IT250 – Skripta by Oto 5460

1. Kako se podaci trajno čuvaju?

Da bi se tim podacima pristupilo i nakon završetka programa sa svojim izvršenjem.Podaci koji se unose danas, mogu se upotrebiti i u budućnosti.Ti trajni podaci mogu biti zapamćeni u datotekama ili bazama podataka.

2. Slogovi u datoteci mogu imati različite forme. Navesti ih.

- **Slogovi fiksne dužine**:Svaki slog se sastoji od više polja od kojih svaki ima fiksnu dužinu u bajtovima.

- **Slogovi promenljive dužine**: Svaki slog se sastoji od izvesnog broja polja od koji svako može imati maksimalnu dužinu, ali i minimalnu dužinu( 0 bajtova).Polja su razdvojena delimiterima.

- **Zaglavlja i detalji**: Postoje dve vrste slogova : svaki slog transakcije se sastoji od sloga zaglavlja koji je praćen izvesnim brojem slogova detalja.Broj slogova detalja se pamti u zaglavlju tako da je moguće reći gde počinje sledeći slog zaglavlja.

- **Tagovani podaci**: Podaci mogu imati složenu strukturu, kao što su na primer objektno orijentisani sistemi- i ponekad je neophodno objekte različitih klasa pamtiti u istom file-u.Svaki objekat i atribut može imati tag koji je opisan i na neki način saopštava programeru koji čita fajlove o kakvom se podatku radi.Ovaj pristup se koristi za podatke u fajlovima koji koriste HTML i XML.

3. Postoji više načina na koje file-ovi mogu biti organizovani, i to:

- **Serijska organizacija◊** Svaki sledeći slog u fajlu se upisuje na kraj prethodnog sloga.Kada se briše, file mora biti kopiran od početka do izbrisanog sloga, koji se preskače, a tada se kopira i ostatak fajla.

- **Sekvencijalna organizacija fajla**◊ Svaki slog se upisuje u fajl po nekom predefinisanom redosledu, koji se najčešće bazira na vrednosti jednog polja u slogu, kao npr. broj frakture.Brisanje isto kao kod serijske organizacije.

- **Random organizacija◊** Slogovi se dodaju u fajl korišćenjem preciznog algoritma koji dozvoljava da se slogovi upišu i čitaju direktno bez obaveze da se čita ostatak fajla.

4. Navesti načine za pristup podacima u file-ovima:

- **Serijski pristup◊** Da bi se stiglo do traženog sloga, iz fajla je potrebno čitati slog po slog dok se ne locira traženi slog.

- **Direktan pristup◊** Korišćenjem algoritama za konverziju vrednosti polja ključa u adresu slogau faju, vrši se lociranje traženog sloga.

5. Šta je baza podataka?

Trajni podaci mogu biti zapamćeni u datotekama ili bazama podataka koje su najčešće relacione ili objektno orijentisane. Korišćenje sistema za upravljanje bazama podataka (DBMS) ima niz prednosti nad datotekama.

6. Šta čini sistem baze podataka?

??????????

7. Tri-šeme arhitektura:

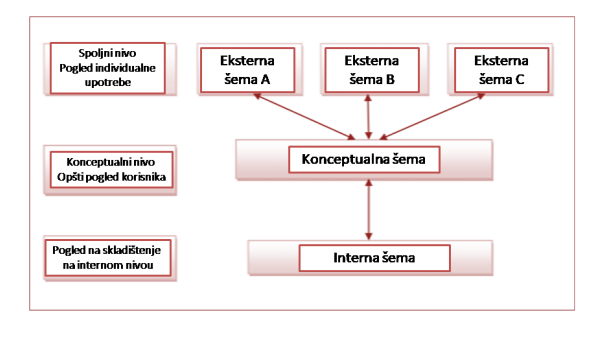
Danas je najčešće u upotrebi Tri-šeme arhitektura koja je prikazana na slici 3. 1 i koristi se pristup podacima u bazama podataka. Sastoji se od:

1. Eksterne šeme koja predstavlja način na koji se podaci iz baza podataka koriste u aplikativnim programima.

2. Konceptualne šeme koja predstavlja logički model podataka i nezavisna je i od eksterne šeme i od detalja o tome kako se podaci pamte.

3. Interne šeme, kojom se definišu file-ovi za pamćenje podataka.

Cilj ovog pristupa je izolovati aplikativne programe od detalja o tome kako je zapamćena bilo koja stavka podataka. To je osnovno za način na koji radi Sistem za upravljanje bazom podataka (DBMS).



8. Šta nudi DBMS?

DBMS obezbeđuje više od samog načina za pamćenje podataka koje mogu da koriste više aplikacija. On obezbeđuje alate koji se mogu koristiti za upravljanje podacima.

1. Data definition language (DDL): se koristi za specifikaciju podataka koji se drže u sistemu za upravljanje bazom podataka i za strukture koje se koriste za njihovo čuvanje

2. Data manipulation language (DML): se koristi da bi se specificirala ažuriranja i pretraživanja podataka u DBMS.

3. Ograničenja integriteta: se specificiraju da bi se održao integritet podataka

4. Upravljanje transakcijama: Ažuriranja nad bazama podataka se specificiraju kao transakcije u okviru kojih se moraju izvršiti sve komponente ažuriranja jer se u suprotnom mora izvršiti roll back transakcije, što znači da se ona neće trajno zapamtiti u bazi podataka (izvršiti commit transakcije).

5. Konkurentnost: Više korisnika može da simultano koristi bazu podataka i ažurira njen sadržaj.

6. Zaštita: pristup podacima u bazi podataka i dozvole koje su dodeljene različitim korisnicima za različite nivoe pristupa (na primer select, update, delete, može biti kontrolisan)

9. Koje su prednosti, a koji su nedostaci korišćenja BP?

???????????

10. Navesti tipove BP:

- RDBMS

- ODBMS

- ORDBMS

11. Koje su osnovne razlike između tipova BP?

**- RDBMS:** Od podataka koji su smešteni u tabelama i relacijama koje postoje između redova tabela.

**- ODBMS:** Objektna orjentacija je pristup u kome se neki sistem organizuje kao kolekcija međusobno povezanih objekata koji ostvaruju postavljene ciljeve. Koriste se kada je potrebno podržati princip nasleđivanja i prednosti su korišćenje kompleksnih tipova podataka, primena principa nasleđivanja i enkapsulacije itd

**- ORDBMS:** Tradicionalni relacioni model se proširuje osnovnim objektnim konceptima kao što su: apstraktni tipovi podataka, enkapsulacija, polimorfizam, nasleđivanje i sl. Objektno-relacioni model i objektno-relacione baze podataka reprezentuju takozvani evolucijski pristup koji se ogleda u integraciji objektne paradigme u relacioni model podataka i njegovim proširenjem objektno-orjentisanim karakteristikama. Objektno-relacione baze se mogu nazvati i hibridne baze.

12. Šta čini koncepcijske osnove relacionog modela?

U teorijske, koncepcijske osnove relacionog modela spadaju:

• Funkcije,

• Dekartov proizvod skupova,

• Odnosi između skupova

13. Kako se definiše relacija u relacionom modelu baze podataka?

Relacija (veza) predstavlja asocijacije koje se mogu uspostaviti između tipova entiteta. Svakoj relaciji se zadaje naziv koji opisuje njenu funkciju.

14. Objasniti šta se podrazumeva pod “relacija kao imenovana tabela”?

????????????

Kada postoji relacija sa kardinalnoscu vise prema vise, po pravilu se tu pravi nova tabela. Nova tabela ce imati ime I sadrzace primarne kljuceve od dve tabele izmedju kojih se nalazi relacija. Ova nova tabela moze sadrzati I dodatne atribute.

????????

15. Objasnite šta je:

* 1. Primarni ključ
  2. Strani ključ
  3. Kandidat ključ
  4. Surogat ključ

**Primarni ključ** (eng.Primary Key): Ovo je glavni identifikator entiteta u modelu podataka. Primarni ključ obezbeđuje jedinstvenost i jednoznačno razlikuje svaki entitet u tabeli

**Strani ključ**: Ako neki skup atributa u posmatranoj relaciji nije ključ, ali je ključ u nekoj drugoj relaciji u modelu, onda se on naziva **strani ključ**.

**Kandidatski ključ** (eng.Candidate Key): To su svi mogući ključevi koji jednoznačno identifikuju entitet, uključujući i primarni i alternativne ključeve.

**Surogat ključ** je veštačka kolona koja se dodaje tabeli da bi služila kao primarni ključ

16. Objasnite operatore relacione algebre pogodne za ažuriranje.

?????????????????????????????????????????? Koga briga za operatore relacione algebre ?

17. Objasnite operatore relacione algebre za izveštavanje.

SELEKCIJA (select) je operacija nad jednom relacijom koja izdvajanje skupa ntorki jedne relacije u novu. Izdvajaju se samo one n-torke koje zadovoljavaju uslov uz operator selekcije.

PROJEKCIJA (project) je operacija nad jednom relacijom koja omogućava izdvajanje skupa atributa jedne relacije u novu relaciju, a pri tom eliminiše duple n-torke.

18. Šta su modeli i čemu služe?

Model podataka definiše strukturu i značenje podataka. Svi modeli predstavljaju ograničenu prezentaciju realnosti. Model podataka prikayuje kako neka struktura treba da iygleda I koje informacije treba da sadrži, nezavisna je od softvera I hardvera. Model predstavlja apstrakciju neke strukture.

19. Osnovni koncepti E/R modela su:

Evo nekoliko osnovnih koncepata vezanih za E/R dijagrame:

**Entiteti (Entities):** Entiteti predstavljaju osnovne objekte ili pojave u sistemu koji želimo da modeliramo. Na dijagramu se obično predstavljaju pravougaonicima.

**Atributi (Attributes):** Atributi su karakteristike ili osobine entiteta koje opisuju informacije o njima. Na dijagramu se često predstavljaju ovalima povezanim sa odgovarajućim entitetima.

**Relacije (Relationships):** Relacije označavaju veze između entiteta. One se prikazuju linijama koje povezuju odgovarajuće entitete. Relacije mogu imati različite stepene (jedan prema jedan, jedan prema više, više prema više).

**Identifikatori (Identifiers):** Identifikatori su posebni atributi koji jedinstveno identifikuju svaku instancu entiteta. Često se označavaju kao primarni ključevi.

**Tipovi entiteta (Entity Types):** Svaki entitet ima svoj tip, koji predstavlja klasu sličnih entiteta. Tip entiteta se obično naziva poimenično i predstavlja skup sličnih objekata.

20. Koji su koraci u kreiranju konceptualnog modela?

Glavni koraci u kreiranje konceptualnog modela koji predstavljamo E/R dijagramom su:

1. Odrediti tipove entiteta

2. Odrediti relacije između tipova entiteta

3. Profiniti definiciju tipova entiteta

21. Šta je entitet?

Entitet predstavlja osnovni objekat ili pojavu koja ima svoje karakteristike i može se identifikovati. U kontekstu modeliranja baye podataka entitet bi predstavljao jednu tabelu.

22. Šta su atributi i kako se mogu klasifikovati?

Atributi predstavljaju informacije o entitetu. Na primer, entitet "Osoba" može imati atribute kao što su ime, prezime, datum rođenja itd. Atributi imaju I svoj tip. (broj, tekst I sl)

23. Šta predstavlja naziv relacije u E/R dijagramu?

U relaciji iymedju entiteta, nazivi relacije opisuju njenu funkciju. (npr. radi\_u)

24. Opisati kardinalnost relacije u relacionom modelu.

Kardinalnost se definiše kao broj entiteta jednog tipa entiteta koji se mogu povezati sa jednim entitetom drugog tipa entiteta. (1-1, 1-n, m-n)

25. Koje su karakteristike rekurzivne relacije?

**Jedan entitet učestvuje više puta**: U rekurzivnoj relaciji, isti entitet može učestvovati više puta sa različitim ulogama. U primeru zaposlenog koji rukovodi drugim zaposlenima, entitet "Zaposleni" učestvuje jednom kao rukovodilac i drugi put kao obični zaposleni.

**Različite uloge entiteta:** U rekurzivnoj relaciji, entitet učestvuje sa različitim ulogama. Na primer, u relaciji "rukovodi", entitet "Zaposleni" ima dve uloge - rukovodilac i običan zaposleni.

