CSCI835 Database Systems

Validation with JSON Schema

Dr Janusz R. Getta

School of Computing and Information Technology -
University of Wollongong

Validations

Outline

Overview

Semistructured and schemaless properties of JSON/BSON data model allow for very flexible manipulation of database structures

In the same moment very flexible and uncontrolled manipulations of database structures open the possibilities for pretty easy corruption of database structures and database contents, for example, due to the random mistakes

Practice shows that certain level of verification of database consistency constraints is always needed

MongoDB provides the capability to validate documents during updates and insertions (and not deletions)

Validation rules are specified within `createCollection()` method using the `validator` option

It is possible use `collMod` command with the `validator` option to turn on/off validation rules

Validations

Outline

JSON Schema validation

JSON Schema is a BSON document that defines the structure of JSON data for validation, documentation, and interaction control

JSON Schema validation of a JBSON document is performed through verification of consistency of the structures and contents of the document with JSON Schema

JSON Schema is based on the concepts "borrowed" from XML Schema

A method `createCollection()` creates an empty collection and associates `validator` with a container

If validation with JSON Schema is required that `validator` is associted with a JSON schema that determines the structures and the contents of the documents in the container

It is possible use `collMod` method with the `validator` option to turn on/off validation rules

Validations

Outline

$jsonSchema operator

In MongoDB an operator $jsonSchema is asociated with a JSON Schema to be used for validation of BSON documents

$jsonSchema operator can also be used to query with find command or with $match aggregation stage

The following application of $jsonSchema operator validates any BSON document in a collection department

```
                                                                $jsonSchema operator
db.createCollection("department",
                    {"validator":{$jsonSchema:{"bsonType":"object"} }
                    } );
```

For example, the following BSON document validates well

```
                                A sample BSON document that validates with $jsonSchema validator above
{"name":"School of Astronomy",
 "address":{"street":"Franz Josef Str",
            "bldg":4},
 "courses":[{"code":"SOA101",
             "title":"Astronomy for Kids",
             "credits":3} ] }
```

Validations

Outline

Validation of any BSON documents

The following JSON Schema validates well any document in a collection `anyDocument`

JSON Schema validator that validates any document

```
db.createCollection("anyDocument",
                    {"validator":{$jsonSchema:{"bsonType":"object"} }
                    } );
```

Note the application of `bsonType` in MongoDB validation instead of `type` in standard JSON schema

Validations

Outline

Validation of flat BSON documents

The following BSON document in a collection `tiny`

A tiny BSON document

```
db.tiny.insert({"name":"Harry Potter"});
```

fails validation against the following JSON schema

$jsonSchema validator

```
db.createCollection( "tiny",
                     {"validator":{$jsonSchema:
{"bsonType":"object",
 "properties":{"name":{"bsonType":"string"} },
 "required":["name"],
 "additionalProperties":false
} } } );
```

A validation fails because a key `additionalProperties` is associated with false and a key `"_id"` included in every BSON document is not on `required` list

Validation of flat BSON documents

The following BSON document in a collection `tiny`

A tiny BSON document

```
db.tiny.insert({"_id":"HP666","name":"Harry Potter"});
```

passes validation against the following JSON schema

$jsonSchema validator

```
db.createCollection( "tiny",
                     {"validator":{$jsonSchema:
{"bsonType":"object",
 "properties":{"_id":{"bsonType":"string"},
               "name":{"bsonType":"string"} },
 "required":["_id","name"],
 "additionalProperties":false
} } } );
```

A validation passes because a key `additionalProperties` is associated with false and a key `"_id"` included in every BSON document is on `required` list

Validation of flat BSON documents

The following BSON documents in a collection `empty`

BSON documents in a collection empty

```
db.empty.insert({"_id":"HP666"});
db.empty.insert({"_id":"HP667","name":"Harry Potter"});
db.empty.insert({"_id":"HP668","name":"Harry Potter","occupation":"wizard"});
```

pass validation against the following JSON schema

$jsonSchema validator

```
db.createCollection( "empty",
                     {"validator":{$jsonSchema:
{"bsonType":"object",
 "properties":{"_id":{"bsonType":"string"} },
 "required":["_id"],
 "additionalProperties":true
} } } );
```

A validation passes because a key `additionalProperties` is associated with true and a key `"_id"` included in every BSON document is on `required` list

Validation of flat BSON documents

The following BSON document in a collection `flat`

A tiny BSON document

```
db.flat.insert({"_id":"HP666","name":"Harry Potter","age":NumberInt("100")});
```

passes validation against the following JSON schema

$jsonSchema validator

```
db.createCollection( "flat",
                    {"validator":{$jsonSchema:
{"bsonType":"object",
 "properties":{"name":{"bsonType":"string",
                       "maxLength":100},
               "age":{"bsonType":"int",
                      "maximum":200,
                      "exclusiveMaximum":true} },
 "required":["name","age","_id"],
 "additionalProperties":false
} } } );
```

A default type of a value associated with a key `"_id"` is `string`

Validation of nested BSON documents

The following BSON documents in a collection `department`

BSON document

```
db.department.insert({"_id":"Finance","budget":123});
db.department.insert({"_id":"Sales","budget":123,"fee":456});
```

pass validation against the following JSON schema

$jsonSchema validator

```
db.createCollection("department",
                    { "validator":{$jsonSchema:
  {"bsonType":"object",
   "properties":{"_id":{"bsonType":"string"},
                 "budget":{"bsonType":"double",
                           "description":"Budget of department"},
                 "fee":{"bsonType":"double",
                        "description":"Fee paid"}
                },
   "required":["_id","budget"],
   "additionalProperties":false
  } } } );
```

