PODATKOVNE BAZE

1. OSNOVNI POJMI

1.1. DEFINICIJE PB

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

2. 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.

3. *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*).

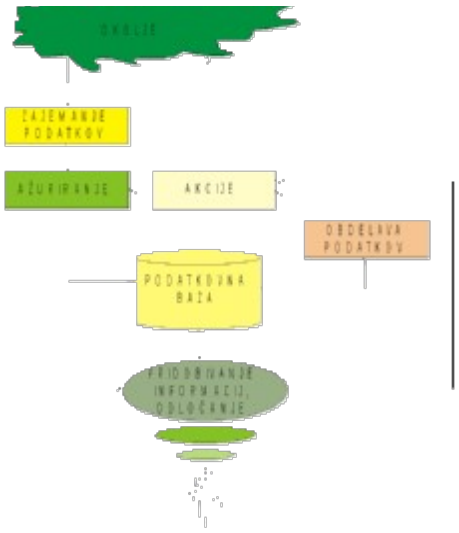
4. *P. Rob, C. Cornel: Database Systems*
   PB je namenjena podpori sprejemanja poslovnih odločitev na vseh nivojih organizacije.

5. *T. J. Teorey: Database Systems*
   PB je zbirka medsebojno povezanih shranjenih podatkov, ki zadovoljujejo potrebe različnih uporabnikov znotraj ene ali več organizacij.

6. *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.

7. *IBM: IMS/VS*
   PB je neredundantna zbirka vzajemno povezanih podatkov, ki se uporabljajo za izvajanje ene ali več aplikacij.

8. *T. Mohorič: Uvod v podatkovne baze*
   PB je model okolja, ki služi kot osnova za sprejemanje odločitev in izvajanje akcij.



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).

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.

- → *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

I – informacija

P – podatek (vpliva na informacijo)

S – sprejemna struktura (razlaga podatke)

t - čas

Pogosto se uporabljata kot sinonima, kar ni dopustno.

1.5. OSTALE 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

1.6.1. PODATKOVNA REVOLUCIJA

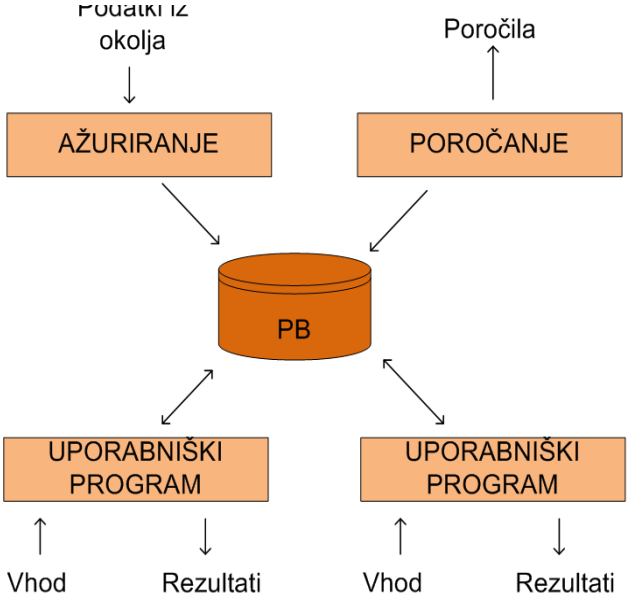
Računalnipko podprto shranjevanje je eden izmed možnih načinov zapisovanja in shranjevanja podatkov – uporaba podatkovnega sistema.

→ **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. Prvo obdobje računalništva (računalnik, strojna oprema)
2. Drugo obdobje (programska revolucija – program, programski jezik)
3. Tretje obdobje (podatkovna revolucija – podatki, podatkovna baza)

I = i(P, S,



1.6.2. PRELOMNICE V ZGODOVINI PB

60. LETA

Rojstvo PB z IMS – hierarhični SUPB proizvajalca IBM

Med 1965 – 1971 aktivnosti delovne skupine DBTG (DataBase Task Group) v okviru CODASYL (Conference on Data Systems Languages), ki se zaključi z mrežnim SUPB.

Hierarhični in mrežni SUPB 1. generacija SUPB

70. LETA

1970 – Ted Codd objavi temeljni članek, ki omogoča uvedbo relacijskega podatkovnega modela

1976 - prestavitev entitetno-relacijskega (E-R) podatkovnega modela (P. Chen).

1979 – popravki T. Codda-a

V pozni sedemdesetih letih se pojavi System R, eden prvih predstavnikov relacijskega SUPB.

Relacijski SUPB = 2. generacija SUPB

80. LETA

V zgodnjih 80-ih letih se pojavijo prvi objektno-orientirani podatkovni modeli.

Rodi se 3. generacija SUPB.

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 je srce/jedro 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.

Izvajalec oblikovanja – oblikovalec PB, nespecialist

Posledice slabega oblikovanja: slabe odločitve

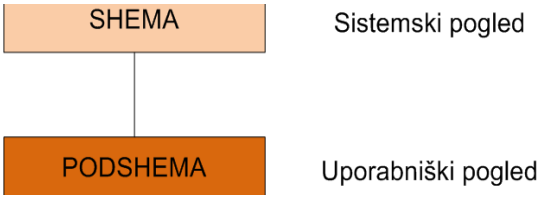
Posledice dobrega oblikovanja:

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

2.1. ARHITEKTURA PB

PB oblikujemo za izbraz ZOS, predstavlja jedro informacijskega sistema.

1971 – DBTG (DATABASE TASK GROUP): DVONIVOJSKA ARHITEKTURA.

