# Projektovanje baza podataka

## Relacioni model podataka

22. april 2026.

# Sadržaj

# Napomena

Slajdovi su nastali obradom slajdova prof. dr Saše Malkova za predmet Projektovanje baza podataka

# Sadržaj

# Značaj relacionog modela podataka

- Relacioni model podataka je iz više razloga najznačajniji model podataka koji je do sada razvijen:
  - Potpuna formalizacija
    - osnov za formalno izučavanje i unapređivanje
    - neprotivrečan
    - nedvosmisleno precizan u primeni
    - pogodan za automatsko obrađivanje i optimizovanje
  - Jedinstvena reprezentacija podataka i metapodataka
    - sve vrste informacija se predstavljaju na isti način – relacijama
    - sve vrste metapodataka se predstavljaju na isti način – relacijama
  - U visokoj meri standardizovan
  - Veoma zastupljen u praksi
    - veliki broj različitih sistema i raznovrsnih primena
    - izuzetno dobro proveren i dokumentovan

# Delovi relacionog modela

- Relacioni model čine:
  - Strukturni deo relacionog modela
    - način modeliranja podataka
  - Manipulativni deo relacionog modela
    - način rukovanja modeliranim podacima
  - Integritetni deo relacionog modela
    - način obezbeđivanja *valjanosti* podataka

# Strukturni deo relacionog modela

- Osnovna ideja je da se **sve** modelira relacijama
  - *sve* znači i *entiteti* i *odnosi*
    - pojmovi entiteta i odnosa ne mogu do kraja da se razdvoje, zato što neke stvari iz jednog ugla mogu da izgledaju kao odnosi a iz drugog kao entiteti, ali je uobičajeno da se posmatraju odvojeno

# Matematičke osnove, model relacije

- Relacioni model je u *potpunosti* formalno matematički zasnovan
- Svakoj matematičkoj relaciji odgovara tačno jedan skup objekata koji zadovoljavaju relaciju
- Taj skup se često upotrebljava kao matematički model relacije
- Ako je $\rho$ n-arna relacija i njen domen označen sa Dom($\rho$)
- Onda je jedan model relacije $\rho$ skup:

$$model(\rho) = \{(a_1, a_2, \ldots, a_n) \mid (a_1, a_2, \ldots, a_n) \in Dom(\rho) \wedge \rho(a_1, a_2, \ldots, a_n)\}$$

- Primetimo da važi:

$$model(\rho) \subseteq Dom(\rho)$$

# Predstavljanje relacija skupovima

- Za sve n-torke iz Dom($\rho$) važi:
  $\rho(a_1, a_2, \ldots, a_n) \Leftrightarrow (a_1, a_2, \ldots, a_n) \in \mathit{model}(\rho)$
- Zbog navedene ekvivalencije često se umesto oznake model($\rho$) upotrebljava i samo oznaka relacije $\rho$ u istom kontekstu:

$$\rho = \{(a_1, a_2, \ldots, a_n) \mid (a_1, a_2, \ldots, a_n) \in \mathit{Dom}(\rho) \wedge \rho(a_1, a_2, \ldots, a_n)\}$$

- T.j. relacije često predstavljamo upravo pomoću njihovog skupovnog modela, bez mogućnosti da to izazove nesporazume

# Primer

- Neka imamo
  - crvenu, plavu, belu i zelenu kutiju
  - i u njima lopte, kocke i pločice
- Binarna relacija *kutijaSadrži(K,P)* je zadovoljena ako kutija $K$ sadrži $P$
- Njen domen je:
  *Dom(kutijaSadrži)* = {crvena, plava, bela, zelena} $\times$ {lopte, kocke, pločice}
- Neka naredni model relacije opisuje šta se nalazi u kojoj kutiji:
  *kutijaSadrži* = {(plava,lopte), (plava,kocke), (zelena,pločice),(crvena,kocke)}
- Iskaz *kutijaSadrži*(plava,kocke) je tačan
- Iskaz *kutijaSadrži*(zelena,lopte) nije tačan
- Iskazi *kutijaSadrži*(žuta,kocke) i *kutijaSadrži*(zelena,knjige) nisu definisani zato što argumenti nisu u domenu relacije

# Primer

Relacije (odnosno njihovi modeli) sa konačnim brojem elemenata mogu da se predstave i pomoću tabela (tablica):

| *kutijaSadrži* | |
|---|---|
| **BOJA** | **SADRŽAJ** |
| plava | lopte |
| plava | kocke |
| zelena | pločice |
| crvena | kocke |

# Entiteti i atributi

- Neka je E neki skup *entiteta*
  - *Entitetima* nazivamo sve različite postojeće elemente ("objekte") sistema koji posmatramo, koje modeliramo bazom podataka
- Kažemo da se skup entiteta karakteriše konačnim skupom atributa $A(E) = \{A1, \ldots, An\}$, u oznaci $E(A1, \ldots, An)$, akko:
  - Svaki atribut Ai predstavlja funkciju koja slika entitete u odgovarajući domen atributa Di:
    $Ai : E \rightarrow Di$
  - Svaki atribut Ai ima jedinstven naziv ti
- Za svaki entitet $e \in E$, vrednost funkcije $Ai(e) \in Di$ predstavlja *vrednost atributa* Ai

# Entiteti i atributi – primer

Neka je $E$ skup radnika koji se karakteriše atributima:

- ime (niska od 20 znakova)
- prezime (niska od 25 znakova)
- osnova_plate (broj oblika 7.2)

Skup svih atributa određuje funkciju:

$$\alpha(x) = (ime(x), prezime(x), osnova\_plate(x))$$

| IME | PREZIME | OSNOVA_PLATE |
|---|---|---|
| Dragana | Pantić | 45000 |
| Marko | Marković | 40000 |
| Dragana | Pantić | 45000 |
| Jelena | Popović | 43000 |
| Dragiša | Đukić | 47000 |

# Entiteti i atributi

- Skup svih atributa određuje funkciju:

$$\alpha(e) = (A_1(e), \ldots, A_n(e))$$

- Slika skupa entiteta funkcijom $\alpha$ je skup:

$$R = \alpha(E)$$

- Podsetimo se, funkcija $\alpha$ je injektivna akko za sve različite originale daje različite slike, tj. akko važi:

$$\alpha(e) = \alpha(u) \text{ akko } e = u$$

- Ako je opisana funkcija $\alpha$ injektivna, onda kažemo da skup atributa (i odgovarajuća funkcija $\alpha$) *dobro karakterišu* skup entiteta $E$

# Predstavljanje entiteta relacijama

Ako je $\alpha$ injektivna funkcija i svaki od atributa je atomičnog tipa, onda kažemo da je slika $R = \alpha(E)$

$$\begin{gathered}\text{relacija } R \text{ sa atributima } A(R) = \{A_1, \ldots, A_n\},\\ \text{domenom relacije } Dom(R) = D_1 \times \cdots \times D_n\\ \text{i nazivima atributa } Kol(R) = (t_1, \ldots, t_n)\end{gathered}$$

i skup entiteta $E$ sa atributima $A(E) = \{A_1, \ldots, A_n\}$ modeliramo relacijom $R$.

- funkcija $\alpha(E)$ je *dobra karakterizacija* skupa entiteta $E$
- formalno, definišemo $R$ tako da je $model(R) = \alpha(E)$
- atribut je *atomičan* (tj. atomičnog tipa) ako njegov domen ne predstavlja proizvod drugih domena, tj. ako je skalarnog tipa

# Zapisivanje

- Ako je $\alpha(e) = (a_1, \ldots, a_n)$, onda $n$-torku $(a_1, \ldots, a_n)$, koja formalno predstavlja „model entiteta $e$“, često nazivamo i „entitetom $e$“
- $A_i(e)$ zapisujemo i kao $e.t_i$
- Najčešće ne pravimo razliku između relacije $R$ i odgovarajućih relacija dobijenih permutovanjem atributa
- t.j. atribute referišemo koristeći imena a ne redne brojeve
- zbog toga imena atributa u relaciji moraju da budu jedinstvena

# Predstavljanje relacije tabelom

- Relaciju $R = \alpha(E)$ često predstavljamo, pa i nazivamo, *tabelom*:
- *Kolone* tabele odgovaraju atributima $A_1, \ldots, A_n$
- *Nazivi kolona* odgovaraju nazivima atributa $t_1, \ldots, t_n$
- *Vrste* tabele odgovaraju *torkama* relacije, tj. entitetima

