# **SBP**

# Realizacija ograničenja šeme RBP putem SUBP

Predavanje 0.

## (Uvod) Realizacija integritentne komponente šeme BP

**Cilj** je da sva naša ograničenja organizujemo na nivou servera BP putem mehanizma SUBP, ideja je da prakticno zastitimo sve naše podatke, tako da odakle god da im pristupimo da ne možemo da ih narušimo.

**Kontrola ograničenja, implementiranih na nivou SUBP je centralna** to u suštini znači da ni jedan softver ni program ne može spolja da ih naruši, korisnici nisu svesni ograničenja sve dok ih ne naruše. Primer prolazimo kroz aerodrom i mi bivamo iskontrolisani a da nismo ni svesni dok ne narušimo neko ograničenje, isto funkcioniše i SUBP. U slučaju da narušimo nešto SUBP će obaviti jednu od ove dve operacije:

* obavestiti nas o greski (korisnicki program i korisnika nas) i prekinuti izvršavanje
* da nekim aktivnim mehanizmom obavi dalje operacije nad BP da bi tu BP vratio u konzistentno stanje

**Implementacija ograničenja BP (na serveru BP),** moramo da definiše parametre u vezi ograničenja (tj da ih saopštimo serveru),

**parametri** su:

* definiše se ograničenje datog tipa
* definišu se operacije tj. događaji koji mogu dovesti do narušavanja ograničenja
* za svaku operaciju je potebno definisati akciju koju ćemo preduzeti da bi očuvali konzistentnost u slučaju narušavanja ograničenja
* svako ograničenje je vezano za određenja obeležja i određene šeme relacija šeme BP
* operacije(događaji) koji mogu dovesti do narušavanja ograničenja:
  + Upis (nove torke u relaciju)
  + Brisanje (neke postojece torke)
  + Modifikacija (postojece torke u relaciji)

Pitanje: Zašto upiti nisu kritična operacija? Upiti ne menjaju stanje, samo konstatuju koji su to podaci u BP, ako su upiti ispravni sve ce biti ok ako nisu ispravni samo će pokazati da nisu ispravni.

Pitanje: Kada možemo da donesemo pretpostavku da će nam upiti vratiti uvek ispravne podatke ? Ako krenemo od početka i ako kontrolišemo svaki upis, brisanje i modifikaciju, posledično svaki naš upit će vratiti konzistentno ispravne podatke. U životu ovo baš nije uvek tako pošto često preuzimamo nasleđene sisteme.

Akcije očuvanja konzistentnosti BP u slučaju pokušaja narušavanja ograničenja, vezuju se sa svaku operaciju koja može narušiti ograničenje, dele se na:

* + Aktivne, obezbeđuju automastsko sprovođenje daljih operacija ažuriranja nad BP, koje će obezbediti očuvanje konzistentnosi BP
  + Pasivne, sprečavaju operaciju koja bi narušila ograničenje
  + Kombinovane, pod određenim uslovima pasivne pod određenim uslovima aktivne

Ograničenja se implementiraju pomoću mehanizama SUBP, imaćemo način da definišemo ograničenje i moraćemo da saopštimo DBMsu kako smo definisali ograničenje, imamo kritične operacije koje mogu narušiti ograničenje opet moramo saopštiti DBMSu koje su kritične operacije, i za savaku kritičnu operaciju moramo saopštiti koja akcija očuvanja konzistentnosti BP će se primeniti

**Kontrola ograničenja šeme BP,** SUBP pokreće mehanizme kontrole važenja ograničenja

* automatski
* nakon izvođenja kritične operacije za ograničenje

**Prednosti**:

* Automatksa kontrola implementiranih ograničenja na nivou SUBP
* Imamo obezbeđenu konzistentnost BP u svakom trenutku
* Standardizacija načina za implementaciju ograničenja (sql2, sql3, etc...)
* Kada postavimo implementaciju ograničenja na nivo servera baze podataka onda možemo da kažemo da nemamo dodatnih razloga zašto bi realizovali ograničenja unutar samih korisničkih programa na srednjem sloju, ili na klijentskom sloju. **Znači implementiramo ograničenja jednom, a ne milion puta na različitim aplikacijama.**

Pitanje: Zašto je ovo jako dobro? Rasterećujemo programere i olakšavamo samu izradu aplikacije kao i njeno održavanje koje jako puno košta.

**Nedostaci:**

* Povišen stepen zavisnosti šeme BP od proizvođača, tipa i verzije SUBP-a, ne podržavaju svi sistemi za upravljanje BP standarde na isti način, standardi generalno nisu idealni, to znači mi ako želimo da koristimo kapacitete DBMS-a za implementaciju ograničenja na nivou servera BP nužno ćemo istrčavati izvan onoga što definiše standard, znači ako želimo da migriramo bazu sa jednog na drugi DBMs moraćemo da preradimo te delove programskog koda, postajemo više zavisni od šeme BP i proizvodjaca tipa i verzije. **Zato je izbor DBMS-a strateski potez jedne kompanije!**
* Ostaje uvek potreba za realizacijom nekih ograničenja unutar programa, zbog obezbedjenja pogodnosti programa za upotrebu, i dvostruke implementacije i kontrole ograničenja. Recimo ako pravimo interfejse bolje pogodnosti (HCI priča) obično želimo da obavestimo korisnika da pravi grešku u trenutku kada je pravi a ne kada klikne submit. Onda imamo potrebu da ista ograničenja implementiramo i na frontu a ne samo na serveru BP.

