Završni ispit iz Baza podataka (40 bodova)

27. lipnja 2019.

Zadaci **1 - 5** odnose se na bazu podataka ***putovanja*** koja se koristi u aplikaciji za planiranje putovanja. Korisnik (***korisnik***) planira putovanje (***put***) na određenu destinaciju (***dest***) i pri tom za svaki dan putovanja (***rbrDan***) određuje lokacije (***obilazak.sifLok***) koje taj dan želi obići. Dodatno se evidentira trenutak dolaska korisnika na lokaciju obilaska (***vrijemeDol***) i odlaska s lokacije (***vrijemeOdl***). Nakon obilaska korisnik može ocijeniti lokaciju cjelobrojnom ocjenom 1-5 (***ocjena***).

Napomena: Pretpostaviti da se evidentira samo prvi dolazak i odlazak s lokacije u danu. Na slici **nisu** prikazane sve ntorke sadržane u relacijama.

korisnik

sifKor	ime	prezime	…	korIme
1	Zlatko	Kovar	…	zkovar
8	Ivana	Horvat	…	ihorvat
...	...	...	...	...

dest

sifDest	nazDest
101	Južna Hrvatska
201	Osijek
…	…

put

sifPut	sifKor	sifDest	datPoc
1003	1	101	29.06.2019
1010	8	201	02.07.2019
…	…	…	..

lok

sifLok	sifDest	nazLok	…
10101	201	Pješački most	…
10102	201	HNK	…
10103	201	Glavna tržnica	…
…	….	….	…

obilazak

sifPut	rbrDan	sifLok	vrijemeDol	vrijemeOdl	ocjena
1003	1	10103	15:30:21	16:18:37	4
1003	2	10101	08:30:04	NULL	NULL
1010	1	10102	NULL.	NULL	NULL
…	…	…	…	…	...

1. (**4 boda**) Napisati jedan SQL upit kojim će se za obilaske, obavljene **vikendom,** destinacija koje u nazivu sadrži riječ 'Hrvatska' ispisati naziv destinacije, naziv lokacije, mjesec u kojem su obilasci obavljeni, prosječno vrijeme provedeno vikendom u tom mjesecu na toj lokaciji i tekst koji ukazuje na duljinu prosječnog trajanja obilaska te lokacije u tom mjesecu: *'kratak'* za prosječni obilazak kraći od 30 minuta, *'srednji'* ako je između 30 minuta i sat i pol i *'dugačak'* ako je dulji od sat i pol. Uzeti u obzir da mogu postojati lokacije jednakog naziva koje nisu iste lokacije.

2. a) (**3 boda**) Uz pretpostavku da su kreirane sve relacije izuzev ***obilazak*** i pri tom, između ostalog, definirani integriteti ključa i referencijski integriteti, napisati naredbu za kreiranje relacije ***obilazak*** i pritom osigurati sljedeće:
   - integritet ključa i entitetski integritet,
   - atributi ***sifPut i sifLok*** smiju poprimiti isključivo vrijednosti istoimenih atributa u relaciji ***put*** odnosno ***lok***, pokušaj brisanja putovanja odnosno lokacije koja je barem jednom obiđena treba spriječiti,
   - osigurati konzistentnost atributa ***ocjena***, ***vrijemeDol***, ***vrijemeOdl*** pri čemu je dozvoljeno: (i) sva tri atributa imaju nepoznatu vrijednost, (ii) samo atribut vrijemeDol ima poznatu vrijednost i (iii) atributi ***vrijemeDol***, ***vrijemeOdl*** imaju poznatu vrijednost pri čemu vrijednost ***vrijemeDol*** mora biti manja od ***vrijemeOdl***. a ***ocjena*** ako je poznata, mora biti između 1 i 5. Ovo integritetsko ograničenje riješiti pomoću CHECK ograničenja, ne pisati okidače.

