Sustavi za skladištenje podataka

Čupić, Luka Ivan

SVEUČILIŠTE U ZAGREBU
FILOZOFSKI FAKULTET
ODSJEK ZA INFORMACIJSKE I KOMUNIKACIJSKE ZNANOSTI
Ak. God 2019/20.

Luka Ivan Čupić

SUSTAVI ZA SKLADIŠTENJE PODATAKA

Diplomski rad

Mentor: dr. sc. Kristina Kocijan, izv.prof.

Zagreb, 2020.

Izjava o akademskoj čestitosti

Izjavljujem i svojim potpisom potvrđujem da je ovaj rad rezultat mog vlastitog rada koji se temelji na istraživanjima te objavljenoj i citiranoj literaturi. Izjavljujem da nijedan dio rada nije napisan na nedozvoljen način, odnosno da je prepisan iz necitiranog rada, te da nijedan dio rada ne krši bilo čija autorska prava. Također izjavljujem da nijedan dio rada nije korišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

_____

(potpis)

Sadržaj

1. Uvod

S obzirom da je informacija kamen temeljac informacijskih i komunikacijskih znanosti, važan naglasak se ujedno stavlja i na samu pohranu tih istih informacija u današnje vrijeme, te neke od načina njihova korištenja. Iz tog razloga sam odabrao sustave za skladištenje podataka; slojeve ispod samih baza podataka gdje se određeni podatci zadržavaju desecima godina čekajući dan kada će biti po prvi puta ili ponovno korišteni. S obzirom na veličinu i širinu polja, rad će pretežito biti upotrebljiv početnicima u području analize podataka, poslodavcima koji žele dobiti sliku o implementaciji i odabiru određenih sustava za skladištenje podataka u samo poslovanje, no nisu pretežito upoznati s njima, ili ljudima koji jednostavno žele dobiti sliku o samome polju istraživanja. Ujedno naglasak stavljam na „skladištenje“ s obzirom da se „pohrana“ dosta često povezuje s bazama podatak koje su važno svojstvo svakog sustava za skladištenje podataka, no nisu tema ovoga rada, već su kao i u skladištima podataka, samo jedna od komponenti.

U samim početcima poslovanja, količina dobivenih informacija i podataka nije bila toliko relevantna za poslovanje određene kompanije, odnosno nije se do te mjere cijenio svaki dobiveni detalj vezan uz poslovanje ili kupce. Danas je situacija nešto drugačija. Uzevši u obzir da živimo u sve ubrzanijem i prilagođenijem svijetu gdje potrebe tržišta i korisnika moraju biti što prije i što kvalitetnije ispunjene, neupitno je da svaka informacija mora biti iskorištena ali problematika nastaje u trenutku kada treba preuzeti, filtrirati i pohraniti relevantne informacije. U tom trenu sustavi za skladištenje podataka postaju bitna komponenta. Razumijevanjem arhitekture samog skladišta, svojstvima i procesima koji se nalaze u njemu, te načinu korištenja u svakodnevici naglasiti ću važnost korištenja sustava radi lakšeg i učinkovitijeg poslovanja poduzeća o čijoj brzini na kraju dana i sami ovisimo.

U prvom poglavlju obraditi ću kratku povijest sustava za skladištenje podataka, razlike između baza podataka i sustava za korištenje podataka i osnovne koncepte vezane uz sustave za skladištenje podataka. Nakon toga ću prikazati i objasniti različite strukture

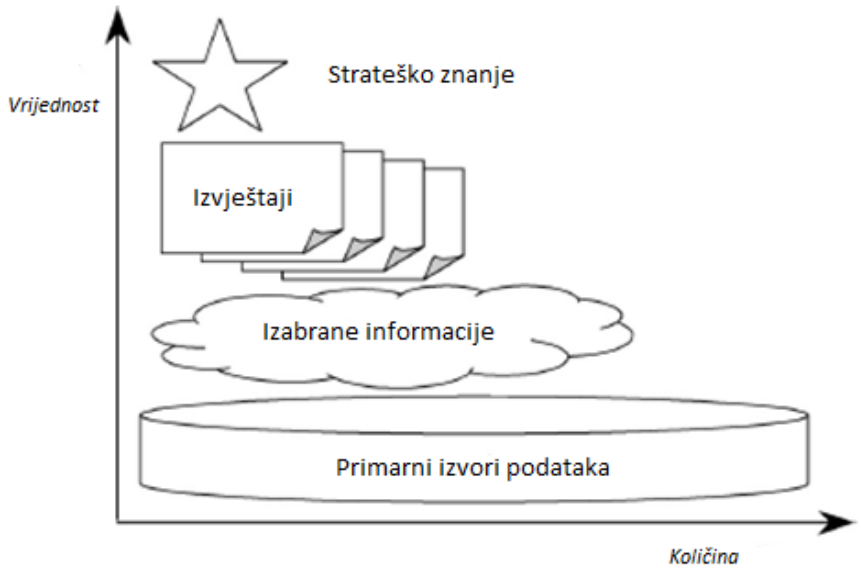
navedenih sustava radi jasnijeg razumijevanja logike koja se koristi prilikom pohrane i obrade podataka, te prikazati što jasnije jedan od najvažnijih koncepata vezanih uz sustave za skladištenje podataka – višedimenzionalnost. Nadolazeće poglavlje fokusirat će se na procese ekstrakcije, transformacije i učitavanja te način na koji se ti podatci mogu iskoristiti u poslovanju. Dobivši svime navedenim jasniju sliku o glavnim funkcionalnostima i logici sustava, izdvojiti ćemo kriterije vezane uz odabir najboljeg sustava s obzirom na mogućnosti i potrebe poslovanja. Nadovezujući se na prethodno poglavlje uspoređivati ćemo sustave za skladištenje podataka u oblaku i tradicionalne sustave za skladištenje podataka navodeći pritom kriterije koje s obzirom na potrebe vremena ispunjavaju ili ne ispunjavaju, nakon čega će se rad pretežito fokusirati na sustave za skladištenje podataka u oblaku.

Poglavlje vezano uz samo istraživanje će biti ključan dio ovoga rada. U njemu ću opisati tri sustava za skladištenje podataka; prvi je *Snowflake* koji se smatra jednim od novijih skladišta podataka, nakon toga slijedi *Amazon Redshift*, stvoren od strane globalno priznate korporacije i na kraju jednako poznat *Google BigQuery*. Odabrao sam tri navedena alata iz razloga što se već niz godina za redom nalaze među najkorištenijim skladištima podataka u svijetu, što se prilagođavaju potrebama poslovanja kontinuirano i što su uz svoju kompleksnost među jednostavnijim skladištima za korištenje. O svakom skladištu ću napraviti kratki uvod, prezentirati osnovnu proceduru za unos i obradu podataka, naglasiti ključne elemente, te prezentirati primjere upotrebe. Poglavlje o usporedbi svih triju sustava će naglasiti njihove razlike i specifične vrline radi objektivnog i lakšeg zaključivanja o tome koji sustav najviše odgovara kojem korisniku.

Rad ću završiti kratkim i sveobuhvatnim opisom prije obrađenih poglavlja te ga finalizirati svojim kratkim mišljenjem i zaključkom vezanim uz cjelokupnu temu.

2. Evolucija sustava za podršku odlučivanju

Kako bi lakše shvatili ulogu sustava za skladištenje podataka, trebamo prvo shvatiti odakle su i zbog kakvih su potreba nastali. Do sredine 1980-ih, baze podataka poduzeća pohranjivale su samo operativne podatke – podatke stvorene kroz poslovne operacije uključene u svakodnevne procese upravljanja, kao što su upravljanje kupnjom, upravljanje prodajom i izdavanje računa. Međutim, svako poduzeće mora imati brz i sveobuhvatan pristup informacijama koje su potrebne prilikom donošenja poslovnih odluka. Navedene potrebne i strateške informacije se dobivaju uglavnom iz velike količine operativnih podataka pohranjenih u bazama podataka poduzeća kroz progresivne postupke selekcije i agregacije, kao što je prikazano na slici 1.



Slika 1: Količina informacija naspram količini podataka[1]

Činjenica da su računala, za menadžere, postala najefikasniji i najproduktivniji alat za donošenje odluka, otvorila je vrata za uvođenje sustava za podršku odlučivanju ili skraćeno DSS (engl. *desicion support systems*). DSS je set proširivih, interaktivnih tehnika

[1] Prilagođeno prema izvoru (Golfarelli & Rizzi, 2009:2).

informacijskih tehnologija (IT) i alata dizajniranih za obradu i analizu podataka kako bi pružali podršku menadžerima prilikom donošenja odluka. Kako bi to postigao, sustavi uparuju individualne resurse menadžera s računalnim resursima radi pospješivanja kvalitete donesene odluke. U praksi, DSS je IT sustav koji pomaže menadžeru oko donošenja odluke ili da izabere između više alternativa. Sam sustav procjenjuje vrijednost i korist svake alternative omogućujući menadžeru da kritički preispita rezultat. Tablica 1 prikazuje moguću klasifikaciju DSS-a s obzirom na njihove funkcionalnosti (Golfarelli & Rizzi, 2009).

Sustavi	Opis
Pasivni DSS	Podržava procese donošenja odluka, ali ne nudi prijedloge po pitanju odluka ili rješenja.
Aktivni DSS	Nudi prijedloge i rješenja.
Kolaborativni DSS	Operira interaktivno i dopušta modifikaciju, integraciju ili rafiniranje prijedloga dobivenih od sustava. Prijedlozi se šalju natrag sustavu radi validacije.
DSS temeljen na modelu	Pospješuju rukovanje statističkim, financijskim, simulacijskim modelima i modelima za optimizaciju.
DSS temeljen na komunikaciji	Daje podršku grupi ljudi koja obavlja određeni zadatak.
DSS temeljen na podatcima	Pospješuje pristup i rukovanje vremenskim serijama[2] korporacijskih ili vanjskih podataka.
DSS temeljen na dokumentima	Rukuje i procesira nestrukturirane podatke u raznim formatima.
DSS temeljen na znanju	Sadrži značajke namijenjene rješavanju problema u obliku činjenica, pravila i procedura.

Tablica 1: Klasifikacija sustava za podršku pri odlučivanju [3]

2.1. Skladišta podataka

Sustavi za skladištenje podataka (engl. *data warehouse systems*) su sustavi na koje je obraćeno najviše pažnje od svih DSS-ova. Golfarelli i Rizzi (2009) opisuju proces

[2] serija indeksiranih podataka (ili grafičkih ili izlistanih) prema vremenskom poretku.
[3] Prilagođeno prema izvoru (Golfarelli & Rizzi, 2009:3).

skladištenja podataka kao skup metoda, tehnika i alata stvorenih kao podrška višim menadžerima, direktorima, menadžerima i analitičarima radi provođenja analize podataka, pružanju pomoći prilikom donošenja odluka i poboljšavanju samih informacijskih resursa. Neke od glavnih karakteristika procesa skladištenja podataka su :

- **pristupačnost** za korisnike koji nisu pretjerano upoznati s IT-em i podatkovnim strukturama,
- **integracija** podataka na bazi standardnih *enterprise* modela,
- **fleksibilnost** upita radi maksimiziranja prednosti dobivenih od postojećih informacija,
- **informacijska konciznost** koja dopušta ciljanu i efektivnu analizu,
- **višedimenzionalan prikaz** koji daje korisnicima intuitivan i lako iskoristiv prikaz informacija,
- **korektnost** i **potpunost** integriranih podataka.

Skladišta podataka su kolekcije podataka koje pomažu pri donošenju odluka. Skladišta podataka su „subjektno orijentirana“ jer ovise o konceptu poduzeća kao što su kupci, proizvodi, prodaja i narudžbe (Golfarelli & Rizzi, 2009). Suprotno tome, operativne baze podataka ovise o različitim poslovnim aplikacijama. Naglasak se ujedno stavlja na integraciju i dosljednost, jer skladišta podataka koriste višestruke izvore podataka, poput podataka iz informacijskih sustava trećih strana ili partnerskih organizacija.

Operativni podatci uglavnom pokrivaju kratak vremenski period, jer većina transakcija uključuje najnovije podatke. Skladište podataka treba omogućavati analize podataka koje pokrivaju zadnjih par godina. Iz tog razloga su skladišta podataka učestalo ažurirana s operativnim podatcima te postepeno rastu. Generalno, podatci nikad nisu izbrisani iz skladišta podataka i ažuriranja se izvršavaju najčešće kad su skladišta podataka van mreže. Skladišta podataka su praktički baze podataka gdje se podatci mogu samo iščitavati (engl. *read-only*), te zbog toga brzo izvršavaju analizu postavljenih upita. Ta vrlina utječe ujedno na tehnologije specificiranih sustava za upravljanje bazama podataka (engl. *Database*

management systems) jer nema potrebe za izvršavanjem naprednih transakcijskih tehnika potrebnih za operativne aplikacije (Golfarelli & Rizzi, 2009).

Druge razlike između operativnih baza podataka i skladišta podataka su povezane s vrstama upita. Operativni upiti izvršavaju transakcije koje obično čitaju / pišu manji broj n-torki od /do tablica povezanih jednostavnim vezama. Primjerice takvi upiti se koriste kada pretražujemo podatke o kupcu kako bi dodali novu narudžbu od kupca. Takav tip upita je tzv. OLTP upit (engl. *On-Line Transaction Processing*), odnosno mrežno transakcijsko procesiranje.

Suprotno tome, vrsta upita koja se koristi u skladištima podataka je OLAP (engl. *On-Line Analytical Processing*) ili mrežno analitičko procesiranje. Navedeni upit sadrži dinamičke, višedimenzionalne analize koje trebaju skenirati ogromnu količinu zapisa za obradu skupova brojčanih podataka koji sažimaju rad poduzeća. Značajke OLAP upita sugeriraju usvajanje višedimenzionalnog prikaza za podatke iz skladišta podataka. Sami podatci se promatraju kao točke u prostoru, čije dimenzije odgovaraju mnogim mogućim dimenzijama same analize. Svaka točka predstavlja događaj koji se odvija ili se odvijao u poduzeću te je opisan nizom mjera relevantnih za proces donošenja odluka. Tablica 2 sažima glavne razlike između operativnih baza podataka i skladišta podataka (Golfarelli & Rizzi, 2009).

Značajke	Operativne baze podataka	Skladišta podataka
Korisnici	Tisuće	Stotine
Tip posla	Trenutačne transakcije	Specifični analitički upiti
Pristup	Stotinama izvještaja koje se može uređivati ili iščitavati	Milijunima podataka koje se uglavnom može samo iščitavati
Cilj	Ovisno o aplikaciji	Sustav za podršku pri odlučivanju
Podatci	Detaljni; numerički i alfanumerički	Sumirani; uglavnom numerički
Integracija podataka	temeljena na aplikaciji	temeljena na subjektu

Značajke	Operativne baze podataka	Skladišta podataka
Kvaliteta	S obzirom na integritet	S obzirom na konzistenciju
Obrađeni podatci s obzirom na vrijeme	Trenutačni podatci	Trenutačni i povijesni podatci
Ažuriranja	Kontinuirana	Periodička
Modeli	Normalizirani	Denormalizirani, višedimenzionalni
Optimizacija	OLTP pristup dijelu baze	OLAP pristup većini baze

Tablica 2: Usporedba operativnih baza podataka i skladišta podataka[4]

2.2. Podatkovni centri

Skladišta podataka uglavnom manipuliraju velikim količinama podataka. Međutim, podatkovne analize trebaju podatke koji se lako pronalaze te koji su spremni za dohvaćanje. Menadžer, ili druga poslovna osoba, ne bi trebao pisati kompleksne upite samo kako bi došao do podataka koji mu trebaju za pisanje izvještaja. U tom trenutku podatkovni centri (engl. *data marts*) olakšavaju posao (vidi sliku 2).

Podatkovni centri su subjektno orijentirane baze podataka koje najčešće predstavljaju određeni segment ili granu samog poduzeća. Podskup podataka koji se nalazi u podatkovnim centrima uglavnom odgovara određenoj poslovnoj jedinici poput prodaje, financija ili marketinga. Podatkovni centri omogućavaju ubrzavanje procesa poslovanja omogućavajući pristup relevantnim informacijama u skladištu podataka unutar par dana, umjesto unutar par mjeseci ili duže zato što sadrže podatke koji se odnose samo na jedan dio poslovanja.

[4] Prilagođeno prema izvoru (Golfarelli & Rizzi, 2009:6).

Slika 2: Podatkovni centri[5]

Podatkovni centri se mogu konstruirati iz postojećih skladišta podataka (pristup odozgora prema dolje (engl. *top down*)) ili iz drugih izvora poput unutarnjih operativnih sustava ili vanjskih podataka. Slično samom skladištu podataka, podatkovni centri su relacijske baze podataka koje pohranjuju transakcijske podatke (vremenske vrijednosti, numerički poredak, reference prema određenim podatcima ili zbirkama podataka) u stupce i redove olakšavajući organizaciju i pristup. Određene odvojene poslovne jedinice mogu stvarati svoje podatkovne centre bazirane na vlastitim podatkovnim potrebama koje spajanjem mogu rezultirati kreiranjem jedinstvenog skladišta podataka (pristup odozdo prema gore (engl. *bottom-up*)).

Postoje tri glavna tipa podatkovnih centara:

- ovisni podatkovni centri (engl. *Dependent data marts*) – stvaraju se ekstrakcijom podataka direktno iz operativnih, vanjskih ili oba izvora. Nude mogućnost centralizacije. Mogu se izgraditi na dva različita načina. U prvom slučaju korisnik može pristupiti podatkovnim centrima i skladištu podataka, ovisno o potrebi. Drugi slučaj nije pretjerano optimalan te se ujedno naziva podatkovno odlagalište (engl. *data junkyard*); svi podatci započinju sa zajedničkim izvorom, ali se postupno razdvajaju i bivaju odbačeni,

[5] Prilagođeno prema izvoru (https://www.talend.com/resources/what-is-data-mart/)

- neovisni podatkovni centri (engl. *Indepentent data marts*) – kreirani bez korištenja centralnog skladišta podataka, te je ovakav pristup idealan za manje grupacije unutar organizacije. U ovome pristupu podatkovni centri nemaju nikakvu vezu s podatcima poduzeća ili drugim centrima. Podatci se unose zasebno te se analize ujedno vrše autonomno. Ovaj pristup se protivi duhu samog skladištenja podataka jer sam cilj poduzeća je centraliziran podatkovni sustav s čijim se raznolikim podatcima mogu koristiti zaposlenici ili korisnici s obzirom na potrebe u danom trenutku,
- hibridni podatkovni centri (engl. *hybrid data marts*) – koriste se i unose podatci osim onih spremljenih u skladištima podataka. Najkorisnije je za sustave koji posjeduju više baza podataka i zahtijevaju brzu implementaciju.

Pozitivne značajke vezane uz podatkovne centre se odnose na brži pristup potrebnim podatcima, sadržavanje skupova podataka bitnih za određene ljude u firmi, kao jeftinija opcija u usporedbi sa skladištima podataka koja mogu s vremenom postati poprilično trošak. Nezanemarive značajke su i jednostavnost korištenja zbog dizajniranja po potrebama određenog poduzeća, sveukupno kraće vrijeme implementacije za razliku od skladišta podataka zbog fokusa na manje skupine podataka, te sadržavanje povijesnih podataka koji omogućuju predviđanje ili određivanje trendova u poslovanju.

S druge strane, negativne se strane uglavnom fokusiraju na greške korisnika koji stvaraju previše zasebnih podatkovnih centara s nevažnim podatcima koji ne služe svrsi, stvarajući tako hrpu zasebnih jedinica koju je teško održavati, te podatkovni centri ne pružaju cjelokupne analize svih podataka poduzeća pošto su im podatci limitirani.

3. Arhitekture skladišta podataka

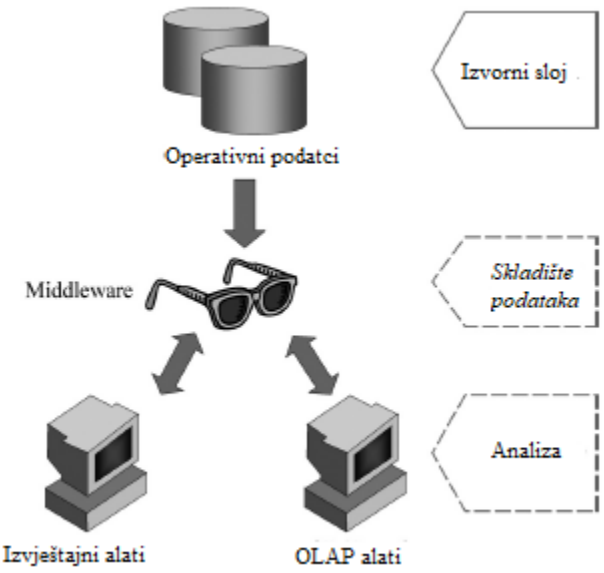
Upoznavši se s nastankom skladišta podataka i konceptom podatkovnih centara, naglasak nadalje stavljamo na arhitekture samih skladišta koje definiraju niz komponenti poput brzine i optimizacije sustava što dugoročno gledano definira njihovu korištenost i popularnost na tržištu. Sa samim skladištima podataka ujedno su se razvijali i drugačiji tipovi arhitektura radi pospješivanja samih funkcionalnosti skladišta koje su se bazirale na najbitnijim elementima skladišta podataka:

- **separacija** – analitički i transakcijski procesi bi se trebali držati podalje jedni od drugih što je više moguće (razlog tome leži u višku informacija nepotrebnih za analitičku analizu od strane menadžera ili voditelja unutar firme),
- **skalabilnost** – hardverske i softverske arhitekture bi trebalo biti lako nadograditi s obzirom kako volumen podataka, koji treba biti obrađen i pohranjen, i broj korisničkih potreba progresivno raste,
- **proširivost** – arhitektura treba imati mogućnost uključivanja i korištenja drugih aplikacija ili tehnologija bez da se cijeli sustav treba redizajnirati,
- **sigurnost** – nadgledanje svih pristupa sustavu je bitno zbog strateških informacija pohranjenih u skladištima podataka,
- **administrativnost** – korištenje i rukovanje skladištem podataka ne bi trebalo biti pretežito komplicirano.

Postoje uglavnom dva različita tipa klasifikacije arhitektura skladišta podataka. Prva klasifikacija, koja će se odnositi na prva tri potpoglavlja, se bazira na strukturnu podjelu, dok će četvrto potpoglavlje prikazati različite slojeve koji se koriste pri kreiranju korporativno-orijentiranih ili odjelno-orijentiranih pregleda skladišta podataka (Golfarelli & Rizzi, 2009).

3.1. Jednoslojna arhitektura

Jednoslojna arhitektura (slika 3) se uglavnom ne koristi u svakidašnjici. Cilj navedene arhitekture je umanjivanje količine podataka koja se sprema; kako bi to postigla, uklanja suvišne podatke. U jednoslojnoj arhitekturi, jedini fizički sloj koji je dostupan je izvorni sloj (engl. *source layer*), a skladište podataka je virtualno.



Slika 3: Jednoslojna arhitektura[6]

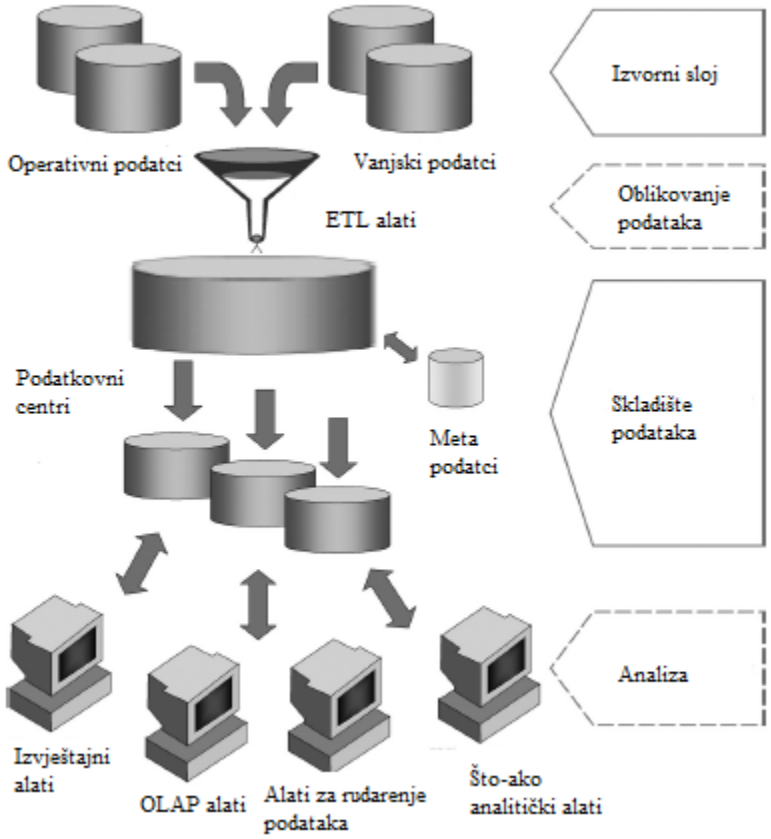
Slabost ove arhitekture leži u tome što ne ispunjava uvjet separacije između analitičkog i transakcijskog procesiranja. Analitički upiti su predani operativnim podatcima nakon što ih međuprogram interpretira, a na taj način upiti utječu na regularan transakcijski dio. Iz tih razloga virtualan pristup skladištima podataka može biti uspješan samo kad su analize ograničenoga tipa te su količine podataka za analizu goleme (Golfarelli & Rizzi, 2009).

[6] Prilagođeno prema izvoru (Golfarelli & Rizzi, 2009:7).

3.2. Dvoslojna arhitektura

Iako se naziva dvoslojnom arhitekturom radi naglaska na odvajanje fizički dostupnih izvora i skladišta podataka, zapravo se sastoji od četiri faza podataka:

1. Izvorni sloj (engl. *source layer*) – sustav skladištenja podataka koristi heterogeni izvor podataka. Ti podatci se čuvaju u bazama podataka poduzeća ili mogu doći iz informacijskih sustava izvan samog poduzeća.
2. Izvođenje podataka (engl. *data staging*) – podatci pohranjeni u izvore trebaju biti izvedeni, očišćeni radi uklanjanja nestalnosti i popunjavanja rupa u podatcima, i integrirani radi spajanja heterogenih izvora u jednu zajedničku shemu. Takozvani ETL alati (engl. *Extraction, Transformation and Loading*) odnosno alati za ekstrakciju, transformaciju i učitavanje, mogu spajati heterogene sheme, izdvajati, transformirati, očistiti, ovjeriti, filtrirati i učitati podatke u skladište podataka.
3. Skladište podataka – centraliziran repozitorij podataka. Može mu se direktno pristupiti ili može biti korišten kao izvor za kreiranje podatkovnih centara koji djelomično repliciraju podatke iz skladišta podataka te su specijalizirani za određene odijele određenog poduzeća. Sami repozitoriji metapodataka spremaju podatke o izvorima, procedurama samog pristupa podatcima, izvođenju podataka, korisnicima, shemama podatkovnih centara itd.
4. Analiza – U ovome sloju, integriranim podatcima se efikasno i fleksibilno može pristupiti radi stvaranja izvještaja, dinamičke analize informacija, i simuliranja hipotetskih poslovnih scenarija. Podržava navigatore za zbirne skupine podataka, kompleksne sustave za optimizaciju upita, te razumljiva grafička korisnička sučelja (engl. *GUI*).



Slika 4: Dvoslojna arhitektura[7]

Skladište podataka, koje koristi dvoslojnu arhitekturu vidljivu na slici 4, odvaja izvore od aplikacija za analizu podataka, a uključuje pogodnosti kao što su (Golfarelli & Rizzi, 2009):

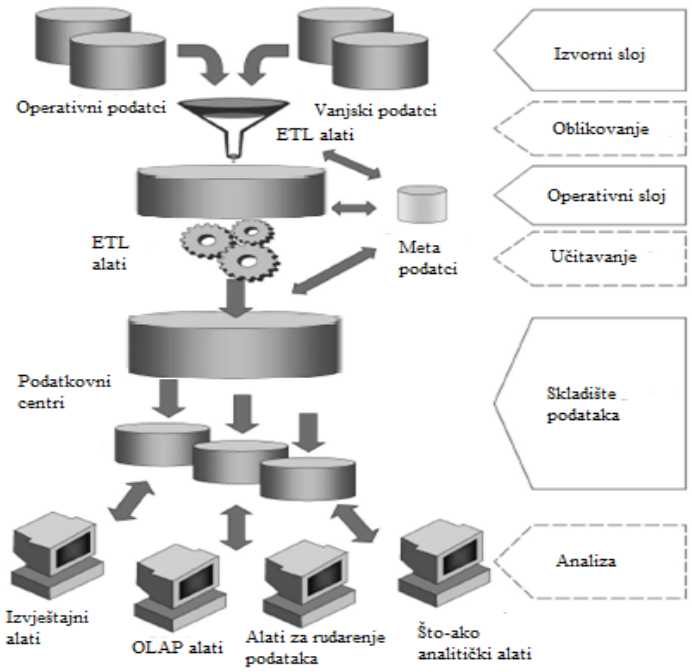
- u sustavu skladišta podataka se uvijek nalaze podatci dobre kvalitete, čak i kad je pristup izvorima onemogućen neko vrijeme zbog tehničkih ili organizacijskih razloga,
- upiti analize skladišta podataka ne utječu na upravljanje transakcijama što je bitno zbog kvalitetnog rada poduzeća na operativnom dijelu,

[7] Prilagođeno prema izvoru (Golfarelli & Rizzi, 2009:9).

- skladišta podataka su logički strukturirana prema višedimenzionalnom modelu, dok su operativni izvori uglavnom bazirani na relacijskim ili polustrukturiranim modelima,
- skladišta podataka mogu koristiti specifična dizajnirana rješenja ciljana na optimizaciju performansi analize i aplikacija za izdavanje izvještaja.

3.3. Troslojna arhitektura

U troslojnoj arhitekturi, vidljivoj na slici 5, treći sloj je operativni sustav pohrane podataka (engl. *operational data store* ili *reconciled data layer*). Nakon što su izvorni podatci integrirani i očišćeni, ovaj sloj materijalizira operativne podatke. Rezultat toga su integrirani, dosljedni, točni, trenutačni i detaljni podatci.