# „Relacija“ ili „tabela“

- U kontekstu teorijskog razmatranja, kao i konceptualnog i logičkog modeliranja, uobičajeno je da se koristi termin „*relacija*”
- U kontekstu fizičkog modeliranja obično se koristi termin „*tabela*”

# Predstavljanje entiteta - primer

- Relacija ne sme da ima jednake elemente

| RADNIK | | |
|---|---|---|
| **IME** | **PREZIME** | **OSNOVA_PLATE** |
| Dragana | Pantić | 42000 |
| Marko | Marković | 40000 |
| Dragana | Pantić | 45000 |
| Jelena | Popović | 43000 |
| Dragiša | Đukić | 47000 |

- Ako je $\alpha(x) = (ime(x), prezime(x), osnova\_plate(x))$, onda ćemo $n$-torku $(ime(x), prezime(x), osnova\_plate(x))$ u zapisima obično da izjednačavamo sa $x$
- $ime(x)$ ćemo da zapisujemo kao $x.ime$ (i slično za ostale atribute)
- Nećemo razlikovati relaciju RADNIK od relacija dobijenih permutovanjem atributa
  - kao ni tabele koje se razlikuju samo po redosledu redova

# Relaciona baza podataka

- **Relaciona baza podataka je skup relacija**
- *Opis relacije* čine domen relacije i nazivi atributa
- *Relaciona schema* je skup opisa relacija koje čine bazu podataka

# Primer baze podataka – tipovi

- Uvedimo opisno sledeće skupove:
- $N(a)$ = „skup celih brojeva zapisanih sa najviše $a$ dekadnih cifara"
- $N(a, b)$ = „skup brojeva koji u dekadnom zapisu sa leve strane decimalne zapete imaju najviše $a$, a sa desne strane najviše $b$ cifara"
- $C(a)$ = „skup niski dužine do $a$ znakova"
- $D$ = „skup datuma (u nekom podrazumevanom opsegu)"
- Navedeni skupovi se najčešće upotrebljavaju kao domeni (tj. tipovi) atributa relacija
- Svi atributi u narednom primeru relacione baze podataka imaju za domen neki od navedenih skupova

# Primer baze podataka – schema

- $Kol(ORG\_JEDINICA) = (\text{'org\_jed\_id'},\text{' nad\_org\_jed\_id'},\text{' naziv'})$
- $Dom(ORG\_JEDINICA) = N(3) \times N(3) \times C(60)$
- $Kol(RADNIK) =$
  $(\text{'radnik\_id'},\text{' org\_jed\_id'},\text{' ime'},\text{' prezime'},\text{' pol'},\text{' dat\_zaposlenja'},\text{' osnova\_plate'})$
- $Dom(RADNIK) = N(4) \times N(3) \times C(30) \times C(50) \times C(1) \times D \times N(10,2)$
- $Kol(KAT\_STAZA) = (\text{'od\_godine'},\text{' do\_godine'},\text{' staz\_bonus'},\text{' naziv'})$
- $Dom(KAT\_STAZA) = N(2) \times N(2) \times N(10,2) \times C(60)$
- $Kol(RESENJE) = (\text{'rad\_id'},\text{' proj\_id'},\text{' dat\_pocetka'},\text{' dat\_prestanka'},\text{' opis'})$
- $Dom(RESENJE) = N(4) \times N(3) \times D \times D \times C(60)$
- $Kol(PROJEKAT) = (\text{'projekat\_id'},\text{' proj\_bonus'},\text{' naziv'})$
- $Dom(PROJEKAT) = N(3) \times N(10,2) \times C(60)$

# Primer baze podataka – schema

| RADNIK | |
|---|---|
| RADNIK_ID | N(4) |
| ORG_JED_ID | N(3) |
| IME | C(30) |
| PREZIME | C(50) |
| POL | C(1) |
| DAT_ZAPOSLENJA | D |
| OSNOVA_PLATE | N(10,2) |

| ORG_JEDINICA | |
|---|---|
| ORG_JED_ID | N(3) |
| NAD_ORG_JED_ID | N(3) |
| NAZIV | C(60) |

| PROJEKAT | |
|---|---|
| PROJEKAT_ID | N(3) |
| PROJ_BONUS | N(10,2) |
| NAZIV | C(60) |

| REŠENJE | |
|---|---|
| RAD_ID | N(4) |
| PROJ_ID | N(3) |
| DAT_POČETKA | D |
| DAT_PRESTANKA | D |
| OPIS | C(60) |

| KAT_STAŽA | |
|---|---|
| OD_GODINE | N(3) |
| DO_GODINE | N(3) |
| STAŽ_BONUS | N(10,2) |
| NAZIV | C(60) |

# Primer baze podataka – sadržaj

| RADNIK | | | | | | |
|---|---|---|---|---|---|---|
| RADNIK_ID | ORG_JED_ID | IME | PREZIME | POL | DAT_ZAPOSLENJA | OSNOVA_PLATE |
| 100 | 10 | Petar | Perić | M | 26.03.1991. | 42000 |
| 101 | 11 | Marko | Marković | M | 30.03.1990. | 43000 |
| 102 | - | Lazar | Lazić | M | 21.04.1981 | 28000 |
| 103 | 11 | Milica | Milić | Ž | 30.06.1979. | 45000 |
| 104 | 30 | Dragica | Dragić | Ž | 10.07.1988. | 44000 |
| 105 | 30 | Gorica | Bojić | Ž | 20.07.2003. | 39000 |
| 106 | 30 | Vladana | Mladić | Ž | 05.01.2007. | 29000 |
| 107 | 40 | Đurđa | Perić | Ž | 29.03.1978. | 35000 |
| 108 | 40 | Đura | Lazić | M | 18.03.1983. | 42000 |

# Primer baze podataka – sadržaj

| ORG_JEDINICA | | |
|---|---|---|
| ORG_ JED_ID | NAD_ORG _JED_ID | NAZIV |
| 40 | - | Softver |
| 30 | 40 | Projektovanje |
| 11 | 10 | Dokumentacija |
| 10 | 40 | Planiranje |
| 50 | 40 | Analiza |
| 60 | 40 | Dizajn i modeliranje |
| 70 | 40 | Kodiranje |

# Primer baze podataka – sadržaj

| KAT_STAZA | | | |
|---|---|---|---|
| OD_ GODINE | DO_ GODINE | STAŽ_ BONUS | NAZIV |
| 1 | 5 | 2000 | Početna |
| 6 | 20 | 5000 | Srednja |
| 21 | 50 | 8000 | Završna |

| PROJEKAT | | |
|---|---|---|
| PROJEKAT_ID | PROJ_ BONUS | NAZIV |
| 100 | - | Crveni |
| 150 | - | LJubičasti |
| 200 | - | Plavi |
| 250 | 1500 | Zeleni |
| 300 | 2000 | Žuti |
| 250 | 1500 | Narandžasti |
| 900 | - | Crni |

# Primer baze podataka – sadržaj

| REŠENJE | | | | |
|---|---|---|---|---|
| RAD_ID | PROJ_ID | DAT_POČETKA | DAT_PRESTANKA | OPIS |
| 100 | 200 | 01.01.1991. | - | privremeno učešće |
| 101 | 200 | 01.01.1991. | - | - |
| 108 | 200 | 01.01.1984. | - | - |
| 104 | 100 | 01.01.1989. | 31.12.1998. | - |
| 105 | 100 | 01.01.2004. | - | - |
| 106 | 100 | 01.01.2008. | - | - |
| 107 | 100 | 01.01.1981. | - | - |
| 100 | 150 | 01.01.1999. | 31.12.1998. | Koordinator |
| 101 | 150 | 01.01.1972. | 31.12.1998. | - |
| 102 | 150 | 01.01.1972. | 31.12.1998. | - |
| 108 | 150 | 01.01.1972. | 31.12.1998. | - |
| 104 | 150 | 01.01.1986. | 31.12.1998. | - |
| 105 | 150 | 01.01.1998. | 31.12.1998. | - |
| 107 | 150 | 01.01.1999. | - | Projektant |
| 108 | 250 | 01.01.1999. | - | - |
| 108 | 300 | 01.01.1999. | - | - |
| 108 | 350 | 01.01.1999. | - | - |
| 105 | 250 | 01.01.1998. | - | - |
| 107 | 350 | 01.01.1998. | - | - |

