Predavanje 5

Sadrzaj

Realizacija ogranicenja pomocu RSUBP

Ogranicenje domena

TipO	*DomCon*	ograničenje domena
T(t)	∅	
TOd	*0*	vanrelaciono ograničenje
TOi	*v*	ograničenje vrednosti
TFz	*id(D) = (Tip, Dužina, Uslov)*	
TPi	*id(D)(d) = (Tip, Dužina, Uslov)(d) = Tip(d) ∧ Dužina(d) ∧ Uslov(d)*	

Na slici vidimo tabelarni prikaz specifikacije tipa ogranicenja domena. Moguci nacini implementirana su: **create domain, create/alter table, constraint check** i **create trigger**.

- Primer
  - domen *DPOL*(*id*(*DPOL*), '*Ž*')

OgrNaz	*DPOL*	
OgrTip	*DomCon*	Ograničenje domena
OgrF	*id*(*DPOL*) = (*String*, *1*, *d* ∈ {*M*, *Ž*})	
T(*o*)	∅	

  - *DPOL* je namenjen za pridruživanje obeležju *POL*
    - Šema relacije
      - » *Radnik*({*MBR*, *PRZ*, *IME*, *POL*, *DATR*, *JMBG*}, *C*)
    - Ograničenje vrednosti obeležja
      - » τ(*Radnik*, *POL*) = (*DPOL*, ⊥)

Realizacija ogranicenja

- CREATE DOMAIN

```
CREATE DOMAIN DPOL CHAR(1)
   DEFAULT 'Ž'
   CONSTRAINT con_dpol CHECK (Value IN ('M', 'Ž'))


CREATE DOMAIN DPOL CHAR(1)
   DEFAULT 'Ž'
   CONSTRAINT con_dpol CHECK (Value  = 'M' OR Value = 'Ž')
```

- CREATE TABLE, CONSTRAINT CHECK

```
CREATE TABLE RADNIK
     (...,
      POL CHAR(1)
        CONSTRAINT con_dpol CHECK (POL IN ('M', 'Ž'))
        DEFAULT 'Ž',
      ...
     )
```

- ALTER TABLE, CONSTRAINT CHECK

```
ALTER TABLE RADNIK
    MODIFY (POL CHAR(1) DEFAULT 'Ž')
ALTER TABLE RADNIK
    ADD CONSTRAINT con_dpol
                   CHECK (POL IN ('M', 'Ž'))
```

Treba napomenuti da ako imamo klauzulu **modify** mi cemo redefinisati citavo to ogranicenje koje menjamo, pa ako je recimo pre toga (prilikom kreiranja npr.) postojalo **not null** ogranicenje a mi ovako kao u nasem primeru definisemo izmenu, mi smo ukinuli zapravo to prethodno ogranicenje, stoga, kada radimo modify, moramo navesti sve sto zelimo da od sad vazi (znaci nemamo patchovanje nego totalno novu redefiniciju !). Klauzule **disable | enable** samo omogucuju da na dalje to ogranicenje vazi ili ne vazi, ali ga ne brise (kao drop sto radi).

Ono sto smo vec rekli ali da spomenemo jos jednom, da bi ogranicenje koje sada uvodimo bilo izvrseno, citav sadrzaj onoga sto to ogranicenje obuhvata mora da zadovolji to samo ogranicenje (iz tog razloga je dobro definisati ogranicenja na pocetku).

Ogranicenje vrednosti obelezja

TipO	AttValCon		ograničenje vrednosti obeležja					
T(t)	$Role_1$	Δ	$Mult_1$	1	$AtStr_1$	set	$AtMult_1$	1
	ins	NoAction, SetNull, SetDefault, <<UserDef>>						
	upd	NoAction, SetNull, SetDefault, <<UserDef>>						
TOd	1	jednorelaciono ograničenje						
TOi	v	ograničenje vrednosti						
TFz	$\tau(N, A) = (id(D), NullSpec)$							
TPi	$\tau(N, A)(d) = (id(D), NullSpec)(d) = id(D)(d) \wedge NullSpec(d)$							

Slika predstavlja podsetnik na tip ogranicenja koji predstavlja ogranicenje vrednosti obelezja. Moguci nacini realizacije su **create/alter table, constraint not null, create trigger.**

- Primer
  - šema relacije
    - *Radnik*({*MBR*, *PRZ*, *IME*, *POL*, *DATR*, *JMBG*}, *C*)
  - ograničenje vrednosti obeležja $\tau(Radnik, POL)$

OgrNaz	AttValCon_POL_DPOL			
OgrTip	AttValCon	ograničenje vrednosti obeležja		
OgrF	$\tau(Radnik, POL) = (DPOL, \perp)$			
T(o)	N_1	Radnik	ρ_1	Δ
	ins	*	NoAction	
	upd	POL	NoAction	

Realizacija ogranicenja

- CREATE TABLE, CONSTRAINT CHECK

```
CREATE TABLE RADNIK
     (...,
      POL DPOL CONSTRAINT con_nullpol NOT NULL,
      ...
     )

CREATE TABLE RADNIK
     (...,
      POL DPOL NOT NULL,
      ...
     )
```

- ALTER TABLE, CONSTRAINT CHECK

```
ALTER TABLE RADNIK
     MODIFY (POL DPOL NOT NULL)

ALTER TABLE RADNIK
     MODIFY (POL CHAR(1)
                  NOT NULL
                  CHECK POL IN ('M', 'Ž')
                  DEFAULT 'Ž'
             )
```

- Primer 2
  - šema relacije
    - *Radnik*({*MBR*, *PRZ*, *IME*, *POL*, *DATR*, *JMBG*}, *C*)
  - ograničenje vrednosti obeležja τ(*Radnik*, *POL*)

OgrNaz	*AttValCon_POL_DPOL*			
OgrTip	*AttValCon*	ograničenje vrednosti obeležja		
OgrF	τ(*Radnik*, *POL*) = (*DPOL*, $\perp$)			
T(*o*)	N_1	*Radnik*	ρ_1	Δ
	ins	*	*SetDefault*	
	upd	*POL*	*SetDefault*	

Kao sto vidimo, isto je ogranicenje vrednosti obelezja (obelezju pol pridruzujemo domen DPOL i pri tome jos vrsimo zabranu dodele null vrednosti za obelezje POL). Medjutim, sada je akcija koja se desava kada dolazi do krsenja ogranicenja vise nije **no action** nego **set default**. Time zelimo da postignemo da imamo neku podrazumevanu vrednosti tog obelezja.

