GEOPROSTORNE BAZE PODATAKA

Materijali za vježbe v1 (11.2017.)

Autor: Adis Hamzić MA geod.

PREDGOVOR

U ovome dokumentu su dati primjeri zadataka sa rješenjima za oblasti koje su obuhvaćene predmetom „Geoprostorne baze podataka“. Pretpostavlja se da su studenti kroz predavanja već upoznati sa materijom pa iz toga razloga teoretska osnova će biti samo kratko spomenuta prije svake grupe zadataka za određenu oblast. Nakon svake oblasti za studente su pripremljeni i zadaci za vježbu kako bi kroz samostalan rad studenti mogli provjeriti koliko dobro su savladali obrađenu oblast.

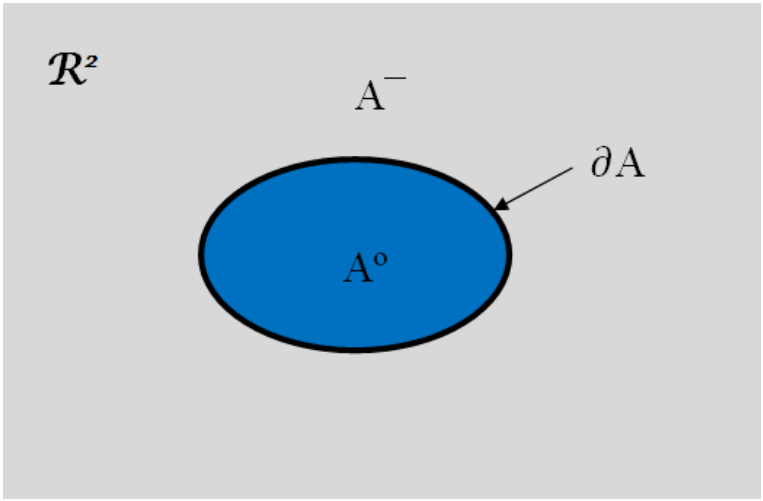
Ukoliko uočite greške u materijalima molimo Vas da javite autoru na sljedeću e-mail adresu: hamzicadis87@gmail.com .

Adis Hamzić MA geod.

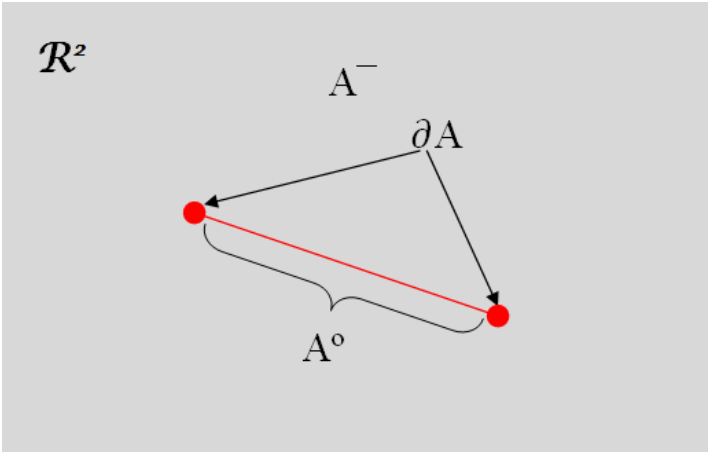
1. Model 9 presjeka

Topologija je grana matematike koja proučava prostorne odnose između geometrijskih objekata. Najvažnija osobina tih odnosa je da se ne mjenjaju prilikom primjene topoloških transformacija (translacija, rotacija, promjena mjerila, rastezanje, gnječenje, i sl.). Dakle, topologija se bavi samo nemetričkim prostorom što znači da metričke vrijednosti (koordinate, dužine, uglovi,...) nisu razmatrane. Za opisivanje toploških odnosa koristi se model 9 presjeka. Topološki odnos između dva objekta određuje se uporedbom unutrašnjosti, granice i vanjštine ta dva objekta.

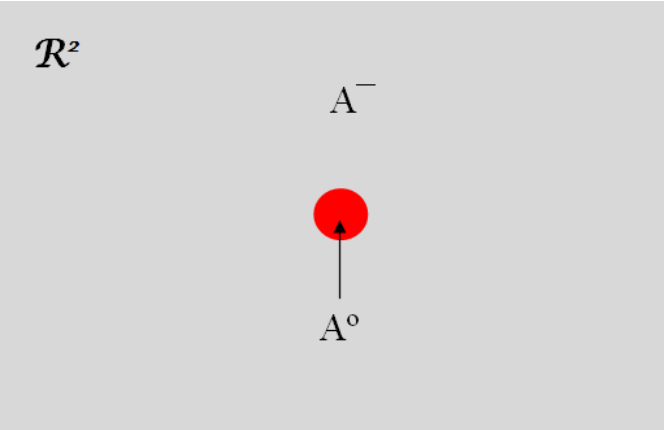
Unutrašnjost, granica i vanjština za poligone, linije i tačke



Slika 1: Unutrašnjost, granica i vanjština poligona



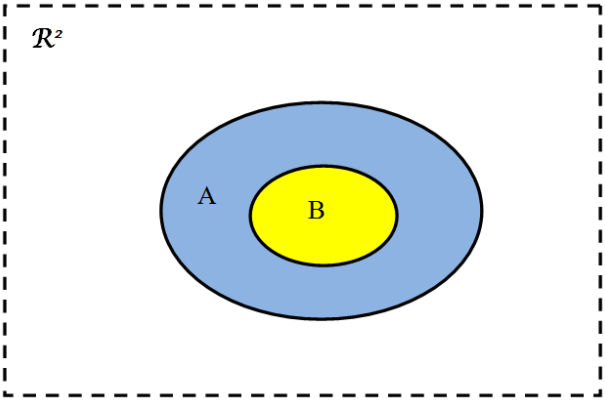
Slika 2: Unutrašnjost, granica i vanjština linije



Slika 3: Unutrašnjost i vanjština tačke

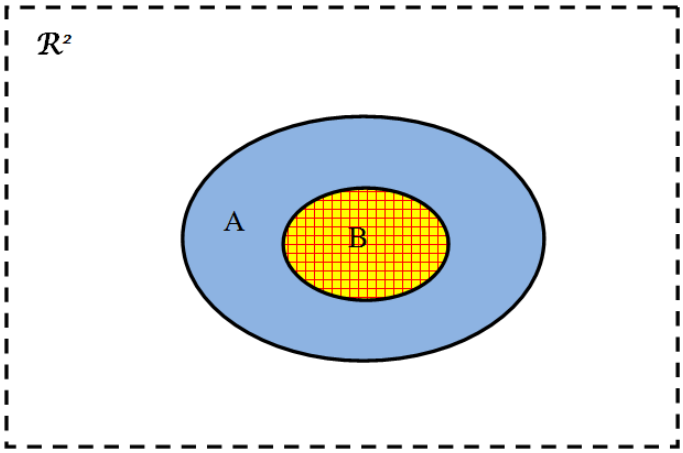
Primjer 1:

Napraviti matricu 9 presjeka za poligone A i B na slici:



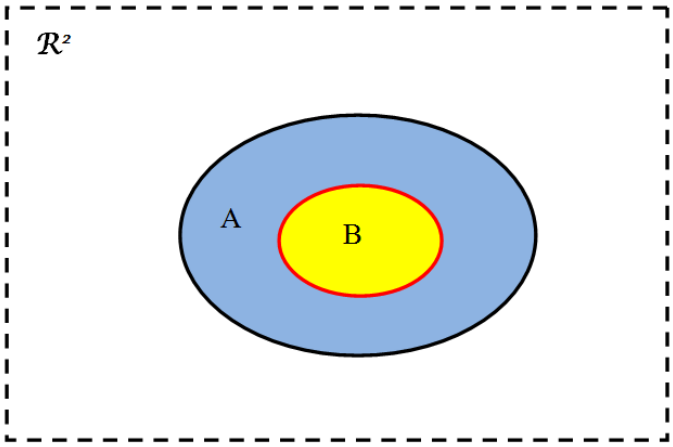
$$R(A,B) = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{pmatrix}$$

1. $A^o \cap B^o$



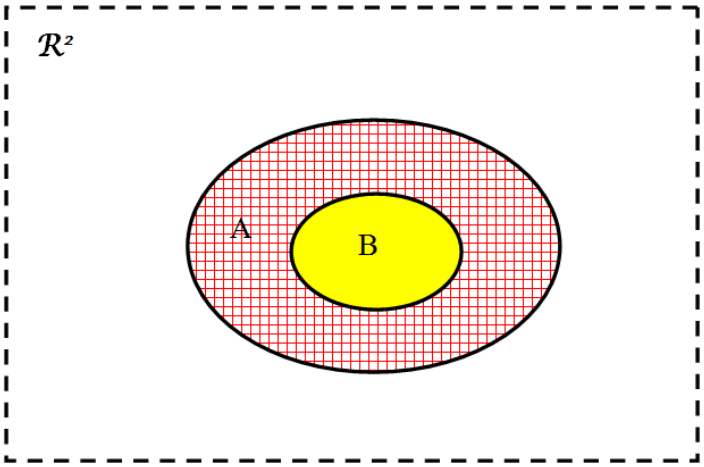
$$R(A,B) = \begin{pmatrix} \boldsymbol{A^o \cap B^o} & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} \neg\emptyset & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{pmatrix}$$

2. $A^0 \cap \partial B$



$$R(A,B) = \begin{pmatrix} A^o \cap B^o & \boldsymbol{A^0 \cap \partial B} & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} \neg\emptyset & \neg\emptyset & ? \\ ? & ? & ? \\ ? & ? & ? \end{pmatrix}$$

3. $A^o \cap B^-$



$$R(A,B) = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & \boldsymbol{A^o \cap B^-} \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ ? & ? & ? \\ ? & ? & ? \end{pmatrix}$$

4. $\partial A \cap B^o$

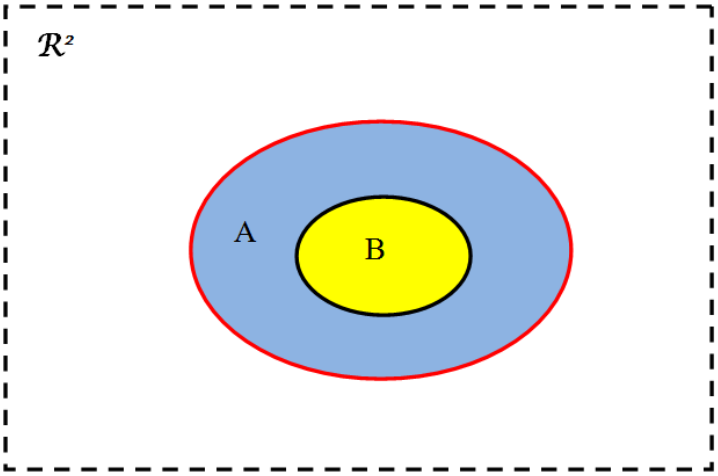
$$\boldsymbol{\partial A \cap B^o = \emptyset}$$

5. $\partial A \cap \partial B$

$$\boldsymbol{\partial A \cap \partial B = \emptyset}$$

$$R(A,B) = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \boldsymbol{\partial A \cap B^o} & \boldsymbol{\partial A \cap \partial B} & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset & ? \\ ? & ? & ? \end{pmatrix}$$

6. $\partial A \cap B^-$



$$R(A,B) = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \boldsymbol{\partial A \cap B^-} \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \\ ? & ? & ? \end{pmatrix}$$

7. $A^- \cap B^o$

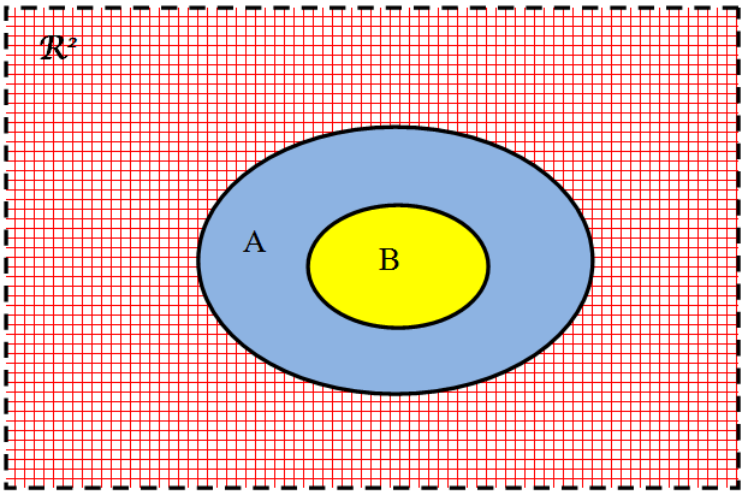
$$\boldsymbol{A^- \cap B^o = \emptyset}$$

8. $A^- \cap \partial B$

$$\boldsymbol{A^- \cap \partial B = \emptyset}$$

$$R(A,B) = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ \boldsymbol{A^- \cap B^o} & \boldsymbol{A^- \cap \partial B} & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \\ \emptyset & \emptyset & ? \end{pmatrix}$$

9. $A^- \cap B^-$

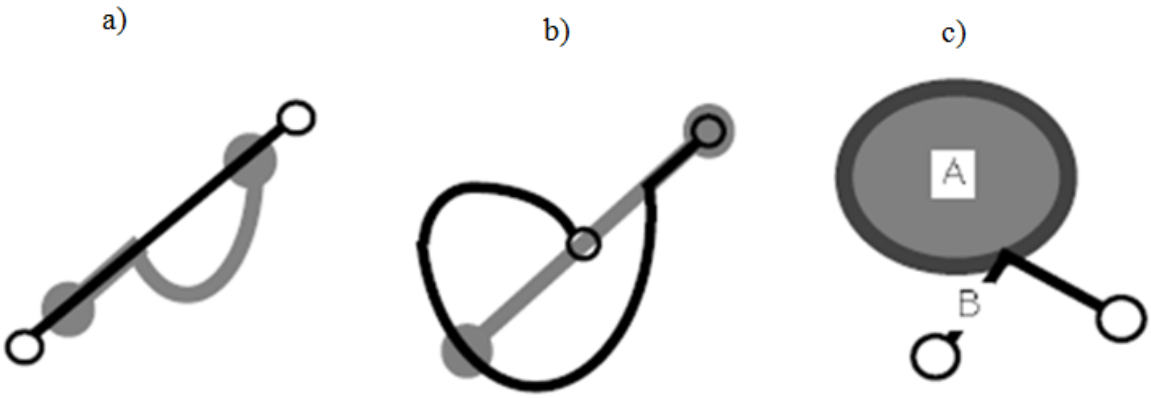


$$R(A,B) = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & \boldsymbol{A^- \cap B^-} \end{pmatrix} = \begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \end{pmatrix}$$

Zadaci za vježbu:

1. Ispiši matrice 9 presjeka za sljedeće slučajeve:

   (Napomena: Crna linija je A, a siva B!)



