Predavanje III.

SQL DDL II

BAZE PODATAKA II

doc. dr. sc. Goran Oreški
Fakultet informatike,
Sveučilište Jurja Dobrile, Pula

Sadržaj

- ponavljanje prethodnih predavanja
  - ograničenja tablice
    - primarnog ključa
    - NOT NULL
    - UNIQUE
    - CHECK
    - stranog ključa
  - kršenja ograničenja
- imenovanje ograničenja
- odgođena ograničenja
- datum i vrijeme
- vremenski intervali
- large data objects
- zadane vrijednosti
- korisnički definirani tipovi
- privremene tablice

Ponavljanje

- ograničenja na tablicama (*engl. Integrity Constraints*)
  - skup pravila koja imaju za cilj osiguravanje kvalitete podataka pohranjenih u baze podataka
  - štite od slučajne štete na podacima baze podataka od strane ovlaštenih korisnika
  - primjeri ograničenja:
    - primarnog ključa – dvije n-torke ne mogu imati iste vrijednosti ID-a
    - NOT NULL vrijednosti – atribut ne može pohranjivati NULL vrijednost
    - UNIQUE – dvije n-torke ne mogu imati iste vrijednosti atributa
    - CHECK – definira predikat koji mora biti zadovoljen za svaku n-torku u relaciji
    - stranog ključa – osigurava da vrijednost koja se pojavljuje u jednoj relaciji za određeni skup atributa se ujedno pojavljuje i u drugoj relaciji za određeni skup atributa (referential integrity)

Ponavljanje

- postoji nekoliko načina za kršenje ograničenja stranog ključa
- ako referencirajuća relacija dobije pogrešnu vrijednost stranog ključa operacija jednostavno nije dopuštena
  - ako se u tablicu *predaje* pokušava dodati red koji ima vrijednost *nastavnik_id* od nepoznatog nastavnika (nije evidentiran u *nastavnik* tablici)
  - ako se u tablicu *predaje* pokušava dodati red koji ima vrijednost *kolegij_sifra* od nepoznatog kolegija (nije evidentiran u *kolegij* tablici)
- kada se rade promjene u referenciranoj relaciji
  - ako se iz tablice nastavnik ukloni red koji je referenciran u tablici predaje?
  - mogućnosti?

Imenovanje ograničenja

- ograničenjima se mogu dodjeljivati imena
  - kada se ograničenje prekrši, sustav greškom javlja o kojem se ograničenju radi
  - baza podataka najčešće sama dodijeli ime ako ga korisnik izostavi
- primjer:

```
CREATE TABLE predaje (
 nastavnik_id INT,
 kolegij_sifra CHAR(6),
 CONSTRAINT pre_pk PRIMARY KEY (nastavnik_id, kolegij_sifra),
 CONSTRAINT pre_nast_sk FOREIGN KEY (nastavnik_id) REFERENCES nastavnik
 CONSTRAINT pre_kol_sk FOREIGN KEY (kolegij_sifra) REFERENCES kolegij);
```

- korisno prilikom ukazivanja na specifično ograničenje

Privremena kršenja ograničenja

- provođenje ograničenja (pogotovo složenijih) je vremenski zahtjevno
  - može drastično utjecati na performanse prilikom operacija uvoza velike količine podataka
- za neke operacije je potrebno privremeno prekršiti ograničenje
  - operacije koje se izvode u okviru velike transakcije (skupa naredbi prema kojima se odnosi kao jednoj cjelini)
  - dok traje transakcija, ograničenja se privremeno krše
  - na kraju transakcije, ograničenja se vraćaju
- primjena ograničenja se odgađa (*engl. defer*) za kraj transakcije
  - na kraju transakcije sve ograničenja se provjeravaju prema odgođenim ograničenjima

Odgođena ograničenja

- u SQL ograničenja se mogu definirati kao odgođena
- u deklaraciji ograničenja:
  - `DEFERRABLE` ograničenja se mogu odgoditi na kraj transakcije
  - `NOT DEFERRABLE` ograničenja se ne mogu odgoditi te se uvjek primjenjuju odmah
- za `DEFERRABLE` ograničenja:
  - `INITIALLY IMMEDIATE` primjenjuju se odmah kao zadano ponašanje
  - `INITIALLY DEFERRED` primjenjuju se na kraju transakcije kao zadano ponašanje

