SIIT

BAZE PODATAKA

4. SEMESTAR SIIT

USMENI KOD SLAVICE KORDIC

1) Osnovni pojmovi
- Odnos realnog sveta i IS
- Zasto gradimo IS
- Od cega se sastoji IS (racunar, BP, aplikacije, korisnici, dokumentacija, maintenance)
- Tipovi struktura podataka (TE TP TPE TPP Obelezje)
  i o svakoj ponesto
- Tipovi obelezja (elementarno, slozeno, skupovno)
- Podatak i kontekst podatka (ime, obelezje, vreme, vrednost)
- Identifikatori (int, ekst) i kljucevi
- Strukture podataka (LSO, LSP, FSP)
  Nacini predstave (graf, tabela)

2) Koncepcija baze podataka
- Klasicna (stara) organizacija podataka
  Sta i kako, i zasto ne valja
- Ideja modernih baza podataka (sve na jednom mestu)
- SUBP, zasto se koristi
- DDL DML QL
- Sema i podsema, sta kako i zasto
- Sistem baze podataka (BP, SUBP, DDLDMLQL, Sema/podsema)

3) Modeli podataka
- Cemu sluzi model podataka (LSO, apstrakcija)
- Od cega se sastoji? (S I O)
- Intenzija i ekstenzija
- Strukturalna komponenta
  Primitivni, pravila, slozeni
- Integritetna komponenta
  Primeri, tipa ogranicenja kljuca, domena, kardinaliteta
  Validacija podataka gde se radi i zasto
- Operacijska komponenta
  - Sta ima svaka operacija (osobine (formalna spec, izvrsenje), pravila, semantika)
  - DDL DML QL (cemu sluze)
  - Specifikacija operacije (aktivnost, selekcija)
  - Tipovi operacije (proceduralna, specifikaciona)

- Razliciti tipovi modela podataka
  ER, mrezni, hijer., relacioni, OOP, XML, ...

4) ER model
- Sta je i cemu sluzi (staticko graficko modeliranje)
- Od cega se sastoji (entiteti poveznici)
- Strukturalna komponenta
  - Vrednost, domen, obelezje
  - Tipovi domena (predefinisani, korisnicki definisani)
  - Obelezje (sta je i sta ga cini)
  - Podatak (entitet, obelezje, vrednost, vreme)
  - TE TP TPE TPP
- Integritetna komponenta
  - Ogr. domena, pojave tipa, kardinaliteta i kljuca
    O svakom ponesto
- Gerund
  - Sta znaci tj. sta predstavlja i kada se koristi
- Agregacija
  - Sta je i kada se koristi
- Tipovi entiteta (slabi, jaki)
- Tipovi zavisnosti (egzistencijalna, ID) i primeri
- IS-A hijerarhija
  - Cemu sluzi i kako funkcionise
- Kategorizacija
  - Cemu sluzi, kada i kako se koristi
- Nivoi detaljnosti na ER dijagramu (globalni, detaljni)

5) Relacioni model - Koncepcija

- Sta je bio problem inicijalnih baza podataka
- Koji su motivi uvodjenja relacionog modela, i kako je razvijen (matematicke apstrakcije, opsta primena itd)
- Relacija (sta je i kako se prikazuje (tabela ugl))
- Selekcija podataka (kako pre, kako u relacionom (adresa nekad, sad kljuc)
- Povezivanje podataka (kako pre, kako u relacionom (isto adresa - kljuc)
- Deklarativni jezik
    Relaciona algebra i racun (algebra unija/presek/razlika, racun nad n-torkama sa ==/!/where in i slicno)
- SQL, deklarativni jezik, sta to znaci, koje su neke ceste komande (select, from, where)
- 12 pravila RMP (mozda nebitno)

7) Eksterna memorijska fizicka organizacija
- Sta je datoteka (FSP, LSP)
- Zasto se koristi eksterna memorija (sta su gubici, sta dobici)
- Razliciti tipovi cuvanja podataka (OM, diskovi)
- Struktura magnetnih diskova (kontroler, memorija) i njihovi poddelovi
- Osnovne komponente diska (cilindar, staza, sektor, celija)
- Uspostava adresnog prostora na disku (OS formatira, ucitava memoriju, slobodno, pokvareno itd)
- Kapacitet diska (mnozimo sve delove diska)
- Kapacitet sektora (efektivni + header + prateci deo)
- Vreme pristupa sektoru (kako tacno ide)
- Sprezni podsistem (kako su povezani OM i disk), bandwidth, blokovi, bafer, kontroler i dr.
- Tipovi sistema disk jedinica (klasteri, RAID)
- Performanse diska, od cega zavise i kako ih unaprediti

8) Datotecki sistem
- Cemu sluzi? (FSP, upravlja razmenom podataka app<->datoteka)
- Nivoi usluge (niski visoki)
- Pripadnost nivoa (niski - OS, visoki - OS, lib, subp)
- Usluge nivoa
- Upravljanje eksternom memorijom
- Katalog
- Rutine za upravljanje fizickom razmenom podataka
- Tabele operativnog sistema (*)
    - Tabela uredjaja TU
    - Sistemska tabela datoteke (STD)
    - Alokaciona tabela datoteke (ATD)
    - Tabela logickih imena datoteke (TLI)
    - Tabela otvorenih datoteka (TOD)
    - Tabela opisa datoteke (TOS)

Sistemski pozivi
- Cemu sluze (niski nivo komunikacije)
- Kakvi pozivi postoje i po cemu se razlikuju (sekvencijalni i direktni)
- O cemu pozivi vode racuna (start, curr, end)
- Tipovi sistemskih poziva (create, read, seek...)

KONCEPCIJA BAZE PODATAKA

- Klasicna organizacija podataka je bila vezana za aplikacije
  Svaka aplikacija je imala svoje skladiste podataka.
  Problem? Losa sinhronizovanost. Nepovezanost podataka. Cvrsta povezanost aplikacije i podataka.
  REDUNDANTNOST podataka -> Potreba da se isti podatak cuva u vise fajlova zbog vise aplikacija.

- Svi gorenavedeni problemi su resivi, osim cinjenice da su podaci bili striktno vezani za aplikacije, i da ih je bilo komplikovano upotrebiti van nje.