Slika 5: Troslojna arhitektura[8]

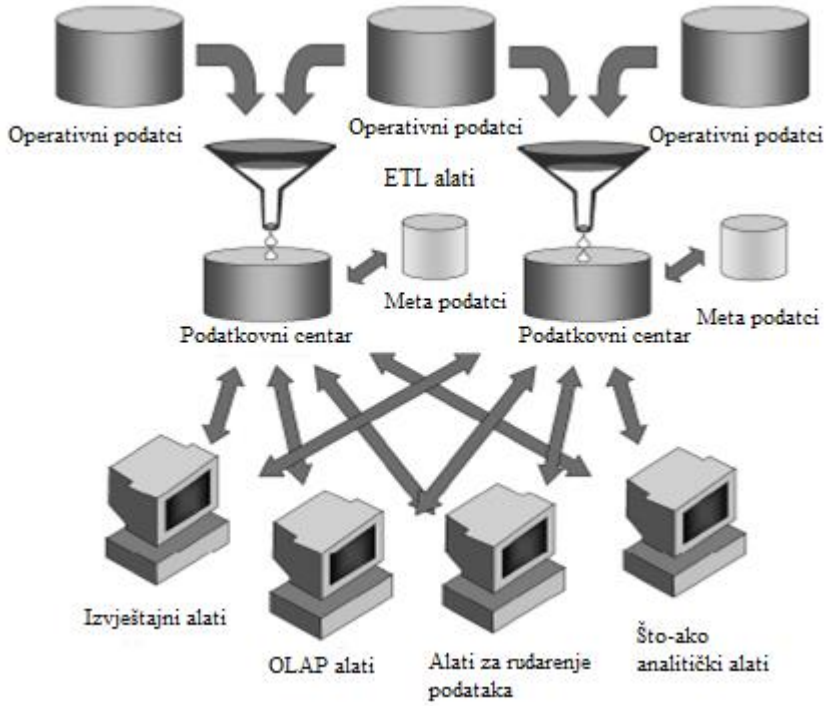
[8] Prilagođeno prema izvoru (Golfarelli & Rizzi, 2009:11).

Glavna prednost trećeg sloja je ta što kreira zajednički referentni podatkovni model za cijelo poduzeće, te ujedno odvaja probleme izvlačenja izvornih podatak i integraciju s postojećim u skladištu podataka. Ujedno dodatni sloj nekad služi kako bi bolje postigao neke operativne zadatke, kao što su produciranje dnevnih izvještaja koji se teško stvaraju i izvode koristeći aplikacije poduzeća, ili periodičnim generiranjem protoka podataka za vanjske procese. Međutim jedan od problema je taj što dolazi do veće redundancije podataka (Golfarelli & Rizzi, 2009).

3.4. Dodatna arhitekturalna klasifikacija

Uz glavne navedene strukture, ujedno postoji dodatna „hibridna“ arhitektura skladišta podataka između jednoslojne arhitekture i dvoslojne ili troslojne arhitekture. Ovaj pristup odnosno arhitektura uzima u obzir da iako je skladište podataka dostupno, nije sposobno obraditi i riješiti sve upite koji su formulirani. To znači kako su korisnici iz tog razloga zainteresirani pristupanju izvornim podatcima direktno iz zbirnih podataka, odnosno podatkovnih skupina. Kako bi se postigao taj cilj, određeni upiti moraju biti prerađeni na osnovi izvornih podataka.

U *arhitekturi neovisnih podatkovnih centara* (engl. *independent data marts*), kao što smo već naveli, različiti podatkovni centri su dizajnirani odvojeno i izgrađeni bez integriranja, kao što pokazuje slika 6. Poduzeća koja nemaju za cilj stvaranje velikih skladišta podataka mogu težiti ovakvoj strukturi ili kad su različiti poslovni dijelovi firme slabije povezani odnosno nisu toliko ovisni jedni o drugima. Međutim, najčešće s vremenom takve strukture bivaju zamijenjene s arhitekturama koje bolje pridonose integraciji podataka i lakšem međusobnom poslovnom izvještavanju.

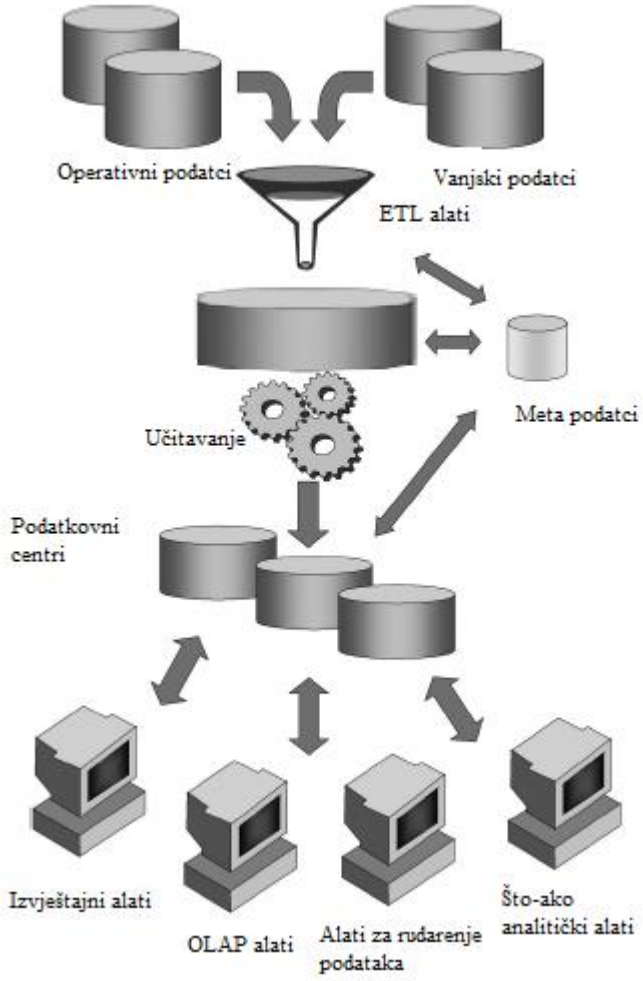


Slika 6: Neovisna arhitektura podatkovnih centara[9]

Bus arhitektura (engl. *bus architecture*) je dosta slična neovisnoj arhitekturi, no razlikuje se po jednoj stvari; skup valjanih dimenzija (engl. *conformed dimensions*), izvedenih pažljivom analizom glavnih procesa poduzeća, je prisvojen i podijeljen kao zajednička smjernica. To osigurava logičku integraciju podatkovnih centara, te pregled informacija na razini cijeloga poduzeća.

U *centralno-razgranatoj arhitekturi* (engl. *hub-and-spoke*) naglašeni su elementi skalabilnosti, proširivanja i pružanja pregleda svih informacija na razini poduzeća. Normalizirani podatci su spremljeni u operativnom sustavu pohrane podataka (engl. *reconciled layer*) te je povezan s određenim brojem podatkovnih centara koji sadrže sažete obrađene podatke u višedimenzionalnome modelu, kao što je prikazano na slici 7.

[9] Prilagođeno prema izvoru (Golfarelli & Rizzi, 2009:12).



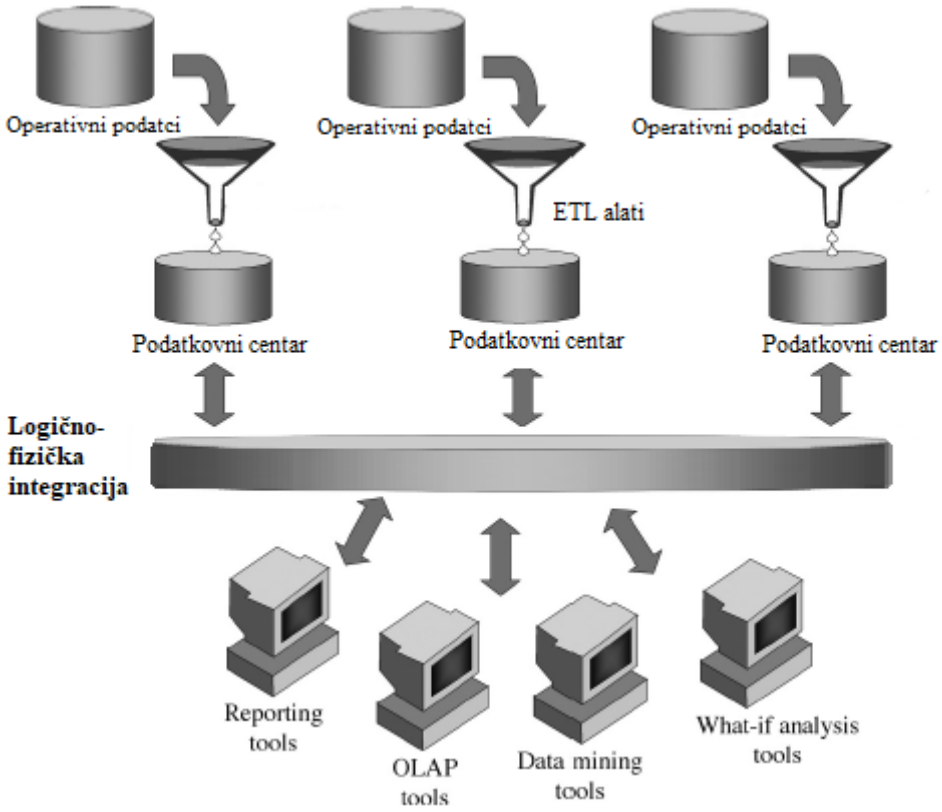
Slika 7: Centralno razgranata arhitektura[10]

Centralizirana arhitektura (engl. *centralized structure*) je određena vrsta implementacije centralno razgranate arhitekture, gdje se operativni sloj i podatkovni centri stapaju skupa u jedinstveni fizički repozitorij.

Federativna arhitektura (engl. *federated architecture*), prikazana na slici 8, se često koristi u dinamičkome kontekstu gdje postojeća skladišta podataka ili/i podatkovni centri moraju biti integrirani kako bi stvorili jedan sustav za pomoć pri donošenju odluka za cijelo poduzeće (slika 9). Svako skladište podataka ili svaki centar je fizički spojen s drugima,

[10] Prilagođeno prema izvoru (Golfarelli & Rizzi, 2009:13).

oslanjajući se na niz naprednih tehnika poput distribuiranja upita, ontologije i interoperabilnosti metapodataka (Golfarelli & Rizzi, 2009).



Slika 8: Federativna arhitektura[11]

[11] Prilagođeno prema izvoru (Golfarelli & Rizzi, 2009:14).

4. Višedimenzionalnost

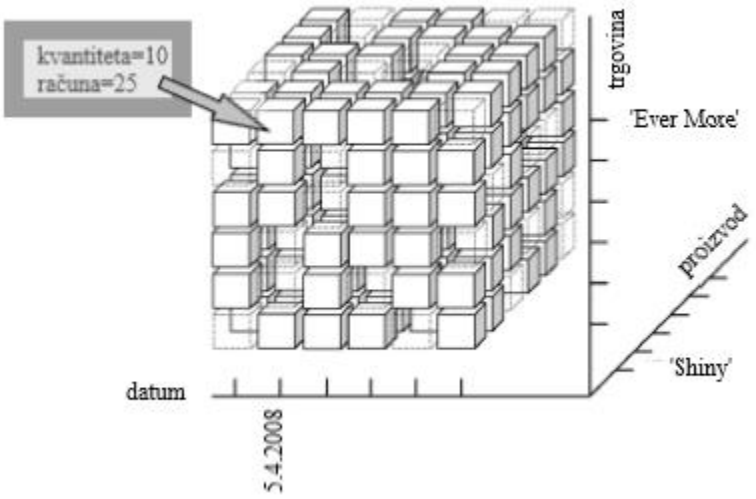
Shvativši neke od osnovnih arhitektura sustava za skladištenje podataka, fokusiramo se na jednu od bitnijih tema rada i bitnijeg svojstva navedenih sustava – načina pohrane dobivenih podataka radi lakše ekstrakcije informacija bitnih za poslovanje poduzeća; korak bitan kako bi u cijelosti shvatili na koji način skladišta podataka funkcioniraju.

U posljednjih nekoliko godina, višedimenzionalno modeliranje je zaokupilo interes javnosti i poslovnog svijeta. Podrazumijeva jaku tehniku konceptualizacije korištene u OLAP (*On-Line Analytical Processing*) aplikacijama. OLAP alati, koristeći višedimenzionalno modeliranje, olakšavaju kompleksne analize i vizualizaciju podataka u skladištu podataka radi procesa donošenja odluka. Jednostavnost korištenja za ljude koji nisu toliko stručni po pitanju informacijskih tehnologija, sličnost analitičkom tipu razmišljanja, pospješivanje performansi upita su jedne od glavnih vrlina i karakteristika OLAP alata te je potaknulo prihvaćanje višedimenzionalnog modela kao idealnog rješenja (Abello A. & Samos J. & Saltor F., 2000).

Korištenjem tradicionalnih jezika, poput *SQLa*, radi definiranja upita može prerasti u poprilično zahtjevan zadatak za korisnike bez informatičkog iskustva, a pokretanje i obrada tih upita može vremenski potrajati. Višedimenzionalni model započinje zaključivanjem da su faktori koji utječu na procese koji donose odluke u sustavu činjenice (engl. *facts*) specifične za poduzeće poput prodaje, isporuke, operacija u bolnici itd. Instance činjenica korespondiraju s događajima (engl. *events*) koji se pojavljuju. Primjerice, svako *uspjelo slane pošiljke* je događaj. Svaka činjenica je opisana mjerilima kolekcije relevantnih mjera (engl. *measures*) koje pružaju kvantitativni opis događaja; *računi o prodaji, količina poslane robe, cijena boravka u bolnici* i *trajanje operacije* su mjere.

Koncept dimenzija je ilustriran najlakše kroz pojam kocke radi prezentiranja višedimenzionalnog modela. Prema toj metafori, događaji su povezani s ćelijama kocke i

njenim rubovima koji predstavljaju analizu dimenzija. Ako postoje više od tri dimenzije, kocka se naziva hiperkockom (engl. *hypercube*). Svaka ćelija dobiva vrijednost za svaku mjeru. Slika 9 prikazuje primjer kocke u kojoj je činjenica prodaja u lancu trgovina. Analize dimenzija u ovom slučaju su trgovina, proizvod i datum. Događaj predstavlja prodaju određenog proizvoda u određenoj trgovini na određeni datum, te je opisan kroz dvije mjere: količina prodanog i računi. Ujedno slika prikazuje prorijeđenost (engl. *sparse*) kocke – jer velik broj događaja se nije dogodio, odnosno ne može se prodati svaki proizvod svakoga dana u svakoj trgovini.



Slika 9 : Trodimenzionalna kocka koja prikazuje prodaju u lancu trgovina: 10 komada "Shiny-a" je prodano 5.4.2008 u "EverMore" trgovini za 25 dolara[12]

Relacijska shema bi izgledala otprilike ovako:

PRODAJA (trgovina, proizvod, datum, kvantiteta, računi)

U ovom slučaju podcrtani pojmovi tvore primarni ključ, a događaji su asocirani s torkama, poput < 'EverMore', 'Shiny', 5.4.2008, 10, 25>. Definiranjem primarnoga ključa zaključujemo kako dva događaja ne mogu biti povezana s individualnom trgovinom,

[12] Prilagođeno prema izvoru (Golfarelli & Rizzi, 2009:20).

proizvodom i datumom, te da svaka vrijednost tih kombinacija *funkcionalno određuje* (engl. *functionally determines*) jedinstvenu vrijednost za *kvantitetu* i jedinstvenu vrijednost za *račune*. To znači da je funkcionalna ovisnost definirana kao:

trgovina, proizvod, datum ➔ *kvantiteta, računi*

Grupa odabranih dimenzija radi prikazivanja činjenice izdvaja jedinstveni događaj u višedimenzionalnom modelu, ali ta ista grupacija ne mora nužno izdvajati jedinstveni događaj u samoj domeni aplikacije. Primjerice, u domeni aplikacije jedinstveni događaj koji uključuje *prodaju* bi trebala biti korisnikova kupovina niza proizvoda u trgovini na određeni datum. U praksi, to je prikazano na računu. Iz perspektive višedimenzionalnog modela, ako činjenica *prodaje* sadrži dimenzije *proizvoda*, *trgovine* i *datuma*, događaj će prezentirati cjelokupnu dnevnu prodaju određenog proizvoda u dućanu. Vidljivo je kako razlika između obje interpretacije ovisi o računima prodaje koji uglavnom uključuju različite proizvode i individualne proizvode koji se često prodaju više puta dnevno u dućanu.

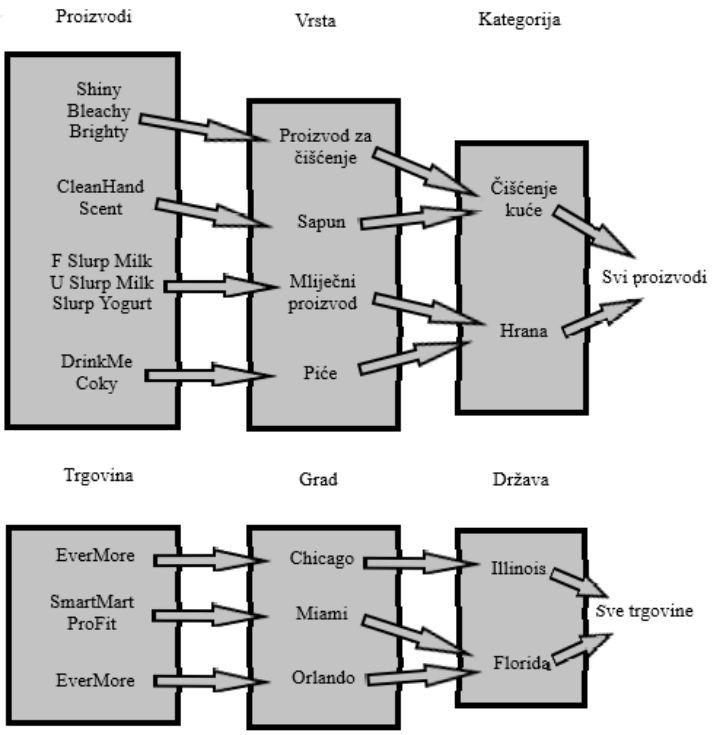
Svaka dimenzija je asocirana s *hijerarhijom* razina agregacije; uzlazna hijerarhija (engl. *roll up*). Uzlazne hijerarhije grupiraju vrijednosti razina agregacije na drugačije načine. Hijerarhije se sastoje od razina nazvanih *dimenzionalnim atributima.* Slika 10 prikazuje jednostavan primjer hijerarhija izgrađenih na dimenzijama *proizvoda* i *trgovine*: proizvodi su podijeljeni u tipove, te su dodatno klasificirani u kategorije. Trgovine su locirane u gradovima koje pripadaju određenim državama u SAD-u. Iz perspektive relacijske teorije, možemo koristiti set funkcionalnih ovisnosti između atributa dimenzija radi izražavanja hijerarhije:

proizvod ➔ *vrsta* ➔ *kategorija*

trgovina ➔ *grad* ➔ *država*

Višedimenzionalna kocka prikazuje set *događaja* čije numeričke vrijednosti prikazuju kvantitativni opis. Svaka osovina kocke prikazuje potencijalnu analizu *dimenzija*. Svaka

dimenzija se može proučavati na detaljnijoj razini specificiranoj pomoću hijerarhijski strukturiranih atributa.



Slika 10: Agregacijska hijerarhija izgrađena na dimenzijama proizvoda i trgovine[13]

Korisnicima je poprilično teško održavati informacije u višedimenzionalnoj kocki zbog njihove količine, pa čak i sažetu verziju informacija pohranjenih u operativnim bazama. Primjerice, ako lanac trgovina sadrži 50 trgovina koje prodaju 1000 proizvoda i određeno skladište podataka pokriva transakcije do tri godine unazad (okvirno 1000 dana), broj potencijalnih događaja je 50 * 1000 * 1000 = $5*10^7$. Uzevši u obzir da dućan može prodati svega 10 posto svih dostupnih proizvoda u danu, broj događaja pada na $5*10^6$, što je i dalje prevelik broj podataka za analizu. Iz tog razloga postoje svega dva načina za sužavanjem broja i količine podataka te dohvaćanje korisnih informacija: *restrikcija* i *agregacija* (Golfarelli & Rizzi, 2009) o kojima će biti više riječi u nastavku.

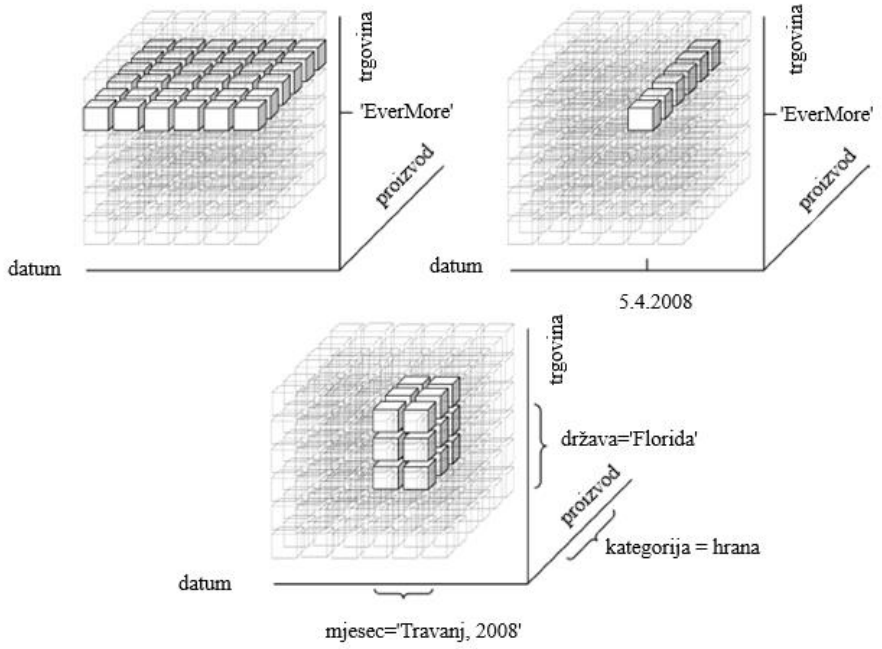
[13] Prilagođeno prema izvoru (Golfarelli & Rizzi, 2009:21).

4.1. Restrikcija

Restrikcija se odnosi na izdvajanje određene količine podataka iz kocke radi označavanja polja za analizu. U relacijskoj algebri to se zove *selektiranje* i/ili *projekcije*.

Najjednostavniji način selekcije je rezanje podataka (engl. *data slicing*). Rezanjem podataka smanjuju se dimenzije kocke postavljanjem jedne ili više dimenzija u određenu vrijednost. Primjerice, postavljanjem određene dimenzije kocke kao što je *trgovina='EverMore'*, dobiveni rezultati prikazuju samo događaje asocirane s proizvodima prodanim u *EverMore* trgovini. Postavljanjem datuma i trgovine dobit ćemo sve proizvode prodane u navedenom dućanu na navedeni datum. Dodavanjem određenih vrijednosti svim dimenzijama, definirati ćemo samo jedan događaj koji korespondira određenom trenutku u trodimenzionalnome prostoru prodaja.

Kubiciranje (engl. *dicing*) je generalizacija rezanja (engl. *slicing*). Postavlja određena ograničenja na trodimenzionalne atribute umanjujući veličinu kocke. Primjerice, odabirom dnevne prodaje hranidbenih proizvoda u travnju 2008. u Floridi (slika 11). Ovim putem, ako je pet dućana locirano u Floridi u kojima je prodano 50 hranidbenih proizvoda, broj događaja koji se treba pregledati je 5 * 50 * 30 = 7500; iz tog razloga projekcija se može definirati kao odabir kojim se zadržava samo jedna podgrupa mjera za svaki događaj odbijajući sve ostale mjere.



Slika 11: tehnike rezanja i kubiciranja nad trodimenzionalnom kockom[14]

4.2. Agregacija

Jedan od najboljih načina za poboljšanje performansi u većem skladištu podataka je implementacija određenih setova agregacijskih mjera koje koegzistiraju s primarnim baznim zapisima. Stvaranje takvog agregatnog programa može biti realizirano skoro u svakom hardveru skladišta podataka ili softverskoj konfiguraciji, uključujući popularne relacijske DBMS-ove kao što su Oracle, Red Brick, Informix, Sybase i DB2 (Kimbal R., 1996). Agregacija igra ključnu ulogu u višedimenzionalnim bazama podataka. Kad bi, na primjer, htjeli analizirati mjesečnu prodaju proizvoda kroz period od tri godine, trebali bi sortirati sve ćelije vezane uz dan svakog mjeseca po proizvodu i trgovini, te ih onda spojiti u jednu makroćeliju.

[14] Prilagođeno prema izvoru (Golfarelli & Rizzi, 2009:23).

	EverMore	EvenMore	SmartMart
1.1.2008	–	–	–
2.1.2008	10	15	5
3.1.2008	20	–	5
.....	.....	.....	.....
1.1.2009	–	–	–
2.1.2009	15	10	20
3.1.2009	20	20	25
.....	.....	.....	.....
1.1.2010	–	–	–
2.1.2010	20	8	25
3.1.2010	20	12	20
.....	.....	.....	.....

	EverMore	EvenMore	SmartMart
Siječanj 2008	200	180	150
Veljača 2008	180	150	120
Ožujak 2008	220	180	160
.....	.....	.....	.....
Siječanj 2009	350	220	200
Veljača 2009	300	200	250
Ožujak 2009	310	180	300
.....	.....	.....	.....
Siječanj 2010	380	200	220
Veljača 2010	310	200	250
Ožujak 2010	300	160	280
.....	.....	.....	.....

	EverMore	EvenMore	SmartMart
2008	2,400	2,000	1,600
2009	3,200	2,300	3,000
2010	3,400	2,200	3,200

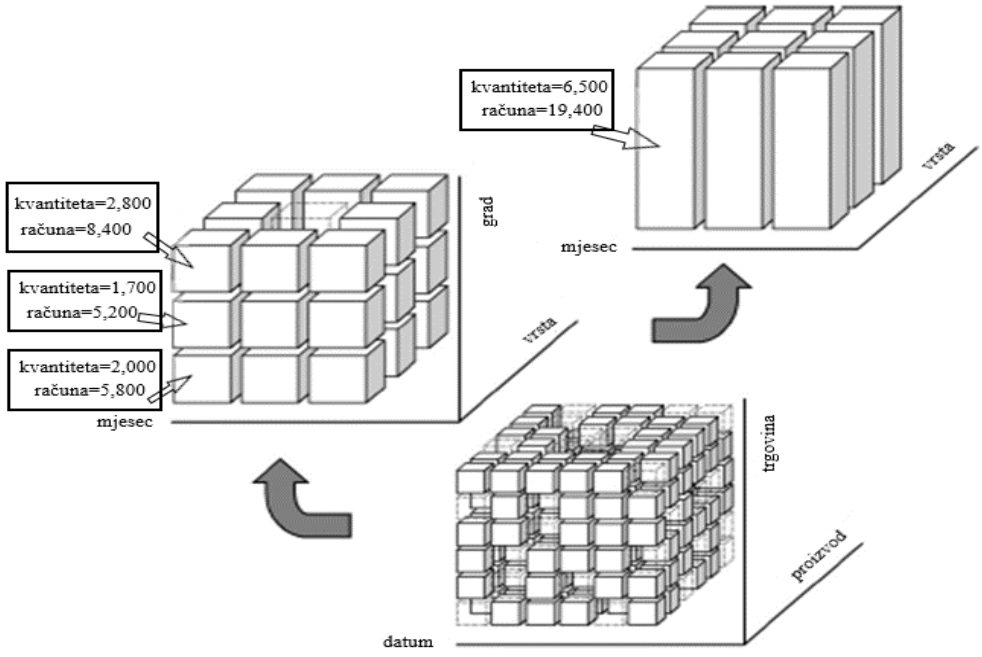
	EverMore	EvenMore	SmartMart
Ukupno	9,000	6,500	7,800

Slika 12: Vremenska hijerarhija agregacije kvantitete proizvoda prodanih pojedinačno u tri različite trgovine. Povlaka označava da proizvod nije prodan[15]

[15] Prilagođeno prema izvoru (Golfarelli & Rizzi, 2009:24).

U agregacijskoj kocki stvorenoj ovim načinom, sveukupni broj događaja (odnosno broj makroćelija) je 50 * 1000 * 36 (36 označava broj mjeseci u tri godine). U navedenom primjeru, sveukupan broj prodanih proizvoda u mjesecu i sveukupan broj računa je dobiven zbrajanjem svake vrijednosti dobivenih mjera (slika 12). Daljnjom agregacijom, dobivaju se tri događaja za svaku kombinaciju trgovina-proizvod: jedan za svaku godinu. Potpunom agregacijom kroz dimenzije, svaka kombinacija trgovina-proizvod odgovara jednom događaju, koji pokazuje sveukupan broj prodanih proizvoda u trgovini kroz tri godine i sveukupan broj računa (Golfarelli & Rizzi, 2009).

Agregacija se može provoditi kroz različite dimenzije istovremeno. Primjerice, na slici 13 vidimo kako možemo grupirati prodaju po mjesecu, tipu proizvoda i trgovini u određenome gradu te po mjesecu i tipu proizvoda. Štoviše, selekcija i agregacija se mogu kombinirati radi provođenja procesa analize koji je usmjeren na potrebe korisnika.



Slika 13: Agregacija kocke na više načina[16]

[16] Prilagođeno prema izvoru (Golfarelli & Rizzi, 2009:25).

5. Pristup skladištima podataka

Prikazavši povijest i arhitekturu sustava za skladištenje podatka i način pohrane, možemo se fokusirati na razlog samog nastanka navedenih sustava i potrebe koje ispunjava za različit tip korisnika. Nakon čišćenja, integracije i transformacije podataka, potrebno je odrediti na koji način izvući potrebite i korisne informacije. Zbog toga će ovo poglavlje pokrivati najbolje pristupe za obradu podataka krajnjih korisnika: izvještaje, OLAP i kontrolne ploče.

5.1. Izvještaji

Izvještaji su namijenjeni za korisnike koji moraju pristupati periodički fiksiranim informacijama. Primjerice, bolnica mora slati mjesečni izvještaj o troškovima pacijenata koji se nalaze u bolnici u gradskom ili regionalnome uredu. Izvještaji uvijek imaju istu formu, što znači da dizajner može napisati upit koji generira izvještaj i „zamrznuti“ ga unutar aplikacije kako bi se izvršio po potrebi korisnika. Izvještaj se najčešće asocira s upitom i prezentacijom. Upit najčešće podrazumijeva selekciju i agregaciju višedimenzionalnih podataka pohranjenih u jednu ili više činjenica. Prezentacija može biti u tabličnoj ili grafičkoj formi (poput dijagrama). Većina alata za izvještaje ujedno dopušta automatsku distribuciju periodičkih izvještaja zainteresiranim korisnicima putem e-maila ili dijeljenje izvještaja preko intranet servera poduzeća radi preuzimanja izvještaja (Rizzi, 2009).

Izvještaji su oduvijek bili glavni alat za menadžere radi evaluacije i planiranja zadataka u bazama podataka. Međutim, pridodavanje skladišta podataka pridonosi izvještajima iz dva razloga. Prvo, koriste pouzdane i točne rezultate jer podatci koji su sumirani u izvještajima konzistentni su i integrirani. Ujedno skladišta podataka ubrzavaju proces izvještavanja jer strukturalno odvajanje između transakcijskih procesa i analiza znatno poboljšava performanse.

