Sveučilište Josipa Jurija Strossmayera u Osijeku,

Odjel za matematiku,

Preddiplomski studij Matematika i računarstvo

# Moderni sustavi baza podataka

# Playstation online service

Profesor:

doc.dr.sc. Slobodan Jelić

Student:

Krunoslav Vrkljan

Asistenti:

Ena Pribisalić

Mateja Đumić

Osijek,srpanj 2020.

# 1.Uvod:

Playstation online service je sustav koji povezuje sve korisnike playstationa te koji omogućuje online kupovanje igara te pretplate na streaming siteove. Preko toga se mogu organizirati zajednice koje povezuju ljude sa sličnim interesima te pravljenje događaja poput turnira i zajedničkog igranja igrica. Sustav omogućuje spremanje kartice radi lakšeg i bržeg nacina plaćanja.

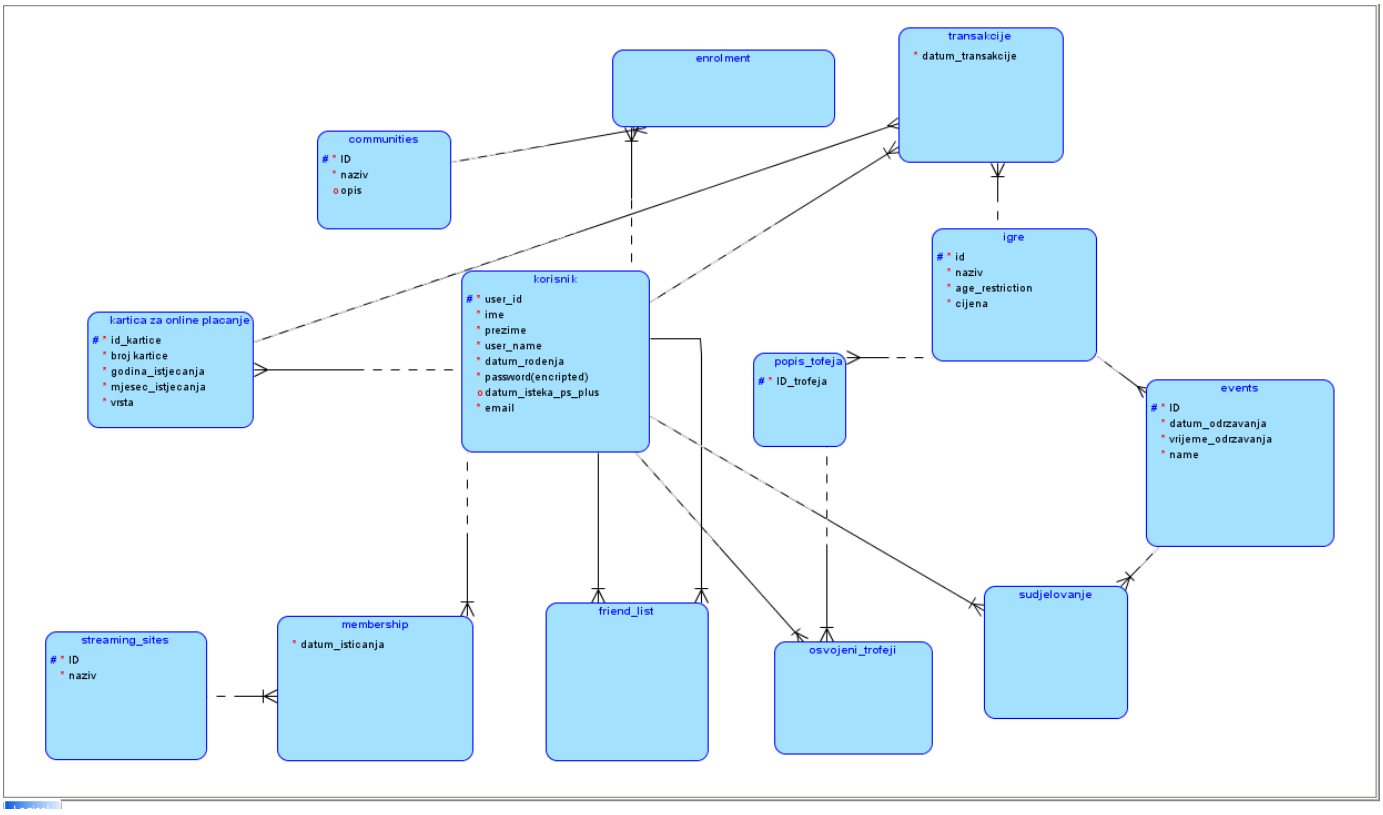
Cilj mog projekta bio je što bolje prikazati bazu podataka za playstationov online sistem.

