UNIVERZITET METROPOLITAN BEOGRAD

IT250 - BAZE PODATAKA

Zaštita i distribucija baza podataka

Lekcija 12

PRIRUČNIK ZA STUDENTE

IT250 - BAZE PODATAKA

Lekcija 12

ZAŠTITA I DISTRIBUCIJA BAZA PODATAKA

UVOD

Šta ćemo naučiti u ovoj lekciji?

U prvom delu predavanja se govori o zaštiti baza podataka koja se može realizovati na nivou baze ili aplikacije.

U drugom delu predavanja se govori o paralelnoj i distribuiranoj obradi baza podataka.

Suština paralelne obrade je korišćenje više procesora. Obično je broj procesora p veliki, po nekoliko stotina ili hiljada.

Paralelna arhitektura se može podeliti na tri grupe:

1. *Kod najčvršće povezane paralelne arhitekture se deli glavna memorija (eng.* Shared -Memory Machines*).*
2. *Kod slabo povezane arhitekture se deli spoljašnja memorija (disk) a ne glavna memorija (eng.* Shared-Disk Machines*) .*
3. *Arhitektura koja se često koristi kod baza podataka se naziva arhitektura kod koje se ništa ne deli (eng.* Shared-Nothing Machines*), kod koje se ne deli disk a procesori su povezani i dele se samo prenosom poruka*

Osnovna karakteristika distribuiranih sistema je velika količina poruka i podataka koji se prenose preko komunikacione mreže. Zato je glavni cilj u postizanju efikasnosti distribuiranih sistema – smanjenje mrežne komunikacije.

Da bi se postigla efikasnost DDBMS-a, sve njegove komponente treba realizovati na način koji smanjuje mrežnu komunikaciju.

Najznačajniji problemi koje tom prilikom treba rešiti su:

1. *fragmentacija podataka*
2. *distribuirana obrada upita*
3. *distribuirano (preneto) ažuriranje*
4. *upravljanje katalogom*
5. *distribuirano izvršenje skupa transakcija, što uključuje konkurentnost, integritet, oporavak i protokole kompletiranja transakcija.*

Poglavlje 1

Zaštita baza podataka

PRETNJE BAZI PODATAKA

Postoji veliki broj pretnji kojima je izložena baza podataka

Bezbednost treba razvijati u dva smera. Jedan je na osnovu uvažavanja i svesti o postojanju pretnji, a drugi na osnovu tehnika koje postoje za njihovo izlečenje.

Pretnje uključuju:
1. **Neovlašćenu modifikaciju**: Promene vrednosti podataka zbog sabotaže, kriminala ili neznanja koje mogu biti omogućene neodgovarajućim bezbednosnim mehanizmima, ili deljenjem password-a, na primer.
2. Neovlašćeno otkrivanje: Kada su informacije koje nije trebalo da budu obelodanjene, otkrivene. Opšte pitanje od ključnog značaja je da li je to slučajno ili namerno.

3. **Gubitak dostupnosti**: Ponekad se naziva uskraćivanje usluge. Kada baza podataka nije dostupna, nastaje gubitak. Dakle, treba izbegavati svaku pretnju koja izaziva prestanak rada sistema, čak i da bi se proverilo da li se nešto dogodilo.

4. **Finansijski gubitak**: Većina finansijskih gubitaka nastalih zbog prevare potiče od zaposlenih. Kontrola pristupa obezbeđuje i zaštitu od krivičnih dela i dokaze o pokušajima (uspešnim ili drugim) da se izvrše radnje koje su štetne za organizaciju, bilo da se radi o prevari, izvlačenju osetljivih podataka ili gubitku dostupnosti.

5. **Ličnu privatnost i zaštitu podataka**: Na međunarodnom planu, lični podaci su obično predmet zakonodavne kontrole. Lični podaci su podaci o pojedincu koji se može identifikovati. Pojedinac mora biti živ, ali metod identifikacije nije u potpunosti propisan. U nekim slučajevima, pojedinca može identifikovati poštanski broj mesta stanovanja, ako samo jedna osoba živi na adresi sa datim poštanskim brojem. Takvi podaci zahtevaju pažljivo rukovanje i kontrolu. Lične podatke treba identifikovati kao takve. Mora postojati kontrola upotrebe tih podataka (što može ograničiti ad-hoc upitima). Kao dokazi se moraju čuvati tragovi revizije svih pristupa i objavljivanja informacija.

6. **Zloupotrebu računara**: pri zloupotrebi računara se primenjuje opšte zakonodavstvo. Zloupotreba uključuje kršenje kontrole pristupa i pokušaje da se prouzrokuje šteta promenom stanja baze podataka ili uvođenjem crva i virusa koji ometaju njeno ispravno funkcioniranje.
7. **Zahteve za revizijom**: Ovo su operativna ograničenja koja su nastala zbog potrebe da se zna ko je šta uradio, ko je pokušao da nešto uradi i gde i kada se sve to dogodilo. Zahtevi za revizijom omogućavaju otkrivanje događaja izvršenih nad bazom podataka (uključujući CONNECT i GRANT transakcije), pružanje dokaza za osiguranje kao i odbranu ili krivično gonjenje.

BEZBEDONOSNI MODELI

Bezbedonosni modeli objašnjavaju raspoložive osobine DBMS-a koje treba koristiti za razvoj i upravljanje bezbednosti baza podataka. Specifični DBMS-i imaju vlastite sigurnosne modele

Bezbedonosni modeli objašnjavaju raspoložive osobine DBMS-a koje treba koristiti za razvoj i upravljanje sigurnosnim sistemima. Oni otelovljuju koncepte, implementiraju politiku i obezbeđuju servere za takve funkcije. Bilo koja greška u bezbedonosnom modelu može prouzrokovati ili nezaštićen rad sistema ili nesiguran sistem. U tu svrhu treba vršiti:

1. *Kontrolu pristupa*
2. *Autentifikaciju i autorizaciju*

Autentifikacija i autorizacija: Svi smo mi kao korisnici upoznati sa zahtevima za prijavljivanje (log-in) na većini sistema. Pristup IT resursima obično zahteva proces prijavljivanja koji je siguran. Ovde se autentifikacija i autorizacija prvenstveno odnosi na pristup sistemima za upravljanje bazama podataka i predstavlja sagledavanje procesa iz perspektive DBA. Drugi modeli sistema se razlikuju u većoj ili manjoj meri, iako osnovni principi ostaju isti.

Kontrola pristupa: Svrha kontrole pristupa mora uvek biti jasna. Kontrola pristupa je skupa u smislu analize, projektovanja i operativnih troškova. Primenjuje se na poznate situacije, na poznate standarde, radi postizanja poznatih ciljeva. Kontrolu pristupa ne treba primenjivati bez svih gore navedenih znanja. Kontrola uvek mora da odgovara situaciji.

Diskreciona kontrola pristupa postoji kada se specifičnoj imovini odrede određene privilegije, tako da je ovlašćeni korisnici mogu koristiti na određeni način.

Da bi se DBMS zaštitio, mora se konstruisati matrica pristupa, koja uključuje objekte kao što su tabele, redovi tabela, pogledi i operacije za svakog korisnika - svaki unos razdvaja privilegije kreiranja, čitanja, unošenja i ažuriranja. Ova matrica može postati veoma složena pošto ovlašćenja mogu varirati od objekta do objekta. Matrica takođe može da postane veoma velika, tako da njena implementacija često zahteva posebnu fizičku implementaciju. Nekada je nemoguće matricu sačuvati u glavnoj memoriji računara. Najjednostavnije rečeno, matrica se može posmatrati kao dvodimenzionalna tabela (slika 1.1):

PROBLEM	ALAT	TEHNIKA
Pouzdanost (operativna bezbednost)	Oporavak nakon gubitka ili oštećenja (Recovery)	BACKUP, logovanje, CHECK-POINT
Bezbednost pristupa	Kontrola pristupa	Lozinke (password)
Integritet (bezbednost šeme)	Obezbeđenje interne konzistentnosti	Pravila za validaciju, ograničenja (constraints)

Slika 1.1 Matrica pristupa [Izvor: Autor]

Poglavlje 2

SQL naredbe za zaštitu baza podataka

DISKRECIONA BEZBEDNOST

Osnove SQL-a su same po sebi diskrecione. Privilegije za korišćenje resursa baze podataka dodeljuju se i uklanjaju pojedinačno.

U ovom delu su predstavljene SQL naredbe potrebne za implementaciju kontrole pristupa kako bi se steklo dovoljno znanja o ovoj oblasti SQL-a. Osnove SQL-a su same po sebi diskrecione. Privilegije za korišćenje resursa baze podataka dodeljuju se i uklanjaju pojedinačno.

Prvo pitanje je kome je šta dozvoljeno da radi sa sigurnosnim podsistemom. Da bi ste mogle primenjivati mere bezbednosti, morate imati visok nivo privilegije. Nažalost, takve uloge ne postoje unutar SQL standarda i variraju od DBMS do DBMS. Uloga je definisana kao skup privilegija.

Kao primer, u Oracle-u postoje sledeće uloge (između ostalih):

1. *SYSOPER*: Pokreće i zaustavljanje DBMS-a.
2. *DBA*: Ima dozvolu da kreira korisnike i upravlja bazom podataka i postojećim korisnicima.
3. *SYSDBA*: ima sva ovlašćenja DBA plus ovlašćenje za kreiranje, pokretanje, zaustavljanje i oporavak DBMS.

Nivoom autorizacije se uspostavljaju neka osnovna prava. Korisnički nalog SYSDBA ima sva prava i može sve promeniti. Prava za pristup tabelama moraju biti posebno dodeljena od strane DBA ili SYSADM.

DBA privilegijom korisnik može definisati:

1. *SELECT* iz bilo koje tabele i pogleda,
2. *CREATE* objekata baze podataka za druge korisnike,
3. *DROP* drugih korisnika objekata baze podataka, uključujući i tabele, sinonime i linkovane baze podataka,
4. *GRANT* varijante privilegija baze podataka,
5. *CREATE* public sinonima i linkovanje baze podataka,
6. *EXPORT* i ***IMPORT*** baze podataka.
7. *WITH GRANT OPTION* daje dozvolu davanja privilegija drugom korisniku.
8. *GRANT* <Privilegija> *ON* <tabele ili pogled> *TO* <korisnik ili grupa korisnika>*[WITH GRANT OPTION]*;

ZNAČENJE NAREDBE GRANT U ZAŠTITI BP

Naredba GRANT se koristi za dodeljivanje privilegija za korišćenje tabela i pogleda drugim korisnicima.

```
GRANT <Privilegija>
    ON <tabele ili pogled>
TO <korisnik ili grupa korisnika>
[WITH GRANT OPTION];
```

Privilegije mogu biti:

1. **SELECT**
2. **UPDATE**
3. **INSERT**
4. **DELETE**
5. **INDEX**
6. **EXPAND** (dodavanje atributa relacije)
7. **ALL** (važi za sve navedene privilegije)
8. **RESOURCE** (omogućuje korisniku kreiranje objekata baze podataka, kao što su: tabele, indeksi, klasteri)
9. **DBA** (obavlja administrativne zadatke, kao što su: **CREATE TABLESPACE** i **CREATE ROLLBACK SEGMENT**)

Tako se na primer korišćenjem naredbe GRANT mogu dodeliti prava za promenu podataka:

GRANT INSERT ON TABLE1 TO U2, U3;
GRANT DELETE ON TABLE1 TO U2, U3;
GRANT UPDATE ON TABLE1(salary) TO U5;
GRANT INSERT, DELETE ON TABLE1 TO U2, U3;

Obratite pažnju da su kod ažuriranja, atributi koji se mogu modifikovati navedeni po imenu kolone. Konačna forma podrazumeva kombinovanje privilegija u jednom izrazu.
Da biste obezbedili opšti pristup, koristite:
GRANT ALL TO PUBLIC;

PRIMER: KORIŠĆENJE NAREDBE GRANT NAD TABELAMA

Primeri dodeljivanja određenih prava nad tabelama različitim korisnicima.