# Ključ i natključ

- Pojam *ključa* ima veoma važnu ulogu u relacionom modelu
- **Natključ** $K$ relacije $R$ je podskup njenih atributa $K \subseteq A(R)$ za koji važi:
  - skup atributa $K$ jednoznačno određuje torke relacije, t.j. ako znamo vrednosti tih atributa onda je jednoznačno određena torka
- **Ključ** (ili **ključ-kandidat**) relacije $R$ je natključ $K$ za koji važi:
  - ne postoji pravi podskup skupa $K$ koji je natključ relacije $R$
- Primetimo:
  - Svaka relacija ima najmanje jedan ključ, a može da ih ima više
  - Svaki natključ sadrži bar jedan ključ

# Modeliranje odnosa

- Odnosi se modeliraju na isti način kao i entiteti – relacijama
  - štaviše, odnos često može da se posmatra kao entitet, pa je to sasvim prirodno
- Opisaćemo formalno predstavljanje binarnih odnosa, a za složenije važe analogni principi
- Neka su entiteti $e \in E$ i $f \in F$ modelirani kao

$$e = (x_1, \ldots, x_n), f = (y_1, \ldots, y_m)$$

, i neka mogu da budu u odnosu $w$, koji nema dodatne atribute (tj. samo znamo da li su dva entiteta u odnosu ili nisu)

# Modeliranje odnosa

- Ako su dva entiteta $e \in E$ i $f \in F$ u nekom odnosu $w$, onda to opisujemo novom relacijom $\rho(e, f)$, čiji je domen $Dom(\rho) = E \times F$
- Ako su entiteti $e \in E$ i $f \in F$ modelirani kao

$$e = (x_1, \ldots, x_n), f = (y_1, \ldots, y_m)$$

, onda se njihov odnos modelira kao

$$\rho(e, f) = (x_1, \ldots, x_n, y_1, \ldots, y_m)$$

# Modeliranje odnosa

- Primetimo, ako su $K_E$ i $K_F$ neki ključevi relacija $E$ i $F$, onda skup atributa $K_R = K_E \cup K_F$ predstavlja ključ relacije $\rho$
  - atributi iz $K_R$ jednoznačno određuju sve atribute iz $E$ i $F$, pa time i iz $\rho$
- Posebno, onda se odnos može opisati „suženom" relacijom $\rho_1$ koja ne sadrži sve atribute iz $E$ i $F$ već samo atribute skupa $K_R$

# Modeliranje odnosa sa atributima

- U opštem slučaju odnosi mogu da imaju dodatne atribute
- Na primer, u jednoj biblioteci odnos *pozajmica* između *knjige* i *člana biblioteke* osim toga što ga čine jedna knjiga i jedan član, može da ima i dodatne atribute, kao na primer:
  - datum pozajmice
  - datum vraćanja
- Tada kažemo da je odnos karakterisan povezanim entitetima i dodatnim atributima, koje nazivamo *opisnim atributima odnosa*

# Integrisani odnosi

- U nekim slučajevima odnos između dva entiteta (ili odnosa) može da bude „integrisan" u okviru relacije koja modelira jedan od tih entiteta (ili odnosa)
- Na primer, relacija *Student* može da sadrži atribute relacije *StudijskiProgram* (sve ili samo ključ), čime se opisuje koji studijski program je student upisao
- Ali to je istovremeno i odnos između studenta i studijskog programa
- Odnos između relacija $E$ i $F$ može da bude *integrisan* u okviru relacije $E$ samo ako svakom entitetu iz te relacije odgovara tačno po jedan entitet relacije $F$
  - (ili najviše jedan, ako su dopuštene nedefinisane vrednosti)

# Primer integrisanog odnosa

- U prethodnom primeru baze podataka već imamo i neke integrisane odnose:
  - Svakom radniku iz RADNIK odgovara tačno jedan entitet iz ORG_JEDINICA
    - radnik radi u tačno jednoj organizacionoj jedinici
  - Zato relacija RADNIK sadrži atribut ključa relacije ORG_JEDINICA

# Sadržaj

# Manipulativni deo modela

- Manipulativni deo relacionog modela definiše formalne načine rukovanja modeliranim podacima
- Ključno mesto u manipulativnom delu modela ima pojam upita
- *Upit* je definicija (ili izračunavanje) nove relacije na osnovu već poznatih relacija baze podataka

# Formalni upitni jezici

- Formalni upitni jezici u relacionom modelu su:
  - Relaciona algebra i
  - Relacioni račun

# Relaciona algebra

- Relaciona algebra je proširenje skupovne algebre
  - Zadržavamo osnovne skupovne operacije
    - Unija, Presek, Razlika, Proizvod
- Osnovne dodatne operacije su:
  - Projekcija
  - Restrikcija
  - Prirodno spajanje
- Rezultati svih ovih operacija su nove relacije

# Imena relacija i atributa

- U slučaju proizvoda i nekih drugih binarnih operacija može da se dogodi da rezultat ima dva atributa istog imena
  - u tom slučaju se naziv atributa dopunjava s leva nazivom relacije iz koje potiče

$$R(A, B) \times Q(B, C) = W(A, R\_B, Q\_B, C)$$

  - Ako je potrebno da se u operaciji upotrebi više puta ista relacija, da ne bi dolazilo do nesporazuma uvode se alijasi (nadimci) relacija
    DEFINE ALIAS R1 FOR R
    R ... R1

# Projekcija

- **Projekcija** se definiše kao izbor kolona $X$ jedne relacije:

$$R[X] = \{x \mid x \in Dom(X) \wedge (\exists y \in Dom(Y))(x, y) \in R\}$$

- gde su $X$ i $Y$ komplementarni skupovi atributa relacije $R$: $Y = A(R) \setminus X$
- Rezultat projekcije može da ima manje elemenata nego polazna relacija
  - Zbog toga što se radi o skupovima $n$-torki, projekcija ne može da sadrži ponovljene elemente
  - Rezultat projekcije sigurno ima isti broj elemenata kao relacija $R$ akko je $X$ natključ relacije $R$
- U terminima tabela, projekcija je vertikalno „sečenje" tabele
- Primer:
  - $RADNIK[ime, prezime] = \{(x.ime, x.prezime) \mid x \in RADNIK\}$

# Restrikcija

- **Restrikcija** se definiše kao izbor torki relacije $R$ u kojima dati atribut $X \in A(R)$ ima datu vrednost $x \in Dom(X)$:

$$R[X = x] = \{e \mid e \in R \wedge X(e) = x\}$$

- Slobodnije posmatrano, $X$ može da bude proizvoljan podskup atributa
- Opštije, restrikcija po proizvoljnom predikatu $p : R \to \{T, \bot\}$ je:

$$R[p] = \{e \mid e \in R \wedge p(e)\}$$

- U terminima tabela, restrikcija je horizontalno „sečenje" tabele
- Primer:
  - $RADNIK[org\_jed\_id = 40] = \{x \mid x \in RADNIK \wedge x.org\_jed\_id = 40\}$

# Prirodno spajanje

- **Prirodno spajanje** relacija $R$ i $Q$ sa zajedničkim skupom atributa $X$ definiše se kao:

$$R * Q = \{(e, f) \mid e \in R \wedge f \in Q \wedge X(e) = X(f)\}$$

$$[X \cup Y \cup Z] = \{(x, y, z) \mid (x, y) \in R \wedge (x, z) \in Q\}$$

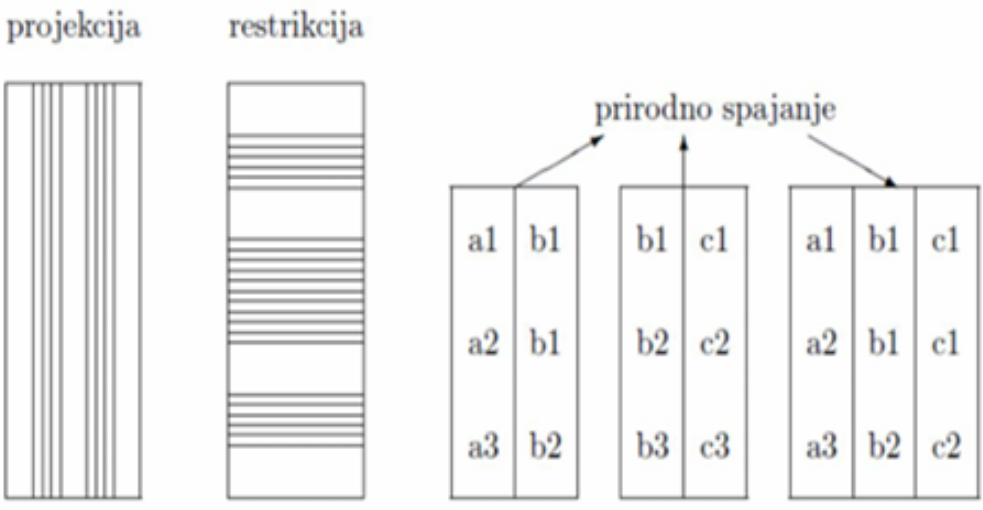
- gde su:
  - $Y = A(R) \setminus X$
  - $Z = A(Q) \setminus X$
- Spajaju se torke jedne relacije sa torkama druge relacije koje imaju jednake vrednosti zajedničkih atributa
- Rezultat sadrži samo jednu kopiju zajedničkih atributa
- Rezultat sadrži najmanje nula torki, ako nema torki sa istim vrednostima zajedničkih atributa
- Rezultat sadrži najviše $M \times N$ torki, gde su $M$ i $N$ brojevi torki relacija $R$ i $Q$, ako sve torke obe relacije imaju iste vrednosti zajedničkih atributa

