PROCESI DOHVATA, TRANSFORMACIJE I UČITAVANJA U SKLADIŠTIMA PODATAKA

Brletić, Matija

PROCESI DOHVATA, TRANSFORMACIJE I UČITAVANJA U SKLADIŠTIMA PODATAKA

Brletić, Matija

Veleučilište u Karlovcu

Odjel Strojarstva

Specijalistički diplomski stručni studij strojarstva

Matija Brletić

PROCESI DOHVATA, TRANSFORMACIJE I UČITAVANJA U SKLADIŠTIMA PODATAKA

ZAVRŠNI RAD

Karlovac, 2020

Karlovac University of Applied Sciences

Professional graduate study of Mechanical Engineering

Matija Brletić

Processes of extraction, transformation and loading in data warehouses

Final paper

Karlovac, 2020

Veleučilište u Karlovcu

Odjel Strojarstva

Specijalistički diplomski stručni studij strojarstva

Matija Brletić

PROCESI DOHVATA, TRANSFORMACIJE I UČITAVANJA U SKLADIŠTIMA PODATAKA

ZAVRŠNI RAD

Mentor:

dr.sc. Adam Stančić

Karlovac, 2020

Izjavljujem da sam u završnom radu na temu „ Procesi dohvata, transformacije i učitavanja u skladištima podataka “ u potpunosti samostalno izradio teorijski i praktični dio rada, koristeći navedene izvore podataka i informacije te stečeno znanje tijekom studija, uz pomoć mentora dr.sc. Adama Stančića.

Tema rada je nastala pod idejom da se uključe i prethodno završeni studij te iskustva iz srednjoškolskog obrazovanja, koji su doveli do toga da smo se odlučili na izradu aplikacije u programskom jeziku Python. Želim se zahvaliti svojim roditeljima na podršci kroz cijelo moje školovanje te Zrinki, koja me je bodrila kada je bilo najteže.

Završni rad se sastoji od dva dijela, prvi dio rada je teorijski dio, gdje je opisan ETL proces, odnosno proces dohvaćanja, transformacije i učitavanja podataka. Opisane su vrste aplikacija koje omogućavaju ETL proces, komercijalne i aplikacije otvorenog koda, kao i prednosti i nedostaci spomenutih aplikacija s njihovim primjerima. U zasebnom poglavlju opisane su baze podataka, i pojmovi vezani za baze podataka i sustav upravljanja bazama podataka. Da bi ETL proces i infrastruktura bili učinkoviti, mora se voditi računa i o nadzoru ETL sustava, te o optimizaciji ETL procesa. Neispravni podaci se pojavljuju u skladištu podataka u različitim razmjerima pristizanja. Bez obzira na učestalost pojave, arhitekt informacijskog sustava mora znati kako detektirati i rukovati neispravnim podacima. Opisan je i ETL u realnom vremenu koji se odnosi na procese dohvaćanja, transformacije i učitavanja podataka iz različitih izvora u centralizirano skladište za izvještavanje i analizu.

U drugom, praktičnom dijelu rada, nakon što su opisane mogućnosti programskog jezika Python, objašnjen je ETL proces na konkretnom primjeru, izrađena je ETL aplikacija u programskom jeziku Python. Transformacijom prikaza podataka iz formata pogodnog za ljude (odnosno čitanje) dobiva se format datoteke koja je pogodna za pohranu u relacijsku bazu podataka.

Ključne riječi: aplikacije, baze podataka, ETL, programiranje, Python

SUMMARY AND KEY WORDS

The final paper is made of two parts, first part of the paper is a theoretical part, which describes the ETL process, that is the process of extract, transformation and loading data. The types of applications that enable the ETL process, commercial and open-source applications are described, as well as the advantages and disadvantages of these applications with their examples. A separate chapter describes databases, and concepts related to databases and the database management system. In order for the ETL process and infrastructure to be effective, care must be taken both to monitor the ETL system and to optimize the ETL process. Invalid data appears in the data warehouse at different scales of arrivals. Regardless of the frequency of occurrence, the information system architect must know how to detect and handle faulty data. Real-time ETL is also described, which refers to the processes of extracting, transforming and loading data from various sources into a centralized reporting and analysis repository.

In the second, practical part of the paper, after the Python programming language is described, the ETL process is explained on a specific example, and the ETL application in the Python programming language is created. By transforming the display data from a human-redable format produces a file format that is suitable for storage in a relational database.

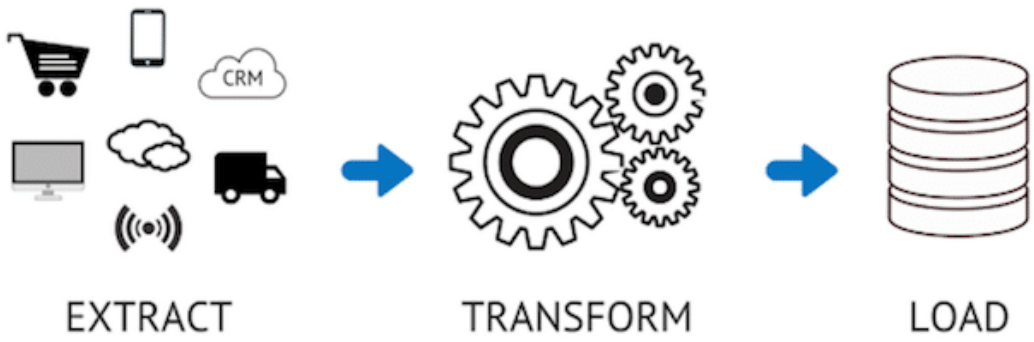
Key words: applications, data bases, ETL, programming, Python

1. UVOD

ETL je skraćenica od pojmova napisanih na engleskom jeziku Extract-Transform-Load, koja bi u prijevodu značila dohvaćanje, transformacija i učitavanje. ETL je sustav temeljnog skladištenja podataka (engl. Data Warehouse – DW ili DWH). Pokretanjem ETL alata vrši se pohrana podataka u skladište podataka. ETL alat ima tri osnovna zadatka [1]:

1. podaci se učitavaju iz različitih izvora podataka,
2. podaci se raspoređuju na područje prikazivanja podataka (engl. Data Staging Area - DSA) gdje se podaci transformiraju i pročišćavaju,
3. podaci se pohranjuju u skladište podataka.

Slika ispod prikazuje ETL proces učitavanja podataka (engl. Extract) iz različitih izvora, transformacije podataka (engl. Transform) i pohrane (engl. Load) u skladište podataka.



Slika 1: ETL proces [2]

ETL alati su kategorija specijaliziranih podataka koji se bave homogenošću skladišta podataka, čišćenjem podataka i problemima s transformacijom i učitavanjem. Homogenost podataka u skladištu podataka podrazumijeva njihovu jednolikost i sličnost, dok se kod čišćenja podaci ispravljaju linijski programiranjem ili sređivanjem nepravilno napisanih redova, stupaca ili polja [3,4,5].

1.1. Odabir ETL alata

Postoje tri opcije kod odabira ETL alata:

1. Kupovina komercijalnog alata,
2. Korištenje open-source[1] alata (hrv. alati otvorenog koda),
3. Pisanje vlastitih aplikacija.

U slučaju pisanja vlastitih aplikacija, aplikacije se mogu napraviti u većini programskih jezika koji su danas dostupni na tržištu i uglavnom su besplatni, a najpopularniji među njima u 2019. godini su [6]:

- Java,
- C,
- Python,
- C++,
- C#,
- Visual Basic .NET.

Ostali komercijalni alati različitih proizvođača su na tržištu dostupni kao komercijalni i open-source alati.

1.1.1. Komercijalni ETL alati

Radi uštede vremena moguće je eliminirati potrebu programiranja vlastitih aplikacija. Na tržištu su dostupni različiti komercijalni ETL alati. Neki od najpopularnijih komercijalnih ETL alata su [7]:

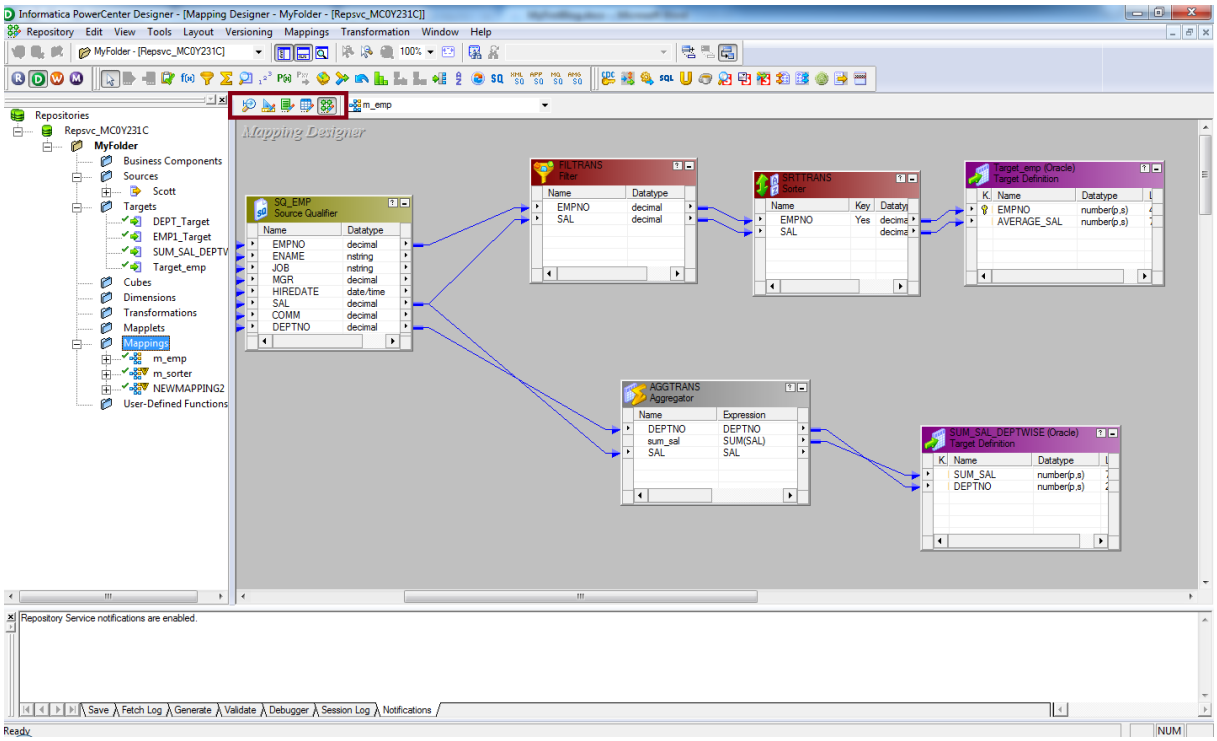
- Improvado,

[1] Odnosi se na softver čiji je izvorni kod dostupan unutar "open source" licence svim korisnicima koji ga mogu mijenjati, prepravljati i poboljašvati njegov sadržaj.

- Skyvia,
- Informatica – PowerCenter,
- IBM – Infosphere Information Server,
- Oracle Data Integrator.

Komercijalni alati olakšavaju rad administratorima baza podataka na način da povezuju različite grane baza podataka i integriraju ili mijenjaju postojeće baze podataka.

Na slici ispod je prikazano korisničku sučelje komercijalnog alata Informatica-PowerCenter u klijentskoj aplikaciji Designer.



Slika 2: Korisničko sučelje komercijalnog ETL alata Informatica-PowerCenter [8]

Glavne značajke koje posjeduju komercijalni ETL alati su [9,10]:

- Podržavaju razne formate datoteka i baze podataka, uključujući Oracle, IBM Db2, SQL Server (engl. Structured Query Language), Teradata, Redshift, Salesforce, MS Dynamics CRM (engl. Customer Relationship Management), EDI (engl. Electronic Data Interchange), COBOL (engl.

Common Business Oriented Language), PDF (engl. Portable Document Format), XML (engl. EXtensible Markup Language), JSON (engl. JavaScript Object Notation), XLS (engl. eXceL Spreadsheet), Google Drive, Dropbox,

- Grafičko, povuci-i-ispusti sučelje olakšava i najzahtjevnije ETL poslove programerima i korisnicima,
- Trenutni pregled i provjera podataka te praćenje i ispravak pogrešaka u stvarnom vremenu,
- Opsežna biblioteka ugrađenih transformacija omogućuje korisnicima da izvršavaju složene zadatke integracije podataka bez pisanja ijednog reda koda,
- Pomoću rasporeda poslova i automatizacije radnog tijeka korisnici mogu definirati tijekove rada ETL procesa i uštedjeti vrijeme i resurse,
- Migracija velike količine podataka u serijama ili trenutno u različite sustave, baze podataka i aplikacije,
- Napredne transformacije podataka s više prolaza,
- Podrška za više ulaza i izlaza podataka,
- Direktan ulaz i izlaz datoteka (strujanje podataka),
- Debugger[2] za interaktivno mapiranje podataka.

1.1.2. Open-source ETL alati

Za open-source ETL alate se najčešće odlučuju nezavisni dobavljači softvera, sistemski integratori i male i srednje velike kompanije, svi iz manje-više istog razloga, a to je da su open-source ETL alati besplatni za korištenje.

Popularniji open-source ETL alati su [11]:

- Apache Airflow,

[2] Debugger je računalni program koji se koristi za uklanjanje grešaka ostalih programa.

