UNIVERZITET U NIŠU

ELEKTRONSKI FAKULTET

Tehnologije za implementaciju real-time data warehouse

Završni diplomski rad
Studijski program: Elektrotehnika i računarstvo
Modul: Računarstvo i informatika

Student:

Vidosava Arsić, 16478

Mentor:

Prof. dr Bratislav Predić

Niš, 2024. godina

UNIVERZITET U NIŠU

ELEKTRONSKI FAKULTET

Tehnologije za implementaciju real-time data warehouse

Završni diplomski rad
Studijski program: Elektrotehnika i računarstvo
Modul: Računarstvo i informatika

Student:

Vidosava Arsić, 16478

Mentor:

Prof. dr Bratislav Predić

Komisija:

Datum prijave: _____

Datum predaje: _____

Datum odbrane: _____

1. _____

2. _____

3. _____

Niš, 2024. godina

SPISAK KORIŠČENIH SKRAĆENICA I OZNAKA

SKRAĆENICE

BI	Business Intelligence
ETL	Extract, Transform, Load
OLAP	Online Anayltical Processing
DSS	Decision Support System
DASD	Direct Access Storage Device
DBMS	Database Managment System
MIS	Managment Information System
OLTP	Online Transaction Processing
SDLC	System Development Life Cycle
CLDS	Cycle Life Development System
RTDW	Real-time Data Warehouse
ODI	Oracle Data Integration
CDC	Changed Data Capture
ODSI	Oracle Data Service Integrator
HTAP	Hybrid Transactional/Analytical Processing
HDFS	Hadoop Distributed File System
DWD	Data Warehouse Detail
ODS	Operational Data Store
DWS	Data Warehouse Service
MPP	Massively Parallel Processing
CNCF	Cloud Native Computing Foundation
IoT	Internet of Things
AWS	Amazon Web Services
GCP	Google Cloud Platform

SADRŽAJ:

1. Uvod

U današnjem svetu vođenom podacima, mnoge kompanije koriste real-time data warehouse za potrebe svoje poslovne logike (BI - Business Intelligence) i analize podataka. Data warehouse omogućuje da se donose bolje odluke, stimuliše rast kompanije i obezbeđuje vrednost pojedinih podataka njihovim korisnicima. Data warehouse predstavlja skladište podataka i sistem za upravljanje dizajnirano sa jednim ciljem: da upravlja podacima i analizira ih sa ciljem uspostavljanja biznis logike. Real-time data warehouse omogućava analizu podataka u realnom vremenu. Tema ovog diploskog rada u osnovi je : šta je data warehouse, šta je real-time data warehouse (RTDW), koji su osnovni koncepti RTDW-a i koja je razlika data warehouse i RTDW-a. Takođe, biće reči i o tome koji su slučajevi korišćenja data warehouse-a i tehnologija kojom se RTDW može implementirati. Biće dat primer implementacije tehnologije. Takođe, biće jasne prednosti i najbolja praksa koju bi trebalo pratiti kako bi se izvukao maksimum.

2. Evolucija

2.1. Problem master fajlova

Priča o data warehouse-u započinje evolucijom informacionih sistema i sistema za odlučivanje (DSS – Decision Support System). Sagledavajući istoriju imamo širu sliku onoga što je značajno u izučavanju data warehouse-a. Sredinom 60-tih godina prošlog veka, došlo je do razvoja medijuma koji su korišćeni za skladištenje podataka. Kako su medijumi za skladištenje postajali fleksibilniji tako je rasla mogućnost rada sa podacima, pa čak i sa onim podacima koji su do tada bili suvišni. Dakle, u slučaju kada se pojedini podaci smatraju za suvišne, oni se odbacuju i ne skladište, jer se prednost daje osnovnim podacima, bez čijeg postojanja ne bi bio moguć rad sistema. Međutim, kako je postalo moguće skladištiti veće količine podataka, stvorili su se novi problemi u računarstvu:

1. Potreba za sinhronizacijom nakon ažuriranja
2. Kompleksnost održavanja programa
3. Kompleksnost razvoja novih programa
4. Potreba za velikom količinom hardvera koji može obrađivati podatke

Dakle nastao je problem master fajlova. Podaci se skladište u velikom broju fajlova i pitanje je kako im najbrže pristupiti.

2.2 Prednost DASD-a

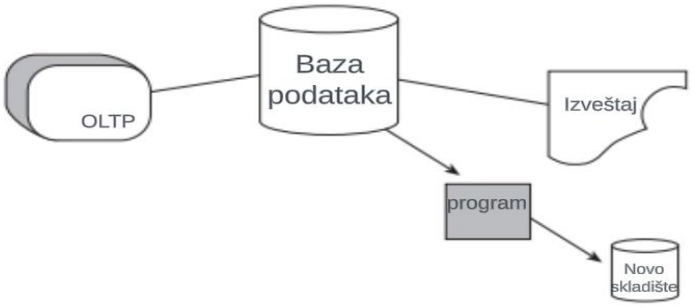
Zahvaljujući prednostima koje pruža DASD – Direct Access Storage Device, a jedna od najvažnijih prednosti je pristup podacima. Pristup 1, 2, 3, ... , n podatku nije neophodan ukoliko želimo pristup n+1 podatku. Ukoliko je adresa n+1 podatka poznata, može mu se direktno pristupiti. Samim tim, vreme pristupa podatku je značajno smanjeno. Vreme pronalaženja podatka meri se milisekundama. Zajedno sa pojmom DASD nastao je tip softverskog sistema poznat kao DBMS - Database Managment System. Svrha DBMS bila je da olakša programerima da skladište podatke i pristupaju istim na DASD-u. Pored toga, DBMS vodi računa o problemima kao što je indeksiranje ali i o mnogim drugim. DBMS i

DASD dali su rešenje problema master fajlova. Sa pojmom DBMS došao je pojam baza podataka. Za razliku od skupa fajlova u kojima se podaci skladište baza podataka predstavlja centralizovano rešenje.

OLTP – Online Transaction Processing je nakon koraka koji je napravljen kreiranjem baza podataka, predstavljao još jedan iskorak. OLTP otvorio je potpuno novi pogled na biznis i procese. Pružao je rešenja sistema koji su do tada bili nerešivi ili neadekvatno rešivi. OLTP sposoban je da podrži aplikacije orijentisane ka transakcijama. Transakcija se u računarstvu posmatra kao celina. Baš zbog toga je jedna od glavnih karakteristika OLTP sistema pozitivan ACID test.

2.3 Nastanak DSS-a

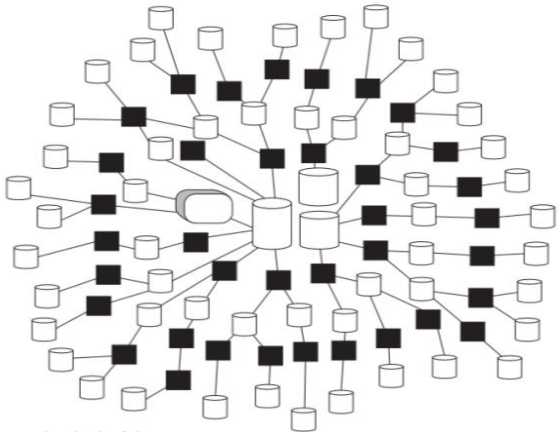
Sredinom 80-tih godina nastaje četvrta generacija jezika, 4GL, to jesu problemom orijentisani jezici, neproceduralni programski jezici. Proceduralni jezik specificira kako će nešto biti izvršeno, a neproceduralni šta će biti izvršeno, ne ulazeći u detalje kako. Programeri i krajnji korisnici koriste 4GL za razvoj softverskih aplikacija. Razvojem PC računara i 4GL tehnologije postalo je jasno da se može više uraditi od jednostavnog procesiranja transakcija odnosno OLTP-a. Ranije poznat kao MIS – Managment Information System, a danas kao DSS – Decision Support System nastao je u ovom trenutku. DSS predstavlja sistem za podršku u odlučivanju, on se za razliku od OLTP sistema ne bavi procesiranjem transakcija već analizom podataka.



Slika 1. Ekstraktovanje podataka, uz pomoć programa, deo podataka se ekstraktuje u zasebno skladište

Nastankom OLTP sistema javljala se potreba za ekstraktovanjem podataka, prikazano na slici 1. Ekstraktovanjem se prolazi kroz bazu podataka ili fajlove i izvlače se podaci koji zadovoljavaju određeni uslov, zatim se unose u neki drugi fajl ili bazu. Sam koncept se otrgao kontroli, postala je svakodnevnica da se podaci ekstraktuju, pri tome kreirajući nova skladišta za podatke. Ovakva praksa stvara takozvanu prirodno razvijenu arhitekturu (naturally evolving architecture). Međutim, postoji problem do koga se dolazi primenjujući ovaj koncept. Poteškoće su sledeće:

1. Pouzdanost podataka
2. Produktivnost
3. Nesposobnost da se podaci transformišu u informacije



Slika 2. Prirodno razvijena arhitektura, nastanak mreže

Na slici 2. možemo videti posledicu kreiranja mreže, veliki broj skladišta podataka dovodi do nastanka "paukove" mreže. Ovakvi sistemi postaju teški za rad. Podaci u ovakvom sistemu postaju nepouzdani. Dolazi do toga da su podaci nekonzistentni. Podaci u jednom skladištu u mreži mogu imati totalno drugačiju vrednost u odnosu na neko drugo skladište. Postavlja se pitanje koje skladište je verodostojno. Zbog ovog i drugih problema, kao što je nesposobnost da od podatka nastanu informacije, cene ovakvih sistema i performansi, korišćenje ovakve arhitekture je loše rešenje.

3. Koncepti data warehouse-a

3.1 OLAP pristup

Pored OLTP-a, razlikujemo i OLAP obradu. OLAP – Online Analytical Processing. Za razliku od OLTP sistema koji su dizajnirani za podršku svakodnevnim transakcijama i rutinskim poslovnim aktivnostima. To uključuje unos, ažuriranje i brisanje podataka. OLAP sistemi su dizajnirani za analizu podataka i podršku poslovnim odlukama. Omogućavaju korisnicima da istražuju i analiziraju podatke, kreiraju izveštaje, i vrše kompleksne upite radi donošenja strategijskih odluka. Cilj je pružiti korisnicima brz pristup velikim količinama podataka za analizu.

Karakteristike OLTP – Online Transaction Processing jesu:

1. Sadrži trenutne podatke
2. Koristan u aplikacijama koje su trenutno aktivne
3. Baziran na ER modelu
4. Obezbeđuje osnovne podatke i podatke visokog nivoa detaljnosti
5. Koriste se za upis podataka u bazu podataka
6. Veličina baze podataka je od 100 MB do 1 GB
7. Obezbeđuju visoke performanse, brzinu
8. Sve transakcije obavljene su od strane klijenata

Karakteristike OLAP – Online Analytical Processing :

1. Uključuje istorijske podatke
2. Koristan je u analizi poslovne logike
3. Baziran je na šemi zvezde, pahulje i tabele činjenica

4. Obezbeđuje sumiranje i konsolidaciju podataka
5. Koristi se za čitanje podataka iz data warehouse-a
6. Veličina baze varira od 100 GB do 1 TB
7. Visoka fleksibilnost, ali ne i brzina
8. Na milione rekorda
9. Transakcije se obavljaju u tačno određeno vreme
10. Koriste se od strane analitičara i menadžera

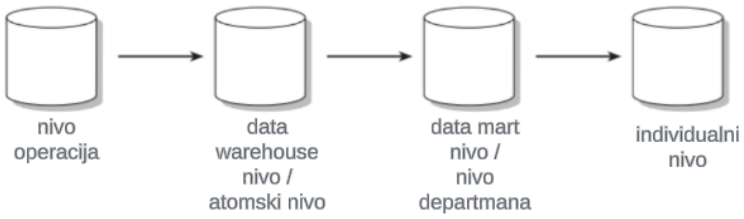
3.2 Primitivni i izvedeni podaci

Kao prvu stavku prilikom razmatranja zahteva za informacijama, DSS analitičari su otkrili da je korišćenje postojećih sistema najgori mogući scenario. DSS analitičari se prilikom analize susreću sa velikim brojem nasleđenih aplikacija. Banka može imati veliki broj odvojenih aplikacija za potrebe klijenata. Na primer sistem za plaćanje karticama, za upravljanje računima, za online kredite i hipoteke, za analizu odobrenja zahteva za kreditom itd. Međutim, neophodno je kreirati sistem koji će na mesečnom nivou obezbediti vizualizaciju prihoda i rashoda klijenata, radi dalje analize novodobijenih podataka. Izvlačiti informacije iz sistema koji su sačinjeni iz različitih aplikacija nije najbolje rešenje. Takav sistem nije bio kreiran da vrši integraciju podataka. Takođe, nije samo integracija podataka problem ovakvih sistema. DSS analitičari razmatraju istorijske podatke, kako bi njihova rešenja bila adekvatna zahtevima. Ovakvi sistemi obično ne sadrže istorijske podatke. Zbog toga nije čudno što DSS analitičari pribegavaju korišćenju data warehouse rešenja. Razlika koja se uočava između podataka je ogromna. Razlikujemo primitivne i izvedene podatke. Razlike jesu sledeće:

1. Primitivni podaci su detaljni podaci kojima se kompanije koriste u svakodnevnim operacijama. Izvedeni podaci predstavljaju sumirane, proračunate podatke koji zadovoljavanju potrebe menadžmenta kompanije.
2. Primitivni podaci mogu biti ažurirani. Izvedeni podaci mogu biti ponovo izračunati ali ne mogu biti ažurirani.
3. Primitivni podaci predstavljaju trenutne podatke, odnosno podatke nekog trenutnog stanja. Izvedeni podaci su istorijski podaci.
4. Primitivni podaci nastaju kao rezultat procesa koji se ponavljaju. Izvedeni podaci posledica su istraživanja.
5. Primitivni podaci podržavaju rad korisnika koji koristi aplikaciju, odnosno bankar u banci. Izvedeni podaci podržavaju rad menadžera koji je zahtevao podatke kako bi odgovarajuće vizualizacije bile moguće.

Zbog vidljive razlike, primitivni i izvedeni podaci ne nalaze se u istim skladištima. I jedni i drugi imaju svoju svrhu ali u različitim sistemima. Podaci nisu manje ili više vredni. Oni samo jesu nastali za sisteme koji rešavaju različite probleme, kao takvi oni imaju svoju svrhu.

3.3 Nivoi podataka



Slika 3. Nivoi podataka

Postoje 4 nivoa podataka u arhitekturnom okruženju. Nivo operacija, data warehouse nivo (atomski nivo), data mart nivo i individualni nivo. Postoje neka uverenja da ovako posmatrani nivoi podataka generišu veliku količinu suvišnih podataka. To možda nije očigledno na prvi pogled, ali to nije istinito. Baš suprotno, paukova mreža o kojoj je ranije bilo reči, odnosno prirodno razvijena arhitektura je ta koja generiše veliku količinu suvišnih podataka.

Nivo operacija može odgovoriti na pitanje "Koje je stanje na računu ovog korisnika?". Dakle u okviru nivoa operacija, postoji rekord koji sadrži neku trenutnu vrednost, ukoliko se ta vrednost iz nekog razloga izmeni, neophodno je izvršiti ažuriranje.

Data warehouse nivo sadrži istorijske podatke. Može odgovoriti na pitanje " Koje transakcije je ovaj klijent obavio u toku decembra 2023. godine?". Bitna osobina ovakvih podataka jeste ta da ne postoji preklapanje. Ukoliko, na primer taj klijent izmeni mesto stanovanja, to ne znači da će se podaci ažurirati, već će se novi rekord zabeležiti, na taj način se informacija o tome da je živeo na staroj lokaciji nije izgubila. Dakle pamtimo istoriju klijenta.

Data mart nivo, drugačije se naziva i OLAP nivo, sadrži informacije koje su korisne za različite departmane kompanije. Zato se ovaj nivo naziva i nivo departmana. Podaci koji se nalaze u ovom nivou su sumirani i prilagođeni operacijama svakog departmana pojedinačno. Tipična primena je mesečni izveštaj svakog klijenta, dakle podaci koji bi odgovarali zahtevima aplikacije moraju biti u određenom vremenskom intervalu.

