

Proiect Securitatea Bazelor de Date

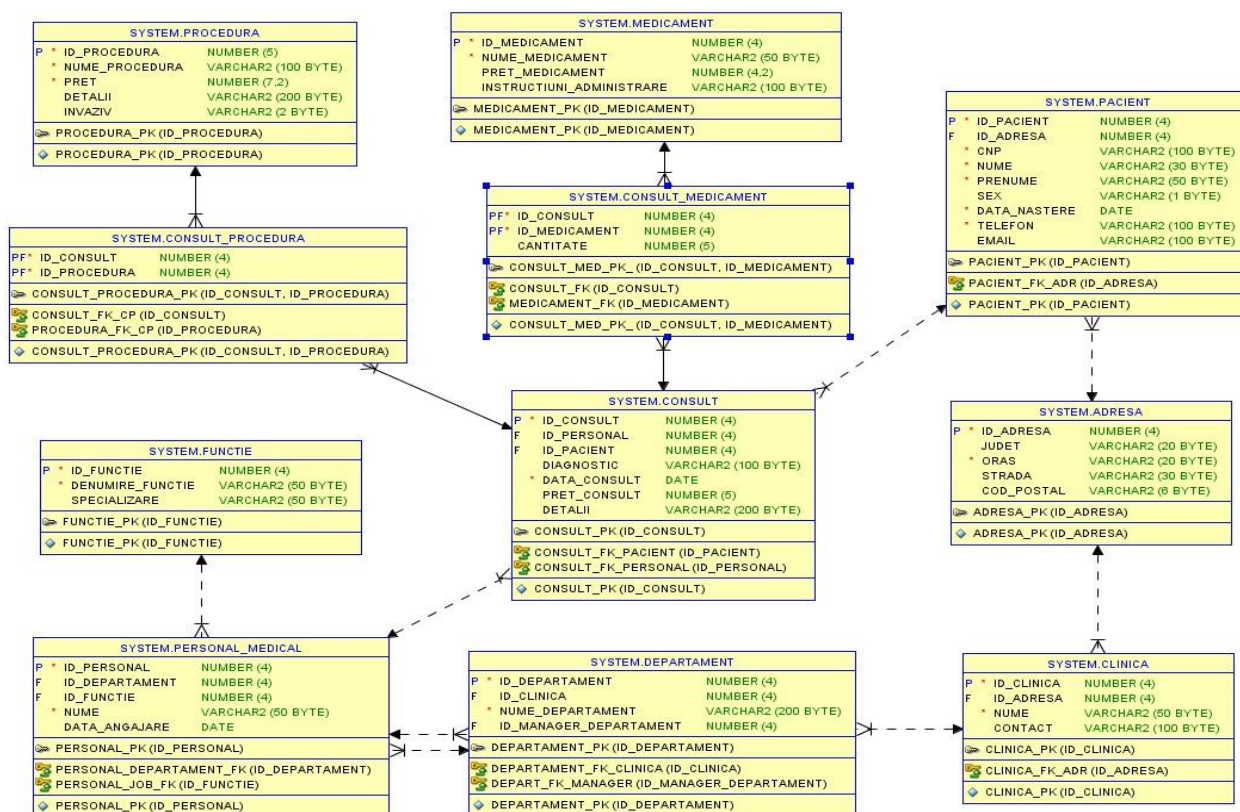
1. Introducere

a. Prezentarea succintă a modelului proiectat și a regulilor sale

În acest proiect este implementată o bază de date relațională ce are ca scop gestionarea consulturilor medicale ale pacienților în diferite clinici medicale din țară. Fiecare clinică aflată la o anumită adresă este împărțită pe departamente în funcție de ariile de specializare medicală, iar în fiecare departament lucrează personalul medical calificat deținând funcția corespunzătoare. În cadrul unui consult, pacientul este văzut de un doctor sau de o asistentă și i se pot efectua anumite proceduri medicale care să ajute la punerea unui diagnostic. Dacă este necesar, în urma consultului medicul va prescrie pacientului medicamente pentru tratament.

Baza de date este formată din 11 tabele, 9 entități independente și 2 tabele asociative pentru a rezolva relații many-to-many.

b. Diagrama conceptuală



c. Schemele relaționale

ADRESA(id_adresa, judet, oras, strada, cod_postal)

PACIENT(id_pacient, id_adresa, cnp, nume, prenume, sex, data_nastere, telefon, email)

CLINICA(id_clinica, id_adresa, nume, contact)

MEDICAMENT(id_medicament, nume_medicament, pret_medicament, instructiuni_administrare)

FUNCTIE(id_functie, denumire_functie, specializare)

DEPARTAMENT(id_departament, id_clinica, nume_departament, id_manager_department)

PERSONAL_MEDICAL(id_personal, id_departament, id_functie, nume, data_angajare)

CONSULT(id_consult, id_personal, id_pacient, diagnostic, data_consult, pret_consult, detalii)

CONSULT_MEDICAMENT(id_consult, id_medicament, cantitate)

PROCEDURA(id_procedura, nume_procedura, pret, detalii, invaziv)

CONSULT_PROCEDURA(id_consult, id_procedura)

d. Crearea tabelelor și inserarea datelor

--Crearea tabelului ADRESA:

```
CREATE TABLE ADRESA(  
    id_adresa NUMBER(4),  
    judet VARCHAR2(20),  
    oras VARCHAR2(20) NOT NULL,  
    strada VARCHAR2(30),  
    cod_postal VARCHAR2(6),  
    CONSTRAINT adresa_pk PRIMARY KEY (id_adresa)  
);  
INSERT INTO ADRESA VALUES(1, 'Prahova', 'Ploiesti', 'Aleea Codrului', '102753');  
INSERT INTO ADRESA VALUES(2, 'Bucuresti', 'Bucuresti', 'Calea Plevnei 132', '010825');  
INSERT INTO ADRESA VALUES(3, 'Brasov', 'Brasov', 'Str. Sperantei', '242052');  
INSERT INTO ADRESA VALUES(4, 'Bucuresti', 'Bucuresti', 'Splaiul Independentei', '050098');  
INSERT INTO ADRESA VALUES(5, 'Prahova', 'Ploiesti', 'Bd. Republicii', '100265');
```

--Crearea tabelului PACIENT:

```
CREATE TABLE PACIENT(  
    id_pacient NUMBER(4),  
    id_adresa NUMBER(4),  
    cnp VARCHAR2(100) NOT NULL,  
    nume VARCHAR2(30) NOT NULL,  
    prenume VARCHAR2(50) NOT NULL,  
    sex VARCHAR2(1) CHECK (sex = 'F' OR sex = 'M'),  
    data_nastere DATE NOT NULL,
```

```

telefon VARCHAR2(100) NOT NULL,
email VARCHAR2(100),
CONSTRAINT pacient_pk PRIMARY KEY (id_pacient),
CONSTRAINT pacient_fk_adr FOREIGN KEY(id_adresa) REFERENCES ADRESA(id_adresa)
);
INSERT INTO PACIENT VALUES(1, 1, '2254173871953', 'Moldoveanu', 'Maria', 'F',
TO_DATE('23-06-1994', 'DD-MM-YYYY'), '0723657512', 'moldo.maria@gmail.com');
INSERT INTO PACIENT VALUES(2, 2, '2736478901255', 'Olaru', 'Andra', 'F',
TO_DATE('11-05-1998', 'dd-MM-yyyy'), '0762619120', 'andrao@gmail.com');
INSERT INTO PACIENT VALUES(3, 3, '1234567890987', 'Popescu', 'Ion', 'M',
TO_DATE('23-11-1970', 'dd-MM-yyyy'), '0734271957', 'ionion@yahoo.com');

```

--Crearea tabelului CLINICA

```

CREATE TABLE CLINICA(
id_clinica NUMBER(4),
id_adresa NUMBER(4),
nume VARCHAR2(50) NOT NULL,
contact VARCHAR2(100),
CONSTRAINT clinica_pk PRIMARY KEY (id_clinica),
CONSTRAINT clinica_fk_adr FOREIGN KEY(id_adresa) REFERENCES ADRESA(id_adresa)
);

```

```

INSERT INTO CLINICA VALUES(1, 4, 'Sanamed', 'sanamed@yahoo.ro');
INSERT INTO CLINICA VALUES(2, 5, 'Spitalul Universitar', '021 318 0523');

