**IT350**

**1. Kako se podaci trajno ĉuvaju?**Podaci se trajno čuvaju na sekundarnim memorijama, tako da im se može pristupiti i kada programi završe sa svojim izvršenjem.  
Trajni podaci mogu biti sačuvani u datotekama ili bazama podataka.

**2. Slogovi u datoteci mogu imati razliĉite forme. Navesti ih.**• **Slogovi fiksne dužine**• **Slogovi promenljive dužine**• **Zaglavlja i detalji**• **Tagovani podaci**

**3. Postoji više naĉina na koje file-ovi mogu biti organizovani, i to ...  
Serijski, sekvencijalno, random.**

**4. Navesti naĉine za pristup podacima u file-ovima.  
Serijski pristup** podrazumeva čitanje file-ova slog po slog; Kod **direktnog** pristupa se koristi algoritam za konverziju vrednosti ključa u adresu sloga.**Indeks-sekvencijalni pristup**: Pronalaženje sloga se vrši sekvencijalnim pretraživanjem područja indeksa, dok se ne nađe traženi ključ, a zatim se na osnovu adrese ključa direktno pristupa traženom slogu

**5. Šta je baza podataka?**Baza podataka predstavlja kolekciju međusobno povezanih podataka koji su organizovani u tabele i druge strukture podataka, a koriste za jednu ili više aplikacija. Podaci mogu biti različitog tipa: tekstualni, numerički, slike, audio i video zapisi i sl.

**6. Šta ĉini sistem baze podataka?  
Sistem baze podataka** predstavlja skup organizovanih podataka za brzu pretragu tj (Baza podataka) kao i sistem za administraciju, memorisanje i organizovanje tih podataka (DBMS). **7.Tri-šeme arhitektura.** Tri-šeme arhitektura 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.  
  
**8. Šta nudi DBMS?  
 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?**

Prednosti:

Dobro napravljena baze moze da nam ustedi dosta vremena i novca, brzu pretragu, laka dostupnost informacija, izvodljivost statistickih proracuna

Mane:

Skupoca ispravke,

Potrebno je vreme za kreiranje dobre baze,

Lose napravljena baza moze da se urusi,

**10. Navesti tipove BP.**Baze podataka mogu biti:  
1. Relacione  
2.Objekne   
3.Objektno relacione **11. Koje su osnovne razlike izmedju tipova BP?**- Za razliku od relacionih modela, u svetu objektno-orijentisanih baza podataka ne postoji univerzalno prihvaćeni dogovor o modelima podataka. Svaki OOMP mora minimalno posedovati sledeće osobine:

• Mora biti obezbeđena funkcionalnost baze podataka   
• Mora podržavati identifikaciju objekata   
• Mora postojati mogućnost enkapsulacije   
• Mora podržavati objekte sa kompleksnim stanjem.

Dok objekto-relacione baze koriste objkne koncepte kao sto su nasledjivanje, abstrakcija, enkapsulacija, polimorfizam u kombijaciji sa relacijama, koncept relacije je proširen dopuštanjem da atributi kao domene ne moraju imati samo proste, nego i korisničke tipove podataka. Ukoliko su elementi relacije primeri korisničkog tipa podataka ona se naziva objektna relacija.. Ukratko ona je kombinacija relacionog i objeknog modela.

**12. Šta ĉini koncepcijske osnove relacionog modela?**Koncepcijske osnove relacionog modela spadaju:   
• Funkcije,   
• Dekartov proizvod skupova,  
• Odnosi između skupova  
**13. Kako se definiše relacija u relacionom modelu baze podataka?**  
Relacija u relacionom modelu se može definisati kao skup parova atributa relacje i imena domena.Relacija se uobičajeno opisuje preko tabele.Sve relacije su tabele, ali sve tabele nisu relacije.Relacija se moze definisati izmedju tabela tako sto cemo zapamtiti primarni kljuc jedne tabe kao strani kljuc u drugoj tabeli. Ovim naznacavamo njihovu vezu i mozemo raditi join operacije sa njima.

**14. Objasniti šta se podrazumeva pod “relacija kao imenovana tabela”?**Odnosi se na samu tu relaciju izmedju dve tabele, Npr ako imamo odnos vise : vise onda koristimo medjutabelu koja ce pamtiti njihove primarne kljuceve i mozemo toj tabeli dati ime tj. Imenovati relaciju.

**15. Objasnite šta je:**

**a. Primarni kljuĉ**Vrednost primarnog ključa jedinstveno identifikuje svaku torku u relaciji (red u tabeli).

**b. Strani kljuĉ  
Strani kljuc** se pamti u tabeli kada imamo vezu izmedju dve tabele i ona predstavlja primarni kljuc n-torke drugoe tabele s kojom je poveza. **\*c. Kandidat kljuĉ  
Kandidat ključevi** su alternativni identifikatori jedinstvenih redova u tabeli. Umesto termina kandidat ključ, često se koristi i termin alternativni ključ.

**\*d. Surogat kljuĉ  
Surogat ključ** je identifikator kojeg generiše RDBMS, njegov redosled je jedinstven u okviru jedne tabele i nikada se ne menja . On se dodeljuje u trenutku kreiranja reda u tabeli a uništava se kada se red obriše.

**16. Objasnite operatore relacione algebre pogodne za ažuriranje.**

Osnovne operacije ažuriranja treba da omoguće: UPIS n-torki u relaciju, BRISANJE n-torki iz relacije, MENJANJE n-torki u relaciji

**17. Objasnite operatore relacione algebre za izveštavanje.**

Operacije za izveštavanje koje se baziraju na uklanjaju delova relacija kod kreiranja rezultata: presek eliminiše sve što nije zajedničko skupovima, selekcija eliminiše neke redove koji ne ispunjavaju tražene uslove kod izveštavanja, i projekcija eliminiše neke kolone.

**18. Šta su modeli i ĉemu služe?  
Model podataka definiše strukturu i značenje podataka.**Ovaj dokument se prvenstveno bavi modelima podataka koji omogućavaju ponovnu upotrebu podataka u različitim aplikacijama, bilo integrisanjem i razmenom podataka unutar jedne baze podataka, ili razmenom podataka na neke druge načine kao što je prenos datoteka.

