# Alternativni pristupi u izgradnji sistema baza podataka

Objektno-orijentisani pristupi u modelima podataka



1

## Sadržaj

- Motivacija
- Osnovni objektno-orijentisani koncepti
- Objektno-orijentisani model podataka
- Objektno-relacioni model podataka
- Poređenje objektnih modela podataka



2

# Motivacija

3

## Motivacija

- Problemi relacionih baza podataka
  - problem ograničenog broja tipova podataka
    - integer, date, string, itd.
    - pogodan za tradicionalne sisteme
      - koji obuhvataju dominantno alfanumeričke (tekstualno orijentisane) podatke
  - problem brzog rasta količine podataka
    - postaju kritične određene karakteristike SUBP-ova
      - kontrola konkurentnog izvršavanja
      - oporavak od grešaka
      - indeksiranje
      - performanse upita
      - izražajnost upitnog jezika

4

4

## Motivacija

- Zahtevi nove generacije aplikacija
  - česta upotreba **kompleksnih podataka**
    - tradicionalno smeštenih izvan baze podataka
      - datoteke operativnog sistema
      - specijalizovane strukture podataka
  - zahtev za proširenjem SUBP-ova konceptom **kompleksnog tipa**
  - novi domeni primene
    - CAM/CAD sistemi
    - geoinformacioni sistemi
    - multimedijalni sistemi

5

5

## Motivacija

- Rešenje
  - upotreba objektno-orijentisanih koncepata u bazama podataka
- Baze podataka sa OO konceptima podržavaju
  - laku integraciju sa OO programskim jezicima
  - **kompleksne tipove**
    - za smeštanje velikih podataka
      - slike, video, veliki tekstualni podaci itd.
  - **duge transakcije**
  - specifikaciju **strukture** i **operacija** nad podacima

6

6

## Motivacija

- Baze podataka sa OO konceptima
  - **Objektno-orijentisane baze podataka**
    - nastale pod uticajem objektno-orijentisanih programskih jezika
      - jezici prošireni funkcionalnostima SUBP-ova
    - **alternativa** relacionim bazama
    - kompleksni tipovi igraju glavnu ulogu
  - **Objektno-relacione baze podataka**
    - **proširenje** relacionih baza
    - predstavljaju objedinjenje relacione i objektne paradigme

7

7

## Osnovni objektno-orijentisani koncepti

8

## Osnovni objektno-orijentisani koncepti

- Pojam objektno-orijentisano (OO)
  - potiče od objektno-orijentisanih programskih jezika
    - kasne 1960. godine jezik SIMULA
    - 1970. godine Smaltalk
      - čisti objektno-orijentisani jezik
    - 1979. godine C++
      - hibridni objektno-orijentisani jezik
    - 1991. godine Python
      - višestruke paradigme
    - 1995. godine Java
    - 2000. godine C#

9

9

## Osnovni objektno-orijentisani koncepti

- Osnovna OO pravila
  - svaki entitet iz realnog sveta modeluje se **objektom**
    - svaki objekat poseduje jedinstveni identifikator
  - svaki objekat je instanca i okarakterisan je **atributima** i **metodama**
    - skup atributa je osnova **strukture objekta**
      - vrednost atributa može predstavljati objekat ili skup objekata
    - skup metoda definiše **ponašanje**
  - vrednosti atributa definišu **stanje** objekta
    - promena ili pristup stanju vrši se **slanjem poruka**
      - pozivanjem metoda nad objektom

10

10

## Osnovni objektno-orijentisani koncepti

- Osnovna OO pravila
  - objekti koji zadovoljavaju istu osobinu (predikat) grupišu se u jednu klasu
    - svaki objekat je instanca klase kojoj pripada
    - svi objekti klase zadovoljavaju istu strukturu i ponašanje, definisano klasom
  - klasa se može definisati **specijalizacijom** jedne ili više postojećih klasa
    - potklasa nasleđuje atribute i metode natklase

11

11

## Osnovni objektno-orijentisani koncepti

- Objekat
  - **privremeni objekat**
    - objekat koji postoji samo tokom izvršenja programa, u memorijskoj zoni programa
    - ograničenog životnog veka, najduže do završetka programa
  - **perzistentni objekat**
    - objekat koji je uskladišten u bazi podataka
    - ima neograničen životni vek
    - sa obezbeđenim pristupom od strane više programa

12

12

## Osnovni objektno-orijentisani koncepti

- Klasa
  - skup atributa (promenljivih, varijabli)
    - koncept atributa analogan je konceptu atributa u relacionom modelu
    - atribut može biti vidljiv izvan klase i skriven
  - skup operacija
    - operacija predstavlja specifikaciju ponašanja objekta
    - operacija se može izvršiti nad objektima date klase
  - objekat predstavlja skup vrednosti atributa klase
    - nad kojim se mogu primenjivati operacije klase
    - vrednostima skrivenih atributa ne može se pristupiti direktno
      - već korišćenjem namenski specificiranih operacija
      - princip enkapsulacije

13

13

## Osnovni objektno-orijentisani koncepti

- Glavni koncepti OO modela
  - identitet objekta
  - konstruktor tipa
  - enkapsulacija operacija
  - kompatibilnost između programskih jezika
  - hijerarhija tipova i nasleđivanje
  - ekstent
  - polimorfizam operacija

14

14

# Identitet objekta

15

## Identitet objekta

- Cilj OO modela podataka
  - da svaki realni entitet bude predstavljen jednim objektom u BP
    - za razliku od relacionog modela podataka, kod kojeg jedan realni entitet može biti dekomponovan u više torki relacija
    - efekat: olakšano očuvanje integriteta i identiteta objekta
- Jedinstvena identifikacija objekta
  - tipično identifikator objekta
    - *engl. object identificator* (OID)
    - nije vidljiv korisnicima
    - jedinstven za svaki objekat
    - koristi se za uspostavljanje veza između objekata

16

16

## Identitet objekta

- Identifikator objekta
  - **nepromenljiv**
    - vrednost identifikatora objekta uvek je stalna
      - od momenta kreiranja, do momenta uništenja objekta
    - ne zavisi od vrednosti ostalih atributa objekta
  - svaka vrednost OID-a iskorišćava se samo jednom
    - vrednosti OID-ova izbrisanih objekata ne koriste se nikada ponovo

17

17

## Identitet objekta

- Identifikator objekta
  - način formiranja vrednosti OID-a
    - upotrebom **fizičke adrese** objekta
      - problem nastaje prilikom promene adrese
        - pointer sa stare adrese na novu adresu
      - karakterističan za prve OO SUBP-ove
        - poboljšane performanse dobavljanja objekata
    - upotrebom **long integer** tipa
      - uobičajeno u savremenim OO SUBP-ovima
      - omogućeno je prevođenje vrednosti OID-a u fizičku adresu objekta
        - upotreba *hash map* preslikavanja

18

18

## Identitet objekta

- Literal (vrednost)
  - nad primitivnim (prostim) tipom podatka
    - može da egzistira kao poseban koncept
  - literal sa OID-om
    - literal kojem je pridružena vrednost OID-a
      - identične vrednosti literala mogu biti označene različitim vrednostima OID-a
    - karakteristično za prve OO modele
  - literal bez OID-a
    - nad primitivnim (prostim) tipom podatka, koji može da egzistira isključivo u okviru objekta
      - ne može se referencirati direktno, kao objekat
    - karakteristično za savremene OO modele

