MongoDB Introduction

Ali Jaafar
SMB214

ali.jaafar86@gmail.com
01-01-2016

MongoDB Introduction

Ali Jaafar
SMB214

ali.jaafar86@gmail.com

01-01-2016

Agenda

- *What is MongoDB*
- *why MongoDB*
- *MongoDB Data Model*
- *MongoDB Deployment*
- *MongoDB Query language*
- *Built-In MapReduce*
- *Aggregation*
- *Demo*

What is MongoDB

MongoDB (from "humongous") is a scalable, high-performance, open source, document-oriented database

MongoDB is classified as NoSQL Database

Why MongoDB

MongoDB focuses on four main things:

- Flexibility
- Power
- Speed
- Easy of Use

Flexibility

- Data stored in JSON documents (serialized to BSON)
- Schema-less
- Maps to native programming languages
- ERDs are not governing the design (like RDBMS)

Power

- Supports secondary indexes
- Dynamic Queries
- Sorting
- Rich Updates
- Easy Aggregations
- "Upserts" --update if document exists, insert if it doesn't

Speed/Scaling

- Related data kept together in documents
- No need for joining various tables
- Auto-shardingallow for scaling clusters linearly
- Can increase capacity with "No Downtime"

Ease of Use

- Very easy to install, configure, maintain, and use
- Very few configuration options
- Works right out of the box
- No need for fine-tuning obscure database configurations

MongoDB Query

Flexibility

- Data stored in JSON documents (serialized to BSON)
- Schema-less
- Maps to native programming languages
- ERDs are not governing the design (like RDBMS)

Power

- Supports secondary indexes
- Dynamic Queries
- Sorting
- Rich Updates
- Easy Aggregations
- “Upserts” –update if document exists, insert if it doesn’t

Speed/Scaling

- Related data kept together in documents
- No need for joining various tables
- Auto-shardingallow for scaling clusters linearly
- Can increase capacity with “No Downtime”

Ease of Use

- Very easy to install, configure, maintain, and use
- Very few configuration options
- Works right out of the box
- No need for fine-tuning obscure database configurations

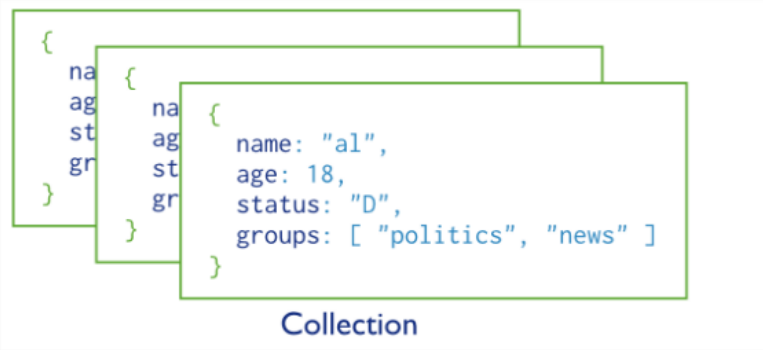
Data Model

MongoDB documents are BSON documents. BSON is a binary representation of JSON with additional type information.

In the documents, the value of a field can be any of the BSON data types, including other documents, arrays, and arrays of documents

```
{
  name: "sue",                          ← field: value
  age: 26,                              ← field: value
  status: "A",                          ← field: value
  groups: [ "news", "sports" ]          ← field: value
```

MongoDB stores all documents in collections. A collection is a group of related documents that have a set of shared common indexes



Data Model

A database holds a set of collections (Mongo DB ~=SQL DB)

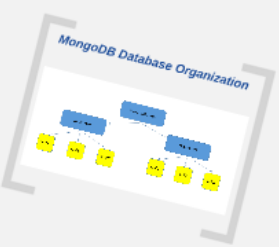
A collection holds a set of documents (Mongo Collection ~= SQL Table)

A document is a set of fields (Mongo Documents~= SQL Record)

