REPUBLIKA E SHQIPERISË

UNIVERSITETI POLITEKNIK I TIRANËS
FAKULTETI I TEKNOLOGJISË SË INFORMACIONIT

Inxhinieri informatike

Detyrë Kursi

Lënda: Bazat e të Dhënave

Grupi: III-B

Punoi: Piro Gjikdhima Pranoi: Prof.Asoc. Hakik Paci

Viti Akademik 2024 - 2025

Kërkesa Bazë:

Të ndërtohet skema dhe objektet e nevojshme për të ndërtuar një bazë të dhënash në **Oracle** e cila do të përdoret për menaxhimin e një rrjeti pikash parkimi. Funksionalitet dhe kërkesat e nevojshme janë si në vijim:

- Klienti (Emri, Mbiemri, Nr Kontakti, eMail, Adresa, Klient ID, Karte Anëtarësie, etj)
- Pika parkimi (Vendodhje Parkimi)
- Mjete te regjistruara
- Kartat e Anëtarësisë (historiku)
- Çmime Parkimi me fasha oraresh dhe e ndryshme sipas pikës se parkimit (psh, pika 1, 0-1 ore 100 Lek, 1-3 ore 200 Lek, 3-8 ore 300 Lek)
- Abonime Mujore ose me numër herësh
- Aktiviteti i parkimit (me dhe pa klient)
- Siguria (Punonjësi i parkimit, Administratori, Kontrollori)
- Mbyllje aktiviteti ditor (gjendja e arkës për secilin punonjës ne secilën pike)
- Shitjes do ti ofrohet mundësia që të bëhet edhe anulim i tyre duke gjeneruar te njëjtën fature me vlere negative ne rastet kur ka pagese. Kjo do realizohet me ane te procedurave ne DB.
- Të mos lejohet shitja nëse një mjet (me te njëjtën targë) të jetë i parkuar në një pikë parkimi në atë çast përdorur trigger-at.

SKEMA E DATABAZËS

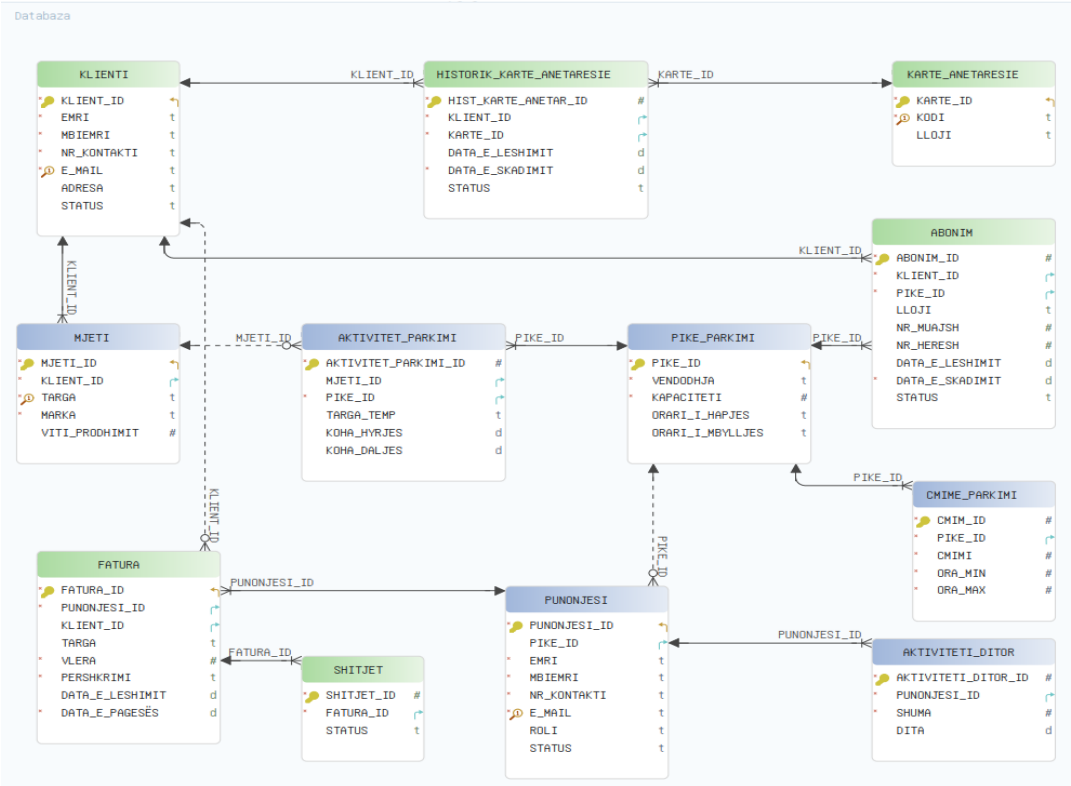

Figura 1. Skema me gjithë tabelat

1. STRUKTURA E TABELAVE

1.1 KLIENTI

```
CREATE TABLE KLIENTI
(
    KLIENT_ID   NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    EMRI        VARCHAR2(20)  NOT NULL,
    MBIEMRI     VARCHAR2(100) NOT NULL,
    NR_KONTAKTI VARCHAR2(20)  NOT NULL,
    E_MAIL      VARCHAR2(100) NOT NULL UNIQUE,
    ADRESA      VARCHAR2(100) DEFAULT NULL,
    STATUS      VARCHAR2(20)  DEFAULT 'AKTIV' CHECK (STATUS IN ('AKTIV',
'JOAKTIV'))
);
```

Përshkrimi: Tabela kryesore për ruajtjen e informacioneve të klientëve të sistemit të parkimit. Çdo klient ka një ID unik automatik dhe status që tregon nëse është aktiv apo jo.

1.2 KARTE_ANETARESIE

```
CREATE TABLE KARTE_ANETARESIE
(
    KARTE_ID NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    KODI     VARCHAR2(10) NOT NULL UNIQUE,
    LLOJI    VARCHAR2(20) DEFAULT 'STANDARD' CHECK ( LLOJI IN ('STANDARD',
'PREMIUM'))
);
```

Përshkrimi: Tabela për ruajtjen e kartave të anëtarësisë. Kartat mund të jenë STANDARD ose PREMIUM me kode unike identifikuese.

1.3 HISTORIK_KARTE_ANETARESIE

```
CREATE TABLE HISTORIK_KARTE_ANETARESIE
(
    HIST_KARTE_ANETAR_ID NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    KLIENT_ID            NUMBER NOT NULL,
    KARTE_ID             NUMBER NOT NULL,
    DATA_E_LESHIMIT      DATE         DEFAULT SYSDATE,
    DATA_E_SKADIMIT      DATE   NOT NULL,
    STATUS               VARCHAR2(20) DEFAULT 'AKTIV' CHECK (STATUS IN
('AKTIV', 'JOAKTIV', 'SKADUAR')),
    CONSTRAINT fk_hist_klient FOREIGN KEY (KLIENT_ID) REFERENCES KLIENTI
(KLIENT_ID),
    CONSTRAINT fk_hist_karte FOREIGN KEY (KARTE_ID) REFERENCES
KARTE_ANETARESIE (KARTE_ID)
);
```

Përshkrimi: Tabela për ruajtjen e historikut të kartave të anëtarësisë të lëshuara për klientët. Mban shënim të datave të lëshimit, skadimit dhe statusit.

1.4 MJETI

```
CREATE TABLE MJETI
(
    MJETI_ID       NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    KLIENT_ID      NUMBER       NOT NULL,
    TARGA          VARCHAR2(10) NOT NULL UNIQUE,
    MARKA          VARCHAR2(30) NOT NULL,
    VITI_PRODHIMIT NUMBER(4),
    CONSTRAINT fk_mjeti_klient FOREIGN KEY (KLIENT_ID) REFERENCES KLIENTI
(KLIENT_ID)
);
```

Përshkrimi: Tabela për ruajtjen e informacioneve të mjeteve të regjistruara në sistem. Çdo mjet i përket një klienti specifik dhe ka targë unike.

1.5 PIKE_PARKIMI

```
CREATE TABLE PIKE_PARKIMI
(
    PIKE_ID          VARCHAR2(10) NOT NULL PRIMARY KEY,
    VENDODHJA        VARCHAR2(20) NOT NULL,
    KAPACITETI       NUMBER       NOT NULL,
    ORARI_I_HAPJES   VARCHAR2(20) DEFAULT '07:00',
    ORARI_I_MBYLLJES VARCHAR2(20) DEFAULT '23:00'
);
```

Përshkrimi: Tabela për ruajtjen e informacioneve të pikave të parkimit. Çdo pikë ka vendodhje, kapacitet dhe orar pune të përcaktuar.