19

19

## Strukture kompleksnih tipova

20

## Strukture kompleksnih tipova

- Tip objekta i tip literala
  - definiše strukturu objekta, odnosno literala
    - skup atributa koji opisuje neki entitet iz realnog sveta
    - može da bude proizvoljne kompleksnosti
  - u relacionim sistemima, podaci o jednom entitetu nalaze se, često, u više torki različitih relacija
  - u objektnim sistemima dozvoljeno je definisanje kompleksnih tipova putem **konstruktora tipa**
    - **atomički konstruktor**
    - **konstruktor torke (strukture)**
    - **konstruktor kolekcije**
  - konstruktor tipa predstavlja pravilo za kreiranje novih tipova, upotrebom postojećih

21

21

## Strukture kompleksnih tipova

- **Konstruktor tipa**
  - **atomički konstruktor**
    - uvodi osnovne tipove u OO model podataka
      - atomički (jednovrednosni) tipovi
        - nisu razloživi na prostije tipove
      - primeri: integer, string, float, boolean, itd.
    - analogan koncept postoji u svakom modelu podataka
  - **konstruktor torke (strukture)**
    - omogućava kreiranje kompleksnih tipova
      - analogno kreiranju domena tipa torke u relacionom modelu
    - odgovara konceptu *struct* deklaracije u C-u

22

22

## Strukture kompleksnih tipova

- **Konstruktori tipa**
  - **konstruktor kolekcije**
    - konstruktor tipova koji predstavljaju kolekcije vrednosti
      - kolekcija objekata ili literala
        - može biti uređena ili neuređena
        - svi elementi su uvek istog tipa
      - *set(T)*, *list(T)*, *bag(T)*, *array(T)* i *dictionary(K,T)*

23

23

## Strukture kompleksnih tipova

- **Primer**

```
define type RADNIK
    tuple( ime:          string;
           prezime:      string;
           jmbg:         string;
           dat_rodj:     DATUM;
           adresa:       string;
           pol:          char;
           plata:        float;
           sef:          RADNIK;
           departman:    DEPARTMAN;
    );
```

24

24

## Strukture kompleksnih tipova

- **Primer**

```
define type DATUM
    tuple( godina:        integer;
           mesec:         integer;
           dan:           integer;
    );

define type DEPARTMAN
    tuple( naziv:         string;
           broj:          integer;
           rukovodilac:   tuple( rukovodilac:   RADNIK; dat_izbora:   DATUM; );
           lokacije:      set(string);
           zaposleni:     set(RADNIK);
    );
```

25

25

## Enkapsulacija i skladištenje objekata

26

## Enkapsulacija i skladištenje objekata

- Enkapsulacija
  - omogućava definisanje apstraktnih tipova i skrivanje informacija
  - tradicionalni (relacioni) sistemi ne enkapsuliraju podatke
    - moguć je pristup svim obeležjima relacije
    - sve definisane operacije nad podacima mogu se, u opštem slučaju, izvršiti nad podacima bilo koje relacije
      - select, insert, delete, update ili složene (korisnički definisane) operacije

27

27

## Enkapsulacija i skladištenje objekata

- Enkapsulacija
  - **enkapsulacija operacija**
    - operacija se specificira upotrebom dva koncepta
      - **potpis ili interfejs operacije**
        - specifikacija naziva, argumenata i tipa povratne vrednosti operacije
      - **telo operacije**
        - specifikacija kompletnog algoritma operacije
    - omogućena nezavisnost upotrebe operacije od načina njene implementacije
      - korisnik je svestan samo **interfejsa** operacije
      - korisnik nije svestan algoritma operacije
      - omogućene su takve izmene algoritma operacije koje ne utiču na potpis i, time, na način korišćenja operacije

28

28

## Enkapsulacija i skladištenje objekata

- Enkapsulacija
  - **potpuna enkapsulacija**
    - skrivanje svih atributa klase
      - neki OO modeli zahtevaju da svi atributi klase budu skriveni, a da se vrednostima atributa upravlja putem operacija date klase
  - mane potpune enkapsulacije
    - korisnik ne upotrebljava i ne vidi eksplicitno nazive atributa
      - da bi zadao operaciju nad objektima klase, korisnik upotrebljava odgovarajuću operaciju klase
        - tj. mora postojati operacija za svaki atribut klase
  - ovaj pristup se može relaksirati u praksi
    - atributi se dele na **vidljive** i **skrivene**
      - određene operacije zadaju se direktno nad vidljivim atributima
      - za skrivene atribute koriste se namenske operacije

29

29

## Enkapsulacija i skladištenje objekata

- Klasa
  - predstavlja specifikaciju tipa sa operacijama
  - klasa uključuje samo interfejse operacija
    - telo operacije implementira se izvan klase
      - u izabranom OO programskom jeziku
- Poziv operacija
  - kao i u OO programskim jezicima
    - `<objekat>.<naziv_operacija>(<lista_parametara>)`

30

30

## Enkapsulacija i skladištenje objekata

- Primer

```
define class RADNIK
    type tuple(     ime:            string;
                    prezime:        string;
                    jmbg:           string;
                    dat_rodj:       DATUM;
                    adresa:         string;
                    pol:            char;
                    plata:          float;
                    sef:            RADNIK;
                    departman:      DEPARTMAN;
    );
    operations      godina:         integer;
                    kreiraj_rad:    RADNIK;
                    obrisi_rad:     boolean;
end RADNIK;
```

31

31

## Enkapsulacija i skladištenje objekata

- Primer

```
define class DEPARTMAN
    type tuple(     naziv:          string;
                    broj:           integer;
                    rukovodilac:    tuple( rukovodilac:RADNIK; dat_izbora:DATUM; );
                    lokacije:       set(string);
                    zaposleni:      set(RADNIK);
    );
    operations      broj_zap:       integer;
                    kreiraj_dep:    DEPARTMAN;
                    obrisi_dep:     boolean;
                    unesi_zap(z: RADNIK): boolean;
                    obrisi_zap(z: RADNIK): boolean;
end DEPARTMAN;
```

32

32

## Enkapsulacija i skladištenje objekata

- Skladištenje objekata
  - **privremeni objekti**
    - postoje samo za vreme izvršavanja programa
  - **perzistentni (trajni) objekti**
    - skladište se u bazi podataka
    - tipični mehanizmi za deklarisanje perzistentnih objekata
      - **mehanizam imenovanja** (engl. *naming*)
      - **mehanizam proširenja dosega** (engl. *reachability*)

33

33

## Enkapsulacija i skladištenje objekata

- Prezistentni objekat
  - **mehanizam imenovanja**
    - dodeljuje se jedinstveno ime objektu
      - ime se koristi u izrazima i operacijama i predstavlja objekat kojem je dodeljeno
    - imenovani objekti su ulazne tačke (engl. *entry points*) za pristup BP
    - mehanizam može biti nepraktičan u slučaju BP s velikim brojem objekata
      - nepraktično je svim objektima dodeliti imena

34

34

## Enkapsulacija i skladištenje objekata

- Prezistentni objekat
  - **mehanizam proširenja dosega**
    - objekat postaje trajan ugrađivanjem u drugi trajni objekat
  - mehanizam proširenja može se primenjivati tranzitivno
    - objekat B je u dosegu objekta A ukoliko se iz objekta A može, bilo posredno (preko drugih objekata) ili neposredno, referencirati objekat B
      - kada postoje tzv. sekvence referenci

