Distribuirane baze podataka

Šlosel, Toni

Sveučilište u Rijeci – Fakultet informatike i digitalnih tehnologija

Sveučilišni diplomski studij Informatike – modul

Informacijski i komunikacijski sustavi

Toni Šlosel

Distribuirane baze podataka

Diplomski rad

Mentor: prof. dr. sc. Patrizia Poščić

Rijeka, prosinac 2022.

FAKULTET INFORMATIKE I DIGITALNIH TEHNOLOGIJA
SVEUČILIŠTE U RIJECI

Rijeka, 30. lipnja 2022.

Zadatak za diplomski rad

Pristupnik: Toni Šlosel

Naziv diplomskog rada: Distribuirane baze podataka

Naziv diplomskog rada na eng. jeziku: Distributed Databases

Sadržaj zadatka:

Za razliku od centralizirane baze podataka koja čuva svoje podatke na uređajima za pohranu koji se nalaze na jednom mjestu, distribuirani sustav baze podataka čuva svoje podatke na uređajima za pohranu koji se mogu nalaziti na različitim zemljopisnim lokacijama i kojima se upravlja pomoću središnjeg DBMS-a. Centralizirana baza podataka je tradicionalni pristup za pohranu podataka u velikim poduzećima. Zadatak diplomskog rada je dati teorijske osnove i postavke distribuirane baze podataka, navesti prednosti i nedostatke te usporediti s centraliziranim bazama. Nadalje, potrebno je opisati različite tipove distribuiranih baza podataka, njihovu arhitekturu te sustav za upravljanje te prikazati gdje se takve baze podataka koriste.

Mentor:

Prof. dr.sc. Patrizia Poščić

Voditeljica za diplomske radove:

Izv. prof. dr. sc. Ana Meštrović

Komentor:

Zadatak preuzet: 30. lipnja 2022.

(potpis pristupnika)

Sažetak

Za razliku od centralizirane baze podataka koja čuva svoje podatke na uređajima za pohranu koji se nalaze na jednom mjestu, distribuirani sustav baze podataka čuva svoje podatke na uređajima za pohranu koji se mogu nalaziti na različitim zemljopisnim lokacijama i kojima se upravlja pomoću središnjeg DBMS-a. Računala unutar sustava su povezana komunikacijskom mrežom, a svako računalo ima vlastiti sustav za upravljanje bazama podataka. Razlikuju se dvije osnovne vrste distribuiranih baza podataka – homogene i heterogene. Implementacija podataka u distribuiranom okruženju je složenija, ali su naknadni troškovi manji u usporedbi s centraliziranim bazama podataka.

Ključne riječi: Distribuirane baze podataka, baze podataka, distribuirani sustavi, DDBMS, arhitektura distribuiranih baza, fragmentacija, replikacija

1. Uvod

Od samog početka razvoja informatičkih tehnologija postoji potreba za obradom, pregledom i pohranom podataka što nam omogućuju baze podataka. Prikupljaju se informacije o ljudima, mjestima ili stvarima. Te informacije su skupljene na jednom mjestu kako bi se mogle promatrati i analizirati. Baze podataka mogu se smatrati organiziranom zbirkom informacija.

Distribuirana baza podataka označava bazu podataka koja nije ograničena na jedan sustav, već je raširena na nekoliko web lokacija, odnosno na više poslužitelja ili putem mreže poslužitelja na više lokacija. Pored toga, distribuirani sustav baza podataka nalazi se na nekoliko web mjesta koja ne dijele fizičke komponente. Korisne su u situacijama ako različiti korisnici širom svijeta trebaju pristupiti određenoj bazi podataka, bilo na istoj mreži ili na potpuno različitim mrežama. Fragmenti baze podataka pohranjuju se na više fizičkih lokacija, a obrada se također dijeli na nekoliko poslužitelja baze podataka.

Zadatak ovog diplomskog rada je dati teorijske osnove i postavke distribuirane baze podataka, odnosno sintetizirati teorijska znanja o distribuiranim bazama podataka. Kratkim pregledom na baze podataka stvorit će se osnova za okruženje teme ovog rada, a pregledom kroz različita obilježja distribuiranih baza podataka, cilj je istaknuti prednosti i nedostatke ovakvog pristupa pohrane podataka te izvršiti kratku usporedbu s centraliziranim pristupom pohrane podataka. Također, sustav za upravljanje bazom podataka je neophodan softver koji omogućava upravljanje bazom podataka pa je u radu opisan, istaknute su njegove prednosti i nedostaci te navedena i opisana arhitektonska rješenja.

U poglavlju Baze podataka pojašnjeno je što su to baze podataka te ukratko objašnjena osnovna obilježja baza podataka i sustava za upravljanje bazom podataka. U poglavlju Distribuirane baze podataka su opisane distribuirane baze podataka te istaknute njihove prednosti i nedostaci. U poglavlju 4 je izvršena usporedba između distribuiranih i centraliziranih baza podataka. U poglavlju 5 su navedene i opisane vrste distribuiranih baza podataka. Poglavlja 6 i 7 su posvećena distribuiranom sustavu za upravljanje bazom podataka te njihovim vrstama arhitektura koje se koriste. U poglavlju 8 su navedeni i opisani načini implementacije podataka u distribuiranim bazama podataka te je u tabličnom prikazu izvršena kratka usporedba. U 9 poglavlju su navedeni neki od primjera softvera koji su u primjeni za rad s distribuiranom bazom podataka. Kao posljednje poglavlje napisan je kratki zaključak.

SADRŽAJ

2. Baze podataka

Osnovna je ideja tehnologije baza podataka da pojedina aplikacija ne stvara vlastite datoteke na disku, nego da sve aplikacije koriste zajedničku kolekciju podataka. Takvu kolekciju podataka naziva se bazom podataka. Baza podataka je skup međusobno povezanih podataka, pohranjenih u vanjskoj memoriji računala, koji su istovremeno dostupni raznim korisnicima i aplikacijskim programima(Manger, 2012). Baze podataka služe za pohranu, održavanje i pristup bilo kojoj vrsti podataka. Prikupljaju se različite informacije poput informacija o ljudima, mjestima ili stvarima i kao takve su prikupljene na jednom mjestu kako bi se mogle promatrati i analizirati. Mogu se smatrati organiziranom zbirkom informacija.

Neke od najčešće korištenih baza podataka su:

- Relacijska baza podataka - koristi tablični pristup koji definira podatke tako da ih je moguće reorganizirati i pristupiti im na više načina. Relacijske baze podataka sastoje se od tablica. Podaci su smješteni u unaprijed definirane kategorije u tim tablicama. Svaka tablica ima stupce s najmanje jednom kategorijom podataka i retke koji imaju određenu instancu podataka za kategorije koje su definirane u stupcima. Informacije u relacijskoj bazi podataka o određenom entitetu organizirane su u retke, stupce i tablice. Oni su indeksirani kako bi se olakšalo pretraživanje pomoću SQL ili NoSQL upita.
- Distribuirana baza podataka - baza podataka pohranjuje zapise ili datoteke na nekoliko fizičkih lokacija. Obrada podataka također je raširena i replicirana po različitim dijelovima mreže. Distribuirane baze mogu biti homogene, gdje sve fizičke lokacije imaju isti temeljni hardver i pokreću iste operativne sustave i aplikacije baze podataka. Mogu biti i heterogene kada se hardver, OS i aplikacije baze podataka mogu nalaziti na različitim lokacijama.
- Oblak (engl. *Cloud*) - baze podataka izgrađene su u javnom, privatnom ili hibridnom oblaku za virtualno okruženje. Korisnicima se baza naplaćuje na temelju količine pohrane i propusnosti koju koriste.
- NoSQL baza podataka - baze podataka korisne su kada se radi s velikim zbirkama distribuiranih podataka. Omogućuju bolje izvođenje velikih količina podataka u odnosu na relacijske baze podataka. Također imaju dobre performanse u analiziranje velikih nestrukturiranih skupova podataka i podataka na virtualnim poslužiteljima u oblaku. Mogu se nazvati i nerelacijskim bazama podataka.

- Objektno orijentirana baze podataka - baze podataka sadrže podatke stvorene pomoću objektno orijentiranih programskih jezika. Usredotočene su na organiziranje objekata, a ne na podatke i logiku.

Iako postoje mnoge vrste baza podataka, svaka baza podataka, neovisno o vrsti, sastoji se od pet osnovnih komponenti:

- Hardver - fizički uređaj na kojem radi softver baze podataka. Hardver baze podataka uključuje računala, poslužitelje i tvrde diskove.
- Softver - aplikacija za baze podataka koja omogućuje upravljanje bazom podataka.
- Podaci – informacije koje baza podataka pohranjuje. Administratori baza podataka organiziraju podatke kako bi bili što smisleniji.
- Jezik baze podataka - programski jezik s kojim se upravlja bazom podataka. Jedan od najčešćih jezika baze podataka je SQL.
- Procedure – pravila koja određuju kako baza podataka radi i kako postupa s podacima.

Prema Lutkevichu (2021), glavne karakteristike dobre baze podataka su:

- Sigurnost podataka - potrebna je jer su podaci vrijedna poslovna imovina. Zaštita baze podataka zahtijeva stručnjake za kibernetičku sigurnost.
- Integritet podataka - osigurava pouzdanost podataka. Potrebno je ograničiti pristup bazama podataka samo onima koji su kvalificirani za rukovanje njima.
- Učinkovitost baze podataka - zahtijeva redovito ažuriranje i održavanje baze podataka. Bez odgovarajuće podrške, funkcionalnost baze podataka može se smanjiti kako se mijenja tehnologija koja podržava bazu podataka ili kako se mijenjaju podaci koje sadrži.
- Integracija baze podataka - može uključivati integraciju izvora podataka iz različitih vrsta baza podataka i struktura u jednu bazu podataka ili u podatkovna jezera i skladišta podataka.

Za definiranje logičke strukture baze podataka potreban je niz pravila koja nazivamo model podataka. Prema Mangeru (2012), model podataka je skup pravila koja određuju kako sve može izgledati logička struktura baze podataka. Model čini osnovu za oblikovanje i implementiranje baze, odnosno logička organizacija podataka u bazi mora biti u skladu sa zahtjevima sustava za upravljanje bazom podataka. Postoje mnogi modeli podataka, a Manger (2012.) navodi slijedeće:

- Relacijski model - osmišljen je na temelju matematičkog pojma relacije. Podaci se nalaze u tablicama s redcima i stupcima.
- Mrežni model - baza je prikazana kao mreža s čvorovima i usmjerenim lukova. Čvorovi predstavljaju tipove zapisa, a lukovi definiraju veze među tipovima zapisa.
- Hijerarhijski model - izvedenica mrežnog modela u kojoj je baza prikazana stablom 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.

2.1 Sustav za upravljanje bazom podataka

Softver koji se koristi za posredovanje između baze podataka i aplikacije naziva se sustav za upravljanje bazom podataka (engl. *Database Management System*, *DBMS*). On je poslužitelj (engl. *Server*) baze podataka koji oblikuje fizički prikaz baze prema traženoj logičkoj strukturi. Njime se omogućavaju takozvane CRUD operacije: upisivanje (engl. *Create*), čitanje (engl. *Read*), promjena (engl. *Update*) i brisanje (engl. *Delete*) podataka. DBMS pruža centralizirani prikaz podataka kojima može pristupiti više korisnika s više lokacija na kontroliran način. Njime se mogu ograničiti podaci koje krajnji korisnici vide te njihov prikaz, pri tome pružajući više prikaza iste sheme baze podataka. DBMS pruža i logičku i fizičku neovisnost podataka kako bi zaštitio korisnike i aplikacije od saznanja da znaju gdje su podaci pohranjeni. Sve dok programi koriste sučelje za programiranje aplikacija (engl. *Application programming interface, API*) za bazu podataka koju pruža DBMS, programeri nemaju potrebu modificirati programe samo zato što su napravljene promjene u bazi podataka.

Slično kao i operacijski sustav, DBMS spada u temeljni softver koji većina korisnika i organizacija ne razvija samostalno, već ga kupuju ili unajmljuju kao gotov proizvod. Prema Mangeru (2012), najkorišteniji sustavi za upravljanje bazom podataka su:

- DB2 - proizvod tvrtke IBM, namijenjen prije svega velikim *mainframe*-računalima.

- Oracle - također proizvod IBM-a, koristi se na svim važnijim računalnim platformama poput UNIX-a, Linux-a i Windows-a.
- MS SQL Server - proizvod tvrtke Microsoft, namijenjen poslužiteljima s operacijskim sustavima Windows.
- MySQL - proizvod tvrtke MySQL AB, koristi se na raznim platformama, najčešće kao podrška web-aplikacijama. Jedini je besplatan od nabrojenih.

