Skladište podataka utemeljeno na jezeru podataka

Batnožić, Ivan

Sveučilište u Zagrebu

Ekonomski fakultet

Menadžerska informatika

Skladište podataka utemeljeno na jezeru podataka

Diplomski rad

Ivan Batnožić

Zagreb, Lipanj, 2023.

Sveučilište u Zagrebu

Ekonomski fakultet

Menadžerska informatika

Skladište podataka utemeljeno na jezeru podataka
Eng. Data Lakehouse

Diplomski rad

Izradio: Ivan Batnožić, JMBAG: 0067552307
Mentor: prof. dr. sc. Katarina Ćurko

Zagreb,Lipanj,2023.

Ivan Batnožić

IZJAVA O AKADEMSKOJ ČESTITOSTI

Izjavljujem i svojim potpisom potvrđujem da je diplomski rad isključivo rezultat mog vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu, a što pokazuju korištene bilješke i bibliografija. Izjavljujem da nijedan dio rada nije napisan na nedozvoljen način, odnosno da je prepisan iz necitiranog rada, te da nijedan dio rada ne krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

Student:

U Zagrebu,_____ _____

(potpis)

Sažetak

Predmet rada su skladišta podataka utemeljena na jezerima podataka. Cilj rada je uz maksimalnu fokusiranost, na tehnički jednostavnim primjerima, jasno i razvidno predstaviti sve izvanredne glavne značajke takvih sustava. Izlaganje je utemeljeno konceptima V dimenzija Velike količine podataka, modelu računalstva u oblaku i svakako na cjelovitom pregledu relevantnih arhitektura - tradicionalnih skladišta podataka, skladišta podataka u oblaku, jezera podataka i modernih skladišta podataka.

Nakon toga su konceptualno obrađeni, u arhitekturnom i funkcionalnom smislu, temeljni elementi potrebni za implementaciju kao što su strukturiranje jezera podataka po medallion arhitekturi, analitički program Apache Spark i Delta Lake format podataka.

U praktičnom dijelu rada korišteni su alati otvorenog koda DBeaver, JupyterLab, Weka i RStudio. Sukladno čestoj praksi u stručnoj literaturi sustav je implementiran kao instalacija na lokalnom računalu radi tehničke jednostavnosti u segmentu svih podešavanja parametara sustava, kao i detaljnosti uvida u način njegovog funkcioniranja. Uvid u način funkcioniranja je iznimno važan kako bi se maksimalno iskoristio temeljni princip distribuirane obrade podataka, prisutne u svakom detalju funkcioniranja sustava.

U praktičnom smislu ustanovljena je velika praktičnost primjene sustava u segmentu poslovne analize zbog velikih mogućnosti uporabe SQL-a u kombinaciji sa Delta Lake formatom podataka. To je konkretno demonstrirano uporabom alata DBeaver, JupyterLab, Weka i RStudio, na skupovima podataka tržišta kapitala u strukturiranom, polu strukturiranom i nestrukturiranom obliku.

Sve brojne pozitivne značajke ispitanog koncepta ukazuju da će imati značajan utjecaj općenito na razvoj arhitektura poslovnih aplikacija u budućnosti.

Ključne riječi: Velika količina podataka, skladište podataka, jezero podataka, Apache Spark, Delta Lake, skladište podataka utemeljeno na jezeru podataka

Summary

The subject of the work are data lakehouses. The aim of the paper is to present all the outstanding main features of such systems clearly and clearly, with maximum focus, using technically simple examples. The presentation is based on the concepts of the V dimensions of Big data, the cloud computing model and certainly on a comprehensive overview of the relevant architectures - traditional data warehouses, cloud data warehouses, data lakes and modern data warehouses.

After that, the fundamental elements needed for the implementation, such as structuring of the data lake according to the medallion architecture, the analytical program Apache Spark and the Delta Lake data format, were conceptually processed, in an architectural and functional sense. In the practical part of the work, the open source tools DBeaver, JupyterLab, Weka and RStudio were used. In accordance with the frequent practice in professional literature, the system is implemented as an installation on a local computer for technical simplicity in the segment of all system parameter settings, as well as detailed insight into the way it functions. Insight into the way it functions is extremely important in order to make the most of the basic principle of distributed data processing, present in every detail of the system's functioning.

In a practical sense, the great practicality of using the system in the segment of business analysis was established due to great possibilities of using SQL in combination with Delta Lake data format. This was concretely demonstrated by using DBeaver, JupyterLab, Weka and RStudio tools, on capital market data sets in structured, semi-structured and unstructured form. All the numerous positive features of the examined concept indicate that it will have a significant impact in general on the development of business application architectures in the future.

Key words: Big Data, Data Warehouse , Data Lake, Apache Spark, Delta Lake, Data Lakehouse

Sadržaj rada

1. Uvod

1.1. Predmet i cilj rada

Predmet rada su skladišta podataka utemeljena na jezerima podataka. Cilj rada je uz maksimalnu fokusiranost, na tehnički jednostavnim primjerima, jasno i razvidno predstaviti sve izvanredne glavne značajke takvih sustava. Stoga je za te potrebe implementirano radno okruženje koje ima minimalan, ali dostatan skup softvera otvorenog koda - Apache Spark, Delta Lake, DBeaver, JupyterLab, Weka i RStudio. Radi potpunosti u završetku izlaganja o tim konkretnim primjerima bit će dan prijedlog cjelovitog sustava, koji se može koristiti u praksi, a temelji se samo na dodavanju komponenata i promjeni konfiguracije sustava, kao što je primjerice MinIO za implementaciju pohrane jezera podataka u oblaku (privatnom, javnom, hibridnom).

1.2. Izvori podataka i metode prikupljanja

Kao izvor podataka korišteni su prije svega znanstveni članci, stručna literatura, te dokumentacija za korištene softvere otvorenog koda, koja se nalazi na njihovim službenim stranicama.

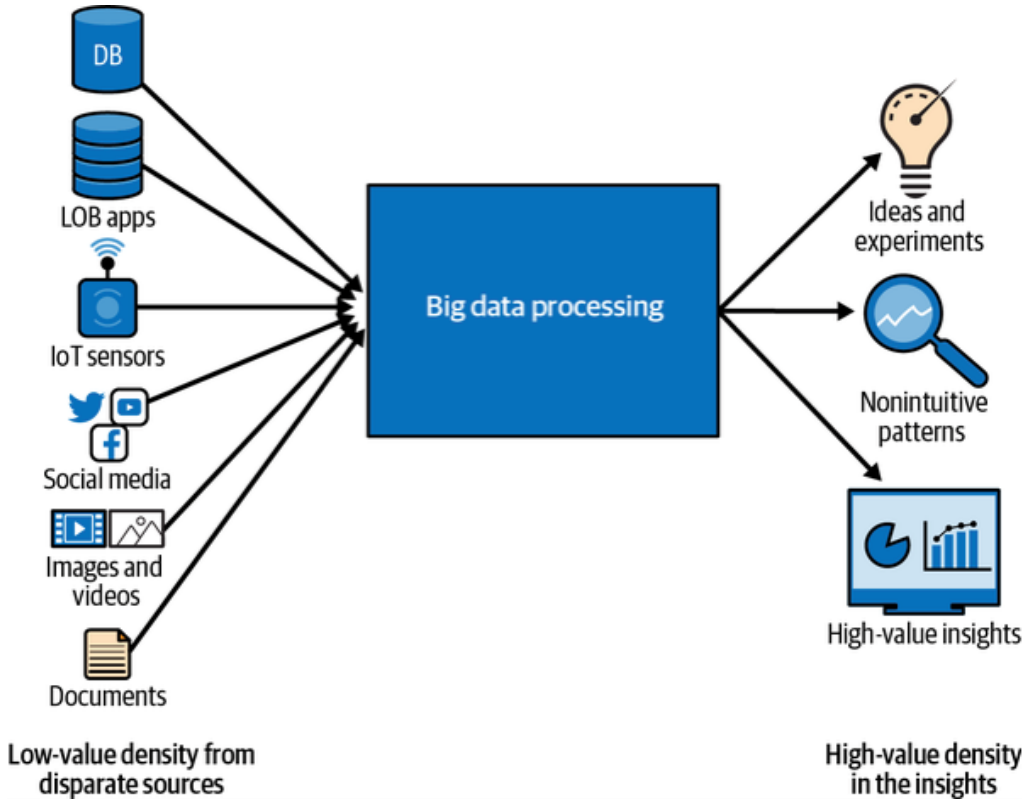
Temeljne postavke rada kao što su V dimenzije Velike količine podataka (eng. Big Data) i model računalstva u oblaku (eng. cloud computing) u kontekstu implementacije podatkovne infrastrukture preuzet je iz stručne literature.

1.3. Sadržaj i struktura rada

Rad se sastoji od tri cjeline. U prvoj cjelini je definirana arhitektura skladišta podataka, jezera podataka i skladišta podataka utemeljenih na jezerima podataka. U drugoj cjelini fokus se sužava na mogućnosti implementacije sustava na četveroslojnoj arhitekturi platforme za upravljanje podataka utemeljene na softverima Apache Spark i Delta Lake. U trećoj prezentiraju se konkretni primjeri na spomenutim softverima uz uporabu dodatnih alata DBeaver, JupyterLab, Weka i RStudio. U zaključku se iznosi ukupna ocjena rezultata iz treće cjeline.

2. Skladišta podataka, jezera podataka i skladišta podataka utemeljena na jezeru podataka

Arhitektura skladišta podataka utemeljenog na jezeru podataka rezultat je procesa koji je krenuo od tradicionalnih skladišta podataka (eng. Enterprise data warehouse), a temelji se na dva osnovna elementa - Koncept Velike količine podataka (eng. Big Data) i računarstvo u oblaku (eng. Cloud computing).
Velika količina podataka (eng. Big Data) je koncept, koji se odnosi na velike količine različitih vrsta podataka, koji nastaju iz raznorodnih izvora. Koncept je dan na slici 1:



Slika 1 Koncept "Velikih podataka" Izvor: Gopalan, 2022

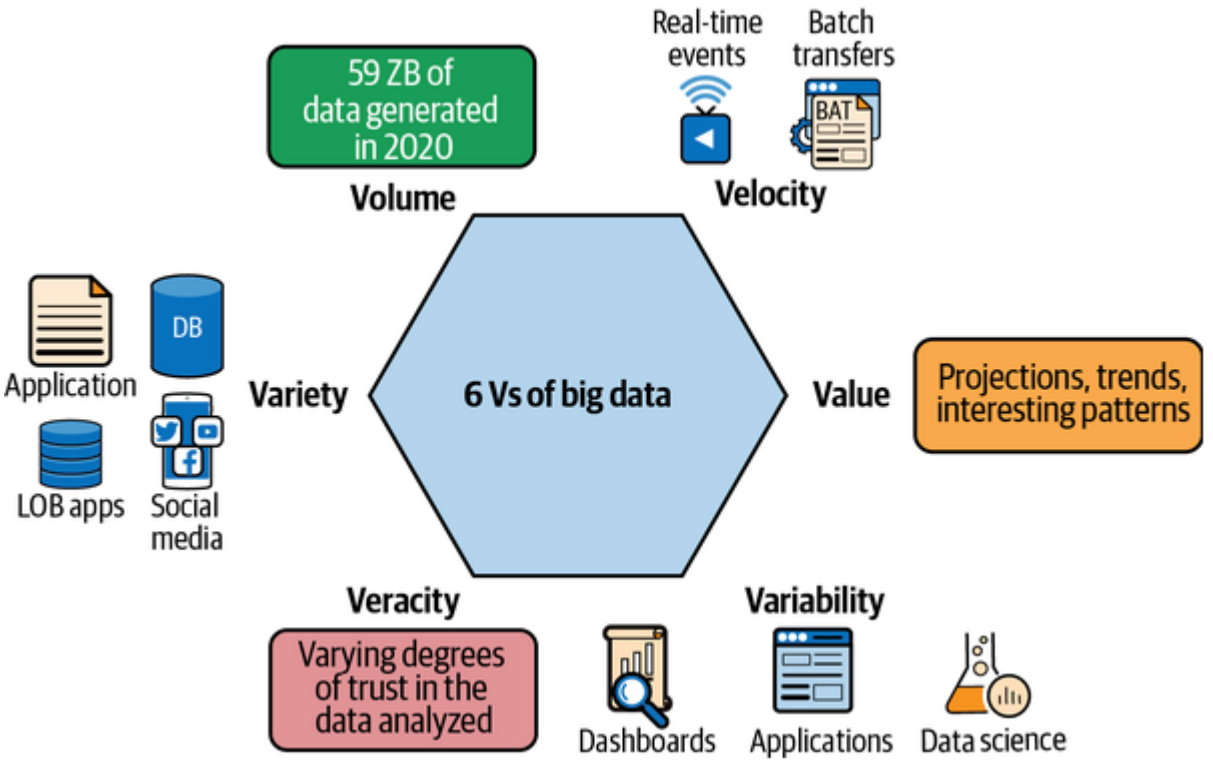
Kroz višegodišnju praksu jasno se iskristaliziralo šest dimenzija koje precizno karakteriziraju sve temeljne značajke velike količine podataka. Te dimenzije su kako slijedi:

- volumen (eng. Volume) - odnosi se na fizičku veličinu podataka koja se u općem slučaju može mjeriti ne samo u TB, već sve češće i u PB. Tu je svakako prikladno

istaknuti potrebu jednostavnog i ekonomičnog skaliranja kapaciteta pohrane koji računalna infrastruktura treba podržati.

- brzina (eng. Velocity) - odnosi se na brzinu dotoka podataka u sustav, koja primarno ovisi o vrsti izvora podataka. U primjeni podaci stižu u većim ili manjim cjelinama (eng. batch, dump), odnosno kontinuirano u obliku tijekova podataka (eng. data streams). Izazov na infrastrukturu nije samo osigurati pouzdan i robustan prihvat podataka u oba navedena slučaja, nego i u sljedećem koraku omogućiti efikasnu obradu, kao što je primjerice korelacija podataka koji dolaze u sustav različitim brzinama, razne vrste analiza same dinamike dolaska u realnom vremenu podataka i sl.
- raznolikost - (eng. Variety) - odnosi se na okolnost da podaci mogu biti u strukturiranom obliku (eng. structured data) ako se radi o podacima iz raznih aplikativnih sustava, preko podataka u polu strukturiranom obliku JSON ili XML (eng. semi-structured data), pa sve do podataka u nestrukturiranom obliku kao što su primjerice tekst, slike, zvuk, video i sl.
- vjerodostojnost - (eng. Veracity) - odnosi se na kvalitetu i porijeklo podataka. Osnovna premisa sustava za veliku količinu podataka je prihvat svih podataka iz svih raspoloživih izvora. To ima za posljedicu da je u cilju osiguranja vjerodostojnosti podataka svakako potrebno u općem slučaju provoditi postupke koji uključuju ispitivanje (eng. data examination) i čišćenje (eng. data cleansing), podataka, prije bilo kakve daljnje obrade.
- varijabilnost (eng. Variability) - odnosi se na mogućnost efikasnog upravljanja i obrade podataka, kojima se struktura u vremenu mijenja. Primjerice pojavljuju se nove kolone u tablicama u sklopu strukturiranih podataka, novi elementi u XML datotekama i sl.
- vrijednost - (eng. Value) - odnosi se na iskoristivost podataka u cilju njihove eksploatacije i ona može varirati u vremenu. Međutim temeljna pretpostavka da se radi cjelovit zahvat podataka omogućava da se maksimizira vrijednost podataka, stoga što kvalitetan cjelovit zahvat omogućava praktično sve moguće vrste uvida u podatke, ne samo za trenutačne, nego i buduće potrebe.

Dimenzije su prezentirane na slici 2:



Slika 2 Dimenzije "Velikih podataka" Izvor: Gopalan, 2022

Ukoliko se dimenzije navedene na slici 2 sagledaju kao cjelina u kontekstu podatkovne infrastrukture potrebne za implementaciju sustava, razvidno je da je temeljna značajka takvog sustava elastičnost kapaciteta (eng. elastic data infrastructure) u svim segmentima - memorijskim, procesnim, mrežnim.

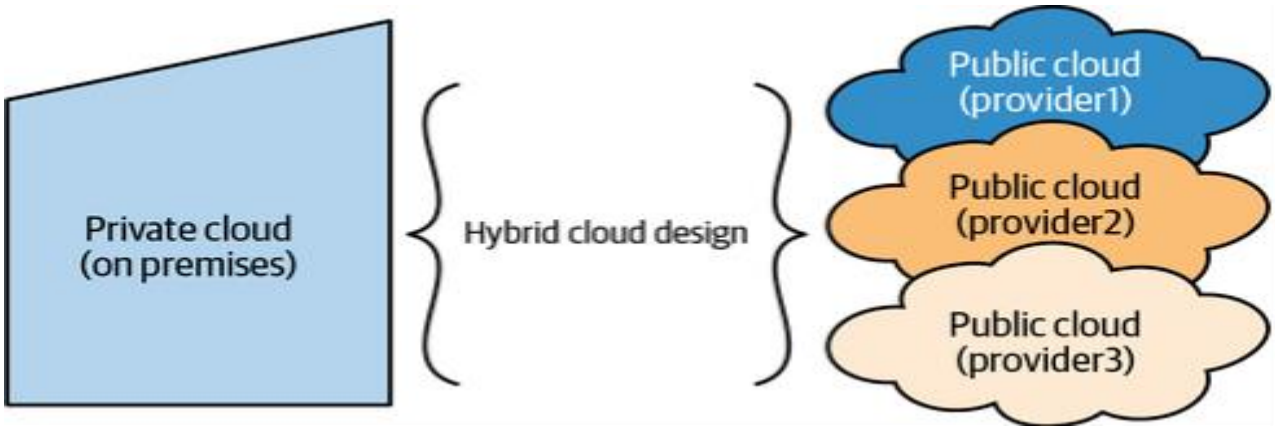
Računarstvo u oblaku (eng. cloud computing) predstavlja izravan odgovor na pitanje mogućnosti uspostave prije svega financijski ekonomične infrastrukture, koja rješava problem elastičnosti, kao glavni preduvjet za veliku količinu podataka. Konkretno, radi se o paradigmi informatičke tehnologije koja opisuje pružanje IT infrastrukture, odnosno njenih resursa u obliku usluge putem Interneta.

S obzirom na fizičku distribuciju IT resursa i njihovo vlasništvo postoji nekoliko tipova računarstva u oblaku:

- javni oblak (eng. public cloud) - čiji vlasnici su najčešće velike tvrtke, koje onda, po principu ekonomije obujma, preko Interneta iznajmljuju IT resurse koji se nalaze na infrastrukturi u njihovom vlasništvu raznim drugim organizacijama i pojedincima.

- privatni oblak (eng. private cloud) - čiji vlasnik je jedna organizacija i njegova infrastruktura raspolaže resursima koji zadovoljavaju specifične potrebe te organizacije. Dodatni zahtjev kod privatnog oblaka je da organizacija mora raspolagati sa potrebnom razinom tehničke ekspertize za njegovu postavku i eksploataciju.
- hibridni oblak (eng. hybrid cloud) - koji predstavlja kombinaciju dva prethodno navedena oblika, a primjenjuje se u slučaju kada se zbog specifičnih potreba kao što je primjerice sigurnost, zakonska regulativa i sl. određeni resursi koriste u privatnom oblaku, dok se drugi resursi iznajmljuju na javnom oblaku iz razloga ekonomičnosti. Pri tome su oba oblaka integrirana preko Interneta, nerijetko na vrlo sofisticirane načine.

Sva tri tipa oblaka dana su na slici 3:



Slika 3 Oblici oblaka Izvor: Gopalan, 2022

U oblaku su raspoloživi sljedeći tipovi resursa, koji se iznajmljuju kao usluga:

- infrastruktura kao usluga (eng. Infrastructure as a service - IaaS) - nudi hardverske resurse kao što su procesni, memorijski, mrežni resursi i sl. Za upravljanje tim virtualiziranim hardverskim resursima dostupne su razne vrste operacijskih sustava (najčešće razne inačice Linux ili Microsoft Windows operacijskih sustava), koji njima upravljaju u obliku tzv virtualnih računala (eng. virtual machine).
- platforma kao usluga (Platform as a service - PaaS) - gdje su ponuđene gotove usluge kao što su poznate relacijske baze podataka (npr. Microsoft Azure SQL, PosgreSql, MySql), NoSQL baze podataka(npr. MongoDB, Microsoft Azure CosmosDB), pa sve do samih skladišta podataka (npr. Snowflake data warehouse).

- softver kao usluga (Software as a service, or SaaS) - različite vrste gotovih aplikacija (npr. Microsoft Office 365, Salesforce) opremljenih web sučeljem, koje omogućavaju njihovu uporabu preko Interneta. Svakako treba spomenuti da je razvoj tih aplikacija daleko odmakao stoga što značajan broji njih omogućava implementaciju prilagodbi za specifične poslovne potrebe, ali uz pomoć poslovno i tehnički kvalificiranih stručnjaka.

U poslovnom smislu računarstvo u oblaku donijelo je čitav niz praktičnih prednosti kao što su primjerice smanjenje ukupnog troška vlasništva, ušteda na hardveru i softveru, plaćanje samo po intenzitetu uporabe, relativno lako držanje koraka sa inovacijama i što je najvažnije maksimalnu elastičnost kapaciteta stoga što se virtualizirani resursi mogu dinamički alocirati tijekom rada sustava u vrlo kratkom roku.
Zaključno, brojni softveri koji se koriste za izgradnju podatkovne infrastrukture iz područja otvorenog koda su raspoloživi u obliku usluga u oblaku, ali istodobno omogućavaju bez ostatka postavku tj. instalaciju i na vlastitoj infrastrukturi (privatnom oblaku). Zbog toga će u nastavku izlaganja fokus biti na njihovim mogućnostima u izgradnji skladišta podataka utemeljenih na jezerima podatka, pri čemu će referenca na određeni tip potrebne infrastrukture biti dana samo ako se radi o nekom specifikumu.

2.1. Skladišta podataka

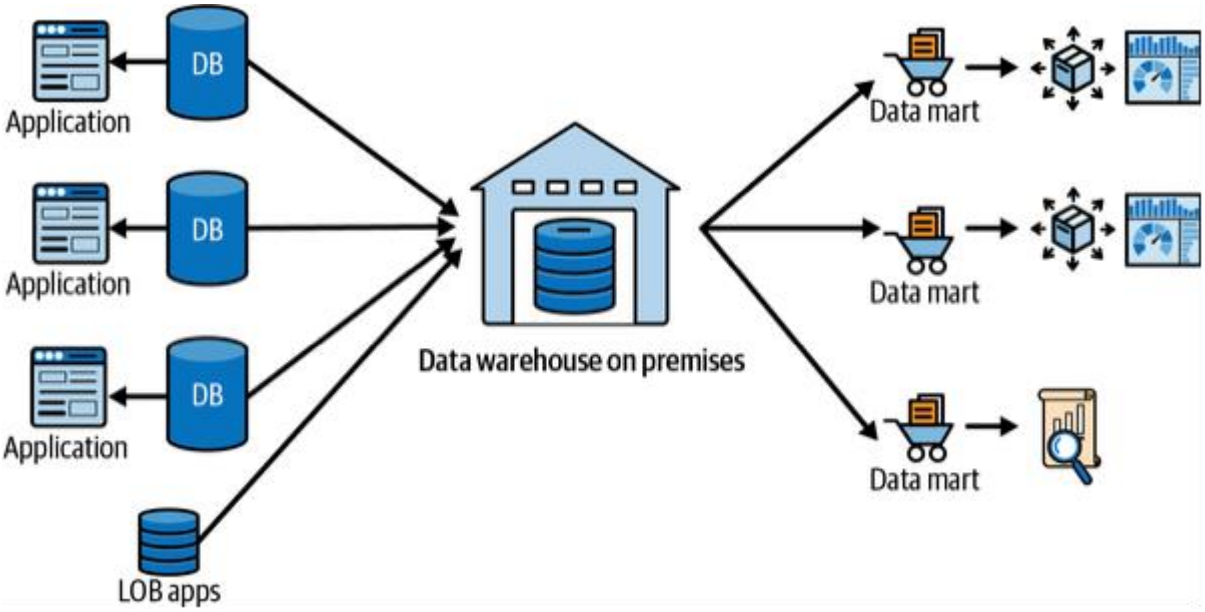
Tradicionalna skladišta podataka (eng. Enterprise Data Warehouse - EDW) nastala su sukladno ustaljenoj praksi svoga vremena pristupom orijentiranom na (trenutačni) poslovni problem (eng. problem-centric approach). Praktično to znači da je problem identificiran, definirana odgovarajuća struktura podataka, prikupljeni podaci sukladno toj strukturi pohranjeni u OLTP bazu podataka, uporabom odgovarajućih ETL (eng Extract Transform Load - ETL) procedura podaci preneseni iz OLTP baze u OLAP bazu (strukturiranu po višedimenzionalnom modelu), koja je onda temeljni izvor podataka za upite i kontrolne ploče, koje rješavaju postavljeni poslovni problem. Tako postavljeni sustav na isključivo vlastitoj infrastrukturi (privatnom oblaku) organizacije sastojao se od tri elementa:

- skladište podataka - relacijska baza podataka u kojoj su fizički smješteni podaci zajedno sa tzv. kataloškim tablicama (eng. catalog tables) u kojima su smješteni metapodaci, koji opisuju smještene podatke. Višedimenzionalnost je utemeljena na Star ili Snowflake shemi. U oba slučaja radi se o konceptu kategorizacije podataka kao

analitičkih dimenzija implementiranih preko pripadajućih tablica, te implementaciji mjera implementiranih u vidu tablica činjenica (eng. fact table). Dimenzijske tablice su povezane se tablicama činjenica preko stranih ključeva (eng. foreign key).