35

35

## Enkapsulacija i skladištenje objekata

- Primer

```
define class SKUP_DEPARTMANA
    type set(DEPARTMAN);
    operations     broj_zap:      integer;
                   kreiraj_skup:  SKUP_DEPARTMANA;
                   obrisi_skup:   boolean;
                   unesi_dep(d: DEPARTMAN):     boolean;
                   obrisi_dep(d: DEPARTMAN):    boolean;
end SKUP_DEPARTMANA;
```

36

36

## Enkapsulacija i skladištenje objekata

- Primer

```
persistent name DEPARTMANI: SKUP_DEPARTMANA;
(* DEPARTMANI su ime dodeljeno trajnom objektu tipa SKUP_DEPARTMANA *)
...
d := kreiraj_dep;
(* kreiraj novi objekat tipa DEPARTMAN u varijabli d *)
...
b := DEPARTMANI.dodaj_dep(d);
(* objekat d postaje trajan dodavanjem u trajni objekat DEPARTMANI *)
```

37

37

## Hijerarhija tipova i nasleđivanje

38

## Hijerarhija tipova i nasleđivanje

- Nasleđivanje
  - koncept koji omogućava kreiranje **hijerarhija klasa i tipova**
    - novi tipovi i klase sadrže strukturu i ponašanje prethodno definisanih tipova i klasa
  - dozvoljava ponovnu iskoristivost i inkrementalno razvijanje tipova i klasa

39

39

## Hijerarhija tipova i nasleđivanje

- Primer

```
GEOMETRIJSKA_FIGURA: Obim, Površina, Referentna_tačka

PRAVOUGAONIK subtype-of GEOMETRIJSKA_FIGURA: Visina, Sirina

TROUGAO subtype-of GEOMETRIJSKA_FIGURA: Stranica1, Stranica2, Ugao

KRUG subtype-of GEOMETRIJSKA_FIGURA: Poluprecnik
```

40

40

## Hijerarhija tipova i nasleđivanje

- Višestruko nasleđivanje
  - nasleđivanje kod kojeg jedna potklasa nasleđuje više natklasa
  - problem nasleđivanja predstavlja postojanje atributa i operacija s istim nazivom
    - moguća rešenja ovog problema:
      - sistemska provera kolizije
      - podrazumevana sistemska operacija ili atribut
      - ne dozvoliti višestruko nasleđivanje

41

41

## Hijerarhija tipova i nasleđivanje

- Selektivno nasleđivanje
  - nasleđuju se samo neke operacije i atributi iz natklase
    - EXCEPT klauzula
  - ne koristi se često u OO modelima

42

42

## Hijerarhija tipova i nasleđivanje

- Ekstent
  - imenovani perzistentni objekat
    - sadrži kolekciju objekata istog tipa
      - trajno smeštenih u bazu podataka
  - obuhvata, rekurzivno, i ekstentove svih podtipova
  - u nekim OO modelima postoji korenski ekstent
    - ROOT ili OBJECT klasa
    - sadrži sve ostale ekstentove

43

43

## Polimorfizam operacija

44

## Hijerarhija tipova i nasleđivanje

- Polimorfizam operacija
  - preklapanje operacija s istim nazivom, a u nečemu različitim potpisom
  - isti naziv vezuje se za dve ili više različitih implementacija operacija
  - od svih preklopljenih operacija, primenjuje se ona čiji potpis odgovara tipu objekta nad kojim se primenjuje

45

45

## Hijerarhija tipova i nasleđivanje

- Polimorfizam operacija
  - izbor odgovarajuće verzije operacije
    - rano (statičko) povezivanje
      - engl. *early binding*
      - u strogo tipiziranim sistemima
      - obavlja se za vreme kompajliranja
        - poznati su svi tipovi
    - kasno (dinamičko) povezivanje
      - engl. *late binding*
      - u netipiziranim ili slabo tipiziranim sistemima
      - prilikom izvršavanja proveravaju se tipovi objekata
        - odabira se ona implementacija koja odgovara tim tipovima

46

46

# Objektno-orijentisani model podataka

47

## Objektno-orijentisani model podataka

- Glavne prednosti relacionih SUBP
  - zasnovanost na relacionom modelu podataka i logičkoj nezavisnosti podataka
  - primena standardnog, deklarativnog jezika SQL
- OO SUBP-ovi
  - nedostatak deklarativnog i standardizovanog jezika podataka
    - sužava krug potencijalnih korisnika
- *Object Data Managemet Group* (ODMG)
  - konzorcijum, formiran 1991. godine
  - predložio standard ODMG-93 (ODMG 1.0)
    - postoje još dve novije verzije

48

48

## Objektno-orijentisani model podataka

- Sadržaj ODMG standarda
  - **objektni model**
  - **jezik za definisanje objekata**
    - *engl. object definition language* (ODL)
  - **objektni upitni jezik**
    - *engl. object query language* (OQL)
  - **način obezbeđenja veze sa OO programskim jezicima**
    - na koji način povezati OO BP sa programima

49

49

## Objektni model

50

## Objektni model

- ODMG objektni model
  - osnovni model za definisanje ODL i OQL
  - obuhvata
    - standardni model za objektne baze podataka
    - standardnu terminologiju
      - koja se, u određenim delovima, razlikuje od standardne OO terminologije

51

51

## Objektni model

- Objekat
  - poseduje identifikator i stanje (vrednost)
    - vrednost može imati kompleksnu strukturu
  - aspekti objekta
    - **identifikator objekta**
    - **naziv objekta**
    - **životni vek objekta**
    - **struktura objekta**
    - **kreiranje objekta**

52

52

## Objektni model

- Objekat
  - aspekti objekta
    - **identifikator objekta**
      - jedinstveno identifikuje svaki objekat
    - **naziv objekta**
      - nije obavezan
        - ukoliko postoji onda je jedinstven u sistemu
        - naziv obično imaju objekti koji predstavljaju kolekcije drugih objekata
      - omogućava postojanje ulazne tačke za pristup BP

53

53

## Objektni model

- Objekat
  - aspekti objekta
    - **životni vek objekta**
      - definiše da li je objekat perzistentan ili privremen
      - nezavisan od tipa objekta
    - **struktura objekta**
      - definiše kompleksnost objekta
        - obuhvata i odabir konstruktora tipa koji se koristi za kreiranje objekta
    - **kreiranje objekta**
      - obuhvata kreiranje objekta
        - operacija *new* interfejsa *Object_Factory*

54

54

## Objektni model

- Atomički objekat
  - korisnički definisan objekat koji ne predstavlja kolekciju drugih objekata
  - tip atomičkog objekta definiše se klasom
  - klasa atomičkog objekta sadrži specifikaciju
    - **atributa**
      - osobina (karakteristika) objekta kojoj se može dodeliti vrednost
    - **veza**
      - osobina koja definiše povezanost dve klase entiteta iz realnog sveta
      - u povezanoj klasi definiše se inverzna veza
    - **operacija**
      - koje definišu ponašanje objekata date klase

55

55

## Objektni model

- Objekat – fabrika
  - kreira druge objekte putem svojih operacija
  - nasleđuje interfejs *ObjectFactory*
- Objekat – baza podataka
  - predstavlja instancu jedne baze podataka
  - nasleđuje interfejs *Database*
  - poseduje ime baze podataka
  - poseduje operacije za
    - dodeljivanje jedinstvenog imena perzistentnim objektima
    - pretraživanje objekata po imenu
    - uklanjanje jedinstvenog imena perzistentnog objekta