2. Nacrtaj neku od mogućih kombinacija dvije geometrije kao primjer sljedećih matrica 9 presjeka:

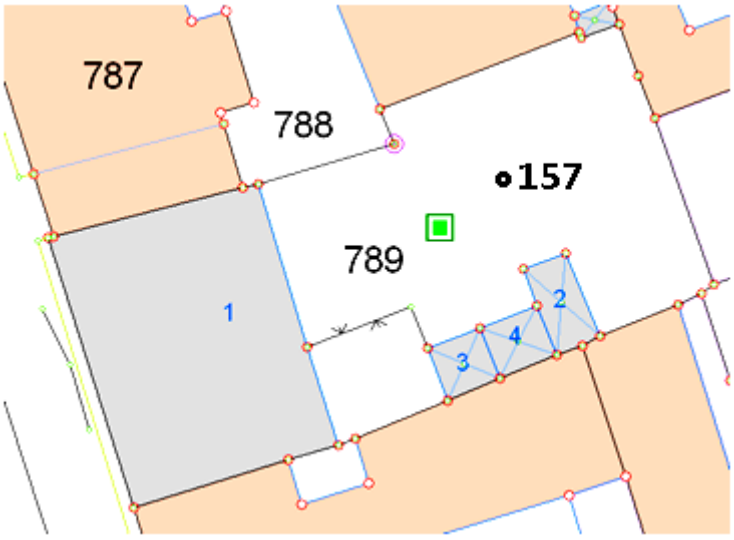
a) $$R(A,B) = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} \emptyset & \emptyset & \neg\emptyset \\ \emptyset & \neg\emptyset & \emptyset \\ \neg\emptyset & \emptyset & \neg\emptyset \end{pmatrix}$$

b) $$R(A,B) = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \emptyset & \neg\emptyset \\ \neg\emptyset & \emptyset & \neg\emptyset \end{pmatrix}$$

c) $$R(A,B) = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} \neg\emptyset & \emptyset & \neg\emptyset \\ \emptyset & \neg\emptyset & \emptyset \\ \neg\emptyset & \emptyset & \neg\emptyset \end{pmatrix}$$

3. Nacrtaj sve topološke relacije između tačke i linije, a zatim ispiši odgovarajuće matrice za svaki od slučajeva!

4. Na parceli 789 se nalaze objekti 1,2,3,4 i poligonska tačka 157. Ispisati matricu 9 presjeka koja opisuje odnos parcele 789 i poligonske tačke 157.



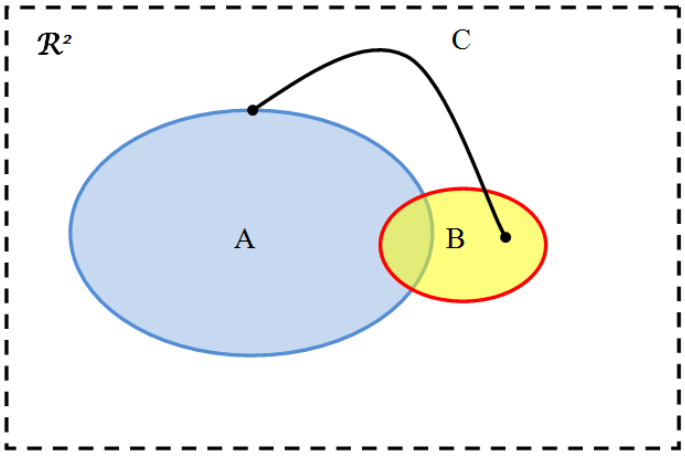
2. Prošireni model 9 presjeka

Nedostatak modela 9 presjeka (9-IM) je što razlikuje samo prazne i neprazne presjeke između granice i unutrašnjosti geometrijskih objekata pa se zbog toga uvodi dimenzijski prošireni model 9 presjeka.

Dimenzijski prošireni model 9-presjeka (DE-9IM) dobije se jednostavnim proširenjem svakog presjeka u 9-IM modelu njegovom dimenzijom:

$$DE9I = \begin{pmatrix} \dim(A^o \cap B^o) & \dim(A^0 \cap \partial B) & \dim(A^o \cap B^-) \\ \dim(\partial A \cap B^o) & \dim(\partial A \cap \partial B) & \dim(\partial A \cap B^-) \\ \dim(A^- \cap B^o) & \dim(A^- \cap \partial B) & \dim(A^- \cap B^-) \end{pmatrix}$$

Primjer: Napisati dimenzijske matrice 9 presjeka za presjeke poligona i presjeke poligona sa linijom na slici ispod!



Rješenje:

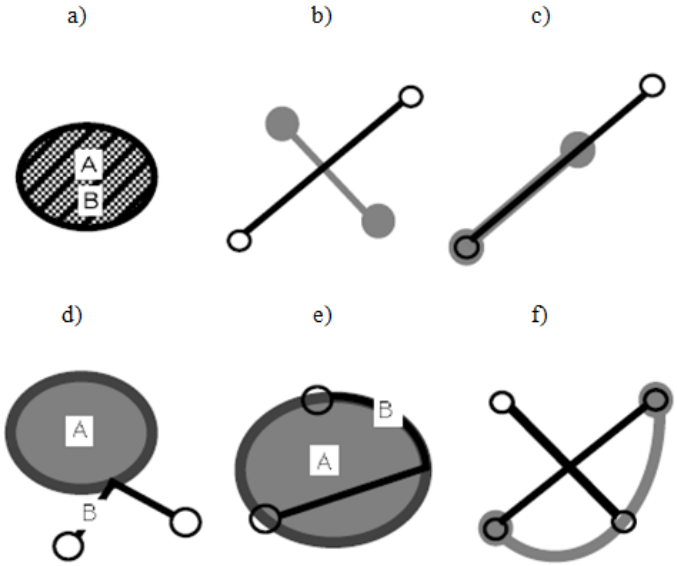
$$R(A,B)_{dim} = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} 2 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 2 \end{pmatrix}$$

$$R(A,C)_{dim} = \begin{pmatrix} A^o \cap C^o & A^0 \cap \partial C & A^o \cap C^- \\ \partial A \cap C^o & \partial A \cap \partial C & \partial A \cap C^- \\ A^- \cap C^o & A^- \cap \partial C & A^- \cap C^- \end{pmatrix} = \begin{pmatrix} -1 & -1 & 2 \\ -1 & 0 & 1 \\ 1 & 0 & 2 \end{pmatrix}$$

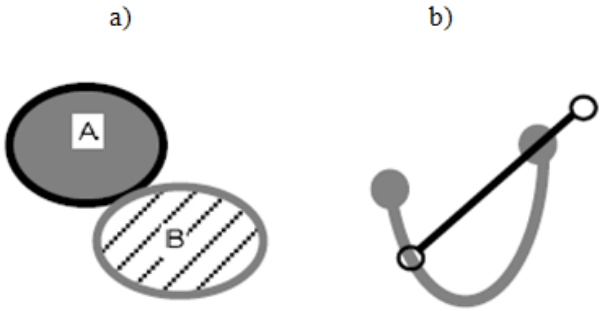
$$R(B,C)_{dim} = \begin{pmatrix} B^o \cap C^o & B^0 \cap \partial C & B^o \cap C^- \\ \partial B \cap C^o & \partial B \cap \partial C & \partial B \cap C^- \\ B^- \cap C^o & B^- \cap \partial C & B^- \cap C^- \end{pmatrix} = \begin{pmatrix} 1 & 0 & 2 \\ 0 & -1 & 1 \\ 1 & 0 & 2 \end{pmatrix}$$

Zadaci za vježbu:

1. Napiši dimenzijske matrice 9 presjeka za sljedeće slučajeve:
   (Napomena: Crna linija je A, a siva B!)



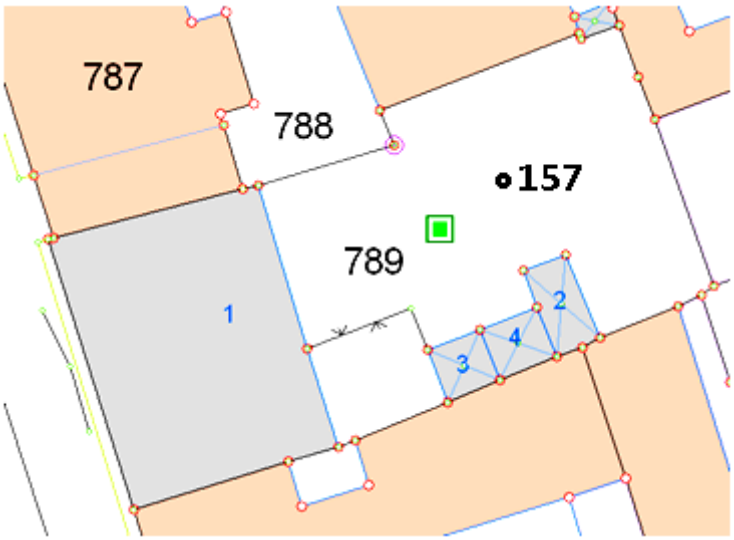
2. Pronađite greške u dimenzijskim matricama 9 presjeka!
   (Napomena: Crna linija je A, a siva B!)



a) $$R(A,B)_{dim} = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} -1 & -1 & 2 \\ -1 & 1 & 1 \\ 2 & 0 & 2 \end{pmatrix}$$

b) $$R(A,B)_{dim} = \begin{pmatrix} A^o \cap B^o & A^0 \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 2 \end{pmatrix}$$

3. Na parceli 789 se nalaze objekti 1,2,3,4 i poligonska tačka 157. Ispisati dimenzijski proširenu matricu 9 presjeka koja opisuje odnos objekata 3 i 4!



3. Jezik za definisanje podataka (DDL)

DDL se koristi za kreiranje, brisanje i modificiranje definicija tabela u bazi podataka. Kreiranje relacijskih šema te dodavanje i brisanje tabela je definisano u DDL komponenti SQL-a. Pretpostavlja se da su studenti upoznati sa tipovima podtaka i naredbama za kreiranje tabela.

Primjer: Kreiranje tabele „Opstine“ (Opštine/Općine/Župe):

Naziv tabele je „Opstine“. Tabela ima 4 atributa i to: ID, Naziv, Povrsina (u kvadratnim kilometrima) i Populacija (br. stanovnika). Identifikacijski broj opštine i br. stanovnika su cjelobrojnog tipa tj. INTEGER. Površina opština, kako je izražena u kvadratnim kilometrima, bit će tipa DOUBLE ili FLOAT (zavisno kako su definisani podaci sa pomičnim zarezom u konkretnoj bazi podataka). Naravno, ako se ne zahtjeva veća preciznost površina može biti i cjelobrojnog tipa (ako se zaokružuje na kvadratni kilometar). Naziv opštine će biti tipa CHAR. Obzirom da u BiH nema opština čiji naziv prelazi 30 znakova u polju naziv ostavit će se prostor za ukupno 30 znakova.

```
CREATE   TABLE Opstine {
         ID   INT,
         Naziv   CHAR (30),
         Povrsina   FLOAT,
         Br_stanovnika   INT}
```

Na same tabele i atribute tabela se mogu stavljati ograničenja. Postoji ukupno 5 ograničenja, to su: UNIQUE KEY, PRIMARY KEY, FOREIGN KEY, NOT NULL i CHECK. Podesimo da je ID opštine primarni ključ i da broj stanovnika mora biti upisan.

```
CREATE   TABLE Opstine {
         ID   INT,
         Naziv   CHAR (30),
         Povrsina   FLOAT,
         Br_stanovnika   INT  NOT NULL
         PRIMARY KEY (ID)}
```

Tabele koje se više ne koriste se mogu ukloniti iz baze podataka pomoću komande **DROP TABLE**. Npr.:

DROP TABLE Opstine;

Za modificiranje relacijske šeme se koristi komanda **ALTER TABLE**, npr. dodavanje novih atributa, modificiranje postojećih atributa, definisanje *default*-ne vrijednosti za novi atribut, brisanje atributa.

