Big Data - NoSQL

重要程度：⭐
难易程度：⭐⭐

- Document
  - Each item is stored as a document (normally BSON or JSON document, but could be XML)
  - Note the variable structure and embedded documents

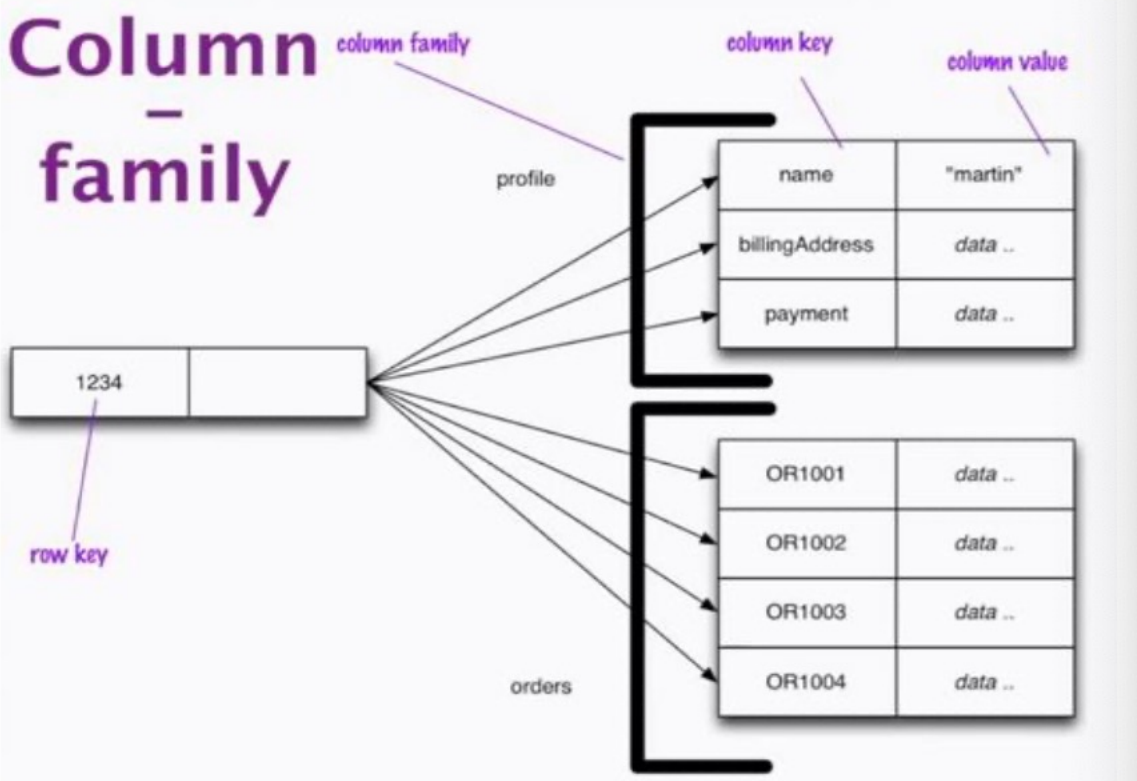
```
{  maker: "M.V. Agusta",
   type: "sportsbike",
   engine: {
      type: "internal combustion",
      cylinders: 4,
      displacement: 750
    },
   rake:7,
   trail:3.93
}

{   maker : "M.V. Agusta",
    type : "Helicopter",
    engine : {
        type : "turboshaft",
        layout : "axial",
        massflow : 1318
    },
    Blades : 4,
    undercarriage : "fixed"
}
```

Big Data - NoSQL

重要程度：⭐
难易程度：⭐⭐

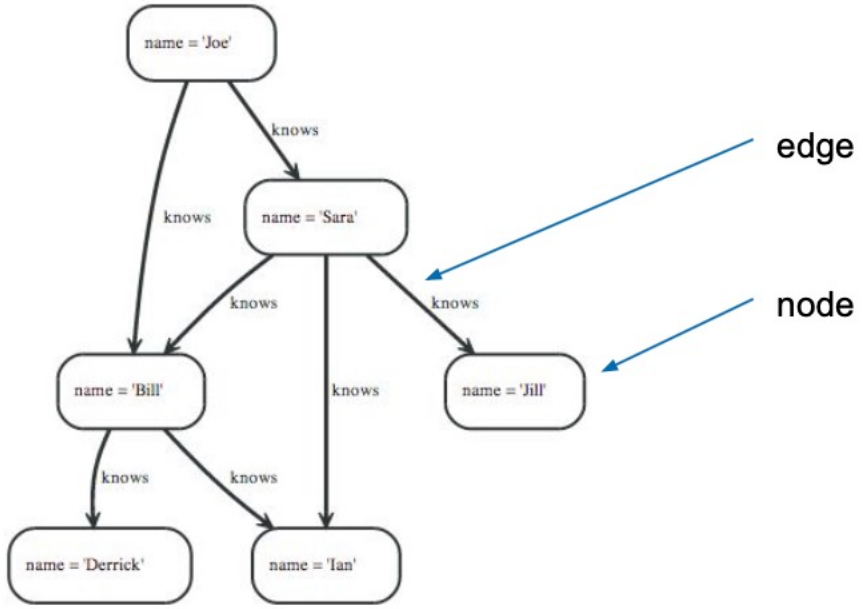
- Column Family (also called Wide Column Store)
  - Key points to a set of multiple column values containing related data arranged by column family



