Sveučilište Josipa Jurja Strossmayera

u Osijeku

Odjel za Matematiku

Moderni sustavi baza podataka

Baza podataka za Lovačke udruge

Seminarski rad

Ime i Prezime: Hrvoje Veber

Mentor: Slobodan Jelić

Sadržaj

Uvod

Cilj ovog seminarskog rada je pobliže opisati i objasniti funkciju projekta „baza lovačkih udruga“ iz Modernih sustava baza podataka.

Baza sadrži sve potrebne podatke o lovačkim udrugama. Pohranjujemo podatke o svim divljim životinjama koje su propisane za lov u Republici Hrvatskoj.

Svaka lovačka udruga ima svoj plan gospodarenja, jer nisu sve životinje jednako rasprostranjene, pa će tako neke životinje biti propisane za lov u jednoj lovačkoj udruzi, ali ne nužnu i u nekoj drugoj.

Za svaku divlju životinju ima propisani datum od kada do kada se smije loviti, što nam je potrebno za izdavanje dozvola članovima lovačkih udruga. Također svaka lovačka udruga mora imati plan gospodarenja, odnosno broj divljači u lovištu koji se smije odstrijeliti.

Pohranjujemo podatke o članovima, trenutno aktivnima i neaktivnima, što znači da čuvamo podatke i o onima koji su bili upisani pa su se ispisali. Svaki član koji ide u lov mora izvaditi mjesečnu dozvolu na kojoj piše koju divljač smije loviti i koliko smije ustrijeliti.

Također članovi plaćaju članarinu koja se određuje prema njihovom statusu. Lovačka udruga određuje koliko će iznositi članarina za određeni status.
Svake godine organiziraju se skupni lovovi (po potrebi), gdje sudjeluju članovi lovačke udruge. U skupnom lovu je također određeno koja se divljač smije loviti i koliko smije biti odstrijeljeno.

Lovačka udruga mora imati lovište na kojem će loviti divljač. Svako lovište pripisano je nekoj udruzi i više udruga ne smije dijeliti isto lovište.

Moguće je postaviti lovno tehničke objekte na lovište. Svaki objekt mora imati svoj broj kojim će se identificirati. Pod lovno tehničke objekte spadaju npr. čeka, hranilište, solana, itd.

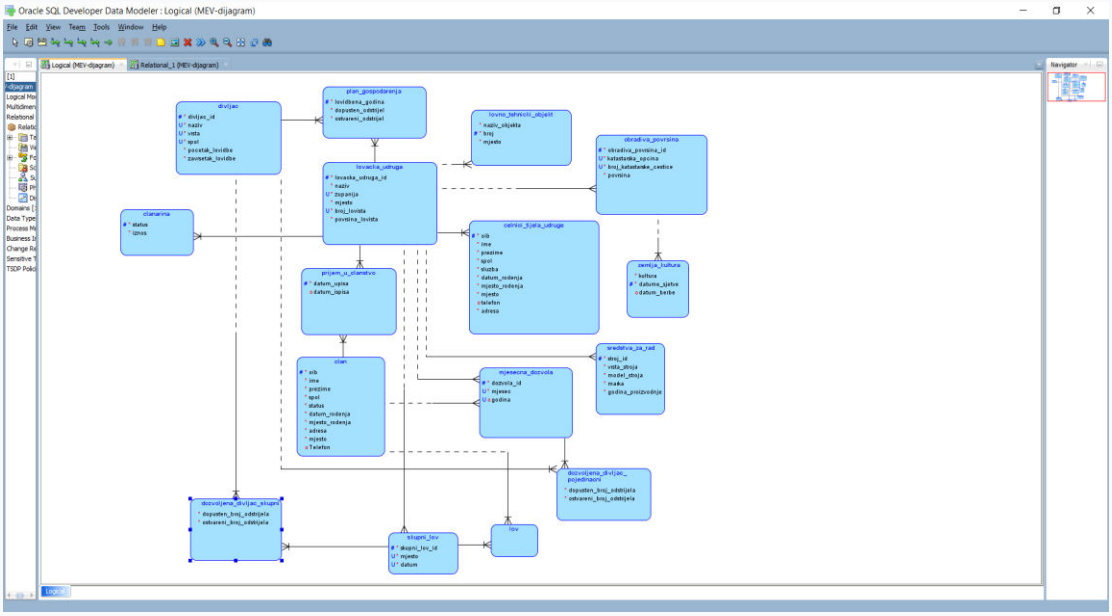
Lovačka udruga može posjedovati i obradive površine. Članovi obrađuju tu zemlju i plodove koriste za hranjenje divljači.

Uz obradivu površinu također su potrebna i sredstva za rad. Pa tako lovačka udruga može posjedovati i sredstva za obrađivanje zemlje.

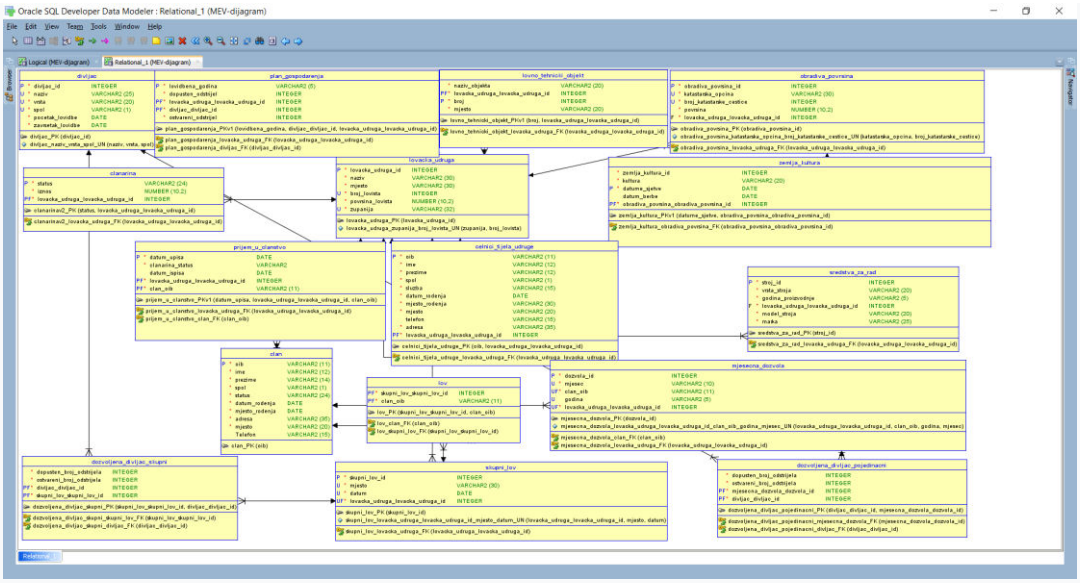
U nastavku ćemo objasniti i prikazati MEV i relacijski model, pa nakon toga sql skriptu.

1. MEV i relacijski model

U ovom poglavlju pobliže ćemo objasniti veze između entiteta i njihove primarne ključeve. Također komentirati ćemo koja je njihova funkcija u bazi, te ćemo priložiti relacijski model.



Slika 1.1 MEV dijagram



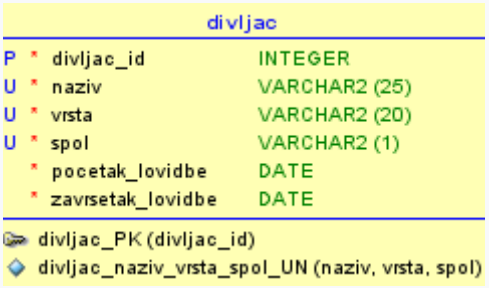
Slika 1.2 Relacijski model

U nastavku, sve veze između entiteta, koje budemo opisivali, odnositi će se na one prikazane na slici (Slika 1.1).

