Database

Recommended GUI client: MongoDB Compass

Development Accounts			
Admin Account		**Student Account**	
User Name	admin2	User Name	XuHan
Password	admin2	Password	123456
_id	60532e8d7ba3514b576ebb23	_id	6083ac2868d7c908dc575bdc
is_moderator	true	is_moderator	false
is_admin	true	is_admin	false
student_number	845462	student_number	123456
__v	11	__v	0

Database Schema

The database schema can be found here.

DB schema validation rules

For safety and security reasons, the schema validation is used here to ensure all the documents in a collection follow a defined set of rules, such as specifying the mandatory fields, the type of a field, the unique key, or the length limit for a value. The database will give an error when a certain condition is not met.

The rules for each collection can be found below. The schema validation script for each collection can be found in DB schema validation script.

Users

Unique key	student_number, account
Mandatory fields	'account', 'student_number', 'password', 'is_moderator', 'is_admin'
Properties	student_number: minimum length: 6, maximum length: 8 password: minimum: 6

Tools

Unique key	name
Mandatory fields	'name'

Subjects

Unique key	name, subject_code
Mandatory fields	'name', 'subject_code', 'created_by', 'description'

Sections

Unique key	None
Mandatory fields	'name','owner','type','articles'

Pending articles

Unique key	None
Mandatory fields	'subjects', 'title', 'editor_id', 'content'

Comments

Unique key	None
Mandatory fields	'content', 'author_id', 'root_comment_id', 'section_id', 'author_name'

Articles

Unique key	title
Mandatory fields	'content', 'title', 'author_id', 'subjects', 'is_pending'

DB schema validation script

For safety and security reasons, the schema validation is used here to ensure all the documents in a collection follow a defined set of rules, such as specifying the mandatory fields, the type of a field, the unique key, or the length limit for a value. The database will give an error when a certain condition is not met.

The schema validation rules can be found in DB schema validation rules.

On this page, 2 ways of adding the schema validation are introduced, which are adding schema validation **using MongoDB Compass** and **through MongoDB shell script**. Both scripts for each collection can be found below.

Using MongoDB Compass

For each collection, there is a validation tap. You can click the "Add Rule" button to add the validation script.

Users

```
{
$jsonSchema: {
required: [
'account',
'student_number',
'password',
'is_moderator',
'is_admin'
],
properties: {
student_number: {
minLength: 6,
maxLength: 8,
description: 'student number has to be 6 to 8 digits'
},
password: {
minimum: 6,
description: 'password needs to be at least 6 characters'
}
}

}
}
```

Tools

```
{
$jsonSchema: {
required: [
'name'
]
}
}
```

Subjects

```
{
$jsonSchema: {
required: [
'name',
'subject_code',
'created_by',
'description'
]
}
}
```

Sections

```
{
$jsonSchema: {
required: [
'name',
'owner',
'type',
'articles'
]
}
}
```

Pending articles

```
{
$jsonSchema: {
required: [
'subjects',
'title',
'editor_id',
'content'
]
}
}
```

Comments

```
{
$jsonSchema: {
required: [
'content',
'author_id',
'root_comment_id',
'section_id',
'author_name'
]
}
}
```

Articles

```
{
$jsonSchema: {
required: [
'content',
'title',
'author_id',
'subjects',
'is_pending'
]
}
}
```

Through MongoDB shell

For each collection, enter the script in the shell.

Users

```
db.runCommand({collMod: "users",validator:{$jsonSchema: {required: [
'account',
'student_number',
'password',
'is_moderator',
'is_admin'
],
properties: {
student_number: {
minLength: 6,
maxLength: 8,
description: 'student number has to be 6 to 8 digits'
},
password: {
minimum: 6,
description: 'password needs to be at least 6 characters'
}
}
}}}
)
```

Tools

```
db.runCommand({collMod: "tools",validator:{$jsonSchema: {required: [
'name'
]}

}})
```

Subjects

```
db.runCommand({collMod: "subjects",validator:{$jsonSchema: {required: [
'name',
'subject_code',
'created_by',
'description'
]}

}})
```

Sections

```
db.runCommand({collMod: "sections",validator:{$jsonSchema: {required: [
'name',
'owner',
'type',
'articles'
]}

}})
```

Pending articles

```
db.runCommand({collMod: "pendingarticles",validator:{$jsonSchema: {required: [
'subjects',
'title',
'editor_id',
'content'
]}

}})
```

Comments

```
db.runCommand({collMod: "comments",validator:{$jsonSchema: {required: [
'content',
'author_id',
'root_comment_id',
'section_id',
'author_name'
]}

}})
```

Articles

```
db.runCommand({collMod: "articles",validator:{$jsonSchema: {required: [
'content',
'author_id',
'root_comment_id',
'section_id',
'author_name'
]}

}})
```

MongoDataBase

MongoDB is a database based on distributed file storage. Written by C + +.