5.2. OLAP

OLAP (engl. *online transaction processing system*) je glavni korisnik informacija u skladištu podataka. Dok korisnici alata za izvještaje uglavnom igraju pasivnu ulogu, OLAP korisnici mogu započeti aktivno s kompleksnom analitičkom seansom, gdje je svaki korak rezultat prijašnjih koraka. Korištenje OLAP sustava u stvarnom vremenu zahtjeva dubinsko poznavanje podataka, kompleksne upite i primjeren dizajn za korisnike koji nisu IT stručnjaci radi lakšeg korištenja. GUI tih alata mora biti fleksibilan, lako upotrebljiv i efektivan.

OLAP seansa se sastoji od *navigacijskog puta* koji odgovara procesu analize za činjenice s obzirom na drugačija stajališta i drugačije razine detalja. Taj put se pretvara u sekvencu upita koji se najčešće ne izdaju direktno, već su diferencijalno izraženi s referencama na prijašnji upit. Rezultati upita su višedimenzionalni, dok su rezultati alata OLAP sustava uglavnom prikazani dijagramima, s više zaglavlja, boja i drugih svojstva radi naglašavanja dimenzija. Svaki korak analize je karakteriziran *OLAP operatorom* koji pretvara zadnji upit u sasvim novi. Najpoznatiji su uzlazni operater, operater bušenja prema dolje, operater kubičnog-rezanja, pivot i bušenja poprijeko.

- **uzlazna tehnika** (engl. *roll-up*) uključuje sažimanje podataka kroz dimenziju. Pravilo postavljeno za sažimanje može biti agregatna funkcija, primjerice primjenjivanje formula poput „profit = prodaja - troškovi“. Ujedno dimenzionalno rotiranje redova i stupaca ili premještanje jednog reda u stupac podrazumijeva uzlazni proces. Primjer prvog slučaja bi bio uzimanje izvještaja koji ima *Vrijeme* poprijeko (*stupaca*) i *Proizvod* dolje (*redovi*) te rotirajući navedeni izvještaj kako bi imao *Proizvod* poprijeko i *Vrijeme* dolje. Primjer drugog slučaja bi podrazumijevalo izmjenjivanje izvještaja koji sadrži *Mjere* i *Proizvode* dolje i *Vrijeme* poprijeko u izvještaj koji sadrži *Mjere* dolje i *Vrijeme* iznad *Proizvoda* poprijeko,

- **okomito bušenje** (engl. *drill-down*) je specifična tehnika gdje korisnik navigira kroz razine podataka koji gdje se nalaze svi od najsažetijih podataka (*up*) sve do najdetaljnijih (*down*). Putevi kojima će se „bušiti“ odnosno prolaziti mogu biti unaprijed definirani hijerarhijama unutar dimenzija ili drugim vezama koje mogu biti dinamičke unutar ili između dimenzija. Primjerice, gledajući prodaju u Sjevernoj Americi, operacija okomitog bušenja u dimenziji *Regije* bi prikazala *Kanadu*, *istočne Sjedinjene Američke Države* i *zapadne Sjedinjene Američke Države*. Daljnjim bušenjem *Kanade* mogli bismo dići do *Toronta*, *Vankuvera*, *Montreala* itd.,
- **rezanje** (engl. *slice*) je podskup višedimenzionalnog reda koji odgovara jednoj vrijednosti jednog ili više članova dimenzija koje se ne nalaze u podskupu, odnosno rezanje je operacija koja smanjuje broj dimenzija kocke postavljanjem jedne od dimenzija na određenu vrijednost. Kubiciranje generira podkocku biranjem dvije ili više vrijednosti iz više dimenzija kocke. Kocka se rotira neovisno na dimenzije,
- **pivotiranje** ili rotiranje se odnosi na izmjenu dimenzionalne orijentacije određenog izvještaja ili prikazane stranice s podatcima. Rotiranje se može sastojati od izmjenjivanja *redova* i *stupaca*, ili mijenjanje određenoga *dimenzionalnog reda* s *dimenzionalnim stupcem*. Cilj pivotiranja je analiza individualne grupe informacija iz drugačijeg gledišta (OLAP Council, 1995),
- **bušenje poprijeko** (engl. *drill-across*) se odnosi na kreiranje poveznice između dvije ili više kocki, koje su u određenom međusobnom odnosu, radi proučavanja njihovih podataka. Primjerice, to se odnosi na izračunavanje izraza koji uključuje mjere dviju kocki poput kocke *Prodaja* koju preklapamo s kockom *Promocije* kako bi usporedili prihode i popuste po tromjesečju i po kategoriji proizvoda.

5.3. Kontrolna ploča

Kontrolne ploče su jedne od metoda koje služe za prikaz informacija pohranjenih u skladištima podataka. Odnose se na GUI koji prikazuje određenu količinu relevantnih podataka u sažetom i lako čitljivom formatu. Kontrolne ploče pružaju prikaz informacija u pravom vremenu o trendovima vezanim uz specifičan fenomen ili za više fenomena koji su povezani jedni s drugima. Koriste ih pretežito viši menadžeri kojima je potreban brz i jednostavan prikaz informacija. Međutim radi prikazivanja kompleksnih fenomena, kontrolne ploče moraju biti povezane s alatima za analizu.

Kontrolne ploče nisu ništa više od indikatora performansi dok je njihova efikasnost rezultat oprezne selekcije relevantnih mjera korištenjem informacija skladišta podataka i određenih standarada kvalitete informacija koje se pohranjuju u sama skladišta. Iz tog razloga kontrolne ploče treba gledati kao sofisticirane ekstenzije skladišta podataka, ali ne kao primarni cilj sustava za skladištenje podataka (Golfarelli & Rizzi, 2009).

6. Kriteriji za odabir sustava za skladištenje podataka

Izmjene u izvorima podataka, volumen i raznolikost, povećana potražnja za boljom analizom podataka i većim pristupom; i razvoj same tehnologije koji je uvelike pridonio sveukupnoj efikasnosti pohrane podataka, pristupu i analizi. Sve navedene promjene utječu na poslovanje određenog poduzeća, točnije, povećavaju potrebu za sve boljim skladištima podataka, sve bržim analizama sakupljenih podataka i što lakše upravljanje izvučenim informacijama. Zbog toga je važno pri odabiru određenog skladišta podataka ustanoviti ispunjava li dugoročno sve kriterije koji su potrebni za što lakše poslovanje određenog poduzeća.

6.1. Pohrana i integracija podataka na jednom mjestu

Polustrukturirani ili ne tradicionalni podatci, poput onih skupljenih preko društvenih mreža, mogu obogatiti uvid analize podataka više i dalje od tradicionalnih podataka, no to zahtjeva drugačiji pristup učitavanju i transformaciji navedenih podataka prije nego samu analizu mogu provesti zaposlenici korporacije ili poduzeća. Velik broj tradicionalnih skladišta podataka žrtvuje performanse ili fleksibilnost kako bi rukovala s takvim podatcima.

Moderno skladište podataka bi trebalo eliminirati potrebu za dizajniranjem i modeliranjem rigidnih, tradicionalnih struktura unaprijed koje zahtijevaju transformaciju polustrukturiranih podataka prije samog učitavanja. Ujedno je optimizacija upita nad tim podatcima dok su još u izvornom stanju nešto što bi moderniji sustavi trebali pokrivati. Samo učitavanje svih podataka na jedno mjesto je krucijalno, no integracija svih tih različitih podataka radi preciznije analitike je nešto sasvim drugo. Moderni sustavi za skladištenje podataka bi automatski trebali integrirati polustrukturirane podatke sa strukturiranim podatcima svojstvenima tradicionalnoj relacijskoj bazi podataka; ne bi

trebala postojati potreba za instalacijom ili konfiguracijom nečega, i podešavanje bi trebalo biti ugrađeno. Najvažnija značajka je ta da se ne bi trebalo plaćati dva odvojena sustava radi rukovanja sa svim podatcima koje poduzeće posjeduje (Kraynak & Baum, 2017).

Relacijske baze podataka funkcioniraju dobro sa strukturiranim podatcima, odnosno podatcima koji se s lakoćom pohranjuju u redove i kolumne. Ako se podatci mogu organizirati u jednu, nešto veću, proračunsku tablicu, tada je relacijska baza podataka idealno rješenje. Suprotno tome ne-relacijske baze podataka iznimno dobro funkcioniraju s velikim količinama polustrukturiranih podataka kao što su emailovi, knjige i sadržaji sa socijalnih mreža (Segment).

6.2. Podržavanje postojećih vještina, alata i ekspertiza

Tradicionalna skladišta podataka su zastarjela zbog tehnološke zastarjelosti od par desetljeća i sustava koji nisu lako prilagodljivi modernim sustavima u oblaku. Ujedno jezik na koji se oslanjaju i kojeg koriste, *SQL*, ostaje kao sama baza te industrije. Iz tog razloga se razvija niz alata za rukovanje podatcima, transformaciju podataka, integraciju, vizualizaciju, poslovnu inteligenciju i analizu radi komunikacije sa *SQL* skladištem podataka.

Tradicionalna skladišta podataka podržavaju *SQL*, ali ne podržavaju sposobnosti potrebne za efektivno pohranjivanje i procesiranje polustrukturiranih podataka. Iz tog razloga se velik broj poduzeća okrenuo prema alternativnim pristupima poput *NoSQL*[17] rješenja. Međutim ograničenja tih sustava predstavljaju novi problem; potrebu za specijaliziranim znanjem i vještinama koja nisu širom dostupna, te mogućnost da ne podržavaju *SQL*.

[17] NoSql, orginalno definiran kao *non SQL ili ne-relacijska baza podataka,* je baza podataka koja služi za pohranu i dohvaćanje podataka. Navedeni tip baze podataka je specifično dizajniran za aplikacije koje koriste velike količine podataka, trebaju fleksibilne podatkovne modele i nisku latenciju.

Moderna skladišta trebaju biti strukturirana s vodećim tehnologijama, no ujedno moraju biti izgrađena na utemeljenim standardima (poput *SQL*a) i kompatibilna s drugim vještinama i alatima koji su učestali u industriji poput Spark, Python i R računalnih jezika (Kraynak & Baum, 2017.).

6.3. Otpornost i oporavak podataka

Rušenje sustava unutar skladišta podataka može uzrokovati gubitak ili nekonzistentnost podataka, zbog toga podatci uvijek moraju biti valjano pohranjeni, dostupni i ažurni. Tradicionalni sustavi štite podatke stvarajući periodičke sigurnosne kopije, koje konzumiraju vrijedne računalne resurse i interferiraju sa zadatcima koji se obavljaju u danom trenutku. Periodičke sigurnosne kopije zahtijevaju dodatno mjesto za pohranu i često ne uključuju najnovije podatke, što rezultira nekonzistentnošću podataka.

Moderna skladišta trebaju održavati sama sebe kad se govori o dugotrajnosti, otpornosti i dostupnosti sustava. Ne bi trebala ometati ikakve zadatke koji se izvršavaju, oslabljivati performanse ili rezultirati nedostupnosti određene usluge zbog stvaranja sigurnosne kopije u pozadini. Ujedno bi trebala pronaći načine da očuvaju podatke bez da ih kopiraju i premjeste negdje drugdje (Kraynak & Baum, 2017).

6.4. Sigurnost podataka

Skladišta podataka u oblaku često stvaraju strahove vezane uz samu sigurnost podataka. Prvi razlog strahova je taj što više aplikacija može međusobno komunicirati prilikom dohvaćanja podataka, te prouzročiti curenje podataka kroz više aplikacija u kojima se sigurnost „ne odnosi“ na slučajno dobivene podatke. Drugi strah se odnosi na virtualnu platformu, odnosno oblak, koja može biti distribuirana preko više fizičkih strojeva, što postavlja pitanje može li druga aplikacija dohvatiti druge ili preostale podatke prilikom migracije (Alibaba Cloud, 2019). Zato moderna skladišta podataka

moraju podržavati kontrolu pristupa temeljenu na ulogama (engl. *role-base access control*) - RBAC. RBAC osigurava da korisnici imaju pristup samo onim podatcima koji su im dopušteni za vidjeti, odnosno za koje imaju valjanu autorizaciju. Poboljšanje takvog sustava je više-faktorska autentikacija - MFA (engl. *multi-factor authentication*). Kad se korisnik ulogira, MFA šalje dodatni zahtjev za potvrdu identiteta, najčešće na mobitel. Time se osigurava kako osoba s ukradenim korisničkim imenom i šifrom ne može pristupiti sustavu.

Upravljanje podatcima osigurava da su podatci određenog poduzeća valjano korišteni, da se svim podatcima rukuje na način da su zaštićeni od provala, te da ih se koristi prema postavljenim regulacijama. Loši podatci mogu dovesti do loših poslovnih odluka, gubitka prihoda i povećanja troškova. Upravitelji podatcima mogu vidjeti kada su podatci korumpirani ili polovični, kada se ne osvježavaju dovoljno kako bi bili relevantni ili kad se analiziraju van danog konteksta

Enkripcija podataka je još jedna od zaštitnih mjera, te upravljanje danim ključem (engl. *key*), odnosno zaštita samog ključa. Kad se podatci kriptiraju, koristit će se određeni ključ radi dekriptiranja istih podataka. Koliko dugo se koristi ključ? Što ako ključ padne u krive ruke? To su sve pitanja koja se moraju uzeti u obzir te primijeniti unaprijed adekvatne mjere zaštite i osiguranja. Primjeri toga su hijerarhijski pristupi omatanja ključa (engl. *key-wrapping*) koji kriptiraju sam ključ ili proces rotacije ključa koji limitira koliko se puta određeni ključ može koristiti. Periodička penetracijska ispitivanja isto tako služe za provjeru sigurnosti samog sustava, a provode ih upravitelji podatcima. Ispitivanja takve vrste se moraju provoditi konstantno i automatski bez da utječu na performanse samog sustava.

6.5. Protok podataka

Protok podataka se uglavnom odnosi na procese ekstrakcije, transformacije i učitavanja podataka koji donose podatke u skladište podataka u formatu koji podržava

upite. Slab protok podataka tjera korisnike, poput analitičara, da troše veliku količinu vremena na čekanje kako bi pristupili podatcima, a ubrzan rast u različitosti, broju i veličini nepovezanih podataka koji dotiču iz različitih izvora samo povećava problem.

Moderna rješenja moraju moći efektivno učitati polustrukturirane podatke u njihovom izvornom formatu i osposobiti ih momentalno za upite bez potrebe za korištenjem drugih sustava, poput *NoSQL*a, radi transformacije podataka. Na taj način se dopušta korisnicima da pristupe podatcima na isti način na koji pristupaju *SQL* bazi podataka. Takva rješenja omogućuju pristup novim podatcima iznimno brzo, umanjujući vremenski proces dohvaćanja i učitavanja podataka s jednog cijelog dana na svega sat vremena (Kraynak & Baum, 2017.).

Aspekti sustava koji su nekad bili manualni trebaju postati automatizirani, te implementacija određenog rješenja ne bi trebala biti komplicirana. Povijesno gledano, sustavi za skladištenje podataka su iznimno kompleksni, skupi i spori, no danas implementacija rješenja treba biti vremenski kratka, konstantno dostupna i koštati kao djelić tradicionalnih sustava. Bitno je znati koliko brzo poduzeće treba prikupiti podatke. To pretežito ovisi o tome koliko se brzo sami upiti izvršavaju, te kako se održava ta brzina u razdobljima visoke obrade i dohvaćanja podataka. Protok podataka i skalabilnost su iz tog razloga usko vezani; performanse rastu zajedno s povećanjem skladišta podataka, odnosno klastera (Segment).

7. Lokalni servisi ili servisi u oblaku

Prilikom odlučivanja koje skladište podataka koristiti, prvo pitanje koje se treba postaviti je gdje će samo skladište biti locirano; u samoj firmi, odnosno poduzeću ili u oblaku. Šesto poglavlje do određene mjere već uzima u obzir tradicionalne sustave uspoređujući ih s današnjim potrebama i sve popularnijim sustavima za skladištenje podataka u oblaku. Zbog tog razloga ćemo se više fokusirati na razlike između dva navedena sustava, te s obzirom na vrijeme u kojem smo i koje nam predstoji, definirati koji sustavi bolje odgovaraju sadašnjim i budućim potrebama poduzeća.

7.1. Usporedba vremena i vrijednosti

Postavljanje i implementacija tradicionalnog skladišta podataka može potrajati najmanje godinu dana i prerasti u popriličan projekt prije prvih izvučenih podataka iz funkcionalnog skladišta. S obzirom na agilnost i brzinu posla danas, postoji vjerojatnost da će ključni ulagači i tehnička podrška odgovorna za uspjeh projekta napustiti tim ili kompaniju prije neko sam projekt bude funkcionalan. Ujedno postoji i mogućnost da u tako dugom vremenu samo poduzeće doživi ekonomske poteškoće ili druge probleme koji mogu prolongirati završetak samog projekta. Nadalje, današnji lokalni sustavi nisu opremljeni s alatima koji mogu rukovati s polustrukturiranim podatcima. To uključuje dodavanje, primjerice *NoSQL* platforme, koja dodaje novi sloj kompleksnosti i produbljuje kompleksnost same faze implementacije.

Istovremeno, skladište podataka u oblaku može biti spremno za rad u svega nekoliko tjedana ili par mjeseci. Najveća količina tog vremena bit će utrošena na ekstrakciju podataka iz drugih izvora podataka samog poduzeća, te stvaranje front-end analitičkog alata radi uvida u skladište podataka (Kraynak & Baum, 2017.).

7.2. Troškovi skladištenja i opreme

Lokalna skladišta podataka su skupa po pitanju hardvera, softvera i administracije. Hardver može uključivati cijenu servera, dodatnih sustava za skladištenje, prostor za podatkovni centar radi pohrane hardvera, brza mreža radi dohvaćanja podataka i struja, te drugi izvori koji će održavati sustav konstantno upaljenim. Osim profesionalnog IT osoblja koje se brine za održavanje sustava kao i nadogradnju, troškove mogu povećati korisnici ili dobavljači kojima je dan pristup skladištu podataka.

Skladišta u oblaku zamjenjuju kapitalne izdatke (engl. *CapEx*) i troškove lokalnih sustava s jednostavnim troškovima poslovanja (engl. *OpEx*) odnosno plaćanju s obzirom na količinu skladišta i usluga koje se iskoriste unutar određenog mjeseca. Ugrubo govoreći, godišnji trošak korištenja skladišta u oblaku je okvirno jedna desetina troška lokalnih sustava (Kraynak & Baum, 2017).

Ujedno ako poduzeće konstantno provodi upite radi obrade podataka, najbolje je pronaći rješenje koje sadrži jeftinije računalne troškove. Ako poduzeće ima puno podataka, ali te podatke uglavnom koristi samo, primjerice, jedan tim, tada je najbolje pronaći skladište koje sadrži jeftinije troškove pohrane (Segment).

7.3. Balansiranje i podešavanje

Radi optimalnih performansi, lokalno skladište podataka mora biti modelirano, balansirano i podešavano, što zahtjeva značajan ulog, te pridodaje ujedno i troškove nadgledanja i administracije. Takve konfiguracije često uključuju:

- određen broj brzih centralnih procesorskih jedinica (CPU),
- određenu količinu memorije,
- određen broj i veličinu diskova radi pohrane,
- input/output propusnost (mjera koja određuje koliko se podataka može prenijeti u određenom trenutku),

- podatkovni model koji definira strukturu skladišta, uključuje tipove podataka i frekvenciju ažuriranja.

Ulog u lokalne sustave, do mjere da cijelu godinu rade najbolje i najefikasnije što mogu, je pretjerano, pogotovo kad su za rad do te mjere potrebni samo omanji dio godine. Primjerice poduzeće možda treba maksimalnu snagu navedenog sustava tek u zadnjoj četvrtini godine, ali mora plaćati maksimalne performanse i uporabu 24 sata dnevno, svaki dan, cijelu godinu jer takvi sustavi ne mogu fluidno povećavati ili smanjivati upotrebu odnosno performanse (Kraynak & Baum, 2017). Elastični sustavi za skladištenje podataka u oblaku moraju uključivati kompleksnost sustava, troškove memorije i administraciju unutar troška pretplate, te ujedno i dinamičku izmjenu u razdobljima kada je poduzeću potrebna veća količina prostora ili bolje performanse.

Za skladišta podataka koja nisu samoodrživa, određeni zaposlenici će trebati provoditi određeno vrijeme optimizirajući, prilagođavajući veličinu i nadgledavajući klaster skladišta kako bi osigurali najbolju izvedbu skladišta (Segment).

7.4. Troškovi pripreme podataka i ETL-a

ETL prodavači prodaju aplikacije za određenu količinu novaca koja se plaća unaprijed, što ponekad može izazvati problem za same financije poduzeća. Nakon što kupe aplikaciju, korisnici moraju nabaviti dodatan hardver, instalirati aplikaciju i integrirati razne komponente prije nego je mogu početi koristiti - proces koji može trajati između 12 i 24 mjeseca. Povremena potreba za nadogradnjom aplikacije zahtjeva dodatne troškove.

Lokalno skladište podataka mora izvući sve podatke iz podatkovnih izvora. Nakon toga treba transformirati podatke u rigidne podatkovne strukture tražene od sustava, prije nego ih učitaju u skladište. Problem koji ujedno nastaje je nužno obavljanje navedenih procesa unutar ograničenog i skupog kapaciteta skladišta. To rezultira obavljanjem ETL procesa u noćnim satima ili generalno satima izvan radnog vremena kako bi izbjegli koliziju s drugim procesima obrade podataka. Ujedno polustrukturirani podatci ne dolaze

u konzistentnim redovima i stupcima inherentnim tradicionalnim strukturama podataka, te uključuju podatke velikog volumena i brzine što čini cijeli proces još skupljim.

Najbolja skladišta podataka u oblaku mogu učitati polustrukturirane podatke direktno bez transformacije, te omogućuju pristup svježim podatcima do 50 puta brže od tradicionalnih skladišta. Uz to manja cijena neograničenog skladišta pruža mogućnost pristupa svim podatcima i obavljanju svih procesa obrade podataka, bez potrebe da ih limitira na određena vremenska razdoblja u danu (Kraynak & Baum, 2017).

7.5. Troškovi specijaliziranih alata za poslovnu analizu

Lokalna skladišta podataka nisu dizajnirana na način da rukuju s volumenom, vrstom i brzinom današnjih podataka, rezultirajući time da poduzeća rade na dvije podatkovne platforme: jedna je lokalna - *SQL* skladište podataka za pohranu tradicionalnih relacijskih podataka i velika *NoSQL* platforma, koja se može pokretati na lokalnim sustavima ili u oblaku, radi pohrane ne-relacijskih podataka. Nažalost ti novi sustavi unose i nove kompleksnosti pri rukovanju i zahtijevaju specijalizirane alate koji nisu ni blizu zastupljeni poput *SQL* alata.

Idealno rješenje je oblak, koji donosi najbolje od oba svijeta – fleksibilnost integracije relacijskih i ne relacijskih podataka zajedno s podrškom za postojeće *SQL* alate i vještine potrebne za provođenje upita nad podatcima (Kraynak & Baum, 2017).

7.6. Skaliranje i elastičnost

Kompanije koje se brzo razvijaju i šire, skalabilnost infrastrukture mogu mjeriti prema troškovima, resursima i jednostavnosti korištenja (Leven, 2017). Skalabilnost se odnosi na sposobnost sustava da rukuje sa sve većom količinom posla, ili na potencijal da obavi veću količinu posla u jednakom vremenu kad se procesorska snaga proširuje kako bi se prilagodila rastu. Sustav je uglavnom skalabilan ako može povećati količinu posla i

propusnost kad se dodaju novi resursi. Skalabilnost omogućava unos bilo kakve količine podataka, bilo koji broj korisnika i bilo koju količinu zadataka i upita nad podatcima. Skalabilnost je ujedno direktno povezana s elastičnošću, mogućnosti da se dinamički izmjenjuje okolina (brzina obrade podataka i upita) s obzirom na potrebe poslovanja u danom trenutku; mogućnost mijenjanja količine korištenih resursa u trenutku izvršavanja određenoga upita. Elastičnost je stupanj do kojeg se sustav može prilagoditi poslovnim promjenama i opterećenjima pridodavanjem i uklanjanjem resursa na zahtjev, na način da u svakom trenutku u vremenu dostupni resursi odgovaraju potražnji ili zahtjevu što je točnije i bliže moguće.

Konvencionalna skladišta podataka doživljavaju česta usporavanja sustava i rušenja jer se korisnici i procesi natječu za limitiran izvor; obradu podataka. Navedeni sustavi usko vežu skladišta i računala u jedan računalni čvor (engl. *cluster*) poskupljujući proces nadogradnje jednog bez nadogradnje drugog sustava. Oblak zato omogućava praktički neograničeno skladište i računalne izvore, te je ujedno velika prednost što odvaja jedno od drugoga. Skalabilnost skladišta podataka u oblaku se dijeli na:

- **skladište**: Skladište je inherentno skalabilno i lako prilagodljivo promjenjivim potrebama poslovanja,
- **računalne izvore**: Resursi korišteni za procesiranje skupina podataka i upita bi trebali biti lako skalabilni, u svakom trenutku, kako se broj i intenzitet zadataka mijenja,
- **korisnici i zadatci**: Rješenja s fiksiranim računalnim resursima usporavaju kako se broj korisnika i zadataka povećava. Organizacije su često prisiljene replicirati podatke u zasebne podatkovne centre, premještati određene zadatke izvan uobičajenih radnih sati, i stavljati ljude u red, radi obavljanja posla, kako bi se sačuvale performanse. Samo oblak može pružiti mogućnost skaliranja dodavanjem „računalnog čvora“ raznih veličina za skoro neograničen broj korisnika ili zadataka koji pristupaju jednoj kopiji podataka, bez da utječu na performanse jedno drugog.

U slučajevima gdje je ekstremno skaliranje potrebno (više od 2 TB podataka) skladište podataka je jedino pravo rješenje jer neće uspostaviti nikakva ograničenja po pitanju nadolazećih podataka. Međutim bitno je svakako provjeriti kako određena skladišta skaliraju u vrijeme velike obrade i potražnje podataka. *Amazon Redshift* zahjteva ručno dodavanje računalnih čvorova primjerice, dok *Snowflake* nudi auto-skalirajuću funkciju koja dinamički mijenja veličinu klastera (Segment).

7.7. Smanjivanje kašnjenja i pada sustava

Velik broj kompanija s lokanim rješenjima imaju dva glavna problema; moraju čekati satima ili više od jednog cijelog dana kako bi podatci, prikupljeni dan ranije, bili dostupni u skladištu podataka. Moraju čekati slično vrijeme kako bi se izvršio kompleksan upit nad povećim skupom podataka. U nekim slučajevima više konkurentnih procesa se može zamrznuti ili srušiti sustav, povećavajući kašnjenje i prekid rada.

Uzevši u obzir virtualnu neograničenu količinu skladišta i računalnih resursa, skladišta podataka u oblaku su izgrađena na način da lakše povećavaju ili smanjuju količinu paralelno provedenih procesa s obzirom na potrebe poslovanja. Međutim, smanjivanje kašnjenja i eliminiranje neplaniranog pada sustava zahtjeva više od poboljšanja samog resursa sustava. Bolja rješenja pojednostavljuju protok podataka i pohranjuju podatke kako bi pospješili izvršavanje upita bez potrebe za ručnim podešavanjem (Kraynak & Baum, 2017).

7.8. Sigurnosni problemi i troškovi

Jedna provala ili probijanje u sustav mogu značiti noćnu moru što se tiče odnosa s javnošću i rezultirati gubljenjem posla i klijenata na duge staze. Iako oblak dosta često djeluje kao sigurnosni rizik, u nekim slučajevima može biti sigurniji od samog podatkovnog centra firme.

Koliko god se govorilo o većoj sigurnosti lokalnih sustava jer je cijeli podatkovni centar unutar same korporacije, u obzir se rijetko uzimaju održavanja sigurnosnog sustava koji dolazi s tim; oprezna i konstantna provjera vatrozida, sigurnosni protokoli, enkripcija podataka prilikom pohrane i prijenosa, postavljanje dopuštenja i restrikcija korisnicima, te nadzor i adaptacija mogućim sigurnosnim prijetnjama. Efektivni sustavi za zaštitu podataka su skupi i kompleksni za implementirati, pogotovo po pitanju ljudskih resursa.

Pošto skladišta podataka u oblaku pružaju usluge većem broju poduzeća, mogu si dopustiti ekspertize i resurse potrebne za zaštitu industrije, klijenata te sveobuhvatnu kraj-s-krajem zaštitu (engl. *end-to-end*) (Kraynak & Baum, 2017). Nakon uspostave povjerenja s partnerom, po pitanju skladišta u oblaku, smanjivanje sveukupne potrošnje i poboljšanje analize se bazira uglavnom na faktorima poput smanjivanja ovisnosti o internim IT resursima, pojednostavljeno dohvaćanje, učitavanje i integriranje podataka, proširivanje grupa unutar poduzeća koje koriste podatke, te mogućnost rada nakon pada sustava (Alibaba Cloud, 2019).

7.9. Zaštita podataka i oporavak podataka

Lokalna skladišta podataka su osjetljiva na gubljenje podataka zbog mogućeg pada sustava, oštećenja i prestanak rada opreme, gubljenja struje, krađe ili vandalizma i prirodnih katastrofa. Kako bi podatci bili osigurani, trebaju se stvarati sigurnosne kopije koje se pohranjuju na druge lokacije. Rezervno napajanje je ujedno bitno imati radi sprječavanja gubljenja podataka i osiguravanja dostupnosti skladišta podataka u svakom trenutku radi procesiranja nadolazećih podataka i upita. Ako stvarno određena šteta nastane, poduzeću će biti potrebna određena količina stručnjaka radi oporavka podataka. Ujedno ako je skladište podataka važno za bolnice, vojne ustanove ili bilo kakve druge javne sustave koji si ne mogu dopustiti prestanak rada neko vrijeme, bit će potreban geografski odvojen podatkovni centar zajedno sa softverom, licencama i procesima koji će osigurati automatski prijenos podataka i poslovanja bez zadržavanja procesa i usluga.

Oblak pruža idealno rješenje za zaštitu i oporavak podataka; po svojoj prirodi pohranjuje podatke izvan prostorija poslovanja (engl. *off premises*). Određena rješenja u oblaku automatski stvaraju sigurnosne kopije podataka na dvije ili više lokacija i rezervna napajanja koja mogu pružati uslugu čak tijekom duljeg nestanka struje. S obzirom da poslužuju velik broj korisnika, pružatelji usluga u oblaku mogu pružiti navedene usluge po puno manjim cijenama od zasebnih sigurnosnih sustava, raspoređujući same troškove na preko tisuću klijenata (Kraynak & Baum, 2017).