**Označavanje unarnih relacija:** Rekurzivne relacije se ponekad nazivaju i unarnim relacijama. To znači da entitet učestvuje u relaciji sa samim sobom. Ovaj tip relacije se posebno koristi kada se entitet povezuje sa samim sobom kroz određenu vezu ili ulogu.

26. Šta je domen?

Domen je skup svih tipova podataka i opsega vrednosti koje mogu da imaju atributi. Definicije domena obično specificiraju neke (ili sve) karakteristike atributa: tip podataka, dužina, format, opseg, dozvoljene vrednosti, značenje, jedinstvenost i null vrednosti.

27. Šta su trigeri i koje komponente imaju operacije za izvršenje trigera?

Trigeri su pravila pod kojim se izvršavaju operacije za manipulaciju podacima kao što su: insert, update i delete. Operacije za izvršenje trigera mogu biti ograničene na atribute jednog entiteta ili atribute dva ili više entiteta. Operacije za izvršenje trigera obično imaju sledeće komponente:

1. Poslovno pravilo: koncizna rečenica kojom se opisuje poslovno pravilo koje treba da bude izvršeno operacijom;

2. Događaj: operacija koja se inicira (insert, delete ili update);

3. Ime tipa entiteta kojem se pristupa ili se modifikuje;

4. Uslov: pod kojim se operacija izvršava

5. Akcija: koja se izvršava kada se operacija trigeruje.

Npr. U bazu se moze upisati datum koji je posle 12-12-2023, svaki datum pre toga se ne sme upisati u bazu, ova provera pre upisa u bazu se pravi trigerima.

28. Zbog čega je proces modeliranja podataka uvek iterativan?

Ko bi ga znao zasto je to tako.

29. Kako glase pravila analize teksta?

Ko bi ga znao kako glase.

30. Kako se određuje kardinalnost ternarne relacije?

Određivanje kardinalnosti ternarne relacije uključuje analizu veza između entiteta i utvrđivanje koliko entiteta jednog tipa može biti povezano sa entitetima drugih tipova. Evo koraka za određivanje kardinalnosti ternarne relacije:

**Identifikacija tipova entiteta:** Prvo, identifikujte sve tipove entiteta koji su uključeni u ternarnu relaciju. Na primer, u vašem opisu, tipovi entiteta su EMPLOYEE, SKILL i PROJECT.

**Povežite sve tipove entiteta sa ternarnom relacijom:** Povežite svaki tip entiteta sa ternarnom relacijom koristeći odgovarajuće binarne veze. U vašem primeru, to su SKILL–PROJECT, PROJECT–EMPLOYEE i EMPLOYEE–SKILL.

**Određivanje kardinalnosti za svaki tip entiteta:**

Za svaki tip entiteta koji nije direktno povezan s ternarnom relacijom, postavite pitanje da li može postojati najviše jedan entitet tog tipa za svaku kombinaciju preostalih entiteta. Ako je odgovor "više", koristi se simbol rakljice (npr. crticu) na liniji koja ulazi u taj tip entiteta. Ako je odgovor "1", koristi se crtica preko te linije.

**Ponovite postupak za sve tipove entiteta:** Ovaj korak treba ponoviti za svaki tip entiteta koji nije direktno povezan s ternarnom relacijom.

31. Kada se javljaju višestruke relacije?

Višestruke relacije mogu se javiti u različitim kontekstima, a pojavljuju se kada postoji potreba za modeliranjem veza između više od dva entiteta. Ove relacije se mogu klasifikovati kao ternarne, kvaternarne, itd., u zavisnosti od broja entiteta koji su povezani.

32. Šta je asocijativni tip entiteta?

Asocijativni tip entiteta je poseban entitet u modeliranju baze podataka koji se koristi za reprezentaciju asocijativnih veza između entiteta u više-prema-više odnosima. Ovaj entitet se takođe naziva i kompozitni entitet ili most entitet. Asocijativni tip entiteta je koristan kada želimo dodati dodatne informacije o samoj vezi, osim veza između dva entiteta.

U modeliranju baze podataka, obično se susrećemo sa situacijama u kojima jedan entitet može biti povezan sa više entiteta druge vrste, i obrnuto. Ovaj tip odnosa se naziva "više-prema-više" (M:N). Kada se javi ovakav odnos između dva entiteta, stvara se asocijativni tip entiteta da bi se bolje modelirala veza između njih.

Na primer, neka imamo entitete Student i Predmet. Svaki student može pohađati više predmeta, a isto tako jedan predmet može biti pohađan od strane više studenata. Ovaj odnos je "više-prema-više". Umesto direktnog povezivanja ovih entiteta, možemo koristiti asocijativni entitet "Enrollment" (Upis) koji sadrži dodatne informacije, kao što su ocene, datum upisa, prisustvo itd.

33. Šta je specijalizacija, a šta generalizacija?

Specijalizacija i generalizacija su dva ključna koncepta u modeliranju baze podataka i u okviru koncepta hijerarhija entiteta. Evo njihovih osnovnih definicija:

**Specijalizacija:**

* Specijalizacija predstavlja proces maksimiziranja razlika između entiteta nekog tipa identifikovanjem karakteristika koje ih razlikuju.
* Ovaj proces se često sprovodi odozgo-nadole, definisanjem nadentiteta i podentiteta.
* Nadentitet je opšti tip entiteta, dok su podentiteti specifični tipovi entiteta koji poseduju dodatne karakteristike.
* Ovo se često koristi kada želimo modelirati hijerarhiju entiteta gde podentiteti imaju specifične atribute koji nadograđuju atribute nadentiteta.

**Generalizacija:**

* Generalizacija je obrnuti proces od specijalizacije.
* Predstavlja proces minimiziranja razlika između entiteta identifikovanjem zajedničkih karakteristika.
* Ovaj proces uključuje identifikaciju zajedničkih atributa i stvaranje nadentiteta koji predstavlja te zajedničke karakteristike.
* Zajednički atributi se postavljaju u nadentitet, dok se specifični atributi smeštaju u podentitete.

34. Kakvi su to ID zavisni entiteti?

ID zavisni entiteti su entiteti čiji identifikator (primarni ključ) uključuje identifikator drugog entiteta. Ovi entiteti zavise od postojanja drugog entiteta, poznatog kao roditelj ili nadređeni entitet. Ključna karakteristika ID zavisnih entiteta je da ne mogu postojati nezavisno, već za njihovo postojanje zavisi od postojanja povezanog entiteta.

35. Šta su slabi, a šta jaki entiteti?

**Jaki entiteti:**

* Definicija: Jaki entiteti su entiteti koji imaju dovoljno informacija da se mogu identifikovati samostalno, bez potrebe za bilo kojim drugim entitetom.
* Ključevi: Jaki entiteti obično imaju primarni ključ koji ih jedinstveno identifikuje.
* Nezavisnost: Jaki entiteti postoje nezavisno od drugih entiteta u modelu i ne zavise od drugih entiteta kako bi bili identifikovani.

Primer: Ako imamo entitet "Student" sa atributima poput "StudentID", "Ime" i "Prezime", taj entitet je jak entitet jer se može identifikovati koristeći samo "StudentID" (primarni ključ).

**Slabi entiteti:**

* Definicija: Slabi entiteti su entiteti koji nemaju dovoljno informacija da se samostalno identifikuju. Oni zavise od povezanosti sa drugim entitetom, poznatim kao vlasnik ili nadređeni entitet.
* Ključevi: Slabi entiteti imaju parcijalni ključ koji, zajedno sa ključem vlasnika, čini potpuni identifikator entiteta.
* Zavisnost: Slabi entitet zavisi od postojanja određenog vlasnika kako bi bio jedinstveno identifikovan.

Primer: Ako imamo entitet "Adresa" koji zavisi od entiteta "Klijent" (vlasnik), tada je "Adresa" slabi entitet. Identifikator "Adresa" može uključivati parcijalni ključ "BrojStana" i "PoštanskiBroj", ali postaje potpuni identifikator kada se kombinuje sa identifikatorom vlasnika "KlijentID".

36. Šta se postiže korišćenjem podklasa i nadklasa?

Korišćenjem podklasa (subclasses) i nadklasa (superclasses) u modeliranju podataka postiže se implementacija hijerarhijske strukture i koncepta specijalizacije/generalizacije. Ovaj pristup omogućava organizaciju entiteta u viši nivo apstrakcije i grupisanje sličnih entiteta u kategorije.

37. Transformacija entiteta

Transformacija entiteta u tabele. Fizički pandan relacijama, odnosno tipovima entitetima, je tabela. Svaki entitet odnosno instanca tipa entiteta će se transformisati u jedan red tabele. U većini slučajeva, svakoj relaciji iz logičkog modela će odgovarati jedna tabela u fizičkom modelu. Međutim, postoje slučajevi kada je neke tabele potrebno spojiti ili razdvojiti da bi se postigle bolje performanse baze podataka.

38. Transformacija atributa tipova entiteta

Transformacija atributa u kolone. Fizički pandan atributima su kolone u tabelama. Atributi svakog objekta treba da budu preslikani u kolone odgovarajuće tabele.

39. Ograničenja podataka (domen, opseg vrednosti, intrarelaciono i interrelaciono ograničenje)

Transformacija domena u tipove podataka i ograničenja. Za svaki logički domen atributa je potrebno odabrati adekvatan tip podataka iz seta tipova koji nudi SUBP. Na primer, za godinu rođenja se može izabrati tip podataka integer, a za matični broj, koji ima čak 13 cifara, je možda bolje izabrati tip podataka text. Neki tipovi podataka zahtevaju bliže definisanje podataka. Na primer tip podataka text zahteva da se specificira maksimalni broj karaktera, mada neki SUBP dozvoljavaju da se specificira da tekst može biti promenljive dužine.

**Intra relaciono ograničenje** u odnosu na kolone iste tabele – ograničava vrednost kolone u odnosu na druge kolone iste tabele. Na primer: datum provere može biti najmanje 3 meseca posle datuma zaposlenja.

**Inter relaciono ograničenje** u odnosu na kolone druge tabele -ograničava vrednost kolone u odnosu na druge kolone drugih tabela. Ograničenje referencijalnog integriteta je jedan tip inter relacionog ograničenja.

40. Normalizacija baze podataka

Normalizacija baze podataka je proces organizacije podataka u tabelama (relacijama) kako bi se smanjila redundancija podataka, izbegle anomalije prilikom ubacivanja, brisanja i ažuriranja, i očuvala doslednost i integritet podataka. Ovaj proces ima za cilj poboljšanje strukture baze podataka, čime se olakšava održavanje, poboljšava efikasnost upita i smanjuje rizik od pojave grešaka u podacima.

41. Predstavljanje 1:1 veze između jakih entiteta

Predstavljanje 1-1 veze između jakih entiteta u modeliranju baza podataka se obično vrši dodavanjem stranog ključa (foreign key) jednog entiteta u drugi. Ova veza označava da svaki entitet iz jedne tabele (relacije) ima tačno jedan povezani entitet u drugoj tabeli, i obrnuto.

42. Predstavljanje 1:M veze između jakih entiteta

Predstavljanje 1:M (jedan prema mnogi) veze između jakih entiteta u modeliranju baza podataka se obično postiže dodavanjem stranog ključa (foreign key) jednog entiteta u drugi entitet. Ovaj strani ključ se koristi kako bi se uspostavila veza između njih, označavajući da svaki entitet iz jedne tabele može imati više povezanih entiteta u drugoj tabeli.

43. Predstavljanje M:N veze između jakih entiteta

Predstavljanje M:N (mnogi prema mnogi) veze između jakih entiteta u modeliranju baza podataka obično uključuje korišćenje dodatne tabelarnje strukture, poznate kao "srednja" ili "povezujuća" tabela. Ova tabela se koristi za mapiranje i održavanje veze između entiteta u M:N relaciji.

44. Asocijativne veze

Transformacija asocijativne veze u modeliranju baza podataka često uključuje korišćenje pomoćne ili asocijativne tabele kako bi se reprezentovala veza između entiteta koji imaju M:N (mnogi prema mnogi) odnos.