- područno skladište (eng Data Mart) - specijalizirana struktura za dohvaćanje podataka za neku specifičnu namjenu kao što je primjerice neka poslovna funkcija. Potrebno je implementirati dodatni skup transformacija, koje uzimaju određeni relevantni segment skladišta podataka i prenose u područno skladište. U cilju povećanja brzine analitičkih upita u pravilu se koriste višedimenzionalne baze podataka implementirane pomoću specijaliziranih indeksnih struktura. Radi se o višedimenzionalnim indeksnim strukturama po modelu kocke (eng. cube) ili u novije vrijeme strukturama pohrane podataka orijentiranih na stupce (eng. columnar store).
- sloj konzumacije/poslovne inteligencije podataka (eng. Consumption /Business Intelligence - BI layer) - gdje se koriste razni alati (npr. Microsoft Excel) za vizualizaciju i poslovne analize podataka, koji se dohvaćaju pomoću upita u područnom skladištu.

Arhitektura takvog sustava dana je na slici 4:



Slika 4 Arhitektura tradicionalnog skladišta podataka Izvor: Gopalan, 2022

Arhitektura prikazana na slici 4 ima značajne nedostatke kako slijedi:

- visoko strukturirani podaci - rezultiraju velikom nefleksibilnosti, stoga što u svakoj fazi rada sustava podaci moraju biti striktno strukturirani i usklađeni. To praktično znači da

dodavanje modifikacija strukture podataka na izvorima ima za posljedicu promjenu na nekoliko mjesta, što čini nadogradnje i održavanje vrlo kompleksnim i skupim. To je za koncept Velike količine podataka potpuno neprihvatljivo, jer je obrada polu strukturiranih i nestrukturirani podataka praktično nemoguća.

- silosi podataka - višestruke kopije podataka pohranjenih nekoliko puta ovisno o njihovoj specifičnoj namjeni tj. načinu uporabe. Praktično rezultiraju problemima kao što su pogreške prilikom kopiranja podataka, nekonzistencija između podataka dok traje proces kopiranja i sl.
- isključivo vertikalna skalabilnost sustava - što praktično znači da se povećanje kapaciteta sustava u bilo kojem smislu može ostvariti isključivo skupom nadogradnjom postojećeg ili nabavom novog hardvera. Uporaba relacijskih baza za pohranu velikih količina, posebice povijesnih podataka, je vrlo problematična, stoga što one tome nisu niti namijenjene. Njihova primarno transakcijska značajka rezultira čitavim nizom problema, koji se u konačnici svode na veliku potrošnju hardverskih resursa.

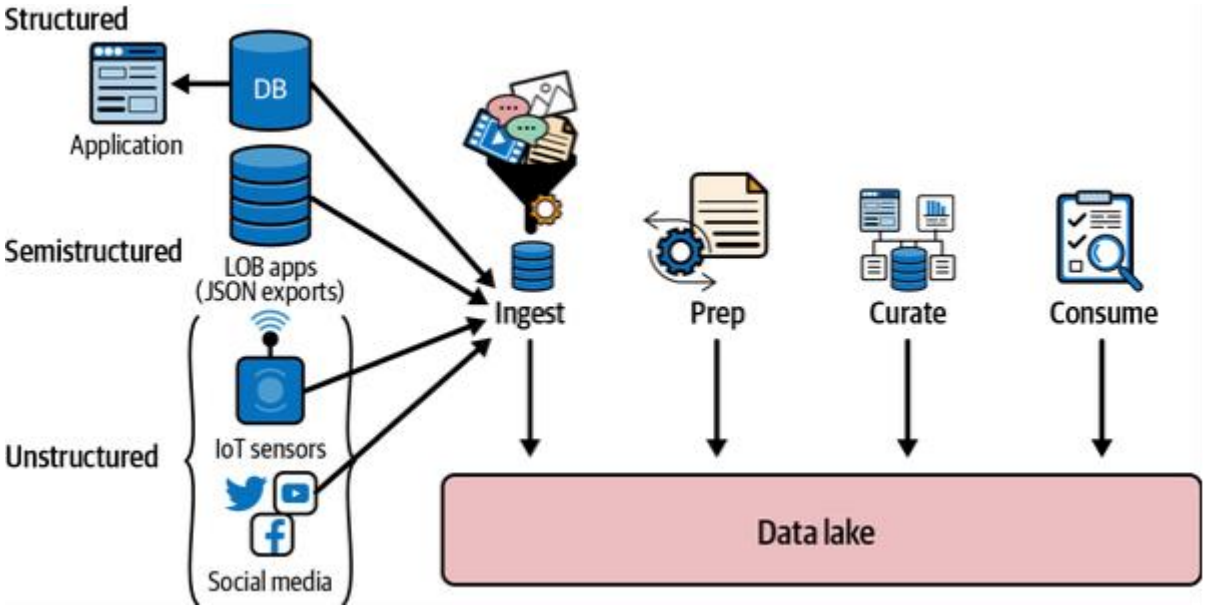
Naposljetku, u praktičnom smislu skladišta podataka imaju primarnu namjenu za poslovnu analitiku temeljenu na višedimenzionalnoj analizi, dok je za čitav niz drugih potreba njihova primjena jednostavno nemoguća. Tipični primjeri su analitika u realnom vremenu (eng. real time analytics) zbog inherentne sporosti protoka podataka (eng. data latency problem) kroz više stupanjski podatkovni cjevovod ili primjena na području znanosti o podacima (eng. data science) gdje je od interesa sasvim drugi pogled na podatke, odnosno njihove obrade i sl.

Skladišta podataka u oblaku (eng. Cloud Data Warehouse) ponuđena su u obliku platforma kao usluga (PaaS) na javnim oblacima, a pokušaj su da se kroz znatno racionalniji način uporabe hardvera koji omogućava računarstvo u oblaku, riješe problemi skalabilnosti i elastičnost prisutni kod tradicionalnih skladišta podataka u tehničkom i financijskom smislu. Već naglašena velika fleksibilnost u primjeni raznih softvera otvorenog koda za izgradnju podatkovne infrastrukture omogućila su i nove praktične mogućnosti kao što su primjerice optimizirane integracije za unos podataka, analitičku obradu i poslovnu inteligencija (BI) kao dio usluge. U konačnici ovo rješenje omogućava elastično skaliranje s rastućim potrebama i potpuno apstrahiranje kompleksnosti infrastrukture, uz istodobno brže performanse i niži trošak vlasništva od tradicionalnih skladišta podataka na vlastitoj infrastrukturi. Najpoznatiji primjeri su Amazon Redshift, Google BigQuery, Microsoft Azure Synapse Analytics i Snowflake Data Cloud.

Međutim, sva ostala ograničenja u kontekstu V dimenzija Velike količine podataka vezana za tradicionalna skladišta podataka i dalje ostaju.

2.2. Jezera podataka

Jezera podataka (eng. Data Lake) nastala su kao pokušaj rješavanja zahtjeva koje postavljaju koncepti Velike količine podataka. Suprotno skladištima podataka gdje se primjenjuje pristup orijentiran na poslovni problem, primjenjuje se pristup prvotno orijentiran na podatke (eng. data-first approach). Posljedično to znači da se suprotno konceptu tradicionalnih skladišta podataka, svi podaci koji ulaze u sustav se smatraju korisnim, bilo odmah ili tek u budućnosti. Praktično to znači da se podaci unose u sustav u njihovom izvornom obliku, bez ikakvih ograničenja izvora, veličine ili formata podataka, pohranjuju u jezeru podataka, implementiranom na visoko skalabilnom sustavu, koji može pohraniti bilo koju vrstu podataka. Potrebno je svakako istaknuti da se radi o neobrađenim podacima različite kvalitete i vrijednosti, tako da su neizbježno potrebne transformacije za postizanje odgovarajuće kvalitete i vrijednosti u poslovnom smislu. Arhitektura sustava dana je na slici 5:



Slika 5 Arhitektura jezera podataka Izvor: Gopalan, 2022

Tek tijekom implementacije spomenutih transformacija podataka definira se odgovarajuća shema nad podacima koja predstavlja najoptimalniji strukturni "pogled" na podatke za konkretnu transformaciju. Upravo ta fleksibilnost u odgovarajućem strukturiranju podataka

ima za posljedicu povećanje njihove vrijednosti, stoga što je struktura maksimalno prilagođena svim poslovnim zahtjevima. Istodobno za potrebe nekog drugog zahtjeva može se formirati sasvim drugi strukturni "pogled". Primjerice tako pripremljeni podaci se mogu koristiti izravno ili mogu biti jednostavno preneseni u neko višedimenzionalno skladište podataka.
U usporedbi sa tradicionalnim skladištima podataka jezera podataka imaju sljedeće prednosti:

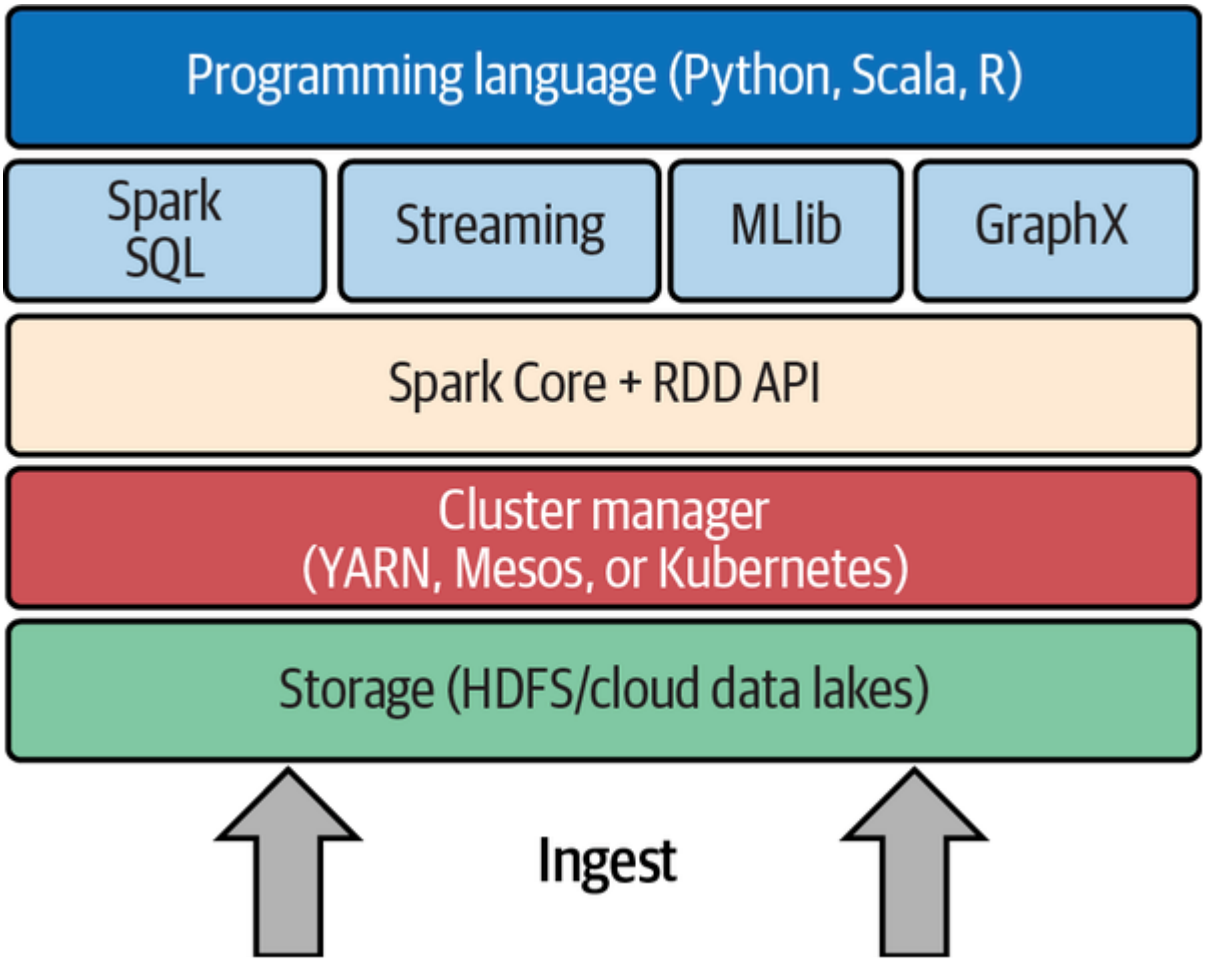
- nema ograničenja na podatke - bilo da se radi o razini strukturiranosti, brzini unosa u sustav, njihovom obimu i ostalim V dimenzijama koje karakteriziraju Velike količine podataka. Dodatno, trošak implementacije infrastrukture je značajno niži primarno stoga što nisu potrebni brzi diskovi i kapaciteti koji se moraju koristiti kod relacijskih baza podataka zbog transakcijskog principa rada
- nema izoliranih silosa podataka - svi podaci su konsolidirani na jednom mjestu tj. nema specijaliziranih segmenata prilagođenih određenim vrstama obrade, pa samim time izostaju svi problemi vezani za kopiranje podataka
- potpuna fleksibilnost obrade podataka koji se obavljaju na istom mjestu - znači fundamentalnu prednost u odnosu na monolitnu arhitekturu relacijskih baza stoga što je pohrana podataka u potpunosti izolirana od njihove obrade
- mogućnost potpuno neovisnog skaliranja kapaciteta za pohranu i obradu podataka - koja je izravna posljedica potpunog odvajanja sloja obrade od sloja pohrane podataka

Tehnološka osnovica jezera podataka sastoji se od tri glavna elementa - pohrana jezera podataka u oblaku (eng. cloud data lake storage), analitički programski alat (eng. analytics engine) i cjevovodi tijekova podataka u realnom vremenu (eng. Real-time stream processing pipelines).
Pohrana jezera podataka u oblaku (eng. cloud data lake storage) je raspoloživa u obliku platforme kao usluga (PaaS) i zadovoljava apsolutno sve V dimenzije Velike količine podataka, koje se odnose na pohranu podataka. Radi se o modelu pohrane podataka u oblaku gdje se podaci pohranjuju u odgovarajućim logičnim skupinama, redundantno, optimizirano, uz mogućnost primjene raznih sigurnosnih modela, na hardverskoj infrastrukturi koja se u općem slučaju sastoji od velikog broja poslužitelja. Sva ta iznimno velika kompleksnost na praktičnoj razini svodi se na uporabu odgovarajućeg programskog sučelja (eng. Application Programming Interface - API) za pristup podacima.
Najpoznatiji primjeri u javnom oblaku su Amazon S3, Azure Data Lake Storage i Google Cloud Storage. Međutim postoje softveri otvorenog koda koji omogućavaju implementaciju na

privatnom, hibridnom i javnom oblaku kao što su primjerice Apache Hadoop HDFS, MinIO i Apache Ozone.

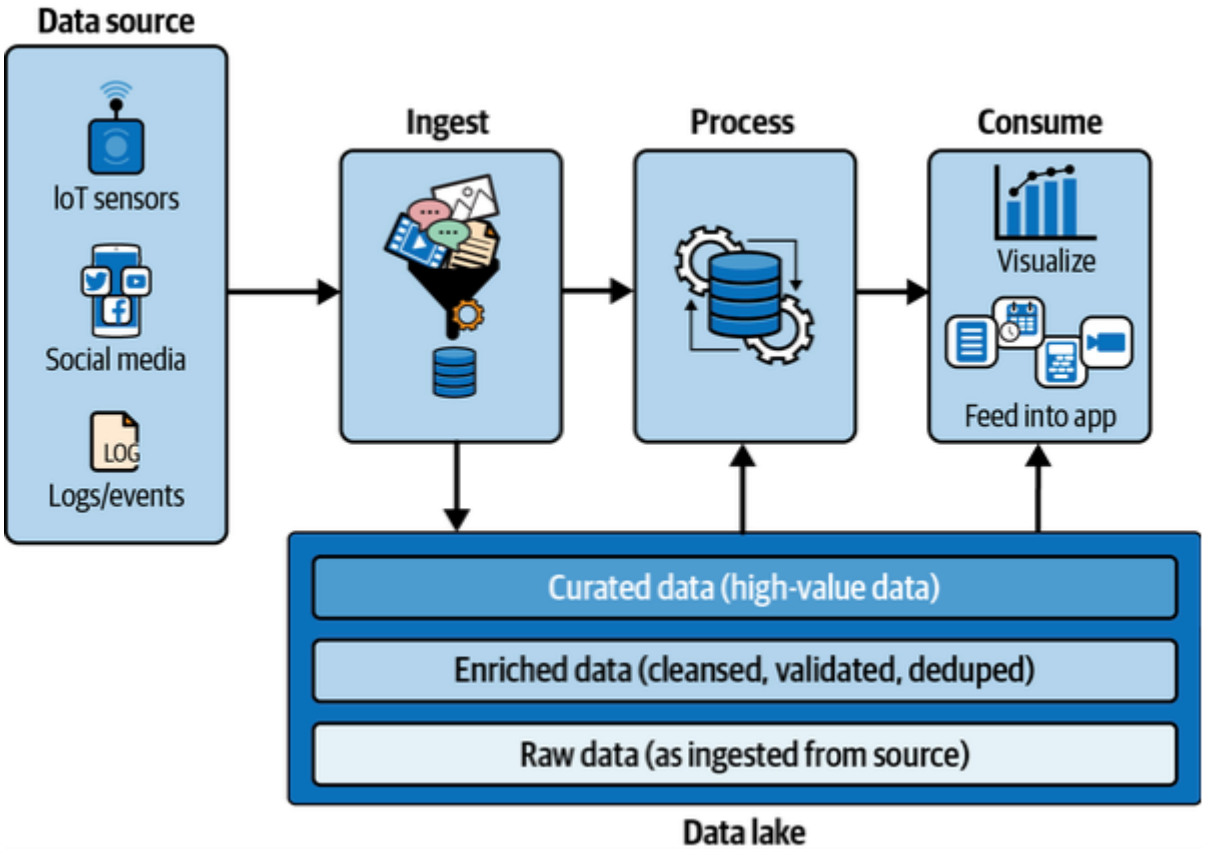
Analitički programski alati (eng. analytics engine) za Veliku količinu podataka omogućavaju sveobuhvatnu obradu podataka, koji se nalaze na sloju pohrane jezera podataka u oblaku. Radi se o softverima koji maksimalno koriste sve značajke tog sloja za pohranu i omogućavaju paraleliziranu obradu s tolerancijom na pogreške velikih količina svih vrsta podataka podataka, brzina unosa u sustav i svega ostalog što je potrebno sukladno V dimenzijama Velike količine podataka u segmentu obrade podataka. Najpoznatiji primjeri analitičkih programskih alata su:

- Apache Hadoop MapReduce - analitički program za obradu podataka u sklopu sustava Apache Hadoop, bazirana na MapReduce algoritmu paralelne obrade podataka s tolerancijom na pogreške. Glavna značajka tog modela paralelne obrade je da se nakon svakog koraka međurezultat koji je rezultat tog koraka i ujedno ulazna jedinica u slijedeći korak, pohranjuje na Apache Hadoop HDFS. Manjkavost takvog načina rada očitovala se u tome da nije bilo moguće implementirati rješenje za interaktivno izvršavanje SQL upita sa u općem slučaju zadovoljavajućim performansama.
- Apache Spark - je samostalni analitički program za obradu podataka, na kome su uspješno riješeni problemi performansi interaktivnih SQL upita te je otvoren prostor za čitav niz dodatnih primjena zbog njegove izvanredno dobre modularne arhitekture. Problem performansi je riješen uvođenjem posebne podatkovne strukture (eng. Resilient Distributed Datasets - RDD) koja predviđa spremanje međurezultata u radnoj memoriji računala na kome se obrada odvija, dok se tek krajnji rezultat pohranjuje na sloju pohrane jezera podataka u oblaku. Time su postignuta višestruka ubrzanja. Zbog toga i čitavog niza svojstava upravo će ovaj analitički program biti korišten kao glavna tehnološka osnovica u nastavku izlaganja. Arhitektura na konceptualnoj razini je dana na slici 6:



Slika 6 Konceptualni prikaz arhitekture jezera podataka Izvor: Gopalan, 2022

Cjevovodi tijekova podataka u realnom vremenu (eng. Real-time stream processing pipelines) predstavljaju sve važniju komponentu svakog sustava Velike količine podataka u današnje vrijeme jer je od interesa praćenje raznih aspekata poslovanja u realnom vremenu. Radi se o sustavima koji prenose poruke u kojima se u općem slučaju nalaze male količine podataka, no istodobno se može raditi doslovno o desetinama tisuća takvih poruka u sekundi. Softvere koji obavljaju tu funkciju karakterizira otpornost na pogreške u radu kroz mogućnost postavke u konfiguracijama visoke raspoloživosti, garancija isporuke poruke i sl. Daleko najpoznatiji primjer je softver otvorenog koda Apache Kafka, ali sve više za čitav niz primjena i Structured Streaming komponenta analitičkog programskog alata Apache Spark. Arhitektura je dana na slici 7:

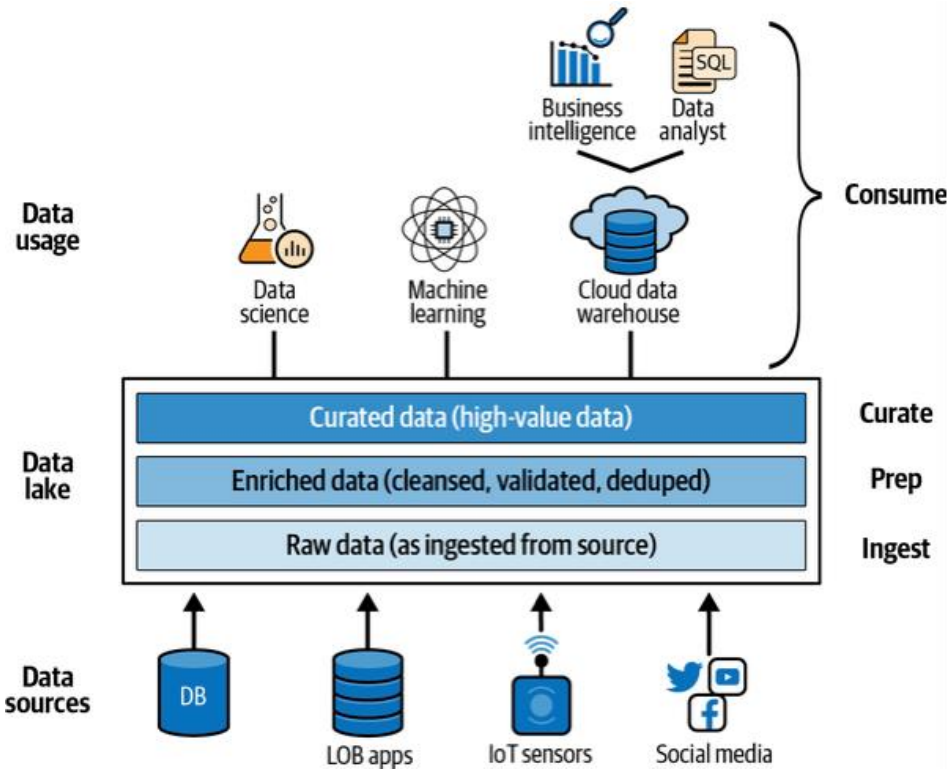


Slika 7 Prikaz arhitekture streaminga podataka pomoću Apache Spark-a Izvor: Gopalan, 2022

2.3. Skladišta podataka utemeljena na jezeru podataka

Skladište podataka utemeljeno na jezeru podataka (eng. Data Lakehouse) nastalo je kao posljednji evolutivni korak s primarnom namjerom konsolidacije svih podataka u samo jednom sloju pohrane - jezeru podataka.