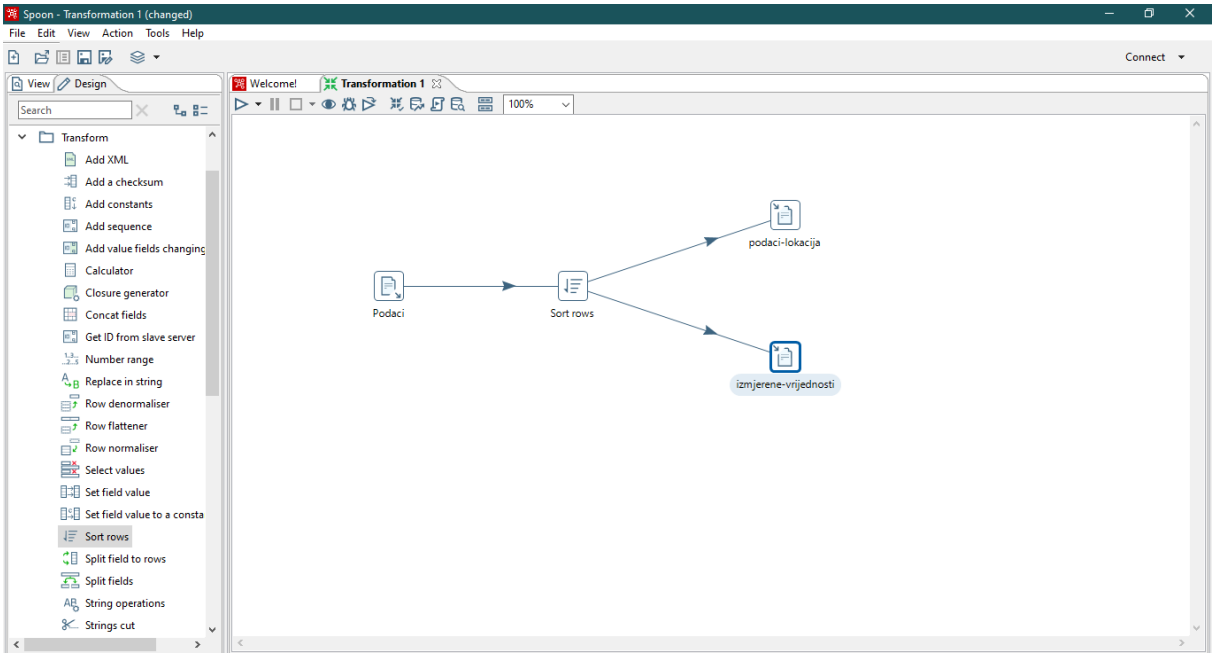
– CloverETL,

– Jaspersoft ETL,

– KETL,

– Pentaho Kettle.

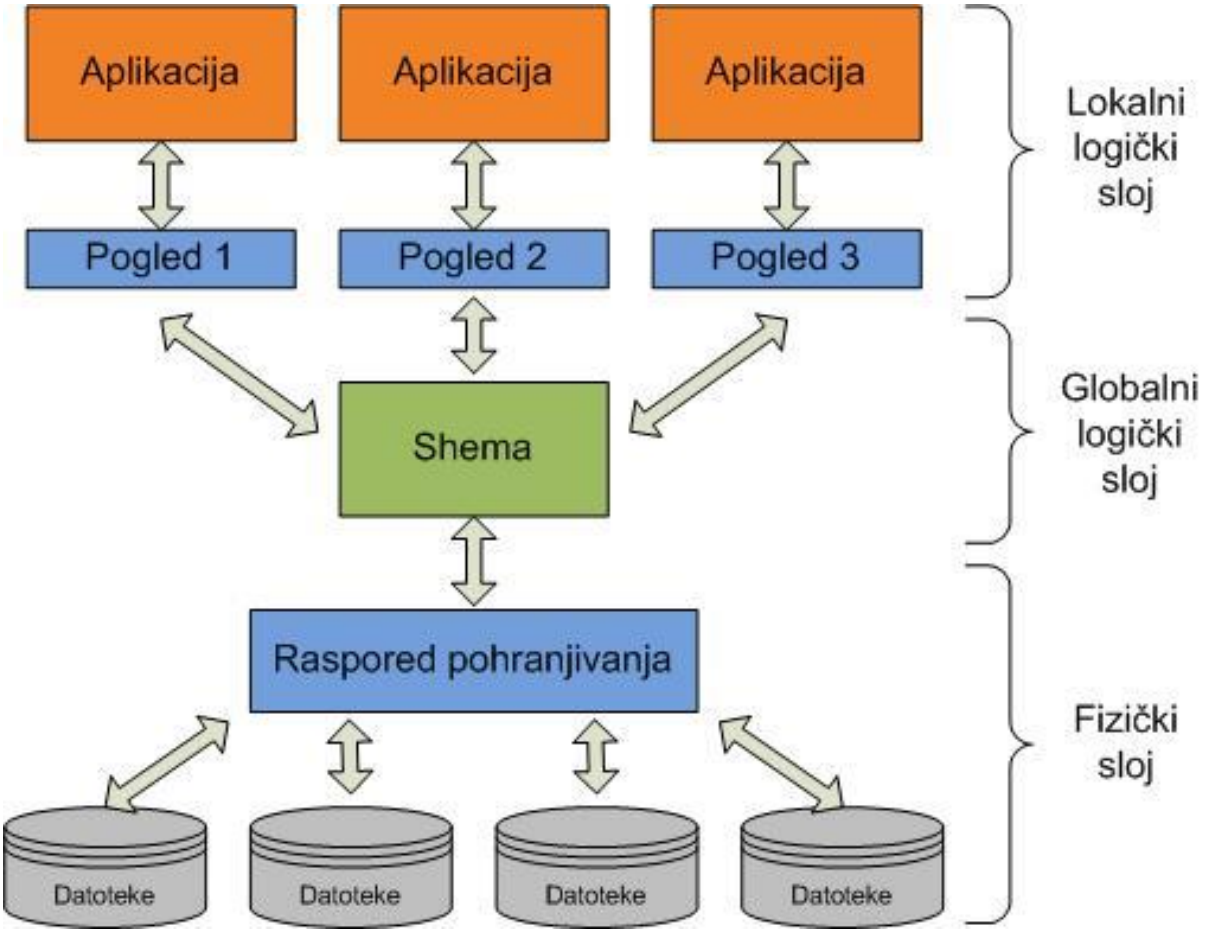
Za korisnike koji koriste open-source ETL alate dovoljno je to da takvi alati imaju mogućnost bržeg i kvalitetnijeg isporučivanja integracijskih rješenja, migraciju transformaciju podataka i ne previše kompleksne zahtjeve zbog manjeg budžeta. Primjer korisničkog sučelja open-source ETL alata je prikazan na slici ispod.



Slika 3: Korisničko sučelje open-source ETL alata Pentaho Kettle [12]

2. Baze podataka

Baza podataka je uređeni skup podataka po određenom kriteriju. Pod podacima se podrazumijevaju zabilježene poznate činjenice koje imaju implicitno[3] značenje, npr. kao što je lokacija mjerenja temperature zraka, temperatura zraka i vrijeme mjerenja. Traženi podaci su možda zabilježeni na web stranici ili su možda pohranjeni na tvrdi disk osobnog računala koristeći softver kao što su MS Excel ili Notepad. Zbirka povezanih podataka s implicitnim značenjem naziva se baza podataka [13,14].



Slika 4: Shematski prikaz višeslojne arhitekture sustava za upravljanje bazama podataka [12]

Sustav za upravljanje bazama podataka (engl. Database Management System - DBMS) je definiran kao višeslojna arhitektura sa strogo definiranim sučeljima između slojeva, čija shema je prikazana na slici iznad, a to su[13,14]:

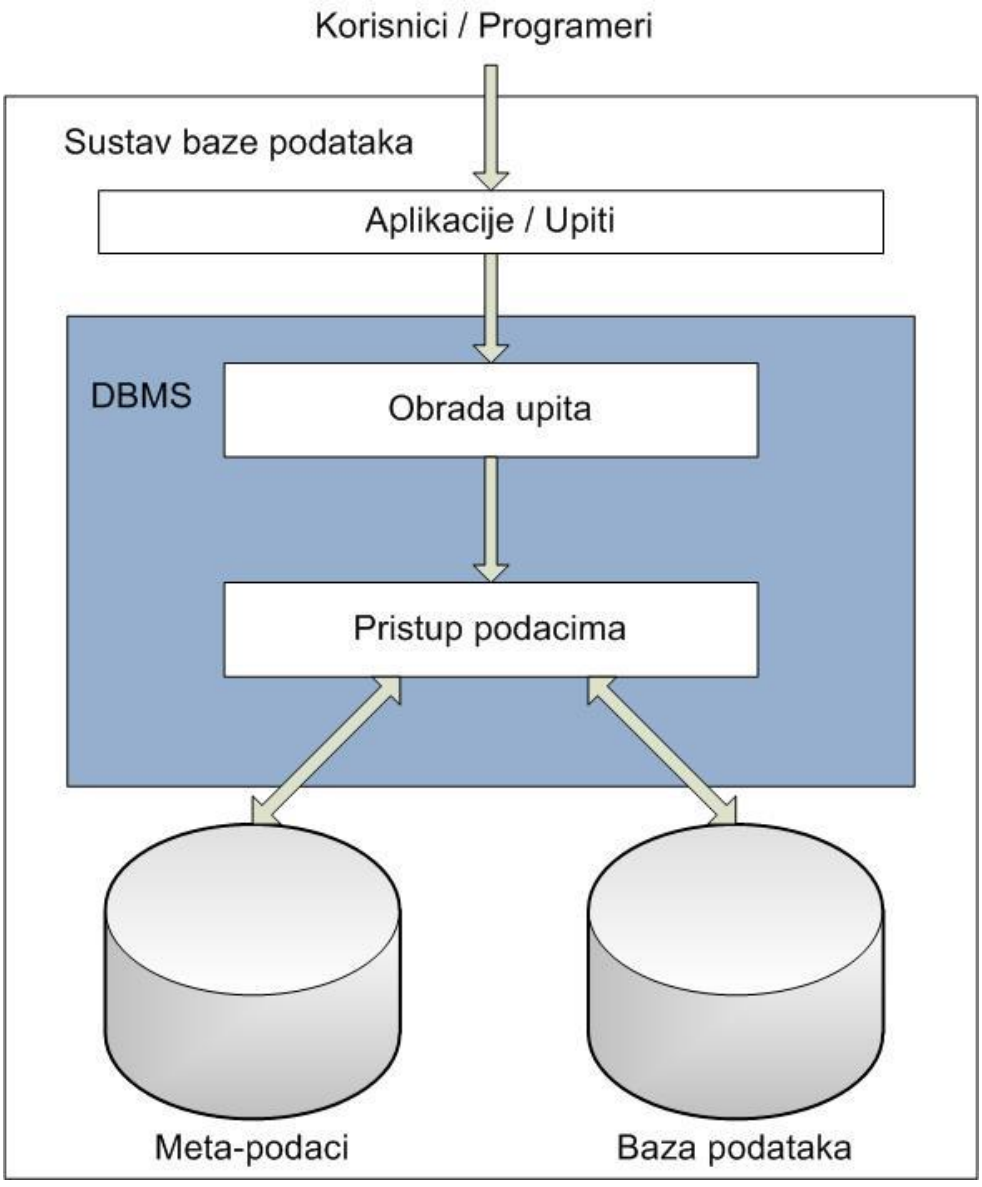
[3] po sebi razumljivo, što se podrazumijeva, bez izravnog navođenja ili spominjanja.

- Fizički sloj – odnosi se na fizički prikaz i raspored podataka na jedinicama vanjske memorije. Fizički sloj odnosi se na raspored podataka na vanjskim uređajima i vidljiv je samo sistem-programerima, tj. onima koji su razvili DBMS. Fizički sloj je sakriven od globalnog logičkog sloja, i obično se može jednostavno mijenjati bez da utječe na aplikacijske programe, za koje se smatra da posjeduju fizičku neovisnost podataka, te se zbog toga podaci ne trebaju prepravljati ako dođe do promjene fizičkog sloja, odnosno izmjena fizičkog sloja ne utječe na globalni logički sloj,
- Globalni logički sloj – odnosi se na logičku strukturu cijele baze podataka. Taj aspekt vidi administrator odnosno projektant baze podataka. Zapis logičke definicije naziva se shema (engl. schema). Logički sloj opisuje cijelu bazu podataka u smislu manjeg broja jednostavnih struktura, a logička shema je ujedno i najvažnija u pogledu njenog utjecaja na aplikacijske programe iz razloga što programeri razvijaju aplikacije koristeći upravo nju. Globalni logički sloj ne utječe na korisnike i aplikacijske programe koji ih koriste, te izmjena logičkog sloja ne mora izazvati izmjenu lokalnog logičkog sloja,
- Lokalni logički sloj – odnosi se na logičku predodžbu o dijelu baze podataka kojeg koristi pojedina aplikacija. Zapis jedne lokalne logičke definicije zove se pogled (engl. view) ili podshema, a vidljiv je korisniku ili aplikacijskom programeru. To je najviša razina apstrakcije koja opisuje samo dio cjelokupne baze podataka. Iako logički sloj koristi jednostavnije strukture, složenost ostaje zbog raznolikosti informacija pohranjenih u bazi podataka. Mnogi korisnici sustava baza podataka ne trebaju sve te informacije, umjesto toga oni trebaju pristup samo određenom dijelu baze podataka. Sustav može pružiti različite poglede na istu bazu podataka.

Izraz „shema baze podataka“ može se odnositi na vizualni prikaz baze podataka, skup pravila koja upravljaju bazom podataka ili na cijeli skup objekata koji pripadaju određenom korisniku. Struktura baze podataka opisana je na formalnom jeziku koji podržava sustav za upravljanje bazama podataka (DBMS). Shema je tekst ili dijagram koji definira logičku strukturu baze. [15].

2.1. Sustavi za upravljanje bazama podataka

Sustav za upravljanje bazama podataka je skup programa koji omogućavaju korisnicima stvaranje i održavanje baza podataka. DBMS je softverski sustav opće namjene što olakšava procese definiranja, konstruiranja, manipuliranja i dijeljenja baze podataka među različitim korisnicima i aplikacijama. Podaci koji opisuju karakteristike nekog izvora u digitalnom obliku, koji su korisni kod pregledavanja, prijenosa i informacijskog sadržaja, nazivaju se metapodaci. Izgradnja baza podataka je proces pohrane podataka na neki medij za pohranu koji je kontroliran od strane sustava za upravljanje bazom podataka.