***Primer 1**: Uvedi privilegije nad tabelom RADNIK, a onda daj privilegije nad tom tabelom korisniku PERA:*
GRANT ON RADNIK TO PERA;

***Primer 2**: Ako bi korisniku VLADA trebalo da se da privilegija samo za SELECT nad tabelom RADNIK, to se piše ovako:*
GRANT SELECT ON RADNIK TO VLADA;
***Primer 3:** Ukoliko bi trebalo da se korisniku STEFAN da privilegija za UPDATE nad tabelom RADNIK, to se piše ovako:*
GRANT UPDATE (PLATA, STIMUL) ON RADNIK TO STEFAN;

Privilegije nad naredbama: SELECT, INSERT, UPDATE i DELETE se mogu davati i za poglede.
***Primer 1**: Ako se ne želi da korisnik ALEKSANDAR ima pogled na kolone: PLATA i STIMUL, onda je bolje da se privilegije ograničavaju na pogled, a ne na tabele. Prvo se definiše pogled na tabelu RADNIK koja ne sadrži kolone: PLATA I STIMUL.*

CREATE VIEW ZAPS AS SELECT SIFRAR, PREZIME, SIFRARM, RUKOV, DATUMZ, SIFRAO FROM RADNIK;

Za davanje privilegije korisniku ALEKSANDAR nad pogledom ZAPS piše se:

GRANT SELECT ON ZAPS TO ALEKSANDAR;

***Primer 2**: Može se definisati i pogled nad tabelom RADNIK koja daje podatke samo o onim zaposlenima koji rade u istoj šifri odeljenja u kojem radi i osoba koja koristi pogled.*

```
CREATE VIEW IMERAD AS
    SELECT * FROM RADNIK WHERE SIFRAO IN
        (SELECT SIFRAO FROM RADNIK WHERE PREZIME = USER);
```

USER definiše ime prijavljenog korisnika. Ako MARIĆ, koji je u odeljenju 10, pristupio pogledu može videti samo zaposlene u odeljenju 10. Ovim pogledom može se dati korisniku MARIĆ privilegija da vrši npr. ažuriranje na sledeći način:

GRANT SELECT, UPDATE ON MARIC TO MARIĆ;

NAREDBA REVOKE ZA ODUZIMANJE PRIVILEGIJA

Primeri naredbe REVOKE

Ako neki zaposleni pređe u drugo odeljenje (polje SIFRAO se menja), novi rukovodilac automatski dobija pristup podacima, dok ga stari rukovodilac gubi.

Kada treba poništiti privilegije koristi se naredba REVOKE.

***Primer:** Ako se želi oduzeti privilegija korisniku ALEKSANDAR za ubacivanje podataka u tabelu RADNIK, to se piše:*

REVOKE INSERT ON RADNIK FROM ALEKSANDAR;

Privilegije se mogu dati određenom čoveku (korisniku) ali se mnogo češće daju grupi ljudi. Ta grupa ljudi se često naziva rola ili grupa korisnika. Davanje dozvola rolama je uobičajeno ali nije obavezno. Međutim, kada se koriste role, neophodno je imati način da se korisnici dodele odgovarajućim rolama.

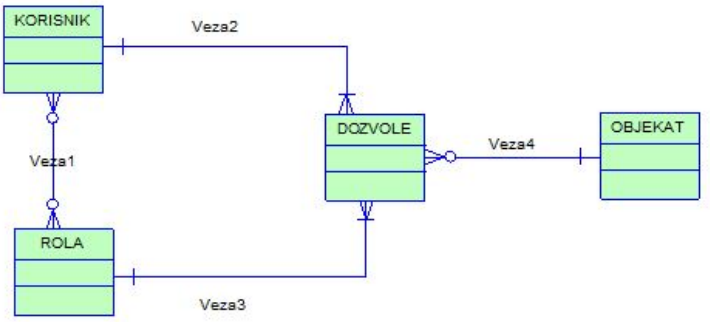
Dodeljivanje privilegija odnosno odgovornosti koje imaju pojedini korisnici ne može omogućiti sam DBMS ili aplikacija koja radi nad bazom podataka već se ono mora odraditi ručnim procedurama i objasniti korisnicima u toku obuke za korišćenje sistema. Zadatak DBMS je da upravlja obradom prava i odgovornosti.

ZAŠTITA NA NIVOU RDBMS

RDBMS ograničava akcije korisnika na dozvole koje su definisane za korisnika ili dozvole koje su definisane na nivou rola koje su korisniku dodeljene.

Generalni model zaštite na nivou DBMS je prikazan na slici 2.1.

Kada se korisnik prijavi na bazu podataka, RDBMS ograničava akcije korisnika na dozvole koje su definisane za korisnika ili dozvole koje su definisane na nivou rola koje su korisniku dodeljene. Određivanje da li je neko zaista onaj za kog se deklariše, je generalno veoma težak zadatak. Svi komercijalni RDBMS koriste verifikaciju korisnika uz pomoć username-a i password-a, i takva zaštita je sasvim dovoljna ukoliko korisnik svoj password čuva samo za sebe.



Slika 2.1 Generalni model zaštite podataka na nivou RDBMS [Izvor: NM IT350 - 2020/2021.]

U tabeli 2.2 su date smernice kojima se može osigurati zaštita sistema baze podataka.

DBMS treba da bude zaštićen firewall-om
Treba koristiti samo neophodne funkcije • Dozvoliti podršku za što manji mogući broj mrežnih protokola • Izbrisati sve sistemske store procedure koje nisu neophodne ili koje se ne koriste • Ako je moguće, ukinuti default login i gues korisnike • Bez obzira što je potrebno, korisnicima ne dozvoljavati da se na bazu podataka interaktivno prijavljuju (preko SQL*PLUS-a)
Zaštititi računar na kojem se nalazi DBMS • Ne dozvoliti korisnicima da na računaru na kojem se nalazi DBMS rade druge poslove • Računar na kojem se nalazi DBMS treba fizički izolovati iza zaključanih vrata • Sve posete odeljenju u kojem se nalazi računar sa RDBM treba posebno beležiti
Upravljati accoun-ima i password-ima • Za DBMS servise treba koristiti korisničke account-e sa niskim privilegijama • Bazu podataka treba zaštiti account-ima sa jakim password-ima (Password treba da ima najmanje 8 karaktera, da sadrži velika i mala slova, brojeve, specijalne znake i kombinaciju nekih ključeva) • Treba pratiti pokušaje pogrešnog logovanja • Treba često proveravati pripadnost korisnika rolama • Accout-ima dodeljivati najmanje moguće privilegije • Ograničiti privilegije DBA account-u.

Slika 2.2 Tabela sa smernicama za osiguranje zaštite sistema baze podataka [Izvor: NM IT350 - 2020/ 2021.]

ZAŠTITA NA NIVOU APLIKACIJE

Generalno, uvek prvo treba obezbediti zaštitu na nivou RDBMS, pa ako se proceni da ona nije adekvatna, treba joj dodati i zaštitu na nivou aplikacije.

Zaštita na nivou RDBMS se može proširiti i zaštitom na nivou aplikacije. Generalno, uvek prvo treba obezbediti zaštitu na nivou RDBMS, pa ako se proceni da ona nije adekvatna, treba joj dodati i zaštitu na nivou aplikacije.

Zaštita na nivou aplikacije se može objasniti na primeru Internet aplikacija kod kojih se zaštita najčešće obezbeđuje preko Web servera. Izvršenje aplikacija na Web serverima predstavlja način da se osetljivi podaci ne prenose kroz mrežu i na taj način zaštite. Pretpostavimo da je Web aplikacija pisana tako da se na klik korisnika na stranici pregledača prema Web serveru a zatim i prema RDBMS šalje sledeći upit:

*SELECT * FROM EMPLOYEE;*

Ako se politikom zaštite na nivou aplikacije korisniku dozvoljava da pristupi samo svojim sopstvenim podacima, Web server bi mogao da doda sledeću WHERE klauzulu.
*SELECT * FROM EMPLOYEE*
WHERE EMPLOYEE.Name = '<% = SESSION ((„EmployeeName“))%>';

U ovakvom upitu će Web server u WHERE klauzuli popuniti ime zaposlenog.

Ovakvu zaštitu ne mora da obezbedi Web server već ona može biti realizovana kroz aplikativne programe ili napisana kao store procedura ili triger i izvršena od strane DBMS u odgovarajućem trenutku.

Poglavlje 3

Paralelna obrada baza podataka

PARALELNI ALGORITMI NAD RELACIJAMA

Suština paralelne obrade je korišćenje više procesora. Obično je broj procesora p veliki po nekoliko stotina ili hiljada.

Dok se mnoge baze podataka nalaze samo na jednoj mašini, postoje i baze podataka koje su distribuirane na više mašina. Takođe postoje i baze podataka koje su smeštene samo na jednom računaru sa više paralelnih procesora. Kada se obrada podataka vrši bilo paralelno bilo distribuirano, javljaju se mnogi implementacioni problemi koje je potrebno rešiti.

Operacije nad bazama podataka često zahtevaju dosta vremena i obuhvataju veliku količinu podataka pa se mogu unaprediti ukoliko se primeni paralelna obrada podataka. Suština paralelne obrade je **korišćenje više procesora.** Obično je broj procesora p veliki, po nekoliko stotina ili hiljada. Pretpostavka je da svaki procesor ima svoju lokalnu keš memoriju. U mnogim organizacijama, svaki procesor ima i svoju spoljašnju memoriju (po svakom procesoru postoji jedan ili više diskova, a u nekim arhitekturama i velika kolekcija diskova kojima procesor može direktno da pristupa).
Osim toga, u paralelnim obradama postoji mogućnost prenosa informacija između procesora (npr. svi procesori u jednom reku su obično povezani)

Paralelna arhitektura se može podeliti na tri grupe:

1. *Kod najčvršće povezane paralelne arhitekture se deli glavna memorija (eng.* Shared -Memory Machines*).*
2. *Kod slabo povezane arhitekture se deli spoljašnja memorija (disk) a ne glavna memorija (eng.* Shared-Disk Machines*) .*
3. *Arhitektura koja se često koristi kod baza podataka se naziva arhitektura kod koje se ništa ne deli (eng.* Shared-Nothing Machines*), kod koje se ne deli disk a procesori su povezani i dele se samo prenosom poruka.*

MAŠINA SA DELJIVOM MEMORIJOM

U ovoj arhitekturi, svaki procesor ima pristup do svih memorija svih drugih procesora.

Mašina sa deljivom memorijom (eng. Shared -Memory Machines):

U ovoj arhitekturi, svaki procesor ima pristup do svih memorija svih procesora. Na taj način,postoji samo jedan fizički adresni prostor za čitavu mašinu, umesto jednog adresnog

prostora za svaki procesor. Procesor nema svoju lokalnu keš memoriju (obično označenu sa M), ali ima direktan pristup do memorije drugih procesora kada je to potrebno. S druge strane, svaki procesor ima svoju lokalnu spoljašnju memoriju (disk) koju koristi kad god je to moguće. Velike mašine ove klase su tipa NUMA (engl. nonuniform memory access) i njima je potrebno više vremena da bi procesor pristupio podacima u memoriji koja pripada drugim procesorima nego što mu je potrebno da pristupi svojoj sopstvenoj memoriji ili memoriji procesora u njegovom lokalnom klasteru. Međutim, u ovoj arhitekturi, razlika u vremenu pristupa memoriji nije velika. Svi pristupi sekundarnoj memoriji, bez obzira gde se nalaze podaci troše mnogo više vremena nego što traje pristup kešu, tako da je kritičan problem da li su ili ne podaci koji su potrebni procesoru na svom sopstvenom disku.

MAŠINA SA DELJIVIM DISKOM

U ovoj arhitekturi, svaki procesor ima svoju sopstvenu memoriju kojoj se ne može direktno pristupiti iz drugih procesora

Mašina sa deljivim diskom (engl. Shared-Disk Machines)

U ovoj arhitekturi, svaki procesor ima svoju sopstvenu keš memoriju kojoj se ne može direktno pristupiti iz drugih procesora. Međutim, ovde se može pristupiti disku iz bilo kog procesora kroz komunikacionu mrežu. Kontroler diska razrešava potencijalno konkurentske zahteve koji dolaze iz različitih procesora.

