概述：node作为一门后端语言，肯定少不了与数据库打交道，与node最合拍的数据库无非为MongoDB，除此之外，也可搭配mysql使用。

1 数据库

1.1 **简介：**数据库（Database）是按照数据结构来组织、存储和管理数据的仓库。

1.2 **数据库类型：** 层次式数据库、网络式数据库和关系式数据库，关系式又细分为关系型与非关系型数据库。

1.3 RDBMS(Relational Database Management System)：关系型数据库

- 数据以表格的形式出现
- 每行为各种记录名称
- 每列为记录名称所对应的数据域
- 许多的行和列组成一张表单
- 若干的表单组成database
- 数据库 > 表 > 列 > 行 > 值

1.4 NoSQL(No Only SQL)：非关系型数据库

数据库 -> collection -> document -> 字段

1.5 SQL： 所有关系型数据库的公共语言

1.6 RDBMS vs NoSQL

SQL	NoSQL
实时一致性	简单便捷
事务	方便扩展
多表联合查询	更好的性能

2 mysql

2.1 **简介：**最流行的关系型数据库，由瑞典MySQL AB公司开发，目前属于Oracle公司。MySQL是一种关联数据库管理系统，关联数据库将数据保存在不同的表中，而不是将所有数

据放在一个大仓库内，这样就增加了速度并提高了灵活性。

2.2：特点

- 无数据结构限制
- 完全的索引支持
- 方便的冗余与扩展
- 良好的支持

2.2 安装：本机配好mysql环境后，使用npm安装模块

```
$ npm install mysql
```

2.3 配置连接：安装模块后我们可以在自己的文件引入该模块，使用*mysql.createConnection()*来配置与mysql的连接

```
var mysql = require('mysql');

var conn = mysql.createConnection({
   user: 'root',
   password: '444666',
   database: 'nodejs'
});
```

参数	**描述**
host	主机地址 （默认：localhost）
user	用户名
password	密码
port	端口号 （默认：3306）
database	数据库名
charset	连接字符集（默认：'UTF8_GENERAL_CI'，注意字符集的字母都要大写）
localAddress	此IP用于TCP连接（可选）
socketPath	连接到unix域路径，当使用 host 和 port 时会被忽略
timezone	时区（默认：'local'）
connectTimeout	连接超时（默认：不限制；单位：毫秒）

stringifyObjects	是否序列化对象
typeCast	是否将列值转化为本地JavaScript类型值 （默认：true）
queryFormat	自定义query语句格式化方法
supportBigNumbers	数据库支持bigint或decimal类型列时，需要设此option为true （默认：f
bigNumberStrings	supportBigNumbers和bigNumberStrings启用 强制bigint或decimal列串类型返回（默认：false）
dateStrings	强制timestamp,datetime,data类型以字符串类型返回，而不是JavaScrip认：false）
debug	开启调试（默认：false）
multipleStatements	是否许一个query中有多个MySQL语句 （默认：false）
flags	用于修改连接标志
ssl	使用ssl参数（与crypto.createCredenitals参数格式一至）或一个件名称的字符串，目前只捆绑Amazon RDS的配置文件

2.4 **连接数据库**：配置好参数后别忘了建立连接

```
conn.connect();
```

2.5 **操作数据库（CRUD）**：关系型数据库CRUD的实现是由SQL语句实现的，mysql模块提供了一个.*query(sqlString, callback)*的方法来执行SQL语句;callback函数会在执行完语句后调用，该函数的三个参数依次是:*err*,*results*,*fields*;
result包含执行结果，fields：将包含有关返回结果字段的信息（如果有的话）

```
conn.query('SELECT * FROM t_user', function (err,results,fields) {
  if (err) throw err;
  console.log(results);
});
```

