PODATKOVNE BAZE ZA MEDIJE

Predavateljica:
Tatjana Welzer Družovec
welzer@uni-mb.si
Kabinet: G2-M.23

Asistent:
Andrej Krajnc
andrej.krajnc@um.si
Kabinet: 1.nadstropje

1. OSNOVNI POJMI

IZHODIŠČE:

Strokovni pogovor je uspešen le ob istovetnem razumevanju vsebine strokovnih pojmov.

1.1. DEFINICIJE PODATKOVNIH BAZ

- PB je zbirka med seboj povezanih podatkov o organiziranem delovno zaključenem sistemu (angl. *enterprise*), ki so namenjeni različnim uporabnikom.

- PB je zbirka med seboj pomensko povezanih podatkov, ki so shranjeni v računalniškem sistemu, dostop do njih je centraliziran in omogočen s pomočjo sistema za upravljanje s podatkovno bazo.

1.1. DEFINICIJE PODATKOVNIH BAZ

- *N. Ryan, D. Smith: Database Systems Engineering*

  PB je skladna zbirka povezanih podatkov, ki predstavljajo model določene (za skupino ljudi zanimive) domene, imenovane tudi “univerzum aplikacije” (angl. *Universe of Discourse*).

- *P. Rob, C. Cornel: Database Systems*

  PB je namenjena podpori sprejemanja poslovnih odločitev na vseh nivojih organizacije.

1.1. DEFINICIJE PODATKOVNIH BAZ

- *T. J. Teorey: Database Systems*

  PB je zbirka medsebojno povezanih shranjenih podatkov, ki zadovoljujejo potrebe različnih uporabnikov znotraj ene ali več organizacij.

- *T. Connoly, C. Begg, A. Strachan: Database Systems*

  PB je zbirka logično povezanih podatkov in njihovih opisov (katere si delijo različni uporabniki), oblikovanih z namenom zadovoljitve informacijskih potreb organizacije.

1.1. DEFINICIJE PODATKOVNIH BAZ

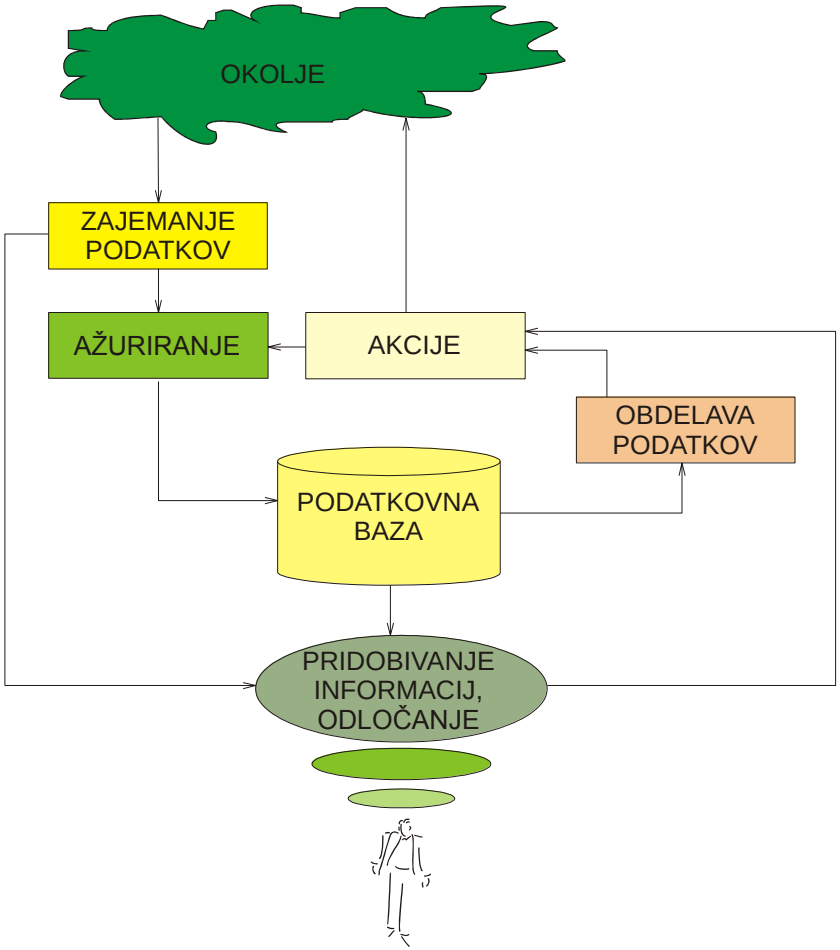
- *IBM: IMS/VS*

  PB je neredundantna zbirka vzajemno povezanih podatkov, ki se uporabljajo za izvajanje ene ali več aplikacij.

- *T. Mohorič: Uvod v podatkovne baze*

  PB je model okolja, ki služi kot osnova za sprejemanje odločitev in izvajanje akcij.

1.1. DEFINICIJE PODATKOVNIH BAZ



Slika 1: Podatkovna baza kot model okolja

1.2. DEFINICIJA PODATKA

- Simbolična predstavitev preprostih spoznanj o obravnavanem svetu.
- Podatek je poljubna predstavitev s pomočjo simbolov ali analognih veličin, ki ji je pripisan ali se ji lahko pripiše nek pomen.
- Podatek je predstavitev dejstva, koncepta ali instrukcije na formalen način (ANSI, ISO).

1.2. DEFINICIJA PODATKA

- Podatki so dejstva, predstavljena z vrednostmi (številke, znaki, simboli), ki imajo pomen v določenem kontekstu (G. C. Everest).

- *M. J. Hernandez: Database Design for Mere Mortals*

  Podatki so statične vrednosti shranjene v PB.

- *P. Rob, C. Coronel: Database Systems*

  Podatki so gola dejstva, zanimiva za končnega uporabnika.

1.3. DEFINICIJA INFORMACIJE

- Informacija je spoznanje, ki poveča vsebino znanja sprejemnika.
- Informacija je pomen, ki ga človek pripiše podatkom s pomočjo znanih konvencij, ki so uporabljene pri njihovi predstavitvi (ANSI, ISO).
- Informacija so ovrednoteni podatki v specifični situaciji (G. C. Everest).
- Informacija je novo spoznanje, ki ga človek doda svojemu poznavanju sveta.

1.3. DEFINICIJA INFORMACIJE

- *M. J. Hernandez: Database Design for Mere Mortals*

  Informacija je podatek, ki je procesiran tako, da zadovoljuje potrebe posameznika.

- *P. Rob, C. Coronel: Database Systems*

  Informacije so dejstva (podatki) prisotna v pomenskih vzorcih.

- *F.R. McFodden: Database Management*

  Informacije so dejstva, ki so bila procesirana in prikazana v formatu, primernem za sprejemanje odločitev.

1.4. ENAKOST IN RAZLIKA, PODATKI TER INFORMACIJE

- Odnos med podatkom in informacijo predstavimo s formulo:

$$I = i(P, S, t)$$

I - informacija
P - podatek
S - sprejemna struktura
t - čas

- Podatek in informacija sta dva pojma, ki se pogosto uporabljata tudi kot **sinonima**, kar pa **NI DOPUSTNO**!

1.5. OSTALE POMEMBNE DEFINICIJE

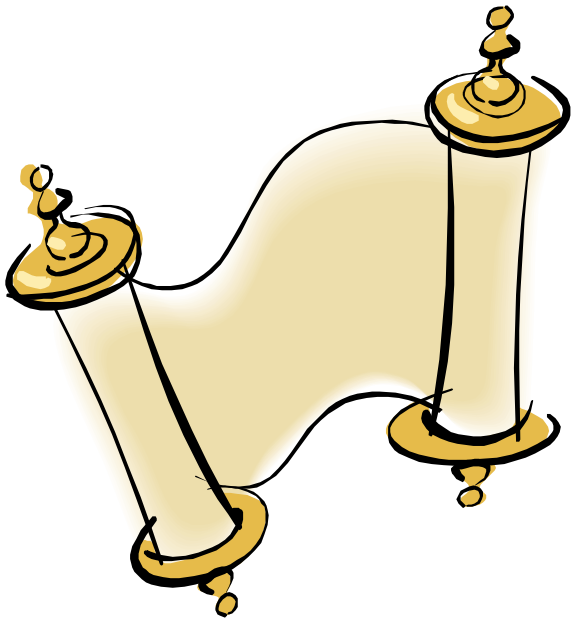
- **Baza znanja** vsebuje zbir preprostih dejstev in eksplicitno izraženih splošnih pravil, ki skupaj predstavljajo podobo obravnavanega sveta.

- **Sistem za upravljanje podatkovne baze** – SUPB (*angl. DataBase Management System - DBMS*) je programski produkt, ki uporabniku omogoča delo s podatkvno bazo in hkrati nadzoruje dostop do podatkovne baze.

- **Meta podatek** je “podatek o podatku”, s pomočjo katerega je izvedena povezava podatkov.

1.6. ZGODOVINA

- **Najbolj razširjen** medij za shranjevanje podatkov je???



1.6.1. PODATKOVNA REVOLUCIJA

- Eden izmed možnih načinov zapisovanja in shranjevanja podatkov pa je tudi računalniško podprto shranjevanje – uporaba podatkovnega sistema.

- **Prednosti:**
  - shranjevanje velikih količin podatkov s hitrim dostopom,
  - hiter in natančen prenos podatkov,
  - hitrer in natančne obdelave ter preoblikovanje podatkov.

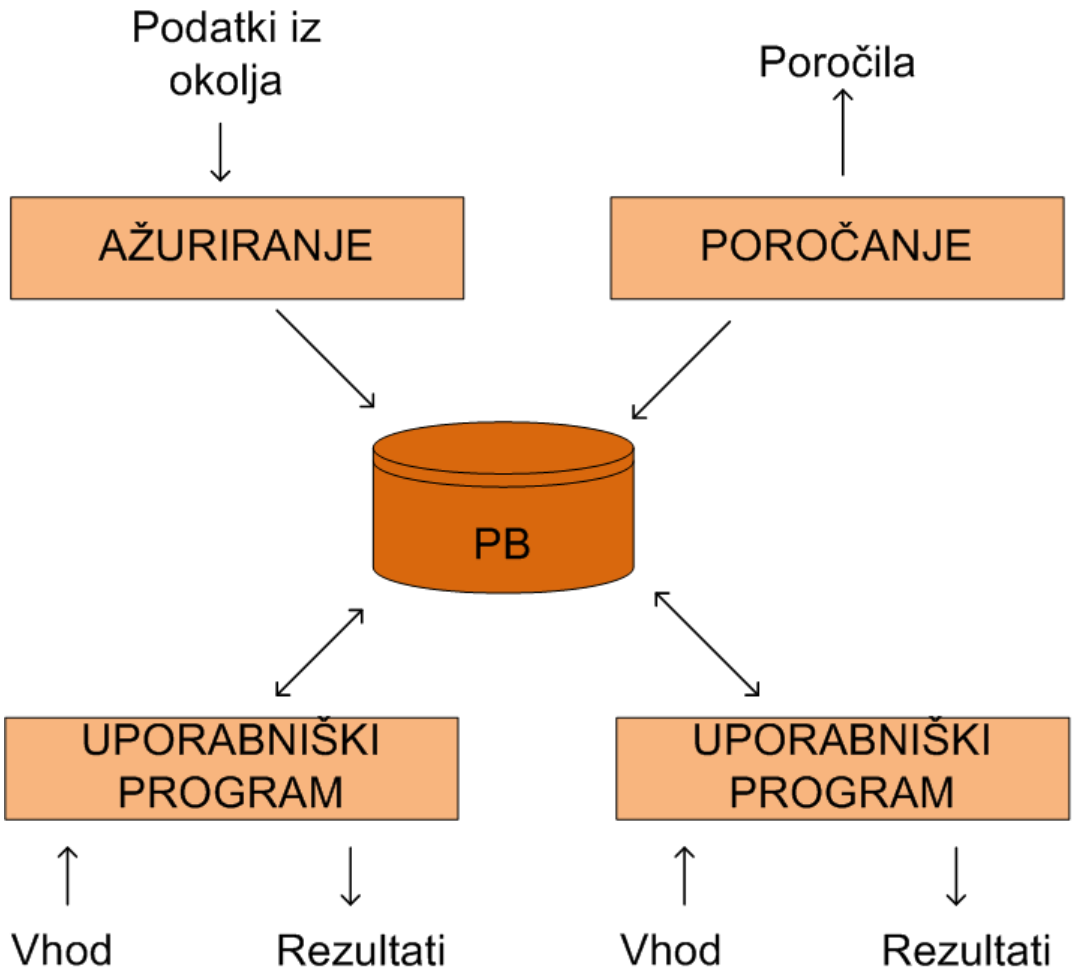
- **Podatkovni sistem:**
  - človek,
  - program,
  - podatki,
  - računalnik.

1.6.1. PODATKOVNA REVOLUCIJA

- **Prvo obdobje** računalništva:
  - računalnik (strojna oprema)
- **Drugo obdobje** računalništva: programska revolucija
  - program (programski jezik)
- **Tretje obdobje** računalništva: podatkovna revolucija
  - podatki (podatkovna baza)

KOPERNIANSKA REVOLUCIJA –
SONCE SO PODATKI

1.6.1. PODATKOVNA REVOLUCIJA



Slika 2: V središču pozornosti so podatki

1.6.2. PRELOMNICE V ZGODOVINI PB

- **60. leta**
  - Rojstvo podatkovne baze v poznih 60-ih letih z IMS - hierarhični SUPB proizvajalca IBM.
  - V letih 1965 - 1971 aktivnosti delovne skupine DBTG (DataBase Task Group) v okviru CODASYL (Conference on Data Systems Languages), ki se zaključi z mrežnim SUPB.
  - Hierarhični + mrežni SUPB = **1. generacija SUPB**

1.6.2. PRELOMNICE V ZGODOVINI PB

- **70. leta**
  - Leta 1970 Ted Codd objavi temeljni članek, ki omogoča uvedbo relacijskega podatkovnega modela.
  - Leta 1976 prestavitev entitetno-relacijskega (E-R) podatkovnega modela (P. Chen).
  - Leta 1979 nekaj popravkov T. Codd-a.
  - V pozni sedemdesetih letih se pojavi System R, eden prvih predstavnikov relacijskega SUPB.
  - Relacijski SUPB = **2. generacija SUPB**

1.6.2. PRELOMNICE V ZGODOVINI PB

- **80. leta**
  - V zgodnjih 80-ih letih se pojavijo prvi objektno-orientirani podatkovni modeli.
  - Rodi se **3. generacija SUPB**

1.6.2. PRELOMNICE V ZGODOVINI PB

- **Danes**
  - Ni industrije ali dejavnosti, ki ne bi bila tako ali drugače povezana s podatkovnimi bazami.

- **Raziskovalne teme današnjih dni in prihodnosti:**
  - transakcijski modeli,
  - optimizacija povpraševanj,
  - prenos podatkov,
  - varnost,
  - podatkovno modeliranje,
  - modeli za specialna področja.

2. OBLIKOVANJE PODATKOVNE BAZE

Izhodišče oblikovanja PB

- PB predstavlja jedro (srce) informacijskega sistema.

Cilj oblikovanja PB

- Učinkovita podatkovna baza, ki:
  - zadovolji vse informacijske zahteve možnih potencialnih uporabnikov za podano področje uporabe,
  - zagotovi “naravno” in lahko razumljivo strukturiranje informacijske vsebine,
  - ohrani celotno semantično informacijo oblikovanja za poznejše preoblikovanje,
  - doseže vse zahteve procesiranja in visoko stopnjo učinkovitosti procesiranja,
  - doseže logično neodvisnost za vprašanja na tem nivoju.

2. OBLIKOVANJE PODATKOVNE BAZE

Izvajalec oblikovanja PB

- Oblikovalec PB
- Nespecialisti

Posledice slabega oblikovanja PB

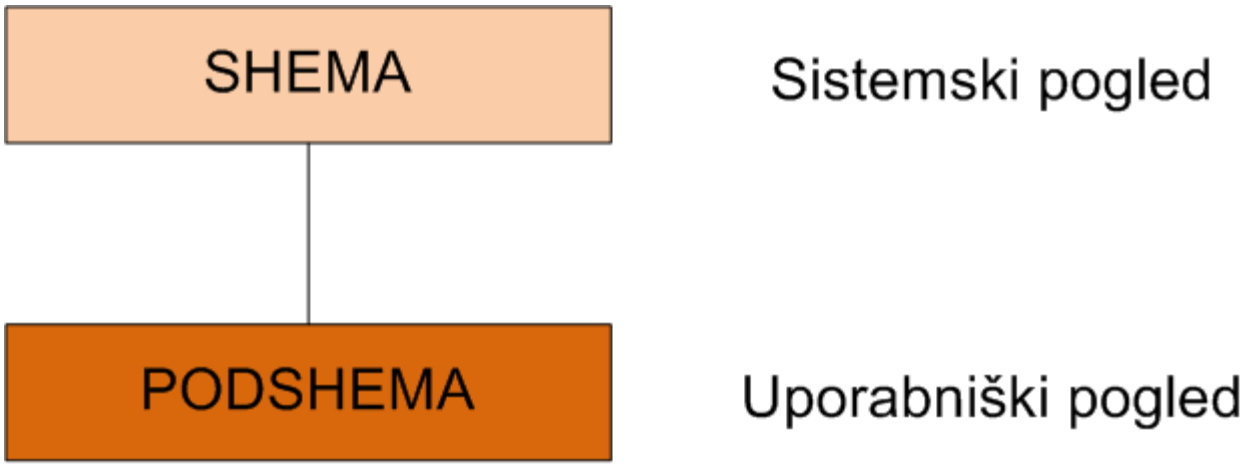
- Slabe odločitve.

Posledice dobrega oblikovanja PB

- Enostavno spreminjanje in vzdrževanje strukture PB.
- Enostavno spreminjanje podatkov.
- Enostavno pridobivanje informacij.
- Enostavno oblikovaje aplikacij.

2.1. ARHITEKTURA PB

- 1971 – DBTG (DataBase Task Group): **dvonivojska arhitektura**



Slika 3: Dvonivojska arhitektura

2.1. ARHITEKTURA PB

- 1975 –

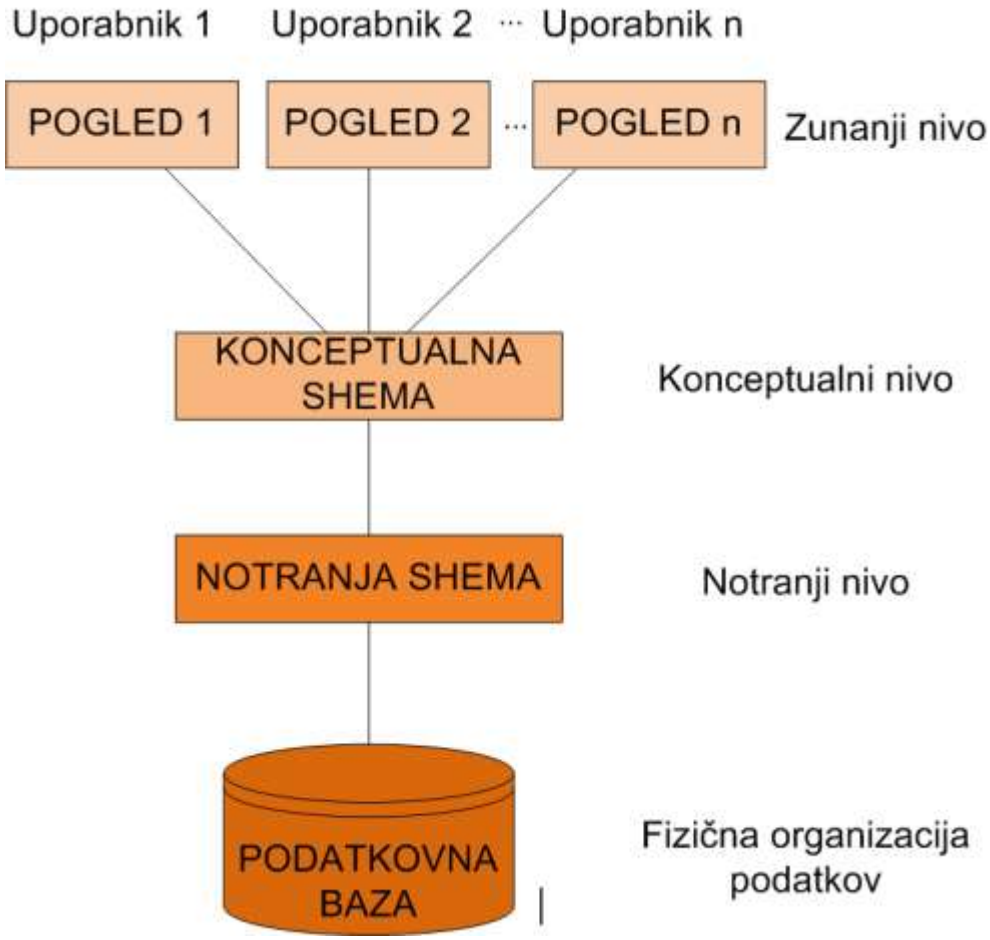
  ANSI-SPARC **trinivojska arhitektura**

ANSI – American National Standard Institute

SPARC – Standard Planning And Requirements Comitee

OPOZORILO!

Ni standarda!



Slika 3: ANSI-SPARC tri-nivojska arhitektura

2.1. ARHITEKTURA PB

Osnovni cilj

- Ločitev izbranega uporabniškega pogleda od njegove fizične predstavitve.

- Razlogi za ločitev:
  - vsi uporabniki uporabljajo iste podatke in lahko spreminjajo svoje poglede,
  - uporabnik naj ne bi imel dostopa do fizičnih podatkov,
  - sprememba podatkovne strukture ne sme vplivati na uporabniške poglede,
  - fizične spremembe naj nimajo vpliva na interno shemo,
  - administrator lahko spremeni konceptualno ali globalno strukturo ne da bi pri tem vplival na uporabnika.

2.1. ARHITEKTURA PB

Zunanji nivo (angl. external level)

- Uporabniški pogled na podatkovno bazo oz. opis dela podatkovne baze, pomembnega za določenega uporabnika, predstavljen z:
  - entitetami, atributi, relacijami lastnega realnega okolja. Različne predstavitve istih podatkov.

Konceptualni nivo (angl. conceptual level)

- Skupen pogled na podatkovno bazo (angl. global view), predstavljen z:
  - vsemi entitetami, relacijami in pripadajočimi atributi,
  - omejitvami,
  - semantičnimi informacijami o podatkih,
  - informacijami, vezanimi na varnost in integriteto.

2.1. ARHITEKTURA PB

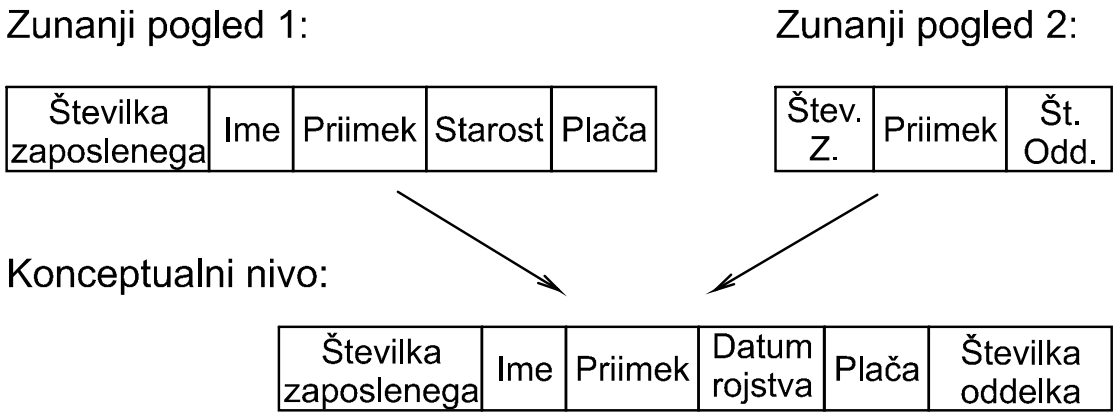
Notranji nivo (angl. internal level)

- Fizična predstavitev podatkovne baze na računalniku. Podan je opis, kako so podatki shranjeni v PB:
  - dodelitev spomina za podatke in indekse,
  - opis zapisov skupaj s podatki,
  - enkripcijske tehnike in stiskanje podatkov.

Fizična organizacija podatkov

- Zanjo je zadolžen operacijski sistem ob podpori SUPB.

2.1. ARHITEKTURA PB



Notranji nivo:

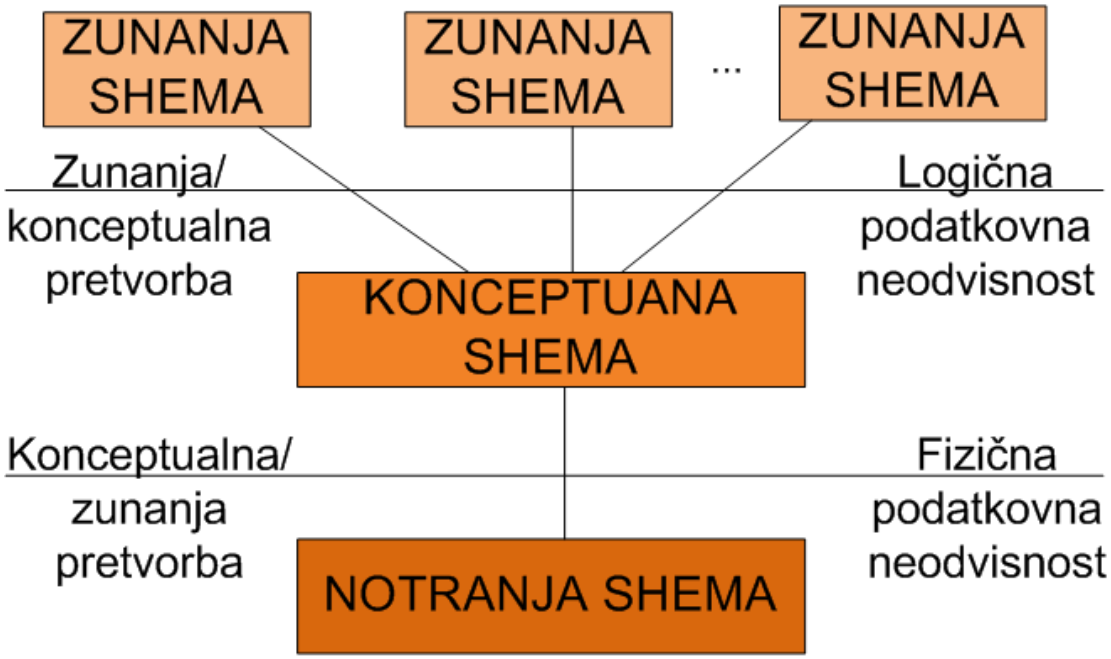
```
Struct OSEBJE {
   int Stevilka_Zaposlenega;
   int Stevilka_Oddelka;
   char Ime[15];
   char Priimek[15];
   Struct date Datum_rojstva;
   float Placa;
   Struct OSEBJE *next;        /* kazalec na naslednje polje */
}

Index Stevilka_Zaposlenega;  /*definicija indeksov za osebje */
Index Stevilka_Oddelka;
```

Slika 4: Primer ANSI-SPARC trinivojska arhitektura

2.2.1. NEODVISNOSTI V ANSI-SPARC ARHITEKTURI

- Trinivojska arhitektura zagotavlja podatkovno neodvisnost.
- Sprememba nižjega nivoja ne vplivajo na višji nivo.



