Distribuirane baze podataka

Rukavina, Barbara

Sveučilište Jurja Dobrile u Puli
Fakultet informatike

BARBARA RUKAVINA

DISTRIBUIRANE BAZE PODATAKA

Završni rad

Pula, svibanj , 2019. godine

Sveučilište Jurja Dobrile u Puli
Fakultet informatike

BARBARA RUKAVINA

DISTRIBUIRANE BAZE PODATAKA

Završni rad

JMBAG: 2798-E, izvanredna studentica
Studijski smjer: Informatika

Predmet: Baze podataka I

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske i komunikacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Mentor: doc. dr. sc. Tihomir Orehovački

Pula, svibanj , 2019. godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisana _____Barbara Rukavina_____, kandidat za prvostupnika _____informatike_____ izjavljujem da je ovaj Završni rad rezultat isključivo mojeg vlastitog rada, da se temelji na mojim istraživanjima, te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno, da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

____Barbara Rukavina____

U Puli, svibanj , 2019. godine

IZJAVA

o korištenju autorskog djela

Ja, _____Barbara Rukavina_____ dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom: _____Distribuirane baze podataka_____ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli, te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu sa Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _svibanj_, _2019._ godine

Potpis

____Barbara Rukavina___

SADRŽAJ

UVOD

Tehnologija baza podataka odvela nas je iz jednostavne obrade podataka u kojoj svaka aplikacija definira i održava vlastite podatke, do one u kojoj se podaci definiraju i upravljaju centralno. U novije vrijeme vidjeli smo brz razvoj mrežnih i podatkovnih komunikacijskih tehnologija, sastavljenih od strane interneta, mobilnih i bežičnih uređaja i računalnih mreža. Sada kombinacijom tih dviju tehnologija, distribuirana tehnologija baze podataka može promijeniti način rada od centraliziranog do decentraliziranog. Ova kombinirana tehnologija je jedan od glavnih kretanja u području baze podataka (Soo, 2018). Glavna motivacija razvoja sustava baza podataka je želja za integracijom operativnih podataka organizacije i osiguranja kontroliranog pristupa podacima. Razvoj računalnih mreža pomiče decentralizirani način rada. Taj decentralizirani pristup odražava organizacijsku strukturu mnogih tvrtki koje se logički distribuiraju gdje svaka jedinica odražava vlastite operativne podatke (Date, 2003). Dijeljenje podataka i učinkovitost pristupa podacima trebalo bi poboljšati razvojem distribuiranog sustava baze podataka koji odražava tu organizacijsku strukturu, čini dostupnim podacima u svim jedinicama i pohranjuje podatke blizu mjesta gdje se najčešće koristi. Centraliziranu bazu je lakše održavati, ali ako dođe do nekog zastoja cijeli sustav pada i gubimo sve podatke. Distribuirane baze podataka su vrlo korisne u slučajevima ako dođe do nekih prirodnih katastrofa. Tada se, ukoliko imamo centraliziranu bazu podataka, svi podaci pohranjeni u njoj brišu. Danas neće doći do toga jer se sustav distribuiranih baza štiti s replikacijom datoteka i dijeljenjem podataka na više geografski udaljenih lokacija. Korištenjem distribuiranih baza podataka znatno se smanjuje trošak implementacije, komunikacije i održavanja baze. Takve baze imaju zahtjevan dizajn i poslužitelj, te se teško oporavljaju ukoliko dođe do pada sustava ili neke velike greške. Jedna od najvećih nedostataka korištenja ovakvog načina rada s bazama je sigurnost. Podaci koji putuju preko komunikacijskih krugova posebno su ranjivi. Sigurnost podataka se provodi na način da se koristi šifriranje, kontrola pristupa, te upravljanje mrežom kako bi se zaštitila i oprema za prijenos podataka. Postoje tri tipa distribuiranih sustava, a to su: homogeni sustavi, heterogeni sustavi i klijent/poslužitelj arhitektura baza podataka. Najpoznatija distribuirana baza podataka je „Spanner“, baza podataka osmišljena i implementirana u Google-u .

Prvo poglavlje govori o bazama podataka kako bi se mogle razjasniti distribuirane baze podataka u drugom poglavlju. U trećem se poglavlju nalazi usporedba centraliziranih i distribuiranih baza podataka. O sustavima upravljanja bazama podataka bavi se četvrto poglavlje. Peto i šesto poglavlje prikazuje tipove i arhitekturu distribuiranih baza podataka. Sedmo poglavlje odnosi se na implementaciju distribuirane baze podataka odnosno fragmentaciju, replikaciju i alokaciju podataka. Osmo i deveto poglavlje se bazira na prijenosu podataka i transakcija. Nakon toga slijedi sigurnost distribuiranih baza podataka te zaključak.

1. BAZE PODATAKA

Tehnologija baze podataka se temelji na tome da aplikacije ne stvaraju vlastite datoteke na disku, nego da koriste zajedničku i objedinjenu kolekciju podataka. Podacima na disku aplikacija nije u mogućnosti pristupiti, te umjesto toga posreduje, služeći se uslugama softvera koji je zadužen za održavanje te kolekcije. Takva kolekcija podataka se naziva baza podataka, a softver koji služi kao posrednik između podataka i aplikacija naziva se sustav za upravljanje bazom podatka. Baza podataka je skup međusobno povezanih podataka, pohranjenih u vanjskoj memoriji računala. Podaci su istovremeno dostupni raznim korisnicima i aplikacijskim programima. Ubacivanje, promjena, brisanje i čitanje podataka obavlja se posredstvom posebnog softvera, sustava za upravljanje bazom podataka. Korisnici i aplikacije pritom ne moraju poznavati detalje fizičkog prikaza podataka, već se referenciraju na neku idealiziranu logičku strukturu baze. Model podataka je skup pravila koja određuju kako sve može izgledati logička struktura baze podataka. Model čini osnovu za oblikovanje i implementiranje baze. Podaci u bazi moraju biti logički organizirani u skladu s onim modelom kojeg podržava odabrani DBMS (Manger, 2012).

Modeli koji postoje su:

- Relacijski model - zasnovan je na matematičkom pojmu relacije. U tablicama koje sadrže samo retke i stupce prikazuju se podaci i veze među podacima. Relacijski model prikazan je na slici 1.
- Mrežni model – baza je predočena mrežom koja se sastoji od čvorova i usmjerenih lukova. Čvorovi predstavljaju tipove zapisa, a lukovi definiraju veze među tipovima zapisa.
- Hijerarhijski model – specijalni slučaj mrežnog. Baza je predočena jednim stablom (hijerarhijom) ili skupom stabala. Čvorovi su tipovi zapisa, a odnos "nadređeni-podređeni" izražava hijerarhijske veze među tipovima zapisa.
- Objektni model – inspiriran je objektno-orijentiranim jezicima. Baza je predočena kao skup trajno pohranjenih objekata koji se sastoje od svojih internih "atributa" (podataka) i "metoda" (operacija) za rukovanje tim podacima. Svaki objekt pripada nekoj klasi. Između klasa se uspostavljaju veze nasljeđivanja, agregacije i druge vrste veza.

Hijerarhijski i mrežni modeli nisu više u upotrebi. Najviše se koristi relacijski model.

Prijelaz s relacijskog na objektni se još nije dogodio, pa svi poznati DBMS-i podržavaju isključivo relacijski model. Tehnologija baza podataka nastoji ispuniti sljedeće uvjete:

- Fizička nezavisnost podataka
- Logička nezavisnost podataka
- Fleksibilnost pristupa podacima
- Istovremeni pristup do podataka
- Čuvanje integriteta
- Mogućnost oporavka nakon kvara
- Zaštita od neovlaštene uporabe
- Zadovoljavajuća brzina pristupaju
- Mogućnost podešavanja i kontrole

Arhitektura baze podataka sastoji se od 3 sloja, a to su fizička razina, globalna logička razina i lokalna logička razina (Manger, 2012).

Prema Date-u (2003) prednosti uporabe baza podataka naprama ručnog vođenja papira su sljedeće:

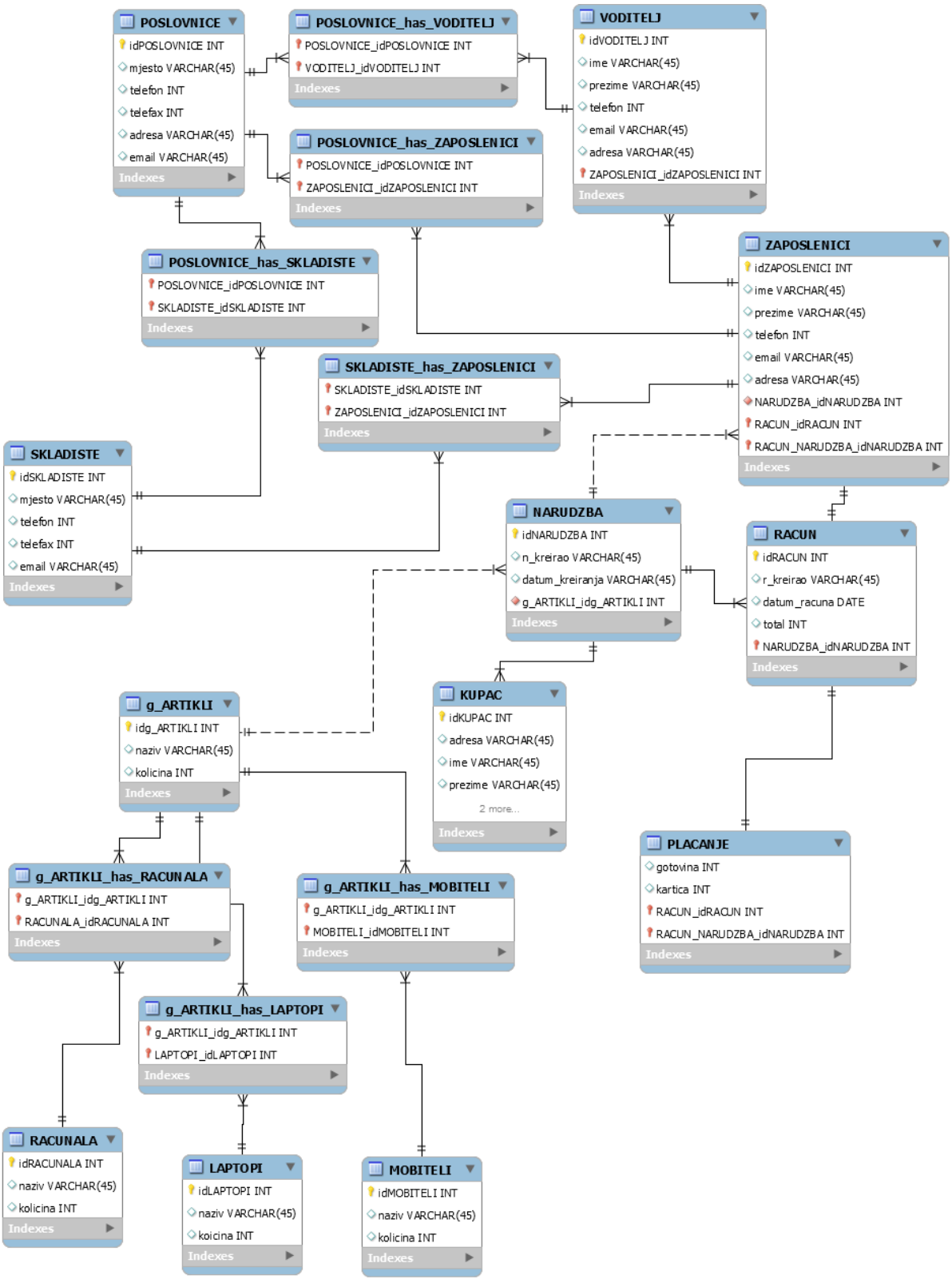
- Kompaktnost – kod baza podataka nema potrebe za korištenjem papira.
- Brzina – računala mogu primiti i ažurirati podatke puno brže nego što je to u mogućnosti čovjek.
- Manje posla – u konačnici je potrebno puno manje rada na održavanju nego li je to kod ručnog upisivanja.
- Optjecaj – točne, ažurne informacije na zahtjev.
- Sigurnost – podaci su bolje zaštićeni protiv nenamjernog gubitka i neovlaštenog korištenja informacija.

Prednosti su izraženije kod multi-user okruženja.

API (engl. Application programming interface) je sučelje u kojem korisnik i program surađuju s bazom podataka. Prema Richardson-u (2015) sučelje ima dvije dimenzije:

- U procesu i izvan procesa – ako se baza podataka pokreće u istom procesu kao i aplikacija korisnika, obično postoji biblioteka funkcijskih poziva koja se izravno poziva na metode u bazi podatka. To rezultira najnižom mogućom memorijom. Pritom se smanjuju mogućnosti, jer to znači da samo jedna korisnička aplikacija može pristupiti podacima istodobno.

- SQL i NoSQL – SQL je deklarativni jezik koji je originalno dizajniran kao mehanizam za pojednostavljivanje pohrane i pronalaženja relacijskih podataka. Najveća inovacija većine NoSQL baza podataka je jednostavno postizanje bržih operacija uklanjanjem transakcija i relacijskih tablica. Mnoge od tih baza podataka počele su podržavati SQL kao jezik API-ja. Ne koriste njegove relacijske značajke, ali su mu zato specifikacije proširene. Ovih dana, NoSQL se može preciznije zvati NoRelational, ali NoSQL je prlično blizu.



