**A black and white sign with a black and white emblem

Description automatically generatedDetyrë 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.

**A computer screen shot of a diagram

AI-generated content may be incorrect.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 llogarritur 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;