Slika 5: Neodvisnosti v ANSI-SPARC trinivojski arhitekturi

2.2.1. NEODVISNOSTI V ANSI-SPARC ARHITEKTURI

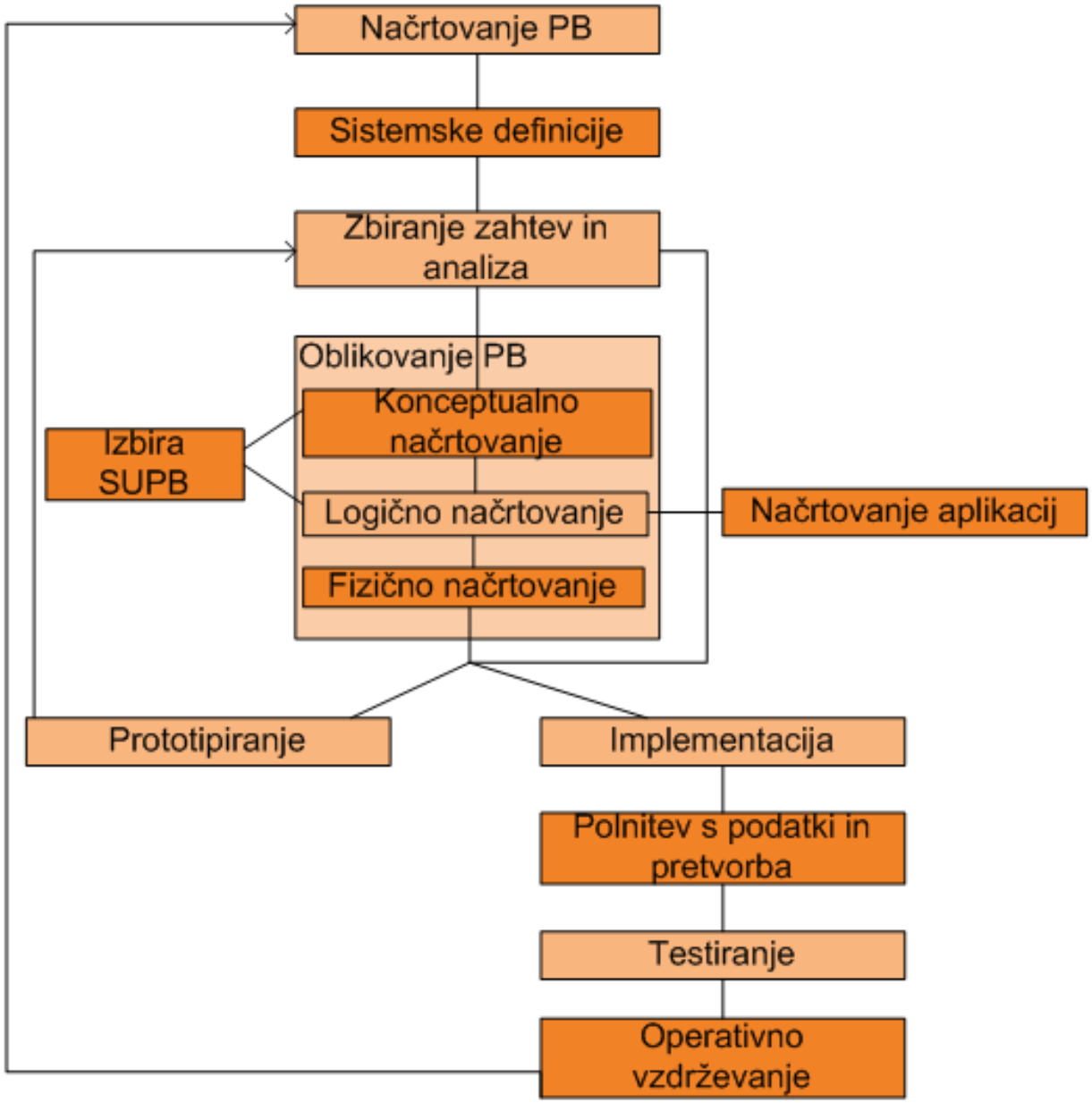
Logična podatkovna neodvisnost

- Zunanji modeli (sheme) so imuni na spremembe v konceptualnem modelu (shemi).

Fizična podatkovna neodvisnost

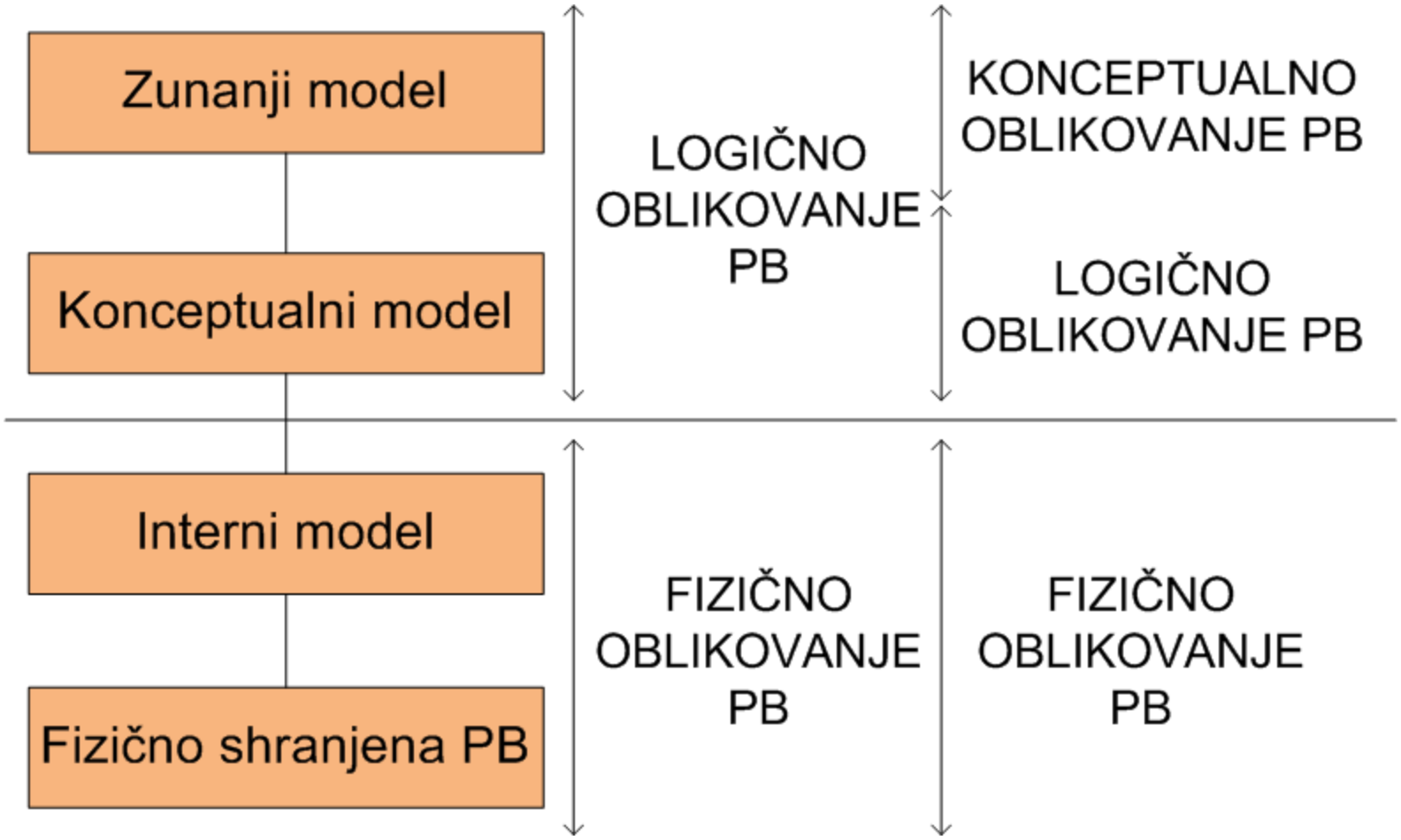
- Konceptualni model (shema) je imuna na spremembe v notranjem modelu (shemi).

2.3. ŽIVLJENJSKI KROG PB



Slika 6: Življenjski krog PB

2.4. PREGLED OBLIKOVANJA PB



Slika 7: Oblikovanje PB in ANSI SPARC arhitektura

2.4. PREGLED OBLIKOVANJA PB

- Oblikovanje podatkovne baze praviloma poteka v 3 fazah:
  - **Konceptualno oblikovanje**
    - Oblikujemo model za informacijsko uporabo izbranega delovno zaključenega organiziranega sistema (angl. enterprise), ki je popolnoma neodvisen od logičnega in fizičnega oblikovanja.
  - **Logično oblikovanje**
    - Oblikujemo model izbranega delovno zaključenega organiziranega sistema za ciljno skupino sistema za upravljanje s podatkovno bazo - SUPB.
  - **Fizično oblikovanje**
    - Proces priprave opisa implementacije podatkovne baze v sekundarnem pomnilniku (opis podatkovne strukture in metod dostopa) za izbrani ciljni SUPB.

2.5 KONCEPTUALNO MODELIRANJE PB

- Izhodiščna točka oblikovanja, katere rezultat je abstrakten in splošen opis realnosti, se imenuje konceptualni model.

- Uporaben je za različne namene:
  - na začetku oblikovanja povezuje različne interese in vidike končnega uporabnika,
  - je uporaben opis, primeren za komunikacijo z uporabniki kakor tudi z nepoznavalci semantike,
  - oblikovalcu podatkovne baze omogoča izgradnjo stabilnega sistema podatkovne baze,
  - omogoča učinkovito predstavitev pravkar oblikovane podatkovne baze.

2.5 KONCEPTUALNO MODELIRANJE PB

- Kaj je podatkovni model?
  - Kombinacija konstruktov, uporabljenih za organizacijo podatkov,
  - predstavitev (običajno grafična) podatkovne strukture kompleksnega “realnega sveta”,
  - zbirka logičnih konstruktov, uporabljena za opis in predstavitev podatkovne strukture, povezav in operacij,
  - zbirka visokonivojskih podatkovnih opisov,
  - model – opis ali analogija, uporabljena za vizualizacijo nečesa, kar ni možno direktno opazovati (npr. nezgrajen most) – Webster’s dictionary.

2.5 KONCEPTUALNO MODELIRANJE PB

- Kaj je podatkovni model? (nadaljevanje)
  - Abstrakcija kompleksnega realnega sveta,
  - relativno preprosta predstavitev (običajno grafična) kompleksnih podatkovnih struktur realnega sveta,
  - komunikacijsko orodje, ki olajša interakcije med načrtovalci, aplikacijskimi programerji in uporabniki ter omogoča razumevanje organizacije:
    - “Ustanovil sem to podjetje, leta sem opravljal ta posel in to je sedaj prvič, da resnično razumem, kako se vsi delci skladajo, kako delujejo in kako se povezujejo”.

2.5 KONCEPTUALNO MODELIRANJE PB

- Zakaj in od kdaj podatkovno modeliranje?
  - Najprej identificiramo z uporabo konceptualne analize preproste elemente, na katere prevedemo vse kompleksne elemente. Nato izvedemo sintezo razumevanja celote z zaznavanjem potrebnih relacij, ki prej omenjene elemente povezujejo. Avtor?
    - René Descartes
    - Clive Finkelstein (oče informacijskega inženiringa)
    - PowerDesigner Data Architect (User's manual)
    - Peter Chen (oče E-R modela)

2.5 KONCEPTUALNO MODELIRANJE PB

- Namen konceptualnega modeliranja je doseči cilj:

VSE, KAR JE POTREBNO, JE TU

in

VSE, KAR JE TU, JE POTREBNO.

- Koraki oblikovanja konceptualnega modela PB:
  - podatkovna analiza in zbiranje zahtev,
  - oblikovanje E-R modela,
  - normalizacija.

2.5.1 PODATKOVNA ANALIZA IN ZBIRANJE ZAHTEV

Opredelitev skupin uporabnikov in področij uporabe:

- opredelitev zaključenega organiziranega sistema in aplikacije,
- opredelitev uporabnikov informacij in njihovih pogledov na PB,
- uporaba informacij.

Analiza operativnega okolja in zahtev procesiranja:

- opredelitev trenutne in bodoče uporabe informacij,
- pogostost uporabe podatkov,
- opredelitev pretvorb, potrebnih za zagotavljanje informacij.

Proučitev izvorov informacij in podatkov:

- pregled obstoječe dokumentacije in sistemov,
- povpraševanja in intervjuji.

2.5.1 PODATKOVNA ANALIZA IN ZBIRANJE ZAHTEV

Vsebina vprašalnika:

- Ime in opis entitete
  - Čeprav že ime entitete nedvoumno predstavi entiteto (predmet), podamo tudi njen opis, ki predstavi uporabo entitete in njene osnovne funkcije.

- Arhiviranje
  - Podamo dobo in način hranjenja podatkov, kakor tudi vzrok, če je znan (npr. zakonski predpisi).

2.5.1 PODATKOVNA ANALIZA IN ZBIRANJE ZAHTEV

- Atribut
  - Za vsak “delček informacije” (atribut), ki opisuje entiteto, poskušamo zagotoviti naslednje informacije:
    - ime in opis: podamo seznam imen, sinonimov in akronimov, ter kratek opis posameznega atributa,
    - izvor (vir) atributa: podan z organizacijskega vidika,
    - lastnost: podamo tip atributa (numerični, znakovni, itd.) ter dopustne in mejne vrednosti,
    - uporaba: navedemo podatke o uporabniku in pogostost uporabe atributa,
    - varnost: podani naj bodo podatki o tem, kdo in kako ima dostop do atributa (branje, vpisovanje, popravljanje, brisanje),
    - pomembnost: kakšna je stopnja pomembnosti atributa za entiteto in celoten organiziran zaključen sistem (nujno potrebni, potrebni, priporočljivi, itd.) .

2.6 ENTITETNO-RELACIJSKI MODEL

- E-R model zagotavlja sistematično predstavitev entitet in relacij, ki dopolnjujejo filozofski pogled na entitete, relacije in omejitve, s ciljem zajeti vse neločljive pomene posamezne aplikacije.

- Najpomembnejši prispevek E-R modela predstavlja diagramska tehnika, ki na jedrnat in opisen način predstavlja aplikacijo.

- E-R diagram predstavlja komunikacijsko orodje za oblikovanje podatkovne baze, zagotavlja notacijo za dokumentiranje oblikovanja PB in s tem predstavitev najpomembnejših lastnosti le-te.

2.6.1 ZGODOVINSKI RAZVOJ E-R MODELA

Vzroki za nastanek

- Oblikovanje skupnega koncepta na osnovi obstoječih elementov (Honeywell).
- Problemi uporabnikov - potreba in želja po metodologiji za predstavitev PB (MIT).

Rešitev

- Slika pove več kot beseda.

Rezultat

- E-R model: *marec 1976: ACM TODS Vol. 1, No. 1, 9-37:* "The Entity Relationship Model: Toward an Unified View of Data"

2.6.1 ZGODOVINSKI RAZVOJ E-R MODELA

Trenutno stanje

- Aktivna uporaba osnovne verzije in številnih sintaktično in semantično dopolnjenih razširitev.

Prednosti

- Enostavno in hitro učenje ter razširjena uporaba v praksi.
- Razširjenost v literaturi.
- Enostavna in čitljiva predstavitev.
- Združljivost s pripomočki, ki jih vsebujejo SUPB.

Slabosti

- Ne obstaja komercialni produkt, ki bi omogočal direktno implementacijo E-R modela.
- Pretvorba v ustrezno shemo izbranega podatkovnega modela.

2.6.2 OSNOVNI GRADNIKI E-R MODELA

- **Entiteta:**
  - je neodvisni podatkovni objekt (fizični, konceptualni) zaključenega organiziranega sistema, ki je po definiciji nosilec podatkov. Lastnosti entitet opisujejo atributi.

- **Entitetni tip:**
  - je množica entitet, ki jih opisujejo isti atributi.

- **Šibka entiteta:**
  - je entiteta brez lastnega ključnega atributa (ni razpoznavna sama po sebi). Vedno je predstavljena skupaj z močno entiteto (v relaciji z njo), katere ključ je predstavljen iz lastnih atributov.

2.6.2 OSNOVNI GRADNIKI E-R MODELA

- **Relacija:**
  - je povezava med dvema ali več entitetami.

- **Relacijski tip:**
  - je povezava med dvema ali več entitetnimi tipi.

POZOR!

Relacija sama po sebi ne obstaja, niti konceptualno niti fizično!

- **Atribut:**
  - zagotavlja informacije o entitetah in tudi relacijah z opisom njihovih lastnosti.

2.6.2 OSNOVNI GRADNIKI E-R MODELA

- **Ključ (identifikator):**
  - Vodilni atribut, ki omogoča identifikacijo posamezne entitete. Ločimo:
    - kandidacijski
    - primarni
    - sekundarni
    - sestavljen in
    - zunanji (tuji) ključ

- **Domena atributa:**
  - je množica dovoljenih vrednosti za posamezen atribut.

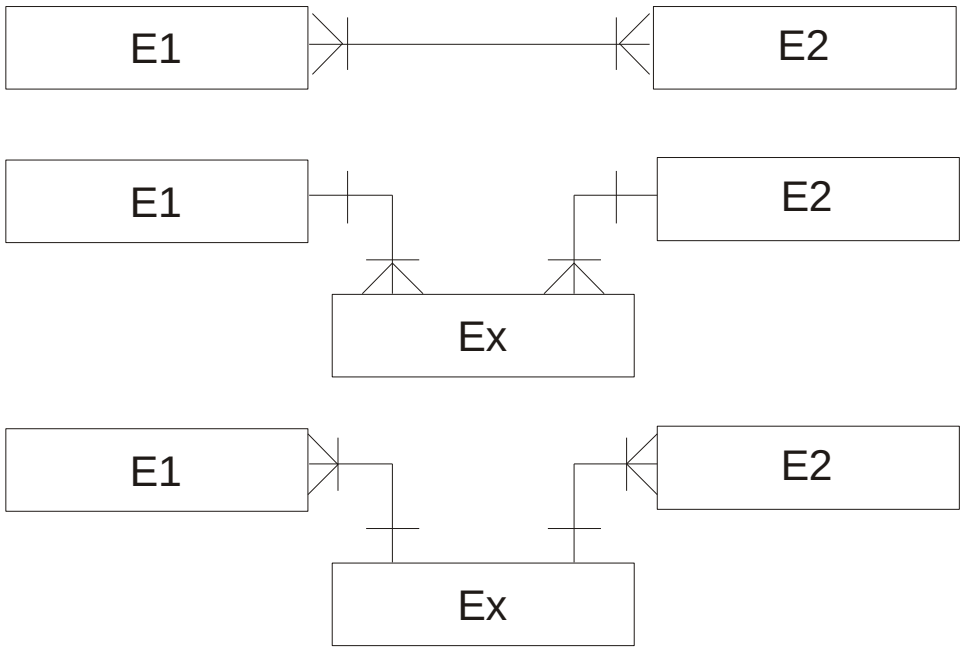
2.6.2 OSNOVNI GRADNIKI E-R MODELA

- **Kardinalnost:**
  - je udeleženost **entitete** v posamezni relaciji. Ločimo kardinalnosti:
    - 1:1 - ena-proti-ena (one-to-one)
    - 1:M - ena-proti-mnogo (one-to-many)
    - M:N - mnogo-proti-mnogo (many-to-many)

POZOR!

Kardinalnost M:N je nezaželjena, zato jo nadomestimo z dvema novima relacijama kardinalnosti 1:N in N:1 ter novo entiteto (glej sliko 9).

2.6.2 OSNOVNI GRADNIKI E-R MODELA



Slika 8: Pretvorba relacij M:N v dve relaciji 1:N in entiteto

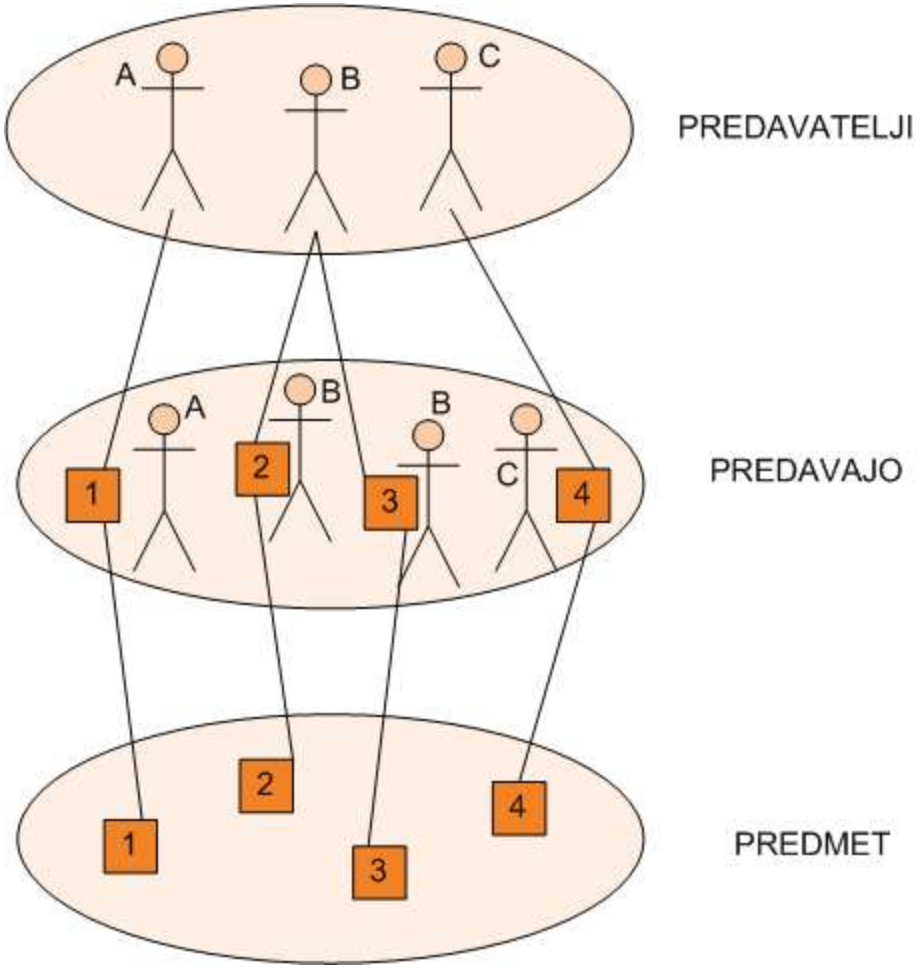
2.6.2 OSNOVNI GRADNIKI E-R MODELA

- Udeleženost entitete v relaciji je lahko opcijska (angl. optional) ali pa obvezna (angl. mandatory). Opisuje jo minimalna kardinalnost, med tem ko udeleženost v tem primeru opisuje maksimalna kardinalnost:

  (min, max)

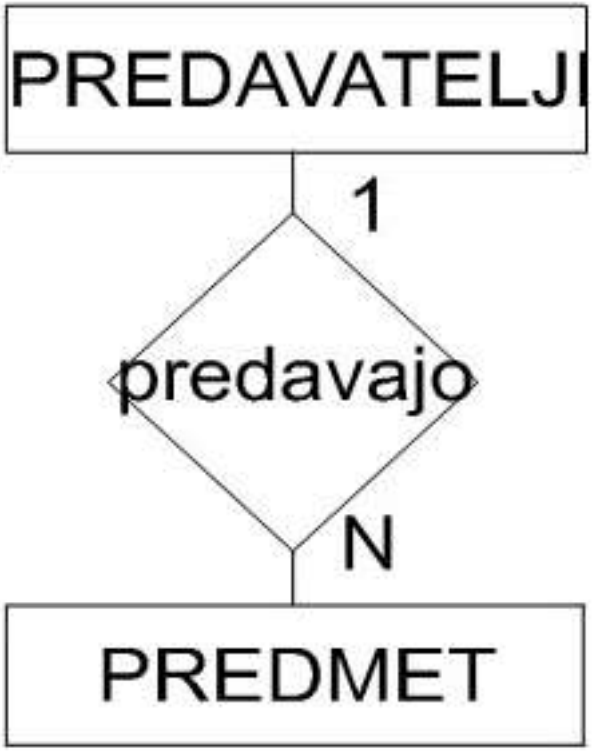
- Možne kombinacije kardinalnosti ob upoštevanju opcijske in obvezne kardinalnosti
  - 1:1=> (1, 1) : (1, 1) ali (0, 1) : (0, 1) ali (1, 1) : (0, 1) ali (0, 1) : (1, 1)
  - 1:N=>(1, N) : (1, 1) ali (0, N) : (0, 1) ali (0, N) : (1, 1) ali (1, N) : (0, 1)
  - M:N=>(1, N) : (1, M) ali (0, N) : (0, M) ali (1, N) : (0, M) ali (0, N) : (1, M)

