ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ
Π.Μ.Σ. Ανάπτυξη Λογισμικού και Νέφος

Διαχείριση και Αναλυτική Δεδομένων στο Υπολογιστικό Νέφος

Geodatabases and Spatial Joins

ΕΥΣΤΑΘΙΟΣ ΙΩΣΗΦΙΔΗΣ
ΑΜ: mai25017

ΘΕΣΣΑΛΟΝΙΚΗ ΙΟΥΝΙΟΣ 2025

Ερώτημα 1

Τα ερωτήματα στην βάση με την χρήση του QGIS (όπως γίνεται η επίδειξη στο βίντεο) είναι τα εξής:

-- query1 Νομοί από τους οποίους περνά το δίκτυο του ΟΣΕ
select distinct n.*
from nomoi as n
join sidhrodromiko_diktyo as ose on st_intersects(n.geometry, ose.geometry);

-- query2 Νομοί με τους οποίους συνορεύει ο νομός Θεσσαλονίκης και έχουν στο έδαφός τους λίμνη
select distinct n2.*
from nomoi as n1
join nomoi as n2 on st_touches(n1.geometry, n2.geometry)
join limnes as l on st_intersects(n2.geometry, l.geometry)
where n1.name_gr like '%ΘΕΣΣ%' and n2.name_gr not like '%ΘΕΣΣ%';

-- query3 Νομοί που δεν έχουν αεροδρόμιο
select n.*
from nomoi as n
left join aerodromia as a on st_contains(n.geometry, a.geometry)
where a.geometry is null;

Βρίσκονται και διαθέσιμα στο αρχείο sql_queries.txt.

Ερώτημα 2

Τρόπος Εργασίας: Προετοιμασία Περιβάλλοντος και Εισαγωγή Δεδομένων

Για την εκτέλεση των χωρικών ερωτημάτων (spatial queries) και την ανάλυση των γεωχωρικών δεδομένων, ακολουθήθηκε μια συγκεκριμένη μεθοδολογία που περιλαμβάνει την προετοιμασία του περιβάλλοντος Docker, την εκκίνηση μιας MongoDB instance και την εισαγωγή των παρεχόμενων GeoJSON αρχείων.

Α. Προετοιμασία Περιβάλλοντος Docker & MongoDB

Τα προαπαιτούμενα για την υλοποίηση της παρούσας ενότητας είναι:

1. Εγκατεστημένο Docker Engine στον τοπικό υπολογιστή.
2. Τα γεωχωρικά δεδομένα σε μορφή GeoJSON Lines (aerodromia.geojsonl.json, iso_metadata.geojsonl.json, limnes.geojsonl.json,

nomoi.geojsonl.json, oikismoi.geojsonl.json, poleis.geojsonl.json, potamoi.geojsonl.json, sid_diktio.geojsonl.json τα οποία παρήχθησαν όπως ανέφεραν οι οδηγίες).

1. Δημιουργία και Εκτέλεση Docker Container για MongoDB

Η διαδικασία εκκίνησης ενός container MongoDB μέσω Docker περιγράφεται στα παρακάτω βήματα, τα οποία εκτελέστηκαν μέσω του τερματικού (terminal/Command Prompt/PowerShell):

α. Δημιουργία Docker Volume (Προαιρετικό αλλά Συνιστώμενο)
Για τη διασφάλιση της ανθεκτικότητας των δεδομένων (data persistence) της MongoDB, ακόμα και μετά τη διαγραφή ή τον τερματισμό του container, δημιουργήθηκε ένας Docker volume με την ονομασία mongo_data:

docker volume create mongo_data

β. Δημιουργία Φακέλου για τα GeoJSON Αρχεία στον Host Υπολογιστή
Δημιουργήθηκε ένας τοπικός φάκελος (π.χ., ~/data) όπου τοποθετήθηκαν όλα τα αρχεία .geojsonl.json. Στη συνέχεια, τροποποιήθηκαν τα δικαιώματα πρόσβασης στον φάκελο αυτό για να επιτρέπεται η ανάγνωση από το Docker container (σε συστήματα Linux/macOS):

chmod -R 755 data/

Μετά την ολοκλήρωση της εισαγωγής των δεδομένων, τα δικαιώματα μπορούν να επαναφερθούν σε πιο περιοριστικές τιμές, εάν επιθυμείτε:

chmod -R 644 data/

γ. Εκτέλεση του MongoDB Container
Το MongoDB container εκτελέστηκε στο παρασκήνιο (-d), με ονομασία my-mongo (--name my-mongo), αντιστοίχιση της προεπιλεγμένης θύρας της MongoDB (-p 27017:27017) και δύο προσαρτήσεις τόμων (volume mounts):