Međutim, tom koraku prethodio je među korak - Moderno skladište podataka (eng. Modern Data Warehouse). Radi se o pragmatičnom hibridnom rješenju koje uključuje jezero podataka i skladište podataka u oblaku, pri čemu svako obavlja svoju specifičnu funkciju. Jezero podataka služi za jeftinu pohranu velikih količina svih vrsta podataka i podržava scenarije iz područja znanosti o podacima i sl., za koje je skladište podataka neprimjenljivo. S druge strane skladište podataka u oblaku komplementarno obavlja funkcije efikasnog izvršavanja SQL upita i višedimenzionalne poslovne analitike za koje je jezero podataka neprimjenjivo. Praktično svi podaci se unose u jezero podataka, , transformiraju uz pomoć analitičkih programskih alata otvorenog koda Apache Hadoop MapReduce ili Apache Spark, tako da višestruki skupovi podataka mogu biti na odgovarajući način agregirani, filtrirani i u konačnici jednostavno preneseni u skladište podataka u oblaku. Arhitektura sustava je dana na slici 8:



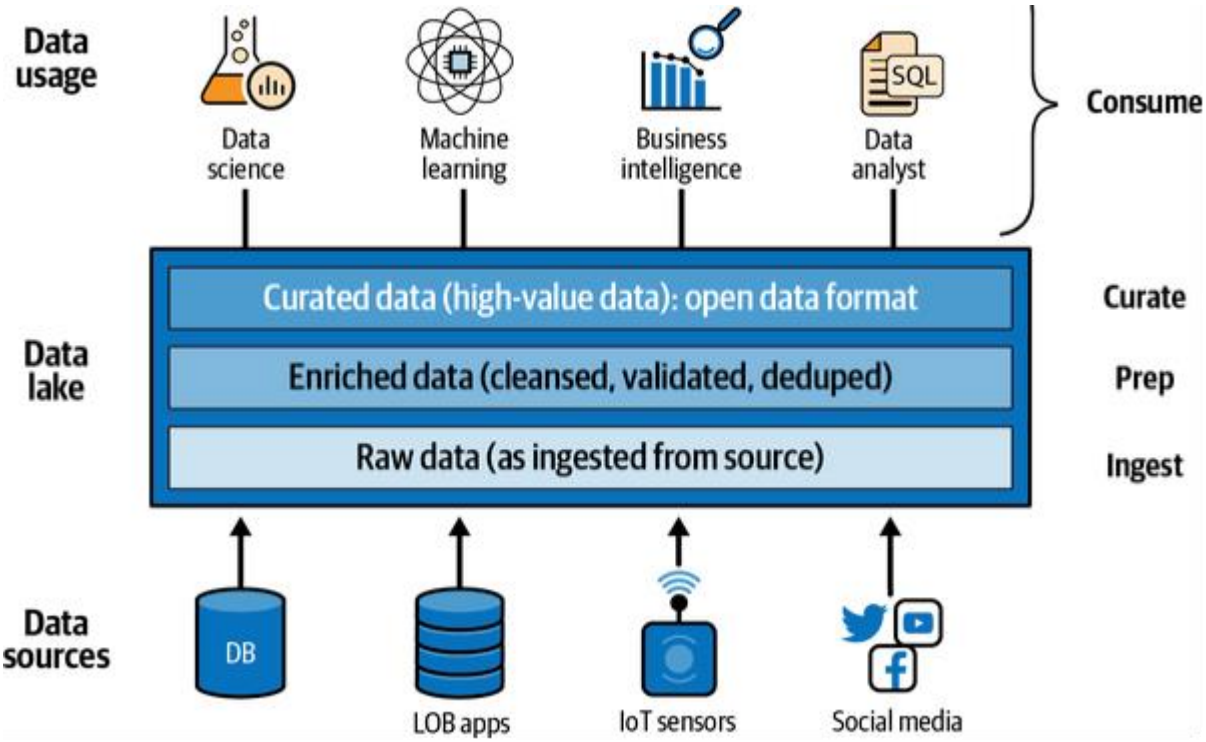
Slika 8 Arhitektura modernog skladišta podataka Izvor: Gopalan, 2022

Svakako treba istaknuti da ova arhitektura uvodi promjenu s ETL principa, na ELT (eng. Extract Load Transform) nužno potreban za uspješnu implementaciju sustava Velike količine podataka po svim V dimenzijama.

Bez obzira što se na prvi pogled može učiniti da se mogla izbjeći, uporaba jezera podataka u kombinaciji sa skladištem podataka u oblaku je iznimno važna stoga što se njenim uvođenjem postavlja temelj za uspješan evolutivni razvoj u pravcu skladišta podataka utemeljenih na jezerima podataka. U praktičnom smislu postojeće potrebe se mogu zadovoljiti skladištem podataka u oblaku i istodobno bez odgađanja započeti sa svim ostalim aspektima rada s podacima koje karakteriziraju V dimenzije Velike količine podataka.

Posljedično, arhitektura skladišta podataka utemeljenog na jezeru podataka mora podržati scenarije za razne vrste analitike vezane za primjerice znanost o podacima i strojno učenje, ali istodobno i uobičajene scenarije na skladištima podataka iz segmenta poslovne inteligencije kao što je interaktivno izvršavanje SQL upita. U praktičnom smislu to donosi evidentne prednosti stoga što je implementacija jezera podataka znatno jeftinija od skladišta podataka, izbacuje se kopiranje podataka iz jezera podataka u skladište podataka i konačno apsolutno svi podaci mogu biti dijeljeni bez ikakvih ograničenja (osim sigurnosnih) između raznih profila

korisnika, kao što su znanstvenici o podacima, poslovni analitičari i sl. Arhitektura je dana na slici 9:



Slika 9 Arhitektura jezera podataka utemeljenog na skladištu podataka Izvor: Gopalan, 2022

Međutim, da bi se ovakav sustav implementirao tj. da bi jezero podataka u potpunosti preuzelo funkciju skladišta podataka bilo je nužno iznaći odgovarajuće tehnološko rješenje koje će omogućiti da se u rad s podacima na jezeru podataka uključe fundamentalne značajke skladišta podataka prisutne na relacijskim bazama kako slijedi:

- osiguranje striktne sukladnosti podataka za zadanom strukturom/shemom (eng. Schema definition and enforcement) - prisutno u skladištima podataka od samih početaka. To znači da se mogu unositi isključivo oni podaci koji su striktno sukladni sa unaprijed definiranom shemom (eng. schema on write)
- transakcijski način rada po ACID modelu (eng. ACID-compliant transactions) - što znači da manipulacije s podacima (transakcije) zadovoljavaju uvijete atomarnosti (eng. Atomicity), konzistentnosti (eng. Consistency), izolacije (eng. Izolation) i trajnosti (eng. Durability)
- optimizirano izvršavanje SQL upita (eng. Optimized for SQL) - cjelovita podrška za optimizirano izvršavanje SQL upita, stoga što je većina analitičkih alata koji se koriste

u poslovnoj inteligenciji u segmentu dohvata podataka temeljena na mehanizmu SQL upita.

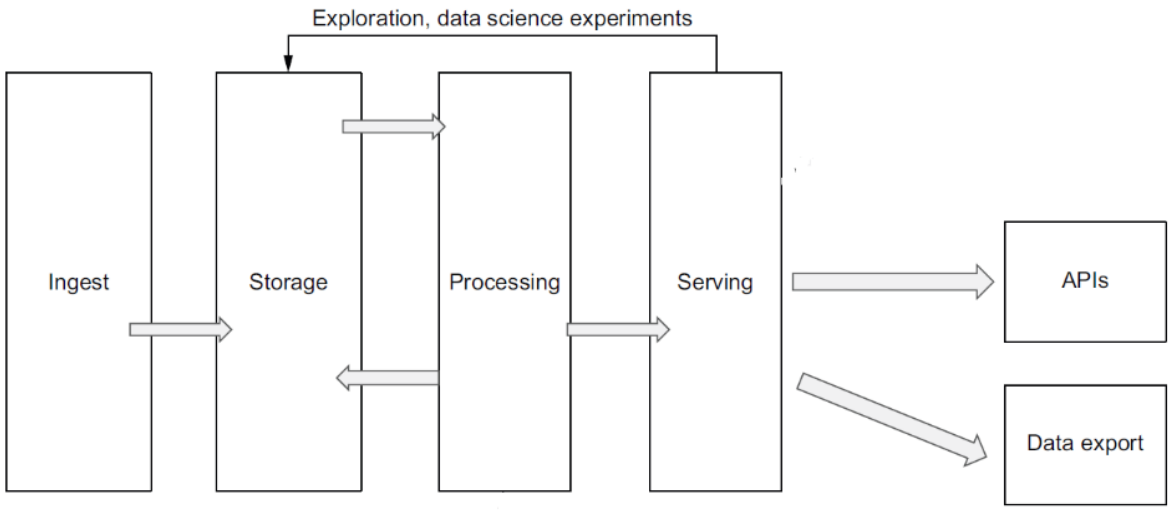
Također, činjenica da se u jezeru podataka pohranjuju strukturirani, polu strukturirani i nestrukturirani podaci impliciraju da ono ima obilježja baze podataka s više modela (eng. multi-model database). I praktičnom smislu to znači da se ovisno o obliku podataka primjenjuje odgovarajući model. U praksi su te baze podataka poznate kao NoSQL beze podataka i nastale su kao odgovor na probleme vezane za transakcijski model rada relacijskih baza. Tipičan primjer NoSQL baze podataka s više modela je OrientDB baza koja omogućava da se podaci u njoj strukturiraju po dokumentnom i graf modelu. Za te baze podataka primjenjuje se BASE model konzistencije podataka što znači da manipulacija sa podacima zadovoljava uvjete osnovne dostupnosti tj. baza je dostupna većinu vremena (eng. Basic Availability), zapis podataka u replikama spremišta podataka ne moraju biti potpuno istovjetno zapisani cijelo vrijeme (eng. Soft state), i eventualne dosljednosti - konzistencija podataka uspostavlja se tek prilikom čitanja (eng. Eventual consistency).

Formati podataka (eng. Data formats) zapisa na jezeru podataka u kojima se zapisuju podaci u jezeru, a koji imaju sve gore navedene tehničke značajke potrebne za skladišta podataka su iznimno prakrični za uporabu. Za razliku od relacijskih baza gdje je format podataka zatvoren, orijentiranost jezera podataka na de facto rad s datotekama, ti novi, iznimno inovativni, formati podataka postavljeni su od samih početaka na sasvim druge osnove. Radi se o otvorenim formatima za koje postoji vrlo detaljna dokumentacija ne samo na službenim web stranicama nego i stručnoj literaturi, koji predviđaju poseban jasno definiran sloj za upravljanje meta podacima tj. shemom. Najpoznatiji primjeri tih formata podataka su Delta Lake, Apache Iceberg i Apache Hudi.
Analitički programi kao što je to primjerice Apache Spark maksimalno su iskoristili otvorenost gore navedenih formata podataka i sve njihove značajke. To je u konačnici rezultiralo iznimno uspješnim rješenjem problema performansi interaktivnih SQL upita.

3. Mogućnosti implementacije skladišta podataka utemeljenog na jezeru podataka na četveroslojnoj arhitekturi

Analiza mogućnosti implementacije skladišta podataka utemeljenog na jezeru podataka na četveroslojnoj arhitekturi platforme za upravljanje podacima kreće od analize konkretnih mogućnosti sastavnica tehnološke osnovice, koje se onda adekvatno alociraju i koriste na pojedinim slojevima arhitekture platforme za upravljanje podacima.

Arhitektura platforme za upravljanje podacima omogućava da se na sustavan način strukturira konkretno rješenje sustava uz pomoć raspoloživih tehnoloških sastavnica prateći na prirodan način tijek podataka. Prvi sloj obuhvaća prikupljanje/unos podataka u sustav u skupnom obliku ili tijek podataka u realnom vremenu. Drugi sloj odnosi se na pohranu podataka u jezeru podataka. Treći sloj odnosi se na obradu podataka uz pomoć analitičkih strojeva. Četvrti sloj odnosi se na na posluživanje podataka što obuhvaća poslužiteljsku stranu koja odgovara na interaktivne SQL upite i dostavlja tražene podatke, te korisničku stranu gdje se koriste razni analitički alati koji za potrebe dohvata podataka postavljaju SQL upite i prihvaćaju rezultate. Arhitektura na konceptualnoj razini dana je na slici 10:



Slika 10 Konceptualni prikaz arhitekture jezera podataka utemeljenog na skladištu podataka Izvor: Zburivsky, Partner, 2021

Tehnološka osnovica iskristalizirala se već tijekom izlaganja koje se odnosi na analizu arhitekture skladišta i jezera podataka, odnosno evolutivni proces koji je od tradicionalnih skladišta podataka doveo u konačnici do skladišta podataka utemeljenih na jezerima podataka.

Ona se sastoji od pohrane podataka u jezeru podataka, formata podataka i analitičkih programskih alata.

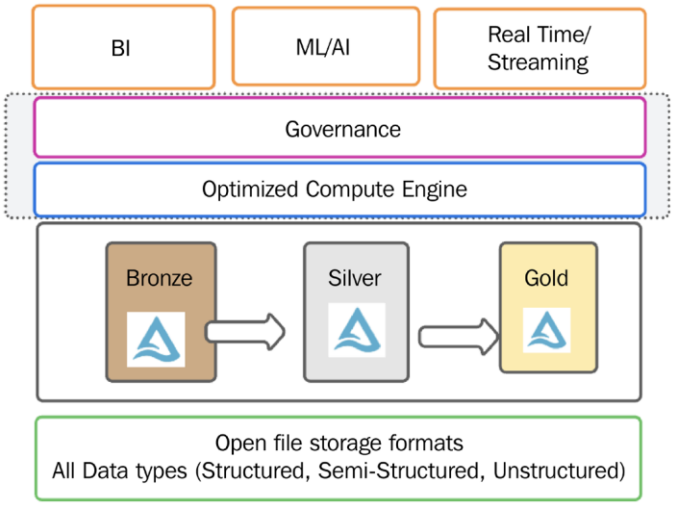
Pohrana podataka u jezeru podataka donijela je u odnosu na relacijske baze utemeljene na zatvorenom, odnosno javno nedostupnom formatu zapisa podataka fundamentalne promjene. Uvodi se pohrana podataka u obliku datoteka (eng. files) čiji format je za razliku od relacijskih baza javno dostupan i dobro dokumentiran. Utemeljenost na datotekama organiziranim pomoću hijerarhije mapa/kontejnera (eng. folders/containers) donosi čitav niz prednosti kao što je brzina manipulacije, fleksibilnost (uvođenje novog formata je potpuno neovisno o formatu prethodno pohranjenih datoteka), velike mogućnosti indeksiranja, visoka razina integriteta podataka u jednoj datoteci, nema nikakvog ograničenja što se tiče formata datoteke. Diskovni prostor za pohranu datoteka u oblaku je jeftin, stoga što za pohranu u obliku datoteka u pravilu nije potreban brzi diskovni prostor kao u slučaju transakcijskih relacijskih baza. U praktičnom smislu bez obzira na konkretnu tehnologiju, način manipulacije je kao da se radi o običnom datotečnom sustavu (eng. file system).

Formati podataka razvijali su se sukladno sve većim potrebama i općenitom trendu da se zbog svih prezentiranih prednosti jezera podataka obrada podataka obavlja izravno na podacima fizički pohranjenim u samom jezeru. Radi se definitivno o radikalnoj promjeni paradigme, koja je rezultirala upravo novom generacijom vrlo sofisticiranih formata podataka, koji mogu podržati sve prije navedene značajke potrebne za izravnu implementaciju skladišta podataka na samom jezeru podataka. Za potrebe daljnjeg izlaganja odabran je format podataka Delta Lake, zbog najbolje integracije sa analitičkim programom Apache Spark. To nije iznenađujuće uzme li se u obzir da je autor tog formata tvrtka Databricks, čiji osnivači su glavni kreatori i utemeljitelji projekta otvorenog koda analitičkog programskog alata Apache Spark. Dodatni razlog za uporabu ovog formata je izvanredno dobra dokumentiranost koja seže od znanstvenih članaka njegovih autora, pa sve do vrlo bogate i opsežne stručne literature. Konkretna primjena primjena Data Lake formata podataka bit će dana u sklopu izlaganja o sloju pohrane.

Analitički programi posljednje generacije kao što je to primjerice Apache Spark 3.x su kroz duži period razvoja značajno nadograđeni u segmentima prikupljanja/unosa i posluživanja podataka, kao i u mogućnosti za izravnu pohranu podataka u jezerima podataka implementiranim na raznim tehnološkim rješenjima. Time je praktično omogućeno da se uporabom modernog analitičkog programskog alata zadovolje potrebe tri sloja četveroslojne arhitekture (prikupljanje/unos, obrada i posluživanje podataka) uz mogućnost već ugrađene podrške za pohranu podataka u jezeru podataka implementiranom na bilo kojoj već spomenutoj tehnologiji. Dodatni razlog za uporabu Apache Spark 3.x u ostatku izlaganja je izvanredno

dobra dokumentiranost koja seže od znanstvenih članaka njegovih autora koji se odnose na njegove ključne komponente, pa sve do vrlo bogate i opsežne stručne literature. Konkretna primjena pojedinih komponenata ovog analitičkog programa bit će dana u izlaganju o pojedinom sloju arhitekture.

Arhitektura skladišta podataka utemeljenog na jezeru podataka implementiranog pomoću Delta Lake formata podataka i analitičkog programskog alata Apache Spark na konceptualnoj razini dana je na slici 11:



Slika 11 Prikaz arhitekture skladišta podataka utemeljenog na jezeru podataka na koncepnualnoj razini Izvor: Mahapatra, 2022

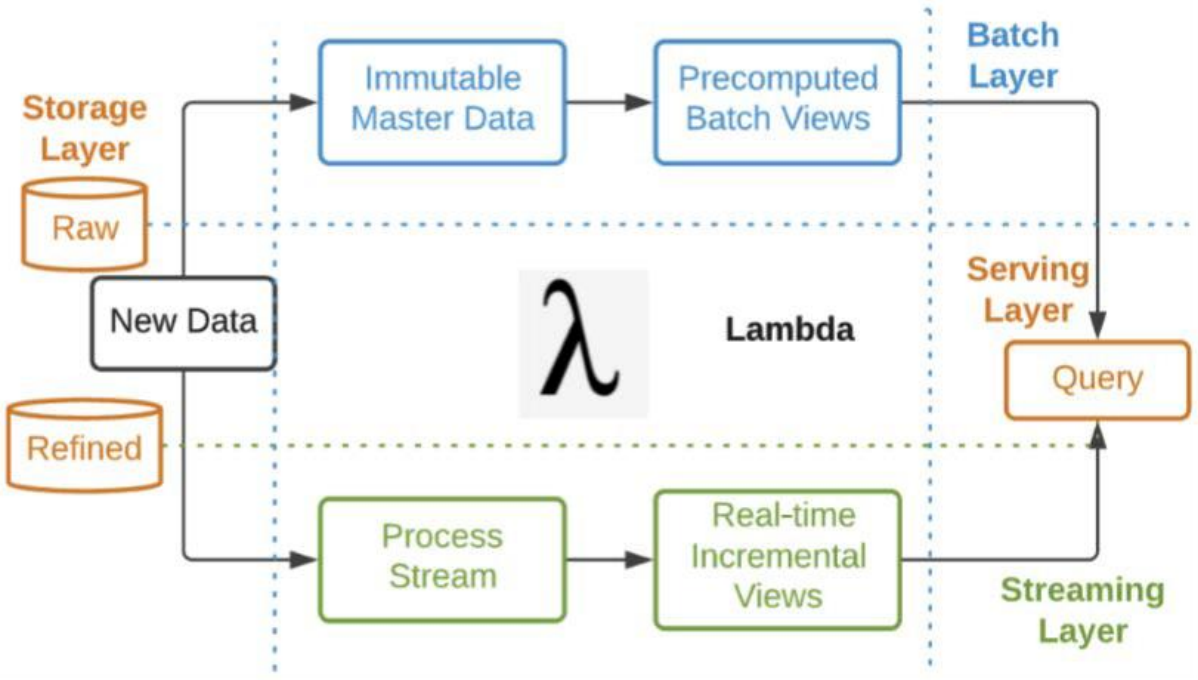
3.1. Sloj arhitekture za prikupljanje podataka (eng. Ingestion layer)

Sloj arhitekture za prikupljanje podataka (eng. Ingestion layer) u kontekstu uporabe Delta Lake formata u odnosu na prethodnu praksu doživljava fundamentalnu promjenu. Dva osnovna načina unosa podataka, skupni unos te unos pomoću podatkovnih tijekova u realnom vremenu koji su se implementirali kao dvije odvojene cjeline sada se mogu svesti na jedan unificirani model. Konkretno radi se o pogledu koji tretira dosadašnji skupni unos podataka kao poseban oblik tijeka podataka koji uključuje vremenske diskontinuitete tj. događa se periodički ili na zahtjev. Rješenje se temelji na konceptu strukturiranih tijekova podataka (eng. structured streaming) ugrađenog u sam analitički program Apache Spark. Taj zapravo jednostavan koncept svodi tijek podataka na virtualnu neograničenu tablicu (eng. unbound table). Na toj tablici se onda izvršavaju standardni upiti i njihovi rezultati (eng. result table) pohranjuju u konkretne odredišne tablice (eng. sinks). Samo pokretanje tih upita implementirano je preko mehanizma okidača (eng. triggers). Arhitektura koncepta dana je na slici 12:

Slika 12 Arhitektura sloja za prikupljanje podataka Izvor: Mahapatra, 2022

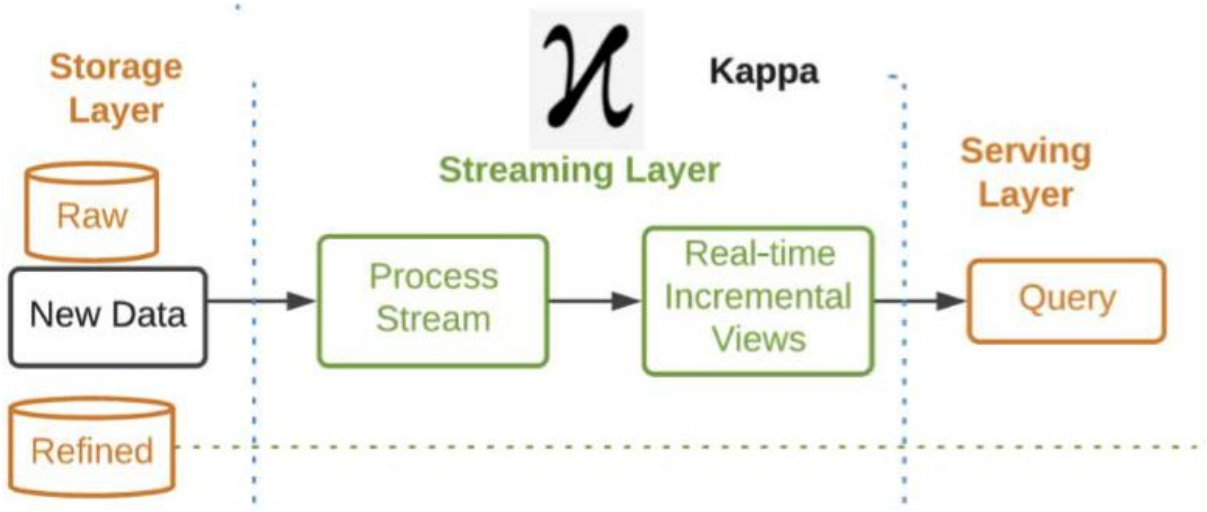
Radi se o arhitekturnom pomaku poznatom u literaturi migracija sa Lambda arhitekture koja odvojeno tretira skupni unosi i tijekove podatka na Kappa arhitekturu koja ih unificira svodeći ih na jedinstveni "tablični" model.

Lambda arhitektura je dana na slici 13:



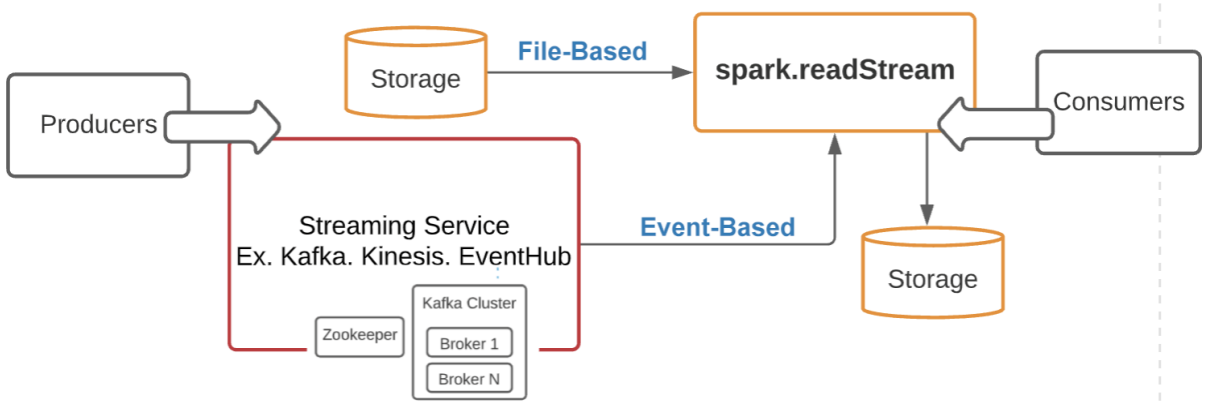
Slika 13 Prikaz lambda arhitekture sloja za prikupljanje podataka Izvor: Mahapatra, 2022

Kappa arhitektura je dana na slici 14:



Slika 14 Prikaz kappa arhitekture sloja za prikupljanje podataka Izvor: Mahapatra, 2022

Na uporabnoj razini postoje dva načina rada sa tijekovima podataka. Prvi je baziran na datotekama (eng file based) i drugi baziran na događajima (eng. event based). Za prvi je očekivano dostatan odgovarajući sloj pohrane za datoteke, dok je za drugi potrebno dodatno upotrijebiti dodatna rješenja kao što je primjerice Apache Kafka. Arhitektura ova dva načina uporabe dana je na slici 15:



Slika 15 Prikaz arhitekture bazirane na datotekama i arhitekture bazirane na događajima Izvor Mahapatra, 2022

U nastavku izlaganja koji se odnosi na konkretnu demonstraciju bit će korišten prvi način rada baziran na datotekama, zbog tehničke jednostavnosti.