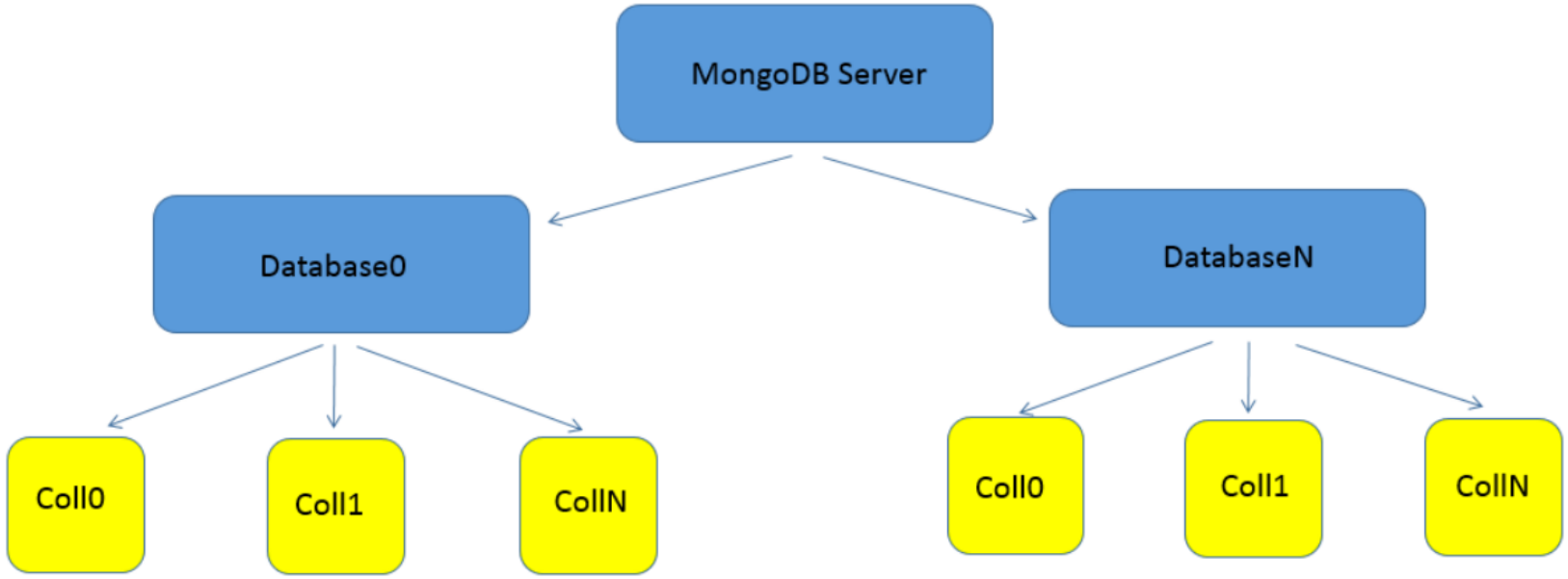
A field is a key-value pair (Mongo Field~=SQl Atttribute)

A key is a name , a value can be :

- a data type like string,integer,float,binary...
- a document
- an array of Values



MongoDB Database Organization



Example Tabular Structure

Person

ID	FIRST_NAME	LAST_NAME
1	Ali	Jaafar

Account

ID	ACCOUNT_TYPE	ACCOUNT_BALANCE	CURRENCY	(FK:Person)
1	Investment	80000.00	USD	1
2	Savings	70400.00	LBP	1

Example Document Oriented Structure

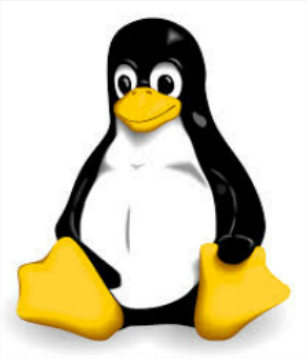
Person Collection (Person is the root entity)


```
[{
    "ID": 1,
    "first_name": "Ali",
    "last_name": "Jaafar",
    "accounts": [

        {
            "id": 1,
            "account_type": "Savings",
            "account_balance": "70400.00",
            "currency": "USD"
        },
        {
            "id": 2,
            "account_type": "Checking",
            "account_balance": "80000.00",
            "currency": "LBP"
        }
    ]
}]
```

MongoDB Deployment

MongoDB runs on most platforms and supports both 32-bit and 64-bit architectures.