Potrebno je da broj diskova i procesora bude identičan. Ova se arhitektura danas pojavljuje u dva oblika, zavisno od jedinice za transfer između diska i procesora:

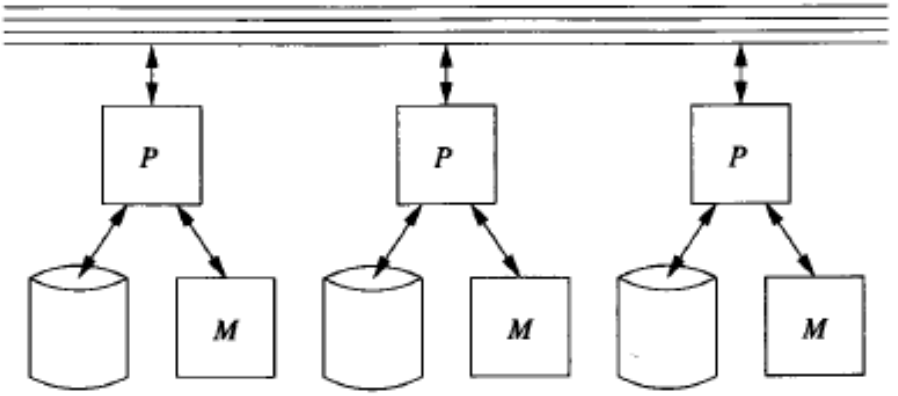
1. kao farma diskova koja se naziva network attached storage (NAS) i
2. kao storage area networks (SAN) koji prenosi blokove sa diska u i iz procesora.

MAŠINA KOD KOJE SE NIŠTA NE DELI

U ovoj arhitekturi, svi procesori imaju svoju sopstvenu memoriju ili svoj sopstveni disk ili diskove

Mašina kod koje se ništa ne deli (engl. Shared-Nothing Machines)

Ovde, svi procesori imaju svoju sopstvenu keš memoriju i svoj sopstveni disk ili diskove kao što je prikazano na slici 3.1. Čitava komunikacija se obavlja preko mreže, od procesora do procesora. Na primer, ako jedan procesor P želi da pročita torke sa diska drugog procesora Q, tada procesor P šalje poruku procesoru Q tražeći od njega podatke. Q dobija torke sa svog diska i isporučuje ih preko mreže drugom porukom koju prima P. Kao što je napomenuto, shared-nothing arhitektura se najčešće koristi u bazama podataka. Ona je relativno jeftina.



Slika 3.1 Mašina kod koje se ništa ne deli [Izvor: NM IT350 - 2020/2021.]

Poglavlje 4

Izvršenje SQL operacija u SNM arhitekturi

OPERACIJE NAD JEDNOM RELACIJOM U SNM KROZ PRIMERE

U ovom slučaju je potrebno razmotriti na koji način je najbolje sačuvati podatke. Ako imamo p procesora, torke svake relacije R treba da budu ravnomerno podeljene na diskove p procesora

Primer 1: *Paralelni algoritam u slučaju shared-nothing machine na primeru naredbe SELECT kojom se vrši selekcija (izdaju redovi tabele prema definisanom uslovu) nad jednom relacijom:*

Najpre je potrebno razmotriti na koji način je najbolje sačuvati podatke. Korisno je podatke distribuirati na što veći broj diskova. Pretpostavka je da imamo jedan disk po procesoru. Tako, ako imamo p procesora torke svake relacije R će biti podeljene na diskove p procesora.

Da bismo izvršili selekciju nad relacijom R za uslov C, možemo koristiti svaki od procesora kako bi ispitao torke relacije R na svom sopstvenom disku. Svaki procesor treba da pronađe torke koje zadovoljavaju uslov C i da ih kopira na izlaz. Kako bi se sprečila komunikacija između procesora, nađene torke t čuvamo u istom procesoru koji torke t ima na svom disku. Na taj način je rezultat selekcije podeljen između procesora kao što je podeljena i relacija R.

Kako selekcija torki R koje zadovoljavaju uslov C može biti ulazna relacija u drugu operaciju i kako želimo da skratimo proteklo vreme i što manje držimo sve procesore zauzete sve vreme, potrebno je da selekcija relacije R bude podeljena jednako među procesorima.

Primer 2*: Paralelni algoritam u slučaju shared-nothing machine na primeru naredbe SELECT kojom se vrši projekcija (izdvajaju se neke kolone tabele):*

Ukoliko bi radili projekciju a ne selekciju, tada bi broj torki u projekciji u svakom procesoru bio isti kao i broj torki R u tom procesoru. Na taj način, ako bi se R distribuirala ujednačeno po procesorima, ujednačeno bi bile distribuirane i njene projekcije. Međutim, selekcija može radikalno promeniti distribuciju torki u rezultatu u poređenju sa distribucijom relacije R.

PRIMERI OPERACIJA NAD JEDNOM RELACIJOM U SNM

Da bi se izbegla situacija da se sve torke nađu u jenom procesoru, najbolje je koristiti hešing funkciju h koja bi se primenila na sve komponente torki

Primer 3: *Pretpostavimo da treba izvršiti selekciju tako da se nađu sve torke relacije R čije su vrednosti atributa a=10. Pretpostavimo takođe da je relacija R podeljena prema vrednosti atributa a. Tada su sve torke relacije R sa a=10 u jednom procesoru tj. čitava selekcija te relacije za vrednost atributa 10 je u samo jednom procesoru.*

Da bi se izbegao problem koji je opisan u prethodnom primeru, treba pažljivo razmisliti o politici podele zapamćene relacije među procesorima. Verovatno najbolje što se može uraditi je koristiti *heš funkciju h* koja bi se primenila na sve komponente torki tako da promena jedne komponente torke t može promeniti h(t).

Na primer, ako želimo da izdvojimo B skupova torki, svakoj komponenti možemo dodeliti integer broj između 0 i B-1, podeliti rezultat sa B i ostatak sačuvati kao broj skupa. Ako je B takođe i broj procesora, tada svaki procesor možemo dodeliti jednom skupu i pridružiti mu sadržaj skupa.

Ako za distribuciju torki relacije R koristimo heš funkciju kako je to prethodno opisano, tada ćemo duple torke relacije R staviti u isti procesor. Selekciju nad delovima od R koji se nalaze u svakom procesoru, možemo proizvesti paralelno primenom standardnog algoritma koji se izvršava nad jednim procesorom.

Primer 4: *Grupisanje i agregacija nad relacijom R*
Da bismo izvršili grupisanje i agregaciju nad relacijom R, potrebno je da distribuiramo torke relacije R korišćenjem hešing funkcije h koja zavisi samo od atributa za grupisanje u listi L. Ako svaki procesor ima sve torke koje odgovaraju skupovima od h, tada možemo izvršiti operaciju agregacije tih torki lokalno, korišćenjem jedno-procesorskog algoritma.

OPERACIJE NAD DVE RELACIJE R I S U SNM

Ako za distribuciju torki relacija R i S koristimo istu hešing funkciju, tada prilikom izvršenja ovih operacija, možemo raditi paralelno nad delovima R i S koji se nalaze u svakom procesoru

Primer 5: *Pretpostavimo da treba izvršiti operacije UNIJA, PRESEK i RAZLIKA nad dve relacije R i S:*

Ukoliko imamo dve relacije R i S i ukoliko koristimo istu heš funkciju da bismo distribuirali torke relacija R i S, tada prilikom izvršenja ovih operacija možemo raditi paralelno nad delovima R i S koji se nalaze u svakom procesoru.

Međutim, ako pretpostavimo da R i S nisu distribuirane korišćenjem iste hešing funkcije a da mi želimo da dobijemo njihovu uniju UNIJU, PRESEK ili RAZLIKU, u tom slučaju, moramo prvo da kopiramo sve torke R i S i distribuiramo ih korišćenjem jedinstvene heš funkcije.
Heš funkcija se izvršava na već opisan način, ali kada se napuni bafer koji odgovara skupu torki u procesoru *j*, umesto da se smesti na disk *j*, sadržaj bafera se prenosi u procesor *i*. Ukoliko u glavnoj memoriji postoji mesta za više blokova za smeštanje skupova torki, možemo sačekati da se torkama *i* napuni nekoliko bafera pre nego što se one smeste u procesor *i*. Na taj način, procesor *i* prima sve torke R i S koje pripadaju torkama *j*.

U drugom koraku, svaki procesor pravi uniju/presek/razliku torki iz R i S. Kao rezultat, relacije R i S će biti distribuirane kroz sve procesore. Ako funkcija heširanja h randomizira raspoređivanje torki po skupovima, tada možemo očekivati da u svakom procesoru bude približno isti broj torki R i S.

Primer 6: *Operacija spajanja JOIN nad dve relacije R i S*

Da bi napravili spajanje (join) R (X , Y) i S (Y ,Z): potrebno je da heširamo torke relacija R i S na broj skupova – grupa koji je jednak broju procesora. Međutim, heš funkcija h koju koristimo mora da zavisi samo od atributa Y, ne od svih atributa, tako da torke koje se spajaju se uvek šalju u isti skup-grupu. Kao i kod unije, mi šaljemo torke skupa i u procesor i. Tada možemo izvršiti join u svakom procesoru korišćenjem bilo kog jedno procesorskog algoritma za spajanje.

VIDEO

Objašnjenje SNA -What is SHARED NOTHING ARCHITECTURE? What does SHARED NOTHING ARCHITECTURE mean?

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 5

Distribuirane baze podataka

KARAKTERISTIKE DISTRIBUIRANIH BAZA

Velika količina poruka i podataka koji se prenose preko komunikacione mreže.

Sistem za upravljanje distribuiranim bazama podataka je i sam distribuiran, pa se označava sa DDBMS – distribuirani DBMS. *Distribuirane baze podataka i DDBMS zajedno obrazuju sistem distribuiranih baza podataka*. Za obe vrste sistema koristi se i kraći termin distribuirani sistem, kada je iz konteksta jasno na koju vrstu sistema se termin odnosi.

Neke od poznatih prototipskih implementacija DDBMS-a su SDD-1 (eng.System for Distributed Databases Computer Corporation of America), R* (eng.IBM Research Lab.) i distribuirani INGRES (eng.University of Califomia, Berkeley). Najpoznatiji komercijalni DDBMS su INGRES/STAR, ORACLE7/DDO (eng."distributed database option"), DB2/DDF (eng."distributed database facility"). Svi nabrojani sistemi su relacioni, s obzirom da nerelacioni DDBMS ne mogu biti uspešni.

Osnovna karakteristika distribuiranih sistema je velika količina poruka i podataka koji se prenose preko komunikacione mreže. **Zato je glavni cilj u postizanju efikasnosti distribuiranih sistema – *smanjenje mrežne komunikacije.*** Ovaj cilj se projektuje na sve ključne funkcije DBMS; one moraju da se razmatraju iz novog ugla, a problemi u njihovoj realizaciji zahtevaju nova rešenja.

Neformalno govoreći, distribuirana baza podataka je baza podataka koja se ne nalazi u celosti na jednoj fizičkoj lokaciji (na jednom računaru), već je razdeljena na više lokacija koje su povezane komunikacionom mrežom. Svaka lokacija, koja se zove i čvor komunikacione mreže, poseduje svoj sopstveni, autonomni sistem za upravljanje bazama podataka, sa sopstvenom kontrolom, upravljačem transakcija i oporavka od pada, i ostalim značajnim funkcijama, a ima i svoj centralni procesor i ulazno/izlazne uređaje.

ŠTA JE POTREBNO OBEZBEDITI U DISTRIBUIRANIM BAZAMA PODATAKA?

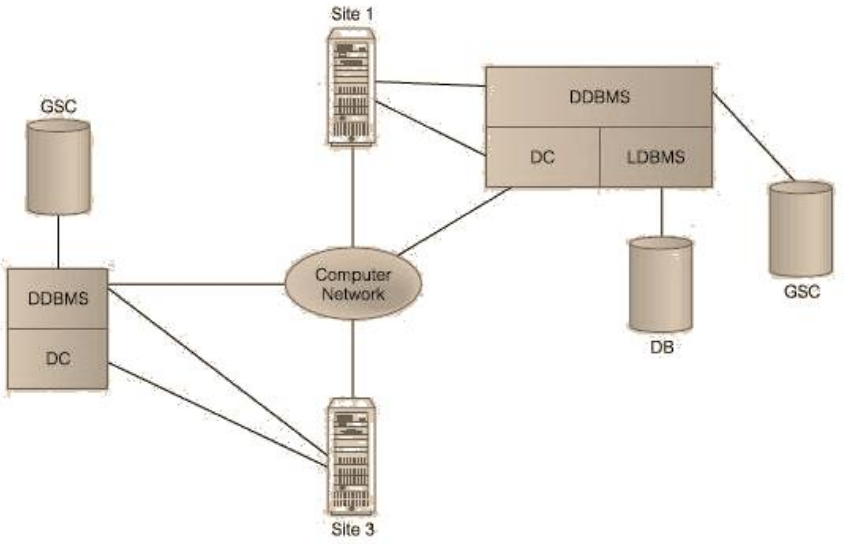
Korisnik i njegov program ne treba da znaju na kojoj se lokaciji u distribuiranom sistemu nalaze podaci koji su im potrebni.