Individualni nivo u arhitekturi podataka obično se odnosi na najniži nivo detalja u podacima, gde se svaki podatak odnosi na jedan pojedinačni entitet ili događaj. Na ovom nivou, podaci se često odnose na specifične instance ili pojedinosti o pojedinačnim entitetima, kao što su klijenti, transakcije, ili drugi identifikovani elementi. Ovaj nivo pruža visok nivo detalja i često se koristi u analizi ili praćenju pojedinaca i njihovih aktivnosti. Na primer, u bankarskom sistemu, individualni nivo podataka može posedovati informacije o svakoj pojedinačnoj transakciji koju je izvršio klijent.

3.4 Procesi integracije

Neophodno je navesti da podaci koji prelaze u data warehouse nivo moraju biti integrisani, u suprotnom postojanje nivoa i ne bi imalo smisla. Sama integracija podataka predstavlja naporan posao. Kako bi se videle prednosti data warehouse-a neophodno je automatizovati ovaj proces. ETL – Extract Transform Load i ELT – Extract Load Transform predstavljaju softvere koji mogu automatizovati ovaj proces.

3.4.1 ETL

ETL (Izdvajanje, Transformacija, Učitavanje) je jedna od najčešćih metoda korišćenih u integraciji podataka. Proces započinje izdvajanjem podataka iz različitih izvora. Kada se podaci prikupe iz izvora, sledeći korak je transformacija. Transformacija se odvija u oblasti pripreme (staging area), gde se podaci organizuju i konvertuju u korisniji format za analizu. To može uključivati sortiranje, sumiranje, spajanje podataka iz različitih izvora itd. Poslednji korak u ETL procesu je učitavanje transformisanih podataka. Ovo je trenutak kada se podaci čuvaju i pripremaju za upotrebu u poslovnoj analizi, izveštavanju ili drugim odlukama koje se baziraju na podacima.

3.4.2 ELT

ELT (Izdvajanje, Učitavanje, Transformacija) je takođe proces korišćen u integraciji podataka. Baš kao i ETL, može prikupljati podatke iz više izvora. Glavna razlika između ova dva metoda je to što ELT prvo učitava podatke pre nego što ih transformiše. Za razliku od ETL-a, ELT koristi procesorsku snagu modernih skladišta podataka za transformaciju podataka. ELT je posebno koristan kada se radi sa ogromnim podacima zbog paralelne obrade. Proces uključuje razbijanje podataka na manje delove i slanje različitim čvorovima u skladištu podataka radi istovremene transformacije, što je brže i efikasnije.

3.4.3 Razlike i sličnosti između ETL-a i ELT-a

Glavna razlika između ETL i ELT je redosled u kojem učitavaju i transformišu podatke. ETL je više sekvencijalan, dok ELT radi na paralelan način. Razlike između ova dva metoda jesu:

1. Proces obrade podataka

Sa sekvencijalnim procesom ETL-a, podaci moraju proći kroz svaki korak, što dovodi do dužeg perioda čekanja pre nego što postanu dostupni za analizu. Takođe zahteva značajne računarske resurse jer će vam biti potreban zaseban server za proces transformacije. Takođe, ETL procesi mogu postati složeniji i resursno zahtevni pri radu sa nestrukturiranim ili polu-strukturiranim podacima. S druge strane, ELT-ova upotreba paralelne obrade omogućava efikasnije rukovanje velikim skupovima podataka. Moderna skladišta podataka korišćena u ovoj metodi mogu rukovati masivnim količinama podataka i složenim transformacijama, tako da možete brzo dobiti tačne podatke.

2. Arhitektura i infrastruktura

Kod ETL-a mora postojati dodatna infrastruktura koja će omogućiti proces transformacije. Kao takav, uključuje postavljanje servera i baza podataka specifično za tu svrhu. Zavisno o potrebama i resursima vaše organizacije, server može biti fizički server ili virtualna mašina. S druge strane, ELT, posebno u modernim, cloud sistemima, koristi snagu data warehouse-a za transformaciju na licu mesta. Ovo obično dovodi do jednostavnijeg postavljanja u poređenju s ETL-om.

3. Performanse i skalabilnost

Mnoge organizacije se odlučuju za ETL kada rukuju strukturiranim podacima i manjim skupovima podataka. Može pružiti pouzdane i dosledne performanse putem batch

procesiranja. Međutim, može se suočiti s izazovima pri rukovanju velikim količinama podataka, što vaša organizacija možda na kraju treba obraditi. S druge strane, ELT je dizajniran za skalabilnost. Može raditi sa velikim skupovima podataka i nestrukturiranim podacima sa lakoćom. Što se tiče performansi, ELT uglavnom zavisi od sposobnosti vašeg skladišta podataka. Kao takav, ne garantuje da će ELT uvek biti brži od ETL-a.

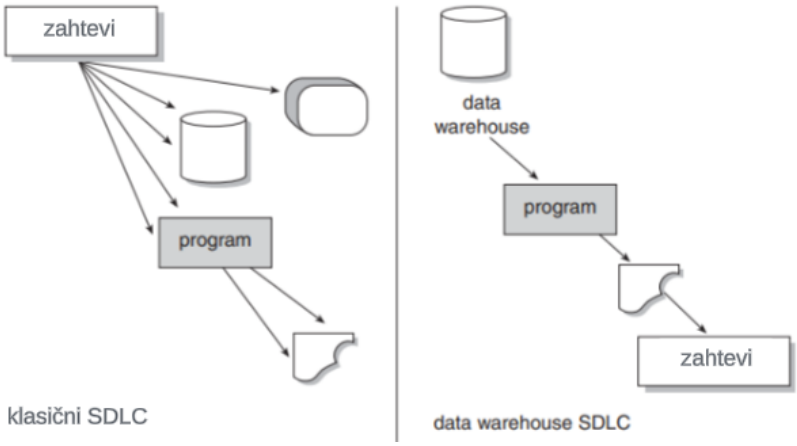
4. Mogućnosti transformacije podataka

ETL pruža obimne i kompleksne mogućnosti transformacije, što je idealno za scenarije koji zahtevaju detaljnu i komplikovanu manipulaciju podacima. Na primer, možete koristiti ETL za izvođenje kompleksnih računica i generisanje izvedenih polja na osnovu podataka iz različitih izvora. ELT je više pogodan za jednostavne potrebe transformacije jer se oslanja na procesorsku snagu skladišta podataka. Može rukovati uobičajenim zadacima poput filtriranja podataka, agregacije ali ne nudi toliko ugrađenih mogućnosti transformacije kao ETL.

5. Obrada podataka u realnom vremenu

Kada je potrebna obrada podataka u realnom vremenu radi brzog donošenja odluka, treba se opredeliti za ELT, koji pruža gotovo trenutnu dostupnost podataka. ETL-u treba više vremena da pripremi podatke za upotrebu jer mora završiti svaki korak pre nego što podaci postanu dostupni za analizu.

3.5 CLDS



Slika 4. Korišćenje SDLC i CLDS

Za razliku od klasičnih sistema, gde funkcioniše SDLC – Software Development Life Cycle, za DSS analitičara novi zahtevi se otkrivaju na kraju životnog ciklusa. SDLC predstavlja zahtevima vođen životni ciklus podataka. Dok obrnuto od ovog ciklusa postoji CLDS. CLDS se naziva i spiralni razvoj. Pristup spiralnog razvoja nalaže da se mali delovi skladišta podataka razviju do kraja, a zatim se drugi mali delovi skladišta razvijaju iterativno. Na slici 4. mogu se uočiti razlike između ova dva pristupa. Koraci kod SDLC-a jesu:

1. Planiranje
2. Analiza
3. Dizajn

4. Implementacija
5. Testiranje
6. Održavanje

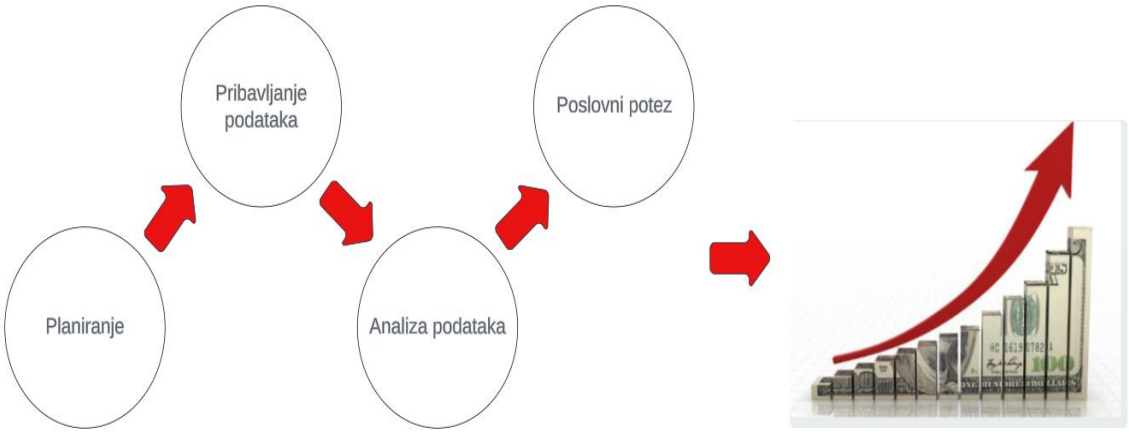
Faze kod CLDS jesu sledeće:

1. Implementacija warehouse-a
2. Integracija podataka
3. Testiranje podataka
4. Pisanje programa u skladu sa podacima
5. Dizajniranje DSS sistema
6. Analiza rezultata
7. Razumevanje zahteva

Korisnici data warehouse imaju potpuno drugačiji pristup korišćenja sistema. Oni ne zadaju zahteve na početku. Korisnici data warehouse se vode logikom : " Daj mi ono što sam ti rekao da želim i onda ti ja mogu reći šta stvarno želim".

4. Data Warehouse

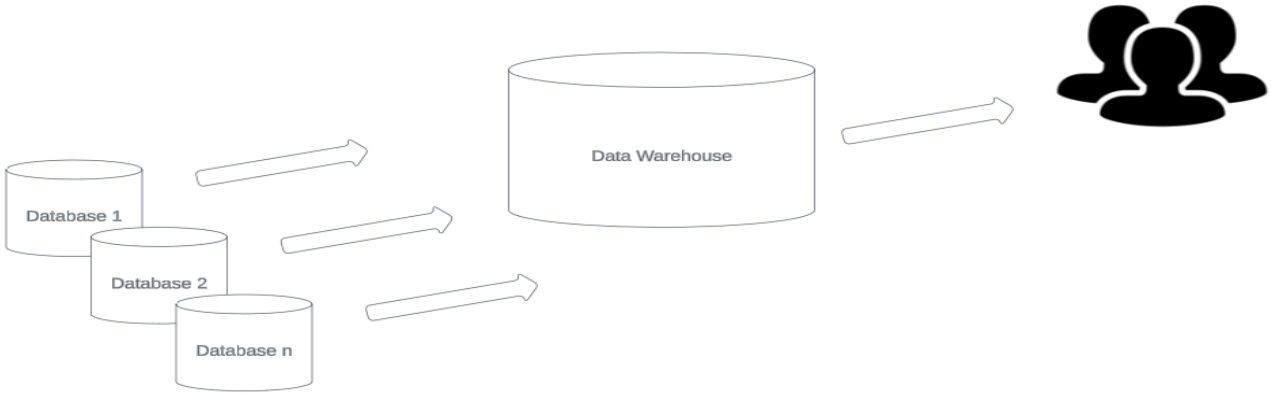
4.1 Business Intelligence



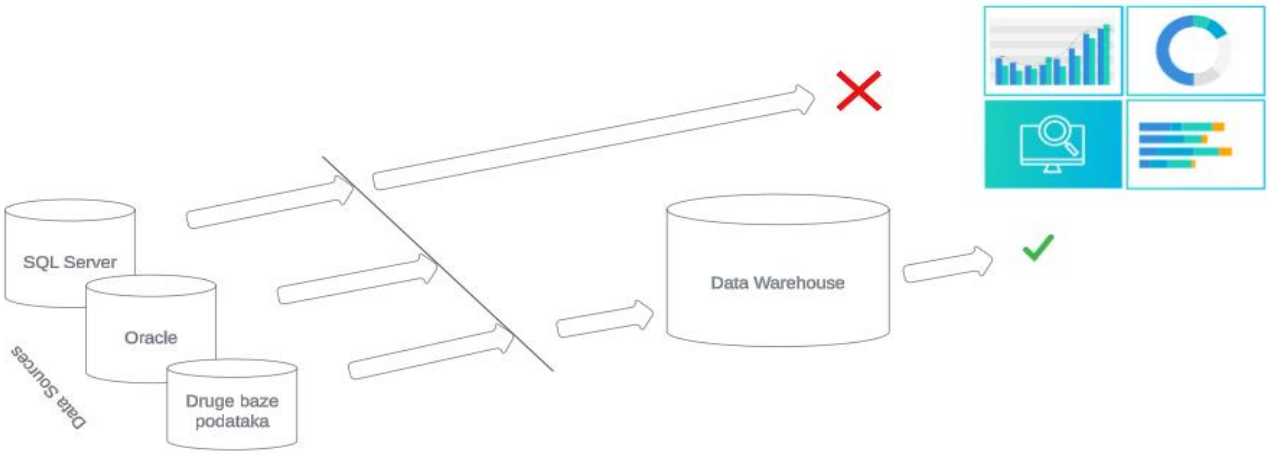
Slika 5. Koraci u kreiranju poslovnog koraka

Velike kompanije, sve više vremena i novca ulažu u Business Intelligence i Data Warehouse. BI – Business Intelligence, odnosno na srpskom poslovna logika doprinosi rastu i razvoju kompanije. U najrealnijim situacijama, iskustvo pokazuje da je svaka velika ideja je u osnovi bila mala i nije imala razmere koje će možda u nekim trenucima dobiti. Zato što sami kreatori u najvećem broju slučajeva kreću sa nekim realnim ciljevima, vremenom, kako se posao širi, postoje potrebe za većim ciljevima i samim tim i resursima. Poslovna logika raste i dobija neke nove dimenzije. Kao što je na slici 5. prikazano, kreće se od planiranja ideja, za ostvarenje bilo kog cilja, neophodni su podaci, zbog toga je naredni korak pribavljanje podataka. Kada imamo neophodne podatke sa tim podacima rade se odgovarajuće analize koje dovode do poslovne akcije. Dakle, BI predstavlja akciju transformisanja podatka u korisnu informaciju za poslovnu analizu.

Kada razumemo BI, možemo da razumemo i data warehouse. Proces analize podataka u data warehouse-u, funkcioniše tako što se, pre svega, ekstraktuju podaci iz određenih izvora.



Slika 6. Prikaz slanja podataka iz različitih izvora u data warehouse



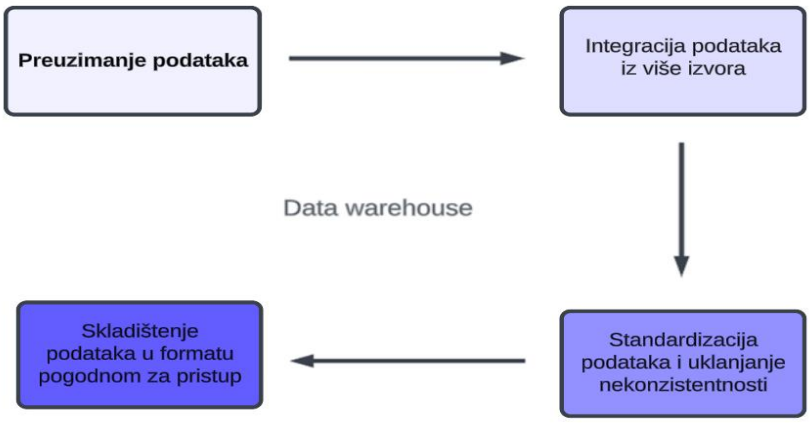
Slika 7. Prikaz korisnosti postojanja data warehouse za vizualizaciju podataka

Kao što se može videti na slici 6. podaci se šalju u data warehouse. Naravno pre nego što se podaci iz različitih izvora smeste u data warehouse neophodno je podatke transformisati. Već pomenuti ETL ima svoju svrhu u ovom trenutku.