1.6 CMIME_PARKIMI

```
CREATE TABLE CMIME_PARKIMI
(
    CMIM_ID NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    PIKE_ID VARCHAR2(10) NOT NULL,
    CMIMI   NUMBER       NOT NULL,
    ORA_MIN NUMBER(2)    NOT NULL,
    ORA_MAX NUMBER(2)    NOT NULL,
    CONSTRAINT fk_cmime_pike FOREIGN KEY (PIKE_ID) REFERENCES PIKE_PARKIMI
(PIKE_ID)
);
```

Përshkrimi: Tabela për ruajtjen e çmimeve të parkimit sipas fashave orore për çdo pikë parkimi. Lejon përcaktimin e çmimeve të ndryshme për intervale të ndryshme orare.

1.7 ABONIM

```
CREATE TABLE ABONIM
(
    ABONIM_ID       NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    KLIENT_ID       NUMBER       NOT NULL,
```

```
    PIKE_ID         VARCHAR2(20) NOT NULL,
    LLOJI           VARCHAR2(20) CHECK (LLOJI IN ('MUJOR', 'NR_HERESH')),
    NR_MUAJSH       NUMBER,
    NR_HERESH       NUMBER,
    DATA_E_LESHIMIT DATE         DEFAULT SYSDATE,
    DATA_E_SKADIMIT DATE         NOT NULL,
    STATUS          VARCHAR2(20) DEFAULT 'AKTIV' CHECK (STATUS IN ('AKTIV',
'JOAKTIV')),
    CONSTRAINT fk_abonim_klient FOREIGN KEY (KLIENT_ID) REFERENCES KLIENTI
(KLIENT_ID),
    CONSTRAINT fk_abonim_pike FOREIGN KEY (PIKE_ID) REFERENCES PIKE_PARKIMI
(PIKE_ID)
);
```

Përshkrimi: Tabela për ruajtjen e abonimeve të klientëve. Abonimet mund të jenë mujore ose me numër herësh të caktuar.

1.8 AKTIVITET_PARKIMI

```
CREATE TABLE AKTIVITET_PARKIMI
(
    AKTIVITET_PARKIMI_ID NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    MJETI_ID             NUMBER,
    PIKE_ID              VARCHAR2(10) NOT NULL,
    TARGA_TEMP           VARCHAR2(10) NULL,
    KOHA_HYRJES          DATE DEFAULT SYSDATE,
    KOHA_DALJES          DATE,
    CONSTRAINT fk_aktivitet_parkim_mjet FOREIGN KEY (MJETI_ID) REFERENCES
MJETI (MJETI_ID),
    CONSTRAINT fk_aktivitet_parkim_pike FOREIGN KEY (PIKE_ID) REFERENCES
PIKE_PARKIMI (PIKE_ID)
);
```

Përshkrimi: Tabela për ruajtjen e aktiviteteve të parkimit. Mban shënim të hyrjes dhe daljes së mjeteve në pika parkimi, duke mbështetur si mjete të regjistruara ashtu edhe ato me targa të përkohshme.

1.9 PUNONJESI

```
CREATE TABLE PUNONJESI
(
    PUNONJESI_ID NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    PIKE_ID      VARCHAR2(10),
    EMRI         VARCHAR2(20)  NOT NULL,
    MBIEMRI      VARCHAR2(100) NOT NULL,
    NR_KONTAKTI  VARCHAR2(20)  NOT NULL,
    E_MAIL       VARCHAR2(100) NOT NULL UNIQUE,
    ROLI         VARCHAR2(20) DEFAULT 'PUNONJESI' CHECK (ROLI IN
('ADMINISTRATORI', 'PUNONJESI', 'KONSTROLLORI')),
    STATUS       VARCHAR2(20) DEFAULT 'AKTIV' CHECK (STATUS IN ('AKTIV',
'JOAKTIV')),
    CONSTRAINT fk_punonjesi_pike FOREIGN KEY (PIKE_ID) REFERENCES
```

```
PIKE_PARKIMI (PIKE_ID)
);
```

Përshkrimi: Tabela për ruajtjen e informacioneve të punonjësve të sistemit. Çdo punonjës ka rol të përcaktuar dhe mund të jetë i caktuar në një pikë parkimi specifike.

1.10 AKTIVITETI_DITOR

```
CREATE TABLE AKTIVITETI_DITOR
(
    AKTIVITETI_DITOR_ID NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    PUNONJESI_ID        NUMBER NOT NULL,
    SHUMA               NUMBER NOT NULL,
    DITA                DATE DEFAULT TRUNC(SYSDATE),
    CONSTRAINT fk_aktivitet_ditor_punonjes FOREIGN KEY (PUNONJESI_ID)
REFERENCES PUNONJESI (PUNONJESI_ID)
);
```

Përshkrimi: Tabela për ruajtjen e aktivitetit ditor të punonjësve, duke përfshirë shumën totale të arkëtuar për çdo ditë.

1.11 FATURA

```
CREATE TABLE FATURA
(
    FATURA_ID       NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    PUNONJESI_ID    NUMBER        NOT NULL,
    KLIENT_ID       NUMBER,
    TARGA           VARCHAR2(10),
    VLERA           NUMBER        NOT NULL,
    PËRSHKRIMI      VARCHAR2(255) NOT NULL,
    DATA_E_LESHIMIT DATE DEFAULT SYSDATE,
    DATA_E_PAGESËS  DATE          NOT NULL,
    CONSTRAINT fk_fatura_punonjes FOREIGN KEY (PUNONJESI_ID) REFERENCES
PUNONJESI (PUNONJESI_ID),
    CONSTRAINT fk_fatura_klient FOREIGN KEY (KLIENT_ID) REFERENCES KLIENTI
(KLIENT_ID)
);
```

Përshkrimi: Tabela për ruajtjen e faturave të lëshuara për shërbime të ndryshme parkimi, karta anëtarësie dhe abonime.

1.12 SHITJET

```
CREATE TABLE SHITJET
(
    SHITJET_ID NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    FATURA_ID  NUMBER NOT NULL,
    STATUS     VARCHAR2(10) DEFAULT 'PERFUNDUAR' CHECK ( STATUS IN
('PERFUNDUAR', 'ANULUAR')),
    CONSTRAINT fk_shitjet_fatura FOREIGN KEY (FATURA_ID) REFERENCES FATURA
(FATURA_ID)
);
```

Përshkrimi: Tabela për ruajtjen e statusit të shitjeve. Lejon gjurmimin e shitjeve të përfunduara dhe të anuluara.

2. FUNKSIONET

2.1 f_llogarit_cmimin

```
CREATE OR REPLACE FUNCTION f_llogarit_cmimin(
    p_pike_id IN VARCHAR2,
    p_koha_hyrjes IN DATE,
    p_koha_daljes IN DATE
) RETURN NUMBER AS
    v_cmimi_total NUMBER := 0;
    v_total_ore   NUMBER;
    CURSOR c_cmime IS
        SELECT CMIMI, ORA_MIN, ORA_MAX
        FROM CMIME_PARKIMI
        WHERE PIKE_ID = p_pike_id
        ORDER BY ORA_MIN;
BEGIN
    v_total_ore := CEIL((p_koha_daljes - p_koha_hyrjes) * 24);

    FOR r_cmim IN c_cmime
        LOOP
            IF v_total_ore BETWEEN r_cmim.ORA_MIN AND r_cmim.ORA_MAX THEN
                v_cmimi_total := r_cmim.CMIMI;
                EXIT;
            END IF;
        END LOOP;

    IF v_cmimi_total = 0 THEN
        SELECT MAX(CMIMI)
        INTO v_cmimi_total
        FROM CMIME_PARKIMI
        WHERE PIKE_ID = p_pike_id;
    END IF;
    DBMS_OUTPUT.PUT_LINE('Llogaritja e cmimit total: ' || v_cmimi_total);
    RETURN v_cmimi_total;
END f_llogarit_cmimin;
/
```

Përshkrimi: Llogarit çmimin e parkimit bazuar në pikën e parkimit dhe kohëzgjatjen e parkimit. Përdor tabelën CMIME_PARKIMI për të gjetur çmimin e përshtatshëm sipas fashave orare.

