Ianuarie 2023

Universitatea din București, Facultatea de Matematică și Informatică

DLarisa

Database Security

Contents

[**1. Introducere:** 2](#_Toc126253354)

[*1.1. Scurtă Prezentare* 2](#_Toc126253355)

[*1.2. Diagramă Conceptuală* 2](#_Toc126253356)

[*1.3. Scheme Relaționale* 3](#_Toc126253357)

[*1.4. Crearea Tabelelor* 5](#_Toc126253358)

[**2. Gestiunea Utilizatorilor unei Baze de Date și a Resurselor Computaționale** 7](#_Toc126253359)

[*2.1. Proiectarea configurației de management a identităților în baza de date* 7](#_Toc126253360)

[Procesele Aplicației: 7](#_Toc126253361)

[Matricea Proces-Utilizator: 7](#_Toc126253362)

[Matricea Entitate-Proces: 7](#_Toc126253363)

[Matricea Entitate-Utilizator: 8](#_Toc126253364)

[*2.2. Configurație de Management a Identităților* 8](#_Toc126253365)

[Memorie Alocată: 8](#_Toc126253366)

[Constrângeri de Sesiune: 8](#_Toc126253367)

[Alocare CPU (Plan de Consum): 8](#_Toc126253368)

[*2.3. Cod* 9](#_Toc126253369)

[**3. Privilegii și Roluri** 13](#_Toc126253370)

[**4. Criptarea Datelor** 17](#_Toc126253371)

[**5. Audit** 20](#_Toc126253372)

[*5.1. Audit Standard* 21](#_Toc126253373)

[*5.2. Triggers de Auditare* 22](#_Toc126253374)

[*5.3. Politici de Auditare* 23](#_Toc126253375)

[**6. Aplicațiile pe baza de date și securitatea datelor** 24](#_Toc126253376)

[**7. Mascarea Datelor** 26](#_Toc126253377)

# **1. Introducere:**

## *1.1. Scurtă Prezentare*

Proiectul va prezenta baza de date a unei clinici medicale private, care oferă diferite servicii pentru clienții săi. Userii acestei aplicații sunt doctorii, asistentele, pacienții, publicul general și adminul aplicației.

* În cadrul acestei clinici private există mai multe secții, cum ar fi *Oftalmologie, Cardiologie, Neurologie,* etc... . Fiecare secție oferă diferite proceduri, cum ar fi *Cardiogramă, Test de Vedere, Ecografie*, etc..., care pot fi realizate doar de medici ce aparțin secției respective. Fiecare procedură este realizată pe parcursul unei ore.
* Clinica are o suită vastă de doctori bine pregătiți în diferite specializări: aceștia aparțin câte unei secții. Există un doctor șef pentru fiecare secție. Doctorii au un program bine definit și lucrează zilnic, între un anumit interval orar.
* Pacienții doresc să beneficieze de procedurile acestei clinici. Pentru aceasta, iau legătura (prin telefon, mail, prezentare personală) cu una dintre asistente, care programează o consultație, sau folosesc direct platforma online a clinicii.
* În urma unei consultații, doctorul poate prescrie o rețetă pentru pacient.

## *1.2. Diagramă Conceptuală*

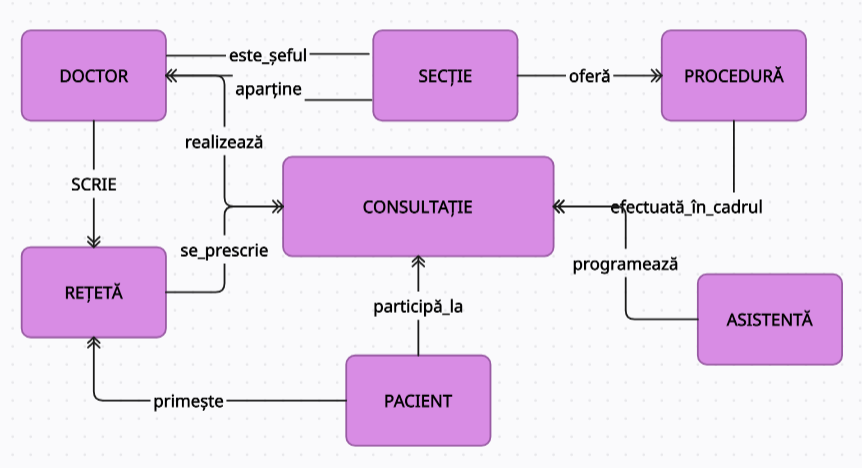
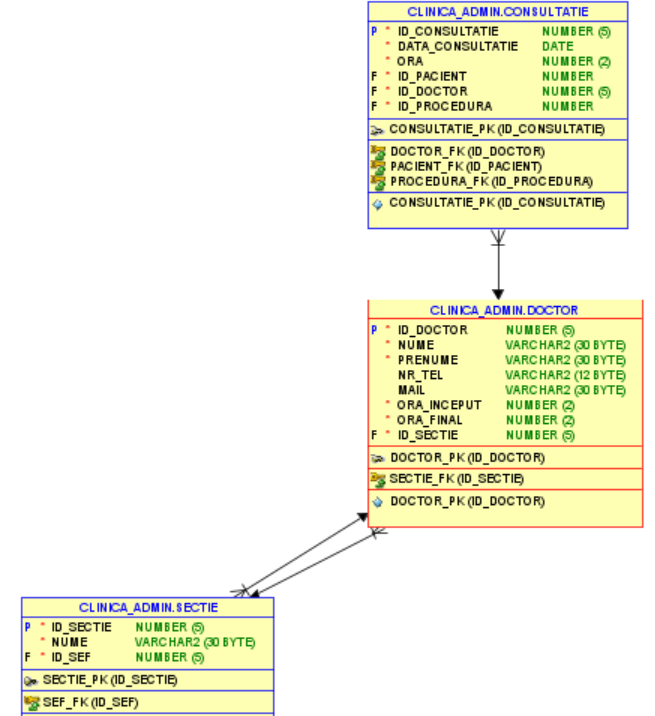


Diagrama conceptuală este formată din tabelele **DOCTOR, SECȚIE, PROCEDURĂ, REȚETĂ, CONSULTAȚIE, PACIENT, ASISTENTĂ**.

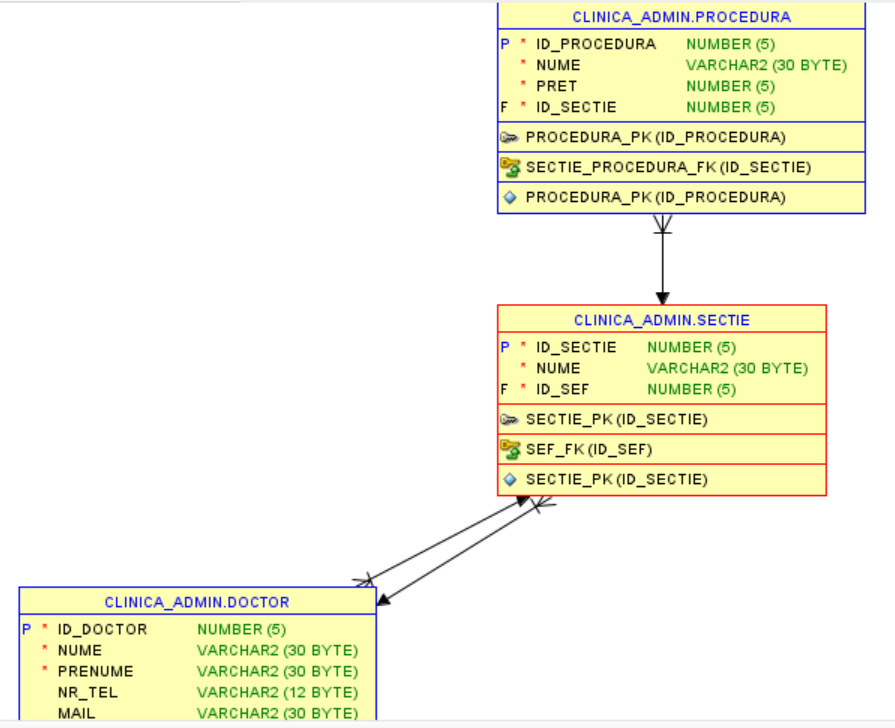
## *1.3. Scheme Relaționale*

Vom prezenta în continuare schema fiecărui tabel în parte:

