1. **Naštejte in kratko opišite faze načrtovanja podatkovne zbirke (podatkovno orientiran pristop)**

**Načrtovanje je:**

* **kompleksen proces** (podatkovne zahteve, pojmovni načrt, logični načrt, fizčni načrt)
* **razgraditev procesa v posamezne podprocese**
  + konceptualno načrtovanje
  + logično načrtovanje
  + fizično načrtovanje

1. **Naštejte in kratko opišite koncepte podatkovnega modeliranja**

**Podatkovni modeli so sredstva za opis realnosti. Kakovost posameznega podatkovnega modela pomembno vpliva na končni rezultat - shemo.**

**Konceptualno načrtovanje podatkovne baze:**

* **Klasifikacija** (definira koncepte kot tipe objektov na osnovi skupnih lastnosti, na kateri e izvede tudi razvrščanje-klasificiranje; vsak list je član (MEMBER OF) korena.)
* **Agregacija** (definira nov tip iz množice tipov, ki predstavljajo komponente – so del (PART-OF) tipa.)
* **Generalizacija** (je postopek prirejanja skupnega nadtipa dvema ali več obstoječim tipom, listi SO (IS\_A) podtipi korena v enonivojskem razredu, kjer so vozlišča tipi. Najpomembnejša lastnost je DEDOVANJE.)

1. **Predstavite razširitve E-R s praktičnimi primeri.**

Vsaka entiteta je lahko element več generalizacijskih hierarhij, za katero lahko postavimo oceno pokritosti.

Lastnosti generalizacijske entitete E podedujejo tudi entitete E1, E2, ..., En.

1. **Naštejte in kratko opišite strategije za načrtovanje shem (načrtovanje kompleksnejših primerov)**

**FUNKCIONALNA ANALIZA:**

- specifikacija aplikacijskih zahtev

- funkcionalna shema - opis aktivnosti in informacijskih tokov

**NAČRTOVANJE APLIKACIJE:**

- funkcionalna shema

- specifikacija aplikacije - opis obnašanja aplikacijskega programa

**NAČRTOVANJE PROGRAMA APLIKACIJE:**

- specifikacija aplikacije

- programska koda

**PRIMERJAVA PODATKOVNO IN FUNKCIONALNO ORIENTIRANEGA PRISTOPA:**

-pristopa se dopolnjujeta

-združitev obeh pristopov

1. **Natančno predstavite izbrano strategijo za načrtovanje shem strukture zbirke podatkov.**

**UML (Unified Modeling Language)**

Ima veliko komponent za grafično modeliranje različnih vidikov programskih konceptov. UML razredni diagram (Class Diagrams) najbolj ustreza E-R diagramu. Vsebuje veliko sprememb

(E-R (entiteta in atributi) vs. UML (na vrhu je entiteta pod njo pa atributi); E-R (relacija binarna-ROMB) vs. UML (binarna relacija je predstavljena s črto, ime relacije je nad njo).

Pri UML je položaj ravno nasproten kot v E-R diagramih 🡪 kardinalnost).

1. **Naštejte probleme, ki se pojavijo pri integraciji različnih pogledov več konceptualnih modelov v en globalni konceptualni model.**

Neskladja med modeli (shemami), ki jih želimo združiti, nastopijo predvsem zaradi razlik v razumevanju in modeliranju zaključenega organiziranega sistema (oz. njegovega dela).

RAZLIČNE PERSPEKTIVE:

Isti objekti so oblikovani z različnih vidikov, ob uporabi različnih nivojev abstrakcije in ob upoštevanju različnih lastnosti.

1. **Predstavite namen uporabe uporabniških podatkovnih struktur (composite types) in zapišite primer ustvarjanja podatkovnega tipa strukture za podan primer.**
2. **Napišite strežniško funkcijo (plpgsql), ki v tabeli osebe preveri identifikacijo uporabnika. Struktura tabele: osebe(id\_osebe integer, priimek varchar(50), ime varchar(50), e\_naslov varchar (120), uporabnisko\_ime varchar(20), geslo varchar(20)) Funkcija: osebe\_identifikacija Vhodni parametri: uporabnik varchar, geslo varchar Rezultat: id\_osebe, če obstaja oseba s tem uporabniškim imenom in geslom.**

CREATE TABLE osebe

(

idOsebe SERIAL NOT NULL PRIMARY KEY,

ime VARCHAR (120),

priimek VARCHAR (120),

eNaslov VARCHAR (120),

uporabniskoIme VARCHAR (120),

geslo VARCHAR (120)

);

CREATE OR REPLACE FUNCTION osebe\_identifikacija

(i$uporabnik VARCHAR, i$geslo VARCHAR )

RETURNS INTEGER AS

$telo$

DECLARE

tIdOsebe INTEGER;