8. Istraživanje

Početnicima u polju analize podataka svakako može biti problem odlučiti koje sustave koristiti, na koji način ili gdje uopće krenuti. Iako navedeni rad ne obuhvaća cjelokupan proces prvih koraka u navedenome polju, svakako može kroz ovo istraživanje koristiti kao primjeren uvod za buduće podatkovne analitičare. Ujedno se fokusira na svega par sustava koje sam s obzirom na brojčanost korisnika, godišnja rangiranja, recenzije i pruženu dokumentaciju ocijenio kao najprimjenjenije i najsveobuhvatnije sustave za skladištenje podataka; rad koji može koristiti menadžerima i poslodavcima da stvore sliku o tržišnoj ponudi i potrebi vremena u kojem živimo.

U nadolazećim stranicama ću se prvotno fokusirati na svaki sustav zasebno, te prikazati njihove funkcionalnosti, izdvojene karakteristike i način korištenja. Parametri po kojima ću ih rangirati i ocijeniti će biti prezentirani u kasnijem tabličnom prikazu koji će zatim biti zaokružen sveobuhvatnim zaključkom o rezultatima istraživanja i potkrijepljen iskustvima drugih korisnika - poduzećima raznih veličina koja koriste navedene sustave.

8.1. Snowflake

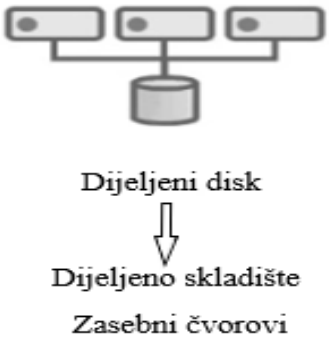
Snowflake je u srži *SQL* skladište podataka u oblaku. Sastoji se od tablica, redova, kolona, a komunikacija se odvija pomoću *SQLa* što s obzirom na dugoročnost i raširenost korištenja *SQLa*, rezultira vrlo ugodnom i prirodnom interakcijom sa sustavom za većinu korisnika. *Snowflake* pruža PaaS (*Platform-as-a-Service*), SaaS (*Software-as-a-Service*) i DWaaS (*Data Warehouse as a Service*) rješenja izgrađena za oblak, u koja ulaze dijeljenja podataka, podatkovna jezera (engl. *data lakes*), dupliciranje podataka, te za potrebe poslovanja prilagođene parametre i usluge. Ujedno se nalazi u partnerstvu s najvećim pružateljima IaaS (*Infrastructure-as-a-Service*) usluga (*Amazon Web Services, Microsoft Azure, Google Cloud Platform*).

Jednostavnost korištenja, skalabilnost, elastičnost, raznolikost i arhitektura su naglašena svojstva *Snowflake*-a uz cijenu korištenja usluga, odnosno plaćanja s obzirom na količinu utrošenog prostora i resursa. Velik broj skladišta podataka je i dalje kompleksan, što rezultira trošenjem vremena na pripremu podataka u određen format ili strukturu, osiguravanje da su podatci pravilno pohranjeni i niz drugih administrativnih i rukovodilačkih stvari. *Snowflake* olakšava taj korak, na način da se oni bave danom arhitekturom sustava omogućavajući korištenje sustava od prvog trenutka kad se korisnik poveže bez potrebe za dodatnim podešavanjem ili prilagođavanjem. Ujedno *Snowflake* podržava skalabilnost kao što je ona opisana u poglavlju 7.6.

Raznolikost poslovanja ujedno sadrži i raznolikost podataka, točnije, polustrukutrirani podatci, podatci iz drugih izvora poput socijalnih mreža mogu biti drugačije prirode od onih koje sadrži već uspostavljeno skladište, a zahtijevati unos u isto. *Snowflake* omogućuje pohranu svih tipova podataka na jednu lokaciju, za razliku od pohrane na više njih, olakšavajući analizu podataka jer se isti upiti ne trebaju izvršavati kroz više sustava povećavajući kompleksnost i sporost analitičkih procesa (*Snowflake*).

8.1.1. Arhitektura

Modeli *arhitekture dijeljenog diska* (slika 14) su platforme poput *SQL* servera i *Oracle* servera što omogućava skaliranje i računanje čvorova. No s obzirom da za sve postoji jedan „kanal“ koji dohvaća podatke i izvršava upite nad bazom, s vremenom taj kanal postaje pretrpan zahtjevima korisnika.



Slika 14: Arhitektura dijeljenog diska[18]

Ne podijeljena arhitektura pruža veću fleksibilnost jer se tražene operacije izvršavaju preko više čvorova, no ovdje postoje dva ključna izazova. Prvi problem je taj što su skladište i računala međusobno povezani, što znači da se ne može u danom sustavu skalirati skladište bez skaliranja računalnog sustava iako poduzeće primjerice nema potrebu za većim računalnim skaliranjem, dok istovremenom ima veliku količinu podataka koju treba obraditi. Ujedno se vidi nedovoljna količina fleksibilnosti; prilikom dodavanja još tri čvora prikazanom sustavu na slici 15, dodani čvorovi ne bi mogli obavljati nikakve zadatke dok se sami podatci ne bi drugačije raspodijelili po čvorovima, točnije unijeli u novo dodane čvorove.

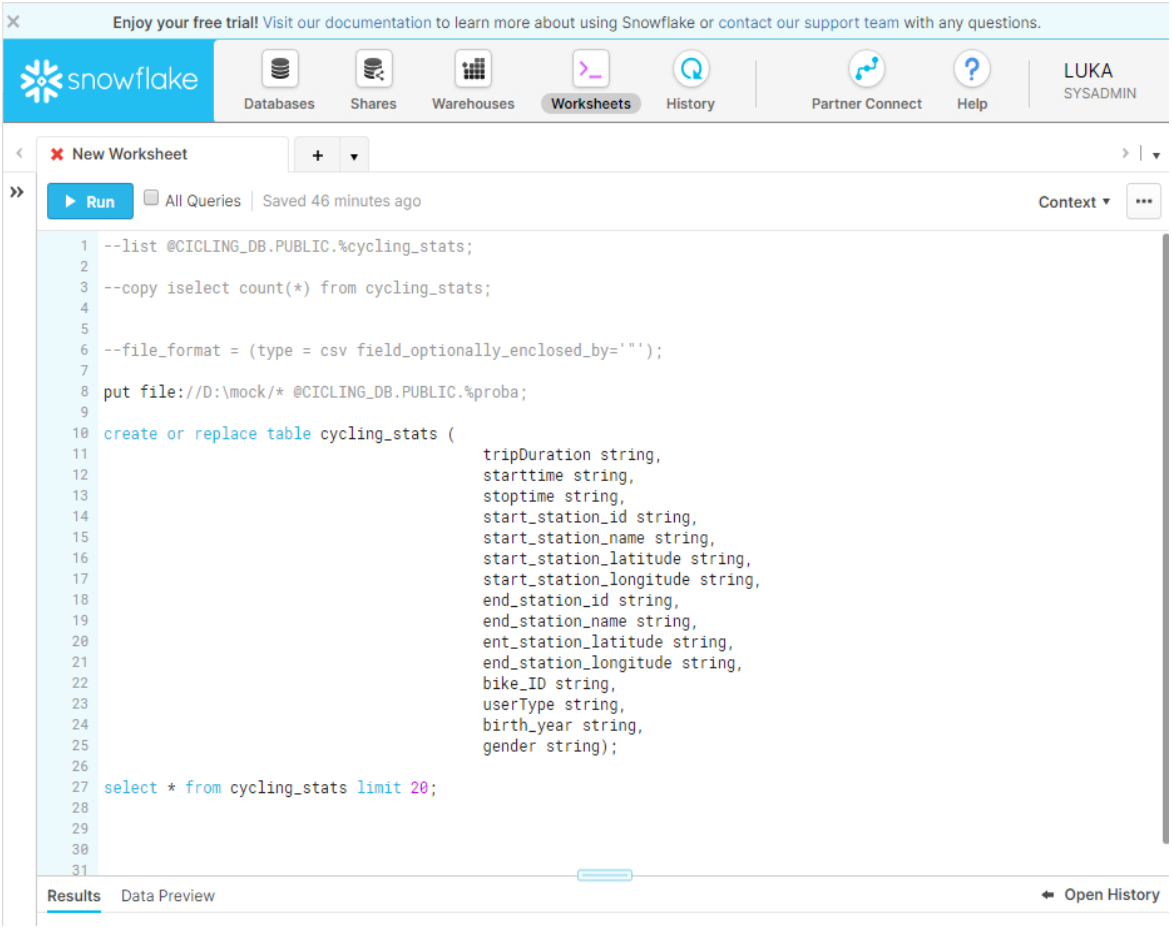
Snowflake pokušava spojiti najbolje od oba svijeta. Ima samo jednu kopiju podataka što znači da se korisnici ne trebaju baviti replikacijama ili „shadow" kopijama, a uz to postoji više zasebnih skalabilnih računalnih klastera ili virtualnih skladišta podataka. Pošto je takav sustav izgrađen u oblaku, ne treba brinuti oko problema i izazova vezanim uz infrastrukturu kao u tradicionalnim arhitekturama (Bock Corp, 2018).

[18] Prilagođeno prema izvoru (Attunitt. *Analist and investor day*). Preuzeto s: https://www.sec.gov/Archives/edgar/data/893821/000117891318002529/exhibit_99-1.htm

Slika 15: Ne podijeljena arhitektura[19]

Snowflake sučelje (slika 16) je lako koristivo s obzirom na intuitivan i interaktivan prikaz, pogotovo za korisnike koji su se već koristili sličnim sustavima. Sastoji se od „baza podataka“ (engl. *database*), komponente preko koje učitavamo postojeće baze, te kroz omanje sučelje ujedno dobivamo mogućnost stvaranja nove baze, kopiranje postojeće i brisanja baze (engl. *drop database*). Dijeljenje podataka je u ovom slučaju onemogućeno. Jedna od bitnih stavki je mogućnost dijeljenja baza i podataka sa željenim zaposlenicima unutar poduzeća nakon što administrator to dozvoli.

[19] Prilagođeno prema izvoru (Attunitt. *Analist and investor day*). Preuzeto s: https://www.sec.gov/Archives/edgar/data/893821/000117891318002529/exhibit_99-1.htm



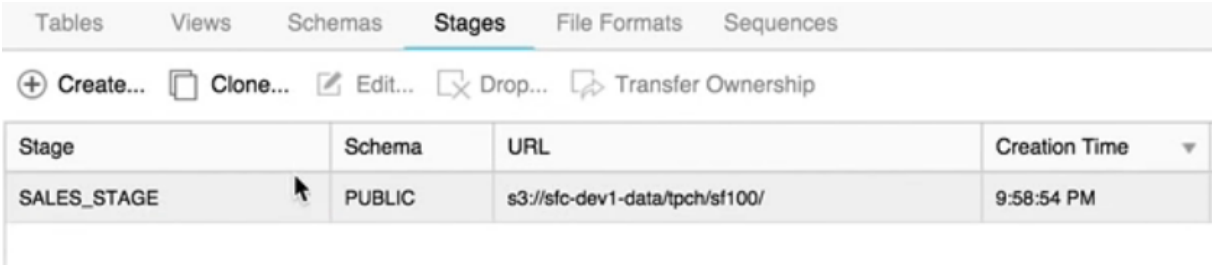
Slika 16: Snowflake sučelje

Opcija skladište (engl. *warehouse*) omogućuje stvaranje novog skladišta podataka, te uređivanjem danog skladišta grafički prikazuje količinu paralelno izvršavanih upita u određenome vremenskome periodu. Ta opcija je korisna prilikom utvrđivanja količine resursa koji se koriste dok više korisnika koristi dano skladište podataka radi optimizacije i prilagodbe sustava.

Radni list (engl. *worksheet*) je sučelje koje se uz CMD najviše koristi. Služi za unos i obradu upita koji se unose radi obrade željenih podataka, dok samim izgledom liči na uobičajeno *SQL server managment studio* sučelje. U podnožju radnog lista se nalazi prikaz izvršenih upita zajedno s vremenom trajanja, te mogućim problemima koji su se pojavili prilikom obrade upita kao što ćemo kasnije vidjeti.

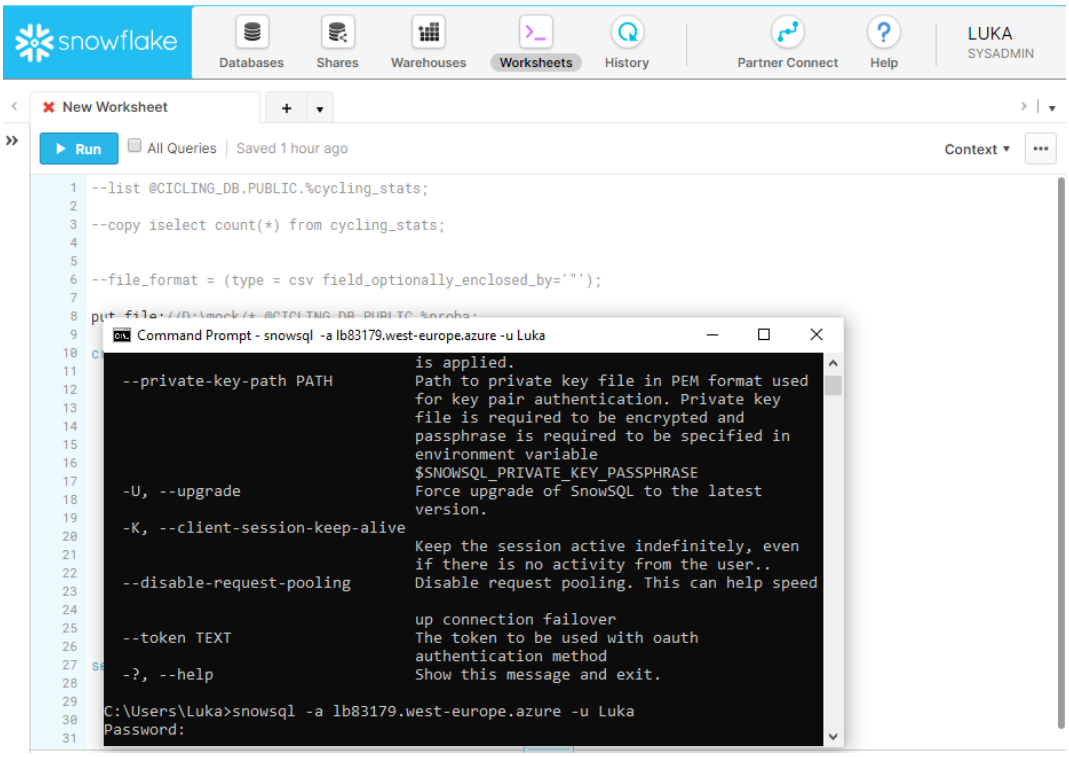
8.1.2. Postavljanje i učitavanje podataka

Snowflake omogućuje učitavanje podataka s *Microsoft Azura*, *Amazon Web Servisa* (AWS), *Google Cloud* platforme i lokalnog računala. U prva tri slučaja podatci se preuzimaju definiranjem putanje na osobni ili poslovni profil, unose se u okolinu za izvođenje podataka gdje se naknadno s njima rukuje, odnosno svrstava u željene baze ili tablice. Na slici 17 se može vidjeti primjer određenih podataka o prodaji unesenih u okolinu za izvođenje (engl. *staging*) koristeći s3 kantu (engl. s3 *bucket*) – opciju pohrane unutar *Amazon web* servisa. Koristeći opciju za formatiranje dokumenata (engl. *file format*) možemo definirati tip podataka koji se nalazi u novonastaloj okolini za izvođenje radi lakše obrade željenih podataka. S3 je *Amazonov* web servis u koji se pohranjuje neograničena količina podataka te se po potrebi dohvaćaju. Generalno služi kao nadopuna bazama podataka ili skladištima koji pohranjuju reference na s3 kantu koja sadrži detaljnije informacije o željenom entitetu.



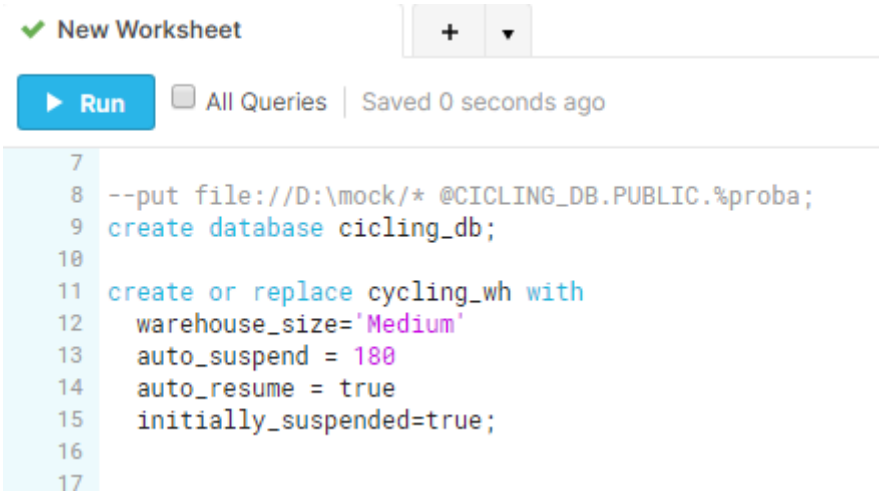
Slika 17: Okolina za izvođenje s putanjom odakle dolaze podatci

Prvi korak prilikom učitavanja podataka se sastoji od prijave na stvoreni račun putem *Snowsqla* kojeg aktiviramo kroz CMD sučelje (slika 18). *SnowSql* se direktno povezuje s prikazanim *Snowflake* sustavom i sučeljem omogućavajući nam korištenje ili radnog sučelja radi unosa i obrade upita ili klasičnog CMD pristupa. Prije samog učitavanja podataka stvaramo bazu podataka koristeći „create database“ funkciju nakon čega unosimo ime željene baze.

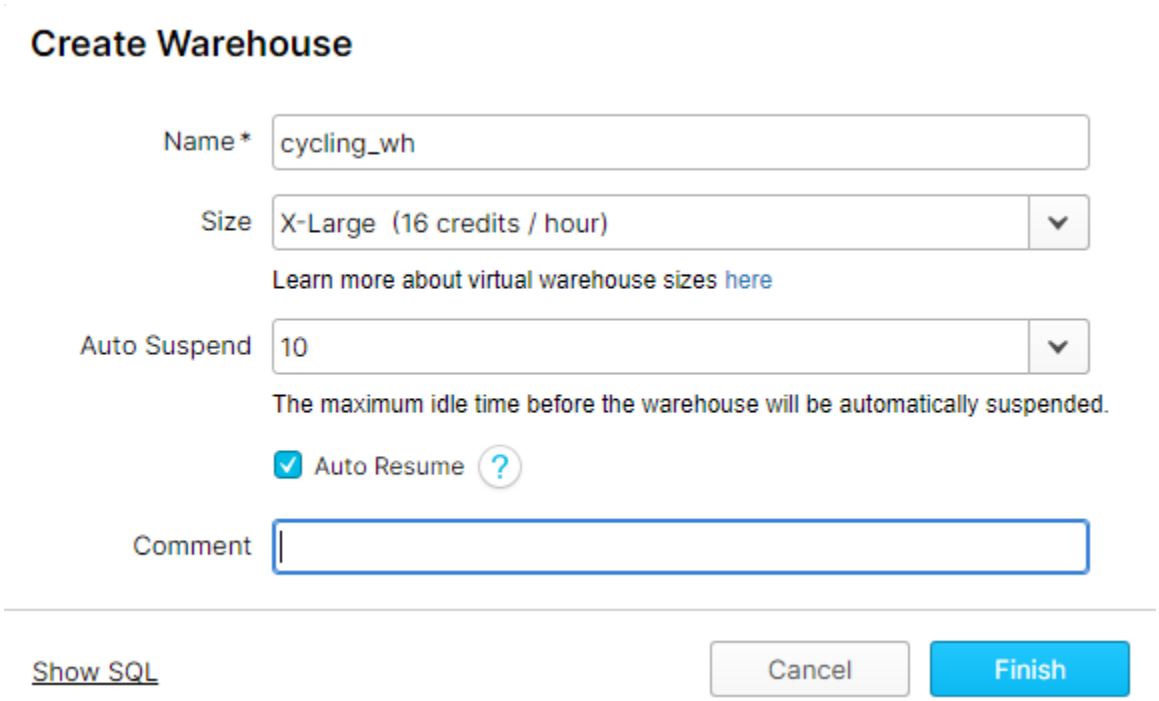


Slika 18: Snowsql autentifikacija

Potom stvaramo skladište podataka koristeći isti princip i na kraju samu tablicu koristeći pritom cijelo vrijeme *SQL*. Karakteristike skladišta podataka možemo definirati kroz CMD, radni list ili koristeći samo sučelje *Snowflakea* (slika 19 i 20).

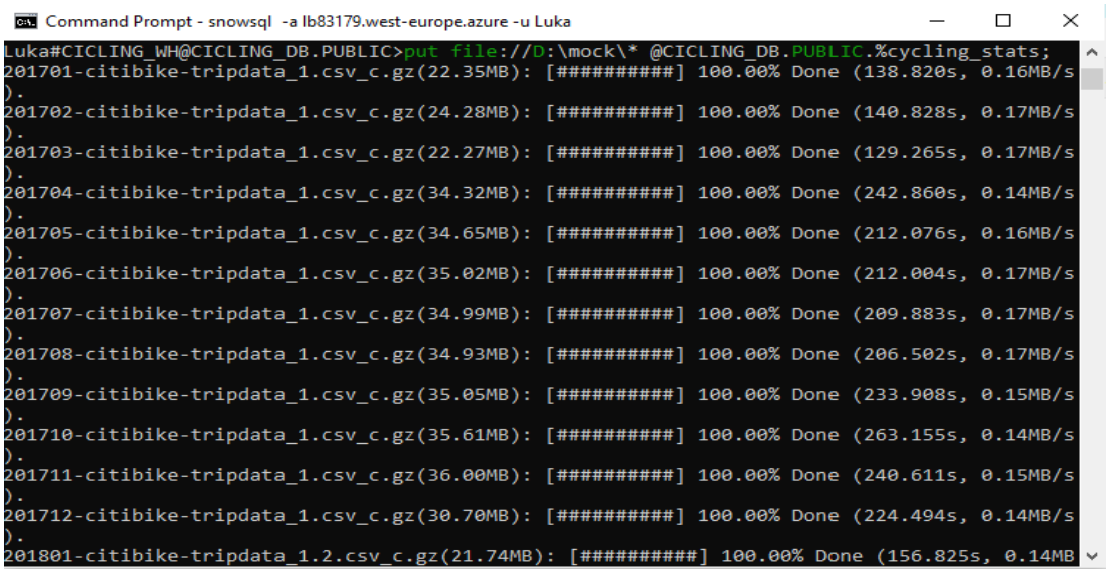


Slika 19: stvaranje skladišta podataka kroz radni list



Slika 20: Stvaranje skladišta podataka kroz sučelje

Prije samoga kraja stvaramo tablicu u koju ćemo svrstati učitane podatke definirajući pritom nazive polja i tipove podataka koja primaju. Unos podataka se izvršava kroz *„put“* naredbu striktno u CMD-u koja se sastoji od putanje koja vodi do podataka, odabrane baze podataka i odabrane tablice (slika 21). Nakon što se podatci kopiraju u željenu tablicu iz cjelokupne okoline za izvođenje, spremni su za obradu koristeći *SQL* u radnom listu ili u CMD-u.



Slika 21: Učitavanje podataka u bazu podataka

8.1.3. Naglašene karakteristike

Bitne naglašene karakteristike su „nul-kopija“ (engl. *zero-copy*), povrat (engl. *undrop*) tablice/baze/sheme, mogućnost izmjene broja čvorova koji procesiraju upite prilikom samog procesiranja i obrada polustrukturiranih podataka:

- nul-kopija je sposobnost kopiranja baze, sheme ili tablice. Preslika podataka prezentiranih u izvornom objektu se uzima prilikom stvaranja klona i daje na korištenje novonastalome objektu, te se klonirani objekt može izmjenjivati bez da promjene utječu na izvorni objekt. Novi objekt ne troši financijske resurse firme jer se koristi kao običan klon postojećoj bazi ne zauzimajući dodatan prostor pritom omogućujući ponovno stapanje s originalnom bazom nakon što su željene promjene unesene i testirane,
- povrat, primjerice, baze je radnja koja omogućuje dohvaćanje obrisane baze podataka u slučaju da programer umjesto, primjerice testne baze obriše originalnu. Isto pravilo vrijedi za tablice i sheme,
- prilikom stvaranja skladišta podataka definiramo broj servera (računalnih čvorova) koji će obrađivati nadolazeće upite. Ako korisnici zaključe da im je trenutačna

obrada upita prespora bilo zbog količine upita, broja ljudi koji izvršavaju upit na isto skladište ili jednostavno žele veću brzinu dohvaćanja podataka, *Snowflake* omogućuje izmjenu postavljenog broja čvorova prilikom obrade upita. Ujedno se vremensko izvršavanje upita pod određenim skladištem i brojem čvorova može pratiti u sučelju pod nazivom „History“. Kada sustav prepozna da nema više upita niti unosa podataka, on se sam gasi sve dok ponovno ne dođe zahtjev za novom obradom ili pohranom čime štedi novac poduzeću,

- obrada polustrukturiranih podataka (poput *JSONa*) i njihova mogućnost pohranjivanja u tablice s definiranom strukturom je jedna od naglašenijih karakteristika *Snowflakea*. Jednostavnim odabirom tipa ili formata podatka koji unosimo, podatci se šalju u okolinu za izvođenje odakle odlaze u željene tablice.

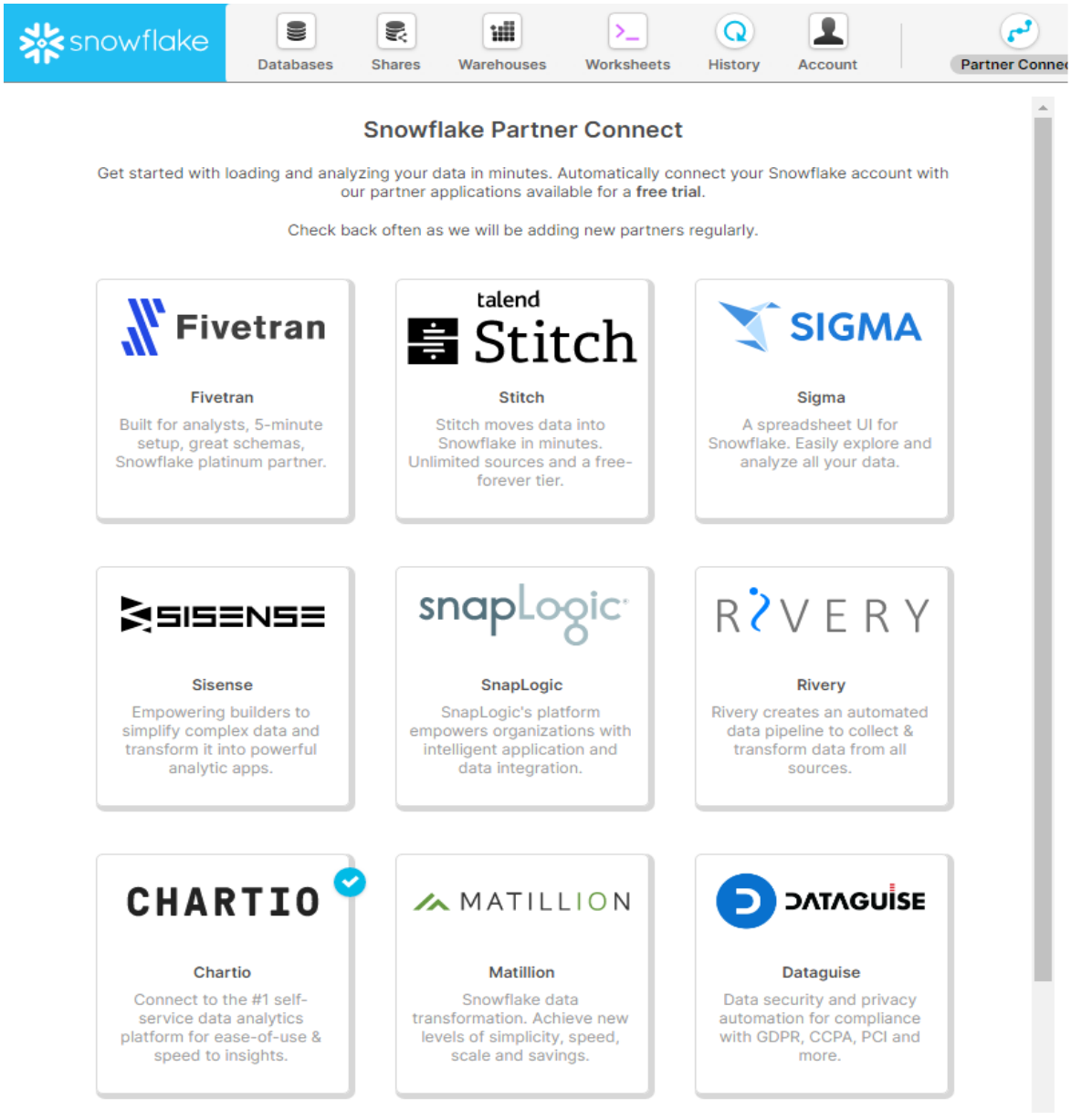
Dodatne funkcionalnosti koje pridonose samome sustavu, a nisu do te mjere vidljive na sučelju samog sustava su :

- sigurnost:
    - automatska enkripcija podataka (koristeći AES 256 enkripciju). Svi podatci koji su pohranjeni u okolinu za izvođenje (engl. *staging*) su automatski enkriptirani (od strane AES 128 ili 256 enkripcije). Ujedno se periodički izmjenjuju ključevi kriptiranih podataka radi veće sigurnosti sustava,
    - višefaktorska provjera autentičnosti radi pristupa korisničkom računu odnosno profilu,
    - zaštita objekata u samome sustavu (korisnika, skladišta podataka, baza podataka, tablica itd.) kroz hibridni model diskrecijske pristupne kontrole i kontrole bazirane na ulozi. Diskrecijska kontrola ograničava ulaz objektima s obzirom na identitet osobe na razini, primjerice, nekog dokumenta (npr. Lista korisničkih imena koji mogu pristupiti dokumentu)… Kontrola s obzirom na ulogu se bazira na definiranju liste rola i dodavanjem svakog korisnika u jednu ili više rola s time da svaka ima određenu razinu prava (kao primjerice „korisnik“, „administrator“, „šef“ itd.).