Ako u tabeli Opstine želite izbrisati kolonu Populacija to ćete učiniti na sljedeći način:

ALTER TABLE Opstine

DROP Populacija;

Ako u tabeli Opstine želite dodati atribut Pripada kako biste naznačili da li se opština nalazi u Federaciji ili RS-u to možete učiniti na sljdeći način:

ALTER TABLE Opstine

ADD Pripada (CHAR (20));

Primjer:

Napraviti tabelu „Zaposlenik“ koja će imati sljedeće kolone: ime, prezime, JMBG, datum rođenja, trajanje zaposlenja. JMBG je primarni ključ i ne smije biti null vrijednost. Nakon kreiranja tabele u tabelu unijeti sljedeće podatke: Ahmed, Ahmić, 0101988112233. *(napomena: paziti na korištene tipove podataka!)*

CREATE TABLE Zaposlenik {
Ime CHAR (30),
Prezime CHAR (30),
JMBG INTEGER NOT NULL,
Datum rođenja DATE,
Trajanje zaposlenja INTERVAL,
PRIMARY KEY JMBG}

INSERT INTO Zaposlenik (Ime, Prezime, JMBG),

VALUES („Ahmed“, „Ahmić“, 0101988112233)

Zadaci za vježbu:

1. Kreirajte tabelu Gradovi koja će imati sljedeće atribute: ID, Naziv, Populacija, X i Y (X i Y su koordinate geometrijskog centra grada u GK projekciji). Vodite računa o tipovima podataka.

2. U novokreiranoj tabeli podesite da je naziv primarni ključ i da je ID jedinstveni ključ, dodatno Populacija mora biti upisana za svaki grad.

3. Iz tabele izbrišite kolone X i Y, a dodajte kolonu Zaposlenost. Zaposlenost je izražena u procentima (%).

4. Napravite tabelu Student. Tabela treba imati minimalno 10 atributa. Definišete primarni ključ, dodajte i druga ograničenja koja smatrate da su potrebna. Vodite računa o tome da tabela treba sadržavati sve relevantne podatke vezane za jednog studenta. Dodatno, vodite računa o veličini polja i tipovima podataka koje ćete koristiti.

4. Jezik za modifikaciju podataka (DML)

Nakon što je tabela kreirana, spremna je za pohranu podataka. Ovaj proces se često naziva „naseljavanje tabele“ i vrši pomoću DML-a. Kao primjer koristit će se tabela Opstine koja je kreirana u prethodnom poglavlju.

Npr., sljedeća naredba dodaje jedan red u tabelu Opstine (pretpostavlja se da nema nikakvih ograničenja):

INSERT INTO Opstine (ID, Naziv, Povrsina)

VALUES („78“, „Banovići“, 184)

Ako nisu specificirani svi atributi relacije tada im se dodjeljuju predodređene vrijednosti (eng. *default values*). Najčešće korištena predodređena vrijednost je NULA.

Neka je ID primarni ključ u tabeli Opstine. Tada će pokušaj dodavanja još jednog reda sa ID-om „78“ biti odbijen od strane SUBP-a zbog ograničenja primarnog ključa koji je specificiran u DDL-u.

Osnovni oblik za brisanje redova iz tabele je sljedeći:

DELETE FROM TABLE WHERE < USLOVI >

Npr., sljedeća naredba uklanja red iz tabele Opstine koji je prethodno upisan.

DELETE FROM Opstine

WHERE Naziv=“Banovići“

Zadaci za vježbu:

1. Kreirajte tabelu Putevi_BiH koja će imati sljedeće atribute: Naziv, Duzina, Tip i Vrsta. Naziv puta je kombinacija slova, brojeva i specijalnih znakova, npr. M-15.6. Dužina puta je izražena u kilometrima i zaokružuje se na desetine metara, npr. dužina puta M27=158.23km. Prema tipu put može biti lokalni, regionalni ili magistralni. Prema vrsti put može biti makadamski i asfaltirani.

2. U tabeli Putevi_BiH uvedite ograničenja: definišite primarni ključ (primary key), definišite barem jednu kolonu u kojoj mora biti upisana vrijednost (not null) i definišite raspon vrijednosti za jednu kolonu (check).

3. Izbrišite kolonu Vrsta iz tabele Putevi_BiH.

4. Dodajte novu kolonu Naplata u tabelu, u ovoj koloni mogu biti upisane samo vrijednosti Da ili Ne.

5. Dodajte 5 redova u tabelu Putevi_BiH pri tome vodeći računa o ograničenjima koja su postavljena nad tabelom.

6. Izbrišite 1 red iz tabele korištenjem atributa Naziv.

5. Relaciona algebra

Relacijska algebra (RA) je upitni jezik povezan sa relacionim modelom. Za manipulaciju podataka u jendoj relaciji, RA obezbjeđuje dvije operacije: select i project. Operacija *select* (σ) vraća podskup redova relacione tabele, a operacija *project* (π) izdvaja podskup kolona. Pretpostavlja se da su studenti kroz predavanja upoznati sa operacijama relacione algebre pa neće biti detaljno obrađene u ovkiru ovoga poglavlja već će kroz nekoliko primjera biti prikazani principi pretraživanja tabela korištenjem RA.

Primjer 1.

Date su relacione šeme:

zaposlenik (osoba-ime, ulica, grad);

posao (osoba-ime, firma-ime, plata);

firma (firma-ime, grad);

upravnik (osoba-ime, upravnik-ime).

Zadaci:

a) Pronaći imena osoba koja rade u firmi “Granit”!

Rješenje:

$$R = \pi_{osoba-ime}(\sigma_{firma-ime} = \text{Granit}(posao))$$

b) Pronaći imena i prebivalište osoba koje rade u firmi “Granit”!

$$R = \pi_{osoba-ime,grad}(zaposlenik \bowtie (\sigma_{firma-ime} = \text{Granit}(posao))$$

ili

$$ZaposleniGranit = \pi_{osoba-ime}(\sigma_{firma-ime} = \text{Granit}(posao))$$

$$R = \pi_{osoba-ime,grad}(zaposlenik \bowtie ZaposleniGranit)$$

c) Pronaći osobe koje rade za “Granit” i imaju platu veću od 700 maraka! Prikazati imena zajedno sa mjestom prebivališta!

$$R = \pi_{osoba-ime,ulica,grad}(\sigma_{(firma-ime=\text{"Granit"} \wedge plata > 700)}posao \bowtie zaposlenik)$$

ili

$$ZaposleniGranit = \pi_{osoba-ime}(\sigma_{firma-ime} = \text{Granit}(posao))$$

$$Plata700 = \pi_{osoba-ime}\sigma_{plata>700}\ (ZaposleniGranit)$$

$$R = Plata700 \bowtie zaposlenik$$

d) Pronaći imena svih zaposlenika koji rade u firmi koja se nalazi u istom gradu u kome oni žive!

$$R = \pi_{osoba-ime}(zaposlenik \bowtie posao \bowtie firma)$$

e) Pronaći imena svih zaposlenika koji žive u istom gradu i istoj ulici kao njihovi upravnici!

$$R = \pi_{osoba-ime}((zaposlenik \bowtie upravnik)$$

$$\bowtie_{(upravnik-ime=zaposlenikK.osoba-ime \wedge zaposlenik.ulica=zaposlenikK.ulica \wedge zaposlenik.grad=zaposlenikK.grad)}$$

$$(\rho_{zaposlenikK}(zaposlenik)))$$

f) Pronaći sve osobe koje ne rade u firmi "Granit"!

Ako osoba može raditi za samo jednu firmu rješenje je:

$$R = \pi_{osoba-ime}(\sigma_{firma-ime} \neq \text{Granit}(posao))$$

Međutim, ako baza dozvoljava da se osoba može nalaziti u tabeli „zaposlenik" ali ne i u tabeli „posao" tada je rješenje:

$$R = \pi_{osoba-ime}(zaposlenik) - \pi_{osoba-ime}(\sigma_{(firma-ime} = \text{"Granit"})(posao))$$

g) Pronaći imena svih koji zarađuju više od svakog zaposlenika firme "Energoinvest"!

$$R = \pi_{osoba-ime}(posao) - (\pi_{posao.osoba-ime}(posao$$

$$\bowtie_{(posao.plata \leq posaoK.plata \wedge posaoK.firma-ime} = \text{"Energoinvest"})_{\rho posaoK}(posao)))$$

h) Pretpostavimo da firme imaju poslovnice u više gradova. Pronaći sve firme koje imaju poslovnicu u istom gradu kao i "Energoinvest"!

$$R = \pi_{firma-ime}(firma \div \left(\pi_{grad}\left(\sigma_{firma-ime} = \text{Energoinvest}(firma)\right)\right))$$

Primjer 2:

Date su relacione šeme:

student (<u>br-indeksa</u>, ime, adresa, apsolvent);

predmet (<u>sifra</u>, naziv);

registriran (<u>br-indeksa</u>, <u>sifra</u>).

a) Pronaći šifre predmeta na kojima je registrovan barem jedan student!

Ako se na neki predmet registruje student tada se šifra predmeta automatski mora pojaviti u tabeli registrovan tako da je rješenje:

π_{sifra}(registrovan)

b) Pronaći nazive svih predmeta na kojima je registrovan barem jedan student!

U tabeli predmeti se nalaze svi predmeti koji se mogu slušati na fakultetu, dakle tu su i oni predmeti koje niko ne sluša (npr. neki dosadni izborni predmeti). Ako tabelu predmet i registrovani prirodno spojimo izvršit će se filtriranje i ostat će samo oni predmeti na kojima ima registrovanih studenata.

π_{naziv}(predmet⋈registrovan)

c) Pronaći šifre predmeta na kojima nema registrovanih studenata!

Rješenje upita pod a je tabela sa šiframa predmeta na kojima ima registrovanih studenata.

Ako sada od svih predmeta oduzmemo one na kojima su registrovani studenti ostat će nam oni predmeti na kojima nema neregistrovanih studenata. Dakle, rješenje je:

R = π_{sifra}(predmet) - π_{sifra}(registrovan)

d) Pronaći nazive predmeta na kojima nema registrovanih studenata!
(prethodno su pronađene šifre, prirodnim spajanjem sa predmetima pronaći nazive).
Isti princip kao u prethodnom slučaju, nađemo one na kojima su registrovani pa oduzmemo od svih – dobijemo one na kojima nema registrovanih.

e) Pronaći imena studenata i nazive predmeta na kojima su registrovani!

Prirodnim spajanjem izvršit će se filtriranje i u istom redu će biti i ime studenta i naziv predmeta na kojem je taj student registrovan. Iz toga izdvojimo onio šta nam treba (ime studenta i naziv predmeta) i dobijemo rješenje. Ovo ćete najbolje vidjeti ako sebi napravite par malih tabela i izvršite prirodno spajanje.

$\pi_{ime,naziv}$(student⋈registrovan⋈predmet)

f) Pronaći br-indeksa za studente koji su registrovani na predmetu “Baze podataka” ili “Satelitska navigacija”!

Pošto trebamo naći brojeve indeksa za određene nazive predmeta trebaju se spojiti sve tabele jer tabele student i predmet nemaju direktne veze. Iz te novonastale tabele u prvom slučaju treba izdvojiti redove u kojima je predmet Baze podataka i zatim na njih dodati one zapise u kojima

je predmet Satelitska navigacija. Dakle imamo dva upita čiji se rezultati spajaju u jedan da bi se dobio konačan rezultat. Dakle, trebat će vam operator UNIJA jer imamo uslov „ili“.

$\pi_{\text{br.indexa}}(\sigma_{naziv="Baze\ podataka"}(\text{student} \bowtie \text{registrovan} \bowtie \text{predmet})) \cup$

$\pi_{\text{br.indexa}}(\sigma_{naziv="Satelitska\ navigacija"}(\text{student} \bowtie \text{registrovan} \bowtie \text{predmet}))$

g) Pronaći br-indeksa za studente koji su registrovani na predmetima “Baze podataka” i “Satelitska navigacija”!

Skoro isti upit kao prethodni samo što umjeto UNIJA koristitite PRESJEK.

h) Pronaći predmete na kojima su registrovani svi studenti!

$\pi_{\text{sifra,br\_indexa}}(\text{registrovan}) / \pi_{\text{br\_indexa}}(\text{student})$

Pretpostavimo da imamo samo 3 studenta radi jednostavnosti:

student	Br.indexa	ime	adresa	apsolvent
	121	Amar	Titova 25	NE
	225	Zoran	Kranjčevićeva 7	DA
	287	Sabina	Koševska 75	DA

Također, radi jednostavnosti pretpostavimo da imamo samo 4 predmeta, i to:

Geodezija – šifra 100, matematika – šifra 200, GIS – šifra 300, Ceste – šifra 400

Tabela

registrovan	Br.indexa	Šifra (predmeta)
	121	100
	225	100
	121	200
	121	400
	287	200
	287	100
	121	300
	225	400

A=$\pi_{\text{sifra,br\_indexa}}$(registrovan), rezultat je čitava tabela „registrovan“ samo što se sada zove „A“

B= π_{br_indexa}(student), rezultat će biti:

B	Br.indexa
	121
	225
	287

Dakle, uslovi za operaciju dijeljenja A/B su ispunjeni jer se svi atributi iz tabele B nalaze u tabeli A.