3.2. Sloj arhitekture za pohranu podataka (eng. Storage layer)

Sloj arhitekture za pohranu podataka (eng. Storage layer) bazira se na već spomenutim tehnologijama nudeći veliku fleksibilnosti praktično u svim aspektima. Međutim, velika fleksibilnost znači da je potrebno pomno paziti na organizaciju podataka u jezeru podataka kako bi se izbjegao poznati problem degeneracije jezera podataka u močvaru podataka (eng. data swamp). Močvara podataka predstavlja opće poznati problem sa jezerima podataka, gdje se u jezero podataka stihijski unose velike količine podataka, koje je onda zapravo teško iskoristiti, jer su neorganizirani i nesistematizirani.

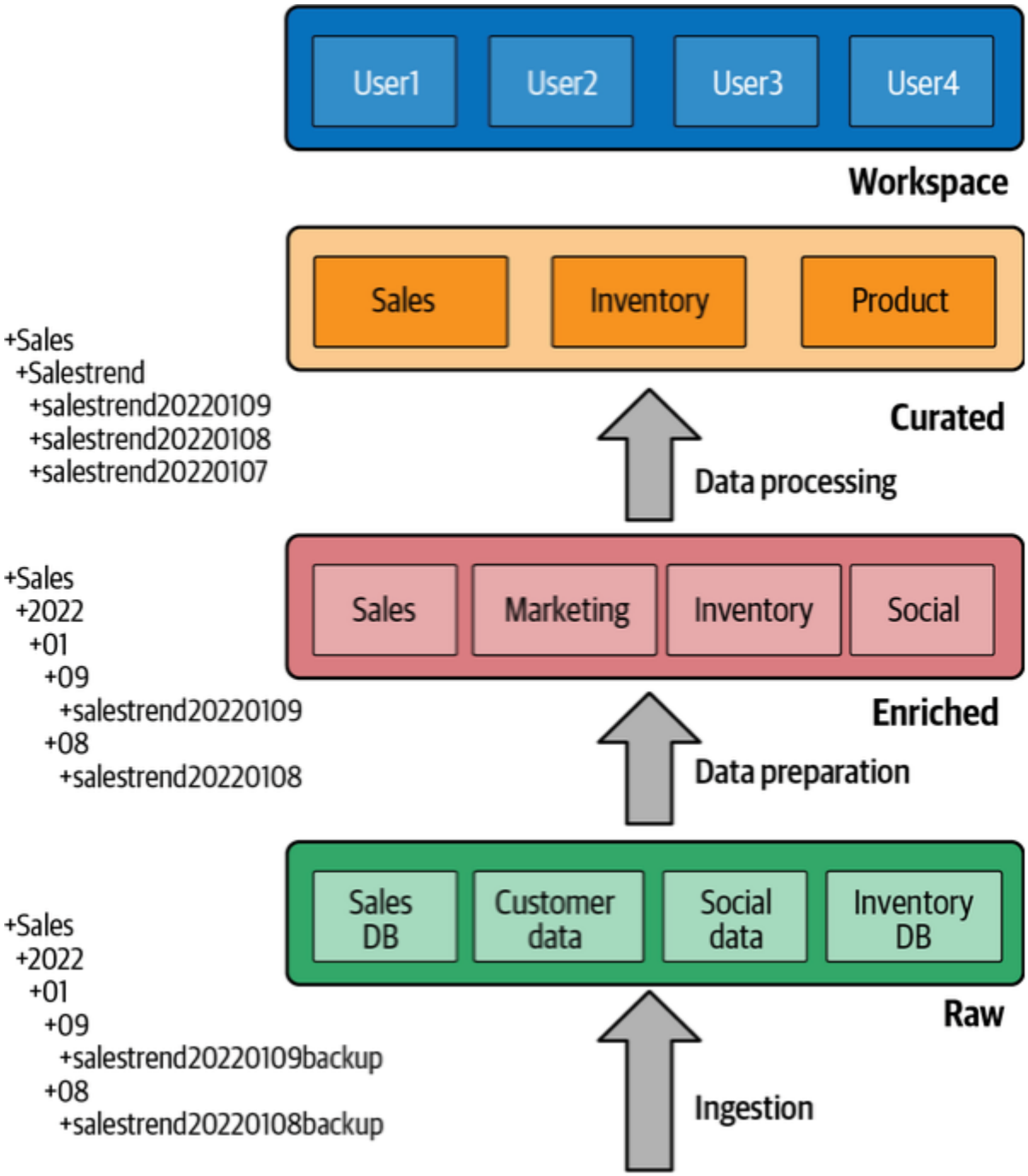
Organizacija podataka predstavlja rješenje kojim se uspješno može izbjeći problem močvare podataka. Kroz dosadašnju praksu iskristalizirala se tzv. Medallion arhitektura (eng. Medallion Architecture). Po toj arhitekturi jezero podataka organizira se u tri zone koje imaju striktnu namjenu. Prva zona je brončana zona (eng. bronze zone) i u njoj se pohranjuju podaci u izvornom obliku. Druga zona je srebrna zona (eng. silver zone) u kojoj se pohranjuju podaci u obliku strukturiranom za određeni poslovni scenarij. Treća zona je zlatna zona (eng. gold zone) u kojoj se nalaze podaci spremni za uporabu.

Brončana zona je dakle mjesto pohrane podataka u izvornom obliku netom nakon preuzimanja iz raznih izvora. Najčešće se unutar te zone podaci dodatno organiziraju po izvorima i po vremenu unosa u sustav.

Srebrna zona služi kao mjesto na kome se pohranjuju podaci koji su prošli određene transformacije. Prva transformacija je strukturiranje podataka, stoga što podaci u brončanoj zoni mogu biti nestrukturirani, polustrukturirani i potpuno strukturirani. U sklopu te transformacije podacima se pridjeljuje definicija njihove strukture - shema (eng. schema definition), koja je prilagođena za određeni poslovni scenarij tj.u praktičnom smislu podaci se svode na tablični oblik. Druga transformacija odnosi se na čišćenje podataka (eng. Data cleansing), kojom se eliminiraju razne vrste pogrešaka. Treća transformacija odnosi se na optimiziranje podataka (eng.) u strukturalnom smislu, kao i što se tiče odabira odgovarajućeg formata podataka, temeljem konkretnih scenarija njihove uporabe.

Zlatna zona (eng. gold zone) sadrži podatke u obliku spremnom za uporabu, koji onda kao takvi imaju najveću dodanu vrijednost u poslovnom smislu. Ti podaci se izravno koriste u raznim analitičkim alatima, prezentiraju na kontrolnim pločama (eng. dashboards) i sl. Tipično nad tim podacima obavljaju se razne agregacije, filtriranja, korelacije sa drugim skupovima podataka, te razne vrste kompleksnih obrada.

Konkretan primjer organizacije jezera podataka po Medallion arhitekturi dan je na slici 16:

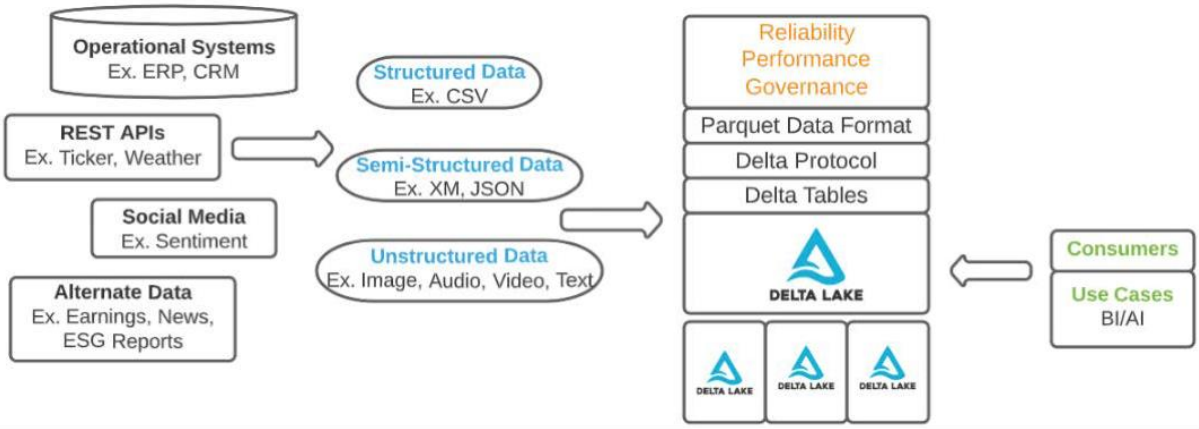


Slika 16 Prikaz medallion arhitekture sloja za pohranu podataka Izvor: Gopalan, 2022

U nastavku izlaganja koji se odnosi na konkretnu demonstraciju će za sloj pohrane biti korišten datotečni sustav jer je evidentno tehnički najjednostavniji.

Delta lake format kao druga glavna sastavnica ima sve potrebne tehničke značajke, elaborirane u dosadašnjem izlaganju, za implementaciju skladišta podataka na jezerima podataka. Na razini implementacije on se sastoji od tri glavna dijela. Prvi je Parquet format podataka (eng. Parquet

data format) utemeljen pa principu pohrane i organizacije podataka orijentiranom na stupce. Drugi dio je Delta protokol (eng. Delta protocol) za rad sa datotekama na sloju za pohranu. Treći i u uporabnom smislu najvažniji dio je tablična struktura Delta tablica (eng. Delta tables), koji omogućava rad s podacima u uobičajenom tabličnom, potpuno strukturiranom obliku. Korisno je istaknuti da sve to rezultira potpunom unifikacijom načina rada s podacima na sloju pohrana gdje se neovisno o bilo kojoj značajki podataka u svim zonama jezera podataka koristi Delta format kako je konceptualno prikazano na slici 17:



Slika 17 Konceptualni prikaz delta formata Izvor:Mahapatra, 2022

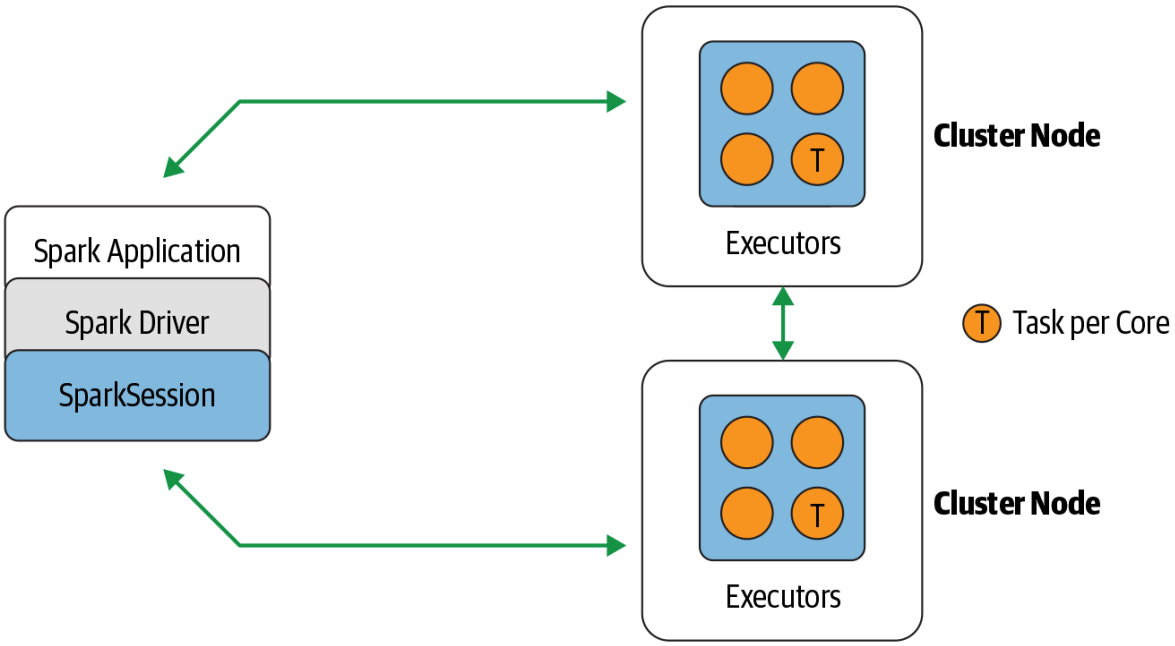
Najvažnije značajke organizacije jezera podataka i samog Delta formata bit će konkretno prikazane u nastavku izlaganja.

3.3. Sloj arhitekture za obradu podataka (eng. Data processing layer)

Sloj arhitekture za obradu podataka (eng. Data processing layer) u potpunosti se zasniva na analitičkom programskom alatu Apache Spark 3.x. Sve njegove značajke nastale su kao rezultat rješavanja problema sa prethodnim analitičkim programima kao što je to Apache Hadoop MapReduce, kao i rapidno rastućih potreba prakse primarno zbog sve veće orijentacije na jezera podataka.

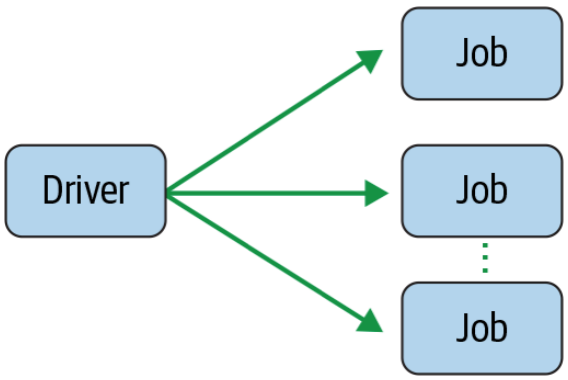
Pojednostavljeno, na razini implementacije arhitektura se sastoji od pet glavnih elemenata. Prvi je Spark aplikacija (eng. spark application) koja koristi sve mogućnosti Apache Spark sustava. Drugi je objekt Spark sesije (eng. SparkSession Object) sa Apache Spark sustavom, kojeg kreira Spark aplikacija pomoću trećeg elementa. Taj treći element je Spark pogonski modul

(eng. Spark Driver) koji izravno komunicira sa skupom povezanih poslužitelja na kojima radi Apache Spark klaster (eng. Spark Cluster). Spark aplikacija preko programskog sučelja objekta Spark sesije pokreće obradu podataka na Spark poslužiteljima (eng. cluster nodes) kako je prikazano na slici 18:



Slika 18 Arhitektura sloja za obradu podataka na razini implementacije Izvor: Damji, Wenig, Das, Lee, 2020

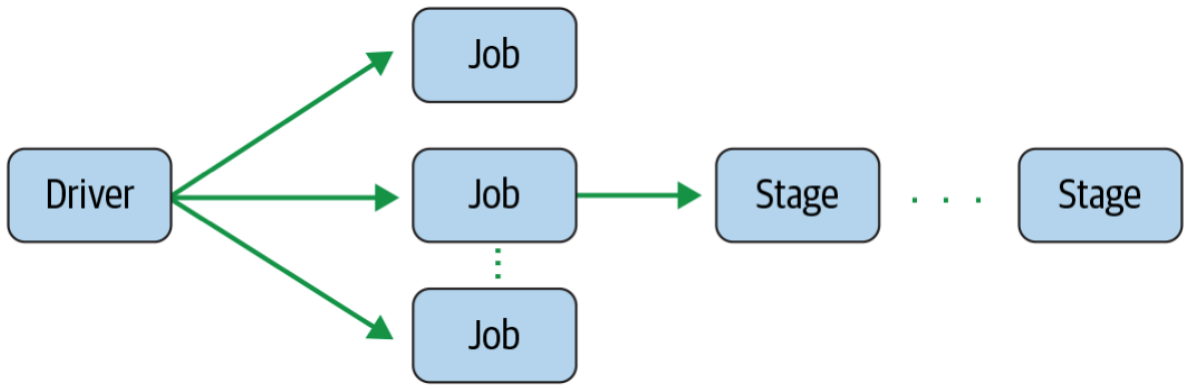
Spark posao (eng. Spark Job) nastaje tako da sukladno potrebama Spark Aplikacije, Spark pogonski modul kreira određeni broj Spark poslova koji se izvršavaju paralelno kako je prikazano na slici 19:



Slika 19 Prikaz paralelnog izvršavanja spark job-ova Izvor: Damji, Wenig, Das, Lee, 2020

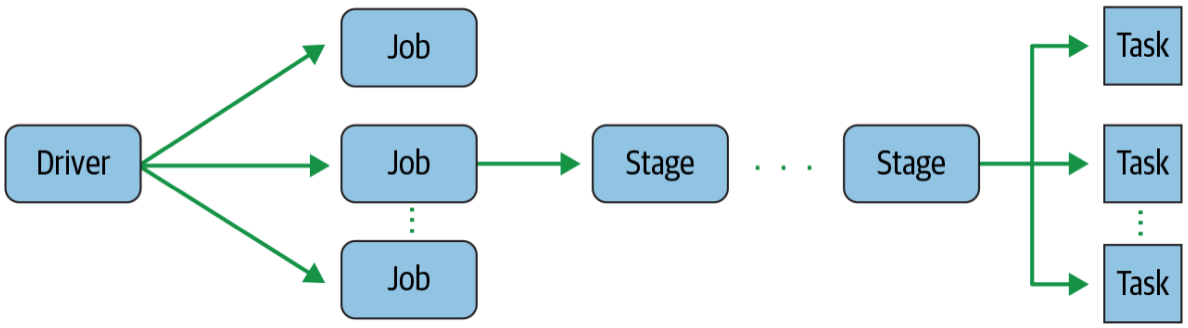
Spark faze (eng Spark Stages) predstavljaju sastavnice izvršenja Spark poslova, međusobno povezane po matematičkom modelu usmjerenog acikličkog grafa (eng. Direct Acyclic Graph - DAG). Na razini implementacije ta struktura predstavlja plan izvršenja (eng. execution plan)

Spark posla, pri čemu se Spark faze obrade formiraju primarno po kriteriju mogućnost paralelnog odvijanja pojedinih operacija obrade podataka, kako je prikazano na slici 20:



Slika 20 Prikaz formiranja Spark faze obrade Izvor: Damji, Wenig, Das, Lee, 2020

Spark zadaci (eng Spark Task) su pak jedinice izvršenja (eng. unit of execution) Spark poslova, koje se ovisno o raspoloživim procesnim kapacitetima mogu izvršavati maksimalno paralelno. Pri tome se jedan zadatak može izvršavati na samo jednoj procesorskoj jezgri (eng. processor core) i samo jednoj podatkovnoj particiji (eng. data partition). Ovakav način rada prikazan je na slici 21:



Slika 21 Prikaz paralelnog izvršavanja Spark zadataka Izvor: Damji, Wenig, Das, Lee, 2020

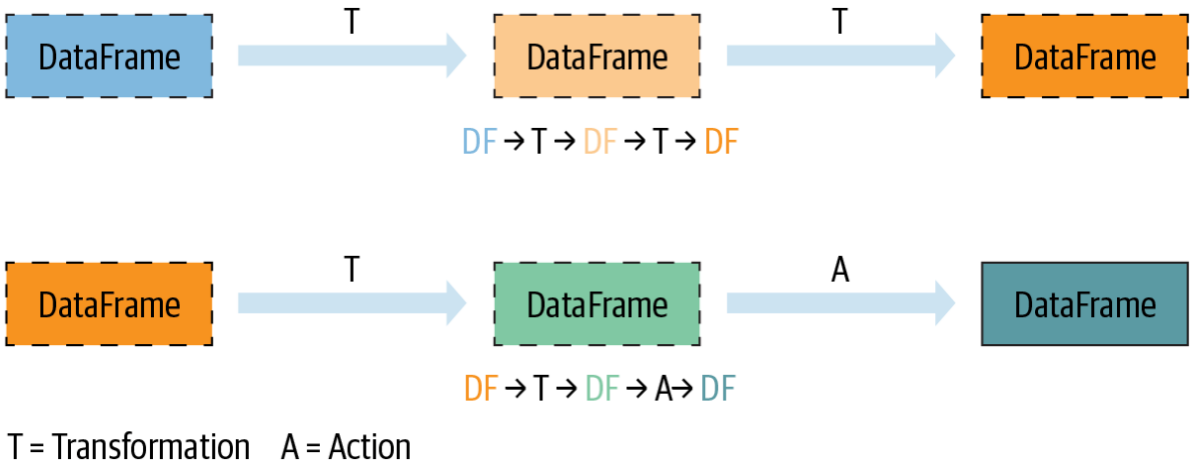
Operacije na podacima dijele se u dvije skupine. Prva skupina su transformacije podataka (eng. data transformations) pri čemu se jedna struktura podataka koja je na programskoj razini uporabe predstavljena DataFrame objektom transformira u drugu DataFrame strukturu. Implementacija DataFrame objekta utemeljena ja na već spomenutoj RDD strukturi, koja predstavlja temelj paralelne obrade na Spark analitičkom programu. Pri tome svakako treba istaknuti da je takav koncept transformacija jamči nepromjenljivost (eng. immutability) podataka. Princip rada je takav da transformacija ne modificira ulazne podatka koje treba

transformirati, nego kreira novu, potpuno odvojenu kopiju podataka kao rezultat. Također, sustav bilježi slijed (eng. lineage) transformacija. Raspoložive su transformacija podataka select, join, filter, groupBy i orderBy, koje evidentno, praktično izravno korespondiraju sa svim glavnim elementima SQL upita.

Također je korisno istaknuti da se operacije transformacija izvode po modelu lijene evaluacije (eng. lazy evaluation), što praktično znači da se neka operacija u planu izvršavanja izvodi tek kad je njen rezultat stvarno neophodan. To znači da Spark analitički programski alat može čak tijekom izvođenja plana izvršenja, ovisno o stanju obrade dodatno modificirati redoslijed izvođenja transformacija u cilju dodatne efikasnosti u radu.

Druga skupina oparacija su akcije na podacima (eng. data actions), čije izvođenje ima za posljedicu pokretanje svih do tada neizvršenih transformacija, čije je izvršenje odgođeno kao posljedica primjene modela lijene evaluacije. Raspoložive su akcije na podacima show, take, count, collect i save, koje zapravo predstavljaju uobičajene operacije na podacima sveprisutne i u drugim sustavima.

Prethodno prezentirani način rada prikazan je na slici 22:



Slika 22 Prikaz toka operacija na podacima Izvor: Damji, Wenig, Das, Lee, 2020

Istaknimo da spomenuta nepromjenjivost podataka u kombinaciji sa bilježenjem slijeda transformacija jamči potpunu konzistenciju i robusnost obrade tj. otpornost na pogreške (eng. fault tolerance), stoga što uvijek postoji kopija podataka za svaki među rezultat obrade, koji se u slučaju pogreške/gubitka može ponovo rekonstruirati iz drugih među rezultata, sukladno zabilježenom slijedu transformacija.

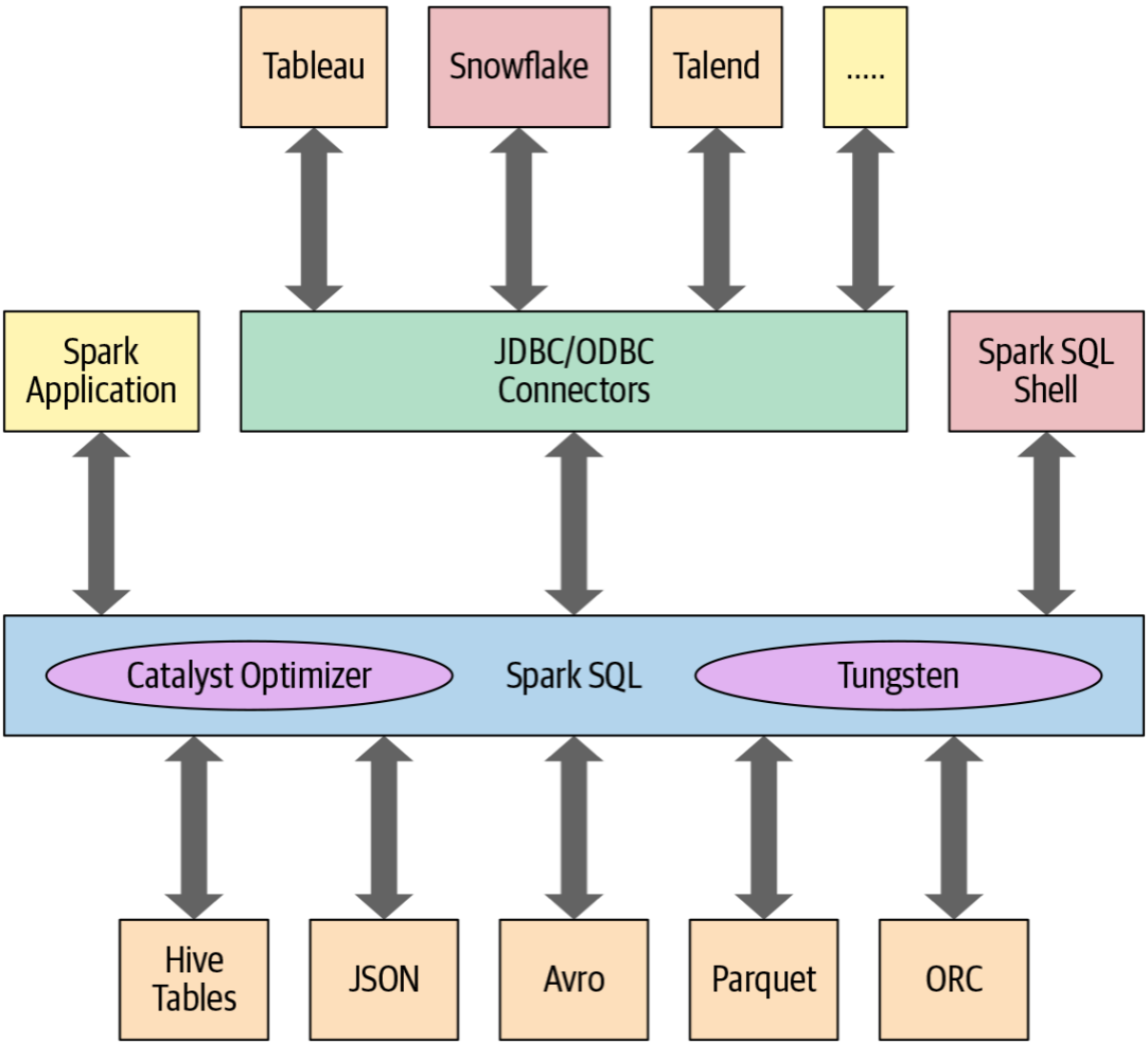
Gore navedeni elementi obrade koje omogućava Spark analitički program bit će konkretno demonstrirani u nastavku izlaganja.

