

# Napredne baze podataka

Ponavljanje



# • Baza podataka

- Problematika?

- Pohranjivanje i organizacija veće količine podataka u vanjskoj memoriji računala

- Ideja?

- Omogućiti aplikacijama korištenje zajedničke i objedinjene kolekcije podataka (umjesto da pojedina aplikacija stvara vlastite datoteke na disku)

- Slijed

- Aplikacija ne pristupa izravno podacima na disku
  - Pristupa posredno služeći se specijaliziranim softverom

- Organizacija podataka

- Raspoređivanje podataka unutar tablice – ‘plošna baza’
  - Izbjegavanje redundancije – baza podataka

# •Baza podataka

## definicija

*Skup međusobno povezanih podataka, pohranjenih zajedno, uz isključenje bespotrebne zalihosti (**redundancije**), koji mogu zadovoljiti različite primjene.*

*Podaci su pohranjeni na način neovisan o programima koji ih koriste.*

*Prilikom dodavanja novih podataka, mijenjanja i pretraživanja postojećih podataka, primjenjuje se zajednički i kontrolirani pristup.*

*Podaci su strukturirani tako da služe kao osnova za razvoj budućih primjena.*

*(J. Martin, 1979)*



# • **Sustav za upravljanje bazom podataka (SUBP)**

Data Base Management System (DBMS)

- Poslužitelj (server) baze podataka
- Specijalizirani softver koji posreduje između aplikacija i podataka
- Obavlja sve operacije nad podacima
  - Definiranje baze podataka (Database Definition)
  - Rad s podacima (Database Management)
- Zadaci DBMS-a su osiguravanje:
  - Dostupnosti podataka
  - Integriteta podataka
  - Sigurnosti podataka
  - Neovisnosti podataka
- DB2 (IBM), Oracle, MS SQL Server, MySQL

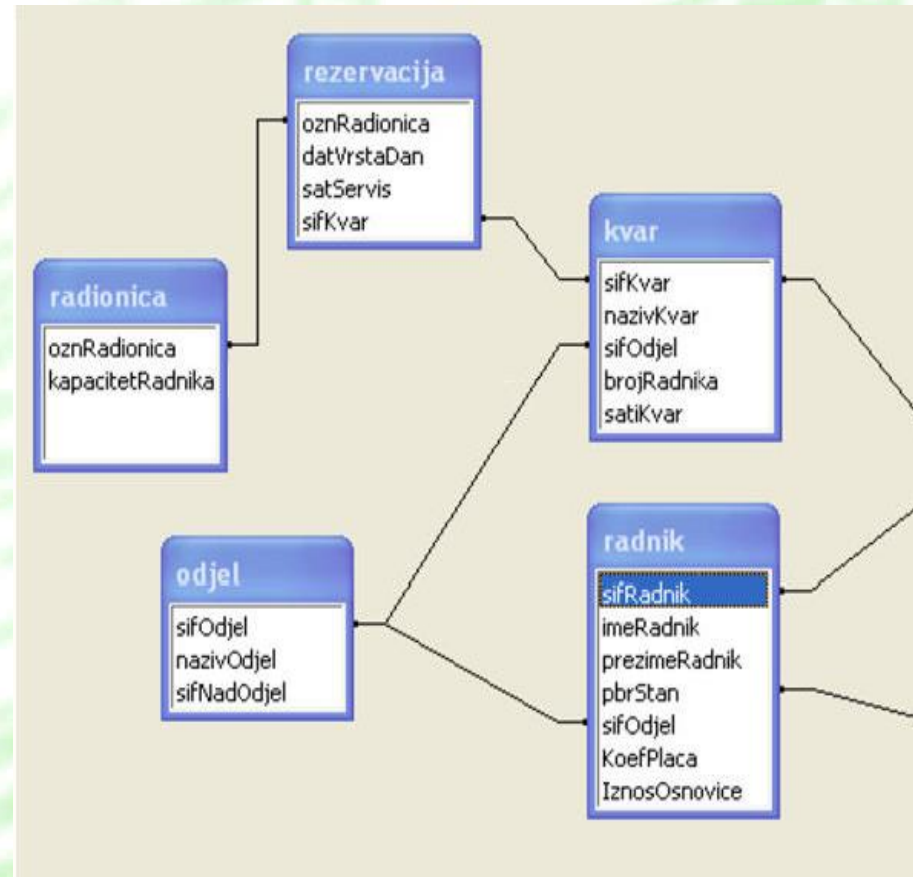
# • Oblikovanje baze podataka

- Konceptualno
  - Konceptualna shema baze, sastavljena od entiteta, atributa i veza
- Logičko
  - Logička shema (koja se u slučaju relacijskog modela sastoji od relacija)
  - Normalizacija
- Fizičko
  - Fizička shema = niz SQL naredbi kojima se relacije iz logičke sheme realiziraju u tablice
  - Dodavanje mehanizama za očuvanje integriteta i sigurnosti podataka
  - DBMS oblikuje fizički prikaz baze u skladu s traženom logičkom strukturom