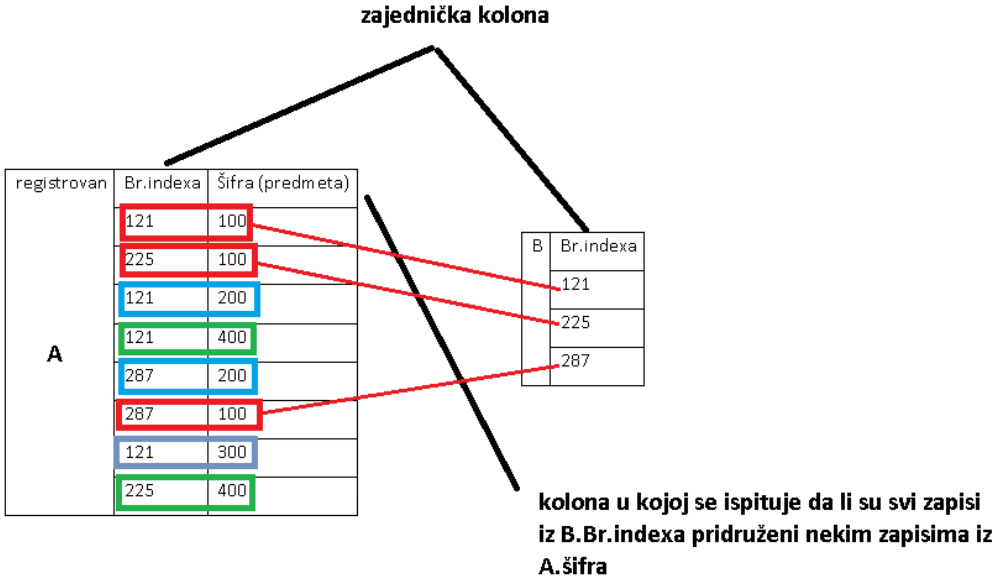
Upit:

π_{sifra,br_indexa}(registrovan) / π_{br_indexa}(student)

Odnosno, nakon što smo preimenovali

A/B

Upit će u tabeli A (i to samo u koloni „šifra“ jer je to jedina kolona u tabeli A koja se razlikuje od tabele B) tražiti zapise kojima su pridruženi svi zapisi iz tabele B.



Jedino je šifri predmeta 100 pridružen svaki broj indexa iz tabele B što znači da su jedino na predmet sa šifrom 100 prijavljeni svi studenti

i) Pronaći spisak predmeta na kojima su svi apsolventi registrovani!

Slično kao u prethodnom primjeru s tim da ovdje imamo još jedan uslov, a to je da se radi o apsolventima.

π_{sifra,br_indexa}(registrovan) / $\pi_{br\_indexa}(\sigma_{apsolvent=DA}$(student))

6. Relacioni SQL upiti

Nakon što su tabele kreirane i popunjene korištenjem DDL i DML-a može se vršiti pretraživanje po tabelama da bi se dobili željeni podaci. Pretpostavlja se da su studenti u okviru predavanja savladali operatore i funkcije SDQL-a kao što su AND, OR, matematički operatori, itd.

Primjer: Neka su date tabele:

zaposlenik (osoba-ime, ulica, grad);

posao (osoba-ime, firma-ime, plata);

firma (firma-ime, grad);

upravnik (osoba-ime, upravnik-ime).

a) Prikazati imena i plate svih zaposlenika firme „Energoinvest“. Grupisati rezultate po gradovima!

SELECT posao.osoba-ime, posao.plata

FROM posao, zaposlenik

WHERE zaposlenik.osoba-ime=posao.osoba-ime AND firma-ime=“Energoinvest“

GROUP BY grad

b) Pronaći najplaćenijeg zaposlenika firme „BH Telecom“ koji živi u Sarajevu!

SELECT posao.osoba-ime, MAX (posao.plata)

FROM zaposlenik, posao

WHERE zaposlenik.osoba-ime=posao.osoba-ime AND zaposlenik.grad=“Sarajevo“

c) Pronaći prosječnu platu za svaku firmu!

SELECT firma-ime, AVG (plata)

FROM posao

GROUP BY firma-ime

i) Pronaći imena svih zaposlenika koji rade u firmi koja se nalazi u istom gradu u kome oni žive!

SELECT zaposlenik.ime

FROM zaposlenik,posao, firma

WHERE zaposlenik.osoba-ime=posao.osoba-ime AND posao.firma-ime=firma.firma-ime

AND zaposlenik.grad=firma.grad

j) Pronaći imena i prosječnu platu zaposlenika koji žive u Sarajevu!

SELECT zaposlenik.ime, AVG (posao.plata)

FROM zaposlenik, posao

WHERE zaposlenik.grad=“Sarajevo“

GROUP BY zaposlenik.grad

k) Pronaći sve osobe koje ne rade u firmi “Granit”!

SELECT posao.osoba-ime

FROM posao

WHERE firma.ime<>“Granit“

Primjer 2:

Neka su date tabele:

Opština (Ime, Površina, Populacija)

Grad (Ime, Populacija)

Rijeka (Ime, Dužina)

Put (Ime, Tip, Dužina)

a) Pronaći opštinu sa najmanjim brojem stanovnika bez korištenja funkcije MIN!

SELECT Opština.Ime

FROM Opština

WHERE Opština.Populacija < ALL (SELECT Opština1.Populacija FROM Opština1,Opština WHERE Opština.Ime<>Opština1.Ime)

b) Pronaći prosječnu dužinu puteva!

SELECT AVG(Put.Dužina)

FROM Put

c) Pronaći ukupan broj stanovnika koji žive u Sarajevu, Tuzli, Mostaru i Banjaluci!

SELECT SUM(Grad.Populacija)

FROM Grad

WHERE Grad.Ime="Sarajevo"

OR Grad.Ime="Tuzla"

OR Grad.Ime="Mostar"

OR Grad.Ime="Banjaluka"

a) Pronaći drugu najveću opštinu po površini.

SELECT Opština.Ime, MAX (Opština.Površina)

FROM Opština

WHERE Opština.Površina < ALL (SELECT MAX (Opština1.Površina) FROM Opština, Opština1 WHERE Opština.Ime<>Opština1.Ime)

b) Pronaći koji od gradova ima najmanju populaciju: Konjic, Mostar, Trebinje, Cazin!

SELECT Grad.Ime, MIN (Grad.Populacija)

FROM Grad

WHERE Grad.Ime="Konjic"

OR Grad.Ime="Mostar"

OR Grad.Ime="Trebinje"

OR Grad.Ime="Cazin"

c) Pronaći sve gradove koji se ne nalaze u opštinama Bihać i Čapljina.

SELECT Grad.Ime

FROM Grad, Opština

WHERE Opština.Ime<>"Bihać"

AND Opština.Ime<>"Čapljina"

Zadatak:

Upite relacione algebre uradite korištenjem SQL upita! Uporedite dvije metode pretraživanja podataka: Kojom metodom ste lakše došli do rezultata? Koje su prednosti i nedostaci jedne metode u odnosu na drugu?

7. Prostorni SQL upiti

Iako su jako dobri alati za procesiranje upita, RA i SQL imaju svoje nedostatke. Glavni nedostatak je da vrše upite samo na jednostavnim podacima kao što su brojevi, datumi i slova. Aplikacije prostornih baza moraju biti sposobne rukovati kompleksnim tipovima podataka kao što su tačke, linije i poligoni. U nastavku su dati primjeri prostornih SQL upita. Pretpostavlja se da su studenti savladali relacijske SQL upite i prostorne operatore i funkcije. Prostorni upiti skoro uvijek koriste kombinaciju relacijskih i prostornih operatora i funkcija.

Primjer 1:

Opština (Ime, Površina, Populacija, Oblik)

Grad (Ime, Populacija, Oblik)

Rijeka (Ime, Dužina, Oblik)

Put (Ime, Tip, Dužina, Oblik)

a) Pronaći opštine koje imaju nadprosječnu površinu.

```
SELECT Opština.Ime
FROM Opština
HAVING Opština.Površina>AVG(Opština.Površina)
```

b) Pronaći grad koji je najbliži presjeku puta M17 i rijeke Neretve.

```
SELECT Grad.Ime, MIN (Distance(Grad.Oblik,Intersection(Put.Oblik, Rijeka.Oblik))
FROM Grad, Rijeka, Put
WHERE Put.Naziv="M17"
AND Rijeka.Ime="Neretva"
```

c) Kolika je ukupna populacija općina kroz koje prolazi put M15?

```
SELECT SUM(Opština.Populacija)
FROM Opština, Put
WHERE Crosses (Optina.Oblik, Put.Oblik)=1
```

d) Koliko susjednih općina ima općina Kladanj?

```
SELECT Opština.Ime, Count (Opština1.Ime)
FROM Opština, Opština1
WHERE Touch (Opština.Oblik,Opština1.Oblik)=1
AND Opština.Ime="Kladanj"
```

e) Gradskim područjem se smatra sve u krugu 10km oko centra grada. Rijeka Bosna se izlila iz svoga korita u području od 1km. Pronaći imena gradova čija su područja ugrožena poplavom. (3)

SELECT Grad.Ime

FROM Grad, Rijeka

WHERE Overlap (Buffer(Grad.Oblik,10), Buffer(Rijeka.Oblik,1))=1

AND Rijeka.Ime="Bosna"

f) Predajnik signala za mobilne uređaje se nalazi u centru Sarajeva. Koliki domet treba imati signal da bi se pomoću njega moglo razgovarati u Konjicu?

SELECT Distance (Grad1.Oblik, Grad2.Oblik)

FROM Grad1,Grad2

WHERE Grad1.Ime="Sarajevo"

AND Grad2.Ime="Konjic"

g) Pronaći udaljenost Mostara od Grinviča (nultog meridijana)!

SELECT Distance(Point(Grad.Oblik.x,0),Grad.Oblik)

FROM Grad

WHERE Grad.Ime="Mostar"

h) Pronaći opštinu sa najvećom površinom kroz koju prolazi rijeka Neretva!

SELECT Opština.Ime, MAX (Opstina.Povrsina)

FROM Opština, Rijeka

WHERE Crosses (Opština.Oblik, Rijeka.Oblik)=1

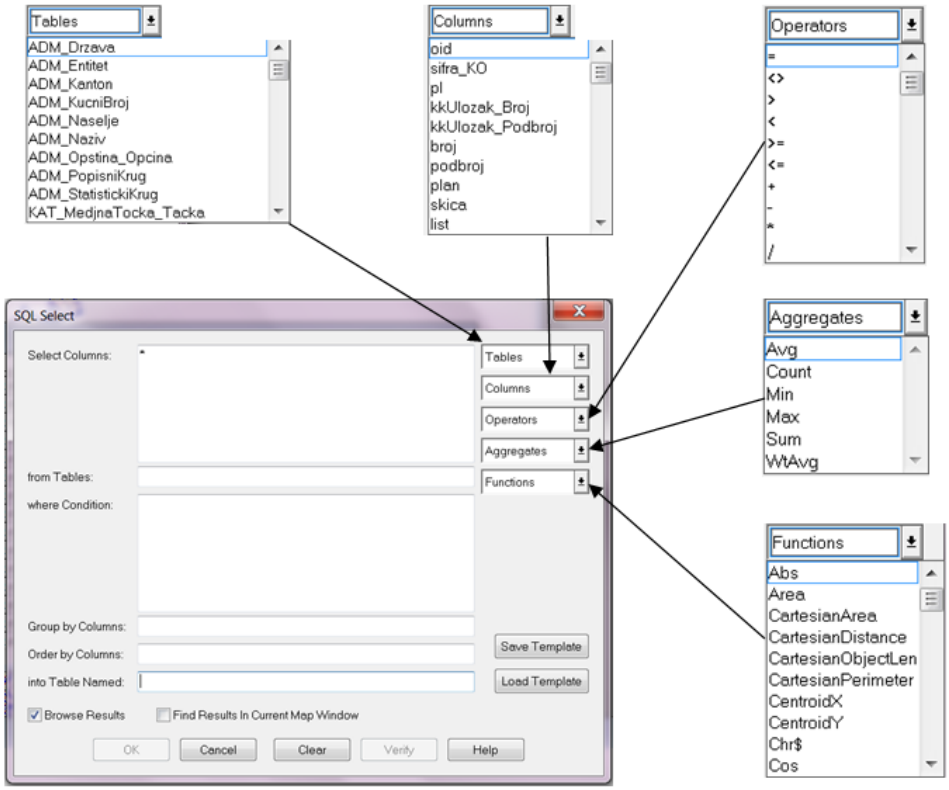
AND Rijeka.Ime="Neretva"

Primjer primjene SQL upita u rješavanju jednostavnih zadataka:

U katastarskoj opštini Donja Jablanica vrši se rekonstrukcija puta. Osovina novog puta pratit će osovinu starog puta koji je na katastarskom planu označen kao katastarska čestica broj 1064. Prema projektu novi put će imati dvije trake od kojih je svaka širine 2,5m, bankine 0,75m i trotoar širine 1,5m. U tu svrhu potrebno je izvršiti eksproprijaciju zemljišta kako bi se moglo početi za izgradnjom. Potrebno je pronaći sve parcele koje će učestvovati u eksproprijaciji i napraviti spisak svih posjednika kako bi se mogli riješiti imovinsko-pravni odnosi za svaku parcelu od interesa.

Za rješavanje zadatka korišten je softver MapInfo Professional 9.5. Ovo možete uraditi i u bilo kojem drugom GIS softveru. Moguće je da će se u manjoj mjeri sintaksa razlikovati, ali važan je princip i redoslijed rješavanja zadatka.

Na slici ispod je prikazan izgled prozora za postavljanje SQL upita u MapInfo-u.



