1. **(4 boda)**

Zadatak riješite **jednom SQL naredbom**.

Za sve filmove koji imaju titlove na hrvatskom jeziku (naziv jezika je 'Croatian'), traju **duže od** sat vremena, a njihov naslov **ne sadrži** znak `:` (dvotočka), ispišite naziv filma zajedno s godinom objave filma (u istom stupcu - pogledati predložak), trajanje filma i zaradu.

Zapise u rezultatu poredajte po nazivu filma abecedno. Godinu objave ispisati u zagradi odvojeno od naziva filma jednim razmakom. Stupce nazvati prema danom predlošku (podaci u predlošku ne moraju odgovarati stvarnima).

tracktitle	duration	boxincome
Airplane Mode (2019)	01:37:00	10854621
Bay In The Morning (2020)	01:04:00	null
...	...	...

```
SELECT tracktitle || ' (' || EXTRACT(YEAR FROM releasedate) || ')'
       AS tracktitle,
     duration, boxincome
FROM movie
   NATURAL JOIN track
   NATURAL JOIN subtitle
   JOIN language
       ON subtitle.subtitlelangid = language.langid
WHERE langname = 'Croatian'
  AND duration > '1 hour'
  AND tracktitle NOT LIKE '%:%'
ORDER BY tracktitle;
```

2. **(4 boda)**

Film „Ghost Rider" dobio je svoj nastavak nazvan „Ghost Rider: Spirit of Vengeance" koji je izdan 11. 12. 2011. u trajanju od 95 minuta s oznakom ageRestriction dozvoljen starijima od 13 godina (uključivo). Ocjena filma u bazi nije dostupna (jer još nijedan korisnik nije pogledao film), a prihod filma bio je 132,6 milijuna dolara.

Potrebno je napisati niz INSERT naredbi koje će:

1) Dodati zapis o filmu u odgovarajuće tablice kako bi se sve poznate informacije ispravno pohranile
2) Dodati zapis o jeziku zvučnog zapisa filma na engleskom jeziku (naziv jezika je 'English') koji je ujedno i izvorni jezik
3) Dodati zapis o prijevodu filma na engleski jezik

INSERT naredbe ne smiju sadržavati eksplicitne vrijednosti stranih ključeva, već ih moraju dinamički dohvaćati tijekom izvedbe naredbe.

```
INSERT INTO track (trackTitle, releaseDate, duration, tageRestriction,
trackrating)
VALUES ('Ghost Rider: Spirit of Vengeance', '11.12.2011.', '95
minutes'::INTERVAL, 13, NULL);

INSERT INTO movie(trackId, boxIncome, prevMovieId)
VALUES      ((SELECT trackId FROM track WHERE lower(trackTitle)='ghost rider:
spirit of vengeance'), 132600000,
            (SELECT trackId FROM track WHERE lower(trackTitle) = 'ghost
rider'));

INSERT INTO audioLang(trackId, audioLangId, isNative)
VALUES ((SELECT trackId FROM track WHERE lower(trackTitle)='ghost rider:
spirit of vengeance'), (SELECT langId FROM language WHERE langname =
'English'), 1);

INSERT INTO subTitle(trackId, subtitleLangId)
VALUES ((SELECT trackId FROM track WHERE lower(trackTitle)='ghost rider:
spirit of vengeance'), (SELECT langId FROM language WHERE langname =
'English'));
```

3. (6 bodova)

Napišite **jednu SQL naredbu** kojom će se za svaku aktivaciju paketa koja je trajala barem 3000 dana (uključujući i trenutno aktivne pakete gledano **trenutak** izvedbe upita) ispisati naziv paketa, email vlasnika računa, **datum** aktivacije paketa, **datum** deaktivacije paketa te nazive svih profila koji tijekom trajanja aktivacije paketa nisu pokrenuli gledanje niti jednog **filmskog** sadržaja.

Ukoliko su svi profili za vrijeme trajanja aktivacije ovakvih paketa pokrenuli barem jedan filmski sadržaj, za ime profila ispisati NULL.