Privremeno uklanjanje ograničenja

- da bi se ograničenja odgodila u trenutnoj transakciji:

```
SET CONSTRAINTS c1, c2, ... DEFERRED;
```

  - navedena ograničenja moraju biti `DEFERRABLE`
- ne podržavaju sve baze podataka odgođena ograničenja
  - jedina opcija u tom slučaju je privremeno ukloniti ograničenje te ga naknadno dodati
  - u pravilu se promjena donosi na sve korisnike baze podataka
  - uklanjanje i ponovno dodavanje ograničenja se vrši pomoću `ALTER TABLE` naredbe

Vrijednosti datuma i vremena

- SQL pruža tipove podataka za datum i vrijeme
- `DATE`
  - kalendarski datum, uključuje godinu, mjesec i dan u mjesecu
- `TIME`
  - vrijeme u danu, uključuje sat, minute i sekunde
  - ne uključuje dijelove sekunde, decimalni dio sekunde
- `TIME(P)`
  - isto kao i vrijeme ali uključuje P znamenki koje predstavljaju dio sekunde
  - u pravilu P = [0, 6]

Vrijednosti datuma i vremena

- mogu se uključiti vremenske zone
  - `TIME WITH TIMEZONE`
  - `TIME(P) WITH TIMEZONE`
- `TIMESTAMP`
  - kombinacija vrijednosti datuma i vremena
  - može se definirati `TIMESTAMP(P)`
  - P = 6 je zadano
  - timestamp može uključiti vremensku zonu
    - `TIMESTAMP WITH TIMEZONE`
    - `TIMESTAMP(P) WITH TIMEZONE`

Vrijednosti datuma i vremena

- postoje mnogi nestandardni tipovi podataka
  - `DATETIME` isto ako i `TIMESTAMP` ali P = 0 kao zadano
  - `YEAR` – četveroznamenkasti broj
  - nestandardni tipovi -> neprenosivi tipovi

- specifikacija vremenskih tipova podataka
  - MySQL
  - PostgreSQL
  - SQLServer

Format datuma i vremena

- vrijednosti datuma i vremena prate određeni format
  - navodi se u jednostrukim navodnicima
- primjeri
  - `SELECT TIMESTAMP(NOW());` **`'2019-10-27 15:16:33'`**
  - `SELECT DATE(NOW());` **`'2019-10-27'`**
  - `SELECT TIME(NOW());` **`'15:17:03'`**
- datumi mogu sadržavati neispravne vrijednosti
  - neispravno vrijeme: **`'25:17:03'`**
  - neispravni datum: **`'2019-02-31'`**
  - neki DBMS-ovi mogu pohranjivati neispravne vrijednosti ukoliko to zahtjeva aplikacija

Format datuma i vremena

- većina DBMS-ova podržava mnoge date/time formate
- široko je podržan standard ISO-8601 za date/time format
  - ISO-8601

    `'2019-10-15 18:44:41.289'`
    - godina-mjesec-dan sat:minute:sekunde:milisekunde
    - ponekad je datum odvojen od vremena sa slovom "T"
    - vrijeme koristi 24 satni format
    - vremenske zone se opcionalno mogu dodati na kraj
  - drugi formati:
    - `'October 27, 2019 04:14:46 PM'`
    - `'27-Oct-2019 16:14:46.113'`

Trenutno vrijeme

- nekoliko funkcija vraća trenutno vrijeme i datum u SQL-u
  - `CURRENT_DATE()`
  - `CURRENT_TIME()`
  - `CURRENT_TIMESTAMP()`
    - uključuju vremensku zonu
  - `LOCALTIME()`
  - `LOCALTIMESTAMP()`
    - ne uključuju vremensku zonu
- i mnoge druge funkcije, kao npr. ranije korištena `NOW()`
  - nestandardna, ali široko podržana

