ETL procesi

Nedić, Magdalena

Master's thesis / Diplomski rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Department of Mathematics / Sveučilište Josipa Jurja Strossmayera u Osijeku, Odjel za matematiku**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:126:004909

Sveučilište J.J. Strossmayera u Osijeku
Odjel za matematiku
Sveučilišni diplomski studij Matematika i računarstvo

Magdalena Nedić

ETL procesi

Diplomski rad

Osijek, 2022.

Sveučilište J.J. Strossmayera u Osijeku
Odjel za matematiku
Sveučilišni diplomski studij Matematika i računarstvo

Magdalena Nedić

ETL procesi

Diplomski rad

Mentor: izv. prof. dr. sc. Domagoj Matijević

Komentor: dr. sc. Mateja Đumić

Osijek, 2022.

Sadržaj

Uvod

U današnje vrijeme, brzih promjena i velikog tehnološkog napretka, kada vlada direktna konkurencija, čak i mala informacija može promijeniti tijek poslovanja za bilo koju branšu posla. Dakle, prava, točna i pravovremena informacija u većini slučajeva je ključ napretka pa tako i uspjeha. Stoga tvrtke trebaju biti u korak s vremenom i koristiti prave procese za integraciju svih relevantnih podataka iz različitih izvora. To je razlog zašto procesi i alati poslovne inteligencije sve više dobivaju na važnosti, a samim time postaju sve više područje interesa i ulaganja, kako velikih vodećih kompanija, tako i malih poduzeća.

Cijelokupan proces prikupljanja i obrade relevantnih podataka koji služe za ilustrativne prikaze i detaljne izvještaje odvija se upravo u skladištu podataka. Prvo poglavlje ovog rada definira i pobliže opisuje pojam skladišta podataka, dizajn i arhitekturu te ističe neke važne karakteristike tog podatkovnog okruženja. Osim toga, kao sastavni dio opisa dizajna skladišta podataka, u prvom poglavlju je stavljen naglasak i na specifičnosti kreiranja centralne baze kao i njenih komponentama: činjenica i dimenzija.

U brojnim literaturama koje pobliže opisuju i definiraju skladište podataka navode se četiri glavne sastavne komponente skladišta: centralna baza podataka, ETL sustav, meta podaci i alati za pristup i analizu. Kako su podaci ti koji svakom skladišu daju na vrijednosti i pomoću kojeg ono ispunjava svoju svrhu, ETL sustav postaje ključan element među nabrojenim komponetama. Razlog tome je što podaci pomoću ETL procesa dolaze u skladište. Drugo poglavlje ovog rada posvećeno je upravo ETL sustavima. U njemu se opisuju procesi ETL sustava: izdvajanje i kombiniranje podataka iz različitih izvora, transformacija podataka kao i njihovo učitavanje u zajednički prostor, odnosno centralnu bazu skladišta podataka.

U trećem poglavlju bavimo se praktičnim primjerom izrađenim na temu ETL sustav. Cilj samog rada bio je dizajniranje i izgradnja skladišta podataka koje sadrži informacije o vremenskim uvjetima i parametrima važnim za kvalitetu zraka. Ideja je da se podaci u skladištu podataka, koje nastaje kao rezultat ovog praktičnog projekta, koriste kao ulazni podaci za treniranje i testiranje algoritama strojnog učenja kojima je cilj provjeriti ovisnost koncetracije peludi u zraku s vremenskim prilikama i kvalitetom zraka.

Posljednje poglavlje posvećeno je primjenama ETL sustava koje prati pregled nekih istaknutijih komercijalnih rješenja za korištenje samog sustava. Za svako od navedenih komercijalnih rješenja dan je kratak opis i navedene su neke prednosti koje izdvajaju taj alat od sličnih alata na tržištu. Za koji od njih se odlučiti u primjeni je individualno pitanje i uvelike ovisi o zahtjevima i potrebama klijenta. Na samom kraju rada napravljena je usporedba kada je bolje koristiti komercijalno rješenje, a kada se ipak odvažiti na kreiranje prilagođenog rješenja.

1 Skladište podataka

U današnjem poslovnom svijetu koje je vrlo dinamično, organizacije se okreću tehnologijama i alatima temeljenim na oblaku za prikupljanje podataka, izvješćivanje i analizu. Skladište podataka u ovom procesu ima ključnu ulogu.

Skladište podataka je relativno sveobuhvatan i složen pojam, ne postoji točna i precizna definicija koja bi ga u potpunosti opisala. Iz tog razloga ovdje navodimo dvije interpretacije ovog pojma koje se mogu pronaći u [6]:

„*Skladište podataka je centralizirani sustav za pohranu podataka koji omogućuje pohranu, analizu i interpretaciju podataka kako bi se olakšalo donošenje odluka. Transakcijski sustavi, relacijske baze podataka i drugi izvori redovito dostavljaju podatke u skladišta podataka.*"

„*Skladište podataka može se definirati kao zbirka organizacijskih podataka i informacija izvučenih iz operativnih izvora i vanjskih izvora podataka. Podaci se povremeno povlače iz raznih internih aplikacija kao što su prodaja, marketing i financije, aplikacije korisničkog sučelja kao i vanjski partnerski sustavi. Ti se podaci zatim stavljaju na raspolaganje donositeljima odluka za pristup i analizu.*"

Osim skladišta podataka (*engl. Data Warehouse*) poznati su i drugi nazivi za ovaj pojam, neke od njih možete pogledati na sljedećem dijagramu:

Slika 1: Termini koji se koriste za skladište podataka
Izvor: [12]

Ono što što je važno za istaknuti, skadište podataka nije proizvod, niti alat, već okruženje. Skladište podataka bi mogli opisati i kao konstrukciju informacijskog sustava koji korisnicima omogućuje uvid u povijesne i trenutne informacije koje im služe za donošenje poslovnih odluka, a kojima je teško pristupiti ili ih je teško prikazati u tradicionalnoj pohrani. Jedna od najvećih prednosti skladišta podataka je što pruža novi dizajn pohrane koji omogućuje smanjenje vremena odgovora te poboljšava izvedbu upita za analitiku i izvješća. Skladište

podataka uglavnom podatke prikuplja pomoću ETL (ekstrahiraj, transformiraj, učitaj) sustava o kojem će biti više riječ u sljedećem poglavlju. Korisnici skladišta podacima pristupaju pomoću alata poslovne inteligencije, SQL naredbi i proračunskih tablica. Cilj skladišta podataka je stavljanje svih dostupnih podataka na jedno mjesto, kako bi se omogućila sveobuhvatna analiza kao i rudarenje podataka. Glavna prednost skladišta podataka je izvlačenje značajnih vrijednosti iz informacija nakupljenih tijekom vremena. Korištenje skladišta podataka bi organizacije trebalo dovesti do bržeg i lakšeg donošenja kvalitetnijih poslovnih odluka, pa posljednično s tim, do razvoja i napretka u poslovanju.

1.1 Vrste skladišta podataka

Prema [12] razlikujemo tri vrste skladišta podataka, a to su:

- **Skladište podataka za poduzeće** (*engl. Enterprise Data Warehouse (EDW)*) - je centralizirano skladište koje pruža usluge odlučivanja na razini cijelog poduzeća. Prednost ove vrste skladišta je u tome što omogućuje pristup međuorganizacijskim informacijama, omogućuje izvođenje složenih upita i nudi jedinstven pristup svakom korisniku kada je riječ o dizajnu skladišta i učitavanju podataka.

- **Operativna pohrana podataka** (*engl. Operational Data Store - ODS*) je specifična po tome što se osvježava u stvarnom vremenu i zato se najčešće koristi za rutinske aktivnosti. Važnost ove vrste skladišta najviše dolazi do izražaja kada sustavi skladišta ne podržavaju potrebe poslovanja za izvješćivanje.

- **Data Mart** - je podskup skladišta podataka koji služi za održavanje određenog odijela, regije ili poslovne jedinice. Svaki poslovni odijel bi trebao imati *data mart* za pohranu podataka, te se s vremena na vrijeme podaci iz *data marta* prebacuju u ODS koji potom podatke prosljeđuje u EDW, gdje se oni pohranjuju i koriste.

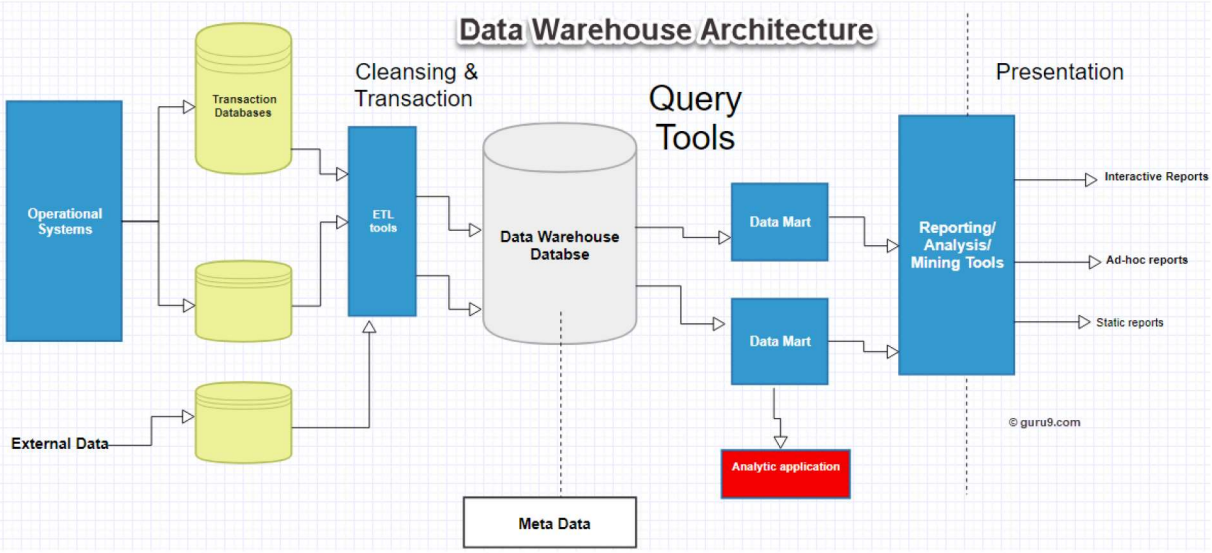
1.2 Arhitekura i karakteristike skladišta podataka

Skladište podataka temelji se na RDBMS (sustav upravljanja relacijskom bazom podataka) poslužitelju. To je centralno spremište informacija koje se okružuje ključnim komponentama kako bi cjelokupno okruženje bilo funkcionalno, upravljivo i pristupačno. Prema tome, arhitekturu skladišta podataka možemo podijeliti u četiri glavne komponente [3]:

- **Centralna baza skladišta podataka**: predstavlja temelj cjelokupnog okruženja i oko nje se formiraju ostale komponente skladišta. Ova baza podataka je implementirana na RDBMS tehnologiji te je samim time ograničena činjenicom da je ovaj sustav optimiziran za obradu transakcijskih podataka, a ne za skladištenje podataka. Stoga se koriste razni alternativni pristupi kako bi se ova baza prilagodila svrsi:
  - Relacijske baze u skladištu podataka raspoređene su paralelno kako bi se postigla skalabilnost i korištenje dijeljene memorije.
  - Koriste se nove strukture indeksa kako bi se zaobišlo skeniranje relacijske tablice i postigla veća brzina pretraživanja.
  - Koriste se višedimenzionalne baze podataka kojim se prevladavaju sva ograničenja zbog relacijskog modela skladišta podataka (Essbase iz Oracle-a).

- **ETL alati**: alati za pronalaženje, ekstrahiranje, transformiranje i učitavanje podataka u objedinjenom formatu u skladište podataka. Više o ovim procesima možete pronaći u drugom poglavlju ovog rada.