Slika: Izgled prozora za postavljanje SQL upita u softveru MapInfo Professional 9.5

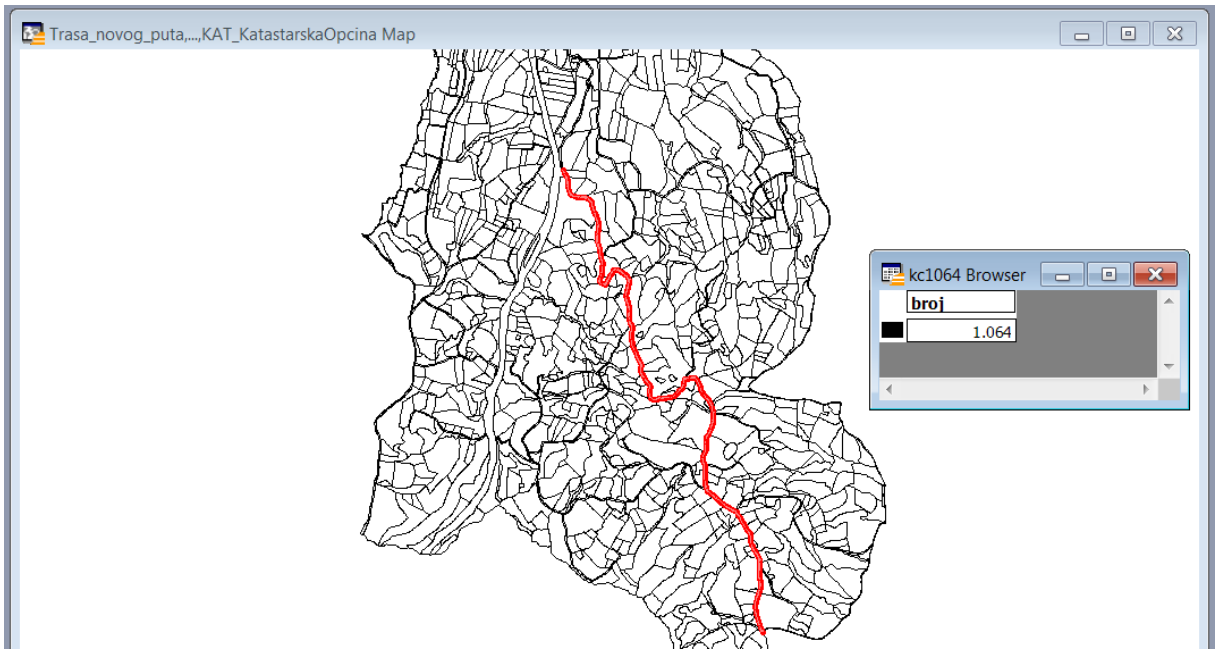
Prilikom postavljanja upita prvo se specificira jedna ili više tabela koje želimo koristiti u upitu. Klikom na listu *Tables* prikazuje se lista svih trenutno otvorenih tabela u MapInfo-u. Nakon odabira tabela koje će se koristiti u upitu odabiru se kolone koje trebaju biti prikazane kao rezultat upita. Klikom na listu *Columns* prikazuju se sve kolone odabranih tabela u prethodnom koraku. U upitu je također moguće koristiti veliki broj operatora (+, -, >, And, Or, ...) i funkcija (Maximum, Minimum, Sin, Cos, ...) uključujući i prostorne operatore (Area, Distance, Centroid, ...). U polju *Group by Columns* specificira se način grupiranja rezultata upita, a u polju *Order by Columns* specificira se način redanja rezultata. U polje *Into Table Named* upisuje se naziv tabele u koju će biti pohranjeni rezultati. Ukoliko će neki upit biti često korišten moguće ga je snimiti klikom na dugme *Save Template* i zatim ga ponovno pozvati klikom na dugme *Load Template*. Klikom na dugme *Load Template* otvara se lista svih snimljenih upita, a zatim se dvostrukim klikom odabere željeni upit. Opcija *Browse Results* služi za tabelarni

prikaz rezultata dok opcija *Find Results In Current Map Window* služi za grafički prikaz rezultata upita. Dugme *Verify* se koristi za provjeru tačnosti sintakse upita. Kada je sintaksa ispravna pojavi se poruka „*Syntax is correct*“, a u slučaju kada nije pojavi se poruka „*Syntax error*“. Dugme *OK* potvrđuje izvođenje upita, dugme *Cancel* otkazuje izvršenje upita, a dugme *Clear* čisti prozor za postavljanje upita. Klikom na dugme *Help* otvara se prozor MapInfo-a za pomoć pri radu.

SQL upit za pronalaženje parcele 1064 će imati sljedeći oblik:

```
SELECT broj
FROM KAT_Parcela
WHERE broj=1064
INTO kc1064
```

Rezultat upita je prikazan na slici ispod. Kao što se vidi sa slike parcela 1064 se proteže od krajnjeg jugoistoka katastarske opštine prema centru katastarske opštine. Rezultat upita je pohranjen u tabelu *kc1064*.



Slika: Rezultat upita za pronalaženje parcele 1064

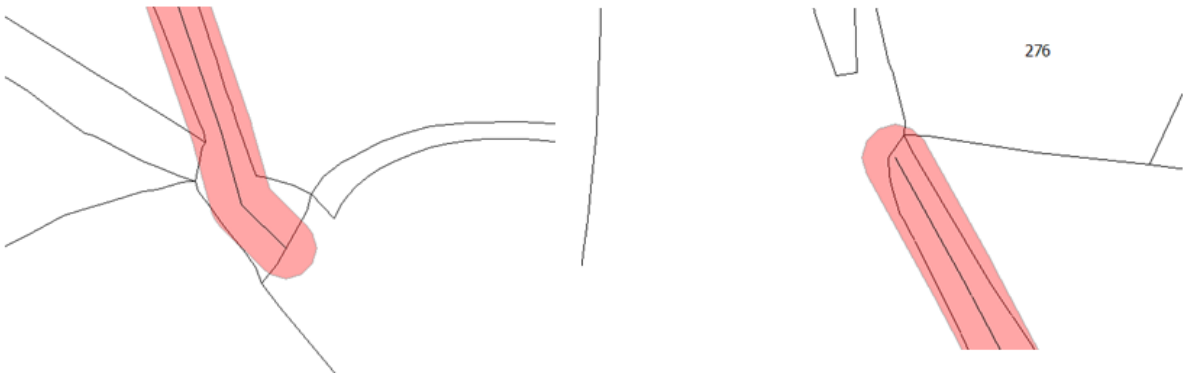
U sljedećem koraku potrebno je kreirati bafer veličine 4,75m (2,5m traka+1,5m trotoar + 0,75m bankina) oko osovine puta. Željeni bafer moguće je kreirati pomoću sljedećih naredbi:

```
CREATE OBJECT AS Buffer
FROM Trasa_novog_puta
```

WIDTH 4.75 UNITS "m"
TYPE Cartesian
RESOLUTION 12
INTO TABLE Bafer_puta

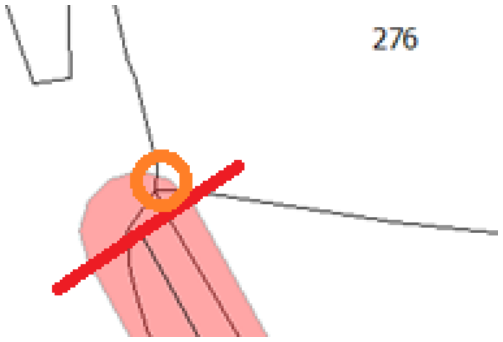
Naredba *Create Object As Buffer* definiše kreiranje novog prostornog objekta koji će biti kreiran kao bafer nekog postojećeg objekta. Klauza *From* pokazuje u kojoj tabeli se nalazi prostorni objekt na osnovu koga će se kreirati bafer (u ovome slučaju to je polilinija koja se nalazi u tabeli *Trasa_novog_puta*. Naredba *Width* definiše širinu bafera, a *Units* jedinice u kojima je izražena širina. Ako je ova vrijednost negativna, a radi se o zatvorenom poligonu, tada će bafer biti objekat koji je manji od izvornog. U slučaju kada se radi o linijskim objektima, tada se uzima apsolutna vrijednost upisana u polje *Width* i na osnovu nje se kreira bafer. *Type* određuje način mjerenja dužine (sferno ili kartezijski), u ovome slučaju radi se o kartezijskim dužinama jer je katastarski plan preslikan u ravan i sve dužine su svedene u ravan. *Resolution* određuje zaobljenost bafera na lomnim mjestima tj. određuje broj segmenata po jednom krugu u operaciji bafer, npr. ako je rezolucija 12 to znači da bi bafer kružnicu aproksimirao dvanaestouglom.

Nakon kreiranja bafera potrebno je izvršiti vizuelnu inspekciju presjeka bafera sa parcelama na krajevima linije. Razlog tome je što je potrebno izvršiti samo proširenje puta, a ne i njegovo produženje. Operacija bafer će osim proširenja izvršiti i produženje puta za vrijednost koju smo upisali u polje *Width*. Ukoliko ne isključimo parcele koje su na ovaj način obuhvaćene baferom moguće je napraviti grešku i u postupak eksproprijacije uključiti parcele koje nisu zahvaćene rekonstrukcijom puta. Slika ispod prikazuje stanje na krajevima polilinije.



Slika: Presjek bafera i parcela na krajevima polilinije trase puta

Sa jugoistočnim krajem bafera neće biti problema jer na tom mjestu put graniči sa drugom katastarskom općinom tako da u rezultatima neće biti suvišnih parcela. S druge strane, na sjeverozapadnom kraju bafera imamo grešku koju smo tražili. Bafer koji je nastao produženjem polilinije se siječe sa parcelom 276 koja očigledno ne bi trebala da učestvuje u postupku eksproprijacije kao što se može vidjeti sa slike ispod.



Slika: Greška – presjek bafera koji je nastao produženjem polilinije i parcele 276

Dakle, potrebno je pronaći presjek bafera puta sa parcelama i iz rezultata izuzeti parcelu broj 276. Upit koji daje spisak parcela koje učestvuju u eksproprijaciji ima sljedeći oblik:

SELECT KAT_Parcela.broj

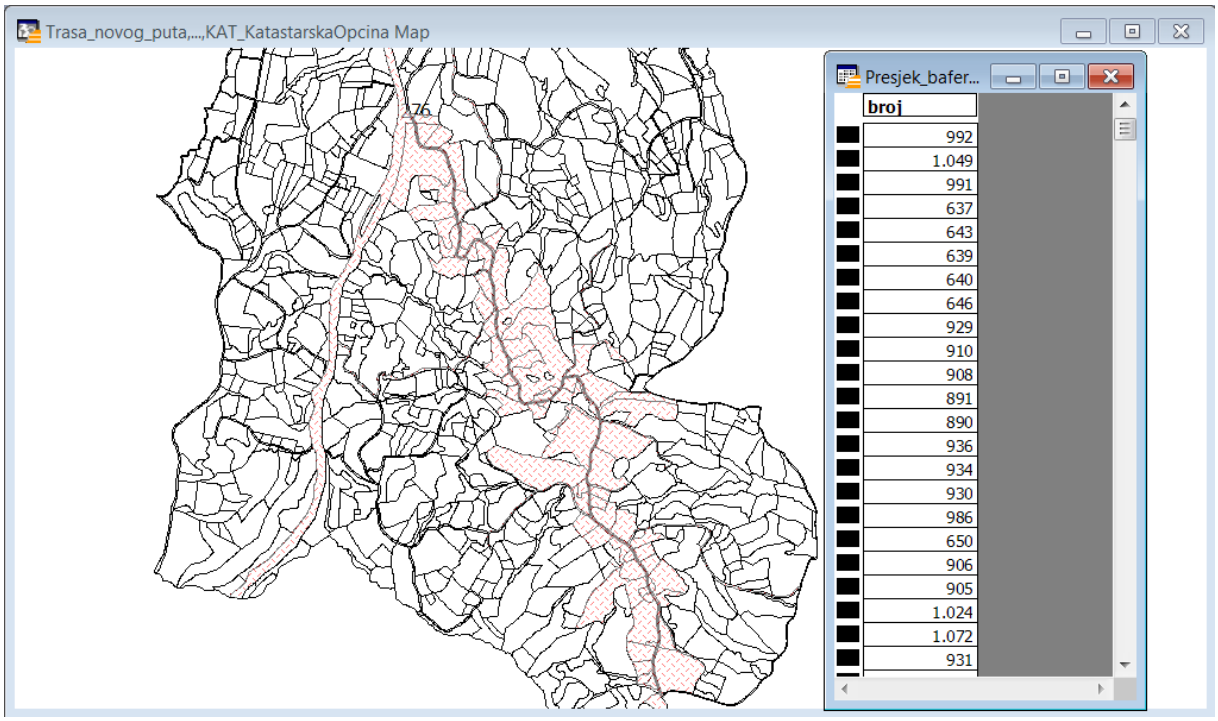
FROM KAT_Parcela, Bafer_puta

WHERE Bafer_puta.obj Intersects KAT_Parcela.obj

AND KAT_Parcela.broj < > 276

INTO Presjek_bafera_i_parcela

Rezultat upita je prikazan na slici ispod.



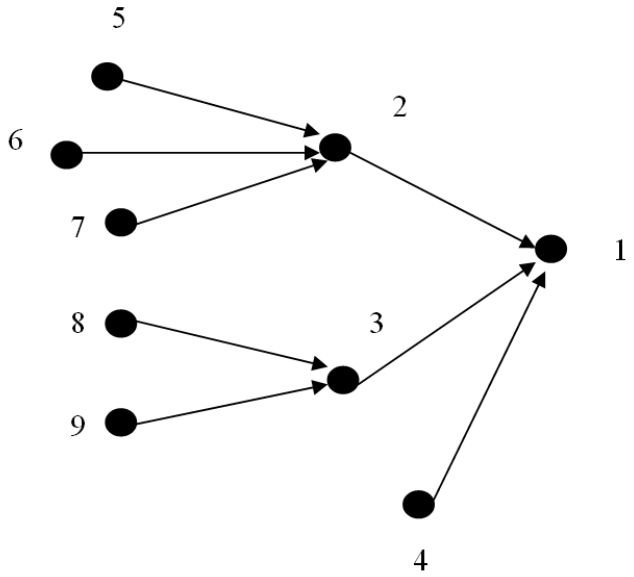
Slika: Rezultat upita za pronalaženje parcela koje učestvuju u postupku eksproprijacije

Zbog zaštite podataka posjednika lični podaci posjednika (ime, prezime, adresa, itd.) nisu bili dostupni prilikom izrade ovoga rada ali u katastarskim uredima gdje su svi podaci dostupni imena posjednika bi se lako mogla dobiti spajanjem tabele sa imenima posjednika i tabele sa brojevima parcela.