# Prirodno spajanje

Primer:

- $RADNIK * ORG\_JEDINICA = \{(org\_jed\_id, x, y) \mid (org\_jed\_id, x) \in RADNIK \wedge (org\_jed\_id, y) \in ORG\_JEDINICA\}$

# Ilustracija operacija



# Dodatne operacije

- Definišu se i druge operacije, kao:
  - slobodno spajanje
  - deljenje

# Slobodno spajanje

- **Slobodno spajanje** relacija $R$ i $Q$ po datom binarnom predikatu $p : R \times Q \to \{T, \bot\}$ se definiše kao:

$$R[p]Q = \{(e, f) \mid e \in R \wedge f \in Q \wedge p(e, f)\}$$

- Spajaju se torke jedne relacije sa torkama druge relacije sa kojima ispunjavaju dati uslov
- Slobodno spajanje je opštije od prirodnog

# Deljenje

- **Deljenje** relacije $R$ relacijom $Q$ po podskupu atributa $Y \subseteq A(R)$, gde je $Z = A(Q)$ skup atributa koji po broju i domenu odgovara skupu $Y$, definiše se kao:

$$R[Y\%Z]Q = \{x \mid x \in R[X] \wedge \{x\} \times Q[Z] \subseteq R\}$$

- gde je $X = A(R) \setminus Y$
- gde je $Z = A(Q)$ skup atributa koji po broju i domenu odgovara skupu $Y$
- Rezultat je maksimalni podskup relacije $R[X]$ takav da je njegov proizvod sa $Q$ podskup relacije $R$
  - izdvajaju se one torke atributa $Y$ koje su u $R$ uparene sa svim torkama iz $Q$
- Primer - izdvojiti radnike koji su radili na svim projektima:
  - $RESENJE[RAD\_ID, PROJ\_ID]$
    $[PROJ\_ID\%PROJEKAT\_ID]$
    $PROJEKAT[PROJEKAT\_ID]$

# Proširena relaciona algebra

- **Proširena relaciona algebra** obuhvata rad sa **nedefinisanim vrednostima**
  - Bez obzira na razlog nedefinisane vrednosti, koristi se jedna ista oznaka nedefinisane ili nedostajuće vrednosti
  - Svaki domen se proširuje nedefinisanom vrednošću
- Time se menja i strukturni deo modela, ali su važnije promene u manipulativnom delu
- Skup istinitosnih vrednosti se proširuje *nepoznatom logičkom vrednošću*
  - Sve relacije poređenja neke vrednosti sa nedefinisanim vrednostima imaju za rezultat *nepoznato*
  - Logičke operacije se proširuju da rade sa nepoznatom vrednošću

# Proširene logičke operacije

| $\wedge$ | T | F | $n$ |
|---|---|---|---|
| T | T | F | $n$ |
| F | F | F | F |
| $n$ | $n$ | F | $n$ |

| $\vee$ | T | F | $n$ |
|---|---|---|---|
| T | T | T | T |
| F | T | F | $n$ |
| $n$ | T | $n$ | $n$ |

| $\neg$ | |
|---|---|
| T | F |
| F | T |
| $n$ | $n$ |

# Proširena restrikcija

- Uvode se nove vrste restrikcije
  - *tačna restrikcija*, odgovara originalnoj restrikciji
    - izdvajaju se torke za koje je vrednost uslova ***tačno***
    - označava se navođenjem indeksa uslova „T"

$$R[p_T] = R[p]_T = \{x \mid x \in R \wedge p(x) = T\}$$

  - *možda restrikcija* predstavlja novu operaciju
    - izdvajaju se torke za koje je vrednost uslova *nepoznat*
    - označava se navođenjem indeksa uslova „n" (ili neki drugi znak, zavisno od izvora)

$$R[p_n] = R[p]_n = \{x \mid x \in R \wedge p(x) = n\}$$

  - *ne-lažna restrikcija* predstavlja uniju prethodne dve
    - izdvajaju se torke za koje vrednost uslova *nije lažan*
    - može da se označava kombinacijom prethodna dva „Tn"

$$R[p_{Tn}] = R[p]_{Tn} = \{x \mid x \in R \wedge p(x) \neq \perp\}$$

# Relacioni račun

- Primena predikatskog računa na relacije
- Kod je definisao dve varijante računa:
  - relacioni račun n-torki
  - relacioni račun domena
- Ekvivalentan relacionoj algebri

# Relacioni račun n-torki

- *Izraz* relacionog računa n-torki je:

$$\{(t_1, t_2, \ldots, t_k) : f\}$$

- za koji važi:
  - promenljive $t_1, t_2, \ldots, t_k$ su **n-torne promenljive** ili **indeksirane n-torne promenljive** oblika $t[i]$ gde je $i$ redni broj atributa torke $t$
  - skup n-tornih promenljivih čini tzv. **ciljnu listu**
  - formula $f$ je tzv. **kvalifikacioni izraz**
  - slobodne promenljive u kvalifikacionom izrazu su one i samo one koje čine ciljnu listu

# Relacioni račun n-torki

- Kvalifikacioni izraz se gradi od atoma i operacija:
  - (a1) Ako je $R$ relacija a $s$ n-torna promenljiva vezana za relaciju $R$ onda oznaka $R(s)$ označava to vezivanje i zove se **atom pripadnosti**
  - (a2) Ako su $s$ i $p$ n-torne promenljive, $a$ neka konstanta i $\Theta$ operacija poređenja, onda oznake $s[i]\ \Theta\ p[j]$ i $s[i]\ \Theta\ a$ predstavljaju poređenje $i$-te komponente promenljive $s$ sa $j$-tom komponentom promenljive $p$ ili konstantom $a$ i nazivaju se **atomi poređenja**
  - (f1) ...
  - (f2) ...
  - (f3) ...
  - (f4) ...
  - (f5) ...

# Relacioni račun n-torki (3)

- Kvalifikacioni izraz se gradi od atoma i operacija:
  - (a1) ...
  - (a2) ...
  - (f1) svaki atom je **formula**; sve n-torne promenljive atoma su slobodne u formuli
  - (f2) ako su $f$ i $g$ formule, onda su formule i: $f$ *AND* $g$, $f$ *OR* $g$, *NOT* $f$; pojavljivanje promenljive u formuli je tačno onakvo kakvo je u $f$ i $g$
  - (f3) ako je $f$ formula, onda su formule i: $(\exists s)(f)$ i $(\forall s)(f)$; promenljiva $s$ je vezana u toj formuli a slobodna u $f$
  - (f4) ako je $f$ formula, onda je formula i $(f)$
  - (f5) ništa drugo nije formula relacionog računa n-torki

# Primeri upita

- Naredni upit izdvaja muškarce koji imaju osnovu plate manju od 32000:

$$\{x : RADNIK(x) AND\ x[5] =' M'\ AND\ x[7] < 32000\}$$

  - $x[5]$ je pol, $x[7]$ je osnova plate
- Imena, prezimena i datum zaposlenja radnika koji zarađuju bar 40000:

$$\{(x[3], x[4], x[6]) : RADNIK(x) AND\ x[7] \geq 40000\}$$

  - $x[3]$ je ime, $x[4]$ je prezime
  - $x[6]$ je datum zaposlenja, $x[7]$ je osnova plate