- **Metapodaci**: podaci o podacima koji definiraju skladište podataka, a koji se koriste za kreiranje, održavanje i upravljanje skladištem podataka. Njihova uloga u skladištu je da određuju izvor, upotrebu, vrijednost i značajke podataka skladišta, a osim toga definiraju i kako se podaci mogu obrađivati i mijenjati.
  Redak u prodajnoj bazi može izgledati ovako: 6050 KJ591 643,98.
  Ovaj podatak je za korisnika besmislen dok ne uključimo metapodatke koji su ključni u tranformaciji podataka u znanje. Uz pomoć metapodataka imamo sljedeće informacije: Broj modela: 6050, Oznaka agenta prodaje: KJ591, Ukupan iznos: 643,98 kn.
  Metapodaci se dijele na dvije kategorije:
  - Tehnički metapodaci koji sadrže informacije o skladištu koji su potrebni administratorima skladišta.
  - Poslovni metapodaci koji sadrže pojedinosti pomoću kojih krajnji korisnici lako razumiju pohranjene podatke u skladištu.
- **Alati za pristup krajnjeg korisnika**: omogućuju korisnicima interakciju sa sustavom skladišta podataka. Ovi alati kategorizirani su u četiri skupine, a to su:
  - *Alati za upite i izvješćivanje*: omogućuje korisnicima generiranje redovitih operativnih izvješća, podržava velike količine poslova kao što su izračunavanje i ispisi, pomaže u rješavanju problema u bazi podataka umetanjem meta sloja između korisnika i baze podataka.
  - *Alati za razvoj aplikacije*: služe za kreiranje prilagođenih izvješća koji su grafički i analitički zahtjevniji od alata koje podržava skladište podataka.
  - *OLAP alati*: temelje se na konceptima višedimenzionalne baze podataka, a omogućuje korisnicima analizu koristeći razrađene i složene višedimenzionalne prikaze.
  - *Alati za rudarenje podataka*: služe kako bi automatizirali procese kao što su otkrivanje smislene nove korelacije, obrazaca i trendova velike količine podataka.



Slika 2: Ilustrativni prikaz arhitekture skladišta podataka
Izvor:[10]

Dizajn i arhitekturu skladišta podataka ne treba shvatiti olako zbog njegove vrijednosti za organizaciju. Dizajn je potrebno dobro isplanirati i analizirati, jer nakon što se ono počne puniti podacima, modificiranje njegove strukture vrlo je rizično i potencijalno skupo.

Osim arhitekture skladišta podataka, važno je istaknuti i neke ključne karakteristike ovog okruženja [10] :

- *Predmetno orijentiran* - pruža informacije o zadanoj temi, a ne o ukupnom poslovnom procesu. Ako želite analizirati podatke o prodaji tvrtke potrebno je izgraditi skladište koje se koncentrira na prodaju. Tako dizajnirano skladište može odgovoriti na pitanja i dati predikcije koje će biti korisne za unaprjeđenje prodaje.

- *Integriran* - podaci se pohranjuju u skladište integracijom iz različitih izvora u konzistentnom i univerzalno prihvatljivom formatu, u smislu naziva, formata i kodiranja.

- *Postojan i nepromjenjiv* - svi podaci u skladištu su samo za čitanje i ne mogu biti obrisani niti promijenjeni.

- *Vremenski varijantan* - svi podaci u skladištu dokumentirani su s komponentom vremena, eksplicitno ili implicitno, npr. primarni ključ koji mora imati elemente vremena.

Nakon što smo pobliže objasnili karakteristike skladišta podataka lakše je podvući razlikovnu crtu između skladišta podataka i tradicionalne baze podataka, kao koncepta. Iako se oba pojma idejno čine dosta slična, glavna razlika je u tome što se u bazi podataka podaci prikupljaju za višestruke transakcijske svrhe, dok se u skladištu podaci u velikoj mjeri prikupljaju za izvođenje analitike. Još jedna od razlika je što baze podataka pružaju podatke u stvarnom vremenu dok skladišta pohranjuju podatke kojima se može pristupiti i za velike analitičke upite. U sljedećem poglavlju bit će dane još neke od komponenti koje razlikuju klasične baze podataka od skladišta podataka.

1.3 Činjenice i dimenzije

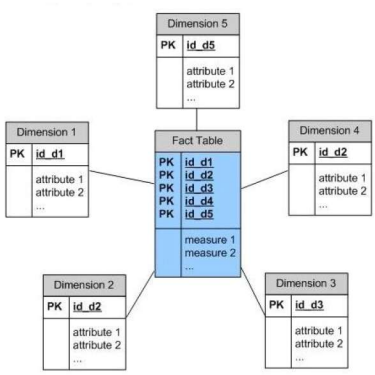
Tehnički gledano, skladište podataka je baza podataka, pa se u skladu s tim sastoji od tablica, polja, ključeva i relacija. Međutim, postoje neke specifičnosti koje razlikuju tablice skladišta podataka od tablica u klasičnoj bazi koja se koristi za obradu transakcija. Tablice skladišta podataka dijele se u dvije kategorije:

- tablice činjenica (*engl. fact tables*)

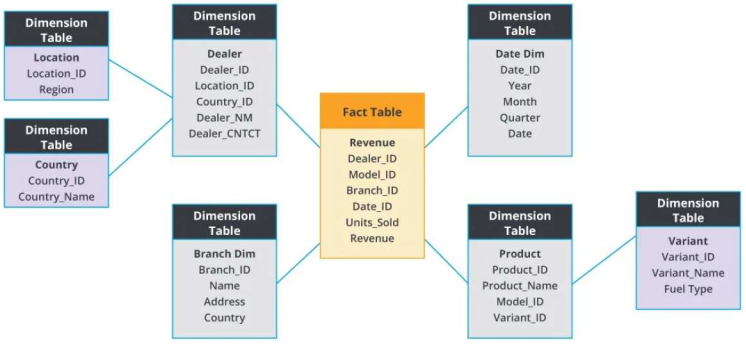
- tablice dimenzija (*engl. dimension tables*)

Osim toga, tablice činjenica i dimenzija u skladištu podataka moraju imati izgled koji odgovara jednoj od dvije glavne topologije: **zvjezdana shema** [1] i **snježna shema** [2]. U zvjezdanoj topologiji, dimenzije okružuju jednu tablicu činjenica, dok snježna shema ima hijerarhiju dimenzija. Primjere hijerahije tablica ove dvije sheme možete vidjeti na sljedećim ilustracijama.

[1] *engl. star schema*
[2] *engl. snowflake schema*



(a) Zvjezdana topologija
Izvor: `https://educatech.in/explain-star-schema/`, 10.9.2022



(b) Snježna topologija
Izvor: `https://streamsets.com/blog/schemas-data-warehouses-star-galaxy-snowflake/`, 10.9.2022.

Slika 3: Shematski prikaz dvije glavne topologije

Kako bi dizajn skladišta podataka bio besprijekoran i točan važno je razlikovati čenjenice od dimenzija. Iz tog razloga, oba pojma će biti detaljnije obrađena u nastavku.

1.3.1 Tablice dimenzija

Dimenzije su zbirke referentnih informacija o činjenicama. One pobliže opisuju i kategoriziraju činjenice u skladištu kako bi omogućile smislene, detaljne i opisne odgovore na postavljene poslovne upite. Pri kreiranju skladišta podataka praksa je da se prvo kreiraju tablice dimenzija, pa potom tablice činjenica koje povezujemo s dimenzijama pomoću stranih ključeva.
Tablica dimenzija sastoji se od primarnog ključa i skupa atributa. Ove tablice uglavnom sadrže velik broj atributa koji su pretežito tekstualni ili vremenski. Karakteristika tablice dimenzija je da ima nisku kardinalnost i maleni broj redaka, pa nemaju velik utjecaj na performanse [4].
Dimenzije skladišta podataka dijele se u nekoliko tipova, koji su definirani na osnovu ponašanja i upotrebe. Postoji velik broj tipova, u nastavku su navedeni i objašnjeni samo neki od njih dok ostatak možete pronaći u [5] i [13].

- **Usklađene dimenzije** (*engl. Conformed Dimensions*) - su dimenzije koje se mogu povezati s različitim tablicama činjenica pri čemu uvijek zadržavaju isto značenje. Klasični primjer usklađene dimenzije je datum, zato što se njegovo značenje ne mijenja u ovisnosti o tablici činjenica.

- **Dimenzije igranja uloga** (*engl. Role-playing dimensions*) - su dimenzije koje se također povezuje s različitim tablicama činjenica, ali za razliku od usklađenih dimenzija, one mijenjaju značenja u ovisnosti o tablici činjenica ili polju s kojim se povezuju. Primjer ovog tipa dimezije ponovno može biti datum, npr. u tablici činjenica o prodaji možemo imati datum narudžbe, datum plaćanja i datum isporuke, i sve ih povezujemo s istom dimenzijom datuma.

- **Sporo mijenjajuće dimenzije** (*engl. Slowly changing dimensions*) - su dimenzije koje su podložne promjenama. Za razliku od događaja, koji se dogode jednom i ostavljaju nepromjenjiv zapis da su se dogodili, dimenzije se mogu mijenjati.

Primjer ove dimenzije mogu biti neki podaci o klijentu, nazivi, lokacije i sl.
Prilikom korištenja ove vrste dimenzije važno je unaprijed defnirati strategiju ažuriranja.

Neke od mogućih strategija ažuriranja su:
- nema ažuriranja - primjenjuje se na nepromjenjive dimenzije npr. datum,
- atributi se uvijek ažuriraju na najnoviju vrijednost,
- promjene na dimenziju primjenjuju se generiranjem novih verzija promijenjenih elemenata, npr. korištenje tablica povijesti ili dodavanjem polja za razlikovanje novih podataka od izvornih podataka.

- **Bezvrijedne dimenzije** (*engl. Junk dimensions*) - činjenice često imaju atribute indikatora kao što su *Bool* vrijednosti ili neki drugi skup niskog kardinaliteta. Kako bi izbjegli stvaranje malih dimenzija za svaki ovaj atribut, stvaraju se bezvrijedne dimenzije koje skupljaju sve te atribute u jednu tablicu. U ovaj tip dimenzije skupljamo sve moguće kombinacije atributa koje obuhvaća, te svakoj kombinaciji dodijelimo jedan primarni ključ. Na taj način optimiziramo veličinu tablice činjenica zato što je dovoljno u nju uključiti jedan strani ključ bezvrijedne dimenzije umjesto niza vrijednosti svakog atributa.

- **Degenerirane dimenzije** (*engl. Degenerate dimensions*) - ove dimenzije su izvedene iz tablice činjenica i nemaju vlastitu dimenziju. Brojevi zaglavlja transakcija koje dodjeljuje operativni poslovni proces obično su degenerirane dimenzije, npr. brojevi narudžbe, ulaznice ili transakcije kreditne kartice. Degenerirane dimenzije mogu biti vrlo korisne za grupiranje povezanih redaka tablice činjenica.

1.3.2 Tablice činjenica

Činjenice su mjerljivi događaji koji se odnose na funkcionalno područje koje pokriva skladište podataka (npr. prodaja je činjenica u skladištu podataka za područje prodaje). Tablice činjenica su temeljne tablice u skladištu podataka koje se sastoje od primarnog ključa i skupa atributa.
Razlikujemo dvije vrste atributa: kvalitativne i kvantitativne. Kvalitativni atributi se uglavnom definiraju kao strani ključ za dimenzijsku tablicu i predstavljaju karakteristike činjenice. Kvantitativni atributi predstavljaju mjeru činjenice, kao što je iznos, količina, vremensko razdoblje ili neku drugu mjerljivu vrijednost. One služe za grupiranje, agregiranje, i druge statističke izračune.
I tablice činjenica se mogu klasificirali u tri kategorije, a to su:

- **Tablice transakcijskih činjenica** - u ovoj tablici svaki redak predstavlja stavke u transakciji, npr. informacije o jednoj stavci fakture. Cilj je zabilježiti što više detalja, pa ovaj tip uključuje velik broj dimenzija.

- **Periodične bilješke** - su tablice činjenica koje vode evidenciju o zapisima događaja za zadana vremenska razdoblja. Primjer ovog tipa je bilježenje ukupne prodaje po danu zbrajanjem podataka iz tablice transakcijskih činjenica.

- **Kumulativne bilješke** - koriste se za označavanje prekretnica poslovnog procesa pri čemu se bilježe datumi u nekoliko stupaca. Svaki redak u ovoj tablici sumira sve događaje koji su se dogodili od početka do kraja procesa. Ova vrsta tablica primjenu pronalazi u skladištima podataka koja prikupljaju i obrađuju zahtjeve i narudžbe.