2.2 f_vende_te_lira

```
CREATE OR REPLACE FUNCTION f_vende_te_lira(
    p_pike_id IN VARCHAR2
) RETURN NUMBER AS
    v_kapaciteti NUMBER;
    v_te_zena    NUMBER;
    v_te_lira    NUMBER;
BEGIN
    SELECT KAPACITETI
    INTO v_kapaciteti
    FROM PIKE_PARKIMI
    WHERE PIKE_ID = p_pike_id;

    SELECT COUNT(*)
    INTO v_te_zena
    FROM AKTIVITET_PARKIMI
    WHERE PIKE_ID = p_pike_id
      AND (KOHA_DALJES IS NULL OR KOHA_DALJES > SYSDATE);

    v_te_lira := v_kapaciteti - v_te_zena;

    RETURN v_te_lira;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 0;
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20020, 'Gabim gjatë llogaritjes së vendeve
të lira: ' || SQLERRM);
        RETURN 0;
END f_vende_te_lira;
/
```

Përshkrimi: Llogarit numrin e vendeve të lira në një pikë parkimi specifike duke krahasuar kapacitetin total me numrin e mjeteve aktualisht të parkuara.

2.3 f_valido_orarin_parkimit

```
CREATE OR REPLACE FUNCTION f_valido_orarin_parkimit(
    p_pike_id IN VARCHAR2,
    p_koha IN DATE DEFAULT SYSDATE
) RETURN VARCHAR2 AS
    v_ora_hapjes VARCHAR2(20);
    v_ora_mbylljes VARCHAR2(20);
    v_ora_aktuale VARCHAR2(5);
BEGIN
    SELECT ORARI_I_HAPJES, ORARI_I_MBYLLJES
    INTO v_ora_hapjes, v_ora_mbylljes
    FROM PIKE_PARKIMI
    WHERE PIKE_ID = p_pike_id;
```

```
    v_ora_aktuale := TO_CHAR(p_koha, 'HH24:MI');

    IF v_ora_aktuale BETWEEN v_ora_hapjes AND v_ora_mbylljes THEN
        RETURN 'HAPUR';
    ELSE
        RETURN 'MBYLLUR';
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        RETURN 'GABIM';
END f_valido_orarin_parkimit;
/
```

Përshkrimi: Verifikon nëse një pikë parkimi është e hapur në një kohë të caktuar bazuar në orarin e punës së pikës.

2.4 f_klienti_ekziston

```
CREATE OR REPLACE FUNCTION f_klienti_ekziston(
    p_klient_id IN NUMBER
) RETURN NUMBER AS
    v_status VARCHAR2(20);
BEGIN
    SELECT STATUS
    INTO v_status
    FROM KLIENTI
    WHERE KLIENT_ID = p_klient_id;

    IF v_status = 'AKTIV' THEN
        RETURN 1;
    ELSE
        RETURN 0;
    END IF;
END;
/
```

Përshkrimi: Verifikon nëse një klient ekziston dhe është aktiv në sistem. Kthen 1 nëse klienti është aktiv, 0 në rast të kundërt.

2.5 f_mjeti_ekziston

```
CREATE OR REPLACE FUNCTION f_mjeti_ekziston(
    p_targa IN VARCHAR2
) RETURN NUMBER AS
    v_count NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO v_count
    FROM AKTIVITET_PARKIMI
    WHERE (TARGA_TEMP = p_targa OR
           MJETI_ID IN (SELECT MJETI_ID FROM MJETI WHERE TARGA = p_targa))
      AND (KOHA_DALJES IS NULL OR KOHA_DALJES > SYSDATE);
```

```
    IF v_count > 0 THEN
        RETURN 1;
    ELSE
        RETURN 0;
    END IF;
END;
/
```

Përshkrimi: Verifikon nëse një mjet me targë të caktuar është aktualisht i parkuar në ndonjë pikë parkimi.

2.6 f_punonjesi_ekziston

```
CREATE OR REPLACE FUNCTION f_punonjesi_ekziston(
    p_punonjes_id IN NUMBER
) RETURN VARCHAR2 AS
    v_status VARCHAR2(20);
BEGIN
    SELECT STATUS
    INTO v_status
    FROM PUNONJESI
    WHERE PUNONJESI_ID = p_punonjes_id;

    IF v_status = 'AKTIV' THEN
        RETURN 1;
    ELSE
        RETURN 0;
    END IF;
END;
/
```

Përshkrimi: Verifikon nëse një punonjës ekziston dhe është aktiv në sistem.

3. PROCEDURAT

3.1 Procedurat për Klientin

sp_regjistro_klient

```
CREATE OR REPLACE PROCEDURE sp_regjistro_klient(
    p_emri IN VARCHAR2,
    p_mbiemri IN VARCHAR2,
    p_nr_kontakti IN VARCHAR2,
    p_email IN VARCHAR2,
    p_adresa IN VARCHAR2 DEFAULT NULL
) AS
BEGIN
    INSERT INTO KLIENTI(EMRI, MBIEMRI, NR_KONTAKTI, E_MAIL, ADRESA)
    VALUES (p_emri, p_mbiemri, p_nr_kontakti, p_email, p_adresa);
    COMMIT;
```

```
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        RAISE_APPLICATION_ERROR(-20001, 'Email ekziston në sistem.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Gabim gjatë regjistrimit të
klientit: ' || SQLERRM);
END sp_regjistro_klient;
/
```

Përshkrimi: Regjistron një klient të ri në sistem me të gjitha informacionet e nevojshme. Kontrollon për duplikate email.

sp_perditeso_klient

```
CREATE OR REPLACE PROCEDURE sp_perditeso_klient(
    p_klient_id IN NUMBER,
    p_emri IN VARCHAR2 DEFAULT NULL,
    p_mbiemri IN VARCHAR2 DEFAULT NULL,
    p_nr_kontakti IN VARCHAR2 DEFAULT NULL,
    p_email IN VARCHAR2 DEFAULT NULL,
    p_adresa IN VARCHAR2 DEFAULT NULL
) AS
BEGIN
    UPDATE KLIENTI
    SET EMRI        = COALESCE(p_emri, EMRI),
        MBIEMRI     = COALESCE(p_mbiemri, MBIEMRI),
        NR_KONTAKTI = COALESCE(p_nr_kontakti, NR_KONTAKTI),
        E_MAIL      = COALESCE(p_email, E_MAIL),
        ADRESA      = COALESCE(p_adresa, ADRESA)
    WHERE KLIENT_ID = p_klient_id;

    IF SQL%ROWCOUNT = 0 THEN
        RAISE_APPLICATION_ERROR(-20003, 'Klienti nuk u gjet.');
    END IF;
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        RAISE_APPLICATION_ERROR(-20004, 'Email ekziston në sistem.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20005, 'Gabim gjatë përditësimit të
klientit: ' || SQLERRM);
END sp_perditeso_klient;
/
```

Përshkrimi: Përditëson informacionet e një klienti ekzistues. Lejon përditësim selektiv të fushave.

sp_fshij_klient

```
CREATE OR REPLACE PROCEDURE sp_fshij_klient(
    p_klient_id IN NUMBER
) AS
BEGIN
    UPDATE KLIENTI
```

```
    SET STATUS = 'JOAKTIV'
        WHERE KLIENT_ID = p_klient_id;

    IF SQL%ROWCOUNT = 0 THEN
        RAISE_APPLICATION_ERROR(-20003, 'Klienti nuk u gjet.');
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20005, 'Gabim gjatë fshirjes së klientit: '
|| SQLERRM);
END sp_fshij_klient;
/
```

Përshkrimi: "Fshin" një klient duke e shënuar si JOAKTIV në vend të fshirjes fizike nga baza e të dhënave.