Big Data - NoSQL

重要程度：⭐
难易程度：⭐⭐

- Graph - based on a graph structure
  - Unlike the previous three which are aggregation oriented, the graph model views data at a highly non aggregated level
  - Based on graph theory
  - Navigate via relationships (edges) between nodes
  - Examples
    - Neo4j
    - HyperGraphDB



To find out the friends of Joe's friends that are not already his friends, the query looks like this:

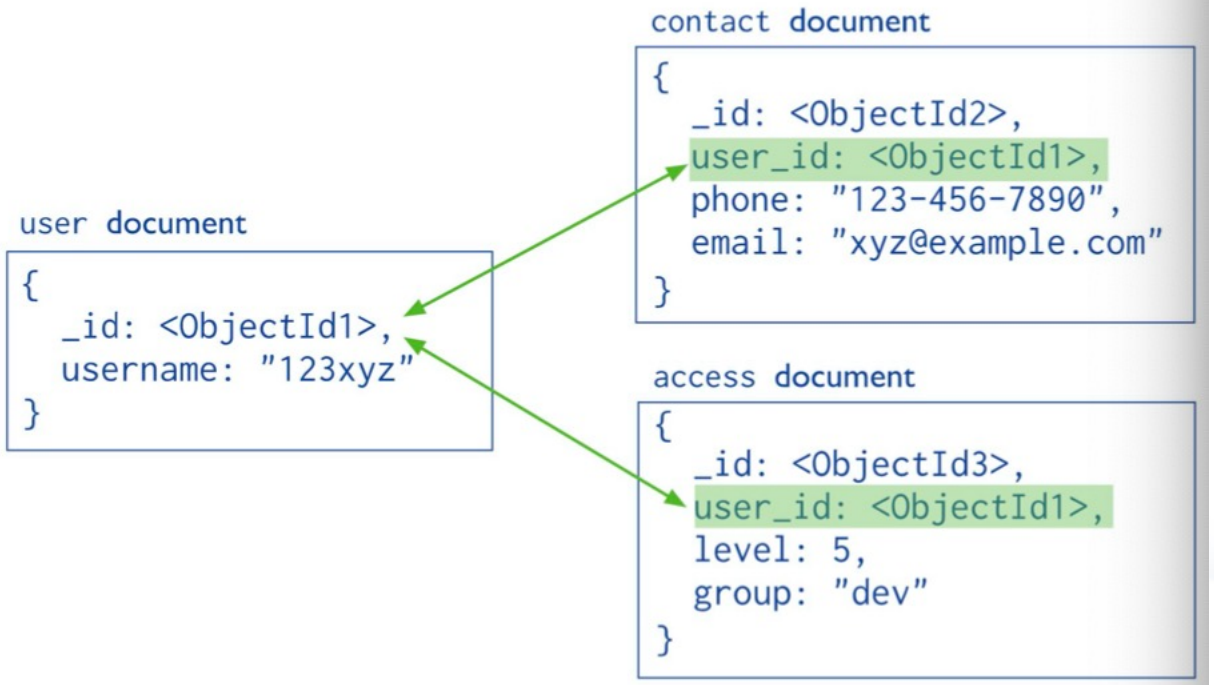
Query.

```
START joe=node:node_auto_index(name = "Joe")
MATCH joe-[:knows*2..2]-friend_of_friend
WHERE not(joe-[:knows]-friend_of_friend)
RETURN friend_of_friend.name, COUNT(*)
ORDER BY COUNT(*) DESC, friend_of_friend.name
```

Big Data - documents

重要程度：⭐
难易程度：⭐⭐

Denormalised – Embedded Documents



Normalised – References

Big Data - documents

重要程度：⭐
难易程度：⭐⭐

```
{
“_id”: 1,
“name” : {“first”: “John”, “last”: “Wang”},
“units”: [“FIT2094”, “FIT1008”, “FIT3138”],
”skills”: [
        {
                “skill”: “Python”,
                “level”: 1
        },
        {
                “skill”:”Java”,
                “level”: 3
        }
                ]
}
```