- Ideja baze podataka:
    - Organizovati sve podatke u jednu veliku datoteku (bazu podataka)
    - Na ovaj nacin bi onda svaka aplikacija pristupala bazi i koristili bi se isti podaci za sve.
- Bilo je neophodno uvesti zaseban softver koji bi kontrolisao bazu podataka (SUBP)

- SUBP (Sistem za upravljanje bazom podataka)
    - Ideja je da postoji program koji ce da se bavi upravljanjem nad bazom
    - Sta je morao da obezbedi:
        - Visekorisnicki pristup (sta ako vise aplikacija trazi podatke odjednom?)
        - Autorizacija korisnika (ne sme svako da vidi sve podatke)
    - Poenta je da svaka aplikacija koja koristi podatke, ne manipulise njima direktno, nego poziva SUBP metode koje su dozvoljene kako
      bi pristupila podacima. SUBP se ponasa kao omotac oko baze, te proverava ko, sta i kako.

    - Sadrzi 3 jezika u sebi:
        1) DDL - Data definition language
            - Jezik pomocu kojeg definise kako ce da izgleda baza. Koristi se za konstruisanje.
        2) DML - Data manipulation language
            - Jezik koji se koristi za manipulaciju podataka (izmene, dodavanje, brisanje itd.)
        3) QL - Query language
            - Jezik koji se koristi za citanje, tj selekciju podataka iz baze

- Sema baze podataka
    - Programi koji koriste usluge SUBP, poznaju samo nacin na koji su podaci organizovani. Ne znaju nista o FSP
    - S obzirom na ceste promene seme, dobro je uvesti koncept PODSEME

- Podsema baze podataka
    - Predstavlja LSO dobijenu na osnovu dela seme BP
    - Jedna podsema je dovoljna(ako je uzeta kako treba) da se za jednu aplikaciju izvrsavaju svi neophodni zadaci.
    - Predstavlja model manjeg/nepotpunog dela realnog sistema
    - Pozeljno je da aplikacije koje koriste BP, to rade preko podsema

- Sta je ideja uvodjenja seme i podseme?
    - Nezavisnost
    - Sema i podsema imaju uloge slicne interfejsima u programiranju. Dakle stvaramo nacin da programi ne budu vezani za same podatke,
      vec za neku apstrakciju, te SUBP moze da se menja, ali nas program ostaje isti.

- Pogled
    - Pojava LSP nad semom ili podsemom
    - Predstavlja sliku baze podataka onako kako je programer vidi
    - Sema ili podsema predstavljaju "prozor" kroz koji programer/korisnik VIDI podatke u BP. Vidi, pa zato pogled.

- Sistem baze podataka
    - Sastoji se od 4 elementa:
        1) Baza podataka
            - Odnosno sami podaci
        2) SUBP (Sistem za upravljanje bazom podataka)
            - Racunari, serveri i softver koji hendluje podatke
        3) Sema baze podataka, koja je implementirana nad SUBP
        4) Jezici i operacije za kreiranje, azuriranje i selektovanje podataka u DB- DDL, DML, QL

MODELI PODATAKA

- Pojam modela podataka
    - Model podatka je matematicka apstrakcija pomocu koje gradimo semu baze podataka
    - Sluzi nam da predstavimo LSO (logicku strukturu obelezja)
    - Osim LSO, pomocu modela mozemo da predstavimo i ogranicenja i odnose medju podacima
    - Laicki receno, model nam sluzi da matematicki skiciramo sta kako iz realnog sveta prenosimo u svet racunara.

      Konkretan primer bi bilo OOP, gde koristimo klase kako bismo mapirali realne objekte u podatke.
      Npr. osobu pretvorimo u

```
Person {
  name : ...,
  surname : ...,
  age : ...
}
```

    jer je to sve sto nam je potrebno za funkcionisanje odredjenog sistema.

- Struktura modela podataka
    - Sastoji se od tri dela (S, I, O)

    1) Strukturalna komponenta
        - Omogucava nam da modeliramo LSO kao staticku strukturu.
    2) Integritetna komponenta
        - Omogucava modeliranje ogranicenja nad podacima
        - "Koji podatak sme da uzme koju vrednost"
    3) Operacijska komponenta
        - Modeliranje dinamike izmene stanja

- Nivoi apstrakcije
    - Postoje 2: INTENZIJA i EKSTENZIJA

    1) Nivo intenzije (kontekst)
        - Nivo tipa
        - LSO je nivo intenzije
        - Npr u Javi/C++, nivo intenzije je klasa, ona samo daje model kako nesto treba da izgleda
    2) Nivo ekstenzije (konkretizacija)
        - Nivo pojave tipa (akcenat na 'pojave')
        - Predstavlja konkretizaciju nivoa intenzije, sa podacima
        - Ako je intenzija klasa, ekstenzija je objekat/instanca te klase

    - Svedeno na cistu Javu bilo bi

```
public class MyObject {
  private int number;                    <--- Intenzija
}
MyObject mObject = new MyObject();       <--- Ekstenzija (instanca)
```

    - Dobar primer je tabela sa primerima kako se intenzije pretapaju u ekstenzije:

Domen	->	Vrednost
Obelezje	->	Podatak
Tip entiteta	->	Pojava tipa entiteta (klasa -> objekat)
Tip poveznika	->	Pojava tipa poveznika

- Razrada komponenti modela podataka
    1) Strukturalna komponenta:
        - Predstavlja koncept klase pojmova
        - Pomocu tih pojmova se predstavljaju objekti iz realnog sveta
        - Sastoji se od:
            - Primitivnih koncepata
                - Svaki sadrzi skup svojih osobina, nacine koriscenja i mogucu semantiku
                - Ovo su primitivne klase u Java/C++, dobijemo ih i ne mozemo da ih razbijemo na nize klase

                  Primera radi, int ne moze da se rastavi na manje delove, niti mozemo da mu menjamo osnovnu semantiku (+, -, ...)

