IT350 – LEKCIJA 1

Osnove baze podataka

1) Zašto trajno čuvati podatke?

- 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.

- Podacima u bazi se upravlja korišćenjem sistem za upravljanje bazama podataka(**DBMS)**.

2) Kako krajnji korisnik vidi podatke sačuvane u bazama podataka?

- DBMS koristi sloj apstrakcije koji od korisnika krije činjenicu da su podaci zapamćeni u fajlovima. Ako je baza podataka **RELACIONA**, tada korisnik baze podataka vidi kao tabele koje sadrže podatke.

- Ako je baza podataka **OBJEKTNA**, tada korisnik vidi objekte i veze među njima.

3)Čuvanje podataka u fajlovima

- File je podeljen na individualne slogove od kojih svaki predstavlja grupu izvesnog broja polja koji predstavlja podatke koje treba zapamtiti u file-ovima.

Forme slogova:

- **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**.

4) Kako fajlovi mogu biti organizovani

- **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.

5)Način pristupa podacima

- **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.

****RELATIVNA ADRESA→**Ovaj metod se koristi kod slogova fiksne dužine i pozitivne sukcesivne vrednosti ključa.Ukoliko se zna da je slog veličine 200 bajtova, tada će prvi slog početi na prvom bajtu, završiti na 200, drugi će početi 201, treći 301...itd.

*****HASHING ADRESIRANJE→**Fajl ima fiksan broj blokova.Da bi se odredilo kojem će se bloku dodeliti dati slog, vrši se **heširanje ključa.**

Funkcija heširanja je algoritam koji uzima ASCII string i konvertuje ga u INTEGER.

*****INDEKS-SEKVENCIJALNI PRISTUP PODACIMA**

- Pronalaženje sloga se vrši sekvencijalnim pretraživanjem područja indeksa, dok se ne nađe traženi ključ, a zatim na osnovu adrese ključa direktno pristupa traženom slogu.

6)Problem čuvanja podataka u fajlovima

- **Redudantnost podataka→**Kako se broj fajlova povećava, sve više i više postoji šansa da se pojave različiti fajlovi koji sadrže iste podatke za različite aplikacije u različitim formatima, tako se podaci dupliraju.
- **Nemogućnost sinhronizovanog ažuriranja podataka→**Npr adresa kupca se promeni u jednom fajlu, ali ne i u drugom
- **Zavisnost programskog koda od organizacije fajlova→**

BAZA PODATAKA

- **Cilj DBMS-a** je da razdvoji detalje o načinu na koji se podaci fizički pamte od načina na koji se oni koriste u aplikativnim programima.

**ARHITEKTURA TRI ŠEME

- **Eksterna šema→** predstavlja način na koji se podaci koriste u aplikativnim programima
- **Konceptualna šema→** je logički model podataka i nezavisna je i od eksterne šeme i od detalaj o tome kako se podaci pamte.
- **Fizička organizacija podataka→** se nalazi u internoj šemi, kojom se definišu file-ovi za pamćenje podataka.

- ****DBMS ALATI→**Data definition language(DDL), Data manipulation language, ograničenja integriteta, upravljanje transakcijama, konkurentnost, zaštita.

- ****NEDOSTACI KORIŠĆENJA DBMS-A→**Potrebno je mnogo novca za investiranje u velike DBMS sisteme, zapošljavanje osoblja z upravljanje DBMS-a skupo košta, postupci konverzije podataka iz baze podataka u formate koje zahtevaju aplikativni programi mogu biti složeni.

- **DDL(Data Definition Language) →** je „sintaksa“ slična kompjuterskom programskom jeziku i koristi se za specifikaciju podataka koji se drže u sistemu za upravljanje bazom podataka i za strukture koje se koriste za njihovo čuvanje.

- **PROCESOR UPITA(Query processing)→**Korisnik ili neki program pokreće neki upit,koristeći jezik za upravljanje podacima(**DML)**.Ove komande ne menjaju strukturu podataka baze podataka, već samo sadržaj unutar same baze.

- **DML-**Data manipulation language-je kompjuterski programski jezik koji se koristi za unos, brisanje i modifiranje(apdejtovanje) podataka u bazi.

*****TRANSAKCIJA***(BITNO)**

- U praksi je uobičajeno da se nekoliko operacija nad bazom podataka grupiše u transakciju.

- predstavlja jedinicu posla koja mora biti izvršena atomično i potpuno izolovana od drugih transakcija, a DBMS obezbeđuje da će rezultati izvršene transakcije biti sačuvani.

***Uloga procesora transakcija je da vrši sledeće operacije**:

1. **Logovanje→** Kako bi se obezbedila dugotrajnost baze podataka svaka promena koja je nastala se beleži u poseban fajl na disku.Ukoliko dođe do neke greške, uz pomoć log fajla je moguće izvršiti povratak baze podataka u konzistentno stanje.Menadžer logovanja uglavnom vrši logovanje na mestima gde najčešće dolazi do pada sistema, a to je u baferima.

2. **Kontrola paralelnog rada→**Transakcija se mora izvršiti izolovano, ali će se u mnogim sistemima više transakcija izvršavati istovremeno.Menadžer paralelnog rada mora da obezbedi da se svaka akcija iz više transakcija izvrši do kraja, samo jedna u istom trenutku nad jednim entitetom i ovo se uglavnom postiže zaključavanjem nekih delova baze u toku njenog izvršenja, odnosno zabranjivanjem drugih transakcija.
3. **Blokada transakcija→**Transakcije se nadmeću međusobno za resurse kako bi bile izvršene, može se doći u situaciju da svakoj treba neki resurs koji druga transakcija trenutno drži zkaljučanim odnosno ostali su zaključani neki delovi baze, pa na red dolaze druge transakcije kojima će možda trebati resursi koji nisu trenutno zaključani od strane drugih transakcija.

PROCESOR UPITA(To je deo DBMS-a koji najviše utiče na perfomanse sistema) se sastoji od dve komponente:

- **Kompajler upita→** prevodi upit u internu formu nazvanu „plan upita“, odnosno niz akcija koje treba izvršiti nad podacima
- **Execution engine→** je odgovoran za izvršenje svakog koraka definisanim planom upita.

RDBMS(Relaciona baza podataka)

- Može se definisati kao celokupnost međusobno povezanih podataka uskladištenih u spoljnoj memoriji računara uz minimalnu redudantnost.Programi koji koriste te podatke ne zavise od načina njihovog pamćenja.

- **Relaciona** baza podataka se sastoji od podataka koji su smešteni u jednu ili više tabela.Zove se relaciona, jer postoje relacije između redova tih tabela.**ZNAČI PAMTE SE TABELE I PODACI I RELACIJE MEĐU PODACIMA.**

- Svaka tabela se sastoji od redova podataka, a svaki red sadrži vrednost atributa(kolone).Svaka kolona sadrži vrednost podatka istog tipa atributa.

SQL- je jezik koji se koristi za upravljanje podacima u RDBMS.(da bi se dobile različite informacije).

METAPODACI→ podaci o podacima→ to su tabele podataka koje opisuj te korisnike podataka.

Aplikacije→ računarski programi sa kojima korisnici imaju direktnu interakciju.App prihvata podatke od korisnika, obrađuje ih prema zahtevima i koristi SQL za transfer podataka prema i od baze podataka.

OBJEKTNO ORIJENTISANA BAZA PODATAKA

- OO je pristup u kome se neki sistem organizuje kao kolekcija međusobno povezanih objekata koji ostvaruju postavljene ciljeve.

- **ODBMS ili OODBMS-** sistemi za upravljanje OO bazama podataka imaju cilj da omoguće prirodniju manipulaciju podacima u aplikacijama koje se razvijaju nad bazom.
- Koncept objektnog modela se sastoji od: **Objekta i Literala.**

Literal→ je osnovna vrednost, podatak koji se koristi u modelu.Npr brojevi i karakteri „atomskih" literala, a datum je primer složenog literala.

Objekat→Entitet koji je sposoban da čuva svoja stanja.Objekat je korisnički definisani, kompleksni tip podataka koji ima svoju strukturu i stanje(Atribut i svojstva), kao i ponašanje(metode i operacije).

- Koncept primarnog ključa u OO modelu ne postoji, već ODBMS za svaki objekat pamti jedinstveni i neprimetljivi identifikator.

Komponente ODBMS→objektni modeli, objektni jezici, objektni upitni jezici, programski jezici.

***NAČINI ZA PROJEKTOVANJE BAZA PODATAKA:**

- Dizajniranje iz postojećih podataka
- Razvoj novih baza podataka
- Redizajniranje baza podataka

IT350 – LEKCIJA 2

Osnove relacionog modela-Relaciona algebra

***DEKARTOV PROIZVOD** predstavlja skup svih uređenih parova dva skupa.

-U- Univerzalni skup

-Množina svih podskupova skupa U se označava sa P(U) i ona se zove **partitivni skup skupa U.**

(a,b) uređeni par

- **Relacija između skupova→**nastaje kada se između elemenata dva skupa **E i F** uspostavi odnos pri kojem jednom elementu iz E odgovara više elemenata skupa F.

- **Codd-ova definicija relacije→** neka su dati skupovi D1,D2,D3..Dn, R je relacija nad ovih n skupova(n>0) ako je to skup n-torki takav da za svaku n-torku vredi da je prvi element n-torke iz D1, drugi iz D2,.. n-ti iz Dn.

- Skup D1 se naziva **domen relacije R.**

- **Domen** je skup elemenata sličnog tipa,npr, skup svih prezimena radnika preduzeća.

- **Relacija R** je skup n-torki pri čemu je redosled domena u Dekartovom proizvodu važan.Imenujući funkciju koja i-tom članu n-torke pridružuje datu vrednost „di“ imenom A1 on postaje nebitan, a n-torka(d1,d2,...dn) relacije R zamenjuje sa n-torkom parova <A1:d1,A2:d2..An:dn) (**NE VERUJEM DA ćE ovo neko da zapamti,čisto sam napisao)**.

  A1 je **atribut** relacije, a broj n-torki je **kardinalnibroj** relacije.