Realizacija ogranicenja

Akcija **SetDefault** nije podržana deklarativnim mehanizmima pa se koristi proceduralni mehanizam **CREATE TRIGGER**. Za obeležje POL se prvo ukida ograničenje **NOT NULL**, ako je bilo deklarisano.

```
ALTER TABLE RADNIK
    DROP CONSTRAINT con_nullpol

ALTER TABLE RADNIK MODIFY (POL DPOL DEFAULT 'Ž')

ALTER TABLE RADNIK MODIFY
  (POL CHAR(1) CHECK POL IN ('M', 'Ž') DEFAULT 'Ž'
  )

– može se ukinuti i CHECK i DEFAULT
    ALTER TABLE RADNIK MODIFY (POL CHAR(1))
```

CREATE TRIGGER

```
CREATE OR REPLACE TRIGGER Radnik_nullpol
    BEFORE INSERT OR UPDATE OF POL
    ON RADNIK
    FOR EACH ROW
    WHEN (NEW.POL IS NULL
                    OR NEW.POL NOT IN ('M', 'Ž'))
    BEGIN
      :NEW.POL := 'Ž';
    END Radnik_nullpol;
```

Kao sto vidimo, imamo par opcija (na slici iznad levo prikazanih). Klauzulu **default** mozemo upotrebiti ali ceo mehanizam tipa podatka i check constrainta podrazumeva aktivnost no action, to znaci da ako pokusamo preko inserta da ubacimo dva znaka za pol recimo ili znak 'F', 'L' za pol, mi cemo *dobiti gresku* i naredba ce biti oborena. Ali mi necemo da dobijemo gresku i da naredba bude oborena, te stoga koristimo **trigere**.

Kao sto vidimo, u gornjem trigeru imamo **row level** triger i znamo da ce se on pokrenuti onoliko puta koliko imamo upisa ili modifikacija u istoj naredbi.

Posto PL/SQL Engine pocinje od klauzule **begin** da bi pristupili van konteksta tog engina, moramo koristiti dvotacku, te stoga kazemo **:NEW** (ona je host promenljiva za pl/sql blok). Voditi racuna da to vazi samo za razlicite engine (ako imamo pl/sql blok u pl/sql bloku, nema potrebe za dvotackom !).

Da li ce ovaj triger ikada da obori izvodjenje insert ili update naredbe ? U normalnim prilikama ne, korisnik nikada nece dobiti obavestenje da je narusio ogranicenje domena (odnosno ogranicenje vrednosti obelezja za obelezje pol). Jedino sto moze da se dogodi, jeste da je korisnik upisivao null vrednosti (probao preskociti upis tog polja) ili da je probao da upise nesto deseto a na kraju mu se svakako pojavi u bazi podataka vrednost 'Z' za obelezje pol.

Ogranicenje torke

TipO	*TupleCon*	ograničenje torke
T(t)	$Role_1$ | Δ | $Mult_1$ | *1* | $AtStr_1$ | *set* | $AtMult_1$ | *	
	ins	*NoAction, SetNull, SetDefault, <<UserDef>>*
	upd	*NoAction, SetNull, SetDefault, <<UserDef>>*
TOd	*1*	jednorelaciono ograničenje
TOi	*t*	ograničenje torke
TFz	$\tau(N) = (\{\tau(N, A) \mid A \in R\}, Con(N))$	
TPi	$\tau(N)(t) = (\{\tau(N, A) \mid A \in R\}, Con(N))(t) = (\forall A \in R)(\tau(N, A)(t[A])) \wedge Con(N)(t)$	

Ono forsira da svakom obelezju moramo dodeliti ogranicenje vrednosti obelezja i jos dodatno mozemo imati logicke uslove koje zadajemo na vrednosti nasih obelezja unutar date torke. Moguci nacini realizacije su **create/alter table, constraint check, create trigger**.

- Primer
  - šema relacije
    - *Radnik*({*MBR*, *PRZ*, *IME*, *POL*, *DATR*, *JMBG*}, *C*)
  - ograničenje torke
    - $\tau(Radnik) = (\{\tau(Radnik, A) \mid A \in R\}, Con(Radnik))$
      - $\tau(Radnik, MBR) = (DMBR, \perp)$, $id(DMBR) = (Number, o, d \geq 1)$
      - $\tau(Radnik, PRZ) = (DPRZ, \perp)$, $id(DPRZ) = (String, 35, \Delta)$
      - $\tau(Radnik, IME) = (DIME, \perp)$, $id(DIME) = (String, 25, \Delta)$
      - $\tau(Radnik, POL) = (DPOL, \perp)$, $id(DPOL) = (String, 1, d \in \{M, Ž\})$
      - $\tau(Radnik, DATR) = (DATUM, \perp)$, $id(DATUM) = (Date, \Delta, \Delta)$
      - $\tau(Radnik, JMBG) = (DJMBG, T)$,
        - » $id(DJMBG) = (String, 13, Length(d) = 13 \wedge ProveraContrBr(d))$
        - » **Napomena:** *DJMBG* dozvoljava samo unos vrednosti dužine *13*, za koje funkcija *ProveraContrBr* vraća *TRUE*
    - $\tau(Radnik) = (\{\tau(Radnik, A) \mid A \in R\}, Con(Radnik))$
      - $Con(Radnik) =$ $Substr(JMBG, 1, 7) = To\_Char(DATR, 'DDMMYYY')$
        - » **Napomena:** Zahteva se da prvih *7* cifara vrednosti za *JMBG* odgovara datumu rođenja *DATR*, zadatom u naznačenom formatu *DDMMYYY*

Isto to samo tabelarno (cak su i svako ogranicenje vrednosti obelzja tabelarno ispod predstavljeni).