Osnovna pretpostavka za uspešnost sistema za upravljanje distribuiranim bazama podataka je *nevidljivost lokacije*. Naime, ovaj sistem treba da obezbedi još jedan nivo

fizičke nezavisnosti podataka: korisnik i njegov program ne treba da znaju na kojoj se lokaciji u distribuiranom sistemu nalaze podaci koji su im potrebni. Za korisnika, distribuirani sistem treba da izgleda identično sa nedistribuiranim (centralizovanim), tj. pristup podacima treba da je isti kao da su svi podaci smešteni u lokalnom čvoru korisnika. Sistem odlučuje (ne uključujući korisnika) o tome da li će potrebne podatke, ukoliko su na udaljenoj lokaciji, doneti na lokalni čvor za obradu, ili će obradu preneti na udaljenu lokaciju.

Sistem distribuiranih baza podataka se može predstaviti grafom čiji su čvorovi – lokacije, a grane – komunikacione linije. Svaka lokacija sadrži centralni procesor (CP), lokalnu bazu podataka (BP) i pripadni DBMS, kao i skup klijentskih računara. Dve lokacije su povezane ako među njima postoji direktna veza koja funkcioniše, ili ako postoji treća lokacija koja je sa njima u direktnoj vezi.

Na različitim lokacijama, *lokalni sistemi za upravljanje bazama podataka mogu biti različiti*. Tada se *DDBMS naziva heterogenim*. Ako su *DBMS na svim lokacijama isti, DDBMS se naziva homogenim.*



Slika 5.1 Povezanost distribuirane baze podataka [Izvor: Autor]

PREDNOSTI DISTRIBUIRANIH BAZA PODATAKA

To su: lokalna autonomija podataka, veći kapacitet i postupni rast, pouzdanost i raspoloživost, efikasnost i fleksibilnost.

1.Lokalna autonomija podataka, upravljanja i kontrole: Okruženje u kome se distribuirane baze primenjuju, obično je i samo logički i fizički distribuirano (npr. univerzitet, fakulteti, odseci, odeljenja; centralna biblioteka, matične biblioteke, ogranci; ministarstvo, regionalni centri, itd.). Distribuiranje baza podataka kao i sistema za upravljanje njima, omogućuje pojedinim grupama da lokalno kontrolišu sopstvene podatke, uz mogućnost pristupa podacima na drugim lokacijama kada je to potrebno;

2.Veći kapacitet i postupni rast: Čest razlog za instaliranje distribuiranog sistema je nemogućnost jednog računara da primi i obrađuje sve potrebne podatke. U slučaju da

potrebe nadmaše postojeće kapacitete, dodavanje čvora distribuiranom sistemu je znatno jednostavnije nego zamena centralizovanog sistema većim;

3. Pouzdanost i raspoloživost: Distribuirani sistemi mogu da nastave svoje funkcionisanje i kada neki od čvorova privremeno izgube funkcionalnost

4.Efikasnost i fleksibilnost: Podaci su fizički blizu onome ko ih stvara i koristi, pa je znatno smanjena potreba za udaljenom komunikacijom.

Jedan jednostavan specijalni slučaj distribuiranog sistema je klijent/server sistem. To je distribuirani sistem u kome su neki čvorovi klijenti a neki serveri, pri čemu su na serverima smešteni podaci (i sistemi za upravljanje podacima) a, na klijentima se izvršavaju aplikacije. Korisnik, odnosno aplikacija vodi računa o tome na kom serveru su smešteni relevantni podaci, što znači da u ovim sistemima nije ostvarena nevidljivost lokacije.

Poglavlje 6

Postizanje efikasnosti u distribuiranim BP

KAKO POSTIĆI EFIKASNOST U DISTRIBUIRANIM BAZAMA PODATAKA?

Smanjivanjem mrežne komunikacije.

Da bi se postigla efikasnost DDBMS-a, sve njegove komponente treba realizovati na način koji smanjuje mrežnu komunikaciju.

Najznačajniji problemi koje tom prilikom treba rešiti su:

1. *fragmentacija podataka*
2. *distribuirana obrada upita*
3. *distribuirano (preneto) ažuriranje*
4. *upravljanje katalogom*
5. *distribuirano izvršenje skupa transakcija, što uključuje konkurentnost, integritet, oporavak i protokole kompletiranja transakcija.*

Podaci u distribuiranom sistemu mogu biti podeljeni (eng.partitioned) ili ponovljeni (eng.replicated) u fizičkoj memoriji.

U slučaju ponavljanja podataka (eng. replicated), **jedan logički objekat može imati više fizičkih reprezentacija (veći broj kopija) na većem broju lokacija.** Ponovljenost podataka povećava raspoloživost podataka i efikasnost pristupa podacima, ali u značajnoj meri usložnjava ažuriranje podataka, koji moraju biti konzistentni u svim svojim kopijama. **Složenost koju nosi sobom strategija ponavljanja podataka mora biti sakrivena od korisnika, tj. mora biti obezbeđena nevidljivost ponavljanja podataka.**

U slučaju deljenja podataka (eng. partitioned), logički skup podataka (skup podataka sa logičkog nivoa) treba na neki način podeliti, a zatim delove – fragmente (eventualno sa ponovljenim kopijama) razdeliti po raznim lokacijama. Logički skup podatka u relacionom sistemu je relacija, a prirodni fragment relacije (koji je opet relacija) jeste neki njen podskup definisan uslovom projekcije i restrikcije. **Fragmentacija mora biti izvedena tako da se spajanjem fragmenata može dobiti polazna relacija**.

FRAGMENTACIJA PODATAKA

Primeri definisanja fragmenata.

***Primer 1**: Relacija, predstavljena tabelom IZDAVACI se može SQL-om podeliti na sledeće fragmente:*

```
DEFINE FRAGMENT SRB1 AS SELECT I_SIF, NAZIV
    FROM IZDAVACI WHERE DRZAVA = 'Srbija'

DEFINE FRAGMENT SRB2 AS SELECT I_SIF, STATUS, DRZAVA
    FROM IZDAVACI WHERE DRZAVA = 'Srbija'
```

I_SIF	NAZIV	STATUS	DRZAVA
i1	Prosveta	30	Srbija
i2	Addison Wesley Publ. Comp	20	SAD
i3	Dečije novine	10	Srbija
i4	Wiley Computer Publishing	30	SAD

Slika 6.1 Relacija IZDAVACI [Izvor: Autor]

```
DEFINE FRAGMENT AM1 AS SELECT I_SIF, NAZIV
    FROM IZDAVACI WHERE DRZAVA = 'SAD'

DEFINE FRAGMENT AM2 AS SELECT I_SIF, STATUS, DRZAVA
    FROM IZDAVACI WHERE DRZAVA = 'SAD'
```

Najčešće primenjivana tehnika u obezbeđivanju očuvanja informacija pri fragmentaciji podataka jeste uvođenje sistemskih identifikatora n-torki (tzv. nametnutih ključeva) kao primarnih ključeva, koji se pamte uz svaki deo pojedine n-torke, i omogućuju njenu rekonstrukciju. Ova tehnika je primenjena u distribuiranim sistemima R* i SDD-1.

DISTRIBUIRANA OBRADA UPITA

Ima za cilj minimizirati cenu obrade i vreme za koje će korisnik dobiti odgovor.

Distribuirana obrada upita **podrazumeva distribuiranu optimizaciju kao i distribuirano izvršavanje upita. Strategije optimizacije upita nad distribuiranom bazom podataka imaju za cilj da minimizuju cenu obrade i vreme za koje će korisnik dobiti odgovor.**

U troškovima obrade najveću stavku čine troškovi mrežne komunikacije, tj. prenosa podataka kroz mrežu, dok su troškovi komunikacije sa ulazno/izlaznim uređajima i korišćenja procesora niži za nekoliko redova veličine. ***Zbog toga je veoma značajno, u zavisnosti od propusnosti mreže (količine podataka koju može da primi u sekundi) i vremena kašnjenja, pravilno odabrati relacije i njihove fragmente koji će biti prenošeni sa jedne lokacije na drugu u cilju obrade upita (globalna optimizacija).***

Razlog za prenošenje podataka može biti to što su podaci na lokaciji različitoj od one na kojoj se postavlja upit, ili što u upitu učestvuje veći broj relacija sa različitih lokacija. **Izbor strategije za izvršenje operacija na jednoj lokaciji poznat je kao lokalna optimizacija.**

Ako se n relacija R_1, R_2,..., R_n koje učestvuju u upitu nalaze na k različitih lokacija l_1,l_2,...,l_k, pri čemu je svaka relacija R_i – u celosti na lokaciji l_j, onda se osnovna strategija distribuirane obrade upita sastoji od sledeća dva koraka:

1. **Maksimalna redukcija svake relacije na njenoj lokaciji** (lokalna restrikcija i projekcija na atribute spajanja i izlazne atribute). Prenošenje dobijenih relacija na jednu lokaciju, ili na više lokacija, redom, na kojima je moguće izvršiti pojedinačna spajanja i projekciju na izlazne atribute.
2. Za drugi korak osnovne strategije vezana je **odluka o tome koje se relacije prenose i na koje lokacije**. Ta odluka se donosi na osnovu procene količine podataka koji se prenose između lokacija u svakom pojedinačnom slučaju; izbor relacija i lokacija vrši se tako da minimizuje protok podataka kroz mrežu.

PRIMER DISTRIBUIRANE OBRADE UPITA

Kako rasporediti relacije i gde izvršiti upit?

K_SIF	NASLOV	OBLAST
k1	Osma ofanziva	roman
k2	Nemam više vremena	poezija
k3	Pionirska trilogija	roman
k4	Srpsko-engleski rečnik	leksikografija
k5	An Introduction to Database Systems	računarstvo
k6	Tražim pomilovanje	poezija

Slika 6.2 Relacija K (KNJIGA) [Izvor: Autor]

I_SIF	NAZIV	STATUS	DRZAVA
i1	Prosveta	30	Srbija
i2	*Addison Wesley Publ. Comp*	20	SAD
i3	Dečije novine	10	Srbija
i4	*Wiley Computer Publishing*	30	SAD

Slika 6.3 Relacija I (IZDAVACI) [Izvor: Autor]

K_SIF	I_SIF	IZDANJE	GODINA	TIRAZ
k1	I1	2	1965	10000
k2	i1	2	1974	7000
k3	i1	I	1975	10000
k4	i1	2	1979	10000
k5	i2	4	1986	5000
k6	I3	1	1966	3000
k6	i4	3	1988	5000

Slika 6.4 Relacija KI (KNJIGA/IZDAVACI) [Izvor: Autor]

Primer: *Neka se relacije K (KNJIGA) na slici 6.2, I (IZDAVACI) na slici 6.3, KI (KNJIGA/IZDAVACI) na slici 6.4, nalaze na lokacijama l1, l2, l3, redom, i neka se na lokaciji l1 postavi sledeći SQL upit (naći naslove romana srpskih izdavača kao i brojeve izdanja i nazive tih izdavača):*

```
SELECT K.NASLOV, I.NAZIV, KI.IZDANJE
    FROM K, KI, I
WHERE I.DRZAVA='Srbija '
AND K.OBLAST='roman'
AND K.K_SIF=KI.K_SIF
AND KI.I_SIF=I.I_SIF;
```

U upitu su prisutne operacije restrikcije nad relacijama K i I, dve operacije spajanja, jedna nad relacijama K i KI a druga nad relacijama KI i I, kao i operacija projekcije na izlazne atribute NASLOV, NAZIV i IZDANJE.

PRENOS RELACIJA NA DRUGE LOKACIJE

Pojedinačna spajanja vršiti na različitim lokacijama.

Moguće je pojedinačna spajanja vršiti na različitim lokacijama.