**19. Osnovni koncepti E/R modela su...**

ER dijagrami se obično implementiraju kao relacione baze podataka. U jednostavnoj implementaciji relacione baze, svaka tabela predstavlja jedan tip entiteta, svaki red tabele predstavlja jednu instancu tipa entiteta, a svako polje u tabeli predstavlja atribut. U relacionoj bazi podataka odnos između entiteta se implementira čuvanjem primarnog ključa jednog entiteta kao pokazivača ili "stranog ključa" u tabeli drugog entiteta.

**20. Koji su koraci u kreiranju konceptualnog modela?**1. Odrediti tipove entiteta   
2. Odrediti atribute tipova entiteta  
3. Odrediti relacije između tipova entiteta

**\*Najčešći scenario je:**

1. Poći od tekstualnog opisa problema   
 2. Analizirati ga i uvesti neophodne pretpostavke   
 3. Kreirati ER dijagram koji odslikava situaciju i eksplicira relacije među tipovima entiteta  
 4. Korak od ER dijagrama ka implementaciji baze podataka je pravolinijski   
 5. Kreiranje baze podataka i jeste naš glavni cilj

**21. Šta je entitet?**Jedan entitet je jedan pojavni oblik datog tipa entiteta. Svaki entitet mora imati identitet različit od svih ostalih entiteta.  
može biti objekat sa fizičkim (stvarnim) postojanjem (osoba, automobil, kuća ili zaposleni) ili objekat sa konceptualnim (apstraktnim) postojanjem (posao, predmet koji se sluša na fakultetu, utakmica itd.)

**22. Šta su atributi i kako se mogu klasifikovati?**Atribut predstavlja svojstvo tipa entiteta (entiteta) ili relacije.  
Domen atributa predstavlja skup vrednosti koje se mogu dodeliti atributu

*Primer: domen atributa Prezime entiteta Zaposleni predstavlja skup svih prezimena koji zaposleni imaju.*

Atributi se mogu klasifikovati na:   
 1. proste i složene,   
 2. sa jednom vrednošću i sa više vrednosti,   
 3. izvedene

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

**24. Opisati kardinalnost relacije u relacionom modelu.**Kardinalnost relacije (eng. cardinality) se definiše kao broj pojavljivanja jednog tipa entiteta koji se mogu povezati sa jednim pojavljivanjem drugog tipa entiteta.U svakoj relaciji se određuje maksimalna kardinalnost i minimalna kardinalnost **25. Koje su karakteristike rekurzivne relacije?**Rekurzivna relacija je relacija u kojoj jedan isti entitet učestvuje više puta sa različitim ulogama. Ovakve relacije se nekada označavaju i ako unarne relacije

**26. Šta je domen?**Domen predstavlja ogranicenje vrednosti atributa tj. Atribut mora da bude odredjenog tipa i mora a bude u odredjenom opsegu koji je definisan domenom atributa.

**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.  
Ima 5 komponenti  
1. Poslovno pravilo je recenica koja opisuje sta se izvrsava i neki uslov koji mora da bude zadovoljen npr. Da bi podigli neki novac moramo da imamo taj iznos ili veci na nasem racunu.  
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 **Iznos (iz ODJAVA) <= Iznos (Iz RACUN).**5 Akcija: koja se izvršava kada se operacija trigeruje

.\***28. Zbog ĉega je proces modeliranja podataka uvek iterativan?**

**29. Kako glase pravila analize teksta?**

**Pravila “analize teksta” koja glase:**   
1. Zajedničke imenice ili imenice koje nisu vlastite su kandidati za tipove entiteta   
2. Vlastite imenice podrazumevaju instance entiteta   
3. Zbirne imenice podrazumevaju tipove entiteta sačinjen od grupe drugih entiteta   
4. Pridev podrazumeva atribut tipa entiteta   
5. Glagol “biti“ podrazumeva relaciju generalizacije između instanci entiteta i tipa entiteta   
6. Glagol „imati“ podrazumeva vezu agregacije ili asocijacije

**30. Da li ternarna relacija iskljuĉuje upotrebu binarnih relacija?**Ako binarni odnos sadrži informaciju koja se razlikuje od one sadržane u ternarnom, binarni odnosi se zadržavaju.

**31. Kako se odreĊuje kardinalnost ternarne relacije?**

1. Prekriti sve relacije (linije) koje idu od ternarne relacije (npr. ESP) prema nekom tipu entitetu, osim jedne. (Na primer, prekriti linije iz ESP prema EMPLOYEE i PROJECT.)   
2. Za preostali tip entiteta, zapitati se: da li može da postoji najviše jedan entitet tog tipa entiteta, za svaku pojedinačnu kombinaciju preostalih entiteta, ili ih može biti više?(U našem primeru, zapitati se: da li neki zaposleni na nekom projektu može imati najviše jednu veštinu koju primenjuje na projektu, ili može primenjivati veći broj veština?) ◦ Ako je odgovor “više”, staviti simbol rakljice na liniju koja ulazi u taj tip entiteta; ◦ ako je odgovor “1”, staviti crticu preko te linije.   
3. Ponoviti proces dok se ne ispita svaki tip entiteta u relaciji.

**32. Kada se javljaju višestruke relacije?**Višestruka relacije se javljaju u slučaju kada dva entiteta grade više od jednog tipa relacije

**33. Šta je asocijativni tip entiteta?**Asocijativni entitet je specijalan slučaj relacije sa maksimalnom kardinalnošću “više prema više” koji zahteva da neki novo kreirani tip entiteta pamti atribute te relacije.