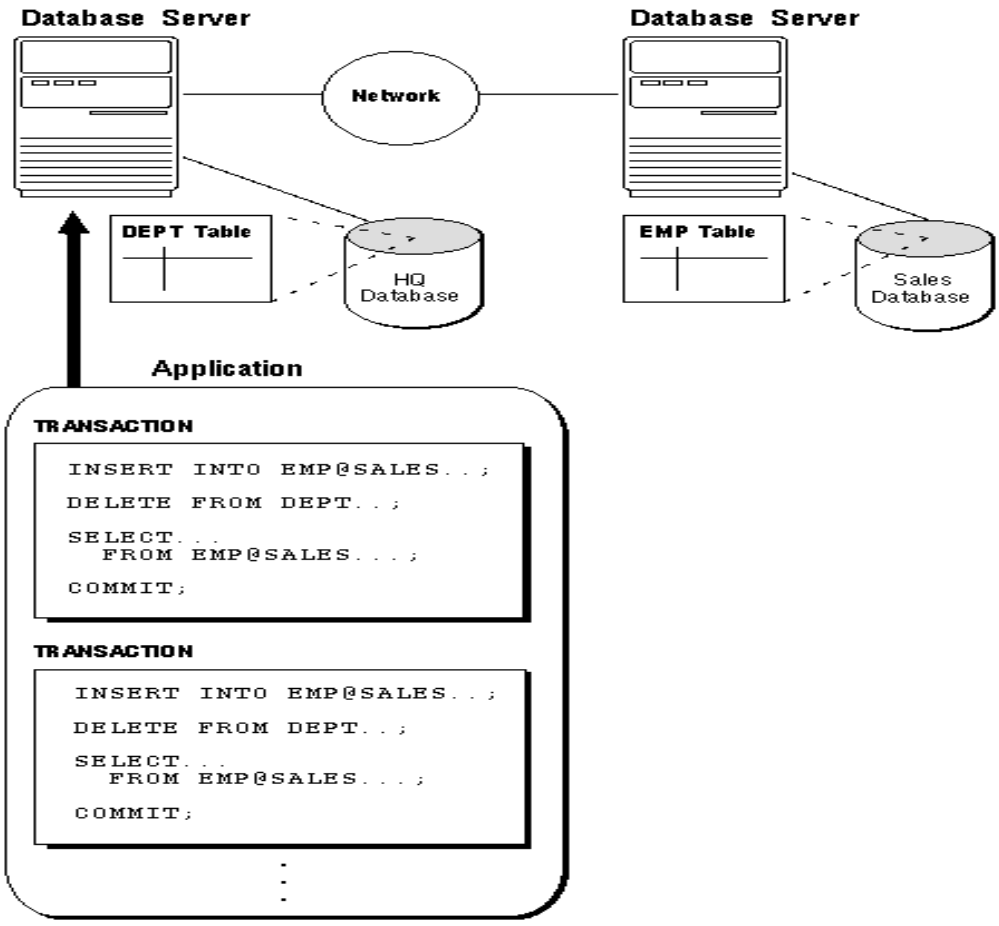
3. Distribuirane baze podataka

Distribuirana baza podataka je baza podataka koja se sastoji od dvije ili više datoteka smještenih na različitim mjestima, bilo na istoj ili na potpuno različitim mrežama. Dijelovi baze podataka pohranjeni su na više fizičkih lokacija, a obrada podataka se izvršava na više čvorova baze podataka. Oracle (n.d.) navodi da distribuirana baza podataka korisniku izgleda kao jedna baza podataka, ali je zapravo skup baza podataka pohranjenih na više računala. Podacima na nekoliko računala moguće je istovremeno pristupati i mijenjati ih putem mreže.

Kada su u zbirci, distribuirane baze podataka međusobno su logično povezane i često predstavljaju jednu logičku bazu podataka. S distribuiranim bazama podataka podaci se fizički pohranjuju na više mjesta i s njima se upravlja neovisno. Procesori na svakom mjestu povezani su mrežom i nemaju višeprocesnu konfiguraciju.

Za upravljanje distribuiranom bazom podataka koristi se centralizirani distribuirani sustav za upravljanje bazom podataka (engl. *distributed database management system*, DDBMS) koji logički integrira podatke tako da se njima može upravljati kao da su svi pohranjeni na istoj lokaciji. DDBMS intervalno sinkronizira sve podatke i osigurava da će izmjene poput ažuriranja i brisanja podataka na jednom mjestu automatski ažurirati podatke pohranjene na drugoj lokaciji. Svaki poslužitelj baze podataka u distribuiranoj bazi podataka kontrolira njegov lokalni DBMS i svaki od njih surađuje kako bi održao konzistentnost globalne baze podataka.

Dakle, poslužitelj baze podataka je softver koji upravlja bazom podataka, a klijent je aplikacija koja traži informacije od poslužitelja. U sustavu distribuirane baze podataka čvor predstavlja računalo, a može biti klijent, poslužitelj ili oboje. Na primjer, na slici ispod, računalo koje upravlja HQ bazom podataka djeluje kao poslužitelj baze podataka kada se izda izvještaj za njegove podatke i ponaša se kao klijent kada izdaje izvještaj za udaljene podatke (npr. izvještaj u svakoj transakciji izdaje se za udaljenu tablicu EMP u bazi podataka SALES).



Slika 1 - Primjer komunikacije poslužitelja i klijenta[1]

Klijent se može spojiti izravno ili neizravno na poslužitelj baze podataka. Na slici iznad, kada klijentska aplikacija izdaje izvještaj za svaku transakciju, klijent je povezan izravno na posredničku HQ bazu podataka i neizravno na SALES bazu podataka koja sadrži udaljene podatke. Među opće značajke distribuiranih baza podataka Marijan (2021) ubraja:

- Neovisnost o lokaciji - Podaci se fizički pohranjuju na više mjesta i njima upravlja neovisni DDBMS.
- Distribuirana obrada upita - Distribuirane baze podataka odgovaraju na upite u distribuiranom okruženju koje upravlja podacima na više mjesta. Upiti visoke razine pretvaraju se u plan izvršenja upita radi jednostavnijeg upravljanja.

[1] preuzeto s:https://docs.oracle.com/cd/A57673_01/DOC/server/doc/SCN73/image070.gif

- Distribuirano upravljanje transakcijama - Pruža dosljednu distribuiranu bazu podataka kroz protokole predaje, distribuirane tehnike kontrole konkurentnosti i distribuirane metode oporavka u slučaju mnogih transakcija i kvarova.
- Integracija na visokom nivou - Baze podataka u zbirci obično predstavljaju jednu logičku bazu podataka i međusobno su povezane.
- Mrežno povezivanje - Sve baze podataka u zbirci povezane su mrežom i međusobno komuniciraju.
- Obrada transakcija - Distribuirane baze podataka uključuju obradu transakcija, to je program koji uključuje skup jedne ili više operacija baze podataka. Obrada transakcija je atomski proces koji se ili u potpunosti izvršava ili se uopće ne izvršava.

Uz navedene značajke važno je spomenuti i transparentnost koju je moguće smatrati i kao jednu od prednosti distribuiranih baza podataka. Naime, funkcionalnost sustava distribuirane baze podataka mora biti osigurana na način da je složenost baze podataka transparentna i korisnicima i administratorima. Distribuirani sustav baze podataka trebao bi sakriti fizičke lokacije objekata u sustavu od aplikacija i korisnika. Transparentnost lokacije postoji ako se korisnik može referirati na istu tablicu na isti način, bez obzira na koji se čvor korisnik povezuje. Oracle (n.d) navodi transparentnost lokacije kao jednu od važnijih iz sljedećih razloga:

- Pristup udaljenim podacima je pojednostavljen, jer korisnici baze podataka ne moraju znati lokaciju objekata.
- Objekti se mogu premještati bez utjecaja na krajnje korisnike ili aplikacije baze podataka.

Potrebno je osigurati više razina transparentnosti, a detaljnije je opisano u poglavlju o DBMS-u.

Oracle (n.d.) ističe još i autonomiju stranice (engl. *Site Autonomy).* Njome se omogućava neovisno upravljanje poslužiteljem koji se nalazi u distribuiranoj bazi podataka (zbog sigurnosnih operacija). Time se postiže da se svakom bazom u distribuiranoj bazi podataka može upravljati na način kao da je riječ o centraliziranoj, nedistributiranoj bazi podataka. Iako baze podataka rade sinkrono kao jedna, one su zasebne baze i njima se upravlja zasebno. Kao prednosti autonomije stranice navodi se slijedeće:

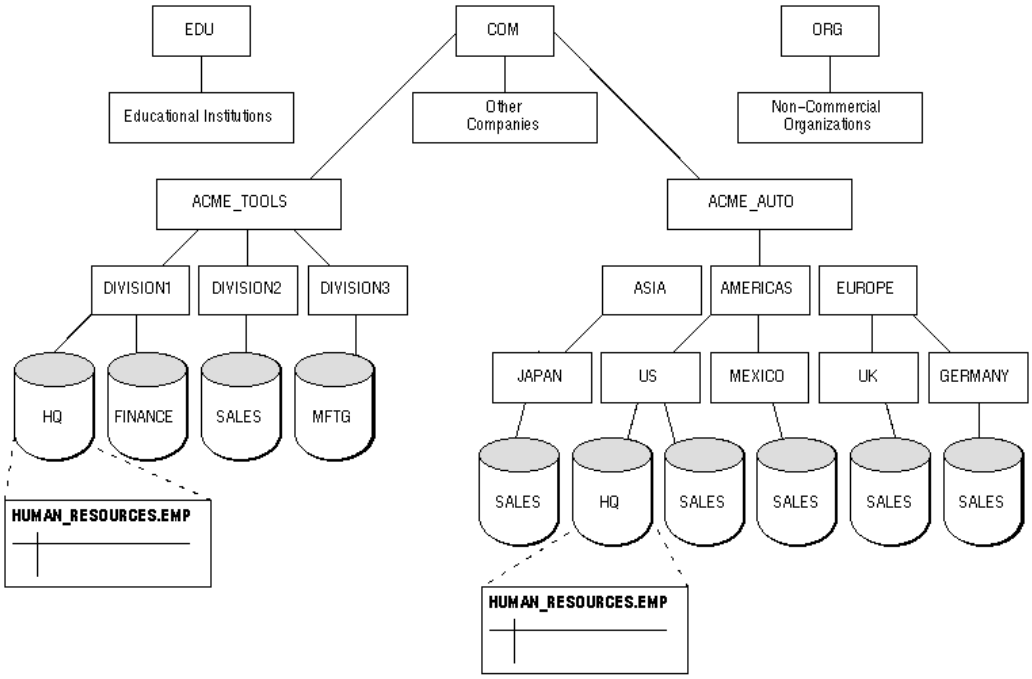
- Čvorovi sustava mogu odražavati logičnu organizaciju tvrtki trebaju održavati odnos "nadohvat ruke".

- Lokalne podatke kontrolira lokalni administrator baze podataka. Stoga je domena odgovornosti svakog administratora baze podataka manja i lakša za upravljanje.
- Smanjena je mogućnost da će kvarovi poremetiti druge čvorove distribuirane baze podataka. Globalna baza podataka djelomično je dostupna sve dok su dostupna jedna baza podataka i mreža; nijedan pojedinačni kvar baze podataka ne mora zaustaviti sve globalne operacije ili dovesti do usporenja u radu baze.
- Otklanjanje kvara izvodi se na bazi pojedinačnog čvora.
- Za svaku lokalnu bazu podataka postoji rječnik podataka.
- Čvorovi mogu samostalno nadograditi softver.

Kada je riječ o objektu sheme kao što je tablica, on je u distribuiranim bazama podataka dostupan iz svih čvorova koji je tvore. Stoga, distribuirani DBMS mora koristiti shemu imenovanja koja osigurava da objekti u cijeloj distribuiranoj bazi podataka mogu biti jedinstveno identificirani i referencirani baš kao što centralizirana lokalna DBMS arhitektura mora osigurati nedvosmislenu shemu imenovanja za jasno referenciranje objekata unutar lokalne baze podataka.

Kako bi se raspoznale reference na objekte unutar jedne baze podataka provodi se proces kojeg nazivamo razlučivost imena (engl. *name resolution)*. U njemu DBMS oblikuje imena objekata koristeći hijerarhijski pristup. To znači da unutar baze podataka, DBMS osigurava da svaka shema ima jedinstveni naziv te da unutar sheme svaki objekt također ima jedinstven naziv. Budući da se jedinstvenost provodi na svakoj razini hijerarhijske strukture, objekt na lokalnoj razini ima osiguran jedinstven naziv unutar baze podataka, a reference na lokalni naziv objekta mogu se lako razriješiti (Oracle, n.d.).

Distribuirani sustavi upravljanja bazama podataka proširuju hijerarhijski model imenovanja korištenjem jedinstvenih naziva baza podataka unutar mreže. Kao rezultat toga, zajamčeno je da je globalni naziv objekta jedinstven unutar distribuirane baze podataka, a reference na globalni naziv objekta mogu se razriješiti između čvorova sustava. Slika ispod prikazuje hijerarhijski raspored baza podataka u mreži.



Slika 2 - Primjer hijerarhijskog rasporeda baze podataka u mreži[2]

Kako bi se olakšale veze i komunikacija između pojedinačnih baza podataka distribuirane baze, koriste se veze baze podataka. Veza baze podataka definira put do udaljene baze podataka. Veze su transparentne jer je ime veze isto kao i globalni naziv baze podataka na koju veza upućuje.

Konkretno na primjeru iznad, slijedeći upit stvara vezu udaljene baze podataka u lokalnoj bazi podataka. Veza baze podataka pod nazivom SALES.DIVISION2.ACME_TOOLS.COM opisuje put do udaljene baze podataka istog imena.(Oracle, n.d.)

```
CREATE PUBLIC DATABASE LINK sales.division2.acme_tools.com ... ;
```

Nakon stvaranja ovakve veze, svaka aplikacija ili korisnik spojen na lokalnu bazu podataka može pristupiti podacima u bazi podataka SALES korištenjem naziva globalnih objekata.

[2] preuzeto s: https://docs.oracle.com/cd/A57673_01/DOC/server/doc/SCN73/image071.gif

3.1 Prednosti distribuiranih baza podataka

Marijan (2021) kao prednosti distribuiranih baza podataka navodi slijedeće:

- Modularni razvoj
- Pouzdanost
- Manji troškovi
- Poboljšane performanse

Modularni razvoj distribuirane baze podataka podrazumijeva da se sustav može proširiti na nove lokacije ili jedinice dodavanjem novih poslužitelja i podataka u postojeću bazu te se povezivanje s distribuiranim sustavom može neometano izvesti. Takva proširenja ne uzrokuju poteškoće u radu distribuiranih baza podataka.

Kada je riječ o pouzdanosti distribuirane baze podataka nude veću pouzdanost u odnosu na centralizirane baze podataka. U distribuiranoj bazi podataka u slučaju kvara baze podataka, sustav nastavlja s radom, ali samo isporučuje smanjene performanse dok se problem ne riješi, dok se u centraliziranoj bazi podataka sustav potpuno zaustavlja.