## 2.MEV i Relacijski model:

Pri izradi baze prije pisanja sql koda bitno je napraviti MEV te iz njega relacijski model za bazu. Pomoću MEV-a možemo jednostavno vizualno prikazati entitete tablice te veze između njih. Relacijski model je koristan jer sadrži koncept relacija između tablica.On podržava programski jezik SQL te tako iz njega možemo napraviti svoju bazu.

Za dizajniranje MEV-a i relacijskog modela koristio sam Oracle SQL Data Developer.

### 2a) MEV Playstation online usluge



Model baze za playstation online sustav ima 13 entiteta. Entiteti su povezani vezama . Glavni entitet u bazi je korisnik(user) te veze iz njega govore sljedeće:

-Svaka katica mora imati svog korisnika , ali svaki korisnik ne mora imati karticu

-korisnik može biti dio communityja , ali i ne mora te svaki community može imati korisnike koji su dio tog communityja , ali I ne mora

-transakcija je određena korisnikom koji je kupio igru, igrom koju je kupio te karticom koju je upotrijebio

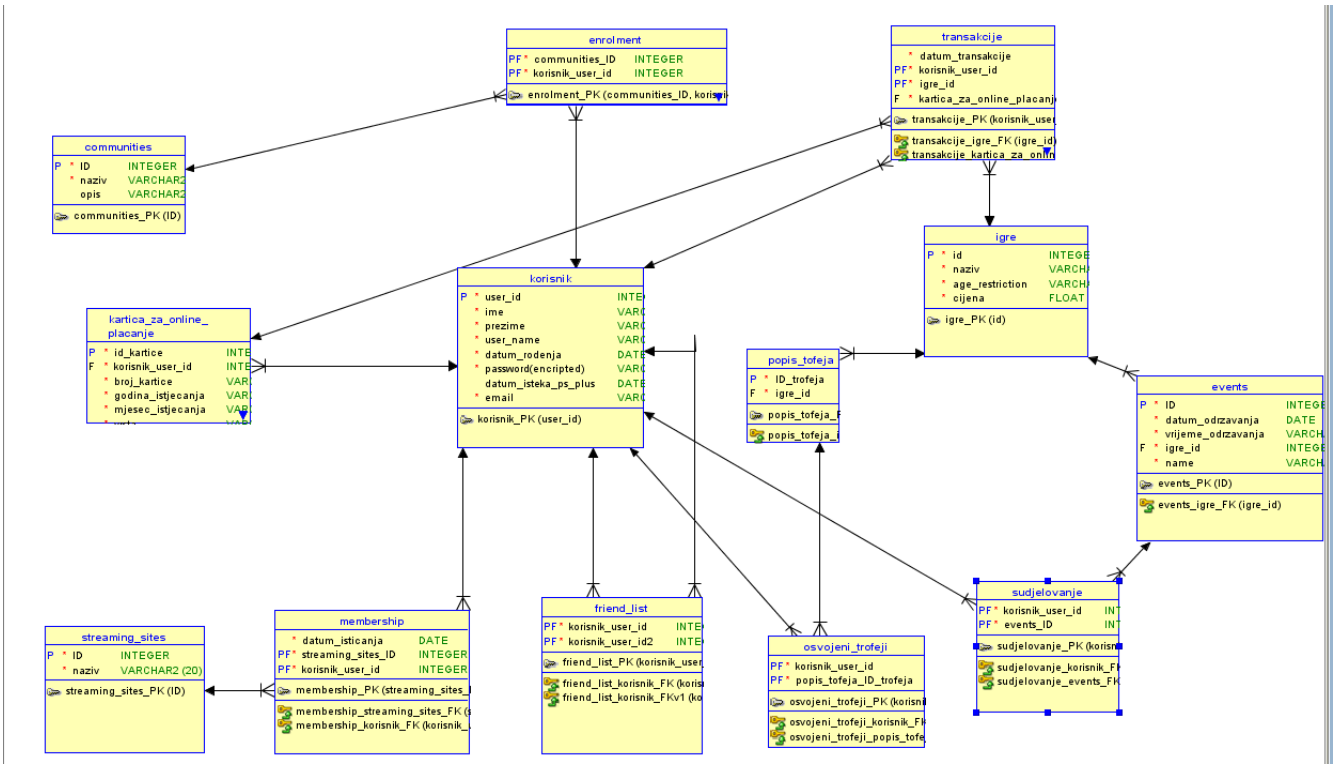
-veza sa friendlistom je zapis dva korisnika koji su ostvarili prijateljstvo