2 ETL

ETL (*engl. Extract, Transform, Load*) je proces koji ekstrahira podatke iz različitih izvora, zatim transformira podatke (primjenom izračuna, ulančavanja, čišćenja i sl.) i na kraju učitava podatke u centralnu bazu skladišta podataka (*engl. Data Warehouse*). Kako bi ETL proces bio potpun potrebno je da sadrži sva tri procesa: ekstrahiranje, transformaciju i učitavanje podataka [3].

Ovakav proces uglavnom se izvodi pomoću softverskih aplikacija, ali operateri sustava to mogu napraviti i ručno. ETL softver obično automatizira cijeli proces te je to ponavljajuća aktivnost (dnevna, tjedna, mjesečna, intervalna) i mora biti agilna, automatizirana i dobro dokumentirana.

Popularizacija ovog koncepta dogodila se 1970-ih godina, kada je njegova primjena zaživila i u praksi. ETL sustavi obično integriraju podatke iz više aplikacija (sustava), koje razvijaju i održavaju različiti partneri ili se nalaze na zasebnom računalnom hardveru. Na primjer, sustav troškovnog računovodstva može kombinirati podatke iz obračuna plaća, prodaje i nabave.

Jedna od glavnih svrha ovog procesa je optimizacija podatka za analitiku. ETL rješava dva temeljna problema koji omogućuju bolju analitiku:

- Kao proizvod ETL procesa nastaje okruženje optimizirano za analizu podataka: Transakcijske baze podataka kao što su MySQL i Postgre SQL izvrsne su u obradi transakcijskih opterećenja. Izvrsni su u čitanju i ažuriranju pojedinačnih redaka podataka s malom latencijom. Dok s druge strane, nisu baš pogodne za provođenje velike analitike na ogromnim skupovima podataka i zato su nam potrebna skladišta podataka.

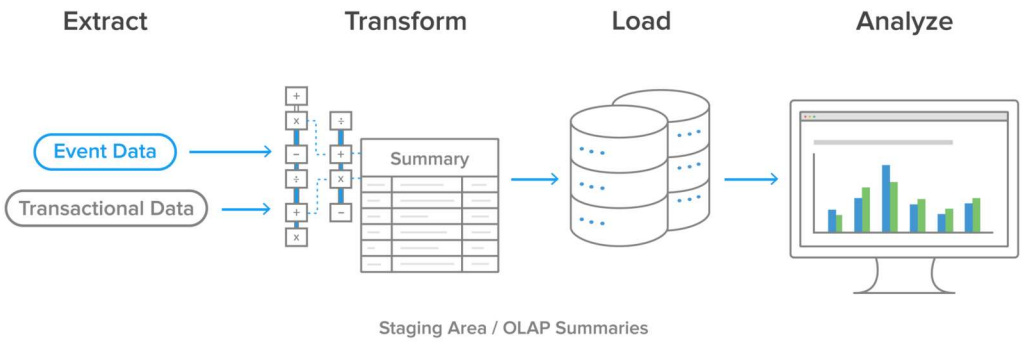
- Analiza među domenama: spajanjem podataka iz različitih izvora, poslovni stručnjaci mogu odgovoriti na dublje poslovne probleme. Ovaj zahtjev postaje sve važniji kako poduzeća postaju složenija, rade bržim tempom i postavljaju sustave u oblak.

2.1 Tradicionalni vs. moderni ETL

Povijesni koncept ETL sustava podrazumijeva da se podaci izdvajaju iz transakcijske baze podataka (služe za online obradu transakcija - OLTP [3]) i drugih izvora podataka. Iako OLTP aplikacije imaju visoku propusnost, s velikim brojem zahtjeva za čitanje i pisanje, nisu pogodne za analizu podataka i zadatke poslovne inteligencije. Na podacima se potom, u pripremnoj fazi, izvršava proces transformacije. Transformacije pokrivaju čišćenje i optimizaciju podataka za analizu. Naposljetku se transformirani podaci učitavaju u analitičku bazu podataka - OLAP.

Timovi za poslovnu inteligenciju pokreću upite na podacima u analitičkoj bazi, a rezultate prezentiraju krajnjim korisnicima ili pojedincima odgovornim za donošenje poslovnih odluka. Rezultati se također koriste kao ulaz za algoritme strojnog učenja ili druge projekte u podatkovnoj znanosti. Ako OLAP sažetci ne mogu podržati vrstu analize koju tim za poslovnu inteligenciju želi napraviti, cijeli se proces mora pokrenuti ponovno, ali s drugačijim transformacijama. To predstavlja jedan od nedostataka tradicionalnog ETL procesa.

[3] *engl. Online transaction processing*

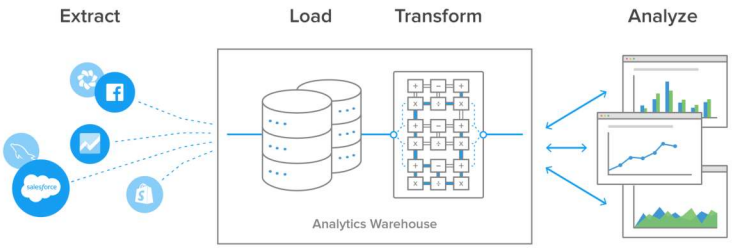


Slika 4: Shematski prikaz povijesnog koncepta ETL sustava
Izvor: `https://www.stitchdata.com/etldatabase/etl-process/`, 15.9.2022.

Nagli skok u napretku tehnologije promijenio je pristup većine organizacija ETL-u iz nekoliko razloga:

- Prvi i najveći razlog je pojava moćnih analitičkih skladišta podataka kao što su Amazon Redshift i Google BigQuery. Nove analitičke baze podataka temeljene na oblaku imaju dovoljno snage za izvođenje transformacija na mjestu umjesto da zahtijevaju posebno područje prikazivanja kao što je to slučaj kod OLAP sažetaka u tradicionalnom pristupu.
- Drugi razlog je brzi prijelaz na SaaS [4] aplikacije koje se temelje na oblaku, a koje sadrže značajne količine podataka ključne za poslovanje u vlastitim bazama podataka. Takvim podacima je moguće pristupiti pomoću API-ja i webhook-ova [5].
- Podaci se često analiziraju u neobrađenom obliku za razliku od unaprijed pripremljenih OLAP sažetaka
- Kako bi zadržao svoju vrijednost kao alata za donositelje odluka, ETL proces se treba mijenjati i prilagođavati poslovnim promjenama.

Navedeni razlozi su doveli do razvoja laganih, fleksibilnih i transparentnih ETL sustava s procesima koji izgledaju otprilike ovako:



Slika 5: Shematski prikaz modernog koncepta ETL sustava
Izvor: `https://www.stitchdata.com/etldatabase/etl-process/`, 15.9.2022.

[4]Softver kao usluga *engl. Software as a service* - način isporuke softvera i licenciranja u kojem se softveru pristupa online putem pretplate, umjesto da se kupuje i instalira na pojedinačna računala

[5]Webhook je funkcija povratnog poziva temeljena na HTTP-u koja omogućuje laganu komunikaciju vođenu događajima između dva sučelja za programiranje aplikacija. Primjer ove metode: automatski primajte e-poštu svakog jutra o svom prvom sastanku u slučaju da zaboravite provjeriti svoj kalendar

Kako bi bolje razumijeli ETL sustav u nastavku ćemo detaljnije objasniti svaki od procesa koji je sastavni dio ETL-a: ekstrahiranje, transformacija i učitavanje podataka.

2.2 Ekstrahiranje podataka

Proces ekstrahiranja podataka prvi je korak u ETL sustavu, a on započinje na način da se identificiraju izvori podataka i informacije koje su nam iz tih izvora potrebni. Ovaj korak se nerijetko smatra i najvažnijim korakom u ETL-sustavu zato što uspjeh i kvaliteta ekstahiranih podataka uvelike određuju uspješnost procesa koji slijede nakon njega. Kao izvor podataka najčešće se uzimaju transakcijske baze podataka koje mogu bili smještene na serveru ili u oblaku, različite host poslovne aplikacije, ERP sustavi, tekstualne datoteke, proračunske tablice i sl. Navedeni izvori uglavnom koriste različitu organizaciju podataka kao i različit format. Neki od formata koje često susrećemo su: relacijske baze, XML, JSON, flat datoteke, nerelacijske strukture podataka poput sustava za upravljanje informacijama (IMS), metoda pristupa virtualnoj pohrani (VSAM) i druge. Osim toga, ovaj proces obuhvaća proces planiranja i procjene za veličinu baze, procese validacije podataka, određivanje prihvatljive latencije i ostalih parametara, ovisno o zahtjevima i složenosti poslovnih procesa. Nakon što su izvori i parametri poznati, podaci se iz izvora ekstrahiraju na jedan od tri načina spomenuta u [11]:

- **Djelomično ekstrahiranje - s obavijesti o ažuriranju**: ovo je najlakši način za ekstrahiranje podataka zato što izvorni sustav ima mogućnost da vas obavijesti kada je neki od zapisa promijenjen. To omogućava da se ažuriraju nastale promjene bez da se svaki puta ekstrahiraju svi podaci. Većina baza podataka pruža ovu mogućnost, dok SaaS aplikacije podržavaju sličnu funkcionalnost pomoću web-dojavnika (*webhooks*).

- **Inkrementalno ekstrahiranje - bez obavijesti o ažuriranju**: Neki izvori nemaju mogućnosti obaviještavanja o ažuriranju, ali mogu identificirati koji su zapisi izmjenjeni i pružiti ih u obliku izvatka promjenjenih zapisa. Nedostatak ovog načina je što ponekad neće detektirati izbrisane zapise u izvornim podacima.

- **Potpuno ekstrahiranje**: Postoje sustavi koji nemaju niti jednu od prethodno navedenih mogućnosti o praćenju ažuriranja zapisa. U takvim izvorima potrebno je ponovno učitati sve podatke. Osim toga, potrebno je zadržati kopiju posljednjeg učitavanja podataka kako bi mogli usporediti koji su zapisi novi ili promijenjeni. Ovaj način ekstahiranja preporučuje se samo ako je neophodan ili ako je riječ o malim količinama podataka.

Bez obzira koju od navedenih metoda za ekstrakciju izabrali, ona ne bi trebala utjecati na performanse i vrijeme odziva izvornih sustava. Gotovo svaki izvorni sustav je živa baza podataka, pa bi trebali voditi računa da operacije koje izvršavamo na tim bazama (zaključavanje zapisa, usporavanje) ne ostavljaju posljedice na performanse izvornog sustava.

Važno je napomenuti da ekstrahiranje uključuje i proces validacije podataka kako bi se potvrdilo imaju li ekstrahirani podaci očekivane vrijednosti, odgovarajućeg tipa u danoj domeni. Ako podaci ne zadovolje pravila validacije, odbijaju se djelomično ili u cijelosti.

Kao što je već prethodno opisano, tradicionalni koncept ETL-a značajno se razlikuje od modernog i to najviše u prvoj fazi, odnosno ekstrahiranju podataka. Najveća razlika leži u izvorima podataka: pa za razliku od tradicionalnog ETL-a koji uglavnom ekstrahira podatke iz transakcijskih baza podataka, moderni ETL sustavi se uglavnom oslanjaju na mnoštvo

SaaS alata o kojim će biti više riječ u nastavku. Proces ekstrahiranja podataka za SaaS proizvod oslanja se na integraciju s njegovim API-ijima. Budući su API-iji dosta podložni promjenama i ovise isključivo o tvrtki koja ih plasira za upotrebu to donosi nove izazove u proces ekstahiranja podataka. Neki od ključnih izazova su:

- Integracija se razlikuje za svaku aplikaciju, a stvari dodatno otežava ako aplikacije ne koriste REST API-ije nego protokole kao što je SOAP.
- API-iji nisu precizno dokumentirani i uglavnom su prilagođeni programerima.
- Podložni su čestim promjenama što zahtijeva značajnu količinu inženjerskog vremena za rješavanje problema s promjenama API-ja.
- Mogu biti jako zamršeni i složeni ako pokrivaju veliko poslovno područje koje može imati mnoštvo ugrađenih krajnjih sredstava.