Slika 1.3 lovačka_udruga-relacijski model

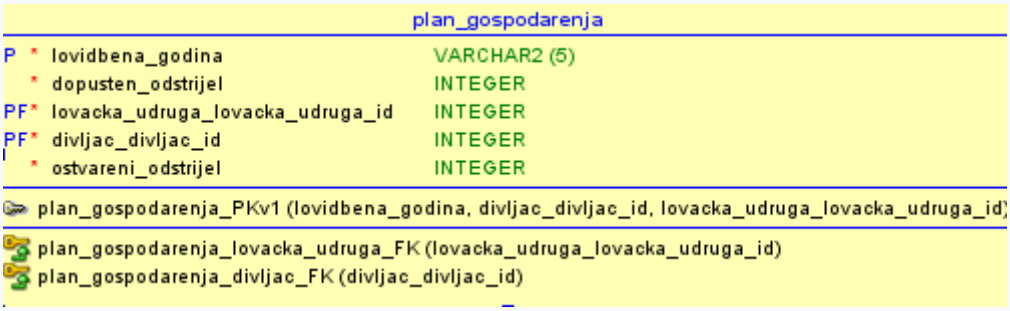
(Slika 1.1) prikazuje MEV dijagram baze podataka za lovačke udruge. Krenuti ćemo od entiteta lovačka udruga. Možemo vidjeti da taj entitet sadrži sve podatke o lovačkoj udruzi odnosno njen naziv, županiju, mjesto, broj lovišta te površinu lovišta. Primarni ključ je integer i vidimo da je tako moguće dodati dvije iste lovačke udruge sa različitim primarnim ključem, što ne smijemo dopustiti. Zato

postavljamo UNIQUE constraint koji će se pobrinuti da vrijednosti županija i broj lovišta čine jedan unikat u toj tablici.



Slika 1.4 divljač-relacijski model

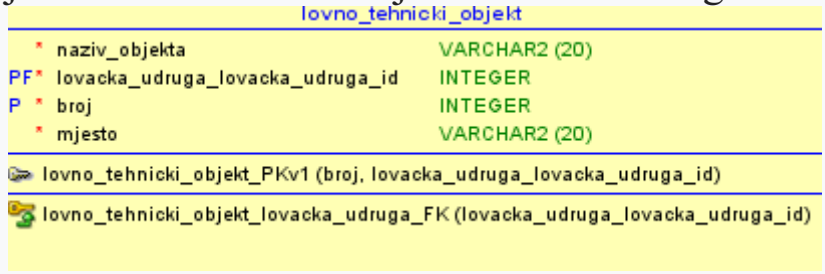
Zatim imamo entitet divljač koji sadrži naziv životinje, vrstu, spol te početka i završetak lovidbe. Za primarni ključ postavili smo integer kao i u lovačkoj udruzi, pa je opet moguće unijeti dvije iste životinje sa različitim primarnim ključem, što ne smijemo dopustiti, te opet postavljamo UNIQUE ključ koji se brine da naziv, vrsta i spol čine jedan unikat u tablici.



Slika 1.5 plan_gospodarenja-relacijski model

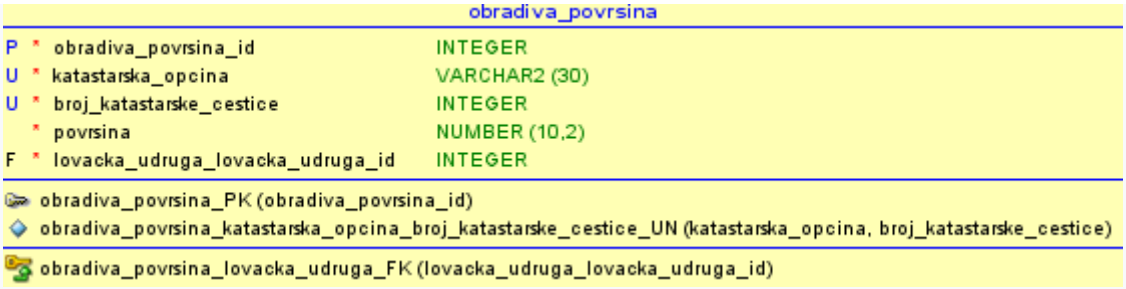
Sada spajamo lovačku udrugu i divljač sa entitetom plan gospodarenja. Obje veze su identifikacijske te 1:n. To znači da se

primarni ključevi iz entiteta lovačka udruga i divljač prenose te sudjeluju u stvaranju primarnog ključa u entitetu plan gospodarenja. Tako da za primarni ključ imamo divljač_id, lovačka_udruga_id te lovidbena_godina. Plan gospodarenja također sadrži i dopušten odstrjel te ostvareni odstrjel za svaku životinju u svakoj lovačkoj udruzi. Baš to je moguće zbog veza 1:n koje dopuštaju da u plan gospodarenja možemo unijeti više lovačkih udruga, sa više divljih životinja te više lovidbenih godina. Veze su obvezne, što znači da plan gospodarenja mora sadržavati divljač i lovačku udrugu.



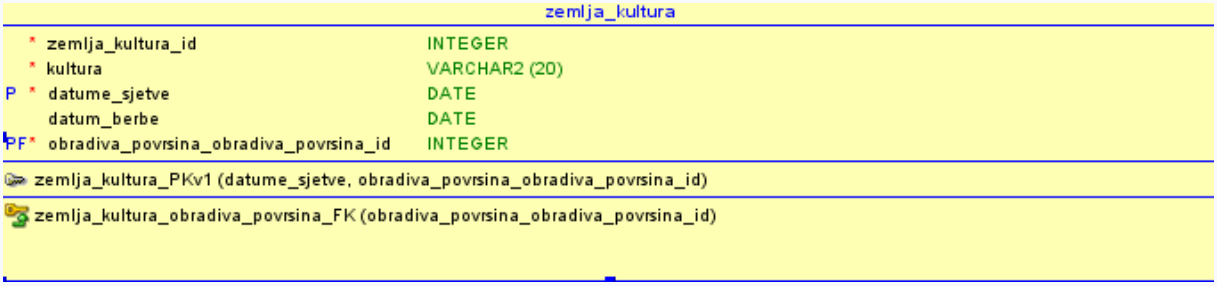
Slika 1.6 lovno_tehnicki_objekt-relacijski model

Entitet lovno tehnički objekti sadrži podatke o objektima koje posjeduje lovačka udruga. Svaki objekt ima svoj broj kojim ga označavamo, a kako više udruga može imati objekte i označiti ih brojem po želji, ovaj entitet moramo povezati sa lovačkom udrugom 1:n identifikacijskom vezom. Tako u primarnom ključu ovog entiteta, uz broj, sudjeluje i lovačka_udruga_id. Također veza je opcionalna, odnosno lovačka udruga može, a i ne mora imati lovno tehničke objekte.



Slika 1.7 obradiva_povrsina-relacijski model

Entitet obradiva_površina spajamo sa lovačkom udrugom, a sadrži sve obradive površine koje lovačka udruga posjeduje. Opet, veza je 1:n, opcionalna i nije identifikacijska. To znači da integer obrdiva_površina_id sam tvori primarni ključ. Zato postavljamo vrijednosti katastarska_općina i broj_katastarske čestice kao UNIQUE vrijednost. Tako zadržavamo jedinstvenost posjedovanja obradive površine, jer ne može više lovačkih udruga posjedovati istu obradivu površinu.



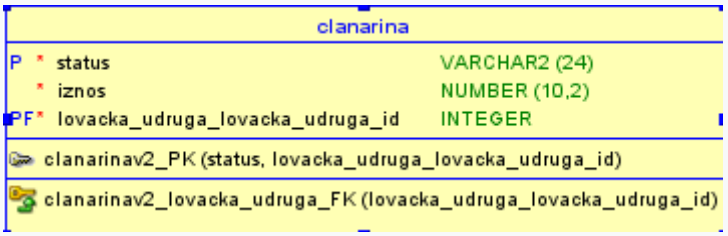
Slika 1.8 zemlja_kultura-relacijski model

Zatim entitet zemlja_kultura spajamo sa obradivom površinom. Veza je 1:n, opcionalna i identifikacijska. Ovaj entitet sadrži sve kulture koje su bile sijane na obradivoj površini. Primarni ključ čini

obradiva_površina_id i datum_sjetve. Također datum berbe nije obvezan.