2.6.3 NOTACIJE E-R DIAGRAMA

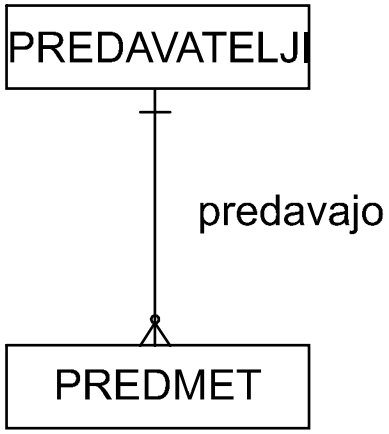


Slika 9: Primer nepraktične notacije

2.6.3 NOTACIJE E-R DIAGRAMA



Slika 10: Primer Chenove notacije



Slika 11: Primer sračje (James Martinove) notacije

2.6.3 NOTACIJE E-R DIAGRAMA

Entiteta,
entitetni tip:

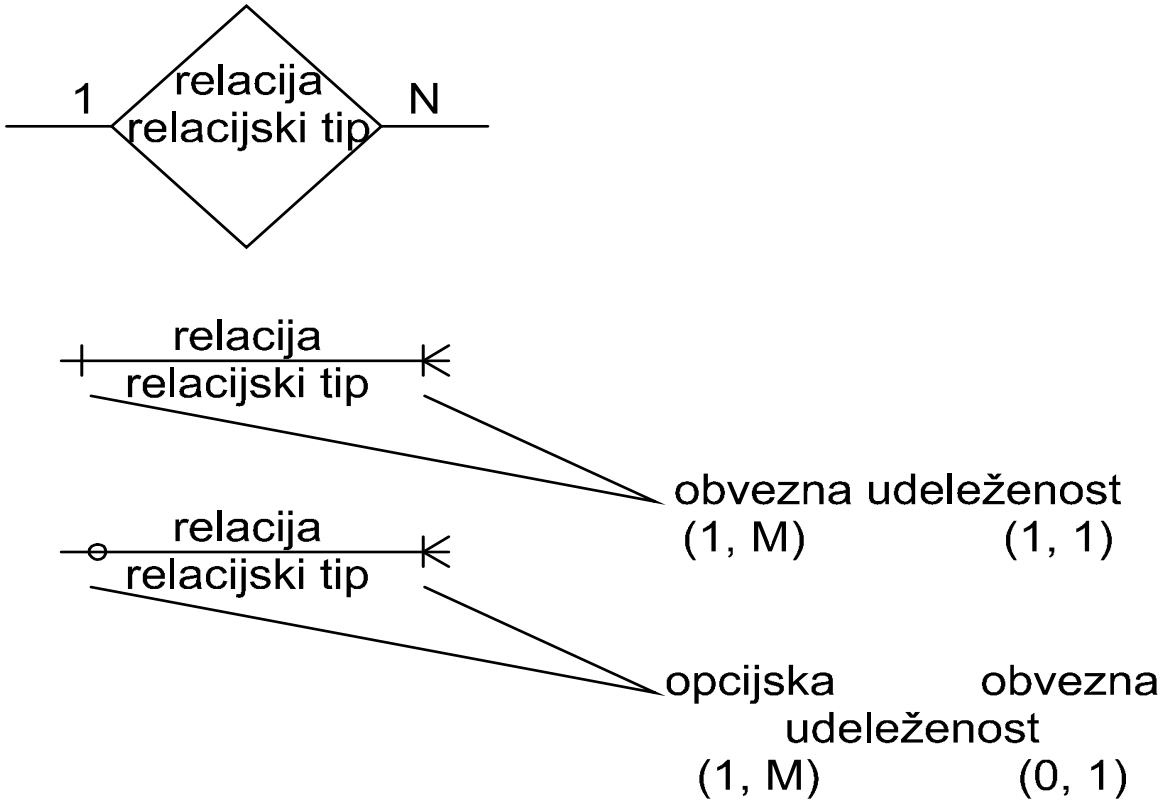
Slika 12: Notacija entitete, entitetnega tipa

Šibka entiteta:

Slika 13: Notacija šibke entitete

2.6.3 NOTACIJE E-R DIAGRAMA

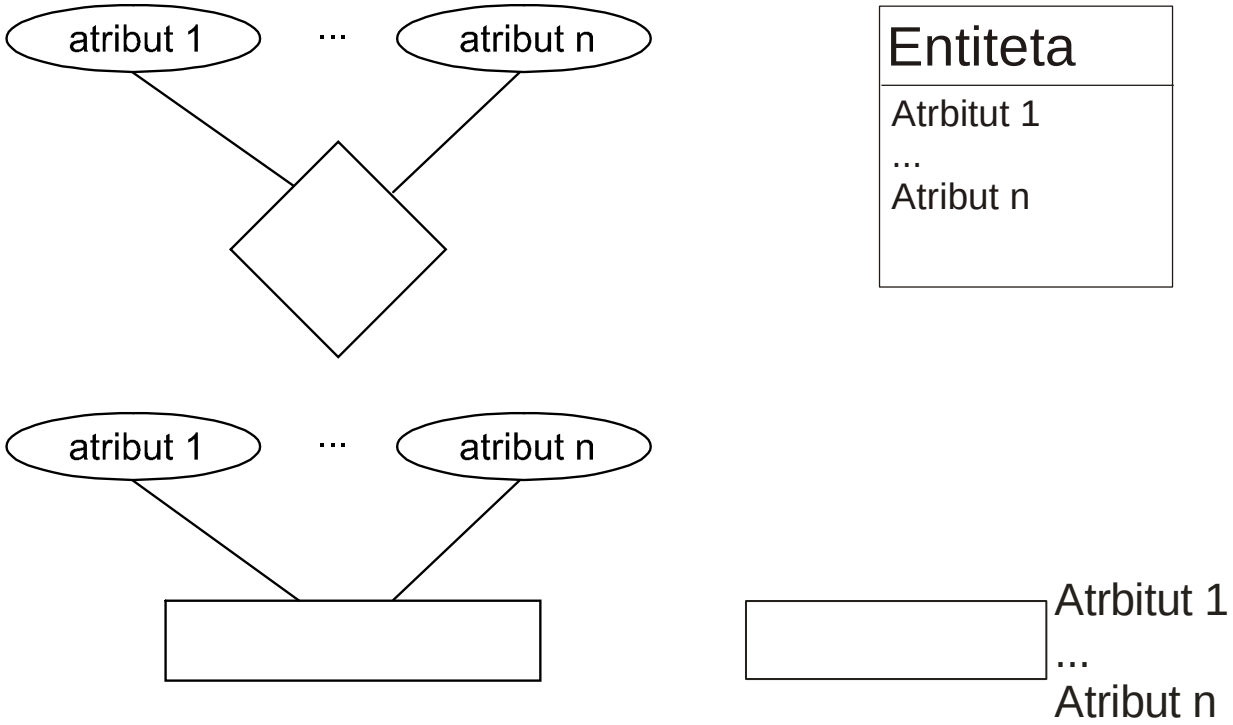
Relacijski tip s kardinalnostjo:



Slika 14: Notacija relacijskih tipov s kardinalnostjo

2.6.3 NOTACIJE E-R DIAGRAMA

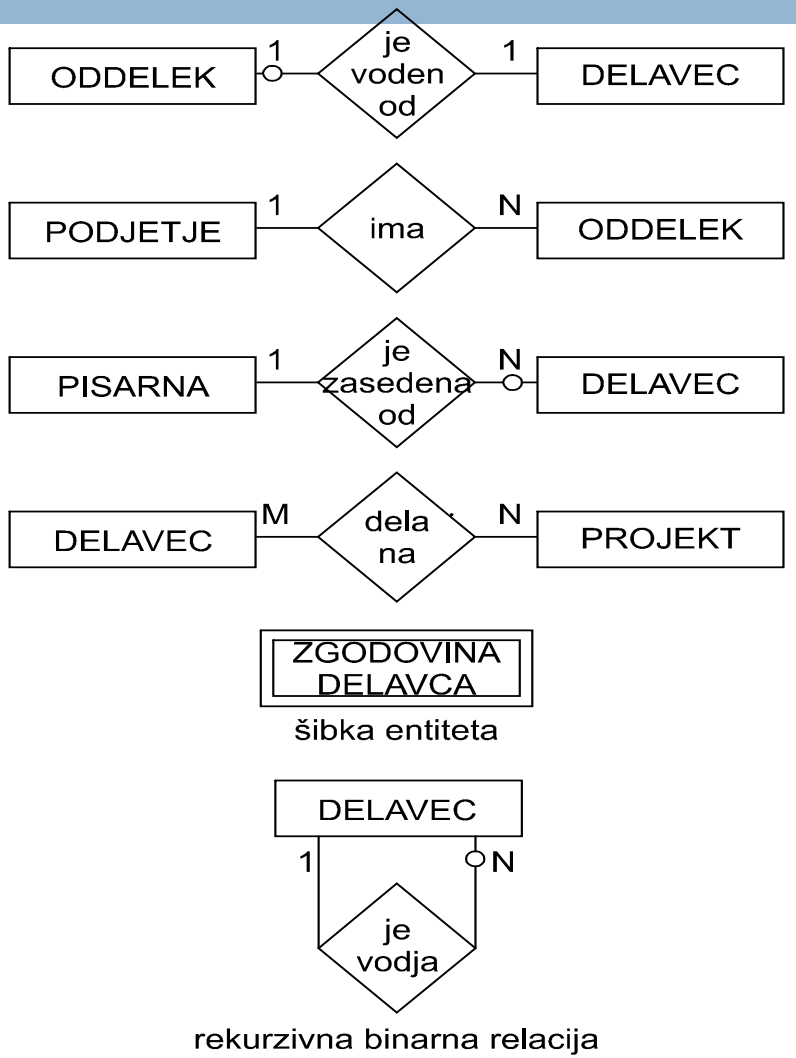
Atribut:



Slika 15: Notacija atributov

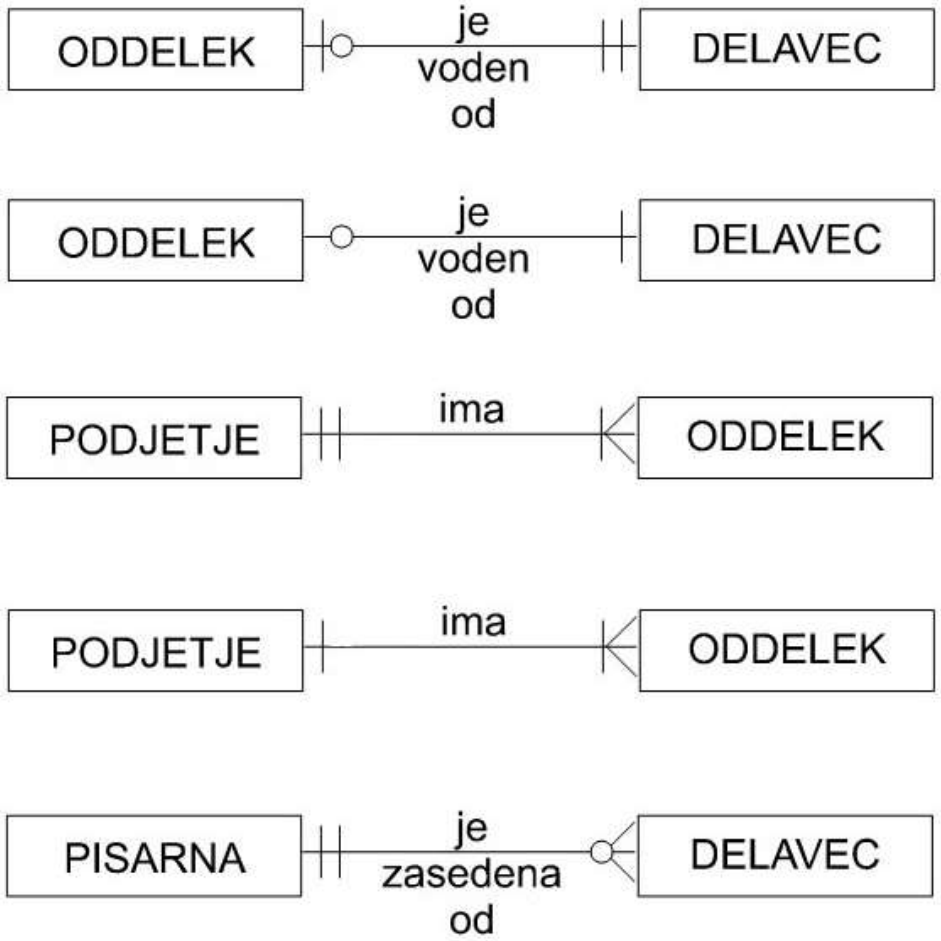
Ključ: Ključni atribut je podčrtan ali kako drugače poudarjen.

2.6.4 PRIMERJAVA RAZLIČNIH NOTACIJ E-R DIAGRAMA



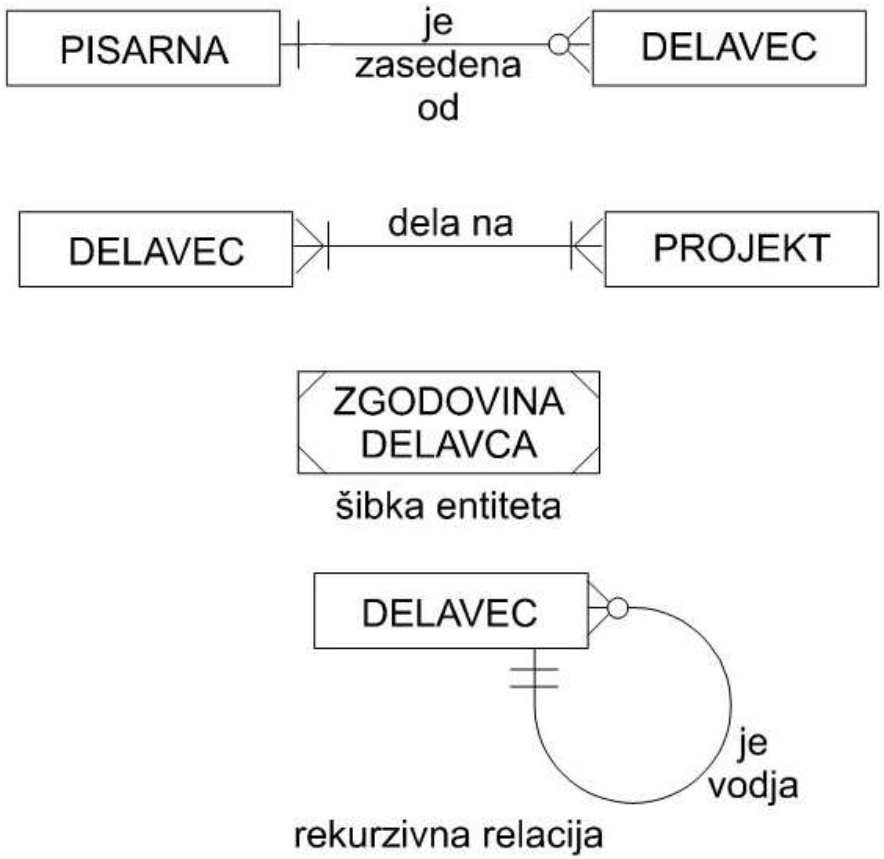
Slika 16: Chenova notacija

2.6.4. PRIMERJAVA RAZLIČNIH NOTACIJ E-R DIAGRAMA



Slika 17: Sračja (James Martinova) notacija

2.6.4. PRIMERJAVA RAZLIČNIH NOTACIJ E-R DIAGRAMA



Slika 18: Sračja (James Martinova) notacija

2.6.5 STANDARDI ZA OBLIKOVANJE E-R MODELA

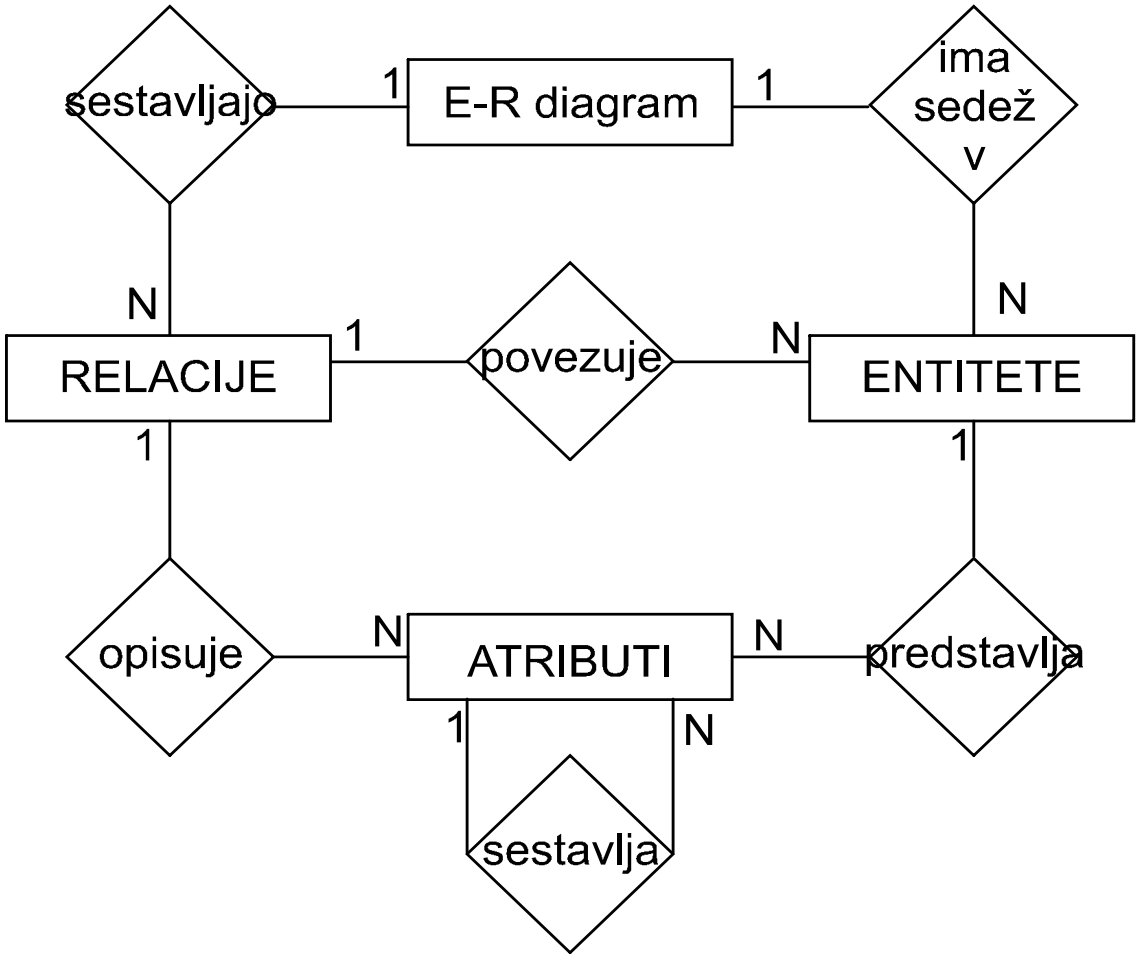
- Splošni standard za oblikovanje E-R diagrama in modela ne obstaja.

- Obstajajo avtorski in interni standardi ter priporočila.

Avtorski standard

- L. Peters:Advanced Structured Analysis and Design, Prentice Hall, 1987

2.6.5 STANDARDI ZA OBLIKOVANJE E-R MODELA



Slika 19: E-R diagram E-R modela

2.6.5 STANDARDI ZA OBLIKOVANJE E-R MODELA

- **Konceptualni standard:**
  - E-R model je konceptualna predstavitev realnega sveta in objektov zaključenega organiziranega sistema.
  - E-R model je sestavljen iz entitet in relacij.
  - Za posamezno podatkovno bazo lahko obstaja več E-R modelov.
  - E-R model vsebuje objekte, ki so pomembni za funkcije in procese zunanjega sveta.
  - E-R model običajno vsebuje informacije, ki so potrebne za celotno predstavitev realnega sveta in zaključenega organiziranega sistema, redkeje pa informacije, ki so vezane na določeno aplikacijo (aplikacije).
  - Relacije so povezave med entitetami; vsaka relacija ima eno ime in eno ali več začetnih entitet.
  - Kardinalno število relacije lahko zavzame eno od naslednjih vrednosti: eden-proti-enemu (1:1), eden-proti-mnogim (1:M) in mnogi-proti-mnogim (M:N).

2.6.5 STANDARDI ZA OBLIKOVANJE E-R MODELA

- **Notacijski standard:**
  - Vsaka entiteta je predstavljena s pravokotnikom.
  - Relacije so predstavljene z rombi ali črto, ki povezuje pravokotnika (entiteti).
  - Entitete so poimenovane s samostalniki.
  - Relacije so praviloma poimenovane z glagoli.

2.6.6 PRAKTIČNI NASVETI ZA OBLIKOVANJE E-R DIAGRAMA

Odkrivanje entitet

- Entitete so temeljni elementi zaključenega organiziranega sistema, o katerem zbiramo podatke. V okviru posameznega zaključenega organiziranega sistema so to lahko:

- **Ljudje**
  - ki so nosilci določenih funkcij (zaposleni, kupci, učitelji, študenti).
- **Predmeti**
  - ki predstavljajo posamezne fizične predmete ali skupine predmetov (naprave, orodja, produkti, stavbe).

2.6.6 PRAKTIČNI NASVETI ZA OBLIKOVANJE E-R DIAGRAMA

- **Kraji**
  - ki so v uporabi ljudi ali so v njih nameščeni predmeti (mesta, pisarne, države...).
- **Organizacije**
  - ki so formalno organizirana skupina ljudi, predmetov ali krajev, z natančno definirano nalogo. Obstoj organizacije je neodvisen od obstoja posameznih elementov te organizacije (ekipe, oddelki, podjetje).
- **Dogodki**
  - so stvari, ki se dogajajo neki entiteti v določenem trenutku (zagovor diplome, projektne faze, finančna nakazila). Pojav dogodka je vezan na trenutek, v katerem se je zgodil in na identifikator entitete, ki je v dogodek vpletena (plačilo računa: dan plačila, identifikator - številka računa).

2.6.6 PRAKTIČNI NASVETI ZA OBLIKOVANJE E-R DIAGRAMA

- **Koncepti**
  - so ideje ali principi, ki jih organizacije uporabljajo oz. imajo nadzor nad njimi (projekti, bančni računi, pritožbe).

Poimenovanje entitet

- Entitete se pojavljajo praviloma v obliki samostalnika (prodajalec) ali ustreznih izpeljav (naročilo_prodajalca). Pričakujemo, da imena niso kodirana, saj tako povedo največ o objektu tako uporabniku kot tudi načrtovalcu. Praviloma jih uporabljamo v množinski obliki (prodajalci).

2.6.6 PRAKTIČNI NASVETI ZA OBLIKOVANJE E-R DIAGRAMA

- Uporaba posameznih orodij to trditev zanika. Tudi Oracle, kjer imena **obvezno** pišemo edninsko.

POZOR!

ENTITETA *IZBRANEGA* ZAKLJUČENEGA ORGANIZIRANEGA SISTEMA NI NUJNO ENTITETA *KATEREGAKOLI DRUGEGA* ZAKLJUČENEGA ORGANIZIRANEGA SISTEMA

  - telefonske_številke (ZOS “Telekom”) <>
  - telefonske_številke (ZOS “Kadrovska služba”)

2.6.6 PRAKTIČNI NASVETI ZA OBLIKOVANJE E-R DIAGRAMA

Odkrivanje relacij

- Relacija je povezava med dvema entitetama, ki predstavlja interakcije med njima. Običajno jo lahko zapišemo v obliki preprostega stavka, ki ga sestavljajo osebek, povedek in predmet: študent obiskuje predavanje. Osebek in predmet sta pri tem entiteti, povedek pa predstavlja relacijo. Cilj odkrivanja relacij so torej stavki oblike ENTITETA 1 glagol ENTITETA 2.

Poimenovanje relacij

- Uveljavila sta se dva načina poimenovanja:
  - z uporabo glagola iz konstrukta E1 G E2
  - s kombinacijo imen obeh entitet, ki ju povezuje E1_E2

2.6.6 PRAKTIČNI NASVETI ZA OBLIKOVANJE E-R DIAGRAMA

OPOZORILO!

Vsak zapis relacije oblike E1 G E2 lahko podamo tudi v obliki E2 G E1.

Primer:

- študent obiskuje predavanja.
- predavanja so obiskovana s strani študenta

2.6.6 PRAKTIČNI NASVETI ZA OBLIKOVANJE E-R DIAGRAMA

Odkrivanje kardinalnosti

- kardinalnost podaja informacijo o udeleženosti posamezne entitete v relaciji.

- Kardinalnost je odvisna od pravil, ki vladajo v določenem zaključenem organiziranem sistemu, za katerega oblikujemo E-R diagram.

- Pri odkrivanju kardinalnosti si izberemo izhodiščno entiteto E1 v relaciji R in se vprašamo, kolikokrat (iščemo maksimalno število) se v tej relaciji glede na entiteto E pojavi entiteta E1. Nato vprašanje obrnemo.

2.6.6 PRAKTIČNI NASVETI ZA OBLIKOVANJE E-R DIAGRAMA

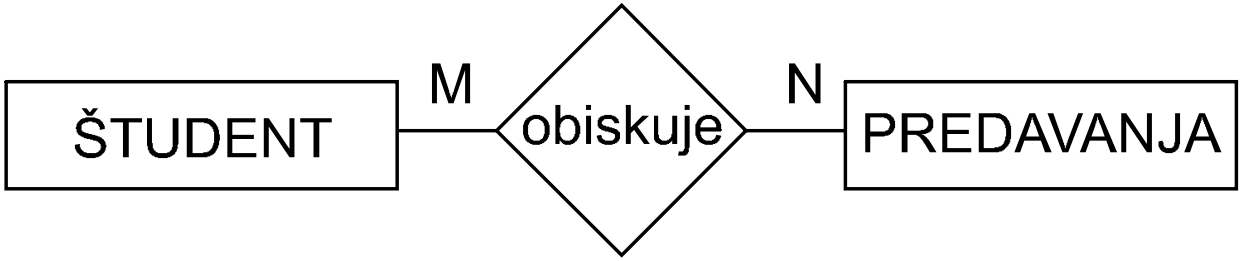
Primer:

- E1 = študent
- R = obiskuje
- E2 = predavanje

Slika 20: E-R diagram “Študent&Predavanja”

2.6.6 PRAKTIČNI NASVETI ZA OBLIKOVANJE E-R DIAGRAMA

- Koliko predavanj obiskuje študent? N
- Koliko študentov obiskuje predavanja? M



Slika 21: E-R diagram z označeno kardinalnostjo

OPOZORILO!