Primer 1*: Relacija I' može se preneti na lokaciju l3, tamo izvršiti spajanje sa relacijom KI, uz projekciju na izlazne atribute celog upita i atribute spajanja preostalog upita:*
SELECT I'.NAZIV, KI.K_SIF, KI.IZDANJE FROM I', KI
WHERE I'.I_SIF=KI.I_SIF;

Neka je dobijena relacija – rezultat IKI(NAZIV, K_SIF, IZDANJE). Sada se relacija IKI može preneti na lokaciju l_1. i tamo izvršiti spajanje sa relacijom K' i projekcija na izlazne atribute:
SELECT K'.NASLOV, IKI.NAZIV, IKI.IZDANJE FROM IKI, K'
WHERE IKI.K_SIF=K'.K_SIF;

(rezultat se nalazi na lokaciji l_1).

Prethodni upit spajanja predstavljaju jednu mogućnost pojedinačnih spajanja na različitim lokacijama. Ti upiti se mogu predstaviti parom izraza relacione algebre sa naznakom lokacije na kojoj se izvršavaju,
*IKI = (I * KI) [NAZIV, K_SIF, IZDANJE] (l3)*
*REZ = (IKI * K') [NASLOV, NAZIV, IZDANJE] (l1).*

Analogne mogućnosti pojedinačnih spajanja na različitim lokacijama opisuju se i sledećim parovima izraza relacione algebre, sa naznakom lokacija:
*IKI = (I * KI) [NAZIV, K_SIF, IZDANJE] (l2)*
*REZ = (IKI * K') [NASLOV, NAZIV, IZDANJE] (l1),*
*KKI = (K' * KI) [NASLOV, I_SIF, IZDANJE] (l1)*
*REZ = (KKI * I') .[NASLOV, NAZIV, IZDANJE] (l2),*
*KKI = (K' * KI) [NASLOV, I_SIF, IZDANJE] (l3)*
*REZ = (KKI * I') [NASLOV, NAZIV, IZDANJE] (l2).*

REDUKCIJA RELACIJA NA NJIHOVIM LOKACIJAMA

Treba izvršiti maksimalnu redukciju svake relacije na njenoj lokaciji.

Neka n relacija R_1, R_2 , ..., R_n učestvuju u upitu, i pri tom su smeštene na lokacijama l_1, l_2,...,l_k , i važi da je relacija R_j u celosti na lokaciji l_j.

Postupak koji treba sprovesti je sledeći:
1. **maksimalna redukcija svake relacije na njenoj lokaciji** (lokalna restrikcija i projekcija na atribute spajanja i izlazne atribute);
2. **prenošenje dobijenih relacija na jednu lokaciju, ili na više lokacija, redom, na kojima je moguće izvršiti pojedinačna spajanja i projekciju na izlazne atribute.**

U našem primeru, primena ovog postupka izgleda ovako:

1. ***Prvi korak osnovne strategije distribuiranog izvršavanja ovog upita: sastoji se u izvršavanju, na lokaciji l1, jednorelacionog upita:***

SELECT NASLOV, K_SIF FROM K WHERE OBLAST='roman';

i u izvršavanju, na lokaciji l_2, jednorelacionog upita:

SELECT NAZIV, I_SIF FROM I WHERE DRZAVA='Srbija';

Prvi upit proizvodi relaciju K'(NASLOV, K_SIF) na lokaciji l_1 , a drugi upit proizvodi relaciju I'(NAZIV, I_SIF) na lokaciji l_2.

2. a)***Pred drugi korak osnovne strategije postavlja se sledeći izbor:***

1. *preneti relacije K' i I' na lokaciju l3*
2. *relacije K' i KI na lokaciju l2,*
3. *odnosno relacije I' i KI na lokaciju l1*

i tamo izvršiti oba spajanja i projekciju na izlazne atribute:
SELECT K'.NASLOV, I'.NAZIV, KI.IZDANJE FROM K', I', KI
WHERE K'.K_SIF=KI.K_SIF AND KI.I_SIF=I'.I_SIF;

a zatim rezultat preneti na lokaciju l1.
b) **Izvršiti pojedinačna spajanja na različitim lokacijama** , npr. na lokaciju l_3 preneti relaciju I' i izvršiti spajanje sa relacijom KI, uz projekciju na izlazne atribute celog upita i atribute spajanja preostalog upita:

```
SELECT I'.NAZIV, KI.K_SIF, KI.IZDANJE
FROM I', KI
WHERE I'.I_SIF=KI.I_SIF
```

DODATNO POJAŠNJENJE REDUKCIJA RELACIJA

Daju se analogne mogućnosti

Dobijena relacija – rezultat IKI(NAZIV, K_SIF, IZDANJE) – relaciju IKI preneti na lokaciju l $_1$, i tamo izvršiti spajanje sa relacijom K'
i projekciju na izlazne atribute:
SELECT K'.NASLOV, IKI.NAZIV, IKI.IZDANJE
FROM IKI, K'
WHERE IKI.K_SIF=K'.K_SIF (rezultat se nalazi na lokaciji l $_1$)
Analogne mogućnosti
• IKI = (I' *KI) [NAZIV, K_SIF, IZDANJE] (l2)
• REZ = (IKI *K') [NASLOV, NAZIV, IZDANJE] (l1),
• KKI = (K' *KI) [NASLOV, I_SIF, IZDANJE] (l1)
• REZ = (KKI *I') [NASLOV, NAZIV, IZDANJE] (l2),
• KKI = (K' *KI) [NASLOV, I_SIF, IZDANJE] (l3)
• REZ = (KKI *I') [NASLOV, NAZIV, IZDANJE] (l2).

EFIKASNOST NABROJANIH VARIJANTI PRENOSA

Izračunavanje vrednosti vremena komunikacije.

Razmotrimo efikasnost nabrojanih varijanti koraka 2 osnovne strategije distribuirane obrade upita.

Neka su veličine relacija date sa:
K (KJSIF, NASLOV, OBLAST) – 10 000 n-torki
I (I_SIF, NAZIV, STATUS, DRZAVA) – 1 000 n-torki
KI (K_SIF, I_SIF, IZDANJE, GODINA, TIRAZ) – 50000 n-torki

i neka svaka n-torka zauzima po 100 bitova. Neka je, dalje, procenjeno da ima 1000 romana u relaciji K, 100 srpskih izdavača u relaciji I i 20000 srpskih izdanja u relaciji KI. Ako se još pretpostavi da je propusnost mreže 50000 bitova u sekundi, a vreme kašnjenja 0.1 sekunda, onda se za optimalnu strategiju proglašava ona koja ima minimalnu vrednost vremena komunikacije.

t = kašnjenje + (količina podataka za prenos / propusnost), tj. u našem slučaju:

t = (broj poruka * 0.1sec) + (broj bitova za prenos / 50000)sec.

Primena raznih varijanti mogućnosti navedenih pod 2.1, 2.2 I 2.3 rezultovala bi različitim ukupnim vremenima komunikacije.

Na primer, u slučaju 2.1 biće

*t = 0.2 + (1100*100) / 50000 = 2.4 sekunde (bez prenošenja rezultata na lokaciju l_1),*

dok u slučaju 2.2 dobijamo

t = 0.2 + (100 + 20000) 100 / 50000 = 40.4sec.*

Jedna od varijanti mogla bi da bude i prenošenje relacije KI na lokaciju l_2, spajanje sa relacijom I' i projekcija na atribute K_SIF, IZDANJE, NAZIV, zatim prenošenje rezultata na lokaciju l_1 i spajanje sa relacijom K'. U ovom slučaju ukupno vreme komunikacije je:

t = 0.2 + (70000) 100 / 50000 = 140.2 sekunde*

što je znatno više od prethodnih vremena.

ALGORITMI ZA IZVOĐENJE OPERACIJA SPAJANJA

Neki od njih se baziraju na operacijama polu-spajanja.

Na sličan način mogu se izračunati ukupna vremena komunikacije i u drugim varijantama, pri čemu je zadatak globalne optimizacije da izabere varijantu sa najmanjim vremenskim utroškom.

Znatno poboljšanje efikasnosti može se postići primenom sofisticiranih algoritama za izvođenje nekih operacija. Jedan od takvih algoritama, primenjen u sistemu SDD-1, odnosi se na izvršenje operacije spajanja i uključuje operaciju polu-spajanja. Rezultat operacije polu-spajanja (eng.semijoin) relacija R i S jednakje rezultatu spajanja tih relacija, projektovanom na atribute relacije R.

Tako, ako je potrebno izvršiti spajanje relacije R (sa lokacije l_1) i relacije S (sa lokacije l_2), umesto da se cela relacija R prenese na lokaciju l_2 i tamo spoji sa relacijom S, moguće je izvršiti sledeći niz radnji:

1. izračunati projekciju relacije S po atributu spajanja, na lokaciji l2 (rezultat je relacija TEMPI)
2. preneti relaciju TEMPl na lokaciju l1
3. izvršiti polu-spajanje relacije R i TEMP1 po atributu spajanja, na lokaciji l1 (rezultat je relacija TEMP2)
4. preneti relaciju TEMP2 na lokaciju l2,
5. izvršiti spajanje relacija TEMP2 i S po atributu spajanja, na lokaciji l2.

Ovaj niz radnji smanjuje količinu prenosa podataka samo ako je:

broj bitova (TEMP1) + broj bitova (TEMP2) < broj bitova (R), a nije efikasan ako se spajanje izvodi po primarnom ključu relacije R koji je strani ključ relacije S (npr. u slučaju relacija I', KI).

PRENETO AŽURIRANJE

Znači da ukoliko se jedan logički objekat nađe na većem broju lokacija, ažuriranje jednog logičkog objekta se mora preneti i na sve fizičke kopije tog objekta.

Kao što je već rečeno, ponavljanje podataka podrazumeva da jedan logički objekat (npr. relacija ili jedan njen fragment) mogu imati više fizičkih reprezentacija (kopija) na većem broju lokacija. Posledica ove ideje je da se, s obzirom na potrebu za konzistentnošću podataka u svim kopijama, ažuriranje jednog logičkog objekta mora preneti i na sve fizičke kopije tog objekta. Međutim, momentalno prenošenje ažuriranja na sve kopije može da onemogući (ili da nedopustivo dugo odloži) uspešno izvršenje ažuriranja u slučaju da je bilo koja od lokacija u padu; time ponavljanje podataka smanjuje umesto da povećava raspoloživost podataka.

Jedan široko prihvaćeni pristup prenošenju ažuriranja oslanja se na koncept primarne kopije, i sastoji se u sledećem postupku:

1. *jedna kopija svakog ponovljenog objekta proglašava se za primarnu kopiju tog objekta, pri čemu primarne kopije različitih objekata mogu biti na različitim lokacijama*
2. *operacija ažuriranja objekta smatra se logički izvršenom čim se izvrši ažuriranje primarne kopije tog objekta; ažuriranje ostalih kopija je sada u nadležnosti lokacije na kojoj je primarna kopija, ali se mora izvršiti pre kompletiranja transakcije.*

Ovaj postupak zahteva primenu protokola dvofaznog kompletiranja transakcije, koji se ne može uspešno sprovesti ako je bar jedna relevantna lokacija u padu, što je čest slučaj. Ukoliko se dopusti ažuriranje kopija i posle kompletiranja transakcije, ne može se garantovati konzistentnost podataka u svim njihovim kopijama. Ipak, neki komercijalni distribuirani sistemi pribegavaju tom rešenju jer potpuno poštovanje ažuriranja svih kopija pre kompletiranja transakcije, može bitno da poveća vreme obrade.

UPRAVLJANJE KATALOGOM

Katalog je sistemska baza podataka koja pored ostalog u slučaju distribuiranog sistema sadrži podatke o lokacijama na koje su podaci razdeljeni i (eventualno) ponovljeni.

Katalog je sistemska baza podataka koja sadrži podatke o baznim relacijama, pogledima, indeksima, korisnicima, a u slučaju distribuiranog sistema, i o načinu i lokacijama na koje su podaci razdeljeni i (eventualno) ponovljeni. **Sam katalog u distribuiranom sistemu može biti centralizovan (samo na jednoj lokaciji), potpuno ponovljen (na svim lokacijama po jedna kopija kataloga), podeljen (na svakoj lokaciji je deo kataloga koji se odnosi na objektesa te lokacije) ili kombinovan (katalog je podeljen, ali na jednoj lokaciji postoji i jedna centralna kopija kompletnog kataloga).**

S obzirom na nedostatke koje ispoljava svaki od navedenih pristupa (zavisnost od centralne lokacije, visoka cena prenošenja ažuriranja kataloga ili skup pristup udaljenoj lokaciji), implementirani sistemi koriste druge strategije.