Slika 5: Sustav baze podataka [12]

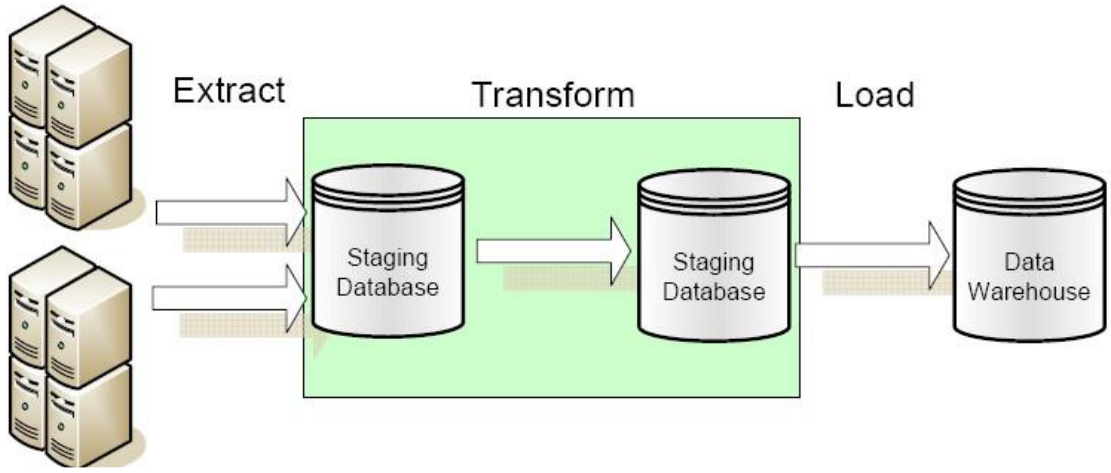
Manipuliranje bazom podataka uključuje funkcije poput upita baze podataka za dohvaćanje određenih podataka, ažuriranje baze podataka u skladu s promjenama u okruženju i stvaranje izvještaja iz podataka. Dijeljenje baze podataka omogućuje istovremeno pristupanje bazi podataka od strane više korisnika istovremeno. Bazu podataka i sustav za upravljanje bazom podataka zajedno možemo nazvati sustavom baze podataka čija slika je prikazana iznad [13].

Ciljevi sustava za upravljanje bazom podataka[16]:

- Primarni cilj sustava za upravljanje bazom podataka je osigurati način za pohranu i dohvaćanje podataka iz baze podataka koji je istovremeno prikladan i učinkovit,
- Upravljanje velikim količinama podataka,
- Omogućavanje prikladnih i učinkovitih načina pohrane i pristupa informacijama,
- Osiguravanje podataka prilikom kvara sustava ili neovlaštenog mijenjanja,
- Davanje dozvola za dijeljene podataka između više korisnika.

3. Pojmovi ETL procesa

Opći okvir za ETL procese prikazan je na slici ispod. Podaci se izdvajaju iz različitih izvora podataka i zatim šire u područje prikazivanja podataka, gdje se podaci transformiraju i pročišćavaju prije učitavanja u skladište podataka (engl. Data Warehouse) [17].



Slika 6: Prikaz ETL procesa [17]

3.1. Dohvaćanje

Prvi dio ETL postupka uključuje dohvaćanje podataka iz izvornog sustava. U mnogim slučajevima to predstavlja najvažniji aspekt ETL-a, jer ispravno dohvaćanje podataka predstavlja osnovu za uspjeh slijedećih postupaka ETL procesa. Većina projekata skladištenja podataka kombinira podatke iz različitih izvora. Svaki zasebni sustav također može koristiti različitu organizaciju podataka ili format podataka. Uobičajeni formati izvora podataka uključuju relacijske baze podataka, EXtensible Markup Language (XML), JavaScript Object Notation (JSON) i flat file (hrv. podaci koji su prikazani u jednoj tablici) najčešće u tekstualnom formatu (engl. txt file), te MS Excel. Također mogu uključivati i ne relacijske strukture baza podataka kao što su sustav upravljanja informacijama (engl. Information Management System - IMS) ili druge strukture podataka kao što su metoda pristupa virtualnoj pohrani (engl. Virtual

Storage Access Method - VSAM) ili indeksirani sekvencijalni pristup metoda (engl. Indexed Sequential Access Method - ISAM) ili čak formati preuzeti iz vanjskih izvora na način kao što je web indeksiranje ili snimak zaslona (engl. screenshot). Učitavanje podataka u stvarnom vremenu (engl. streaming) je još jedan način izvođenja ETL-a kada nije potrebno pohranjivanje podataka između posrednika.

Općenito, faza ekstrakcije ima za cilj konvertirati podatke u jedan format prikladan za obradu i transformaciju jer neki različiti formati pohrane podataka mogu utjecati na razmjenu podataka sa ostalim sustavima.

Svojstveni dio dohvaćanja podataka uključuje i provjeru podataka da bi se potvrdilo da li podaci dohvaćeni iz izvora imaju točne ili očekivane vrijednosti u određenoj domeni (kao što su uzorak ili popis vrijednosti), ako podaci ne prođu provjeru, oni se odbacuju u cijelosti ili djelomično. Odbačeni podaci se vračaju u izvorni sustav (engl. source system) radi daljnje analize radi prepoznavanja i ispravljanja netočnih zapisa [5,18,19].

3.1.1. Metode dohvata podataka

Odabir metode kojom se vrši dohvat podataka najviše ovisi o oblikovanju skladišta podataka za poslovne potrebe i izvornog sustava

Postoje dvije metode dohvata podataka a to su logičke i fizičke.

Logičke metode dohvata podataka dijelimo na [18]:

- Potpuni dohvat,
- Postupni dohvat.

Potpuni dohvat podataka znači da se podaci dohvaćaju iz izvornog sustava u cijelosti. Budući da ovaj način dohvaća sve podatke koji se nalaze u izvornom sustavu, nema potrebe pratiti promjene u izvoru podataka od posljednjeg uspješnog dohvaćanja. Izvorni podaci prikazuju se takvima kakvi jesu i nisu potrebne dodatne logičke informacije na izvornom mjestu.

Kod postupnog dohvata podataka, dohvaćaju se podaci koji su se promijenili nakon što je definiran događaj iz prošlosti. Ovaj događaj može biti datum zadnjeg dohvata podataka ili neki složeniji poslovni događaj poput zadnjeg dana fiskalnog perioda. Da se može utvrditi koje su se promjene dogodile, potrebno je znati vremensko obilježje koji se nalazi u izvornim tablicama, ili ako u izvornim tablicama postoji dodatna tablica koja služi za praćenje promjena. Ako u izvornom sustavu ne postoje takve informacije, onda mu je potrebno dodati jednu od te dvije logike dohvata.

Mnoga skladišta podataka ne koriste ni jednu od navedenih tehnika detekcije promjena kao dio postupka dohvaćanja podataka. Umjesto toga, cijele tablice izvornih sustava učitavaju se u skladište podataka ili u područje poklapanja, gdje se te tablice uspoređuju s prethodnim izvodom iz izvornog sustava radi identificiranja promjena podataka. Ovakav pristup nema značajan utjecaj na izvorne sustave, ali može predstavljati veliko opterećenje na procese skladišta podataka, posebno u slučaju velike količine podataka.

Ovisno o izabranoj metodi logičkog dohvata podataka i mogućnostima i ograničenjima u izvoru podataka, podaci se mogu fizički dohvatiti pomoću dva mehanizma. Podaci se mogu dohvatiti pomoću izvornog sustava isto na dva načina [18]:

1. Mrežno (engl. online),
2. Izvanmrežno (engl. offline).

Pri mrežnom dohvatu podataka, podaci se iz izvornog sustava izvlače izravno. Postupak dohvaćanja se može spojiti direktno na izvorni sustav radi pristupa izvornim tablicama ili neki međusustav koji pohranjuje podatke na unaprijed konfiguriran način. Taj međusustav se nužno ne mora fizički razlikovati od izvornog sustava.

Kod izvanmrežnog dohvata, podaci se ne izvlače izravno iz izvornog sustava već su pohranjeni u nekom unaprijed definiranom formatu u običnu tekstualnu datoteku [5,18,19].

3.2. Transformacija podataka

U fazi transformacije podataka primjenjuje se niz pravila ili funkcija na izvučene podatke kako bi se ti podaci pripremili za učitavanje u odredište. Važna funkcija transformacije je „čišćenje“ podataka koje ima za zadatak da u odredište prenese samo ispravne podatke. Izazov je kada postoji više različitih sustava u međusobnoj komunikaciji, te ako postoji skup znakova koji su dostupni u jednim sustavima, možda nisu dostupni u drugim sustavima. U drugim slučajevima mogu se javiti i problemi kao što su [5,19,20]:

- Odabir samo određenih stupaca za učitavanje ili odabir nul-stupaca koji se ne učitavaju, npr. ako izvorni podaci imaju tri stupca, redni broj, dob i plaću, tada alat za odabir može izabrati samo redni broj i plaću, ili može zanemariti sve one zapise u kojima je vrijednost plaće navedena s nula,
- Prevođenje kodiranih vrijednosti, npr. ako izvorni sustav kodira muško kao „1“ a žensko kao „2“, ali skladište podataka kodira muške kao „M“ a ženske kao „F“,
- Kodiranje vrijednosti slobodnog oblika, npr. mapiranje muškog u „M“,
- Izvođenje nove izračunate vrijednosti, npr. prodaja_količina = količina * cijena,
- Razvrstavanje ili poredak podataka na temelju popisa stupaca radi poboljšanja performansi pretraživanja podataka,
- Spajanje podataka iz više izvora,
- Zbrajanje vrijednosti stupaca ili redova,
- Prenošenje ili pomicanje stupaca, pretvaranje više stupaca u više redova i obratno,
- Dijeljenje stupaca u više stupaca, npr. pretvaranje popisa razdvojenog zarezom koji je specificiran kao niz u jednom stupcu u pojedinačne vrijednosti u različitim stupcima,
- Razdvajanje stupaca koji se ponavljaju,
- Pretraživanje i provjeravanje relevantnih podataka iz tablica ili referentnih datoteka,

- Primjena bilo kojeg oblika provjere podataka; neuspješna provjera može rezultirati djelomičnim, potpunim odbacivanjem podataka ili čak uopće nema odbijanja, pa se stoga neki ili ni jedan podatak ne predaju slijedećem koraku.

3.3. Učitavanje podataka

Faza učitavanja učitava podatke u odredište, a to može biti bilo koje spremište podataka, uključujući jednostavnu datoteku ili skladište podataka. Neka skladišta podataka mogu prebrisati postojeće informacije kumulativnim informacijama, npr. kada se ažuriranje izvučenih podataka vrši svakodnevno, tjedno ili mjesečno. Ostala skladišta podataka mogu dodavati nove podatke u povijesnom obliku u pravilnim intervalima, npr. po satu. Shema baze podataka opisuje predmete koji su prikazani u bazi podataka, te odnose među njima. Shema baze podataka je kostur cijele baze podataka koja definira tablice, prikaze i ograničenja, a može se odnositi i na vizualni prikaz cijele baze podataka. Dok se izvršava faza učitavanja podataka koja djeluje u interakciji s bazom podataka, primjenjuju se ograničenja definirana u shemi baze podataka, npr. jedinstvenost, referentni integritet, obavezna polja, koji također doprinose ukupnom učinku kvalitete podataka ETL procesa, na primjer [5,15]:

- Financijska institucija može imati podatke o klijentu u nekoliko odjela, a svaki odjel može imati podatke o tom kupcu navedene na drugačiji način. Odjel za članstvo može navesti kupca po imenu, dok računovodstveni servis može kupca navesti prema broju. ETL može složiti sve te elemente podataka i objediniti ih u jednolikom prikazu, kao što je spremanje u bazu podataka ili skladište podataka,

- Drugi način na koji tvrtke koriste ETL je trajno premještanje podataka u drugu aplikaciju, npr. nova aplikacija može koristiti jednog dobavljača baze podataka ali drugačiju shemu baze podataka. ETL se može iskoristiti za pretvorbu podataka u format pogodan za čitanje nove aplikacije.

Faza učitavanja ETL postupka uvelike ovisi o tome što se namjerava učiniti s podacima jednom kada ih se učita u skladište podataka. Upotrebe mogu biti [21]:

- Postavljanje poslovnog ili analitičkog alata skladišta podataka kao prioritetnog alata,

- Izrada alata za pretragu web mjesta,
- Izrada algoritma za otkrivanje prijevara,
- Primjena sustava upozoravanja u stvarnom vremenu.

Bez obzira na krajnji cilj, potrebno je s razumijevanjem implementirati funkciju učitavanja podataka u skladište, jer ovisno o količini podataka, strukturi, cilju i vrsti učitavanja može se negativno utjecati na međusobno povezani sustav računala prilikom učitavanja podataka. Poanta je da postupak učitavanja mora biti točno definiran za mjesto gdje se ti podaci učitavaju.

Postoje dvije metode prilikom učitavanja podataka u skladište podataka [21]:

1. Potpuno učitavanje: koristi se kada se podaci učitavaju prvi put u skladište,
2. Postupno povećanje učitavanja: podaci se učitavaju u redovitim intervalima. Posljednji datum pohrane podataka je sačuvan te se učitavaju podaci samo koji su nastali nakon tog datuma.

Postupak postupnog povećanja učitavanja podataka je prilično jednostavan kada postoji manje podataka za učitavanje u skladište podataka, ali kada se poveća količina podataka koje se učitaju u skladište dolazi se do tri glavna problema [21]:

1. Redoslijed: dolazak velike količine podataka može dovesti do toga da podaci ne budu procesuirani po redoslijedu dolaska u skladište podataka. Npr. ako su podaci ažurirani ili obrisani, procesuiranje po krivom redoslijedu će dovesti do pojave netočnih podataka,
2. Promjena sheme: shema će se promijeniti ako se dodaje ili obriše neki od atributa ili se pošalju krivi tipovi podataka u skladište podataka, a u tom slučaju dolazi do nekonzistentnosti podataka,
3. Nadzor: kada postoji velika količina podataka koji dolaze iz puno izvora, moguća je i veća količina neispravnih podataka. Ti podaci su neizbježni, ali ih je bitno što prije uočiti.