  Zapis relacije : naziv relacije(atr1,atr2..atrn)

  Kadar(#kadra, prezime,ime,starost) a sastoji se od n-torke : <# kadra:111, prezime:patka:ime patka starost:12113god>

<u>OGRANIČENJA RELACIONOG MODELA(Postoji 3)</u>

Svojstvo 1 : Svi elementi skupa su različiti

- To znači da bilo koje dve n-torke, odnosno bilo koja dva reda tabele nisu jednaka;iz ovog proizilazi definicija ključa relacije.

- **Ključ relacije→** je onaj podskup atributa čije vrednosti jedinstveno identifikuju n-torke relacije.

- **Kandidati za ključ→**To je skup koji čine svi ključevi relacije.

- **Strani ključ→**Ako neki skup atributa u posmatranoj relaciji nije ključ ali je ključ u nekoj drugoj tabeli, onda se naziva stranim ključem.

*****Da bi neki skup atributa relacije bio kandidat za ključ mora zadovoljiti dva uslova*** :**

1. **Uslov jedinstvenosti**: Vrednost ključa svake n-torke relacije jedinstveno određuje n-torku(ne postoje dva reda u tabeli takva dai maju sve iste vrednosti svih adtributa koje čine ključ).

2. **Uslov ne redudantonsti**: Ne postoji ni jedan atribut kao deo ključa koji se može izostaviti iz ključa, a da se pri tome uslov jedinstvenosti ne gubi tj. ključ je unija minimalnog broja atributa.

<u>PRAVILA INTEGRITETA BP</u>

- **Integritet entiteta→**Neka je atribut A deo primarnog ključa relacije R.Tada atribut A ne sme poprimiti null vrednost.To znači da ne postoji n-torka relacije R takva da je vrednost atributa A te n-torke jednaka null vrednosti.
- **Referencijalni integritet→**Neka postoji domen D i relacija S sa prostim primarnim ključem A definisanim nad D.Atribut A relacije R, koji postoji kao primarni ključ u relaciji S, naziva se spoljni ključ.

Svojstvo 2 : poredak elemenata skupa je nebitan

- Poredak n-torki relacije može biti proizvoljan u bazi podataka.Ovo svojstvo ne zahteva specifikaciju ograničenja.

Svojstvo 3 : Relacija je podskup dekartovog proizvoda skupova

Preslikavanje dva skupa može biti sledeće:

- Jednom elementu skupa A odgovara jedan ili nijedan element skupa B
- Jednom elementu skupa A odgovara nijedan,jedan ili više elemenata skupa B

<u>KARDINALNOST PRESLIKAVANJA→</u> je broj elemenata skupa B koji se mogu pridružiti elementu skupa A.

PRIMER RELACIJE „MOVIES“

title	*year*	*length*	*genre*
Gone With the Wind	1939	231	drama
Star Wars	1977	124	sciFi
Wayne's World	1992	95	comedy

Slika 4.1 Rlacija Movies

Atributi→ Kolone relacije „movies“

Šema relacije(ime relacije praćeno listom atributa koji se stavljaju u zagrade)→ predstavlja ime relacije i skup atributa relacije.

PRIMER- **Movies(title, year, length, genre)**

Relaciona šema baze podataka→je skup šema tih relacija baze podataka

Torke→Redovi relacija – torka ima po jednu komponentu za svaki atribut relacije.

NPR**. prva torka**sa slike ima 4 komponente : Gone With the Wind,1939,231,drama.

-Relacioni model zahteva da svaka komponenta bude atomična.

Domen→skup mogućih vrednosti koje mogu biti dodeljene komponentama bilo koje torke relacije.

Šema za movies : Movies(title:string, year:integer, length:integer,genre:string)

****KLJUČ→** je kombinacija jedne ili više kolona koje se koriste za identifikaciju pojedinačnih redova u tabeli.Postoji kandidat ključ,primarni ključ i surogat ključ

*****KANDIDAT KLJUČ→** je determinanta koja određuje sve druge kolone u relaciji.Jedinstveno identifikuju redove u relaciji.

*****SUROGAT KLJUČ→** je veštačka kolona koja se dodaje tabeli da bi služila kao primarni ključ.Surogat ključ se koristi kada je primarni ključ isuviše veliki ili kada se ne može definisati.

*****STRANI KLJUČ→** je kolona ili skup kolona koji su primarni ključ neke tabele koja nije tabela u kojoj se one pojavljuju.

RELACIONA ALGEBRA (BITNO)

Operacije tradicionalne relacione algebre se mogu razvrstati u četiri klase:

- **Podrazumevani set operacija – unija i razlika**
- **Operacije za izveštavanje – presek,selekcija,projekcija**
- **Operacije koje kombinuju zapise dve relacije – join**
- **Operacije preimenovanja**

PODRAZUMEVANI SET OPERACIJA NAD RELACIJAMA

- **Unija**(r U s) – unija skupova r i s predstavlja skup elemenata koji pripadaju skupu r, skupu s ili skupu r i skupu s.
- **razlika(r – s) -** predstavlja skup elemenata koji pripadaju skupu r, a ne pripadaju skupu s.

OPERACIJE AŽURIRANJA*

- upis,brisanje i menjanje n-torki u relaciji
- **Unija** – je operacija nad dve relacije koja omogućava dodavanje različitih n-torki jedne relacije drugoj relaciji, te omogućava upis u relaciju.Oznaka za operator je „U“.
- **Razlika –** je operacija nad dve relacije koja omogućava izbacivanje istih n-torki u obe relacije.
- **Ažuriranje-** je proces uzastopne primene prvo operatora razlike, pa zatim operatora unije.

Operacije za izveštavanje

- **PROJEKCIJA→** je operacija nad jednom relacijom koja omogućava izdvajanje skupa atributa jedne relacije u novu relaciju, i pri tom eleminiše duple n-torke. Primer: R1(a,b,c,d,e), relacija r2(a,d) je rezultat projekcije.

  R2(a,d) = r1[a,d](a,b,c,d,e)

- **SELEKCIJA→** je operacija nad jednom relacijom koja izdvajanje skupa n-torki jedne relacije u novu.Izdvajaju se samo one n-torke koje zadovoljavaju uslov uz operator selekcije. Uslov B>15

  R2(A,B,C) = R1[B > 15] (A , B, C)

- **PRESEK →**je operator koji za rezultat daje relaciju koja sadrži samo one n-torke koje se istovremeno nalaze u obe relacije.Oznake za operator je „ "∩"

OPERACIJE KOJE KOMBINUJU ZAPISE DVE RELACIJE

-Pored dekartovog proizvoda, za izveštavanje iz više relacija uveden je operator JOIN(kontrolisano spajanje). Neka „theta“ označava jedan od operatora poređenja: **jednako, nejednako, manje nego, više nego, manje nego ili jednako, veće nego ili jednako, veće**

- **THETA JOIN→** je operator spajanja dve relacije.Nova relacija ima sve atribute prethodnih relacija a-ntorke nastaju konkateniranjem n-torki relacije.

  R1(a,b,c) R2(d,f,g) THETA JOIN : R3(a,b,c,d,f,g)

- **PRIRODNI JOIN→** vrši uparivanje samo onih torki iz skupova R i S koji se slažu po bilo kojim atributima koji su zajednički za R i S.

KOMBINOVANJE OPERACIJA PRILIKOM FORMIRANJA UPITA

-Relaciona algebra dozvoljava formiranje izraza proizvoljne složenosti primenom operacija na rezultate izveštaja drugih operacija.

IT350 – LEKCIJA 3

NORMALIZACIJA I DENORMALIZACIJA

- **FUNKCIONALNA ZAVISNOST→** je srce procesa projektovanja baze podataka.To znači da kada vrednost jednog ili više atributa određuje vrednost drugih atributa.

Primer:

CenaKeksa = BrojKutija * 50 dinara → U ovom slučaju iz primera vidimo da cenaKeksa zavisi od brojaKutija. Može se reći **da je cenaKeksa funkcionalno zavisna od BrojaKutijašto se može napisati** kao:

- **BrojKutija→CenaKeksa** (BrojKutija određuje CenuKeksa)

- Varijabla sa leve strane, u ovom slučaju BrojKutija se naziva **determinanta.**

Cena = Količina * JediničnaCena

(Količina, JediničnaCena)→Cena

<u>NORMALIZACIJA</u>

- Normalizacija se koristi da bi se eleminisala redudantnost podataka i izbegla moguća korupcija baze podataka.
- „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“.

<u>KORUPCIJA BP</u>

- Može nastati prilikom brisanja,unosa i žuriranja podataka.,

StavkaBroj	Tip	CenaKoštanja	OpravkaBroj	OpravkaDatum	OpravkaCena
100	Presa	3500	2000	05.05.2007	375
200	Bušilica	4750	2100	05.07.2007	255
100	Presa	3500	2200	01.02.2007	178
300	Strug	27300	2300	10.02.2007	1785
100	Presa	3500	2400	12.03.2007	0
100	Presa	3500	2600	15.04.2007	275

Slika 2.1 Primer tabele nad kojom je moguća korupcija

- Primer za **anomaliju brisanja→**Ukoliko se izbriše slog čiji je OpravkaBroj 2100, izgubili bi se podaci ne samo o obavljenoj opravci već i o mašini nad kojom je rađena opravka.Brisanjem ovog reda, izgubili bi smo informacije o dve različite stvari:mašini i njenoj opravci.

- **Primer za anomaliju unosa**→Pretpostavimo da želimo da unesemo nove podatke o nekoj izvršenoj opravci.Da bi smo to odradili optrebno je da znamo broj opravke,datum i cene opravke već i podatke o samoj mašini-TIP i cenu koštanja što može biti problem jer oni koji rade na opravci obično te podatke ne znaju.
- **Primer za anomaliju ažuriranja**→Ukoliko menjamo broj opravke,datum i cene opravke, ne bi trebalo da dođe do problema. Ali ukoliko menjamo vrednosti koje se odnose na broj, tip i cenu koštanja mašine,može nastati problem.