OgrNaz	TupleCon_Radnik	
OgrTip	TupleCon	ograničenje torke
OgrF	$\tau(Radnik) = (\{\tau(Radnik, A) \mid A \in R\}, Con(Radnik))$, $Con(Radnik)$: $Substr(JMBG, 1, 7) = To\_Char(DATR, 'DDMMYYY')$	
T(o)	N_1	Radnik / ρ_1 / Δ
	ins	* / NoAction
	upd	{DATR, JMBG} / NoAction

OgrNaz	AttValCon_DATR_DATUM			
OgrTip	AttValCon	ograničenje vrednosti obeležja		
OgrF	$\tau(Radnik, DATR) = (DATUM, \perp)$			
T(o)	N_1	Radnik	ρ_1	Δ
	ins	*	NoAction	
	upd	DATR	NoAction	

OgrNaz	DATUM	
OgrTip	DomCon	Ograničenje domena
OgrF	$id(DATUM) = (Date, \Lambda, \Lambda)$	
T(o)	$\varnothing$	

OgrNaz	AttValCon_JMBG_DJMBG			
OgrTip	AttValCon	ograničenje vrednosti obeležja		
OgrF	$\tau(Radnik, JMBG) = (DJMBG, T)$			
T(o)	N_1	Radnik	ρ_1	Δ
	ins	*	NoAction	
	upd	JMBG	NoAction	

OgrNaz	DJMBG	
OgrTip	DomCon	Ograničenje domena
OgrF	$id(DJMBG) = (String, 13, Length(d) = 13 \wedge ProveraContrBr(d))$	
T(o)	$\varnothing$	

OgrNaz	AttValCon_JMBG_DJMBG			
OgrTip	AttValCon	ograničenje vrednosti obeležja		
OgrF	$\tau(Radnik, JMBG) = (DJMBG, T)$			
T(o)	N_1	Radnik	ρ_1	Δ
	ins	*	NoAction	
	upd	JMBG	NoAction	

OgrNaz	DJMBG	
OgrTip	DomCon	Ograničenje domena
OgrF	$id(DJMBG) = (String, 13, Length(d) = 13 \wedge ProveraContrBr(d))$	
T(o)	$\varnothing$	

OgrNaz	AttValCon_MBR_DMBR			
OgrTip	AttValCon	ograničenje vrednosti obeležja		
OgrF	$\tau(Radnik, MBR) = (DMBR, \perp)$			
T(o)	N_1	Radnik	ρ_1	Δ
	ins	*	NoAction	
	upd	MBR	NoAction	

OgrNaz	DMBR	
OgrTip	DomCon	Ograničenje domena
OgrF	$id(DMBR) = (Number, 6, d \geq 1)$	
T(o)	$\varnothing$	

OgrNaz	AttValCon_IME_DIME			
OgrTip	AttValCon	ograničenje vrednosti obeležja		
OgrF	$\tau(Radnik, IME) = (DIME, \perp)$			
T(o)	N_1	Radnik	ρ_1	Δ
	ins	*	NoAction	
	upd	IME	NoAction	

OgrNaz	DIME	
OgrTip	DomCon	Ograničenje domena
OgrF	$id(DIME) = (String, 25, \Lambda)$	
T(o)	$\varnothing$	

OgrNaz	AttValCon_POL_DPOL			
OgrTip	AttValCon	ograničenje vrednosti obeležja		
OgrF	$\tau(Radnik, POL) = (DPOL, \perp)$			
T(o)	N_1	Radnik	ρ_1	Δ
	ins	*	NoAction	
	upd	POL	NoAction	

OgrNaz	DPOL	
OgrTip	DomCon	Ograničenje domena
OgrF	$id(DPOL) = (String, 1, d \in \{M, Ž\})$	
T(o)	$\varnothing$	

Da bi imali precizno specificiranu ovu specifikaciju dalje moramo da razradimo odnosno da definisemo logicku funkciju (ProveraContrBr(d)).

Realizacija ogranicenja

- CREATE DOMAIN - za ograničenja domena

```
CREATE DOMAIN DMBR NUMBER(6)
  CONSTRAINT con_dmbr CHECK (Value ≥ 1);

CREATE DOMAIN DPRZ VARCHAR(35);

CREATE DOMAIN DIME VARCHAR(25);

CREATE DOMAIN DPOL CHAR(1) DEFAULT 'Ž'
  CONSTRAINT con_dpol CHECK (Value IN ('M', 'Ž'));

CREATE DOMAIN DATUM DATE;

CREATE DOMAIN DJMBG VARCHAR(13)
  CONSTRAINT con_djmbg CHECK (
        Length(Value) = 13 AND ProveraContrBr(Value)
                                    );
```

- CREATE FUNCTION
  - za realizaciju korisnički definisanih funkcija na serveru BP

```
CREATE OR REPLACE FUNCTION
                        ProveraContrBr (Jmbg IN VARCHAR)
RETURN BOOLEAN IS
    ...
BEGIN
    ...
END;
```

Stringovi se iz spoljneg konteksta(kao i u C) prenose u funkciju/proceduru iskljucivo po *referenci* te stoga ne navodimo duzinu (samo kazemo varchar) i posto je varchar podrazumeva se da je u pitanju **no copy** klauzula parametra.