Za izražanje kardinalnosti večje od 1 nikoli ne uporabljamo konkretnih številk (N=10; M=36), temveč le splošen zapis M oz. N.

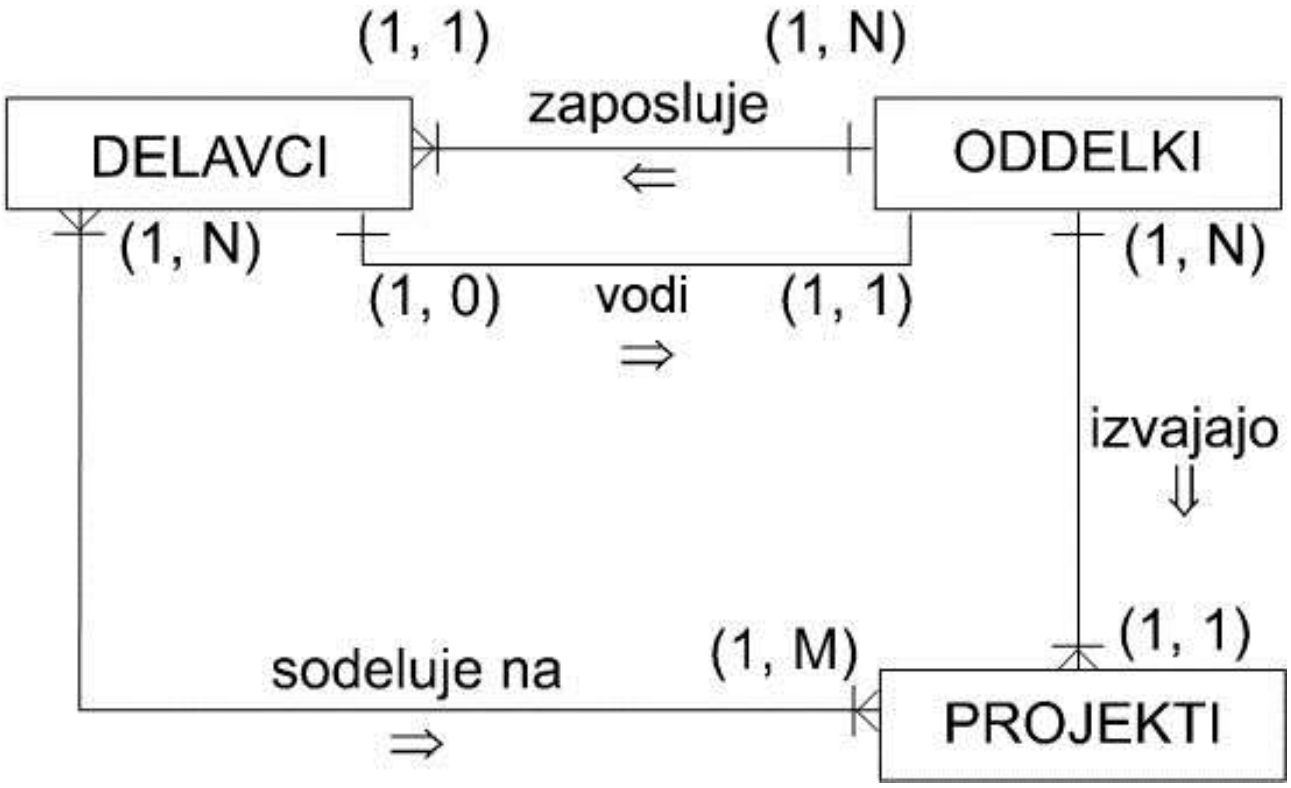
2.6.7 PRIMER OBLIKOVANJA KONCEPTUALNEGA MODELA PB

Primer:

Zaključen organiziran sistem podjetja predstavite s pomočjo E-R diagrama tako, da bo predstavljena udeležba delavcev posameznih oddelkov (ki jih vodi eden izmed njih) na posameznih projektih.

- Koraki oblikovanja konceptualnega modela:
  - analiza,
  - oblikovanje E-R diagrama:
    - entitet,
    - relacij,
    - atributov,
    - kardinalnosti.

2.6.7 PRIMER OBLIKOVANJA KONCEPTUALNEGA MODELA PB



Slika 22: Rešitev primera

2.6.7 PRIMER OBLIKOVANJA KONCEPTUALNEGA MODELA PB

DELAVEC:
matična številka
- ime
- priimek
- naslov
- rojstni datum
- OD

ODDELEK:
številka oddelka
- ime oddelka
- lokacija oddelka

PROJEKT:
številka projekta
- naziv projekta
- lokacija projekta

SODELUJE:
- št. ur

ZAPOSLUJE:
- datum zaposlitve

VODI:
- datum nastopa funkcije

IZVAJA:
- začetek projekta
- predviden zaključek projekta

3. NORMALIZACIJA

- Normalizacija je tehnika, ki omogoča oblikovanje množice entitet z želenimi lastnostmi, ki izhajajo iz podatkovnih zahtev zaključenega organiziranega sistema.

- To je proces, ki zagotavlja, da entitete ne bodo vsebovale redundantnih ali dvoumnih podatkov, ki ne bodo predmet nepravilnosti pri vnosu, brisanju in popravljanju le-teh. Normalizacijo obravnavamo kot proceduro, ki poteka od spodaj navzgor in dopolnjuje E-R model.

3. NORMALIZACIJA

- Pogosto je normalizacija predstavljena tudi kot serija testov za potrditev oz. zavrnitev normalnih oblik.

- Normalne oblike so pravila o združevanju atributov v entitete ob upoštevanju logičnih odvisnosti (funkcionalne, večvrednostne, združitveno-projekcijske in ključno-domenske odvisnosti).

3.1. FUNKCIONALNA ODVISNOST

- Opisuje odnose med atributi v entiteti. Če sta A in B atributa entitete E, je B funkcionalno odvisen od A, A $\rightarrow$ B (A funkcionalno določa B), če za vsako vrednost A-ja v E obstaja natanko ena vrednost B-ja.

- Običajno funkcionalno odvisnost definiramo med množicami atributov znotraj entitete E.

3.1. FUNKCIONALNA ODVISNOST

Formalna predstavitev funkcionalne odvisnosti

- Za R (A1, A2, ..., An) obstajata X in Y kot podmnožica množice R ( ).

  $X, Y \subseteq R$

  $X \rightarrow Y$

- Funkcionalna odvisnost X Y za dano entiteto obstaja, če za vsak par n-teric t1 in t2 velja:

  če je $t1[X] = t2[X] \Rightarrow t1[Y] = t2[Y]$

3.1. FUNKCIONALNA ODVISNOST

Popolna funkcionalna odvisnost:

- Atribut v entiteti je popolno funkcionalno odvisen, če je odvisen od celotnega ključa in ne le od dela ključa.

- Y je funkcionalno popolnoma odvisen od X, če po odstranitvi kateregakoli atributa A iz X funkcionalna odvisnost preneha obstajati.

$X \rightarrow Y;\ A \in X;\quad (X-\{A\}) \not\rightarrow Y$ popolna funkcionalna odvisnost

$X \rightarrow Y;\ A \in X;\quad (X-\{A\}) \rightarrow Y$ delna funkcionalna odvisnost

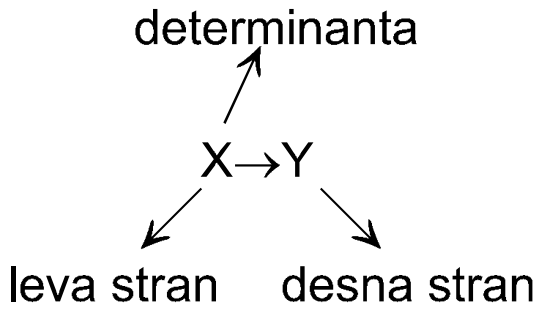
3.1. FUNKCIONALNA ODVISNOST

Primer:

- št_del, št_proj $\not\rightarrow$ št_ur
- št_del $\not\rightarrow$ št_ur popolna funkcionalna odvisnost
- št_proj $\not\rightarrow$ št_ur

- št_del, št_proj $\not\rightarrow$ priimek_del
- št_del $\rightarrow$ priimek_del delna funkcionalna odvisnost
- št_proj $\not\rightarrow$ priimek_del

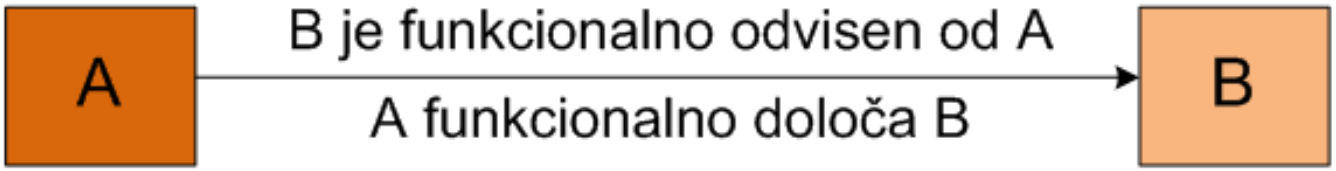
3.1. FUNKCIONALNA ODVISNOST



Slika 23: Elementi funkcionalne odvisnosti

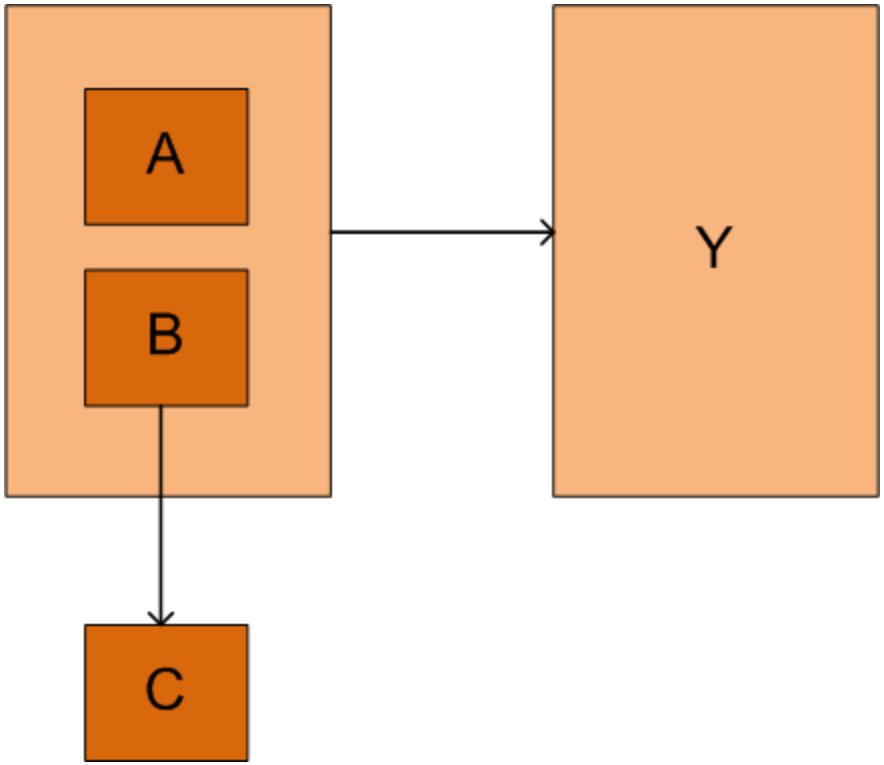
Determinanta funkcionalne odvisnosti predstavlja atribut ali skupino atributov z leve strani funkcionalne odvisnosti.

Diagram funkcionalnih odvisnosti



3.1. FUNKCIONALNA ODVISNOST

$X=\{A, B\}, X \rightarrow Y \wedge B \rightarrow C$



Slika 24: Diagram funkcionalnih odvisnosti

3.1. FUNKCIONALNA ODVISNOST

Popolna funkcionalna odvisnost

- Y je funkcionalno popolnoma odvisen od X, če za pravo podmnožico $X_1 (X_1 \subset X)$ velja, da funkcionalno ne določa Y.

$$X \rightarrow Y$$

$$X_1 \subset X, X_1 \not\rightarrow Y$$

3.1. FUNKCIONALNA ODVISNOST

Delna funkcionalna odvisnost

- Y je funkcionalno delno (parcialno) odvisen od X, če za pravo podmnožico $X_1 (X_1 \subset X)$ velja, da funkcionalno določa Y.

$\rightarrow$

X Y $\rightarrow$

$X_1 \subset X$, X_1 Y

3.1.1. LASTNOSTI FUNKCIONALNE ODVISNOSTI

Osnovne lastnosti

Enoličnost

Za dano domeno in kodomeno obstaja največ ena funkcionalna odvisnost.

$f: X \rightarrow Y \wedge g: X \rightarrow Y \Rightarrow f=g$

Projektivnost

Množica funkcionalno določa vse svoje podmnožice

$X \subseteq Y \Rightarrow Y \rightarrow X$

3.1.1. LASTNOSTI FUNKCIONALNE ODVISNOSTI

Aditivnost

$X \rightarrow Y,\ X \rightarrow Z \Rightarrow X \rightarrow YZ$

Tranzitivnost

$X \rightarrow Y \wedge Y \rightarrow Z \Rightarrow X \rightarrow Z$

Funkcionalna odvisnost $X \rightarrow Z$ je tranzitivna, če obstaja množica atributov Y (ki niso podmnožica ključa relacije R) in veljata odvisnosti $X \rightarrow Y$ in $Y \rightarrow Z$.

3.1.1. LASTNOSTI FUNKCIONALNE ODVISNOSTI

Izpeljane lastnosti

Distributivnost

$X \rightarrow YZ \Rightarrow X \rightarrow Y$ in $X \rightarrow Z$

$X \rightarrow YZ \Rightarrow YZ \rightarrow Y \wedge YZ \rightarrow Z \Rightarrow$

$X \rightarrow YZ \wedge YZ \rightarrow Y \Rightarrow X \rightarrow Y$

$X \rightarrow YZ \wedge YZ \rightarrow Z \Rightarrow X \rightarrow Z$

3.1.1. LASTNOSTI FUNKCIONALNE ODVISNOSTI

Pseudotranzitivnost

$X \rightarrow Y \wedge YW \rightarrow Z \Rightarrow XW \rightarrow Z$

$X \rightarrow Y \wedge W \rightarrow W \Rightarrow XW \rightarrow YW$

$XW \rightarrow YW \wedge YW \rightarrow Z \Rightarrow XW \rightarrow Z$

3.1.1. LASTNOSTI FUNKCIONALNE ODVISNOSTI

Razširitev

$X \rightarrow Y \wedge W \rightarrow Z \Rightarrow XW \rightarrow YZ$

$XW \rightarrow W \wedge W \rightarrow Z \Rightarrow XW \rightarrow Z$

$XW \rightarrow X \wedge X \rightarrow Y \Rightarrow XW \rightarrow Y$

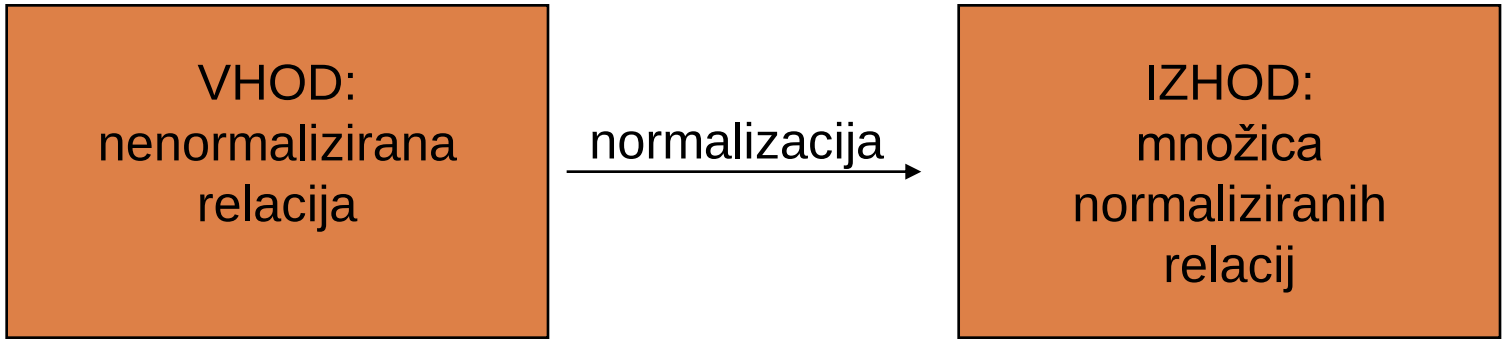
$XW \rightarrow Z \wedge XW \rightarrow Y \Rightarrow XW \rightarrow YZ$

3.2. NORMALNE OBLIKE

Each attribute is placed in an entity where it is dependent on the key, the whole key and nothing but the key. So help me Codd!

C Finkelstein

- Normalizacija: proces dekompozicije enitet (relacij)

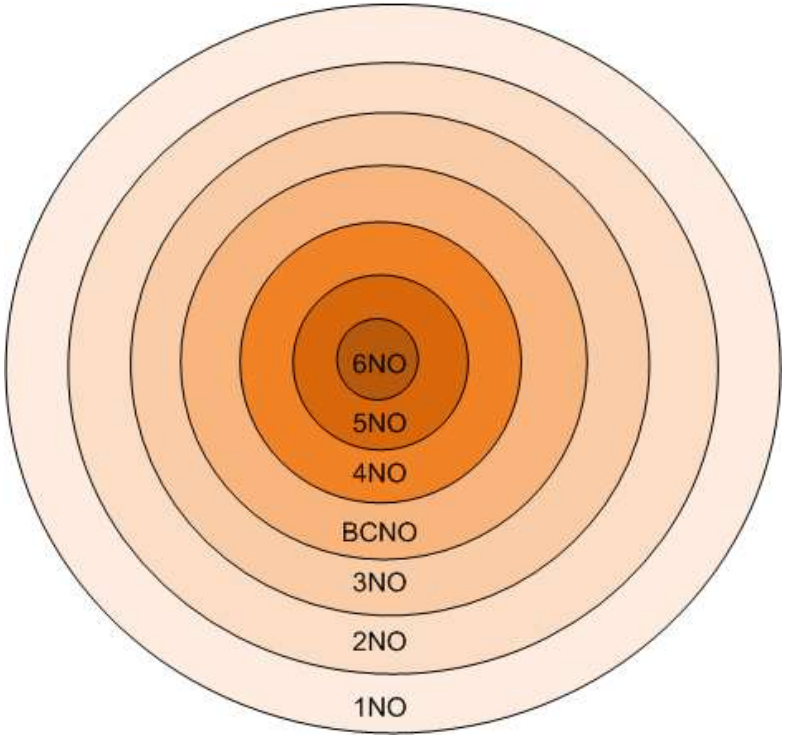


Slika 25: Proces normalizacije

3.2. NORMALNE OBLIKE

- **Normalne oblike:**
  - pravila o grupiranju atributov v entitete ob upoštevanju logičnih odvisnosti (funkcionalnega, večpomenskega, projekcijsko združitvenega in ključnega tipa).
- **Mejniki:**
  - 1972, 1974, 1977, 1979
- **Avtorji:**
  - E.F. Codd, R. Boyce, R. Fagin
- **Cilj:**
  - odprava nepravilnosti pri vnosu, brisanju in popravljanju podatkov

3.2. NORMALNE OBLIKE



Slika 26: Hierarhija normalnih oblik

Če entiteta izpolnjuje pogoje n-te normalne oblike (NO), izpolnjuje tudi pogoje normalnih oblik od 1 do n, ne pa pogojev normalnih oblik od n do 6.

3.2.1 FORMALNA DEFINICIJA NORMALNIH OBLIK

- **PRVA NORMALNA OBLIKA:**
  - Entiteta E(A1, A2, …, An) je v prvi normalni obliki, če in samo če so vrednosti v domenah osnovne za vsak atribut A_i v entiteti E(A1, A2, …, An).

3.2.1 FORMALNA DEFINICIJA NORMALNIH OBLIK

- **DRUGA NORMALNA OBLIKA:**
  - Entiteta E(A1, A2, …, An) je v drugi normalni obliki, če in samo če je v prvi normalni obliki in je vsak neključen atribut popolno funkcionalno odvisen od primarnega ključa entitete E(A1, A2, …, An).
- **OPOMBA:**
  - Za dano relacijo $R(A_1, A_2, ..., A_n)$ in podmnožici atributov X in Y je R[XY] projekcija relacije $R(A_1, A_2, ..., A_n)$ po atributih $X \cup Y$. Funkcionalna odvisnost X $\Rightarrow$ Y (Y je funkcionalno odvisen od X, oz. X funkcionalno določa Y) obstaja, če in samo če je v vsakem trenutku relacija R[XY] v bistvu funkcija R[X] $\Rightarrow$ R[Y]. To pomeni, da kadarkoli sta xy in xy' elementa relacije R[XY] velja, da je y=y'. To je pogoj, ki zagotavlja, da je relacija R[XY] funkcija R[X] $\rightarrow$ R[Y].

3.2.1 FORMALNA DEFINICIJA NORMALNIH OBLIK

- **TRETJA NORMALNA OBLIKA:**
  - Entiteta E(A1, A2, …, An) je v tretji normalni obliki, če in samo če je v drugi normalni obliki in nobeden od njenih neključnih atributov ni tranzitivno odvisen od ključa entitete.

- **BOYCE-CODDOVA NORMALNA OBLIKA:**
  - Entiteta E(A1, A2, …, An) je v Boyce-Coddovi normalni obliki, če in samo če je v tretji normalni obliki in je vsaka determinanta ključ.
- **OPOMBA:**
  - Determinanta je atribut ali množica atributov, ki funkcionalno popolnoma določa nekatere atribute (popolna FO).

3.2.1 FORMALNA DEFINICIJA NORMALNIH OBLIK

- **<u>ČETRTA NORMALNA OBLIKA:</u>**
  - Entiteta E(A1, A2, …, An) je v četrti normalni obliki, če in samo če je v BC normalni obliki in ne vsebuje večvrednostnih odvisnosti.
- **OPOMBA:**
  - Večvrednostna odvisnost v entiteti E(A1, A2, A3) obstaja, če obstaja za vsak atribut A1 množica atributov A2 in A3. Množici atributov A2 in A3 sta medsebojno neodvisni.

3.2.1 FORMALNA DEFINICIJA NORMALNIH OBLIK

- **PETA NORMALNA OBLIKA:**
  - Entiteta E(A1, A2, …, An) je v peti normalni obliki, če in samo če je v četrti normalni obliki in ne vsebuje projekcijsko združitvene odvisnosti, ki ni posledica kandidacijskega ključa.

- **ŠESTA NORMALNA OBLIKA:**
  - Entiteta E(A1, A2, …, An) je v šesti normalni obliki, če in samo če je v peti normalni obliki in ne obstaja ključna odvisnost.

3.2.2. UPORABNA DEFINICIJA NORMAILNIH OBLIK

- **PRVA NORMALNA OBLIKA:**
  - Pri normalizaciji v prvo normalno obliko poiščemo in izločimo ponavljajoče skupine atributov. Izločimo jih v novo entiteto. Primarni ključ tako oblikovane entitete je sestavljen iz primarnega ključa nenormalizirane entitete in ključa, ki pripada ponavljajoči se skupini atributov. Kot primarni ključ torej izberemo atribut, ki izpolnjuje uporabnikove potrebe in zahteve.
- **OPOMBA:**
  - Ponavljajoča skupina atributov je zbirka logično povezanih atributov, ki se večkrat pojavijo v okviru dane entitete.

3.2.2. UPORABNA DEFINICIJA NORMAILNIH OBLIK

- **DRUGA NORMALNA OBLIKA:**
  - V novo entiteto prenesemo atribute, ki so le delno funkcionalno odvisni od primarnega ključa, ali pa so odvisni le od dela sestavljenega primarnega ključa in enega ali več drugih ključnih atributov.

- **TRETJA NORMALNA OBLIKA:**
  - Iz obstoječe entitete prenesemo v novo entiteto tiste atribute, ki so odvisni od neključnega atributa.

3.2.2. UPORABNA DEFINICIJA NORMAILNIH OBLIK

- **ČETRTA NORMALNA OBLIKA:**
  - Entiteta izpolnjuje pogoje četrte NO, če
  - 1. Izpolnjuje pogoje 3NO in atributi niso odvisni le od ključa, temveč tudi od njegove vrednosti
    ALI
  - 2. Če prenesemo atribut iz ene entitete v drugo tako, da je le-ta popolnoma funkcionalno odvisen od ključa druge relacije.
- **PETA NORMALNA OBLIKA:**
  - Relacija je v peti normalni obliki, če smo v relacijo prenesli večkratno pojavnost iste relacije.

3.3 PRIMER: PREDMETNIK

Nenormalizirana entiteta:

PREDMETNIK							
MatičnaŠt Študenta	Priimek Študenta	Smer	Šifra Predmeta	NazivPredmeta	Nosilec Predmeta	Številka Kabineta	Težavnostna Stopnja
38214	Kos	INF	I350	Baze podatkov	Date	B104	A
38214	Kos	INF	I465	Sistemska analiza	DeMarco	B213	C
9173	Lev	PRO	I465	Sistemska analiza	DeMarco	B213	A
			P300	Programski jeziki	Wirth	B317	B
			P440	Operacijski sistemi	Hansen	B215	C

Vzroki:

- podatki niso osnovni,
- prisotnost ponavljajoče skupine,
- redundanca podatkov,
- problem kandidacijskega ključa.

3.3.1 NORMALIZACIJA V PRVO NORMALNO OBLIKO

Recept: izloči ponavljajoče skupine.

