
mongoDB

Traditional Database	MongoDB
Relational	Document-Orientated
Server	Server
Database	Database
Table	Collection
Row	Document
Column	Attribute
SQL Query	BSON Query
Index	Index

```
doc =
{ _id: new ObjectId("4c4b1476238d3b4dd5003981"),
  slug: "wheel-barrow-9092",
  sku: "9092",
  name: "Extra Large Wheel Barrow",
  description: "Heavy duty wheel barrow...",

  details: {
    weight: 47,
    weight_units: "lbs",
    model_num: 4039283402,
    manufacturer: "Acme",
    color: "Green"
  },

  total_reviews: 4,
  average_review: 4.5,
```

ObjectId("507f1f77bcf86cd799439011")

ObjectId("507f1f77bcf86cd799439011")

a 4-byte value representing the seconds since the Unix epoch,
a 3-byte machine identifier,
a 2-byte process id, and
a 3-byte counter, starting with a random value.

Normalization & Denormalization

http://makble.com/normalization-and-denormalization-with-mongodb

Normalized post

```
array(
    'title' => 'post title',
    'category' => 'software',
    'author' => $authorid

);

array(
    'authorid' => $authorid,
    'author_name' => 'james',
    'author_description' => 'A software engineer'
);
```

Querying normalized posts

```
$results  = $collection->findone(array('title' => 'post title'));
$author = $collection->findone('authorid' => $results['author']);
$results['author'] = $author;
```

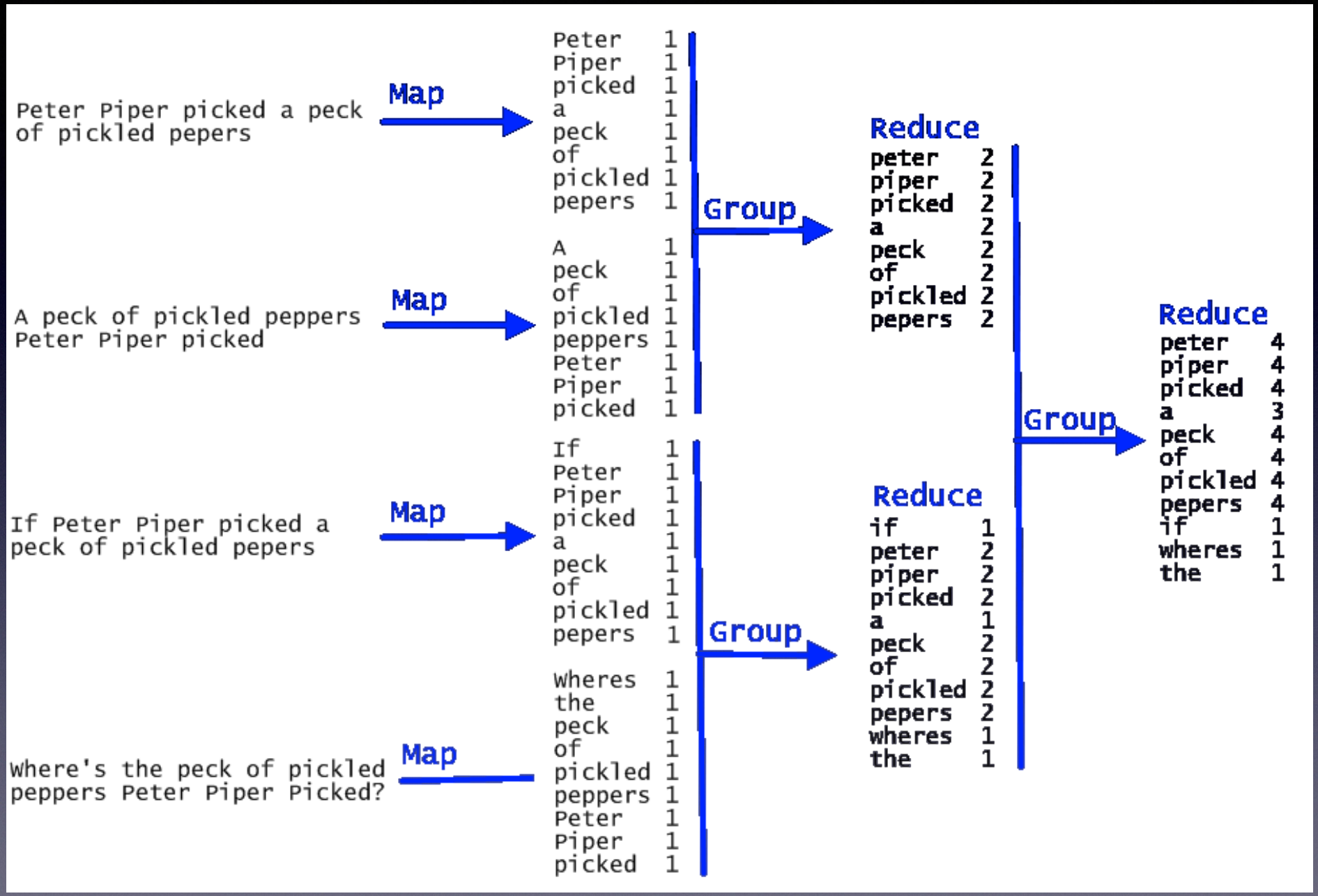
De-normalized post

```
array(
    'title' => 'post title',
    'category' => 'software',
    'author' => array(
        'authorid' => $authorid,
        'author_name' => 'james',
        'author_description' => 'A software engineer'
        )
    );
```

