BAZE PODATAKA – 2. MI

*****INSERT
* bez navedene liste atributa
 - moraju se navesti imena sviha atributa
* s navedenom listom atributa
 - ako neki atribut nije naveden, postavlja se DEFAULT
 - ako DEFAULT nije definiran, postavlja se NULL
 - ako je polje NOT NULL, javlja gresku i n-torka se ne zapisuje
* VALUES clause ili SELECT statement
* SELECT statement
 - ne smije sadrzavati ORDER BY, FIRST n, UNION
 - imena atributa u SELECT-u ne moraju odgovarati onima iz definicije tablice, ali tipovi podatka moraju biti kompatibilni
*****DELETE
* u WHERE dijelu DELETE naredbe dopusteno koristiti sve oblike uvjeta kao i u WHERE dijelu SELECT naredbe, ali u FROM dijelu podupita nije dopusteno koristi relaciju nad kojom se DELETE obavlja (ne smijemo iz iste tablice brisati i raditi podupit)
* ako se WHERE dio ne navede, brisu se sve n-torke
*****UPDATE
* kao i kod DELETE-a, u WHERE dijelu podupita, ne smijemo koristi relaciju nad kojom
izvrsavamo UPDATE
* ako se WHERE dio ne navede, sve n-torke se postavljaju na nove vrijednosti
* dopusteno koristiti CASE u definiciji vrijednosti na koje postavljamo atribute
* dva sintaksna oblika:
 - SET atr1=vr1, atr2=vr2 (…)
 - SET (atr1,atr2,…)=(vr1,vr2,…)
*****OBLIKOVANJE SHEME BAZE PODATAKA
* Lose koncipirane sheme:
 - redundancija
  * neracionalno koristenje prostora za pohranu
  * anomalije unosa, izmjene brisanja
 - pojava laznih n-torki
* Za definiranje sheme baze potrebno je poznavati semantiku (znacenje) podataka te medjuzavisnosti podataka.
*****FUNKCIJSKA ZAVISNOST
 - Relacija r sa shemom R, a X i Y su skupovi atributa, za koje vrijedi X podskup R, Y podskup R
 - Funkcijska zavisnost FZ, X -> Y vrijedi na shemi R ukoliko u svim dopustenim stanjima relacije r(R) svaki par n-torki t1 i t2 koje imaju jednake X vrijednosti, takodjer imaju jednake i Y vrijednosti:
t1(X) = t2(X) => t1(Y) = t2(Y)
 - Primjer: FZ postBr -> grad vrijedi na shemi jer sve vrijednosti postBr imaju jednake vrijednosti atributa grad
 - FZ proizlaze iz semantike (znacenja), a ne iz stanja relacije – iz stanja relacije mozemo utvrditi da neka potencijalna FZ ne vrijedi, ali nikad ne mozemo sa sigurnoscu zakljuciti da vrijedi – semantika podataka odlucuje.
*****ARMSTRONGOVI AKSIOMI
Na rel. shemi R, neka su X, Y, Z skupovi atributa i neka vrijedi: X podskup R, Y podskup R, Z podskup R
 - (A-1) Refleksivnost - Ako je Y podskup X tada vrijedi X -> Y
 - (A-2) Uvecanje - Ako u R vrijedi X -> Y, tada vrijedi i XZ -> Y
 - (A-3) Tranzitivnost - Ako u R vrijedi X -> Y i Y -> Z, tada vrijedi i X -> Z
Na rel. shemi R, neka su X, Y, Z, V skupovi atributa i neka vrijedi: X podskup R, Y podskup R, Z podskup R, V podskup R
 - (P-1) Pravilo unije (pravilo o aditivnosti) - Ako u R vrijedi X -> Y i X -> Z, tada vrijedi i X -> YZ
 - (P-2) Pravilo dekompozicije (pravilo o projektivnosti) - Ako u R vrijedi X -> YZ, tada vrijedi i X -> Y
 - (P-3) Pravilo o pseudotranzitivnosti - Ako u R vrijedi X -> Y i VY -> Z, tada vrijedi i XV -> Z
 PRAVILO O AKUMULACIJI - Ako u R vrijedi X -> VZ i Z -> W, tada vrijedi i X -> VZW
*****KLJUC RELACIJE
Kljuc relacije je skup atributa koji nedvosmisleno odredjuje n-torke, a ima svojstvo da funkcijski odredjuje atribute u preostalom dijelu relacije. Oznacavamo:
KRELACIJA = {skup_atributa}
