Scripte

- Datenbank, Tabelle, Datensätze erstellen, löschen , Datenbank kompirmieren

Datenbank, Tabelle, Datensätze erstellen, löschen , Datenbank kompirmieren

Beschreibung:

Wir erstellen drei Scripte zum verwalten einer Testdatenbank. Diese scripte können auch als Vorlage um Progragrammtechnisch die Datenbank mit Daten zu füllen.
Das letzte script db_free_space.py, kann überall benutzt werden um die Datenbankgröße zu optimieren.

Allerdings sollte das Script ausgeführt wrerden wenn keiner drin arbeitet.
Währenddessen sind Schreibzugriffe blockiert.
Sollte man wissen!

- Eins das die Test-Datenbank und Test-Tabelle erstellt : `create_db.sh`
- Eins das die Datenbank um 100 MB pro Aufruf des Scriptes anwachsen lässt : `fill_data.py` Dieses ist ein python Script
- Eins das die Tabelle wieder leert : delete_all.py Dieses ist auch ein python Script
- Ein Script um den Speicher nach dem löschen wieder frei zu geben : db_free_space.py

Dazu muss python3 und die python mongo erweiterung installiert sein

```
apt install python3 python3-pymongo
```

Inhalt create_db.sh

```
#!/bin/bash


# Datenbank und Sammlung erstellen
```

```
mongosh <<EOF
use meineTestDatenbank
db.createCollection("testTabelle")
EOF

echo "Datenbank und Testtabelle erstellt."
```

Ausgabe:

```
MongoDB shell version v5.0.25
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("559ca2a0-399c-412a-9ad9-5120ff4be092") }
MongoDB server version: 5.0.25
================
switched to db meineTestDatenbank
{ "ok" : 1 }
bye
Datenbank und Testtabelle erstellt.
```

Inhalt `fill_data.sh`

Python script, mit der Anzahl der Dokumente spielen um die richtige Beleggröße zu bekommen. Dieses Script schreibt ungefähr 400 kb dazu.

```
from pymongo import MongoClient, InsertOne
import random

# MongoDB-Client konfigurieren
client = MongoClient('localhost', 27017)

# Datenbank und Sammlung auswählen
db = client['meineTestDatenbank']
collection = db['testTabelle']

# Anzahl der Dokumente, die eingefügt werden sollen
num_docs = 10000

# Liste für Bulk-Insert-Operationen vorbereiten
operations = []
```

```
for i in range(num_docs):
    doc = {
        "name": f"Testname {i}",
        "beschreibung": f"Dies ist eine Testbeschreibung mit einer zufälligen Zahl: {random.randint(1, 10000)}",
        "nummer": i
    }
    # Füge die InsertOne-Operation zur Liste hinzu
    operations.append(InsertOne(doc))

    # Führe Bulk-Insert in Chargen aus, um Speicherprobleme zu vermeiden
    if len(operations) == 1000:
        collection.bulk_write(operations)
        operations.clear()  # Liste nach dem Bulk-Insert leeren
        print(f"{i+1} Dokumente eingefügt...")

# Restliche Dokumente einfügen
if operations:
    collection.bulk_write(operations)
    print(f"Die letzten {len(operations)} Dokumente wurden eingefügt.")

print(f"Es wurden insgesamt {num_docs} Dokumente zur 'testTabelle' in 'meineTestDatenbank' hinzugefügt.")
```

Speicher nach dem einfügen (das script wurde schon mehrmals laufen gelassen deshalb 2 MB Größe)

```
test> show databases
admin               40.00 KiB
config             108.00 KiB
local               40.00 KiB
meineTestDatenbank    2.74 MiB
```

Löschen aller eintrage delete_all.py

```
from pymongo import MongoClient

# MongoDB-Client konfigurieren
client = MongoClient('localhost', 27017)

# Datenbank und Sammlung auswählen
```

```
db = client['meineTestDatenbank']
collection = db['testTabelle']

# Alle Dokumente aus der Sammlung entfernen
result = collection.delete_many({})

# Ergebnis ausgeben
print(f"Anzahl der entfernten Dokumente: {result.deleted_count}")
```

Speicher nach dem löschen

```
test> show databases
admin               40.00 KiB
config             108.00 KiB
local               40.00 KiB
meineTestDatenbank    2.13 MiB
```

Speicherplatz freigeben

db_free_space.py

Inhalt

```
from pymongo import MongoClient

# Konfiguration
db_name = 'meineTestDatenbank'
collection_name = 'testTabelle'

# MongoDB-Client initialisieren
client = MongoClient('localhost', 27017)

# Zugriff auf die Datenbank und Sammlung
db = client[db_name]

# Führe den Compact-Befehl aus
try:
    print(f"Starte Compact-Vorgang für die Sammlung '{collection_name}' in der Datenbank '{db_name}'...")
    result = db.command({'compact': collection_name})
    print("Compact-Vorgang abgeschlossen.")
    print(result)
```

```
except Exception as e:
    print(f"Ein Fehler ist aufgetreten: {e}")

# Schließe die Verbindung
client.close()
```

Speicher nach dem Compacten:

```
test> show databases
admin               64.00 KiB
config              96.00 KiB
local               64.00 KiB
meineTestDatenbank  24.00 KiB
test>
```