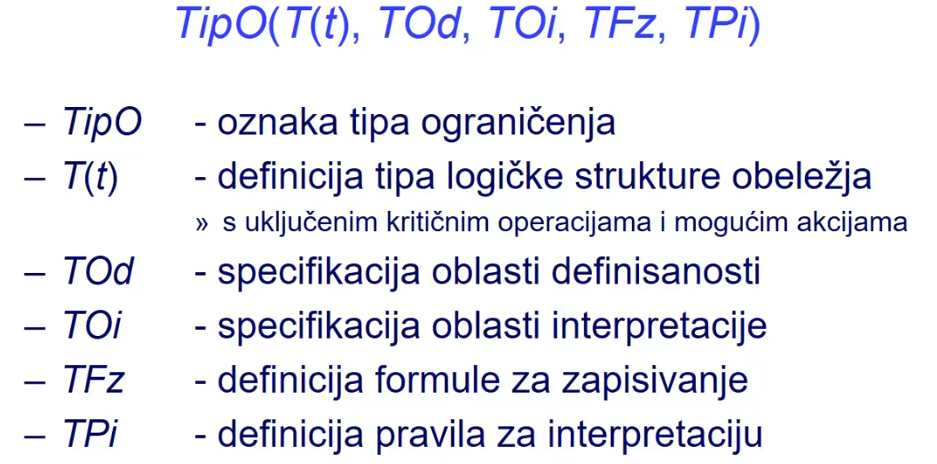
Predavanje 1.

## Specifikacija tipa ograničenja

**Karakteristike tipa ograničenja u modelu podataka (MP)**

* Oblast definisanosti, to je tip logičke strukture nad kojom se ograničenje definiše
* Oblast interpretacije, to je tip logičke strukture nad kojom se ograničenje interpretira
* Formalizam za zapisivanje, to je nekakv parametrizovani šablon na osnovu kog pišemo ograničenja
* Pravilo za interpretaciju odnosn ovalidaciju
* Skup kritičnih operacija nad bazom podataka koje mogu dovesti do narušavanja ograničenja datog tipa, mi idemo sa pretpostavkom da imamo praznu bazu podataka
* Skup mogućih akcija, definiše se za svaku kritičnu operaciju

**Specifikacija tipa ograničenja u MP**



* Sve komponente specifikacije TipO treba da budu specificirane formalno( u cilju njihove lakše programske implementacije). U ove potrebe potrebno je koristiti koncepte nekog konkretnog MP (mi ćemo koristiti primer relacionog modela podataka).
* TipO – oznaka tipa ograničenja, to je nekakav niz znakova, ne poste neka konkretna pravila kako zadajemo taj niz znakova ali očekujemo da taj niz znakova jedinstveno identifikuje svaki uvedeni tip ograničenja, formira se sasvim slobodno na osnovu naziva tipa ograničenja u datom MP

## Tipovi ograničenja u RMP

**Mogući tipovi ograničenja u RMP**



* + - * Specijalne vrste zavisnosti sadrzavanja ( ograničenje referencijalnog integriteta i ograničenje inverznog referencijalnog integriteta)
      * , to su korisnički definisani, za koje kažemo da u njih spadaju svi koji prethodno nisu definisani, štos je u tome da što manje toga ostane u ovoj kategoriji a da što više ograničenja pokrijemo sa unapred definisanim.

Pitanje: Zašto imamo oviliki spisak definisanih ograničenja? Rasterećujemo projektante, tj u suštini mi smo unapred prepoznali više situacija u realnom sistemu, i lakše ih modelujemo.

Pitanje: Zašto uopšte uvodimo pojam „tip ograničenja“ ? Zato što uvođenjem ovog pojma mi uvodimo znanje o tome kako treba specificirati, u koliko nemamo pojam tipa ograničenja svaki projektant može da radi šta hoće...

## Specifikacija tipa ograničenja

**Specifikacija tipa ograničenja u RMP**



Nad definicijom tipa logičke strukture obeležja se kasnija ograničenja datog tipa definišu, moramo da je specificiramo ne smemo da je preskočimo, to je nekakav skup n-torki, dozvojeno je cak da T(t) bude prazan skup.

Svaki tip logičke strukture obeležja se definiše se skupom uloga:



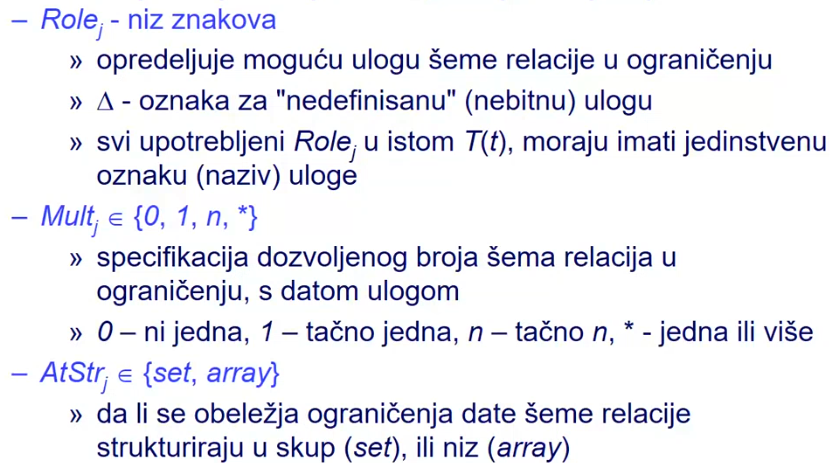
a svaku tu ulogu definišemo jednom ovakvom n-torkom:



to će ustvari biti uloga nekih šema relacija u našem ograničenju.