Validations

Outline

Validation of nested BSON documents

The following BSON document in a collection `nested`

A nested BSON document

```
db.nested.insert({"_id":"HP667",
                  "name":"Harry Potter",
                  "address":{"city":"Dapto",
                             "code":NumberInt("2530") }
                 });
```

passes validation against the following JSON schema

$jsonSchema validator

```
db.createCollection( "nested",
                    {"validator":{$jsonSchema:
{"bsonType":"object",
 "properties":{"_id":{"bsonType":"string"},
               "name":{"bsonType":"string",
                       "maxLength":100},
               "address":{"bsonType":"object",
                          "properties":{"city":{"bsonType":"string",
                                                "minLength":5,"maxLength":30},
                                        "code":{"bsonType":"int",
                                                "maximum":9999,
                                                "exclusiveMaximum":false} },
                          "required":["city","code"],
                          "additionalProperties":false } },
 "required":["name","address","_id"],
 "additionalProperties":false } } } );
```

Validation of nested BSON documents

The following BSON document in a collection `nested`

A nested BSON document

```
db.nested.insert({"_id":"HP667",
                  "name":"Harry Potter",
                  "cars":["Ferrari","Mercedes","Hyundai"]
                 });
```

passes validation against the following JSON schema

$jsonSchema validator

```
db.createCollection( "nested",
                     {"validator":{$jsonSchema:
{"bsonType":"object",
 "properties":{"_id":{"bsonType":"string"},
               "name":{"bsonType":"string",
                       "maxLength":100},
               "cars":{"bsonType":"array",
                       "items":{"bsonType":"string"},
                       "uniqueItems":true }
              },
 "required":["name","cars","_id"],
 "additionalProperties":false
} } } );
```

Validation of nested BSON documents

The following BSON document in a collection `month`

BSON document

```
db.month.insert( {"_id":"Winter",
                  "months":["June","July","August"]} );
```

passes validation against the following JSON schema

$jsonSchema validator

```
db.createCollection( "month",
                     {"validator":{$jsonSchema:
{"bsonType":"object",
 "properties":{"_id":{"bsonType":"string"},
               "months":{"bsonType":"array",
                         "description":"Winter",
                         "items":{"bsonType":"string"} } },
 "required":["_id","months"],
 "additionalProperties":false
} } } );
```

Validation of nested BSON documents

The following BSON document in a collection `addressBook`

BSON document

```
db.month.insert( {"_id":"addressBook",
                  "address":[{"city":"Sydney",
                              "street":"25 Victoria St."},
                             {"city":"Wollongong",
                              "street":"125 Northfields Ave."}] } );
```

passes validation against the following JSON schema

$jsonSchema validator

```
db.createCollection("department",
                    {"validator":{$jsonSchema:
{"bsonType":"object",
 "properties":{"_id":{"bsonType":"string"},
               "address":{"bsonType":"array",
                          "description":"Cities and streets",
                          "items":{"bsonType":"object",
                                   "properties":{"city":{"bsonType":"string"},
                                                 "street":{"bsonType":"string"} },
                                   "required":["city","street"],
                                   "additionalProperties":false
                                  }
                         }
              },
 "required":["_id","address"],
 "additionalProperties":false
} } } );
```

Validation of nested BSON documents

The following BSON documents in a collection `school`

BSON document

```
db.school.insert(
{ "name":"School of Astronomy",
    "code":"SOA",
    "total_staff_number":25,
    "budget":10000,
    "address":{"street":"Franz Josef Str",
               "bldg":4,
               "city":"Vienna",
               "country":"Austria"},
   "courses":[ {"code":"SOA101",
                "title":"Astronomy for Kids",
                "credits":3},
               {"code":"SOA201",
                "title":"Black Holes",
                "credits":6},
               {"code":"SOA301",
                "title":"Dark Matter",
                "credits":12}
             ]
  } );
```

validates well against the following JSON schema ...

Validation of nested BSON documents

JSON schema

```
db.createCollection("school", { "validator": { "$jsonSchema": {
"bsonType":"object",
"required":["name","code","total staff number","budget","address","courses"],
"properties":{ "name":{ "bsonType":"string",
                        "description":"Name of a department" },
               "code":{ "bsonType":"string",
                        "description":"Code of a department" },
               "total staff number":{ "bsonType":"double",
                                      "description" :"Total staff number" },
               "budget":{"bsonType":"double",
                         "description":"Budget of a department" },
               "address":{ "bsonType":"object",
                           "required":["street","bldg","city","country"],
                           "properties":{ "street":{"bsonType":"string",
                                                    "description": "Street name" },
                                          "bldg":{"bsonType":"double",
                                                  "description":"Building number" },
                                          "city":{"bsonType":"string",
                                                  "description":"City name" },
                                          "country":{"bsonType":"string",
                                                     "description":"Country name" } } },
               "courses":{ "bsonType":"array",
                           "items":{ "bsonType":"object",
                                     "required":["code","title","credits"],
                                     "properties":{"code":{"bsonType":"string",
                                                           "description":"Subject code"},
                                                   "title":{"bsonType":"string",
                                                            "description":"Subject title"},
                                                   "credits":{"bsonType":"double",
                                                              "description":"Total credit points"}}}}}});
```

References

JSON Schema

Understanding JSON Schema

MongoDB - JSON Schema validation

MongoDB - $jsonSchema operator