```

--Crearea tabelului MEDICAMENT:

```

CREATE TABLE MEDICAMENT(
id_medicament NUMBER(4),
nume_medicament VARCHAR2(50) NOT NULL,
pret_medicament NUMBER(4,2) DEFAULT 0,
instructiuni_administrare VARCHAR2(100),
CONSTRAINT medicament_pk PRIMARY KEY (id_medicament)
);
INSERT INTO MEDICAMENT VALUES(1, 'Bengay', 11, 'la nevoie');
INSERT INTO MEDICAMENT VALUES(2, 'Nurofen', 25, 'la 8 ore');
INSERT INTO MEDICAMENT VALUES(3, 'Aspirina', 20, 'la durere');
INSERT INTO MEDICAMENT VALUES(4, 'Algocalmin', 15, 'de 2 ori pe zi');
INSERT INTO MEDICAMENT VALUES(5, 'Paracetamol', 3, 'la fiecare 6 ore');
INSERT INTO MEDICAMENT VALUES(6, 'Antibiotic', 40, 'la 12 ore');
INSERT INTO MEDICAMENT VALUES(7, 'Antiinflamatoare', 30, 'de 3 ori pe zi');
INSERT INTO MEDICAMENT VALUES(8, 'Aspenter', 12, 'in fiecare zi dimineata');

```

--Crearea tabelului FUNCTIE

```

CREATE TABLE FUNCTIE(
id_functie NUMBER(4),
denumire_functie VARCHAR2(50) NOT NULL,
specializare VARCHAR2(50),

```

```

    CONSTRAINT functie_pk PRIMARY KEY (id_functie)
);
INSERT INTO FUNCTIE VALUES(1, 'Doctor', 'Ortopedie');
INSERT INTO FUNCTIE VALUES(2, 'Specialist', 'Ortopedie');
INSERT INTO FUNCTIE VALUES(3, 'Doctor', 'Cardiologie');
INSERT INTO FUNCTIE VALUES(4, 'Asistent', 'Ortopedie');
INSERT INTO FUNCTIE VALUES(5, 'Doctor', 'Pediatrie');
INSERT INTO FUNCTIE VALUES(6, 'Medic rezident', 'Oftalmologie');

--Crearea tabelului DEPARTAMENT
CREATE TABLE DEPARTAMENT(
    id_departament NUMBER(4),
    id_clinica NUMBER(4),
    nume_departament VARCHAR2(200) NOT NULL,
    id_manager_departament NUMBER(4),
    CONSTRAINT departament_pk PRIMARY KEY (id_departament),
    CONSTRAINT departament_fk_clinica FOREIGN KEY(id_clinica) REFERENCES
CLINICA(id_clinica)
);
INSERT INTO DEPARTAMENT VALUES (1, 1, 'Ortopedie si traumatologie', 1);
INSERT INTO DEPARTAMENT VALUES (2, 1, 'Cercetare', 1);
INSERT INTO DEPARTAMENT VALUES (3, 2, 'Cardiologie', 3);
INSERT INTO DEPARTAMENT VALUES (4, 2, 'Cercetare', 3);

--Crearea tabelului PERSONAL_MEDICAL
CREATE TABLE PERSONAL_MEDICAL(
    id_personal NUMBER(4),
    id_departament NUMBER(4),
    id_functie NUMBER(4),
    nume VARCHAR2(50) NOT NULL,
    data_angajare DATE,
    CONSTRAINT personal_pk PRIMARY KEY (id_personal),
    CONSTRAINT personal_job_fk FOREIGN KEY (id_functie) REFERENCES
FUNCTIE(id_functie),
    CONSTRAINT personal_departament_fk FOREIGN KEY (id_departament) REFERENCES
DEPARTAMENT(id_departament)
);
INSERT INTO PERSONAL_MEDICAL VALUES(1, 1, 1, 'Zaharescu Lucian',
TO_DATE('19-01-2008', 'dd-MM-yyyy'));
INSERT INTO PERSONAL_MEDICAL VALUES(2, 1, 4, 'Marin Iulia', TO_DATE('12-09-2018',
'dd-MM-yyyy'));
INSERT INTO PERSONAL_MEDICAL VALUES(3, 3, 3, 'Albu Marian', TO_DATE('23-02-2004',
'dd-MM-yyyy'));
INSERT INTO PERSONAL_MEDICAL VALUES(4, 4, 6, 'Miron Ilinca', TO_DATE('02-06-2020',
'dd-MM-yyyy'));
ALTER TABLE DEPARTAMENT
ADD CONSTRAINT depart_fk_manager FOREIGN

```

```
KEY(id_manager_departament)REFERENCES PERSONAL_MEDICAL(id_personal);
```

--Crearea tabelului CONSULT:

```
CREATE TABLE CONSULT(  
    id_consult NUMBER(4),  
    id_personal NUMBER(4),  
    id_pacient NUMBER(4),  
    diagnostic VARCHAR2(100),  
    data_consult DATE NOT NULL,  
    pret_consult NUMBER(5),  
    detalii VARCHAR2(200),  
    CONSTRAINT consult_pk PRIMARY KEY (id_consult),  
    CONSTRAINT consult_fk_pacient FOREIGN KEY (id_pacient) REFERENCES  
PACIENT(id_pacient),  
    CONSTRAINT consult_fk_personal FOREIGN KEY (id_personal) REFERENCES  
PERSONAL_MEDICAL(id_personal)  
);  
INSERT INTO CONSULT VALUES(1, 1, 1, 'Entorsa', TO_DATE(sysdate, 'dd-MM-yyyy'), 300,  
'entorsa sportiva');  
INSERT INTO CONSULT VALUES(2, 3, 3, 'Infarct', TO_DATE(sysdate, 'dd-MM-yyyy'), 200,  
'infarct miocardic in urma cu 6 luni');  
INSERT INTO CONSULT VALUES(3, 4, 3, 'Miopie', TO_DATE(sysdate, 'dd-MM-yyyy'), 100,  
'miopie pe ambii ochi cu dioptrie -1.5');
```

--Crearea tabelului CONSULT_MEDICAMENT:

```
CREATE TABLE CONSULT_MEDICAMENT(  
    id_consult NUMBER(4),  
    id_medicament NUMBER(4),  
    cantitate NUMBER(5) DEFAULT 1,  
    CONSTRAINT consult_med_pk_ PRIMARY KEY(id_consult, id_medicament),  
    CONSTRAINT medicament_fk FOREIGN KEY (id_medicament) REFERENCES  
MEDICAMENT(id_medicament),  
    CONSTRAINT consult_fk FOREIGN KEY (id_consult) REFERENCES CONSULT(id_consult)  
);  
INSERT INTO CONSULT_MEDICAMENT VALUES(1,1,1);  
INSERT INTO CONSULT_MEDICAMENT VALUES(2,8,5);  
INSERT INTO CONSULT_MEDICAMENT VALUES(3,7,1);
```

--Crearea tabelului PROCEDURA:

```
CREATE TABLE PROCEDURA(  
    id_procedura NUMBER(5),  
    nume_procedura VARCHAR2(100) NOT NULL,  
    pret NUMBER(7,2) NOT NULL,  
    detalii VARCHAR2(200),  
    invaziv VARCHAR2(2) CHECK (invaziv ='DA' OR invaziv = 'NU'),  
    CONSTRAINT procedura_pk PRIMARY KEY (id_procedura)  
);
```

```

INSERT INTO PROCEDURA VALUES(1, 'Radiografie', 100, 's-a constatat entorsa', 'DA');
INSERT INTO PROCEDURA VALUES(2, 'Masaj', 150, 'pentru vanatai', 'NU');
INSERT INTO PROCEDURA VALUES(3, 'EKG', 300, 'variatiile batailor inimii', 'NU');
INSERT INTO PROCEDURA VALUES(4, 'Aparat oftalmologic', 0, 'pentru a testa vederea', 'NU');

--Crearea tabelului CONSULT_PROCEDURA:
CREATE TABLE CONSULT_PROCEDURA(
    id_consult NUMBER(4),
    id_procedura NUMBER(4),
    CONSTRAINT consult_fk_cp FOREIGN KEY (id_consult) REFERENCES
CONSULT(id_consult),
    CONSTRAINT procedura_fk_cp FOREIGN KEY (id_procedura) REFERENCES
PROCEDURA(id_procedura),
    CONSTRAINT consult_procedura_pk PRIMARY KEY (id_consult, id_procedura)
);
INSERT INTO CONSULT_PROCEDURA VALUES(1,1);
INSERT INTO CONSULT_PROCEDURA VALUES(1,2);
INSERT INTO CONSULT_PROCEDURA VALUES(2,3);
INSERT INTO CONSULT_PROCEDURA VALUES(3,4);

commit;

```

Table ADRESA created.