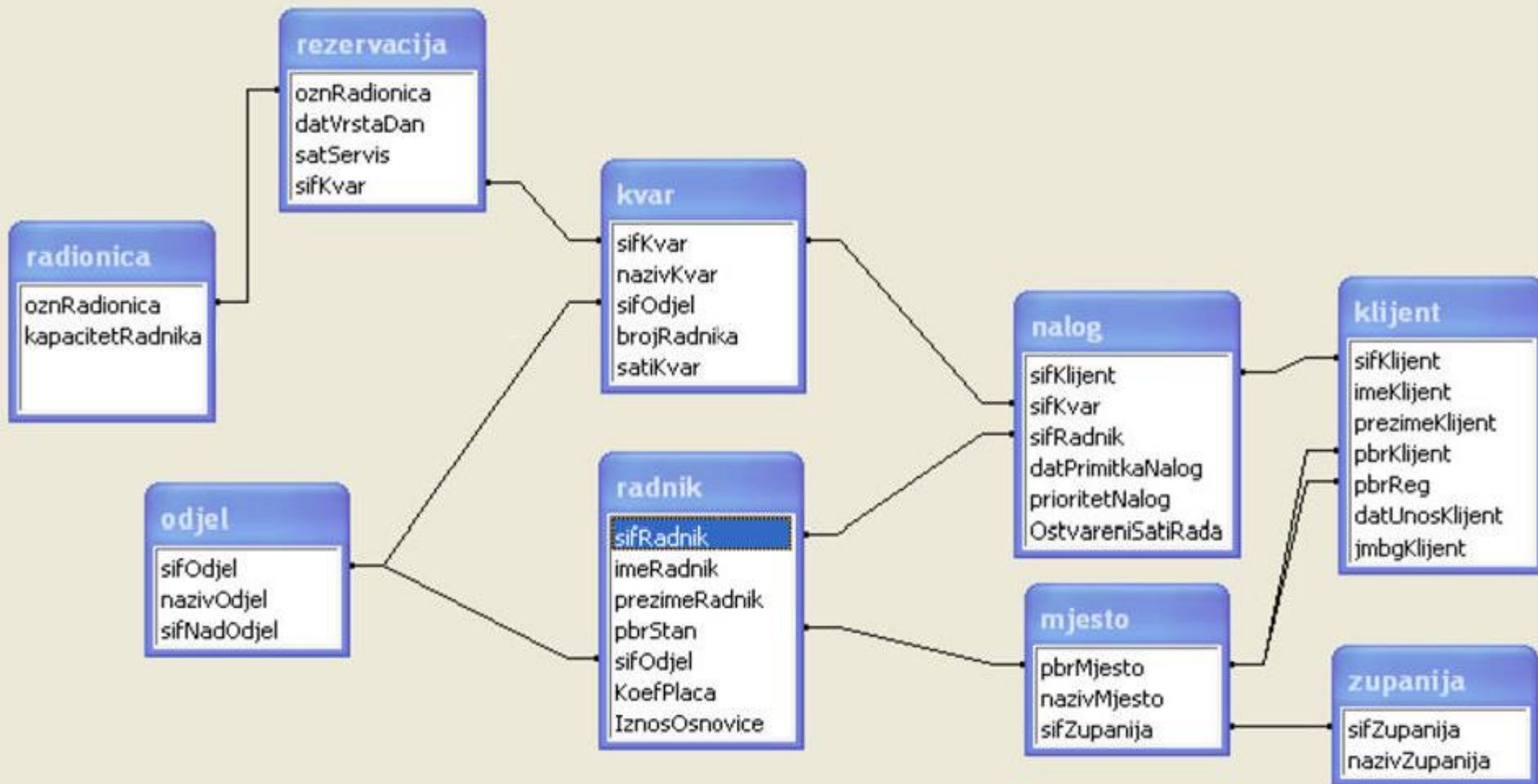
# • Relacijski model

Logičko oblikovanje baze podataka

- ENTITET
- ATRIBUT
  - Tip atributa
  - Domena
- N-TORKA
  - (redak, zapis)
- KLJUČEVI
  - Primarni ključ (*Primary Key*)
  - Strani ključ (*Foreign Key*)



# • Baza s labosa (autoradionica)



# • Baza podataka

## • Entitet: klijent

Shema baze podataka

Atributi klijenta

sifKlijent	imeKlijent	prezimeKlijent	pbrKlijent	pbrReg	datUnosKlijent	jmbgKlijent
1137	Jure	Ribarić	22000	22000	1986-10-29	2910986392304
1139	Niko	Marušić	48000	48000	1987-08-19	1908987173977
1140	Davor	Vurnek	20000	20000	1987-10-26	2610987300802
1141	Zoran	Habajec	21000	21000	1987-07-22	2207987301807

Primarni ključ

Opis entiteta  
n-torka

Strani ključ

Još jedan  
Mogući primarni ključ

Podaci baze podataka (tijelo)



# •Konceptualno oblikovanje

- **ENTITETI**

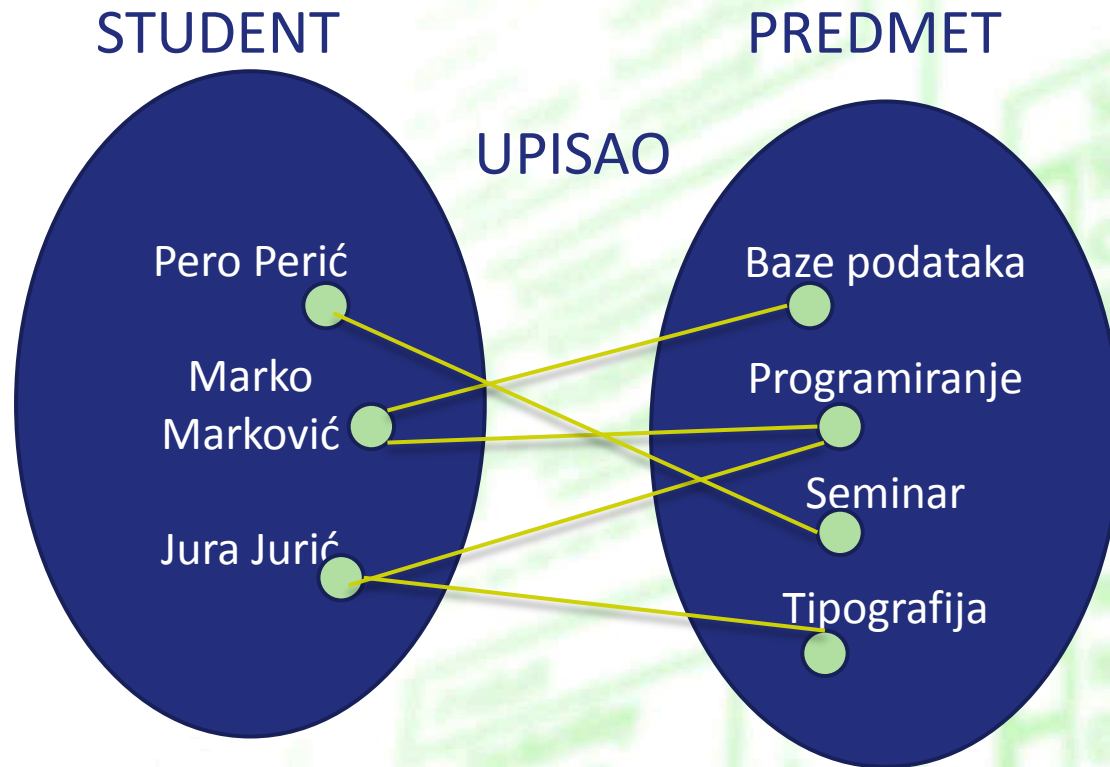
- **VEZE**

- 1:1
- 1:M
- M:1
- M:M

- **ATRIBUTI**

- **Primjer:**

- M:M (student, predmet)
- 1:1 (pročelnik, zavod)
- M:1 (nastavnik, zavod)



