ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

HTML Scrapping με χρήση Python

Του φοιτητή
Φωτίου Παναγιώτη
Αρ. Μητρώου: 144188

Επιβλέπων
Ονοματεπώνυμο Μιχαήλ
Σαλαμπάσης
Βαθμίδα ……

Ημερομηνία 18-03-2020

Τίτλος Δ.Ε. HTML Scrapping με χρήση Python
Κωδικός Δ.Ε. 20107
Ονοματεπώνυμο φοιτητή/τών Παναγιώτης Φωτίου
Ονοματεπώνυμο εισηγητή Μιχαήλ Σαλαμπάσης
Ημερομηνία ανάληψης Δ.Ε. 18-03-2020
Ημερομηνία περάτωσης Δ.Ε. …

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Αφιέρωση

Αφιερώνω την πτυχιακή μου εργασία στους γονείς μου.

Πρόλογος

Η εκπόνηση της πτυχιακής εργασίας έγινε στα πλαίσια σπουδών στο Δ.Ι.Π.Α.Ε Θεσσαλονίκης Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων για την απόκτηση του τίτλου σπουδών. Το διαδίκτυο είναι το μεγαλύτερο μέσο μαζικής ενημέρωσης και επικοινωνίας στην εποχή μας. Η πληροφορία που διαθέτει είναι τεράστια και πολλές φορές σημαντική ώστε να συλλεχθεί από κάποιον άνθρωπο ή οργανισμό, το web scraping είναι η αυτοματοποιημένη διαδικασία που συλλέγει στοχευμένα δεδομένα. Ο σκοπός της εργασίας είναι η συλλογή δεδομένων θεατρικών παραστάσεων μέσω του viva.gr. Έχω επιλέξει αυτήν την πτυχιακή για το ενδιαφέρον μου στις εφαρμογές διαδικτύου και την δυνατότητα να εξοικειωθώ με την γλώσσα προγραμματισμού python μαζί με τις απαραίτητες βιβλιοθήκες για το web scraping. Οι δυσκολίες που αντιμετώπισα στην υλοποίηση του προγράμματος με βοήθησαν να εξελιχθώ ως προγραμματιστής ανάπτυξης εφαρμογών.

Περίληψη

Η στόχος της εργασίας είναι η υλοποίηση ενός συστήματος HTML scraper οπού συλλέγει δεδομένα θεατρικών παραστάσεων από το viva.gr και τα αποθηκεύει σε μια βάση δεδομένων. Αυτά τα δεδομένα βρίσκονται σε μορφή html μέσα σε ετικέτες οι οποίες επιλέγονται στοχευμένα από το σύστημα scraping. Με μια συνεχής ροή ο scraper περιηγείται στις ιστοσελίδες με μια συγκεκριμένη διαδρομή και συνεχείς ελέγχους και συλλέγει τα δεδομένα. Το πρόγραμμα είναι γραμμένο σε python με χρήση της βιβλιοθήκης beautifulsoup οπού παρέχει συναρτήσεις για το web scraping. Υπάρχουν λειτουργίες μέσω γραφικού περιβάλλον για την διαχείριση της βασικής λειτουργίας και την εμφάνιση βασικών στοιχείων στην οθόνη. Επίσης υπάρχει η δυνατότητα «αθόρυβης εκτέλεσης» μέσω windows services. Το σύστημα αποθήκευσης είναι μια σχεσιακή βάση δεδομένων SQL με αρκετά πεδία στους πίνακες ώστε να καλύπτουν όλη την χρήσιμη πληροφορία από ένα θεατρικό έργο.

Ένα σύστημα web scraping μπορεί να υλοποιηθεί ως λύση μιας διαδικασίας που γίνεται χειροκίνητα από κάποιον άνθρωπο σε μια αυτοματοποιημένη διαδικασία.

Τα αποτελέσματα δείχνουν, ότι με βάση των θεατρικών παραστάσεων του ιστότοπου viva.gr, οι πληροφορίες συλλέγονται και αποθηκεύονται με ορθότητα.

HTML Scrapping using Python

Fotiou Panagiotis

Abstract

The purpose of this thesis is the implementation of an HTML scraper system where it collects data of theatrical performances from viva.gr and stores them in a database. This data is in html format in tags which are selected and targeted by the scraping system. The scraper browses the web pages with a specific path and continuous scouting and collects the data. The program is written in python using the beautifulsoup library where it provides functions for web scraping. There are graphical interface functions for managing the basic function and displaying data on the screen. There is also the ability of "silent execution" through windows services. The storage system is a relational SQL database with enough fields in the tables to cover all the useful information from a theatrical play.

A web scraping system can be implemented as a solution to a process done manually by a person in an automated process.

The results show that based on the theatrical performances of the viva.gr website, the information is collected and stored correctly.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την οικογένεια μου για την συνεχής υποστήριξη κατά την εκπόνηση της διπλωματικής εργασίας και τον επιβλέποντα καθηγητή Μιχαήλ Σαλαμπάση για την εύκαιρα που μου δόθηκε να επιλέξω αυτό το θέμα μαζί με την καθοδήγηση και την συμπαράστασή του σε όλη την διάρκεια της διπλωματικής.

Περιεχόμενα

Κατάλογος Σχημάτων


```
markup = "<b><!--Hey, buddy. Want to buy a used parser?--></b>"
soup = BeautifulSoup(markup)
comment = soup.b.string
```

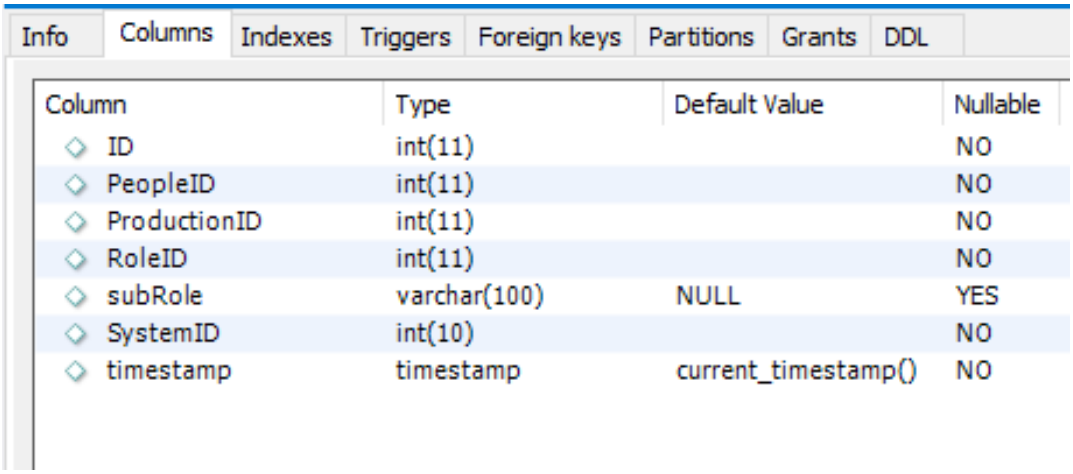

Info | Columns | Indexes | Triggers | Foreign keys | Partitions | Grants | DDL

Column	Type	Default Value	Nullable
ID	int(11)		NO
PeopleID	int(11)		NO
ProductionID	int(11)		NO
RoleID	int(11)		NO
subRole	varchar(100)	NULL	YES
SystemID	int(10)		NO
timestamp	timestamp	current_timestamp()	NO

```
1   DELIMITER $$
2   CREATE TRIGGER role_insert_trigger AFTER INSERT ON roles
3   FOR EACH ROW
4   BEGIN
5     INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
6     VALUES('insert', 'roles', NEW.ID , 'ID');
7     INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
8     VALUES('insert', 'roles', NEW.Role , 'Role');
9     INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
10    VALUES('insert', 'roles', NEW.SystemID , 'SystemID');
11  END$$
12  DELIMITER ;
13
14  DELIMITER $$
15  CREATE TRIGGER role_update_trigger AFTER UPDATE ON roles
16  FOR EACH ROW
17  BEGIN
18    INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
19    VALUES('update', 'roles', CONCAT(OLD.ID, ' >> ', NEW.ID), 'ID');
20    INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
21    VALUES('update', 'roles', CONCAT(OLD.Role, ' >> ', NEW.Role), 'Role');
22    INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
23    VALUES('update', 'roles', CONCAT(OLD.SystemID, ' >> ', NEW.SystemID), 'SystemID');
24  END$$
25  DELIMITER ;
26
27  DELIMITER $$
28  CREATE TRIGGER role_delete_trigger AFTER DELETE ON roles
29  FOR EACH ROW
30  BEGIN
31    INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
32    VALUES('delete', 'roles', OLD.ID, 'ID');
33    INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
34    VALUES('delete', 'roles', OLD.Role, 'Role');
35    INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
36    VALUES('delete', 'roles', OLD.SystemID, 'SystemID');
37  END$$
38  DELIMITER ;
```



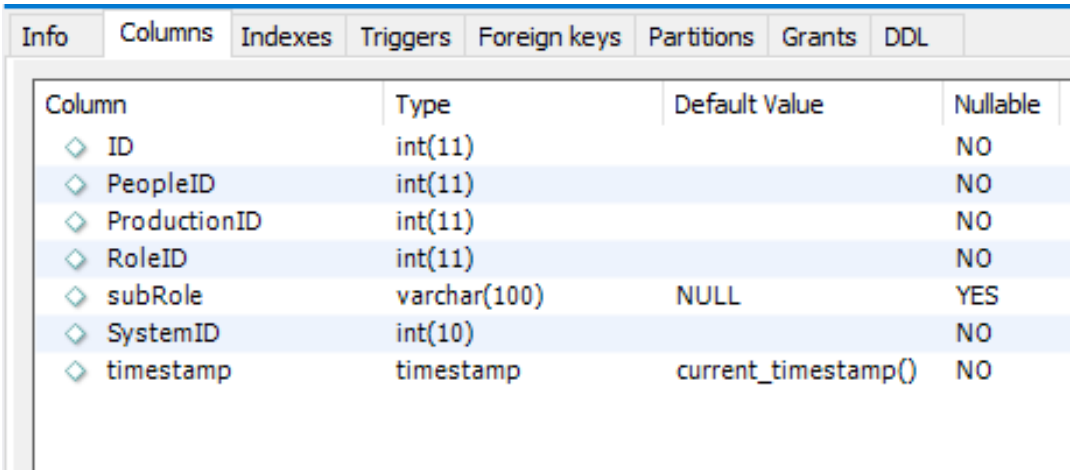
```
markup = "<b><!--Hey, buddy. Want to buy a used parser?--></b>"
soup = BeautifulSoup(markup)
comment = soup.b.string
```



```
1   DELIMITER $$
2   CREATE TRIGGER role_insert_trigger AFTER INSERT ON roles
3   FOR EACH ROW
4   BEGIN
5     INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
6     VALUES('insert', 'roles', NEW.ID , 'ID');
7     INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
8     VALUES('insert', 'roles', NEW.Role , 'Role');
9     INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
10    VALUES('insert', 'roles', NEW.SystemID , 'SystemID');
11  END$$
12  DELIMITER ;
13
14  DELIMITER $$
15  CREATE TRIGGER role_update_trigger AFTER UPDATE ON roles
16  FOR EACH ROW
17  BEGIN
18    INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
19    VALUES('update', 'roles', CONCAT(OLD.ID, ' >> ', NEW.ID), 'ID');
20    INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
21    VALUES('update', 'roles', CONCAT(OLD.Role, ' >> ', NEW.Role), 'Role');
22    INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
23    VALUES('update', 'roles', CONCAT(OLD.SystemID, ' >> ', NEW.SystemID), 'SystemID');
24  END$$
25  DELIMITER ;
26
27  DELIMITER $$
28  CREATE TRIGGER role_delete_trigger AFTER DELETE ON roles
29  FOR EACH ROW
30  BEGIN
31    INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
32    VALUES('delete', 'roles', OLD.ID, 'ID');
33    INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
34    VALUES('delete', 'roles', OLD.Role, 'Role');
35    INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
36    VALUES('delete', 'roles', OLD.SystemID, 'SystemID');
37  END$$
38  DELIMITER ;
```



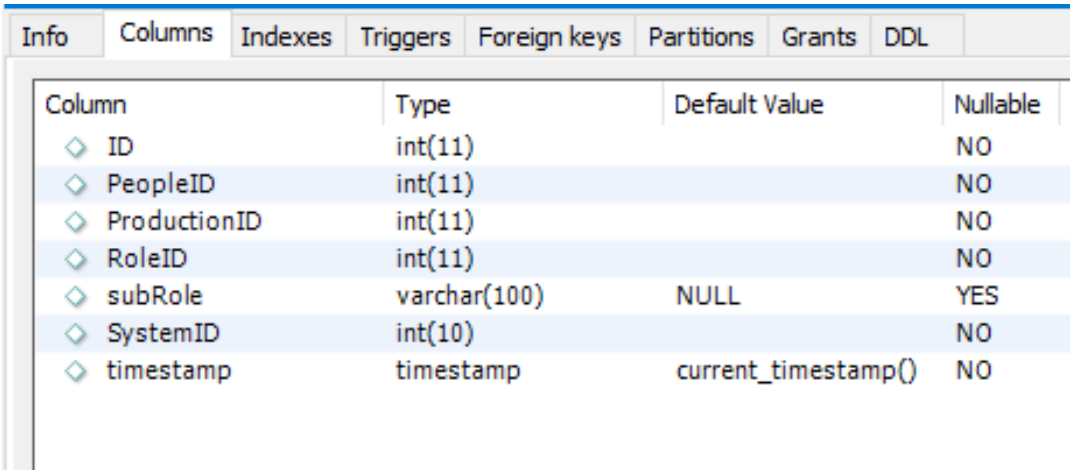
```
markup = "<b><!--Hey, buddy. Want to buy a used parser?--></b>"
soup = BeautifulSoup(markup)
comment = soup.b.string
```



```
DELIMITER $$
CREATE TRIGGER role_insert_trigger AFTER INSERT ON roles
FOR EACH ROW
BEGIN
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('insert', 'roles', NEW.ID , 'ID');
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('insert', 'roles', NEW.Role , 'Role');
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('insert', 'roles', NEW.SystemID , 'SystemID');
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER role_update_trigger AFTER UPDATE ON roles
FOR EACH ROW
BEGIN
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('update', 'roles', CONCAT(OLD.ID, ' >> ', NEW.ID), 'ID');
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('update', 'roles', CONCAT(OLD.Role, ' >> ', NEW.Role), 'Role');
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('update', 'roles', CONCAT(OLD.SystemID, ' >> ', NEW.SystemID), 'SystemID');
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER role_delete_trigger AFTER DELETE ON roles
FOR EACH ROW
BEGIN
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('delete', 'roles', OLD.ID, 'ID');
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('delete', 'roles', OLD.Role, 'Role');
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('delete', 'roles', OLD.SystemID, 'SystemID');
END$$
DELIMITER ;
```



```
markup = "<b><!--Hey, buddy. Want to buy a used parser?--></b>"
soup = BeautifulSoup(markup)
comment = soup.b.string
```



```
DELIMITER $$
CREATE TRIGGER role_insert_trigger AFTER INSERT ON roles
FOR EACH ROW
BEGIN
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('insert', 'roles', NEW.ID , 'ID');
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('insert', 'roles', NEW.Role , 'Role');
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('insert', 'roles', NEW.SystemID , 'SystemID');
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER role_update_trigger AFTER UPDATE ON roles
FOR EACH ROW
BEGIN
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('update', 'roles', CONCAT(OLD.ID, ' >> ', NEW.ID), 'ID');
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('update', 'roles', CONCAT(OLD.Role, ' >> ', NEW.Role), 'Role');
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('update', 'roles', CONCAT(OLD.SystemID, ' >> ', NEW.SystemID), 'SystemID');
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER role_delete_trigger AFTER DELETE ON roles
FOR EACH ROW
BEGIN
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('delete', 'roles', OLD.ID, 'ID');
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('delete', 'roles', OLD.Role, 'Role');
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('delete', 'roles', OLD.SystemID, 'SystemID');
END$$
DELIMITER ;
```


Κατάλογος Πινάκων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
API	Application Programming Interface
HTML	HyperText Markup Language
XML	Extensive Markup Language
DOM	Document Object Model
SQL	Structured Query Language

Κεφάλαιο 1ο: Εισαγωγή στο Web scraping

1.1 Εισαγωγή

Στο πρώτο κεφάλαιο θα γίνει μια εισαγωγή με τις βασικές έννοιες και τεχνολογίες που χρησιμοποιεί το web scraping, για ποιόν λόγο να υπάρχει το web scraping, σύγκριση με εναλλακτικές τεχνολογίες, τις τεχνικές του και ανάλυση της δομής θεατρικών έργων viva.gr

1.2 Web scraping

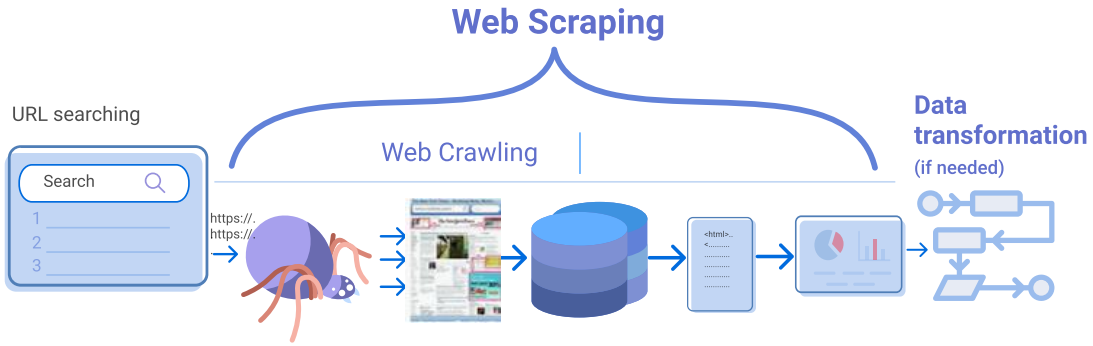
Web scraping, επίσης γνωστή ως εξαγωγή πληροφοριών του ιστού, είναι μια τεχνική για την εξαγωγή δεδομένων από το World Wide Web (WWW) και αποθηκεύεται σε ένα σύστημα αρχείων ή βάση δεδομένων με σκοπό τη μεταγενέστερη ανάκτηση ή ανάλυση των δεδομένων.

Το web scraping μπορεί να χρησιμοποιηθεί για μια μεγάλη ποικιλία σεναρίων, όπως παρακολούθηση/σύγκριση αλλαγής τιμών προϊόντων, ανίχνευση αλλαγών σε ιστοσελίδες , ερευνητικές αναλύσεις, ανάλυση διαφήμισης-αγοράς, παρακολούθηση δεδομένων καιρού, ανίχνευση αλλαγών ιστότοπου και ενοποίηση δεδομένων ιστού. Στην περίπτωση μας στην συλλογή δεδομένων για θεατρικές παραστάσεις.

Λόγω του γεγονότος ότι ένας τεράστιος όγκος ετερογενών δεδομένων δημιουργείται συνεχώς στον παγκόσμιο ιστό, το web scraping είναι ευρέως αναγνωρισμένη ως μια αποτελεσματική και ισχυρή τεχνική για τη συλλογή μεγάλων δεδομένων. Για να προσαρμοστούν σε μια ποικιλία σεναρίων, οι τρέχουσες τεχνικές web scraping έχουν προσαρμοστεί από μικρότερες ανθρωποβοηθημένες διαδικασίες στη χρήση πλήρως αυτοματοποιημένων συστημάτων που είναι ικανά να μετατρέψουν ολόκληρους ιστότοπους σε πλήρες οργανωμένα σύνολα δεδομένων.

Συνήθως, τα δεδομένα συλλέγονται χρησιμοποιώντας το πρωτόκολλο μεταφοράς υπερκειμένου (HTTP) ή μέσω ενός φυλλομετρητή. Αυτό επιτυγχάνεται είτε χειροκίνητα από τον χρήστη ή αυτόματα από ένα bot.

Η διαδικασία scraping από το Διαδίκτυο μπορεί να χωριστεί σε δύο διαδοχικά βήματα, στην συλλογή δεδομένων ιστού και στη συνέχεια εξαγωγή των επιθυμητών πληροφορίων από τα ληφθέντα δεδομένα. Μόλις το αίτημα ληφθεί με επιτυχία και υποβληθεί σε επεξεργασία από τον στοχευμένο ιστότοπο, ο αιτούμενος πόρος θα ανακτηθεί από τον ιστότοπο και στη συνέχεια θα σταλεί πίσω στο πρόγραμμα scraping. Ο πόρος μπορεί να είναι σε πολλές μορφές, όπως ιστοσελίδες που έχουν δημιουργηθεί από HTML, ροές δεδομένων XML ή JSON ή δεδομένα πολυμέσων, όπως εικόνες, αρχεία ήχου ή βίντεο. Μετά τη λήψη των δεδομένων, η διαδικασία εξαγωγής συνεχίζει να αναλύει, να διαμορφώνει ξανά και να οργανώνει τα δεδομένα.



Σχήμα 1.1: Web scraping overview

1.3 Web crawlers

Web crawlers είναι ένα bot μηχανής αναζήτησης πραγματοποιεί λήψη και ευρετηρίαση περιεχομένου από όλο το Διαδίκτυο. Ο στόχος του είναι να μάθει τι αφορά κάθε ιστοσελίδα στον ιστό, έτσι ώστε οι πληροφορίες να μπορούν να ανακτηθούν όταν χρειάζονται. Αυτά τα bot λειτουργούν σχεδόν πάντα από μηχανές αναζήτησης. Εφαρμόζοντας έναν αλγόριθμο αναζήτησης στα δεδομένα που συλλέγονται από web crawlers, οι μηχανές αναζήτησης μπορούν να παρέχουν σχετικούς συνδέσμους ως απάντηση σε ερωτήματα αναζήτησης χρηστών, δημιουργώντας τη λίστα ιστοσελίδων που εμφανίζονται αφού ο χρήστης πληκτρολογήσει μια αναζήτηση στο Google ή σε άλλη μηχανή αναζήτησης. Η λειτουργία του web scraping είναι συνήθως πολύ πιο στοχευμένη από έναν web crawler. Οι web scrapers ενδέχεται να κοιτούν δεδομένα από συγκεκριμένες σελίδες ή συγκεκριμένους ιστότοπους μόνο, ενώ τα crawlers θα συνεχίσουν να ακολουθούν συνδέσμους και να ανιχνεύουν σελίδες συνεχώς. Επίσης, τα web scraper bot ενδέχεται να αγνοήσουν την πίεση που ασκούν στους διακομιστές ιστού, ενώ τα web crawlers, ειδικά αυτά από μεγάλες μηχανές αναζήτησης, θα υπακούσουν σε κανόνες που έχει θέσει η ιστοσελίδα (robots.txt) και θα περιορίζουν τα αιτήματά τους, ώστε να μην κάνουν κατάχρηση πόρων του διακομιστή.

1.4 Γιατί Web scraping;

Εάν ο μόνος τρόπος πρόσβασης στο Διαδίκτυο ήταν μέσω ενός προγράμματος περιήγησης, χάνετε ένα τεράστιο φάσμα δυνατοτήτων. Παρόλο που τα προγράμματα περιήγησης είναι εύχρηστα για την εκτέλεση JavaScript, την εμφάνιση εικόνων και τη διάταξη αντικειμένων σε πιο αναγνώσιμη από τον άνθρωπο μορφή, οι web scrapers είναι εξαιρετικοί στη συλλογή και επεξεργασία μεγάλων ποσοτήτων δεδομένων. Επιπλέον, οι web scrapers μπορούν να πάνε σε μέρη που οι παραδοσιακές μηχανές αναζήτησης δεν μπορούν.

1.5 API

Μια διεπαφή προγραμματισμού εφαρμογών ή API, επιτρέπει στις εταιρείες να ανοίγουν τα δεδομένα και τη λειτουργικότητα των εφαρμογών τους σε εξωτερικούς τρίτους προγραμματιστές, επιχειρηματικούς συνεργάτες και εσωτερικά τμήματα των εταιρειών τους. Αυτό επιτρέπει στις υπηρεσίες και τις εφαρμογές να επικοινωνούν μεταξύ τους και να αξιοποιούν τα δεδομένα και τη λειτουργικότητα του άλλου μέσω μιας τεκμηριωμένης διεπαφής. Οι προγραμματιστές δεν χρειάζεται να γνωρίζουν πώς υλοποιείται ένα API. χρησιμοποιούν απλώς τη διεπαφή για να επικοινωνούν με άλλες εφαρμογές και υπηρεσίες. Οι εφαρμογές, ονομάζονται ως «API endpoints»

Λοιπόν, τα API μπορεί να είναι ιδανικότερα από τους web scrapers, αναλόγως τους σκοπούς σας. Γενικά, είναι προτιμότερο να χρησιμοποιείτε ένα API εάν υπάρχει η δυνατότητα αντί να δημιουργήσετε ένα bot για να λαμβάνετε τα ίδια δεδομένα. Ωστόσο, υπάρχουν λόγοι για τους οποίους ένα API ενδέχεται να μην υπάρχει:

- **Συγκεντρώνετε δεδομένα σε μια συλλογή τοποθεσιών που δεν διαθέτουν public API.**
- **Το σύνολο των δεδομένων που θέλετε είναι λίγα, οπού ο ιδιοκτήτης της ιστοσελίδας δεν πιστεύει ότι δικαιολογεί ένα API.**
- **Η πηγή δεν έχει την υποδομή ή την τεχνική ικανότητα να δημιουργήσει ένα API.**

Επίσης μπορεί να υπάρχουν και λόγοι για την προτίμηση του scraping ενώ υπάρχει API:

- **Περιορισμένη διαθεσιμότητα στα δεδομένα ενός API.**
- **Το API υπάρχει ωστόσο υπάρχει τιμή για να δοθεί πρόσβαση.**
- **Η έλλειψη προσαρμοστικότητας του API.**
- **Ανωνυμία στην συλλογή πληροφορίας.**

Αντιθέτως κάποιοί λόγοι που το web scraping δεν είναι εφικτό

- **Δεν επιτρέπουν όλοι οι ιστότοποι το web scraping (π.χ. Facebook)**
- **Είναι παράνομο ή ανήθικο**

Εν ολίγοις, το scraping και η χρήση του API έχουν τον ίδιο στόχο – την πρόσβαση στα απαιτούμενα δεδομένα. Ωστόσο, το scraping είναι μια προτιμότερη επιλογή όταν πρόκειται για συλλογή τεράστιων ποσοτήτων δεδομένων από διαφορετικούς πόρους.

Η επιλογή μεταξύ web scraping από API εξαρτάται από τους επιχειρηματικούς στόχους. Εάν χρειάζεται να συλλέγετε δεδομένα από τον ίδιο ιστότοπο συνεχώς, το API είναι η κατάλληλη επιλογή. Δεδομένα που scraper κάνει μεγάλο χρονικό διάστημα να συλλέξει, ένα API μπορεί να τα εμφανίσει με μια κλήση. Είναι ένας από τους πιο δημοφιλείς τρόπους ανταλλαγής δεδομένων μεταξύ επιχειρήσεων.

1.6 Μεγάλα Δεδομένα

Τα μεγάλα δεδομένα (big data) που είναι διαθέσιμα στον Παγκόσμιο ιστό αποτελούνται από δομημένα, ημι-δομημένα, μη δομημένα ποσοτικά και ποιοτικά δεδομένα που διανέμονται με τη μορφή ιστοσελίδων, πινάκων HTML, βάσεων δεδομένων, email και άλλες μορφές δεδομένων. Η αξιοποίηση των μεγάλων δεδομένων απαιτεί την αντιμετώπιση ορισμένων τεχνικών ζητημάτων που σχετίζονται με τον όγκο, την ποικιλία, την ταχύτητα και την ακρίβεια των δεδομένων. Πρώτον, τα δεδομένα στον Ιστό συχνά χαρακτηρίζονται από τεράστιο όγκο που μετράτε σε Zettabytes. Δεύτερον, αυτά τα τεράστια αποθετήρια δεδομένων που είναι διαθέσιμα στον Ιστό διατίθενται σε διάφορες μορφές και βασίζονται σε μια ποικιλία τεχνολογικών και κανονιστικών προτύπων. Τρίτον, τα δεδομένα στον Ιστό δεν είναι στατικά. Αλλάζουν με εξαιρετική ταχύτητα. Το τελευταίο χαρακτηριστικό των μεγάλων δεδομένων είναι η αξιοπιστία τους σαν πληροφορία. Λόγω των ανοιχτών, εθελοντικών και συχνά ανώνυμων αλληλεπιδράσεων στον Ιστό, υπάρχει μια αβεβαιότητα που σχετίζεται με τη διαθεσιμότητα και την ποιότητα των δεδομένων.

1.7 Ανάλυση Δεδομένων

Η ανάλυση δεδομένων είναι η διεξοδική μελέτη ενός συνόλου πληροφοριών, στόχος του οποίου είναι η εξαγωγή συμπερασμάτων που επιτρέπουν σε μια εταιρεία ή οντότητα να λάβει απόφαση. Δηλαδή, αναφερόμαστε στην εξέταση και την ερμηνεία μιας βάσης δεδομένων. Αυτό, για να επιτευχθεί η επίλυση ενός προβλήματος ή ερώτησης. Κατά τη διάρκεια αυτής της ανάλυσης, τα δεδομένα μπορούν να ανταλλαχθούν, για παράδειγμα, για τη λήψη στατιστικών δεικτών. Πρέπει να σημειωθεί ότι πρόκειται για μια διαδικασία επιστήμης δεδομένων που λαμβάνει χώρα μετά τη συλλογή των πληροφοριών. Δηλαδή, αυτή η ανάλυση περιλαμβάνει όλα τα εργαλεία που μπορούμε να χρησιμοποιήσουμε για τη μελέτη μιας βάσης δεδομένων, συμπεριλαμβανομένων οπτικών όπως το ιστόγραμμα, το διάγραμμα ράβδων, το γράφημα πίτας, μεταξύ άλλων.