- Μία για τα δεδομένα της βάσης (-v mongo_data:/data/db), συνδέοντας το Docker volume που δημιουργήθηκε στο βήμα 1.α.
- Μία για τα GeoJSON αρχεία (-v ./data:/import_data:z), καθιστώντας τα προσβάσιμα εντός του container. Η διαδρομή ./data αντικαταστάθηκε με την πραγματική διαδρομή προς τον φάκελο που δημιουργήθηκε στο βήμα 1.β.

Η εντολή εκτέλεσης ήταν η ακόλουθη:

docker run -d --name my-mongo -p 27017:27017 -v mongo_data:/data/db -v ./data:/import_data:z mongo:latest

Σημειώσεις για την εντολή:

- mongo:latest: Χρησιμοποιήθηκε η τελευταία διαθέσιμη έκδοση (image) της MongoDB.
- Το flag :z: Υποδεικνύει στο Docker να επανα-ετικετοποιήσει (relabel) τον host φάκελο για κοινόχρηστη χρήση με το container, κυρίως σχετικό με SELinux.
  Επιτυχής εκτέλεση της εντολής επέστρεψε ένα μοναδικό αναγνωριστικό (ID) του container, π.χ.:

80a7fc919f8c94cd6dd0d56c5b457923a3c6f66b3addf185246380b9ccb0e36a

δ. Επαλήθευση Λειτουργίας του Container
Η ορθή λειτουργία του container επιβεβαιώθηκε με την εντολή:

docker ps

Στην έξοδο της εντολής, το container my-mongo εμφανίστηκε με κατάσταση Up:

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
80a7fc919f8c mongo:latest "docker-entrypoint.s…" 3 minutes ago Up 3 minutes 0.0.0.0:27017->27017/tcp, :::27017->27017/tcp my-mongo

2. Εισαγωγή των Αρχείων GeoJSON στη MongoDB

Με το MongoDB container σε λειτουργία, τα δεδομένα εισήχθησαν χρησιμοποιώντας την εντολή mongoimport εντός του περιβάλλοντος του container.

α. Πρόσβαση στο Shell του MongoDB Container
Η πρόσβαση στο bash shell του my-mongo container επιτεύχθηκε με την εντολή:

docker exec -it my-mongo bash

β. Εισαγωγή των GeoJSON Αρχείων (Εντός του Container Shell)
Κάθε αρχείο GeoJSON εισήχθη ως μια ξεχωριστή συλλογή (collection) στη βάση δεδομένων greece_db. Χρησιμοποιήθηκε η παράμετρος --jsonArray καθώς τα παρεχόμενα αρχεία .geojsonl.json αντιμετωπίστηκαν ως πίνακες αντικειμένων JSON για την εισαγωγή. Οι εντολές που εκτελέστηκαν ήταν (1 σειρά η κάθε μία, 8 στο σύνολο):

mongoimport --db greece_db --collection aerodromia --file /import_data/aerodromia.geojsonl.json

mongoimport --db greece_db --collection iso_metadata --file /import_data/iso_metadata.geojsonl.json

mongoimport --db greece_db --collection limnes --file /import_data/limnes.geojsonl.json

mongoimport --db greece_db --collection nomoi --file /import_data/nomoi.geojsonl.json

mongoimport --db greece_db --collection oikismoi --file /import_data/oikismoi.geojsonl.json

mongoimport --db greece_db --collection poleis --file /import_data/poleis.geojsonl.json

mongoimport --db greece_db --collection potamoi --file /import_data/potamoi.geojsonl.json

mongoimport --db greece_db --collection sid_diktio --file /import_data/sid_diktio.geojsonl.json

Για κάθε επιτυχή εισαγωγή, εμφανίστηκε σχετικό μήνυμα, π.χ., imported XX documents.