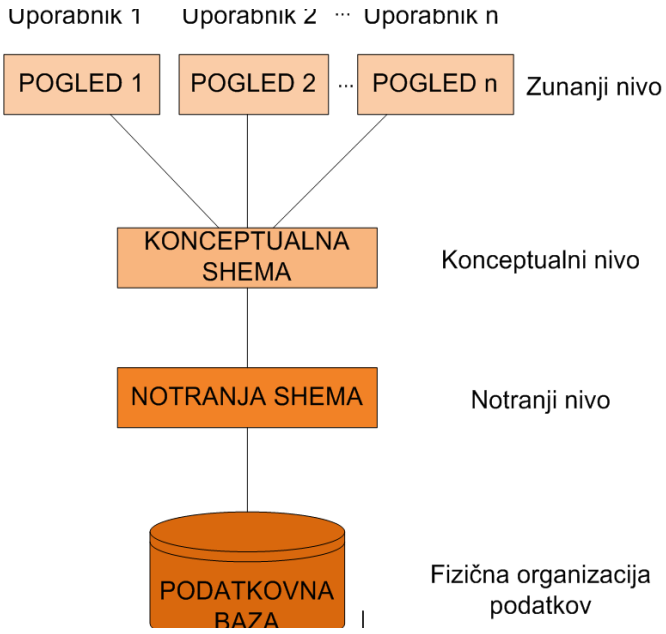


1975 – ANSI-SPARC TRINIVOJSKA ARHITEKTURA

ANSI – American National Standard Institute

SPARC – Standard Planning And Requirements Comitee

Ni standarda.



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.

Zunanji nivo logična podatkovna neodvisnost

Uporabniški pogled na PB je predstavljen z:

→ entitetami, atributi, relacijami lastnega realnega okolja
→ različnimi pogledi na podatke.

Konceptualni nivo fizična podatkovna neodvisnost

Skupen pogled na PB predstavljen z:

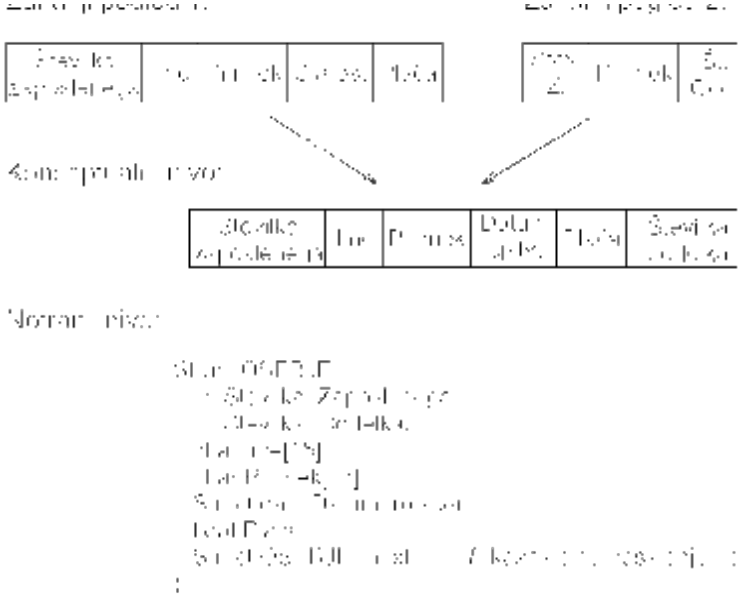
→ vsemi entitetami, relacijami in pripadajočimi atributi,
→ omejitvami,
→ semantičnimi informacijami o podatkih,
→ informacijami, vezanimi na varnost in integriteto.

Notranji nivo

Fizična predstavitev PB na računalniku. Podan opis, kako so podatki shranjeni v PB:

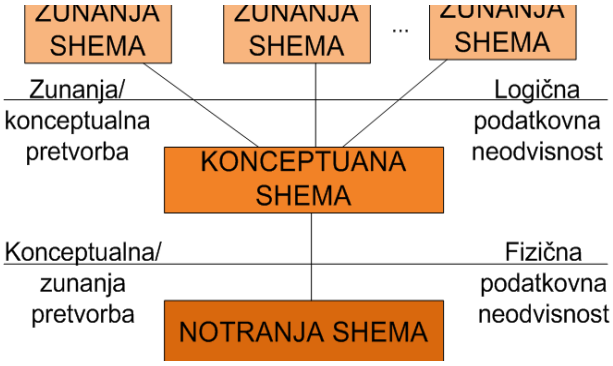
→ dodelitev spomina za podatke in indekse,
→ opis zapisov skupaj s podatki,
→ enkripcijske tehnike in stiskanje podatkov.

Za fizično organizacijo podatkov je zadolžen operacijski sistem od podpori SUPB.