ŠTUDENT		
MatičnaŠtevilkaŠtudenta #	PriimekŠtudenta	Smer
38214	Kos	INF
69173	Lev	PRO

ŠTUDENT_PREDMETNIK					
MatičnaŠtevilkaŠtudenta #	Šifra Predmeta #	NazivPredmeta	Nosilec Predmeta	Številka Kabineta	TežavnostnaStopnja
38214	1350	Baze podatkov	Date	B104	A
38214	I465	Sistemska analiza	DeMarco	B213	C
69173	1465	Sistemska analiza	DeMarco	B213	A
69173	P300	Programski jeziki	Wirth	B317	B
69173	P440	Operacijski sistemi	Hansen	B215	C

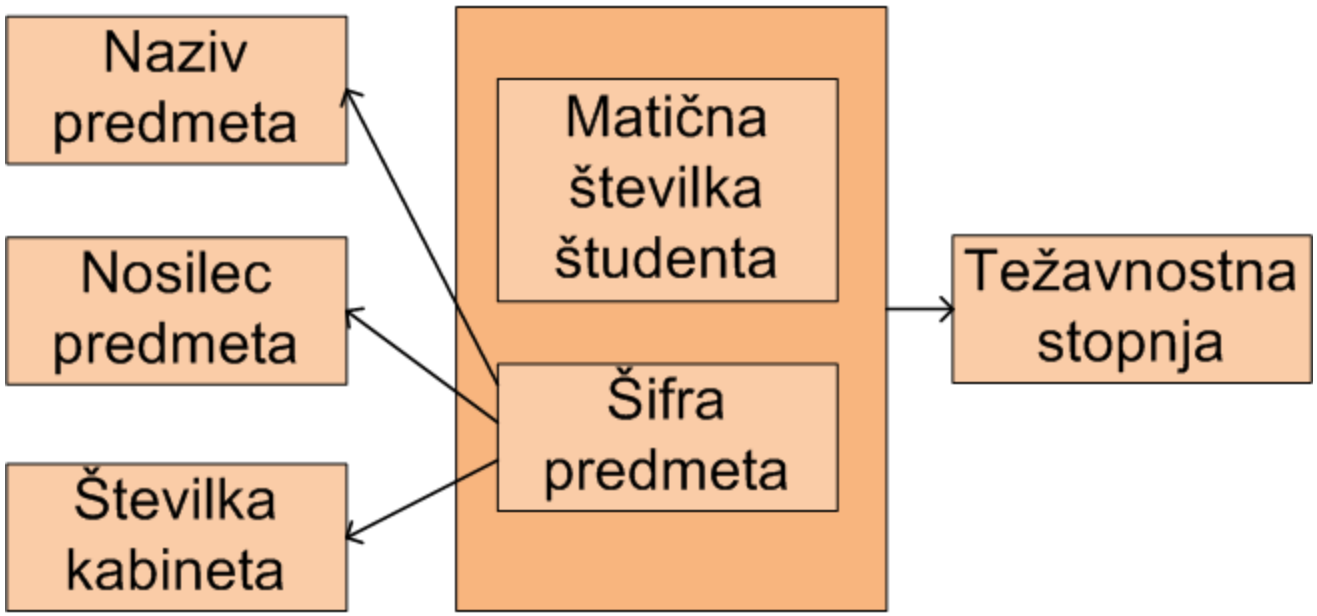
3.3.1 NORMALIZACIJA V PRVO NORMALNO OBLIKO

- Komentar:
  - Entiteta ŠTUDENT je v 3 NO.
  - Entiteta ŠTUDENT_PREDMETNIK ima sestavljen ključ.

- Nepravilnosti:
  - vnos - nov predmet v predmetniku,
  - brisanje - študent zapusti šolo,
  - popravljanje - sprememba naziva predmeta.

3.3.1 NORMALIZACIJA V PRVO NORMALNO OBLIKO

- Vzrok za nepravilnosti:
  - neključni atributi so odvisni le od dela primarnega ključa.
- Diagram funkcionalnih odvisnosti:



3.3.1 NORMALIZACIJA V PRVO NORMALNO OBLIKO

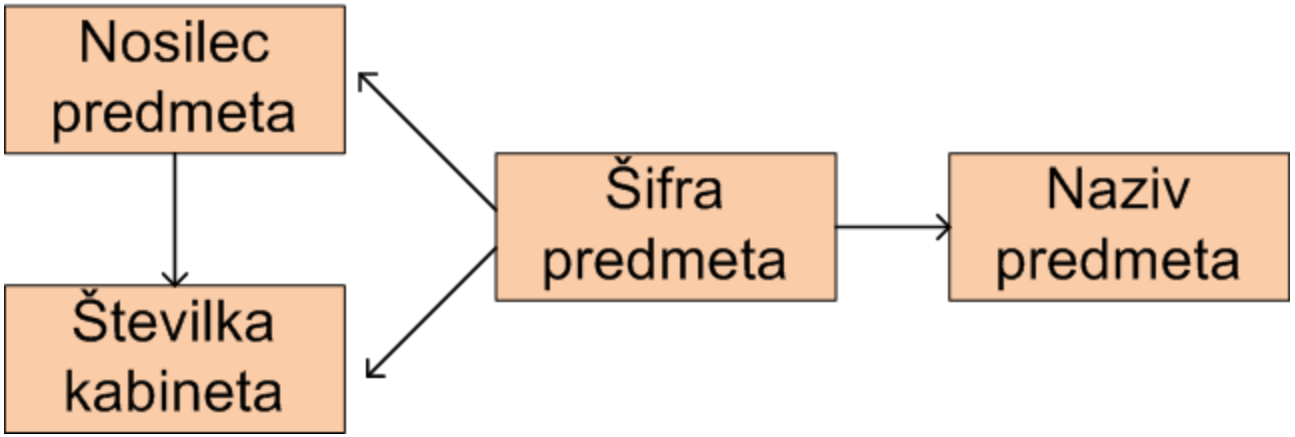
- Recept: izloči delne funkcionalne odvisnosti.

VPIS		
MatičnaŠtevilkaŠtudenta #	Šifra Predmeta #	TežavnostnaStopnja
38214	1350	A
38214	*1465*	*C*
69173	1465	*A*
69173	*P300*	B
69173	P440	*C*

PREDMET-NOSILEC PREDMETA			
Šifra Predmeta #	Naziv Predmeta .	Nosilec Predmeta	Številka Kabineta
I350	Baze podatkov	Date	B104
I465	Sistemska analiza	DeMarco	B213
I465	Sistemska analiza	DeMarco	B213
P300	Programski jeziki	Wirth	B317
P440	Operacijski sistemi	Hansen	B215

3.3.2 NORMALIZACIJA V DRUGO NORMALNO OBLIKO

- Komentar:
  - Entita VPIS je v 3 NO.
- Nepravilnosti:
  - vnos - nov predavatelj,
  - brisanje - sprememba v predmetniku,
  - popravljanje - selitev nosilca predmeta.
- Diagram funkcionalnih odvisnosti:



3.3.2 NORMALIZACIJA V DRUGO NORMALNO OBLIKO

- Recept: izloči tranzitivne odvisnosti.

PREDMET		
Šifra Predmeta #	NazivPredmeta	NosilccPredmeta
I350	Baze podatkov	Date
1465	Sistemska analiza	DeMarco
P300	Programski jeziki	Wirth
P440	Operacijski sistemi	Hansen

PREDAVATELJ	
NosilccPredmeta #	Številka Kabineta
Date	B 104
DeMarco	B213
Wirth	B317
Hansen	B215

Komentar: prvotna relacija PREDMETNIK je normalizirano v 3 NO.

4. LOGIČNO MODELIRANJE

- Logično modeliranje je proces oblikovanja logičnega modela za informacijsko uporabo v zaključenem organiziranem sistemu, pri čemer je model vezan na enega izmed podatkovnih modelov, toda neodvisen od SUPB in drugih fizičnih vidikov.

- Logični podatkovni model je, kot rezultat logičnega modeliranja, model namenjen informacijski uporabi v zaključenem organiziranem sistemu in temelji na izbranem podatkovnem modelu ciljnega SUPB. V fazi oblikovanja logičnega modela preverjamo model tudi na uporabniške zahteve. Pri tem uporabljamo proces normalizacije za preverjanje korektnosti logičnega modela.

4. LOGIČNO MODELIRANJE

Relacijski podatkovni model

- Rezultat razočaranj nad obstoječimi podatkovnimi modeli.
- Oblikovan je na osnovi matematične strukture - teorije o relacijah in predikatnega računa prvega reda.
- Osnovni gradnik predstavlja tabela oz. relacija.

- **Prednosti:**
  - preprostost,
  - stopnja podatkovne neodvisnosti je veliko večja,
  - uporabnik je neodvisen od poznavanja fizične strukture,
  - dostop do podatkov je neomejen glede na število različnih pristopov,
  - dostop do podatkov je enostaven,
  - vgrajena je večnivojska podatkovna integriteta,

4. LOGIČNO MODELIRANJE

- **Slabosti:**
  - programski produkti, ki delujejo v povezavi z relacijskim podatkovnim modelom, so počasni,
  - do nedavnega sta bila mrežni in drevesni podatkovni model bolj razširjena.

4.1. PREHOD IZ E-R MODELA V LOGIČNI PODATKOVNI MODEL

- Konceptualni model predstavlja vhod v logično modeliranje, ki rezultira v logičnem podatkovnem modelu.
- Prehod iz E-R modela v logični podatkovni model poteka v dveh korakih:
  - **Prehod, neodvisen od sistema**
    - V tem koraku izvedemo neodvisno pretvorbo E-R modela v izbran podatkovni model. Za posamezen podatkovni model obstaja algoritem, ki nam olajša prehod. Še večjo podporo pa nudijo orodja, ki omogočajo avtomatsko pretvorbo, kar pomeni, da imajo vgrajen algoritem za prehod, neodvisen od sistema.
  - **Prilagoditev logičnega podatkovnega modela specifičnemu SUPB**
    - Prilagoditev rezultirajočih modelov iz predhodnega koraka na posebne lastnosti in omejitve SUPB.

4.2. RELACIJSKI PODATKOVNI MODEL

Ozadje

- **Šestdeseta leta:**
  - Edgar F. Codd - raziskovalec IBM-ovega razvojnega laboratorija v San Jose raziskuje podatkovne baze z velikim številom podatkov.
  - Uvedba discipline in strukture iz področja matematike.
  - Junij 1970 - Communications of the ACM, pp. 377-387: A Relational Model of Data for Large Shared Databanks - prva predstavitev RDM (Relational Data Model).
  - Formalna (teoretična osnova):
    - teorija o relacijah (teorija množic),
    - predikatni račun prvega red.

4.2. RELACIJSKI PODATKOVNI MODEL

- **Sedemdeseta leta:**
  - Razvoj prototipnega IBM-ovega SUPB System R.
  - SUPB INGRES (Interactive Graphics Retrival System) projekt univerze Berkeley, Kalifornija.
  - Projekt Peterlee Relational Test Vehicle IBM-ovega znanstvenega centra v Peterlee-ju, Velika Britanija se je ukvarjal s teoretičnimi problemi, povezanimi z povpraševalnimi jeziki in njihovo optimizacijo.

- **Osemdeseta leta** - razvoj komercialnih SUPB:
  - Veliki sistemi: DB2, SQL/DS, ORACLE.
  - Mali sistemi: Paradox, dBase IV, Access, FoxPro.

4.2.1. TERMINOLOGIJA IN OSNOVNA STRUKTURA

- **Relacija:**
  - Je dvodimenzionalna tabela s stolpci in vrsticami.
    - Relacija (matematični pojem).
    - Tabela (fizični pojem).
- **Atribut:**
  - predstavlja ime stolpca relacije.
- **Domena:**
  - je množica dopustnih vrednosti za en ali več atributov.
- **N-terica (angl. tuple):**
  - je vrstica relacije, ki predstavlja posamezen zapis oz. podaja podatke za posamezen objekt (entiteto) zaključenega organiziranega sveta.

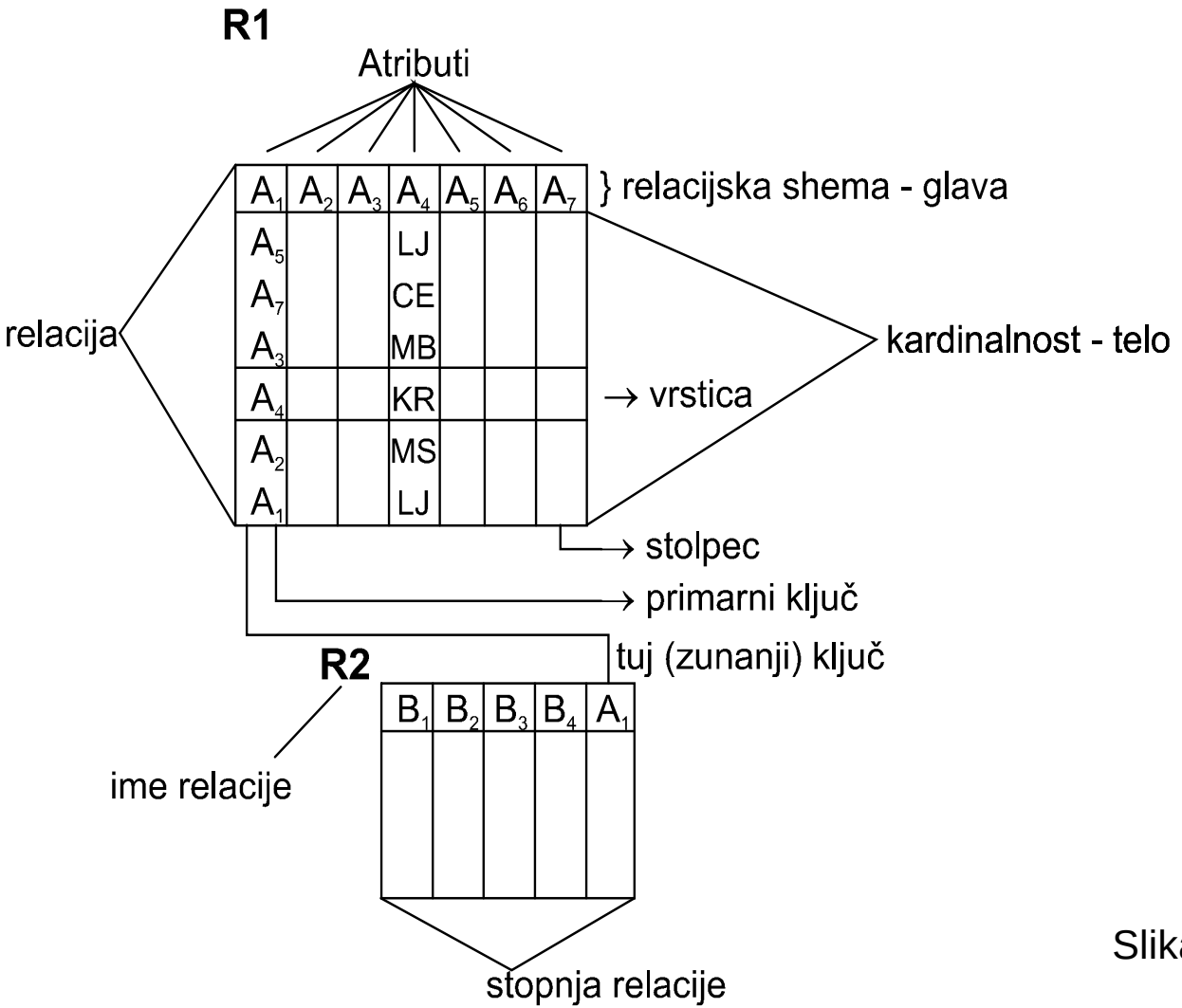
4.2.1. TERMINOLOGIJA IN OSNOVNA STRUKTURA

- **Stopnja relacije:**
  - je predstavljena s številom atributov relacije (unarna, binarna, trinarna, ..., n-arna /n-ary/ relacija)

- **Kardinalnost relacije:**
  - je predstavljena s številom vrstic, ki jih relacija vsebuje.

- **Relacijska shema:**
  - je ime relacije, ki mu sledi množica atributov: R (A1, A2, ..., An) oz. ime relacije, ki mu sledi množica parov, sestavljenih iz atributov in domen.
  - A1, A2, ..., An naj bodo atributi z domenami D1, D2, ..., Dn. Potem je množica {A1:D1, A2:D2,..., An:Dn} relacijska shema relacije R.

4.2.1. TERMINOLOGIJA IN OSNOVNA STRUKTURA



Slika 27: Predstavitev relacije

4.2.1. TERMINOLOGIJA IN OSNOVNA STRUKTURA

- **Ključ:**
  - je atribut ali množica atributov, ki unikatno določajo n-terico znotraj relacije.
- **Kandidacijski ključ:**
  - je ključ, katerega nobena prava podmnožica ni nadključ relacije.
- **Sestavljen ključ:**
  - je ključ, ki ga sestavlja več kot en atribut.

4.2.1. TERMINOLOGIJA IN OSNOVNA STRUKTURA

- **Alternativni ključ:**
  - je kandidacijski ključ, ki ni bil izbran za primarni ključ.
- **Tuji - zunanji ključ:**
  - je atribut ali skupina atributov znotraj relacije, ki je primarni ključ druge relacije (izhodiščne relacije).
- **Relacijska podatkovna baza:**
  - je zbirka normaliziranih relacij.

4.2.1.1. MATEMATIČNA PREDSTAVITEV RELACIJE

- Za dano množico domen D1, D2, ..., Dn je kartezični produkt

  definiran kot:

$$D1 \times D2 \times D3 \times \ldots \times Dn =$$

$$\{(d1, d2, d3, \ldots, Dn) \mid d1 \in D1, d2 \in D2, \ldots, dn \in Dn\}$$

- Katerakoli množica n-teric iz tega kartezičnega produkta predstavlja relacijo nad množico n domen.

4.2.2. LASTNOSTI RELACIJ

- Imena relacij so unikatna.
- Vrednosti atributov so enostavne.
- Imena atributov so unikatna znotraj relacije.
- Vrednosti posameznega atributa imajo isto domeno.
- Vrstni red atributov ni pomemben.
- Vse vrstice se med seboj razlikujejo; v tabeli ni ponavljajočih se vrstic.
- Vrstni red vrstic v relaciji je nepomemben.

4.2.2. LASTNOSTI RELACIJ

- Nekatere izmed trditev izhajajo iz lastnosti matematične relacije:
  - relacija je množica; vrstni red elementov v množici ni pomemben (n-terice, atributi dokler ni postavljena struktura relacije),
  - v množici ni ponavljajočih se elementov (n-terica),
  - v relaciji so vrednosti za posamezno pozicijo določene z množico oz. domeno, v tabeli pa pričakujemo vrednosti posameznega stolpca iz pripadajoče domene.

4.2.2. LASTNOSTI RELACIJ

- **Entitetna celovitost**
  - V relaciji primarni ključ ne more imeti vrednosti *null*.

- **Referenčna celovitost**
  - Če v relaciji obstaja tuji (zunanji) ključ, mora biti njegova vrednost identična vrednosti v njegovi izhodiščni relaciji, ali pa ima vrednost *null*.

NULL - vrednost atributa, ki je trenutno neznana.

- **Omejitve zaključenega organiziranega sistema**
  - Omejitve, ki jih definirajo uporabniki ali administrator(ji) podatkovne baze.

4.3. PRETVORBA IZ E-R MODELA V RELACIJSKI PODATKOVNI MODEL

1. KORAK

- Vsako entiteto E iz E-R modela prevedemo v relacijo R:

$$E\ (A1, A2, ..., An) \rightarrow R\ (A1, A2, ..., An)$$

4.3. PRETVORBA IZ E-R MODELA V RELACIJSKI PODATKOVNI MODEL

2. KORAK

- Za vsako relacijo kardinalnosti 1:1 določimo pripadajoči relaciji (tabeli) S in T, ki pripadata entitetama, ki ju relacija povezuje. Eni izmed relacij (tabel) dodamo tuji ključ (primarni ključ druge relacije) Relaciji s tujim ključem dodamo tudi morebitne atribute relacije.

E1 (E11, E12, ..., E1n) $\rightarrow$ S (S1, S2, ..., Sn)

E2 (E21, E22, ..., E2m) $\rightarrow$ T (T1, T2, ..., Tm)

Re (A1, A2, ..., An) $\rightarrow$

S (S1, S2, ..., Sn, T1, A1, A2, ..., An)

ali

T (T1, T2, ..., Tm, S1, A1, A2, ..., An)

4.3. PRETVORBA IZ E-R MODELA V RELACIJSKI PODATKOVNI MODEL

3. KORAK

- Za vsako relacijo kardinalnosti 1:N določimo pripadajoči relaciji (tabeli) S in T, pri čemer prevedemo v relacijo S entiteto na strani kardinalnosti 1, v relacijo T pa entiteto na strani N. V relacijo T dodamo primarni ključ relacije S in morebitne atribute relacije, ki povezuje entiteti E1 in E2.

$$E1\ (E11, E12, ..., E1j) \rightarrow S\ (S1, S2, ..., Sj)$$

$$En\ (En1, En2, ..., Enn) \rightarrow T\ (T1, T2, ..., Tn)$$

$$Re\ (A1, A2, ..., An) \rightarrow T\ (T1, T2, ..., Tn, S1, A1, A2, ..., Aj\ )$$

4.3. PRETVORBA IZ E-R MODELA V RELACIJSKI PODATKOVNI MODEL

4. KORAK

- Za vsako relacijo kardinalnosti M:N določimo pripadajoči relaciji (tabeli) S in T, za relacijo pa uvedemo novo relacijo (tabelo), v katero prenesemo kot tuja ključa primarna ključa relacij S in T in dodamo morebitne atribute relacije, ki povezuje entiteti En in Em.

$$En\ (En1, En2, ..., Enm) \rightarrow S\ (S1, S2, ..., Sm)$$

$$Em\ (Em1, Em2, ..., Emk) \rightarrow T\ (T1, T2, ..., Tk)$$

$$Re\ (A1, A2, ..., An) \rightarrow R\ (S1, T1, A1, A2, ..., An\ )$$

4.3. PRETVORBA IZ E-R MODELA V RELACIJSKI PODATKOVNI MODEL

Primer :

1. korak: vsako entiteto pretvorimo v relacijo

Delavec (*matična številka*, ime, priimek, naslov, rojstni datum, OD)

Oddelek (*številka oddelka*, ime oddelka, lokacija oddelka)

Projekt (*številka projekta*, naziv projekta, lokacija projekta)

4.3. PRETVORBA IZ E-R MODELA V RELACIJSKI PODATKOVNI MODEL

2. korak: entitete v relaciji s kardinalnostjo 1:1

Delavec vodi oddelek (1:1).

Vodi (datum nastopa funkcije) ®

Delavec (*matična številka*, ime, priimek, naslov, rojstni datum, OD, ODD-številka oddelka, datum pričetka funkcije)

Oddelek (*številka oddelka*, ime oddelka, lokacija oddelka)

ALI

Vodi (datum nastopa funkcije) ®

Delavec (*matična številka*, ime, priimek, naslov, rojstni datum, OD)

Oddelek (*številka oddelka*, ime oddelka, lokacija oddelka, DEL-matična številka, datum pričetka funkcije)

4.3. PRETVORBA IZ E-R MODELA V RELACIJSKI PODATKOVNI MODEL

3. korak: entitete v relaciji s kardinalnostjo 1:N

Oddelek zaposluje delavce (1:N).

Oddelek izvaja projekte (1:N).

Zaposluje (datum zaposlitve)

Oddelek (*številka oddelka*, ime oddelka, lokacija oddelka)

Delavec (*matična številka*, ime, priimek, naslov, rojstni datum,

OD, ODD-številka oddelka, datum zaposlitve)

4.3. PRETVORBA IZ E-R MODELA V RELACIJSKI PODATKOVNI MODEL

Izvaja (začetek projekta, predviden zaključek projekta)

*** Oddelek (*številka oddelka*, ime oddelka, lokacija oddelka)

Projekt (*številka projekta*, naziv projekta, lokacija projekta,

ODD-številka oddelka, začetek projekta, predviden zaključek

projekta)

4.3. PRETVORBA IZ E-R MODELA V RELACIJSKI PODATKOVNI MODEL

4. korak: entitete v relaciji s kardinalnostjo M:N

Delavci sodelujejo na projektih (M:N).

Sodeluje (št. ur)

*** Projekt (*številka projekta*, naziv projekta, lokacija projekta, ODD številka oddelka, začetek projekta, predviden zaključek projekta)

*** Delavec (*matična številka*, ime, priimek, naslov, rojstni datum, OD, ODD-številka oddelka, datum zaposlitve)

*** Sodelovanje (*PRO-številka projekta*, *DEL-matična številka*, št.ur)

*** - končne relacije

5. SISTEM ZA UPRAVLJANJE S PODATKOVNO BAZO – SUPB

- SUPB je programski produkt, ki podpira zanesljivo hranjenje podatkov, implementira strukturo povezav in omejitev ter omogoča vračanje podatkov (povpraševaje po podatkih). V bistvu SUPB zagotavlja potrebno organizacijsko strukturo za uspešno hranjenje in dostopanje do velike količine podatkov.

- Je programski sistem, ki uporabniku omogoča definiranje, oblikovanje in vzdrževanje podatkovne baze, hkrati pa omogoča nadzor nad dostopom do podatkovne baze.

5. SISTEM ZA UPRAVLJANJE S PODATKOVNO BAZO – SUPB

- SUPB omogoča:
  - definiranje podatkovne baze - jezik za definiranje podatkov (angl. data definition language - DDL),
  - vnašanje, popravljanje, brisanje in vračanje podatkov - jezik za delo (ravnanje) s podatki (angl. data manipulation language - DML),
  - povpraševanje z uporabo povpraševalnega jezika (angl. query language),
  - nadzor nad dostopom do podatkovne baze:
    - nadzor dostopa do podatkovne baze,
    - integriteta podatkovne baze,
    - vzpostavitev nadzornega sistema,
    - uporabniško dostopen opis podatkov,
  - pregled podatkov.

5.1. FUNKCIJE SUPB

- SUPB bi naj zagotavljal naslednje funkcije z različnih področij:
  - **Shranjevanje, vračanje, popravljanje (izboljšanje) podatkov:**
    - SUPB mora oskrbeti uporabnika z možnostmi za shranjevanje, dostopanje in popravljanje podatkov in podatkovne baze.
  - **Uporabniško dostopen katalog:**
    - SUPB mora zagotoviti uporabniku dostopen katalog (podatkovni slovar, repozitorij), v katerem so shranjeni opisi o shranjenih podatkih - metapodatki.

