INSERT, UPDATE, DELETE NAREDBE

VII auditorne vježbe

Teme

- Indeksi u SQL-u
  - Jedinstvenost vrijednosti u indeksu
  - Uticaj poretka vrijednosti u indeksu na mogućnost korištenja indeksa za sortiranje
  - Upotreba indeksa
- INSERT naredba
  - Načini korištenja INSERT naredbe
  - INSERT naredba i SERIAL tip podatka
- DELETE naredba
- UPDATE naredba

Indeksi u SQL-u

- Brzina pristupa podacima u relaciji je važno svojstvo sistema za upravljanje podacima.
- Najjednostavniji način pristupa, sekvencijalna pretraga, u većini slučajeva ne zadovoljava.
- Kreiranjem indeksa formira se struktura B-stabla koja omogućava nesekvencijalni pristup do n-torke u relaciji.
- Nesekvencijalni pristup moguć je prema vrijednostima onih atributa nad kojima je izgrađena indeksna struktura.
- Nad jednom relacijom može biti izgrađeno više indeksa, od kojih svaki može sadržavati jedan ili više atributa.
  - Ako je nad relacijom R kreiran indeks za atribut A, može se nesekvencijalno pristupiti do n-torki uz korištenje atributa A kao ključa za dohvat
  - Ako je nad relacijom R kreiran kompozitni indeks za npr. atribute A, B, C do n-torke se može pristupiti korištenjem bilo koje od navedenih kombinacija atributa kao ključa za dohvat – A, AB, ABC

Indeksi u SQL-u

- Prilikom postavljanja upita u jeziku SQL, sistem sam određuje koji će se od indeksa koristiti za pristup.
- Ukoliko odgovarajućeg indeksa nema, pristup n-torkama može biti jedino sekvencijalan.
- Osim radi pobljšanja performansi sitema, indeksi se kreiraju i radi osiguranja jedinstvenosti vrijednosti atributa u relaciji (npr. u relaciji mjesto ne mogu postojati dva grada s poštanskim brojem jednakim 75000).
- Ukoliko je indeks kreiran kao indeks s jedinstvenim vrijednostima (unique index), sistem ne dozvoljava da se u relaciji pojace dvije n-torke koje bi imale jednake vrijednosti atributa nad kojima je izgrađen takav indeks.
- Ovakva karakteristika indeksa koristi se za osigravanje jedinstvenosti ključa i relaciji.

Indeksi u SQL-u

- Indeks može biti izgrađen tako da su vrijednosti u indeksnim blokovima poredane od manjih prema većim, ili obratno.
- SUBP može indeks pretraživati od naprijed ili straga, tako da se indeks može koristiti za sortiranje zapisa u smjeru u kojem su poredane vrijednosti u indeksnim blokovima, ali i u obrnutom smjeru.
- To znači da poredak vrijednosti u indeksnim blokovima indeksa sastavljenih od samo jednog atributa nije bitan.
- Ako je indeks sastavljen od više atributa, te se prema tim atributima obavlja sortiranje, o poretku vrijednosti u indeksu ovisi mogućnost korištenja tog indeksa prilikom sortiranja.
  - Npr. ukoliko je indeks kreiran za atribute x DESC, y DESC, tada se taj indeks može koristiti za sortiranje u smjeru x DESC, y DESC, te za sortiranje u smjeru x ASC, y ASC, ali ne i za sortiranje oblika x ASC, y DESC ili x DESC, y ASC. Indeks koji omogućava posljednja dva navedena oblika je indeks oblika x DESC, y ASC (također je to moguće ostvariti i indeksom oblika x ASC, y DESC).

Upotreba indeksa

- Opšti oblik naredbe za kreiranje indeksa je sljedeći:

```
CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX index_name
[index_type] ON tbl_name (index_col_name,...)
[index_option] [algorithm_option | lock_option] ...

index_col_name:
     col_name [(length)] [ASC | DESC]
```

