mongoDB

Why I am using it

by pete brumm

2012/04/19

BACKGROUND

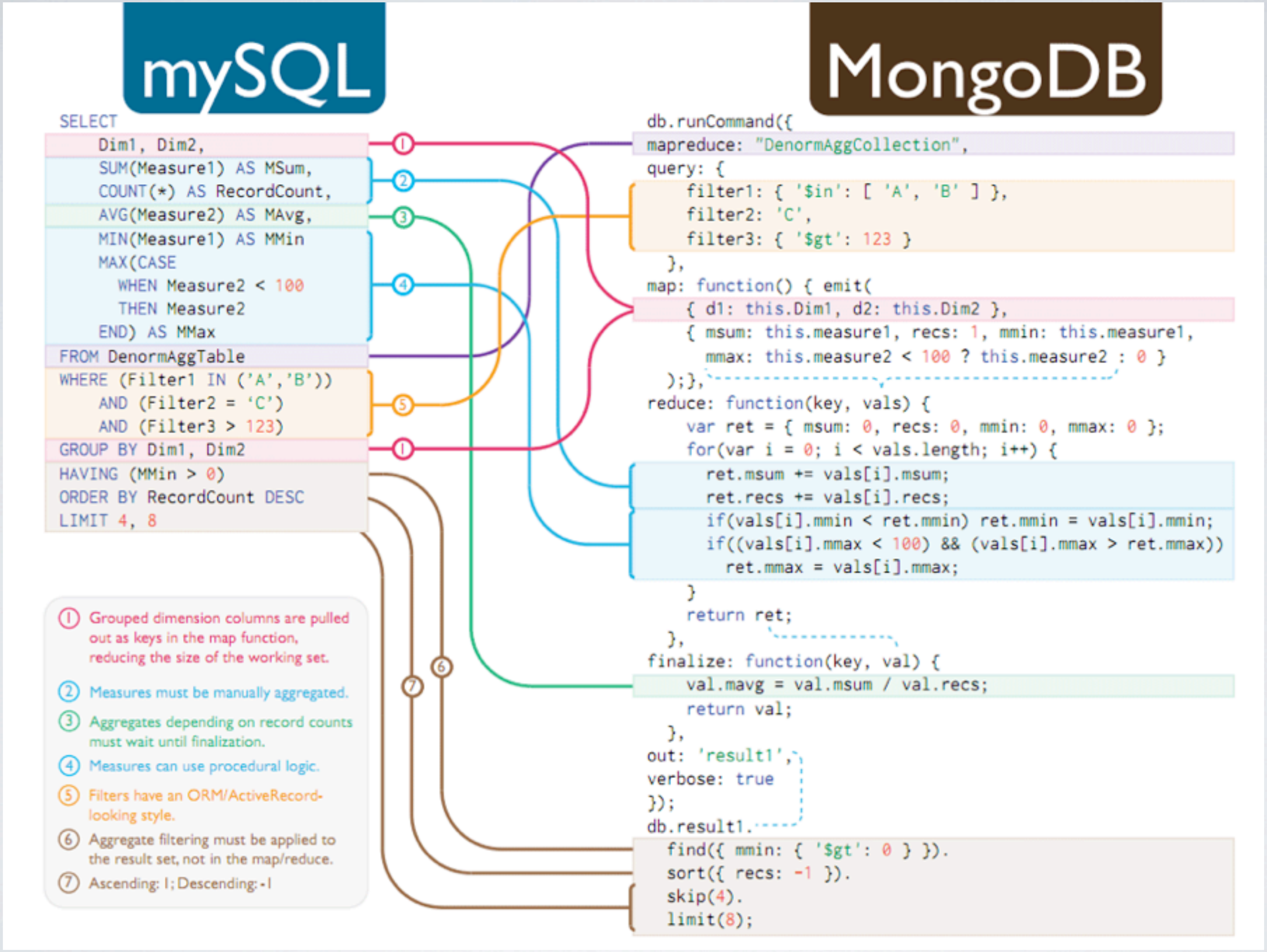
- 15 years in professional software development
  - 10 years java, 6 years ruby
- 15 successful years with SQL databases
  - (Oracle, Mysql, Sybase, Sql Server)
- Designed SQL database apps with DB file sizes > 500GB
- Apps that average 100M page views a day