5.1. FUNKCIJE SUPB

- V podatkovnem slovarju shranjujemo:
  - imena, podatkovne tipe in velikost podatkov,
  - imena relacij (povezav),
  - celovitostne omejitve nad podatki,
  - imena avtoriziranih uporabnikov, ki imajo dostop do podatkov,
  - zunanji, konceptualni in notranji model ter prehod v logični podatkovni model,
  - uporabna statistika (frekvenca transakcij, število omejitev).

5.1. FUNKCIJE SUPB

- Še nekaj prednosti podatkovnega slovarja:
  - zbiranje informacij o podatkovnem slovarju za podatkovni slovar zagotavlja nadzor nad podatki,
  - definiramo lahko pomen podatkov,
  - komunikacija je enostavna,
  - redundanco je lažje odkriti,
  - beležijo se spremembe nad podatkovno bazo,
  - možni sta zaščita in varnost.

5.1. FUNKCIJE SUPB

- **Podpora transakcijam**
  - SUPB mora zagotoviti mehanizem za zagotavljanje beleženja posameznih transakcij.
- **Soglasen nadzor**
  - SUPB mora zagotoviti popravljanje podatkovne baze.
- **Sistem ponovne vzpostavitve**
  - SUPB mora zagotoviti ponovno vzpostavitev podatkovne baze po poškodbi.
- **Sistem avtorizacije**
  - SUPB mora zagotoviti sistem, ki samo avtoriziranim uporabnikom omogoča dostop do podatkovne baze.

5.1. FUNKCIJE SUPB

- **Podpora za komuniciranje**
  - SUPB mora biti pripravljen na integracijo s komunikacijsko programsko opremo.
- **Podpora podatkovni neodvisnosti**
  - SUPB mora zagotavljati podporo neodvisnosti programov od aktualne strukture podatkovne baze.
- **Podporni servisi**
  - SUPB zagotavljajo množico podpornih servisov.

5.1. FUNKCIJE SUPB

- Podporni servisi pomagajo administratorju podatkovne baze pri administriranju le-te:
  - orodje za uvoz in izvoz (angl. export in import) podatkov,
  - nadzor nad uporabo podatkovne baze in operacijami nad njo,
  - programi za statistično analizo,
  - pripomočki za reorganizacijo indeksov,
  - fizična odstranitev brisanih podatkov in relacij.

5.2. RELACIJSKI SUPB

Nekaj sto različnih relacijskih SUPB za različna okolja.

- **PROBLEM:**

  Mnogi ne sledijo natančno definiciji relacijskega podatkovnega modela.

- **REŠITEV:**

  Uvedba trinajstih pravil za relacijski SUPB (Codd 1985):
  - osnovna pravila,
  - strukturna pravila,
  - pravila celovitosti,
  - pravila za delo (ravnanje) s podatki,
  - pravila podatkovne neodvisnosti.

5.2. RELACIJSKI SUPB

Osnovna pravila:

- Vsak SUPB mora izpolniti pravili 0 in 12, da ga lahko imenujemo relacijski SUPB.

- **Pravilo 0 - osnovno pravilo**
  - Vsak sistem SUPB, ki se imenuje ali želi imenovati relacijski, mora omogočati upravljanje podatkovne baze v celoti v skladu z relacijskimi lastnostmi.

- **Pravilo 12 - pravilo nezrušitve**
  - Če relacijski sistem vsebuje jezik “nizkega” nivoja, ga ne moremo uporabiti za rušitev pravil celovitosti oz. jih obiti z jezikom “višjega” nivoja.

5.2. RELACIJSKI SUPB

Strukturna pravila:

- Osnova struktura relacijskega podatkovnega modela je relacija.

- **Pravilo 1 - predstavitev informacije**
  - Vse informacije relacijskega podatkovnega modela so predstavljene na logičnem nivoju nedvoumno in natančno z vrednostmi in tabelami.

- **Pravilo 6 - izboljšanje pogledov**
  - Poglede, ki jih je teoretično možno izboljšati, lahko izboljša (popravi) tudi sistem.

5.2. RELACIJSKI SUPB

Pravila celovitosti:

- Podpora celovitosti podatkov je pomembno merilo primernosti SUPB. Kakovost podatkov se veča s količino nadzora celovitosti, ki ga opravlja SUPB v primerjavi z aplikacijskim programom.

- **Pravilo 3 - sistematično obravnavanje vrednosti NULL**
  - NULL vrednost je uporabljena za sistematično predstavitev manjkajočih vrednosti, neodvisno od podatkovnega tipa.

- **Pravilo 10 - celovitostna odvisnost**
  - Celovitnostno omejitev določene relacijske podatkovne baze definiramo s pomočjo podjezika relacijskih podatkov in sinonima v katalogu (podatkovnem slovarju) in ne v

5.2. RELACIJSKI SUPB

Pravila za delo (ravnanje) s podatki:

- Idealen SUPB bi naj podpiral 18 različnih kategorij za ravnanje s podatki.

- **Pravilo 2 - zagotovljen dostop**
  - Vsaka vrednost v podatkovni bazi je zanesljivo dostopna s kombinacijo ime relacije, primarni ključ in ime stolpca (atributa).

- **Pravilo 4 - dinamičen "on-line" katalog na osnovi relacijskega podatkovnega modela**
  - Opis podatkovne baze na logičnem nivoju je predstavljen na identičen način kot podatki, kar omogoča avtoriziranemu uporabniku uporabo istega povpraševalnega jezika.

5.2. RELACIJSKI SUPB

- **Pravilo 5 - vsestranski podatkovni podjezik**
  - Relacijski SUPB lahko podpira različne jezike in modele za terminalsko uporabo, vendar mora obstajati vsaj en jezik, ki omogoča:
    - definiranje podatkov,
    - definiranje pogledov,
    - delo (ravnanje) s podatki,
    - zagotavljanje omejitev celovitosti,
    - avtorizacijo,
    - transakcijske operacije (begin, commit, rollback).

5.2. RELACIJSKI SUPB

- **Pravilo 7 - “visokonivojski” vnos, popravljanje, brisanje**
  - Možnost obravnavanja podatkovne baze kot relacije ne omogoča le vračanja podatkov, temveč tudi njihov vnos, popravljanje in brisanje.

Pravila podatkovne neodvisnosti:

- Za predstavitev neodvisnosti podatkov od aplikacij, ki uporabljajo te podatke, so definirana tri pravila.

- **Pravilo 8 - fizična neodvisnost podatkov**
  - Aplikacijski programi in terminalske aktivnosti ostajajo logično nespremenjene ob kakršnikoli spremembi pomnilniške predstavitve oz. metode dostopa.

5.2. RELACIJSKI SUPB

- **Pravilo 9 - logična neodvisnost podatkov**
  - Aplikacijski programi in terminalske aktivnosti ostanejo logično nespremenjene, kadarkoli so informacijske spremembe izvedene nad relacijami podatkovne baze.

- **Pravilo 11 - distribucijska (porazdeljena) neodvisnost**
  - Podjezik za delo s podatki relacijskega SUPB mora omogočiti aplikacijskim programom in povpraševanjem, da ostanejo logično nespremenjeni, kadar so podatki fizično centralizirani oz. porazdeljeni (distribuirani).

5.3. POMANJKLJIVOSTI RELACIJSKEGA SUPB

- Pomanjkljivosti relacijskega podatkovnega modela so najpogosteje omenjene pri zagovornikih objektno orientiranega pristopa.

- **Predstavitev entitet “realnega sveta”**
  - Proces normalizacije povzroči pojav entitet oz. relacij, ki v “realnem svetu” ne obstajajo.

- **Semantična preobremenjenost**
  - Relacijski podatkovni model uporablja za predstavitev podatkov in podatkovni povezav le relacije, kar pomeni da ni nobene ločitve med predstavitvijo entitet in relacij oz. relacij različnih kardinalnosti.

5.3. POMANJKLJIVOSTI RELACIJSKEGA SUPB

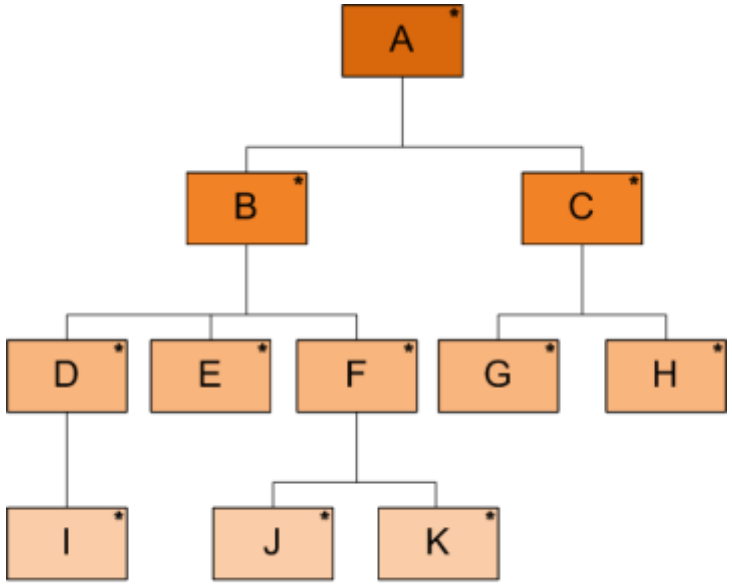
- **Istovrstnost (homogenost) podatkov**
  - Relacijski podatkovni model omogoča horizontalno in vertikalno homogenost (istovrstnost):
    - horizontalna homogenosti - vsaka n-terica je sestavljena iz istih atributov,
    - vertikalna homogenost - vrednosti posameznega stolpca pripadajo isti domeni,
    - presek vrstice in stolpca je tako osnovna vrednost, ki pa je v realnem svetu pogosto predstavljena s kompleksnejšo strukturo (kompleksni objekti, NF2).

5.3. POMANJKLJIVOSTI RELACIJSKEGA SUPB

- **Omejitev operacij**
  - Relacijski model ima končno množico operacij.
- **Rekurzivno povpraševanje**
  - Predstavitev rekurzivnega povpraševanja je zahtevno. Rešitev omogoča širitev visokonivojskih programskih jezikov z SQL.
- **Ostali problemi:**
  - transakcije z dolgo “življenjsko dobo” so izjema,
  - spremembe sheme so zahtevne.

5.4. DREVESNI PODATKOVNI MODEL

- Osnovna struktura drevesnega podatkovnega modela je drevo, sestavljeno iz vozlišč v katerih se nahajajo posamezni zapisi (segmenti) in vej (linki), ki vozlišča povezujejo.



Slika 29: Osnova struktura drevesnega modela

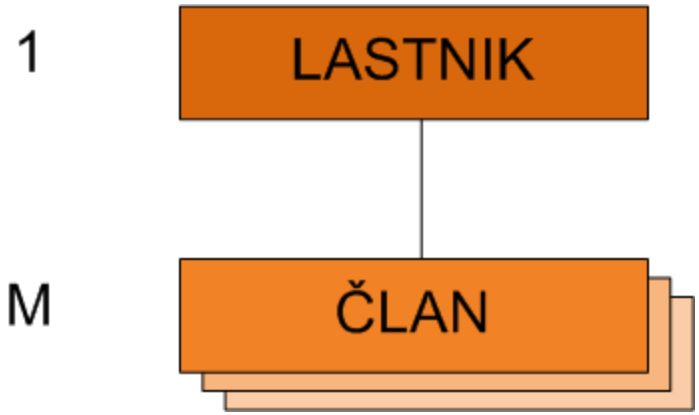
5.4. DREVESNI PODATKOVNI MODEL

- V drevesnem podatkovnem modelu obstajajo povezave starši-otroci (1:M):
  - starši imajo lahko več otrok,
  - vsak otrok ima le en starše,
  - starše najvišjega nivoja imenujemo koren,
  - otrok brez otrok je list.

- **Hierarhična pot** – urejeno zaporedje segmentov, ki omogočajo sledenje drevesni strukturi.

5.5. MREŽNI PODATKOVNI MODEL

- Po mrežni terminologiji imenujemo relacijo “množico” (ang. set), ki jo sestavljata dva tipa zapisov:
  - lastnik (ang. owner) in
  - član (ang. member).



Slika 30: Množica (set)

5.5. MREŽNI PODATKOVNI MODEL

- Kardinalnost znotraj množice je 1:M, vendar ima lahko član več lastnikov.

- Lastnosti množice (seta):
  - množica ima samo enega lastnika,
  - množica ima lahko enega ali več članov,
  - posamezen zapis ne more biti hkrati lastnik in član iste množice,
  - zapis je lahko lastnik in član poljubnega števila množic.

5.6. PRIMERJAVA PODATKOVNIH MODELOV

Entitetno-relacijski model	Relacijski model - formalno	Relacijski model - neformalo	Mrežni model	Drevesni model
E-R diagram	Relacijska shema	Opis tabel	Diagram zapisov	Diagram zapisov
Entiteta	Relacija	Tabela	Zapis	Zapis
Primerek entitete	N-terica	Vrstica	Primerek zapisa	Primerek zapisa
Relacija 1:N			Množica	Relacija starš-otrok (RSO)
Primerek 1:N relacije			Primerek množice	Primerek RSO
Atribut	Atribut	Stolpec	Polje ali podatkovni el.	Polje ali podatkovni el.

Primerjava terminologij

5.6. PRIMERJAVA PODATKOVNIH MODELOV

Entitetno-relacijski model	Relacijski model - formalno	Relacijski model - neformalo	Mrežni model	Drevesni model
Ključ	Kandidatni ključ, primarni ključ	Kandidatni ključ, primarni ključ	Ključ ali unikatno polje	Sekvenčni ključ ali sekvenčno polje
Več-vrednostni atribut			Vektor ali ponavljajoča se skupina	
Sestavljen atribut			Ponavljajoča se skupina	

Primerjava terminologij

5.6. PRIMERJAVA PODATKOVNIH MODELOV

Koncept E-R modela	Relacijski model	Mrežni model	Drevesni model
Entiteta	Relacija	Zapis	Zapis
Relacija kardinalnosti 1:1	Vključi primarni ključ druge relacije kot tuj ključ ali združi obe v eno relacijo	Uporabi množico, katere elementi smejo imeti le en zapis člana ali združi v en zapis	Uporabi relacijo, starš-otrok, katere starši smejo imeti le enega otroka ali združi v en zapis
Relacija kardinalnosti 1:N	Vključi primarni ključ s strani “1”, kot tuj ključ v stran “N”	Uporabi množico	Uporabi relacijo starš-otrok
Relacija kardinalnosto M:N	Uporabi novo relacijo, ki vsebuje kot tuja ključa primarna ključa obeh sodelujočih relacij	Uporabi nov povezujoč zapis in ga napravi kot člana množice, katere lastnika sta povezana zapisa	a) Uporabi eno drevesno strukturo in uporabljaj redundančne zapise ali b) uporabi več drevesnih struktur in relacije starš-otrok z navideznimi starši

Primerjava uporabe E-R konceptov

5.6. PRIMERJAVA PODATKOVNIH MODELOV

Koncept	Relacijski model	Mrežni model	Drevesni model
Fizična podatkovna struktura	Dvo-dimenzionalne tabele	Mrežna predstavitev, ki združuje množice povezav s kazalci	Drevesna struktura s korenom in odvisnimi zapisi
Datotečna organizacija	Zaporedne datoteke z indeksom, datoteke z direktnim dostopom, kompleksne drevene strukture	Metode direktnega dostopa in indeksno-sekvenčni dostop	HIDAM, HDAM, HISAM, HSAM
Oblikovanje	Lahek za oblikovanje in razmevanje	Potrebna so natančna navodila	Prepuščeno ekspertu

Primerjava konceptov podatkovne baze

5.6. PRIMERJAVA PODATKOVNIH MODELOV

Koncept	Relacijski model	Mrežni model	Drevesni model
Dostop do podatkov	Povpraševalni jeziki, aplikacijski vmesniki	Aplikacijski vmesniki, dodatno programski vmesniki	Aplikacijski vmesniki, dodatni programski vmesniki
Implementacija	Zapisi z vrednostmi, ki nadomeščajo logične kazalce	Zapisi in kazalci	Običajno zapisi in kazalci
Strandard	Obstajajo množice standardov, ki so različno implementirani	Množica standardov obstaja za CODASYL z različnimi implementacijami	Ni natančnih standardov

Primerjava konceptov podatkovne baze

6. RELACIJSKA ALGEBRA

- **Osnovne operacije:**
  - unija,
  - razlika,
  - kartezični produkt,
  - projekcija,
  - selekcija.

- **Izvedene operacije:**
  - presek,
  - Θ-stik,
  - naravni stik,
  - količnik.

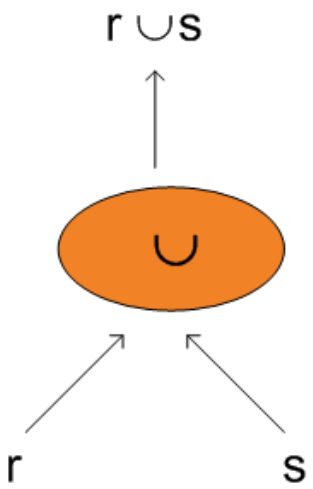
6. RELACIJSKA ALGEBRA

- **Sosledje operacij:**
  - selekcija,
  - projekcija,
  - kartezični produkt,
  - Θ-stik,
  - naravni stik,
  - količnik,
  - presek,
  - unija,
  - razlika.

- Sosledje operacij lahko spremenimo z oklepaji.

6.1. UNIJA r ∪ s

- Predpogoj:
  - Kompatibilnost unije relaciji r(A_1, A_2, …, A_n) in s(B_1, B_2, …, B_n) izpolnjujeta pogoj kompatibilnosti unije, če sta stopnje n in dom(A_i) = Dom(B_i) za $1 \leq i \leq n$.
- Definicija:
  - Rezultat operacije r ∪ s je relacija, ki vsebuje vse n-terice, ki so v relaciji r ali v relaciji s ali v obeh. Ponavljajoče se vrstice so izločene.
- Grafična predstavitev:



- Opomba: Operacija je komutativna.
  - r ∪ s = s ∪ r

6.1. UNIJA r ∪ s

Primer: Oddelek ∪ Oddelek1

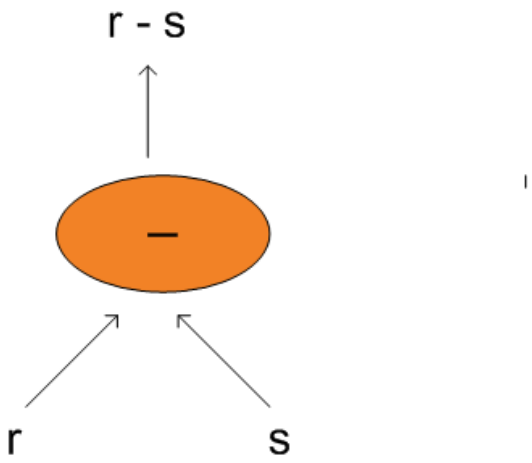
ODDELEK		
ŠtOdd	NazivOdd	Kraj
o_1	n_1	k_1
o_2	n_2	k_1
o_3	n_3	k_2

ODDELEK1		
ŠtOdd	NazivOdd	Kraj
o_1	n_1	k_1
o_4	n_4	k_2
o_5	n_5	k_3

ODDELEK ∪ ODDELEK1		
ŠtOdd	NazivOdd	Kraj
o_1	n_1	k_1
o_2	n_2	k_1
o_3	n_3	k_2
o_4	n_4	k_2
o_5	n_5	k_3

6.2. RAZLIKA r - s

- Predpogoj:
  - Kompatibilnost unije.
- Definicija:
  - Rezultat operacije r - s je relacija, ki vsebuje vse n-terice, ki so v relaciji r in niso v relaciji s.
- Grafična predstavitev:



- Opomba: Operacija ni komutativna.
  - r - s ≠ s - r

6.2. RAZLIKA r – s

Primer: Oddelek - Oddelek1

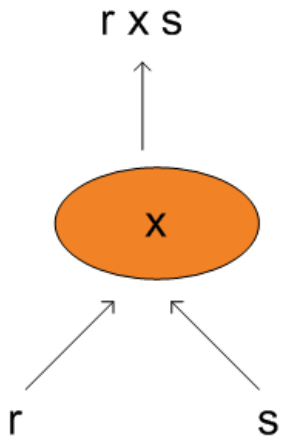
ODDELEK - ODDELEK1		
ŠtOdd	NazivOdd	Kraj
o_2	n_2	k_1
o_3	n_3	k_2

Primer: Oddelek1 - Oddelek

ODDELEK1 - ODDELEK		
ŠtOdd	NazivOdd	Kraj
o_4	n_4	k_2
o_5	n_5	k_3

6.3. KARTEZIČNI PRODUKT r × s

- Predpogoj:
  - Kompatibilnost unije ni zahtevana.
- Definicija:
  - Rezultat operacije r × s je relacija Q z n+m atributi: $Q(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m)$.
- Grafična predstavitev:



- Opomba: Kartezični produkt se redko uporablja kot samostojna operacija.

6.3. KARTEZIČNI PRODUKT r × s

Primer: Oddelek × Projekt

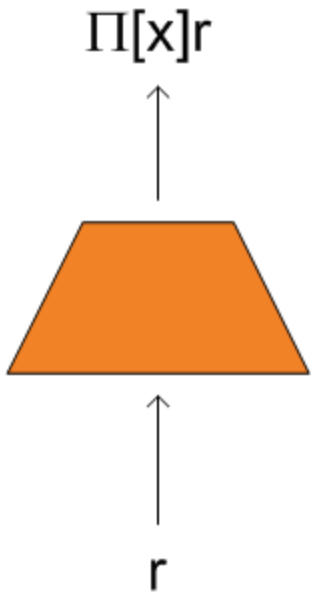
ODDELEK		
ŠtOdd	NazivOdd	Kraj
o_1	n_1	k_1
o_2	n_2	k_1
o_3	n_3	k_2

PROJEKT		
ŠtPro	Naziv	Sredstva
o_1	n_1	k_1
o_4	n_4	k_2
o_5	n_5	k_3

ODDELEK × PROJEKT					
ŠtOdd	NazivOdd	Kraj	ŠtPro	Naziv	Sredstva
o_1	n_1	k_1	o_1	n_1	k_1
o_2	n_2	k_1	o_4	n_4	k_2
o_3	n_3	k_2	o_5	n_5	k_3
o_1	n_1	k_1	o_4	n_4	k_2
o_2	n_2	k_1	o_1	n_1	k_1
o_3	n_3	k_2	o_1	n_1	k_1
o_1	n_1	k_1	o_5	n_5	k_3
o_2	n_2	k_1	o_5	n_5	k_3
o_3	n_3	k_2	o_4	n_4	k_2

6.4. PROJEKCIJA Π[X] r

- Predpogoj:
  - Operacija je smiselna, če je X $\subseteq$ r.
- Definicija:
  - Operacija projekcije izvede selekcijo množice X (X je množica izbranih atributov) iz relacije r.
- Grafična predstavitev:



- Opomba: Izločitev ponavljajočih se vrstic.

6.4. PROJEKCIJA Π[X] r

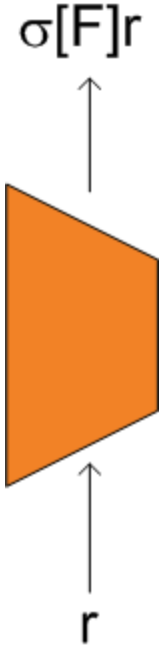
Primer: Π [POKLIC, ODD] DELAVEC

DELAVEC			
ŠtDel	ImeDel	Poklic	ŠtOdd
d_1	i_1	p_1	o_1
d_2	i_2	p_2	o_2
d_3	i_3	p_3	o_3
d_4	i_4	p_1	o_1
d_5	i_5	p_2	o_3

POKL-ODD	
Poklic	ŠtOdd
p_1	o_1
p_2	o_2
p_3	o_3
p_2	o_3

6.5. SELEKCIJA σ[F] r

- Definicija:
  - Selekcija omogoča izbor podmnožice n-teric iz dane relacije r, ki izpolnjujejo pogoj F. Pogoj (angl. Formula, selection condition) vsebuje spremenljivke (imena atributov), konstante in logične operatorje (=, <, ≤, >, ≥, ≠).
- Grafična predstavitev:



- Opomba: Vrstni red upoštevanja pogojev ni pomemben. Operacija je komutativna.
  - σ[F1] (σ[F2]r) = σ[F2] (σ[F1]r).

6.5. SELEKCIJA σ[F] r

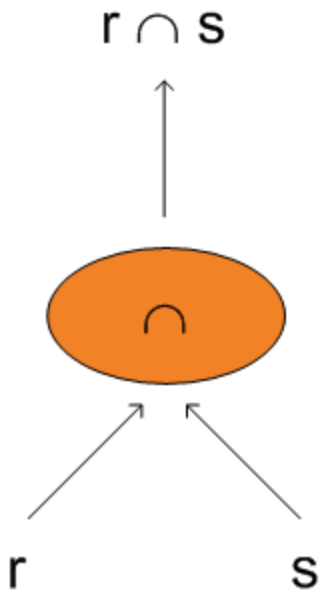
Primer: σ [NALOGA=r2] DELOVNE NALOGE

DELOVNE NALOGE		
ŠtDel	ŠtPro	Naloga
d_1	p_1	r_2
d_2	p_1	r_3
d_2	p_2	r_3
d_3	p_2	r_1
d_3	p_1	r_1
d_4	p_1	r_1
d_5	p_2	r_2

DELOVNE NALOGE		
ŠtDel	ŠtPro	Naloga
d_1	p_1	r_2
d_5	p_2	r_2

6.6. PRESEK r ∩ s

- Predpogoj:
  - Kompatibilnost unije.
- Definicija:
  - Rezultat operacije r ∩ s je relacija, ki vsebuje vse n-terice, ki so v relaciji r in v relaciji s. Ponavljajoče se vrstice so izločene.
- Definicija z osnovnimi operacijami:
  - r ∩ s ≡ r - (r - s)
- Grafična predstavitev:
- Opomba: Operacija je komutativna.
  - r ∩ s = s ∩ r



6.6. PRESEK r ∩ s

Primer: Oddelek ∩ Oddelek1

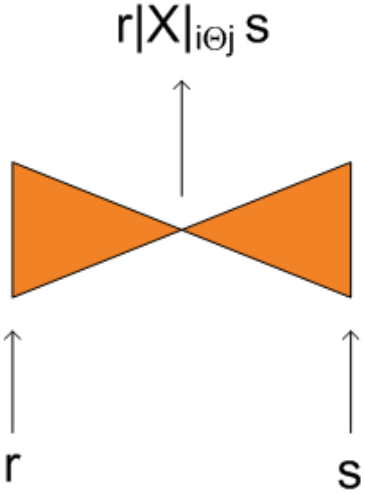
ODDELEK		
ŠtOdd	NazivOdd	Kraj
o_1	n_1	k_1
o_2	n_2	k_1
o_3	n_3	k_2

ODDELEK1		
ŠtOdd	NazivOdd	Kraj
o_1	n_1	k_1
o_4	n_4	k_2
o_5	n_5	k_3

ODDELEK ∩ ODDELEK1		
ŠtOdd	NazivOdd	Kraj
o_1	n_1	k_1

6.7. Θ - STIK r $|x|_{i\Theta j}$ s

- Definicija:
  - Rezultat operacije r (A_1, A_2, …, A_n) $|x|_{i\Theta j}$ s(B_1, B_2, …, B_n) je relacija Q z m+n atributi, ki izpolnjuje pogoj stika i Θ j. I in j sta atributa,Θ pa eden izmed logičnih operaterjev (=, <, ≤, >, ≥, ≠). Pogoj i Θ j je lahko zapisan tudi v obliki konjunkcije.
- Definicija z osnovnimi operacijami:
  - r $|x|_{i\Theta j}$ s ≡ σ[F] (r × s)
- Grafična predstavitev:



- Opomba: Rezultirajoča relacija Q je prazna, če pogoj stika ni izpolnjen in preide v kartezični produkt, če pogoja ni.