Primer: *U sistemu R* svaka lokacija sadrži sledeće kataloške informacije:*

1. slog kataloga za svaki objekat "rođen" na toj lokaciji (tj. čija je prva kopija kreirana na toj lokaciji); ovaj slog sadrži i informaciju o tekućoj lokaciji objekta (ako je premešten)

2. slog kataloga za svaki objekat koji je trenutno smešten na toj lokaciji

3. tabelu sinonima za svakog korisnika prijavljenog na toj lokaciji, koja preslikava sinonime objekata, definisane iskazom kreiranja sinonima, u sistemske identifikatore objekata, jedinstvene u celom distribuiranom sistemu.

Sistemski identifikator objekta, koji se nikada ne menja (dok se objekat ne ukloni), sastoji se od identifikatora korisnika koji je kreirao prvu kopiju tog objekta, lokacije sa koje je kreirana, lokalnog imena objekta (koje mu je dao korisnik pri "rođenju") i lokacije na kojoj je "rođen". Pronalaženje objekta, kada je dato njegovo lokalno ime ili sinonim, počinje preslikavanjem lokalnog imena (automatski), odnosno sinonima (pregledanjem tabele sinonima), u sistemski identifikator objekta. Zatim se, prema sistemskom identifikatoru, nalazi lokacija rođenja objekta, pristupa joj se i u slogu koji odgovara tom objektu, pronalazi se lokacija na kojoj je trenutno smešten. U sledećem koraku pristupa se (u opštem slučaju udaljenoj) lokaciji na kojoj je objekat smešten, i lokalnim operacijama pronalazi objekat.

Poglavlje 7

Pokazna vežba

NAČIN ORGANIZACIJE POKAZNIH VEŽBI

Vežba je organizovana kroz uvod deo i deo za samostalni rad studenata

Vežba je organizovana kroz uvod deo i deo za samostalni rad studenata.

1. U uvodnom delu se daje pokazni primer koji studentima treba da pomognu u samostalnom rešavanju zadataka.
2. Zadatke koji su zadati za samostalni rad student samostalno rešava uz pomoć asistenta.

7.1 Pokazni primer - 1. deo

KARAKTERISTIKE DISTRIBUIRANIH BAZA PODATAKA (8 MIN.)

Razdeljena je na više lokacija koje su povezane mrežom. Svaka lokacija, ima svoj vlastiti, autonomni sistem za upravljanje sa vlastitom kontrolom.

Distribuirana baza podataka je baza podataka koja se ne nalazi u celini na jednom računaru, već je razdeljena na više lokacija koje su povezane mrežom.

Svaka lokacija, koja se zove i čvor mreže, ima svoj vlastiti, autonomni sistem za upravljanje bazama podataka, sa vlastitom kontrolom, upravljačem transakcija i opravka od pada, i ostalim važnim funkcijama, a ima i svoj procesor i ulazno/izlazne uređaje.

Aplikacije istovremeno pristupaju i menjaju podatke na više različitih baza podataka u mreži, gde mreža može biti LAN ili WAN.

Osnovna svojstva distribuiranih baza podataka:

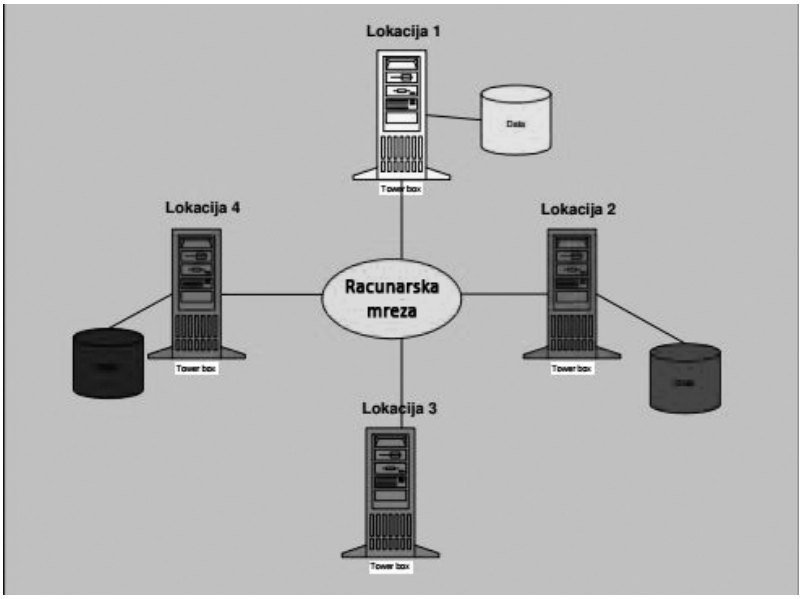
1. *Skup logički povezanih deljivih podataka*
2. *Podaci su razdvojeni na više fragmenata*
3. *Fragmenti se mogu replicirati*
4. *Fragmenti/Replikacije pripadaju lokacijama*
5. *Lokacije su povezane komunikacijskom mrežom*

6. *Podaci na svakoj lokaciji su pod nadzorom DBMS-a*
7. *DBMS na svakoj lokaciji može upravljati lokalnim aplikacijama autonomno*
8. *Svaki DBMS učestvuje u najmanje jednoj globalnoj aplikaciji.*

PREDNOSTI DISTRIBUIRANIH BAZA PODATAKA (8 MIN.)

Fleksibilnost i efikasnost, lokalna autonomija podataka, veći kapacitet i postupni rast, pouzdanost i raspoloživost.

1. **Fleksibilnost i efikasnost**: Podaci su fizički blizu onome ko ih stvara i koristi. Smanjena potreba za udaljenom komunikacijom.
2. **Lokalna autonomija podataka, upravljanja i kontrole**. Okruženje u kojem se distribuirane baze podataka primenjuju je i samo distribuirano (npr. fakulteti, zavodi...). Distribuiranje baza podataka kao i sistema za upravljanje njima, omogućava pojedinim grupama da lokalno kontrolišu vlastite podatke, uz mogućnost pristupa podacima na drugim lokacijama kada je to potrebno.
3. **Veći kapacitet i postupni rast.** Čest razlog za instaliranje distribuiranog sistema je nemogućnost jednog računara da primi i obrađuje sve potrebne podatke. U slučaju da potrebe nadmaše postojeći kapacitet, dodavanje čvora distribuiranom sistemu je znatno jednostavnije nego zamena sistema većim.
4. **Pouzdanost i raspoloživost.** Distribuirani sistemi mogu nastaviti svoje funkcioniranje i kada neki od čvorova izgube funkcionalnost.



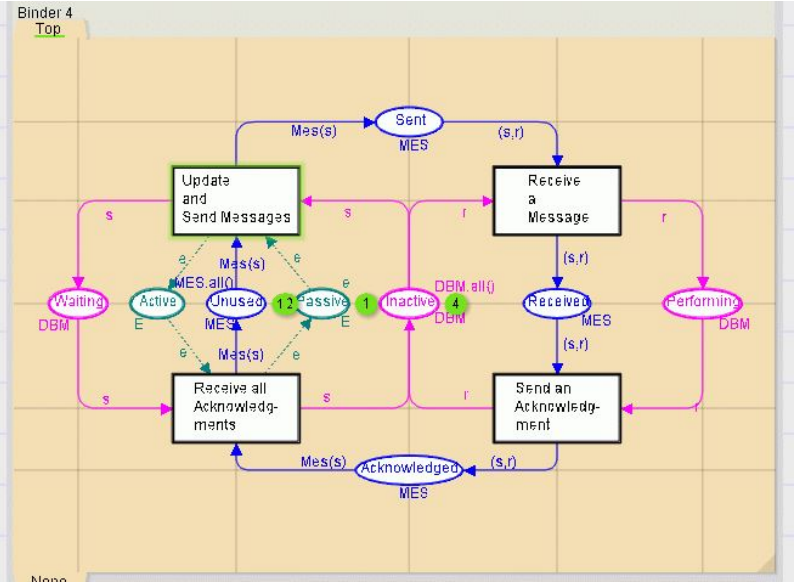
Slika 7.1.1 . Primer distribuirane baze [Izvor: NM IT350 - 2020/2021.]

PRIKAZ PRIMERA DISTRIBUIRANIH BAZA PODATAKA (7 MIN.)

Primer pokazuje komunikaciju između upravljača baze podataka u distribuiranim sistemima.

Na slici 8.2 je prikazan primer koji pokazuje komunikaciju između upravljača baze podataka u distribuiranim sistemima. Upravljači bi trebalo da održavaju podatke svojih baza kako bi ostali identični. Svako ažuriranje mora biti praćeno na svim menadžerima i mora da ih tera da obavljaju slično ažuriranje.

Primer je preuzet iz članka [AMAP1997] K.Jensen-a.



Slika 7.1.2 Komunikacija između upravljača baze podataka u distribuiranim sistemima. [Izvor: NM IT350 - 2020/2021.]

HETEROGENI DISTRIBUIRANI SISTEMI (7 MIN.)

Pored nezavisnosti pristupa podacima i obrade podataka od lokacije, heterogeni distribuirani sistem mora da obezbedi i nezavisnost od SUBP na pojedinim lokacijama.

Pretpostavka o homogenosti DSUBP, tj. pretpostavka da sve lokacije u DSUBP poseduju isti DBMS pokazuje se kao prejako ograničenje u današnjim uslovima, kada značajne količine podataka i aplikacija postoje na raznim računarima, pod različitim operativnim sistemima i pod kontrolom različitih DBMS. Potreba za istovremenim pristupom ovakvim podacima unutar jedne aplikacije, ili čak i jedne transakcije, postavlja zahtev pred proizvođače DBMS da obezbede podršku heterogenim DDBMS. To znači da, pored nezavisnosti pristupa podacima i obrade podataka od lokacije, fragmentacije, ponavljanja podataka, mašine, operativnog sistema i mrežnog protokola, heterogeni distribuirani sistem mora da obezbedi i nezavisnost od SUBP na pojedinim lokacijama.

To znači da, pored nezavisnosti pristupa podacima i obrade podataka od lokacije, fragmentacije, ponavljanja podataka, mašine, operativnog sistema i mrežnog protokola, heterogeni distribuirani sistem mora da obezbedi i nezavisnost od SUBP na pojedinim lokacijama.

Postoje dva bitno različita pristupa rešavanju ovog problema.

PRVI PRISTUP U REŠAVANJU PROBLEMA: SISTEM MULTI BAZA PODATAKA SMBP (8 MIN.)

Jedna od komponenti SMBP je jedinstveni jezik za kreiranje podataka i manipulisanje podacima koji su pod kontrolom heterogenih DBMS.

Jedan je izgradnja tzv. sistema multi-baza podataka, SMBP (eng.multi data base system). SMBP je programski sistem koji se sastoji od niza komponenti. Jedna od tih komponenti je jedinstveni jezik za kreiranje podataka i manipulisanje podacima koji su pod kontrolom heterogenih DBMS. DBMS omogućava da kroz njega korisnici i aplikacije mogu da komuniciraju sa raznorodnim sistemima.

Druga komponenta SMBP je globalni upravljač transakcija. DBMS obezbeđuje, pored lokalnih transakcija (nad jednim SUBP), i upravljanje globalnim transakcijama. Globalne transakcije se sastoje od većeg broja pod-transakcija koje se izvršavaju nad pojedinačnim (različitim) SUBP, i sa njihovog aspekta se ponašaju kao lokalne transakcije.

SMBP uključuje i skup servera, po jedan za svaki lokalni DBMS, koji se ponašaju kao veza između globalnog upravljača transakcija i lokalnog DBMS. Svaka globalna transakcija predaje globalnom upravljaču transakcija operacije čitanja i upisa, a ovaj može da ih preda na obradu lokalnim SUBP (preko odgovarajućih servera), da odloži ili da prekine transakciju.. Kada globalni upravljač transakcija odluči da kompletira globalnu transakciju, on upućuje komandu za kompletiranje lokalnim SUBP. Globalni upravljač transakcija je odgovoran i za funkcionalnost i održavanje svojstava globalnih transakcija (ACID svojstva, globalna linearizovanost izvršenja skupa transakcija, izbegavanje ili razrešavanje uzajamnog blokiranja, oporavak od sistemskih padova).

Oblast sistema multibaza je još uvek otvorena istraživačka oblast. Osnovni cilj ovih sistema je da kroz upravljanje globalnim transakcijama održavaju konzistentnost multibaze.

