# BAZE PODATKA – 2. MEĐUISPIT

## PODUPITI

- Mogu se ugraditi u WHERE dio, HAVING dio i SELECT listu.
- Može sadržavati sve osim ORDER BY.
- Nekad bolje rješavati JOIN-ovima nego podupitima, ali neke upite je moguće samo podupitima riješiti.

### KORELIRANI PODUPIT

- korelirani podupit: ako se u podupitu koriste atributi iz vanjskog upita
- u podupitu se mogu koristiti atributi iz vanjskog upita, ali ne i obrnuto
- ako se imena atributa i/ili relacija vanjskog upita podudaraju s onima iz podupita, imena navedena u podupitu odnose se na imena atributa/relacija iz podupita, odnosno imena relacija iz vanjskog upita odnose se na vanjski upit

### JEDNOSTUPČANI PODUPIT

- *izraz* { < | <= | = |<> | > | > | >=} ALL (*podupit*)
  - *true* ako izraz odgovara svim vrijednostima podupita
- *izraz* { < | <= | = |<> | > | > | >=} SOME (*podupit*)
  - *true* ako izraz odgovara barem jednom od vrijednosti podupita
- SOME i ANY su sinonimi
- *izraz* IN *podupit*
  - ekvivalentno s: *izraz* = SOME (*podupit*)
- *izraz* NOT IN *podupit*
  - ekvivalentno s: *izraz* <> ALL (*podupit*)
- EXISTS (*podupit*)
  - *true* ako rezultat podupita sadrži barem jednu n-torku, bez obzira na broj atributa i njihove vrijednosti
  - pogodan za presjek skupa
- NOT EXISTS (*podupit*)
  - *true* ako rezultat podupita ne sadrži niti jednu n-torku
  - pogodan za razliku skupa

# 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 grešku i n-torka se ne zapisuje
- VALUES *clause* ili SELECT *statement*
- SELECT *statement*
  - ne smije sadržavati 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 dopušteno koristiti sve oblike uvjeta kao i u WHERE dijelu SELECT naredbe, ali u FROM dijelu podupita nije dopušteno koristi relaciju nad kojom se DELETE obavlja (ne smijemo iz iste tablice brisati i raditi podupit)
- ako se WHERE dio ne navede, brišu se sve n-torke

# UPDATE

- kao i kod DELETE-a, u WHERE dijelu podupita, ne smijemo koristi relaciju nad kojom izvršavamo UPDATE
- ako se WHERE dio ne navede, sve n-torke se postavljaju na nove vrijednosti
- dopušteno koristiti CASE u definiciji vrijednosti na koje postavljamo atribute
- dva sintaksna oblika:
  - SET atr1=vr1, atr2=vr2 (…)
  - SET (atr1,atr2,…)=(vr1,vr2,…)

# BAZE PODATKA – SAŽETAK 6. PREDAVANJA

## OBLIKOVANJE SHEME BAZE PODATAKA

- Loše koncipirane sheme:
  - redundancija
    - neracionalno korištenje prostora za pohranu
    - anomalije unosa, izmjene brisanja
  - pojava lažnih n-torki
- Za definiranje sheme baze potrebno je poznavati semantiku (značenje) podataka te međuzavisnosti podataka.

## FUNKCIJSKA ZAVISNOST

- Relacija *r* sa shemom *R,* a *X* i *Y* su skupovi atributa, za koje vrijedi $X \subseteq R, Y \subseteq R$
- Funkcijska zavisnost FZ, $X \rightarrow Y$ vrijedi na shemi R ukoliko u svim dopuštenim stanjima relacije *r(R)* svaki par n-torki t1 i t2 koje imaju jednake X vrijednosti, također imaju jednake i Y vrijednosti:

$$t_1(X) = t_2(X) \Rightarrow t_1(Y) = t_2(Y)$$

- Primjer: FZ $postBr \rightarrow grad$ vrijedi na shemi jer sve vrijednosti *postBr* imaju jednake vrijednosti atributa *grad*
- FZ proizlaze iz semantike (značenja), a ne iz stanja relacije – iz stanja relacije možemo utvrditi da neka potencijalna FZ ne vrijedi, ali nikad ne možemo sa sigurnošću zaključiti da vrijedi – semantika podataka odlučuje.

