Universitatea
Ştefan cel Mare
Suceava

Facultatea de Inginerie
Electrică și Știința
Calculatoarelor

ing. **Elisabeta BUCUR (Zagan)**

TEZĂ de DOCTORAT

– rezumat –

Contribuții privind procesele de ingestie, stocare și transformare a datelor în arhitecturi Data Lake

Conducător științific:

Conf. univ. dr. ing. **Mirela DANUBIANU**

Suceava, 2023

Această lucrare a beneficiat de suport financiar prin proiectul

Excelență academică și valori antreprenoriale - sistem de burse pentru asigurarea oportunităților de formare și dezvoltare a competențelor antreprenoriale ale doctoranzilor și postdoctoranzilor

ANTREPRENORDOC

***Cod Proiect*: SMIS 123847**
***Cod Contract*: POCU/380/6/13**
***Axa prioritară* 6** - Educație și competențe
Prioritatea de investiții – Îmbunătățirea utilității sistemelor de educație și formare pentru piața muncii, facilitarea trecerii de la educație la muncă și consolidarea sistemelor de educație și formare profesională și a calității lor, inclusiv prin mecanisme pentru anticiparea competențelor, adaptarea programelor de învățământ și crearea și dezvoltarea de sisteme de învățare bazate pe muncă, inclusiv sisteme de învățare duale și de ucenicie.

Cuprins

1 Introducere

În ultimii ani este folosit tot mai des conceptual de Big Data datorită creșterii explozive a volumului de date, de natura diversă, colectate pentru a fi stocate și prelucrate, ca ulterior să fie interpretate cu scopul de a susține deciziile. La nivel mondial a fost înregistrată o creștere exponențială a volumului de date provenite din activități de natură variată: social media, comerciale, industriale, sanitare, școlare, etc., astfel încât a devenit o adevărată provocare gestionarea cu succes a acestora.

Paradigma Internet of Things (IoT) constituie un element de transformare revoluționară a interacțiunii noastre cu mediul înconjurător. Aceasta implică dezvoltarea de aplicații inovative în diferite domenii cum ar fi cel al caselor inteligente, orașelor inteligente, supravegherea digitală a sănătății, proceselor industriale, etc. Abundența datelor generate de aceste aplicații permit monitorizarea și analiza corespunzătoare a domeniilor vizate [1]. Un bun indicator al performanțelor unui sistem sunt fișierele log provenite de la rețea, servere, dispozitive de securitate, servicii, aplicații etc. Analiza pe scară largă a acestor log-uri aduce un plus de informații. În [2] autorii oferă o analiză sistematică a literaturii recente referitoare la diferitele tipuri de fișiere log care sunt utilizate în cercetare. Importanța fișierelor log a fost dovedită în practică, astfel încât identificarea unui mediu performant de stocare și analiză a acestora este imperios necesară în zilele noastre.

În prezent se urmărește să se extragă cât mai multe avantaje de pe urma analizei datelor. În ultima perioadă, se caută tehnici cât mai performante și eficiente pentru a obține rapid și ușor informații necesare studiilor, pentru noi strategii de dezvoltare și pentru a obține un profit maxim pe baza cunoștințelor obținute din toate datele disponibile, care de multe ori, nu mai pot fi gestionate și prelucrate cu tehnologii, tehnici și metode tradiționale. Necesitatea de a analiza și a explora conținutul datelor colectate a condus la apariția conceptului de Analytics, care a reprezentat un răspuns eficient la nevoile societății în plină evoluție. În societatea informațională este evident că datele sunt un factor esențial de succes, astfel încât analiza acestora poate conduce la informații anterior necunoscute, dar valoroase pentru decizii de bună calitate.

În jurul anilor 1980, a apărut conceptul Data Warehouse (DW) care a avut o contribuție importantă prin crearea unei surse optimizate de analiză a datelor dintr-o organizație, din perspective multiple. Deși la momentul apariției lor au constituit un salt înainte, depozitele de date sunt limitate în ceea ce privește tipurile de date ce pot fi stocate și prelucrate (se folosesc tipuri de date structurate) și de cele mai multe ori au costuri foarte mari.

Apariția unor noi tipuri de date, nestructurate sau semi-structurate, provenite din mediile web, social media, comentarii, servere, senzori și diferite dispozitive au făcut ca aceste depozite de date să fie depășite. Acum 15 - 20 de ani nu era previzibil ca într-un viitor apropiat să fie atât de importantă evidența “like-urilor” de pe diferitele medii sociale, deși acestea pot furniza informații vitale, ele fiind un feedback direct al utilizatorilor din mediul virtual. Serverele web sunt accesate de milioane de utilizatori în fiecare zi. Astfel, utilizatorii își lasă în urmă așa numitul “comportament de vizitare” prin înregistrarea activităților lor în mediul on-line.

Serverele păstrează aceste înregistrări în diferite fișiere, cum ar fi **access logs**, error logs, piped logs, script logs, etc. Prin analiza fișierelor **web server access logs (WSAL)** înregistrate pe servere este posibilă descoperirea "comportamentelor de vizitare" [3], ce poate conduce la îmbunătățirea și personalizarea serviciilor oferite utilizatorilor, așa cum vom vedea pe parcursul acestei teze de doctorat. Analiza fișierelor WSAL are o mare importanță în aplicațiile practice [4].

Într-un Data Warehouse, aceste tipuri de date vor fi ignorate, neputând fi stocate în formă structurată. Ca și consecință, în ultimii ani, explozia acestor tipuri de date noi a determinat apariția și dezvoltarea de noi concepte, tehnologii și tehnici de gestionare a datelor. **Data Lake (DL)** este un concept revoluționar aflat în centrul atenției în ultima perioadă. DL, după cum o spune și numele, poate fi văzut ca o acumulare naturală de date, o structură unde pot fi stocate toate datele în forma lor brută [5].

1.1 Obiectivele și structura tezei

Cercetările din această teză de doctorat se concentrează pe oferirea de soluții economice, sigure și de înaltă performanță pentru stocarea de fișiere web server access log brute, pe perioade îndelungate de timp. Sistemul propus implementează o arhitectură Data Lake în cloud folosind Azure Data Lake Storage Gen2 (ADLS Gen2) pentru pipeline-ul Extract–Load–Transform (ELT) specific DL. Arhitectura propusă permite stocarea unor volume mari de date în forma lor brută. Ulterior, acestea sunt supuse unor procese de transformare și de analiză fără a fi nevoie de o schemă de scriere structurată, așa cum se întâmplă în cazul depozitelor de date.

Obiectivul general al tezei de doctorat vizează efectuarea de cercetări privind ingestia, stocarea și transformarea datelor într-o arhitectura DL. Pentru atingerea acestui obiectiv mi-am propus următoarele **obiective specifice**:

OS1. Analiza stadiului actual al cercetărilor privind arhitecturile DL;

OS2. Identificarea diferitelor metodologii de implementare a DL;

OS3. Identificarea și evidențierea importanței stocării datelor de tip WSAL pe termen lung în arhitecturi DL;

OS4. Proiectarea și implementarea unei arhitecturi DL pentru stocarea de fișiere WSAL;

OS5. Dezvoltarea proceselor de ingestie și stocare a datelor de tip WSAL în DL.

OS6. Proiectarea unui model de structurare ierarhică a datelor pentru stocarea eficientă a acestora în DL.

OS7. Implementarea procesului de transformare a datelor brute semi-structurate în date structurate și pregătirea lor pentru aplicarea de tehnici de analiză avansată.

Principala contribuție a acestei teze de doctorat este de a oferi o soluție economică și accesibilă pentru a realiza ingerarea, stocarea și transformarea fișierelor WSAL prin intermediul celei mai noi tehnologii, și anume Data Lake. Ca și contribuții derivative, am propus utilizarea de comenzi cross-platform pentru ingestia datelor în DL, folosind Azure Command-Line Interface (CLI). Ulterior, am implementat o funcție Azure Blob Trigger pentru a realiza algoritmul de transformare a fișierelor WSAL în fișiere de tip parquet. În urma procesului de transformare s-a obținut o reducere cu 90% a spațiului de stocare, ceea ce implică o reducere semnificativă a costurilor de stocare. De asemenea, am propus un model ierarhic de stocare a datelor în DL pentru accesul partajat la date pe diferite niveluri din cadrul arhitecturii

DL, peste care am desemnat aplicarea de reguli de gestionare a ciclului de viață al datelor (Data Lifecycle Management - DLM) pentru eficientizarea costurilor de stocare. De asemenea am propus ingerarea fișierelor WSAL într-un DL implementat în cloud datorită ușurinței de implementare și a costurilor reduse de stocare. Scopul este de a menține aceste date pe termen lung, pentru a fi utilizate în viitoarele procese de analiză avansată prin încrucișarea cu alte date organizaționale sau externe. Deși soluția propusă se bazează în mod explicit pe ADLS Gen2, aceasta reprezintă un punct de referință important în implementarea unui DL.

Teza de doctorat este structurată pe opt capitole la care se adaugă lista de referințe bibliografice și anexe.

În **capitolul 1**, am prezentat o descriere generală a lucrării în care am punctat obiectivele propuse, contribuțiile și modul de diseminare al rezultatelor.

Pe baza cercetărilor bibliografice, în **capitolul 2**, am realizat un studiu ce prezintă stadiul actual al cercetărilor în domeniul DL. După o scurtă descriere a principalelor caracteristici ale unui DL, am realizat un studiu comparativ între cele mai performante medii de stocare a datelor și anume Data Warehouse și Data Lake. Am evidențiat diferențele dintre cele două, dar și noutățile cu care se remarcă DL și de ce este acesta o inovație și o necesitate în era digitală. Tot aici am prezentat diferențele dintre pipeline-urile ETL și ELT ce definesc cele două medii de stocare și am evidențiat avantajele majore ce aduc flexibilitate, scalabilitate și performanță în stocarea datelor. În contextul Big Data, DL vine cu o serie de avantaje față de metodele de stocare existente care nu mai fac față volumelor mari de date, vitezei cu care sunt generate și a diversității acestora. Spre finalul acestui capitol am prezentat o serie de implementări practice, întâlnite în cercetările bibliografice efectuate, care demonstrează eficiența acestui nou concept în era Big Data. Astfel, sunt prezentate diferențele dintre cele mai semnificative medii de stocare a datelor, provocările cu care se confruntă acestea și totodată sunt propuse diferite mecanisme pentru îmbunătățirea și gestionarea volumelor mari de date din diferite domenii de aplicabilitate.

O arhitectură DL, în funcție de mediul de stocare, poate fi implementată în patru moduri diferite și anume *On-Premise*, în *Cloud*, *Multi–Cloud* sau *Hybrid.* Fiecare dintre cele patru tipuri de implementări vine cu o serie de avantaje și dezavantaje, dezbătute pe larg în **capitolul 3**. Sunt, de asemenea, enumerate diferite aspecte care trebuie luate în considerare în faza inițială de proiectare unui DL, pentru a evalua care tip de stocare se adaptează mai bine resurselor proprii unei organizații. Sunt evaluate atât aspectele financiare, cât și cele ce țin de spațiul fizic, de resursele inginerești, de tipul, volumul și confidențialitatea datelor, aspecte ce stau la baza implementării și gestionării unui DL. Contribuțiile din acest capitol constau în prezentarea diferitelor caracteristici tehnice care trebuie luate în considerare în faza premergătoare implementării unei arhitecturi DL.

În **capitolul 4** am prezentat mediul de stocare Hadoop Distributed Files System (HDFS) ce stă la baza sistemului de stocare a datelor în cadrul unei arhitecturi DL implementat atât pe plan local (on-premises), cât și în cloud. Hadoop este un framework open-source cu procesare distribuită care poate procesa în paralel seturi mari de date, permițând aplicațiilor să utilizeze mii de noduri. Este una dintre cele mai folosite tehnologii de depozitare a datelor de tip DL datorită avantajelor care sunt prezentate în cadrul acestui capitol. Unul dintre cele mai mari avantaje constă în capacitatea framework-ului Hadoop de a permite stocarea și analiza de date

Big Data. Când vorbim despre Hadoop, de fapt ne referim la un întreg ecosistem care este prezentat în detaliu în cadrul acestui capitol. Acesta, are la bază diferite componente cum ar fi, sistemul de fișiere distribuit HDFS, MapReduce și YARN. HDFS este cea mai importantă componentă a ecosistemului Hadoop care permite stocarea seturilor mari de date de diferite tipuri: structurate, nestructurate și semi-structurate. Sistemul de fișiere distribuite Hadoop HDFS este un sistem de fișiere bazat pe Java care oferă un sistem de stocare a datelor scalabil, cu toleranță la erori, fiabil și eficient din punct de vedere al costurilor pentru Big Data. HDFS permite distribuția datelor în nodurile unui cluster. Pentru a face față oricăror defecțiuni, ce pot duce la indisponibilitatea unor date la un moment dat, sunt implementate tehnici de replicare a datelor pentru a maximiza disponibilitatea acestora. HDFS oferă un randament ridicat și latențe scăzute în timpul operațiilor de intrare-ieșire. **MapReduce** este a doua componentă centrală a ecosistemului Hadoop care asigură procesarea datelor. MapReduce este un cadru software care procesează cantități mari de date stocate în HDFS și permite procesarea de seturi mari de date într-un mod distribuit și paralel pe nodurile unui cluster. Framework-ul Hadoop, față de versiunea sa inițială 1, a fost îmbunătățit prin completarea lui cu noi module. În Hadoop 2 a fost introdus un nou framework **Hadoop YARN**, care aduce îmbunătățiri majore pe partea de gestionare a task-urilor și a resurselor prezente în cluster. Hadoop este o tehnologie de înaltă performanță ce permite implementarea arhitecturilor de gestionare a datelor Big Data, ideală pentru a suporta procesele de analiză avansată.

Folosind framework-ul Hadoop am implementat local două modele de stocare HDFS: Single-Node și Multi-Node Cluster pentru a evidenția diferențele dintre acestea și când anume se pretează folosirea lor. Nu cu mult timp în urmă un DL putea fi implementat doar local, cu ajutorul framework-ului Hadoop, proces ce implica timpi mari de implementare și costuri ridicate. În ultimii ani a emers o nouă metodă de implementare, în cloud, care facilitează accesul la acest nou concept de stocare Big Data.

În **capitolul 5** am realizat un studiu comparativ al arhitecturilor DL on-premises versus cloud DL. Cercetările rezultate au avut ca scop evidențierea diferențelor majore din punct de vedere al costurilor, al timpilor de implementare, al resurselor inginerești, al resurselor hardware și software necesare. În acest capitol am prezentat o serie de caracteristici pe care trebuie să le dețină un serviciu de cloud DL. Trebuie menționat faptul că nu toți furnizorii de servicii cloud oferă posibilitatea utilizatorilor finali să implementeze un DL. Cei care dețin această tehnologie ca serviciu contracost se bazează pe diferite tehnici de implementare. În acest sens am prezentat mai multe ecosisteme DL ce aparțin diferiților furnizori de servicii cloud DL pentru a evidenția aspectele tehnice ce caracterizează pe fiecare în parte și pentru a oferi direcții de orientare în alegerea unui furnizor.

Capitolul 6 abordează implementarea unei arhitecturi DL în cloud prin utilizarea tehnologiei puse la dispoziție de Microsoft și anume Azure Data Lake Storage Gen2. ADLS Gen2 se bazează de asemenea pe Apache Hadoop și Apache YARN și este o soluție care nu necesită instalarea de sisteme hardware sau software din partea utilizatorului. Este un serviciu cloud contracost în care se achită spațiul de stocare utilizat. Aceste costuri pot fi optimizate în funcție de necesitate prin reguli de DLM. În acest capitol am prezentat o arhitectură ADLS Gen2 unde sunt sintetizate vizual diferitele niveluri de procesare a datelor. Am detaliat aspecte tehnice importante ce stau la baza implementării unui DL în Azure, aspecte ce definesc fiecare

nivel din cadrul arhitecturii. Am evidențiat diferențele dintre stocarea pe obiecte și stocarea ierarhică specifică ADLS Gen2 și avantajele pe care le oferă aceasta din urmă. Am prezentat un model, general valabil, de stocare ierarhică a datelor cu distribuirea acestora în diferite zone din cadrul arhitecturii. Am propus tehnici de optimizare a costurilor bazate pe reguli de DLM. În realizarea unui lac de date în cloud trebuie să fie luate în considerare o serie de aspecte tehnice importante, cum ar fi: mecanismele de securitate a datelor, redundanța datelor, tehnici de procesare a datelor, aspecte care au fost evidențiate în acest capitol. Chiar dacă ele sunt prezentate explicit pentru mediul ADLS Gen2, sunt un punct de reper important în abordarea unei soluții cloud puse la dispoziție de oricare alt furnizor.

Cercetările teoretice se finalizează cu implementarea în practică a unui lac de date ADLS Gen2 pentru stocarea fișierelor log de mari dimensiuni. În **capitolul 7** am prezentat arhitectura DL pentru stocarea fișierelor WSAL, astfel încât acestea să poată fi supuse proceselor de analiză avansată pe termen lung. Este prezentată importanța stocării acestor fișiere log datorită informațiilor valoroase pe care le pot furniza în urma proceselor de analiză. Originalitatea acestei lucrări constă în faptul că sursa de date brute, de tipul fișierelor WSAL, pentru un DL nu a mai fost întâlnită în cercetările bibliografice realizate. În acest capitol sunt prezentate diferitele niveluri ce stau la baza arhitecturii și anume: cel de ingestie, de stocare, de explorare, de pregătire și cel de modelare și de servire a datelor. Am propus, de asemenea, un model de structurare ierarhică a datelor în cadrul lacului de date ce deservește întreaga arhitectură ADLS Gen2 pe parcursul diferitelor procese ELT. Pentru procesul de ingestie a datelor WSAL în DL s-a apelat la o tehnică *cross-platform command-line* și anume *Azure CLI*, ce permite execuția de comenzi *serverless* administrative în ADLS Gen2. Fișierele log încărcate în lacul de date, sunt supuse unui proces automat de transformare în fișiere parquet cu ajutorul unei funcții *Azure Blob Trigger* scrisă în limbajul Python, prezentată în acest capitol. Prin transformarea fișierelor log în fișiere parquet se realizează o comprimare a datelor, ceea ce va reduce cu mult dimensiunea inițială a fișierelor și implicit a costurilor de stocare. Din testele practice se observă o reducere de aproximativ 90% din dimensiunea inițială a datelor. Un alt avantaj al transformării fișierelor log în fișiere parquet constă în faptul că acestea sunt fișiere în format colonar, astfel datele devin structurate, prin urmare ușor de interogat. În nivelul de stocare au fost propuse tehnici de DLM asupra datelor cu scopul de a reduce ulterior costurile de stocare. Contribuția principală constă în proiectarea și implementarea unei arhitecturi DL ce deservește procesele de stocare și transformare a unor volume mari de date generate de serverele web.

Capitolul 8 prezintă contribuțiile și concluziile finale ale acestei lucrări de cercetare. Sunt enumerate contribuțiile aduse în acest domeniu, direcțiile viitoare de cercetare cât și lista lucrărilor publicate și susținute în reviste și conferințe internaționale, diseminând astfel rezultatele obținute în această perioadă de cercetare. Teza de doctorat se încheie cu o listă de referințe bibliografice și trei anexe ce cuprind informații cu privire la implementarea practică a arhitecturii DL propuse.

1.2 Contribuțiile lucrării de doctorat

Obiectivul principal al acestei teze constă în proiectarea și implementarea unei arhitecturi DL ce deservește diferitele procese ELT specifice unui DL, și anume: ingestia, stocarea și

transformarea unui volum mare de date generate de serverele web, folosind tehnica Azure Data Lake Storage Gen2 pusă la dispoziție de Microsoft.

Am abordat stocarea datelor WSAL în cloud DL deoarece această tehnică nu implică utilizarea de resurse hardware, software sau inginerești locale, resurse care nu sunt întotdeauna disponibile sau accesibile și care pot fi extrem de costisitoare. DL în cloud este tehnologia de vârf pentru gestionarea volumelor mari de date ce permite efectuarea de analize avansate în timp real. Cu ajutorul acestei noi tehnologii se dezvăluie potențialul datelor neanalizate anterior.

Contribuțiile tezei referă atât aspecte teoretice cât și aplicative.

- Analiza stadiului actual privind implementarea și utilizarea lacurilor de date. Un studiu comparativ între Data Lake și Data Warehouse (capitolul 2);
- Prezentarea elementelor tehnice care stau la baza implementării unei arhitecturi DL, fiind un domeniu nou ce vine cu un concept diferit de RDBMS (capitolul 3 și 6);
- Identificarea metodelor de implementare a unui DL. Un studiu comparativ între Data Lake on-premises și cloud Data Lake (capitolul 2 și 5);
- Identificarea și sintetizarea stocării datelor de tip WSAL pe termen lung în forma lor brută. Evidențierea beneficiilor și importanța informațiilor ce pot fi obținute în urma analizei acestor date (capitolul 7);
- Identificarea principalelor avantaje ale transformării fișierelor log în fișiere parquet.
- Propunerea unui algoritm pentru procesul de transformare a datelor (capitolul 7);
- Propunerea și implementarea unei arhitecturi DL care să deservească procesele de ingestie, stocare, prelucrare și analiză a fișierelor WSAL;
- Proiectarea unui model eficient de structurare ierarhică a datelor de tip WSAL, în nivelul de stocare din cadrul arhitecturii DL;
- Recomandări privind diferitele posibilități de îmbunătățire a structurii ierarhice în funcție de frecvența cu care sunt generate datele;
- Recomandări privind soluții de optimizare a costurilor bazat pe DLM;
- Proiectarea și implementarea unui proces pentru ingestia datelor, bazat pe o secvență de comenzi cross-platform prin intermediul Azure CLI, realizând o soluție serverless de încărcare a datelor în lacul de date, ce poate fi inclusă într-un proces automat batch de transferare a datelor de pe server în DL, la intervale regulate de timp;
- Proiectarea și implementarea unei funcții serverless *Azure Function* de tip *Blob Trigger* scrisă în limbajul Python care să realizeze procesul de transformare a fișierelor log în fișiere parquet în mod automat;
- Efectuarea de teste practice care să demonstreze eficiența și avantajele transformării fișierelor log în fișiere parquet.

Analiza fișierelor WSAL împreună cu metodologia de proiectare DL, respectiv procesul de încărcare cu ajutorul comenzilor Azure CLI și cel de transformare a fișierelor WSAL în fișiere *.parquet* prin realizarea unei funcții Azure Blob Trigger scrisă în Python reprezintă o noutate în domeniu.

Contribuțiile originale au fost comunicate și publicate la nivel național și internațional în cadrul revistelor de specialitate, conferințelor și rapoartelor de cercetare, după cum urmează:

Articole publicate în reviste aflate în zona galbenă (Q2) indexate WoS:

1. Zagan Elisabeta, Danubianu Mirela, "Data Lake Architecture for Storing and Transforming Web Server Access Log Files," in IEEE Access, vol. 11, pp. 40916-40929, **2023**, **FI 3.476,** doi:10.1109/ACCESS.2023.3270368

Articole publicate în reviste indexate WoS:

2. Zagan Elisabeta, Danubianu Mirela, "HADOOP: A Comparative Study between Single-Node and Multi-Node Cluster", International Journal of Advanced Computer Science and Applications (IJACSA), vol. 12 Issue 2, **2021**, doi:10.14569/IJACSA.2021.0120207
3. Zagan Elisabeta, Danubianu Mirela, "From Data Warehouse to a New Trend in Data Architectures - Data Lake", International Journal of Computer Science and Network Security, vol. 19, No. 3, pp. 30-35, **2019**

Articole prezentate la conferințe indexate WoS și IEEE:

4. Zagan Elisabeta, Danubianu Mirela, "ADLS Gen2 for web server log data analysis", 2022 International Conference on Development and Application Systems (DAS), 26-28 mai, **2022**, Suceava, România, pp. 161-166, doi:10.1109/DAS54948.2022.9786071
5. Zagan Elisabeta, Danubianu Mirela, "Cloud DATA LAKE: the new trend of data storage", 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications, 11-13 iunie, **2021**, Ankara, Turcia, doi:10.1109/HORA52670.2021.9461293
6. Zagan Elisabeta, Danubianu Mirela, "Data Lake Approaches: A Survey", International Conference on Development and Application Systems (DAS), 21-23 mai, **2020**, Suceava, România, pp. 189-193, doi:10.1109/DAS49615.2020.9108912

Articole prezentate la conferințe indexate BDI:

7. Zagan Elisabeta, Danubianu Mirela, "Data block saving policy in the Hadoop Architecture", 16th International Conference on European Integration - Realities and Perspectives, Danubius University, 14-15 mai, **2021**, Galați, România

2 Stadiul actual al cercetărilor în domeniul Data Lake

2.1 Conceptul Data Lake

James Dixon a prezentat pentru prima dată conceptul Data Lake. El a susținut că datele provenite din Data Marts extrase din Data Warehouse, pot fi a asemănate cu o sticlă de apă, "***curățată, ambalată și structurată pentru consum ușor***", în timp ce un **Data Lake** este asemănător unui "***lac de apă în stare naturală pe care diverși utilizatori pot veni să îl exploreze, să se scufunde sau să preia probe***" [6]. Lacul de date se bazează pe o nouă metodă de lucru care simplifică și îmbunătățește stocarea, gestionarea și analiza volumelor mari de date, provenite din surse diferite în forma lor brută. În [7] autorii definesc Data Lake ca fiind „o metodologie ce constă într-un depozit masiv de date bazat pe tehnologii cu costuri reduse, care îmbunătățește captarea, filtrarea, arhivarea și explorarea datelor brute în cadrul unei organizații". De asemenea, în [8] și [9] sunt formulate diferite definiții privind lacurile de date, cum ar fi „stocarea centralizată de ieri este lacul de date de astăzi", sau „Data Lake este un depozit de date scalabil care deține un volum mare de date brute în format original, ce sunt ingerate de motoarele de procesare fără a compromite structura datelor".

DL este un nou concept revoluționar care prioritizează stocarea datelor, prin faptul că mută procesul de transformare după cel de încărcare și astfel permite stocarea datelor în forma lor naturală. Scopul principal al unui lac de date este de a face accesibile datele organizaționale, provenite din diferite surse, diverșilor utilizatori finali, cum ar fi Business Analysts, Data Engineers, Data Scientists, etc. [10], care să valorifice informațiile ce vor avea ca țintă obținerea de performanțe în diferite branșe. Într-un sistem de tip Data Lake, datele sunt stocate în forma lor naturală fără a mai exista constrângerea de a avea un scop final bine definit încă de la început. Datele pot fi accesate de diferiți utilizatori care urmăresc obținerea de informații variate pentru scopuri diverse. Arhitectura de tip DL permite ca scopul final să determine căutarea și selectarea datelor din lacul de date, sistemul de căutare a datelor nemaifiind limitat de o structură predefinită pentru un anumit tip de analiză.

Data Lake-urile sunt concepute pentru a stoca și gestiona date în numeroase domenii de aplicații din viața reală, cum ar fi: Internetul lucrurilor (IoT) și orașele inteligente [11], producție [12], medicină [13], servicii de mobilitate (de exemplu, Uber) [14], biologie [15], aeronautică [16], rețele inteligente [17], etc. Astfel DL a devenit o nouă tehnică de stocare a datelor care să corespundă cerințelor și evoluției din domeniul IT.

2.2 Data Lake versus Data Warehouse

Inmon a conceput pentru prima dată definiția acceptată a unui depozit de date ca fiind „o colecție de date orientată pe subiect, nevolatilă, integrată, variabilă în timp, care vine în sprijinul deciziilor de management" [18]. O definiție completă este întâlnită în [19], și anume „un Data Warehouse este un sistem care reunește date din diferite surse într-un singur depozit de date central și coerent pentru a sprijini analiza datelor, extragerea datelor, inteligența

artificială (AI) și învățarea automată. Un sistem de depozit de date permite unei organizații să ruleze analize puternice pe volume uriașe (petabytes) de date istorice, spre deosebire de o bază de date standard". Principala contribuție a unui depozit de date constă în capacitatea sa de a transforma datele în informații strategice, accesibile responsabililor cu luarea deciziilor la cele mai înalte niveluri din cadrul unei organizații [20]. Astfel, un **Data Warehouse** poate fi văzut ca un depozit central de date integrate din una sau mai multe surse diferite, depozit de date care stochează date structurate, filtrate și procesate, date care au fost prelucrate pentru un scop specific, în timp ce, un **Data Lake** este un vast bazin de date pentru care scopul nu este bine definit [9]. Într-un DW datele sunt foarte atent curățate și verificate înainte de a fi stocate, garantând integritatea acestora. DW-urile sunt utilizate pe scară largă pentru stocarea datelor din cadrul unei organizații și au ca scop principal alimentarea aplicațiilor de business intelligence (BI) și business analytics (BA) [21], [22]. Datele găsite într-un depozit de date sunt folosite în scopuri precise în cadrul unei organizații. Cele mai importante caracteristici ale unui DW sunt următoarele:

- Datele sunt bine organizate și structurate;
- Datele sunt curățate în mod corespunzător de inexactități, de date corupte, de date duplicat sau discrepanțe;
- Datele nu sunt stocate decât atunci când scopul și utilizarea lor este bine definită;
- Toate datele unei organizații sunt într-un singur loc și respectă același format de date;
- Colectarea datelor se bazează pe pipeline-ul **Extract - Transform - Load (ETL)** care asigură acuratețea datelor.

Un inconvenient major în cazul DW constă în faptul că, premergător stocării datelor în DW, au loc procese consumatoare de timp în cadrul etapelor de transformare și agregare. Într-un DL, datele sunt ingerate în forma lor naturală fără a fi supuse proceselor de transformare ca în cazul DW. Datele stocate în forma lor brută pot fi accesate de diferiți utilizatori cu scopul de a fi procesate conform necesităților.

Odată ce un DL devine operațional, utilizatorii pot accesa toate datele colectate. Pentru aceasta au fost dezvoltate diferite motoare de căutare, care sunt special concepute pentru a interoga diverse tipuri de date. Aceste motoare de căutare diferă fundamental de cele utilizate în depozitele de date. În cazul DL-urilor organizarea și stocarea datelor este mult mai flexibilă.

Principalele caracteristici ale unui DL sunt:

- Posibilitatea de a obține rezultate din combinarea diferitelor tipuri de date;
- Posibilitatea de a stoca date structurate, semi-structurate, nestructurate într-un singur loc;
- Mai multă flexibilitate, deoarece există posibilitatea integrării de noi procese de analiză a datelor care să permită extragerea de noi rapoarte importante din datele brute;
- Abilitatea de a stoca date brute. Avantajul major constă în faptul că nu trebuie să existe un scop bine definit pentru a putea stoca datele;
- Posibilități nelimitate de interogare a datelor;
- Posibilitatea utilizării de diferite instrumente de BI, analiză avansată, analiză în timp real, machine learning, pentru a obține o perspectivă asupra datelor;
- Eliminarea silozurilor de date, ce sunt considerate ca fiind colecții de date deținute de un grup care nu sunt ușor sau complet accesibile altor grupuri din aceeași organizație;

- Utilizarea unei platforme eficiente de gestionare a datelor;
- Fluxul de colectare a datelor se bazează pe procesele **Extract-Load-Transform**.

În Figura 2-1 este reprezentată la nivel conceptual arhitectura unui DL.

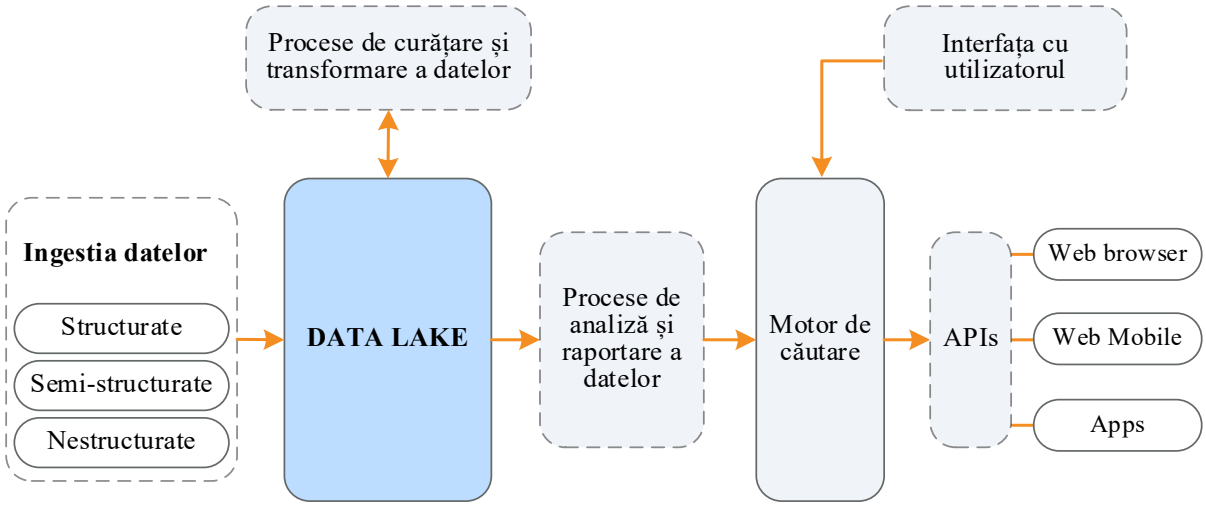


Figura 2-1: Arhitectura Data Lake la nivel conceptual [23]

Lacurile de date și depozitele de date sunt amândouă utilizate pe scară largă pentru stocarea Big Data dar în moduri și scopuri diverse. Un Data Lake este un mediu de stocare pentru volume mari de date brute care nu au un scop bine definit încă de la început. De cealaltă parte, un Data Warehouse este un depozit de date structurate, filtrate, care au fost deja prelucrate pentru a servi unui anumit scop final [24].

ETL este procesul de extragere a datelor din diferite sisteme sursă, de transformare a acestora prin aplicarea de metode adecvate precum concatenări, divizări, calcule, standardizări și de încărcare în sisteme țintă, spre exemplu într-un DW. În ETL datele merg dinspre sursă către destinație trecând printr-o etapă de transformare [25]. ELT diferă de ETL din punct de vedere al modului de migrare a datelor. ELT permite extragerea datelor din diferite surse pe care le încarcă într-o destinație, ca mai apoi datele să fie supuse proceselor de transformare și pregătite de consum. Eliminarea etapei intermediare de transformare dintre sursă și destinație eficientizează procesul de încărcare a datelor. ELT este utilizat de obicei în baze de date No-SQL cum ar fi HBase, Data Appliance [26] sau în lacuri de date. Tabelul 2-1 prezintă un studiu comparativ între cele două procese ETL și ELT.

Tabel 2-1: Studiu comparativ între ETL și ELT

	ETL specific Data Warehouse	**ELT specific Data Lake**
Tipuri de date	Structurate	Structurate, semi-structurate, nestructurate, brute
Procesarea datelor	**Schema on write** - Evaluarea și extragerea datelor din surse. - Validarea, curățarea și transformarea datelor.	**Schema on read** - Colectarea tuturor datelor în forma lor naturală.

	- Definirea structurilor de date pe baza unor reguli bine definite. - Stocarea datelor deja prelucrate. Complexitatea procesării datelor crește odată cu volumul de date.	Platforma țintă poate procesa rapid o cantitate semnificativă de date.
Obiective	Realizarea rapoartelor finale ce servesc unui scop precis.	- Evaluarea datelor. - Definirea și aplicarea diferitelor scheme de prelucrare a datelor. - Realizarea rapoartelor finale în scopuri diverse.
Volum de date de procesat	Ideal pentru seturi mari date cu cerințe complicate de transformare.	Ideal pentru seturi mari, nelimitate de date structurate sau nestructurate care necesită viteză și eficiență.
Transformarea	Transformarea are loc în stadiul inițial înaintea salvării pe serverul țintă.	Transformarea are loc pe serverul țintă după ce datele au fost deja salvate.
Timpi de încărcare	Încărcarea datelor în sistemul țintă depind de timpii de transformare a acestora, deci un consum mare de timp.	Datele se încarcă direct în sistemul țintă. Timpi de încărcare relativ reduși.
Timpul de transformare	Procesul de transformare depinde mult de volumul de date prelucrat. Pe măsură ce dimensiunea datelor extrase crește, crește și timpul de transformare.	Transformarea în ELT nu depinde de procesul de extragere.
Maturitatea procesului	Procesul este folosit de peste două decenii. Acesta este bine documentat și sunt disponibile cele mai bune practici.	Concept relativ nou și complex de implementat.
Suport pentru date nestructurate	Suportă în principal datele structurate.	Suportă date de orice tip.

Deși există diferențe între ETL și ELT, acestea sunt utilizate pentru a îndeplini exact aceeași cerință, și anume pregătirea datelor pentru analiză și utilizarea lor pentru îmbunătățirea procesului de luare a deciziilor. ELT spre deosebire de ETL este caracterizat de o mai bună flexibilitate și performanță. Proiectarea și execuția ELT poate necesita mai mult efort, dar oferă mai multe beneficii decât ETL pe termen lung.

În cele ce urmează sunt prezentate principalele diferențe dintre Data Wharehouse și Data Lake, văzute din perspectiva următoarelor criterii:

- **Volumul datelor stocate**. În etapa de proiectare și dezvoltare a unui DW este dedicat foarte mult timp analizării datelor sursă și proiectării fluxului lor de transformare astfel încât să se obțină formate adecvate structurii tabelelor ce constituie depozitul central de date. O atenție sporită în cadrul acestui proces este acordată selecției datelor. Conform definiției date de Inmon [10], datele care nu corespund scopului urmărit vor fi ignorate, tocmai pentru a economisi spațiu de stocare, ceea ce va conduce la un DW cu performanțe mai bune și implicit costuri mai reduse. În contrast, DL are capacitatea de a stoca toate datele chiar dacă, inițial, parte din acestea nu par să fie importante în etapa

finală de analiză. Acest avantaj nu este unul de neglijat pentru că cerințele pot să difere în timp chiar și în cazul DL-urilor. Având toate datele stocate în forma lor naturală încă de la începutul procesului de dezvoltare va exista capacitatea de a răspunde oricărei noi cerințe. Stocarea completă a datelor oferă, de asemenea posibilitatea de a face orice revenire în timp cu scopul refacerii analizelor și a generării de noi rapoarte corespunzătoare momentelor vizate.