2.3 Transformacija podataka

Ekstahirani podaci dobiveni iz nekog izvornog sustava su neobrađeni (sirovi) i uglavnom ih se ne može koristiti u izvornom obliku. Takve podatke potrebno je očistiti, mapirati i transformirati. Upravo taj koncept primjene skupa pravila i funkcija nad podacima predstavlja drugi važan proces u ETL sustavu. Ovaj proces je ključan, zato što dodaje i prilagođava podatke tako da bi se mogla odraditi analiza i generirati točna i precizna BI [6] izvješća. Ako su podaci u izvornom sustavu dovoljno dobri da ne zahtijevaju nikakvu obradu, proces se onda naziva *dikretno premještanje* ili *prolaz kroz podatke* [11].
Postoje dva pristupa ETL tansformaciji:

- **Višestupanjska transformacija podataka**: Ekstrahirani podaci premještaju se u pripremno područje (*staging area*) gdje se nad njima, prije učitavanja u skladište podataka, izvršavaju transformacije. Ovo je pristup karakterističan za tradicionalne ETL sustave.
- **Transformacija podataka u skladištu**: u ovom pristupu ekstrahirani podaci se direktno učitavaju u analitičku bazu te se transformacije nad njima izvršavaju tamo. Ovo je odlika modernijih ETL sustava koji u sve većoj mjeri nalikuju ELT [7] sustavima.

Posljednih godina, kako se organizacije sve više okreću prema SaaS aplikacijama i modernom ETL pristupu, došlo je i do pomaka prema transformaciji padataka unutar analitičke baze, odnosno skladišta podataka. Puno bolje performanse moderne analitičke baze podataka, kao i mogućnost da se transformacije na bazi pišu u SQL jeziku dovele su do prevlasti pristupa *transformacija podataka u skladištu.*

Bez obzira gdje se proces transformacije odvija, njegova važnost za pripremu podataka za analizu je jednaka. U nastavku navodimo neke od osnovnih vrsta transformacija koje se provode nad podacima [3]:

- **Čišćenje**: npr. mapiranje NULL vrijednosti u defaultnu vrijednost, formatiranje datuma, promjena oznaka kategorizacije (npr. „Žensko” u „Ž”) i sl.

[6] Poslovno obavještavanje (*engl. Business intelligence*)

[7] ELT je proces koji prvo učita podatke u analitičku bazu podataka, pa tek u njoj izvršava transformacije nad podacima. ELT se obično koristi s bazama podataka bez SQL, podatkovnim uređajima ili instalacijom u oblaku.

- **Deduplikacija**: detekcija i uklanjanje dvostrukih zapisa.
- **Provjera formata**: pretvorba mjernih jedinica, promjena formata vremena, datuma, vremenskih zona i sl.
- **Restrukturiranje ključeva**: uspostavljanje odnosa među tablicama (primarni, strani ključ).
- **Validacija obaveznih polja**: polja označena kao obavezna moraju imati vrijednosti.

Osim osnovnih transformacija, kojima uglavnom manipuliramo formatiranjem podataka i uklanjamo viškove iz baze, postoje i napredne transformacije koje su fokusiranje više na logiku i izačune. Neke od njih su:

- **Derivacija**: primjena poslovnih pravila na postojeće podatke (npr. stvaranje metrike prihoda koja iz iznosa oduzima porez).
- **Filtriranje**: odabir samo određenih redaka/stupaca.
- **Pridruživanje**: povezivanje podataka iz više izvora (npr. podaci o kvaliteti zraka povezani s podacima o razini peludi zraku).
- **Dijeljenje**: podjela jednog stupca u više stupaca (kategorizacijski stupac podijeli na više boolean stupaca - pripada navedenoj kategoriji ili ne).
- **Provjera valjanosti podataka**: provjera zadovoljava li podataka tip podatka, pripada li očekivanoj domeni i sl.
- **Sažimanje**: vrijednosti se sažimaju kako bi se dobile ukupne brojke koje se izračunavaju i pohranjuju kao poslovna metrika (npr. zbrajanje kupnji koje je izvršio kupac kao metrika dugotrajne vrijednosti kupca).
- **Integracija**: svakom jedinstvenom podatkovnom elementu dodijeli se jedno ime s jednom standarnom definicijom. Ova transformacija služi za usklađivanje različitih naziva i vrijednosti za isti podatak.
- **Agregacija**: podaci se prikupljaju iz više izvora.

2.4 Učitavanje podataka

Posljedni korak u ETL sustavu je učitavanje podataka u bazu skladišta podataka. Proces učitavanja podataka u skladište ovisi o tome koja je njihova svrha nakon što se učitaju. Neke od upotreba podataka u skladištu mogu biti:

- korištenje alata za poslovnu inteligenciju (BI) ili analitiku,
- izrada alata za pretraživanje web stranica,
- korištenje algoritama za strojno učenje,
- izrada sustava za donošenje poslovnih odluka.

Neovisno o krajnjem cilju važno je razumijeti zahtjeve i ciljeve koje očekujemo od cjelokupnog procesa u danom okruženju. Parametri koji su promijenjivi u procesu učitavanja su: volumen podataka, učestalost učitavanja, struktura, cilj i vrsta učitavanja.

Pod pretpostavkom da učitavamo podatke u analitičko skladište razlikujemo tri vrste učitavanje podataka:

- **Početno učitavanje**: popunjavanje svih tablica skladišta podataka.
- **Inkrementalno učitavanje**: povremeno učitavanje najnovijih promjena u bazu. Datumi posljednjeg učitavanja podataka u bazu pohranjuju se na način da se u sljedećoj iteraciji učitaju samo zapisi nakon tog datuma. Ovaj tip učitavanja dijeli se u ovisnosti o količini podataka koja se učitava. Pa tako razlikujemo:
  - *streaming inkrementalno učitavanje*: učitavanje malih količina podataka,
  - *skupno inkrementalno učitavanje*: učitavanje velikih količina podataka.
- **Potpuno osvježenje**: Brisanje zapisa jedne ili više tablica i ponovno učitavanje sa najnovijim podacima.

Inicijalno učitavanje podataka u sve tablice skladišta podataka nije složen poces, ali je vremeski zahtjevniji od inkrementalnog učitavanja podataka. Iako je vremenska složenost prednost inkrementalog učitavanja podataka, to je poprilično složen proces koji ima nekoliko ključnih kritičnih područja [1]:

- **Redoslijed**: podatkovni cjevovodi često su distribuirani sustavi kako bih održali dobre performanse. Međutim, to znači da se podaci mogu obrađivati drugačijim redoslijedom od onoga kojim su primljeni, što može predstavljati ozbiljan problem ako se podaci ažuriraju ili brišu. Kako bi se ovakvi problemi izbjegli, važno je voditi računa o redoslijedu učitavanja podataka.
- **Evolucija sheme**: dodavanje novog svojstva ili promjena postojećeg mogu dovesti do desktruktivnih promjena na bazi. Zato je potrebno sinkronizirati i evoluirati shemu baze pri svakoj većoj promjeni, kako podaci u njoj ne bi postali nedosljedni.
- **Mogućnost praćenja**: kada imamo sustave gdje podaci dolaze iz velikog broja izvora, kvarovi i problemi su uglavnom neizbježni. Većina ovih problema nastaje zbog neodržavanja API-ja, njegovih pristupnica, ili problema s vezom.

Ako dođe do nepredviđenog ponašanja u pocesu učitavanja ili se on iz nekog razloga prijevremeno zaustavi, potrebno je imati mehanizme za opravak koji bi trebali ponovo pokrenuti učitavanje od trenutka kvara, kako ne bi izgubili integritet podataka. Iako je ovo relativno samostalan i automatiziran proces, nadzor administratora skadišta podataka je od velike važnosti.

3 Praktični zadatak

3.1 Opis i motivacija

Kao sastavni dio ovog diplomskog rada dizajniran je i implementiran prilagođeni ETL cjevovod. Glavna svrha tog alata je prikupljanje podataka o vremenskim uvjetima te podataka o parametrima važnim za kvalitetu zraka. Kao rezultat ovog praktičnog dijela nastalo je skladište podataka koje sadrži povijesne i aktualne podatke koji će služiti kao polazna podatkovna točka za treniranje i testiranje algoritama za strojno učenje čiji je cilj provjeriti ovisnosti vremenskih prilika i parametara kvalitete zraka s razinom peludi u zraku. Ideja koja stoji iza ovog rada je: prikupiti nekoliko kvalitetnih izvora podataka koji nude uvid u vremenske prilike i parametre važne za kvalitetu zraka, zatim ih transformirati i objediniti u uniformni format i naposljetku učitati u bazu podataka gdje se mogu koristiti kao cjelovita informacija. Podatke o razini peludi u atmosferi trenutno nemamo dostupne u bazi iz razloga što je cilj bio lokalna analiza upotrebom strojeva koji mjere koncetraciju peludnih čestica u zraku za neposrednu okolinu. Međutim, budući se razina peludnih čestica u zraku uglavnom dobiva pomoću nekih mjernih strojeva koji nisu izravan izvor podataka nego se oni obrađuju i budu pohranjeni u obliku različitih datoteka, ovo rješenje je otvoreno za dodavanje novih izvora podataka i to u bilo kojem obliku koje Python podržava.
U nastavku ovog poglavlja opisat ćemo cijeli proces kreiranja ovog cjevovoda podataka, dati uvid u izvorni kod napisan u Pythonu te dati pregled podataka koji se nalaze u odredišnoj bazi podataka. Cijeli kod ovog projekta može se pronaći u Git repozitoriju koji je dostupan u: [2].

3.2 Izvori podataka i ekstrahiranje

Prvi korak u dizajniranju ETL alata je izbor odgovarajućih izvora podataka. U klasičnim primjerima korištenja ETL alata unutar neke organizacije ili tvrtke, kao izvori podataka uglavnom se uzimaju neke lokalne baze, spremišta podataka različitih aplikacija ili podaci iz oblaka. Međutim, u situaciji kao što je ova, kada želimo napraviti analizu podataka koji nisu isključivo interne prirode i koje direktno ne posjedujemo, onda je najbolje kao izvor podataka koristiti dostupne API-ije. Što je zapravo API? API je akronim za aplikacijsko programsko sučelje, a predstavlja softverskog posrednika koji omogućuje da dvije aplikacije međusobno komuniciraju.

Budući da je tema vremenske prognoze kojom se ovdje bavimo, vrlo česta u primjeni, postoji mnoštvo dostupnih API-ija. Većina njih zahtijeva korištenje mjesečne pretplate, no postoje i oni koji su besplatni, ali s ograničenim pristupom podacima. U ovom radu smo se bazirali na besplatno dostupne API-ije, a one koje smo koristili su: *OpenWeatherMap API* 2.5, *WeatherApi*, *One Call API* 3.0, *Air Pollution API* i *Air Quality API*.

Kako je prethodno već navedeno, prikupljaju se povijesni i trenutni podaci o vremenskim prilikama. API koji nam je omogućio odgovarajući pristup povijesnim podacima je *One Call API* 3.0 koji je dio OpenWeather platforme, koja glasi za jednu od najkvalitetnijih i najpouzdanijih izvora kada je riječ o vremenskim uvijetima i sličnim podacima.
One Call API pruža sljedeće vremenske podatke za bilo koje geografske koordinate:

- minutna prognoza za 1 sat,
- satna prognoza za 48 sati,

- dnevna prognoza za 8 dana,
- nacionalna vremenska upozorenja,
- povijesne vremenske podatke za 40+ godina unatrag (od 1. siječnja 1979.).

S obzirom na to da ovaj API ima ograničen broj poziva na 1000 ako se koristi u besplatnoj verziji, odlučili smo koristiti ga samo za dohvaćanje povijesnih podataka. Prilikom kreiranja API zahtjeva potrebno je proslijediti određene parametre:

- *lat, lon* - geografske kordinate (širina i dužina) mjesta za koje želimo dobiti podatke, obavezan parametar,
- *dt* - točan datum i vrijeme za koje želimo dobiti podatke u unix formatu, obavezan parametar,
- *appid* - jedinstveni API ključ koji se dobije kao dio pristupnih podataka API-iju, obavezan parametar,
- *units* - jedinice mjere podataka u odgovoru, postoje tri opcije: standard, metric, imperial, opcionalni parametar,
- *lang* - omogućava dohvaćanje odgovora na odabranom jeziku, opcionalni parametar.