Slika 1.9 sredstva_za_rad-relacijski model

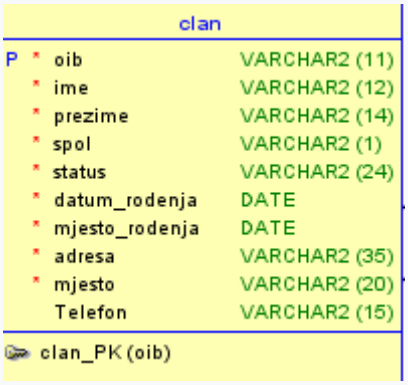
Da bismo obrađivali zemlju, moramo imati i strojeve za rad. Pa tako imamo entitet sredstva_za_rad. Ovaj entitet spajamo sa lovačkom udrugom 1:n, opcionalnom vezom. Veza nije identifikacijska. Primarni ključ čini vrijednost stroj_id. Lovačka udruga može imati više istih strojeva pa nije potrebno postavljati UNIQUE ključeve.



Slika 1.10 članarina-relacijski model

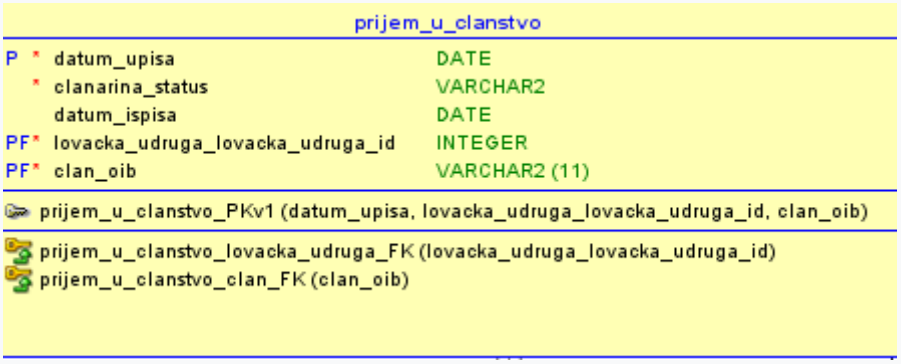
Svaka lovačka udruga ima članarinu, koju sama određuje. Iznos članarine ovisi o statusu člana. Ovaj entitet povezan je sa lovačkom udrugom 1:n, obveznom, identifikacijskom vezom. Tako da primarni

ključ se sastoji od vrijednosti lovačka_udruga_id i status. Što znači ako neka lovačka udruga ima istu članarinu kao i neka druga, moći ćemo unijeti u tablicu jer se razlikuju po vrijednosti lovačka_udruga_id.



Slika 1.11 član-relacijski model

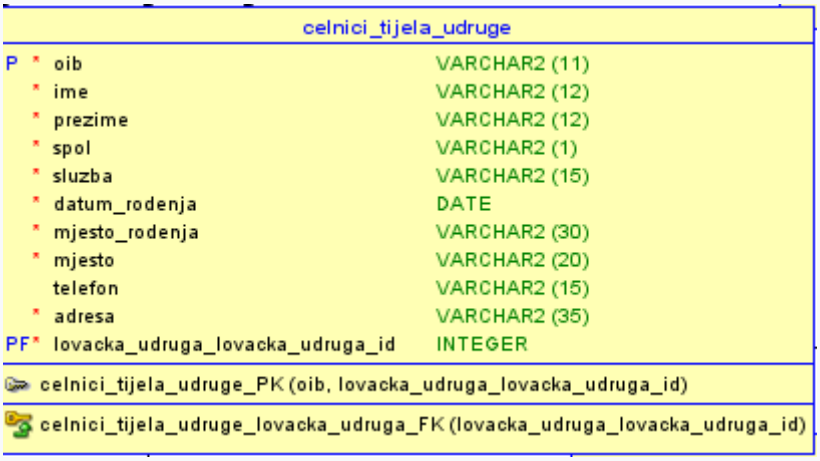
Entitet član sadrži sve članove svih lovačkih udruga. Ovdje se nalaze svi podatci o članovima, a primarni ključ je oib.



Slika 1.12 prijem_u_članstvo-relacijski model

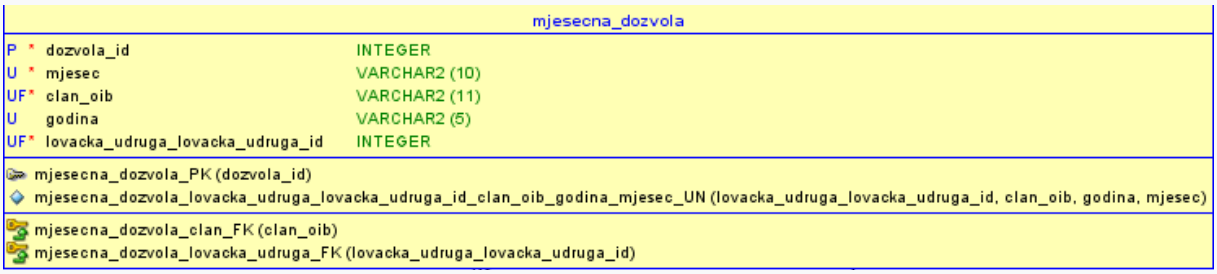
Sada imamo entitet prijem_u_članstvo koji spajamo sa lovačkom udrugom i članom. Veze su 1:n, obvezne i identifikacijske. Primarni

ključ čine vrijednosti datum_upisa, lovačka_udruga_id i oib. Datum_upisa sudjeluje u stvaranju primarnog ključa kako bi mogli pokriti slučaj ako se neki član upiše, ispiše te ponovno upiše. Ako je vrijednost datum_ispisa različita od NULL znači da je član ispisan. Tako da ako npr. želimo provesti upit u kojem dohvaćamo broj članova neke udruge, to ćemo napraviti tako da brojimo sve one retke u entitetu prijem_u_clanstvo gdje je datum ispisa jednak NULL



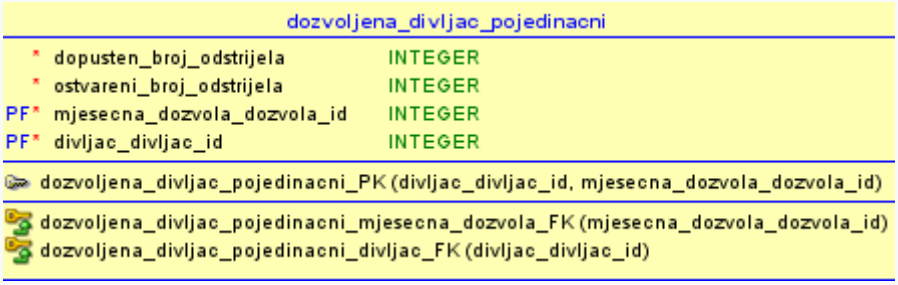
Slika 1.13 čelnici_tijela_udruge-relacijski model

Zatim slijedi entitet čelnici_tijela_udruge. Povezan je sa lovačkom udrugom 1:n, obveznom, identifikacijskom vezom. U ovaj entitet spremamo sve podatke o čelnicima i tijelima udruge. To su uglavnom članovi lovačkih udruga, ali i ne moraju biti članovi. Primarni ključ čine vrijednosti oib i lovačka_udruga_id. Tako da neka osoba može imati službenu ulogu u više lovačkih udruga.