3.4. Sloj arhitekture za posluživanje podataka (eng. Serving layer)

Sloj arhitekture za posluživanje podataka (eng. Serving layer) u kontekstu primjene Spark analitičkog programskog alata u kombinaciji sa formatom Delta Lake odnosi se primarno na modul koji se naziva Spark SQL. Taj modul omogućava vrlo efikasan rad sa strukturiranim podacima na dva načina: Prvi način je programski model utemeljen na već spomenutom DataFrame objektu, za koji onda postoje konkretna programska sučelja za najkorištenije programske jezike kao što su primjerice Java, Python i Scala u kojima se onda mogu izrađivati Spark aplikacije. Drugi način je upotreba interaktivnih SQL upita što predstavljam temeljni način funkcioniranja praktično svih alata koji se koriste u području poslovne inteligencije. Ovaj način rada je implementiran tako što je u Spark analitički program dodana podrška za, dugi niz godina, potpuno standardizirana programska sučelja za rad sa relacijskim bazama JDBC/ODBC. Raspoloživost ovih sučelja omogućila je izravnu vezu sa praktično svim analitičkim alatima od kojih su svakako najrašireniji Power BI, Tableau, te u segmentu otvorenog koda Weka.

Svakako treba istaknuti dvije temeljne sastavnice Spark SQL modula. Prva se naziva Catalyst Optimizer za optimizirano izvršenje SQL upita tj. obrađivač upita (eng. query processor i vrši kroz kompleksan postupak pretvorbu SQL upita u plan izvršavanja. Druga komponenta se naziva Tungsten i služi upravo za prije opisano izvršenje plana izvršavanja.

Arhitektura Spark SQL modula dana je na slici 23:



Slika 23 Prikaz arhitekture Spark SQL modula Izvor: Damji, Wenig, Das, Lee, 2020

Mogućnosti upotrebe Spark SQL modula u kombinaciji sa analitičkim alatom otvorenog koda Weka bit će prezentirani u nastavku izlaganja.

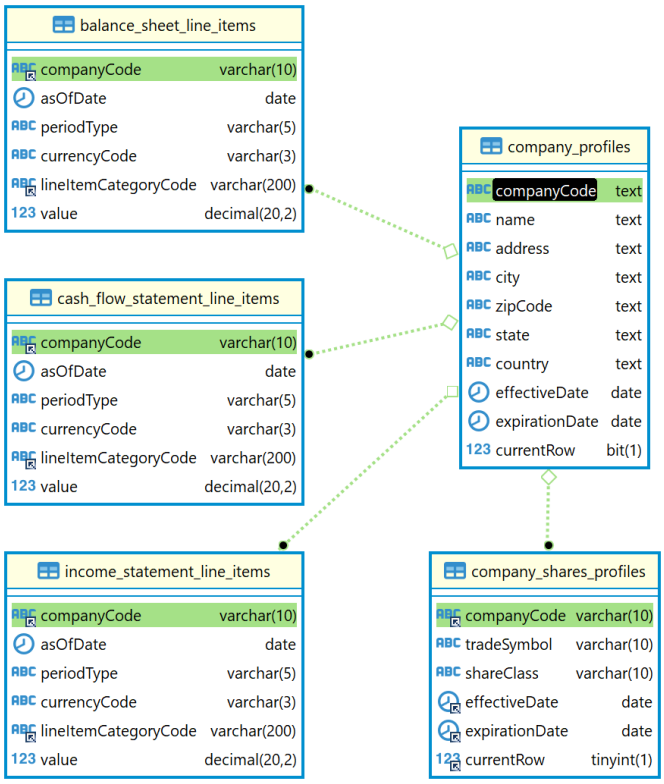
4. Implementacija i primjena skladišta podataka utemeljenog na jezeru podataka na podacima cijena tržišta kapitala

Praktična primjena svakog sustava, osim što naravno mora zadovoljavati svim svojim tehničkim značajkama, svakako ovisi i o jednostavnosti uporabe. Na primjerima koji slijede u ovom poglavlju bit će pokazano da se većina operacija na podacima u skladištu podataka utemeljenom na jezeru podataka može obaviti uporabom SQL-a pomoću Spark SQL modula, odnosno da je za ostale operacije dostatno solidno temeljno poznavanje programskog jezika Python. Treba naglasiti da je potrebno dobro poznavati način funkcioniranja analitičkog programskog alata Apache Spark (njegova programska sučelja u Pyspark modulu), kao i Delta Lake formata kako bi se izbjeglo nepotrebno programiranje funkcija koje su već raspoložive, ali i maksimalno optimirane za distribuirani rad sa velikim količinama podataka.
Pored analitičkog programskog alata Apache Spark za implementaciju skladišta podataka utemeljenog na jezeru podataka korišteni su:

- Weka - poznati i često korišteni softver otvorenog koda namijenjen za rudarenje podataka i strojno učenje
- RStudio - glavni razvojni alat za široko etablirani analitički programski jezik R
- Jupyter Lab – alat temeljen na konceptu interaktivnih bilježnica koje omogućavaju da se u jednoj datoteci nalazi određeni programski kod u kombinaciji sa rezultatima njegovog izvršavanja (Dostupna je potpuna integraciju sa modulima Apache Spark SQL i Pyspark)
- Dbeaver - alat otvorenog koda specijaliziran za sve aspekte primjene SQL, od kreiranja strukture tablica, do manipulacije podacima na raznim izvorima podataka (Konkretno korištena je integracija sa MySQL poslužiteljem i Apache Spark SQL modulom)

4.1. Obilježja korištenih podataka

Primjer izvora strukturiranih podataka implementiran je u vidu relacijske baze reportdb postavljene na MySQL 8 poslužitelju. ER dijagram dan je na slici 24:

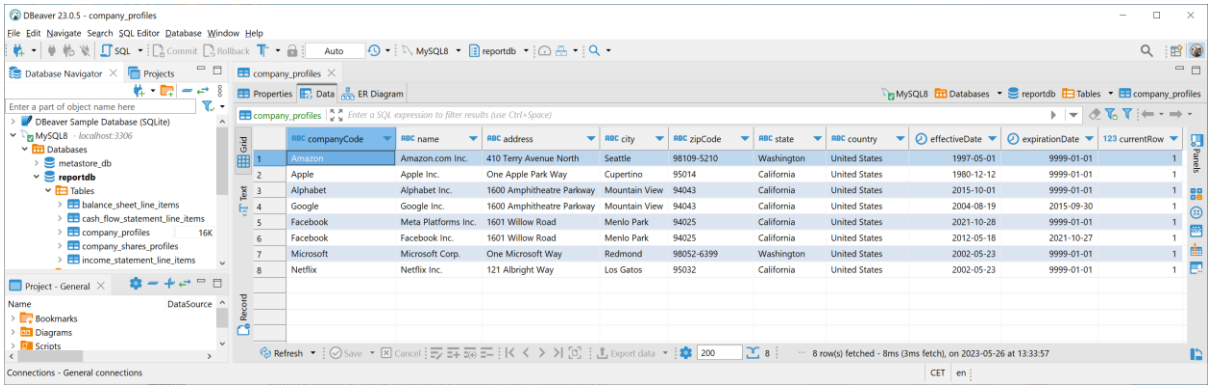


Slika 24 ER dijagram baze podataka reportdb Izvor:Ekranski prikaz računala od autora

Iz dijagrama na slici 24 je vidljivo da se radi o klasično strukturiranoj izvještajnoj bazi, djelomično denormaliziranoj, pri čemu je referencijalni integritet uspostavljen preko prirodnih (ujedino govorećih) ključeva (eng. natural keys). U sadržajnom smislu baza se sastoji od izvezenih stavki temeljnih računovodstvenih dokumenata bilance, računa dobiti i gubitka, te izvještaja o novčanom tijeku (eng. balance sheet line items, income statement line items, cash_flow_statement line items). Također su izvezene dvije tablice matičnih podataka koje se odnose na osnovne podatke o tvrtkama, i njihovim vrijednosnicama. Specifikum ovih dvaju tablica jest da su temporalnog karaktera tj. u njima su zabilježene promjene na podacima relevantne za burzovno poslovanje. Općenito temporalni karakter podataka vezan je za implementaciju vremenskih baza podataka u kojima su pohranjeni podaci koji se odnose na vremenske instance tj. vremenske vrste podataka. Takvi podaci uključuju informacije koje se odnose na prošlo, sadašnje i buduće vrijeme. Ovdje se radi o tablicama iz jednovremenske baze podataka, koja uključuje samo jednu vremensku dimenziju (eng. uni-temporal database). Pohranjena dimenzija vremena je valjano vrijeme (eng. valid time). To vrijeme odnosi se na životni ciklus poslovnog objekta u stvarnom vremenu (npr. datum izlaska na burzu). Koriste se polja effectiveDate (od kad su vrijednosti u kolonama važeće), expirationDate (do kad su

vrijednosti u kolonama važeći) i oznaka trenutno važećeg retka currentRow (važeći podaci kojima je expirationDate postavljen na datum u dalekoj budućnosti).

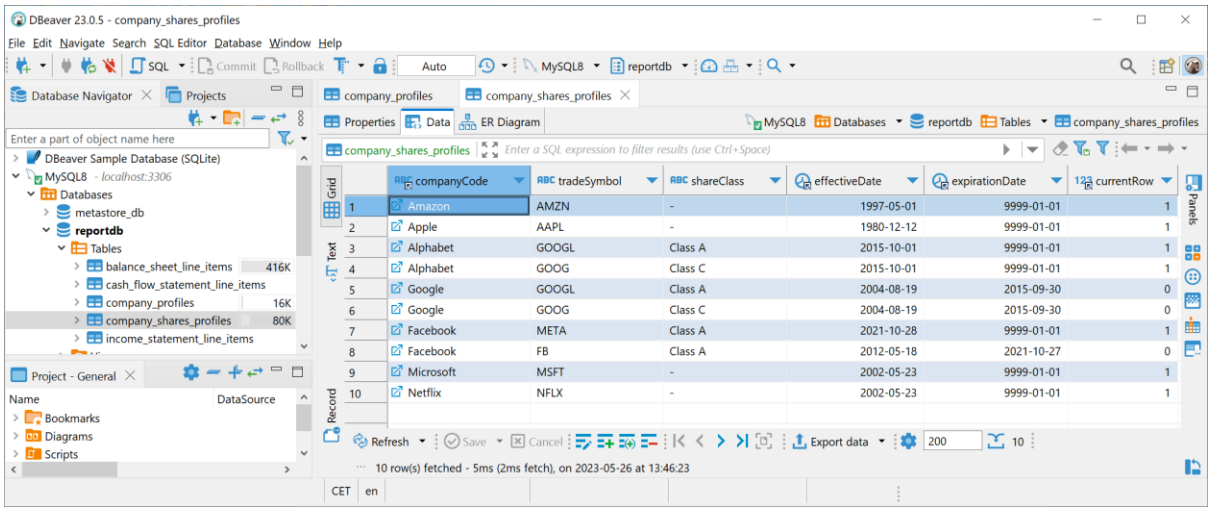
Sadržaj tablice company_profiles dohvaćen pomoću alata DBeaver prikazan je na slici 25:



Slika 25 Prikaz sadržaja tablice company_profiles Izvor:Ekranski prikaz računala od autora

Vidljivo je da se prate promjene koje su relevantne za burzovno poslovanje. Prvi slučaj je kompanija Google Inc., koju je u sklopu restrukturiranja zamijenila kompanija Alphabet Inc. kao pravni slijednik. Drugi slučaj je kompanija Facebook Inc., koja je u jednom trenutku samo promijenila ime u Meta Platforms Inc., pa je posljedično prirodni ključ companyCode ostao isti, jer se radi o istoj pravnoj osobi. Također effectiveDate za prvi zapis vezan za kompaniju nije datum osnivanja kompanije, nego datum izlaska na burzu.

Druga temporalna tablica company_shares_profiles, zaokružuje skup referentnih podataka vezan za burzovno poslovanje koje se odnosi na dioničke simbole (eng. trade symbol - ticker), koji predstavljaju temeljni identifikator vrijednosnica kojima se trguje. Sadržaj ove tablice dohvaćen pomoću alata DBeaver prikazan je na slici 26:



Slika 26 Prikaz tablice company_shares_profiles Izvor:Ekranski prikaz računala od autora

Ovdje se prate promjene vezane na dioničke simbole. Tako je prilikom promjene imena kompanije Facebook Inc. došlo i do promjene dioničkog simbola iz FB u META. S druge strane može se vidjeti da je u sklopu restrukturiranja, kompanija Alphabet Inc. preuzela dioničke simbole od kompanije Google Inc. Po burzovnoj praksi, bez obzira na vrstu promjene, dionički simbol u trenutačnoj uporabi preuzima svu povijest dioničkog simbola koji je prije korišten.

Kao primjer izvora polu strukturiranih i ujedno referentnih podataka iskorišten je internetski servis Yahoo Finance, koji omogućava dohvat svih relevantnih informacija vezano za burzovno poslovanje. Dohvat podataka implementiran je preko REST programskog sučelja koje koristi polu strukturirani JSON format zapisa. Sukladno uobičajenoj praksi za komunikaciju sa REST programskim sučeljima, spomenutog internetskog servisa, korišten je Python programski paket yahooquery. Glavna prednost uporabe ovog programskog paketa i njemu sličnih (npr. yfinance, yahoo_fin) je što on uključuje gotove funkcije koje omogućavaju dohvat različitih vrsta burzovnih podataka, koje vraćaju gotove Panda dataframe Python objekte. Time se ukida potreba za dosta zahtjevnim ručnim programiranjem funkcija koje iz JSON formata ekstrahiraju tražene podatke. Osim toga, programsko sučelje Pyspark modula omogućava kreiranje strukturno ekvivalentnog spark dataframe Python objekta izravno iz spomenutog Pandas oblika.

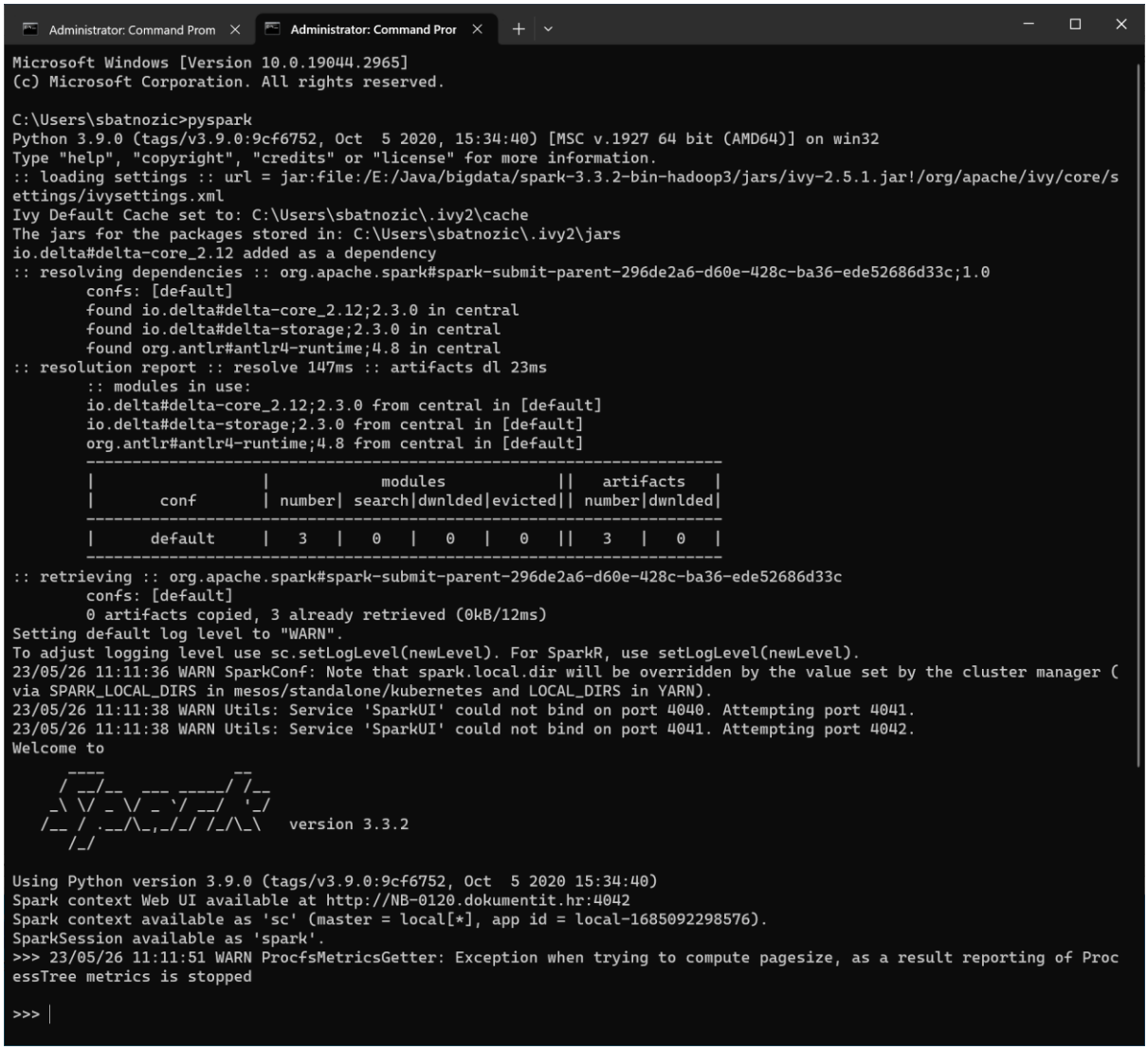
Kao primjer izvora nestrukturiranih podataka iskorišteni su transkripti poziva o zaradi (eng. Earnings Calls) (u njima javno poduzeće raspravlja o financijskim rezultatima izvještajnog razdoblja). Preuzeti su ručno s web stranica internetskog servisa capedge.com u formatu PDF datoteka.

4.2. Implementacija skladišta podataka utemeljenog na jezeru podataka

Postavka sustava je tehnički maksimalno pojednostavljena što znači da je Apache Spark 3.3.2 korišten izravno instaliran na operacijskom sustavu (uz uporabu OpenJDK 8) kako bi se moglo na jednostavan način vršiti podešavanje svih parametara sustava, ali i na najnižoj razini po potrebi izvršiti uvid u njegovo funkcioniranje. Provedeno je testiranje na dva operacijska sustava Windows 10 Professional Edition i Ubuntu Linux 22.04. Korištena inačica Apache Sparka pokazala je potpunu stabilnost u radu u oba slučaja.

Kao i svi analitički programi otvorenog koda Apache Spark-u je urođeno da podržava rad na Linux operacijskom sustavu. Postoje određene razlike između funkcioniranja Linux ext* datotečnog sustava i Windows NTFS datotečnog sustava na sistemskoj razini, poglavito u segmentu kontrole prava pristupa. Zbog toga su na Windows 10 sustavu, bila potrebna manja dodatna podešavanja.

Prezentirani rezultati u nastavku izlaganja dobiveni su na Windows 10 operacijskom sustavu. Primjerice način rada Apache Spark sustava može se vidjeti pokretanjem Pyspark modula danom na slici 27:



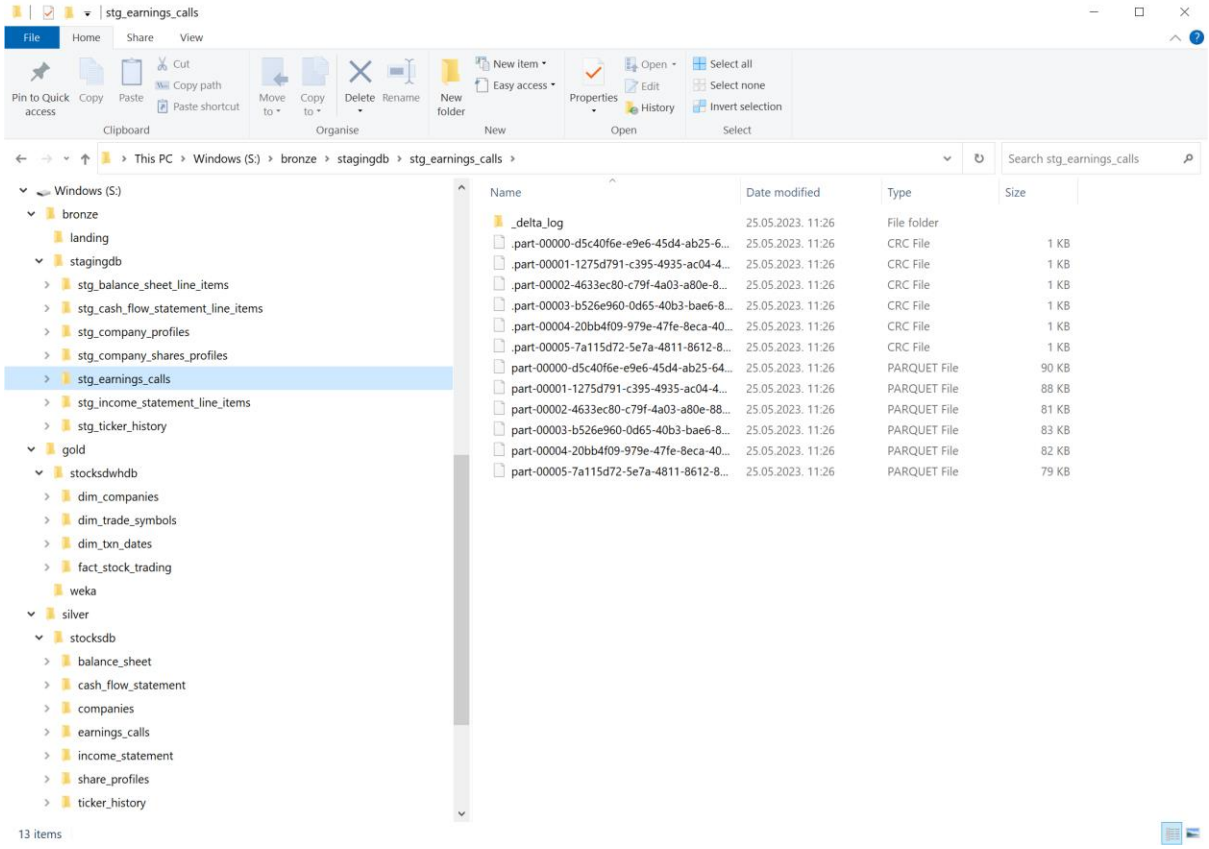
Slika 27 Prikaz pokretanja Pyspark modula Izvor:Ekranski prikaz računala od autora

Na slici 27 je jasno vidljivo da je postavljena podrška za Delta Lake format, ali i upozorenje na sistemskoj razini proizašlo iz nekompatibilnosti između Windows i Linux operacijskih sustava u načinu upravljanja procesima, koje rezultira pogrešnim upozorenjem, koje se može

ignorirati. U posljednjoj inačici Apache Spark 3.4 taj benigni problem je riješen. Spomenuta inačica nije korištena, stoga što je podrška Delta Lake format za tu inačicu još uvijek u razvojnoj fazi. Činjenica da je taj benigni problem na Windows operacijskom sustavu ipak otklonjen, predstavlja nagovještaj promjene u načinu primjene analitičkih programskih alata općenito. Radi se o tome da dolazi nova generacija prijenosnih računala opremljenih ARM procesorima i GPU procesorima visokih performansi, čiji predvodnik je svakako Apple sa M1 Pro, M1 Max, and M2 Pro, M2 Max serijom prijenosnih računala što otvara prostor za čitav niz primjena posebno u području dubokog učenja (eng. deep learning), bez potrebe za uporabom grozda (eng. cluster) poslužitelja u oblaku. Testiranja su provedena na prijenosnom računalu visokih performansi HP ZBook G4, koji ima 8 logičkih procesora i 64 GB radne memorije, te SSD disk.