- Skup formalnih pravila
    - Pravila na osnovu kojih od PRIMITIVNIH i SLOZENIH koncepata mozemo da prosirimo model podataka
- Slozenih koncepata
    - Koncepti kojima dodelimo osobine, semantiku i nacin koriscenja
    - Primer slozenog koncepta bi bila klasa u Java/C++.
    Imamo slozen objekat, koji radi samo ono sto mu dozvolimo da radi pomocu metoda.

2) Integritetna komponenta
- Skup TIPOVA OGRANICENJA
    - Uz definiciju kako da se izvrsi validacija datih ogranicenja
    - Uz pravila kako se koriste ogranicenja
- Skup pravila za izvodjenje zakljucka o vazenju ogranicenja
- Skup pravila o kreiranju novih tipova ogranicenja
    - Podjemo od poznatih koncepata da bismo kreirali neki novi tip ogranicenja
- Za sta nam sluze?
    - Dozvoljene vrednosti obelezja (podatka)
    - Moguce vrednosti izmedju razlicitih pojava tipova.
- Primer:
    - OGRANICENJE KLJUCA
    Student ({IME, PRZ, JMBG, GODINE}, {JMBG}) <--- Tip entiteta (LSO, odnosno sema entiteta)

    - OGRANICENJE DOMENA
    Dom(GODINE) ::= {d < 26 & d > 15} <-- Ogranicenje domena za godine (student mora imati izmedju 15 i 26 godina)
        ~ Semantika ovoga nije sasvim korektna

    - OGRANICENJE KARDINALITETA
        - Jedan nastavnik moze predavati najvise jedan predmet
        - Student iz jednog predmeta ima najvise jednu ocenu

- Gde je najbolje uraditi validaciju podataka?
    - Mozemo u transakcione programe (ili na normalnom jeziku aplikacije koje koriste bazu)
    - U SUBP (sistem za upravljanje bazom podataka)
    - Idealno je ugraditi u SUBP tako da nista ne moze da prodje bez verifikacije na "vratima" baze podataka
    - Nije lose ni neka ogranicenja ugraditi u aplikacije, kako bi korisnicima odmah izasla greska ako nesto
    lose uneli (primer web sajtovi kada nam odmah izbace gresku bez slanja podataka na server)

3) Operacijska komponenta
- Modelira dinamiku izmene stanja u BP
- Laicki receno, govori nam kako stvari smeju da se desavaju i sta tacno sme da se desava sa podacima

- Svaka operacija ima:
    - Skup svojih osobina
        - Formalna specifikacija (npr. public void method(arguments...))
        - Izvrsenje nad podacima
    - Skup pravila za koriscenje
    - Opisana moguca semantika

- Koristimo je da definisemo DDL, DML, QL
    - DDL
        - Jezik za kreiranje i modifikaciju seme, fizicke strukture, prava pristupa u BP
        - U principu ovim pravimo bazu podataka, govorimo sta gde i kako
    - DML
        - Tipovi operacija za izmenu stanja BP

- QL
    - Tipovi operacija za selekciju podataka iz BP

- Specifikacija operacije sadrzi sledece komponente:
    - Aktivnost
        - STA zelimo da uradimo (koju akciju)
    - Selekcija
        - Koji deo BP ili deo seme BP zelimo da targetiramo akcijom

- Operacijska komponenta moze biti
    - Proceduralna (low level)
        - Govorimo programu STA i KAKO
        - Primer su svi klasicni programski jezici (C, Java, C++ i dr.)
        - Dakle kazem sta hocemo, ali onda kazemo programu i kako to hocemo tako sto mu damo set instrukcija
          koje on izvrsava kako bismo dosli do cilja. Sami pisemo funkcije i sl.
    - Specifikaciona (high level)
        - Govorimo programu STA
        - Primera radi SQL, kazemo samo SELECT * FROM RADNIK i on zna sta radi, mi ne moramo nista vise da kucamo
        - Da je OOP u pitanju, morali bismo da iteriramo kroz liste, da formiramo odgovor pa da ga vracamo.

- Razliciti tipovi modela podataka

- ER Model
    - Predstavljamo graficki odnos entiteta i poveznika. Na konkretnom BP1 kursu je radjeno tako da je entitet
      pravougaonik, poveznik linija, i postojao je gerund kao romb.

- Hijerarhijski model
    - Strukture stabla (slicno kao sto funckionisu fascikle u ormaricima)
    - Proceduralna operacijska komponenta (iteracije i slicno)

- Mrezni model
    - Nema nista pametno o ovome u prezentaciji. Zamisliti mrezu entiteta koji su medjusobno uvezani ko sa kim treba.

- Relacioni model
    - Stuktura tabela slogova (slog je grupa obelezja? Odnosno jedan slog je valjda jedan konkretan objekat/instanca?)
    - Operacijska komponenta je specifikaciona (neproceduralna)

- Logicki i verovatnosni modeli
    - Uvodjenje dedukcije u bazu podataka? Ovo bi bilo dobro guglati jako je nejasno na prez.

- Objektno orijentisani model
    - Zasnovan na mreznom i semantickom modelu
    - Iza njega stoji OOP paradigma (koncepti klase, operacije, interfejsa, tipa podatka)

- Objektno relacioni model
    - Kombinuje osobine relacionog i OO modela

- XML model
    - Koristi XML jezik za definiciju
    - Ugnjezdena struktura, tj hijerarhijski model podataka
    - Ima specificnu sintaksu sa <tagovima> koji se slazu jedan u drugi

```
<tag1>
    <tag2>
        // content
    </tag2>
</tag1>
```

ER (Entity relationship) diagram

- OSNOVNI POJMOVI

    - Sluzi za graficku, staticku predstavu odnosa entiteta i poveznika.
    - Koristi prethodno navedene pojmove:
        - Obelezje i domen
        - Tip entiteta i tip poveznika (TE, TP)
        - Pojavu tipa entiteta i pojavu tipa poveznika (PTE, PTP)

    - Klasa entiteta
        - Osnovna jedinica poslovanja/posmatranja
        - Predstavlja skup entiteta koji imaju zajednicke osobine
        - Klasican primer entiteta bi bio STUDENT u bazi podataka fakulteta

    - Klasa poveznika
        - Veza izmedju 2 ili vise entiteta ili prethodno napravljenih poveznika
        - Skup poveznika koji poseduju zajednicke osobine

    Primer upotrebe entiteta i poveznika:
        - Recimo da imamo studente na fakultetu.
        - Uz studente imamo i predmete
        - Relacija u stvarnom svetu je STUDENT ----POHADJA---> PREDMET
        - Preneto na model, ovo bi znacilo:
            - Entiteti: Student, Predmet
            - Poveznici: Pohadja
            - U poluformalnom zapisu: Pohadja(Student, Predmet)

- STRUKTURALNA KOMPONENTA

    - Primitivni koncepti strukturalne komponente ER modela:
        - Vrednost (podatka)
        - Domen
        - Obelezje

    Odnosno, kada spojimo u definiciju, svaki atribut ima vrednost koja mora pripadati odredjenom domenu vrednosti.