Normalne forme

-Normalne forme služe da se eleminišu anomalije prouzrokovane postojanjem funkcionalne zavisnosti:To su 1NF,2NF,3NF,BCNF.

PRVA NORMALNA FORMA 1NF

-Relacija je u prvoj normalnoj forma ako svi njeni atributi imaju samo atomske(nedeljive) vrednosti.Npr ako je torka IME i u njoj je atribut Pera Perić, to nije forma 1NF. Biće jedan 1NF ukoliko podelimo na ime i prezime u kolonama.

DRUGA NORMALNA FORMA 2NF

- Ako je svaki atribut koji nije primarni ključ funkcionalno zavistan 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 zavistan od potpunog skupa atributa primarnih ključeva

Primer:

ZAPOSLENI(Zaposleni_ID,Ime, Odeljenje, Plata, Kurs, Datum_Zavrsetka)

Funkcionalne zavisnosti u ovoj relaciji:

Zaposleni_ID→Ime,Odeljenje,Plata

Zaposleni_ID,Kurs→Datum_Završetka

Primarni ključ u ovoj relaciji je sastavljen od zaposleni_ID i kursa, ostali atributi su funkcionalno zavisni samo od PK-a zaposleni_ID, a ne od Kursa.ZAPOSLENI ima redudansu što stvara teškoće tabele ažuriranja.

Da bi 2NF bila zadovoljena, potrebno je da se ZAPOSLENI dekomponuje u sledeće dve relacije:

ZAPOSLENI(Zaposleni_ID,Ime,Odeljenje,Plata) : zadovoljen prvi uslov 2NF-a

Zaposleni_KURS(Zaposleni_ID,Kurs,Datum) : treći uslov 2NF-a

TREĆA NORMALNA FORMA 3NF

-Relacija je u 3NF ako je u drugoj normalnoj formi i ako su svi atributi koji nisu deo primarnog ključa međusobno **NEZAVISNI**.

JMBG	IME	PREZIME	MENTOR	MENTOR_TELEFON
16029851234	Marko	Marković	Miroslav Trajnović	063 556 777
19019873454	Petar	Petrović	Nebojša Zdravković	064 563 456
21019864567	Munir	Hundrović	Svetlana Cvetanović	062 222 333

Slika 5.1 Torke relacije STUDENTI

- Sa slike se vidi da nije u 3NF jer Mentor_telefon zavisi od atributa „MENTOR“ a ni jedan od njih nije ključ.

- Sledeće slike su slike kada je relacija u 3NF

MENTOR_ID	MENTOR_IME	MENTOR_TELEFON
1	Miroslav Trajnović	063 556 777
2	Nebojša Zdravković	064 563 456
3	Svetlana Cvetanović	062 222 333

Slika 5.2 Torke relacije MENTORI

JMBG	IME	PREZIME	MENTOR_ID
16029851234	Marko	Marković	1
19019873454	Petar	Petrović	2
21019864567	Munir	Hundrović	3

Slika 5.3 Novi oblik relacije STUDENTI

BOYCE-CODD normalna forma (BCNF)

- Relacija je u BNCF kada svaka determinanta postane kandidat za ključ.Dovođenjem relacije na BCNF se u potpunosti eleminiše funkcionalna zavisnost.
- Relacija ne ispunjava BCNF ukoliko u njoj postoje funkcionalne zavisnosti kod kojih determinanta nije kandidat za ključ.

Strategija:

1. Identifikovati svaku funkcionalnu zavisnost
2. Identifikovati svaki kandidat ključ
3. Ako postoji funkcionalna zavisnost koja ima determinantu koja nije kandidat ključ:
   A) Kolone te funkcionalne zavisnosti treba prebaciti u novu relaciju
   B)Determinantu te funkcionalne zavisnosti treba proglasiti primarnim ključem nove relacije
   C)Kopiju determinante treba ostaviti kao sekundarni ključ originalne relacije
   D)Kreirati ograničenje referencijalnog integriteta između originalne i nove relacije
4. Ponavljati korak 3 sve dok je svaka determinanta svake relacije kandidat ključ

POGLEDATI PRIMER(Lekcija 3, Strana 18,19,20,21)

ČETVRTA NORMALNA FORMA 4NF

- Javlja se kada jedna determinanta određuje skup vrednosti a ne samo jednu vrednost.

Primer takvih zavisnosti dat je na slici 1.

Zaposleni Jones ima dva stepena stručne spreme BS i AA dok Greene ima PhD, MS i BS što se predstavlja kao:

Zaposleni →→ St.Sprema

Zaposleni →→ Sestra/Brat

Zaposleni	St.Sprema
Jones	BS
Jones	AA
Greene	PhD
Greene	MS
Greene	BS
Jon	BS

Zaposleni	Sestra/Brat
Jones	Fred
Jones	Sally
Jones	Frank
Greene	Nikki
Jon	Janathan
Jon	Eileen

Slika 7.1 Tabele u kojima postoji zavisnost od više vrednosti

Suprotno funkcionalnoj zavisnosti, determinanta zavisnosti od više vrednosti ne može nikada biti primarni ključ. U oba prethodna primera primarni ključ se sastoji od kompozicije dve kolone u svakoj u tabela.

Primarni ključ tabele ZAPOSLENI ST.SPREMA je složen: (Zaposleni, St.Sprema).

Naredna slika predstavlja relaciju u 4NF formu, spajanjem ove dve tabele.Zavisnost od više vrednosti se može eleminisati ako se ona izoluje u posebnu tabelu.Rezultat je uvek tabela sa dve kolone koje zajedno čine primarni ključ.

-Tabela je u 4NF ako je u BCNF i ako nema zavisnost od više vrednosti.

Zaposleni	St.Sprema	Sestra/Brat
Jones	AA	Fred
Jones	BS	Fred
Jones	AA	Sally
Jones	BS	Sally
Jones	AA	Frank
Jones	BS	Frank
Greene	PhD	Nikki
Greene	MS	Nikki
Greene	BS	Nikki
Jon	BS	Jonathan
Jon	BS	Eileen

DENORMALIZACIJA

- je proces suprotan normalizaciji.Poboljšava prefomanse.
-
- Karakteristike procesa denormalizacije je da se :
    - Može vršiti sa tabelama ili kolonama
    - Pretpostavlja prethodnu normalizaciju
    - Zahteva temeljno znanje o načinu korišćenja podataka

- Nedostaci denormalizacije:
    - Obično ubrzava dobijanje ali može usporiti ažuriranje podataka
    - Uvek je specifična za svaku aplikaciju i treba je ponovo proceniti ako se aplikacija izmeni
    - Može uvećati tabele
    - U nekim slučajevima pojednostavljuje kodiranje u drugim ga ugrožava

DENORMALIZACIJA SPAJANJEM TABELA:

→Primenjuje se ako se upitima sa dve ili više tabela redovno spajaju,a cena njihovog spajanja je velika

DENORMALIZACIJA DUPLICIRANJEM/DELJENJEM:

→Dupliciranje se koristi kada treba napraviti tabelu za izveštavanje ili izvršiti dupliciranje tabela.Deljenje tabela se od strane različitih korisnika pristupa samo pojedinim delovima tabele.

Tabele za izveštavanje→Ovaj tip denormalizacije se koristi u slučajevima kad nije moguće razviti izveštaj za korisnika uz pomoć SQL-a.Ako neki kritično ili često korišćeni izveštaj ovakve prirode treba da se urade u realnom vremenu može se koristiti tabela koja predstavlaj taj izveštaj.Podaci za ovakav izveštaj se kreiraju najčšeće aplikativnim programom ili SQL-om u batch okruženju a onda se učitaju u tabelu kao sekvencijalni podaci.

Dupliciranje tabela→Koristi se ako je grupi korisnika redovno potreban samo podskup podataka iz neke tabele onda se kritilna tabela može ponoviti u podskup za tu grupu.

Deljenje tabela:

- **Horizontalno**: Kod ove podele tabele slogovi su klasifikovani u grupe u zavisnosti od načina grupisanja ključa tabele.
- **Vertikalno**:Vertikalno podeljena tabela odvaja samo neke kolone u nove tabele, odnosno jedan skup kolona je u jednoj, a drugi u drugoj tabeli.Primarni ključ se nalazi u obe nove tabele.

DENORMALIZACIJA DELJENJEM KOLONA i DENORMALIZACIJA DODAVANJEM KOLONA

IT350 – LEKCIJA 4

SQL:DDL i DML

- SQL-Standardni jezik za pristup bazama podataka

- -Karakteristike SQL-a →jednostavnost i jednoobraznost pri korišćenju, mogućnost interaktivnog i klasičnog programiranja, neproceduralan.