3.4. Prednosti ETL procesa

Prednosti ETL procesa su [22,23]:

- Jednostavnost korištenja - Najveća prednost komercijalnih ETL alata je jednostavnost upotrebe. Alat sam određuje izvore podataka i pravila za dohvat i obradu podataka, a zatim implementira postupak i učitava te podatke. Ovo eliminira potrebu za kodiranjem u smislu potrebe pisanja procedura i programskog koda,
- Vizualni tok - Jedna od većih prednosti ETL alata je ta da omogućava pregled vizualnog sistemskog logičkog toka. Svaki ETL alat prikazuje ove tokove različito, ali čak i najmanje privlačni ETL alati povoljno se uspoređuju s prilagođenim sustavima koji se sastoje od SQL-a, pohranjenih procedura, sistemskih skripti, i drugih tehnologija,
- Strukturirani dizajn sustava - ETL alati dizajnirani su za specifične probleme integracije podataka: popunjavanje skladišta podataka ili integriranje podataka iz više izvora ili čak samo prebacivanje podataka s jednog mjesta podataka na drugo mjesto, npr. s diska na web mjesto. Imajući na umu održivost i proširivost, programerima pružaju strukturu metapodataka. To je posebno velika prednost za timove koji grade svoje prvo skladište podataka,
- Operativna otpornost - Mnoga skladišta podataka mogu imati problema prilikom uhodavanja operativnih procesa. ETL alati pružaju funkcionalnost i standarde za rad i nadzor sustava u proizvodnji. Svakako je moguće dizajnirati i izgraditi dobro ručno kodiranu ETL aplikaciju, unatoč tome timovima za skladište podataka i poslovnu inteligenciju je lakše nadograditi ETL alat u otporniji ETL sustav,
- Napredno profiliranje i čišćenje podataka - Većina skladišta podataka strukturno je složena, s mnogo izvora i odredišta. U isto vrijeme, zahtjevi za transformaciju često su prilično jednostavni, a sastoje se prije svega od pretraživanja i izmjene podataka. Ako postoji složen sustav za transformaciju, treba nabaviti dodatni modul za ETL rješenja takvog tipa. Mnogi ETL alati nude takve značajke i potrebno ih je usporediti s drugim ETL alatima i njihovim značajkama,

- Performanse i složene situacije upravljanja podacima - ETL alati nude bolju uslugu za transfer velikih količina podataka i njihov prijenos u skupinama masovnim paralelnim obradama, simetričnim i višestrukim obradama, manipuliranjem nizovima, promjenama podataka i integraciji više skupova podataka,
- Poboljšana poslovna inteligencija - ETL alati poboljšavaju pristup podacima, jer pojednostavljuju proces dohvaćanja podataka iz raznih izvora, pretvaranja i učitavanja. Poboljšani pristup informacijama izravno utječe na strateške i operativne odluke koje se temelje na činjenicama utemeljenima na podacima obrađenima ETL alatima. ETL alati također omogućuju poslovnim liderima da dobivaju informacije na temelju njihovih specifičnih potreba i u skladu s tim donose odluke,
- Budući da se ETL alati temelje na grafičkom korisničkom sučelju (engl. Graphical User Interface - GUI), pružaju vizualni tok logike sustava,
- ETL alati imaju funkciju upravljanja pogreškama zbog koje imaju radnu otpornost,
- Učinkovitost ETL alata znatno je bolja jer struktura njegove platforme pojednostavljuje izgradnju visokokvalitetnog sustava skladištenja podataka.

3.5. Nedostaci ETL procesa

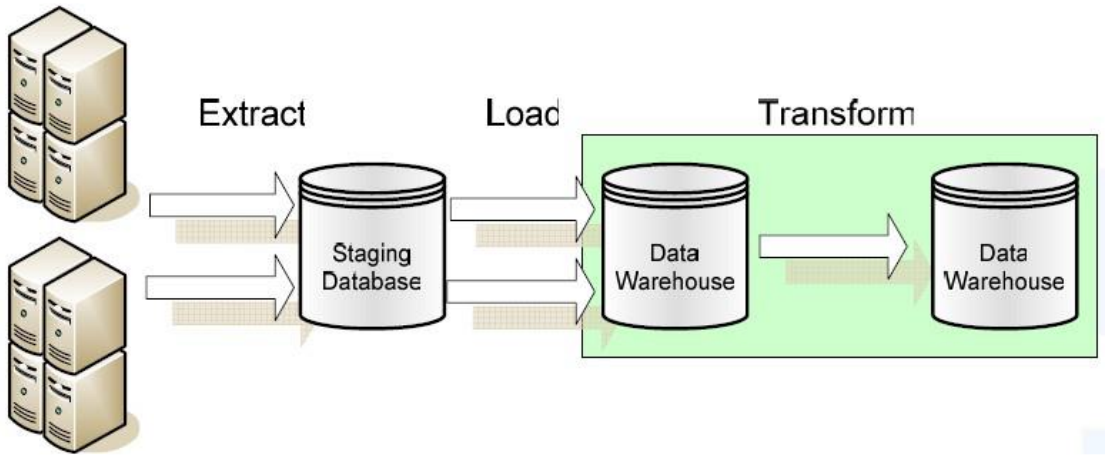
Nedostaci ETL procesa su [24]:

- Fleksibilnost – Zbog ciljanog učitavanja bitnih podataka u skladište dovodi do toga da će se budućim zahtjevima koji će trebati podatke koji možda nisu učitani u izvorni dizajn, trebati dodati u ETL postupak naknadno,
- Hardver – Većina alata treće strane koristi vlastite pogone za implementaciju ETL procesa, što može zahtijevati ulaganje u dodatni hardver,

- Investiranje u vještine – Upotreba alata trećih strana za implementaciju ETL procesa automatski nameće potrebu za učenjem novih aplikativnih ili skriptnih jezika i procesa,
- Krivulja učenja – Primjena alata trećih strana koji koriste strane procese i jezike rezultira blagom krivuljom učenja zbog nedostatka znanja i iskustva,
- ETL pogoni koji vrše transformaciju podataka i provjeru ispravnosti podataka na način red-po-red, može odvesti do uskog grla u cjelokupnom procesu,
- Ovisno o mjestu skladišta podataka, podaci se nekada i dva puta prenose putem mreže. Jednom između izvora i ETL poslužitelja i opet između ETL poslužitelja i ciljanog skladišta podataka,
- Budući da su u skladištu podataka pohranjeni samo podaci koji su izdvojeni od strane programera koji ih je izdvojio putem programskog koda, ostali podaci koji ne postoje u skladištu podataka a trebalo bi ih ubaciti zbog budućeg korištenja mogu dovesti do potrebe za temeljnim redizajnom programskog koda. Kao rezultat toga povećavaju se troškovi i vrijeme redizajna aplikacije,
- Dugi razvojni ciklusi koji mogu trajati i po nekoliko mjeseci.

4. ELT postupak

ELT postupak (engl. Extract, Load, Transform), iako sličan ETL postupku, razlikuje se po nekoliko značajki od ETL postupka, ponajprije po tome što se transformacija podataka izvršava nakon što su podaci učitani u skladište podataka (slika 7).



Slika 7: Prikaz ELT procesa [17]

Prvi dio ELT procesa odnosi se na povlačenje podataka, njihovu integraciju i učitavanje podataka, dok je drugi dio procesa transformacija podataka. ELT ima dio koji je zadužen za transformaciju i nalazi se na odredišnom sustavu, gdje može poslužiti i za neke druge poslove. Iz razloga jer se dio koji je zadužen za transformaciju nalazi na odredišnom sustavu, ETL može biti i puno ekonomičniji od ELT procesa [17,24].

Prednosti ELT postupka [24]:

- Upravljanje projektima – ako postoji mogućnost podijeliti skladišni postupak na specifične i izolirane zadatke, dovodi do toga da se projekt osmisli na manjoj osnovi zadataka, pa se projekt može raščlaniti na lakše obradive dijelove,
- Fleksibilnost – općenito, u ELT implementaciji svi podaci iz izvora učitavaju se u skladište podataka kao dio postupka dohvaćanja i učitavanja. To u kombinaciji s izoliranim procesom transformacije dovodi do toga da se budući zahtjevi mogu lako uklopiti u strukturu skladišta,

- Smanjenje rizika – uklanjanjem bliskih međuovisnosti između svake faze procesa gradnje skladišta podataka, pruža izvrsnu i sigurnu platformu za promijene, održavanje i upravljanje,
- Korištenje postojećeg hardvera – u provedbi ELT -a kao procesa izgradnje skladišta mogu se koristiti ugrađeni alati dobiveni s mehanizmom za izradu baze podataka,
- Korištenje postojećih vještina – korištenjem funkcionalnosti koji pružaju mehanizmi baza podataka, postojeće vještine korištenjem kroz baze podataka ponovno se koriste za razvoj skladišta.

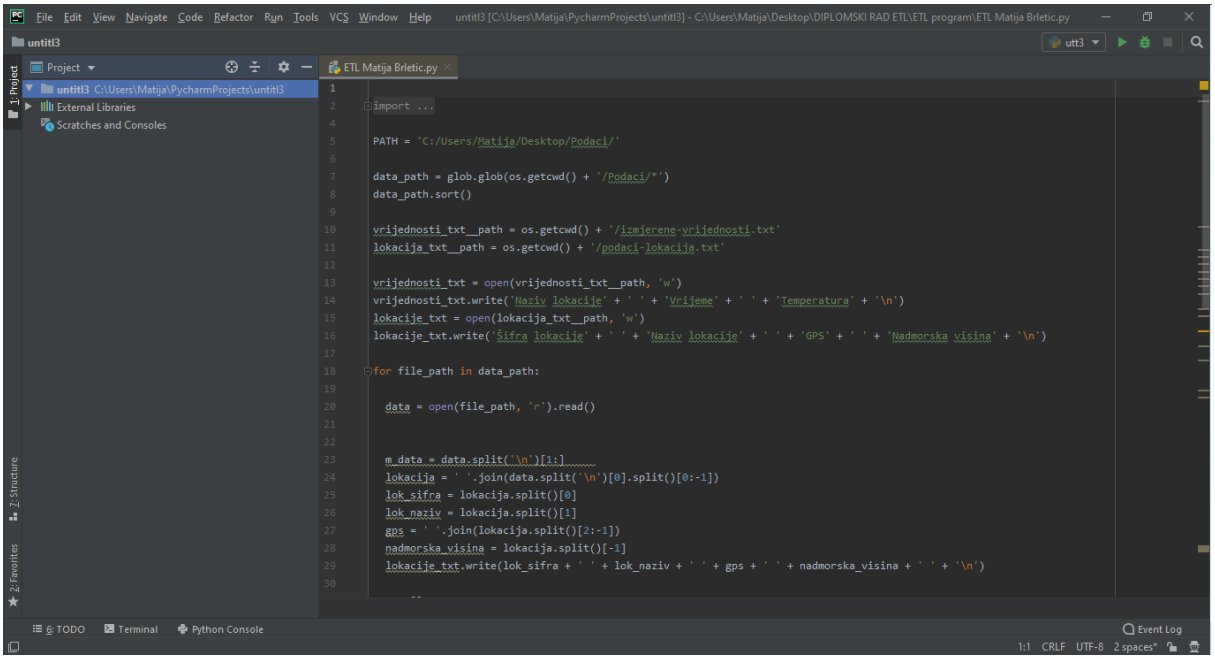
Nedostaci ELT procesa [24]:

- Protiv norme – ELT proces je novi pristup koji se koristi za dizajniranje i razvoj skladišta podataka. Iako se više puta dokazao svojom obilnom upotrebom u implementacijama širom svijeta, zahtjeva promjenu mentaliteta dizajnerskog pristupa u odnosu na tradicionalne metode,
- Dostupnost alata – kao novi pristup dizajniranja i skladišta podataka, ELT pati od ograničene dostupnosti alata,
- ELT pristup djeluje bolje za veću količinu podataka koje treba obraditi. Kako se količina podataka koje treba obraditi povećava, tako se povećava i učinak [17,24].

5. Primjer izrade i korištenja komercijalnog ETL alata

5.1. Programski jezik Python

Python je programski jezik opće namjene, koji dopušta programerima korištenje nekoliko stilova programiranja. Objektno orijentirano, strukturno i aspektno orijentirano programiranje. Python-ova jednostavna sintaksa, koje se lako uči, naglašava čitljivost i time smanjuje troškove održavanja programa. Python podržava module i pakete, što potiče modularnost programa i ponovnu upotrebu koda. Python-ov interpretator i opsežna standardna knjižnica dostupni su u izvornom ili binarnom obliku bez naplate na sve glavne platforme i mogu se slobodno distribuirati [25].