   Napomena: Domene atributa odabrati proizvoljno ali smisleno.

   b) (**4 boda**) Napisati niz SQL naredbi za kreiranje svih potrebnih objekata kojima će se pri unosu ntorke u relaciju ***obilazak*** osigurati monotono rastuća vrijednost (1,2,3 a ne npr. 2,1,3) atributa ***rbrDan*** za isto putovanje. Uzeti u obzir da korisnik može nadopunjavati plan obilaska za već evidentirane dane. Pri narušavanju opisanog ograničenja korisniku javiti sljedeću grešku: „*Pogreška: neispravan redni broj dana – 2!*“. U prethodnoj poruci broj 2 je vrijednost atributa ***obilazak.rbrDan*** koja narušava opisano pravilo.

3. (**5 bodova**) Administrator baze podataka je nakon kreiranja baze podataka i relacija u shemi ***public*** obavio sljedeće SQL naredbe:

```
REVOKE CONNECT ON DATABASE putovanja FROM PUBLIC;
REVOKE ALL ON SCHEMA public FROM PUBLIC;
```

   Napisati SQL naredbe kojima će administrator:
   a) svim korisnicima (trenutnim i budućim) koji imaju (ili će tek dobiti) ovlasti uspostave korisničke sjednice i spajanja na bazu podataka ***putovanja*** omogućiti
      - pristup objektima sadržanima u shemi ***public***
      - pregled podataka o svim destinacijama i lokacijama

   b) korisniku *zkovar* dodijeliti ovlasti za uspostavu korisničke sjednice (lozinku mu postaviti na *pwd4562*), spajanje na bazu podataka ***putovanja*** te pregled, unos, izmjenu i brisanje podataka o vlastitom korisničkom računu, vlastitim putovanjima i vlastitim obilascima.

4. (**3 boda**) Relacija **korisnik**(*sifKor*, *ime, prezime, datRod, korIme*) sadrži ntorke sa sljedećim vrijednostima atributa *sifKor*: 1, 8, 13, 16, 19, 23, 25, 37, 39, 43, 45, 48, 51. Nacrtajte B^+ stablo reda 5 za atribut *sifKor* tako da popunjenost stabla bude minimalna.

5. **(5 bodova)** Prikazan je sadržaj relacije **dest** u trenutku započinjanja naredbe {1}. Uz pretpostavku da osim transakcija A, B i C niti jedna druga transakcija ne obavlja nikakvu operaciju nad relacijom **dest**, odredite rezultate izvođenja naredbi {6}, {7}, {8}, {10} i {11}. Pojavljuju li se u transakcijama karakteristični problemi istodobnog pristupa? Ako da, navedite naziv problema, pri obavljanju koje naredbe je primijećen i kojom naredbom je uzrokovan.

dest

sifDest	nazDest
101	Južna Hrvatska
201	Osijek

	Transakcija A		**Transakcija B**		**Transakcija C**
{1}	`BEGIN TRANSACTION;`	{2}	`BEGIN TRANSACTION;` `SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;`	{3}	`BEGIN TRANSACTION;`
{4}	`INSERT INTO dest` `VALUES (301, 'Plitvice');`	{5}	`UPDATE dest SET nazDest = 'Smiljan'` `WHERE sifDest = 101;`		
{6}	`SELECT * FROM dest;`	{7}	`SELECT * FROM dest;`	{8}	`SELECT * FROM dest;`
{9}	`COMMIT TRANSACTION;`	{10}	`SELECT * FROM dest;`	{11}	`SELECT * FROM dest;`
		{12}	`COMMIT TRANSACTION;`	{13}	`COMMIT TRANSACTION;`

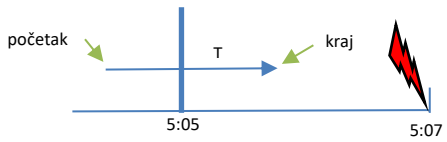
6. **(6 bodova)** U bazi podataka na PostgreSQL SUBP kreirane su relacije r, s i t sa shemama: R(A, B, C, D), S(E, F, G) i T(H, E, B, T). Podcrtani atributi su primarni ključevi relacija. U bazi podataka su definirana ograničenja primarnih ključeva i stranog ključa kojim se relacija t preko atributa E poziva na relaciju s, te su kao posljedica kreirani odgovarajući indeksi (oni koje PostgreSQL kreira automatski). Dodatno je nad relacijom ***t*** definiran indeks `CREATE INDEX i1 on t(T)`. Optimizator upita raspolaže sljedećim statističkim podacima:

r(A, B, C, D)	s(E, F, G)	t(H, E, B, T)
N(r) = 90000	N(s) = 60000	N(t) = 90000
V(B, r) = 300	V(F, s) = 600	V(E, t) = 300
V(C, r) = 500	V(G, s) = 800	V(B, t) = 600
V(D, r) = 300		V(T, t) = 600