3.2 Procedurat për Mjetet

sp_regjistro_mjet

```
CREATE OR REPLACE PROCEDURE sp_regjistro_mjet(
    p_klient_id IN NUMBER,
    p_targa IN VARCHAR2,
    p_marka IN VARCHAR2,
    p_viti_prodhimit IN NUMBER
) AS
    v_klient NUMBER;
    v_targa NUMBER;
BEGIN

    v_klient := f_klienti_ekziston(p_klient_id);
    v_targa := f_mjeti_ekziston(p_targa);

    DBMS_OUTPUT.PUT('Klienti ekziston: ' || v_klient || ', Mjeti me këtë
targë ekziston: ' || v_targa);

    IF v_klient = 0 OR v_targa = 1 THEN
        RAISE_APPLICATION_ERROR(-20006, 'Gabmin në regjistrimin e mjetit:
Klienti nuk ekziston ose mjeti me këtë targë ekziston tashmë.');
    END IF;

    INSERT INTO MJETI(KLIENT_ID, TARGA, MARKA, VITI_PRODHIMIT)
    VALUES (p_klient_id, p_targa, p_marka, p_viti_prodhimit);
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20006, 'Gabim gjatë regjistrimit të mjetit:
' || SQLERRM);
END sp_regjistro_mjet;
/
```

Përshkrimi: Regjistron një mjet të ri për një klient specifik. Verifikon nëse klienti ekziston dhe nëse targa është unike.

sp_perditeso_mjet

```
CREATE OR REPLACE PROCEDURE sp_perditeso_mjet(
    p_mjeti_id IN NUMBER,
    p_targa IN VARCHAR2 DEFAULT NULL,
    p_marka IN VARCHAR2 DEFAULT NULL,
    p_viti_prodhimit IN NUMBER DEFAULT NULL
) AS
    v_count NUMBER;
BEGIN
    v_count := f_mjeti_ekziston(p_targa);
    IF v_count = 1 THEN
        RAISE_APPLICATION_ERROR(-20007, 'Mjeti me këtë targë ekziston
tashmë.');
    END IF;

    UPDATE MJETI
    SET TARGA          = COALESCE(p_targa, TARGA),
        MARKA          = COALESCE(p_marka, MARKA),
        VITI_PRODHIMIT = COALESCE(p_viti_prodhimit, VITI_PRODHIMIT)
    WHERE MJETI_ID = p_mjeti_id;

    IF SQL%ROWCOUNT = 0 THEN
        RAISE_APPLICATION_ERROR(-20007, 'Mjeti nuk u gjet.');
    END IF;
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        RAISE_APPLICATION_ERROR(-20008, 'Targa ekziston në sistem.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20009, 'Gabim gjatë përditësimit të mjetit:
' || SQLERRM);
END sp_perditeso_mjet;
/
```

Përshkrimi: Përditëson informacionet e një mjeti ekzistues.

3.3 Procedurat për Kartat e Anëtarësisë

sp_krijo_karte_anetaresie

```
CREATE OR REPLACE PROCEDURE sp_krijo_karte_anetaresie(
    p_kodi IN VARCHAR2,
    p_lloji IN VARCHAR2 DEFAULT 'STANDARD'
) IS
BEGIN
    INSERT INTO KARTE_ANETARESIE(KODI, LLOJI) VALUES (p_kodi, p_lloji);
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        RAISE_APPLICATION_ERROR(-20004, 'Kodi ekziston në sistem.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20005, 'Gabim gjatë krijimit të kartës: ' ||
SQLERRM);
END sp_krijo_karte_anetaresie;
/
```

Përshkrimi: Krijon një kartë anëtarësie të re me kod unik dhe lloj të specifikuar.

sp_lesho_karte_anetaresie

```
CREATE OR REPLACE PROCEDURE sp_lesho_karte_anetaresie(
    p_punonjesi_id IN NUMBER,
    p_klient_id IN NUMBER,
    p_karte_id IN NUMBER,
    p_data_skadimit IN DATE
) AS
    v_lloji          VARCHAR2(20);
    v_cmimi_premium  NUMBER := 2000;
    v_cmimi_standard NUMBER := 500;
    v_fatura_id      NUMBER;
BEGIN
    SELECT LLOJI INTO v_lloji FROM KARTE_ANETARESIE WHERE KARTE_ID =
p_karte_id;

    INSERT INTO HISTORIK_KARTE_ANETARESIE(KLIENT_ID, KARTE_ID,
DATA_E_LESHIMIT, DATA_E_SKADIMIT, STATUS)
    VALUES (p_klient_id, p_karte_id, SYSDATE, p_data_skadimit, 'AKTIV');

    IF v_lloji = 'PREMIUM' THEN
        INSERT INTO FATURA(PUNONJESI_ID, KLIENT_ID, VLERA, TARGA,
DATA_E_LESHIMIT, DATA_E_PAGESËS, PËRSHKRIMI)
        VALUES (p_punonjesi_id, p_klient_id, v_cmimi_premium, NULL, SYSDATE,
SYSDATE,
                'Lëshim kartë anëtarësie PREMIUM për klientin: ' ||
p_klient_id)
        RETURNING FATURA_ID INTO v_fatura_id;
    ELSE
        INSERT INTO FATURA(PUNONJESI_ID, KLIENT_ID, VLERA, TARGA,
DATA_E_LESHIMIT, DATA_E_PAGESËS, PËRSHKRIMI)
        VALUES (p_punonjesi_id, p_klient_id, v_cmimi_standard, NULL, SYSDATE,
SYSDATE,
                'Lëshim kartë anëtarësie STANDARD për klientin: ' ||
p_klient_id)
        RETURNING FATURA_ID INTO v_fatura_id;
    END IF;
    INSERT INTO SHITJET(FATURA_ID, STATUS) VALUES (v_fatura_id,
'PERFUNDUAR');
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20005, 'Gabim gjatë lëshimit të kartës: ' ||
SQLERRM);
END sp_lesho_karte_anetaresie;
/
```

Përshkrimi: Lëshon një kartë anëtarësie për një klient, gjeneron faturën përkatëse dhe kryen transaksionin e shitjes.

sp_çaktivizo_karte_anetaresie

```
CREATE OR REPLACE PROCEDURE sp_çaktivizo_karte_anetaresie(
    p_klient_id IN NUMBER,
    p_karte_id IN NUMBER
) AS
BEGIN
    UPDATE HISTORIK_KARTE_ANETARESIE
    SET STATUS = 'JOAKTIV'
    WHERE KLIENT_ID = p_klient_id
      AND KARTE_ID = p_karte_id
      AND STATUS = 'AKTIV';

    IF SQL%ROWCOUNT = 0 THEN
        RAISE_APPLICATION_ERROR(-20031, 'Kartë aktive nuk u gjet për këtë
klient.');
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20032, 'Gabim gjatë çaktivizimit të kartës:
' || SQLERRM);
END sp_çaktivizo_karte_anetaresie;
/
```

Përshkrimi: Çaktivizon një kartë anëtarësie aktive për një klient specifik.

sp_kontrollo_skadimin_e_kartave

```
CREATE OR REPLACE PROCEDURE sp_kontrollo_skadimin_e_kartave
    IS
BEGIN
    UPDATE HISTORIK_KARTE_ANETARESIE
    SET STATUS = 'SKADUAR'
    WHERE STATUS = 'AKTIV'
      AND DATA_E_SKADIMIT < SYSDATE;
    COMMIT;
END sp_kontrollo_skadimin_e_kartave;
/
```

Përshkrimi: Kontrollon dhe përditëson statusin e kartave të anëtarësisë të skaduara.

3.4 Procedurat për Punonjësit

sp_shto_punonjes

```
CREATE OR REPLACE PROCEDURE sp_shto_punonjes(
    p_emri IN VARCHAR2,
    p_mbiemri IN VARCHAR2,
    p_nr_kontakti IN VARCHAR2,
    p_email IN VARCHAR2,
    p_pike_id IN VARCHAR2,
    p_roli IN VARCHAR2 DEFAULT 'PUNONJESI'
) AS
BEGIN
    INSERT INTO PUNONJESI(PIKE_ID, EMRI, MBIEMRI, NR_KONTAKTI, E_MAIL,ROLI)
    VALUES (p_pike_id, p_emri, p_mbiemri, p_nr_kontakti, p_email,p_roli);
```

```
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        RAISE_APPLICATION_ERROR(-20021, 'Email ekziston');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20022, 'Gabmin në shtimin e punonjësit: ' ||
SQLERRM);
END sp_shto_punonjes;
/
```

Përshkrimi: Shton një punonjës të ri në sistem me rol dhe pikë pune të specifikuar.

sp_perditeso_punonjes

```
CREATE OR REPLACE PROCEDURE sp_perditeso_punonjes(
    p_punonjesi_id IN NUMBER,
    p_emri IN VARCHAR2 DEFAULT NULL,
    p_mbiemri IN VARCHAR2 DEFAULT NULL,
    p_nr_kontakti IN VARCHAR2 DEFAULT NULL,
    p_email IN VARCHAR2 DEFAULT NULL,
    p_pike_id IN VARCHAR2 DEFAULT NULL,
    p_roli IN VARCHAR2 DEFAULT 'PUNONJESI'
) AS
    v_count NUMBER;
BEGIN
    v_count := f_punonjesi_ekziston(p_punonjesi_id);
    IF v_count = 0 THEN
        RAISE_APPLICATION_ERROR(-20020, 'Punonjësi nuk ekziston.');
    END IF;

    UPDATE PUNONJESI
    SET EMRI        = COALESCE(p_emri, EMRI),
        MBIEMRI     = COALESCE(p_mbiemri, MBIEMRI),
        NR_KONTAKTI = COALESCE(p_nr_kontakti, NR_KONTAKTI),
        E_MAIL      = COALESCE(p_email, E_MAIL),
        PIKE_ID     = COALESCE(p_pike_id, PIKE_ID),
        ROLI        = p_roli
    WHERE PUNONJESI_ID = p_punonjesi_id;
    COMMIT;
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        ROLLBACK;
        RAISE_APPLICATION_ERROR(-20021, 'Email ekziston në sistem.');
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE_APPLICATION_ERROR(-20022, 'Gabim gjatë përditësimit të
punonjësit: ' || SQLERRM);
END sp_perditeso_punonjes;
/
```

Përshkrimi: Përditëson informacionet e një punonjësi ekzistues.