# Relacioni račun n-torki – sintaksa

- Uvodi se sintaksa prilagođena radu na računaru
- Ukratko:
  - umesto indeksiranja kolona koriste se imena kolona iza tačke
    - umesto $x[i]$ koristimo $x.kolona_i$
  - umesto zapisa u zagradi $\{(t_1, t_2, \ldots, t_k) : f\}$ koristimo zapis:
    - $t_1, t_2, \ldots, t_k$ WHERE $f$
  - u formulama se koriste logički operatori AND, OR, NOT i predikati
    - EXISTS promenljiva (formula)
    - FORALL promenljiva (formula)
  - pre zapisa formule vezujemo imena promenljivih za domene, tj. relacije
    - RANGE OF promenljiva IS relacija
    - obično podrazumevamo da su promenljive oblika $k, k_1, k_2, \ldots$ vezane za relaciju $k$

# Primeri upita

- Naredni upit izdvaja muškarce koji imaju osnovu plate manju od 32000:

  radnik WHERE radnik.pol='M' AND radnik.osnova_plate $<$ 32000

- Imena, prezimena i datum zaposlenja radnika koji zarađuju bar 40000:

  radnik.ime, radnik.prezime, radnik.dat_zaposlenja WHERE radnik.osnova_plate $\geq$ 40000

# Primeri upita 1.1

- Imena i prezimena zaposlenih u "Planiranju"
- Relaciona algebra:

$$(ORG\_JEDINICA[naziv =' Planiranje'] * RADNIK)[ime, prezime]$$

# Primeri upita 1.2

- Imena i prezimena zaposlenih u "Planiranju", čiji staž ulazi u kategoriju "Srednja"
- Relacioni račun n-torki:
  $radnik1.ime, radnik1.prezime$
  WHERE EXISTS $org\_jedinica1(radnik1.org\_jed_id = org\_jedinica1.org\_jed_id \wedge org\_jedinica1.naziv =' Planiranje')$

# Značaj formalnih upitnih jezika

- Iz ugla teorije, ključno je da se utvrdi da je upitni jezik dovoljno dobar da se njime može iskazati bilo koji upit
- U praksi je potrebna i efikasnost
- *Optimizacija upita* obuhvata poslove analize upita i pronalaženja najefikasnijeg puta za njegovo izračunavanje
- Relacioni upitni jezici omogućavaju relativno jednostavno automatsko modifikovanje, pa time i optimizaciju, imajući u vidu formalnu zasnovanost
- *Optimizacija baze podataka* obuhvata poslove analize stvarne ili procenjene upotrebe baze podataka i projektovanja fizičkog modela baze podataka u cilju omogućavanja sveukupno efikasnijeg rada sistema

# Ažuriranje baze podataka

- Pored upita, važno mesto zauzima ažuriranje baze podataka
- Ima više pristupa, ali suština je da zahteva proširenja i nove pojmove ili promene definicija postojećih pojmova:
  - **relaciona promenljiva** = promenljiva relacije
  - **relacija** = vrednost relacije = sadržaj relacione promenljive
  - **promenljiva baze podataka** = niz promenljivih relacija
  - **baza podataka** = vrednost baze podataka = sadržaj promenljive baze podataka

# Ažuriranje baze podataka

- **Ažuriranje baze podataka** se definiše kao zamenjivanje vrednosti promenljive baze podataka novom vrednošću baze podataka
  - Promenljivost podataka baze podataka se na taj način apstrahuje sekvencom vrednosti promenljive baze podataka koje ona dobija u različitim vremenskim tačkama (trenucima)
  - Takva definicija omogućava i razmatranje saglasnih konkurentnih promena

# Sadržaj

# Integritet i konzistentnost

- **Konzistentnost** baze podataka je *saglasnost* sadržaja baze podataka sa stvarnim prostorom koji se bazom podataka modelira
  - predstavlja *tačnost* ili *ispravnost* sadržaja baze podataka
  - baza podataka nije konzistentna ako sadrži netačne ili nepotpune podatke
- **Integritet** baze podataka podrazumeva *logičku ispravnost* sadržaja baze podataka
  - logička ispravnost se ostvaruje proveravanjem ispunjenosti logičkih pravila integriteta
- U praksi provera integriteta predstavlja deo provere konzistentnosti koji može da se formalizuje / automatizuje

# Integritet i konzistentnost

- Konzistentnost baze podataka se ostvaruje na dva osnovna načina:
  - automatskim staranjem o integritetu
    - SUBP automatski proverava integritet baze podataka proveravanjem ispunjenosti definisanih pravila integriteta
  - implementacijom transakcionih mehanizama
    - SUBP se stara da svaka jedinica promene (transakcija) bude ili u potpunosti izvedena ili da ne ostavi nikakav trag (u slučaju prekida)
- Naravno, konzistentnost u velikoj meri zavisi i od ispravnosti konkretnih podataka i konkretnih izvedenih jedinica promena
  - taj deo odgovornosti ne može da se poveri SUBP-u

# Integritetni deo modela

- Teorijski model baze podataka može da bude od velike pomoći pri definisanju i ostvarivanju pravila integriteta
- **Integritetni deo modela** čine koncepti i mehanizmi koji omogućavaju da se automatizuje proveravanje zadovoljenosti određenih uslova
- Relacioni model je u tome posebno dobar

# Integritetni deo relacionog modela

- Baza podataka (tj. njena vrednost) se opisuje relacijama
- Uslovi integriteta u relacionom modelu se opisuju predikatima (istinotosnim formulama) nad relacijom ili bazom podataka
- Formalnost strukturnog dela modela olakšava formulisanje uslova integriteta

# Vrste uslova integriteta

- Opšti uslovi integriteta (implicitni)
  - podrazumevaju se na nivou modela i implementacije SUBP
  - odgovarajuća logička pravila su ugrađena u model i implementaciju
    - ne definišu se eksplicitno za svaku relaciju ili atribut
    - pri određivanju strukture baze podataka se određuju i konkretne specifične instance (tj. parametri primene) tih uslova
- Specifični uslovi integriteta (eksplicitni)
  - oni koji se odnose na pojedinu relaciju, atribut ili bazu podataka
  - određuju se pri određivanju strukture baze podataka
  - eksplicitno se navode odgovarajući logički uslovi

# Opšti uslovi integriteta

- Najvažnije vrste logičkih pravila integriteta
  - Integritet domena
    - pravilo: svaka vrednost nekog atributa u bazi podataka mora pripadati odgovarajućem domenu atributa
  - Integritet (primarnog, jedinstvenog) ključa
    - pravilo: jedinstvenost torki u relaciji se ostvaruje definisanjem primarnog ključa, čija vrednost jednoznačno određuje torku relacije
  - Integritet stranog ključa
    - pravilo: integritet odnosa među relacijama se ostvaruje definisanjem stranog ključa, čija vrednost jednoznačno određuje torku povezane relacije
- Za svaki konkretan slučaj (atribut, relacija, odnos) se definišu konkretne pojedinosti

# Integritet domena

- *"Svaka relacija mora da ima tačno određen domen"*
  - Svaka vrednost nekog atributa u bazi podataka mora pripadati odgovarajućem domenu atributa
  - Posledica strukturnog dela modela:
    - atribut je funkcija koja slika entitet u zadati domen atributa
- Tehnički se ostvaruje određivanjem domena pri definisanju atributa u okviru naredbe za pravljenje tabele baze podataka
  - Domen je neki od podržanih tipova baze podataka i može da obuhvata
    - dužinu podatka
    - opcionu deklaraciju jedinstvenosti
    - opcionu deklaraciju podrazumevane vrednosti
    - (većina implementacija omogućava i deklarisanje da li domen obuhvata i tzv. nedefinisane vrednosti)
  - Neke implementacije omogućavaju i korisnički definisane tipove - UDF

# Integritet domena

- U slučaju RSUBP DB2 integritet domena se implementira određivanjem tipova kolona pri pravljenju tabela:
  CREATE TABLE <ime tabele> (
  ...
  <ime kolone> <tip kolone>... ,
  ...
  )...