S obzirom da podatke imamo sačuvane u različitim izvorima, može se postaviti pitanje da li je neophodno kreirati data warehouse. Podatke koji su neophodni za vizualizaciju korisnici mogu koristiti isključivo iz data warehouse, podatke za vizualizaciju ne možemo koristiti iz pojedinačnih izvora, prikazano na slici 7. Dakle data warehouse predstavlja centralizovano mesto gde se podaci skladište.

Korisnici pristupaju date warehouse-u kad god im je to neophodno. Data warehouse se ne učitavaju uvek kada se podaci dodaju u bazu. Dakle, podaci se nalaze u različitim bazama, te podatke uz pomoć ETL-a prikupljamo u data warehouse. Iz data warehouse-a koristimo OLAP tako da su podaci pripremljeni za krajnjeg korisnika.

4.2 Koraci u kreiranju Data warehouse-a



Slika 8. Koraci u okviru data warehouse-a

Data warehouse nije produkt koji kompanija kupuje, već se dizajnira i zavisi od uslova koje kompanija postavlja, u skladu sa uslovima poslovanja. Data warehouse nije produkt, već strategija koju bi trebalo prilagoditi kako bi podaci bili pogodniji, što i predstavlja glavnu prednost celog koncepta data warhouse.

Na slici 8. vidimo korake koji se odvijaju u okviru data warehouse-a. Prvi korak u okviru koga se iz različitih izvora, baza podataka, master fajlova, izvlače podaci. Nakon agregacije podataka, podaci se integrišu. Obavlja se transformisanje podataka onako kako je to na osnovu dizajna to neophodno uraditi. Integracija podataka se odnosi na proces spajanja, povezivanja i usklađivanja podataka iz različitih izvora kako bi se stvorila sveobuhvatna i koherentna slika informacija. Ovaj proces omogućava organizacijama da efikasno koriste podatke iz različitih sistema, baza podataka i izvora kako bi donosile odluke i poboljšale ukupne poslovne operacije. Podaci se dalje standardizuju i uklanja se nekonzistentnost. Podaci mogu biti nekonzistentni s obzirom da mogu dolaziti iz različitih izvora. Kao primer, navodi se da je podatak koji se odnosi na entitet ime klijenta, u jednom skladištu zapamćeno kao “ime” dok u drugom kao “ime_korisnika”. Kao završni korak podaci se skladište u formatu koji je pogodan za pristup.

4.3 Karakteristike data warehouse-a

Posao DSS analitičara kada postoji data warehouse je neuporedivo lakši u poređenju s klasičnim nasleđenim okruženjem, jer postoji jedinstven izvor podataka i zato što su granularni podaci u skladištu podataka lako dostupni. Neki od aspekata vezanih za data warehouse o kojima mora biti reči jesu ti da je data warehouse podaci predmetno orijentisani, integrisani, nepromenljivi i postoji vremenska varijabilnost podataka.

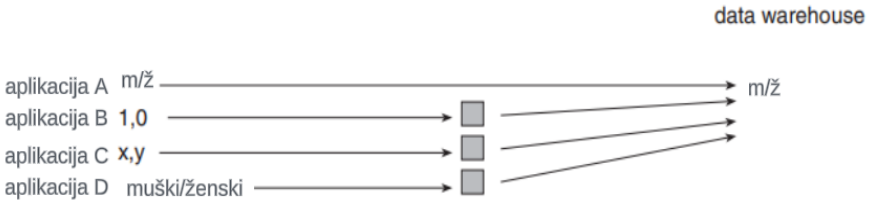
1. Subjektno orijentisani podaci

U klasičnim sistemima uočava se da je sve organizovano oko aplikacija kompanije. Ako govorimo o osiguravajućim kućama, aplikacije bi mogle biti vezane za auto, zdravstveno osiguranje, životno osiguranje i osiguranje od štete. Glavne oblasti subjekta u osiguravajućem društvu mogu biti klijent, polisa, premija i potraživanje. Za proizvođača, glavne oblasti

subjekta mogu biti proizvod, narudžbina, dobavljač, lista materijala i sirovine. Za prodavca, glavne oblasti entiteta mogu biti proizvod, prodaja, dobavljač, i tako dalje. Svaki tip kompanije ima svoj jedinstven skup entiteta, predmeta odnosno subjekta.

2. Integrisani podaci

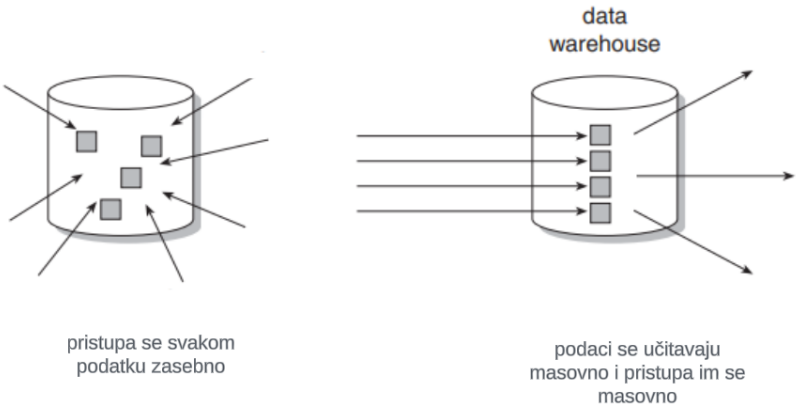
Druga veoma istaknuta karakteristika data warehouse-a jeste ta da se radi o integrisanim podacima, baš zbog toga što se podaci pre nego što se skladište u data warehouse-u nalaze u različitim bazama, a zatim se smeste na jedinstveno mesto. Ne radi se samo o skladištenju, nego o svemu što skladištenje prati. Podaci koji su pribavljeni se konvertuju, sumiranju, predefinišu itd. Primer za to može biti oznaka pola, ukoliko imamo 4 aplikacije kao na slici 9. U aplikacijama A, B, C i D, podaci o polu se drugačije beleže. U data warehouse-u postoji podatak o polu, ali je izabrana jedna od oznaka.



Slika 9. Podaci iz različitih izvora se integrišu

3. Nepromenljivost podataka

Treća važna karakteristika skladišta podataka je da je nepromenljivost podataka. Slika 10. ilustruje nepromenljivost podataka i pokazuje da se podacima u realnim okruženjima redovno pristupa i njima se manipuliše. Podaci se ažuriraju kao redovan deo procesa. Podaci u skladištu podataka se učitavaju (obično masovno) i pristupa im se, ali se ne ažuriraju (u opštem smislu). Umesto toga, kada se podaci u data warehouse-u učitaju, učitavaju se u obliku rekorda. Kada se kasnije dese promene, zapisuje se novi rekord. Na taj način se održava istorija podataka u data warehouse-u.



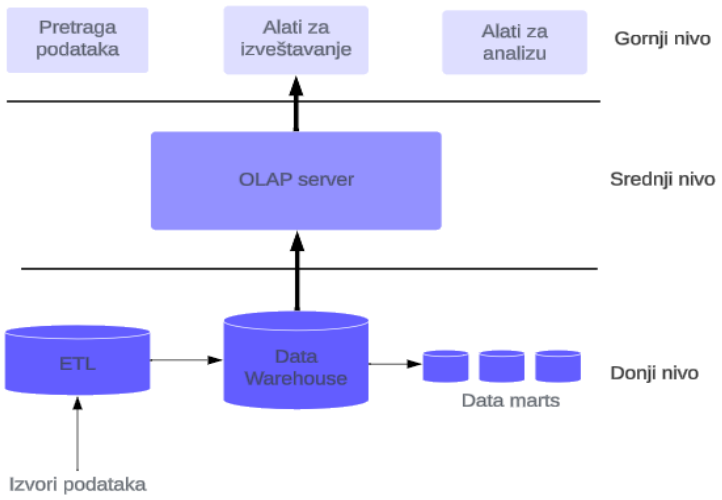
Slika 10. Učitavanje i čitanje podataka u običnim okruženjima i data warehouse-u

4. Vremenska varijabilnost podataka

Poslednja bitna karakteristika data warehouse-a je vremenska varijabilnost. Svaki rekord jeste tačan u određenom trenutku u vremenu. U nekim slučajevima, rekordima je dodata

vremenska markica. U drugim slučajevima, postoji datum transakcije.

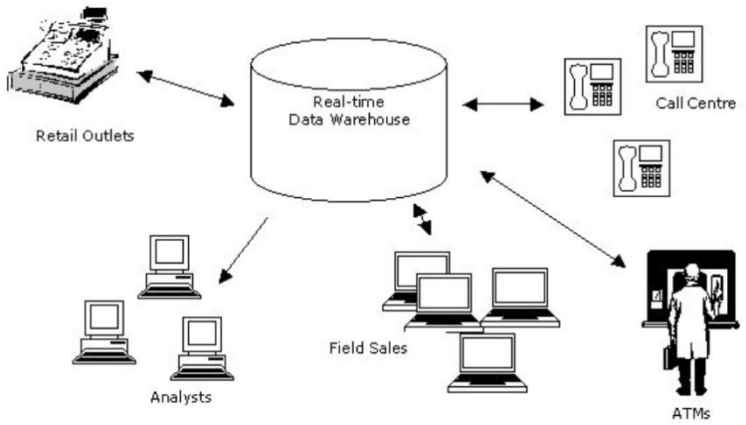
4.4 Arhitektura tri nivoa data warehouse-a



Slika 11. Arhitektura tri nivoa data warehouse-a

Data warehouse je dizajniran da daje smisao podacima. U tehničkom smislu, data warehouse je informacioni sistem koji se koristi da skladišti i da organizuje podatke iz različitih izvora kako bi se mogli dobiti korisni poslovni uvidi. Tradicionalne baze podataka, kao što su MySQL i MongoDB su odlične za svakodnevne operacije. Kada analiza podrazumeva velike količine podataka ovi sistemi postaju prilično spori i neefikasni. Zbog toga na scenu stupa data warehouse. Data warehouse je dizajniran za analizu, a ne transakcije. Analiza se efikasno sprovodi transformacijom podataka u korisne informacije koje su dostupne korisnicima. Data warehouse je odvojen od ostalih baza podataka kompanije i dozvoljava pristup trenutnim i istorijskim podacima koji se koriste kako bi se donosile odluke. Kada dođe do analize podataka, data warehouse štedi vreme, poboljšava performanse u realnom vremenu, smanjujući vreme odgovora i poboljšavajući performanse upita. Postoje različite varijante arhitekture, ali najrasprostranjenija odnosno najčešća je arhitektura tri nivoa, prikazana na slici 11. Ova arhitektura sadrži: Donji nivo (Bottom Tier), koji predstavlja skladište podataka, zatim Srednji nivo (Middle Tier), koji predstavlja OLAP server odnosno Online Analytical Processing. Poslednji nivo je Gornji nivo (Top Tier), koji predstavlja front-end klijentskog nivoa. U ovoj arhitekturi, Donji nivo skladišti isprocesirane i transformisane podatke, dok srednji nivo prezentuje apstraktni pogled baze podataka krajnjem korisniku. Gornji nivo obezbeđuje pristup podacima kroz alate, alati za upite, alati za obaveštavanje i alati za analizu. Obezbeđuju konzistentnost različitih tipova podataka iz različitih izvora, pri tom podaci su tačni i nisu podložni ažuriranju.

5. Real-time data warehouse



Slika 12. Prikaz uticaja aplikacija u realnom vremenu na RTDW

5.1 Pojam real-time data warehouse

Slika 13. Prikaz slanja podataka iz izvora u RTDW

Real-time data warehouse (RTDW) dozvoljava procesiranje podataka u realnom vremenu, dajući na taj način efikasan i brz uvid u poslovne operacije. Zahvaljujući tome postoji uvid u najsvežije podatke. Nije neophodno čekanje na pojedine podatke ili rad sa podacima koji više nisu važeći. Sa real-time data warehouse–om ostaje se na vrhuncu zadatka i prave se prave odluke. Dakle, podaci koji se u nekom sistemu trenutno generišu, šalju se automatski i u real-time data warehouse, prikaz na slici 12.

Tajna brzog procesiranja je korišćenje real-time data pipelines. Postoje open-source rešenja za kreiranje real-time data pipeline, na primer rešenje koje je skalabilno je Estuary koje obezbeđuje grafički baziran alat. Pipelines na brz i efikasan način vrši transfer podataka iz različitih izvora u data warehouse, dajući pristup svim podacima na jednom, centralizovanom mestu, prikaz na slici 13. Važno je uočiti trenutak kada je neophodno poslati podatke u RTDW. Bitan je način na koji se podaci iz izvora šalju u RTDW. Takođe, bitno je u kom se obliku ti podaci prosleđuju. Kao što je na slici 13. naglašeno radi se o serijama (batches). Real-time data warehouse zatim procesira ove podatke brzo i dozvoljava da se prave odluke bazirane na najnovijim podacima. Podaci stižu u data warehouse brže i transformišu odmah kako bi učinili upite efikasnijim. Upiti se izvršavaju velikom brzinom. Ukoliko podaci sadrže grešku koju je neophodno ispraviti, potrebno je da se to uradi odmah,

pre čuvanja podataka. Jedna od bitnih stvari za naglasiti je da je RTDW teško održavati ali je svakako uz odgovarajuće alate to moguće izvesti na efikasan način.

5.2 Razlike između data warehouse i RTDW

Tradicionalni data warehouse i real-time data warehouse možda zvuče isto ali imaju neke ključne razlike. Tradicionalni data warehouse skladišti istorijske podatke jedne organizacije. Ti podaci koriste se u donošenju odluka. Kako god tradicionalni data warehouse predstavlja stabilan pogled na podatke organizacije koji se mogu sakupiti iz različitih izvora, analiza zasnovana na podacima predstavlja pogled na neki prošli period, dan, nedelju, mesec, godinu u okviru kojeg su podaci učitani. RTDW ide korak dalje. Pored toga što su istorijski podaci dostupni, RTDW zadovoljavaju potrebu za real-time podacima. Podaci u RTDW-u se osvežavaju konstantno, na taj način obezbeđuju realnu sliku o trenutnom stanju podataka organizacije. Zahvaljujući tome moguće je doneti strategijske i taktičke odluke.
Ako upoređujemo data warehouse i RTDW, neke ključne razlike jesu:

1. Tradicionalni data warehouses skladišti podatke periodično dok real-time data warehouse skladišti podatke u realnom vremenu.
2. Tradicionalni data warehouse mogu prihvatati na dnevnom, nedeljnom, mečnom, godišnjem nivou dok real-time data warehouse prihvataju podatke u realnom vremenu koji su dostupni za nekoliko minuta bez potrebe za velikim izmenama procesa učitavanja ili modela podataka.
3. Tradicionalni data warehouse može jedino da se koristi za dugoročne odluke dok RTDW može se koristiti za dugoročne i kratkoročne odluke.
4. Pošto su podaci u RTDW svežiji nego u data warehouse-u, poslovne odluke koje se donose su relevantnije.
5. RTDW zahteva kontinuirano ažuriranje podataka, bez gašenja baze.

5.3 Pristup u kreiranju RTDW-a

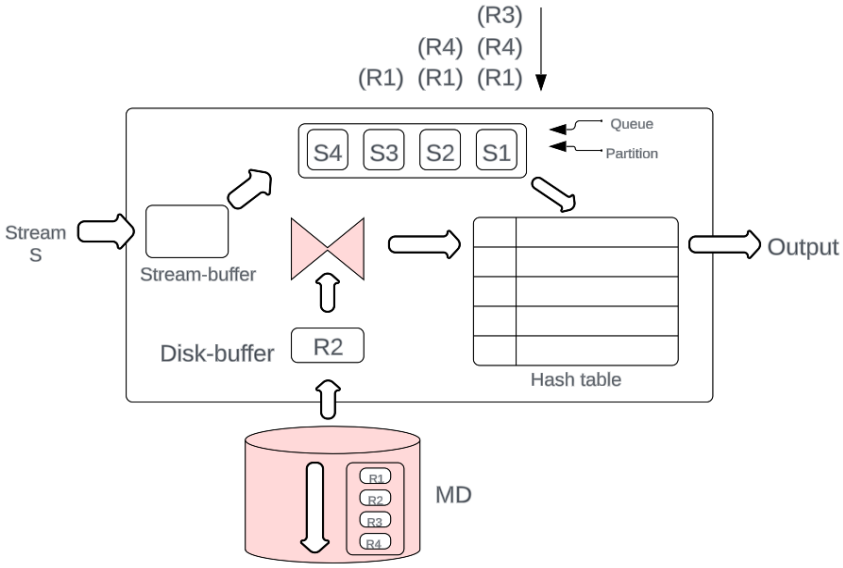
Da bi kreirali RTDW neophodno je implementirati real-time ETL ili barem ETL koji će približno u realnom vremenu funkcionisati. Dešava se da su pojedini podaci nepotpuni, kada dođu do ETL-a podaci se u velikom broju slučajeva upotpune podacima iz master fajlova.