- DDL-Naredbe za definisanje podataka

- DML-naredbe za manipulaciju podacima

- **DLL naredbe :**CREATE DATABASE, CREATE TABLE, CREATE INDEX, ALTER TABLE, DROP TABLE

- **Primeri:**
    - **CREATE DATABASE IF NOT EXIST ImeBaze;**
    - **SHOW DATABASE –** Da bi prikazali sve baze podataka
    - **USE imeBaze –** Naredba da bismo pristupili novoj bazi
    - **DROP DATABASE imeBaze –** brisanje baze podataka
    - **CreateTable** – naredba koja se koristi za kreiranje nove tabele,definisanje kolona,ograničenja koja se odnose na kolone i kreiranje relacija.
    - CreateTable naredba ima pet tipova ograničenja : PrimaryKey,UNIQUE, NULL/NOT NULL,FOREIGN KEY,CHECK
    - **CHECK-** služi za definisanje ograničenja koji se mogu primeniti na vrednost podataka
    - **CREATE SEQUENCE-** ovom naredbom se generiše sekvencijalna serija jedinstvenih brojeva za potrebe surogat ključa (CREATE SEQUENCE CustID Increment by 1 start with 1000);
    - **CREATE INDEX –** indeksi se kreiraju kako bi se naglasila jedinstvenost kolona, olakšalo sortiranje i kako bi se omogućilo brže pretraživanje po vrednostima nekih kolona(**CREATE INDEX KupacimeIdx ON KUPAC(ime);**

- **DDL ALTER TABLE→** Koristi se ukoliko želimo da promenimo strukturu postojeće tabele

  ALTER TABLE KUPAC ADD NovaKolona Char(5) NULL;

- ***Naredba TRUNCATE→** Ukoliko hoćemo da obrišemo sve podatke koje tabela sadrži, ali ne i da obrišemo i definiciju tabele.(**TRUNCATE TABLE ImeTabele).**

- **DML naredbe :** INSERT, UPDATE, DELETE, SELECT

- **INSERT →**INSERT ime tabele(imena kolona u koje se unose podaci) VALUES(lista_podataka)

- **UPDATE→** Za menjanje sadržaja tabele (**UPDATE tabela SET kolona1=izraz1,[kolona2=izraz2][WHERE kriterijum selekcije];**

- **DELETE→** naredba za brisanje sadržaja dela ili cele tabele
  DELETE FROM imeTabele WHERE kriterijumi

- **SELECT→**Naredba za izveštavanje iz relacione baze podataka.
  SELECT<lista atributa> FROM imeTabele WHERE kriterijumi;

IT350 – LEKCIJA 5

SQL NAD JEDNOM TABELOM

- Naredbom **SELECT**se može dobiti modifikovan sadržaj jedne ili više tabela U. i to primenom aritmetičkih funkcija nad kolonama numeričkog tipa ili primenom odgovarajućih funkcija nad nizom karaktera.

- Modifikovan sadržaj tabele se može dobiti i primenom operacije restrikcije tj. Upotrebom klauzule **WHERE**uz koju se mogu koristiti različiti operandi koji se odnose na numeričke I nenumeričke sadržaje vrednosti atributa.

- Naročito je važna upotreba sumarnih funkcijakoje se primenjuju najčešće uz korišćenje grupne funkcije **GROUPBY**. Klauzula **GROUPBY**se primenjuje za dobijanje srednjih, minimalnih, maksimalnih, sumarnih vrednosti na nivou grupa podataka u tabeli.

- Napomenimo i da **WHERE**klauzulu ne koristimo isključivo u sprezi sa **SELECT**klauzulom, već se koristi i sa **UPDATE**, **DELETE**, itd. klauzulama.

- Operatore poređenja koje možemo koristi sa klauzulom WHERE su:

  = (jednako)
  != (nije jednako) -
  <> (nije jednako) - drugi način predstavljanja !=
  > (veće od)
  < (manje od)
  >= (veće od ili jednako)
  <= (manje ili jednako)
  !> (nije veće od)
  !< (nije manje od)

- Logički operatori uz **klauzule WHERE**su :

  1. **AND** - omogućava višestruke uslove u SQL upitu, ali svi uslovi moraju biti ispunjeni da
  2. **OR** - koristi se za kombinvanje više uslova u upitu, tako da mora biti ispunjen samo jedan uslov
  3. **IN** - koristi se za upoređivanje vrednosti sa listom vrednosti koje su navedene u uslovu
  4. **LIKE** - za upoređivanje sličnih vrednosti koristeći wildcard operatore
  5. **NOT** - negacija, okreće značenje logičkih opratora sa koji se koristi
  6. **BETWEEN** - koristi se za pretraživanje vrednosti u nekom opsegu, zadavajući granice opsega
  7. **ALL** - koristi se za poređenje sa svim vrendostima u drugom setu.
  8. **ANY** - koristi se za upoređivanje vrednosti sa svako vrednošću u listi prema navedenom uslovu.
  9. **EXISTS** - koristi se za pretragu određenog reda u tabeli koji ispunjava određeni kriterijum,

10. **ISNULL** - operator koji poredi vrednost is tabele sa NULL vrednošću, pa vraća true ili false ako je (nije) ispunjen taj uslov
11. **UNIQUE** - oprator pretražuje svaki red navedene table za jedinstvenim vrednostima - tj da nema duplikata.

- Klauzula **GROUPBY**omogućava dobijanje sumarne informacije za svaku različitu vrednost kolone ili grupe kolona po kojoj se vrši grupisanje. Klauzula GROUP BY se gotovo uvek koristi uz neku funkciju za dobijanje sumarnih informcija (MIN, MAX, AVG, COUNT, SUM).

- Klauzula **HAVING**određuje kriterijume za selekciju grupa pošto su grupe već formirane sa **GROUPBY** klauzulom. Kada se koriste agregatne fukncije (**MIN**, **MAX**, **AVG**..), dobijene vrednosti ne mogu da se ispituju sa **WHERE** klauzulom pa se za njih koristi **HAVING**.

- Rezultujuću tabeluje moguće sortirati po jednom ili više atributa u rastućem ili opadajućem redosledu. Za specifikaciju rastućeg redosleda koristi se klauzula **ASC**, a za specifikaciju opadajućeg redosleda klauzula **DESC**.

- **NULL** vrednost može označavati vrednosti koje su nedefinisane. Između NULL vrednosti i vrednosti nula postoji značajna semantička razlika . Bez obzira o kom tipu NULL vrednost se radi, određene kolone se na NULL vrednost mogutestirati pomoću dve specijalne klauzule: IS NULL ili IS NOT NULL za šta se koriste operatoripoređenja.

- Da bi se izračunavanje ipak omogućilo, koristi se **NVL funkcija**(u Oracle ili ekvivalenti ove fnkcije) koja privremeno menja NULL vrednost sa vrednošću za koju se sami odlučimo, tj. vrednošću koja je neutralna u odnosu na željenu operaciju.

```
SELECT IMEPREZIME, POSAO, LD,
LD+NVL(PREMIJA, 0)
FROM RADNIK
WHERE S_RJ = 30;
```

Izraz **LD+NVL(PREMIJA,0)** se izvršava se na sledeći način:
• ukoliko je PREMIJA nedefinisana, NULL vrednost se zamenjuje sa nulom i sabira se sa ličnim dohotkom (LD),
• inače se uzima konkretno definisana vrednost premije i sabira sa ličnim dohotkom.

IT350 – LEKCIJA 6

SQL: Naredba SELECT za rad saviše tabela; Kreiranje pogleda

- 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.

Postoje:

- **INNER JOIN**
- **LEFT JOIN**
- **RIGHT JOIN**
- **SELF JOIN**
- **FULL JOIN**

- **JOIN (INNER JOIN, EQUIJOIN, ...)**povezuje n-torke različitih tabela korišćenjem zajedničkih atributa, odnosno atributa definisanih nad istim domenima.

- **CARTESIAN JOIN (CROSS JOIN)**se izvršava ukoliko se u WHERE klauzuli izostavi uslov spajanja. Ako se primenjuje na dve tabele tada se svaka n-torka prve spaja sa svakom n-torkom druge tabele. Korišćenjem CARTESIAN JOIN naddve tabele se dobija DEKARTOV PROIZVOD tabela pri čemuse spaja svaka torka jedne tabele sa svakom torkom druge tabele.

- **LEFT (OUTER) JOIN**daje kompletni skup zapisa iz tabele A sa odgovarajućim podacima iz tabele B koji postoje i u tabeli A.

- **RIGHT (OUTER) JOIN** daje kompletni skup zapisa iz tabele B sa odgovarajućim podacima iz tabele A koji se nalaze i u tabeli b.

- **FULL (OUTER) JOIN**selektuje sve rekorde koje se nalaze i u levoj ili desnoj tabeli.

- Takođe se govorilo i o pogledima (VIEW) koji pretstavljaju virtualne tabele koje se primenjuju kada želimo da:

  • sakrijemo neke kolone ili redove tabele
  • prikažemo rezultate nekih sračunavanja
  • sakrijemo složenu SQL sintaksu
  • slojevito ugradimo neke funkcije
  • omogućimo nivo izolacije između podataka tabele i korisnikovog pogleda na podatke
  • nad istom tabelom različitim korisnicima dodelimo različite dozvole
  • nad istom tabelom različitim view-ovima dodelimo različite trigere

IT350 – LEKCIJA 7

Modeliranje podataka korišćenjem E/R dijagrama

- Tip entiteta **je kolekcija entiteta sa zajedničkim osbinama.**
- Entitet je jedno pojavljivanje tipa entiteta.

- Tip entiteta je osnovni koncept ER modela i predstavlja skup objekata u stvarnom svetu koji imaju ista svojstva. Entiteta je pojavljivanje (instanca)jednog tipa entiteta.

- Svaki tip entitet se identifikuje nazivom i listom svojstava. Baza podataka najčešće sadrži više različitih tipova entiteta.

- Atribut **predstavlja svojstvo entiteta ili tipa relacije.**

- Svaki atribut može imati jednu od skupa mogućih vrednosti koji se naziva domen atributa. Domen atributa predstavlja skup vrednosti koje se mogu dodeliti atributu.

- Atributi se mogu klasifikovati na:

    - proste i složene,
    - sa jednom vrednošću i sa više vrednosti,
    - izvedene

- **Prost atribut**se sastoji od jedne komponente koja ima nezavisnu egzistenciju. Ovakvi atributi se ne mogu dalje deliti. U literaturi se često ovakvi atributi označavaju ako atomski atributi. Primeri ovakvih atributa su Pol i Plata.

- **Složenatribut**se sastoji od više komponenti od kojih svaka ima nezavisnu egzistenciju.

- Ovakvi atributi se mogu dalje deliti na podkomponente.

- **Atributsajednomvrednošću**sadrži jednu vrednost za jedan entitet. Većina atributa u modelima predstavljaju atributi sa jednom vrednošću.

- **Atributsaviševrednosti**sadrži više vrednosti za jedan entitet.
  Na primer: atribut Telefon može sadržati više vrednosti (broj kućnog telefona i broj mobilnog telefona).

- **Izvedeniatribut**predstavlja vrednost koja je izvedena na osnovu vrednosti nekog povezanog atributa ili skupa njih, pri čemu oni ne moraju pripadati istom entitetu.

- **Identifikatorentiteta**predstavljaju atribute koji identifikuju neku instancu entiteta.