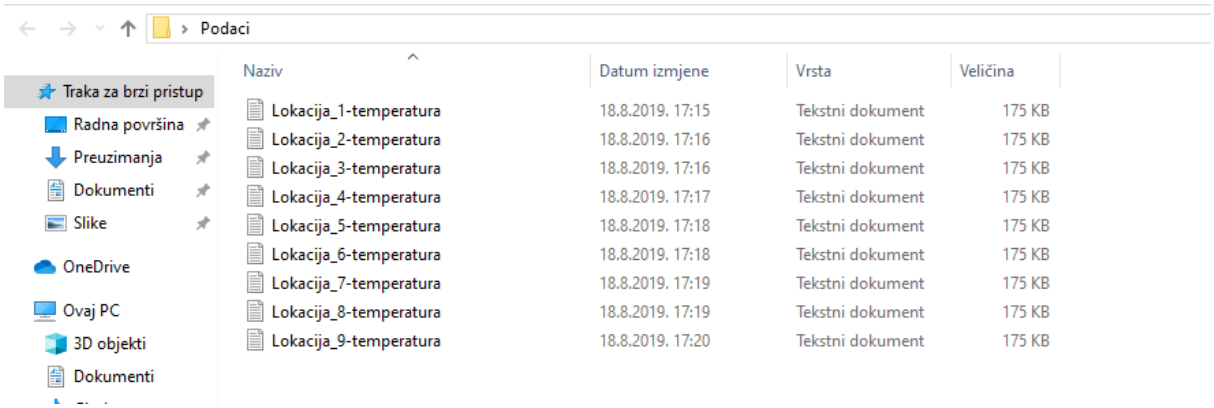


Slika 8: Integrirano razvojno okruženje PyCharm [12]

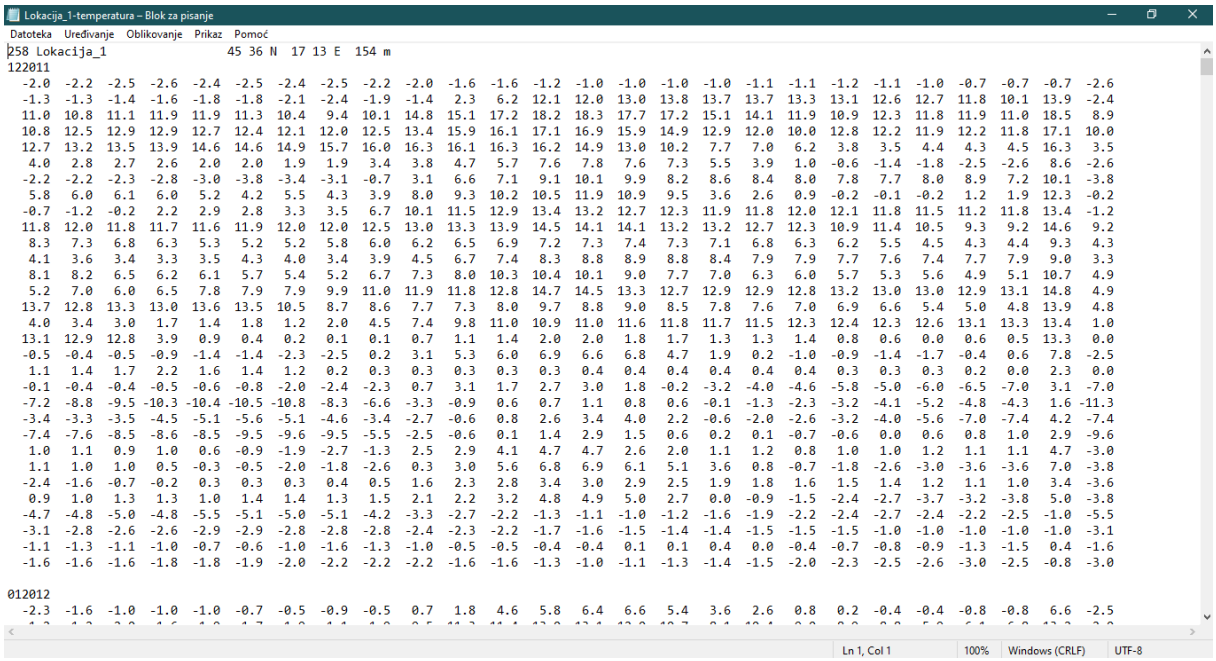
Na slici iznad je prikaz integriranog razvojnog okruženja PyCharm (engl. integrated development environment - IDE), koje se koristi u programiranju, za programski jezik Python, u kojem je razvijen jednostavni ETL alat, koji se koristi za praktični dio ovog rada. PyCharm pruža analizu koda, grafički program za uklanjanje pogrešaka, ispitivač integrirane jedinice, integracija sa sustavima za upravljanje verzijama i podrška za web razvoj.

5.2. Izrada ETL aplikacije

Da bi bilo moguće objasniti ETL na konkretnom primjeru, u praktičnom dijelu rada izrađena je ETL aplikacija u programskom jeziku Python, čiji se kod nalazi pod prilozima na kraju rada. Zadano je 9 tekstualnih datoteka (slika 9) u kojima su zapisane očitane temperature zraka na 9 mjernih postaja (slika 10). Problem je u tome što su podaci pozicionirani tako da ih ljudi jednostavno čitaju, a ETL zahtjeva transformaciju prikaza podataka u oblik pogodan za obradu podataka. Prema navedenom primjeru izrađena je aplikacija koje će podatke pogodne za čitanje konvertirati u datoteku i u oblik pogodan za bazu podataka. Obrađeni podaci se pohranjuju u dvije različite datoteke. Prva datoteka „izmjerene-vrijednosti.txt“ sadrži podatke o lokaciji (Naziv lokacije, Vrijeme i temperaturu). Druga datoteka „podaci-lokacija.txt“ sadrži informacije o lokaciji (Šifra lokacije, Naziv lokacije, GPS, Nadmorska visina).



Slika 9: Datoteke zadatka [12]



Slika 10: Prikaz očitanih temperatura 9 mjernih postaja [12]

5.3. Kod ETL aplikacije s komentarima

– Aplikacija započinje uvođenjem biblioteka i funkcija koje omogućavaju daljnji rad:

```
from datetime import datetime, timedelta
import glob, os
from timeit import default_timer as timer
```

– pokretanje tajmera:

```
start1 = timer()
```

– putanja do datoteka zadatka:

```
PATH = 'C:/Users/Matija/Desktop/Podaci/'
```

– lista s putanjama do svih datoteka u 'C:/Users/Matija/Desktop/Podaci/':

```
data_path = glob.glob(os.getcwd() + '/Podaci/*')
```

– sortiranje po imenu:

```
data_path.sort()
```

– spemanje putanja do txt datoteka koje se kreiraju:

```
vrijednosti_txt__path = os.getcwd() + '/izmjerene-vrijednosti.txt'
lokacija_txt__path = os.getcwd() + '/podaci-lokacija.txt'
```

– otvaranje dvije txt datoteke sa prethodno definiranim putanjama, postavljanje datoteka u mod za zapisivanje 'w' te se zapisuju podaci koje će svaka pojedinačna datoteka sadržavati:

```
vrijednosti_txt = open(vrijednosti_txt__path, 'w')
vrijednosti_txt.write('Naziv lokacije' + ' ' + 'Vrijeme' + ' ' + 'Temperatura' +
'\n')
lokacije_txt = open(lokacija_txt__path, 'w')
lokacije_txt.write('Šifra lokacije' + ' ' + 'Naziv lokacije' + ' ' + 'GPS' + ' ' +
'Nadmorska visina' + '\n')
```

– pojedinačno se otvara svaka txt datoteka definirana na 6. liniji koda, te se potrebne datoteke zapisuje u novu txt datoteku:

```
for file_path in data_path:

  data = open(file_path, 'r').read()
```

– lista s redovima od drugog do zadnjeg reda:

```
m_data = data.split('\n')[1:]
```

– prvi red s izbačenim višestrukim razmacima i izbačen „m“ na kraju reda, npr. '258 Lokacija_1 45 36 N 17 13 E 154', split() = ['258', 'Lokacija_1', '45', '36', 'N', '17', '13', 'E', '154']

```
lokacija = ' '.join(data.split('\n')[0].split()[0:-1])
```

– prvi član iz gornjeg primjera:

```
lok_sifra = lokacija.split()[0]
```

– drugi član:

```
lok_naziv = lokacija.split()[1]
```

– od trećeg do predzadnjeg člana:

```
gps = ' '.join(lokacija.split()[2:-1])
```

– zadnji član:

```
nadmorska_visina = lokacija.split()[-1]
```

– zapisivanje u datoteku s razmakom između podataka:

```
lokacije_txt.write(lok_sifra + ' ' + lok_naziv + ' ' + gps + ' ' +
nadmorska_visina + ' ' + '\n')
```

– kako u „m_data“ postoje spremljeni podaci u listi po redovima, jedan red - jedan član liste, svaki red se podijeli tako da se dobije svaki zapis kao zasebni član liste (dvodimenzionalna lista), npr. jedan član liste' -2.0 -2.2 -2.5 -2.6 -2.4 -2.5 -2.4 -2.5 -2.2 -2.0 -1.6 -1.6 -1.2 -1.0 -1.0 -1.0 -1.0 -1.1 -1.1 -1.2 -1.1 -1.0 -0.7 -0.7 -0.7 -2.6', taj isti član podijeljen (split) u 26 članova (onoliko koliko ima zapisa): ['-2.0', '-2.2', '-2.5', '-2.6', '-2.4', '-2.5', '-2.4', '-2.5', '-2.2', '-2.0', '-1.6', '-1.6', '-1.2', '-1.0', '-1.0', '-1.0', '-1.0', '-1.1', '-1.1', '-1.2', '-1.1', '-1.0', '-0.7', '-0.7', '-0.7', '-2.6']

```
a = []
date = 0
a = [i.split() for i in m_data]
```

– iteriranje kroz dobivenu 2D listu i provjerava se koliko članova ima podliste

```
for sublist in a:
```

– ako ima 26 članova to znači da ima 24 zapisa po satu i min i max vrijednosti:

```
if len(sublist) == 26:
```

– svi članovi se zapisuju od prvog do 24-og (ili do 2. odostraga) u datoteku:

```
for item in sublist[0:-2]:
```

– za svaki zapis povećava se vrijednost sata za jedan:

```
date += timedelta(hours=1)
```

– ako podlista ima samo jedan član, znaci da je to npr. "122011" koji označava mjesec i godinu zapisa:

```
elif len(sublist) == 1:
```

– iz zapisa se izvlači godina, odnosno od druge znamenke pa do kraja, npr. "2011" zatim se izvlači mjesec, sublist [0][0:2], prve dvije znamenke, npr. "12" koje se koriste kako bi se saznao datum izmjerenih vrijednosti, a vrijeme postavlja na 00:00:

```
date = datetime(int(sublist[0][2:]), int(sublist[0][0:2]), 1)
```

– zatvaranje i spremanje txt datoteka:

```
continue

vrijednosti_txt.close()
lokacije_txt.close()
```

– zaustavljanje tajmera i ispis brzine izvođenja aplikacije

```
end1 = timer()
print('Trajanje: ' + str(end1 - start1))
```

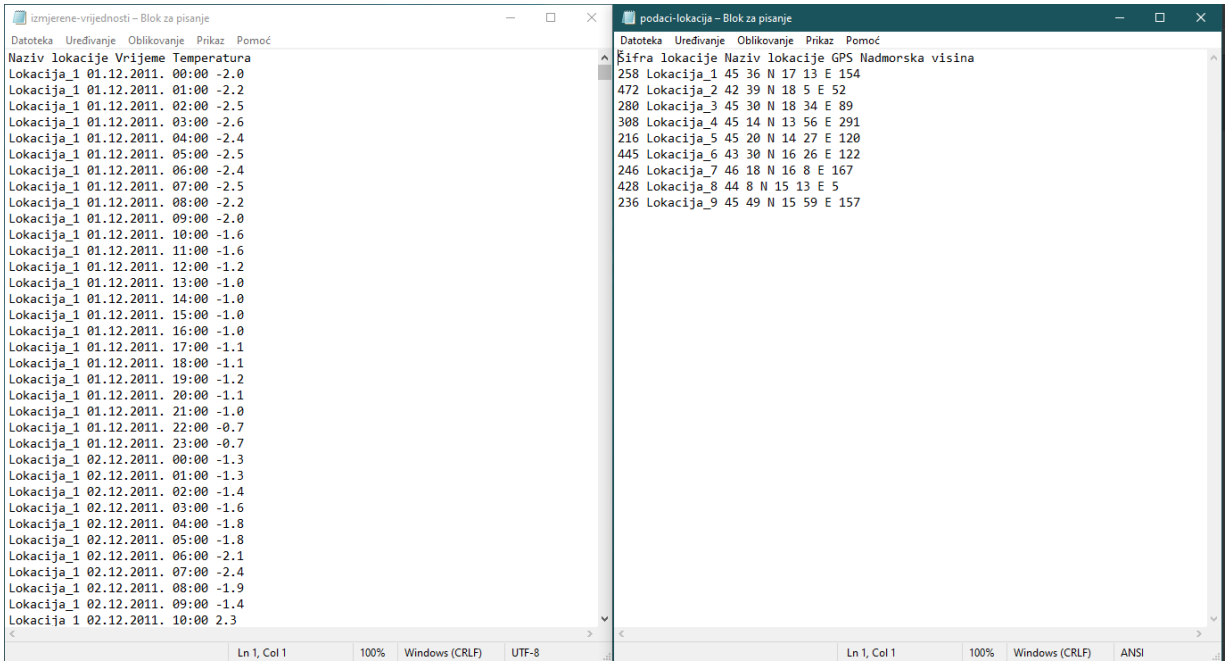
Cjelokupan kod aplikacije je dostupan u prilozima (Prilog 1).

5.4. Pokretanje ETL aplikacije

Nakon što je aplikacija napisana, sprema se na željeno odredište pod nazivom „ETL_aplikacija.py“. Pokretanjem aplikacije izvršava se čitanje skripte koja, kao što je i napisano u kodu aplikacije, stvara dvije tekstualne datoteke, transformira i zapisuje podatke u datoteke u oblik prikladan za baze podataka te ih sprema pod pripadajućim imenima, prikazano na slici ispod:

– „izmjerene-vrijednosti.txt“,

– „podaci-lokacija.txt“.



Slika 11: Tekstualne datoteke dobivene nakon pokretanja aplikacije [12]

5.5. Mjerenje brzine izvođenja ETL aplikacije

U kod ETL aplikacije je ubačen tajmer, koji daje informaciju o brzini izvođenja koda ETL aplikacije u milisekundama. Samim time brzina izvođenja koda ETL aplikacije pokazuje učinkovitost ETL aplikacije na različitim operativnim sustavima i različitim komponentama računala na kojem se izvodi pokretanje ETL aplikacije.