- putovanje kroz vrijeme (engl. *time travel*) – omogućuje pristup povijesnim podatcima koji su izmijenjeni ili izbrisani u određenome trenutku definiranog perioda. U standardnom paketu, izbrisani ili izmijenjeni podatci se mogu dohvatiti natrag unutar 24 sata, dok se u Enterprise verziji trajne baze podataka mogu zadržavati, odnosno dohvatiti, između 0 i 90 dana. Koristeći tu opciju korisnik može:
    - vratiti objekte poput tablica, shema, baza podataka koje s slučajno ili namjerno obrisane,
    - duplicirati ili stvoriti sigurnosnu kopiju,
    - analizirati korištenje i manipulaciju podataka kroz određeno vremensko razdoblje.
- siguran pad (engl. *fail safe)* – osigurava da su povijesni podatci osigurani u slučaju pada sustava ili drugog nekakvog katastrofalnog događaja. *Snowflake* ne sadrži opciju sigurnosnog kopiranja kao takvog, gdje se podatci periodički u potpunosti dupliciraju što dugoročno može dvostruko ili čak trostruko povećati količinu pohranjenih podataka u skladištu. Samo vrijeme za povratak podataka, vremensko trajanje pada cijelog sustava i gubitak podataka s obzirom na zadnju kopiju mogu prouzročiti velike troškove. Zbog toga navedena opcija omogućuje sedmodnevni period prilikom kojeg se podatci mogu vratiti. To razdoblje počinje od trenutka kada zadržavanje podataka u sklopu „putovanja kroz vrijeme“ prestaje,
- mikro particioniranje (engl. *micro-partitioning*) - svi podatci u *Snowflake* tablicama su automatski podijeljeni u mikro-particije (jedinice pohrane). Svaka particija koristi između 50 i 500 MB nekompresiranih podataka, gdje su grupe redova u tablicama, mapirane u individualne mikro-particije, organizirane u kolumne. Takva struktura i veličina omogućuju postupno dijeljenje većih tablica te njihovo kompresiranje u milijune ili u stotine milijuna mikro particija čime se postiže brža obrada upita.

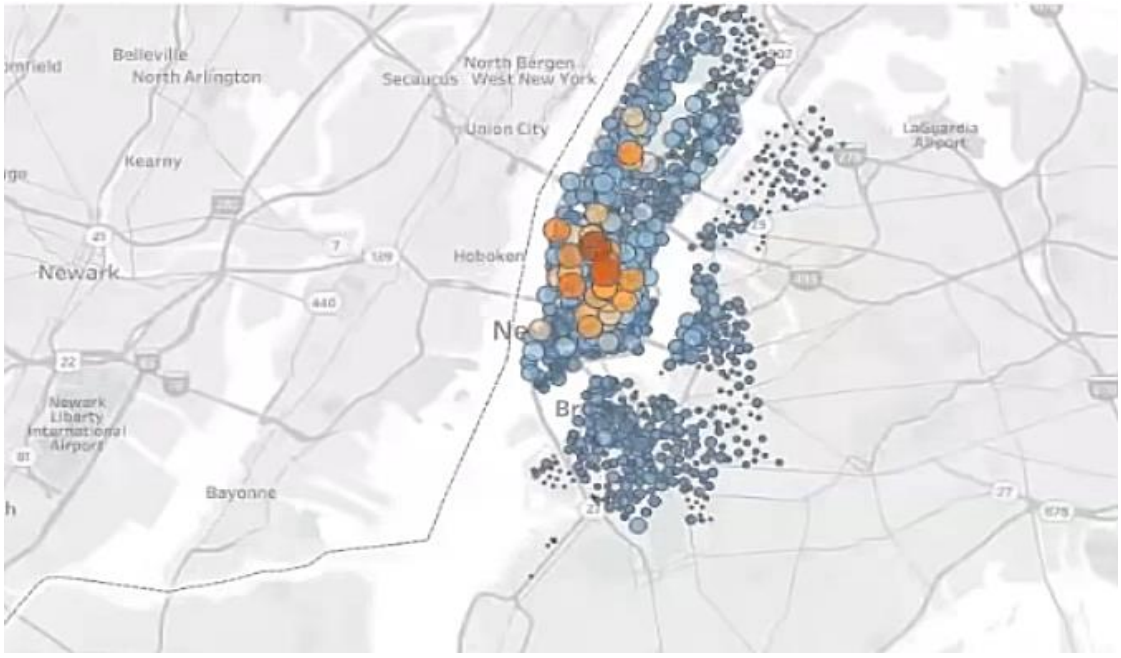
8.1.4. Primjer korištenja

Učitani podatci iz baze, odnosno skladišta mogu biti uobičajeno izlistani s obzirom na upit kao u svakom sustavu baziranom na *SQL*. *Snowflake* nudi osim besplatnog probnog razdoblja i besplatno probno razdoblje u suradnji s drugim partnerskim organizacijama poput *Charito* ili *Tableau* (slika 22).



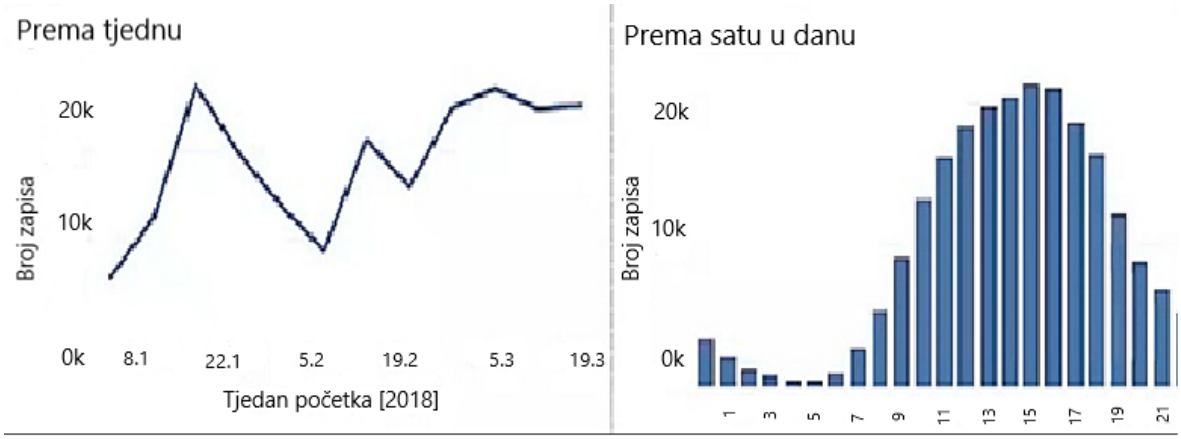
Slika 22: Partnerske organizacije

Krajnja svrha i cilj skladišta podataka je generiranje istih podataka kroz izvještaje, grafičke i slikovne prezentacije ili čak interaktivne prikaze temeljene na *SQLu*. *Tableau* je primjer softvera korištenog za vizualni prikaz filtriranih podataka, te s obzirom na primjerice biciklističke podatke sa stranice *CitiBike* koja vodi evidenciju o broju biciklista koji prođu na dnevnoj bazi u New Yorku, možemo izdvojiti određeni tip podataka u suradnji sa *Snowflakeom* radi lakšeg prikaza željenih informacija (slika 23 i 24).



Slika 23: najčešća mjesta za ostavljanje bicikla u New Yorku[20]

[20] Prilagođeno prema izvoru (https://www.youtube.com/watch?v=e3acougGe2A&list=PLC0beSIViTVYywdE5ugoRHy8BIfR_SF8t&index=2&t=2231s)



***Slika 24: Grafički prikaz količine vozača s obzirom na tjedan i sat u odabranome danu*[21]**

Interaktivno sučelje, lakoća korištenja, broj opcija koje se nude na osnovnoj razini i ponude poput spajanja s drugim aplikacijama radi obrade podataka vizualnim putem, s razlogom čine *Snowflake* jednim od korištenijih skladišta za pohranu podataka u oblaku.

8.2. Amazon Redshift

Amazon Redshift je potpuno održavano skladište podataka u oblaku koja se mjeri u petabajtima, a stvoreno je za pohranjivanje i analiziranje velikih skupina podataka. *Redshift* se, poput *Snowflakea*, *Google BigQueryja* i drugih modernih skladišta podataka u oblaku bazira na C-dućanu (engl. *C-store*), spajajući se na *SQL* bazirani klijent i alate namijenjene poslovnoj inteligenciji (engl. *business intelligence*) kako bi učinio podatke dostupnima u pravom vremenu (Amazon Web Services).

Amazon Redshift se bazira na *PostgreSQLu*[22] što znači da većina postojećih *SQL* aplikacija radi s minimalnim promjenama. Ujedno je strukturiran kao klaster, odnosno niz čvorova gdje je jedan od čvorova glavni čvor; omogućujući na taj način komunikaciju s klijentskom aplikacijom i distribuciju kompiliranog koda drugim čvorovima radi paralelnog

[21] Prilagođeno prema izvoru (https://www.youtube.com/watch?v=e3acougGe2A&list=PLC0beSIViTVYywdE5ugoRHy8BIfR_SF8t&index=2&t=2231s)
[22] PostgreSQL je sustav za upravljanje relacijskim bazama podataka baziran na *SQL-u.*

procesiranja. Jednom kada čvorovi vrate raspodijeljene podatke (po poljima), glavno čvorište kombinira rezultate i izvrši nad njima agregaciju. S obzirom na potrebe poslovanja i skladištenja podataka, korisnici mogu početi s malim klasterom, povećavajući njihov broj s vremenom s obzirom na kompleksnost i količinu upita, količine podataka i druge parametre. Odluči li korisnik koristiti klaster godinu dana ili dulje, ujedno može uštediti rezervirajući računalne čvorove na razdoblje između jedne do tri godine. Rezerviranjem računalnih čvorova se može uštediti poprilična količina novca, u usporedbi s plaćanjem po satu odnosno po zahtjevu (engl. *on demand*) kao što ćemo vidjeti na kasnijim stranicama istraživanja.

Klasteri imaju mogućnost povratka u određeni trenutak u vremenu koristeći zabilježeni snimak (engl. *snapshot*). Postoje dva snimka: automatski i manualni. *Amazon Redshift* pohranjuje te snimke unutar *Amazon Simple Storage Servicea (Amazon S3*) koristeći kriptiranu *SSL* (engl. *Secure Socket Layer*) vezu. Ako korisnik ima potrebu vraćanja klastera, *Amazon Redshift* stvara novi klaster i unosi podatke i snimke koji su specificirani.

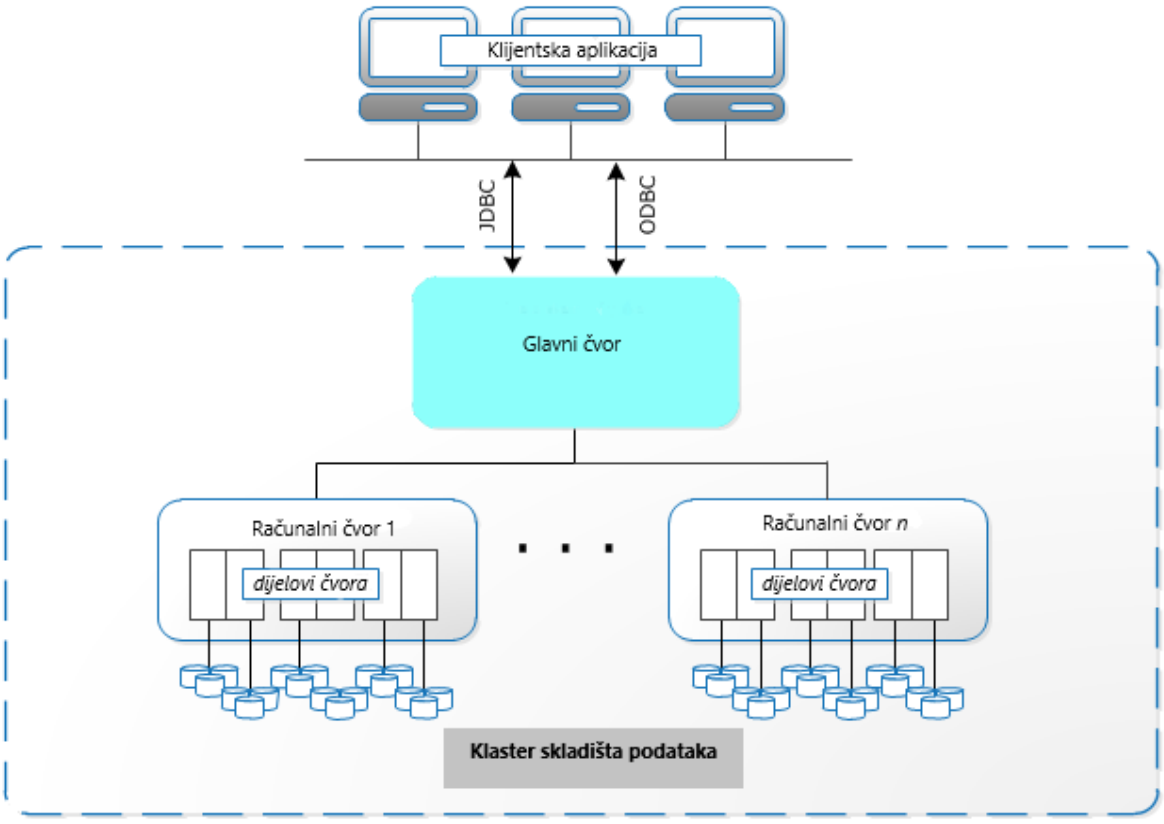
8.2.1. Arhitektura

Amazon Redshift je integriran s nizom alata za učitavanje, ETL alatima i alatima vezanim uz poslovnu inteligenciju, analizu i generiranje izvještaja. Njegova komunikacija s klijentskim aplikacijama se bazira na *JDBC* i *ODBC* upravljačkim programima. *JDBC* (*Java Database Connectivity*) je sučelje za programiranje za programski jezik Java, koji definira kako klijent može pristupiti podatcima, pogotovo u relacijskim bazama podataka. Služi kao posrednička aplikacija između aplikacije izgrađenoj u Javi i baze podataka, odnosno skladišta podataka. *ODBC* (*Open Database Connectivity*) koristi sučelje koje je stvorio Microsoft a koje dopušta aplikacijama da se isto tako spoje na sustav za upravljanje bazom podataka koristeći *SQL* kao standard za dohvaćanje podataka.

Komunicirajući preko sučelja, klijent šalje željene upite na već spomenuti glavni čvor klastera. Glavni čvor distribuira *SQL* zahtjeve računalnim čvorovima samo u slučaju kada

se upit referencira na jednu od tablica koje su pohranjene na jednom od čvorova. Svi upiti se izričito izvode na glavnom računalnom čvorištu naglašavajući kako je *Redshift* ujedno dizajniran na način da određene *SQL* funkcionalnosti izvodi samo na glavnom čvoru. Upit koji izvršava bilo koju od tih funkcionalnosti (kao što je primjerice *current schema*) će vratiti grešku ako referencira tablice koje se nalaze van glavnog računalnog čvora.

Svaki računalni čvor ima svoj zaseban CPU, memoriju i skladište na disku koji su određeni s obzirom na čvor. Kako količina posla i podataka raste, može se postepeno povećati računalni kapacitet i kapacitet skladišta određenog klastera povećavajući broj čvorova, nadograđujući tip čvora itd. (Amazon Web Services).



Slika 25: Arhitektura Amazon Redshifta[23]

[23] Prilagođeno prema izvoru (https://docs.aws.amazon.com/redshift/latest/dg/c_high_level_system_architecture.html)

Kao što vidimo na slici 25, računalni čvor je podijeljen na dijelove. Svakom dijelu je dodijeljen određeni dio memorije čvora i prostora na disku, gdje odrađuje dio posla. Glavno čvorište distribuira podatke svim dijelovima i šalje dio svakog upita, ili neke druge operacije u bazi, u određeni dio određenog čvora nakon čega dijelovi čvora rade paralelno radi obrade upita.

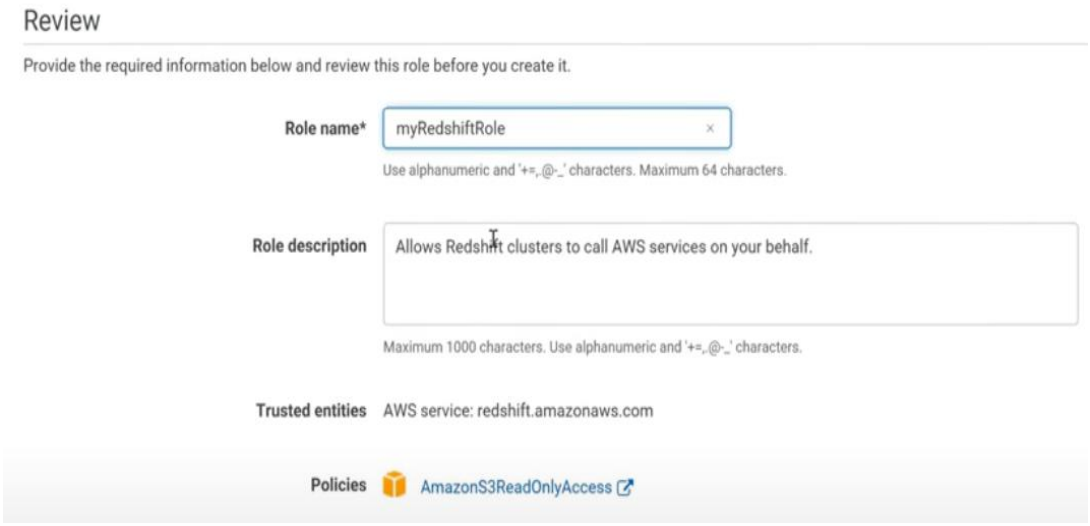
Svaki klaster se sastoji od jedne ili više baza podataka i iako je *Amazon Redshift* sustav za upravljanje relacijskim bazama podataka koji sadrži funkcionalnosti tipične za takav sustav, poput OLTPa ili dodavanja i brisanja podataka, ujedno je optimiziran za rad s velikim količinama podataka, brzu analizu i stvaranje izvještaja nad velikim skupinama podataka.

8.2.2. Postavljanje i učitavanje podataka

Za razliku od *Snowflakea* koji je zahtijevao samo registraciju, nakon čega nam je postavio i razinu autorizacije, u *Redshiftu* je proces stvaranja prvog klastera radi unosa podataka nešto dulji i donekle kompleksniji. Pristupanje samoj službenoj stranici može na prvu biti zastrašujuće kad korisnik shvati koliko funkcionalnosti i alata *Amazon* nudi, te koliko će vremena trebati da se snađe u samom sustavu koji ima namjere koristiti. No, priložena dokumentacija i objašnjavanje procedura korak po korak uvelike ublažavaju početni dojam.

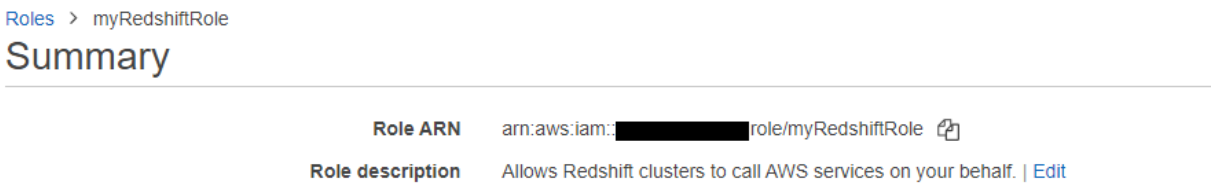
Amazon Redshift koristi port 5439 prilikom pokretanja. U slučaju da taj port nije autoriziran od strane vatrozida, trebat će se ručno odobriti na računalu ili promijeniti u neki drugi postojeći kako bi se mogla nastaviti koristiti usluga. Prvi korak nakon registracije se svodi na otvaranje *IAM* konzole, biranje „Role“ u navigacijskom meniju i kreiranja nove uloge. Korisnički klaster ima potrebu za autorizacijom radi pristupa vanjskom katalogu podataka u *AWS Glue* ili *Amazon Anthena* sustavima i podatcima u *Amazon* s3 kantama te se iz tog razloga definiraju *IAM* (Access Managment) uloge. Potom se bira servis koji će koristiti ulogu koja se kreira, u ovom slučaju je to *Redshift*, nakon čijeg biranja dolazimo

do opcije filtriranja ograničenja. Pretražujemo i biramo „*AmazonS3ReadOnlyAccess*“; s vremenom se mogu pridodavati i druge uloge s obzirom na stvorenu ulogu i potrebe poslovanja kao što su: *AmazonRedshiftFullAccess*, *AmazonDynamoDBFullAccess* itd., no prilikom prvog upoznavanja sa sustavom ili prilikom prvog postavljanja biramo samo navedenu. Uloge će se kasnije po potrebi moći dodjeljivati ili oduzimati.



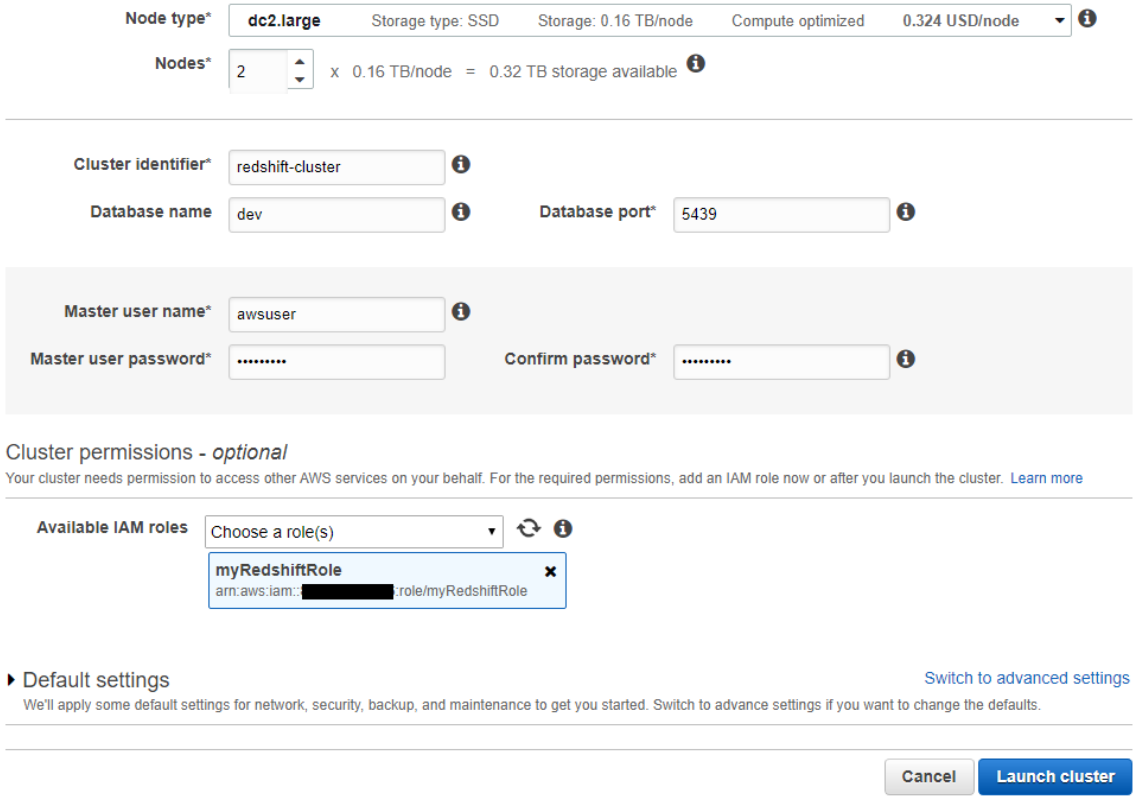
Slika 26: Redshift kreiranje uloge

Otvaranje nove uloge i kopiranje njenog „Role ARNa“ je sljedeći korak (slika 26 i 27). Preporuča se spremiti *ARN* na računalo lokalno radi kasnijeg korištenja kao što ćemo vidjeti. *ARN* se odnosi na *Amazon Resource Name*, koji je, kako i samo ime kaže, jedinstveno ime korišteno za identifikaciju određenih *AWS* resursa. Koristi se u slučajevima kada je potrebno specificirati resurs kroz više *AWS* usluga, poput *IAM* politika, *Amazon* servisa za relacijske baze podataka ili pozive na API (engl. *application programming interface*).



Slika 27: ARN role

Prije same aktivacije klastera pod naslovom „brzo pokretanje klastera“ (engl. *quick launch cluster*) pronalaskom *Amazon Redshifta* pod opcijom „Usluge“, biramo jednu od regija u gornjem desnom kutu koje su nam lokacijski najbliže radi stvaranja klastera kako bi dugoročno gledano smanjili latentnost i troškove. Nakon toga pristupamo navedenoj usluzi i stvaramo klaster kao što je prikazano na slici 28.



Slika 28: Stvaranje klastera

Redshift nudi četiri tipa čvorova s obzirom na trenutačne potrebe. Čvorovi se mogu mijenjati po potrebi tako da se u ovom koraku njihov broj ne definira striktno. U paketu besplatnog testiranja, u trajanju od 2 mjeseca i 750 sati sveukupno, dc2.large čvor je jedini koji spada pod kategoriju besplatnog koštajući izvan navedene besplatne okoline 0.324 dolara po čvoru, te omogućujući 0.16 terabajta pohrane po čvoru. Što više čvorova biramo to je veći kapacitet skladišta, a s time i cijena klastera. Odabravši željenu veličinu definiramo ime baze i klastera, port na kojem će se pokretati baza podataka, korisničko ime koje služi za pristup željenoj bazi i skladištu, lozinka, te rolu koju smo kreirali prilikom

stvaranja klastera. Nakon kreiranja će proći određeno vrijeme dok se klaster ne stvori što naglašava opcija klaster status (engl. *cluster status*) nakon čega dobivamo podatke o novom klasteru kliknuvši na njegovo naziv.[24]

Postavljanje sigurnosnih grupa za VPC (*Virtual Private Cloud*) služi stvaranju jedne vrste vatrozida radi kontrole nadolazećeg i odlazećeg prometa. VPC omogućuje pokretanje *Amazon* resursa u virtualnoj mreži koju smo definirali, što poprilično sliči tradicionalnoj mreži kojom bi poduzeće operiralo u vlastitom podatkovnom centru, s vrlinom da pritom koristi skalabilnu infrastrukturu *AWSa*. Stvorivši novo *TCP* pravilo, definirajući TCP[25] protokol, odredivši port (u ovom slučaju 5439) i postavljanje destinacije ili izvora za nadolazeće podatke 0.0.0.0/0. Uzevši u obzir da je ovo testni klaster koristimo navedeni izvor (engl. *source*) iz razloga što dopušta pristup s bilo kojeg računala na internetu. U pravoj okolini kreiraju se pravila s obzirom na postavke mreže poduzeća.

Inbound rules

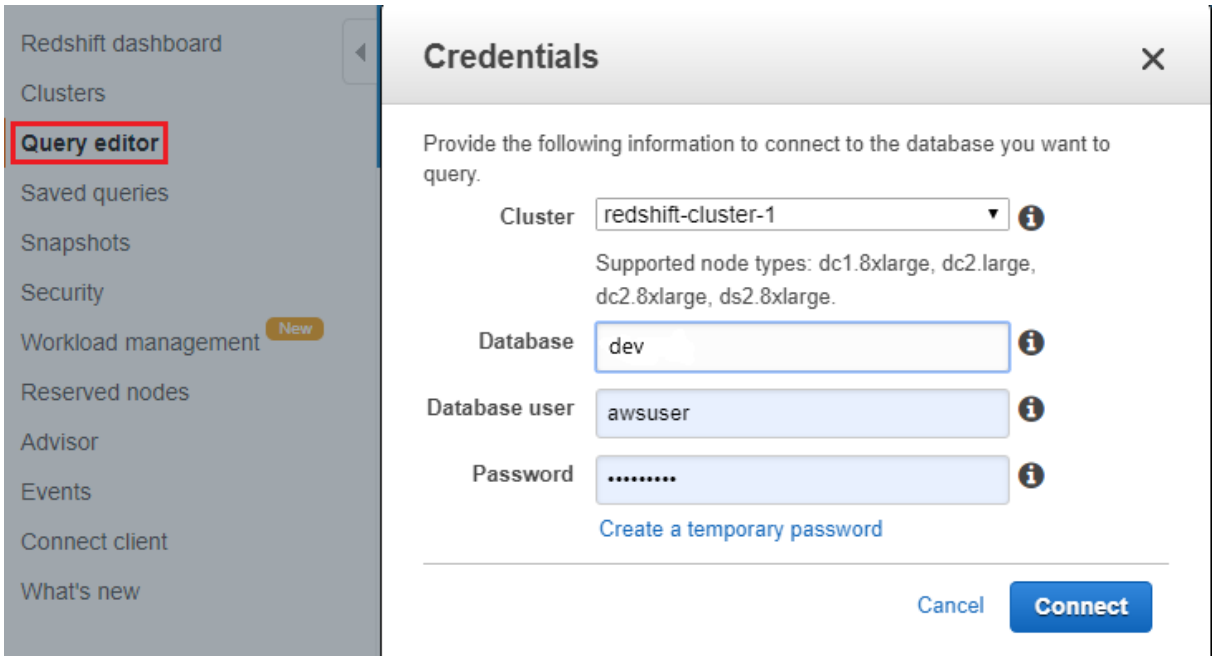
Type	Protocol	Port range	Source
Redshift	TCP	5439	0.0.0.0/0

Slika 29: Postavljanje pravila za nadolazeće podatke

Radi izgradnje željene baze biramo između urednika za upite (engl. *query editor*) ili *SQL* klijenta (engl. *SQL client*). U ovom primjeru koristim urednik za upite, no ujedno se može koristiti i primjerice *SQL Workbench*; stvar je preferencija. Kako bi se koristio urednik za upite moramo napraviti novi *IAM* korisnički račun. Za razliku od *IAM* korisnika, *IAM* uloga ne nosi nikakve akreditive (šifra ili ključ za pristup) koji su asocirani s njom. Ulogu može posjedovati tko god želi s obzirom na potrebu u danom trenutku, dok *IAM* korisnik predstavlja osobu koja komunicira s uslugom. Stvorivši račun preko *IAM* usluge s opcijom „Create user“ i pridodavši novom korisniku *AmazonRedshiftQueryEditor* i

[24] *Amazon Redshift features.* Preuzeto s https://aws.amazon.com/redshift/features/
[25] TCP protokol (engl. *Transmission Control Protocol*) je protokol za kontrolu prijenosa podataka. Dok se IP brine za identifikaciju i vezu s mrežom, TCP se brine o razmjeni podataka s danom mrežom.