- CREATE TABLE, CONSTRAINT CHECK

```
CREATE TABLE RADNIK
    ( MBR DMBR NOT NULL,
      PRZ DPRZ NOT NULL,
      IME DIME NOT NULL,
      POL DPOL NOT NULL,
      DATR DATUM NOT NULL,
      JMBG DJMBG,
      CONSTRAINT TupleCon_Radnik CHECK (
            Substr(JMBG, 1, 7) = To_Char(DATR, 'DDMMYYY')
                                    )
    )
```

Na kraju imamo proveru jos da je substring jmbg-a jednak datumu.

- Primer 2
  - šema relacije
    - *Radnik({MBR, PRZ, IME, POL, DATR, JMBG}, C)*
  - ograničenje torke $\tau(Radnik)$

OgrNaz	*TupleCon_Radnik*			
OgrTip	*TupleCon*	ograničenje torke		
OgrF	$\tau(Radnik) = (\{\tau(Radnik, A) \mid A \in R\}, Con(Radnik))$, *Con(Radnik)*: *Substr(JMBG, 1, 7) = To_Char(DATR, 'DDMMYYY')*			
T(o)	N_1	*Radnik*	ρ_1	Δ
	ins	{*JMBG*}	*SetNull*	
	upd	{*JMBG*}	*SetNull*	

Zelimo u ovom primeru sada da ne prekidamo naredbu i obavestavamo korisnika ako pogresi prilikom unosa/modifikacije, nego da postavimo null vrednost.

- akcija *SetNull*
  - specificirana je samo za obeležje *JMBG*
    - » τ(*Radnik*, *JMBG*) = (*DJMBG*, T)
    - » τ(*Radnik*, *DATR*) = (*DATUM*, $\perp$)
  - nije podržana deklarativnim mehanizmima
  - CONSTRAINT Radnik_TupleCon CHECK se izostavlja

```
CREATE TABLE RADNIK
        ( MBR DMBR NOT NULL,
          PRZ DPRZ NOT NULL,
          IME DIME NOT NULL,
          POL DPOL NOT NULL,
          DATR DATUM NOT NULL,
          JMBG DJMBG
        )
```

- korišćenje proceduralnih mehanizama
- CREATE TRIGGER

- CREATE TRIGGER

```
CREATE OR REPLACE TRIGGER TupleCon_Radnik
    BEFORE INSERT OR UPDATE OF DATR, JMBG
    ON RADNIK
    FOR EACH ROW
    WHEN (Substr(NEW.JMBG, 1, 7) !=
            To_Char(NEW.DATR, 'DDMMYYY'))
    BEGIN
      :NEW.JMBG := NULL;
    END TupleCon_Radnik;
```

Prosireno ogranicenje torke

TipO	*ExTupleCon*		proširenо ograničenje torke					
T(t)	$Role_1$	Δ	$Mult_1$	*	$AtStr_1$	set	$AtMult_1$	*
	ins	*NoAction*, *SetNull*, *SetDefault*, *<<UserDef>>*						
	upd	*NoAction*, *SetNull*, *SetDefault*, *<<UserDef>>*						
TOd	*	višerelaciono ograničenje						
TOi	*t*	ograničenje torke						
TFz	$\tau_{ex}(N_1 \triangleright\triangleleft \dots \triangleright\triangleleft N_m) = Con(N_1 \triangleright\triangleleft \dots \triangleright\triangleleft N_m)$							
TPi	$\tau_{ex}(N_1 \triangleright\triangleleft \dots \triangleright\triangleleft N_m)(t) = Con(N_1 \triangleright\triangleleft \dots \triangleright\triangleleft N_m)(t)$							

Karakteristika je da se zadaje logicki uslov koji obuhvata vise sema relacije I prakticno da bi smo dosli do kontrole ogranicenja konkretne torke, mi moramo pospajamo neke torke iz nekih relacija. Moguci nacini realizacije su **create assertion, create/alter table, constraint check** i **create trigger**. Od svih njih vecina DBMS-ova jedino moze da podrzi *triggere*.

- Primer
  - šeme relacija
    - *Građanin*({*JMBG*, *PRZ*, *IME*, *POL*, *DATR*}, C_1)
    - *Dokument*({*TIP*, *SBROJ*, *DATIZ*, *STAT*, *JMBG*}, C_2)
  - proširenо ograničenje torke
    - τ_{ex}(*Građanin* $\triangleright\triangleleft$ *Dokument*) = *Con*(*Građanin* $\triangleright\triangleleft$ *Dokument*)
      - *DATIZ* $\geq$ *DATR*
  - ograničenje referencijalnog integriteta
    - *Dokument*[*JMBG*]$\subseteq$ *Građanin*[*JMBG*], *Key*(*Građanin*,{*JMBG*})
      - u trenutku upisa nove torke u *r*(*Građanin*) ne postoji odgovarajuća torka u *r*(*Dokument*)
      - upis nove torke u *r*(*Građanin*) ne može narušiti uslov *Con*(*Građanin* $\triangleright\triangleleft$ *Dokument*)
      - za šemu relacije *Građanin*, (*ins*, At_j, act^{ij}) se ne specificira

OgrNaz	*ExTupleCon_GradDok*			
OgrTip	*ExTupleCon*	proširenо ograničenje torke		
OgrF	τ_{ex}(*Građanin* $\triangleright\triangleleft$ *Dokument*) = *Con*(*Građanin* $\triangleright\triangleleft$ *Dokument*): *DATIZ* $\geq$ *DATR*			
T(o)	N_1	*Građanin*	ρ_1	Δ
	upd	{*DATR*}	*NoAction*	
	ins	Δ	Δ	
	N_2	*Dokument*	ρ_2	Δ
	ins	*	*NoAction*	
	upd	{*DATIZ*, *JMBG*}	*NoAction*	

Pored standardnog iznad se nalazi I tabelarni prikaz. Kao sto vidimo, u Gradjaninu imamo insert ali njega ne specificiramo posebno jer on nije kritican zbog ogranicenja referencijalnog integriteta (jer znamo da ne postoji u Dokumentu sigurno u tom trenutku, takva torka da krsi ovo ogranicenje).