Deployment on windows

- Download the Zip File 32-Bit or 64-Bit (recommended)
- Unzip the Download
- Create a Data Directory

```
C:\> mkdir \data
C:\> mkdir \data\db
```

Run & Connect to the Server

```
C:\> cd \my_mongo_dir\bin
C:\my_mongo_dir\bin> mongod
```

MongoDB Query language

MySQL term	Mongo term/concept
database	database
table	collection
index	index
row	BSON document
column	BSON field
join	embedding and linking
primary key	_id field
group by	aggregation

MongoDB Query language

`CREATE TABLE USERS (a Number, b Number)`	implicit; can also be done explicitly with `db.createCollection("mycoll")`
`ALTER TABLE users ADD ...`	implicit
`INSERT INTO USERS VALUES(3,5)`	`db.users.insert({a:3,b:5})`
`CREATE INDEX myindexname ON users(name)`	`db.users.ensureIndex({name:1})`
`CREATE INDEX myindexname ON users(name,ts DESC)`	`db.users.ensureIndex({name:1,ts:-1})`

MongoDB Query language

SELECT a,b FROM users	db.users.find({}, {a:1,b:1})
SELECT * FROM users	db.users.find()
SELECT * FROM users WHERE age=33	db.users.find({age:33})
SELECT a,b FROM users WHERE age=33	db.users.find({age:33}, {a:1,b:1})
SELECT * FROM users WHERE age=33 ORDER BY name	db.users.find({age:33}).sort({name:1})

MongoDB Query language

`EXPLAIN SELECT * FROM users WHERE z=3`	`db.users.find({z:3}).explain()`
`UPDATE users SET a=1 WHERE b='q'`	`db.users.update({b:'q'}, {$set:{a:1}}, false, true)`
`UPDATE users SET a=a+2 WHERE b='q'`	`db.users.update({b:'q'}, {$inc:{a:2}}, false, true)`
`DELETE FROM users WHERE z="abc"`	`db.users.remove({z:'abc'});`

Built-In MapReduce

Data processing paradigm for condensing large volumes of data into useful aggregated results

```
db.orders.mapReduce(
    function() { emit( this.cust_id, this.amount ); },
    function(key, values) { return Array.sum( values ) },
    {
      query: { status: "A" },
      out: "order_totals"
    }
  )
```

Collection

map

reduce

query

output

Built-In MapReduce

Map Reduce is the process of processing the entire database with 2 steps Map and Reduce

Map Phase–processes each document and emits one or more objects for each document

Reduce Phase–combines the output of the map operation

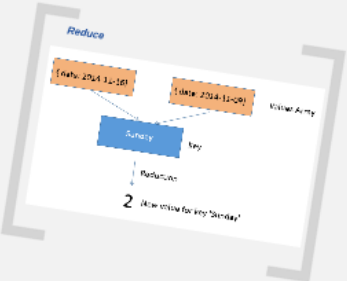
Finalize (optional) –used to make final modifications to the output