45. Transformacija podklasa i njihovih nadklasa se može se vršiti na tri načina. Navesti ih.

Transformacija podklasa i njihovih nadklasa se može se vršiti na tri načina, korišćenjem:

1. **ER stila transformacije podklasa i njihovih superklasa**: u tom slučaju za svaki tip entiteta E u hijerarhiji se kreira relacija koja kao ključ uključuje atribute sa korena hijerarhije i atribute koji pripadaju entitetu E.

2. **Objektno orijentisanog pristupa transformacije podklasa i njihovih superklasa**: tipovi entiteti se tretiraju kao objekti koji pripadaju jednoj klasi: za svako moguće podstablo koje se nalazi ispod korena, kreira se jedna relacija čija šema uključuje sve atribute svih tipova entiteta u podstablu.

3. **Null vrednosti za transformaciju podklasa i njihovih superklasa**: kreira se jedna relacija sa svim atributima svih tipova entiteta u hijerarhiji. Svaki entitet je predstavljen jednom torkom i ta torka ima NULL vrednosti za bilo koji atribut koji entitet nema.

46. Kada se radi reinženjering baza podataka?

Postoji nekoliko situacija u kojima se može zahtevati reinženjering baze podataka:

1. Problemi sa performansama softvera uzrokovani ne efikasnom bazom podataka.

2. Promene u zahtevima za softverom koji zahtevaju ažuriranje strukture podataka.

3. Značajne razlike u najnovijoj verziji baze podataka u poređenju sa ranijim verzijama, gde ove razlike zahtevaju promene u bazi podataka.

47. Normalizacija

?????????????????????????????????

Jedna od važnijih tema u oblasti upravljanja bazama podataka je proces normalizacije tabela u relacionoj bazi podataka. Ideja normalizacije je relativno jednostavna. Normalizacijom želimo da projektujemo skup relacija-tabela za našu bazu podataka koje:

1. sadrže sve podatke neophodne za primenu date baze podataka

2. imaju što manje ponavljanja

3. dozvoljavaju efikasno ažuriranje podataka u bazi podataka i

4. izbegavaju opasnosti gubitka podataka bez znanja.

Primarni razlog za normalizaciju baza podataka leži u činjenici da je normalizacija moćno oružje protiv moguće korupcije baza podataka koja potiče od onog što nazivamo "anomalije prilikom ubacivanja", "anomalije prilikom brisanja" i "anomalije prilikom ažuriranja". Ovi tipovi grešaka mogu da postoje u nedovoljno normalizovanim bazama podataka.

48. Denormalizacija

Denormalizacija je proces unazadivanja normalizovane baze podataka, odnosno dodavanje redundancije podataka u cilju poboljšanja performansi sistema. Suprotno normalizaciji, denormalizacija može rezultirati gubitkom nekih prednosti u vezi s očuvanjem integriteta podataka, ali se koristi kako bi se ubrzao pristup podacima.

49. Prva normalna forma (1NF)

Tabela se smatra u prvoj normalnoj formi ako ispunjava određene uslove koji se odnose na organizaciju podataka u toj tabeli. Osnovni zahtev za 1NF je da svi atributi tabele moraju imati samo atomske (nedeljive) vrednosti.

Prva normalna forma se često smatra početkom procesa normalizacije baze podataka. Ako tabela nije u 1NF, to može ukazivati na potrebu daljeg organizovanja podataka kako bi se eliminisala redundancija i očuvala struktura podataka.

Da bismo normalizovali tabelu i doveli je u prvu normalnu formu (1NF), potrebno je eliminisati ponavljajuće grupe i organizovati podatke tako da svaki redak predstavlja jedinstven entitet. Evo primera kako to možete uraditi:

Pretpostavimo da imate sledeću tabelu koja ne ispunjava uslove 1NF:

| StudentID | Ime | Predmeti |
| --- | --- | --- |
| 1 | Marko | Matematika, Fizika |
| 2 | Ana | Hemija, Matematika |
| 3 | Jovan | Fizika, Hemija, Geografija |

U ovoj tabeli, kolona "Predmeti" sadrži niz vrednosti koji predstavljaju grupu predmeta koje je student pohađao. Da bismo ovo doveli u 1NF, trebamo razdvojiti ove vrednosti i svaku od njih staviti u zaseban red. Evo kako može izgledati nakon normalizacije:

Tabela Studenti:

| StudentID | Ime |
| --- | --- |
| 1 | Marko |
| 2 | Ana |
| 3 | Jovan |

Tabela Predmeti:

| StudentID | Predmet |
| --- | --- |
| 1 | Matematika |
| 1 | Fizika |
| 2 | Hemija |
| 2 | Matematika |
| 3 | Fizika |
| 3 | Hemija |
| 3 | Geografija |

Sada imamo dve tabele. Prva tabela "Studenti" sadrži informacije o studentima, dok druga tabela "Predmeti" sadrži informacije o predmetima koje su studenti pohađali. Svaki red u tabeli "Predmeti" je povezan sa određenim studentom putem identifikatora StudentID. Ovako organizovane tabele ispunjavaju uslove prve normalne forme (1NF).

50. Druga normalna forma (2NF)

Druga normalna forma (2NF) je stepen normalizacije u projektovanju baze podataka koji eliminiše određene vrste redundancije podataka. Tabela se smatra u drugoj normalnoj formi ako je u prvoj normalnoj formi (1NF) i ako su eliminisane zavisnosti delimičnih ključeva.

Relacija je u drugoj normalnoj formi (2NF) ako je svaki atribut koji nije primarni ključ funkcionalno zavisan od celog primarnog ključa

Druga normalna forma je zadovoljena ako se može primeniti jedan od sledećih uslova:

1. Primarni ključ se sastoji od samo jednog atributa

2. U relaciji ne postoje atributi koji nisu primarni ključevi

3. Svaki atribut koji nije primarni ključ je funkcionalno zavisan od potpunog skupa atributa primarnih ključeva.

Pretpostavimo da imamo sledeće podatke u relaciji "ZAPOSLENI":

| Zaposleni\_ID | Ime | Odeljenje | Plata | Kurs | Datum\_Zavrsetka |
| --- | --- | --- | --- | --- | --- |
| 101 | Marko | Računovodstvo | 5000 | Excel | 2022-05-01 |
| 102 | Jovana | Marketing | 6000 | PowerPoint | 2022-06-15 |
| 103 | Stefan | Razvoj | 7000 | Java | 2022-07-10 |

Ovde je Zaposleni\_ID primarni ključ, a atributi Ime, Odeljenje, Plata zavise samo od Zaposleni\_ID, . Atribut Datum\_Zavrsetka zavisi od Zaposleni\_ID i od Kurs. Svaki atribut mora biti zavistan od celog primarnog kljuca, sto nije slucaj za atribut Datum\_Zavrsetka. Pravi se nova tabela.

Sada ćemo dekomponovati ovu relaciju u dve relacije koje zadovoljavaju drugu normalnu formu:

Relacija "ZAPOSLENI":

| Zaposleni\_ID | Ime | Odeljenje | Plata |
| --- | --- | --- | --- |
| 101 | Marko | Računovodstvo | 5000 |
| 102 | Jovana | Marketing | 6000 |
| 103 | Stefan | Razvoj | 7000 |

Relacija "ZAPOSLENI\_KURS":

| Zaposleni\_ID | Kurs | Datum\_Zavrsetka |
| --- | --- | --- |
| 101 | Excel | 2022-05-01 |
| 102 | PowerPoint | 2022-06-15 |
| 103 | Java | 2022-07-10 |

Sada svaka relacija ima jedinstven primarni ključ i zadovoljava zahteve druge normalne forme. Ova dekompozicija omogućava efikasno upravljanje podacima i održavanje konzistentnosti.

51. Treća normalna forma (3NF)

Treća normalna forma (3NF) je jedna od normalnih formi u oblasti normalizacije relacijskih baza podataka. Cilj normalizacije je eliminisati redundanciju podataka i održavati integritet podataka. Da biste postigli 3NF, tabela mora zadovoljiti sledeće uslove:

* Biti u drugoj normalnoj formi (2NF): Ovo znači da svaki neključni atribut treba biti potpuno zavisan od celokupnog primarnog ključa, a ne samo od dela ključa.
* Nemati tranzitivne zavisnosti: Nijedan neključni (koji nije primarni kljuc) atribut ne sme biti funkcionalno zavisan od drugog neključnog atributa.

Da bismo bolje razumeli 3NF, evo primera:

Tabela "Zaposleni":

| Zaposleni\_ID | Ime | Odeljenje | Odeljenje\_Lokacija |
| --- | --- | --- | --- |
| 101 | Marko | IT | Beograd |
| 102 | Jovana | Marketing | Novi Sad |
| 103 | Nenad | Finansije | Beograd |

U ovoj tabeli, "Odeljenje\_Lokacija" zavisi samo od "Odeljenje" (zaposleni pripadaju odeljenju, a ne lokaciji). Odeljenje lokacija zavisi od atributa koji nije primarni kljuc (odeljenje). Tabela nije u 3NF jer postoji tranzitivna zavisnost. Da biste postigli 3NF, tabelu bismo podelili na dve:

Tabela "Zaposleni":

| Zaposleni\_ID | Ime | Odeljenje |
| --- | --- | --- |
| 101 | Marko | IT |
| 102 | Jovana | Marketing |
| 103 | Nenad | Finansije |

Tabela "Odeljenja":

| Odeljenje | Lokacija |
| --- | --- |
| IT | Beograd |
| Marketing | Novi Sad |
| Finansije | Beograd |

Sada je svaka tabela u 3NF, i tranzitivna zavisnost je eliminisana. Ove dve tabele mogu se spojiti korišćenjem zajedničkog atributa "Odeljenje" u upitu, ako je potrebno.

52. Boyce Codd normalna forma (BCNF)

Tabela u bazi podataka je u BCNF kada je svaki atribut u tabeli zavistan od superkljuca. U svakoj funkcionalnoj zavisnosti u tabeli, npr. X -> Y, X mora da bude superkljuc.

**Super kljuc** je skup atributa koji mogu jedinstveno da odrede taj red u tabeli.

Prvo odredjujem zavisnosti izmedju kolona u tabeli, tj. koja kolona zavisi od kog kljuca u tabeli. Ako imam vise zavisnosti i neki X atribut nije deo superkljuca onda ta tabela nije u BCNF. Razdvajam na dve ili vise tabela, sve zavisi.

Jbg nema primer, mrzi me.

53. Četvrta normalna forma (4NF)

Javlja se kada jedna determinanta određuje skup vrednosti a ne samo jednu vrednost. Determinanta zavisnosti od više vrednosti ne može nikada biti primarni ključ

Ce bude

54. Veze između normalnih formi

Svaka normalna forma zahteva da je normalna forma pre nje ispunjena.

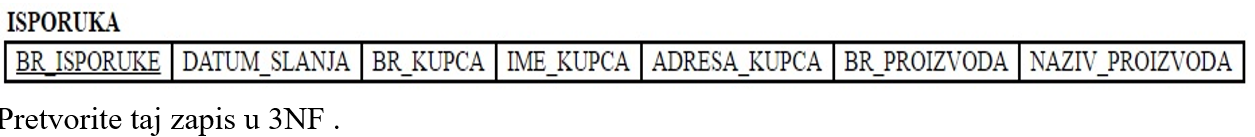
56. Fabrika isporučuje svoje proizvode kupcima. Jedna isporuka šalje se jednom kupcu i može sadržavati više komada raznih proizvoda. Situacija je prikazana sledećim zapisom:

Tabela **Isporuka:**

BR\_ISPORUKE | BR\_KUPCA | BR\_PROIZOVDA | DATUM\_SLANJA

Tabela **Kupac:**

BR\_KUPCA | IME\_KUPCA | ADRESA\_KUPCA

Tabela **Proizvod:**

BR\_PROIZOVDA | NAZIV\_PROIZVODA

57.