Slika 1.14 mjesečna_dozvola-relacijski model

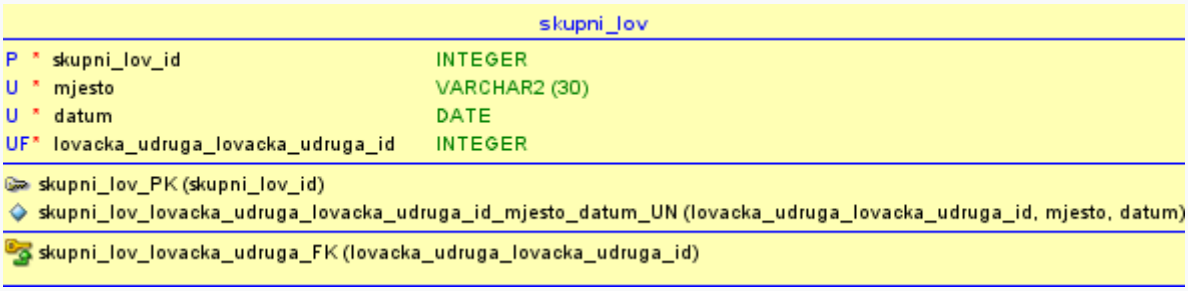
Dolazimo do entiteta mjesečna_dozvola. Povezan je sa lovačkom udrugom i članom. Veze su 1:n, opcionalne te nisu identifikacijske. Primarni ključ čini vrijednost dozvola_id. Jedinstvenost redaka u ovaj tablici očuvana je UNIQUE constraint-om koji čine vrijednosti lovačka_udruga_id, oib, mjesec, godina.



Slika 1.15 dozvoljena_divljač_pojedinačni-relacijski model

Mjesečnu dozvolu i divljač vežemo za entitet dozvoljena_divljač_pojedinačni. Veza divljač – dozvoljena divljač pojedinačni je 1:n, opcionalna i identifikacijska. Veza mjesečna dozvola – dozvoljena divljač pojedinačni je 1:n, obvezna i

identifikacijska. U entitetu dozvoljena_divljač_pojedinačni primarni ključ čine dozvola_id i divljač_id.



Slika 1.16 skupni_lov-relacijski model

Entitet skupni_lov povezan je sa lovačkom udrugom 1:n, opcionalno ali ne-identifikacijskom vezom. Ovdje unosimo podatke u skupnim lovovima lovačkih udruga, odnosno mjesto i datum. Primarni ključ čini vrijednost skupni_lov_id. A jedinstvenost je očuvana UNIQUE constraint-om koji čine vrijednosti lovačka_udruga_id, mjesto, datum.

Slika 1.17 lov-relacijski model

Entitet lov vežemo sa članom i skupnim lovom. Veza skupni lov – lov je 1:n, obvezna i identifikacijska. Veza član – lov je 1:n, opcionalna i

identifikacijska. Ovdje unosimo sve članove koji su sudjelovali u skupnom lovu. Primarni ključ čine vrijednosti skupni_lov_id i oib.

1.18 dozvoljena_divljač_skupni-relacijski model

Na kraju dolazimo do entiteta dozvoljena_divljač_skupni. Ovdje upisujemo dopušten odstrjel i ostvareni odstrjel divljači. Ovaj entitet povezan je sa entitetima divljač i skupni lov. Veza divljač – dozvoljena divljač skupni je 1:n, opcionalna i identifikacijska. Veza skupni lov – dozvoljena divljač skupni je 1:n, obvezna i identifikacijska. Primarni ključ čine vrijednosti skupni_lov_id i divljač_id.

2. SQL skripta

U ovom poglavlju prikazat ćemo kreiranje tablica, unos podataka u tablice, upite nad tablicama, procedure, okidače i indekse.

2.1. Kreiranje tablica

Kod kreiranja tablica bitno je naglasiti da se prvo kreiraju one koje sudjeluju u nekoj drugoj tablici pri stvaranju primarnog ključa. Jasno je da se ne možemo referencirati na neku vrijednost iz tablice koju još nismo kreirali. Iz istog razloga, to je bitno i kod unosa podataka.

```
CREATE TABLE lovacka_udruga (
lovacka_udruga_id INTEGER CONSTRAINT lovacka_udruga_pk PRIMARY KEY,
naziv VARCHAR(30) NOT NULL,
zupanija VARCHAR(32) NOT NULL,
mjesto VARCHAR(30) NOT NULL,
broj_lovista INTEGER NOT NULL,
povrsina_lovista DECIMAL(10, 2) NOT NULL,
CONSTRAINT zupanija_broj_uq UNIQUE(zupanija, broj_lovista)
);
```

Slika 2.1 lovačka_udruga-sql

Za početak uzmimo tablicu lovačka_udruga. Možemo vidjeti (Slika 2.1) da za primarni ključ postavljamo vrijednost lovacka_udruga_id

čiji je tip integer. Kako smo u prvom poglavlju naglasili da za različiti primarni ključ možemo imati dvije iste lovačke udruge, što nije dopušteni, vidimo da postavljamo UNIQUE constraint koji se sastoji od vrijednosti županija i broj_lovista. Time smo osigurali da imamo jedinstvene retke u tablici lovačka udruga.

```
INSERT INTO lovacka_udruga (lovacka_udruga_id, naziv, zupanija, mjesto, broj_lovista, povrsina_lovista)
VALUES (1, 'Fazan', 'Osjecko-baranjska', 'Podgajci Podravski', 143, 6818);

INSERT INTO lovacka_udruga (lovacka_udruga_id, naziv, zupanija, mjesto, broj_lovista, povrsina_lovista)
VALUES (2, 'Lane', 'Vukovarsko-srijemska', 'Bogdanovci', 112, 3218);

INSERT INTO lovacka_udruga (lovacka_udruga_id, naziv, zupanija, mjesto, broj_lovista, povrsina_lovista)
VALUES (3, 'Muflon', 'Primorsko-goranska', 'Bribir', 240, 2393);
```

Slika 2.2 Unos lovačkih udruga

Na slici (Slika 2.2) možemo vidjeti unos podataka u tablicu. Potrebno je unijeti onaj tip podatka kako smo ga definirali u tablici. Ako u prvu zagradu stavimo sve vrijednosti koje su obvezne za unijeti, onda nije bitan redoslijed. Inače ako izostavimo vrijednosti iz prve tablice, bitno je da unesemo vrijednosti redom kako smo ih definirali u tablici.

Pošto su uglavnom sve tablice kreirane na gotovo pa jednak način, prikazati ćemo još jednu i prijeći na upite.

```
CREATE TABLE clanarina (
lovacka_udruga_id INTEGER CONSTRAINT clanarina_fk_lovacka_udruga
REFERENCES lovacka_udruga(lovacka_udruga_id),
status VARCHAR(25) NOT NULL,
iznos DECIMAL(10, 2) NOT NULL,
CONSTRAINT clanarina_pk PRIMARY KEY(lovacka_udruga_id, status)
);
```

Slika 2.3 clanarina-sql

Ovdje imamo tablicu članarina. Možemo primijetiti da se primarni ključ sastoji od više vrijednosti, a to su lovčka_udruga_id i status. Na ovaj način osiguravamo da svaka lovačka udruga može imati proizvoljan iznos članarine. Još vrijedi spomenuti da ako želimo da pri unosu neka vrijednost bude obvezna, nakon označavanja tipa podatka, dodamo oznaku NOT NULL, ako pak ta vrijednost nije obvezna pri unosu, jednostavno izostavimo tu oznaku.

```
INSERT INTO clanarina (lovacka_udruga_id, status, iznos)
VALUES (1, 'povlasteni clan', 300);

INSERT INTO clanarina (lovacka_udruga_id, status, iznos)
VALUES (1, 'pocasni clan', 0);

INSERT INTO clanarina (lovacka_udruga_id, status, iznos)
VALUES (1, 'redovni clan', 1200);
```

Slika 2.4 Unos članarina

2.2 Upiti nad tablicama

Upiti nad tablicama služe nam za dohvaćanje podataka iz tablice ili pak više njih. U upitima možemo brojati retke, izračunati sumu redaka u nekom stupcu, grupirati retke, itd. U nastavku ćemo obraditi većinu mogućnosti.

```
SELECT l.naziv AS "LOVACKA UDRUGA", o.naziv_objekta AS "OBJEKT", o.mjesto, o.broj
FROM lovacka_udruga l INNER JOIN lovno_tehnicki_objekt o
USING (lovacka_udruga_id)
ORDER BY l.naziv, o.naziv_objekta;
```

Slika 2.5 upit_1

Ovdje ispisujemo lovačke udruge i njihove lovno tehničke objekte. Bitno je naglasiti da pošto koristimo INNER JOIN ispisat će se samo one lovačke udruge koje imaju jedan ili više lovno tehničkih objekata, a one koje nemaju neće biti na popisu.