Real Example

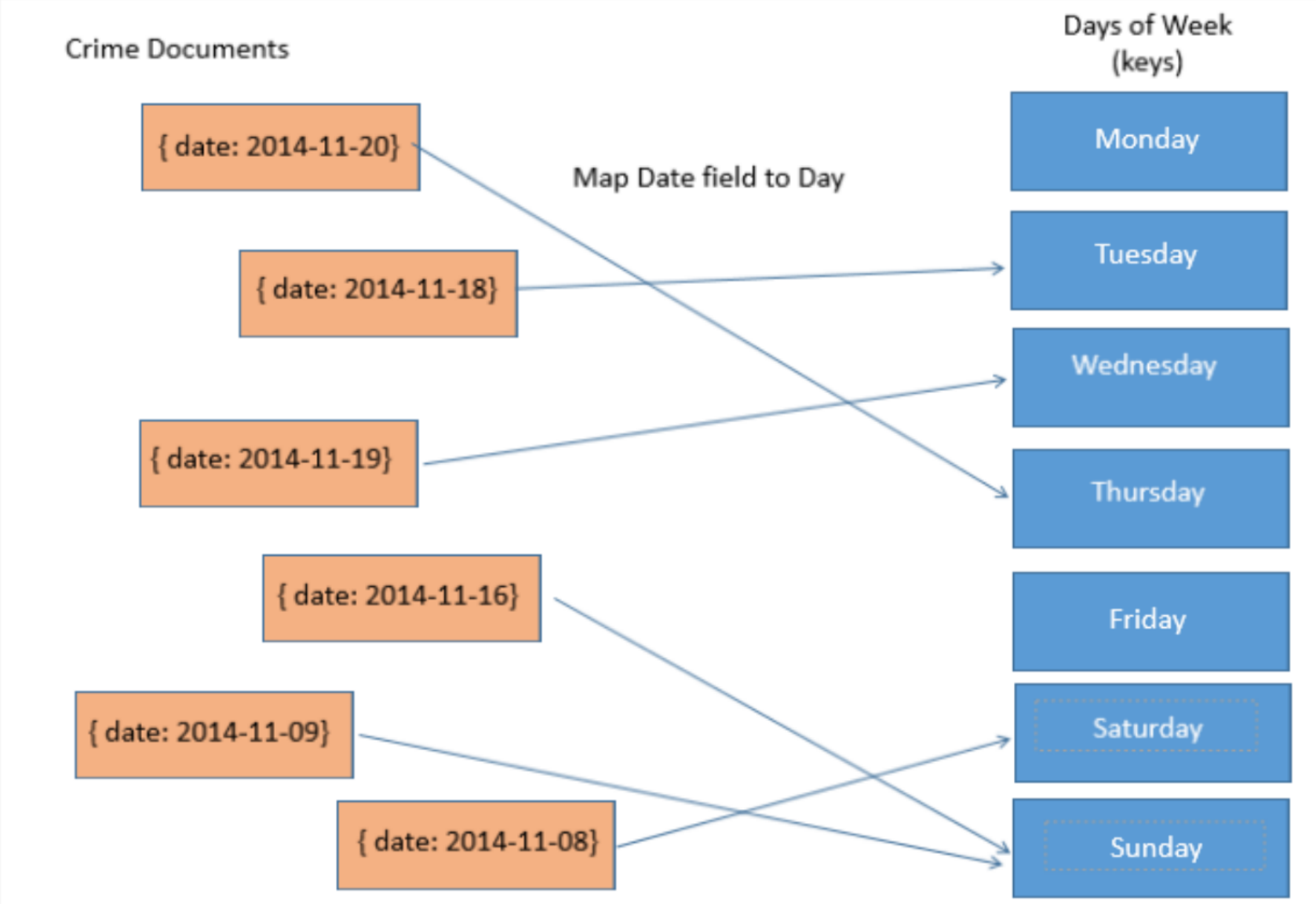
The example data set we will be dealing with is City of Chicago crime police report data from November 2014 to December 2015.

{ "_id" : ObjectId("5462725476ecd357dbbc721e"), "ID" : 9844675, "Case Number" : "HX494115", "Date" : "11/03/2014 11:51:00 PM", "Block" : "056XX S MORGAN ST", "IUCR" : 486, "Primary Type" : "BATTERY", "Description" : "DOMESTIC BATTERY SIMPLE", "Location Description" : "ALLEY", "Arrest" : "false", "Domestic" : "true", "Beat" : 712, "District" : 7, "Ward" : 16, "Community Area" : 68, "FBI Code" : "08B", "X Coordinate" : 1170654, "Y Coordinate" : 1867165, "Year" : 2014, "Updated On" : "11/10/2014 12:43:02 PM", "Latitude" : 41.790980835, "Longitude" : -87.649786614, "Location" : "(41.790980835, -87.649786614)" }