Ispis poredati po e-mailu vlasnika računa abecedno, a potom po imenu profila abecedno. Stupce nazvati prema danom predlošku (podaci u predlošku ne moraju odgovarati stvarnima).

nazivpaketa	emailVlasnika	datumAkt	datumDeakt	profilBezFilmskihSadrzaja
Standard	aaron.eckhart@gmail.com	02.09.2009.	02.03.2017.	null
Premium	abe.williams@gmail.co.uk	12.03.2012.	null	mildJean
Premium	abe.williams@gmail.co.uk	12.03.2012.	null	savageSteve
Basic	ace.ventura@aol.com	02.06.2004.	02.03.2016.	bobDaBuilder
...	...	...		

```
SELECT packname AS nazivPaketa, oemail AS emailVlasnika,
    packStartDateTime::DATE AS datumAkt, packEndDateTime::DATE AS datumDeakt,
    profile.profilename AS profilBezFilmskihSadrzaja
FROM pack NATURAL JOIN ownerpack NATURAL JOIN owner
LEFT JOIN profile ON profile.ownerid = owner.ownerid
  AND (SELECT COUNT(*)
        FROM trackview NATURAL JOIN track NATURAL JOIN movie
        WHERE trackview.profilename = profile.profilename
         AND trackview.ownerid = profile.ownerid
         AND viewStartDateTime > ownerpack.packstartDateTime
         AND (viewStartDateTime < ownerpack.packEndDateTime OR
packEndDateTime IS NULL))                                  = 0
WHERE packEndDateTime - packStartDateTime >= '3000 days' OR (
      packEndDateTime IS NULL AND (CURRENT_TIMESTAMP - packStartDateTime >=
'3000 days'))
ORDER BY oemail, profilename;


--ovo dalje nije dio rješenja –Upit za testiranje CURRENT_TIMESTAMP vs
CURRENT_DATE:



SELECT packname AS nazivPaketa, oemail AS emailVlasnika,
      CASE WHEN packEndDateTime IS NULL THEN CURRENT_TIMESTAMP -
packStartDateTime
        ELSE packEndDateTime - packStartDateTime
        END AS timestampCalc,
         CASE WHEN packEndDateTime IS NULL THEN CURRENT_DATE -
packStartDateTime
        ELSE packEndDateTime - packStartDateTime
        END AS dateCalc,

    packStartDateTime::DATE AS datumAkt, packEndDateTime::DATE AS datumDeakt,
    profile.profilename AS profilBezFilmskihSadrzaja
FROM pack NATURAL JOIN ownerpack NATURAL JOIN owner
LEFT JOIN profile ON profile.ownerid = owner.ownerid
  AND (SELECT COUNT(*)
        FROM trackview NATURAL JOIN track NATURAL JOIN movie
        WHERE trackview.profilename = profile.profilename
         AND trackview.ownerid = profile.ownerid
         AND viewStartDateTime > ownerpack.packstartDateTime
         AND (viewStartDateTime < ownerpack.packEndDateTime OR
packEndDateTime IS NULL))                                  = 0
WHERE packEndDateTime - packStartDateTime >= '3000 days' OR (
      packEndDateTime IS NULL AND (CURRENT_TIMESTAMP - packStartDateTime >=
'3000 days'))
ORDER BY oemail, profilename;
```

4. (4 boda)

Relacijska shema AKTIVNOST_STUD sadrži sljedeće atribute:

JMBAG	JMBAG studenta
ime	ime studenta
prezime	prezime studenta
akgodina	akademska godina (npr. 2020)
sifPredmet	šifra predmeta
nazPredmet	naziv predmeta
sifAktivnost	šifra aktivnosti
nazAktivnost	naziv aktivnosti (npr. 1. domaća zadaća)
sifVrAktivnost	šifra vrste aktivnosti
nazVrAktivnost	naziv vrste aktivnosti (npr. domaća zadaća)
maksBrBodAktPred	maksimalan broj bodova za aktivnost na predmetu
brBodStudAktivnost	broj bodova koje je student postigao na nekoj aktivnosti na određenom **predmetu** u određenoj **akademskoj godini**

Potrebno je normalizirati relacijsku shemu AKTIVNOST_STUD **postepeno** na 1NF, 2NF i 3NF te navesti nazive relacijskih shema od kojih se sastoji konačno rješenje (shema baze podataka AKTIVNOST_STUDENTA), ako vrijede sljedeća pravila:

- student isti predmet može pohađati više puta (u različitim akademskim godinama)
- različiti predmeti mogu imati iste aktivnosti, s istim ili različitim maksimalnim brojem bodova;
- aktivnost pripada jednoj vrsti aktivnosti;
- aktivnosti za predmet (i maksimalni broj bodova) uvijek su iste za isti predmet (ne ovise o akademskoj godini);

1NF

R1 = JMBAG, akgodina, sifPredmet, sifAktivnost, ime, prezime, nazPredmet, nazAktivnost, sifVrAktivnost, nazVrAktivnost, maksBrBodAktPred, brBodStudAktivnost