Slika 1. Relacijski model baze podataka[1]

[1] Relacijski model je napravljen u programu MySQL Workbench.

Relacijski model prikazan na slici 1 je napravljen kao trgovina za prodaju računala, laptopa i mobitela. EER dijagram[2] se sastoji od sljedećih tablica: poslovnice, voditelj, zaposlenici, skladište, g_artikli, računala, laptopi, mobiteli, kupac, narudžba, račun , plaćanje i veza među njima.

[2] EER dijagram je model visoke razine ili konceptualni model koji uključuje ekstenzije na originalni model entiteta i veza, koji se koristi za dizajn baza podataka.

2. DISTRIBUIRANE BAZE PODATAKA

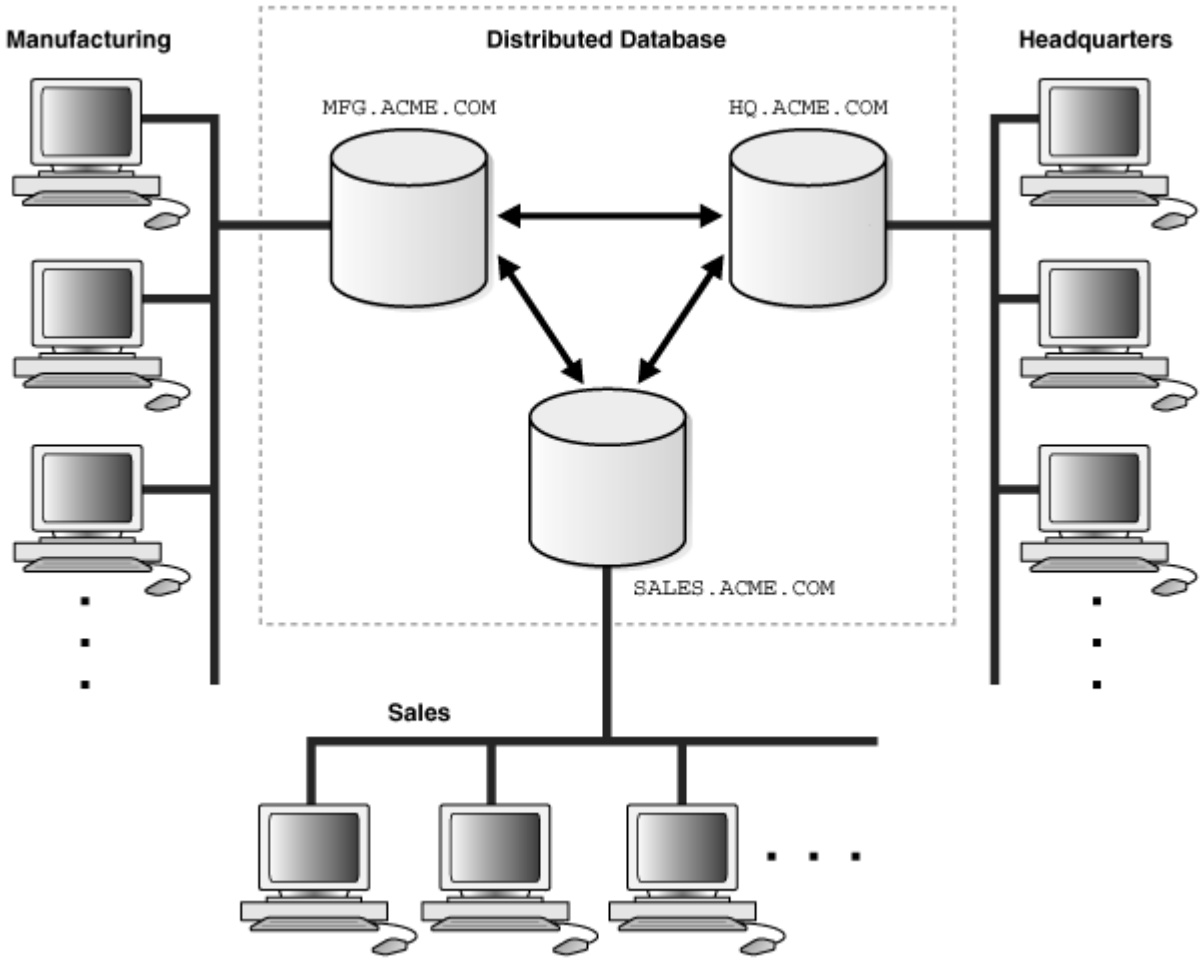
Distribuirana baza podataka je baza podataka podijeljena na više računala, koja se nalaze na različitim lokacijama. Ta računala su povezana komunikacijskom mrežom. Svako računalo ima vlastiti sustav za upravljanje bazama podataka ili DBMS. Distribuirane baze podataka se korisniku prikazuju kao jedna baza podataka koja je zapravo set bazi pohranjen na mnogobrojnim računalima. Prikazana je na slici 2. Podacima se na različitim računalima može pristupiti istovremeno i upravljati s njima preko mreže. Svaki je poslužitelj baze u distribuiranom okruženju upravljan od strane lokalnog DBMS-a i sa svim kooperantima, kako bi održali konzistenciju globalne baze. Poslužitelj je softver koji upravlja bazom podataka, a klijent je aplikacija koja zahtjeva informacije od poslužitelja.

Svako računalo u sustavu je čvor. Čvor u distribuiranom sustavu može biti klijent, poslužitelj ili oboje. Oracle podržava heterogeno, klijent/poslužitelj okruženje u kojemu klijent i poslužitelj koriste različite skupove znakova. Skup znakova korišten od strane klijenta definiran je vrijednošću NLS_LANG[3] parametra. Klijent se može spojiti direktno ili indirektno na poslužitelj. Svaki poslužitelj koji sudjeluje u distribuiranoj bazi podataka upravljan je samostalno, od strane drugih baza, kao da je svaka baza centralizirana. To se zove autonomija lokacije ili *site autonomy.* Iako sve baze mogu raditi zajedno, one se vode kao zasebni dio te se posebno upravlja sa svakom. Neke od karakteristika autonomije lokacije, odnosno onoga što lokacija na kojoj se distribuirana baza nalazi su sljedeće (Oracle, 2018):

- Čvorovi sustava mogu odraziti logičku organizaciju tvrtke ili organizacije
- Lokalni podaci su kontrolirani od strane administratora baze podataka. Domena odgovornosti svakog administratora baze je sve manja i izvodljivija.
- Manji kvarovi neće poremetiti ostale čvorove distribuirane baze. Globalna baza je samo djelomično dostupna dok je god jedna baza i/ili mreža slobodna.
- Oporavak od pada sustava se često izvodi na jednom čvoru.
- Rječnik podataka postoji za svaku lokalnu bazu, što znači, da nije potreban globalni katalog da bi pristupili podacima.

[3] NLS_LANG je parametar koji pruža podršku za oblikovanje i analizu datuma, vremena, brojeva i valuta. Postavlja jezik i teritorij koji koristi aplikacija klijenta i poslužitelj baze podataka.

- Čvorovi se nadograđuju samostalno.



Slika 2. Distribuirana baza podataka (Oracle, 2018)

Shematskom prikazu objekta možemo pristupiti preko svih čvorova koji formiraju distribuiranu bazu. Samo kod centraliziranih arhitektura moraju se postaviti nedvosmislena imena kako bi se razlikovale reference objekata unutar lokalne baze. Kod distribuiranih se sustava mora koristiti shema imenovanja, koja osigurava da će objekt unutar baze biti identificiran i referenciran. Kako bi se riješio problem referenciranja objekta (proces koji se zove engl. *Name resolution*) unutar baze, DBMS obično formira imena objekta koristeći se hijerarhijskim pristupom. DDBMS jednostavno proširuje hijerarhijski model za imenovanje tako da postavlja jedinstvena imena baza podataka unutar mreže. Kao rezultat, garantirano je da će objekt *global object name* biti jedinstven unutar distribuirane baze i da reference objekta *global object name* mogu biti riješene među čvorovima sustava. Kako bi se ostvarila veza

između individualnih baza u distribuiranom okruženju, Oracle koristi linkove. Link se definira kao put do udaljene baze podataka. Oni su uglavnom transparentni za korisnika,, jer je njegovo ime isto kao i globalno ime baze na koji pokazuje. Bilo koja aplikacija ili korisnik može se spojiti na lokalnu bazu i pristupiti podacima putem linka. Udaljeni upit je upit koji odabire podatke od jedne ili više udaljenih tablica, koje se nalaze na istom čvoru. Udaljeno ažuriranje je ažuriranje koje modificira podatke u jednoj ili više tablica koje se nalaze na istom čvoru. Distribuirani upit prima informacije od dva ili više čvora. Distribuirano ažuriranje modificira podatke od dva ili više čvora. Takvo ažuriranje je moguće koristeći se programskim jedinicama kao što su procedure ili okidači[4] (engl. Triggers), koji uključuju dva ili više udaljena ažuriranja. Ona dalje pristupaju podacima na različitim čvorovima (Oracle, 2018).

2.1. Prednosti distribuiranih baza podataka

DDBMS odražava promjenjivu strukturu organizacije. Organizacije danas karakterizira decentralizirana struktura, koja se često širi na mnogim zemljopisnim lokacijama (Nemoto i sur., 2015). Poboljšana je podjela i lokalna autonomija, što znači da se zemljopisna distribucija organizacije može odraziti na distribuciju podataka. Korisnici na jednom mjestu mogu pristupiti podacima pohranjenim na drugim web mjestima. Mogu se postaviti blizu korisnika koji obično koriste te podatke. Na taj način korisnici imaju lokalnu kontrolu, te time mogu uspostaviti i provoditi lokalne politike vezane za korištenje tih podatka. Globalni administrator baze podataka odgovoran je za cijeli sustav. Općenito, dio te odgovorosti prenosi se na lokalnu razinu, tako da lokalni administrator baza može upravljati lokalnim DBMS-om. Još jedna od prednosti je poboljšana dostupnost. U centraliziranom DBMS, računalni kvarovi obustavljaju sve operacije koje su bile izvođene. Kod distribuiranog sustava, kvar na jednom mjestu ili kvar na komunikacijskoj mreži ne utječu na cijeli sustav i sustav neomatano nastavlja raditi. Distribuirani sustavi su dizajnirani da i dalje funkcioniraju usprkos takvim kvarovima. Ako je jedan čvor u kvaru, sustav može preusmjeriti zahtjeve takvog čvora na drugu web lokaciju. Budući da se podaci mogu replicirati tako da postoje na više od

[4] Okidači su spremljeni programi koji se automatski izvode nakon nekog događaja, odnosno nakon određenih SQL naredbi (npr. Insert, delete, create…).

jednog web mjesta, kvar čvora ili komunikacijske mreže ne mora nužno učiniti podatke nedostupnima, čime je poboljšana pouzdanost. Pošto se podaci nalaze u blizini mjesta „najveće potražnje“, brzina pristupa bazama podataka može biti bolja od onoga ostvarivog u centraliziranoj,što govori o poboljšanju performansi. U 1960.-ima, računalna snaga je bila izračunata prema kvadratu troškova opreme, što je značilo da bi trostruki trošak osiguravao devet puta veću snagu. To je bilo poznato kao Groschov zakon[5]. Međutim, sada je opće prihvaćeno da mnogo manje košta stvaranje manjih računala s ekvivalentnom snagom jednog velikog računala. Na taj način korporacijski odjeli nabavljaju odvojena računala jeftinije. Također je isplativije dodavati radne stanice u mrežu nego da se ažurira glavni sustav. Druga potencijalna ušteda troškova događa se kada su baze podataka geografski udaljene i aplikacije zahtijevaju pristup distribuiranim podacima. U takvim slučajevima, zbog relativnih troškova prijenosa podataka preko mreže, može biti mnogo ekonomičnije particionirati aplikaciju i izvesti obradu lokalno na svakom mjestu odnosno lokaciji. U distribuiranom okruženju je mnogo lakše upravljati ekspanzijom tj. proširenjem. Nove se web lokacije mogu dodati u mrežu bez utjecaja na rad drugih web lokacija. Ova fleksibilnost omogućuje organizaciji da se razvije relativno lako. Povećanje veličine baze se obično može napraviti dodavanjem procesa i snage pohrani u mreži. U centraliziranom DBMS-u, rast može dovesti do promjene u hradveru i softveru (Connelly i Begg, 2005).

Integracija naslijeđenih sustava jedan je od posebnih primjera koji pokazuje kako su neke organizacije prisiljene oslanjati se na distribuiranu obradu podatka kako bi dopustili njihovim naslijeđenim sustavima da zajedno postoje s modernim sustavima. Istodobno, nijedan paket ne može pružiti sve funkcionalnosti koje organizacija danas zahtjeva. Stoga je važno da organizacije mogu integrirati softverske komponente različitih dobavljača kako bi ispunile svoje specifične zahtjeve.

Postoji niz relativno novijih događaja koji se jako oslanjaju na tehnologiju distribuiranih baza podataka, kao što su: e-Business, suradnički rad koji je podržan na računalu i upravljanje radnim procesom. Mnoga su poduzeća morala reorganizirati svoje poslovanje i koristiti distribuiranu tehnologiju baza podatka kako bi ostala konkurentna(Connelly i Begg, 2005).