1. Simple
2. Composite
3. Multi-valued
4. String vs number vs boolean

Big Data - MongoDB

重要程度：⭐
难易程度：⭐⭐

1. Document - based (需要熟悉JSON 格式)
2. NoSQL
3. CRUD (CREATE, RETRIEVE, UPDATE, DELETE)

Big Data – MongoDB - CREATE

重要程度：⭐
难易程度：⭐⭐

InsertOne: 只insert一条记录
insertMany: insert多条记录（要用array）

- create collection by inserting documents
  - db.collection.insertOne (….. JSON …..);
  - db.collection.insertMany - insert an array of JSON documents
    - insertMany ([JSON1, JSON2, …]);

```
> db.famous.insertOne ({
...    "_id": 1,
...    "name" : { "first" : "Edward", "last" : "Codd" },
...    "contribs" : [ "Database Relational Model", "Database Technology", "OLAP"
 ],
...    "awards" : [
...      {
...        "award" : "Turing Award",
...        "year" : 1981,
...        "by" : "Association for Computing Machinery"
...      }
...    ]
... });
{ "acknowledged" : true, "insertedId" : 1 }
>
```

Big Data – MongoDB - RETRIEVE

重要程度：⭐
难易程度：⭐⭐

Documents retrieved by find method on collection

- db.famous.find ({}); or db.famous.find ({}).pretty()
  - find all

```
> show collections
famous
> db.famous.find ({});
{ "_id" : 1, "name" : { "first" : "Edward", "last" : "Codd" }, "contribs" : [ "D
atabase Relational Model", "Database Technology", "OLAP" ], "awards" : [ { "awar
d" : "Turing Award", "year" : 1981, "by" : "Association for Computing Machinery"
 } ] }
{ "_id" : 2, "name" : { "first" : "Chris", "last" : "Date" }, "contribs" : [ "Da
tabase Relational Model" ], "awards" : [ ] }
{ "_id" : 3, "name" : { "first" : "Peter", "last" : "Chen" }, "contribs" : [ "En
tity Relationship Modelling", "CASE", "IoT" ], "awards" : [ { "award" : "AAAI Al
len Newell Award", "year" : 1981, "by" : "Association for Computing Machinery" }
, { "award" : "Harry H. Goode Memorial Award", "year" : 2003, "by" : "IEEE Compu
ter Society" } ] }
{ "_id" : 4, "name" : { "first" : "Michael", "last" : "Stonebraker" }, "contribs
" : [ "Database Technology", "OLAP", "Ingres", "postgres" ], "awards" : [ { "awa
rd" : "Edgar F. Codd Innovations Award", "year" : 1992, "by" : "Association for
Computing Machinery" }, { "award" : "John von Neumann Medal", "year" : 2005, "by
" : "IEEE Computer Society" }, { "award" : "Turing Award", "year" : 2014, "by" :
 "Association for Computing Machinery" } ] }
```

$and:
db.famous.find({$and:[{cond1},{cond2}..})

$or:
db.famous.find({$or:[{cond1},{cond2}..})

.count()
db.famous.find().count()

.sum()
db.famous.find().sum()

.pretty()
db.famous.find().pretty()

Big Data – MongoDB - RETRIEVE

重要程度：⭐
难易程度：⭐⭐

```
- db.famous.find ({_id: 2});
- db.famous.find ({contribs: "Database Technology"})
- db.famous.find ({contribs: /.*Database.*/})
- db.famous.find ({ "name.first": "Peter"})
- db.famous.find ({"awards.by": "IEEE Computer Society" })
- db.famous.find ({"awards.by": "IEEE Computer Society" }).count()
```

Big Data – MongoDB - UPDATE

重要程度：⭐
难易程度：⭐⭐

updateOne: 只update一条记录
updateMany: update多条记录

Db.famous.updateOne({condition},{$set:output})

If update array：”enrolment.$.unit”:”FIT1008” 符合条件的情况下，只更新第一个element的值

$push: 更新array信息 {$push: {”skills":{"code":”python", ”level:4} }

- db.famous.updateOne ({"_id": 2},{ $set: {"name.first": "Christopher"}})

```
> db.famous.findOne({_id:2},{"name.first": 1,"name.last": 1})
{ "_id" : 2, "name" : { "first" : "Chris", "last" : "Date" } }
> db.famous.updateOne ( {"_id": 2},{ $set: {"name.first": "Christopher"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.famous.findOne({_id:2},{"name.first": 1,"name.last": 1})
{ "_id" : 2, "name" : { "first" : "Christopher", "last" : "Date" } }
```

Big Data – MongoDB - DELETE

重要程度：⭐
难易程度：⭐⭐

deleteOne: 只delete一条记录
deleteMany: delete多条记录

Db.famous.deleteOne({condition})