3.5 Procedurat për Pikat e Parkimit

sp_shto_pike_parkimi

```
CREATE OR REPLACE PROCEDURE sp_shto_pike_parkimi(
    p_pike_id IN VARCHAR2,
    p_vendodhja IN VARCHAR2,
    p_kapaciteti IN NUMBER,
    p_orari_i_hapjes IN VARCHAR2,
    p_orari_i_mbylljes IN VARCHAR2
) AS
BEGIN
    INSERT INTO PIKE_PARKIMI(PIKE_ID, VENDODHJA, KAPACITETI, ORARI_I_HAPJES,
ORARI_I_MBYLLJES)
    VALUES (p_pike_id, p_vendodhja, p_kapaciteti, p_orari_i_hapjes,
p_orari_i_mbylljes);
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        RAISE_APPLICATION_ERROR(-20021, 'Pika e parkimit ekziston');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20022, 'Gabmin në shtimin e pikës së
parkimit: ' || SQLERRM);
END sp_shto_pike_parkimi;
/
```

Përshkrimi: Shton një pikë parkimi të re me të gjitha specifikimet e nevojshme.

sp_perditeso_pike_parkimi

```
CREATE OR REPLACE PROCEDURE sp_perditeso_pike_parkimi(
    p_pike_id IN VARCHAR2,
    p_vendodhja IN VARCHAR2 DEFAULT NULL,
    p_kapaciteti IN NUMBER DEFAULT NULL,
    p_orari_i_hapjes IN VARCHAR2 DEFAULT NULL,
    p_orari_i_mbylljes IN VARCHAR2 DEFAULT NULL
) AS
BEGIN
    UPDATE PIKE_PARKIMI
    SET VENDODHJA        = COALESCE(p_vendodhja, VENDODHJA),
        KAPACITETI       = COALESCE(p_kapaciteti, KAPACITETI),
        ORARI_I_HAPJES   = COALESCE(p_orari_i_hapjes, ORARI_I_HAPJES),
        ORARI_I_MBYLLJES = COALESCE(p_orari_i_mbylljes, ORARI_I_MBYLLJES)
    WHERE PIKE_ID = p_pike_id;

    IF SQL%ROWCOUNT = 0 THEN
        RAISE_APPLICATION_ERROR(-20040, 'Pika e parkimit nuk u gjet.');
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20041, 'Gabim gjatë përditësimit të pikës së
parkimit: ' || SQLERRM);
END sp_perditeso_pike_parkimi;
/
```

Përshkrimi: Përditëson informacionet e një pike parkimi ekzistuese.

3.6 Procedurat për Çmimet e Parkimit

sp_shto_cmime_pike_parkimi

```
CREATE OR REPLACE PROCEDURE sp_shto_cmime_pike_parkimi(
    p_pike_id IN VARCHAR2,
    p_cmimi IN NUMBER,
    p_ora_min IN NUMBER,
    p_ora_max IN NUMBER
) AS
BEGIN
    INSERT INTO CMIME_PARKIMI(PIKE_ID, CMIMI, ORA_MIN, ORA_MAX)
    VALUES (p_pike_id, p_cmimi, p_ora_min, p_ora_max);
EXCEPTION
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20024, 'Gabim gjatë shtimit të cmimit të
pikës së parkimit: ' || SQLERRM);
END sp_shto_cmime_pike_parkimi;
/
```

Përshkrimi: Shton një fashë çmimi për një pikë parkimi specifike.

sp_ndrysho_cmime_pike_parkimi

```
CREATE OR REPLACE PROCEDURE sp_ndrysho_cmime_pike_parkimi(
    p_pike_id IN VARCHAR2,
    p_cmimi IN NUMBER,
    p_ora_min IN NUMBER,
    p_ora_max IN NUMBER
) AS
BEGIN
    UPDATE CMIME_PARKIMI
    SET CMIMI   = p_cmimi,
        ORA_MIN = p_ora_min,
        ORA_MAX = p_ora_max
    WHERE PIKE_ID = p_pike_id;
END sp_ndrysho_cmime_pike_parkimi;
/
```

Përshkrimi: Ndryshon çmimet e një pike parkimi për një fashë orare specifike.

3.7 Procedurat për Aktivitetin e Parkimit

sp_fillo_parkimin_e_mjetit

```
CREATE OR REPLACE PROCEDURE sp_fillo_parkimin_e_mjetit(
    p_mjeti_id IN NUMBER DEFAULT NULL,
    p_targa_temp IN VARCHAR2 DEFAULT NULL,
    p_koha_hyrjes IN DATE DEFAULT SYSDATE,
    p_koha_daljes IN DATE DEFAULT NULL,
    p_pike_id IN VARCHAR2
) AS
    v_status VARCHAR2(10);
BEGIN
    v_status := F_VALIDO_ORARIN_PARKIMIT(p_pike_id);
```

```
    IF v_status = 'HAPUR' THEN
        INSERT INTO AKTIVITET_PARKIMI(PIKE_ID, MJETI_ID, TARGA_TEMP,
KOHA_HYRJES, KOHA_DALJES)
        VALUES (p_pike_id, p_mjeti_id, p_targa_temp, p_koha_hyrjes,
p_koha_daljes);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Parkimi është i mbyllur në këtë orë.');
    END IF;
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE_APPLICATION_ERROR(-20009, 'Gabim gjatë fillimit të aktivitetit
të parkimit: ' || SQLERRM);
END sp_fillo_parkimin_e_mjetit;
/
```

Përshkrimi: Fillon aktivitetin e parkimit për një mjet. Mbështet si mjete të regjistruara ashtu edhe ato me targa të përkohshme.