6.7. Θ - STIK r $|x|_{i\Theta j}$ s

Primer: ODDELEK $|x|_{kraj = k2}$ PROJEKT

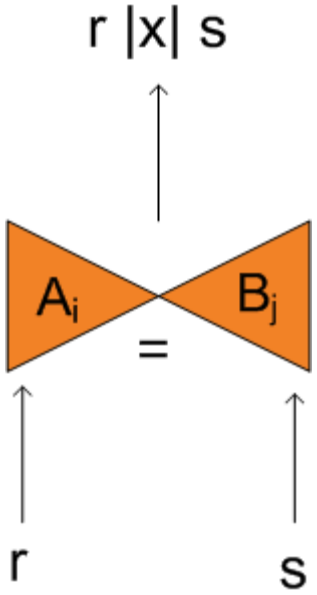
ODDELEK		
ŠtOdd	NazivOdd	Kraj
o_1	n_1	k_1
o_2	n_2	k_1
o_3	n_3	k_2

PROJEKT		
ŠtPro	Naziv	Sredstva
o_1	n_1	k_1
o_4	n_4	k_2
o_5	n_5	k_3

ODDELEK $\|x\|_{KrajΘk2}$ PROJEKT					
ŠtOdd	NazivOdd	Kraj	ŠtPro	Naziv	Sredstva
o_3	n_3	k_2	o_1	n_1	k_2
o_3	n_3	k_2	o_4	n_4	k_2
o_3	n_3	k_2	o_5	n_5	k_3

6.8. NARAVNI STIK r |x| s

- Definicija:
  - Rezultat operacije r relaciji r |x| s je relacija Q z m+n-k atributi (kot pri Θ-stiku), pri čemer je v pogoju stika vedno uporabljen le logični operator = in atribut $B_j(r.A_i=s.B_j)$ ni vključen v rezultirajoči relaciji.
- Definicija z osnovnimi operacijami:
  - $r \,|x|\, s \equiv \Pi\,[x](\sigma[F]\,(r \times s))$
- Grafična predstavitev:



- Opomba: Operacija je asociativna.

6.8. NARAVNI STIK r |x| s

Primer: DELAVEC |x| $_{d.ŠtOdd = o.ŠtOdd}$ ODDELEK

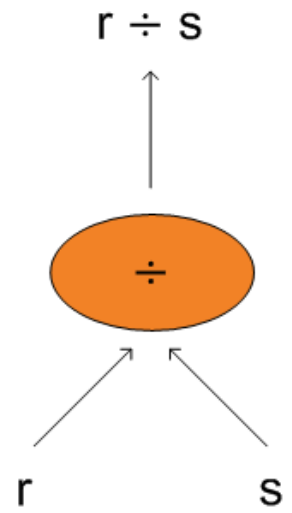
DELAVEC			
ŠtDel	ŠtOdd	Poklic	ImeDel
d_1	o_1	p_1	i_1
d_2	o_2	p_2	i_2
d_3	o_3	p_3	i_3
d_4	o_1	p_1	i_4
d_5	o_3	p_2	i_5

ODDELEK		
ŠtOdd	NazivOdd	Kraj
o_1	n_1	k_1
o_2	n_2	k_1
o_3	n_3	k_2

DELAVEC |x| ODDELEK					
ŠtDel	ŠtOdd	ImeDel	Poklic	NazivOdd	Kraj
d_1	o_1	i_1	p_1	n_1	k_1
d_2	o_2	i_2	p_2	n_2	k_1
d_3	o_3	i_3	p_3	n_3	k_2
d_4	o_1	i_4	p_1	n_1	k_1
d_5	o_3	i_5	p_2	n_3	k_2

6.9. KOLIČNIK r / s

- Predpogoj:
  - Operacija je smiselna, če $S \subseteq R$.
- Definicija:
  - Rezultat operacije r relaciji r / s je relacija Q z n-terico t, za katero velja, da za vsako n-terico ts v s obstaja n-terica tr v r, ki izpolnjuje naslednja pogoja:
    $t_r[s] = t_s[s]$ in $t_r[r - s] = t[r - s]$.
- Definicija z osnovnimi operacijami:
  - $r / s \equiv \Pi [x] r - \Pi [x] (((\Pi [x] r) \times s) - r); x=R-S$
- Grafična predstavitev:



- Opomba: operacija količnik je tipična za zahteve tipa “all” (iščemo šifre delavcev in delovnih nalog tistih delavcev, ki so zaposleni pri vseh projektih).

7. POVPRAŠEVALNI JEZIKI

- Komercialni produkti:
  - SQL, QBE, QUEL.
- Pogoj:
  - relacijska kompletnost:
    - povpraševalni jezik je relacijsko kompleten, če z njim lahko izrazimo katerokoli vprašanje, ki ga lahko zapišemo s pomočjo relacijskega računa.

7. POVPRAŠEVALNI JEZIKI

- Omogoča:
  - primerljivost izrazne moči posameznih povpraševalnih jezikov.
- Stanje:
  - izrazna moč povpraševalnih jezikov običajno presega relacijsko algebro in relacijski račun.

7.1. SQL – STRUCTURED QUERY LANGUAGE

- Rojstvo:
  - zgodnja 70-ta leta,
  - IBM raziskovalni laboratorij.
- Projekt:
  - System R.
- Ime:
  - SEQUEL (Structured English QUEry Language).
- Osnova:
  - n-terični relacijski račun, relacijska algebra.

7.1. SQL - STRUCTURED QUERY LANGUAGE

- Standard:
  - 1986 ANSI SQL (technical Committee X3H2 - Database),
  - 1992 SQL2 standard,
  - IBM ASS-SQl Standard.

- Cilji standarda:
  - specifikacija sintakse in semantike, SQL/DDL in SQL/DML,
  - definicija podatkovne strukture in osnovnih operacij za oblikovanje, dostop, vzdrževanje, nadzor in zaščito SQL baze podatkov,
  - zagotovitev prenosljivosti definicij baze podatkov in operacijskih modulov med ustreznimi SUPBji,
  - specifikacija standardov za vključitev v produkt,
  - zagotovitev začetnega standarda za probleme integritete,

7.1. SQL - STRUCTURED QUERY LANGUAGE

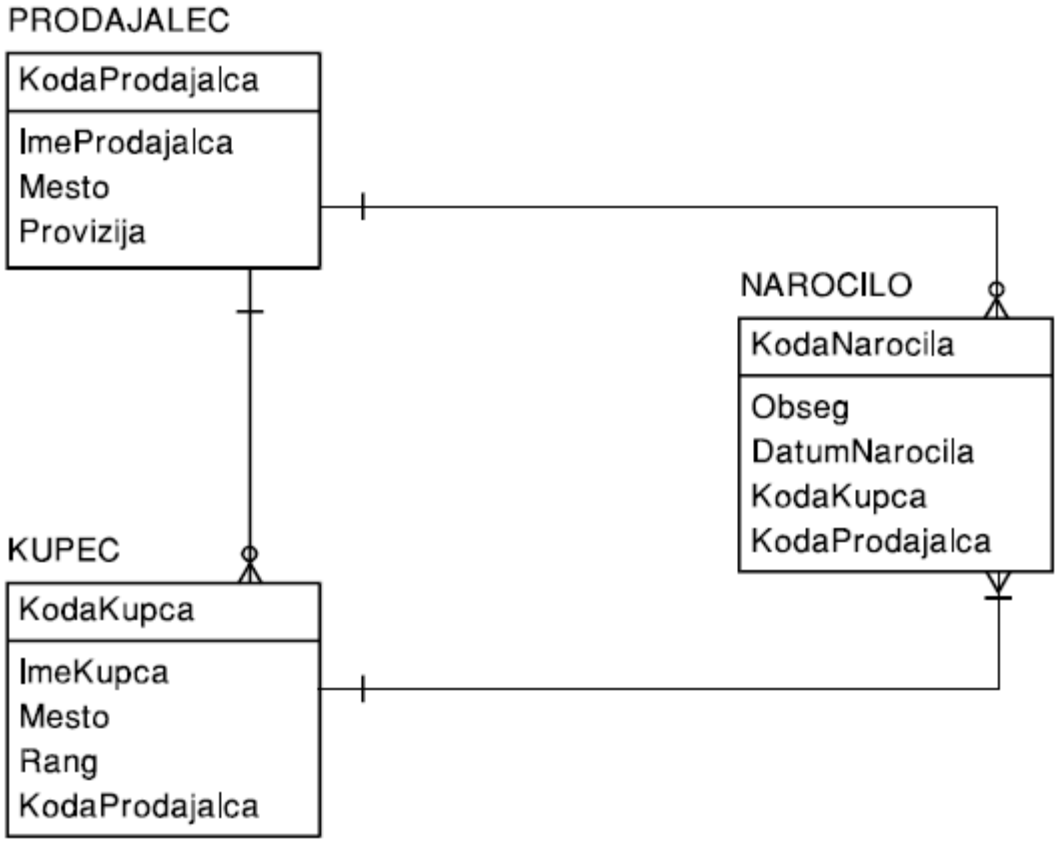
- Lastnosti:
  - enodimenzionalna sintaksa,
  - lahek za učenje.
- Kaj zmore:
  - DDL (data definition language): Zagotavlja ukaze za definiranje relacijske sheme, oblikovanje indeksov in definiranje pogledov (CREATE), brisanje relacij (DROP), spreminjanje relacijske sheme (ALTER) in dodeljevanje pravic dostopa (GRANT/REVOKE),
  - DML (data manipulation language): Vključuje povpraševalni jezik in ukaze za vnos (INSERT), brisanje (DELETE) in popravljanje (UPDATE) n-teric,
  - omogoča izvajanje omejitev integritete in nadzor nad transakcijami.

7.1. SQL - STRUCTURED QUERY LANGUAGE

- Osnovna struktura vprašanja:
  - SELECT $A_1, A_2, \ldots, A_n$
    FROM $r_1, r_2, \ldots, r_m$
    WHERE P
    - SELECT: omogoča oblikovanje seznama rezultirajočih atributov. Istovetimo ga z operacijo projekcija iz relacijske algebre.
    - FROM: oblikuje seznam potrebnih relacij. Istovetimo ga z operacijo kartezični produkt iz relacijske algebre.
    - WHERE: vsebuje pogoje, izražene z atributi relacij iz stavka FROM. Istovetimo ga z operacijo selekcija iz relacijske algebre.
- Ekvivalenten zapis:
  - $\Pi\,[A_1, A_2, \ldots, A_n](\sigma[P]\,(r_1 \times r_2 \times \ldots \times r_m)))$

7.1.1. SQL PRIMER

- E-R diagram



Slika 28: E-R diagram

7.1.1. SQL PRIMER TRGOVINA

- Vsebina tabel

PRODAJALEC			
KodaProdajalca	ImeProdajalca	Mesto	Provizija
1001	Sisek	Ljubljana	0.12
1002	Kres	Novo Mesto	0.13
1003	Palek	Ljubljana	0.11
1004	Rakus	Maribor	0.15
1005	Dekar	Celje	0.21
1006	Aron	Portorož	0.10

7.1.1. SQL PRIMER TRGOVINA

- Vsebina tabel

KUPEC				
KodaKupca	ImeKupca	Mesto	Rang	*KodaProdajalca*
2001	Hord	Ljubljana	100	1001
2002	Gregr	Rogla	200	1003
2003	Lukic	Novo Mesto	200	1002
2004	Porkaro	Koper	300	1002
2005	Cikl	Ljubljana	100	1001
2006	Curcio	Novo Mesto	300	1007
2007	Pristl	Rogla	100	1004

7.1.1. SQL PRIMER TRGOVINA

- Vsebina tabel

NAROCILO				
KodaNarocila	Obseg	DatumNarocila	*KodaKupca*	*KodaProdajalca*
3001	18.69	02.10.1990	2008	1007
3003	767.19	03.10.1990	2001	1001
3002	1900.1	09.10.1990	2007	1004
3005	5160.45	10.10.1990	2003	1002
3006	1098.16	11.10.1990	2008	1007
3009	1713.12	12.10.1990	2002	1003
3007	75.75	12.10.1990	2004	1002
3008	4723	14.10.1990	2006	1001
3010	1309.95	22.10.1990	2004	1002
3011	9891.88	23.10.1990	2006	1001

7.1.1. SQL PRIMER TRGOVINA

- Izpišite kodo, obseg in datum vseh naročil iz tabele Naročilo! (vertikalna izbira - projekcija)

```
SELECT      KodaNarocila, Obseg, DatumNarocila
FROM        Narocilo;
```

- Izpišite vse kupce, ki kupujejo pri prodajalcu z kodo 1001! (horizontalnha izbira - selekcija)

```
SELECT      *
FROM        Kupec
WHERE       KodaProdajalca=1001;
```

7.1.1. SQL PRIMER TRGOVINA

- Izpišite vsa polja tabele Prodajalec (projekcija) v naslednjem vrstnem redu: Mesto, ImeProdajalca, KodaProdajalca, Provizija!

```
SELECT   Mesto, ImeProdajalca, KodaProdajalca, Provizija
FROM     Prodajalec;
```

- Izpišite rang in ime vseh kupcev (projekcija) iz mesta Ljubljana (selekcija)!

```
SELECT   Rang, ImeKupca
FROM     Kupec
WHERE    Mesto='Ljubljana';
```

7.1.1. SQL PRIMER TRGOVINA

- V nekaterih primerih, pri SQL povpraševanju, dobimo kot rezultat večkraten zapis istega atributa, saj pri povpraševanju pregledamo vse zapise v tabelah. To podvajanje je nezaželeno in se mu želimo izogniti.

- Izpišite kode vseh kupcev, ki so naročili izdelek ali storitev (so zapisani v tabeli Naročilo).

```
SELECT     KodaKupca
FROM       Narocilo;
```

Kot rezultat dobimo naslednje zapise: 2008, 2001, 2007, 2003, 2008, 2002, 2004, 2006, 2004, 2006.

Vidimo, da se zapisi ponavljajo, čemur pa se želimo izogniti, zato uporabimo argument **DISTINCT**.

7.1.1. SQL PRIMER TRGOVINA

- Izpišite kode vseh kupcev, ki so naročili izdelek ali storitev (so zapisani v tabeli Naročilo), brez ponavljanja vrednosti! (uporaba argumenta `DISTINCT`)

```
SELECT     DISTINCT KodaKupca
FROM       Narocilo;
```

Kot rezultat dobimo naslednje zapise: 2008, 2001, 2007, 2003, 2002, 2004, 2006.

- Izpišite vsa naročila večja od 1000 enot!

```
SELECT     *
FROM       Narocilo
WHERE      Obseg>1000.00;
```

7.1.1. SQL PRIMER TRGOVINA

- Nekateri pomembni argumenti:
  - **AND** – Povezava pogojev, kjer morajo biti izpolnjeni vsi pogoji.
  - **OR** – Povezava pogojev, kjer mora biti izpolnjen vsaj eden.
  - **BETWEEN** – Omogoča, da specificiramo dve mejni številski vrednosti znotraj katerih noato iščemo.
  - **IN** – Z njim specificiramo besedo ali del besede in jo uporabimo v povpraševanju.
  - **LIKE** – Z pomočjo like, lahko iščemo po besedah s glede na črke.

7.1.1. SQL PRIMER TRGOVINA

- Izpišite vse kupce, ki imajo rang večji od 100 in vse kupce iz Celja ne glede na njihov rang!

```
SELECT      *
FROM        Kupec
WHERE       Rang>100
OR          Mesto='Celje';
```

ali

```
SELECT      *
FROM        Kupec
WHERE       NOT Rang<=100
OR          Mesto='Celje';
```

ali

7.1.1. SQL PRIMER TRGOVINA


```
SELECT    *
FROM      Kupec
WHERE     NOT(Rang<=100 AND Mesto!='Celje');
```

- Izpišite vsa naročila katerih obseg je večji od 1000 enot ali naročila, ki jih ni naročil kupec z kodo 2003 in niso izvršena dne 3.10.1990!

```
SELECT    *
FROM      Narocilo
WHERE     Obseg<1000
OR        NOT(DatumNarocila='3.10.1990' AND
          KodaKupca=2003)
```

7.1.1. SQL PRIMER TRGOVINA

- Izpišite vsa naročila izvedena 3. in 4. oktobra 1990!

```
SELECT      *
FROM        Narocilo
WHERE       DatumNarocila IN('3.10.1990','4.10.1990');
```

ali

```
SELECT      *
FROM        Narocilo
WHERE       DatumNarocila BETWEEN '3.10.1990' AND
            '4.10.1990';
```

7.1.1. SQL PRIMER TRGOVINA

- Izpišite imena vseh kupcev s kodo od 2000 do vključno 2100!

```
SELECT    *
FROM      Kupec
WHERE     KodaKupca BETWEEN 2000 AND 2100;
```

- Izpišite vse kupce prodajalcev Sisek in Rakus (moramo vedeti kodo prodajalcev)!

```
SELECT    *
FROM      Kupec
WHERE     KodaProdajalca IN(1001,1004);
```

7.1.1. SQL PRIMER TRGOVINA

- Izpišite vse kupce prodajalcev Sisek in Rakus (ne vemo kode prodajalcev)!

```
SELECT      *
FROM        Kupec, Prodajalec
WHERE       Kupec.KodaProdajalca =
            Prodajalec.KodaProdajalca
AND         ImeProdajalca = 'Sisek'
OR          ImeProdajalca = 'Rakus';
```

7.1.1. SQL PRIMER TRGOVINA

- Izpišite imena vseh kupcev s kodo od 2000 do vključno 2100!

```
SELECT     *
FROM       Kupec
WHERE      KodaKupca BETWEEN 2000 AND 2100;
```

7.1.1. SQL PRIMER TRGOVINA

- Izpišite vse kupce katerih ime se začne na črko P!

```
SELECT    *
FROM      Kupec
WHERE     ImeKupca LIKE 'P%';
```

- Izpišite vse kupce za katere nismo vnesli mesta!

```
SELECT    *
FROM      Kupec
WHERE     Mesto IS NULL;
```

7.1.1. SQL PRIMER TRGOVINA

- Preštejte vsa naročila izvršena 3. oktobra 1990!

```
SELECT     COUNT(*)
FROM       Narocilo
WHERE      DatumNarocila='3.10.1990';
```

- Preštejte vse različne ne NULL vrednosti mest v tabeli Kupec!

```
SELECT     COUNT(DISTINCT Mesto)
FROM       Kupec
WHERE      Mesto IS NOT NULL;
```

7.1.1. SQL PRIMER TRGOVINA

- Izpišite najmanjše naročilo vsakega kupca!

```
SELECT    KodaKupca, MIN(Obseg)
FROM      Narocilo
GROUP BY  KodaKupca;
```

- Izpišite ime prvega kupca, katerega ime se začne z črko P!

```
SELECT    MIN(ImeKupca)
FROM      Kupec
WHERE     ImeKupca LIKE 'P%';
```

7.1.1. SQL PRIMER TRGOVINA

- Izpišite najvišji rang kupca za vsako mesto!

```
SELECT     Mesto, MAX(Rang)
FROM       Kupec
GROUP BY   Mesto;
```

- Preštejte vsa naročila kupca s kodo 2002!

```
SELECT     COUNT(*)
FROM       Narocilo
WHERE      KodaKupca=2002;
```

7.1.1. SQL PRIMER TRGOVINA

- Izpišite kodo naročila in ime kupca, ki je naročilo izvršil!

```
SELECT     KodaNarocila, ImeKupca
FROM       Narocilo, Kupec
WHERE      Kupec.KodaKupca=Narocilo.KodaKupca;
```

7.1.1. SQL PRIMER TRGOVINA

- Včasih želimo imeti rezultate povpraševanja urejene glede na določen kriterij. To dosežemo z uporabo argumenta `ORDER BY`.

7.1.1. SQL PRIMER TRGOVINA

- Izpišite imena kupcev in prodajalcev za vsako naročilo, urejeno po kodi naročila!

```
SELECT     KodaNarocila, ImeKupca, ImeProdajalca
FROM       Narocilo, Kupec, Prodajalec
WHERE      Kupec.KodaKupca=Narocilo.KodaKupca
AND        Prodajalec.KodaProdajalca=
           Narocilo.KodaProdajalca
ORDER BY   KodaNarocila;
```

7.1.1. SQL PRIMER TRGOVINA

- Izpišite polja ImeKupca, ImeProdajalca in Provizija za vse kupce servisirane iz strani prodajalcev, ki imajo provizijo nad 12%!

```
SELECT     ImeKupca, ImeProdajalca, Provizija
FROM       Prodajalec, Kupec
WHERE      Prodajalec.KodaProdajalca=
           Kupec.KodaProdajalca;
AND        Provizija>0.12;
```

7.1.1. SQL PRIMER TRGOVINA

- Izračunajte obseg prodajalčeve provizije za vsako izvršeno prodajo kupcev z rangom večjim od 100!

```
SELECT    KodaNarocila, Provizija*Obseg
FROM      Prodajalec, Narocilo, Kupec
WHERE     Rang>100
AND       Narocilo.KodaKupca=Kupec.KodaKupca
AND       Narocilo.KodaProdajalca=
          Prodajalec.KodaProdajalca;
```

7.1.1. SQL PRIMER TRGOVINA

- Izpišite vse pare naročil za vsakega kupca! Izpis izloči kombinacije naročil samih s seboj in obratne izpise.

```
SELECT     ImeKupca, A.KodaNarocila, B.KodaNarocila
FROM       Narocilo A, Narocilo B, Kupec
WHERE      A.KodaKupca=B.KodaKupca
AND        A.KodaKupca=Kupec.KodaKupca
AND        A.KodaNarocila<B.KodaNarocila;
```

7.1.1. SQL PRIMER TRGOVINA

- Izpišite vsa imena in mesta kupcev z enakim rangom kot kupec Hord! Povpraševanje uporablja Hordovo kodo in ne vrednost ranga. Tako oblikovano vprašanje je uporabno tudi po spremembi Hordovega ranga.

```
SELECT    A.ImeKupca, A.Mesto
FROM      Kupec A, Kupec B
WHERE     A.Rang=B.Rang
AND       B.KodaKupca=2001;
```

7.1.1. SQL PRIMER TRGOVINA

- Izpišite vsa naročila kupca Porkaro! Poznamo le ime kupca in ne njegove kodne številke.

```
SELECT     *
FROM       Narocilo
WHERE      KodaKupca=
           (SELECT     KodaKupca
            FROM       Kupec
            WHERE      ImeKupca='Porkaro');
```

ali

7.1.1. SQL PRIMER TRGOVINA


```
SELECT     *
FROM       Narocilo
WHERE      KodaKupca IN
           (SELECT     KodaKupca
            FROM       Kupec
            WHERE      ImeKupca='Porkaro');
```

7.1.1. SQL PRIMER TRGOVINA

- Izpišite vsa imena in range vseh kupcev, ki so izvršili naročila nad poprečjem vseh naročil!

```
SELECT     DINSTINCT ImeKupca, Rang
FROM       Kupec, Narocilo
WHERE      Obseg>
           (SELECT     AVG(Obseg)
            FROM       Narocilo
            WHERE      Narocilo.KodaKupca=
                       Kupec.KodaKupca);
```

7.1.1. SQL PRIMER TRGOVINA

- Včasih želimo izračunati rezultate na osnovi vrednosti atributov, ki se nahajajo le v določenih vrsticah tabel. Da to dosežemo tvorimo skupino vrstic in z njihovimi atributi izvajamo izračune.
- Velikokrat bo potrebno narediti tudi izračune, ki jemljejo atribute za izračune iz različnih skupin, ki so opisane zgoraj in tukaj pride v poštev argument **`GROUP BY`**.
- Včasih pa želimo omejiti rezultate povpraševanja kjer uporabimo **`GROUP BY`**. To storimo z argumentom **`HAVING`**.