Lokalno pohranjivanje podataka smanjuje troškove komunikacije za manipulaciju podacima u distribuiranim bazama podataka. Za razliku od distribuiranih baza podataka, lokalna pohrana podataka nije moguća u centraliziranim bazama podataka.

Distribucija podataka u sustavu distribuirane baze podataka omogućuje brži odgovor kada se zahtjevi korisnika izvrše lokalno što dovodi do poboljšanih performansi. U centraliziranim bazama podataka korisnički zahtjevi prolaze kroz središnju jedinicu koji obrađuje sve zahtjeve. Rezultat je povećanje vremena odgovora.

S druge strane, Özsu i Valduriez (2011) ističu da se sve prednosti mogu svesti na četiri temelja tehnologije distribuiranih baza podataka. Uz poboljšane performanse, pouzdanost i jednostavnije proširenje sustava, navode i transparentno upravljanje distribuiranim i repliciranim podacima. Transparentnost se odnosi na odvajanje semantike sustava više razine od pitanja implementacije niže razine. Drugim riječima, transparentan sustav "skriva" detalje implementacije od korisnika. Prednost potpuno transparentnog DBMS-a je visoka razina podrške koju pruža za razvoj složenih aplikacija. Detaljnije o transparentnosti u poglavlju o DBMS-u.

Vrijedi istaknuti još lokalnu kontrolu koja omogućava da se podaci distribuiraju na način da je svaki njihov dio lokalan za neke stranice (poslužitelje), a stranica na kojoj je pohranjen dio

podataka je vlasnik podataka. Postoji i usklađenost s ACID svojstvima jer distribuirane transakcije zahtijevaju atomičnost, dosljednost, izolaciju i pouzdanost, a moguće je postići poboljšanu izvedbu i paralelizam u izvršavanju transakcija.

3.2 Nedostaci distribuiranih baza podataka

Nedostaci sustava baza podataka poprimaju dodatnu složenost u distribuiranom okruženju, iako su temeljni principi isti. Tri su glavna faktora koji utječu na složenost distribuiranog okruženja:

- Repliciranje podataka u distribuiranom okruženju – Dizajn distribuirane baze podataka može biti takav da se cijela baza ili dijelovi nalaze na različitim mjestima mreže. Dupliciranje podataka moguće je zbog razmatranja pouzdanosti i učinkovitosti, a zbog toga sustav distribuirane baze podataka je odgovoran za odabir jedne od pohranjenih kopija traženih podataka za pristup u slučaju dohvaćanja i da se ažuriranje odvija na svakoj kopiji podataka.
- Zakazivanje lokacija ili komunikacijskih veza tijekom ažuriranja - sustav mora osigurati da će sva ažuriranja izvršiti na nedostupnim podacima čim se sustav oporavi od kvara.
- Sporija sinkronizacija transakcija - stranica nema informacije o radnjama koje se trenutno provode na drugim stranicama pa sinkronizacija transakcija na više stranica je znatno kompliciranija nego za centralizirani sustav.

Navedeni faktori ukazuju na potencijalne probleme s distribuiranim DBMS-ovima, dok su prema Marijanu (2021) neki od nedostataka distribuiranih baza podataka:

- Povećani troškovi repliciranja resursa - Brojne operacije na više mjesta zahtijevaju izračune i stalnu sinkronizaciju kada se koristi replikacija baze podataka, što uzrokuje opterećenje obrade.
- Upravljanje distribucijom - Odaziv na zahtjeve korisnika ovisi o pravilnoj distribuciji podataka, što znači da se odaziv može smanjiti ako podaci nisu pravilno raspoređeni na više stranica.
- Sigurnosna pitanja – U distribuiranom DBMS-u, za razliku od centraliziranog sustava gdje je jednostavno kontrolirati pristup podacima, zbog repliciranih podataka pristup se mora kontrolirati na više lokacija, a potrebna je i sigurnost mreže. Tijekom povijest mreže su se

smatrale nesigurnim komunikacijskim medijem, ali u današnje vrijeme razina sigurnosti je viša i učinjen je napredak kako bi mreže postale sigurnije.

- Cijena - Osiguravanje transparentnosti podataka i koordinacije na više mjesta često zahtijeva korištenje skupog softvera u sustavu distribuirane baze podataka.
- Integritet podataka - Mogući problem pri korištenju replikacije baze podataka je integritet podataka, koji je ugrožen ažuriranjem podataka na više mjesta. Za upotrebu ograničenja integriteta potrebna je velika količina podataka koji definiraju to ograničenje, ali se ne koriste za ažuriranje samih operacija.

Potrebno je još spomenuti i mogućnost da se neke poruke i podaci mogu izgubiti u mreži tijekom prelaska s jednog čvora na drugi te da je baza podataka povezana s distribuiranim sustavima prilično je kompliciranа i teška za rukovanje u usporedbi s centraliziranim sustavom. Također, u mreži može doći do preopterećenja ako svi čvorovi distribuiranog sustava pokušaju slati podatke odjednom. Postoji i problem nedostatka standarda i iskustva. Nedostatak standarda značajno ograničava potencijal distribuiranih DBMS-ova, kao i izostanak alata ili metodologija koje bi omogućile pretvorbu centraliziranih DBMS u distribuirani DBMS. Tek u novije vrijeme dolazi do pojave standardne komunikacije i protokola za pristup podacima. Distribuirani DBMS nisu široko prihvaćeni, iako su mnogi protokoli i problemi razumljivi. Posljedično, još ne postoji ista razina iskustva u industriji kao s centraliziranim DBMS-ovima.

4. Usporedba distribuiranih i centraliziranih baza podataka

Centralizirane baze podataka sadrže podatke na jednom računalu, lokaciji, a za pristup informacijama potrebno je pristupiti glavnom računalu sustava, poznatom kao poslužitelj.

S druge strane, distribuirana baza podataka funkcionira kao jedna logička podatkovna mreža, postavljena u nizu računala (čvorova) smještenih na različitim geografskim lokacijama koja su međusobno povezana kako bi se pružila cjelovitost i dostupnost informacija s bilo kojeg mjesta. Svi čvorovi sadrže informacije i svi su klijenti sustava. Kada je riječ o razlici između distribuiranih i centraliziranih baza podataka, usporedbu je najjednostavnije prikazati komparirajući različite karakteristike baza podataka kao što su vrijeme pristupa, upravljanje podacima, troškovi, učinkovitost i slično, a osnovna razlika je način pohrane podataka. Dakle, kao što je već navedeno, kod centraliziranih baza podaci se nalaze na jednom mjestu, dok se kod distribuirane baze podataka podaci nalaze na više fizičkih lokacije raspoređeni u međusobno povezane baze podataka. Sakshi(2022) navodi usporedbu po nekoliko kategorija:

Vrijeme potrebno za pristup podacima ukoliko im pristupa više korisnika je duže u centraliziranoj bazi podataka, odnosno kraće u distribuiranoj bazi podataka. Zbog više lokacija u distribuiranim bazama, podacima se pristupa brže.

Upravljanje, modificiranje i sigurnosno kopiranje centralizirane baze podataka je jednostavnije jer se svi podaci nalaze na istoj lokaciji. Za razliku od centralizirane baze, iste operacije su znatno složenije za izvršavanje na distribuiranoj bazi zbog njezine složenosti i različitih fizičkih lokacija.

Centralizirana baza podataka pruža ujednačen i potpun pregled korisniku dok je korisniku teško pružiti ujednačen prikaz u distribuiranoj bazi. Centralizirane baze podataka imaju bolju dosljednost podataka u usporedbi s distribuiranom bazom podataka zbog replikacija podataka koje se mogu javiti u distribuiranim bazama.

Kod pojave kvarova na bazi podataka, prednost ima distribuirana baza podataka. Naime, zbog različitih lokacija podataka, ukoliko jedna baza podataka zakaže, korisnici imaju pristup drugim bazama podataka što znači da korisnici mogu pristupiti svim podacima koji ne nalaze na bazi s kvarom. U centraliziranoj bazi podataka korisnici ne mogu pristupiti podacima u slučaju kvara baze podataka sve dok se kvar ne otkloni.

Trošak postavljanja i opreme te održavanje su često vrlo važan faktor kod kreiranja novih baza podataka. Centralizirana baza podataka je znatno jeftinija od distribuirane baze jer zahtjeva znatno manje opreme i prostor te se nalazi na jednoj lokaciji. Održavanje je također jednostavnije zbog podataka i informacija dostupnih na jednom mjestu i stoga im je lakše pristupiti. Distribuirana baza zahtjeva veći napor za održavanje zbog distribucije podataka i informacija na različitim mjestima. Dakle, postoji i potreba za provjerom problema s redundancijom podataka i kako osigurati i održati dosljednost podataka.

Kada se govori o učinkovitosti, distribuirana baza podataka je učinkovitija od centralizirane baze podataka zbog podjele podataka na više mjesta što čini pronalaženje podataka jednostavnijim i manje vremenski zahtjevnim. Centralizirana baza podataka manje je učinkovita jer je pronalaženje podataka složeno zbog pohranjivanja podataka i informacija na određeno mjesto. Vrijedi spomenuti još i brzinu odziva koja je veća kod centralizirane baze u usporedbi s distribuiranom bazom podataka.

U tabličnom prikazu ispod navedene su glavne prednosti i nedostaci uspoređujući ove dvije vrste baza podataka.

Kako bi se odabrala baza prikladna za određenu upotrebu potrebno je dakle sagledati više segmenata kao što su cijene, praktičnost, performanse i sigurnost. Velika tvrtka može primijeniti oba rješenja za različite svrhe, ovisno o osjetljivosti i zahtjevima podataka.

Ako je za područje upotrebe potrebna visoka učinkovitost i brzi rezultati, distribuirano rješenje je bolje, čak i ako iziskuje veće troškove. Njime se sprječavaju uska grla i osiguravaju bolje redundancije. Ukoliko je važnija sigurnost ispred performansi, rješenja centralizirane baze podataka su bolja. One osiguravaju jednostavniju zaštitu podataka i svode rizik na minimum. Preporuka je dodatno ulaganje u rješenja za sigurnosno kopiranje i oporavak kako bi se zaštitili podaci od kvarova hardvera, nestanaka struje i sličnog

Dakle, iako obje vrste baze podataka imaju i svojih prednosti i nedostataka, kod odabira je najvažnije izvršiti kvalitetnu analizu kako bi se odabrala baza podataka s boljim performansama za tražene uvjete. Dobro planiranje smanjuje rizik od daljnjih neželjenih troškova i poteškoća pa je poznavanje prednosti i nedostataka ključni prvi korak.

Tablica 1 - Prednosti i nedostaci centraliziranih i distribuiranih baza podataka[3]

	CENTRALIZIRANE BAZE	DISTRIBUIRANE BAZE
PREDNOSTI	Integritet podataka	Visoke performanse zbog podjele radnog opterećenja.
	Sigurnost	Visoka dostupnost zbog spremnosti dostupnih čvorova za obavljanje posla.
	Jednostavan pristup svim informacijama	Neovisni čvorovi i bolja kontrola nad resursima
	Podaci su lako prenosivi	
NEDOSTACI	Pretraživanje podataka zahtijeva vrijeme	Prilično je velik i složen sustav pa ga je teško koristiti i održavati.
	U slučaju kvara centraliziranog poslužitelja, cijela baza podataka bit će izgubljena.	Teško je pružiti sigurnost
	Ako više korisnika pokuša pristupiti podacima u isto vrijeme, mogući su problemi.	Povećanje zahtjeva za pohranom i infrastrukturom
		Rukovanje kvarovima prilično je težak zadatak

Postoji razlika i između sustava za upravljanja centraliziranim bazama podataka i sustava za upravljanje distribuiranim bazama podataka. o čemu će biti riječ u jednom od slijedećih poglavlja.

[3] Preuzeto s: https://www.geeksforgeeks.org/difference-between-centralized-database-and-distributed-database/

5. Vrste distribuirane baze podataka

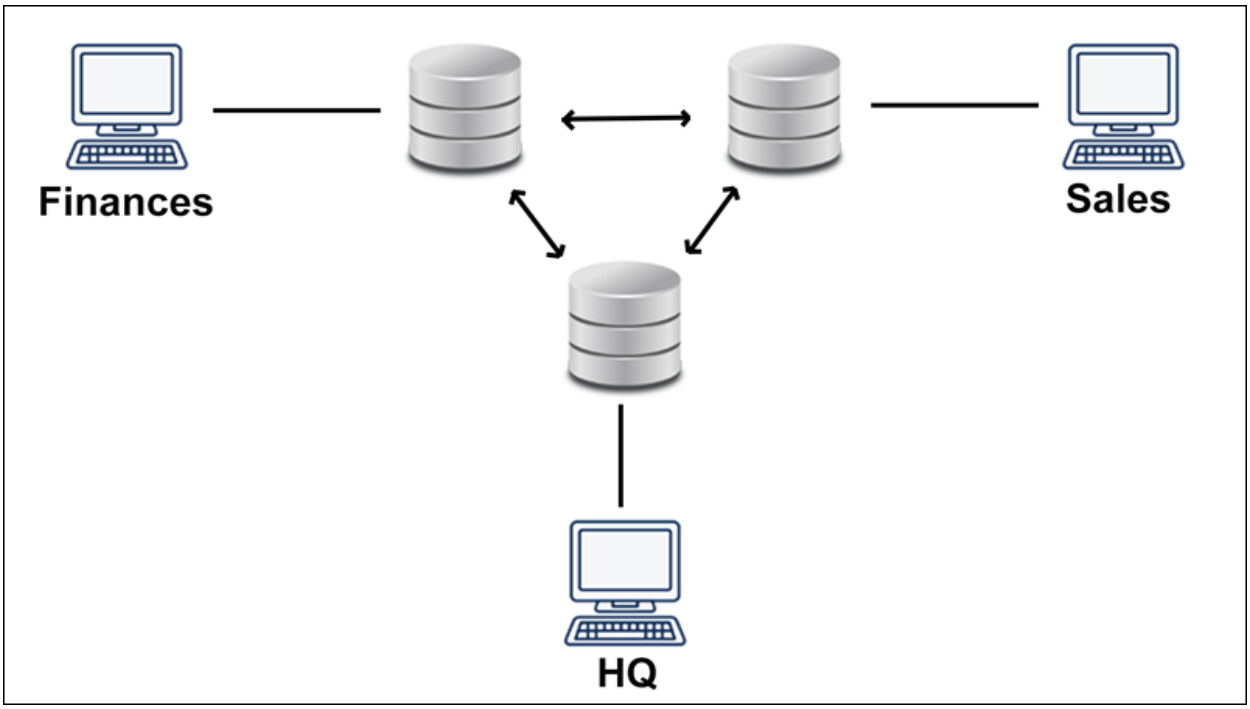
Distribuirane baze podataka mogu se podijeliti na dvije osnovne vrste, a to su **homogena** i **heterogena** okruženja baze podataka. Svaka od ovih vrsta se dalje dijeli na svoje podvrste. Homogene distribuirane baze podataka dijele se na autonomne i neautonomne distribuirane baze podataka, dok se heterogene dijele na federirane i *multidatabase* distribuirane baze podataka. (Shah, 2021)