- **Tipuri de date stocate**. În general, depozitele de date stochează date extrase din sistemele tranzacționale. Sursele de date non-tradiționale, cum ar fi fișierele log ale serverelor web, datele capturate de la diferiți senzori, sau provenite de pe rețelele de socializare, textele și imaginile sunt în mare parte ignorate. Într-o arhitectură de tip DW procesul de stocare și prelucrare a datelor este cunoscut ca **Schema on Write**, aceasta însemnând că datele trebuie să fie mai întâi prelucrate și apoi transformate astfel încât să corespundă unei anumite scheme în momentul depozitării. Odată cu creșterea rapidă a volumului de date această metodă s-a dovedit a fi nepotrivită dezvoltându-se o nouă abordare și anume **Schema on Read,** utilizată pentru implementarea DL, deoarece salvarea datelor în forma lor naturală oferă mai multă flexibilitate în procesul de stocare și analiză. Datele pierdute prin metoda Schema on Write s-au dovedit a fi date importante, astfel că a fost necesară găsirea unei noi tehnologii prin care să se evite pierderea acestora.
- **Utilizatori.** În majoritatea organizațiilor, un procent de 80% este reprezentat de utilizatorii așa-numiți "utilizatori operaționali" care urmăresc să obțină rapoartele zilnice. DW, deoarece este bine structurat, ușor de utilizat și înțeles, este construit special să poată oferi răspunsuri la problemele acestora. Restul sunt utilizatori ce fac parte din categoria decidenților de nivel superior, care necesită informații de tip tendințe sau tipare comportamentale necesare pentru predicții, informații rezultate din modelarea datelor. Într-un procent de 10% pot fi menționați acei utilizatori care execută analize mai aprofundate asupra datelor și care adesea vor fi nevoiți să facă noi căutări în fișierele sursă pentru a obține date care nu sunt incluse în depozit, iar uneori vor aduce date din afara organizației. Un procent redus de utilizatori este reprezentat de acei "utilizatori cercetători" care fac analize aprofundate și care pot crea noi surse de date bazate pe cercetare. Aceștia au abilitatea de a încrucișa diferite tipuri de date generând astfel noi seturi de întrebări cu scopul de a obține noi rezultate. Acești utilizatori se bazează foarte puțin, uneori aproape deloc, pe suportul oferit de depozitul de date, cel mai adesea fiind nevoiți să depășească limitele acestuia, utilizând noi instrumente analitice avansate în scopuri de cercetare. DL-urile suportă aceste tipuri de utilizatori fără probleme, iar toate resursele de care au aceștia nevoie se găsesc într-un singur loc.
- **Versatilitate.** Un mare dezavantaj pe care îl prezintă DW este faptul că solicită timpi mari pentru a aduce modificări structurale. În primul rând necesită un timp considerabil de mare pentru a crea o structură corectă a unui depozit de date. Modificarea acestei structuri necesită timpi mari de lucru din partea proiectanților datorită complexității din procesele de încărcare a datelor și a celor de analiză și raportare. Uneori acești timpi sunt prea mari, organizațiile având nevoie de răspunsuri înainte ca proiectanții să reușească să adapteze depozitul de date la noile cerințe. Odată cu evoluția tehnologiei

întrebările au venit cu necesitatea de a avea răspunsuri imediate. Într-un DL datele pot fi, în orice moment, accesate de diferiți utilizatori autorizați cu scopul de a le explora prin diverse metode, astfel încât să găsească răspunsurile solicitate fără a fi limitați de o structură rigidă și bine pusă la punct ca în cazul DW.

- **Timp de acces la date.** După cum ne putem da seama din diferențele dintre DW și DL, deducem faptul că, DL oferă utilizatorilor autorizați răspunsurile dorite mult mai rapid, deoarece lacurile de date permit accesul la date înainte ca acestea să fie supuse proceselor de transformare care sunt mari consumatoare de timp. Datele sunt disponibile imediat ce ele sunt încărcate în lacul de date.

După analizarea celor două metode de stocare a datelor, putem afirma cu certitudine că nu există nici o concurență între **Data Lake** și **Data Warehouse**, ele fiind două concepte complet diferite. În concluzie, spre deosebire de DW, un DL este mai versatil, cu capacitate de stocare nelimitată, cu autoscalare și costuri mai mici de stocare.

În Tabelul 2-2 sunt prezentate comparativ caracteristicile principale ale DW și DL.

Tabel 2-2: Principalele diferențe dintre Data Warehouse și Data Lake [23]

	Data Warehouse	Data Lake
Costuri	Atât costurile de stocare cât și cele operaționale sunt semnificative.	Costurile de stocare sunt relativ mici. Gestionarea lacurilor de date necesită mai puțin timp, ceea ce reduce și costurile operaționale.
Date	Date precise, de cele mai multe ori structurate sau semi-structurate.	Date în forma lor brută (structurate, semi-structurate, nestructurate, date binare, date provenite de la sistemele tranzacționale, log-uri, date de la senzori, date provenite de la diferite aplicații, etc).
Schemă	Schema on Write (ETL process): Schema fiind definită înainte de stocarea datelor.	Schema on Read (ELT process): Schema fiind definită după stocarea datelor.
Tipuri de analiză	Vizualizare de date, BI, analiza datelor.	Analiză predictivă, învățare automată, vizualizare de date, BI și analiză Big Data.
Agilitate	Este nevoie de timp pentru a face modificări în structura datelor, fiind un depozit de date bine structurat. Rezultă o structură mai puțin agilă, cu o configurație fixă.	Oferă posibilitatea de a schimba cu ușurință modele și interogările efectuate asupra datelor. Este o structură foarte agilă, poate fi configurată și reconfigurată ori de câte ori este necesar.
Securitate	Performanțe ridicate în ceea ce privește securitatea.	Grad înalt de securitate cu mecanisme noi în continuă dezvoltare.
Utilitate	Aplicat mai ales în afaceri de mici și medii dimensiuni.	Folosit în principal în domeniile științifice și în organizațiile cu un volum mare de date.

DL-urile devin din ce în ce mai importante pentru marile organizații. Dacă ne uităm mai atent la evoluția rapidă a conceptului IoT, și în general la varietatea datelor și viteza cu care

sunt generate acestea, DL reprezintă o soluție performantă pentru stocarea datelor Big Data datorită diversității surselor ce generează datele și diferitelor tipuri de date ce sunt generate cu o viteză uluitor de mare.

2.3 Data Lake în Big Data Analytics

În era Big Data, datele provenite din diverse domenii și din diverse aplicații sunt generate în mod continuu, iar cantitatea de date crește exponențial. Sub presiunea impactului acestor resurse uriașe de date, mediul de afaceri, mediul academic, industrial și alte domenii sunt într-o continuă căutare de noi soluții performante care să satisfacă stocarea și procesarea acestor date [27]. În acest context, metodele tradiționale nu mai reușesc să facă față volumelor mari și variate de date [28], iar pentru a satisface aceste noi necesități a apărut conceptul Data Lake, un depozit de date în care sunt colectate și stocate strategic toate datele generate, fie din surse interne sau provenite din surse externe, indiferent dacă ele sunt considerate, la momentul respectiv, utile sau nu, fără a ține seama de natura sau forma lor. Aceste „fluxuri" de date vin în mai multe formate, și anume date structurate provenite din baze de date relaționale sau din foi de calcul, date nestructurate provenite din rețele sociale, video, e-mail, text, date provenite de la senzori, etc., sau date semi-structurate (date de tip log , XML, etc.). Este esențială stocarea acestor date deoarece au sau ar putea avea sensuri ascunse care pot influența semnificativ procesul de luare a deciziilor. Data Lake este ideal pentru Big Data analytics într-un mediu mixt de date, deoarece permit analize rapide, uneori chiar în timp real, pe date complexe și de volume mari. DL devine o metodă modernă de organizare și dezvoltare a următoarei generații de sisteme performante care să facă față cu succes noilor provocări legate de volumele mari de date [29].

2.4 Analiza critică a literaturii de specialitate

Literatura de specialitate este bogată în materiale științifice care prezintă diferențele dintre DW și DL, dar proiectarea sau punerea în aplicare a unui DL este încă la început. Încă se fac cercetări în modul de abordare din punct de vedere al calității datelor (data quality), securității datelor (data security), ciclului de viață a datelor (data life cycle) sau al gravitației datelor (data gravity).

În articolul [30] autorii expun modul de trecere de la DW la DL și prezintă avantajele dar și neajunsurile pe care le poate ridica o arhitectură modernă de tip DL. Se conturează o definiție a unui DL și se introduce conceptul data gravity care aduce un aport major în proiectarea și implementarea fizică a unui lac de date. În viziunea autorilor, Data Warehouse și Data Marts au fost considerate, până recent, ca fiind singura soluție de furnizare de date precise și de încredere în cadrul marilor organizații, și totuși, odată cu apariția paradigmei IoT, a utilizării smartphone-urilor și a diferitelor aplicații, acestea au fost depășite și puse în imposibilitatea de a mai putea stoca atât de multe și variate date. În 2014 Apache Hadoop a făcut publică apariția unui nou concept DL care să permită stocarea acestor noi tipuri de date. În literatura de specialitate au apărut diferite propuneri și soluții care abordează acest nou concept. Totuși, proiectarea sau punerea în aplicare a unui DL se află încă într-un stadiu incipient. Autorii articolului încearcă să demonstreze că DL este mult mai mult decât o metodologie. DL este mai degrabă o nouă arhitectură de date compusă din partea hardware, software și design

conceptual. Prin definiția DL propusă, autorii vor să prezinte la nivel teoretic o soluție legată de principiile de administrare a datelor (Data Governance Principles) ca fiind principii cheie ale lacului de date. Totodată se ridică problema legată de data gravity și anume, faptul că datele, la un moment dat, pot fi supuse unor reguli de migrare astfel încât nu vor mai putea fi mutate. Conformitatea datelor, sensibilitatea, securitatea și desigur volumul de date limitează mutarea datelor, fiind astfel evidențiate problemele puse de data gravity. Lucrarea [30] este prima care introduce acest factor: impactul pe care îl are data gravity în proiectarea arhitecturii fizice a unui DL.

În [31] autorii au ca scop principal identificarea unor legături dintre cele trei concepte diferite, și anume **Big Data**, **Data Lake** și **Fast Data** (Figura 2-2).

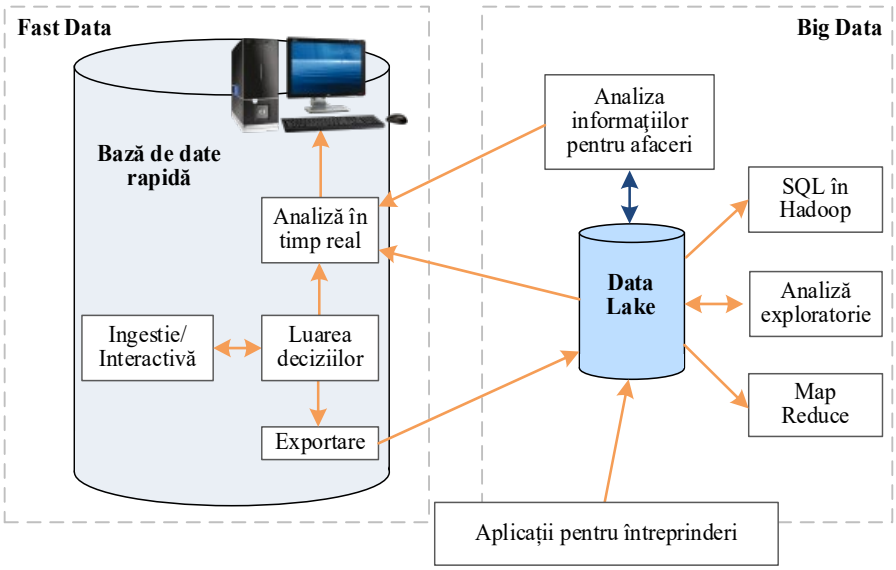


Figura 2-2: Exemplu de arhitectură modernă de date care unesc cele trei concepte Big Data, Fast Data și Data Lake [31]

Caracteristicile principale scoase în evidență sunt:

- Datele de tip **Big Data** pot fi structurate, semi-structurate și nestructurate și sunt caracterizate prin volum, viteză, varietate, veridicitate, variabilitate, valoare și vizibilitate.
- Un **DL** conține o cantitate mare de date brute în formatul lor original și are capacitatea de a permite prelucrarea datelor fără să fie afectată structura acestora. Poate fi imaginat ca un lac în care pot fi aduse în timp real, într-un singur loc, atât date vechi/istorice, cât și date noi acumulate în timp real fără a se respecta o anumită schemă. Datele sunt transformate în forma potrivită atunci când sunt interogate. Un DL este descris într-un catalog centralizat, este bine gestionat și protejat, are o mare disponibilitate, și utilizează sisteme de analiză și de urmărire avansate.
- **Fast Data** conține date structurate și nestructurate care sunt time-sensitive, motiv pentru care sunt colectate și prelucrate imediat. Acestea sunt folosite în procesele în timp real care au nevoie de răspunsuri extrem de rapide în rezolvarea problemelor. În acest scop este nevoie de un sistem de streaming capabil să transmită evenimente la fel de repede precum au și venit, și de o locație de stocare a datelor capabilă să permită procesarea fiecărei date imediat după ingestie.

Autorii au ajuns la concluzia că Fast Data intră în categoria Big Data atunci când trebuie procesate volume mari de date. În acest caz aceste două concepte prezintă puncte comune. Prima lor concluzie este că Big Data sunt date în repaus, în timp ce Fast Data sunt date în mișcare. Comparând Big Data cu DL s-a ajuns la concluzia că DL este o continuare a evoluției primului concept Big Data astfel ajungându-se într-un fel de spirală care leagă cele trei concepte. În Figura 2-2 este prezentată interdependența logică dintre cele trei concepte, așa cum este văzută de autori în [31].

Daniel E. O'Leary a încercat să examineze noțiunea de DL punând-o în contrast cu soluțiile deja existente, luând ca termen de comparație un DW [32]. Totodată a punctat riscurile care pot apărea investigând încorporarea diferitelor aplicații ale inteligenței artificiale și ale inteligenței umane în lacul de date. Pentru minimizarea acestora au propus o abordare numită Master Data Management (MDM) care urmărește și armonizează datele în cadrul diferitelor aplicații. Inițierea și aplicarea MDM conduc la implementarea cu succes a unui DL, în timp ce ignorarea acestuia poate genera multe probleme. Într-o arhitectura DL combinația dintre AI și crowdsourcing poate aduce numeroase beneficii legate în primul rând de calitatea datelor care să conducă la inferențe adecvate și de generarea de metadate complete și curate care permit integrarea diferitelor surse de date. De asemenea această combinație poate fi utilă pentru generarea de cunoștințe și ontologii pe care să se bazeze procesul MDM. AI și crowdsourcing pot fi folosite pentru a genera etichete pe date care pot facilita utilizarea și definirea acestora. AI poate fi utilizat pentru a asocia factori suplimentari legați de date. În acest caz se au în vedere, spre exemplu, acele abordări care ar putea genera automat etichete care să faciliteze utilizarea ulterioară a datelor. La rândul său, crowdsourcing este folosit pentru recunoașterea acestor etichete, care au capacitatea de a micșora eforturile depuse pentru MDM și pot fi folosite în generarea unei structuri ierarhice a datelor în folosul utilizatorilor care pot înțelege modul în care relaționează sursele diferite de date.

AI de asemenea poate fi utilizat pentru eliminarea datelor duplicat. Enrico Coiera și colaboratorii săi [33] au dezvoltat un sistem de eliminare a datelor duplicat dintr-o bază de date ce ținea evidența clinică a unor pacienți. Pei-Yun Hsueh et al. [34] au examinat datele crowdsourcing pentru a îmbunătăți calitatea datelor. Pe de altă parte, utilizarea DL-ului necesită o gestionare adecvată a datelor pentru a evita problemele de confidențialitate. DL ar trebui să fie capabil să integreze și să analizeze simultan mai multe surse de date ca parte a calculului în memorie (in-memory computing) [35]. În acest design conceptual, întregul set de date conținute în sursele de date multiple ar fi disponibil pentru analiza în timp real.

În altă perspectivă [36], se prezintă **Data Lake open source - CoreDB** care oferă cercetătorilor și dezvoltatorilor o singură aplicație REST API pentru organizarea, indexarea și interogarea datelor și metadatelor. CoreDB este un open source care gestionează mai multe tipuri de baze de date (de la baze de date relaționale, NoSQL) și oferă un design integrat pentru securitate și urmărire a modificărilor efectuate asupra datelor.

Autorii doresc să ofere o soluție simplă de stocare și manevrare a datelor și metadatelor în contextul în care organizațiile în ultima perioadă se confruntă cu problema gestionării volumelor mari de date, de tipuri diferite. Analiștii sunt nevoiți să se ocupe de o mulțime de date digitale generate prin intermediul rețelelor de socializare, bloguri, comunități online și aplicații mobile care pot să formeze un lac de date complex. Organizarea și indexarea acestui

volum imens de date este o provocare și poate fi gestionat cu ajutorul noilor tehnologii apărute. **CoreDB** este o soluție oferită în acest sens. În Figura 2-3 este prezentată arhitectura open source CoreDB completă.

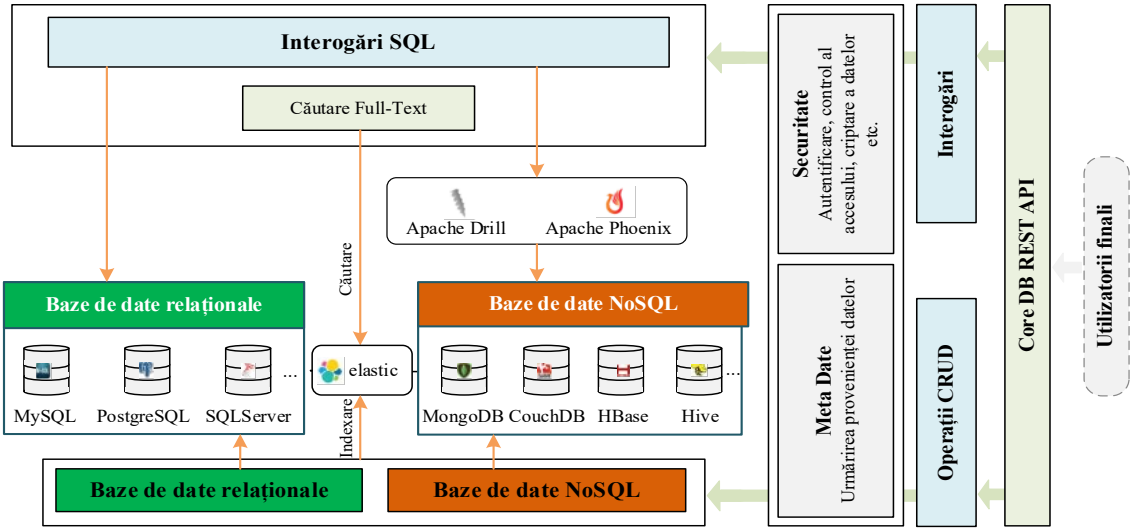


Figura 2-3: Arhitectura CoreDB [36]

Autorii au implementat un DL în CoreDB pentru stocarea a peste 15 milioane de informații din social media. Ulterior s-a creat o bază de date relațională în care au fost salvate date legate de programul de sănătate bugetar. Peste toate aceste date s-au efectuat diferite căutări full-text pentru a scoate în evidență capacitatea de indexare a datelor în CoreDB.

Munshi și colaboratorii prezintă în [37] un ecosistem **Smart Grid Big Data** bazat pe arhitectura Lambda, care este capabilă să gestioneze cantități enorme de date și să efectueze operațiuni în batch și în timp real. Ecosistemul prezentat folosește Hadoop Big Data Lake pentru a stoca diverse tipuri de **Smart Grid Data**, inclusiv **Smart Meter**, imagini și date video care pot fi exploatate mai târziu prin intermediul diferitelor procese. S-a utilizat stocarea datelor într-un **Cloud Data Lake,** întrucât se permite depozitarea a multiple tipuri de date provenite din surse diferite într-un singur loc. Pentru a testa capacitatea ecosistemului prezentat în Figura 2-4, s-au efectuat aplicații de vizualizare și extragere a datelor în timp real.

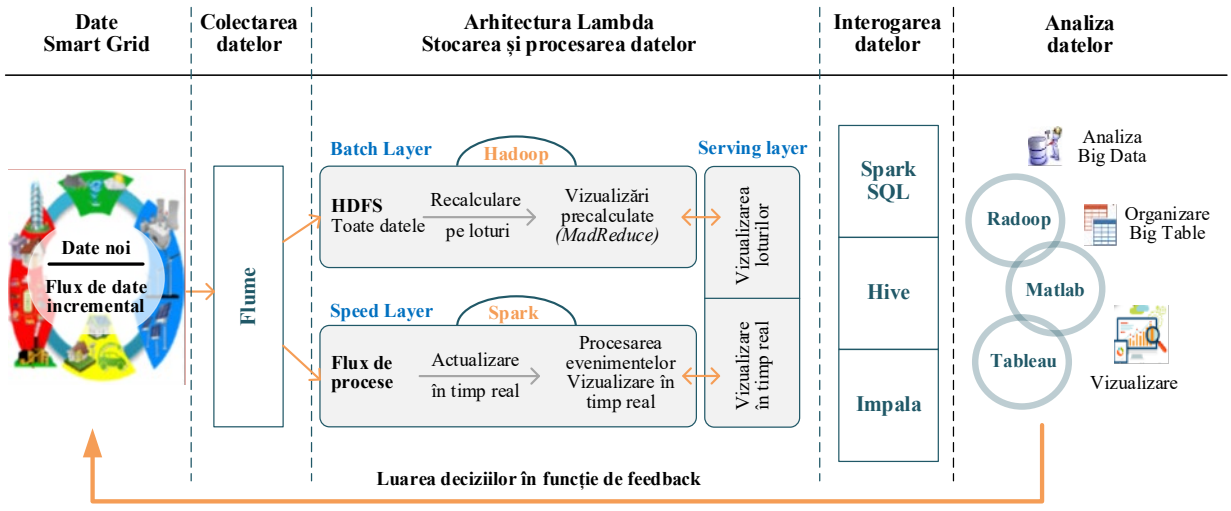


Figura 2-4: Ecosistemul Smart Grid Big Data [37]

ByungRae Cha și alții prezintă în [38] importanța pe care o au tehnologiile de stocare și prelucrare a volumelor mari de date, pentru a satisface tendințele din zilele noastre în domeniul tehnologiei informației și comunicațiilor, concentrate pe IoT, Big Data, Cyber Physical System (CPS) și AI. Pentru a evalua performanțele de stocare, au fost executate o serie de teste experimentale pe mediul open source Ceph, utilizând mediul de stocare Abyss, iar performanțele de răspuns au fost evaluate în rețea folosind Korea Advanced Research Network (KOREN). Pentru a îmbunătăți performanțele și securitatea de stocare a datelor distribuite în clusterul Abyss, au fost executate diverse teste de performanță pe suporturile de stocare te tip disc, teste asupra traficului și legăturilor în rețea, teste de securitate ale sandboxului Cuckoo și teste de detectare de malware Yara. Pentru a rezolva problema one-way DL numit Garbage Dump, autorii au propus aplicarea topologiei matematice și a tehnologiei de învățare automată precum și proiectarea unui framework pentru lacul de date.

În [39] este propus un framework pentru lacuri de date publice care să controleze și să protejeze confidențialitatea datelor. Obiectivul principal al acestei lucrări este stimularea distribuirii datelor valoroase în procesele de analiză Big Data și promovarea dezvoltării de noi tehnologii. Autorii propun o arhitectură (Figura 2-5) în care fiecare furnizor de date apelează la stocarea în cloud pentru a depozita datele ce urmează a fi făcute publice.

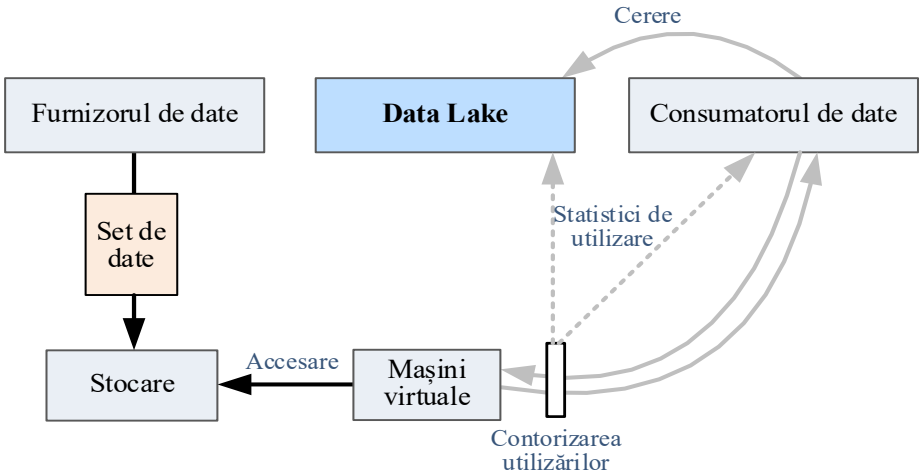


Figura 2-5: Protocol de distribuire a datelor în arhitectura Data Lake [39]

În cloud consumatorii autorizați pot accesa datele prin conexiuni remote Procedure Calls (RPC) sau prin intermediul aplicațiilor Application Programming Interfaces (API). Astfel DL va îndeplini doar funcția de stocare a metadatelor și a informațiilor de gestionare a seturilor de date. Metadatele vor păstra informații legate de volum, formă, sursă, limitări de utilizare, prețuri, etc., informații necesare consumatorilor în scopul închirierii acestora.

În [40], Fang punctează importanța DL în noua eră tehnologică pentru dezvoltarea de noi sisteme care să gestioneze multitudinea de date. În perspectiva autorului, un DL este o metodologie activată de un depozit masiv de date bazat pe tehnologii cu costuri reduse care îmbunătățesc captarea, definirea, arhivarea și explorarea datelor brute în cadrul unei organizații.

DL a devenit popular deoarece reușește în mod eficient să facă față provocărilor Big Data. Datele salvate în forma lor naturală oferă mai multă flexibilitate analiștilor care procesează datele cu scopul de a obține informații cât mai detaliate și variate. În Figura 2-6 este propusă o

arhitectură hibridă alcătuită dintr-un Data Lake și un Data Warehouse. DL are ca scop stocarea datelor de unde se alimentează DW.

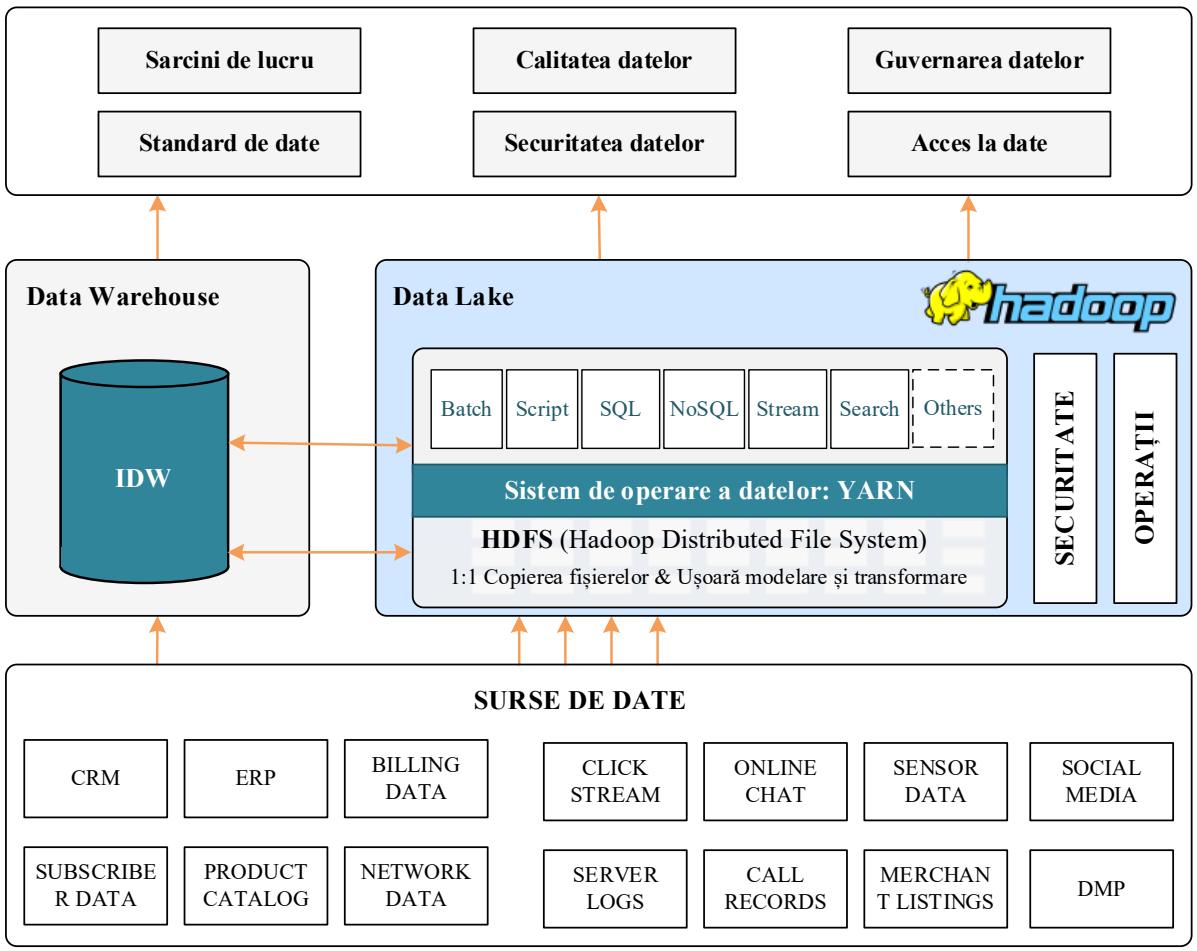


Figura 2-6: Ecosistem hibrid Data Lake și Data Warehouse [40]

Ecosistemul hibrid de gestionare a datelor, format din DL și DW, este optim pentru mediile cu volume mari de date și permite îndeplinirea cerințelor utilizatorilor fără ca aceștia să poată avea acces direct la date.

Farrugia și alții au construit un instrument pentru inspectarea și gestionarea lacurilor de date [41]. Instrumentul funcționează prin extragerea de metadate din baza de date Hive, pe o platformă comună **Hadoop**, care conține un volum de date reale de ordinul multi-terabyte. Aceste metadate sunt folosite pentru a trasa un grafic al relațiilor dintre entități pe baza corelării coloanelor, cu scopul de a permite aplicarea de tehnici de analiză a rețelelor sociale Social Network Analysis (SNA), în vederea descoperirii unor proprietăți importante ale datelor acumulate, cum ar fi depistarea de relații anterioare necunoscute între entitățile de date.

Autorii consideră că DL vor fi folosite masiv în viitor pentru a alimenta cu date diverse aplicații. Pentru a obține informații valoroase, este esențial să fie identificate seturile de date utile. Au fost constatate următoarele provocări majore:

- Găsirea datelor relevante și utilizarea acestora;
- Gestionarea riscurilor și securitatea datelor.

Autorii demonstrează eficiența instrumentului Data Lake Introspection (DLI) care permite identificarea de noi relații între date prin coroborarea tabelelor Hive stocate în mediul Hadoop.

În [42] autorii evidențiază o serie de caracteristici ale lacurilor de date, prin realizarea unui studiu comparativ între un DL și un DW și subliniază faptul că DL pare a fi o arhitectură ce prezintă un interes major în mediul afacerilor mai mult decât în mediul academic. Datorită faptului că DL este un concept relativ nou, definițiile, caracteristicile, arhitectura, implementarea și utilizarea sunt mult mai răspândite în articolele web și blogurile de specialitate decât în lucrările academice. Autorii sunt preocupați de faptul că actualele lacuri de date se concentrează pe stocarea masivă de date și ignoră cum sunt datele utilizate, guvernate, definite și securizate. Aceste lipsuri majore pot transforma lacurile de date în *mlaștini de date* greu de analizat. În momentul de față lacurile de date încearcă să rezolve provocările impuse de caracteristicile Big Data, și anume: volumul, viteza, veridicitatea, varietatea și valoarea. În ultima perioadă DL au primit o atenție sporită devenind tot mai populare datorită evoluției surselor de generare a datelor precum IoT, rețele sociale, dispozitive inteligente, etc. Cu toate acestea, în prezent, lacurile de date nu amenință să înlocuiască depozitele de date, deoarece nu au rezolvat încă problemele și provocările menționate. Autorii preconizează că cele două concepte Data Warehouse și Data Lake vor fi combinate în viitorul apropiat, adică DW și DL vor deveni un singur concept prin unificarea proprietăților celor două.

În [43] autorii au exemplificat o metodă de analiză a volumelor mari de date cu ajutorul librăriilor Fuzzy Search. Această analiză a fost efectuată pe un Data Lake Azure situat pe o platformă cloud. Soluția dezvoltată de autori permite un control complet și complex asupra datelor pe parcursul întregului proces EPS (Extract, Process and Store) într-un Big Data Lake. Autorii urmăresc să demonstreze potențialul pe care îl au librăriile Fuzzy din cadrul Azure Data Lake în procesul de căutare a datelor Big Data. Soluția prezentată oferă capacitatea de a prelucra nu numai date structurate, dar și nestructurate, stocate în formatul natural în DL, utilizând o interfață care reprezintă datele ca un set de rânduri. Soluția propusă simplifică, de asemenea, procesul de prelucrare și transformare fuzzy a datelor prin utilizarea expresiilor **U-SQL SELECT**, spre deosebire de procesarea bazată pe Hadoop/Spark, care necesită adaptarea la un anumit model de procesare, de exemplu, **MapReduce** și implementarea de funcții de procesare dedicate. Tehnicile fuzzy furnizate sub formă de funcții (UDF) care sunt apelate direct din interogările U-SQL, permit nu numai procesarea datelor într-un mod declarativ, ci și optimizarea planurilor de execuție U-SQL pentru a aloca mai puține resurse și pentru a reduce costul necesar procesării datelor. Intenția autorilor a fost să dezvolte o bibliotecă de căutare Fuzzy pentru Big Data Lake ca un instrument universal care oferă metode de scalare, procesare fuzzy a datelor pe cloud Azure pentru diverse domenii de date analizate.

În [44] autorii propun o arhitectura Personal Data Lake (PDL) cu scopul de a întâmpina și rezolva o serie de probleme cu care se confruntă un DL. Se pornește de la ideea că rezolvarea unor probleme într-un plan restrâns, ca în cazul PDL, și aplicarea unor strategii de stocare și procesare a datelor pe plan restrâns/personal va putea duce ulterior către o arhitectură la nivel global care să îndeplinească cu succes problemele de confidențialitate, securitate și gravitație a datelor. Termenul Data Gravity a fost inventat de Dave McCrory [45] pentru a descrie fenomenul în care numărul sau cantitatea și viteza, cu care serviciile, aplicațiile și chiar clienții sunt atrași de date, crește pe măsură ce masa datelor crește.

În Figura 2-7 este prezentată o arhitectura Data Lake în ecosistemul Hadoop, propusă de autori în lucrarea [46]. Această arhitectură înglobează diferite niveluri, în care datele sunt stocate, organizate și pregătite pentru consum, și anume:

- *Nivelul de achiziție a datelor* este nivelul unde are loc ingestia și migrarea datelor în fișierele sistem HDFS, indiferent de tipul acestora.
- *Nivelul de curățare a datelor,* în arhitecturile de date moderne Hadoop, joacă un rol principal în stocarea și curățarea datelor. Toate datele sunt stocate în lacul de date și sunt organizate, prelucrate ca mai apoi să fie puse la dispoziția utilizatorilor prin intermediul nivelurilor superioare.
- *Nivelul de furnizare a datelor*, unde rapoartele operaționale și de analiză sunt realizate cu ajutorul tehnologiilor tradiționale, utilizate în Relational Database Management System (RDBMS), au capacitatea de a interoga și analiza datele.
- *Nivelul de consum* este reprezentat de interfața utilizatorilor. Aici sunt disponibile o gamă variată de tehnologii care să îndeplinească acest scop. În unele situații acest nivel poate permite utilizatorilor să se conecteze direct la lacul de date pentru preluarea de informații.

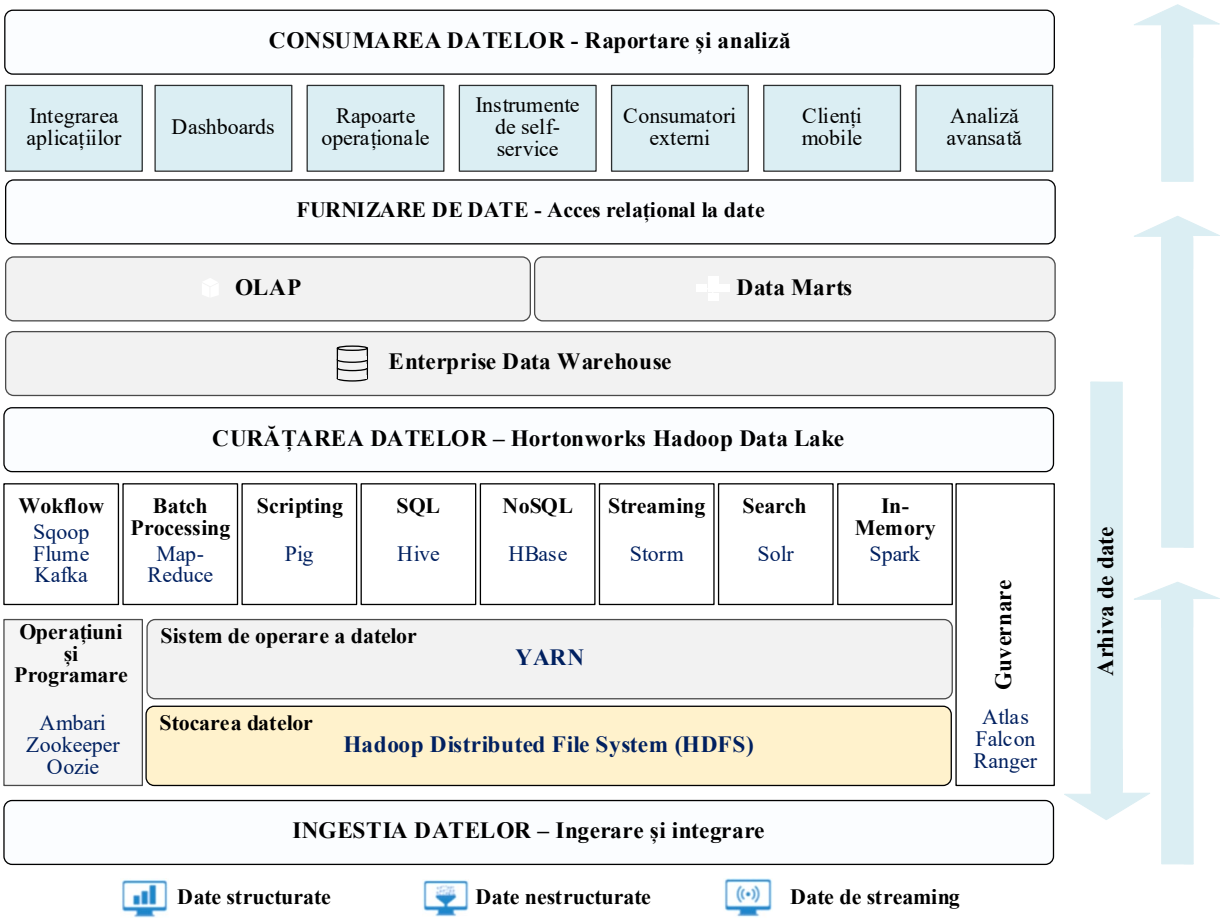


Figura 2-7: Arhitectura Data Lake propusă în [46]

Acest model corespunde nivelurilor fizice care stau la baza întregii arhitecturi DL implementat în Hadoop.