[5] Groschov zakon je nazvan po Herbu Grosch koji je 1953. rekao da je „ekonomija kao kvadratni korijen brzine.“

2.2. Nedostaci distribuiranih baza podataka

Glavni nedostaci su (Iacob i Moise, 2015):

- Kompleksnost – Distribuirani DBMS koji skriva distribuiranu prirodu od korisnika osigurava prihvatljivu razinu izvedbe, pouzdanost i dostupnost je složeniji nego centralizirani DBMS. Činjenica da se podaci mogu replicirati dodatno utječe na kompleksnost. Ako softver ne podnosi replikaciju podataka adekvatno, doći će do degradacije u dostupnosti, pouzdanosti i učinkovitosti kada usporedimo s centraliziranim sustavom. Tada prednosti postaju nedostaci.
- Troškovi – Povećana složenost znači da možemo očekivati troškove nabave i održavanja za DDBMS puno veće oda onih u centraliziranom DBMS-u. Nadalje, distribuirani DBMS zahtijeva dodatni hardver za uspostavljanje mreže između web mjesta. Komunkacijski troškovi su nastali prilikom korištenja te mreže, kao i dodatni troškovi rada za upravljanje i održavanje lokalnih DBMS-ova i osnovne mreže.
- Sigurnost – U centraliziranom sustavu pristup podacima može se lako kontrolirati. Međutim, u distribuiranom sustavima, ne samo da pristup repliciranim podacima mora biti kontroliran u više mjesta već i sama mreža mora biti sigurna. U prošlosti se promatraju mreže kao nesiguran komunikacijski medij. Iako je to još uvijek djelomično istinito, postignuti su značajni pomaci u razvoju sigurnosti mreže.
- Kontrola integriteta – odnosi se na valjanost i konzistentnost pohranjenih podataka. Integritet je obično izražen u smislu ograničenja koja su zapravo pravila konzistencije koje baza ne smije kršiti. Provođenje ograničenja integriteta općenito zahtijeva pristup velikoj količini podataka koji definiraju to ograničenje, ali nisu uključeni u aktualno ažuriranje samih operacija. U distribuiranom sustavu troškovi komunikacije i obrade koji su potrebni za provođenje ograničenja integriteta mogu biti prepreka.
- Kompliciran dizajn – osim uobičajenih poteškoća u izradi centralizirane baze podatka, kod dizajna distribuirane baze se mora uzeti u obzir fragmentacija podatka, raspodjela fragmenata na određene lokacije i replikacija podataka.

Connelly i Begg u svojoj knjizi iz 2005. godine navode još kao nedostatak iskustvo te nedostatak standarda. Tada tehnologija još nije bila razvijena kao što je to danas. Danas su za skoro svaki nedostatak dana rješenja raznih znanstvenika, profesora i inženjera diljem svijeta. Primjer takvog rješenja je „Perfopticon“, alat razvijen na odjelu Sveučilišta u Washingtonu (Moritz, Halperin, Howe. i Heer, 2015). To je interaktivni alat za vizualizaciju i za razumijevanje protoka podataka i performansi u distribuiranoj bazi podataka. Perfopticon vizualizira zapise izvršenja upita u koordiniranim vizualizacijama na različitim razinama apstrakcije, kako bi se omogućilo učinkovito profiliranje i ispravljanje pogrešaka.

2.3. Google globalna distribuirana baza podataka „Spanner“

„Spanner“ je skalabilna, globalno distribuirana baza podataka dizajnirana, izgrađena i implementirana u Google-u. Replika se koristi za globalnu dostupnost i geografsku lokaciju. Spanner automatski prebacuje podatke preko strojeva kako se količina podataka ili broj poslužitelja mijenja da bi uravnotežio opterećenje kao odgovor na kvarove. Dizajniran je da bude na milijun strojeva preko stotina podatkovnih centara i bilijuna redaka baze podataka. Aplikacije mogu koristiti Spanner za visoku dostupnost, čak i pri prirodnim katastrofama, na način da replicira svoje podatke unutar ili čak preko kontinenta. Glavni fokus Spanner-a je upravljanje podacima koji se prenose unakrsnim podatkovnim centrima. Utrošeno je mnogo vremena u osmišljavanju i provedbi važnih značajki baze na vrhu infrastrukture distribuiranih sustava. Razvio se u više verzija baze podataka. Podaci su pohranjeni u shematiziranim tablicama. Svaka se verzija automatski ažurira. Aplikacije mogu čitati podatke koji su zastarjeli. Podržava se opća namjena transakcija i pruža SQL upitnik. Neke značajke: aplikacije mogu odrediti ograničenja za kontrolu podatkovnih centara i podaci mogu biti dinamički i transparentno premješteni između podatkovnih centara prema sustavu za ravnotežu korištenja resursa preko centara (Google, 2013).

3. CENTRALIZIRANE vs. DISTRIBUIRANE BAZE PODATAKA

Kod centraliziranih baza podataka s podacima upravlja jedan DBMS (engl. *Database Management System*) postavljen na jednom čvoru. Koriste ju samo korisnici koji su distribuirani na istoj mreži. Glavne prednosti ovakvog tipa baza podataka su određene pomoću dobre integracije podataka koja osigurava konzistenciju podataka i jednostavno upravljanje transakcijama s ACID[6] svojstvima (*Atomicity, Consistency, Isolation, Durability*). Mane centraliziranih baza su: preveliki trošak komunikacije, jako niska pouzdanost i slaba dostupnost. Bilo koja greška ili kvar koji blokira pristup bazi podataka odmah prekida svu aktivnost na mreži.

Distribuirane baze podataka se sastoje od skupa lokalnih baza podatka, geografski smještenih na različitim lokacijama. To su čvorovi umreženih računala koja su logički povezani s funkcionalnim vezama. Globalno se sva računala i baze koje se nalaze na njima gledaju kao jedna baza podataka. U distribuiranim okruženjima dolazimo do novih problema koji nisu bitni u centraliziranim okruženjima. Neki od tih problema su fragmentacija i replikacija podataka. Fragment podataka tvori podskup originalne baze. Replika tvori kopiju cjeline ili dijela baze. Fragmentacija i replikacija se ne mogu kombinirati jer za to postoji alokacija podataka. Veza se može razdijeliti na nekoliko dijelova i može imati mnogobrojne replike svakog fragmenta. Kako bi se DBMS sustav distribuirao mora biti sukladan s 12 pravila koje je predstavio C.J Date 1987. godine. Ta pravila su sljedeća (Iacob i Moise, 2015):

- lokalna autonomija
- neovisnost o centralnoj lokaciji
- kontinuirane operacije
- neovisnost o lokaciji
- neovisnost o fragmentaciji
- neovisnost o replikaciji
- distribuirana obrada upita
- distribuiran menadžment transakcija
- neovisnost o hardveru

[6] ACID je akronim i mnemonički uređaj za učenje i pamćenje četiri primarna atributa koja su osigurana u svakoj transakciji, valentnost, konzistentnost, izolacija i trajnost.

- neovisnost o operacijskom sustavu
- neovisnost o komunikacijskoj infrastrukturi
- neovisnost o DBMS

S perspektive distribuiranih baza podataka, kao efekt decentralizacije, integracije podataka, minimalne redundancije i ACID svojstva trebali bi se opustiti jer je te zahtjeve teško postići, ali je zato povećana dostupnost velika prednost. Odluka o bilo kojem rješenju se može uzeti tek nakon detaljne analize aplikacijskih zahtjeva, veličine baze, karakteristika slobodnih infrastruktura zajedno s evaluacijom globalnih sistemskih performansi. Velika prednost upotrebljavanja distribuiranih baza je u tome što prilikom dijeljenja baze preko mnogobrojnih čvorova dobivamo dodatne ekstenzije memorije i dodatnu obradu resursa. Iako se računalna snaga znatno povećala posljednjih godina, procesiranje velikih podataka može dovesti do sveukupno slabih performansi. Distribuiranje podataka preko mnogobrojnih procesnih centara može dodati velike izvedbene prednosti radi paralelne obrade podataka preko mnogobrojnih čvorova (tzv. Paralelizam), ali održavanje transakcija i ACID svojstva je puno teže postići. Distribuirane baze nude i neke prednosti (Iacob i Moise, 2015):

- Odražava organizacijsku strukturu mnogih organizacija, s obzirom na činjenicu da su mnoge tvrtke distribuirane, odnosno smještene svugdje po svijetu.
- Povećana pouzdanost i dostupnost. Pouzdana je u usporedbi s centraliziranom bazom.
- Lokalna kontrola. Podaci su distribuirani na način da je svaki njegov dio lokalan nekim poslužiteljima. Dio podataka koji je spremljen na poslužitelj pripada njemu.
- Modularni rast (elastičnost). Rast je jednostavniji naspram centraliziranog okruženja. Ne trebamo prekidati ni jedan funkcionalni čvor da dodamo novi. Tako je proširenje cijelog sustava jednostavnije. Uklanjanje čvora na lokaciji isto ne predstavlja problem.
- Komunikacija je jeftinija (manji trošak). Ekonomičnija je. Podaci se distribuiraju na takav način da su dostupni blizu lokacija gdje su potrebni. Takav način dovodi do manjih troškova prilikom komunikacije. Usporedba svih troškova centraliziranih i distribuiranih baza podataka prikazana je na slici 3.
- Brži odgovor. Većina podataka je lokalna i nalazi se u blizini. Zahtjevi se mogu ispuniti brže u odnosu na centralizirane baze podataka.

- Osiguran menadžment distribuiranih podataka. Transparentnost mreže, fragmentacija i replikacija implementirani su da skriju stvarne implementacijske detalje cijelog distribuiranog sustava.
- Robusnost. Sustav i dalje radi ukoliko dođe do kvara. Npr. Replicirana distribuirana baza dalje radi unatoč prekida rada nekog čvora.
- Sukladan s ACID svojstvima. Distribuirane transakcije zahtijevaju valentnost, konzistenciju, izolaciju i pouzdanost.
- Poboljšane su performanse u izvršavanju transakcija.

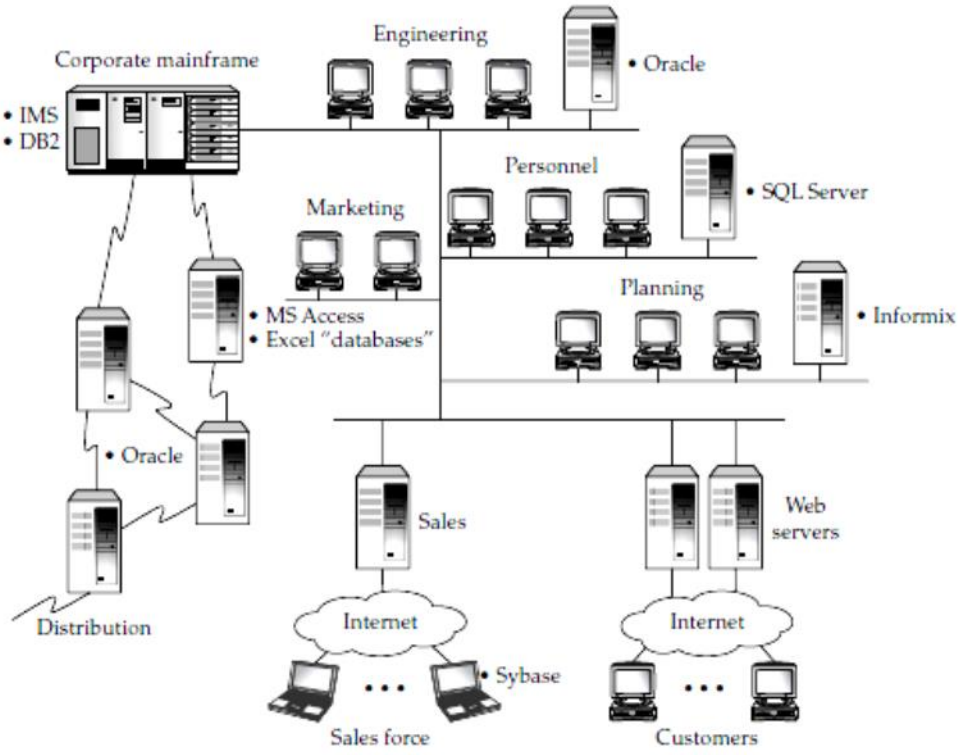
	Centralized	Distributed		
		Fragmented	Complete replication	Partial replication
Storage costs	Small	Small	Large	Medium
Reliability and availability	Small	Small	Large	Medium
Update speed	Medium	Large	Small	Small
Communication costs	Large	Small	Large	Small
Redundancy	Small	Small	Large	Large
Concurrency access to data	Large	Large	Small	Small
Update time	Small	Small	Large	Large
Retrieval time	Large	Small	Small	Small

Slika 3. Troškovi kod centraliziranih i distribuiranih baza podataka (Iacob i Moise, 2015).

4. DATABASE MANAGEMENT SYSTEMS (DBMS)

Sustav za upravljanje bazom podataka (engl. *Database Management System*) ili skraćeno DBMS je poslužitelj (*server*) baze podataka. On oblikuje fizički prikaz baze u skladu s traženom logičkom strukturom. U stanju je podržati razne baze, od kojih svaka može imati svoju logičku strukturu, no u skladu s istim modelom. Brine se za sgurnost podataka, te automatizira administrativne poslove s bazom. DBMS pripada softveru kojeg većina korisnika i organizacija ne razvija samostalno nego ga kupuju s računalom. Korištenje DBMS-a u tvrtkama prikazano je na slici 4. Neki od široko zastupljenih DBMS-a su (Manger, 2012):

- DB2 - proizvod tvrtke IBM.
- Oracle - proizvod tvrtke Oracle koja pokriva sve računale platforme.
- MS SQL Server - proizvod tvrtke Microsoft koji je namijenjen operacijskim sustavima MS Windows.
- MySQL - besplatni proizvod tvrtke MySQL AB, popularan na raznim platformama, prvenstveno kao podrška web aplikacijama.