**34. Šta je specijalizacija, a šta generalizacija?  
Generalizacija je** proces minimizacije razlika između entiteta identifikovanjem zajedničkih karakteristika. Možemo razmišljati i o obrnutom procesu apstrakcije u kojem potiskujemo razlike između nekoliko tipova entiteta, identifikujemo njihove zajedničke karakteristike i generalizujemo ih u jednu superklasu čiji su originalni tipovi entiteta posebne podklase  
**Specijalizacija je** proces maksimiziranja razlika između entiteta nekog tipa entiteta identifikovanjem karakteristika koje ih razlikuju. Ovaj proces predstavlja prilaz odozgo-nadole, definisanjem superklase i njenih podklasa. Podklase se definišu na osnovu njihovih specifičnih karakteristika (atributa) u odnosu na nadklase

**35. Kakvi su to ID zavisni entiteti?  
To su entiteti** čiji identifikator uključuje identifikator drugog entiteta.

Na primer: razmotrimo slučaj apartmana (sobe) koji koriste studenti prilikom studiranja. Identifikator takvog entiteta je složen (BrojApartmana, ImeZgrade), gde je ImeZgrade identifikator tipa entiteta ZGRADA a BrojApartmana sam po sebi nije dovoljan da nekome kažete gde živite

**36. Šta su slabi, a šta jaki entiteti?**Slabi tipovi entiteti su oni čije postojanje zavisi od prisustva drugih entiteta.

**37. Šta se postiže korišćenjem podklasa i nadklasa?**Prvi je da se određeni atributi mogu odnositi na neke a ne i na sve entitete super klase;   
Drugi je da u nekim tipovima relacija mogu da učestvuju samo entiteti koji su članovi pod klasa  
  
**38. Transformacija entiteta**Transformacija tipova entiteta u logički i fizički model baze podataka se vrši tako da svaki tip entiteta iz E-R dijagramu postaje jedna relacija u relacionom, odnosno tabela u fizičkom modelu baze podataka.Identifikator tipa entiteta postaje primarni ključ tabele a drugi atributi tipa entiteta postaju atributi ne primarnih ključeva relacije (tabele).

**39. Transformacija atributa tipova entiteta**Prilikom transformacije tipova entiteta iz E/R modela u tabele baze podatka treba proveriti da li su tabele normalizovane . Drugim rečima, treba dati odgovor na pitanje da li su tabele najmanje u trećoj normalnoj formi (3NF) i da li su otklonjene sve funkcionalne zavisnosti.  
Takodje Kada se vrši transformacija atributa tipova entiteta u atribute relacije treba obraditi pažnju na sledeće:  
• null status • tip podataka   
• default vrednosti • ograničenja podataka

**40. Ograniĉenja podataka (domen, opseg vrednosti, intrarelaciono i interrelaciono  
ograniĉenje)**1. **Domen** ogranicava opseg skupa vrednosti. Na primer. KodZaposlenog može biti ograničen vrednostima ( ’novo zaposleni’, ’stalno zaposlen’, ’honorarno zaposlen’ itd.)   
**2. Opseg vrednosti** – kojim se vrednosti ograničavaju na određeni interval. Na primer: datum zaposlenja može biti u opsegu od 1. januara 1990 do 31. decembra 2007. godine.   
**3. 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.   
**4. 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.

**41. Normalizacija baze 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 ponavljan

3. dozvoljavaju efikasno ažuriranje podataka u bazi podataka i

4. izbegavaju opasnosti gubitka podataka bez znanja.

**42. Predstavljanje 1:1 veze izmeĊu jakih entiteta**

Maksimalna kardinalnost tipa jedan-prema-jedan: U 1:1 relaciji, instanca entiteta jednog tipa se odnosi na bar jednu instancu entiteta drugog tipa.

**43. Predstavljanje 1:M veze izmeĊu jakih entiteta**

Maksimalna kardinalnost tipa jedan-prema-više: U ovom slučaju jedna instanca jednog tipa entiteta može biti u vezi sa više instanci drugog tipa entiteta.

**44. Predstavljanje M:N veze izmeĊu jakih entiteta**

Minimalna kardinalnost je broj instanci tipa entiteta koji mora učestvovati u relaciji. Generalno, minimum je određen sa nula ili jedan.

**45. Asocijativne veze**

Najslabiji tip veze.

Asocijativni entitet je specijalan slučaj relacije sa maksimalnom kardinalnošću “više prema više” koji zahteva da neki novo kreirani tip entiteta pamti atribute te relacije. **46. Transformacija podklasa i njihovih nadklasa se može se vršiti na tri naĉina. Navesti ih.**Transformacija podklasa i njihovih nadklasa se može se vršiti na tri načina korišćenjem:  
1. **E/R 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.  
**47. 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.

**48. Normalizacija**  
I**deja 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 ponavljan

3. dozvoljavaju efikasno ažuriranje podataka u bazi podataka i

4. izbegavaju opasnosti gubitka podataka bez znanja.

**49. Denormalizacija**Denormalizacija predstavlja odstupanja od normalnih formi da bi se olaksalo koriscenje same baze podataka to moze da se radi kada je potrebno koristiti vise join operacija posto one uzimaju resurse pa mozemo iz tog razloga odstupiti od normalnih formi radi olaksanja posla. Ako je moguce moze se napraviti jedna strana koja je normalizovana i jedna koja je denormalizovana da normalizovanu koristili za transakcije a drugu kada su nam je potrebna ekstrakcija podataka.  
**Tipovi denormalizacije:**  
Denormalizacija spajanjem tabela  
Denormalizacija dupliciranjem/deljenjem tabela  
Denormalizacija deljenjem kolona  
Denormalnizacija dodavaljem kolona