Primjer funkcije napisane u Pythonu koja šalje zahtjev za podacima API-iju priložen je u nastavku.

```
import pandas as pd
import requests
import config

def CallOpenWatherApiCall(city, date):
    response_list = []
    API_KEY = config.api_key
    url = config.OpenWatherHistoryApiUrl
    lon, lat = FindTimezoneName(city)
    query_string = {"lat" : lat, "lon": lon, "dt": date, "appid": API_KEY,
    "units": 'metric'}
    r = requests.get(url, params=query_string)
    response_list.append(r.json())

     #city data
    response_list[0]['city'] = GetCityData(city)
    df = pd.DataFrame.from_dict(response_list)
    return df
```

Program 1: Primjer funkcije koja poziva One Call API s parametrima

Ako su svi proslijeđeni parametri ispravni i API zahtjev se uspješno izvrši, te kao rezultat poziva dobivamo podatke u *json* formatu. Jedan takav primjer API odgovora možete vidjeti na Slici 6a. Podatke u *json* formatu potom formatiramo u Python *dataframe* nad kojim možemo jednostavno provoditi operacije transformacije. Objašnjenje svakog pojednog parametra u *json* odgovoru možete pronaći u službenoj dokumentaciji *One Call API-ija* [9].

```
{
"lat": 52.2297,
"lon": 21.0122,
"timezone": "Europe/Warsaw",
"timezone_offset": 3600,
"data": [
  {
    "dt": 1645888976,
    "sunrise": 1645853361,
    "sunset": 1645891727,
    "temp": 279.13,
    "feels_like": 276.44,
    "pressure": 1029,
    "humidity": 64,
    "dew_point": 272.88,
    "uvi": 0.06,
    "clouds": 0,
    "visibility": 10000,
    "wind_speed": 3.6,
    "wind_deg": 340,
    "weather": [
      {
        "id": 800,
        "main": "Clear",
        "description": "clear sky",
        "icon": "01d"
      }
    ]
  }
]
}
```


(a) Povijesni podaci
Izvor: [9], 19.09.2022


```
{
  "coord": {
    "lon": 10.99,
    "lat": 44.34
  },
  "weather": [
    {
      "id": 501,
      "main": "Rain",
      "description": "moderate rain",
      "icon": "10d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 298.48,
    "feels_like": 298.74,
    "temp_min": 297.56,
    "temp_max": 300.05,
    "pressure": 1015,
    "humidity": 64,
    "sea_level": 1015,
    "grnd_level": 933
  },
  "visibility": 10000,
  "wind": {
    "speed": 0.62,
    "deg": 349,
    "gust": 1.18
  },
  "rain": {
    "1h": 3.16
  },
  "clouds": {
    "all": 100
  },
  "dt": 1661870592,
  "sys": {
    "type": 2,
    "id": 2075663,
    "country": "IT",
    "sunrise": 1661834187,
    "sunset": 1661882248
  },
  "timezone": 7200,
  "id": 3163858,
  "name": "Zocca",
  "cod": 200
}
```


(b) Trenutni podaci
Izvor: [8], 19.09.2022

Slika 6: Primjer API odgovora u json formatu

Drugi važan API, koji nam omogućava trenutne podatke o vremenskim prilikama je *OpenWeatherMap API* 2.5. Pomoću njega možemo pristupiti trenutnim vremenskim podacima za bilo koju lokaciju na zemlji uključujući više od 200000 gradova. Podaci kojim pristupamo stižu iz različitih izvora kao što su globalni i lokalni vremenski modeli, sateliti, radari i mreža meteoroloških stanica.
Budući da oba opisana API-ija dolaze s iste plaforme *OpenWather*, parametri prilikom kreiranja zahtjeva su gotovo identični. Jedina razlika je parametar *mode* koji je prisutan u ovom API-iju, a koji određuje u kojem formatu želimo da bude dobiveni odgovor. Dostupni formati su JSON, XML i HTML. Više detalja o parametrima i odgovoru API-ija može se pronaći u službenoj dokumentaciji [8].
Primjer API odgovora za trenutno vrijeme u *json* formatu možete vidjeli na slici 6b. Već pogledom na dvije usporedne slike API odgovora može se zaključiti da je odgovor opširniji kada je riječ o trenutnim podaci. Tu svakako igra ulogu činjenica da se povijesni podaci dugoročno čuvaju u bazi i zauzimaju određenu količinu memorije. Osim toga važno je istaknuti

da API koji vraća povijesne podatke može vratiti nekoliko stotina zapisa u samo jednom odgovoru dok API koji vraća trenutno stanje vremenske prognoze uvijek vraća jedan zapis.

AirPollution API pruža trenutne, prognozirane i povijesne podatke o onečišćenju zraka za sve koordinate na zemlji. Osim osnovnog indeksa kvalitete zraka, API vraća podatke o zagađujućim plinovima, kao što su ugljični monoksid (CO), dušikov monoksid (NO), dušikov dioksid (NO_2), ozon (O_3), sumporni dioksid (SO_2), amonijak (NH_3), i čestice ($PM_{2,5}$ i PM_{10}). Prognoza onečišćenja zraka dostupna je za 5 dana s preciznošću po satu, dok su povijesni podaci dostupni od 27. studenog 2020.
Ovaj API koristimo za dohvaćanje povijesnih i trenutnih podataka. Parametri koji su potrebni za kreiranje API zahtjeva su sljedeći:

- *lat, lon* - geografske koordinate mjesta/grada, obavezan parametar,
- *start* - početni datum u unix formatu, obavezan parametar, samo za dohvaćanje povijesnih podataka,
- *end* - završni datum u unix formatu, obavezan parametar, samo za dohvaćanje povijesnih podataka,
- *appid* - jedinstveni API ključ koji se dobije kao dio pristupnih podataka API-iju, obavezan parametar.

Ovo je primjer API-ija koji u zahtjevu za povijesne podatke u samo jednom odgovoru dobije više zapisa, zato što radi na princip od datuma do datuma, pri čemu vraća zapise u pravilnom vremenskom razmaku od sat vremena. Svi odgovori su u *json* formatu. Budući da koristimo isti API za dohvaćanje povijesnih i trenutnih podataka, poziv API-iju napisan je u obliku jedne funkcije koja je priložena u nastavku.

```
import pandas as pd
import requests
import config

def CallOpenWatherAirQualityCall(lon, lat, hist, startDate=0, end=0):
    response_list = []
    API_KEY = config.api_key

    if hist:
        url = config.OpenWatherAirPollutionURL_HIST
        endDate = ConvertDateTimetoUnixTime(datetime.datetime.now())
        querystring = {"lon":lon,"lat":lat, "start": startDate , "end":
   endDate, "appid": API_KEY}
    else:
        url = config.OpenWatherAirPollutionURL
        querystring = {"lon" : lon, "lat" : lat, "appid" : API_KEY}

    r = requests.get(url, params=querystring)
    response_list.append(r.json())

    df = pd.DataFrame.from_dict(response_list)
    return df
```

Program 2: Primjer funkcije koja poziva Air Pollution API s parametrima

```
{
  "coord":[
    50,
    50
  ],
  "list":[
    {
      "dt":1605182400,
      "main":{
        "aqi":1
      },
      "components":{
        "co":201.94053649902344,
        "no":0.01877197064459324,
        "no2":0.7711350917816162,
        "o3":68.66455078125,
        "so2":0.6407499313354492,
        "pm2_5":0.5,
        "pm10":0.540438711643219,
        "nh3":0.12369127571582794
      }
    }
  ]
}
```


Slika 7: Primjer odgovora Air Pollution API-ija u json formatu
Izvor: [7], 20.9.2022

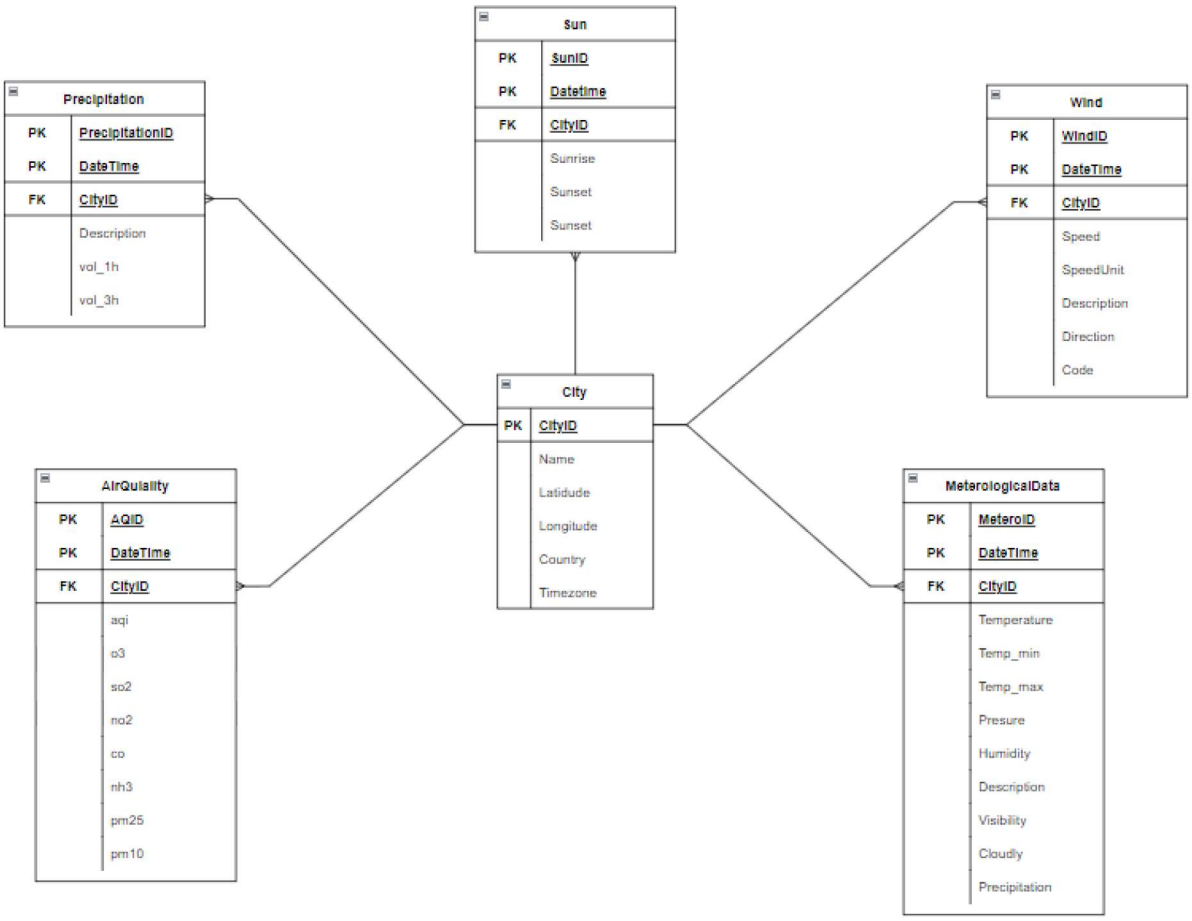
Na slici 7 možete vidjeti primjer jednog API odgovora u *json* formatu. Polja koja sadrži su sljedeća (preuzeto iz službene dokumentacije [7]):

- *cord*: geografske koordinate (dužina, širina) lokacije koju smo zadali u pozivu,
- *dt*: vrijeme i datum u unix formatu za koje su zabilježena mjerenja,
- *aqi*: indeks kvalitete zraka koji može poprimiti jednu od pet vrijednosti 1 = Dobar, 2 = Uredan, 3 = Umjeren, 4 = Siromašan, 5 = Jako siromašan,
- *co*: koncentracija ugljičnog monoksida izražena u $\mu g/m^3$ (mikrogram po kubičnom metru zraka),
- *no*: koncentracija dušikovog monoksida izražena u $\mu g/m^3$,
- *no2*: koncentracija dušikovog dioksida izražena u $\mu g/m^3$,
- *so2*: koncentracija sumpornog dioksida izražena u $\mu g/m^3$,
- *pm25*: koncentracija sitnih čestica $PM_{2.5}$ izražena u $\mu g/m^3$,
- *pm10*: koncentracija sitnih čestica PM_{10} izražena u $\mu g/m^3$,
- *nh3*: koncentracija amonijaka izražena u $\mu g/m^3$.