2.2.1 NEODVISNOST V ANSI-SPARC ARHITEKTURI

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



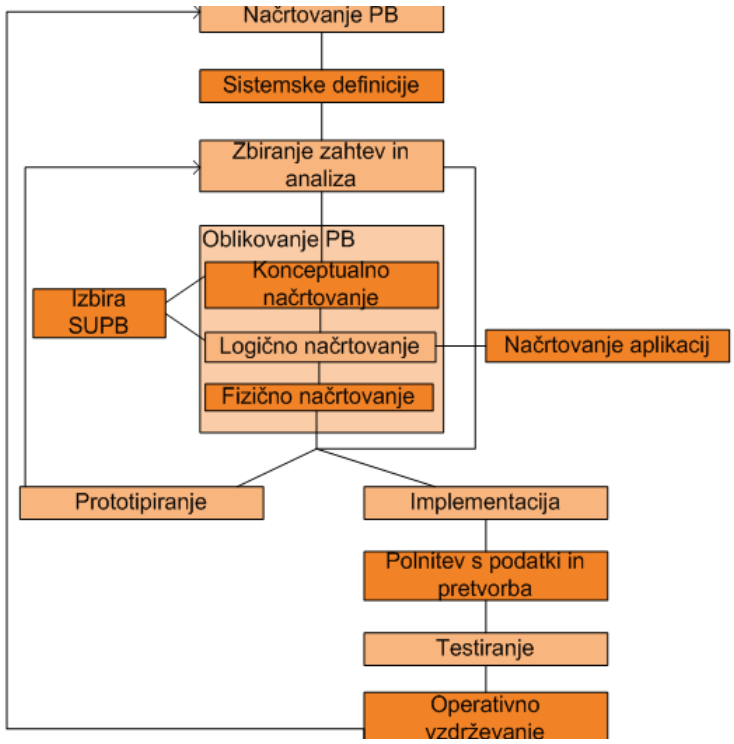
Logična podatkovna neodvisnost

Zunanji modeli so imuni na spremembe v konceptualnem modelu.

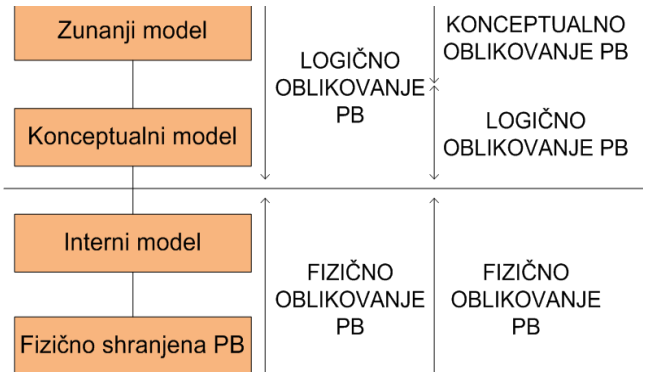
Fizična podatkovna neodvisnost

Konceptualni model je imun na spremembe v notranjem modelu.

2.3. ŽIVLJENSKI KROG PB



2.4. PREGLED OBLIKOVANJA PB



Oblikovanje PB praviloma poteka v 3 fazah:

1. **Konceptualno oblikovanje**
   Oblikujemo model za informacijsko uporabo izbranega delovno zaključenega organiziranega sistema (angl. enterprise), ki je popolnoma neodvisen od logičnega in fizičnega oblikovanja.
2. **Logično oblikovanje**
   Oblikujemo model izbranega delovno zaključenega organiziranega sistema za ciljno skupino sistema za upravljanje s podatkovno bazo - SUPB.
3. **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

Konceptualni model – izhodiščna točka oblikovanja, katere rezultat je abstrakten in splošen opis realnosti.

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.

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.
- → 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:
    - o "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".

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)

Namen konceptualnega modeliranja je doseči cilj:

- → vse, kar je potrebno, je tu,
- → vse, kar je tu, je potrebno.

Koraki oblikovanja konceptualnega modela PB:

- o podatkovna analiza in zbiranje zahtev,
- o oblikovanje E-R modela,
- o 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.

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).

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"

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.

ENTITETA

→ ni podatek
→ je neodvisni podatkovni objekt (fizični, konceptualni) ZOS, ki je po definiciji nosilec podatkov. Lastnosti entitet opisujejo atributi.

ENTITETNI TIP

→ je množica entitet, ki jih opisujejo isti atributi (Ime, priimek, leto vpisa, vpisna št.).

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.

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.
→ Atributi niso enakovredni.
→ Obvezen atribut je identifikator.
→ Sestavljen atribut – več atributov, ki tvorijo celoto (ulica in hišna številka).

KLJUČ (IDENTIFIKATOR)

Vodilni atribut, ki omogoča identifikacijo posamezne entitete. Ločimo:

→ kandidacijski (ima vlogo kandidata, lahko prevzame vlogo ključa – EMŠO, davčna)
→ primarni (izbran za dostop, podčtran ali #)
→ sekundarni (izbrali bi ga, če nebi bilo primarnega)
→ sestavljen (v opisu je več podatkov – ulica, hišna številka) in
→ zunanji (tuji) ključ (povezuje tabele, je ključ druge entitete, lahko e pojavi v napi, kot navaden atribut).

DOMENA ATRIBUTA

→ je množica dovoljenih vrednosti za posamezen atribut.

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)

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

OBVEZNA IN OPCJSKA UDELEŽENOST ENTITETE

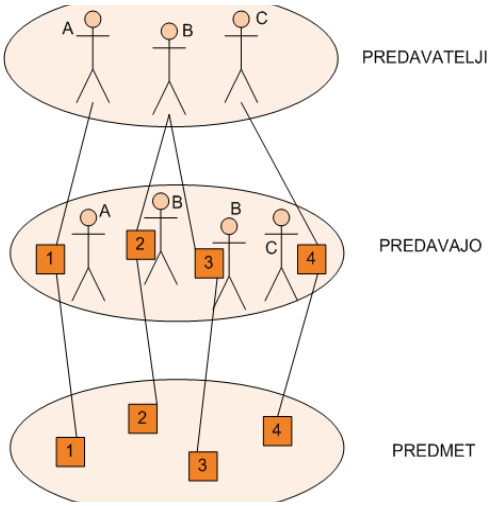
Udeleženost entitete v relaciji je lahko opcijska (angl. optional) ali pa obvezna (angl. mandatory). Opisuje jo minimalna kardinalnost, med tem ko udeleženost v tem primeru opisuje maksimalna kardinalnost:

(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)

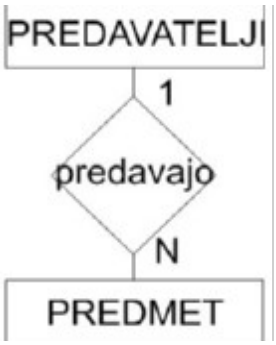
2.6.3. NOTACIJE E-R DIAGRAMA



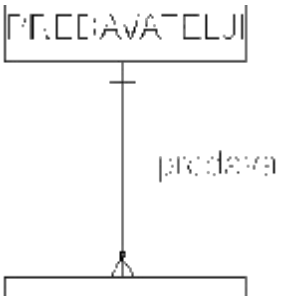
Notacija entitete (tipa)

Notacija šibke entitete

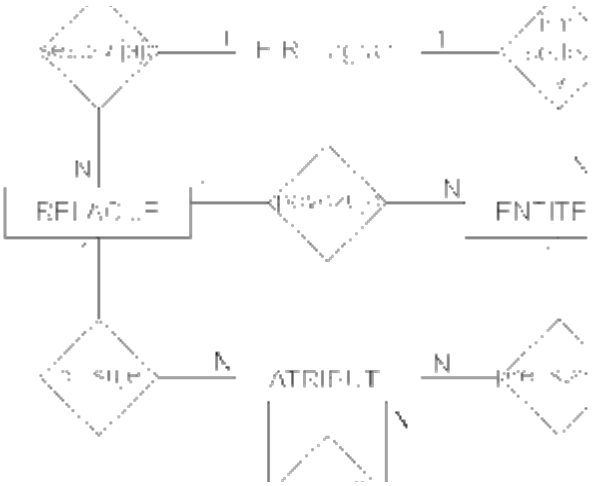
CHENOVA NOTACIJA



SRAČJA (JAMES MARTINOVA) NOTACIJA



2.6.5. STANDARDI ZA OBLIKOVANJE E-R DIAGRAMA



E-R diagram 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).

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.

Praktični nasveti za oblikovanje E-R diagrama

<u>Odkrivanje entitet</u>

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

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

→ **Kraji**

  o ki so v uporabi ljudi ali so v njih nameščeni predmeti (mesta, pisarne, države...).

→ **Organizacije**

  o 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).

→ **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).

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

POZOR □ Entiteta *izbranega* zaključenega organiziranega sistema ni nujno entiteta *kateregakoli drugega* zaključenega organiziranega sistema

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

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.

Primer:

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

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.

DELAVEC
maticna_st
Ime
Priimek
Naslov
Rojstni_datum
OD

ODDELEK
st_oddelka
Ime_oddleka
Lokacija

PROJEKT
#st_projekta
Naziv_projekta
Lokacija_projekta

SODELUJE
st_ur

ZAPOSLUJE
Datum_zaposlitve

VODI
Datum_nastopa_funkcije

IZVAJA
Zacetek_projekta
Predviden_zakljucek

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.

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 B (A funkcionalno določa B), če za vsako vrednost A-ja v E obstaja natanko ena vrednost B-ja.

Je odnos ,ed ključnimi in neključnimi atributo v entiteti. Opisuje atribute (odnose) v posamezni entiteti.s

Običajno funkcionalno odvisnost definiramo med množicami atributov znotraj entitete E.

Formalna predstavitev funkcionalne odvisnosti

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


R ( ).

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

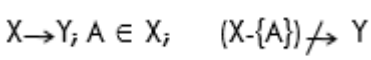
če je t1[X] = t2[X] t1[Y] = t2[Y]

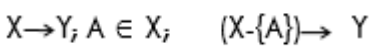
Popolna funkcionalna odvisnost:

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.

Popolna funkcionalna odvisnost



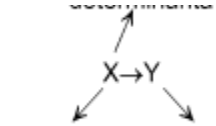


Delna funkcionalna odvisnost

Primer:

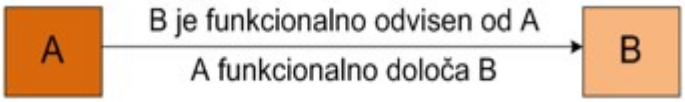
- št_del, št_proj → št_ur
- št_del ↛ št_ur popolna funkcionalna odvisnost
- št_proj ↛ št_ur

- št_del, št_proj → priimek_del
- št_del → priimek_del delna funkcionalna odvisnost
- št_proj ↛ priimek_del

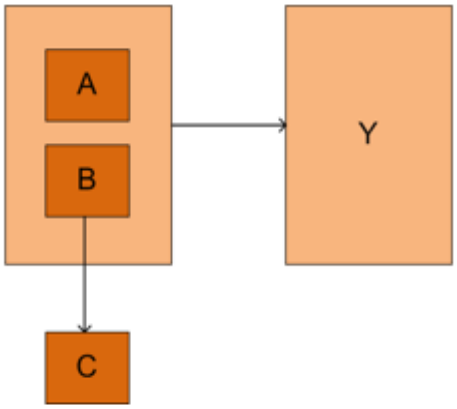


Determinanta funkcionalne odvisnosti predstavlja atribut ali skupino atributov z leve strani funkcionalne odvisnosti.

Diagram funkcionalnih odvisnosti



X={A, B}, X→Y ∧ B→C



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 Y

$X_1 \subset X$, X_1 Y

→ Študent (ime, priimek, vpisna številka)
→ opravljen izpit (vpisna številka, šifra predmeta, ocena, datum)

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.