Obavlja se operacija: $\sigma_{C<'c' \wedge D='d' \wedge G<'g' \wedge T='t'}(r \rhd\lhd t \rhd\lhd s)$. Nacrtajte **inicijalno** i **konačno** stablo upita nakon provedene heurističke optimizacije. Prilikom crtanja inicijalnog stabla upita pretpostavite da je redoslijed spajanja relacija određen redoslijedom kojim su one navedene u izrazu relacijske algebre. Redoslijed spajanja relacija prilikom crtanja konačnog stabla upita odrediti temeljem procjene broja ntorki u rezultatima spajanja. Navesti izraze prema kojima je obavljena procjena broja ntorki u svim međurezultatima. U stablu upita naznačiti očekivani broj ntorki u svim međurezultatima te korištene metode pristupa.

7. **(3 boda)** U SUBP je došlo do pogreške na razini računalnog sustava u **5:07h**. Dnevnik izmjena uspješno je očuvan. Skicirajte transakcije T1, T2 i T3 tako da su za svaku transakciju naznačeni mogući početak i završetak s obzirom na vremenski interval od **05:05h** kada je bila posljednja kontrolna točka do **05:07h** kada je nastao kvar (primjer je prikazan na slici), ako je poznato sljedeće:
   - za transakciju T1 ne treba obaviti nikakvu akciju
   - transakciju T2 treba poništiti
   - transakciju T3 treba ponovno obaviti

   Napomena: Interval između dvije kontrolne točke je 5 minuta.



8. (**7 bodova**) U bazi podataka se evidentiraju podaci o poslovanju tvrtke koja se bavi prijevozom putnika (tvrtka nalik npr. Uberu). Korisnici aplikacije mogu biti putnici(klijenti) i/ili vozači. Za sve korisnike se evidentira OIB, ime, prezime i datum rođenja. Jedini mogući način plaćanja vožnji je karticom, pa se za putnike dodatno evidentiraju podaci o kartici (samo jednoj) i to vrsta kartice (šifra vrste kartice, naziv vrste kartice, npr 1- Amex, 2- Diners,…), broj kartice, datum isteka te sigurnosni kod. NAPOMENA: podatci o karticama spremat će se u bazu podataka u kriptiranom obliku, ali to ne utječe na rješenje ovog zadatka. Za vozača se dodatno evidentira kategorija i datum izdavanja aktualne vozačke dozvole.

   Tvrtka izrađuje plan radnog vremena vozača tako da se evidentiraju periodi u kojima vozači trebaju obavljati vožnje (šifra perioda, trenutak početka i završetka, vozač i vozilo). Može postojati više perioda s istim trenutkom početka i završetka (ali različitom šifrom), ovisno o tome s koliko vozača tvrtka raspolaže. Svaki period se dodjeljuje jednom vozaču tijekom kojeg vozi jedno vozilo tvrtke (šifra vozila, datum prve registracije, registarska oznaka), ne nužno isto tijekom svakog perioda. Isto vozilo mogu koristiti različiti vozači, ali u različitim periodima.

   Za pojedinu vožnju, pored putnika, evidentira se redni broj vožnje obavljen u konkretnom periodu (za kojeg je poznat vozač i vozilo), trenutak početka i trenutak završetka vožnje, geografska širina i geografska dužina početne te konačne lokacije.

   Nacrtajte ER model baze podataka. Navedite sheme entiteta i sheme veza (označiti ključeve). Svaki entitet (osim slabih entiteta) opisati **isključivo vlastitim atributima**. Nužno je da sve sheme zadovoljavaju **3NF**.