- Primjer:

```
CREATE UNIQUE INDEX uniqueZupanija
ON zupanija (nazZupanija)
```

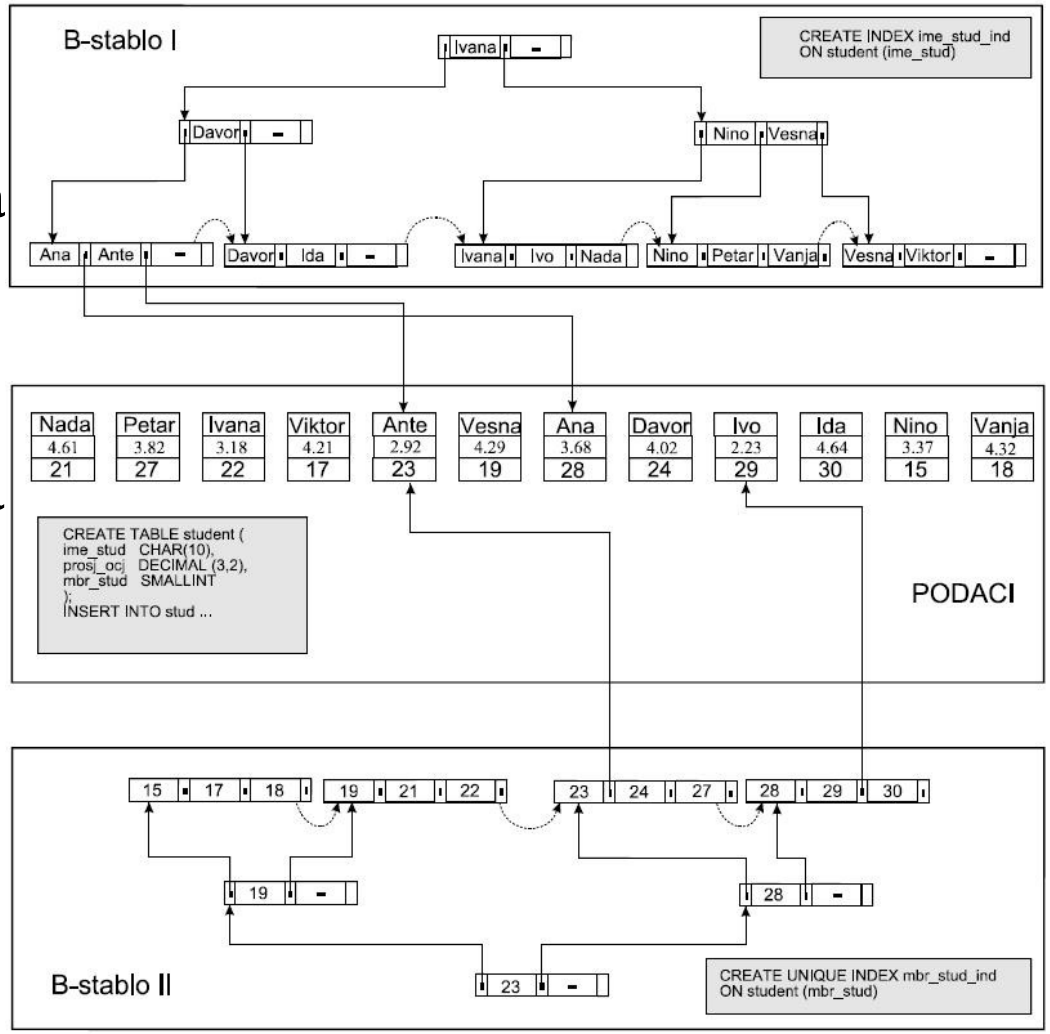
- Uništiti indeks se može naredbom:

```
DROP INDEX indexName
```

- Indekse bi u principu trebalo primjenjivati u slijedećim slučajevima:
  - za atribute prema kojima se obavlja spajanje relacija
  - za atribute koji se često koriste za postavljanje uslova selekcije
  - za atribute prema kojima se često obavlja grupisanje ili sortiranje

Upotreba indeksa

- Na slici je prikazana (strogo logički gledano!) struktura dva B-stabla koja su izgrađena nad istom relacijom.
- Sistem može efikasno pristupati podacima prema ključu pretrage koji sadrži atribut **ime_stud** ili atribut **mbr_stud**. Ako ključ pretrage sadrži atribut **prosj_ocj**, pretraga može biti jedino sekvencijalna.