    - Vrednost je neka vrednost iz bilo kog skupa vrednosti
    - Domen je specifikacija mogucih vrednosti obelezja. Vezuje se za jedan tip vrednosti (npr realni brojevi)
    - Obelezje je naziv obelezja za koje se vezuje vrednost

    - Domen
        - Predefinisani
            - Ugradjen u definiciju modela podataka
            - Zavisi od okruzenja koje koristimo (koji jezik uglavnom)
            - Teoretski imamo (N, Z, Q, R...)
            - Prakticno imamo (int, float, double, string, boolean, ...)
        - Korisnicki definisani
            - Koristimo vec postojece domene da ih definisemo
            - Npr. ako hocemo da definisemo skup vrednosti za ocenu studenta:
              DOMEN_OCENE ::= {d -> N | d >= 5 && d <= 10}
              DOMEN_IME ::= String(30) -> String max duzine 30

    - Obelezje
        - Osobine klase realnih entiteta
        - Svako obelezje ima svoju oznaku
        - Obelezja se u OOP prenose u obicna polja u klasi, dakle (objekat.obelezje)
        - Svako obelezje ima tacno 1 domen vezano za sebe

    - Podatak
        - Uredjena cetvorka (ENTITET, OBELEZJE, VREDNOST, VREME)
        - Entitet

- Oznaka entiteta kojem pripada vrednost
- Obelezje
  - Oznaka obelezja kojem pripada vrednost
- Vrednost
  - Vrednost samog podatka
- Vreme
  - Vremenska odrednica

- Tip entiteta
  - Model klase realnih entiteta u IS
  - Nastaje kao posledica mapiranja realnih entiteta u informacioni svet, radi postizanja odredjenih ciljeva
  - Sastoji se od:
    - Naziva
    - Skupa obelezja
    - Skupa ogranicenja
    - Skupa kljuceva

- Pojava tipa entiteta
  - Jedna instanca tipa entiteta u IS
  - Konkretizovane vrednosti obelezja
  - Predstavlja skup podataka koji se ponasaju onako kako tip entiteta nalaze (definicije, domeni i dr.)

- Tip poveznika
  - Model veze izmedju drugih struktura (TE ili TP)
  - Sastoji se od:
    - Naziva
    - Naziva povezanih TE ili TP
    - Skupa obelezja
    - Ogranicenja TP
    - Skupa kljuceva
  - Poveznici ne moraju da povezuju samo 2 TE/TP, mogu i vise
    Takvi TP se zovu N-arni.
    Npr. poveznik moze da vezuje Studenta, Predmet i Profesora. 3 komada.

- Pojava tipa poveznika
  - Reprezentuje jedan konkretan poveznik, tj instancu poveznika u IS

- INTEGRITETNA KOMPONENTA
  - Tipovi ogranicenja u ER modelu:
    - Ogranicenje domena
    - Ogranicenje pojave tipa
    - Kardinalitet TP
    - Ogranicenje kljuca

  - Ogranicenje domena
    - D(id(D), predef)
      D -> Naziv domena
      id(D) -> Ogranicenje
      predef -> Predefinisana vrednost domena
    - id(D) se sastoji od 3 komponente:
      id(D) = (Tip, Duzina, Uslov)
      - Tip je tip podatka koji se trazi. Jedini obavezan!
      - Duzina je duzina podatka koja je dozvoljena. Navodi se samo kod tipova podataka koji ga zahtevaju (string npr.)
      - Uslov je logicki uslov koji svaki podatak mora da ispuni da bi bio validan (tipa ocena > 4 && ocena < 11)

      - Primeri ogranicenja:
        - DOMEN_DATUMA((Date, -, d >= "1.1.1900."), -)
          ~ Tip podatka je Date, nema ogranicenje duzine, mora biti veci od 1.1.1900. i nema predef vrednost.

- Nula (nedostajuca vrednost)
    - Kada hocemo da naznacimo da nekom polju nedostaje vrednost
    - U programerskom kontekstu, ovo je klasican NULL
    - NULL moze da znaci: Nepoznato, nepostojece, neinformativno.

- Ogranicenje vrednosti obelezja
    - ((Domen, Null), Predef)
        - Domen je dozvoljena vrednost
        - Null je True/False vrednost koja govori DA LI SME atribute da bude nepostojeci. Tipa ako imamo username, to ne sme biti null. Sa druge strane ako imamo broj telefona, to mozda i moze da bude null.

    - Predef je predefinisana vrednost
        - Mozemo je ili mi definisati
        - Ili mozemo da pustimo da uzme default vrednost domena kojem pripada podatak

- Ogranicenje pojave tipa
    id(N) = ({id(N,A)}, Uslov)
    - Ili normalnim jezikom receno, uslov koji svaka pojava tipa mora da zadovolji (entitet ili poveznik)
    - U cemu je razlika u odnosu na ogranicenje vrednosti obelezja?
        ~ Ogranicenje vrednosti obelezja se bazira iskljucivo na tom obelezju i nema uvid u ostala obelezja
        Odnosno, u ogr. obel. mozemo samo da kazemo npr. ovo obelezje mora biti vece od 5.
        Sa druge strane, u ogranicenju pojave tipa, imamo pristup svim obelezjima date pojave tipa,
        i mozemo da sprovodimo provere koje ukljucuju medjusobne odnose vise obelezja.
        Primera radi, student ne sme da bude druga godina i da nema nijedan polozen ispit.
        Student sme da ima obelezje "GODINA = 2" i obelezje "POL_ISPITI=0" kada se gledaju zasebno.
        Ali ako se gledaju zajedno u okviru jedne pojave tipa, ovo nije dozvoljeno, jer bez polozenog
        ijednog ispita, nije mogao da upise drugu godinu.