2NF

STUDENT = JMBAG, ime, prezime

PREDMET = sifPredmet, nazPredmet

AKT = sifAktivnost, nazAktivnost, sifVrAktivnost, nazVrAktivnost

PRED_AKT = sifPredmet, sifAktivnost, maksBrBodAktPred

OSTVARIO= JMBAG, akgodina, sifPredmet, sifAktivnost, brBodStudAktivnost

3NF

AKTIVNOST = sifAktivnost, nazAktivnost, sifVrAktivnost

VRAKTIVNOST = sifVrAktivnost, nazVrAktivnost

AKTIVNOST_STUDENTA = STUDENT, PREDMET, PRED_AKT, OSTVARIO, AKTIVNOST, VRAKTIVNOST

5. (5 bodova)

Gledatelj iza pojedinog korisničkog profila može ocijeniti sviđa li mu se određeni sadržaj ili ne (***profileTrack.liked***, poprima vrijednost 1 ili -1). Za pojedini medijski sadržaj bilježi se ocjena (***track.trackRating***) koja se računa prema sljedećem izrazu:

$$\mathrm{MAX}\left(0.00, 100.00 * \frac{zbroj\ svih\ ocjena}{ukupan\ broj\ svih\ ocjena}\right)$$

Na primjer, ako je određeni sadržaj ocijenjen 4 puta, od čega je vrijednost tri puta bila jednaka 1, a jednom -1, rezultat je 50.00. U slučaju da se dobije negativan rezultat, ocjenu medijskog sadržaja (***track.trackRating***) treba postaviti na 0.00.

Kada korisnik (preko korisničkog profila) prvi put pokrene neki medijski sadržaj prvo nastaje n-torka u ***profileTrack***, a tek potom n-torka u ***trackView*** za taj profil i medijski sadržaj. Zbog toga nema smisla dozvoliti ocjenjivanje sadržaja pri unosu n-torke u ***profileTrack***.

Napisati niz SQL naredbi za kreiranje svih potrebnih objekata kojima će se osigurati sljedeće:

- zabraniti ocjenjivanje medijskog sadržaja pri unosu n-torke u ***profileTrack***. Pri narušavanju ovog ograničenja korisniku javiti sljedeću grešku: 'Pogreška: nije dozvoljeno ocijeniti sadržaj prije početka gledanja.'.
- prilikom evidencije informacije o tome sviđa li se nekome određeni medijski sadržaj (***profileTrack.liked)*** ažurnom održavati njegovu ocjenu (***track.trackRating***).

```
CREATE FUNCTION promijeniTrackRating() RETURNS TRIGGER AS $$
  DECLARE  p_trackRating     track.trackRating%TYPE;
  BEGIN
     SELECT CASE
                 WHEN (100. * SUM(liked)/COUNT(liked))  > 0
                 THEN ROUND(100. * SUM(liked)/COUNT(liked),2)
                  ELSE 0.00
            END
     INTO p_trackRating
     FROM profiletrack
     WHERE trackid = NEW.trackid;

     UPDATE track
     SET trackRating = p_trackRating
     WHERE trackid = NEW.trackid;
```

```
      RETURN NEW;
     END;
$$ LANGUAGE plpgsql;

CREATE FUNCTION chkLikedNotNull() RETURNS TRIGGER AS $$
   BEGIN
    RAISE EXCEPTION 'Pogreška: nije dozvoljeno ocijeniti sadržaj prije početka
gledanja.';
RETURN NEW;
   END;
$$ LANGUAGE plpgsql;


--DROP TRIGGER updProfileTrack ON profileTrack
CREATE TRIGGER updProfileTrack
AFTER UPDATE OF liked ON profileTrack
FOR EACH ROW
WHEN (NEW.liked <> OLD.liked)
EXECUTE FUNCTION promijeniTrackRating();

--DROP TRIGGER insProfileTrack ON profileTrack
CREATE TRIGGER insProfileTrack
BEFORE INSERT ON profileTrack
FOR EACH ROW
WHEN (NEW.liked IS NOT NULL)
EXECUTE FUNCTION chkLikedNotNull();


--ovo dalje nije dio rješenja – služi za provjeru

/*

Provjera npr.:

update profiletrack set liked = 1 where ownerid=152 and profilename =
'MelissaPowell' and trackid = 4790;

select * from profileTrack where trackid = 4790;

select * from track where trackid = 4790;

update profiletrack set liked = -1 where ownerid=152 and profilename =
'MelissaPowell' and trackid = 4790;

*/
```

6. (5 bodova)

Administrator baze podataka je nakon kreiranja baze podataka *streamflix* i tablica u shemi *public* obavio sljedeću SQL naredbu:

```
REVOKE ALL ON SCHEMA public FROM public;
```

a) Napisati SQL naredbe kojima će administrator sustava:

- kreirati ulogu *horrorRatingAdmin*,
- ulozi *horrorRatingAdmin* dati ovlasti za pristup objektima sadržanima u shemi *public* s mogućnošću stvaranja novih objekata,
- omogućiti ulozi *horrorRatingAdmin* **pregled** šifre, naziva i ocjene (track*Rating*) medijskih sadržaja (*track*) koji su serije (imaju definirani podatak o epizodi u *showEp*) i spadaju u žanr strave (*Horror*). Ulozi *horrorRatingAdmin* treba omogućiti **izmjenu** samo ocjene (*trackRating*) takvih medijskih sadržaja. Kreirajte sve objekte potrebne za dodjelu opisanih ovlasti ulozi *horrorRatingAdmin*,
- kreirati korisnika *wadamson* s lozinkom '*wayne123*',
- korisniku *wadamson* dati mogućnost korištenja uloge *horrorRatingAdmin*,

b) Nakon što je korisnik *wadamson* kreirao tablicu *ratingStats* u shemi *public*, napisati sve potrebne SQL naredbe kojima će korisnik *wadamson* omogućiti korisniku *sking*:

- pregled sadržaja tablice *ratingStats,* uz mogućnost dodjeljivanja tih istih ovlasti ostalim korisnicima.

U odgovorima naznačite koji korisnik obavlja koje SQL naredbe.

```
a)
admin:
CREATE ROLE horrorRatingAdmin;
GRANT USAGE,CREATE ON SCHEMA public TO horrorRatingAdmin;

CREATE VIEW horror_series AS
select trackid,tracktitle,trackrating from track
where trackId in (select trackId
                  from track
                       natural join trackgenre
                       natural join genre
                       natural join showep
                  where genreName='Horror')
WITH CHECK OPTION;

GRANT SELECT,UPDATE(trackRating) ON horror_series TO horrorRatingAdmin;
CREATE USER wadamson WITH PASSWORD 'wayne123';
GRANT horrorRatingAdmin TO wadamson;

b)
wadamson:
GRANT SELECT ON ratingStats TO sking WITH GRANT OPTION;
```

7. **(4 boda)**

a) Objasnite riječima što je rezultat sljedećeg izraza relacijske algebre:

$$\pi_{genreid,genrename}(\sigma_{prevMovieId\ IS\ NOT\ NULL\ '}\ (movie \bowtie track \bowtie trackgenre \bowtie genre)) \setminus \pi_{genreid,genrename}\ (\sigma_{duration > '02:30:00'\ \wedge\ releaseDate > '01.01.2015.'}\ (movie \bowtie track \bowtie trackgenre \bowtie genre))$$

Napomena:

- Rješenje napišite kao komentar u SQL upitu

b) Za zadani izraz iz a) djela zadatka napišite pripadni SQL upit.

Napomena:

- Zadatak riješite jednim SQL upitom

a) Svi žanrovi (identifikator i naziv žanra) kojima pripadaju filmovi - nastavci nekog prethodno snimljenog filma (filmovi s prethodnikom), s tim da u žanru nema nijednog filma s prethodnikom koji je dulji od 2 sata i trideset minuta koji je pri tom prvi put prikazan nakon 1.1. 2015.

```
b)
SELECT DISTINCT genreId, genrename
FROM movie NATURAL JOIN track NATURAL JOIN trackgenre NATURAL JOIN genre
WHERE prevmovieid IS NOT NULL
EXCEPT
SELECT DISTINCT genreId, genrename
FROM movie NATURAL JOIN track NATURAL JOIN trackgenre NATURAL JOIN genre
WHERE duration > '02:30:00'::INTERVAL AND releasedate > '01.01.2015'::DATE
```

8. **(5 bodova)**

Navedite po jedan primjer za a) odnosno b) dio zadatka, koji koristi isključivo podatke iz tablice ***genre,*** kojima ćete demonstrirati da se u jednoj od transakcija pojavljuje problem:

a) **neponovljivo čitanje**
b) **sablasne n-torke**

Primjere izraditi uz pretpostavku korištenja Postgres-a i njegove implementacije protokola upravljanja istodobnim pristupom – MVCC.

Svaki od dva primjera treba sadržavati minimalan broj transakcija koje se paralelno odvijaju i minimalan broj SQL naredbi potrebnih za demonstraciju traženog. Naznačite:

- razine izolacije koje u primjerima koristite
- sadržaj n-torke/n-torki temeljem kojih uočavaju se problemi pod a), odnosno b)

- redoslijed izvršavanja naredbi u transakcijama (globalni redoslijed za sve transakcije koje u primjeru koristite)
- u kojoj od transakcija se pojavljuje traženi problem, čime je uzrokovan i kako se to vidi.

NAPOMENA: svoje odgovore unesite u okvir za slobodni unos teksta ispod teksta zadatka.