# Pojam ključa

- Podsetimo se:
  - **Natključ** K relacije R je podskup njenih atributa $K \subseteq A(R)$ za koji važi:
    - skup atributa K jednoznačno određuje torke relacije
  - **Ključ** (ili **ključ-kandidat**) relacije R je natključ K za koji važi:
    - ne postoji pravi podskup skupa K koji je natključ relacije R
- Primetimo:
  - Svaka relacija ima najmanje ključ, a može da ih ima više
  - Svaki natključ sadrži bar jedan ključ
- Posledice:
  - Za svake dve torke x i y relacije R, važi $K(x) \neq K(y)$
  - Projekcija relacije R na atribute ključa K ima isti broj elemenata kao i R

# Vrste ključeva

- Još neke važne pojmove u vezi sa ključevima:
  - *Primarni ključ* – jedan izabran ključ se proglašava za primarni
    - SUBP proverava jedinstvenost torki relacije proveravanjem jedinstvenosti vrednosti primarnog ključa
  - *Alternativni ključ* ili *jedinstveni ključ* – svaki ključ osim primarnog
  - *Strani ključ* – skup kolona koji predstavlja ključ povezane relacije
    - vrednost stranog ključa jednoznačno određuje torku povezane relacije
  - *Složeni ključ* – ključ koji se sastoji od više od jednog atributa
  - *Objedinjeni ključ* – složeni ključ koji se sastoji od dva ili više stranih ključeva (i samo od njih)
  - *Surogatski ključ* – veštački uveden atribut sa ulogom ključa
    - uvodi se kada relacija nema prirodan ključ ili kada je prirodan ključ nepraktičan za upotrebu (ima mnogo atributa ili je nezgodan tip)

# Integritet primarnog ključa

- "*Svaka relacija ima bar jedan ključ*"
  - Posledica je strukturnog dela modela:
    - relacija je skup, pa mora da sadrži jedinstvene vrednosti
- Tehnički se ostvaruje definisanjem tačno jednog *primarnog ključa* pri pravljenju tabele
  - Za primarni ključ se proglašava jedan od ključeva relacije
    - za koji se očekuje da će najčešće biti upotrebljavan za pristupanje podacima
    - čiji domen ne sadrži nedefinisane vrednosti
  - Primarni ključ može da se sastoji od jedne ili više kolona
  - SUBP koristi primarni ključ za proveravanje jedinstvenosti torki relacije

# Integritet primarnog ključa

- U slučaju RSUBP DB2 integritet primarnog ključa se implementira određivanjem primarnog ključa pri pravljenju tabele:
  CREATE TABLE <ime tabele> (
  ...
  [CONSTRAINT <ime ključa>]
  PRIMARY KEY ( <lista imena kolona> ) ...
  ...
  )...

# Integritet jedinstvenosti

- Integritetu primarnog ključa se pridružuje i *integritet jedinstvenosti* (naziva se i *integritet entiteta*):
  - "*Primarni ključ ne sme da sadrži nedefinisane vrednosti, niti dve torke jedne relacije smeju imati iste vrednosti primarnih ključeva*"
- U doslednim implementacijama to je isto kao integritet primarnog ključa
  - primarni ključ po definiciji ne sme da sadrži nedefinisane vrednosti
  - jedinstvenost vrednosti primarnog ključa je implicirana jedinstvenošću torki u relaciji
- Ipak, u nedoslednim implementacijama:
  - integritet jedinstvenosti je jači od integriteta ključa
    - zato što postoji podrška za nedefinisane vrednosti
  - mogu da postoje dve identične torke u relaciji

# Integritet jedinstvenog ključa

- Pored primarnog ključa, mogu da se eksplicitno deklarišu i *jedinstveni ključevi*
  - imaju iste karakteristike kao primarni ključevi
    - načelno *iste* a u praksi mogu da budu *slične*
  - očekuje se da se koriste pri referisanju torki ređe od primarnog ključa
- Osnovna namena jedinstvenog ključa je implementacija dodatnih uslova integriteta jedinstvenosti torki
  - ako relacija ima više ključeva, jedan se proglašava za primarni a ostali (alternativni ključevi) mogu da se definišu kao jedinstveni

# Integritet jedinstvenog ključa

- U slučaju RSUBP DB2 integritet jedinstvenog ključa se implementira definisanjem jedinstvenog ključa pri pravljenju tabele:
  CREATE TABLE <ime tabele> (
  ...
  [CONSTRAINT <ime ključa>]
  UNIQUE ( <lista imena kolona> ) ...
  ...
  )...

# Integritet jedinstvenog ključa

- U nekim slučajevima može da bude potrebno da se proverava jedinstvenost ključeva koji mogu da imaju nedefinisane vrednosti
  - tj. ne ispunjava uslov integriteta jedinstvenosti
  - na primer, ako za neki atribut X nije moguće odrediti vrednosti za sve torke, ali one koje su poznate moraju da budu jedinstvene
- Dosledne implementacije to ne omogućavaju
  - ali neke nude zaobilazni put
  - na primer, na DB2 može da se napravi indeks koji uz indeksiranje obavlja i proveru jedinstvenosti definisanih vrednosti:
    - CREATE UNIQUE INDEX ... EXCLUDE NULL KEYS ...

# Referencijalni integritet

- Referencijalni integritet predstavlja uslove o međusobnim odnosima koje moraju da zadovoljavaju torke dveju relacija:
  - *ne sme se obrisati torka na koju se odnosi neka torka neke relacije u bazi podataka, niti se sme tako izmeniti da referenca postane neispravna*
  - *ne sme se dodati torka sa neispravnom referencom (takvom da ne postoji torka na koju se odnosi)*
- U relacionom modelu se referencijalni integritet ostvaruje putem integriteta stranog ključa

# Referencijalni integritet

- U slučaju nedefinisane vrednosti, uslovi referencijalnog integriteta su izmenjeni:
  - *ne sme se obrisati torka na koju se odnosi neka torka neke relacije u bazi podataka, niti se sme tako izmeniti da referenca postane neispravna*
  - *ne sme se dodati torka sa neispravnom referencom (takvom da ne postoji torka na koju se odnosi)*
  - *referenca koja sadrži nedefinisane vrednosti je ispravna (i ne referiše ništa) akko je u potpunosti nedefinisana (tj. svi njeni atributi su nedefinisani)*
- U većini implementacija treće pravilo je dodatno relaksirano:
  - *referenca koja sadrži nedefinisane vrednosti je ispravna (i ne referiše ništa) akko je makar jedan njen deo nedefinisan*

# Integritet stranog ključa

- Skup FK atributa relacije R je njen **strani ključ** koji se odnosi na *baznu relaciju* B akko važe sledeći uslovi:
  - relacija B ima primarni ključ PK
  - domen ključa FK je identičan domenu ključa PK
  - svaka vrednost stranog ključa FK u torkama relacije R je identična ključu PK bar jedne torke relacije B
    - (integritet ključa implicira jedinstvenost)
- Za relaciju R se kaže da je *zavisna* od bazne relacije B
- Bazna relacija B se naziva i roditeljskom relacijom

# Integritet stranog ključa

- U slučaju nedefinisane vrednosti, skup uslova je izmenjen:
  - relacija B ima primarni ključ PK
  - domen ključa FK je identičan domenu ključa PK, ili je proširen nedefinisanim vrednostima
  - svaka vrednost ključa FK u torkama relacije R je ili nedefinisana ili je identična ključu PK bar jedne torke relacije B
  - vrednost stranog ključa FK u torkama relacije R je nedefinisana akko sve vrednosti atributa stranog ključa imaju nedefinisanu vrednost
- U većini implementacija je četvrto pravilo dodatno relaksirano:
  - vrednost stranog ključa FK u torkama relacije R je nedefinisana ako bar jedan atribut stranog ključa ima nedefinisanu vrednost

# Integritet stranog ključa

- Poštovanje integriteta stranog ključa pri menjanju sadržaja baze podataka se određuje:
  - pravilom brisanja (bira se tačno jedno) i
  - pravilom ažuriranja (bira se tačno jedno)

# Pravilo brisanja

- Pravila brisanja (bira se tačno jedno): "U slučaju pokušaja brisanje torke bazne relacije $B$, za koju postoji zavisna torka relacije $R$ (ona čiji je strani ključ jednak primarnom ključu torke koja se pokušava obrisati) onda..."
  - aktivna zabrana brisanja – RESTRICT
    - "...zabranjuje se brisanje torke iz relacije $B$."
  - pasivna zabrana brisanja – NO ACTION
    - slično kao aktivna zabrana, ali se odlaže provera do samog kraja brisanja, zato što možda postoji linija kaskadnih pravila koja će obrisati zavisnu torku
  - kaskadno brisanje – CASCADE
    - "...brišu se sve odgovarajuće zavisne torke $R$."
  - postavljanje nedefinisanih vrednosti – SET NULL
    - "...u svim zavisnim torkama relacije $R$ atributi stranog ključa se postavljaju na nedefinisane vrednosti."
  - postavljanje podrazumevanih vrednosti – SET DEFAULT
    - "...u svim zavisnim torkama relacije $R$ atributi stranog ključa se postavljaju na podrazumevane vrednosti."