Za implementaciju sloja pohrane iskorišten je običan datotečni sustav stoga što se opseg rada odnosi na prezentaciju temeljnih koncepta i principa rada skladišta podataka utemeljenog na jezeru podataka. To isključuje čitav niz slučajeva, kao što je pohrana velikih količina podataka, distribuirana obrada i sl., za koje bi rješenja kao štu su Hadoop HDFS, AWS S3, MinIO bili relevantni.

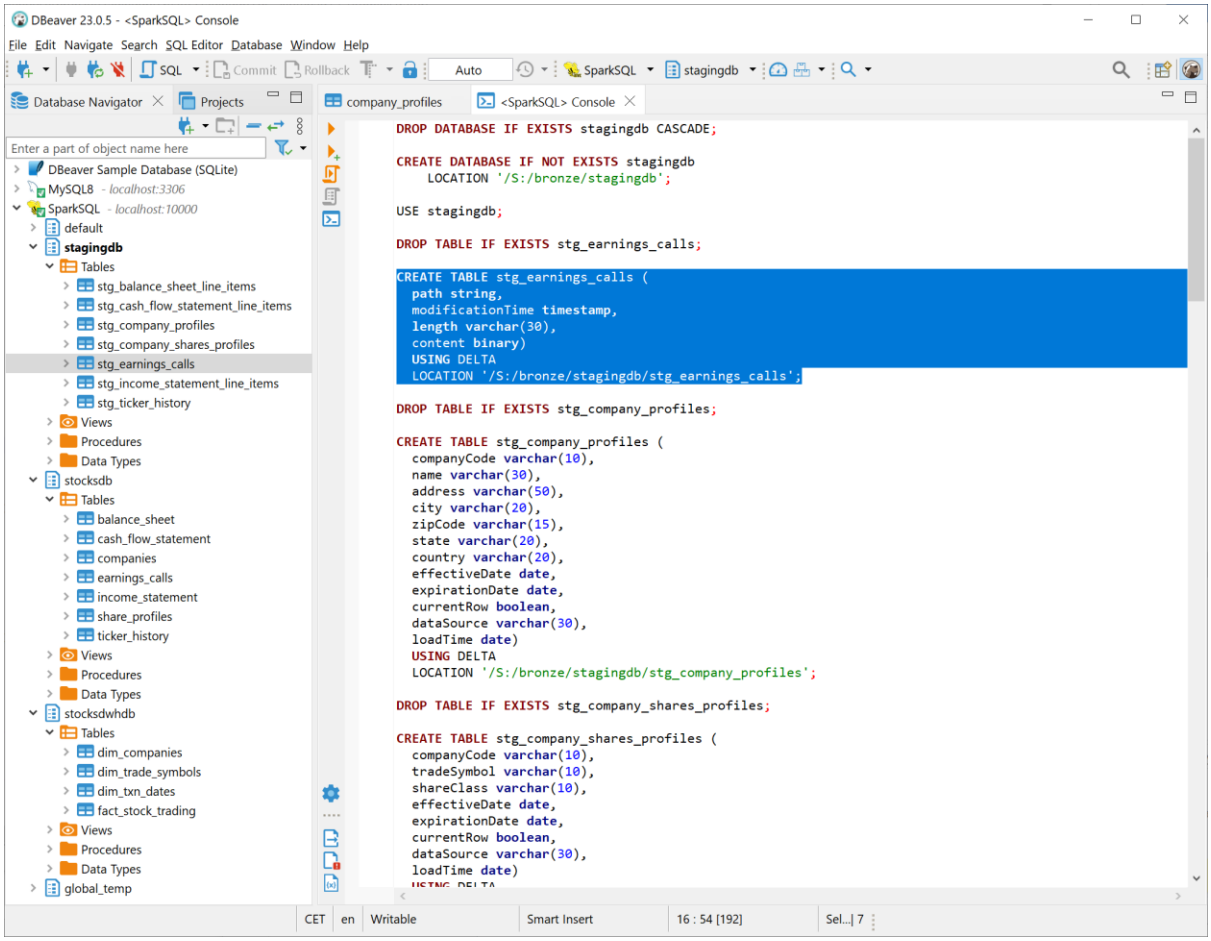
Sukladno tome struktura fizičke pohrane na datotečnom sustavu skladišta podataka utemeljenog na jezeru podataka dana je na slici 28:



Slika 28 Prikaz strukture pohrane na datotečnom sustavu skladišta podataka utemeljenog na jezeru podataka Izvor:Ekranski prikaz računala od autora

Iz slike 28 je u potpunosti razvidna primjena koncepta jezera podataka da se sve pohranjuje u strukturi mapa i datotekama. Jezero je po Medallion arhitekturi organizirano u tri vršne mape za brončani, srebrni i zlatni sloj (mape bronze, silver, gold). Također može se vidjeti da se unutar tih slojeva nalaze pohranjene baze podataka u pripadajućim mapama, u kojima se nalaze mape pojedinih tablica. Također može se vidjeti struktura jedne tablice (stg_earnings_calls) pohranjene u Delta Lake formatu.

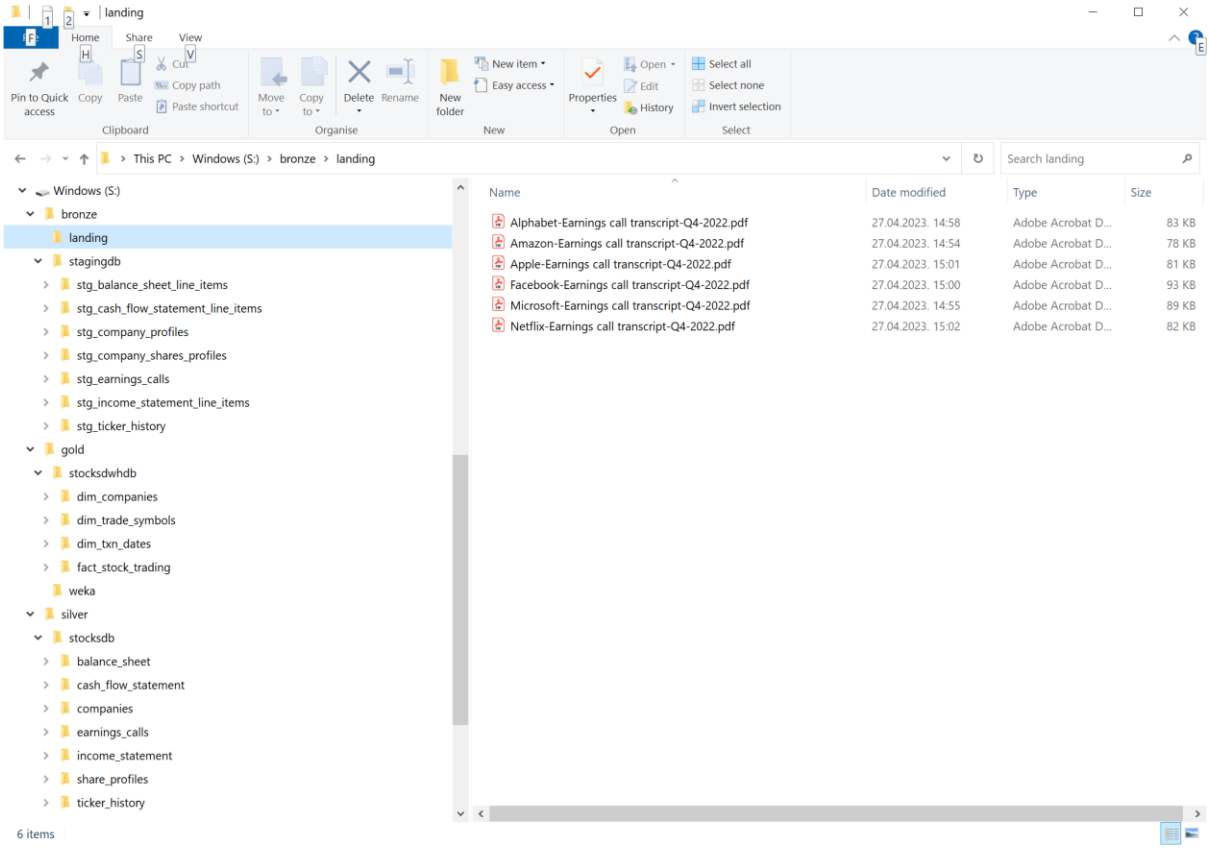
Pored fizičkog sloja pohrane postoji i funkcionalni dio koji omogućava manipulaciju podacima. On se može najbolje vidjeti uz pomoć alata DBeaver kao što je prikazano na slici 29:



Slika 29 Prikaz funkcionalnog dijela pohrane Izvor:Ekranski prikaz računala od autora

Na slici 29 je vidljivo da u funkcionalnom smislu jezero podataka uz pomoć Spark SQL modula omogućava da se rad s podacima u dijelu kreiranja njihove strukture u potpunosti svede na uporabu SQL naredbi. Posebno je označena prethodno spomenuta tablica stg_earnings_calls gdje se u sklopu CREATE TABLE SQL naredbe specificira da će se koristiti Delta Lake format i lokacija koja je bila vidljiva na prethodnoj slici za prikaz fizičkog sloja pohrane. Jednako tako je vidljivo da je baza stagingdb kreirana pomoću standardne CREATE DATABASE SQL naredbe, uz specifikaciju lokacije u jezeru podataka, također vidljive na slici 29 .

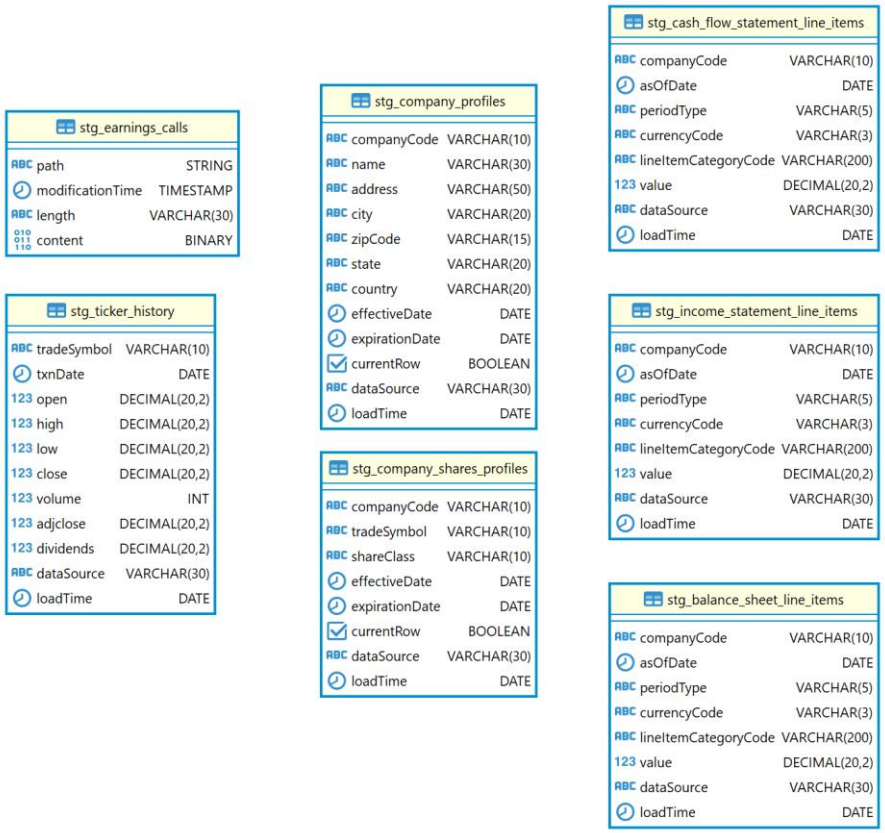
Brončani sloj (eng. bronze layer) ima primarnu funkciju prijema i pohrane podataka u potpuno neizmijenjenom i/ili čak izvornom obliku. U općem slučaju on se sastoji landing zone implementirane u vidu mape landing, gdje se pohranjuju podaci u izvornom obliku. Za primjer takvog slučaja iskorištene su PDF datoteke sa transkriptima poziva o zaradi kao što se može vidjeti na slici 30:



Slika 30 Prikaz landing mape brončanog sloja Izvor:Ekranski prikaz računala od autora

Uporaba Delta Lake formata omogućava potpunu konsolidaciju svih podataka koji ulaze u jezero podataka. Za to je predviđena staging zona implementirana u obliku baze podataka stagingdb koja sadržava tablice u Delta Lake formatu. Podaci iz svih izvora se pohranjuju u posebnim tablicama u tom konsolidiranom formatu. Razlog za takav pristup je unificirana i vrlo fleksibilna daljnja manipulacija podacima.

ER dijagram stagingdb baze dan je na slici 31:



Slika 31 Prikaz ER dijagrama stagingdb baze podataka Izvor:Ekranski prikaz računala od autora

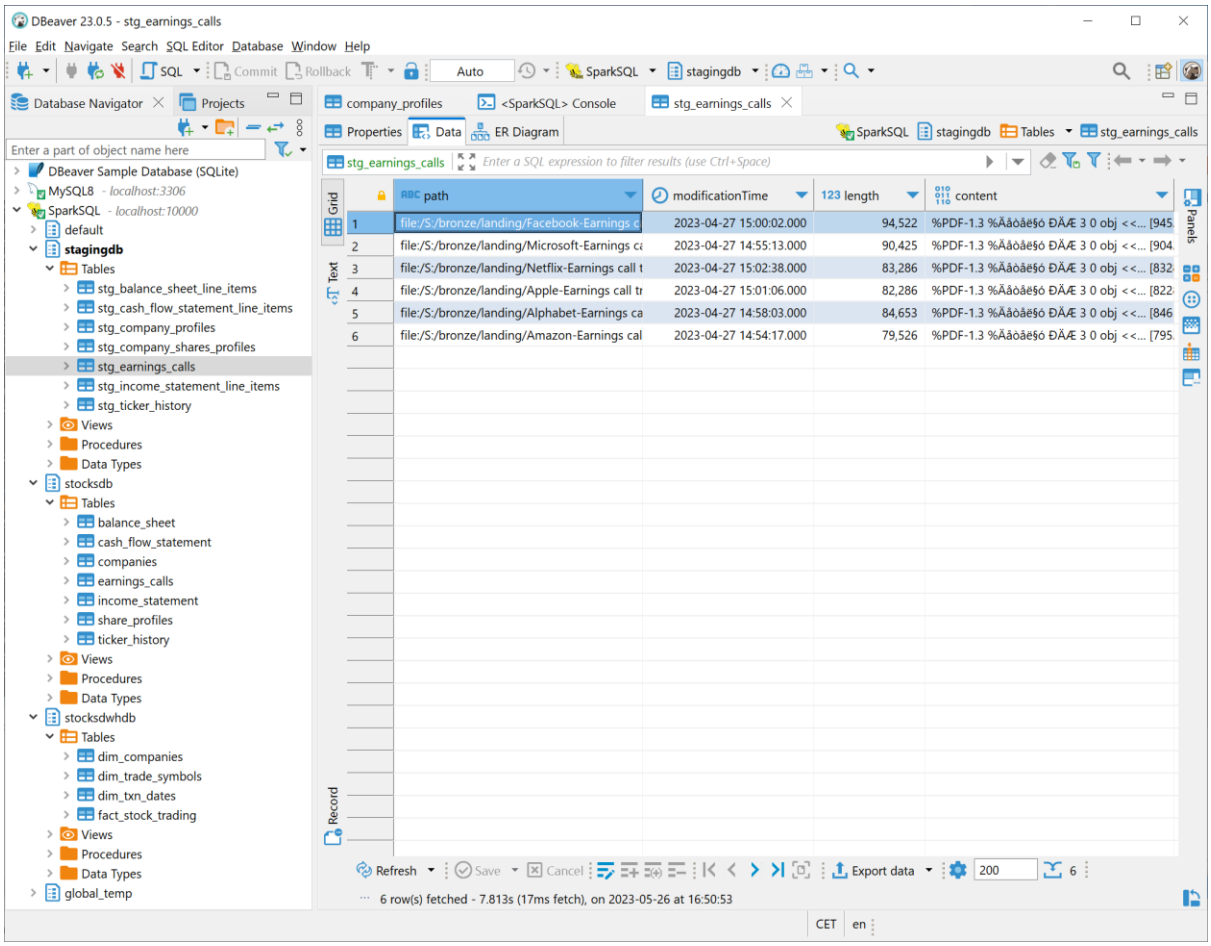
Prvo što se na ER dijagramu slike 31 može primijetiti jest da između tablica nema nikakvih relacija, stoga što tablice služe isključivo za konsolidiranu pohranu podataka iz svih raspoloživih izvora tj. njihovo povezivanje se vrši na višim slojevima sukladno konkretnim potrebama. U slučaju strukturiranih i polu strukturiranih podataka, tablice po strukturi u potpunosti odgovaraju strukturi podataka u izvorima, uz dodatak dvaju kolona. Prva kolona je dataSource i sadrži naziv izvora podataka, a druga kolona loadTime sadrži vrijeme preuzimanja. Iznimka je tablica stg_earnings_calls u koju se iz landing mape skupno preuzimaju nestrukturirani podaci pohranjeni u određenom broju PDF datoteka. Funkcija u Pyspark programskom sučelju, koja obavlja spomenuto skupno preuzimanje pretpostavlja određenu strukturu tablice, kao što je prikazano u JupyterLab bilježnici u kojoj je implementiran podatkovni cjevovod dan u tablici 1:

Podatkovni cjevovod: pipe-line-stg-earnings-calls.ipynb

```
from pyspark.sql import SparkSession
spark = SparkSession. \
    builder. \
    enableHiveSupport(). \
    appName('pipe-line-stg-earnings-calls'). \
    master('local[*]'). \
    getOrCreate()

df3 = spark.read.format("binaryFile").load(r"S:\bronze\landing")

df3.write.mode("overwrite").format("delta").saveAsTable("stagingdb.stg_earnings_calls")
```

Tablica 1 Prikaz podatkovnog cjevovoda iz JupyterLab-a Izvor:Autor

Sadržaj tablice 1 nakon izvršenja podatkovnog cjevovoda prikazan je pomoću alata DBeaver na slici 32:



Slika 32 Prikaz sadržaja tablice stg_earnings_calls u DBeaver-u Izvor:Ekranski prikaz računala od autora

Na slici se može vidjeti da su preuzete PDF datoteke pohranjene u izvornom binarnom obliku, koji zapravo predstavlja PDF format.

Podatkovni cjevovod za punjenje polu strukturiranih podataka preko REST programskog sučelja internetskog servisa Yahoo finance koji se pohranjuju u tablici dan je stg_ticker_history dan je u tablici 2:

Podatkovni cjevovod: pipe-line-stg-ticker-history.ipynb

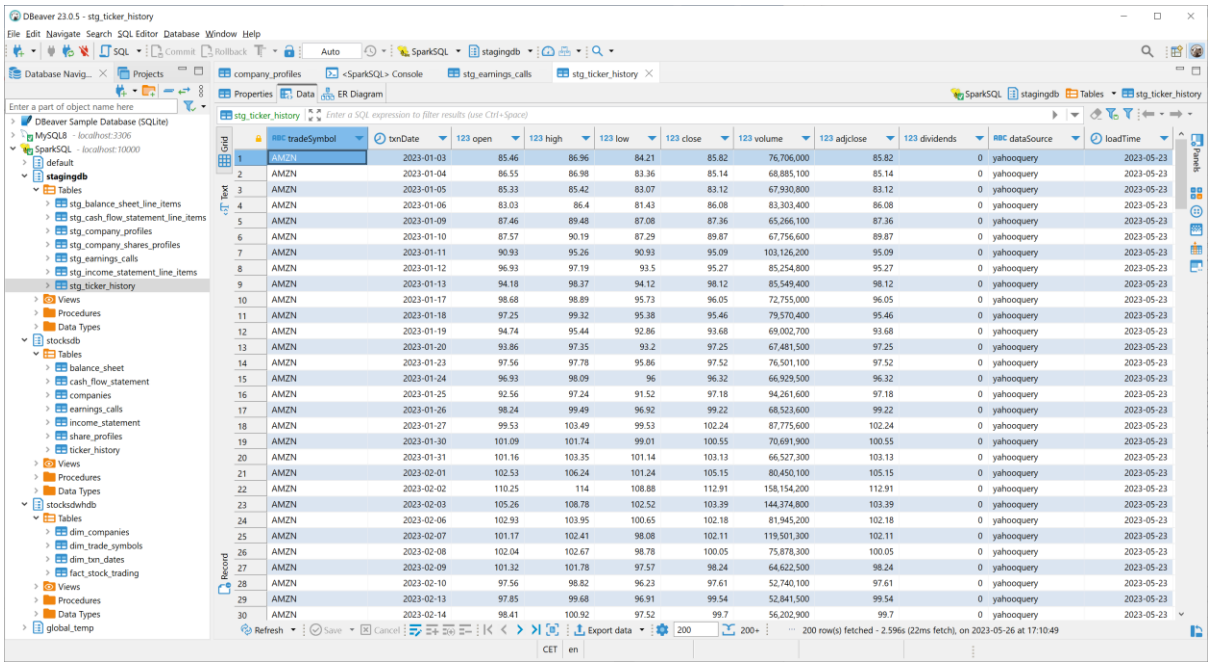
```
from pyspark.sql import SparkSession
spark = SparkSession. \
    builder. \
    enableHiveSupport(). \
    appName('pipe-line-stg-ticker-history'). \
    master('local[*]'). \
    getOrCreate()

import pandas as pd
from yahooquery import Ticker
ts = 'AMZN, AAPL, NFLX, META, GOOGL, MSFT'
t = Ticker(ts)
dfyq = t.history()
dfyq.reset_index(inplace=True)
dfyq.to_csv(r'S:\bronze\landing\tmp_ticker_history.csv', encoding='utf-8', sep=';')

%load_ext jupyterlab_sql_editor.ipython_magic.sparksql

%%sparksql
USE stagingdb;

%%sparksql
DROP VIEW IF EXISTS tv_ticker_history;

%%sparksql
CREATE TEMPORARY VIEW tv_ticker_history
USING CSV
OPTIONS (
  header  "true",
  delimiter ";",
  path  'S:\\bronze\\landing\\tmp_ticker_history.csv',
  inferSchema  "true"
);

%%sparksql
INSERT INTO stg_ticker_history
   (tradeSymbol, txnDate, open,
    high, low, close, volume,
    adjclose, dividends,
    dataSource, loadTime)
SELECT tvth.symbol, tvth.date, tvth.open,
       tvth.high, tvth.low, tvth.close, tvth.volume,
       tvth.adjclose, tvth.dividends,
       'yahooquery' dataSource, current_date() loadTime
FROM tv_ticker_history tvth;
```

Tablica 2 Prikaz podatkovnog cjevovoda iz JupyterLab-a 2 Izvor:Autor

Iz gore navedene bilježnice može se vidjeti da se za komunikaciju sa Yahoo Finance servisom koristi Python modul yahooquery, koji obavlja konverziju polu strukturiranog JSON formata u strukturirani objektni oblik definiran poznatim Python paketom za rad s podacima Pandas.

Također, zbog inkompatibilnosti posljednje inačice (2.x) Pandas paketa sa korištenom inačicom analitičkog programa Apache Spark, primijenjeno je alternativno rješenje, pretvorbe Pandas oblika u .csv datoteku, preko jednostavne funkcije koju taj paket omogućava. Ujedno je demonstrirana velika fleksibilnost Spark SQL-a, stoga što je manipulacija s tom CSV datotekom svedena na kreiranje privremenog prikaza (eng. View). Nakon toga je daljnja manipulacija moguća u potpuno strukturiranom obliku preko SQL upita, kao i sa bilo kojim drugim objektom u bazi podataka.

Sadržaj tablice nakon izvršenja podatkovnog cjevovoda prikazan je pomoću alata DBeaver na slici 33:



Slika 33 Prikaz sadržaja tablice stg_ticker_history u DBeaver-u Izvor:Ekranski prikaz računala od autora

Može se uočiti cjeloviti skup podataka preuzet u polu strukturiranom JSON formatu, preko REST programskog sučelja u konačnici konsolidiran u potpuno strukturirani format uz minimalnu uporabu Python programskog jezika.