Upotreba indeksa

- Prilikom kreiranja indeksa treba voditi računa i o nekim njihovim negativnim aspektima, te ih treba koristiti samo tamo gdje je njihova upotreba opravdana, jer:
  - indeksi zauzimaju značajan prostor
  - ažuriranje vrijednosti atributa nad kojima je izgrađen indeks traje znatno duže nego ažuriranje vrijednosti nad kojima nema indeksa
- Indekse ne bi trebalo primjenjivati ukoliko:
  - vrijednosti atributa za koji se gradi indeks imaju relativno mali broj različitih vrijednosti (npr. spol_osobe sa dozvoljenim vrijednostima M, Ž u relaciji s 30 000 n-torki)
  - u relaciji predstoji veliki broj upisa, izmjena ili brisanja n-torki. Preporučljivo je u takvim slučajevima postojeće indekse izbrisati, te ih ponovo izgraditi tek nakon obavljenih promjena nad podacima
  - relacija sadrži vrlo mali broj n-torki (npr. do stotinu). U takvim slučajevima sistem lakše pristupa sekvencijalnom pretragom, nego prolaskom kroz strukturu B-stabla

INSERT naredba

- Naredba je namijenjena za upis jedne ili više n-torki u relaciju. Opšti oblik INSERT naredbe definisan je slijedećom sintaksom:

```
INSERT INTO table_name ili view_name [(column_name)]
VALUES clause ili SELECT statement
```

- Lista s nazivima atributa (*column name*) navodi se u slučaju kada je redoslijed vrijednosti koje se upisuju u relaciju različit od redoslijeda atributa u relaciji ili se ne namjeravaju navesti sve vrijednosti za n-torku.
- Oblikom INSERT naredbe s *VALUES clause* upisuje se jedna n-torka, dok se drugim oblikom (korištenjem *SELECT statement*) upisuju sve n-torke koje su dobivene kao rezultat navedene SELECT naredbe.
- SELECT naredba kojom se određuju n-torke za upis u relaciju ne smije u svom FROM dijelu sadržavati ime relacije u koju se podaci upisuju. Također, SELECT naredba ne smije sadržavati *ORDER BY clause*

INSERT naredba

```
CREATE TABLE mjestoStanovanja(
   pbr             INTEGER
  ,nazMjesto       CHAR(40)
);
```

- Primjeri:

```
INSERT INTO mjestoStanovanja VALUES (75000, 'Tuzla');
INSERT INTO mjestoStanovanja (nazMjesto, pbr) VALUES
('Tuzla', 75000);
INSERT INTO mjestoStanovanja (pbr) VALUES (75000);
```

- U prethodnoj naredbi će za naziv mjesta biti postavljen NULL. Ukoliko bi uz atribut `nazMjesto` prilikom definicije relacije bila definirana default vrijednost, za naziv mjesta bi bila postavljena ta default vrijednost, a ne NULL.

```
INSERT INTO mjestoStanovanja
   SELECT DISTINCT mjesto.pbr, mjesto.nazmjesto
      FROM mjesto INNER JOIN stud
      ON mjesto.pbr = stud.pbrStan
```

INSERT naredba i SERIAL tip podatka

- Ako se za vrijednost atributa tipa SERIAL pomoću INSERT naredbe upiše vrijednost 0, SUBP samostalno određuje serijski broj (prvi slijedeći nakon najveće do tada upisane vrijednosti, pri čemu se kao najveća vrijednost uzima najveća do tada upisana vrijednost, bez obzira je li n-torka s tom vrijednosti u međuvremenu obrisana).
- Ukoliko se upiše broj različit od 0, u relaciju će se upisati unesena vrijednost (neće biti generisan serijski broj). SERIAL tip podatka za atribut ne implicira da je taj atribut ključ relacije.

```
CREATE TABLE stavka(
      RbrStavke         SERIAL
      ,nazivStavke      CHAR(20)
);
INSERT INTO stavka VALUES (0, '1. stavka');    -> 1     1. stavka
INSERT INTO stavka VALUES (0, '2. stavka');    -> 2     2. stavka
INSERT INTO stavka VALUES (200, '3. stavka'); -> 200   3. stavka
INSERT INTO stavka VALUES (0, '4. stavka');    -> 201   4. stavka
INSERT INTO stavka VALUES (100, '5. stavka'); -> 100   5. stavka
INSERT INTO stavka VALUES (0, '6. stavka');    -> 202   6. stavka
INSERT INTO stavka VALUES (201, '7. stavka'); -> Duplicate entry '201'for
                                                 key 'rbrStavke'
```

DELETE naredba

- Naredbom se briše jedna ili više n-torki iz relacije. Opći oblik naredbe je:

```
DELETE FROM table name ili view name
    WHERE condition
```