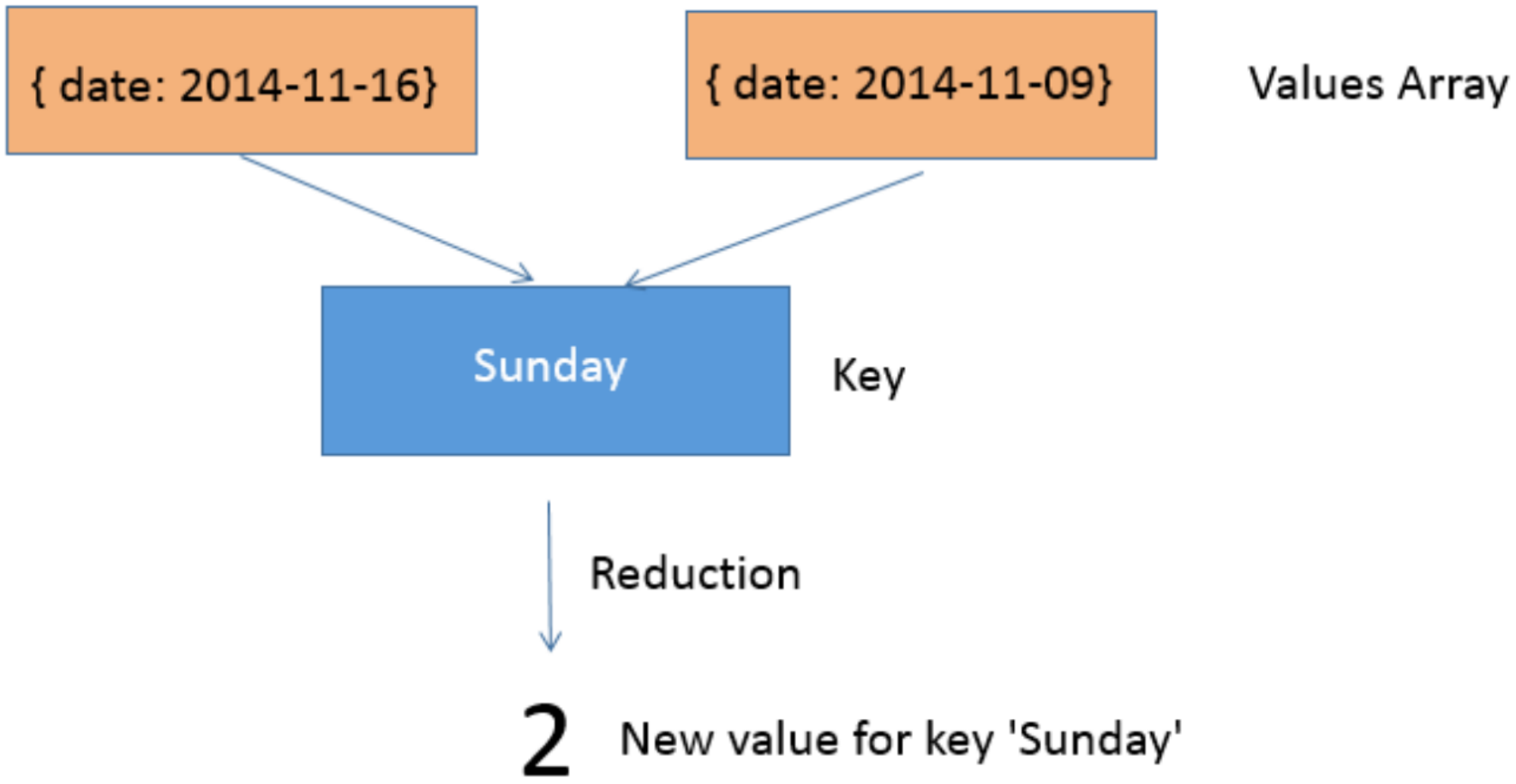
What day of the week has the most crime incidents recorded in Chicago?



Map


Crime Documents
Days of Week
(keys)
{ date: 2014-11-20}
Map Date field to Day
Monday
{ date: 2014-11-18}
Tuesday
Wednesday
{ date: 2014-11-19}
Thursday
{ date: 2014-11-16}
Friday
{ date: 2014-11-09}
Saturday
{ date: 2014-11-08}
Sunday

Reduce



Code Mongo shell

```
//map function
var map=function ()
{
    var milis = Date.parse(this.Date);
    var date = new Date(milis);
    var daysOfWeek = ["Sunday", "Monday","Tuesday","Wednesday","Thursday","Friday",  "Saturday"];
    //emit the mapping between this document and the Day which it cooresponds to.
    emit(daysOfWeek[date.getDay()], 1);
}

// reduce function
var reduce= function (key, values){  return   Array.sum(values);  }

// result
var result = db.runCommand
(
 {  "mapreduce" : "crimes",
     "map" : map,
     "reduce" : reduce,
     "out" : "crimesfreq"
  }
)
```

Results

Number of crimes based on each day of the week

{ "_id" : "Friday", "value" : 44449 }
{ "_id" : "Monday", "value" : 43027 }
{ "_id" : "Saturday", "value" : 44305 }
{ "_id" : "Sunday", "value" : 42866 }
{ "_id" : "Thursday", "value" : 41974 }
{ "_id" : "Tuesday", "value" : 42071 }
{ "_id" : "Wednesday", "value" : 42569 }