Η ανάλυση δεδομένων μπορεί να είναι δύο τύπων:

- **Ποσοτικός: Οι πληροφορίες είναι αριθμητικές από τις οποίες μπορούν να συγκεντρωθούν ακριβή στατιστικά στοιχεία. Για παράδειγμα, οι βαθμοί που αποκτήθηκαν από τους μαθητές μιας τάξης κατά το τελευταίο εξάμηνο.**
- **Ποιοτικός: Είναι πληροφορίες που λαμβάνονται από μια βάση δεδομένων που συνήθως παρουσιάζεται σε μορφή κειμένου. Για παράδειγμα, μια ομάδα-στόχος όπου οι συμμετέχοντες έχουν ζητήσει τη γνώμη τους για ένα νέο προϊόν.**

Για την ανάλυση δεδομένων υπάρχουν διαφορετικά εργαλεία που προέρχονται από τομείς σπουδών όπως στατιστικά, οικονομετρικά ή μαθηματικά. Έτσι, για παράδειγμα, θα μπορούσαμε να χρησιμοποιήσουμε στατιστικές μετρήσεις όπως ο μέσος όρος, η τυπική απόκλιση ή ο διάμεσος για να λάβουμε πληροφορίες σχετικά με τη συμπεριφορά μιας μεταβλητής. Από την πλευρά της, η οικονομετρία μας παρέχει βασικά εργαλεία όπως η ανάλυση παλινδρόμησης. Σε αυτές τις γραμμές, μπορούμε επίσης να χρησιμοποιήσουμε γραφικά που παρέχουν οπτικές πληροφορίες. Για παράδειγμα, από ένα ιστόγραμμα.

Ωστόσο, αξίζει να σημειωθεί ότι η ανάλυση δεδομένων δεν είναι χωρίς τους περιορισμούς της. Αυτό καθώς υπάρχουν μεταβλητές που είναι δύσκολο να ποσοτικοποιηθούν με ακρίβεια. Αυτός είναι ο λόγος για τον οποίο στην ανάλυση δεδομένων είναι συνηθισμένο να μιλάμε ως προς την πιθανότητα.

Η ανάλυση δεδομένων μπορεί να έχει διαφορετικές εφαρμογές, τόσο για εταιρείες όσο και για κρατικούς οργανισμούς ή για εκείνους με μη κερδοσκοπικούς στόχους. Για παράδειγμα, μια οντότητα που επιδιώκει να μειώσει τον παιδικό υποσιτισμό σε μια χώρα θα αξιολογεί συνεχώς τα ποσοστά αναιμίας των παιδιών σε ένα συγκεκριμένο ηλικιακό εύρος. Ομοίως, μια εταιρεία μπορεί να αναλύσει τα δεδομένα ικανοποίησης που δείχνουν οι πελάτες της. Αυτό, αφού πραγματοποίησε μια έρευνα για όλα τα άτομα που προσέλαβαν τις υπηρεσίες τους τον προηγούμενο μήνα. Με αυτόν τον τρόπο, μπορείτε να λάβετε αποφάσεις για την επιχειρηματική σας στρατηγική. Η ανάλυση δεδομένων γίνεται σχετική σε περιόδους Big Data, που είναι τόσο μεγάλα σύνολα δεδομένων που υπερβαίνουν την ικανότητα των παραδοσιακών εφαρμογών υπολογιστών να τις χειρίζονται σε εύλογο χρονικό διάστημα. Σήμερα, οι εταιρείες μπορούν να έχουν τεράστιες βάσεις δεδομένων, για παράδειγμα, όταν δημιουργούν εφαρμογές στις οποίες έχουν πρόσβαση όλοι οι πελάτες και το κοινό-στόχο τους.

1.8 Νομιμότητα και ηθική του Scraping

Ενώ πολλά εργαλεία και τεχνολογίες είναι διαθέσιμα για να βοηθήσουν τους ερευνητές με το Web Scraping, η νομιμότητα του Web Scraping εξακολουθεί να είναι μια «γκρίζα περιοχή» στο νομικό πεδίο.

Εδώ ορίζουμε τη νομιμότητα ως συμμόρφωση με τους ισχύοντες νόμους και τις νομικές θεωρίες. Προς το παρόν δεν υπάρχει νομοθετικό σώμα που απευθύνεται απευθείας στο Web Scraping. Καθοδηγείται από ένα σύνολο σχετικών, θεμελιωδών νομικών θεωριών και νόμων, όπως «παραβίαση πνευματικών δικαιωμάτων». Μερικές συγκεκριμένες λεπτομέρειες για το πώς αυτές οι θεμελιώδεις νομικές θεωρίες εφαρμόζονται στο Web Scraping παρέχονται παρακάτω.

- **Όροι χρήσης**

  Έχει υποστηριχθεί συχνά στο νομικό πεδίο ότι ένας κάτοχος ιστότοπου μπορεί να αποτρέψει αποτελεσματικά την πρόσβαση μέσω προγραμματισμού σε έναν ιστότοπο, απαγορεύοντας ρητά κάτι τέτοιο στην πολιτική «όρων χρήσης» που δημοσιεύεται στον ιστότοπο. Η μη συμμόρφωση με αυτούς τους όρους μπορεί να οδηγήσει σε «παραβίαση της σύμβασης» από την πλευρά του χρήστη του ιστότοπου. Για να διώξει κάποιον για παραβίαση των «όρων χρήσης», ο χρήστης του ιστότοπου πρέπει να συνάψει μια ρητή συμφωνία με τον κάτοχο του ιστότοπου για τη συμμόρφωση με την πολιτική «όρων χρήσης», συνήθως γίνεται όταν ο χρήστης επισκέπτεται την ιστοσελίδα και μια φόρμα αποδοχής ορών χρήσης εμφανίζεται. Επομένως, η απλή απαγόρευση της ανίχνευσης και της απόσυρσης ιστού στον ιστότοπο δεν μπορεί να αποκλείσει κάποιον από την ανίχνευση του ιστότοπου από νομική άποψη.

- **Υλικό που προστατεύεται από πνευματικά δικαιώματα**

  Το scraping και η αναδημοσίευση δεδομένων ή πληροφοριών που ανήκουν και προστατεύονται ρητά από πνευματικά δικαιώματα από τον κάτοχο του ιστότοπου μπορεί να οδηγήσει σε «παραβίαση πνευματικών δικαιωμάτων». Ωστόσο, ένας ιστότοπος δεν κατέχει απαραίτητα τα δεδομένα που δημιουργούνται από τους χρήστες του. Έτσι, μπορεί κανείς να χρησιμοποιήσει δεδομένα που προστατεύονται από πνευματικά δικαιώματα για να δημιουργήσει περιλήψεις δεδομένων που προστατεύονται από πνευματικά δικαιώματα. Τέλος, οποιοσδήποτε μπορεί να χρησιμοποιήσει υλικό που προστατεύεται από πνευματικά δικαιώματα σε περιορισμένη κλίμακα σύμφωνα με την αρχή της «δίκαιης χρήσης».

- **Σκοπός Web Scraping**

  Οποιαδήποτε παράνομη ή δόλια χρήση δεδομένων που λαμβάνονται μέσω Web Scraping απαγορεύεται από διάφορους νόμους. Στον διαδίκτυο, αυτό συμβαίνει συχνά όταν κάποιος αποκτά εν γνώσει του πρόσβαση σε "περιεχόμενο premium" και στη συνέχεια το μεταπωλεί ή συνεχίζει να έχει πρόσβαση στο περιεχόμενο μέσω μη εξουσιοδοτημένου καναλιού αφού λάβει μια επιστολή "παύσης και διακοπής" από τον ιδιοκτήτη του ιστότοπου.

- **Βλάβη στον Ιστότοπο**

  Εάν το Web Scraping υπερφορτώσει ή καταστρέψει έναν ιστότοπο ή έναν διακομιστή, τότε το άτομο που ευθύνεται για τη ζημιά μπορεί να διωχθεί σύμφωνα με τον νόμο. Ωστόσο, η ζημιά πρέπει να είναι υλική και να αποδεικνύεται εύκολα στο δικαστήριο, προκειμένου ο κάτοχος του διακομιστή να δικαιούται οικονομική αποζημίωση.

Ενώ οι υπάρχοντες νόμοι και οι νομικές θεωρίες έχουν εφαρμοστεί στο Web Scraping τόσο στα δικαστήρια όσο και στη νομική βιβλιογραφία, η ηθική του Web Scraping δεν έχει εξεταστεί από προηγούμενη βιβλιογραφία. Αν και υπάρχουν πολλές προοπτικές για την ηθική, για τους σκοπούς αυτού του ερευνητικού έργου. Εκτός από την παραβίαση της υπάρχουσας νομοθεσίας, το Web Scraping μπορεί να έχει ως αποτέλεσμα ακούσια βλάβη στους ανθρώπους που σχετίζονται με έναν συγκεκριμένο

ιστότοπο, όπως οι ιδιοκτήτες ή οι πελάτες του ιστότοπου. Αυτές οι επιβλαβείς συνέπειες, εξ ορισμού, είναι δύσκολο να προβλεφθούν. Ωστόσο, ορισμένες πιθανές επιβλαβείς συνέπειες του Web Scraping συζητούνται παρακάτω.

- **Ατομικό Απόρρητο**

Ένα ερευνητικό έργο που βασίζεται σε δεδομένα που συλλέγονται από έναν ιστότοπο ενδέχεται να θέσει σε κίνδυνο άθελα το απόρρητο των ατόμων που συμμετέχουν στις δραστηριότητες που παρέχονται από τον ιστότοπο. Ακόμη και αν δεν παραβιαστεί το ατομικό απόρρητο, το πρόβλημα είναι ότι οι πελάτες ενός ιστότοπου ενδέχεται να μην έχουν συναινέσει σε χρήση των δεδομένων τους από τρίτους. Επομένως, η χρήση αυτών των δεδομένων χωρίς συναίνεση αποτελεί παραβίαση των δικαιωμάτων των υποκειμένων της έρευνας. Αυτές οι παραβιάσεις απορρήτου και δικαιωμάτων μπορούν να οδηγήσουν σε σοβαρές συνέπειες για τον κάτοχο του ιστότοπου, δεδομένης της έντονης ανησυχίας για το απόρρητο στο διαδίκτυο υπό το φως των σκανδάλων απορρήτου που αφορούν τέτοιους οργανισμούς.

- **Απόρρητο και εμπορικά μυστικά**

Ακριβώς όπως τα άτομα έχουν δικαίωμα στην ιδιωτική ζωή, έτσι και οι οργανισμοί έχουν επίσης το δικαίωμα να διατηρήσουν εμπιστευτικές ορισμένες πτυχές των εργασιών τους. Το Web Scraping μπορεί να αποκαλύψει άθελος εμπορικά μυστικά ή απλώς εμπιστευτικές πληροφορίες σχετικά με τον οργανισμό που κατέχει έναν ιστότοπο. Μπορεί να αποκαλύψει ορισμένες λεπτομέρειες και, ενδεχομένως, ελαττώματα στον τρόπο αποθήκευσης των δεδομένων από τον ιστότοπο. Όλα αυτά μπορούν να βλάψουν τη φήμη της εταιρείας πίσω από τον ιστότοπο και να οδηγήσουν σε οικονομικές απώλειες.

- **Μείωση της κερδοφορίας για την εταιρία**

Εάν κάποιος αποκτήσει πρόσβαση στον ιστότοπο παραλείποντας τη διεπαφή της ιστοσελίδας που είναι κατασκευασμένη για ανθρώπους, τότε το άτομο δεν θα εκτεθεί στις διαφημίσεις που χρησιμοποιεί ο ιστότοπος για τη δημιουργία εσόδων από το περιεχόμενό του. Επιπλέον, η παρουσίαση των δεδομένων που δημιουργείται με τη βοήθεια του Web Scraping, μπορεί άμεσα ή έμμεσα να ανταγωνιστεί την επιχείρηση του ιδιοκτήτη του ιστότοπου. Όλα αυτά μπορεί να οδηγήσουν σε οικονομικές απώλειες στον ιδιοκτήτη του ιστότοπου ή, τουλάχιστον, σε άδικη διανομή της αξίας από την ιδιοκτησία των δεδομένων.

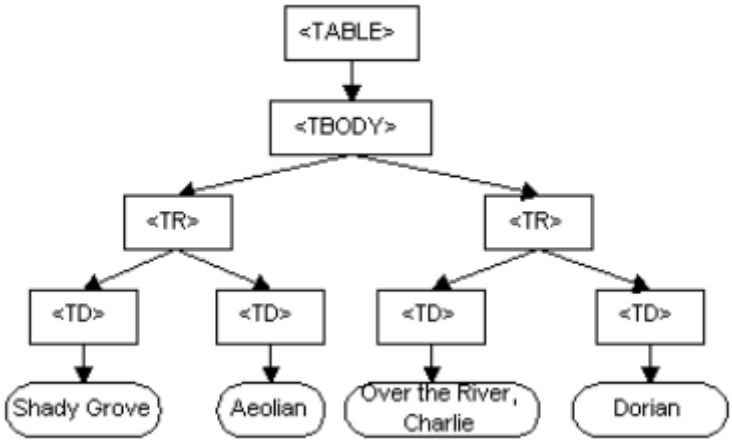
1.9 Document Object Model (DOM)

Το Document Object Model (DOM) είναι μια διεπαφή προγραμματισμού εφαρμογών (API) για έγκυρη HTML και XML. Καθορίζει τη λογική δομή των σελίδων και τον τρόπο που γίνεται η πρόσβαση και χειραγώγηση. Στην προδιαγραφή DOM, ο όρος "document" χρησιμοποιείται με την ευρεία έννοια, η XML χρησιμοποιείται ως τρόπος αναπαράστασης πολλών διαφορετικών ειδών πληροφοριών που μπορεί να είναι αποθηκευμένα σε διαφορετικά συστήματα και πολλά από αυτά θα θεωρούνταν παραδοσιακά ως δεδομένα και όχι ως έγγραφα(σελίδες). Ωστόσο, η XML παρουσιάζει αυτά τα δεδομένα ως έγγραφα και το DOM μπορεί να χρησιμοποιηθεί για τη διαχείριση αυτών των δεδομένων.

Με το DOM, οι προγραμματιστές μπορούν να δημιουργήσουν έγγραφα, να περιηγηθούν στη δομή τους, να προσθέσουν, να τροποποιήσουν ή να διαγράψουν στοιχεία και περιεχόμενο.

Οτιδήποτε βρίσκεται σε ένα έγγραφο HTML ή XML μπορεί να προσπελαστεί, να αλλάξει, να διαγραφεί ή να προστεθεί χρησιμοποιώντας το DOM με λίγες εξαιρέσεις.

Ως προδιαγραφή του W3C, ένας σημαντικός στόχος για το DOM είναι να παρέχει ένα τυπικό API που μπορεί να χρησιμοποιηθεί σε μεγάλη ποικιλία περιβαλλόντων και εφαρμογών. Το DOM έχει σχεδιαστεί για να χρησιμοποιείται με οποιαδήποτε γλώσσα προγραμματισμού.



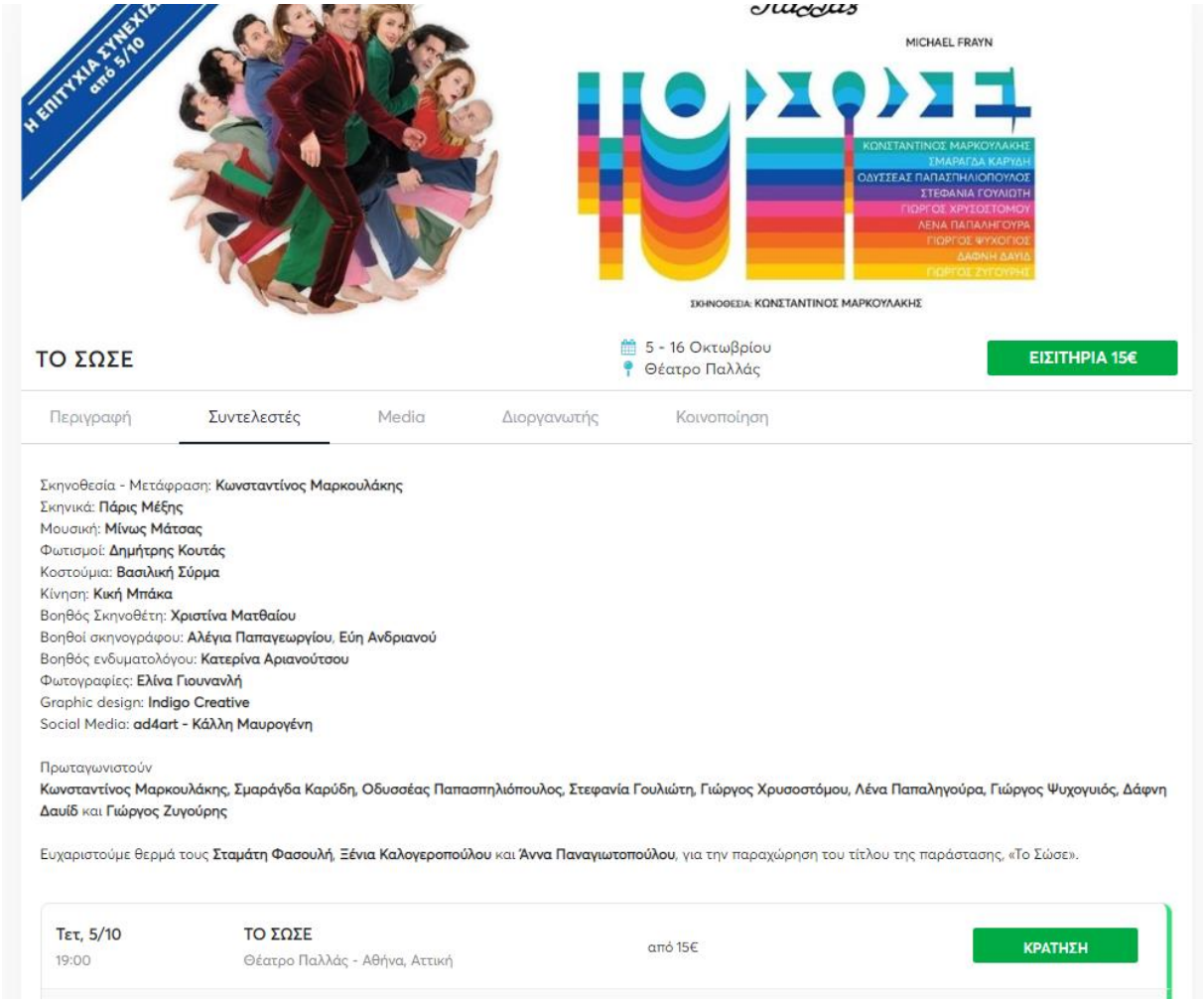
Σχήμα 1.2: DOM Tree

Στο DOM, τα έγγραφα έχουν μια λογική δομή που μοιάζει πολύ με ένα δέντρο. Κάθε έγγραφο περιέχει μηδέν ή έναν κόμβους τύπου doctype, έναν κόμβο ριζικού στοιχείου και μηδέν ή περισσότερα σχόλια ή οδηγίες επεξεργασίας. το ριζικό στοιχείο χρησιμεύει ως η ρίζα του δέντρου στοιχείων για το έγγραφο. Ωστόσο, το DOM δεν διευκρινίζει ότι τα έγγραφα πρέπει να υλοποιούνται ως δέντρο, ούτε διευκρινίζει πώς θα υλοποιηθούν οι σχέσεις μεταξύ των αντικειμένων. Το DOM είναι ένα λογικό μοντέλο που μπορεί να εφαρμοστεί με οποιονδήποτε βολικό τρόπο. Το όνομα "Μοντέλο αντικειμένου εγγράφου" επιλέχθηκε επειδή είναι ένα "μοντέλο αντικειμένου" με την παραδοσιακή αντικειμενοστραφή σχεδίαση: τα έγγραφα μοντελοποιούνται χρησιμοποιώντας αντικείμενα και το μοντέλο περιλαμβάνει όχι μόνο τη δομή ενός εγγράφου, αλλά και τη συμπεριφορά ενός εγγράφου και τα αντικείμενα από τα οποία αποτελείται. Με άλλα λόγια, οι κόμβοι στο παραπάνω διάγραμμα δεν αντιπροσωπεύουν δομή δεδομένων, αντιπροσωπεύουν αντικείμενα, τα οποία έχουν συναρτήσεις και ταυτότητα. ο DOM προσδιορίζει: Τι είναι το μοντέλο αντικειμένου εγγράφου οι διεπαφές και τα αντικείμενα που χρησιμοποιούνται για την αναπαράσταση και τον χειρισμό ενός εγγράφου τη σημασιολογία αυτών των διεπαφών και αντικειμένων - συμπεριλαμβανομένης της συμπεριφοράς και των χαρακτηριστικών των σχέσεων και συνεργασιών μεταξύ αυτών των διεπαφών και αντικειμένων. Η δομή των εγγράφων SGML παραδοσιακά αντιπροσωπεύεται από ένα αφηρημένο μοντέλο δεδομένων, όχι από ένα μοντέλο αντικειμένου. Σε ένα αφηρημένο μοντέλο δεδομένων, το μοντέλο επικεντρώνεται γύρω από τα δεδομένα. Στις αντικειμενοστραφείς γλώσσες προγραμματισμού, τα ίδια τα δεδομένα ενσωματώνονται σε αντικείμενα που κρύβουν τα δεδομένα, προστατεύοντάς τα από άμεσο εξωτερικό χειρισμό. Οι συναρτήσεις που σχετίζονται με αυτά τα αντικείμενα καθορίζουν τον τρόπο χειρισμού των αντικειμένων και αποτελούν μέρος του μοντέλου αντικειμένου.

1.10 Viva.gr - δομή θεατρικών έργων

Το viva.gr είναι μια ιστοσελίδα-πλατφόρμα που παρέχει εισιτήρια για εκδηλώσεις θεατρικού περιεχομένου και όχι μόνο. Χρησιμοποιούμε το viva.gr για την ανάκτηση όλων των πληροφορίων του συστήματος web-scraping. Ο τομέας που συλλέγει δεδομένα ο scraper είναι το θέατρο. Στην κεντρική σελίδα του τομέα «θέατρο» υπάρχει μια λίστα με όλες τις διαθέσιμες θεατρικές παραστάσεις, κάθε παράσταση έχει την αντίστοιχη της σελίδα με τής εξής πληροφορίες:

- **Τίτλος Έργου**
- **Ημερομηνίες Παραστάσεως**
- **Τοποθεσία**
- **Περιγραφή**
- **Διάρκεια Έργου**
- **Συντελεστές**
- **Media**
- **Διοργανωτής**
- **Λίστα κράτησης εισιτηρίων**



Σχήμα 1.3: Θεατρικό έργο στο Viva.gr

Στα επόμενα κεφαλαία γίνεται ανάλυση για το πώς γίνεται η ανάκτηση αυτών τον πληροφορίων και το σύστημα αποθήκευσης δεδομένων.

1.11 Επίλογος

Στο πρώτο κεφαλαίο έγινε η απαραίτητη ανάλυση, η σύγκριση με παρόμοιές τεχνολογίες, το ποιος είναι ο σωστός τρόπος της διαδικασίας scraping. Όλα αυτά ώστε να καταλάβουμε πως έγινε η υλοποίηση του συστήματος scraping με python στο επόμενο κεφάλαιο.

Κεφάλαιο 2ο: Υλοποίηση του scraping με Python

2.1 Εισαγωγή

Στο δεύτερο κεφάλαιο περιγράφεται η γλώσσα Python, οι βιβλιοθήκες που χρειάστηκαν για την υλοποίηση του συστήματος scraper, η χρήση του multithreading, η βιβλιοθήκη για το γραφικό περιβάλλον και τα windows services.

2.2 Η γλώσσα Python

Η Python είναι μια γλώσσα γενικής χρήσης, υψηλού επιπέδου. Ο σχεδιασμός του το καθιστά πολύ ευανάγνωστο, κάτι που είναι πιο σημαντικό από ό,τι ακούγεται. Κάθε πρόγραμμα υπολογιστή γράφεται μόνο μία φορά, αλλά διαβάζεται και αναθεωρείται πολλές φορές, συχνά από πολλούς ανθρώπους. Το να είναι ευανάγνωστο καθιστά επίσης ευκολότερη την εκμάθηση και την απομνημόνευση, άρα και πιο εύκολη στην εγγραφή. Σε σύγκριση με άλλες δημοφιλείς γλώσσες, η Python έχει μια ήπια καμπύλη εκμάθησης, ωστόσο έχει βάθος , με την κατάλληλη τεχνογνωσία. Η Python σας δίνει τη δυνατότητα να γράψετε ένα πρόγραμμα που είναι πολύ μικρότερο από το αντίστοιχο σε μια στατική γλώσσα. Μελέτες έχουν δείξει ότι οι προγραμματιστές τείνουν να παράγουν περίπου τον ίδιο αριθμό γραμμών κώδικα ανά ημέρα —ανεξαρτήτως γλώσσας— επομένως, η σύνταξη των μισών γραμμών κώδικα διπλασιάζει την παραγωγικότητά σας, ακριβώς έτσι. Η Python είναι το όχι και τόσο μυστικό όπλο πολλών εταιρειών. Είναι επίσης η πιο δημοφιλής γλώσσα για την αξιολόγηση των δεξιοτήτων προγραμματισμού. Και φυσικά, είναι δωρεάν. Με την Python μπορείτε να γράψετε και να χρησιμοποιήστε το οπουδήποτε, ελεύθερα Ίσως ο καλύτερος λόγος για να χρησιμοποιήσετε την Python είναι ένας απροσδόκητος λόγος: γενικά αρέσει στους ανθρώπους. Πραγματικά απολαμβάνουν τον προγραμματισμό με αυτήν, αντί να το αντιμετωπίζουν ως απλώς ένα άλλο εργαλείο για να ολοκληρώσουν τα πράγματα. Συχνά θα πουν ότι τους λείπει κάποιο χαρακτηριστικό της Python όταν χρειάζεται να εργαστούν σε άλλη γλώσσα. Και αυτό είναι που διαχωρίζει την Python από τίς υπόλοιπες γλώσσες προγραμματισμού. Η Python δεν είναι η καλύτερη γλώσσα για κάθε περίπτωση. Δεν είναι εγκατεστημένη παντού από προεπιλογή. Πρέπει να γίνει εγκατάσταση της Python εάν δεν την έχετε ήδη στον υπολογιστή σας. Είναι αρκετά γρήγορο για τις περισσότερες εφαρμογές, αλλά μπορεί να μην είναι αρκετά γρήγορο για μερικές από τις πιο απαιτητικές. Εάν το πρόγραμμά σας αφιερώνει τον περισσότερο χρόνο του στον υπολογισμό των πραγμάτων (ο τεχνικός όρος είναι CPU-bound), ένα πρόγραμμα γραμμένο σε C, C++ ή Java θα τρέχει γενικά πιο γρήγορα από το αντίστοιχο Python.

Η Python 2 υπάρχει για πάντα και είναι προεγκατεστημένη σε υπολογιστές Linux και Apple. Στις γλώσσες υπολογιστών, όπως και σε πολλούς άλλους τομείς, ορισμένα λάθη είναι αισθητικά και διορθώνονται εύκολα, ενώ άλλα είναι δύσκολα. Οι σκληρές επιδιορθώσεις δεν είναι συμβατές: τα νέα προγράμματα που γράφτηκαν με αυτές δεν θα λειτουργούν στο παλιό σύστημα Python και τα παλιά προγράμματα που γράφτηκαν πριν από την επιδιόρθωση δεν θα λειτουργούν στο νέο σύστημα. Ο δημιουργός της Python (Guido van Rossum) και άλλοι αποφάσισαν να συνδυάσουν τις σκληρές επιδιορθώσεις και να το ονομάσουν Python 3. Η Python 2 είναι το παρελθόν και η Python 3 είναι το μέλλον. Το σύστημα web-scraping χρησιμοποιεί Python 3.

Η Python εκτελεί τον κώδικα γραμμή προς γραμμή, δηλαδή, η Python είναι μια γλώσσα διερμηνέα. Όπως γνωρίζουμε ο υπολογιστής δεν μπορεί να καταλάβει τη γλώσσα μας, μπορεί να καταλάβει μόνο τη γλώσσα μηχανής, δηλαδή τη δυαδική γλώσσα. Έτσι, ο διερμηνέας Python μετατρέπει τον κώδικα

που γράφτηκε στη γλώσσα python από τον χρήστη στη γλώσσα που μπορεί να κατανοήσει το υλικό ή το σύστημα του υπολογιστή. Το κάνει συνέχεια κάθε φορά που εκτελείτε το python script.

Ο διερμηνέας Python πρώτα διαβάζει την εντολή, μετά αξιολογεί την εντολή, εκτυπώνει τα αποτελέσματα και στη συνέχεια την επαναφέρει για να διαβάσει την εντολή και γι' αυτό μόνο η Python είναι γνωστή ως REPL, δηλαδή (Read, Evaluate, Print, Loop).

Λόγοι για τους οποίους η Python είναι τόσο διαδεδομένη στις εφαρμογές λογισμικού.

- **Αναγνώσιμος και διατηρήσιμος κώδικας.**
- **Διαθέσιμο υλικό με παραδείγματα κώδικα.**
- **Συμβατό με πλατφόρμες και συστήματα.**
- **Ισχυρή βιβλιοθήκη.**
- **Πληθώρα σε frameworks και εργαλεία ανοιχτού κώδικα.**
- **Απλοποίηση της διαδικασίας σύνθετης ανάπτυξης λογισμικού.**
- **Δοκιμαστική προσέγγιση ανάπτυξης (TDD).**

Ένας γνωστός λόγος που εταιρίες χρησιμοποιούν την Python είναι για την γρήγορη δημιουργία του πρωτότυπου της εφαρμογής λογισμικού. Επίσης, την δημιουργία της εφαρμογής απευθείας από το πρωτότυπο απλώς ανακατασκευάζοντας τον κώδικα. Η Python διευκολύνει ακόμη και στην εκτέλεση ταυτόχρονης κωδικοποίησης με δοκιμές, υιοθετώντας τη δοκιμαστική προσέγγιση ανάπτυξης (test driven development). Επίσης υπάρχει η δυνατότητα να γραφτούν οι απαιτούμενες δοκιμές πριν την συγγραφή του κώδικα και να χρησιμοποιηθούν στις δοκιμές για να αξιολογείτε συνεχώς ο κώδικας της εφαρμογής. Οι δοκιμές μπορούν επίσης να χρησιμοποιηθούν για τον έλεγχο εάν η εφαρμογή πληροί προκαθορισμένες απαιτήσεις με βάση τον πηγαίο κώδικα της.

Ωστόσο, η Python, όπως και άλλες γλώσσες προγραμματισμού, έχει τις δικές της ελλείψεις. Δεν διαθέτει ορισμένες από τις ενσωματωμένες δυνατότητες που παρέχονται από άλλες σύγχρονες γλώσσες προγραμματισμού. Ως εκ τούτου, πρέπει να χρησιμοποιήσετε βιβλιοθήκες, λειτουργικές μονάδες και πλαίσια Python για να επιταχύνετε την ανάπτυξη προσαρμοσμένου λογισμικού. Επίσης, αρκετές μελέτες έχουν δείξει ότι η Python είναι πιο αργή από πολλές ευρέως χρησιμοποιούμενες γλώσσες προγραμματισμού, όπως η Java και η C++. Πρέπει να επιταχύνετε την εφαρμογή Python κάνοντας αλλαγές στον κώδικα της εφαρμογής ή χρησιμοποιώντας προσαρμοσμένο χρόνο εκτέλεσης. Αλλά μπορείτε πάντα να χρησιμοποιείτε την Python για να επιταχύνετε την ανάπτυξη λογισμικού και να απλοποιήσετε τη συντήρηση λογισμικού.