# •SQL

## Structured Query Language

- Medij za komunikaciju s bazom (DBMS-om) -> RDMBS
- Jezik relacijskih baza podataka
- Jezik za pristupanje podacima
- Neproceduralni odnosno deklarativni jezik
- Prvi komercijalni DBMS koji je podržavao SQL je Oracle (1979)

# •SQL

## Structured Query Language

### DATA DEFINITION LANGUAGE (DDL)

**CREATE**  
**ALTER**  
**DROP**

#### DATA CONTROL LANGUAGE (DCL)

- Dio DDL-a (Napredne baze podataka 😊)
- Naredbe za kontrolu pristupa bazi i dodjelu dozvola nad podacima

**GRANT**  
**REVOKE**

### DATA MANIPULATION LANGUAGE (DML)

**INSERT**  
**UPDATE**  
**SELECT**  
**DELETE**

# • DDL

- definiranje baze i objekata uključujući kreiranje, izmjenu i brisanje tablica te postavljanje indeksa

**CREATE** Database, Table, View, Trigger, Index, ...

```
CREATE TABLE `mjesto` (  
  `pbrMjesto` int(11) DEFAULT NULL,  
  `nazivMjesto` varchar(255) COLLATE  
    cp1250_croatian_ci DEFAULT NULL,  
  `sifZupanija` int(11) DEFAULT NULL,  
  UNIQUE KEY `nazivMjesto` (`nazivMjesto`),  
  UNIQUE KEY `pbrMjesto` (`pbrMjesto`),  
  KEY `FK_mjesto` (`sifZupanija`)  
) ENGINE=InnoDB DEFAULT CHARSET=cp1250  
COLLATE=cp1250_croatian_ci CHECKSUM=1  
DELAY_KEY_WRITE=1 ROW_FORMAT=DYNAMIC
```

**DROP TABLE** radnik;

ŠTO SE BRIŠE?



# • DML - INSERT

Unos podataka u tablicu

- 3 načina (unos vrijednosti za sve / neke attribute):

```
INSERT INTO mjesto VALUES (10000, 'Zagreb', 1);
```

```
INSERT INTO mjesto (pbrMjesto, nazivMjesto)  
VALUES (10000, 'Zagreb');
```

```
INSERT INTO mjesto SET pbrMjesto=10000,  
nazivMjesto='Zagreb', sifZupanija=1;
```

- Naprednije (podupit):

```
INSERT INTO mjesto  
SELECT * FROM neko_drugo_mjesto....
```

# • DML - UPDATE

Izmjena postojećih podataka

- Potrebno je navesti tablicu i sve attribute koje je potrebno promijeniti

```
UPDATE radnik SET sifOdjel=3;
```

```
UPDATE radnik  
    SET radnik.imeRadnik = „Ivan”,  
        radnik.prezimeRadnik = "Horvat",  
        radnik.koefplaca = 2  
    WHERE radnik.sifRadnik = 146;
```

# • DML - DELETE

Brisanje postojećih podataka

- Potrebno je navesti tablicu iz koje se briše podatak



Treba li \* ?

**DELETE** FROM radnik;

**DELETE** FROM radnik  
WHERE radnik.sifRadnik = 122;

# • Selekcija VS. projekcija

## SELEKCIJA

- Izdvajanje n-torki
- Kako? Upotrebom uvjeta u WHERE dijelu upita!
- Ispisati sve kvarove čiji naziv započinje s riječi *Zamjena*.

```
SELECT *  
    FROM kvar  
    WHERE nazivKvar LIKE 'Zamjena%';
```

ili striktno i pomoću regularnih izraza:

```
SELECT kvar.*  
    FROM kvar  
    WHERE kvar.nazivKvar REGEXP '^Zamjena';
```



# • Selekcija VS. projekcija

## PROJEKCIJA

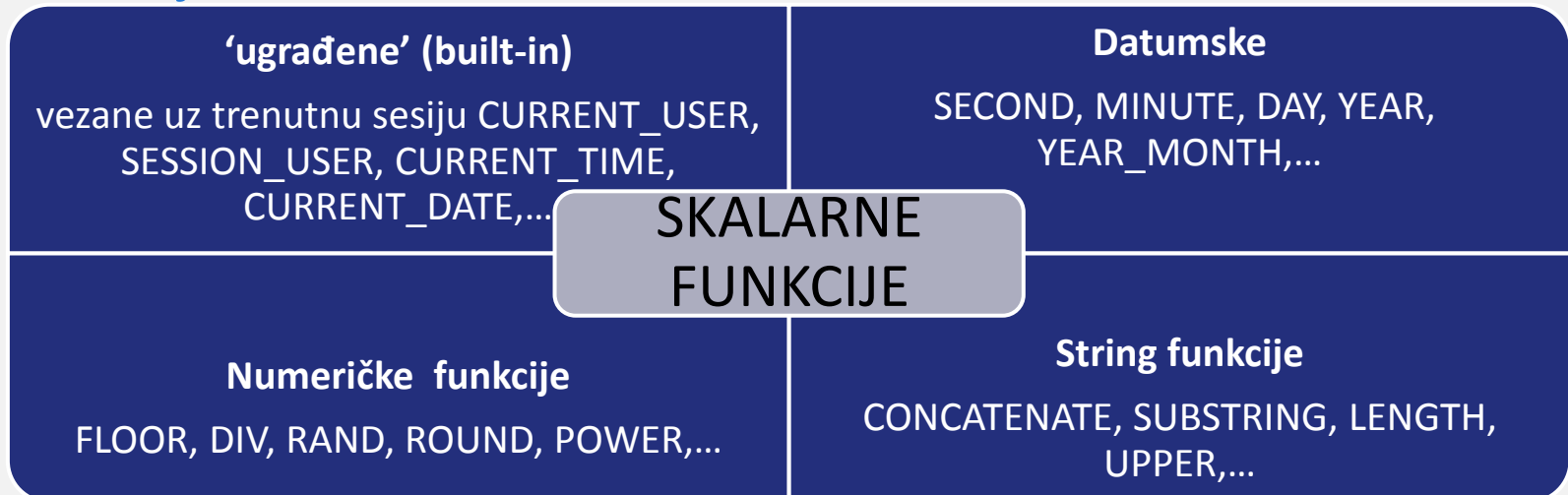
- Filtriranje atributa (stupaca)
- Ispisati poštanske brojeve svih mjesta u kojima žive radnici (koji su zaposleni u autoradionici).

```
SELECT pbrStan FROM radnik;
```