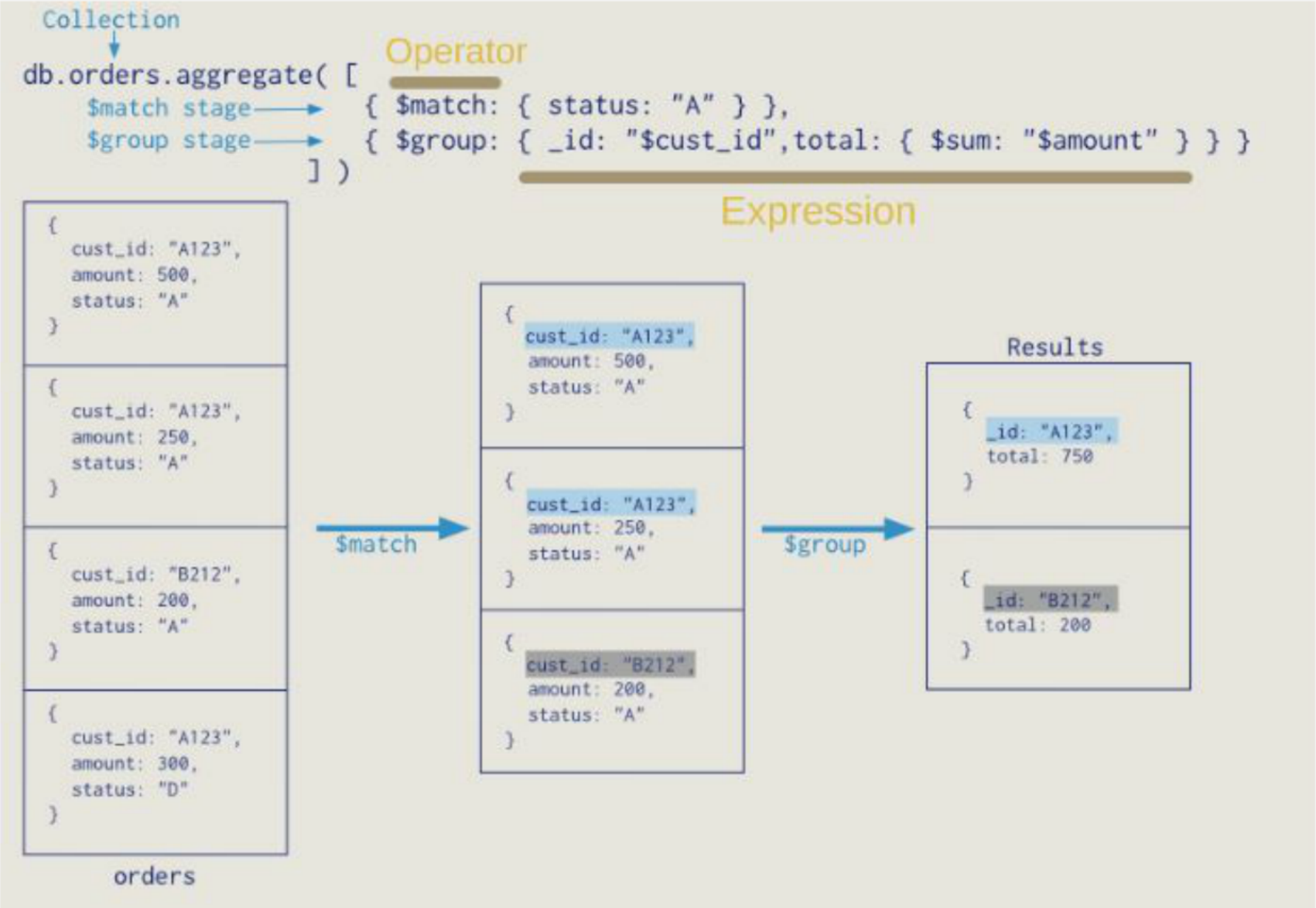
Aggregation

Aggregations are operations that process data records and return computed results.

Aggregations operations process data records and return computed results. Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result.

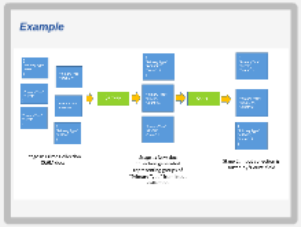
In sql count(*) and with group by is an equivalent of mongodb aggregation.

