Geospatial query：通过 db.collection.createIndex() 函数创建索引

形式为 db.collection.createIndex({<location field>: "2dsphere"})

db.collection.createIndex({<location field>: "2d",
<additional field>: <value>},
{<index-specification options>})

<index-specification options> 为可选的选项

{min: <lower bound>, max: <upper bound>, bit: <bit precision>}

在 db.collection.find() 函数中可以使用 $geoWithin 运算符

基本形式为 db.collection.find({
<location field>: {$geoWithin:
{<operator>: <coordinate>}})

可用的 operator 选择有

$box: [<bottom left>, <upper right>]

用于定义一个方形区域

$polygon: [[<x1>,<y1>], [<x2>,<y2>], ..., [<xn>,<yn>]]

用于定义一个多边形区域

$center: [[<x>,<y>], <radius>]

用于定义以(<x>,<y>)为圆心，<radius>为半径的圆形区域

$centerSphere：与$center类似，但必须使用"2dsphere"

[redacted] 返回距离(<x>,<y>)最近的点，("2d")

{$near: [<x>,<y>], $maxDistance: <distance in radians>}

在 db.collection.aggregate() 函数中可以使用 $geoNear 运算符

基本形式为 db.collection.aggregate([
{$geoNear: {
near: <point>,
distanceField: <location field>,
[limit: <integer>] (default: 100)
[num: <integer>,] (override num)
[maxDistance: <distance in radians>,]
[minDistance: <distance in radians>,]}})

output documents in order of nearest to farthest from specified point