U homogenoj bazi podataka svaka zasebna baza pohranjuje podatke na isti način. Također, operativni sustav i sustav za upravljanje bazom podataka kao i strukture podataka koje se koriste su identični što omogućuje jednostavnije upravljanje distribuiranom bazom podataka. Drugim riječima, ovakva baza je mreža identičnih baza podataka pohranjenih na više mjesta s istim operativnim sustavom, DDBMS-om i strukturom podataka. Svaka lokacija je svjesna svih drugih lokacija i surađuje s njima kako bi se obradio zahtjev korisnika, a samoj bazi podataka se pristupa preko jedinstvenog sučelja kao da se radi o jednoj bazi podataka. Ono omogućava istovremeno pristupanje ili modificiranje podataka u nekoliko baza podataka u istom distribuiranom okruženju pri čemu su lokacija i platforma baza podataka transparentni. Postoje razlike između pojedinih homogenih distribuiranih baza podataka u autonomiji. Autonomija se u ovom kontekstu odnosi na raspodjelu kontrole, a ne podataka. Ona označava stupanj do kojeg pojedinačni DBMS-ovi mogu raditi neovisno. Autonomija je funkcija brojnih čimbenika kao što su razmjena informacija u sastavnim sustavima, mogu li neovisno izvršavati transakcije i smije li ih se mijenjati. Zahtjevi autonomnog sustava specificirani su slijedećim (Özsu i Valduriez, 2011):

- Na lokalne operacije pojedinačnih DBMS-ova ne utječe njihovo sudjelovanje u distribuiranom sustavu.
- Način na koji pojedinačni DBMS obrađuju upite i optimiziraju ih ne bi trebao biti pod utjecajem izvršavanja globalnih upita koji pristupaju više baza podataka.
- Konzistentnost ili rad sustava ne bi trebali biti ugroženi kada se pojedinačni DBMS-ovi pridruže ili napuste distribuirani sustav.

U autonomnoj homogenoj bazi je svaka baza podataka neovisna i funkcionira samostalno. Integriraju se pomoću upravljačke aplikacije i koriste prijenos poruka za dijeljenje ažuriranja podataka. Kod neautonomnih homogenih distribuiranih baza podataka distribucija podataka se

odvija preko homogenih čvorova, a središnji ili glavni DBMS koordinira ažuriranje podataka na svim lokacijama.



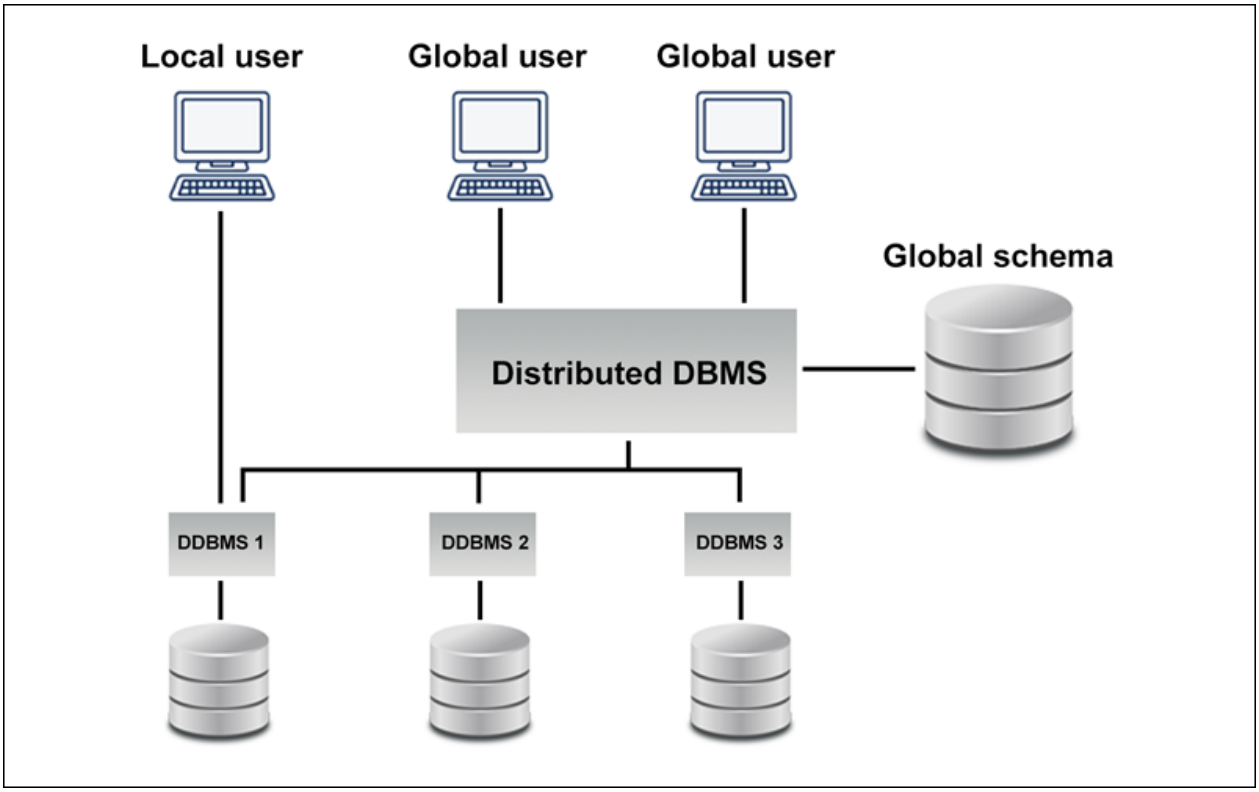
Slika 3 - Homogene distribuirane baze podataka[4]

Za razliku od homogene distribuirane baze, u heterogenoj distribuiranoj bazi podataka, različite lokacije imaju različite operativne sustave, DBMS-ove i modele podataka. Dakle, različite lokacije mogu koristiti različite sheme i softver pa može doći do problema u obradi upita i transakcijama. Također, moguće je da određena lokacija nema spoznaju o postojanju neke druge lokacije. Računala unutar heterogene baze podataka mogu koristiti različite operativne sustave i različite aplikacije baze podataka, a mogu čak koristiti različite modele podataka za bazu podataka. DBMS mogu biti različiti, pa tako na jednoj lokaciji DBMS može biti npr. relacijski, mrežni, dok će na drugoj lokaciji isti biti hijerarhijski ili objektno orijentirani. Kako bi komunikacija između pojedinih lokacija bila ostvariva potrebni su prijevodi, a obrada upita je zbog različitih shema složenija. Također, zbog različitih softvera, i obrada transakcija je složenija.

Razlikujemo federirane i *multidatabase* heterogene distribuirane baze podataka. Temeljna razlika odnosi se na definiciju globalne konceptualne sheme. U slučaju federirane, globalna shema

[4] preuzeto s: https://phoenixnap.com/kb/wp-content/uploads/2021/05/heterogeneous-database.png

definira pogled na cijelu bazu podataka, dok u slučaju distribuiranih multidatabaseova, ona predstavlja samo skup nekih od lokalnih baza podataka koje svaki lokalni DBMS želi dijeliti. U potonjoj je globalna baza podataka jednaka uniji lokalnih baza podataka. Dakle, u federiranim bazama podataka heterogeni sustavi baza podataka neovisni su po prirodi i međusobno integrirani tako da funkcioniraju kao jedan sustav baza podataka, dok kod *multidatabase* baza podataka sustavi baza podataka koriste središnji koordinirajući modul preko kojeg se pristupa bazama podataka.



Slika 4 - Heterogene distribuirane baze podataka[5]

[5] preuzeto s: https://phoenixnap.com/kb/wp-content/uploads/2021/05/heterogeneous-database.png

6. Distribuirani sustav za upravljanje bazom podataka

U distribuiranoj bazi podataka podaci se dijele ili repliciraju između nekoliko baza podataka koje su fizički odvojene jedna od druge. Te su baze podataka povezane putem mreže tako da se korisniku prikazuju kao jedna baza podataka. Upotrebom softverskog sustava za upravljanje distribuiranom bazom podataka (engl. *Distributed Database Management System*, DDBMS) omogućava se upravljanje distribuiranom bazom podataka kako bi se korisnicima prikazala kao jedna baza podataka. Svaka baza podataka u distribuiranom sustavu ima svoj zasebni DBMS softver.

Distribuirani sustav za upravljanje bazom podataka se sastoji od logičke baze podataka koja je podijeljena na više fragmenata, a svaki od fragmenata je pohranjen na jednom ili više računala. Računala su zasebno pod kontrolom DBMS-a i povezana su na mrežu, a kako bi se izvršio upit, DDBMS upit izvodi na više lokalnih baza podataka i rezultati se objedinjuju te se stvara dojam jedinstvene baze podataka. Drugi riječima, to je softver koji pomaže u izradi distribuirane baze podataka i pruža pristup koji pomaže u izradi transparentne distribucije za korisnika. Korisnik pristupa distribuiranoj bazi podataka putem lokalnih i globalnih aplikacija. Lokalne aplikacije ne zahtijevaju podatke s drugih stranica, dok globalne zahtijevaju. Potrebno je da DDBMS ima barem jednu globalnu aplikaciju.

Kako bi se točno odredilo što je DDBMS, postoji dvanaest Dateovih (2003) pravila koja ga precizno definiraju. Pravila su sljedeća:

- Lokalna autonomija – Svaki fragment ima vlastite operacije i djeluje kao neovisni autonomni centralizirani DBMS. Svaki fragment je zasebno zadužen za sigurnost, sigurnosno kopiranje i oporavak.
- Neovisnost središnjeg fragmenta – Svi su fragmenti jednaki i niti jedan ne ovisi o središnjem fragmentu za obavljanje bilo koje zadaće. Drugim riječima, ne postoji fragment bez kojeg sustav ne radi. Upravljanje transakcijama, optimizacija upita, otkrivanje zastoja i upravljanje globalnim sistemskim katalogom moguće je bez središnjeg poslužitelja.
- Kontinuirani rad – nema utjecaja kvarova na sustav. Sustav nastavlja svoj rad čak i u slučaju kvara fragmenta ili bilo kakvog širenja mreže.

- Lokalna neovisnost – Za dohvaćanje bilo kakvih podataka u sustavu, potrebno je znati samo gdje su podaci pohranjeni.
- Neovisnost o fragmentaciji – Korisnik može vidjeti samo jednu logičku bazu podataka. Postoji transparentnost fragmentacije podataka za korisnika. Da bi se dohvatio bilo koji fragment baze podataka, nije potrebno znati naziv fragmenata baze podataka.
- Neovisnost replikacije – Podaci se mogu replicirati i pohraniti na različitim mjestima. DDBMS upravlja svim fragmentima transparentno korisniku.
- Neovisnost distribuiranog upita – Za izvršavanje jednog upita na različitoj lokaciji, DDBMS ne može zadovoljiti transparentan zahtjev. Stoga je optimizacija upita ključna i DDBMS je izvodi transparentno.
- Neovisnost distribuirane transakcije – Transakcija može transparentno ažurirati podatke na različitim mjestima, ali se kontrola oporavka postiže korištenjem agenata.
- Neovisnost o hardveru – DDBMS radi na različitim hardverskim platformama.
- Neovisnost operativnog sustava – DDBMS radi na različitim platformama operativnih sustava.
- Neovisnost o mreži – DDBMS sustav može raditi na bilo kojoj mrežnoj platformi.
- Neovisnost baze podataka – Sustav mora podržavati bilo kojeg dobavljača proizvoda baze podataka.

Kada je riječ o centraliziranim DBMS-ima mogu se istaknuti dvije osnovne karakteristike, pružanje potpunog pregleda podataka te jednostavnije upravljanje u odnosu na distribuirane sustave za upravljanje. Distribuirani sustavi za upravljanje moraju pružiti funkcije kao i centralizirani, uz dodatak slijedećih funkcija koje ga čine distribuiranim:

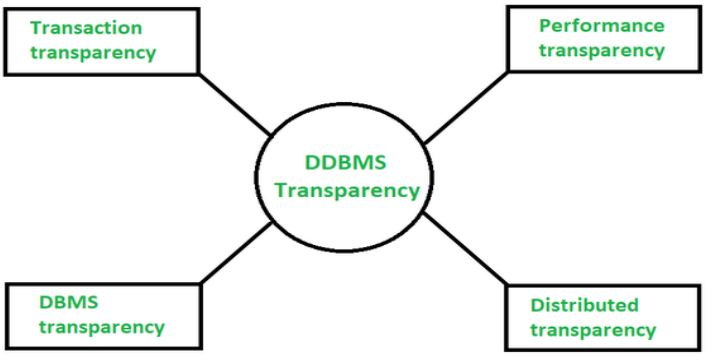
- Praćenje podataka – Osnovna funkcija DDBMS-a je praćenje distribucije podataka, fragmentacije i replikacije.
- Distribuirana obrada upita –DDBMS omogućava pristup udaljenim lokacijama i osigurava prijenos upita i podataka između lokacija putem komunikacijske mreže.
- Upravljanje repliciranim podacima – sustav odlučuje kojoj kopiji repliciranih podataka treba pristupiti te održava dosljednost kopija.