3 Arhitectura Data Lake

Înainte de apariția conceptului Big Data existau puține posibilități de procesare a seturilor de date de ordinul terabytes sau mai mare. Pe măsură ce a crescut capacitatea de generare a datelor, a crescut și necesitatea stocării și procesării acestora. Volumul mare de date a adus cu sine necesitatea unor schimbări semnificative în arhitectura sistemelor de stocare și procesare. În ultimii ani, este întâlnit tot mai des conceptul DL, care poate fi văzut ca un lac de date, unde pot fi depozitate tipuri diferite de date și care pot fi accesate de diverși utilizatori pentru a fi explorate [47]. Astfel, DL poate fi definit ca un depozit centralizat pentru stocarea volumelor mari de date structurate, semistructurate sau nestructurate, provenite din surse diferite, care permite efectuarea de procese avansate de analiză a datelor. DL se bazează pe o nouă tehnologie cu un mare potențial, astfel încât se preconizează că în viitorul apropiat va deveni un sistem important de stocare și gestionare a datelor pentru a putea exploata la maxim informațiile care pot fi obținute din varietatea de date pe care le deține [48].

O arhitectură de tip DL susține procese complexe de colectare, stocare, transformare și analiză avansată a datelor într-un singur loc, procese asociate anumitor niveluri așa cum se prezintă în Figura 3-1. Astfel, în metodologia de proiectare a unui DL trebuie să se țină cont de toate aceste procese care implică utilizarea de resurse hardware și software.

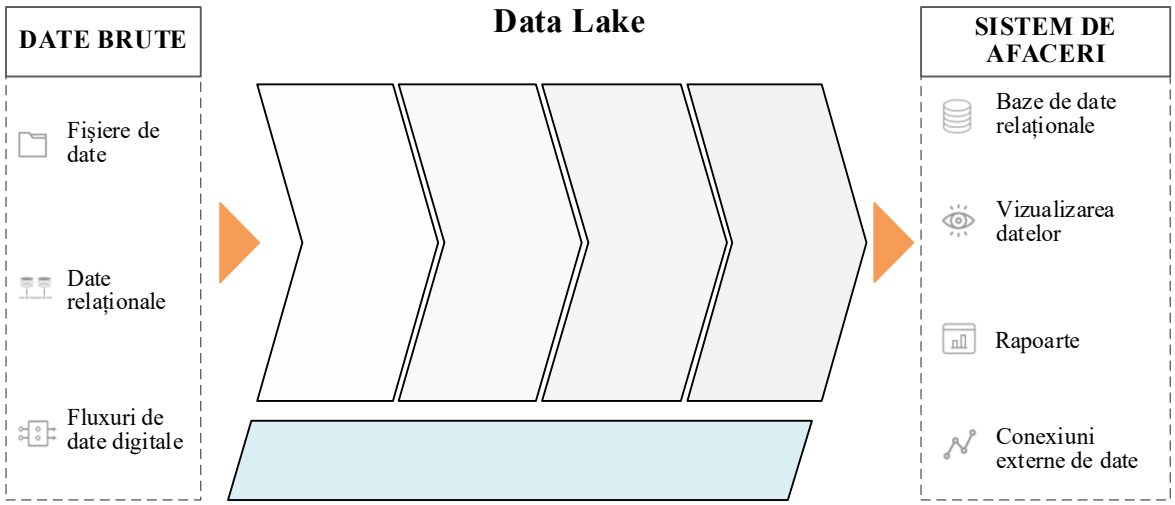


Figura 3-1: Procesele Data Lake [47]

În funcție de mediul de stocare ales pentru implementarea unui DL putem distinge patru tipuri de arhitecturi și anume:

- On-Premise;
- Cloud;
- Multi–Cloud;
- Hybrid Data Lake.

Fiecare din cele patru metode de implementare prezintă o serie de avantaje dar și riscuri, și fiecare implică resurse hardware și software diferite care vor determina costuri diferite de implementare.

3.1 Data Lake on-premises

Data Lake *on-premises* este o arhitectură implementată fizic la nivel local în cadrul unei organizații. Această implementare necesită gestionarea în mod autonom, pe plan local, a tot ceea ce implică construirea unui lac de date. Va trebui să se țină seama de cerințele pentru spațiul fizic și consumul de energie, pașii de proiectare a întregii arhitecturi, dispozitivele hardware și software necesare, licențele software, administrarea acestora și de costurile continue ce implică gestionarea acestora pe plan local. Este necesară o monitorizare constantă a resurselor hardware pentru ca datele ingerate să nu atingă limita disponibilă de stocare și să ducă la probleme de pierdere de date sau erori de funcționare. Aceste sarcini complexe implică resurse inginerești și experți în domeniu care să le gestioneze și care să fie mereu la curent cu noile tehnologii. Astfel, este necesar un efort financiar constant pentru deținerea unei echipe de ingineri care să ofere garanția unui serviciu de calitate. Pentru acest tip de implementare nu se poate avea o idee exactă asupra costurilor totale și există riscul major de a supraevalua sau subevalua aceste costuri. Un alt risc este acela că, odată implementată arhitectura, aceasta să nu fie gestionată corect datorită lipsei de experiență și atunci lacul de date poate să devină neutilizabil și să se transforme într-o mlaștină de date [49].

Timpii mari de proiectare și implementare a unei arhitecturi DL la nivel local nu sunt nici ei de neglijat. Acest tip de arhitectură este recomandat atunci când este nevoie de autonomie deplină asupra datelor. Eforturile financiare pentru implementarea și gestionarea pe plan local [50] a unui DL performant sunt ridicate și nu oricine poate susține aceste costuri.

Tehnologia cea mai des întâlnită pentru implementarea unei arhitecturi Data Lake *on-premises* este oferită de Apache prin framework-ul open-source Hadoop, care este abordat în capitolul 4 în cadrul acestei teze de cercetare. Chiar și cei mai mari furnizori de servicii cloud au apelat la framework-ul Hadoop pentru a implementa sistemul de stocare a datelor Big Data pe propriile platforme.

3.2 Cloud Data Lake

Cloud Data Lake este o arhitectură implementată în cloud. Cei mai mari furnizori de servicii cloud s-au adaptat noilor cerințe din domeniul Big Data și permit astfel implementarea cu succes a arhitecturilor DL. În acest caz, o organizație este degrevată de costurile ce implică stocarea datelor pe plan local, și anume costuri pentru spațiul fizic, pentru energia consumată, pentru dispozitivele hardware și resurse software necesare, pentru echipa de ingineri care să se ocupe de configurarea și gestionarea constantă a acestora, pentru timpii de proiectare și implementare. De asemenea se elimină implicit și riscul de evaluare greșită a costurilor.

Avantajele majore [51] în acest caz sunt date de disponibilitatea imediată pentru crearea unui lac de date, de scalabilitatea hardware automată și de costurile reduse. Un alt avantaj al implementării unui DL în cloud este dat de faptul că se plătesc doar serviciile utilizate fără a exista riscul de estimare greșită a costurilor de implementare fizică.

În cadrul unei tendințe conexe, multe organizații migrează în prezent către platformele cloud pentru funcționalitatea, elasticitatea, administrarea facilă și controlul costurilor. În momentul de față există posibilitatea de a alege între diferiți furnizori de servicii cloud, care urmăresc în mod constant să-și îmbunătățească serviciile oferite contracost.

3.3 Multi-Cloud Data Lake

Acest tip de arhitectură DL presupune utilizarea și interconectarea de servicii cloud oferite de diferiți furnizori [52]. Astfel, există posibilitatea de a folosi în paralel atât serviciile cloud puse la dispoziție de AWS cât și de Microsoft Azure. Prin menținerea mai multor lacuri de date pe platforme cloud diferite, se urmărește obținerea diverselor beneficii oferite de fiecare platformă în parte. Această alegere necesită abilități tehnice mai complexe de gestionare a acestora și respectiv pentru a realiza comunicația între ele, spre deosebire de o arhitectură cloud DL.

3.4 Hybrid Data Lake

Arhitectura DL hibridă permite menținerea lacurilor de date atât la nivel local cât și în cloud [53]. Avantajul major al acestui tip de arhitectură constă în faptul că datele mai puțin relevante pot fi stocate pe plan local, pentru a reduce costurile de stocare, beneficiind totuși de viteza serviciilor din cloud pentru datele importante. Totuși această arhitectură implică costuri mult mai mari decât în cazul arhitecturii cloud DL, deoarece inginerii IT ar trebui să dețină abilități tehnice cu ambele medii de stocare, pentru a realiza gestionarea și comunicația între ele.

Cea mai răspândită tehnologie de stocare și procesare a volumelor mari de date, fie ele on-premises sau în cloud, este oferită de framework-ul Hadoop [54]. Principiul de funcționare constă în gruparea mai multor sisteme hardware pentru a realiza mult mai rapid stocarea și analiza în paralel a seturilor mari de date.

4 Data Lake on-premises bazat pe framework-ul HADOOP

4.1 Sistemul de stocare HDFS

Google a dat startul către o nouă tehnologie, astfel în 2003 și 2004 a lansat două lucrări academice care prezentau sistemul de fișiere Google File System (GFS) [55] și MapReduce [56]. Google a creat o platformă pe care puteau fi implementate mai multe aplicații de gestionare a datelor. Doug Cutting în același timp începuse să lucreze la noi implementări open-source pe ideile sugerate de Google și prin urmare a luat ființă Hadoop [57]. Hadoop este un framework gratuit și open-source care permite procesarea distribuită a seturilor mari de date în cadrul unei rețele de calculatoare [58], [59]. Astfel, Hadoop a devenit unul dintre cele mai fiabile motoare de procesare cu sistem de stocare distribuit [60]. Framework-ul Hadoop este fiabil, economic, scalabil, flexibil și prezintă **viteză mare de prelucrare.**

Hadoop se bazează pe sistemul de stocare HDFS. Utilizarea sistemului de fișiere distribuite Hadoop permite ca datele să fie distribuite către mai multe noduri, acestea fiind astfel citite în paralel, ceea ce reduce cu mult timpii de citire. Hadoop funcționează după principiul "scrie o dată și citește de mai multe ori". [61].

Hadoop se bazează pe o arhitectura de tip *master/slave*. Există un master și mai mulți slave, unde *master-ul* gestionează toate activitățile Hadoop iar stațiile *slave* ocupă rolul de stocare a datelor.

Framework-ul Hadoop ilustrat în Figura 4-1 este format din nivelul de stocare **HDFS layer** și nivelul de procesare **MapReduce layer**. Componenta ***Client*** nu face parte din framework-ul Hadoop dar se poate considera ca fiind un gateway de intrare și accesare a datelor.

HDFS reprezintă nivelul de stocare bazat pe o arhitectură de tip master/slave și este reprezentat de:

- **Name node** (master daemon);
- **Data nodes** (slave daemons);
- **Secondary name node**.

MapReduce este nivelul de procesare bazat de asemenea pe o arhitectură de tip master/slave și este reprezentat de:

- **Job trackers (master daemon);**
- **Task trackers (slave daemons).**

Comunicația dintre HDFS și MapReduce este realizată de mașina client. În mașina client vor fi localizate fișierele .jar și fișierele *config*. Nici un *daemon* nu rulează la nivelul acestei mașini, astfel încât rolul *daemon-ului* este de a funcționa ca un server terminal, fiind folosit pentru comunicația cu cluster-ul pe post de mesager [62].

Clusterul Hadoop este, o colecție de diverse componente hardware care lucrează împreună ca o singură unitate. În clusterul Hadoop, există o mulțime de noduri (pot fi computere și servere) care conțin un nod master și mai multe noduri slave. Altfel spus, un

cluster este reprezentat de unul sau mai multe noduri conectate între ele prin rețele de mare viteză, unde un nod poate fi un computer sau o mașină virtuală. Mărimea clusterelor poate varia, în funcție de nevoie, de la câteva la mii de noduri. Acestea sunt organizate în interiorul centrului de date în rafturi (*rack*), conectate între ele prin intermediul comutatoarelor, care pot conține un număr variabil de servere.

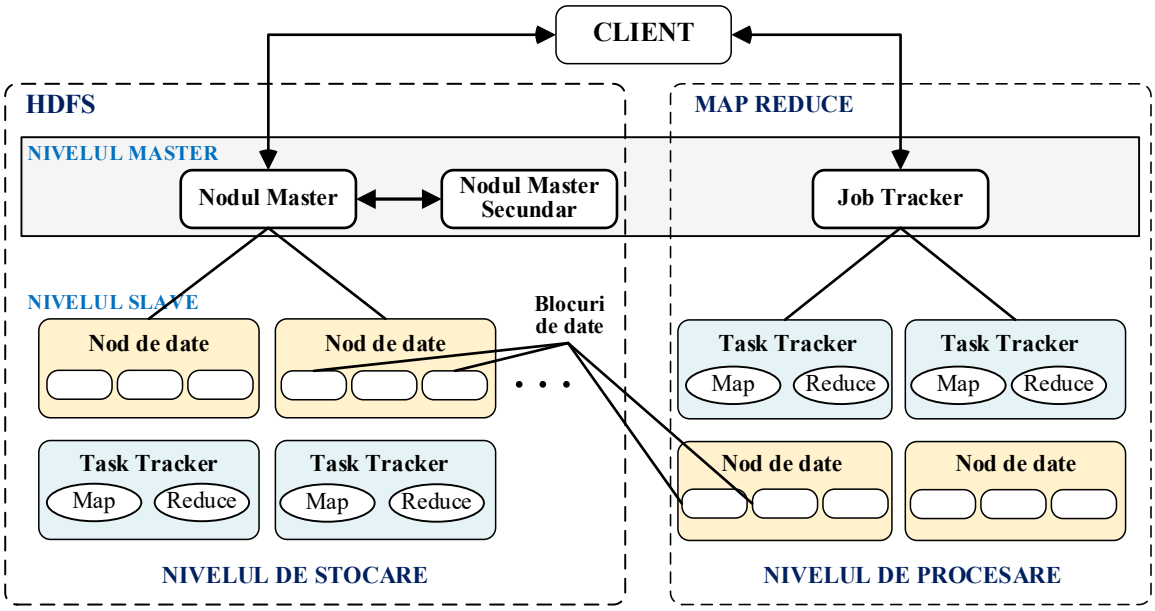


Figura 4-1: Framework-ul Hadoop 1 master/slave [63]

HDFS Layer reprezintă nivelul de stocare a datelor bazat pe o arhitectură de tip master/slave, iar **NameNode (NN)** este daemon-ul principal care întreține și gestionează toate nodurile slave DataNodes. NN este de fapt master-ul din arhitectura master/slave din HDFS, de aceea mai este denumit și **master node** sau nodul master principal.

NameNode stochează informații de tip metadate de sistem și totodată gestionează cererile clienților. În momentul în care se stochează date în HDFS, NameNode va primi toate informațiile de stocare. NameNode este nodul principal al arhitecturii și dacă acest nod eșuează atunci tot sistemul Hadoop va fi indisponibil.

NameNode-ul NN este responsabil pentru următoarele funcții:

- Întreține și gestionează nodurile slave DataNodes (DN);
- Înregistrează metadatele tuturor blocurilor de date stocate în cluster, de exemplu locația blocurilor stocate, dimensiunea fișierelor, permisiuni, ierarhie, etc.;
- Înregistrează fiecare schimbare care are loc la nivelul metadatelor;
- Dacă un fișier este șters în HDFS, NameNode va înregistra imediat acest lucru într-un fișier EditLog;
- Primește în mod regulat un semnal de la toate nodurile de date din cluster pentru a se asigura că acestea sunt funcționale și operative;
- Păstrează o evidență a tuturor blocurilor de date din HDFS și a nodurilor de date în care acestea sunt stocate.

Metadatele sunt seturi de date care furnizează informații cruciale despre fișierele sistem, cum ar fi liste de fișiere și directoare din HDFS, liste de blocuri și locațiile unde acestea sunt stocate, informații legate de drepturile de citire, scriere și execuție asupra datelor, timpii de

acces, etc., astfel încât **metadatele** sunt fișiere log gestionate de **NameNode**. Acestea sunt alcătuite din fișierele ***FSimage*** și ***EditLogs***. ***FSimage*** conțin imaginea fișierelor sistem și sunt salvate la nivelul memoriei RAM, iar ***EditLogs*** dețin toate log-urile generate de NN pe parcursul funcționării sale. Toate acțiunile și modificările care au loc în HDFS vor fi salvate pe hard drive în aceste fișiere EditLogs.

DataNode este daemon-ul slave care rulează individual pe câte o mașină slave. În Hadoop datele sunt stocate sub formă de blocuri, iar aceste blocuri vor fi salvate în DataNodes denumite și noduri de date sau noduri slave. Funcțiile principale ale unui **DataNode** sunt:

- Rulează pe fiecare mașină slave;
- Stochează date reale;
- Răspunde cererilor de citire și scriere lansate de clienți;
- Este responsabil de crearea, ștergerea și reproducerea blocurilor de date pe baza deciziilor luate de către nodul master;
- Trimite periodic semnale către NameNode pentru a raporta starea generală a HDFS. În mod implicit, această frecvență este setată la trei secunde.

Odată cu creșterea volumului de date, numărul de DataNode-uri, crește și el pentru a mări capacitatea de stocare [64].

Nodul master poate fi văzut ca și inima întregului sistem care procesează și primește informații/semnale în mod continuu. Aceste semnale informează nodul master de faptul că datele sunt pregătite pentru a putea fi accesate. Toate instrucțiunile care sunt executate la nivelul unui DataNode la un moment dat, vor fi în mod constant trimise către NameNode. Astfel, din punct de vedere fizic, NN are nevoie de cele mai bune resurse hardware deoarece acolo vor fi memorate toate metadatele.

Secondary NameNode (SNN) denumit și nodul master secundar este folosit pentru a crea puncte de referință, cunoscute și ca puncte de restart al nodului master principal. După cum s-a menționat, nodul master deține toate metadatele salvate în memoria principală, el este modulul cel mai important din Hadoop, acesta este și punctul cel mai sensibil. În caz de erori ale NN tot sistemul Hadoop ar fi compromis, de aceea NN este considerat punct unic de eșec. Astfel, a apărut necesitatea realizării unui nod master secundar denumit SNN, dar care nu poate fi considerat o copie fidelă a NN.

MapReduce este sistemul care permite elaborarea și procesarea datelor stocate la nivelul HDFS cu scopul de a extrage informațiile necesare. MapReduce este un framework utilizat de Apache Hadoop destinat aplicațiilor care procesează și analizează în paralel seturi mari de date pe clustere hardware într-un mod scalabil, fiabil și tolerant la erori [65].

MapReduce este nivelul de procesare a datelor bazat de asemenea pe o arhitectură de tip master/slave. JobTracker și TaskTracker sunt cele două procese esențiale implicate în execuția MapReduce din Hadoop 1. Procesul complet de execuție (Map and Reduce) în Hadoop 1 este controlat de două tipuri de entități denumite Job Tracker și Task Trackers. În cadrul acestei arhitecturi de tipul master/slaves vom avea un singur *job tracker* și mai mulți *task tracker*.

Job Tracker acționează la nivelul nodului master și ține evidența asupra task-urilor Task Trackers ce deservesc nodurile de date. Job Tracker permite identificarea datelor în nodurile de date. Job Tracker, primește task-urile de la client și pe baza informațiilor deținute de nodul

master atribuie task-urile bazate pe reguli de proximitate daemon-urilor slave Task Tracker. O astfel de abordare a procesării datelor reduce semnificativ coada de execuție a task-urilor care ar încetini întregul sistem. Procesul Job Tracker este esențial pentru cluster-ul Hadoop în ceea ce privește execuția MapReduce.

Task Tracker efectuează task-urile atribuite de către Job Tracker. Principala sa responsabilitate este să țină evidența tuturor acțiunilor ce se execută la nivelul nodurilor de date, informând Job Trackers de starea acestora. Eșecul Task Tracker nu este considerat a fi fatal. Când un Task Tracker nu mai răspunde la comenzi, Job Tracker va atribui sarcina executată de TaskTracker-ul eșuat unui alt nod.

Salvarea datelor se face în paralel cu operația de citire, iar multiplicarea datelor se face secvențial la nivel fizic [66]. Spre exemplu, să presupunem că avem un fișier de 400MB pe care trebuie să îl salvăm. Dacă au fost salvate doar 250MB de date și un client dorește deja să acceseze aceste date, atunci Hadoop permite citirea lor chiar dacă salvarea întregului fișier nu a fost încă finalizată. O caracteristică extrem de importantă a sistemului **Hadoop** constă în faptul că permite citirea până la ultimul bloc de date scris cu succes, fără a aștepta ca întreg fișierul de date să fie salvat.

4.2 Configurarea și implementarea clusterelor în Hadoop 1

Pentru implementarea clusterelor în Hadoop 1 am folosit un PC cu sistemul de operare Ubuntu Desktop 20.04.1 LTS pe care am instalat Hadoop 1. Pentru crearea mașinilor virtuale am folosit VMware Workstation 16 Pro pe care am instalat sistemul de operare Linux Centos -Centos-6.3, care este cunoscut ca fiind unul dintre cele mai stabile versiuni. Etapele corespunzătoare modului de setare și configurare ale Hadoop 1 au respectat metodologia propusă în [67], [68] și sunt prezentate în Anexa I.

4.2.1 Single-Node Cluster

Single-Node Cluster este o metodă de implementare și setare a tuturor daemon-urilor pe o singură mașină virtuală. Această metodă de setare este în general folosită pentru cercetare și pentru faza de testare sau în medii cu date puține. Totuși, dacă setul de date nu este suficient de mare atunci nu se pot observa avantajele framework-ului Hadoop. După instalarea, configurarea și pornirea proceselor *ssh* ale unui *Single-Node Cluster,* lansând comanda *jps* care este folosită pentru monitorizarea proceselor de la nivelul mașinii virtuale, se poate verifica status-ul tuturor *Hadoop daemons*: **NameNode**, **SecondaryNameNode**, **JobTracker**, **TaskTracker**, **DataNodes** care rulează pe aceeași mașină virtuală pe care am denumit-o **vmDataLake1**.

4.2.2 Multi-Node cluster

Implementarea Hadoop Multi-Node Cluster implică utilizarea a mai mult de o mașină virtuală. După cum se poate observa în Figura 4-2, fiecare nod de date rulează practic pe o mașină virtuală diferită. Acest tip de implementare se folosește pentru analiza datelor Big Data. În realitate, când sunt procesate date de ordinul PetaBytes este nevoie ca acestea să fie distribuite în sute de mașini pentru a putea fi procesate în timp real. Pentru implementarea

Multi-Node Cluster am clonat și configurat 6 mașini virtuale diferite, fiecare dintre acestea au fost create pentru a deservi în mod independent fiecare daemon al arhitecturii.

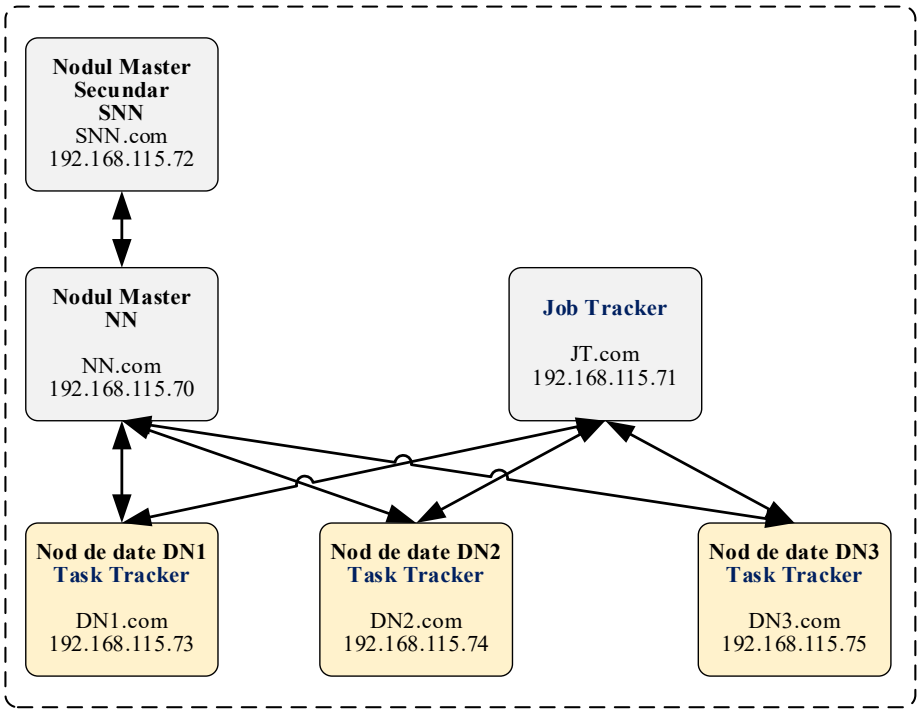


Figura 4-2: Hadoop 1 Multi-Node cluster

NN este mașina virtuală dedicată nodului master principal, JT reprezintă JobTracker și SNN este SecondaryNameNode. Pentru DataNodes sunt create trei mașini virtuale și anume: DN1, DN2 și DN3. În acest caz daemon-urile rulează pe câte o VM dedicată. Prin lansarea în execuție a comenzii /*hadoop dfsadmin -report* ca super-utilizator *hduser* în mașina virtuală corespunzătoare NN se poate verifica dacă totul funcționează corect și dacă nu sunt erori. Aceasta este una dintre comenzile utilizate frecvent ca și administrator pentru a obține un raport complet asupra nodurilor de date [69].

După Hadoop 1 au apărut alte versiuni care au adus îmbunătățiri în ceea ce privește remedierea erorilor și reducerea spațiului de stocare, fără a influența integritatea datelor [70]. Prima versiune de Hadoop, adică Hadoop 1, a fost lansată de fundația software Apache în 2011 [71]. Ca urmare a limitărilor în etapa de procesare și scalare a datelor, ulterior versiunii Hadoop 1 au apărut Hadoop 2 și Hadoop 3. Astfel a fost introdusă, odată cu versiunea Hadoop 2, o nouă componentă denumită Hadoop YARN. Framework-ul Hadoop 2 are patru componente de bază, și anume: *Hadoop common*, care este o bibliotecă comună utilizată de celelalte module, *sistemul de fișiere distribuite HDFS* care este utilizat pentru a stoca cantități uriașe de date cu viteză mare de accesare, *framework-ul Hadoop YARN* pentru planificarea task-urilor și *framework-ul Hadoop MapReduce* pentru procesarea paralelă a seturilor mari de date [72]. Dacă Hadoop 1 conține doar două componente, HDFS și MapReduce, începând cu versiunea Hadoop 2 există trei componente, și anume HDFS, MapReduce și YARN [73]. YARN asigură o gestionare mai performantă a resurselor și reprezintă un suport stabil pentru alte framework-uri de procesare a datelor. În Hadoop 1 MapReduce se ocupă atât de procesarea batch cât și de gestionarea clusterelor, dar începând cu versiunea 2, YARN preia

funcția de gestionare a clusterelor. Dacă Hadoop 1 este limitat la 4000 de noduri per cluster, în Hadoop 2 sunt permise mai mult de 10000 de noduri per cluster. O altă diferență este dată de faptul că Hadoop 1 are doar un singur NameNode care să gestioneze metadatele, pe când Hadoop 2 are un al doilea NameNode aflat în standby și care în caz de eșec permite recuperarea automată.

În Hadoop 2, toleranța la erori poate fi gestionată prin replicarea datelor, care implică un consum foarte mare de spațiu de stocare, în timp ce în Hadoop 3 această situație poate fi gestionată prin *erasure coding*. În Hadoop 2 spațiul de depozitare pentru replicarea datelor este de 200%, iar în Hadoop 3 este de doar 50%. Dacă în Hadoop 2, șase blocuri ocupă de fapt 18 blocuri datorită factorului de replicare, în Hadoop 3, șase blocuri vor ocupa doar nouă blocuri de date din care șase sunt ocupate de datele efective iar trei blocuri sunt pentru procesele de paritate folosite de tehnica *erasure coding*. Totodată Hadoop 3 permite comunicația cu Microsoft Azure Data Lake Storage. Dacă Hadoop 1 putea fi instalat doar pe un sistem Linux, începând cu versiunea 2 acesta se poate instala și pe Windows [74]. Totuși, în implementările industriale nu este recomandat folosirea sistemului de operare Windows pentru a rula framework-ul Hadoop.

5 Data Lake în cloud

Până în 2019 termenul Data Lake, era asociat cu Apache Hadoop, datele organizaționale erau salvate în mod fizic pe plan local la nivelul clusterelor. Din păcate, multe dintre aceste proiecte cu lacuri de date implementate local, au eșuat datorită complexității de implementare a unei astfel de arhitecturi. Gestionarea unui DL pe plan local implică o serie de sarcini consumatoare de timp, executate manual, complicate și implicit costisitoare cum ar fi proiectarea, implementarea și configurarea întregii arhitecturi, definirea și monitorizarea diferitelor procese din cadrul arhitecturii, încărcarea datelor din surse variate, monitorizarea fluxurilor de date, configurarea sistemelor hardware pentru scalare, activarea și gestionarea cheilor de acces la date, reorganizarea datelor în coloane, gestionarea datelor redundante, etc. Adoptarea serviciilor cloud a devenit o tendință incontestabilă în ultimul deceniu [75]. Multe organizații cu volume mari de date, gestionate în regim propriu pe platforme Hadoop, tind să migreze către cloud datorită multiplelor avantaje pe care acestea le oferă.

5.1 Data Lake: de la on-premises la cloud

Majoritatea lacurilor de date implementate pe plan local nu pot gestiona în mod eficient toate seturile de date din cadrul unei organizații. Fără o calitate adecvată a datelor și o administrare corectă a acestora, chiar și lacurile de date bine construite local pot deveni rapid date neorganizate care sunt dificil de utilizat, înțeles și partajat cu diverși utilizatori. Astfel, cu cât este mai mare cantitatea și varietatea datelor această problemă devine mai semnificativă. Alte probleme apărute în lacurile de date locale includ performanțe slabe și dificultăți în gestionare și scalare.

Noul trend este de a implementa, sau chiar muta, lacurile de date în cloud [76], deoarece furnizorii de cloud oferă mai multe servicii integrate și interoperabile care facilitează următoarele operații:

- Colectarea, ingerarea, catalogarea și guvernarea datelor;
- Crearea, securizarea și gestionarea lacurilor de date;
- Analizarea și interogarea datelor.

Platformele cloud au pus mare accent pe partea de securitate a datelor, astfel au dezvoltat mecanisme performante și flexibile de protecție privind accesul la date, de criptare a acestora, de redundanță și de control la nivel de rețea. Cloud DL poate fi văzut ca o platformă unică de stocare care facilitează ingestia, procesarea și vizualizarea datelor, care acceptă cele mai comune framework-uri de analiză.

5.2 Avantajele și dezavantajele unui DL în cloud

Mutarea datelor în Cloud a devenit o tehnică accesibilă unei game largi de organizații, care pot beneficia de următoarele avantaje:

- **Capacitate de stocare:** În cloud spațiul de stocare este nelimitat. Se elimină astfel problemele care pot apărea atunci când se urmărește extinderea și întreținerea unei rețele locale de stocare a datelor.

- **Costuri eficiente:** Furnizorii de servicii de stocare în cloud oferă servicii la costuri diferite, oferind astfel organizațiilor posibilitatea de a plăti exact serviciile de care au nevoie în diferitele etape de dezvoltare, spre deosebire de implementarea locală care presupune analiza și estimarea aproximativă a costurilor având în vedere perspectiva de dezvoltare ulterioară a acestora.
- **Depozit central:** Stocarea în cloud prezintă avantajul unei locații centralizate de stocare a tuturor tipurilor de date și de accesare a acestora din orice locație. Acest lucru simplifică cu mult complexitatea operațiilor, echipa de ingineri fiind degrevată de responsabilitățile care ar implica implementarea acestor servicii de stocare pe plan local.
- **Securitatea datelor:** Toate organizațiile au responsabilitatea de a proteja datele pe care le dețin. Cu lacurile de date concepute pentru a stoca toate tipurile de date, inclusiv informații precum date financiare sau date sensibile legate de clienții unei organizații, securitatea devine astfel prioritară și fundamentală în stocarea datelor. Furnizorii de servicii cloud garantează securitatea datelor conform modelului de securitate S*hared Security Responsability*.
- **Scalare automată:** Serviciile cloud moderne sunt concepute pentru a oferi scalabilitatea imediată a datelor, astfel încât organizațiile nu trebuie să se îngrijoreze de extinderea capacității hardware atunci când este necesar.
- **Interfață prietenoasă:** Un alt avantaj al serviciilor cloud este interfața prietenoasă pusă la dispoziția utilizatorilor. Acest avantaj nu este unul de neglijat, el facilitând și accesul celor mai puțini experimentați în domeniu pentru a consulta datele și a avea o perspectivă asupra lor.

Migrarea datelor și a infrastructurii în cloud prezintă multiple avantaje care simplifică costurile operaționale în cadrul organizațiilor, cu toate acestea există o serie de provocări, și anume:

- **Costurile în timp:** Furnizorii de cloud percep organizațiilor costuri referitoare la serviciile de stocare în funcție de perioada de stocare și dimensiunea acestora. Pe perioade mari de timp este posibil ca un cloud să fie mai costisitor decât un mediu de stocare local. [77].
- **Analiza datelor:** Principalul beneficiu al înființării unui lac de date constă în analiza datelor brute. Capacitatea de a transforma, organiza și combina surse de date diferite este un avantaj imens adus de lacurile de date, dar necesită o soluție de analiză la fel de robustă. Majoritatea furnizorilor de cloud oferă soluții de analiză avansate dar nu mereu ușor de personalizat [78].
- **Migrarea datelor:** Mutarea datelor în cloud poate întâmpina unele dificultăți, fiind un proces destul de complex care are nevoie de o analiză preliminară pentru alegerea unui furnizor de servicii cloud care să se potrivească nevoilor specifice.

Organizațiile care vor să descopere beneficiile unui DL pot începe cu serviciile oferite de furnizorii de cloud. Pe măsură ce avansează în acest nou trend pot lua decizii asumate asupra costurilor și posibilităților fizice de a investi resurse pentru implementarea unei arhitecturi DL pe plan local în regim propriu. Tabelul 5-1 prezintă caracteristicile principalele a metodelor de implementare a unui DL în cloud sau local.

Tabel 5-1: On-premises Data Lake versus cloud Data Lake

	On-premises Data Lake	Cloud Data Lake
Complexitate	Implementarea pe plan local a unui DL necesită gestionarea în mod autonom atât a resurselor hardware cât și software. De asemenea sunt necesare cunoștințe suplimentare de proiectare, configurare și gestionare a întregii arhitecturi.	Cloud pune la dispoziție servicii de stocare avansate, astfel o organizație este degrevată de dificultatea implementării pe plan local a unui asemenea sistem complex de stocare a datelor.
Timpi de realizare	Sunt necesari timpi de proiectare și implementare deloc de neglijat, care pot fi de ordinul lunilor de zile.	Disponibilitate imediată.
Costuri	Nu se poate avea o idee exactă asupra costurilor totale și există riscul major de a supraevalua sau subevalua aceste costuri. Costurile includ: costuri pentru spațiul fizic de depozitare a serverelor, costuri hardware și licențe software, consumul de energie și costurile pentru deținerea unei echipe IT care să le gestioneze.	Se plătesc doar resursele care se utilizează. Se elimină riscul de evaluare greșită a costurilor. Pe termen lung costurile în cloud pot depăși pe cele on-premises, totul depinde de serviciile accesate.
Scalabilitate	Pentru extinderea spațiului de stocare este necesară achiziționarea și configurarea manuală a dispozitivelor hardware și software. Trebuie monitorizate permanent resursele disponibile.	În cloud scalarea se face în mod automat. Poate apărea necesitatea schimbării abonamentului pentru serviciile de cloud.
Viteză	O arhitectură performantă implementată pe plan local care să lucreze într-o rețea locală este mai rapidă, timpii de răspuns fiind mai mici.	Viteza de răspuns depinde de viteza de transfer a datelor din rețeaua de Internet și de serviciile cloud utilizate.
Fiabilitate	În acest caz disponibilitatea permanentă a datelor cade în responsabilitatea organizației. Este nevoie de resurse hardware de încredere și de o echipă de asistență care poate rezolva problemele zi sau noapte.	Furnizorii de cloud copie datele în mai multe clustere în mod automat pentru a asigura fiabilitatea maximă.
Securitate	Chiar dacă pe plan local, la prima vedere, securitatea datelor pare cea mai sigură soluție, din păcate personalul care are acces la datele sensibile le poate copia și exporta în afara firmei.	În majoritatea cazurilor, soluțiile cloud sunt mai sigure decât soluțiile locale. Furnizorii de cloud acordă mare atenție securității datelor.

Printre cei mai mari furnizori de cloud DL amintim Microsoft Azure, Amazon, Google [79], Oracle, urmați de IBM și alții care se străduiesc să țină pasul cu noile cerințe.

6 Proiectarea și implementarea unei arhitecturi ADLS Gen2

În proiectarea unui DL este necesar să se ia în considerare, după cum am văzut, o serie de factori importanți raportați la complexitatea implementării. Nu există o metodă general valabilă de implementare, pentru fiecare proiect, metoda este unică. Un Data Lake de succes trebuie să țină seama de caracteristicile fiecărui nivel din arhitectură, cum ar fi ingestia, stocarea, transformarea și analiza datelor, precum și de nivelul de pregătire a datelor pentru extragerea de rapoarte [80].

ADLS Gen2, tehnologie pusă la dispoziție de Microsoft, oferă posibilitatea implementării unui DL scalabil, avantajos și productiv pentru analize de date de mari dimensiuni. ADLS Gen2 are la bază mediul de stocare Apache Hadoop și ca instrument de gestionare a datelor Apache YARN. Este o soluție bazată complet pe cloud care nu necesită instalarea unui sistem hardware din partea utilizatorului, se scalează în mod automat și permite ingerarea datelor din surse și la viteze diferite.