- KARDINALITET
    - Ogranicava u koliko pojava TP moze ucestvovati pojava povezanog tipa
    - Ili seljackim jezikom, govori u koliko relacija moze da bude objekat. Tipa ako imamo radnika u fabrici, on sme da radi
      SAMO na jednom radnom mestu, ni manje ni vise. To je kardinalitet {1, 1}.
    - Minimalni kardinalitet {0, 1}
    - Maksimalni kardinalitet {1, N}, N >= 2
    - Sto se integriteta kardinaliteta tice, postoji par opcija:
        - Jedan na prema vise
            - Jedan radnik moze imati jednu ulogu, ulogu moze da radi vise radnika
        - Vise na prema vise
            - Jedan radnik moze imati vise uloga, i uloge moze da radi vise radnika
            * Rekurzivni:
                - Jedan radnik moze imati vise sefova, a jedan radnik moze biti sef za vise drugih radnika
                - Entitet je u relaciji sa samim sobom
        - Jedan na prema jedan
            - Jedan radnik ima jednu ulogu, i uloga je samo vezana za jednog radnika

- GERUND
    - U nemackom znaci glagolska imenica
    - U ER modelu predstavlja entitet koji se dobio transformacijom poveznika
    - Gerund je istovremeno i poveznik i entitet
    - Oznacava se na dijagramu bas tako, pravougaonik sa rombom oko sebe/u sebi.
    - Seljackim jezikom receno, gerund bi bio recimo:
        - Imamo studente. Imamo ekskurziju. Studenti ---(idu)---> na ekskurziju.
          Recimo da imamo jos jedan entitet, recimo dodela nagrada za studente sa ekskurzije.

Dakle student --(dobija)---> nagradu. Ali, SAMO studenti koji su BILI na ekskurziji smeju dobiti nagradu.
Znaci onaj poveznik (ide) iz prve recenice, ce nam postati GERUND, koji ce da se ponasa kao entitet, i znacice:
"Ovo je entitet studenata koji su isli na ekskurziju". I onda nagrade povezemo sa tim gerundom da izrazimo uslov.

- AGREGACIJA
    - Objedinjavanje slozenih ER struktura
    - Posmatramo celu ER strukturu kao jedan tip entiteta
    - Gerund je tip agregacije
    - Primer:
        - Imamo radnika u firmi. Firma ima masine na kojima mogu da rade radnici. Ali ne mogu svi radnici da rade na toj masini,
          nego samo oni radnici koji rade na projektima smeju da je koriste.

- TIPOVI ENTITETA
    - Slabi
        - Egzistencijalna zavisnost
            - Njihovo postojanje je uvezano za postojanje drugog entiteta
            - Primer:
                - Radnik i radno mesto.
                - Radno mesto postoji uvek. Radnik radi na njemu.
                - Ako se ukine radno mesto, radnik gubi posao i vise ne postoji.
        - Identifikaciona zavisnost (podskup egzistencijalne zavisnosti)
            - Moraju se identifikovati preko ID-ja super-entiteta
            - Primer:
                - Radnik ima dete koje se belezi u neki spisak.
                - Dete ima svoje ime. Ipak, da bi se dete naslo u bazi, neophodno je uneti ko mu je roditelj.
                - Odnosno identifikacija deteta direktno zavisi od toga ko mu je roditelj, i ne postoji van konteksta roditelja.
            - Kao posledica identifikacije preko super-entiteta, kardinalitet ka super-entitetu mora biti {1,1}
              Jer logicno, ako bi pripadao uz vise super-entiteta, kako bismo ga pronasli, ako s e vezuje uz roditeljski ID.
    - Jaki
        - Postoje nezavisno od drugih

- IS-A HIJERARHIJA
    - Uvodi koncept superklase i potklase
    - Koriste se koncepti generalizacije i specijalizacije
    - Specijalizacija se primenjuje kada superklasa poseduje potklase koje od nje nasledjuju osobine
    - U superklasi ostaju primarni kljuc, i sve zajednicke osobine potklasa

    U potklasu se prenose samo promenljive vrednosti obelezja koje se menjaju od potklase do potklase
    - Kardinalitet potklase:
        - Minimalni:
            - 1 -> Totalna IS-A hijerarhija
            - 0 -> Parcijalna IS-A hijerarhija
            ~ Ako je 1, znaci da superklasa mora biti produzena bar jednom potklasom.
              Ako je 0, znaci da je opcionalno
        - Maksimalni:
            - 1 -> Nepresecna IS-A
            - N -> Presecna IS-A
            ~ Ako je 1, znaci da superklasa moze da se nastavi u samo 1 potklasu
              Ako je N, moze u vise potklasa
    - Primeri:
        {1, 1}
        Radnik moze da bude Programer, Dizajner, ili Sef. Ne moze 2 odjednom, vec samo jedno.
        {1, N}
        Radnik moze da bude Operativac na traci, sef magacina, ili kurir. A moze i sve odjednom, ili 2 odjednom.
        {0, 1}

Radnik moze da bude sef ali ne mora da bude.

- Bitne osobine:
    - Potklasa koristi PK (primarni kljuc) superklase
    - Potklasa moze imati svoj kljuc
    - Potklasa je identifikaciono zavisna od superklase (logicno donekle)
    - Potklasa moze dalje da se produzi svojim potklasama

- KATEGORIZACIJA
    - Koristi se za klasifikovanje TE
    - Svaki TE moze da se uveze sa vise kategorija (klasa TE)
    - POJAVA TE moze da bude u najvise jednoj kategoriji
    - Ne postoji ID-zavisnost izmedju kategorije i TE
    - Primer:
        - Clan kluba moze da bude Fizicko ili Pravno lice
    - Ako sam ja dobro uhvatio razliku izmedju ovoga i IS-A, kategorije nemaju koncept superklase/potklase i nasledjivanja.
      Vec sluze samo kako bi se neki entitet kategorisao.