Realizacija ogranicenja

Kada bi **assertion** bio implementabilan u nekom DBMS-u on bi izgledao ovako (slika desno).

Posto nas ne interesuju podaci, nego samo logicki izraz (da li jeste ili nije). Problem je sto ovo nije implementabilno kod DBMS-ova. Ali mozemo probati preko **constraint** klauzule.

- CREATE ASSERTION

```
CREATE ASSERTION ExTupleCon_GradDok
  CHECK ( NOT EXISTS
                ( SELECT 0
                  FROM Građanin g, Dokument d
                  WHERE g.JMBG = d.JMBG
                      AND d.DATIZ < g.DATR
                )
        )
```

- ALTER TABLE, CONSTRAINT CHECK

```
ALTER TABLE Građanin
  ADD CONSTRAINT ExTupleCon_GradDokG
  CHECK ( DATR <=ALL (      SELECT d.DATIZ
                            FROM Dokument d
                            WHERE d.JMBG = JMBG
                          )
        );
ALTER TABLE Dokument
  ADD CONSTRAINT ExTupleCon_GradDokG
  CHECK ( DATIZ >= ( SELECT g.DATR
                     FROM Građanin g
                     WHERE g.JMBG = JMBG
                   )
        );
```

Prvi check kaze da je datum rodjenja gradjanina manji ili jednak od svih selektovanih datuma izdavanja dokumenata tog gradjanina.

A drugi check definisan nad tabelom Dokument kaze da je datum izdavanja dokumenta veci ili jednak od datuma rodjenja gradjanina.

U gornjem ogranicenju imamo all a dole ne zato sto dole znamo da za svaki dokument imamo samo jednog gradjanina.

Ali zbog toga sto imamo select naredbu unutar check constrainta ni ovo nece proci kod DBMS-ova. Takodje pravi problem to sto su tabele Gradjanin i Dokument vec uvezane ogranicenjem stranog kljuca (pa bi to bila dupla tj. ciklicna veza).

Ono sto nam ostaje je pravljenje trigera.

Triger je na gradjaninu I testiramo pokusaj da neko pomera datum rodjenja na mladji.

Kao sto vidimo select nam govori o kom se gradjaninu radi I proverava da li je novi datum rodjenja manji od datuma izdavanja dokumenta, ako imamo neke torke koje krse ovo ogranicenje, okidamo izuzetak.

Voditi racuna da se select *ne moze* pojaviti izmedju begin i end bez sluzbene reci **into**. Moze se posredno koristiti u kursorskim podrucjima (u radu sa kursorom), ali ne ovako direktno izmedju begina i enda. Ocekivanje naseg selekta kada se nalazi unutar

- CREATE TRIGGER

```
CREATE OR REPLACE TRIGGER ExTupleCon_GradDokG
    BEFORE UPDATE OF DATR
    ON Građanin FOR EACH ROW ...

CREATE OR REPLACE TRIGGER ExTupleCon_GradDokD
    BEFORE INSERT OR UPDATE OF DATIZ, JMBG
CREATE OR REPLACE TRIGGER ExTupleCon_GradDokG
  BEFORE UPDATE OF DATR
  ON Građanin
  FOR EACH ROW
  WHEN (NEW.DATR > OLD.DATR)
  DECLARE
     l_BrTorki NUMBER := 0;
  BEGIN
     SELECT Count(*)
     INTO l_BrTorki
     FROM Dokument d
     WHERE d.JMBG = :OLD.JMBG AND d.DATIZ < :NEW.DATR;
     IF l_BrTorki != 0 THEN
        Raise_Application_Error(-20999, '<Poruka>');
     END IF;
  END ExTupleCon_GradDokG;
```

begina i enda jeste da vrati jednu torku, a posto u nasem slucaju imamo agregacionu f-ju, znamo da ce to i sigurno uraditi. Ukoliko select vrati vise od jedne torke, izaziva se *izuzetak 'to_many_rows'*. Ukoliko select ne vrati pak nista, izuzetak je '*no_data_found*'. Stoga, dobro bi bilo da njih i obradimo u exception hendleru.

Posto vidimo da je ovo **row level** triger, znamo da je ta update naredba mogla da vec izmeni 20 torki i ako smo bacili gresku mi cemo imati *implicitni roll back* cele update naredbe (odnosno ponisticemo izmene nad svih tih 20 torki). Voditi racuna da se *cela sekvenca* obara (svih 6 stavi i svaka izmena u svakoj od stavki !). Tj. da smo imali sve okej i da smo u 6-om koraku sekvence (after update) rekli *raise_application_error* mi bismo sve prethodno ponistili !

Ogranicenje kljuca

TipO	*KeyCon*		ograničenje ključa					
T(t)	$Role_1$	Δ	$Mult_1$	1	$AtStr_1$	set	$AtMult_1$	*
	ins	*NoAction*, <<UserDef>>						
	upd	*NoAction*, <<UserDef>>						
TOd	1	jednorelaciono ograničenje						
TOi	r	relaciono ograničenje						
TFz	$Key(N, X), X \subseteq R$							
TPi	1^0: $(\forall u, v \in r(N))(u[X] = v[X] \Rightarrow u = v) \wedge$ 2^0: $(\forall X' \subset X)(\neg\ 1^0(X'))$							

I ovde je za rekapitulaciju data tabelarna predstava specifikacije ogranicenja kljuca. A moguci nacini realizacije su **create/alter table, constraint primary key**(za primarni kljuc), **unique, not null**(za ostale, ekvivalentne kljuceve) i **create trigger**.