Obje transakcije koriste defaultnu razinu izolacije – READ COMMITTED, ali može se koristiti i READ UNCOMMITTED.

a)

	T1		T2
1	`BEGIN TRANSACTION;`	2	`BEGIN TRANSACTION;`
3	`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	4	`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`
5	`SELECT * FROM genre WHERE genreId = 4;`	6	`UPDATE genre SET genreName = 'Spy thriller' WHERE genreId = 4;`
		7	`COMMIT TRANSACTION;`
8	`SELECT * FROM genre WHERE genreId = 4;`		
9	`COMMIT TRANSACTION;`		

Problem neponovljivog čitanja se pojavljuje u transakciji T1, a sastoji se u tome da ponovljenim čitanjem istog objekta (Select naredba pod rbr 5 i 8) T1 pročita istu n-torku ali je njen sadržaj drugačiji u ponovljenom čitanju. Neponovljivo čitanje je posljedica potvrđene izmjene koju je obavila transakcija T2 na naredbom 6 i potvrdila naredbom 7 i to nakon početka transakcije T1.

{5} Početni sadržaj n-torke koja se koristi u primjeru:

genreId	genreName
4	Thrillers

{8} Sadržaj n-torke nakon potvrđene (COMMIT) promjene od strane transakcije T2:

genreId	genreName
4	Spy thriller

b)

	T1		T2
1	`BEGIN TRANSACTION;`	2	`BEGIN TRANSACTION;`
3	`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	4	`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`
5	`SELECT * FROM genre WHERE genreName LIKE '%hriller%';`	6	`INSERT INTO genre VALUES (2, 'Spy thriller');`
		7	`COMMIT TRANSACTION;`
8	`SELECT * FROM genre WHERE genreName LIKE '%hriller%';`		
9	`COMMIT TRANSACTION;`		

Problem sablasnih n-torki se pojavljuje u transakciji T1, a sastoji se u tome da ponovljenim čitanjem istog objekta (Select naredba pod rbr 5 i 8) T1 čita isto i očekuje isto, ali dobije 1 n-torku više. Sablasne n-torke su posljedica potvrđenog unosa (INSERT) kojeg je obavila transakcija T2 na naredbom 6 i potvrdila naredbom 7 i to nakon početka transakcije T1.

{5} Početni sadržaj n-torke koja se koristi u primjeru:

genreId	genreName
4	Thrillers

{8} Sadržaj n-torke nakon potvrđene (COMMIT) promjene od strane transakcije T2:

genreId	genreName
2	Spy thriller
4	Thrillers

9. (5 bodova)

Za rješavanje sljedećeg zadatka koristite stvarne statističke podatke koji odgovaraju *StreamFlix* bazi podataka dostupnoj na Edgaru koja se koristi u ovoj provjeri. U spomenutoj bazi podataka definirana su ograničenja primarnih i stranih ključeva.

Izvodi se upit:

```
SELECT *
  FROM movie, track, subtitle
WHERE track.trackId = movie.trackId
   AND track.trackId = subtitle.trackId
  AND track.trackId < 2000
   AND trackRating < 40
  AND boxIncome < 100000000;
```

1. Nacrtajte (na papir) stablo upita za **početni plan** izvođenja upita pri čemu je redoslijed spajanja tablica određen redoslijedom kojim su tablice navedene u FROM dijelu SELECT naredbe.
2. Provedite heurističku optimizaciju. Redoslijed spajanja tablica odredite temeljem procjene broja n-torki u međurezultatima.

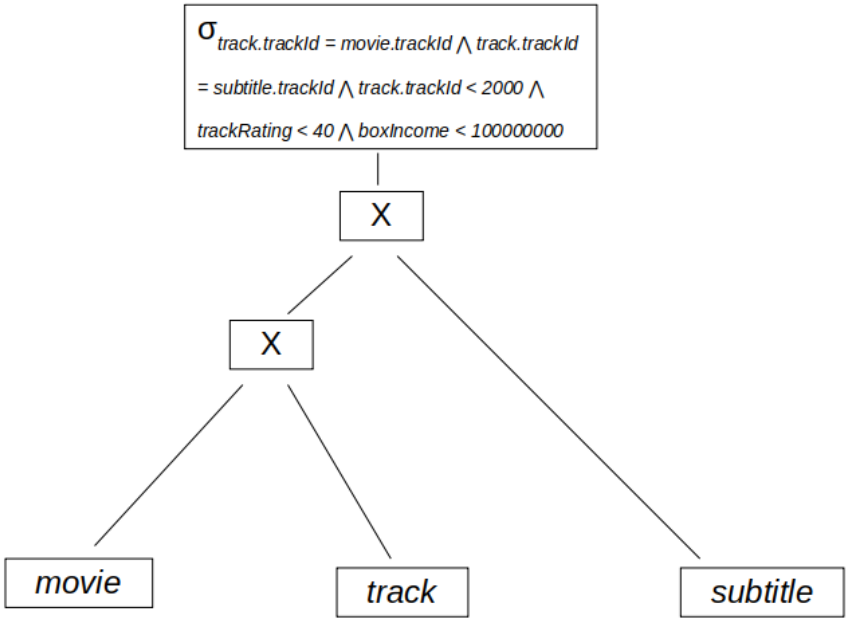
   Navedite sve statističke izračune i izraze prema kojima je obavljena procjena broja n-torki u međurezultatima.