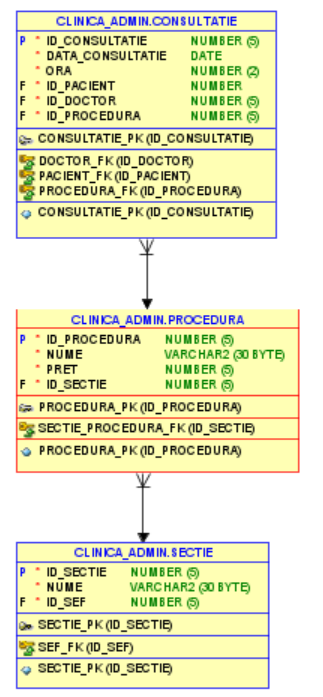
* **DOCTOR** (id\_doctor# (**PK**), nume, prenume, nr\_tel, mail, ora\_început, ora\_final, id\_secție (*FK*));



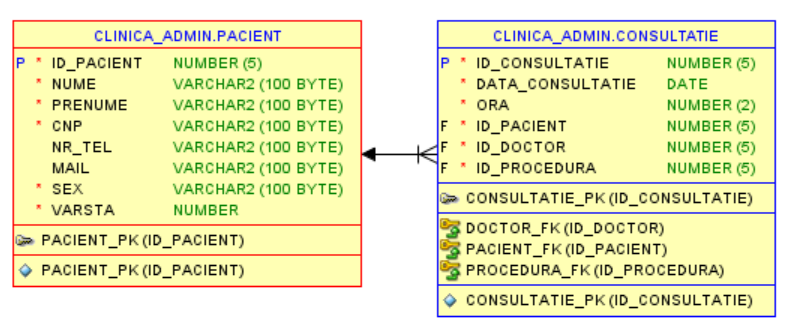
* **SECȚIE** (id\_secție# (**PK**), nume, id\_șef (*FK*));



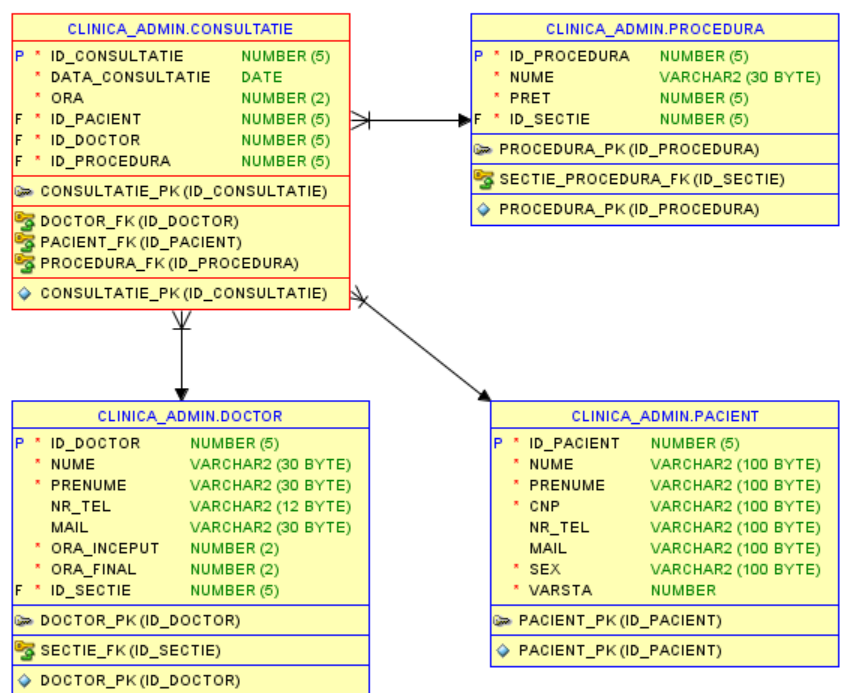
* **PROCEDURĂ** (id\_procedură# (**PK**), nume, preț, id\_secție (*FK*));



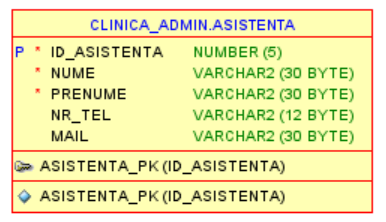
* **PACIENT** (id\_pacient# (**PK**), nume, prenume, CNP, nr\_tel, mail, sex, vârstă);



* **CONSULTAȚIE** (id\_consultație# (**PK**), dată, oră, id\_pacient (*FK*), id\_doctor (*FK*), id\_procedură (*FK*));

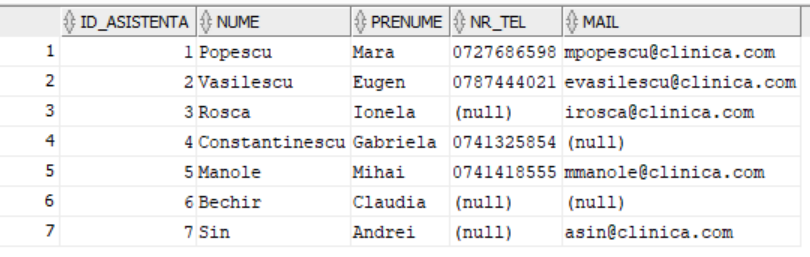


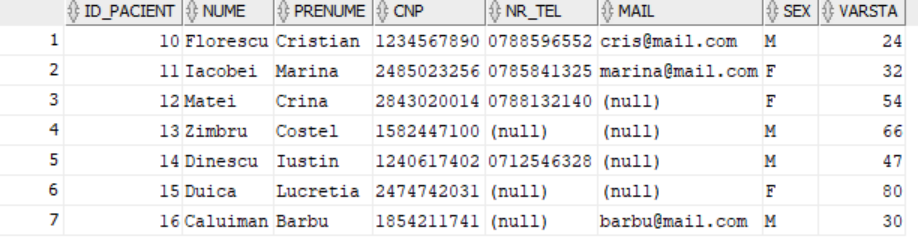
* **REȚETĂ** (id\_rețetă# (**PK**), medicament, indicații, reducere, id\_doctor (*FK*), id\_consultație (*FK*));
* **ASISTENTĂ** (id\_asistentă# (**PK**), nume, prenume, nr\_tel, mail).



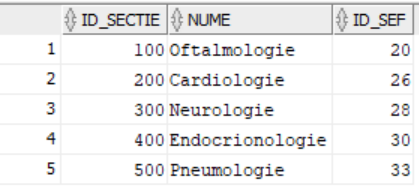
## *1.4. Crearea Tabelelor*

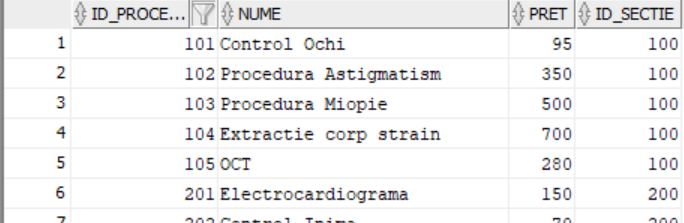
Atașăm screenshot-uri cu datele din tabelele create.