Definicija strukture elementa:



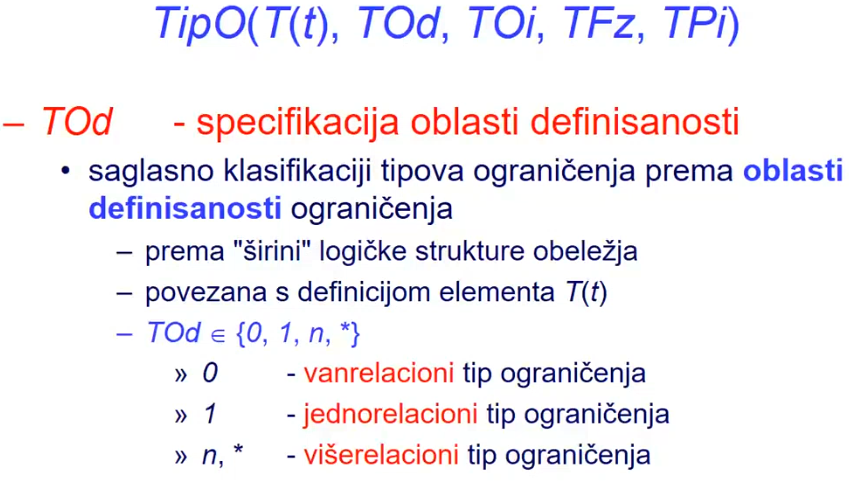
 

**Tipovi akcija očuvanja konzistentnosti**

Pri očuvanju narušavanja ograničenja BP, akcije mogu biti:

* Pasivne akcije
  + NoAction (Restrict) , ova akcija zabranjuje sprovođenje operacije koja bi izavala narušavanje kontrolisanog ograničenja. To znači npr. ako bi neko pokušao nekom operacijom ažuriranja narušio neko ograničenje, ova akcija bi rekla zaustavi i poništi izvršavanje ove operacije.
* Akrivne akcije
  + Cascade, kaskadna propagacija operacije, npr. ako je to bilo brisanje kaskadno ce nastaviti da brise dalje druge relacije.. isto i za modifikaciju itd...
  + SetNull, svodi kriticne vrednosti atributa na nula vrednosti
  + SetDefault, svodi na predefinisanje inicijalne defaut vrednosti
  + <<UserDef>> , kaže napiši onu proceduru, onu logiku koja tebi kao projektantu odgovara za datu situaciju. Posebno su isprojektovane i isprogramirane i specificirane unaped određenom sintaksom.

**Specifikacija tipa ograničenja u RMP**



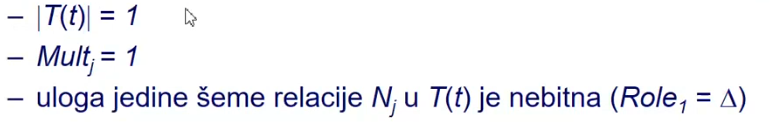
U okviru oblasti definisanosti mi govorimo koja je to širina te logičke strukture obeležja koja je povezana sa definiciojom naših elemenata u tipu logičke strukture obeležja, pa govorimo o vanrelacionim, jednorelacionim i viserelacionim tipovima ograničenja.

**Oblasti definisanosti tipa ograničenja u RMP**

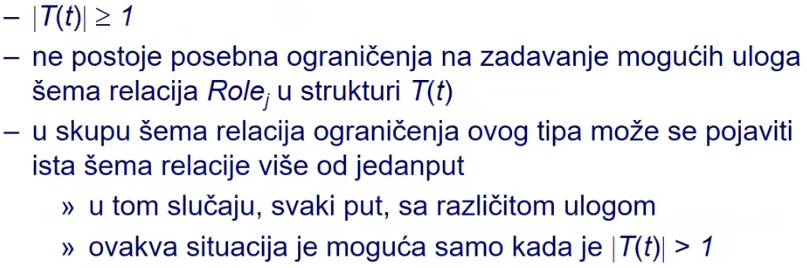
* vanrealcioni (bezrelacioni) tip ograničenja, ovo ograničenje se ne definiše nad šemama realcije šeme BP, definiše se nad predefinisanim ili korisnički definisanima domenima (tipovima)



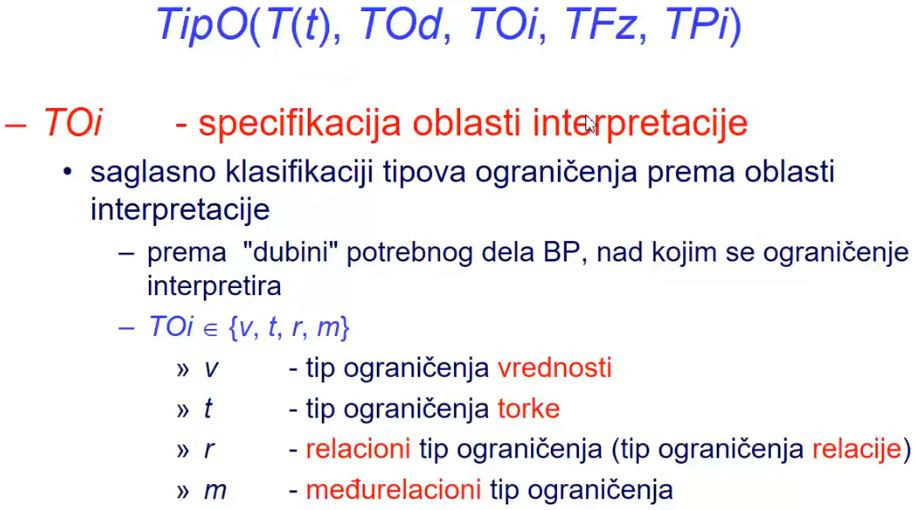
* jednorelacioni (unutarrelacioni, lokalni) tip ograničenja, ograničenje se definiše nad tačno jednom šemom relacije



* višerelacioni tip ograničenja, ograničenje se definiše nad skupom šema relacija, koji sadrži bar dva člana