- Ako ispisujemo sve attribute, tada se rezultat ne smatra pravom projekcijom
- Najčešće: kombinacija selekcija & projekcija

# •Skalarne i agregatne funkcije

- **Skalarne** funkcije rade nad jednom vrijednošću; rezultat je jedna vrijednost



- **Agregatne** funkcije - odnose se na operacije nad skupinom vrijednosti; za rezultat daju jednu vrijednost



# •Skalarne i agregatne funkcije

- Ispisati ukupnu sumu sati kvara (*satiKvar*) iz tablice *kvar*. Stupcu s rezultatom dodijeliti naziv *Ukupno sati*.

```
SELECT SUM(satiKvar)
      AS 'Ukupno sati'
FROM kvar;
```

- Od imena klijenata napraviti e-mail adrese pisane malim slovima s nastavkom *@tvz.hr*. Neka atribut poprimi naziv *novi mail*. Rezultat sortirati abecedno (uzlazno).

```
SELECT LCASE(CONCAT(imeKlijent, '.',
                    prezimeKlijent, '@tvz.hr'))
      AS "novi_mail"
FROM klijent ORDER BY 1;
```



**ASC /  
DESC ?**

# •Projekcija s više tablica

- Kartezijev produkt
  - $M \times N$  rezultata (n-torki) -> rezultat je umnožak stupaca tablica
  - Pri čemu je M broj n-torki tablice *tbl1*, N broj n-torki tablice *tbl2*
- Prirodni spoj
  - Spajanje dvije ili više tablica koje su vezane relacijski preko određenih stupaca u tim tablicama (ključevi)
  - Broj rezultata?
    - Ne možemo znati rezultat izvođenja unaprijed (ne znamo koja će se n-torka iz *tbl1* spojiti s kojom n-torkom u *tbl2*)
- JOIN
  - Algoritam spajanja brži od prirodnog spoja (hash algoritmi)
  - Unutarnji spoj: (INNER) JOIN, NATURAL JOIN
  - Vanjski spoj: LEFT (OUTER) JOIN, RIGHT (OUTER) JOIN, FULL (OUTER) JOIN



# •Projekcija s više tablica

- Ispisati ime i prezime klijenta te mjesto stanovanja za sve klijente koji se prezivaju Babić.

```
SELECT imeKlijent, prezimeKlijent, nazivMjesto  
FROM klijent, mjesto  
WHERE klijent.pbrKlijent=mjesto.pbrMjesto  
AND prezimeKlijent='Babić';
```

VEZA

ili

UVJET

```
SELECT imeKlijent, prezimeKlijent, nazivMjesto  
FROM klijent JOIN mjesto  
ON  
klijent.pbrKlijent=mjesto.pbrMjesto  
WHERE prezimeKlijent = 'Babić';
```

VEZA

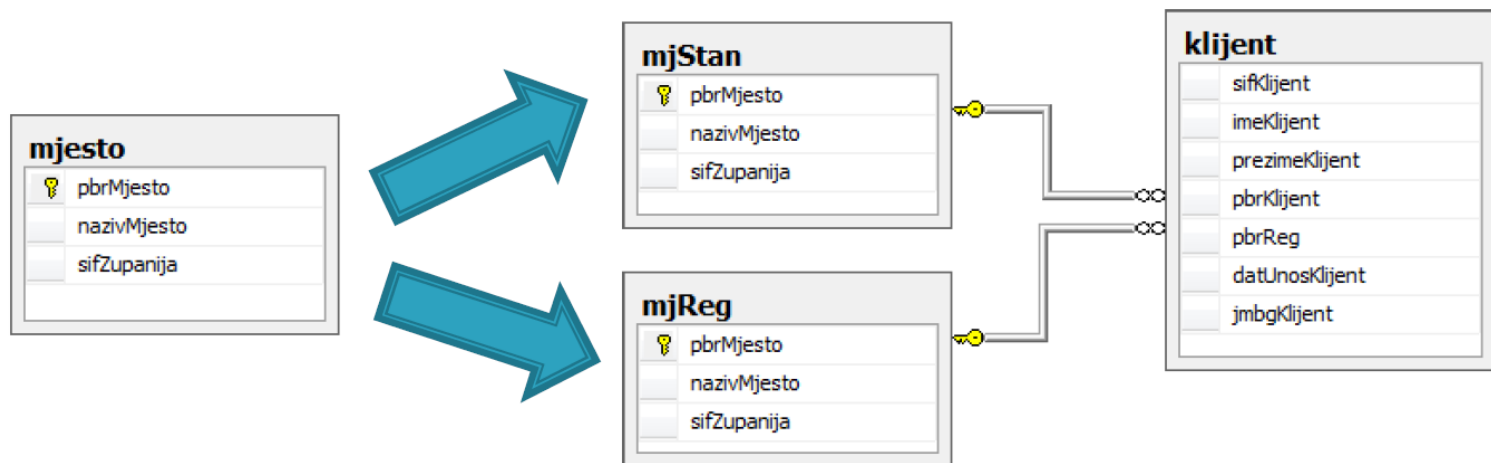
UVJET

# • Aliasi

- Zamjenski nazivi za
  - Attribute
  - Tablice
- Kako i kada koristiti aliase na nazive (kojih) tablica?
- Problematika – koristimo istu tablicu dva ili više puta, ali za dobivanje različitih podataka
- ***Izbjegavanje paralelne veze!***

# • Aliasi

- Ispisati za svakog klijenta njegovo ime i prezime te nazive mjesta gdje stanuje i gdje je registrirao vozilo.



```
SELECT imeKlijent, prezimeKlijent,
       mjStan.nazivMjesto, mjReg.nazivMjesto
FROM klijent
JOIN mjesto AS mjStan
      ON pbrKlijent=mjStan.pbrMjesto
JOIN mjesto AS mjReg
      ON pbrReg= mjReg.pbrMjesto
```

# • Aliasi

- Ispisati nazive mjesta stanovanja klijenata i stanovanja radnika po nalogima klijenata za radnike, uz uvjet da klijent i radnik žive u različitom gradu ali u istoj županiji.

```
SELECT imeRadnik, prezimeRadnik, imeKlijent, prezimeKlijent,  
mjKlijent.nazivMjesto, mjRadnik.nazivMjesto,  
zupRadnik.nazivZupanija, zupKlijent.nazivZupanija  
FROM nalog NATURAL JOIN klijent  
NATURAL JOIN radnik  
JOIN mjesto AS mjKlijent ON  
    klijent.pbrKlijent=mjKlijent.pbrMjesto  
JOIN mjesto AS mjRadnik ON  
    radnik.pbrStan=mjRadnik.pbrMjesto  
JOIN zupanija AS zupKlijent ON  
    zupKlijent.sifZupanija=mjKlijent.sifZupanija  
JOIN zupanija AS zupRadnik ON  
    zupRadnik.sifZupanija=mjRadnik.sifZupanija  
WHERE mjKlijent.pbrMjesto!=mjRadnik.pbrMjesto  
AND zupRadnik.sifZupanija=zupKlijent.sifZupanija ;
```

Može li  
kraće?