-Praćenje pretplate na streaming siteove određeno je korisnikom I streaming siteom na koji je pretplaćen

-Osvajanje trofeja je određeno korisnikom i popisom trofeja koji se još veže na igru kojoj trofej pripada

-Sudjelovanje u događajima je određeno korisnicima koji sudjeluju u događaju i samim događajem koji se još veže sa igrom u kojoj se sam događaj održava

## 2b) Relacijski model  Playstation online usluge




Relacijski model koji smo dobili iz prethodnog mev-a omogućuje nam pravljenje sql baze. Bit će opisani entiteti transakcije ,kartice za online plaćanje ,friend_list.Ostali se opisuju analogno.

Transakcije se sastoje od tri strana ključa (korisnika,igre i kartice) koji su integeri te primarni kljuc transakcije čine u paru strani ključevi igre i korisnika te se sastoji od atributa datum_transakcije koji je datum te su svi atributi obvezni.

Kartice za online plaćanje sastoje se od generiranog primarnog ključa koji je integer te stranog ključa korisnika koji je također integer koji su obvezni te od obveznih atributa broj kartice koji je integer, godina isticanja koji je varchar,mjesec isticanja koji je varchar te vrsta kartice koji je varchar.

Friend lista se sastoji od dva strana ključa koji su dva id-a korisnika iz tablice korisnik te su u paru primarni ključ.

## 3.Kreiranje tablice:

Kreiranje tablica služi za pravljenje entiteta u bazi podataka.

Sinstaksa za kreiranje tablica:

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

Brisanje tablica:

```
Drop table table_name;
```

Mogući tipovi podataka:INT,VARCHAR,NUMBER,BOOL,DATE…

Primjeri:

```
CREATE TABLE users (
    user_id INT NOT NULL ,
    FirstName VARCHAR(15) NOT NULL,
    LastName VARCHAR(15) NOT NULL,
    user_name VARCHAR(30) NOT NULL,
    date_of_birth_day DATE NOT NULL,
    password VARCHAR(30) NOT NULL,
    datum_isteka_ps_plus DATE,
    email VARCHAR(45) NOT NULL  ,
    CONSTRAINT PK_users PRIMARY KEY (user_id),
    CONSTRAINT UC_email UNIQUE(email)
);
```

```
CREATE TABLE events(
    event_id int NOT NULL PRIMARY KEY ,
    date_of_event DATE NOT NULL,
    time_of_event VARCHAR(5) NOT NULL,
    game_id int NOT NULL CONSTRAINT fk_game_id_e REFERENCES games(game_id),
    name VARCHAR (50) NOT NULL
);
```

```
CREATE TABLE transaction (
    date_of_transaction DATE NOT NULL,
    user_id int NOT NULL CONSTRAINT fk_user_id_tr REFERENCES users(user_id),
    game_id int NOT NULL CONSTRAINT fk_game_id_tr REFERENCES games(game_id),
    card_id int NOT NULL CONSTRAINT fk_card_id REFERENCES bank_cards(card_id),
    CONSTRAINT PK_transaction PRIMARY KEY (user_id,game_id)
);
```

## 4.Unosi u tablicu:

Unosi služe za punjenje baze sa podatcima.

Sintaksa za unos u tablicu:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

ili

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

Pri unosu podataka koristio sam insert all naredbu.

Primjer:

```
insert all
into streaming_sites (stream_site_id,name) VALUES (1,'HBO GO')
into streaming_sites (stream_site_id,name) VALUES (2,'Netflix')
into streaming_sites (stream_site_id,name) VALUES (3,'Prim Video')
into streaming_sites (stream_site_id,name) VALUES (4,'Tennis TV')
SELECT * FROM dual;
COMMIT;
```

## 5.Upiti:

Upiti nad tablicom služe za dohvaćanje informacija iz baze. Postoje jednostavni i složeni upiti nad tablicom. Jednostavni upiti su upiti nad jednom tablicom dok su složeni upiti upiti nad više od jedne tablice.