Rješenja:

1. (4 boda)

```
SELECT nazDest, nazLok, EXTRACT(MONTH FROM datPoc+rbrDan-1), AVG(vrijemeOdl-vrijemeDol),
       CASE
          WHEN AVG(vrijemeOdl-vrijemeDol) < '30 min' THEN 'kratak'
          WHEN AVG(vrijemeOdl-vrijemeDol) BETWEEN  '30 min' AND '1H 30 min' THEN 'srednji'
          ELSE 'dugačak'
       END
FROM obilazak
NATURAL JOIN lok
NATURAL JOIN dest
NATURAL JOIN put
WHERE nazDest LIKE '%Hrvatska%'
AND EXTRACT(DOW FROM datPoc+rbrDan-1) IN (0,6)
GROUP BY nazDest, nazLok, sifLok, EXTRACT(MONTH FROM datPoc+rbrDan-1)
ILI:
SELECT nazDest, nazLok, EXTRACT(MONTH FROM datPoc+rbrDan-1), AVG(vrijemeOdl-vrijemeDol),
       CASE
          WHEN AVG(vrijemeOdl-vrijemeDol) < '30 min' THEN 'kratak'
          WHEN AVG(vrijemeOdl-vrijemeDol) BETWEEN  '30 min' AND '1H 30 min' THEN 'srednji'
          ELSE 'dugačak'
       END
FROM obilazak
  INNER JOIN put ON put.sifPut = obilazak.sifPut
  INNER JOIN dest ON dest.sifDest = put.sifDest
  INNER JOIN lok ON lok.sifLok = obilazak.sifLok
    AND lok.sifDest = dest.sifDest
WHERE nazDest LIKE '%Hrvatska%'
AND EXTRACT(DOW FROM datPoc+rbrDan-1) IN (0,6)
GROUP BY nazDest, nazLok, obilazak.sifLok, EXTRACT(MONTH FROM datPoc+rbrDan-1);
ILI:
SELECT nazDest, nazLok, EXTRACT(MONTH FROM datPoc+rbrDan-1), AVG(vrijemeOdl-vrijemeDol),
       CASE
          WHEN AVG(vrijemeOdl-vrijemeDol) < '30 min' THEN 'kratak'
          WHEN AVG(vrijemeOdl-vrijemeDol) BETWEEN  '30 min' AND '1H 30 min' THEN 'srednji'
          ELSE 'dugačak'
       END
FROM obilazak
    INNER JOIN put USING(sifPut)
    INNER JOIN dest USING(sifDest)
    INNER JOIN lok USING(sifLok, sifDest)
WHERE nazDest LIKE '%Hrvatska%'
AND EXTRACT(DOW FROM datPoc+rbrDan-1) IN (0,6)
GROUP BY nazDest, nazLok, sifLok, EXTRACT(MONTH FROM datPoc+rbrDan-1);
```

2.

a) (3 boda)

```
CREATE TABLE obilazak (
    sifPut INT CONSTRAINT fkObilazakPut REFERENCES put(sifPut),
    rbrDan INT,
    sifLok INT CONSTRAINT fkObilazakLok REFERENCES lok(sifLok),
    vrijemeDol TIME,
    vrijemeOdl TIME,
    ocjena SMALLINT ,
    CONSTRAINT chkVrijemeOcjena CHECK
          ((vrijemeDol IS NULL AND vrijemeOdl IS NULL AND ocjena IS NULL) OR
           (vrijemeDol IS NOT NULL AND vrijemeOdl IS NULL AND ocjena IS NULL) OR
           (vrijemeDol IS NOT NULL AND vrijemeOdl IS NOT NULL AND vrijemeOdl > vrijemeDol AND (ocjena IS
NULL OR ocjena BETWEEN 1 AND 5))),
    PRIMARY KEY(sifPut, rbrDan, sifLok)
 );

ILI:
```

```
CREATE TABLE obilazak (
    sifPut INT CONSTRAINT fkObilazakPut REFERENCES put(sifPut),
    rbrDan INT,
    sifLok INT CONSTRAINT fkObilazakLok REFERENCES lok(sifLok),
    vrijemeDol TIME,
    vrijemeOdl TIME,
    ocjena SMALLINT ,
    CONSTRAINT chkVrijemeOcjena CHECK
          ((vrijemeOdl IS NULL AND ocjena IS NULL) OR
           (vrijemeOdl > vrijemeDol AND (ocjena IS NULL OR ocjena BETWEEN 1 AND 5))),
    PRIMARY KEY(sifPut, rbrDan, sifLok)
 );
```