- **Relacija** (**veza**) predstavlja asocijacije koje se mogu uspostaviti između tipovaentiteta.

- **Stepenrelacije**predstavlja broj entiteta koji učestvuju u relaciji. Relacija čiji je stepen dva naziva se binarna relacija. Relacija čiji je stepen tri naziva se **ternarna relacija**.

KARDINALNOSTI

- **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. U E/R modelu se relacije klasifikuju prema svojoj kardinalnosti.

- **Maksimalnakardinalnost**predstavlja maksimali broj instanci entiteta koje učestvuju u relaciji.

- **Minimalnakardinalnost**predstavlja minimalni broj instanci entiteta koje učestvuju u relaciji.

- **Maksimalnakardinalnost** može biti: jedan-prema-jedan, jedan-premaviše i više-prema-više. U 1:1 relaciji, instanca entiteta jednog tipa se
odnosi na bar jednu instancu entiteta drugog tipa.

- Tri najčešće maksimalne kardinalnosti su tipa:

  • jedan-prema-jedan
  • jedan-prema-više
  • više-prema-više

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

- **Maksimalnakardinalnosttipajedan-prema-više**: U ovom slučaju jedna instanca jednog tipa entiteta može biti u vezi sa više instanci drugog tipa entiteta

- **Maksimalnakardinalnosttipa više-prema-više**: U ovom slučaju jedna instanca jednog tipa entiteta može biti u vezi sa više instanci drugog tipa entiteta ali i jedna instanca drugog tipa entiteta može biti u vezi sa više instanci prvog tipa entiteta

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

- Ako je **minimalnakardinalnost** nula, učešće u relaciji je opciono a ako je jedan, tada bar jedna instanca entiteta mora učestvovati u relaciji.

- Relacijama se mogu zadati nazivi **uloga** (role names) da bi se prikazala namena koju svaki učesnik relacije ima.

- Relacija u kojoj jedan isti tip entiteta učestvuje više puta sa različitim ulogama naziva se **rekurzivnarelacija**. Relacija sa atributima se predstavlja kao asocijativni entitet.

- **Višestrukirelacije**se javljaju u slučaju kada dva entiteta grade više od jednog tipa relacije.

- **Aasocijativnientitet**je specijalan slučajrelacije sa maksimalnom kardinalnošću“više prema više” koji zahteva da neki novo kreirani tip entitetapamti informaciju o toj relaciji.

-

ID ZAVISNI ENTITETI

- **IDzavisnientiteti** su entiteti čiji identifikator uključuje identifikator drugog entiteta.

- **Slabientiteti**su entiteti čije postojanje zavisi od prisustva drugih entiteta. Svi ID zavisni entitet su slabi entiteti. Međutim, neki entiteti koji su slabi, ne moraju biti ID zavisni.

SPECIJALIZACIJA

- **Specijalizacija**je proces maksimiziranja razlika između članova entiteta identifikovanjem karakteristika koje ih razlikuju. Ovaj proces predstavlja prilaz odozgo-nadole pri definisanjunad entiteta i njegovih pod entiteta. Pod entitetse definišu na osnovu njihovih specifičnihkarakteristika u odnosu na nad entitet. Svakom pod entitetu se dodaju atributi kojipredstavljaju tu karakteristiku, a zatim se definiše veza između pod entiteta i nad entiteta.

GENERALIZACIJA

- **Generalizacija**je proces minimizacije razlika između entiteta identifikovanjem zajedničkih karakteristika. Ovaj proces predstavlja prilaz odozdo-nagore, a kao rezultat daje identifikaciju i generalizaciju nad entiteta na osnovu originalnih pod entiteta.

- Nad **entite** predstavlja entitet koji uključuje odvojene pod entitete. Pod entitet je entitet koji ima jedinstvenu ulogu i deo je nad entiteta.

ŠTA PRETSTAVLJAJU POSLOVNA PRAVILA?

- Specifikacije koje čuvaju integritet lokalnog modela podataka.Poslovna pravila su specifikacije koje čuvaju integritet lokalnog modela podataka. Četiriosnovna poslovna pravila se odnose na:

  • **Integritet entiteta** :Svaka instanca jednog tipa entiteta mora da ima jedinstveni identifikator koji nije null.

  • **Ograničenja referencijalnog integriteta:** Pravila se odnose na veze izmedju tipovaentiteta.

  • **Domene** :Ograničenja na validne vrednosti atributa

  • Poslovna pravila koja štite validnost vrednosti atributa

- Generalno, poslovna pravila se skupljaju za vreme utvrdjivanja zahteva i pamte se u CASE repozitorijumu gde se dokumentuju.

- **Integritetom** entiteta se definiše nemogućnost postajanja dva primerka entiteta u istom tipu entiteta koja imaju istu vrednost atributa koji čine identifikator.

- **Referencijalniintegritet**se definiše da vrednost sekundarnog ključa u jednoj tabeli mora odgovarati vrednosti primarnog ključa u drugoj tabela čijim je spuštanjem nastao taj sekundarni ključ.

- **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.

- **Trigeri**su pravila pod kojim seizvršavajuoperacijeza 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:

  • **Korisničkopravilo**:koncizna rečenica kojom se opisuje poslovno pravilo koje treba da bude izvršeno operacijom;
  • **Događaj**:operacija koja se inicira (insert, delete ili update);
  • **Imeentiteta**:kojem se pristupa ili se modifikuje;
  • **Uslov**:pod kojim se operacija izvršava;
  • **Akcija: koja se izvršava kada se operacija trigeruje.**

IT350 – LEKCIJA 9

Transformacija ER modela u fizički relacioni model baze podataka

LOGIČKI MODEL BAZE PODATAKA

- **Logičko** projektovanje relacionih baza podataka je transformisanje konceptualnog modela u šemu koji se korišćenjem DDL unosi u SUBP. Logički model sesastoji od potpuno normalizovanih objekata čiji su svi atributi definisani. Pored toga, definisanje tip podataka i domen svih atributa, kao i atributi koji su kandidati za ključeve.

- **Primarniključ** treba da ima sledeće svojstva:
  - Vrednost primarnog ključa jedinstveno identifikuje svaku torku u relaciji.
  - Ključ mora da bude neredudantan; nema atributa u ključu koji može biti izbrisan a da se ne naruši njegova jedinstvena identifikacija

- **Surogatključ**je identifikator kojeg generiše DBMS, njegov redosled je jedinstvena u okviru jedne tabele i nikada se ne menja . On se dodeljuje u trenutku kreiranja reda utabeli a uništava se kada se red obriše.

- **Kandidatključevi**su alternativni identifikatori jedinstvenih redova u tabeli. Umesto termina kandidat ključ, često se koristi i termin alternativni ključ.

TRANSFORMACIJA ATRIBUTA

- Kada se vršitransformacija atributa tipova entiteta uatribute relacije treba obraditi pažnju na sledeće:
  - null status
  - tip podataka
  - default vrednosti
  - ograničenja podataka

- **Null status atributa:**Definiše da kolona ne mora imati vrednost.Dozvoljena null vrednost se definiše frazom NULL dok se nedozvoljena definiše sa NOT NULL. NULL ne znači da je kolona uvek NULL već znači da su dozvoljene NULL vrednosti.

- **Tip podataka atributa:** Svaki DBMS ima neke svoje specifične tipove podataka koje koristi.Međutim ako se želi postići nezavisnost od konkretnog DBMS mogu se specificirati generički tipovi podataka gde spadaju:

  - CHAR(n) – karakter string fiksne dužine
  - VARCHAR(n) - karakter string promenljive dužine

- DATE
- TIME
- MONEY
- INTEGER
- DECIMAL

- **Default vrednost atributa:**To je vrednost koju generiše DBMS priliko kreiranja redova tabele.Ta vrednost može biti konstanta ili rezultat neke funkcije kao što je sistemski datum ili vreme. Nekada se defaul vrednosti sračunavaju korišćenjem mnogo složenije logike, za šta se najčešće koriste trigeri.

Ograničenja na vrednosti podataka

Ograničenja podataka: Postoje nekoliko različitih ograničenjana vrednosti podataka:
Domen – kojim se vrednost kolona ograničava na određeni skup vrednosti.
Opseg vrednosti – kojim se vrednosti ograničavaju na određeni interval.

Na primer. KodZaposlenog može biti ograničen vrednostima ('novo zaposleni', 'stalno zaposlen', 'honorarno zaposlen' itd.)

- 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.

- Intra relaciono ograničenje – ograničava vrednost kolone u odnosu na druge kolone iste tabele.

Na primer: datum provere može biti najmanje 3 meseca posle datuma zaposlenja.

TRANSFORMACIJA VEZA MAKSIMALNEKARDINALNOSTI 1:1

- Transformacija veza maksimalne kardinalnosti 1:1 vrši se dodavanjem primarnog ključa relacije A kao stranog ključa u B ili obrnuto.

TRANSFORMACIJA VEZA MAKSIMALNE KARDINALNOSTI 1:M

- Vrši se spuštanjem atributa primarnog ključa relacije koji je na strain veze jedan, kao stranog ključa u relaciju koja je na strani veze više.

TRANSFORMACIJA VEZA MAKSIMALNE KARDINALNOSTI N:M

- Za takvu vezu se kreira posebna tabela. Primarni ključ te tabele je složeni ključ koji se sastoji od primarnih ključeva svake od dve tabele koje učestvuju u vezi.

TRANSFORMACIJA REKURZIVNE RELACIJE

- Transformacija rekurzivne relacije kardinalnosti jedan prema više se vrši kao i u slučaju drugih tipova 1:M relacija, tako što se tip entiteta nad kojim postoji rekurzivna relacijamodelira kao relacija. Njen primarni ključ je isti kao i idnetifikator tipa entiteta. Tada se relaciji dodaje rekurzivni strani ključ koji se referencira na vrednost primarnog ključa iste relacije.

