mongoDB.

Implement model

Mag. Thomas Griesmayer

Introduction

- Schema validation lets you create validation rules for your fields, such as allowed data types and value ranges.
- MongoDB uses a flexible schema model, which means that documents in a collection do not need to have the same fields or data types by default.
- When your application is in the early stages of development, schema validation may impose unhelpful restrictions because you don't know how you want to organize your data. Specifically, the fields in your collections may change over time.
- Once you've established an application schema, you can use schema validation to ensure there are no unintended schema changes or improper data types.

https://www.mongodb.com/docs/manual/core/schema-validation/

Introduction

- You can use schema validation in the following scenarios:
  - For a users collection, ensure that the password field is only stored as a string. This validation prevents users from saving their password as an unexpected data type, like an image.
  - For a sales collection, ensure that the item field belongs to a list of items that your store sells. This validation prevents a user from accidentally misspelling an item name when entering sales data.
  - For a students collection, ensure that the gpa field is always a positive number. This validation catches typos during data entry.

https://www.mongodb.com/docs/manual/core/schema-validation/

Introduction

- When you create a new collection with schema validation, MongoDB checks validation during updates and inserts in that collection.
- When you add validation to an existing, non-empty collection:
  - Newly inserted documents are checked for validation.
  - Documents already existing in your collection are not checked for validation until they are modified.
  - Specific behavior for existing documents depends on your chosen validation level.
- Failed validation:
  - By default, when an insert or update operation would result in an invalid document, MongoDB rejects the operation and does not write the document to the collection.
  - Alternatively, you can configure MongoDB to allow invalid documents and log warnings when schema violations occur.

https://www.mongodb.com/docs/manual/core/schema-validation/

```
db.createCollection("students", {
   validator: {
      $jsonSchema: {
         bsonType: "object",
         required: [ "name", "age", "email" ],
         properties: {
            name: {
               bsonType: "string",
               description: "First- and lastname is required!"
            },
            age: {
               bsonType: "int",
               minimum: 15,
               maximum: 150,
               description: "The age ist required (15..150)!"
            },
```

```
            email: {
                bsonType: "string",
                pattern: "^[A-Za-z0-9._]+@[A-Za-z0-9.-]+\.[A-Za-z]
                          {2,4}$",
                description: "Avalid email adress is required"
            },
              faculty: {
                enum: [ "Informatics", "Physics",
                        "Mathematics", null ],
                description: "Use a valid faculty (Informatics,
                              Physics, Mathematics or null)!"
              }
           }
        }
    }
})
```

```
db.students.insertOne({ "name": "Thomas Griesmayer" })
db.students.insertOne({ "name": "Thomas Griesmayer", "age": -3 })
db.students.insertOne({ "name": "Thomas Griesmayer", "age": 19 })
db.students.insertOne({ "name": "Thomas Griesmayer", "age": 19,
                        "email": "thomas@demo" })
db.students.insertOne({ "name": "Thomas Griesmayer", "age": 19,
                        "email": "thomas@demo.com" })
```

```
db.createCollection("lectures", {
   validator: {
      $jsonSchema: {
         bsonType: "object",
         required: [ "name", "semester" ],
         properties: {
            name: {
               bsonType: "string",
               description: "Name of the lecture is required!"
            },
```

```
properties: {
   semesterid: {
      bsonType: "int",
      minimum: 1,
      maximum: 8,
      description: "The semester ist
                     required (1..8)!"
},
semestername: {
      bsonType: "string",
   pattern: "^[WS]S[0-9]{4}$",
      description: "The year and semester
               of the lecture is not required!"
},
```

```
      classname: {
            bsonType: "string",
            description: "The class of the lecture!"
      },
         faculty: {
            enum: [ "Informatics", "Physics",
                    "Mathematics" ],
            description: "Use a valid faculty
                          (Informatics, Physics or
                          Mathematics)!"
         }
   }
},
```

```
            grades: {
               bsonType: "array",
               description: "Minimum 10 Students are required",
               minItems: 10,
               items: {
                 bsonType: "object",
                 description: "The students and their grades!",
                 required: [ "student_id", "grade" ],
                 properties: {
                    student_id: {
                       description: "The id of the student!"
                    },
                    grade: {
                       enum: [ "1", "2", "3", "4", "5",
                           "nicht Beurteilt", "befreit", null ],
                       description: "The grade of the student!"
                    }
})
```

```
db.lectures.createIndex({ "grades.student_id":1 }, {})

db.lectures.insertOne({ name: "Introduction Java",
                        semester: { semesterid: 1 } })
db.lectures.insertOne({ name: "Introduction Oracle",
                        semester: { semesterid: 1,
                        semestername: "WS2023",
                        classname: "3AKIF",
                        faculty: "Informatics" } })
```

```
db.lectures.insertOne({ name: "Introduction Oracle",
   semester: { semesterid: 1, semestername: "WS2023",
   classname: "3AKIF", faculty: "Informatics" },
   grades: [ {student_id: "1",  grade: "4"},
             {student_id: "2",  grade: "2"},
             {student_id: "3",  grade: "1"},
             {student_id: "4",  grade: "5"},
             {student_id: "5",  grade: "nicht Beurteilt"},
             {student_id: "6",  grade: "befreit"},
             {student_id: "7",  grade: "1"},
             {student_id: "8",  grade: "2"},
             {student_id: "9",  grade: "1"},
             {student_id: "10", grade: "3"},
             {student_id: "11", grade: "2"} ] })
```