Aggregation

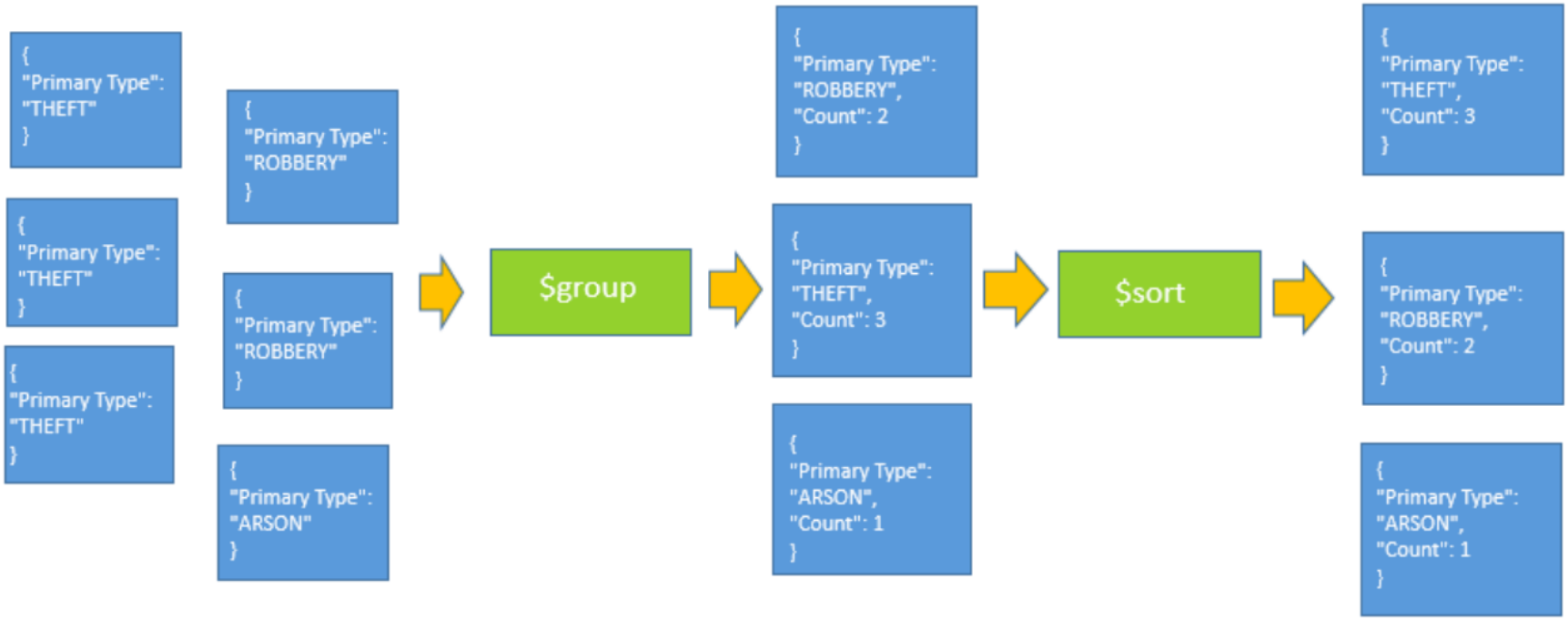


Real Example

What is the most common type of crime committed in Chicago between November 2014 and December 2015?



Example



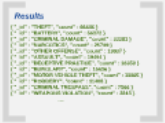
Stage 0: Crime Collection (5.6M docs)

Stage 1: New doc collection generated representing groups of "Primary Type" from input collection

Stage 2: Input collection is sorted by 'Count' field

Code

```
db.crimes.aggregate(
        {
            $group : {  _id: "$Primary Type" ,  count:{$sum: 1}   }
        },
        {
             $sort: { count: -1 }
        }
            )
```



Results

```
{ "_id" : "THEFT", "count" : 66446 }
{ "_id" : "BATTERY", "count" : 56372 }
{ "_id" : "CRIMINAL DAMAGE", "count" : 33281 }
{ "_id" : "NARCOTICS", "count" : 25799 }
{ "_id" : "OTHER OFFENSE", "count" : 19937 }
{ "_id" : "ASSAULT", "count" : 19494 }
{ "_id" : "DECEPTIVE PRACTICE", "count" : 16359 }
{ "_id" : "BURGLARY", "count" : 15636 }
{ "_id" : "MOTOR VEHICLE THEFT", "count" : 11805 }
{ "_id" : "ROBBERY", "count" : 11408 }
{ "_id" : "CRIMINAL TRESPASS", "count" : 7566 }
{ "_id" : "WEAPONS VIOLATION", "count" : 3813 }
```

.....

DEMO

Thanks!

MongoDB Introduction

Ali Jaafar
SMB214

ali.jaafar86@gmail.com

01-01-2016