ArangoDB

An e-commerce app in action built on top of a multi-model database

6 June 2017

Max Neunhöffer

The Multi-Model Approach

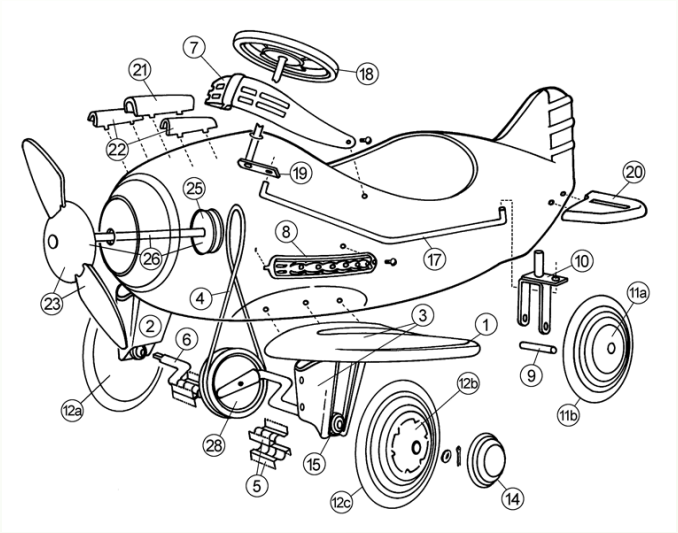
Native multi-model database

A native multi-model database combines a document store with a graph database and is at the same time a key/value store,
with a common query language for all three data models.

Important:

- Is able to compete with specialised products on their turf.
- Allows for **polyglot persistence** using a single database technology.
- In a microservice architecture, there will be several **different** deployments.

Use case: Aircraft fleet management



Use case: Aircraft fleet management

One of our customers uses ArangoDB to

- store each part, component, unit or aircraft as a document
- model containment as a graph
- thus can easily find all parts of some component
- keep track of maintenance intervals
- perform queries orthogonal to the graph structure
- thereby getting good efficiency for all needed queries

http://radar.oreilly.com/2015/07/data-modeling-with-multi-model-databases.html

Why is multi-model possible at all?

Document stores and key/value stores

Document stores: have primary key, are key/value stores.

Without using secondary indexes, performance is nearly as good as with opaque data instead of JSON.

Good horizontal scalability can be achieved for key lookups.

https://www.arangodb.com/2015/10/benchmark-postgresql-mongodb-arangodb/

Why is multi-model possible at all?

Document stores and graph databases

Graph database: would like to associate arbitrary data with vertices and edges, so JSON documents are a good choice.

- A good edge index, giving fast access to neighbours.
  This can be a secondary index.
- Graph support in the query language.
- Implementations of graph algorithms in the DB engine.

https://www.arangodb.com/2015/10/benchmark-postgresql-mongodb-arangodb/

Powerful query language

AQL

The built in Arango Query Language allows

- complex, powerful and convenient queries,
- with transaction semantics,
- allowing to do joins,
- and to do graph queries,
- AQL is independent of the driver used and
- offers protection against injections by design.

Example AQL queries 1

Show new customers from 2016

```
FOR c IN customers
  FILTER c.memberSince >= "2016"
  RETURN c
```

Example AQL queries 2

Count all customers

```
FOR c IN customers
  COLLECT WITH COUNT INTO count
  RETURN {"number of customers": count}
```

Example AQL queries 3

Show the number of new customers in 2015

```
FOR c IN customers
  FILTER c.memberSince >= "2015" && c.memberSince < "2016"
  COLLECT WITH COUNT INTO count
  RETURN {"number of new customers in 2015": count}
```

Example AQL queries 4

Statistics about signup numbers over years

```
FOR c IN customers
  COLLECT y = SUBSTRING(c.memberSince,0,4) WITH COUNT INTO count
  SORT y DESC
  RETURN { year: y, noNewCustomers: count }
```

Example AQL queries 5

Show 10 first items with a price over 200

```
FOR i IN items
  FILTER i.price >= 200
  LIMIT 10
  RETURN i
```

Example AQL queries 6

Show all sales of October 2016

```
FOR s IN sales
  FILTER s.date >= "2016-10-01" && s.date < "2016-11-01"
  SORT s.date
  RETURN { date: s.date, billingId: s.billingId }
```

Example AQL queries 7

Join the price

```
FOR s IN sales
  FILTER s.date >= "2016-10-01" && s.date < "2016-11-01"
  SORT s.date
  FOR i IN items
    FILTER i._id == s._to
    RETURN {date:s.date,billingId:s.billingId,price:i.price}
```

Example AQL queries 8

Show complete orders

```
FOR s IN sales
  FOR i IN items
    FILTER i._id == s._to
    COLLECT bill = s.billingId INTO items
    FOR c IN customers
      FILTER c._id == items[0].s._from
      RETURN { date: items[0].s.date, billingId: bill,
               name: c.name, price: SUM(items[*].i.price),
               items: items[*].i }
```

Example AQL queries 9

Find top sellers (using graph)

```
FOR item IN items
  LET buyers =
    (FOR v IN 1..1 INBOUND item GRAPH "sales" RETURN v)
  LET nr = LENGTH(buyers)
  SORT nr DESC
  RETURN { description: item.description,
           orderId: item.orderId, nrSales: nr }
```

Example AQL queries 10

Given a customer, find all items he has bought

```
LET customer = DOCUMENT("customers/hugo")
FOR i IN 1..1 OUTBOUND customer GRAPH "sales"
  COLLECT orderId = i.orderId INTO list
  RETURN { orderId, description: list[0].i.description,
           count: LENGTH(list), customer }
```

Example AQL queries 11

Find recommendations

```
LET customer = DOCUMENT("customers/lulu")
FOR i, e, p IN 3..3 ANY customer GRAPH "sales"
  OPTIONS { uniqueVertices: "none", uniqueEdges: "none" }
  FILTER p.vertices[2]._key != customer._key
  COLLECT orderId = i.orderId INTO list
  RETURN { orderId, description: list[0].i.description,
           count: LENGTH(list), customer: customer.name,
           path: list[0].p.vertices[*]._key }
```

Links

https://www.arangodb.com

https://docs.arangodb.com/cookbook/index.html

https://docs.arangodb.com/3.1/Manual/Foxx/

https://github.com/hapijs/joi