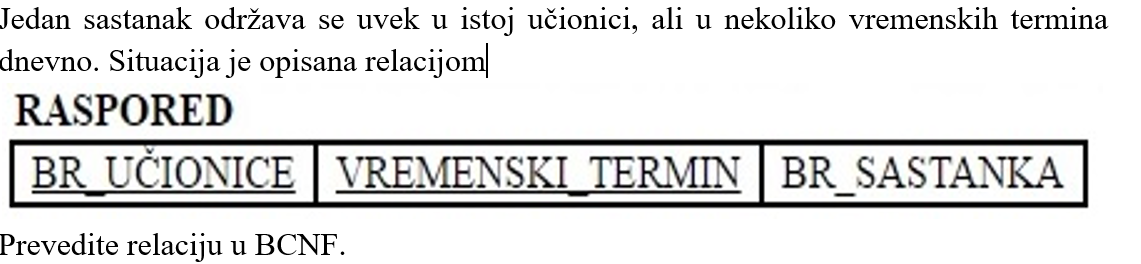


Tabela **Raspored:**

BR\_UCIONICE | BR\_SASTANKA

Tabela **Termin:**

VREMENSKI\_TERMIN | BR\_SASTANKA

58. Relacija na Slici 1. pokazuje da autor za svaku prodatu knjigu od svojih izdavača dobija određeni honorar. Iznos honorara zavisi od autora, knjige i izdanja knjige. Primarni ključ relacije čine ISBN i AutorKnjigeID.   
Normalizujte ovu relaciju tako da dobijete model baze podataka u 3NF i predstavite je E/R dijagramom. Na dijagramu označite identifikatore tipova entiteta.

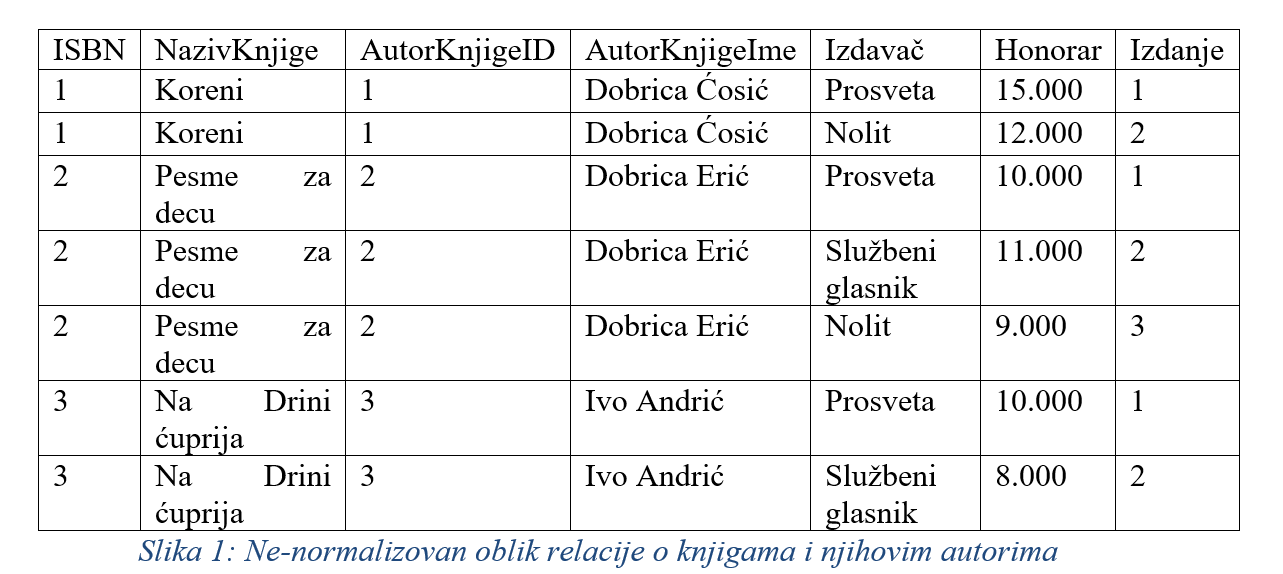


Tabela mora prvo zadovoljiti drugu normalnu formu, a to je da nema parcijalnih zavisnosti. Tabela se prvo prebacuje u drugu normalnu formu.

Tabela **Knjiga: Knjiga -> Izdanje (1 - N), Knjiga -> Autor (N, 1)**

ISBN | NazivKnjige | AutorKnjigeID

Tabela **Izdanje: Izdanje -> Honorar (1, N)**

Izdanje | Izdavac | ISBN

Tabela **Autor:**

AutorKnjigeID | AutorKnjigeIme

Tabela **Honorar:**

Izdanje | Honorar

Ovim resenjem ne postoje ni tranzitivne zavisnosti tako da su tabele u 3NF.

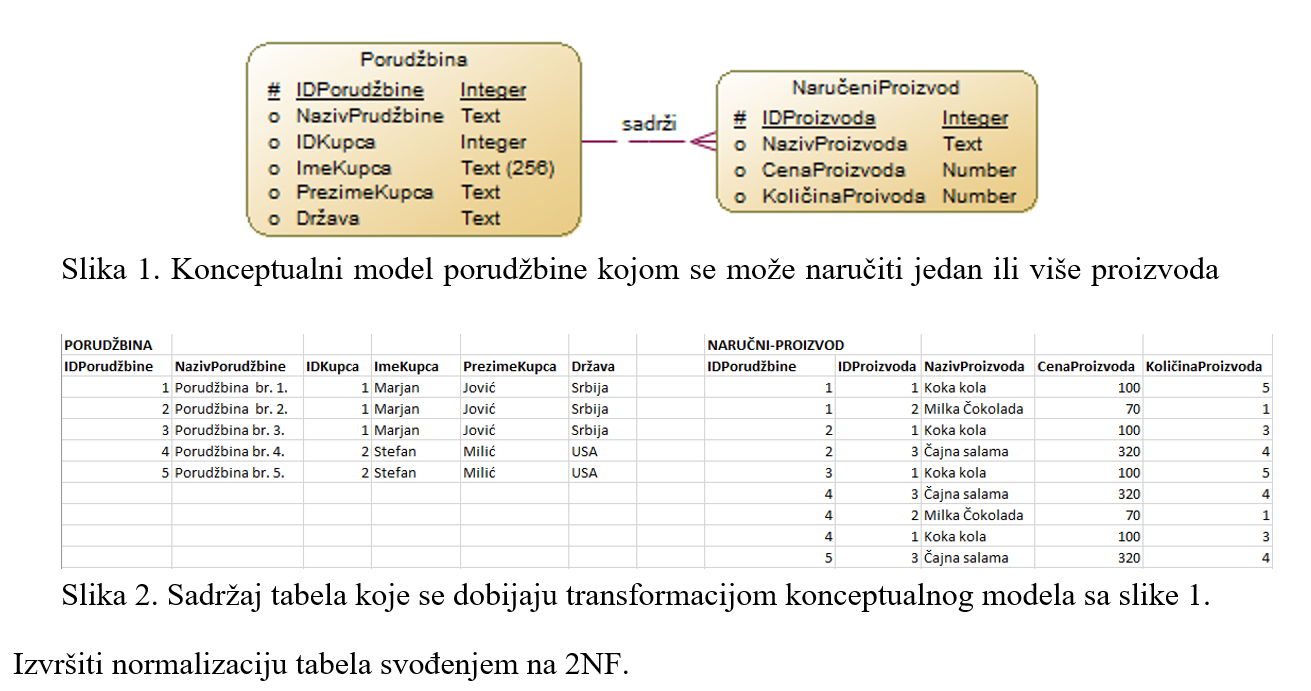
59. Na slici 1. je dat konceptualni model jedne porudžbine kojom se može naručiti jedan ili više proizvoda a na slici 2. je prikazan sadržaj tabela koje se dobijaju transformacijom ovog konceptualnog modela.  


Tabela **Kupac:**

IDPorudzbine | NazivPorudzbine | IDKupca |

Tabela **Kupac:**

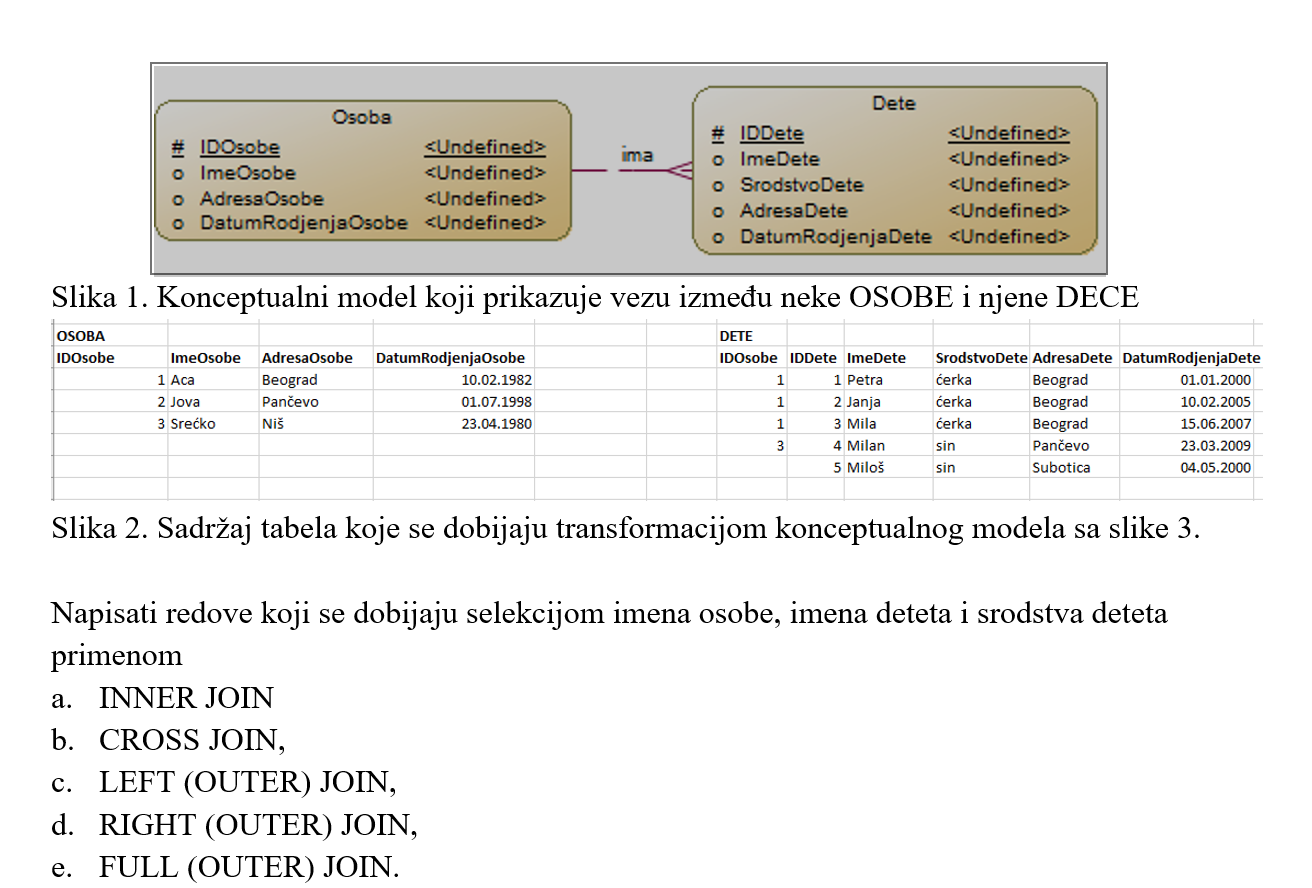
IDKupca | ImeKupca | PrezimeKupca | Drzava

Tabela **Proizvod:**

IDProizvoda | NazivProizvoda | CenaProizvoda

Tabela **NaruceniProizvod:**

IDPorudzbine | IDProizvoda | KolicinaProizvoda

60. Na slici 1. je dat konceptualni model koji prikazuje vezu između neke OSOBE i njene DECE. Osoba može imati više dece, može nemati ni jedno dete a za dete ne mora uvek da se zna koja je OSOBA njen roditelj. Na slici 2. je prikazan sadržaj tabela koje se dobijaju transformacijom ovog konceptualnog modela

A. INNER JOIN

Osoba.Aca -> Dete.Petra

Osoba.Aca -> Dete.Janja

Osoba.Aca -> Dete.Mila

Osoba.Srecko -> Dete.Milan

B. CROSS JOIN

Svaki red iz prve tabele se povezuje sa svkim redom iz druge

C. LEFT OUTER JOIN

Uzimaju se svi redovi iz ttabele Osoba I samo oni redovi iz tabele dete gde ima preklapanja

D. RIGHT OUTER JOIN

Uzimaju se svi redovi iz tabele dete I samo oni redovi iz tabele osoba gde ima preklapanja

E. FULL OUTER JOIN

Uzimaju se svi redovi iz obe tabele

61. Tabelu koja je predstavljena na slici 1, normalizovati i normalizovan oblik tabele predstaviti u obliku E/R dijagrama



Tabela **Employee:**

Id | name | zip | state | city | nationality | department

Tabela **Department:**

Department | dept\_type | dept\_no\_of\_emp

62. Karakteristike SQL-a

Njegove osnovne karakteristike su:

1. Jednostavnost i jednoobraznost pri korišćenju. Tabela (relacija) se kreira jednom izvršnom naredbom. Odmah po kreiranju, tabela je raspoloživa za korišćenje. Svi podaci memorisani su u tabelama i rezultat bilo koje operacije se logički prikazuje u obliku tabele.