### ARMSTRONGOVI AKSIOMI

Na rel. shemi *R*, neka su *X, Y, Z* skupovi atributa i neka vrijedi: $X \subseteq R, Y \subseteq R, Z \subseteq R$

- (A-1) Refleksivnost
  Ako je $Y \subseteq X$ tada vrijedi $X \rightarrow Y$

- (A-2) Uvećanje
  Ako u *R* vrijedi $X \rightarrow Y$, tada vrijedi i $XZ \rightarrow Y$

- (A-3) Tranzitivnost
  Ako u *R* vrijedi $X \rightarrow Y$ i $Y \rightarrow Z$, tada vrijedi i $X \rightarrow Z$

Na rel. shemi *R*, neka su *X, Y, Z, V* skupovi atributa i neka vrijedi: $X \subseteq R, Y \subseteq R, Z \subseteq R, V \subseteq R$

- (P-1) Pravilo unije (pravilo o aditivnosti)
  Ako u *R* vrijedi $X \rightarrow Y$ i $X \rightarrow Z$, tada vrijedi i $X \rightarrow YZ$

- (P-2) Pravilo dekompozicije (pravilo o projektivnosti)
  Ako u *R* vrijedi $X \rightarrow YZ$, tada vrijedi i $X \rightarrow Y$

- (P-3) Pravilo o pseudotranzitivnosti
  Ako u *R* vrijedi $X \rightarrow Y$ i $VY \rightarrow Z$, tada vrijedi i $XV \rightarrow Z$

### PRAVILO O AKUMULACIJI

- Ako u $R$ vrijedi $X \rightarrow VZ$ i $Z \rightarrow W$, tada vrijedi i $X \rightarrow VZW$

## KLJUČ RELACIJE

Ključ relacije je skup atributa koji nedvosmisleno određuje n-torke, a ima svojstvo da funkcijski određuje atribute u preostalom dijelu relacije.

Označavamo:

$K_{RELACIJA}$ = {skup_atributa}

Imamo primarne ključeve, alternativne ključeve te atribute koji nisu dio ključa (atributi zavisnog dijela relacije).

# BAZE PODATKA – SAŽETAK 7. PREDAVANJA

## NORMALIZACIJA

Znanja o FZ koriste se pri normalizaciji. Cilj je ukloniti redundanciju i spriječiti pojavu lažnih n-torki. Točno definiranom metodom se određuje dobra zamjena za loše koncipiranu relacijsku shemu.

### NORMALNE FORME

- Prva normalna forma – 1NF
- Druga normalna forma – 2NF
- Treća normalna forma – 3NF
- Boyce-Coddova normalna forma – BCNF

  *^ temelje se na funkcijskim zavisnostima*

- Četvrta normalna forma 4NF
  *temelji se na višeznačnim zavisnostima*
- Projekcijsko spojna normalna forma – PJNF
  *temelji se na spojnim zavisnostima*

Postupak:

- *Dekompozicija* – početne relacije se dekomponiraju na temelju uočenih 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 sadrže samo jednostavne (nedjeljive) vrijednosti
- Vrijednost svakog atributa je samo jedna vrijednost iz domene tog atributa
- Neključni atributi relacije ovise o ključu 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**
Početna relacija se dekomponira na više relacija, izdvaja se skup atributa koji se ponavlja s jednakom kratnošću, zajedno s ključem originalne relacije. (vidi slajdove 7/16).

**1NF promjenom ključa**
Atribute ne razdvajamo nego promijenimo ključ 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 ključu 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 određuje Y.