7.1.1. SQL PRIMER TRGOVINA

- Izpišite vsoto vseh prodaj tistih prodajalcev, ki so prodali več kot je obseg največjena naročila v tabeli Naročilo!

```
SELECT     KodaProdajalca, SUM(Obseg)
FROM       Narocilo
GROUP BY   KodaProdajalca
HAVING     SUM(Obseg)>
           (SELECT     MAX(Obseg)
            FROM       Narocilo);
```

7.1.1. SQL PRIMER TRGOVINA

- Izpišite tiste kupce, ki imajo vsoto naročil večje, kot je povprečna vrednost naročil!

```
SELECT     KodaKupca, SUM(Obseg)
FROM       Narocilo
GROUP BY   KodaKupca;
```

Povpraševanje izpiše le vsoto po posameznih kupcih, želeli pa bi imeti le tiste vsote, ki so večje od povprečja. Zato potrebujemo “začasno” tabelo – pogled, ki ga izdelamo kot (skupinske funkcije je potrebno poimeovati z aliasom – v tem primeru jo poimenujemo **Vsota**):

7.1.1. SQL PRIMER TRGOVINA


```
CREATE VIEW VsotaProdaje AS
SELECT      KodaKupca, SUM(Obseg), Vsota
FROM        Narocilo
GROUP BY    KodaKupca;
```

Sedaj lahko iz te navidezne tabele izberemo tiste zapise, ki ustrezajo pogoju.

```
SELECT      KodaKupca, Vsota
FROM        VsotaProdaje
WHERE       Vsota>
            (SELECT AVG(Vsota)
             FROM   VsotaProdaje);
```

7.1.2. SQL PRIMER SVETOVNO GLASBENO ZDRUŽENJE

- Vsebina tabel

ORKESTER			
ImeOrkestra	Mesto	Drzava	GlasbeniManager

GLASBENIK				
StevilkaGlasbenika	ImeGlasbenika	Glasbilo	LetnaPlaca	*ImeOrkestra*

IZOBRAZBA			
StevilkaGlasbenika	StopnjaIzobrazbe	Univerza	Leto

7.1.2. SQL PRIMER SVETOVNO GLASBENO ZDRUŽENJE

- Vsebina tabel

SKLADATELJ		
ImeSkladatelja	Drzava	DatumRojstva

SKLADBA		
NaslovSkladbe	*ImeSkladatelja*	Leto

SNEMANJE			
ImeOrkestra	*NaslovSkladbe*	Leto	Cena

7.1.2. SQL PRIMER SVETOVNO GLASBENO ZDRUŽENJE

- Koliko znaša skupna letna plača vseh violinistov Simfoničnega orkestra Maribor?

```
SELECT    SUM(LetnaPlaca)
FROM      Glasbenik
WHERE     ImeOrkestra='Simfonicni orkester Maribor'
AND       Glasbilo='Violina';
```

7.1.2. SQL PRIMER SVETOVNO GLASBENO ZDRUŽENJE

- V abecednem vrstnem redu, izpišite vse univerze, ki so jih obiskovali čelisti iz Indije! Vsaka se naj izpiše samo enkrat.

```
SELECT    DISTINCT Univerza
FROM      Orkester, Glasbenik, Izobrazba
WHERE     Orkester.ImeOrkestra=Glasbenik.ImeOrkestra
AND       Glasbenik.StevilkaGlasbenika=
          Izobrazba.StevilkaGlasbenika
AND       Glasbilo='Celo'
AND       Drzava='Indija'
ORDER BY Univerza;
```

7.1.2. SQL PRIMER SVETOVNO GLASBENO ZDRUŽENJE

- Koliko znaša skupna letna plača violinistov, ki so del orkestrov iz Kanade? V rezultate vključite le tiste orkestre, katerih skupna letna plača violinistov presega 150.000 dolarjev.

```
SELECT    ImeOrkestra, SUM(LetnaPlaca)
FROM      Orkester, Glasbenik
WHERE     Orkester.ImeOrkestra=Glasbenik.ImeOrkestra
AND       Drzava='Kanada'
AND       Glasbilo='Violina'
GROUP BY  ImeOrkestra
HAVING    SUM(LetnaPlaca)>150,000;
```

7.1.2. SQL PRIMER SVETOVNO GLASBENO ZDRUŽENJE

- Izpišite ime najbolje plačanega pianista!

```
SELECT    ImeGlasbenika
FROM      Glasbenik
WHERE     Glasbilo='Klavir'
AND       LetnaPlaca=
          (SELECT MAX(LetnaPlaca)
           FROM   Glasbenik
           WHERE  Glasbilo='Klavir');
```

ali

7.1.2. SQL PRIMER SVETOVNO GLASBENO ZDRUŽENJE


```
SELECT    ImeGlasbenika
FROM      Glasbenik
AND       LetnaPlaca=
          (SELECT MAX(LetnaPlaca)
           FROM   Glasbenik
           WHERE  Glasbilo='Klavir');
```

7.1.2. SQL PRIMER SVETOVNO GLASBENO ZDRUŽENJE

- Izpišite ime najbolje plačanega pianista, ki sodeluje pri enem izmed orkestrov v Avstraliji!

```
SELECT    ImeGlasbenika
FROM      Glasbenik, Orkester
WHERE     Glasbenik.ImeOrkestra=Orkester.ImeOrkestra
AND       Glasbilo='Klavir'
AND       Drzava='Avstralija'
AND       LetnaPlaca=
          (SELECT MAX(LetnaPlaca)
           FROM  Glasbenik, Orkester
           WHERE Glasbenik.ImeOrkestra=Orkester.ImeOrkestra
           AND   Glasbilo='Klavir'
           AND   Drzava='Avstralija');
```

7.1.3 PRIPOROČILA ZA PISANJE SQL POVPRAŠEVANJ

- Premislite, kaj naj bi dobili kot rezultat povpraševanja. Gleda na to zapišite potrebne atribute in funkcije, ki jih boste uporabili v stavku SELECT. Čeprav se vam ti podatki zdijo očitni že na prvi pogled je zelo priporočljivo, da to storite.
- Iz tega boste lahko tudi ugotovili ali je potrebno v povpraševanju uporabiti GROUP BY ali ugnezden stavek. Če je odgovor pritrdilen je priporočljivo, da poleg SELECT stavka osnovnega povpraševanja napišete še GROUP BY stavek in SELECT stavek ugnezdenega povpraševanja.

7.1.3 PRIPOROČILA ZA PISANJE SQL POVPRAŠEVANJ

- Določite katere tabele iz podatkovne baze boste pri povpraševanju potrebovali in njihova imena zapišite v stavek FROM.
- Vključite le tiste, ki so resnično potrebne za povpraševanje. Včasih nas pri tem lahko kaj zavede. Na primer, če potrebujete atribut, ki je primarni ključ neke tabele se lahko zgodi, da boste takoj vključili to tabelo v povpraševanje. Preden to storite je koristno pogledati ali enak atribut morda nastopa kot tuj ključ v kateri izmed tabel, ki so že vključene v povpraševanje. Če nastopa, tabele, kjer je atribut primarni ključ ne rabimo vključiti v povpraševanje, razen če seveda iz nje potrebujemo še kakšen drug atribut razen primarnega ključa.

7.1.3 PRIPOROČILA ZA PISANJE SQL POVPRAŠEVANJ

- Če povpraševanje vključuje več tabel, je potrebno le te najprej povezati. To storite v stavku WHERE z enačenjem ustreznih atributov.
- Ko ste to storili lahko nadaljujete s postavljanjem pogojev v stavku WHERE, ki morajo biti izpolnjeni v povpraševanju.

7.1.3 PRIPOROČILA ZA PISANJE SQL POVPRAŠEVANJ

- Nato nadaljujte z dopolnjevanjem pogojev v WHERE, GROUP BY stavku in morebitnih ugnezdenih povpraševanjih.

7.1.3 PRIPOROČILA ZA PISANJE SQL POVPRAŠEVANJ

- Če ste začetnik pri pisanju SQL stavkov imate pa nekaj izkušenj s programiranjem vas bo morda zamikalo, da bo združevanje tabel nadomestili z ugnezdenimi povpraševanji. Tega ne storite. Za to obstajata dva razloga:
  - združevanja so bistveni del koncepta relacijskih podatkovnih baz,
  - pisanje več stopenj ugnezdenih povpraševanj pa lahko vodi do tega, da hitreje naredimo in nato tudi spregledamo morebitno napako, pojavijo pa se lahko tudi težave pri razhroščevanju (debugging).

7.2. QUEL – QUERY LANGUAGE

- Rojstvo:
  - zgodnja 70-ta leta,
  - University of California, Berkeley.
- Projekt:
  - INGRES - International Graphics and Retrival System.
- Osnova:
  - n-terični relacijski račun.
- Lastnosti:
  - interaktiven povpraševalni jezik,
  - možna vključitev v programski jezik,
  - ne dopušča vgnezdenega povpraševanja.

7.2. QUEL – QUERY LANGUAGE

- Kaj zmore:
  - ob povpraševanju omogoča:
    - vnos (append),
    - popravljanje (replace) in
    - brisanje (delete) n-teric.

- Osnovna struktura vprašanja:

```
range of t1 is r1
range of t2 is r2
...
range of tm is rm
retrieve (ti1.Aj1, ti2.Aj2, …, tin.Ajn)
where P
```

7.2. QUEL – QUERY LANGUAGE

- Osnovna struktura vprašanja:
  - range of: definira n-terične spremenljivke ti v relacijah ri,
  - retrieve: ima podobno funkcijo kot selekcija, Ajk je atribut,
  - where: vsebuje pogoj.

- Ekvivalentni zapis:
  - $\{t \mid \exists t1 \in r1, t2 \in r2, \ldots, tm \in rm ($
    $t[ri1.Aj1] = ti1[Aj1] \wedge$
    $t[ri2.Aj2] = ti2[Aj2] \wedge \ldots \wedge$
    $t[rin.Ajn] = tin[Ajn] \wedge$
    $P(t1, t2, \ldots, tn))\}$

7.3. QBE – QUERY BY EXAMPLE

- Rojstvo:
  - zgodnja 70-ta leta,
  - IBM raziskovalni laboratorij.
- Osnova:
  - domenski relacijski račun.
- Lastnosti:
  - dvodimenzionalna sintaksa,
  - lahek za učenje,
  - vprašanja so oblikovana v obliki primerov.

7.3. QBE – QUERY BY EXAMPLE

- Kaj zmore:
  - ob povpraševanju omogoča:
    - vnašanje (I.),
    - popravljanje (U.) in
    - brisanje (D.) n-teric.

- Osnovna struktura vprašanja:
  - ogrodje tabele, ki predstavlja relacijsko shemo in ga napolnimo s “primerkom vrstice”,
  - “primerek vrstice” (angl. Example row) sestavljajo konstante in “primerki elementov” domenske spremenljivke (_X).

7.3.1. QBE PRIMER BANKA

PODRUŽNICA	**NazivPodružnice**	**Sredstva (€)**	**Mesto**
	Center	20.000	Maribor
	Tabor	10.000	Maribor
	Bežigrad	35.000	Ljubljana

STRANKA	**PriimekStranke**	**Naslov**	**Mesto**
	Novak	Cankarjeva 19	Maribor
	Kos	Štantetova 2	Maribor
	Kranjc	Dunajska 43	Ljubljana
	Kovač	Mariborska 23	Celje

7.3.1. QBE PRIMER BANKA

POLOG	NazivPodružnice	ŠtevilkaRačuna	PriimekStranke	ZnesekPologa (€)
	Center	2323987	Kos	500
	Tabor	2323988	Novak	100
	Bežigrad	1245676	Kovač	50
	Tabor	2323987	Kos	100

POSOJILO	NazivPodružnice	ŠtevilkaPosojila	PriimekStranke	ZnesekPosojila (€)
	Tabor	1	Novak	200
	Tabor	2	Novak	100
	Bežigrad	3	Krajnc	1000

7.3.1. QBE PRIMER BANKA

- Osnovna struktura vprašanja
  - Ogrodje tabele, ki predstavlja relacijsko shemo napolnimo s “primerkom vrstice”.

POLOG	**NazivPodružnice**	**ŠtevilkaRačuna**	**PriimekStranke**	**ZnesekPologa (€)**

  - Primerke vrstice pri tem sestavljajo:
    - konstante in
    - “primerki elementov” domenske spremenljivke _X.

7.3.1. QBE PRIMER BANKA

- Izpišite stranke, ki imajo bančni račun v podružnici Centrala.

POLOG	**NazivPodružnice**	**ŠtevilkaRačuna**	**PriimekStranke**	**ZnesekPologa (€)**
	Centrala		P._X	

- Zapisano vprašanje od sistema zahteva, da poišče vse n-terice tabele **POLOG**, ki imajo za **NazivPodružnice** vrednost Center in za vsako takšno vrstico izpiše vrednost spremenljivke **X**, v našem primeru je to **PriimekStranke**. Pri tem **P.** pomeni print oziroma izpis.

7.3.1. QBE PRIMER BANKA

- Zapis je ekvivalenten zapisu domenskega n-teričnega računa:

  {<x> | ∃ NazivPodružnice, ŠtevilkaRačuna, ZnesekPologa

  (NazivPodružnice, ŠtevilkaRačuna, ZnesekPologa ∈ POLOG IN NazivPodružnice="Center") }

- **Opomba:**
- Če v vprašanju nastopa le ena spremenljivka, lahko njen zapis izpustimo (zapišemo brez _X)

POLOG	**NazivPodružnice**	**ŠtevilkaRačuna**	**PriimekStranke**	**ZnesekPologa (€)**
	Center		P.	

7.3.1. QBE PRIMER BANKA

- QBE, za razliko od SQL, že sam izloči ponavljajoče vrstice. Če tega ne želimo v povpraševanje napišemo besedo ALL.

POLOG	**NazivPodružnice**	**ŠtevilkaRačuna**	**PriimekStranke**	**ZnesekPologa (€)**
	Center		P.ALL.	

7.3.1. QBE PRIMER BANKA

- Izpišite imena podružnic, ki **niso** locirana v Celju.

PODRUŽNICA	**NazivPodružnice**	**Sredstva (€)**	**Mesto**
	P.		¬ Celje

- ¬ pomeni zanikanje. Iščemo torej vse podružnice, ki se **NE** nahajajo v Celju.

7.3.1. QBE PRIMER BANKA

- Izpišite stranke, ki imajo bančni račun **hkrati** v poslovalnici Center in v poslovalnici Tabor.

POLOG	**NazivPodružnice**	**ŠtevilkaRačuna**	**PriimekStranke**	**ZnesekPologa (€)**
	Center		P._X	
	Tabor		_X	

- Pri povpraševanju iščemo enak **PriimekStranke** pri čemer je enkrat **NazivPodružnice** Center, drugič pa Tabor. Izpišemo torej tiste, ki imajo račun v poslovalnici Center **IN** poslovalnici Tabor.

7.3.1. QBE PRIMER BANKA

- Izpišite stranke, ki imajo bančni račun v podružnici Center **ali** v podružnici Tabor **ali** v obeh.

POLOG	**NazivPodružnice**	**ŠtevilkaRačuna**	**PriimekStranke**	**ZnesekPologa (€)**
	Center		P._X	
	Tabor		P._Y	

- Pri povpraševanju iščemo stranke, ki imajo račun v **eni**, v **drugi** pa v **obeh** navedenih poslovalnicah in vse izpišemo.

7.3.1. QBE PRIMER BANKA

- Če želimo izpis celotne vrstice, umestimo P. v vse stolpce oziroma ga zapišemo pod ime tabele.

POLOG	**NazivPodružnice**	**ŠtevilkaRačuna**	**PriimekStranke**	**ZnesekPologa (€)**
P.				

7.3.1. QBE PRIMER BANKA

- Izpišite številko računa, katerega Znesek pologa je večji od 200 €.

POLOG	**NazivPodružnice**	**ŠtevilkaRačuna**	**PriimekStranke**	**ZnesekPologa (€)**
		P.		>200

- QBE dovoljuje uporabo **matematičnih operatorjev** :
  - <, <=, =, >, >= in ¬.

7.3.1. QBE PRIMER BANKA

- Katere stranke so opravile pologe v isti podružnici kot gospod Kos?

POLOG	**NazivPodružnice**	**ŠtevilkaRačuna**	**PriimekStranke**	**ZnesekPologa (€)**
	_X		Kos	
	_X		P._Y	

- V povpraševanju iščemo iste podružnice kot tiste v katerih je gospod Kos opravil polog. Nato izpišemo ustrezne priimke strank.

7.3.1. QBE PRIMER BANKA

- Izpišite priimek in mesto stranke, ki je prejela posojilo v podružnici Center.

POSOJILO	**NazivPodružnice**	**ŠtevilkaPosojila**	**PriimekStranke**	**ZnesekPosojila (€)**
	Center		_X	

STRANKA	**PriimekStranke**	**Naslov**	**Mesto**
	P._X		P._Y

- QBE omogoča postavljanje vprašanj, ki povežejo več relacij (analogija z kartezičnim produktom oz. Naravnim stikom). Povezavo omogočajo domenske spremenljivke.

7.3.1. QBE PRIMER BANKA

- Izpišite stranke, ki imajo v podružnici Center odprt račun, vendar v tej podružnici nimajo posojila.

POLOG	**NazivPodružnice**	**ŠtevilkaRačuna**	**PriimekStranke**	**ZnesekPologa (€)**
	Center		P._X	

POSOJILO	**NazivPodružnice**	**ŠtevilkaPosojila**	**PriimekStranke**	**ZnesekPosojila (€)**
¬	Center		_X	

- Operator ¬ pod imenom relacije pomeni "**ne obstaja**". Operator pod imenom atributa je ekvivalenten **≠**.

7.3.1. QBE PRIMER BANKA

- Izpišite številke bančnih računov na katerih je znesek pologa med 200 in 1000 €.

POLOG	NazivPodružnice	ŠtevilkaRačuna	PriimekStranke	ZnesekPologa (€)
		P.		_X

Pogoj
_X>=200
_X<=1000

- Za predstavitev pogojev, ki jih ne moremo zapisati direktno v ogrodja tabel, uporabljamo **pogojne okvirje** (condition box).

7.3.1. QBE PRIMER BANKA

- Katere podružnice imajo večja sredstva kot podružnica iz Celja?

PODRUŽNICA	**NazivPodružnice**	**Sredstva (€)**	**Mesto**
	P._X	_Y	
		_Z	Celje

Pogoj
_Y>_Z

- Pogoj ima lahko obliko kompleksnega aritmetičnega izraza, npr. sredstva podružnice naj bo 2-krat večja kot sredstva podružnice iz Celja. (v pogoj zapišemo:

7.3.1. QBE PRIMER BANKA

- Če je rezultat povpraševanja **nova** rezultirajoča tabela, moramo zagotoviti njen nastanek oz. oblikovanje tako, da oblikujemo relacijsko shemo in v njo vnesemo rezultate.
- Analogija z relacijsko algebro (združitev in projekcija).

7.3.1. QBE PRIMER BANKA

- Za stranke, ki imajo račun v podružnici Center izpišite priimek, mesto in številko računa.

POLOG	**NazivPodružnice**	**ŠtevilkaRačuna**	**PriimekStranke**	**ZnesekPologa (€)**
	Center	_Z	_X	

STRANKA	**PriimekStranke**	**Naslov**	**Mesto**
	_X		_Y

REZULTAT	**PriimekStranke**	**ŠtevilkaRačuna**	**Mesto**
P.	_X	_Z	_Y

7.3.1. QBE PRIMER BANKA

- V naraščajočem vrstnem redu razvrstite stranke podružnice Centrala.

POLOG	**NazivPodružnice**	**ŠtevilkaRačuna**	**PriimekStranke**	**ZnesekPologa (€)**
	Center		P.AO.	

- QBE nam omogoča razvrščanje v naraščajočem – **AO** oz. Padajočem – **DO** vrstnem redu.

7.3.1. QBE PRIMER BANKA

- Razvrščanje lahko izvedemo po posameznih atributih, npr. Stranke banke razvrstimo v naraščajočem vrstnem redu, njihova pripadajoče zneske pologa pa v padajočem vrstnem redu:

POLOG	**NazivPodružnice**	**ŠtevilkaRačuna**	**PriimekStranke**	**ZnesekPologa (€)**
	Center		P.AO(1).	P.DO(2).

- (1) pomeni, da se najprej razvrstijo priimki strank, nato pa se razvrstijo zneski pologa, torej (2).

7.3.1. QBE PRIMER BANKA

- Izpišite skupen znesek pologa vseh računov gospoda Kosa.

POLOG	**NazivPodružnice**	**ŠtevilkaRačuna**	**PriimekStranke**	**ZnesekPologa (€)**
			Kos	P. SUM.ALL.

- QBE omogoča uporabo **funkcij** SUM, AVG, MIN, MAX, CNT (štetje). Ti operatorji so vedno povezani z operatorjem ALL.

7.3.1. QBE PRIMER BANKA

- Iščemo povprečje zneska pologa vseh podružnic, ki imajo povprečen znesek pologa večji od 200 €.

POLOG	**NazivPodružnice**	**ŠtevilkaRačuna**	**PriimekStranke**	**ZnesekPologa (€)**
	P.G.			P.AVG.ALL._X

Pogoj
AVG.ALL._X >200

- **G** zagotavlja oblikovanje skupin nad katerimi izvajamo posamezne operacije (analogija z GROUP BY v jeziku SQL).

7.4. PRIMERJAVA POVPRAŠEVALNIH JEZIKOV

- **SQL :QUEL**
  - QUEL in SQL temeljita na n-teričnem relacijskem računu (SQL tudi na relacijski algebri),
  - SQL omogoča n vgnezditev (vgnezditev na n nivojih) SELECT … FROM … WHERE; QUEL dovoljuje le en nivo,
  - SQL dovoljuje le eno grupiranje (GROUP BY … HAVING) na vprašanje. QUEL omogoča z n-kratno uporabo stavka BY … WHERE znotraj RETRIEVE in WHERE oblikovanje n skupin znotraj enega vprašanja,
  - SQL dovoljuje uporabo operacij iz relacijske algebre (UNION),
  - QUEL dovoljuje uporabo začasnih datotek za zagotovitev rezultatov vprašanja, medtem ko moramo v okviru SQL s pomočjo stavka CREATE oblikovati primerno tabelo,
  - SQL je standard.

7.4. PRIMERJAVA POVPRAŠEVALNIH JEZIKOV

- **SQL, QUEL : QBE**
  - vprašanja v QBE so podana v obliki primerov; uporablja dvodimenzionalno sintakso,
  - QBE temelji na domenskem relacijskem računu,
  - QBE je uporabniško prijaznejši od SQL in QUEL,
  - QBE pod QMF (Query Management Facility) ni relacijsko kompleten (eksplicitna uporaba kvantifikatorja $\exists$ in $\forall$).

8. ADMINISTRIRANJE PODATKOVNE BAZE

- Za pregled in nadzor nad aktivnostmi, povezanimi z življenskim krogom podatkovne baze in pripadajočimi podatki ZOS (zaključenega organiziranega sistema) sta odgovorna administrator podatkov in administrator podatkovne baze.

Koraki življenjskega kroga PB	**Glavna vloga**	**Stranska vloga**
Načrtovanje PB	AP	APB
Definicija sistema	AP	APB
Zbiranje in analiza zahtev	AP	APB
Konceptualno oblikovanje PB	AP	APB
Izbor SUPB	APB	AP
Logično oblikovanje PB	AP	APB
Fizično oblikovanje PB	APB	AP
Implementacija	APB	AP
Testiranje	APB	AP
Polnjenje PB	APB	AP
Operacijsko vzdrževanje	APB	AP

8.1. ADMINISTRATOR PODATKOV

- Upravljanje podatkovnih virov, vključno z načrtovanjem PB, razvoj in vzdrževanje standardov, skrb za politiko in procedure ter konceptualno in logično oblikovanje PB so zadolžitve administratorja podatkov.

Funkcije administratorja podatkov:

- izbor primernih orodij,
- pomoč pri razvoju strategije IS s povdarkom na študiji izvedljivosti in načrtovanju podatkovne baze,
- oblikovanje ogrodnega podatkovnega modela ali modela ZOR:
  - funkcionalnost PB,
- določitev podatkovnih zahtev,
- določitev standarda podatkov in pripadajočih formatov.

8.1. ADMINISTRATOR PODATKOV

<u>Funkcije administratorja podatkov:</u> (nadaljevanje)

- določitev obsega podatkov skupaj z naraščanjem le-teh,
- določitev vzorcev uporab in frekvence uporabe podatkov,
- določitev poti dostopa do podatkov,
- začetno konceptualno in logično oblikovanje,
- sodelovanje z administratorjem podatkovne baze in sodelavci za zagotovitev upoštevanja vseh zahtev v PB,
- izobraževanje uporabnikov glede podatkovnih standardov in zakonskih odgovornosti,
- sodelovanje z oblikovalci PB,
- zagotavljanje dokumentacije,
- upravljanje podatkovnega slovarja,
- povezovanje z uporabniki za določitev morebitnih novih zahtev.

8.2. ADMINISTRATOR PODATKOVNE BAZE

- Upravljanje fizične izvedbe sistema podatkovne baze, vključno z fizičnim oblikovanjem podatkovne baze, implementacijo, postavitvijo sistema varovanja in nadzora integritete, spremljanje učinkovitosti sistema ter reorganizacije le-tega.

Funkcije administratorja podatkovne baze:

- ocenitev SUPB-jev in priporočilo za izbor,
- izvedba fizičnega oblikovanja podatkovne baze,
- implementacija fizičnega modela PB,
- začetno polnjenje podatkovne baze,
- definiranje integritetnih omejitev,

8.2. ADMINISTRATOR PODATKOVNE BAZE

Funkcije administratorja podatkovne baze: (nadaljevanje)

- postavitev sistema varovanja,
- sodelovanje z oblikovalci aplikacij,
- razvoj strategije testiranja,
- izobraževanje uporabnikov,
- prevzem aplikacije pred uporabo,
- spremljanje učinkovitosti sistema,
- ugotavljanje primernosti PB,
- zagotavljanje rezervnih kopij,
- zagotavljanje mehanizmov ponovne vzpostavitve,
- dokumentiranje,
- spremljanje novosti na področju programske in strojne opreme,
- prehod na nove verzije.

8.3. PRIMERJAVA

Zaželene sposobnosti AP in APB:

Upravljavske:

- razumevanje poslovnih procesov,
- sposobnost koordinacije in planiranja,
- analitične sposobnosti,
- sposobnosti pogajanja,
- sposobnost sprejemanja odločitev,
- pisno in ustno komuniciranje,
- sposobnost vodenja in motiviranja,
- sposobnost prenašanja pritiska in strasa ob pri upravljanju sprememb.

8.3. PRIMERJAVA

Tehnične:

- poznavanje podatkovnih obdelav,
- znanja strukturnih metodologij:
  - diagrami pretoka podatkov,
  - strukturni diagrami,
  - oblikovanje programskih produktov,
- oblikovanje PB,
- modeliranje PB,
- upravljanje podatkovnega slovarja.