56

56

## Objektni model

- Literal
  - ne poseduje identifikator
  - poseduje stanje (vrednost)
    - vrednost može imati kompleksnu strukturu
- Tipovi literala
  - **atomički literal**
  - **strukturirani literal**
  - **literal kolekcije**

57

57

## Objektni model

- Tipovi literala
  - **atomički literal**
    - tip je predefinisan u okviru modela podataka
  - **strukturirani literal**
    - literal dobijen kreiranjem strukture
      - koristi se rezervisana reč **STRUCT**
      - analogno konstruktoru torke
  - **literal kolekcije**
    - predstavlja kolekciju objekata
      - sama kolekcija ne poseduje identifikator
      - objekti u kolekciji poseduju sopstvene identifikatore

58

58

## Objektni model

- Nasleđivanje
  - nasleđivanje ponašanja
    - ponašanje se nasleđuje isključivo iz nadtipa
    - notacija obuhvata korišćenje simbola „ **:** "
    - nadtip mora biti interfejs
    - podtip može biti interfejs ili klasa
  - nasleđivanje ponašanja i strukture
    - nasleđuju se i ponašanje i struktura iz nadtipa
      - struktura obuhvata atribute i veze
    - notacija obuhvata korišćenje reči „**extends**"
    - nadtip i podtip moraju biti klase
    - **višestruko nasleđivanje nije dozvoljeno**

59

59

## Objektni model

- Ekstent
  - poseduje ime i sadrži sve perzistentne objekte date klase
  - ponaša se kao **imenovana kolekcija svih objekata** te klase
  - može se definisati samo u okviru klase
- Klasa sa ekstentom može imati jedan ili više ključeva
  - koji se sastoje od jednog ili više atributa
  - obezbeđuju svojim vrednostima jedinstvenu identifikaciju svakog objekta u ekstentu

60

60

## Objektni model

- Primer

```
class RADNIK
(   extent  SVI_RADNICI
    key     Jmbg
) {
    attribute       string          Ime;
    attribute       string          Jmbg;
    attribute       date            Dat_rodj;
    attribute       enum Pol{M, Z} P;
    attribute       short           Godina;
    relationship    DEPARTMAN       Radi_za
            inverse         DEPARTMAN::Ima_radnike;
    void raspodeli_radnika(in string novi_dep)
            raises(naziv_dep_nije_validan);
};
```

61

61

## Objektni model

- Primer

```
class DEPARTMAN
(   extent   SVI_DEPARTMANI
    key      Naziv, Broj
){
    attribute       string          Naziv;
    attribute       short           Broj;
    attribute       struct Dep_ruk {RADNIK Rukovodilac, date dat_izbora} Ruk;
    attribute       set<string>     Lokacije;
    attribute       struct Projekti {string Proj_naziv, time Broj_casova) Projekti;
    relationship    set<RADNIK> Ima_radnike
            inverse RADNIK::Radi_za;
    void dodaj_radnika(in string Ime_radnika) raises (ime_nije_validno);
    void promeni_ruk(in string Ime_novog_ruk; in date dat_izbora);
};
```

62

62

# Jezik za definisanje objekata

63

## Jezik za definisanje objekata

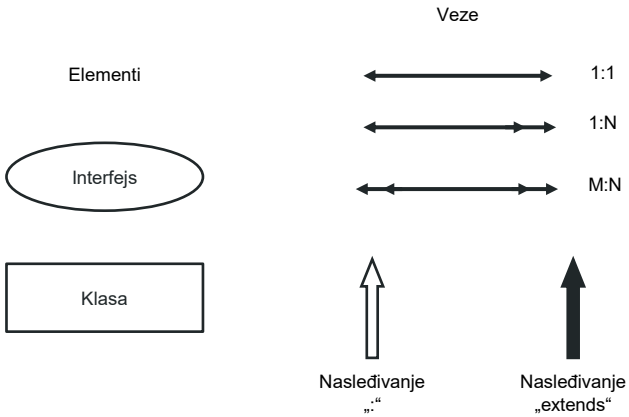
- Zasnovan na ODMG modelu
  - sadrži semantičke konstrukte koncepata modela
  - nezavisan od bilo kojeg programskog jezika
- Osnovni cilj je omogućavanje kreiranja specifikacije šeme baze podataka

64

64

## Jezik za definisanje objekata

- Primer

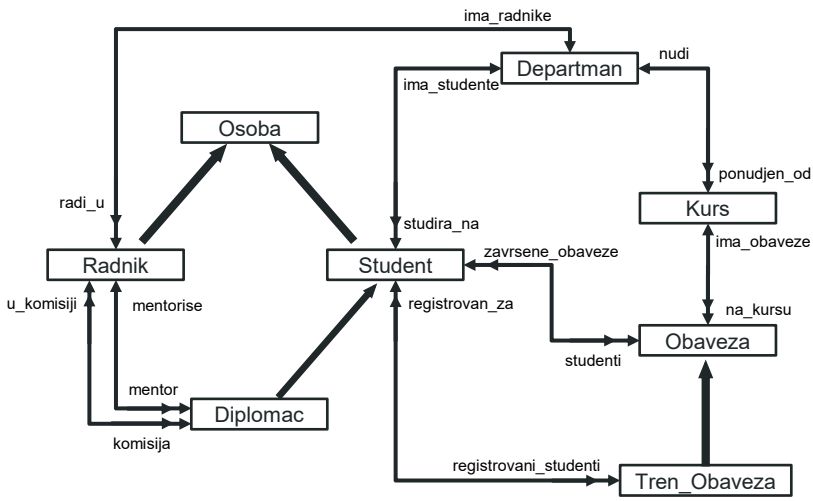



65

65

## Jezik za definisanje objekata

- Primer




66

66

## Objektni model

- Primer

```
class OSOBA
(    extent    OSOBE
     key       Jmbg
) {  attribute struct SIme{ string Ime,
                            string   Srednje_Ime,
                            string   Prezime} Puno_Ime;
     attribute string  Jmbg;
     attribute date    Dat_Rodj;
     attribute enum EPol{M, Z} Pol;
     attribute struct SAdresa { short Broj,
                            string   Ulica,
                            short    Br_Stan,
                            string   Grad,
                            string   Drzava,
                            short    Zip_Kod } Adresa;
     short    Godina();
};
```

67

67

## Objektni model

- Primer

```
class RADNIK extends OSOBA
(    extent  RADNICI     )
{
     attribute       string  Pozicija;
     attribute       float   Plata;
     attribute       string  Kancelarija;
     attribute       string  Telefon;

     relationship DEPARTMAN radi_u inverse DEPARTMAN::ima_radnike;
     relationship set<DIPLOMAC> mentorise inverse DIPLOMAC::mentor;
     relationship set<DIPLOMAC> u_komisiji inverse DIPLOMAC::komisija;

     void povisi_platu(in float povisica);
     void unapredi(in string nova_pozicija);
};
```

68

68

## Objektni model

- Primer

```
class OCENA
(   extent  OCENE )
{
    attribute       short   Ocena;
    relationship OBAVEZA obaveza inverse OBAVEZA::studenti;
    relationship STUDENT student inverse STUDENT::zavrsene_obaveze;
};

class DIPLOMA
{
    attribute       string  Fakultet;
    attribute       string  Diploma;
    attribute       string  Godina;
};
```