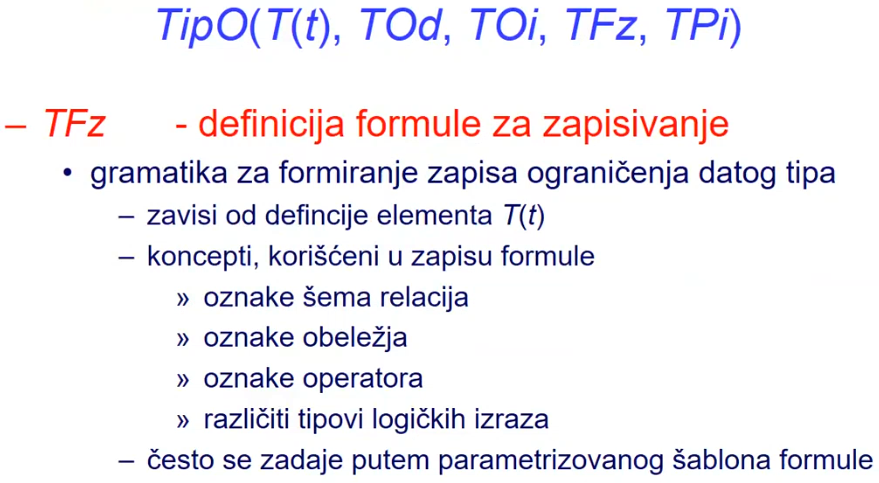
**Specifikacija tipa ograničenja u RMP**



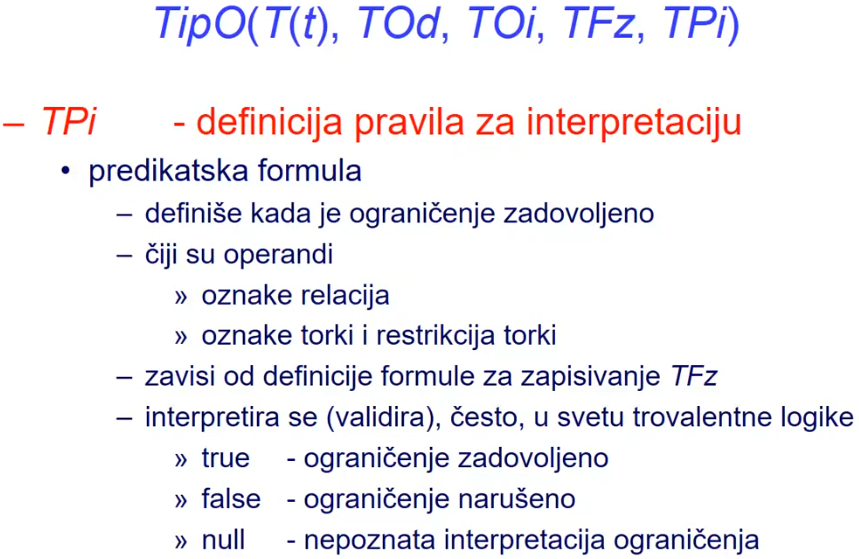
**Oblasti interpretacije tipa ograničenja u RMP**

* tip ograničenja vrednosti, interpretira se nad tačno jedno vrednošću nekog budućeg obeležja.
* tip ograničenja torke, intrepetira se nad jednom torkom bilo koje relacije.
* relacioni tip ograničenja, intrepetira se nad skupom torki bilo koje relacije.
* međurelacioni tip ograničenja, intrepetira se nad barem dve, bilo koje relacije, koje ne moraju biti međusobno različite. Napomena bilo koja relacija znači relacija iz baze podataka ili neka relacija koja predstavlja definisani pogled. (Pogled je relacija koja je nastala primenom izrazne relacione algebre nad jednom ili više drugih relacija, ili primenom operatora spajanja)

**Specifikacija tipa ograničenja u RMP**



**Specifikacija tipa ograničenja u RMP**



## Tipovi ograničenja u RMP

**Ograničenje domena**



O ovom ograničenju bi trebalo sve već da znamo iz baza podataka 1 sada radimo samo rekapitulaciju.



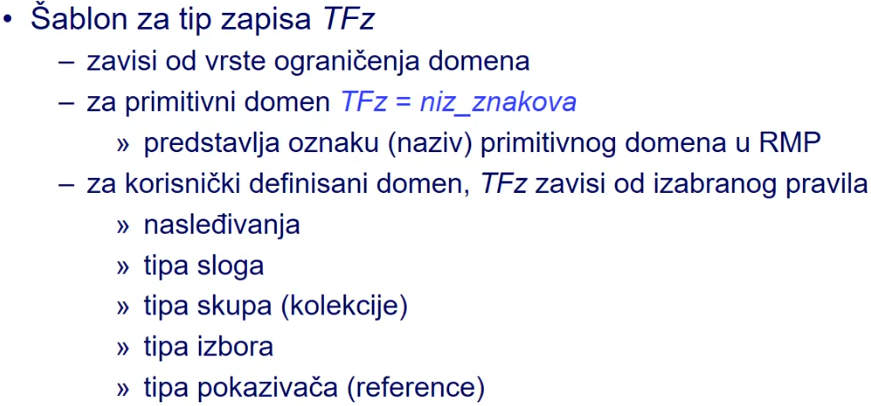
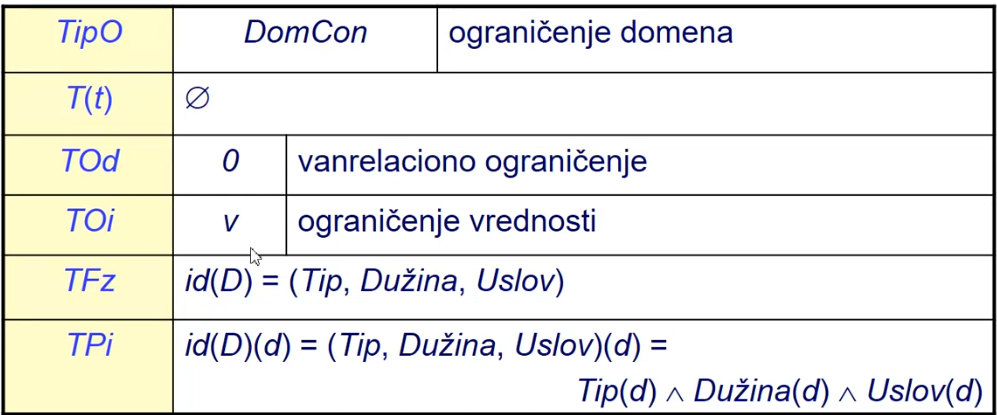
Pre svega ograničenje domena je bez relaciono ogranilčenje (precrtana nula u zagradi kod DomCon) sto znači nema nikakve posebne tipove logičke strukture obeležja nad kojim se definiše.