# •Projekcija – grupiranje podataka

- GROUP BY

- Grupiranje rezultata prema podudarajućim vrijednostima određenog atributa u jedan rezultat (redak)
- Prilikom izvođenja agregatnih funkcija
- U GROUP BY klauzulu potrebno je smjestiti **sve** attribute koji nisu u agregatnoj funkciji, a ispisuju se

- HAVING

- Filtriranje grupe rezultata

```
SELECT atr1, atr2, agregatna_funkcija  
FROM ime_tablice  
GROUP BY atr1, atr2  
HAVING uvjet  
ODRER BY atr;
```

# •Projekcija – grupiranje podataka

GROUP BY

- Ispisati ukupan broj mjesta po županiji:

```
SELECT nazivZupanija,  
       COUNT(mjesto.pbrMjesto) AS brMjesta  
FROM mjesto  
INNER JOIN zupanija ON mjesto.sifZupanija =  
zupanija.sifZupanija  
GROUP BY nazivZupanija
```

## Napomena:

Kako bi rezultati bili smisleni,  
potrebno je ispisati i  
atribut po kojem se grupira! U  
protivnom se neće znati  
koji rezultat pripada kojoj grupi.


nazivZupanija	brMjesta
Bjelovarsko-bilogorska	11
Brodsko-posavska	11
Dubrovačko-neretvanska	19
Grad Zagreb	6
Istarska	14
Karlovačka	13
Koprivničko-križevačka	7
Krapinsko-zagorska	10
Ličko-senjska	10
Međimurska	5
Osječko-baranjska	19
Požeško-slavonska	9
Primorsko-goranska	19
Sisačko-buzovačka	11
Šibenik	11
Vukovarsko-srijem	11

# •Projekcija – grupiranje podataka

## HAVING

- Ispisati koliki je prosječni koeficijent plaće u svakom odjelu. Potrebno je ispisati samo one odjele u kojima je taj prosjek veći od 1.5.

```
SELECT sifOdjel, AVG(koefPlaca)
FROM radnik
GROUP BY sifOdjel
HAVING AVG(koefPlaca)>1.5;
```



U WHERE dijelu ne  
možemo imati agregatnu  
funkciju

# • Podupiti

- Podupit kao vrijednost
  - Podupit prilikom ovog načina korištenja mora vraćati samo jednu vrijednost
  - Usporedba izraza iz vanjskog upita s rezultatom podupita
  - Izbjegavanje agregatne funkcije u WHERE dijelu upita
- Sadrži li podupit vrijednost
  - Ispitivanje je li neki izraz iz vanjskog upita sadržan u rezultatu podupita
  - Naredba IN
- Provjera skupa vrijednosti
  - Ispitivanjem je li se kao rezultat podupita pojavljuje barem jedna n-torka
  - Naredbe ANY i ALL



# • Podupiti

Podupit kao vrijednost

- Ispisati radnike koji imaju koeficijent plaće veći od prosječnog.

```
SELECT sifRadnik, koefPlaca
      FROM radnik
     WHERE koefPlaca >
           (SELECT AVG(koefPlaca) FROM radnik);
```

- Ispisati koliki je prosječni koeficijent plaće u svakom odjelu. Potrebno je ispisati samo one odjele u kojima je taj prosjek veći od prosječnog koeficijenta plaće po svim radnicima.

```
SELECT nazivOdjel, AVG(koefPlaca) AS prosjek
      FROM radnik NATURAL JOIN odjel
     GROUP BY nazivOdjel
    HAVING prosjek > (SELECT AVG(koefPlaca) FROM radnik)
   ORDER BY prosjek;
```

# •Podupiti

Provjera skupa vrijednosti

- Ispisati sve radnike u čijim mjestima stanovanja ne živi ni jedan klijent.

```
SELECT * FROM radnik  
WHERE pbrStan NOT IN  
(SELECT pbrKlijent FROM klijent);
```

- Ispisati sve radnike u čijim mjestima stanovanja živi barem jedan klijent:

```
SELECT * FROM radnik  
WHERE pbrStan IN  
(SELECT pbrKlijent FROM klijent);
```

# •Podupiti

Sadrži li podupit vrijednost

- Ispisati sve radnike u čijim mjestima stanovanja ne živi ni jedan klijent (isti primjer kao i s naredbom NOT IN):

```
SELECT * FROM radnik  
WHERE pbrStan <> ALL  
      (SELECT pbrKlijent FROM klijent);
```

- Ispisati kvarove kod kojih su vrijednosti atributa satiKvar veće od svih vrijednosti ostvarenih sati rada iz tablice nalog. (Upit ne vraća rezultat!)

```
SELECT * FROM kvar  
WHERE satiKvar > ALL  
      (SELECT ostvareniSatiRada FROM nalog);
```

# •Datumi

- Ispišite sve klijente koji su u bazu uneseni između siječnja 2011 (uključujući) i kolovoza 2013 (uključujući).
- ```
SELECT * FROM klijent  
WHERE datUnosKlijent>='2011-01-01'  
AND datUnosKlijent<='2013-08-31';
```
- Ispišite sve klijente koji su u bazu uneseni prije 1988. godine.
- ```
SELECT * FROM klijent  
WHERE YEAR(datUnosKlijent)<1988;
```

Dohvat trenutnog vremena?