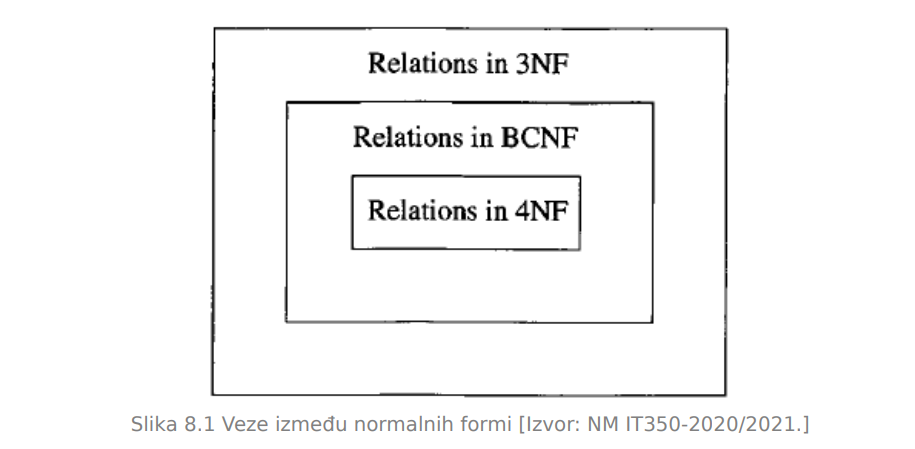
**50. Prva normalna forma (1NF)**Mozemo reci da tabelia postuje 1NF ako su svi atributi tabele atomski tj nedeljivi.

**51. Druga normalna forma (2NF)**Relacija je u drugoj normalnoj formi (2NF) ako je svaki atribut koji nije primarni ključ funkcionalno zavisan od celog primarnog ključa.

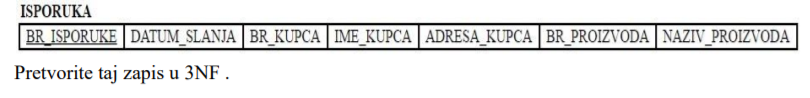
**52. Treća normalna forma (3NF)**Relacija je u trećoj normalnoj formi (3NF) ako je u drugoj normalnoj formi i ako su svi atributi koji nisu deo ni jednog ključa međusobno nezavisni, što se drugačije kaže da ne postoji tranzitivna zavisnost.

**53. Boyce Codd normalna forma (BCNF)**Često je dovođenje relacije na 3NF dovoljno da se eliminišu sve funkcionalne zavisnosti, ali postoje slučajevi gde problemi ostaju. Ti problemi se mogu eliminisati ukoliko se relacije dizajniraju ili redizajniraju svođenjem na Boyce-Codd-ovu normalnu formu (BCNF).   
Relacija ne ispunjava BCNF ako u njoj postoje funkcionalne zavisnosti kod kojih determinanta nije kandidat za ključ.  
BCNF garantuje da je sprečeno postojanje funkcionalnih zavisnosti a sve normalne forme garantuju da su sprečene zavisnosti od više vrednosti.

**54. Ĉetvrta 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

**55. Veze izmeĊu normalnih formi**4NF podrazumeva da je zadovoljena BCNF, koja podrazumeva da je zadovoljena 3NF.  


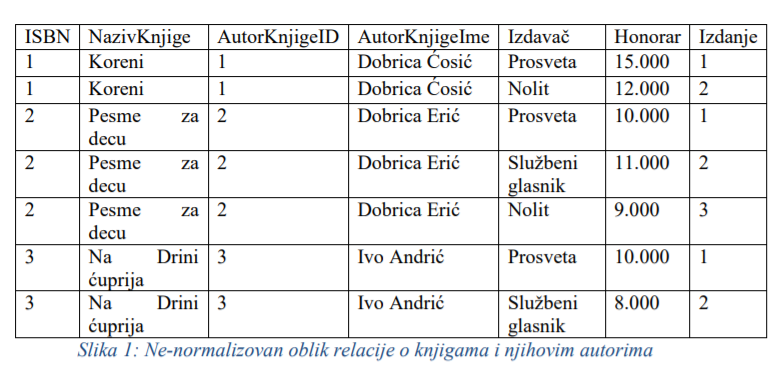
**56. Fabrika isporuĉuje svoje proizvode kupcima. Jedna isporuka šalje se jednom kupcu i  
može sadržavati više komada raznih proizvoda. Situacija je prikazana sledećim  
zapisom:**

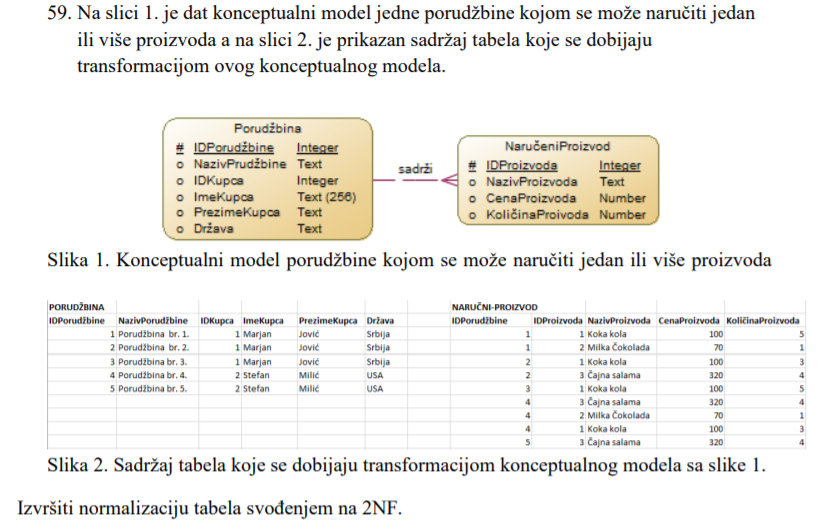


**57. Jedan sastanak održava se uvek u istoj uĉionici, ali u nekoliko vremenskih termina dnevno. Situacija je opisana relacijom**