X Y

$X_1 \subset X$, X_1 Y

Izpit (vpisna številka, šifra predmeta, naziv, ime, priime, ocena...)

3.1.1 LASTNOSTI FUNKCIONALNE ODVISNOSTI

Osnovne lastnosti:

1. **Enoličnost določa F.O.**

Za dano domeno in kodomeno obstaja največ ena funkcionalna odvisnost.

f: A B

g: A B

E (A,B)

f = g

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

2. **Projektivnost**

Množica funkcionalno določa vse svoje podmnožice.

ABC A

ABC B

ABC C

$X \subset Y \Rightarrow Y \to X$

3. **Aditivnost**

$X \to Y \;\; X \to / \Rightarrow X \to Y /$

4. **Tranzitivnost**

Temelj, osnova za 3NO.

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

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

Izpeljane lastnosti:

1. **Distributivnost**

$$X \rightarrow YZ \Rightarrow X \rightarrow Y \text{ in } X \rightarrow Z$$

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

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

2. **Pseudotranzitivnost**

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

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

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

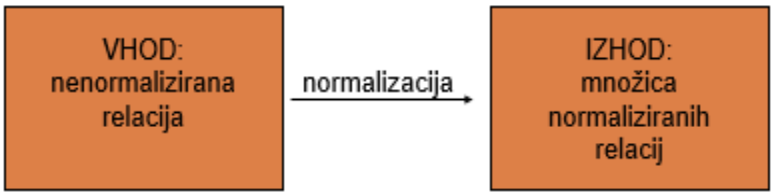
3. **Razširitev**

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

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

3.2. NORMALNE OBLIKE

Normalizacija: proces dekompozicije enitet (relacij)



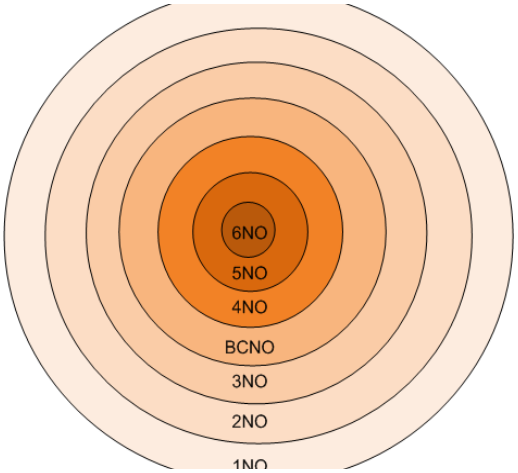
→ Normalne oblike:
  - o pravila o grupiranju atributov v entitete ob upoštevanju logičnih odvisnosti (funkcionalnega, večpomenskega, projekcijsko združitvenega in ključnega tipa).

→ Mejniki:
  - o 1972, 1974, 1977, 1979

→ Avtorji:
  - o E.F. Codd, R. Boyce, R. Fagin

→ Cilj:
  - o odprava nepravilnosti pri vnosu, brisanju in popravljanju podatkov

3.2.1. FORMALNA DEFINICIJA NORMALNIH OBLIK



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.

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).

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).

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).

Četrta normalna oblika

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.

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 NORMALNIH OBLIK

1NO

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.

2NO

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.

3NO

Iz obstoječe entitete prenesemo v novo entiteto tiste atribute, ki so odvisni od neključnega atributa.

4NO

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.

5NO

Relacija je v peti normalni obliki, če smo v relacijo prenesli večkratno pojavnost iste relacije.

→ **1NO**
  - Problem: obstoj ponavljajoče skupine
  - Aktivnost: izločitev PS (dodaj primarni ključ nenormalizirane entitete)

→ **2NO**
  - Problem: obstoj DFO
  - Cilj opazovanja: sestavljen ključ (ugotovimo ali je DFO, če ni, gremo naprej)

→ **3NO**
  - Problem: obstoj TO (tranzitne odvisnosti)
  - Cilj opazovanja: neključni atributi
  - Aktivnost: izločitev TO

3.3. PRIMER: PREDMETNIK

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 1NO

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

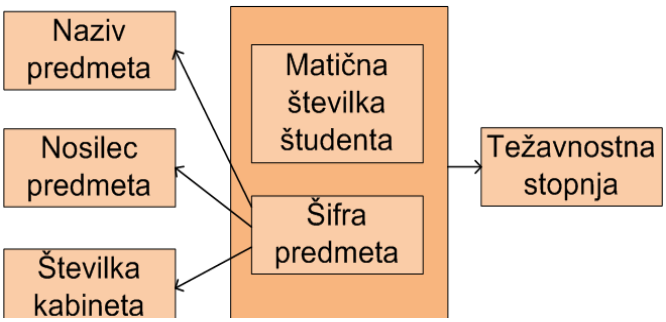
→ Komentar:
  - Entiteta ŠTUDENT je v 3 NO.
  - Entiteta ŠTUDENT_PREDMETNIK ima sestavljen ključ.

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

→ Vzrok za nepravilnosti:

- neključni atributi so odvisni le od dela primarnega ključa.

→ Diagram funkcionalnih odvisnosti:



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 2NO

→ Komentar:

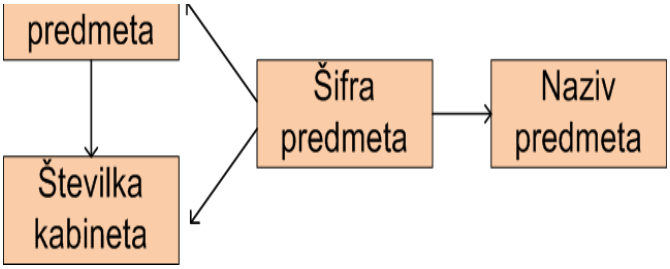
- Entita VPIS je v 3 NO.