Table PACIENT created.

Table CLINICA created.

Table MEDICAMENT created.

Table FUNCTIE created.

Table PERSONAL_MEDICAL created.

Table DEPARTAMENT altered.

Table CONSULT created.

Table CONSULT_MEDICAMENT created.

Table PROCEDURA created.

Table CONSULT_PROCEDURA created.

e.Prezentarea regulilor de securitate care vor fi aplicate asupra modelului

Trebuie implementate anumite reguli pentru a fi respectată securitatea aplicației, adică pentru ca datele cu caracter personal ale pacienților și datele confidențiale din cadrul clinicii să nu fie dezvăluite și să nu poată fi sparte de către cei cu intenții rele.

Fiind necesar ca anumite date despre pacienți și despre personalul medical să fie păstrate pentru înregistrare și pentru monitorizarea istoricului, pentru acestea trebuie respectată integritatea, confidentialitatea și disponibilitatea. În consecință, datele cu caracter personal ale pacientului vor fi criptate, vor fi implementate anumite roluri în cadrul bazei de

date pentru acces personalizat la informații și va fi implementat mecanismul de audit pentru a monitoriza activitatea asupra bazei de date.

2. Criptarea datelor

Vom cripta datele personale ale pacientului precum CNP, telefon și email-ul, cât și diagnosticele puse la un consult pentru a păstra confidențialitatea datelor. Cei care vor avea acces la aceste date prin cheile de criptare vor fi secretarii/recepționistii și doctorii pentru a avea informațiile necesare în vederea exercitării meseriei.

Pentru criptarea datelor a fost folosit algoritmul simetric de criptare AES cu schema de padding PKCS#5 și chaining de tip CBC. Cheile sunt de 16 bytes, generate automat și stocate într-o tabela specială, TABEL_CHEI, iar fiecărei înregistrări îi corespunde o cheie.

```
drop sequence secv_id_cheie;
create sequence secv_id_cheie start with 1 increment by 1;
--cnp, telefon, email, adresa
DROP TABLE TABEL_CHEI;
CREATE TABLE TABEL_CHEI(
    id_cheie NUMBER,
    cheie RAW(16) NOT NULL,
    id_inregistrare NUMBER(4) NOT NULL,
    nume_tabel VARCHAR2(30) NOT NULL
);

CREATE OR REPLACE PROCEDURE ENCRYPT_DATA(id_cautat IN NUMBER, info
IN VARCHAR2, tabel IN VARCHAR2, rezultat OUT VARCHAR2) AS
    raw_cheie raw(16);
    raw_info raw(200);

    mod_operare pls_integer;

    contor INTEGER := 0;
BEGIN
    select count(*) into contor from system.tabel_chei where id_inregistrare = id_cautat and
lower(nume_tabel) = lower(tabel);

    if contor > 0 then
        select cheie into raw_cheie
        from system.tabel_chei
        where id_inregistrare = id_cautat;
    else
```

```

        raw_cheie := dbms_crypto.randombytes(16);
        insert into system.tabel_chei values (secv_id_cheie.nextval, raw_cheie, id_cautat,
tabel);
    end if;

    mod_operare := dbms_crypto.encrypt_aes128 + dbms_crypto.pad_pkcs5 +
dbms_crypto.chain_cbc;
    raw_info := utl_i18n.string_to_raw(info, 'AL32UTF8');

    rezultat:= dbms_crypto.encrypt(raw_info, mod_operare, raw_cheie);
END;
/

```

Ulterior am creat un trigger care criptează datele necesare înainte de inserarea unui pacient sau a unui consult.

```

--Create trigger PACIENTI
CREATE OR REPLACE TRIGGER encrypt_data_pacienti
BEFORE INSERT ON PACIENT
FOR EACH ROW
DECLARE
    encrypted_data VARCHAR2(200);
BEGIN
    ENCRYPT_DATA(:NEW.id_pacient, :NEW.cnp, 'pacient', encrypted_data);
    :NEW.cnp := encrypted_data;

    ENCRYPT_DATA(:NEW.id_pacient, :NEW.telefon, 'pacient', encrypted_data);
    :NEW.telefon := encrypted_data;

    ENCRYPT_DATA(:NEW.id_pacient, :NEW.email, 'pacient', encrypted_data);
    :NEW.email := encrypted_data;
END;
/

--Create trigger CONSULT
CREATE OR REPLACE TRIGGER encrypt_data_consult
BEFORE INSERT ON CONSULT
FOR EACH ROW
DECLARE
    encrypted_data VARCHAR2(200);
BEGIN
    ENCRYPT_DATA(:NEW.id_consult, :NEW.diagnostic, 'consult', encrypted_data);

```



```

:NEW.diagnostic := encrypted_data;
END;
/

```

```

Procedure ENCRYPT_DATA compiled

```

```

Trigger ENCRYPT_DATA_PACIENTI compiled

```

```

Trigger ENCRYPT_DATA_CONSULT compiled

```

Încercăm să inserăm în PACIENT și observăm că a fost adăugată o nouă cheie în tabel și datele din tabel sunt criptate.

```

INSERT INTO PACIENT VALUES(11, 1, '123456789098', 'Dinu', 'Ionut', 'M',
TO_DATE('19-04-2000','DD-MM-YYYY'),'0735271864','ionutd@gmail.com');

select * from pacient;
select * from tabel_chei where nume_tabel = 'pacient';

```

ID_PACIENT	ID_ADRESA	CNP	NUME	PRENUME	SEX	DATA_NASTERE	TELEFON	EMAIL
1	1	11254173871953	Moldoveanu	Maria	F	23-JUN-94	0723657512	moldo.maria@gmail.com
2	11	12718788E0188A6D2F428213915250233	Dinu	Ionut	M	19-APR-00	C484662A9346DCA4FA7F3B5488C24CAC	A8A7B54C3296A57D133933B528EEFF6B96FBE335175BCC2F3F4E3F9B24

ID_CHEIE	CHEIE	ID_INREGISTRARE	NUME_TABEL
1	10DEBE89C3154FF8D50A990BE8E586B27	11	pacient

De asemenea, am implementat și o procedură care decriptează datele unui pacient sau a unui consult pentru care id-ul, cât și numele tabelului sunt date ca parametri.

```

CREATE OR REPLACE PROCEDURE DECRYPT_DATA(id_cautat IN NUMBER, tabel
IN VARCHAR2)
AS

```

```

    raw_cheie raw(16);
    mod_operare PLS_INTEGER;
    cnp_criptat VARCHAR2(100);
    telefon_criptat VARCHAR2(100);
    email_criptat VARCHAR2(100);
    diagnostic_criptat VARCHAR2(100);
    rezultat_cnp raw(200);
    rezultat_telefon raw(200);
    rezultat_email raw(200);
    rezultat_diagnostic raw(200);

