02 Esquemas Schema Validation

Specify Validation Rules

Validation rules are on a per-collection basis.

To specify validation rules when creating a new collection, use db.createCollection() with the validator option.

To add document validation to an existing collection, use collMod command with the validator option.

MongoDB also provides the following related options:

- validationLevel option, which determines how strictly MongoDB applies validation rules to existing documents during an update, and
- validationAction option, which determines whether MongoDB should error and reject documents that violate the validation rules or warn about the violations in the log but allow invalid documents.

JSON Schema

Starting in version 3.6, MongoDB supports JSON Schema validation. To specify JSON Schema validation, use the $jsonSchema operator in your validator expression.

NOTE JSON Schema is the recommended means of performing schema validation.

Práctica

```
> db.createCollection("pacientes",
...    { validator: {
...       $jsonSchema: {
...          bsonType: "object",
...          required: ["nombre", "apellidos", "fechaNac", "direccion"],
...          properties: {
...             nombre: {
...                bsonType: "string",
...                description: "debe ser un string y es obligatorio"
...             },
...             apellidos: {
...                bsonType: "string",
```

```
...                 description: "debe ser un string y es obligatorio"
...             },
...             fechaNac: {
...                 bsonType: "date",
...                 description: "debe ser date y es obligatorio"
...             },
...             dirección: {
...                 bsonType: "object",
...                 required: ["calle", "localidad"],
...                 properties: {
...                     calle: {
...                         bsonType: "string",
...                         description: "debe ser un string y es obligatorio"
...                     },
...                     cp: {
...                         bsonType: "string",
...                         description: "debe ser un string"
...                     },
...                     localidad: {
...                         bsonType: "string",
...                         description: "debe ser un string y es obligatorio"
...                     }
...                 }
...             }
...         }
...     }
... })

{ "ok" : 1 }

> db.pacientes.insert({a:1})
WriteResult({
        "nInserted" : 0,
        "writeError" : {
                "code" : 121,
                "errmsg" : "Document failed validation"
        }
})

> db.pacientes.insert({nombre: "Juan", apellidos: "Pérez", fechaNac:
ISODate("1973-11-18"), direccion: {calle: "Gran Via", localidad: "Madrid"}})
WriteResult({ "nInserted" : 1 })
```

Existing Documents

The validationLevel option determines which operations MongoDB applies the validation rules:

- If the validationLevel is strict (the default), MongoDB applies validation rules to all inserts and updates.
- If the validationLevel is moderate, MongoDB applies validation rules to inserts and to updates to existing documents that already fulfill the validation criteria. With the moderate level, updates to existing documents that do not fulfill the validation criteria are not checked for validity.

Práctica

```
> db.runCommand({
... collMod: "pacientes",
... validator: {
...         $jsonSchema: {
...            bsonType: "object",
...            required: ["nombre", "apellidos", "fechaNac", "direccion"],
...            properties: {
...               nombre: {
...                  bsonType: "string",
...                  description: "debe ser un string y es obligatorio"
...               },
...               apellidos: {
...                  bsonType: "string",
...                  description: "debe ser un string y es obligatorio"
...               },
...               fechaNac: {
...                  bsonType: "date",
...                  description: "debe ser date y es obligatorio"
...               },
...               dirección: {
...                  bsonType: "object",
...                  required: ["calle", "localidad"],
...                  properties: {
...                     calle: {
...                        bsonType: "string",
...                        description: "debe ser un string y es obligatorio"
...                     },
...                     cp: {
...                        bsonType: "string",
...                        description: "debe ser un string"
```

```
...                     },
...                     localidad: {
...                         bsonType: "string",
...                         description: "debe ser un string y es obligatorio"
...                     }
...                 }
...             }
...         }
...     }
... , validationLevel: "moderate"
... })
{ "ok" : 1 }

> db.pacientes.insert({nombre: "Pepe"})
WriteResult({
        "nInserted" : 0,
        "writeError" : {
                "code" : 121,
                "errmsg" : "Document failed validation"
        }
})

> db.pacientes.update({_id: ObjectId("5e3078860a4428bb0b3d74eb")},{$set:
{nombre: "Juan Pedro"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

Accept or Reject Invalid Documents

The validationAction option determines how MongoDB handles documents that violate the validation rules:

- If the validationAction is error (the default), MongoDB rejects any insert or update that violates the validation criteria.
- If the validationAction is warn, MongoDB logs any violations but allows the insertion or update to proceed.

Práctica

```
...     }
... , validationAction: "warn"
... })
```

See collection schema validation

```
> db.getCollectionInfos( {name: "pacientes"} )
```