Kao što se moglo vidjeti iz prethodnih primjera koda, sve ključne varijable, parametri, pristupni podaci i linkovi do API-ija nalaze se u posebnom *config.py* modulu. Osim zbog povjerljivosti navedenih podataka, parametri su odijeljeni u posebni modul kako bi se njima moglo lakše upravljati i kako bi se cijeli projekt mogao konfigurirati na jednom mjestu.

3.3 Modeliranje baze podataka

Baza podataka, koja predstavlja odredište ekstrahiranih i transformiranih podataka, kreirana je pomoću PostgreSQL servera. Sastoji se od 6 tablica, a to su: City, Sun, Wind, Precipitation, Meteorological Data i Air Quality. Glavna tablica u bazi je City i ona predstavlja tablicu činjenica. Tablica City sadrži jedan primarni ključ *CityID* i ona je tablica činjenica iz razloga što se podaci iz drugih tablica povezuju i agregiraju na osnovu nje, odnosno na osnovu podatka *CityID*. Glavni razlog ovog poretka tablica je taj što smo sve podatke iz API-ija dobivali pomoću lokacije zadane geografskim koordinatama grada. Sve ostale tablice su tablice dimenzija i svima im je zajedničko da imaju jedan strani ključ - *CityID*. Također, primarni ključ svake tablice dimenzija sastoji se iz dva dijela: prvi je ID zapisa u tablici, a drugi je vrijeme i datum koji predstavlja točno vrijeme za koje smo očitali vremensku prognozu. Na taj način smo osigurali da se podaci mogu povezati na razini lokacije i točnog vremena mjerenja. Na Slici 8 prikazan je ER (*engl. Entity relationship*) dijagram opisane baze podataka. Iz shematskog prikaza se može zaključiti da baza ima strukturu koja odgovara zvjezdanoj topologiji.
Baza podataka, kao i sve tablice u njoj kreiraju se pokretanjem instalacijske skripte *installation.py* koja se može pronaći u priloženom Git repozitoriju. Proces kreiranja baze i učitavanja povijesnih podataka, za gradove po izboru korisnika, objedinjen je u sklopu *installation.bat* skripte.



Slika 8: ER dijagram centralne baze podataka

3.4 Primjeri transfomacije podataka

Moderni ETL sustavi ostavljaju većinu transformacija za fazu analitike i umjesto toga se fokusiraju na učitavanje minimalno obrađenih podataka u skladište. To povećava fleksibilnost ETL procesa i daje analitičarima veću kontrolu nad modeliranjem podataka. S obzirom na to da je svrha prikupljanja podataka u ovom praktičnom radu znanastvene-istraživačke prirode, u procesu transformacije podataka vodili smo se karakteristikama modernog ETL-a kako bi ostavili što više slobode korisnicima podataka u skladištu. Unatoč tome, potrebno je obaviti osnovno čišćenje podataka tijekom ETL procesa. Pod osnovno čišćenje spadaju svi procesi validacije izvornih podataka kao i uklanjanje *null* vrijednosti i drugih anomalija. Prije učitavanja podataka u skladište potrebno je obraditi određene scenarije koji se često javljaju:

- **Restrukturiranje podataka**: često način na koji su podaci strukturirani u izvornoj pohrani podataka otežava učitavanje u skladište podataka. To je slučaj kod parsiranja ugniježđenih struktura iz JSON API-ja ili prevođenja MongoDB-a u relacijsku strukturu.
- **Upisivanje podataka**: morate upisati podatke koji ulaze u sustav i održavati tu vrstu dok putuju kroz vaš ETL proces (npr. valuta, datum i sl.).
- **Evolucija podatkovne sheme**: ovo je čest slučaj za interne sustave, ali se najviše primjenjuje kada imate posla s vanjskim izvorima podataka kao što su SaaS API-ji. Ako SaaS alat doda novi stupac svojoj zbirci podataka, vaš sustav mora biti spreman da prihvati promjenu, idealno bez ljudske intervencije.

Pored svih klasičnih transformacija podataka te navedenih scenarija koji su uzeti u obzir prilikom implementacije ETL cjevovoda, ovdje ćemo navesti i nekoliko specifičnih transformacija s kojim smo se susreli tijekom implementacije. Većina njih poizlazi iz kombiniranja različitih API-ija čije se skale ili metrike ne podudaraju.

```
import datetime
import time

def ConvertDateTimetoUnixTime(date_time):
    return int(time.mktime(date_time.timetuple()))

def ConvertUnixTimeToDateTime(unix):
    dt = datetime.datetime.fromtimestamp(unix)
    return dt
```

Program 3: Primjer funkcija za promjenu formata datuma

```
def checkPrecipitation(df):
    precipitation = False
    if 'rain' in df or 'snow' in df:
        precipitation = True
    return precipitation
```

Program 4: Primjer funkcije koja popunjava boolean stupac u tablici u ovisnosti prisutnosti/odsutnosti stupca u izvornim podacima

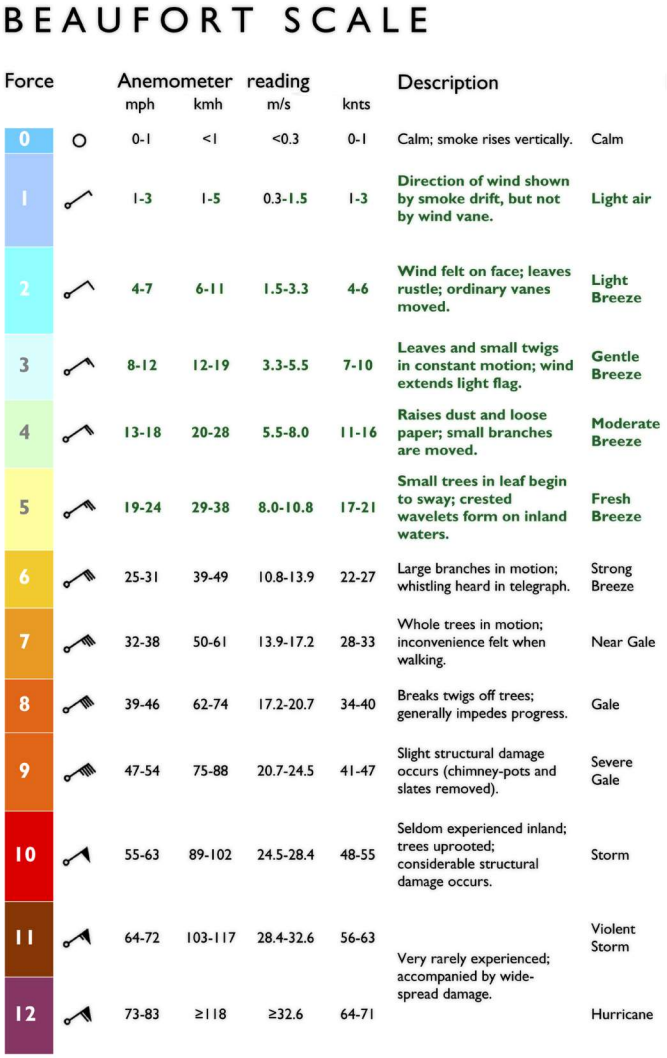
```
def Transform_MeteroData(df):
    metero_record = list()
    metero_record.append(df['main'].to_dict()[0]['temp'])
    metero_record.append(df['main'].to_dict()[0]['temp_min'])
    metero_record.append(df['main'].to_dict()[0]['temp_max'])
    metero_record.append(df['main'].to_dict()[0]['pressure'])
    metero_record.append(df['main'].to_dict()[0]['humidity'])
    metero_record.append(df['weather'].to_dict()[0][0]['description'])
    metero_record.append(df['visibility'].to_dict()[0])
    metero_record.append(df['clouds'].to_dict()[0]['all'])
    metero_record.append(checkPrecipitation(df))
    metero_record.append(df['id'].to_dict()[0])
    metero_record.append(ConvertUnixTimeToDateTime(df['dt'].to_dict()[0]))
    metero_record = tuple(metero_record)
    return metero_record
```

Program 5: Primjer funkcije koja formatira izvorne podatke iz json formata u format pogodan za učitavanje u skladište podataka

Osim transformacije postojećih izvornih podataka, moguće je dodati nove stupce podataka koji nastaju kombinacijom postojećih ili su temeljeni na postojećim podacima. Kao primjer takve situacije navodimo dvije funkcije koje kreiraju dva nova stupca na temelju nekih izvornih podataka. Prva fukcija, koja određuje opis vjetra, temelji se na brzini vjetra (izražena u m/s) i metrici koja je definirana kao Beaufortova skala (prikazana na Slici 9).
Druga funkcija kreira stupac koji sadrži podatak o smjeru vjetra formatiranom u obliku strane svijeta, a temelji se na brojčanoj vrijednosti kuta strane svijeta. Primjer fukcije dan je u Program 7.

```
def GetWindDescription(speed):
    Description = ''
    if speed < 0.2 or speed == 0.2:
        Description = 'Calm'
    elif speed > 0.2 and speed < 1.5:
        Description = 'Light air'
    elif speed > 1.5 and speed < 3.3:
        Description = 'Light breeze'
    elif speed > 3.3 and speed < 5.5:
        Description = 'Gentle breeze'
    elif speed > 5.5 and speed < 7.9:
        Description = 'Moderate breeze'
    elif speed > 7.9 and speed < 10.7:
        Description = 'Fresh breeze'
    elif speed > 10.7 and speed < 13.8:
        Description = 'Strong breeze'
    elif speed > 13.8 and speed < 17.1:
        Description = 'High wind, moderate gale, near gale'
    elif speed >17.1 and speed < 20.7:
        Description = 'Gale, fresh gale'
    elif speed > 20.7 and speed < 24.4:
        Description = 'Strong/severe gale'
    elif speed > 24.4 and speed < 28.4:
        Description = 'Storm, whole gale'
    elif speed > 28.4 and speed < 32.6:
        Description = 'Violent storm'
    elif speed > 32.6:
        Description = 'Hurricane force'
    return Description
```

Program 6: Primjer funkcije koja kreira novi stupac u bazi na temelju postojećeg



Slika 9: Beaufort skala za određivanje opisa i jačine vjetra
Izvor: `https://billboyheritagesurvey.wordpress.com/2012/01/13/beaufort-scale/`

Još jedan od važnih koncepata modernog ETL-a kojeg je važno istaknuti je implementacija paralelne obrade. To je relativno nova značajka ovog sustava, a ponajviše je zastupnjena upravo u koraku transformacije podataka. Paralelizacija omogućava niz metoda za poboljšanje ukupne izvedbe ETL-a kada se radi s velikim količinama podataka.
ETL aplikacije implementiraju tri glavne vrste paralelizma:

- Podaci: dijeljenjem jedne sekvencijalne datoteke u manje podatkovne datoteke kako bi se omogućio paralelni pristup i omogućila paralelna obrada podataka.
- Cjevovod: omogućuje istovremeno pokretanje nekoliko komponenti na istom toku podataka, npr. traženje vrijednosti u zapisu 1 u isto vrijeme kada se dodaju dva polja u zapis 2.
- Komponenta: istodobno izvođenje više procesa na različitim tokovima podataka u istom poslu, npr. sortiranje jedne ulazne datoteke uz uklanjanje duplikata u drugoj datoteci.