```

```

BEGIN

```

```

select cheie
into raw_cheie
from system.tabel_chei
where id_inregistrare = id_cautat and lower(ume_tabel) = lower(tabel);

DBMS_OUTPUT.PUT_LINE('Cheie ' || raw_cheie);

mod_operare := dbms_crypto.encrypt_aes128 + dbms_crypto.pad_pkcs5 +
dbms_crypto.chain_cbc;

if (lower(tabel) = 'pacient') then
    select cnp, telefon, email
    into cnp_criptat, telefon_criptat, email_criptat
    from system.pacient where id_pacient = id_cautat;

    rezultat_cnp := dbms_crypto.decrypt(cnp_criptat, mod_operare, raw_cheie);
    rezultat_telefon := dbms_crypto.decrypt(telefon_criptat, mod_operare, raw_cheie);
    rezultat_email := dbms_crypto.decrypt(email_criptat, mod_operare, raw_cheie);

    DBMS_OUTPUT.PUT_LINE('CNP: ' || utl_i18n.raw_to_char(rezultat_cnp,
'AL32UTF8'));
    DBMS_OUTPUT.PUT_LINE('TELEFON: ' || utl_i18n.raw_to_char(rezultat_telefon,
'AL32UTF8'));
    DBMS_OUTPUT.PUT_LINE('EMAIL: ' || utl_i18n.raw_to_char(rezultat_email,
'AL32UTF8'));
    elsif (lower(tabel) = 'consult') then
        select diagnostic
        into diagnostic_criptat
        from system.consult where id_consult = id_cautat;

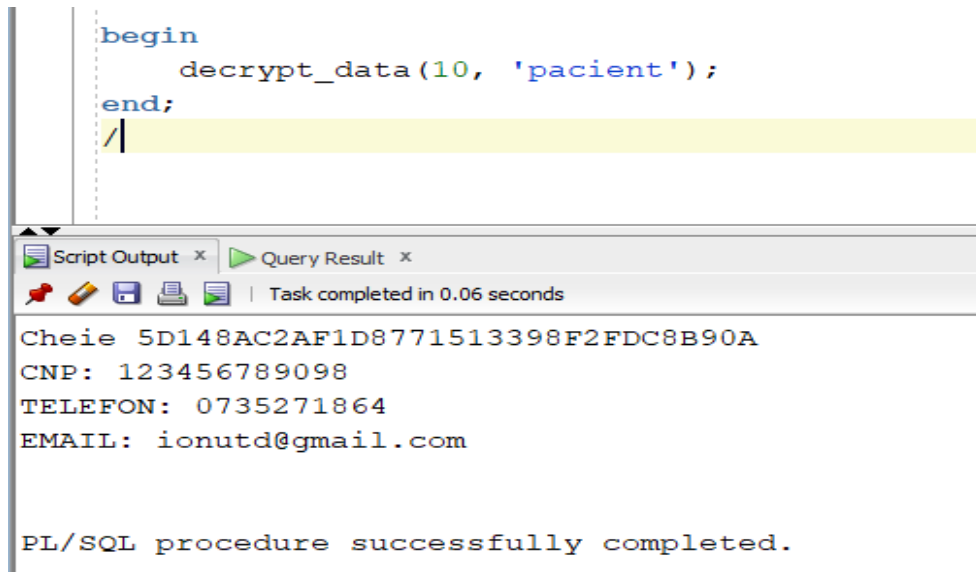
        rezultat_diagnostic := dbms_crypto.decrypt(diagnostic_criptat, mod_operare,
raw_cheie);

        DBMS_OUTPUT.PUT_LINE('Diagnostic: ' ||
utl_i18n.raw_to_char(rezultat_diagnostic, 'AL32UTF8'));
    else
        DBMS_OUTPUT.PUT_LINE('Nu exista rezultate pentru acest tabel');
    end if;
END;
/

```

Exemplu pentru un pacient:

```
begin
    decrypt_data(10, 'pacient');
end;
/
```



3. Auditarea activităților asupra bazei de date

Auditul reprezintă o parte importantă a configurării securității bazei de date, deoarece acesta monitorizează activitatea angajaților, a interogărilor și a modificărilor asupra bazei de date și așa putem studia și afla la momentul potrivit dacă există vreo breșă în securitate pentru a nu pune în pericol pacienții și clinica.

a. Auditarea standard

Pentru primul tip de auditare, auditarea standard, am ales să setez auditul pentru toate comenzile de tip Select. Acest lucru ne va ajuta în realizarea anumitor statistici (frecvența accesării) sau pentru a ne da seama dacă o persoană neautorizată a avut acces la niște informații.

Prima dată vom seta parametrul *audit_trail* să salveze informațiile în baza de date. Această informație va fi actualizată după repornirea bazei de date din SQL plus cu comenzile *shutdown immediate* și *startup*.

```
show con_name;
alter session set container = cdb$root;
show parameter audit_trail;

alter system set audit_trail = db,extended scope = spfile;
show parameter audit_trail;
```

Script Output x Query Result x		
Task completed in 0.054 seconds		
NAME	TYPE	VALUE

audit_trail	string	DB, EXTENDED

Acum schimbăm conexiunea pe *orclpdb* și activăm auditul pentru select.

```
-- alter pluggable database orclpdb open; din sys
alter session set container = orclpdb;

audit select table;

select oras from adresa
where id_adresa = 1;
```

Testăm:

<pre>select obj\$name, sqltext, userid, ntimestamp# from SYS.aud\$ where lower(obj\$name) = 'adresa' order by ntimestamp# desc;</pre>			
Script Output x Query Result x Query Result 1 x			
All Rows Fetched: 19 in 0.114 seconds			
OBJ\$NAME	SQLTEXT	USERID	NTIMESTAMP#
1 ADRESA	select oras from adresa where id_adresa = 1	SYSTEM	29-DEC-22 05.44.46.101000000 PM

De asemenea, toate informațiile de auditare vor fi arhivate periodic și șterse, pentru a evita supraîncărcarea memoriei.

```
delete from sys.aud$ where obj$name in ('consult', 'consult_procedura', 'consult_procedura',
'medicament', 'pacient', 'adresa', 'functie', 'departament', 'personal_medical', 'procedura',
'clinica');
```

Vom seta și auditul pentru pacienți în care sunt urmărite toate acțiunile asupra tabelului.

```
audit select, insert, update, delete on pacient;
```

<pre>select object_name, object_type, owner, sel, ins, upd, del from dba_obj_audit_opts where lower(object_name) = 'pacient';</pre>							
Script Output x Query Result x Query Result 1 x Query Result 2 x							
All Rows Fetched: 1 in 0.214 seconds							
	OBJECT_NAME	OBJECT_TYPE	OWNER	SEL	INS	UPD	DEL
1	PACIENT	TABLE	SYSTEM	A/A	A/A	A/A	A/A

b. Triggeri de auditare

Se pot implementa, de asemenea, triggeri care urmăresc informațiile de interes din baza de date, iar acestea pot fi stocate într-un tabel. Am creat 3 triggeri pe care i-am testat și care vor fi enumerați în cele ce urmează.

- Trigger pentru situațiile în care se modifică prețul unei proceduri, iar pacienții trebuie anunțați

```
drop sequence secv_aud_proc;  
create sequence secv_aud_proc start with 1 increment by 1;  
  
DROP TABLE TAB_AUDIT_PROC_PRET;  
CREATE TABLE TAB_AUDIT_PROC_PRET(  
    id_secv NUMBER(4) PRIMARY KEY,  
    user_ VARCHAR2(20),  
    session_ NUMBER(10),  
    host_ VARCHAR2(100),  
    timp DATE,  
    nume_proc VARCHAR2(100),  
    pret_nou_proc NUMBER(7,2),  
    pret_vechi NUMBER(7,2)  
);  
  
CREATE OR REPLACE TRIGGER T_AUDIT_PRET_PROC  
AFTER UPDATE OF pret ON procedura  
FOR EACH ROW  
BEGIN  
    insert into tab_audit_proc_pret  
    values(secv_aud_proc.nextval,sys_context('userenv', 'session_user'),  
sys_context('userenv', 'sessionid'), sys_context('userenv','host'), sysdate,  
:NEW.nume_procedura, :NEW.pret, :OLD.pret);  
END;  
/
```

```

Sequence SECV_AUD_PROC dropped.

Sequence SECV_AUD_PROC created.

Table TAB_AUDIT_PROC_PRET dropped.

Table TAB_AUDIT_PROC_PRET created.

Trigger T_AUDIT_PRET_PROC compiled

1 row updated.

```

UPDATE PROCEDURA

SET pret = 200

where id_procedura = 1;

```
SELECT * FROM TAB_AUDIT_PROC_PRET;
```

Script Output x Query Result x

SQL | All Rows Fetched: 1 in 0.002 seconds

ID_SECV	USER_	SESSION_	HOST_	TIMP	NUME_PROC	PRET_NOU_PROC	PRET_VECHI
1	1	SYSTEM	270163	DESKTOP-OI77CPA	29-DEC-22	Radiografie	200 150

- Trigger care urmărește schimbarea tratamentului pacientului, pentru a avea o imagine clară asupra evoluției pacientului și a ne asigura că schimbarea este benefică.

```

drop sequence secv_aud_med;
create sequence secv_aud_med start with 1 increment by 1;
drop table TAB_AUDIT_MEDICAMENTE;
CREATE TABLE TAB_AUDIT_MEDICAMENTE(
  id_secv NUMBER(4) PRIMARY KEY,
  user_ VARCHAR2(20),
  session_ NUMBER(10),
  host_ VARCHAR2(100),
  timp DATE,
  nume_med VARCHAR2(50),
  cant_veche NUMBER(5),
  cant_noua NUMBER(5),
  id_pac NUMBER(4),
  id_doc NUMBER(4)
);

```

```

CREATE OR REPLACE TRIGGER t_audit_medicamente_cant_upd
AFTER UPDATE ON consult_medicament
FOR EACH ROW
DECLARE
    id_pac NUMBER(4);
    id_doc NUMBER(4);
    nume_med VARCHAR2(50);
BEGIN
    select id_pacient, id_personal, nume_medicament
    into id_pac, id_doc, nume_med
    from consult c, medicament m
    where c.id_consult = :NEW.id_consult and m.id_medicament = :NEW.id_medicament;

    insert into tab_audit_medicamente
    values(secv_aud_med.nextval, sys_context('userenv', 'session_user'),
sys_context('userenv', 'sessionid'), sys_context('userenv', 'host'), sysdate, nume_med,
:NEW.cantitate, :OLD.cantitate, id_pac, id_doc);
END;
/

UPDATE CONSULT_MEDICAMENT
SET cantitate = 2
WHERE id_consult = 1 and id_medicament = 1;

```

```

Sequence SECV_AUD_MED dropped.

Sequence SECV_AUD_MED created.

Table TAB_AUDIT_MEDICAMENTE dropped.

Table TAB_AUDIT_MEDICAMENTE created.

Trigger T_AUDIT_MEDICAMENTE_CANT_UPD compiled

1 row updated.