2.3 Beautiful soup

Το Beautiful Soup είναι μια βιβλιοθήκη της Python, που χρησιμοποιείται για εξαγωγή, ανάλυση και επεξεργασία πληροφοριών στο δέντρο DOM των ιστοσελίδων από αρχεία HTML και XML. Παρέχει μια σειρά συνοπτικών DOM διεπαφών, βοηθώντας τους προγραμματιστές να δημιουργήσουν γρήγορα ένα πρωτότυπο συστήματος και να αποκτήσουν δεδομένα. Επιπλέον, έχει υψηλή ευελιξία μεταξύ πλατφορμών.

Οι προγραμματιστές που χρησιμοποιούν το Beautiful Soup μπορούν, ανάλογα με τις ανάγκες τους, να εγκαταστήσουν συγκεκριμένες μηχανές ανάλυσης HTML/XML. Λαμβάνοντας ως παράδειγμα το lxml, οι χρήστες μπορούν να αρχικοποιήσουν το Beautiful Soup με την ακόλουθη κλήση: Beautiful Soup (σήμανση,"lxml"). Μετά την προετοιμασία, το Beautiful Soup αποκτά όλες τις αντίστοιχες δενδρώδεις δομές DOM για έγγραφα HTML. Στη συνέχεια, χρησιμοποιώντας μια ενσωματωμένη σειρά

συναρτήσεων διεπαφής που αντιστοιχούν στο DOM API, επισκέπτεται, αποκτά και επεξεργάζεται τις τιμές ή τα σύνολα των καθορισμένων κόμβων του δέντρου DOM.

```
markup = "<b><!--Hey, buddy. Want to buy a used parser?--></b>"
soup = BeautifulSoup(markup)
comment = soup.b.string
```

Σχήμα 2.1: Beatifulsoup παράδειγμα 1

Το Beautiful Soup δεν έχει μόνο μια ενσωματωμένη σειρά DOM API, αλλά και πολλές νέες διεπαφές που προστέθηκαν πρόσφατα που δεν διαθέτουν τα τυπικά API DOM, διευκολύνοντας τις κλήσεις για τους προγραμματιστές. Οι χρήστες μπορούν να ζητήσουν τον τύπο περιεχομένου μέσα σε μια ετικέτα:

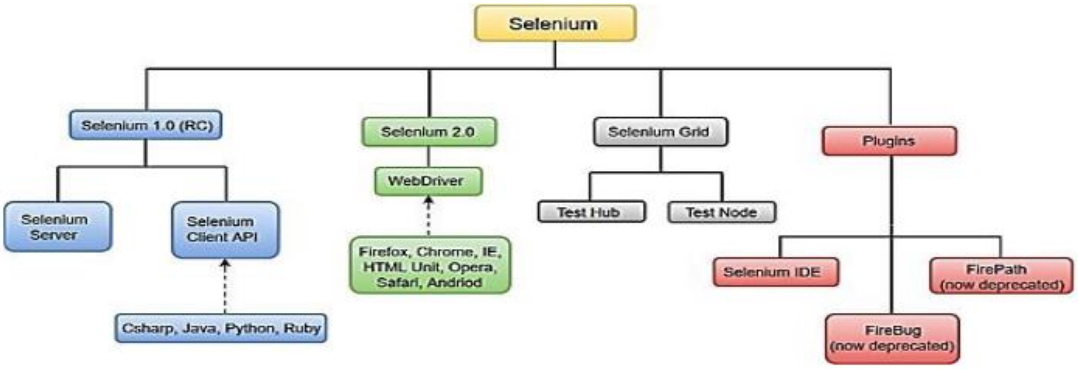
```
type(comment)
# <class 'bs4.element.Comment'>
```

Σχήμα 2.2: Beatifulsoup παράδειγμα 2

Το Beautiful Soup θα αναλύσει κάθε έγγραφο που του έχει περάσει, συμπεριλαμβανομένων όλων των ειδών εγγράφων HTML/XML και κανονικών εγγράφων. Αλλά μόνο με έγγραφα HTML και XML που πληρούν τις προδιαγραφές, το Beautiful Soup θα δημιουργήσει ένα αντίστοιχο δέντρο DOM και θα πραγματοποιήσει μια σειρά από κλήσεις API για τη λήψη των καθορισμένων δεδομένων. Χρησιμοποιώντας το HTML ως παράδειγμα, το Beautiful Soup χρησιμοποιεί πρώτα μια μηχανή ανάλυσης HTML για να μετατρέψει το έγγραφο HTML σε δέντρο DOM και, στη συνέχεια, ο χρήστης μπορεί να χρησιμοποιήσει τις παρεχόμενες λειτουργίες αναζήτησης και επεξεργασίας του Beautiful Soup για να διαχειριστεί ένα δέντρο DOM.

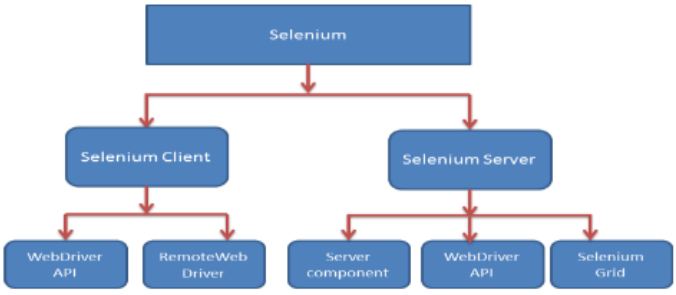
2.4 Selenium

Το Selenium είναι μια αυτοματοποιημένη δοκιμαστική σουίτα ανοιχτού κώδικα που βοηθά στην αυτοματοποιημένη δοκιμή εφαρμογών Ιστού. Το Selenium εστιάζει στην αυτοματοποιημένη δοκιμή λογισμικού που βασίζεται στο web. Το εργαλείο Selenium δημιουργήθηκε από τον Jason Huggins το 2004. Selenium είναι μια δωρεάν εφαρμογή που υποστηρίζει διαφορετικές πλατφόρμες, όπως Linux, Windows, Mac κ.λπ. Το Selenium μπορεί να χρησιμοποιηθεί για προγράμματα περιήγησης όπως το Google-Chrome, το Firefox και το Internet-Explorer. Οι γλώσσες προγραμματισμού Java, C#, Python, Ruby μπορούν να χρησιμοποιηθούν για τη σύνταξη δοκιμαστικών περιπτώσεων. Το Selenium είναι ένα εργαλείο ανοιχτού κώδικα που εστιάζει στη συμβατότητα που υποστηρίζει επίσης διαφορετικά προγράμματα περιήγησης, μορφές, γλώσσες και πλαίσια. Ανεξάρτητα από τη διαμόρφωση προϊόντος και δοκιμής, το Selenium παρέχει την απαραίτητη υποστήριξη.



Σχήμα 2.4: Αρχιτεκτονική του Selenium

Η αρχιτεκτονική του Selenium Web Tool περιλαμβάνει δύο θεμελιώδη στοιχεία: Client και Server. Στο πρόγραμμα-πελάτη περιλαμβάνεται ένας WebDriver API για αλληλεπιδράσεις ιστοσελίδων και άλλες δυνατότητες εφαρμογής. Παρέχει επίσης το πρόγραμμα οδήγησης Web απομακρυσμένης κλάσης που επικοινωνεί με τον απομακρυσμένο διακομιστή Selenium. Ο διακομιστής Selenium αποτελείται από ένα τμήμα διακομιστή που χρησιμοποιείται για τη λήψη αιτημάτων από την κλάση Remote Web Driver του πελάτη Selenium. Περιλαμβάνει επίσης το Driver API, το οποίο μπορεί να χρησιμοποιηθεί για τη δοκιμή του προγράμματος περιήγησης ιστού σε μια μηχανή διακομιστή. Το τέταρτο μέρος είναι το Selenium Grid, το οποίο χρησιμοποιεί ο Selenium Server μέσω παραμέτρων γραμμής εντολών για χαρακτηριστικά πλέγματος, έχοντας έναν κεντρικό διανομέα και διάφορους κόμβους και αγαπημένες ικανότητες προγράμματος περιήγησης. Το Grid είναι μια μέθοδος που επιτρέπει τη διεξαγωγή παράλληλων πειραμάτων σε πολλά μηχανήματα και διάφορα προγράμματα περιήγησης, γεγονός που επηρεάζει τον μειωμένο χρόνο εκτέλεσης.



Σχήμα 2.5: Τύποι API Selenium

Στο σύστημα μάς χρησιμοποιείτε το Selenium Client WebDriver API και γίνεται μέσω Firefox σε headless μορφή στο παρασκήνιο των windows, περισσότερες πληροφορίες δίνονται στο Κεφάλαιο 4.

2.4.1 Mozzila geckodriver

Το Gecko είναι μια μηχανή περιήγησης ιστού. Υπάρχουν αρκετές εφαρμογές που απαιτούν Gecko. Συγκεκριμένα, οι εφαρμογές που αναπτύσσονται από το Mozilla Foundation και την Mozilla Corporation. Το Gecko είναι επίσης μια ανάγκη για πολλά έργα λογισμικού ανοιχτού κώδικα. Το Gecko είναι γραμμένο σε C++ και JavaScript.

Το Web Browser Engine δεν είναι παρά ένα πρόγραμμα λογισμικού. Η κύρια λειτουργία αυτού του προγράμματος είναι η συλλογή του περιεχομένου (όπως HTML, XML, εικόνες) και η μορφοποίηση των πληροφοριών (όπως CSS) και η εμφάνιση αυτού του μορφοποιημένου περιεχομένου στην οθόνη.

Εφαρμογές όπως προγράμματα περιήγησης Ιστού, προγράμματα-πελάτες ηλεκτρονικού ταχυδρομείου, προγράμματα ανάγνωσης ηλεκτρονικών βιβλίων, συστήματα ηλεκτρονικής βοήθειας κ.λπ. χρειάζονται εμφάνιση περιεχομένου Ιστού. Και για να εμφανιστεί το περιεχόμενο Ιστού, απαιτείται η μηχανή προγράμματος περιήγησης Ιστού και είναι μέρος όλων αυτών των εφαρμογών. Υπάρχουν διαφορετικές μηχανές προγράμματος περιήγησης Ιστού για κάθε πρόγραμμα περιήγησης Ιστού.

Πίνακας 2.1: Προγράμματα περιήγησης

Εφαρμογή	Μηχανή περιήγησης ιστού
Internet Explorer	Trident
Firefox	Gecko
Safari	WebKit
Chrome	Blink
Oper	Presto

Το GeckoDriver είναι σε ζεύξη σύνσεσης με το πρόγραμμα περιήγησης Firefox για τα script στο Selenium. Το GeckoDriver είναι ένας διακομιστής μεσολάβησης που βοηθά στην επικοινωνία με τα προγράμματα περιήγησης που βασίζονται σε Gecko (Firefox), για τα οποία παρέχει το HTTP API.

Ο Firefox (έκδοση 47 και νεότερη) έχει κάνει κάποιες αλλαγές σε αυτό και για κάποιους λόγους ασφαλείας, δεν επιτρέπει σε κανένα πρόγραμμα οδήγησης τρίτου κατασκευαστή να αλληλεπιδρά απευθείας με τα προγράμματα περιήγησης. Το Selenium3 μπορεί να αλληλεπιδράσει απευθείας με το πρόγραμμα περιήγησης Firefox χρησιμοποιώντας έναν διακομιστή μεσολάβησης, που δεν είναι παρά το GeckoDriver.

Για την εγκατάσταση του geckodriver στο σύστημα scraper υπάρχει ο εξής κώδικας:

```
349
350  from selenium.webdriver.firefox.options import Options as FirefoxOptions
351
352  def scrap_events(url):
353      options = FirefoxOptions()
354      options.add_argument("--headless")
355      driver = webdriver.Firefox(executable_path=r'C:\geckodriver.exe', options=options)
356      driver.get(url)
357      html = driver.page_source
358      soup = BeautifulSoup(html, 'html.parser')
359
```

Σχήμα 2.5: Κώδικας εγκατάστασης geckodriver

2.4.2 Java εναντίον Python στο Selenium

Η Java και η Python είναι πολύ δημοφιλείς γλώσσες προγραμματισμού. Στην αρχή με το Selenium χρησιμοποιήθηκε κυρίως η Java. Αλλά η Python έχει γίνει η πιο δημοφιλής γλώσσα του Selenium. Η

Python έχει πολλά πλεονεκτήματα σε σχέση με άλλες γλώσσες προγραμματισμού γενικά. Σε αντίθεση με την Java, η Python είναι εύχρηστη και πιο ελαφριά. Σε σύγκριση με άλλες γλώσσες προγραμματισμού, η Python είναι επίσης αρκετά απλή. Καθώς χρησιμοποιεί βασικές αγγλικές λέξεις-κλειδιά που είναι πολύ εύκολα κατανοητές, είναι επίσης πολύ φιλικό προς το χρήστη. Επίσης, τα συντακτικά του προβλήματα είναι πολύ λιγότερο περίπλοκα από άλλες γλώσσες προγραμματισμού. Όταν χρησιμοποιείται με το Selenium, η Python είναι ακόμη πιο χρήσιμη από την Java. Σε αντίθεση με τα προγράμματα Java, τα προγράμματα Python τρέχουν πιο γρήγορα. Η Java χρησιμοποιεί συμβατικές αγκύλες για την αρχή και το τέλος των μπλοκ, ενώ η Python χρησιμοποιεί εσοχές. Η Java χρησιμοποιεί στατική πληκτρολόγηση, ενώ η Python χρησιμοποιεί δυναμική πληκτρολόγηση. Σε γενικές γραμμές, μπορούμε να πούμε ότι η Python, σε σύγκριση με την Java, είναι πιο απλή και συμπαγής.

2.5 Regular Expressions

Η επεξεργασία συμβολοσειρών είναι μια καθημερινή ρουτίνα που οι περισσότερες εφαρμογές πρέπει να αντιμετωπίσουν με τον ένα ή τον άλλο τρόπο. Δεν πρέπει να προκαλεί έκπληξη το γεγονός ότι πολλές γλώσσες προγραμματισμού διαθέτουν τυπικές βιβλιοθήκες εξοπλισμένες με μια ποικιλία εξειδικευμένων λειτουργιών για κοινές ανάγκες χειρισμού συμβολοσειρών. Ωστόσο, καμία σταθερή λειτουργικότητα δεν θα μπορούσε να καλύψει όλες τις ανάγκες, καθώς τα φυσικά ευέλικτα δεδομένα κειμένου χρειάζονται ευέλικτες λύσεις.

Εδώ είναι χρήσιμες οι κανονικές εκφράσεις, που συχνά ονομάζονται συνοπτικά regexes. Τα Regexes είναι απλή αλλά ισχυρή γλώσσα για τον καθορισμό μοτίβων για σύνολα χορδών. Σε συνδυασμό με την αντιστοίχιση προτύπων, την εξαγωγή δεδομένων και την αντικατάσταση. Θεωρούνται τόσο σημαντικά που πολλές γλώσσες προγραμματισμού παρέχουν ενσωματωμένη υποστήριξη για κανονικές εκφράσεις. Ωστόσο, το να είναι ενσωματωμένο δεν συνεπάγεται απαραίτητα ταχύτερη επεξεργασία ή περισσότερες δυνατότητες.

Οι κανονικές εκφράσεις (regexes) είναι ένας ισχυρός μηχανισμός για την επίλυση προβλημάτων αντιστοίχισης συμβολοσειρών. Υποστηρίζονται από όλες τις σύγχρονες γλώσσες προγραμματισμού και έχει υπολογιστεί ότι εμφανίζονται σε περισσότερο από το ένα τρίτο των έργων Python και JavaScript. Ωστόσο, οι υπάρχουσες μελέτες έχουν επικεντρωθεί κυρίως σε μια πτυχή του προγραμματισμού regex: την αναγνωσιμότητα. Γνωρίζουμε λίγα για το πώς αντιλαμβάνονται και προγραμματίζουν οι προγραμματιστές τα regexes, ούτε για τις δυσκολίες που αντιμετωπίζουν. Τα regexes είναι δύσκολα. Όχι μόνο είναι δύσκολο να διαβαστούν, αλλά είναι δύσκολο να αναζητηθούν, δύσκολο να επικυρωθούν και δύσκολο να τεκμηριωθούν. Είναι επίσης δύσκολο να τελειοποιηθούν, η πλειονότητα των προγραμματιστών δεν γνωρίζει τους κρίσιμους κινδύνους ασφάλειας που μπορεί να προκύψουν κατά τη χρήση regexes και όσοι γνωρίζουν τους κινδύνους δεν τους αντιμετώπισαν με αποτελεσματικούς τρόπους.

Το BeautifulSoup και οι regexes πάνε χέρι-χέρι όταν πρόκειται για το scraping του Ιστού. Στην πραγματικότητα, οι περισσότερες συναρτήσεις που λαμβάνουν ένα όρισμα συμβολοσειράς (π.χ. find (id = "aTagIdHere")) θα λαμβάνουν επίσης μια κανονική έκφραση εξίσου. Αν θέλαμε να πάρουμε διευθύνσεις URL σε όλες τις εικόνες προϊόντων, μπορεί να φαίνεται αρκετά απλό στην αρχή: απλά πιάστε όλες τις ετικέτες εικόνας χρησιμοποιώντας το .findAll ("img"), σωστά; Αλλά υπάρχει ένα πρόβλημα. Εκτός από τις προφανείς «επιπλέον» εικόνες (π.χ. λογότυπα), οι σύγχρονοι ιστότοποι έχουν συχνά κρυφές εικόνες, κενές εικόνες που χρησιμοποιούνται για την απόσταση και τη στοίχιση στοιχείων και άλλες τυχαίες ετικέτες εικόνων που μπορεί να μην γνωρίζετε. Σίγουρα, δεν μπορείτε να υπολογίζετε ότι οι μόνες εικόνες στη σελίδα είναι εικόνες προϊόντων. Ας υποθέσουμε επίσης ότι η διάταξη της

σελίδας μπορεί να αλλάξει ή ότι, για οποιονδήποτε λόγο, δεν θέλουμε να εξαρτηθούμε από τη θέση της εικόνας στη σελίδα για να βρούμε τη σωστή ετικέτα. Αυτό μπορεί να συμβαίνει όταν προσπαθείτε να συλλάβετε συγκεκριμένα στοιχεία ή κομμάτια δεδομένων που είναι διάσπαρτα τυχαία σε έναν ιστότοπο. Ένα regex μπορεί να εισαχθεί ως οποιοδήποτε όρισμα σε ένα BeautifulSoup function, επιτρέποντάς μεγάλη ευελιξία στην εύρεση στοιχείων στόχων.

2.6 Error handling

Το error handling αναφέρεται στην πρόβλεψη, τον εντοπισμό και την επίλυση σφαλμάτων προγραμματισμού, εφαρμογής και επικοινωνίας. Ειδικά προγράμματα που αποκαλούνται error handlers, είναι διαθέσιμα σε μερικές εφαρμογές. Οι καλύτερες εφαρμογές αυτού του είδους, αν είναι δυνατόν αποτρέπουν τα σφάλματα, ανακτούν ότι μπορούν από αυτά χωρίς να τερματίσουν την εφαρμογή ή αν αποτύχουν όλα, τερματίζουν την εφαρμογή και καταγράφουν το ένα αρχείο με τις πληροφορίες του σφάλματος.

Στον προγραμματισμό, ένα συντακτικό ή λογικό σφάλμα ανάπτυξης μπορεί να αποφευχθεί. Τα συντακτικά (ή αλλιώς τυπογραφικά) λάθη, αντιμετωπίζονται με αυστηρή διόρθωση. Τα λογικά σφάλματα (αναφέρονται και απλώς ως σφάλματα), δημιουργούνται όταν ο εκτελούμενος κώδικας δεν παράγει τα επιθυμητά αποτελέσματα. Τα λογικά σφάλματα αντιμετωπίζονται καλύτερα με συχνό εντοπισμό των σφαλμάτων του προγράμματος. Μπορεί να είναι μια συνεχής διαδικασία, που περιλαμβάνει πέρα από την παραδοσιακή debugging ρουτίνα, μια πρόωρη δοκιμή πριν την επίσημη κυκλοφορία όπου μετά την κυκλοφορία να υπάρχουν σχόλια πελατών.

Ένα σφάλμα run-time, λαμβάνει μέρος κατά την εκτέλεση του προγράμματος και συνήθως συμβαίνει λόγω δυσμενών παραμέτρων ή μη έγκυρων δεδομένων κατά την εισαγωγή. Ένα παράδειγμα θα μπορούσε να είναι η έλλειψη επαρκούς μνήμης για την εκτέλεση της εφαρμογής, ή να χρησιμοποιείτε η μνήμη από κάποιο άλλο πρόγραμμα. Στο διαδίκτυο, ένα run-time σφάλμα, μπορεί να παραχθεί εξαιτίας electrical noise, από διάφορες μορφές malware ή λόγω μιας εξαιρετικά μεγάλης ζήτησης στον server. Τα run-time σφάλματα αντιμετωπίζονται ή ελαχιστοποιούνται με την χρήση error handling προγραμμάτων με την επαγρύπνηση εκ μέρους των διαχειριστών δικτύου και διακομιστών και με αντίμετρα ασφαλείας εκ μέρους των χρηστών του Διαδικτύου.

Στην γλώσσα python, τα σφάλματα μπορούν να αντιμετωπιστούν χρησιμοποιώντας την εντολή try. Το σημείο που μπορεί να δημιουργήσει εξαιρέσεις, τοποθετείται μέσα στην εντολή try. Ο κώδικας που χειρίζεται τις εξαιρέσεις γράφεται κάτω από την εντολή except. Έτσι μπορούμε να επιλέξουμε τι θα κάνει ο κώδικας αν πιάσει μια εξαίρεση.

Με την εντολή “try” μπορούμε να έχουμε οποιοδήποτε exception από την λίστα με τα exceptions για να χειριστεί διαφορετικά exceptions, όμως μόνο ένα θα εκτελεστεί σε περίπτωση που υπάρχει ένα exception. Μπορούμε να χρησιμοποιήσουμε πολλές τιμές για να καθορίσουμε πολλαπλές εξαιρέσεις στην λίστα με τα exceptions.

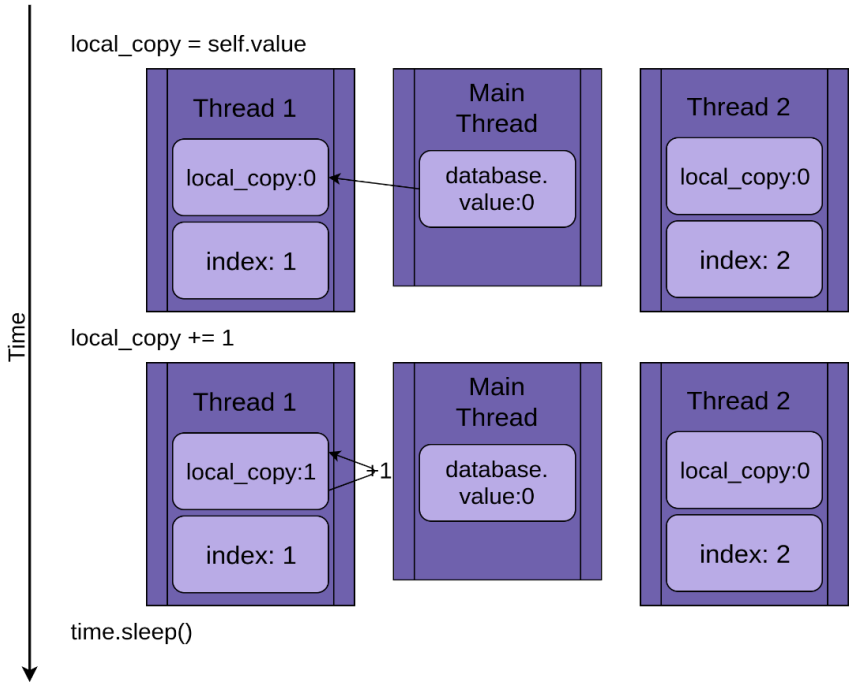
Στην γλώσσα της Python δημιουργούνται exceptions όταν υπάρχουν σφάλματα κατά τον χρόνο εκτέλεσης. Μπορούμε επίσης να παράγουμε exceptions χρησιμοποιώντας την εντολή “raise”. Μπορούμε προαιρετικά να περάσουμε τιμές στην εξαίρεση για να δούμε γιατί δημιουργήθηκε.

Σε συγκεκριμένες περιπτώσεις, μπορούμε να εκτελέσουμε ένα συγκεκριμένο μπλοκ κώδικα, εάν το μπλοκ κώδικα μέσα στο “try” εκτελούταν χωρίς σφάλματα. Για αυτές τις περιπτώσεις, μπορούμε να χρησιμοποιήσετε την προαιρετική εντολή “else” με τη δήλωση “try”. Ας δούμε ένα παράδειγμα:

Για παράδειγμα, μπορεί να είμαστε συνδεδεμένοι σε ένα απομακρυσμένο κέντρο δεδομένων μέσω του δικτύου ή να εργαζόμαστε με ένα αρχείο ή μια γραφική διεπαφή χρήστη (GUI). Σε όλες αυτές τις περιπτώσεις, πρέπει να καθαρίσουμε τους πόρους πριν σταματήσει το πρόγραμμα, είτε εκτελέστηκε με επιτυχία είτε όχι.

2.7 Πολυνημάτωση επεξεργαστή (Multithreading)

Ένα νήμα (thread) είναι μια ξεχωριστή ροή εκτέλεσης. Αυτό σημαίνει ότι στο πρόγραμμά συμβαίνουν δύο πράγματα ταυτόχρονα. Αλλά για τις περισσότερες υλοποιήσεις Python τα διαφορετικά νήματα δεν εκτελούνται στην πραγματικότητα ταυτόχρονα αλλά απλώς φαίνονται. Τα νήματα μπορεί να εκτελούνται σε διαφορετικούς επεξεργαστές, αλλά θα εκτελούνται μόνο ένα κάθε φορά. Η ταυτόχρονη εκτέλεση πολλών εργασιών απαιτεί μια μη τυπική υλοποίηση της Python, τη σύνταξη μέρους του κώδικά σας σε διαφορετική γλώσσα ή τη χρήση πολυεπεξεργασίας που συνοδεύεται από κάποια επιπλέον επιβάρυνση. Λόγω του τρόπου με τον οποίο λειτουργεί η υλοποίηση της Python CPython, το threading ενδέχεται να μην επιταχύνει όλες τις εργασίες. Αυτό οφείλεται σε αλληλεπιδράσεις με το GIL που ουσιαστικά περιορίζουν ένα νήμα Python να εκτελείται κάθε φορά. Οι εργασίες που ξοδεύουν μεγάλο μέρος του χρόνου τους περιμένοντας εξωτερικά συμβάντα είναι γενικά καλοί υποψήφιοι για νήμα. Προβλήματα που απαιτούν μεγάλους υπολογισμούς CPU και αφιερώνουν λίγο χρόνο αναμονής για εξωτερικά συμβάντα ενδέχεται να μην εκτελούνται ταχύτερα. Αυτό ισχύει για κώδικα που είναι γραμμένος σε Python και εκτελείται στην τυπική υλοποίηση CPython. Εάν τα νήματα είναι γραμμένα σε C, έχουν τη δυνατότητα να απελευθερώνουν το GIL και να εκτελούνται ταυτόχρονα. Εάν εκτελείτε μια διαφορετική εφαρμογή Python, ελέγξτε με την τεκμηρίωση και δείτε πώς χειρίζεται τα νήματα. Εάν εκτελείτε μια τυπική υλοποίηση Python, γράφοντας μόνο σε Python και υπάρχει πρόβλημα με τη δέσμευση της CPU, θα πρέπει να ελέγξετε τη λειτουργική μονάδα πολλαπλής επεξεργασίας. Η αρχιτεκτονική του προγράμματός σας για χρήση νήματος μπορεί επίσης να προσφέρει κέρδη στη σαφήνεια του σχεδιασμού.



Σχήμα 2.6: Παράδειγμα με 2 threads

Η τυπική βιβλιοθήκη Python παρέχει το threading. Για να ξεκινήσει ένα ξεχωριστό νήμα, χρειάζεται ένα στιγμιότυπο Thread και στη συνέχεια .start():

```
120
121     from threading import *
122
123     def threading():
124         t1 = Thread(target=begin_productions_scraping)
125         t1.start()
126
```

Σχήμα 2.7: Απόκομμα κώδικα threading

Ο λόγος που χρειάστηκε threads το σύστημα είναι γιατί εκτελούνται παράλληλα πολλές συναρτήσεις όπως:

- **Παράθυρο διεπαφής χρήσης**
- **Συνάρτηση scraping**
- **Διάφορες λειτουργίες μεσώ της διεπαφής**

2.8 Διεπαφή Χρήστη με χρήση της βιβλιοθήκης tkinter

Η Python έχει πολλά πλαίσια GUI, αλλά το Tkinter είναι το μόνο πλαίσιο που είναι ενσωματωμένο στην τυπική βιβλιοθήκη Python. Το Tkinter έχει πολλά δυνατά σημεία. Είναι cross-platform, επομένως ο ίδιος κώδικας λειτουργεί σε Windows, macOS και Linux. Τα οπτικά στοιχεία αποδίδονται χρησιμοποιώντας εγγενή στοιχεία λειτουργικού συστήματος, έτσι οι εφαρμογές που έχουν δημιουργηθεί με Tkinter μοιάζουν σαν να ανήκουν στην πλατφόρμα όπου εκτελούνται. Αν και το Tkinter θεωρείται το de facto πλαίσιο Python GUI, δεν είναι χωρίς κριτική. Μια αξιοσημείωτη κριτική είναι ότι τα GUI που έχουν δημιουργηθεί με το Tkinter φαίνονται ξεπερασμένα. Εάν θέλετε μια λαμπερή, μοντέρνα διεπαφή, τότε το Tkinter μπορεί να μην είναι αυτό που ψάχνετε. Ωστόσο, το Tkinter είναι ελαφρύ και σχετικά ανώδυνο στη χρήση σε σύγκριση με άλλα πλαίσια. Αυτό το καθιστά μια συναρπαστική επιλογή για τη δημιουργία εφαρμογών GUI στην Python, ειδικά για εφαρμογές όπου η μοντέρνα γυαλάδα είναι περιττή και η κορυφαία προτεραιότητα είναι να χτίσετε γρήγορα κάτι που να είναι λειτουργικό και cross-platform.

