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

**Načrtovanje je:**

* **kompleksen proces (podatkovne zahteve, pojmovni načrt, logični načrt, fizični 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) k vorena.)**

**i**

* **Agregacija (definira nov tip iz množice tipov, ki predstavljajo komponente – so del (PART-OF) tipa.)**
* **Generalizacija (je postopek prirejanja skupnega nad tipa dvema ali več obstoječim tipom, listi SO (IS\_A) podtipi korena v eno nivojskem 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 (angl. Unified Modeling Language) ima veliko komponent za grafično modeliranje različnih vidikov programskih konceptov. UML razredni diagram (angl. 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 osebe1 (**

**idOsebe SERIAL NOT NULL PRIMARY KEY,**

**priimek VARCHAR (30),**

**ime VARCHAR (30),**

**eNaslov VARCHAR (120),**

**uporabniskOIme VARCHAR (20),**

**geslo VARCHAR (20)**

**);**

**CREATE OR REPLACE FUNCTION osebe\_identifikacija(uporabnik VARCHAR, i$geslo VARCHAR ) RETURNS INTEGER AS**

**$telo$**

**DECLARE**

**tIdOsebe INTEGER; --vračanje rezultata funkcije**

**BEGIN**

**SELECT INTO tIdOsebe IdOsebe**

**FROM osebe1**

**WHERE uporabniskoIme = uporabnik AND**

**osebe1.geslo = i$geslo;**

**--vrnemo rezultat**

**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 drzava(iddrzava SERIAL PRIMARY KEY NOT NULL, |
| Ime\_drzave VARCHAR(50)); |
| CREATE SEQUENCE seq\_drzava START 1; |
| INSERT INTO drzava(iddrzava,ime\_drzave) |
| VALUES(nextval('seq\_drzava'), 'Slovenija'); |

|  |
| --- |
| CREATE TABLE drzava(iddrzava SERIAL PRIMARY KEY NOT NULL,  Ime\_drzave VARCHAR(50));  CREATE SEQUENCE seq\_drzava START 1;  INSERT INTO drzava(iddrzava,ime\_drzave)  VALUES(nextval('seq\_drzava'), 'Slovenija'); (Tudi dela)  CREATE OR REPLACE FUNCTION drzava(  i$iddrzava integer,  i$ime\_drzave varchar)  RETURNS integer AS  $body$  DECLARE  kljuc integer;  BEGIN  kljuc = nextval('seq\_drzava');  INSERT INTO drzava(iddrzava, Ime\_drzave)  VALUES (kljuc, i$ime\_drzave);  RETURN kljuc;  END;  $body$  LANGUAGE 'plpgsql' |

|  |
| --- |
| Entiteta “drzava” |
| CREATE TABLE log\_drzava  (  idlogdrzava integer not null primary key,  datumspremembe date,  uporabnik varchar,  operacija varchar,  iddrzava integer  );  CREATE SEQUENCE seq\_log\_drzava START 1;  CREATE OR REPLACE FUNCTION drzavaTG()  RETURNS TRIGGER AS $telo$  BEGIN  IF UPPER(TG\_OP) = 'INSERT' THEN  -- vstavljanje v log tabelo  INSERT INTO log\_drzava  (idlogdrzava, datumspremembe, uporabnik, operacija, iddrzava)  VALUES(nextval('seq\_log\_drzava'), now(),  current\_user, TG\_OP, NEW.iddrzava);  END IF;  -- konec funkcije, vrnemo vrednost null  RETURN NULL;  END;  $telo$ LANGUAGE 'plpgsql';  -- dodatno Trigger  CREATE TRIGGER drzavaTG\_UD  AFTER DELETE OR UPDATE OR INSERT  ON drzava  FOR EACH ROW  EXECUTE PROCEDURE drzavaTG();  INSERT INTO public.drzava  (imedrzave, oznakadrzave, eu)  VALUES ( 'Iran', 'IRN', false); |

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.
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.

**CREATE OR REPLACE FUNCTION osebeD1 (i$Priimek VARCHAR, i$kraj VARCHAR)**

**RETURNS INTEGER AS**

**$telo$**

**DECLARE steviloIzbrisanihVrstic INTEGER;**

**BEGIN**

**-- izvedemo delete stavek**

**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;**

**EXCEPTION**

**WHEN others THEN**

**RAISE EXCEPTION 'NAPAKA...';**

**END;--prestrezanje izjem**

**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 tNaloga12 AS (**

**nazivK VARCHAR (30),**

**prodajanacena numeric (8,2),**

**kategorija VARCHAR (120)**

);

**CREATE OR REPLACE FUNCTION izdelkiStoritveS12(i$nazivK VARCHAR)**

**RETURNS SETOF tNaloga12 AS**

**$telo$**

**DECLARE**

**tVrstica tNaloga12; -- spremenljivka za vračanje rezultata**

**BEGIN**

**FOR tVrstica IN SELECT nazivK, prodajnacena, kategorija**

**FROM izdelkistoritve LEFT OUTER JOIN**

**kategorije USING (idKategorije)**

**WHERE**

**upper(nazivK) LIKE upper(i$nazivK) || '%'**

**LOOP**

**-- vračanje vrstice**

**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.

|  |
| --- |
| Entiteta “drzava” |
| CREATE TABLE log\_drzava  (  idserial integer not null primary key,  datumspremembe date,  uporabnik varchar,  operacija varchar,  iddrzava integer  );  CREATE SEQUENCE seq\_log\_drzava START 1;  CREATE OR REPLACE FUNCTION drzavaTG()  RETURNS TRIGGER AS $telo$  BEGIN  -- ? operacija  IF UPPER(TG\_OP) = 'UPDATE' THEN  -- vstavljanje v log tabelo  INSERT INTO log\_drzava  (idlogdrzava, datumspremembe, uporabnik, operacija, iddrzava)  VALUES(nextval('seq\_log\_drzava'), now(),  current\_user, TG\_OP, OLD.iddrzava);  END IF;  IF UPPER(TG\_OP) = 'DELETE' THEN  -- vstavljanje v log tabelo  INSERT INTO log\_drzava  (idlogdrzava, datumspremembe, uporabnik, operacija, iddrzava)  VALUES(nextval('seq\_log\_drzava'), now(),  current\_user, TG\_OP, OLD.iddrzava);  END IF;  IF UPPER(TG\_OP) = 'INSERT' THEN  -- vstavljanje v log tabelo  INSERT INTO log\_drzava  (idlogdrzava, datumspremembe, uporabnik, operacija, iddrzava)  VALUES(nextval('seq\_log\_drzava'), now(),  current\_user, TG\_OP, NEW.iddrzava);  END IF;  -- konec funkcije, vrnemo vrednost null  RETURN NULL;  END;  $telo$ LANGUAGE 'plpgsql';  CREATE TRIGGER drzavaTG\_UD  AFTER DELETE OR UPDATE OR INSERT  ON drzava  FOR EACH ROW  EXECUTE PROCEDURE drzavaTG();  INSERT INTO public.drzava  (imedrzave, oznakadrzave, eu)  VALUES ( 'Iran', 'IRN', false); |

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).
2. Napišite strežniško funkcijo, ki za izbrani načrt zbirke podatkov vrne zahtevan rezultat (statistična obdelava).

|  |
| --- |
| Entiteta “zaposleniIzobrazba” |
| 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' |

**PAZI ZOKI GLEDA ;)**