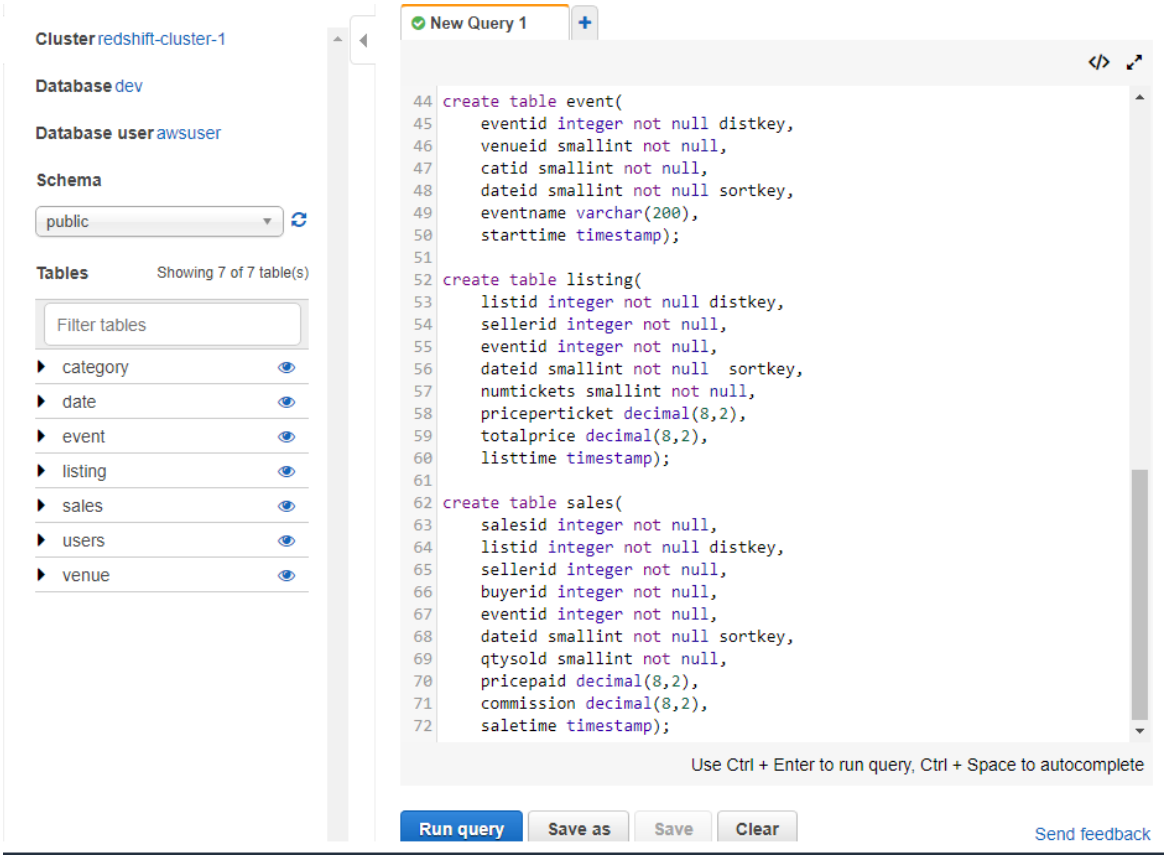
AmazonRedshiftReadOnlyAccess, prijavljujemo se te s lijeve strane ekrana biramo „Query editor“ (slika 30).



Slika 30: Query urednik

Stvaranjem željene baze podataka, čije smo ime na samome početku definirali, i stvaranjem više tablica s definiranim poljima, kao što vidimo na slici 31, spremni smo unijeti podatke iz, u ovom slučaju, S3 kante u navedenu bazu podataka. Stvorivši 7 tablica unutar baze, popunio sam ih s podatcima prema poljima.

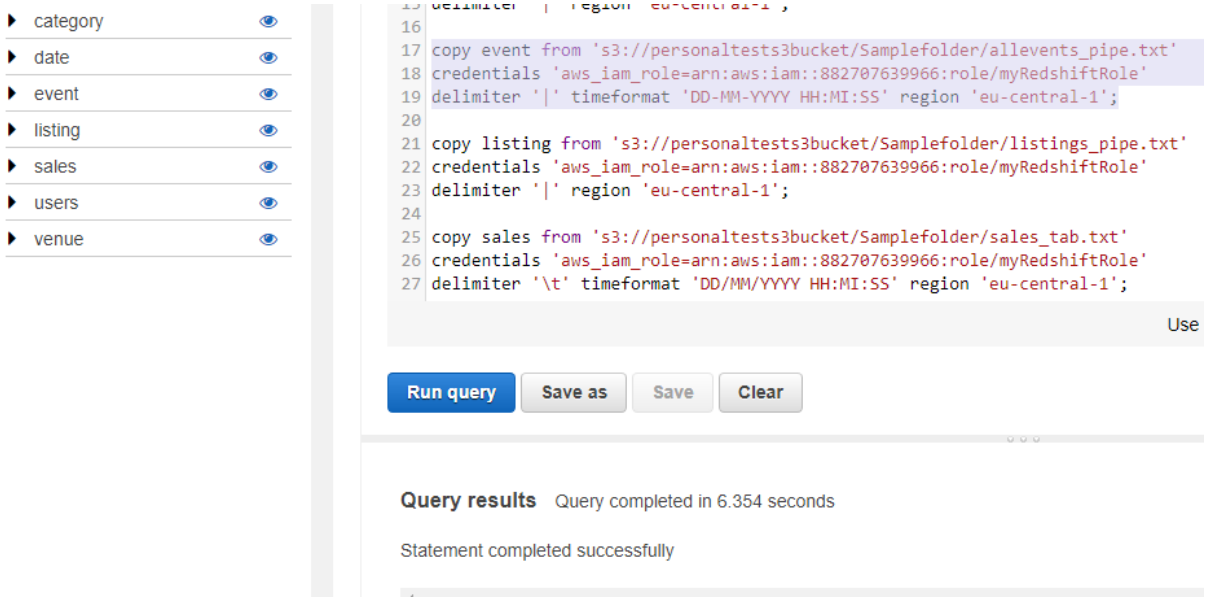
U korisnik tablicu (engl. *users*) je uneseno 49 990 zapisa unutar četiri sekunde, u događanja tablicu (engl. *events*) je uneseno 8798 zapisa unutar 6.84 sekunde, u tablicu kategorija (engl. *category*) je uneseno 11 zapisa unutar 1.12 sekundi, 364 zapisa su unesena u tablicu datumi (engl. *dates*) unutar 3.22 sekunde, tablica popis (engl. *listing*) je popunjena s 235 497 zapisa unutar 4.58 sekundi, tablica prodaja (engl. *sales*) sa 172 456 zapisa unutar 10.32 sekunde i tablica sastajalište (engl. *venue*) sa 404 zapisa unutar 2,5 sekunde. Način za unos podataka preko s3 kante je vidljiv na slici 32.



Slika 31: Kreirane tablice u bazi

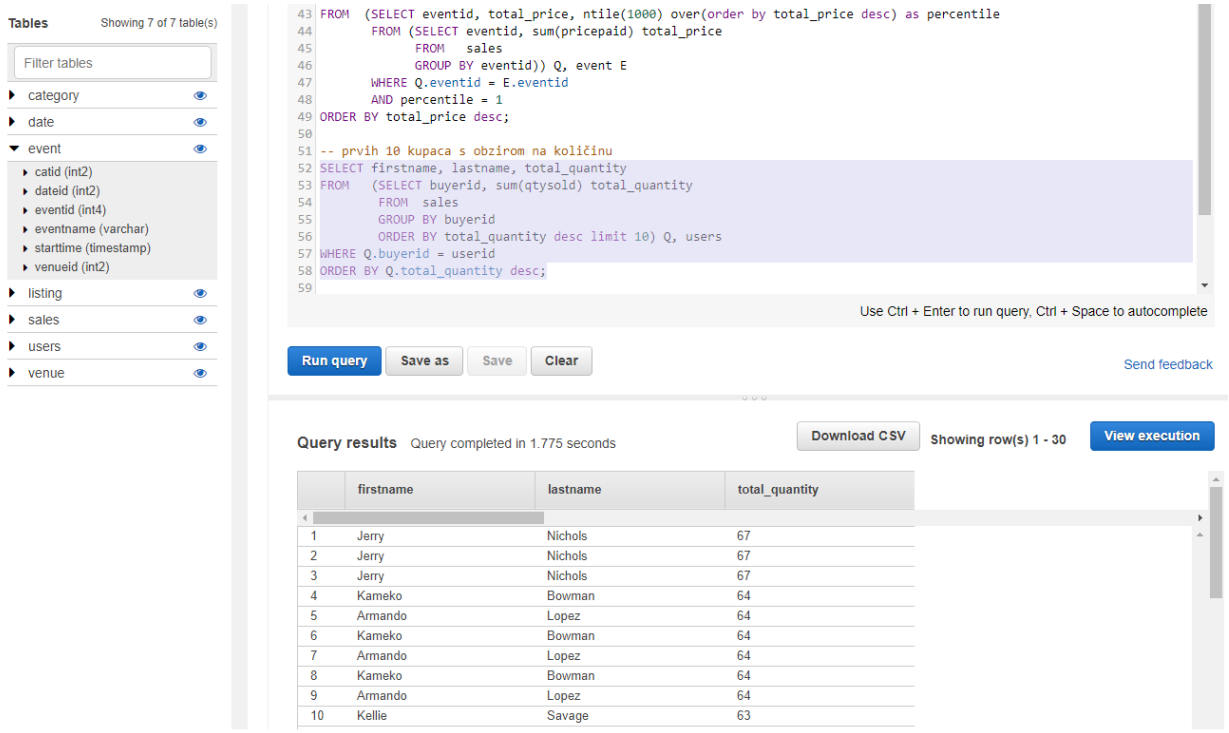
Dokument[26] koji sam koristio radi popunjavanja tablice se nalazi na službenim stranicama *Amazona*. Brzina unosa podataka je skoro pa zanemariva, no vidljivo je kako za unos datuma i sata te njihova formatiranja u željeni oblik treba nešto više vremena od unosa primjerice imena ili pojmova (u tablicama polja tipa *varchar*). Ujedno se u obzir prilikom mjerenja vremena izvršavanja treba uzeti i broj čvorova koji se koristi, tip čvorova, distribuciju podataka, konkurentnost operacija koje se izvršavaju itd.

[26]Preuzeto s https://docs.aws.amazon.com/redshift/latest/gsg/samples/tickitdb.zip.



Slika 32: Unos podataka preko S3 kante

Kompleksnost i kvaliteta upita znatno utječu na brzinu ispisivanja podataka, no uzevši u obzir veličinu i kompleksnost sustava koji je stvoren da na njemu radi veći broj korisnika paralelno, te nadgledajući pritom trenutnu ulogu i ulogu svakog korisnika, normalno je za zaključiti kako bazični upiti poput ovih prikazanih na slici 33 nisu pretjerano zahtjevni sustavu. No uzevši u obzir da velik broj poslovanja koristi jednostavne upite radi iteracije kroz nizove podataka kako bi dobili primjerice sliku o potrošnji kupaca u određenome vremenskome razdoblju, radi grafičkih prikaza i usporedbe podataka, sasvim je legitimno za naglasiti da je ovom izvedbom dobar dio usluga zadovoljen.



Slika 33: Upiti provedeni nad bazom

8.2.3. Naglašene karakteristike

Jedna od prvih i dosta naglašenih karakteristika kojom se AWS ponosi je MPP ili masovno paralelno procesiranje (engl. *Massively Parallel Processing*) koje omogućuje izvršavanje iznimno kompleksnih upita nad velikom količinom podataka. Velik broj računalnih čvorova obrađuje dobivene upite gdje svaki sloj svakog čvora izvršava iste kompilirane segmente upita koji se odnose na podatke. Odabirom odgovarajućeg distribucijskog ključa za svaku tablicu, optimizira se distribucija podataka radi balansiranja količine posla i umanjivanja prijenosa podataka iz čvora u čvor (Stitch).

Iako je već naveden stupčasti način pohrane podataka (engl. *columnar data storage* ili *c-store*) bitno ga je naglasiti ponovno jer se i dalje velik broj skladišta podataka bazira na pohranjivanje podataka u redovima. Stupčasti način pohrane tablica unutar baza podataka smanjuje broj zahtjeva na disku i smanjuje količinu podataka koju treba učitati s diska. Učitavanjem manje količine podataka u memoriju omogućuje *Amazon Redshiftu*

izvršavanje više procesa unutar memorije prilikom izvršavanja upita. Kompresija podataka povećava prostor skladišta, smanjujući pritom procese na disku i pospješujući izvođenje upita. Učitavanje manje količine podataka u memoriju omogućuje alociranje veće količine memorije radi obrade podataka. Zbog navedenog tipa pohrane, koji pohranjuje slične podatke sekvencijalno, *Amazon Redshift* uspijeva primijeniti adaptivne kompresijske kodove specifično povezane sa stupčastim tipovima podataka. Navedeni kodovi pomažu oko kompresije manjih grupa podataka i optimiziraju brzinu izvršavanje upita (Stonebraker, 2005).

Izolacija mreže (engl. *network isolation*) se odnosi na virtualni privatni oblak baziran na *Amazon VPC* usluzi – privatnoj izoliranoj mreži u *AWS* oblaku. *Amazon VPC* je mrežni sloj povezan s *Amazon EC2*. Ovaj sloj usluga se odnosi na poduzeća koja dodatno žele zaštititi svoje podatke. Glavne karakteristike *VPC*a su:

- virtualna privatna mreža,
- podmreže – IP adrese povezane s korisnikovim *VPCom,*
- tablica rute – nizovi pravila, zvanih rute, radi lakšeg utvrđivanja gdje se mrežni promet upućuje i na koji način se koristi,
- internetski prolaz – omogućuje komunikaciju između *VPC* mreže i interneta,
- *VPC*ova kranja točka – omogućuje spajanje *VPCa* na *AWS* usluge i servise od strane *PrivateLinka* bez potrebe za internetskim prolazom, NAT napravama, *VPN* vezi itd. Promet između *VPCa* i drugih usluga ne izlazi iz *Amazon* mreže.

Tolerancija kvarova (engl. *fault tolerance*) je osobina sustava da radi iako neke od njegovih komponenti koje služe za izgradnju sustava ne rade. Usluge poput *EC2* i *Redshifta* nude specifične usluge poput zona dostupnosti (engl. *availability zones*), elastične IP adrese i snimke sustava koje sustav poput *AWSa* u potpunosti iskorištava. Tolerancija kvarova je u ovom slučaju usko povezana s geografskim regijama i dostupnim zonama unutar regije (engl. *availability zones*) koje omogućuju jednostavan pristup lokacijama s redundantnim podatcima. Dostupne zone u suradnji s ponuđenim regijama unutar *AWSa*

omogućuju toleranciju kvarova stvarajući više lokacija s istim podatcima koje posjeduje samo poduzeće, odnosno glavna odabrana regija za skladištenje podataka.

Ograničenje konkurentnosti (engl. *concurrency limits*) određuje maksimalan broj čvorova koje korisnik može koristiti u određenome trenutku. Takva ograničenja osiguravaju određenu količinu čvorova za sve korisnike demokratizirajući skladište podataka time. Međutim ugrađena je i doza fleksibilnosti; broj čvorova koji je dostupan klasteru se određuje s obzirom na tip klastera. Limitacije se ujedno definiraju s obzirom na regije, ne limitirajući sve korisnike jednako. No s obzirom na potrebe, korisnici mogu predati zahtjev za povećanjem limita koji koriste (Amazon Web Services).

8.2.4. Primjer korištenja

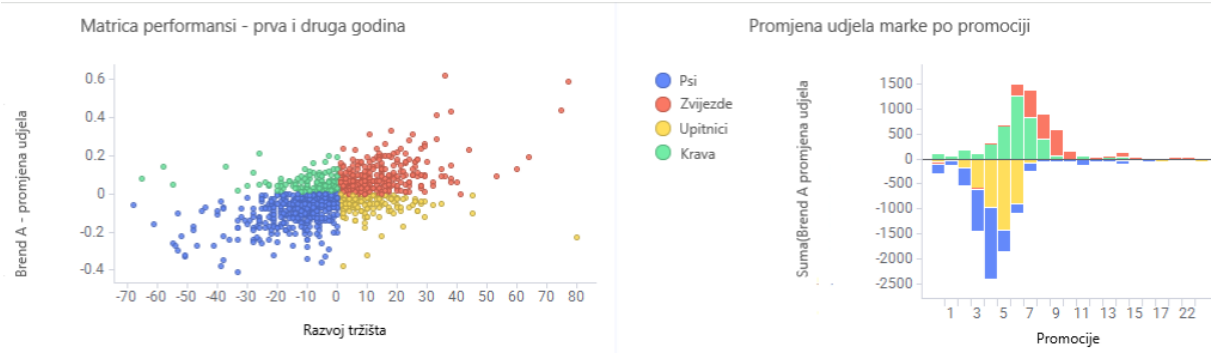
Jedan od modernijih, aktualnijih i većih primjera kako se koristi *AWS Redshift* je *Pinterest* aplikacija. *Pinterest* je vizualni sustav za otkrivanje recepata, stilova oblačenja i dizajna prostora, knjiga i raznih drugih zanimljivosti u poljima interesa. S obzirom na milijune korisnika diljem svijeta, *Pinterest* ima potrebu za obradom velike količine podataka radi dostave najboljeg proizvoda s obzirom na interes svakog korisnika zasebno. U procesiranje danih podataka spada i filtriranje neželjene pošte, korisničkih podataka, predlaganje novih proizvoda s obzirom na povijest pretraživanja, dodavanje novih proizvoda i sadržaja generalno, prijedlozi oko praćenja (engl. *follow*) drugih korisnika sa sličnim interesima itd.

Koristeći manja skladišta do 2013. godine poput *Hadoopa* ili *Hivea*, *Pinterest* je dobio potrebu za većim i jačim skladištima koja neće rigorozno povećati njihove troškove. *AWS* instance (engl. *instances*) omogućavaju balansiranje računalnih, memorijskih i mrežnih resursa te se mogu koristiti za širok spektar poslova poput *A1* instanci koje omogućuju aplikacijama direktan pristup fizičkim resursima poslužiteljskog servera, poput procesora ili memorije. Na zahtjev (engl. *on demand*) instanca je instanca koja se koristi na zahtjev i po potrebi. Automatizmi su mogući, no nad tom instancom korisnik ima potpunu kontrolu

– odlučuje kada se aktivira, kad zaustavlja, kad hibernira i kada se terminira., te u slučaju *AWSa* naplaćuje samo vrijeme dok se aktivno koristi što je 0.84 dolara po satu. Uglavnom se ova mogućnost koristi od strane *Pinteresta* za testiranje softvera u raznim konfiguracijama kako bi procijenili dobivaju li željene rezultate od sustava prije samog kupovanja ili rezerviranja instance. Instanca na trogodišnji ugovor ispada 999 dolara na terabajt podataka u skladištu što je jeftinije od konkurenata kao što ćemo vidjeti u kasnijem poglavlju.

Ujedno se stavlja naglasak na automatizirano održavanje i dograđivanje sustava, gdje u trenutku stvaranja instance korisnik više ne mora brinuti o hardveru ili softveru koji stoji iza toga svega, te izgled same konfiguracije radi ručne optimizacije sustava. Ugrađena administratorska kontrolna ploča koja se fokusira na postavljanje restrikcija, pristupa sigurnosti, s3 kanti, autorizacije pristupa drugim korisnicima i mnogih drugih komponenti je relativno jednostavna za korištenje i poprilično direktnog pristupa i opisa s obzirom na količinu proizvoda i komponenti koje *AWS* nudi.

Tibco kao alat koji služi za vizualnu prezentaciju podataka pomoću grafova, grafikona i drugih vizualnih komponenti (slika 34) je jedan od partnera *AWS*, posebice povezan s *Amazon Redshift* uslugom.



Slika 34: Matrica performansi generirana uz pomoć Tibcoa[27]

[27] Prilagođeno prema izvoru (https://docs.tibco.com/pub/spotfire_server/10.0.0/doc/html/en-US/TIB_sfire_bauthor-consumer_usersguide/GUID-B5CB3CA8-6EBF-47E0-A411-FAAFA579BEF9.html)

Koristeći podatke iz skladišta podataka i definirajući upite njihove obrade, *Tibco* na slici 34 prikazuje izvještaj o prodaji uspoređujući dvije godine. U svakom slučaju je ovo jedan od manjih i osnovnijih primjera prikaza filtriranog sadržaja radi stvaranja određenog tipa izvještaja koristeći pritom *Redshift* kao uslugu s obzirom da se većina obrađenih i testiranih upita koristi interno s obzirom na potrebe poduzeća. U slučaju *Pinteresta* je to generalno aktivnost korisnika i sadržaj koji je usko povezan s njima i koji se nalazi u raznim formatima, dok za druge to može biti jednostavan prikaz zabilježene prodaje u određenom vremenskom periodu. Raznovrsnost, količina usluga i broj poznatih klijenata dovoljno opisuju *AWS* i *Amazon Redshift* kao, s razlogom, jedan od top 3 sustava za skladištenje podataka u svijetu.

8.3. Google BigQuery

Google BigQuery je potpuno uređeno i upravljano, skalirajuće petabajtno skladište podataka. *BigQuery* je *NoOps*[28]. S obzirom na rečeno, vidljivo je kako je *Google BigQuery* podosta sličan *Snowflakeu* po metodi plaćanja i *AWSu* s obzirom na naglašenu veličinu skladišta i same kompanije. Naglasak je stavljen na pristup koji se vrši kroz konzolu u oblaku (engl. *Cloud Console*) ili preko klasičnog web korisničkog sučelja koristeći alat naredbenog retka ili koristeći *BigQuery REST API* preko kojeg postoji mogućnost za pozivanjem klijentskih knjižnica kao što su *Java, .NET ili Python* (Data Warehouse Guide). Postoji niz alata koji u suradnji s *Google BigQueryem* omogućuju vizualizaciju i učitavanje podataka kao što ćemo kasnije vidjeti u primjeru korištenja.

Google BigQuery se ne bazira na serveru odnosno smatra se „bez servernom“ uslugom (engl. *serverless*) ili skladištem podataka kao uslugom (engl. *data warehouse as a service*). Ne postoje serveri koji se trebaju nadzirati ili softver vezan uz baze podataka koji se treba

[28] NoOps je u potpunosti automatizirano skladište, bez potrebe za vođenjem infrastrukture i bez potrebe za administratorom baze podataka. Potpunom automatizacijom postiže lakše analiziranje podataka radi pronalaženja relevantnih informacija koristeći pritom SQL, te se plaća s obzirom na količinu upotrebe u danom trenutku.

instalirati; svime time se bavi *BigQuery* zajedno uz održavanje infrastrukture, što uključuje skalabilnost.

Google BigQuery za razliku od *Snowflakea* i *Amazon Redshifta* ne zahtjeva da se unaprijed odredi veličina i jačina klastera koja će se koristiti za obradu dobivenih upita, već sam procjenjuje broj čvorova koji će se koristiti za obradu upita koji se baziraju na *Dremel* sustavu koji koristi arhitekturu višeslojnog stabla radi skaliranja *SQL* upita (Ogilvy, 2018).

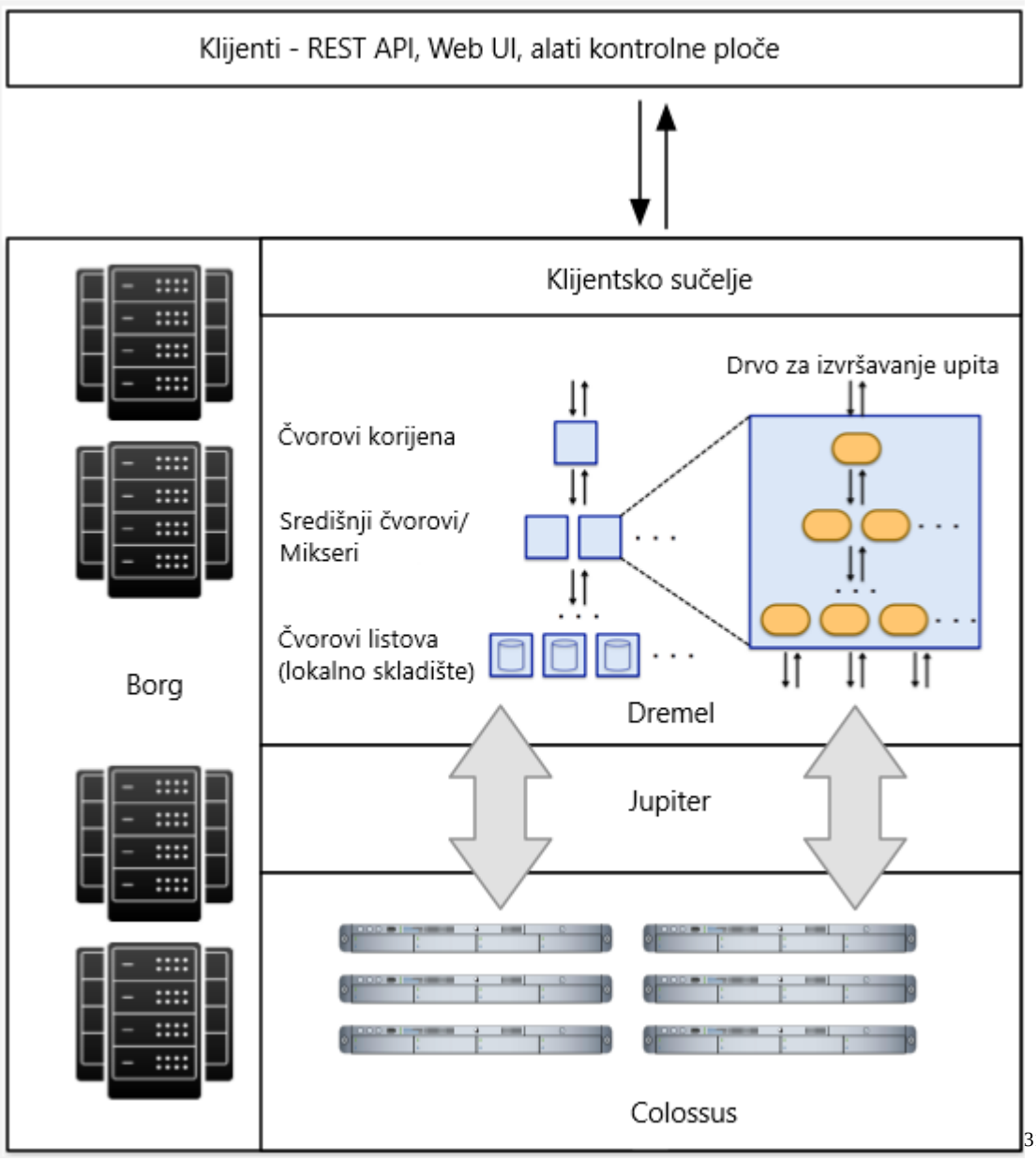
8.3.1. Arhitektura

BigQuery je izgrađen na bazi *Dremel* tehnologije koja je u produkciji interno u *Googlu* od 2006. godine. *Dremel* je *Googleov* interaktivni ad-hoc[29] sustav za analizu samoisčitivih podataka (engl. *read-only*). *BigQuery* i *Dremel* dijele istu temeljnu strukturu. Inkorporirajući stupičastu pohranu i *Dremelovu* višeslojnu arhitekturu stabla, *BigQuery* nudi jedinstven pristup i značajne performanse sustava. *BigQuery* klijent aktivno komunicira s *Dremel* sustavom preko klijentskog sučelja. *Borg* – *Googleov* sustav koji upravlja klasterima velikih razmjera – alocira računalni kapacitet za *Dremelov* posao (Google, 2012).

Dremelovi zadatci se odnose na čitanje podataka iz *Google Colossus* podatkovnog sustava koji koristi *Jupiter* mrežu[30], izvođenjem raznih *SQL* operacija i vraćanjem rezultata klijentu. Poput *Snowflakea*, arhitektura *BigQueryja* odvaja računalni dio sustava i skladište, dopuštajući neovisno skaliranje jednog i drugog – bitna karakteristika za stvaranje elastičnog skladišta za pohranu podataka.

[29] Ad-Hoc je proces poslovne inteligencije dizajniran kako bi odgovorio na jedinstveno, specifično poslovno pitanje. Korisnici stvaraju izvještaj koji još ne postoji radi dubinskog promatranja određenog statičkog izvještaja radi, primjerice, dobivanja detaljnih informacija o korisničkim računima, transakcijama ili zbirkama podataka.

[30] Jupiter je mreža kreirana od strane Googlea koja se koristi interno radi obrade i pohrane podataka dobivenih iz svijeta. Projekt nastao nakon 5 generacija mreža stvorenih za podatkovne centre radi optimizacije troškova, smanjenja operativnih kompleksnosti i povećanja limitiranog skaliranja od strane mreža podatkovnih centara.



[31]

Slika 35: Arhitektura BigQuery usluge

BigQuery pohranjuje učitane podatke u stupčasti format pod nazivom *Capacitor*. Prilikom unosa podataka, *BigQuery* dekodira svaku kolumnu odvojeno u *Capacitor*

[31] Prilagođeno prema izvoru (https://panoply.io/data-warehouse-guide/bigquery-architecture/)

format. Nakon što su svi podatci u kolumnama dekodirani bivaju ponovno zapisani u *Colossus* (slika 35). Prilikom dekodiranja se velik broj statistika o podatcima dohvaća i pohranjuje što s vremenom služi za planiranje upita. *Colossus* je *Googleov* najnoviji distribuirani podatkovni sustav koji rukuje s replikacijom cijelih klastera, oporavkom sustava, dekodiranjem i raspodijeljenim upravljanjem. Prilikom zapisivanja podataka u *Colossus*, *BigQuery* donosi određene odluke oko inicijalnih strategija dijeljenja koje se baziraju na upite i načine pristupa podatcima. Sveobuhvatno gledano, *Colossus* dopušta dijeljenje podataka u više particija radi omogućavanja brzog paralelnog iščitavanja dok *Capacitor* omogućava veću propusnost prilikom skeniranja i lakšu kompresiju stupaca (Google Cloud Blog, 2016).

U prije navedenoj strukturi stabla, korijenski server (engl. *root server*) prima nadolazeće klijentske upite i stvara rute za upite do sljedeće razine. On je odgovoran za slanje rezultata upita natrag klijentima. Čvorovi lista služe za iščitavanje podataka iz *Colossusa* vršeći filtriranje i parcijalnu agregaciju. Kako bi paralelizirali upit, svaki sloj (korijen i mikser) izvršava svojevrsno prerađivanje upita nakon čega modificirani i particionirani upiti dolaze do čvorova lista radi izvršavanja. Svaki čvor lista omogućava nit za izvršavanje ili određen broj jedinica za procesiranje. Bitno je naglasiti kako *BigQuery* automatski kalkulira broj jedinica koji se koristi za svaki upit, što na kraju ovisi o kompleksnosti upita i njegovoj veličini. Radi lakšeg shvaćanja kako *Dremel* radi i kako struktura stabla izvršava upite, prikazat ćemo proces oko obrade jednostavnog upita (Data Warehouse Guide).

SELECT A, COUNT(B), FROM T GROUP BY A

Nakon što korijenski server zaprimi ovaj upit, prvo što čini je prevođenje upita u formu s kojom sljedeći sloj arhitekture može rukovati. Određuje sve fragmente tablice *T* i zatim pojednostavljuje upit.

SELECT A, SUM(c) FROM (R1i UNION ALL ... R1n) GROUP BY A

Nadalje mikseri modificiraju nadolazeće upite kako bi ih proslijedili čvorovima lista. Čvorovi lista zaprimaju navedene upite i čitaju pritom podatke iz fragmenata *Colossusu*. Središnji čvor čita podatke za kolumne ili polja navedena u upitu, te pritom skenira fragmente, prolazeći paralelno otvorenim kolumnama dokumenta, red po red.