Sintaksa za upite:

```
SELECT column1, column2, ...
FROM table_name;
```

Primjeri:

```
SELECT uk.user_name FROM
(SELECT u.user_name,u.email, COUNT(g.name) as broj_igara FROM
games g INNER JOIN trophies t USING (game_id)
INNER JOIN acquired_trophies at USING (trophy_id)
INNER JOIN users u USING (user_id)
GROUP BY u.email,u.user_name) uk WHERE
uk.broj_igara>1;
```

Priloženi upit pokazuje korisnike koji imaju trofeje iz više od jedne igre.

Unutar upita napravljen je pod upit iz kojega glavni upit uzima informacije. U podupitu pomoću ključne riječi count brojimo broj igara po korisniku pošto je group by email i user_name.Tako je napravljen group by jer email mora bit UNIQUE dok user_name ne mora , a želimo izlistati user_nameove koji imaju osvojen trofej u više od jedne igre. Tablice se povezane sa INNER JOIN naredbom koji preko stranog ključa povezuje dvije tablice

```
SELECT user_name FROM users
WHERE datum_isteka_ps_plus > CURRENT_DATE
ORDER BY user_name;
```

Priloženi upit propitkuje koji sve korisnici imaju aktivnu ps_plus pretplatu. Upit je jednostavan te korištenjem ključne riječi WHERE postavljamo uvjet koji provjerava je li zapisan datum isteka pretplate veći od trenutnog dana.

## 6.Procedure:

Procedure su pripremljen sql kod koji se može izvršavati više puta. Proceduri možemo proslijediti vrijednosti tako da nam sql kod može ovisiti o vrijednosti koju smo proslijedili.

Procedure su pisane u jednostavnom proceduralnom jeziku te na njih možemo gledati kao funkcije ili metode.

Prednosti procedura:

- Moguće mnogostruko korištenje
- Optimizacija aplikacije

Nedostatak:

- Spremanje procedure može zauzimati puno memorije

Sintaksa procedure:

```
CREATE [OR REPLACE] PROCEDURE procedure_name
[(parameter_name [IN | OUT | IN OUT]
type [, ...])]{IS | AS}
BEGIN
procedure_body
END procedure_name;
/
```

Poziv procedure:

```
Call procedure_name(parametar1,…) ;
```

Brisanje procedure:

```
DROP PROCEDURE procedure_name;
```

Primjer:

```
CREATE OR REPLACE PROCEDURE InsertNewUser(
    p_FirstName IN VARCHAR,
    p_LastName IN VARCHAR,
    p_user_name IN VARCHAR ,
    p_date_of_birth_day IN DATE ,
    p_password IN VARCHAR,
    p_email IN VARCHAR)
    AS
        BEGIN
            INSERT INTO users (user_id,FirstName,LastName,user_name,date_of_birth_day,password,datum_isteka_ps_plus,email)
            VALUES (auto_increment.NEXTVAL,p_FirstName,p_LastName,p_user_name,p_date_of_birth_day,p_password,NULL,p_email);
            COMMIT;
        EXCEPTION
            WHEN OTHERS THEN
                ROLLBACK;
        END InsertNewUser;
/
```

Priložena procedura prihvaća vrijednosti p_FirstName ,p_LastName, p_user_name, p_date_of_birth_day, p_password, p_email te ih inserta u tablicu users s tim da za id je napravljen SEQUENCE te je datum_isteka_ps_plus uvijek na početku NULL. Naredbom COMMIT potvrđujemo unos novog korisnika u tablicu. Ako se dogodi neka greška pri izvršavanju procedure procedura ulazi u EXCEPTION blok te pomoću ROLLBACK naredbe poništava sve promjene.

```
CREATE OR REPLACE PROCEDURE ReNewPsPlus
(u_id IN INT)AS
BEGIN
    UPDATE users  SET datum_isteka_ps_plus = CURRENT_DATE + 30 WHERE user_id = u_id ;
    COMMIT;
END ReNewPsPlus;
/
```

Priložena procedura obnavlja pretplatu ps_plusa te joj stavlja datum isticanja 30 dana nakon što je obnovljen.Procedura prima id od korisnika kojem se produljuje pretplata.

