ЛР2. MongoDB. Аутентификация. Управление привилегиями

Для противодействия несанкционированному доступу в большинстве современных СУБД реализована многоуровневая система обеспечения безопасности, включающая три процедуры:

- **идентификация** – назначение пользователю (процессу) уникального имени;
- **аутентификация** – процедура проверки подлинности пользователя, представившего свой идентификатор;
- **авторизация** – процедура предоставления пользователю определенных ресурсов и прав на их использование

Прежде чем пользователь будет авторизован на доступ к определенным данным и на выполнение определенных действий в базе данных, необходимо, чтобы он получил доступ к экземпляру СУБД, что в свою очередь требует, чтобы пользователь был аутентифицирован. В случае успешной аутентификации для данного пользователя устанавливается соединение, и все действия и запросы к данным, выполняемые в рамках этого соединения, контролируются системой авторизации в соответствии с привилегиями, определенными для этого пользователя. В случае, если аутентификация не пройдена, соединение не будет установлено.

Аутентификация в MongoDB

Версия MongoDB Community поддерживает ряд механизмов аутентификации:

- SCRAM (по умолчанию) для аутентификации пользователей по паролю. Поддерживаются метод SCRAM-SHA-1, использующий хеш-функцию SHA-1, и SCRAM-SHA-256, использующий хеш-функцию SHA-256.
- Аутентификация на основе сертификата x.509 (вместо имени пользователя и пароля). Цифровой сертификат x.509 использует общепринятый стандарт инфраструктуры открытых ключей (PKI) x.509 для проверки того, что открытый ключ принадлежит тому, кто его предъявляет.

Основные правила настройки аутентификации в MongoDB:

- Пользователь в MongoDB создается в определенной базе данных. Эта БД является базой данных аутентификации пользователя; для этой цели можно использовать любую базу данных. База данных имени пользователя и аутентификации служит уникальным идентификатором пользователя.

Однако привилегии пользователя не ограничиваются базой данных аутентификации.

- Сначала создается пользователь – администратор, а затем создаются дополнительные пользователи в БД для каждого пользователя/приложения, которое обращается к системе.
- Как и в любых других информационных системах, рекомендуется следовать принципу наименьших привилегий, т.е. сначала создать роли, определяющие точные права доступа, требуемые набором пользователей. Затем создать пользователей и назначить им только те роли, которые необходимы для выполнения их операций.

В MongoDB аутентификация и авторизация не активируются по умолчанию. MongoDB также не создает пользователя с правами администратора по умолчанию при активации аутентификации и авторизации, независимо от того, какой режим аутентификации используется. Его необходимо создать вручную.

Для создания пользователя используется метод `db.createUser()` со следующими основными параметрами:

- `user` – имя пользователя;
- `pwd` – пароль;
- `roles` – роли, назначенные пользователю.

Для настройки SCRAM-аутентификации для сервера, запущенного без режима репликации, необходимо выполнить следующие шаги:

1) Запустить экземпляр сервера MongoDB без контроля доступа и выполнить подключение к серверу. Далее необходимо создать пользователя-администратора в БД `admin` с ролями `userAdminAnyDatabase` и `readWriteAnyDatabase`:

```
use admin
db.createUser( {
    user: "myUserAdmin",
    pwd: passwordPrompt(), // или указать пароль
    roles: [
  { role: "userAdminAnyDatabase", db: "admin" },
  { role: "readWriteAnyDatabase", db: "admin" }
    ]
} )
```

Роль `userAdminAnyDatabase` позволяет этому пользователю:

- создавать пользователей;
- предоставлять или отзывать роли у пользователей;
- создавать или изменять роли.

При необходимости можно назначить пользователю дополнительные встроенные или пользовательские роли.

База данных, в которой создается пользователь (в примере – `admin`), является базой данных аутентификации пользователя, в которой ему необходимо пройти аутентификацию. База данных аутентификации пользователя не ограничивает привилегии пользователя.

2) После создания пользователя-администратора необходимо перезапустить экземпляр MongoDB в режиме контроля доступа:

- если экземпляр mongod запускается из командной строки, необходимо добавить параметр `–auth`;
- если экземпляр `mongod` запускается с помощью файла конфигурации, в том числе как служба в Windows, в файл конфигурации необходимо добавить параметр `security.authorization`.

```
security:
    authorization: enabled
```

После запуска сервера в режиме контроля допуска для аутентификации необходимо либо указать параметры при подключении, либо выполнять аутентификацию после подключения.

В случае явной авторизации пользователя после подключения необходимо выбрать БД аутентификации и вызвать метод `auth` с указанием имени пользователя и пароля:

```
use admin
db.auth("myUserAdmin", passwordPrompt()) // или указать пароль
```

3) После аутентификации в качестве администратора методом `db.createUser()` можно создавать дополнительных пользователей.

Для получения списка пользователей из выбранной БД можно воспользоваться методом `db.getUsers( )`. Для получения указанного пользователя – командой `db.getUser(user_name)`.

Привилегии MongoDB

MongoDB поддерживает ролевую модель управления доступом, предоставляет доступ к данным и командам посредством авторизации на основе ролей, поддерживает как встроенные роли, обеспечивающие различные уровни доступа, обычно необходимые в СУБД, так и определяемые пользователем роли.

Роль предоставляет привилегии для выполнения указанных действий над ресурсом. Каждая привилегия либо явно указывается в роли, либо наследуется от другой роли, либо и то, и другое. Привилегия состоит из указанного ресурса и действий, разрешенных для ресурса. Ресурс — это база данных, коллекция, набор коллекций или кластер. Если ресурсом является кластер, выполняемые действия влияют на состояние системы, а не на конкретную базу данных или коллекцию.