U upitu spajamo lovačku udrugu i lovno tehnički objekt koristeći ključnu riječ USING. USING koristimo ukoliko upit koristi equi-spoj, te ako stupci po kojima se vrši equi-spoj imaju isto ime(strani ključ i primarni ključ na koji se referencira imaju isto ime)

Ispis na kraju poredamo prvo po nazivu lovačke udruge, a zatim po nazivu objekta.

Ako želimo promijeniti naziv stupca koristimo sljedeću naredbu: ime_stupca AS “novo_ime_stupca“.

```
SELECT l.naziv AS "LOVACKA UDRUGA", p.katastarska_opcina AS "KATASTARSKA OPCINA", p.broj_katastarske_cestice AS "BROJ KATASTARSKE CESTICE", k.kultura, k.datum_sjetve AS "DATUM SJETVE", k.datum_berbe AS "DATUM BERBE"
FROM lovacka_udruga l FULL OUTER JOIN obradiva_povrsina p
USING (lovacka_udruga_id)
FULL OUTER JOIN zemlja_kultura k
USING (obradiva_povrsina_id)
ORDER BY l.naziv;
```

Slika 2.6 upit_2

U ovom upitu ispisujemo sve lovačke udruge te njihove obradive površine uz sve kulture koje su bile sađene na njima. Spajamo tablice lovačka_udruga, obradiva_površina i zemlja_kultura. Do ovih podataka smo mogli doći i spajanjem samo obradiva_površina i zemlja_kultura, ali tada ne bismo mogli ispisati naziv lovačke udruge, već samo lovačka_udruga_id. Ovdje možemo primijetiti da koristimo FULL OUTER JOIN, što znači da ćemo ispisati i one lovačke udruge koje nemaju obradivih površina.

```
SELECT c.ime, c.prezime, c.status, c.mjesto, m.dozvola_id, m.godina, d.naziv, di.dopusten_broj_odstrijela, di.ostvaren_broj_odstrijela
FROM clan c INNER JOIN mjesecna_dozvola m
ON(c.oib = m.oib)
INNER JOIN dozvoljena_divljac_pojedinacni di
ON(di.dozvola_id = m.dozvola_id)
INNER JOIN divljac d
ON(d.divljac_id = di.divljac_id)
WHERE m.lovacka_udruga_id = 1 and c.oib = '25469874653'
ORDER BY m.godina;
```

Slika 2.7 upit_3

Ovdje ispisujemo podatke o članu sa oib-om “25469874653“ koji je upisan u lovačku udrugu sa id-em “1“. Spajamo tablice član, mjesečna_dozvola, dozvoljena_divljač_pojedinačni i divljač. Zatim postavljamo gore navedene uvijete, te na kraju poredamo retke po godinama.

```sql
SELECT l.naziv AS "LOVACKA UDRUGA", ISNULL(upisani.broj_upisanih, 0) AS "BROJ UPISANIH", ISNULL(ispisani.broj_ispisanih, 0) AS "BROJ ISPISANIH"
FROM lovacka_udruga l FULL OUTER JOIN
(SELECT lovacka_udruga_id, COUNT(datum_upisa) as broj_upisanih
FROM prijem_u_clanstvo
WHERE datum_ispisa IS NULL
GROUP BY lovacka_udruga_id) upisani
ON(l.lovacka_udruga_id = upisani.lovacka_udruga_id)
FULL OUTER JOIN
(SELECT lovacka_udruga_id, COUNT(datum_upisa) as broj_ispisanih
FROM prijem_u_clanstvo
WHERE datum_ispisa IS NOT NULL
GROUP BY lovacka_udruga_id) ispisani
ON(l.lovacka_udruga_id = ispisani.lovacka_udruga_id)
ORDER BY l.lovacka_udruga_id;
```

Slika 2.8 upit_4

U ovom upitu ispisujemo broj upisanih i ispisanih članova u lovačkoj udruzi. Koristimo funkciju ISNULL(varijabla, 0) kako bi zamijenili NULL vrijednosti sa nulom. Zatim spajamo tablicu lovačka_udruga sa tablicama upisani i ispisani koje smo dobili kao podupit. Također koristimo FULL OUTER JOIN kako bi ispisali i one lovačke udruge koje nemaju ispisanih članova. Obje tablice grupiramo i spajamo po vrijednosti lovačka_udruga_id. Dodajemo i agregirajuću funkciju COUNT() koja djeluje na svim recima istovremeno, te vraća jedan izlazni redak. Može se koristiti i ključna riječ DISTINCT uz agregirajuću funkciju kako bi se isključile duplicirajuće vrijednosti.

```
SELECT lovacka_udruga_id, stroj_id, vrsta_stroja, model_stroja, marka, godina_proizvodnje
FROM sredstva_za_rad
WHERE godina_proizvodnje <
    (SELECT AVG(godina_proizvodnje)
    FROM sredstva_za_rad)
ORDER BY lovacka_udruga_id;
```

Slika 2.9 upit_5

U ovom upitu ispisujemo sva sredstva za rad koja su starija od prosjeka svih ostalih strojeva, te lovačke udruge koje ih posjeduju. Da bi smo dobili prosjek godina, pravimo podupit i koristimo agregirajuću funkciju AVG() koja nam vraća srednju vrijednost godina proizvodnje svih strojeva, svih lovačkih udruga. U ovom slučaju to je jednoretčani podupit koji vraća jedan redak i jedan stupac u vanjski upit, a potom tu vrijednost uspoređujemo u WHERE klauzuli.

```
SELECT d.divljac_id, d.naziv, d.spol, REPLACE(p.lovidbena_godina, '2020', 'pojedinacni') AS "VRSTA LOVA", SUM(di.ostvaren_broj_odstrijela) AS "OSTVARENI ODSTRIJEL"
FROM mjesecna_dozvola m INNER JOIN dozvoljena_divljac_pojedinacni di
ON(m.dozvola_id = di.dozvola_id)
INNER JOIN divljac d
ON(d.divljac_id = di.divljac_id)
INNER JOIN plan_gospodarenja p
ON(d.divljac_id = p.divljac_id)
WHERE m.lovacka_udruga_id=1 and m.godina = '2020' and p.lovidbena_godina = '2020' and p.lovacka_udruga_id = 1
GROUP BY p.dopusten_odstrijel, d.spol, d.divljac_id, d.naziv, p.lovidbena_godina
UNION
SELECT div.divljac_id, div.naziv, div.spol, REPLACE(ps.lovidbena_godina, '2020', 'skupni') AS "VRSTA_LOVA", SUM(dsk.ostvareni_broj_odstrijela) AS "OSTVARENI ODSTRIJEL"
FROM skupni_lov sk INNER JOIN dozvoljena_divljac_skupni dsk
ON(sk.skupni_lov_id = dsk.skupni_lov_id)
INNER JOIN divljac div
ON(div.divljac_id = dsk.divljac_id)
INNER JOIN plan_gospodarenja ps
ON(div.divljac_id = ps.divljac_id)
WHERE sk.lovacka_udruga_id = 1 and EXTRACT(YEAR FROM sk.datum) = '2020' and ps.lovidbena_godina = '2020' and ps.lovacka_udruga_id = 1
GROUP BY ps.dopusten_odstrijel, div.spol, div.divljac_id, div.naziv, ps.lovidbena_godina
ORDER BY divljac_id;
```