   Ako postoji više jednakovrijednih najboljih redoslijeda spajanja, odlučite se za jedan.

   Ako prilikom izračuna broja n-torki dobijete decimalnu vrijednost, zaokružite rezultat na prvi veći cijeli broj.

   Izračunate statističke podatke, postupak i izračun za procjenu broja n-torki **unesite u prostor za slobodni unos teksta ispod teksta zadatka**.
3. Nacrtajte (na papir) **konačno** stablo upita. U stablu upita naznačiti očekivani broj n-torki za svaku operaciju.

1.



2.

N(track) = 27176	N(subtitle) = 47664	N(movie) = 5040

Selekciju trackId < 2000 treba potisnuti na **subtitle**, **track** i **movie**

- `subtitle.trackId < 2000 -> N(subtitle') = N(subtitle)/3 = 15888`

Potiskivanje selekcije trackId < 2000 i trackRating < 40 na track

- `track.trackId < 2000 AND track.trackRating < 40 -> N(track') = N(track)/(3*3) = 27176/9 ≈ 3020`

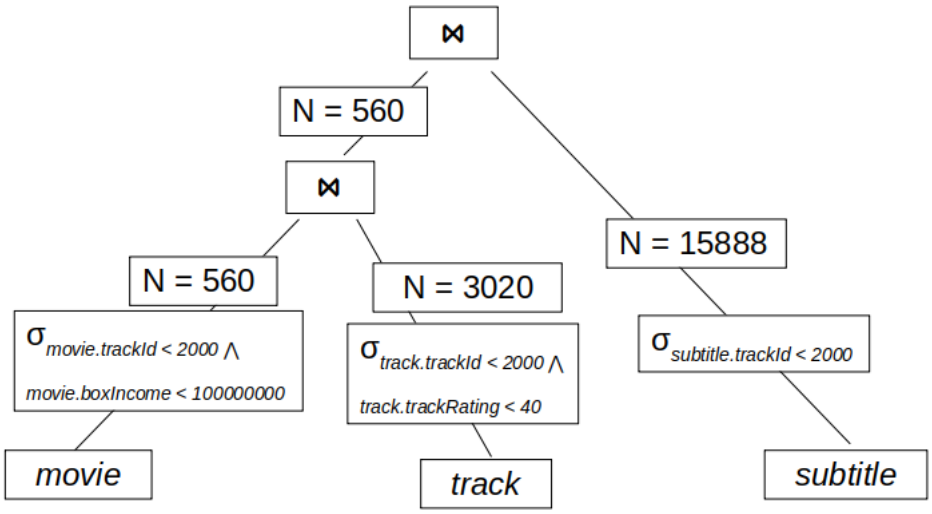
Potiskivanje selekcije trackId < 2000 i boxIncome < 100000000 na movie

- `movie.trackId < 2000 AND movie.boxIncome < 100000000 -> N(movie') = N(movie)/(3*3) = 5040/9 = 560`

Međurezultati spajanja:

- `N(track' ⊳⊲ subtitle') = (track.trackId je strani ključ u subtitle) = 15888`
- **`N(track' ⊳⊲ movie') = (track.trackId je strani ključ u movie) = 560`**
- `N(subtitle' ⊳⊲ movie') = (movie.trackId je strani ključ u subtitle) = 15888`

3.



10. (8 bodova)

Potrebno je oblikovati ER model baze podataka za evidentiranje podataka o igračima i njihovoj igri na Europskom nogometnom prvenstvu.

Evidentiraju se podatci o reprezentativcima koji sudjeluju na Europskom prvenstvu bilo kao igrači, bilo kao članovi stručnog tima.

Za reprezentativce se bilježi jedinstveni identifikator, ime i prezime. Za reprezentativca koji je igrač bilježi se dodatno i datum rođenja te pozicija na kojoj igrač igra. Pozicija je opisana jedinstvenim identifikatorom i nazivom pozicije. Jedan igrač može igrati na više pozicija.