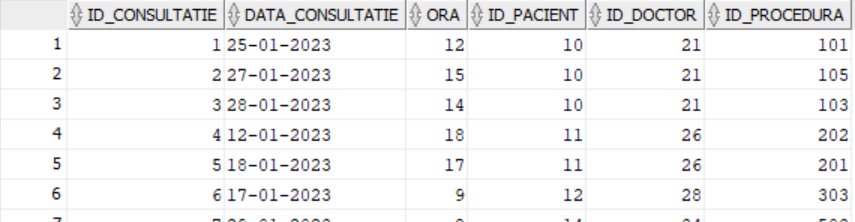


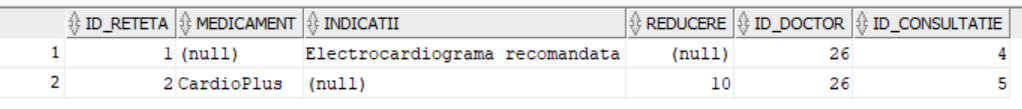












# **2. Gestiunea Utilizatorilor unei Baze de Date și a Resurselor Computaționale**

## *2.1. Proiectarea configurației de management a identităților în baza de date*

### Procesele Aplicației:

Procesele care pot fi inițiate în cadrul aplicației sunt:

P1: Configurarea userilor;

P2: Configurarea secțiilor medicale;

P3: Numirea unui doctor drept șef de secție / Revocarea unui doctor drept șef de secție;

P4: Adăugarea/Ștergerea unui doctor dintr-o secție;

P5: Configurarea procedurilor medicale;

P6: Configurarea consultațiilor medicale;

P7: Configurarea rețetelor medicale;

P8: Vizualizarea rețetelor medicale ale unui pacient;

P9: Vizualizarea istoricului de consultații al unui pacient;

P10: Vizualizarea consultațiilor programate pentru un doctor;

P11: Vizualizarea doctorilor și orelor de lucru ale acestora;

P12: Vizualizarea tuturor procedurilor din clinică / procedurilor dintr-o secție anume;

P13: Vizualizarea tuturor secțiilor din clinică;

P14: Vizualizarea tuturor doctorilor din clinică.

### Matricea Proces-Utilizator:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11 | P12 | P13 | P14 |
| Doctor |  |  |  |  |  |  | **×** | **×** | **×** | **×** |  | **×** | **×** | **×** |
| Șef Secție |  |  |  | **×** | **×** |  |  | **×** | **×** | **×** | **×** | **×** | **×** | **×** |
| Asistentă |  |  |  |  |  | **×** |  |  | **×** | **×** | **×** | **×** | **×** | **×** |
| Pacient |  |  |  |  |  | **×** |  | **×** | **×** |  |  | **×** | **×** | **×** |
| Public General |  |  |  |  |  |  |  |  |  |  |  | **×** | **×** | **×** |
| Admin | **×** | **×** | **×** |  |  |  |  |  |  |  |  | **×** | **×** | **×** |

### Matricea Entitate-Proces:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11 | P12 | P13 | P14 |
| Doctor | I,U |  |  | U |  |  |  |  |  | S | S |  |  | S |
| Asistentă | I,U |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Pacient | I,U |  |  |  |  |  |  | S | S |  |  |  |  |  |
| Secție |  | I,U | U |  |  |  |  |  |  |  |  | S | S | S |
| Procedură |  |  |  |  | I,U |  |  |  |  |  |  | S |  |  |
| Consultație |  |  |  |  |  | I,U,D |  |  | S | S |  |  |  |  |
| Rețetă |  |  |  |  |  |  | I,U,D | S |  |  |  |  |  |  |

Legendă: I = Insert; U = Update; D = Delete; S = Select

### Matricea Entitate-Utilizator:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Doctor | Șef Secție | Asistentă | Pacient | Public | Admin |
| Doctor | S | U,S | S | S | S | I,U,S |
| Asistentă |  |  |  |  |  | I,U,S |
| Pacient | S | S |  | S |  | I,U |
| Secție | S | S | S | S | S | I,U,S |
| Procedură | S | I,U,S | S | S | S | S |
| Consultație | S | S | I,U,D,S | I,U,D,S |  |  |
| Rețetă | I,U,D,S | I,U,D,S |  | S |  |  |

Legendă: I = Insert; U = Update; D = Delete; S = Select

## *2.2. Configurație de Management a Identităților*

### Memorie Alocată:

* Admin-ul are acces nelimitat la resurse (adică, memorie nelimitată), pentru că este proprietarul majorității tabelelor din schemă.
* Pentru fiecare doctor, este alocată o memorie de 10MB pentru că ei dețin tabelul **REȚETĂ** în schema lor.
* Pentru restul de utilizatori (pacient, asistente, public) nu se alocă spațiu de memorie, pentru că schemele lor sunt goale (ei realizează query-uri pe tabelele din baza de date a adminului sau a doctorului).

### Constrângeri de Sesiune:

* Pentru doctori, asistente: 1 sesiune / user (fiecare are un cont personal, generat de admin), parola trebuie să fie schimbată o dată la 3 luni, au voie să greșească credențialele de login de maxim 3 ori și 6000 putere de CPU.
* Pentru pacienți: 1 sesiune / user (fiecare are un cont personal, generat de admin), parola nu trebuie să fie schimbată întrucât este CNP-ul lor, au voie să greșească credențialele de login de 10 ori și 6000 putere de CPU.
* Pentru publicul general: 3 sesiuni / user, idle time 5 minute și perioada alocată unei conexiuni este de maxim 30 de min.

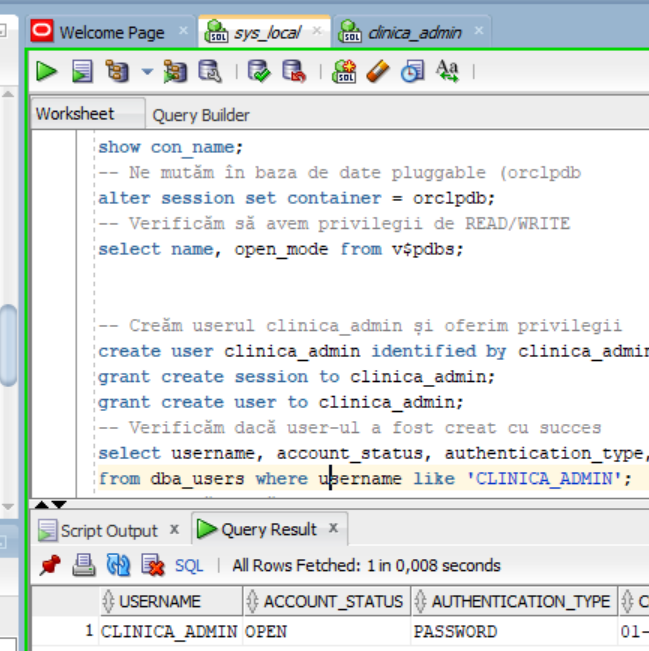
### Alocare CPU (Plan de Consum):

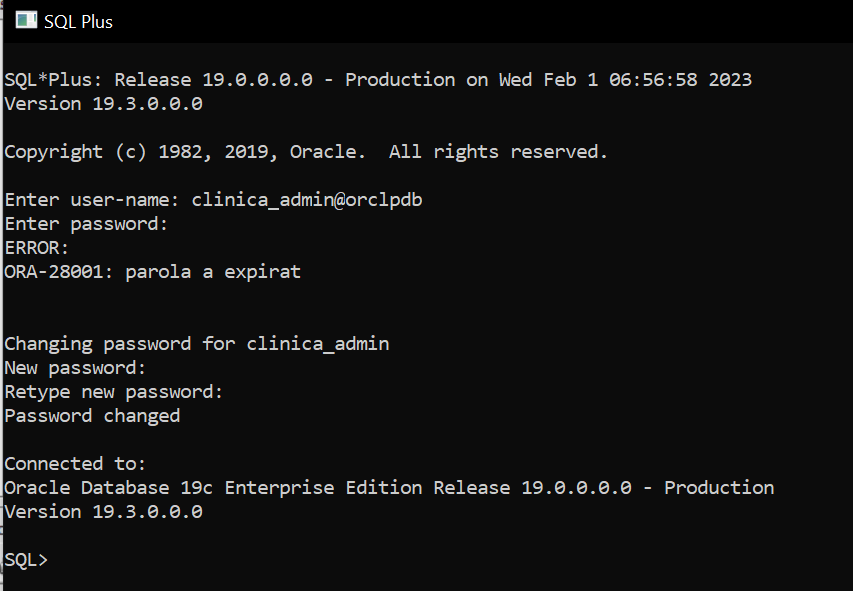
Atunci când CPU-ul ajunge la 100%, definim următoarele reguli de folosire a CPU-lui:

* Admin: 30%;
* Doctori și Asistente: 35%;
* Pacienți: 25%;
* Publicul General: 10%