TRANSFORMACIJA POD ENTITETA I NJIHOVIH NAD ENTITETA

Transformacija pod entiteta i njihovih nad entiteta se može se vršiti na tri načina:

1. Korišćenje E/R stila : 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. Korišćenje objektno orijentisanog pristupa: entiteti se tretiraju kao objekata koji pripadaju jednoj klasi : za svako moguće pod stablo koje se nalazi ispod korena, kreira se jedna relacija čija šema uključuje sve atribute svih tipova entiteta u pod stablu.

3. Korišćenjem null vrednosti: 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.

OBJEKTNO ORIJENTISANI PRSTUP

- Kreira se jedna relacija za entitete u svakom pod stablu. Šema te relacije ima sve attribute bilo kog tipa entiteta u pod stablu

TRANSFORMACIJA KORIŠĆENJEM NULL VREDNOSTI

- Hijerarhiju tipova entiteta možemo predstaviti kao jednu jedinstvenu relaciju. Ta relacija ima sve atribute koji pripadaju bilo kom tipu entiteta u hijerarhiji

KORACI U IMPLEMENTACIJI FIZIČKOG MODELA

- **Transformacijarelacija u tabele**. Fizički pandam relacijama, odnosno entitetima je tabela. 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.

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

- **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.

  Pored tipa podataka, ovde je potrebno definisati i ograničenja vrednosti podataka. Na primer,

atribut BrojRadnihSatiDnevno se može ograničiti na vrednosti između 0 i 12, jer su takvi poslovni zahtevi. Ovakva ograničenja znatno smanjuju mogućnost unosa pogrešnih vrednosti atributa. Takođe treba definisati da li atribut može imati Null (nema vrednosti) vrednost, ili neku podrazumevajuću vrednost koja se automatski upisuje kao vrednost atributa prilikom kreiranja novog zapisa.

- **Specifikacija primarnih ključeva.**Prilikom izrade logičkog modela primarni ključevi su definisani i treba ih preslikati u fizički model. U slučajevima kada to iz nekih razloga nijemoguće treba izabrati surogat.

- **Specifikacija redosleda kolona.** Sa aspekta tačnosti izvršenja operacija redosled kolona u tabelama je irelevantan. Rezultati upita će biti isti bez obzira da li je neka kolona treća ili osma po redu. Međutim, redosled kolona može bitno da utiče na brzinu izvršavanja operacija, pa je u nekim slučajevima potrebno promeniti redosled koji nudi logički model podataka.

- **Definisanje referencijalnih ograničenja za sve veze.**Da bi se definisala referencijalna ograničenja -potrebno je definisati primarni ključ u jednoj tabeli i strani ključ u zavisnoj tabeli.

  Referencijalna ograničenja povezuju primarni ključ za strani. Za svako referencijalno ograničenje je potrebno definisati set pravila koja određuju status kolone sa stranim ključem prilikom insertovanja i ažuriranja, kao i status zavisne vrste kada se primarni ključ briše.
  Na primer, kada se briše primarni ključ koji referencira na postojeći strani ključ, ovo pravilo definiše da li će SUBP izbeći brisanje primarnog ključa ili će obrisati i primarni i strani ključ, ili će se kao vrednost stranog ključa upisati Null vrednost. Ovakvim pravilima se garantuje referencijalni integritet baze podataka.

- **Definisanje prostora tabela (tablespace).** Iako su podaci relacionog modela fizički organizovani u tabele, odgovarajuće datoteke u kojima su smešteni podaci nisu jednostavno preslikane kolone i vrste. Tabele se preslikavaju u fizičke strukture koje se nazivaju proctor tabela (tablespace). Baza podataka se sastoji iz jednog ili više prostora tabela. Svaki proctor tabela može da sadrži jednu ili više tabela. Pored planiranja prostora tabela, potrebno je proceniti potreban fizički prostor na spoljnim memorijama potrebnim za čuvanje podataka i indeksa. Takođe je potrebno definisati potrebu za kompresijom podataka metod koji će se koristiti.

IT350 – LEKCIJA 10

Administracija baza podataka

ŠTA ZNAČI OPORAVAK BAZE PODATAKA?

- Povratak baze podataka u stanje pre softverskog ili hardverskog otkaza sistema zbog grešaka u operativnom sistemu, greške u programiranju, greške u samom SUBP-u i dr.

- Proces oporavka izvodi se globalno izvodi u tri koraka:

  • periodično kopiranje (save) baze podataka na eksternu memoriju
  • zapisivanje transakcija (promena) nad bazom podataka u žurnal, tzv. log file. Transakcije
  se u log file moraju 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 ne izvršenoj transakciji. Ako bi se transakcijom prvo ažurirala baza podataka a zatim ona upisivala u log file, postojala bi 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.
  • oporavak baze podataka koji se može vršiti na više načina

OPORAVAK BAZA PODATAKA KROZ PONOVNU OBRADU

- Podrazumeva periodično pravljenje kopije baze podataka; Kada dođe do greške, vrši se restore baze i ponovo obrađuju sve transakcije od trenutka pravljenja save-a do trenutka pada sistema.

OPORAVAK BAZE PODATAKA KROZ ROLLBACK ILI ROLLFORWARD

- **Rollforward–** izvrši se restore baze podataka na osnovu napravljenog save-a i vrate sadržaji promenjenih slogova baze podataka, koji su od trenutka pravljenja save-a do trenutka otkaza sačuvani u log file-u.

- **Rollback–** baza podataka se ostavlja u zatečenom stanju i vrate se nepromenjeni sadržaji slogova, koji su od trenutka pravljenja save-a do trenutka otkaza sačuvani u log file-u.

BEFORE IMAGE I AFTER IMAGE SLOGOVI

- **Dabisetransakcijeponištile(undo),**log file mora da sadrži kopiju svakog sloga baze podataka pre nego što je on promenjen. Takvi slogovi se zovu **beforeimage**. Transakcije se poništavaju primenom before image-a svih promena nad bazom podataka.

- **Dabisetransakcijevratile** (**redo**), log file mora da sadrži kopije svih slogova baze podataka pošto su nad njima izvršene promene. Ti slogovi se zovu **afterimage**. Transakcije se vraćaju primenom after image-a svih promena nad bazom podataka.

CHECKPOINT

- **Checkpoint**je tačka sihronizacije između baze podataka i transakcija log file-a. Prilikom izvršenja checkpoint-a, DBMS odbacije novi zahtev, završava obradu zaostalih zahteva i njihove bafere upisuje na disk.

KADA SE VRŠI OPORAVAK OD PADA SISTEMA

- U slučaju pada sistema, sadržaj unutrašnje memorije je izgubljen. Zato se, po ponovnom startovanju sistema, za oporavak koriste podaci iz log datoteke da bi se poništili efektitransakcija koje su se obrađivale u trenutku pada sistema. Ove transakcije se moguidentifikovati čitanjem sistemskog loga unazad, kao transakcije za koje postoji BEGINTRANSACTION slog ali ne postoji COMMIT slog.

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.

PLANIRANO I NEPLANIRANO IZVRŠENJE TRANSAKCIJE

- Do planiranog završetka dolazi izvršenjem COMMIT operacije kojom se uspešno kompletira transakcija, ili eksplicitne ROLLBACK operacije, koja se izvršava kada dođe do greške za koju postoji programska provera (tada se radnje transakcije poništavaju a program nastavlja saradom, izvršenjem sledeće transakcije).

- Neplanirani završetak izvršenja transakcije događa se kada dođe do greške za koju ne postoji programska provera; tada se izvršava implicitna (sistemska) ROLLBACK operacija, radnje transakcije se poništavaju a program prekida sa radom.

- Izvršavanjem operacije COMMIT:

  • efekti svih ažuriranja transakcije postaju trajni, tj. više se ne mogu poništiti procedurom
  oporavka.
  • u log datoteku se upisuje odgovarajući slog o kompletiranju transakcije (COMMIT slog), a
  svi katanci koje je transakcija držala nad objektima – oslobađaju se.

KAKO SE OMOGUĆAVA OPORAVAK OD PADA TRANSAKCIJE?

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

ZAŠTO SE PREDUZIMAJU MERE KONTROLE KONKURETNOSTI ?

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

ACID SVOJSTVA TRANSAKCIJE

- **Atomičnost, konzistentnost, izolacija, trajnost.**

- Transakcija se karakteriše sledećim važnim svojstvima (poznatim kao **ACID svojstva**):
  • atomičnost (Atomicity): transakciju treba izvršiti u celosti ili je uopšte ne treba izvršiti (ni jednu njenu radnju);
  • konzistentnost (Consistency): transakcija prevodi jedno konzistentno stanje baze u drugo konzistentno stanje baze;
  • izolacija (Isolation): efekti izvršenja jedne transakcije su nepoznati drugim transakcijama sve dok se ona uspešno ne kompletira;
  • trajnost (Durability): svi efekti uspešno kompletirane transakcije su trajni, tj. mogu se poništiti samo drugom transakcijom.

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

KADA T1 I T2 MOGU BAZU DOVESTI U NEKONZISTENTNO STANJE?

- Ako se jedna transakcija kompletno izvrši između prve i druge radnje druge transakcije. (lekcija 10 strana 14)

KADA DOLAZI DO PONIŠTAVANJA AŽURIRANJA?

- Npr. kada 1. transakcija ažurira slog, a zatim 2. pročita slog i završi svoje izvršavanje. Ako 1. transakcija poništi ažuriranje sloga, 2. transakcija je pročitala pogrešnu vrednost.

Rešenje problema konkurentnosti: zaključavanje

- **Zaključavanjeobjekta**(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). Pretpostavimo da u jednostavnom modelu postoji samo jedna vrsta katanca, i da pri izvršenju skupa transakcija nijedna transakcija ne može zaključati već zaključani objekat. U realnim modelima transakcija postoji više vrsta katanaca, od kojih su neki deljivi a neki ekskluzivni; pri tom više transakcija može da postavi deljivi katanac na jedan objekat, ali najviše jedna može da postavi ekskluzivni katanac.

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

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

ZAKLJUČAVANJE U SLUČAJU DVOFAZNIH TRANSAKCIJA

- Kod **dvofazne transakcije**postoje dve serijske faze:
    - faze zaključavanja objekata i
    - faze otključavanja objekata.