Za reprezentativce koji su članovi stručnog tima dodatno se bilježi uloga. Uloga je opisana jedinstvenim identifikatorom i nazivom (trener, liječnik, fizioterapeut…). Svaki član stručnog tima ima točno jednu ulogu, a istu ulogu može obavljati više članova stručnog tima.

Reprezentativci čine reprezentaciju koja predstavlja određenu državu na europskom prvenstvu. Reprezentacija je određena ISO oznakom države koju predstavlja i godinom prvenstva, a za nju se dodatno bilježi i naziv te trenutna pozicija na FIFA rang listi. Država je opisana ISO oznakom i nazivom države koji su jedinstveni. Reprezentacije sudjeluju u utakmicama, jedna kao reprezentacija *domaćin*, a druga kao reprezentacija *gost* (*napomena*: iako se definicija *reprezentacije domaćina* i *reprezentacije gosta* ne može primijeniti na svaku utakmicu europskog prvenstva, pretpostavite da ipak može). Svaka utakmica je određena jedinstvenim identifikatorom, a dodatno se bilježi datum i vrijeme održavanja.

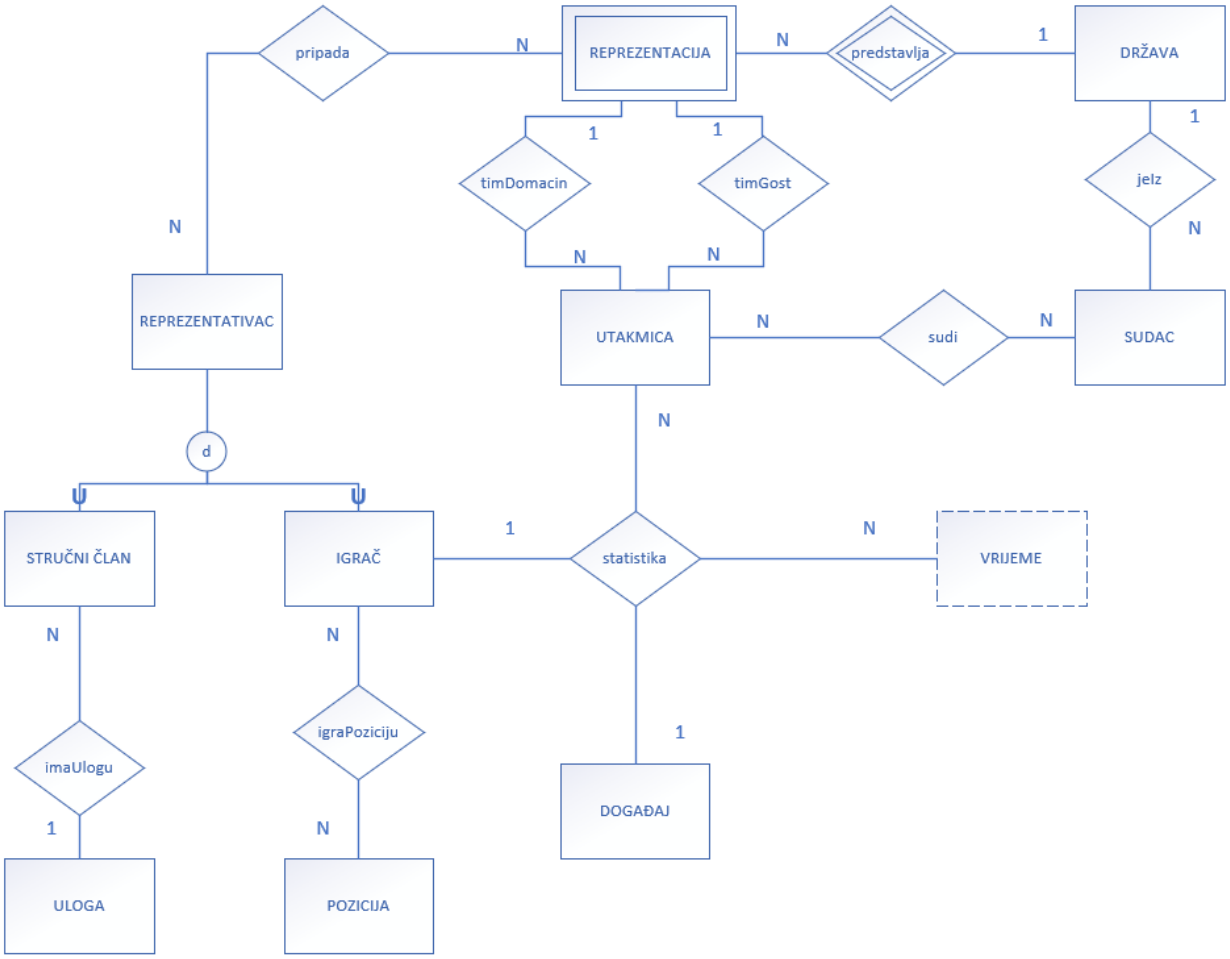
Za svaku utakmicu bilježe se i sudci koji su sude na toj utakmici (na jednoj utakmici može suditi više sudaca). Za sudca se bilježi jedinstveni identifikator, ime i prezime te država iz koje sudac dolazi.