```

`select * from tab_audit_medicamente;`

ID_SECV	USER_	SESSION_	HOST_	TIME	NUME_MED	CANT_VECH	CANT_NOUA	ID_PAC	ID_DOC
1	1 SYSTEM	270163	DESKTOP-OI77CPA	29-DEC-22	Bengay	2	2	1	1

- Trigger care monitorizează dacă un doctor și-a schimbat funcția și în același timp este verificat dacă a fost mutat și în alt departament.


```

drop sequence secv_aud_doct;
create sequence secv_aud_doct start with 1 increment by 1;
drop table TAB_AUDIT_PERS_MEDICAL;
CREATE TABLE TAB_AUDIT_PERS_MEDICAL(
    id_secv NUMBER(4) PRIMARY KEY,
    user_ VARCHAR2(20),
    session_ NUMBER(10),
    host_ VARCHAR2(100),
    timp DATE,
    nume_pers VARCHAR2(50),
    functie_veche NUMBER(4),
    functie_noua NUMBER(4),
    id_depart NUMBER(4),
    id_depart_nou NUMBER(4)
);

CREATE OR REPLACE TRIGGER t_audit_pers_medical
AFTER UPDATE OF id_functie ON personal_medical
FOR EACH ROW
BEGIN
    if (:NEW.id_departament = :OLD.id_departament) then
        insert into TAB_AUDIT_PERS_MEDICAL(id_secv, user_, session_, host_, timp,
nume_pers, functie_veche, functie_noua, id_depart)
        values(secv_aud_med.nextval, sys_context('userenv', 'session_user'),
sys_context('userenv', 'sessionid'), sys_context('userenv', 'host'), sysdate, :NEW.nume,
:OLD.id_functie, :NEW.id_functie, :OLD.id_departament);
    else
        insert into TAB_AUDIT_PERS_MEDICAL
        values(secv_aud_med.nextval, sys_context('userenv', 'session_user'),
sys_context('userenv', 'sessionid'), sys_context('userenv', 'host'), sysdate, :NEW.nume,
:OLD.id_functie, :NEW.id_functie, :OLD.id_departament, :NEW.id_departament);
    end if;
END;
/

UPDATE PERSONAL_MEDICAL
SET id_functie = 2
WHERE id_personal = 1;

UPDATE PERSONAL_MEDICAL
SET id_functie = 2, id_departament = 1
WHERE id_personal = 1;

```

```

Sequence SECV_AUD_DOCT dropped.

Sequence SECV_AUD_DOCT created.

Table TAB_AUDIT_PERS_MEDICAL dropped.

Table TAB_AUDIT_PERS_MEDICAL created.

Trigger T_AUDIT_PERS_MEDICAL compiled

1 row updated.

1 row updated.

select * from tab_audit_pers_medical;

```

The screenshot shows the SQL Developer interface with the query result displayed in a table. The table has 10 columns: ID_SECV, USER_, SESSION_, HOST_, TIMP, NUME_PERS, FUNCTIE_VECH, FUNCTIE_NOUA, ID_DEPART, and ID_DEPART_NOU. There are two rows of data.

ID_SECV	USER_	SESSION_	HOST_	TIMP	NUME_PERS	FUNCTIE_VECH	FUNCTIE_NOUA	ID_DEPART	ID_DEPART_NOU
1	6 SYSTEM	270163	DESKTOP-OI77CPA	29-DEC-22	Zaharescu Lucian	2	2	2	(null)
2	7 SYSTEM	270163	DESKTOP-OI77CPA	29-DEC-22	Zaharescu Lucian	2	2	2	1

c. Politici de auditare

Am creat o politică de auditare pentru tabelul PACIENT, aceasta fiind o altă metodă prin care se poate realiza auditarea bazei de date. Pentru ca aceasta să funcționeze am creat și procedura *proc_audit_alert*, handler-ul politicii.

```

begin
DBMS_FGA.DROP_POLICY(
  object_schema => 'system',
  object_name   => 'PACIENT',
  policy_name   => 'policy_pacient');
end;
/

create or replace procedure proc_audit_alert(object_schema varchar2, object_name
varchar2,
                                     policy_name varchar2)
as
begin
  dbms_output.put_line('Incercare modificare informatii pacient');
end;
/

CREATE OR REPLACE PROCEDURE proc_audit_pac as

```

```

BEGIN
  dbms_fga.add_policy(
    object_schema => 'system',
    object_name => 'PACIENT',
    policy_name => 'policy_pacient',
    enable => false,
    statement_types => 'UPDATE',
    handler_module => 'PROC_AUDIT_ALERT');
END;
/

```

```

execute proc_audit_pac;

```

```

PL/SQL procedure successfully completed.

```

```

Procedure PROC_AUDIT_ALERT compiled

```

```

Procedure PROC_AUDIT_PAC compiled

```

```

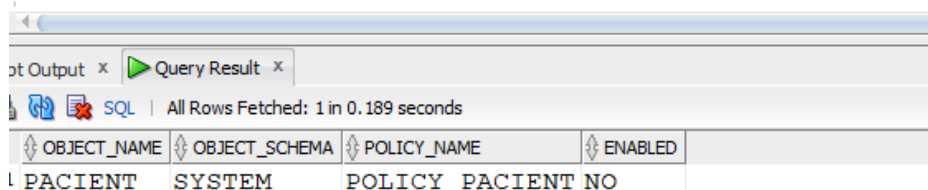
PL/SQL procedure successfully completed.

```

```

select object_name, object_schema, policy_name, enabled
from all_audit_policies;

```



OBJECT_NAME	OBJECT_SCHEMA	POLICY_NAME	ENABLED
PACIENT	SYSTEM	POLICY_PACIENT	NO

Următorul pas este să marcăm politica de auditare ca enabled și să actualizăm informațiile pentru un pacient pentru a o testa interogând tabelul *dba_fga_audit_trail*.

```

begin
  dbms_fga.enable_policy(object_schema => 'SYSTEM',
    object_name => 'PACIENT',
    policy_name => 'policy_pacient');
end;
/

```

```
193
194 update pacient
195 set nume = 'Popescu'
196 where id_pacient = 1;
```

Script Output x Query Result x

Task completed in 0.036 seconds

Incercare modificare informatii pacient

1 row updated.

```
197
198 select db_user, userhost, policy_name, timestamp, sql_text
199 from dba_fga_audit_trail;
```

Script Output x Query Result x

All Rows Fetched: 3 in 0.002 seconds

DB_USER	USERHOST	POLICY_NAME	TIMESTAMP	SQL_TEXT
SYSTEM	DESKTOP-OI77CPA	POLICY_PACIENT	27-DEC-22	update pacientset nume = 'Popescu'where id

4. Gestiunea utilizatorilor unei baze de date și a resurselor computaționale

a. Proiectarea configurației de management a identităților în baza de date

Utilizatorii aplicației identificați în cadrul bazei de date sunt:

- Administratorul aplicației
- Pacienții (identificați unic cu ajutorul CNP-ului)
- Medicii cu diferite specializări responsabili de anumiți pacienți
- Asistentele medicale
- Secretarele clinicii
- Contabilul clinicii

Procesele identificate în cadrul aplicației sunt:

P1: Înființarea și configurarea clinicilor medicale

P2: Crearea departamentelor din fiecare clinică

P3: Actualizarea adresei/numelui/contactului clinicii

P4: Vizualizarea personalului medical din fiecare departament

P5: Adăugarea unui nou personal medical

P6: Împărțirea cadrelor medicale pe departamente

P7: Vizualizarea listei funcțiilor personalului dintr-o clinică

P8: Adăugarea sau modificarea informațiilor unui pacient

P9: Vizualizarea datelor personale ale pacientului

P10: Consultarea și atribuirea unui diagnostic unui pacient

P11: Prescrierea unor medicamente pacientului

P12: Realizarea unei/unor proceduri medicale pacientului

P13: Vizualizarea listei de consultatii si a diagnosticelor unui pacient

P14: Consultarea prețurilor procedurilor și a consultațiilor

P15: Administrare utilizatori

- Construirea matricei proces utilizator:

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15
Administrator aplicație	X	X	X	X	X	X	X							X	X
Pacienții				X					X				X	X	
Medici				X			X		X	X	X	X	X	X	
Asistente				X			X		X			X	X	X	
Secretare				X	X	X	X	X	X					X	
Contabil														X	

- Identificarea entităților din aplicație

În cadrul aplicației este monitorizată activitatea desfășurată de anumite clinici medicale din țară. Fiecare clinică dispune de personal medical care deține o anumită funcție și care este împărțit pe diferite departamente. Acestea sunt responsabile de pacienții care vin la consulturi medicale. În cadrul acestora, pot fi efectuate anumite proceduri medicale pentru a depista diagnosticul și pot fi prescrise anumite medicamente pentru tratament.

- Construirea matricei entitate-proces:

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15
PACIENT								I,U,D	S						S,I,U,D
CLINICA	I,U,D		U				S								
ADRESA			I,U					I,U	S						
CONSULT										I	S	S	S	S	
PROCEDURA												S		S	
CONSULT_PROCEDURA												I,U,D			
MEDICAMENT											S				
CONSULT_MEDICAMENT											I,U,D				
PERSONAL_MEDICAL				S	I	U	S								S,I,U,D
DEPARTAMENT		I,U,D		S		I,U,D	S								
FUNCTIE							S								

S=Select, I=Insert, U=Update, D=Delete

- Construirea matricei entitate utilizator:

	Administrat or aplicație	Pacienții	Medici	Asistente	Secretare	Contabil
PACIENT		S	S	S	S,I,U,D	
CLINICA	S,I,U,D	S	S	S	S	
ADRESA	I,U,D	S	S	S	S,I,U	
CONSULT		S	S,I,U	S	S	
PROCEDURA	I,U	S	S	S		S
CONSULT_PROCEDURA		S	S,I,U	S,I,U		S

MEDICAMENT	I,U	S	S	S		S
CONSULT_MEDICAMENT		S	S,I,U	S		S
PERSONAL_MEDICAL	S,I,U,D	S	S	S	S	
DEPARTAMENT	S,I,U,D	S	S	S	S	
FUNCTIE	I,U,D	S	S	S	S	

b. Implementarea configurației de management a identităților în baza de date

Pentru aplicație vor fi necesare 1 cont de administrator, 6 conturi de cadru medical (doctori, asistente), 10 de pacienți, 2 secretare, 1 contabil.

alter session set container = ORCLPDB;

```
--Crearea adminului
create user administrator identified by administrator password expire;
grant create session to administrator;

--Crearea pacientilor
create user pacient1 identified by pacient1 password expire;
grant create session to pacient1;

create user pacient2 identified by pacient2 password expire;
grant create session to pacient2;

create user pacient3 identified by pacient3 password expire;
grant create session to pacient3;

create user pacient4 identified by pacient4 password expire;
grant create session to pacient4;

create user pacient5 identified by pacient5 password expire;
grant create session to pacient5;

create user pacient6 identified by pacient6 password expire;
grant create session to pacient6;

create user pacient7 identified by pacient7 password expire;
grant create session to pacient7;
```



```
create user pacient8 identified by pacient8 password expire;  
grant create session to pacient8;
```

```
create user pacient9 identified by pacient9 password expire;  
grant create session to pacient9;
```

```
create user pacient10 identified by pacient10 password expire;  
grant create session to pacient10;
```

--Createa doctorilor

```
create user doctor1 identified by doctor1 password expire;  
grant create session to doctor1;
```

```
create user doctor2 identified by doctor2 password expire;  
grant create session to doctor2;
```

```
create user doctor3 identified by doctor3 password expire;  
grant create session to doctor3;
```

```
create user doctor4 identified by doctor4 password expire;  
grant create session to doctor4;
```

--Crearea asistentelor medicale

```
create user asistenta1 identified by asistenta1 password expire;  
grant create session to asistenta1;
```

```
create user asistenta2 identified by asistenta2 password expire;  
grant create session to asistenta2;
```

--Creare secretarelor

```
create user secretara1 identified by secretara1 password expire;  
grant create session to secretara1;
```

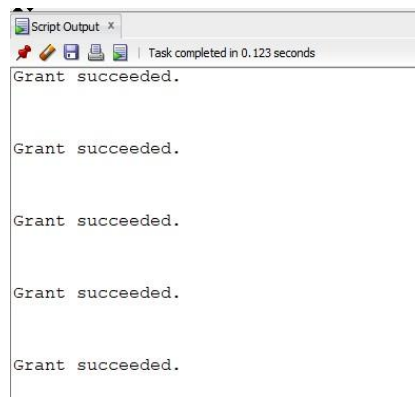
```
--grant execute on dbms_crypto to secretara1;
```

```
create user secretara2 identified by secretara2 password expire;  
grant create session to secretara2;
```

```
--grant execute on dbms_crypto to secretara2;
```

--Crearea personalului financiar

```
create user contabil identified by contabil password expire;  
grant create session to contabil;
```



După creare, aceștia pot fi vizualizați.

```

71 -- Vizualizarea user-urilor
72 SELECT USERNAME, AUTHENTICATION_TYPE, ACCOUNT_STATUS, CREATED,
73        EXPIRY_DATE
74 FROM DBA_USERS
75 ORDER BY CREATED desc;

```

Script Output x Query Result x

SQL | Fetched 50 rows in 0.036 seconds

	USERNAME	AUTHENTICATION_TYPE	ACCOUNT_STATUS	CREATED	EXPIRY_DATE
1	CONTABIL	PASSWORD	OPEN	27-DEC-22	25-JUN-23
2	SECRETARA2	PASSWORD	OPEN	27-DEC-22	25-JUN-23
3	SECRETARA1	PASSWORD	OPEN	27-DEC-22	25-JUN-23
4	ASISTENTA2	PASSWORD	OPEN	27-DEC-22	25-JUN-23
5	ASISTENTA1	PASSWORD	OPEN	27-DEC-22	25-JUN-23
6	DOCTOR2	PASSWORD	OPEN	27-DEC-22	25-JUN-23
7	DOCTOR1	PASSWORD	OPEN	27-DEC-22	25-JUN-23
8	DOCTOR4	PASSWORD	OPEN	27-DEC-22	25-JUN-23
9	DOCTOR3	PASSWORD	OPEN	27-DEC-22	25-JUN-23
10	PACIENT10	PASSWORD	OPEN	27-DEC-22	25-JUN-23
11	PACIENT9	PASSWORD	OPEN	27-DEC-22	25-JUN-23
12	PACIENT8	PASSWORD	OPEN	27-DEC-22	25-JUN-23
13	PACIENT7	PASSWORD	OPEN	27-DEC-22	25-JUN-23
14	PACIENT6	PASSWORD	OPEN	27-DEC-22	25-JUN-23
15	PACIENT4	PASSWORD	OPEN	27-DEC-22	25-JUN-23
16	PACIENT3	PASSWORD	OPEN	27-DEC-22	25-JUN-23
17	PACIENT2	PASSWORD	OPEN	27-DEC-22	25-JUN-23
18	PACIENT1	PASSWORD	OPEN	27-DEC-22	25-JUN-23
19	ADMINISTRATOR	PASSWORD	OPEN	27-DEC-22	25-JUN-23

În continuare, vom împărți utilizatorilor cotele de spațiu de stocare al tablespace-ului implicit USERS.

```

alter user administrator quota unlimited on users;
alter user doctor1 quota 10M on users;
alter user doctor2 quota 10M on users;
alter user doctor3 quota 10M on users;
alter user doctor4 quota 10M on users;
alter user pacient1 quota 5M on users;
alter user pacient2 quota 5M on users;
alter user pacient3 quota 5M on users;
alter user pacient4 quota 5M on users;
alter user pacient5 quota 5M on users;
alter user pacient6 quota 5M on users;