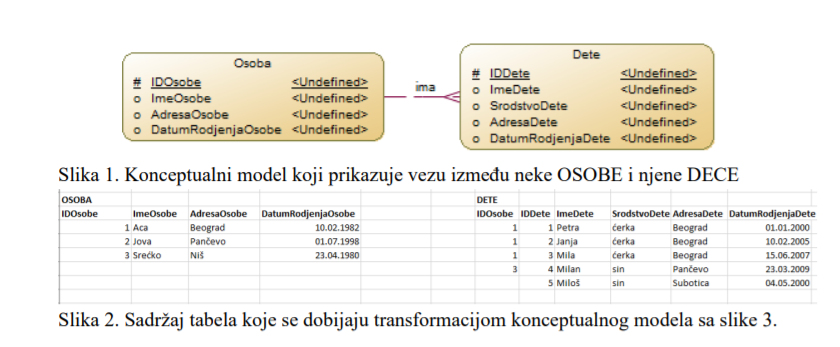


**58. Relacija na Slici 1. pokazuje da autor za svaku prodatu knjigu od svojih izdavaĉa dobija odreĊeni honorar. Iznos honorara zavisi od autora, knjige i izdanja knjige. Primarni kljuĉ relacije ĉine ISBN i AutorKnjigeID. Normalizujte ovu relaciju tako da dobijete model baze podataka u 3NF i predstavite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





**60. Na slici 1. je dat konceptualni model koji prikazuje vezu izmeĊu 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.**



**Napisati redove koji se dobijaju selekcijom imena osobe, imena deteta i srodstva deteta**

**primenom**

**a. INNER JOIN**

**b. CROSS JOIN,**

**c. LEFT (OUTER) JOIN,**

**d. RIGHT (OUTER) JOIN,**

**e. FULL (OUTER) JOIN.**

**61. Tabelu koja je predstavljena na slici 1, normalizovati i normalizovan oblik tabele**

**predstaviti u obliku E/R dijagrama**



**62. Karakteristike SQL-a**

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 zahtev+e 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.

**63. Ĉemu služe naredbe iz grupa DDL, DML, DCF?**

Naredbe za definisanje podataka (Data Definition Statements - DDL): Omogućuju definisanje resursa relacione baze podataka.

Naredbe za manipulisanje podacima (Data Manipulation Statements - DML):Omogućuju ažuriranje u širem smislu značenja te reči i izveštavanje iz relacione baze podataka.

Naredbe za kontrolne funkcije (Data Control Functions - DCF): Omogućavaju oporavak, konkurentnost, sigurnost i integritet relacione baze podataka.

**64. Navesti naredbe iz grupe DDL.**

CREATE DATABASE, ALERT TABLE, SHOW DATABASE, DROP DATABASE

**65. Navesti naredbe iz grupe DML.**

INSERT, UPDATE, DELETE, SELECT

**66. Navesti naredbe iz grupe DCF.**

**67. Ĉemu služi naredba SELECT?**

SELECT naredba u SQL-u se može sastojati od do šest klauzula, ali samo prve dve - SELECT i

FROM - su obavezne. Upit se kao i svaka druga SQL naredba završava se tačkom i zarezom.

Klauzule su navedene u sledećem redosledu, a klauzule između uglastih zagrada [...] su

opcione:

1. SELECT klauzula navodi atribute ili funkcije koje treba naći.

2. FROM klauzula specificira sve relacije (tabele) koje su potrebne u upitu, uključujući spojene (join-ovane) relacije, ali ne one u ugnježdenim upitima.

3. WHERE klauzula specificira uslove za izbor torki iz ovih relacija, uključujući i uslove spajanja ako je potrebno.

4. GROUP BY specificira atribute grupisanja, dok HAVING specificira uslov za izbor grupa, a ne za pojedinačne torke.

[HAVING... ]

[ORDER BY... ]

**68. Navesti i opisati sintaksu naredbe SELECT.**

SELECT [predikat] { \* | tabela.\* | [tabela.]polje1 [AS pseudonim1] [, [tabela.]polje2 [AS pseudonim2] [, ...]]}

FROM izraztabele [, ...] [IN spoljnabazapodataka]

[WHERE... ]

[GROUP BY... ]

[HAVING... ]

[ORDER BY... ]

[WITH OWNERACCESS OPTION]

Izraz SELECT sastoji se od sledećih delova:

Predikat - Neki od sledećih predikata: ALL, DISTINCT, DISTINCTROW ili TOP. Predikat koristite da ograničite broj vraćenih zapisa. Ako nije naveden, podrazumevani predikat je ALL.

\* - Navodi da su sva polja iz navedene tabele ili tabela izabrana.`

*polje1*, *polje2 -* Imena polja koja sadrže podatke koje želite da preuzmete. Ako uključite više od jednog polja, ona se preuzimaju po navedenom redosledu.

|  |  |
| --- | --- |
| *pseudonim1*, *pseudonim2* | Imena koja se koriste kao zaglavlja kolona umesto originalnih imena kolona u delu *tabela*. |
| *Izraztabele -* | Ime tabele ili tabela koje sadrže podatke koje želite da preuzmete. |
| *spoljnabazapodataka* | Ime baze podataka koja sadrži tabele u delu izraztabele ako se one ne nalaze u trenutnoj bazi podataka. |

**69. Formi ranje upita nad dve ili više tabela**

Join

**70. Ograniĉenja podupita**

Podupiti su veoma moćno sredstvo za spajanje dveju tabela, a li oni imaju i 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.

Operacija JOIN može biti specificirana kao operacija CARTESIAN PRODUCT koju prati SELECT operacija. Međutim, JOIN je vrlo važna operacija jer se vrlo često koristi prilikom specificiranja upita za bazu podataka.

Opšti oblik JOIN operacije za dve relacije R (A1, A2, ..., An) i S (B1, B2, ..., Bm) je:

R JOIN[join condition] S

Rezultat JOIN-a je relacija Q sa n + m atributa Q(A1, A2, ..., An, B1, B2, ..., Bm) u datom redosledu; Q ima jednu torku za svaku kombinaciju torki - jednu iz R i jednu iz S - kad god kombinacija zadovoljava uslov spajanja.

THETA JOIN jeoperacija JOIN sa opštim uslovom spajanja oblika Ai θ (teta) Bj, Ai je atribut iz R, Bj je atribut iz S, a teta je jedan od operatora poređenja { , , ≤, >, ≥, ≠}.