DRUGI PRISTUP U REŠAVANJU PROBLEMA: IZGRADNJA APLIKATIVNIH PROGRAMA, TZV. PROLAZA (7 MIN.)

Ostvaruje sledeće komponente i zadatke: protokol za razmenu informacija između SUBP1 i SUBP2, relacioni server za SUBP2, preslikavanja upitnih jezika dva sistema itd.

Drugi pristup heterogenim DDBMS je manje ambiciozan, ali i komercijalno zastupljeniji. Njegova suština je u izgradnji aplikativnih programa, tzv. *prolaza* (eng.gateway), na primer, sistema $DBMS_1$ prema sistemu $SUBP_2$, koji omogućuju korisniku $DBMS_1$ (na lokaciji l_1) da komunicira sa $SUBP_2$ (koji je na lokaciji l_2), na isti način na koji komunicira sa sistemom DBMS.

Realizacija aplikativnog programa prolaza sistema $DBMS_1$ prema sistemu $DBMS_2$ je u nadležnosti sistema $DBMS_1$, a program se izvršava nad sistemom $DBMS_2$. Na primer, ako je $DBMS_1$ – sistem Oracle a $DBMS_2$ – sistem DB2, onda bi prolaz sistema Oracle prema sistemu DB2 omogućio korisniku sistema Oracle da komunicira sa sistemom DB2, "lažno" predstavljajući sistem DB2 kao Oracle.

Aplikativni program *prolaz(gateway)* ostvaruje sledeće komponente i zadatke :

1. protokol za razmenu informacija između SUBP1 i SUBP2
2. relacioni server za SUBP2
3. preslikavanja između tipova podataka i upitnih jezika dva sistema
4. preslikavanje strukture kataloga sistema SUBP2 u strukturu kataloga sistema SUBP1
5. učešće u dvofaznom protokolu kompletiranja transakcija
6. doslednu primenu mehanizma zaključavanja, itd.

7.2 Pokazni primer - 2. deo

VRSTE NAPADA NA BAZE PODATAKA (10 MIN.)

Napadi koji se izvršavaju na baze podataka mogu biti spoljašnji i unutrašnji.

Spoljni napadi na baze podataka izvode hakeri koji dolaze sa eksterne mreže, oni deluju na zaštitni zid poslovnog okruženja s ciljem da prođu kroz njega i dođu do bitnih informacija ili da naruše operativnost sistema. Zaštita unutar poslovnog okruženja postaje jedna od najvažnijih stvari prilikom zaštite baza podataka. Taj vid zaštite je izuzetno važan zbog korisnika koji imaju pristup mrežnim resursima iznutra.

Kada dođe do takvog vida pretnje, tada pretnje ne dolaze samo spolja već i iznutra od strane zaposljenih koji imaju pristup bitnim podacima pri čemu je teško definisati preventivno polje zaštite. Tada zaštitni zid nije od neke koristi. Prilikom zaštite podataka iznutra, mora se znati ko može imati pristup kom delu baza podataka.

U daljem tekstu biće prikazane najkritičnije pretnje, kao i detaljna objašnjenja i predlog mogućih rešenja. Spisak je preuzet sa sajta GTD-a (Global terrorism database). [CODEP]

1. Zloupotreba preterane privilegije (excessive privilege abuse) se javlja u slučaju kada se korisnicima ili aplikacijama odobre privilegije za pristup bazi podataka, pa se takve privilegije zloupotrebe. Kao primer možemo uzeti naš univerzitet. Administrator na univerzitetu ima sposobnost da izmeni informacije o raznim informacijama kako o zaposlenima tako i o studentima. Ukoliko bi došlo do zloupotrebe privilegija, na primer u bazi bi mogla da se izmeni ocena studenta. Kako bi se sprečila zloupotreba zbog preteranih privilegija, može se uvesti kontrola pristupa na nivou upita. Ona se odnosi na neki mehanizam koji ograničava privilegije za bazu podataka na minimalni broj SQL operacija i podataka.
2. Zloupotreba legitimne privilegije je zloupotreba koja se javlja od strane legitimnog korisnika. Na primer, na fakultetu postoje podaci o ispitnim pitanjima koja se nalaze na sistemu. Ispit se formira tako što se pitanja sa testova, sa svih petnaest predavanja sakupe i random listaju. Samo unošenje testova i realizacija istih se razvija preko Interneta. Ukoliko zlonamerni student ima dovoljno iskustva može zaobići ograničenja i ući u bazu podataka. Snimajući podatke, student može koristiti to na ispitu. Sprečavanje zloupotrebe kada se radi

o legitimnom pristupu je kontrola pristupa bazi podataka koja se primenjuje ne samo na specifične upite već i na kontekst koji se odnosi na pristup bazi podataka.

UVEĆAVANJE PRIVILEGIJA I TAČKE RANJIVOSTI (10 MIN.)

Uvećavanje privilegija nastaje kada se iskoristi ranjivost BP i povećaju se privilegije korisnika. Tačke ranjivosti mogu biti operativni sistemi i dodatni servisi instalirani na serveru.

3. Do uvećavanja privilegije dolazi tako što sami napadači iskoriste ranjivost softverske platforme baze podataka i tako povećaju privilegije pristupa uobičajnog korisnika. Običan korisnik tada dobija privilegije administratora i samim tim mu se sva vrata otvaraju. Ranjive tačke se mogu pronaći u skladištenim procedurama, ugrađenim funkcijama, implementacijama protokola, SQL upita. Kako bi se sprečio ovakav način upada uvode se kombinacije tradicionalnih sistema za detekciju upada i kontrola pristupa na nivou upada. Sami ISP ispituje saobraćaj i javlja o aktivnostima koje se dešavaju. Za razliku od zaštitnog zida on nema mogućnost da zaustavi saobraćaj već samo da ga detektuje i prijavi. Tada se preuzimaju odgovarajući koraci.

4. Sami operativni sistemi sadrže ranjive tačke, i dodatni servisi koji se instaliraju u server baze podataka te mogu dovesti do pojave neovlašćenog pristupa, oštećenja podataka ili do odbijanja izvršenja usluge. Kako bi se sprečio napad na platformu, potrebno je stalno ažuriranje softvera i sistema za sprečavanje upada. To znači da je potrebno redovno krpiti sistem kako bi sistem bio što sigurniji.

SQL INJEKCIJA (15 MIN.)

Sastoji se od upisivanja odgovarajućeg teksta u polja obrasca koja aplikacija koristi u svojim SQL upitima, tako da upiti iz baze vrate podatke koje inače ne bi.

5. SQL injekcija je napadačka tehnika koja se koristi kako bi se ugrozila sigurnost web aplikacije koja konstruiše SQL upite iz korisnički unesenih podataka. SQL injekcija je tehnika koja se sastoji od upisivanja odgovarajućeg teksta u polja obrasca koja aplikacija koristi u svojim SQL upitima, tako da upiti iz baze vrate podatke koje inače ne bi. Ova metoda, kao i većina hakerskih postupaka, počiva na metodi pokušaja i grešaka i temelji se na pogađanju strukture SQL upita i imena atributa i tablica. Teško je reći da postoji siguran način da se spreče SQL injekcije. Kako bi se smanjile verovatnoće njihovih događaja mogu se koristiti neke od poznatih tehnika:

- *Proveravati ulaz*
- *Zabraniti navodnike i escape znakove u ulazu*
- *Koristiti ograničene parametre*
- *Zaključavati sistem*
- *Koristiti uskladištene procedure za pristup bazi podataka*
- *Izolovati web server*

- *Podesiti prijavljivanje grešaka*
- *Enkripcija podataka*

6. Slaba autentikacija napadačima omogućava da pribave identitet legitimnih korisnika baze podataka kada putem krađe ili na neki drugi način dobiju akreditive neophodne za prijavljivanje.

7. Skladištenje arhiviranih baza podataka često je nezaštićeno od napada. Kao rezultat toga, nekoliko ozbiljnih previda u pogledu bezbednosti su doveli do krađa traka sa arhiviranim podacima iz baze podataka. Kako bi se sprečilo otkrivanje podataka iz arhive svi podaci iz baze bi trebali biti šifrovani.

Slika 7.2.1 SQL injekcija [Izvor: NM IT350 - 2020/2021.]

ODBIJANJE PRUŽANJA USLUGA I RANJIVOST KOMUNIKACIONIH PROTOKOLA (10 MIN.)

Odbijanje pružanja usluga znači da se budućim korisnicima ne dozvoljava pristup aplikacijama ili podacima na mreži. Mnogi komunikacioni protokoli su ranjivi.

8. Odbijanje pružanja usluge (eng.Denial of Service - DOS) je kategorija generalnog napada kod kojeg se budućim korisnicima ne dozvoljava pristup aplikacijama na mreži ili podacima. Odbijanje pružanja usluge može se izvesti na osnovu mnogih tehnika. DOS se može realizovati tako što bi se iskoristila ranjivost platforme u bazi podataka sa ciljem da se prekine rad servera. Uobičajene tehnike DOS-a obuhvataju oštećene podataka, preplavljivanja mreže, i preopterećenje resursa servera. Preopterećenje servera naročito je uobičajno u okruženjima baze podataka. Zaštita od aktiviranja DOS zahteva preuzimanje zaštite na višestrukim nivoima. Neophodne su zaštite na svim nivoima mreže aplikacije i baze podataka.

9. Postoji sve veći broj ranjivih tačaka u oblasti bezbednosti u komunikacionim protokolima u bazi podataka. Kao što se može i videti u GTD-u [CODEP] :" *Četiri od sedam bezbednosnih intervencija kod dva najnovija IBM DB2 FixPacks odnose se na ranjivosti 1 protokola. Na sličan način, 11 od 23 ranjivih tačaka kod baza podataka koje su otklonjene u najnovijoj četvoromesečnoj zakrpi (patch) Oracle-a odnose se na protokole. Malverzantske aktivnosti koje se ciljno usmeravaju na ovakve ranjive tačke mogu da variraju po obimu od neovlašćenog pristupa podacima, preko nanošenja štete podacima, pa sve do odbijanja*

pružanja usluge. Crv SQL Slammer2, na primer, je iskoristio slabu tačku kod Microsoft SQL Server protokola da bi iznudio odbijanje pružanja usluge. I da situacija bude još gora, nikakvi zapisi o ovakvim malverzantskim vektorima neće postojati u lokalnim dnevničkim zapisima budući da operacije protokola ne obuhvataju mehanizme upisivanje u dnevnik aktivnosti."
Napad na komunikacioni protokol u bazi podataka može se predupredi uz pomoć tehnologije koja se naziva validacija protokola. Ona u suštini raščlanjuje saobraćaj u bazi podataka i upoređuje ga sa očekivanim. U slučaju da aktivni saobraćaj ne odgovara očekivanjima, mogu se dati upozorenja i mogu se blokirati aktivnosti.

7.3 Zadaci za samostalni rad

PRIMERI ZADATAKA ZA SAMOSTALNI RAD (VREME IZRADE 75 MIN.)

Uradite sledeće zadatke:

ZADATAK 1: Pretpostavite da u bazi podataka Univerziteta Metropolitan imate tabele koje je kreirao korisnik (rola) administrator:

1. STUDENT koja sadrži matične podatake o studentima
2. PROFESOR koja sadrži matične podatke o profesorima
3. PREDMET koja sadrži podatke o predmetima
4. ANGAZOVANJE koja sadrži podatke o angažovanju profesora na predmetima
5. GODINA_UPISA koja sadrži podatke o upisanim godinama studenata

Pretpostavka je takođe da postoje tri korisnika (role) koje imaju pravo pristupa ovim podacima: referent, student i profesor

Korišćenjem naredbe GRANT omogućite korisniku:

1. PROFESOR da samo čita podatke iz tabela PROFESOR i PREDMET
2. REFERENT da upisuje, čita i ažurira podatke iz svih tabela
3. STUDENT da vidi i ažurira podatke iz tabele STUDENT

Za rešavanje zadatka je potrebno 30 minuta.
ZADATAK 2: Neka se relacije K (KNJIGA), I (IZDAVACI) i KI (KNJIGA/IZDAVACI) nalaze na lokacijama l3, l1, l2 respektivno, i neka se na lokaciji l2 postavi SQL upit kojim treba naći naslove izdanja svih izdavača kao i brojeve izdanja i nazive tih izdavača.