sp_mbaro_parkimin_e_mjetit

```
CREATE OR REPLACE PROCEDURE sp_mbaro_parkimin_e_mjetit(
    p_aktivitet_id IN NUMBER,
    p_punonjes_id IN NUMBER
) AS
    v_pike_sakte   VARCHAR2(10);
    v_koha_hyrjes  DATE;
    v_koha_daljes  DATE   := SYSDATE;
    v_targa        VARCHAR2(10);
    v_pike_id      VARCHAR2(10);
    v_mjeti_id     NUMBER;
    v_vlera        NUMBER := 0;
    v_data_pagese  DATE;
    v_klient_id    NUMBER;
    v_abonim_aktiv NUMBER := 0;
    v_fatura_id    NUMBER;
BEGIN

    SELECT KOHA_HYRJES, PIKE_ID, MJETI_ID, TARGA_TEMP
    INTO v_koha_hyrjes, v_pike_id, v_mjeti_id, v_targa
    FROM AKTIVITET_PARKIMI
    WHERE AKTIVITET_PARKIMI_ID = p_aktivitet_id
      AND KOHA_DALJES IS NULL;

    SELECT PIKE_ID into v_pike_sakte FROM PUNONJESI WHERE PUNONJESI_ID =
p_punonjes_id;

    IF v_pike_sakte != v_pike_id THEN
        RAISE_APPLICATION_ERROR(-20010, 'Pika e parkimit e punonjësit nuk
përputhet me atë të aktivitetit.');
    END IF;

    UPDATE AKTIVITET_PARKIMI
    SET KOHA_DALJES = v_koha_daljes
```

```
    WHERE AKTIVITET_PARKIMI_ID = p_aktivitet_id;

    IF v_mjeti_id IS NOT NULL THEN
        SELECT KLIENT_ID, TARGA
        INTO v_klient_id,v_targa
        FROM MJETI
        WHERE MJETI_ID = v_mjeti_id;

        SELECT COUNT(*)
        INTO v_abonim_aktiv
        FROM ABONIM
        WHERE KLIENT_ID = v_klient_id
          AND PIKE_ID = v_pike_id
          AND STATUS = 'AKTIV'
          AND DATA_E_SKADIMIT >= SYSDATE;
    END IF;

    IF v_abonim_aktiv = 0 THEN
        v_vlera := f_llogarit_cmimin(v_pike_id, v_koha_hyrjes,
v_koha_daljes);
    ELSE
        UPDATE ABONIM
        SET NR_HERESH = NR_HERESH - 1
        WHERE KLIENT_ID = v_klient_id
          AND PIKE_ID = v_pike_id
          AND STATUS = 'AKTIV'
          AND LLOJI = 'NR_HERESH'
          AND NR_HERESH > 0;
    END IF;

    IF v_vlera != 0 THEN
        v_data_pagese := v_koha_daljes;

        INSERT INTO FATURA(PUNONJESI_ID, KLIENT_ID, VLERA, TARGA,
DATA_E_LESHIMIT, DATA_E_PAGESËS, PËRSHKRIMI)
        VALUES (p_punonjes_id, v_klient_id, v_vlera, v_targa, v_koha_daljes,
v_data_pagese,
                'Pagesa për mjetin me targë: ' || v_targa)
        RETURNING FATURA_ID INTO v_fatura_id;

        INSERT INTO SHITJET(FATURA_ID, STATUS)
        VALUES (v_fatura_id, 'PERFUNDUAR');

    END IF;

    DBMS_OUTPUT.PUT_LINE('Mjeti doli nga parkimi');
    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        ROLLBACK;
        RAISE_APPLICATION_ERROR(-20010, 'Aktivitet parkimi nuk u gjet ose
është mbyllur tashmë.');
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE_APPLICATION_ERROR(-20011, 'Gabim gjatë mbylljes së aktivitetit
të parkimit: ' || SQLERRM);
```

```
END;
/
```

Përshkrimi: Mbaron aktivitetin e parkimit, llogarit çmimin, gjeneron faturën dhe kryen transaksionin e pagesës.

3.8 Procedurat për Abonimet

sp_krijo_abonim

```
CREATE OR REPLACE PROCEDURE sp_krijo_abonim(
    p_punonjesi_id IN NUMBER,
    p_klient_id IN NUMBER,
    p_pike_id IN VARCHAR2,
    p_lloji IN VARCHAR2,
    p_nr_muajsh IN NUMBER DEFAULT NULL,
    p_nr_heresh IN NUMBER DEFAULT NULL,
    p_data_e_leshimit IN DATE DEFAULT SYSDATE
) AS
    v_data_skadimit DATE;
    v_abonimi_id    NUMBER;
    v_cmimi_mujor   NUMBER := 5000;
    v_cmimi_heresh  NUMBER := 400;
    v_vlera         NUMBER;
    v_fatura_id     NUMBER;
BEGIN
    IF p_lloji = 'MUJOR' AND p_nr_muajsh IS NULL THEN
        RAISE_APPLICATION_ERROR(-20012, 'Numri i muajve duhet të specifikohet
për abonim mujor.');
    ELSIF p_lloji = 'NR_HERESH' AND p_nr_heresh IS NULL THEN
        RAISE_APPLICATION_ERROR(-20013, 'Numri i herëve duhet të specifikohet
për abonim me numër herësh.');
    END IF;

    IF p_lloji = 'MUJOR' THEN
        v_data_skadimit := ADD_MONTHS(SYSDATE, p_nr_muajsh);
        v_vlera := v_cmimi_mujor * p_nr_muajsh;
    ELSIF p_lloji = 'NR_HERESH' THEN
        v_data_skadimit := SYSDATE + p_nr_heresh;
        v_vlera := v_cmimi_heresh * p_nr_heresh;
    ELSE
        RAISE_APPLICATION_ERROR(-20014, 'Lloji i abonimit duhet përcaktuar');
    END IF;

    INSERT INTO ABONIM(KLIENT_ID, PIKE_ID, LLOJI, NR_MUAJSH, NR_HERESH,
DATA_E_LESHIMIT, DATA_E_SKADIMIT,
                       STATUS)
    VALUES (p_klient_id, p_pike_id, p_lloji, p_nr_muajsh, p_nr_heresh,
p_data_e_leshimit, v_data_skadimit,
            'AKTIV')
    RETURNING ABONIM_ID INTO v_abonimi_id;

    INSERT INTO FATURA(PUNONJESI_ID, KLIENT_ID, VLERA, TARGA,
DATA_E_LESHIMIT, DATA_E_PAGESËS, PËRSHKRIMI)
    VALUES (p_punonjesi_id, p_klient_id, v_vlera, NULL, v_data_skadimit,
```

```
SYSDATE,
            'Lëshim Abonimi për klientin: ' || p_klient_id)
    RETURNING FATURA_ID INTO v_fatura_id;
    INSERT INTO SHITJET(FATURA_ID, STATUS) VALUES (v_fatura_id,
'PERFUNDUAR');
EXCEPTION
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20015, 'Gabim gjatë krijimit të abonimit: '
|| SQLERRM);
END sp_krijo_abonim;
/
```

Përshkrimi: Krijon një abonim të ri për një klient, llogarit çmimin dhe gjeneron faturën e pagesës.

3.9 Procedurat për Shitjet

sp_anulo_shitje

```
CREATE OR REPLACE PROCEDURE sp_anulo_shitje(
    p_shitje_id IN NUMBER,
    p_punonjes_id IN NUMBER,
    p_arsyeja IN VARCHAR2
) AS
    v_fatura_id        NUMBER;
    v_klient_id        NUMBER;
    v_vlera            NUMBER;
    v_targa            VARCHAR2(10);
    v_fatura_anulim_id NUMBER;
    v_Përshkrimi VARCHAR2(255);
BEGIN

    SELECT f.FATURA_ID, f.VLERA, f.TARGA, f.KLIENT_ID,f.PËRSHKRIMI
    INTO v_fatura_id, v_vlera, v_targa,v_klient_id, v_Përshkrimi
    FROM SHITJET s
             JOIN FATURA f ON s.FATURA_ID = f.FATURA_ID
    WHERE s.SHITJET_ID = p_shitje_id
      AND s.STATUS = 'PERFUNDUAR';

    UPDATE SHITJET
    SET STATUS = 'ANULUAR'
    WHERE SHITJET_ID = p_shitje_id;

    IF v_klient_id IS NOT NULL AND v_targa IS NULL THEN
        IF INSTR(v_Përshkrimi, 'Lëshim kartë anëtarësie') > 0 THEN
            UPDATE HISTORIK_KARTE_ANETARESIE
            SET STATUS = 'JOAKTIV'
            WHERE KLIENT_ID = v_klient_id
              AND STATUS = 'AKTIV';

            IF SQL%ROWCOUNT = 0 THEN
                RAISE_APPLICATION_ERROR(-20020, 'Kartë anëtarësie nuk u gjet
ose është anuluar tashmë.');
            END IF;
        ELSE
```

```
            UPDATE ABONIM
            SET STATUS = 'JOAKTIV'
            WHERE KLIENT_ID = v_klient_id
              AND STATUS = 'AKTIV';
        END IF;
    END IF;

    INSERT INTO FATURA(PUNONJESI_ID,KLIENT_ID, VLERA, TARGA, DATA_E_LESHIMIT,
DATA_E_PAGESËS, PËRSHKRIMI)
    VALUES (p_punonjes_id,v_klient_id, -1 * v_vlera, v_targa, SYSDATE,
SYSDATE, p_arsyeja)
    RETURNING FATURA_ID INTO v_fatura_anulim_id;

    INSERT INTO SHITJET(FATURA_ID, STATUS)
    VALUES (v_fatura_anulim_id, 'PERFUNDUAR');

    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        ROLLBACK;
        RAISE_APPLICATION_ERROR(-20015, 'Shitja nuk u gjet ose është anuluar
tashmë.');
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE_APPLICATION_ERROR(-20016, 'Gabim gjatë anulimit të shitjes: '
|| SQLERRM);
END sp_anulo_shitje;
/
```

Përshkrimi: Anulon një shitje dhe gjeneron faturë me vlerë negative për rimbursim.