2. Mogućnost interaktivnog i klasičnog (aplikativnog) programiranja. Koristeći SQL dobijaju se odgovori na trenutne, unapred nepredviđene zahteve ili se SQL blokovi "ugrađuju" u klasični viši programski jezik (Java, PHP) omogućujući klasičnu obradu.

3. Neproceduralnost (tj. proceduralnost u minimalnom stepenu). Ni za jedan jezik se ne može reći da je potpuno neproceduralan, već da je neproceduralan u većem ili manjem stepenu. SQL je u velikoj meri neproceduralan, jer definiše ŠTA, a ne KAKO: koji podaci se žele, koje tabele se referenciraju i koji uslovi treba da budu ispunjeni, bez specifikacije procedure za dobijanje željenih podataka.

63. Čemu služe naredbe iz grupa DDL, DML, DCF?

Naredbe iz grupa DDL (Data Definition Language), DML (Data Manipulation Language) i DCL (Data Control Language) koriste se za upravljanje bazama podataka, ali imaju različite svrhe.

64. Navesti naredbe iz grupe DDL.

**DDL (Data Definition Language) - Jezik za Definisanje Podataka**:

DDL naredbe se koriste za definisanje, organizovanje i struktuiranje podataka u bazi podataka.

Tipične DDL naredbe uključuju:

* CREATE: Koristi se za kreiranje objekata u bazi podataka, kao što su tabele, indeksi ili baze podataka.
* ALTER: Koristi se za izmenu strukture postojećih objekata, poput dodavanja, brisanja ili modifikacije kolona u tabeli.
* DROP: Koristi se za brisanje objekata iz baze podataka, kao što su tabele, indeksi ili baze podataka.

65. Navesti naredbe iz grupe DML.

**DML (Data Manipulation Language) - Jezik za Manipulaciju Podacima:**

DML naredbe se koriste za manipulaciju podacima unutar baze podataka.

Tipične DML naredbe uključuju:

* SELECT: Koristi se za izbor podataka iz baze podataka.
* INSERT: Koristi se za dodavanje novih redova u tabelu.
* UPDATE: Koristi se za ažuriranje postojećih podataka u tabeli.
* DELETE: Koristi se za brisanje podataka iz tabele.

66. Navesti naredbe iz grupe DCF.

**DCL (Data Control Language) - Jezik za Kontrolu Podataka:**

DCL naredbe se koriste za kontrolu pristupa podacima u bazi podataka.

Tipične DCL naredbe uključuju:

* GRANT: Koristi se za davanje prava pristupa korisnicima ili ulogama.
* REVOKE: Koristi se za oduzimanje prava pristupa koje su prethodno dodeljene.

67. Čemu služi naredba SELECT?

Naredba SELECT u jeziku SQL (Structured Query Language) koristi se za izbor podataka iz baze podataka. Ova naredba se koristi za postavljanje upita (query) nad podacima u tabelama kako bi se dobio određeni skup rezultata koji zadovoljava zadate uslove.

68. Navesti i opisati sintaksu naredbe SELECT.

SELECT column1, column2, ...

FROM table\_name

WHERE condition

Gde:

* column1, column2, ...: Predstavljaju kolone koje želite izabrati. Možete navesti više kolona odvojenih zarezima.
* table\_name: Predstavlja tabelu iz koje želite izabrati podatke.
* condition: (Opciono) Predstavlja uslov koji moraju zadovoljiti redovi kako bi bili uključeni u rezultat. Ako želite izabrati sve redove, možete izostaviti ovu klauzulu.

69. Formiranje upita nad dve ili više tabela

SQL raspolaže sa dve različite tehnike za pravljenje upita nad više tabela:

1. korišćenje podupita tj. umetanje upita nad jednom relacijom u upit nad drugom i

2. JOIN operacije za kombinovanje više tabela.

70. Ograničenja podupita

Korišćenje podupita je povezano sa nekim ograničenjima pa je zato u mnogim situacijama za selektovanje podataka iz više tabela, korisnija upotreba JOIN operacija.

Podupiti su veoma moćno sredstvo za spajanje dveju tabela, a li oni imaju i niz ograničenja:

1. mogu se selektovati podaci samo iz tabele koja se koristi na najvišem nivou;

2. podupit se ne može koristiti za dobijanje podataka iz više od jedne tabele.

71. JOIN operacije za spajanje više tabela

Operacija JOIN, se koristi za kombinovanje povezanih torki iz dve relacije u pojedinačne "duže" torke. Ova operacija je veoma važna za bilo koju relacionu bazu podataka sa više od jedne relacije, jer nam omogućava da obradimo uspostavljene relacije (eng. relationships) koje postoje među relacijama (eng. relation).

SELECT Zaposleni.ID, Zaposleni.Ime, Odeljenje.Naziv

FROM Zaposleni

INNER JOIN Odeljenje ON Zaposleni.Odeljenje\_ID = Odeljenje. Odeljenje\_ID;

72. Korišćenje pogleda – VIEW

Rad sa pogledima je moguć korišćenjem naredbi

1. CREATE VIEW,

2. UPDATE VIEW i

3. DROP VIEW. U SQL-u,

pogled (VIEW) je virtualna tabela koja se dobija kao rezultat skupa SQL naredbi. Za kreiranje VIEW-a se koristi naredba CREATE VIEW.

CREATE VIEW BasicCustomerData AS

SELECT Name, AreaCode, PhoneNumber

FROM CUSTOMER;

73. Kada se UPDATE VIEW ne može izvršiti ?

Ažuriranje preko pogleda se ne može izvršiti ako je:

1. pogled definisan nad više tabela. T akvo ažuriranje bi značilo da se jednom naredbom vrši ažuriranje više tabela ba ze podataka, što je suprotno definiciji INSERT, DELETE i UPDATE naredbi.

2. definicija pogleda napravljena tako da se iza SELECT klauzule nalaze funkcije i aritmetički izrazi.

3. definicija pogleda napravljena tako da u nju nisu uključene sve NOT NULL kolone tabele nad kojom je pogled definisan.

74. SQL Server transakcije

SQL podržava transakcije, što omogućava izvršavanje skupa operacija kao jedne celine. Transakcije obezbeđuju da se promene na podacima izvrše u celosti ili se uopšte ne izvrše ukoliko neka od operacija ne uspe.

75. Transakcija i greške

Ako se desi greška u transakciji, SQL Server pruža mehanizme za upravljanje tim situacijama. Dva ključna elementa koja se koriste za upravljanje transakcijama su COMMIT i ROLLBACK.

COMMIT: Ako su sve operacije unutar transakcije uspešno izvršene i nema grešaka, možete koristiti naredbu COMMIT da potvrdite transakciju. Ovo znači da se promene koje su izvršene u okviru transakcije čuvaju i postaju trajne u bazi podataka.

BEGIN TRANSACTION; -- Početak transakcije

-- Operacije unutar transakcije

UPDATE Racun SET Stanje = Stanje - 100 WHERE KorisnikID = 1;

INSERT INTO Transakcije (KorisnikID, Iznos) VALUES (1, -100);

-- Ako su sve operacije uspešne, potvrda transakcije

COMMIT;

ROLLBACK: Ako se desi greška ili neka neočekivana situacija tokom izvršavanja transakcije, možete koristiti naredbu ROLLBACK da se ponište sve izmene koje su napravljene unutar transakcije. Ovo osigurava da se baza podataka vrati u stanje pre početka transakcije.

BEGIN TRANSACTION; -- Početak transakcije

-- Operacije unutar transakcije

UPDATE Racun SET Stanje = Stanje - 100 WHERE KorisnikID = 1;

-- Simulacija greške

-- SELECT 1/0; -- Ovde bi došlo do greške

-- Ako se desi greška, poništavanje transakcije

ROLLBACK;

76. Oporavak (RECOVERY) baze podataka

Jedan od veoma važnih elemenata, vezanih za server baze podataka, jeste mogućnost oporavka baze podataka (engl. data base recovery), što predstavlja povratak baze podataka u stanje pre softverskog ili hardverskog otkaza sistema. Razlozi za otkaz sistema mogu biti:

1. greške u operativnom sistemu,

2. greške u programiranju,

3. greške u samom SUBP-u ili

4. padanje glave diska, nestanak struje i dr.

77. Kako se izvodi proces oporavka ?

Proces oporavka izvodi se globalno izvodi u tri koraka:

1. periodično kopiranje (save) baze podataka na eksternu memoriju

2. zapisivanje transakcija (promena) nad bazom podataka u žurnal, tzv. log file.

◦ Transakcije se u log file mogu upisivati pre nego što se njima ažurira baza podataka, tako da ako sistem padne u trenutku između upisa transakcije u log file i ažuriranja baze podataka, postoji trag o neizvršenoj transakciji.

◦ Transakcije se u log file mogu upisivati pošto se njima ažurira baza podataka, kada postoji mogućnost da se baza ažurira ali da nema traga o tom ažuriranju u log-u pa se može desiti da korisnik ponovo unese transakciju koja je već završena.

3. oporavak baze podataka koji se može vršiti na više načina: ◦ oporavak kroz ponovnu obradu transakcija. ◦ oporavak kroz rollback ili rollforward.

78. Kako se može koristiti log file ?

- zapisivanje transakcija (promena) nad bazom podataka u žurnal, tzv. log file.

◦ Transakcije se u log file mogu upisivati pre nego što se njima ažurira baza podataka, tako da ako sistem padne u trenutku između upisa transakcije u log file i ažuriranja baze podataka, postoji trag o neizvršenoj transakciji.

79. Oporavak BP od pada sistema

Kada u toku izvršenja neke obrade padne sistem zbog npr. prestanka električnog napajanja. Oporavak se preduzima u slučaju da se, u toku izvršenja neke obrade, otkrije neki razlog koji onemogućava njeno uspešno kompletiranje. Taj razlog može biti:

1. u samoj transakciji, kao što je prekoračenje neke od dozvoljenih vrednosti (pad transakcije)

2. u sistemu, npr. prestanak električnog napajanja (pad sistema), ili

3. u disku na kome je baza podataka, npr. oštećenje glava diska (pad medija).

WAL (WRITE AHEAD LOG) PROTOKOL Obezbeđuje da se pri izvršenju operacije COMMIT, slog prvo fizički upisuje u log datoteku, pa se zatim podaci upisuju iz bafera podataka u bazu.

80. Oporavak od pada transakcije

Do planiranog završetka dolazi izvršenjem COMMIT operacije ili eksplicitne ROLLBACK operacije; Neplanirani završetak nastaje kada dođe do greške za koju ne postoji provera.

Vraćanje baze u konzistentno stanje omogućava se upisom svih informacija o ovim radnjama i operacijama u sistemski log.

81. Efekti naredbe COMMIT

Naredba COMMIT u SQL Server transakcijama ima ključne efekte koji utiču na stanje baze podataka. Kada se izvrši COMMIT, to znači da se transakcija završava uspešno i da su sve izmene koje su unutar transakcije izvršene sada trajne i vidljive za druge korisnike baze podataka. Evo ključnih efekata naredbe COMMIT:

* Potvrda izmena: Sve izmene (INSERT, UPDATE, DELETE) koje su izvršene unutar transakcije postaju trajne i vidljive svim drugim transakcijama koje pristupaju bazi podataka.
* Oslobađanje resursa: Resursi koji su bili zaključani tokom izvršavanja transakcije, poput redova ili stranica, sada su oslobođeni. Ovo omogućava drugim transakcijama da pristupe tim resursima.
* Zatvaranje transakcije: COMMIT zatvara transakciju. Nakon ovog koraka, transakcija više nije aktivna, a SQL Server prelazi u normalni režim rada.