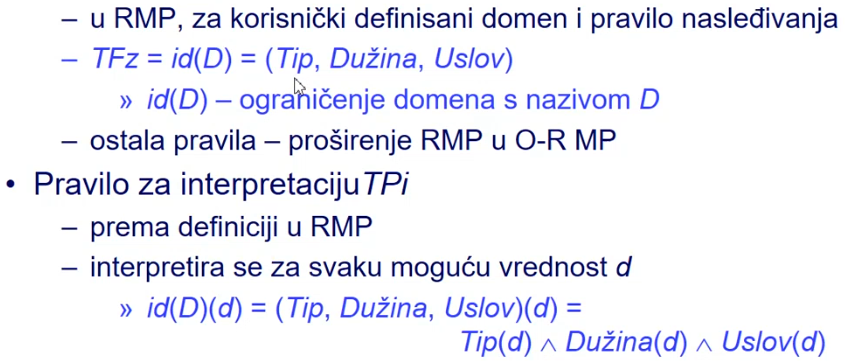


Ne definiše se ni na jednom jedinom šemom relacije ( nula u zagradi kod DomCon)



Definiše se kao ograničenje vrednosti, za njegovu intrepertaciju trebaće nam bilo koja vrednost.

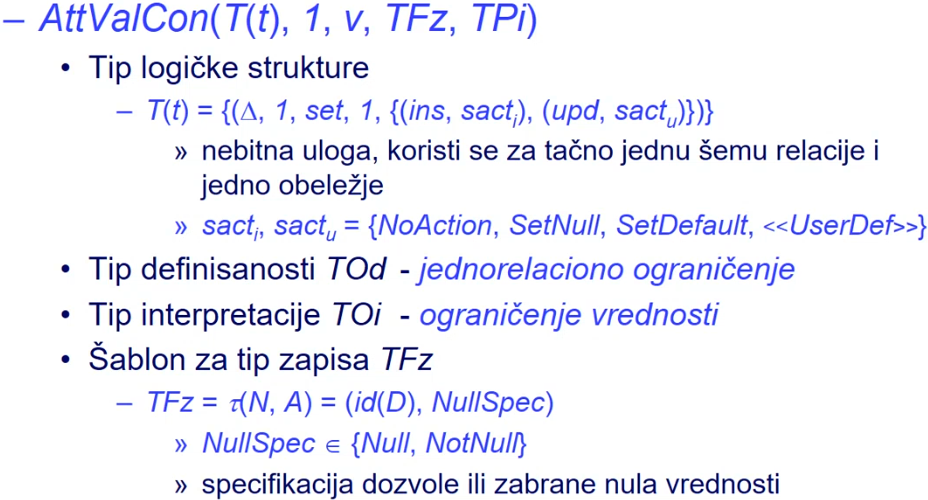
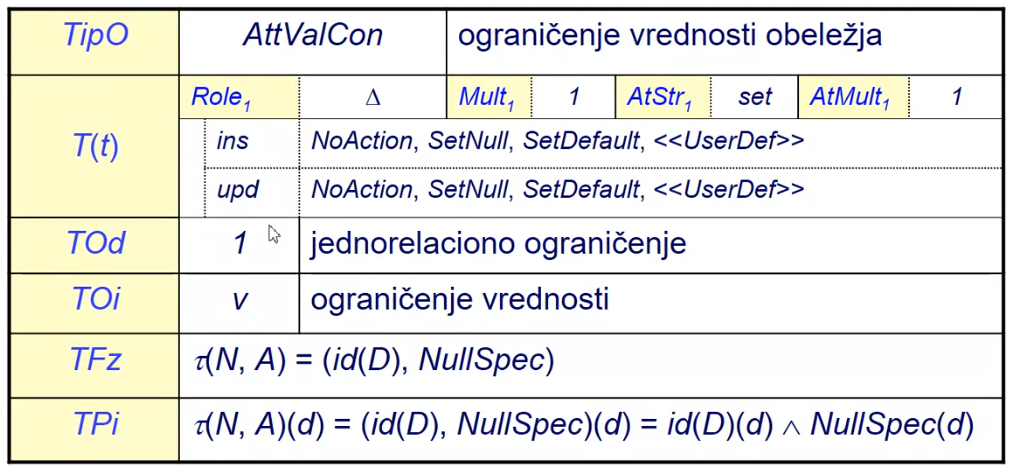
 



Vrednosti tip, dužina I uslov primenjujemo na bilo koje vrednosti, primenjujemo ih prvo kao proveru tipa pa proveru dužine pa proveru uslova. Ovo “and” nije ono and koje mi potnajemo nego “cond” (conditional and), “conditional and” ne zahteva da svaki operand u svakom trenutku bude interpretabilan, redosled pisanja je važan, npr ako leva strana vrati false desna strana ne mora uopšte da bude intrepertabilna.

**Ograničenje vrednosti obeležja**

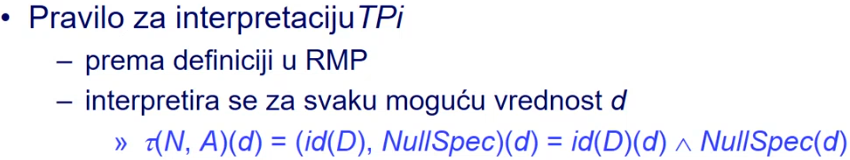
Prvi put uvodimo sada tip logičke strukture obeležja koji neće biti prazan skup, ograničenje vrednosti obeležja definišemo kao jednorelaciono ograničenje to znači da će biti definisano nad jednom šemom relacija u krajnjoj liniji i da će imati svoju formulu za zapisivanje i pravilo za interpretaciju.