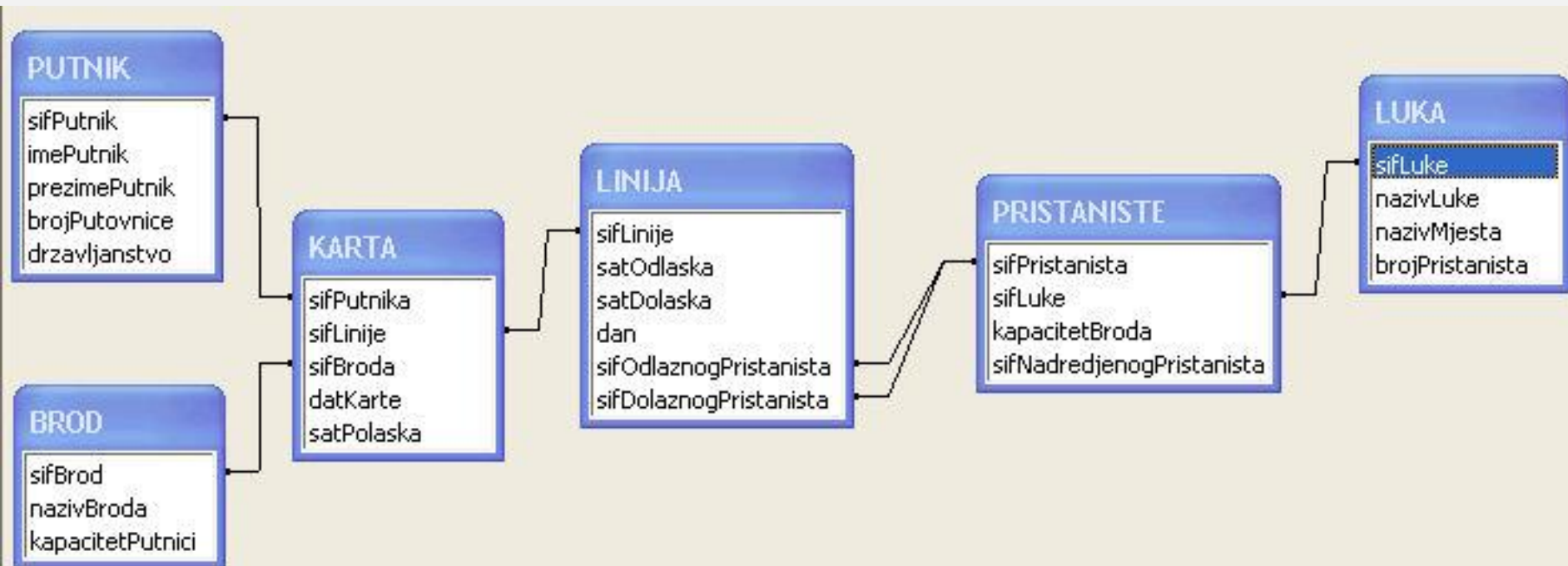


# •Zadaci s roka

- Napisati naredbu za stvaranje tablice *karta* sa svim njezinim atributima.

CREATE TABLE karta

(sifPutnika INT, sifLinije INT, sifBroda INT,  
datKarte DATE, satPolaska INT);



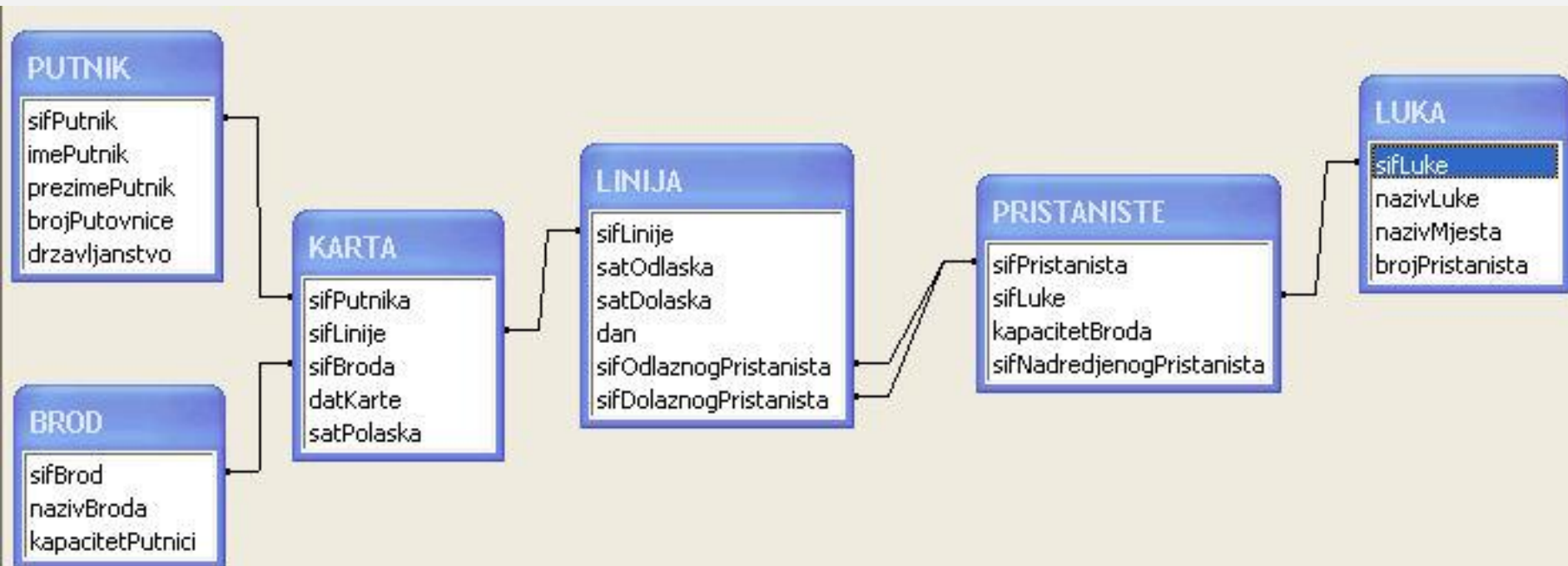
# •Zadaci s roka

- U tablici brod postaviti atribut *kapacitetPutnici* na 200 za brodove naziva *Galeb*.