- Oporavak distribuirane baze podataka – Mogućnost oporavka od pada pojedinačnih fragmenata uslijed nastalih kvarova.
- Sigurnost – DDBMS izvršava distribuirane transakcije uz osiguravanje sigurnosti podataka i autorizacijom/ovlastima pristupa korisnika.
- Distribuirano upravljanje transakcijama – DDBMS ima sposobnost izvršavanja za upite i transakcije koje pristupaju podacima s više od jednog mjesta te sinkronizira pristup distribuiranim podacima i održava integritet baze podataka.

Dakle, iz navedenih karakteristika može se zaključiti da korištenjem distribucije dolazi do povećane složenosti u dizajnu i implementaciji sustava, ali se postižu prednosti poput mrežne transparentnosti i poboljšanih performansi.

Još jedna od važnih karakteristika DDBMS-a je sposobnost da distribuciju učini transparentnom, odnosno nevidljivom korisniku. Dakle, činjenica da je distribuirana baza podataka podijeljena na fragmente koji se mogu pohraniti na različitim računalima i potencijalno replicirati, treba biti nedostupna korisniku. Cilj transparentnosti je učiniti da distribuirani sustav izgleda kao centralizirani sustav. U sustavu za upravljanje distribuiranom bazom podataka postoje četiri vrste transparentnosti, a to su (Singh, 2021):

- Transparentnost transakcija
- Transparentnost izvedbe
- Transparentnost DBMS-a
- Transparentnost distribucije



Slika 5 - Transparentnost DDBMS-a[6]

[6] preuzeto s: https://media.geeksforgeeks.org/wp-content/uploads/20210616193256/12-660x402.PNG

Transparentnost transakcija osigurava da sve transakcije koje se distribuiraju očuvaju dosljednosti i integritet distribuirane baze podataka. Podaci koji se pohranjuju distribucijskim transakcijama pohranjeni su na više lokacija. Svaka transakcija je podijeljena u manje transakcije, a DDBMS osigurava njihovu atomičnost, odnosno izvršavanje manjih dijelova kako bi se izvršila cijela transakcija.

Transparentnost izvedbe omogućava da DDBMS radi kao da je centralizirani sustav za upravljanje bazom podataka. Sustav ne bi trebao odstupati u performansama izvođenja jer je njegova arhitektura distribuirana. Isto tako, DDBMS mora imati distribuirani procesor upita koji može mapirati zahtjev za podacima u uređeni niz operacija na lokalnoj bazi podataka.

Transparentnost DBMS-a je primjenjiva samo na heterogene tipove DDBMS-a (baze podataka koje imaju različite stranice i koriste različite operativne sustave i modele podataka) jer skriva činjenicu da lokalni DBMS mogu biti različiti. Riječ je o jednoj od složenijih transparentnosti.

Transparentnost distribucije služi kako bi korisnik spoznao bazu podataka kao jedinstveni logički entitet. Ova transparentnost se dijeli na pet podvrsta: transparentnost fragmentacije, lokacije, replikacije, imenovanja i lokalnog mapiranja. Kod transparentnosti fragmentacije korisnik ne mora znati za fragmentirane podatke, pa se pristup bazi podataka temelji na globalnoj shemi. Vrlo je slično SQL pogledu, gdje korisnik koristi pogled tablice, a ne same tablice. Ukoliko transparentnost lokacije pruža DDBMS, potrebno je da korisnik zna kako su podaci fragmentirani, ali nije potrebno znati gdje se podaci nalaze, odnosno njihovu lokaciju. Transparentnost replikacije povezana je s transparentnošću istovremenosti i transparentnošću neuspjeha. Kad god korisnik modificira podatke, ažuriranje se odvija u svim kopijama tablice. Korisnik ne bi trebao biti upoznat s ovom operacijom. Kod transparentnosti lokalnog mapiranja, korisnik treba definirati nazive fragmenata i lokaciju podataka uzimajući u obzir sva moguća dupliciranja. Riječ je o vrlo zahtjevnom upitu za korisnika u DDBMS transparentnosti. Kod transparentnosti imenovanja svaka stavka u bazi podataka mora imati jedinstven naziv. Iz toga slijedi da DDBMS mora osigurati da dvije stranice ne stvaraju objekt baze podataka s istim nazivom. Moguće je stvoriti središnji poslužitelj naziva za stvaranje jedinstvenih naziva objekata u sustavu ili dodati objekt koji počinje s identifikatorom web mjesta gdje je naziv objekta stvoren.

6.1 Nedostaci distribuiranog sustava za upravljanje bazom podataka

Iako svojim karakteristikama distribuirani sustav za upravljanje bazom podataka omogućuje bržu obradu podataka i poboljšane performanse rada baze podataka, DDBMS ima i svojih nedostataka. Nedostaci se najviše odnosno na kompleksnost i troškove, ali i na pitanja sigurnosti i integriteta.

Kada je riječ o kompleksnosti distribuirane baze podataka su mreža mnogih računala prisutnih na različitim lokacijama i pružaju izvanrednu razinu performansi, dostupnosti i, naravno, pouzdanosti. Stoga je struktura distribuiranog DBMS-a relativno složenija od centraliziranog DBMS-a. Također, DDBMS ne osigurava replikaciju podataka, što povećava složenost sustava.

Troškovi poput troškova održavanja, nabave, hardvera te troškovi mreže/komunikacije i, troškovi su veći i čine distribuirani DBMS skupljim u odnosu na centralizirani DBMS.

Sigurnost podataka i mreže u distribuiranoj bazi podataka, uz održavanje redundantnosti podataka, glavni je problem. Mreža se može lako napasti te može doći do krađe podataka i zlouporabe.

U sustavu distribuirane baze podataka važno je održavanje dosljednosti podataka. Sve promjene podataka na jednom mjestu moraju se odraziti na sve stranice. Trošak komunikacije i obrade je visok u distribuiranom DBMS-u kako bi se osigurao integritet podataka.

Nedostatak standarda:

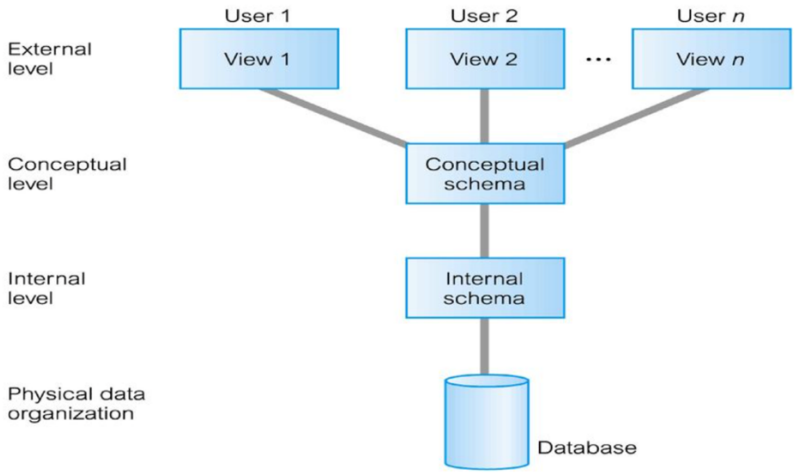
Iako pruža učinkovitu komunikaciju i dijeljenje podataka, ne postoje standardna pravila i protokoli za pretvaranje centraliziranog DBMS-a u distribuirani DBMS. Nedostatak standarda smanjuje puni potencijal distribuiranog DBMS-a. Također zbog nedostatka odgovarajućih komunikacijskih standarda, nije moguće povezati različitu opremu koju proizvode različiti dobavljači u mrežu kako bi ona fluidno funkcionirala.

7. Arhitektura sustava za upravljanja baze podataka

Arhitektura distribuiranog sustava definira njegovu strukturu. Drugim riječima, komponente sustava su identificirane, funkcija svake komponente je specificirana te su definirani međusobni odnosi i interakcije među tim komponentama. Specifikacija arhitekture sustava zahtijeva identifikaciju različitih modula, s njihovim sučeljima i međusobnim odnosima, u smislu protoka podataka i upravljanja kroz sustav.

7.1 ANSI/SPARC

Prvi standard razvijen je sedamdesetih godina prošlog stoljeća i naziva se ANSI/SPARC arhitektura. Odbor za računala i obradu informacija Američkog nacionalnog instituta za standarde (ANSI) osnovao je Studijsku grupu za baze podataka pod nazivom Sustavi upravljanja pod pokroviteljstvom Odbora za planiranje standarda i zahtjeve (SPARC) pa je tako i nastala kratica. Predložena je standardizacija sučelja i definiran arhitektonski okvir koji je sadržavao 43 sučelja, od kojih se 14 bavilo podsustavom fizičke pohrane računala. Izgled ANSI/SPARC arhitekture prikazan je na slici ispod. Tri su pogleda na podatke: vanjski pogled koji je predstavljao pogled krajnjeg korisnika, kao što je programer. Zatim unutarnji pogled, odnosno pogled na sustav te konceptualni pogled, odnosno pogled poduzeća. Za svaki od ovih pogleda potrebna je odgovarajuća definicija sheme.



Slika 6 - ANSI/SPARC Arhitektura[7]

[7] preuzeto s: https://www.techsofttutorials.com/wp-content/uploads/2020/10/ANSI-SPARC-architecture.png

Na slici 6 prikazana je struktura ANSI/SPARC arhitekture. Na najnižoj razini arhitekture nalazi se unutarnji pogled zadužen za fizičku definiciju i organizaciju podataka. Zadužen je za rješavanje lokacije podataka na različitim uređajima za pohranu i mehanizme pristupa koji se koriste za dohvaćanje i manipuliranje podacima.

Na središnjoj razini nalazi se konceptualna shema, koja je apstraktna definicija baze podataka. Predstavlja pogled na stvarnu situaciju poduzeća koje se modelira u bazi podataka, odnosno podatke i odnose među podacima.

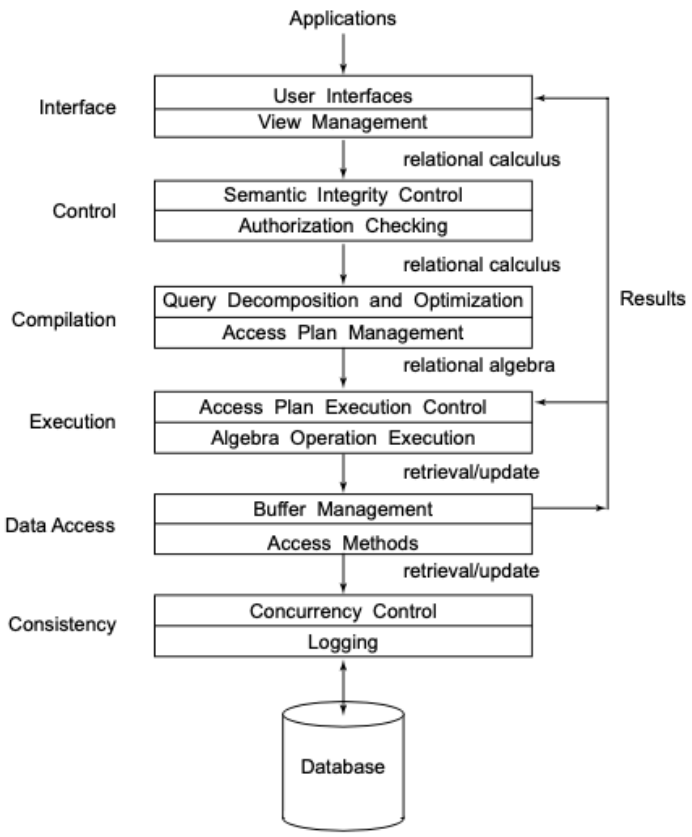
Na najvišoj razini je vanjski pogled, koji stvara prikaz kako korisnici vide bazu podataka. Pogled korisnika predstavlja dio baze podataka kojem će taj korisnik pristupiti, kao i odnose koje bi korisnik želio vidjeti među podacima. Pogled se može dijeliti između više korisnika, pri čemu zbirka korisničkih pogleda čini vanjsku shemu.

Ova je arhitektura je važna jer pruža osnovu za neovisnost podataka. Odvajanje vanjskih shema od konceptualne sheme omogućuje logičku neovisnost podataka, dok odvajanje konceptualne sheme od unutarnje sheme omogućuje fizičku neovisnost podataka.

7.2 Arhitektura centraliziranog DBMS-a

DBMS je softver koji dijeli više procesa (transakcija) koji pokreću programe baze podataka. Povezan je s dvije komponente: komunikacijskim podsustavom i operativnim sustavom. Komunikacijski podsustav dopušta povezivanje DBMS-a s drugim podsustavima radi komunikacije s aplikacijama, operativni sustav osigurava sučelje između DBMS-a i resursa računala (procesor, memorija, diskovi itd.).

Özsu i Valduriez (2011) navode funkcije koje izvodi DBMS mogu se slojevito rasporediti na sučelje, kontrolu, obradu, izvođenje, pristup podacima i upravljanje konzistentnosti.



[8]

Slika 7 - Slojevi centraliziranog DBMS-a

Na slici 7 su prikazani slojevi centraliziranog sustava za upravljanje bazom podataka. Sloj sučelja upravlja sučeljem u odnosu prema aplikacijama. Aplikacijski programi baze podataka izvode se prema vanjskim pogledima baze podataka. Pogled u relacijskim DBMS-ovima je virtualna relacija izvedena iz osnovnih relacija primjenom operacija relacijske algebre. Upravljanje pregledom sastoji se od prevođenja korisničkog upita iz vanjskih podataka u konceptualne podatke.

Kontrolni sloj kontrolira upit dodavanjem predikata semantičkog integriteta i predikata autorizacije. Izlaz ovog sloja je obogaćeni upit u jeziku visoke razine koji prihvaća sučelje.