- Primer
  - šema relacije
    - *Radnik*({*MBR*, *PRZ*, *IME*, *POL*, *DATR*, *JMBG*}, *C*)
  - ograničenje ključa
    - $Key(Radnik, \{MBR\}), K_p(Radnik) = \{MBR\}$
  - pravilo poslovanja
    - zabranjena modifikacija vrednosti *MBR*
      - u tom slučaju, operacija modifikacije *MBR* ne može narušiti ograničenje ključa
      - za šemu relacije *Radnik*, (*upd*, {*MBR*}, act_j^{ij}) se ne specificira

OgrNaz	*KeyCon_Radnik*			
OgrTip	*KeyCon*	ograničenje ključa		
OgrF	$Key(Radnik, \{MBR\}), K_p(Radnik) = \{MBR\}$			
T(o)	N_1	*Radnik*	ρ_1	Δ
	ins	*	*NoAction*	
	upd	{*MBR*}	Δ	

Nista necemo specificirati za update maticnog broja u radniku jer cemo ustvari jer cemo implementirati pravilo poslovanje koje kaze da je zabranjena modifikacija vrednosti MBR.

Realizacija ogranicenja

- CREATE TABLE,
  CONSTRAINT PRIMARY KEY

```
CREATE TABLE RADNIK
    ( MBR DMBR,
      ...,
      CONSTRAINT KeyCon_Radnik PRIMARY KEY (MBR),
      ...
    )
```

CREATE UNIQUE INDEX ind_mbr_rad
ON Radnik (MBR ASC)

Nad MBR-om ce biti automatski kreiran *unique indeks* . Unique indeks je onaj indeks koji ne dozvoljava duplikate maticnih brojeva radnika (u nasem primeru).

Zabrana nula vrednosti se podrazumeva, te ne moramo da se specificiramo.

Da bi realizovali pravilo poslovanja zabrane modifikacije vrednosti mbr-a u tabeli radnik, moramo definisati triger. (<> znaci !=)

Umesto befora bi smo mogli da kazemo after, ali cim mozemo da testiramo neko ogranicenje ne treba to odlagati nego sto pre testirati zbog performansi.

- CREATE TRIGGER

```
CREATE OR REPLACE TRIGGER PP_Radnik_ZabModPK
    BEFORE UPDATE OF MBR
    ON RADNIK
    FOR EACH ROW
    WHEN (NEW.MBR <> OLD.MBR)
    BEGIN
       Raise_Application_Error(-20000, '<Poruka>');
    END PP_Radnik_ZabModPK;
```

OgrNaz	*KeyCon_Radnik*			
OgrTip	*KeyCon*	ograničenje ključa		
OgrF	*Key*(*Radnik*, {*MBR*}), K_p(*Radnik*) = {*MBR*}			
T(*o*)	N_1	*Radnik*	ρ_1	Δ
	ins	{*MBR*}	*GenNextVal*	
	upd	{*MBR*}	Δ	

Za razliku od prethodnog primera, sada imamo korisnicku definisanu akciju za pokusaj krsenja operacije upisa. Na nama je da opisemo tu funkciju. (jedan od nacina za ovakvu f-ju je sekvencer)

- deklaracija i inicijalizacija generatora sekvenci
- CREATE TABLE, INSERT

```
CREATE TABLE SeqNum
    ( KOLNAZ VARCHAR(30),  -- naziv kolone generatora
      KOLVRED NUMBER,      -- tekuća vrednost brojača
      KOLKORAK NUMBER      -- korak brojanja
      CONSTRAINT con_SeqNumpk PRIMARY KEY (KOLNAZ)
     );

INSERT INTO SeqNum          -- inicijalizacija brojača
    VALUES ('MBR', '1', '1');
```

Jedan od nacina je da mi sami pravimo sekvencer, a drugi nacin je da vec iskoristimo gotov oraclov objekat tipa sekvencer koji cemo da kreiramo za tu priliku. Prvo sto vidimo je nacin na koji mi samo kreiramo sekvencer.

Posto sada hocemo da obezbedimo da se automatski maticni broj inkrementira stalno za jedan jedini nacin da to postignemo je preko trigera. I to before trigerom (posto hocemo da podmetnemo vrednost za maticni broj).

- CREATE TRIGGER

```
CREATE OR REPLACE TRIGGER KeyCon_Radnik_PK_GenSeq
    BEFORE INSERT
    ON RADNIK
    FOR EACH ROW
    BEGIN
       :NEW.MBR := GenNextVal('MBR');
    END KeyCon_Radnik_PK_GenSeq;
```

Drugi nacin je preko oraclovog sekvencera.

Cycle ako imamo, kada dodjemo do maksimalne vrednosti, krenucemo iz pocetka, dok kod *no cycle* cemo javiti gresku kad dodjemo do max vrednosti.

Fora je da je triger duzi i treba mu vise nego za jedno prosto generisanje sledece vrednosti sekvencera(oraclov nacin). Posto se sekvencer u dosta manjem opsegu vremena zakljucava.

Sekvencer(oraclov) blokira ostale procese samo za momenat izvrsenja *nextval* funkcije ali ne I za nivo cele transakcije.

Negativna strana je to sto se on zakljuca samo za taj momenat I vrati vrednost, a mi iz nekog razloga na kraju nardbe mozemo imati rollback. Sto nas dovodi do toga da imamo situaciju (rupu) u kojoj nije iskoriscena ta vrednost sekvencera.