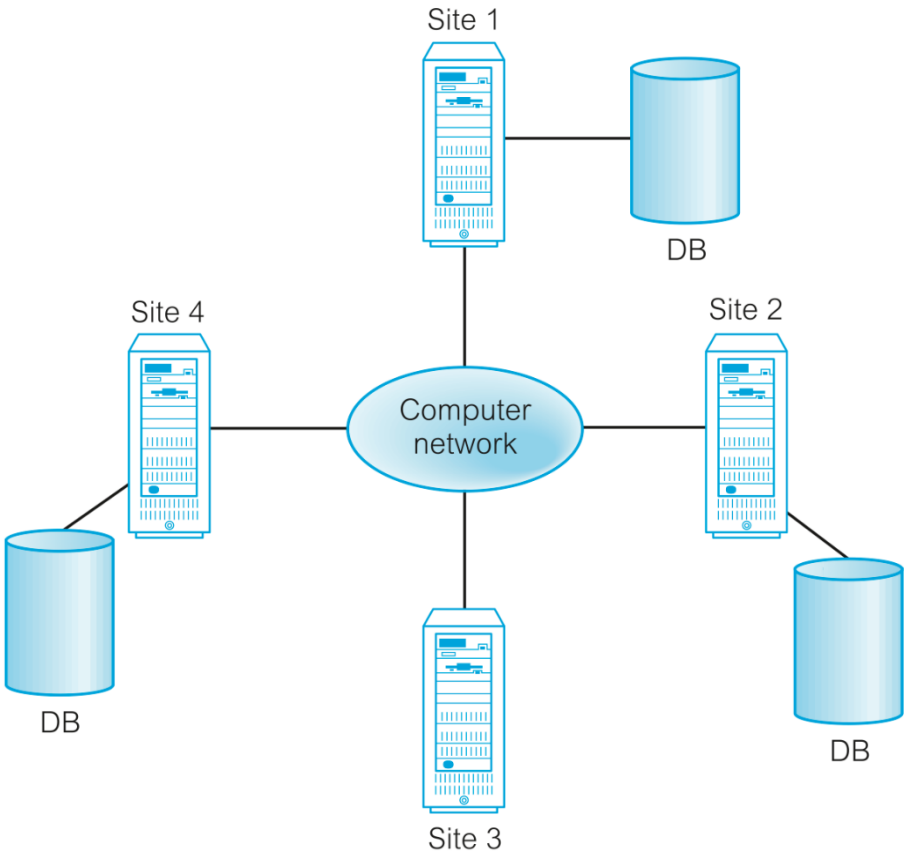
Slika 4. DBMS-a u tvrtkama (Petrini, 2009)

4.1. DDBMS (*Distributed Database Management Systems*)

Distribuirani sustav za upravljanje bazama podtaka (engl. *Distributed database management system*) ili skraćeno DDBMS je centralizirana aplikacija koja upravlja distribuiranom bazom podataka, kao da je sve pohranjeno na istom računalu. DDBMS sinkronizira sve podatke periodično i u nekim slučajevima, gdje više korisnika mora pristupiti istim podacima, osigurava da se sve promjene uvedene na toj lokaciji automatski reflektiraju na podatke na nekoj drugoj lokaciji. Sustav se sastoji od jedne lokalne baze podataka koja je podijeljena na fragmente. Svaki fragment je spremljen na jedno ili više računala pod kontrolom odvojenih DBMS (*Database Management Systems*) s računalima spojenim preko komunikacijske mreže. Komunikacijska mreža je prikazana na slici 5. Svako mjesto je sposobno nezavisno obraditi zahtjeve korisnika koji traže pristup lokalnim podacima. Korisnici pristupaju distribuiranim bazama preko aplikacija koje su klasificirane kao one koje ne zahtijevaju podatke od drugih strana (lokalne aplikacije) i one koje zahtijevaju podatke od drugih čvorova (globalne aplikacije). DDBMS mora imati barem jednu globalnu aplikaciju. Pristup preko lokalnog i distribuiranog poslužitelja prikazan je na slici 5. Karakteristike DDBMS su (Soo, 2018):

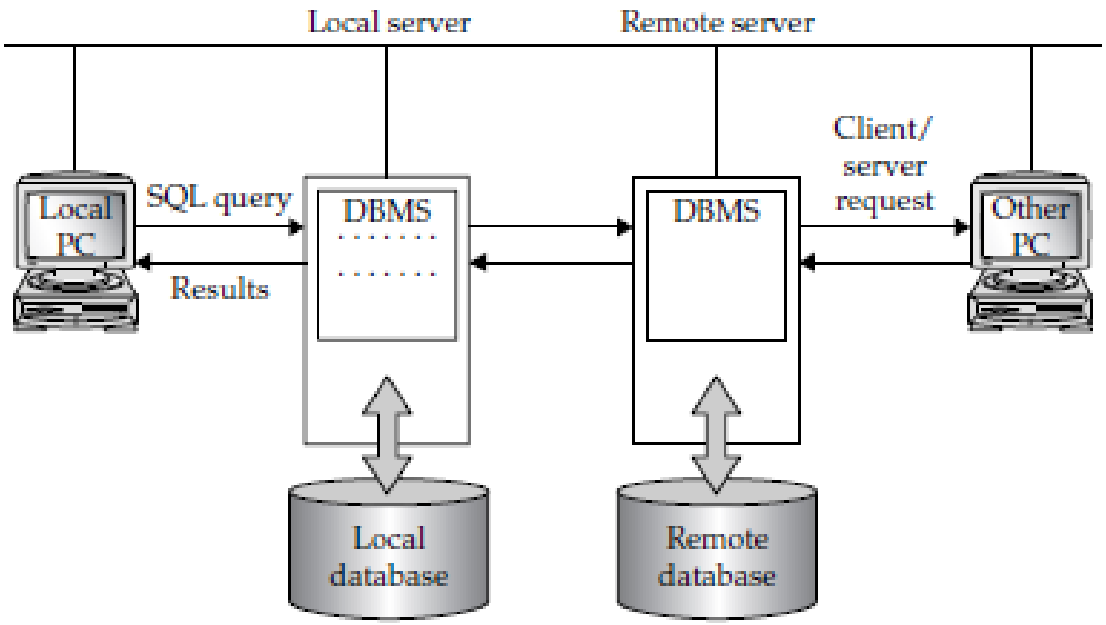
- ima zbirku logički povezanih dijeljenih podataka
- podaci su fragmentirani
- fragmenti se mogu replicirati
- fragmenti su alocirani
- čvorovi su povezani preko komunikacijske mreže
- podaci svakog čvora su pod kontrolom DBMS
- DBMS na svakom mjestu upravlja samostalno
- svaki DBMS sudjeluje u bar jednoj globalnoj aplikaciji

Od sustava se očekuje da je distribucija podataka transparentna, odnosno nevidljiva korisniku. Cilj transparentnosti podataka je da se distribuirani sustav prikazuje kao centralizirani sustav. To se smatra kao temeljni princip distribuiranih sustava. Zahtjevi pružaju značajnu funkcionalnost za krajnjeg korisnika, ali se pritom stvara puno dodatnih problema koji moraju biti obrađeni DDBMS sustavom (Soo, 2018).



Slika 5. Komunikacijska mreža (Soo, 2018)

Distribuirana obrada podataka je naziv koji se koristi kod centralizirane baze podataka kojoj se može pristupiti preko računalne mreže. Distribuirani DBMS je sustav koji je fizički distribuiran na više mjesta u mreži. Ako su podaci centralizirani i ako drugi korisnici pokušavaju njima pristupiti preko mreže, ne smatramo ih distribuiranim DBMS nego distribuiranom obradom podataka (Soo, 2018).



Slika 6. Pristup poslužitelju preko lokalne i distribuirane baze podataka (Petrini, 2009)

4.2. Paralelni DBMS

DBMS koji se proteže preko više procesora i diskova koji su dizajnirani da izvršavaju operacije paralelno, kad je god to moguće, kako bi se poboljšale performanse, nazivaju se paralelni DBMS. Temeljeni su na pretpostavci da jedan sustav procesora više ne može ispuniti rastuće zahtjeve za ekonomičnu skalabilnost, pouzdanost i performanse. Snažna i financijski atraktivna alternativa takvom DBMS je paralelan DBMS, koji je upravljan s više procesora. Povezuju više malih uređaja kako bi postigli istu propusnost kao jedan veliki uređaj, često s više skalabilnosti i pouzdanosti nego jedan DBMS. Kako bi osigurali više procesora s uobičajenim pristupom jednoj bazi podataka, paralelni DBMS mora osigurati menadžment dijeljenja resursa. Koji će se resursi dijeliti i kako će se ti resursi implementirati, direktno utječe na performanse i skalabilnost sustava. Na taj se način provjerava podobnost aplikacija. Tri glavne arhitekture za paralelne DBMS su (Soo, 2018):

- Dijeljena memorija

- Dijeljeni disk
- Dijeljeno ništa

Dijeljena memorija je čvrsto povezana arhitektura u kojoj više procesora unutar jednog sustava dijeli memoriju. Poznato je još kao simetrično višestruka obrada (engl. *Symetric multiprocessing*) ili skraćeno SMP. Ovakav pristup je postao popularan na većem broju platforma. Arhitektura osigurava brzi pristup limitiranom broju procesora, do 64 procesora unutar mreže.

Dijeljeni disk je slabije spojena arhitektura optimizirana za aplikacije koje su centralizirane i traže visoku dostupnost i performanse. Svaki proces može pristupiti diskovima direktno, ali svaki ima svoju privatnu memoriju. Kao i arhitektura dijeljeno ništa ova arhitektura eliminira dijeljenu memoriju. Dijeljeni disk sustavi se ponekad zovu klasteri (engl. *Clusters*)(Soo, 2018).

Dijeljeno ništa je često poznato kao masivno paralelno procesiranje (engl. *Massive parallel processing*) ili skraćeno MPP. To je arhitektura više procesora u kojoj je svaki procesor dio kompletnog sustava sa svojom memorijom i memorijom na disku. Baza podataka je podijeljena između svih diskova u sustavu i njeni su podaci transparentno dostupni korisnicima. Arhitektura je više skalabilna nego li je to dijeljena memorija, i može lako podupirati veliki broj procesora. Performanse su optimalne jedino kad su traženi podaci spremljeni lokalno.

Paralelna tehnologija je tipično korištena za veoma velike baze podataka veličine u terabajtima ili sustavima koji moraju obraditi tisuće transakcija u sekundi. Takvi sustavi trebaju imati pristup velikom obujmu podataka i moraju osigurati pravovremene odgovore zadanim upitima. Paralelni DBMS može koristiti osnovnu arhitekturu kako bi poboljšao performanse kompleksnih upita koristeći se pritom paralelnim skeniranjem, spajanjem i sortiranjem koja dozvoljavaju automatsko dijeljenje podataka (Soo, 2018).

5. TIPOVI DISTRIBUIRANIH BAZA PODATAKA

Distribuirani sustav baza podataka dopušta aplikacijama pristup podacima preko lokalnih i udaljenih baza podataka (Oracle, 2018).

Tipovi distribuiranih baza podataka su:

1. Homogeni sustavi distribuiranih baza podataka
2. Heterogeni sustavi distribuiranih baza podataka
3. Klijent/poslužitelj arhitektura baza podataka

U homogenim distribuiranim bazama sustav se sastoji od mreže i dvije ili više Oracle baza, koje su raspoređene na jedno ili više računala. Aplikacija može istovremeno pristupati i mijenjati podatke u nekoliko baza podataka u jednom distribuiranom okruženju. Kod klijentske aplikacije, lokacija i platforma baze su transparentne. Mogu se kreirati sinonimi za objekte u distribuiranim sustavima kako bi im korisnik mogao pristupiti s istim sintaksama kao da je lokalni objekt. Ovi tipovi baza podataka su usko povezani ali različiti. U distribuiranoj bazi podataka sustav upravlja s jednom kopijom podataka i podržava objekte baze. Aplikacije distribuirane baze podataka koriste distribuirane transakcije za pristup lokalnim i udaljenim podacima. Na taj način prilagođavaju se i upravljaju globalnom bazom u realnom vremenu. Replikacija se odnosi na operacije kopiranja i održavanja objekata koji se nalaze u velikom broju baza jednog distribuiranog sustava. Iako replikacija ovisi o distribuiranoj tehnologiji, takve baze pružaju aplikacijama neke prednosti koje nisu moguće u distribuiranim okruženjima. U heterogenim distribuiranim sustavima najmanje jedna baza podataka nije od Oracle sustava. Aplikaciji se heterogeni distribuirani sustavi pojavljuju kao jedna lokalna Oracle baza podataka. Lokalni Oracle server skriva distribuciju i heterogenost podataka. Pristupa toj bazi koristeći Oracle heterogene usluge (engl. *Heterogeneous Services*) ili skraćeno HS. Heterogene usluge su integrirana komponenta u Oracle serveru. To je tehnologija koja omogućava transparentni pristup proizvodima. HS pruža uobičajenu arhitekturu i administrativne mehanizme za transparentnost proizvoda i druge heterogene sadržaje. Podržava i sve prijašnje verzije. Za svaki pristup sustavu baza podataka koja nije Oracle, heterogene usluge mogu koristiti sučelje agenta transparentnog pristupa. Agent je specifičan za takav sustav, pa svaki tip zahtjeva različitog agenta. Agent olakšava komunikaciju između Oracle baze podataka i baze koja nije Oracle, te koristi heterogene usluge na

Oracle serveru. Izvršava SQL i transakcijske zahtjeve na drugoj bazi u ime Oracle poslužitelja. Generička povezanost dopušta spajanje na drugu bazu koristeći se heterogenim uslugama putem agenta. Bilo koji izvor podataka, koji je kompatibilan s ODBC[7] i OLE DB[8] standardima, može biti pristupljen koristeći se generičkim agentom za povezivanje. Prednost generičke povezanosti je da se od korisnika ne traži kupnja različitih sustava. Korisnik može koristiti ODBC OLE DB driver koji se može povezati s agentom. Međutim, neke značajke pristupa podacima su dostupne samo s transparentnim pristupnim agentima. Poslužitelj baze podataka je Oracle softver koji upravlja bazom, a klijent je aplikacija koja traži informacije od poslužitelja. Svako računalo u mreži je čvor koji može imati jednu ili više baza podataka. Svaki se čvor u distribuiranim sustavima može ponašati kao klijent, poslužitelj ili jedno i drugo, ovisno o situaciji. Klijent se može povezati direktno ili indirektno na poslužitelj. Direktna veza se ostvaruje kad se klijent spoji na poslužitelj i pristupi informacijama u bazi podataka koja se nalazi na tom poslužitelju. Indirektna veza se događa kad se klijent spoji na poslužitelj i nakon toga pristupi informacijama koje se nalaze u bazi podataka na drugom poslužitelju (Oracle, 2018).