Sa naredbom commit potvrđujemo navedene promjene.

```
CREATE OR REPLACE PROCEDURE FriendList(
u_user_id IN INT) AS
curs SYS_REFCURSOR;
u_user_pom VARCHAR(40);
u_ukupno INT;
BEGIN
    SELECT COUNT (distinct uk.user_id_1) + COUNT (distinct ukp.user_id_2) into u_ukupno  FROM
    (SELECT user_id_1 FROM
    friend_list
    where user_id_2 = u_user_id) uk ,
    (SELECT user_id_2 FROM
    friend_list
    where user_id_1 = u_user_id) ukp;

    OPEN curs FOR
        SELECT u.user_name FROM
        users u INNER JOIN friend_list f ON f.user_id_1 = u.user_id
        where user_id_2 = u_user_id
        UNION
        SELECT u.user_name FROM
        users u INNER JOIN friend_list f ON f.user_id_2 = u.user_id
        where user_id_1 = u_user_id;
        FOR i IN  1..u_ukupno LOOP
            FETCH curs INTO u_user_pom;
        DBMS_OUTPUT.PUT_LINE('Prijatelj:' || u_user_pom);
    END LOOP;

END FriendList;
/
```

Priložena procedura izlistava sve prijatelje određenog korisnika.Procedura prima ID korisnika kojemu treba izlistati listu prijatelja. Na početku navodimo pomoćne varijable. Prvi upit nam govori koliko prijatelja ima određeni korisnik  da bi u idućem upitu pomoću kursora došli do svakog korisnika koji je friendan sa korisnikom te ih sve ispisati.

## 7.Okidači:

Okidači su vrste procedura koji se pokreću nakon naredbi INSERT,DELETE ili UPDATE

Dijelimo ih prema vremenu pokretanja(before/after) i prema načinu djelovanja (row-level/ statement level).

Sintaksa:

CREATE [OR REPLACE] TRIGGER trigger_name

{BEFORE | AFTER | INSTEAD OF | FOR} trigger_event

ON table_name

[FOR EACH ROW]

[{FORWARD | REVERSE} CROSSEDITION]

[{FOLLOWS | PRECEDES} schema.other_trigger}

[{ENABLE | DISABLE}]

[WHEN trigger_condition]]

BEGIN

trigger_body

END trigger_name;

Primjer:

```
CREATE OR REPLACE TRIGGER RenewedMembership
AFTER UPDATE OF  datum_isteka_ps_plus ON users
BEGIN
    DBMS_OUTPUT.PUT_LINE('Your mebership has been renewed');
END;
/
```

Priloženi okidač RenewedMebership je okidač na razini tablice koji se okida nakon nadoplate ps_plus članstva.

```
CREATE OR REPLACE TRIGGER Popust_igre
BEFORE UPDATE of cijena ON games
FOR EACH ROW WHEN (NEW.cijena< OLD.cijena * 0.75)
BEGIN
    DBMS_OUTPUT.PUT_LINE('Smanjenje cijena za vise od 25%');
END;
/
```

Priloženi okidač Popust_igre je okidač na razini retka koji javlja korisniku ako je bilo koja igra smanjena za 25 % ili više.

## 8.Indeksi:

Indeksi nam služe za ubrzavanje pretraživanja tablica. Prigodni su za stupce koji imaju velik broj različitih vrijednosti. Njihov način pretraživanja temelji se na strukturi podataka koji nazivamo B-stablo.

Sintaksa:

CREATE [UNIQUE] INDEX index_name ON

table_name(column_name[, column_name ...])

TABLESPACE tab_space; Brisanje indeksa:

```
DROP INDEX index_name;
```

Primjer:

```
CREATE INDEX PretragaTrofeja ON trophies(game_id);
```

Priloženi indeks indeksira trofeje po igrama te ubrzava pretragu trofeja.

## 9.)Zaključak:

Cilj ovog projekta bio je samostalno sa stečenim znanjima i vještinama napraviti bazu podataka , u ovom slučaju za playstation online sistem te usavršiti svoje SQL znanje.

Cjelokupan dojam rada projekta je bio pozitivan zbog samostalnog rad na bazi te samostalno rješavanje problema.