```

```

alter user pacient7 quota 5M on users;
alter user pacient8 quota 5M on users;
alter user pacient9 quota 5M on users;
alter user pacient10 quota 5M on users;
alter user secretara1 quota 0M on users;
alter user secretara2 quota 0M on users;
alter user contabil quota 0M on users;

```

select * from dba_ts_quotas;

TABLESPACE_NAME	USERNAME	BYTES	MAX_BYTES	BLOCKS	MAX_BLOCKS	DROPPED
USERS	PACIENT1	0	5242880	0	640	NO
USERS	PACIENT2	0	5242880	0	640	NO
USERS	PACIENT3	0	5242880	0	640	NO
USERS	PACIENT4	0	5242880	0	640	NO
USERS	PACIENT6	0	5242880	0	640	NO
USERS	PACIENT7	0	5242880	0	640	NO
USERS	PACIENT8	0	5242880	0	640	NO
USERS	PACIENT9	0	5242880	0	640	NO
USERS	PACIENT10	0	5242880	0	640	NO
USERS	DOCTOR1	0	10485760	0	1280	NO
USERS	DOCTOR2	0	10485760	0	1280	NO
USERS	DOCTOR3	0	10485760	0	1280	NO
USERS	DOCTOR4	0	10485760	0	1280	NO
USERS	PACIENT5	0	5242880	0	640	NO

Vom crea și profiluri pentru doctori și pacienți.

```

create profile profil_doctor limit
  cpu_per_call 6000
  sessions_per_user 1
  password_life_time 14
  failed_login_attempts 3;

create profile profil_pacient limit
  cpu_per_call 6000
  sessions_per_user 1
  password_life_time 14
  failed_login_attempts 3;

```

<pre>select * from dba_profiles where lower(profile) LIKE 'profil_%';</pre>						
Script Output x Query Result x Query Result 1 x Query Result 2 x						
SQL All Rows Fetched: 34 in 0.019 seconds						
PROFILE	RESOURCE_NAME	RESOURCE_TYPE	LIMIT	COMMON	INHERITED	IMPLICIT
1 PROFIL_DOCTOR	COMPOSITE_LIMIT	KERNEL	DEFAULT	NO	NO	NO
2 PROFIL_PACIENT	COMPOSITE_LIMIT	KERNEL	DEFAULT	NO	NO	NO
3 PROFIL_DOCTOR	SESSIONS_PER_USER	KERNEL	1	NO	NO	NO
4 PROFIL_PACIENT	SESSIONS_PER_USER	KERNEL	1	NO	NO	NO
5 PROFIL_DOCTOR	CPU_PER_SESSION	KERNEL	DEFAULT	NO	NO	NO
6 PROFIL_PACIENT	CPU_PER_SESSION	KERNEL	DEFAULT	NO	NO	NO
7 PROFIL_DOCTOR	CPU_PER_CALL	KERNEL	6000	NO	NO	NO
8 PROFIL_PACIENT	CPU_PER_CALL	KERNEL	6000	NO	NO	NO
9 PROFIL_DOCTOR	LOGICAL_READS_PER_SESSION	KERNEL	DEFAULT	NO	NO	NO
10 PROFIL_PACIENT	LOGICAL_READS_PER_SESSION	KERNEL	DEFAULT	NO	NO	NO
11 PROFIL_DOCTOR	LOGICAL_READS_PER_CALL	KERNEL	DEFAULT	NO	NO	NO
12 PROFIL_PACIENT	LOGICAL_READS_PER_CALL	KERNEL	DEFAULT	NO	NO	NO
13 PROFIL_DOCTOR	IDLE_TIME	KERNEL	DEFAULT	NO	NO	NO
14 PROFIL_PACIENT	IDLE_TIME	KERNEL	DEFAULT	NO	NO	NO
15 PROFIL_DOCTOR	CONNECT_TIME	KERNEL	DEFAULT	NO	NO	NO
16 PROFIL_PACIENT	CONNECT_TIME	KERNEL	DEFAULT	NO	NO	NO

5. Privilegii și roluri

Privilegiile utilizatorilor vor fi acordate în concordanță cu diagrama entitate-utilizator definită anterior.

Acestea vor fi acordate pe rând pentru administratorul aplicației, pentru secretare și pentru contabil, iar pentru pacienți, doctor și asistente vor fi create roluri care le vor fi asignate.

--Privilegii adminstrator

```
grant select, insert, update, delete on clinica to administrator;
grant insert, update, delete on adresa to administrator;
grant insert, update on procedura to administrator;
grant insert, update on medicament to administrator;
grant select, insert, update, delete on personal_medical to administrator;
grant select, insert, update, delete on departament to administrator;
grant insert, update, delete on functie to administrator;
grant create any table to administrator;
```

--Privilegii secretare

```
grant select, insert, update, delete on pacient to secretara1;
grant select on clinica to secretara1;
grant select, insert, update on adresa to secretara1;
grant select on consult to secretara1;
grant select on departament to secretara1;
grant select on functie to secretara1;
```

```
grant select, insert, update, delete on pacient to secretara2;
grant select on clinica to secretara2;
grant select, insert, update on adresa to secretara2;
```

```
grant select on consult to secretara2;
grant select on departament to secretara2;
grant select on functie to secretara2;

--Privilegii contabil
grant select on procedura to contabil;
grant select on consult_procedura to contabil;
grant select on medicament to contabil;
grant select on consult_medicament to contabil;

--Privilegii pentru pacienti, creare rol pacient
create role rol_pacient;
grant select any table to rol_pacient;

grant rol_pacient to pacient1;
grant rol_pacient to pacient2;
grant rol_pacient to pacient3;
grant rol_pacient to pacient4;
grant rol_pacient to pacient5;
grant rol_pacient to pacient6;
grant rol_pacient to pacient7;
grant rol_pacient to pacient8;
grant rol_pacient to pacient9;
grant rol_pacient to pacient10;

--Creare rol pentru asistente
create role rol_asistenta;
grant select on pacient to rol_asistenta;
grant select on clinica to rol_asistenta;
grant select on adresa to rol_asistenta;
grant select on consult to rol_asistenta;
grant select on procedura to rol_asistenta;
grant select, insert, update on consult_procedura to rol_asistenta;
grant select on medicament to rol_asistenta;
grant select on consult_medicament to rol_asistenta;
grant select on personal_medical to rol_asistenta;
grant select on departament to rol_asistenta;
grant select on functie to rol_asistenta;

grant rol_asistenta to asistenta1;
grant rol_asistenta to asistenta2;

--Creare rol pentru doctori
```

```

create role rol_doctor;
grant select on pacient to rol_doctor;
grant select on clinica to rol_doctor;
grant select on adresa to rol_doctor;
grant select, insert, update on consult to rol_doctor;
grant select on procedura to rol_doctor;
grant select, insert, update on consult_procedura to rol_doctor;
grant select on medicament to rol_doctor;
grant select, insert, update on consult_medicament to rol_doctor;
grant select on personal_medical to rol_doctor;
grant select on departament to rol_doctor;
grant select on functie to rol_doctor;

grant rol_doctor to doctor1;
grant rol_doctor to doctor2;
grant rol_doctor to doctor3;
grant rol_doctor to doctor4;

```

Acum că rolurile sunt definite, putem oferi acces la procedura de decriptare definită în capitolul 2 celor două secretare și doctorilor, dând acces la această rolului de doctor.

```

grant execute on decrypt_data to secretara1;
grant execute on decrypt_data to secretara2;
grant execute on decrypt_data to rol_doctor;

```

De asemenea, un alt scenariu posibil este acela în care secretara trebuie să încaseze banii pacientului la finalul unui consult. Pentru acest lucru are nevoie de o funcție care să calculeze costul consultului (preț consult + preț proceduri efectuate) la care să aibă acces.

```

CREATE OR REPLACE FUNCTION cost_consult (id_pac IN NUMBER) RETURN
NUMBER AS
    p_consult NUMBER(7,2) := 0;
    id_con NUMBER(4);
    p_proc NUMBER(7,2) := 0;
    suma NUMBER := 0;
BEGIN
    select pret_consult, id_consult
    into p_consult, id_con
    from system.consult
    where id_pacient = id_pac and data_consult like sysdate;

    select sum(pret)
    into p_proc
    from system.consult_procedura cp, system.procedura p