- U fazi zaključavanja nema nijedne radnje otključavanja objekta, odnosno u fazi otključavanja više nema radnji zaključavanja (naravno, obe ove faze uključuju i druge operacije nad objektima kao što su čitanje, obrada, upis objekata).Ova strategija omogućava da sezaključavanje izvrši kada je to potrebno, ali kada se otključa prvi slog, zaključavanja višenema. Faza otključavanja objekata počinje prvom radnjom otključavanja.

- Specijalan slučaj dvofaznog zaključavanja je korišćenje komandi COMMIT i ROLLBACK što je prisutno u mnogim RDBMS sistemima. U tom slučaju se zaključavanje vrši samom transakcijom, ali se zaključavanje ne ukida sve dok se ne izvrši komanda COMMIT ili ROLLBACK.

ŠTA JE DEAD LOCK?

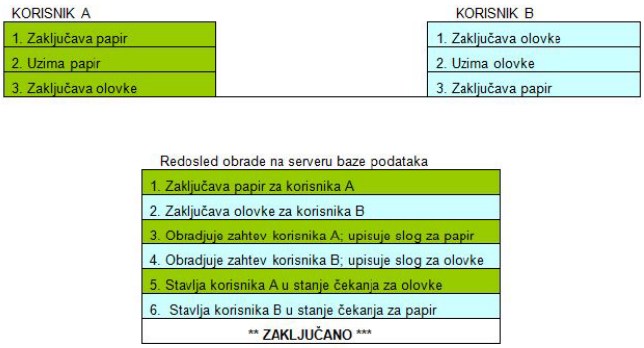
- Uzajamno blokiranje transakcija.
- Do deadlocka dolazi kada se dva procesa međusobno “takmiče” za iste resurse na način koji ne može sam da se razreši. Kada se ovo sedi server mora da blokira jedan od procesa, zbog čega transakcija ne uspeva da se izvrši.

- Osim što obezbeđuje linearizovanost, dvofaznost transakcija može proizvesti i neželjeni efekat poznat kao uzajamno blokiranje transakcija **(deadlock).**

- Nijedna transakcija ne može da nastavi sa radom, pa se jedna transakcija nasilno prekida uz poništenje svih njenih radnji i oslobađanje katanaca; zatim se ta transakcija ponovno aktivira.

Primer: Razmotrimo još jedan primer blokiranja transakcija u slučaju da dva korisnika žele da naruče dve stavke sa zaliha. Predpostavimo da korisnik A želi da naruči papir i ako ga dobije, želi da naruči olovke. Korisnik B želi da naruči olovke i ako ih dobije želi da naruči

papir. Redosed obrade je prikazan na slici 1 .



Slika 3.2.1 . Primer zaključavanja

OPTIMISTIČNO I PESIMISTIČNO ZAKLJUČAVANJE

- Generalno, postoje dva osnovna stila zaključavanja:
  - optimistično i
  - pesimistično.

- Kod **optimističnogzaključavanja**, 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 koflikt pojavio, transakcija se ponavlja sve dok se ona ne obradi bez konflikta.

- Kod **pesimističnogzaključavanja**, pretpostavka je da će se konflikt pojaviti. Najpre se vrši zaključavanje, zatim obrađuje transakcija, a zatim se vrši otključavanje.

- Prednost optimističnog zaključavanja je da zaključavanje traje mnogo kraće nego kod pesimističnog zaklj.

OPTIMALNA STRATEGIJA ZAKLJUČAVANJA: COMMIT I ROLLBACK

- **Naredbe Commit i Rollback mogu obezbediti da DBMS uključuje I isključuje zaključavanja i menja** nivo i tip zaključavanja.

- **NaredbaCOMMIT**označava uspešan kraj transakcije i trajan upis efekata svih radnji transakcije u bazu

- **NaredbaROLLBACK**označava neuspešan kraj transakcije i poništenje svih efekata koje su proizvele radnje te transakcije.

IT350 – LEKCIJA 12

NoSQL baze podataka

ŠTA SU XML ENABLE 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 document

- U interakciji između XML-a i baza podataka, XML se nože 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)

- Treba posmatrati dva procesa:

    • izdvajanje podataka iz baze i konstruisanje XML dokumenta – što je poznato kao publikovanje ili kompozicija.
    • izdvajanje podataka iz XML dokumenata i pamćenje tih podataka u bazi podataka – što je poznato kao isecanje ili dekompozicija.

- Bez obzira da li se XML dokumenti seku ili publikuju, treba naglasiti dve važne stvari:

    • U samoj bazi podataka XML se "ne vidi" – tj. XML dokument je potpuno izvan baze podatake. On se konstruiše na osnovu podataka koji se nalaze u bazi podataka ili se koristi kao izvor podataka koji se pamte u bazi.
    • Drugo, šema baze podataka se slaže sa XML šemom, što znači da je za svaku šemu baze
    podataka potrebna različita XML šema. Baza podataka koja na taj način koristi XML je poznata kao XML-enabled baza podataka.

MAPIRANJE ŠEME BAZE PODATAKA U XML ŠEMU I OBRNUTO

- Mapiranje šeme baze podataka u XML šemu je moguće na tri načina:
    1. bazirano na tabelama (table-based)
    2. objektno-relaciono mapiranje (object-relational)
    3. upitni jezici

- Objektno-relaciono mapiranje se može posmatrati na dva načina:
    1. primenom principa baziranog na tabelama,

2. objektno-relacionim mapiranjem

Mapiranje bazirano na tabelama (table-based) 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.

OBJEKTNO-RELACIONO MAPIRANJE

- Objekti koji sačinjavaju XML stablo se mapiraju u tabele, svojstvaobjekata u kolone a međusobne veze između objekata u relacijeprimarni ključ / strani ključ

Native XML baze podataka

POTREBA ZA NATIVE XML BAZAMA PODATAKA

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

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

- Polu-struktuirani 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.

- Transakcije čije izvršenje traje dugo. Mnoge transakcije elektronske trgovine danas koriste XML dokumente kao način za razmenu informacija između različitih delova aplikacije. Poštoove transakcije često mogu da traju nekoliko nedelja (na primer, 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.

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

XML MODEL PODATAKA

- Opisuje koji su delovi XML dokumenta logički značajni.

- **Native XML baza podataka** je baza podataka koja:

• **Definiše XML model podataka**. Minimalni model uključuje elemente, atribute,tekst I redosled dokumenata
• **XML dokument** koristi kao osnovnu logičku jedinicu za pamćenje podataka.

- **XML modeli podatakase koriste kao osnova za XML upitne jezike** i definiše minimalnukoličinu informacija koju native XML baze podataka moraju da pamte.

OSNOVNA JEDINICA ZA PAMĆENJE PODATAKA

- U relacionim bazama podataka to je red a u native XML bazama osnovna jedinica za pamćenje podataka je dokument. Obzirom da je bilo koji fragment dokumenta koji je označenkao jedan element potencijalno XML dokument, izbor šta će činiti jedan XML dokument je čisto stvar odluke dizajnera.

- Na primer:

  - **Knjiga pisana u XML-u**. Mada će se mnogi ljudi složiti da knjiga treba da bude podeljena na više dokumenata a da je takođe smešno da za svaki paragraf postoji poseban dokument, nije jasno šta je idealna veličina dokumenta. Na primer, treba li svaka knjiga da sadrži poglavlje? Sekciju? Podsekciju? To u potpunosti zavisi od knjige i uslova pod kojim će se ona koristi.

  - **Medicinski podaci zapamćeni kao XML dokumenti.** Poseban XML dokument možepostajati za svakog pacijenta, za svakog pružaoca medicinskih usluga (doktor, bolnica, klinika itd.) i za svaku osiguravajuću kuću. Na taj način je šema baze podataka do izvesnog stepena normalizovana, mada ta normalizacija nije kompletna u meri u kojoj se to može postići u relacionoj bazi podataka.

NAČIN NA KOJI SE MOGU IMPLEMENTIRATI NATIVE XML BAZEPODATAKA

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

- PamćenjemCelih XML dokumenata u CLOB strukturama relacionih baza podataka i indeksiranjem
  individualnih vrednosti elemenata i atributa.

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

- Parsiranih XML dokumenata kao DOM stabla u objekno-orijentisanoj bazi podataka.
- Parsiranih XML dokumenata u indeksiranom skupu heš tabela

XQUERY

- Xquery je dizajniran za upite nad XML dokumentima

POREĐENJE XQUERY-SQL

- Najbolji način da se objasni XQuery jezik je reći da je XQuery za XML isto što je i SQL za baze podataka. XQuery je dizajniran da pravi upite nad XML-om ne samo nad XML fajlovima većsvim podacima koji mogu biti predstavljeni XML-om pa čak i baze podataka, i poznat je joškao **XML query**. Dizajniran je tako da omogući lako manipulisanje metodama i upitima da bise pristupilo podacima iz XML-a. Za učenje XQuery-ja potrebno je prethodno znanje HTML /XHTML, XML / XML namespace -a i XPath-a.

- XQueryje jezik za pronalaženje, ekstrakciju i manipulaciju elementima i atributima iz XML dokumenta. On svaki XML dokument definiše kao stablo čiji su čvorovi elementi u odgovarajućem XML-u

NoSQL baze podataka

KORIŠĆENJE RELACIONE BAZE POGODNE ZA ČUVANJE POLUSTRUKTUIRANIH PODATAKA

Dok su relacione baze podataka prvobitno namenjene za smeštanje tabelarnih struktura one se nisu dobro pokazale pri pokušaju modeliranja ad-hoc relacija između polustruktuiranih podatak

ZAŠTO RELACIONE BAZE NISU POGODNE?

Kod relacionih baza podataka se javlja problem čvrsto povezanih domena.

KARAKTERISTIKE NOSQL BAZA PODATAKA

- Četri osnovne funkcionalnosti u kojima se ogledaju razlike između NoSQL-a i standardnih SQL sistema su:

  ▸**Nepotrebnost šeme:** Šema baze podataka je opis svih mogućih podataka i njihovih tipovakoji se u bazi mogu naći. NoSQL nije baziran na šemi, tako da šema nije neophodna. Korisniciovime dobijaju fleksibilnost u načinu na koji strukturiraju podatke.