Komponente datuma i vremena

- vrijednosti datuma i vremena nisu nedjeljive (engl. atomic)
  - djeljive vrijednosti ne bi trebale biti uključene u relacijski model
  - u stvarnosti, mnogi SQL tipovi nisu nedjeljivi
- SQL ima funkciju kojom se dohvaćaju komponente tipova podataka datum i vrijeme

  ```
  EXTRACT (field FROM value)
  ```

  - mogu se dohvatiti:
    - `YEAR, MONTH, DAY, HOUR, MINUTE, SECOND`
    - `TIMZEONE_HOUR, TIMEZONE_MINUTE`
  - druge nestandardne komponente
    - tjedan u godini, dan u godini, dan u tjednu, kvartal, stoljeće,…

Operacije na datumu

- podaci o predanim zadaćama

```
CREATE TABLE predana_zadaca{
    predana_zadaca_id INTEGER,
    student_id INTEGER NOT NULL,
    zadaca_id INTEGER NOT NULL,
    datum_predaje TIMESTAMP NOT NULL,
    …}
```

- *ispišite mjesec svih predanih zadaća*

```
SELECT predane_zadace_id, EXTRACT(MONTH FROM datum_predaje) AS
mjesec_predaje FROM predana_zadaca;
```

Umetanje datuma

- format datuma zadanog u bazi se može razlikovati od formata koji se unosi
  - sintaksa se može razlikovati ovisno o sustavu (prikaz MySQL)
- npr.

```
INSERT INTO zadaca (zadaca_id, naziv, datum_predaje)
   VALUES ('1', 'SPI', '01.10.2019.');
```

```
+-----------+------+--------------+------+
| zadaca_id | naziv | datum_predaje | opis |
+-----------+------+--------------+------+
|         1 | SPI   | 2001-10-20    | NULL |
+-----------+------+--------------+------+
```

```
INSERT INTO zadaca (zadaca_id, naziv, datum_predaje)
   VALUES ('1', 'SPI', STR_TO_DATE('01.10.2019.', '%d.%m.%Y.'));
```

```
+-----------+------+--------------+------+
| zadaca_id | naziv | datum_predaje | opis |
+-----------+------+--------------+------+
|         1 | SPI   | 2019-10-01    | NULL |
+-----------+------+--------------+------+
```

Vremenski intervali

- tip podatka kojim se pohranjuju vremenski intervali u SQL-u je `INTERVAL`
  - podržava operacije na datumima i vremenima
  - također podržava i veću preciznost `INTERVAL(P)`
- ako su ***x*** i ***y*** datumi tada ***x – y*** stvara `INTERVAL`
- ako je ***i*** `INTERVAL` vrijednost ***x + i*** ili ***x – i*** rezultira datumom
- ključna riječ `INTERVAL` se može koristiti i da bi se odredio fiksni interval
  - `INTERVAL 1 WEEK`
  - `+ INTERVAL 1 WEEK`

Primjer

- shema tablice zadaća

```
CREATE TABLE unipu.zadaca (
    zadaca_id INT NOT NULL,
    naziv VARCHAR(45) NOT NULL,
    datum_predaje DATE NOT NULL,
    opis VARCHAR(200),
    PRIMARY KEY (zadaca_id));
```

- *dohvatite podatke za nadolazeće zadaće*

```
SELECT * FROM zadaca
    WHERE datum_predaje >= CURRENT_DATE() AND
    datum_predaje <= (CURRENT_DATE() + INTERVAL 2 WEEK);
```

Primjer

- isti upit se može napisati pomoću BETWEEN sintakse

```
SELECT * FROM zadaca
 WHERE datum_predaje BETWEEN
 CURRENT_DATE() AND (CURRENT_DATE()+INTERVAL 2 WEEK);
```

- funkcije za trenutni datum i vrijeme se primjenjuju samo jednom tijekom upita
  - upit će koristiti samo jednu vrijednost za CURRENT_TIME() iako se izvodi kroz duži period vremena
- funkcije se mogu koristiti u `INSERT` naredbi