BEGIN

SELECT INTO tIdOsebe IdOsebe FROM osebe

WHERE osebe.uporabniskoIme = i$uporabnik AND osebe.geslo = i$geslo;

RETURN  tIdOsebe;

END;

$telo$

LANGUAGE 'plpgsql';

1. **Napišite strežniško funkcijo (plpgsql), ki v izbrano tabelo vstavi nov zapis oz. vrstico s podatki. Vrednost polja ključa tabele določite z objektom sequence, ki ga ustvarite.**

CREATE TABLE osebe

(

idOsebe SERIAL NOT NULL PRIMARY KEY,

ime VARCHAR (120),

);

CREATE SEQUENCE seq\_osebe START 1;

INSERT INTO osebe(idosebe,ime)

VALUES(nextval('seq\_osebe'), 'Gorazd');

CREATE OR REPLACE FUNCTION osebe

(i$idosebe INTEGER,i$ime VARCHAR)

RETURNS integer AS

$telo$

DECLARE

kljuc INTEGER;

BEGIN

kljuc = nextval('seq\_osebe');

INSERT INTO osebe(idosebe, ime)

VALUES (kljuc, i$ime);

RETURN kljuc;

END;

$telo$

LANGUAGE 'plpgsql';

1. **Napišite strežniško funkcijo (plpgsql), ki iz izbrane tabele izbriše zapis oz. vrstico. Kriterij brisanja je vrednost v polju ključa tabele, ki je vhodni parameter tabele. Funkcija v rezultatu vrne število izbrisanih zapisov tabele.** (enaka kot 11.naloga)
2. **Napišite strežniško funkcijo (plpgsql), ki iz izbrane tabele izbriše zapise oz. vrstice. Kriterij brisanja je vrednost dveh izbranih polj tabele, ki sta vhodna parametra tabele. Funkcija v rezultatu vrne število izbrisanih zapisov tabele.**

**Iz tabele osebe izbriše zapise, kriterij: priimek se začne z 'N' in oseba je iz kraja 'Maribor'.**

CREATE OR REPLACE FUNCTION osebe

(i$priimek VARCHAR, i$kraj VARCHAR)

RETURNS INTEGER AS

$telo$

DECLARE

steviloIzbrisanihVrstic INTEGER;

BEGIN

DELETE FROM osebe

WHERE UPPER(priimek) LIKE UPPER(i$priimek) || '%'

AND

idpostnestevilke IN (SELECT idpostnestevilke FROM postneStevilke

WHERE UPPER(krajmesto) = UPPER(i$kraj));

-- ugotovimo število izbrisanih vrstic

GET DIAGNOSTICS steviloIzbrisanihVrstic = ROW\_COUNT;

RETURN steviloIzbrisanihVrstic;

END;

$telo$

LANGUAGE 'plpgsql';

1. **Napišite strežniško funkcijo (plpgsql), ki za izbrano tabelo vrne seznam oz. tabelo dveh stolpcev, glede na izbran kriterij.**

CREATE TYPE tOsebe AS

(

ime VARCHAR (120),

priimek VARCHAR (120),

prekrsek VARCHAR (120)

);

CREATE OR REPLACE FUNCTION osebePrekrski

(i$prekrsek VARCHAR)

RETURNS SETOF tOsebe AS

$telo$

DECLARE

tVrstica tOsebe;

BEGIN

FOR tVrstica IN SELECT ime, priimek, prekrsek

FROM osebe LEFT OUTER JOIN

prekrski USING (idPrekrski)

WHERE

UPPER(prekrsek) LIKE upper(i$prekrsek i$) || '%'

LOOP

RETURN NEXT tVrstica;

END LOOP;

END;

$telo$

LANGUAGE 'plpgsql';

1. **Napišite prožilec, ki v primeru spreminjanja podatkov v izbrani tabeli evidentira spremembe v pomožni tabeli. Pomožno tabelo ustvarite z naslednjimi podatki: id serial(ključ), datum spremembe, uporabnik, operacija, vrednost ključa spremenjenega zapisa.**

CREATE TABLE log\_drzave

(

idLogDrzave INTEGER NOT NULL PRIMARY KEY,

datumSpremembe DATE,

uporabnik VARCHAR,

operacija VARCHAR,

idDrzave INTEGER

);

CREATE SEQUENCE seq\_log\_drzave START 1;

CREATE OR REPLACE FUNCTION drzave\_u ()

RETURNS TRIGGER AS

$telo$

BEGIN

IF UPPER(TG\_OP) = 'UPDATE' THEN

INSERT INTO log\_drzave

(idLogDrzave , datumSpremembe, uporabnik, operacija, idDrzave)

VALUES(nextval('seq\_log\_drzave'), now(),

current\_user, TG\_OP, OLD.iddrzavljanstvo);

END IF;

END;

$telo$

LANGUAGE 'plpgsql';