http://docs.mongodb.org/ecosystem/use-cases/

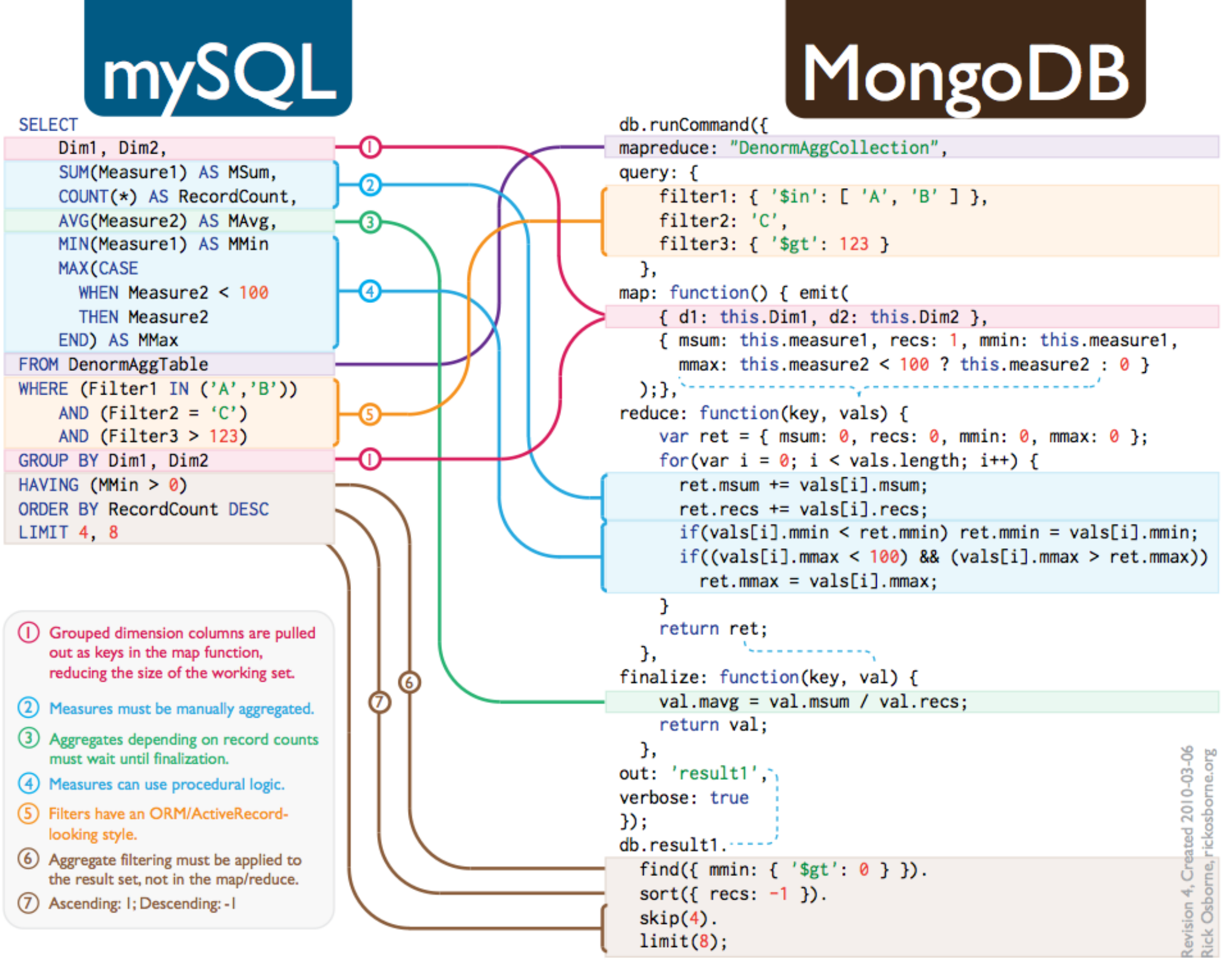
http://docs.mongodb.org/manual/reference/operator/query/

