ArangoDB

The multi-model database for graph and beyond

(NDBI040) Jindřich Bär, 2022

Introduction

- Created in 2011, latest v3.9.1 (07/04/2022)
- Free and open-source multi-model DBMS
- Supports distributed deployment modes
- Provides a native REST API over HTTP

Data models

- **Key/value data**
- **Hierarchical (JSON) documents**
- **Graph data**
- All can be combined in one query
- Under the hood, all these are **documents**

Key-Value example

```
db.coll.save({_key: "fightclub", title: "Fight Club"})

db.coll.save({_key: "se7en", title: "Se7en"})

db.coll.document("fightclub").title

> Fight Club
```

- The `_key` property is indexed with the primary index
- The “value” does not need to be flat, can be anything (anything JSON-able)

ArangoDB

Document example

```
db.coll.save({
_key: "fightclub",
info: { year: 1999 },
title: "Fight Club" });
```

```
db.coll.save({
_key: "se7en",
info: { year: 1995 },
title: "Se7en" });
```

```
db.coll.byExample({ info: { year: 1995 }}).toArray();
> {
   ...
    "info" : { "year" : 1995 },
    "title" : "Se7en"
  }
```

ArangoDB

Graph ...example?

```
[{
    "id": "person/dfincher",
    "firstName" : "David",
    "familyName" : "Fincher"
  },
  {
    "id": "person/enorton",
    "firstName" : "Edward",
    "familyName" : "Norton"
  }
...
]
```

```
[{
   "_from":"person/enorton",
   "_to":"person/dfincher",
   "since" : 1999
 },
 {
   "_from":"person/bpitt",
   "_to":"person/dfincher",
   "since": 1995
 }, ...
]
```

ArangoDB

Graph functions

- **ArangoDB** provides a myriad of graph-related functions
  - ._neighbors(vertex)
  - ._distanceTo(vertex, vertex)
  - .shortestPath(vertex, vertex), ...
- **Pregel** - Distributed Iterative Graph Processing
  - PageRank, Connected Components, Community Detection, Single source shortest path, ...

AQL

- `arangosh` runs on JS (see prior examples)
- AQL – only data **manipulation** language
  - missing DDL elements (CREATE, ALTER, DROP)
  - database structure needs to be created in advance
  - syntax similar to other query languages

AQL – examples #1

- (WHERE) Select all movies newer than 1996:

```
FOR m in movie FILTER m.info.year >= 1996 RETURN m
```

- (GROUP BY) Count movies by years:

```
FOR m in movie COLLECT year = m.info.year WITH COUNT into c
RETURN { year: year, count: c }
```

- (UPDATE) Age all actors by 5 years

```
FOR a in actor UPDATE a WITH {year: a.year - 5} IN actor
```

ArangoDB

AQL – examples #2

- (JOIN) For every actor, list their movies

```
FOR a in actor FOR m in movie FILTER
CONTAINS_ARRAY(m.actors, a._id) RETURN {actor: a, movie: m}
```

- (Graph) All the people who know Edward Norton

```
FOR x in ANY 'person/enorton' know RETURN x
```

- (Graph) Find the shortest chain of people between two actors

```
FOR person IN ANY SHORTEST_PATH 'person/enorton' TO
'person/mfreeman' know RETURN person)
```

ArangoDB

Pros / cons

Pros

- Distributed
- Competitive performance
- Extensive graph-related features
- Steep learning curve (JS)

Cons

- Relatively small user base
- …?

ArangoDB

Thank you for your attention