

# MongoDB - Schema Validation

Topics : [MongoDB](#)

Written on [December 30, 2023](#)

In MongoDB, schema validation allows you to enforce specific rules on the structure and content of documents within a collection. While MongoDB is schema-less, schema validation provides a way to ensure that documents meet certain criteria. Validation rules are defined using JSON Schema.

Here's an overview of schema validation in MongoDB:

## Basic Schema Validation:

### 1. Create a Validation Rule:

- Define a JSON Schema that describes the expected structure of documents.

```
{
  $jsonSchema: {
    bsonType: "object",
    required: ["name", "email"],
    properties: {
      name: {
        bsonType: "string",
        description: "must be a string and is required"
      },
      email: {
        bsonType: "string",
        pattern: "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$",
        description: "must be a string and match the regular expression pattern"
      }
    }
  }
}
```

### 2. Apply Validation Rule to a Collection:

- Use the collMod command to apply the validation rule to a collection.

```
db.runCommand({
  collMod: "users",
  validator: {
    $jsonSchema: {
```

```

bsonType: "object",
required: ["name", "email"],
properties: {
  name: {
    bsonType: "string",
    description: "must be a string and is required"
  },
  email: {
    bsonType: "string",
    pattern: "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$",
    description: "must be a string and match the regular expression pattern"
  }
}
},
validationLevel: "moderate", // Options: "off", "strict", "moderate"
validationAction: "error" // Options: "error", "warn"
});

```

## Validation Level and Action:

- **Validation Level:**

- "off": No validation (default).
- "strict": Apply strict validation rules.
- "moderate": Apply rules, but allow existing documents that do not meet the validation rules.

- **Validation Action:**

- "error": Reject documents that do not meet validation rules.
- "warn": Log a warning but allow documents that do not meet validation rules.

## Example:

```

db.createCollection("users", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["name", "email"],
      properties: {
        name: {
          bsonType: "string",
          description: "must be a string and is required"
        },
        email: {
          bsonType: "string",
          pattern: "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$",
          description: "must be a string and match the regular expression pattern"
        }
      }
    }
  }
});

```

```
}  
}  
}  
});
```

In this example, the users collection is created with a validation rule that requires documents to have name and email fields, and the email field must match a specific pattern.

© Copyright **Aryatechno**. All Rights Reserved. Written tutorials and materials by [Aryatechno](#)

ARYATECHNO