Imamo primarne kljuceve, alternativne kljuceve te atribute koji nisu dio kljuca (atributi zavisnog dijela relacije).
*****NORMALIZACIJA
Znanja o FZ koriste se pri normalizaciji. Cilj je ukloniti redundanciju i sprijeciti pojavu laznih n-torki. Tocno definiranom metodom se odredjuje dobra zamjena za lose koncipiranu relacijsku shemu.
*****Postupak:
* Dekompozicija – pocetne relacije se dekomponiraju na temelju uocenih FZ
* Sinteza – zadan je skup atributa, nad njima skup zadanih FZ iz kojih se sintetiziraju relacijske sheme koje zadovoljavaju 3NF
*****PRVA NORMALNA FORMA (1NF)
* Domene atributa sadrze samo jednostavne (nedjeljive) vrijednosti
* Vrijednost svakog atributa je samo jedna vrijednost iz domene tog atributa
* Nekljucni atributi relacije ovise o kljucu relacije
Shema baze podataka R={A,B,C,D} je u 1NF ako je svaka rel. shema A,B,C,D u 1NF.
1NF izdvajanjem atributa
Pocetna relacija se dekomponira na vise relacija, izdvaja se skup atributa koji se ponavlja s jednakom kratnoscu, zajedno s kljucem originalne relacije. (vidi slajdove 7/16).
1NF promjenom kljuca
Atribute ne razdvajamo nego promijenimo kljuc relacije.
*****DRUGA NORMALNA FORMA (2NF)
Relacijska shema je u 2NF ako je u 1NF i ako je svaki atribut iz zavisnog dijela potpuno funkcijski ovisan o svakom kljucu relacije.
Shema baze podataka R={A,B,C,D} je u 2NF ako je svaka rel. shema A,B,C,D u 2NF.
Skup atributa Y potpuno je funkcijski ovisan o skupu atributa X ako Y funkcijski ovisi o X i ne postoji pravi podskup od X koji funkcijski odredjuje Y.
Normalizacijom na 2NF nastaju:
* Relacijska shema koja sadrzi skup atributa koji su bili nepotpuno funkcijski ovisni o
kljucu i dio kljuca o kojem su potpuno funkcijski ovisni.
* Relacijska shema koja sadrzi kljuc origianlne relacije i skup atributa koji su potpuno funkcijski ovisni o kljucu.
*****TRECA NORMALNA FORMA (3NF)
Relacijska shema je u 3NF ako je u 1NF i ako niti jedan atribut iz zavisnog dijela nije tranzitivno funkcijski ovisan o bilo kojem kljucu relacije.
Shema baze podataka R={A,B,C,D} je u 3NF ako je svaka rel. shema A,B,C,D u 3NF.
Normalizacija na 2NF nije nuzni preduvjet za normalizaciju na 3NF jer se nepotpune FZ mogu promatrati kao tranzitivne FZ.
*****BOYCE-CODDOVA NORMALNA FORMA (BCNF)
Normalizaciju na BCNF nije nuzno provoditi jer su rijetki slucajevi da je relacijska shema u 3NF, a da istovremeno nije i u BCNF.
*****FIZICKA ORGANIZACIJA PODATAKA
* Ne utjece na rezultate operacija s podacima, ali ima vrlo velik utjecaj na ucinkovitost SUBP-a.
* Podaci se spremaju u sekundarnu memoriju (tvrdi disk) radi kapaciteta i postojanosti podataka.
* Potrebno minimizirati broj U/I operacija pri pohrani i dohvatu podataka te utrosak prosotra za pohranu.
* Dvije najcesce metode: heap (neporedana) datoteka i binarno stablo (B-stablo)
*****HEAP DATOTEKA
U prosjeku je potrebno obaviti n/2 UI operacija, gdje je n broj fizickih blokova, odnosno n-torki. U najgorem slucaju, kad nema zapisa, potrebno je obaviti n operacija – linearno pretrazivanje. Koristi se za mali broj n-torki.
*****BINARNA STABLA
Definicije pojmova:
* Razina cvora: duljina puta od korijena do cvora
* Dubina stabla: najveca duljina puta od korijena do lista
* Red stabla: najveci broj djece koje cvor moze imati