γ. Δημιουργία Geospatial Indices

Μετά την επιτυχή εισαγωγή των δεδομένων, κρίθηκε απαραίτητη η δημιουργία χωρικών δεικτών (geospatial indices) τύπου 2dsphere στα πεδία geometry κάθε συλλογής. Αυτό είναι θεμελιώδες για την αποδοτική εκτέλεση χωρικών ερωτημάτων.
Για τη δημιουργία των δεικτών, αρχικά συνδεθήκαμε στο MongoDB shell μέσω της εντολής mongosh (ή mongo για παλαιότερες εκδόσεις MongoDB) εντός του container:

mongosh

Στη συνέχεια, εντός του mongosh shell, εκτελέστηκαν οι παρακάτω εντολές:

// Επιλογή της βάσης δεδομένων
use greece_db;

// Δημιουργία δεικτών 2dsphere για κάθε σχετική συλλογή
db.aerodromia.createIndex({ geometry: "2dsphere" });

db.iso_metadata.createIndex({ geometry: "2dsphere" });
// Προστέθηκε για πληρότητα, αν και τα metadata σπάνια περιέχουν γεωμετρία

db.limnes.createIndex({ geometry: "2dsphere" });

db.nomoi.createIndex({ geometry: "2dsphere" });

db.oikismoi.createIndex({ geometry: "2dsphere" });

db.poleis.createIndex({ geometry: "2dsphere" });

db.potamoi.createIndex({ geometry: "2dsphere" });

db.sid_diktio.createIndex({ geometry: "2dsphere" });

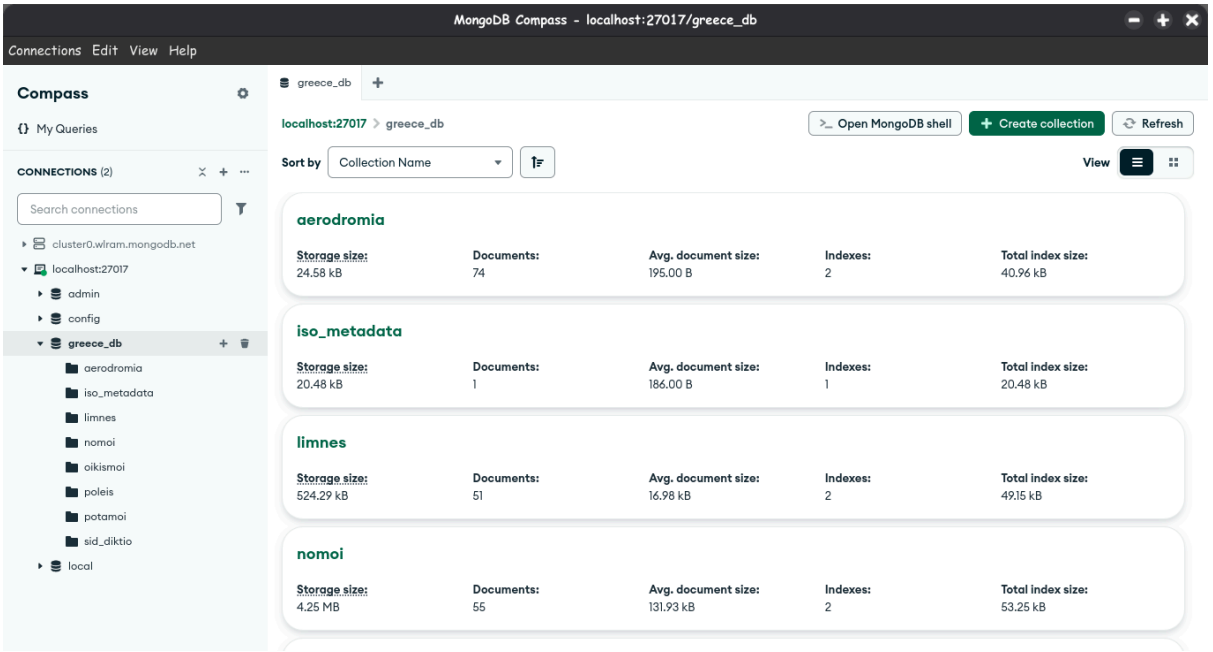
Η επιτυχής δημιουργία κάθε δείκτη επιβεβαιώθηκε από μηνύματα της μορφής {"numIndexesBefore": X, "numIndexesAfter": Y, "ok": 1}.