Роль может включать в свое определение одну или несколько существующих ролей, и в этом случае роль наследует все привилегии включенных ролей.

Для просмотра привилегий для роли можно выполнить команду `rolesInfo` с указанием имени роли и базы данных, а также указанием параметров `showPrivilegies` и `showBuiltinRoles` для отображения привилегий и встроенных ролей.

```
db.runCommand(
   { rolesInfo : { role: <name>, db: <db> },
     showPrivileges: true,
     showBuiltinRoles: true
   })
```

MongoDB предоставляет встроенные пользовательские роли и роли администратора базы данных в каждой БД. Все остальные встроенные роли доступны только в базе данных `admin`.

Встроенные роли пользователя БД:

- `read` – предоставляет возможность чтения данных для всех несистемных коллекций и коллекции `system.js`, разрешая выполнять следующие действия: `find`, `dbStats`, `dbHash`, `listCollections`, `listIndexes` и др.
- `readWrite` – предоставляет все привилегии роли `read`, а также возможность изменять данные, выполняя следующие действия над

коллекциями: `createCollection`, `dropCollection`, `createIndex`, `dropIndex`, `insert`, `update`, `remove` и др.

Встроенные роли администратора БД:

- `dbAdmin` – предоставляет возможность выполнять административные задачи, связанные с управлением схемой, индексированием и сбором статистики.
- `dbOwner` – владелец базы данных, который может выполнять любые административные действия с базой данных. Эта роль сочетает в себе привилегии, предоставляемые ролями `readWrite`, `dbAdmin` и `userAdmin`.
- `userAdmin` – предоставляет возможность создавать и изменять роли и пользователей в текущей базе данных.

Встроенные роли администратора кластера:

- `clusterAdmin` – обеспечивает максимальный доступ к управлению кластером.
- `clusterManager` – обеспечивает действия по управлению и мониторингу в кластере.
- `clusterMonitor` – предоставляет доступ только для чтения к инструментам мониторинга.
- `hostManager` – предоставляет возможность мониторинга и управления серверами.
- `backup` – предоставляет минимальные привилегии, необходимые для резервного копирования данных.
- `restore` – предоставляет необходимые привилегии для восстановления данных из резервных копий.

Если существующие роли не могут описать требуемый набор привилегий, можно создавать новые роли. MongoDB поддерживает набор методов для управления ролями:

Оператор	**Описание**
`db.createRole()`	Создает роль и указывает ее привилегии.
`db.dropRole()`	Удаляет пользовательскую роль.
`db.dropAllRoles()`	Удаляет все пользовательские роли, связанные с

	базой данных.
db.getRole()	Возвращает информацию об указанной роли.
db.getRoles()	Возвращает информацию обо всех пользовательских ролях в базе данных.
db.grantPrivilegesToRole()	Назначает привилегии пользовательской роли.
db.revokePrivilegesFrom Role()	Удаляет указанные привилегии из пользовательской роли.
db.grantRolesToUser()	Назначает роли пользователю.
db.revokeRolesFromUser()	Снимает роли с пользователя.
db.updateRole()	Обновляет пользовательскую роль.

Например, метод db.createRole() в качестве параметра принимает документ следующего вида:

```
{
  role: "<name>",
  privileges: [
     { resource: { <resource> }, actions: [ "<action>",
... ] },
     ...
  ],
  roles: [
     { role: "<role>", db: "<database>" } | "<role>",
      ...
  ],
  …
}
```

Для роли можно определить, как явные привилегии (поле privilegies), так и неявно унаследовать привилегии от существующих ролей (поле roles). Пример создания роли имеет следующий вид:

```
db.createRole(
   {
     role: "myClusterwideAdmin",
     privileges: [
```

```
       { resource: { cluster: true }, actions: [
"addShard" ] },
       { resource: { db: "config", collection: "" },
actions: [ "find", "update", "insert", "remove" ] },
       { resource: { db: "users", collection:
"usersCollection" }, actions: [ "update", "insert",
"remove" ] },
       { resource: { db: "", collection: "" }, actions: [
"find" ] }
     ],
     roles: [
       { role: "read", db: "admin" }
     ]
   }
)
```

ЛР2. Вопросы для контроля

1. Компоненты (процедуры) системы управления доступом.
2. Управление доступом. Ключевые понятия. Принципы политики управления доступом.
3. Модели управления доступом.
4. Аутентификация в MongoDB.
5. Ролевая модель в MongoDB.
6. Встроенные роли в MongoDB.

ЛР2. Задания

1. Настроить SCRAM-аутентификацию для сервера MongoDB.
2. Создать в выбранной БД (из ЛР1) следующих пользователей, использующих встроенные роли:
    a. `owner` – может выполнять любые действия с БД.
    b. `reader` – имеет права на чтение данных из всех коллекций.
    c. `editor` – имеет права на чтения и запись данных из всех коллекций.
    d. `view_reader` – права отсутствуют.

    Продемонстрировать выполнение запросов на чтение и запись данных от каждого пользователя.
3. Создать представление `filmStats`, возвращающее агрегированные характеристики фильмов из коллекции `films` по категории: количество фильмов в категории, средняя продолжительность фильма.
4. Создать в выбранной БД следующие пользовательские роли:
    a. `view_read` – позволяет читать данные из представления `filmStats`.
    b. `collection_edit` – позволяет редактировать (добавлять / изменять / удалять) данные в коллекции `films`.
5. Изменить права пользователя `view_reader`, назначив ему роль `view_read`. Показать назначенные пользователю роли. Продемонстрировать изменения, выполнив запросы на чтение данных из представления.
6. Изменить права пользователя `editor`, оставив ему права на чтение данных из всех коллекций и на редактирование данных в коллекции `films` (с помощью роли `collection_edit`). Продемонстрировать изменения, выполнив запросы модификации данных.