Sloj obrade upita preslikava upit u optimizirani niz operacija niže razine te se bavi izvedbom. Rastavlja upit u stablo algebarskih operacija i pronalazi optimalan redoslijed operacija,

[8] preuzeto iz: Özsu M. T., Valduriez P. (2011), *Principles of Distributed Database Systems*, Third Edition, Springer, New York, str 24.

a rezultat se pohranjuje u planu pristupa. Izlaz ovog sloja je upit izražen kodom niže razine (algebarske operacije).

Sloj izvođenja zadužen je za upravljanje transakcijama i sinkronizaciju algebarskih operacija. Tumači relacijske operacije pozivanjem sloja pristupa podacima kroz zahtjeve za dohvaćanje i ažuriranje. Sloj pristupa podacima upravlja podatkovnim strukturama koje implementiraju datoteke, indekse, itd. Također upravlja međuspremnicima spremanjem u predmemoriju podataka kojima se najčešće pristupa.

Sloj konzistentnosti upravlja kontrolom istovremenosti i bilježenjem zahtjeva za ažuriranje. Ovaj sloj omogućuje oporavak transakcije, sustava i medija nakon kvara.

7.3 Arhitektura distribuiranog DBMS-a

Za distribuirani DBMS Özsu i Valduriez (2011) ističu da se koristi klasifikacija koja oblikuje sustav prema tri karakteristike, autonomija lokalnih sustava, distribuciju istih i njihovu heterogenost.

Autonomija se odnosi na raspodjelu kontrole, a ne podataka. Označava stupanj do kojeg pojedinačni DBMS-ovi mogu raditi neovisno. Predstavlja funkciju čimbenika kao što su razmjena informacija u sustavima, tj. pojedinačnim DBMS-ovima, mogu li neovisno izvršavati transakcije i smije li ih se mijenjati.

Zahtjevi autonomnog sustava definirani su:

1. Na lokalne operacije pojedinačnih DBMS-ova ne utječe njihovo sudjelovanje u distribuiranom sustavu.
2. Način na koji pojedinačni DBMS obrađuju upite i optimiziraju ih ne bi trebao biti pod utjecajem izvršavanja globalnih upita koji pristupaju više baza podataka.
3. Konzistentnost ili rad sustava ne bi trebali biti ugroženi kada se pojedinačni DBMS-ovi pridruže ili napuste distribuirani sustav.

Tri najpopularnije varijante za autonomne sustave su uska integracija, poluautonomni sustavi i potpuna izolacija.

Kod uske integracije, slika cijele baze podataka dostupna je svakom korisniku koji želi podijeliti informacije, a koje se mogu nalaziti u više baza podataka. Iz perspektive korisnika,

podaci su logički integrirani u jednu bazu podataka. Upravitelji podataka implementirani su tako da jedan od njih kontrolira obradu svakog korisničkog zahtjeva čak i ako taj zahtjev obrađuje više od jednog upravitelja podataka. Upravitelji podataka obično ne rade kao neovisni DBMS-ovi iako obično imaju funkcionalnost za to.

Poluautonomni sustavi sastoje se od DBMS-ova koji mogu raditi neovisno, ali rade za cjelini kako bi svoje lokalne podatke dijelili. Svaki od DBMS-ova određuje koje će dijelove vlastite baze podataka učiniti dostupnima drugim DBMS-ova. Oni nisu potpuno autonomni sustavi jer ih je potrebno modificirati kako bi omogućili međusobnu razmjenu informacija.

Potpuna izolacija, je varijanta u kojoj su pojedinačni sustavi samostalni DBMS-ovi koji ne znaju za postojanje drugih DBMS-ova i kako s njima komunicirati. U takvim sustavima, obrada korisničkih transakcija koje pristupaju distribuiranim bazama podataka je teška jer ne postoji globalna kontrola nad izvođenjem pojedinačnih DBMS-ova.

Kod distribucije lokalnih sustava riječ je o podacima. Postoji više načina na koje su DBMS-ovi distribuirani, a podijeljeni su u dvije grupe: distribucija klijent/poslužitelj i puna distribucija.

Arhitektura klijent/poslužitelj dužnosti upravljanja podacima postavlja na poslužitelje, dok klijenti pružaju aplikacijsko okruženje uključujući korisničko sučelje. Komunikacija se odvija između klijentskih računala i poslužitelja. Klijent/poslužitelj DBMS-ovi predstavljaju praktični kompromis za funkcionalnost distribucije. Postoje različiti načini njihovog strukturiranja, a svaki pruža različitu razinu distribucije. Stranice na mreži se razlikuju kao klijenti i poslužitelji i njihova funkcionalnost je različita. Relativno je jednostavno implementirati ovakvu arhitekturu zbog g odvajanja funkcionalnosti i zato što je poslužitelj centraliziran. Skupa poslužiteljska računala nisu opterećena sa svakodnevnim korisničkim interakcijama, koje se odvijaju na klijentskim računalima.

Arhitektura klijent-poslužitelj ne dopušta da jedan upit obuhvati više poslužitelja jer bi proces klijenta morao biti sposoban razdvojiti takav upit na odgovarajuće podupite koji bi se izvršili na različitim mjestima, a zatim spojiti odgovore na podupite. Klijentski proces bi stoga bio prilično složen, a njegove mogućnosti počele bi se preklapati s poslužiteljskim. Za uklanjanje ovakve razlike koriste se kooperativni poslužiteljski sustavi. Postoji skup poslužitelja baze podataka, od kojih je svaki sposoban pokretati transakcije prema lokalnim podacima, koji kooperativno izvršavaju transakcije koje obuhvaćaju više poslužitelja. Kada poslužitelj primi upit koji zahtijeva pristup podacima na drugim poslužiteljima, on generira odgovarajuće podupite koje

trebaju izvršiti drugi poslužitelji i spaja rezultate kako bi izračunao odgovore na originalni upit. U idealnom slučaju, dekompozicija upita trebala bi se provesti pomoću optimizacije temeljene na troškovima, uzimajući u obzir troškove mrežne komunikacije kao i lokalne troškove obrade Subramanian (2021).

Razlikujemo još i Middleware sustave kao vrstu klijent/ poslužitelj arhitekture. Arhitektura Middleware-a dizajnirana je da omogući jednom upitu da obuhvati više poslužitelja, bez potrebe da svi poslužitelji baze podataka budu sposobni upravljati takvim strategijama izvršavanja na više stranica.

Potreban je samo jedan poslužitelj baze podataka koji je sposoban upravljati upitima i transakcijama koje obuhvaćaju više poslužitelja, a preostali poslužitelji trebaju obrađivati lokalne upite i transakcije. Poslužitelj se može promatrati kao sloj softvera koji koordinira izvršenje upita i transakcija preko jednog ili više neovisnih poslužitelja baze podataka koji se naziva međuprogramom. Međuprogramski sloj sposoban je izvršavati spajanja i druge relacijske operacije na podacima dobivenim s drugih poslužitelja, ali obično sam ne održava nikakve podatke. Osim paraleliziranja pojedinačnih operacija, moguće je paralelno izvršavati različite operacije u upitu i paralelno izvršavati više upita.

U peer-to-peer sustavima ne postoji razlika između klijenta i poslužitelja. Svako računalo ima punu funkcionalnost DBMS-a i može komunicirati s drugim računalima radi izvršavanja upita i transakcija. Većina sustava distribuiranih baza podataka pretpostavlja arhitekturu pune distribucije.

Heterogenost se može pojaviti u različitim distribuiranim sustavima, od hardverske heterogenosti i razlika u mrežnim protokolima do varijacija u upraviteljima podataka. Prikaz podataka uz pomoć različitih alata za modeliranje stvara heterogenost zbog inherentnih sposobnosti i ograničenja pojedinačnih modela podataka, dok heterogenost u upitnim jezicima uključuje korištenje potpuno različitih paradigmi pristupa podacima u različitim modelima podataka te pokriva razlike u jezicima i kada pojedinačni sustavi koriste isti model podataka. Iako je SQL sada standardni jezik za relacijske upite, postoji mnogo različitih implementacija i jezik svakog dobavljača se razlikuje (Özsu i Valduriez, 2011).

8. Implementacija podataka u distribuiranim bazama podataka

Već je spomenuto da se baza podataka distribuira preko mreže i pohranjuje na različitim mjestima na geografski različitim lokacijama radi lakšeg pristupa. Postoje tri glavna načina za distribuciju, odnosno alociranje podataka baze podataka na različitim lokacijama,a to su:

1. **Fragmentacija podataka** -U ovom pristupu, odnosi su fragmentirani (tj. podijeljeni su na manje dijelove) i svaki od fragmenata pohranjen je na različitim mjestima gdje su potrebni. Mora se osigurati da su fragmenti takvi da se mogu koristiti za rekonstrukciju izvorne relacije (tj. da nema gubitka podataka). Fragmentacija je korisna jer ne stvara kopije podataka,a dosljednost nije problem.
2. **Replikacija podataka** - kod replikacije, podaci se pohranjuju na više mjesta u vidu istih kopija na različitim mjestima. Cijela baza podataka može se reproducirati i održavati na svim ili određenim stranicama, a možemo reći i da se određena tablica može reproducirati i održavati na svim ili samo na nekoliko mjesta.
3. **Hibridni pristup** – ovaj pristup je zapravo kombinacija gore navedenih pristupa. Na primjer, može biti kombinacija fragmentacije i replikacija nekoliko ili svih fragmentacija.

8.1. Fragmentacija

Fragmentacija uključuje razdvajanje relacije (tablice) na dva ili više fragmenata i svaki od fragmenata pohranjen je na različitim mjestima gdje su potrebni. Mora se osigurati da se fragmenti mogu koristiti za rekonstrukciju izvorne relacije (tj. da nema gubitka podataka). Ako se potrebni podaci nalaze na drugim lokacijama, DDBMS ih dohvaća putem komunikacijske veze. Razdvajanje može biti po redcima, odnosno horizontalno koje se naziva horizontalna fragmentacija i po stupcima, odnosno vertikalno što se naziva vertikalna fragmentacija. Ukoliko se razdvajanje izvršava i horizontalno i vertikalno, tada je riječ o hibridnoj fragmentaciji. Razdvajanje relacija se uglavnom odvija radi poboljšanja dostupnosti podataka krajnjem korisniku i krajnjim korisničkim programima.

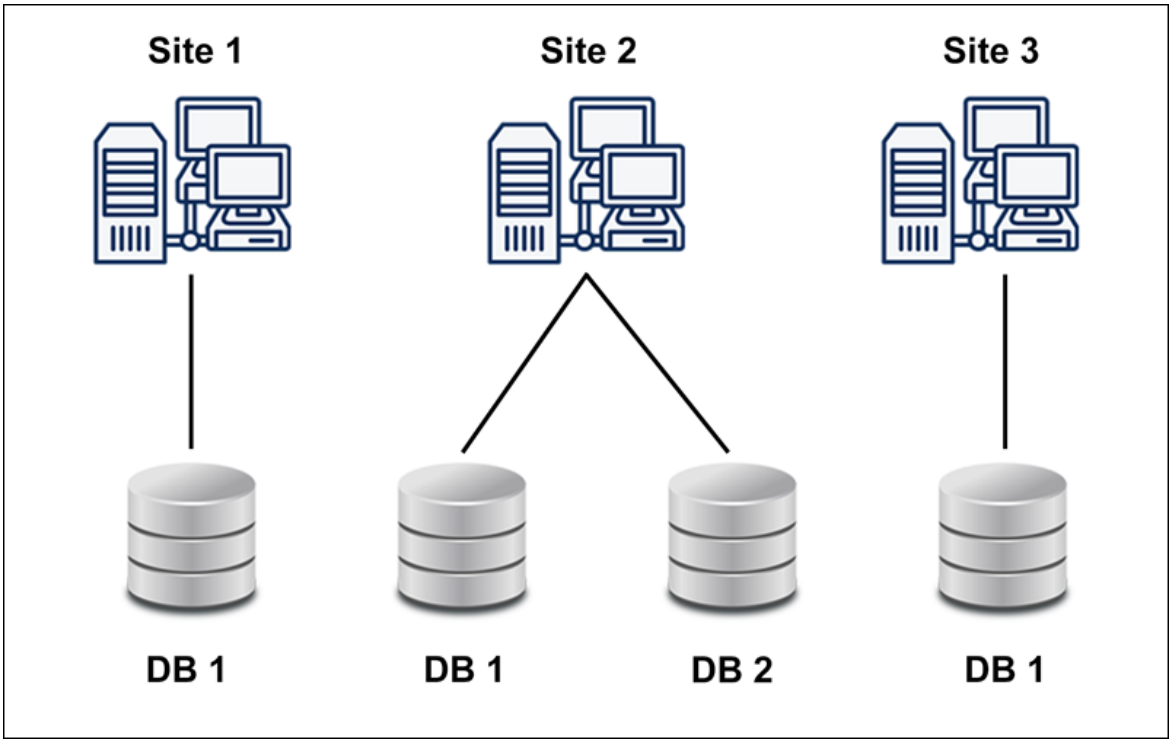
Fragmentacija se dakle dijeli na sljedeće vrste (Tutorialspoint, n.d.):

1. Horizontalna fragmentacija
    a. Primarna horizontalna fragmentacija – pristup fragmentacije tablice u kojoj se fragmentira jedna tablica, fragmentacija se izvodi po redu i koristi se skup jednostavnih uvjeta.
    b. Izvedena horizontalna fragmentacija - proces kreiranja vodoravnih fragmenata tablice na temelju već stvorenih horizontalnih fragmenata druge relacije (primjerice bazne tablice).
2. Vertikalna fragmentacija - shema relacije je fragmentirana u manje sheme, a svaki fragment sadrži zajednički ključ kandidata koji jamči spajanje bez gubitaka. Radi se o vertikalnom particioniranju tablice, tj. dekompoziciji. Stoga će podjela tablica na različitim lokacijama imati različitu strukturu. Dvije tablice nastale vertikalnom fragmentacijom,mogu biti pohranjene na različitim lokacijama radi lakšeg pristupa prema, na primjer, definiranim pravilima organizacije.
3. Hibridna fragmentacija

Korištenje fragmentacije u distribuiranoj bazi podataka izvodi se iz različitih svrha. Moguće je paralelno izvršavanje upita od strane različitih stranica tako da se upit može podijeliti u nekoliko podupita i može se izvršiti istovremeno na nekoliko različitih mjesta. Također, baza podataka je fragmentirana i fragmenti su dostupni na različitim mjestima što značajno smanjuje iskorištenost prostora na disku. Upravljanje podacima je jednostavno jer su fragmenti manji u usporedbi s cijelom bazom podataka, a osigurana je povećana dostupnost podataka lokalnim korisnicima/upitima za web mjesto na kojem su podaci pohranjeni. Većina aplikacija radi s pogledima, a ne s cijelim relacijama. Dakle, u smislu upotrebe, fragmentacija je prednost.