69

69

## Objektni model

- Primer

```
class STUDENT extends OSOBA
(   extent  STUDENTI )
{
    attribute       date    Dat_Upisa;
    relationship DEPARTMAN studira_na inverse DEPARTMAN::ima_Studente;
    relationship set<OCENA> zavrsene_obaveze inverse OCENA::student;
    relationship set<TREN_OBAVEZA> registrovan_za inverse
            TREN_OBAVEZA::registrovani_studenti;
    void promeni_departman(in string dep_naziv)
            raises(dep_naziv_nije_validan);
    float prosek();
    void registruj(in short obaveza_br) raises(obaveza_nije_validna);
    void dodeli_ocenu(in short obaveza_br; in short ocena)
            raises(obaveza_nije_validna, ocena_nije_validna);
};
```

70

70

## Objektni model

- Primer

```
class DIPLOMAC extends STUDENT
(   extent  DIPLOMCI )
{
    attribute set<DIPLOMA> Diplome;
    relationship RADNIK mentor inverse RADNIK::mentorise;
    relationship set<RADNIK> komisija inverse RADNIK::u_komisiji;
    void assign_advisor(in string Prezime; in string Ime)
            raises(radnik_nije_validan);
    void assign_committee_member(in string Prezime; in string Ime)
            raises(radnik_nije_validan);
};
```

71

71

## Objektni model

- Primer

```
class DEPARTMAN
(   extent  DEPARTMANI
    key     Dep_Naziv
){  attribute       string  Dep_Naziv;
    attribute       string  Dep_Telefon;
    attribute       string  Dep_Kancelarija;
    attribute       string  Dep_Fakultet;
    attribute       RADNIK  Dep_Rukovodilac;
    relationship set<RADNIK> ima_radnike inverse RADNIK::radi_u;
    relationship set<STUDENT> ima_studente inverse STUDENT::studira_na;
    relationship set<KURS> nudi inverse KURS::ponudjen_od;
};
```

72

72

## Objektni model

- Primer

```
class KURS
(    extent   KURSEVI
     key      Kurs_Br
){
     attribute          string    Kurs_Naziv;
     attribute          string    Kurs_Br;
     attribute          string    Kurs_Opis;
     relationship set<OBAVEZA> ima_obaveze inverse OBAVEZA::na_kursu;
     relationship DEPARTMAN ponudjen_od inverse DEPARTMAN::nudi;
};
```

73

73

## Objektni model

- Primer

```
class OBAVEZA
(    extent    OBAVEZE )
{    attribute         short    Obav_Br;
     attribute         string   Godina;
     attribute         enum ESemestar{Zimski, Letnji} Semestar;
     relationship set<OCENA> studenti inverse OCENA::obaveza;
     relationship KURS na_kursu inverse KURS::ima_obaveze;
};
class TREN_OBAVEZA extends OBAVEZA
(    extent    TREN_OBAVEZA )
{    relationship set<STUDENT> registrovani_studenti
              inverse STUDENT::registrovan_u
     void registruj_studenta(in string Jmbg)
              raises(student_nije_validan, obaveza_puna);
};
```

74

74

# Objektni upitni jezik

75

## Objektni upitni jezik

- Objektni upitni jezik (OQL)
  - predložen za specifikaciju upita u ODMG modelu
  - predložen za upotrebu u programskim jezicima prihvaćenim u ODMG modelu
    - svaki upit vraća objekte
      - tip objekta mora biti podržan u pogramskom jeziku
    - metode su implementirane u tim jezicima
  - sintaksa slična SQL-u, sa dodacima
    - identiteta
    - kompleksnih objekata
    - operacija
    - nasleđivanja
    - polimorfizma
    - veza između objekata

76

76

## Objektni upitni jezik

- Mora postojati ulazna tačka pristupa BP za svaki upit
  - u većini slučajeva ulaznu tačku predstavlja ime ekstenta neke klase
  - u opštem slučaju, ulaznu tačku predstavlja bilo koji **imenovani perzistentni objekat**
    - ne mora biti kolekcija

```
select       D.Dep_Naziv
from   D in DEPARTMANI
where  D.Dep_Fakultet = 'FTN';
```

77

77

## Objektni upitni jezik

- Specifikacija iteratora
  - `D in DEPARTMANI`
  - `DEPARTMANI D`
  - `DEPARTMANI as D`
- Rezultat prethodnog upita
  - `bag<string>`
  - `set<string>`
    - ukoliko se koristi **distinct**
  - `list<string>`
    - ukoliko koristimo **order by**

78

78

## Objektni upitni jezik

- Rezultat upita
  - u opštem slučaju, može da bude bilo koji objekat ODMG modela podataka
  - upit ne mora da prati *select...from...where...* strukturu
    - bilo koje ime perzistentnog objekta može se posmatrati kao upit
    - primeri
      - DEPARTMANI;
        - vraća sve departmane iz baze podataka
      - RA_DEPARTMAN;
        - vraća jedan departman koji ima naziv RA_DEPARTMAN

79

79

## Objektni upitni jezik

- Izraz putanje
  - opisuje putanju do povezanih atributa i objekata
    - definiše se nakon definisanja ulazne tačke
  - počinje imenom perzistentnog objekta ili iteratora
  - navodi se nula ili više naziva veza ili atributa razdvojenih „ . “
  - primeri

```
RA_DEPARTMAN.Dep_Rukovodilac;
RA_DEPARTMAN.Dep_Rukovodilac.Pozicija;
RA_DEPARTMAN.ima_radnike;
```

80

80

## Objektni upitni jezik

- Izraz putanje
  - primeri

```
select  F.Pozicija
from    F in RA_DEPARTMAN.ima_radnike;

select  distinct F.Pozicija
from    F in RA_DEPARTMAN.ima_radnike;
```

81

81

## Objektni upitni jezik

- Izraz putanje
  - primer
    - upiti koji vraćaju kompleksnu strukturu
    - *Pronaći sve studente čiji mentor je rukovodilac departmana za računarstvo i automatiku.*

```
RA_DEPARTMAN.Rukovodilac.Mentorise;
```

82

82

# Objektni upitni jezik

- Izraz putanje
  - primer
    - upiti koji vraćaju kompleksnu strukturu
    - *Pronaći ime i prezime kao i sve diplome studenata kojima je mentor rukovodilac departmana za računarstvo i automatiku.*

```
select struct ( ime: struct (prezime: S.Puno_Ime.Prezime,
        ime:S.Puno_Ime.Ime),
                diplome:( select struct (diploma: D.Diploma,
                        godina: D.Godina, fakultet: D.Fakultet)
                        from D in S.Diplome ))
from S in RA_DEPARTMAN.Rukovodilac.Mentorise;
```

83



# Objektni upitni jezik

- Izraz putanje
  - OQL je **ortogonalan** u odnosu na izraze putanje
    - ne zavisi od koncepata korišćenih prilikom specifikacije izraza putanje
      - u izrazima putanje mogu se koristiti
        - atributi
        - veze
        - operacije
      - sve dok struktura OQL upita nije narušena

84

## Objektni upitni jezik

- Izraz putanje
  - primer
    - *Pronaći prosek svih studenata upisanih 2012. godine na departman za računarstvo i automatiku.*

```
select struct ( prezime: S.Puno_Ime.Prezime, ime: S. Puno_Ime.Ime,
          prosek: S.prosek )
from S in RA_DEPARTMAN.ima_studente
where S.Dat_Upisa = ’2012’
order by prosek desc, prezime asc, ime asc;
```