Σημαντικές Επισημάνσεις κατά τη Διαδικασία:

- Η ύπαρξη των 2dsphere δεικτών είναι **καθοριστική** για τη σωστή και γρήγορη απόκριση των χωρικών τελεστών της MongoDB (π.χ., $geoIntersects, $near). Χωρίς αυτούς, τα ερωτήματα θα ήταν εξαιρετικά αργά ή δεν θα παρήγαγαν τα αναμενόμενα αποτελέσματα.
- Εάν κάποια συλλογή δεν περιείχε γεωχωρικά δεδομένα ή δεν προβλεπόταν να χρησιμοποιηθεί σε χωρικά ερωτήματα, ο αντίστοιχος δείκτης θα μπορούσε να παραλειφθεί. Ωστόσο, για τις ανάγκες της παρούσας εργασίας και για λόγους ομοιομορφίας, δημιουργήθηκαν για όλες τις συλλογές που δυνητικά περιέχουν γεωμετρικά πεδία.
- Σε περίπτωση σφαλμάτων κατά την εισαγωγή ή τη δημιουργία δεικτών, ελέγχθηκαν:
    - Η ενεργή βάση δεδομένων (use greece_db;).
    - Η ορθή ονομασία των συλλογών και η επιτυχής εισαγωγή τους (π.χ. με db.collectionName.find().limit(1)).
    - Η ύπαρξη και εγκυρότητα του πεδίου geometry (ως GeoJSON) στα έγγραφα των συλλογών.

Με την ολοκλήρωση της δημιουργίας των δεικτών, το περιβάλλον ήταν έτοιμο για την εκτέλεση των χωρικών ερωτημάτων. Η αποσύνδεση από το mongosh shell έγινε με την εντολή exit, και ακολούθως η αποσύνδεση από το bash shell του container με μια δεύτερη εντολή exit.

Έγινε και έλεγχος με το compass να δούμε αν εισήχθησαν και μπορεί να συνδεθεί κάποιος με την βάση.



Εικόνα 1: MongoDB Compass

Β. Εκτέλεση Ερωτημάτων και Παρουσίαση Αποτελεσμάτων μέσω Node.js Script

Μετά την επιτυχή προετοιμασία του περιβάλλοντος MongoDB και την εισαγωγή των γεωχωρικών δεδομένων, το επόμενο στάδιο περιλάμβανε την ανάπτυξη και εκτέλεση των ερωτημάτων (queries) που σχεδιάστηκαν για την ανάλυση αυτών των δεδομένων. Για την αυτοματοποίηση αυτής της διαδικασίας και την ευκολότερη διαχείριση των ερωτημάτων και των αποτελεσμάτων τους, επιλέχθηκε η χρήση ενός Node.js script.

1. Προετοιμασία Περιβάλλοντος Node.js και Απαιτούμενες Βιβλιοθήκες

Για την εκτέλεση των ερωτημάτων MongoDB εκτός του mongosh shell, και συγκεκριμένα μέσω ενός Node.js script, απαιτήθηκε η εγκατάσταση των παρακάτω στοιχείων:

1. **Node.js Runtime Environment:** Εξασφαλίστηκε ότι το Node.js ήταν εγκατεστημένο στον υπολογιστή εργασίας. Το Node.js παρέχει το περιβάλλον εκτέλεσης για JavaScript κώδικα εκτός του προγράμματος περιήγησης.

MongoDB Node.js Driver: Για την επικοινωνία του Node.js script με την MongoDB instance, εγκαταστάθηκε η επίσημη βιβλιοθήκη (driver) της MongoDB για Node.js. Η εγκατάσταση πραγματοποιήθηκε στον φάκελο του project (όπου θα δημιουργούνταν το αρχείο mongo_queries.js) μέσω του Node Package Manager (npm) με την εντολή:

```
npm install mongodb
```

Προηγουμένως, συνιστάται η αρχικοποίηση ενός package.json αρχείου στον φάκελο του project (αν δεν υπάρχει ήδη) με την εντολή

npm init -y

καθώς και των εξαρτήσεων

npm install @turf/turf

που χρησιμοποιήθηκαν.