Pristup u kreiranju RTDW-a:

1. Razumevanje zahteva projekta: Prvi korak u kreiranju RTDW jeste proći kroz zahteve i razumeti ih, uključujući vremenski rok, ciljeve i rezultate. Važno je obratiti pažnju na meshjoin algoritam, zvezdasti model šeme i detaljno pregledati specifikaciju podataka.
2. Postavljanje razvojnog okruženja: Ovaj korak uključuje instalaciju i konfiguraciju neophodnog softvera, kao što je instalacija integrisanog razvojnog okruženja i baze podataka. Takođe, kreiraju se neophodni folderi i fajlovi.
3. Implementacija meshjoin algoritma: Ovo uključuje pisanje koda kako bi rezervisali memoriju, uskladištili transakcije kupaca u heš tabele, organizovali dolazeće podatke u red i istražili poklapanje torki za generisanje izlaza. Ovo je značajno za testiranje korektnosti.
4. Dizajniranje zvezdaste šeme: Zvezdasta šema je korišćena za mapiranje multidimenzionalnih podataka u relacionu bazu podataka. U ovom koraku dizajnira se šema zahtevima projekta, uključujući činjenice i atribute. Koristi se šema kako bi se kreiralo više izvora podataka koji predstavljaju različite aspekte poslovnih operacija.
5. Kreiranje i popunjavanje baze podataka: U ovom koraku, kreira se baza podataka.

Takođe se popune tabele podacima prema specifikacijama.

6. Izgradnja prototipa skladišta podataka u realnom vremenu: Koristeći meshjoin algoritam i zvezdasti model šeme, izgradi se prototip RTDW-a. Ovo uključuje implementaciju alata za ekstrakciju, transformaciju i učitavanje podataka u skladu sa real-time zahtevima, odnosno odmah po pojavljivanju transakcija u izvorima podataka. Testira se prototip skladišta podataka temeljno kako bi osigurali da ispunjava zahteve projekta i pruža željene rezultate.
7. Analiza prototipa skladišta podataka: U ovom koraku, analizira se prototip skladišta podataka kako bi se ocenila njegova efikasnost. Koriste se alati poput vizualizacije podataka i izveštavanja kako bi se izvukli smisleni uvidi iz skladišta podataka.

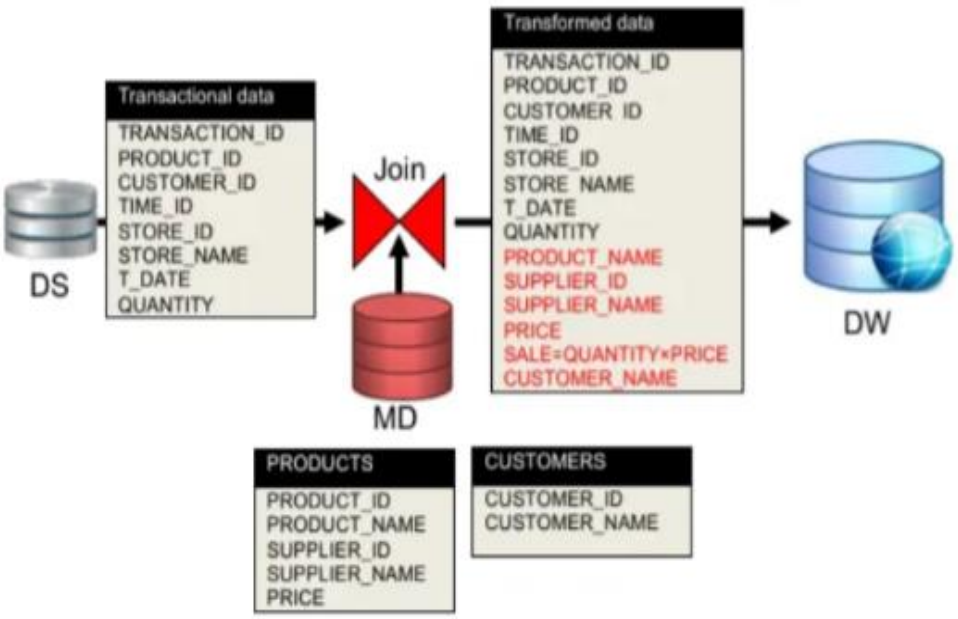


Slika 14. Slikovit prikaz meshjoin algoritma

Meshjoin algoritam koristi se u fazi transformacije u ETL-u. Kreiran je 2008 godine sa ciljem implementacije operacije spajanja. Operacija spajanja "Stream-Relation join" predstavlja proces povezivanja podataka koji neprestano pristižu (stream podaci) sa podacima koji su statični i smešteni u relacijskoj bazi podataka (relation podaci).
Glavne komponente meshjoin-a jesu bafer na disku i bafer u radnoj memoriji. Bafer na disku će biti red koji se koristi za čitanje torki koje nadolaze. Bafer u radnoj memoriji koji predstavlja heš tabelu i koji koristi učitane torke koje će biti spojene sa nadolazećim torkama. Mesh, koji je graf struktura koja mapira nadolazeće torke sa torkama iz bafera u radnoj memoriji. Join uređaji, koji su odgovorni za izvođenje join operacija između nadolazećih torki iz bafera na disku i torki relacije koji su im mapirani pomoću mesh strukture.
Meshjoin algoritam prati sledeće:

1. Nadolazeće torke se smeštaju u bafer na disku.
2. Torke koje će biti spojene sa nadolazećim se smeštaju u bafer u radnoj memoriji.
3. Mesh je konstruisan heširanjem torki u bafer u memoriji sa odgovarajućim čvorovima u Mesh strukturi.
4. Nadolazeće torke su procesirane i mapirane u Mesh čvorovima.
5. Join uređaj koristi Mesh kako bi identifikovao torke koji se mogu spojiti sa nadolazećim torkama.
6. Join uređaj izvodi operacije između nadolazećih i identifikovanih torki.

7. Spojene torke se šalju na izlaz.



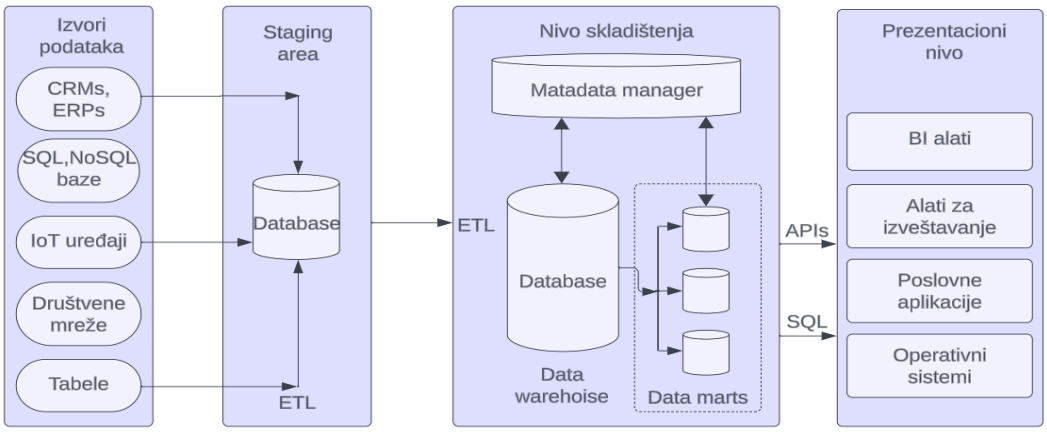
Slika 15. Primer bogaćenja

Zvezdasta šema je tehnika modelovanja podataka koja se koristi da mapira višedimenzionalne podatke u relacione podatke. Zvezdasta šema doprinosi jednostavnijoj implementaciji modela za višedimenzionalnu analizu podataka koja čuva relacijske strukture na kojima se zasniva operativna baza podataka. Zvezdasta šema predstavlja agregaciju podataka za specifične poslovne aktivnosti. Agregacija može uključivati različite nivoe hijerarhije produkta. Nakon ETL i DW analize naredni korak je bogaćenje podataka dodavanjem dodatnih informacija kako bi proširili kvalitet podataka i korisnost.
Naredni koraci obavljaju se u fazi obogaćenja:

1. "Čišćenje" podataka kako bi se uklonili duplikati, vrednosti koje nedostaju i nekonzistentnosti.
2. Transformacija podataka kako bi se kreirale nove vrednosti ili modifikovale postojaće.
3. Integracija podataka radi kombinovanja podataka iz više izvora.

Bogaćenje podataka može pomoći da se otkriju neotkriveni izvori, ispravi tačnost podataka i obogati sposobnost donošenja odluka.

5.4 Arhitektura RTDW



Slika 16. Arhitektura RTDW-a

Ključne komponente i tehnologije koji sačinjavaju arhitekturu RTDW-a:

1. Uzimanje podataka**:** Uzimanje podataka predstavlja prvi nivo. Uključuje podatke sa različitih IoT uređaja, baza podataka, web sajtova itd. Prikupljeni podaci se skladište privremeno u staging zoni (oblast za pripremu podataka) pre nego se procesiraju i učitaju u data warehouse.
2. Procesiranje podataka**:** Drugi nivo je procesiranje podataka koji uključuje transformaciju podataka u strukturni format koji je pogodan za analizu. Ovo uključuje "čišćenje", transformaciju i izmenu podataka u skladu sa formatom koji se neophodan.
3. Skladište podataka**:** Nakon procesiranja, podaci se skladište. RTDW obično koriste arhitekturu u vidu kolona koja omogućava brzo i efikasno ispitivanje velikih količina podataka. Za razliku od ključ-vrednost baza podataka, u kolonskim bazama podataka podacima se pristupa po kolonama, a ne po vrednostima. Obrada po kolonama prilikom paralelnog procesiranja doprinosi boljim performansama.
4. Vizualizacija podataka**:** Finalni korak je vizualizacija koja uključuje prezentaciju podataka za zainteresovane strane. Što uključuje čartove, grafike i kontrolne table za analizu i uvid u podatke.

5.5 Real-time integracija podataka

Neophodno je da integracija podataka u realnom vremenu bude bez latencije za neke slučajeve ili da ona bar bude minimalna. Različite arhitekture za prikupljanje podataka iz operativnih izvora mogu se koristiti za popunjavanje data warehouse-a. Ove tehnike variraju najviše u zavisnosti od latentnosti integracije podataka, od svakodnevnih do real-time integracija. Prikupljanje podataka iz izvora se vrši ili putem inkrementalnih upita koji obavljaju filtriranje na osnovu vremenske oznake ili markice ili putem mehanizama koji prate promenu podataka. Arhitekture se dalje dele između pull i push operacija, gde pull operacije povlače u fiksnim intervalima, dok push operacije učitavaju podatke kada se pojave promene.

Dnevni mehanizam je najpogodniji kada je reč o podacima koji se računaju samo jednom

dnevno, na primer informacija o dnevnom pazaru u nekoj radnji. Učitavanje serije može se obaviti u vremenskom periodu u kom sistem ne funkcioniše, naravno ovo je moguće samo ukoliko poslovna logika ne zahteva dostupnost podataka 24 sata dnevno. Različite tehnike, postoje kako bi minimizovale uticaj učitavanja podataka onih sistema kod kojih ne postoji vreme kada sistem ne funkcioniše.

6. Tehnologije za implementaciju RTDW-a

6.1 Pristupi implementacije RTDW

Postoje različiti pristupi kojima možemo implementirati RTDW, neki od njih jesu:

1. Kupovina RTDW rešenja. Danas postoje veliki broj ovakvih rešenja na klaudu. Kao što su na primer Materialize, Rockset, Firebolt i drugi. Ovi sistemi pružaju ugrađene alate za analizu u realnom vremenu i sopstvene setove integracija u realnom vremenu za određene izvore podataka. Kada odaberete jedan, pobrinite se da će se moći povezati sa drugim sistemima za skladištenje podataka i alatima koje koristite.

2. S obzirom da je razlika između RTDW i data warehouse je postojanje pipeline-a. Real-time data warehouse može se implementirati dodavanjem pipeline-a. Pipeline pribavljanjem i transformisanjem podataka u realnom vremenu omogućava RTDW. Za data warehouse dobar izbor jesu Snowflake i BigQuery. Za implementiranje pipeline-a, treba izabrati alat koji je fleksibilan i podržava sve izvore podataka. Može se izabrati Estury Flow ili kreirati sopstveni pipeline sa Apache Kafka.

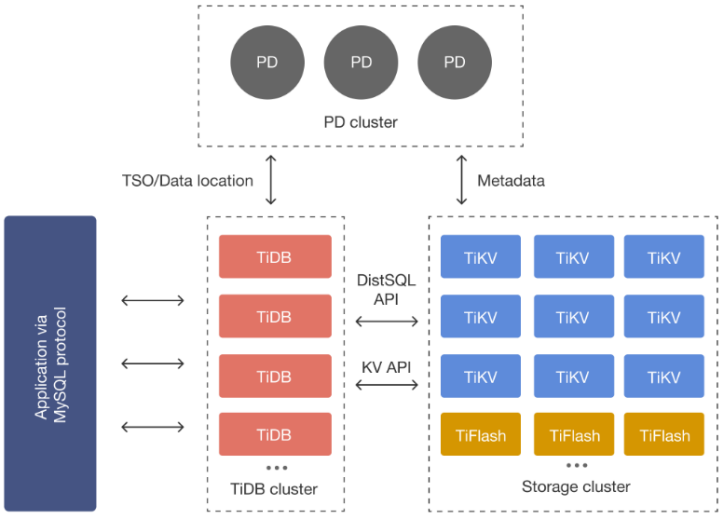
3. Kreirati sopstveni RTDW. Naravno najkomplikovanija opcija. Ali najviše prilika u izboru karakteristika koje najviše odgovaraju problemima koje RTDW rešava. Ne postoje neka striktna ograničenja prilikom kreiranja RTDW, ali kao minimum neophodno je da postoje ove ključne tehnologije:
    1. Kolonske baze podataka: RTDW tipično koriste kolonske baze podataka čija je karakteristika velika brzina izvršavanja upita i rad sa velikom količinom podataka. Primer jesu Apache Paquet , Apache Cassandra, TiDB.
    2. Platforme za strimovanje: RTDW imaju potrebu za striming platformama kao što su Apache Kafka, Estuary Flow, Pravega za prihvatanje podataka u realnom vremenu, Kreiranje Pipeline-a.
    3. Baza podataka u memoriji: Koriste se da skladište podatke u memoriji, prednost im je brzina izvršavanja upita i analiza podataka. Kao na primer SAP HANA i Oracle Times Ten.

Okruženje za real-time procesiranje podataka: Za procesiranje i analizu velike količine podataka koristi se okruženje kao što je Apache Spark.