b) (4 boda)

```
CREATE FUNCTION chkObilazak() RETURNS trigger AS $$
   BEGIN
      --kada ne postoji nijedan zapis u obilazak za taj sifPut, rbrDan mora biti 1
      IF (NEW.sifPut NOT IN (SELECT sifPut FROM obilazak) AND NEW.rbrDan <> 1) THEN
         RAISE EXCEPTION 'Pogreška: Neispravan redni broj dana - %!', NEW.rbrDan;
         RETURN NULL;
      END IF;
      --kada  postoji barem jedan zapis u obilazak za taj sifPut, rbrDan može biti neki od postojećih ili
za 1 veći od najvećeg rbrDan bez netom insertiranog
      IF (NEW.rbrDan NOT IN (SELECT rbrDan FROM obilazak WHERE sifPut = NEW.sifPut) AND
          (SELECT MAX(rbrDan) FROM obilazak WHERE sifPut = NEW.sifPut) != NEW.rbrDan - 1) THEN
         RAISE EXCEPTION 'Pogreška: Neispravan redni broj dana - %!', NEW.rbrDan;
         RETURN NULL;
      END IF;
      RETURN NEW;
   END;
$$ LANGUAGE plpgsql;
ILI:
CREATE FUNCTION chkObilazak() RETURNS trigger AS $$
   BEGIN
      IF (NEW.sifPut NOT IN (SELECT sifPut FROM obilazak) AND NEW.rbrDan <> 1) OR
         ((SELECT MAX(rbrDan) FROM obilazak WHERE sifPut = NEW.sifPut) != NEW.rbrDan - 1) THEN
         RAISE EXCEPTION 'Pogreška: Neispravan redni broj dana - %!', NEW.rbrDan;
      END IF;
      RETURN NEW;
   END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER insObilazak
BEFORE INSERT ON obilazak
FOR EACH ROW
EXECUTE FUNCTION chkObilazak();
```

3. (5 bodova)

a)

```
GRANT USAGE ON SCHEMA public TO public;
GRANT SELECT ON lok, dest to public;
```

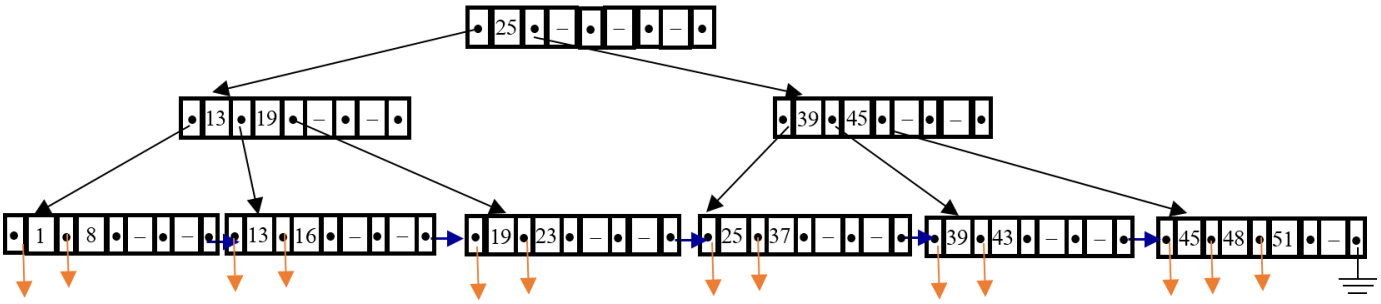
b)

```
CREATE VIEW vMojKorisnik AS
SELECT *
  FROM korisnik
 WHERE korIme = SESSION_USER //ovdje može i CURRENT_USER
WITH CHECK OPTION;

CREATE VIEW vMojaPut AS
SELECT *
  FROM put
 WHERE sifKor IN (SELECT sifKor
                    FROM korisnik
                   WHERE korIme = SESSION_USER)
```

```
WITH CHECK OPTION;

CREATE VIEW vMojiObilasci AS
SELECT *
  FROM obilazak
 WHERE EXISTS (SELECT *
                 FROM put
                 NATURAL JOIN korisnik
                 WHERE korIme = SESSION_USER
                   AND put.sifPut = obilazak.sifPut)
WITH CHECK OPTION;

ILI:

CREATE VIEW vMojKorisnik AS
SELECT *
  FROM korisnik
WHERE korIme = SESSION_USER
WITH CHECK OPTION;

CREATE VIEW vMojPut AS
SELECT *
  FROM put
WHERE sifKor IN (SELECT sifKor
                   FROM vMojKorisnik)
WITH CHECK OPTION;

CREATE VIEW vMojiObilasci AS
SELECT *
  FROM obilazak
WHERE sifPut IN (SELECT sifPut
                 FROM vMojPut)
WITH CHECK OPTION;

CREATE USER zkovar WITH PASSWORD 'pwd4562';
GRANT CONNECT ON DATABASE putovanje TO zkovar;
GRANT SELECT, INSERT, UPDATE, DELETE ON vMojKorisnik, vMojaPut, vMojiObilasci TO zkovar;
```