2.6 连接池

2.7**关闭数据库**：因为数据库的连接是有限制的，所以操作完数据库要关闭数据库

```
conn.end();
```

3 MongoDB

3.1 简介

MongoDB是一个面向文档的数据库系统。使用C++编写，不支持SQL，但有自己功能强大的查询语法。

mongodb使用BSON作为数据存储和传输的格式。BSON是一种类似JSON的二进制序列化文档，支持嵌套对象和数组。

3.2 配置

MongoDB配置

环境变量：MongoDB是通过命令行使用的，为了在任意目录下使用，可以配置环境变量

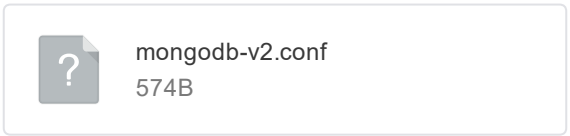
```
数据库路径 data
mongodb日志 mongod.log
配置文件 mongodb.conf
```

数据，日志目录配置、启动服务器：

```
$ mongod --dbpath=D:\MongoDB\data\db  --directoryperdb    --
logpath=D:\MongoDB\data\logs\mongo.log --logappend
```

启动客户端： MongoDB Shell是MongoDB自带的交互式Javascript shell,用来对MongoDB进行操作和管理的交互式环境。

```
$ mongo
```



以配置文件方式启动MongoDB服务

```
$ mongod --config D:\MongoDB\mongodb.cfg
```

配置window服务：每次都要用命令行启动，太麻烦了，在windows下可创建一个服务

```
$ mongod --config D:\MongoDB\mongodb.cfg --serviceName "mongod" -install
```

linux配置服务

```
$  sudo  service  mongodb  start
```

启动服务

```
$ net start mogod
```

关闭mongodb服务

mongo

$ use admin;

$ db.shutdownServer();

3.3 概念解析

SQL术语/概念	MongoDB术语/概念	解释/说明
database	database	数据库
table	collection	数据库表/集合
row	document	数据记录行/文档
column	field	数据字段/域
index	index	索引
table joins		表连接,MongoDB不支持
primary key	primary key	主键,MongoDB自动将_id字段设置为主键

数据库：一个mongodb可以建立多个数据库，每个数据库都是独立的，都有自己的集合和权限。

Mongodb对数据库的名字有一些要求：

- UTF-8字符串
- 不能是空字符串（""）。
- 不得含有' '（空格)、.、$、/、\和\0 (空字符)。
- 应全部小写。

- 最多64字节。

有几个保留的数据库名：
- admin：root数据库
- local：本地数据库
- config： 当Mongo用于分片设置时，config数据库在内部使用，用于保存分片的相关信息。

show dbs 命令可以查看所有数据库

db 命令可以像是当前数据库对象或集合

use *database* 连接到一个指定的数据库

集合：MongoDB文档组，类似于RDBMS中的表格，集合没有固定的结构，但MongoDB对命名做了一个规范
- 不能是空字符串""。
- 不能含有\0字符（空字符），这个字符表示集合名的结尾。
- 不能以"system."开头，这是为系统集合保留的前缀。
- 不能含有保留字

show collections 显示已选中的数据库集合

db.createCollection(*colname*) 创建一个集合

db.*collection*.find() 查找集合的所有文档

db.*collection*.find().count() 查找集合的文档数量

db.*collection*.find().skit(*n*) 跳过前n个文档

db.collection.find().limit(*n*) 显示n个文档

capped collections：固定大小的collection

文档：文档是一组键值(key-value)对，即(BSON)。 MongoDB 的文档不需要设置相同的字段，并且相同的字段不需要相同的数据类型，然而文档也有一些需要遵循的规范

- 文档中的键/值对是有序的。
- 文档中的值不仅可以是在双引号里面的字符串，还可以是其他几种数据类型（甚至可以是整个嵌入的文档)。
- MongoDB区分类型和大小写。
- MongoDB的文档不能有重复的键。
- 文档的键是字符串。除了少数例外情况，键可以使用任意UTF-8字符。