Ovisno o upitu podatci mogu biti pomiješani (engl. *shuffled*) među čvorovima listova. Bitno je znati količinu potrebnog miješanja koje zahtijevaju upiti jer miješanje utječe na performanse i brzinu izvršavanja upita. Zbog toga se predlaže što točnije i preciznije definiranje željenih podataka što ranije u upitu kako ne bi dolazilo do miješanja čvorova i konstantnog prelaska iz jednih u druge; miješanje unutar određenog seta podataka je idealno.

Uzevši u obzir da *BigQuery* naplaćuje za svaki terabajt podataka koji je skeniran od strane čvorova lista, treba izbjegavati pretjerano i učestalo skeniranje. Particioniranje je jedan od načina da se to spriječi – razdjeljivanje svake tablice na manje dijelove, fragmente, kako bi mogli definirati manje opsežni upiti čiji će se rezultati spremati u posredne tablice kako bi nadolazeći upiti skenirali što manju količinu podataka; limitiranje i bolje definiranje upita zajedno s pravilnom arhitekturom omogućava bolje performanse dugoročno gledano.

Na svakoj razini „stabla“ se primjenjuju razne optimizacije kako bi čvorovi vratili rezultate čim su spremni za prikazivanje; to uključuje radnje poput reda prioriteta (engl. *priority queue*) ili učitavanja rezultata (engl. *streaming results*)[32].

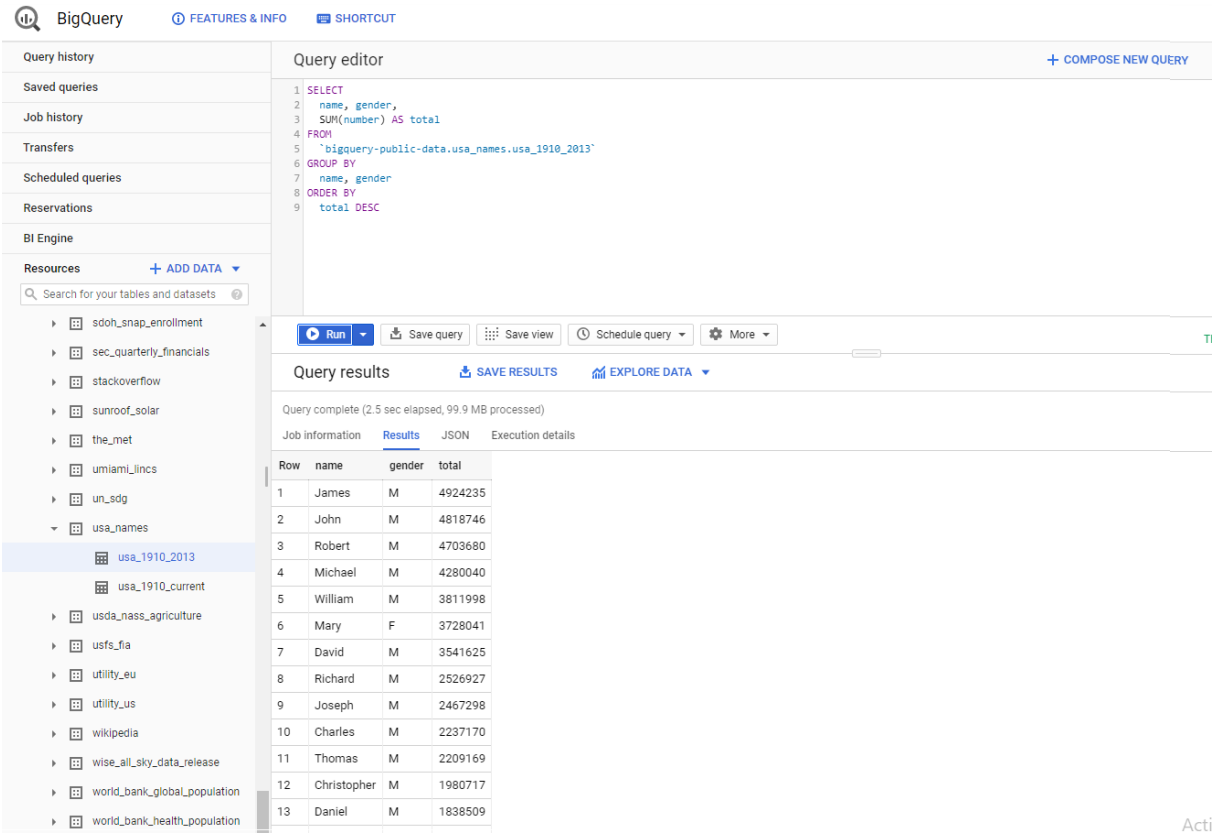
[32] Panoply. *A Deep Dive Into Google BigQuery Architecture.* Preuzeto s https://panoply.io/data-warehouse-guide/bigquery-architecture/

8.3.2. Postavljanje i učitavanje podataka

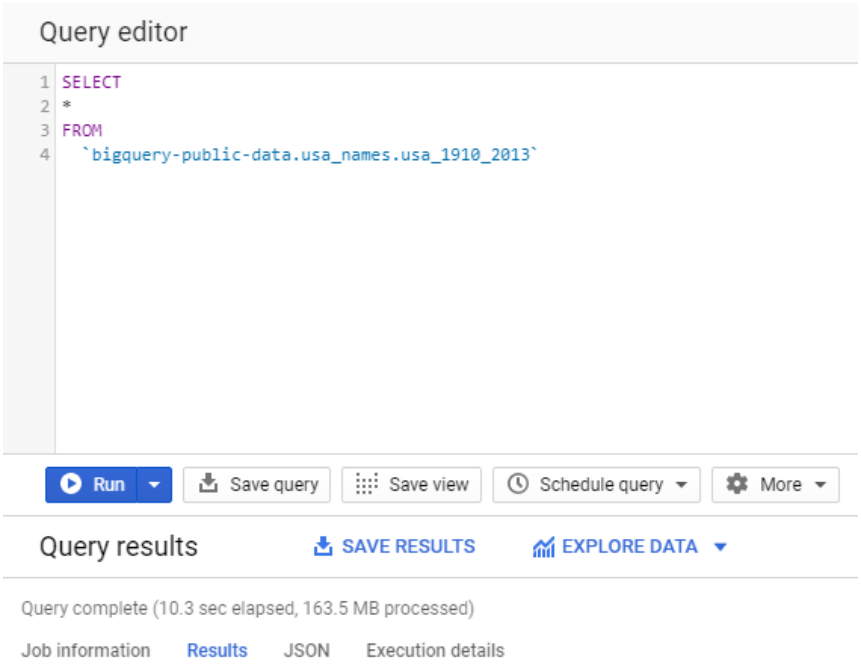
U nadolazećem primjeru odlučio sam se koristiti web korisničkim sučeljem unutar *Googleove Cloud* konzole, no ujedno se može po želji koristiti i naredbeni redak. Pri samome početku korisnik mora pristupiti kontrolnoj ploči na kojoj se nalazi niz ili će se nalaziti niz parametara povezanih s odabranim projektom. Ondje ujedno stvara novi projekt kojemu dodjeljuje parametre poput naziva i odabira partnerske organizacije s kojom se povezuje. Nakon toga pristupamo „API & Services“ opciji unutar platforme gdje na sličnoj razini kao i u *AWSu* definiramo razinu autorizacije za naš korisnički račun. S obzirom da sam u ovome slučaju ja jedini korisnik, postavljam sva administratorska prava na sebe, no s obzirom na usluge koje se koriste i podjelu ljudi u firmi, svatko će imati drugačije definirana prava pristupa. Samim time je u kratkom vremenu definirana uglavnom sva autorizacija unutar sustava.

Potom pristupamo *BigQuery* opciji koja se nalazi na početnoj stranici pod opcijom „big data" na meniju s lijeve strane. Uzevši u obzir da postoji opcija za korištenjem *BigQuery* javnih skupina podataka u ovome primjeru nećemo generirati vlastite niti pretraživati web za pretraživanje setova podataka valjanih veličina.

U danome primjeru, na slici 36, sam koristio listu imena od 1910. do 2013. godine. Vidimo kako je 99.9 MB podataka procesirano unutar svega 2.5 sekunde, dok u slučaju definiranja istoga upita, no sa sveobuhvatnijom naredbom, dobivamo 163.5 MB podataka unutar 10.3 sekunde (slika 37).



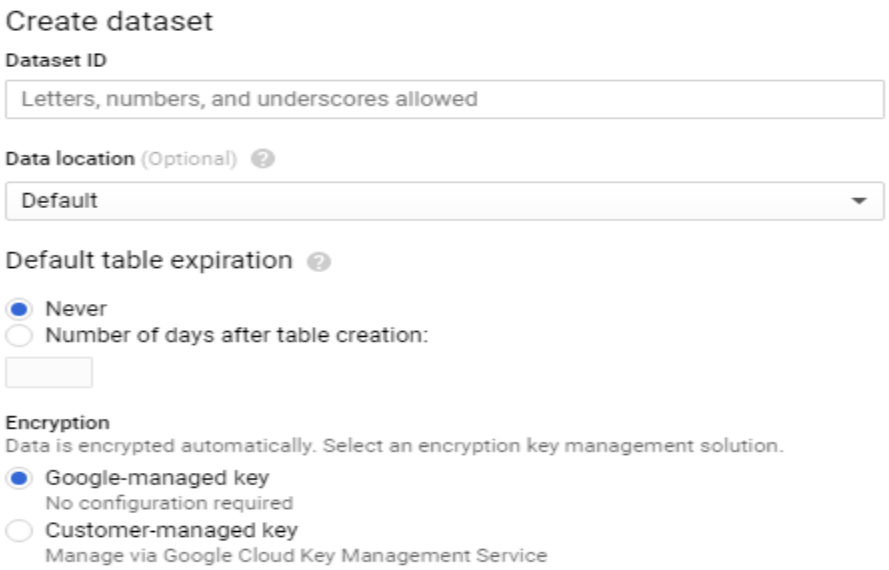
Slika 36: Izvršavanje prvog upita nad listom imena



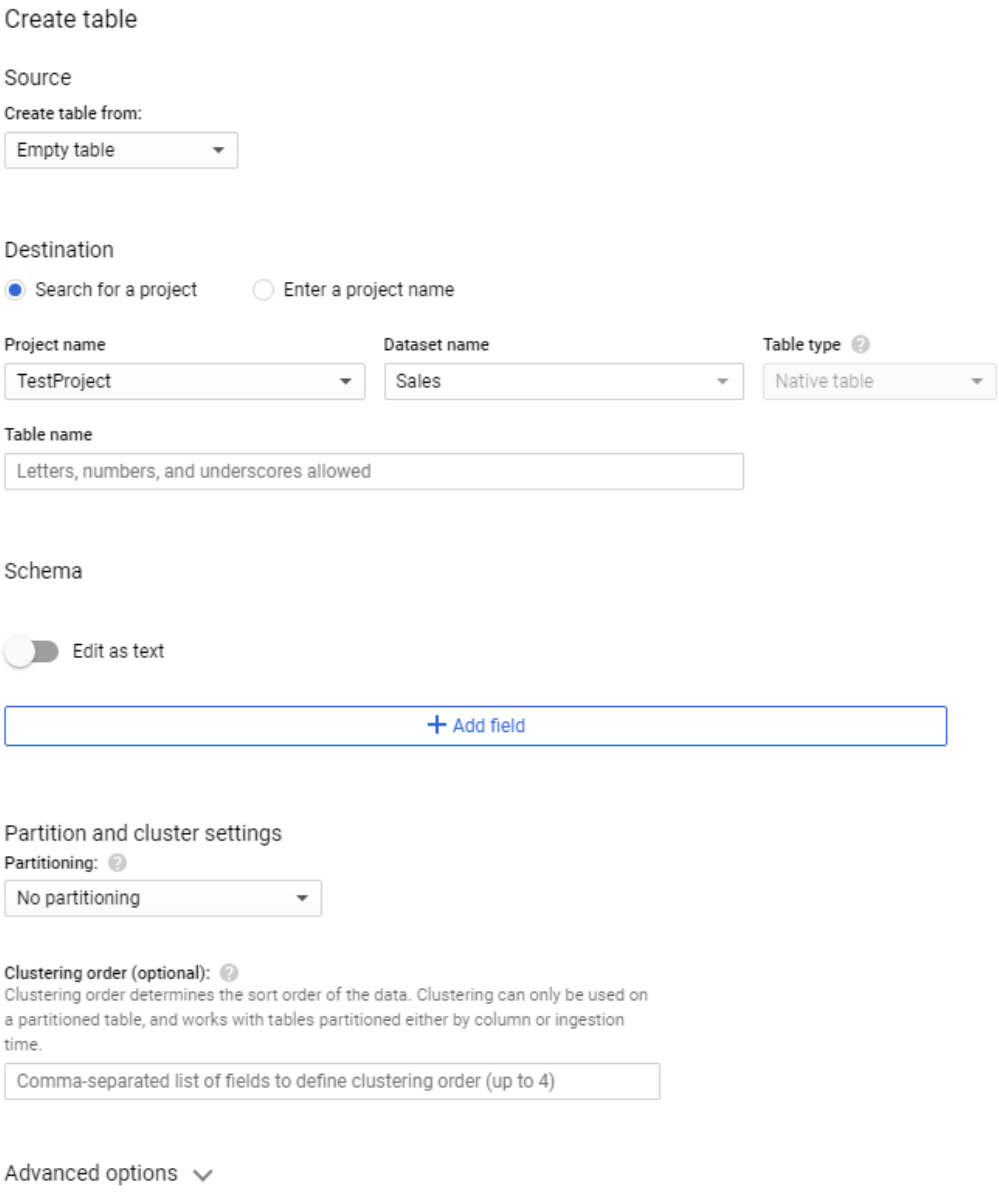
Slika 37: Izvršavanje drugog upita nad listom imena

U današnjici je standardno očekivati kako veliki sustavi poput *BigQuerya*, i onih prije navedenih, brzo obrađuju jednostavne upite gdje se nekad dohvaća i poveći broj podataka, kao što vidimo i u ovome primjeru, no u kasnijoj usporedbi svih triju sustava ćemo vidjeti po čemu se najviše razlikuju s obzirom na dugoročnost korištenja, kompleksnost upita, količinu podataka te broj paralelnih izvršavanja radnji.

Ukoliko želimo unijeti podatke iz postojećih, primjerice, CSV dokumenata koristimo opciju „Create dataset“ gdje postavljamo vrijednosti poput lokacije gdje se stvara set podataka, odnosno poput u *Amazon Redshiftu*, lokacija podatkovnog centra gdje će se stvarati klaster. Nakon što stvorimo set podataka, kao na slici 38, stvaramo tablicu po želji definirajući prvo hoće li biti prazna tablica, povezujemo li ju s *Google Cloud* pohranom, popunjavamo li ju s podatcima s računala itd. Nakon toga biramo projekt pod kojim se stvara tablica, ime tablice i, u ovome slučaju, njen tip kao što je primjerice CSV, te definiramo polja u koja ćemo unijeti podatke. U ovome slučaju koristim datoteku s imenima, spolom i brojem pojava određenih imena iz 2014. godine. Dodatno korisnik može definirati broj dopuštenih grešaka bez da se prekine unos podataka, ignoriranje nepoznatih vrijednosti i niz drugih opcija (slika 39).



Slika 38: Stvaranje seta podataka



Slika 39: Stvaranje i definiranje tablice

BigQuery nudi poprilično jednostavan pristup po pitanju postavljanja rade okoline radi unosa tablica, definiranja upita, iščitavanja podataka i povijesti procesuiranih upita. Postoje ujedno i opcije poput označivanja određenog projekta u slučaju da ih ima veći broj ili se jednostavno žele označiti najaktualniji, unos ugniježđenih i repetitivnih *JSON* podataka, prijenos podataka iz s3 kanti, *Snowflakea* i drugih podatkovnih skladišta.

Ujedno je *Google Cloud* konzola poprilično jednostavna za snalaženje i korištenje, nudeći niz usluga u meniju s lijeve strane dok istovremeno prikazuje podatke aktualnog projekta s kojim se bavimo, olakšavajući prve korake pri ulasku u svijet *BigQuerya* i stvarajući kvalitetno korisničko iskustvo.

8.3.3. Naglašene karakteristike

S obzirom da *BigQuery* uvelike inkorporira niz karakteristika koje smo vidjeli u prijašnjim sustavima, neke od njih ću vjerojatno ponovno naglasiti zbog njihove važnosti trudeći se izdvojiti one jedinstvene. Tako su jedna od prvih karakteristika navedenog skladišta **utori** (engl. *slot*). *BigQuery* utor je jedinica računalnog kapaciteta potrebna da izvrši određeni *SQL* upit. *BigQuery* automatski izračunava koliko je utora potrebno za svaki upit ovisno o veličini upita i kompleksnosti. Pristup većem broju utora ne garantira bržu obradu upita, međutim veći broj utora bi mogao pospješiti performanse većih i kompleksnijih upita, kao i performanse paralelno izvršavanih radnji (Google Cloud).

BigQuery automatski rukuje brojem utora korisnika s obzirom na korisnikovu povijest, količinu korištenja i potrošnju. Kada *BigQuery* izvrši određeni posao, konvertira deklerativnu *SQL* izjavu u graf radnji (engl. *graph of execution*) koji je podijeljen u određen broj stadija koji su sami po sebi sastavljeni od više setova koraka za izvršavanje. *BigQuery* koristi paralelnu arhitekturu kako bi što veći broj korisnika mogao izvršavati radnje paralelno, te ujedno uz to svaki utor izvršava individualnu radnju zasebno na svakom stadiju upita. Primjerice, ako *BigQuery* zaključi da je faktor optimalne paralelizacije stadija upita 10, traži 10 utora kako bi procesirali stadij upita. Ujedno ako upit ima potrebu za, primjerice, 2000 utora, ali ih je samo 1000 slobodno, *BigQuery* će konzumirati dozvoljenih 1000 te rezervira sljedećih 1000 kad se oslobode. Što znači da u trenutku kada 100 utora završi posao automatski prelaze na stadij upita koji se trenutačno obrađuje. Svaki put kada je novi upit predan, kapacitet se automatski ponovno razdjeljuje preko upita koji se

izvršavaju. Uz to se individualne jedinice mogu pauzirati, nastaviti izvršavati ili staviti u red za čekanje (Google Cloud).

Tradicionalno gledano **ukrcavanje** (engl. *onboarding*) novih podatkovnih analitičara u projekt zahtjeva podosta vremena. Radi omogućavanja provođenja jednostavnih upita analitičarima bilo je potrebno pokazati im gdje se svi potrebni podatci nalaze i postaviti niz veza i alata radi pristupanja tim podatcima. Danas kako bi se postiglo ukrcavanje, analitičarima se pridodaju prava pristupanja na određeni projekt. Najrelevantniji podatci analitičarima bi se danas nalazili unutar kanti za pohranu u oblaku (engl. *cloud storage buckets*) gdje mogu kolabirati, identično s3 kantama u *AWSu*.

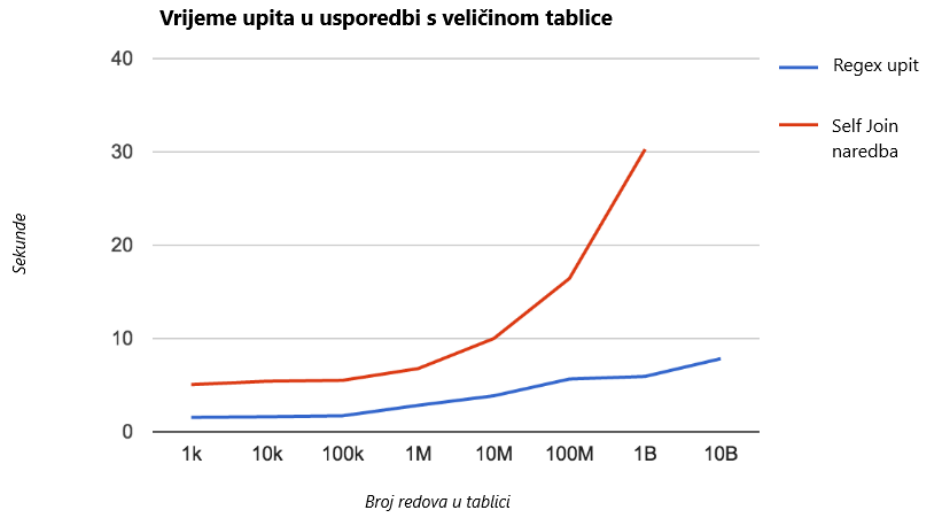
Denormalizacija inače podrazumijeva zapisivanje činjenice, zajedno sa svim njenim dimenzijama, u jednu tabličnu strukturu. Primjerice, za prodajne transakcije, svaka činjenica bi se unijela u zapisnik zajedno s dimenzijama koje ju prate poput tipa narudžbe i informacija o kupcu. Suprotno tome, metoda denormalizacije iskorištava BigQueryjevu izvornu podršku za ugniježđene i repetitivne strukture (engl. *nested and repeated*) u *JSON* ili *Avro* obliku. Prezentirajući zapise koristeći ugniježđenu i repetitivnu strukturu pruža prirodniji pregled postojećih podataka. U slučaju prodajne transakcije, vanjski dio *JSON* strukture sadrži narudžbu i informacije o korisniku, dok unutrašnji dio sadrži individualno naručene proizvode, koji su prikazani kao ugniježđeni, repetitivni elementi, primjerice:

```
{
  "narudzbaID": "NARUDZBA",
  "korisnikID": "EMAIL",
  "korisnikIme": "IME",
  "Datum": "VRIJEME",
  "lokacija": "LOKACIJA",
  "kupljeniArtikli": [
   {
    "sku": "SKU",
    "opis": "OPIS",
    "kolicina": "KOL",
    "cijena": "CIJENA"
   },

   {
```

```
    "sku": "SKU",
    "opis": " OPIS ",
    "kolicina": "KOL",
    "cijena": "CIJENA"
   }
  ]
}
```

Pozitivna strana denormalizacije se odnosi na mogućnost bržeg izvršavanja OLAP radnji i zadataka bez korištenja JOIN funkcionalnosti. Izvršavajući OLAP operacije nad normaliziranim tablicama, veći broj tablica se treba spojiti (engl. *joining*) radi provođenja željene agregacije. JOIN radnje su omogućene u *BigQueryu*, no predloženo ih je koristiti samo nad manjim tablicama jer je u većini slučajeva denormalizirana struktura optimiziranija (slika 40).



Slika 40: JOIN u usporedbi s Regex upitom

Particioniranje u *BigQueryju* je omogućeno po datumu unosa podataka, *TIMESTAMP* ili *DATE* kolumni u tablicama ili s obzirom na integer kolumnu. Particionirana tablica je podijeljena na segmente, particije, koji joj omogućavaju lakše kontroliranje podataka i provođenje upita nad istim. Smanjujući veću tablicu u manje particije postižu se bolje performanse, te se lakše kontroliraju troškovi ograničavajući broj bajtova koje upit

iščitava. Navedena opcija se omogućava prilikom procesa stvaranja tablice. Dodatno tome, korisnik može definirati rok trajanja podatcima koji su particionirani (Google Cloud).

BigQuery zapisuje sve upite u tablicu. Tablica je ili definirana striktno od strane korisnika ili je privremeno stvorena **predmemorirana** tablica rezultata (engl. *cached results table*). Ne postoje nikakvi troškovi za takve tipove kratkoročnih tablica, međutim ako se podatci zapisuju u dugotrajnu tablicu, naplaćuje se pohrana podataka. Određena ograničenja navedene vrline *BigQuerya* su:

- kada se pokreće duplicirani upit, *BigQuery* pokušava ponovno koristiti predmemorirane rezultate. Kako bi se dohvatili podatci iz predmemorije, duplicirani upit mora biti jednak izvornome,
- izjave, poput *DELETE/UPDATE/INSERT*, vezane uz jezik podatkovne manipulacije (engl. *Data Manipulation Language*) se ne mogu koristiti nad podatcima u predmemoriji,
- kako bi upiti ostali zabilježeni unutar predmemorirane tablice rezultata, rezultati moraju biti manji od maksimalne dopuštene veličine odgovora na upit. Da jedinica biva definirana s obzirom na kompresirane podatke što znači da je 10 GB, koji su navedeni kao okvirno pravilo, promjenjivo.

Prijenos (engl. *streaming*) podataka u *BigQuery* za razliku od učitavanja (engl. *loading*) podataka u skladište olakšava određene poslove podatkovnim analitičarima. Prijenos podataka omogućava unos 1,000,000 redova po sekundi po projektu, 1GB podataka po projektu u sekundi, a tu opciju se može postići fragmentiranjem tablica. U slučajevima gdje se setovi podataka rijetko i sporo mijenjaju, prijenos podataka direktno u *BigQuery* je svakako opcija koja se može iskoristiti, dok je cijena cijelog procesa svega 0.01 dolar na 200MB (Google Cloud).

8.3.4. Primjeri korištenja

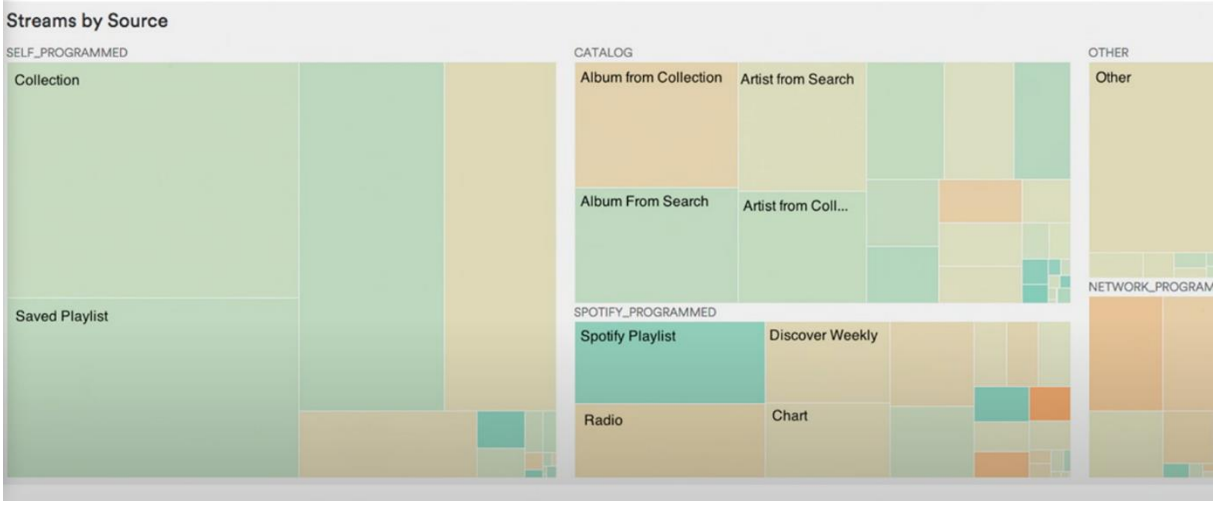
Spotify je servis koji digitalno prenosi muziku, emisije i videe pružajući pristup milijunima pjesama i drugog sadržaja. Kompanija koji ima najveći *Hadoop* cluster u Europi, 23 000 računalnih čvorova i izvršava 11 000 poslova dnevno. U početku su koristili *Scalding* i *Apache Crunch* koji su bili vezani uz produkcijske poslove, dok su za određene *AdHoc* analize koristili *Hive*.

Sve navedene tehnologije su odlične za skaliranje, međutim ne postižu potrebnu brzinu za iteraciju kroz elemente; upitima ponekad treba između par minuta i par sati da se izvrše što nikako ne odgovara količini i veličini posla koji konstantno raste. Primjerice uklanjanje glazbe određenoga umjetnika može rezultirati otežanim generiranjem količine sadržaja koju je taj glazbenik imao, brojčanog pregleda pjesama, količine komentara itd. No iz toga proizlazi još jedan problem, a to je problem analitičkog osvrta kada korisnik želi pregledati, primjerice, navike korisnika koji su slušali tog umjetnika prije uklanjanja glazbe i poslije uklanjanja glazbe, stvarajući okolinu gdje ne samo da zaposlenici nisu sigurni koji im brojevi točno trebaju za analizu, već im je i potreban velik broj testiranja što nas ponovno vraća na brzinu.

Ujedno greške koje se znaju javiti pri samome kraju učitavanja podataka gdje nakon par minuta ili par sati cijeli proces stane i propadne zbog omanje greške u podatcima, situacije u kojima se gubi znatna količina vremena i resursa (Data & Analytics, 2016). Isto tako, velika količina alata je potrebna kako bi se olakšali poslovi analitičarima, poput internih korisničkih sučelja radi lakše preglednosti poslova i olakšavanje poslovnih procesa, te samim time treba te interne alate održavati i prilagođavati poslu korporacije što ponovno uvelike košta.

BigQuery ima znatno veću brzinu, ne posjeduje infrastrukturu koju korisnici trebaju održavati i baziran je na čistom *SQLu*. Lakoća korištenja autorizacije i svrstavanja u grupe s obzirom na projekte ili jednokratne pristupe određenim podatcima koristeći *Google* grupe kao jedan od ekstenzija pridonosi i lakoći nadzora. Prilagodivši sustav svojim potrebama, a ne potrebe sustavu, poduzeće dobiva veći broj mogućnosti s količinom

vremena i resursa koji su se otvorili usavršavajući primjerice analitičku mapu gdje za svakog korisnika *Spotifya* zaposlenik može vidjeti njegove preference po pitanju onoga što sluša i gleda; olakšavajući time posao analitičarima i štedeći ponovno vrijeme (slika 41).



Slika 41: Analitički prikaz korištenja Spotify usluge od strane korisnika[33]

Osposobljujući kompaniju sa sustavom i skladištem potrebnim za analizu i pohranu sve veće količine podataka, uzimaju si za pravo napraviti sljedeći korak u kojemu bi podigli kvalitetu dobivenih podataka fokusirajući se ujedno i na Velike Podatke (engl. *Big Data*), te otvarajući tzv. *Data University* – kurikulum u trajanju od nekoliko tjedana za inženjere koji su zainteresirani za Velike Podatke generalno. Napredak takve razine je nemoguć bez potrebnih elemenata pod koje spada i *Googleov BigQuery* i *Googleova* platforma u oblaku.

[33] Preuzeto s https://www.youtube.com/watch?v=LTVFg6YOjWo (25.5.2020.)