- deklaracija i inicijalizacija generatora sekvenci
- generator sekvenci – poseban objekat SUBP
- CREATE SEQUENCE

```
CREATE SEQUENCE SeqNum_Mbr
    START WITH 1            -- početna vrednost
    INCREMENT 1             -- korak brojanja
    NO CYCLE                -- "nekružni" brojač
```

- CREATE TRIGGER

```
CREATE OR REPLACE TRIGGER KeyCon_Radnik_PK_GenSeq
  BEFORE INSERT
  ON RADNIK
  FOR EACH ROW
  BEGIN
    SELECT SeqNum_Mbr.NEXTVAL
    INTO :NEW.MBR
    FROM SYS.DUAL;
  END KeyCon_Radnik_PK_GenSeq;
```

Ogranicenje jedinstvenosti

TipO	*UniqueCon*		ograničenje jedinstvenosti					
T(t)	$Role_1$	Δ	$Mult_1$	1	$AtStr_1$	set	$AtMult_1$	*
	ins	*NoAction*, *SetNull*, *<<UserDef>>*						
	upd	*NoAction*, *SetNull*, *<<UserDef>>*						
TOd	1	jednorelaciono ograničenje						
TOi	*r*	relaciono ograničenje						
TFz	$Unique(N, X), X \subseteq R$							
TPi	1^0: $(\forall u, v \in r(N))((u[X] \neq \omega \wedge v[X] \neq \omega) \Rightarrow (u[X] = v[X] \Rightarrow u = v)) \wedge$ 2^0: $(\forall X' \subset X)(\neg\, 1^0(X'))$							

Moguci nacini realizacije su **create/alter table, constraint unique, create trigger**. Unique constraint takodje koristimo za sve one kljuceve koji nisu primarni kljuc(alternativni kljucevi).

- Primer
  - šema relacije
    - *Radnik*({*MBR*, *PRZ*, *IME*, *POL*, *DATR*, *JMBG*}, *O*)
  - ograničenje jedinstvenosti *Unique*(*Radnik*, {*JMBG*})

OgrNaz	*UniqCon_Radnik*				
OgrTip	*UniqueCon*	ograničenje jedinstvenosti			
OgrF	*Unique*(*Radnik*, {*JMBG*})				
T(*o*)	N_1	*Radnik*	ρ_1	Δ	
	ins	{*JMBG*}	*NoAction*		
	upd	{*JMBG*}	*NoAction*		

Realizacija ogranicenja

- CREATE TABLE, CONSTRAINT UNIQUE

```
CREATE TABLE RADNIK
    ( ...,
      JMBG DJMBG,
      ...,
      CONSTRAINT UniqCon_Radnik UNIQUE (JMBG),
      ...
    )
```

Ogranicenje referencijalnog integriteta

TipO	*RefInCon*	ograničenje referencijalnog integriteta						
T(*t*)	$Role_1$	*referencing*	$Mult_1$	*1*	$AtStr_1$	*array*	$AtMult_1$	*
	ins	*NoAction*, *SetNull*, *SetDefault*, *<<UserDef>>*						
	upd	*NoAction*, *SetNull*, *SetDefault*, *<<UserDef>>*						
	$Role_2$	*referenced*	$Mult_m$	*1*	$AtStr_m$	*array*	$AtMult_m$	*
	del	*NoAction*, *Cascade*, *SetNull*, *SetDefault*, *<<UserDef>>*						
	upd	*NoAction*, *Cascade*, *SetNull*, *SetDefault*, *<<UserDef>>*						
TOd	*2*	višerelaciono ograničenje ("dvorelaciono")						
TOi	*m*	međurelaciono ograničenje						
TFz	$N_i[X] \subseteq N_j[Y]$, $Key(N_j, Y)$							
TPi	$\pi_X(r(N_i)) \subseteq \pi_Y(r(N_j))$							

Moguci nacini realizacije su **create/alter table, constraint foreign key, create trigger**.

- Primer
  - šeme relacija
    - *Radnik*({*MBR*, ...}, C_1)
    - *Projekat*({*SPR*, ...}, C_2)
    - *Angažovanje*({*SPR*, *MBR*, *BRC*}, C_3)
      - *Key*(*Angažovanje*, {*SPR*, *MBR*})
  - ograničenja referencijalnog integriteta
    - *Angažovanje*[*SPR*] $\subseteq$ *Projekat*[*SPR*], *Key*(*Projekat*, *SPR*)
    - *Angažovanje*[*MBR*] $\subseteq$ *Radnik*[*MBR*], *Key*(*Radnik*, *MBR*)
  - identifikuje se pravilo poslovanja za
    - *Angažovanje*({*SPR*, *MBR*, *BRC*}, C_3)
      - *Key*(*Angažovanje*, {*SPR*, *MBR*})
    - zabranjuje se direktno modifikovanje vrednosti ključa *SPR+MBR*
    - dozvoljava se samo posredna modifikacija vrednosti ključa *SPR+MBR*, kao posledica specifikacija:
      - (*upd*, {*SPR*}, *Cascade*) u *RICon_Angaz_Proj* i
      - (*upd*, {*MBR*}, *Cascade*) u *RICon_Angaz_Radn*

OgrNaz	*RICon_Angaz_Proj*			
OgrTip	*RefInCon*	ograničenje referencijalnog integr.		
OgrF	*Angažovanje*[*SPR*] $\subseteq$ *Projekat*[*SPR*], *Key*(*Projekat*, *SPR*)			
T(*o*)	N_1	*Angažovanje*	ρ_1	*referencing*
	ins	*	*NoAction*	
	upd	{*SPR*}	*NoAction*	
	N_2	*Projekat*	ρ_2	*referenced*
	del	*	*NoAction*	
	upd	{*SPR*}	*NoAction*	

Realizacija ogranicenja

- ALTER TABLE, CONSTRAINT FOREIGN KEY

```
ALTER TABLE Angažovanje
  ADD CONSTRAINT RICon_Angaz_Proj
    FOREIGN KEY (SPR) REFERENCES PROJEKAT(SPR)
                        ON DELETE RESTRICT /* NO ACTION */;

ALTER TABLE Angažovanje
  ADD CONSTRAINT RICon_Angaz_Radn
    FOREIGN KEY (MBR) REFERENCES RADNIK(MBR)
                                  ON DELETE CASCADE;
```

```
CREATE OR REPLACE TRIGGER Cons_Angaz_ProjRad_FK
    BEFORE INSERT OR UPDATE OF SPR, MBR
    ON Angažovanje FOR EACH ROW
    WHEN (OLD.MBR IS NULL OR OLD.MBR != NEW.MBR
            OR OLD.SPR IS NULL OR OLD.SPR != NEW.SPR)
BEGIN
      IF UPDATING AND GlobConsVar.Cons_Angaz_DozvKon THEN
         /* Zabranjuje se direktna modifikacija SPR ili MBR */
         Raise_Application_Error (-20000, '<Poruka>');
      ELSIF INSERTING THEN
         IF NOT (Cons_FK_ProveraProjekat (:NEW.SPR)
                  AND Cons_FK_ProveraRadnik (:NEW.MBR)) THEN
           /* Provera referenciranja ključa SPR i ključa MBR */
           Raise_Application_Error (-20000, '<Poruka>');
         END IF;
      END IF;
END Cons_Angaz_ProjRad_FK;
```