85

85

## Objektni upitni jezik

- Izraz putanje
  - primer
    - *Pronaći prosek svih studenata upisanih 2012. godine na departman za računarstvo i automatiku.*

```
select struct ( prezime: S.Puno_Ime.Prezime, ime: S. Puno_Ime.Ime,
          prosek: S.prosek )
from S in STUDENTI
where S.studira_na.Dep_Naziv = ‘Racunarstvo i automatika’ and
          S.Dat_Upisa = ’2012’
order by prosek desc, prezime asc, ime asc;
```

86

86

## Objektni upitni jezik

- Pogled
  - imenovani upit
    - ime pogleda mora biti jedinstveno u sistemu
      - uzimajući u obzir sve prethodno imenovane koncepte, kao npr.
        - imenovani objekti, klase, metode, funkcije u šemi, itd.
      - ukoliko već postoji drugi koncept sa istim imenom, njegovo ime više neće biti vidljivo
    - može da sadrži parametre u definiciji
  - rezervisana reč **DEFINE**

87

87

## Objektni upitni jezik

- Pogled
  - primer

```
define Studenti_Departmana(D_Naziv) as
select S
from S in STUDENTI
where S.studira_na.Dep_Naziv = D_Naziv;
```

  - korišćenje pogleda:

```
Studenti_Departmana('Računarstvo i automatika');
```

88

88

## Objektni upitni jezik

- Ekstrakcija elementa iz singleton kolekcije
  - singleton kolekcija
    - kolekcija koja sadrži samo jedan element
  - ekstrakcija elementa
    - rezervisana reč **ELEMENT**
      - **garantuje** vraćanje samo jednog elementa
    - ukoliko kolekcija ima više od jednog, ili nema ni jedan element
      - izaziva se izuzetak

89

89

## Objektni upitni jezik

- Ekstrakcija elementa iz singleton kolekcije
  - primer

```
element ( select D
          from   D in DEPARTMANI
          where  D.Dep_Naziv = 'Racunarstvo i automatika' );
```

90

90

## Objektni upitni jezik

- Operacije nad kolekcijom
  - agregatori
    - count, min, max, sum, avg, itd.
    - primenjuju se samo nad tipovima za koje su definisani modelom podataka
  - operacije sadržavanja i kvantifikatori
    - vraćaju boolean vrednost

91

91

## Objektni upitni jezik

- Agregatori
  - primeri

```
count ( S in Studenti_Departmana('Racunarstvo i automatika'));

avg ( select S.prosek
    from S in STUDENTI
    where S.studira_na.Dep_Naziv = 'Racunarstvo i automatika'
           and S.Dat_Upisa = '2012' );
```

92

92

## Objektni upitni jezik

- Operacije sadržavanja i kvantifikatori
  - mogući oblici
    - `E in C`
      - vraća *true* ukoliko kolekcija C sadrži element E
    - `for all V in C : B`
      - vraća *true* ukoliko **svi** elementi kolekcije C zadovoljavaju logički uslov B
    - `exists V in C : B`
      - vraća *true* ukoliko postoji bar jedan element u kolekciji C koji zadovoljava logički uslov B

93

93

## Objektni upitni jezik

- Operatori sadržavanja i kvantifikatori
  - primer
    - *Da li Vladimir studira na departmanu za računarstvo i automatiku?*
      - pretpostavka da je VLADIMIR imenovani objekat tipa STUDENT

```
VLADIMIR in Studenti_Departmana('Racunarstvo i automatika');
```

94

94

## Objektni upitni jezik

- Operatori sadržavanja i kvantifikatori
  - primer
    - *Da li je svim studentima na departmanu za računarstvo i automatiku mentor radnik sa tog departmana?*

```
for all G in
    (       select S
            from S in DIPLOMCI
            where S.studira_na.Dep_Naziv = 'Racunarstvo i automatika')
    : G.Mentorise in RA_DEPARTMAN.ima_radnike;
```

95

95

## Objektni upitni jezik

- Operatori sadržavanja i kvantifikatori
  - primer
    - *Da li na departmanu za računarstvo i automatiku postoji diplomac koji ima prosek 10?*

```
exists G in
    ( select S
    from S in DIPLOMCI
    where S.studira_na.Dep_Naziv = 'Racunarstvo i automatika')
: G.prosek = 10;
```

96

96

## Objektni upitni jezik

- Izrazi nad uređenim kolekcijama
  - operacije nad listama i nizovima
    - operacija koja vraća element sa neke pozicije
    - operacija koja vraća potkolekciju
    - operacija koja spaja dve kolekcije

97



## Objektni upitni jezik

- Izrazi nad uređenim kolekcijama
  - primer
    - *Pronaći radnika sa najvećom platom.*

```
first ( select struct(ime: F.Puno_Ime.Prezime, plata: F.Plata)
   from F in RADNIK
   order by plata desc );
```

98

## Objektni upitni jezik

- Izrazi nad uređenim kolekcijama
  - primer
    - *Pronaći tri najbolja studenta na departmanu za računarstvo i automatiku.*

```
( select struct( prezime: S.Puno_Ime.Prezime,
                 ime: S.Puno_Ime.Ime,
                 prosek: S.prosek )
from S in RA_DEPARTMAN.ima_studente
order by prosek desc ) [0:2];
```

99



## Objektni upitni jezik

- Izrazi grupisanja
  - koristi se klauzula **group by**
    - slično kao u SQL-u
    - definiše eksplicitnu referencu na svaku kolekciju koja sačinjava grupu (particiju)
      - rezervisana reč **PARTITION**
    - **group by** *F*1: *E*1, *F*2: *E*2, ..., *Fk*: *Ek*
      - lista grupišućih atributa
    - osim group by klauzule moguće je definisati i **having** klauzulu
  - rezultat primene grupisanja
    - **set** <**struct**(*F*1: *T*1, *F*2: *T*2, ..., *Fk*: *Tk*, **partition**: **bag**<*B*>)>

100

## Objektni upitni jezik

- Izrazi grupisanja
  - primer
    - *Pronaći broj sudenata na svakom departmanu.*

```
select struct(dep_naziv, broj_studenata: count (partition) )
from S in STUDENTI
group by dep_naziv : S.studira_na.Dep_Naziv;
```

101

101

## Objektni upitni jezik

- Izrazi grupisanja
  - primer
    - *Pronaći ukupan prosek na svakom departmanu koji ima više od 100 studenata.*

```
select dep_naziv, avg_prosek: avg (
          select P.prosek from P in partition
          )
from S in STUDENTI
group by dep_naziv : S.studira_na.Dep_Naziv
having count (partition) > 100;
```

102

102

# Objektno-orijentisani SUBP-ovi

103

## Objektno-orijentisani SUBP-ovi

- GemStone
  - 1982. godina
    - jedan od prvih OO SUBP-ova na tržištu
  - model podataka zasnovan na OO jeziku Smaltalk
    - zamišljen kao proširenje ovog jezika
  - **Opal**
    - upitni jezik
    - jezik za manipulaciju podacima
  - SUBP zasnovan na K/S arhitekturi
  - podržane platforme
    - DEC, Sun i IBM RS6000

104

104

## Objektno-orijentisani SUBP-ovi

- GemStone
  - aplikacije mogu biti razvijene u programskim jezicima Smaltalk, C, C++ i Pascal
  - sadrži namenske alate za
    - generisanje interfejsa
    - pretragu šeme baze podataka