În ceea ce privește costurile de implementare, ADLS Gen2 vine cu același preț de stocare ca și stocarea pe obiecte *blob storage*, despre care se știe deja că este cel mai economic sistem de stocare pentru volume mari de date. În ADLS Gen2 se achită doar spațiul de stocare folosit, și nu există conceptul de rezervare a unui anumit spațiu de stocare la un cost fix. Costurile de stocare pot fi optimizate în funcție de necesități prin Data Lifecycle Management (DLM). Trebuie subliniat faptul că, totuși, costurile pentru diferitele operații sunt ușor mai mari în cazul stocării ierarhice [81].

În Figura 6-1 este prezentat un flux al proceselor ce stau la baza unui DL implementat în Cloud Azure. Putem observa că există patru niveluri principale, și anume: nivelul de ingestie, nivelul de stocare și procesare, nivelul de servire și cel de consum.

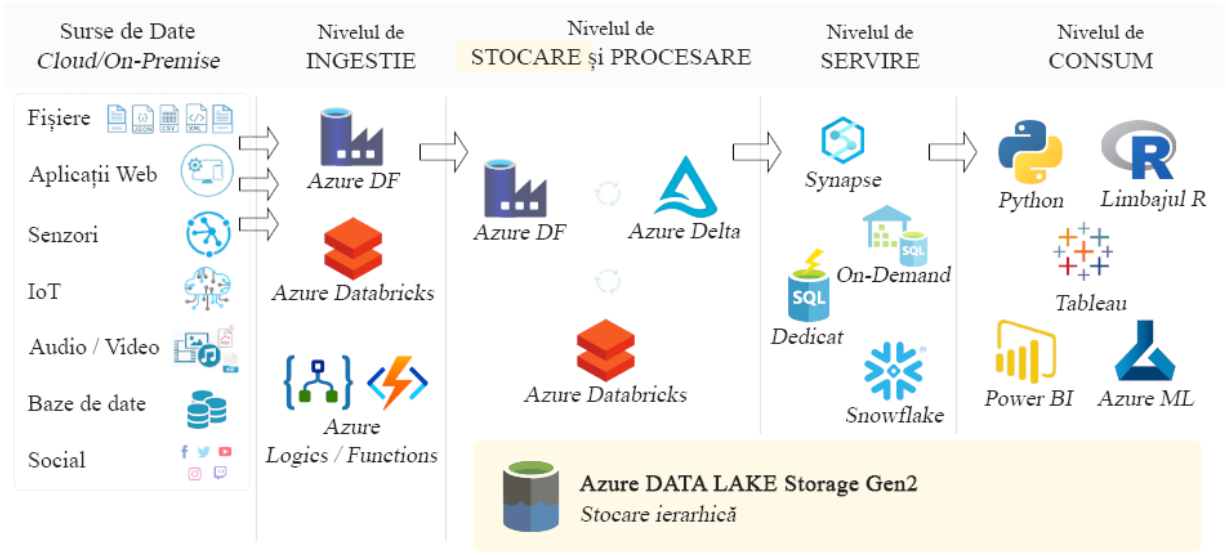


Figura 6-1: Fluxul proceselor în arhitectura ADLS Gen2 (adaptată după [82])

Nivelul de **ingestie** permite încărcarea rapidă în DL a diferite tipuri de date brute provenite de la diferite surse externe. Nu există nici o modificare a datelor în cadrul acestui nivel. Pot fi ingerate date brute în timp real sau în loturi, ulterior putând fi organizate într-o structură logică de dosare. În nivelul de **stocare și procesare** din cadrul arhitecturii DL are loc conversia datelor provenite din nivelul de ingestie într-un format mai structurat și mai potrivit pentru analiză. Aici sunt interpretate datele brute și sunt transformate în seturi de date structurate, care ulterior sunt organizate în directoare și fișiere. În această etapă, datele sunt curățate și standardizate din punct de vedere al formatului și al tipului. Tot aici este implementată logica ce va sta la baza proceselor de analiză avansată. În nivelul de **servire** datele pot fi accesate de utilizatorii autorizați, pentru prelucrarea datelor prin tehnici de analiză avansată cu scopul de a obține informații și rapoarte care vor fi distribuite ulterior către nivelul de **consum** pentru utilizatorii finali. Fiecare nivel în parte este deservit de o serie de procese ce pot fi implementate cu ajutorul diferitelor tehnologii built-in puse la dispoziție de Microsoft.

În cadrul acestei teze de cercetare am implementat un Data Lake cu stocare ierarhică bazat pe tehnologia Cloud ADLS Gen2. Pentru ingestia datelor am apelat la Azure CLI ce permite executarea de comenzi serverless cross-platform prin intermediul unui terminal care se poate conecta la DL. Pentru transformarea datelor am realizat o funcție *Blob Trigger* în Python pe care am integrat-o cu ajutorul Azure Function, care de asemenea este o soluție serverless ce permite rularea de cod pe serverele cloud la costuri reduse. Funcțiile Azure pot fi scrise în mai multe limbaje, cum ar fi C#, Java, JavaScript, TypeScript și Python. Am optat pentru limbajul Python datorită librăriilor ce facilitează procesul de transformare a datelor.

Azure oferă diferite conturi de stocare, fiecare cu propria listă de caracteristici și costuri. Pentru a crea și utiliza un ADLS Gen2 este foarte important tipul de cont, facilitățile și resursele alocate, respectiv caracteristicile de stocare. Pentru a crea un Azure Data Lake Storage Gen2 am selectat tipul **General-purpose V2** care acceptă stocarea pentru obiectele *blob*, *azure file*, *queue* și *table*. În alegerea numelui pentru contul de stocare Azure, trebuie ținut cont de faptul că acesta va face parte din endpoint-ul de acces al serviciului de stocare și poate avea o lungime între 3 și 24 de caractere care pot fi doar numere și litere.

În funcție de tipul de stocare folosit se va folosi următorul endpoint de acces la date:
https://<azure_storage_account>.dfs.core.windows.net.

6.1 Ingestia datelor

Nivelul de ingestie, după cum putem observa în Figura 6-1, este primul nivel din întreaga arhitectură DL. De obicei, ingestia datelor implică doi pași de bază: extragerea datelor din surse și încărcarea lor în DL. În unele situații poate deveni importantă și introducerea unei etape intermediare de transformare a datelor înaintea procesului de încărcare în DL. Varietatea surselor de date precum și a tipurilor acestora complică procesele de ingestie a datelor [83].

Există o mare diversitate de platforme și framework-uri care, nu numai că automatizează încărcarea datelor în DL, dar pot să realizeze și o serie de verificări a calității datelor. Framework-urile de ingestie a datelor pot colecta date din mai multe surse, unde detectează și captează modificările și le reproduc în DL. Majoritatea framework-urilor de ingestie a datelor constau din două componente principale, și anume *Data Collector* și *Data Integrator*. Componenta *Data Collector* colectează datele, în timp ce funcția modulului *Data Integrator*

este de a ingera și integra datele în DL [84]. Un proces de integrare a datelor în DL include citirea datelor dintr-un depozit de date sursă, serializarea/ deserializarea, compresia/ decompresia și cartografierea coloanelor, respectiv scrierea datelor în DL.

Ingestia de date este un proces aproape complet automatizat fiind dedicat transferului de volume mari de date. Ingestia datelor într-un DL poate avea loc în două moduri diferite: fie în timp real, fie în modul *batch*. Procesul de ingestie în timp real este caracterizat de faptul că modificările survenite în datele sursă sunt aplicate DL-ului în timp real. Pe de altă parte, în modul *batch* datele sunt transferate în loturi. Modul batch captează modificările survenite la nivelul sursei și le reproduc în DL la un anumit interval de timp predefinit [84]. Pentru ingestia datelor în DL, Azure pune la dispoziție o serie de instrumente flexibile și performante [85], [86].

6.2 Stocarea și structurarea datelor în ADLS Gen2

Azure Blob Storage este o soluție de stocare pe obiecte în cloud care permite stocarea unor cantități masive de date nestructurate, cum ar fi text sau date binare. Se pot stoca cantități mari de date nestructurate într-o singură ierarhie (flat namespace). ADLS Gen2 vine cu un mecanism nou ce permite structurare ierarhică, care permite organizarea datelor blob în directoare și subdirectoare, stochează metadate despre fiecare director/subdirector și despre fișierele stocate în ele. Această combinație între stocarea blob și mecanismul de structurare ierarhic măresc performanțele corespunzătoare diferitelor procese de stocare și de analiză.

ADLS Gen2 este o metodă de depozitare ierarhică ce permite ca datele să fie plasate în containere, directoare și subdirectoare pentru o organizare mai bună a diferitelor zone din DL. ADLS Gen2 profită de avantajele sistemului de fișiere ierarhic fără a renunța la scalabilitatea și rentabilitatea pusă la dispoziție de stocarea pe obiecte *blob storage* [87]. Având la bază performanțele și securitatea operației de stocare prezentate anterior, în foarte multe cazuri, organizațiile apelează la ADLS Gen2 pentru a profita de avantajele sistemului de fișiere ierarhic. Există o serie de diferențe importante între stocarea pe obiecte și stocarea ierarhică în ceea ce privește performanța și securitatea, printre care putem enumera:

- Performanța interogărilor;
- Performanță în încărcarea și mutarea datelor;.
- Consistența datelor și operațiile atomice;
- Securitate granulară la nivelul directoarelor și/sau a fișierelor;
- Acces multi-protocol.

După cum bine știm, un DL acceptă date de diverse tipuri și formate (date structurate, semi-structurate și nestructurate) din surse eterogene și furnizează datele unei varietăți de utilizatori (*Data engineers*, *Data scientists*, *Data analysts*, etc.) în scopuri diferite, promovând astfel reutilizarea datelor [88]. O organizare corectă a datelor în DL va permite și o gestionare corectă a datelor din punct de vedere al accesului la acestea. Lacurile de date permit o organizare mai flexibilă a seturilor de date, astfel analiștii au posibilitatea de a alege setul de date potrivit pentru analiza lor [89]. Structurarea datelor într-un DL diferă de la caz la caz și de cele mai multe ori nu poate fi prevăzută de la început. Este recomandată implementarea mai multor zone în structura de bază (Figura 6-2), astfel încât aceasta să permită o extindere și o modelare ulterioară ușoară [90].

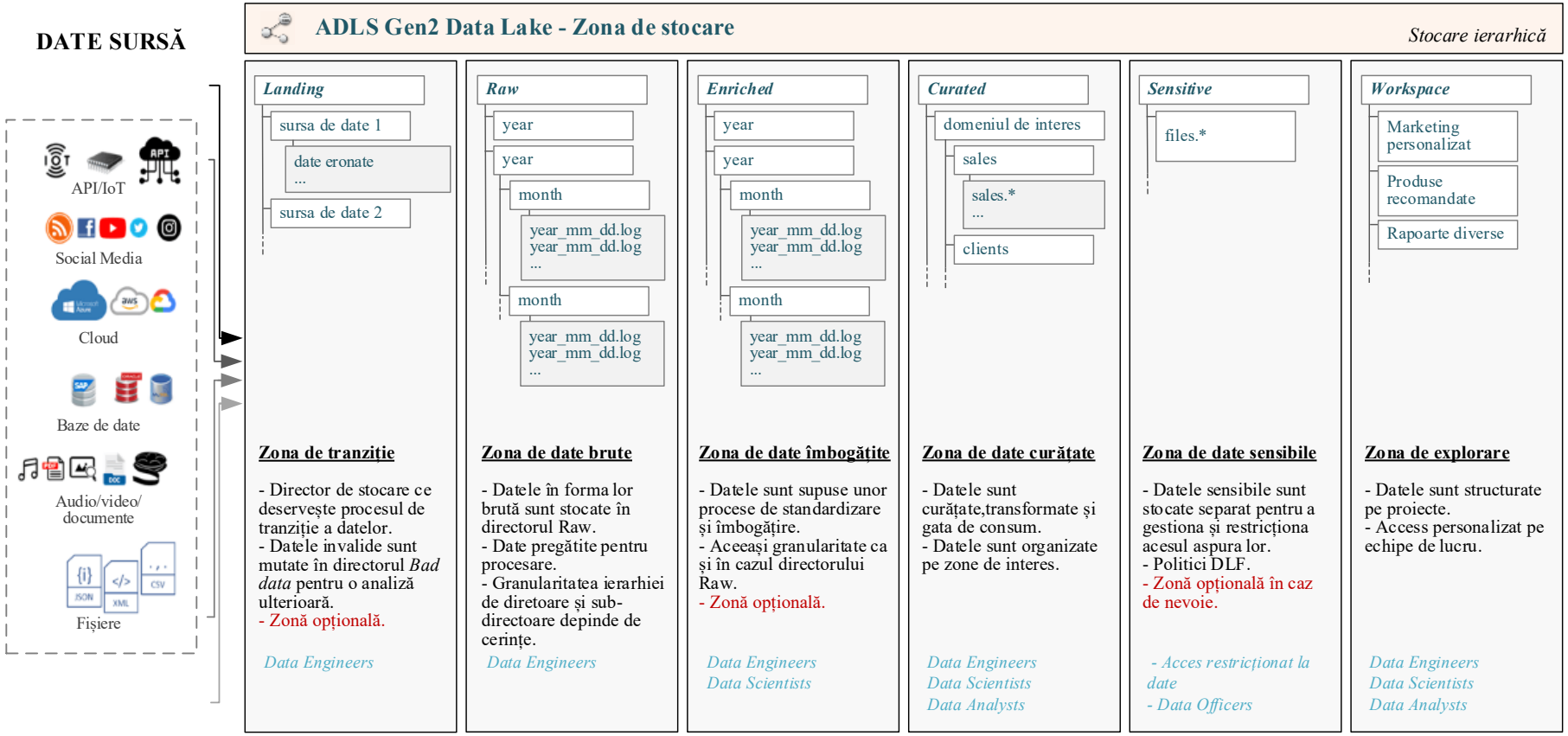


Figura 6-2: ADLS Gen2 - Zone de stocare a datelor în Data Lake

Zonele din arhitectura de bază pentru structurarea datelor în ADLS Gen2 sunt arii partajate ce delimitează datele în procesul lor de transformare și permit acces personalizat pentru asigurarea securității acestora. Un model general de structurare a datelor cuprinde următoarele zone de date:

- **Landing Zone:** Această zonă este opțională și de tranziție care permite trecerea datelor de la sursă în Data Lake. Este o zonă în care se pot distinge datele valide de cele invalide. Datele valide sunt procesate automat în zona *Raw*, iar celelalte vor fi stocate într-o zonă de carantină pentru verificări manuale ulterioare.
- **Raw Zone:** Această zonă este locul unde sunt stocate datele în forma lor brută. Zona este una cu acces restricționat, fiind de cele mai multe ori utilizată de motoare de analiză, cum ar fi Spark, care efectuează operații de curățare și îmbogățire a datelor brute pentru a genera datele de calitate superioară din *Enriched Zone*. Acest nivel de date este controlat de către *Data engineers* și rareori se oferă acces altor consumatori. Zona poate fi organizată folosind un director asociat fiecărei surse, fiecare proces de ingestie având acces de scriere doar în directorul asociat. Structura directoarelor este dată în general de frecvența cu care sunt transferate sau se pot configura datele în funcție de necesitate, astfel:
  - *Data Source > Year > Month > Day > Hour*
  - *Data Source > Entity > Year > Month > Day*
- **Enriched, Cleansed** sau **Standardized zone:** Această zonă stochează datele supuse unui proces grosier de curățare și de îmbunătățire, un așa numit proces de standardizare a datelor pregătite pentru zona următoare denumită **Curated zone**. Principalele procese de prelucrare a datelor care pot avea loc aici sunt cele pentru definirea tipurilor de date (de exemplu fișiere log în format .parquet, etc.), de ștergere a datelor inutile, sau de aplicarea unor reguli de curățare a datelor. Tot în această zonă pot avea loc procese de îmbunătățire a datelor prin combinarea seturilor de date cu scopul de a obține informații mai valoroase. Structura datelor din această zonă este similară cu cea din zona *raw*. Această zonă poate fi considerată ca fiind o zonă de staționare asupra căreia sunt asociate permisiuni de scriere corespunzătoare anumitor task-uri automate sau pot fi atribuite permisiuni de citire pentru *Data scientists*.
- **Curated zone:** Este zona cu date valoroase, care au fost supuse proceselor de transformare, curățare și îmbogățire pentru a putea fi gata de consum. Resursele de date situate pe acest nivel sunt foarte bine guvernate și documentate. Această zonă poate fi accesată de *Data analysts*, *Data scientists* dar și de *Data engineers* pentru procese avansate de analiză, interogări ad-hoc, și nu numai. Structura datelor existente aici va fi mai degrabă personalizată în funcție de ariile de interes, ca de exemplu:
  Subject Areas > Year > Month
  Subject Areas > Region
  Subject Areas > Sales
- **Sensitive zone:** Este zona cu date sensibile care este caracterizată de permisiuni de acces mai rigide. Această zonă permite definirea unei politici separate de gestionare a ciclului de viață a datelor folosind reguli de tip *prefix matching*. Este o zonă opțională, deoarece nu este necesară în toate situațiile. Structura directoarelor este dată în funcție

de necesitate și în mare parte depinde de tipul regulilor pentru acces la datele din această zonă.

- **Sandbox, Laboratory** sau **Workspace zone:** Aceasta este zona de explorare și experimentare accesată de *Data scientist, Data engineers*, *Data analysts*, care înțeleg datele și nevoile organizației. Aici fiecare are posibilitatea de a crea propriile modele și seturi de date cu scopul final de a obține informații importante pentru analiză. Datele sunt structurate pe proiecte, astfel încât se poate acorda acces personalizat pe echipe de lucru.

În Tabelul 6-1 sunt evidențiate caracteristicile cele mai importante pentru structurarea și gestionarea datelor ce aparțin diferitelor zone din DL.

Tabel 6-1: Caracteristici de bază pentru organizarea și gestionarea datelor în DL

	Zona de date brute (*Raw Zone*)	**Zona de date standardizate / îmbogățite** (*Enriched Zone*)	**Zona de date curățate** (*Curated Zone*)	**Zona de explorare** *(Sandbox Zone)*
Structura datelor	Structura fișierelor este dată de formatul setat în faza de ingestie. Exemplu: *Raw/Data Source/Year/Month/ Day/Hour*	Structura datelor este similară cu cea din zona *raw*. Exemplu: *Enriched/Data Source/Year/Month/ Day/Hour*	Structura datelor va fi mai degrabă personalizată în funcție de ariile de interes. Exemplu: *Curated/ Subject Areas/ Files/ Region*	Datele sunt structurate pe proiecte sau echipe de lucru. Exemplu: *Sandbox/sales*
Permisiunile de acces asupra datelor	Blocat, cu acces pentru *Data engineers.*	Accesibil pentru *Data engineers*, acces de citire pentru *Data scientists* .	Accesibil pentru *Data engineers*, acces citire/scriere pentru *Data scientists* și *Data analysts.*	Accesibil pentru *Data engineers*, *Data scientists* și *Data analysts.*
Ciclul de viață al datelor	Odată ce datele au fost procesate și transportate în zona *Enriched* pot fi mutate pe un *cooler tier (nivel mai rece)* cu scopul de a micșora costurile.	Datele sunt menținute în *hot tier (nivel fierbinte)*. Datele mai vechi și nefolosite pot fi mutate pe un *cooler tier.*	Datele sunt menținute în *hot tier*. Este recomandată prezența unor procese și politici de curățare a datelor, pentru îndepărtarea celor care nu sunt necesare.	Consumatorii finali dețin controlul asupra acestui spațiu de lucru.

Atunci când se decide organizarea datelor în DL trebuie luată în considerare atât semantica datelor cât și tipul de consumatori care accesează datele, după cum a fost specificat anterior. Astfel, poate fi eliminată una dintre aceste zone sau, dacă este nevoie, se pot crea noi zone pentru o mai bună gestionare. De asemenea, nu toate zonele trebuie să se afle fizic în același DL [91], [92] astfel încât pot fi implementate pe medii de stocare diferite.

Implementarea unui DL se adaptează nevoilor unei organizații în funcție de sursa datelor, de frecvența cu care sunt generate, de cerințele de organizare și accesare a acestora, și nu

numai. Cea mai utilizată ierarhie într-un DL este alcătuită din zonele *raw*, *enriched*, *curated* și *sandbox*.

Datele stocate în DL pot crește exponențial, de aceea este important ca acestea să fie organizate în funcție de frecvența cu care sunt accesate și de perioadele de stocare. Azure pune la dispoziție diferite tipuri de stocare blob cu costuri diferite. Alegerea tipului de stocare va depinde de frecvența cu care datele sunt accesate: *hot tier*, *cool tier* sau *archive tier*. Pe lângă aceste trei tipuri de *tier,* Azure mai pune la dispoziție posibilitatea de a seta un cont Azure ca fiind unul de tipul *Premium blob storage*, optimizat pentru date accesate cu o frecvență ridicată și cu timpi foarte mici de latență. Stocarea de tipul *Premium blob storage* nu poate fi transformată ulterior într-un alt tip de *tier*. Cum, de altfel, nici un cont creat ca fiind *hot* sau *cool* prin activarea contului *Azure Standard* nu poate fi transformat în *Azure Premium*.

Cu politica de management al ciclului de viață a datelor se pot optimiza atât costurile cât și performanțele procesării datelor prin aplicarea următoarelor tehnici:

- Pentru optimizarea performanțelor: Trecerea datelor din *cool tier* în *hot tier* când acestea sunt accesate frecvent.
- Pentru optimizarea costurilor de stocare: Trecerea datelor din *hot* în *cool* când datele nu au fost accesate sau modificate pentru anumite perioade de timp.
- Introducerea de reguli bazate pe DLM care să fie rulate o dată pe zi la nivelul contului de stocare sau la nivelul subseturilor de date pentru optimizări multiple.

În [93] și [94] pot fi regăsite mai multe informații în ceea ce privește regulile bazate pe DLM, și respectiv cum pot fi optimizate costurile prin gestionarea automată a ciclului de viață a datelor. Gestionarea ciclului de viață a datelor în cloud este un proces important care poate fi realizat apelând la politicile de DLM. Azure Storage oferă politici de management bazate pe reguli care pot fi folosite pentru a gestiona corect tipurile de acces a datelor [95].

Un DL poate fi implementat prin realizarea mai multor containere și directoare, astfel încât trebuie acordată o atenție deosebită regulilor de accesare a datelor pentru asigurarea securității acestora.

ADLS Gen2 suportă următoarele **mecanisme de gestionare a accesului la date**:

- Autorizare bazată pe chei partajate (*Shared Key* - SK);
- Semnătură de acces partajat (*Shared Access Signature* - SAS);
- Controlul accesului bazat pe roluri (*Role-Based Access Control* - RBAC);
- Lista de control al accesului (*Access Control List* - ACL) [96].

Azure Storage stochează întotdeauna mai multe copii ale datelor, astfel încât acestea să fie protejate de evenimente nefericite cum ar fi de exemplu defecțiunile hardware, întreruperea alimentării cu energie electrică, întreruperea comunicațiilor de rețea, dezastre naturale, etc. Redundanța asigură disponibilitatea permanentă a datelor iar alegerea unuia sau altuia dintre aceste tipuri pentru crearea unui DL influențează costurile finale. Există mai multe tipuri de redundanță [97], și anume:

- *Locally Redundant Storage* (LRS) realizează trei copii ale datelor într-un singur centru de date din regiunea principală.

- *Zone-Redundant Storage* (ZRS) realizează de asemenea trei replici ale datelor din Azure Storage în mod sincron în trei zone diferite aflate în vecinătatea regiunii principale.
- *Geo-Redundant Storage* (GRS) copiază datele în mod sincron de trei ori într-o singură locație fizică din regiunea principală folosind conceptul LRS, după care copie datele în mod asincron într-o singură locație fizică din regiunea secundară. În regiunea secundară, datele sunt copiate tot de trei ori sincron folosind LRS. Regiunea secundară este situată la mare distanță fizică față de regiunea principală.
- *Geo-zone-redundant storage* (GZRS) copiază datele în mod sincron în trei zone disponibile Azure din regiunea principală folosind conceptul ZRS descris anterior. Apoi, se copiază datele în mod asincron într-o singură locație fizică din regiunea secundară. În regiunea secundară, datele sunt copiate sincron de trei ori folosind LRS.

Pentru un acces de tip citire la regiunea secundară, trebuie configurat contul de stocare cu *Read-access geo-redundant storage (RA-GRS)* respectiv *Read-access geo-zone-redundant storage (RA-GZRS)*.

6.3 Procesarea și analiza datelor

ADLS Gen2 permite implementarea unei game variate de arhitecturi pentru analiza avansată a volumelor mari de date. Prelucrarea datelor constă într-o succesiune de procese de transformare a datelor nestructurate pentru a fi distribuite în diferitele zone. Odată prelucrate, datele sunt pregătite pentru a putea fi analizate și consumate. Microsoft pune la dispoziție o serie de resurse built-in pentru procesarea, analiza și vizualizarea datelor (Databricks, Azure HDInsight, Azure Synapse Analytics, Azure Data Lake Storage query acceleration, Power BI). Pe lângă aceste resurse puternice și stabile puse la dispoziție de Microsoft Azure, se mai pot aplica și procese create ad-hoc în diferitele framework-uri sau limbaje compatibile cu ADLS Gen2.

6.4 Avantajele ADLS Gen2 în implementarea unui DL

În cadrul cercetărilor am apelat la tehnologia ADLS Gen2 pentru implementarea unui DL și respectiv stocarea fișierelor WSAL. Caracteristicile care stau la baza acestei tehnologii sunt următoarele:

- Se bazează pe framework-ul Hadoop și YARN, fiind un mediu compatibil cu Hadoop;
- Structura ierarhică sub formă de directoare și subdirectoare permite acces de înaltă performanță la date comparabil cu capacitatea și prețul stocării pe obiecte;
- Este un mediu ideal pentru analiza datelor Big Data;
- Prezintă cost și performanță optimizate;
- Permite optimizarea costurilor datorită diferitelor tipuri de stocare: *hot, cool, archive și premium*;
- Furnizează un model de securitate cu granulație fină;
- Pune la dispoziție spațiu de stocare nelimitat cu autoscalare;
- Vine cu o interfață grafică ușor de utilizat;
- Permite proiectarea de sisteme hibride.

7 Proiectarea și implementarea unui DL pentru fișiere WSAL în ADLS Gen2

7.1 Fișierele WSAL

Un fișier WSAL este un document text care conține toate activitățile unui server. Acesta este creat și întreținut automat de server și poate oferi o serie de informații importante și valoroase despre evenimente legate de activitățile clienților din mediul online, activitățile serverului și răspunsul acestuia la solicitările clienților, etc. Este important să se găsească tehnici rentabile și ușor accesibile pentru stocarea și analiza acestor fișiere, considerate a fi de un nivel ridicat de confidențialitate, deoarece dețin informații clasificate drept date personale. Multe instrumente de analiză a fișierelor WSAL au fost create pentru a prezice, analiza și monitoriza comportamentul utilizatorilor [98] față de conținutul unui site, iar aceasta este o ramură care primește din ce în ce mai multă atenție datorită informațiilor ce se ascund în aceste date nestructurate generate de servere. Există diverse tehnici de analiză a fișierelor WSAL pe piață [99], [100], dar principala problemă constă în stocarea acestora în forma lor brută pentru a permite executarea în viitor a diverse procese de analiză care să permită extragerea de informații în diferite scopuri. Majoritatea serverelor generează fișiere WSAL care pot fi stocate în două formate diferite: Common Log Format și Combined Log Format.

Fișierele WSAL sunt fișiere cu date brute greu de înțeles și interpretat. Într-o perioadă scurtă de timp pot ocupa mult spațiu de stocare, motiv pentru care acestea sunt șterse automat la intervale de timp prestabilite. De exemplu, unele servere sunt configurate pentru a șterge fișierele log după două săptămâni, altele șterg fișierele după ce depășesc o anumită dimensiune [101]. Se estimează că peste 80% [102] din datele deținute de o organizație se pierd deoarece acestea nu pot ține pasul cu volumul de stocare, viteza cu care se generează datele și varietatea datelor. În interiorul organizațiilor se preconizează că 80% din date sunt nestructurate, astfel între 60% și 73% din toate aceste date nu sunt niciodată analizate [103]. Acestea sunt așa-numitele date auxiliare sau *dark data*, colectate din diverse surse, cum ar fi datele provenite de la senzori, statistici, social media, servicii de monitorizare, fișiere log, date audio și video, etc. [104]. Fișierele WSAL sunt fișiere în care, fiecare linie reprezintă o solicitare făcută către server și oferă o colecție valoroasă de date. Un studiu al conținutului acestor fișiere a condus la evidențierea următoarelor categorii de date [105], [106], [107]:

- Adresa IP și identitatea dispozitivului care a plasat o solicitare server-ului;
- Numele, locația și dimensiunea fișierului solicitat;
- Ora și data solicitării, respectiv metoda de solicitare;
- Pagina de pe care utilizatorul a părăsit site-ul;
- Răspunsul server-ului, HTTP status code;
- Durata și numărul de pagini vizitate de utilizator;
- Tipul motorului de căutare folosit și termenul de căutare introdus;
- Dacă este o accesare directă a unui utilizator sau o redirecționare de pe un alt site.

Toate aceste date pot furniza, în urma analizei, o serie de informații care să ajute la îmbunătățirea securității site-ului web, la îmbunătățirea structurii site-ului, la îmbunătățirea performanțelor server-ului web, la creșterea traficului către site, la depistarea erorilor de pe site, la detectarea atacurilor care pot apărea spontan pe anumite segmente și care, uneori nu pot fi observate imediat și este o unealtă importantă care să ducă la îmbunătățirea Search Engine Optimization (SEO) a unui site web. O analiză a fișierelor log poate oferi informații legate de cum indexează și accesează roboții motoarelor de căutare site-urile, astfel încât să se poată aplica tehnici de optimizare pentru a obține un clasament cât mai bun. Deoarece fișierele log sunt generate în mod automat, cantitatea de date poate deveni rapid foarte mare și este direct raportată la numărul de vizitatori. Pe măsură ce datele log cresc, crește și necesitatea stocării fișierelor pe termen lung, ceea ce obligă organizațiile să găsească o soluție optimă care să permită indexarea și analiza acestora [108]. Astfel, cercetătorii se confruntă cu nevoia de analiză a fișierelor WSAL de mari dimensiuni pentru asigurarea serviciilor de fiabilitate, siguranță și performanță. Pentru site-urile de mici dimensiuni analiza fișierelor log poate fi făcută ușor, fie direct prin citirea lor, fie apelând la diverse servicii on-line sau open-source [109], care oferă un suport solid pentru analiză. Aceste servicii vin însă cu o serie de limitări, fie legate de dimensiunea fișierelor, fie de procesul de analiză, astfel că pentru analiza fișierelor log de mari dimensiuni trebuie găsită o altă abordare. După cum am specificat mai sus, una dintre problemele log-urilor de mari dimensiuni o reprezintă procesul de stocare. Soluția propusă constă în implementarea unui DL care oferă servicii de stocare a datelor în forma lor naturală, cu autoscalare, ce permite stocarea de cantități nelimitate de date.

În urma cercetărilor bibliografice efectuate, am identificat și sintetizat, principalele avantaje ale salvării fișierelor log într-un DL și anume:

- Dat fiind faptul că fișierele WSAL sunt salvate local, pe servere, pentru perioade limitate de timp, utilizarea unui DL conferă flexibilitate maximă în memorarea datelor pe termen lung, cu costuri rezonabile.
- Se păstrează controlul asupra tuturor datelor din fișierele WSAL.
- Instrumentele de analiză web existente, se bazează pe coduri Java Script sau cookie-uri, care pot aduce timpi de întârziere în încărcarea unei pagini și devin mai susceptibile în apariția de diverse probleme tehnice. În cazul în care utilizatorul are blocate în browser-ul personal executarea script-urilor și a cookie-urilor atunci rapoartele obținute din servicii de tip Google Analytics pot să nu fie concludente. Astfel, stocarea în DL a fișierelor log oferă posibilitatea de a implementa procese de analiză dedicate cu scopul de a extrage informații și rapoarte personalizate fără a influența încărcarea unei pagini web și fără a depinde de setările din browser-ul clientului.
- Permite încrucișarea datelor deținute de fișierele log cu alte date existente în cadrul unei organizații, cu scopul de a obține rapoarte specifice diferitelor domenii de interes.

Abilitatea de a examina fișierele log chiar și după perioade mari de timp devine o necesitate. Astfel am propus implementarea unei arhitecturi DL folosind ADLS Gen2 pentru realizarea practică a proceselor ELT asupra fișierelor WSAL. Pentru ingestia datelor am apelat la tehnologia Azure CLI cu scopul de a implementa o secvență de comenzi cross-platform, care în realitate pot fi incluse într-un proces batch ce poate fi executat în background

la intervale regulate de timp. Astfel ingestia fișierelor log în DL se poate face în cadrul unui proces complet autonom. Odată realizată ingestia datelor în DL acestea sunt supuse unui proces de transformare a fișierelor log în fișiere parquet, proces pentru care au fost evidențiate principalele avantaje. Pentru automatizarea acestui proces am implementat o funcție serverless *Azure Function* de tip *Blob Trigger* scrisă în limbajul Python. Pe partea de stocare am propus de asemenea tehnici de DLM la nivelul lacului de date cu scopul de a reduce costurile de stocare.

7.2 Arhitectură DL pentru analiza fișierelor WSAL implementată în ADLS Gen2

Ca rezultat al cercetărilor efectuate pentru implementarea unui DL am propus arhitectura prezentată în Figura 7-1. Pentru realizarea obiectivului tezei au fost vizate cu precădere problemele legate de *nivelul de ingestie* a datelor ce implică doi pași de bază, extragerea datelor și încărcarea lor în DL, și *nivelul de stocare* ce reprezintă nivelul în care datele sunt salvate fizic și organizate ierarhic în diferite zone în DL.

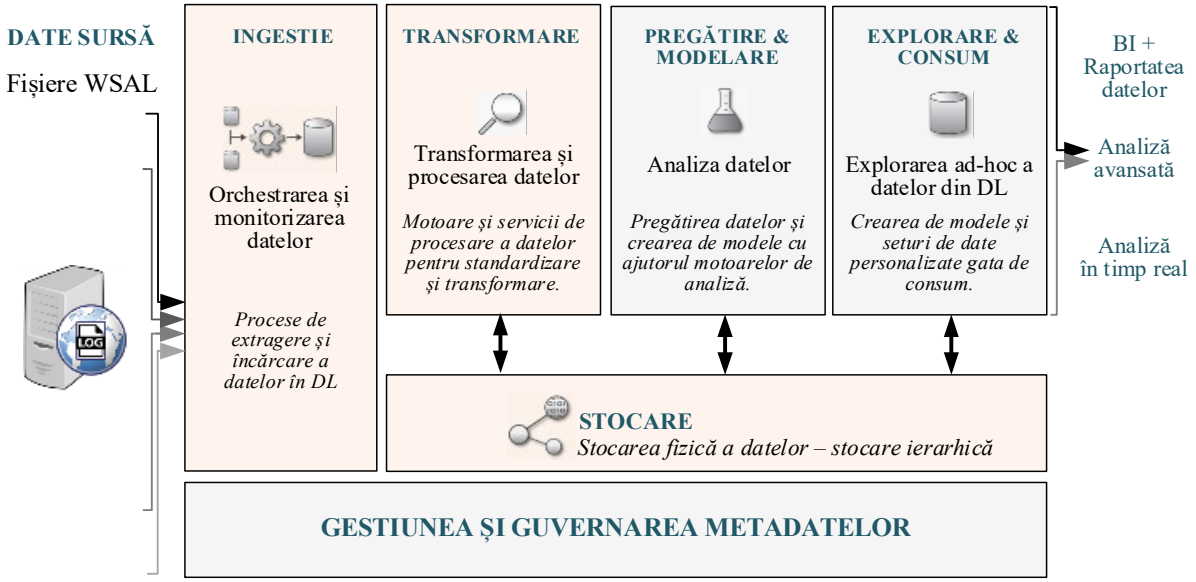


Figura 7-1: Arhitectura Data Lake implementată pentru stocarea fișierelor WSAL

Arhitectura propusă este compusă din cinci niveluri, nivelul central al întregii arhitecturi fiind nivelul de stocare. Peste acest nivel se regăsesc nivelurile Transformare, Pregătire & Modelare și nivelul de Explorare & Consum, care servesc diferitelor etape de transformare și procesare a datelor pentru ca, ulterior, folosind tehnici adecvate de analiză avansată, să poată fi extrase informații valoroase.

Nivelul de *Ingestie*, permite execuția de procese rapide de încărcare a datelor brute provenite din diverse surse externe în DL. În acest nivel nu au loc procese de modificare a datelor. Datele brute pot fi ingerate în timp real sau pe loturi, în mod batch. În urma procesului de ingestie, datele sunt salvate în nivelul de *Stocare* al DL într-o zonă dedicată datelor brute, în forma lor naturală. În nivelul *Transformare*, au loc procese de curățate și standardizate a datelor în tipuri de date adecvate. Datele brute sunt transformate în seturi de date mai structurate, care sunt plasate în directoare și subdirectoare bine organizate în zona de explorare

din cadrul nivelului de stocare. Nivelul *Pregătire & Modelare* al arhitecturii este cel în care datele transformate anterior sunt pregătite și modelate cu ajutorul unor motoare de analiză avansată. În nivelul *Explorare & Consum*, datele sunt accesate de utilizatori care cunosc datele și nevoile unei organizații. Noile procese de modelare sunt realizate cu ajutorul unor tehnici de analiză adecvate pentru a obține informații utile.

Fiecare dintre aceste niveluri este deservit de procese diferite care pot fi implementate fie prin intermediul tehnologiilor integrate furnizate de Microsoft, cum ar fi Azure DF, Azure Databricks, fie prin procese dezvoltate ad-hoc scrise în diferite limbaje de programare, care pot fi integrate cu ușurință prin intermediul Azure Logics sau Azure Functions.

Aceste cercetări se concentrează în principal pe nivelul de *Ingestie*, pe nivelul de *Stocare* și pe procesul de transformare a datelor, în cadrul nivelului de *Transformare*, care conduce la standardizarea și pregătirea datelor pentru procesele succesive de analiză avansată. Această arhitectură reprezintă o soluție fiabilă de stocare a volumelor mari de date de tip WSAL pe perioade mari de timp pentru a putea fi supuse ulterior diferitelor procese de analiză avansată.

7.3 Resurse utilizate

Pentru implementarea unui DL în primă fază este important să se studieze seturile de date care vor defini ulterior resursele necesare pentru implementarea lacului de date.