4. (3 boda)

	min	max
korijen	2	n = 5
Interni čvor	$\lceil n/2 \rceil = 3$	n = 5
list	$\lceil (n-1)/2 \rceil = 2$	n-1 =4



5. (5 bodova)

{6}

sifDest	nazDest
101	Južna Hrvatska
201	Osijek
301	Plitvice

{7}

sifDest	nazDest
101	Smiljan
201	Osijek

{8}

sifDest	nazDest
101	Južna Hrvatska
201	Osijek

{10}

sifDest	nazDest
101	Smiljan
201	Osijek

{11}

sifDest	nazDest
101	Južna Hrvatska
201	Osijek
301	Plitvice

U transakciji C pri obavljanju naredbe {11} prisutan je problem poznat pod nazivom „Sablasne n-torke“. Transakcija

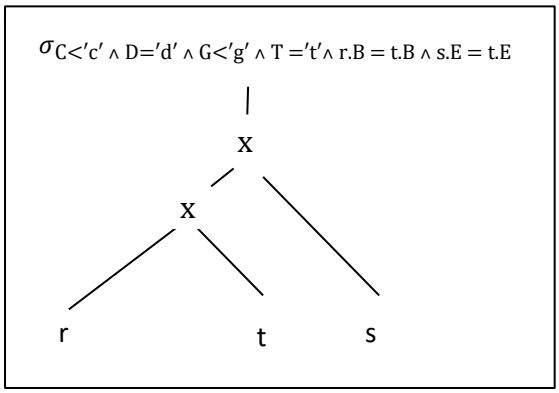
ponovljenim čitanjem (nakon {8} obavlja identičnu naredbu {11}) dobiva različit rezultat koji je posljedica INSERT naredbe transakcije A – {4}.

6. (6 bodova)