Za svakog igrača se na svakoj utakmici bilježe događaji (asistencije, udarci u okvir gola, pogreške…). Svaki događaj opisan je jedinstvenim identifikatorom te nazivom. Za svaki događaj igrača na utakmici se također bilježi i vrijeme (**minuta i sekunda**) utakmice u kojoj se dogodio taj događaj (npr. u 56. min i 3. sek

utakmice igrač puca penal i zabija gol) te binarna oznaka koja opisuje je li događaj rezultirao pogotkom. U jednom trenutku tijekom jedne utakmice neki događaj može biti realiziran od strane samo jednog igrača. Neki igrač može isti događaj ponoviti više puta tijekom utakmice. U jednom trenutku na jednoj utakmici jedan igrač može sudjelovati samo u jednom događaju. Isti igrač, isti događaj može ponoviti u istom vremenu (minuta i sekunda utakmice) u više različitih utakmica.

a) Nacrtajte ER model baze podataka **na priloženom papiru**. **Nemojte crtati atribute**.

b) U **prostor za slobodni unos teksta** ispod teksta zadatka navedite **sheme entiteta i sheme veza** (označite primarne i alternativne ključeve). Svaki entitet (osim slabih entiteta) opisati **isključivo vlastitim atributima**. Nužno je da sve sheme zadovoljavaju 3NF.



DRŽAVA = **<u>ISO</u>**, nazivDržava

REPREZENTACIJA = **<u>ISO, godina,</u>** rang, nazivReprezentacija

REPREZENTATIVAC = **<u>idRep</u>,** imeRep, prezimeRep

IGRAČ = **<u>idRep</u>,** datumRodjenja

K1 $_{\text{DRŽAVA}}$ = ISO; K2 $_{\text{DRŽAVA}}$ = nazivDržava

K $_{\text{REPREZENTACIJA}}$ = ISO, godina

K $_{\text{REPREZENTATIVAC}}$ = idRep

K $_{\text{IGRAČ}}$ = idRep

POZICIJA = **idPozicija**, nazivPozicija

STRUČAN_ČLAN = **idRep**

ULOGA = **idUloga**, nazivUloga

UTAKMICA = **idUtakmica**, datumVrijemePoc

SUDAC = **idSudac**, imeSudac, prezimeSudac

DOGAĐAJ = **idDogađaj**, nazivDogađaj

---

predstavlja = **ISO, godina**

pripada = **ISO, godina, idRep**

timDomacin = **idUtakmica,** ISO, godina

timGost = **idUtakmica,** ISO, godina

sudi = **idSudac, idUtakmica**

statistika = **idUtakmica, vrijeme, idDogađaj**, idRep, pogodakBin

imaUlogu = **idRep**, idUloga

igraPoziciju = **idRep, idPozicija**

jeIz = **idSudac,** ISO

K $_{POZICIJA}$ = idPozicija

K $_{STRUČAN\_ČLAN}$ = idRep

K $_{ULOGA}$ = idUloga

K $_{UTAKMICA}$ = idUtakmica

K $_{SUDAC}$ = idSudac

K $_{DOGAĐAJ}$ = idDogađaj

K $_{predstavlja}$ = ISO, godina

K $_{pripada}$ = ISO, godina, idRep

K $_{timDomacin}$ = idUtakmica

K $_{timGost}$ = idUtakmica

K $_{sudi}$ = idSudac, IdUtakmica

K $1_{statistika}$ = idUtakmica, vrijeme, idDogađaj

K $2_{statistika}$ = idUtakmica, vrijeme, idRep

K $_{imaUlogut}$ = idRep

K $_{igraPoziciju}$ = idRep, idPozicija

K $_{jeIzt}$ = idSudac

Kriterij:
Ternarne veze s degeneriranim entitetom – 2 boda
Paralelna veza **REPREZENTACIJA** – **UTAKMICA** – 1 bod
Gen/spec 1 bod
Slabi entiteti i veze 1 bod
Ostali entiteti i veze 1 bod
Opis entiteta i veza 1 bod
Alternativni ključevi kod opisa entiteta: 0.5 bod za alternativni ključ kod veze **statitsika**
0.5 bod kod alternativni ključ kod **DRŽAVA**