Slika 2.10 upit_6

U ovom upitu ispisujemo podatke o divljači te koji je broj odstrjela u 2020. godini. Koristimo skupovnu operaciju UNION tako da

dobijemo podatke o kojem je lovu riječ(skupni ili pojedinačni). Tako dobivamo jednu tablicu koja je nastala unijom dvije tablice. Kako bi ovo bilo izvedivo broj stupaca kao i tipovi podataka u stupcima moraju se poklapati u obje tablice, ali ne i nazivi stupaca. UNION operacija vraća sve retke upita bez duplikata. Ovdje također imamo agregirajuću funkciju SUM() koja sumira sve vrijednosti po redcima u odgovarajućem stupcu.

```
SELECT l.lovacka_udruga_id, l.naziv, l.zupanija, (EXTRACT(YEAR FROM CURRENT_DATE) - FLOOR(AVG(EXTRACT(YEAR FROM c.datum_rodenja)))) AS "PROSIJEK GODINA"
FROM lovacka_udruga l INNER JOIN prijem_u_clanstvo p
ON(l.lovacka_udruga_id = p.lovacka_udruga_id)
INNER JOIN clan c
ON(p.oib = c.oib)
WHERE p.datum_ispisa IS NULL and l.lovacka_udruga_id = p.lovacka_udruga_id
HAVING (EXTRACT(YEAR FROM CURRENT_DATE) - FLOOR(AVG(EXTRACT(YEAR FROM c.datum_rodenja)))) <=
(SELECT (EXTRACT(YEAR FROM CURRENT_DATE) - FLOOR(AVG(EXTRACT(YEAR FROM ca.datum_rodenja))))
FROM lovacka_udruga la INNER JOIN prijem_u_clanstvo pa
ON(la.lovacka_udruga_id = pa.lovacka_udruga_id)
INNER JOIN clan ca
ON(pa.oib = ca.oib)
WHERE pa.datum_ispisa IS NULL and la.lovacka_udruga_id = pa.lovacka_udruga_id)
GROUP BY l.lovacka_udruga_id, l.naziv, l.zupanija;
```

Slika 2.11 upit_7

U ovom upitu ispisujemo sve lovačke udruge u kojima je prosjek godina manji ili jednak prosjeku godina u svim lovačkim udrugama. Koristimo funkciju EXTRACT(YEAR FROM varijabla) kako bi iz varijable tipa DATE izvukli godinu. Zatim koristimo funkciju FLOOR() kako bi zaokružili na nižu cijelu vrijednost. Na kraju imamo opet funkciju AVG() koju smo komentirali u nekom od prethodnih upita. Ovdje vidimo da smo dodali HAVING klauzulu. U ovom slučaju WHERE klauzula selektira retke prema danom uvjetu,

GROUP BY klauzula grupira preostale retke, a HAVING klauzula selektira grupe prema danom uvjetu.

```
SELECT ca.oib, ca.ime, ca.prezime, suma.clan_suma
FROM clan ca INNER JOIN
(SELECT c.oib, SUM(d.ostvaren_broj_odstrijela) as clan_suma
FROM clan c INNER JOIN mjesecna_dozvola m
ON(c.oib = m.oib)
INNER JOIN dozvoljena_divljac_pojedinacni d
ON(m.dozvola_id = d.dozvola_id)
WHERE m.lovacka_udruga_id = 3
GROUP BY c.oib) suma
ON(ca.oib = suma.oib)
WHERE suma.clan_suma =
(SELECT MAX(prva.prva_suma)
FROM(
SELECT c.oib, SUM(d.ostvaren_broj_odstrijela) as prva_suma
FROM clan c INNER JOIN mjesecna_dozvola m
ON(c.oib = m.oib)
INNER JOIN dozvoljena_divljac_pojedinacni d
ON(m.dozvola_id = d.dozvola_id)
WHERE m.lovacka_udruga_id = 3
GROUP BY c.oib) prva )
and ca.oib = suma.oib;
```

Slika 2.12 upit_8

U ovom upitu dohvaćamo podatke o članu sa najviše odstrijeljene divljači u određenoj lovačkoj udruzi. Možemo prepoznati korelirani upit koji se izvršava za svaki redak vanjskog upita. Vanjski upit “proslijedi“ vrijednost ca.oib i onda se podupit izvrši za tu vrijednost. Na taj način vanjski upit i njegov podupit postaju korelirani.

2.3 Procedure

U ovom poglavlju obraditi ćemo procedure koje sadrže grupu SQL i PL/SQL naredbi. Koriste se za organizaciju programerske logike u bazi podataka kojoj pristupaju različite aplikacije.

```
CREATE SEQUENCE lovacka_udruga_sequence
    START WITH 4
    INCREMENT BY 1
    MAXVALUE 5000;

CREATE PROCEDURE lovacka_udruga_insert (
l_naziv in lovacka_udruga.naziv%TYPE,
l_zupanija in lovacka_udruga.zupanija%TYPE,
l_mjesto in lovacka_udruga.mjesto%TYPE,
l_broj_lovista in lovacka_udruga.broj_lovista%TYPE,
l_povrsina_lovista in lovacka_udruga.povrsina_lovista%TYPE
) AS
l_counter INTEGER := 0;
BEGIN
    SELECT COUNT(*)
    INTO l_counter
    FROM lovacka_udruga
    WHERE zupanija = l_zupanija and broj_lovista = l_broj_lovista;

    IF l_counter = 0 THEN
        INSERT INTO lovacka_udruga VALUES(lovacka_udruga_sequence.nextval, l_naziv, l_zupanija, l_mjesto, l_broj_lovista, l_povrsina_lovista);
        COMMIT;
    ELSE
        dbms_output.put_line('Pogreska kod unosa! Lovacka udruga vec unesena.');
        ROLLBACK;
    END IF;

END lovacka_udruga_insert;
/
```

Slika 2.13 procedura_1

Za početak imamo proceduru koja se brine o unosu podataka u tablicu lovačka_udruga(Slika 2.13). Prvo deklariramo ulazne parametre koji moraju biti inicijalizirani u trenutku pozivanja procedure i ne mogu se mijenjati unutar same procedure. Zatim inicijaliziramo varijable koje ćemo koristiti u proceduri. Tijelo procedure nalazi se u BEGIN/END bloku. U varijablu l_counter spremamo broj udruga koje imaju iste parametre kao i ona koju trenutno želimo unijeti. Odnosno

uspoređujemo parametre županija i broj lovišta, jer su oni jedinstveni za svaki redak u tablici. Ako takvih nema u tablici, tada možemo unijeti trenutnu, a u suprotnom ispišemo grešku i ništa ne unosimo. Bitno je napomenuti da je primarni ključ generiran nizom kojeg smo konstruirali izvan procedure.

```
CREATE PROCEDURE lovacka_udruga_update (
l_naziv in lovacka_udruga.naziv%TYPE,
l_zupanija in lovacka_udruga.zupanija%TYPE,
l_mjesto in lovacka_udruga.mjesto%TYPE,
l_broj_lovista in lovacka_udruga.broj_lovista%TYPE,
l_povrsina_lovista in lovacka_udruga.povrsina_lovista%TYPE
) AS
l_counter INTEGER := 0;
BEGIN
    SELECT COUNT(*)
    INTO l_counter
    FROM lovacka_udruga
    WHERE zupanija = l_zupanija and broj_lovista = l_broj_lovista;

    IF l_counter = 1 THEN
        UPDATE lovacka_udruga
        SET naziv = l_naziv, mjesto = l_mjesto, povrsina_lovista = l_povrsina_lovista
        WHERE zupanija = l_zupanija and broj_lovista = l_broj_lovista;
        COMMIT;
    ELSE
        dbms_output.put_line('Pogreska kod unosa! Podatci se ne mogu azurirati. Lovacka udruga ne postoji.');
        ROLLBACK;
    END IF;

END lovacka_udruga_update;
/
```