- ER dijagram
    - Koristi odredjene standarde za graficki prikaz elemenata BP
    - Kako se prikazuju elementi:
        - TE -> Pravougaonik
        - TP -> Paralelogram
        - Domen -> Elipsa
        - Obelezje -> Oblik pilule povezan sa TE/TP kojem pripada

    - Ako imamo PRIMARNI kljuc, podvucemo ga da istaknemo da je primarni
    - Nivoi detaljnosti:
        - Globalni nivo
            - Prikazuje imena TE i TP
        - Detaljni nivo (nivo obelezja i domena)
            - Za svaki entitet prikazuje njegova polja/obelezja, kao i njihove domene

    - Kod projektovanja, proces je uglavnom da
        - Imenica znaci entitet
        - Glagol znaci poveznik
        - "<imenica> koja <glagol>" uglavnom znaci gerund
          'studenti koji su polozili ispit mogu da _____'

RELACIONI MODEL - KONCEPCIJA

- Inicijalni model BP je bio mrezni i hijerarhijski
- Konkretno 70te godine proslog veka pre uvodjenja relacionog modela

- Sta je bio problem?
    - Cvrsta uvezanost programa i baze podataka
    - Neretko je manipulacija podacima bila direktno ugradjena u aplikacije
    - Proceduralni jezici za upravljanje podacima (nema sql nema nista)
    - Razvijanje modela niz put, bez upotrebe matematickih formalizama

- Dalje na scenu nastupa razvoj relacionog modela podataka
- Imao je za ulogu da otkloni nedostatke koje sam gore naveo vezane za mreznu i hij. arhitekturu
- Motiv je bio da se napravi nesto za laksu i opsteprimenjiviju upotrebu
- Insistirano je na matematickim osnovama i formalizmima prilikom razvoja

- Glavni zahtev je bio
    Naciniti programe nezavisno od podataka, tj da BP postoji van konteksta aplikacije

- Strukturalna jednostavnost
    - Upotreba koncepta relacije
        - Relacija predstavlja osnovnu logicku strukturu u RMP
        - Ne sadrzi nikakve informacije o fizickom cuvanju podataka
        - Sluzi da bude razumljiva korisnicima
        - Cesto se koristi tabela za prikaz podataka

    - Selekcija podataka
        - U starijim sistemima, koristilo se adresno pristupanje (apsolutne ili relativne adrese)
        - Kod relacionog modela, upotreba kljuca za pristup
        - Bilo koji podatak se selektuje na uniforman nacin, zadavanjem uslova, koji je najcesce kljuc
        - Kada se zada kljuc, sam SUBP vodi racuna o nacinu pretrage i konvertovanju simbolicke adrese u realnu/fizicku

    - Povezivanje podataka
        - U starijim sistemima upotreba FIZICKIH ADRESA u funkciji pokazivaca.
        - Kod relacionog modela simbolicke adrese, odnosno najcesce kljucevi
        - Uvodjenje pojma Stranog Kljuca (Foreign Key)
        - Aplikacija koja koristi BP takodje ne vodi racuna o smestanju ili nacinu dobavljanja podataka, sve radi SUBP

- Deklarativni jezik
    - Dva alata za query language
        - Relaciona algebra
            - Definisana pomocu teorije skupova i skupovnih operacija
            - Operacije poput Unije, Preseka, Razlike
            - Specijalizovani operatori poput (join, select, i dr.)

        - Relacioni racun
            - Relacioni racun nad n-torkama i domenima

- SQL - Structured Query Language
    - Zasnovan na relacionom racunu nad n-torkama
    - Deklarativni jezik (govorimo sta, ali ne i kako, odnosno samo ZAHTEVAMO)
    - Osnovne komande

        SELECT (lista obelezja)
        FROM (lista relacija)
        WHERE (logicki izraz)

        SELECT Ime, Prezime FROM Radnici WHERE (Godiste > 1975)
            -> Dobavlja ime i prezime svih radnika koji su mladji od 1975-tog godista

- 12 PRAVILA RMP

    0. Sistem mora da bude relacioni, baza podataka, i sistem za upravljanje

    1. Svaka informacija mora da bude predstavljena na samo jedan nacin, pomocu vrednosti u redu tabele, gde su kolone atributi

    2. Svaki podatak mora biti dostupan preko tabele, i primarnog kljuca reda u kom se podatak nalazi

    3. Sistem mora da dozvoli upotrebu NULL vrednosti kao nacin da se specificira odsustvo vrednosti u tom obelezju

    4. Nesto vezano za online katalog, ne kapiram sta hoce da kazu
        'The system must support an online, inline, relational catalog that is accessible to authorized users by means of their
        regular query language. That is, users must be able to access the database's structure (catalog) using the same
        query language that they use to access the database's data.'

5. Sistem mora da podrzi bar jedan relacioni jezik

6. Svi 'pogledi' (?) koji su otvoreni za apdejt, moraju da budu dostupni za izmenu vrednosti od strane sistema

7. Sistem mora da podrzava SET, INSERT, DELETE metode

8. Fizicka nezavisnost, tj. izmene na fizickom nivou ne smeju da se odraze na nacin na koji se radi sa podacima

9. Logicka nezavisnost, tj. izmene na logickom nivou (tabele, kolone, redovi) ne smeju da uticu na nacin na koji aplikacije rade sa BP

10. Integritetna ogranicenja moraju biti zasebno definisana. Promena ogranicenja ne sme bezrazlozno da remeti podatke.

11. Ako BP ima razlicite distribucije na razlicitim mestima, ta distribucija mora ostati sakrivena od korisnika

12. Ako sistem koristi nizi jezik u pozadini, nizi jezik ne sme da se koristi da bi se zaobisla ogranicenja na visem nivou

RELACIONI MODEL PODATAKA

- Od cega se se sastoji?
    - Strukturalna komponenta
        - Primitivni i slozeni koncepti modela podataka
        - Pravila za kreiranje slozenih koncepata
        - ~ Sluzi da se modelira LSO
    - Operacijska komponenta
        - QL, DDL, DML
        - Definise dinamiku izmene stanja
    - Integritetna komponenta
        - Skup tipova ogranicenja
        - Sluzi da definisemo ogranicenja nad podacima (sta sme da uzme koju vrednost)