82. Pad transakcije

Pad transakcije nastaje kada transakcija ne završi svoje izvršenje planirano. Tada sistem izvršava implicitnu – prinudnu ROLLBACK operaciju, tj. sprovodi aktivnost oporavka od pada transakcije.

83. Kontrola konkurentnosti

Da bi se sprečilo da rad jednog korisnika nad bazom podataka loše utiče na rad drugog. Mere kontrole konkurentnosti se preduzimaju da bi se sprečilo da rad jednog korisnika loše utiče na rad drugog. U mnogim slučajevima, mere konkurentnosti obezbeđuju da korisnik u obradi u kojoj učestvuje više korisnika dobije iste rezultate koje bi dobio kada bi radio sam. Kontrola konkurentnosti se vezuje za pojam transakcije koja se definiše kao skup akcija koje predstavljaju logičku jedinicu posla nad bazom podataka.

Primer: U sistemu baza podataka jedne banke, prenošenje novca sa jednog računa na drugi bi predstavljalo jednu transakciju. Ona se sastoji od bar dve akcije: isplate sa jednog i uplate na drugi račun, ali su te radnje logički nedeljive i samo se zajedno smatraju jednim poslom. Na efekte transakcije ne sme da utiče eventualno istovremeno izvršenje drugih transakcija, ili nestanak struje.

84. ACID karakteristike transakcije

Atomičnost, konzistentnost, izolacija, trajnost. Transakcija se karakteriše sledećim važnim svojstvima (poznatim kao ACID svojstva):

1. atomičnost (engl. Atomicity): transakciju treba izvršiti u celosti ili je uopšte ne treba izvršiti (nijednu njenu radnju);

2. konzistentnost (engl. Consistency): transakcija prevodi jedno konzistentno stanje baze u drugo konzistentno stanje baze;

3. izolacija (engl. Isolation): efekti izvršenja jedne transakcije su nepoznati drugim transakcijama sve dok se ona uspešno ne kompletira;

4. trajnost (engl. Durability): svi efekti uspešno kompletirane transakcije su trajni, tj. mogu se poništiti samo drugom transakcijom.

Ove osobine znače da kada se transakcija završi, njeni podaci su konzistentni i stabilni na disku.

85. Konfliktne radnje transakcije

Kao osnovne komponente transakcije, posmatraćemo objekte (npr. slogove) i dve radnje:

1. čitanje i

2. upis (ili ažuriranje).

Dve radnje u ovom modelu su konfliktne ako se obavljaju nad istim objektom a jedna od njih je radnja upisa. To znači da parovi konfliktnih radnji mogu biti samo parovi 1. (čitanje, upis) i 2. (upis, upis).

86. Problemi konkurentnosti

Problemi konkurentnosti opisani u prethodnom odeljku mogu se rešiti mehanizmom dinamičkog zaključavanja i otključavanja objekata, koji obezbeđuje linearizovanost izvršenja.

Zaključavanje objekta (postavljanje katanca na objekat) je postupak koji obezbeđuje transakciji pristup objektu, i kojim transakcija istovremeno sprečava druge transakcije da pristupe tom objektu.

Svaka transakcija na kraju svog izvršavanja otključava sve objekte koje je sama zaključala (a koje nije već otključala).

**Eksluzivno zaključavanje** zaključava objekat za bilo kakav drugi pristup, nijedna druga transakcija ne može da čita ili da menja taj objekat.

**Deljivo zaključavanje** zaključava objekat za menjanje ali ne i za čitanje, što znači da druge transakcije mogu da ga čitaju ali ne i da ga menjaju.

87. Neka su date 2 transakcije T1 i T2 koje se sastoje od po dve akcije koje se istovremeno izvršavaju nad relacijom ***Stanje\_zaliha (IdProizvoda, naziv proizvoda, stanje\_zaliha)***

**Za T1: *Akcija 1****: čita red iz relacije Stanje\_zaliha koja se odnosi na proizvod Čokolada*

***Akcija 2****: smanjuje stanje zaliha za proizvod Čokolada za 3*

**Za T2: *Akcija 1****: čita red iz relacije Stanje\_zaliha koja se odnosi na proizvod Čokolada*

***Akcija 2****: smanjuje stanje zaliha za proizvod Čokolada za 5*

* 1. Opisati scenario koji pri izvršenju ovih transakcija može dovesti do izgubljenog ažuriranja.
  2. Da li do izgubljenog ažuriranja može doći ukoliko se transakcijama T1 i T2 čitaju i ažuriraju različiti proizvodi (npr. stanje zaliha za proizvod čokolada i euro krem). *Obavezno obrazložite svoj odgovor.* 
     + - 1. *Posto obe transakcije pristupaju istom redu u isto vreme, jedna od njih dve ce izbrisati ovu drugu. Ako uzmemo da je zaliha cokolade 10, prva transakcija cita tu vrednost I smanjuje za 3, medjutim I druga transakcija cita 10 I smanjuje za 5, ona ne polazi od toga da je 3 vec smanjeno, tako da jedno azuriranje ce biti izbrisano.*
         2. *Posto transakcije pristupaju razlicitim redovima u bazi, ne dolazi do izgubljenog azuiranja.*

88. Ako se nad bazom podataka paralelno izvršavaju dve transakcije:

***TR-1 koja se sastoji od akcija ak1 i ak2*** i

***TR-2 koja se sastoji od akcija ak3 i ak4***

naznačite bar jedan redosled izvršenja ovih akcija transakcija koju bazu mogu dovesti u nekonzistentno stanje.

Recimo da se akcijom ak1 cita neka vrednost iz baze, a u isto vreme akcija a3 azurira tu vrednost, dolazi do nekonzistentnog stanja, posto nije procitana azurirana vrednost.

89. Šta predstavlja zaključavanje?

Zaključavanje objekta (postavljanje katanca na objekat) je postupak koji obezbeđuje transakciji pristup objektu, i kojim transakcija istovremeno sprečava druge transakcije da pristupe tom objektu.

90. Resursi koji se mogu zaključavati su ...

??????????????????????

91. Kod zaključavanja, postoje dve faze. Navesti ih i ukratko opisati.

Kod zaključavanja, postoje dve faze:

1. faze zaključavanja objekata i

2. faze otključavanja objekata. U fazi zaključavanja nema nijedne radnje otključavanja objekta, odnosno u fazi otključavanja više nema radnji zaključavanja

92. Optimistično zaključavanje

**Optimistično**: Pretpostavka je da se konflikt neće dogoditi.

Kod optimističnog zaključavanja, pretpostavka je da se konflikt neće dogoditi. Podaci se čitaju, transakcije se obrađuju, vrše se ažuriranja i tada se proverava da li je došlo do konflikta. Ako nije, transakcija se završava. Ako se konflikt pojavio, transakcija se ponavlja sve dok se ona ne obradi bez konflikta.

93. Pesimistično zaključavanje

**Pesimistično**: pretpostavka je da će se konflikt pojaviti.

Kod pesimističnog zaključavanja, pretpostavka je da će se konflikt pojaviti. Najpre se vrši zaključavanje, zatim obrađuje transakcija, a zatim se vrši otključavanje.

94. Stanja transakcije

Stanja transakcije u SQL Server-u odnose se na različite faze kroz koje transakcija prolazi tokom svog izvršavanja. Postoje četiri osnovna stanja koja transakcija može imati:

* **Pocetak (BEGIN):** Ovo je početno stanje u kojem transakcija ulazi kada se izda naredba BEGIN TRANSACTION. Tokom ovog stanja, transakcija još nije izvršena, ali su operacije unutar transakcije spremne za izvršavanje.
* **Izvršavanje (EXECUTING):** Transakcija prelazi u stanje izvršavanja kada počnu operacije unutar transakcije. Tokom ovog stanja, transakcija vrši izmene u bazi podataka, kao što su INSERT, UPDATE ili DELETE operacije.
* **Potvrda (COMMIT):** Ako su sve operacije unutar transakcije uspešno izvršene i nema grešaka, transakcija se potvrđuje naredbom COMMIT. Ovo stanje označava da su sve izmene postale trajne u bazi podataka.
* **Poništenje (ROLLBACK):** Ako se desi greška tokom izvršavanja transakcije ili ako postoji potreba za poništenjem izmena, transakcija se poništava naredbom ROLLBACK. Ovo stanje vraća bazu podataka u stanje pre početka transakcije.

95. Pretnje bazi podataka

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.

96. Bezbedonosni modeli

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.

97. Sigurnost (zaštita) baza podataka

???????????

98. Kako se dodeljuju privilegije?

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

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)

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;

99. Paralelna obrada baza podataka

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.

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.

100. Šta je distribuirana baza podataka?

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

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.

101. Prednosti distribuiranih baza podataka

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 Poglavlje 5 Distribuirane baze podataka 18 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

102. Distribuirana obrada upita

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.

103. 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.

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.

104. Šta su XML baze podataka?

XML (eXtensible Markup Language) baze podataka su vrsta baza podataka koje su specijalizovane za skladištenje, upravljanje i pretraživanje XML dokumenata. XML je standardni format za predstavljanje struktuiranih podataka, a XML baze podataka omogućavaju efikasno čuvanje i manipulisanje podacima u ovom formatu. Evo nekoliko ključnih karakteristika XML baza podataka:

* **Struktura podataka:** XML baze podataka često omogućavaju skladištenje podataka u formatu XML dokumenata. Ovo znači da podaci imaju hijerarhijsku strukturu sa označenim elementima.
* **Upitni jezik:** XML baze podataka obično pružaju sopstvene upitne jezike koji omogućavaju pretragu, filtriranje i dohvatanje podataka iz XML dokumenata. Najčešće korišćeni jezik za upite nad XML bazama podataka je XQuery.
* **Indeksiranje:** Kako bi se poboljšala performansa pretrage, XML baze podataka obično pružaju mehanizme indeksiranja. Indeksi omogućavaju brži pristup i pretragu podataka unutar XML dokumenata.
* **Validacija:** XML baze podataka često podržavaju mehanizme validacije kako bi se osigurala ispravnost strukture XML dokumenata pre njihovog unosa ili izmene.
* **Podrška za transakcije:** Da bi se očuvala doslednost podataka, XML baze podataka podržavaju transakcije. To znači da se niz operacija može izvršiti kao jedna logička celina, čime se osigurava atomičnost i doslednost promena.
* **Manipulacija podacima:** XML baze podataka pružaju funkcionalnosti za dodavanje, ažuriranje i brisanje podataka u XML formatu.
* **Primena u web servisima**: XML baze podataka često se koriste u kontekstu web servisa gde se podaci prenose između različitih sistema u XML formatu.
* **Razmena podataka:** XML baze podataka omogućavaju jednostavnu razmenu podataka između različitih aplikacija, sistema ili platformi, jer XML pruža format koji je nezavisan od programskog jezika i platforme.

105. Mapiranje šeme baze podataka u XML šemu je moguće na tri načina. Navesti ih.

Mapiranje šeme baze podataka u XML šemu je moguće na tri načina:

1. bazirano na tabelama (eng.table-based)

2. objektno-relacionim mapiranjem (eng.object-relational)

3. upitnim jezicima

Mapiranje bazirano na tabelama: individualni redovi tabela se umeću u XML dokument, s tim što relacije primarni ključ/ strani ključ koje postoje u bazi podataka određuju kako će ti redovi biti umetnuti.

106. Objektno relaciono mapiranje – iz XML perspective

U objektno-relacionom mapiranju XML dokument se posmatra kao serijsko stablo povezanih objekata kojim se na jedan očigledan način ukazuje na relacije koje postoje između njih. Objekti iz XML dokumenta se u bazu podataka mapiraju tehnikom objektno-relacionog mapiranja, što podrazumeva da se objekti mapiraju u tabele, svojstva objekata u kolone a međusobne veze između objekata u relacije primarni ključ / strani ključ.

107. Native baze podataka

Native XML baza podataka je baza podataka koja:

1. Definiše XML model podataka. Minimalni model uključuje elemente, atribute, tekst i redosled dokumenata

2. XML dokument koristi kao osnovnu logičku jedinicu za pamćenje podataka Može koristiti bilo koju fizičku strategiju za pamćenje

XML modeli podataka se koriste kao osnova za XML upitne jezike i definišu minimalnu količinu informacija koju native XML baze podataka moraju da pamte.