8. SQL u grafovima

SQL pravila kod grafova:

- Prolaz grafom se kontroliše pomoću "START WITH" i "CONECT BY" klauzule.
- Odnos *roditelj* - *dijete* je definisan "CONNECT BY" klauzulom.
- Smjer traženja se upravlja korištenjem 'PRIOR' operatora.



Slika: Graf „Rijeke“

Operator „Prior“

SELECT Izvor
FROM Rijeke
CONNECT BY PRIOR Izvor= Odredište
START WITH Odredište= 1

Ovaj upit služi za pretraživanje grafa, a riječ PRIOR tj. mjesto na koje je postavljena određuje pravac pretraživanja - određuje hoće li se tražiti prethodnici (djeca) ili sljedbenici (roditelji) od odabranog čvora. Dakle:

CONNECT BY PRIOR source = dest traži djecu od čvora koji je odabran u START WITH dijelu.

CONNECT BY source = PRIOR dest traži roditelje od čvora koji je odabran u START WITH dijelu.

Primjer 1:

Pronaći koje rijeke će biti zagađene ako je došlo do curenja u rijeci 6?

Napomena: Ustvari treba pronaći roditelje od 6, pa roditelja od roditelja od 6 itd.

```
SELECT Izvor
FROM Rijeke
CONNECT BY Izvor= PRIOR Odredište
START WITH Odredište= 6
```

Primjer 2:

Znamo da je zagađena rijeka 3. Odrediti potencijalne zagađivače!

```
SELECT Izvor
FROM Rijeke
CONNECT BY PRIOR Izvor= Odredište
START WITH Odredište= 3
```

Primjer 3:

Prikazati spisak svih direktnih i indirektnih pritoka rijeke 1 !

```
SELECT  Izvor
FROM  Rijeke
CONECT  BY PRIOR Izvor=Odredište
START  WITH Odredište= 1
```

Primjer 4:

Koliko rijeka može biti zagađeno ako je zagađena rijeka 1?

Napomena: Dodajemo tabelu "Rijeka" sa kolonama "ID" i "Ime"!

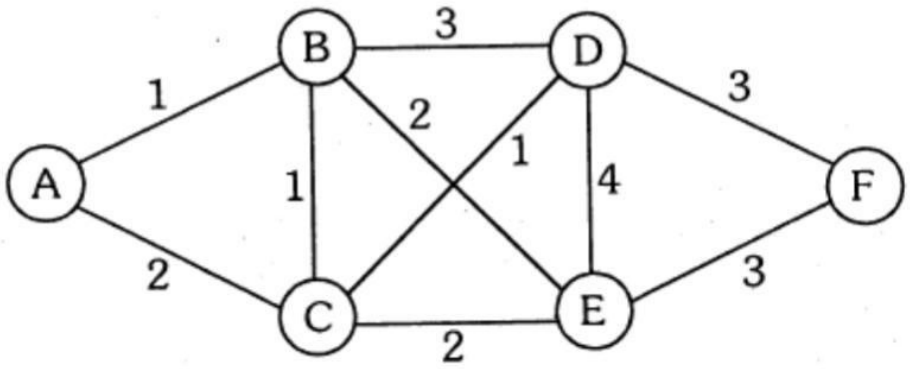
```
SELECT  COUNT(Izvor)
FROM  Rijeke
CONNECT  BY Izvor= PRIOR dest
START  WITH Izvor IN
(SELECT  Rijeka ID
FROM Rijeka
WHERE  Ime='R1')
```

9. Problem najkraćeg puta

Dijkstra algoritam

Ideja Dijkstra algoritma je gradi stablo koje se sastoji od bridova koji čine minimalne puteve od početnog vrha do svih ostalih vrhova. U jednom koraku biramo brid koji spaja vrh koji nije u stablu sa stablom, ali pri tome za svaki vrh računamo udaljenost od početnog vrha. Od svih bridova koje u jednom koraku možemo izabrati uzimamo onaj za koji je pripadni vrh, koji nije u stablu, najmanje udaljen od početnog vrha.

Primjer: Za graf na slici ispod pronaći najkraći put od vrha A do vrha F!



Prvi korak:

-Postavlja se da je udaljenost do A=0 jer je to početni čvor, a za sve ostale se piše beskonačno.
-Iz A se može doći do B i C. AB=1, a AC=2. Pišu se privremene vrijednosti za čvorove B(1) i C(2). Odabere se najmanja privremena vrijednost što je u ovom slučaju B(1) i ta vrijednost postaje konačna.
-Nakon što je završen prvi korak postoje dvije konačne vrijednosti i to: **A(0) i B(1)** i jedna privremena C(2).

Drugi korak:

-Iz B se može doći do C(1+1=2), D(1+3=4) i E(1+2=3). C već ima privremenu vrijednost 2 tako da nema promjene vrijednosti, za D se upisuje 4 jer je manje od beskonačno, za E se piše 3 jer je manje od beskonačno. Odabere se najmanja privremena vrijednost što je u ovome slučaju C(2) koja postaje konačna.
-Nakon što je završen drugi korak postoje 3 konačne vrijednosti: **A(0), B(1), C(2)** i dvije privremene D(4) i E(3).

Treći korak:

-Iz C se može doći do B (međutim B već ima konačnu vrijednost pa nema potrebe za računanjem), D(2+1=3) i E(2+2=4). Za D se piše nova privremena vrijednost jer je novodobivena manja od postojeće, dakle D(3), a za E nema promjena. Sada postoje dvije privremne vrijednosti koje su jednake, D(3) i E(3), i odabire se samovoljno koja će biti konačna. Recimo da je E(3) konačna.

-Nakon završenog trećeg koraka postoje 4 konačne vrijednosti: **A(0), B(1), C(2), E(3)** i jedna privremena D(3).

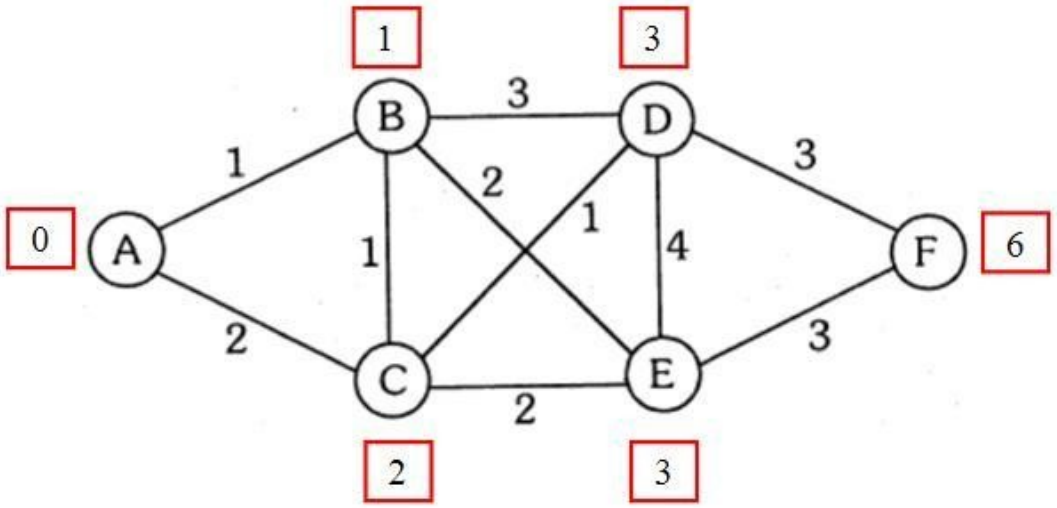
Četvrti korak:

-Iz E se može doći do D(3+4=7) i F(3+3=6). Za D se neće pisati novodobivena vrijednost jer je veća od postojeće, a za F se piše nova privremena vrijednost F(6). -Nakon završenog četvrtog koraka postoji 5 konačnih vrijednosti: **A(0), B(1), C(2), E(3), D(3)** i jedna privremena F(6).

Peti korak:

-Iz D se može doći do F(3+3=6). Ova vrijednost je jednaka postojećoj tako da nema promjena ali se može zaključiti da će biti dvije najkraće putanje u grafu od čvora A do čvora F.

-U petom koraku su dobijene sve najkraće udaljenosti od A do ostalih čvorova u grafu, dakle dobijeno je: **B(1), C(2), E(3), D(3) i F(6).**



Određivanje najkraće putanje

Da biste odredili najkraću putanju idete unazad, od čvora F prema A. Za određivanje najkraćeg puta se vršilo sabiranje težina, a za određivanje najkraće putanje ćete vršiti oduzimanje. Dakle ispitujete da li je vrijednost u F umanjena za vrijednost težine do susjednog čvora jednaka vrijednosti u odabranom čvoru. Ako jeste taj brid je dio najkraće putanje, ako nije taj brid nije dio najkraće putanje. Ovaj postupak se ponavlja dok se ne dođe do čvora A.
U ovome slučaju je:
Prvi korak:

1. FD jer je 6-3=3
2. FE jer je 6-3=3 Drugi korak: 1.DC jer je 3-1=2 2.EB jer je 3-2=1

Treći korak:
1.CA jer je 2-2=0 2.BA jer je 1-1=0

Dakle najkraće putanje su:
1.ACDF
2.ABEF.

Floydov algoritam

Korištenjem Floydovog algoritma odredite najkraće putanje između čvorova za graf iz prethodnog primjera. U svakom koraku objasnite zašto su upisane odgovarajuće vrijednosti u matricu susjedstva.

10. SpatiaLite - izrada jednostavne prostorne baze podataka

Softver i korišteni podaci su preuzeti sa:

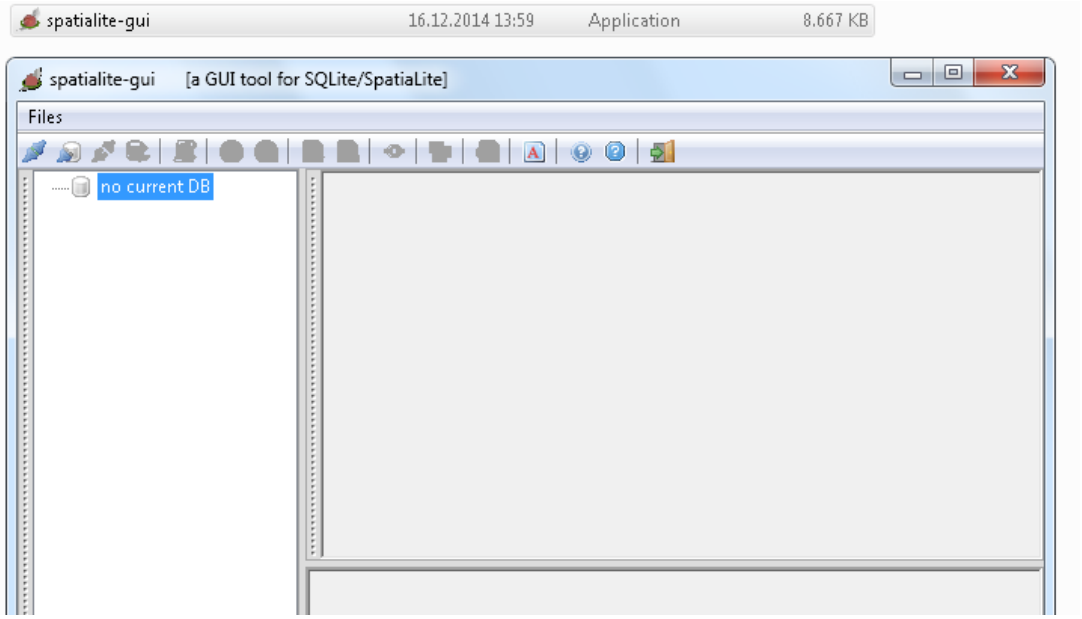
http://www.gaia-gis.it/gaia-sins/index.html

http://www.gaia-gis.it/gaia-sins/spatialite-cookbook/html/start.html

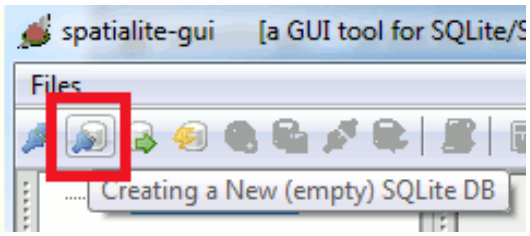
SpatiaLite je sistem za upravljanje prostornim bazama podataka. SpatiaLite podržava standarde SQL92 i OGC-SFS **(***Open Geospatial Consortium - Simple Feature Specification).* Baziran na SQLite-u, ustvari SQLite je zadužen za standardni SQL, a SpatiaLite je zadužen za prostornu ekstenziju. Dobar za pojedinačne korisnike, loš kada se zahtijeva višestruki pristup bazi podataka. Prednost je to što je jednostavan za korištenje i što je baza podataka jedan fajl koji se može prenositi između različitih računara čak i ako ti računari ne koriste isti operativni sistem.