Map reduce


Peter Piper picked a peck of pickled pepers
Map
Peter 1
Piper 1
picked 1
a 1
peck 1
of 1
pickled 1
pepers 1
A peck of pickled peppers Peter Piper picked
Map
A 1
peck 1
of 1
pickled 1
peppers 1
Peter 1
Piper 1
picked 1
If Peter Piper picked a peck of pickled pepers
Map
If 1
Peter 1
Piper 1
picked 1
a 1
peck 1
of 1
pickled 1
pepers 1
Where's the peck of pickled peppers Peter Piper Picked?
Map
Wheres 1
the 1
peck 1
of 1
pickled 1
peppers 1
Peter 1
Piper 1
picked 1
Group
Group
Reduce
peter 2
piper 2
picked 2
a 2
peck 2
of 2
pickled 2
pepers 2
Reduce
if 1
peter 2
piper 2
picked 2
a 1
peck 2
of 2
pickled 2
pepers 2
wheres 1
the 1
Group
Reduce
peter 4
piper 4
picked 4
a 3
peck 4
of 4
pickled 4
pepers 4
if 1
wheres 1
the 1

```
function map(String name, String document):
  // name: document name
  // document: document contents
  for each word w in document:
    emit (w, 1)

function reduce(String word, Iterator partialCounts):
  // word: a word
  // partialCounts: a list of aggregated partial counts
  sum = 0
  for each pc in partialCounts:
    sum += ParseInt(pc)
  emit (word, sum)
```


mySQL
MongoDB
SELECT
Dim1, Dim2,
SUM(Measure1) AS MSum,
COUNT(*) AS RecordCount,
AVG(Measure2) AS MAvg,
MIN(Measure1) AS MMin
MAX(CASE
WHEN Measure2 < 100
THEN Measure2
END) AS MMax
FROM DenormAggTable
WHERE (Filter1 IN ('A','B'))
AND (Filter2 = 'C')
AND (Filter3 > 123)
GROUP BY Dim1, Dim2
HAVING (MMin > 0)
ORDER BY RecordCount DESC
LIMIT 4, 8
db.runCommand({
mapreduce: "DenormAggCollection",
query: {
filter1: { '$in': ['A', 'B'] },
filter2: 'C',
filter3: { '$gt': 123 }
},
map: function() { emit(
{ d1: this.Dim1, d2: this.Dim2 },
{ msum: this.measure1, recs: 1, mmin: this.measure1,
mmax: this.measure2 < 100 ? this.measure2 : 0 }
);},
reduce: function(key, vals) {
var ret = { msum: 0, recs: 0, mmin: 0, mmax: 0 };
for(var i = 0; i < vals.length; i++) {
ret.msum += vals[i].msum;
ret.recs += vals[i].recs;
if(vals[i].mmin < ret.mmin) ret.mmin = vals[i].mmin;
if((vals[i].mmax < 100) && (vals[i].mmax > ret.mmax))
ret.mmax = vals[i].mmax;
}
return ret;
},
finalize: function(key, val) {
val.mavg = val.msum / val.recs;
return val;
},
out: 'result1',
verbose: true
});
db.result1.
find({ mmin: { '$gt': 0 } }).
sort({ recs: -1 }).
skip(4).
limit(8);
1
2
3
4
5
1
6
7
1 Grouped dimension columns are pulled out as keys in the map function, reducing the size of the working set.
2 Measures must be manually aggregated.
3 Aggregates depending on record counts must wait until finalization.
4 Measures can use procedural logic.
5 Filters have an ORM/ActiveRecord-looking style.
6 Aggregate filtering must be applied to the result set, not in the map/reduce.
7 Ascending: 1; Descending: -1
Revision 4, Created 2010-03-06
Rick Osborne, rickosborne.org