Najčešća upotreba JOIN-a uključuje samo uslove spajanja koji podrazumevaju poređenje na jednakost. Takav JOIN, gde je jedini operator poređenja koji se koristi =, naziva se EQUIJOIN. Primetimo da u rezultatu EQUIJOIN uvek imamo jedan ili više para atributa koji imaju identične vrednosti u svakoj torci.

u.Pošto je jedan od para atributa sa istim vrednostima suvišan, kreirana je nova operacija nazvana NATURAL JOIN- označena sa \* - kako bi se oslobodili drugog (suvišnog) atributa u EQUIJOIN uslovu. Standardna definicija NATURAL JOIN zahteva da dva atributa po kojima se vrši spajanje (ili svaki par atributa po kojima se vrši spajanje) imaju isto ime u obe relacije. Ako to nije slučaj, prvo se preimenjuje operacija preimenovanja.

1. (INNER) JOIN, 2. CROSS JOIN (CARTESIAN JOIN), 3. LEFT (OUTER) JOIN (LEFTOUTERJOIN), 4. RIGHT (OUTER) JOIN (RIGHT OUTER JOIN ), 5. FULL (OUTER) JOIN. 6. SELF JOIN

**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 ko ja se dobija kao rezultat skupa SQL naredbi. Za

kreiranje VIEW-a se koristi naredba CREATE VIEW

VIEW se može koristiti da se neke kolone sakriju kako bi se uprostila tabela ili sprečilo prikazivanje osetljivih podataka

Svrha VIEW-ova je da se prikažu rezultati izračunatih kolona bez obaveze korisnika da unese izraz za izračunavanje.

Kao što je rečeno, u WHERE klauzuli SQL naredbe se ne mogu koristiti funkcije ili izrazi kojima se nešto sračunava. Umesto toga, treba kreirati VIEW u kojem je to sračunavanje obavljeno, a zatim SELECT naredbu koja u WHERE klauzuli koristi taj VIEW.

VIEW-ovi se mogu koristiti za izolaciju izvornih tabela od aplikativnog koda

Korišćenjem VIEW-ova programeri ne moraju da unose komplikovane SQL naredbe kada žele da dobiju neki rezultat.

Naredba DROP VIEW služi za izbacivanje predhodno kreiranog pogleda.

**95. Pretnje bazi podataka**

Postoji veliki broj pretnji kojima je izložena baza podataka

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.

4. Finansijski gubitak: Većina finansijskih gubitaka nastalih zbog prevare potiče od zaposlenih.

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.

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.

**96. Bezbedonosni modeli**

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

U tu svrhu treba vršiti:

1. Kontrolu pristupa - Kontrola pristupa je skupa u smislu analize, projektovanja i operativnih troškova. Primenjuje se na poznate situacije, na poznate standarde, radi postizanja poznatih ciljeva.

2. Autentifikaciju i autorizaciju - Ovde se autentifikacija i autorizacija prvenstveno odnosi na pristup sistemima za upravljanje bazama podataka i predstavlja sagledavanje procesa iz perspektive DBA.

**97. Sigurnost (zaštita) baza podataka**

Potrebno je koristiti bezbednosne modele.

**98. Kako se dodeljuju privilegije?**

Uz pomoc naredbe **GRANT**

GRANT <Privilegija>

ON <tabele ili pogled>

TO <korisnik ili grupa korisnika>

[WITH GRANT OPTION];

**99. Paralelna obrada 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.

**100. Šta je distribuirana baza podataka?**

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.

**101. Prednosti distribuiranih baza podataka**

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.

**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.

**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

**104. Šta su XML baze podataka?**

U slučaju XML enabled baze podataka, XML se koristi za razmenu podataka između baze i neke aplikacije ili druge baze. U samoj bazi podataka se “ne vidi” XML dokument

U interakciji između XML-a i baza podataka, XML se može koristiti na dva načina:

1. u bazi podataka se može pamtiti sam XML dokument (native baze podataka)

2. XML se smešta u relacionu bazu podataka kao skup tabela eksplicitno dizajniranih za pamćenje XML dokumenata (XML enabled baze podataka)

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

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

1. bazirano na tabelama (table-based)

2. objektno-relacionim mapiranjem (object-relational)

3. upitnim jezicima

**106. Objektno relaciono mapiranje – iz XML perspective**

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

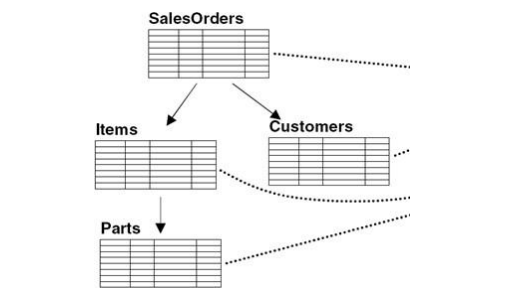
2. XML dokument koristi kao osnovnu logičku jedinicu za pamćenje podataka. Može koristiti bilo koju fizičku strategiju za pamćenje. XML modeli podataka se koriste kao osnova za XML upitne jezike i definišu minimalnu količinu informacija koju native XML baze podataka moraju da pamte.

**108. Potreba za native XML bazama podataka**

Postoji kada treba upravljati: dokumentima, polu-struktuiranim podacima, transakcijama čije izvršenje traje dugo ili kada se radi o arhiviranju dokumenata

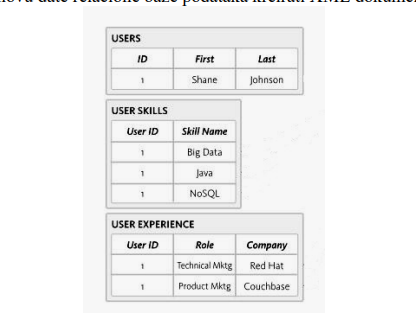
**109. Na osnovu date relacione baze podataka prikazane na slici, kreirati XML**

**dokument.**

****

**Relaciona BP Prodaja**

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



**111. Implementacija native XML BP – navesti naĉine**

**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 XML upitni jezik koji je definisan od strane W3C i koji još uvek nije završen. On je dizajniran za upite nad XML dokumentima a ne relacionim bazama podataka.