```

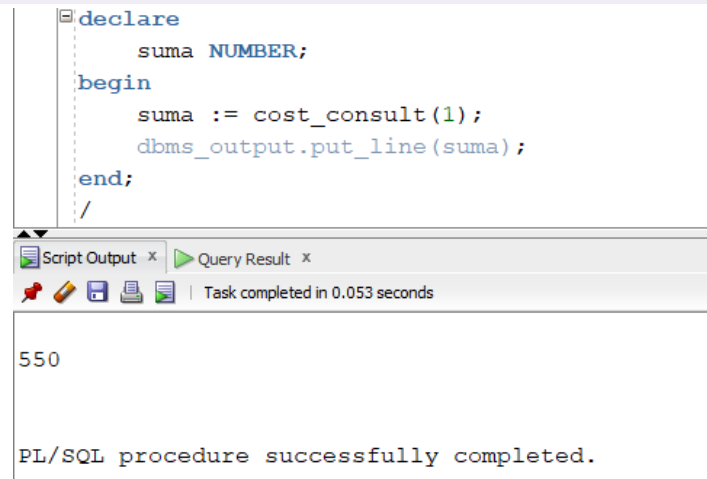
```

where id_consult = id_con and cp.id_procedura = p.id_procedura;

suma := p_proc + p_consult;
return suma;
END;
/

grant execute on cost_consult to secretara1;
grant execute on cost_consult to secretara2;

```



```

declare
    suma NUMBER;
begin
    suma := cost_consult(1);
    dbms_output.put_line(suma);
end;
/

```

Script Output x Query Result x

Task completed in 0.053 seconds

550

PL/SQL procedure successfully completed.

Pentru a exemplifica ierarhiile de privilegii, presupunem că la un moment dat o asistentă a avut voie sa realizeze un consult în lipsa unui doctor, dar acest drept i-a fost revocat.

```

grant insert on consult to asistenta1;
grant insert on consult to rol_asistenta;
grant rol_asistenta to asistenta1;
revoke insert on consult from rol_asistenta;

```

Grant succeeded.

Grant succeeded.

Grant succeeded.

Revoke succeeded.

6. Aplicațiile pe baza de date și securitatea datelor

a. Contextul aplicației

Vom crea un context care permite consulturilor să se realizeze doar între orele 8 și 20 de luni până sâmbătă, pentru a fi în cadrul programului clinicii, pentru a evita statul după program și consulturile neprogramate.

```
drop context aplicatie_medical_ctx;
create context aplicatie_medical_ctx using proced_aplicatie_ctx;

create or replace procedure proced_aplicatie_ctx is
    v_oră number(3);
    v_zi_sapt VARCHAR2(20);
begin
    select to_number(to_char(sysdate, 'hh24'))
    into v_oră
    from dual;

    select to_char(to_char(sysdate, 'DAY'))
    into v_zi_sapt
    from dual;

    dbms_output.put_line('Ora curenta: ' || v_oră);
    dbms_output.put_line('Ziua curenta: ' || v_zi_sapt);

    if v_oră < 8 or v_oră > 20 or lower(v_zi_sapt) = 'sunday' then
        dbms_output.put_line('Sunteti in afara orelor de program. ');
        dbms_session.set_context('aplicatie_medical_ctx', 'ora_ziua_potrivita', 'nu');
    else
        dbms_session.set_context('aplicatie_medical_ctx', 'ora_ziua_potrivita', 'da');
    end if;
end;
/

exec proced_aplicatie_ctx;
```

```
Context APLICATIE_MEDICAL_CTX dropped.
```

```
Context APLICATIE_MEDICAL_CTX created.
```

```
Procedure PROCED_APLICATIE_CTX compiled
```

```
Ora curenta: 21
```

```
Ziua curenta: THURSDAY
```

```
Sunteti in afara orelor de program.
```

```
PL/SQL procedure successfully completed.
```

Creăm atât un trigger de logon, cât și un trigger pentru inserarea unor noi înregistrări în tabela CONSULT care să verifice contextul.

```
create or replace trigger tr_insert_consult
before insert on consult
for each row
declare
    v_poate varchar2(4);
begin
    v_poate := sys_context('aplicatie_medical_ctx', 'ora_ziua_potrivita');
    if (v_poate = 'nu') then
        dbms_output.put_line ('Nu aveti voie sa efectuati consulturi in afara orelor de program');
        RAISE_APPLICATION_ERROR(-20000,'Incerca consult in afara orelor de program');
    end if;
end;
/

create or replace trigger tr_after_logon
after logon on database
declare
    v_user varchar2(30);
begin
    v_user := sys_context('userenv', 'session_user');

    if lower(v_user) like '%admin%' or lower(v_user) like '%doctor%' then
        proced_aplicatie_ctx;
    end if;
end;
/
```

Trigger TR_INSERT_CONSULT compiled

Trigger TR_AFTER_LOGON compiled

Am testat triggerul, încercând să inserez un consult în afara orelor de program, la ora 21.

Nu aveti voie sa efectuati consulturi in afara orelor de program

Error starting at line : 104 in command -

INSERT INTO CONSULT VALUES(9, 3, 4, 'Miopie', TO_DATE(sysdate, 'dd-MM-yyyy'), 100, 'miopie pe ambii ochi cu dioptrie -1.5')

Error report -

ORA-20000: Incercare consult in afara orelor de program

ORA-06512: at "SYSTEM.TR_INSERT_CONSULT", line 7

ORA-04088: error during execution of trigger 'SYSTEM.TR_INSERT_CONSULT'

b. SQL Injection

Pentru a exemplifica fenomenul SQL Injection, vom crea o procedura care ne ajută să obținem adresa pacientului cunoscând numele, prenumele și CNP-ul său.

```
CREATE OR REPLACE PROCEDURE adresa_pacient(p_numeuser IN VARCHAR2,
p_prenumeuser IN VARCHAR2, p_cnp IN VARCHAR2) AS
    v_oras VARCHAR2(100);
    v_judet VARCHAR2(100);
    v_strada VARCHAR2(100);
BEGIN
    EXECUTE IMMEDIATE
    'SELECT judet, oras, strada
    FROM PACIENT p, ADRESA a
    WHERE p.id_adresa = a.id_adresa AND nume="|| p_numeuser || " AND PRENUME= " ||
    p_prenumeuser || " AND CNP = "|| p_cnp
    INTO v_judet, v_oras, v_strada;

    dbms_output.put_line('Judet: ' || v_judet);
    dbms_output.put_line('Oras: ' || v_oras);
    dbms_output.put_line('Strada: ' || v_strada);
END;
/
```

Această procedură nu este deloc sigură, pentru că pot obține adresa pacientului și fără să cunosc CNP-ul său, ci doar numele.

```
begin
    adresa_pacient('Moldoveanu', 'Maria'--, '');
end;
/
```

Script Output x

Task completed in 0.079 seconds

Procedure ADRESA_PACIENT compiled

Judet: Prahova
Oras: Ploiesti
Strada: Aleea Codrului

PL/SQL procedure successfully completed.

Adăugând '--' se va comenta restul interogării și rezultatul va fi returnat. Acest lucru poate fi evitat și reparat dacă procedura va fi construită astfel:

```
CREATE OR REPLACE PROCEDURE adresa_pacient_safe(p_numeuser IN
VARCHAR2, p_prenumeuser IN VARCHAR2, p_cnp IN VARCHAR2) AS
    v_oras VARCHAR2(100);
    v_judet VARCHAR2(100);
    v_strada VARCHAR2(100);
BEGIN
    SELECT judet, oras, strada
    INTO v_judet, v_oras, v_strada
    FROM PACIENT p, ADRESA a
    WHERE p.id_adresa = a.id_adresa AND nume= p_numeuser AND prenume =
p_prenumeuser AND cnp = p_cnp;

    dbms_output.put_line('Judet: ' || v_judet);
    dbms_output.put_line('Oras: ' || v_oras);
    dbms_output.put_line('Strada: ' || v_strada);
END;
/
```

```
begin
    adresa_pacient_safe('Moldoveanu', 'Maria'--, '');
end;
/
```

Script Output x

Task completed in 0.084 seconds

Procedure ADRESA_PACIENT_SAFE compiled

Error starting at line : 98 in command -

```
begin
    adresa_pacient_safe('Moldoveanu', 'Maria'--, '');
end;
```

Error report -

ORA-01403: no data found

ORA-06512: at "SYSTEM.ADRESA_PACIENT_SAFE", line 6

ORA-06512: at line 2

01403. 00000 - "no data found"

*Cause: No data was found from the objects.

*Action: There was no data from the objects which may be due to end of fetch.