UPDATE brod

SET kapacitetputnici=200

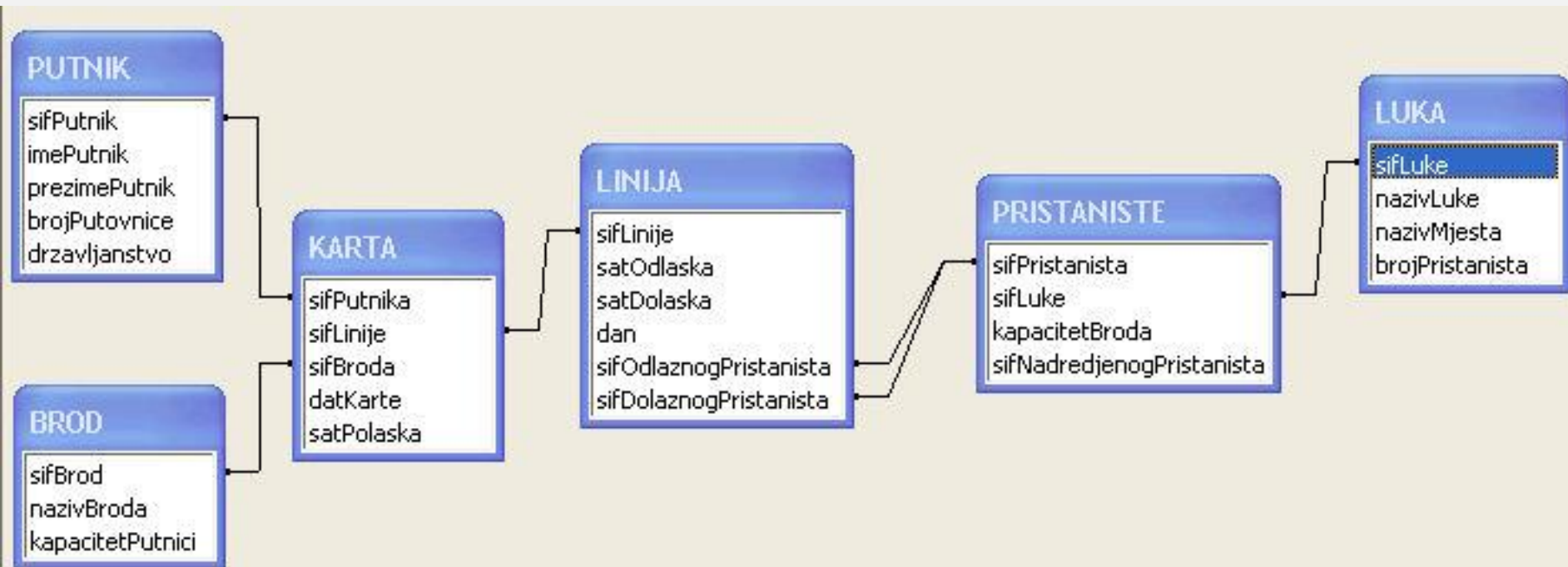
WHERE nazivBroda='Galeb' ;



# •Zadaci s roka

- Umetnite novi zapis u tablicu *brod* s podacima:  
sifBroda: 1000, naziv broda: Galeb.  
Svi ostali podaci neka budu postavljeni na NULL vrijednosti.

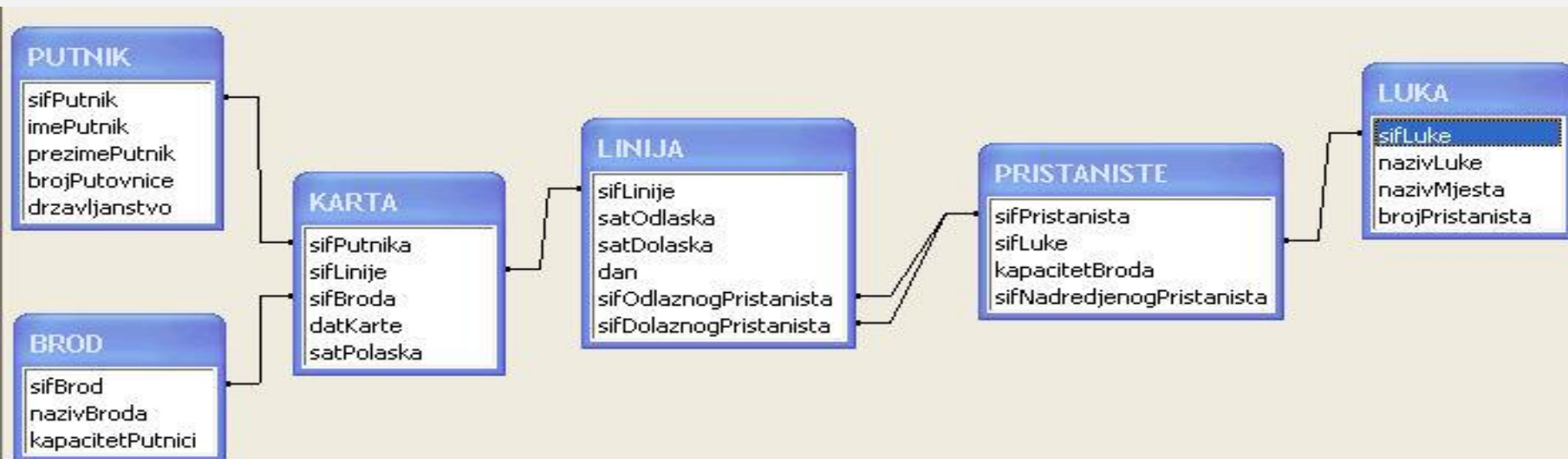
```
INSERT INTO brod VALUES (1000, 'Galeb', NULL);
```



# •Zadaci s roka

- Ispisati sve atribute iz tablice *linija* kod kojih je naziv odlaznog pristaništa *Dubrovnik*. Ispis poredati po satu dolaska (veći -> manji), a zatim po satu odlaska (manji -> veći).

```
SELECT linija.* FROM linija,pristaniste,luka
WHERE linija.sifodlaznogpristanista=pristaniste.sifpristanista
AND pristaniste.sifLuke=luka.sifLuke
AND luka.nazivluke='Dubrovnik'
ORDER BY linija.satDolaska DESC, linija.satOdlska ASC;
```