7.3.1 Seturile de date WSAL

Pentru activitatea de cercetare, datele supuse proceselor ELT sunt date de tip WSAL ce au fost descărcate de pe site-ul universității Harvard secțiunea „Harvard Dataverse”, subsecțiunea „Online Shopping Store - Web Server Logs”. Sunt date provenite de la un website Iranian e-Commerce *zanbil.ir* [110] pe care l-am denumit *access-weblog.log* și care deține 3.0GB de date brute. Datorită dificultăților cu care m-am confruntat în obținerea unui astfel de set, din cauza confidențialității ridicate a acestor tipuri de date, am apelat la seturile de date puse la dispoziție de universitatea Harvard, care oferă date publice în scopuri de cercetare. După o evaluare preliminară a liniilor din log observăm că este vorba de un fișier de tipul *Combined Log Format* [111].

Principala problemă cu WSAL constă în faptul că acestea sunt de obicei date text semi-structurate, ceea ce face dificilă interogarea lor pentru obținerea de orice informații utile. Pentru a realiza o analiză eficientă a acestor fișiere este necesar ca ele să fie supuse unui proces de normalizare/standardizare.

7.3.2 Resurse caracteristice ADLS Gen2

Pentru crearea unui Data Lake în Azure, ca mediu de stocare a datelor brute WSAL, sunt necesare trei resurse care definesc principalele caracteristici ale unui DL:

- *Subscription:* Este un abonament Azure, ce reprezintă o entitate logică care este utilizată pentru a defini partea administrativă și financiară a resurselor Azure utilizate. În funcție de abonamentul selecționat vom avea acces sau limitări la resursele ADLS Gen2.
- *Resource group:* Reprezintă un container logic ce stochează metadate despre resursele necesare pentru ca o soluție Azure să poată fi gestionată împreună ca și grup.

- *Storage account:* Este o resursă Azure care conține toate obiectele de date *Azure Storage*, cum ar fi blob, fișiere, tabele, etc.

Pentru implementarea lacului de date am creat un cont *Azure Blob Storage* cu opțiunea *hierarchical namespace* activă [112]. Pentru resursa *Subscription* s-a creat un abonament *Azure for Students* cu 100$ credit disponibil pentru o perioadă de 12 luni, cu limitări în accesarea gratuită a unor resurse Azure, dar cu suficiente resurse pentru implementarea proiectului și demonstrarea contribuțiilor aduse în cadrul acestei teze de doctorat, care se bazează mai puțin pe resurse Azure built-in și mai mult pe funcții *serverless*. Datele deținute sub acest abonament sunt durabile, disponibile, sigure și scalabile, identificate printr-un *namespace* unic în lume, accesibil de oriunde în lume prin HTTP sau HTTPS. Odată creat acest abonament, în portalul Azure (https://portal.azure.com/), am creat lacul de date cu setările adaptate cerințelor, care sunt prezentate mai jos.

În ceea ce privește *Resource group*, s-a creat un grup de resurse în cadrul DL, care facilitează administrarea serviciilor și aplicațiilor. Se pot crea diferite grupuri de resurse în funcție de necesitate. Pentru cercetările realizate am creat un grup de resurse *rg-adlgen2-webserver2* pentru mediul de stocare ce va deține controlul asupra fișierele WSAL provenite de la *server1*. Ca și locație (*Region*) pentru grupul de resurse este recomandată selectarea celei mai apropiate locații față de punctul de lucru, în acest caz fiind selectată opțiunea West Europe. În grupul de resurse *rg-adlgen2-webserver2* a fost creată o nouă resursă de tip *Storage account* cu numele *adlswebserver2*, conform procedurii descrise pe larg în [112]. În faza de creare este important ca în tab-ul *Advanced* să se bifeze *Enable hierarhical namespace* specific mediului de stocare ierarhic pentru un ADLS Gen2. Ca factor de replicare am ales *LRS* datorită faptului că este un mediu de cercetare și nu unul de producție care ar avea nevoie de un nivel de protecție a datelor mai ridicat.

După crearea acestor resurse necesare în implementarea ADLS Gen2, în portalul Azure se regăsesc resursele disponibile ilustrate în Figura 7-2, unde *adlswebserver2* este mediul de stocare care va deservi lacul de date.

Figura 7-2: Resursele create în Portalul Azure

7.4 Ingestia, stocarea și transformarea fișierelor WSAL

În etapa următoare am implementat procesele ELT în ADLS Gen2. Pe baza considerentelor teoretice, am propus un model de structurare ierarhică ce permite organizarea datelor în cadrul lacului de date. Acest model este un bun suport pentru crearea ierarhiei de directoare și subdirectoare ce corespund diferitelor zone din cadrul nivelului de stocare al DL. Odată delimitate zonele din nivelul de stocare putem trece la implementarea proceselor de ingestie, stocare și transformare.

7.4.1 Nivelul de stocare și structura ierarhică propusă

Nivelul de stocare, este nivelul principal care stă la baza întregii arhitecturi DL. Datele sunt stocate în containerul principal *adlswebserver2*, creat anterior, date ce provin de la sursa *server1*. Pentru datele provenite de la servere diferite este recomandat să se creeze noi containere sub grupuri de resurse dedicate, pentru gestionarea separată și securitatea partajată.

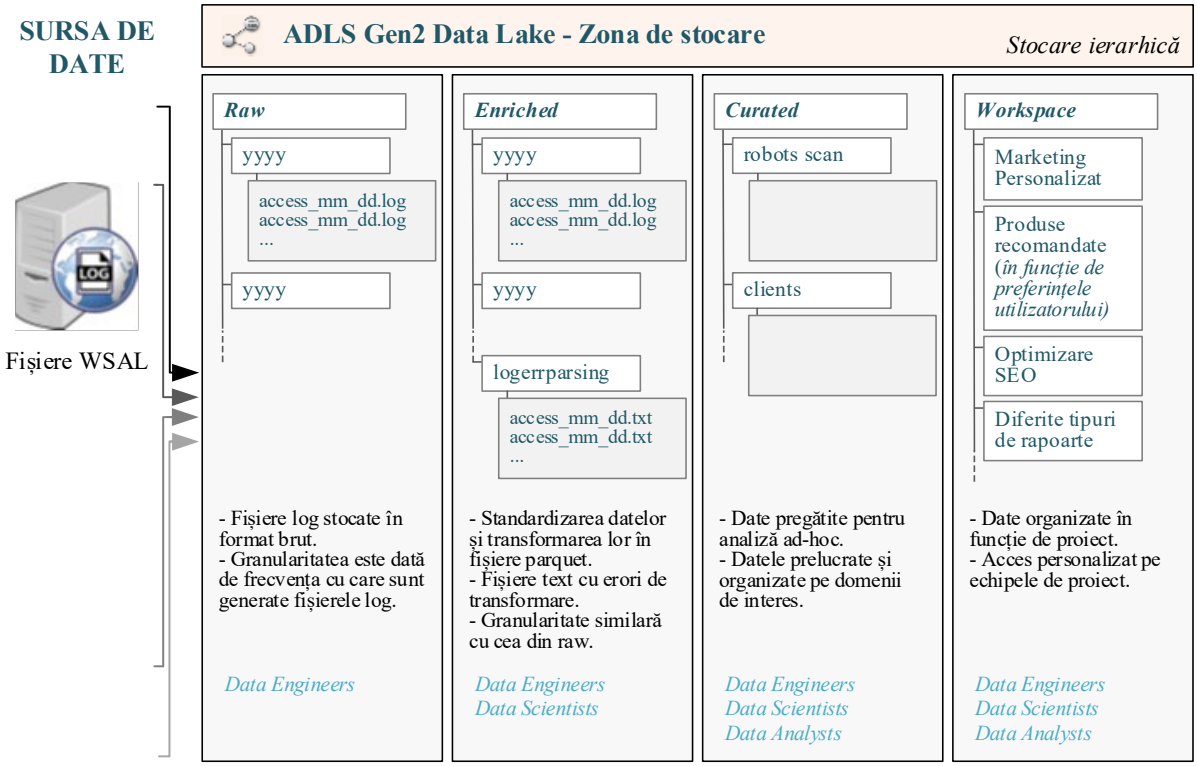


Figura 7-3: Organizarea ierarhică a datelor de tip WSAL în ADLS Gen2

Pe baza informațiilor generale prezentate în capitolul 6, am propus un model de organizare ierarhică a datelor, prezentat în Figura 7-3, care stă la baza întregii arhitecturi DL pentru a susține diferitele procese ELT.

În modelul propus pentru stocarea datelor în DL dispunem de următoarele zone:

- **Zona de date brute (*raw*)**: Este zona unde sunt încărcate datele brute provenite direct de la sursă. Această zonă poate fi organizată folosind o ierarhie de directoare de tipul *yyyy/mm/dd*, în funcție de frecvența cu care sunt generate fișierele log. În acest caz voi opta pentru o ierarhizare grosieră. Fiecare proces de ingestie va avea acces doar de scriere pentru directorul asociat.
- **Zona de date transformate (*enriched*)**: Este zona de date brute provenite din zona *raw* care au fost supuse unui proces de transformare și de standardizare și salvate ulterior în această zonă *enriched* ca fișiere în format *.parquet.* În zona *raw* rămân datele brute. Pentru reducerea costurilor de stocare este recomandată transferarea acestora pe un *cooler tier* cu scopul de a fi arhivate pe termen lung. În zona *enriched* există un folder *logerrparsing* unde vor fi stocate fișierele de tip text cu liniile de eroare rezultate în urma procesului de transformare.

- **Zona de date gata de consum (*curated*)**: Datele stocate în zona *enriched* vor fi supuse diferitelor procese de curățare, ulterior fiind transferate în zona *curated*. Pot fi aplicate pe același set de date, metode de curățare diferite, pentru a servi unor tehnici diferite de analiză în funcție de scopul urmărit. În această zonă datele vor fi regrupate și stocate într-o nouă ierarhie în funcție de domeniile de interes.
- **Zona de explorare a datelor (*workspace*)**: Este zona accesată de cercetătorii de date, inginerii de date, analiștii de date, care înțeleg datele și nevoile organizației. Datele sunt structurate pe proiecte cu acces dedicat pe echipe de lucru.

Datelor organizate ierarhic în directoare și subdirectoare, în funcție de necesitate, le sunt atribuite diferite drepturi de acces pentru a menține integritatea și securitatea acestora.

7.4.2 Ingestia datelor

Procesul de ingestie a datelor bazat pe ADLS Gen2 constă din încărcarea fișierelor log în directorul *raw/*. Pentru testarea procesului de ingestie s-au încărcat fișiere *access-weblogs.log* cu dimensiunea de circa 1.0GB.

Încărcarea fișierelor log *access-weblogs.log* din *local* în cloud DL a fost realizată apelând la tehnica Azure CLI, care este un instrument *cross-platform command-line* ce permite execuția de comenzi administrative asupra resurselor din contul Azure Storage. Pe stația de lucru este necesară instalarea și alocarea următoarelor resurse:

- Mediul de dezvoltare *VS Code*;
- Extensiile VS Code:
  - *Azure CLI Tools extension*;
  - *Python extension*.

Am proiectat un proces de ingestie a datelor pe care l-am implementat printr-o succesiune de comenzi Azure CLI.

```
#conectare la Azure Storage Account
az login

#activarea opțiunii param-persist care stochează setările pentru utilizarea continuă
az config param-persist on

#resursele Azure Storage Account
$adls_acct_name = “adlswebserver2”
$obj_id = “f716d1d0-d716-4fd7-a634-e4476----”
$resource_group_name = “rg-adlgen2-webserver2”

# Asignarea rolului ”Storage ADLS Data Contributor”' pentru utilizatorul adlswebserver2
az role assignment create --assignee $obj_id --role 'Storage Blob Data Contributor' --resource-group
$resource_group_name

# Crearea unui container
$adls_container_name=“webserver2”
az storage fs create -n $adls_container_name --account-name $adls_acct_name --auth-mode login
```

```
# Crearea directorului raw
$adls_directory_name=“raw”
az storage fs directory create -n $adls_directory_name -f $adls_container_name --account-name $adls_container_name --auth-mode login
# Crearea ierarhiei de directoare...
...
#Încărcarea datelor în DL
$adls_acct_name = “adlswebserver2”
$logFileToUpload = “C:\IZA\doctorat\VisualStudio\ADLS\CLI_prj\data\access-weblogs.log”
$adls_dir_name = “webserver2/raw/”
az storage blob upload --account-name $adls_acct_name --container-name $adls_dir_name --name 'access-weblogs.log' --file $logFileToUpload --auth-mode login
```

Prin aceste comenzi s-a realizat autentificarea în contul Azure, crearea ierarhiei de directoare și subdirectoare, respectiv lansarea comenzilor de încărcare a fișierelor log în zona *raw* din ADLS Gen2. Ierarhia de directoare poate fi realizată direct din Azure Storage Explorer. În acest caz datele de acces la portalul Azure trebuie să fie împărtășite direct unei părți terțe, ceea ce nu este o practică recomandată [113]. Etapele implementării procesului de ingestie sunt prezentate mai jos.

Procesul de ingestie poate fi transformat într-un proces batch care să ruleze pe serverul sursă și care să execute funcția de upload/ingestie a datelor la intervale regulate de timp de pe server în DL. În Figura 7-4 sunt prezentate rezultatele obținute în urma procesului de ingestie a datelor în lacul de date.

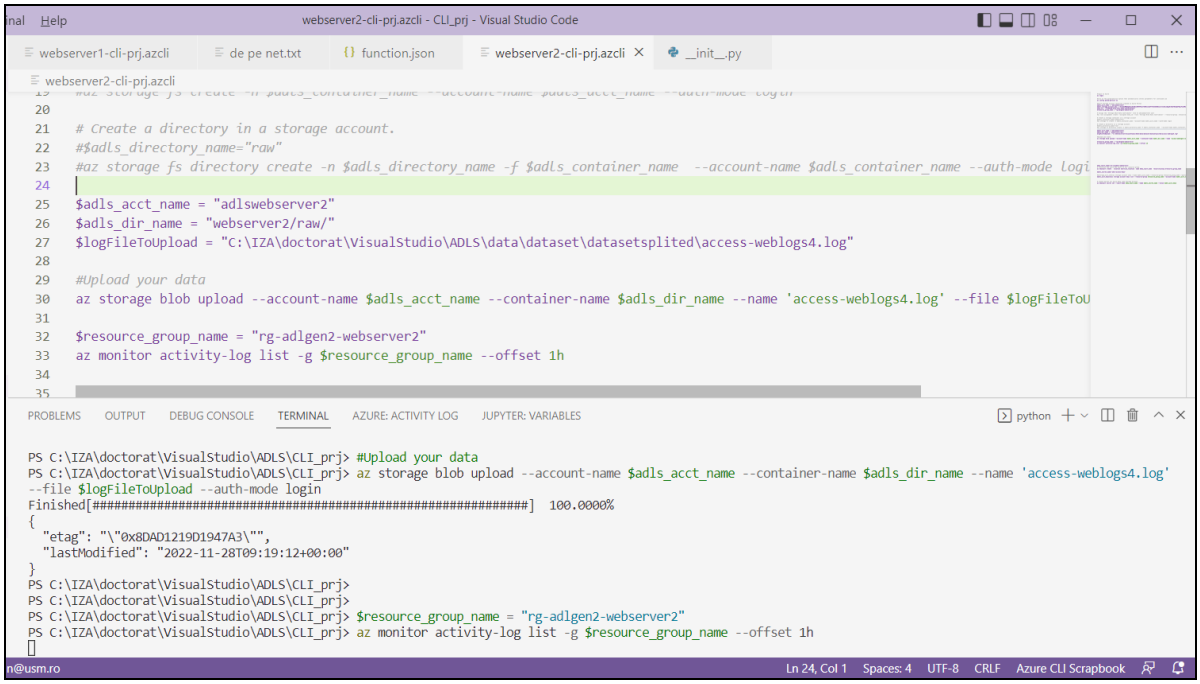


Figura 7-4: Rezultatul implementării procesului de ingestie a datelor în DL

În Figura 7-5 este prezentată zona de date brute *raw* din portalul Azure după ce datele au fost încărcate cu succes.

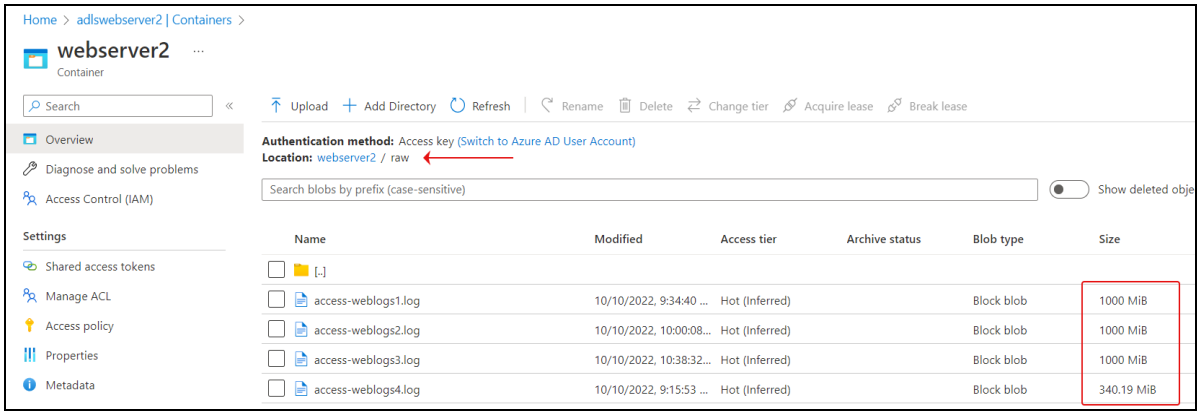


Figura 7-5: Datele brute stocate în zona raw/ din DL

Resursele folosite sunt reduse dar validează propunerea de realizare a unui DL bazat pe ADLS Gen2. Este evident că într-o abordare la nivel de producție, zona *raw* va stoca datele log încărcate de pe un server web la intervale de timp regulate în DL. Astfel, este necesar să se realizeze o ierarhie mai granulară în *raw/* prin crearea de directoare și subdirectoare care să grupeze fișierele WSAL provenite de la server sub forma *yyyy/mm/dd,* în funcție de frecvența cu care sunt generate acestea și respectiv de dimensiunea lor. Pentru încărcarea datelor în DL pot fi utilizate de asemenea diferite tehnologii built-in puse la dispoziție de Microsoft.

7.4.3 Procesarea fișierelor log din zona *curated*

În nivelul de procesare au loc transformări asupra datelor brute pentru a obține date structurate ce pot fi ușor de analizat și interpretat. Fișierele WSAL salvate în DL în zona de date brute *raw/* sunt prelucrate și încărcate în zona de date *enriched/* cu scopul de a fi transformate în fișiere de tip *.parquet*, ce distribuie conținutul în coloane. Datele salvate pot fi citite individual, îmbunătățind astfel semnificativ performanțele sistemului deoarece datele din formă textuală semi-structurată vor fi transformate în date structurate. Prin transformarea fișierelor *.log* în fișiere *.parquet* se realizează totodată o comprimare a datelor ceea ce reduce cu mult dimensiunea inițială a fișierelor și implicit se reduc semnificativ costurile de stocare. Erorile rezultate în urma procesului de transformare a fișierelor log în fișiere *.parquet* sunt încărcate în directorul *enriched/logerrparsing/.* Este recomandat ca fișierele log din directorul *raw/,* care au fost deja procesate cu succes, să fie transferate într-un grup de resurse de tip *archive tier* dacă se dorește în continuare păstrarea lor în forma originală pe perioade mari de timp. Astfel se reduc semnificativ costurile de stocare a datelor brute. Nu este recomandat ca acestea să rămână pe perioade mari de timp în zona *raw* care este de tip *hot tier,* deoarece cresc semnificativ costurile de stocare. Pentru procesul de transformare a fișierelor *.log* în fișiere *.parquet* am proiectat și implementat o funcție serverless *Azure Function* de tip *Blob Trigger* care să permită automatizarea procesului de transformare.

Azure Functions este o extensie pentru Visual Studio Code ce permite realizarea de funcții serverless în local, care pot fi ulterior încărcate în Azure. Funcțiile pot fi scrise în diferite limbaje de programare, cum ar fi: C# compiled, C# script*, JavaScript, Java, PowerShell sau Python. Pentru cercetările aferente acestei teze s-a ales Python pentru implementarea procesului de transformare. Odată ce sunt încărcate noi fișiere de tip *.log* în zona *raw*, se

declanșează automat execuția funcției de tip Blob Trigger urmată de procesul de transformare. Operațiile de tip modificare/ștergere de la nivelul directorului *raw/* nu declanșează execuția funcției, ci doar cele de încărcare de fișiere log.

7.4.3.1 Transformarea datelor de tip WSAL

Pentru implementarea procesului de transformare a datelor de tip *.log* au fost alocate și utilizate următoarele resurse:

- *Azure storage account for Students,* dar poate fi oricare cont General v2;
- *VS Code* instalat pe un sistem de operare, Windows în cazul de față (disponibil și varianta Linux și Mac);
- Python 3.9, dar poate fi orice versiune Python 3.x;
- Extensiile Python pentru VS Code;
- *Azure Functions Core Tools @4*, sau oricare versiune mai mare decât 2;
- *Azure Functions Extension* pentru VS Code.

După instalarea și conectarea resurselor la contul Azure folosind cheile de acces, am realizat funcția serverless Blob Trigger în VS Code [114].

Prin utilizarea mediului de dezvoltare VS Code am creat un nou proiect pentru implementarea funcției de tip Blob Trigger (ANEXA II). Înainte de crearea unui nou proiect, este important ca în VS Code să fie deja realizată conexiunea la contul Azure, astfel încât în momentul în care se generează fișierele pentru noul proiect, să se preia în mod automat parametrii de conectare la Azure storage și astfel să fie permis accesul la resursele din cloud. În etapa de creare a funcției am ales zona din DL pentru care se dorește monitorizarea, în cazul nostru *raw/*. Funcția nou realizată a fost denumită *fadl-blob Trigger-webserver2*. Proiectul conține o colecție de fișiere care definesc funcția de transformare.

La nivel de proiect local se găsește fișierul *local.settings.json* unde sunt definite în mod automat șirurile de conexiune la contul Azure ce conțin cheile de acces atribuite contului de stocare *adlswebserver2.*

În fișierul *function.json* (ANEXA II) sunt definite intrările și ieșirile pentru funcția aferentă procesului de transformare a datelor, astfel sunt definiți un blob de intrare și două blob-uri de ieșire: blob-ul de intrare care transportă datele de intrare, respectiv fișierul WSAL încărcat în zona *raw* ce urmează a fi transformat, al doilea este un blob de ieșire ce va prelua fișierele *.parquet* obținute în urma procesului de transformare, iar cel de al treilea blob de ieșire preia fișierele de tip text cu liniile din fișierul WSAL ce nu au respectat criteriile de transformare definite în specificațiile *regex*, din fișierul *__init__.py*.

Fiecare Blob este definit de o serie de parametri:

- *"name"* reprezintă numele fișierului la care se referă funcția principală;
- *"path"* reprezintă adresa directorului din DL care deservește fișierul. În cazul în care este un blob de intrare, atunci acesta reprezintă adresa de la care se citesc datele din DL, iar în cazul în care este un blob de ieșire, atunci reprezintă adresa la care se scriu datele returnate de funcția principală. Pentru blob-ul de intrare, este important să se precizeze tipul de date așteptat de către funcția principală și anume să fie sub forma unui fișier .log, pentru a evita lansarea în execuție a acesteia pentru fișiere diferite de tipul .log, ceea ce ar duce la erori de execuție;

- *"direction"* definește dacă este un parametru de intrare sau de ieșire a funcției;
- *"connection"* reprezintă datele de conectare la contul Azure.

În fișierul *requirements* am adăugat bibliotecile necesare implementării acestui proiect: azure-functions, pandas, regex, tqdm, pyarrow, dotnet. Odată cu lansarea în execuție a funcției Trigger fișierul *requirements* este accesat. Bibliotecile prezente în acest fișier sunt instalate în cazul în care acestea nu sunt detectate. Aceste biblioteci sunt utilizate de fișierul Python *__init__.py*. Bibliotecile pot fi instalate și manual din command line prin utilizarea comenzii *pip install <nume librărie>*.

În fișierul Python *__init__.py* este specificată funcția *main* care este apelată de către Trigger în momentul în care un blob nou este încărcat cu succes la nivelul directorului monitorizat de acesta. În funcția *main* am realizat secvența de cod scrisă în Python ce îndeplinește procesul de transformare a fișierelor de tip log în fișiere de tip parquet.

Algoritmul 1 prezintă pașii realizați de funcția main lansată de Trigger pentru inițierea procesului de transformare a fișierului log în fișier parquet.

Algoritm 1: Funcția Python Blob Trigger

Intrări: *myblob*
Ieșiri: *myblobout*, *errmyblobout*

```
1       Begin main function
2           regex = <definește expresia rațională>
3           columns = <definește coloanele din fișierul parquet>
4           myblob = <reprezintă datele din blob-ul de intrare myblob în octeți (bytes)>
5           mytxt_obj = <conversia octeților în obiect Unicode pentru a se putea efectua citirea
6                       linie cu linie>
7:          array_log_lines = <declararea unui array ce va capta liniile de log valide>
8:
9:          (Verifică fiecare linie din fișierul log dacă respectă formatul definit în regex)
10:         For each log line
11:         begin
12:           If (check for regex format)
13:            Adaugă linii de log valide în array_log_lines
14:           else
15:            Scrie liniile de log nevalide într-un fișier text ASCII
16:         end for
17:           df_log_line = <încarcă array_log_lines într-un data frame>
18:           apelați funcția to_parquet pentru a converti data frame-ul în fișier parquet
19:
20:         (Setează blob-urile de ieșire cu rezultatele obținute în urma transformării)
21:           myblobout = <atribuie datele în format parquet primului blob de ieșire myblobout>
22:           errmyblobout = <atribuie fișierul text cu erorile de transformare în al doilea blob de
23:                          ieșire errmyblobout>
24:     End main function
```

Funcția Blob Trigger implementată pe baza algoritmului prezentat, care realizează transformarea fișierelor log în fișiere parquet, este prezentată în Figura 7-6.

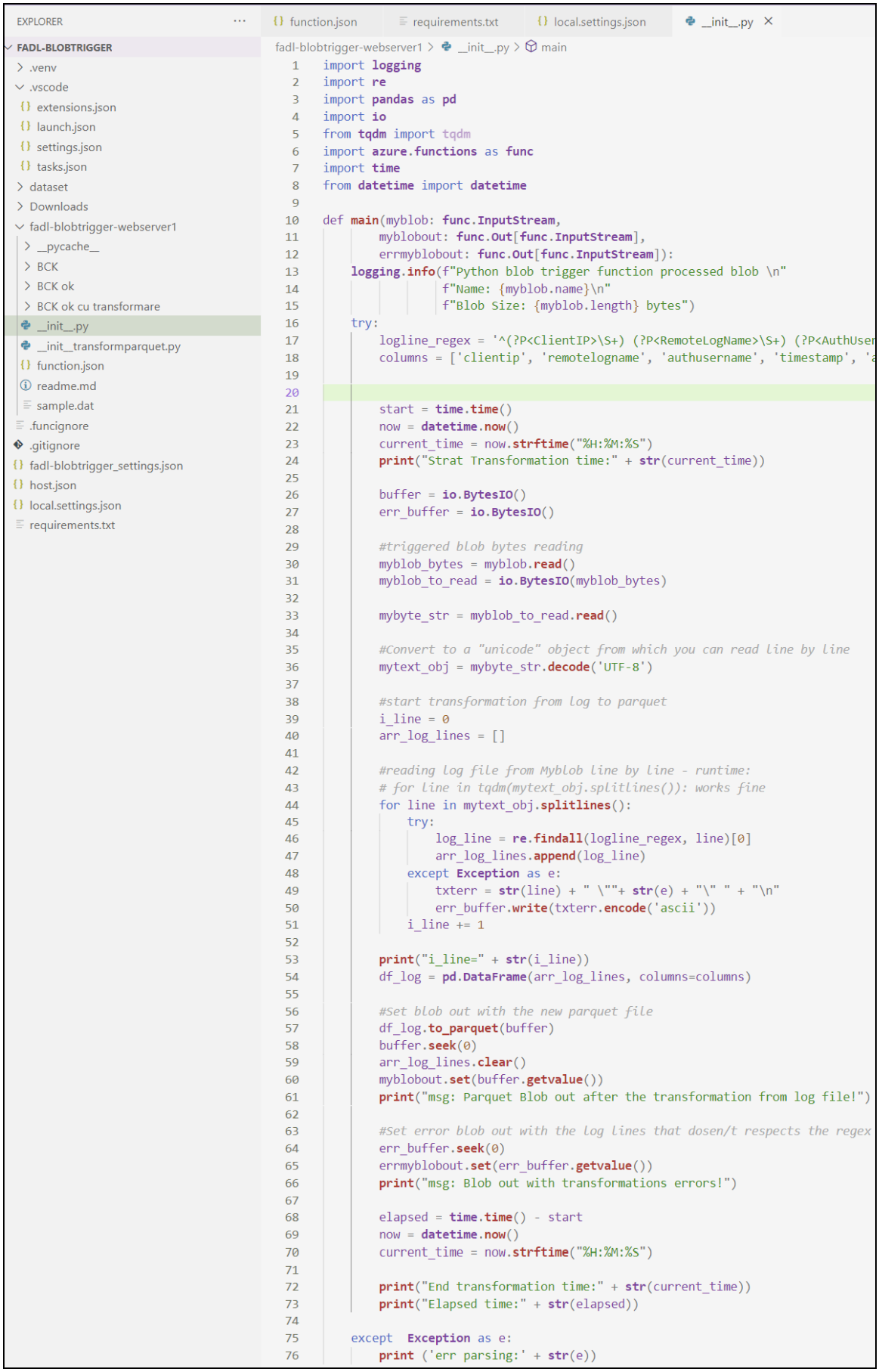


Figura 7-6: Funcția de transformare fadl-blob Trigger-webserver2

Fișierul încărcat în blob-ul *myblob,* ce reprezintă primul parametru de intrare al funcției Blob Trigger, este inițial neformatat. Fișierul este în format binar, prin urmare trebuie decodificat în format UTF-8 pentru a putea fi analizat linie cu linie și supus procesului de transformare într-un format structurat pe coloane. Se realizează astfel citirea octeților până când se atinge EOF, apoi se decodifică octeții citiți în format UTF-8 și sunt atribuiți unui obiect de tip șir de caractere.

Funcția Python bytes *decode*() este utilizată pentru a converti octeții în obiecte de tip string. Odată creat obiectul de tip șir de caractere, prin utilizarea funcției python *splitlines*(), se poate împărți șirul în linii pentru a verifica fiecare linie dacă se potrivește cu formatul regex din Figura 7-7.

```
logline_regex = '^(?P<ClientIP>\S+) (?P<RemoteLogName>\S+)
(?P<AuthUserName>\S+) \[(?P<TimeStamp>[^\]]+)\] "(?P<AccessMethod>[A-Z]+)
(?P<AccessRequest>[^"]+)? HTTP/[0-9.]+" (?P<ResultStatus>[0-9]{3})
(?P<SizeBytes>[0-9]+|-) "(?P<ReferrerURL>[^"]*)" "(?P<UserAgent>[^"]*)"
(?P<extrainfo>\S+)'
columns = ['clientip', 'remotelogname', 'authusername', 'timestamp',
'accessmethod', 'accessrequest', 'resultstatus', 'sizebytes',
'referrerurl', 'useragent', 'extrainfo']
```

Figura 7-7: Definiția Regex parser

Liniile care îndeplinesc criteriile de transformare sunt încărcate într-un Panda DataFrame, care permite organizarea datelor în tabele bidimensionale, cu dimensiuni variabile, cu conținut eterogen [115]. Pe DataFrame-ul obținut *df_log* se aplică ulterior funcția de transformare *df_log.to_parquet(buffer)*. Funcția python *to_parquet()* realizează transformarea datelor din DataFrame într-un format binar parquet, care va fi alocat în buffer-ul de ieșire *io.BytesIO* atribuit primului blob de ieșire *myblobout* și încărcat în cloud DL în directorul *enriched/*.

Liniile care nu respectă formatarea din regex sunt în cele din urmă salvate într-un fișier text codificat ASCII și transferate în al doilea blob de ieșire *errmyblobout*. Astfel, *myblobout* încarcă noul fișier Parquet în directorul *enriched/*, iar *errmyblobout* încarcă fișierul .txt care conține liniile de eroare în directorul *enriched/logerrparsing/* în DL.

7.4.3.2 Execuție și rezultate

După ce este lansată în execuție funcția Blob Trigger, se trece la încărcarea unui fișier log în DL cu ajutorul comenzilor Azure CLI. Pentru un fișier de tip log de 1,0GB, odată ce upload-ul pe server s-a încheiat cu succes, în terminalul din VS Code va apărea o secvență de mesaje ce confirmă faptul că Trigger-ul a depistat un fișier nou încărcat în DL și a determinat execuția funcției ce realizează procesul de transformare la nivelul DL-ului. În Figura 7-8, sunt prezentate rezultatele obținute, astfel putem observa că în urma procesului de ingestie avem blob-ul de intrare de 1048576000 bytes. Fișierul log este supus procesului de citire linie cu linie pentru verificarea respectării formatului din regex pentru fiecare linie. În total, au fost identificate 3170024 de linii de log prelucrate. În urma procesului de transformare dintr-un fișier *log* de 1000 Mebibytes (MiB) echivalentul a 1048576000bytes, am obținut un fișier *parquet* de numai 108,2MiB echivalentul a 113455923,2bytes, ce reprezintă doar 10,82% din

dimensiunea originală. Timpul de transformare variază mult în funcție de mașina pe care se realizează procesul. În cazul de față, pentru un fișier log de 1,0485GB procesul de transformare s-a realizat în circa 76 secunde din momentul în care Trigger-ul a confirmat că un nou fișier log s-a încărcat cu succes în zona *raw*. În procesul de transformare se întâlnește, de asemenea, un timp de întârziere de circa 8÷10minute din momentul în care s-a terminat upload-ul până când Trigger-ul confirmă faptul că s-a depistat un fișier încărcat cu succes. Acești timpi de întârziere apar datorită faptului că Trigger-ul are la bază un sistem de scanare a fișierelor log de pe server-ul din cloud care confirmă încărcarea unui nou fișier în zona monitorizată de Trigger. Acești timpi pot fi îmbunătățiți conform [116]. În practică acești timpi pot varia în funcție de tipul de cont Azure ales, de viteza de comunicație din rețeaua WAN, etc. Trebuie precizat faptul că cercetările actuale se realizează în Azure dintr-un cont student cu resurse limitate.

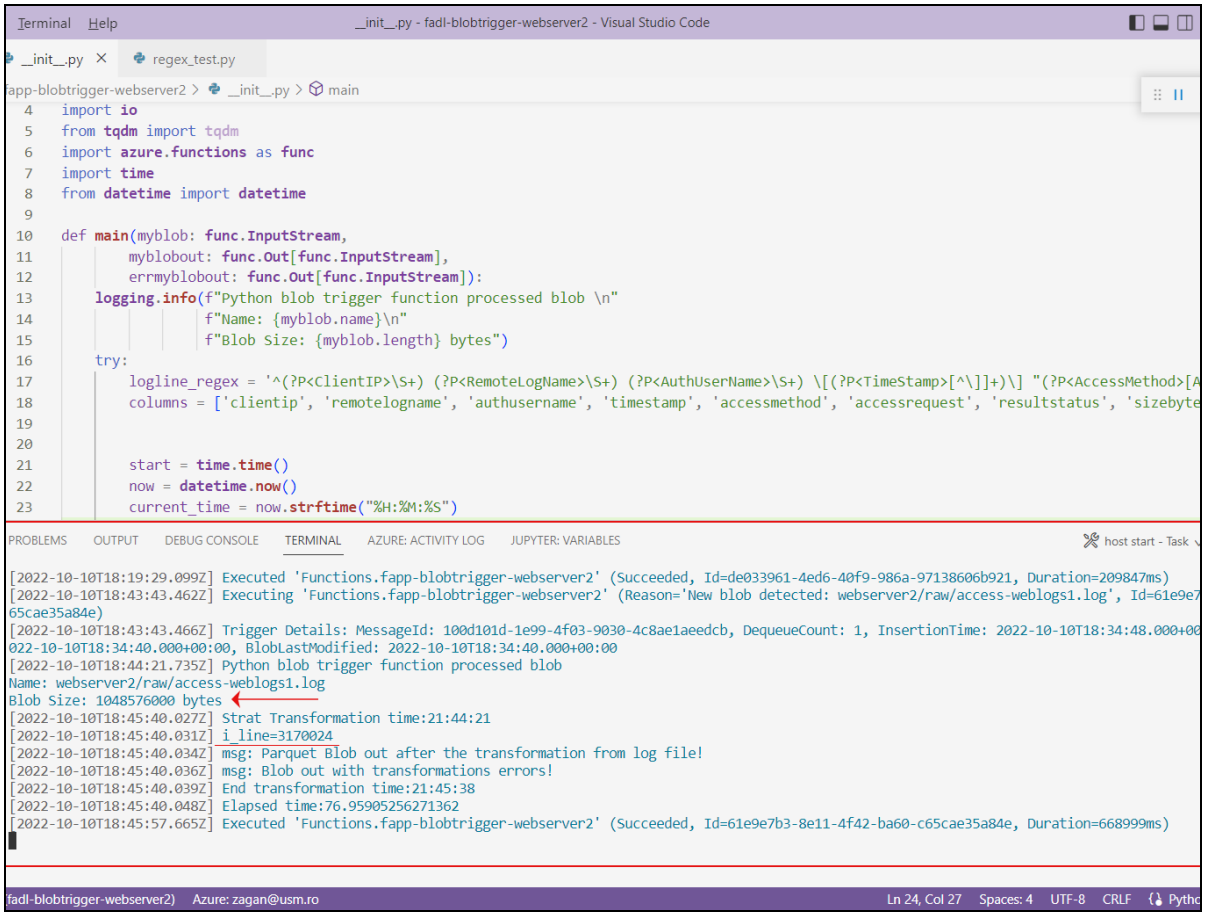


Figura 7-8: Execuția funcției serverless din VS Code

După transformare s-a obținut, din fișierul log cu date brute de 1,0485GB (1000MiB), un fișier *parquet* denumit *access-weblog1.parquet* de doar 108,39MiB și un fișier text denumit *access-weblog1.txt* cu erori de transformare ce ocupă 102,02KiB. Fișierul sursă a fost redus la 10,8% din dimensiunea originală, de unde deducem că spațiul de stocare se optimizează într-un procent de aproximativ 89%. În fișierele text se captează liniile log ce nu au putut fi transformate deoarece nu au respectat constrângerile impuse de regex. Figura 7-9 ilustrează o secvență de acest tip ce a fost generată în urma procesului de transformare.