[7] ODBC ili engl. *Open Database Connectivity* je protokol koji se koristi za povezivanje baze podataka s vanjskim izvorom podataka kao što je Microsoft SQL Server.

[8] OLE DB ili *Object Linking and Embedding, Database* je aplikacijsko programsko sučelje, dizajnirano od Microsofta koje dopušta pristup podacima sarazličitih izvora.

6. ZAHTJEVI PLATFORME

Čak i u najboljim implementacijama, distribuirano okruženje (engl. *Distributed Database Environment*) je u konačnici kompliciranа, računalna aplikacija. U teoriji, moguće je da svaki dio softvera arhitekture bude razmješten na različitom, zasebnom računalu. Očito, u praksi to nije slučaj te je dvojbeno da postoji značajna prednost implementacije arhitekture s tolikom distribucijom. Nitko ne bi odlučio implementirati softver s tim stupnjem distribucije samo zato što treba biti moguće. Međutim, činjenica da možemo na taj način implementirati okruženje, trebala bi djelovati kao značajan motivator u određivanju temeljnih zahtjeva za računala i softver koji koristimo za implementaciju distribuiranog okruženja. U pokušaju da se spriječi pristranost prema određenom hardverskom i operacijskom sustavu, lokalnom okruženju baza podataka, programskom jeziku ili nekoj drugoj tehnologiji napravit ćemo kombinaciju svih gore navedenih. Platforma može biti zastrašujući zadatak. Može se odabrati homogeno ili heterogeno okruženje prema potrebi, dokle god se osigurava da će svi odabiri biti u mogućnosti integrirati se u okruženje i moći učinkovito komunicirati sa svima. U prošlosti je to bio teži problem nego danas. Tehnološki napredak u računalnoj snazi, tehnologijama umrežavanja i programskim jezicima učinio je distribuirano računalstvo razumno lako, učinkovito i gotovo sveprisutno. Isto tako, sada imamo niz različitih softverskih paketa uključujući komercijalnu off-the shelf platformu, kao i besplatnu open-source platformu za odabir prilikom pokušaja da osiguramo prenosivost i interoperabilnost za djelove naše arhitekure, koje implementiramo u različitim hardverskim i operacijskim sustavima te programskim jezicima.

Postoje nekoliko različitih vrsta besplatnog softvera. Primarna razlika je u detaljima licenciranja. Jedni od njih su GNU operacijski sustav koji je sličan *Unixu* te *Apache*[9] poslužitelj. Komponente mogu biti implementirane pomoću nekoliko različitih tehnologija, a mogu uključivati kôd, podatke i konfiguraciju pojedinosti. Podsustav je zbirka jedne ili više komponenti (i možda jedan ili više drugih podsustava) koji djeluje prema zajedničkom cilju. On je dio cjelokupne arhitekture, ali je i sustav sam po sebi. Teoretski, neki podsustavi se mogu sastojati od jedne komponente, ali je vjerojatnije

[9] *Apache* je HTTP server projekt koji nastoji razviti i održavati open-source HTTP poslužitelj za suvremene operacijske sustave, uključujući UNIX i WIndows.

da će podsustav uključiti nekoliko komponenti. Komponente i podsustavi su slični u mnogim aspektima, a primarna im je razlika pitanje opsega i razmjera (Rahimi i Haug, 2010).

Postoji nekoliko različitih pristupa koje možemo birati kada pokušavamo uhvatiti način na koji se dijelovi naše arhitekture uklapaju u stvaranje okruženja. Model s tri razine je korisna tehnika za vizualizaciju i raspravu o distribuiranim aplikacijskim arhitekturama. Pristup toj vizualizaciji je bio popularan s mnogim implementacijskim rješenjima 90-ih godina i često se koristio za opisivanje distribuiranih aplikacija pisanih pomoću objektno-orijentiranog sustava *Microsoft Common Object Model*. Model s tri razine se dijeli na tri različita nivoa. Svaka komponeneta ili podsustav koji sadrži izvršni kôd je dodijeljen jednoj od razina baziranih na primarnoj funkciji koja se izvodi. Te tri razine su nazvane:

- Razina korisničkih usluga
- Razina poslovnih usluga
- Razina podatkovnih usluga

Primarna funkcija razine korisničkih usluga je informativna prezentacija. Softver na toj razini direktno ne izvodi integraciju CRUD (*Create, read, update and delete*) operacija. Na drugoj razini se provode poslovna pravila. To su pravila o validaciji na temelju poslovne logike i druge poslovne semantike. Dakle, primarna je funkcija provođenje poslovne semantike. Treća i konačna razina pruža rutine koje se bave pohranjivanjem, pronalaženjem i manipulacijom temeljnih podataka. Obrađuju se informacije kao jednostavne vrste podatka bez ikakve poslovne semantike. Uslužno-orijentirana arhitektura (engl. Service-oriented architecture) je još jedna korisna tehnika za vizualizaciju arhitekture aplikacija. U posljednje vrijeme prima mnogo pozornosti zbog svoje primjenjivosti na više modernih tehnologija, kao što su XML[10] web usluge i prednosti koje ona ima za distribuirane aplikacije. Rađe nego usredotočujući se na razine, ova se arhitektura bavi interakcijom između usluga. Zagovornici tvrde da je to praktičnije rješenje za upravljanje složenošću uključenih u implementaciju i održavanje usluga u stvarnom svijetu (Rahimi i Haug, 2010).
Integrirani softver razvijen pomoću različitih računalnih jezika često zahtijeva neki prijevod vrste podataka ili neke iznimke i funkcija. U takvim slučajevima nam je

[10] XML je kratica za *Extensible Markup Language*, odnosno jezik za označavanje podataka.

potreban dodatni dio softvera kako bi to sve spojio. Tada nastupa middleware. *Middleware* je bilo koi dio softvera koji može imati razne dobavljače, verzije i ciljeve, ali za ove potrebe služi kao „ljepilo“ koje spaja sve komponenente i podsustave. Može biti implementiran pomoću pristupa (tehnika i jezika) koji je objektno ili neobjektno orijentiran. Obično se piše na nižim razinama kako bi nam dopustio ponovno efektivno korištenje u arhitekturi.

Postoje tri podkategorije unutar razvojnih zahtjeva, a to su: zahtjevi integracije, zahtjevi za proširivost (*framework*) i zahtjevi za prenosivost. Integracijski zahtjevi su usredotočeni na razvojne zahtjeve koji nam pomažu integrirati fizički komad kôda. Zahtjevi za proširenje su usredotočeni na logičke odnose između dijelova kôda, a zahtjevi za prenosivost se odnose na jednostavnu ideju da kôd treba biti što je moguće više prenosiviji. Ti zahtjevi obuhvaćaju sve potrebne aspekte za pokretanje kôda na svim platformama sadržanim u okruženju.

Nakon što razvijemo kôdove, trebamo implementirati instance na odgovarajuće mjesto unutar okruženja. Aktivnosti životnog ciklusa implementacije uključuju fizički prijenos kôda i podataka, kao i izvršavannje svih potrebnih koraka za stvaranje okoliša. Većina suvremenih operacijskih sustava ima vrlo moćne mogućnosti za kontroliranje izvršenja programa. Izvođennje programa se obično naziva procesom, ali postoje i drugi načini na koje se kôd može izvršiti.

Povezivanje s bazom podataka se odvija preko različitih dobavlajča i inačica na patformi. Microsoft koristi *Open Database Connectivity* i *.NET Framework Data Providers*[11]. Java pruža *Java Database Connectivity* kao i druge mehanizme za povezivanje s nekoliko različitih platformi baza podataka (Rahimi i Haug, 2010).

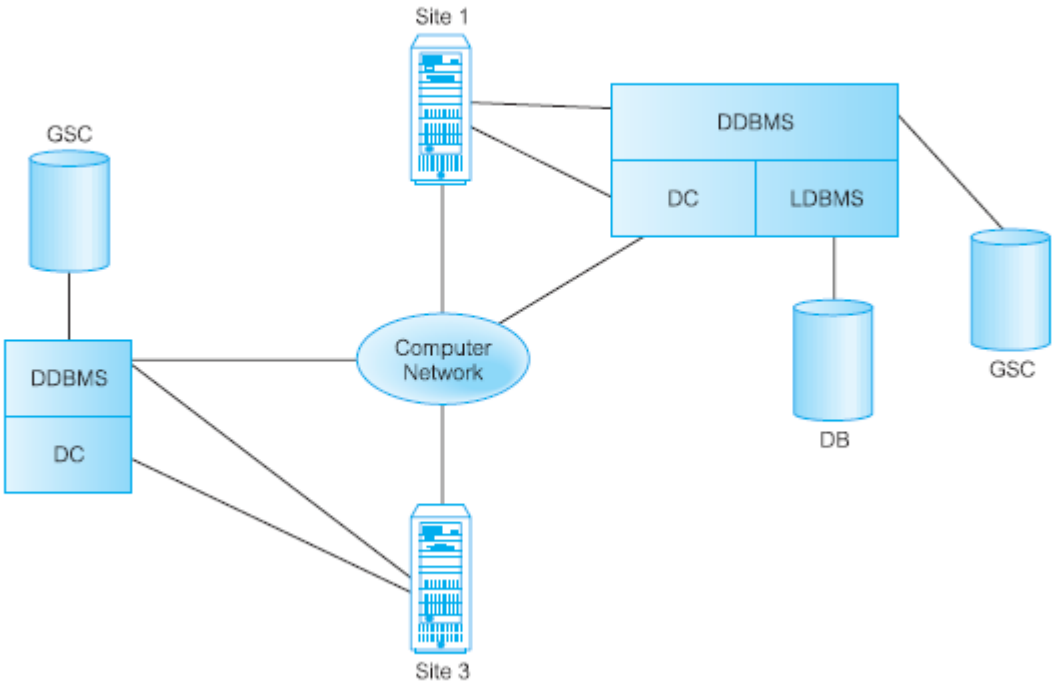
[11] *.NET Framework Dana Provider* je pružatelj podataka koji se koristi za povezivanje sa bazom, izvršavannje naredbi i dohvaćanje rezultata.

7. ARHITEKTURA DISTRIBUIRANIH BAZA PODATAKA

Neovisno o referentnoj arhitekturi, možemo identificirati komponentnu arhitekturu za DDBMS koja se sastoji od četiri glavne komponente:

- Lokalna DBMS komponenta
- Komponenta podataka komunikacije
- Globalni katalog sustava (GSC)
- Distribuirana DBMS komponenta

Komponente su prikazane na slici 7.



Slika 7. Komponente u referentnoj arhitekturi (Connolly i Begg, 2005.)

Lokalna DBMS komponenta je standardni DBMS, odgovoran za kontrolu lokalnih podataka na svako mjesto koje ima bazu podataka. Ima svoj vlastiti katalog lokalnih sustava koji pohranjuje informacije o podacima održanim na tom mjestu. U homogenom sustavu ova komponenta bi bila isti proizvod koji je repliciran na svakom mjestu. U heterogenom sustavu bilo bi barem dva mjesta s različitim DBMS proizvodima i/ili platformama. Komponenta za komunikaciju podatka je softver koji

svim web stranicama omogućuje međusobnu komunikaciju. Sadrži informacije o web stranicama i vezama.