Sve tri vrste paralelizma obično djeluju kombinirano u jednom poslu ili zadatku.

```
def GetWindDirectionCode(direction):
    if direction > 348.75  or direction < 11.25:
        Code = 'N';
    elif direction > 11.25 and direction < 33.75:
        Code = 'NNE';
    elif direction > 33.75 and direction < 56.25:
        Code = 'NE';
    elif direction > 56.25 and direction < 78.75:
        Code = 'ENE';
    elif direction > 78.75 and direction < 101.25:
        Code = 'E';
    elif direction > 101.25 and direction < 123.75:
        Code = 'ESE';
    elif direction > 123.75 and direction < 146.25:
        Code = 'SE';
    elif direction > 146.25 and direction < 168.75:
        Code = 'SSE';
    elif direction > 168.75 and direction < 191.25:
        Code = 'S';
    elif direction > 191.25 and direction < 213.75:
        Code = 'SSW';
    elif direction > 213.75 and direction < 236.25:
        Code = 'SW';
    elif direction > 236.25 and direction < 258.75:
        Code = 'WSW';
    elif direction > 258.75 and direction < 281.25:
        Code = 'W';
    elif direction > 281.25 and direction < 303.75:
        Code = 'WNW';
    elif direction > 303.75 and direction < 326.25:
        Code = 'NW';
    elif direction > 326.25 and direction < 348.75:
        Code = 'NNW';
    return Code;
    \llabel{test}
```

Program 7: Primjer funkcije koja kreira novi stupac smjer vjetra u formatu strane svijeta na osnovu brojčane vrijednosti smjera

3.5 Učitavanje podataka u skladište podataka

Treći korak u ETL procesu je učitavanje podataka u centralnu bazu. S obzirom na to da je cijeli projekt implementiran u programskom jeziku Python, a baza podataka se nalazi na PostgreSQL serveru, biblioteka koja nam je omogućila ovaj korak je *Psycopg2*.
Psycopg je najpopularniji PostgreSQL adapter baze podataka za programski jezik Python. Njegove glavne značajke su potpuna implementacija Python DB API 2.0 specifikacije i sigurnost niti (nekoliko niti/threadova može dijeliti istu vezu).
Psycopg2 je *libpq* omotač koji je implementiran u *C*-u, što rezultira učinkovitošću i sigurnošću. Sadrži pokazivače na strani klijenta i poslužitelja, asinkronu komunikaciju i obavijesti, te podršku za kopiranje. Mnogi Python tipovi podataka prilagođeni su odgovarajućim PostgreSQL tipovima podataka što olakšava validaciju i sam proces obrade podataka.
Veza s bazom se stvara pomoću tvorničke funkcije *connect()*, kojoj proslijedimo sve pristupne parametre za server i trenutnu bazu. Pomoću *cursor* klase Python može izvršiti PostgreSQL naredbu u sesiji. Primjer koda fukcije napisane u Pythonu koja učitava podatke u PostgreSQL bazu priložen je u nastavku.

```
def insert_SyncData(tableName, data, precipitationData):
    try:
        connection = psycopg2.connect(
                        database=config.database_newDatabaseName,
                        user=config.database_user,
                        password=config.database_password,
                        host=config.database_host,
                        port= config.database_port)

        cursor = connection.cursor()
        #define sql query to execute from config
        sql_insert_query = config.SQL_QUERIES[tableName]

        #insert data to all table except in Precipitation
        cursor.execute(sql_insert_query, data)

        #insert line/lines in Precipitation if exists
        if len(precipitationData) > 0:
            sql_insert_query = config.SQL_QUERIES['Precipitation']
            execute_values(cursor, sql_insert_query, precipitationData)

        connection.commit()
        print(cursor.rowcount, "Record inserted successfully into {} table
    ".format(tableName))
    except (Exception, psycopg2.Error) as error:
        print("Failed inserting record into {} table {}".format(tableName,
    error))

    finally:
        # closing database connection
        if connection:
            cursor.close()
            connection.close()
            print("PostgreSQL connection is closed")
```

Program 8: Primjer funkcije koja učitava podatke u bazu

Kako bi cijeli ovaj alat za prikupljanje podataka učinili što kompaktnijim i minimizirali broj potrebnih ljudskih intervencija za njegovo izvršavanje, proces sinkronizacije je postavljen na automatsko pokretanje. To smo postigli pomoću Planera zadataka (*engl. Windows Task Scheduler*) koji je sastavni dio Windows 10 i viših verzija. Prilikom pokretanja instalacijske skripte, koja će osim kreiranja baze podataka, kreiranja potrebnih tablica unutar baze i učitavanja povijesti također smjestiti proces sinkronizacije u planer zadataka. On omogućava da se odabrana skripta pokreće po točno definiranom rasporedu te da se najnoviji podaci učitaju u bazu u točno definirano vrijeme. Na taj način je osigurano da skladište podataka uvijek sadrži aktualne podatke i redovito se ažurira bez da se skripte pokreću ručno. Prilikom sikronizacije, obavijesti o pokretanju i uspješnosti pokrenutog procesa ispisuju se u *log* tekstualnu datoteku kako bi mogli voditi evidenciju i jednostavnije detektirali problem kada do njega dođe.
Po završetku implementacije i testiranja cjelokupnog projekta, uslijedio je još jedan proces - *dockerizacija*. S obzirom na to da je projekt implementiran lokalno i vezan je za lokalno instaliran PostgreSQL server, cilj je bio učiniti ovaj projekt dostupnim, pa smo iz tog razloga postavili projekt u *Docker* kontejner. Docker je alat otvorenog koda koji isporučuje vašu aplikaciju sa svim potrebnim funkcijama kao jedan paket, dok je spomenuti proces dockerizacije proces pakiranja, postavljanja i pokretanja aplikacija pomoću Docker spremnika.

4 Primjene i komercijalna rješenja

ETL alati su važan mehanizam da se relevantni podaci okupe na jednom mjestu kako bi bili djelotvorni, odnosno kako bi se analizirali i omogućili rukovoditeljima i drugim dionicima da na temelju njih donose poslovne odluke. Analiza podataka i donošenje poslovnih odluka je generalno širok pojam kojeg možemo smjestiti u različite sfere kako poslovanja, tako i znanosti. Kako bi dobili bolju predodžbu koja je svrha ETL alata u praksi, navest ćemo neke od istaknutijih primjena:

- **Skladištenje podataka**: ETL se koristi za prikupljanje podataka iz više izvora u skladište podataka gdje se podaci analiziraju.
- **Strojno učenje i umjetna inteligencija**: Strojno učenje je način davanja smisla podacima bez eksplicitnog programiranja analitičkih modela. Umjesto toga, sustav uči iz podataka koristeći tehnike umjetne inteligencije. ETL se u ovom konktekstu može koristiti za premještanje svih potrebnih podataka na jednu lokaciju za potrebe strojnog učenja.
- **Integracija marketinških podataka**: podrazumijeva sakupljanje svih marketinških podataka (npr. podaci o klijentu, podaci s drušvenih mreža, podaci web analize, podaci reakcija na oglase i sl.) na jedno mjesto, gdje se oni mogu analizirati i biti iskustveni temelj za buduće poslovne planove. U ovom konktekstu se ETL koristi za prikupljanje i pripremu marketinških podataka.
- **Integracija IoT podataka**: IoT [8] (internet stvari) je pojam koji podrazumijeva skup povezanih uređaja koji su sposobni da prikupljaju i prenose podatke pomoću senzora ugrađenih u hardver. Primjeri IoT uređaja su tvornička oprema, mrežni poslužitelji, pametni telefoni, pametni strojevi i sl. Uloga ETL alata u ovom konktekstu je da premješta podatke iz više IoT uređaja na jedno mjesto gdje se oni mogu analizirati i koristiti za potencijalna unapređenja uređaja.
- **Replikacija baze podataka**: je proces koji uzima podatke iz vanjskih baza podataka kao što su Oracle, MongoDB, Microsoft SQL Server, MySQL, PostgreSQL i sl. i kopira ih u skladište podataka u oblaku. ETL se ovdje koristi za proces replikacije. To može biti jednokratna operacija ili proces koji traje kada se podaci ažuriraju.
- **Migracija u oblak**: Organizacije sve češće premještaju svoje podatke i aplikacije s lokalnih servera u oblak kako bi učinile aplikacije skalabilnijima, uštedjele novac za memoriju i hardvere i učinile svoje podatke sigurnijima. ETL ovdje ima važnu ulogu zato što se pomoću njega pokreću migracije podataka.
- **Automatizacija**: U prošlosti podacima i podatkovnim cjevovodima se upravljalo isključivo ručno. Korištenje ETL alata značajno je doprinijelo automatizaciji te omogućilo prenošenje velike količine podataka u proizvodne pogone ili skladišta podataka bez ikakve intervencije, osim postavljanja početnih parametara.

[8] *engl. Internet of things*

Podaci zauzimaju važno mjesto u današnjem marketinškom svijetu, stoga mnoge tvrke koriste podatke kako bi poboljšale učinkovitost svojeg poslovanja. Kako bi u potpunosti iskoristili dostupne podatke, postoje brojni gotovi komercijalni ETL alati koji mogu uštedjeti vrijeme koje je potrebno da se podaci organiziraju i postanu zaista korisni. Obzirom da ETL alati na tržistu postoje više 30 godina pojavile su se različite vrste gotovih rješenja, a ovdje ćemo izdvojiti po neku iz svake kategorije.

- **ETL softveri za poduzeća** (*engl. Enterprise Software ETL*) su komercijalni alati temeljeni na aplikacijama koje kao proizvod nudi nekoliko tvrtki, zahtijevaju nabavu licenci. Neki od primjera na tržištu su:
  - **IBM DataStage**: alat za integraciju podataka izgrađen oko dizajna klijent-poslužitelj. Zadaci se stvaraju iz Windows klijenta i izvršavaju prema središnjem skladištu podataka na poslužitelju. Alat je osmišljen za podršku ETL i ELT modela i podržava integracije podataka iz više izvora i aplikacija uz održavanje visokih performansi. Ovaj alat se bazira na lokalnoj primjeni, ali dostupan je i u verziji u oblaku: DataStage za IBM Cloud Pak.
  - **Oracle Data Integrator**: platforma dizajnirana za izgradnju, upravljanje i održavanje tijeka rada integracije podataka u organizacijama. Podržava cijeli spektar zahtjeva za integraciju podataka od masovnih skupnih učitavanja do podatkovnih usluga orijentirane na usluge. Također podržava paralelno izvršavanje zadataka za bržu obradu podataka i nudi ugrađene integracije s Oracle GoldenGate i Oracle Warehouse Builder.
  - **Fivetran**: platforma koja ima za cilj poboljšati proces upravljanja podacima sa svojim praktičnim alatima. Sadrži softver koji prati ažuriranja API-ja i izvlači najnovije podatke iz baze podataka u nekoliko minuta, a pored toga je i jednostavan za korištenje. Uz ETL alate, Fivetran nudi usluge sigurnosti podataka, replikaciju baze podataka i podršku 24/7.
- **ETL alati otvorenog koda** (*engl. Open Source ETL*) besplatni su za korištenje, mogu se koristiti putem interneta ili unutar mreže, najčešće su pisani u Pythonu. Najpoznatiji open-source alati koji podržavaju ETL su:
  - **Pentaho**: alat za poslovnu inteligenciju koji omogućava integraciju, čišćenje i pohranu podataka, izvješćivanje, nadzorne ploče i sl. Omogućava dijeljenje informacija s krajnjim korisnicima i podržava pristup za IoT tehnologije kako bi olakšao strojno učenje. Postoji open-source verzija s ograničenim mogućnostima te licencirana verzija s dodatnim značajkama.
  - **Talend Open Studio**: open-source alat dizajniran za brzu izgradnju podatkovnih cjevovoda. Podaci se mogu učitati pomoću Open Studio GUI-ja iz Excela, Dropboxa, Oraclea, Salesforcea, Microsoft Dynamicsa i drugih izvora podataka. Talend Open Studio ima ugrađene konektore za izvlačenje informacija iz različitih okruženja, uključujući sustave upravljanja relacijskim bazama podataka, SaaS platforme, aplikacijske pakete.
- **ETL alati u oblaku** (*engl. ETL Cloud Services*) dostupni su putem interneta i obično im se pristupa putem web preglednika. Mnoge usluge u oblaku pružaju ETL mogućnosti kao uslugu uz pohranu svojih podataka na platformi u oblaku. Neki od istaknutijih alata u oblaku za ETL su:

- **Azure Data Factory**: patforma za integracije podataka bez poslužitelja izgrađena na modelu plaćanja prema korištenju. Platforma nudi sučelja bez koda i sučelja koja se temelje na kodu i može povući podatke iz više od 90 ugrađenih konektora. Osim toga, Azure Data Factory integrira se s Azure Synapse Analytics za pružanje napredne analize podataka i vizualizacije. Platforma također podržava Git za kontrolu verzija i kontinuiranu integraciju ili implementaciju radnih procesa za DevOps timove.
- **Google Cloud Dataflow**: potpuno je upravljana usluga obrade podataka izgrađena za optimizaciju računalne snage i automatiziranje upravljanja resursima. Usluga je usmjerena na smanjenje troškova obrade kroz fleksibilno planiranje i automatsko skaliranje resursa kako bi se osiguralo da korištenje odgovara potrebama. Uz to, Google Cloud Dataflow nudi AI mogućnosti za pokretanje prediktivne analize i otkrivanje anomalija u stvarnom vremenu dok se podaci transformiraju.