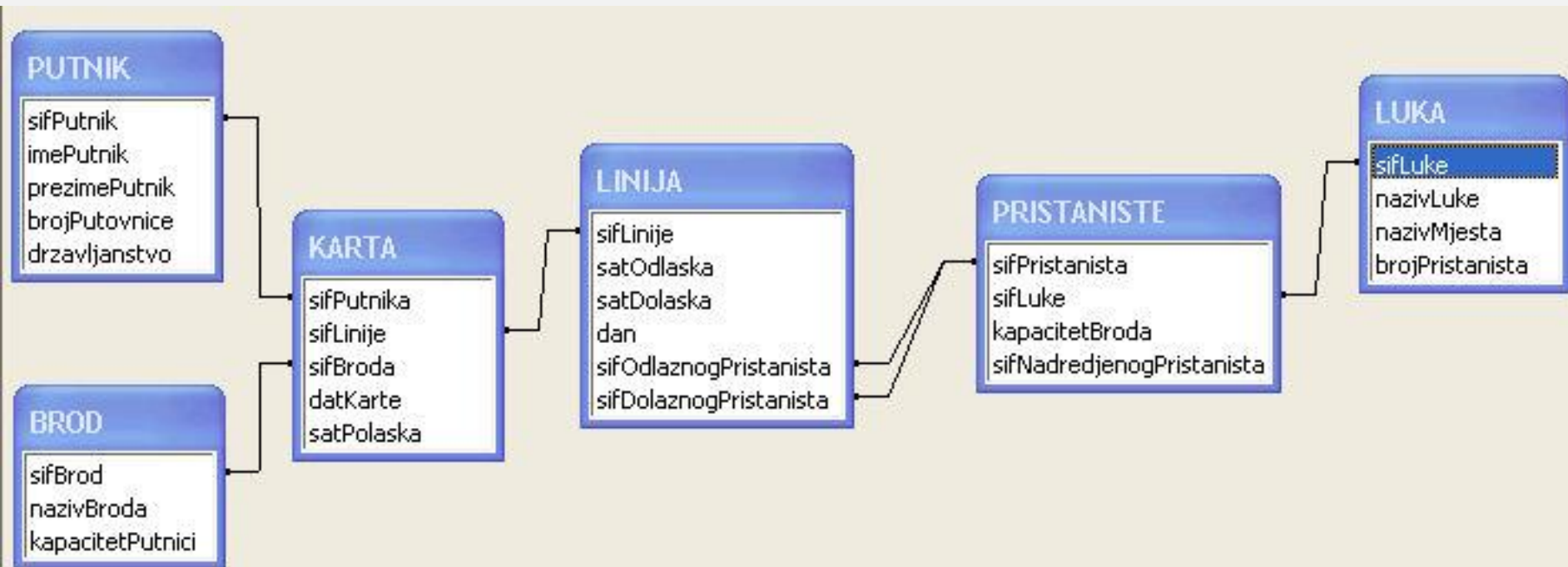




## •Zadaci s roka

- Ispisati broj zapisa iz tablice *brod*. Neka se broj zapisa zove *broj*.

```
SELECT COUNT(*) as broj FROM brod;
```



# •Zadaci s roka

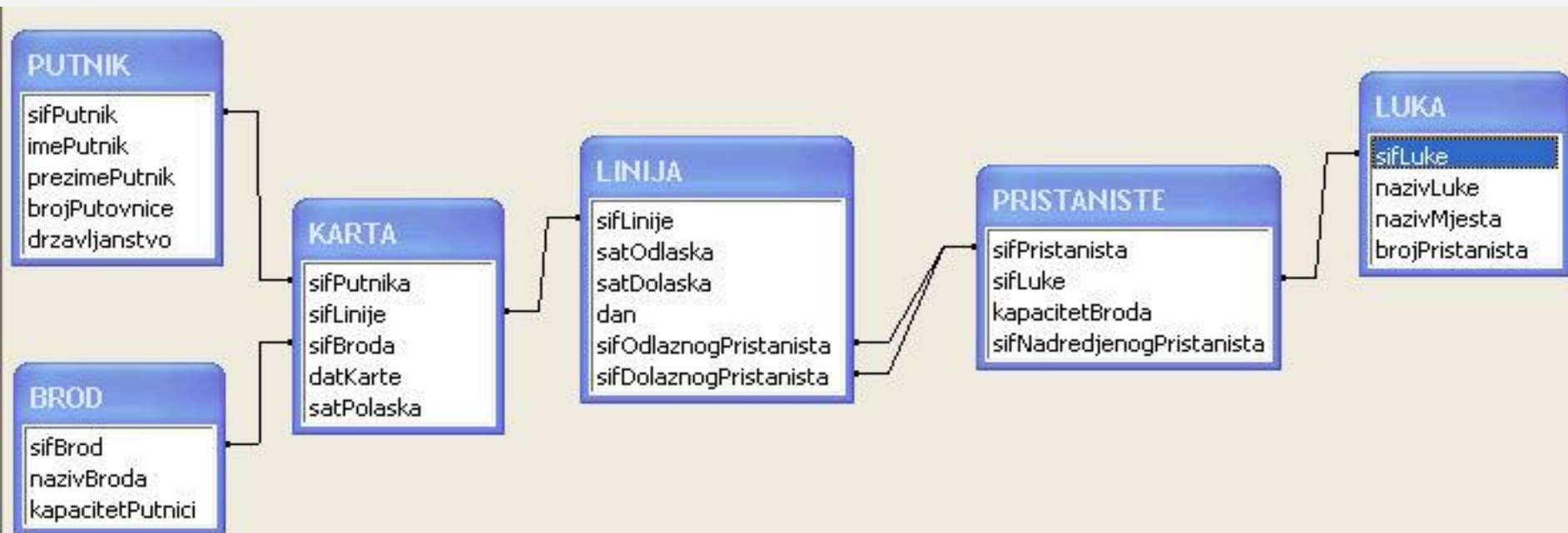
- Ispišite sve atribute iz tablice *karta* za linije čiji je naziv luke odlaznog pristaništa *Hvar*, a dolaznog pristaništa *Vis*. Ispisati samo linije za prošlu godinu.

```
SELECT karta.* FROM karta,linija, pristaniste P1, pristaniste P2,  
      luka L1, luka L2  
WHERE karta.siflinije=linija.siflinije  
      AND linija.sifodlaznogpristanista=P1.sifpristanista  
      AND linija.sifdolaznogpristanista=P2.sifpristanista  
      AND p1.sifluke=L1.sifluke AND P2.sifluke=L2.sifLuke  
      AND L1.nazivluke='Hvar' AND L2.nazivluke='Vis'  
      AND YEAR(karta.datKarte)=YEAR(CURDATE()) -1;
```



# •Zadaci s roka

- Ispisati sve atribute iz tablice *pristanište* za ona pristaništa kod kojih je *kapacitetBroda* veći od najvećeg kapaciteta putnika iz tablice *brod*.
- ```
SELECT pristaniste.* FROM pristanista  
WHERE kapacitetBroda >  
(SELECT MAX(kapacitetPutnici) FROM brod);
```



# •Važno!

Podsjetiti se gradiva iz kolegija ***Baze podataka*** rješavajući zadatke i primjere iz ***skripte sa tog kolegija!***