Figura 7-9: Linii log returnate ca erori în urma procesului de transformare

După verificarea liniilor log din Figura 7-9, se observă că acestea sunt linii la care serverul a returnat ca răspuns codurile HTTP Status Code 400 și 414 cererilor primite de la clienți:

- 400 corespunde situației în care serverul nu poate sau nu vrea să proceseze cererea din cauza unei probleme care este percepută ca fiind o eroare a clientului.
- 414 indică faptul că serverul refuză să răspundă la solicitare deoarece cererea este prea lungă pentru ca serverul să o poată interpreta.

Trebuie evaluată natura erorilor și eventual adaptat regex-ul astfel încât să nu existe pierderi de date în timpul procesului de transformare.

Pentru validarea studiului din cadrul acestui proiect de cercetare au fost executate o serie de teste diferite care să ofere o perspectivă asupra beneficiilor și eficienței proceselor de ingestie, stocare și transformare a datelor de tip WSAL în DL. Pentru aceasta a fost efectuată o primă împărțire a fișierului log original în patru fișiere separate în funcție de dimensiunea dorită. S-au obținut trei fișiere de 1GB și un ultim fișier de 356MB. Valorile din Tabelul 7-1 au fost executate de pe un sistem de calcul (P1) cu următoarele caracteristici: procesor Intel(R) Core(TM) i5-4310U CPU @ 2.00GHz 2.60 GHz, RAM 8.00 GB, sistem de operare Windows 10.

Tabel 7-1: Rezultate experimentale obținute în urma procesului de transformare cu sistemul P1.

Fișier log	Log (MB)	Număr de linii procesate	Timp de execuție (s)	DL log (MiB)	DL err txt (KiB)	DL parquet (MiB)	Spațiul de stocare utilizat
access-weblog1.log	1024,00	3170024	76.95	1000.00	102.02	108.39	10,23%
access-weblog2.log	1024,00	3037405	75.51	1000.00	21.99	111.20	11,12%
access-weblog3.log	1024,00	3203638	107.60	1000.00	18.78	116.06	11,66%
access-weblog4.log	348,35	954088	17.86	340.19	8.35	31.26	13,00%
Total	**3420,35**	**10365155**	**277.92**	**3340.19**	**151.14**	**366.91**	**11%** *(media)*

Din acest tabel se observă că spațiul de stocare, după transformarea datelor, este redus cu aproximativ 89%. Dimensiunea fișierelor log și parquet obținute în urma procesului de transformare reprezintă doar 11% din dimensiunea datelor brute.

În urma testelor efectuate în portalul Azure (Figura 7-10), în fiecare din cele trei directoare *raw*, *enriched* și *enriched/logerrparsing* sunt stocate cele patru fișiere ce corespund fiecărui upload efectuat în DL.

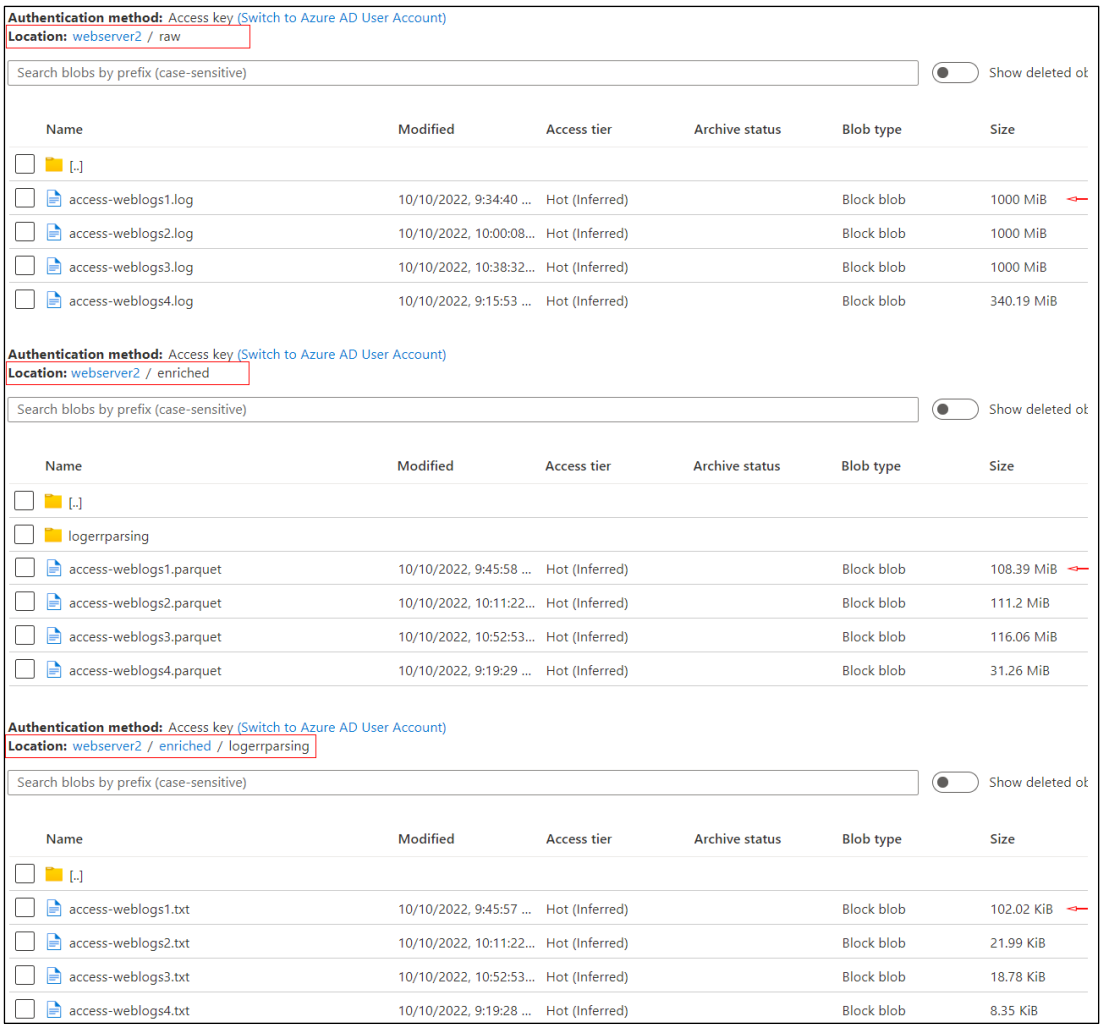
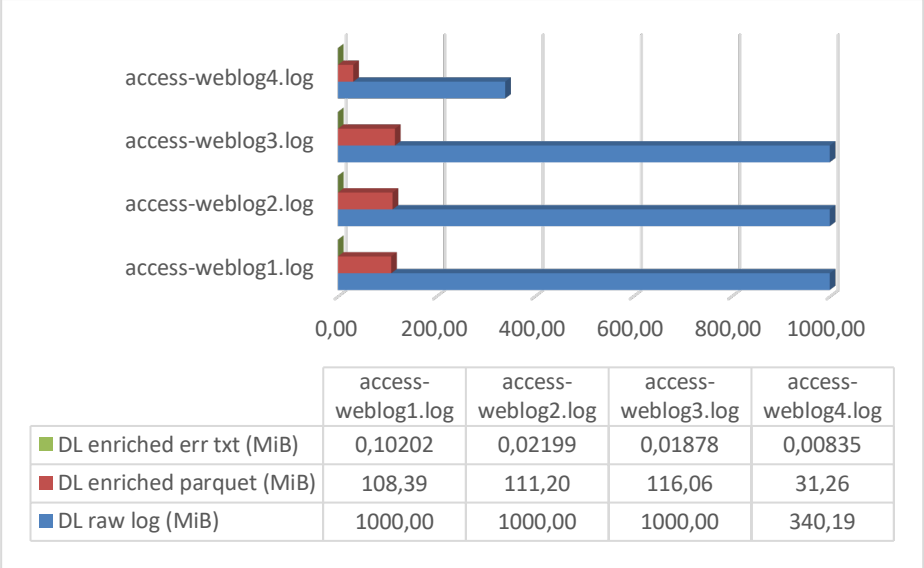


Figura 7-10: Rezultatele transformărilor în Azure DL

Graficul din Figura 7-11 ilustrează reducerea spațiului de stocare a datelor brute prin transformarea în fișiere parquet.



	access-weblog1.log	access-weblog2.log	access-weblog3.log	access-weblog4.log
DL enriched err txt (MiB)	0,10202	0,02199	0,01878	0,00835
DL enriched parquet (MiB)	108,39	111,20	116,06	31,26
DL raw log (MiB)	1000,00	1000,00	1000,00	340,19

Figura 7-11: Reducerea spațiului de stocare în urma procesului de transformare cu sistemul P1

Deoarece procesul de transformare se execută serverless timpii de execuție depind de sistemul pe care rulează această funcție. Pentru a valida această afirmație au fost realizate o nouă serie de teste cu aceleași fișiere log pe un sistem de calcul P2 cu următoarele caracteristici: procesor Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz, RAM 64.00 GB, sistem de operare Windows 11Pro. În Tabelul 7-2 sunt menționați timpii de transformare obținuți cu sistemele P1 și P2, indicându-se procentul de îmbunătățire aferent utilizării sistemului P2.

Tabel 7-2: Timpii de transformare obținuți în urma proceselor de transformare cu P1 și P2

Log File	Timp de execuție P1 (s)	Timp de execuție P2 (s)	Diferența P1-P2 (s)	Îmbunătățire P2 (%)
access-weblogs-1.log	76,95	25,46	51,49	33,09%
access-weblogs-2.log	75,51	22,49	53,02	29,78%
access-weblogs-3.log	107,6	26,23	81,37	24,38%
access-weblogs-4.log	17,86	7,47	10,39	41,83%
Total	**277,92**	**81,65**	**196,27**	**29,38%**

În Figura 7-12 sunt vizibile îmbunătățirile aduse de sistemul P2 în procesul de transformare datorită performanțelor acestuia.

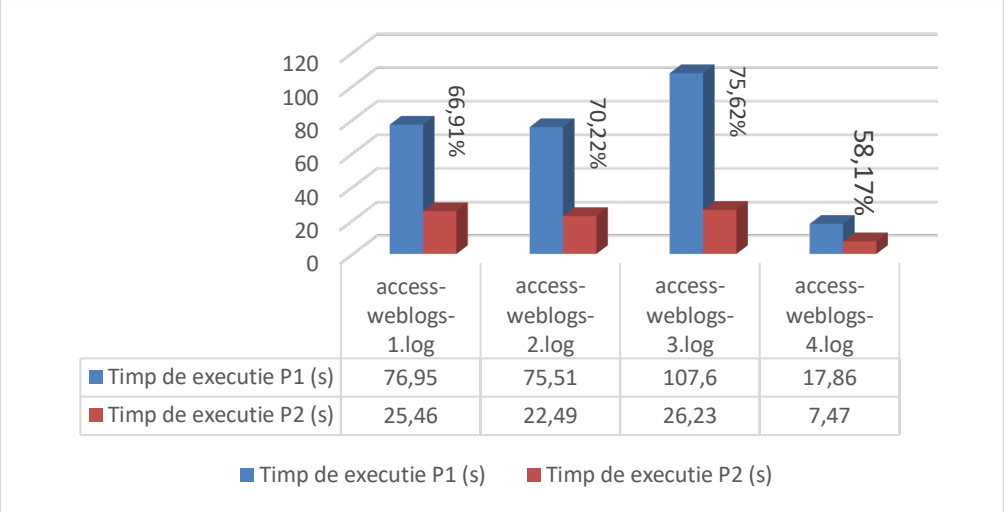
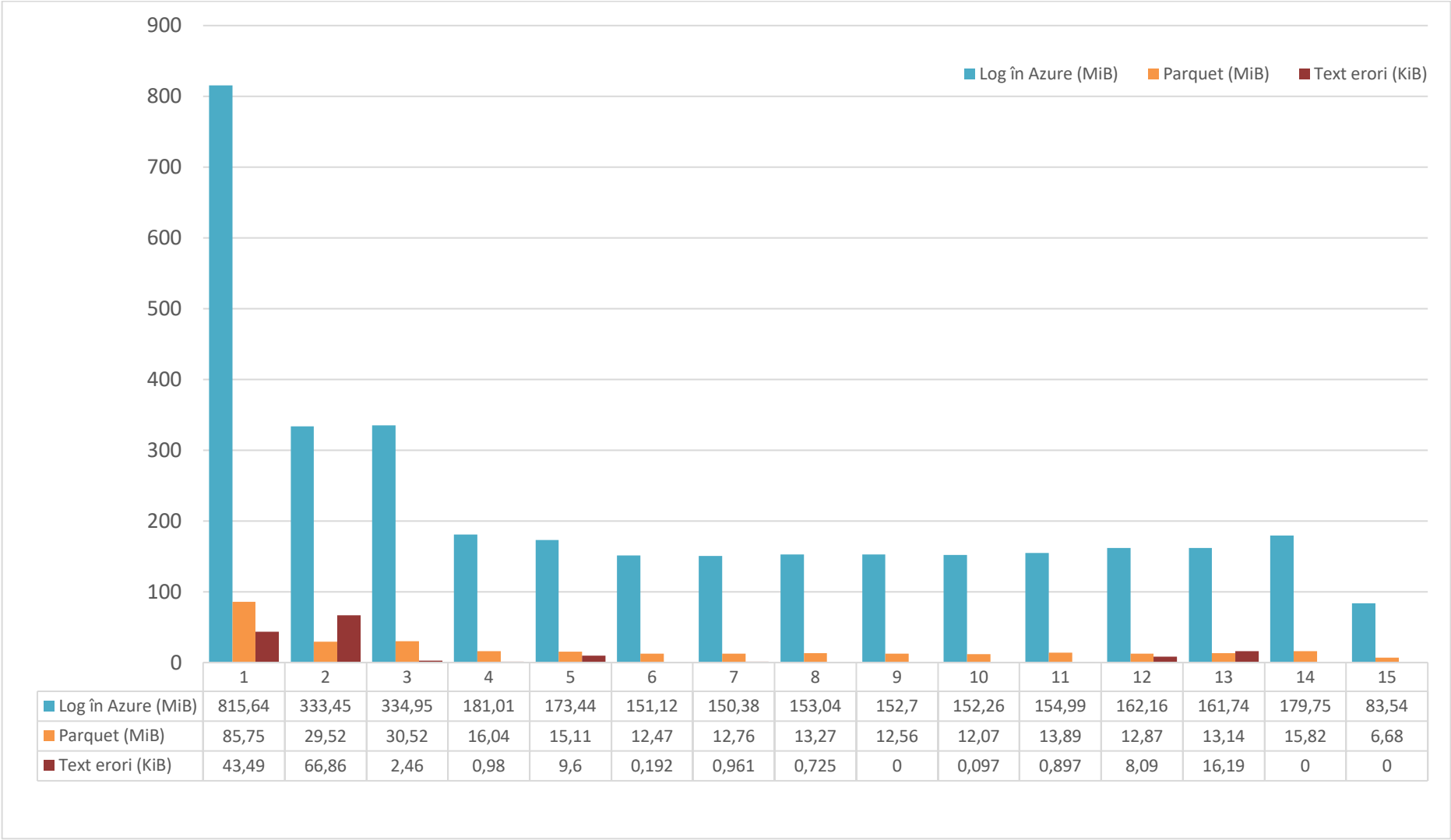


Figura 7-12: Dependența timpilor de transformare conform sistemului care execută procesul

Un nou set de teste a fost efectuat pentru a demonstra eficiența algoritmului propus și pentru a arăta faptul că procentul de reducere a spațiului de stocare se menține ridicat. Astfel, a fost realizată o nouă împărțire a fișierului log original, prin intermediul aplicației Git Bash, care permite divizarea unui fișier log în funcție de numărul de linii dorit. Au fost realizate diverse diviziuni și subdiviziuni. Tabelul 7-3 prezintă datele obținute în urma proceselor de încărcare și transformare în DL, fiind specificate următoarele informații: dimensiunea originală a fișierelor sursă stocate local, spațiul ocupat de datele brute în DL după ingestie, numărul de linii identificate în fișierele sursă, spațiul ocupat în DL de fișierele parquet și text (ANEXA III), liniile log contorizate ca fiind încărcate în fișierele parquet și text în timpul procesului de transformare și de asemenea timpii de transformare obținuți cu sistemul de calcul P2 (Figura 7-13).

Tabel 7-3: Rezultatele obținute în urma proceselor de transformare cu P2

	Fișiere încărcate	Log în local (MB)	Log în Azure (MiB)	Parquet (MiB)	Text erori (KiB)	Linii procesate	Linii transformate *parquet*	Linii cu erori de procesare fișierul *text*	Timp de execuție
1	access-weblogs-5182576-a.a.log	835,22	815,64	85,75	43,49	**2591288**	2591271	17	18.66
2	access-weblogs-5182576-a.b-1025212.a.log	341,45	333,45	29,52	66,86	**1025212**	1025193	19	8,65
3	access-weblogs-5182576-a.b-1025212.b.log	342,99	334,95	30,52	2,46	**1025212**	1025203	9	8,69
4	access-weblogs-5182576-a.b-1025212.c.log	185,36	181,01	16,04	0,98	**540864**	540859	5	4,58
5	access-weblogs-5182576-b.495632.a.log	177,60	173,44	15,11	9,6	**495632**	495623	9	3,66
6	access-weblogs-5182576-b.495632.b.log	154,75	151,12	12,47	0,192	**495632**	495630	2	3,67
7	access-weblogs-5182576-b.495632.c.log	153,99	150,38	12,76	0,961	**495632**	495627	5	3,72
8	access-weblogs-5182576-b.495632.d.log	156,72	153,04	13,27	0,725	**495632**	495628	4	3,75
9	access-weblogs-5182576-b.495632.e.log	156,37	152,7	12,56	0	**495632**	495632	0	3,61
10	access-weblogs-5182576-b.495632.f.log	155,92	152,26	12,07	0,097	**495632**	495631	1	3,65
11	access-weblogs-5182576-b.495632.g.log	158,71	154,99	13,89	0,897	**495632**	495629	3	3,68
12	access-weblogs-5182576-b.495632.h.log	166,06	162,16	12,87	8,09	**495632**	495631	1	3,69
13	access-weblogs-5182576-b.495632.i.log	165,62	161,74	13,14	16,19	**495632**	495630	2	3,64
14	access-weblogs-5182576-b.495632.j.log	184,07	179,75	15,82	0	**495632**	495632	0	3,75
15	access-weblogs-5182576-b.495632.k.log	85,54	83,54	6,68	0	**226256**	226256	0	1,72
		3420,35	**3340,17**	**302,47**	**150,542**	**10365152**	**10365075**	**77**	**60,46**



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Log în Azure (MiB)	815,64	333,45	334,95	181,01	173,44	151,12	150,38	153,04	152,7	152,26	154,99	162,16	161,74	179,75	83,54
Parquet (MiB)	85,75	29,52	30,52	16,04	15,11	12,47	12,76	13,27	12,56	12,07	13,89	12,87	13,14	15,82	6,68
Text erori (KiB)	43,49	66,86	2,46	0,98	9,6	0,192	0,961	0,725	0	0,097	0,897	8,09	16,19	0	0

Figura 7-13: Reducerea spațiului de stocare în urma procesului de transformare conform Tabelului 7-3

Cu ajutorul instrumentului PowerBI Desktop (ANEXA IV) am recurs la verificarea înregistrărilor conținute atât de fișierul log original cât și de fișierele parquet și text, astfel în Tabelul 7-4 se poate observa că nu există pierderi de date în procesul de transformare.

Tabel 7-4: Numărul total de linii din fișierul sursă și fișierele obținute

Power BI Desktop	Număr de linii
Numărul total de înregistrări din fișierele parquet	**10365075**
Numărul total de înregistrări din fișierele text cu erori	**77**
Numărul total de înregistrări din fișierul log	**10365152**

După acest nou set de teste, am obținut o diferență de 3037,5495MiB între spațiul alocat pentru fișierele log brute (3340,17MiB) și spațiul ocupat de fișierele parquet (302,47MiB) și text (0,1505MiB) în DL, rezultând un procent de 90,93% de reducere a spațiului de stocare.

În urma rezultatelor practice observăm că se menține procentul de 89% ~ 90% pentru optimizarea spațiului de stocare. Astfel procesul de transformare a datelor brute semi-structurate în date în format colonar, vine cu două avantaje importante atât din punct de vedere a reducerii semnificative a costurilor prin reducerea spațiului de stocare cât și al accesibilității datelor de către motoarele de analiză avansată, care își îmbunătățesc astfel timpii de procesare. Odată ce fișierele log din zona *raw* au fost transformate în fișiere *parquet*, se poate trece la mutarea fișierelor log într-un *Archive tier* pentru arhivarea acestora pe termen lung și pentru reducerea semnificativă a costurilor de stocare.

ADLS Gen2 este un mediu de stocare eficient din punct de vedere al costurilor și nu prezintă costuri inițiale. Este un serviciu de tip *pay-per-use*, ce permite utilizatorilor să plătească doar pentru numărul de gigaocteți stocați și pentru numărul de tranzacții (citiri și scrieri) care au loc asupra datelor. Costurile diferă în funcție de tipul de abonament, de tipul de redundanță a datelor LRS, ZRS, GRS, RA-GRS, sau de regiunea selectată.

Azure oferă, un calculator online pentru estimarea prețului. Orientativ, pentru un cont standard cu stocare LRS ierarhică și redundantă activă, costurile pot începe de la 0,01699euro/GB hot tier, 0,00924euro/GB cool tier, 0,00092euro/GB archive tier pentru primii 50TB/lună. Ca și costuri per tranzacție pentru operațiile de scriere (la fiecare 4MB, la 10000 de operații): hot tier 0,06002€, cool tier 0,12003€, archive tier 0,12003€ și operațiile de citire hot tier 0,00481€, cool tier 0,01201€, archive tier 6,00130€ [117].

8 Contribuții personale și direcții viitoare de cercetare

Obiectivul principal al tezei constă în realizarea de cercetări privind noua tehnologie de stocare a datelor brute și anume Data Lake. A fost identificată importanța fișierelor web server access log și problema majoră cu care se confruntă specialiștii în domeniu pentru stocarea acestora pe termen lung, cu scopul de a fi supuse proceselor de analiză avansată. Informațiile pe care le dețin aceste fișiere pot aduce îmbunătățiri în diferite domenii de interes, după cum am văzut în capitolul 7.1. Această teză de cercetare își propune să ofere o soluție economică, sigură și performantă pentru dezvoltarea și implementarea proceselor de ingestie, stocare și transformare a fișierelor WSAL folosind tehnologia ADLS Gen2. Pe lângă proiectarea unei arhitecturi DL dedicate, un alt element de noutate și originalitate a lucrării constă în faptul că, utilizarea fișierelor WSAL ca sursă de date brute pentru un DL nu a mai fost întâlnită în cercetările bibliografice realizate. Această teză prezintă atât o descriere teoretică detaliată, cât și rezultatele experimentale obținute în urma cercetărilor privind procesele de ingestie, stocare și transformare a datelor în DL. Deși soluția oferită se bazează în mod explicit pe ADLS Gen2, aceasta reprezintă un punct de referință important în abordarea unei soluții DL pentru stocarea fișierelor WSAL, fie ea implementată în cloud sau local.

Pentru atingerea acestui obiectiv mi-am propus următoarele **obiective specifice**:

OS1. Analiza stadiului actual al cercetărilor privind arhitecturile DL și aprofundarea acestui nou concept emers datorită evoluției tehnologice, a vitezei cu care datele sunt generate și a varietății acestora. Am evidențiat neajunsurile care au dus la dezvoltarea stringentă a acestui nou concept care a transformat modul de implementare a unui mediu de stocare a datelor, prin inversarea proceselor de transformare cu cel de încărcare din pipeline-ul ETL, specific tehnologiilor precedente. Acest nou pipeline, denumit ELT, facilitează în primul rând stocarea tuturor datelor în forma lor brută cu multiple beneficii, prezentate în această lucrare de cercetare. Acest nou concept, datorită avantajelor majore pe care le aduce în domeniul BD, a forțat tehnicile specifice diferitelor niveluri (stocare, încărcare, transformare și analiză) să se adapteze, să se dezvolte, fiind constant îmbunătățite pentru a satisface caracteristicile datelor big data (*big data v's*). Diseminarea acestor studii bibliografice au fost prezentate în cadrul conferinței **DAS2020**. Cercetările au continuat cu realizarea unui studiu comparativ extins între cele două pipeline-uri ETL și ELT, cât și a mediilor de stocare Data Warehouse și Data Lake. Aceste studii comparative au fost publicate în jurnalul **IJCSNS2019**.

OS2. Identificarea diferitelor metodologii de implementare a DL prin realizarea unui studiu comparativ din punct de vedere al costurilor și al resurselor necesare. În bibliografia actuală acest concept nou este abordat mai puțin la nivel practic și mai mult la nivel teoretic. A fost necesar realizarea unui studiu bibliografic vast pentru identificarea metodologiilor și diferitelor tehnici de implementare existente pe piață (*On-Premises, Cloud, Multi-Cloud* și *Hybrid*), pentru punerea în practică a unui DL care să implice resurse minime disponibile pentru realizarea acestui proiect de cercetare. În urma acestui stadiu de cercetare a fost publicat un articol în cadrul conferinței **HORA2021**.

OS3. Realizarea de cercetări privind **framework-ul Hadoop prin implementarea practică a *Single-Node* și *Multi-Node Cluster*** în local, pe un sistem Linux, cu scopul de a aprofunda stocarea HDFS. Rezultatele acestor implementări practice și abordări teoretice au fost publicate în jurnalul IJACSA2021 și conferința DANUBIUS2021. A fost de asemenea realizat un studiu bibliografic a diferitelor versiuni apărute, cu scopul de a urmări îmbunătățirile aduse framework-ului Hadoop în decursul anilor. În urma acestui stadiu de cercetare au putut fi evidențiate avantajele dar și dezavantajele implementării unei astfel de arhitecturi pe plan local (*On-Premise*s) utilizând framework-ul Hadoop. Inconvenientul major constă în faptul că o astfel de implementare solicită resurse hardware, software și inginerești de înaltă performanță.

OS4. Cercetările au continuat pentru **identificarea unei soluții de implementare a unei arhitecturi DL în Cloud** care vine cu o serie de avantaje dar și dezavantaje ce au fost punctual evidențiate pe parcursul acestei lucrări. A fost necesar realizarea unui studiu bibliografic a diferiților furnizori de servicii Cloud care permit implementarea unei arhitecturi DL. Este important de menționat faptul că nu toți furnizorii de servicii Cloud au pus la dispoziție o platformă de implementare a unui DL în Cloud. Dintre furnizorii care dispun de tehnica și instrumentele necesare amintim: Microsoft, AWS, Google, Oracle, IBM, Cloudera, etc. Tehnologiile folosite de aceste companii sunt diferite, iar serviciile și tehnicile puse la dispoziție utilizatorilor sunt și ele diferite. Alegerea unui furnizor trebuie făcută în baza a diferite criterii. În lucrarea de față sunt prezentate diferite ecosisteme DL, Microsoft Azure Data Lake Storage Gen2, Amazon S3, Google Cloud BigLake, OCI Data Lake cu scopul de a evidenția diversitatea tehnologiilor de implementare și a tehnicilor de utilizare. Implementarea în Cloud a unui DL necesită efectuarea unui studiu de caz pentru găsirea unei soluții optime care se pliază pe nevoile fiecărei organizații în parte. A fost întocmit un studiu comparativ între cele două metode de implementare a unei arhitecturi DL, și anume în Cloud și On-Premises cu scopul de a evidenția diferențele majore din punct de vedere al costurilor, al timpilor de implementare, a resurselor hardware și software necesare, a complexității de implementare, a fiabilității, etc. Pe baza cercetărilor efectuate asupra diferiților furnizori de servicii cloud, a fost propusă implementarea unei arhitectură DL utilizând Microsoft ADLS Gen2 pentru stocarea și prelucrarea fișierelor WSAL. Au fost prezentate aspecte teoretice ce stau la baza implementării unui DL în ADLS Gen2, prin detalierea diferitelor aspecte tehnice referitoare la nivelurile de ingestie, stocare, gestionare, securitate, acces, redundanță, procesare și analiză a datelor. Astfel au putut fi evidențiate diferitele avantaje ce stau la baza implementării unui DL în ADLS Gen2.

OS5. Am proiectat și implementat o arhitectură DL în cloud pentru stocarea de fișiere WSAL pe termen lung. Fișierele WSAL, sunt fișiere ce conțin volume mari de date nestructurate pentru care este necesar un mediu de stocare autoscalabil. WSAL sunt dificil de stocat, procesat și analizat la scară largă, astfel am propus stocarea acestora într-o arhitectură de tip DL. În acest scop am propus un **model de stocare ierarhică** care să deservească diferitele niveluri din arhitectura DL. *Zona de date raw*, în urma procesului de ingestie, stochează datele în forma lor brută. *Zona de date enriched* stochează datele provenite din *zona raw* ce au fost supuse unui proces automat de transformare și standardizare, datele brute nestructurate fiind transformate în date structurate și salvate în fișiere de tip parquet. În această

zonă sunt, de asemenea, stocate într-un director separat denumit *logerrparsing* și liniile log care nu au putut fi transformate conform criteriului impus. *Zona de date curated* stochează date preprocesate pregătite pentru consum în funcție de scopul urmărit. Pe ultimul nivel este colocată *zona de date workspace* cu date complet transformate și agregate pentru a putea fi accesate de către diferitele echipe de lucru care înțeleg nevoile organizației. Pe baza acestui model au fost făcute recomandări ce privesc reguli de DLM pentru reducerea costurilor de stocare.

OS6. Am dezvoltat un proces de ingestie a datelor care să permită încărcarea datelor în DL în *zona de date raw.* A fost utilizată tehnica open-source Azure CLI ce permite execuția de comenzi administrative *cross-platform.* Astfel, am realizat o secvență de comenzi CLI care să permită conectarea la DL, crearea ierarhiei de directoare și subdirectoare la nivelul DL și încărcarea datelor în DL în zona de date brute. Sistemul de ingestie propus poate fi inclus într-un proces batch pe server, cu execuție automată la intervale regulate de timp. În urma procesului de ingestie datele sunt stocate în forma lor brută fără a suferi transformări. Deoarece WSAL în formă brută sunt date greu de analizat și interpretat, s-a propus transformarea lor în fișiere parquet. În urma acestor implementări practice a fost publicat un articol la **DAS2022**.

OS7. Pentru **transformarea automată a datelor din fișiere log în fișiere parquet** am realizat o funcție Azure de tip Blob Trigger scrisă în Python. Această funcție de tip trigger monitorizează zona raw din lacul de date. În momentul în care se depistează faptul că un nou fișier de tip log a fost încărcat cu succes, triggerul lansează în execuție funcția de transformare. Au fost realizate diverse teste de încărcare a datelor brute în DL cu scopul de a ajusta criteriul de transformare astfel încât să rezulte cât mai puține linii log care nu au fost conforme și au fost redirecționate în directorul *logerrparsing*. În urma transformării fișierelor log în fișiere parquet s-a observat o reducere a spațiului de stocare cu aproximativ 89% față de dimensiunea originală. Această transformare vine cu două avantaje majore: reducerea semnificativă a costurilor pentru spațiul de stocare și transformarea datelor în date structurate ușor de procesat. Cercetările efectuate în această etapă au contribuit la publicarea unui articol științific în jurnalul **IEEEACCESS2023** cu factor de impact **3,476**.

Contribuțiile personale constau în propunerea unei arhitecturi DL folosind Azure Data Lake Storage Gen2 pentru realizarea practică a proceselor ELT asupra fișierelor WSAL.

În prima parte a tezei am prezentat o serie de elemente tehnice, care stau la baza implementării unei arhitecturi DL, obținute în urma studiului unei vaste bibliografii. În urma cercetărilor efectuate am reușit să proiectez și să implementez o arhitectură stabilă, fiabilă și economică prin utilizarea unor tehnici rentabile pentru atingerea scopului urmărit. Pentru ingestia datelor am apelat la tehnologia Azure CLI și am implementat o secvență de comenzi cross-platform, care în producție pot fi incluse într-un proces batch ce poate fi executat în background la intervale regulate de timp. Astfel, ingestia în DL a fișierelor de tip log se poate face în cadrul unui proces complet autonom. Odată realizată ingestia datelor în DL acestea sunt supuse unui proces de transformare a fișierelor de tip log în fișiere parquet. Pentru automatizarea acestui proces am implementat o funcție serverless Azure Function de tip Blob Trigger scrisă în limbajul Python. Am propus de asemenea tehnici de DLM la nivelul lacului de date pentru a reduce costurile de stocare.

Soluția propusă în cadrul prezentei teze de doctorat, precum și contribuțiile cercetărilor teoretice respectiv practice aferente acesteia, sunt la nivel de abordare ideatică, prin urmare trebuie specificat faptul că resursele folosite sunt reduse dar validează conceptul de stocare și procesare a fișierelor WSAL în DL. Prin implementarea și validarea arhitecturii descrise în această teză s-a obținut o soluție performantă care deservește diferitele etape de ingestie, stocare și transformare a fișierelor de tip WSAL pentru a putea fi supuse ulterior diferitelor procese de analiză avansată cu scopul de a extrage informații valoroase din acestea.

Spre finalul tezei, sunt prezentate concluziile cercetărilor efectuate, contribuțiile aduse în acest domeniu cât și direcțiile viitoare de cercetare. Cercetările efectuate pentru elaborarea prezentei teze de cercetare s-au finalizat printr-o serie de contribuții teoretice și practice, rezultatele științifice au fost diseminate în cadrul conferințelor și jurnalelor de specialitate.

8.1 Concluzii

În cadrul acestei teze s-a prezentat noul concept Data Lake, s-au evidențiat noutățile aduse în domeniu, s-au subliniat avantajele și dezavantajele față de un Data Warehouse și totodată s-a încercat să se dea un răspuns la întrebarea dacă acesta va înlocui proiectele de Data Warehouse în viitorul apropiat.

După cum am putut observa noua tehnologie DL, care nu cu mulți ani în urmă era considerată ca fiind o tehnologie destinată eșecului datorită complexității sale, astăzi ocupă primul loc în ierarhia tehnologiilor de stocare a datelor Big Data. Astăzi DL este o tehnologie în plină dezvoltare care de foarte puțin timp a reușit să fie disponibilă și în Cloud și astfel să devină mult mai accesibilă și ușor de implementat.

Cele patru moduri diferite de implementare a unei arhitecturi DL, și anume on-premises, cloud, hybrid și multi-cloud, acoperă cu succes cerințele pieței din ziua de astăzi din perspectiva datelor Big Data. Totuși noul trend este de a muta și implementa DL în cloud, care oferă servicii complete pentru toate procesele de ingestie, stocare, procesare și analiză a datelor cu un nivel de securitate ridicat și mereu îmbunătățit de furnizorii de servicii cloud.

Au fost aprofundate pe parcursul acestei teze de cercetare tehnologiile de stocare hadoop și cloud pentru arhitecturile DL. Fiecare dintre cele două tehnologii vor deservi în scopuri foarte bine studiate astfel încât ambele vor ocupa în continuare un loc principal în diferite domenii de activitate. Alegerea unei tehnologii în defavoarea celeilalte va depinde întotdeauna de un proces aprofundat de analizare a avantajelor și dezavantajelor pe care le prezintă fiecare în parte, astfel încât să deservească cât mai optim scopul urmărit. Unii specialiști în domeniu, după cum am văzut în [14], au venit cu propunerea chiar de a îmbina cele două tehnologii DL și DW cu scopul utilizării beneficiilor celor două.

Acest proiect de cercetare demonstrează importanța utilizării noii tehnologii DL pentru stocarea volumelor mari de date nestructurate:

- Stocarea ieftină a unui număr nelimitat de date;
- Spațiu centralizat pentru stocarea datelor de tipuri diferite;
- Permite colectarea diferitelor tipuri de date “pentru orice eventualitate” care să poată fi analizate în caz de nevoie. Astfel, o organizație poate avea acces la informații variate și nu doar informații ce vizează un anumit subiect;
- Permite integrarea ușoară a datelor indiferent de tipul acestora;

- Datele sunt stocate fără a fi necesară o modelare care să preceadă procesul de ingestie - "*schema on read*";
- Poate fi folosit și ca metodă de completare, sau ca o sursă de alimentare a depozitelor DW existente deja în cadrul organizațiilor;
- Permite utilizarea de tehnologii built-in sau create ad-hoc pentru a filtra volume mari de date cu performanțe mult mai bune decât în cazul depozitelor de date;
- Acces rapid la date, care oferă multiple beneficii pentru cercetătorii de date, chiar și pentru utilizatorii finali mai puțin experți în domeniu;
- Multiple metode de gestionare/restricționare a accesului la date cu scopul de a întări securitatea acestora;
- Este un mediu ideal de stocare a datelor de streaming IoT;
- Permite realizarea de backup-uri pentru datele din depozitele de date existente;
- În cazul ADLS Gen2 bazat pe Hadoop, disponibilitatea datelor este ridicată și există funcții încorporate pentru recuperarea lor în caz de dezastru;
- Poate ingera rapid fișiere mari cu diferite tipuri de redundanță a datelor la costuri diferite (LRS, ZRS, RA-GRS, RA-GZRS, GRS, GZRS);
- Permite crearea unei arhive de rezervă a datelor brute pe termen lung la costuri reduse;
- Permite ca datele să fie utilizate de mai multe ori pentru diferite nevoi analitice și cazuri de utilizare;
- Este mult mai economică decât un DW datorită faptului că permite diferite criterii de stocare (*tiers*) pentru gestionarea ciclului de viață a datelor;
- Un alt avantaj major al DL constă în faptul că permite simultan aplicarea de diferite tehnici de calcul asupra datelor spre deosebire de RDBMS;
- Utilizatorii finali au posibilitatea de a accesa cu ușurință datele din orice locație.

În cadrul acestei teze de cercetare a fost propus, proiectat și implementat un sistem eficient și fiabil cu costuri reduse pentru stocarea volumelor mari de fișiere WSAL apelând la tehnologia cloud Azure Data Lake Storage Gen2 oferită de Microsoft, cu scopul de a putea fi supuse proceselor de analiză avansată pentru obținerea de informații valoroase. Au fost evidențiate avantajele stocării pe termen lung a WSAL și totodată îmbunătățirile care pot fi aduse organizațiilor în urma proceselor de analiză a acestora.

În ceea ce privește domeniile de aplicație, implementarea propusă, bazată pe tehnologia DL în cloud, este o soluție de perspectivă cu costuri reduse, orientată spre viitor, cu performanțe remarcabile în ceea ce privește stocarea și analiza datelor de tip WSAL pentru a obține informații valoroase de la serverele web.

