Ραφαήλ Ματζιρίδης ics23206

PP1 – Programming Project

```
-- 1. Δημιουργία του τελικού πίνακα
CREATE TABLE IF NOT EXISTS docs (
    id int GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    title text,
    abstract text
);

--2 . Προσθήκη στηλών tsvector για αναζήτηση πλήρους κειμένου
ALTER TABLE docs
ADD COLUMN IF NOT EXISTS title_tsv tsvector,
ADD COLUMN IF NOT EXISTS abstract_tsv tsvector;

-- 3. Δημιουργία προσωρινού πίνακα για τη φόρτωση
CREATE TEMP TABLE temp_json (
    line_content TEXT
);

-- 4. Φόρτωση από το αρχείο (Το αρχείο πρέπει να βρίσκεται στο /tmp/data.json
μέσα στο container)
COPY temp_json (line_content)
FROM '/tmp/data.json'
WITH (FORMAT csv, QUOTE e'\x01', DELIMITER e'\x02');

-- 5. Μεταφορά στον docs και εξαγωγή των πεδίων JSON
INSERT INTO docs (title, abstract)
SELECT
    (line_content::jsonb) ->> 'title',
    (line_content::jsonb) ->> 'abstract'
FROM temp_json
WHERE line_content IS NOT NULL AND line_content != '';

-- 6. Δημιουργία των tsvectors
UPDATE docs SET
    title_tsv = to_tsvector('english', COALESCE(title, '')),
    abstract_tsv = to_tsvector('english', COALESCE(abstract, ''));

-- 7. Δημιουργία των ευρετηρίων GIN
CREATE INDEX IF NOT EXISTS idx_title_tsv ON docs USING GIN(title_tsv);
CREATE INDEX IF NOT EXISTS idx_abstract_tsv ON docs USING
GIN(abstract_tsv);
```

Ερώτημα 1: Πλήθος εγγράφων με τους όρους ‘rat’ ή ‘liver’ στον τίτλο

- **SQL Query:**

  ```
  SELECT count(*) AS total_docs FROM docs
  WHERE to_tsvector('english', title) @@ to_tsquery('english', 'rat | liver');
  ```

- **Αποτέλεσμα: 3.645**

Ερώτημα 2: Πλήθος εγγράφων με τους όρους ‘rat’ ή ‘liver’ στην περίληψη (abstract)

- **SQL Query:**

  ```
  SELECT count(*) AS total_docs FROM docs
  WHERE to_tsvector('english', abstract) @@ to_tsquery('english', 'rat | liver');
  ```

- **Αποτέλεσμα: 8.674**

Ερώτημα 3: Πλήθος εγγράφων με τους όρους ‘rat’ ή ‘liver’ στον τίτλο ή στην περίληψη

- **SQL Query:**

  ```
  SELECT count(*) AS total_docs FROM docs
  WHERE (to_tsvector('english', title) || to_tsvector('english', abstract)) @@ to_tsquery('english', 'rat | liver');
  ```

- **Αποτέλεσμα: 9.214**

Ερώτημα 4: Πλήθος εγγράφων που περιέχουν τους όρους ‘rat’ ή ‘liver’ και στα δύο πεδία (Title & Abstract)

- **SQL Query:**

  ```
  SELECT count(*) AS total_docs FROM docs
  WHERE to_tsvector('english', title) @@ to_tsquery('english', 'rat | liver')
    AND to_tsvector('english', abstract) @@ to_tsquery('english', 'rat | liver');
  ```

- **Αποτέλεσμα: 3.105**

Ερώτημα 5: Πλήθος εγγράφων που περιέχουν συνδυαστικά τους όρους ‘rat’ και ‘liver’

- **SQL Query:**

```
SELECT count(*) AS total_docs FROM docs
WHERE (to_tsvector('english', title) || to_tsvector('english', abstract)) @@
to_tsquery('english', 'rat & liver');
```

- **Αποτέλεσμα: 1.129**

Ερώτημα 6: Εύρεση εγγράφων που περιέχουν στο abstract τους όρους ‘cancer’ και ‘liver’ με φθίνουσα κατάταξη βάσει σχετικότητας.