Normalizacijom na 2NF nastaju:

- Relacijska shema koja sadrži skup atributa koji su bili nepotpuno funkcijski ovisni o ključu i dio ključa o kojem su potpuno funkcijski ovisni.
- Relacijska shema koja sadrži ključ origianlne relacije i skup atributa koji su potpuno funkcijski ovisni o ključu.

## TREĆA 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 ključu 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 nužni preduvjet za normalizaciju na 3NF jer se nepotpune FZ mogu promatrati kao tranzitivne FZ.

## BOYCE-CODDOVA NORMALNA FORMA (BCNF)

Normalizaciju na BCNF nije nužno provoditi jer su rijetki slučajevi da je relacijska shema u 3NF, a da istovremeno nije i u BCNF.

# BAZE PODATKA – SAŽETAK 9. PREDAVANJA

## FIZIČKA ORGANIZACIJA PODATAKA

- Ne utječe na rezultate operacija s podacima, ali ima vrlo velik utjecaj na učinkovitost 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 utrošak prosotra za pohranu.
- Dvije najčešće metode: *heap* (neporedana) datoteka i binarno stablo (B-stablo)

### HEAP DATOTEKA

U prosjeku je potrebno obaviti $\frac{n}{2}$ UI operacija, gdje je *n* broj fizičkih blokova, odnosno n-torki. U najgorem slučaju, kad nema zapisa, potrebno je obaviti *n* operacija – linearno pretraživanje. Koristi se za mali broj n-torki.

### BINARNA STABLA

Definicije pojmova:

- Razina čvora: duljina puta od korijena do čvora
- Dubina stabla: najveća duljina puta od korijena do lista
- Red stabla: najveći broj djece koje čvor može imati
- Balansirano stablo: ako je dubina stabla jednaka za svaki list u stablu; oznaka $B^+$

- U $B^+$ stablu reda *n*, <u>interni čvor</u> sadrži:
  - najviše *n* kazaljki
  - najmanje $|n/2|$ kazaljki
    *oznaka* |n| *odnosi s na najmanji cijeli broj veći ili jednak* n
    *ovo ograničenje ne vrijedi za korijen stabla*
  - u *p* kazaljki u čvoru, broj pripadnih vrijednosti $K_i$ u čvoru je *p-1*
  - $K_i$ je vrijednost ključa
- U $B^+$ stablu reda *n*, <u>list</u> sadrži:
  - najviše *n-1* vrijednosti $K_i$ i pripadnih kazaljki na zapise
  - najmanje $|\frac{n-1}{2}|$ vrijednosti $K_i$ i pripadnih kazaljki na zapise
  - svi listovi sadrže kazaljku na sljedeći list

- Važno <u>dobro</u> svojstvo algoritama B-stabla:
  dubina stabla se automatski prilagođava broju zapisa – čvorovi 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 sadržan u indeksu, a ako takve n-torke već postoje, indeks se nad njima neće moći niti kreirati.

Indekse raditi za atribute koji su ključevi relacije ili atributi koji se često koriste za postavljanje uvjeta selekcije, po kojima sortiramo ili grupiramo izraze.

Indekse ne raditi nad atributima s relativno malim brojem različitih vrijednosti, te ako relaciji predstoji velik broj upisa, izmjena ili brisanja n-torki te ako relacija sadrži relativno mali broj n-torki.

Složeni indeksi: `CREATE INDEX nazivIndeksa ON relacija (atr1, atr2, ...)`

Uz atribute `atr1, atr2...` dozvoljeno je stavljati DESC oznaku ako želimo koristiti taj indeks za inverzno sortiranje.

# BAZE PODATKA – SAŽETAK 10. PREDAVANJA

## INTEGRITET BAZE PODATAKA

Integritet može biti narušen zbog:

- Slučajne pogreške korisnika kod unosa/izmjene
- Slučajne pogreške programera ili sustava

Shema baze sastoji se od skupa relacijskih shema **R** i skupa integritetskih ograničenja (integrity constraints) **IC**.
Ispravna instanca baze podataka mora zadovoljavati sva integritetska ograničenja.
Definicije **IC**-ova pohranjuju se u rječnik podataka.