"Large Object" tipovi podataka

- SQL `CHAR(N)` i `VARCHAR(N)` tipovi podataka imaju ograničenu veličinu
  - za `CHAR(N)`, uobičajeno N < 256
  - za `VARCHAR(N)`, uobičajeno N < 65536
- `BLOB` i `CLOB` podržavaju veće veličine podataka
  - "LOB" = large object
    - B – binary, C – character
  - koristi se za pohranu slika, dokumenata i sl.
  - podrška varira ovisno o sustavu
  - `TEXT` tip je također čest
    - veliki tekstualni podaci, npr. GB tekst podataka
    - otprilike odgovara CLOB tipu

Primjer sheme

- shema tablice nastavnik

```
CREATE TABLE nastavnik (
    id                INT PRIMARY KEY,
    nastavnik_sifra CHAR(10) NOT NULL,
    ime               VARCHAR(30) NOT NULL,
    prezime           VARCHAR(30) NOT NULL,
    slika             BLOB,
    zvanje            VARCHAR(40) NOT NULL,
    primanja          NUMERIC(10,2) NOT NULL,
    zivotopis         CLOB NOT NULL,
    UNIQUE(nastavnik_sifra)
    );
```

Large Objects dodatak

- podrška za "velike objekte" je u pravilu zamišljena za objekte manje veličine !?
  - ne veće od nekoliko desetaka KB-ta
  - nekoliko MB je gornja granica
- najzahtjevniji dio za bazu podatka je premještanje velikih objekata u i iz baze podataka
- relacijske baze podataka ne pohranjuju takve tipove podataka, velike objekte, vrlo učinkovito
  - drugi načini pohrane

Large Objects dodatak

- za pohranu objekata koji su veći od ~100 KB bolje je koristiti datotečni sustav
  - za to je namijenjen
  - u bazu se pohranjuje putanja do datoteke
- za manje objekte koji se često koriste, pohranjivanje na datotečni sustav može rasteretiti bazu podataka
  - npr. korisnička ikona za web stranicu
  - omogućava web serveru da je direktno dohvaća s datotečnog sustava, to je tip operacije koju on može brže izvesti
- neki DBMS-ovi imaju posebnu podršku za učitavanje i manipuliranje velikim objektima - prenosivost?

Zadane vrijednosti

- za stupac se može definirati zadana vrijednost

  colname type `DEFAULT` *expr*

  - može se navesti točna vrijednost

  ```
  broj_ectsa INT DEFAULT 5
  ```

  - ili se može navesti izraz

  ```
  datum_predaje TIMESTAMP DEFAULT NOW()
  ```

- ako se zadana vrijednost ne navede, DBMS se koristiti `NULL`
- utječe na `INSERT` naredbu
  - stupci sa zadanom vrijednost ne moraju biti navedeni
  - stupci bez zadane vrijednosti moraju biti navedeni u trenutku umetanja novog zapisa

Korisnički definirani tipovi

- SQL podržava dva oblika korisnički definiranih tipova podataka
  - ***distinct types***
  - ***structured data types***
- različite domene na konceptualnoj razini a ne fizičkoj i potreba za prepoznavanjem dodjele pogrešne domene tipu
  - npr. ime i naziv_odjela ili različite valute (funte i euri)
  - dodjela imena odjela nekoj osobi je u pravilu greška iako oboje imaju dodijeljen isti tip podataka
- dobar sustav ne bi trebao dozvoliti takve pogreške i u tu svrhu se koriste *distinct types*

Korisnički definirani tipovi

- za kreiranje novog tipa podataka se koriste ključne riječi *CREATE TYPE*

```
CREATE TYPE kune AS NUMERIC(12,2);
CREATE TYPE euri AS NUMERIC(12,2);
```

  - definira se na temelju *built-in* tipa
  - iako je fizička reprezentacija oba tipa jednaka, njihova usporedba ili međusobna dodjela nije moguća
    - takve operacije su najčešće rezultat programerske greške
    - deklariranjem različitih tipova mogu se spriječiti slične greške
  - potrebno je koristiti CAST operator
  - novi tip se može koristiti prilikom deklariranje tablice kao i bilo koji ugrađeni tip podataka

Korisnički definirani tipovi

```
CREATE TABLE fakultet
 (fakultet_naziv VARCHAR(20),
  zgrada VARCHAR(15),
  budzet KUNE);
```