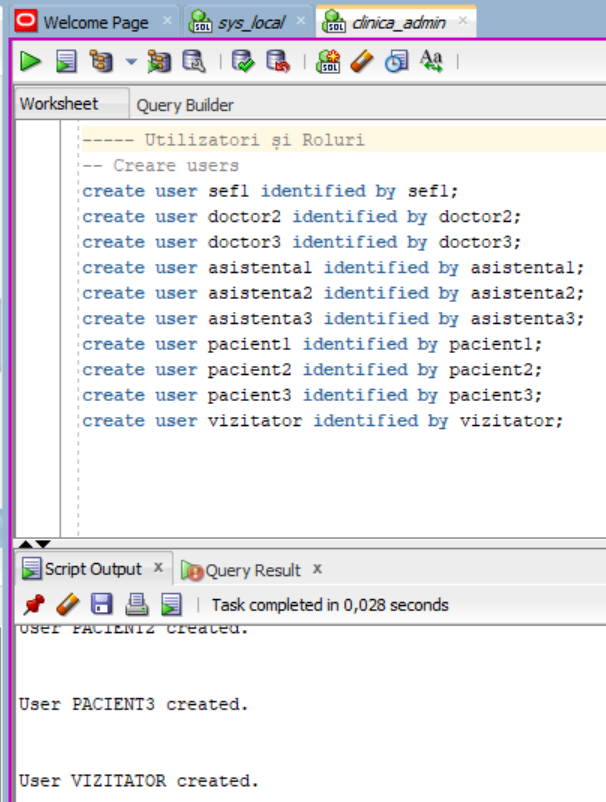
## *2.3. Cod*

Aplicația este administrată de user-ul ***clinica\_admin***. Acesta a fost creat de ***sys\_local*** și a primit privilegii de **CREATE SESSION** și de **CREATE USER**, pentru a putea să se conecteze și să creeze conturile de utilizatori. La prima conectare a sa, va trebui să își schimbe parola.

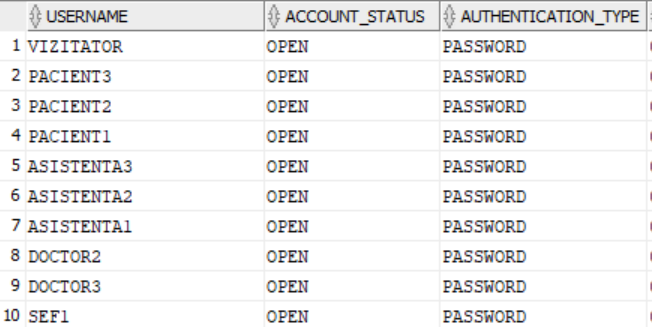




În continuare, ***clinica\_admin*** va crea userii care vor folosi această bază de date: ***sef1, doctor2, doctor3, doctor4, asistenta1, asistenta2, asistenta3, pacient1, pacient2, pacient3, vizitator***.



În ***sys\_local*** putem verifica dacă userii au fost creați cu succes:



În ***sys\_local*** oferim resurse de memorie utilizatorilor, așa cum este definit în secțiunea [2.2](#_Memorie_Alocată:):

-- Memorie

**alter** **user** clinica\_admin quota unlimited **on** users;

**alter** **user** vizitator quota **0**M **on** users;

**alter** **user** asistenta1 quota **0**M **on** users;

**alter** **user** asistenta2 quota **0**M **on** users;

**alter** **user** asistenta3 quota **0**M **on** users;

**alter** **user** pacient1 quota **0**M **on** users;

**alter** **user** pacient2 quota **0**M **on** users;

**alter** **user** pacient3 quota **0**M **on** users;

**alter** **user** sef1 quota **5**M **on** users;

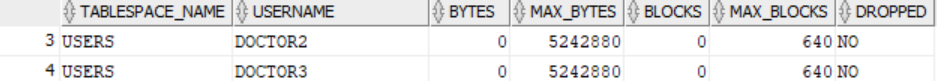
**alter** **user** doctor2 quota **5**M **on** users;

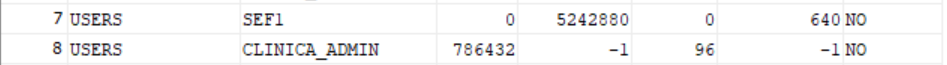
**alter** **user** doctor3 quota **5**M **on** users;

**alter** **user** doctor4 quota **5**M **on** users;

-- Vedem informațiile despre memoria alocată

**select** \* **from** dba\_ts\_quotas **where** tablespace\_name **like** 'USERS';





În ***sys\_local*** creăm profile pentru management-ul resurselor bazei de date, așa cum este definit în secțiunea [2.2](#_Constrângeri_de_Sesiune:):

-- Profile (detalii sesiune, idle\_time, failed\_login\_attempts, password\_life\_time, etc...)

**create** profile profil\_vizitator **limit**

sessions\_per\_user **3**

idle\_time **5**

connect\_time **30**;

**alter** **user** vizitator profile profil\_vizitator;

**create** profile profil\_angajat **limit**

sessions\_per\_user **1**

password\_life\_time **90**

failed\_login\_attempts **3**

cpu\_per\_call **6000**

idle\_time **15**;

**alter** **user** sef1 profile profil\_angajat;

**alter** **user** doctor2 profile profil\_angajat;

**alter** **user** doctor3 profile profil\_angajat;

**alter** **user** doctor4 profile profil\_angajat;

**alter** **user** asistenta1 profile profil\_angajat;

**alter** **user** asistenta2 profile profil\_angajat;

**alter** **user** asistenta3 profile profil\_angajat;

**create** profile profil\_pacient **limit**

sessions\_per\_user **1**

failed\_login\_attempts **10**

cpu\_per\_call **6000**

idle\_time **15**;

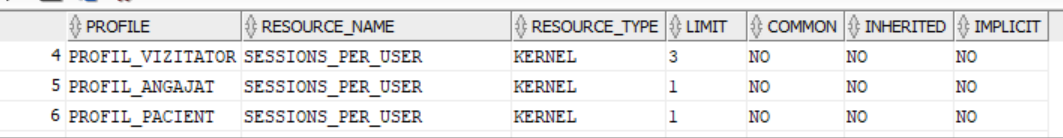
**alter** **user** pacient1 profile profil\_pacient;

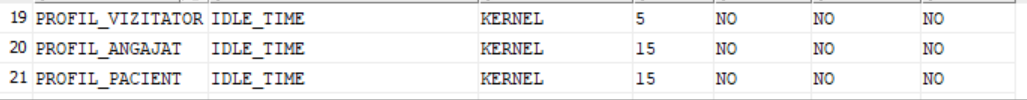
**alter** **user** pacient2 profile profil\_pacient;

**alter** **user** pacient3 profile profil\_pacient;

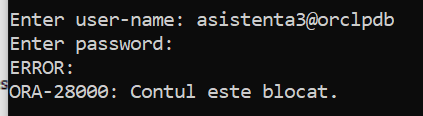
-- Vizualizare informații despre profile

**select** \* **from** dba\_profiles **where** profile **like** 'PROFIL%';

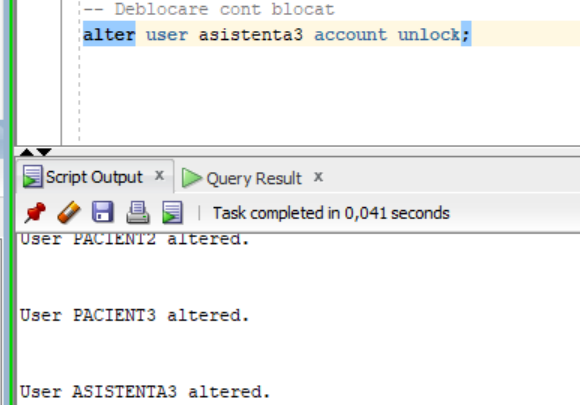
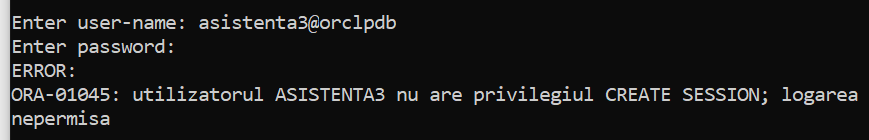