Navedene su brzine izvođenja ETL aplikacije deset puta za redom na različitim operativnim sustavima s različitim komponentama računala, a njihove prosječne vrijednosti mjerene u milisekundama iznose:

1. MS Windows 10 Enterprise, 64-bit (CPU: Intel Core i7-3630qm, RAM: 8 GB):
   – Trajanje: 4.488 ms

2. Linux Ubuntu 16.04 LTS, 64-bit (CPU: AMD A8-6600K, RAM: 16 GB):
    - Trajanje: 1.811 ms
3. OS X 10.14.6 Mojave, 64-bit (CPU: Intel Core i5-5350U, RAM 8 GB):
    - Trajanje: 1.838 ms

Kod Python ETL aplikacije je napisan na jednom računalu i operativnom sustavu MS Windows 10 Enterprise, 64-bit (CPU: Intel Core i7-3630qm, RAM: 8 GB) te je bez korekcija korišten na još dva različita računala s vlastitim operativnim sustavima. Kod ETL aplikacije u programskom jeziku C je pokrenut na računalu s operativnim sustavom Linux Ubuntu 16.04 LTS, 64-bit (CPU: AMD A8-6600K, RAM: 16 GB), te je korišten usporedbu brzine izvođenja ETL aplikacije napisan u programskom jeziku Python.

Podatak o brzini izvođenja ETL aplikacije u programskom jeziku C:

1. Linux Ubuntu 16.04 LTS, 64-bit (CPU: AMD A8-6600K, RAM: 16 GB):
    - Trajanje: 0,001812 ms

Unutar informatičke zajednice česte su kritike na brzinu izvođenja kodova Python programa koja je u usporedbi s programskim jezikom C i do tisuću puta sporija. Razlog sporosti programskog jezika Python je taj što je Python interpreterski jezik i potreban mu je interpreter. Interpreter je program koji u realnom vremenu izvršava izvorni kod napisan u nekom programskom jeziku, umjesto da ga, prije izvršavanja cijelog prevede u strojni jezik, što inače radi jezični prevoditelj ili kompajler. Programski jezik C je kompajlerski jezik te je iz tog razloga brži od Python-a [26].

6. Vrste ETL alata

Komercijalni ETL alati koji postoje na tržištu u puno slučajeva su jednostavniji način za obavljanje poslova dohvaćanja, transformiranja i obrade podataka, jer je lakše upravljati ETL alatom nego pisati vlastiti kod. Oni omogućuju korisnicima da dizajniraju i izvršavaju ETL procese, a programerima daje vizualni prikaz ETL procesa putem ETL tokova. Ti su tokovi samoopisani, tako da programeri ne trebaju dodatne komentare da bi ih shvatili. Stoga je lakše pregledati i promijeniti ETL postupak na ETL platformi, a ne baviti se dugačkim redovima kodova, posebno tijekom dugog vremenskog razdoblja.

Komercijalne alate održava tvrtka koje je specijalizirana za taj posao, a ne sami programeri, ipak svi programeri znaju da održavanje ETL koda nije dovoljno. Ovisno o programeru koji ga je napisao, može biti u Javi, SQL-u, Pythonu, C-u ili nekom drugom programskom jeziku. Kod može biti vrlo dobro strukturiran i dobro organiziran, ali i ne mora tako biti. Izvršiti optimizacije ili čak promijeniti ETL postupak može biti vrlo dugotrajan i skup proces. S druge strane kod komercijalnih alata postoje nadogradnje i različite verzije alata koje taj posao mogu olakšati.

Da bi se moglo odgovoriti na pitanje, da li je jednostavnije kupiti komercijalni alat ili programirati vlastiti kod, moraju se usporediti prednosti jednog i drugog načina ETL procesa [27,28].

6.1. Prednosti komercijalnih ETL alata

Prednosti komercijalnih ETL alata su [22,27,29]:

- Brži, jednostavniji i jeftiniji razvoj. Trošak izrade alata će se nadoknaditi ubrzo nakon kupnje alata. Što je veći projekt, to će se prije nadoknaditi trošak izrade alata. Mjesečna pretplata na komercijalni ETL alat može iznositi nekoliko stotina dolara ili eura, dok bi programera trebalo platiti isto toliko ako ne i više, dok bi razvoj aplikacije mogao potrajati i do nekoliko mjeseci,

- ETL alate mogu učinkovito koristiti ljudi sa širokim poslovnim vještinama te koji nisu profesionalni programeri,
- Operativnost – Komercijalni ETL alati su više organizirani nego ručno kodirani ETL alati, iz razloga što se pri pisanju vlastitog koda mora paziti da se generiraju dobro oblikovani zapisnici, obrade iznimke i pogreške te se sve pohrani u dobro organizirano spremište. Komercijalni ETL alati vode računa o svemu tome umjesto programera,
- Mnogi ETL alati imaju integrirana spremišta metapodataka koja mogu sinkronizirati metapodatke iz izvorišnih sustava, ciljanih baza podataka i alata poslovne inteligencije, a mogu čak i automatski generirati metapodatke na svakom koraku ETL procesa, iz razloga osiguravanja postojane metodologije koju svi programeri moraju slijediti,
- U slučaju pogrešaka komercijalni ETL alati moraju pružati podršku za upravljanje međuovisnostima pojedinih dijelova sustava i mogućnost rukovanja pogreškama, te mogućnost rukovanja svim vrstama složenih pretvorbi podataka,
- Komercijalni ETL alati nude linijsko kodiranje i sposobnost komprimiranja podataka,
- Odlične performanse za čak i za vrlo velike skupove podataka,
- Komercijalni ETL alat može lako riješiti problem prilikom mogućeg zastoja (engl. deadlock) prijenosa podataka među serverima,
- Prilikom promjene sheme, većina alata će izvršiti automatsku analizu utjecaja promjena za povezane procese i aplikacije.

Korištenje komercijalnih ETL alata ima puno prednosti, te se na temelju navedenih karakteristika može se zaključiti da su takvi alati jednostavniji za korištenje, vremenom postaju sve lakši za upotrebu, trebaju manje održavanja, bolje su organizirani, mogu uključivati jednostavnije upravljanje tijekom rada i obično koštaju manje, iako ne moraju svi ETL alati posjedovati navedene opcije. Prednosti komercijalnih ETL alata ujedno su i nedostaci ručno kodiranih ETL alata i obratno.

6.2. Prednosti ručno kodiranih ETL alata

Prednosti ručno kodiranih ETL alata su [22,27,29]:

- Ručnim kodiranjem ETL alata dobiva se puno više fleksibilnosti. Mogu se napisati složene transformacije ili jedinstveni algoritmi koje grafičko sučelje kod komercijalnih ETL alata ne može pružiti. Ako korisnici zahtijevaju takvu razinu fleksibilnosti, onda ovakav način postaje nužan, ne samo koristan,
- Kod objektno orijentiranog programiranja može se zadržati konzistentnost transformacije podataka,
- Moguće je izravnije utjecati na upravljanje metapodacima u ručno kodiranim alatima, ali je potrebno stvoriti vlastito sučelje za metapodatke,
- Pristup temeljen na komercijalnim alatima ograničiti će korisnika na jedan jedinstveni skriptni jezik, ali sustav s ručnim kodom može se razviti na uobičajenim i dobro poznatim skriptnim jezicima.

Slika 12: Korisničko sučelje programa za ručno kodiranje - Python [12]

Nakon analize prednosti komercijalnih ETL alata i prednosti ručno kodiranih ETL alata (ručno kodirani ETL alat prikazan na slici iznad) može se doći do zaključka da

više pozitivnih argumenata imaju prednosti komercijalnih ETL alata, ali još bolji odgovor bi bio „ovisi o potrebama korisnika“. Prije svega, tvrtke koje se odlučuju na kupnju ili ručno kodiranje ETL alata moraju znati za što će im ti alati koristiti i koliko fleksibilni ti alati moraju biti s obzirom na posao koji bi obavljali [27].

7. Nadzor ETL sustava

Nadzor ETL sustava obavlja tim odgovoran za nadzor ETL sustava. Posao tima je da osvježava skladište podataka po dogovorenim intervalima ili kontinuirano, a ispunjavanje te odgovornosti ne može uspjeti bez učinkovitih i upornih učitavanja podataka. ETL tim mora pratiti i ocjenjivati ETL poslove kako bi se uvjerili da učinkovito djeluju i učitavaju podatke u skladište podataka.

ETL tim za nadzor uzima u obzir mnoge aspekte procesa. Resursi izvan opsega ETL sustava kao što su hardver i infrastruktura administracije i upotrebe, igraju značajnu ulogu u ukupnoj učinkovitosti ETL sustava. Ovdje je navedeno nekoliko pokazatelja izvedbe ETL sustava koji govore kako dobro izvršavati procese nadzora ETL sustava. Pokazatelji su dio operativnih metapodataka i trebali bi se pohraniti u spremište kako bi bili analizirani tokom vremena od strane ETL tima za nadzor podataka [27].

7.1. Mjerenje specifičnih pokazatelja učinkovitosti ETL procesa

Korisnici koji su izloženi sustavu ili administraciji baze podataka svjesni su da postoje specifična mjerenja za različite pokazatelje učinkovitosti ETL procesa. Pokazatelji ETL procesa specifični su za fizičko kretanje i upravljanje stvarnim podacima. Oni su korak ispred tipičnih pokazatelja uspješnosti. To znači da se ne misli na mjerenje na razini operacijskog sustava ili hardverskih resursa, već unutar samog ETL procesa.

Pokazatelji učinkovitosti ETL procesa [27]:

- Trajanje u sekundama – Ova jednostavna računica je temelj za sve ostale proračune. Trajanje je razlika između vremena početka i vremena završetka ETL procesa u sekundama. Na primjer ako ETL proces započne u 6:00 u jutro, a završi u 6:15 sati, trajanje ETL procesa iznosi 900 sekundi,

- Obrađenih redova u sekundi – Označava ekvivalent za učitane redove u sekundi izračuna. Na primjer 1 500 000 redaka obrađenih u 20 minuta (1 500 000 / (20*60) = 1250) iznosi 1250 redaka u sekundi,
- Pročitanih redova u sekundi – Broj redaka, koji su dohvaćeni iz izvornog sustava, podijeljen s trajanjem u sekundama. Broj redaka može se smanjivati ili povećavati ovisno o ETL procesu,
- Zapisani redova u sekundi – Broj zapisanih redova (koji su transformirani, podijeljeni s trajanjem u sekundi) može biti veći ili manji od pročitanih redaka. To se događa u slučaju kada se jedan red učitanih podataka transformira u oblik gdje se dobiva više redova,
- Propusnost – Broj redaka obrađenih u sekundi i pomnoženo s brojem bajtova u svakom redu. Propusnost, kao i kod svih mjerenja performansi, vrijednost je koja bi se trebala koristiti kao osnova za poboljšavanje procesa.

Većina ETL alata pruža potrebne mjerne podatke za izvedbu ETL procesa. Mjerenje koje najviše ukazuje na učinkovitost ETL procesa je utrošeno vrijeme na izvršavanje određenog postupka obrade podataka. Cilj ETL sustava, osim stvaranja kvalitetnih informacija, je učitavanje skladišta podataka unutar dodijeljenog vremenskog prozora za učitavanje. Ako procesu treba 20 minuta za obradu neke količine podataka, potrebno je znati koliko je memorijsko zauzeće obrađenih podataka. Dali je to 50 milijuna redaka u tih 20 minuta ili samo 100 redaka. Iz tog razloga koriste se mjerenja specifičnih pokazatelja učinkovitosti ETL procesa [27].

7.2. Mjerenje pokazatelja učinkovitosti infrastrukture

Mnoga mjerenja pokazatelja učinkovitosti infrastrukture su dostupna putem različitih aplikacija za nadzor ili su zapisana u izvješćima ručno kodiranih ETL aplikacija.

Mjerenja koja direktno ukazuju na usko grlo u ETL procesu uključuju [27]:

- Iskoristivost procesora,
- Dodjela memorije,

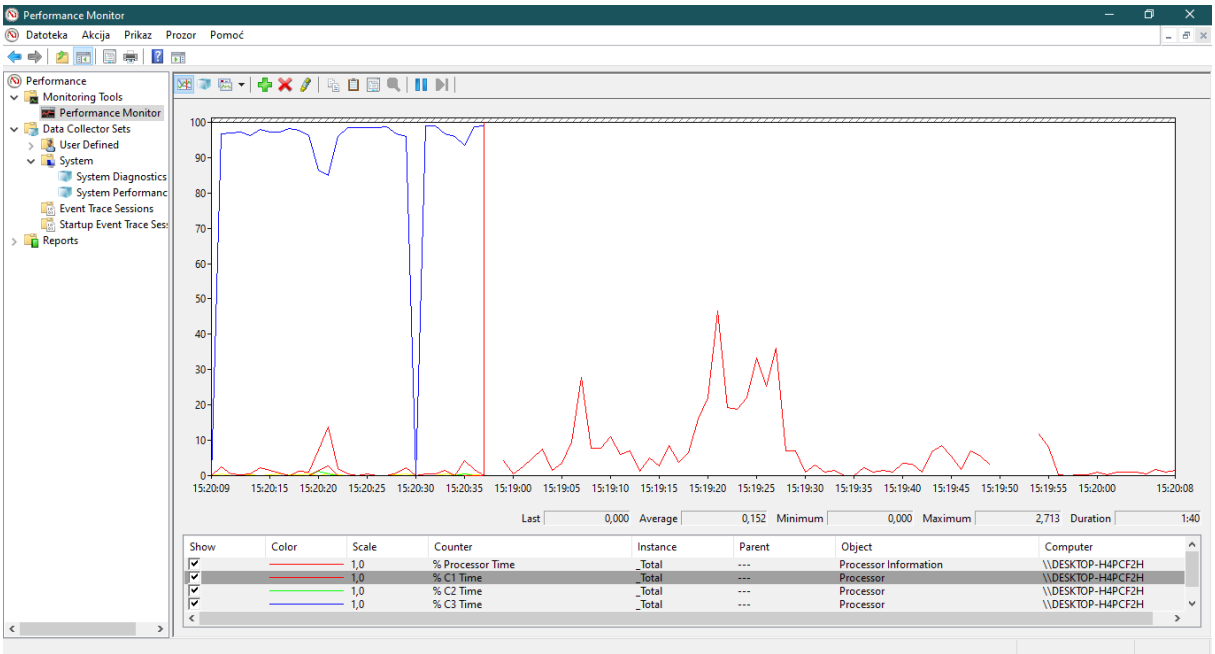
– Natjecanje za pristupom sadržaju.