3.10 Procedurat për Aktivitetin Ditor

sp_mbyll_aktivitetin_ditor

```
CREATE OR REPLACE PROCEDURE sp_mbyll_aktivitetin_ditor(
    p_punonjes_id IN NUMBER,
    p_dita IN DATE DEFAULT TRUNC(SYSDATE)
) AS
    v_shuma NUMBER := 0;
    v_count NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO v_count
    FROM AKTIVITETI_DITOR
    WHERE PUNONJESI_ID = p_punonjes_id
      AND DITA = TRUNC(p_dita);

    IF v_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20017, 'Aktiviteti ditor për këtë punonjës
dhe datë është mbyllur tashmë.');
    END IF;

    SELECT SUM(VLERA)
    INTO v_shuma
```

```
    FROM FATURA
    WHERE PUNONJESI_ID = p_punonjes_id
      AND TRUNC(DATA_E_LESHIMIT) = TRUNC(p_dita);

    DBMS_OUTPUT.PUT_LINE('Shuma totale për punonjësin ' || p_punonjes_id || '
për datën ' || TO_CHAR(TRUNC(p_dita), 'DD-MM-YYYY') || ' është: ' ||
v_shuma);
    INSERT INTO AKTIVITETI_DITOR(PUNONJESI_ID, SHUMA, DITA)
    VALUES (p_punonjes_id, v_shuma, TRUNC(p_dita));

EXCEPTION
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20018, 'Gabim gjatë mbylljes së aktivitetit
ditor: ' || SQLERRM);
END sp_mbyll_aktivitetin_ditor;
/
```

Përshkrimi: Mbyll aktivitetin ditor të një punonjësi duke llogaritur shumën totale të arkëtuar për ditën.

4. TRIGGER-AT

4.1 trg_kontrollo_parkim_dublikat

```
CREATE OR REPLACE TRIGGER trg_kontrollo_parkim_dublikat
    BEFORE INSERT
    ON AKTIVITET_PARKIMI
    FOR EACH ROW
DECLARE
    v_count   NUMBER;
    v_te_lira NUMBER;
BEGIN

    IF :NEW.MJETI_ID IS NOT NULL THEN
        SELECT COUNT(*)
        INTO v_count
        FROM AKTIVITET_PARKIMI
        WHERE MJETI_ID = :NEW.MJETI_ID
          AND (KOHA_DALJES IS NULL OR KOHA_DALJES > SYSDATE);

        IF v_count > 0 THEN
            RAISE_APPLICATION_ERROR(-20021, 'Mjeti është aktualisht i
parkuar.');
        END IF;
    END IF;

    IF :NEW.TARGA_TEMP IS NOT NULL THEN
        SELECT COUNT(*)
        INTO v_count
        FROM AKTIVITET_PARKIMI
        WHERE (TARGA_TEMP = :NEW.TARGA_TEMP OR
               MJETI_ID IN (SELECT MJETI_ID FROM MJETI WHERE TARGA =
:NEW.TARGA_TEMP))
          AND (KOHA_DALJES IS NULL OR KOHA_DALJES > SYSDATE);
```

```
        IF v_count > 0 THEN
            RAISE_APPLICATION_ERROR(-20022, 'Mjeti është aktualisht i
parkuar.');
        END IF;
    END IF;

    v_te_lira := f_vende_te_lira(:NEW.PIKE_ID);

    IF v_te_lira <= 0 THEN
        RAISE_APPLICATION_ERROR(-20023, 'Nuk ka vende të lira në këtë pikë
parkimi.');
    END IF;
END;
/
```

Përshkrimi: Parandalon parkimin e dublikuar të të njëjtit mjet në pika të ndryshme parkimi në të njëjtën kohë. Kontrollon gjithashtu disponueshmërinë e vendeve të lira.

4.2 trg_shto_historik_karte

```
CREATE OR REPLACE TRIGGER trg_shto_historik_karte
    BEFORE INSERT
    ON HISTORIK_KARTE_ANETARESIE
    FOR EACH ROW
DECLARE
    v_lloji VARCHAR2(10);
BEGIN
    IF :NEW.KARTE_ID IS NOT NULL THEN
        SELECT LLOJI INTO v_lloji FROM KARTE_ANETARESIE WHERE
KARTE_ANETARESIE.KARTE_ID = :NEW.KARTE_ID;
        IF v_lloji = 'STANDARD' THEN
            :NEW.DATA_E_SKADIMIT := ADD_MONTHS(SYSDATE, 2);
        ELSE
            :NEW.DATA_E_SKADIMIT := ADD_MONTHS(SYSDATE, 12);
        END IF;
    END IF;
END;
/
```

Përshkrimi: Automatikisht përcakton datën e skadimit të kartës së anëtarësisë bazuar në llojin e kartës (2 muaj për STANDARD, 12 muaj për PREMIUM).

4.3 trg_ndalim_modif_fshirje_aktviteti

```
CREATE OR REPLACE TRIGGER trg_ndalim_modif_fshirje_aktviteti
    BEFORE UPDATE OR DELETE
    ON AKTIVITET_PARKIMI
    FOR EACH ROW
BEGIN
    IF :OLD.KOHA_DALJES IS NOT NULL AND (:OLD.AKTIVITET_PARKIMI_ID !=
:NEW.AKTIVITET_PARKIMI_ID OR DELETING) THEN
        RAISE_APPLICATION_ERROR(-20024, 'Nuk mund të modifikohet ose fshihet
një aktivitet i përfunduar parkimi.');
```

```
    END IF;
END;
/
```

Përshkrimi: Parandalon modifikimin ose fshirjen e aktiviteteve të parkimit që janë përfunduar (kanë koha_daljes të plotësuar).

4.4 trg_ndalim_shitje_mjet_parkuar

```
CREATE OR REPLACE TRIGGER trg_ndalim_shitje_mjet_parkuar
    BEFORE INSERT OR UPDATE
    ON FATURA
    FOR EACH ROW
DECLARE
    v_count NUMBER := 0;
    v_mjet_id NUMBER;
    v_pike_parkuar VARCHAR2(10);
BEGIN
    IF :NEW.TARGA IS NOT NULL THEN

        SELECT MJETI_ID INTO v_mjet_id FROM MJETI WHERE TARGA = :NEW.TARGA;

        SELECT COUNT(*), MAX(PIKE_ID)
        INTO v_count, v_pike_parkuar
        FROM AKTIVITET_PARKIMI ap
                 LEFT JOIN MJETI m ON ap.MJETI_ID = m.MJETI_ID
        WHERE ((m.MJETI_ID = v_mjet_id AND TARGA = :NEW.TARGA) OR
(ap.TARGA_TEMP = :NEW.TARGA))
          AND (ap.KOHA_DALJES IS NULL OR ap.KOHA_DALJES > SYSDATE);

        IF v_count > 0 AND INSTR(UPPER(:NEW.PËRSHKRIMI), 'PAGESA PËR MJETIN')
= 0 THEN
            RAISE_APPLICATION_ERROR(-20070,
                                    'Nuk mund të bëhet shitje për mjetin me
targë ' || :NEW.TARGA ||
                                    ' sepse është aktualisht i parkuar në
pikën ' || v_pike_parkuar ||
                                    '. Duhet të përfundojë parkimi para se të
bëhet shitje.');
        END IF;
    END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        NULL;
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20071, 'Gabim gjatë kontrollimit të statusit
të parkimit: ' || SQLERRM);
END;
/
```

Përshkrimi: Parandalon krijimin e faturave për mjete që janë aktualisht të parkuara, përveç atyre që janë për pagesën e parkimit.