Tip logičke strukture obeležja se sastoji iz jedne jedine komponente, znači definišemo jednu jedinu ulogu I kažemo da je naziv te uloge nebitan (znak delta u zagradi kod T(t)), dalje kažemo da će se u instanci pod tom nebitnom ulogom pojavljivati samo jedna šema relacije (broj jedan u zagradi kod T(t), I da će se pojavljivati skup od samo jednog atributa (set u zagradi kod T(t), I da su kritične operacije ins I upd (insert I update), za kritičnu operaciju insert I update definišemo skup mogućih akcija (NoAction, SetNull, SetDefault, <<UserDef>>) tj definišemo šta se može primeniti u slučaju da neko naruši ograničenje.

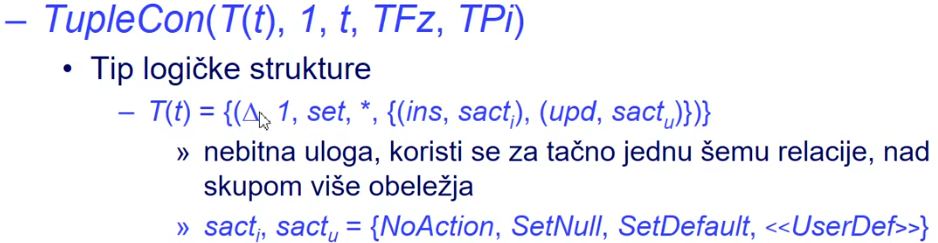
Šablon zapisa formule je onaj koji već znamo iz BP1, ograničenje vrednosti obeležja ovde označeno kao 

Definisemo kao par čija prva komponenta je ograničenje domena a druga komponenta je specifikacija ograničenja nula vrednosti 



**Ograničenje torke**

Ograničenje torke podrazumeva da moramo da imamo ograničenje svakog obeležja u šemi relacije i još dodatno ako postoji zadajemo logički uslov koji međusobno vrednosti obeležja u bilo kojoj torci moraju da zadovoljavaju.



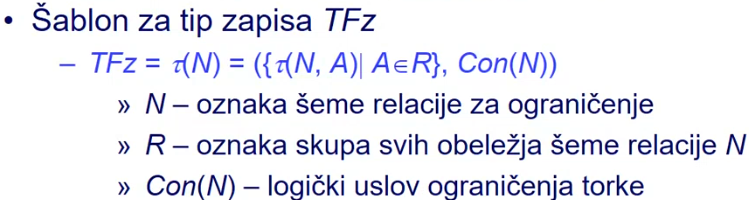
Tip logičke strukture obeležja opet kaže da i dalje radimo da radimo sa jednom jedinom ulogom šeme relacije čiji je naziv nebitan, radimo i dalje sa jednom jedinom šemom relacije čiji je naziv nebitan, set \* znači da radimo nad skupom od više atributa, i ovo ograničenje ima svoje kritične operacije (ins, update) i skup akcija.



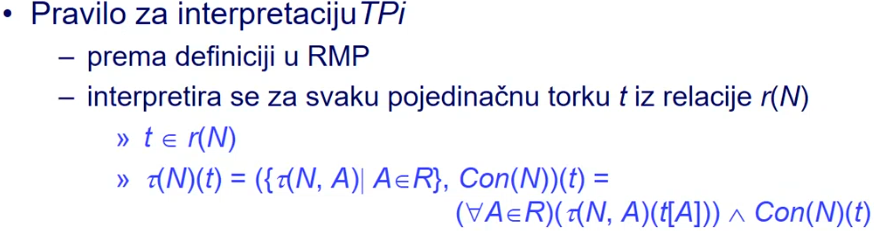
Ovo je i dalje jednorelaciono ograničenje jer se definiše nad jednom jedinom šemom relacije sa nebitnom ulogom.



Ograničenje torke zato što koristimo sada više obeležja, i koristimo skup obeležja jedne šeme relacije, što znači da njegova interpretacija ne može da ide nad jednom jedinom vrednošću, nego moramo za njegovu interpretaciju da podmetneš celu torku da bi graničenje bilo validirano



Ograničenje torke definišemo sa nekim tau od en, i onda kažemo da je prva komponenta ovog ograničenja je skup svih ograničenja vrednosti obeležja za sva obeležja iz šeme relacije koja je navedena, iz njenog skupa obeležja R, a druga komponenta Con(N) je logiči uslov ograničenja torke (logički constraint).



Ograničenje se iterpretira za bilo koju torku iz relacije nad datom šemom relacije, da bi smo ga interpretirali potreban nam je sadržaj cele moje torke ( t element r(N) ), primenjujemo specifikaciju ograničenja nad jednom instanciranom torkom iz relacije ( tau(N)(t) ), i primenjujemo tu torku nad celom specifikacijom , tako što za svaki atribut iz skupa atributa moje šeme relacije primenim nad ograničenjem vrednosti obeležja nad onom vrednošću torke koja odgovara tom mom obeležju, onda opet imamo conditional and koji kaže ukoliko su ti ograničenja vrdnosti obeležja zadovoljena za sva obeležja tvoje šeme relacije ond aproveri i da li je logički uslov ograničenja torke definisan u ovom ograničenju takođe validan za ovu torku t.

Sve ovo od gore prikazano u tabelarnom obliku:



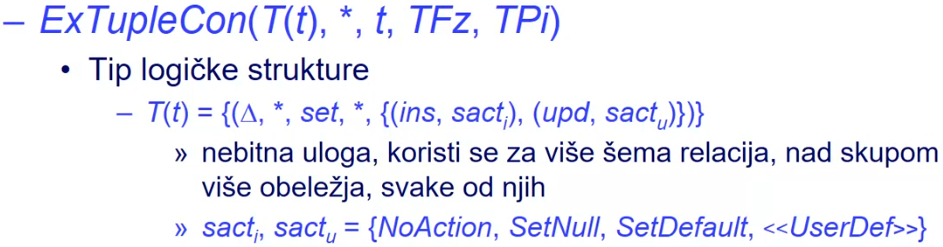
Pitanje: Kako intrepetiramo ograničenje torke i kada ga vršimo ? Potrebna su nam ograničenja vrednosti svakog obeležja u toj torci, potrebna su nam ograničenja vrednosti dva ili više obeležja unutar iste torke a da bi smo evaluirali potrebna nam je vrednost cele torke da bi na njoj to izvršili.