Testăm și introducem parola greșită de 3 ori pentru contul ***asistenta3***:



Pentru a debloca acest cont, din ***sys\_local*** rulăm următoarea comandă (urmează să oferim privilegii, momentan nu ne lasă să ne conectăm la baza de date):

În ***sys\_local*** creăm planul de consum, așa cum este el definit în secțiunea [2.2](#_Alocare_CPU_(Plan):

-- Plan de Consum CPU

**create** **or** **replace** **procedure** clinica\_plan\_cpu **as**

nr number := **0**;

**begin**

-- Creare zona de lucru și plan

dbms\_resource\_manager.create\_pending\_area();

dbms\_resource\_manager.create\_plan(plan => 'CLINICA\_PLAN\_CPU', **comment** => 'Plan de consumptie pentru clinica privata.');

-- Grupuri de consum

dbms\_resource\_manager.create\_consumer\_group(consumer\_group => 'admin', **comment** => 'Grup de sesiune pt administratori bd.');

dbms\_resource\_manager.create\_consumer\_group(consumer\_group => 'angajati', **comment** => 'Grup de sesiune pt angajati.');

dbms\_resource\_manager.create\_consumer\_group(consumer\_group => 'pacienti', **comment** => 'Grup de sesiune pt pacienti.');

-- Va fi creat dacă nu există deja

**select** **count**(\*) **into** nr

**from** dba\_rsrc\_consumer\_groups

**where** consumer\_group = 'OTHER\_GROUPS';

if nr = **0** **then**

dbms\_resource\_manager.create\_consumer\_group(consumer\_group => 'OTHER\_GROUPS', **comment** => 'Grup de sesiune pt vizitatori.');

**end** if;

-- Mapare între grupuri de consum și users (users nu pot fi OTHER\_GROUPS)

dbms\_resource\_manager.set\_consumer\_group\_mapping(dbms\_resource\_manager.oracle\_user, 'clinica\_admin', 'admin');

dbms\_resource\_manager.set\_consumer\_group\_mapping(dbms\_resource\_manager.oracle\_user, 'sef1', 'angajati');

dbms\_resource\_manager.set\_consumer\_group\_mapping(dbms\_resource\_manager.oracle\_user, 'doctor2', 'angajati');

dbms\_resource\_manager.set\_consumer\_group\_mapping(dbms\_resource\_manager.oracle\_user, 'doctor3', 'angajati');

dbms\_resource\_manager.set\_consumer\_group\_mapping(dbms\_resource\_manager.oracle\_user, 'doctor4', 'angajati');

dbms\_resource\_manager.set\_consumer\_group\_mapping(dbms\_resource\_manager.oracle\_user, 'asistenta1', 'angajati');

dbms\_resource\_manager.set\_consumer\_group\_mapping(dbms\_resource\_manager.oracle\_user, 'asistenta2', 'angajati');

dbms\_resource\_manager.set\_consumer\_group\_mapping(dbms\_resource\_manager.oracle\_user, 'asistenta3', 'angajati');

dbms\_resource\_manager.set\_consumer\_group\_mapping(dbms\_resource\_manager.oracle\_user, 'pacient1', 'pacienti');

dbms\_resource\_manager.set\_consumer\_group\_mapping(dbms\_resource\_manager.oracle\_user, 'pacient2', 'pacienti');

dbms\_resource\_manager.set\_consumer\_group\_mapping(dbms\_resource\_manager.oracle\_user, 'pacient3', 'pacienti');

-- Planificare directive (CPU) pentru fiecare grup de consum

dbms\_resource\_manager.create\_plan\_directive(plan => 'CLINICA\_PLAN\_CPU', group\_or\_subplan => 'admin',

**comment** => 'Planificare directiva pt administratori.', MGMT\_P1 => **30**);

dbms\_resource\_manager.create\_plan\_directive(plan => 'CLINICA\_PLAN\_CPU', group\_or\_subplan => 'angajati',

**comment** => 'Planificare directiva pt angajati.', MGMT\_P1 => **35**);

dbms\_resource\_manager.create\_plan\_directive(plan => 'CLINICA\_PLAN\_CPU', group\_or\_subplan => 'pacienti',

**comment** => 'Planificare directiva pt pacienti.', MGMT\_P1 => **25**);

dbms\_resource\_manager.create\_plan\_directive(plan => 'CLINICA\_PLAN\_CPU', group\_or\_subplan => 'OTHER\_GROUPS',

**comment** => 'Planificare directiva pt vizitatori.', MGMT\_P1 => **10**);

-- Validare zona de lucru și submit

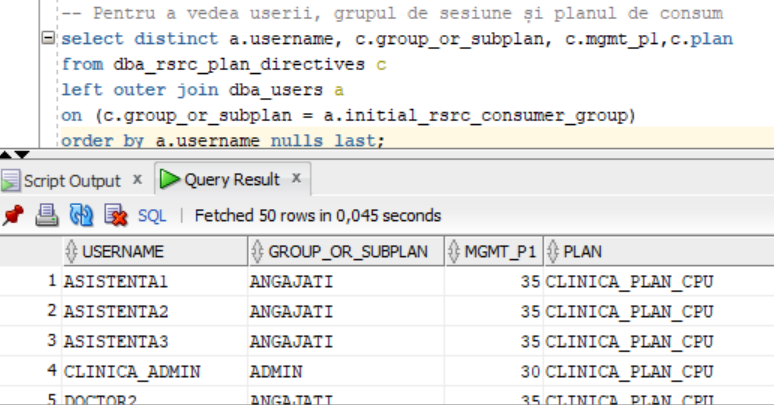
dbms\_resource\_manager.validate\_pending\_area();

dbms\_resource\_manager.submit\_pending\_area();

**end**;

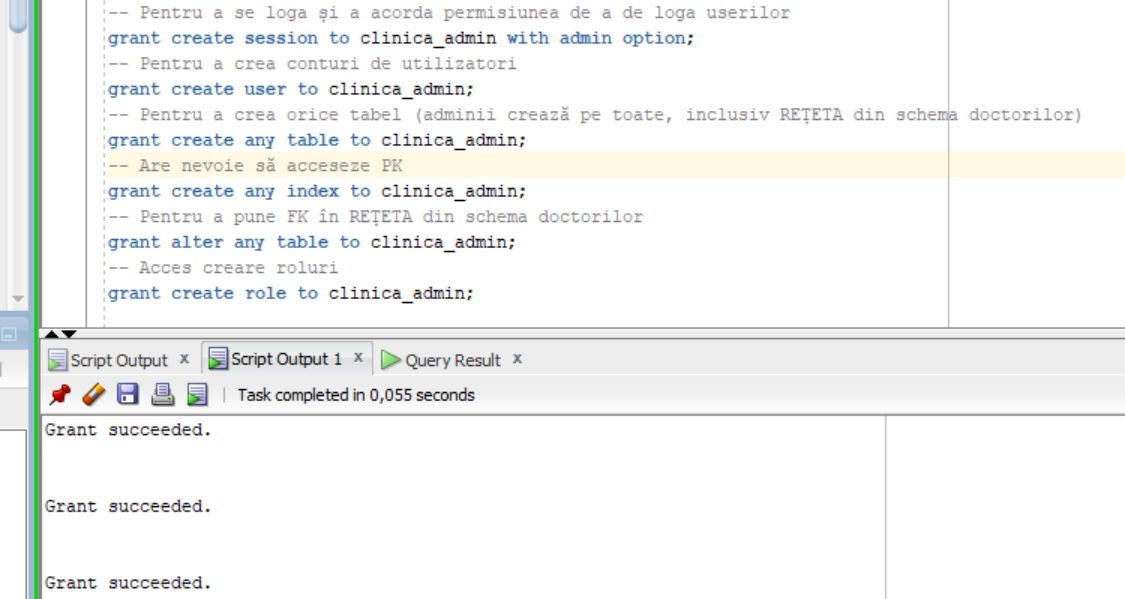
/

**execute** clinica\_plan\_cpu;

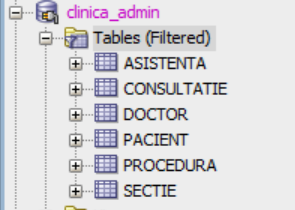


# **3. Privilegii și Roluri**

În ***sys\_local*** vom acorda privilegii utilizatorului ***clinica\_admin***: acesta are dreptul de a crea toate tabelele bazei de date (inclusiv **REȚETA** din schema doctorilor) și de a crea roluri pentru utilizatori.