- Nivoi apstrakcije:
    - Intenzija
        - Nivo konteksta, ili nivo tipa
        - Sluzi da se modelira LSO
    - Ekstenzija
        - Nivo konkretizacije, ili nivo pojave tipa
        - Sluzi da se modelira LSP
        - Ekstenzija predstavlja konkretne podatke stavljene u kontekst intenzije

- Strukturalna komponenta
    - Primitivni koncepti
        - Nivoa intenzije:
            - Obelezje
                - Predstavlja osobinu entiteta ili poveznika u sistemu
            - Domen
                - Predstavlja dozvoljeni opseg vrednosti za obelezje (jedno obelezje = jedan domen, ni manje ni vise)
        - Nivoa ekstenzije
            - Vrednost
    - Slozeni koncepti
        - Kombinuju primitivne i postojece koncepte, pomocu dozvoljenih pravila

    - Svaki entitet ili poveznik ima svoj skup obelezja (T = {A, B, C}, gde je T-tip, a ABC obelezja)

    - Torka (N-torka)
        - Predstavlja jednu pojavu entiteta ili poveznika
        - Pomocu nje, se svakom obelezju iz skupa obelezja TE/TP dodeljuje vrednost

- Npr. jedna torka za studenta bi mogla biti Student = {"Pera", "Peric", "RA-1/2000"}
-> Konkretne vrednosti koje popunjavaju polja koja diktira LSO, tj intenzija

- Skracenja (restriktovana) torka
  - Torka koja sadrzi PODSKUP obelezja neke druge torke
  - (Ja mislim da je ovo objasnjenje) Ako imamo tabelu Radnik u sql, ali nam odatle trebaju samo imena i prezimena
    svih radnika, koristimo komandu 'SELECT ime,prz from Radnik'
    Ta komanda nam vraca niz {ime, prezime} podataka, tj (skracenih)torki od 2 elementa, koje su podskup
    torke Radnik koja ima vise polja (tipa Radnik ima JMBG, dan rodjenja i svasta nesto, ali mi fetchujemo samo ova 2)

- Relacija
  - Konacan skup torki
  - Predstavlja skup realnih entiteta ILI skup poveznika
  - Relacija u kontekstu SQL-a znaci TABELA. Tj. svaka tabela u sql je jedna relacija
  - U relaciji se ne mogu pojaviti 2 identicne torke (sa svim istim obelezjima), jer je to onda i sta torka prikazana dva puta
  - Tabela se logicno predstavlja kao redovi + kolone, redovi su torke, kolone su obelezja
  - Redosled torki u tabeli, i redosled obelezja ne utice na sam sadrzaj tabele

- Sema relacije
  N(R,O)
  - N ~ Naziv seme relacije
  - R ~ Skup obelezja seme relacije (skup obelezja LSO, tj skup kolona tabele)
  - O ~ Skup ogranicenja seme relacije (dozvoljene vrednosti domena)
  Ili ukratko, kako se zove tabela, koje kolone ima, i kakve vrednosti smeju da idu u te kolone. Logicno.
- Pojava nad semom relacije
  (R,O)
  - Bilo koja relacija R, takva da zadovoljava sve O (oznake iz gornjeg zapisa)

- RELACIONA sema baze podataka (nije isto sto i sema relacije)
  (S, I)
  - S ~ Skup sema relacije (skup sema tabela odnosno)
  - I ~ Skup medjurelacionih ogranicenja
  Primera radi, ako imamo seme/tabele Radnik, i Projekat kao entitete, i Angazovanje (radnika na projektu) kao poveznik.
  Relaciona sema bi bila, "Ne moze se isti radnik angazovati na istom projektu vise od jedanput."
  Dakle modelira medjutabelarne odnose.
- Relaciona baza podataka
  - Jedna pojava nad zadatom RELACIONOM semom baze podataka (S, I)
  - Svakoj semi relacije odgovara samo jedna pojava

  -Baza podataka:
    - Predstavlja JEDNO stanje realnog sistema (BP se stalno menja, te stalno prikazuje razlicita stanja)
    - Odnos seme BP i BP, je isti kao i odnos LSO i LSP, kao i nivoa intenzije i ekstenzije.
      Sema predstavlja staticku strukturu BP, dok je sama BP instanca sa realnim v rednostima stavljenim u tu semu.
      Pritom se napominje, da je sema BP sporo promenljiva (retko menjamo strukturu baze), dok se sama BP konstantno menja

Sto se svega ovoga iznad tice (vezano za seme), moze se svesti na sledece objasnjenje:
Baza podataka se bazira na semi baze podataka. Sema joj govori kako izgleda, koja je njena struktura, koje vrednosti smeju polja da imaju.
Sema baze podataka se sastoji od vise sema relacija, i ogranicenja izmedju tih sema relacija.
Sema relacije (sema tabele) se sastoji od osnovnih (donekle i logicnih) komponenti: Naziv, Obelezja, Ogranicenja (iliti, ime tabele, kolone tabele, i ogranicenja za podatke)

- Konzistentnost baze podataka
    1) Formalno konzistentno stanje
        - Ako sve vrednosti relacija zadovoljavaju medjurelaciona ogranicenja
    2) Sustinski konzistentno stanje
        - Ako vazi formalno konz. stanje (sve relacije su korektne)
        - Sve relacije predstavljaju sliku stvarnog stanja sistema (tj. moglo bi se reci, ako je BP up-to-date)
    * SUBP kontrolise samo formalno konzistentno stanje