Katalog globalnog sustava ima istu funkcionalnost kao i katalog centraliziranog sustava. Sadrži informacije specifične za distribuiranu prirodu sustava, kao što je fragmentacija, replikacija i alokacija. Sama se može upravljati kao distribuirana baza podataka i tako može biti fragmentirana, distribuirana, potpuno replicirana ili centralizirana. Za odnose uspostavljene na nekim mjestima, odgovornost je lokalnog kataloga i web lokacije da snime definiciju i repliku svakog fragmenta i da snime gdje se nalazi svaki fragment ili replika. Kada se god ulomak ili replika pomakne na drugu lokaciju, lokalni katalog na adresi mora ažurirati sva mjesta. Distribuirana DBMS komponeneta je kontrolna jedinica cijelog sustava. DDBMS može pružati različite razine transparentnosti. Međutim, svi oni sudjeluju u zajedničkom cilju, a to je omogućiti korištenje distribuirane baze podataka ekvivalentne onoj centraliziranoj bazi podataka. U DDBMS-u možemo identificirati četiri glavne vrste transparentnosti:

- Transparentnost distribucije
- Transparentnost transakcija
- Transparentnost poslovanja
- Transparentnost DBMS-a

Transparentnost distribucije omogućuje korisniku da percipira bazu podataka kao jedinstveni, logični entitet.

Transaprentonost transakcija u DDBMS okruženju osigurava da sve distribuirane transakcije održe integritet i dosljednost baze. Distriburiane transakcije pohranjuju podatke na više lokacija, te je svaka transakcija podijeljena u nekoliko subtransakcija gdje svaka pripada jednoj lokaciji.

Transparentnost izvedbe zahtijeva da se DDBMS izvodi kao centraliziran DBMS. U distribuiranom okruženju sustav ne smije trpjeti nikakvu degradaciju performansi zbog distribuirane arhitekture, kao na primjer pristupnost mreže. Zahtijeva još i utvrđivanje najisplativije strategije koja će ispunjavati zahtjeve.

Transparentnost DBMS skriva znanje da lokalni DBMS mogu biti različiti pa se stoga primjenjuje samo na heterogene sustave. To je jedna od najtežih transparentnosti.

7.1. Arhitektura distribuiranih baza podataka u višejezgrenim sustavima

Sustav je sastavljen od 3 razine: klijent web servisa, sustav za razmjenu poruka i agent baze podataka. Distribuirani sustav je sastavljen od dvije ili više PostgreSQL[12] baze koje prebivaju na jednom ili više višejezgrenih sustava. Posrednički softver komunicira s agentima web servisa i bazom podataka koji se pokreće na višejezgrenim računalima. Web servis se ponaša kao upravitelj usluga upita i kao rezultat agregatne usluge za heterogene klijente web usluga.

Klijent web servisa - Upit klijenta se širi putem poruka i servisnog sustava od poslužitelja baze podataka preko agenata baze. Klijenti web servisa istovremeno pristupaju podacima u svim bazama podataka u distribuiranom okruženju.

Sustav za razmjenu poruka - Koristili smo poruku i uslužni posrednički softver koji podržava objavljivanje i pretplatu na poruke. Sustav poruka i usluga pruža mehanizam za istovremeno širenje upita i dohvaćanja rezultata upita od i iz distribuiranih baza podataka.

Agent baze podataka - Agent prihvaća zahtjeve upita od korisnika, prenosi zahtjeve u poslužitelj baze podataka i vraća rezultate s poslužitelja. Provodi konsenzus odgovora koji se javljaju iz podataka. Ima komunikacijska sučelja za istovarivanje računarskih potreba. Također, agent generira više niti povezanih s više baza podataka (Kangseok, 2017).

[12] PostgreSQL je objektno-relacijski sustav za upravljanje bazom podataka. Najnapredniji open-source sustav baza podataka.

8. IMPLEMENTACIJA DISTBUIRANIH BAZA PODATAKA

Tri su pristupa koja se mogu koristiti kada se implementira distribuirana baza podataka:

- Fragmentacija
- Replikacija
- Alokacija

Postotak pri kojemu se ova tri pristupa kombiniraju je kritičan faktor u određivanju zahtjeva komunikacije podataka za bilo koji DDBMS. Svaki pristup ispunjava zahtjeve različitih komunikacijskih podataka.

8.1. Fragmentacija podataka

Fragmentacija se bavi razgranjivanjem i dijeljenjem tablice između različitih lokacija. Horizontalna fragmentacija prekida tablicu u redove, čuvajući sva polja (stupce) unutar tablice na zasebnom mjestu, ali samo podskup svojih redaka. Vertikalna fragmentacija pohranjuje podskup stupca tablice između različitih mjesta. Mješovita fragmentacija kombinira vertikalnu i horizontalnu fragmentaciju. Na svakoj lokaciji pohranjuju se samo podaci specifični za web mjesto. Ako su podaci potrebni iz drugih lokacija, DDBMS je dohvaća putem komunikacijske veze. Fragmentacija je važan zadatak koji se mora provesti u objektno orijentiranom distribuiranoj bazi podataka (engl. *Distributed Object Oriented Database*), skraćno DOODB. Svrha je distribucije baze podataka povećati obradu upita i postići visoku vrijednost. Slično relacijskom modelu, fragmentacija u DOODB provodi se vodoravno i koristi se zajedno. Svaki objekt ima istu strukturu i drukčije stanje ili sadržaj. Dakle, horizontalni fragment klase sadrži podskup cijelog proširenja klase. Vertikalna fragmentacija razbija logičku strukturu klase: atribute i metode, te ih distribuira preko njegovih fragmenata. Cilj je ovdje navesti atribute klase i metode kojima se zajednički pristupa podskupom atributa i metoda (Darabant, 2004).

Četiri su razloga za fragmentiranje veza (Connolly I Begg, 2005):

- Upotreba - općenito, aplikacije rade s pregledima (*view*), a ne sa cijelim odnosima. Stoga, za raspodjelu podataka, čini se primjerenim raditi s podskupovima odnosa kao jedinice distribucije.
- Učinkovitost - Podaci o učinkovitosti spremaju se tamo gdje se najčešće koriste. Osim toga, podaci koji nisu potrebni lokalnoj aplikaciji, nisu pohranjeni.
- Paralelizam – s fragmentima kao i jedinicom distribucije, transakcija se može podijeliti u nekoliko podupita koji djeluju na fragmentima. To bi trebalo povećati stupanj konkurenstnosti u sustavu čime se omogućava transakcijama, koje to mogu učiniti sigurno, da se izvrše paralelno.
- Sigurnosni podaci – koji nisu potrebni lokalnim aplikacijama se ne pohranjuju I nisu dostupni neovlaštenim korisnicima.

Fragmentacija ima dva primarna nedostatka (Connolly i Begg, 2005):

- Izvedba – performanse globalnih aplikacija koje zahtijevaju podatke iz nekoliko fragmenata koji se nalaze na različitim mjestima mogu biti sporiji
- Integritet – kontrola integriteta može biti teža ako su podaci i funkcionalne ovisnosti fragmentirane i locirane na različitim mjestima.

8.2. Replikacija i alokacija podataka

Replikacija baze podataka važan je mehanizam, jer omogućuje organizacijama pristup aktualnim podacima gdje i kada im je potrebna. Promjene primjenjene na jednom mjestu bilježe se i pohranjuju lokalno prije prosljeđivanja na udaljene lokacije. Replikacija koristi distribuiranu tehnologiju baze podataka koja dijeli podatke između više web mjesta, ali replicirane i distribuirane baze podataka nisu iste. U raspodijeljenoj bazi podataka, podaci su dostupni na mnogim lokacijama, ali određeni odnos ne nalazi se samo na jednom mjestu. Neke od glavnih prednosti replikacije podataka su: dostupnost, pouzdanost, izvođenje, smanjenje opterećenja, isključeno računanje, podržavanje mnogih korisnika i podržavanje naprednih aplikacija (Conolly i Begg, 2005).

Dostupnost se odnosi na način na koji replikacija povećava dostupnost podataka za korisnike i aplikacija putem pružanja alternativnih mogućnosti pristupa podacima. Ako jedno mjesto posatne nedostupno, korisnici mogu nastaviti s upitom ili čak ažurirati

preotale lokacije.

Pouzdanost se odnosi na činjenicu da više kopija podataka dostupnih preko sustava pruža izvrsnu opremu za oporavak u slučaju neuspjeha sustava na jednom ili više mjesta.

Izvedba se osobito poboljšava za transakcije upita kada se uvede replikacija u sustav, koji je pretrpio značajno preopterećenje centraliziranih resursa. Replikacija omogućuje brz i lokalni pristup podijeljenim podacima, jer uravnotežuje aktivnost više mjesta. Neki korisnici mogu pristupiti jednom poslužitelju, dok drugi korisnici pristupaju različitim poslužiteljima, čime se odražavaju razine performansi na svim poslužiteljima. Redukcija opterećenja odnosi se na način na koji replikacija može biti korištena za distribuciju podataka preko više udaljenih lokacija. Zatim, korisnici mogu pristupiti različitim udaljenim poslužiteljima, umjesto da pristupaju jednom središnjem poslužitelju. Ova konfiguracija može značajno smanjiti mrežni promet. Također, korisnici mogu pristupiti podacima s mjesta replikacije s najnižim troškom pristupa, što je obično zemljopisno najbliže njima.

Isključeno računanje odnosi se na način na koji replikacija može biti podržana snimkama. Snimka je cjelovita ili djelomična kopija cijelog odnosa iz jedne točke u vremenu. Snimke korisnicima omogućuju rad na podskupu korporativne baze podataka dok su isključene od glavnoh poslužitelja baze podataka. Kasnije, kada se veza ponovno uspostavlja, korisnici mou sinkronizirati (osvježiti) snapshotoe s korporacijskom bazom podataka prema potrebi. To može značiti da snimka prima ažuriranja iz korporativne baze podataka ili snimke ažuriranje te baze. Bez obzira na akciju snimljene podatke u snimci i korporativna baza podataka još jednom je dosljedna.

Podržava mnoge korisnike koji se odnose na način na koji organizacije sve više trebaju implementirati mnoge aplikacije koje zahtijevaju mogućnost korištenja i manipulacije podacima. Replikacija može stvoriti više prilagođenih snimki koje zadovoljavaju zahtjeve svakog korisnika ili grupe korisnika sustava.

Podrška naprednim aplikacijama se odnosi na sve organizacije, koje sve više trebaju učiniti korporativne podatke dostupne, ne samo za tradicionalne online transakcije, već i za analizu podataka kao što je skladištenje podataka i online analitička obrada (OLAP). Nadalje, putem replikacije korporativni podaci također mogu biti dostupni za podršku sve popularnijeg trenda mobilnog računanja (Conolly i Begg, 2005). Replikacija podataka je kopiranje podataka i može postojati neovisno o tome

postoji li ili ne postoji distribuirana baza podataka. Postoji mogućnost da se repliciraju podaci unutar iste baze, ili se mogu replicirati između nepovezanih baza. Pod replikacijom se smatra replikacija unutar distribuirane baze podataka. To je proces kreiranja kopija podataka i sinkroniziranja podataka u bazama koje čine distribuiranu bazu podataka. Sve promjene koje se naprave na jednoj bazi moraju se odraziti i na kopije u ostalim bazama. Oracle ima više vrsta replikacije. Postoje bazična replikacija, napredna replikacija i novija replikacija ili *Oracle Streams*[13] (Brueggen i Sooun, 1995). Alokacija je kombinacija replikacije i fragmentacije. Zapisi se pohranjuju na čvorovima koji pokazuju najveću upotrebu tih podataka, čime se maksimizira lokalna obrada (Brueggen i Sooun, 1995). Usporedba strategija kod alokacije podataka prikazana je na slici 8.

	Locality of reference	Reliability and availability	Performance	Storage costs	Communication costs
Centralized	Lowest	Lowest	Unsatisfactory	Lowest	Highest
Fragmented	High[a]	Low for item; high for system	Satisfactory[a]	Lowest	Low[a]
Complete replication	Highest	Highest	Best for read	Highest	High for update; low for read
Selective replication	High[a]	Low for item; high for system	Satisfactory[a]	Average	Low[a]

[a] Indicates subject to good design.

Slika 8. Usporedba strategija kod alokacije podataka (Conelly i Begg, 2005)

[13] *Oracle Streams* pruža felksibilnu infrastrukturu koja zadovoljava široku paletu potreba za dijeljenjem podataka.

9. DISTRIBUIRANE TRANSAKCIJE

U distribuiranoj transakciji baze imaju različite uloge: globalni koordinator, commit point site, poslužitelj baze podataka i klijent. Transakcija je prikazana na slici 9. Značenja tih uloga su sljedeća (Sirotić, 2014):

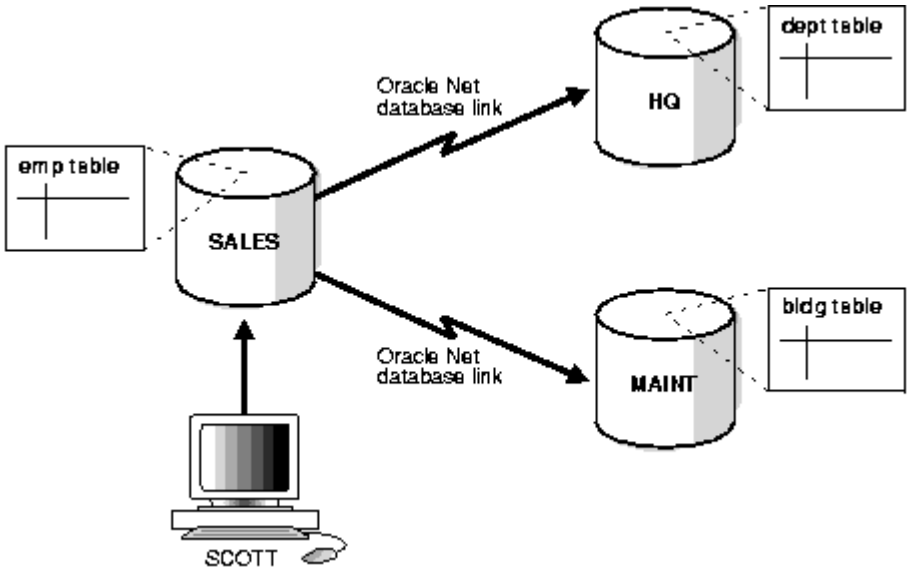
- Globalni koordinator – je baza koja je izvor distribuirane transakcije. Aplikacija koje je pokrenula distribuiranu transakciju je vezana za bazu koja je globalni koordinator. Koordinator vrši sljedeće operacije prilikom provođenja transakcije:

1. Šalje SQL naredbe udaljenih procedura na određenu bazu.
2. Šalje svim bazama s kojima je direktno povezan, osim *commit point site* bazi, naredbe da se pripreme.
3. Ako sve baze odgovore da su spremne, šalje *commit point site* bazi naredbu da izvrši *commit*.
4. Ako neka baza odgovori s abort, šalje naredbu da se izvrši globalni *rollback*.