文档键命名规范：

- 键不能含有\0 (空字符)。这个字符用来表示键的结尾。
- .和$有特别的意义，只有在特定环境下才能使用。
- 以下划线"_"开头的键是保留的(不是严格要求的)。

db.*collection*.insert(*document*) 向指定的集合插入文档（如果该集合不存在，将自动创建该集合）

db.*collection*.find() 查找集合数据

元数据

数据类型： 下表为MongoDB中常用的几种数据类型。

数据类型	描述
String	字符串。存储数据常用的数据类型。在 MongoDB 中，UTF-8 编码的字符串才是合法的。
Integer	整型数值。用于存储数值。根据你所采用的服务器，可分为 32 位或 64 位。
Boolean	布尔值。用于存储布尔值（真/假）。
Double	双精度浮点值。用于存储浮点值。
Min/Max keys	将一个值与 BSON（二进制的 JSON）元素的最低值和最高值相对比。
Arrays	用于将数组或列表或多个值存储为一个键。
Timestamp	时间戳。记录文档修改或添加的具体时间。
Object	用于内嵌文档。
Null	用于创建空值。
Symbol	符号。该数据类型基本上等同于字符串类型，但不同的是，它一般用于采用特殊符号类型的语言。
Date	日期时间。用 UNIX 时间格式来存储当前日期或时间。你可以指定自己的日期时间：创建 Date 对象，传入年月日信息。
Object ID	对象 ID。用于创建文档的 ID。
Binary Data	二进制数据。用于存储二进制数据。
Code	代码类型。用于在文档中存储 JavaScript 代码。
Regular expression	正则表达式类型。用于存储正则表达式。

3.4 数据库操作

```
# 创建数据库：MongoDB并不会创建一个空的数据库，只有当有文档字段时，数据库才会被隐式创建

# 删除选中的数据库
db.dropDatabase()

#创建集合
db.createCollection(name)

# 删除集合
```

```
db.collection.drop()

# 插入文档
db.collection.insert(document)
# 插入一条
db.collection.insertOne()
# 插入多条
db.collection.insertMany()

# 更新文档
db.collection.update()

# 替换已有文档
db.collection.save()

# 删除文档
db.collections.remove()

# 查询文档
db.collection.find()

# 美化查询
db.collection.find().pretty()
```

3.5：条件操作符

```
$gt  --------  greater  than   >
$gte  ---------  gt  equal   >=
$lt  --------  less  than   <
$lte  ---------  lt  equal   <=
$ne  -----------  not  equal   !=
$eq   --------   equal   =
```

```
#获取"col"集合中 "likes" 小于等于 150 的数据
$ db.col.find({likes : {$lte : 150}})
```

```
#类似于SQL语句：
Select * from col where likes <= 150;
```

3.6： $type操作符

基于BSON类型来检索集合中匹配的数据类型，并返回结果。

类型	数字	备注
Double	1	
String	2	
Object	3	
Array	4	
Binary data	5	
Undefined	6	已废弃。
Object id	7	
Boolean	8	
Date	9	
Null	10	
Regular Expression	11	
JavaScript	13	
Symbol	14	
JavaScript (with scope)	15	
32-bit integer	16	
Timestamp	17	
64-bit integer	18	
Min key	255	Query with -1.
Max key	127	

3.7：索引

索引通常能够极大的提高查询的效率，如果没有索引，MongoDB在读取数据时必须扫描集合中的每个文件并选取那些符合查询条件的记录。

索引分类：
- _id索引
- 单键索引
- 多键索引
- 复合索引
- 过期索引
- 全文索引
- 地理位置索引

db.*colname*.getIndexes()　　查询索引
db.*colname*.ensureIndex()　创建索引

createIndex

3.8 备份(mongodump)与恢复(mongorestore)

在Mongodb中我们使用mongodump命令来备份MongoDB数据。该命令可以导出所有数据到指定目录中。

```
>mongodump -h dbhost -d dbname -o dbdirectory
```

- -h：mongDB所在服务器地址，例如：127.0.0.1，当然也可以指定端口号：127.0.0.1:27017

- -d：需要备份的数据库实例，默认是除local之外的所有数据库

- -o：备份的数据存放位置，例如：c:\data，在备份完成后，系统建立一个dump目录，这个目录里面存放该数据库实例的备份数据，默认是命令行路径下建立dump

mongodb使用 mongorestore 命令来恢复备份的数据。

```
>mongorestore -h <hostname><:port> -d dbname <path>
```