- Operacijska komponenta
    - Ima 3 potkomponente, tj 3 tipa jezika za razlicite svrhe:
        1) DDL ~ Data definition language
            - Koristi se za upravljanje SEMOM baze podataka
            - Koristimo ga recimo da ubacimo novu tabelu, da izbrisemo staru, da definisemo ogranicenja i slicno
        2) DML ~ Data manipulation language
            - Operacije za upravljanje samim podacima
            - Kada hocemo da dodamo novi podatak, apdejtujemo postojeci, obrisemo postojeci i slicno, koristimo ovaj jezik
            - U sql komande INSERT, DELETE, UPDATE (cini mi se, mozda gresim)
        3) QL ~ Query language
            - Koristi se za pravljenje upita da bismo dobili podatke od baze podataka
            - Primer u sql je ono sto se odmah prvo uci: 'SELECT * FROM Radnik'
            - Sastoji se od:
                - Operatora za sacinjavanje upita (kao keywords)
                - Pravila za formiranje upita/izraza (sintaksa i slicno)
                - Pravila za primenu operatora
            - Vrste upitnih jezika u RMP
                - Relaciona algebra
                    - Zasnovana na teoriji skupova i skupovnih operacija
                    - Unija, presek, razlika
                - Relacioni racun
                    - Logicki racun
                    - Moze biti nad torkama ili na domenima
    - Selekcija
        - Komanda pomocu koje vrsimo selekciju podataka iz relacije
        - Omogucava izbor torki po nekom kriterujumu F
        - F je logicka formula (mora da se evaluira na TRUE/FALSE, inace ne valja)
            npr. SELECT * FROM Radnik WHERE 'GodRodjenja > 1980' => Ovo na kraju je boolean logicki izraz

    - Projekcija (restrikcija) relacije
        - Izdvajanje restriktovanih torki iz relacije na osnovu nekog kriterijuma
        - Formalnije, projektovanje relacije na podskup skupa njenih obelezja
        - Seljackim jezikom, isto sto sam napisao za restriktovanu torku, samo sto ovo vazi ne samo za jednu torku, nego za sve torke iz relacije.

    - Prirodni spoj relacija
        - Spajanje torki iz razlicitih relacija na osnovu istih ZAJEDNICKIH obelezja
            Tipa ako imamo u 2 tabele/relacije nesto poput
            a1 b1 c1 e1 g1 b1
            To mozemo da iskombinujemo u
            a1 b1 c1 g1 e1
    - Dekartov proizvod relacija
        - Spajanje kada formiramo sve moguce kombinacije torki iz 2 relacije
    - Theta spajanje
        - Spajanje torki po nekom kriterijumu

- Integritetna komponenta (ogranicenja)
    - Definisana putem tipova ogranicenja
    - Sadrzi:

- Definiciju (formalnu)
- Nacin za validaciju

- Karakterisitke:
  - Skup operacija kojima se moze dovesti do narusenja ogranicenja (opasnost)
  - Skup akcija kojima se obezbedjuje ocuvanje validnosti baze podataka (odbrana)

- Tipovi ogranicenja u RMP:
  - Domena
  - Vrednosti obelezja
  - Torke (medju obelezjima jedne torke)
  - Kljuca
  - Jedinstvenosti

- Oblast definisanosti:
  - Vanrelaciono
    - Definise se izvan seme
  - Jednorelaciono
    - Definise se u okviru jedne seme/relacije
  - Viserelaciono
    - Definise se za vise sema/relacija

- Ogranicenje domena
  D(id(D), predef)
  - D ~ naziv domena
  - id(D) ~ Ogranicenje domena (T, D, U) => Tip, duzina, uslov (tip jedini obavezan)
  - Predef ~ Predefinisana vrednost domena

- Ogranicenja obelezja
  id(N, A) = (Domen, Null, Uslov)
  - Domen ~ naziv domena za to obelezje (domen je ovo od gore, TDU, tako da je ogr. obelezja ustv ((TDU) Null Uslov)
  - Null ~ da li polje sme biti nepostojece
  - Uslov ~ uslov koji svako obelezje tog imena mora da ispuni

  * Napomena: uslov domena nije isto sto i uslov obelezja.
  Intedzer kao domen sme da bude -1. Godine u polju korisnika ne smeju biti -1.
  - Domen i Null su obavezni kod navodjenja! Uslov nije. Ako uslov nije definisan, nasledjuje se domenski uslov.

- Ogranicenje torke
  - Izrazava potencijalna ogranicenja izmedju vrednosti obelezja u okviru jedne torke
  - Sta ovo znaci? Pa recimo, glup primer, ako korisnik odvojeno unosi godiste i broj godina, moze da unese godiste 1970
  a da stavi da ima 15 godina. Ove 2 vrednosti su u kontekstu ogranicenja obelezja u redu, ispunjavaju uslov domena.
  Sa aspekta cele torke, ovo nema smisla. Dakle u ovim ogranicenjima se bavimo medjusobnim odnosom vrednosti obelezja, tj.
  gledamo da li su medjusobno korektne.

- Ogranicenje kljuca
  * Za sva obelezja kljuca, vrednost NULL je zabranjena!
    - Svaka sema/relacija/tabela (ovo je sve jedno ime za istu stvar) mora da ima J jedan primarni kljuc (kljuc moze biti vise obelezja)
  * Ogranicenje jedinstvenosti
    - Zahteva da svaki kljuc u okviru relacije bude jedinstven
    Tj. da za svaku torku, kombinacija obelezja koja sluze kao identifikator, budu unikatna u toj relaciji.
    Npr za studente, imamo jedinstven broj indeksa (SMER-XXX-YYYY)
    Za ulogu radnika u fabrici bismo mogli imati 2 obelezja kao kljuc, {SIFRA_RADNIKA + SIFRA_ULOGE}, ako bi jednom

radniku bilo dozvoljeno da radi vise poslova/uloga odjednom.

- Vrste kljuceva:
  - Primarno (glavno) obelezje
  - Neprimarno (sporedno) obelezje

- Skup svih ogranicenja seme relacije
  - Kombinacija:
    Ogranicenja kljuceva + Ogranicenja jedinstvenosti + Ogranicenja torke (torka preuzima domen i obelezja na sebe)
  - Primera radi za tabelu Radnik = (MBR, IME, PRZ, JMBG, GOD), imamo sledece:
    K = {MBR} => Primarni kljuc
    UNIQUE(JMBG)=> Svaki JMBG mora biti jedinstven
    id(Radnik) => Svaki Radnik mora da ispuni uslove koje mi definisemo u SUBP

// Nedostaju zavisnost sadrzavanja, funkcionalna zavisnost, armstrongova pravila, ogranicenje referencijalnog integriteta

// Nedostaje Univerzalna relacija