Neka su veličine relacija date sa:

1. K (KJSIF, NASLOV, OBLAST) – 500 n-torki
2. I (I_SIF, NAZIV, STATUS, DRZAVA) – 200 n-torki
3. KI (K_SIF, I_SIF, IZDANJE, GODINA, TIRAZ) – 5 000 n-torki

Prateći korake osnovne strategije distribuiranog izvršavanja upita, treba sračunati vreme izvršenja ovog upita.

Za rešavanje zadatka je potrebno 45 minuta.

Poglavlje 8

Domaći zadatak

DOMAĆI ZADATAK 12 - VREME IZRADE 150 MIN.

Domaći zadatak uradite koristeći jednu od sledećih baza podataka (1-2):

ZADATAK 1: Za bazu podataka koju ste kreirali u lekciji 8. pretpostavite da su neke od tabela distribuirane i to na opisani način.
Propusnost je 50 000 bita u sekundi, veličina torke je 100b, dok je kašnjenje 0,1s.

BAZA PODATAKA 1:
LOKACIJA I1: tabela Pacijent (Ime , prezime, JMBG, adresa, telefon, imeDoktora) LOKACIJA I2: tabela Bolest (Naziv, Opis, Slika)
LOKACIJA I3: tabela BolujeOd (Pacijent.Ime, Bolest.Naziv, datum_dijgnoze))
Neka su veličine tabela date sa:
Pacijent (Ime , prezime, JMBG, adresa, telefon, imeDoktora) – 1 000 n-torki
Bolest (Naziv, Opis, Slika) - 500 n-tork
BolujeOd (Pacijent.Ime, Bolest.Naziv, datum_dijgnoze)) - 10000 n-torki
Prateći korake osnovne strategije distribuiranog izvršavanja upita, treba sračunati vreme izvršenja upita kojim će se naći svi pacijenti kojima broj telefona počinje sa 065, opise njihovih bolesti i datum dijagnoze, a pod pretpostavkom da se da se upit izvršava na lokaciji I3. Prenos relacija na druge lokacije vršiti po sopstvenom izboru koji treba opisati u rešenju.
Pri rešavanju zadatka koristiti primer iz sekcije "Efikasnost nabrojanih varijanti prenosa".

BAZA PODATAKA 2:

LOKACIJA I1: tabela Korisnik (Ime , prezime, JMBG, adresa, telefon)

LOKACIJA I2: tabela Knjiga (Naziv, Opis, Slika, imeAutora)

LOKACIJA I3: tabela Iznajmljivanje (Korisnik.ime, Knjiga.Naziv, Zaposleni.Ime, datum_izd)

Neka su veličine tabela date sa:

Korisnik (Ime , prezime, JMBG, adresa, telefon) - 500 n-torki

Knjiga (Naziv, Opis, Slika, imeAutora) - 2000 n-torki

Iznajmljivanje (Korisnik.ime, Knjiga.Naziv, datum_izd) - 9000 n-torki

Prateći korake osnovne strategije distribuiranog izvršavanja upita, treba sračunati vreme izvršenja upita kojim će se naći sve knjige koje u svom naslovu sadrže reči "Database

Systems", imena njihovih korisnika i datum kada su te knjige izdate, a pod pretpostavkom da se da se upit izvršava na lokaciji I3. Prenos relacija na druge lokacije vršiti po sopstvenom izboru koji treba opisati u rešenju.

Pri rešavanju zadatka koristiti primer iz sekcije "Efikasnost nabrojanih varijanti prenosa".

NASTAVAK FORMULACIJE DOMAĆEG ZADATKA 12

Domaći zadatak uradite koristeći jednu od navedenih baza podataka (3-4):

BAZA PODATAKA 3:

LOKACIJA I1: tabela Profesor (Ime, prezime, JMBG, adresa, telefon, slika, imeSmera)

LOKACIJA I2: tabela Predmet (Naziv, fond_casova_predavanja, fond_casova_vežbi)

LOKACIJA I3: tabela Predaje (Profesor.Ime, Predmet.Naziv, skolska_god) Neka su veličine tabela date sa:

Profesor (Ime, prezime, JMBG, adresa, telefon, slika, imeSmera) - 100 n-torki

Predmet (Naziv, fond_casova_predavanja, fond_casova_vežbi) - 150 n-torki Predaje (Profesor.Ime, Predmet.Naziv, skolska_god) - 3000 n-torki

Prateći korake osnovne strategije distribuiranog izvršavanja upita, treba sračunati vreme izvršenja upita kojim će se naći sve predmete čiji je fond časova predavanja veći od 2, imena profesora koji drže te predmete i školsku godinu kada profesor drži taj predmet, a pod pretpostavkom da se da se upit izvršava na lokaciji I3. Prenos relacija na druge lokacije vršiti po sopstvenom izboru koji treba opisati u rešenju.

Pri rešavanju zadatka koristiti primer iz sekcije "Efikasnost nabrojanih varijanti prenosa".

BAZA PODATAKA 4:

LOKACIJA I1: tabela Gost (Ime , prezime, JMBG, adresa, telefon)

LOKACIJA I2: tabela Soba (Broj, Opis, Slika)

LOKACIJA I3: tabela RezervacijaSobe (Gost.ime, Soba.Broj, datum_od, datum_do)

Neka su veličine tabela date sa:

Gost (Ime , prezime, JMBG, adresa, telefon) - 1000 n-torki

Soba (Broj, Opis, Slika) - 100 n-torki

RezervacijaSobe (Gost.ime, Soba.Broj, datum_od, datum_do) - 5000 n-torki Prateći korake osnovne strategije distribuiranog izvršavanja upita, treba sračunati vreme izvršenja upita kojim će se naći svi gosti čiji telefon počinje na 064, opise njihovih soba i datum od koga

su rezervisali tu sobu, a pod pretpostavkom da se da se upit izvršava na lokaciji I3. Prenos relacija na druge lokacije vršiti po sopstvenom izboru koji treba opisati u rešenju.

Pri rešavanju zadatka koristiti primer iz sekcije "Efikasnost nabrojanih varijanti prenosa".

TREĆI DEO FORMULACIJE DZ12

Domaći zadatak uradite koristeći jednu od datih baza podataka (5):

BAZA PODATAKA 5:

LOKACIJA 1: tabela Materijal (Naziv, Opis, Slika)
LOKACIJA 2: tabela Proizvod (Naziv, Opis, Slika)
LOKACIJA 3: tabela Sastavnica (Proizvod.Naziv, Matrijal.Naziv, količina)
Neka su veličine tabela date sa:
Materijal (Naziv, Opis, Slika) - 2000 n-torki
Proizvod (Naziv, Opis, Slika) - 500 n-torki
Sastavnica (Proizvod.Naziv, Matrijal.Naziv, količina) - 10000 n-torki
Prateći korake osnovne strategije distribuiranog izvršavanja upita, treba sračunati vreme izvršenja upita kojim će se naći svi materijali koji u svom nazivu sadrže reč "Koža", nazive njegovih proizvođača i količinu iz sastavnice, a pod pretpostavkom da se da se upit izvršava na lokaciji I3. Prenos relacija na druge lokacije vršiti po sopstvenom izboru koji treba opisati u rešenju.
Pri rešavanju zadatka koristiti primer iz sekcije "Efikasnost nabrojanih varijanti prenosa".

ZADATAK 2: Navesti što je potrebno uraditi da bi se JOIN dve tabele efikasno obavio na računaru sa više procesora?

DRUGI DEO - DOMAĆI ZADATAK 12

Drugi deo domaćeg zadatka - upiti

1. Napisati upit kojim se prikazuju podaci o svim studentima koji su položili predmet CS101 i imaju prosečnu ocenu veću od 8,25.

2. Napisati upit kojim se prikazuju podaci o svim profesorima koji predaju najmanje dva predmeta.
3. Napisati upit kojim se prikazuju podaci o 3 najneuspešnija predmeta, tj. predmeta koja je položio najmanji broj studenata.
4. Napisati upit kojim se prikazuju podaci o smeru na kome ima najviše studenata koji su položili Engleski 1.

5. Napisati upit kojim se za svaki predmet koji najmanje 4 časa predavanja i vežbi zajedno, prikazuje broj studenata koji su taj predmet položili u koloni koja je imenovana sa “položili”,

zatim ukupan broj studenata koji je taj predmet nisu položili u koloni koja se zove “nisu položili” i u poslednjoj koloni prikazati prosečnu ocenu na tom predmetu i kolonu imenovati sa “prosečna ocena”.
6. Prikazati podatke o predmetu i studentu čija je ocena upisana prva u tabelu overa. Poslednja ocena ima najveći datum u tabeli overa.

7. Napisati upit kojim se prikazuje prosečna ocena za sve studente koji su rođeni u 2, 5, 6, 8, 9, ili 11. mesecu.

8. Napisati upit kojim se za sve studente čiji broj telefona ne počinje sa 062 ili 063 i koji su tradicionalni studenti, prikazuje broj predmeta koji su položili.

9. Napisati upit kojim se prikazuju podaci o profesorima koji predaju predmete koji imaju najmanje 7 espb-a.

10. Napisati upit kojim se prikazuju svi podacima o studentima koji imaju barem jednu ocenu 10.

11. Napisati upit kojim se prikazuju svi studenti čija je prosečna ocena veća od prosečne ocene svih studenata koji studiraju na smerovima IT i SI.

12. Napisati upit kojim se kreira pogled koji prikazuje koje predmete predaju nastavnici. Spisak sortirati po prezimenima nastavnika u rastućem redosledu. Poziv pogleda i brisanje.

NAČIN PREDAJE ZADATKA

Zadatak predajte prema sledećin instrukcijama

Napomena: Zadatke sačuvati u dokumentu sa nazivom IT250-DZ12-Br_INdeksa-Ime_Prezime.txt ili .docx formatu, gde su ime, prezime i broj indeksa vaši podaci.

Za drugi deo DZ12: **Svaki student radi 3 zadatka**.

Redni brojevi zadatka se računaju po formulama:

1. zadatak = (Broj_indeksa mod 12) + 1

2. zadatak = ((Broj_indeksa + Redni broj zadatka 1) mod 12) +1

3. zadatak = ((Broj_indeksa + Redni broj zadatka 2) mod 12) +1
Rešeni zadatak šaljete kao SQL file (arhiviran) na mail adresu asistenta.

Prilikom slanja domaćih zadatka, neophodno je da ispunite sledeće:

Subject mail-a mora biti IT250-DZbr (u slučaju kada šaljete domaći za ovu nedelju to je IT250-DZ12)
U prilogu mail-a treba da se nalazi arhiviran projekat koji se ocenjuje imenovan na sledeći

način: IT250-DZbr-BrojIndeksa. Telo mail-a treba da ima pozdravnu poruku.

Arhivu sa zadatkom poslati na adresu predmetnog asistenta:
milica.vlajkovic@metropolitan.ac.rs (studenti u Beogradu i online studenti) ili
tamara.vukadinovic@metropolitan.ac.rs (studenti u Nišu).
Svi poslati mail-ovi koji ne ispunjavaju navedene uslove NEĆE biti pregledavani. Za sva pitanja ili nedoumice u vezi zadatka, možete se obratiti asistentu

ZAKLJUČAK

Šta smo naučili u ovoj lekciji?

U prvom delu predavanja se govorilo o zaštiti baza podataka što predstavlja veliki problem u korišćenju baza podataka. Da bi podaci smešteni u bazama podataka bili što bolje sačuvani, zaštita se vrši na više nivoa.

U drugom delu predavanja se govorilo o paralelnoj i distribuiranoj obradi baza podataka.

Pored moguće arhitekture koja se može koristiti u paralelnoj obradi podataka, u predavanju je kroz više primera objašnjen način izvršenja nekih karakterističnih operacija nad jednom i dve relacije u slučaju korišćenja više procesora i efikasnosti performansi tih operacija.
U predavanju je takođe bilo reči o potrebama za distribucijom podataka a zatim i o problemima koje ovakav način pamćenja podataka nosi. Najveći problem u ovakvom okruženju je omogućiti da se podaci dislocirani na različitim lokacija što efikasnije obrade i smanji prenos podataka kroz mrežu. U predavanju se opisane metode kojima se ovaj problem može rešiti.

LITERATURA

Za pisanje ove lekcije korišćena je sledeća literatura:

1. Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom, DATABASE SYSTEMS –The Department of Computer Science, Stanford University, 2009 by Pearson Education Inc.
2. C. J. Date, An introduction to Database Systems, Addison-Wesley Publishing Company, 1990