U nastavku će biti prikazan postupak izrade jednostavne baze podataka korištenjem preuzetog softvera i podataka.

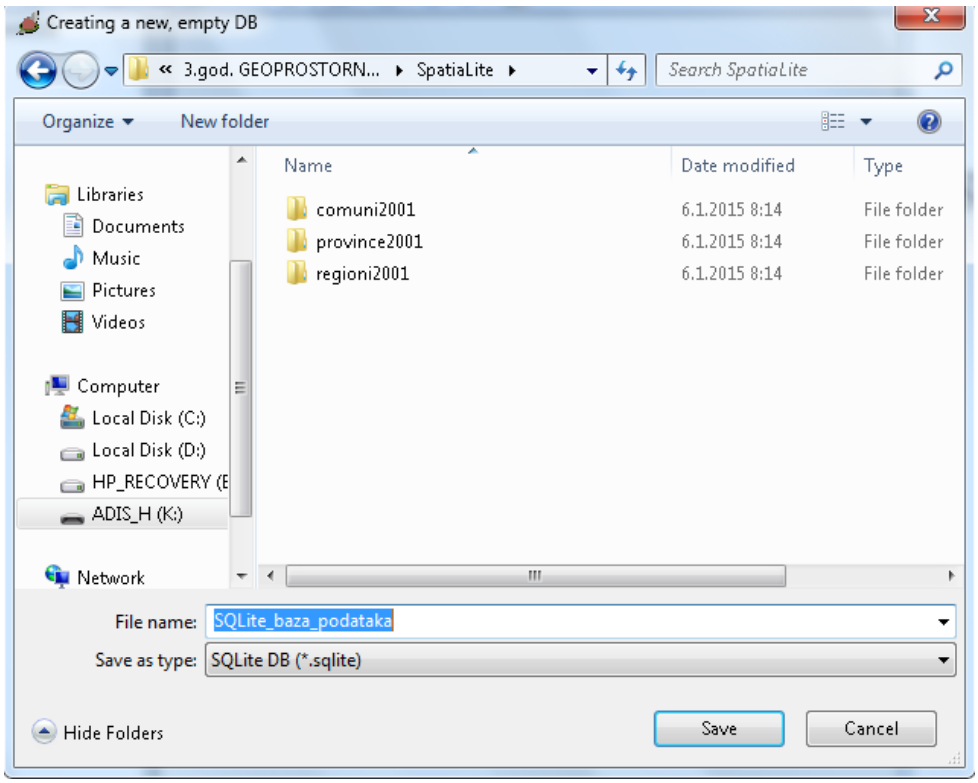
1. Dvoklikom na aplikaciju pojavi se prozor kao na slici ispod. (**Napomena**: Postoji i konzolna varijanta SpatiaLite-a, a ovdje će se koristiti verzija sa grafičkim interfejsom.)



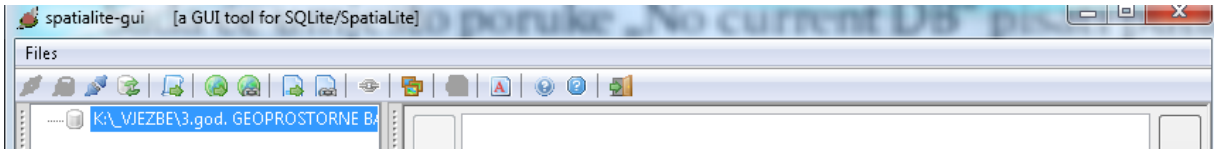
2. Klikom na dugme „Creating a New (empty) SQLite DB“ pojavit će se novi prozor u kome kreirate vašu bazu podataka.



3. U polje **File name** upišite ime vaše baze podataka, npr. SQLite_baza_podataka.



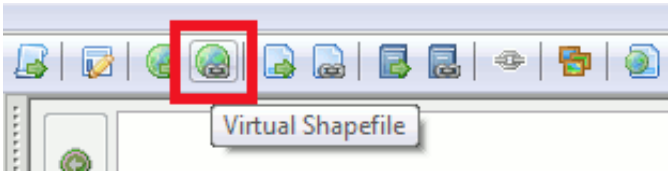
4. Sada će umjesto poruke „No current DB“ pisati putanja novokreirane baze podataka.



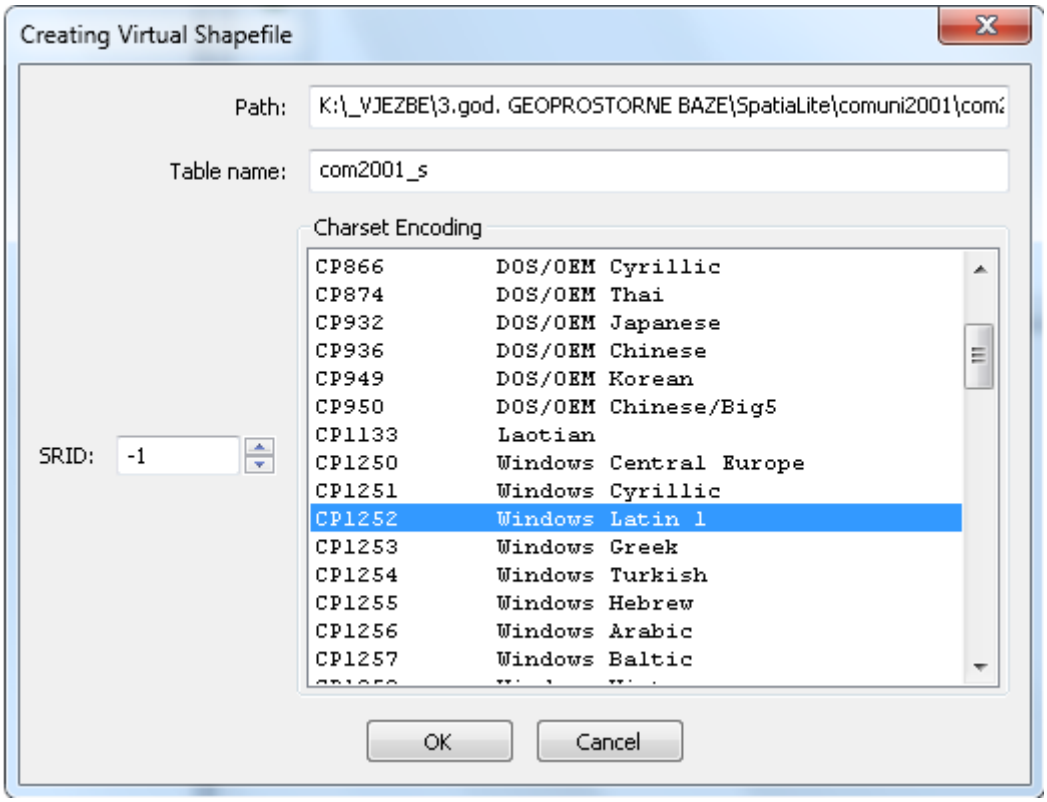
Baza podataka je kreirana, sada ju je potrebno popuniti podacima. U bazu je moguće dodati već postojeće podatke, a moguće je kreirati i nove tabele. Naredbe za kreiranje, ažuriranje i pretraživanje tabela u bazi podataka se u nekim segmentima razlikuju (iako su veoma slične) od onih koje su spominjane u prethodnim poglavljima. Prvo ćete dodati preuzete podatke u bazu, a kasnije ćete krirati i vlastite tabele.

5. Dodavanje postojećih podataka u bazu se vrši pomoću dugmića sa trake alata.

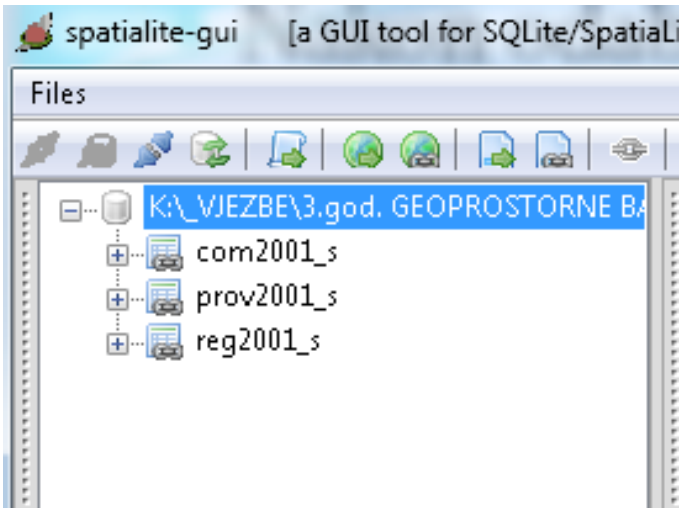
Zavisno od vrste podataka koju želite unijeti u bazu kliknite na odgovarajuće dugme. Preuzeti podaci su tipa *shapefile* pa ćete kliknuti na dugme **VirtualShapefile** sa trake alata.



6. Pojavit će se novi prozor u kome trebate odabrati fajl koji želite unijeti u bazu podataka. Nakon odabira željenog fajla pojavit će se prozor u kome odabirete željeno kodiranje Odaberite Windows Latin 1.

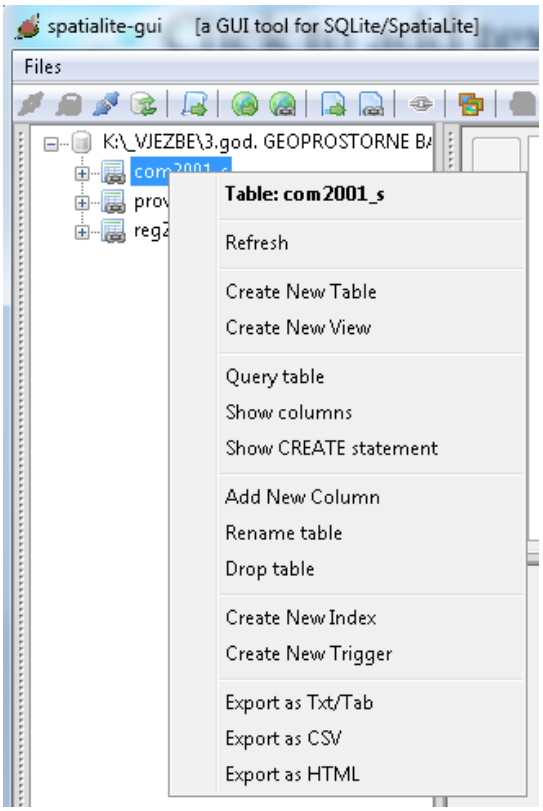


Na isti način dodajte sve podatke tipa *shapefile*. Dodane tabele će se pojaviti ispod naziva baze podataka.



Desnim klikom na ime tabele pojavit će se prozor u kome su ponuđene različite naredbe kojima možete manipulirati tabelom ili kreirati nove tabele. Sve ove naredbe moguće je upisati konzolno u prozor koji se (obično) nalazi desno od prozora sa spiskom tabela. Npr. Naredba **Query table** je jednaka SQL izrazu:

SELECT * FROM ime_tabele



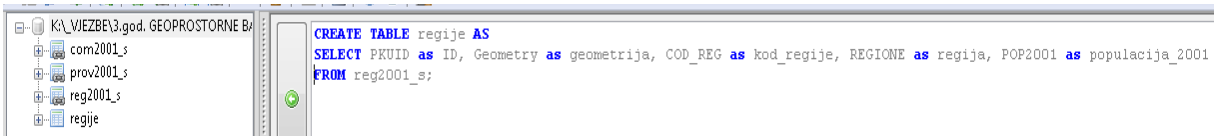
7. Preimenujte tabelu reg2001_s u regije, a zatim kolone PKUID, Geometry, COD_REG, REGIONE i POP2001 u: ID, geometrija, kod_regije, regija, populacija_2001. SQL upit koji radi sve ovo bi imao sljedeći oblik:

CREATE TABLE regije **AS**

SELECT PKUID **as** ID, Geometry **as** geometrija, COD_REG **as** kod_regije, REGIONE **as** regija, POP2001 **as** populacija_2001

FROM reg2001_s;

Na ovaj način je kreirana nova tabela koja ima iste podatke kao tabela red_2001s ali sa drugim nazivima kolona.