- *Condition* određuje koje n-torke će biti obrisane.
- Ukoliko se uslov ne navede, sve n-torke iz relacije će biti obrisane.
- Uslov može biti sastavljen kao i u WHERE dijelu SELECT naredbe, ali ne smije sadržavati podupite koji bi u svom FROM dijelu sadržavali relaciju iz koje se zapisi brišu.
- **Primjer:** Obrisati sve n-torke iz relacije **zupanija**, čije se šifre ne koriste u relaciji **mjesto**

```
DELETE FROM zupanija
    WHERE sifZupanija NOT IN
      (SELECT sifZupanija FROM mjesto)
```

UPDATE naredba

- SQL naredba za izmjenu jedne ili više vrijednosti atributa u jednoj ili više n-torki relacije. Pomoću slijedećih primjera ilustrovani su karakteristični načini pridruživanja vrijednosti atributima relacije.
- **Primjer:** ažurira se relacija s imenom *table*. Atributu *atr1* pridružuje se vrijednost izraza *expr1*, a atributu *atr2* vrijednost izraza *expr2*. Atributu *atr3* pridružuje se vrijednost dobivena pomoću SELECT naredbe. Ažuriraju se one n-torke relacije table koje zadovoljavaju navedeni *condition*.

```
UPDATE table SET atr1 = expr1,
                 atr2 = expr2,
                 atr3 = (SELECT expr3 FROM
                         ... WHERE ...)
    WHERE condition
```

UPDATE naredba

- **Primjer:** ažurira se relacija s imenom *table*. Atributu *atr4* pridružuje se vrijednost izraza *expr4*, a atributu *atr5* vrijednost dobivena SELECT naredbom. Atributima *atr6* i *atr7* pridružuju se vrijednosti dobivene jednom SELECT naredbom. Ažuriraju se one n-torke relacije table koje zadovoljavaju navedeni *condition*.

```
UPDATE table SET (atr4, atr5) = (expr4, (SELECT expr5 FROM ...
                                          WHERE ...)),
                (atr6, atr7) = ((SELECT expr6, expr7 FROM ...
                                 WHERE ...))
   WHERE condition
```

- **Primjer:** ažurira se relacija s imenom *table*. U ovom primjeru se umjesto taksativnog navođenja imena svih atributa relacije koristi zvjezdica. Značenje naredbe je: vrijednosti atributa relacije *table* postaviti na vrijednosti koje su navedene na desnoj strani znaka =. Redoslijed pridruživanja vrijednosti je jednak redoslijedu atributa prilikom definiranja relacije *table*.

```
UPDATE table SET * = (expr1, expr2, (SELECT expr3 FROM ...
                                     WHERE ...),
                      expr4, (SELECT expr5 FROM ... WHERE ...),
                     (SELECT expr6, expr7 FROM ... WHERE ...))
   WHERE condition
```

UPDATE naredba

- SELECT naredba kojom se određuje nova vrijednost smije vratiti samo jednu n-torku.
- U listi za selekciju treba biti onoliko atributa ili izraza koliko je potrebno da bi se odgovarajući broj novih vrijednosti pridružio odgovarajućem broju atributa relacije koja se ažurira.
- Pomoću *Condition* određuje se koje će n-torke biti ažurirane.
- Ukoliko se uslov ne navede, bit će ažurirane sve n-torke iz relacije.
- WHERE uslov može sadržavati podupite, ali oni ne smiju u svom FROM dijelu sadržavati relaciju čije se n-torke ažuriraju, što znači da se ne smije obavljati upit nad relacijom čiji se zapisi mijenjaju.
- U MySQL-u je podržan samo prvi način upotrebe UPDATE naredbe (izmjene pojedinih atributa razdvojene zarezom)

UPDATE naredba

- **Primjer:** Svim n-torkama u relaciji **mjesto**, čiji je poštanski broj veći od 30000, nazive mjesta promijeniti tako da se ispred postojećeg naziva doda znakovna konstanta 'GRAD-', a poštanski broj uvećati za 10. prikazana su tri oblika rješenja:

```
UPDATE mjesto SET nazMjesto = CONCAT('GRAD-`,nazMjesto),
                  pbr = pbr + 10
   WHERE pbr > 30000

UPDATE mjesto SET * = (pbr + 10, CONCAT('GRAD-`, nazMjesto),
                       sifZupanija)
   WHERE pbr > 30000

UPDATE mjesto SET (nazMjesto, pbr) = (CONCAT('GRAD-`,
                                      nazMjesto),  pbr + 10)
   WHERE pbr > 30000
```