It aims to provide scalable high-performance data storage solutions for web applications.

MongoDB is a product between a relational database and a non-relational database, which has the most abundant functions and is most like a relational database.

MongoDB	
Address	mongodb+srv://root:root@cluster0.uhlwx.mongodb.net/myFirstDatabase?authSource=admin&replicaSet=atlas-hv89z3-shard-0&w=majority&readPreference=primary&appname=MongoDB%20Compass&retryWrites=true&ssl=true
Created	19 Mar 2021
Status	ACTIVE (Updated: 24 Apr 2021)
MongoDB Version	4.4.5
Host Provider	MongoDB Cloud
Location	AWS / Sydney (ap-southeast-2)
Contact	Runfeng Du

Reference list

https://www.runoob.com/mongodb/mongodb-tutorial.html

Redis

Remote dictionary server (redis) is a key value storage system written by Salvatore Sanfilippo.

It is a cross platform non relational database.

Redis is an open source key value database written in ANSI C language, complying with BSD protocol,

supporting network, memory based, distributed, optional persistence, and provides multiple language API.

Redis is usually called data structure server, because value can be string, hash, list, sets and sorted sets.

Redis	
Address	localhost:6379
Created	21 Apr 2021
Status	FINISH 25 May 2021
Redis Version	Windows x64 5.0.10
Contact	Xu Han

Installation of Ubuntu apt command

To install redis on the Ubuntu system, you can use the following commands:

```
# sudo apt update
# sudo apt install redis-server
```

Start redis

```
# redis-server
```

Check whether redis is started

```
# redis-cli
```

The above command will open the following terminals

```
redis 127.0.0.1:6379>
```

127.0.0.1 is the local IP and 6379 is the redis service port. Now we enter the ping command

```
redis 127.0.0.1:6379> ping
PONG
```

The above shows that redis has been successfully installed.

Redis configuration

The redis configuration file is located in the redis installation directory, and the file name is **redis.conf**.
You can view or set configuration items through the config command.

The redis config command format is as follows

```
redis 127.0.0.1:6379> CONFIG GET CONFIG_SETTING_NAME
```

More information

About redis
Redis is completely open source and complies with BSD protocol. It is a high-performance key value database.
Redis and other key - value caching products have the following three characteristics:
Redis supports data persistence. It can save the data in memory in disk and load it again when it is restarted.
Redis not only supports simple key value data, but also provides list, set, Zset, hash and other data structure storage.
Redis supports data backup, that is, data backup in master slave mode.

Advantages of redis
Extremely high performance - redis can read 110000 times / s and write 81000 times / s.
Rich data types - redis supports string, lists, hashes, sets and ordered sets data type operations of binary cases.
Atom – all operations of redis are atomic, meaning that they are either executed successfully or not executed at all. Individual operations are atomic.
Multiple operations also support transactions, i.e. atomicity, wrapped by multi and exec instructions.
Rich features - redis also supports publish / subscribe, notification, key expiration and other features.

Differences between redis and other key value storage
Redis has a more complex data structure and provides atomic operations on them, which is a different evolutionary path from other databases. Redis's data types are based on the basic data structure and transparent to programmers, without additional abstraction.
Redis runs in memory, but it can persist to disk. Therefore, when reading and writing different data sets at high speed, we need to weigh the memory, because the amount of data cannot be greater than the hardware memory. Another advantage of in memory database is that compared with the same complex data structure on disk, it is very simple to operate in memory. In this way, redis can do many things with strong internal complexity. At the same time, in terms of disk format, they are compact and generated by appending, because they do not need random access.

References list:

https://www.runoob.com/redis/redis-tutorial.html

https://www.runoob.com/redis/redis-install.html

https://www.runoob.com/redis/redis-conf.html

https://www.runoob.com/redis/redis-intro.html