* Balansirano stablo: ako je dubina stabla jednaka za svaki list u stablu; oznaka B+
* U B+ stablu reda n, interni cvor sadrzi:
- najvise n kazaljki
- najmanje najvece_cijelo(n/2) kazaljki
oznaka najvece_cijelo(n) odnosi s na najmanji cijeli broj veci ili jednak n
ovo ogranicenje ne vrijedi za korijen stabla
- u p kazaljki u cvoru, broj pripadnih vrijednosti Ki u cvoru je p-1
- Ki je vrijednost kljuca
* U B+ stablu reda n, listsadrzi:
- najvise n-1 vrijednosti Ki i pripadnih kazaljki na zapise
- najmanje najvece_cijelo((n-1)/2) vrijednosti Ki i pripadnih kazaljki na zapise
- svi listovi sadrze kazaljku na sljedeci list
* Vazno dobro svojstvo algoritama B-stabla:
dubina stabla se automatski prilagodjava broju zapisa – cvorovi stabla (osim korijena) su uvijek barem 50% popunjeni

*****INDEKSI

Obavljanjem naredbe za kreiranje indeksa nad relacijom, nad blokovima s podacima relacije formira se struktura B-stabla.
Sintaksa – kreiranje: CREATE (UNIQUE) INDEX nazivIndeksa ON relacija (atribut)
Sintaksa – brisanje: DROP INDEX nazivIndeksa
Ako je indeks UNIQUE, ne mogu se kreirati n-torke s duplikatom vrijednosti atributa koji je sadrzan u indeksu, a ako takve n-torke vec postoje, indeks se nad njima nece moci niti kreirati.
Indekse raditi za atribute koji su kljucevi relacije ili atributi koji se cesto koriste za postavljanje uvjeta selekcije, po kojima sortiramo ili grupiramo izraze.
Indekse ne raditi nad atributima s relativno malim brojem razlicitih vrijednosti, te ako relaciji predstoji velik broj upisa, izmjena ili brisanja n-torki te ako relacija sadrzi relativno mali broj n- torki.
Slozeni indeksi: CREATE INDEX nazivIndeksa ON relacija (atr1, atr2, ...)
Uz atribute atr1, atr2... dozvoljeno je stavljati DESC oznaku ako zelimo koristiti taj indeks za inverzno sortiranje.

*****INTEGRITET BAZE PODATAKA

Integritet moze biti narusen zbog:
* Slucajne pogreske korisnika kod unosa/izmjene
* Slucajne pogreske programera ili sustava
Shema baze sastoji se od skupa relacijskih shema R i skupa integritetskih ogranicenja (integrity constraints) IC.
Ispravna instanca baze podataka mora zadovoljavati svaintegritetska ogranicenja.
Definicije IC-ova pohranjuju se u rjecnik podataka.
SUBP provjerava IC prvi obavljanju svake operacije koja mijenja sadrzaj baze.
Promjene koje bi narusile bilo koji uvjet iz skupa IC nece biti izvrsene.

*****VRSTE INTEGRITETSKIH OGRANICENJA:

*** Entitetski integritet (entity integrity)
Niti jedan atribut primarnog kljucane smije poprimiti NULL vrijednost.
*** Integritet kljuca (key integrity)
U relaciji ne smiju postojati 2 n-torke s jednakim vrijednostima kljuca (za sve kljuceve).
*** Domenski integritet (domain integrity)
Atribut moze poprimiti samo jednu vrijednost iz domene atributa.
(npr. skup cijelih brojeva iz zadanog intervala)
*** Ogranicenja NULL vrijednosti (constraints on NULL)
Za odredjene atribute, moze se definirati ogranicenje prema kojem vrijednost atributa ne smije biti NULL.
*** Referencijski integritet (referential integrity)
Ref. integritet odnosi se na konzistentnost medju n-torkama dviju relacija (ili iste relacije). N- torka iz jedne relacije, koja se poziva na drugu relaciju, moze se pozivati samo na postojece n- torke u toj relaciji.
*** Pojam stranog kljuca: primarni kljuc jedne relacije strani je kljuc druge relacije (ili iste kod slucaja hijerarhijske raspodjele), ako vrijedi referencijski integriret izmedju tih relacija, tj. referencijski integriret proizlazi iz pojma stranog kljuca.
*** Opca integritetska ogranicenja (geneial integrity constraints)
Odnose se na ogranicenja generalnog oblika, npr. poslovna pravila.
Mogu se definirati:
- u okviru naredbe za kreiranje relacije - CREATE TABLE
- naknadnim definiranjem pomocu naredbe - ALTER TABLE
Primjeri dodavanja primarnog kljuca (SUBP osigurava entitetski integritet i integritet kljuca):
***U sljedecem slucaju definiramo PK neposredno uz definiciju samog atributa:

```
CREATE TABLE nastavnik (
sifNast INTEGER PRIMARY KEY, jmbgNast CHAR(13),
prezNast CHAR(40) );
```