Το θεμελιώδες στοιχείο ενός Tkinter GUI είναι το window. Τα Windows είναι τα κοντέινερ στα οποία βρίσκονται όλα τα άλλα στοιχεία GUI. Αυτά τα άλλα στοιχεία GUI, όπως πλαίσια κειμένου, ετικέτες και κουμπιά, είναι γνωστά ως γραφικά στοιχεία. Τα γραφικά στοιχεία περιέχονται στο εσωτερικό των παραθύρων.

Πίνακας 2.2: Γραφικά στοιχεία tkinter

Γραφικό στοιχείο	Περιγραφή
Label	Ένα γραφικό στοιχείο που χρησιμοποιείται για την εμφάνιση κειμένου στην οθόνη
Button	Ένα κουμπί που μπορεί να περιέχει κείμενο και μπορεί να εκτελέσει μια ενέργεια όταν πατηθεί

Entry	Ένα γραφικό στοιχείο εισαγωγής κειμένου που επιτρέπει μόνο μία γραμμή κειμένου
Text	Ένα γραφικό στοιχείο εισαγωγής κειμένου που επιτρέπει την εισαγωγή κειμένου σε πολλές γραμμές
Frame	Μια περιοχή που χρησιμοποιείται για την ομαδοποίηση σχετικών γραφικών στοιχείων ή την παροχή κενών μεταξύ γραφικών στοιχείων

2.8.1 Έλεγχος διάταξης με διαχειριστές γεωμετρίας

Η διάταξη της εφαρμογής στο Tkinter ελέγχεται με διαχειριστές γεωμετρίας (geometry managers). Το Tkinter έχει τρία:

- **.pack()**
- **.place()**
- **.grid()**

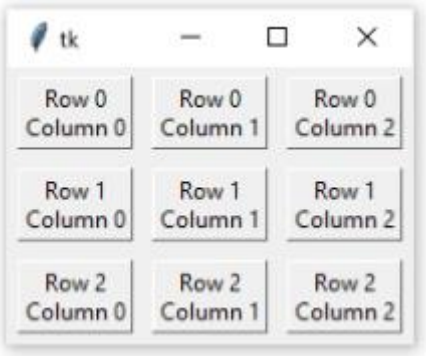
Ο διαχειριστής γεωμετρίας .pack() χρησιμοποιεί έναν αλγόριθμο για να τοποθετήσει γραφικά στοιχεία σε ένα πλαίσιο ή παράθυρο με μια καθορισμένη σειρά. Για ένα δεδομένο γραφικό στοιχείο, ο αλγόριθμος pack έχει δύο κύρια βήματα:

1. **Υπολογίζεται μια ορθογώνια περιοχή που ονομάζεται πακέτο που είναι αρκετά ψηλό (ή φαρδύ) ώστε να συγκρατεί το γραφικό στοιχείο και γεμίζει το υπόλοιπο πλάτος (ή ύψος) στο παράθυρο με κενό χώρο.**
2. **Κεντράρει το γραφικό στοιχείο στο πακέτο, εκτός εάν έχει καθοριστεί διαφορετική τοποθεσία.**

Το .pack() είναι ισχυρό, αλλά μπορεί να είναι δύσκολο να απεικονιστεί.

Ο διαχειριστής γεωμετρίας .place() είναι για να ελέγξετε την ακριβή θέση που πρέπει να καταλαμβάνει ένα γραφικό στοιχείο σε ένα παράθυρο ή ένα πλαίσιο. Πρέπει να παρέχετε δύο ορίσματα λέξεων-κλειδιών, x και y, τα οποία καθορίζουν τις συντεταγμένες x και y για την επάνω αριστερή γωνία του γραφικού στοιχείου. Τόσο το x όσο και το y μετρούνται σε pixel. Λάβετε υπόψη ότι η αρχή, όπου το x και το y είναι και τα δύο 0, είναι η επάνω αριστερή γωνία του πλαισίου ή του παραθύρου.

Ο διαχειριστής γεωμετρίας που πιθανότατα θα χρησιμοποιείτε πιο συχνά είναι το .grid(), το οποίο παρέχει όλη τη δύναμη του .pack() σε μια μορφή που είναι πιο κατανοητή. Το .grid() λειτουργεί χωρίζοντας ένα παράθυρο ή ένα πλαίσιο σε γραμμές και στήλες. Καθορίζετε τη θέση ενός γραφικού στοιχείου καλώντας το .grid() και μεταβιβάζοντας τους δείκτες γραμμής και στήλης στα ορίσματα λέξης-κλειδιού γραμμής και στήλης, αντίστοιχα. Και οι δύο δείκτες σειρών και στηλών ξεκινούν από το 0, επομένως ένας δείκτης γραμμής 1 και ένας δείκτης στήλης 2 λένε στο .grid() να τοποθετήσει ένα γραφικό στοιχείο στην τρίτη στήλη της δεύτερης σειράς.



Σχήμα 2.8: Grid διάταξη

Στο σύστημα scraping χρησιμοποιεί το Tkinter για την γραφική διεπαφή χρήστη με σύστημα διάταξης grid.

```
class GUI(object):
    def __init__(self, master):
        self.master = master
        self.master.title("Python HTML Scraping")
        # self.master.geometry('900x400')
        self.master.resizable(0, 0)
        self.Console = Text(master, height=15, width=100)
        self.scroll = Scrollbar(master, borderwidth=50)
        self.scroll.config(command=self.Console.yview)
        self.Console.grid(column=0, row=1, columnspan=2, rowspan=5, padx=(10, 0))
        self.scroll.grid(column=1, row=1, sticky=N + S + E, rowspan=5)
        self.Console.config(state=DISABLED)

    def quit(self):
        self.master.destroy()

    def write(self, *message, end="\n", sep=" "):
        self.Console.config(state=NORMAL)
        text = ""
        for item in message:
            text += "{}".format(item)
            text += sep
        text += end
        self.Console.insert(INSERT, text)
        self.Console.see("end")
        self.Console.config(state=DISABLED)

    def clearConsole(self):
        self.Console.config(state=NORMAL)
        self.Console.delete('1.0', END)
        self.Console.config(state=DISABLED)


root = Tk()
app = GUI(root)
var = IntVar()
```

Σχήμα 2.9: Απόκομμα κώδικα TKinter με Grid διάταξη 1

```
669     f0 = Frame(root)
670     f1 = Frame(root)
671     timerLabelText = Label(f0, text="Χρόνος :")
672     timerLabel = Label(f0, text="")
673     btn = Button(f0, text="Ξεκίνα το  Scraping", command=lambda: threading())
674     btn1 = Button(f1, text="Συνέχεια", command=lambda: resumeScraping())
675     btn1['state'] = DISABLED
676     btn11 = Button(f1, text="Πάυση", command=pauseScraping)
677     btn11['state'] = DISABLED
678     btn111 = Button(f1, text="Κλείσιμο", command=ExitApplication, bg='brown', fg='white')
679     btn2 = Button(root, text="Άδειασμα Πινάκων", command=deleteTablesConfirm)
680     btn3 = Button(root, text="Καθαρισμός", command=lambda: GUI.clearConsole(app))
681     btn4 = Button(root, text="Ανανέωση Συνόλων", command=lambda: getSums())
682     f0.grid(column=0, row=0, pady=(10, 10))
683     btn.pack(side="left", padx=6)
684     timerLabel.pack(side="right", padx=3)
685     timerLabelText.pack(side="right")
686     f1.grid(column=1, row=0, pady=(10, 10))
687     btn1.pack(side="left")
688     btn111.pack(side="right")
689     btn11.pack(side="right")
690     btn2.grid(column=1, row=6, sticky=E, pady=(10, 10))
691     btn3.grid(column=0, row=6, sticky=W, padx=(10, 0), pady=(10, 10))
692     btn4.grid(column=2, row=6, columnspan=2, pady=(10, 10))
693
694     label = Label(root, text="ΣΥΝΟΛΑ", font='Helvetica 16 bold')
695     label.grid(column=2, row=0, columnspan=2)
696     label2 = Label(root, text="Έργα:")
697     label3 = Label(root, text="Διοργανωτές:")
698     label4 = Label(root, text="Παραστάσεις:")
699     label5 = Label(root, text="Ηθοποιοί:")
```

Σχήμα 2.10: Απόκομμα κώδικα TKinter με Grid διάταξη 2

Σχήμα 2.21: Γραφική διεπαφή προγράμματος

2.9 Windows Service

Οι Microsoft Windows Services, δίνουν τη δυνατότητα να δημιουργήσετε μακροχρόνιες εκτελέσιμες εφαρμογές που εκτελούνται στις δικές τους περιόδους λειτουργίας των Windows. Αυτές οι υπηρεσίες μπορούν να ξεκινήσουν αυτόματα κατά την εκκίνηση του υπολογιστή, μπορούν να τεθούν σε παύση και επανεκκίνηση και να μην εμφανίσουν κανένα περιβάλλον χρήστη. Αυτές οι δυνατότητες καθιστούν τις υπηρεσίες ιδανικές για χρήση σε διακομιστή ή όποτε χρειάζεστε μακροχρόνια λειτουργικότητα που δεν παρεμβαίνει σε άλλους χρήστες που εργάζονται στον ίδιο υπολογιστή. Μπορείτε επίσης να εκτελέσετε υπηρεσίες στο πλαίσιο ασφαλείας ενός συγκεκριμένου λογαριασμού χρήστη που είναι διαφορετικός από τον συνδεδεμένο χρήστη ή τον προεπιλεγμένο λογαριασμό υπολογιστή. Για

περισσότερες πληροφορίες σχετικά με τις υπηρεσίες και τις περιόδους λειτουργίας των Windows, ανατρέξτε στην τεκμηρίωση του Windows SDK.

Μπορείτε να δημιουργήσετε εύκολα υπηρεσίες δημιουργώντας μια εφαρμογή που είναι εγκατεστημένη ως υπηρεσία. Για παράδειγμα, ας υποθέσουμε ότι θέλετε να παρακολουθείτε τα δεδομένα του μετρητή απόδοσης και να αντιδράτε σε τιμές κατωφλίου. Θα μπορούσατε να γράψετε μια εφαρμογή Windows Service που ακούει τα δεδομένα του μετρητή απόδοσης, να αναπτύξει την εφαρμογή και να αρχίσει να συλλέγει και να αναλύει δεδομένα.

2.9.1 Διάρκεια ζωής windows service

Μια υπηρεσία περνά από πολλές εσωτερικές καταστάσεις κατά τη διάρκεια της ζωής της. Αρχικά, η υπηρεσία εγκαθίσταται στο σύστημα στο οποίο θα εκτελεστεί. Αυτή η διαδικασία εκτελεί τα προγράμματα εγκατάστασης για το έργο υπηρεσίας και φορτώνει την υπηρεσία στο Services Control Manager για αυτόν τον υπολογιστή. Το Services Control Manager είναι το κεντρικό βοηθητικό πρόγραμμα που παρέχεται από τα Windows για τη διαχείριση υπηρεσιών.

Αφού φορτωθεί η υπηρεσία, πρέπει να ξεκινήσει. Η εκκίνηση της υπηρεσίας της επιτρέπει να αρχίσει να λειτουργεί. Μπορείτε να ξεκινήσετε μια υπηρεσία από τη Διαχείριση Ελέγχου Υπηρεσιών, από Εξερεύνηση διακομιστή ή από κώδικα καλώντας τη μέθοδο Start. Η μέθοδος Start μεταβιβάζει την διεργασία στη μέθοδο OnStart της εφαρμογής και επεξεργάζεται οποιονδήποτε κώδικα έχετε ορίσει εκεί.

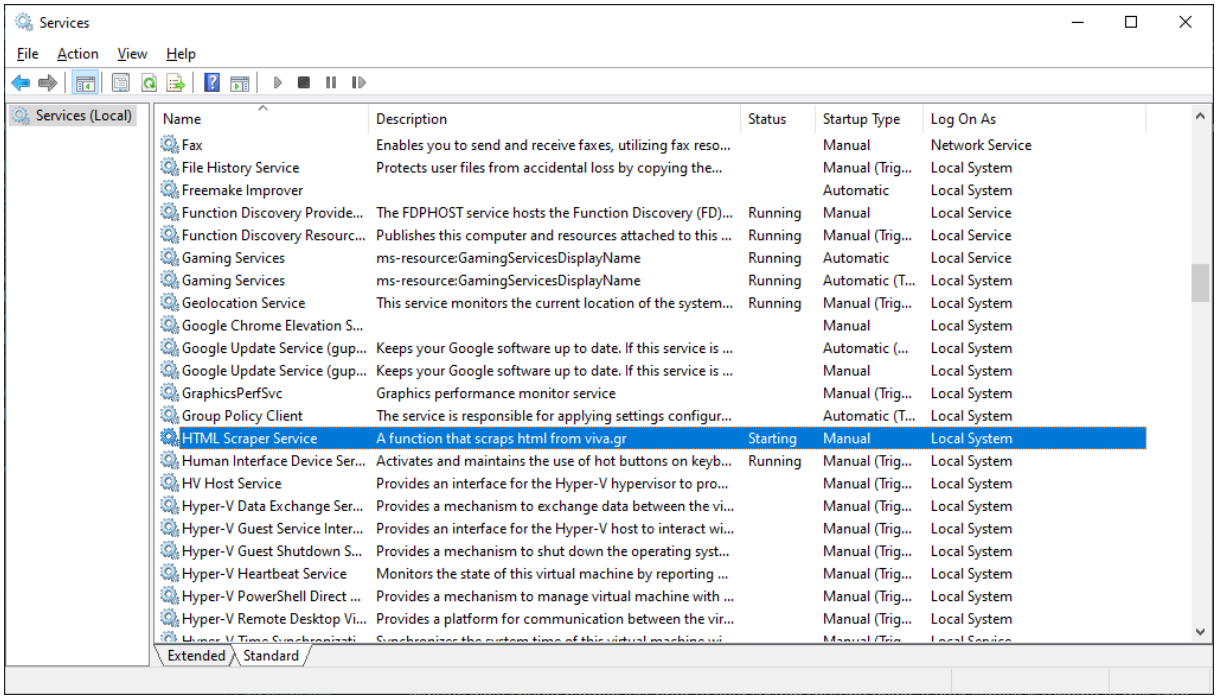
Μια υπηρεσία που εκτελείται μπορεί να υπάρχει σε αυτήν την κατάσταση επ' αόριστον μέχρι να διακοπεί ή να τεθεί σε παύση ή μέχρι να τερματιστεί η λειτουργία του υπολογιστή. Μια υπηρεσία μπορεί να υπάρχει σε μία από τις τρεις βασικές καταστάσεις: Εκτέλεση, Παύση ή Διακοπή. Η υπηρεσία μπορεί επίσης να αναφέρει την κατάσταση μιας εντολής σε εκκρεμότητα: ContinuePending, PausePending, StartPending ή StopPending. Αυτές οι καταστάσεις υποδεικνύουν ότι έχει εκδοθεί μια εντολή, όπως μια εντολή παύσης μιας υπηρεσίας που εκτελείται, αλλά δεν έχει εκτελεστεί ακόμη. Μπορείτε να ρωτήσετε την Κατάσταση για να προσδιορίσετε σε ποια κατάσταση βρίσκεται μια υπηρεσία ή να χρησιμοποιήσετε το WaitForStatus για να εκτελέσετε μια ενέργεια όταν εμφανίζεται κάποια από αυτές τις καταστάσεις.

Μπορείτε να θέσετε σε παύση, να διακόψετε ή να συνεχίσετε μια υπηρεσία από το Services Control Manager, από την Εξερεύνηση διακομιστή ή καλώντας μεθόδους σε κώδικα. Κάθε μία από αυτές τις ενέργειες μπορεί να καλέσει μια σχετική διαδικασία στην υπηρεσία (OnStop, OnPause ή OnContinue), στην οποία μπορείτε να ορίσετε πρόσθετη επεξεργασία που θα εκτελείται όταν η υπηρεσία αλλάζει κατάσταση.

2.9.2 Windows service python scraper

Στο σύστημα scraper υπάρχει δυνατότητα εκτέλεσης της διαδικασίας scraping με την εκτέλεση windows service. Η διαδικασία του scraping διαρκεί περίπου 1-3 ώρες με αποτέλεσμα να ενοχλεί η γραφική διεπαφή για τόσο μεγάλό χρονικό διάστημα. Επίσης η εκτέλεση μέσω windows service στο παρασκήνιο μπορεί να γίνει σε προγραμματισμένη μορφή αυτόματα (π.χ. 2 φορές την μέρα). Για να υλοποιηθεί αυτή η διαδικασία χρειάστηκε ένα αρχείο .py(python) και εγκατάσταση του service file μέσω windows cli.

```
C:\Users\Panos\PycharmProjects\vivaHtmlScraping>python serviceInstance.py install
Installing service HTMLScraperService
Service installed

C:\Users\Panos\PycharmProjects\vivaHtmlScraping>
```



Σχήμα 2.32: Windows services manager

Το αρχείο service δηλώνει στοιχεία όπως όνομα και περιγραφή service και τις 3 μεθόδους καταστάσεων.

```python
import time
import random
from pathlib import Path
from productions import begin_productions_scraping
from service import SMWinservice

class PythonCornerExample(SMWinservice):
    _svc_name_ = "HTMLScraperService"
    _svc_display_name_ = "HTML Scraper Service"
    _svc_description_ = "A function that scraps html from viva.gr"

    def start(self):
        self.isrunning = True

    def stop(self):
        self.isrunning = False

    def main(self):
        begin_productions_scraping()

if __name__ == '__main__':
    PythonCornerExample.parse_command_line()
```

Σχήμα 2.43: Service file python κώδικας

2.10 Github

Το Git είναι ένα κατανεμημένο σύστημα version control, που σημαίνει ότι ολόκληρη η βάση κώδικα και το ιστορικό είναι διαθέσιμα στον υπολογιστή κάθε προγραμματιστή, κάτι που επιτρέπει την εύκολη διακλάδωση και συγχώνευση.

Το version control βοηθά τους προγραμματιστές να παρακολουθούν και να διαχειρίζονται αλλαγές στον κώδικα ενός έργου λογισμικού. Καθώς ένα έργο λογισμικού μεγαλώνει, το version control γίνεται ουσιαστικό.

Αντίθετα, ο έλεγχος έκδοσης επιτρέπει στους προγραμματιστές να εργάζονται με ασφάλεια μέσω branching και merging. Με τη διακλάδωση, ένας προγραμματιστής αντιγράφει μέρος του πηγαίου κώδικα . Ο προγραμματιστής μπορεί στη συνέχεια να κάνει αλλαγές με ασφάλεια σε αυτό το μέρος του κώδικα χωρίς να επηρεάσει το υπόλοιπο έργο.

Στη συνέχεια, μόλις ο προγραμματιστής κάνει το μέρος του κώδικα να λειτουργεί σωστά, μπορεί να συγχωνεύσει αυτόν τον κώδικα ξανά στον κύριο πηγαίο κώδικα για να τον κάνει επίσημο. Στη συνέχεια, όλες αυτές οι αλλαγές παρακολουθούνται και μπορούν να επαναφερθούν αν χρειαστεί.

Το σύστημα scraping είναι αποθηκευμένο στο github στην εξής διεύθυνση: https://github.com/PanagiotisFotiou/python-bs4-html-scraping

2.11 Επίλογος

Μετά από αυτό το κεφάλαιο οπού έχουν αναλυθεί οι τεχνικές που χρειάστηκαν για το κομμάτι της συλλογής των δεδομένων, της διεπαφής του χρήστη και την εκτέλεση στο παρασκήνιο μέσω windows service. Στο επόμενο κεφάλαιο περιγράφεται το αποθηκευτικό σύστημα.

Κεφάλαιο 3ο: Αποθήκευση πληροφορίας

3.1 Εισαγωγή

Στο παρόν κεφάλαιο θα περιγράφει η SQL, οι πίνακες με όλα τα πεδία τις σχέσεις της βάσης δεδομένων του scraper, η σχεσιακή δομή και το σύστημα logger στην βάση μέσω SQL triggers. Επιπλέον θα αναλυθούν δομές βάσεων δεδομένων.

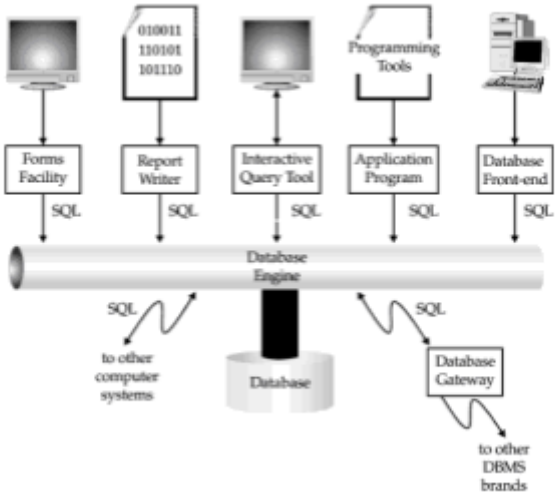
3.2 SQL

Η SQL είναι μια υπογλώσσα δεδομένων για πρόσβαση σε σχεσιακές βάσεις δεδομένων που διαχειρίζονται από συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων (RDBMS). Ο κώδικας SQL διαθέτει μια σειρά από εντολές εισαγωγής, εξαγωγής, διαγραφής και ενημέρωσης εγγραφών σε πίνακες που διαθέτει μια βάση δεδομένων. Κάθε φορά που εκτελούμε μια διαδικασία μέσα σε μια βάση δεδομένων, τότε αυτή η «πράξη» ονομάζεται ερώτημα (query). Ο τρόπος με τον οποίο κάνουμε όλες αυτές τις πράξεις ερωτημάτων είναι μέσω της γλώσσας SQL τα αρχικά της οποίας προέρχονται από τα Structure Query Language, δηλαδή σε ελεύθερη μετάφραση: «Δομημένη Γλώσσα Ερωτημάτων».

Η SQL διαθέτει ένα σύνολο από εντολές που εκτελούν διάφορες βασικές δυνατότητες όπως η εισαγωγή, η εξαγωγή, η ενημέρωση, η εύρεση συγκεκριμένων τιμών, μερικές από αυτές είναι:

- Ανάγνωση δεδομένων
- Εισαγωγή δεδομένων
- Ενημέρωση δεδομένων
- Διαγραφή δεδομένων
- Γενική διαχείριση Βάσεων Δεδομένων.

Η SQL δεν είναι η ίδια ένα σύστημα διαχείρισης βάσεων δεδομένων, ούτε είναι ένα αυτόνομο προϊόν. Δεν μπορείτε να πάτε σε ένα κατάστημα υπολογιστών και να "αγοράσετε SQL". Αντίθετα, η SQL είναι αναπόσπαστο μέρος ενός συστήματος διαχείρισης βάσεων δεδομένων, μια γλώσσα και ένα εργαλείο για την επικοινωνία με το DBMS. Το σχήμα δείχνει μερικά από τα στοιχεία ενός τυπικού DBMS και πώς η SQL λειτουργεί ως «κόλλα» που τα συνδέει μεταξύ τους.



Σχήμα 3.1: Σύστημα Database

Η μηχανή βάσης δεδομένων είναι η καρδιά του DBMS, υπεύθυνη για την πραγματική δομή, αποθήκευση και ανάκτηση των δεδομένων στη βάση δεδομένων. Δέχεται αιτήματα SQL από άλλα στοιχεία του DBMS, όπως μια εγκατάσταση φορμών, τη δυνατότητα εγγραφής αναφορών ή τη δυνατότητα διαδραστικών ερωτημάτων, από προγράμματα εφαρμογών που έχουν γραφτεί από τον χρήστη, ακόμη και από άλλα συστήματα υπολογιστών.

- **Η SQL είναι μια διαδραστική γλώσσα ερωτημάτων. Οι χρήστες πληκτρολογούν εντολές SQL σε ένα διαδραστικό πρόγραμμα SQL για να ανακτούν δεδομένα και να τα εμφανίζουν στην οθόνη, παρέχοντας ένα βολικό, εύχρηστο εργαλείο για ad hoc ερωτήματα βάσης δεδομένων.**

- **Η SQL είναι μια γλώσσα προγραμματισμού βάσης δεδομένων. Οι προγραμματιστές ενσωματώνουν εντολές SQL στα προγράμματα εφαρμογής τους για πρόσβαση στα δεδομένα μιας βάσης δεδομένων. Τόσο τα γραμμένα από τον χρήστη προγράμματα όσο και τα βοηθητικά προγράμματα βάσεων δεδομένων (όπως οι συντάκτες αναφορών και τα εργαλεία εισαγωγής δεδομένων) χρησιμοποιούν αυτήν την τεχνική για πρόσβαση στη βάση δεδομένων.**

- **Η SQL είναι μια γλώσσα διαχείρισης βάσης δεδομένων. Ο διαχειριστής της βάσης δεδομένων που είναι υπεύθυνος για τη διαχείριση ενός μικροϋπολογιστή ή μιας βάσης δεδομένων mainframe χρησιμοποιεί SQL για να καθορίσει τη δομή της βάσης δεδομένων και να ελέγξει την πρόσβαση στα αποθηκευμένα δεδομένα.**

- **Η SQL είναι μια γλώσσα πελάτη/διακομιστή. Τα προγράμματα προσωπικών υπολογιστών χρησιμοποιούν την SQL για να επικοινωνούν μέσω δικτύου με διακομιστές βάσης δεδομένων που αποθηκεύουν κοινόχρηστα δεδομένα. Αυτή η αρχιτεκτονική πελάτη/διακομιστή έχει γίνει πολύ δημοφιλής για εφαρμογές εταιρικής κατηγορίας.**

- **Η SQL είναι μια γλώσσα πρόσβασης δεδομένων στο Διαδίκτυο. Οι διακομιστές Ιστού Διαδικτύου που αλληλοεπιδρούν με εταιρικά δεδομένα και διακομιστές εφαρμογών**

Διαδικτύου χρησιμοποιούν όλοι την SQL ως τυπική γλώσσα για την πρόσβαση σε εταιρικές βάσεις δεδομένων.

- **Η SQL είναι μια κατανεμημένη γλώσσα βάσης δεδομένων. Τα συστήματα διαχείρισης κατανεμημένων βάσεων δεδομένων χρησιμοποιούν SQL για να βοηθήσουν στη διανομή δεδομένων σε πολλά συνδεδεμένα συστήματα υπολογιστών. Το λογισμικό DBMS σε κάθε σύστημα χρησιμοποιεί SQL για να επικοινωνεί με τα άλλα συστήματα, στέλνοντας αιτήματα για πρόσβαση σε δεδομένα.**

- **Η SQL είναι μια γλώσσα πύλης βάσης δεδομένων. Σε ένα δίκτυο υπολογιστών με συνδυασμό διαφορετικών προϊόντων DBMS, η SQL χρησιμοποιείται συχνά σε μια πύλη που επιτρέπει σε μια μάρκα DBMS να επικοινωνεί με μια άλλη επωνυμία. Η SQL έχει αναδειχθεί έτσι ως ένα χρήσιμο, ισχυρό εργαλείο για τη σύνδεση ανθρώπων, προγραμμάτων υπολογιστών και συστημάτων υπολογιστών με τα δεδομένα που είναι αποθηκευμένα σε μια σχεσιακή βάση δεδομένων.**

3.2.1 Δυνατότητες και πλεονεκτήματα της SQL

Η SQL είναι ταυτόχρονα μια εύκολα κατανοητή γλώσσα και ένα ολοκληρωμένο εργαλείο για τη διαχείριση δεδομένων. Ακολουθούν ορισμένα από τα κύρια χαρακτηριστικά της SQL και των δυνάμεων της αγοράς που την έχουν κάνει επιτυχημένη:

- **Ανεξαρτησία προμηθευτή**
- **Φορητότητα σε συστήματα υπολογιστών**
- **Πρότυπα SQL**
- **Έγκριση της IBM (DB2)**
- **Δέσμευση της Microsoft (ODBC και ADO)**
- **Θεμέλια σχέσης**
- **Υψηλή επίπεδο, δομή παρόμοια με τα αγγλικά**
- **Διαδραστικά, ad hoc ερωτήματα**
- **Πρόσβαση μέσω προγραμματισμού**
- **Πολλαπλές προβολές δεδομένων**
- **Πλήρης γλώσσα βάσης δεδομένων**
- **Δυναμικός ορισμός δεδομένων**
- **Αρχιτεκτονική πελάτη/διακομιστή**
- **Επεκτασιμότητα και τεχνολογία αντικειμένων**
- **Πρόσβαση στη βάση δεδομένων στο Διαδίκτυο**

Αυτοί είναι οι λόγοι για τους οποίους η SQL έχει αναδειχθεί ως το τυπικό εργαλείο για τη διαχείριση δεδομένων σε προσωπικούς υπολογιστές, μικρούς υπολογιστές και κεντρικούς υπολογιστές.

3.2.2 Αρχιτεκτονικές SQL

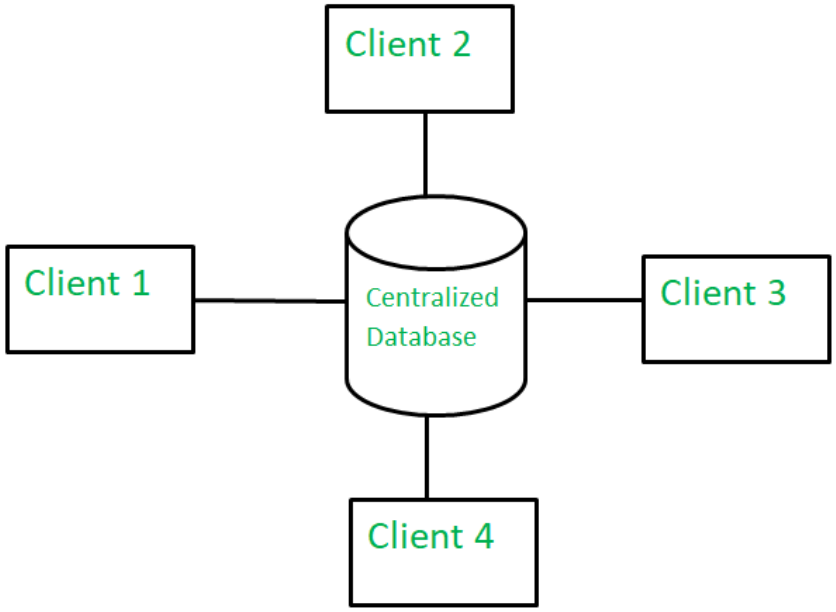
Η δραματική ανάπτυξη της δικτύωσης υπολογιστών την τελευταία δεκαετία είχε σημαντικό αντίκτυπο στη διαχείριση της βάσης δεδομένων και έδωσε στην SQL μια νέα εξέχουσα θέση. Καθώς τα δίκτυα έγιναν πιο κοινά, οι εφαρμογές που παραδοσιακά εκτελούνταν σε έναν κεντρικό μίνι υπολογιστή ή κεντρικό υπολογιστή μετακινήθηκαν σε τοπικά δίκτυα επιτραπέζιων σταθμών εργασίας και διακομιστών. Σε αυτά τα δίκτυα η SQL παίζει κρίσιμο ρόλο ως ο σύνδεσμος μεταξύ μιας εφαρμογής που εκτελείται σε έναν επιτραπέζιο σταθμό εργασίας με γραφικό περιβάλλον χρήστη και του DBMS που διαχειρίζεται τα κοινά δεδομένα σε έναν οικονομικά αποδοτικό διακομιστή. Πιο πρόσφατα, η

εκρηκτική δημοτικότητα του Διαδικτύου και του Παγκόσμιου Ιστού έχει ενισχύσει τον ρόλο του δικτύου για την SQL. Στην αναδυόμενη αρχιτεκτονική "τριών επιπέδων" του Διαδικτύου, η SQL παρέχει για άλλη μια φορά τη σύνδεση μεταξύ της λογικής της εφαρμογής (τώρα εκτελείται στο "μεσαίο επίπεδο", σε έναν διακομιστή εφαρμογής ή διακομιστή ιστού) και της βάσης δεδομένων που βρίσκεται στο back-end.