Pitanje: Kritične operacije su insert I update, zašto delete nije ? Zato što je samo unutar jedne torke ograničenje, ako obrišemo nećemo narušiti ništa unutar neke druge torke, tj zato što pravilo interpretacije ne dotiče ni jednu drugu torku osim te torke koje se tiče to ograničenje.

Pitanje: Zašto nema Cascade u skupu kritičnih operacija? Zato što je jednorelaciono ograničenje I zato što pravilo interpretacije ne dotiče ni jednu drugu torku osim te torke koje se tiče to ograničenje.

**Prošireno ograničenje torke**

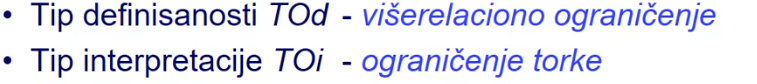
Ima dosta dodirnih tačaka sa običnim ograničenjem torke, samo sada možemo taj logički uslov da protegnemo na dve ili više šema relacija.



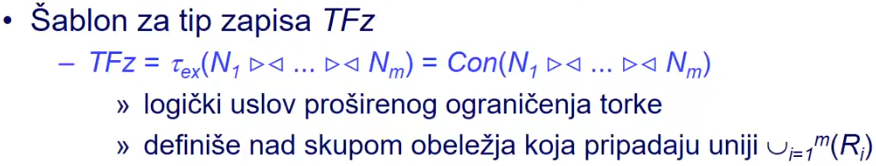
Primer o kom priča profesor na predavanju vezano za banku, ekpoziture i blagajne, poenta priče i pitanje je koje ograničenje je potrebno uvesti da bi zabranili blagajnici da isplati veću sumu novca nego što je stanje blagajne ? Kako ćemo iskazati to ograničenje ?

To ograničenje je između dva obeležja između različitih šema relacije, jedno je nejno stanje blagajne na dan i smenu koju je otvorila na početku, a drugo je nalog za transakciju šeme relacije u kojoj postoji iznos transakcije. Iznos transakcije iz šeme relacije transakcija povezujemo sa stanje blagajne šeme relacije stanje blagajne na dan. I kažemo da stanje blagajne mora da bude vece ili jednako od bilo koje izlaze transakcije isplate novce. Kojim ograničenjem ovo može da se iskaže? Da li može običnim ograničenjem torke? Pa ne može zato što obuhvata dve šeme relacija. Potreban nam je neki vid spajanja sa torkom stanja blagajne.

Tip logičke strukture obeležja takođe obuhvata jednu ulogu ( delta u T(t) ) ali se u toj jednoj ulozi može pojaviti više šema relacija ( \* u T(t) ), i nad tih više šema relacija na dalje se može pojaviti skup obeležja iz više šema relacija ( set, \* u T(t) ), kritične operacije su update i delete. Skupovi mogućih akcija su definisani iznad.



Ovo je sada višerelaciono ograničenje ali je i dalje ograničenje torke jer se validira na jednom jedinom torkom.

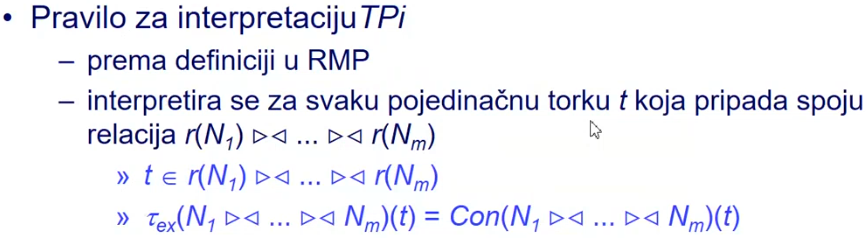


Tau ex ograničenje torke nad šemama relacija koje dovodimo u neku vezu zbog prirodog spajanja



I definišemo logički uslov koji definiše kako ćemo proveravati vrednosti naših obeležja nad nekom torkom

. 

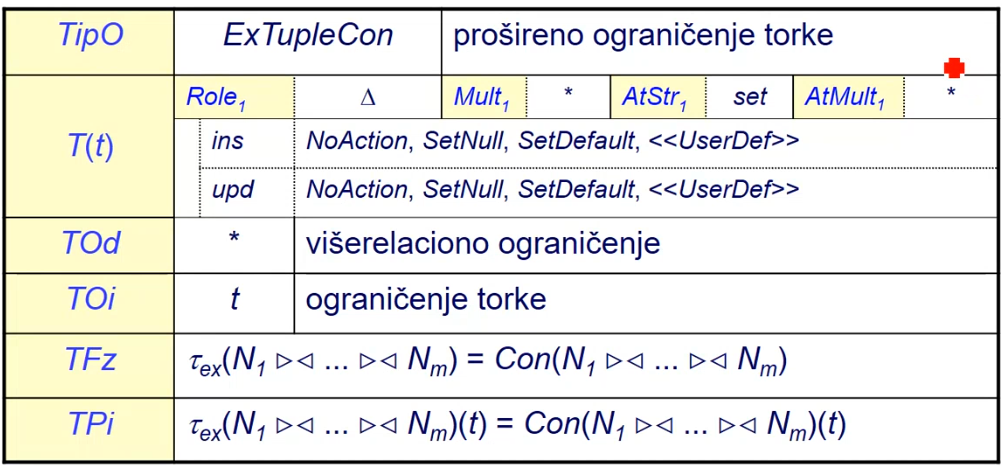


Pravilo interpretacije kaže sledeće, uzmi nad svim prethodno iskorišćenim šemama relacija relacije, spoj te relacije, uzmi bilo koju torku iz te relacije, za bilo koju torku te spojene relacije koja je nastala prirodnim spajanje tih relacija i intrepetiraj to ograničenje nad tom torkom tako što ćeš taj logički uslov intrepetirati tačno nad tom torkom. Šta to znači ? To znači da mi imamo sadržaje šema relacija, napravimo spoj, i svaka torka iz tog prirodnog spoja mora da zadovoljava ovaj definisan uslov.