- **Prilagođeni ETL** (*engl. Custom ETL*) je alat izgrađen kako bi ispunio posebne zahtjeve pojedine tvrtke i najčešće je izgrađen unutar same tvrtke. Programski jezici koji mogu poslužiti u izradi vlastitog rješenja su:
  - **SQL**: Ako su izvor i odredište podataka isti, korištenje samo SQL-a također je vrlo moćna opcija. SQL je učinkovit u izvođenju osnovnih transformacija, međutim nije baš pogodan za složene transformacije koja uključuju stabla odlučivanja i pozivanje vanjskih izvora. Korištenje samo SQL za obradu podataka je i financijski pogodno, zato što nisu potrebne nikakve naknade za licencu ili tehnologije.
  - **Python**: programski jezik opće namjene koji je popularan alat za obavljanje ETL zadataka zbog jednostavne upotrebe i velikog broja dostupnih biblioteka za pristup bazama i pohrani podataka. Može se koristiti umjesto ETL alata zato što je moćan i fleksibilan.
  - **Java**: programski jezik opće namjene koji se može koristiti za izgradnju ETL alata. Java je jedan od najpopularnijih programskih jezika i ima opsežnu podršku za različite izvore podataka i transformaciju podataka.

U nekim izvorima navodi se i pojam **Obrnuti ETL** (*engl. Reverse-ETL*). To je proces sinkronizacije podataka iz skladišta podataka u druge operativne alate koje koriste poslovni timovi. Drugim riječima, to je suprotno od ETL procesa gdje se podaci učitavaju u skladište podataka. Takve su integracije vrlo korisne kada podatke ili uvide treba dijeliti između različitih odjela organizacije.

4.1 Prednosti korištenja komercijalnog rješenja

Kada je riječ o organizacijama koje ETL sustave koriste za standardno prikupljane podataka, integracije ili slične, komercijalnim rješenjima predviđene procese, kupovina gotovog rješenja je najbolja opcija. Također, gotovo komercijalno rješenje je opcija i za organizacije koje nemaju vremena, želje, potrebe ili stručnosti za izgradnju i upravljanje vlastitim cjevovodima. Neke od prednosti korištenja gotovog rješenja:

- Ušteda vremena i novca zato što možete odmah započeti koristiti gotov alat: Tvrtka koja vam omogućava ETL alat uložila je svoje vrijeme, energiju i najbolje ljude da vam omogući proizvod koji je njihova specijalnost. Organizacija koja kupuje proizvod može se fokusirati na rješavanje drugih problema koji su njena specijalnost, fokusirajući se samo na dobrobiti i rezultate koje dobiva koristeći gotov sustav.

- Kada koristite plaćeni/licencirani proizvod imate određenu garanciju da je proizvod dobro dokumentiran, da se pravovremeno održava i da imate službu za korisnike kojoj se možete obratiti za rješavanje problema koji su u njihovoj domeni.

- Upravlja se protokom podataka i ugrađena je skalabilnost: Upravljana automatizacija transformira više tokova podataka u korisni sloj vizualizacije koji omogućuje krajnjim korisnicima da postavljaju upite podacima bez potrebe da razumiju kako bilo koji od njih funkcionira. Također, bilo koji profesionalni alat trebao bi podjednako dobro rukovoditi podacima bez obzira na povećanje veličine podataka, povećanje broja izvora podataka, bez da to utiče negativno na performanse.

4.2 Prednosti dizajniranja vlastitog ETL rješenja

Ponekad gotova komercijalna rješenja neće u potpunosti zadovoljiti sve zahtjeve koje stavljate pred sustav ili jednostavno neće biti dovoljno prilagođeni i precizni. Pogotovo ako imate vrlo specifične podatke, ili koristite alate koji nisu kompatibilni s postojećim rješenjima. Neke od ključnih prednosti kada se odlučite za scenarij dizajniranja i izgradnju vlastitog prilagođenog rješenja su: znate na što se želite fokusirati, kako želite da sve bude konfigurirano i koje metrike su važne. Dakle, imate potpunu kontrolu nad svojim podatkovnim sustavom. Prilagođeno rješenje nudi fleksibilnost koju ne može pružiti niti jedan gotovi alat budući da kodiranjem možete izgraditi što god želite i omogućiti izravno upravljanje svim metapodacima ovisno o poslovnim potrebama. Dakako, važno je napomenuti da apsolutna kontrola nad sustavom donosi i neke negativne stvari za sobom, kao što je upravljanje kvarovima i neredima u podatkovnom cjevovodu. Dizajniranje i izgradnja kvalitetnog i funkcionalnog ETL sustava svakako nije jednostavan zadatak.
Ako je vrsta integracije podataka koju organizacija zahtijeva vrlo jedinstvena, ručno kodiranje ETL-a osim što je najbolja, ujedno je i financijski najpogodnija opcija. Svakako treba uzeti u obzir da izgradnja prilagođenog ETL sustava donosi sa sobom određene financijske izdatke. Međutim, ako u konačnici prilagođeni ETL alat zadovolji sve zahtjeve za efikasnije poslovanje, opcija izgradnje vlastitog rješenja postaje isplativija.

Literatura

[1] C. Cote, M. Lah, D. Sarka, *SQL Server 2017 Integration Services Cookbook : ETL tehniques to load and transform data from various sources using SQL Server 2017 Integration Services*, Packt Publishing Ltd., Birmingham, UK, 2017.

[2] Git, *Repozitorij*, Git repozitorij sa cjelokupnim kodom praktičnog projekta dostupan je na: `https://github.com/lnedic/MeteroPollen`.

[3] R. Kimball, J. Caserta, *The Data Wrehouse ETL Toolkit*, Wiley Publishing, Inc., Crosspoint Boulevard, Indianapolis, 2004.

[4] R. Kimball, M. Ross, *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, Third Edition*, John Wiley & Sons, Inc., Crosspoint Boulevard, Indianpolis, 2013.

[5] N.N., *Dimensions and Facts in Terms of Data Warehousing*, Mozart Data Platform (2021.), Dostupno na: `https://www.mozartdata.com/post/dimensions-and-facts-in-terms-of-data-warehousing`.

[6] N.N., *What Is a Data Warehouse: Overview, Concepts and How It Works*, SimpliLearn (2022.), Dostupno na: `https://www.simplilearn.com/data-warehouse-article`.

[7] OpenWather platforma, *Air Pollution API dokumentacija*, Službena dokumentacija, dostupna na: `https://openweathermap.org/api/air-pollution`.

[8] OpenWather platforma, *Current Weather data API dokumentacija*, Službena dokumentacija, dostupna na: `https://openweathermap.org/current`.

[9] OpenWather platforma, *One Call API dokumentacija*, Službena dokumentacija, dostupna na: `https://openweathermap.org/api/one-call-3`.

[10] J. Smith, *Data Warehouse Architecture, Components Diagram Concepts*, Guru99 (2022.), Dostupno na: `https://www.guru99.com/data-warehouse-architecture.html`.

[11] D. Taylor, *ETL (Extract, Transform, and Load) Process in Data Warehouse*, Guru99 (2022.), Dostupno na:`https://www.guru99.com/etl-extract-load-process.html`.

[12] D. Taylor, *What is Data Warehouse? Types, Definition Example*, Guru99 (2022.), Dostupno na: `https://www.guru99.com/data-warehousing.html`.

[13] A. Zelen, *Dimensions of Dimensions: A Look at Data Warehousing's Most Common Dimensional Table Types*, Vertabelo (2016.), Dostupno na: `https://www.vertabelo.com/blog/dimensions-of-dimensions-a-look-at-data-warehousings-most-common-dimensional-table-types/`.

Sažetak

Tema ovog diplomskog rada su ETL sustavi. Glavni cilj rada je objasniti što su ETL sustavi, čemu služe, koji su njegovi sastavni procesi i gdje ih možemo pronaći u primjeni. Na samom početku ovog rada bavimo se odredištem podataka, koje ekstrahiramo i transformiramo pomoću ETL sustava, odnosno skladištem podataka. U radu su opisane glavne karakteristike skladišta, nabrojane i definirane njegove glavne kompomente, te je jedno poglavlje posvećeno arhitekturi skladišta. Nakon definiranja odredišta i svrhe pohranjenih podataka, opisan je postupak pomoću kojeg podaci dolaze u skladište: ekstrahiranje podataka iz različitih izvora, obradu podataka i u konačnici pohranjivanje podataka u skladište gdje oni postaju vrijedne infomacije. Na primjeru dizajniranja i izgradnje ETL sustava za MeteroPollen bazu objasnili smo kako od nule kreirati vlastiti prilagođeni ETL sustav. U sklopu ovog ilustrativnog primjera navedeni su i dijelovi koda napisani u programskog jeziku Python. Posebno poglavlje posvećeno je vrstama ETL alata gdje je navedeno nekoliko primjera komerijalnih rješenja svake vrste. Na samom kraju rada naveli smo neke od prednosti korištenja gotovih alata te istaknuli kada se ipak bolje odvažiti za kreiranje vlastitog rješenja.

Ključne riječi: skladište podataka, centralna baza podataka, dimenzije, činjenice, ETL, ekstrahiranje podataka, transformiranje podataka, učitavanje podataka, API, analiza podataka, poslovna inteligencija

ETL processes

Summary

The topic of this master thesis is ETL systems. The main goal of this thesis was to explain what ETL systems are, what they are for, what are its constituent processes and where we can find them in use. At the beginning of this paper, we deal with the data destination, which we extract and transform using ETL, that is, the data warehouse. The paper describes the main characteristics of the warehouse, enumerates and defines its main components, and a chapter is dedicated to the architecture of the data warehouse. After defining the destination and purpose of the stored data, we described the process by which the data comes to the storage: extracting data from various sources, processing the data and finally storing the data in the warehouse where it becomes valuable information. Using the example of designing and building an ETL system for the MeteroPollen database, we explained how to create your own customized ETL system from scratch. As part of this illustrative example, parts of the source code written in the Python programming language are attached. A separate chapter is dedicated to the types of ETL tools, where several examples of commercial solutions of each type are listed. At the end of this paper, we mentioned some of the advantages of using commercial tools and pointed out when it is better to create your own solution.

Key words: data warehouse, cetral database, dimensions, facts, ETL, extract, transform, load, API, data analysis, bussines inteligence

Životopis

Magdalena Nedić, rođena 18.10.1998. godine u Vinkovcima.
Pohađala je Osnovnu školu fra Ilije Starčević u Tolisi, BiH te Gimnaziju u ŠC fra Martina Nedića u Orašju, opći smjer. Obrazovanje nastavlja na preddiplomskom studiju Matematike i računarstva na Odjelu za matematiku 2017. godine te ga završava 2020. godine s temom završnog rada „Konvergencija nizova slučajnih varijabli" kod mentorice izv. prof. dr. sc. Dragane Jankov Maširević. U sklopu prediplomskog studija odradila je stručnu praksu u firmi Mega-IN2 u Orašju. Prediplomsku razinu studija završila je s *cum lude* pohvalom. Akademske 2019/2020. godine nagrađena je Rektorovom nagradom za izvrstan seminarski rad iz predmeta Statistički praktikum pod naslovom „Testiranje stereotipa o ljevacima".
Odmah po završetku prediplomskog studija upisuje diplomski studij Matematike i računarstva na Odjelu za matematiku. Na diplomskom studiju odradila je stručnu praksu u firmi BE-terna u kojoj je trenutno zaposlena. Dobitnica je Pročelnikove nagrade i pohvale za uspješnost za akademsku 2020/21.