- --host <:port>, -h <:port>：

MongoDB所在服务器地址，默认为： localhost:27017

- --db , -d ：
需要恢复的数据库实例，例如：test，当然这个名称也可以和备份时候的不一样，比如test2

- --drop：
恢复的时候，先删除当前数据，然后恢复备份的数据。就是说，恢复后，备份后添加修改的数据都会被删除，慎用哦！

- <path>：
mongorestore 最后的一个参数，设置备份数据所在位置，例如：c:\data\dump\test。
你不能同时指定 <path> 和 --dir 选项，--dir也可以设置备份目录。

- --dir：
指定备份的目录
你不能同时指定 <path> 和 --dir 选项。

3.9：安全

Mongodb默认并未设置任何安全限制，但是作为一个数据库，可以通过以下方式提高安全级别
- 物理隔离与网络隔离
- IP白名单隔离
- 用户名密码授权

1： 物理隔离=>不现实
2： 网络隔离=>内网
3： 防火墙隔离=>限定IP
4： 用户名密码=>权限设置

默认情况下Mongodb并没有为自己开启权限认证

开启方法：- auth

开启权限后需要用户名密码方可使用数据库

```
#创建一个用户
$ db.createUser({user:"admin",pwd:"123",roles:[{role:"useAdmin",db:"admin"}]})
#登录
$ mongo -u username -p password
```

用户角色分类
数据库角色(read, readWrite, dbAdmin, dbOwer, userAdmin)
集群角色(clusterAdmin, clusterManager...)
备份角色(backup, restore...)
其他特殊权限(DBAdminAnyDatabase ...)

3.10：使用开发语言访问MongoDB
MongoDB支持Perl、PHP、Java、C#、JavaScript、Ruby、C 和C++语言的驱动程序，MongoDB提供了当前所有主流开发语言的数据库驱动包，开发人员使用任何一种主流开发语言都可以轻松编程，实现访问MongoDB 数据库。

node中的mongodb

node安装mongodb模块

```
npm install mongodb --save
```

数据库基本使用流程
- 配置连接
- 连接数据库
- 操作数据(CRUD)
- 关闭数据库

标准 URI 连接语法

```
mongodb://[username:password@]host1[:port1][,host2[:port2],...[,hostN[:portN]]][/[database][?options]]
```

- **mongodb:// 这是固定的格式，必须要指定。**
- **username:password@ 可选项，如果设置，在连接数据库服务器之后，驱动都会尝试登陆这个数据库**
- **host1 必须的指定至少一个host, host1 是这个URI唯一要填写的。它指定了要连接服务器的地址。如果要连接复制集，请指定多个主机地址。**
- **portX 可选的指定端口，如果不填，默认为27017**
- **/database 如果指定username:password@，连接并验证登陆指定数据库。若不指定，默认打开 test 数据库。**
- **?options 是连接选项。如果不使用/database，则前面需要加上/。所有连接选项都是键值对name=value，键值对之间通过&或;（分号）隔开**

连接Mongdb数据库实例

```
//  1.创建一个Mongo客户端
var  MongoClient  = require('mongodb').MongoClient;

// 2.配置连接地址
var url = 'mongodb://localhost:27017/' + dbname;

// 3.建立链接
MongoClient.connect(url, function (err, db) {
   if(err) throw err;
   console.log('连接成功');
   // 4. 关闭数据库
   db.colse();
})
```

查询数据：

```
// 1.声明查询函数
var findDocuments = function(db, callback) {
   // 2.获取集合
   var collection = db.collection(colname);
   // 3.在指定集合查询
   collection.find({}).toArray(function (err, docs) {
```

```
        if(err) throwww err;
        console.log(docs);
        callback();
    })
}

MongoClient.connect(url, function (err, db) {
    if(err) throw err;
    console.log('连接成功');
    // 4.在连接的回掉函数中调用查询方法
    findDocuments(db, function () {
        db.close();
    })
})
```

聚合：MongoDB中聚合(aggregate)主要用于处理数据(诸如统计平均值,求和等)，并返回计算后的数据结果。有点类似sql语句中的 count(*)。

复制：MongoDB复制是将数据同步在多个服务器的过程，复制提供了数据的冗余备份，并在多个服务器上存储数据副本，提高了数据的可用性， 并可以保证数据的安全性。

管道： MongoDB 的聚合管道将 MongoDB 文档在一个管道处理完毕后将结果传递给下一个管道处理。管道操作是可以重复的。