Pošto ovo ograničenje kontrolišemo u procesu ažuriranja podataka kritične operacije su insert i update ali u bilo koju od ovih relacija.

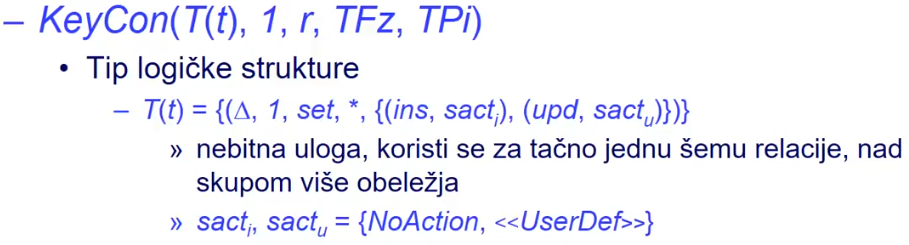
Sve ovo od gore prikazano u tabelarnom obliku:



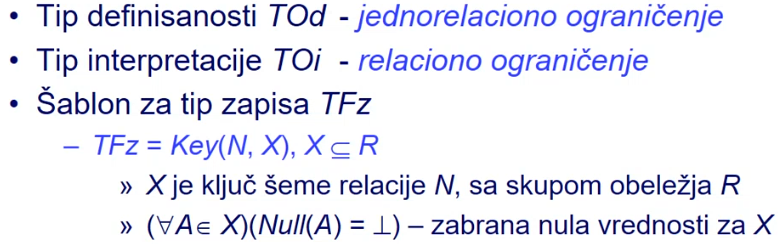
Predavanje 2.

**Ograničenje ključa**

Ovo ograničenje je veoma poznato iz baza 1 i 2, ovo bi zapravo trebalo da bude rekapitulacija.



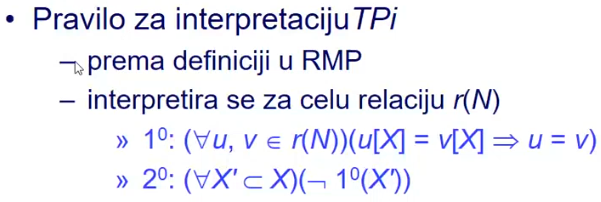
Tip logičke strukture obeležja uključuje jednu jedinu ulogu (delta u zagradi kod T(t)), ta uloga je nebitna, dozvoljava jednu šemu relacija sa navedenom ulogom i koristimo skup obeležja te šeme relacija da bi smo iskazali šta je to ograničenje ključa. Kritične operacije su insert i update. Akcije koje se preduzimaju prilikom kršenja su NoAction i neke korisničke definisane akcije.



Jednorelaciono je ograničenje zato što se definiše nad jednom šemom relacije, takođe je i relaciono ograničenje zato što se proverava nad celo šemom relacije.

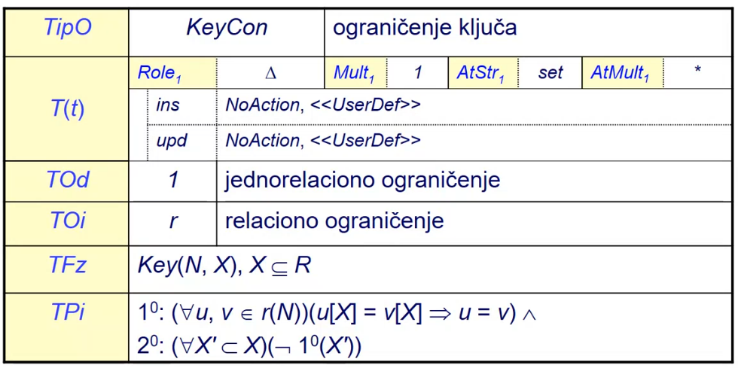
U formuli za zapisivanje zaključujemo samo da je X ključ šeme relacije N. Takva formulacija sa sobom obavezno donosi da su nula vrednosti za X zabranjene.

Pitanje: Zašto delete nije kritična operacija za ograničenje ključa ? Zato što je ključ unique vrednost, kad brišemo znamo da ne možemo ništa da narušimo, kad radimo insert i update možemo narušiti ovo ograničenje. Zanimljivost ovde je što se minimalnsot uopšte ne kontroliše, prepuštamo to na verovanje da je projektant odradio posao kako treba. Tu zaključujemo da se deo problema rešava na projektantskom nivou a deo se prepušta sistemu da rešava zadavanjem ograničenja.



1. Pravilo za interpetaciju kaže, za svake dve torke ako su one jednake na X-u onda su one jednake generalno. Odnosno ako idemo obrat po kontrapoziciji ove inplikacije rećemo za svake dve torke iz relacije mora da važi ako su svake dve torke različite makar moraju biti različite na X-u, na ovaj način i SBP proverava.
2. Drugo svojstvo postoji kao projektantsko svojstvo, ono kaže da svaki pravi podskup od X ne sme da ostane u bilo kojoj relaciji zadovoljeno svojstvo 1, tj biće narušeno svojstvo 1. To znači da će uvek psotojati neka relacija koja će narušiti svojstvo 1, ne mora značiti da će svaka narušiti.

Sve ovo od gore prikazano u tabelarnom obliku:



Pitanje: Zašto su SetDefault i SetNull ovde otpali? Null logično zbog ključa otpada, a Default zbog Uniquea, zbog prvog uslova jedinstvenosti torke, Default je ponovljena vrednost i zato to ovde nema smisla.