Indikator mrežnog prometa može utjecati na proces ETL-a, ali mjerenja nisu toliko stabilna i postojana da bi ih se moglo koristiti s pouzdanošću. Štoviše, podrijetlo mrežnog prometa je izuzetno teško identificirati ili duplicirati u testnom okruženju. Ako postoji sumnja za mrežnim problemom, za pomoć se potrebno javiti mrežnom administratoru [27].

7.2.1. Iskoristivost procesora

Proces ETL-a se pokreće na središnjoj procesorskoj jedinici (engl. Central Processing Unit - CPU) svog poslužitelja. Većina ETL poslužitelja sadrži više procesora za obradu velike količine izračuna potrebnih za dohvaćanje, transformaciju i učitavanje podataka u skladište podataka.

U operacijskom sustavu Windows može se koristiti monitor performansi (engl. performance monitor), grafičko korisničko sučelje koje omogućuje dodavanje novih brojača već dostupnim brojačima performansi, prikazan je na slici ispod.



Slika 13: Monitor performansi u operacijskom sustavu Windows [12]

Alati za nadzor CPU-a pokazuju dali se ETL proces ravnomjerno raspoređuje po svim dostupnim procesorima i procesorskim jezgrama i dali procesori često dosežu svoje maksimalne kapacitete tijekom ETL postupka [27].

7.2.2. Dodjela memorije

Memorija nasumičnog pristupa (engl. Random Access Memory - RAM) može se dodijeliti ETL procesu. Prije svega, memorija mora biti fizički dostupna na serveru. Prilikom kupovine komercijalnog ETL alata, proizvođač bi morao biti u mogućnosti pružit hardverske specifikacije i preporučiti koliko RAM-a će biti potrebno za obradu podataka bez usporavanja performansi. Ako ETL proces intenzivno koristi dostupni tvrdi disk tada će navedeni postupak biti znatno sporiji. Odgovarajuća dodjela memorije u ETL proces utječe na performanse transformacije više nego bilo koja druga postavka, zato je potrebno posebno se pobrinuti za RAM.

ETL alat bi trebao moći pokazati koliko je memorije dodijeljeno za svaki postupak, koliko se zapravo memorije koristi u procesu i u kojoj mjeri se koristi virtualna memorija (memorija spremljena na disk). Neki ETL alat čine upravljanje memorijom potpuno preglednim i jednostavnim, dok se kod ostalih određene konfiguracije moraju ručno postaviti.

Postavke koje se mogu naći u nekim ETL alatima uključuju [27]:

- Zajedničku memoriju – Kada ETL alati čitaju i upisuju podatke, tada koriste dodijeljeno područje u memoriji koji sa naziva zajednička memorija. Zajednička memorija je memorija gdje se redovi podataka zaustavljaju prije ulaska ili izlaska iz fizičkog dijela transformacije ETL procesa,
- Veličinu bloka međuspremnika – pravilno postavljanje veličine bloka međuspremnika ovisi o veličini redova podataka u procesu transformacije. Ako ETL alat zahtjeva prilagođene prilagodbe veličine međuspremnika, dobavljač ETL alata može preporučiti izračun za optimalne postavke.

Pri ručno kodiranom ETL alatu moguće je posebnim naredbama pratiti dodjelu memorije, pratiti izvješća o upotrebi virtualne i stvarne memorije te pratiti aktivnosti na disku [27].

7.2.3. Sadržaj servera

Još jedan potencijalni „ubojica“ performansi javlja se prilikom pokušaja korištenja istog resursa od strane više procesa. Najčešće se događa da dva ETL procesa pokušaju pristupiti istim podacima te uzrokuju zastoj. Proces A pokušava zaključati proces B, a proces B pokušava zaključati proces A i sustav može biti blokiran. Sustav za upravljanje bazom podataka (DBMS) obično obavlja dobar posao u upravljanju pristupa podacima, ali prije ili kasnije u karijeri ETL programera će se dogoditi situacija kada će doći do zastoja servera. Procesi se neprestano natječu za resurse i sudari su neminovni, ali najbolja obrana je izbjeći istovremene procese, osim kada svaki proces ima namjenske procesne tokove i određene particije podatka.

Ostale vrste natjecanja za pristup sadržaju [27]:

- Memorija – kada se na istom poslužitelju pokreću mnoge aplikacije ili procesi, svakome od njih je potreban fizički RAM za rad. Nažalost, RAM je ograničen resurs a pojedinačni procesi se moraju boriti za to,
- Disk – većina diskova ima ograničen broj pristupa i količine podataka koji se mogu zapisivati ili čitati u nekom trenutku. Kada se dosegne granica zapisivanja i čitanja u nekom trenutku, procesi moraju čekati u redu za pristup disku,
- Baza podataka – ako ETL procesi pokušaju ažurirati podatke neke tablice u isto vrijeme, također može doći do zastoja, međusobnog blokiranja, ili neodređenog čekanja u redu,
- Procesor – ponekad može doći do paralelne obrade na softverskoj razini, ako hardver nije konfiguriran za paralelno pokretanje. Kada postoji više procesa koji se pokušavaju istovremeno pokrenuti, može doći do preopterećenja CPU-a i smanjenje performansi obrade podataka [26].

7.3. Optimizacija ETL procesa

Do sada su opisane aktivnosti u okviru opsega ETL sustava, kao i hardvera i infrastrukture ETL sustava. Četvrta ključna komponenta koja služi za pretvaranje sirovih podataka kupcu u koristan format su mjere korištenja skladišta podataka. Mjerenja opisana u ovom odlomku neizravno utječu na ETL sustav, ali ih je važno zapaziti i analizirati jer imaju utjecaj na proces učitavanja podataka.

ETL tim bi trebao iskoristiti izvješća o korištenju skladišta podataka i potražiti mogućnosti za promjenu rasporeda opterećenja, izmjenu opterećenja učestalosti ili eliminirati održavanje poslova koji učitavaju tablice koje nisu bile otvarane i izmijenjene dugo vremena. Ako se takvoj tablici pristupa samo prvog dana u mjesecu, ne bi se trebala svakodnevno ažurirati. To dovodi do veće učinkovitosti ETL procesa. Još jedan faktor povećanja učinkovitosti ETL procesa može se postići analizom indeksa upotrebe. Veliki dio obrade ETL procesa uključuje indekse obnove u skladištu podataka nakon svakog opterećenja. Upotreba mjerenja koja se koriste za upravljanje procesima ETL-a uključuju [27]:

- Upotreba tablice – sadržaj izvještaja o upotrebi tablice može varirati, ali korisno izvješće može sadržavati popis svake tablice, vremensku oznaku prvog i zadnjeg pristupanja tablici, broj upita različitih korisnika koji traže pristup tablici, i broj pregleda tablice. Tablice kojima se najčešće pristupa trebaju biti najviše dostupne. Tablice koje se upotrebljavaju samo jednom mjesečno mogu se isključiti iz svakodnevnog procesa opterećenja i prebaciti u mjesečno opterećenje. Tablice koje imaju kontinuiranu upotrebu pravi su kandidati za stavljanje na vrh liste dostupnosti. Visoko dostupne tablice imaju dvostruku strukturu učitanu u pozadini,
- Indeks upotrebe – indeksi su velika pomoć prilikom pronalaženja tablica skladišta podataka, ali su i teret ETL procesa jer svako učitavanje podataka prati i uključivanje i isključivanje indeksa. Kada arhitekt baze podataka gradi

baze podataka, on ima sklonost indeksirati[4] što veći mogući broj stupaca kako bi korisnik imao što bolje performanse,

- Agregatna upotreba – agregati[5] se obično ugrađuju kao i indeksi, radi poboljšanja performansi. Kao u slučaju s indeksima, neki agregati su izgrađeni, ali nikada nisu korišteni,
- Uspavani (ne korišteni podaci) – izvještaj o stanju takvih podataka je uvijek zanimljiv jer se skladište podataka stvara kao rezultat ispitivanja korisnika koji otkrivaju koji su elementi podataka potrebni za provođenje analize potrebnih za provedbu njihovih poslova. Izvješće o uspavanim podacima može pomoći ETL timu da identificira i dovede u pitanje učinkovitost mjera i atributa dimenzija koji se nikad ne odaberu.

Postoji nekoliko načina sakupljanja statistike o korištenju skladišta podataka. Neki sustavi za upravljanje bazama podataka nude podatke o korištenju, a može se zaposliti dodatni tim za praćenje statistike upotrebe skladišta podataka pomoću posebno izrađenih aplikacija za tu namjenu [27].

[4] Dodjela oznaka (indeksa) dokumentu, kojima se omogućuje njegovo pronalaženje u skupu drugih dokumenata, npr. u arhivi, dokumentaciji, memoriji računala, na internetu.
[5] Cjelina izgrađena od nekoliko zasebnih dijelova.

8. Neispravni podaci u skladištu podataka

Arhitekt sustava mora znati što učiniti s neispravnim podacima koji se pojavljuju u skladištu podataka. Prva pretpostavka je da neispravni podaci dolaze u skladište podataka iznimno tj. u rijetkim slučajevima. Ako se neispravni podaci unose u skladište podataka u velikim količinama, tada je potrebno pronaći grešku u ETL alatu i ispraviti je. Nerijetko, čak i kada postoji najbolji ETL alat, u slučaju pojave greške među podacima u skladištu podataka, moguće je koristiti više različitih metoda detekcije i korekcije grešaka.

Prije postupka obrade potrebno je definirati razmjere pristizanja neispravnih podataka u skladište podataka, te se nakon toga pristupa opciji obrade neispravnih podataka red-po-red, te sve-ili-ništa [30]:

- Obrada neispravnih podataka red-po-red - Obrada neispravnih podataka red-po-red je čest pristup na koji se korisnici odlučuju. Koristeći ovaj pristup, ETL proces će filtrirati sve sumnjive redove podataka baziranim na kriterijima koji su postavljeni u prethodnoj fazi, nakon toga će se pročistiti, preusmjeriti ili obrisati svaki od tih redova,
- Obrada neispravnih podataka sve-ili-ništa - U poslovnoj praksi moguća je pojava slučajeva koji zahtijevaju obradu sumnjivih podataka pristupom sve-ili-ništa kada cijeli pokrenuti proces obrade prolazi uspješno ili neuspješno. Sve-ili-ništa proces zahtjeva mehanizam koji se pokreće prilikom neuspjeha te dovodi do povratka na staro stanje podataka ili potpuno brisanje podataka. Npr. u slučaju ako postoji samo jedan red podataka koji je prema zadanim kriterijima neispravno zapisan, može dovesti do brisanja cijelog skupa podataka. U tim slučajevima je ponekad dobro ne imati zapisane podatke nego imati djelomično (samim time i neispravno) zapisan skup podataka u odredište.

Sve-ili-ništa procesi su puno kompleksniji nego red-po-red procesi, te je bitno da ih se koristi kada to zahtijevaju poslovne ili tehničke potrebe. Kada se ne zna koji proces obrade sumnjivih podataka koristiti, uvijek treba krenuti od procesa red-po-red [30].

8.1. Obrada neispravnih podataka u ETL-u

Dohvaćanje, transformacija i učitavanje su tri procesa odgovorna premještanje podataka u zajedničku bazu podataka. Navedeni proces može biti proces s najviše problema integracije podataka, jer pogreška u jednom koraku procesa može uzrokovati neispravne podatke na kraju procesa. Svaki sustav dolazi s vlastitim nizom jedinstvenih problema s kojima se korisnik može susresti.

Nakon što se utvrde neispravni podaci i definiraju do koje mjere se smatraju neispravnima, pristupa se otklanjanju problema koji je doveo do pojave neispravnih ili beskorisnih podataka. Najčešće se pristupa nesređenim podacima nepotpunih vrijednosti, neusklađenih oblikovanja, neispravnih zapisa ili besmislenih vrijednosti. Otklanjanje problema će uvijek biti tehničke prirode od strane programera, a prilikom toga mora se paziti da kako će to utjecati na tvrtku čiji su podaci u pitanju.

Navedeni su pristupi čišćenja neispravnih podataka [31]:

- Analiza podataka - Kako bi bile otkrivene vrste neispravnih podataka koje trebaju biti uklonjene, potrebna je detaljna analiza podataka. Pored ručnog pregleda podataka ili uzoraka podataka, programi za analizu podataka trebaju se koristiti za dobivanje metapodataka o svojstvima podataka i otkrivanju problema s kvalitetom podataka,
- Definicija transformacije tijeka rada i pravila mapiranja – ovisno o količini izvora podataka i stupnju „neočišćenosti“ podataka, možda će se morati izvršiti velik broj transformacija podataka. Rani koraci čišćenja podataka mogu ispraviti probleme s jednim izvorom podataka i pripremiti podatke za integraciju. Kasniji koraci čišćenja bave se sa problemima integracije podataka sheme i baze podataka i čišćenjem više izvora podataka, kao što su npr. duplikati,
- Verifikacija – Ispravnost i učinkovitost transformacije tijeka rada i definicija transformacija moraju biti testirane i vrednovane, npr. na uzorku ili kopiji izvornih podataka. Može biti potrebno i više ponavljanja koraka analize i provjere, npr. jer neke greške postaju očite tek nakon primjene nekih transformacija,
- Transformacija – Izvođenje koraka transformacije pokretanjem protokola ETL za dohvaćanje i osvježavanje skladišta podataka,

- Povratni tok očišćenih podataka – nakon uklanjanja pogrešaka, očišćene podatke treba zamijeniti s neočišćenim podacima u originalnim izvornima, kako bi se zastarjelim aplikacijama također dali očišćeni podaci, i da bi se izbjeglo ponavljanje procesa čišćenja.