WHY MONGODB OVER SQL

- Fewer Tables.   link tables can be embedded
- no more ALTER TABLE's.
- Atomic operations. $inc $addToSet $pop
- Upsert operations.
    - `db.people.update( { name:"Joe" }, { $inc: { x:1, y:1 } }, true );`
- Easier Replication/Failover (although sql db's are catching up)
- GridFS for storing Files.   Replicates with rest of data

WHY MONGODB OVER OTHER NOSQL

- Powerful Ad hoc Query language (similar to sql)
  - Doesn't depend on map-reduce to access data
- multi field indexes
  - including indexing into arrays, hashes



```
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 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.

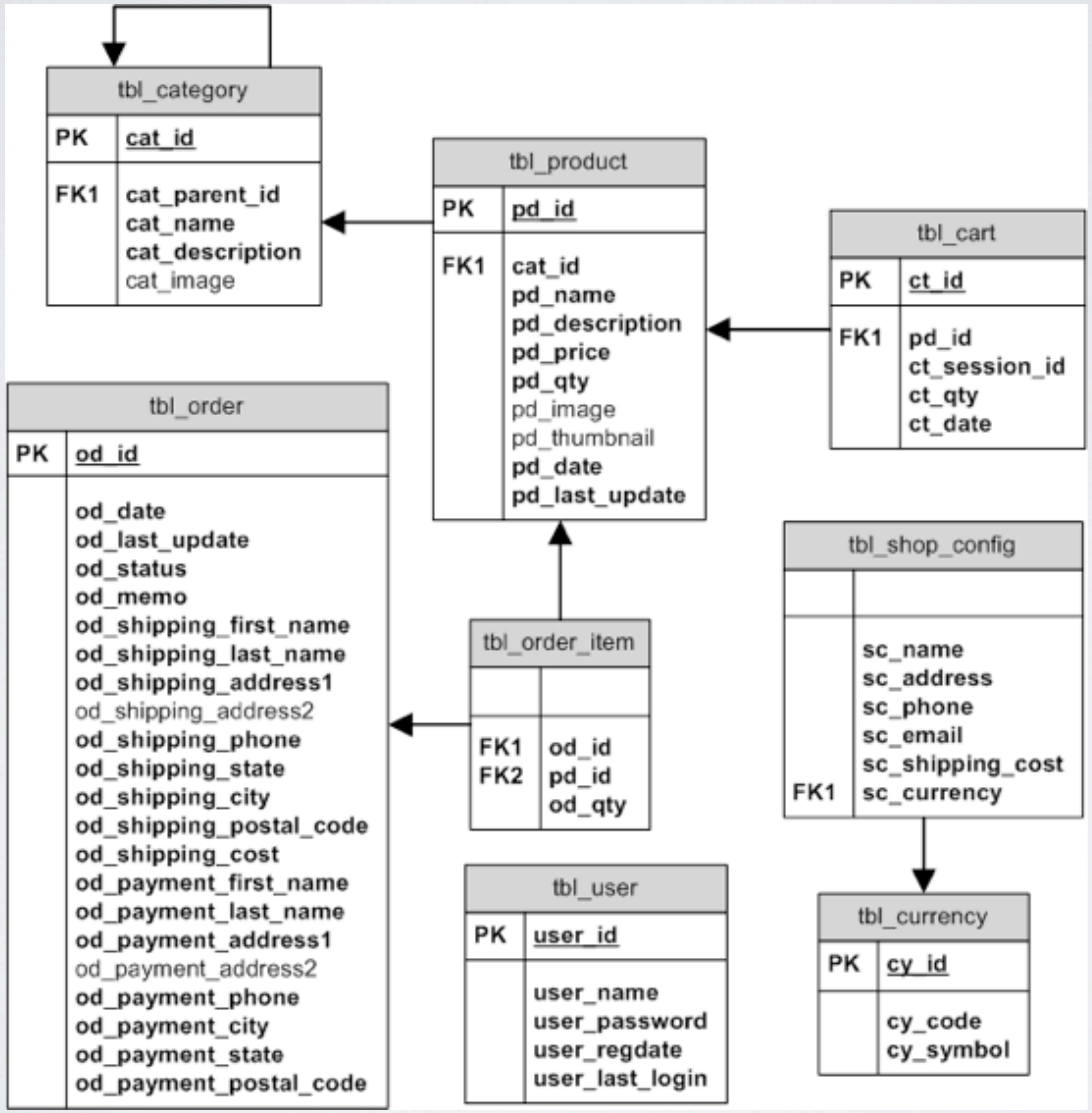
5 Filters have an ORM/ActiveRecord-looking style.

6 Aggregate filtering must be applied to the result set, not in the map/reduce.

7 Ascending: 1; Descending: -1

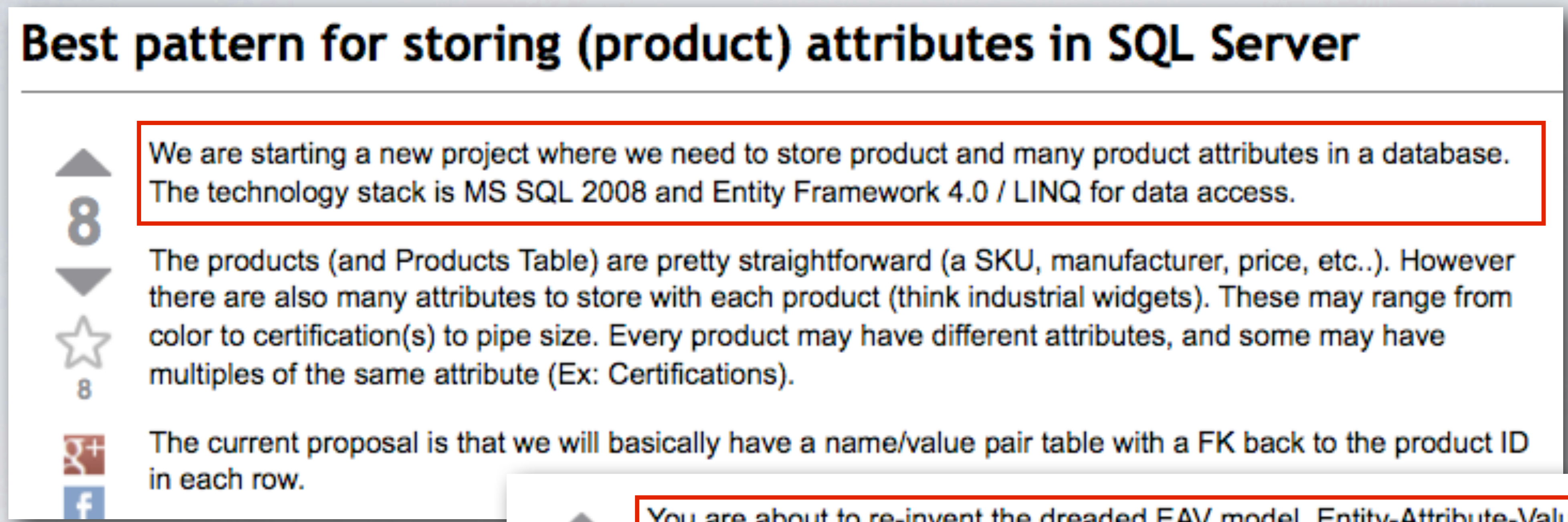
http://nyccto.wordpress.com/2011/03/10/sql-to-mongodb-mapping-infographic/

Simple Sql Cart Design