8.2 Contribuții teoretice, practice și experimentale

Contribuțiile din această teză sunt rezultatul cercetărilor teoretice și aplicative privind stocarea volumelor mari de date nestructurate cu ajutorul celei mai noi tehnologii existente pe piață și anume Data Lake. Fiind un domeniu nou, literatura de specialitate este vastă dar puțin ambiguă din punct de vedere tehnic, această tematică fiind dezbătută în mare parte doar la nivel teoretic.

O arhitectură DL poate fi implementată în patru moduri diferite: *on-premises, cloud, multi-cloud* și *hybrid*. Prin abordarea practică a acestei tehnologii am realizat o serie de cercetări bibliografice și implementări concrete ce au permis realizarea un studiu comparativ între DL *on-premises* și *cloud* DL cu scopul de a evidenția diferențele majore din punct de vedere al costurilor, al timpilor de implementare, al resurselor hardware și software necesare. Folosind framework-ul Hadoop am realizat implementarea *Single-Node* și *Multi-Node Cluster*, cunoscut fiind faptul că HDFS este una dintre cele mai folosite tehnici de depozitare a datelor Big Data, fie ele local sau în cloud. Am evidențiat diferențele dintre acestea și am subliniat când anume se pretează folosirea acestora în ambientul de lucru. În cea de a doua parte tezei de cercetare am abordat metoda de implementare cloud DL, care a evoluat uluitor de mult în ultimii doi ani, pentru a evidenția care sunt avantajele utilizării unei astfel de tehnologii și totodată pentru a contura stadiul actual al serviciilor pe care furnizorii de cloud le oferă momentan în acest domeniu. S-a observat o evoluție și un interes fantastic pentru această nouă tehnologie, motiv pentru care domeniul este în continuă dezvoltare. Nu toți furnizorii de cloud oferă acest serviciu de stocare de tip DL, dar liderii mondiali din domeniu au reușit cu succes să implementeze și să pună la dispoziția experților un serviciu complet, complex și de mare performanță, ce avansează din ce în ce mai mult datorită puterii sale de stocare și procesare a volumelor mari de date brute și datorită faptului că se poate plia pe orice tip de date și analiză. Practic lacurile de date oferă o capacitata de stocare nelimitată pentru orice tip de date pe care ulterior se pot aplica tehnici de analiză avansată, fie apelând la serviciile built-in fie prin implementarea ad-hoc de noi metode de analiză realizate în diferite limbaje de programare, cu care, pe baza protocoalelor de comunicație, se pot conecta la DL. Am prezentat cei mai importanți furnizori de cloud DL cu scopul de a evidenția serviciile oferite de fiecare, întrucât acestea diferă de la un furnizor la altul.

Alegerea unei tehnologii de implementare a unei arhitecturi DL este un pas important în procesul de proiectare DL, etapă care depinde de o multitudine de factori și totodată de scopul urmărit. Eforturile de cercetare concretizate în prezenta teză, se materializează în următoarele contribuții:

- Am identificat, sintetizat și evidențiat importanța stocării datelor de tip WSAL pe termen lung, în condițiile în care în momentul de față organizațiile sunt constrânse să piardă aceste date datorită creșterii lor exponențiale în volum. Au fost subliniate beneficiile și importanța informațiilor ce pot fi obținute în urma analizei acestor date.
- În urma studiului unei vaste bibliografii, am extras și prezentat elementele tehnice care stau la baza implementării arhitecturii DL propuse în cadrul acestei teze. Este important de menționat faptul că implementarea unui DL depinde de foarte multe aspecte, cum ar fi: tehnica folosită, scopul urmărit, costurile de implementare și de mentenanță, costurile de stocare, securitate, etc. Astfel, implementarea unui DL este originală și unică pentru fiecare scop urmărit. Am considerat imperios să prezint succint elementele tehnice utilizate pentru arhitectura propusă, fiind un domeniu nou ce vine cu un concept diferit de RDBMS.
- Am propus și implementat o arhitectură Data Lake care să deservească procesele de ingestie, stocare și transformare a fișierelor WSAL, care se știe că sunt fișiere text ce

dețin un volum imens de date semi-structurate greu de stocat, procesat și analizat la scară mare.

- Am propus un model de structurare ierarhică a datelor de tip WSAL în nivelul de stocare, ce va deservi întreaga arhitectură DL în diferitele procese ELT cu scopul de a îmbunătăți performanțele de stocare și accesare a datelor.
- Am evidențiat diferitele posibilități de îmbunătățire a structurii ierarhice în funcție de frecvența cu care datele sunt generate.
- Am recomandat soluții de optimizare a costurilor bazat pe tehnici de DLM.
- Pentru ingestia datelor, am proiectat și implementat un proces bazat pe o secvență de comenzi cross-platform utilizând Azure CLI. Prin intermediul comenzilor cross-platform am realizat conexiunea la ADLS Gen2, am creat structura ierarhică de directoare și subdirectoare la nivelul zonei de stocare din cadrul lacului de date și am realizat procesul de încărcare a datelor în DL, oferind astfel o soluție serverless de ingestie a datelor în lacul de date ce poate fi inclusă într-un proces automat batch de transferare a datelor de pe server în DL la intervale regulate de timp.
- Am propus un algoritm pe baza căruia am implementat o funcție serverless *Azure Function* de tip *Blob Trigger* scrisă în limbajul Python care să permită automatizarea procesului de transformare a fișierelor log în fișiere parquet odată ce un fișier nou este încărcat în zona *raw*.
- În urma diferitelor teste efectuate, am evidențiat eficiența și avantajele transformării fișierelor log în fișiere parquet.

În urma cercetărilor privind arhitectura DL am publicat șapte lucrări științifice indexate WOS, IEEE și BDI, dintre care **un articol a fost publicat în jurnalul IEEE Access, ISSN 2169-3536, având un factor de impact de 3.476, aflat în zona galbenă conform ediției JCR 2021 din 28 iunie 2022**.

8.3 Direcții viitoare de cercetare

După cum am văzut, arhitectura DL este un sistem foarte complex alcătuit din mai multe niveluri principale: *Ingestie*, *Stocare*, *Transformare*, *Pregătire & Modelare* și *Explorare & Consum*. În cadrul acestei teze de cercetare, au fost abordate primele trei niveluri din cadrul acestei arhitecturi pentru a realiza ingestia, stocarea și transformarea fișierelor WSAL.

În direcțiile viitoare de cercetare se va urmări studierea și implementarea proceselor specifice nivelurilor din arhitectura DL care nu au fost abordate:

- Pregătire & Modelare: Nivel în cadrul căruia au loc procese de pregătire a datelor provenite din nivelul de Transformare și crearea de modele cu ajutorul motoarelor de analiză avansată;
- Explorare & Consum: Datele derivate din procesele de analiză în cadrul acestui nivel pot fi accesate de utilizatorii finali pentru extragerea de informații specifice scopului urmărit.

Pe fiecare dintre aceste niveluri rulează procese diferite care pot fi implementate fie prin intermediul tehnologiilor integrate furnizate de Microsoft, cum ar fi Azure Synapse Analytics,

Azure HDInsight, Databricks, Power BI, etc. sau prin procese dezvoltate ad-hoc, scrise în diferite limbaje de programare.

Având la bază studiile și lucrările practice efectuate până în prezent intenționez să îmbunătățesc procesul de transformare pentru a îmbogăți datele extrase din fișierele WSAL și să aprofundez diferite procese de analiză avansată a datelor pentru extragerea de diverse rapoarte. Scopul este acela de a demonstra ușurința cu care pot fi interogate datele stocate la nivelul unui DL apelând la serviciile built-in puse la dispoziție de Microsoft. Tehnicile de analiză pot fi foarte variate de la cele mai simple la cele mai complexe, bazate pe tehnologii built-in sau construite ad-hoc în funcție de scopul final urmărit.

8.4 Diseminarea rezultatelor

Contribuțiile originale au fost comunicate și publicate la nivel național și internațional în cadrul revistelor de specialitate, conferințelor și rapoartelor de cercetare, după cum urmează:

Articole publicate în reviste aflate în zona galbenă (Q2):

1. Zagan Elisabeta, Danubianu Mirela, "Data Lake Architecture for Storing and Transforming Web Server Access Log Files," in IEEE Access, vol. 11, pp. 40916-40929, **2023**, **FI 3.476**, doi:10.1109/ACCESS.2023.3270368

Articole publicate în reviste indexate WoS:

2. Zagan Elisabeta, Danubianu Mirela, "From Data Warehouse to a New Trend in Data Architectures - Data Lake", International Journal of Computer Science and Network Security, vol. 19, No. 3, pp. 30-35, **2019**
3. Zagan Elisabeta, Danubianu Mirela, "HADOOP: A Comparative Study between Single-Node and Multi-Node Cluster", International Journal of Advanced Computer Science and Applications (IJACSA), vol. 12, Issue 2, **2021**, doi:10.14569/IJACSA.2021.0120207

Articole prezentate la conferințe indexate WoS și IEEE:

4. Zagan Elisabeta, Danubianu Mirela, "Data Lake Approaches: A Survey", International Conference on Development and Application Systems (DAS), 21-23 mai, **2020**, Suceava, România, pp. 189-193, doi:10.1109/DAS49615.2020.9108912
5. Zagan Elisabeta, Danubianu Mirela, "Cloud DATA LAKE: the new trend of data storage", 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications, 11-13 iunie, **2021,** Ankara, Turcia, doi:10.1109/HORA52670.2021.9461293
6. Zagan Elisabeta, Danubianu Mirela, "ADLS Gen2 for web server log data analysis", 2022 International Conference on Development and Application Systems (DAS), 26-28 mai, **2022**, Suceava, România, pp. 161-166, doi:10.1109/DAS54948.2022.9786071

Articole prezentate la conferințe indexate BDI:

7. Zagan Elisabeta, Danubianu Mirela, "Data block saving policy in the Hadoop Architecture", 16th International Conference on European Integration - Realities and Perspectives, Danubius University, 14-15 mai, **2021**, Galați, România.

Bibliografie

[1] S.-A. Voicu; F. Pop, C. Negru; A. Stoica, „Monitorizarea sistemului și a log-urilor pentru Data Hub Software (DHuS),” Romanian Journal of Information Technology and Automatic Control, ISSN:1220-1758, vol. 30(1), pp. 71-86, 2020. https://doi.org/10.33436/v30i1y202006.

[2] Ł. Korzeniowski; K. Goczyła, „Landscape of Automated Log Analysis: A Systematic Literature Review and Mapping Study,” in IEEE Access, vol. 10, pp. 21892-21913, 2022, doi:10.1109/ACCESS.2022.3152549.

[3] J.M.R.S. Tavares; B.K. Mishra; R. Kumar; N. Zaman; M. Khari, „Handbook of e-Business Security,” 1st ed.. Auerbach Publications, 2018, https://doi.org/10.1201/9780429468254.

[4] Z. Yongjian, „Web Log Analysis Based on Hadoop Technology,” International Conference on Smart Grid and Electrical Automation (ICSGEA), Xiangtan, China, 2019, pp. 587-590, doi:10.1109/ICSGEA.2019.00136.

[5] „Introduction to Azure Data Lake,” „https://dzone.com/articles/introduction-to-azure-data-lake” (Accesat: Octombrie 2019).

[6] J. Dixon, „Pentaho, Hadoop and Data Lakes,” „https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/” (Accesat: Februarie 2020).

[7] H. Fang, „Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem,” IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), Shenyang, China, 2015, pp. 820-824, doi:10.1109/CYBER.2015.7288049.

[8] N. Miloslavskaya; A. Tolstoy, „Application of Big Data, Fast Data, and Data Lake Concepts to Information Security Issues,” 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), Vienna, Austria, 2016, pp. 148-153, doi:10.1109/W-FiCloud.2016.41.

[9] A. Nambiar; D. Mundra, „An Overview of Data Warehouse and Data Lake in Modern Enterprise Data Management,” Big Data Cogn. Comput., 2022, 6, 132. https://doi.org/10.3390/bdcc6040132.

[10] J. Singh; G. Singh; B. S. Bhati, „The Implication of Data Lake in Enterprises: A Deeper Analytics,” 8th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2022, pp. 530-534, doi:10.1109/ICACCS54159.2022.9784986.

[11] H. Mehmood et al., „Implementing Big Data Lake for Heterogeneous Data Sources,” 35th International Conference on Data Engineering Workshops (ICDEW), Macao, China, 2019, pp. 37-44, doi:10.1109/ICDEW.2019.00-37.

[12] J. Pennekamp et al., „Towards an Infrastructure Enabling the Internet of Production,” IEEE International Conference on Industrial Cyber Physical Systems (ICPS), Taipei, Taiwan, 2019, pp. 31-37, doi:10.1109/ICPHYS.2019.8780276.

[13] J. Eder; V. Shekhovtsov, „Data quality for federated medical data lakes,” International Journal of Web Information Systems, 2021. doi:10.1108/IJWIS-03-2021-0026.

[14] F. Yupeng; S. Chinmay, „Real-time Data Infrastructure at Uber,” International Conference on Management of Data (SIGMOD '21, New York, NY, USA, pp. 2503-2516, 2021. https://doi.org/10.1145/3448016.3457552.

[15] C. Haoyang; D. Yucong, „On the Logical Design of a Prototypical Data Lake System for Biological Resources,” Frontiers in Bioengineering and Biotechnology, 2020, 8. 553904. doi:10.3389/fbioe.2020.553904.

[16] M. A. Martínez-Prieto; A. Bregon; I. García-Miranda; P. C. Álvarez-Esteban; F. Díaz; D. Scarlatti, „Integrating flight-related information into a (Big) data lake,” 36th Digital Avionics Systems Conference (DASC), St. Petersburg, FL, USA, 2017, pp. 1-10, doi:10.1109/DASC.2017.8102023.

[17] B. B. Cardoso; S. B. Righetto; E. L. Martins; M. A. Izumida Martins; A. L. Pereira; S. de Francisci, „Data Lake Architecture for Distribution System Operator," IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), Washington, DC, USA, 2021, pp. 1-5. doi:10.1109/ISGT49243.2021.9372181.

[18] W.H. Inmon, „Building the Data Warehouse," 4th ed.; Wiley Publishing: Indianapolis, IN, USA, 2005, ISBN-10 0764599445.

[19] IBM, „What is a data warehouse?," „https://www.ibm.com/topics/data-warehouse" (Accesat: Aprilie 2023)

[20] F. R. S. Paim; J. F. B. de Castro, „DWARF: an approach for requirements definition and management of data warehouse systems," 11th IEEE International Requirements Engineering Conference, Monterey Bay, CA, USA, 2003, pp. 75-84. doi:10.1109/ICRE.2003.1232739.

[21] G. Garani; A. Chernov; I. Savvas; M. A. Butakova, „Data Warehouse Approach for Business Intelligence," 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Napoli, Italia, 12-14 iunie, 2019, pp. 70–75.

[22] V. Gupta; J. Singh, „A Review of Data Warehousing and Business Intelligence in different perspective," Int. J. Comput. Sci. Inf. Technol. 2014, 5, pp. 8263-8268.

[23] E. Zagan; M. Danubianu, „From Data Warehouse to a New Trend in Data Architectures - Data Lake," International Journal of Computer Science and Network Security, Vol. 19, No. 3, pp. 30-35, 2019.

[24] F. Ravat; Y. Zhao, „Data lakes: Trends and perspectives," International Conference on Database and Expert Systems Applications, pp. 304-313, Springer, august, 2019. Cham.

[25] P. Petrova; V. Jotsov; V. Sgurev, „Puzzle Methods for Automatic Selection of Data Cleansing Techniques," International Conference on Intelligent Systems (IS), Funchal - Madeira, Portugal, 2018, pp. 820-826. doi:10.1109/IS.2018.8710580.

[26] „ETL vs ELT – Difference Between Them," „https://www.guru99.com/etl-vs-elt.html" (Accesat: August 2019).

[27] Y. Wang; Y. Li; J. Sui; Y. Gao, „Data Factory: An Efficient Data Analysis Solution in the Era of Big Data," 5th IEEE International Conference on Big Data Analytics (ICBDA), Xiamen, China, 2020, pp. 28-32. doi:10.1109/ICBDA49040.2020.9101284.

[28] M. Chen; S. Mao; Y. Liu, „Big Data: A Survey," Mob. Netw. Appl. 19, 2 (April 2014), pp. 171-209. https://doi.org/10.1007/s11036-013-0489-0

[29] A. Cuzzocrea, „Big Data Lakes: Models, Frameworks, and Techniques," IEEE International Conference on Big Data and Smart Computing (BigComp), Jeju Island, Korea (South), 2021, pp. 1-4, doi:10.1109/BigComp51126.2021.00010.

[30] C. Madera; and A. Laurent, „The next information architecture evolution: the data lake wave," 8th International Conference on Management of Digital EcoSystems (MEDES), ACM, New York, NY, USA, pp. 174-180, 2016. doi:https://doi.org/10.1145/3012071.3012077.

[31] N. Miloslavskaya; A. Tolstoy, „Application of Big Data, Fast Data, and Data Lake Concepts to Information Security Issues," 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), Vienna, 2016, pp. 148-153. doi:10.1109/W-FiCloud.2016.41.

[32] D. E. O'Leary, „Embedding AI and Crowdsourcing in the Big Data Lake," IEEE Intelligent Systems, vol. 29, no. 5, pp. 70-73, Sept.-Oct. 2014. doi:10.1109/MIS.2014.82.

[33] E. Coiera; M. Walther; K. Nguyen; NH. Lovell, „Architecture for knowledge-based and federated search of online clinical evidence," J Med Internet Res. 2005 Oct 24;7(5):e52. doi:10.2196/jmir.7.5.e52. PMID: 16403716; PMCID: PMC1550689.

[34] P.-Y. Hsueh; P. Melville; V. Sindhwani, „Data Quality from Crowdsourcing: A Study of Annotation Selection Criteria," in Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing, pp. 27-35, 2009, Boulder, Colorado.

[35] H. Plattner; A. Zeier, „In-Memory Data Management: Technology and Applications," Springer Science & Business Media, 2012, ISBN 9783642295744.

[36] A. Beheshti; B. Benatallah; R. Nouri; V. M. Chhieng; HT. Xiong; X. Zhao, „CoreDB: a Data Lake Service," in Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM '17). ACM, New York, USA, 2451-2454, 2017, https://doi.org/10.1145/3132847.3133171.

[37] A. A. Munshi; Y. A. I. Mohamed, „Data Lake Lambda Architecture for Smart Grids Big Data Analytics," in IEEE Access, vol. 6, pp. 40463-40471, 2018. doi:10.1109/ACCESS.2018.2858256.

[38] C. Byung; S. Park; J. Kim; SB. Pan; JH. Shin; „International Network Performance and Security Testing Based on Distributed Abyss Storage Cluster and Draft of Data Lake Framework," Security and Communication Networks, pp. 1-14, 2018. doi:10.1155/2018/1746809.

[39] Y. Chen; H. Chen; P. Huang, „Enhancing the data privacy for public data lakes," 2018 IEEE International Conference on Applied System Invention (ICASI), Chiba, 2018, pp. 1065-1068. doi:10.1109/ICASI.2018.8394461.

[40] H. Fang, „Managing data lakes in Big Data era: What's a data lake and why has it became popular in data management ecosystem," 2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), Shenyang, 2015, pp. 820-824. doi:10.1109/CYBER.2015.7288049.

[41] A. Farrugia; R. Claxton; S. Thompson, „Towards social network analytics for understanding and managing enterprise data lakes," IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), San Francisco, CA, 2016, pp. 1213-1220. doi:10.1109/ASONAM.2016.7752393.

[42] K. Pwint; S. W, Zhao. „Data Lake: A New Ideology in Big Data Era," ITM Web of Conferences 17, 03025 (2018), WCSN 2017, At Wuhan, China, ianuarie 2018, doi:10.1051/itmconf/20181703025.

[43] B. Małysiak-Mrozek; M. Stabla; D. Mrozek, "Soft and Declarative Fishing of Information in Big Data Lake," in IEEE Transactions on Fuzzy Systems, vol. 26, no. 5, pp. 2732-2747, octombrie 2018. doi:10.1109/TFUZZ.2018.2812157.

[44] C. Walker; H. Alrehamy, „Personal Data Lake with Data Gravity Pull," IEEE Fifth International Conference on Big Data and Cloud Computing, Dalian, 2015, pp. 160-167. doi:10.1109/BDCloud.2015.62.

[45] D. McCrory, „Data Gravity – in the Clouds," „http://blog.mccrory.me/2010/12/07/data-gravity-in-the-Clouds/", 2014. (Accesat: Octombrie 2020)

[46] „Data Lakes in a Modern Data Architecture," eBook by BlueGranite (Accesat: Noiembrie 2018).

[47] M. Hajdarbegovic, „Data Lake Architecture: A Comprehensive Guide," „https://virtasant.com/blog/data-lake-architecture" (Accesat: Noiembrie 2020)

[48] B. Sharma, „Architecting Data Lakes," - second edition , Aprilie 2018, Published by O'Reilly Media, ISBN: 9781492032991.

[49] N. Feldmann, „Data Lake or Data Swamp?", „https://www.nvisia.com/insights/data-swamp" (Accesat: Septembrie 2021)

[50] A. Adshead, „Data lake storage: Cloud vs on-premise data lakes," „https://www.computerweekly.com/feature/Data-lake-storage-Cloud-vs-on-premise-data-lakes" (Accesat: Martie 2021)

[51] Dremio, „Introduction to Data Lakes," „https://www.dremio.com/data-lake/cloud-data-lakes/" (Accesat: Martie 2021)

[52] A. Woodie, „Back to Basics: Big Data Management in the Hybrid, Multi-Cloud World," „https://www.datanami.com/2021/06/07/back-to-basics-big-data-management-in-the-hybrid-multi-cloud-world/" (Accesat: Aprilie 2021)

[53] H. Li, „Why Build a Hybrid Data Lake for Analytics and AI," „https://www.dataversity.net/why-build-a-hybrid-data-lake-for-analytics-and-ai/.da (Accesat: Aprilie 2021)

[54] HDFS Tutorial, „10 Best Hadoop Cloud Service Providers," „https://www.hdfstutorial.com/blog/hadoop-cloud-service-providers-2/.da" (Accesat: Martie 2021)

[55] S. Ghemawat; H. Gobioff; S.-T. Leung, „The Google file system," In Proceedings of the nineteenth ACM symposium on Operating systems principles (SOSP '03). Association for Computing Machinery, New York, NY, USA, pp. 29-43, 2003. „https://doi.org/10.1145/945445.945450".

[56] J. Dean; S. Ghemawat. 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51, 1 (January 2008), pp. 107-113. „https://doi.org/10.1145/1327452.1327492".

[57] G. Turkington; G. Modena, „Big data con Hadoop," Apogeo, 2015. „https://www.unilibro.it/libro/turkington-garrymodena-gabriele/big-data-con-hadoop/9788850333431" (Accesat: Ianuarie 2021)

[58] S. Landset; T.M. Khoshgoftaar; A.N. Richter; et al. „A survey of open source tools for machine learning with big data in the Hadoop ecosystem," Journal of Big Data 2, 24 (2015). https://doi.org/10.1186/s40537-015-0032-1.

[59] „Apache Hadoop," „https://hadoop.apache.org/" (Accesat: Ianuarie 2019)

[60] M. K. Gupta; S. K. Pandey; A. Gupta, „HADOOP- An Open Source Framework for Big Data," 2022 3rd International Conference on Intelligent Engineering and Management (ICIEM), London, United Kingdom, 2022, pp. 708-711, doi:10.1109/ICIEM54221.2022.9853179.

[61] R.C.M. Correia; G. Spadon; P.H. De Andrade Gomes; D.M. Eler; R.E. Garcia; C. Olivete Junior, „Hadoop Cluster Deployment: A Methodological Approach," Information 2018, 9, 131.

[62] C. Uzunkayaa; T.a Ensaria; Y. Kavurucub, „Hadoop Ecosystem and Its Analysis on Tweets," Procedia - Social and Behavioral Sciences, Volume 195, 3 July 2015, pp. 1890-1897.

[63] „Hadoop Training Program," „https://www.educba.com/" (Accesat: Ianuarie 2021)

[64] C. Roy; M. Pandey; S. SwarupRautaray, „A Proposal for Optimization of Data Node by Horizontal Scaling of Name Node Using Big Data Tools," 2018 3rd International Conference for Convergence in Technology (I2CT), Pune, India, 2018, pp. 1-6, doi:10.1109/I2CT.2018.8529795.

[65] T. White, „MapReduce and the hadoop distributed file system," in Hadoop: The definitive guide, 1st edition, O'Reilly Media, Inc., Yahoo press, 2012

[66] A. Shah; M. Padole, „Apache Hadoop: A Guide for Cluster Configuration & Testing. International Journal of Computer Sciences and Engineering. 7. pp. 792-796, 2019. doi:10.26438/ijcse/v7i4.792796

[67] O. Prem, „Installation and Configuration Documentation," Sep 2017, „https://readthedocs.org/projects/doctuts/downloads/pdf/latest/" (Accesat: Decembrie 2020)

[68] V. G. Anisha Gnana Vincy; T. Karthija; J. Sunil, „Understanding Hadoop Framework Through Single-Node Cluster Installation," 2019 International Conference on Recent Advances in Energy-efficient Computing and Communication (ICRAECC), Nagercoil, India, 2019, pp. 1-4, doi:10.1109/ICRAECC43874.2019.8995060.

[69] E. Sammer, „Hadoop Operations," September 2012, Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

[70] „What is New in Hadoop 3? Explore the Unique Hadoop 3 Features," „https://data-flair.training/blogs/what-is-new-in-hadoop-3/" (Accesat: Martie 2021)

[71] K. Koitzsch, „Pro Hadoop Data Analytics: Designing and Building Big Data Systems Using the Hadoop Ecosystem," USA, Apress, 2016, Book, ISBN-10: 9781484219096, ISB-13: 978-1484219096.

[72] G. S. Bhathal; A. S. Dhiman, „Big Data Solution: Improvised Distributions Framework of Hadoop," 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2018, pp. 35-38, doi:10.1109/ICCONS.2018.8663142.

[73] „https://beyondcorner.com/learn-apache-hadoop/hadoop-1x-vs-hadoop-2x-and-hadoop-2x-vs-hadoop-3x/" (Accesat: Noiembrie 2021)

[74] „Hadoop2OnWindows,” „https://cwiki.apache.org/confluence/display/HADOOP2/Hadoop2OnWindows” (Accesat: Septembrie 2021)

[75] Flexera. „Flexera 2021 State of the Cloud Report,” 2022. Available: „https://info.flexera.com/CM-REPORT-State-of-the-Cloud” (Accesat: Ianuarie 2023)

[76] W. McKnight, „Top Trends in Data Lakes,” „https://www.informationweek.com/big-data/top-trends-in-data-lakes/a/d-id/1338651? ” (Accesat: Februarie 2021)

[77] „The Definitive Guide to Cloud Storage Pricing,” „https://www.hubstor.net/blog/the-definitive-guide-to-cloud-storage-pricing/” (Accesat: Aprilie 2021)

[78] A. Laurent; D Laurent; C. Madera, „Data Lakes,” Vol. 2, 2020, ISBN 978-1-78630-585-5

[79] P. Pierleoni; R. Concetti; A. Belli; L. Palma, „Amazon, Google and Microsoft Solutions for IoT: Architectures and a Performance Comparison,” in IEEE Access, vol. 8, pp. 5455-5470, 2020, doi:10.1109/ACCESS.2019.2961511.

[80] X. Liyuan; Q. Li; C. Zhijun; W. Zhenxin, „The Bidirectional Data Flow Based On The Data-Lake,” 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), pp. 971-976, October 2020. doi:10.1109/CISP-BMEI51763.2020.9263496.

[81] „Azure Data Lake Storage pricing,” „https://azure.microsoft.com/en-us/pricing/details/storage/data-lake/” (Accesat: Februarie 2023)

[82] „Ingestion and Processing Layers in Azure Data Lakehouse,” „https://www.mssqltips.com/sqlservertip/7037/azure-data-lakehouse-ingestion-processing-options/” (Accesat: Decembrie 2021)

[83] A. Tunjic, „The Automation of the Data Lake Ingestion Process from Various Sources”, 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1276-1281, mai 2019. doi:10.23919/MIPRO.2019.8756864.

[84] G. Saurabh; G. Venkata, „Practical Enterprise Data Lake Insights,” 2018, ISBN-13 (pbk): 978-1-4842-3521-8

[85] „Load Data Into Azure Data Lake Storage Gen2 With Azure Data Factory,“ „https://docs.microsoft.com/enus/azure/data-factory/load-azure-data-lake-storage-gen2” (Accesat: Ianuarie 2022)

[86] „Best practices for using Azure Data Lake Storage Gen2,” „https://docs.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-best-practices” (Accesat: Ianuarie 2022)

[87] C. Melissa, „Azure Data Lake Storage Gen2: 10 things you need to know,” „https://www.bluegranite.com/blog/10-things-to-know-about-azure-data-lake-storage-gen2” (Accesat: Noiembrie 2021)

[88] T. J. Skluzacek; K. Chard; I. Foster, „Klimatic: A Virtual Data Lake for Harvesting and Distribution of Geospatial Data,” 2016 1st Joint International Workshop on Parallel Data Storage and data Intensive Scalable Computing Systems (PDSW-DISCS), pp. 31-36, noiembrie 2016. doi:10.1109/PDSW-DISCS.2016.010.

[89] S. Tovernic; V. Banovic; Z. Hrastic; K. Plantic; A. Šandić; M. Baranovic, „Solution for detecting sensitive data inside a data lake,” 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1284-1288, mai 2018. doi:10.23919/MIPRO.2018.8400232.

[90] N. Lynn, „How to Best Design a Data Lake Storage?,” „https://www.altisconsulting.com/au/insights/how-to-best-design-a-data-lake-storage/” (Accesat: Martie 2022)

[91] N. Hurt, „Building your Data Lake on Azure Data Lake Storage gen2,” https://medium.com/microsoftazure/building-your-data-lake-on-adls-gen2-3f196fc6b430 (Accesat: Ianuarie 2022)

[92] „Key considerations in designing your data lake,” „https://azure.github.io/Storage/docs/analytics/hitchhikers-guide-to-the-Data Lake/?WT.mc_id=helloworld-17228-cxa#key-considerations-in-designing-your-data-lake” (Accesat: Octombrie 2021)

[93] „Configure a lifecycle management policy,” „https://docs.microsoft.com/en-us/azure/storage/blobs/lifecycle-management-policy-configure?tabs=azure-portal” (Accesat: Octombrie 2021)

[94] „Optimize costs by automatically managing the data lifecycle,” „https://docs.microsoft.com/en-us/azure/storage/blobs/lifecycle-management-overview” (Accesat: Octombrie 2021)

[95] H. B. Hamadou; T. Bach Pedersen; C. Thomsen, “The Danish National Energy Data Lake: Requirements, Technical Architecture, and Tool Selection,” IEEE International Conference on Big Data (Big Data), pp. 1523-1532, decembrie 2020. doi:10.1109/BigData50022.2020.9378368.

[96] „Access control model in Azure Data Lake Storage Gen2,” „https://docs.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-access-control-model” (Accesat: Ianuarie 2022)

[97] „Azure Storage redundancy,” „https://docs.microsoft.com/en-us/azure/storage/common/storage-redundancy” (Accesat: Decembrie 2021)

[98] S. Hernández; P. Álvarez; J. Fabra; J. Ezpeleta, „Analysis of Users’ Behavior in Structured e-Commerce Websites,” in IEEE Access, vol. 5, pp. 11941-11958, 2017, doi:10.1109/ACCESS.2017.2707600.

[99] J. M. P. Jeba; M. S. Bhuvaneswari; K. Muneeswaran, „Extracting usage patterns from web server log,” 2nd International Conference on Green High Performance Computing (ICGHPC), Nagercoil, India, 2016, pp. 1-7, doi:10.1109/ICGHPC.2016.7508074.

[100] P. Ghavare; P. Ahire, „Big Data Classification of Users Navigation and Behavior Using Web Server Logs,” Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 2018, pp. 1-6, doi:10.1109/ICCUBEA.2018.8697606.

[101] „Server log files in a nutshell,” „https://www.graylog.org/post/server-log-files-in-a-nutshell” (Accesat: Octombrie 2021)

[102] B. Plejic; B. Vujnovic; R. Penco, „Transforming unstructured data from scattered sources into knowledge,” IEEE International Symposium on Knowledge Acquisition and Modeling Workshop, Wuhan, China, 2008, pp. 924-927, doi:10.1109/KAMW.2008.4810643.

[103] „Hadoop is Data’s Darling for a Reason,” „https://www.forrester.com/blogs/hadoop-is-datas-darling-for-areason/” (Accesat: Noiembrie 2022)

[104] „The Unseen Data Conundrum,” „https://www.forbes.com/sites/forbestechcouncil/2022/02/03/the-unseendata-conundrum/?sh=4a07b7ac7fcc” (Accesat: Noiembrie 2022)

[105] D. S. Sisodia; S. Verma. „Web usage pattern analysis through web logs: A review,” Ninth International Conference on Com-puter Science and Software Engineering (JCSSE), Bangkok, 2012 (pp. 49-53).

[106] http://www.cheat-sheets.org/saved-copy/http-response-codes-1.pdf” (Accesat: Noiembrie 2021)

[107] „Ultimate guide log file analysis,” „https://en.ryte.com/magazine/ultimate-guide-log-file-analysis” (Accesat: Octombrie 2021)

[108] G. Amran; H. Aldheleai; H. Al-Sanabani, „Understanding the Classification of Data Mining and Web Mining,” June, 2021.