6.2 TiDB

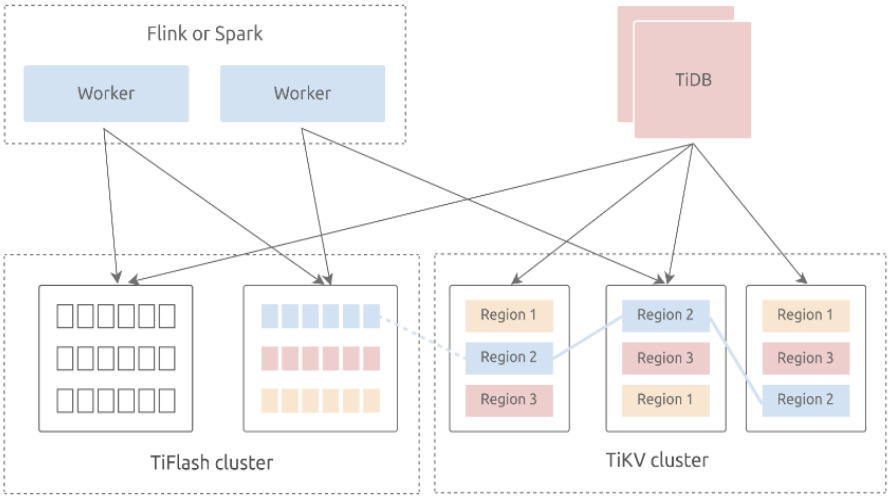
RTDW sadrži 4 komponente: nivo prikupljanja podataka, nivo skladištenja podataka, nivo proračuna u realnom vremenu i aplikativni nivo. Integrišući više tehnologija u jedinstvenu arhitekturu, može se izgraditi proširiva arhitektura za velike podatke koja podržava analizu i rudarenje podataka, online transakcije i ujedinjeno procesiranje tokova i serija. Postoje različite varijante skladištenja podataka, ali nisu sve od njih pogodne za real-time data warehouse: Hadoop ili tradicionalne baze podataka ne mogu obezbediti odgovarajuće real-time procesiranje. NoSQL solucije kao što je HBase mogu da procesiraju podatke u realnom

vremenu, ali ne mogu da obezbede analizu. Relacione baze podataka ne mogu se proširivati kako bi podržale ogromne količine podataka. TiDB sa druge strane zadovoljava sve ove potrebe.



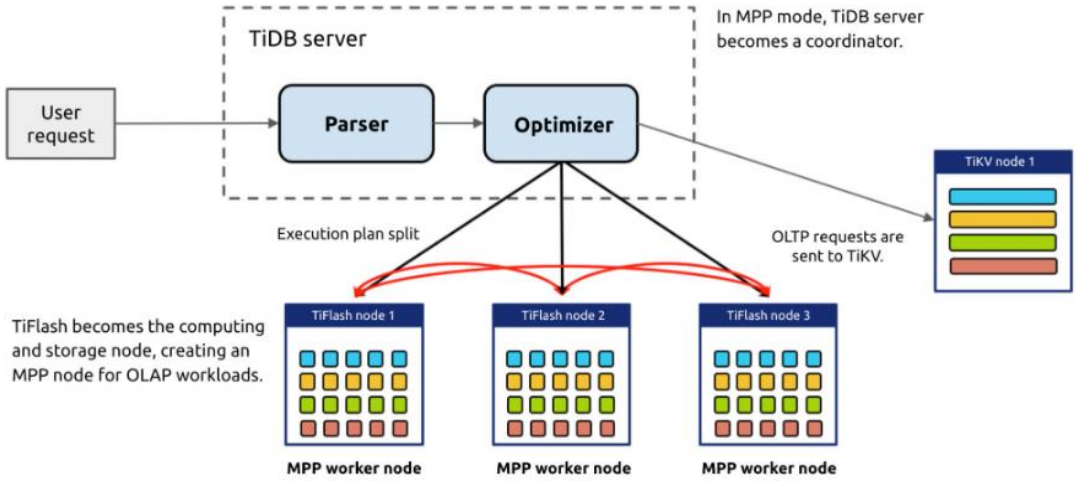
Slika 17. Prikaz arhitekture TiDB-a, klasteri koji ga sačinjavaju

TiDB (Ti - Titanium) kreiran od strane PingCAP-a, je open-source, horizontalno skalabilna distribuirana baza podataka, visoke dostupnosti i stroge konzistentnosti, koja podržava HTAP – Hybrid Transactional and Analytical Processing, predstavlja hibrid transakcionog i analitičkog procesiranja. TiDB poseduje hibridni nivo skladišta koji se sastoji od TiKV klastera za skladištenje po redovima i TiFlash klastera za skladištenje po kolonama, vidljivo na slici 18. TiDB odgovara OLAP i OLTP zahtevima i pribavlja podatke od bilo kog klastera. Postoji TiDB Cloud koji predstavlja servis TiDB-a. Trenutno dostupan na AWS-u i GCP-u (Amazon Web Service i Google Cloud Platform). PingCAP se prilikom kreiranja TiDB-a nije bavio samim skladištenjem podataka, pa je tražio rešenje koje će implementirati skladište, a to je RocksDB.



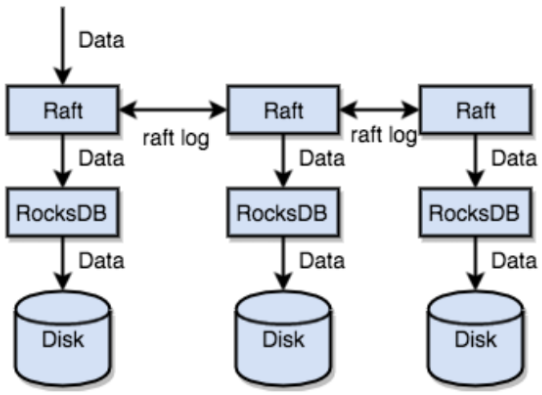
Slika 18. TiDB, prikaz TiFlash i TiKV klastera

TiDB 5.0 uvodi arhitekturu masivnog paralelnog procesiranja(MPP – Massively Parallel Processing). U MPP modu, TiFlash dopunjuje sposobnosti TiDB. Korisnici šalju zahteve TiDB serveru i svi TiDB serveri izvode spajanje tabela i potvrđuju rezultat optimizer-u za donošenje odluka. Optimizer pristupa svim mogućim rešenjima i bira najoptimalnije. Bira da li je optimalnije da se podaci pribavljaju iz TiKV ili TiFlash klastera. Podaci u TiFlash klasterima predstavljaju replike podataka u TiKV. Podaci se isključivo unose u TiKV, a zatim se na osnovu algoritama replike ažuriranja prenose u TiFlash. Bitno je naglasiti da u TiDB bazama postoje replike TiKV klastera, koje garantuju pouzdanost. TiKV klasteri podatke pamte u vidu key-value podataka, a sami podaci raspoređeni su po regionima.



Slika 19. Postojanje Optimizer-a u sistemu

Sistem za obradu porudžbina može doživeti nagli porast saobraćaja tokom rasprodaje. Za to vreme, preduzeća moraju brzo sprovoditi analizu kako bi pravovremeno reagovala i odgovorila na ponašanje korisnika. Tradicionalni skladišni prostori podataka teško se nose s naglim prilivom podataka u kratkom vremenskom periodu, i moglo bi potrajati dugo vremena da se izvrši naknadna analitička obrada podataka. Zahvaljujući MPP-u, TiDB može predvideti saobraćaj i dinamički proširiti klaster kako bi obezbedio više resursa za kampanju. Zatim može lako odgovoriti na zahteve za agregacijom i analizom u roku od nekoliko sekundi. Kada se TiDB koristi sa Pravega-om, ovo skladište podataka sposobno je zadovoljiti različite zahteve korisnika za velikim podacima i obrađivati OLTP i OLAP radne zadatke na jednom mestu.

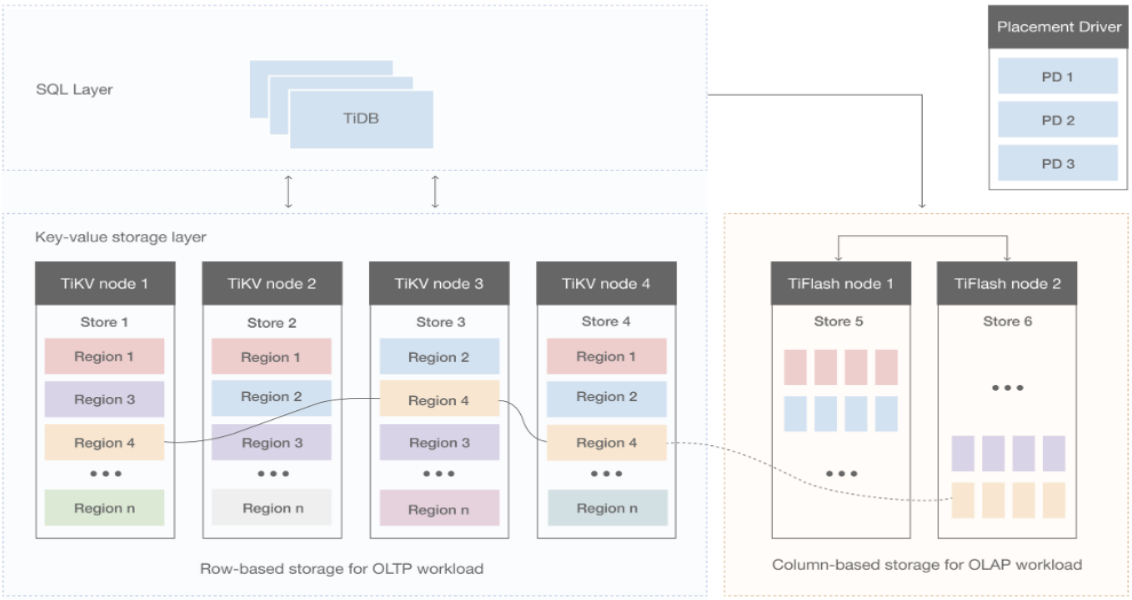


Slika 20. Prikaz više TiKV klastera, odnosno Raft region

TiKV koristi Raft za replikaciju podataka, pri čemu se svaka promena podataka beleži kao Raft log. Kroz funkciju replikacije logova Raft-a, podaci se sigurno i pouzdano sinhronizuju na više čvorova Raft grupe. Raft grupu čine isti regioni u TiKV klasterima. Raft je algoritam za postizanje saglasnosti i pruža tri važne funkcije:

1. Izbor vođe (Leader election)
2. Promena članstva (Membership change)
3. Replikacija logova (Log replication)

Bilo koji trajni mehanizam za skladištenje čuva podatke na disku, a TiKV nije izuzetak. Međutim, TiKV ne upisuje podatke direktno na disk. Umesto toga, podaci se skladište u RocksDB, a zatim je RocksDB odgovoran za skladištenje podataka. Kako bi se obezbedila netaknutost i tačnost podataka kada jedan klaster otkaže, dobro je replicirati podatke na više klastera. Tada, kada jedna mašina prestane sa radom, imamo replike na drugim klasterima. Važno je napomenuti da rešenje za replikaciju treba da bude pouzdano, efikasno i da može nositi se sa situacijom nevažeće replike. Zvuči teško, ali Raft to čini mogućim. Raft je algoritam za postizanje konsenzusa.



Slika 21. TiDB arhitektura, osvrt na prikaz Raft regiona i prikaz kopije u TiFlash klasterima

TiFlash ima sledeće ključne karakteristike:

1. Asinhrona replikacija. Replika u TiFlash-u nastaje asinhrono. To znači da kada TiFlash čvor prestane raditi ili dođe do visokog kašnjenja mreže, aplikacije u TiKV-u i dalje mogu normalno funkcionisati. Ovaj mehanizam replikacije daje dve prednosti TiKV-a: automatsko balansiranje opterećenja i visoku dostupnost. Sve dok podaci nisu izgubljeni u TiKV-u, možete u svakom trenutku obnoviti repliku u TiFlash-u.
2. Doslednost. TiFlash pruža isti nivo doslednosti (Snapshot Isolation) kao i TiKV, i obezbeđuje čitanje najnovijih podataka, što znači da možete čitati podatke koji su prethodno upisani u TiKV. Svaki put kada TiFlash primi zahtev za čitanje, replika Regiona šalje zahtev za validaciju napretka (RPC zahtev) lideru replike. TiFlash

izvršava operaciju čitanja samo ako trenutni napredak replikacije uključuje podatke obuhvaćene vremenskom oznakom zahteva za čitanje.
3. Inteligentan izbor. TiDB može automatski odabrati da koristi TiFlash ili TiKV, ili koristiti oba u istom upitu kako bi obezbedio najbolje performanse. Ovaj mehanizam selekcije sličan je onome u TiDB-u koji bira različite indekse za izvršavanje upita. Optimizator TiDB-a pravi odgovarajući izbor na osnovu statistike.

Ključne karakteristike TiDB-a:

1. Horizontalna skalabilnost. U arhitekturnom dizajnu TiDB-a, razdvajaju se proračuni od skladištenja. Dozvoljavajući da se proširi ili smanji prostor za proračune i prostor za skladištenje, po potrebi. Sam proces skaliranja je transparentan.
2. Real-time HTAP. TiDB podržava dva tipa klastera TiKV i TiFlash. Odnosno razlikujemo skladištenje po vrstama kod TiKV i po kolonama kod TiFlash. Skladištenje po vrstama značajno je za brzo skladištenje transakcija, a po kolonama za brz pristup analitičkim procesima. TiFlash koristi Multi-Raft protokol kako bi u realnom vremenu replicirao podatke iz TiKV-a, osiguravajući konzistentnost dva skladišta.
3. Visoka dostupnost. Podaci se čuvaju u više replika, a za dobijanje evidencije transakcija koristi se Multi-Raft protokol. Transakcija može biti potvrđena samo kada su podaci uspešno upisani u većinu replika. Ovo garantuje snažnu doslednost i dostupnost kada manjina replika prestane sa radom. Možete konfigurisati geografsku lokaciju i broj replika prema potrebi kako biste ispunili različite nivoe tolerancije na greške.
4. Cloud podrška. TiDB je distribuirana baza podataka dizajnirana za cloud, pružajući fleksibilnu skalabilnost, pouzdanost i sigurnost na cloud platformi. U TiDB-u, svaki podatak ima najmanje 3 replike, koje se mogu rasporediti u različitim cloud zonama dostupnosti kako bi tolerisale prekid rada celog data centra. TiDB Cloud, najlakši je, najekonomičniji i najotporniji način za otključavanje punog potencijala TiDB-a, omogućavajući vam da implementirate i pokrećete TiDB klaster sa samo nekoliko klikova.
5. Kompatibilan sa MySQL protokolom. U mnogim slučajevima, za migraciju aplikacija na TiDB, nije potrebno promeniti niti jedan red koda, ili je potrebno samo izmeniti malu količinu koda. Osim toga, TiDB pruža niz alata za migraciju podataka kako bi olakšao jednostavnu migraciju podataka aplikacije u TiDB.

Primena TiDB

1. Finansije. TiDB je idealan za scenarije u finansijskoj industriji sa visokim zahtevima za doslednošću podataka, pouzdanošću, dostupnošću, skalabilnošću i tolerancijom na greške. Tradicionalna rešenja su skupa i neefikasna, sa niskom iskorišćenošću resursa i visokim troškovima održavanja.
2. Masivni podaci i visoka konkurentnost. Tradicionalne samostalne baze podataka ne mogu zadovoljiti zahteve za kapacitetom podataka brzo rastućih aplikacija. TiDB je ekonomično rešenje koje koristi odvojenu arhitekturu za računanje i skladištenje, omogućavajući lako skaliranje računarskog ili skladišnog kapaciteta zasebno.
3. Real-time HTAP. TiDB je idealan za scenarije sa masivnim podacima i visokom konkurencijom koji zahtevaju obradu u realnom vremenu. TiDB uvodi TiFlash u verziji 4.0.
4. Agregacija podataka. TiDB je pogodan za kompanije koje trebaju agregirati rasute podatke u istom sistemu. U poređenju sa Hadoop-om, TiDB je mnogo jednostavniji. Podatke možete replicirati u TiDB koristeći ETL alate ili alate za migraciju podataka

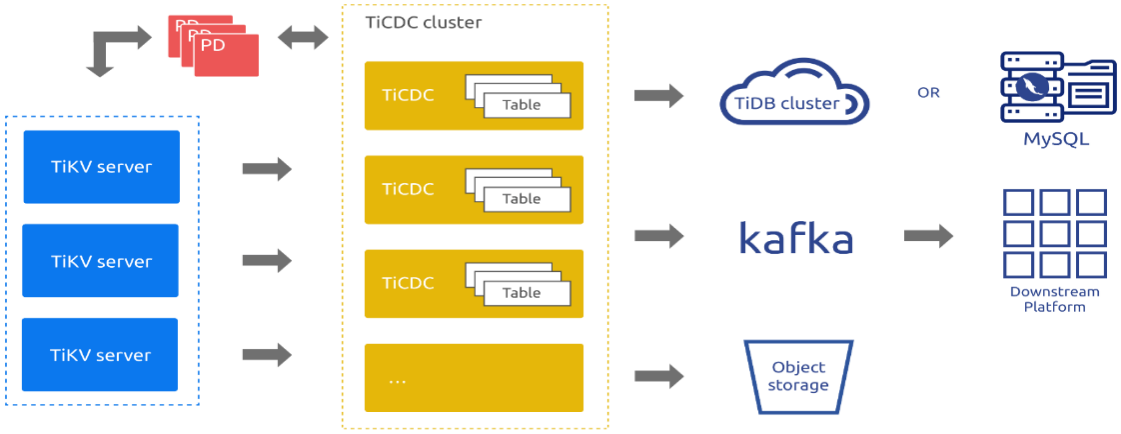
koje pruža TiDB. Izveštaji se mogu direktno generisati korišćenjem SQL naredbi.