- **SQL Query:**

```
SELECT title,
      ts_rank_cd(abstract_tsv, query) AS rank
FROM docs, to_tsquery('english', 'cancer & liver') query
WHERE abstract_tsv @@ query
ORDER BY rank DESC
LIMIT 10;
```

- **Σχολιασμός:** Η συνάρτηση ts_rank_cd (Cover Density Ranking) χρησιμοποιείται για την αξιολόγηση της σχετικότητας των εγγράφων. Αντίθετα με την απλή καταμέτρηση, η ts_rank_cd λαμβάνει υπόψη πόσο κοντά βρίσκονται οι όροι αναζήτησης μέσα στο κείμενο. Έτσι, έγγραφα που περιέχουν τις λέξεις "cancer" και "liver" στην ίδια πρόταση κατατάσσονται υψηλότερα από αυτά που τις περιέχουν σε διαφορετικές παραγράφους.

Ανάλυση Συχνότητας Όρων (Term Frequencies)

Ερώτημα 7: Top 10 όροι ως προς το Document Frequency (ndoc)

- **SQL Query:**

```
SELECT word, ndoc FROM ts_stat('SELECT to_tsvector(''english'',
abstract) FROM docs')
ORDER BY ndoc DESC LIMIT 10;
```

- **Αποτέλεσμα:**
- result | 52517
- use | 47230
- studi | 45943
- cell | 37906
- conclus | 32587

- also | 28823
- background | 28735
- method | 28414
- show | 27660
- effect | 27571

Ερώτημα 8: Top 10 όροι ως προς το Collection Frequency (nentry)

- **SQL Query:**

  SQL

  SELECT word, nentry FROM ts_stat('SELECT to_tsvector(''english'', abstract) FROM docs')
  ORDER BY nentry DESC LIMIT 10;

- **Αποτέλεσμα:**
- cell | 165369
- use | 89734
- studi | 80767
- result | 77081
- protein | 71550
- patient | 68564
- gene | 64737
- activ | 60729
- express | 55068
- effect | 50111
- (10 rows)

Web Εφαρμογή & Deployment

Πέραν των SQL ερωτημάτων, το project ολοκληρώθηκε με την ανάπτυξη μιας πλήρως λειτουργικής **Web εφαρμογής αναζήτησης**. Η εφαρμογή επιτρέπει στον χρήστη να εκτελεί δυναμικά ερωτήματα στη βάση δεδομένων PubMed μέσω ενός φιλικού περιβάλλοντος διεπαφής.

Τεχνική Υλοποίηση

- **Backend:** Αναπτύχθηκε σε **Python** με τη χρήση του framework **Flask**. Ο κώδικας διαχειρίζεται τη σύνδεση με την PostgreSQL και την εκτέλεση των Full Text Search queries σε πραγματικό χρόνο.

- **Containerization:** Η εφαρμογή και η βάση δεδομένων έχουν πλήρως "πακεταριστεί" με **Docker** και **Docker Compose**, διασφαλίζοντας την εύκολη μεταφερσιμότητα και αναπαραγωγιμότητα του περιβάλλοντος (Infrastructure as Code).
- **Reverse Proxy & Security:** Χρησιμοποιήθηκε **Nginx** ως reverse proxy για τη διαχείριση της κίνησης, ενώ η σύνδεση προστατεύεται με κρυπτογράφηση **SSL/TLS** (Let's Encrypt).

Live Demo

Η εφαρμογή είναι πλέον "ζωντανή" και προσβάσιμη στην παρακάτω διεύθυνση: **https://searchengine.rmat.gr**

Repository

Ο πλήρης κώδικας της εφαρμογής, τα Dockerfiles, καθώς και οι ρυθμίσεις του Nginx έχουν ενσωματωθεί στο σχετικό **GitHub Repository**, παρέχοντας όλες τις οδηγίες για το τοπικό στήσιμο (local setup) ή την παραγωγική λειτουργία (deployment). https://github.com/RafMat98/SimpleSearchEngine