108. Potreba za native XML bazama podataka

Najbolji način da se razume potreba za native XML bazama podataka je analizirati primere iz realnog okruženja. Na primer:

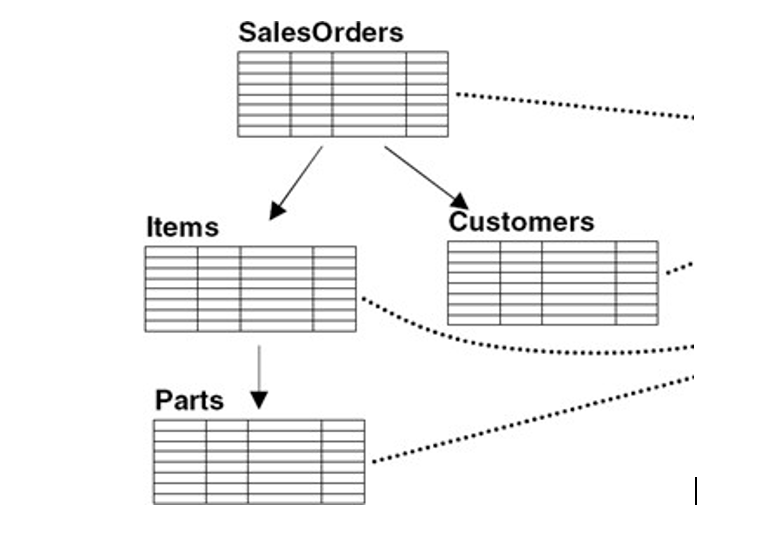
1**. Upravljanje dokumentima** kao što je korisnička dokumentacija, marketinške brošure, Web stranice itd. Native XML baze podataka posebno odgovaraju ovim slučajevima jer se XML modeli podataka dobro uklapaju u ovu vrstu dokumenata.

2. **Polustruktuirani podaci** koji se takođe dobro uklapaju u XML model podataka iz dva razloga. Prvo, XML model podataka može smestiti razgranatu šemu bez gubljenja prostora. Drugo, XML model podataka je proširiv, pa se ne zahteva fiksna šema.

3. **Transakcije čije izvršenje traje dugo**. Mnoge transakcije elektronske trgovine danas koriste XML dokumente kao način za razmenu informacija između različitih delova aplikacije. Pošto ove transakcije često mogu da traju nekoliko nedelja (npr, svaki deo mora da bude potvrđen od strane nekog ovlašćenog lica), native XML baze podataka su dobar način za pamćenje jednog stanja transakcije dok je ona u fazi izvršenja, čak i kada se krajnje stanje te transakcije pamti u relacionoj bazi podataka. Native XML baza podataka dozvoljava da se nad tekućim stanjem transakcije postavi upit korišćenjem XML upitnih jezika, a omogućava i korišćenje drugih XML alata kao što je XSLT transformacija.

4. **Arhiviranje dokumenata**. Mnoge kompanije, kao što su one u farmaceutskoj i finansijskoj industriji, moraju da arhiviraju dokumente jer ih na to obavezuju pravni propisi. Ako su to XML dokumenti, tada je native XML baza podataka prirodni izbor za arhiviranje, jer njena podrška za XML upitni jezik dozvoljava da se dokumenti koriste kao izvor istorijskih podataka koji se često koriste pri analizi trendova.

109. Na osnovu date relacione baze podataka prikazane na slici, kreirati XML dokument.



<Database>

<Customers>

<Customer>

<CustomerID>1</CustomerID>

<CustomerName>Customer1</CustomerName>

</Customer>

</Customers>

<Items>

<Item>

<ItemID>101</ItemID>

<SalesOrderID>1</SalesOrderID>

<PartID>201</PartID>

</Item>

</Items>

<Parts>

<Part>

<PartID>201</PartID>

<PartName>Part1</PartName>

</Part>

</Parts>

<SalesOrders>

<SalesOrder>

<SalesOrderID>1</SalesOrderID>

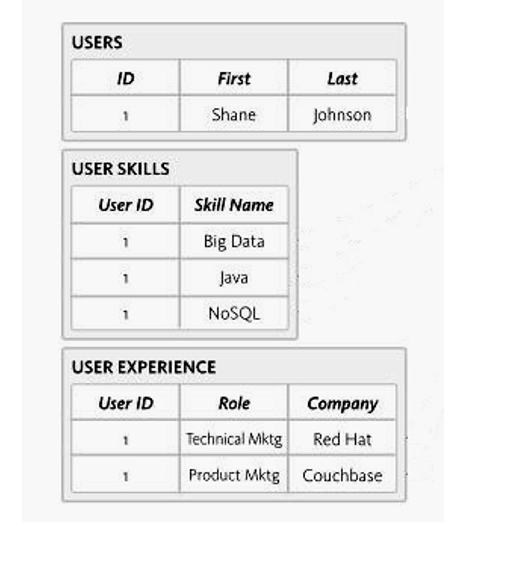
<CustomerID>1</CustomerID>

</SalesOrder>

</SalesOrders>

</Database>

110. Na osnovu date relacione baze podataka kreirati XML dokument.



**<**Database>

<Users>

<User>

<ID>1</ID>

<First>Shane</First>

</User>

</Users>

</Database>

Itd.

111. Implementacija native XML BP – navesti načine

Native XML baze podataka se mogu implementirati na jedan od sledećih načina. Pamćenjem

1. Celih XML dokumenata u CLOB strukturama relacionih baza podataka i indeksiranjem individualnih vrednosti elemenata i atributa.

2. Parsiranih XML dokumenata u fiksnim skupovima tabela (elementi, atributi, tekst itd.) u relacionoj bazi podataka.

3. Parsiranih XML dokumenata kao DOM stabla u objektno-orijentisanoj bazi podataka.

4. Parsiranih XML dokumenata u indeksiranom skupu heš tabela

112. Xquery

XQuery je deklarativni upitni jezik dizajniran za pretragu, izdvajanje i manipulaciju XML dokumenata. Razvijen od strane W3C (World Wide Web Consortium), XQuery je postao standard za obradu XML podataka. Ovaj jezik omogućava korisnicima da definišu kompleksne upite i transformacije nad XML dokumentima, pružajući snažne alatke za rad sa struktuiranim XML podacima.

113. NoSQL baze podataka - "Not only SQL"

NoSQL (Not Only SQL) baze podataka predstavljaju klasu baza podataka koje se razlikuju od tradicionalnih relacijskih baza podataka. Ova vrsta baza podataka je projektovana za rukovanje sa raznolikim, distribuiranim, i često velikim skupovima podataka.

114. Karakteristike NoSQL baza podataka

Ključne karakteristike NoSQL baza podataka uključuju:

* **Šema otvorenog tipa:** Za razliku od relacijskih baza podataka koje zahtevaju unapred definisanu šemu, NoSQL baze podataka često podržavaju šemu otvorenog tipa. To znači da se podaci mogu dodavati bez potrebe za unapred definisanjem strukture.
* **Fleksibilnost u modeliranju podataka:** NoSQL baze podržavaju različite modele podataka, uključujući ključ-vrednost, dokument, kolonijalnu i grafovsku bazu podataka. Ovo omogućava odabir odgovarajućeg modela u zavisnosti od potreba aplikacije.
* **Horizontalno skaliranje:** NoSQL baze podataka su često dizajnirane za horizontalno skaliranje, što znači da se mogu proširivati dodavanjem novih servera u sistem kako bi se rukovalo povećanjem opterećenja.
* **Brza dostupnost i visoka dostupnost:** NoSQL baze često prioritizuju brzi pristup podacima i visoku dostupnost, čak i u uslovima kada su neki od servera nedostupni.
* **Otpornost na otkazivanje:** NoSQL sistemi su često otporni na otkazivanje i projektovani tako da nastave sa radom čak i ako jedan ili više servera nisu dostupni.
* **Optimizovane za specifične zadatke:** NoSQL baze podataka su često optimizovane za specifične tipove zadatka, kao što su brza čitanja ili pisanja, analiza velikih skupova podataka ili upravljanje podacima u realnom vremenu.
* **Primena u različitim domenima:** NoSQL baze podataka se često koriste u različitim domenima, uključujući veb aplikacije, analizu podataka, IoT (Internet of Things), geoprostorno modeliranje, i druge oblasti gde se javljaju veliki, distribuirani ili polustrukturirani podaci.

115. Konzistentnost NoSQL baza podataka

Konzistentnost je atribut baze kojioznačava kakav pogledna podatke ima baza podataka tj. da li svako čitanje posle upisivanja odmah prikazuje upisanu vrednosti podataka ili ne. Visoka konzistentnost je potrebna kod proizvoda koji koriste transakcije. Konzistentnost se često deli u dva nivoa:

1. ACID konzistentnost: predstavlja atomske transakcije i obično znači da će svaki upit nakon upisa videti izmenjene podatke.

2. BASE konzistentnost: znači da će upisani podaci pre ili kasnije početi da se prikazuju u upitima.

116. BASE konzistentnost

ACID znači:

1. Atomičnost (eng. Atomic): Sve operacije u transakciji moraju biti uspešne ili se transakcija mora poništiti ( rolled back).

2. Konzistentnost (eng. Consistent): Po završetku transakcije, baza podataka je u konzistentnom (ispravnom) stanju.

3. Izolovanost (eng. Isolated): Akcije transakcija se sekvencijalno izvršavaju jedna nakon druge. Nije moguće nesekvencijalno, naizmenično izvršenje akcija transakcija. 24

4. Trajnost (eng. Durable): Rezultat primene transakcija je trajan, osim ako ne postoji greška. Ove osobine znače da kada se transakcija završi, njeni podaci su konzistentni i stabilni na disku.

------------------------------------------------------------------------------------------------------------------------------------------

BASE konzistentnost znači dabaza nema ACID garancije. BASE ima sledeće značenje:

1. Osnovna raspoloživost (eng. Basic availability): Ima se utisak da se sve vreme vrši pamćenje podataka

2. Soft-state: Pamćenje ne mora da bude konzistentno sa upisom, niti različita ažuriranja moraju biti međusobno konzistentna sve vreme.

3. Konačna konzistentnost (eng. Eventual consistency): Konzistentnost se javlja u nekom kasnijem trenutku

117. Modeli podataka za čuvanje u NoSQL bazama

U NoSQL bazama podataka korite se sledeći model podataka:

1. čuvanje dokumenata po ID-u,

2. čuvanje ključeva-vrednosti (eng. Key-Value Stores),

3. čuvanje familija kolona (eng. Column Family) i

4. graf baze podataka

118. Čuvanje dokumenata po ID-u

Čuvanje dokumenata po ID-u: Na najjednostavnijem nivou, dokumenti se mogu čuvati i pretraživati po ID. U opštem slučaju, pamćenje podataka se bazira na indeksima koji olakšavaju pristup dokumentima na osnovu njihovih atributa. Na primer, indeksi se mogu koristiti za predstavljanje različitih tipova proizvoda kako bi se nudili potencijalnim prodavcima

119. Čuvanje ključeva-vrednosti (eng. Key-Value Stores)

Čuvanje ključeva-vrednosti (eng. Key-Value) je sličan pamćenju familije dokumenata, i potiče sa Amazon’s Dynamo baze podataka. On deluju kao velika distribuirana hashmap struktura podataka koja čuva i pretražuje nerazumljive vrednosti po ključevima. Prostor sa ključevima hashmap-a se širi preko velikog broja cilindara (eng. buckets) na mreži. Zbog tolerancije na greške, svaki cilindar se replicira na nekoliko mašina.

Sa tačke gledišta korisnika, Key-Value način pamćenja je lak za korišćenje. Klijent čuva element podataka heširanjem nekog identifikatora koji je specifičan za domen problema (ključ).