9. Usporedba prezentiranih skladišta podataka

Ovo poglavlje je ključno i finalno poglavlje koje daje sveobuhvatni pregled navedenih sustava radi lakše preglednosti. Nakon samog uvoda u način funkcioniranja skladišta podataka i shvaćanja onoga što ona jesu, te nakon okvirnog upoznavanja sa sustavima koji se smatraju jednima od najpopularnijih u svijetu, potrebno je dobiti generalni pregled i usporedbu navedenih skladišta radi lakšeg odlučivanja koje koristiti bilo za vježbu, učenje ili vođenje posla. Kao što je navedeno već par puta u radu, s obzirom na potrebe poslovanja i prijašnje vođenje poduzeća krajnji sud o tome koje je najbolje od navedenih skladišta može biti samo subjektivne prirode. Krajnji zaključak o tome može donijeti svaki čitatelj za sebe.

9.1. Fivetran usporedba

Kakav tip podataka se koristi? Koliko? Koji tip upita se koristi? Odgovori na ta pitanja su ključni pri ovakvom odabiru; izmjenom podataka koji su pohranjeni i strukture upita i najbrže skladište podataka postaje najsporije.

Fivetran je podatkovni cjevovod koji spaja niz podataka iz aplikacija, baza podataka i dokumenata u korisnikovo skladište podataka, te s obzirom da svakodnevno rade s obrađenim skladištima podataka koja smo naveli, odlučili su napraviti usporedbu između navedenih sustava u koju su uključena još *Azure* i *Presto* skladišta. Korišteni je set podataka, *TPC-DS*[34], između 100GB i 1TB. Set ima 24 tablice u *Snowflake* shemi; tablice predstavljaju web, katalog i prodaje imaginarnog trgovca gdje je najveća tablica imala 400 milijuna redova na 100GB i 4 bilijuna redova na skali od 1TB. Pokrenuto je 99 *TPC-DS* upita

[34] TPC-DS je DSS koji modelira nekoliko generalnih aspekata sustava za potporu pri odlučivanju, uključujući upite i održavanje podataka. Ispitivanje ovakve vrste pruža reprezentativnu evaluaciju performansi sustava za potporu pri odlučivanju, što uključuje i skladišta podataka, ispitujući brzinu izvršavanja upita u modelu s jednim korisnikom i s više korisnika. Provjerava ujedno i performanse održavanja podataka određenoga hardvera, operativnog sustava, i sustava za procesiranje podataka.

koji su poprilično kompleksne prirode; sadrže velik broj agregacija, spajanja i podupita, te je svaki upit pokrenut samo jednom kako bi se spriječilo hvatanje prijašnjih rezultata iz predmemorije. Prikazi navedenih vremena koji će se vidjeti uključuju vrijeme kompajliranja i izvršavanje upita s obzirom da su samo jednom pokrenuti; ovdje se odmah stavlja naglasak na *Redshift* koji je osjetljiv po tom pitanju kompajlirajući upite dulje nego što ih izvršava.

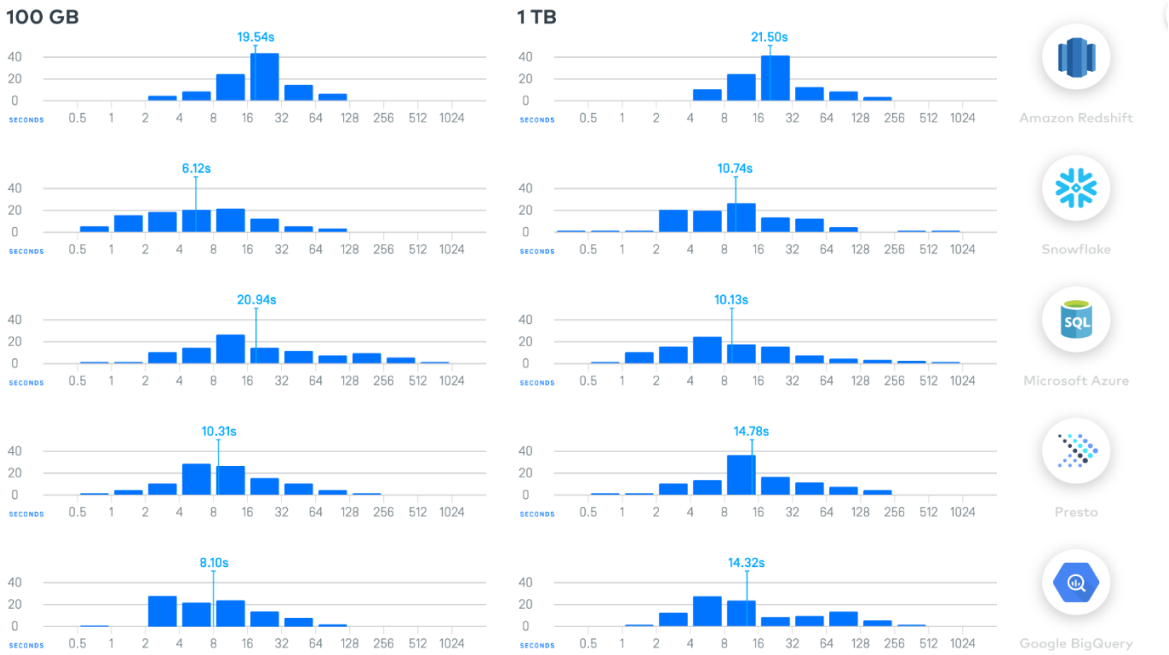
Svako skladište je postavljeno na razini male i velike konfiguracije skladišta radi skaliranja između 100GB i 1TB (slika 42). Ujedno sva skladišta nude razne konfiguracije s kojima se može pospješiti izvršavanje upita poput sortiranja ključeva, particioniranja podataka itd. Niti jedna od tih konfiguracija nije korištena osim kompresije kolumna u *Redshiftu* jer *Snowflake* i *BigQuery* kompresiraju automatski (Fraser, O'Connor, 2018).

	100 GB		1TB	
	Configuration	Cost / Hour	Configuration	Cost / Hour
Redshift	8x dc2.large	$2.00	4x dc2.8xlarge	$19.20
Snowflake	X-Small	$2.00	Large	$16.00
Azure [4]	DW200	$2.42	DW1500c	$18.12
Presto [5]	4x n1-standard-8	$1.23	32x n1-standard-8	$9.82
BigQuery [6]	On-demand		On-demand	

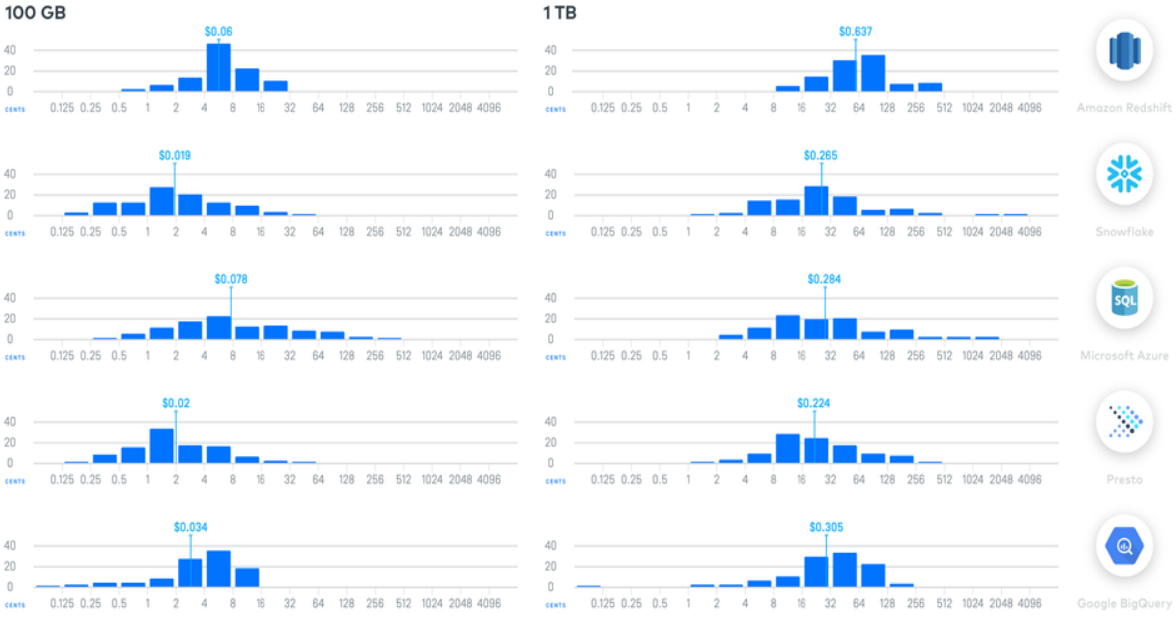
Slika 42: Početna konfiguracija skladišta za testiranje

Sva skladišta su imala odličnu brzinu izvršavanja, pogodnu za *Adhoc* analizu, kao što je vidljivo na slici 43. *Redshift* je bio generalno sporiji zbog sporijeg planera za upit; u slučaju gdje se slični upiti ponavljaju, drugi upit će biti poprilično brže obrađen od prvog. *Snowflake* je svega 2 sekunde brži od *Google BigQuerya* nad dokumentom od 100GB, dok je na razini od 1TB brži svega 4 sekunde; pogledamo li način obrade podataka, vidimo ujedno kako *Snowflake* od samog početka upita vrši određene procese obrade prije

BigQuerya, dok *BigQuery* na neki način neravnomjerno obrađuje dobivene upite rezultirajući nešto sporijim vremenom.



Slika 43: Usporedba brzine



Slika 44: Usporedba cijene između sustava s obzirom na potrošnju

BigQuery naplaćuje po upitu tako da je na slici 44 prikazana stvarna cijena od strane *Google Clouda*. Radi kalkuliranja cijene po upitu za druga skladišta, donijela se okvirna spekulacija oko toga koliko određeno skladište provede vremena bez posla. Primjerice ako se pokrene *X-Small* skladište unutar *Snowflakea* za cijenu od dva dolara po satu, i u to vrijeme pokrenete jedan upit kojemu treba 30 minuta da se izvrši, taj upit je koštao dva dolara, a skladište je bilo bez funkcije 50% vremena. U drugu ruku, ako se pokrenu dva upita od 30 minuta i skladište provede 0% vremena bez posla, svaki upit je koštao samim time jedan dolar[35]. Ovaj način izračuna se bazira na formuli cijene-po-upitu:

*[Cijena upita] = [Vrijeme izvršavanja upita] * [Cijena klastera] / (1 – [Vrijeme bez posla unutar klastera])*

Naravno svako poslovanje treba gledati količinu posla i vrijeme izvršavanja posla zasebno; ako poduzeće ima nagli porast prometa u određenim razdobljima dok u drugima generalno stagnira, *BigQuery* bi bio znatno jeftiniji od drugih skladišta. Međutim ako poslovanje održava određenu konstantu, *BigQuery* bi bio znatno skuplji, dok bi možda *Redshift*, ovisno o dugoročnosti korištenja, bio najjeftiniji (Fraser, O'Connor, 2018).

Cijena i brzina nisu jedini elementi koje treba uspoređivati. Navedena skladišta imaju važne razlike po pitanju kvalitete. S obzirom na stvarne korisnike *Fivetrana*, ustanovljeno je 5 ključnih karakteristika koja razlikuju skladišta podataka:

- Elastičnost: Koliko brzo može skladište povećati i smanjiti kapacitet s obzirom na količinu posla?
- Dostupnost: Kako skladište dozvoljava što veću vremensku iskoristivost bez pauziranja nečijeg rada?
- *JSON* podrška: Može li pohranjivati *JSON* i obrađivati ga?
- Mogu li se postići bolje performanse prilikom korištenja *WHERE* naredbe uz pomoć particioniranja podataka?

[35] Fivetran. *Cloud Dana Benchmark: Redshift, Snowflake, Azure, Presto and BigQuery*. 2018.

- Mogu li se postići bolje performanse prilikom korištenja *JOIN* naredbe uz specificiranje distribucije podataka?

Usporedivši tri skladišta doneseni su zaključci prikazani na tablici 3.

Karakteristike	Redshift	Snowflake	Google BigQuery
Elastičnost	Sati	Minute	Upiti
Dostupnost	Rezervni „prostor“	Distribuirani sustav	Distribuirani sustav
JSON podrška	Bez arraya	Native	Funckije definirane od strane korisnika
Optimizacija WHERE naredbe?	Sortirajući ključ (engl. *Sort key*)	Particioniranje	Particioniranje po datumu
Optimizacija JOIN naredbe?	DistKey/ DistStyle	Ne	Ne

Tablica 3: Usporedne razlike između Redshifta, Snowflakea I Google BigQuerya prema Fivetranu[36]

9.2. Usporedba s prijašnjim istraživanjima

U listopadu 2016. godine, *Amazon* je koristio verziju *TPC-DS* upita u *Redshift* skladištu i *BigQuery* skladištu. Na kraju samog istraživanja je *Amazon* naveo kako je *Redshift* u prosjeku 6 puta brži i da su *BigQuery* upiti izvršavani više od jedne minute. Međutim za razliku od *Fivetranovog* istraživanja, *Redshift* je u tom istraživanju koristio 10 puta veći set podataka (10TB nasprem 1TB) i 2 puta veći klaster koji košta skoro 40 dolara po satu naspram 20 dolara po satu. Pospješili su rad *Redshifta* koristeći *sort* i *dist* ključeve, dok je u prije navedenome primjeru korišten svaki program čist od dodatnih funkcionalnosti, te je u to vrijeme *BigQuery* još uvijek bio u beta fazi, što znači da se zasigurno u dvije godine razvio po pitanju brzine.

[36] Prilagođeno prema izvoru (https://fivetran.com/blog/warehouse-benchmark)

Istraživanja provedena od strane trgovaca koji tvrde da je njihov proizvod najbolji bi trebala biti uzeta sa zrnom soli. Međutim i dalje velik broj informacija nije dan; je li *Redshift*, osim što je koristio enorman klaster, ujedno alocirao svu memoriju na jednog korisnika kako bi se upiti procesirali što brže iako to nije realna konfiguracija? Kolika je usporedna cijena u konačnici nakon dužeg korištenja? Sličnu usporedbu je *Google* napravio 2016. godine naglasivši od 22 provedena upita, svega jedan u kojem nije bio bolji od *Redshifta* i time se prikazavši kao dominantniji. U krajnjoj liniji ne treba vjerovati samim korporacijama oko njihovog proizvoda u potpunosti, već korisnicima.

U listopadu 2016. godine *Periscope data* je usporedio *Redshift*, *Snowflake* i *BigQuery* koristeći upit koji je spajao jedan bilijun redova tablice u manju tablicu, izvršavajući taj proces nekoliko sati. Po njihovom istraživanju *Redshift* i *BigQuery* su bili u prosjeku jednake brzine dok je *Snowflake* bio dva puta sporiji. Ključne razlike su bile:

- provodili su isti upit nekoliko puta, eliminirajući *Redshiftovo* sporo vrijeme kompajliranja,
- upiti su im bili daleko jednostavniji od onih od *TPC-DS* upita.

Problem kod ispitivanja s jednostavnim upitima je taj što će svako skladište proći dosta dobro u tim testiranjima. Brzina pri tim upitima nije važna, već je ona po pitanju upita koji traju satima i satima uz mogućnost pojave greške pri samome kraju (Fraser, O'Connor, 2018).

9.3. Usporedba na dubljem sloju

Do sad smo se fokusirali na brzinu i cijenu, dvije najbitnije točke u poslovanju, ostavljajući po strani do sada navedene karakteristike samih sustava i prikaz o tome ispunjavaju li potrebna očekivanja, te do koje mjere. Kroz rad smo spomenuli i naglasili neke od funkcionalnosti skladišta podataka, zato ćemo ih u ovome poglavlju tablično prikazati kako bi stvorili krajnji i finalni pregled svih sustava navodeći uz to generalne probleme ili pogodnosti vezane uz sloj koji obrađujemo.

Karakteristike [37]	Snowflake	Amazon Redshift	Google BigQuery
Hibridna arhitektura	Balansira boreći se između postavljanja i razvoja tradicionalne arhitekture i arhitekture bazirane na oblaku (za korisnike koji imaju hibridnu okolinu; podržava usluge tradicionalne prirode i usluge u oblaku).	Balansira boreći se između postavljanja i razvoja tradicionalne arhitekture i arhitekture bazirane na oblaku (za korisnike koji imaju hibridnu okolinu; podržava usluge tradicionalne prirode i usluge u oblaku).	Skladište podataka u oblaku.
Tradicionalne usluge/ usluge u oblaku	Usluge u oblaku.	Usluge u oblaku.	Usluge u oblaku.
Pohrana i računalni resursi	Odvaja skladištenje i procesiranje podataka od same konzumacije podataka.	Ne odvaja skladište od računalnih resursa.	Potpuno odvaja skladište od računalnih resursa.
Skalabilnost	Odlično odvaja skladištenje podataka, te rukuje s računalnim resursima i metapodatcima. Trilijuni redova mogu biti razdijeljeni paralelno od više korisnika. Skladište i računalni resursi mogu biti slobodno skalirani odvojeno u bilo kojem trenutku, te će se paralelno tome metapodatci automatski skalirati.	Postavke lokalne konfiguracije; korisnici ne mogu skalirati resurse slobodno. Izmjena bilo kakvog tipa zahtjeva rekonfiguraciju klastera, što može trajati do nekoliko sati dok se podatci redistribuiraju.	Potpuna elastičnost. Automatska i brza provizija po pitanju većih računalnih resursa radi obrade većih setova podataka.
Sigurnost	Sadrži zadovoljavajuću razinu zaštite, te ispunjava sve potrebne industrijske standarde.	Velik fokus na sigurnosti. Uz sigurnost baze podataka, nude se i ostale sigurnosne funkcionalnosti poput: • akreditiva, • SSL veza, • enkripcije učitanih podataka.	Velik fokus na sigurnosti. Svi podatci su enkriptirani prilikom mirovanja i prilikom prijenosa.

[37] Karakteristike koje sadrži moderno skladište podataka u oblaku

Karakteristike [37]	**Snowflake**	**Amazon Redshift**	**Google BigQuery**
Dinamična pohrana u predmemoriji	Svi upiti su pohranjeni 24 sata (90 dana za enterprise razinu kupca) ili dok se pohranjeni podatci ne izmijene.	Pohranjeni podatci se vraćaju momentalno umjesto da se upit ponovno pokrene kada se podatci nisu izmijenili.	Ako se podatci nisu izmijenili, svi upiti su pohranjeni 24 sata.
SQL jezik	Snowflake SQL.	Amazon Redshift SQL.	BigQuery Standard SQL. BigQuery legacy SQL.
SQL sposobnosti	Podržava JSON, XML, Avro i Parquet koristeći poseban tip podatka. Standardna i nova SQL podrška: • višetablični INSERT,MERGE i multi-merge, • jednokratne tablice, • lateralni pregled, • statističke agregatne funkcionalnosti, • skalarne i tabularne funkcije definirane od strane korisnika itd.	Korisnici koji traže klasičan SQL pristup i sposobnosti mogu ostati razočarani shvaćajući kako ne mogu izvršiti navedene radnje: • particioniranje tablica, • ograničenja (strani ključ, primarni ključevi itd.), • indeksiranje, • konsturkcija redova, • pohranjivanje procedura i okidača, • pretraživanje cijeloga teksta.	ANSI 2011 kompatabilnost s ugnježđenim i repetitivnim tipovima podataka. Mjesta za napredak po pitanju kompleksnih JOIN naredbi. Podrška za JSON i XML tip podataka.
Održavanje	Mala količina održavanja.	Kompleksnost integracije. Limitacije: • zahtjeva periodičku analizu tablica.	Mala količina održavanja. Limitacije: • nema indeksa, • nema kolumnarnih restrikcija, • nema mogućnosti za poboljšanjem performansi.
Rukovođenje	Nikakva količina rukovođenja od strane korisnika.	Kompleksan za rukovoditi. Korisnici mogu provesti poprilično količinu vremena radi održavanja ili ažuriranja.	Potpuno održavana usluga od strane pružatelja usluge bez potrebe za korisničkim održavanjem. Backend konfiguracijom i optimizacijom upravlja Google.

Karakteristike [37]	Snowflake	Amazon Redshift	Google BigQuery
Cijena	Plaća se s obzirom na potrošnju. Osigurava vrlo jeftino skladište i više računalnih resursa po dolaru, ne dižući pretjerano cijenu.	Globalno poznat po niskoj cijeni, plati-kako-ideš model, s popustom na dugoročni ugovor.	Nudi plati-kako-ideš model gdje korisnik plaća unos podataka u *BigQuery* i zatim svaki upit zasebno kako se provodi. Fiksirano plaćanje je opcija rezervirana za veće korisnike.

Tablica 4: Sveukupna usporedba svih triju sustava na dubljem sloju[38]

Sva tri navedena skladišta imaju odlične performanse, zato se ne trebamo pretjerano čuditi kada vidimo da su podosta slična. Način za stvaranje brzog kolumnarnog skladišta je bio poznat od kad je objavljen *C-Store* članak 2005. godine što omogućava svim skladištima korištenje jednakih trikova za postizanje prezentiranih performansi. Jedine razlike između samih sustava čine odluke o njihovom internom dizajnu. Neka skladišta stavljaju naglasak na lakšoj mogućnosti optimizacije, druga lakoću korištenja. Kako bi se odabralo najbolje skladište za određeno poduzeće, potrebno je izvršiti više testiranja na odabranim skladištima, te zaključiti koje najbolje odgovara radnicima, strukturi upita i vrsti podataka koju poduzeće koristi.

[38] Preuzeto s https://medium.com/@richiebachala/snowflake-redshift-bigquery-b84d2cb60168#482c (5.6.2020)

10. Zaključak

Rad se osim na tri navedena sustava za skladištenje podataka ujedno fokusirao i na opći pregled i prezentaciju cijele logike skladištenja podataka. U samome uvodu govorimo o sustavima za podršku pri odlučivanju, bazičnoj ideji iz koje su proizašla skladišta za pohranu podataka diferencirajući se s vremenom od samih baza podataka. Nakon toga smo spomenuli podatkovne centre radi proširivanja pogleda i upoznavanja čitatelja s generalnim konceptima vezanim uz sustave za podršku pri odlučivanju. Arhitektura samih skladišta, točnije njena evolucija, je bitno poglavlje za shvaćanje, na dubljem sloju, koliko sama struktura skladišta dugoročno utječe na performanse sustava, bez obzira na kvalitetu definiranih upita i naknadna održavanja. Višedimenzionalnost je po mom osobnom mišljenju jedna od najistaknutijih i najbitnijih komponenti navedenih sustava, predstavljajući time sasvim novi način razmišljanja i pohrane podataka koji je rezultirao, zajedno s drugim komponentama, razvijanjem skladišta kakva danas poznajemo; ujedno pri tome naglašavajući funkcionalnosti poput restrikcije i agregacije, elemente na koje se još uvijek stavlja naglasak. Poglavlje o pristupu skladištima podataka govori pretežito o načinima izvlačenja i prezentacije tih podataka s obzirom na potrebe poslovanja u danom trenutku. Kriteriji za odabir skladišta podataka se temelje na modernome pregledu i poslovnoj potrebi niza korporacija pokušavajući što jednostavnije i sveobuhvatnije predočiti najbitnije karakteristike i norme koje skladišta moraju imati kako bi dugoročno bila iskoristiva, bez potrebe za konstantnom optimizacijom od strane korisnika.

Na kraju smo dali i uvid u provedeno istraživanje; dio rada koji se fokusirao na što kraće i što jasnije predstavljanje navedenih triju skladišta. *Snowflake*, *Amazon Redshift* i *Google BigQuery* su bili odabrani sustavi s obzirom da su jedni od najpopularnijih za korištenje, te se oko njihovog valjanog rangiranja raspravlja već duži niz godina. Prikazavši kroz uvod, arhitekturu, primjere korištenja sustava i primjere korištenja na globalnoj razini većinu karakteristika i funkcionalnosti navedenih skladišta, valjalo je donijeti zaključak o samome ishodu te dugogodišnje borbe, koristeći pritom provedena istraživanja, uspoređujući

bitne komponente, testirajući potrošnju i računajući sveukupnu cijenu svakog sustava zasebno.

Dakako, kao što sam već naveo u radu, nemoguće je s obzirom na provedena testiranja od jednog korisnika, nad svega nekoliko upita koji se mogu tretirati bazičnima, zaključiti kvalitetu svakog skladišta. Zbog toga se dio istraživačkog dijela bavio samom prezentacijom skladišta osvrnuvši se tek naknadno na istraživanja koja su provele kompanije i poduzeća koja imaju valjanu opremu i mogućnost postavljanja restrikcija kako bi provele što točnije istraživanje.

Na samome kraju zaključak se svodi na osobni odabir s obzirom na potrebe poduzeća. Koliko je stručno osoblje koje će se koristiti sustavom? Ima li poduzeće dovoljan broj ljudi koji može održavati skladište podataka? Stavlja li se naglasak više na cijenu? Gleda li se dugoročno ili kratkoročno? Koju količinu podataka posjeduje? Posluju li kroz godinu kontinuirano ili ima velikih razlika iz mjeseca u mjesec? Koliku je ekspanziju poduzeća realno očekivati? Postavlja se i niz dodatnih pitanja za koja treba pronaći odgovor. Iz tog razloga, bez obzira na popularnost, poslovne prilike i raširenost određenih skladišta podataka, treba tražiti sustav koji se najviše i najlakše prilagođava radnoj okolini i postavljenim temeljima poduzeća. Ne postoji bolje ili lošije skladište podataka, samo bolji ili lošiji odabir.

11. Literatura

1. Alibaba Cloud. *7 Factors to Consider When Selecting a Cloud Data Warehouse*. 2019. URL: https://www.alibabacloud.com/blog/7-factors-to-consider-when-selecting-a-cloud-data-warehouse_594740 (22.6.2020.).
2. Alon, S.; Attunity 2018 Analyst &Investor day. 2018. URL: https://www.sec.gov/Archives/edgar/data/893821/000117891318002529/exhibit_99-1.htm (3.5.2020.).
3. Amazon Web Services. Amazon Redshift features. (bez.dat). URL:https://aws.amazon.com/redshift/features/ (20.5.2020.).
4. Bock Corp. Introduction to Snowflake, the modern data warehouse built for cloud. 2018. URL: https://www.youtube.com/watch?v=e3acougGe2A&list=PLC0beSIViTVYywdE5ugoRHy8BIfR_SF8t&index=2&t=2232s (4.5.2020.).
5. Data Warehouse Guide. A Deep Dive Into Google BigQuery Architecture. (bez. dat.). URL: https://panoply.io/data-warehouse-guide/bigquery-architecture/ (25.5.2020).
6. Google Cloud Platform. *Data & Analytics*. 2016. URL: https://www.youtube.com/watch?v=LTVFg6YOjWo (25.5.2020.).
7. Fraser, G; O'Connor, E. *Cloud Data Warehouse Benchmark: Redshift, Snowflake, Azure, Presto and BigQuery*. 2018. URL: https://fivetran.com/blog/warehouse-benchmark (29.5.2020).
8. Golfarelli & Rizzi. *Data Warehouse Design: Modern Principles and Methodologies*. 2009. URL: https://www.academia.edu/6144550/CompRef8_Data_Warehouse_Design_Modern_Principles_and_Methodologies_Golfarelli_and_Rizzi_039-1_1_Introduction_to_Data_Warehousing (15.3.2020.).
9. Google Cloud. *BigQuery Documentation*. (bez.dat). URL: https://cloud.google.com/bigquery/docs (25.5.2020.).

10. Google Cloud Blog. *BigQuery under the hood*. (bez.dat.). URL: https://cloud.google.com/blog/products/gcp/bigquery-under-the-hood (26.5.2020.).
11. Kraynak, J.; Baum, D. *Cloud Data Warehousing for dummies (2nd Edition)*. 2017. URL: https://www.snowflake.com/resource/cloud-data-warehousing-dummies/ (20.3.2020.).
12. Leven, Y.; *Criteria for Selecting a Data Warehouse Platform*. 2017. URL: https://dzone.com/articles/criteria-for-selecting-a-data-warehouse-platform (21.6.2020.).
13. Ogilvy Hamish. *BigQuery, a serverless data warehouse*. 2018. URL: https://medium.com/sajari/bigquery-a-serverless-data-warehouse-9a4d623f3f43 (27.5.2020.).
14. Segment. *How to choose the right data warehouse*. URL: https://segment.com/academy/choosing-stack/how-to-choose-the-right-data-warehouse/ (22.6.2020.).
15. Snowflake. *Snowflake Documentation*. (bez. dat.). URL: https://docs.snowflake.com/en/ (1.5.2020)
16. Stitch. *6 Redshift features that changed the data warehouse game*. (bez.dat.). URL: https://www.stitchdata.com/resources/redshift/ (20.5.2020.).
17. Stonebraker, M.[et al.]. *C-Store: A Column-oriented DBMS*. 2005. URL: http://people.brandeis.edu/~nga/papers/VLDB05.pdf (19.5.2020.).

Sažetak

U ovome radu govoriti ćemo o skladištenju podataka, te o razlozima zbog čega se i na koji način koriste ti kompleksni sustavi. Informacije i podatci su, u današnje vrijeme, neizmjerno vrijedni u radu svakog poduzeća bez obzira na njegovu veličinu, te iz tog razloga moraju biti pravilno pohranjeni i lako dostupni. Najveći problem koji se pritom pojavljuje je dostupnost prevelike količine podataka, odnosno nisu svi podatci, koji su pohranjeni i prezentirani jednom poduzeću, ujedno i relevantni podatci za bolje i produktivnije poslovanje tog poduzeća. Akumulacijom velike količine elektroničkih podataka u posljednjih par godina se pojavila potreba da se ti podatci, osim da se adekvatno pohrane, ujedno i koriste za postizanje ciljeva koji nadilaze rutinske zadatke, a sustavi za skladištenje podataka ispunjavaju upravo te potrebe. Pristupačnost sustava, integracija podataka, informacijska konciznost, višedimenzionalan prikaz, točnost i cjelovitost pohranjenih podataka su samo jedni od atributa sustava za skladištenje podataka, te se iz tih razloga nećemo pretežito baviti pitanjem „zašto koristiti navedene sustave?“, već koje sustave koristiti s obzirom na potrebe poslovanja određenog poduzeća.

Ključne riječi: *skladištenje podataka, informacija, podatci, pohrana, pristupačnost, integracija, višedimenzionalnost*

Data warehouse

Abstract

In this paper, we will talk about data warehouse systems and we will break down why and how are these complex systems used. Information and data are nowadays very valuable in the work of every company, regardless of its size, and they must be properly stored and easily accessible. The biggest problem that arises is the availability of great amounts of data, but not all of the data ,that is stored and presented to a company, is relevant data to create a better and more productive business for those companies. With the accumulation of large amounts of electronic data in the last few years, there has been a need not only to save that data, but to store it adequately also so there could be goals achieved that go beyond daily routine tasks and data warehouse systems fulfill precisely those needs.

System accessibility, data integration, information conciseness, multidimensional representation, accuracy and integrity of stored data are just one of the few attributes of a data warehouse system, and for this reason we will not turn into addressing the "why should they be used?" question, but what systems to use considering the needs of the business or company.

Key words: *data warehousing, information, data, storage, accessibility, integration, multidimensionality*