- *Commit point site* – je baza koja inicira globalni commit ili globalni rollback, kada dobije takvu instrukciju od koordinatora. *Commit point site* baza bi trebala biti ona koja čuva najvažnije podatke. Razlikuje se od ostalih baza po dvije karakteristike:

1. Nikada ne ulazi u pripravno stanje, dok ostale baze u slučaju greške ostaju u pripravom stanju dok se god ne riješi *in-doubt* transakcija.
2. Izvršava *commit* prije svih ostalih baza, nakon čega je distribuirana transakcija izvršena.

- Poslužitelj baze podataka – je bilo koja baza podataka koja sadrži podatke koje druge baze referenciraju.
- Klijent – baza koja referencira podatke na drugim bazama.



Slika 9. Distribuirane transakcije (Oracle, 2018)

9.1. Dvofazni commit protokol

Dvofazni *commit* protokol se sastoji od tri faze: faza pripreme (*prepare phase*), faza potvrde (*commit phase*) i faza zaboravljanja (*forget phase*). Ove tri faze rade samo kada distribuirana transakcija završava s naredbom *commit*. U slučaju da završava s *rollback*, globalni koordinator šalje svim bazama da provedu naredbu *rollback*. U fazi pripreme globalni koordinator traži da se pripreme ostale baze *(prepared state*). To se jedino traži od baze koja je commit point site. Baze prelaze u pripravno stanje na način da spreme sve podatke u svoju redo log datoteku bez obzira hoće li kasnije izvršiti jednu od dviju naredbi. U ovoj se fazi postavlja lokot na modificirane tablice. To je ključan korak jer lokot sprječava čitanje podataka pa ako dođe do nekog problema s distribuiranom transakcijom podaci ostaju zaključani i za čitanje. Nakon pripreme baza čeka daljnje upute. U mogućnosti je odgovoriti na tri načina, a to su (Sirotić, 2014):

Prepared - baza javlja da je pripravna i zaključava tablice za čitanje. Kod toga koraka baza je u osjetljivom stanju i ostaje u njemu dok se god sve promjene ne izvrše ili povrate na staro.

Read-only – baza javlja da ona nije nikakav DML (engl. *Data manipulation language*)

Abort – baza javlja da nije u mogućnosti ući u prepared fazu te otključava sve zaključane retke i radi povratak na staro.

Ako globalni koordinator od barem jedne baze dobije odgovor *abort*, šalje svim bazama naredbu *rollback*. Ako od svih baza dobije odgovor *prepared*, započinje fazu potvrde (*comnmit phase*). Šalje naredbu *commit point site* bazi da izvrši naredbu *commit*. Ona izvršava tu naredbu i obavještava koordinatora. Tada se smatra da je distribuirana transakcija izvršena, bez obzira na to da je možda došlo do neuspjeha kod ostalih baza. Globalni i lokalni koordinatori šalju naredbe svim svojim podređenim bazama da naprave naredbu *commit* (Sirotić, 2014).

U trećoj fazi (*forget phase*) koordinator kaže commit point bazi da izbriše stanja o transakciji koja vodi kod sebe, te da mu da povratnu informaciju. Nakon toga koordinator briše informacije o transakciji.

Do uspješne distribuirane transakcije dolazimo na ovaj način:

1. Aplikacija šalje pozive udaljenih procedura po bazu i završava sa *commit* naredbom.
2. Započinje *prepare phase* u kojoj glavni koordinator određuje koja je baza *commit point site*.
3. Šalje se naredba da se baze pripreme.
4. Baze odgovaraju da su spremne.
5. Počinje *commit phase* u kojoj koordinator javlja *commit point* bazi da izvrši *commit* naredbu lokalno.
6. Izvodi se naredba.
7. Koordinatori javljaju ostalim bazama da naprave naredbu.
8. Sve baze izvršavaju *commit* naredbu.
9. Počinje *forget phase* u kojoj koordinator javlja *commit point* bazi da zaboravi distribuiranu transakciju.
10. *Commit site* baza briše sve podatke i javlja globalnom koordinatoru.
11. Globalni koordinator briše sve podatke o napravljenoj transakciji.

9.2. In-doubt distibuirane transakcije

Dvofazni *commit* protokol osigurava da sve baze uspješno izvedu c*ommit* ili *rollback* naredbu, a transakcija postaje *in-doubt,* ako u nekoj od faza dođe do greške. Može se dogoditi da padne računalo na kojem je baza, da se prekine veza između baza i da se dogodi neki softverski problem. Najbolje je ostaviti da baza sama riješi *in-*

doubt transakciju. U slučaju da je zadovoljen jedan od dva uvjeta treba ručno riješiti takvu transakciju. Uvjet jedan je, da je *in-doubt* transakcija zaključala kritične podatke, a drugi uvjet je, da je došlo do kvara na računalu ili nekog drugog problema koji se ne može riješiti u kratkom vremenu (Sirotić, 2014).

Postupak ručnog rješavanja *in-doubt* transakcije je kompleksan. Rješava se u nekoliko koraka. Prvi korak je, da se pronađe identifikacijski broj *in-doubt* transakcije, zatim se postavi upit nad sistemskim view-ovima kako bi odredili jesu li baze izvršene. Ako je potrebno, forsirati ćemo *commit* naredbu sa *commit force* naredbom, a *rollback* s *rollback force* naredbom (Sirotić, 2014).

Još neke metode i protokoli su:

- *Trofazni commit protokol*
- *Paxos commit protocol*
- *Transaction Guard*
- *Application Continuity*
- *Active Data Guard*
- *Real Application Clusters*

10. SIGURNOST U DISTRIBUIRANOM OKRUŽENJU

Sigurnost, prijetnje te rješenja za sigurnost su jedanka kao i u centraliziranom okruženju, s time da su distribuirane baze podataka ugroženije jer više računala na različitim lokacijama, ima pristup podacima koji se nalaze u bazi podataka.

Sigurnosna razmatranja ne odnose se samo na podatke koji se nalaze u bazi. Kršenja pravila sigurnosti mogu utjecati i na druge dijelove sustava što zauzvrat može utjecati na samu bazu. Sigurnost baze podataka obuhvaća: hardver, softver, ljude i podatke. Kako bi bi se učinkovito provodila sigurnost, potrebne su odgovarajuće kontrole koje su definirane u specifičnim ciljevima sustava. Sigurnost je nekad bila zanemarivana, ali danas sve veći broj organizacija prepoznaje potrebu za sigurnošću jer ne žele riskirati gubitak ili nedostupnost ključnih korporativnih podataka (Connolly i Begg, 2005). Sigurnost se provodi kod sljedećih situacija:

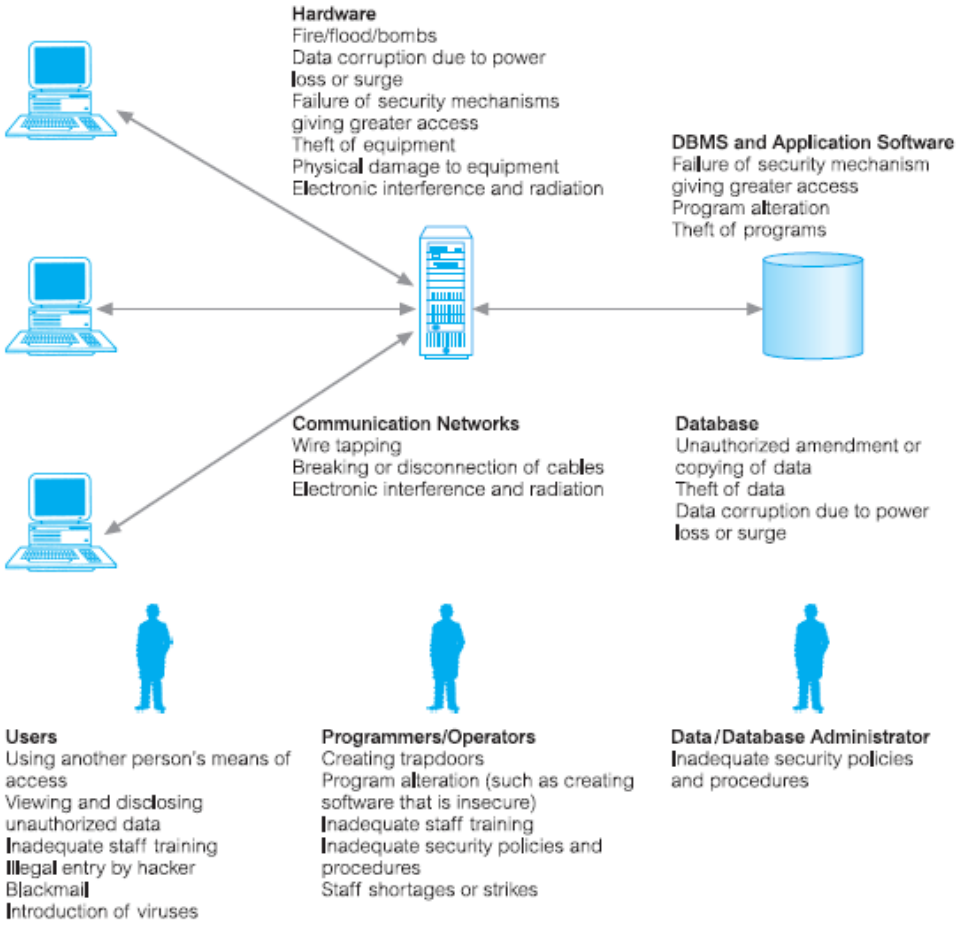
- krađa i prijevara
- gubitak povjerljivosti (tajnosti)
- gubitak privatnosti
- gubitak integriteta
- gubitak dostupnosti

Ove situacije uglavnom predstavljaju područja kojima bi organizacija trebala težiti kako bi smanjila rizik, odnosno mogućnost nastanka gubitka ili štete. U nekim slučajevima ta su područja usko povezana tako da aktivnost koja dovodi do gubitka na jednom području može dovesti do gubitka na drugom području. Osim toga, događaji kao što su prijevara ili gubitak privatnosti, mogu nastati zbog bilo namjernih ili nenamjernih radnji i ne moraju nužno rezultirati nekim vidljivim promjenama u bazi podataka ili računalnom sustavu. Krađa i prijevara utječu ne samo na okruženje baze podataka, već i na cijelu organizaciju. Pozornost treba usmjeriti na smanjenje mogućnosti takvog događaja. Prilikom toga se ne mijenjaju nužno podaci kao što je to slučaj za aktivnosti koje dovode do gubitka privatnosti. Povjerljivost se odnosi na potrebu održavanja tajnosti nad podacima, obično samo onim koja su ključna za organizaciju. Privatnost se odnosi na potrebu zaštite podataka o pojedincima. Primjerice, povrede sigurnosti koje bi dovele do gubitka povjerljivosti mogu dovesti do gubitka konkurentnosti i gubitka privatnosti, što na kraju može rezultirati poduzimanjem pravnih radnji protiv organizacije. Gubitak integriteta podataka dovodi do nevažećih ili

oštećenih podataka, što može ozbiljno utjecati na rad organizacije. Gubitak dostupnosti znači da se podacima, sustavu ili jednom i drugom ne može pristupiti, što utječe na financijsku izvedbu organizacija. Ponekak događaji koji utječu na dostupnost sustava mogu uzrokovati korupciju podataka. Sigurnost baze podataka ima za cilj minimizirati gubitke uzrokovane očekivanim događajima na isplativ način bez nepotrebnog ograničavanja korisnika. U novije doba, kriminalne aktivnosti su se znatno povećale i očekuje se da će i dalje rasti tijekom sljedećih nekoliko godina (Connolly i Begg, 2005).