Kompatibilnost sa MySQL-om

TiDB je visoko kompatibilan sa MySQL protokolom i zajedničkim funkcijama i sintaksom MySQL verzija 5.7 i 8.0. Sistem alata za MySQL (PHPMyAdmin, Navicat, MySQL Workbench, DBeaver i druge) i MySQL klijent mogu se koristiti i za TiDB. Neke od MySQL funkcionalnosti nisu podržane u TiDB-u. To može biti zato što sada postoji bolji način za rešavanje problema (na primer, korišćenje JSON-a umesto XML) ili zbog nedostatka trenutnog zahteva u odnosu na potrebni napor. Takođe, neke funkcionalnosti mogu biti teške za implementaciju u distribuiranom sistemu.

Važno je napomenuti da TiDB ne podržava MySQL replikacioni protokol. Umesto toga, pružaju se posebni alati za replikaciju podataka sa MySQL-om:

1. Replikacija podataka iz MySQL-a: TiDB Data Migration (DM) je alat koji podržava potpunu migraciju podataka i inkrementalnu replikaciju podataka iz MySQL-a ili MariaDB u TiDB.
2. Replikacija podataka ka MySQL-u: TiCDC je alat za repliciranje inkrementalnih podataka TiDB-a pomoću TiKV promena logova. TiCDC koristi MySQL sink za replikaciju inkrementalnih podataka TiDB-a u MySQL.



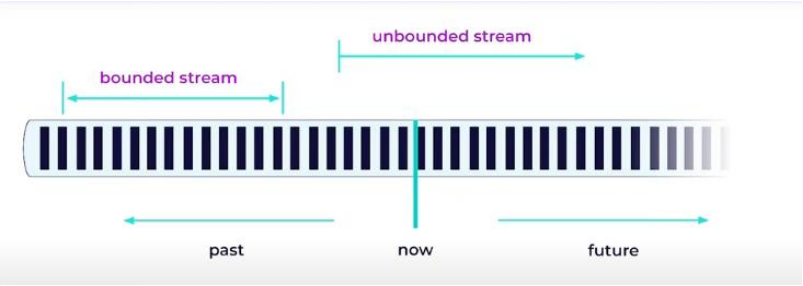
Slika 22. Prikaz postojanja TiCDC-a u sistemu

Više TiCDC procesa povlači promene podataka sa TiKV čvorova. TiCDC sortira i spaja promene podataka. TiCDC replikuje promene podataka na više sistemskih destinacija putem više zadataka za replikaciju (changefeeds). Kada se dogode promene podataka, TiKV čvorovi šalju te promene, kao promene logova, TiCDC čvorovima. PD modul je odgovoran za raspoređivanje podataka u klasteru i obično se sastoji od tri PD čvora. PD pruža visoku dostupnost putem etcd klastera. U etcd klasteru, TiCDC čuva svoje metapodatke, kao što su informacije o statusu čvorova i konfiguracije changefeed-a.

6.3 Flink

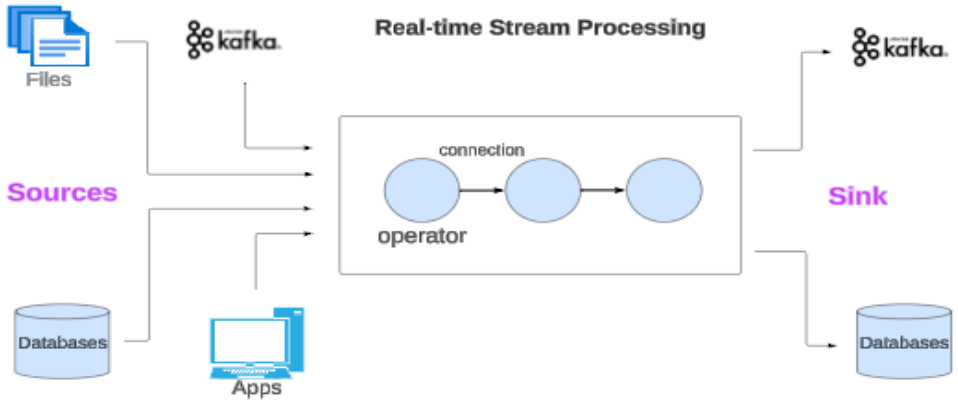
Apache Flink je okruženje i distribuirani klaster za procesiranje neograničenih i ograničenih tokova podataka. Zahvaljujući Flink 1.11 proširenoj podršci za SQL jezik i TiDB-ovoj HTAP karakteristici može se sagraditi RTDW koja je efikasna, laka sa korišćenje, horizontalne skalabilnosti i visoke dostupnosti. Flink je open-source platforma za obradu podataka u realnom vremenu. On omogućava analizu, transformaciju i manipulaciju podataka, obezbeđujući nisku latenciju, visok kapacitet prenosa. Karakterističan je jedinstven pristup obrade podataka u realnom vremenu. Obrađuje podatke kao tokove i serije (stream i batch). Široko rasprostranjeno rešenje kada se radi o zahtevima za trenutan proračun, daje mogućnost za "tačno jednom" semantiku. To znači da će svaka poruka biti obrađena tačno jednom, čak i u slučaju grešaka ili ponovnog pokretanja sistema. Ova funkcionalnost čini Flink pogodnim za aplikacije gde je ključno garantovati tačnost i doslednost podataka.

Apache Flink posatao je sastavni deo mnogih kompanija, kako bi rukovodio njihovim poslovima. Razlog tome je korisnička potreba. Korisnici žele da imaju instant pristup podacima. Ukoliko je neko iskoristio tvoju kreditnu karticu, želite da budete obavešteni o tome u trenutku. Ukoliko ste to bili vi želite da imate uvid u to, odnosno u vašu potrošnju.



Slika 23. Prikaz sekvence događaja, ograničeni i neograničeni stream

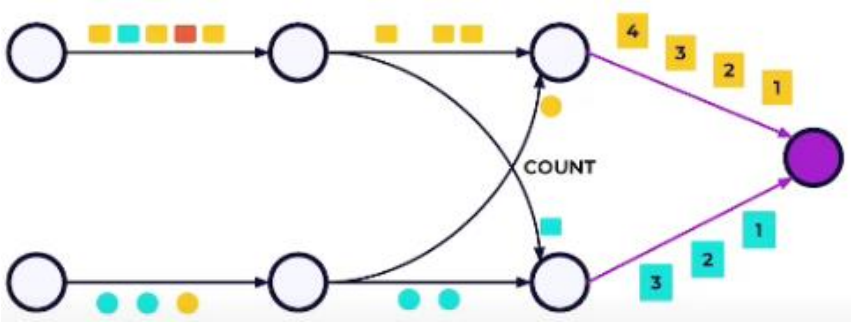
Apache Flink je framework koji povezuje, obogaćuje i procesuira podatke u realnom vremenu. Današnji poslovi, su definisani softverom i procesiranje se uglavnom obavlja automatski. Procesiranje podataka nije bilo izvodljivo dok nisu uvedene strimig platforme, kao što su Kafka i Flink. Što je dalo veliki uticaj na tradicionalnu industriju, kao što su bankarstvo, telekomunikacija i prodaja. Kompanije kao što su Alibaba, Netflix, Uber i Goldman Sachs koriste Flink. Flink nudi API u Java-i, Python-u i SQL-u, na taj način omogućuje korisnicima da budu produktivniji. Omogućuje stream i batch procesiranje, što nudi široki spektar korišćenja. Stream je sekvenca događaja. Stream dakle predstavlja hvatanje događaja u realnom vremenu onako kako se pojavljuju. Sekvence se mogu obrađivati u realnom vremenu, formirajući neograničeni stream. Stream je ograničen ukoliko postoje početni i krajnji timestamp. Dakle, razlikujemo ograničen i neograničen stream, kao što se vidi na slici 23.



Slika 24. Prikaz izvora i sink-a, job operatora i connection-a u okviru Flink-a

Flink preuzima podatke, iz jednog ili više izvora (fajlovi, baze podataka, aplikacije, kafka topic) i šalje podatke jednom ili više sink-ova (baza podataka, kafka topic) kao što je prikazano na slici 24. Koristeći neki od Flink-ovih API-ja, neophodno je definisati biznis logiku i Flink izvršava kod u Flink klasteru. U okviru Flink klastera postoje job-ovi. Kreira se job graf od job-ova odnosno job predstavlja čvor grafa - operator. U grafu razlikujemo operator i konekciju. Operatori u grafu povezani su strelicama.

Stream procesiranje se odvija paralelno. Razdvajanjem stream-ova u paralalne substream-ove. Svaki od njih može se izvršavati nezavisno. Ovo je ključno, pogotovo za skalabilnost. Zato što paralelni operatori ne dele ništa i mogu se pokretati pod punom brzinom, prikaz na slici 25.



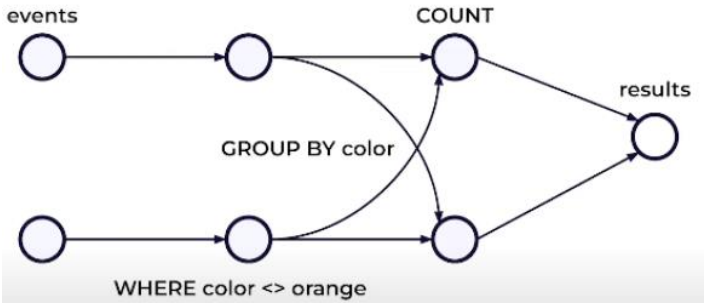
Slika 25. Prikazi job-ova koji se izvršavaju paralelno

Postoje termini kao što su forwarding, repartitioning i rebalancing, koji se vezuju za Flink i bitno ih je definisati. Forwarding se odnosi na proces preusmeravanja podataka ili poruka s jednog čvora na drugi u Flink-ovoj topologiji obrade podataka. Kada podaci stižu na jednom čvoru, oni se mogu proslediti drugim čvorovima u skladu sa specifičnim algoritmima raspodele ili procesiranja. Repartitioning se koristi kada se podaci emituju i raspoređuju na različite čvorove obrade, ponovno razvrstavanje se koristi za promenu raspodele podataka među čvorovima kako bi se postigla ravnomerna raspodela opterećenja ili kako bi se izvršili određeni tipovi operacija poput grupisanja ili spajanja podataka. Rebalancing stream-ova se odnosi na dinamičko preraspoređivanje resursa ili opterećenja između čvorova u Flink-ovom klasteru kako bi se postigla ravnoteža između performansi i resursa. Flink nudi veliki broj paterna, uključujući broadcast, za distribuiranje kroz klaster i joining stream za bogaćenje podataka.

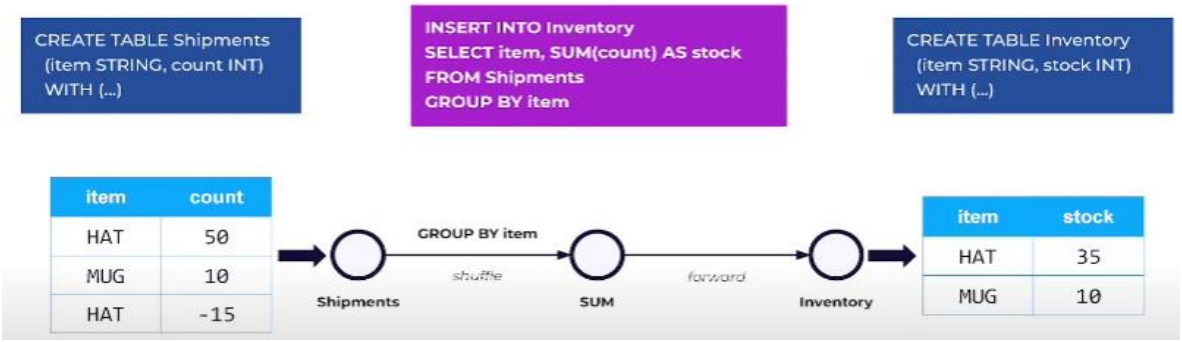
Flink SQL je SQL sistem u skladu sa standardima za obradu kako batch, tako i streaming

podataka sa skalabilnošću, performansama i doslednošću Apache Flink-a. Ovo je veoma izražajan API, baziran na moćnim apstrakcijama. Flink SQL može transformisati SQL upit kao na slici 26.

```
INSERT INTO results
SELECT color, COUNT(*)
FROM events
WHERE color < > orange,
GROUP BY color;
```

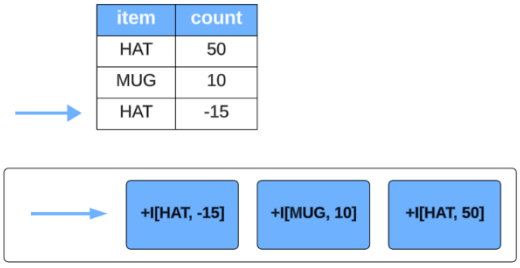


Slika 26. Primer izvršenja sql upita po job-ovima



Slika 27. Prikaz izvršenja sql upita, Shipments tabela kao izvor i Inventory tabela kao sink

Na slici 27. uočavamo da je “Shipments” tabela izvor, a da je “Inventory” tabela sink. Za razliku od baze podataka, Flink ne predstavlja skladište, on koristi standardnu SQL sintaksu da opiše procesiranje koje želimo da Flink odradi. Tabela koju Flink poseduje je metadata koji opisuje šemu i propertije konektora koji su neophodni da se poveže na Kafku, da se isporuče podaci na dobar način do job-ova. Flink SQL poseduje dinamičke tabele koje se menjaju sa vremenom. Svaka tabela je ekvivalentna sa changelog stream-om koji opisuje promene. Dakle tabela shipment, bila bi poslata kao na slici. Kada se desi promena, novi red se doda u tabelu. Ova akcija bi odgovarala insert-u u tabelu. Dešavaju se i druge promene sem inserta , (-U) predstavlja poništavanje prethodnog rezultata. (+U) označava ažuriranje koje se dešava kasnije, ažurira prethodni rezultat. (-D) označava brisanje, briše prethodni rezultat.