→ Nepravilnosti:

- vnos - nov predavatelj,
- brisanje - sprememba v predmetniku,

- popravljanje - selitev nosilca predmeta.

→ Diagram funkcionalnih odvisnosti:



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

4. LOGIČNO OBLIKOVANJE

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.

Relacijski podatkovni model

+	-

Preprostost	Programski produkti so počasni
Stopnja podatkovne neodvisnosti je večja	Mrežni in drevesni podatkovni model bolj razširjena (do nedavnega)
Uporabnik neodvisen od pozanavnja fizične strukture	
Dostop do podatkov je neomejen glede na št. različnih pristopov	
Enostaven dostop do podatkov	
Vgrajena večnivojska podatkovna integriteta	
Podatki so logično in fizično neodvisni od lokacije	

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.

Ozadje

→ 60. 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.

→ 70. 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.

→ 80. 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 strolpci in vrsticami (n-terice).

→ Relacija (matematični pojem – A1,A2, A3, A4...)
→ Tabela (fizični pojem)

Atribut – predstavlja ime stolpca relacije

Domena je množica dopustnih vrednosti (zaloga vrednosti) za en ali več atributov.

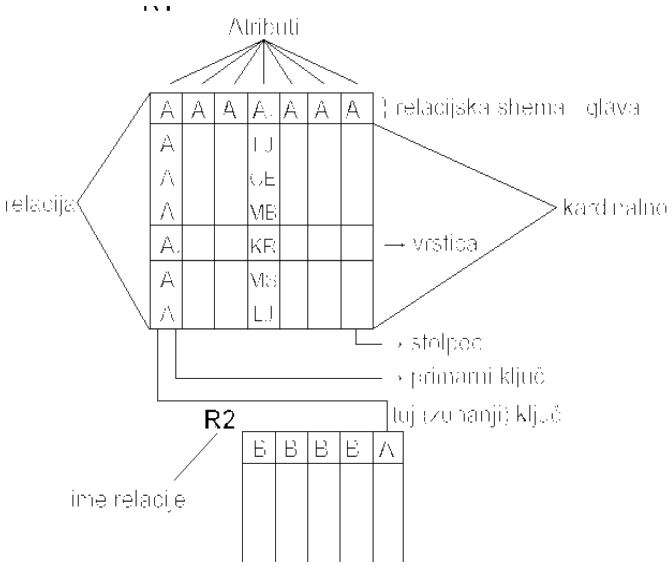
N-terica (tuple) je vrstica relacije, ki predstavlja posamezen zapis oz. podaja podatke za posamezen objekt ZOS. Relacijska shema, predstavljena kot kartezini produkt (D1xD2xD3xD4).

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.



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.

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 x D2 x D3 x ...x Dn = {(d1, d2, d3, ..., Dn) | d1 D1, d2 D2, ..., dn Dn}

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

4.2.2. LASTNOSTI REACIJ

Predstavitev relacijsko-podatkovnega modela in lastnosti relacij

1. Imena relacij so unikatna znotraj PB.
2. Imena atributov so unikatna, znotraj ene relacije.
3. Vrstni red atributov ni pomemben.
4. Vrstni red vrstic ni pomemben.
5. Vrstice se morajo med sebj razlikovati.
6. Ponovljenih zapisov v relaciji ni.

7. Vrednost vsakega atributa mora imeti isto domeno.
8. Vrednosti atributpv so enostavne.

UTEMELJITEV: Izhajajo iz teorije o relacijah/množicah. Množica ne pozna ponavljajočih se elementov, vrstni red ni pomemben/ne obstaja.

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.

→ **Entitetna celovitost**

V relaciji primarni ključ ne more imeti vrednosti null.

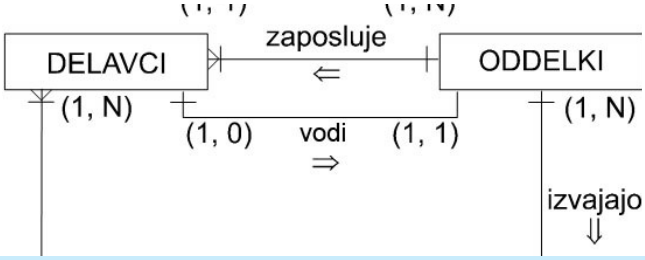
→ **Referenčna celovitost**

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 (E□R)

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

E (A1, A2, ..., An) □ R (A1, A2, ..., An)

2.,3. in 4. korak se ukvarjajo s kardinalnostjo.

2. KORAK (Re $_{1:1}$ □ R)

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.

$$E1\ (E11, E12, \ldots, E1n) \rightarrow S\ (S1, S2, \ldots, Sn)$$
$$E2\ (E21, E22, \ldots, E2m) \rightarrow T\ (T1, T2, \ldots, Tm)$$

$$Re\ (A1, A2, \ldots, An) \rightarrow$$
$$S\ (S1, S2, \ldots, Sn, T1, A1, A2, \ldots, An\ )$$
ali
$$T\ (T1, T2, \ldots, Tm, S1, A1, A2, \ldots, An\ )$$

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.

$$E1\ (E11, E12, \ldots, E1j) \rightarrow S\ (S1, S2, \ldots, Sj)$$
$$En\ (En1, En2, \ldots, Enn) \rightarrow T\ (T1, T2, \ldots, Tn)$$

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

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.

$$En\ (En1, En2, \ldots, Enm) \rightarrow S\ (S1, S2, \ldots, Sm)$$
$$Em\ (Em1, Em2, \ldots, Emk) \rightarrow T\ (T1, T2, \ldots, Tk)$$
$$Re\ (A1\ \ A2 \quad \ldots \quad An) \rightarrow R\ (S1\ \ T1\ \ A1\ \ A2 \quad \ldots \quad An\ )$$

Primer

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:
  - o nadzor dostopa do podatkovne baze,
  - o integriteta podatkovne baze,
  - o vzpostavitev nadzornega sistema,
  - o uporabniško dostopen opis podatkov,
- → pregled podatkov.