5. SISTEMI I SIGURISË (ROLE-ET)

5.1 Punonjesi Parkimi

```
CREATE ROLE punonjesi_parkimi;


GRANT EXECUTE ON f_vende_te_lira TO punonjesi_parkimi;
GRANT EXECUTE ON f_punonjesi_ekziston TO punonjesi_parkimi;
GRANT EXECUTE ON f_klienti_ekziston TO punonjesi_parkimi;
GRANT EXECUTE ON f_mjeti_ekziston TO punonjesi_parkimi;
GRANT EXECUTE ON f_llogarit_cmimin TO punonjesi_parkimi;
GRANT EXECUTE ON f_valido_orarin_parkimit TO punonjesi_parkimi;

GRANT EXECUTE ON sp_anulo_shitje TO punonjesi_parkimi;
GRANT EXECUTE ON sp_fillo_parkimin_e_mjetit TO punonjesi_parkimi;
GRANT EXECUTE ON sp_fshij_klient TO punonjesi_parkimi;
GRANT EXECUTE ON sp_kontrollo_skadimin_e_kartave TO punonjesi_parkimi;
GRANT EXECUTE ON sp_lesho_karte_anetaresie TO punonjesi_parkimi;
GRANT EXECUTE ON sp_perditeso_klient TO punonjesi_parkimi;
GRANT EXECUTE ON sp_perditeso_mjet TO punonjesi_parkimi;
GRANT EXECUTE ON sp_çaktivizo_karte_anetaresie TO punonjesi_parkimi;
GRANT EXECUTE ON sp_mbaro_parkimin_e_mjetit TO punonjesi_parkimi;
GRANT EXECUTE ON sp_mbyll_aktivitetin_ditor TO punonjesi_parkimi;
GRANT EXECUTE ON sp_anulo_shitje TO punonjesi_parkimi;
GRANT EXECUTE ON sp_regjistro_klient TO punonjesi_parkimi;
GRANT EXECUTE ON sp_regjistro_mjet TO punonjesi_parkimi;


GRANT SELECT ON PIKE_PARKIMI TO punonjesi_parkimi;
GRANT SELECT ON CMIME_PARKIMI TO punonjesi_parkimi;
GRANT SELECT ON AKTIVITET_PARKIMI TO punonjesi_parkimi;
GRANT SELECT ON MJETI TO punonjesi_parkimi;
GRANT SELECT ON KLIENTI TO punonjesi_parkimi;
GRANT SELECT ON ABONIM TO punonjesi_parkimi;
GRANT SELECT ON PUNONJESI TO punonjesi_parkimi;
GRANT SELECT ON FATURA TO punonjesi_parkimi;
GRANT SELECT ON SHITJET TO punonjesi_parkimi;
GRANT SELECT ON AKTIVITETI_DITOR TO punonjesi_parkimi;
```

Përshkrimi: Rol i cili lejon punonjësit e parkimit të kryejnë operacionet ditore si fillimi/mbarimi i parkimit, regjistrimi i klientëve të rinj dhe mbyllja e aktivitetit ditor etj.

5.2 Administratori

```
CREATE ROLE administratori;


GRANT EXECUTE ON f_vende_te_lira TO administratori;
GRANT EXECUTE ON f_punonjesi_ekziston TO administratori;
GRANT EXECUTE ON f_klienti_ekziston TO administratori;
GRANT EXECUTE ON f_mjeti_ekziston TO administratori;
GRANT EXECUTE ON f_llogarit_cmimin TO administratori;
GRANT EXECUTE ON f_valido_orarin_parkimit TO administratori;
```

```
GRANT EXECUTE ON sp_regjistro_klient TO administratori;
GRANT EXECUTE ON sp_perditeso_klient TO administratori;
GRANT EXECUTE ON sp_fshij_klient TO administratori;
GRANT EXECUTE ON sp_regjistro_mjet TO administratori;
GRANT EXECUTE ON sp_perditeso_mjet TO administratori;
GRANT EXECUTE ON sp_krijo_karte_anetaresie TO administratori;
GRANT EXECUTE ON sp_lesho_karte_anetaresie TO administratori;
GRANT EXECUTE ON sp_çaktivizo_karte_anetaresie TO administratori;
GRANT EXECUTE ON sp_kontrollo_skadimin_e_kartave TO administratori;
GRANT EXECUTE ON sp_shto_punonjes TO administratori;
GRANT EXECUTE ON sp_perditeso_punonjes TO administratori;
GRANT EXECUTE ON sp_shto_pike_parkimi TO administratori;
GRANT EXECUTE ON sp_perditeso_pike_parkimi TO administratori;
GRANT EXECUTE ON sp_shto_cmime_pike_parkimi TO administratori;
GRANT EXECUTE ON sp_ndrysho_cmime_pike_parkimi TO administratori;
GRANT EXECUTE ON sp_fillo_parkimin_e_mjetit TO administratori;
GRANT EXECUTE ON sp_mbaro_parkimin_e_mjetit TO administratori;
GRANT EXECUTE ON sp_krijo_abonim TO administratori;
GRANT EXECUTE ON sp_anulo_shitje TO administratori;
GRANT EXECUTE ON sp_mbyll_aktivitetin_ditor TO administratori;

GRANT SELECT, INSERT, UPDATE, DELETE ON KLIENTI TO administratori;
GRANT SELECT, INSERT, UPDATE, DELETE ON MJETI TO administratori;
GRANT SELECT, INSERT, UPDATE, DELETE ON KARTE_ANETARESIE TO administratori;
GRANT SELECT, INSERT, UPDATE, DELETE ON HISTORIK_KARTE_ANETARESIE TO
administratori;
GRANT SELECT, INSERT, UPDATE, DELETE ON PIKE_PARKIMI TO administratori;
GRANT SELECT, INSERT, UPDATE, DELETE ON CMIME_PARKIMI TO administratori;
GRANT SELECT, INSERT, UPDATE, DELETE ON PUNONJESI TO administratori;
GRANT SELECT, INSERT, UPDATE, DELETE ON ABONIM TO administratori;
GRANT SELECT, INSERT, UPDATE, DELETE ON AKTIVITET_PARKIMI TO administratori;
GRANT SELECT, INSERT, UPDATE, DELETE ON FATURA TO administratori;
GRANT SELECT, INSERT, UPDATE, DELETE ON SHITJET TO administratori;
GRANT SELECT, INSERT, UPDATE, DELETE ON AKTIVITETI_DITOR TO administratori;
```

Përshkrimi: Rol me qasje të plotë në sistem për menaxhimin e klientëve, mjeteve, punonjësve, pikave të parkimit, çmimeve dhe gjenerimin e raporteve.

Kemi 2 View për punonjësin:

```
CREATE OR REPLACE VIEW v_aktiviteti_punonjesit AS
SELECT ad.*,
       p.EMRI,
       p.MBIEMRI,
       p.PIKE_ID
FROM AKTIVITETI_DITOR ad
         JOIN PUNONJESI p ON ad.PUNONJESI_ID = p.PUNONJESI_ID
WHERE UPPER(p.EMRI || p.MBIEMRI) = SYS_CONTEXT('USERENV', 'SESSION_USER');

CREATE OR REPLACE VIEW v_fatura_pike_punonjesit AS
SELECT f.*, p.PIKE_ID
FROM FATURA f
         JOIN PUNONJESI p ON f.PUNONJESI_ID = p.PUNONJESI_ID
WHERE UPPER(p.EMRI || p.MBIEMRI) = SYS_CONTEXT('USERENV', 'SESSION_USER');
```

```
GRANT SELECT ON v_aktiviteti_punonjesit TO punonjesi_parkimi;
GRANT SELECT ON v_fatura_pike_punonjesit TO punonjesi_parkimi;
```

Përshkrimi: Paraqitje e të dhënave që lidhen me punonjësit.

5.3 Kontrollori

```
CREATE ROLE kontrollori;

GRANT EXECUTE ON f_vende_te_lira TO kontrollori;
GRANT EXECUTE ON sp_kontrollo_skadimin_e_kartave TO kontrollori;

GRANT SELECT ON KLIENTI TO kontrollori;
GRANT SELECT ON MJETI TO kontrollori;
GRANT SELECT ON KARTE_ANETARESIE TO kontrollori;
GRANT SELECT ON HISTORIK_KARTE_ANETARESIE TO kontrollori;
GRANT SELECT ON PIKE_PARKIMI TO kontrollori;
GRANT SELECT ON CMIME_PARKIMI TO kontrollori;
GRANT SELECT ON PUNONJESI TO kontrollori;
GRANT SELECT ON ABONIM TO kontrollori;
GRANT SELECT ON AKTIVITET_PARKIMI TO kontrollori;
GRANT SELECT ON FATURA TO kontrollori;
GRANT SELECT ON SHITJET TO kontrollori;
GRANT SELECT ON AKTIVITETI_DITOR TO kontrollori;
```

Përshkrimi: Rol vetëm për lexim për auditim, monitorim dhe gjenerimin e raporteve analitike.

Shembull i krijimit të users dhe dhënia e roleve:

```
CREATE USER PiroGjikdhima IDENTIFIED BY p1;
CREATE USER AltinHana IDENTIFIED BY k1;
CREATE USER MetonVoku IDENTIFIED BY a1;

GRANT CREATE SESSION TO PiroGjikdhima;
GRANT CREATE SESSION TO AltinHana;
GRANT CREATE SESSION TO MetonVoku;

GRANT punonjesi_parkimi TO PiroGjikdhima;
GRANT kontrollori TO AltinHana;
GRANT administratori TO MetonVoku;
```