http://www.phpwebcommerce.com/shopping-cart-database-design.php

WHAT ABOUT PRODUCT ATTRIBUTES



Best pattern for storing (product) attributes in SQL Server

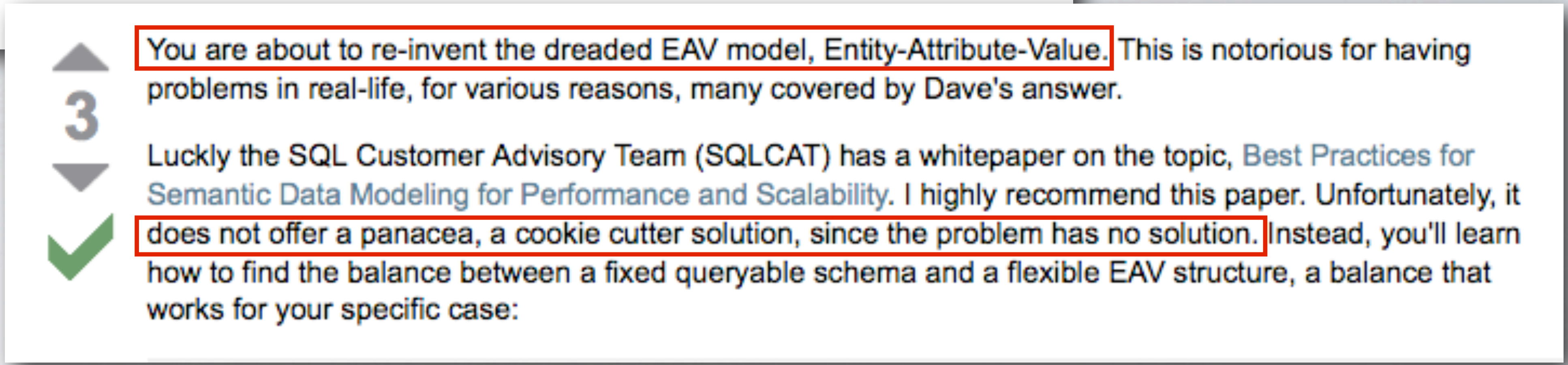
8

8

We are starting a new project where we need to store product and many product attributes in a database. The technology stack is MS SQL 2008 and Entity Framework 4.0 / LINQ for data access.

The products (and Products Table) are pretty straightforward (a SKU, manufacturer, price, etc..). However there are also many attributes to store with each product (think industrial widgets). These may range from color to certification(s) to pipe size. Every product may have different attributes, and some may have multiples of the same attribute (Ex: Certifications).

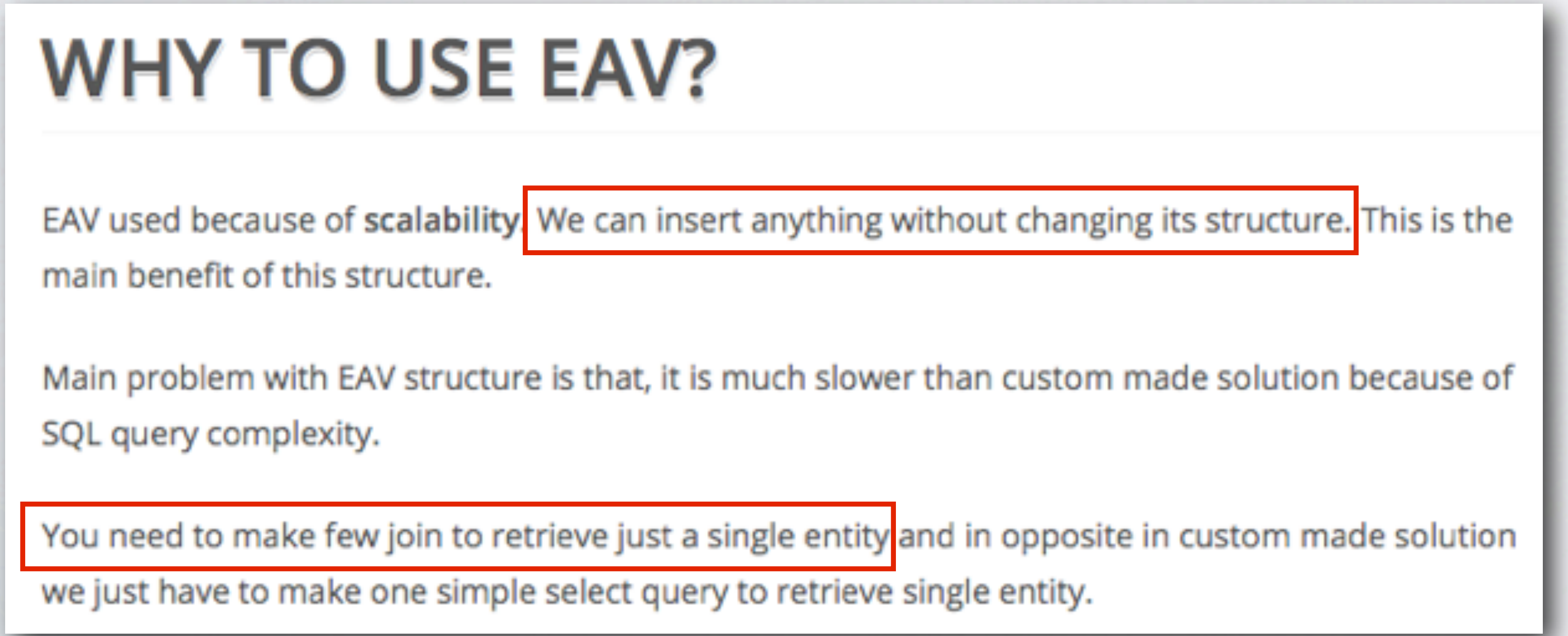
The current proposal is that we will basically have a name/value pair table with a FK back to the product ID in each row.



3

You are about to re-invent the dreaded EAV model, Entity-Attribute-Value. This is notorious for having problems in real-life, for various reasons, many covered by Dave's answer.

Luckly the SQL Customer Advisory Team (SQLCAT) has a whitepaper on the topic, Best Practices for Semantic Data Modeling for Performance and Scalability. I highly recommend this paper. Unfortunately, it does not offer a panacea, a cookie cutter solution, since the problem has no solution. Instead, you'll learn how to find the balance between a fixed queryable schema and a flexible EAV structure, a balance that works for your specific case:



WHY TO USE EAV?

EAV used because of **scalability**. We can insert anything without changing its structure. This is the main benefit of this structure.

Main problem with EAV structure is that, it is much slower than custom made solution because of SQL query complexity.

You need to make few join to retrieve just a single entity and in opposite in custom made solution we just have to make one simple select query to retrieve single entity.

MongoDB Way

users