[109] S. Kundu; L. Garg, “Web log analyzer tools: A comparative study to analyze user behavior,” 2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence, 2017, pp. 17-24, doi:10.1109/CONFLUENCE.2017.7943117

[110] Z. Farzin, „Online Shopping Store-Web Server Logs,” „http://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/3QBYB5” (Accesat: Septembrie 2021)

[111] „Come leggere il log del web server”, https://www.evemilano.com/come-leggere-il-log-del-web-server/” (Accesat: Noiembrie 2021)

[112] „Exercise - Create an Azure Storage account for Azure Data Lake Storage,” „https://docs.microsoft.com/en-us/learn/modules/upload-data-to-azure-data-lake-storage/2-create-azure-data-lake” (Accesat: Ianuarie 2022)

[113] „Secure your Azure Storage account,” „https://docs.microsoft.com/en-us/learn/modules/secure-azure-storage-account/1-introduction” (Accesat: Decembrie 2021)

[114] „Quickstart: Create a function in Azure with Python using Visual Studio Code,” „https://docs.microsoft.com/en-us/azure/azure-functions/create-first-function-vs-code-python” (Accesat: Ianuarie 2022)

[115] „pandas.DataFrame,” „https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html” (Accesat: Decembrie 2021)

[116] „Azure Blob storage bindings for Azure Functions overview,” „https://learn.microsoft.com/en-us/azure/azure-functions/functions-bindings-storage-blob?tabs=in-process%2Cfunctionsv2%2Cextensionv3&pivots=programming-language-python# Trigger---polling” (Accesat: August 2022)

[117] „Azure Data Lake Storage Pricing,” „https://azure.microsoft.com/en-us/pricing/details/storage/datalake/” (Accesat: Februarie 2023)

ANEXA I. Hadoop 1 Single-Node și Multi-Node Cluster pentru testele de laborator

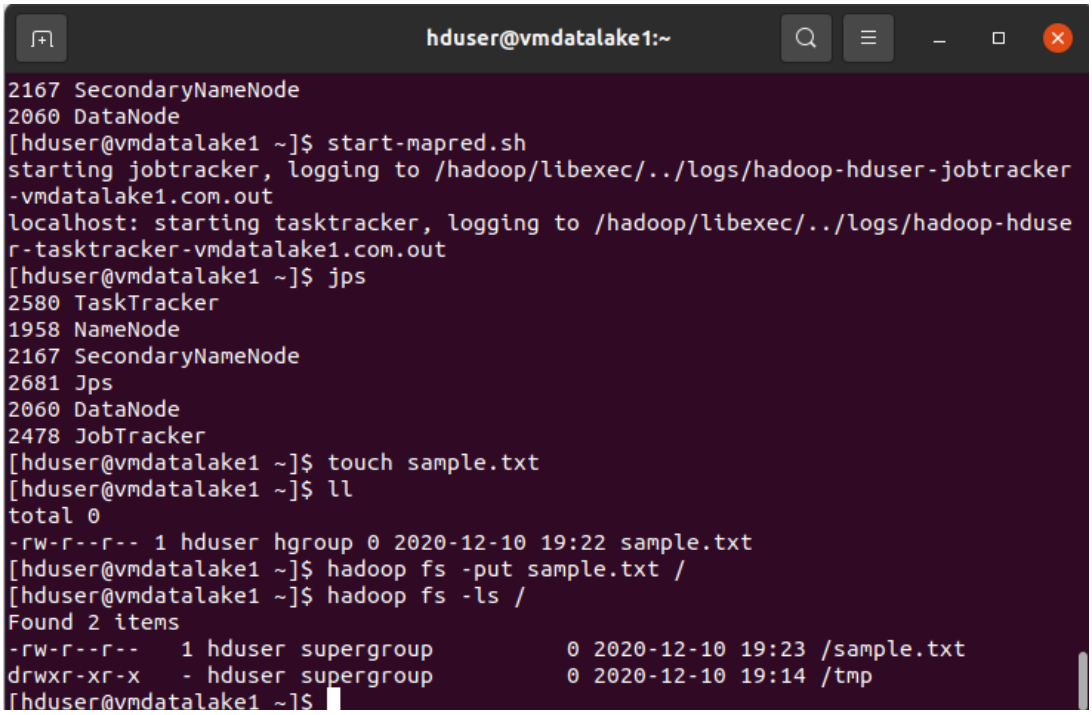


Figura I-1: Hadoop Single-Node Cluster

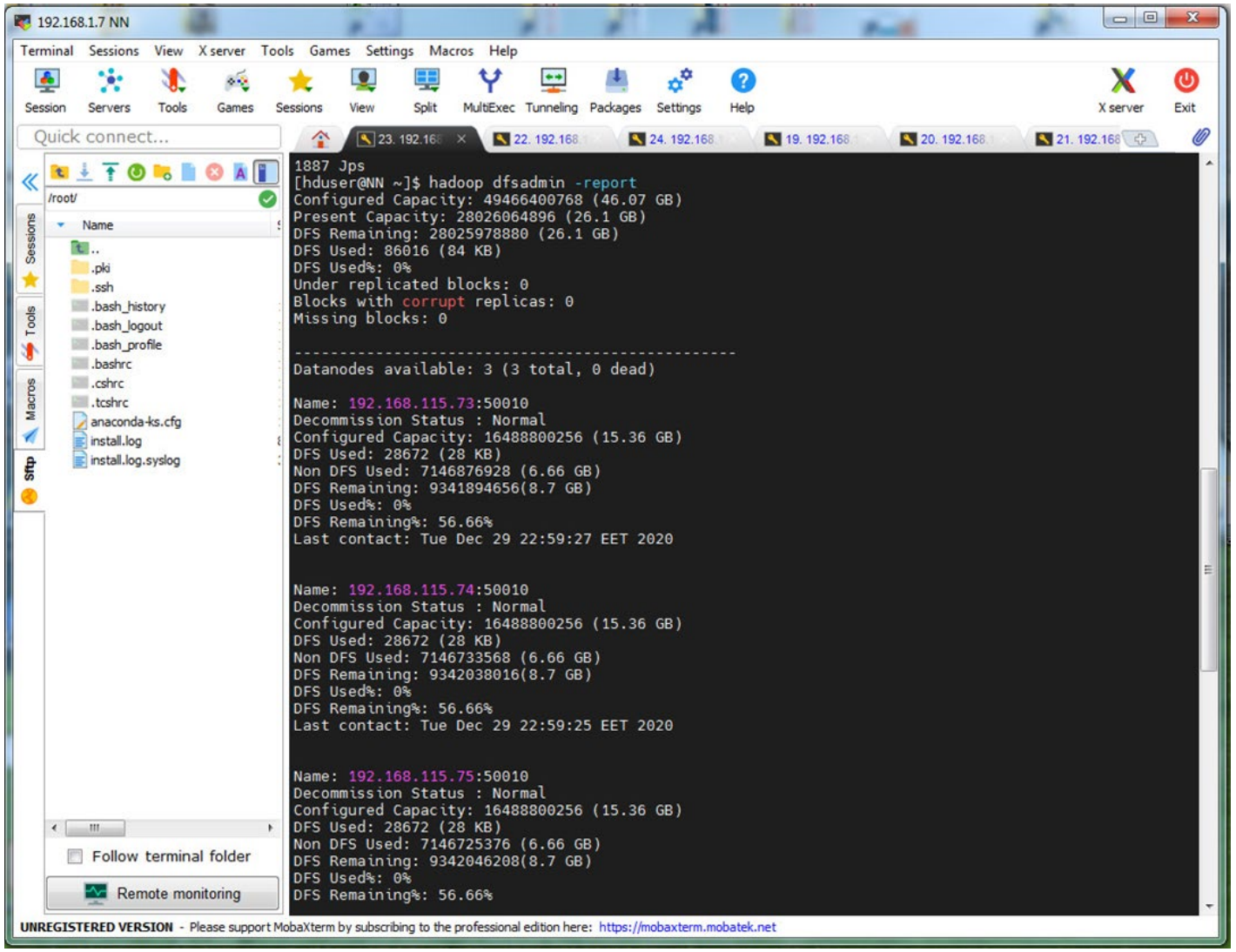


Figura I-2: Hadoop Multi-Node Cluster – MobaXterm

ANEXA II. Proiectul implementat în VSCode pentru Azure Blob Trigger Function

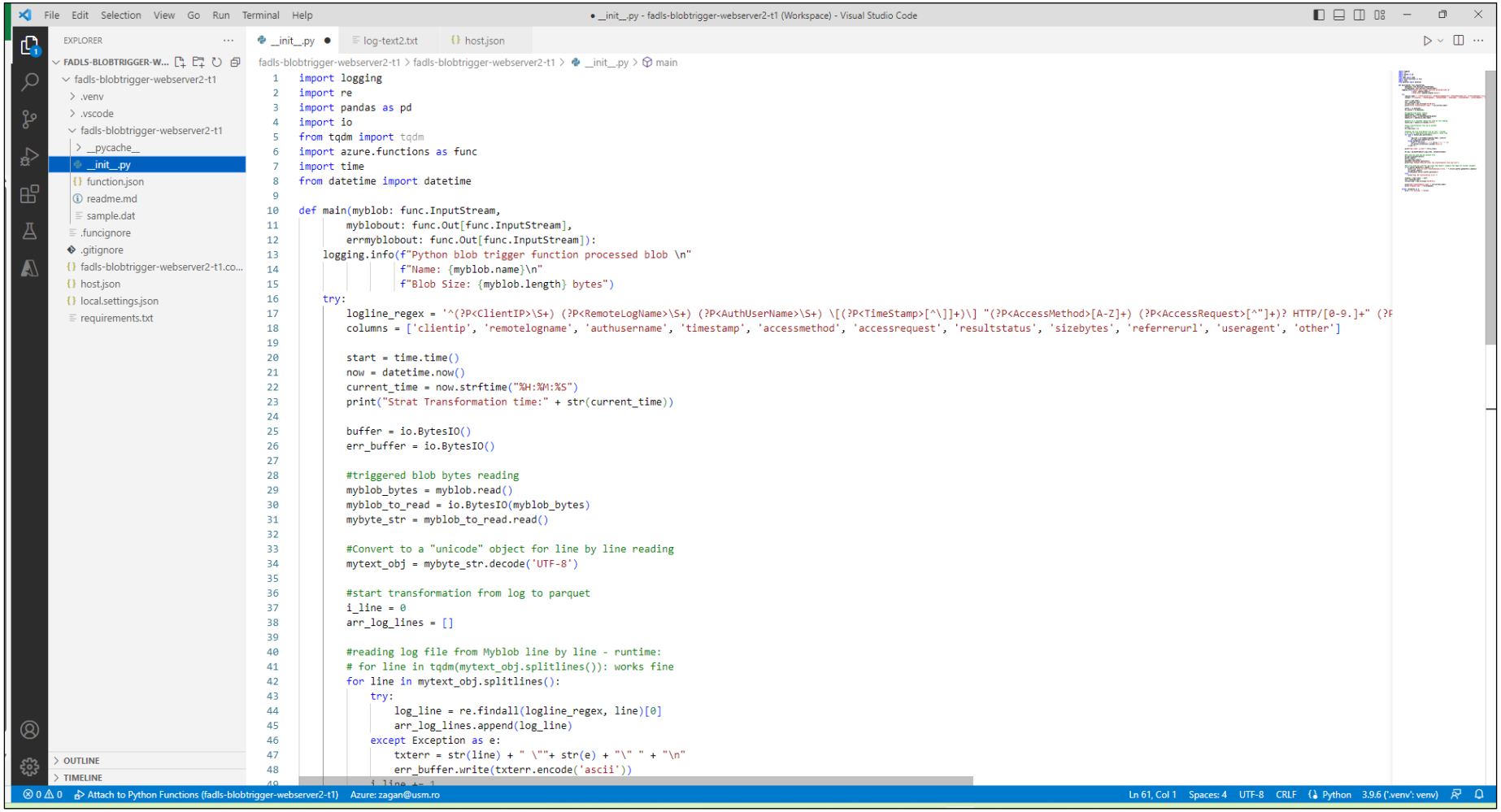


Figura II-1.a: Funcția principală - Azure Blob Trigger

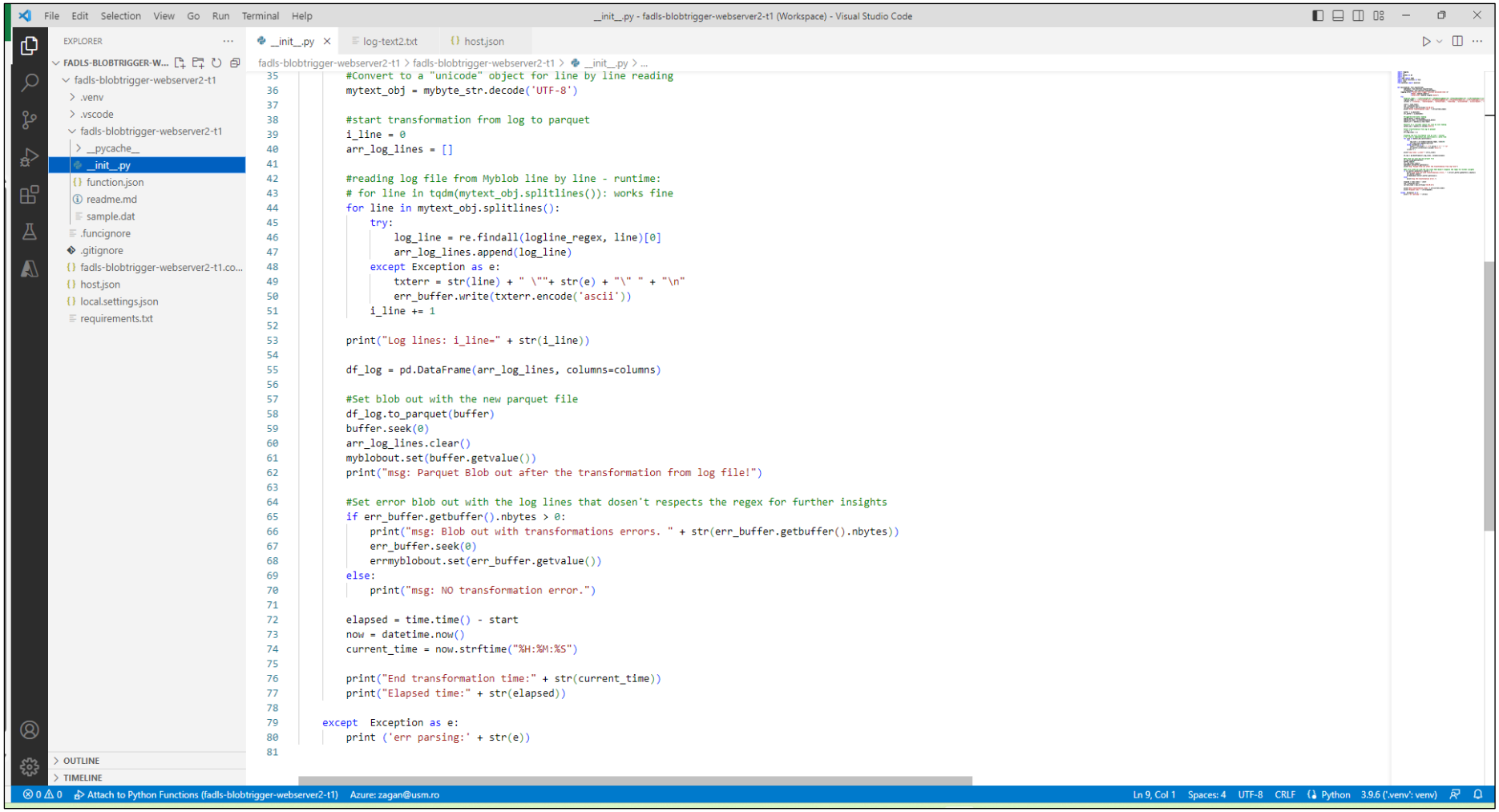


Figura II-1.b: Funcția principală - Azure Blob Trigger

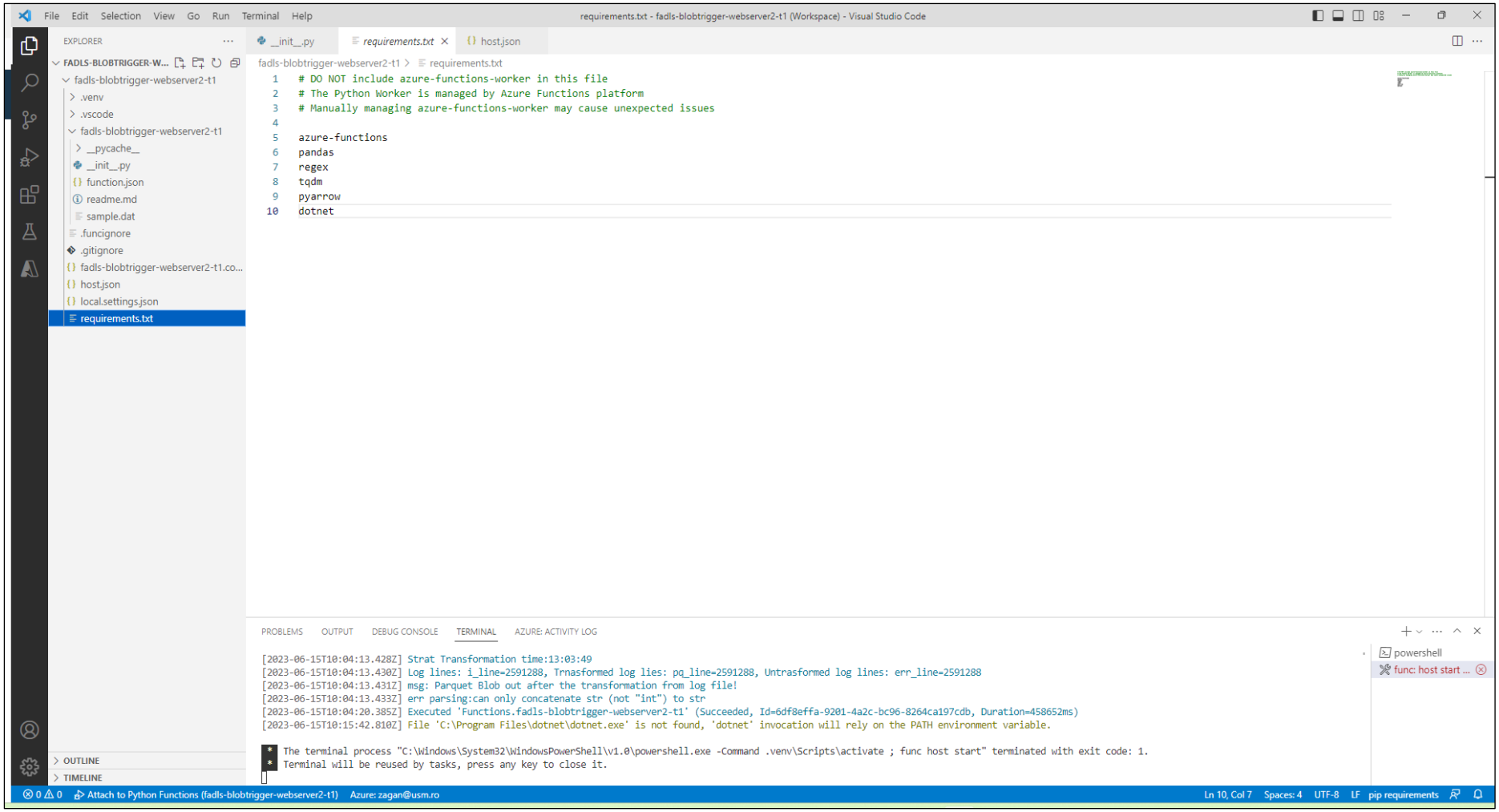


Figura II-1.c: Fișierul cu librării

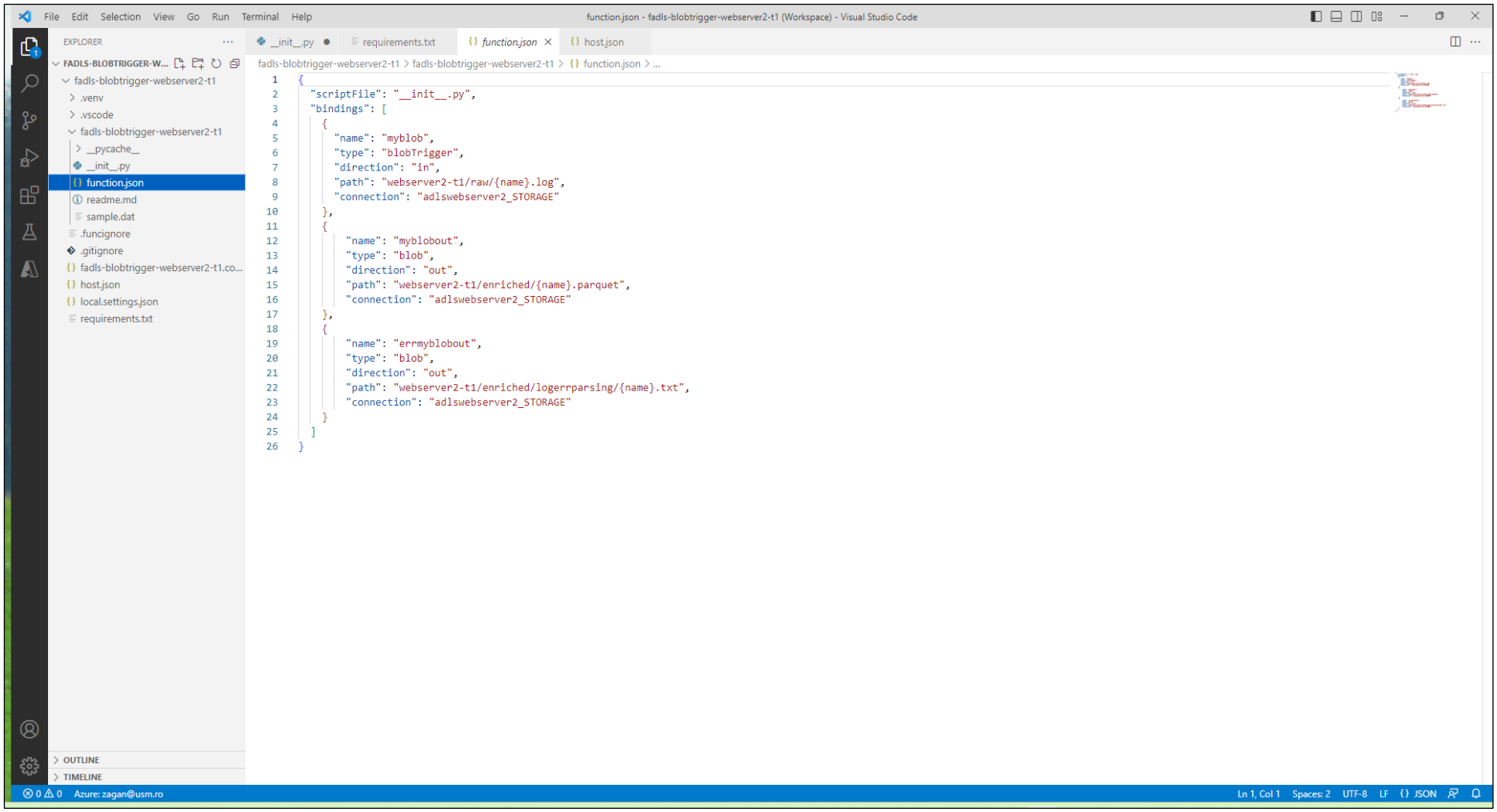


Figura II-1.d: Funcția Bindings - Azure Blob Trigger

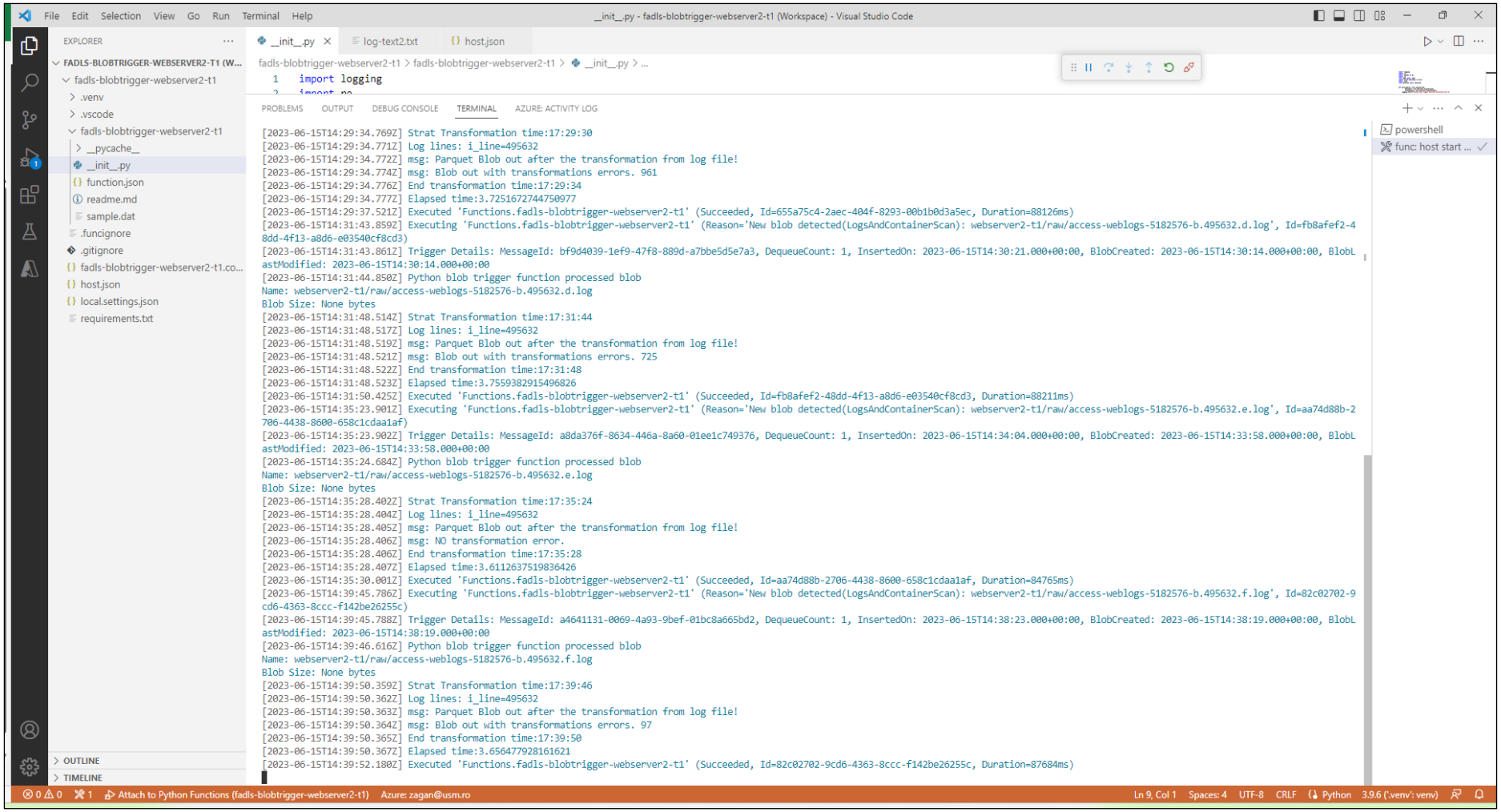


Figura II-2.a: Execuția funcției Blob Trigger pentru transformarea fișierelor log în fișiere parquet cu sistemul P1
(enriched_access-weblogs-5182576-b.495632.d.parquet, enriched_access-weblogs-5182576-b.495632.e.parquet, enriched_access-weblogs-5182576-b.495632.f.parquet)

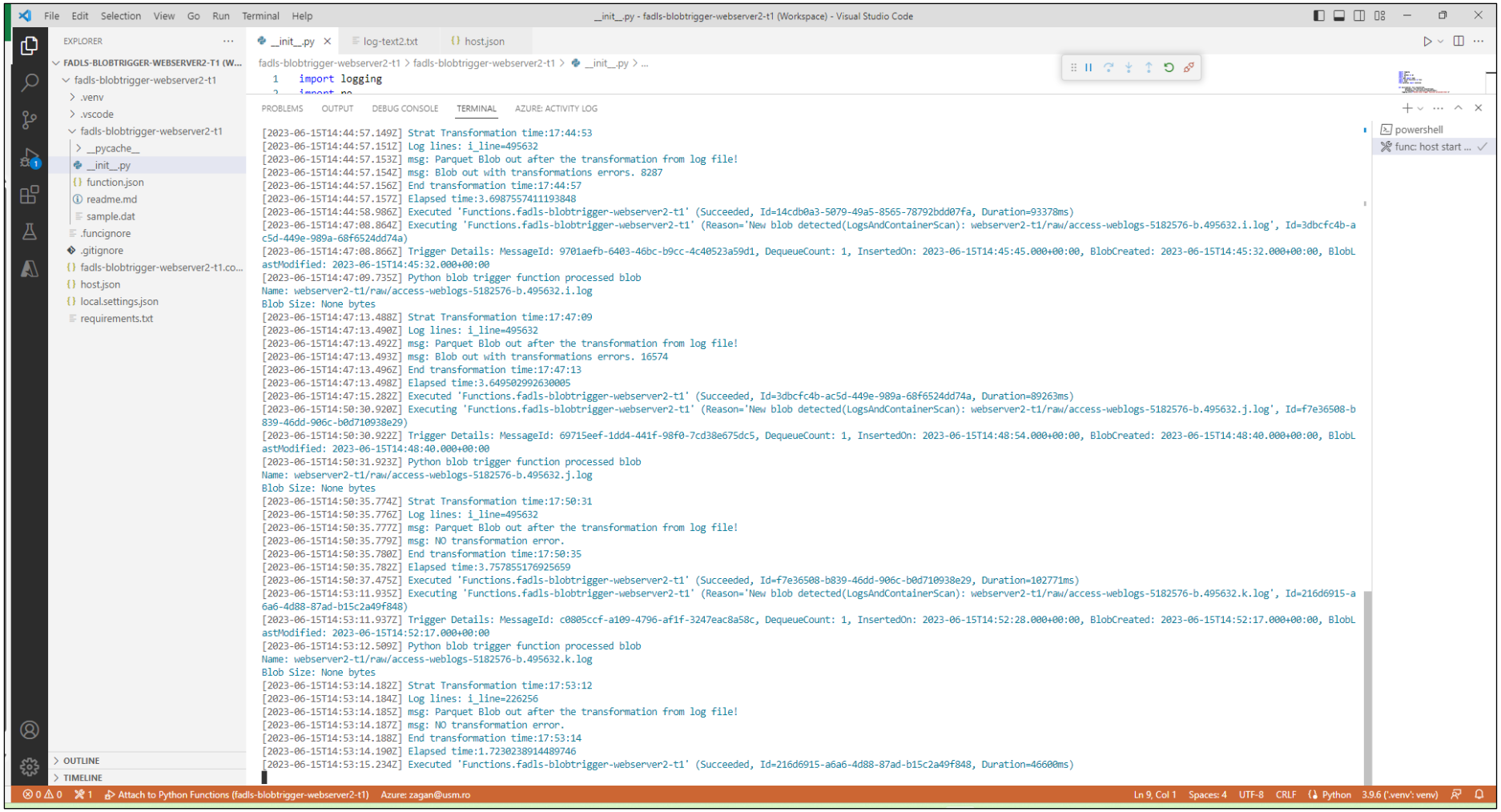
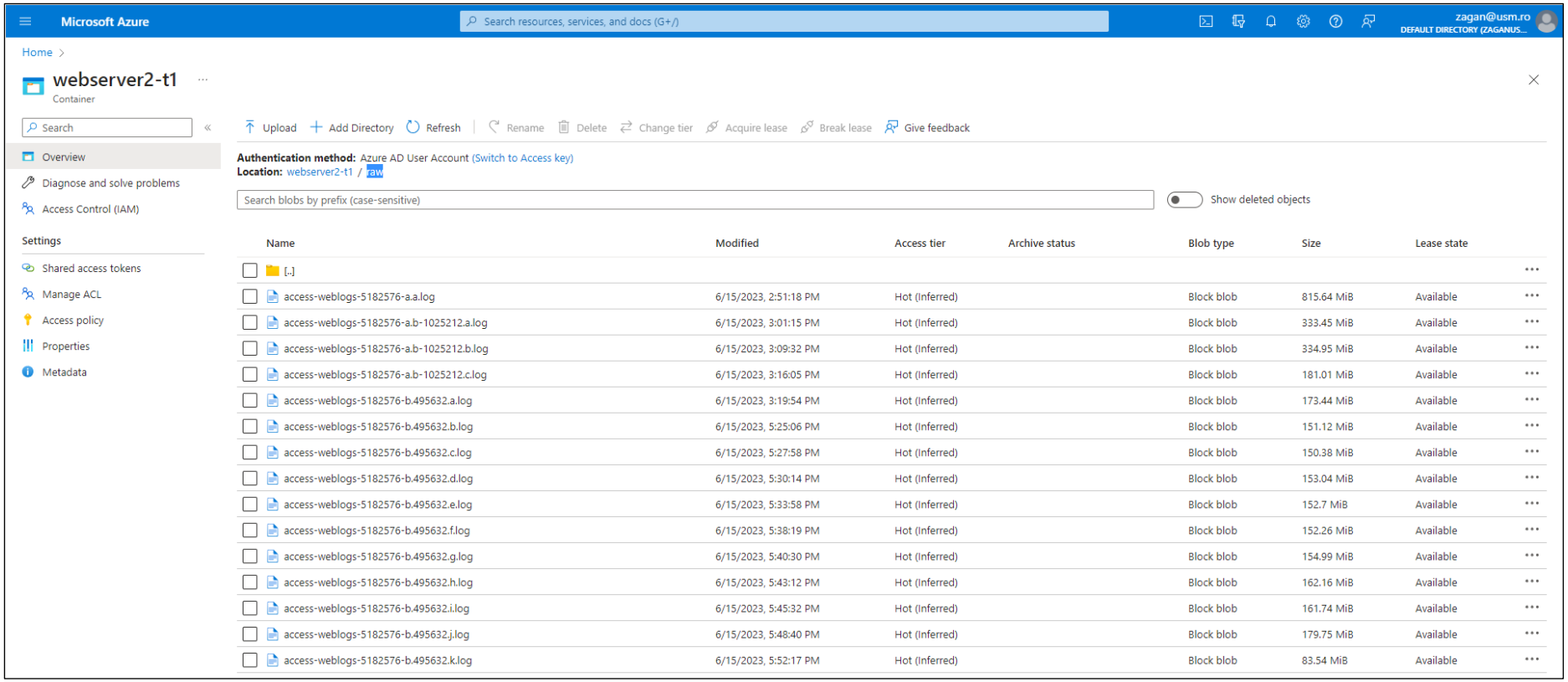


Figura II-2.b: Execuția funcției Blob Trigger pentru transformarea fișierelor log în fișiere parquet cu sistemul P2
(enriched_access-weblogs-5182576-b.495632.i.parquet, enriched_access-weblogs-5182576-b.495632.j.parquet, enriched_access-weblogs-5182576-b.495632.k.parquet)

ANEXA III. Stocarea datelor WSAL, parquet și text în ADLS Gen2



Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
[..]						
access-weblogs-5182576-a.a.log	6/15/2023, 2:51:18 PM	Hot (Inferred)		Block blob	815.64 MiB	Available
access-weblogs-5182576-a.b-1025212.a.log	6/15/2023, 3:01:15 PM	Hot (Inferred)		Block blob	333.45 MiB	Available
access-weblogs-5182576-a.b-1025212.b.log	6/15/2023, 3:09:32 PM	Hot (Inferred)		Block blob	334.95 MiB	Available
access-weblogs-5182576-a.b-1025212.c.log	6/15/2023, 3:16:05 PM	Hot (Inferred)		Block blob	181.01 MiB	Available
access-weblogs-5182576-b.495632.a.log	6/15/2023, 3:19:54 PM	Hot (Inferred)		Block blob	173.44 MiB	Available
access-weblogs-5182576-b.495632.b.log	6/15/2023, 5:25:06 PM	Hot (Inferred)		Block blob	151.12 MiB	Available
access-weblogs-5182576-b.495632.c.log	6/15/2023, 5:27:58 PM	Hot (Inferred)		Block blob	150.38 MiB	Available
access-weblogs-5182576-b.495632.d.log	6/15/2023, 5:30:14 PM	Hot (Inferred)		Block blob	153.04 MiB	Available
access-weblogs-5182576-b.495632.e.log	6/15/2023, 5:33:58 PM	Hot (Inferred)		Block blob	152.7 MiB	Available
access-weblogs-5182576-b.495632.f.log	6/15/2023, 5:38:19 PM	Hot (Inferred)		Block blob	152.26 MiB	Available
access-weblogs-5182576-b.495632.g.log	6/15/2023, 5:40:30 PM	Hot (Inferred)		Block blob	154.99 MiB	Available
access-weblogs-5182576-b.495632.h.log	6/15/2023, 5:43:12 PM	Hot (Inferred)		Block blob	162.16 MiB	Available
access-weblogs-5182576-b.495632.i.log	6/15/2023, 5:45:32 PM	Hot (Inferred)		Block blob	161.74 MiB	Available
access-weblogs-5182576-b.495632.j.log	6/15/2023, 5:48:40 PM	Hot (Inferred)		Block blob	179.75 MiB	Available
access-weblogs-5182576-b.495632.k.log	6/15/2023, 5:52:17 PM	Hot (Inferred)		Block blob	83.54 MiB	Available

Figura III-1: Zona de date *raw* din DL

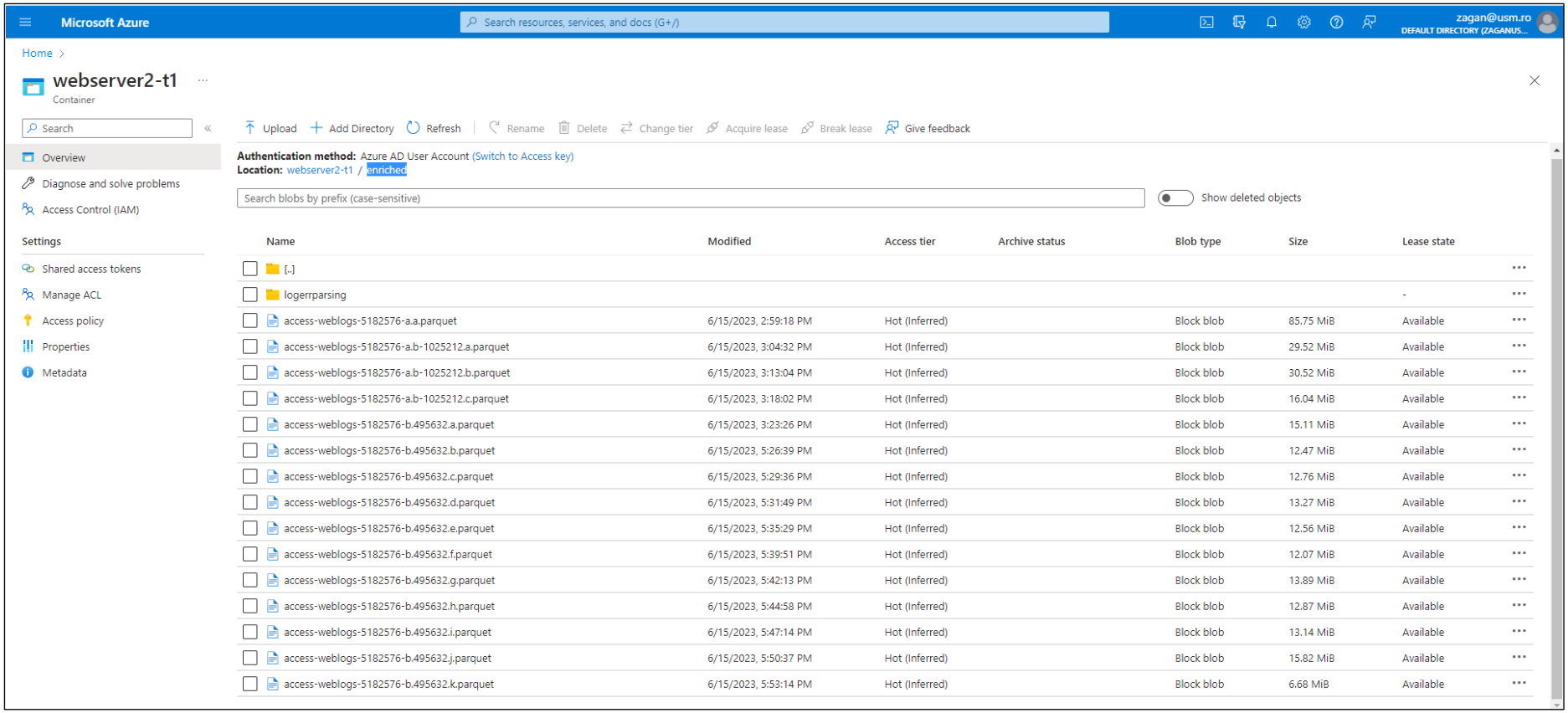


Figura III-2: Zona de date *enriched* din DL

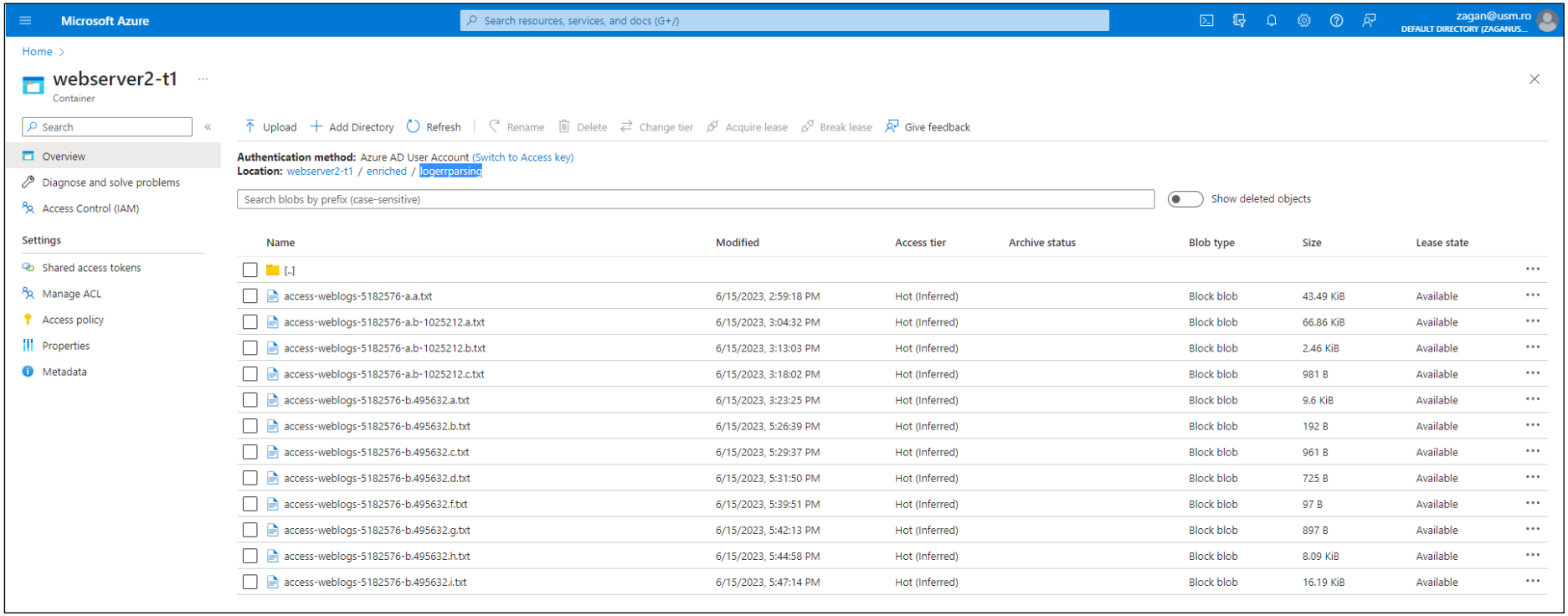


Figura III-3: Zona de date *enriched/logerrparsing* din DL