2. Δημιουργία του Script mongo_queries.js

Δημιουργήθηκε ένα αρχείο JavaScript με την ονομασία mongo_queries.js. Το αρχείο αυτό περιέχει τον κώδικα που είναι απαραίτητος για:

1. **Σύνδεση με τον MongoDB Server:** Καθιέρωση σύνδεσης με την instance της MongoDB που εκτελείται στο Docker container (στη διεύθυνση mongodb://localhost:27017).
2. **Επιλογή της Βάσης Δεδομένων:** Στόχευση στη βάση δεδομένων greece_db όπου έχουν εισαχθεί τα δεδομένα.
3. **Ορισμός και Εκτέλεση των Ερωτημάτων:** Υλοποίηση των διαφόρων ερωτημάτων (queries) που αναπτύχθηκαν για την ανάλυση των γεωχωρικών δεδομένων. Κάθε ερώτημα εκτελείται έναντι της κατάλληλης συλλογής (collection).
4. **Επεξεργασία και Εμφάνιση Αποτελεσμάτων:** Ο κώδικας χειρίζεται τα αποτελέσματα που επιστρέφονται από κάθε ερώτημα και τα εκτυπώνει στην κονσόλα (terminal) με δομημένο τρόπο για ευκολότερη ανάγνωση και επαλήθευση.

```
// Παράδειγμα Δομής του mongo_queries.js (απλουστευμένο)
const { MongoClient } = require('mongodb');
const turf = require('@turf/turf');

async function main() {
  const uri = "mongodb://localhost:27017"; // URI σύνδεσης στη MongoDB
  const client = new MongoClient(uri);

  try {
    await client.connect();
    console.log("Επιτυχής σύνδεση στη MongoDB!");

    const database = client.db("greece_db");
    const poleisCollection = database.collection("poleis");

    // ... (Περισσότερα ερωτήματα και επεξεργασία αποτελεσμάτων) ...
```

```
    } catch (e) {
        console.error("Σφάλμα κατά την εκτέλεση:", e);
    } finally {
        await client.close();
        console.log("Η σύνδεση με τη MongoDB έκλεισε.");
    }
}

main().catch(console.error);
```

3. Εκτέλεση του Script και Λήψη Αποτελεσμάτων

Αφού ολοκληρώθηκε η συγγραφή του mongo_queries.js και η εγκατάσταση των απαραίτητων πακέτων, το script εκτελέστηκε από το τερματικό, μέσα από τον φάκελο που το περιείχε, με την ακόλουθη εντολή:

node mongo_queries.js

Είναι σημαντικό να σημειωθεί ότι για την επιτυχή εκτέλεση του script, το Docker container της MongoDB (my-mongo) έπρεπε να είναι σε λειτουργία (Up).

Τα αποτελέσματα των ερωτημάτων, όπως αυτά μορφοποιήθηκαν και εκτυπώθηκαν από το script, εμφανίστηκαν απευθείας στο τερματικό. Μια σύνοψη της εξόδου του τερματικού, που απεικονίζει τα αποτελέσματα των εκτελεσθέντων ερωτημάτων, παρατίθεται στην Εικόνα 2 έως 4.

```
Connected to MongoDB
--- Starting MongoDB Queries ---

Query 0a: Nomoi that have oikismous whose name begins with Ω

Query 0a result: [
  'Ν. ΙΩΑΝΝΙΝΩΝ',
  'Ν. ΑΝΑΤΟΛΙΚΗΣ ΑΤΤΙΚΗΣ',
  'Ν. ΕΥΒΟΙΑΣ',
  'Ν. ΑΡΚΑΔΙΑΣ',
  'Ν. ΗΛΕΙΑΣ',
  'Ν. ΞΑΝΘΗΣ',
  'Ν. ΘΕΣΣΑΛΟΝΙΚΗΣ'
]

Query 0b: Cities within 10km of Aliakmonas river

Query 0b result: [ 'Βέροια', 'Γρεβενά', 'Καστοριά' ]
```