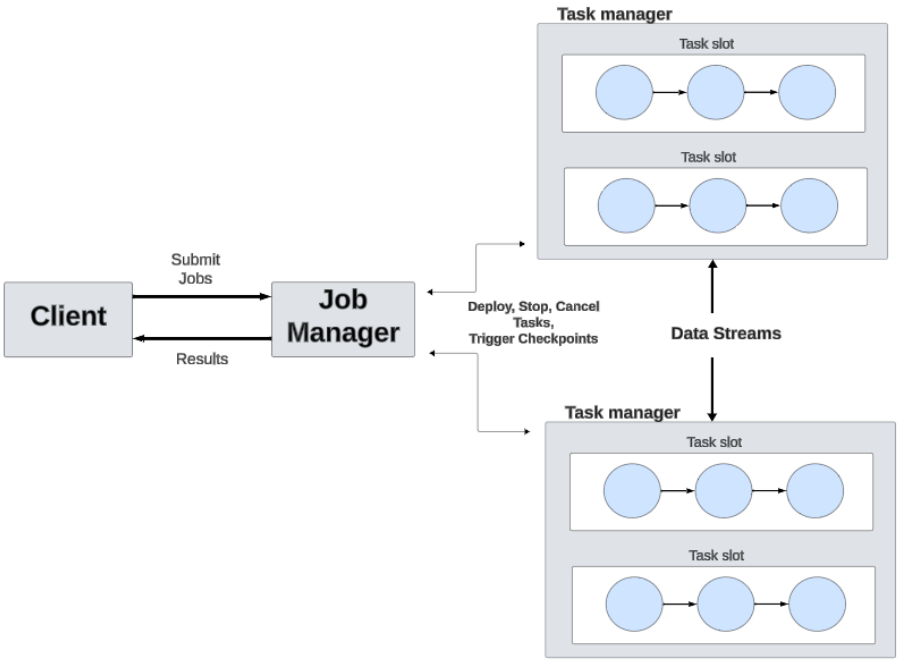


Slika 28. Prikaz tabele u vidu stream-a

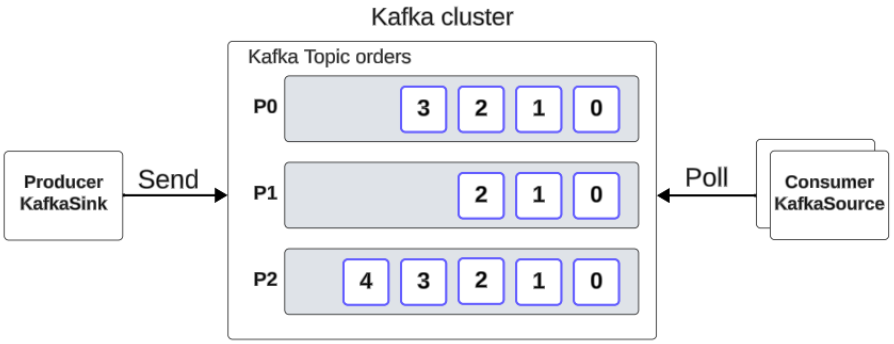
Kako će podaci biti definisani kada se šalju u sink zavisi od toga koji konektor se koristi i koji format. Sink tabela može biti u JDBC bazi ili Kafka topic gde se stream update-a upisuje u changelog stream formatu kao što je Debezium. U Flink SQL postoje karakteristike koje se vežu za Streaming i Batch, postoje i one koje se vežu samo streaming procesiranje, kao na primer order može da se vrši samo po vremenu, inner join može se vršiti sa temporal tabelom i external lookup tabelom. Kod batch procesiranja order se može izvršiti po bilo čemu. Flink SQL po default-u radi u streaming modu.

Kada klijent napiše Flink aplikaciju, on postaje Flink klijent. Job se šalje JobManager-u. Kada JobManager dobije job, on će pronaći ili kreirati resurse koji su neophodni da pokrenu job. JobManager kreira TaskManager-e koliko su neophodni da postigne neophodni paralelizam. Svaki TakManager obezbeđuje određeni task slots. JobManager postaje odgovoran da koordinira aktivnosti Flink klastera. Na primer koordinira checkpointing i restartuje TaskManager ukoliko je pao. TaskManager povlači podatke iz različitih izvora i transformiše ih i šalje podatke drugom TakManager-u ukoliko je to neophodno zbog rebalansiranja i particionisanja i TaskManager izbacuje rezultate u sink.

Razlika između stream i batch procesiranja. Flink garantuje tačno jednom semantiku uprkos izvršenju van redosleda i restartu zbog grešaka. Streaming procesiranje podržava ograničene i neograničene stream-ove, čitav pipeline mora biti pokrenut uvek i input mora da se procesira onako kako pristiže, rezultati mogu biti prikazivani onako kako oni postaju spremni, oporavak od greške otpočinje se od poslednjeg snapshot-a. Batch procesiranje podržava samo ograničeno procesiranje, input mora da se ponovo sortirano po vremenu i ključu, rezultati se prikazuju na kraju job-a, oporavak od greške podrazumeva potpuni restart. Kada je neophodan odgovor u trenutku. Kašnjenje kod batch procesiranja isključuje ga kao opciju. Stream procesiranje je moćnije od batch procesiranja



Slika 29. Prikaz JobManager-a i TaskManager-a



Slika 30. Kafka cluster

Kafka može biti i source i sink za Flink. Ukoliko se u Kafka-u upisuju podaci, onda se koristi kao sink, ukoliko se iz Kafka-e izvlače podaci onda se koristi kao source. Flink je kompatibilan sa Kafka Connect, ksqlDB, Kafka Stream i Schema Registry. Kafka topic je izdeljen po particijama. Kao što se vidi na slici 30. Kafka događaji imaju strukturu. Primarni payload je key-value pair. Kada koristimo Kafku sa Flink SQL, moramo da mapiramo tabelu. Flink ne skaldišti tabele na svojoj strani. Sledeći primer pokazuje kako je Orders tabela mapirana na orders topic u Kafka okruženju.

```
CREATE TABLE Orders (
  `order_id` BIGINT,
  `order_time` TIMESTAMP_LTZ(3) METADATA FROM 'timestamp',
  `price` DECIMAL(32,2),
  `quantity` INT
) WITH (
  'connector' = 'kafka',
  'topic' = 'orders',
  'key.format' = 'json',
```

```
  'value.format' = 'json',
  ...
);
```

Flink mora da zna i format sa kojim se radi. Zasebno se naglašava format za key i za value. Ponekad podaci koji nam trebaju, ne nalaze se u Kafka podacima, već su deo METADATA. Što je u redu, s obzirom da Flink SQL pruža mogućnost pristupa svakom delu Kafka-e. Flink se ponaša kao nivo proračuna Kafke, omogućujući real-time aplikacije i pipeline. Flink periodično beleži stanje kopirajući u remote trajno skladište kao što je S3. Filtriranje je stateless u Flink-u. Nema oko čega brinuti. Dok je agregacija i spajanje potencijalno opasno, ove operacije nazivamo materializing operacije, zato što moraju da se prate sve vreme.

6.4 Kombinovanje tehnologija za implementaciju RTDW-a

Prednosti kombinovanja TiDB-a i Flink-a u RTDW-u jesu sledeće:

1. Brzina. Pružaju mogućnost procesiranja tokova podataka u sekundama i real-time analizu podataka.
2. Horizontalna skalabilnost. Može se poboljšati moć računanja dodavanjem novih čvorova i u Flink i u TiDB.
3. Visoka dostupnost. Ukoliko jedna instanca ne uspe, klaster je i dalje nepromenjen i podaci ostaju kompletni i dostupni. Podržava stvaranje rezervnih kopija i rad sa njima. Tako da omogućava vraćanje prethodne verzije sistema.

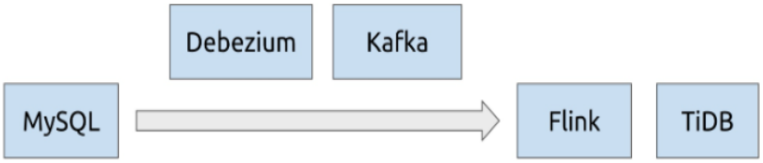
MySQL kao izvor podataka

TiDB je kompatibilan sa MySQL 5.7 protokolom. U Flink 1.11 može se koristiti Flink SQL sintaksa i moćni konektori za pisanje i potvrdu zadataka. Ako koristimo CDC koji je Ververica kreirao, dobijamo mogućnost da Flink ne bude samo nivo koji će prikupljati podatke, već i obrađuje ih. Omogućava spajanje i preagregaciju. Ova arhitektura je jednostavna i pogodna. ODI – Oracle Data Integrator stvorio je okvir za praćenje promenjenih podataka – Change Data Capture. CDC prenosi samo promenjene podatke ka ciljnim sistemima i može se integrisati sa Oracle Golden Gate-om, čime obezbeđuje integraciju u realnom vremenu. CDC prepoznaje i beleži podatke dok se insertuju, ažuriraju ili brišu iz skladišta podataka i čini promenjene podatke dostupnim za integracione procese.

Slika 31. Arhitektura u kojoj je MySQL izvor podataka

Povezivanje Kafka-e i Flink-a

Flink može primiti podatke kroz Flink Kafka konektor. Dakle kada želimo da se podaci iz MySQL skladište u nekim skladištima kao što je Kafka, preporučuje se da se koristi Canal ili Debezium za skladištenje podataka. Flink može da obavlja parsovanje.



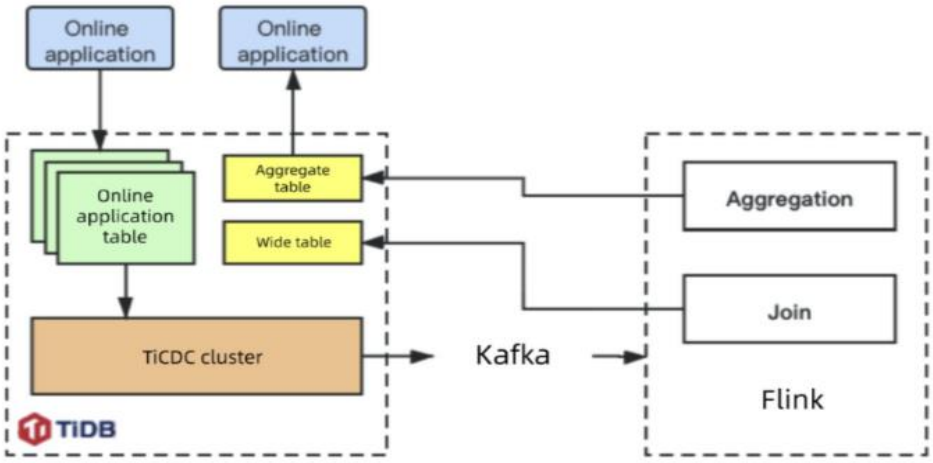
Slika 32. Prikaz da i Kafka može biti iskorišćen

TiDB kao izvor podataka

TiDB poseduje i TiCDC. To je open-source rešenje koje prati inkrementalne promene TiDB-a. Kroz mehanizam Change Data Capture, TiCDC beleži promene u podacima, kao što su ažuriranja, brisanja i umetanja, omogućavajući korisnicima praćenje i reagovanje na te promene. TiCDC se koristiti da se izlazne promene u TiDB-u prenesu u red poruka, a zatim ih Flink može izvući i nastaviti dalju obradu.

Slika 33. Prikaz korišćenja TiCDC-a

U TiDB 4.0.8 može se konektovati TiDB sa Flinkom kroz TiCDC Open Protocol. U kasnijim verzijama, TiCDC podržava canal-json format izlaza.



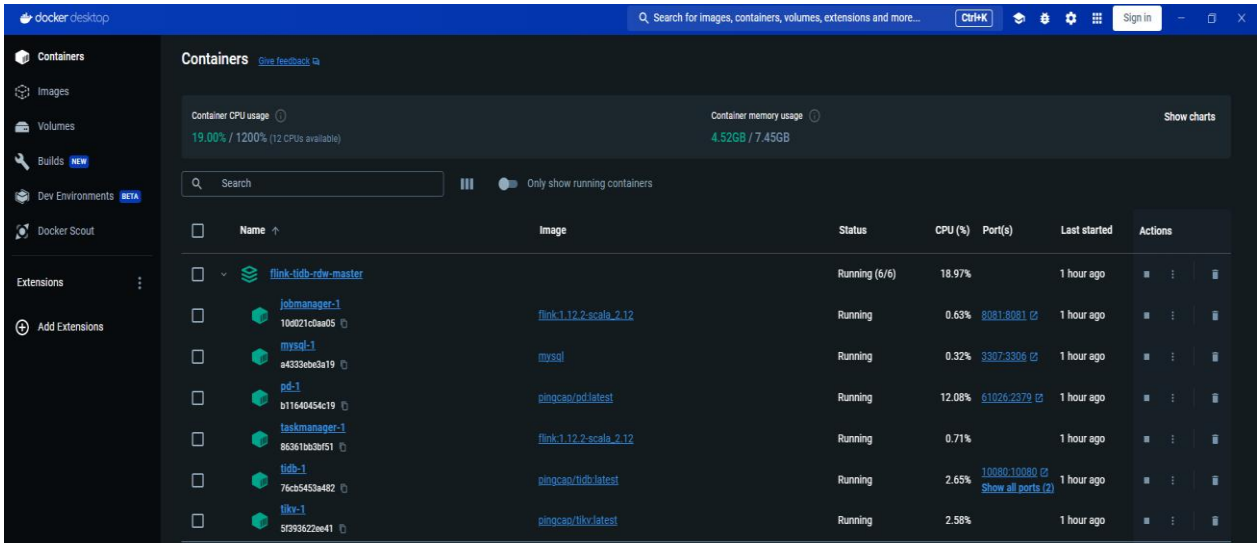
Slika 34. Primer arhitekture u kojoj figurišu TiDB, Kafka i Flink

Online aplikacije koje su aktivne izvršavaju OLTP zadatke. Podaci se šalju u tabele. TiCDC klasteri registruju promene i šalju promene u Kafku. Flink čita promene iz Kafke i izvodi proračune, kao što je spajanje i agregacija tabela. Flink upisuje rezultate u TiDB.

7. Implementacija RTDW-a

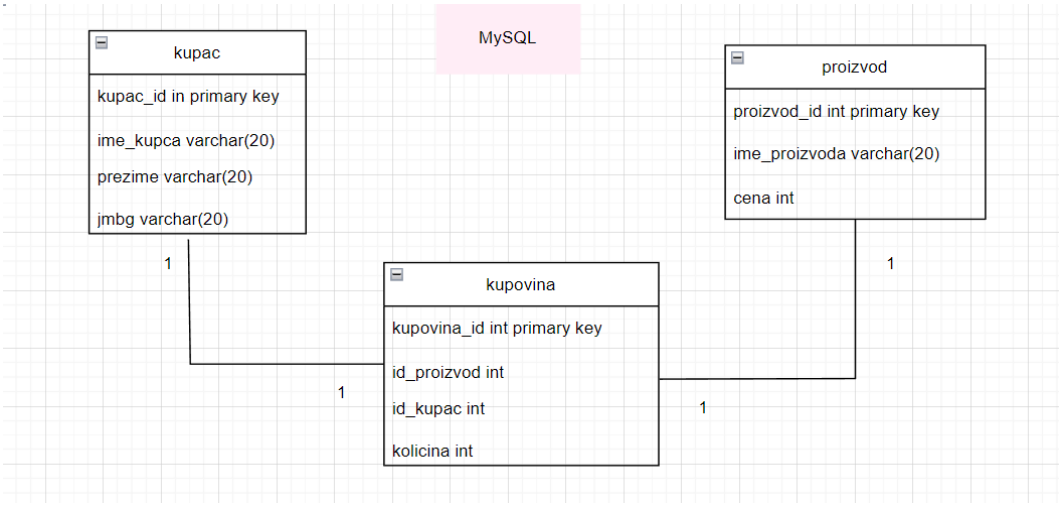
Slika 35. Prikaz arhitekture projekta

MySQL baza u mom slučaju je izvor podataka, a Flink je zadužen za integraciju podataka koji pristižu iz MySQL-a. U TiDB pristižu podaci iz Flink-a. Podaci se prosleđuju između MySQL-a i Flink-a, kao i između Flink-a i TiDB-a zahvaljujući konektorima koji su napisani u javi, koje sam uključila u projekat. To su mysql-cdc i jdbc konektori. Kada se integrisani podaci upišu u TiDB, oni se prikazuju u web aplikaciji. Slikovit prikaz arhitekture projekta može se videti na slici 35.



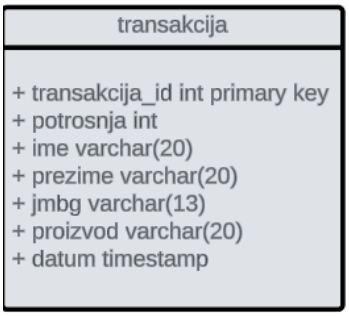
Slika 36. Prikaz pokrenutih aplikacija iz docker-a

Kao što je prikazano na slici 36, u okviru docker-compose-a pokrenute su aplikacije. Komandom docker-compose podižem kontejner, na osnovu pokrenutih slika. U okviru MySQL baze kreirala sam tabele koje se mogu videti na slici 37.



Slika 37. Dijagram klasa, klase u okviru mysql-a

U okviru TiDB baze kreirala sam tabelu transakcija koja se može videti na slici 38.