***U sljedecem slucaju definiramo PK nad skupom atributa:

```
CREATE TABLE tablica ( atribut1 INTEGER, atribut2 INTEGER, atribut3 CHAR(40),
PRIMARY KEY (atribut1, atribut2) );
```

U sljedecem slucaju definiramo UNIQUE atribut, SUBP osigurava integritet kljuca:

```
CREATE TABLE tablica ( atribut1 INTEGER, atribut2 INTEGER, atribut3 CHAR(40),
UNIQUE (atribut1) );
```

***... ili kao gore, ali neposredno uz definiciju atributa:

```
CREATE TABLE tablica (
atribut1 INTEGER UNIQUE, atribut2 INTEGER,
atribut3 CHAR(40) );
```

*** Domenski integritet djelomicno je definiran samom definicijom tipa podatak za atribut (primjerice, SMALLINT [-32767,32767]). Dodatno, mozemo dodati CHECK kao provjeru domenskog integriteta:

```
CREATE TABLE tablica ( atribut1 INTEGER, postanskiBroj INTEGER, atribut3
CHAR(40),
CHECK (postanskiBroj BETWEEN 10000 AND 99999) );
```

*** Kao i u gornjim slucajevima, na isti nacin dodajemo CHECK neposredno uz atribut, no ogranicenja koja se ticu odnosa medju vrijednostima atributa se ne mogu napisati neposredno uz definiciju atributa, nego kao u gornjem slucaju, na kraju definicije relacije. ..... CHECK (datZap-datRod >= 16*365 AND datZap-datRod <= 65*365)
*** Ogranicenje NULL vrijednosti postizemo dodavanjem rezerviranih rijeci NOT NULL iza tipa podatka pri definiciji atributa.

```
CREATE TABLE tablica (
atribut1 INTEGER NOT NULL, atribut2 INTEGER,
atribut3 CHAR(40) );
```

*** Dodavanje stranog kljuca (foreign key), navodimo najprije koji je atribut FK, a zatim odakle taj FK potjece. Primjerice:

```
CREATE TABLE tablica ( sifOrgJed INTEGER, sifNadOrgJed INTEGER, atribut3
CHAR(40),
FOREIGN KEY (sifNadOrgJed) REFERENCES tablica(sifOrgJed) );
```

*** U gornjem primjeru, SUBP osigurava referencijski integritet stranog kljuca, koji se poziva na primarni kljuc (u ovom slucaju iste relacije). Jednako kao i u prethodnim primjerima, mozemo FK dodavati neposredno uz atribut, tako da nakon tipa podatka navedemo REFERENCES nazivRelacije(nazivAtributa).
*** Dodatna ogranicenja pri stranim kljucevima:
ON DELETE NO ACTION
zanemaruje operaciju brisanja pozivane n-torke
ON DELETE SET NULL
vrijednosti FK u n-torkama koje se pozivaju na obrisanu n-torku postaviti na NULL vrijednosti
ON DELETE SET DEFAULT
kao gore, ali na default vrijednosti
ON DELETE CASCADE
brisu se pozivajuce n-torke
ON UPDATE NO ACTION
odbija se operacija izmjene pozivane n-torke
ON UPDATE SET NULL
prilikom izmjene vrijednosti, n-torke koje se pozivaju na izmijenjenu n-torku postaju NULL
ON UPDATE SET DEFAULT
kao i gore, samo postaju default vrijednosti
ON UPDATE CASCADE
izmjenuju se sve n-torke na novu vrijednost P.K. pozivane n-torke
*** Imenovanje integritetskih ogranicenja
Dodaje se rezervirana rijec CONSTRAINT i nakon toga naziv I.O. Primjerice:

```
FOREIGN KEY (mbrStud) REFERENCES stud(mbrStud) CONSTRAINT fkIspitStud
```