Ogranicenje prosirenog referencijalnog integriteta

TipO	*ExRefInCon*		ograničenje proširenog ref. integriteta					
T(t)	$Role_1$	*referencing*	$Mult_1$	*	$AtStr_1$	*array*	$AtMult_1$	*
	ins	*NoAction*, *SetNull*, *SetDefault*, *<<UserDef>>*						
	upd	*NoAction*, *SetNull*, *SetDefault*, *<<UserDef>>*						
	$Role_2$	*referenced*	$Mult_m$	*	$AtStr_m$	*array*	$AtMult_m$	*
	del	*NoAction*, *Cascade*, *SetNull*, *SetDefault*, *<<UserDef>>*						
	upd	*NoAction*, *Cascade*, *SetNull*, *SetDefault*, *<<UserDef>>*						
TOd	*	višerelaciono ograničenje						
TOi	*m*	međurelaciono ograničenje						
TFz	$(\triangleright\triangleleft N_{k=i_1}^{\,i_m})[X] \subseteq (\triangleright\triangleleft N_{l=j_1}^{\,j_m})[Y]$, $(\exists l \in \{j_1,\ldots,j_m\})(Key(N_l, Y))$							
TPi	$\pi_X(\triangleright\triangleleft_{k=i_1}^{\,i_m}(r(N_k))) \subseteq \pi_Y(\triangleright\triangleleft_{l=j_1}^{\,j_m}(r(N_l)))$							

Moguci nacini realizacije su **create/alter table, constraint check, create trigger, create assertion**.

- Primer
  - šeme relacija
    - *Porudžbenica*({*POIDB*, *PPIDB*}, C_1)
      - *Key*(*Porudžbenica*, {*POIDB*})
    - *PorStavka*({*POIDB*, *ROIDB*, *KOLIC*}, C_2)
      - *Key*(*PorStavka*, {*POIDB*, *ROIDB*})
    - *Cenovnik*({*PPIDB*, *ROIDB*, *CENA*}, C_3)
      - *Key*(*Cenovnik*, {*PPIDB*, *ROIDB*})
  - ograničenje proširenog referencijalnog integriteta
    - (*PorStavka* ▷◁ *Porudžbenica*)[(*PPIDB*, *ROIDB*)] ⊆ *Cenovnik*[(*PPIDB*, *ROIDB*)]

Realizacija ogranicenja

Ogranicenje inverznog referencijalnog integriteta

Realizacija ogranicenja

Zabranjeno je da iz trigera vrsimo bilo kakvu operaciju nad istom tabelom nad kojom je triger i pokrenut ili nad bilo kojom drugom tabelom koja je drugim trigerima ili constraintovima uvezana sa nasom tabelom.

Jedino sto prolazi, jeste *before insert for each* pa onda neki *select*. Ali za ostale tipove trigera, ne mozemo da vrsimo bilo kakve operacije tom tabelom nad kojom je definisan triger (ako sam rekao *on Stavka* ne mogu u *begin* delu da imam bilo kakav insert, update, delete nad tabelom stavka..). Recimo,

imamo triger nad tabelom Radnik operacije insert, u slucaju da u begin delu trigera imamo jos neki insert nad istom tabelom, mi onda imamo beskonacnu petlju, jer bi taj unutrasnji insert opet okinuo triger inserta nad tom tabelom i tako u krug. U statement trigeru je ovo moguce, jer on se izvrsava jednom ,te nema beskonacnih petlji a I vec je poznato stanje tabele nakon/pre operacije a kod row level-a to stanje nije poznato.

Mehanizam kako resavamo medjusobno blokirajuce operacije

```
CREATE OR REPLACE TRIGGER Cons_INVFK_DelStavBDS
  BEFORE DELETE
  ON Stavka...

CREATE OR REPLACE TRIGGER Cons_INVFK_DelStavBDR
  BEFORE DELETE
  ON Stavka FOR EACH ROW...

CREATE OR REPLACE TRIGGER Cons_INVFK_DelStavADS
  AFTER DELETE
  ON Stavka...
```

Idemo sa setom od ova trigera (vrlo cesto se u praksi koristi na rekurzivnim vezama, recimo tabela sa samom sobom itd). Napravimo jedan statement lvl triger. Na stavci koji ce prvo inicijalno jednu globalnu strukturu u paketu, na prazno. Onda u before delete for each row trigeru (row level trigeru) pokupimo sve oznake dokumenta za koje vrsimo brisanje. Na kraju, u after delete statemnet lvl trigeru, izvrsimo proveru da li za svaku nasu brisanu stavku ipak ostaje barem jos jedna stavka za taj isti dokument ako da sve uredu ako ne, onda ne moze. Bitno je da znamo da u statement lvl trgieru imamo mogucnost da napravimo rollback odnosno da napravimo raise_application_error ukoliko je to potrebno. Ovde igramo na redosled u izvodjenju trigeru (na onu sekvencu izvodjenja trigera). Sto znaci da ako vidimo da se nacinila greska (mozemo u after statement level trigeru da uradimo rollback I ponistiimo sve prethodne akcije u okviru sekvence naravno).