Kako su podaci dostupni u blizini mjesta gdje se najčešće koriste, povećava se i učinkovitost sustava u smislu obrade upita, obrade transakcija, a podaci koji nisu potrebni lokalnim aplikacijama ne pohranjuju se lokalno. To rezultira smanjenim prijenosom podataka između stranica i povećane sigurnosti.

Fragmentacija ima dva glavna nedostatka, pa tako postoji nedostatak sigurnosnih kopija podataka u distribuiranom okruženju što može uzrokovati neučinkovitost baze podataka ako web mjesto ne radi. Drugi nedostatak je problem integriteta pa u slučaju da su podaci i funkcionalne ovisnosti fragmentirane te locirane na različitim mjestima, kontrola integriteta se može provesti otežano. Na slici ispod nalazi se prikaz baze podataka nakon fragmentacije.



Slika 8 . Pojednostavljeni prikaz baze podataka nakon fragmentacije[9]

[9] preuzeto s: https://phoenixnap.com/kb/wp-content/uploads/2021/05/database-replication.png

8.2. Replikacija

Replikacija podataka je proces u kojem se relacija (tablica) ili dio relacije (fragment tablice) duplicira i duplicirane kopije se pohranjuju na više mjesta (poslužitelja) kako bi se povećala dostupnost podataka.Izvođenje replikacije podataka osigurava dosljednu kopiju baze podataka na svim čvorovima u distribuiranom sustavu što služi kako bi podaci bili široko dostupni i zaštićeni od gubitka. Replicirani podaci mogu biti potpuni ili djelomični te mogu se pohraniti lokalno, izvan lokalnog okruženja ili u oblaku. U slučaju zastoja, tvrtke vraćaju podatke i održavaju kontinuitet poslovanja obnavljanjem sa sigurnosne lokacije.

Podaci se repliciraju sinkrono ili asinkrono. Kod sinkrone replikacije podaci se istovremeno upisuju u primarnu bazu podataka i sve njezine replike dok se kod asinkrone replikacije podaci prvo zapisuju u primarnu bazu podataka, a zatim se kasnije kopiraju u replike.

Postoji nekoliko različitih metoda za repliciranje baze podataka. Na temelju svrhe repliciranih podataka i načina na koji im se pristupa, odabire se odgovarajuća metoda. Prema Kovačeviću (2021) razlikujemo:

1. Transakcijsku replikaciju
2. Replikaciju spajanja
3. Heterogenu replikaciju
4. Peer-to-peer transakcijsku replikaciju

Transakcijska replikacija stvara punu kopiju baze podataka, s novim podacima koji dolaze kako se baza podataka mijenja. Podaci se kopiraju u stvarnom vremenu prema redoslijedu promjena, što osigurava dosljednost.

Transakcijsku replikaciju je preporučeno koristiti za osiguravanje inkrementalnih promjena podataka u stvarnom vremenu. To poboljšava izvedbu i smanjuje kašnjenje, a istovremeno pruža veliku količinu aktivnosti čitanja, pisanja i brisanja.

Replikacija spajanja kombinira podatke iz nekoliko izvora u jednu bazu podataka. Korištenje replikacije spajanja omogućuje da više korisnika izvrši promjenu podataka i primjeni sve promjene na novu repliku.

Replikacija spajanja pomaže u brzom otkrivanju i rješavanju sukobljenih promjena. Također omogućuje korisnicima da izvrše promjene izvan mreže prije sinkronizacije s poslužiteljem.

Heterogena replikacija koristi se za replikaciju podataka između poslužitelja različitih dobavljača. Na primjer, omogućuje kopiranje podataka sa SQL poslužitelja na ne-SQL poslužitelj.

Peer-to-peer replikacija temelji se na transakcijskoj replikaciji. Omogućuje svim korisnicima i poslužiteljima da šalju podatke jedni drugima da se pritom ažuriranja odvijaju gotovo u stvarnom vremenu.

Peer-to-peer replikacija posebno je korisna za web aplikacije. Njegova fleksibilnost pomaže povećati broj korisnika bez utjecaja na performanse. Također čini sustav robusnijim, dopuštajući serverima da se isključe radi održavanja.

Kovačević (2021) navodi da se za replikaciju baze podataka koriste sljedeće sheme replikacije:

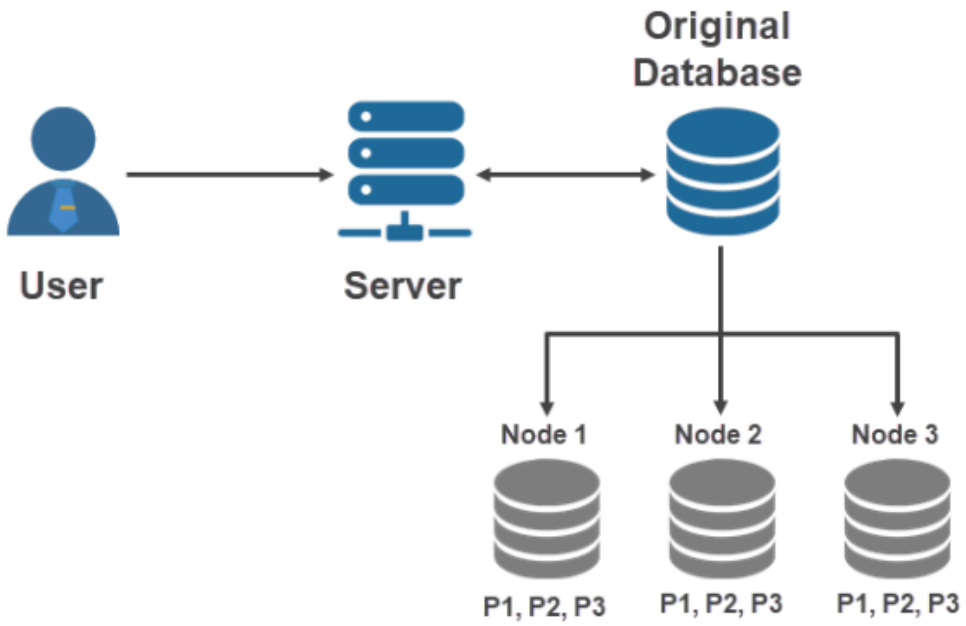
1. Potpuna replikacija - Izvođenje pune replikacije znači kopiranje kompletne baze podataka u svaki čvor distribuiranog sustava. Ovaj pristup maksimizira redundantnost podataka, povećava globalnu izvedbu i dostupnost podataka. Podaci su dostupni sve dok je jedan čvor funkcionalan.
2. Djelomična replikacija - Kopiranje samo određenih dijelova baze podataka naziva se djelomična replikacija. Obično se odlučuje koliko je važno imati određene podatke dostupne na svakoj lokaciji.
3. Bez replikacije - Svaki čvor u distribuiranom sustavu prima samo kopiju jednog dijela baze podataka. Ova shema replikacije je najbrža za izvođenje, ali ima tendenciju smanjenja dostupnosti podataka i ostavlja bazu podataka ranjivom na gubitak podataka.

Upotrebom replikacije baze podataka postiže se povećana pouzdanost i dostupnost.

Postoji mnogo kopija istih podataka na nekoliko različitih lokacija (obično različite geografske lokacije). Stoga kvar bilo koje stranice (poslužitelja) neće utjecati na transakcije.Upiti koji zahtijevaju replicirane kopije podataka uvijek su brži (posebno upiti za čitanje) Distribuirana baza podataka osigurava dostupnost podataka tamo gdje su potrebni. Cijela se tablica sama učitava lokalno. Stoga se na te upite može brzo odgovoriti s lokalnog mjesta na kojem su pokrenuti. Smanjeni su troškovi komunikacije pa kada se na stranici generira veći broj upita za čitanje, na sve njih može se odgovoriti lokalno. Samo upiti koji uključuju drugu tablicu ili upiti koji pokušavaju nešto napisati trebaju koristiti komunikacijske veze za kontaktiranje drugih stranica.

Za replikaciju je potrebno više prostora za pohranu u usporedbi s centraliziranim sustavom – replikacija bi značila dupliciranje bilo koje tablice i pohranjivanje na svakom mjestu što zahtijeva više prostora na svakom mjestu. Operacija ažuriranja je zahtjevna ako imamo više kopija istih

podataka učitanih na različitim stranicama. Potrebno je ažurirati sve replike kad god se mijenjaju podaci. Održavanje integriteta podataka je složeno te uključuje složene postupke za održavanje dosljedne baze podataka (Connolly i Begg, 2005). Na slici ispod nalazi se prikaz baze podataka nakon potpune replikacije.



Slika 9 - Prikaz baze podataka nakon potpune replikacije[10]

[10] preuzeto s: https://phoenixnap.com/kb/wp-content/uploads/2021/06/database-replication-partial-replication-new.png

8.3. Usporedba replikacije i fragmentacije

U tablici 2 nalazi se usporedba potpune replikacije, djelomične replikacije i fragmentacije po nekoliko karakteristika kao što su obrada upita, upravljanje rječnikom, kontrola istodobnosti, pouzdanost i primjena.

Tablica 2 - Usporedba potpune replikacije, djelomične replikacije i fragmentacije[11]

	Potpuna replikacija	Djelomična replikacija	Fragmentacija
Obrada upita	Ista kopija baze podataka dostupna je na različitim stranicama, stoga je obrada upita jednostavna.	Upit koji je generiran na različitim mjestima za podatke koji se nalaze na nekim drugim mjestima potrebno je obraditi kako bi se razumjela lokacija na kojoj se podaci nalaze i kako bi se pronašla optimalna strategija izvršenja.	
Upravljanje rječnikom	Sve su tablice slične strukture, stoga ne treba globalni katalog. Dovoljan je obični rječnik podataka koji se koristi u centraliziranim sustavima baza podataka	Potreban je globalni katalog kako bi podaci i njihove lokacije bili poznati.	
Kontrola istodobnosti	Istodobno čitanje može se dopustiti na svim replikama bilo kojim brojem transakcija. Istodobno pisanje može biti dopušteno samo na jednoj replici odjednom. Svaka transakcija pisanja mora promijeniti podatke na svim lokacijama.	Podaci su fragmentirani i replicirani. Teško je rukovati istodobnim transakcijama koje uključuju podatke iz repliciranih kopija ili fragmentiranih i repliciranih kopija.	U particioniranoj bazi podataka podaci na različitim mjestima jedinstveni su za onu stranicu na kojoj se podaci nalaze, a rukovanje istodobnim transakcijama isto je kao rukovanje istodobnim izvršavanjima u centraliziranoj bazi podataka. Stoga se smatra lakim.
Pouzdanost	Višestruke kopije iste baze podataka povećavaju dostupnost. Kvar bilo koje stranice ne bi uzrokovao probleme u pristupu podacima. Podacima se ipak može pristupiti sa stranica koje su aktivne. Pouzdanost je vrlo visoka.	Dostupnost više kopija za nekoliko fragmenata i dalje bi podržavala pouzdanost.	Budući da su pojedinačni fragmenti na pojedinačnim mjestima jedinstveni, osjetljivi su na jednu točku kvara. Pouzdanost je niska.
Primjena	Moguća primjena	Izgledna primjena	Moguća primjena

[11] Preuzeto s: https://www.exploredatabase.com/2015/02/comparison-of-replication-alternatives-in-distributed-database-design.html

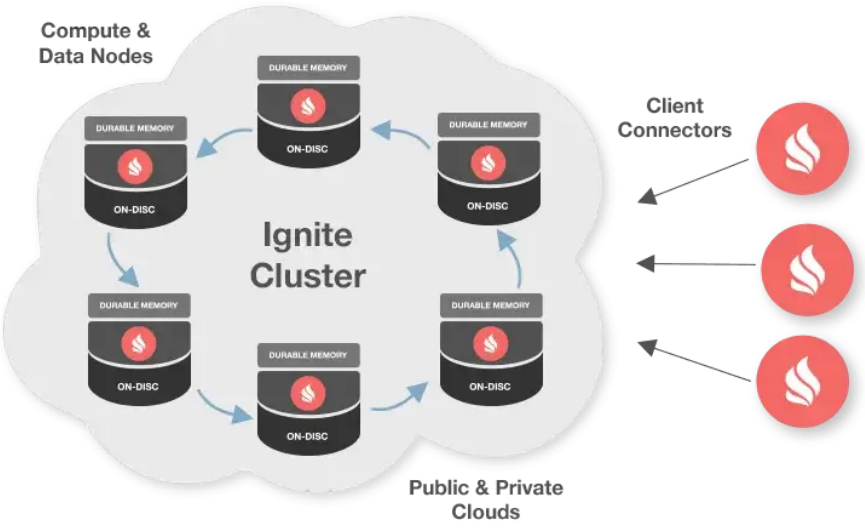
9. Primjeri softvera za distribuirane baze podataka

Distribuirane baze podataka pomažu u sigurnijoj i pouzdanijoj pohrani i postavljanju upita. U ovom odlomku navedene su neke od najboljih distribuiranih baza podataka otvorenog koda i komercijalne baze podataka koje zadovoljavaju povećane potrebe za pohranom podataka. Neki od softvera za distribuirane baza podataka su: Apache Ignite, Apache Cassandra, Apache HBase, Couchbase Server, AWS SimpleDB, FoundationDB i drugi.