- izraz kao (*fakultet.budzet + 150000.50*) nije valjan zbog različitih tipova
  - potrebno je koristiti CAST operator

```
CAST(fakultet.budzet TO NUMERIC(12,2))
```

  - te cast nazad u kune ukoliko na taj način pohranjujemo podataka
- mogu se koristiti `ALTER TYPE` i `DROP TYPE` naredbe
- implementacija varira, MySQL ne podržava KDT

Domene

- prije nego što su KDT dodani u SQL (SQL:1999) sa sličnu namjenu su korištene domene
  - ključna riječ `DOMAIN`
  - vrlo male razlike u odnosu na KDT, glavna razlika je da domene mogu zadavati ograničenja na osnovni tip

```
CREATE DOMAIN d_kune AS NUMERIC(12,2) NOT NULL;
```

- *d_kune* se kao tip može koristiti identično kao i *distinct types*
- glavne razlike
  - mogu se zadavati ograničenja npr. NOT NULL
  - dok je osnovni tip jednak, mogu se dodjeljivati vrijednosti jedne domene drugoj

Domene

- primjer

```
CREATE DOMAIN mjesecna_placa NUMERIC(8,2)
   CONSTRAINT mjesecna_placa_test CHECK(VALUE >= 3200.00);
```

- CHECK ograničenje omogućava postavljanje predikata koji mora zadovoljiti svaki atribut iz pripadajuće domene
- u primjeru, vrijednost koja se unosi za neki atribut tipa *mjesecna_placa* mora biti veći od 3200 kuna, čime se osigurava minimalna mjesečna plaća
- imenovanje ograničenja nije obvezno

Serial primarni ključ

- mnoge baze podataka pružaju dodatnu podršku za primarne ključeve tipa INT
  - baza može generirati jedinstvene vrijednosti za primarni ključ
- primjer:
  - MySQL i PostgreSQL

```
CREATE TABLE nastavnik (
    nastavnik_id SERIAL PRIMARY KEY
    …
```

  - Microsoft SQLServer

```
CREATE TABLE nastavnik (
    nastavnik_id INT IDENTITY PRIMARY KEY
    …
```

Shema nastavnika ponovno

```
CREATE TABLE nastavnik (
   id                 SERIAL PRIMARY KEY,
   nastavnik_sifra CHAR(10) NOT NULL UNIQUE,
   ime                VARCHAR(30) NOT NULL,
   prezime            VARCHAR(30) NOT NULL,
   slika              BLOB,
   zvanje             VARCHAR(40) NOT NULL,
   primanja           NUMERIC(10,2) NOT NULL,
   datum_zaposljavanja DATE DEFAULT CURRENT_DATE(),
   zivotopis          CLOB NOT NULL);
```

- svaki novi nastavnik dobiva dodijeljen jedinstveni ID
- datum zapošljavanja se postavalja CURRENT_DATE()

Izmjena sheme tablice

- SQL `ALTER TABLE` omogućava izmjenu sheme tablice
- široki raspon mogućnosti
  - preimenovanje tablice
  - dodavanje ili brisanje ograničenja
  - dodavanje ili brisanje stupaca
  - izmjena tipova stupaca
  - izmjena zadanih vrijednosti stupaca
- vrlo korisno za migraciju sheme na novu verziju
  - migracija mora biti vrlo oprezno dizajnirana
- postoje male razlike u podrški različitih DBMS-ova

Primjeri izmjene

- preimenovanje tablice nastavnik

```
ALTER TABLE nastavnik
   RENAME TO profesor;
```

- brisanje stupca u kojem se pohranjuje slika

```
ALTER TABLE nastavnik
   DROP COLUMN slika;
```

- dodavanje ograničenja na primanja stupac u tablici nastavnik

```
ALTER TABLE nastavnik
   ADD CHECK (primanja > 3200);
```

Privremene tablice

- *engl. temporary tables*
- ponekad je potrebno generirati i pohraniti relacije privremeno
  - složene operacije implementirane pomoću više upita
  - operacija dodjele u relacijskoj algebri: ←
- SQL pruža privremene tablice za takve slučajeve
  - sadržaj tablica je povezan sa sesijom klijenta
  - klijenti međusobno ne mogu pristupati privremenim tablicama
- SQL standard definira globalne privremene tablice
    - globalne privremene tablice imaju globalno ime i shemu
    - sadržaj privremenih tablica je privatan za svakog korisnika
    - kada se klijent odjavi, sadržaj se čisti
    - kada se završe sve konekcije koje referenciraju tablicu ona se briše