9. ETL stvarnog vremena

Kako se tvrtke bore za konsolidaciju (spajanje) strujanja podataka iz više izvora, usprkos neprekidnom rastu podataka i potražnji za pravovremenom i preciznom poslovnom inteligencijom, više tvrtki traži prelazak sa serijskog orijentiranog ETL sustava na ETL u stvarnom vremenu.

ETL se odnosi na procese dohvaćanja, transformacije i učitavanja podataka iz različitih izvora podataka u centralizirano skladište podataka za izvještavanje i analizu. Međutim, pomoću uobičajenog ETL alata, primjena ETL-a je općenito složen i dugotrajan proces, koji često dovodi do skupih kašnjenja i rizika u djelatnostima poslovne inteligencije. Tvrtke koje dugoročno razmišljaju zainteresirane su za ETL rješenja u stvarnom vremenu koja im mogu pomoći pratiti količinu i brzinu dolaznih podataka i mogu brzo reagirati na promjene na tržištu [32].

Dva su glavna načina kako su tvrtke uspjele ubrzati ETL projekte poslovne inteligencije [27]:

1. Upotreba ETL rješenja koje generira ETL skripte i tokove te poboljšava performanse i dosljednost ETL radnih tokova. Omogućujući korisnicima ugrađene konektore za različite izvore i ciljeve, integrirana spremišta metapodataka i intuitivno grafičko korisničko sučelje s kojeg mogu nadzirati i upravljati protocima podataka, ETL integracijsko rješenje čini izvršavanje, izmjenu i rješavanje problema ETL procesa jednostavnijim i lakšim,
2. Korištenje rješenja za automatizaciju skladišta podataka na način da rješenja automatiziraju zadatke pohrane podataka koji se ponavljaju i koji zahtijevaju rad kao što su generiranje shema i ETL kodiranje, omogućujući ne samo ETL automatizaciju, već i automatizaciju cjelokupnog životnog ciklusa skladištenja podataka od dizajna i stvaranja skladišta podataka do analize utjecaja i upravljanja promjenama.

9.1. Potrebe korisnika ETL-a stvarnog vremena

Da bi potrebe timova za skladište podataka i poslovnu inteligenciju bile uspješno zadovoljene, potrebno je dizajnirati tokove podataka s obzirom na brzinu isporuke podataka. U tome pomaže podjela od tri kategorije nazvane trenutačno, unutardnevno i dnevno [33]:

1. Trenutačno – znači da su podaci vidljivi na zaslonu i predstavljaju pravo stanje izvora transakcija u svakom trenutku. Kada se status izvornog sustava mijenja i podaci na zaslonu se odmah mijenjaju. Izvorni sustav je odgovoran za ažuriranje podataka bez kašnjenja na ekranima udaljenih korisnika. Primjer situacije trenutačnog rješenja u stvarnom vremenu je praćenje stanja zaliha, kada proizvođač vrši uvid u dostupne zalihe na skladištu,
2. Unutardnevno – znači da se podaci vidljivi na zaslonu ažuriraju više puta dnevno, ali ne znači da su podaci koji su prikazani na zaslonu istiniti trenutačnom stanju podataka. Primjer unutardnevnog prikaza podataka su dionice na burzi, kada su podaci aktualni unutar 15 minuta ali nisu trenutačni,
3. Dnevno – znači da podaci vidljivi na zaslonu vrijede od trenutnog preuzimanja ili obnavljanja na kraju prethodnog dana. Često se ETL procesi dohvaćanja, transformacije i učitavanja podataka na izvornom sustavu izvode na kraju radnog dana. Dnevno ažurirani podaci obično sadrže čitave serije podataka prethodnog dana pripremljenih od strane izvorišnog sustava i to predstavlja najjednostavniji i najsigurniji scenarij ETL procesa.

10. ZAKLJUČAK

Cilj završnog rada je bio opisivanje procesa dohvata, transformacije i učitavanja u skladištima podataka, te opisati i usporediti vlastita i komercijalna programska rješenja u navedenom postupku. Bilo je potrebno opisati i moguća rješenja postojećih problema pripreme podataka za skladišta podataka.

Ovisno o zahtjevima i potrebama tržišta, korisnik se na temelju prednosti i nedostataka pojedinih programskih rješenja mora odlučiti koja će programska rješenja koristiti u ETL postupku. Opcije koje se nude pri odabiru ETL alata su kupovina komercijalnog alata, korištenje open-source alata i pisanje vlastitih aplikacija.

U sklopu praktičnog dijela završnog rada, ETL aplikacija napisana u programskom jeziku Python je pokrenuta deset puta za redom na različitim operativnim sustavima i računalima s različitim komponentama. Brzina izvođenja aplikacije u Python-u je bila i do tisuću puta sporija nego kod brzina izmjerenih prilikom pokretanja aplikacije u programskom jeziku C.

Nakon pokretanja ETL aplikacije, od početnih devet tekstualnih datoteka formata pogodnog za ljude, formiraju se dvije tekstualne datoteke s uređenim prikazom podataka i sa svim korisnim podacima pogodnima za pohranu u bazu podataka.

Skladište podataka je živi sustav čiji se izvori i ciljevi informacija mogu promijeniti. Te promjene moraju se pratiti kroz životni vijek sustava bez promjene ili brisanja starih informacija u ETL procesu i tijek svakog pojedinog zapisa u skladištu podataka može se u idealnom slučaju rekonstruirati u bilo kojem trenutku u budućnosti.

LITERATURA

[1] Shaker H. Ali El-Sappagh: „A proposed model for data warehouse ETL processes“, King Saud University, 2011.

[2] https://www.talend.com/resources/what-is-etl/ (B. Babb; „What is ETL (Extract, Transform, Load)?“, pristupljeno 17.10.2019.)

[3] https://www.guru99.com/etl-extract-load-process.html („ETL (Extract, Transform, and Load) Process“, pristupljeno 17.10.2019.

[4] https://www.passionned.com/data-integration/etl/ („ETL: Extract Transform and Load“, pristupljeno 17.10.2019.)

[5] https://en.wikipedia.org/w/index.php?oldid=520752426 („Extract, transform, load“, pristupljeno 17.10.2019.)

[6] https://www.tiobe.com/tiobe-index/ („TIOBE Index for November 2019“, pristupljeno 17.10.2019.)

[7] https://www.softwaretestinghelp.com/best-etl-tools/ („15 Best ETL Tools In 2019“, pristupljeno 17.10.2019.)

[8] https://senturus.com/wp-content/uploads/2018/06/Data-modeling-in-Informatica-PowerCenter.png (pristupljeno 17.10.2019.)

[9] https://www.altova.com/ (pristupljeno 17.10.2019.)

[10] https://www.astera.com/ (pristupljeno 17.10.2019.)

[11] https://solutionsreview.com/data-integration/top-free-and-open-source-etl-tools-for-data-integration/ (T. King: „Top 12 Free and Open Source ETL Tools for Data Integration“, pristupljeno 17.10.2019.)

[12] Autorova slika

[13] Elmasri R; Navathe S: „Fundamentals of Database Systems Sixth Edition“ SAD, (2010.), ISBN-13: 978-0-136-08620-8

[14] Silberschatz A; Korth H; Sudarshan S: „Database system concepts - sixth edition“, New York, (2011.), ISBN 978-0-07-352332-3

[15] https://www.lucidchart.com/pages/database-diagram/database-schema („What is a Database Schema“, pristupljeno 17.10.2019.)

[16] http://www.crazytutorialpoint.com/dbms-tutorial/what-is-dbms.php („What is database management system(DBMS)“, pristupljeno 17.10.2019.)

[17] Vikas R; „A Comparative Study between ETL (Extract-Transform-Load) and E-LT (Extract-Load-Transform) approach for loading data into a Data Warehouse“ 2009,

[18] https://docs.oracle.com/database/121/DWHSG/extract.htm#DWHSG8278 („Database Data Warehousing Guide“, pristupljeno 17.10.2019.)

[19]https://www.immagic.com/eLibrary/ARCHIVES/GENERAL/WIKIPEDI/W121031E.pdf („Extract, transform, load“, pristupljeno 17.10.2019.)

[20] https://medium.com/hashmapinc/etl-understanding-it-and-effectively-using-it-f827a5b3e54d („ETL — Understanding It and Effectively Using It“, pristupljeno 17.10.2019.)

[21] https://www.stitchdata.com/etldatabase/etl-load/ („ETL Load“, pristupljeno 17.10.2019.)

[22] https://www.quora.com/What-are-the-advantages-of-using-an-ETL-tool-vs-handed-coding-And-is-Talend-a-good-ETL-tool (pristupljeno 17.10.2019.)

[23] https://www.passionned.com/the-7-biggest-benefits-of-etl-tools/ (Van der Linden R: „The 7 biggest benefits of ETL tools“, pristupljeno 17.10.2019.)

[24] Davenport R: „ETL vs ELT A Subjective View Part of the series of the Insource Commercial Aspects of BI discussion papers“ California State University, 2008.

[25] https://www.python.org/ (pristupljeno 17.10.2019.)

[26] https://hr.wikipedia.org/wiki/Python_(programski_jezik) (pristupljeno 7.1.2020)

[27] Kimball R; Caserta J: „The Data Warehouse ETL Toolkit“, SAD, 2004; eISBN: 0-764-57923-1

[28] https://www.xplenty.com/blog/using-an-etl-platform-vs-writing-your-own-code/ (Neumann. S: „Using An ETL Platform VS Writing Your Own Code“, pristupljeno 17.10.2019.)

[29] https://www.inetsoft.com/business/solutions/etl_pros_and_cons/ („ETL Pros and Cons“, pristupljeno 17.10.2019.)

[30] https://www.timmitchell.net/post/2017/02/16/managing-bad-data-in-etl/ (Mitchell. T: „Managing Bad Data In ETL“, pristupljeno 17.10.2019.)

[31] Rahm E; Hai Do H: „Data Cleaning: Problems and Current Approaches“, University of Leipzig, Germany.

[32] https://www.attunity.com/real-time-etl/ („Real-Time ETL“, pristupljeno 17.10.2019.)

[33] Kimball R; Ross M: „The Data Warehouse Toolkit: The Defi nitive Guide to Dimensional Modeling, Third Edition“, SAD, 2013; ISBN: 978-1-118-53080-1

PRILOZI

Prilog 1: Kod ETL aplikacije

```
from datetime import datetime, timedelta
import glob, os
from timeit import default_timer as timer

start1 = timer()

PATH = 'C:/Users/Matija/Desktop/Podaci/'

data_path = glob.glob(os.getcwd() + '/Podaci/*')
data_path.sort()

vrijednosti_txt__path = os.getcwd() + '/izmjerene-vrijednosti.txt'
lokacija_txt__path = os.getcwd() + '/podaci-lokacija.txt'

vrijednosti_txt = open(vrijednosti_txt__path, 'w')
vrijednosti_txt.write('Naziv lokacije' + ' ' + 'Vrijeme' + ' ' + 'Temperatura' + '\n')
lokacije_txt = open(lokacija_txt__path, 'w')
lokacije_txt.write('Šifra lokacije' + ' ' + 'Naziv lokacije' + ' ' + 'GPS' + ' ' + 'Nadmorska visina'
+ '\n')

for file_path in data_path:

  data = open(file_path, 'r').read()


  m_data = data.split('\n')[1:]
  lokacija = ' '.join(data.split('\n')[0].split()[0:-1])
  lok_sifra = lokacija.split()[0]
  lok_naziv = lokacija.split()[1]
  gps = ' '.join(lokacija.split()[2:-1])
  nadmorska_visina = lokacija.split()[-1]
  lokacije_txt.write(lok_sifra + ' ' + lok_naziv + ' ' + gps + ' ' + nadmorska_visina + ' ' + '\n')

  a = []
  date = 0

  a = [i.split() for i in m_data]

  for sublist in a:
    if len(sublist) == 26:
      for item in sublist[0:-2]:
        vrijednosti_txt.write(lokacija.split()[1] + ' ' + date.strftime("%d.%m.%Y. %H:%M") + ' '
+ item + '\n')
        date += timedelta(hours=1)
    elif len(sublist) == 1:
      date = datetime(int(sublist[0][2:]), int(sublist[0][0:2]), 1)
      continue

vrijednosti_txt.close()
lokacije_txt.close()

end1 = timer()
print('Trajanje: ' + str(end1 - start1))
```

POPIS KRATICA

COBOL	(engl. Common Business Oriented Language)
CPU	(engl. Central Processing Unit), središnja procesorska jedinica
CRM	(engl. Customer Relationship Management)
DBMS	(engl. Database Management System), sustav za upravljanje bazama podataka
DSA	(engl. Data Staging Area), područje prikazivanja podataka
DW / DWH	(engl. Data Warehouse), sustav temeljnog skladištenja podataka
EDI	(engl. Electronic Data Interchange)
ELT	(engl. Extract, Load, Transform), dohvaćanje-učitavanje-skladištenje
ETL	(engl. Extract-Transform-Load), dohvaćanje-transformacija-učitavanje
GUI	(engl. Graphical User Interface), grafičko korisničko sučelje
IDE	(engl. integrated development environment), integrirano razvojno okruženje
IMS	(engl. Information Management System), sustav upravljanja informacijama
ISAM	(engl. Indexed Sequential Access Method), indeksirani sekvencijalni pristup metoda
JSON	(engl. JavaScript Object Notation),
PDF	(engl. Portable Document Format)
RAM	(engl. Random Access Memory), memorija nasumičnog pristupa
SQL	(engl. Structured Query Language)
VSAM	(engl. Virtual Storage Access Method), metoda pristupa virtualnoj pohrani
XLS	(engl. eXceL Spreadsheet)
XML	(engl. EXtensible Markup Language)

POPIS SLIKA