Slika 38. Klasa u TiDB-u

U okviru Flink-a kreiram tabele ekvivalentne tabelama u MySQL-u i TiDB-u. One u delu connector imaju referencu na my-cdc ili jdbc konektor. Primer kreiranja tabele u Flinku koja ima za konektor mysql-cdc može se videti na slici 39, dok primer tabele koja za konektor ima jdbc, na slici 40.

```
Flink SQL> create table kupac (
>   kupac_id int primary key,
>   ime_kupca varchar(20),
>   prezime varchar(20),
>   jmbg varchar(13)
> ) WITH (
>     'connector' = 'mysql-cdc',
>     'hostname' = 'mysql',
>     'port' = '3306',
>     'username' = 'root',
>     'password' = '',
>     'database-name' = 'vezba',
>     'table-name' = 'kupac'
> );
[INFO] Table has been created.

Flink SQL>
```

Slika 39. Primer kreiranja tabele u Flink-u

```
Flink SQL> create table transakcija(
>     transakcija_id int primary key,
>     potrosnja int,
>     ime varchar(20),
>     prezime varchar(20),
>     jmbg varchar(13),
>     proizvod varchar(20),
>     datum TIMESTAMP
> ) WITH (
>     'connector'  = 'jdbc',
>     'driver'     = 'com.mysql.cj.jdbc.Driver',
>     'url'        = 'jdbc:mysql://tidb:4000/vezba?rewriteBatchedStatements=true',
>     'table-name' = 'transakcija',
>     'username'   = 'root',
>     'password'   = ''
> );
[INFO] Table has been created.

Flink SQL>
```

Slika 40. Primer kreiranja tabele u Flinku

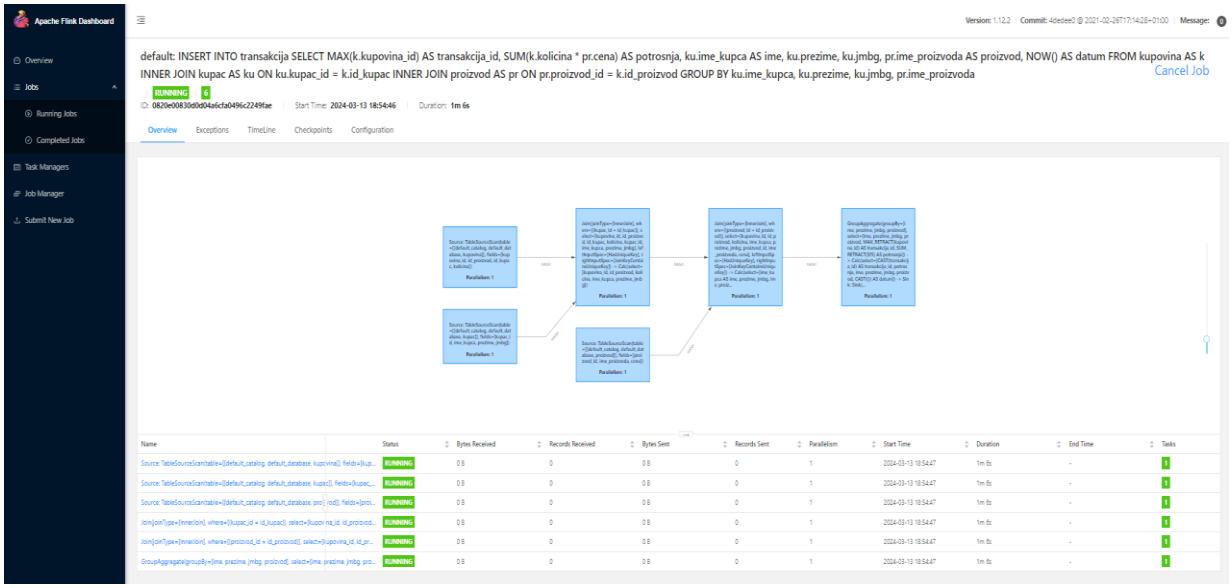
Pokretanjem job-a obezbeđujem da se podaci upisuju u tabelu transakcija u TiDB-u.

```
Flink SQL> INSERT INTO transakcija
> SELECT
>     MAX(k.kupovina_id) AS transakcija_id,
>     SUM(k.kolicina * pr.cena) AS potrosnja,
>     ku.ime_kupca AS ime,
>     ku.prezime,
>     ku.jmbg,
>     pr.ime_proizvoda AS proizvod,
>     NOW() AS datum
> FROM
>     kupovina AS k
> INNER JOIN kupac AS ku ON ku.kupac_id = k.id_kupac
> INNER JOIN proizvod AS pr ON pr.proizvod_id = k.id_proizvod
> GROUP BY ku.ime_kupca, ku.prezime, ku.jmbg, pr.ime_proizvoda;
[INFO] Submitting SQL update statement to the cluster...
[INFO] Table update statement has been successfully submitted to the cluster:
Job ID: 7a8f0391d6aa0c71d1c12eb9b75430e4


Flink SQL>
```

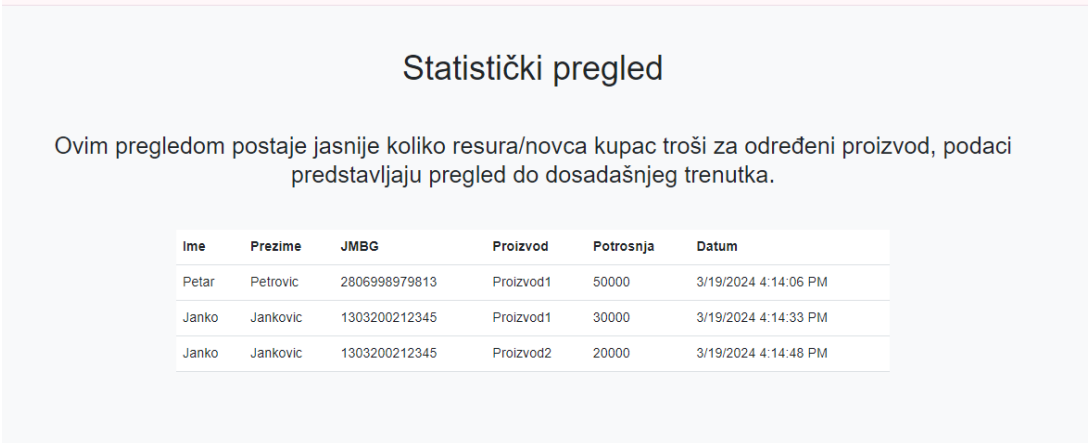
Slika 41. Primer kreiranja job-a u Flink-u

Kada se job kreira postaje vidljiv u jobamanager-u na portu 8081, kao sto je vidljivo na slici 42.



Slika 42. Prikaz Pokrenutog job-a

Web aplikacija prikazana na slici 43, povezana je sa TiDB bazom i prikazuje na osnovu datuma podatke iz tabele transakcija, koji su grupisani na osnovu proizvoda i kupca.



Slika 43. Prikaz web aplikacije

8. Prednosti primene RTDW-a

Od brzog i efikasnog donošenja odluka do poboljšanog pristupa podacima, RTDW zaista poboljšava način na koji organizacije pristupaju analizi i upravljanju podacima. Poboljšanja, odnosno prednosti uvođenja real-time data warehouse-a jesu:

1. Brže donošenje odluka. Zahvaljujući RTDW-u preduzeća imaju pristup ažuriranim informacijama iz različitih izvora. To im omogućava da donose odluke brže i na fleksibilniji način.
2. Poboljšani pristup podacima. Svi u organizaciji imaju pristup trenutnim i istorijskim podacima, što dovodi do donošenja boljih odluka.
3. Personalizovano iskustvo kupaca. RTDW pruža personalizovano korisničko iskustvo. RTDW nije gotov produkt, prilagođava se potrebama korisnika.
4. RTDW smanjuje opterećenje na izvoru podataka i eliminiše potrebu za identifikovanim vremenskim prozorom za učitavanje podataka, olakšavajući održavanje tačnosti i doslednosti.
5. Brži oporavak. U slučaju problema sa konverzijom ili učitavanjem podataka, RTDW omogućava brži proces oporavka.
6. Konzistentnost podataka. Uklanjajući batch prozor i pokrećući hijerarhiju u bazi, RTDW se rešava da se nekonzistentni podaci reflektuju u upitima i obezbeđuje na taj način da RTDW pruža stabilnije podatke.
7. Veća tačnost podataka. RTDW pomaže organizacijama da dokažu tačnost njihovih podataka. Tako je jer dozvoljava organizacijama da procesiraju i analiziraju podatke čim postanu dostupni, na taj način daju mogućnost da se reše mogućih grešaka. Organizacije na taj način sprečavaju mogućnost grešaka koje ih mogu skupo koštati.
8. Bolja kolaboracija. RTDW omogućavaju bolju kolaboraciju članova. Postiže se bolja kolaboracija jer delovi organizacije dele uvid u odluke koje se postižu u realnom vremenu, dakle ne postoji čekanje, odluke se donose trenutno. Odluke u smislu proračuna koji se mogu trenutno koristiti, odnosno postaju dostupni delovima organizacije koji ih tumače za sopstvene potrebe.
9. Skalabilnost. Kapacitet skladišta podataka se može proširiti, odnosno može se postignuti fleksibilnost i skalabilnost koja će doprineti rastu i razvoju poslovne logike, koja podstiče rast i razvoj kompanije.

9. Najbolja praksa, saveti, prilikom kreiranja RTDW-a

Korišćenje RTDW predstavlja u osnovi dobar koncept. Koristiti RTDW na pravi način predstavlja jako bitnu stvar. Kao i svaka druga tehnologija, njen puni potencijal postiže se onda kada je na pravi način implementiramo. Zato je i bitno istaknuti pojedinosti koje treba ispratiti i koji predstavljaju dobru praksu prilikom kreiranja RTDW-a. S obzirom da RTDW nije gotov produkt nego nešto što je individualno rešenje za svaku kompaniju, najbolja praksa je sledeća:

1. Testiraj pre nego investiraš. Praksa je pokazala da je jako bitno dati vreme za testiranje. Pokretati simulacije i testove kako bi osigurali da je algoritam koji je kreiran čvrst. Jedna od najvećih katastrofa koja se desila je slučaj vezan za Knight Capital, koji je doživeo veliki gubitak za kratak vremenski period.
2. Budi kreativan prilikom analize. Kreativnošću se postiže pun kapacitet RTDW, jer same prednosti ne dolaze samo od postojanja, već i od što bolje primene.
3. Paralelno procesiranje. Baš zbog obrade velike količine podataka, postojanje

paralelnih hardvera za procesiranje podataka, je neizbežna opcija.

4. Imati planove za vanredne situacije. Moraju postojati alternative za slučajeve. Neke situacije se uvek mogu predvideti. Odnosno, treba ostaviti prostora za drugačija rešenja i obezbediti način njihovog rešavanja.
5. Prelazak na cloud. Treba imati na umu prednosti koje pruža cloud i iskoristiti te prednosti imajući na umu pre svega cenu koja može značajno uticati na sistem koji kreira RTDW. Dakle RTDW koji koristi sopstvene resurse za skladištenje i RTDW koji koristi klaud se u mnogome mogu razlikovati. Dobra praksa je o tome razmišljati u startu prilikom projektovanja RTDW-a.
6. Čuvati na sigurnom. O tome da je čuvanje podataka na sigurnom mestu jako važna stavka, takođe treba voditi računa na početku. Podaci koji se skladište trebaju biti zaštićeni na najbolji način, obezbeđujući kriptovanje, kontrolu pristupa, autorizaciju, autentifikaciju, fairwall itd.
7. Imati plan za oporavak. Greške nisu nešto što se može u potpunosti izbeći bilo koji sistem da je u pitanju. Zbog toga bitno je imati strategiju za oporavak od grešaka koje mogu nastupiti. Oporavak od greške mora podrazumevati postojanje bekapovanja podataka. Takođe, bitno je da je samo bekapovanje efikasno. Uz minimalne troškove u vidu novca i ostalih resursa.

10. Neki najčešći primeri korišćenja RTDW-a

Postoje različite primene RTDW-a, neke od njih jesu:

1. Finansije: RTDW primenjuje se u finansijskim sistemima radi detekcije prevara i prekidanja transakcija u takvim situacijama.
2. Elektronska kupovina i prodaja: primenjuje se u sistemima za prodaju na mestu prodaje radi dodatnog prodavanja proizvoda i usluga. Takođe olakšava predviđanje ponašanja kupaca i usmerava marketinške napore.
3. Dostava i logistika: primenjuje se u sektoru dostave i logistike radi praćenja statusa isporuka, optimizacije rute, i poboljšanja efikasnosti i preciznosti celokupnog lanca snabdevanja.
4. Personalizovano korisničko iskustvo: primenjuje se da obezbedi personalizovano korisničko iskustvo u elektronskoj kupovini.
5. Bezbednosna analiza: primenjuje se kako bi se detektovalo i odgovorilo na potencijalne pretnje. Dakle na osnovu trenutnih poteza i na osnovu istorije treba zaključiti da li je potez potencijalna pretnja.
6. Optimizacija lanca prodaje: primenjuje se u upravljanju lancem snabdevanja radi povećanja operativne efikasnosti.
7. Isporuka hrane: primenjuje se kako bi korisnicima pružila uvid u trenutno stanje njihove narudžbine.
8. Menadžment za odnose sa mušterijama: predstavlja podršku u radu sa mušterijama, obezbeđuje uvide u ponašanje i poteze mušterije. Ovo obezbeđuje brze odgovore potrebama mušterija, poboljšavajući zadovoljstvo mušterije i na taj način stvara verne potrošače.

11. Zaključak

RTDW jeste inovativna tehnologija koja obezbeđuje organizacijama brzo i efikasno procesiranje ogromnih količina podataka u skoro pa realnom vremenu. Rast RTDW-a predstavlja posledicu važnosti podataka u današnjim okruženjima. S obzirom da organizacije teže da iskoriste podatke kako bi postigle poslovni uspeh, RTDW postaje ključan alat za omogućavanje brzog i preciznog pristupa velikim količinama podataka.

Kroz pravilno implementiran RTDW, organizacije mogu efikasno pristupiti, analizirati i interpretirati velike količine podataka, što im omogućava donošenje bržih i preciznijih odluka. Kombinujući RTDW sa najboljim praksama poslovnog upravljanja i korišćenjem odgovarajućih tehnologija, organizacije mogu maksimizirati prednosti skladištenja podataka u realnom vremenu. Tehnologije poput MySQL-a, Flink-a i TiDB-a koje sam odabrala pružaju snažne alate za implementaciju RTDW-a. Kroz njihovu integraciju i upotrebu odgovarajućih konektora, omogućava se glatko kretanje podataka od izvora do skladišta, što je ključno za uspeh projekta.

12. Literatura

[1] Building the Data Warehouse Third Edition - W. H. Inmon - 19.03.2024
[2] What is a data warehouse and why are they important 19.03.2024
[3] What is a real time data warehouse? Benefits and Best Practises. 19.03.2024
[4] Real-Time Data Warehouse Examples (Real World Applicatons) 19.03.2024
[5] Real Time Data Warehousing 19.03.2024
[6] Demo Realtime Data Warehousing 19.03.2024
[7] How To Achieve High Performance Data Ingestion to tidb in Apache Flink 19.03.2024
[8] How to achieve high performance data ingestion to tidb in apache flink 19.03.2024
[9] Flink on TIDB 19.03.2024
[10] TiDB 19.03.2024
[11] Tidb internal data storage 19.03.2024
[12] ETL vs ELT 19.03.2024
[13] Data warehouse testing 19.03.2024
[14] Apache flink web ui 19.03.2024
[15] Flink overview 19.03.2024
[16] Flink Dynamic tables 19.03.2024
[17] Flink SQL 19.03.2024
[18] Conlfluent Cloud Apache Flink 101 19.03.2024
[19] Flink First steps 19.03.2024
[20] Build an apache flink application from scratch in 5 minutes 19.03.2024
[21] Apache Flink stream processing exercise 19.03.2024
[22] How to write simple and efficient Flink SQL 20.03.2024
[23] Building a .Net core web API and connect to TiDB MySQL Server 20.03.2024
[24] Flink SQL 20.03.2024