Slika 2.14 procedura_2

Sljedeća procedura koja se veže uz prvu brine se za ažuriranje podataka nad tablicom lovačka_udruga(Slika 2.14). Proces je sličan kao u prethodnoj proceduri. Prvo provjerimo je li se takva lovačka udruga već nalazi u tablici. Ako se ne nalazi, ispišemo grešku i ništa ne ažuriramo, u suprotnom, ažuriramo onaj redak u kojem se podudaraju vrijednosti županija i broja lovišta. Ne moramo brinuti o

tome hoće li biti više takvih vrijednosti jer smo kod kreiranja tablica postavili UNIQUE constraint koji se brine o jedinstvenosti redaka u ovoj tablici.

```
CREATE PROCEDURE lovno_tehnicki_objekt_insert_update (
o_lovacka_udruga_id IN INTEGER,
o_naziv_objekta IN lovno_tehnicki_objekt.naziv_objekta%TYPE,
o_mjesto IN lovno_tehnicki_objekt.mjesto%TYPE,
o_broj IN INTEGER
) AS
o_counter INTEGER := 0;
BEGIN
    SELECT COUNT(*)
    INTO o_counter
    FROM lovno_tehnicki_objekt
    WHERE o_broj = broj and lovacka_udruga_id = o_lovacka_udruga_id;

    IF o_counter = 0 THEN
        INSERT INTO lovno_tehnicki_objekt VALUES(o_lovacka_udruga_id, o_naziv_objekta, o_mjesto, o_broj);
        COMMIT;
    ELSE
        UPDATE lovno_tehnicki_objekt
        SET mjesto = o_mjesto
        WHERE broj = o_broj and lovacka_udruga_id = o_lovacka_udruga_id;
        COMMIT;
    END IF;

    EXCEPTION
    WHEN OTHERS THEN
    dbms_output.put_line('Error');
        ROLLBACK;

END lovno_tehnicki_objekt_insert_update;
/
```

Slika 2.15 procedura_3

Ovdje imamo proceduru koja se brine za unos i ažuriranje podataka u tablici lovno_tehnički_objekt(Slika 2.15). Provjeravamo ima li već unesen objekt sa istim brojem u navedenoj lovačkoj udruzi. Ako nema unosimo ga kao novi objekt, a u suprotnom ažuriramo vrijednosti odgovarajućeg objekta, odnosno ažuriramo mu lokaciju.

```
CREATE PROCEDURE ispis_clana (
i_lovacka_udruga_id IN INTEGER,
i_oib IN clan.oib%TYPE,
i_datum_ispisa IN DATE
) AS
i_counter INTEGER := 0;
BEGIN

    SELECT COUNT(*)
    INTO i_counter
    FROM prijem_u_clanstvo
    WHERE lovacka_udruga_id = i_lovacka_udruga_id and oib = i_oib and datum_ispisa IS NULL;

    IF i_counter = 0 THEN
        dbms_output.put_line('Pogreska! Aktivni clan sa navedenim oib-om u navedenoj lovackoj udruzi ne postoji.');
        ROLLBACK;
    ELSE
        UPDATE prijem_u_clanstvo
        SET datum_ispisa = i_datum_ispisa
        WHERE lovacka_udruga_id = i_lovacka_udruga_id and oib = i_oib;
    END IF;

    EXCEPTION
    WHEN OTHERS THEN
    dbms_output.put_line('Error');
        ROLLBACK;

END ispis_clana;
/
```

Slika 2.16 procedura_4

Ova procedura(Slika 2.16) koristi se za ispisivanja člana iz udruge. Parametri koje joj proslijedimo su lovačka_udruga_id, oib člana, te datum ispisa. U proceduri tražimo člana iz tablice prijem_u_članstvo gdje se poklapaju vrijednosti. Ako smo pronašli takvog, ažuriramo mu vrijednost datum_ispisa, a u suprotnom izbacimo grešku i ništa ne ažuriramo.

```
CREATE PROCEDURE lista_clanova (
l_naziv IN lovacka_udruga.naziv%TYPE,
l_zupanija IN lovacka_udruga.zupanija%TYPE,
l_broj_lovista IN lovacka_udruga.broj_lovista%TYPE
) AS
l_oib VARCHAR(11);
l_ime VARCHAR(12);
l_prezime VARCHAR(14);
l_mjesto VARCHAR(20);
lista_clanova_cursor SYS_REFCURSOR;
l_counter INTEGER := 0;

BEGIN
    SELECT COUNT(*)
    INTO l_counter
    FROM lovacka_udruga
    WHERE naziv = l_naziv and zupanija = l_zupanija and broj_lovista = l_broj_lovista;

    IF l_counter = 1 THEN
        OPEN lista_clanova_cursor FOR
            SELECT c.oib, c.ime, c.prezime, c.mjesto
            FROM lovacka_udruga l INNER JOIN prijem_u_clanstvo p
            ON(l.lovacka_udruga_id = p.lovacka_udruga_id)
            INNER JOIN clan c
            ON(p.oib = c.oib)
            WHERE l.zupanija = l_zupanija and l.broj_lovista = l_broj_lovista and p.datum_ispisa IS NULL
            ORDER BY c.ime;

        LOOP
            FETCH lista_clanova_cursor
            INTO l_oib, l_ime, l_prezime, l_mjesto;
            EXIT WHEN lista_clanova_cursor%NOTFOUND;
            DBMS_OUTPUT.PUT_LINE(
            'oib = ' || l_oib ||
            ', ime = ' || l_ime ||
            ', prezime = ' || l_prezime ||
            ', prebivaliste = ' || l_mjesto);
        END LOOP;
        CLOSE lista_clanova_cursor;
    ELSE
        DBMS_OUTPUT.PUT_LINE('Navedena lovacka udruga ne nalazi se u bazi podataka.');
    END IF;
END lista_clanova;
/
```

Slika 2.17 procedura_5

Slijedi zadnja procedura(Slika 2.17) pomoću koje ispisujemo listu članova iz određene lovačke udruge. Parametri koje proslijedimo proceduri su naziv lovačke udruge, županija i broj lovišta. Zatim kao i svaki put do sada, pokušamo pronaći takvu lovačku udrugu. Ako postoji, tada otvaramo kursor koji smo deklarirali u DECLARE bloku. Kursor nam služi za dohvat redaka iz tablice koja je rezultat upita. Te

retke, odnosno stupce redaka spremamo u varijable koje smo prethodno deklarirali i zatim otvaramo petlju u kojoj redom dohvaćamo spremljene podatke, te ih ispisujemo.

2.4 Okidači

Okidači su aktivni elementi SQL-a, odnosno PL/SQL-a. Pozivaju se automatski kada se želi izvršiti neka od JMP naredbi (INSERT, UPDATE ili DELETE). Dijelimo ih prema vremenu pokretanja na BEFORE i AFTER, te prema načinu djelovanja na ROW-LEVEL i STATEMENT-LEVEL.