  ▸**Relacionalnost:** U relacionalnim bazama pravimo puno relacija između tabela u kojima čuvamo podatke. Na primer lista transakcija može da bude povezaan sa korisnicima. Sa NoSQL bazama ova informacija je sačuvana kao agregacija, jedinstveni slog koji sadrži sve uvezi transakcija, uključujući i podatke o korisniku.

  ▸**Jaftiniji hardver:** Neke baze su dizajnirane da rade samo na najboljlim serverima, NoSQLbaza je drugačija i omogućava korisnicima da korišćenjem jeftinijih servera i spajanjem više njih postignu izuzetne performanse.

  ▸**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.

Sa NoSQL je moguće mnogo brže primeniti iterativni razvoj i dodavanje novina u odnosu naDBMS. Primenom NoSQL principa se ne smanjuje samo cena i olakša život programerima većse rešava i niz menadžerskih problema.

KONZISTENTNOST BAZA PODATAKA

ACID i BASE konzistentnost

- Konzistentnostje atribut baze koji znači da baza ima konzistentan pogled na podatke. Ovo praktično znači da svako čitanje posle upisivanja odmah prikazuje upisanu vrednost. Visoka konzistentnost je potrebna kod proizvoda koji koriste transakcije.

- Konzistentnost se često deli u dva nivoa:

  ▸ACID konzistentnost:predstavlja atomske tranzakcije i obično znači da će svaki upit nakonupisa videti izmenjene podatke.
  ▸BASE konzistentnost: znači da će upisani podaci pre ili kasnije početi da se prikazuju u upitima.

ACID KONZISTENTNOST

- Generalni skup principa za transakcione sisteme dok ih većina NoSQL baza podataka ne podržava

- U svetu relacionih baza podataka smo familijarni sa ACID transakcijama kojima se garantuje sigurno okruženje u kojem se radi sa podacima. ACID je generalni skup principa zatransakcione sisteme i nije vezan jedino za baze već se javlja i u puno drugih oblasti. ACIDznači:

  • **Atomičnost** (eng, Atomic)
  Sve operacije u transakciji moraju biti uspešne ili se transakcija mora poništiti (rolled back).
  • **Konzistentnost** (eng. Consistent)
  Po završetku transakcije, baza podataka je u konzistentnom (ispravnom) stanju.
  • **Izolovanost** (eng. Isolated)Akcije transakcija se sekvencijalno izvršavaju jedna nakod druge. Nije moguće nesekvencijalno, naizmenično izvršenje akcija transakcija.
  • **Trajnost** (eng. Durable)Rezultat primene transakcija je trajan, osim ako ne postoji greška.

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.

BASE znači da baza nema ACID garancije.

- BASE ima sledeće značenje:
  - **Osnovnaraspoloživost** (eng. Basic availability)- Ima se utisak da se sve vreme vrši pamćenje podataka

  - **Soft-state** - Pamćenje ne mora da bude kozistentno sa upisim niti različita ažuriranja moraju bitimeđusobno konzistentna sve vreme.

  - **Konačnakonzistentnost** (eng. Eventual consistency) - Konzistentnost se javlja u nekom kasnijem trenutku

- U **BASE** pamćenju podataka, podaci postaju raspoloživi ali se ne garantuje konzistentnost u trenutku upisa. BASE pamćenje ne obezbeđuje striktnu sigurnost, podaci postaju konzistentni kasnije, možda u vreme čitanja, ili će biti konzistentni ali samo za neke procese. Prednost BASE pristupa je što se tranzakcije sporije komituju, što znači i brže čitanje.

- Mane ovoga su što klijenti koji se konektuju da čitaju replike podatakamogu da vide informacije koje više nisu validne.

- Ovakav sistem je primenjiv na socijalnim mrežama, ukoliko napravite post na facebook-u, a vaši prijatelji to vide tek kroz nekoliko minuta, gubitak nije veliki, a poboljšanja u performansama će biti velika.

ČUVANJE DOKUMENATA PO ID-U

- Dokumenta se najčešće sačinjavaju mape i liste, koje omogućavajuuspostavljanje prirodne hijerarhije – najčešće korišćenjem formata kaošto su XML i JSON

- Generalno, kod čuvanja dokumenata po ID-u u NoSQL bazama podataka indeksi se koriste da bi se pretražio skup povezanih dokumenata za potrebe neke aplikacije.

GRAF BAZE PODATAKA (ENG. GRAPH DATABASES)

- U graf bazama podataka, veze između entiteta ne prikazujuuniformnost na nivou domena već je domen promenljivo struktuiran

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.

Čuvanje dokumenata u NoSQL bazama podataka

OSNOVNI ELEMENTI

- **Baza podataka, kolekcija i document**

- **Database**– bazu čini fizički kontejner strukture koja se zove kolekcija. Svaka baza ima svoj sopstveni skup file-ova na file sistemu. Jedan DB server obično ima više baza na sebi.Kolekcija **je slična RDBMS tabelama. Imena kolekcija su unikatna. Obično sukolekcije koje se sadrže u istoj bazi povezane ali nije neophodno naznačiti kako supovezane, što predstavlja ključnu razliku u odnosu na RDBMS.**

- **Dokumenti**– Ovo je najjednostavnija jedinica podataka u NoSQL bazama podataka, obično se sastoji od skupa parova ključ-vrednosti. Za razliku od unosa u RDBMS, NoSQL dokumenti Imaju dinamičku šemu. Dokumenti u jednoj kolekciji ne moraju da imaju uvek istu strukturu.

JSON FORMAT ZA ČUVANJE 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 je tekstualni format koji je kompletno nezavisan od programskog Jezika

IT350 – LEKCIJA 13

Java programiranje sa bazamapodataka

ŠTA JE JDBC?

- Standardni Java API koji omogućava da se veliki broj baze podataka različitih proizvođača koriste iz Java programskog jezika.JDBC(Java DataBase Connectivity) prestavlja standardni Java API koji omogućava bazama podataka da se nezavisno koriste iz Java programskog jezika sa velikim brojem različitihproizvođača samih baza.

- JDBC biblioteka uključuje API koji omogućava izvršenje svakog posla koji se obično povezuje sa radom nad bazama:
    1. Uspostavljenje konekcije sa bazom podataka.
    2. Kreiranje SQL ili MySQL naredbi.
    3. Izvršenje SQL ili MySQL upita u bazi podataka.
    4. Pregledavanje i modifikacija rezultujućih slogova.

- JDBC je specifikacija obezbeđuje kompletan skup interface-a koji omogućavaju portabilnost nad različitim bazama. Java može da se koristi za različite vrste aplikacija:
    - Java Applications
    - Java Applets
    - Java Servlets
    - Java ServerPages (JSPs)
    - Enterprise JavaBeans (EJBs).

O ČEGA SE SASTOJI JDBC API?

- Sastoji se od dva nivoa: JDBC API i JDBC Driver API JDBC API se sastoji od dva nivoa:
    - **JDBC API**
    - **JDBC Driver Manager**

- JDBC API koristi JDBC Driver manager i drivere za konkretnu bazu kako bi omogućio heterogenu konekciju na različite RDBMS-e. JDBC menager obezbeđuje da se driveri za konkrektnu bazu iskoriste za sve izvore podataka.JDBC Driver manager je u stanju da podrži više različitih drivera koji su konektovani na više
različitih baza.

- JDBC API obezbeđuje naredne interface-e i klase:

  DriverManager: Ova klasa barata listom drivera za različite baze podataka, povezuje zahteve za konekciju iz java aplikacije sa odgovarajućim driver-om korišćenjem pod protokola. Koristi

se onaj driver koji se prvi registruje i prepozna podprotokol.
Driver: Ovaj interface je zadužen za komunikaciju sa serverom baze. Naša interakcija će se retko odvijati direktno sa driver objektima, najčešće ćemo koristiti objekat tipa DriverManager.
Connection: Ovo je interface sa svim metodama za konektovanje sa bazom. Objekat konekcije predstavlja komunikacioni kontekst, sve komunikacije sa bazom idu kroz konekciju.
Statement: Objekti ovog tipa predstavljaju upite ka bazi podataka. Ovi upiti se izvršavaju kroz Connection objekat.
ResultSet: Ovi objekti sadrže podatke iz baze podataka nakon izvršavanja SQL upita korišćenjem Statement objekta i Connection objekta. ResultSet se koristi kao iterator koji nam omogućava da prolazimo kroz skupove podataka.
SQLException: Predstavlja klasu za izuzetke prilokom rada sa JDBC-em.

KAKO KREIRATI APLIKACIJU KORIŠĆENJEM JDBC-A?

- **Kroz šest koraka**

Šest osnovnih koraka za kreiranje JDBC aplikacije:

1. Importovanje paketa: zahteva uključivanje paketa koji sadrže JDBC klase neophodne za programiranje sa bazama podataka. Često je dovoljno iskoristiti import java.sql.*.
2. Registrovanje JDBC drajvera:Zahteva inicijalizaciju drajvera tako da se može otvoriti komunikacioni kanal sa Jave-om.
3. Otvaranje konekcije: Zahteva korišćenje metode DriverManager.getConnection() kako bi se kreirao Connection object, koji predstavlja fizičku konekciju sa bazom podataka.
4. Izvršavanje upita: zahteva korišćenje objekta tipa Statement kako bi se kreirale SQL naredbe nad bazom podataka.
5. Izvdajanje podataka iz skupa rezultata:Zhateva korišćenje odgovarajuće metode ResultSet.getXXX() za pretraživanje podataka iz skupa rezultata.
6. Čišćenje okruženja: Zahteva eksplicitno zatvaranja svih resursa baze podataka

ŠTA JE JDBC DRIVER?

- JDBC driver je implementacija definisanog interface-a u JDBC API-u koja omogućava konkretnu vezu sa konkretnim RDBMS-om.