SUBP provjerava **IC** prvi obavljanju svake operacije koja mijenja sadržaj baze. Promjene koje bi narušile bilo koji uvjet iz skupa **IC** neće biti izvršene.

### VRSTE INTEGRITETSKIH OGRANIČENJA:

- Entitetski integritet (*entity integrity*)
  *Niti jedan atribut primarnog ključa ne smije poprimiti NULL vrijednost.*

- Integritet ključa (*key integrity*)
  *U relaciji ne smiju postojati 2 n-torke s jednakim vrijednostima ključa (za sve ključeve).*

- Domenski integritet (*domain integrity*)
  *Atribut može poprimiti samo jednu vrijednost iz domene atributa.*
  (npr. skup cijelih brojeva iz zadanog intervala)

- Ograničenja NULL vrijednosti (*constraints on NULL*)
  *Za određene atribute, može se definirati ograničenje prema kojem vrijednost atributa ne smije biti NULL.*

- Referencijski integritet (*referential integrity*)
  *Ref. integritet odnosi se na konzistentnost među n-torkama dviju relacija (ili iste relacije). N-torka iz jedne relacije, koja se poziva na drugu relaciju, može se pozivati samo na postojeće n-torke u toj relaciji.*

  Pojam stranog ključa: primarni ključ jedne relacije strani je ključ druge relacije (ili iste kod slučaja hijerarhijske raspodjele), ako vrijedi referencijski integriret između tih relacija, tj. referencijski integriret proizlazi iz pojma stranog ključa.

- Opća integritetska ograničenja (*geneial integrity constraints*)
  *Odnose se na ograničenja generalnog oblika, npr. poslovna pravila.*

# IMPLEMENTACIJA INTEGRITETSKIH OGRANIČENJA U SQL-U

Mogu se definirati:

- u okviru naredbe za kreiranje relacije - CREATE TABLE
- naknadnim definiranjem pomoću naredbe - ALTER TABLE

Primjeri dodavanja primarnog ključa (SUBP osigurava entitetski integritet i integritet ključa):

U sljedećem slučaju definiramo **PK neposredno uz definiciju samog atributa**:

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

);
```

U sljedećem slučaju definiramo **PK nad skupom atributa**:

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

U sljedećem slučaju definiramo **UNIQUE atribut**, SUBP osigurava integritet ključa:

```
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** djelomično je definiran samom definicijom tipa podatak za atribut (primjerice, SMALLINT [-32767,32767]). Dodatno, možemo 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 slučajevima, na isti način dodajemo CHECK neposredno uz atribut, no ograničenja koja se tiču odnosa među vrijednostima atributa se ne mogu napisati neposredno uz definiciju atributa, nego kao u gornjem slučaju, na kraju definicije relacije.

```
........ CHECK (datZap-datRod >= 16*365 AND datZap-datRod <= 65*365)
```

**Ograničenje NULL vrijednosti** postižemo dodavanjem rezerviranih riječi NOT NULL iza tipa podatka pri definiciji atributa.

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

**Dodavanje stranog ključa** (foreign key), navodimo najprije koji je atribut FK, a zatim odakle taj FK potječe. 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 ključa, koji se poziva na primarni ključ (u ovom slučaju iste relacije). Jednako kao i u prethodnim primjerima, možemo FK dodavati neposredno uz atribut, tako da nakon tipa podatka navedemo `REFERENCES nazivRelacije(nazivAtributa)`.

**Dodatna ograničenja pri stranim ključevima:**

`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`
brišu se pozivajuće 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 ograničenja**

Dodaje se rezervirana riječ CONSTRAINT i nakon toga naziv I.O. Primjerice:

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