U nastavku ćemo pokazati 4 primjera okidača koji će biti dovoljni za opisivanje ovog elementa SQL-a.

```
CREATE TRIGGER trigger_lovacka_udruga
    AFTER UPDATE ON lovacka_udruga
    FOR EACH ROW WHEN
    (NEW.mjesto != OLD.mjesto)
    BEGIN
        dbms_output.put_line('Promjena lokacije lovacke udruge!');
    END trigger_lovacka_udruga;
/
```

Slika 2.18 okidač_1

Ovdje imamo primjer (Slika 2.18), AFTER, ROW-LEVEL okidača. Okida se nakon ažuriranja redaka u tablici lovačka_udruga, onda kada lovačka udruga promjeni svoju lokaciju. U tom slučaju ispisujemo poruku o promjeni. Također okida se onoliko puta koliko redaka ažuriramo(ROW-LEVEL).

```
CREATE TRIGGER lovacka_udruga_loviste_trigger
    BEFORE INSERT ON lovacka_udruga
    FOR EACH ROW WHEN
    (new.povrsina_lovista = 0 or new.broj_lovista = 0)
    BEGIN
        raise_application_error(-20100, 'Nemoguce unjeti lovacku udrugu bez lovista!');

    END lovakca_udruga_loviste_trigger;
/
```

Slika 2.19 okidač_2

Zatim slijedi okidač koji sprječava unos lovačke udruge koja nema lovište(Slika 2.19). Okidač se okida prije unosa te pokreće ROLLBACK naredbu.

```
CREATE OR REPLACE TRIGGER azuriranje_ostvarenog_odstrijela
    AFTER INSERT ON dozvoljena_divljac_pojedinacni
    FOR EACH ROW WHEN
    (NEW.ostvaren_broj_odstrijela > 0)
    BEGIN
        UPDATE plan_gospodarenja
        SET ostvareni_odstrijel = ostvareni_odstrijel + :new.ostvaren_broj_odstrijela
        WHERE lovacka_udruga_id = (
            SELECT lovacka_udruga_id
            FROM mjesecna_dozvola
            WHERE dozvola_id = :new.dozvola_id)
            and
            lovidbena_godina = (
            SELECT godina
            FROM mjesecna_dozvola
            WHERE dozvola_id = :new.dozvola_id)
            and
            divljac_id = :new.divljac_id;

            DBMS_OUTPUT.PUT_LINE('Ostvareni odstrijel azuriran!');
    END azuriranje_ostvarenog_odstrijela;
/
```

Slika 2.20 okidač_3

Okidač ažuriranje_ostvarenog_odstrijela (Slika 2.20) okida se nakon unosa odstrjela divljači koji je veći od 0. Zadaća mu je da poveća ukupan broj odstrijeljene divljači u drugoj tablici, odnosno u tablici plan_gospodarenja. Također, i ovo je ROW-LEVEL okidač, jer želimo da se pokrene za svaki unos.

```
CREATE TRIGGER plan_gospodarenja_trigger
    BEFORE UPDATE ON plan_gospodarenja
    BEGIN
        dbms_output.put_line('Mijenjanje dopusteno u slucaju pogreske. Potrebno priloziti obrazlozenje.');
    END plan_gospodarenja_trigger;
/
```

Slika 2.21 okidac_4

I zadnji okidač, nad tablicom plan_gospodarenja okida se kod ažuriranja podataka. Dopuštamo promjenu, ali ispisujemo poruku kojom napominjemo o ažuriranju. Ovaj okidač je STATEMENT-LEVEL i okida se samo jednom, neovisno o tome koliko redaka je promijenjeno u tablici.

Ovime smo pokrili sve vrste okidača i možemo prijeći na indekse.

2.5 Indeksi

Indeksi u bazama podataka su dodatni podaci koji se koriste za brzi pristup određenim redcima tablice. Nedostatak indeksa je vrijeme potrebno za dodavanje indeksa za svaki novi redak kao i dodatni memorijski prostor potreban za pohranjivanje indeksa.

Indeksi se zasnivaju na strukturi podataka koju zovemo B-stablo te su prigodni za stupce koji sadrže veliki broj različitih vrijednosti.

Također postoje i BITMAP indeksi koji su pogodni za malen broj različitih vrijednosti.

```
--BITMAX INDEX - POGODAN ZA STUPCE SA MALO RAUZLICITIH VRIJEDNOSTI
-------------------------------------------------------------------------------------------------------
CREATE BITMAP INDEX plan_gospodarenja_Bindex ON plan_gospodarenja(lovidbena_godina);

--Svi upiti vezani za lovdibenu godinu biti ce ubrzani
SELECT * FROM plan_gospodarenja WHERE lovidbena_godina = '2019';

--INDEX - POGODAN ZA STUPCE SA PUNO RAZLICITIH VRIJEDNOSTI
-------------------------------------------------------------------------------------------------------
CREATE INDEX dozvoljena_divljac_pojedinacni_index ON dozvoljena_divljac_pojedinacni(dozvola_id);

--Svi upiti vezani uz id dozvole biti ce ubrzani
SELECT * FROM dozvoljena_divljac_pojedinacni WHERE dozvola_id = 240;


--BITMAP INDEX - POGODAN ZA STUPCE SA MALO RAZLICITIH VRIJEDNOSTI
-------------------------------------------------------------------------------------------------------
CREATE BITMAP INDEX dozovljena_divljac_pojedinacni_Bindex ON dozvoljena_divljac_pojedinacni(divljac_id);

--Svi upiti vezani uz divljac_id biti ce ubrzani
SELECT * FROM dozvoljena_divljac_pojedinacni WHERE divljac_id = 9;


--INDEX - POGODAN ZA STUPCE SA MALO RAZLICITIH VRIJEDNOSTI
-------------------------------------------------------------------------------------------------------
CREATE BITMAP INDEX mjesecna_dozvola_Bindex ON mjesecna_dozvola(godina);
DROP INDEX mjesecna_dozvola_Bindex;
--Svi upiti vezani uz godinu dozvole biti ce ubrzani
SELECT * FROM mjesecna_dozvola WHERE godina = '2018';
```

Slika 2.22 indeksi

Na slici (Slika 2.22) možemo vidjeti implementacije indeksa i koje upite oni ubrzavaju. Uglavnom, svi upiti vezani za stupac nad kojim smo postavili indeksaciju biti će ubrzani.

3. Zaključak

Predstavili smo implementaciju baze podataka za lovačke udruge.

Modeliranje podataka je prvi korak u procesu razvoja baze podataka. Sastoji se od prikupljanja i analiziranja podataka koje poslovni proces treba pratiti. Model entiteta i veza je primjer konceptualnog modela. Važan je jer obuhvaća i opisuje informacije potrebne za poslovni proces, smanjuje mogućnost grešaka i nesporazuma, te opisuje kako bi trebala izgledati dokumentacija idealnog sustava.
U drugom koraku razvoja baze podataka, informacije koje su modelirane MEV-om pretvaraju se u tablice. To se naziva relacijski model i osnovni organizacijski element je tablica. Tablice daju detaljniju specifikaciju o podacima koje nose. Osnovni cilj organizacije podataka u tablice je jednostavno manipuliranje podacima, njihova promjena i dohvat podataka prema različitim upitima.

Stoga smo prikazali MEV dijagram i relacijski model, objasnili veze između tablica i proces sastavljanja primarnih i jedinstvenih ključeva.

Zatim naredbama SQL-a izrađujemo fizičku strukturu baze podataka. SQL također koristimo za unos podataka, te pristup i manipuliranje nad istima. Tako da smo nakon MEV-a i relacijskog modela prešli na

sql skriptu i objasnili implementaciju istih tih tablica koje smo pravili u MEV dijagramu. Obradili smo upite nad njima, procedure, okidače i indekse. To je više-manje sve što je potrebno za izradu jedne baze podataka koja je funkcionalna i upotrebljiva.

Možemo zaključiti kako je baza podataka centralizirani strukturirani skup podataka spremljen na nekakvom računalnom sustavu. Omogućen nam je dohvat, dodavanje, modificiranje i brisanje podataka. Obrađujemo podatke na smislen način i transformiramo ih u željene informacije.
Uglavnom unutar same baze podaci su spremljeni u sirovom obliku, a tek kada se podaci dohvate ili se izvrši nekakav upit nad njima, transformiraju se u korisnu informaciju.

4. Literatura

- Materijali sa predavanja