**113. NoSQL baze podataka - "Not only SQL"**

NoSQL nije baziran na šemi, tako da šema nije neophodna. Korisnici ovim dobijaju fleksibilnost u načinu na koji strukturiraju podatke.

Na primer, lista transakcija može da bude povezana sa korisnicima. Sa NoSQL bazama ova informacija je sačuvana kao agregacija, jedinstveni slog koji sadrži sve u vezi transakcija, uključujući i podatke o korisniku.

NoSQL omogućava korisnicima da korišćenjem jeftinijih servera i spajanjem više njih postignu izuzetne performanse.

Sa NoSQL-om, klaster servera može da sadrži jednu veliku bazu podataka. NoSQL proizvođači olakšavaju rad svojim korisnicima

**114. Karakteristike NoSQL baza podataka**

Četiri osnovne funkcionalnosti u kojima se ogledaju razlike između NoSQL baza podataka i

standardnih SQL sistema su:

1. Nepotrebnost šeme: Šema baze podataka je opis svih mogućih podataka i njihovih tipova

koji se u bazi mogu naći. NoSQL nije baziran na šemi, tako da šema nije neophodna. Korisnici

ovim dobijaju fleksibilnost u načinu na koji strukturiraju podatke.

2. Relacionalnost: U relacionim bazama pravimo mnogo relacija između tabela u kojima

čuvamo podatke. Na primer, lista transakcija može da bude povezana sa korisnicima. Sa

NoSQL bazama ova informacija je sačuvana kao agregacija, jedinstveni slog koji sadrži sve u

vezi transakcija, uključujući i podatke o korisniku.

3. Jeftiniji hardver: Neke baze su dizajnirane da rade samo na najboljim serverima, NoSQL

baza je drugačija i omogućava korisnicima da korišćenjem jeftinijih servera i spajanjem više

njih postignu izuzetne performanse.

4. Visoka distribuiranost: Distribuirane baze mogu da skladište podatke na više od jednog

uređaja. Sa NoSQL-om, klaster servera može da sadrži jednu veliku bazu podataka. NoSQL

proizvođači olakšavaju rad svojim korisnicima

**115. Konzistentnost NoSQL baza podataka**

Konzistentnost je atribut baze koji označava kakav pogled na 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**

U NoSQL svetu ACID transakcije se ne koriste obzirom da kod njih ne postoji striktan zahtev za trenutnom konzistentnošću, osvežavanjem podataka i tačnošću, kako bi se postigli neki drugi benefiti. Umesto korišćenja ACID, pojavio se termin BASE kao popularan način za opisivanje svojstva optimističnije strategije čuvanja podataka.

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 poU NoSQL bazama podataka korite se sledeći model podataka:**

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. Ĉuvanje dokumenata po ID-u**

Na najjednostavnijem nivou, dokumenti se čuvaju i pretražuju po ID. U opštem slučaju, pamćenje i nalaženje podataka se bazira na indeksima koji olakšavaju pristup dokumentima.

**119. Ĉuvanje kljuĉeva-vrednosti (eng. Key-Value Stores)**

Key-Value način pamćenja je lak za korišćenje. Klijent čuva element podataka heširanjem nekog identifikatora koji je specifičan za domen problema (ključ). Deluju kao velika distribuirana hashmap struktura podataka koja čuva i pretražuje nerazumljive vrednosti po ključu.

**120. Ĉuvanje familija kolona (eng. Column Family)**

Kolone se pamte kao redovi tabele i kada red sadrži samo kolone, on je poznat kao familija kolona a kada red sadrži super kolonu, on je poznat kao super familija kolona Čuvanje familija kolona (engl. 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.

**121. Kolonski orijentisane baze podataka**

Pošto svaki red daje kompletan pogled na entitet, pogled orijentisan na kolone prirodno indeksira pojedinačan aspekt tog entiteta na nivou čitavog skupa podataka.

**122. Graf baze podataka (eng. Graph Databases)**

U graf bazama podataka (eng. Graph Databases), povezani podaci se pamte na diskovima kao povezani. Na primer, razmotrimo socijalnu mrežu prikazanu na slici 6. U ovoj socijalnoj mreži, kao i u mnogim slučajevima iz realnog okruženja veze između entiteta ne prikazuju uniformnost na nivou domena, već je domen promenljivo struktuiran

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

NoSQL baze podataka čuvaju skupove nepovezanih dokumenata/ vrednosti/kolona što ih takođe čini teškim za korišćenje u slučajevima povezanih podataka ili grafova. Pošto u njoj ne postoje identifikatori koji ukazuju na veze unazad, gubimo mogućnost da izvršavamo druge interesantne upite nad bazom.

**124. Ĉuvanje dokumenata u NoSQL bazama podataka**

Na najjednostavnijem nivou, dokumenti se čuvaju i pretražuju po ID. U opštem slučaju, pamćenje i nalaženje podataka se bazira na indeksima koji olakšavaju pristup dokumentima.

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) - U column family bazama podataka svaki red u tabeli predstavlja pojedinačan sveobuhvatni entitet

4. graf baze podataka -veze između entiteta ne prikazuju uniformnost na nivou domena već je domen promenljivo struktuiran

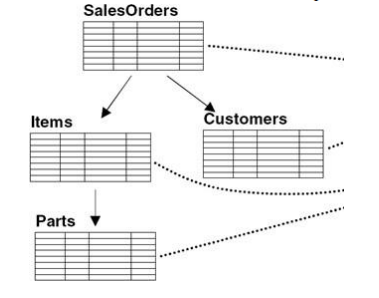
**125. JSON format za ĉuvanje podataka**

JSON (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, rekordi, strukture, rečnici… 2. Raspoređenu listu vrednosti koja se često naziva niz, vector, list, ili sekvenca