```
{
  "_id": 5,
  "email": "pete@petebrumm.com",
  "address": {"addr1": "Test Address", "addr2": "", "city": "", "state": "", "zip": ""}
}
```

products

```
{
  "_id": 3,
  "name": "Test Product",
  "category_ids": [1],
  "price": 19.95,
  "currency": "usd",
  'attrs': [
    {"id": 2, v: "White"}
  ]
}
```

categories

```
{
  "_id": 1,
  "name": "Root Category",
  "path": [],
}

{
  "_id": 7,
  "name": "Health",
  "path": [1],
}
```

attributes

```
{
  "_id": 2,
  "name": "Color",
  "searchable": true,
  "values": ["Red", "White",
"Green"],
  "category_ids": [1],
  "val_products": {
    "red": [5],
    "white": [3]
  }
}
```

carts

```
{
  "user_id": 5,
  "status": "open",
  "items": [
    {
      "product_id": 3,
      "qty": 1,
      "name": "Test Product",
      "price": 19.95,
      "currency": "usd"
    }
  ]
}
```

Demo

Gotchas & Recommendations

- don't forget to use $set or other atomic operation on updates
    - Otherwise you are replacing existing doc instead of updating it
- run on 64bit OS
- I don't recommend windows for production
- Map reduce is not for realtime queries. (also not multi core)
- Use Replica Sets
- indexes can only have one array
- Use MongoID ruby gem

LINKS

- http://www.mongodb.org/
- Rails and MongoDB with MongoID
    - http://railsapps.github.com/tutorial-rails-mongoid-devise.html
- http://www.10gen.com/presentations
- Sql to mongo query mapping