3.2.3 Κεντρικοποιημένη αρχιτεκτονική

Η παραδοσιακή αρχιτεκτονική βάσης δεδομένων που χρησιμοποιείται από τις DB2, SQL/DS και τις αρχικές βάσεις δεδομένων μικρών υπολογιστών όπως η Oracle και η Ingres. Σε αυτήν την αρχιτεκτονική, το DBMS και τα φυσικά δεδομένα βρίσκονται και τα δύο σε έναν κεντρικό μικροϋπολογιστή ή σύστημα mainframe, μαζί με το πρόγραμμα εφαρμογής που δέχεται είσοδο από το τερματικό του χρήστη και εμφανίζει δεδομένα στην οθόνη του χρήστη. Το πρόγραμμα εφαρμογής επικοινωνεί με το DBMS χρησιμοποιώντας SQL.

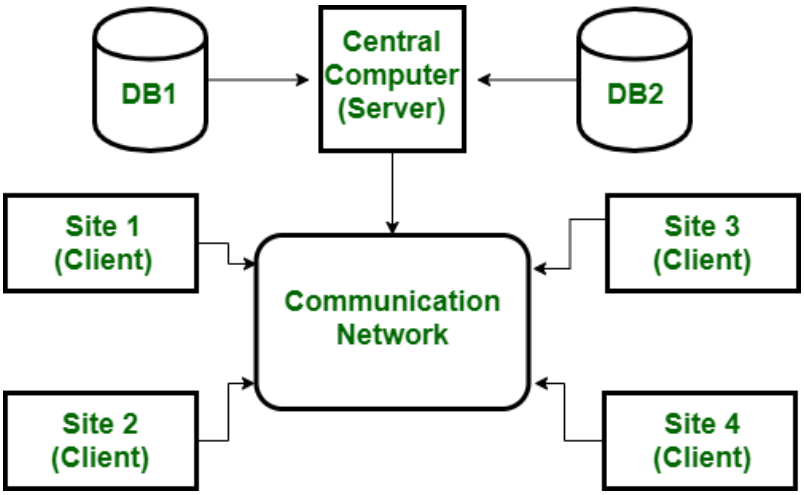
Ας υποθέσουμε ότι ο χρήστης πληκτρολογεί ένα ερώτημα που απαιτεί διαδοχική αναζήτηση μιας βάσης δεδομένων, όπως ένα αίτημα για εύρεση της μέσης ποσότητας εμπορευμάτων όλων των παραγγελιών. Το DBMS λαμβάνει το ερώτημα, σαρώνει μέσω της βάσης δεδομένων φέρνοντας κάθε εγγραφή δεδομένων από το δίσκο, υπολογίζει τον μέσο όρο και εμφανίζει το αποτέλεσμα στην οθόνη του τερματικού. Τόσο η επεξεργασία της εφαρμογής όσο και η επεξεργασία της βάσης δεδομένων πραγματοποιούνται στον κεντρικό υπολογιστή, επομένως η εκτέλεση αυτού του τύπου ερωτήματος (και στην πραγματικότητα, όλων των ειδών ερωτημάτων) είναι πολύ αποτελεσματική. Το μειονέκτημα της κεντρικής αρχιτεκτονικής είναι η επεκτασιμότητα. Καθώς προστίθενται όλο και περισσότεροι χρήστες, καθένας από αυτούς προσθέτει φόρτο εργασίας επεξεργασίας εφαρμογών στο σύστημα. Επειδή το σύστημα είναι κοινόχρηστο, κάθε χρήστης αντιμετωπίζει υποβαθμισμένη απόδοση καθώς το σύστημα φορτώνεται περισσότερο.



Σχήμα 3.2: Κεντρικοποιημένη αρχιτεκτονική

3.2.4 Αρχιτεκτονική πελάτη / διακομιστή

Σε αυτό το σχήμα, οι προσωπικοί υπολογιστές συνδυάζονται σε ένα τοπικό δίκτυο με έναν διακομιστή βάσης δεδομένων που αποθηκεύει κοινόχρηστες βάσεις δεδομένων. Οι λειτουργίες του DBMS χωρίζονται σε δύο μέρη. Τα "μπροστινά" βάσεων δεδομένων, όπως διαδραστικά εργαλεία ερωτημάτων, συντάκτες αναφορών και προγράμματα εφαρμογών, εκτελούνται στον προσωπικό υπολογιστή. Η μηχανή βάσης δεδομένων back-end που αποθηκεύει και διαχειρίζεται τα δεδομένα εκτελείται στον διακομιστή. Καθώς η αρχιτεκτονική πελάτη/διακομιστή αυξήθηκε σε δημοτικότητα κατά τη διάρκεια της δεκαετίας του 1990, η SQL έγινε η τυπική γλώσσα βάσης δεδομένων για την επικοινωνία μεταξύ των εργαλείων του μπροστινού τμήματος και του μηχανισμού υποστήριξης σε αυτήν την αρχιτεκτονική.



Σχήμα 3.3: Διαχείριση δεδομένων σε αρχιτεκτονική πελάτη/διακομιστή

Στην αρχιτεκτονική πελάτη/διακομιστή, το ερώτημα ταξιδεύει μέσω του δικτύου στον διακομιστή βάσης δεδομένων ως αίτημα SQL. Η μηχανή βάσης δεδομένων στον διακομιστή επεξεργάζεται το αίτημα και σαρώνει τη βάση δεδομένων, η οποία βρίσκεται επίσης στον διακομιστή. Όταν υπολογιστεί το αποτέλεσμα, η μηχανή βάσης δεδομένων το στέλνει πίσω στο δίκτυο ως μία μόνο απάντηση στο αρχικό αίτημα και η εφαρμογή front-end το εμφανίζει στην οθόνη του υπολογιστή.

Η αρχιτεκτονική πελάτη/διακομιστή μειώνει την κίνηση του δικτύου και διαχωρίζει το φόρτο εργασίας της βάσης δεδομένων. Λειτουργίες έντασης χρήστη, όπως ο χειρισμός εισόδου και η εμφάνιση δεδομένων, συγκεντρώνονται στον υπολογιστή του χρήστη. Λειτουργίες υψηλής έντασης δεδομένων, όπως η είσοδος/εξόδου αρχείων και η επεξεργασία ερωτημάτων, συγκεντρώνονται στον διακομιστή της βάσης δεδομένων. Το πιο σημαντικό, η γλώσσα SQL παρέχει μια καλά καθορισμένη διεπαφή μεταξύ των συστημάτων front-end και back-end, επικοινωνώντας τα αιτήματα πρόσβασης στη βάση δεδομένων με αποτελεσματικό τρόπο.

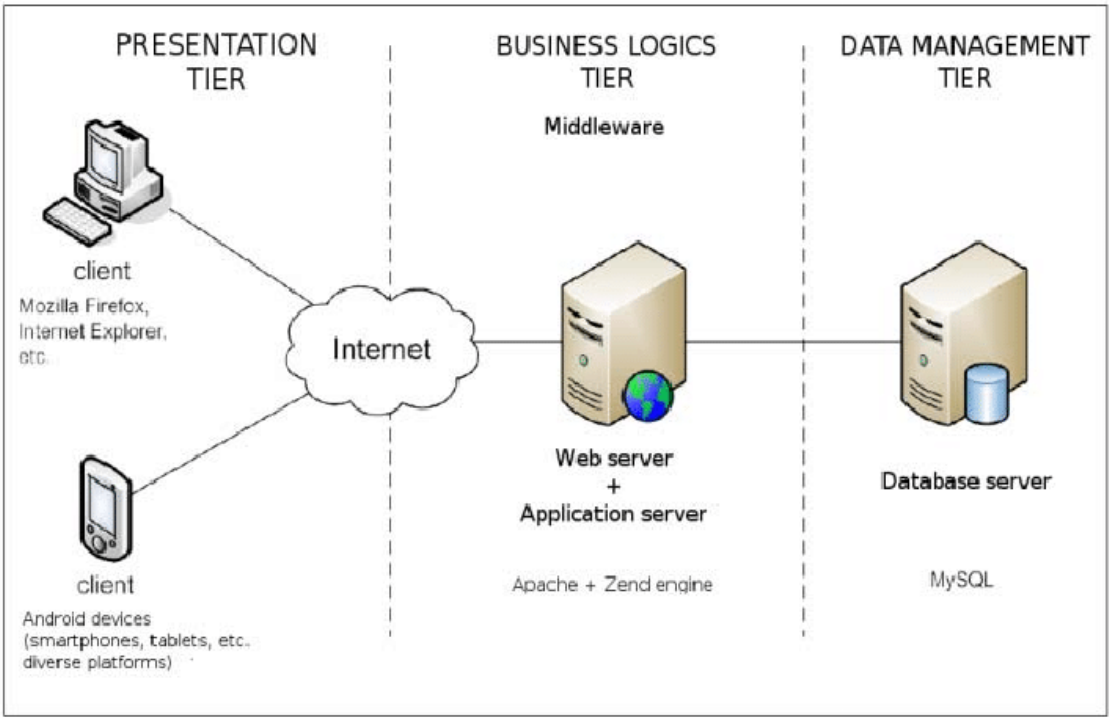
Στα μέσα της δεκαετίας του 1990, αυτά τα πλεονεκτήματα έκαναν την αρχιτεκτονική πελάτη/διακομιστή το πιο δημοφιλές σχήμα για την υλοποίηση νέων εφαρμογών. Όλα τα πιο δημοφιλή προϊόντα DBMS — Oracle, Informix, Sybase, SQL Server, DB2 και πολλά άλλα— προσφέρουν δυνατότητα πελάτη/διακομιστή. Ο κλάδος των βάσεων δεδομένων αναπτύχθηκε και περιλαμβάνει πολλές εταιρείες που προσφέρουν εργαλεία για τη δημιουργία εφαρμογών πελάτη/διακομιστή. Μερικά από αυτά προήλθαν από τις ίδιες τις εταιρείες βάσης δεδομένων. άλλα προέρχονταν από ανεξάρτητες εταιρείες.

Όπως όλες οι αρχιτεκτονικές, ο πελάτης/διακομιστής είχε τα μειονεκτήματά του. Το πιο σοβαρό από αυτά ήταν το πρόβλημα της διαχείρισης του λογισμικού εφαρμογών που τώρα διανεμήθηκε σε εκατοντάδες ή χιλιάδες επιτραπέζιους υπολογιστές αντί να εκτελείται σε έναν κεντρικό μίνι υπολογιστή ή κεντρικό υπολογιστή. Για να ενημερώσει ένα πρόγραμμα εφαρμογής σε μια μεγάλη εταιρεία, το τμήμα πληροφοριακών συστημάτων έπρεπε να ενημερώσει χιλιάδες συστήματα Η/Υ, ένα κάθε φορά. Η κατάσταση ήταν ακόμη χειρότερη εάν οι αλλαγές στο πρόγραμμα εφαρμογής έπρεπε να συγχρονιστούν με αλλαγές σε άλλες εφαρμογές ή στο ίδιο το σύστημα DBMS. Επιπλέον, με τους προσωπικούς υπολογιστές στα γραφεία των χρηστών, οι χρήστες έτειναν να προσθέτουν νέο προσωπικό λογισμικό ή να αλλάζουν τη διαμόρφωση των συστημάτων τους. Τέτοιες αλλαγές συχνά διαταράσσουν τις υπάρχουσες εφαρμογές, αυξάνοντας το φόρτο υποστήριξης. Οι εταιρείες ανέπτυξαν στρατηγικές για να αντιμετωπίσουν αυτά τα ζητήματα, αλλά μέχρι τα τέλη της δεκαετίας του 1990 υπήρχε αυξανόμενη ανησυχία σχετικά με τη δυνατότητα διαχείρισης των εφαρμογών πελάτη/διακομιστή σε μεγάλα, κατανεμημένα δίκτυα Η/Υ.

3.2.5 Αρχιτεκτονική πολλαπλών επιπέδων

Με την εμφάνιση του Διαδικτύου και ιδιαίτερα του Παγκόσμιου Ιστού, η αρχιτεκτονική της βάσης δεδομένων του δικτύου έχει κάνει ένα ακόμη βήμα. Αρχικά, ο Ιστός χρησιμοποιήθηκε για πρόσβαση στα στατικά έγγραφα και εξελίχθηκε εκτός του κόσμου της βάσης δεδομένων. Αλλά καθώς η χρήση των προγραμμάτων περιήγησης ιστού έγινε ευρέως διαδεδομένη, δεν άργησαν οι εταιρείες να σκεφτούν να τα χρησιμοποιήσουν ως έναν απλό τρόπο για να παρέχουν πρόσβαση και σε εταιρικές βάσεις δεδομένων. Για παράδειγμα, ας υποθέσουμε ότι μια εταιρεία αρχίζει να χρησιμοποιεί τον Ιστό για να παρέχει πληροφορίες προϊόντων στους πελάτες της, καθιστώντας διαθέσιμες περιγραφές προϊόντων και γραφικά στον ιστότοπο της. Ένα φυσικό επόμενο βήμα είναι να δοθεί στους πελάτες πρόσβαση στις τρέχουσες πληροφορίες διαθεσιμότητας προϊόντων μέσω της ίδιας διεπαφής προγράμματος περιήγησης ιστού. Αυτό απαιτεί τη σύνδεση του διακομιστή web με το σύστημα βάσης δεδομένων που αποθηκεύει τα (διαρκώς μεταβαλλόμενα) επίπεδα αποθέματος προϊόντων.

Οι μέθοδοι που χρησιμοποιούνται για τη σύνδεση διακομιστών Ιστού και συστημάτων DBMS έχουν εξελιχθεί γρήγορα τα τελευταία αρκετά χρόνια και έχουν συγκλίνει στην αρχιτεκτονική δικτύου τριών επιπέδων. Η διεπαφή χρήστη είναι ένα πρόγραμμα περιήγησης ιστού που εκτελείται σε υπολογιστή ή σε κάποια άλλη συσκευή "thin client" στην "μπροστινή" βαθμίδα. Επικοινωνεί με έναν διακομιστή ιστού στο "μεσαίο επίπεδο". Όταν το αίτημα χρήστη αφορά κάτι πιο περίπλοκο από μια απλή ιστοσελίδα, ο διακομιστής ιστού μεταβιβάζει το αίτημα σε έναν διακομιστή εφαρμογών του οποίου ο ρόλος είναι να χειριστεί την επιχειρηματική λογική που απαιτείται για την επεξεργασία του αιτήματος. Συχνά, το αίτημα περιλαμβάνει πρόσβαση σε μια υπάρχουσα εφαρμογή που εκτελείται σε ένα κεντρικό σύστημα ή σε μια εταιρική βάση δεδομένων. Αυτά τα συστήματα λειτουργούν στην «πίσω» βαθμίδα της αρχιτεκτονικής. Όπως και με την αρχιτεκτονική πελάτη/διακομιστή, η SQL είναι σταθερά εδραιωμένη ως η τυπική γλώσσα βάσης δεδομένων για την επικοινωνία μεταξύ του διακομιστή εφαρμογών και των βάσεων δεδομένων back-end. Όλα τα συσκευασμένα προϊόντα διακομιστή εφαρμογών παρέχουν ένα καλούμενο API που βασίζεται σε SQL για πρόσβαση στη βάση δεδομένων.



Σχήμα 3.4: Διαχείριση βάσεων σε αρχιτεκτονική Διαδικτύου τριών επιπέδων

3.3 Σχεσιακή Δομή Βάσης Δεδομένων

Μια Σχεσιακή Βάση Δεδομένων είναι ένας τύπος δομών δεδομένων που αποθηκεύει και παρέχει πρόσβαση στα σημεία δεδομένων που σχετίζονται μεταξύ τους. Οι Σχεσιακές Βάσεις Δεδομένων βασίζονται στο σχεσιακό μοντέλο, ο οποίος είναι ένας απλός, διαισθητικός τρόπος αναπαράστασης δεδομένων σε πίνακες.

Σε μια Σχεσιακή Βάση Δεδομένων, κάθε γραμμή στον πίνακα είναι μια εγγραφή με ένα μοναδικό χαρακτηριστικό, που ονομάζεται κλειδί. Οι στήλες του πίνακα περιέχουν χαρακτηριστικά των δεδομένων. Κάθε εγγραφή έχει συνήθως μια τιμή για κάθε χαρακτηριστικό, που το κάνει πιο εύκολο στον καθορισμό των σχέσεων μεταξύ των σημείων δεδομένων.

Για παράδειγμα, μια μικρή επιχείρηση θα μπορούσε να χρησιμοποιεί 2 πίνακες για την επεξεργασία παραγγελιών και για τα προϊόντα της. Ο πρώτος πίνακας θα περιέχει πληροφορίες για τους πελάτες, επομένως κάθε εγγραφή θα περιέχει ένα όνομα, μια διεύθυνση και άλλα στοιχεία επικοινωνίας. Κάθε bit πληροφοριών θα βρίσκεται στην δική του στήλη θα εκχωρεί ένα μοναδικό ID σε κάθε σειρά.

Στον δεύτερο πίνακα, δηλαδή στον πίνακα με τις παραγγελίες των πελατών, κάθε εγγραφή θα περιλαμβάνει το αναγνωριστικό του πελάτη που έκανε την παραγγελία, το προϊόν που παρήγγειλε, την ποσότητα, το επιλεγμένο μέγεθος και άλλα, και όχι μόνο το όνομα ή τα στοιχεία επικοινωνίας του πελάτη. Αυτοί οι δύο πίνακες έχουν μόνο ένα κοινό, την στήλη με το ID ή αλλιώς το primary key. Εξαιτίας αυτής της κοινής στήλης, η Σχεσιακή Βάση Δεδομένων μπορεί να δημιουργήσει μια σχέση μεταξύ αυτών των 2 πινάκων.

Όταν η εφαρμογή επεξεργασίας παραγγελιών της εταιρείας υποβάλει μια παραγγελία στην βάση δεδομένων, η βάση δεδομένων μπορεί να αντλήσει από τον πίνακα των παραγγελιών των πελατών τις

σωστές πληροφορίες σχετικά με την παραγγελία προϊόντος και να χρησιμοποιήσει το μοναδικό ID του πελάτη από τον πίνακα για να αναζητήσει πληροφορίες σχετικά με την χρέωση και την αποστολή του πελάτη στον πίνακα πληροφοριών του πελάτη. Η αποθήκη, μπορεί στην συνέχεια να "τραβήξει" το σωστό προϊόν, έπειτα ο πελάτης μπορεί να λάβει την έγκαιρη παράδοση της παραγγελίας και η εταιρεία μπορεί να πληρωθεί.

Το σχεσιακό μοντέλο σημαίνει ότι οι λογικές δομές δεδομένων, δηλαδή οι πίνακες δεδομένων, οι προβολές και τα ευρετήρια, είναι ξεχωριστά από τις φυσικές δομές αποθήκευσης. Αυτός ο διαχωρισμός σημαίνει ότι οι διαχειριστές των βάσεων δεδομένων μπορούν να διαχειριστούν την φυσική αποθήκευση των δεδομένων χωρίς να επηρεάσουν την πρόσβαση αυτά τα δεδομένα. Για παράδειγμα, η μετονομασία μιας βάσης δεδομένων, δεν μετονομάζει τους πίνακες που είναι αποθηκευμένοι σε αυτόν.

Η διάκριση μεταξύ λογικού και φυσικού εφαρμόζεται και στις βάσεις δεδομένων, οι οποίες καθορίζουν τις ενέργειες που επιτρέπουν σε εφαρμογές να διαχειρίζονται δεδομένα και τις δομές των βάσεων δεδομένων. Οι λογικές λειτουργίες επιτρέπουν στις εφαρμογές να καθορίσουν το περιεχόμενο που χρειάζονται, ενώ οι φυσικές λειτουργίες καθορίζουν τον τρόπο πρόσβασης στα δεδομένα και στην συνέχεια εκτελούν την εργασία.

Για να διασφαλιστεί ότι τα δεδομένα θα είναι πάντα ακριβές και σωστά, οι Σχεσιακές Βάσεις Δεδομένων ακολουθούν ορισμένους κανόνες ακεραιότητας. Για παράδειγμα, ένας κανόνας ακεραιότητας μπορεί να προσδιορίσει ότι διπλότυπες σειρές δεν επιτρέπονται σε κάποιον πίνακα, έτσι ώστε να εξαλειφθούν οι πιθανότητες εσφαλμένες πληροφορίες να εισαχθούν στον πίνακα.

Στα αρχικά στάδια των βάσεων δεδομένων, κάθε εφαρμογή αποθήκευε τα δεδομένα της στην δική της μοναδική δομή. Όταν οι προγραμματιστές ήθελαν να χτίσουν μια εφαρμογή από τα δεδομένα αυτά, έπρεπε να γνωρίζουν πολλά σχετικά με την συγκεκριμένη δομή για να βρουν τα δεδομένα που χρειάζονταν. Αυτές οι δομές δεδομένων δεν ήταν αποτελεσματικές, ήταν δύσκολο να διατηρηθούν και ήταν πολύ δύσκολο να βελτιστοποιηθούν για να παρέχουν καλές αποδόσεις. Το μοντέλο των Σχεσιακών Δεδομένων δημιουργήθηκαν για να αντιμετωπίσουν αυτό το πρόβλημα.

Αυτό ο μοντέλο Σχεσιακών Δεδομένων παρείχε έναν τυπικό τρόπο αναπαράστασης και αναζήτησης δεδομένων που μπορούν να χρησιμοποιηθούν από οποιαδήποτε εφαρμογή. Από την αρχή, οι προγραμματιστές αναγνώρισαν ότι η κύρια δύναμη του μοντέλου των Σχεσιακών Βάσεων Δεδομένων ήταν η χρήση πινάκων, οι οποίοι ήταν αποτελεσματικοί και είχαν έναν ευέλικτο τρόπο αποθήκευσης και πρόσβασης στα δεδομένα.

Με την πάροδο του χρόνου, μια άλλη δύναμη του σχεσιακού μοντέλου εμφανίστηκε, εφόσον οι προγραμματιστές άρχισαν να χρησιμοποιούν μια δομημένων ερωτημάτων, την γνωστή σήμερα SQL, για να γράφουν και να αναζητούν δεδομένα σε μια βάση δεδομένων. Για πολλά χρόνια, η SQL χρησιμοποιούταν ευρέος ως η γλώσσα για τα ερωτήματα των βάσεων δεδομένων. Βασισμένο στην σχεσιακή άλγεβρα, η SQL παρείχε εσωτερικά συνεπή μαθηματική γλώσσα που διευκολύνει την βελτίωση της απόδοσης όλων των ερωτημάτων των βάσεων δεδομένων. Συγκριτικά, άλλες προσεγγίσεις πρέπει να καθορίζουν μεμονωμένα ερωτήματα.

Το απλό αλλά ισχυρό σχεσιακό μοντέλο χρησιμοποιείται από οργανισμούς ανεξάρτητα από τους τύπους ή το μέγεθος τους για μια ευρεία ποικιλία πληροφοριών. Οι Σχεσιακές Βάσεις Δεδομένων χρησιμοποιούνται για την παρακολούθηση εμπορευμάτων, την επεξεργασία συναλλαγών στο ηλεκτρονικό εμπόριο, την διαχείριση τεράστιων ποσοτήτων πληροφοριών των πελατών και άλλα πολλά. Μια Σχεσιακή Βάση Δεδομένων μπορεί να θεωρηθεί για οποιαδήποτε πληροφορία χρειαστεί

στην οποία τα δεδομένα σχετίζονται μεταξύ τους και πρέπει να διαχειριστούν με έναν ασφαλή, βασισμένο σε κανόνες και συνεπή τρόπο.

Οι Σχεσιακές Βάσεις Δεδομένων υπάρχουν από το 1970. Σήμερα τα πλεονεκτήματα του σχεσιακού μοντέλου συνεχίζουν να είναι το ποιο ευρέως αποδεκτό μοντέλο για τις βάσεις δεδομένων.

Το σχεσιακό μοντέλο είναι το καλύτερο για την διατήρηση μεταξύ των εφαρμογών και των αντιγράφων των βάσεων δεδομένων, τα οποία ονομάζονται και στιγμιότυπα. Παραδείγματος χάρη όταν ένας πελάτης καταθέτει χρήματα σε ένα ATM και έπειτα βλέπει το υπόλοιπό του μέσα από ένα κινητό τηλέφωνο, ο πελάτης περιμένει να δει το καινούργιο υπόλοιπο από την στιγμή που έκανε την κατάθεση. Οι Σχεσιακές Βάσεις Δεδομένων υπερτερούν σε αυτού του είδους διατήρησης των δεδομένων, διασφαλίζοντας ότι πολλαπλά στιγμιότυπα μιας βάσης δεδομένων έχουν τα ίδια δεδομένα μέσα τους συνεχώς.

Είναι δύσκολο για άλλους τύπους βάσεων δεδομένων να διατηρήσουν αυτό το επίπεδο συνοχής με μεγάλες ποσότητες δεδομένων. Ορισμένες πρόσφατες βάσεις δεδομένων, όπως η NoSQL, μπορούν να παρέχουν μόνο την "τελική συνέπεια". Σύμφωνα με αυτήν την αρχή, όταν μια βάση δεδομένων είναι κλιμακούμενη ή όταν πολλαπλοί χρήστες αποκτούν πρόσβαση ταυτόχρονα στα ίδια δεδομένα, τα δεδομένα χρειάζονται κάποιον χρόνο για να καλύψουν την διαφορά.

Η τελική συνέπεια είναι αποδεκτή για κάποιες χρήσεις, όπως για παράδειγμα για την διατήρηση λιστών σε έναν κατάλογο προϊόντων, αλλά για κρίσιμες επιχειρηματικές δραστηριότητες όπως είναι οι συναλλαγές σε ένα καλάθι αγορών, οι Σχεσιακές Βάσεις Δεδομένων εξακολουθεί να είναι ο "χρυσός κανόνας".

Οι Σχεσιακές Βάσεις Δεδομένων διαχειρίζονται επιχειρηματικούς κανόνες και πολιτικές σε ένα πολύ αναλυτικό επίπεδο, με αυστηρές πολιτικές σχετικά με την δέσμευση, δηλαδή την μόνιμη αλλαγή στην βάση δεδομένων. Για παράδειγμα ας θεωρήσουμε μια βάση δεδομένων αποθέματος που παρακολουθεί τρία μέρη τα οποία χρησιμοποιούνται πάντα μαζί. Όταν ένα αφαιρεθεί από το απόθεμα, πρέπει να αφαιρεθούν και τα άλλα δύο. Εάν ένα από τα τρία μέρη δεν είναι διαθέσιμο, κανένα από τα τρία μέρη δεν πρέπει να αφαιρεθεί –πρέπει και τα τρία μέρη να είναι διαθέσιμα ώστε να μπορέσει η βάση δεδομένων αναλάβει οποιαδήποτε αλλαγή.

Μια Σχεσιακή Βάση Δεδομένων δεν θα αλλάξει ένα μέρος τους μέχρι να ξέρει ότι και τα τρία μέρη είναι διαθέσιμα. Αυτή η πολύπλευρη ικανότητα δέσμευσης ονομάζεται ατομικότητα. Η ατομικότητα είναι το "κλειδί" της ακρίβειας των δεδομένων στην βάση δεδομένων και την συμμόρφωση τους με τους κανόνες, τους κανονισμούς και τις πολιτικές της επιχείρησης.

Υπάρχουν τέσσερεις κρίσιμες που ορίζουν τις σχεσιακές συναλλαγές των βάσεων δεδομένων: η ατομικότητα(Atomicity), η συνέπεια(Consistency), η απομόνωση(Isolation) και η ανθεκτικότητα(Durability) και συνήθως αναφέρονται ως ACID. Παρακάτω θα αναλύσουμε περιληπτικά τι είναι το καθένα από αυτά.

- **Ατομικότητα: Ορίζει όλα τα δεδομένα που χρειάζονται για να δημιουργήσουν μια πλήρη συναλλαγή βάσεων δεδομένων.**

- **Συνέπεια: Ορίζει τους κανόνες για την διατήρηση των σημείων των δεδομένων στην σωστή κατάσταση μετά από μια συναλλαγή.**

- **Απομόνωση: Διατηρεί το αποτέλεσμα μιας συναλλαγής αόρατο στους υπόλοιπους μέχρι να ολοκληρωθεί έτσι ώστε να αποφευχθούν συγχίσεις.**

- **Ανθεκτικότητα: Διασφαλίζει ότι οι αλλαγές δεδομένων γίνονται μόνιμες μόλις η συναλλαγή ολοκληρωθεί.**

Η πρόσβαση στα δεδομένα πολλές επαναλαμβανόμενες ενέργειες. Όπως είναι ένα απλό ερώτημα για την λήψη πληροφοριών από έναν πίνακα δεδομένων μπορεί να χρειαστεί να επαναληφθεί εκατοντάδες ή χιλιάδες φορές έτσι ώστε να παραχθεί το επιθυμητό αποτέλεσμα. Αυτές οι λειτουργείες πρόσβασης δεδομένων απαιτούν κάποιου τύπου κώδικα για να αποκτήσουν πρόσβαση στην βάση δεδομένων.