10.1. Prijetnje

Prijetnje može uzrokovati situacija ili događaj koji uključuje osobu, radnju ili okolnost koja će vjerojatno nanijeti štetu organizaciji. Primjer nekih potencijalnih prijetnji prikazan je na slici 10. Šteta može biti opipljiva, poput gubitka hardvera, softvera, podataka, ili neopipljiva kao što je vjerodostojnost podataka i povjerenje klijenata. Problem s kojim se suočava bilo koja organizacija jest identificiranje svih mogućih prijetnji. Dakle, kao minumun, organizacija bi trebala uložiti vrijeme i trud u prepoznavanje najozbiljnijih prijetnji. Neke vrste prijetnji mogu biti namjerne ili nenamjerne, ali u oba slučaja njihov utjecaj ostaje isti. Namjerne prijetnje uključuju ljude i mogu se počiniti od strane ovlašenih korisnika i neovlaštenih korisnika, od kojih neki mogu biti izvan organizacije. Svaka prijetnja mora se promatrati kao potencijalno kršenje sigurnosti koje će, ako bude uspješno, imati određeni utjecaj. Opseg posljedica prijetnje u kojoj organizacija pati ovisi o nizu čimbenika, kao što je postojanje protumjere i planova za nepredviđene okolnosti. Naprimjer, ako dođe do kvara hardvera koji oštećuje sekundarnu pohranu, sve aktivnosti obrade moraju prestati dok se problem ne riješi. Oporavak će ovisiti o broju čimbenika koji se uključuju kada su posljednje sigurnosne kopije snimljene i vrijeme potrebno za vraćanje sustavu u prethodno stanje. Organizacija mora identificirati vrste prijetnji koje može podnijeti i pokrenuti odgovarajuće planove i protumjere, imajuću u vidu troškove njihova provođenja. Nije isplativo trošiti mnogo vremena, truda i novca na potencijalne prijetnje koje rezultiraju samo manjim neugodnostima. Posao organizacije također može utjecati na prijetnje, ali rijetko. Treba uzeti u obzir rijetke događaje, osobito ako bi njihov učinak bio značajan (Connolly i Begg, 2005).



Slika 10. Potencijalne prijetnei u računalnom sustavu (Connolly i Begg, 2005)

10.2. Protumjere

Vrste protumjera prema prijetnjama računalnih sustava kreću se od fizičkih kontrola do upravnih postupka. Unatoč rasponu računalnih kontrola koje su dostupne, vrijedi napomenuti da je uglavnom sigurnost DBMS-a samo dobra kao ona operacijskog sustava. Računalne sigurnosne kontrole za višekorisničko okruženje su sljedeće:

- Autorizacija
- Kontrola pristupa
- Prikazi (*Views*)
- Sigurnosno kopiranje i oporavak
- Integritet
- Šifriranje

- RAID tehnologija

Autorizacija je davanje prava ili povlastice koja subjektu omogućuje legitiman pristup sustavu ili objektu sustava. Autentifikacija je mehanizam koji određuje je li korisnik zaista onaj kojim se predstavlja. Administrator sustava obično je odgovoran za dopuštanje pristupa korisnicima stavranjem pojedinih koisničkih računa. Svaki je korisnik jedinstven identifikator koji operacijski sustav koristi kako bi utvrdio tko su oni. Lozinka je udružena sa svakim identifikatorom koju korisnik sam odabere i koja je poznata operacijskom sustavu. Pomoću nje sustav može provjeriti ili otkriti tko je korisnik s kojim se predstavlja (Connolly i Begg, 2005).

Kontrola pristupa za sustav baze podataka se temelji na dodjeli i povlačenju povlastica. Povlastica omogućuje korisniku stvaranje ili pristup (čitanje, pisanje i izmjena) nekom objektu poput odnosa, prikaza ili indeksa. Korisnicima se dodjeljuju povlastice za obavljanje zadataka. Budući da prekomjerno odobravanje nepotrebnih povlastica može ugroziti sigurnost, privilegiju treba dati samo korisniku ako taj korisnik nije u mogućnosti izvršiti svoj posao bez te privilegije. Korisnik koji stvara objekt baze podataka kao što je veza ili prikaz (view) automatski dobiva sve privilegije na tom objektu. DBMS naknadno prati kako se te povlastice dodjeljuju drugim korisnicima te ih može opozvati u svakom trenutku (Connolly i Begg, 2005).

Prikazi (*Views*) su dinamičan rezultat jednog ili više relacijskih operacija na kojima se rade temeljni odnosi za stvaranje drugog odnosa. Prikaz je virtualni odnos koji zapravo ne postoji u bazi podataka, ali se radi na zhtjev određenog korisnika. Mehanizam prikaza osigurava snažan i fleksibilan sigurnosni mehanizam skrivajući dijelove baze podataka od određenih korisnika (Connolly i Begg, 2005).

Sigurnosno kopiranje je postupak povremenog uzimanja kopije baze podataka i log datoteke (i moguće programe) na offline medij za pohranu. Uvijek je poželjno napraviti sigurnosne kopije baze podataka i log datoteke u redovitim razmacima i kako bi se osiguralo kopiranje na sigurnoj lokaciji. U slučaju neuspjeha koji čini bazu podataka neupotrebljivom, sigurnosnu kopiju i detalje zabilježene u log datoteci koristimo za vraćanje baze podataka na najnovije moguće dosljedno stanje (Connolly i Begg, 2005).

Ograničenja inegriteta također pridonose održavanju sigurne baze podataka na način da sprečavaju podacima da postanu nevažeći i da ne daju pogrešne ili netočne

rezultate.

Šifriranje je kodiranje podataka posebnim algoritmom, koji daje podatke nečitljive bilo kojim programom bez ključa za dešifriranje. Ako sustav baze podataka sadrži posebno osjetljive podatke, nužno je kodirati ih kao mjeru opreza protiv mogućih vanjskih prijetnji ili pokušaja pristupa (Connolly i Begg, 2005).

RAID (engl. *Redundant Array of Independent Disks*) radi na način da ima velik disk koji obuhvaća raspored više nezavisnih diskova koji su organizirani kako bi poboljšali pouzdanost i istodobno povećali performanse. Poboljšanje performansi povećava se podatkovnim prugama tako da se podaci segmentiraju u jednake veličine. Odnosno, particije koje su zatim transparentno distribuirane na više diskova. Prividno daje izgled jednog velikog, brzog diska gdje se zapravo podaci distribuiraju preko nekoliko manjih diskova (Connolly i Begg, 2005).

ZAKLJUČAK

Distribuirane baze podataka koriste tvrtke i organizacije koje su smještene u različitim dijelovima svijeta. Za njihov rad im je potreban pristup podacima u realnom vremenu. Za takav pristup im je potrebna komunikacijska mreža te softverski paketi uključujući komercijalnu off-the shelf platformu kao i besplatnu open-source platformu, za odabir prilikom pokušaja da osiguramo prenosivost i interoperabilnost za djelove naše arhitekure koje implementiramo u različitim hardverskim i operacijskim sustavima te programskim jezicima.

Postoje nekoliko različitih vrsta besplatnog softvera. Svaki je poslužitelj baze u distribuiranom okruženju upravljan od strane lokalnog DBMS-a i sa svim kooperantima kako bi održali konzistenciju globalne baze. U radu s distribuiranim bazama podataka ne trebamo prekidati ni jednu funkcionalnu stranicu da dodamo novu. Uklanjanje stranica nije zahtjevno kao ni proširenje cijelog sustava. Ako se podaci distribuiraju, obrada se može nastaviti i pritom se štedi i vrijeme i novac. Za komunikaciju se izdvaja manji trošak nego kod centraliziranih baza podataka. Distribuirane baze su ekonomičnije i podaci se distribuiraju na takav način da su dostupni blizu lokacija gdje su potrebni. Zahtjevi se mogu ispuniti brzo u usporedbi sa centraliziranim bazama podataka. Korisnici imaju lokalnu kontrolu, te time mogu uspostaviti i provoditi lokalne politike vezane za korištenje podatka.

Transakcije se u distribuiranim bazama podataka ne vode samo preko poslužitelja i klijenta. U transakciji sudjeluju još i globalni koordinator te commit point site. Svatko od njih ima svoju ulogu. Transparentnost mreže, fragmentacija i replikacija implementirane su da skriju stvarne implementacijske detalje cijelog distribuiranog sustava. No najveća prednost je ta što sustav nastavlja s radom ukoliko dođe do nekog pada sustava li greške. Za to je zaslužna replikacija podataka, koja pravi kopije baze i svi podaci ostanu sačuvani.Replikacija baze podataka važan je mehanizam jer omogućuje orgaizacijama pristup aktualnim podacima gdje i kada im je potrebna. Promjene primjenjene na jednom mjestu bilježe se i pohranjuju lokalno prije prosljeđivanja na udaljene lokacije.

Iako imaju puno prednosti, distribuirane baze imaju i nedostataka. Neki od nedostataka su kompleksnost sustava, veliki troškovi i kompliciran dizajn samog

sustava. Kompleksan je radi replikacije podataka koja može narušavati dostupnost. Trošak je velik kod implementacije, iako su dugoročno isplativije od centraliziranih. Gubitk važnih i ključnih informacija organizacije bi bio neprocjenjiv trošak. Dizajn kompliciranim čini fragmentacija. Iako su ovo veliki nedostaci, ipak je najveći nedostatak sigurnost. Danas ima jako puno prijetnji. S razvojem tehnologije i većom uključenošću u digitalni svijet, dolazi do većeg broja krađa, prijevara, gubitka privatnosti itd. Organizacije trebaju unaprijed razmišljati o takvim problemima te bolje provoditi protumjere kako bi svi podaci ostali sigurni, a i time sami ljudi koji su zahvaljujući velikom razvoju tehnologija uključeni u svakojake aktivnosti.

LITERATURA

1. Breuggen D., Seoun L. : Distributed Data Base Systems, Information Systems Management, Vol. 12, No. 2, 1995.
2. Connolly T., Begg C. : Database Systems: A Practical Approach to Design, Implementation and Management, Addison Wesley, London, 2005.
3. Darabant A. S., Campan A.: Optimal Class Fragmentation Ordering in Object Oriented Databases, Vol. 45, No. 1, 2004.
4. Date C. J. : An Introduction to Database Systems, Addison-Wesley Educational Publishers Inc, 2003.
5. Google Inc: Spanner: Google's Globally Distributed Database Systems, ACM Transactions on Computer Systems, Vol. 12, No.2, 2013.
6. Iacob, N. M., Moise M. L. : Centralized vs. Distributed Databases. Case Study, *Academic Journal of Economic Studies*, Vol. 1, No.4, 2015.
7. Kangseok K. : Performance of Distributed Database Systems built on Multicore Systems, Journal of Internet Computing and Services, 2017.
8. Manger, R. : *Baze podataka*, Element, Zagreb, 2012.
9. Moritz D., Halperin D., Howe B., Heer J.: Perfopticon: Visual Query for Distributed Databases, Eurographics Conference on Visualization, Vol. 34, No. 3, 2015.
10. Nemoto T., Masumoto S., Nanogaki S., Raghavan V., Yonezawa G. : Development of a Distributed Database System for Borehole Dana using Free and Open Source Software, International Journal of Geometrics, Vol.11, No. 3, 2015.
11. Oracle,URL: https://docs.oracle.com/cd/B28359_01/server.111/b28310/ds_concepts001.htm#ADMIN12074
12. Oracle,URL: https://docs.oracle.com/cd/A57673_01/DOC/server/doc/SCN73/ch21.htm
13. Oracle,URL: https://docs.oracle.com/cd/A57673_01/DOC/server/doc/SCN73/ch21.htm
14. Petrini M. : Distributed Databases Management using Remote Access Method, Annals of University of Petrosani, Economics, Vol. 9, No. 2, 2009.
15. Rahimi S.K., Haug F.S. : Distributed Database Management Systems: A Practical Approach, A john Wiley and Sons Publication, USA, 2010.

16. Richardson R. :Disambiguating Databases, Communications of the ACM, Vol. 58, No. 1, 2015.
17. Sirotić Z. :Trebaju li nam distribuirane baze u vrijeme oblaka?, IstraTech, 2014.
18. Sung Soo, URL: http://sungsoo.github.io/2014/05/17/distributed-database-management-system-introduction.html
19. Techtarget, URL: https://searchsqlserver.techtarget.com/definition/DDBMS
20. Techtarget, URL: http://searchoracle.techtarget.com/definition/distributed-database

POPIS SLIKA

SAŽETAK

Distribuirana baza podataka je baza podataka koja je podijeljena na više računala koja se nalaze na različitim lokacijama. Ta računala su povezana komunikacijskom mrežom. Svako računalo ima vlastiti sustav za upravljanje bazama podataka ili DBMS. Distribuirane baze se dijele na tri tipa sustava, a to su heterogeni i homogeni sustav te klijent/poslužitelj arhitektura. Implementacija distribuiranih baza podataka je složenija, ali su kasnije troškovi puno manji u odnosu na centralizirane baze podataka. Sa stalnim pristupom internetu i tehnologijama koje nam pruža, dolazi do većeg razvoja prijetnji. Organizacije moraju biti u pripravnosti i poduzeti sve mjere kako bi zaštitile svoje podatake u distribuiranim bazama podatka.

Ključne riječi: Distribuirane baze podataka, DDBMS, fragmentacija, replikacija, sigurnost baze podataka

ABSTRACT

Distributed database is database that is divided on several computers on different locations. That computers are connected via comunication network. Each computer has got it's own Database Management System or DBMS. There are three types of distributed systems: heterogeneous, homogenous and Client/Server architecture. Implementation of a distributed database is complex, but overall cost is a lot smaller than in centralized databases. With constant access to the Internet and all of the tehnologies we are given, there are bigger chances of encountering all kinds of threaths. Organisations need to be on alert all the time so they can take mesaures to protect their data in distributed database on time.

Keywords: *Distributed databases, DDBMS, fragmentation, replication, database security*