5.1. FUNKCIJE SUPB

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

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).

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.

- → Podpora transakcijam
  - o SUPB mora zagotoviti mehanizem za zagotavljanje beleženja posameznih transakcij.
- → Soglasen nadzor
  - o SUPB mora zagotoviti popravljanje podatkovne baze.

- → Sistem ponovne vzpostavitve
  - o SUPB mora zagotoviti ponovno vzpostavitev podatkovne baze po poškodbi.
- → Sistem avtorizacije
  - o SUPB mora zagotoviti sistem, ki samo avtoriziranim uporabnikom omogoča dostop do podatkovne baze.
- → Podpora za komuniciranje
  - o SUPB mora biti pripravljen na integracijo s komunikacijsko programsko opremo.
- → Podpora podatkovni neodvisnosti
  - o SUPB mora zagotavljati podporo neodvisnosti programov od aktualne strukture podatkovne baze.
- → Podporni servisi
  - o SUPB zagotavljajo množico podpornih servisov.

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 - Vsak SUPB mora izpolniti pravili 0 in 12, da ga lahko imenujemo relacijski SUPB,
- → strukturna pravila - Osnova struktura relacijskega podatkovnega modela je relacija,
- → 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,
- → pravila za delo (ravnanje) s podatki - Idealen SUPB bi naj podpiral 18 različnih kategorij za ravnanje s podatki,
- → pravila podatkovne neodvisnosti - Za predstavitev neodvisnosti podatkov od aplikacij, ki uporabljajo te podatke, so definirana tri pravila.

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.

→ **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).

→ **Omejitev operacij**

- Relacijski model ima končno množico operacij.

→ **Rekurzivno povpraševanje**

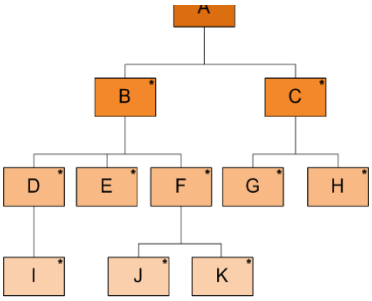
- Predstavitev rekurzivnega povpraševanja je zahtevno. Rešitev omogoča širitev visokonivojskih programskih jezikov z SQL.

→ **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.



V drevesnem podatkovnem modelu obstajajo povezave starši-otroci (1:M):

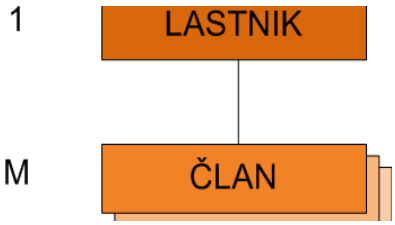
- starši imajo lahko več otrok,
- vsak otrok ima le en starše,
- starše najvišjega nivoja imenujemo koren,
- otrok brez otrok je list.

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).



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. RELACIJSKA ALGEBRA

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

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

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

6.1. UNIJA R ∪ S

1. Predpogoj

Kompatibilost unije relcij $r(A_1, A_2, \ldots, A_n)$ in $s(B_1, B_2, \ldots, B_n)$ izpolnjujeta pogoj kompatibilnosti unije, če sta stopnje n in $dom(A_i) = Dom(B_i)$ za $1 \leq i \leq n$.

2. Definicija:

Rezultat operacije $r \cup s$ je relacija, ki vsebuje vse n-terice, ki so v relaciji r ali v relaciji s ali v obeh. Ponavljajoče se vrstice so izločene.

Opomba: Operacija je komutativna.

- $r \cup s = s \cup r$

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

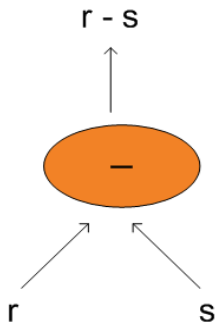
ODDELEK ∪ ODDELEK1		
ŠtOdd	NazivOdd	Kraj
o_1	n_1	k_1
o_2	n_2	k_1
o_3	n_3	k_2
o_4	n_4	k_2
o_5	n_5	k_3

6.2. RAZLIKA R - S

1. Predpogoj - kompatibilnost unije

2. Definicija: Rezultat operaacije r-s je relacija, ki vsebuje vse n-terice, ki so v relaciji e in niso v relaciji s.

Opomba: Operacija ni komutativna. $r - s \neq s - r$



Primer: Oddelek - Oddelek1

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

Primer: Oddelek1 - Oddelek

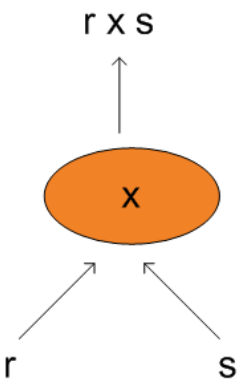
ODDELEK1 - ODDELEK		
ŠtOdd	NazivOdd	Kraj
o_4	n_4	k_2
o_5	n_5	k_3

6.3. KARTEZIČNI PRODUKT R X S

1. Predpogoj: Kompatibilnost unije **ni** zahtevana.

2. Definicija: Rezultat operacije r × s je relacija Q z n+m atributi:
$Q(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m)$.