UPDATE naredba

- **Primjer:** Svim nastavnicima koji su pozitivno ocijenili više od jednog studenta, smanjiti koeficijent za onoliko postotaka koliko iznosi prosjek pozitivnih ocjena kojima je taj nastavnik ocijenio sve svoje ispite

```
UPDATE nasT SET koef = koef * (1 - 0.01*
        (SELECT AVG(ocjena) FROM ispit
           WHERE ocjena > 1
           AND ispit.sifnastavnik =   nasT.sifnastavnik))
   WHERE sifnastavnik IN (
       SELECT sifnastavnik FROM ispit
         WHERE ocjena > 1
         GROUP BY sifnastavnik
         HAVING COUNT (ocjena) > 1);
```

UPDATE naredba

- U slijedećem primjeru pokazano je kako se u relaciji mjesto, stari poštanski brojevi mogu zamijeniti novim, uz pomoć vrijednosti iz tablice za konverziju.

Mjesto

Pbr	**nazMjesto**
41000	ZAGREB
51400	PAZIN
52000	PULA
54000	OSIJEK

Konverzija

stariPbr	**noviPbrnovi**	**NazMjesto**
41000	10000	Zagreb
51400	52000	Pazin
52000	52100	Pula
54000	31000	Osijek

- Zamjena starih poštanskih brojeva novim vrijednostima

```
UPDATE mjesto SET pbr = (SELECT noviPbr FROM konverzija
                         WHERE stariPbr = mjesto.pbr)
```

Primjeri

- Kreirana je relacija **ispiti(sifPred, nazPred, polozeno, prosjek)** koja sadrži podatke o predmetima, broju položenih ispita i prosječnoj ocjeni položenih ispita. Relaciju **ispiti** napuniti podacima o predmetima na kojima je prosječna ocjena položenih ispita veća od prosječne ocjene položenih ispita ostalih predmeta sa iste organizacione jedinice.

```
INSERT INTO ispiti
SELECT pred.sifPred, nazPred, COUNT(*), AVG(ocjena)
FROM ispit INNER JOIN pred
  ON ispit.sifPred = pred.sifPred
  WHERE ocjena > 1
  GROUP BY 1, 2
  HAVING AVG(ocjena) >
        (SELECT AVG(ocjena) FROM ispit ispitP INNER JOIN pred predP
               ON ispitP.sifPred = predP.sifPred
               WHERE pred.sifOrgjed = predP.sifOrgjed
               AND pred.sifPred <> predP.sifPred
               AND ocjena > 1)
```

Primjeri

- Izbrisati sve podatke o ispitima koje su polagali studenti rođeni u istom mjestu u kojem stanuje nastavnik koji ih je ispitivao.

```
DELETE FROM ispit
WHERE EXISTS (SELECT * FROM stud, nastavnik
  WHERE stud.mbrStud = ispit.mbrStud
  AND nastavnik.sifNastavnik = ispit.sifNastavnik
  AND nastavnik.pbrStan = stud.pbrRod)
```

Primjeri

- Svim predmetima na kojima je položeno manje ispita nego što trenutno ima upisano studenata povećati broj sati sedmično za prosječnu ocjenu svih ispita iz tog predemeta.

```
UPDATE pred SET brojSatiTjedno = brojSatiTjedno
  + (SELECT ROUND(AVG(ocjena),0) FROM ispit
      WHERE ispit.sifPred = pred.sifPred)
WHERE upisanoStud > (SELECT COUNT(*) FROM ispit
      WHERE ispit.sifPred = pred.sifPred
      AND ocjena > 1)
```

Primjeri

- Svu nastavu za predmete koja se ne može održati u rezervisanim dvoranama prebaciti u dvoranu sa najvećim kapacitetom. Zanemariti ograničenje vezano za primarni ključ relacije rezervacija i postojanje više od jedne dvorane sa najvećim kapacitetom.

```
UPDATE rezervacija SET
oznDvorana = (SELECT oznDvorana FROM dvorana
WHERE kapacitet = (SELECT MAX(kapacitet) FROM dvorana))
WHERE sifPred IN (SELECT sifPred FROM pred, dvorana
      WHERE pred.sifPred = rezervacija.sifPred
      AND rezervacija.oznDvorana = dvorana.oznDvorana
      AND upisanoStud > kapacitet)
```