Εικόνα 2: *Αποτελέσματα εκτέλεσης του script mongo_queries.js στο τερματικό, απεικονίζοντας την έξοδο των ερωτημάτων 0a και 0b.*

```
Query 1: Nomoi through which the OSE network passes

Query 1 result: [
  'Ν. ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ',     'Ν. ΠΙΕΡΙΑΣ',
  'Ν. ΠΕΛΛΑΣ',              'Ν. ΛΑΡΙΣΑΣ',
  'Ν. ΕΒΡΟΥ',               'Ν. ΚΟΖΑΝΗΣ',
  'Ν. ΠΕΙΡΑΙΩΣ ΚΑΙ ΝΗΣΩΝ', 'Ν. ΣΕΡΡΩΝ',
  'Ν. ΗΜΑΘΙΑΣ',             'Ν. ΑΧΑΪΑΣ',
  'Ν. ΦΛΩΡΙΝΑΣ',            'Ν. ΦΘΙΩΤΙΔΑΣ',
  'Ν. ΑΘΗΝΩΝ',              'Ν. ΑΝΑΤΟΛΙΚΗΣ ΑΤΤΙΚΗΣ',
  'Ν. ΡΟΔΟΠΗΣ',             'Ν. ΚΟΡΙΝΘΟΥ',
  'Ν. ΕΥΒΟΙΑΣ',             'Ν. ΑΡΚΑΔΙΑΣ',
  'Ν. ΔΡΑΜΑΣ',              'Ν. ΒΟΙΩΤΙΑΣ',
  'Ν. ΚΑΒΑΛΑΣ',             'Ν. ΑΡΓΟΛΙΔΑΣ',
  'Ν. ΜΑΓΝΗΣΙΑΣ',           'Ν. ΗΛΕΙΑΣ',
  'Ν. ΚΑΡΔΙΤΣΑΣ',           'Ν. ΞΑΝΘΗΣ',
  'Ν. ΚΙΛΚΙΣ',              'Ν. ΘΕΣΣΑΛΟΝΙΚΗΣ',
  'Ν. ΤΡΙΚΑΛΩΝ',            'Ν. ΜΕΣΣΗΝΙΑΣ'
]
```

Εικόνα 3: *Αποτελέσματα εκτέλεσης του script mongo_queries.js στο τερματικό, απεικονίζοντας την έξοδο του ερωτημάτος 1.*

```
Query 2: Nomoi bordering Thessaloniki and having a lake
Query 2 result: [ 'Ν. ΠΕΛΛΑΣ', 'Ν. ΚΙΛΚΙΣ', 'Ν. ΗΜΑΘΙΑΣ', 'Ν. ΣΕΡΡΩΝ' ]

Query 3: Nomoi without airports

Query 3 result: [
  'Ν. ΕΥΡΥΤΑΝΙΑΣ', 'Ν. ΦΩΚΙΔΑΣ',
  'Ν. ΣΕΡΡΩΝ',     'Ν. ΧΑΛΚΙΔΙΚΗΣ',
  'Ν. ΠΡΕΒΕΖΑΣ',   'Ν. ΓΡΕΒΕΝΩΝ',
  'Ν. ΚΟΡΙΝΘΟΥ',   'Ν. ΛΕΥΚΑΔΑΣ',
  'Ν. ΔΡΑΜΑΣ',     'Ν. ΡΕΘΥΜΝΟΥ',
  'Ν. ΚΑΡΔΙΤΣΑΣ',  'ΑΓΙΟ ΟΡΟΣ',
  'Ν. ΞΑΝΘΗΣ',     'Ν. ΑΡΤΑΣ',
  'Ν. ΚΙΛΚΙΣ',     'Ν. ΘΕΣΠΡΩΤΙΑΣ',
  'Ν. ΤΡΙΚΑΛΩΝ'
]
MongoDB connection closed
```

Εικόνα 4: *Αποτελέσματα εκτέλεσης του script mongo_queries.js στο τερματικό, απεικονίζοντας την έξοδο των ερωτημάτων 2 και 3.*