8. Sračunajte površinu regija iz geometrije, zatim spremite podatke u novu kolonu površina i sortirajte regije po površini:

SELECT ID, regija, **area**(geometrija) **as** povrsina

FROM regije

ORDER BY povrsina;

```
select ID, regija, area(geometrija) as povrsina
from regije
order by povrsina;
```

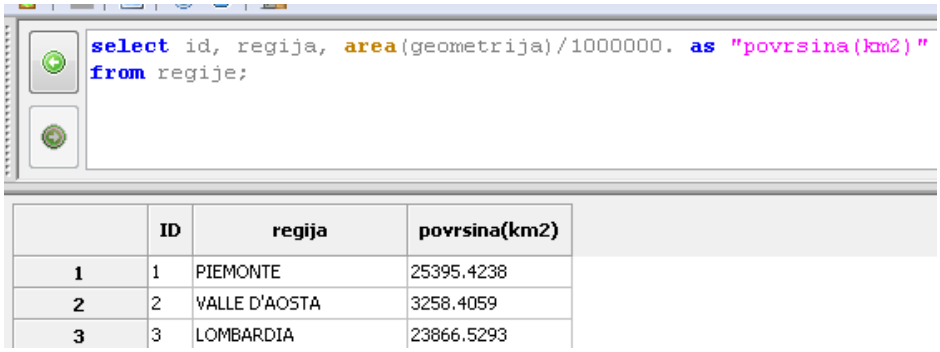
	ID	regija	povrsina
1	2	VALLE D'AOSTA	3258405868.0317
2	14	MOLISE	4458305204.5605
3	7	LIGURIA	5417606377.8444
4	6	FRIULI VENEZIA GIULIA	7865277505.3047
5	10	UMBRIA	8462847968.4921
6	11	MARCHE	9729862859.9186
7	17	BASILICATA	10070896918.2618
8	13	ABRUZZO	10831570514.9943
9	4	TRENTINO-ALTO ADIGE	13611925010.8815
10	15	CAMPANIA	13666322145.3713
11	18	CALABRIA	15214180923.6680
12	12	LAZIO	17224498263.7601
13	5	VENETO	18405161417.0920
14	16	PUGLIA	19535673049.0993
15	8	EMILIA-ROMAGNA	22120268017.0731
16	9	TOSCANA	22956355019.0774
17	3	LOMBARDIA	23866529330.5325
18	20	SARDEGNA	24048141797.2683
19	1	PIEMONTE	25395423847.6244
20	19	SICILIA	25735673661.0270

current block: 1 / 20 [20 rows]

9. Kao što se vidi sa slike brojevi u koloni povrsina su jako veliki. Razlog tome je to što je površina izražena u metrima kvadratnim. Pomoću SQL-a tu površinu možete izraziti u kvadratmim kilometrima. SQL upit koji vrši pretvaranje bi imao sljedeći oblik:

   SELECT id, regija, **area**(geometrija)/1000000. **as** "povrsina(km2)"

   FROM regije;



```
select id, regija, area(geometrija)/1000000. as "povrsina(km2)"
from regije;
```

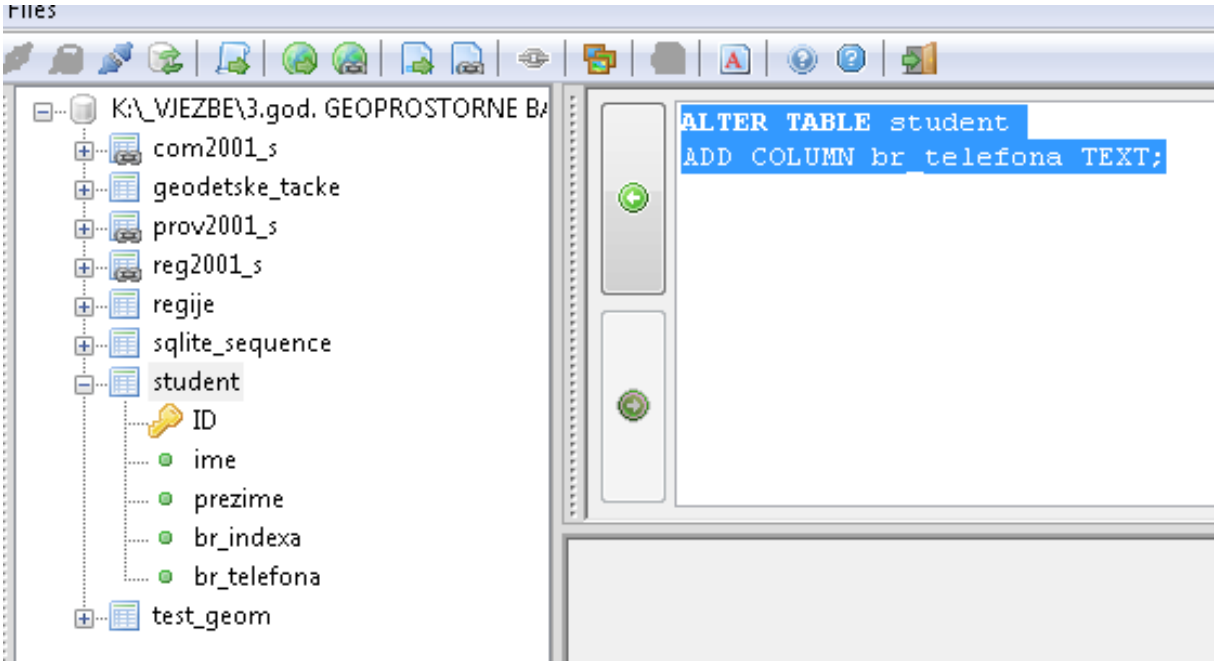
	ID	regija	povrsina(km2)
1	1	PIEMONTE	25395.4238
2	2	VALLE D'AOSTA	3258.4059
3	3	LOMBARDIA	23866.5293

Primjer:

Kreirajmo sada tabelu student. Tabela student ima sljedeće kolone: ID, ime, prezime, br_indexa.

CREATE TABLE student (
ID integer,
ime TEXT,
prezime TEXT,
br_indexa TEXT);
Možemo dodati primarni ključ i ograničenja u tabelu, npr.:
CREATE TABLE student (
ID integer **NOT NULL PRIMARY** KEY,
ime TEXT,
prezime TEXT,
br_indexa TEXT);

Dodajmo u tabelu student kolonu br_telefona:
ALTER TABLE student
ADD COLUMN br_telefona TEXT;



Dodajmo dva reda u tabelu student:

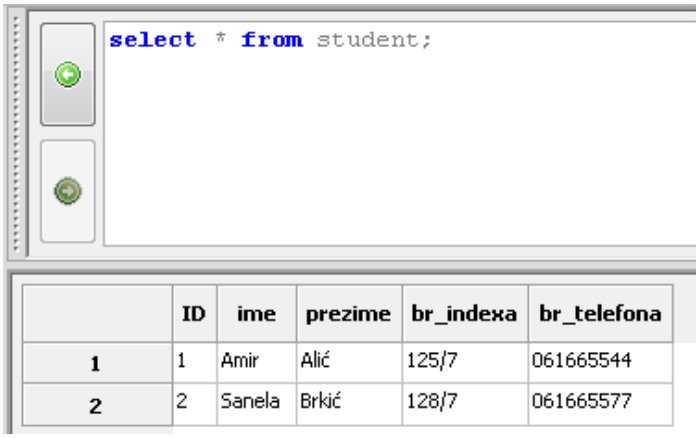
INSERT INTO student

VALUES (1,"Amir","Alić","125/7","061665544");

INSERT INTO student

VALUES (2,"Sanela","Brkić","128/7","061665577");

Prikažimo ažuriranu tabelu Student:



	ID	ime	prezime	br_indexa	br_telefona
1	1	Amir	Alić	125/7	061665544
2	2	Sanela	Brkić	128/7	061665577

Ako je potrebno dodati kolonu sa geometrijom, naredba za dodavanje takve kolone ima sljedeći oblik:

SELECT AddGeometryColumn(

ime_tabele,

ime_kolone,

EPSG_sifra_koordinatnog_sistema, tip_geometrije, dimenzionalnost);

Kreirajmo tabelu geodetske_tacke:

CREATE TABLE geodetske_tacke **(**

ID integer **NOT NULL PRIMARY** KEY, broj TEXT, tip TEXT**)**

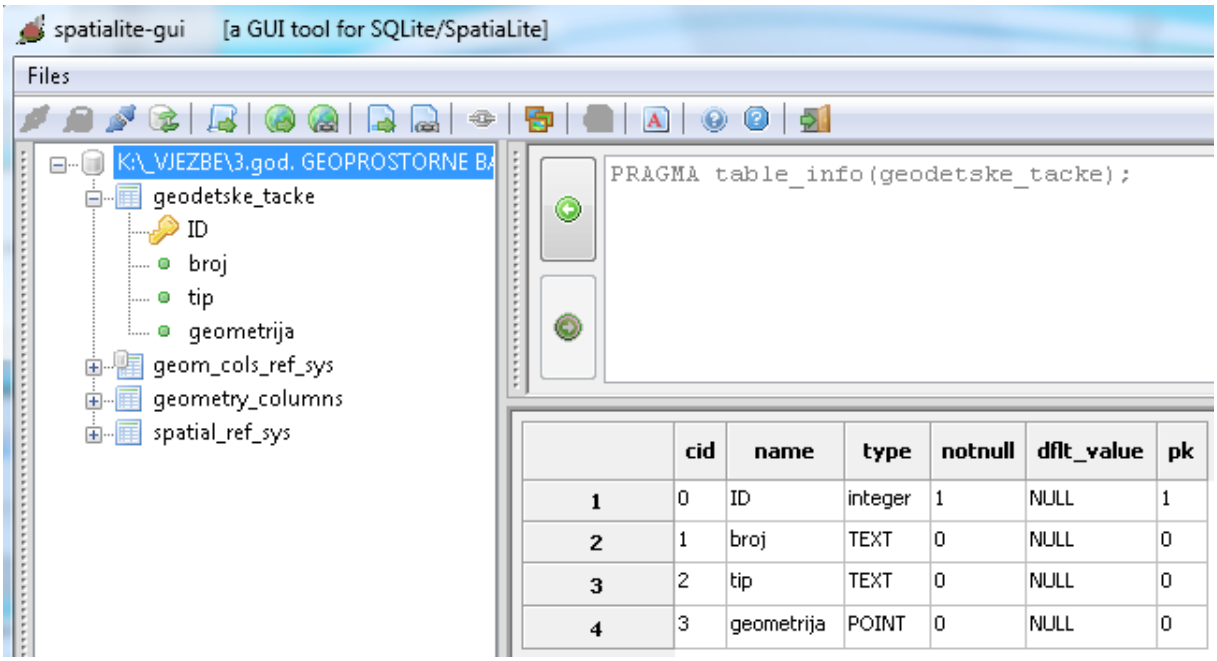
Napomena*: neposredno nakon kreiranja baze potrebno je pozvati metatabele naredbom:*

SELECT InitSpatialMetaData();

Dodavanje kolone sa geometrijskim podacima:

SELECT AddGeometryColumn('geodetske_tacke', 'geometrija', 3908, 'POINT', 2);

PRAGMA table_info(geodetske_tacke);



	cid	name	type	notnull	dflt_value	pk
1	0	ID	integer	1	NULL	1
2	1	broj	TEXT	0	NULL	0
3	2	tip	TEXT	0	NULL	0
4	3	geometrija	POINT	0	NULL	0

INSERT INTO geodetske_tacke (id, broj, tip, geometrija)

VALUES (1, 1575, "poligonska", **GeomFromText** ('POINT(1000 2000)', 3908));

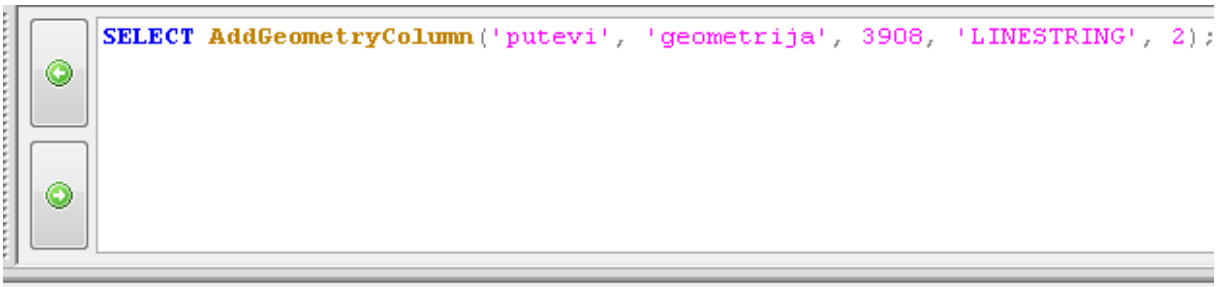
	ID	broj	tip	geometrija
1	1	1575	poligonska	BLOB sz=60 GEOMETRY

Kreiranje tabele putevi:

CREATE TABLE putevi (

ID integer **NOT NULL PRIMARY** KEY, naziv TEXT, tip TEXT);

SELECT AddGeometryColumn('putevi', 'geometrija', 3908, 'LINESTRING', 2);

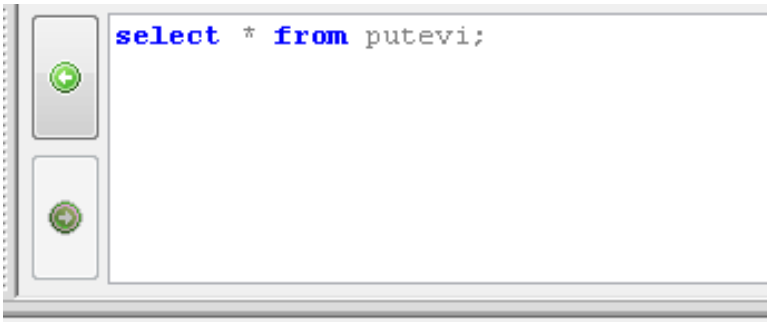


	cid	name	type	notnull	dflt_value	pk
1	0	ID	integer	1	NULL	1
2	1	naziv	TEXT	0	NULL	0
3	2	tip	TEXT	0	NULL	0
4	3	geometrija	LINESTRING	0	NULL	0

Dodavanje nove linije u tabelu

INSERT INTO putevi (id, naziv, tip, geometrija)

VALUES (11, "M-17", "magistralni", **GeomFromText** ('LINESTRING (100.0 200.0, 201.5 102.5, 1234.56 123.89)', 3908))



	ID	naziv	tip	geometrija
1	11	M-17	magistralni	BLOB sz=60 GEOMETRY

Zadaci za vježbu:

1. Dopunite tabele geodetske_tacke i putevi sa minimalno 5 redova.
2. Kreirajte tabelu Rijeke i dodajte minimalno 5 redova u tabelu.
3. Kreirajte tabelu koja će sadržavati objekte poligonalnog tipa.