# Pravilo ažuriranja

- Pravila ažuriranja (bira se tačno jedno): "Ako se pokuša menjanje primarnog ključa torke relacije $B$, za koju postoji zavisna torka relacije $R$ (ona čiji je strani ključ jednak primarnom ključu torke koja se pokušava obrisati) onda..."
  - aktivna zabrana menjanja – RESTRICT
    - "...zabranjuje se menjanje torke relacije $B$."
  - pasivna zabrana menjanja – NO ACTION
    - slično kao aktivna zabrana, ali se odlaže provera do samog kraja menjanja, zato što možda postoji linija kaskadnih pravila koja će promeniti zavisnu torku;
  - kaskadno menjanje – CASCADE
    - "...na isti način se menjaju atributi stranog ključa svim zavisnim torkama relacije $R$."
  - postavljanje nedefinisanih vrednosti – SET NULL
    - "...u svim zavisnim torkama relacije $R$ atributi stranog ključa se postavljaju na nedefinisane vrednosti."
  - postavljanje podrazumevanih vrednosti – SET DEFAULT
    - "...u svim zavisnim torkama relacije $R$ atributi stranog ključa se postavljaju na podrazumevane vrednosti."

# Integritet stranog ključa

- U slučaju sistema DB2 strani ključ se definiše pri pravljenju tabele:
- CREATE TABLE <ime tabele>(
  ...
  [CONSTRAINT <ime ključa>]
  FOREIGN KEY( <lista imena kolona> )
  REFERENCES <ime tabele>[ ( <lista imena kolona> ) ]
  [ON DELETE <pravilo brisanja>]
  [ON UPDATE <pravilo ažuriranja>]
  ...
  )...
- Pravilo brisanja može da bude:
  - RESTRICT, NO ACTION, CASCADE, SET NULL
- Pravilo ažuriranja može da bude:
  - RESTRICT, NO ACTION
- U oba slučaja podrazumevano pravilo je NO ACTION

# Specifični uslovi integriteta

- *Specifični uslovi integriteta* se definišu u obliku logičkih pravila koja uvek moraju da budu ispunjena
- U odnosu na opseg podataka koji se proveravaju, uslovi mogu da budu definisani na:
  - atributu
  - torki
  - relaciji
  - bazi podataka

# Uslovi integriteta na atributu

- *Uslovi integriteta na atributu*
  - lokalnog je karaktera, odnosi se na vrednost jednog atributa jedne torke
  - uslov može da zavisi samo od vrednosti atributa
  - može da se koristi za dodatno sužavanje domena
    - na primer, ako celobrojni atribut mora da bude u zadatom opsegu
  - može da se koristi za proveru ispravnosti složenih tipova podataka
    - na primer, ako tekstualni podatak mora da ima neku zadatu formu

# Uslovi integriteta na atributu

- Definiše se pri opisivanju atributa tabele:
  CREATE TABLE <ime tabele>(
  ...
  <ime atributa> <tip atributa> CHECK (<logički uslov>)
  ...
  ) ...
- Navedeni logički uslov
  - sme da referiše samo taj jedan atribut za koji se definiše
  - mora da bude ispunjen za svaki red tabele
  - u slučaju nedefinisanih vrednosti uslov mora da bude *ne-lažan*
- Podržano u većini implementacija SUBP

# Uslovi integriteta na atributu

- Primer:
  CREATE TABLE ...
  (
  GODINA INT CHECK ( GODINA BETWEEN 1980 AND 2100 ),
  ... )

# Uslovi integriteta na torki

- *Uslovi integriteta torke*
  - lokalnog karaktera, odnosi se na vrednost jedne torke
  - uslov može da zavisi od vrednosti svih atributa torke
  - koristi se za proveru ispravnosti složenijih saglasnosti atributa u okviru jedne torke
    - na primer, ako se navodi neki interval, može da se proverava da li je početna vrednost intervala manja od krajnje vrednosti intervala

# Uslovi integriteta na torki

- Definiše se pri pravljenju tabele:
  CREATE TABLE <ime tabele>(
  ...
  CONSTRAINT [<ime uslova>] CHECK (<logički uslov>)
  ...
  ) ...
- Navedeni logički uslov
  - sme da referiše sve atribute relacije
  - mora da bude ispunjen za svaki red tabele
  - u slučaju nedefinisanih vrednosti uslov mora da bude *ne-lažan*
- Podržano u većini implementacija SUBP

# Uslovi integriteta na torki

- Primer:
  CREATE TABLE ... (
  ... GOD_OD INT,
  GOD_DO INT,
  CONSTRAINT OPSEG CHECK ( GOD_OD <= GOD_DO )
  ..
  . )...

# Uslovi integriteta na relaciji

- *Uslovi integriteta relacije*
  - globalnog karaktera, može da se odnosi na sve torke jedne relacije
  - uslov može da zavisi od vrednosti svih atributa svih torki
  - koristi se za proveru ispravnosti složenijih saglasnosti vrednosti u okviru jedne relacije
    - na primer, može da se proverava da li broj torki ili suma po nekom atributu zadovoljavaju određene uslove

# Uslovi integriteta na relaciji

- Definiše se pri pravljenju tabele:
  CREATE TABLE <ime tabele>(
  ...
  CONSTRAINT [<ime uslova>] CHECK (<logički izraz sa podupitima>)
  ...
  ) ...
- Alternativna sintaksa je ista kao za uslove integriteta na bazi podataka
- Obično su implementirani slično kao i uslovi integriteta na torkama
  - logički izraz počiva na podupitima nad samom tom relacijom
  - mora da bude ispunjen za tabelu kao celinu
  - u slučaju nedefinisanih vrednosti uslov mora da bude *ne-lažan*
- Ne podržava ih većina implementacija

# Uslovi integriteta na relaciji

- Primer:
  CREATE TABLE T1 (
  ...
  A INT,
  B INT,
  CONSTRAINT C1 CHECK ( (select sum(A) from t1) < (select sum(B) from t1) )
  ...
  )...

# Uslovi integriteta na bazi podataka

- *Uslovi integriteta na bazi podataka*
  - globalnog su karaktera, odnose se na sadržaj različitih relacija
  - koriste se za proveru složenijih uslova integriteta
    - na primer, ako sadržaj jedne relacije mora da bude u skladu sa sadržajem druge relacije, ali na način koji prevazilazi mogućnosti običnog referencijalnog integriteta

# Uslovi integriteta na bazi podataka

- Definiše se posle pravljenja tabela na koje se odnosi:
  CREATE ASSERTION <ime> CHECK (<logički izraz sa podupitima>)
- Pri tome:
  - logički izraz počiva na podupitima nad bilo kojim definisanim relacijama
  - mora da bude ispunjen za bazu podataka kao celinu
  - u slučaju nedefinisanih vrednosti uslov mora da bude *ne-lažan*
- Ne podržava ih većina implementacija

# Uslovi integriteta na bazi podataka

- Primer:
  CREATE ASSERTION ASRT_1 CHECK (
  (SELECT COUNT(*) FROM T1)
  <
  (SELECT COUNT(*) FROM T2) )

# Kada se proveravaju uslovi?

- Uslovi se proveravaju pri svakoj promeni sadržaja baze podataka
- Nekada je potrebno da se uslovi proveravaju tek na kraju transakcije
- Većina RSUBP to omogućava na neki način
  - može da se menja podrazumevani način rada
  - može da se menja način rada u tekućoj transakciji
- Naredna naredba (DB2, Oracle, PostgreSQL,...), odlaže sve provere u tekućoj transakciji odlažu do njenog završetka:
  SET CONSTRAINTS ALL DEFERRED
- *postoji veći broj opcija, mogu se birati tabele i uslovi koji se odlažu

# Aktivno održavanje integriteta

- *Aktivno* održavanje integriteta podrazumeva da se integritet održava tako što se na određene promene reaguje pokretanjem eksplicitno definisanog programskog koda
  - taj programski kod pravi neophodna usklađivanja povezanih podataka
- Termin *aktivno* potiče od izvršavanja eksplicitno zadatih programa
- Mehanizam aktivnog održavanja integriteta su *okidači* (engl. trigger)