Οι προγραμματιστές εφαρμογών δεν θέλουν να γράφουν καινούργιο κώδικα για αυτές τις λειτουργίες σε κάθε μια καινούργια εφαρμογή. Ευτυχώς, οι Σχεσιακές Βάσεις Δεδομένων επιτρέπουν τις αποθηκευμένες διαδικασίες, οι οποίες είναι μπλοκ από κώδικα που είναι προσβάσιμο με μια απλή κλήση μέσα από την εφαρμογή. Για παράδειγμα, μια μεμονωμένη αποθηκευμένη διαδικασία μπορεί να παρέχει συνεχές προσθήκες ετικετών για χρήστες από διάφορες εφαρμογές. Οι αποθηκευμένες διαδικασίες βοηθούν επίσης τους προγραμματιστές να διασφαλίσουν ότι ορισμένες λειτουργίες δεδομένων της εφαρμογής υλοποιούνται με συγκεκριμένο τρόπο.

Μπορεί να προκύψουν συγκρούσεις σε μια βάση δεδομένων όταν πολλαπλοί χρήστες ή εφαρμογές επιχειρούν να αλλάξουν τα ίδια δεδομένα, την ίδια χρονική στιγμή. Τεχνικές κλειδώματος και συγχρονισμού μειώνουν την πιθανότητα για συγκρούσεις ενώ ταυτόχρονά διατηρούν την ακεραιότητα των δεδομένων.

Το κλείδωμα αποτρέπει άλλους χρήστες και εφαρμογές να έχουν πρόσβαση σε δεδομένα την στιγμή που αναβαθμίζονται. Σε μερικές βάσεις δεδομένων, το κλείδωμα εφαρμόζεται σε ολόκληρους πίνακες, οι οποίοι δημιουργούν ένα αρνητικό αντίκτυπο στην απόδοση των εφαρμογών. Σε άλλες βάσεις δεδομένων, όπως είναι η Σχεσιακές Βάσεις Δεδομένων της Oracle, εφαρμόζεται κλείδωμα σε επίπεδο εγγραφής, αφήνοντας άλλες εγγραφές εντός του πίνακα διαθέσιμες, βοηθώντας έτσι την διασφάλιση καλύτερων αποδόσεων των εφαρμογών.

Ο συγχρονισμός διαχειρίζεται την δραστηριότητα όταν πολλαπλοί χρήστες ή εφαρμογές καλούν ερωτήματα ταυτόχρονα στην ίδια βάση δεδομένων. Αυτή η ικανότητα, παρέχει την σωστή πρόσβαση στους χρήστες και στις εφαρμογές σύμφωνα με πολιτικές που έχουν καθοριστεί για τον έλεγχο δεδομένων.

Το λογισμικό που χρησιμοποιείται για την αποθήκευση, την διαχείριση, και την ανάκτηση δεδομένων που είναι αποθηκευμένα σε μια Σχεσιακή Βάση Δεδομένων, ονομάζεται Σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων ή Relational Database Management System(RDBMS). Το RDBMS παρέχει μια διεπαφή μεταξύ των χρηστών, των εφαρμογών, της δομής δεδομένων, καθώς παρέχει διοικητικές λειτουργίες για την διαχείριση αποθήκευσης δεδομένων, πρόσβασης και απόδοσης.

Οι παράγοντες είναι διάφοροι που μπορεί κανείς να επιλέξει μεταξύ τύπων βάσεων δεδομένων και προϊόντων Σχεσιακών Βάσεων Δεδομένων. Το RDBMS που θα επιλεχθεί, εξαρτάται από τις ανάγκες που έχει η επιχείρηση. Μπορεί κανείς να καταλάβει το RDBMS που χρειάζεται κάνοντας τις εξής ερωτήσεις:

- **Πόσο ακριβές πρέπει να είναι τα δεδομένα; Η αποθήκευση και η ακρίβεια δεδομένων θα βασίζονται στην επιχειρηματική λογική; Θα έχουν τα δεδομένα αυστηρές απαιτήσεις όσων αφορά την ακρίβεια (για παράδειγμα, οικονομικά δεδομένα και κρατικές αναφορές);**

- **Χρειάζεται η επεκτασιμότητα; Ποια είναι η κλίμακα των δεδομένων προς διαχείριση και ποια είναι η αναμενόμενη ανάπτυξή τους; Θα χρειαστεί το μοντέλο βάσης δεδομένων να υποστηρίζει αντίγραφα βάσης δεδομένων αντικατοπτρισμού για επεκτασιμότητα; Εάν ναι, μπορεί να διατηρήσει τη συνοχή των δεδομένων σε αυτές τις περιπτώσεις;**

- **Πόσο σημαντικός είναι ο συγχρονισμός; Θα χρειάζονται πολλοί χρήστες και εφαρμογές ταυτόχρονη πρόσβαση σε δεδομένα; Υποστηρίζει το λογισμικό της βάσης δεδομένων ταυτόχρονη προστασία των δεδομένων;**

- **Ποιες είναι οι ανάγκες για απόδοση και αξιοπιστία; Χρειάζεται το προϊόν υψηλή απόδοση και αξιοπιστία; Ποιες είναι οι απαιτήσεις για την απόδοση ερωτήματος-απόκρισης;**

Με την πάροδο των χρόνων, οι Σχεσιακές Βάσεις Δεδομένων έχουν γίνει καλύτερες, γρηγορότερες, δυνατότερες και ευκολότερες στην χρήση τους. Αλλά επίσης έχουν γίνει πιο περίπλοκες και η διαχείριση των βάσεων δεδομένων έχει γίνει από καιρό δουλειά πλήρους απασχόλησης. Αντί να χρησιμοποιούν την τεχνογνωσία τους για να επικεντρωθούν στην ανάπτυξη καινοτόμων εφαρμογών οι οποίες θα προσδίδουν αξία στις επιχειρήσεις, οι προγραμματιστές έπρεπε να ξοδέψουν τον περισσότερο χρόνο τους στην διαχείριση που απαιτείται για τη βελτιστοποίηση της απόδοσης της βάσης δεδομένων.

Σήμερα, η αυτόνομη τεχνολογία χτίζεται με βάση τις δυνάμεις των σχεσιακών μοντέλων, των τεχνολογιών cloud βάσεων δεδομένων, και της εκμάθησης των μηχανών για την παράδοση νέων τύπων Σχεσιακών Βάσεων Δεδομένων. Η αυτοοδηγούμενη βάση δεδομένων, γνωστή και ως αυτόνομη βάση δεδομένων, διατηρεί την δύναμη και τα πλεονεκτήματα ενός σχεσιακού μοντέλου αλλά χρησιμοποιεί τεχνητή νοημοσύνη, την μηχανική μάθηση και τον αυτοματισμό για να βελτιώσει την απόδοση των ερωτημάτων και την διαχείριση των εργασιών.

Για παράδειγμα, για να βελτιωθεί η απόδοση των ερωτημάτων, η βάση δεδομένων αυτοοδήγησης, μπορεί να υποθέσει και να δοκιμάσει ευρετήρια για να κάνει τα ερωτήματα πιο γρήγορα και στην συνέχεια να προωθήσει τα καλύτερα προς την παραγωγή. Η αύτοοδήγητη βάση δεδομένων βελτιώνεται συνεχώς, χωρίς να χρειάζεται κάποιον ανθρώπινο παράγοντα.

Η αυτόνομη τεχνολογία απελευθερώνει τους προγραμματιστές από τα καθήκοντα της διαχείρισης των βάσεων δεδομένων. Για παράδειγμα, οι προγραμματιστές, δεν χρειάζεται πλέον από την αρχή να αποφασίσουν τις απαιτήσεις που χρειάζονται. Αντίθετα, με μια αυτοοδηγούμενη βάση δεδομένων, μπορούν να προσθέσουν αποθηκευτικό χώρο και να υπολογίσουν πόρους, όπως χρειάζεται για να αναπτυχθεί μια βάση δεδομένων. Με λίγα μόνο βήματα, οι προγραμματιστές μπορούν εύκολα να δημιουργήσουν μια αυτόνομη Σχεσιακή Βάση Δεδομένων, επιταχύνοντας τον χρόνο που χρειάζεται μια εφαρμογή για να αναπτυχθεί.

3.4 SQL triggers

Ένα SQL trigger είναι ένα αντικείμενο βάσης δεδομένων το οποίο ενεργοποιείτε όταν εμφανιστεί κάποιο συμβάν μέσα στην βάση δεδομένων. Μπορούμε να εκτελέσουμε ένα ερώτημα τύπου SQL το οποίο θα κάνει κάτι σε μια βάση δεδομένων όταν συμβεί μια αλλαγή σε κάποιο table της βάσης δεδομένων, όπως για παράδειγμα όταν μια εγγραφή εισαχθεί, ανανεωθεί ή διαγραφεί.

Δηλαδή, μπορούμε να ορίσουμε έναν κανόνα όταν ένα trigger εισάγεται στην βάση δεδομένων, όπως είναι η αύξηση των αριθμών των ιστολογιών σε έναν πίνακα αναφορών, όπου μπορούμε να δημιουργήσουμε τον κανόνα στον πίνακα Blogs στο INSERT και να ενημερώσουμε τον πίνακα Αναφορές αυξάνοντας τον αριθμό ιστολογιών σε ένα.

Υπάρχουν 2 ειδών triggers, τα Data Definition Language (DDL) triggers και τα Data Manipulation Language (DML). Τα DDL (Data Definition Language) triggers ενεργοποιούνται με DDL εντολές που ξεκινούν με Create, Alter και Drop όπως είναι οι εντολές Create_table, Create_view, drop_table, Drop_view and Alter_table.

```
1  create trigger saftey
2  on database
3  for
4  create_table,alter_table,drop_table
5  as
6  print'you can not create ,drop and alter table in this database'
7  rollback;
```

Σχήμα 3.5: Παράδειγμα κώδικα DDL

Τα DML (Data Manipulation Language) triggers ανταποκρίνονται στις εντολές DML που ξεκινούν με τις εντολές Insert, Update και Delete. Όταν πραγματοποιούμε Insert, Update ή Delete σε μια βάση δεδομένων, εμφανίζεται το παρακάτω κείμενο:

“You can not insert, update and delete table in this database”

Ακολουθεί παράδειγμα DLM trigger:

```
1  create trigger deep
2  on emp
3  for
4  insert,update,delete
5  as
6  print'you can not insert,update and delete this table i'
7  rollback;
```

Σχήμα 3.6: Παράδειγμα κώδικα DLM trigger

Υπάρχουν δύο ειδών DML triggers, τα AFTER triggers και τα INSTEAD triggers. Τα AFTER triggers εκτελούνται μετά από τις εντολές INSERT, UPDATE ή DELETE. Τα INSTEAD triggers ενημερώνουν την μηχανή της βάσης δεδομένων να εκτελεστεί το trigger αντί για το statement. Για παράδειγμα ένα INSERT trigger εκτελείτε όταν πραγματοποιείτε ένα συμβάν αντί για το statement που θα εισήγαγε τιμές στον πίνακα.

```sql
create trigger insertt
on emp
after insert
as
begin
insert into empstatus values('active')
end
```

100 %

Results | Messages

	status
1	active
2	active

Σχήμα 3.7: Παράδειγμα κώδικα ALTER Trigger

3.5 Η βάση δεδομένων του scraper

Η βάση δεδομένων είναι το τελικό προϊόν του συστήματος scraper. Όλη η πληροφορία που ανακτάτε αποθηκεύεται σε μια σχεσιακή βάση δεδομένων. Επίσης σε αυτήν την βάση το σύστημα scraper python δεν είναι το μόνο που απεργάζεται την βάση, υπάρχει παρόμοιο πρόγραμμα σε γλώσσα C# από πτυχιακή συμφοιτητή Κωνσταντίνου Γεωργιάδη με όνομα εργασίας «HTML Scrapping με χρήση .NET» το οποίο κάνει scraping από το viva.gr και αποθηκεύει πληροφορίες στην βάση. Για να υποδηλώσουμε το σύστημα scraping, σε κάθε πίνακα υπάρχει το πεδίο SystemID, αυτό το πεδίο συμπληρώνεται από τον scraper. Στην περίπτωση του python scraper το πεδίο έχει την τιμή «2». Έτσι μπορούμε να καταλάβουμε από πιο σύστημα έχει αποκτηθεί η πληροφορία.

Η βάση δεδομένων αποτελείτε από εννιά πίνακες:

- **Production**
- **Events**
- **Contributions**
- **Venue**
- **Organizer**
- **Persons**
- **Roles**
- **System**
- **changeLog**

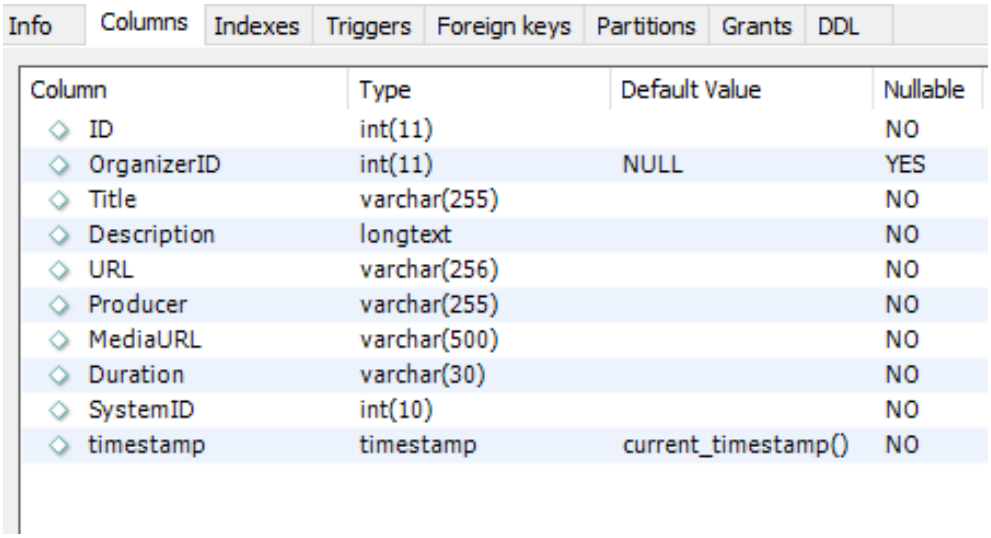
Στην συνέχεια αναλύεται ο κάθε πίνακας ξεχωριστά.

3.5.1 Ο πίνακας Production

Ο πίνακας Production αντιστοιχεί στο θεατρικό έργο, την κύρια σελίδα του post στο viva.gr. Η σελίδα αυτή διαθέτει πληροφορίες όπως τον τίτλο, την περιγραφή , τον διοργανωτή , την διάρκεια , την κύρια εικόνα. Ο πίνακας Production είναι η αφετηρία στην λογική του συστήματος.

Αναλυτικότερα τα πεδία του πίνακα:

- **ID: Το κύριο κλειδί του πίνακα – αριθμείται αυτόματα από την βάση**
- **OrganizerID: Ξένο κλειδί του πίνακα Organizer**
- **Title: Ο τίτλος του έργου**
- **Description: Η περιγραφή του έργου**
- **Url: Ο σύνδεσμος του έργου**
- **Producer: Ο διοργανωτής του έργου**
- **MediaURL: Η κύρια εικόνα στης σελίδας**
- **Duration: Η διάρκεια του έργου**
- **SystemID: Το σύστημα οπού έκανε INSERT η UPDATE την γραμμή**
- **timestamp: Η χρονική σήμανση της γραμμής – συμπληρώνεται αυτόματα από την βάση**



Column	Type	Default Value	Nullable
ID	int(11)		NO
OrganizerID	int(11)	NULL	YES
Title	varchar(255)		NO
Description	longtext		NO
URL	varchar(256)		NO
Producer	varchar(255)		NO
MediaURL	varchar(500)		NO
Duration	varchar(30)		NO
SystemID	int(10)		NO
timestamp	timestamp	current_timestamp()	NO

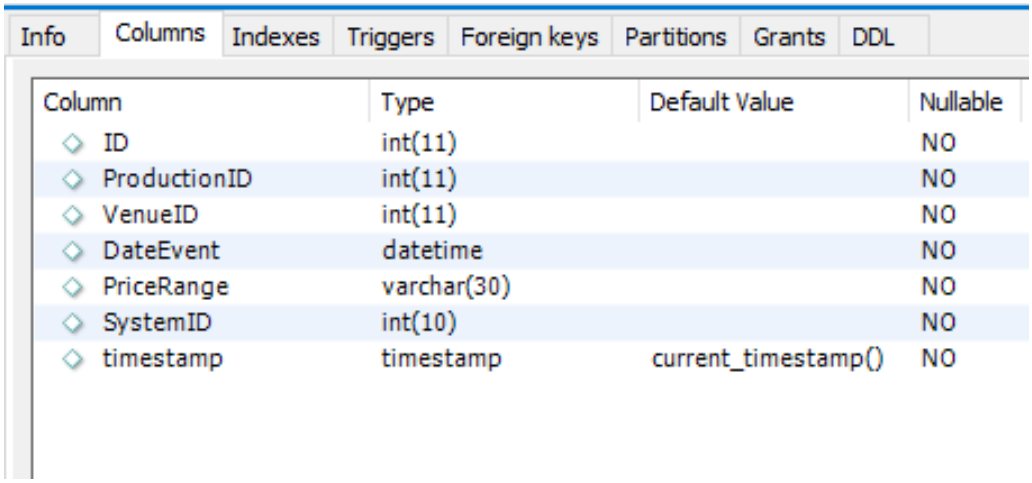
Σχήμα 3.8: Πίνακας Production

3.5.2 Ο πίνακας Events

Ο πίνακας Events διαθέτει πληροφορίες για την κάθε εκδήλωσή του έργου η αλλιώς το πρόγραμμα παραστάσεων. Μεσώ του viva.gr δίνεται η δυνατότητα κράτησης εισιτηρίου παράστασης, η κάθε παράσταση έχει μια ημερομηνία , ώρα , έργο, τοποθεσία και τιμή.

Αναλυτικότερα τα πεδία του πίνακα:

- **ID: Το κύριο κλειδί του πίνακα – αριθμείται αυτόματα από την βάση**
- **ProductionID: Ξένο κλειδί του πίνακα Production**
- **VenueID: Ξένο κλειδί του πίνακα Venue**
- **DateEvent: Η ημερομηνία της παραστάσεως**
- **PriceRange: Η τιμή η το εύρος τιμών της παράστασης**
- **SystemID: Το σύστημα οπού έκανε INSERT η UPDATE την γραμμή**
- **timestamp: Η χρονική σήμανση της γραμμής – συμπληρώνεται αυτόματα από την βάση**



Info | Columns | Indexes | Triggers | Foreign keys | Partitions | Grants | DDL

Column	Type	Default Value	Nullable
ID	int(11)		NO
ProductionID	int(11)		NO
VenueID	int(11)		NO
DateEvent	datetime		NO
PriceRange	varchar(30)		NO
SystemID	int(10)		NO
timestamp	timestamp	current_timestamp()	NO

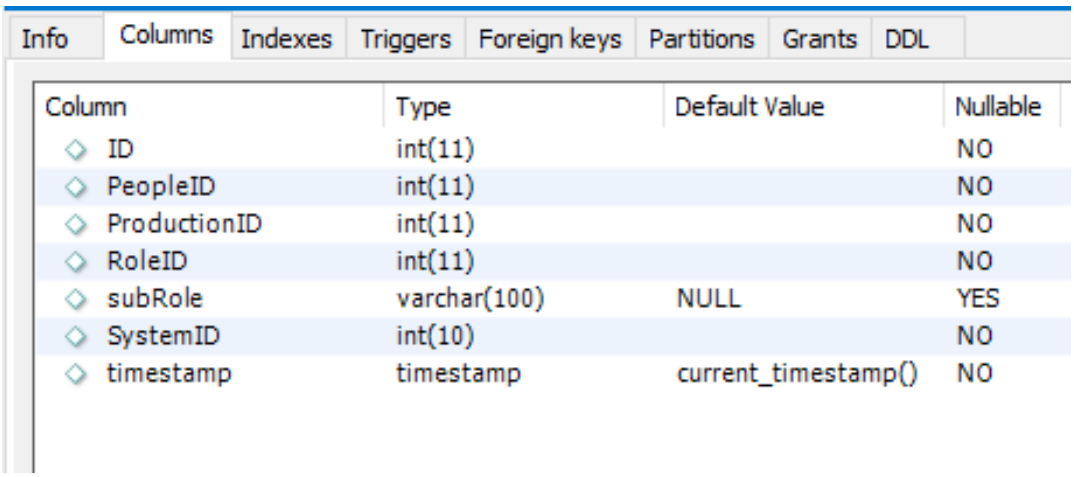
Σχήμα 3.9: Πίνακας Events

3.5.3 Ο πίνακας Contributions

Ο πίνακας Contribution είναι ένας πίνακας που συνδέει της οντότητες του Ατόμου, ρολού η συντελεστή και παραστάσεως. Με αυτόν τον πίνακα μπορούμε να καταλάβουμε ποιος έπαιξε σε ποια παράσταση με τί ρόλο η ποιος συνείσφερε σε μια παράσταση και με τί ειδικότητα. Για παράδειγμα ο 4795 έχει πρωταγωνιστεί στον Production 514 με Role 1695(Ηθοποιός), η ο 5822 έχει την Μουσική επιμέλεια στο Production 514 με ρόλο 2282(Υπεύθυνοι ήχου). Αποτελείτε κυρίως από ξένα κλειδιά.

Αναλυτικότερα τα πεδία του πίνακα:

- **ID: Το κύριο κλειδί του πίνακα – αριθμείται αυτόματα από την βάση**
- **PeopleID: Ξένο κλειδί του πίνακα Venue**
- **ProductionID: Ξένο κλειδί του πίνακα Production**
- **RoleID: Ξένο κλειδί του πίνακα Role**
- **subRole: Ο θεατρικός ρόλος αν η ειδικότητα είναι Ηθοποιός**
- **SystemID: Το σύστημα οπού έκανε INSERT η UPDATE την γραμμή**
- **timestamp: Η χρονική σήμανση της γραμμής – συμπληρώνεται αυτόματα από την βάση**



Info | Columns | Indexes | Triggers | Foreign keys | Partitions | Grants | DDL

Column	Type	Default Value	Nullable
ID	int(11)		NO
PeopleID	int(11)		NO
ProductionID	int(11)		NO
RoleID	int(11)		NO
subRole	varchar(100)	NULL	YES
SystemID	int(10)		NO
timestamp	timestamp	current_timestamp()	NO

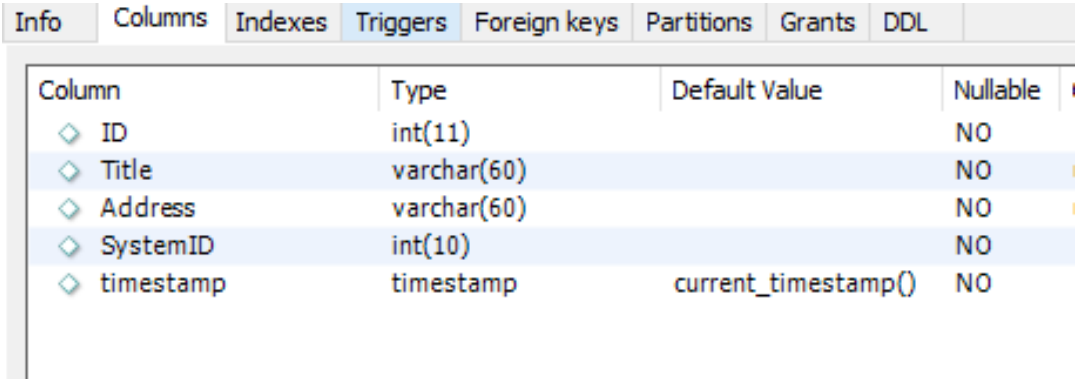
Σχήμα 3.20: Πίνακας Contributions

3.5.4 Ο πίνακας Venue

Ο πίνακας Venue παρέχει πληροφορίες σχετικά με τον χώρο εκδηλώσεων. Ο χώρος έχει ένα όνομα και μια διεύθυνση.

Αναλυτικότερα τα πεδία του πίνακα:

- **ID: Το κύριο κλειδί του πίνακα – αριθμείται αυτόματα από την βάση**
- **Title: Ο τίτλος του χώρου**
- **Address: Η διεύθυνση**
- **SystemID: Το σύστημα οπού έκανε INSERT η UPDATE την γραμμή**
- **timestamp: Η χρονική σήμανση της γραμμής – συμπληρώνεται αυτόματα από την βάση**



Σχήμα 3.13: Πίνακας Venue

3.5.5 Ο πίνακας Organizer

Ο πίνακας Organizer παρέχει πληροφορίες για τον διοργανωτή του έργου. Στο viva.gr υπαρχεί σχετική ενότητα με τον διοργανωτή σε κάθε έργο με σχετικές πληροφορίες.

Αναλυτικότερα τα πεδία του πίνακα:

- **ID: Το κύριο κλειδί του πίνακα – αριθμείται αυτόματα από την βάση**
- **Name: Όνομα διοργανωτή**
- **Address: Διεύθυνση**
- **Town: Πόλη**
- **postcode: Τ.Κ.**
- **Phone: Τηλέφωνο**
- **Email: email**
- **Doy: Δ.Ο.Υ.**
- **Afm: Α.Φ.Μ.**
- **SystemID: Το σύστημα οπού έκανε INSERT η UPDATE την γραμμή**
- **timestamp: Η χρονική σήμανση της γραμμής – συμπληρώνεται αυτόματα από την βάση**

Info | Columns | Indexes | Triggers | Foreign keys | Partitions | Grants | DDL

Column	Type	Default Value	Nullable
ID	int(11)		NO
Name	varchar(80)		NO
Address	varchar(50)		NO
Town	varchar(100)		NO
postcode	varchar(20)		NO
Phone	varchar(30)		NO
Email	varchar(100)		NO
Doy	varchar(30)		NO
Afm	varchar(30)		NO
SystemID	int(10)		NO
timestamp	timestamp	current_timestamp()	NO

Σχήμα 3.12: Πίνακας Organizer

3.5.6 Ο πίνακας Persons

Ο πίνακα Person περιλαμβάνει όλα τα ανθρώπινα πρόσωπα που συνεισφέρουν σε κάθε έργο.

Αναλυτικότερα τα πεδία του πίνακα:

- **ID: Το κύριο κλειδί του πίνακα – αριθμείται αυτόματα από την βάση**
- **Fullname: Ονοματεπώνυμο**
- **SystemID: Το σύστημα οπού έκανε INSERT η UPDATE την γραμμή**
- **timestamp: Η χρονική σήμανση της γραμμής – συμπληρώνεται αυτόματα από την βάση**

Info | Columns | Indexes | Triggers | Foreign keys | Partitions | Grants | DDL

Column	Type	Default Value	Nullable
ID	int(11)		NO
Fullname	varchar(600)		NO
SystemID	int(10)		NO
timestamp	timestamp	current_timestamp()	NO

Σχήμα 3.43: Πίνακας Persons

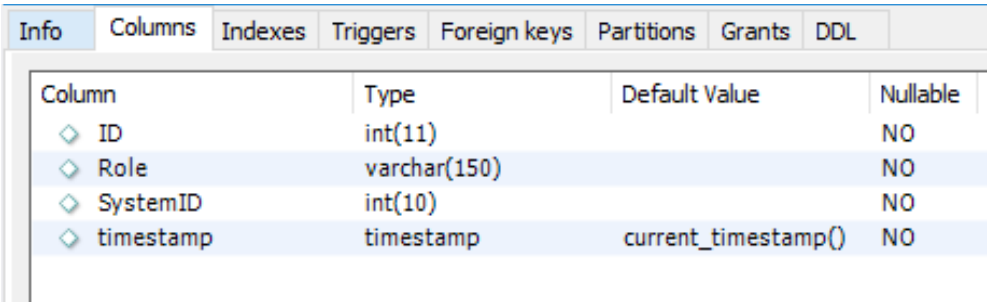
3.5.7 Ο πίνακας Roles

Ο πίνακα Roles περιλαμβάνει όλες τις επαγγελματικές θέσεις που μπορεί να βρίσκονται σε μια θεατρική παράσταση.

Αναλυτικότερα τα πεδία του πίνακα:

- **ID: Το κύριο κλειδί του πίνακα – αριθμείται αυτόματα από την βάση**

- **Role: Ειδικότητα**
- **SystemID: Το σύστημα οπού έκανε INSERT η UPDATE την γραμμή**
- **timestamp: Η χρονική σήμανση της γραμμής – συμπληρώνεται αυτόματα από την βάση**



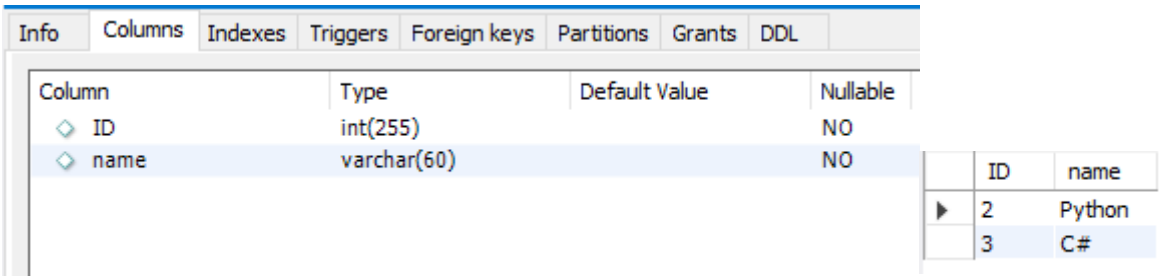
Σχήμα 3.54: Πίνακας Roles

3.5.8 Ο πίνακας System

Ο πίνακας System περιλαμβάνει τα συστήματα scraper. Υπάρχουν 2 συστήματα Python και C#.

Αναλυτικότερα τα πεδία του πίνακα:

- **ID: Το κύριο κλειδί του πίνακα – αριθμείται αυτόματα από την βάση**
- **name: Όνομα συστήματος**



Σχήμα 3.65: Πίνακας System

3.5.9 Ο πίνακας changelog

Ο πίνακας changeLog καταγράφει όλες τις κινήσεις που γίνονται στην βάση από τα συστήματα scraping όπως ένα logger. Λειτουργεί αυτόνομα μεσώ εντολών SQL Trigger, μια γραμμή συμπληρώνεται όταν κάποιο σύστημα εκτελεί την εντολή insert,update η delete σε οποιονδήποτε πίνακα για κάθε κελί.