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

**Slika 1 Relaciona BP Prodaja**

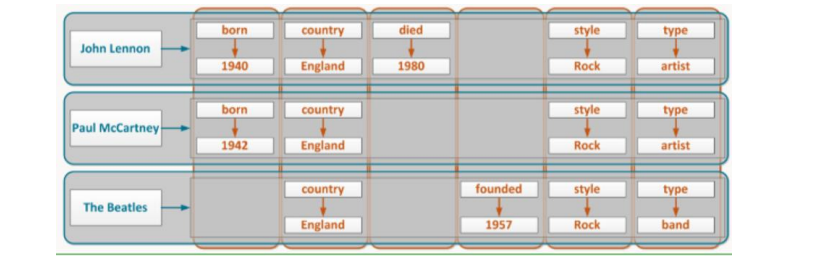
****

**\*127. Na slici 1. je data familija kolona sa orijentacijom na redove koja sadrži podatke o**

**izvoĊaĉima moderne muzike.**

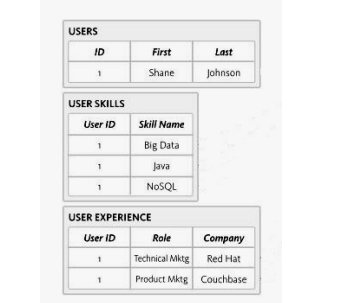
**Podatke sadržane u ovoj familiji kolona predstavite u JSON formatu pod**

**pretpostavkom da je ime izvoĊaĉa kljuĉ svakog reda u familiji kolona.**

****

**Slika 1: Familija kolona sa orijentacijom na redove o izvoĊaĉima moderne muzike**

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



**129. Elementi NoSQL baze orijentisane na kolonama**

Gledajući model podataka odozgo na dole, on se sastoji od sledećih elemenata:

1. Klaster je najudaljenija struktura NoSQL modela podataka kada je on dizajniran tako da se može distribuirati na nekoliko mašina

2. Keyspace-ova - Keyspace je krajnji kontejner za podatke

3. Familija kolona je kontejner za uređenu kolekciju redova, od kojih svaki predstavlja uređenu kolekcija kolona.

4. Kolona - Kolona je najosnovnija jedinica strukture podataka u NoSQL modelu podatakaorijentisanom na kolonama.

5. Super kolona -I regularne i super kolone su parovi ime / vrednost, ali regularna kolona čuva vrednost u obliku niza byte-ova, dok je vrednost super kolone mapa pod kolona (koje čuvaju vrednosti)

**130. Široki redovi i tanki redovi**

Široki red (eng. Wide Rows)označava red koji sadrži mnogo, mnogo (možda desetine hiljada ili čak miliona) kolona. Obično postoji mali broj redova sa toliko kolona. Suprotno tome, mogli biste imati redove bliže relacionom modelu, gde definišete manji broj kolona i koristiti mnogo različitih redova - to je mršavi model.

**\*131. Poredjenje relacionog modeliranja i modeliranja grafova**

**132. Da li se NoSQL baze mogu koristiti umesto graf baza podataka? Obrazložiti**

NoSQL baze podataka čuvaju skupove nepovezanih dokumenata/ vrednosti/kolona što ih takođe čini teškim za korišćenje u slučajevima povezanih podataka ili grafova.

Pošto u njoj ne postoje identifikatori koji ukazuju na veze unazad, gubimo mogućnost da izvršavamo druge interesantne upite nad bazom.

**133. PHP Data Objects**

PHP Data Objects (PDO) ekstenzije definišu lake i konzistentne interfejse za pristup bazama podataka u PHP-u. PDO\_MySQL - implementira PHP Data Objects (PDO) interfejs kako bi sa aplikaiji omogućio pristup MySQL bazi iz PHP-a.

**134. MySQL API-iji**

U zavisnosti koju verziju PHP-a koristimo, postoje dva ili tri PHP API-ja za pristup MySQL bazama podataka.

Ukoliko radimo sa PHP 5 verzijom možemo odbrati neki od sledećih ekstenzija

1. mysql (zastarela verzija)

2. mysqli

3. PDO\_MySQL

Počev od verzije PHP 7, ekstenzija mysql nije više podržana i ovakav pristup više nije moguć, već se koriste sledeći

1. mysqli

2. PDO\_MySQL

1. mysql - Ova ekstenzija je zastarela sa verzijom 5.5 PHP-a (i dalje ima podršku) i ukljonjena je u verziji 7 PHP-a. Umesto nje, u nadogradnjama postojećih aplikacija, potrebno je preći na mysqli ili PDO\_MySQL ekstenzije. Sve nove aplikacije ne bi smele da koriste ovu ekstenziju za pristup bazama podataka. Koristi samo proceduralni interfejs.

2. mysqli - je skraćenica za MySQL Improved ekstenziju i može se koristi sa verzijama MySQL-a počev od verzije 4.1. Ovaj ekstenzija pruža dva pristupa (intefesja) u radu sa bazama podataka

a. Objektno-orijentisani

b. Proceduralni

3. PDO\_MySQL - implementira PHP Data Objects (PDO) interfejs kako bi sa aplikaiji omogućio pristup MySQL bazi iz PHP-a. Podržane verzije MySQL-a su 3.x, 4.x, 5.x . Kao i prethodna ekstenzija, koristi OO i proceduralni interfejs.

**135.Konekcija na bazu podataka putem PHP-a Da li koristiti MySQLi ili PDO pristup otvaranja konekcije?**

Ako vam treba kratak odgovor, on bi glasio "Koji god vam se više sviđa". I MySQLi i PDO imaju svoje prednosti:

1. PDO radi sa 12 različitih sistema upravljanja bazama podataka - dok MySQLi radi

samo sa MySQL bazama podataka

2. Oba su objektno-orijentisana, dok MySQLi nudii proceduralni pristup.

3. Oba podržavaju Prepared Statements, što je veoma bitno za bezbednost web

aplikacija i štite od SQL Injection-a napada.

4. Ako morate ili mislite da će u budućnosti morati da promenite SUBP, PDO će vam

olakšati ovaj proces. Jedino što trebate da promenite je string konekcije i par upita.

Sa MySQLi morali bii ste da prepravite ceo kod - uključujući i upite.

5. Sa druge strane, MySQL je optimizovaniji za rad sa MySQL bazama od PDO-a, pa

trba i to uzeti u obzir.