105

105

## Objektno-orijentisani SUBP-ovi

- $O_2$
  - rane 1990.-te godine
  - model podataka napravljen „od nule"
    - nije zasnovan ni na jednom OO jeziku
  - $CO_2$
    - jezik za implementaciju metoda
    - zasnovan na C-u
  - SUBP zasnovan na K/S arhitekturi
  - podržana platforma
    - Unix

106

106

## Objektno-orijentisani SUBP-ovi

- $O_2$
  - sadrži
    - korisnički interfejs za
      - vizuelno kreiranje šeme baze podataka
      - pretragu šeme baze podataka
    - okruženje za pisanje OO koda

107



## Objektno-orijentisani SUBP-ovi

- ObjectStore
  - 1991. godina
  - model podataka zasnovan na programskom jeziku C++
  - upravljanje podacima obavlja se pomoću konstrukata napisanih u C++-u
    - proširenje C++-a za upravljanje podacima
      - posebni konstrukti

108

## Objektno-orijentisani SUBP-ovi

- db4o
  - 2000. godina
    - v8 izdata 2011. godine
  - kao model podataka koristi model klasa aplikacije
  - može se koristiti kao ugrađena baza podataka
  - SUBP zasnovan na K/S arhitekturi
  - namenjena za korišćenje u Javi i .Net-u

109

109

## Objektno-orijentisani SUBP-ovi

- VelocityDB
  - 2012. godina
  - platforma
    - .Net
  - nije isključivo objektna baza podataka
    - NoSql
    - Graph Data Store
    - ugrađena baza podataka

110

110

# Objektno-relacioni model podataka

111

## Objektno-relacioni model podataka

- Objektno-relaciona baza podataka
  - proširenje relacione BP objektnim konceptima
- Standard SQL:1999 (SQL3)
  - proširen konceptima OO modela podataka

112

112

## Objektno-relacioni model podataka

- Objektno-relaciona baza podataka
  - objektna proširenja SQL-a
    - neki **konstruktori tipa**
      - specifikacija kompleksnih tipova
      - **konstruktor tipa reda**
        - odgovara konstruktoru torke (strukture)
      - **konstruktor tipa niza**
        - odgovara konstruktoru kolekcije
        - ostali tipovi kolekcija su naknadno dodati
          - *array, multiset, list* i *set*

113

113

## Objektno-relacioni model podataka

- Objektno-relaciona baza podataka
  - objektna proširenja SQL-a
    - mehanizam za specifikaciju **identifikatora objekta**
    - **enkapsulacija operacija**
      - **korisnički definisani tipovi**
        - sadrže operacije kao deo deklaracije
        - slični apstraktnim tipovima
    - **nasleđivanje**
      - rezervisana reč **UNDER**

114

114

## Objektno-relacioni model podataka

- Korisnički definisani tipovi (KDT)
  - omogućavaju kreiranje objekata sa kompleksnom strukturom
    - izvan definicije tabele
  - kreirani tip je deo šeme baze podataka
  - KDT se koristi kao **tip atributa** ili **tip tabele**
    - kreiranjem atributa sa KDT unutar definicije drugog KDT-a
      - kreira se kompleksna struktura

```
CREATE TYPE <NAZIV_TIPA> AS (
    <LISTA ATRIBUTA I NJIHOVIH TIPOVA>
    <DEKLARACIJE FUNKCIJA (METODA)>
);
```

115

115

## Objektno-relacioni model podataka

- Primer

```
CREATE TYPE ADRESA_ULICE_TIP AS (
    BROJ_OBJEKTA     VARCHAR (5),
    IME_ULICE        VARCHAR (25),
    BROJ_STANA       VARCHAR (5)
);
CREATE TYPE ADRESA_TIP AS (
    ADRESA_ULICE     ADRESA_ULICE_TIP,
    GRAD             VARCHAR (25),
    ZIP              VARCHAR (10)
);
CREATE TYPE TELEFON_TIP AS (
    VRSTA_TELEFONA   VARCHAR (5),
    POZIVNI_BROJ     CHAR (3),
    BROJ_TELEFONA    CHAR (7)
);
```

116

116

## Objektno-relacioni model podataka

- Primer

```
CREATE TYPE OSOBA_TIP AS (
    IME             VARCHAR (35),
    POL             CHAR,
    DATUM_RODJENJA  DATE,
    TELEFONI        TELEFON_TIP ARRAY [4],
    ADRESA          ADRESA_TIP
INSTANTIABLE
NOT FINAL
REF IS SYSTEM GENERATED
INSTANCE METHOD STAROST() RETURNS INTEGER;
CREATE INSTANCE METHOD STAROST() RETURNS INTEGER
    FOR OSOBA_TIP
    BEGIN
            RETURN /* IZRAČUNATU STAROST OSOBE */
    END;
);
```

117

117

## Objektno-relacioni model podataka

- Primer

```
CREATE TYPE RADNIK_TIP UNDER OSOBA_TIP AS (
    KOD_POSLA       CHAR (4),
    PLATA           FLOAT,
    JMBG            CHAR (11)
INSTANTIABLE
NOT FINAL    );

CREATE TYPE RUKOVODILAC_TIP UNDER RADNIK_TIP AS (
    DEPARTMAN       CHAR (20)
INSTANTIABLE );

CREATE TYPE OCENA_TIP AS (
    KURS            CHAR (8),
    SEMESTAR        VARCHAR (8),
    GODINA          CHAR (4),
    OCENA           INTEGER          );
```

118

118

## Objektno-relacioni model podataka

- Primer

```
CREATE TYPE STUDENT_TIP UNDER OSOBA_TIP AS (
    OZNAKA_DEPARTMANA       CHAR (4),
    BROJ_INDEKSA            CHAR (12),
    DIPLOMA                 VARCHAR (5),
    OCENE                   OCENA_TIP ARRAY [100]
INSTANTIABLE
NOT FINAL
INSTANCE METHOD PROSEK() RETURNS FLOAT;
CREATE INSTANCE METHOD PROSEK() RETURNS FLOAT
    FOR STUDENT_TIP
    BEGIN
            RETURN /* IZRAČUNATI PROSEK*/
    END;
);
```

119

119

## Objektno-relacioni model podataka

- Primer

```
CREATE TABLE OSOBA OF OSOBA_TIP
    REF IS OSOBA_ID SYSTEM GENERATED;
CREATE TABLE RADNIK OF RADNIK_TIP
    UNDER OSOBA;
CREATE TABLE RUKOVODILAC OF RUKOVODILAC_TIP
    UNDER RADNIK;
CREATE TABLE STUDENT OF STUDENT_TIP
    UNDER OSOBA;
```

120

120

## Objektno-relacioni model podataka

- Primer

```
CREATE TYPE KOMPANIJA_TIP AS (
    NAZIV_KOMPANIJE  VARCHAR (20),
    LOKACIJA         VARCHAR (20)    );

CREATE TYPE ZAPOSLENJE_TIP AS (
    Radnik           REF (RADNIK_TIP) SCOPE (RADNIK),
    Kompanija        REF (KOMPANIJA_TIP) SCOPE (KOMPANIJA)    );

CREATE TABLE KOMPANIJA OF KOMPANIJA_TIP (
    REF IS KOMPANIJA_ID SYSTEM GENERATED,
    PRIMARY KEY (NAZIV_KOMPANIJE)    );

CREATE TABLE ZAPOSLENJE OF ZAPOSLENJE_TIP;
```