Αναλυτικότερα τα πεδία του πίνακα:

- **ID: Το κύριο κλειδί του πίνακα – αριθμείται αυτόματα από την βάση**
- **EventType: Τύπος εντολής SQL**
- **TableName: Όνομα πίνακα**
- **Value: Τιμή**

- **CollumnName: Όνομα κελιού**
- **timestamp: Η χρονική σήμανση της γραμμής – συμπληρώνεται αυτόματα από την βάση**

Info | Columns | Indexes | Triggers | Foreign keys | Partitions | Grants | DDL

Column	Type	Default Value	Nullable
ID	int(11)		NO
EventType	varchar(100)		NO
TableName	varchar(100)		NO
Value	varchar(200)		NO
CollumnName	varchar(100)		NO
timestamp	timestamp	current_timestamp()	NO

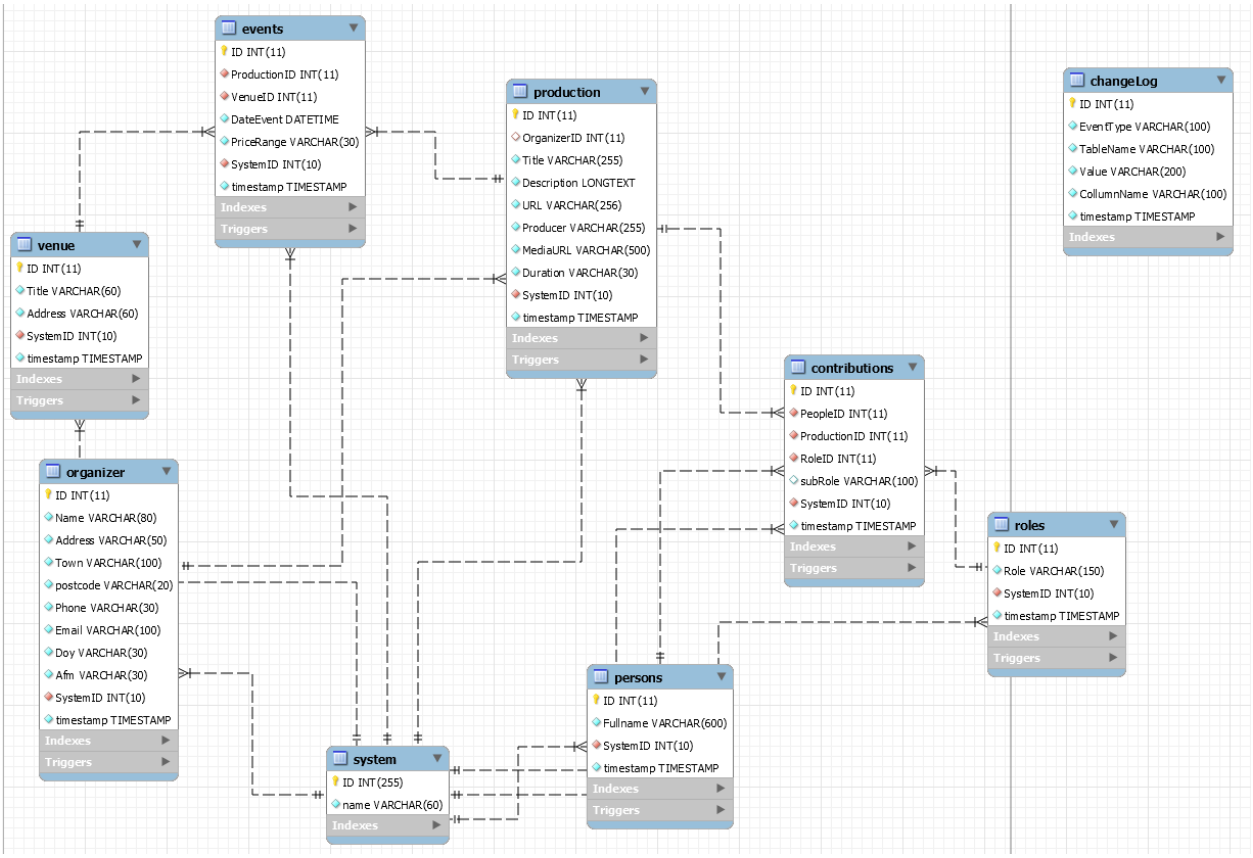
Σχήμα 3.76: Πίνακας changelog

```
DELIMITER $$
CREATE TRIGGER role_insert_trigger AFTER INSERT ON roles
FOR EACH ROW
BEGIN
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('insert', 'roles', NEW.ID , 'ID');
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('insert', 'roles', NEW.Role , 'Role');
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('insert', 'roles', NEW.SystemID , 'SystemID');
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER role_update_trigger AFTER UPDATE ON roles
FOR EACH ROW
BEGIN
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('update', 'roles', CONCAT(OLD.ID, ' >> ', NEW.ID), 'ID');
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('update', 'roles', CONCAT(OLD.Role, ' >> ', NEW.Role), 'Role');
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('update', 'roles', CONCAT(OLD.SystemID, ' >> ', NEW.SystemID), 'SystemID');
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER role_delete_trigger AFTER DELETE ON roles
FOR EACH ROW
BEGIN
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('delete', 'roles', OLD.ID, 'ID');
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('delete', 'roles', OLD.Role, 'Role');
  INSERT INTO changeLog (EventType, TableName, Value, ColumnName)
  VALUES('delete', 'roles', OLD.SystemID, 'SystemID');
END$$
DELIMITER ;
```

Σχήμα 3.87: Απόσπασμα κώδικα SQL Trigger του πίνακα Roles

3.5.10 Σχεσιακή δομή πινάκων

Η σχέσεις των πινάκων:



Σχήμα 3.98: Σχεσιακός πίνακας

3.6 Επίλογος

Με το τέλος αυτού του κεφαλαίου έχουμε μια ιδέα για την δομή της βάσης που χρησιμοποιείτε από το σύστημα scraping στο επόμενο κεφαλαίο θα αναλυθεί το μεγαλύτερο κομμάτι της εργασίας, η υλοποίηση του συστήματος με την γλώσσα python.

Κεφάλαιο 4ο: Σύστημα scraper

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα δούμε πως έγινε η υλοποίηση του συστήματος scraping, ποιες είναι οι βασικές λειτουργίες του προγράμματός, την ροή του και τον βασικό στόχο. Επίσης την διαδρομή που ακολουθεί για την συλλογή της πληροφορίας και τους ελέγχους που κάνει το πρόγραμμα σε κάθε κίνηση.

4.2 Λειτουργίες προγράμματος

Το πρόγραμμα είναι κυρίως back-end φύσεως παρόλο αυτά ο χρήστης έχει δυνατότητες μεσώ γραφικού περιβάλλον. Αποτελείται από τρία βασικά σημεία:

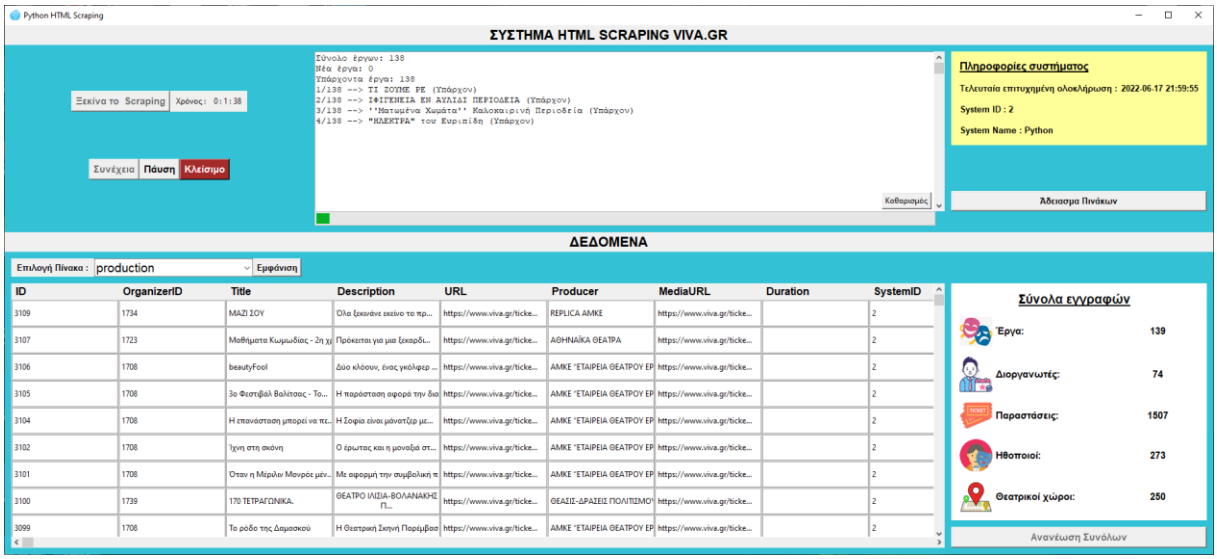
- **Το κομμάτι με την εκκίνηση της διαδικασίας scraping**
- **Την οθόνη οπού εκτυπώνει σημαντικά στοιχεία κατά την εκτέλεση της διαδικασίας**
- **Η επιλογή του πίνακα για εμφάνιση μέσω dataGrids**
- **Ο πίνακας με κάποια σύνολα της βάσης δεδομένων**

Στο πρώτο κομμάτι υπάρχουν τέσσερα κουμπιά: «Ξεκίνα το scraping» οπού εκτελείτε η διαδικασία του scraping, «Συνέχεια» οπού συνεχίζεται η διαδικασία αν έχει γίνει παύση, «Παύση» οπού σταματάει προσωρινά η διαδικασία όταν είναι εφικτό και «Κλείσιμο». Επίσης υπάρχει ένα χρονόμετρο που μετράει τον συνολικό χρόνο που έκανε η διαδικασία.

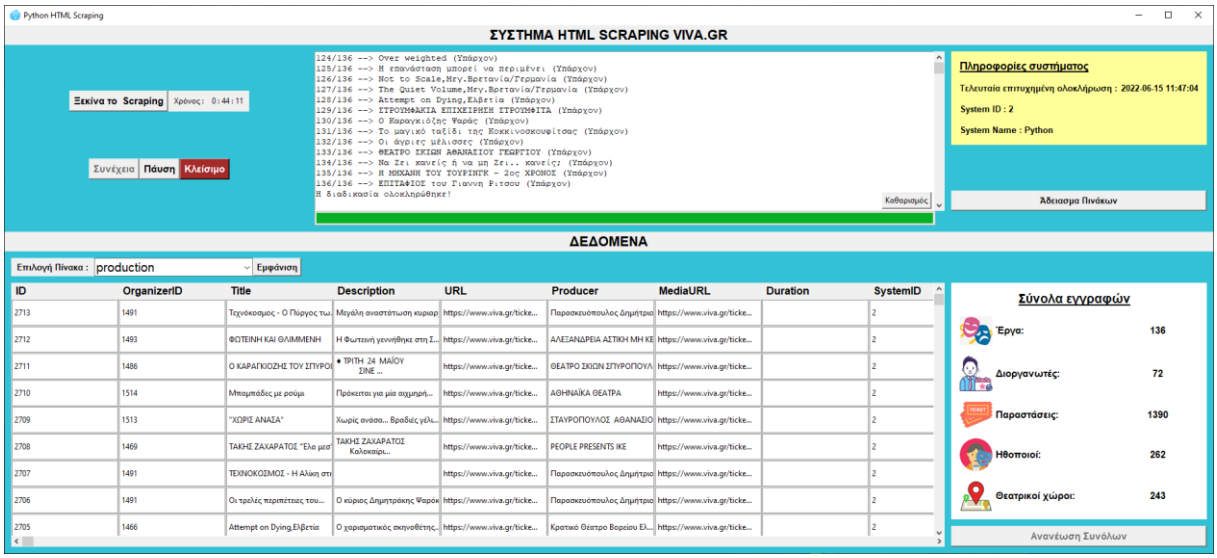
Στο δεύτερο κομμάτι βλέπουμε μια οθόνη η οποία τυπώνει σε νέα γραμμή κάποιές βασικές πληροφορίες όταν ξεκινήσει η διαδικασία και για κάθε έργο ώστε να δίνεται ένα είδος προόδου της διαδικασίας στον χρήστη. Στο τέλος υπάρχει και το κουμπί «Καθαρισμός» οπού καθαρίζει την οθόνη.

Στο τρίτο κομμάτι είναι ο πίνακας σε μορφή dataGrid οπού εμφανίζει τους πίνακες της βάσης με ταξινόμηση ως προς την ημερομηνία αποθήκευσης ώστε να βλέπουμε τις πρόσφατες αλλαγές.

Στο τελευταίο κομμάτι υπάρχει ο πίνακας συνόλων οπού αντλεί κάποια σύνολα από την βάση δεδομένων. Επίσης στο τέλος υπάρχει το κουμπί «Ανανέωση Συνόλων» οπού γίνεται ανανέωση του πίνακα και «Άδειασμα Πινάκων» οπού αδειάζει την βάση δεδομένων.



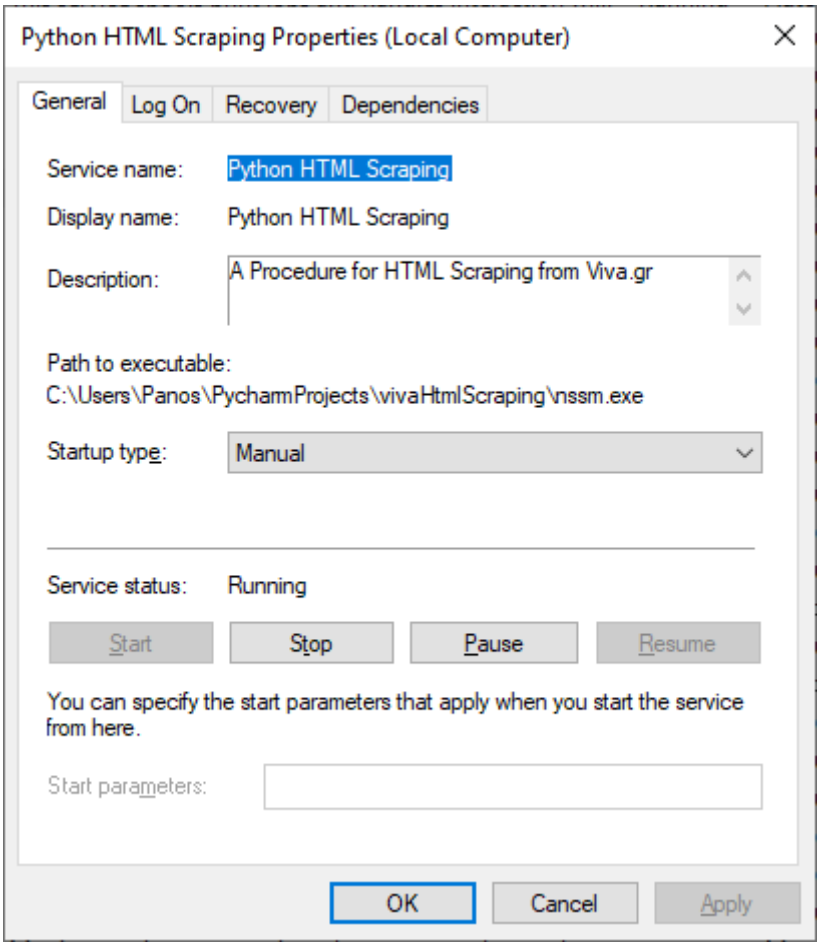
Σχήμα 4.1: Γραφικό περιβάλλον 1



Σχήμα 4.2: Γραφικό περιβάλλον 2

4.3 Αθόρυβη λειτουργία

Το σύστημα scraper διαθέτει επιλογή αθόρυβης λειτουργίας, αυτό σημαίνει η εκτέλεση της κύριας διαδικασίας στο παρασκήνιο χωρίς την επέμβαση του χρήση για την εκκίνηση της διαδικασίας η την εμφάνιση γραφικής διεπαφής κατά την διάρκεια. Η διαδικασία είναι ρυθμισμένη ώστε να γίνεται εκτελείτε κάθε 24ώρο, για παράδειγμα μια φορά κάθε μέρα στις 12:00. Γίνεται έλεγχός από το πρόγραμμα αν έχουν περάσει 24 ώρες από την τελευταία εκτέλεση αν όχι περιμένει μια ώρα και γίνεται ξανά έλεγχος. Με αυτήν την ικανότητα η βάση δεδομένων απομένει ενημερωμένη σε καθημερινή βάση χωρίς κάποια επιπλέον ενέργεια.



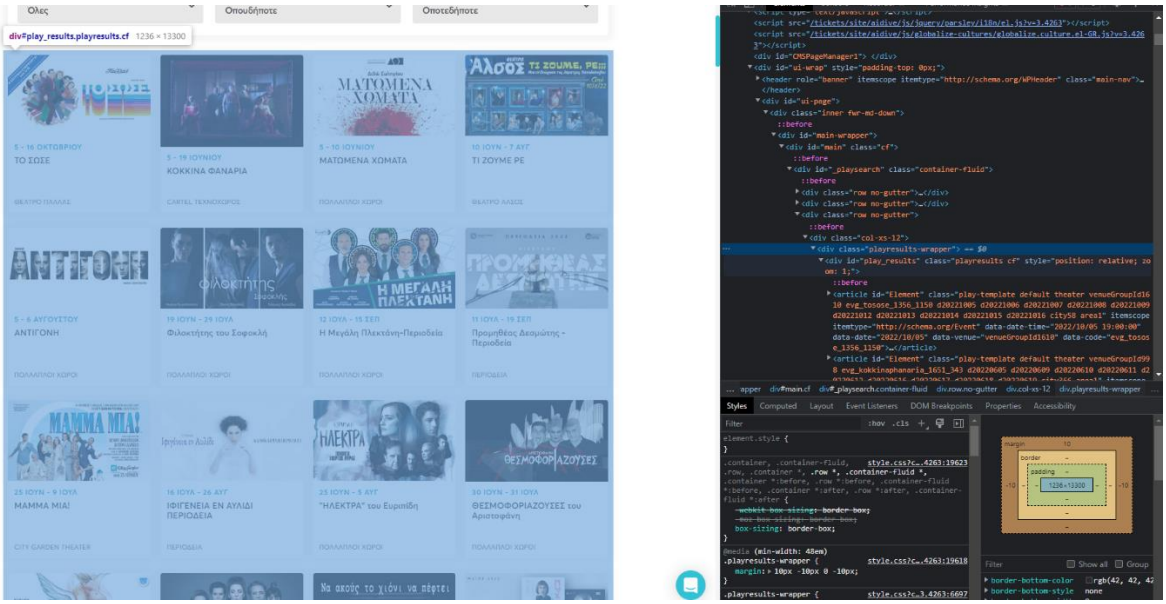
Σχήμα 4.3: Λειτουργία μέσω windows service

4.4 Ροή προγράμματος

Σε αυτήν την ενότητα θα γίνει μια αναλυτική περιγραφή της ροής του προγράμματος, πως γίνεται η περιήγηση στο DOM η συλλογή των δεδομένων και ο έλεγχος σε περίπτωση υπάρχον πληροφορίας.

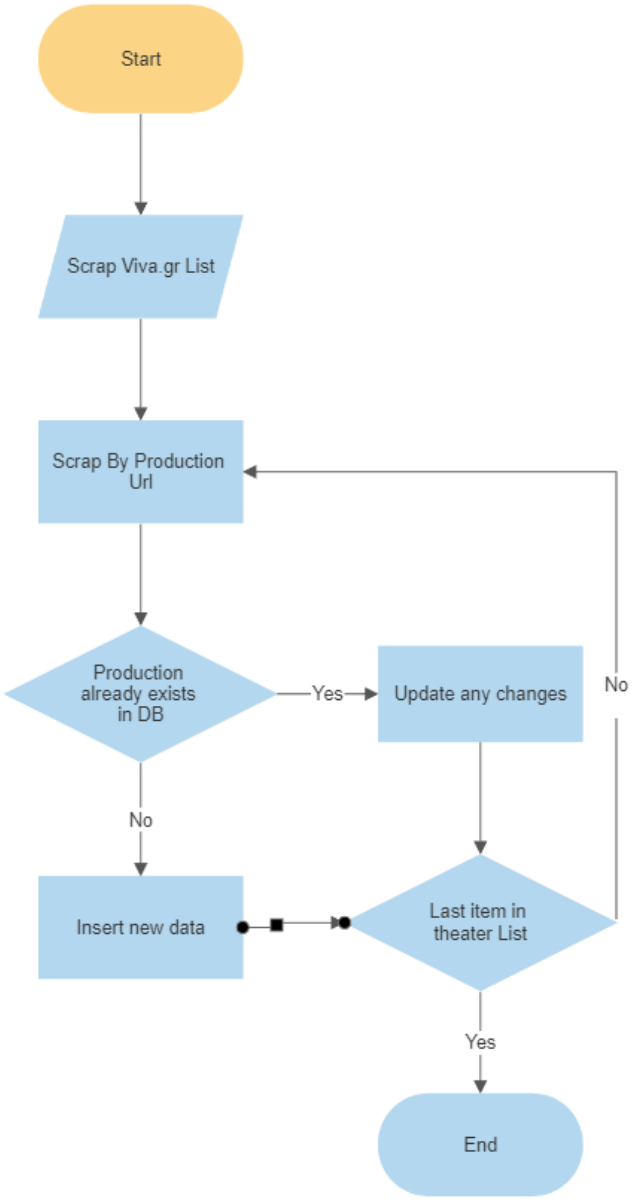
Τα βήματα του προγράμματος είναι τα εξής:

- **Συλλογή λίστας θεατρικών παραστάσεων από το viva.gr/tickets/theatre/**
- **Συλλογή πληροφοριών για κάθε θεατρικό και αποθήκευση στην βάση**
- **Συλλογή πληροφορίας για τον οργανωτή και αποθήκευση στην βάση**
- **Συλλογή πληροφορίας για τον χώρο εκδήλωσης και αποθήκευση στην βάση**
- **Συλλογή πληροφορίας για τις κρατήσεις και αποθήκευση στην βάση**
- **Συλλογή πληροφορίων για τους συντελεστές του έργου και αποθήκευση στην βάση**



Σχήμα 4.4: Λίστα θεατρικών έργων Viva.gr

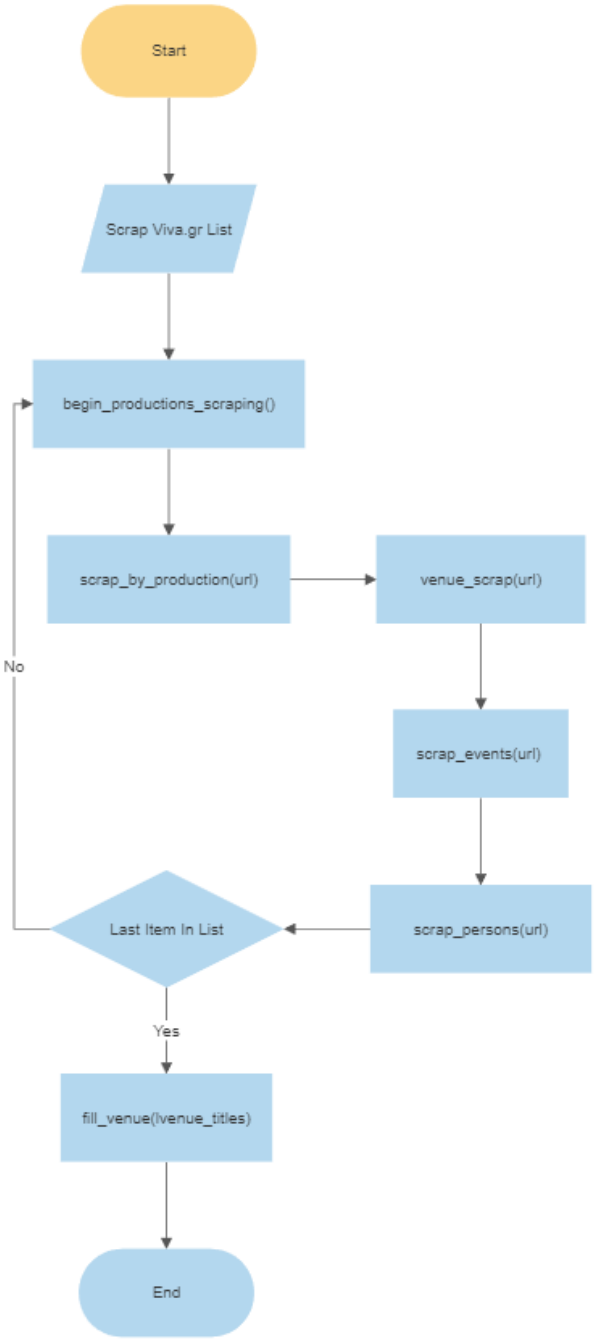
Μεσώ της λίστας δημιουργείται ένας βρόγχος για κάθε θεατρικό έργο, επίσης το σύνολο των έργων τυπώνετε στην οθόνη ξεχωρίζοντας τα νέα με τα υπάρχοντα έργα. Στην συνέχεια ο scraper συλλέγει δεδομένα μέσα από την σελίδα του έργου για τον ίδιο το έργο τον οργανωτή τους συντελεστές τον χώρο εκδήλωσης και τις κρατήσεις. Εάν το έργο υπάρχει ήδη στην βάση αυτό συμβαίνει όταν το url κάνει match σε υπάρχον εγγραφή, τότε η συλλογή δεδομένων πάλι γίνεται σε περίπτωση αλλαγής πληροφορίας. Η διαδικασία αυτή μπορεί να γίνει και με windows service ώστε να γίνεται «κρυφά» στο παρασκήνιο ανά κάποιο χρονικό διάστημα αυτόματα. Μεσώ του selenium χρησιμοποιείτε ένας headless Firefox drive ώστε να μην πετάγονται παράθυρα από τον scraper κατά την διάρκεια. Η διαδικασία κατά μέσο όρο διαρκεί μία με τρεις ώρες, ένας από τους λόγους είναι γιατί τα έργα στην λίστα είναι συνήθως πολλά (150 συνήθως) και ο έλεγχος γίνεται με σειρά ένα προς ένα. Το διάγραμμα ροής βρίσκεται παρακάτω:



Σχήμα 4.5: Γενικό Διάγραμμα Ροής

Αναλυτικότερα οι μέθοδοι του προγράμματος είναι οι εξής:

- **begin_productions_scraping(): Αρχικοποίηση της διαδικασίας**
- **scrap_by_production(url): Συλλογή δεδομένων θεατρικού έργου**
- **venue_scrap(url): Συλλογή δεδομένων του χώρου εκδήλωσης**
- **scrap_events(url): Συλλογή δεδομένων κρατήσεων**
- **scrap_persons(url): Συλλογή δεομένων συντελεστών**
- **fill_venue(lvenue_titles): Αποθήκευση χώρων εκδηλώσεων μαζικά στην βάση**

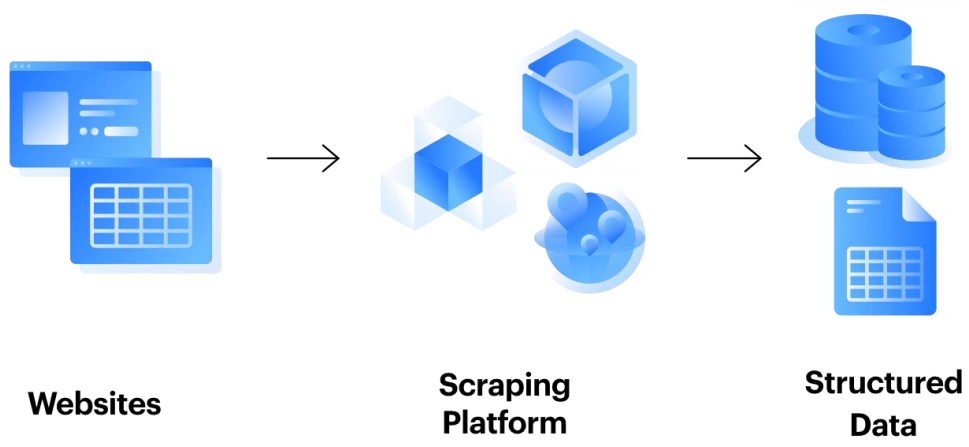


Σχήμα 4.6: Διάγραμμα Ροής μεθόδων

4.5 Στόχος Προγράμματος

Ο στόχος του προγράμματος είναι να συλλέξει στοχευμένα δεδομένα από θεατρικά έργα στο viva.gr και να τα αποθήκευση σε μια καλά δομημένη βάση δεδομένων. Τα δεδομένα αυτά επιλέγονται με βάση κριτηρίων και επεξεργάζονται ανάλογα με κύριο στόχο την βέλτιστη αποθήκευση των δεδομένων ώστε σε επόμενο στάδιο αυτή η βάση να χρησιμοποιηθεί σαν μια πηγή πληροφοριών θεατρικών έργων χωρίς περαιτέρω μετατροπή των τιμών. Επίσης μέσω «κρυφής λειτουργιάς» με την βοήθεια headless browser driver και windows service η εκτέλεση του αλγορίθμου για την συλλογή των πληροφοριών γίνεται πολύ

εύκολα από έναν προσωπικό υπολογιστή και ανά τακτά χρονικά διαστήματα ώστε η πηγή των πληροφορίων να είναι ενημερωμένη.



Σχήμα 4.7: Web scraping

4.6 Επίλογος

Με το τέλος αυτού του κεφαλαίο έχουμε συνοψίσει όλη την υλοποίηση του συστήματος μαζί με τον βασικό στόχο. Στην συνέχεια θα γίνει εξέταση των αποτελεσμάτων για να δούμε αν όντων τα δεδομένα είναι ορθά. Τέλος θα καταλήξω σε κάποια συμπεράσματα και προτάσεις βελτιώσεις.