120. Čuvanje familija kolona (eng. Column Family)

Čuvanje familija kolona (eng. Column Family) način čuvanja podataka je modelovan na Google’s BigTable. Model podataka se bazira na retko popunjenim tabelama čiji redovi mogu da sadrže proizvoljne kolone, ključeve koji omogućavaju prirodno indeksiranje. Na slici 6.2 se vide četiri zajednička bloka za izgradnju (eng. building blocks) koja se koriste u Column Family bazama podataka:

1. Kolona (eng. Column)

2. Super kolona (eng. Super Column) koja se dobija kombinacijom bilo kog broja kolona

3. Familija kolona (eng. Column Family) koja se dobija kada se više kolone zapamti kao red tabele koji ima svoj ključ

4. Super familija kolona (eng. Super column family) koja se dobija kada se više super kolona zapamti kao red tabele koji ima svoj ključ

121. Kolonski orijentisane baze podataka

Kolonski orijentisane baze podataka (eng. column-oriented databases) su vrsta baza podataka koje su projektovane da efikasno rukuju sa velikim količinama podataka i analizom podataka. Ova vrsta baza podataka se razlikuje od tradicionalnih relacijskih baza podataka koje su redno orijentisane (engl. row-oriented), gde su podaci organizovani po redovima. Kod kolonski orijentisanih baza podataka, podaci su organizovani po kolonama.

Ključne karakteristike kolonskih orijentisanih baza podataka uključuju:

* **Skladištenje po kolonama:** Umesto da čuva sve vrednosti za red u jednom bloku, kolonski orijentisane baze podataka čuvaju sve vrednosti za jednu kolonu u istom bloku. Ovo omogućava efikasnu analizu podataka, jer se podaci u jednoj koloni nalaze zajedno u memoriji.
* **Visok stepen kompresije:** Zbog toga što su podaci slični u jednoj koloni, često se mogu koristiti algoritmi kompresije koji daju visok stepen kompresije podataka. Ovo smanjuje zahtev za prostorom za skladištenje podataka.
* **Efikasna analiza i agregacija podataka:** Kolonski orijentisane baze su posebno efikasne kada je potrebno izvršiti analizu nad velikim skupovima podataka, posebno kada su potrebni agregatni rezultati ili kada su tražene samo određene kolone.
* **Optimizacija za rad sa velikim količinama podataka:** Kolonske baze podataka su često optimizovane za rad sa velikim količinama podataka, što ih čini pogodnim za skladištenje podataka u data warehouse-ovima i analitičkim sistemima.
* **Podrška za paralelizaciju i horizontalno skaliranje:** Kolonski orijentisane baze podataka često podržavaju paralelizaciju operacija, što znači da se analitički upiti mogu izvršavati brže. Takođe, horizontalno skaliranje (dodavanje novih servera) se lako postiže zbog organizacije podataka po kolonama.

122. Graf baze podataka (eng. Graph Databases)

Grafovne baze podataka su poseban tip baza podataka dizajniran za efikasno upravljanje i analizu podataka koji su predstavljeni u vidu grafova. Grafovne baze koriste strukturu podataka koja se sastoji od čvorova (nodes) koji predstavljaju entitete i grana (edges) koje predstavljaju veze između entiteta. Ova struktura omogućava jednostavno modeliranje i analizu složenih odnosa i povezanosti između različitih entiteta.

Ključne karakteristike grafovih baza podataka uključuju:

* **Model podataka zasnovan na grafu:** Grafovne baze podataka koriste graf kao osnovni model podataka. Graf se sastoji od čvorova koji predstavljaju entitete i grana koje predstavljaju veze između tih entiteta. Svaki čvor i grana mogu sadržavati dodatne atribute.
* **Brza navigacija i upiti:** Grafovne baze omogućavaju brzu navigaciju između čvorova i izvršavanje upita koji traže specifične obrasce u grafu. To čini grafovne baze pogodnim za analizu mreža i povezanosti.
* **Otpornost na promene:** Grafovne baze podataka su fleksibilne i dobro se prilagođavaju promenama u strukturi podataka, što ih čini pogodnim za modele podataka koji često evoluiraju.
* **Modeliranje kompleksnih odnosa:** Grafovne baze omogućavaju modeliranje kompleksnih i međusobno povezanih odnosa između entiteta. Ovo je korisno u situacijama gde su odnosi između podataka jednako važni kao i sami podaci.
* **Upotreba u preporukama i analizi socijalnih mreža:** Grafovne baze su često korišćene u sistemima preporuka i analizi socijalnih mreža, gde se mogu efikasno predstavljati veze između korisnika, proizvoda ili bilo kojih drugih entiteta.

123. Da li se NoSQL baze mogu koristiti umesto graf baza podataka?

NoSQL baze podataka i grafovne baze podataka imaju različite modele podataka i primene, ali u određenim situacijama NoSQL baze podataka mogu se koristiti kao alternativa grafovima. Međutim, važno je razumeti razlike između ove dve vrste baza podataka kako biste doneli informisane odluke o njihovom korišćenju.

NoSQL baze podataka obuhvataju različite tipove baza podataka, uključujući ključ-vrednost, dokumentarne, kolonijalne i orijentisane na porodice. Ove baze podataka su opšteg karaktera i mogu rukovati različitim vrstama podataka i modela. NoSQL baze podataka su često fleksibilne i skalabilne, što ih čini atraktivnim za određene scenarije.

Grafovne baze podataka, s druge strane, su specifično dizajnirane za rad sa podacima predstavljenim u obliku grafa. Ove baze podataka su optimizovane za brze upite i analize povezanosti između entiteta. Grafovne baze su posebno korisne kada je važna analiza relacija i struktura podataka.

Kada se NoSQL baze podataka koriste umesto grafovih baza, to se obično radi u situacijama gde je analiza veza manje ključna, a fleksibilnost modela podataka i brza skalabilnost NoSQL sistema više odgovara potrebama aplikacije. Na primer, dokumentarne baze podataka koje podržavaju složene strukture podataka (npr. JSON dokumenti) mogu se koristiti za čuvanje podataka koji sadrže veze između entiteta.

124. Čuvanje dokumenata u NoSQL bazama podataka

U NoSQL bazama podataka, podaci se čuvaju u JSON formatu: to je tekstualni format koji je kompletno nezavisan od programskog jezika. JSON (eng.JavaScript Object Notation) je format za predstavljanje podataka, lak je za čitanje i pisanje, a s druge strane je lak i mašinama za generisanje i parsiranje.

JSON koristi dve strukture:

1. Kolekciju parova ime/vrednost koje se u različitim jezicima mogu zvati objekti, slogovi, strukture, rečnici…

2. Raspoređenu listu vrednosti koja se često naziva niz, vektor, lista, ili sekvenca.

125. JSON format za čuvanje podataka

**Objekat** je set ime/vrednost parova. Definicija objekta počinje otvorenim vitičastim zagradama { i završava se zatvorenim vitičastim zagradama }. Svaki naziv prate dve tačke a ime/vrednost parovi su međusobno odvojeni zarezima.

{"firstName":"John", "lastName":"Doe"}

Definicija **niz-a** u JSON-u počinje otvorenom uglastom (srednjom) zagradom i završava se zatvorenom uglastom zagradom. Vrednosti u niz-u su odvojeni zarezom.

{"employees":[

{"firstName":"John", "lastName":"Doe"},

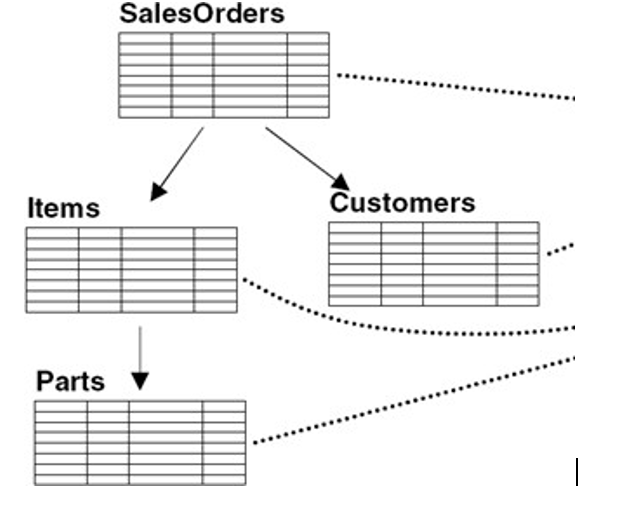
{"firstName":"Anna", "lastName":"Smith"},

{"firstName":"Peter", "lastName":"Jones"}

]}

**Vrednosti** u JSON-u mogu da budu string, broj, true, false ili null, ili objekat ili niz. Ove strukture mogu da budu ugnježdene.

126. Na osnovu date relacione baze podataka (slika 1) kreirati JSON dokument.



{

"Database": {

"Customers": [

{

"CustomerID": 1,

"CustomerName": "Customer1"

}

],

"Items": [

{

"ItemID": 101,

"SalesOrderID": 1,

"PartID": 201

}

],

"Parts": [

{

"PartID": 201,

"PartName": "Part1"

}

],

"SalesOrders": [

{

"SalesOrderID": 1,

"CustomerID": 1

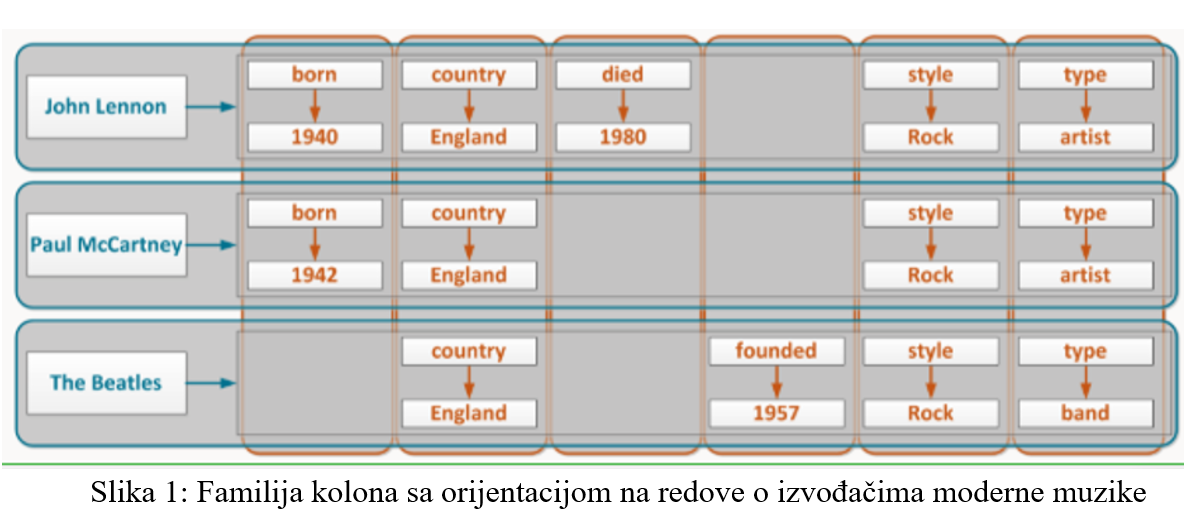
}

]

}

}

127. Na slici 1. je data familija kolona sa orijentacijom na redove koja sadrži podatke o izvođačima moderne muzike.   
Podatke sadržane u ovoj familiji kolona predstavite u JSON formatu pod pretpostavkom da je ime izvođača ključ svakog reda u familiji kolona.



{

“John Lenon” : {

“born”: 1940,

“country”: “England”,

“died”: 1980,

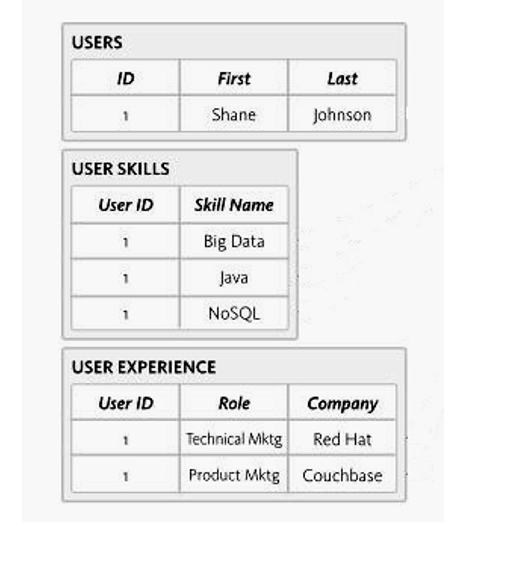
“style”: “Rock”,

“type”: “artist

}, itd……………………

}

128. Na osnovu date relacione baze podataka kreirati JSON dokument.



{

“Users”: [

{

“ID” : 1,

“First”: “Shane”,

“Last”: “Johnson”

}

], ………………itd

}

129. Elementi NoSQL baze orijentisane na kolonama

????