D1F Praktischer Nachweis

Erstelle eine Rolle dataEditor, die für die Datenbank productdb in der Collection inventory nur Lese- und Schreibrechte besitzt (kein Löschen). Erstelle anschliessend den Benutzer editorUser, weise ihm diese Rolle zu und überprüfe, dass:

- Lesen ✅
- Schreiben ✅
- Löschen ❌ nicht erlaubt ist

Vorbereitung

Docker Container:

```
services:
  mongodb:
    image: mongo:8.0
    container_name: editor_mongo
    command: ["mongod", "--auth"]
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: example
    ports:
      - "27018:27017"
    volumes:
      - mongo_data:/data/db
      - ./products.csv:/tmp/products.csv:ro
    restart: unless-stopped

volumes:
  mongo_data:
```

Daten (CSV):

```
product,price,quantity
Laptop,1200,5
Mouse,25,50
Keyboard,45,30
```

Import:

```
docker exec editor_mongo mongoimport `
  --username root `
  --password example `
```

```
  --authenticationDatabase admin `
  --db productdb `
  --collection inventory `
  --type csv `
  --headerline `
  --file /tmp/products.csv
```

Ausführung

1. Rolle erstellen:

Öffne die Mongo-Shell:

```
docker exec -it editor_mongo mongosh -u root -p example --authenticationD
```

```
# docker exec -it test_mongo mongosh -u root -p example --authenticationDatabase admin
Current Mongosh Log ID: 684ac8225bca9f1438d861df
Connecting to:          mongodb://<credentials>@127.0.0.1:27017/?directConnection=true&
serverSelectionTimeoutMS=2000&authSource=admin&appName=mongosh+2.5.0
Using MongoDB:          8.0.9
Using Mongosh:          2.5.0
```

Dann in der productdb:

```
use productdb
```

```
db.createRole({
  role: "dataEditor",
  privileges: [
    {
      resource: { db: "productdb", collection: "inventory" },
      actions: ["find", "insert", "update"]
    }
  ],
  roles: []
});
```

```
productsdb> use productdb
switched to db productdb
productdb> db.createRole({
...   role: "dataEditor",
...   privileges: [
...     {
...       resource: { db: "productdb", collection: "inventory" },
...       actions: ["find", "insert", "update"]
...     }
...   ],
...   roles: []
```

```
... });
{ ok: 1 }
```

2. Benutzer mit dieser Rolle erstellen

```
db.createUser({
  user: "editorUser",
  pwd: "editpass123",
  roles: [{ role: "dataEditor", db: "productdb" }]
});
```

```
productdb> db.createUser({
...     user: "editorUser",
...     pwd: "editpass123",
...     roles: [{ role: "dataEditor", db: "productdb" }]
... });
{ ok: 1 }
```

3. Test: Leserechte prüfen

```
docker exec -it editor_mongo mongosh `
  --username editorUser `
  --password editpass123 `
  --authenticationDatabase productdb
```

```
# docker exec -it test_mongo mongosh `
>   --username editorUser `
>   --password editpass123 `
>   --authenticationDatabase productdb
Current Mongosh Log ID: 684ac88b19b2990231d861df
Connecting to:          mongodb://<credentials>@127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&authSource=productdb&appName=mongosh+2.5.0
Using MongoDB:          8.0.9
Using Mongosh:          2.5.0

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/
```

Dann:

```
use productdb
db.inventory.find().pretty()
```

```
productdb> db.inventory.find().pretty()
[
  {
    _id: ObjectId('684ac7be87b2bcb1a377548a'),
    product: 'Mouse',
    price: 25,
    quantity: 50
  },
  {
    _id: ObjectId('684ac7be87b2bcb1a377548b'),
```

```
    product: 'Keyboard',
    price: 45,
    quantity: 30
  },
  {
    _id: ObjectId('684ac7be87b2bcb1a377548c'),
    product: 'Laptop',
    price: 1200,
    quantity: 5
  }
]
```

4. Test: Schreibrechte (Update & Insert)

Insert:

```
db.inventory.insertOne({
  product: "Monitor",
  price: 200,
  quantity: 10
});
```

```
productdb> db.inventory.insertOne({
...     product: "Monitor",
...     price: 200,
...     quantity: 10
... });
{
  acknowledged: true,
  insertedId: ObjectId('684aca8b19b2990231d861e0')
}
```

Update:

```
db.inventory.updateOne(
  { product: "Mouse" },
  { $set: { price: 30 } }
);
```

```
productdb> db.inventory.updateOne(
...     { product: "Mouse" },
...     { $set: { price: 30 } }
... );
{
  acknowledged: true,
```

```
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

5. Test: Löschversuch

```
db.inventory.deleteOne({ product: "Keyboard" });
```

```
productdb> db.inventory.deleteOne({ product: "Keyboard" });
MongoServerError[Unauthorized]: not authorized on productdb to execute command { delete
: "inventory", deletes: [ { q: { product: "Keyboard" }, limit: 1 } ], ordered: true, ls
id: { id: UUID("e5baf0b6-e038-4e9a-98d0-c051a48660e9") }, $db: "productdb" }
```

Zusammenfassung

Mit dieser Konfiguration wurde eine benutzerdefinierte Rolle erstellt, die gezielte Rechte auf eine Collection gewährt. Das verhindert unbeabsichtigte oder böswillige Löschoperationen – ein nützliches Prinzip für die Rechtevergabe in produktiven MongoDB-Systemen.