Acum lucrăm în ***clinica\_admin***. Mai întâi creăm tabelele bazei de date de care avem nevoie (inclusiv **REȚETA** din schema doctorilor, dar nu sunt vizibile în poza de mai jos):



Acum, ***clinica\_admin*** va crea rolurile (cu permisiunile necesare), definite ca în secțiunea [2.1](#_Matricea_Entitate-Utilizator:), pentru utilizatorii bazei de date și le va asigna fiecărui cont în parte.

-- Creare Roluri

**create** **role** doctor;

**create** **role** sef\_sectie;

**create** **role** asistenta;

**create** **role** pacient;

**create** **role** public\_general;

În ***sys\_local*** oferim lui ***clinica\_admin*** următoarele privilegii:

----- Roluri și Privilegii

-- Permisiuni Admin (tb sa permita la alți useri sa faca operții pe rețete)

**grant** **select**, **update**, **insert**, **delete** **on** sef1.reteta **to** clinica\_admin **with** **grant** **option**;

**grant** **select**, **update**, **insert**, **delete** **on** doctor2.reteta **to** clinica\_admin **with** **grant** **option**;

**grant** **select**, **update**, **insert**, **delete** **on** doctor3.reteta **to** clinica\_admin **with** **grant** **option**;

**grant** **select**, **update**, **insert**, **delete** **on** doctor4.reteta **to** clinica\_admin **with** **grant** **option**;

În ***clinica\_admin*** acordăm următoarele privilegii:

--- Alocare Permisiuni

-- Permisiune de Creare Sesiuni

**grant** **create** **session** **to** doctor;

**grant** **create** **session** **to** sef\_sectie;

**grant** **create** **session** **to** asistenta;

**grant** **create** **session** **to** pacient;

**grant** **create** **session** **to** public\_general;

-- Permisiuni Doctor

**grant** **select** **on** clinica\_admin.doctor **to** doctor;

**grant** **select** **on** clinica\_admin.pacient **to** doctor;

**grant** **select** **on** clinica\_admin.sectie **to** doctor;

**grant** **select** **on** clinica\_admin.procedura **to** doctor;

**grant** **select** **on** clinica\_admin.consultatie **to** doctor;

-- Permisiuni Șef Secție

**grant** **select**, **update** **on** clinica\_admin.doctor **to** sef\_sectie;

**grant** **select** **on** clinica\_admin.pacient **to** sef\_sectie;

**grant** **select** **on** clinica\_admin.sectie **to** sef\_sectie;

**grant** **select**, **update**, **insert** **on** clinica\_admin.procedura **to** sef\_sectie;

**grant** **select** **on** clinica\_admin.consultatie **to** sef\_sectie;

-- Permisiuni Asistenta

**grant** **select** **on** clinica\_admin.doctor **to** asistenta;

**grant** **select** **on** clinica\_admin.sectie **to** asistenta;

**grant** **select** **on** clinica\_admin.procedura **to** asistenta;

**grant** **select**, **update**, **insert**, **delete** **on** clinica\_admin.consultatie **to** asistenta;

-- Permisiuni Pacient

**grant** **select** **on** clinica\_admin.doctor **to** pacient;

**grant** **select** **on** clinica\_admin.pacient **to** pacient;

**grant** **select** **on** clinica\_admin.sectie **to** pacient;

**grant** **select** **on** clinica\_admin.procedura **to** pacient;

**grant** **select**, **update**, **insert**, **delete** **on** clinica\_admin.consultatie **to** pacient;

**grant** **select** **on** sef1.reteta **to** pacient;

**grant** **select** **on** doctor2.reteta **to** pacient;

**grant** **select** **on** doctor3.reteta **to** pacient;

**grant** **select** **on** doctor4.reteta **to** pacient;

-- Permisiuni Vizitator (Public General)

**grant** **select** **on** clinica\_admin.doctor **to** public\_general;

**grant** **select** **on** clinica\_admin.sectie **to** public\_general;

**grant** **select** **on** clinica\_admin.procedura **to** public\_general;

-- Permisiuni speciale pt tabel REȚETA (șefii de secție pot vedea toate rețetele colegilor din subordinea lor)

**grant** **select**, **update**, **insert**, **delete** **on** sef1.reteta **to** sef\_sectie;

**grant** **select**, **update**, **insert**, **delete** **on** doctor2.reteta **to** sef\_sectie;

**grant** **select**, **update**, **insert**, **delete** **on** doctor3.reteta **to** sef\_sectie;

**grant** **select**, **update**, **insert**, **delete** **on** doctor4.reteta **to** sef\_sectie;

--- Asignare Roluri userilor

**grant** doctor **to** doctor2;

**grant** doctor **to** doctor3;

**grant** doctor **to** doctor4;

**grant** sef\_sectie **to** sef1;

**grant** asistenta **to** asistenta1;

**grant** asistenta **to** asistenta2;

**grant** asistenta **to** asistenta3;

**grant** pacient **to** pacient1;

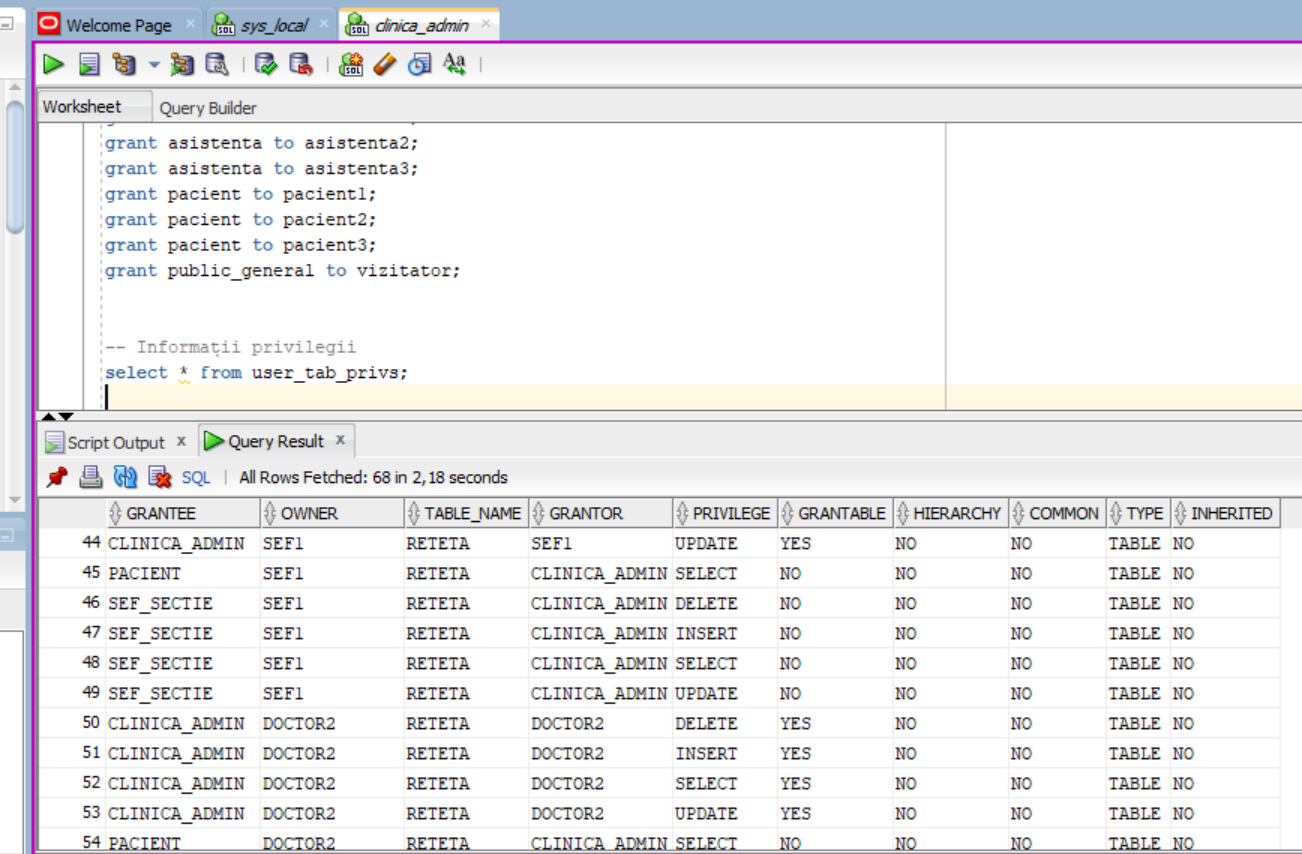
**grant** pacient **to** pacient2;