Apache Ignite je distribuirana baza podataka otvorenog koda s velikom brzinom pohrane i mogućnostima visokih performansi. Poznat je po svojoj upotrebi u predmemoriranju (pojam predmemorije koristi se iz povijesnih razloga jer je u početku baza podataka podržavala memorijski sloj) podataka i nudi izdržljivo, visoko dostupno i dosljedno očuvanje podataka sa skalabilnom SQL podrškom. Riječ je o brzom i intuitivnom rješenju distribuirane baze podataka s punom podrškom za vanjske baze podataka kao što je Cassandra.

Ignite baza podataka organizira podatke u obliku parova ključ-vrijednost u distribuiranim predmemorijama. Općenito, svaka predmemorija predstavlja jedan tip entiteta kao što je zaposlenik ili organizacija. Svaka predmemorija podijeljena je u fiksni skup particija koje su ravnomjerno raspoređene među čvorovima klastera pomoću algoritma haširanja najveće slučajne težine. Uvijek postoji jedna primarna i nula, jedna ili više rezervnih kopija particije, a broj kopija konfiguriran je parametrom faktora replikacije. Ako je konfiguriran način pune replikacije, tada će svaki čvor klastera pohraniti kopiju particije. Particije se ponovno balansiraju automatski ako se čvor doda ili ukloni iz klastera kako bi se postigla ravnomjerna distribucija podataka i rasporedilo radno opterećenje. Parovi ključ-vrijednost čuvaju se u particijama. Apache Ignite preslikava par u particiju uzimajući vrijednost ključa i prosljeđujući je posebnoj hash funkciji.

Apache Igniteu se može pristupiti pomoću SQL API-ja postavljenih putem JDBC i ODBC upravljačkih programa i izvornih biblioteka razvijenih za programske jezike Java, C#, C++. Na slici ispod je grafički prikaz Apache Ignite-a.



Slika 10 - Apache Ignite[12]

Apache Cassandra je u početku razvijena od strane Facebooka, a riječ je o NoSQL distribuiranoj bazi podataka koja nudi dostupnu i učinkovitu pohranu podataka. To je skalabilno rješenje koje koriste velike tehnološke tvrtke, poput Netflixa, eBaya i Ubera.

Cassandra je operativni sustav i neovisan je o platformi s otpornošću, sigurnosti i visokoj dostupnosti koja mu pomaže u obradi upita s malom latencijom. To je alat otvorenog koda, ali je također dostupan putem dobavljača trećih strana s komercijalnim uslugama podrške. Svaki čvor u klasteru ima istu ulogu, a ne postoji niti jedna točka neuspjeha. Podaci se distribuiraju po klasteru (tako da svaki čvor sadrži različite podatke), ali ne postoji glavni jer svaki čvor može poslužiti bilo koji zahtjev. Podaci se automatski repliciraju na više čvorova radi otpornosti na greške. Podržana je replikacija u više podatkovnih centara, a neispravni čvorovi mogu se zamijeniti bez zastoja. Cassandra je predstavila Cassandra Query Language (CQL). CQL je jednostavno sučelje za pristup Cassandri, kao alternativa tradicionalnom Structured Query Language (SQL). (Apache Cassandra, n.d.)

Apache HBase je još jedna usluga distribuirane baze podataka iz Apachea. To je nerelacijska baza podataka za Apache Hadoop, zbirku softverskih uslužnih programa otvorenog koda koji olakšavaju korištenje mreže mnogih računala za rješavanje problema koji uključuju

[12] preuzeto s: https://miro.medium.com/max/4800/1*MF0p86V7csS4KjHf8T0MGQ.webp

ogromne količine podataka i računanja. Nastao je po uzoru na Googleov Bigtable za pohranu velikih skupova podataka na skalabilan, dosljedan i vrlo dostupan način.

Tablice u HBase mogu poslužiti kao ulaz i izlaz za MapReduce poslove koji se izvode u Hadoopu, a može im se pristupiti putem Java API-ja, ali i putem REST, Avro ili Thrift gateway API-ja. HBase je prihvaćena u širim krugovima zbog povezanosti s Hadoopom i HDFS-om. HBase radi na vrhu HDFS-a i prikladan je za brze operacije čitanja i pisanja na velikim skupovima podataka s visokom propusnošću i malom ulazno/izlaznom latencijom.

HBase nije zamjena za klasičnu SQL bazu podataka, međutim projekt Apache Phoenix pruža SQL sloj za HBase kao i JDBC upravljački program koji se može integrirati s raznim analitičkim i aplikacijama poslovne inteligencije. Projekt Apache Trafodion pruža mehanizam za SQL upite s ODBC i JDBC upravljačkim programima i distribuiranu ACID zaštitu transakcija u više izjava, tablica i redaka koji koriste HBase kao mehanizam za pohranu.

Couchbase Server je distribuirana NoSQL baza podataka na razini poduzeća. To je baza podataka koja koristi ključ-vrijednost način pohrane podataka. Otvorenog je koda i pruža skalabilnost i fleksibilnost potrebnu u distribuiranom oblaku i rubnim okruženjima. Dizajnirana je tako da ima visoke performanse i idealna je za korištenje u oblaku, mobilnim i rubnim računalnim aplikacijama.

Svaki Couchbase čvor sastoji se od podatkovne usluge, usluge indeksa, usluge upita i komponente upravitelja klastera. Počevši s verzijom 4.0, tri usluge mogu se distribuirati za rad na zasebnim čvorovima klastera ukoliko je to potrebno. Upravitelj klastera nadzire konfiguraciju i ponašanje svih poslužitelja u klasteru Couchbase-u. Konfigurira i nadzire ponašanje među čvorovima poput upravljanja tokovima replikacije i operacijama ponovnog balansiranja. Također pruža metričku agregaciju i funkcije konsenzusa za klaster te RESTful sučelje za upravljanje klasterom. Upravitelj klastera koristi programski jezik Erlang i Open Telecom Platform.

Upravitelj podataka u Couchbase-u pohranjuje i dohvaća dokumente kao odgovor na podatkovne operacije iz aplikacija. Asinkrono zapisuje podatke na disk nakon potvrde klijentu. Moguće je izborno osigurati da se podaci zapisuju na više od jednog poslužitelja ili na disk prije potvrde pisanja klijentu. Parametri definiraju starost stavki koje utječu na to kada se podaci zadržavaju i kako se rukuje maksimalnom memorijom i migracijom iz glavne memorije na disk.

AWS SimpleDB, dio Amazon web usluga, je distribuirana baza podataka napisana u Erlangu koja se integrira s drugim AWS uslugama, uključujući EC2 i Amazon S3. Neke od

njegovih značajki uključuju visoku dostupnost, fleksibilnost, učinkovitost, skalabilnost i, poput ostalih AWS usluga, isplativost. Pruža olakšano uklanjanje složenosti operacija i koristi jednostavan API za pristup i pohranu podataka koji se automatski indeksiraju kako bi se smanjio administrativni teret. Međutim, ima konačnu, odnosno slabiji stupanj dosljednosti i smanjenu pohrane u usporedbi s drugim dostupnim uslugama distribuirane pohrane. To se često smatra ograničenjem, jer otežava pisanje ispravnih programa koji koriste SimpleDB. Takvo ograničenje je rezultat temeljnog kompromisa dizajna. Navedenom dosljednošću, sustav može postići još dva vrlo poželjna svojstva:

- dostupnost – komponente sustava mogu otkazati, ali usluga će nastaviti ispravno raditi.
- particijska tolerancija – komponente u sustavu su međusobno povezane računalnom mrežom. Ako komponente ne mogu međusobno komunicirati putem mreže, rad sustava će se nastaviti.

Pretpostavlja se da su kvarovi komponenti neizbježni pa su se oba svojstva smatrala potrebnima za pružanje pouzdane web usluge. CAP teorem kaže da nije moguće da sustav pruži oba svojstva zajedno s dosljednošću što je rezultiralo da su se dizajneri morali zadovoljiti slabijim oblikom dosljednosti.

Idealna je za pohranjivanje podataka o igrama na mreži, pokazivače indeksiranja Amazon S3 objekata i zapisivanje revizija i metrika analize.

FoundationDB je NoSQL distribuirana baza podataka otvorenog koda s arhitekturom pohrane podataka s više modela. Omogućuje pohranu različitih tipova podataka u jednu bazu podataka. Tolerantan je na greške i visoko skalabilan s visokim performansama za jednostavna i velika radna opterećenja. Zahvaljujući pohrani podataka s više modela, FoundationDB je idealan za mnoge slučajeve, uključujući aplikacije u oblaku i rubne aplikacije.

FoundationDB je dizajniran za implementaciju na distribuiranim klasterima standardnog hardvera koji pokreće Linux, a uz podršku standardnog čitanja i pisanja temeljenog na ključu, svojstvo naručivanja omogućuje čitanje raspona koje može učinkovito skenirati velike dijelove podataka. Replikacija se izvodi na način da se pohranjuje svaki dio podataka na više strojeva prema faktoru replikacije koji se može konfigurirati, a trostruka replikacija je preporučeni način za klastere od 5 ili više strojeva. Ograničenja ove baze podataka je što ne podržava transakcije dulje

od pet sekundi, a ukupna veličina transakcije ne može iznositi više od 10 MB. Svaki ključ može imati maksimalnu veličinu od 10kB, a vrijednosti 100kB.

10. Zaključak

Distribuirani sustav baze podataka čini cjelokupnu arhitekturu složenijom, ali olakšava podešavanje sustava za propusnost. S distribuiranim sustavom je moguće postaviti propusnost upita za čitanje ili upite za pisanje u izolaciji. Povrh toga, distribuirani sustav omogućuje rješavanje kvarova hardvera. Katastrofe se događaju, pa je gubitak podatkovnog centra zbog prekida normalna pojava. Ukoliko tvrtke mogu podnijeti gubitke podataka, korištenje distribuiranih baza podataka nije potrebno, ali za, na primjer, globalno korištenu web aplikaciju koja mora biti online 24/7, distribuiranje je gotovo neophodno.

Pohrana podataka na više lokacija osigurava sigurnost i sprječava gubitak podataka. Pruža visoku dostupnost podataka zbog više čvorova koji su u svakom trenutku spremni odgovoriti na upit. Distribuirane baze podataka nude visoke performanse zbog podjele radnog opterećenja i bolju kontrolu nad resursima. Distribuirane baze podataka imaju i svoje negativne strane poput zahtjeva za većom pohranom i infrastrukturom. Sustav je teško održavati zbog svoje veličine i složenosti, a iz istog razloga je i teže osigurati. Trošak implementacije je znatno viši u odnosu na centralizirane baze podataka, ali ipak su dugoročno isplativije.

Arhitektura distribuiranih sustava koji služe za upravljanje bazama temelje se u osnovi na modelima centraliziranih sustava za upravljanje uz svojevrsnu nadogradnju kako bi bili u skladu za distribuirano okruženje. Postoje različite arhitekture koje se temelje na različitim karakteristikama, a implementacija podataka omogućena je fragmentacijom i replikacijom kojima se podaci pohranjuju na više stranica, odnosno čvorova unutar mreže.

Zbog količine podataka koja se pohranjuje u distribuiranim bazama podataka, one su najprikladnije za veće tvrtke i korporacije. Tvrtke trebaju unaprijed razmišljati o prednostima i nedostacima koje pružaju ovi sustavi te osigurati protumjere kako bi se zadržala sigurnost podataka.

11. Literatura

1. Apache Cassandra (n.d.), *Open Source NoSQL Database,* Apache Cassandra, preuzeto s: https://cassandra.apache.org/_/index.html
2. Connolly T., Begg C. (2005), *Database Systems: A Practical Approach to Design, Implementation and Management*, Addison Wesley, London, 2005.
3. Date C. J. (2003) *An Introduction to Database Systems*, Addison-Wesley Educational Publishers Inc
4. Kovačević A. (2021, 17. lipanj), *What Is Data Replication?,* PhoenixNAP, preuzeto s: https://phoenixnap.com/kb/data-replication
5. Lutkevich B. (2021), Database (DB), TechTarget, preuzeto s:https://www.techtarget.com/searchdatamanagement/definition/database
6. Manger R. (2012), *Baza podataka*, Element, Zagreb
7. Marijan B. (2021, 6. svibanj), *What Is a Distributed Database?,* PhoenixNAP, preuzeto s: https ://phoenixnap.com/kb/distributed-database
8. Oracle (n.d.), *Distributed Databases*, Oracle7 Server Concepts Manual, preuzeto s: https://docs.oracle.com/cd/A57673_01/DOC/server/doc/SCN73/ch21.htm#intro%20dist%20db
9. Özsu M. T., Valduriez P. (2011), *Principles of Distributed Database Systems*, Third Edition, Springer, New York
10. Sakshi (2022. 7. lipanj), *Difference between Centralized Database and Distributed Database,* GeeksforGeeks, preuzeto s: https://www.geeksforgeeks.org/difference-between-centralized-database-and-distributed-database/
11. Singh T. (2021, 21. lipanj), *Transparencies in DDBMS*, GeeksforGeeks, preuzeto s: https://www.geeksforgeeks.org/transparencies-in-ddbms/
12. Shah R. (2021. 29. rujan), Types of Distributed Databases, BenchPartner, preuzeto s: https://benchpartner.com/types-of-distributed-databases#
13. Subramanian V. K. (2021, 12. kolovoz), *Distributed DBMS Architectures*, Medium, preuzeto s: https://medium.com/@VijayKumarSubramanian/distributed-dbms-architectures-a79c1c1e449d

14. Tutorialspoint (n.d.), *Distributed DBMS - Design Strategies,* Tutorialspoint, preuzeto s:https://www.tutorialspoint.com/distributed_dbms/distributed_dbms_database_environments.htm

12. Popis slika

13. Popis tablica