Opomba: Kartezični produkt se redko uporablja kot samostojna operacija.



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

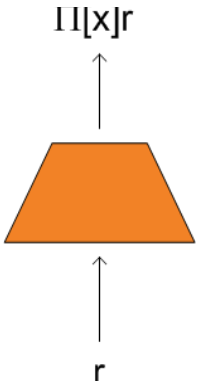
6.4. PROJEKCIJA Π[X] R

1. Predpogoj: Operacija je smiselna, če je X r.

2. Definicija: Operacija projekcije izvede selekcijo množice X (X je množica izbranih atributov) iz relacije r.

Opomba: Izločitev ponavljajočih se vrstic.

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

1. 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 (=, <, ≤, >, ≥, ≠).

Opomba: Opomba: Vrstni red upoštevanja pogojev ni pomemben. Operacija je komutativna.

σ[F1] (σ[F2]r) = σ[F2] (σ[F1]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

1. Predpogoj: Kompatibilnost unije.

2. 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: $r \cap s \equiv r - (r - s)$

Opomba: Operacije je komutativna. $r \cap s = s \cap r$

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 $|X|_{iΘj}$ S

1. Definicija: Rezultat operacije r (A1, A2, …, An) $|x|_{iΘj}$ s(B1, B2, …, Bn) 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.

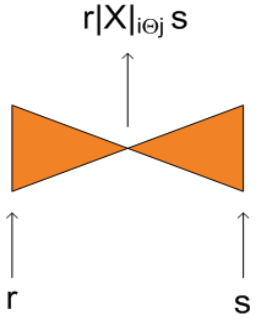
Definicija z osnovnimi operacijami: $r |x|_{i\,\Theta\,j}\, s \equiv \sigma[F]\,(r \times s)$

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

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 $\|x\|_{Kraj\Theta k2}$ 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

1. 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.

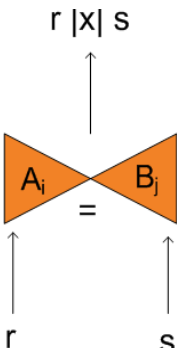
Definicija z osnovnimi operacijami: $r \,|x|\, s \equiv \Pi\,[x](\sigma[F]\,(r \times s))$

Opomba: Operacija je asociativna.

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

1. Predpogoj: Operacija je smiselna, če S ⊆ R.

2. 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: tr[s] = ts[s] in tr[r - s] = t[r - s].

Definicija z osnovnimi operacijami: $r / s \equiv \Pi [x] r - \Pi [x] (((\Pi [x] r) \times s) - r); x=R-S$

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

Omogočajo primerljivost 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.

→ 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, upravljanja transakcij, uporabniških funkcij in združitvenih operacij v okviru SQL.

→ 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.

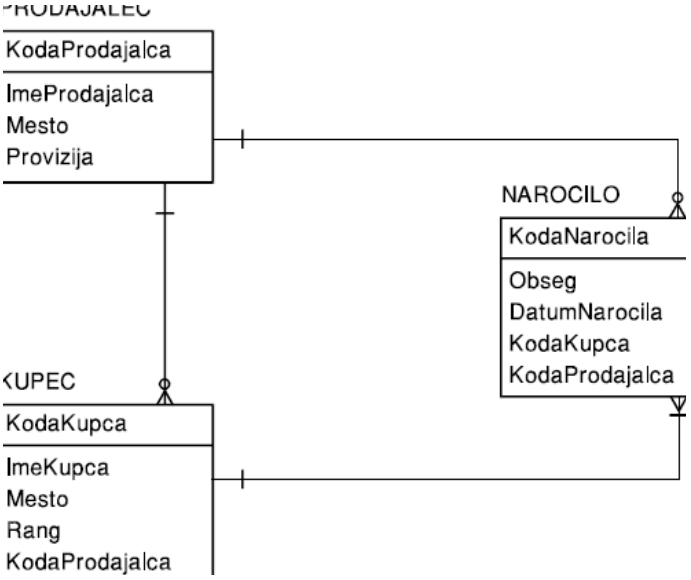
→ Osnovna struktura vprašanja:

- SELECT A_1, A_2, …, A_n
  FROM r_1, r_2, …, 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:

- Π [A_1, A_2, …, A_n](σ[P] ($r_1 \times r_2 \times ... \times r_m$)))

E-R diagram



Primer: slide-i.

- ☐ 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.

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.

- ☐ 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.

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

Kaj zmore:

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

→ 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 | ∃t1 ∈ r1, t2 ∈ r2, …, tm ∈ rm (
  t[ri1.Aj1] = ti1[Aj1] ∧
  t[ri2.Aj2] = ti2[Aj2] ∧ … ∧
  t[rin.Ajn] = tin[Ajn] ∧
  P(t1, t2, …, 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.

→ 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).

Primer – slide-i

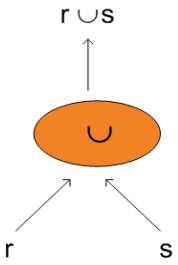
7.4. PRIMERJAVA POVPRAŠEVALNIH JEZIKOV

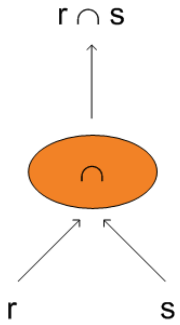
☐ **SQL :QUEL**

- 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.

☐ **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 $\forall$).


r ∪ s
∪
r
s


r ∩ s
∩
r
s