Privremene tablice

- mnoge baze podataka pružaju i lokalne privremene tablice
  - shema tablice je lokalna za sesiju klijenta
  - kada se klijent odjavi, tablica se briše
  - različiti klijenti mogu koristiti isto ime za tablicu s različitom shemom
- klijent može i ručno očistiti privremenu tablicu kada se za to javi potreba
  - u slučaju lokalnih privremenih tablice, mogu se obrisati bilo kada tijekom sesije

Privremene tablice sintaksa

- jednostavna varijacija `CREATE TABLE` naredbe
  - dodaje se `TEMPORARY` (ili `GLOBAL TEMPORARY`) naredbi
- primjer:
  - napravite privremenu tablicu za pohranu ukupnog broja studenta prema mjesecu

```
CREATE TEMPORARY TABLE prisustvo_po_mjesecu(
   broj_studenata INT NOT NULL,
   mjesec INT NOT NULL);
```

Primjer privremene tablice

Napraviti shemu relacijske baze koja vodi evidenciju o studentima, predavanjima i njihovoj prisutnosti!
ROK: tjedan dana
Prvo dobro rješenje 1 bod!

- privremena tablica se može popuniti s izračunatim vrijednostima

```
INSERT INTO prisustvo_po_mjesecu
    SELECT EXTRACT (MONTH FROM datum) AS mje, COUNT(*)
    FROM prisustvo GROUP BY mje;
```

  - samo se jednom izvodi računanje upita
  - može se poboljšati učinkovitost velikih operacija i operacija od više koraka
  - privremeni rezultati se čiste nakon završetka sesije
- upiti na privremenoj tablici se pišu jednako kao i na običnoj tablici

```
SELECT broj_studenata, opis_predavanja
    FROM prisustvo_po_mjesecu
    JOIN predavanja USING (mjesec);
```

Korištenje privremene tablice

- privremene tablice mogu značajno poboljšati performanse nekih upita
- pristup:
  - stvoriti privremenu tablicu za pohranu međurezultata koji su korisni ali zahtjevni za izračun
    - ne preporuča se korištenje puno (ijedno) ograničenja jer može usporiti izvođenje
  - popuniti privremenu tablicu pomoću `INSERT` … `SELECT` naredbe
  - korištenje privremene tablice za računanje željenih rezultata
  - privremena tablica završava nakon prekida transakcije ili završetka sesije

Alternativne sintakse za privremene tablice

- baze podataka često podržavaju alternativnu sintaksu za stvaranje i popunjavanje privremenih tablica
- česta sintaksa za MySQL, Postgres, Oracle

```
CREATE TEMPORARY TABLE tbl_name AS select_stmt;
```

  - shema tablice odgovara shemi upita
- druga, također česta sintaksa Postgres, SQLServer

```
SELECT … INTO TEMPORARY TABLE …;
```

  - stvara se tablica na temelju sheme SELECT upita i popunjava se podacima dohvaćenim iz upita
- s obje sintakse se mogu stvoriti i obične tablice

Stvaranje tablica - dodatak

- često je potrebno stvoriti tablicu koja ima istu shemu kao i postojeća tablica

  `CREATE TABLE … AS …` sintaksa (ili `CREATE TABLE … LIKE …`)

- spremanje rezultata složenog upita u tablicu (privremenu)
  - primjer

```
CREATE TABLE t1 AS
 (SELECT * FROM nastavnik WHERE zvanje= 'docent') WITH DATA;
```

- `WITH DATA` ujedno popunjava tablicu s podacima
  - u mnogim sustavima se može izostaviti jer je takvo ponašanje zadano

Literatura

- Pročitati
  - [DSC] poglavlje 4.5.
  - Caltech CS121 - 8
  - **Vježbe: [DSC] Poglavlje 4**
- Slijedeće predavanje
  - [DSC] poglavlje 5.2.
  - Caltech CS121 - 9