Konačno preuzimanje strukturiranih podataka iz baze reportdb na MySQL 8 poslužitelju odvija se preko izravne veze s poslužiteljem kako je to pokazano u podatkovnom cjevovodu danom u tablici 3:

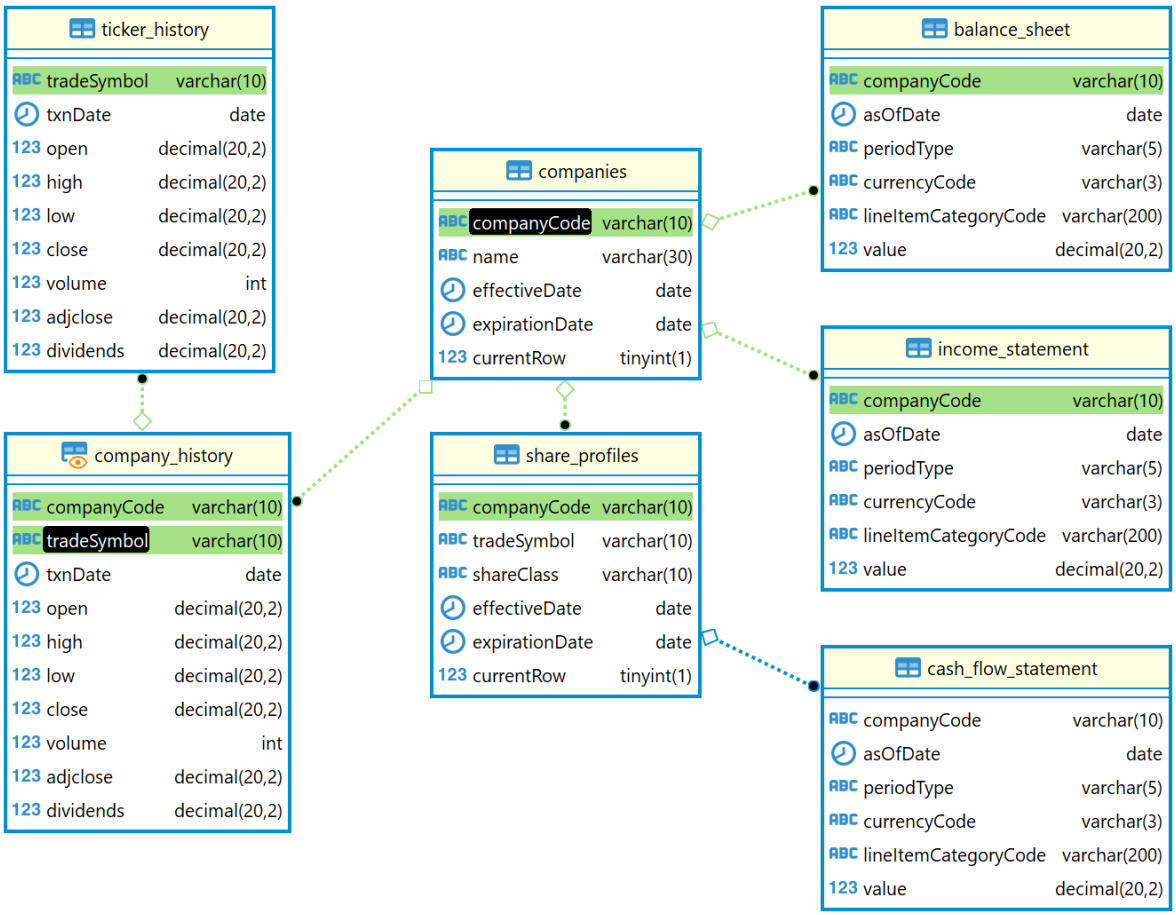
Podatkovni cjevovod: pipe-line-stg-company-profiles.ipynb

```
from pyspark.sql import SparkSession

spark = SparkSession. \
    builder. \
    enableHiveSupport(). \
    appName('pipe-line-stg-company-profiles'). \
    master('local[*]'). \
    getOrCreate()

%load_ext jupyterlab_sql_editor.ipython_magic.sparksql

%%sparksql
use stagingdb;

%%sparksql
CREATE TEMPORARY VIEW tv_company_profiles
USING org.apache.spark.sql.jdbc
OPTIONS (
  url "jdbc:mysql://localhost:3306",
  dbtable "reportdb.company_profiles",
  user 'admin',
  password 'Asdjkl098321'
)

%%sparksql
INSERT INTO stg_company_profiles
            (companyCode, name, address, city,
            zipCode, state, country, effectiveDate,
            expirationDate, currentRow, dataSource, loadTime)
SELECT lcp.companyCode, lcp.name, lcp.address, lcp.city,
       lcp.zipCode, lcp.state, lcp.country, lcp.effectiveDate,
       lcp.expirationDate, lcp.currentRow,
                        'reportdb' dataSource, to_date('2023-01-01') loadTime
FROM tv_company_profiles lcp;
```

Tablica 3 Prikaz podatkovnog cjevovoda iz JupyterLab-a 3 Izvor:Autor

Svakako treba istaknuti da se u navedenoj JupyterLab bilježnici prikazanoj u tablici 3 podatkovnog cjevovoda za punjenje tablice stg_company_profiles može vidjeti da je praktično sva manipulacija sa podacima obavljena uz pomoć SQL naredbi, uz minimalnu uporabu Python koda isključivo za uspostavu veze (sesije) sa Apache Spark sustavom.

Srebrni sloj (eng. silver layer) služi za pohranu pročišćenih i transformiranih podataka koji se nalaze na brončanom sloju i implementiran je u vidu baze stocksdb koja u sebi sadrži tablice pogodne za analize iz područja poslovanja na burzi. ER dijagram stocksdb baze dan je na slici 34:



Slika 34 ER dijagram stocksdb baze podataka Izvor:Ekranski prikaz računala od autora

Prije svega je korisno naglasiti da su relacije na dijagramu slike 34 logičke prirode i nemaju težinu kao u relacijskoj bazi gdje se mogu čvrsto uspostaviti uporabom CONSTRANT mehanizama. Prikladno je istaknuti da su sve tablice pohranjene u Delta Lake formatu, tako da bez obzira na sličnost sa relacijskom bazom način rada je fundamentalno drugačiji. To se poglavito odnosi primjerice na spajanje podataka iz više tablica (eng. join) koji se ovdje izvodi bazirano na podatkovnoj strukturi orientiranoj na stupce (eng. column store). Podaci na ovom sloju u općem slučaju predstavljaju podskup podataka sa brončanog sloja i usmjereni su ka konkretnoj primjeni. To praktično znači da je nazivlje tablica prilagođeno terminologiji koja se koristi u raznim internetskim servisima koji se koriste za dohvat burzovnih podataka. Tako primjerice tablica companies sadrži samo najnužnije kolone iz tablice stg_company_profiles. S obzirom da strukturirani i polu strukturirani podaci dolaze iz prihvatljivo kvalitetnih izvora, cjevovodi za njihovo prebacivanje ne uključuju nikakve provjere kvalitete ili bilo kakve druge transformacije, tako da su implementirani na isti način, kao i cjevovodi za punjenje strukturiranih podataka u stagingdb baze iz reportdb baze, pa ih ne treba posebno analizirati.

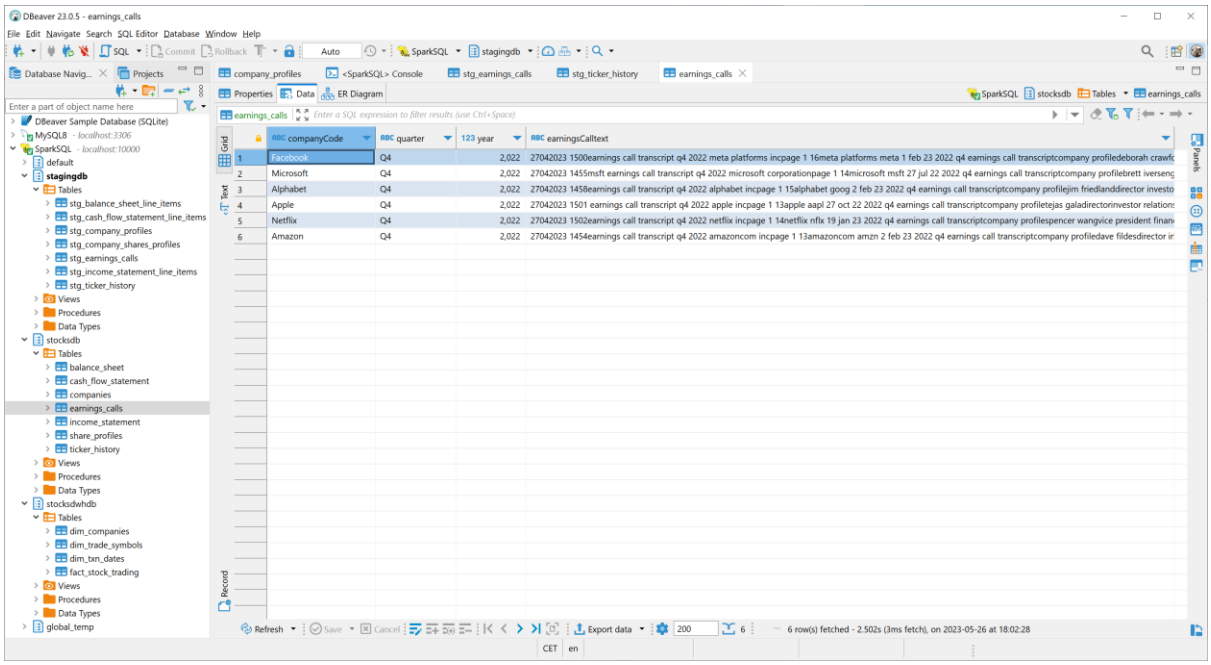
Međutim, situacija je znatno zanimljivija kad se radi o nestrukturiranim podacima koji su u tablici u BLOB koloni contents tablice stg_earnings_calls pohranjeni u izvornom binarnom. Posljedično potrebno je izvršiti ekstrakciju i čišćenje teksta iz binarnog PDF format. Upravo tu zadaću obavlja podatkovni cjevovod dan u tablici 4:

Podatkovni cjevovod: pipe-line-earnings-calls.ipynb

```
from pyspark.sql import SparkSession
spark = SparkSession. \
    builder. \
    enableHiveSupport(). \
    appName('pipe-line-earnings-calls'). \
    master('local[*]'). \
    getOrCreate()

from pyspark.sql import Row
from pyspark.sql.types import StructType, StructField, IntegerType, StringType, DateType
import re
import nltk.corpus
from nltk.corpus import stopwords

earningsCallColumns = StructType([
    StructField('companyCode', StringType(), True),
    StructField('quarter', StringType(), True),
    StructField('year', IntegerType(), True),
    StructField('earningsCalltext', StringType(), True)
])
earningsCallList = []
df5 = spark.sql("select path, modificationTime, length, content from stagingdb.stg_earnings_calls")
for stg_row in df5.rdd.toLocalIterator():
    # extract companyCode from path
    stg_path = stg_row['path']
    stg_contents_file = stg_path[-(len(stg_path) - stg_path.find("landing/") - 8):]
    s_companyCode = stg_contents_file[:stg_contents_file.find("-")]
    # extract quarter from path
    s_quarter = stg_path[stg_path.find("-Q") + 1: stg_path.find("-Q") + 3]
    # extract quarter from path
    s_year = stg_path[stg_path.find(s_quarter) + 3: stg_path.find(s_quarter) + 7]

    # extract text from contents
    pdf_content = fitz.open("pdf", stg_row['content'])
    s_text = ""
    for page in pdf_content:
        s_text = s_text + page.get_text()
    pdf_content.close()

    # clean text
    #    convert all to lower case
    s_text2 = s_text.lower()
    #    clean special characters
    s_text3 = re.sub(r"(@\[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)|^rt|http.+?", "", s_text2)
    #    remove stop words
    s_stop_wors = stopwords.words('english')
    s_text4 = " ".join([word for word in s_text3.split() if word not in (s_stop_wors)])
    earningsCallList.append(Row(s_companyCode, s_quarter, int(s_year), s_text4))

df6 = spark.createDataFrame(data=earningsCallList, schema = earningsCallColumns)
df6.printSchema()
df6.show()

df6.write.mode("overwrite").format("delta").saveAsTable("stocksdb.earnings_calls")
```

Tablica 4 Prikaz podatkovnog cjevovoda iz JupyterLab-a 4 Izvor:Autor

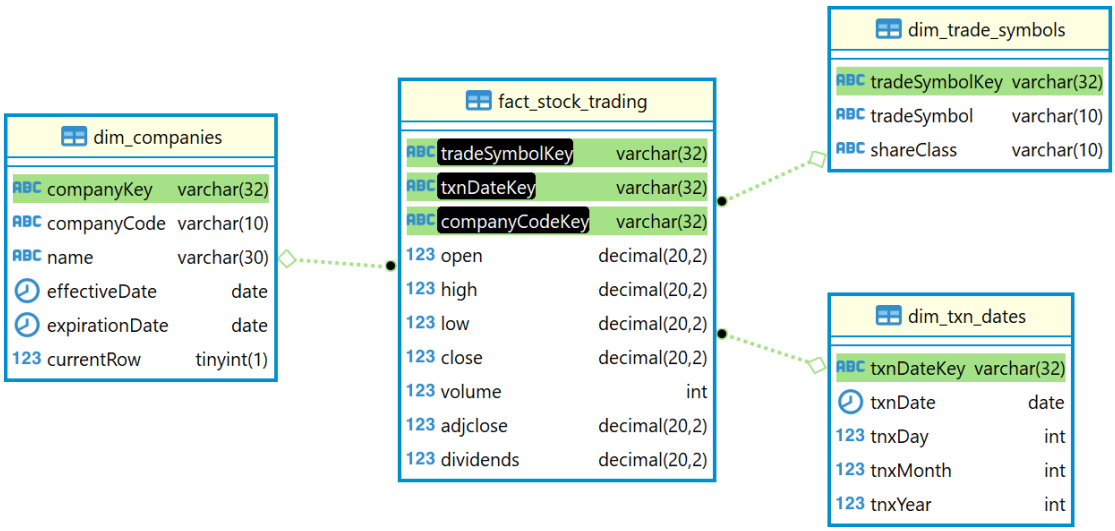
Ovdje se evidentno radi o situaciji gdje se programiranje u Pythonu nije moglo izbjeći, kako bi se napravio čitav niz operacija na binarnom PDF formatu kao što su primjerice ekstrakcija teksta, pročišćavanje teksta, da bi se u konačnici tako dobiveni tekst pohranio u u koloni earningsCalltext u tablici earnings_calls, kako je to prikazano pomoću programa DBeaver na slici 35:



Slika 35 Prikaz sadržaja tablice earnings_calls u DBeaver-u Izvor:Ekranski prikaz računala od autora

Uvidom u kolonu earningsCalltext može se zamijetiti da je binarni sadržaj zamijenjen čitljivim tekstom.

Zlatni sloj (eng. gold layer) služi za pohranu podataka najveće razine kvalitete koji su nerijetko strogo strukturirani sukladno principima Kimball-ovog višedimenzionalnog modela, odnosno potpuno prilagođeni za uporabu na području poslovne inteligencije sa alatima specijaliziranim za rad sa strukturiranim podacima kao što su primjerice Microsoft Power BI, Tableau i sl. Zlatni sloj je implementiran u bazi stocksdwhdb čiji je ER dijagram prikazan na slici 36:



Slika 36 Prikaz ER dijagrama stocksdwhdb baze podataka Izvor:Ekranski prikaz računala od autora

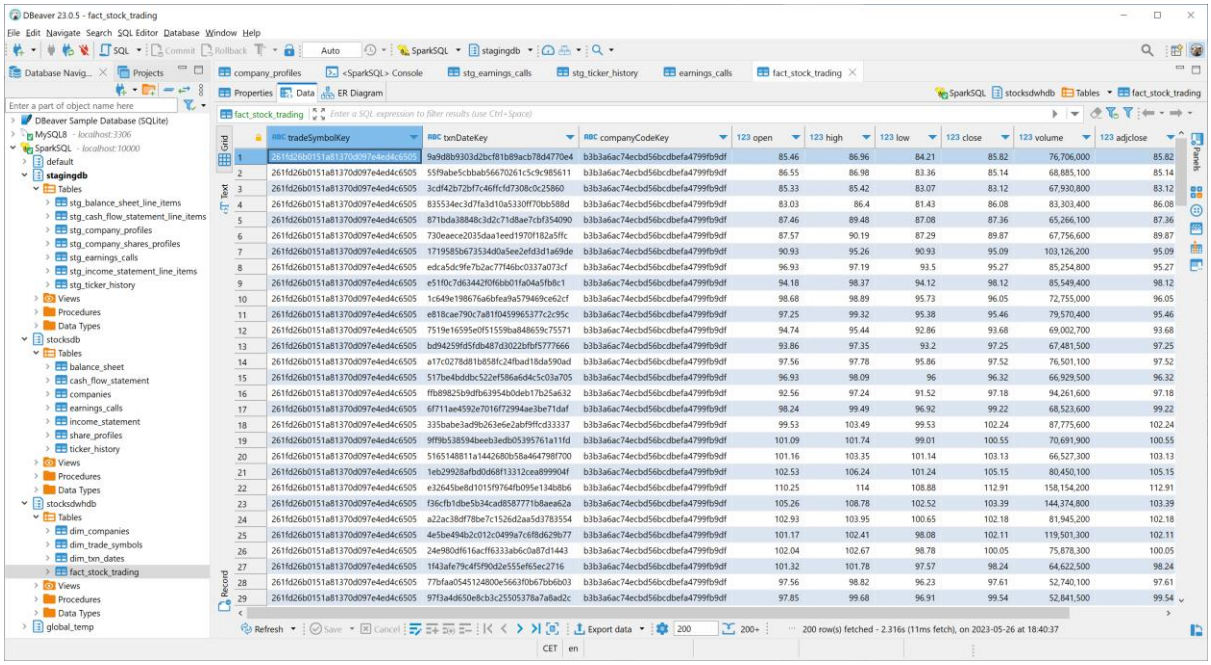
Može se uočiti da je referencijalni integritet između tablice činjenica (eng. fact table) i dimenzijskih tablica na slici 36 uspostavljen preko zamjenskih ključeva (eng. surrogate keys), koji su za razliku od implementacije u relacijskog bazi generirani pomoću MD5 algoritma (prema preporukama stručnjaka iz tvrtke Databricks), a ne iz sekvence stoga što se radi o distribuiranom sustavu gdje bi generiranje jedinstvene sekvence rezultiralo da se njeno generiranje mora centralizirati u jedan računalni čvor, koji bi potencijalno postao usko grlo u radu sustava. Dimenzijska tablica dim_companies predstavlja dimenziju koja se sporo mijenja, ali bilježi promjene (eng. Slowly Changing Dimension Type 2 - SCD2). Dimenzijska tablica dim_trade_symbols predstavlja dimenziju koja se sporo miijenja, ali ne bilježi promjene (eng. Slowly Changing Dimension Type 1 - SCD1). Konačno, dimenzijska tablica dim_txn_dates, sadržava vremensku dimenziju implementiranu na uobičajen način. Za ilustraciju je najzanimljiviji podatkovni cjevovod za punjenje tablice činjenica fact_stock_trading dan u tablici 5:

Podatkovni cjevovod: pipe-line-fact-stock-trading.ipynb

```
from pyspark.sql import SparkSession

spark = SparkSession. \
    builder. \
    enableHiveSupport(). \
    appName('pipe-line-fact-stock-trading'). \
    master('local[*]'). \
    getOrCreate()

%load_ext jupyterlab_sql_editor.ipython_magic.sparksql

%%sparksql
use stocksdwhdb;

%%sparksql
INSERT INTO fact_stock_trading
   (tradeSymbolKey, txnDateKey, companyCodeKey,
    open, high, low, close,
            volume, adjclose, dividends)
SELECT dts.tradeSymbolKey tradeSymbolKey, dtd.txnDateKey txnDateKey,
       dc.companyKey companyCodeKey,
       sch.open, sch.high, sch.low, sch.close, sch.volume,
       sch.adjclose, sch.dividends
FROM stocksdb.company_history sch, dim_trade_symbols dts,
     dim_txn_dates dtd, dim_companies dc
WHERE sch.tradeSymbol=dts.tradeSymbol AND
      sch.txnDate=dtd.txnDate AND
      sch.companyCode=dc.companyCode AND
      sch.txnDate BETWEEN dc.effectiveDate AND dc.expirationDate;
```

Tablica 5 Prikaz podatkovnog cjevovoda iz JupyterLab-a 5 Izvor:Autor

Rezultat rada prikazan je u programu DBeaver na slici 37:



Slika 37 Prikaz sadržaja tablice fact_stock_trading u DBeaver-u Izvor:Ekranski prikaz računala od autora

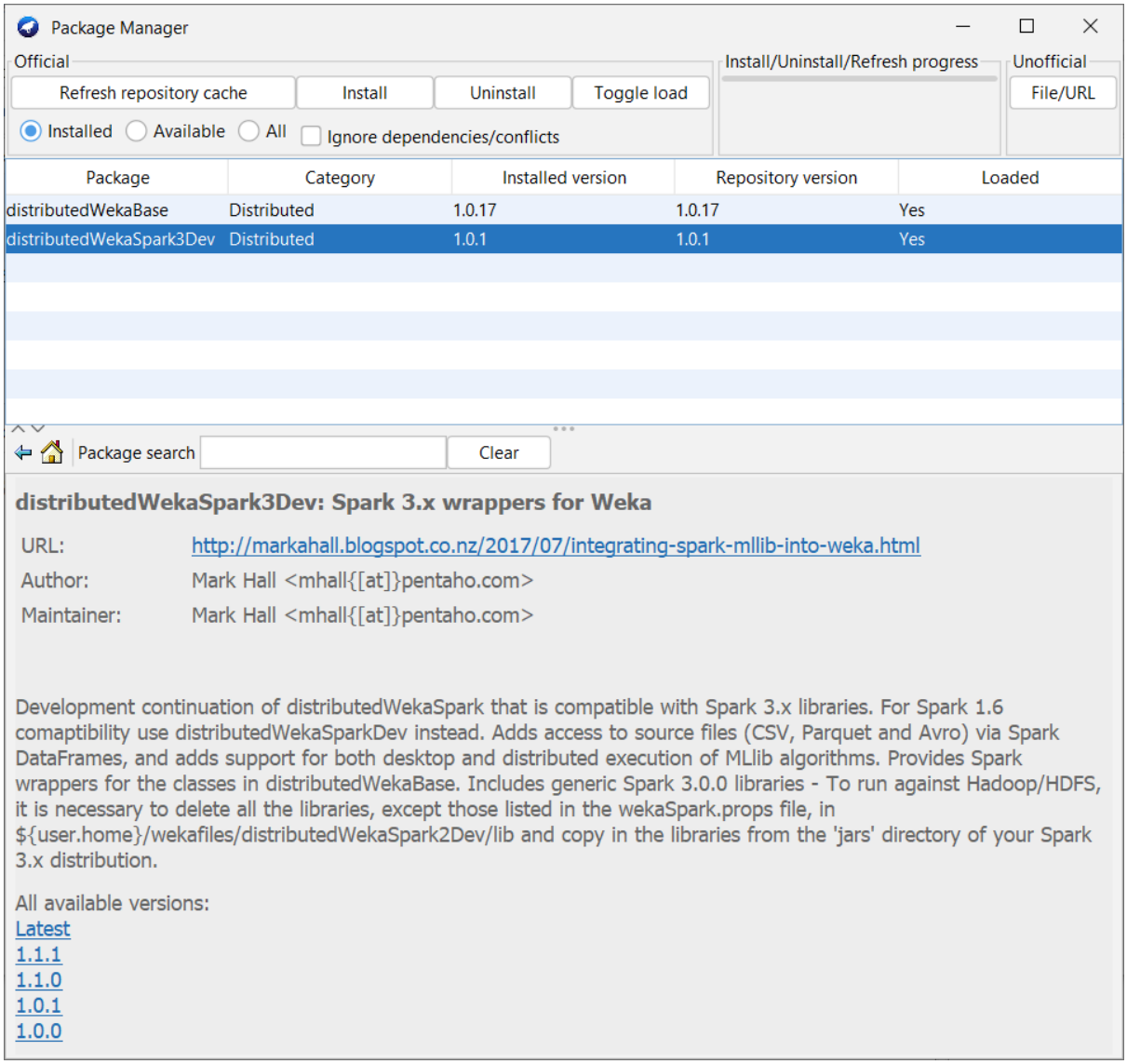
Na slici 37 se mogu primijetiti da su zamjenski ključevi dugi, jer su generirani po MD5 algoritmu.

Podatkovni cjevovodi predstavljeni do sada za ilustraciju prijenosa podataka između pojedinih slojeva implementirani su u obliku JupyterLab bilježnica iz niza razloga. Razvoj samog cjevovoda je znatno olakšan, stoga što se prilikom izrade bilježnice mogu jednostavno provjeravati svi međurezultati. Moguće je u istoj bilježnici u različitim ćelijama miješati programski kod u Pythonu i SQL naredbe, što je vrlo praktično za provjeru među rezultata koje proizlaze iz izvršavanja Python koda. Kao što se moglo vidjeti bilježnica može sadržavati samo SQL naredbe što je čini još jednostavnijom. Međutim, bilježnice se mogu iskoristiti i kao izvršne jedinice za sustave kao što je primjerice Apache Airflow u kome se može definirati međuovisnost izvršavanja bilježnica i sl.

