Binary trees

Key/value based approach:

```
{ "data":3,
  "lft": {
    "data":1,
    "lft": {"data":0},
    "rgt": {"data":2}
  },
  "rgt": {
    "data":5,
    "lft": {"data":4},
    "rgt": {"data":6}
  }
}
```

Schema:

```
{"type": "object",
 "oneOf": [{"$ref": "#/definitions/leaf"},
           {"$ref": "#/definitions/branch"}],
 "definitions": {
    "leaf": {
      "type": "object",
      "properties": {
        "data": {"type": "integer"},
        "rgt" : false,
        "lft" : false
      },
      "required": ["data"]},
    "branch": {
      "type": "object",
      "properties": {
        "data": {"type": "integer"},
        "lft": {"$ref": "#"},
        "rgt": {"$ref": "#"}},
      "required": ["data","lft","rgt"]}
 }
}
```

Variant schema: Just remove required "data" from branch

Queries:

```
$..data
$..lft.data
$.*.*.data
$..[?(@.data>3)].data
$..[?(@.data>$.data)].data
```

Array-based approach:

```
{ "data":3,
  "children": [
    {"data":1,
     "children": [{"data":0},{"data":2}]},
    {"data":5,
     "children": [{"data":4},{"data":6}]}
  ]
}
```

Schema:

```
{
  "properties": {
    "data": {"type": "integer"},
    "children": {"type": "array",
                 "items" : {"$ref": "#"},
                 "minItems":2,
                 "maxItems":2}},
  "required": ["data"]
}
```

Variant schema: Kind of tricky. Requires a separate definitions for branches/leafs, with leafs requiring "data" and branches requiring "children" (and having data optional).

Queries:

```
$..data
$..[0].data  (doesn't work in Jayway :| )
$.children.*.children.*.data
$..[?(@.data>3)].data
$..[?(@.data>$.data)].data
```

Flights

Key/value pairs can be used to have primary keys (because there are no compound keys!) Tiny example data (used to build schema):

```
{ "Airports": {"GOT": "Gothenburg"},
  "FlightCodes": {"SK111": "SAS"},
  "Flights": {"SK111" : {dep : "GOT", dest: "FRA"}}
}
```

Schema:

```
{"type": "object",
 "properties": {
   "Airports": {
     "type":"object",
     "additionalProperties":{"type":"string"}
   },
   "FlightCodes": {
      "type":"object",
      "additionalProperties":{"type":"string"}
   },
   "Flights": {
      "type":"object",
      "additionalProperties":{
         "type":"object",
         "properties":{
         "dep":{"type":"string"},
                "dest":{"type":"string"}},
            "required": ["dep","dest"],
            "additionalProperties": false
      }
   }},
 "required": ["Airports","FlightCodes","Flights"]
}
```

Complete data:

```
{ "Airports": {"GOT": "Gothenburg",
               "FRA": "Frankfurt",
               "ORY": "Paris",
               "MUC": "Munich",
               "MLA": "Malta"},
  "FlightCodes": {"SK111": "SAS",
                  "AF222": "Air France",
                  "AB222": "Air Berlin",
                  "KM111": "Air Malta"} ,
  "Flights": {"SK111" : {dep : "GOT", dest: "FRA"},
              "AF222" : {dep : "ORY", dest: "MLA"},
              "AB222" : {dep : "FRA", dest: "MUC"},
              "KM111" : {dep : "MUC", dest: "MLA"}}}
```

Applications

(A) Here we use applicant id numbers as keys, and associate it with the applicants name and a list of all their choices. We array positions to represent choice numbers (indexed from 0 instead of 1).

```
{
  "a1": {"name": "Andersson", "choices": [
      {"code":"MPSOF","meritScore":750},
      {"code":"MPALG","meritScore":750},
      {"code":"MPCSN","meritScore":800}
    ]
  },
  "a2": {"name": "Jonsson", "choices": [
      {"code":"MPALG","meritScore":700}
    ]},
  "a3": {"name": "Larsson", "choices": [
      {"code":"MPCSN","meritScore":850},
      {"code":"MPALG","meritScore":850}
    ]}
}
```

(B)

```
{
  "type": "object",
  "additionalProperties": {
    "type": "object",
    "properties":{
      "name":{"type":"string"},
      "choices":{
        "type":"array"
      }
    },
    "required":["name", "choices"]
  }
}
```

(C) Note that application 1 has array index 0

```
$.*.choices[0][?(@.meritScore>800)]
```

```
BONUS QUERY!:
$.*[?(@.choices[0].meritScore>800)].name
```

Finds the name of the applicant in question