# Okidači

- Okidači podrazumevaju da se na određene promene reaguje pokretanjem odgovarajućeg programskog koda
- Svaki okidač je definisan
  - relacijom na kojoj se prate promene
  - vrstom promene na koju se reaguje
  - programskim kodom koji se izvršava
  - trenutkom izvršavanja – pre ili posle promene
  - granularnošću izvršavanja – da li se izvršava po jedan put za svaku promenjenu torku ili po jedan put za celu naredbu menjanja
  - podacima koji se referišu u programskom kodu

# Okidači

- Vrsta promene može da bude
  - INSERT – dodavanje torki relaciji
  - UPDATE – menjanje torki relacije
  - DELETE – brisanje torki relacije
- Trenutak izvršavanja može biti
  - BEFORE – pre promene
  - AFTER – posle promene
- Referisanje torki može biti
  - OLD – stara vrednost pre promene (ne može za INSERT)
  - NEW – nova vrednost posle promene (ne može za DELETE)
- Granularnost može da bude
  - FOR EACH ROW – izvršava se za svaki promenjen red
  - FOR EACH STATEMENT – izvršava se za svaku izvršenu naredbu

# Okidači

- Programski kod koji je vezan za okidač se definiše na različite načine
- U slučaju sistema DB2 to može da bude:
  - jedna naredba SQL-a
  - blok naredbi SQL-a
  - serverska procedura
  - i drugo
- Blok naredbi SQL-a počinje sa BEGIN i završava se sa END

# Definisanje okidača

- Sintaksa naredbe (pojednostavljena):
  CREATE [OR REPLACE] TRIGGER <ime okidača>
  { [NO CASCADE] BEFORE | AFTER }
  { INSERT | UPDATE | DELETE }
  ON <ime tabele>
  [ REFERENCING { OLD | NEW | OLD TABLE | NEW TABLE } AS <ime>]*
  FOR EACH { ROW | STATEMENT }
  [ WHEN <uslov> ]
  <programski kod>

# Primer okidača

- Primer okidača: automatski izračunava osnovnu platu novog službenika na osnovu nivoa obrazovanja:
  CREATE OR REPLACE TRIGGER insert_set_salary
  NO CASCADE BEFORE INSERT ON employee
  REFERENCING NEW AS N
  FOR EACH ROW
  BEGIN
  SET n.salary = CASE n.edlevel WHEN 18 THEN 50000 WHEN 16 THEN 40000
  ELSE 25000
  END

# Primer okidača

Primer okidača: ako bi u odeljenjima postojao broj radnika u odeljenju, on bi mogao da se ažurira okidačima pri dodavanju i brisanju radnika:

```
CREATE OR REPLACE TRIGGER insert_radnik
AFTER INSERT ON radnik
REFERENCING NEW AS N FOR EACH ROW
BEGIN
UPDATE org_jedinica
SET broj_radnika = broj_radnika + 1
WHERE org_jed_id = n.org_jed_id
END
```

# Primer okidača

```
CREATE OR REPLACE TRIGGER delete_radnik
BEFORE DELETE ON radnik
REFERENCING OLD AS R FOR EACH ROW
BEGIN
UPDATE org_jedinica
SET broj_radnika = broj_radnika - 1
WHERE org_jed_id = r.org_jed_id
END
```

# Kada koristimo aktivno održavanje integriteta?

- Radi održavanja ispravnih vrednosti povezanih podataka u bazi podataka
- Radi proveravanja integriteta
- Radi izvođenja akcija van baze podataka

# Okidači i održavanje podataka u bazi

- Održavanje ispravnih vrednosti podataka u bazi podataka pomoću okidača je najčešće *loša ideja*
  - to je signal da u bazi podataka postoje redundantni podaci, što je potencijalno ozbiljan problem
- Obično se opravdava performansama
  - na primer, da ne bismo stalno iznova izračunavali prosečnu ocenu, možemo da je dodamo kao kolonu i održavamo okidačima
  - ali tada postoje i drugačija, potencijalno bolja rešenja, npr. materijalizovani pogledi

# Okidači i proveravanje integriteta

- U odsustvu mogućnosti definisanja specifičnih pravila integriteta na nivou tabele ili baze podataka, mogu da se koriste okidači
- Ideja je da se pri nekoj izmeni proverava da li je sve u redu i prijavljuje se greška (i prekida transakcija) ako nije:
  ...SIGNAL SQLSTATE 'ER001' SET MESSAGE_TEXT = 'Neispravno menjanje!'

# Okidači i akcije van baze podataka

- Upotreba okidača radi izvođenja akcija van baze podataka može da bude dobar način za povezivanje baze podataka sa drugim elementima informacionog sistema
  - na primer, kada stanje proizvoda u skladištu padne ispod određenog nivoa, okidač može da pošalje imejl za naručivanje proizvoda od dobavljača
  - ili, ako je potrebno ažurirati sadržaj neke druge baze podataka

# Okidači na pogledima

- Savremeni SUBP nude okidače na pogledima
  - Izvršavaju se *umesto* naredbe za dodavanje, menjanje ili brisanje torki
  - Omogućavaju preusmeravanje izmena na relacije na kojima počiva pogled, ali i dalje od toga, na sasvim druge relacije
- Primer okidača na pogledu: Ako pogled V1 počiva na relacijama T1 i T2, okidačem se rešava kako se menjaju tabele "kroz pogled"

```
CREATE TRIGGER V1_UPDATE
INSTEAD OF UPDATE ON V1
REFERENCING NEW AS n OLD AS o
FOR EACH ROW
UPDATE T1 SET (c1, c2) = (n.c1, n.c2)
WHERE c1 = o.c1 AND c2 = o.c2
```

# Okidači na pogledima

- Okidači na pogledima omogućavaju skrivanje veoma složenih operacija kojima se spoljašnja shema razdvaja od konceptualne, ili konceptualna od interne
- Primer okidača na pogledu: implementira skriveno implicitno enkriptovanje lozinki pri zapisivanju u bazu podataka:

```
CREATE TRIGGER UPDATE_LOGINS
INSTEAD OF UPDATE ON LOGINS
REFERENCING OLD AS o NEW AS n
FOR EACH ROW
UPDATE USERS U
SET system = n.system, login = n.login, password = encrypt(n.password)
WHERE system = o.system AND login = o.login AND U.user = USER
```

# Okidači na pogledima i razdvajanje nivoa modela

- Osnovno sredstvo razdvajanja nivoa shema u relacionom modelu su pogledi
  - Na nižem nivou (interna ili konceptualna shema) relacije su normalizovane radi stabilnosti i nerredundantnosti
  - Na višem nivou (konceptualna ili spoljašnja shema) relacije mogu da budu denormalizovane, tj. spajaju se radi lakšeg postavljanja upita i pristupanja podacima
  - Privilegije na pogledima mogu da se potpuno razlikuju od privilegija na relacijama
- Osnovno ograničenje pogleda:
  - Pogledi nad više relacija ne mogu da se ažuriraju
  - To značajno ograničava primenu pogleda u razdvajanju nivoa modela
- Okidači na pogledima potpuno prevazilaze to ograničenje:
  - Omogućavaju potpunu kontrolu nad upotrebom pogleda za razdvajanje nivoa modela
  - Upotpunjuju i zaokružuju relacioni model

# Rezime relacionog modela

- Puna matematička zasnovanost
  - neprotivrečan i nedvosmislen model
- Jedinstveni potpun koncept relacije
  - relacija modelira i entitete i odnose
  - tj. sav sadržaj baze podataka
- Jednostavan koncept integritetnog dela modela
  - stroga formalnost modela omogućava formalnost i preciznost integritetnog dela modela
- Univerzalan pristup na svim nivoima
  - isti model može da se koristi na svim nivoima od interne, preko konceptualne do spoljašnje sheme
  - tehnike relacionog modela mogu da se koriste za potpuno razdvajanje različitih nivoa shema

# Literatura za temu


- Gordana Pavlović-Lažetić, Uvod u relacion e baze podataka, 2. izd. Matematički fakultet, 1999.
- dostupno onlajn: http://poincare.matf.bg.ac.rs/ gordana/urbp-2016.htm
- Ramakrishnan, Gehrke, Database Management Systems, 2. ed, 2000.
- Codd, A relational model of data for large shared data banks, Comm. ACM, 13(6), 1970.
- Codd, Extending the database relational model to capture more meaning, ACM ToDS, 4(4), 1979.
- Codd The Relational Model for Database Management – Version 2, Addison Wesley Publ. Inc., 1990.
- Darwen, Date, The Third Manifesto, 1995.
- IBM, Database Administration Concepts and Configuration Reference, 2012.