121

121

## Objektno-relacioni model podataka

- Primer
  - **koncept tipa reda**

```
CREATE TYPE ADRESA_TIP AS (
    ADRESA_ULICE   ROW (  BROJ_OBJEKTA   VARCHAR (5),
                          IME_ULICE      VARCHAR (25),
                          BROJ_STANA     VARCHAR (5)
                   ),
    GRAD           VARCHAR (25),
    ZIP            VARCHAR (10)   );
```

123

123

## Objektno-relacioni model podataka

- Identifikator objekta
  - načini generisanja vrednosti identifikatora
    - **sistemski generisan**
      - `REF IS SYSTEM GENERATED`
      - svakom kreiranom objektu pridružuje se jedinstveni sistemsko-generisani identifikator
        - moguće je i koristiti tradicionalni ključ umesto OID-a
    - **korisničkom metodom**
      - `REF IS <OID_ATRIBUT> <VRSTA_GENERISANJA>;`
      - OID_ATRIBUT će sadržati vrednost identifikatora
      - VRSTA_GENERISANJA
        - **sistemski generisan**
        - **izveden**
          - iz vrednosti atributa
          - tradicionalni ključ

124

124

## Objektno-relacioni model podataka

- Kreiranje tabela
  - za svaki KDT definiše se da li može biti tip tabele
    - rezervisana reč **INSTANTIABLE**
      - ukoliko nije navedena KDT može biti samo tip atributa

125

125

## Objektno-relacioni model podataka

- Enkapsulacija operacija
  - u SQL-u korisnički definisani tipovi mogu imati definisane metode
  - metode se mogu implementirati
    - u posebnoj datoteci
    - u samoj specifikaciji tipa
  - opšti oblik

```
INSTANCE METHOD <NAZIV> (<LISTA_PARAMETARA>)
        RETURNS <POVRATNA_VREDNOST>;
```

126

126

## Objektno-relacioni model podataka

- Enkapsulacija operacija
  - svaki tip poseduje ugrađene funkcije
    - **konstruktor**
      - vraća novi objekat datog tipa
      - atributi su inicijalizovani na podrazumevane vrednosti
    - **funkcija povratka vrednosti**
      - za svaki atribut vraća njegovu vrednost
    - **funkcija postavljanja vrednosti**
      - za svaki atribut postavlja njegovu vrednost
  - potrebna je privilegija **EXECUTE** nad bazom podataka da bi se pristupilo ovim funkcijama

127

127

## Objektno-relacioni model podataka

- Enkapsulacija operacija
  - podela funkcija prema mestu implementacije
    - **interne SQL funkcije**
      - implementirane u SQL/PSM
    - **eksterne funkcije**
      - implementirane u jeziku aplikacije
      - opšti oblik

```
DECLARE EXTERNAL <NAZIV_FUNKCIJE> <POTPIS>
    LANGUAGE <NAZIV_PROGRAMSKOG_JEZIKA>;
```

128

128

## Objektno-relacioni model podataka

- Nasleđivanje
  - dve vrste nasleđivanja u OR SUBP-ovima
    - **nasleđivanje tipova**
    - **nasleđivanje tabela**

129

129

## Objektno-relacioni model podataka

- Nasleđivanje
  - **nasleđivanje tipova**
    - rezervisana reč **UNDER**
    - nasleđuju se atributi i operacije
    - tip se može naslediti samo ukoliko je definisana opcija **NOT FINAL**
    - pravila nasleđivanja tipova
      - svi atributi se nasleđuju
      - redosled nadtipova u UNDER klauzuli definiše hijerarhiju
      - podtip može redefinisati bilo koju funkciju nadtipa
        - potpis mora biti identičan
      - poziva se ona funkcija čiji tipovi argumenata najbolje odgovaraju prosleđenim parametrima
      - kod dinamičkog povezivanja, uzimaju se u obzir tipovi parametara

130

130

## Objektno-relacioni model podataka

- Nasleđivanje
  - **nasleđivanje tabela**
    - rezervisana reč **UNDER**
    - svaki novi zapis u podtabeli se takođe upisuje u sve nadtabele
      - operacija se propagira do vrha hijerarhije

131

131

## Objektno-relacioni model podataka

- Reference
  - atribut neke torke može biti referenca ka torki druge relacije
    - rezervisana reč **REF**
    - ime tabele čije se torke mogu referencirati
      - rezervisana reč **SCOPE**
  - **izrazi putanje**
    - nad ROW atributima
      - pomoću karaktera „ . “
    - nad REF atributima
      - pomoću karaktera „ -> “
      - dereferenciranje se obavlja kao u programskom jeziku C

132

132

## Objektno-relacioni model podataka

- Primer
  - *Pronaći sve radnike koji rade u kompaniji FTN.*

```
SELECT     Z.Radnik->IME
FROM       ZAPOSLENJE AS Z
WHERE      Z.Kompanija->NAZIV_KOMPANIJE = 'FTN';
```

133

133

# Poređenje objektnih modela podataka

134

## Poređenje objektnih modela podataka

- Poređenje objektnih modela
  - može se posmatrati kroz poređenje SUBP-ova
    - koji podržavaju koncepte predviđene modelima
    - RSUBP
      - podržava samo relacione koncepte
    - ORSUBP
      - podržava relacione i određen broj objektnih koncepata
    - OOSUBP
      - podržava samo objektne koncepte

135

135

## Poređenje objektnih modela podataka

- RSUBP - ORSUBP
  - RSUBP ne sadrži objektna proširenja koja sadrži ORSUBP
  - RSUBP koristi manje kompleksne tipove podataka
    - lakša optimizacija upita
    - lakše korišćenje zbog manjeg broja opcija
  - ORSUBP je prilagodljiviji za različite domene primene

136

136

## Poređenje objektnih modela podataka

- ORSUBP – OOSUBP: sličnosti
  - OO koncepti
    - korisnički definisani tipovi
    - struktuirani tipovi
    - identifikator objekta
    - nasleđivanje
  - poseduju upitne jezike za rad sa tipovima kolekcija
    - prošireni SQL
    - ODL/OQL
  - proširuje se skup funkcionalnosti SUBP
    - upravljanje konkurentnim izvršavanjem
    - oporavak od grešaka

137

137

## Poređenje objektnih modela podataka

- ORSUBP – OOSUBP: razlike
  - OOSUBP nameću proširenja OO programskih jezika naredbama za rad s bazama podataka
    - jedinica obrade podataka predstavlja objekat
      - jednom učitani objekti koriste se duži vremenski period
        - duge transakcije
        - ponekad se dobavljaju povezani objekti
      - objekti mogu biti veliki
        - dobavljaju se u delovima

138

138

## Poređenje objektnih modela podataka

- ORSUBP – OOSUBP: razlike
  - ORSUBP predstavljaju proširenja RSUBP novim tipovima podataka
    - jedinica obrade podataka predstavlja torka ili kompleksniji objekat
      - veliko iskorišćenje diska
        - optimizacija
      - relativno kratke transakcije

139

139

## Reference


- Elmasri R, Navathe S B, „*Fundamentals of Database Systems*“, Šesto izdranje, Addison-Wesley, SAD, 2011
  - poglavlje 11
- Ramakrishnan R, Gehrke J, „*Database Management Systems*“, Treće izdanje, McGraw Hill, SAD, 2003
  - poglavlje 23
- Bertino E, Catania B, Zarri G P, „*Inteligent Database Systems*“, Prvo izdranje, Addison-Wesley, SAD, 2001
  - poglavlje 2




140