CREATE TRIGGER drzave\_TG

AFTER UPDATE

ON drzave

FOR EACH ROW

EXECUTE PROCEDURE drzave\_u();

1. **Napišite strežniško funkcijo, ki za izbrano tabelo vrne vse vrednosti in dodatno polje iz druge izbrane tabele, katero vrednost določa vrednost izbranega povezovalnega polja (tuj ključ).** (enaka kot 15.naloga)
2. **Napišite strežniško funkcijo, ki za izbrani načrt zbirke podatkov vrne zahtevan rezultat (statistična obdelava).**

CREATE TYPE tosebedrzavljanstvo AS

(

drzavljanstvo varchar (120),

delez numeric (4,2)

);

CREATE OR REPLACE FUNCTION osebedrzavljanstvostatistika()

RETURNS SETOF tosebedrzavljanstvo AS

$telo$

DECLARE

vrsticarezultat tosebedrzavljanstvo;

drzavljanstvo1 integer;

stevilooseb numeric (8,3);

tdrzavljanstvo drzavljanstvo%ROWTYPE;

tosebedrzavljanstvostatistika osebedrzavljanstvo%ROWTYPE;

BEGIN

SELECT INTO stevilooseb COUNT (\*) FROM osebe;

FOR tdrzavljanstvo IN SELECT \* FROM drzavljanstvo

LOOP

SELECT INTO drzavljanstvo1 COUNT (\*) FROM osebedrzavljanstvo WHERE

tdrzavljanstvo.iddrzavljanstvo = osebedrzavljanstvo.iddrzavljanstvo;

vrsticarezultat.delez = drzavljanstvo1 / stevilooseb;

vrsticarezultat.drzavljanstvo = tdrzavljanstvo.drzavljanstvo;

return NEXT vrsticarezultat;

END LOOP;

END;

$telo$ LANGUAGE 'plpgsql'

CREATE TYPE tzaposlen AS

(

izobrazba varchar (120),

delez numeric (4,2)

);

CREATE OR REPLACE FUNCTION zaposlenstatistika()

RETURNS SETOF tzaposlen AS

$telo$

DECLARE

vrsticarezultat tzaposlen; -- spremenljivka vrstico rezultata

izobrazba1 integer;

izbrazb numeric (8,3);

tizobrazba izobrazba%ROWTYPE;

tzaposleni zaposleni%ROWTYPE;

BEGIN

SELECT INTO izbrazb COUNT (\*) FROM zaposleni;

FOR tizobrazba IN SELECT \* FROM izobrazba

LOOP

SELECT INTO izobrazba1 COUNT (\*) FROM zaposleni WHERE

tizobrazba.idizobrazba = zaposleni.idizobrazba;

vrsticarezultat.delez = izobrazba1 / izbrazb;

vrsticarezultat.izobrazba = tizobrazba.izobrazba;

return NEXT vrsticarezultat;

END LOOP;

END;

$telo$ LANGUAGE 'plpgsql'

# **SINTAKSE FUNKCIJ**

**/\* insert, update funkcija \*/**

CREATE OR REPLACE FUNCTION drzave\_iu

(i$id\_drzave integer, i$drzava varchar, i$oznaka\_drzave varchar)

RETURNS integer AS

$body$

DECLARE

ID INTEGER;

BEGIN

IF (i$id\_drzave IS NULL OR i$id\_drzave = 0) THEN

ID := NEXTVAL('seq\_drzave');

BEGIN

INSERT INTO drzave (id\_drzave, drzava, oznaka\_drzave, zastava )

VALUES (ID,i$drzava, i$oznaka\_drzave, i$zastava);

RETURN ID;

ELSE

BEGIN

UPDATE drzave SET

drzava = i$drzava,

oznaka\_drzave = i$oznaka\_drzave,

zastava = i$zastava

WHERE id\_drzave = i$id\_drzave;

ID := i$id\_drzave;

RETURN ID;

END IF;

END;

$body$

LANGUAGE 'plpgsql';

**/\* delete funkcija \*/**

CREATE OR REPLACE FUNCTION drzave\_d

(id integer)

RETURNS integer AS

$body$

DECLARE

steviloVrstic INTEGER;

BEGIN

BEGIN

DELETE FROM drzave

WHERE ID = id\_drzave;

GET DIAGNOSTICS steviloVrstic := ROW\_COUNT;

RETURN steviloVrstic;

END;

$body$

LANGUAGE 'plpgsql';

**/\* select funkcija \*/**

CREATE OR REPLACE FUNCTION drzave\_s ()

RETURNS SETOF drzave AS

$body$

DECLARE

zapis drzave;

BEGIN

FOR zapis IN SELECT \* FROM drzave

LOOP

RETURN NEXT zapis;

END LOOP;

RETURN;

END;

$body$

LANGUAGE 'plpgsql';