Κεφάλαιο 5ο: Συμπεράσματα

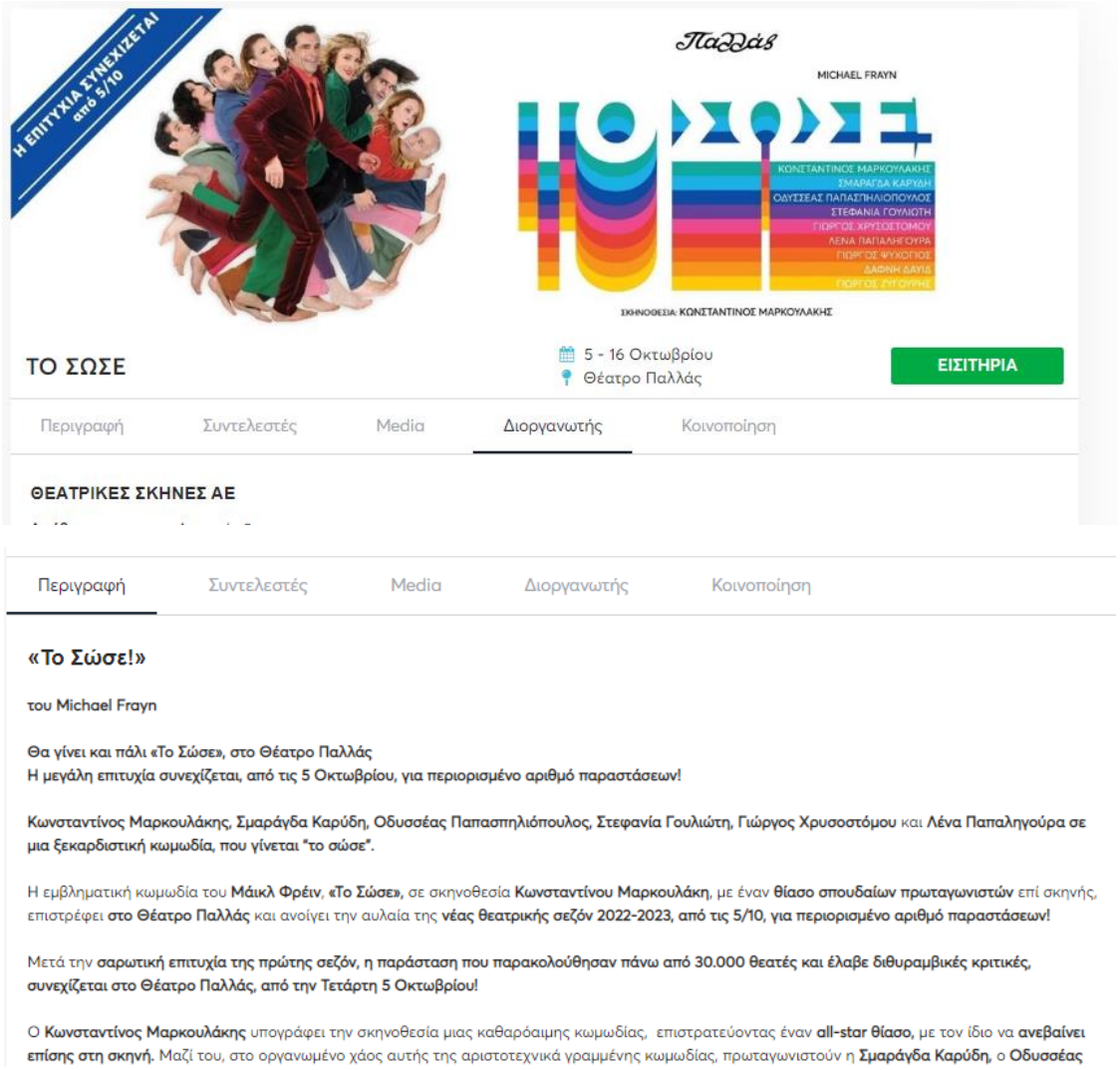
5.1 Έλεγχος αποτελεσμάτων

Στην παρακάτω ενότητα θα γίνει έλεγχος δεδομένων μεσώ αντιστοίχισης πληροφορίας από το viva.gr με τις εκάστοτε εγγραφές στην βάση δεδομένων. Δηλαδή μια σύγκριση αποτελεσμάτων διπλά-δίπλα για να καταλάβουμε αν όντως τα δεδομένα συλλέγονται σωστά.

Το θεατρικό έργο που θα εξεταστή ονομάζεται «ΤΟ ΣΩΣΕ» του Michael Frayn με σύνδεσμο: https://www.viva.gr/tickets/theater/to-swse/

Σημείωση ότι το συγκεκριμένο θεατρικό υπάρχει από 10 Φεβρουάριου και η μέρα που έγινε η σύγκρισή ήταν 5 Ιουνίου. Οπότε πέρα από τα τωρινά δεδομένα θα δούμε δεδομένα στην βάση που δεν υπάρχουν πλέον στο viva.gr αυτό είναι επίσης ένας λόγος χρήσης των scraper ως μια πηγή πληροφορίων που δεν υπάρχει πλέον στο διαδίκτυό λόγω ενημέρωσης της εκάστοτε ιστοσελίδας.

Αρχικά στον πίνακα Production οπού αποθηκεύονται τα θεατρικά έργα:

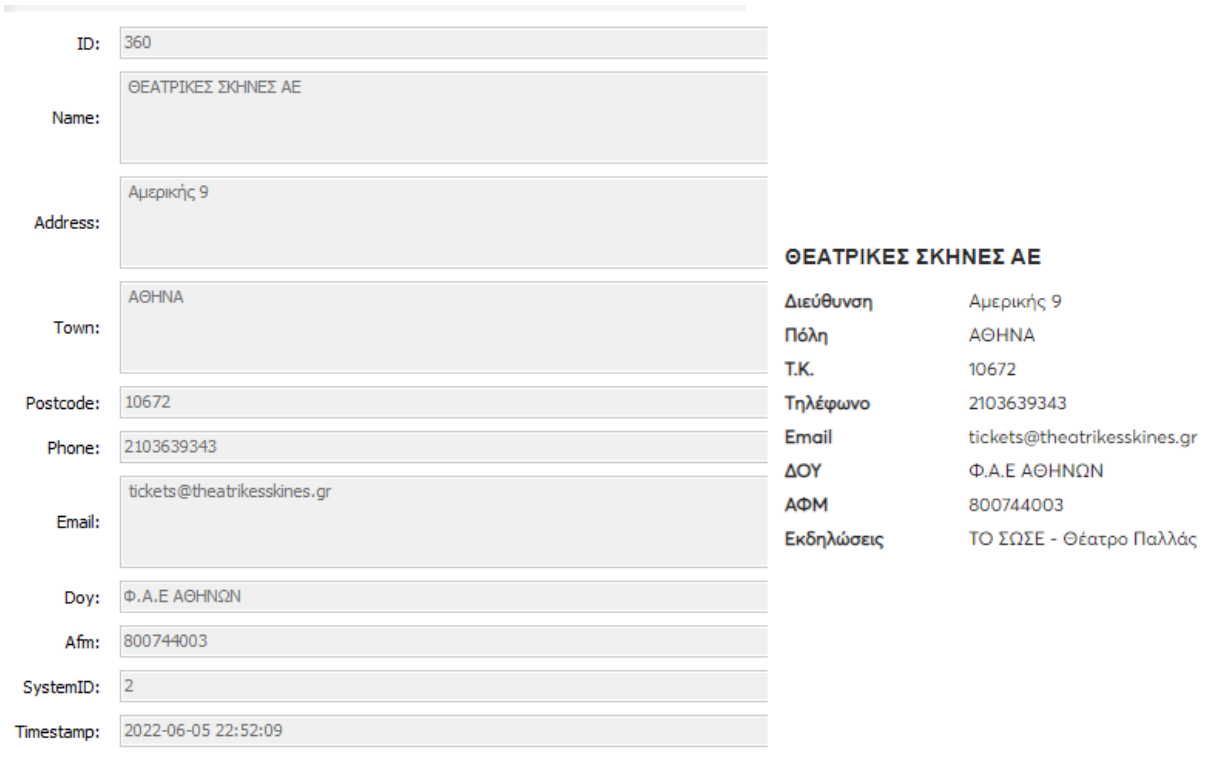


Σχήμα 5.1: Θεατρικό έργο στο viva.gr

Σχήμα 5.2: Ερώτημα στον πίνακα Productions

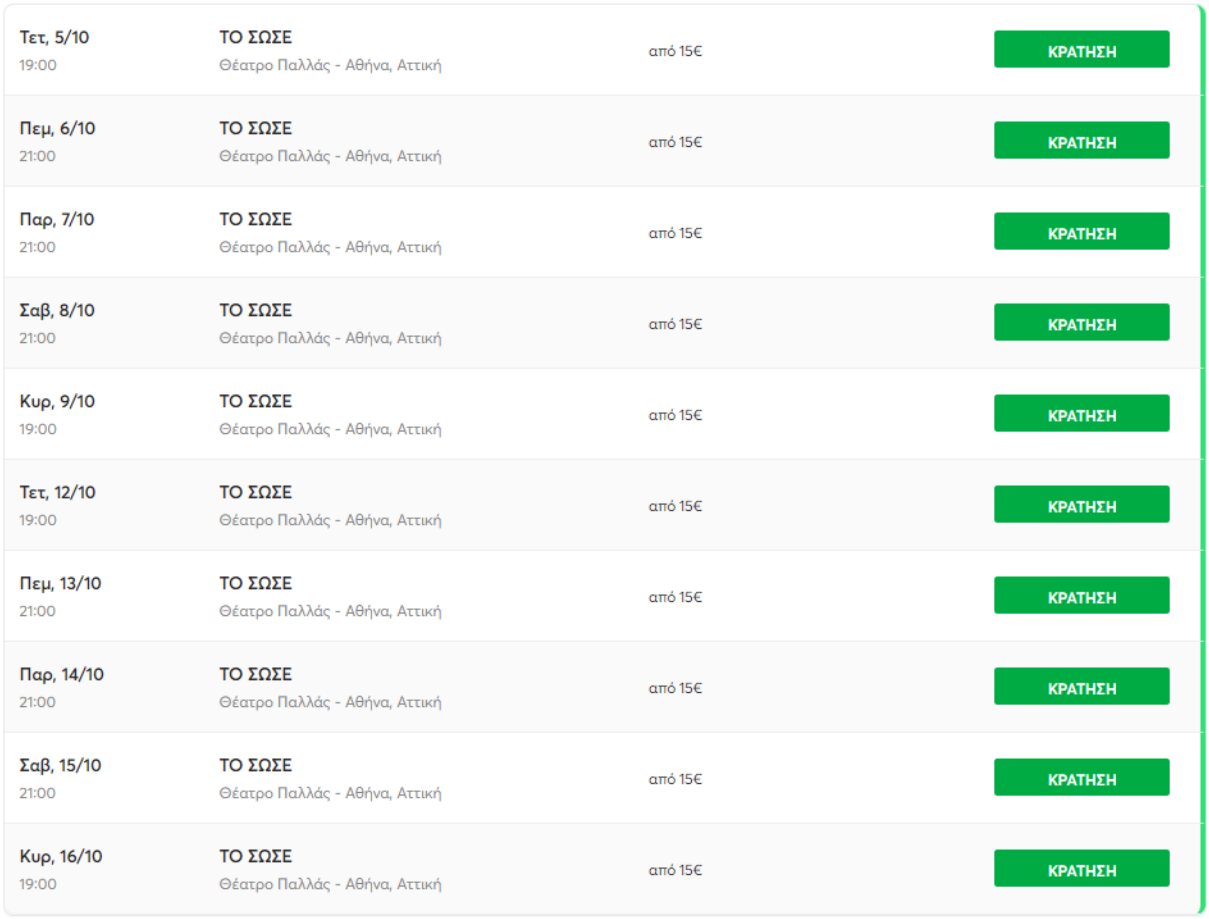
Μπορούμε να δούμε ότι όλα τα πεδία είναι σωστά το OrganizerID είναι 360 και θα γίνει ο έλεγχος του Οργανωτή στην συνέχεια. Το πεδίο duration σωστά είναι κενό διότι δεν υπάρχει η αντίστοιχη πληροφορία στο viva

Ο πίνακας Organizer οπού έχει τους διοργανωτές:



Σχήμα 5.3: Οργανωτής στον πίνακα σε σύγκριση με το viva.gr

Ο διοργανωτής είναι σε πλήρη αντιστοιχία.

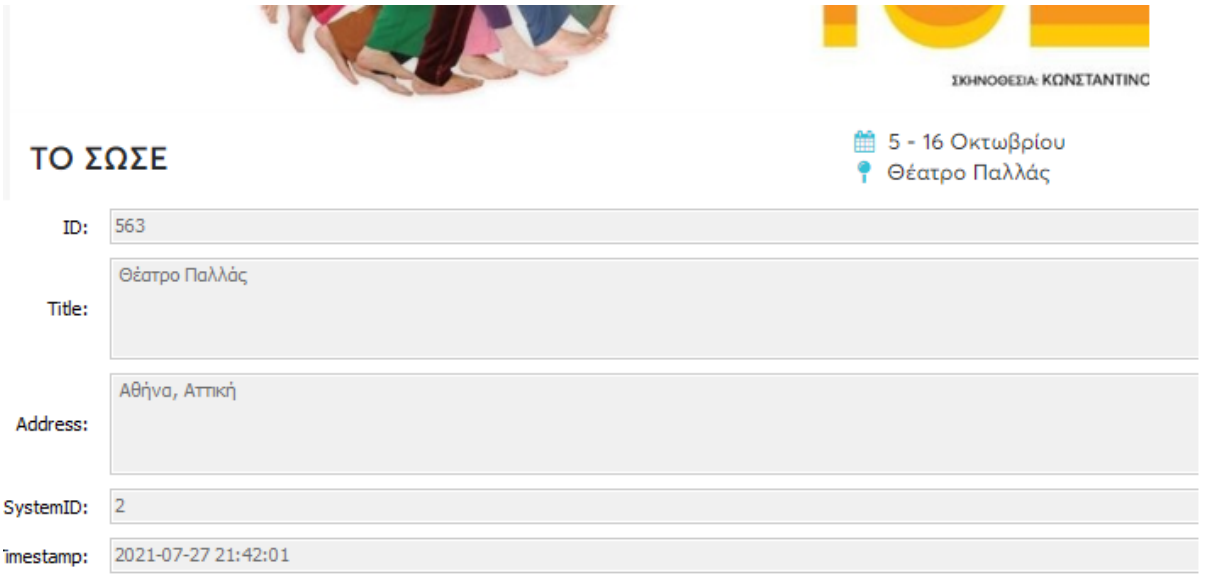


Σχήμα 5.4: Λίστα κρατήσεων στο viva.gr

ID	ProductionID	VenueID	DateEvent	PriceRange	SystemID	timestamp
25582	1151	563	2022-10-16 19:00:00	από 15€	2	2022-05-22 18:54:10
25581	1151	563	2022-10-15 21:00:00	από 15€	2	2022-05-22 18:54:10
25580	1151	563	2022-10-14 21:00:00	από 15€	2	2022-05-22 18:54:09
25579	1151	563	2022-10-13 21:00:00	από 15€	2	2022-05-22 18:54:09
25578	1151	563	2022-10-12 19:00:00	από 15€	2	2022-05-22 18:54:09
25577	1151	563	2022-10-09 19:00:00	από 15€	2	2022-05-22 18:54:08
25576	1151	563	2022-10-08 21:00:00	από 15€	2	2022-05-22 18:54:08
25575	1151	563	2022-10-07 21:00:00	από 15€	2	2022-05-22 18:54:08
25574	1151	563	2022-10-06 21:00:00	από 15€	2	2022-05-22 18:54:07
25573	1151	563	2022-10-05 19:00:00	από 15€	2	2022-05-22 18:54:07
24082	1151	563	2022-05-15 20:00:00	από 15€	2	2022-04-15 21:38:23
24081	1151	563	2022-05-14 21:00:00	από 15€	2	2022-04-15 21:38:23
24080	1151	563	2022-05-13 21:00:00	από 15€	2	2022-04-15 21:38:22
24079	1151	563	2022-05-12 21:00:00	από 15€	2	2022-04-15 21:38:21
24078	1151	563	2022-05-11 20:00:00	από 15€	2	2022-04-15 21:38:21
24077	1151	563	2022-05-08 20:00:00	από 15€	2	2022-04-15 21:38:20
24076	1151	563	2022-05-07 21:00:00	από 15€	2	2022-04-15 21:38:20
24075	1151	563	2022-05-06 21:00:00	από 15€	2	2022-04-15 21:38:19
24074	1151	563	2022-05-05 21:00:00	από 15€	2	2022-04-15 21:38:19
24073	1151	563	2022-05-04 20:00:00	από 15€	2	2022-04-15 21:38:18
24072	1151	563	2022-05-01 20:00:00	από 15€	2	2022-04-15 21:38:18
24071	1151	563	2022-04-30 21:00:00	από 15€	2	2022-04-15 21:38:17
24070	1151	563	2022-04-29 21:00:00	από 15€	2	2022-04-15 21:38:17
24069	1151	563	2022-04-28 21:00:00	από 15€	2	2022-04-15 21:38:16
24068	1151	563	2022-04-27 20:00:00	από 15€	2	2022-04-15 21:38:16
21292	1151	563	2022-04-17 18:30:00	από 15€	2	2022-02-19 12:57:43
21291	1151	563	2022-04-16 21:00:00	από 15€	2	2022-02-19 12:57:42
21290	1151	563	2022-04-15 21:00:00	από 15€	2	2022-02-19 12:57:42
21289	1151	563	2022-04-14 21:00:00	από 15€	2	2022-02-19 12:57:41
21288	1151	563	2022-04-13 19:00:00	από 15€	2	2022-02-19 12:57:41
21287	1151	563	2022-04-10 18:30:00	από 15€	2	2022-02-19 12:57:40
21286	1151	563	2022-04-09 21:00:00	από 15€	2	2022-02-19 12:57:40
21285	1151	563	2022-04-08 21:00:00	από 15€	2	2022-02-19 12:57:39
21284	1151	563	2022-04-07 21:00:00	από 15€	2	2022-02-19 12:57:39
21283	1151	563	2022-04-06 19:00:00	από 15€	2	2022-02-19 12:57:38
21282	1151	563	2022-04-03 18:30:00	από 15€	2	2022-02-19 12:57:38
21281	1151	563	2022-04-02 21:00:00	από 15€	2	2022-02-19 12:57:37

Σχήμα 5.5: Ερώτημα στον πίνακα Events

Στον πίνακα Events παρατηρούμε ότι τα δεδομένα είναι ορθά. Επίσης περιέχει παλαιότερες εκδηλώσεις οι οποίες δεν υπάρχουν πλέον στο viva γιατί η ημέρα του συμβάντος έχει περάσει αλλά το σύστημα το έχει συλλέξει σε προηγούμενο χρόνο.



Σχήμα 5.6: Χώρος εκδήλωσης στον πίνακα σε σύγκριση με το viva.gr

Στον πίνακα venue τα δεδομένα είναι σωστά.

Στην συνέχεια θα γίνει σύγκριση των ανθρώπινων συντελεστών, για να δούμε τα δεδομένα στην βάση πρέπει να γίνει προβολή τριών πινάκων Contributions, Roles, Persons.

Σχήμα 5.7: Συντελεστές στο viva.gr

ID	PeopleID	ProductionID	RoleID	subRole	SystemID	timestamp
13272	6104	1151	2771		2	2022-02-19 12:57:45
13273	6106	1151	1691		2	2022-02-19 12:57:45
13274	6107	1151	1707		2	2022-02-19 12:57:46
13275	9365	1151	1693		2	2022-02-19 12:57:47
13276	4802	1151	1692		2	2022-02-19 12:57:47
13277	6109	1151	1773		2	2022-02-19 12:57:48
13278	6110	1151	1694		2	2022-02-19 12:57:49
13279	5291	1151	2772		2	2022-02-19 12:57:49
13280	6111	1151	2772		2	2022-02-19 12:57:50
13281	7545	1151	1734		2	2022-02-19 12:57:51
13282	6115	1151	1736		2	2022-02-19 12:57:51
13283	9366	1151	2051		2	2022-02-19 12:57:52
13284	9367	1151	2052		2	2022-02-19 12:57:53
13285	8145	1151	2052		2	2022-02-19 12:57:53

ID	Role	SystemID	timestamp
1691	Σκηνικά	2	2021-01-31 12:32:58
1692	Κοστούμια	2	2021-01-31 12:32:58
1693	Φωτισμοί	2	2021-01-31 12:32:59
1694	Βοηθός σκηνοθέτη	2	2021-01-31 12:33:00
1695	Ηθοποιός	2	2021-01-31 12:33:01
1707	Μουσική	2	2021-01-31 12:33:42
1734	Βοηθός ενδυματολόγου	2	2021-01-31 12:39:10
1736	Φωτογραφίες	2	2021-01-31 12:39:12
1773	Κίνηση	2	2021-01-31 12:42:03
2051	Graphic design	2	2021-06-15 21:16:30
2052	Social Media	2	2021-06-15 21:16:31
2771	Σκηνοθεσία - Μετάφραση	2	2022-02-19 12:57:44
2772	Βοηθοί σκηνογράφου	2	2022-02-19 12:57:49

ID	Fullname	SystemID	timestamp
4802	Βασιλική Σύρμα	2	2021-01-31 12:40:33
5291	Αλέγια Παπαγεωργίου	2	2021-03-05 21:26:24
6104	Κωνσταντίνος Μαρκουλάκης	2	2021-07-27 21:42:13
6106	Πάρις Μέξης	2	2021-07-27 21:42:16
6107	Μίνως Μάτσας	2	2021-07-27 21:42:17
6109	Κική Μπάκα	2	2021-07-27 21:42:19
6110	Χριστίνα Ματθαίου	2	2021-07-27 21:42:20
6111	Εύη Ανδριανού	2	2021-07-27 21:42:21
6115	Ελίνα Γιουνανλή	2	2021-07-27 21:42:25
7545	Κατερίνα Αριανούτσου	2	2021-10-09 18:12:51
8145	Κάλλη Μαυρογένη	2	2021-11-26 19:40:39
9365	Δημήτρης Κουτάς	2	2022-02-19 12:57:46
9366	Indigo Creative	2	2022-02-19 12:57:51
9367	ad4art	2	2022-02-19 12:57:52

Σχήμα 5.8: Πίνακας Contribution, Roles, Persons

Αρχικά ο πίνακας Contributions είναι υπεύθυνος για την σύνδεση του Ατόμου με το έργο και την ειδικότητα, για να καταλάβουμε τις ειδικότητες και τα ονόματα των ατόμων πρέπει να ψάξουμε τα αντίστοιχα id στους πίνακες Roles και Persons. Το πεδίο subRole του πίνακα Contributions είναι κενό διότι ο scraper δεν βρήκε κάποιον ρόλο στην περιγραφή του έργου ώστε να κάνει την αντιστοίχιση.

5.2 Αποτελέσματα και συμπεράσματα

Μετά την εξέτασή μπορούμε να πούμε ότι τα αποτελέσματα της βάσης είχαν μια καλή αντιστοιχία με το viva.gr βέβαιο κάθε θεατρικό έχει διαφορετικά δεδομένα. Ένα έργο μπορεί να έχει ελλιπή δεδομένα, για παράδειγμα να μην υπάρχει η καρτέλα του Διοργανωτή η των Συντελεστών. Αυτό έχει ως αποτέλεσμα κενών κελιών στην βάση η απαλλαγή κάποιων οντοτήτων για την εκάστοτε εγγραφή. Ο δομή του DOM στο viva είναι κυρίως σταθερή και δεν υπάρχει αστοχία στην συλλογή των δεδομένων εκτός της ενότητας των συντελεστών. Οι συντελεστές είναι ένα ενιαίο πεδίο κειμένου οπού περιγράφονται οι συντελεστές του έργου χωρίς κάποια δομή. Ο scraper εξετάζει αυτό το πεδίο γραμμή προς γραμμή και παραλείπει ότι δεν βγάζει νόημα ή έχει περίπλοκη δομή ώστε να συλλεχθεί ως πληροφορία.

Με τον καιρό διαπιστώνεται ότι η βάση δεδομένων έχει συλλέξει αρκετά δεδομένα θεατρικών έργων τα οποία τα περισσότερα δεν υπάρχουν στο viva λόγω παλαιότητας. Το σύνολο των θεατρικών έργων είναι στα 1045 με 302 διοργανωτές 4377 ηθοποιούς 1524 θεατρικούς χώρους και 18181 παραστάσεις. Όλα αυτά τα δεδομένα μπορούν να χρησιμοποιηθούν για την εξαγωγή κάποιων στατιστικών στοιχείων.

Γενικότερα το web scraping μπορεί να βοηθήσει στην συλλογή στοχευμένης πληροφορίας μέσα από την πληθώρα που προσφέρει το διαδίκτυο με κάποιον απώτερο σκοπό. Επίσης η αυτοματοποίηση μιας τέτοιας διαδικασίας μπορεί να σώσει πολλές ώρες δουλείας και όταν ο μόνος τρόπος συλλογής πληροφορίας είναι μόνο μέσω του διαδικτύου μεσώ http client και δεν υπάρχει η επιλογή κάποιου API τότε το scraping είναι μονόδρομος.

5.3 Προτάσεις και βελτίωση

Το πρώτο σημείο προς βελτίωση είναι η επέκταση του προγράμματος ώστε να συλλέγει πληροφορίες είτε από περισσότερες ιστοσελίδες είτε από μηχανές αναζήτησης.

Ο χρόνος εκτέλεσης της διαδικασίας scraping ως την ολοκληρώσει διαρκεί περίπου δύο ώρες, αυτό μπορεί να βελτιωθεί με χρήση παραλλήλων ροών στον αλγόριθμο για παράδειγμα χρήση πάνω από έναν http client στην συλλογή των δεδομένων.

Μια βελτίωση ως την πρακτικότητα είναι η υλοποίηση της διαδικασίας σε περιβάλλον server. Η φύση του προγράμματος δεν ταιριάζει σε έναν προσωπικό υπολογιστή windows, αντίθετος ένας server μπορεί διαθέσει περισσότερη υπολογιστική δύναμη και χρονική διαθεσιμότητα επειδή συνήθως είναι ενεργός όλο το εικοσιτετράωρο. Επίσης μπορεί να προγραμματιστεί η εκτέλεση της διαδικασίας μέσω cron jobs για την τακτική ενημέρωση των δεδομένων.

Το πρόγραμμα διαχειρίζεται την βάση δεδομένων απευθείας μεσώ εντολών SQL. Εδώ δημιουργείτε το πρόβλημα άμεσης επικοινωνίας με την βάση. Σε περίπτωση αποσύνδεσης επικοινωνίας με την βάση το πρόγραμμα θα τερματίσει με σφάλμα. Η χρήση συνόλου δεδομένων(dataset) είναι η λύση σε αυτό το πρόβλημα. Ο σκοπός ενός συνόλου δεδομένων είναι να λειτουργεί ως ένα τοπικό αντίγραφο των δεδομένων, ώστε να μην χρειάζεται η συνεχής επικοινωνία με την βάση δεδομένων.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Βιβλία

[1] Gábor László Hajba, *Website scraping with Python: using BeautifulSoup and Scrapy*. New York: Apress, 2018.

[2] Ryan Mitchell, *Web Scraping with Python*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA, 2015.

[3] R. G. Lawson, *Web scraping with Python : scrape data from any website with the power of Python*. Birmingham Mumbai Packt Publishing, 2015.

[4] B. Lubanovic, *Introducing Python : modern computing in simple packages*. Sebastopol, Ca: O'reilly Media, Inc, 2020.

[5] Giancarlo Zaccone, *Python parallel programming cookbook : master efficient parallel programming to build powerful applications using Python*. Birmingham: Packt Publishing, August, 2015.

[6] J. Melton and A. R. Simon, *Understanding the new SQL : a complete guide*. San Mateo, Calif.: Morgan Kaufmann Publishers, 1993.

[7] P. N. Weinberg, J. R. Groff, and A. J. Oppel, *SQL : the complete reference*. New York, Ny: Mcgraw-Hill, 2010.

Papers

[1] A. Sen, P. A. Dacin, and C. Pattichis, "Current trends in web data analysis," *Communications of the ACM*, vol. 49, no. 11, pp. 85–91, Nov. 2006, doi: 10.1145/1167838.1167842.

[2] D. Glez-Peña, A. Lourenço, H. López-Fernández, M. Reboiro-Jato, and F. Fdez-Riverola, "Web scraping technologies in an API world," *Briefings in Bioinformatics*, vol. 15, no. 5, pp. 788–797, Apr. 2013, doi: 10.1093/bib/bbt026.

[3] C. Zheng, G. He, and Z. Peng, "A Study of Web Information Extraction Technology Based on Beautiful Soup," *Journal of Computers*, vol. 10, no. 6, pp. 381–387, Nov. 2015, doi: 10.17706/jcp.10.6.381-387.

[4] V. Krotov, L. Johnson, and L. Silva, "Tutorial: Legality and Ethics of Web Scraping," *Communications of the Association for Information Systems*, vol. 47, 2020, doi: 10.17705/1CAIS.04724.

[5] L. G. Michael, J. Donohue, J. C. Davis, D. Lee and F. Servant, "Regexes are Hard: Decision-Making, Difficulties, and Risks in Programming Regular Expressions," *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2019, pp. 415-426, doi: 10.1109/ASE.2019.00047.

[6] S. Nyamathulla, Dr. P. Ratnababu, Nazma Sultana Shaik, Bhagya Lakshmi. N, “A Review on Selenium Web Driver with Python,” *Annals of R.S.C.B.*, ISSN:1583-6258, Vol. 25, Issue 4, 2021, Pages. 16760-16768

[7] Anand V. Saurkar, Kedar G. Pathare, Shweta A. Gode, “An Overview On Web Scraping Techniques And Tools,” *International Journal on Future Revolution in Computer Science & Communication Engineering*, ISSN: 2454-4248, Vol. 4, Issue 4, April 2018, Pages. 363–367

Internet Sites

[1] ”Beautiful Soup Documentation” [Online]. Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[2] Document Object Model (DOM) Level 1 Specification (Second Edition) Version 1.0 W3C Working Draft 29 September, 2000 [Online]. Available: https://www.w3.org/TR/2000/WD-DOM-Level-1-20000929/DOM.pdf

[3] Fredrik Lundh, “An Introduction to Tkinter”, 1999. [Online]. Available: http://www.tcltk.co.kr/files/TclTk_Introduction_To_Tkiner.pdf

[4] Real Python “An Intro to Threading in Python”, [Online]. Available: https://realpython.com/intro-to-python-threading/

[5] Real Python “Pyhon GUI Programming with tkinter”, [Online]. Available: https://realpython.com/python-gui-tkinter/

[6] Tech Target “Exception handling”, [Online]. Available: https://www.techtarget.com/searchsoftwarequality/definition/error-handling

[7] Microsoft Docs “Introduction to Windows Service Applications”, [Online]. Available: https://docs.microsoft.com/en-us/dotnet/framework/windows-services/introduction-to-windows-service-applications

[8] The python corner “Create a Windows Service in Python” [Online]. Available: https://thepythoncorner.com/posts/2018-08-01-how-to-create-a-windows-service-in-python/

[9] Geeksforgeeks “Difference between Client /Server and Distributed DBMS”, [Online]. Available: https://www.geeksforgeeks.org/difference-between-client-server-and-distributed-dbms/

[10] C-sharpcorner “Triggers in SQL Server”, [Online]. Available: https://www.c-sharpcorner.com/UploadFile/63f5c2/triggers-in-sql-server/

[11] Oracle “What is a Relational Database (RDBMS)?”, [Online]. Available: https://www.oracle.com/database/what-is-a-relational-database/