**grant** pacient **to** pacient3;

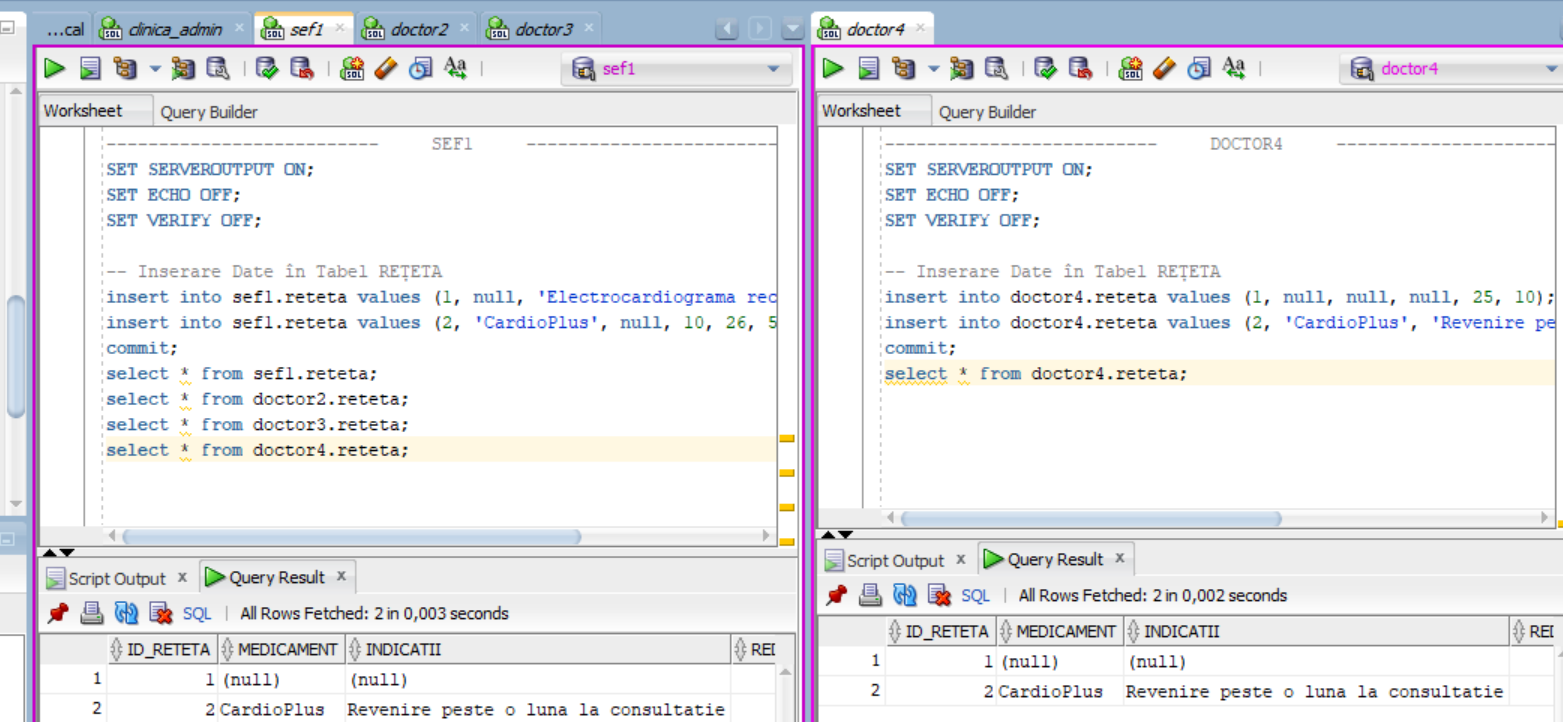
**grant** public\_general **to** vizitator;

-- Informații privilegii

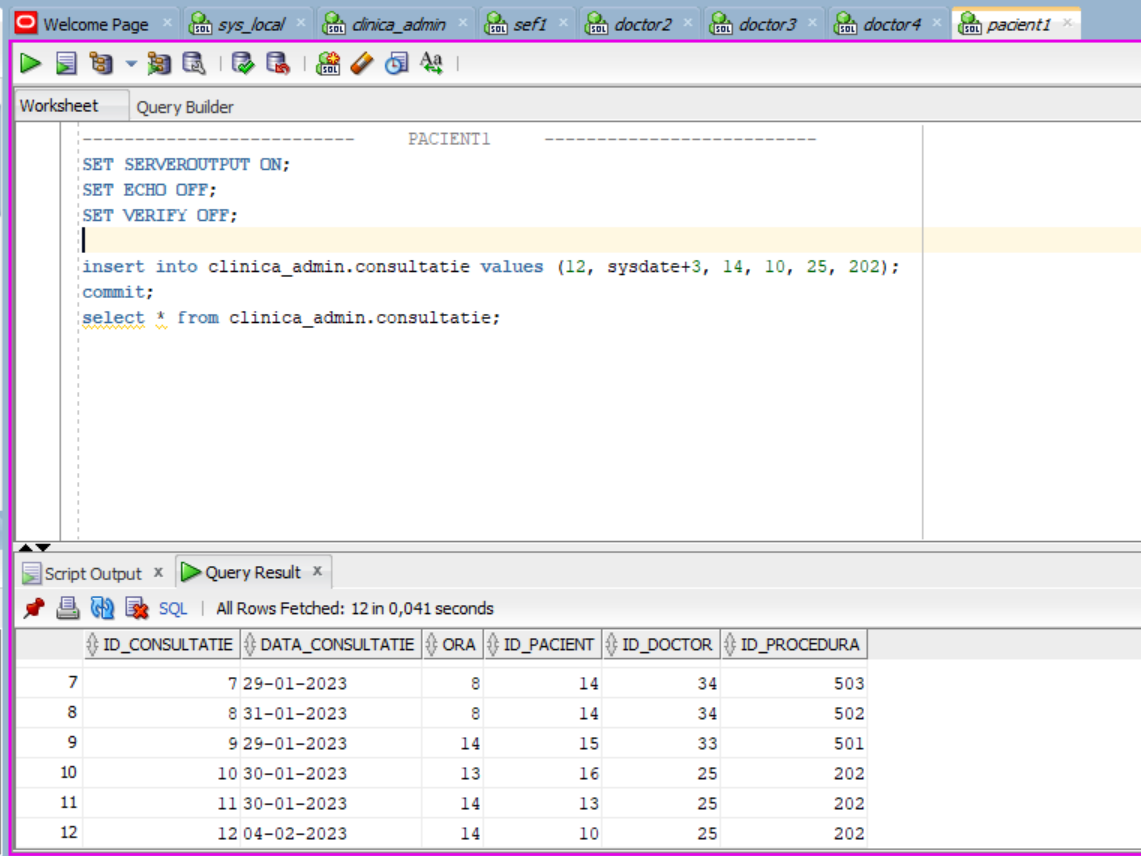
**select** \* **from** user\_tab\_privs;



Acum putem să ne conectăm la sesiunile doctorilor (***sef1, doctor2, doctor3, doctor4***) și să inserăm datele în tabelul **REȚETA**. Conform regulilor definite de noi, ***sef1*** poate vedea toate tabelele cu rețete, de la toți doctorii, dar doctorii pot vedea doar tabelele proprii.