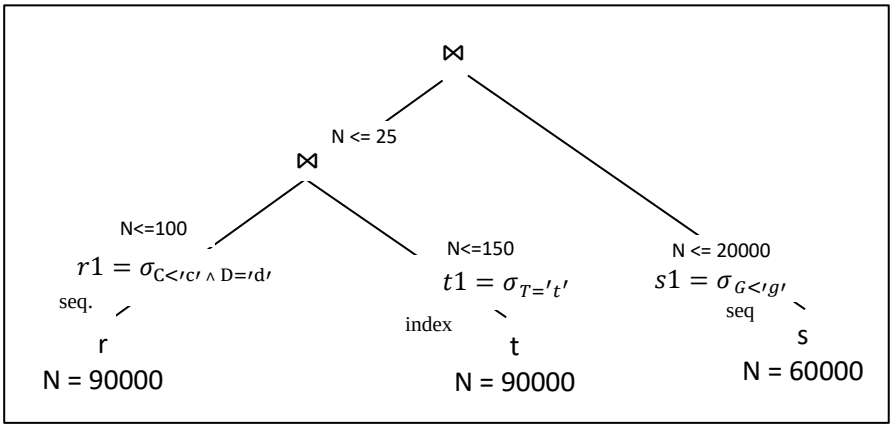
Procjena veličine međurezultata nakon potiskivanja selekcije:	*Procjena veličine međurezultata za različite redoslijede spajanja:*
r1= $\sigma_{C<'c' \wedge D='d'}(r)$ N(r1) = N(r)/3/V(D,r) = 90000/3/300 = **100**	(r1 ⊳⊲ s1) – Kartezijev produkt
s1= $\sigma_{G<'g'}$ N(s1) = N(s)/3 = 60000/3 = **20000**	N(r1⊳ ⊲s1) = N(r1)*N(s1) = 100 * 20000 = **2 000 000**
t1= $\sigma_{T='t'}$ N(t1) = N(t)/V(T, t) = 90000/600 = **150**	(s1 ⊳⊲ t1) – strani ključ u t je primarni ključ u s
	N(s1⊳ ⊲t1) <= N(t1) = **150**
	(r1 ⊳⊲ t1) – preko atributa B (nije ključ niti u r1 niti u t1)
	N(r1⊳ ⊲t1) = N(r1)*N(t1)/max(V(B, r), V(B, t)) = (100*150)/MAX(300, 600) = **25**

Redoslijed spajanja: prvo (r1 ⋈ t1) pa potom međurezultat sa s1

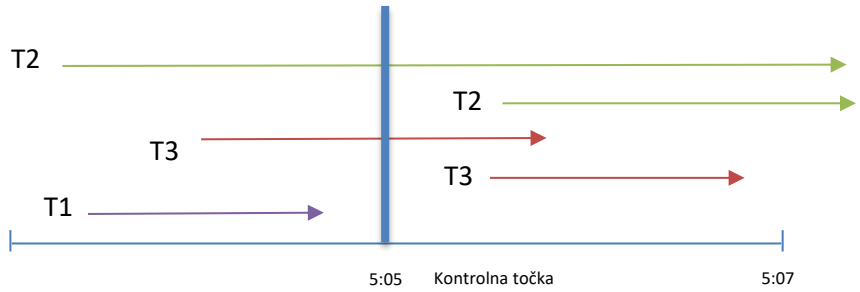
Inicijalno stablo:



Konačno stablo:



7. (3 boda)



- T1 - transakcija T1 mora započeti i završiti prije kontrolne točke
- T2 – neovisno u kojem trenutku između intervala 05:00h i 05:07h je transakcija započela, nije završila prije 05:07h (mogla je započeti i prije i poslije kontrolne točke)
- T3 – neovisno u kojem trenutku između intervala 05:00h i 05.07h je transakcija započela završila je prije 05:07h (mogla je započeti i prije i poslije kontrolne točke)

8. (7 bodova)

KORISNIK	**PUTNIK**	**VOZAČ**	**VRKART**	**VOZILO**	**PERIODRADVR**	**VOZNJA**
OIB	OIB	OIB	sifVrKart	sifVozilo	sifPeriodRV	sifPeriodRV
ime		datVoz	nazVrKart	datPrvaReg	datVrijPoc	rbrVoznja
prez		katVoz		regOzn	datVrijKraj	datVrijPoc
datRod						datVrijKraj
						geoDuzina
						geoSirina
K ={OIB}	K ={OIB}	K ={OIB}	K ={sifVrKart	K1=PK={sifVozilo}	K = { sifPeriodRV}	K1 = PK ={ sifPeriodRV, rbrVoznja}
				K2 = {regOzn}		K2 ={ sifPeriodRV, datVrijemeOd}

putPlaca	**putnikVoz**	**periodVoz**	**vozacRasp**	**voziloRasp**
OIBPutnik	sifPeriodRV	sifPeriodRV	sifPeriodRV	sifPeriodRV
sifVrKart	rbrVoznja	rbrVoznja	OIBVozac	sifVozilo
brKartica	OIBPutnik			
datIstek				
sigKod				
K = {OIBPutnik}	K = {sifPeriodRV, rbrVoznja}	K = {sifPeriodRV, rbrVoznja}	K = { sifPeriodRV}	K = { sifPeriodRV}