Radi potpunosti potrebno je istaknuti da su implementirani cjevovodi maksimalno pojednostavljeni u funkciji prezentacije koncepta skladišta podataka utemeljenih na jezerima podataka, tako da je izostavljen čitav niz tehničkih detalja potrebnih za produkcijski rad kao što su primjerice inkrementalno učitavanje samo novih, odnosno izmijenjenih podataka, optimizacija za učitavanja velikih količina podataka te razne tehnike provjere kvalitete podataka i sl.

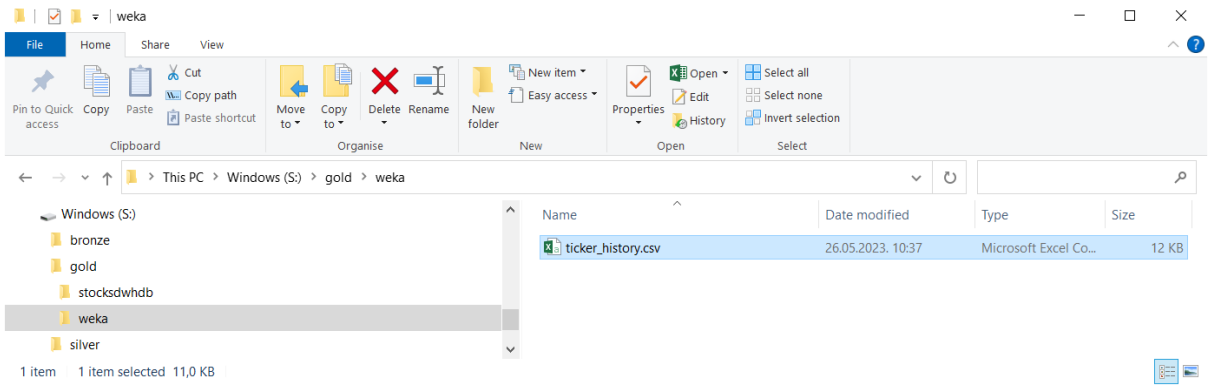
4.3. Primjena alata Weka, RStudio i JupyterLab za analizu podataka na skladištu podataka utemeljenom na jezeru podataka

Za programski alat Weka je dostupna integracija sa Apache Spark pomoću instalacije dodatnih paketa kao što se može vidjeti na slici 38:



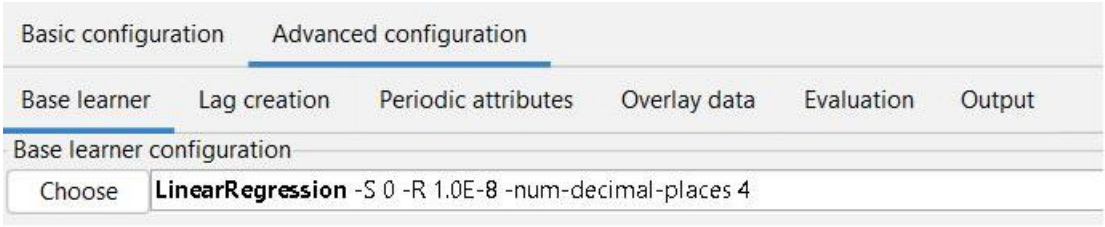
Slika 38 Prikaz package manager-a u Weka-i Izvor:Ekranski prikaz računala od autora

Na slici se može vidjeti da postoji ograničenje uporabe formata podataka koji se mogu koristiti kao izvori. Zbog toga je pripremljena datoteka ticker_history.csv u jezeru podataka na zlatnom sloju kako je to prikazano na slici 39:



Slika 39 Prikaz datoteke ticker_history.csv pohranjene u zlatnom sloju Izvor:Ekranski prikaz računala od autora

Integracija je implementirana na način da se Weka standardne funkcije izvršavaju distribuirano na Apache Spark sustavu, što je stvar konfiguracije. U praktičnom smislu provodi se standardna procedura u ovom slučaju jednostavna linearna regresija kao što se vidi na slici 40:



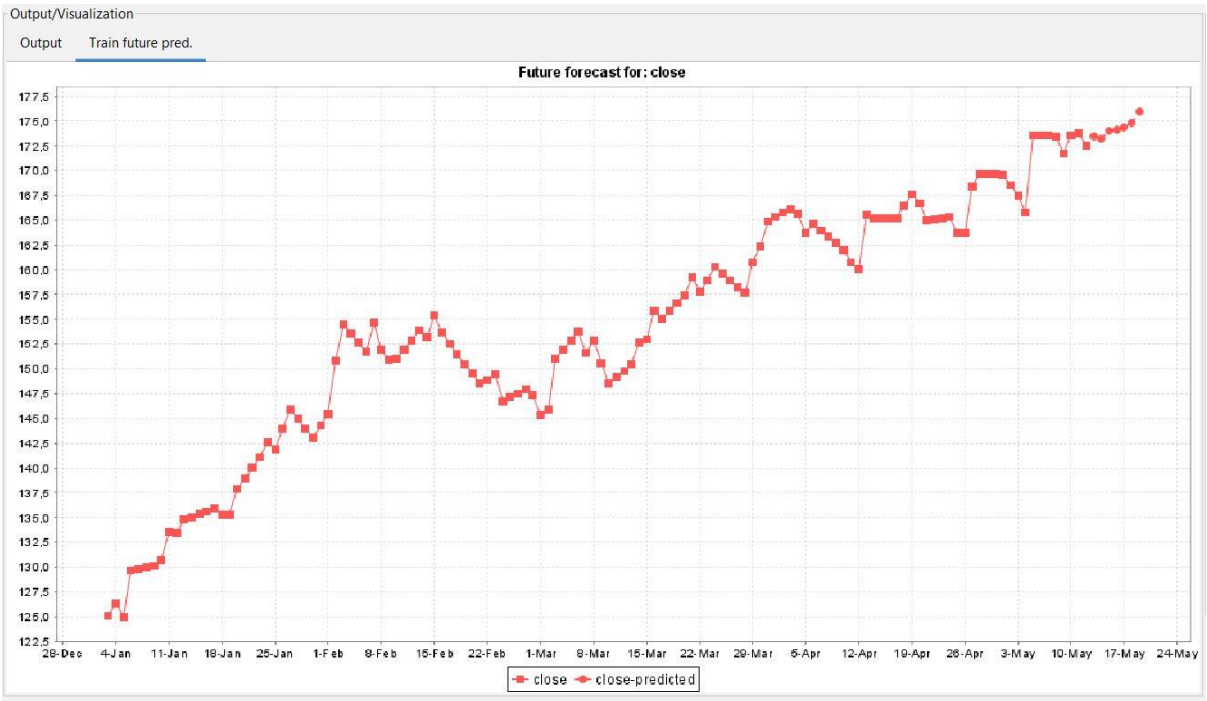
Slika 40 Prikaz osnovnih parametara korištenog modela linearne regresije Izvor:Ekranski prikaz računala od autora

Radi se o predviđanju cijena iz povijesnih podataka cijena dionica pridruženih odgovarajućim dioničkim simbolima, koje se nalaze u spomenutoj .csv datoteci. U konačnici rezultati analize u alfanumeričkom obliku prezentirani su na slici 41:

```
2023-05-06  173.5467
2023-05-07  173.5233
2023-05-08     173.5
2023-05-09    171.77
2023-05-10    173.56
2023-05-11    173.75
2023-05-12    172.57
2023-05-13*  173.4993
2023-05-14*   173.186
2023-05-15*  174.0133
2023-05-16*  174.1079
2023-05-17*  174.3927
2023-05-18*   174.878
2023-05-19*  175.9786
```

Slika 41 Prikaz rezultata predviđanja u tekstualnom obliku Izvor:Ekranski prikaz računala od autora

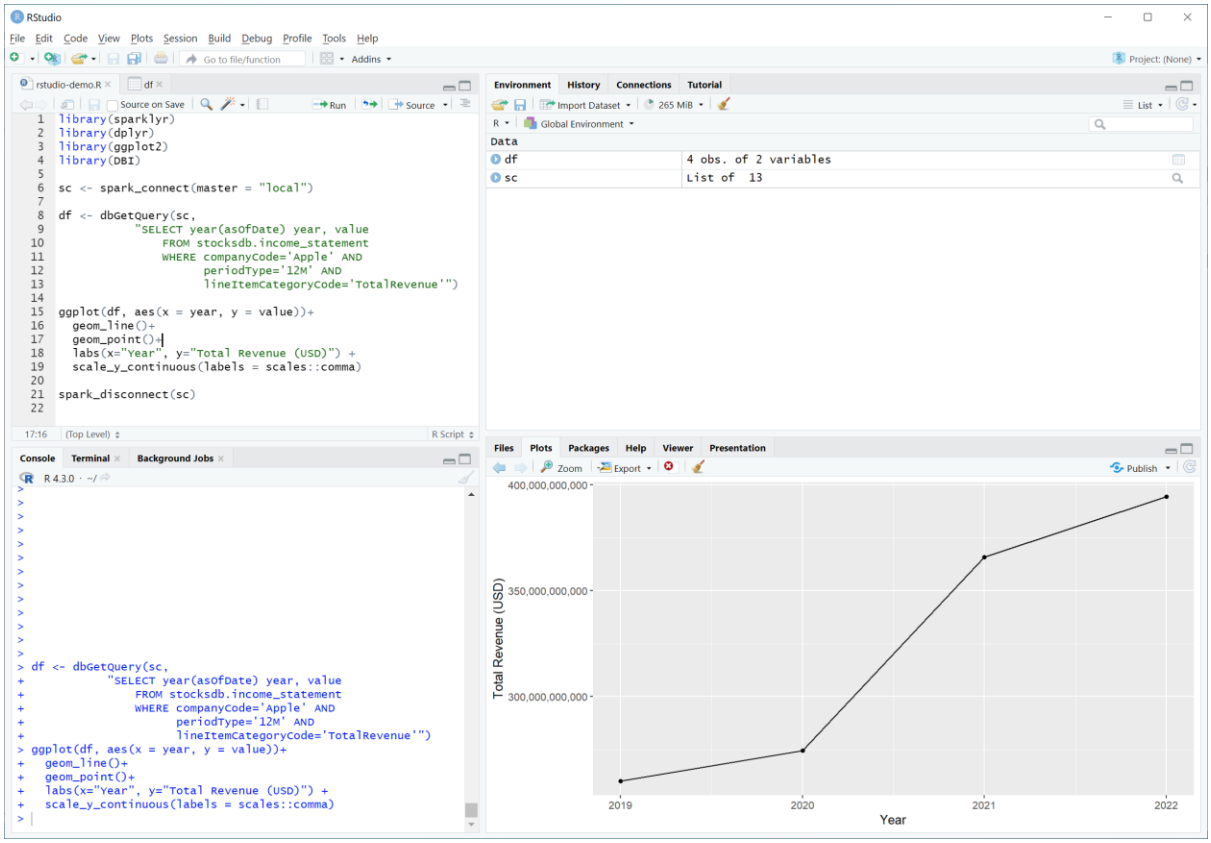
Grafički prikaz rezultata dan je na slici 42:



Slika 42 Prikaz rezultata predviđanja u grafičkom obliku Izvor:Ekranski prikaz računala od autora

Izostanak izravne podrške za Delta Lake format rezultira potrebom za implementaciju dodatnog podatkovnog cjevovoda za generiranje .csv datoteke. Također nema nikakva mogućnost obrade podataka preko SQL-a, što značajno ograničava iskorištavanje svih mogućnosti skladišta podataka utemeljenog na jezeru podataka.

U R Studio-u se koristi za njega raspoloživ paket sparklyr, koji omogućava potpunu integraciju sa Apache Spark analitičkim programom. Primjer integracije dan je na jednostavnom primjeru na slici:



Slika 43 Prikaz integracije Apache Spark-a i R jezika u R Studio-u Izvor:Ekranski prikaz računala od autora

Jednostavnost koda u R programskom jeziku na prvi pogled može zavarati, a zapravo ta značajka na najbolji mogući način ilustrira njenu kvalitetu i cjelovitost integracija. Radi se o tome da je kompleksnost Apache Sparka u potpunosti sakrivena od korisnika koji radi u programskom jeziku R. Na sintaktički vrlo jednostavan način korisnik običnim SQL upitom dohvaća podatke, koji se odmah interno formatiraju kao R DataFrame spreman za daljnju uporabu. U konkretnom primjeru on je izravno proslijeđen u funkciju za grafički prikaz podataka.

JupyterLab je u dosadašnjem izlaganju ekstenzivno korišten za implementaciju podatkovnih cjevovoda. Međutim, on se još ekstenzivnije koristi u analitičke svrhe zbog iznimnih mogućnosti prezentacije podataka, pomoću velikog broja paketa koji se mogu dodatno instalirati u JupyterLab okruženje. Primjer uporabe u analitičke svrhe prezentiran je u JupyterLab bilježnici danoj u tablici 6:

Podatkovni cjevovod: world-cloud-demo.ipynb

```
from pyspark.sql import SparkSession
spark = SparkSession. \
    builder. \
    enableHiveSupport(). \
    appName('word-cloud-demo'). \
    master('local[*]'). \
    getOrCreate()

df3 = spark.sql("select * from stocksdb.earnings_calls
            where companyCode='Apple' and quarter='Q4' and year=2022");
df3.show();

+-----------+-------+----+--------------------+
|companyCode|quarter|year|    earningsCalltext|
+-----------+-------+----+--------------------+
|      Apple|     Q4|2022|27042023 1501 ear...|
+-----------+-------+----+--------------------+


s_text4 = df3.first().earningsCalltext

from wordcloud import WordCloud
import matplotlib.pyplot as plt
%matplotlib inline

wordcloud = WordCloud(background_color="#e2e1eb", max_words=20).generate(s_text4)

plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis("off")

(-0.5, 399.5, 199.5, -0.5)
```

Tablica 6 Prikaz podatkovnog cjevovoda iz JupyterLab-a 6 i wordcloud-a Izvor:Autor

Preko Spark SQL modula izvršava se SQL naredba koja iz tablice earnings_calls dohvaća pročišćeni transkript pozva na zaradu za kompaniju Apple za četvrti kvartal 2022. godine. Taj tekst se onda koristi za obradu po world cloud modelu i prezentira grafički.

5. Zaključak

Temeljem provedenog istraživanja može se zaključiti da uporaba analitičkog programskog alata Apache Spark uz primjenu Delta Lake formata omogućava uspješnu izgradnju skladišta podataka utemeljena na jezeru podataka.

Prezentirane mogućnosti integracije sa etabliranim alatima JupyterLab, DBeaver, Weka i RStudio razvidno potvrđuju široku primjenljivost skladišta podataka utemeljenih na jezerima podataka. Tu posebno treba istaknuti ekstenzivne mogućnosti uporabe SQL-a u svim aspektima rada sa podacima.

Međutim, da bi se maksimalno iskoristile sve te mogućnosti, potrebno je uložiti značajne napore u izučavanju načina funkcioniranja Apache Spark analitičkog programa i Delta Lake formata.

Za te potrebe u ovom radu je primijenjen pristup koji se često viđa u korištenoj literaturi, a to je uporaba analitičkog programa Apache Spark instaliranog na lokalnom računalu. Međutim, to implicira određenu tehničku razinu poznavanja operacijskih sustava (u ovom slučaju Linux i MS Windows), koja je potrebna za instalaciju svih potrebnih komponenata sustava. Za samu instalaciju pojedinih komponenata sustava na Internetu se može naći veliki broj preporuka i primjera, primjenljivih samo ako postoji spomenuta razina tehničkog predznanja. Također, sam analitički program Apache Spark ima vrlo ekstenzivnu dokumentaciju u obliku web stranica, koja nije za početnike. To ne iznenađuje uzme li se u obzir poslovni model softvera otvorenog koda, gdje je sam softver u svakom aspektu dostupan i besplatan, no za bilo kakvu stručnu pomoć potrebno je platiti podršku, koju pružaju razne specijalizirane tvrtke.

Ono što se može istaknuti kao izuzetno dobro je velika zastupljenost i detaljnost opisa (od praktične primjene, pa sve do implementacije na najnižoj sistemskoj razini) korištenog Delta Lake formata u stručnoj literaturi, tako da se ovaj format može preporučiti kao prvi format za početak bavljenja skladištima podataka utemeljenim na jezeru podataka. Za druge spomenute formate Apache Iceberg i Apache Hudi, postoji značajno manja količina informacija bilo koje vrste.

Velika sintaktička sličnost SQL dijalekta na Spark SQL modulu sa SQL dijalektom na nekoj relacijskoj bazi kao što je primjerice u ovom radu korišteni MySQL može zavarati, stoga što je način funkcioniranja sustava u odnosu na relacijsku bazu radikalno drugačiji, orijentiran na konzistentnu primjenu distribuirane/paralelne obrade podataka. Upravo tu najviše dolazi do izražaja raspoloživost sustava na lokalnom računalu i dostupnost svih elemenata sustava kako bi se do kraja mogao izučiti njegov način rada, stoga što se SQL naredbe translatiraju u slijed

funkcija za distribuirani način rada s podacima. Za efikasnu uporabu SQL-a potrebno se upoznati sa temeljnim principima tog procesa.

Naposljetku, uporaba analitičkog programa Apache Spark na lokalnom računalu ne svodi se više samo na proces učenja, nego i na konkretnu primjenu u praksi. Umjesto na nekoliko poslužiteljskih računala (što je do sada najčešće bio slučaj), pokazalo se da sustav može vrlo efikasno iskoristiti logičke procesore raspoložive na jednom računalu (parametar local[*], korištenom u ovom radu, kod uspostave veze sa lokalno instaliranim sustavom znači da sustav iskoristi sve raspoložive logičke procesore za paralelnu obradu podataka). Ta opcija će biti sve zanimljivija dolaskom novih modela računala visokih performansi kao što spomenuta ponuda tvrtke Apple, poznata pod tržišnim nazivom Apple Silicon.

Zbog svega navedenog u konačnici se može zaključiti da su skladišta podataka utemeljena na jezerima podataka rješenje koje će u okolnostima vrlo intenzivnog razvoja, što se najbolje vidi kroz razvoj drugih formata (kao što su Apache Iceberg i Apache Hudi, koji nude dodatne vrlo specifične mogućnosti), u budućnosti sve više koristiti i značajno promijeniti pristup u implementaciji poslovnih rješenja općenito.

6. Literatura

1. Chaudhuri, S., Daya, U. (1997) “An Overview of Data Warehousing and OLAP Technology”, ACM SIGMOD Record, Vol. 26, No. 1, str. 65–74.
2. Kimball, R., Ross,M. (2013) “The Data Warehouse Toolkit, Treće izdanje, Indianapolis, Wiley
3. Rizzi, S., Lechtenbörger, J., Abello, A., Trujillo, J., (2006) “Research in data warehouse modeling and design: Dead or alive?”, ACM 9th International Workshop on Data Warehousing and OLAP (str. 3-10), Arlington
4. Sawadogo, P., Darmont, J. (2021) On data lake architectures and metadata management. J Intell Inf Syst 56, 97–120 https://doi.org/10.1007/s10844-020-00608-7
5. Nargesian,F., Zhu, E., Miller, R. J., Pu, K.P., Arocena, P.C. (2019) “Data lake management: challenges and opportunities”, Proceedings of the VLDB Endowment, Vol. 12 No. 12 str. 1986-1989
6. Llave, M.R. (2018) ” Data lakes in business intelligence: reporting from the trenches”, Procedia Computer Science, Vol. 138, str. 516-524
7. Giebler, C., Gröger, C., Hoos, E., Schwarz, H., & Mitschang, B. (2019). Modeling Data Lakes with Data Vault: Practical Experiences, Assessment, and Lessons Learned
8. Begoli, E., Goethert, I., Knight, K. (2021) "A Lakehouse Architecture for the Management and Analysis of Heterogeneous Data for Biomedical Research and Mega-biobanks," IEEE International Conference on Big Data, Str. 4643-4651
9. T. Hlupić and J. Puniš (2021) "An Overview of Current Trends in Data Ingestion and Integration," 2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO), Str. 1265-1270, Opatija
10. Shiyal, S. (2021) “Beginning Azure Synapse Analytics”, Prvo izdanje, California, Apress Berkeley
11. Poduzeće Databricks Inc., (2021) “The Big Book of Data Engineering” preuzeto s: https://databricks.com/p/ebook/the-big-book-of-data-engineering
12. Inmon, B., Levins, M., Srivastava, R. (2021) “Building the Data Lakehouse”, Prvo izdanje, Basking Ridge, Technics Publications
13. Tovarňák, D., Raček, M., Velan, P. (2021) "Cloud Native Data Platform for Network Telemetry and Analytics," 2021 17th International Conference on Network and Service Management (CNSM),, Str. 394-396
14. Oreščanin, D., Hlupić, T. (2021) "Data Lakehouse - a Novel Step in Analytics Architecture," 2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO), Str. 1242-1246
15. Armbrust, M., Das, T., Sun, L., Yavuz, B., Zhu, S., Murthy, M., Torres, J.,van Hovell, H., Ionescu, A., Łuszczak, A., Świtakowski, M., Szafrański, M., Li, X., Ueshin, T., Mokhtar, M, Boncz, P., Ghodsi, A., Paranjpye, S., Senster, P., Xin, R., Zaharia, M. (2020) “Delta lake: high-performance ACID table storage over cloud object stores”. Proceedings of the VLDB Endowment, Vol. 13, No. 12 Str. 3411–3424.
16. Zburivsky, D., Partner, L. (2021) “Designing Cloud Data Platforms”, Prvo izdanje, Shelter Island, Manning Publications Co.

17. Bureva, V. (2019) “Index matrices as a tool for data lakehouse modelling” , Annual of “Informatics” Section Union of Scientists in Bulgaria,Vol 10, Str. 81-105
18. Zaharia, M., Ghodsi, A., Xin, R., Armbrust, M. (2021) “Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics” In 11th Conference on Innovative Data Systems Research, CIDR 2021
19. Divyansh, J. (2021) “Lakehouse: A Unified Data Architecture”, International Journal for Research in Applied Science and Engineering Technology, Vol. 9., No. 3, Str. 881-887.
20. Behm, A. (2021) “Photon: A High-Performance Query Engine for the Lakehouse”, [e-publikacija], Preuzeto s: http://www.cidrdb.org/cidr2022/papers/a100-behm.pdf
21. Fourny, G., Dao, D., Cikis, C. B., Zhang, C., Alonso, G. (2021) “RumbleML: program the lakehouse with JSONiq.”, [e-publikacija],Preuzeto s: https://arxiv.org/pdf/2112.12638.pdf
22. Behm, A., Palkar, S., Agarwal, U., Armstrong, T., Cashman, D., Dave, A., Greenstein, T., Hovsepian, S., Johnson, R., Krishnnan, A.S., Leventis, P., Luszcak, A., Mennon, P., Mokhtar, M., Pang, G., Paranjpye, S., Rahn, G., Samwel, B., van Bussel, T., van Hovell, H., Xue, M., Xin, R., Zaharia, M. (2021) “Photon: A Fast Query Engine for Lakehouse Systems”, [e-publikacija], Preuzeto s: https://www-cs.stanford.edu/~matei/papers/2022/sigmod_photon.pdf
23. Gopalan, R., (2022) The Cloud Data Lake: A guide to Building Robust Cloud Data Arhitecture“, Prvo izdanje, Sebastopol, O ’Reily Media
24. Mahapatra, A., (2022) „Engineering and Analytics with Delta: Create analytics-ready dana that fuels artificial intelligence and business intelligence“, Prvo izdanje, Birmingham, Pact Publishing
25. Damji, J.S., Wenig, B., Das, T., Lee, D. (2020) „Learning Spark: Lightning-Fast Dana Analytics“, Drugo izdanje, Sebastopol, O ’Reily Media

7. Popis slika

8. Popis tablica

europass

Ivan Batnožić

Date of birth: 27/08/1997 | **Nationality:** Croatian | **Gender:** Male | **Email address:** ivan.batnozic@gmail.com |

Address: Ulica grada Vukovara 269F, Dokument IT d.o.o. , Tower V1, Green Gold Business Centre, 10000, Zagreb, Croatia (Work)

WORK EXPERIENCE

19/01/2021 – 01/03/2022 Croatia

DATA ANALYSTS DOKUMENT IT D.O.O.

Analysing data for business purposes.

EDUCATION AND TRAINING

30/09/2015 – CURRENT Zagreb, Croatia

MANAGERIAL INFORMATICS MASTERS Faculty of Economics and Business Zagreb from University of Zagreb

Address Trg John F. Kennedy 6, 10000, Zagreb, Croatia | **Website** https://www.efzg.unizg.hr/en

LANGUAGE SKILLS

Mother tongue(s): **CROATIAN**

Other language(s):

	UNDERSTANDING		SPEAKING		WRITING
	Listening	Reading	Spoken production	Spoken interaction	
ENGLISH	C1	C1	C1	C1	C1
GERMAN	B2	B1	B1	B1	B2

Levels: A1 and A2: Basic user; B1 and B2: Independent user; C1 and C2: Proficient user

DIGITAL SKILLS

Microsoft Office | Microsoft Powerpoint | R, R Studio, R Markdown | Jupyter lab | Power Query ,Power BI | Google (Google Drive, Google Docs, Google Slides, Google Sheets, Google Meets, Google Trends))