Un alt exemplu, pentru a ne testa:



# **4. Criptarea Datelor**

Criptarea datelor se va realiza pe tabelul **PACIENT**, datorită detaliilor cu caracter personal, care trebuiesc protejate. Se va crea un tabel nou **pacient\_criptat** în care vor fi memorate id\_pacient, prenumele, cnp-ul și sexul pacientului, criptate cu *AES, pad\_pkcs5* și *chain\_cbc*, dar și valoarea criptată a hash-ului rezultat din concatenarea celor 4 valori originale. În mod similar, se va crea o procedură de decriptare, care va și compara integritatea datelor.

**drop** **table** chei;

**create** **table** chei (

id\_chei number,

cheie raw(**16**) **not** **null**,

nume\_tabel varchar2(**30**) **not** **null**,

**constraint** pk\_chei **primary** **key** (id\_chei)

);

**drop** **table** pacient\_criptat;

**create** **table** pacient\_criptat (

id\_pacient varchar2(**100**),

prenume varchar2(**100**) **not** **null**,

cnp varchar2(**100**) **not** **null**,

sex varchar2(**100**) **not** **null**,

integritate varchar2 (**100**),

**constraint** pacient\_criptat\_pk **primary** **key** (id\_pacient)

);

-- Procedura va crea un tabel nou care va reține datele criptate cu AES și un hash al datelor tot criptat cu AES

**create** **or** **replace** **procedure** criptare\_pacient **as**

cheie\_crypt raw(**16**);

operation\_mode pls\_integer;

-- cursorul care parcurge liniile tabelului PACIENT

**cursor** **c** **is** **select** \* **from** pacient;

raw\_id raw(**100**);

raw\_prenume raw(**100**);

raw\_cnp raw(**100**);

raw\_sex raw(**100**);

raw\_integritate raw(**500**);

encrypt\_id raw(**100**);

encrypt\_prenume raw(**100**);

encrypt\_cnp raw(**100**);

encrypt\_sex raw(**100**);

encrypt\_integritate raw(**500**);

concat\_val varchar2(**500**);

raw\_concat\_val raw(**500**);

**begin**

cheie\_crypt := dbms\_crypto.randombytes(**16**);

**insert** **into** chei **values** (**1**, cheie\_crypt, 'PACIENT');

operation\_mode := dbms\_crypto.encrypt\_aes128 + dbms\_crypto.pad\_pkcs5 + dbms\_crypto.chain\_cbc;

**for** i **in** **c** loop

raw\_id := utl\_i18n.string\_to\_raw(i.id\_pacient, 'AL32UTF8');

raw\_prenume := utl\_i18n.string\_to\_raw(i.prenume, 'AL32UTF8');

raw\_cnp := utl\_i18n.string\_to\_raw(i.cnp, 'AL32UTF8');

raw\_sex := utl\_i18n.string\_to\_raw(i.sex, 'AL32UTF8');

encrypt\_id := dbms\_crypto.encrypt(raw\_id, operation\_mode, cheie\_crypt);

encrypt\_prenume := dbms\_crypto.encrypt(raw\_prenume, operation\_mode, cheie\_crypt);

encrypt\_cnp := dbms\_crypto.encrypt(raw\_cnp, operation\_mode, cheie\_crypt);

encrypt\_sex := dbms\_crypto.encrypt(raw\_sex, operation\_mode, cheie\_crypt);

concat\_val := i.id\_pacient || i.prenume || i.cnp || i.sex;

raw\_concat\_val := utl\_i18n.string\_to\_raw(concat\_val);

raw\_integritate := dbms\_crypto.hash(raw\_concat\_val, dbms\_crypto.hash\_md5);

encrypt\_integritate := dbms\_crypto.encrypt(raw\_integritate, operation\_mode, cheie\_crypt);

**insert** **into** pacient\_criptat **values** (encrypt\_id, encrypt\_prenume, encrypt\_cnp, encrypt\_sex, encrypt\_integritate);

**end** loop;

**commit**;

**end**;

/

**execute** criptare\_pacient;

**select** \* **from** pacient\_criptat;

-- Facem procedura inversa, de decriptare

**drop** **table** pacient\_decriptat;

**create** **table** pacient\_decriptat (

id\_pacient varchar2(**100**),

prenume varchar2(**100**) **not** **null**,

cnp varchar2(**100**) **not** **null**,

sex varchar2(**100**) **not** **null**,

**constraint** pacient\_deriptat\_pk **primary** **key** (id\_pacient)

);

**create** **or** **replace** **procedure** decriptare\_pacient **as**

cheie\_derypt raw(**16**);

operation\_mode pls\_integer;

-- cursorul care parcurge liniile tabelului PACIENT

**cursor** **c** **is** **select** \* **from** pacient\_criptat;

raw\_id raw(**100**);

raw\_prenume raw(**100**);

raw\_cnp raw(**100**);

raw\_sex raw(**100**);

raw\_integritate raw(**500**);

decrypt\_id varchar2(**100**);

decrypt\_prenume varchar2(**100**);

decrypt\_cnp varchar2(**100**);

decrypt\_sex varchar2(**100**);

decrypt\_integritate raw(**500**);

concat\_val varchar2(**500**);

raw\_concat\_val raw(**500**);

**begin**

**select** cheie **into** cheie\_derypt **from** chei **where** id\_chei = **1**;

operation\_mode := dbms\_crypto.encrypt\_aes128 + dbms\_crypto.pad\_pkcs5 + dbms\_crypto.chain\_cbc;

**for** i **in** **c** loop

raw\_id := dbms\_crypto.decrypt(i.id\_pacient, operation\_mode, cheie\_derypt);

raw\_prenume := dbms\_crypto.decrypt(i.prenume, operation\_mode, cheie\_derypt);

raw\_cnp := dbms\_crypto.decrypt(i.cnp, operation\_mode, cheie\_derypt);

raw\_sex := dbms\_crypto.decrypt(i.sex, operation\_mode, cheie\_derypt);

decrypt\_id := utl\_i18n.raw\_to\_char(raw\_id, 'AL32UTF8');

decrypt\_prenume := utl\_i18n.raw\_to\_char(raw\_prenume, 'AL32UTF8');

decrypt\_cnp := utl\_i18n.raw\_to\_char(raw\_cnp, 'AL32UTF8');

decrypt\_sex := utl\_i18n.raw\_to\_char(raw\_sex, 'AL32UTF8');

concat\_val := decrypt\_id || decrypt\_prenume || decrypt\_cnp || decrypt\_sex;

raw\_concat\_val := utl\_i18n.string\_to\_raw(concat\_val);

raw\_integritate := dbms\_crypto.hash(raw\_concat\_val, dbms\_crypto.hash\_md5);

decrypt\_integritate := dbms\_crypto.decrypt(i.integritate, operation\_mode, cheie\_derypt);

if (raw\_integritate = decrypt\_integritate) **then**

**insert** **into** pacient\_decriptat **values** (decrypt\_id, decrypt\_prenume, decrypt\_cnp, decrypt\_sex);

**else**

dbms\_output.put\_line('Integrity failed at id: ' || decrypt\_id);

**end** if;

**end** loop;

**commit**;

**end**;

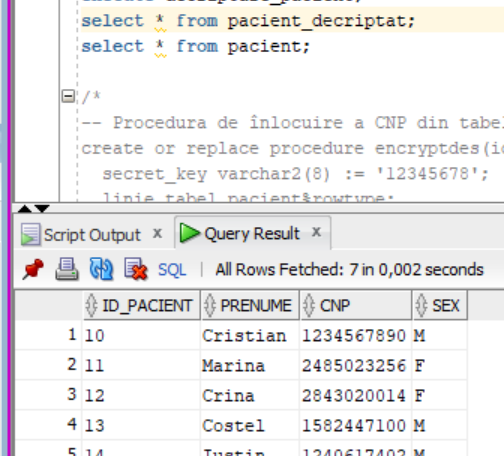
/

**execute** decriptare\_pacient;

**select** \* **from** pacient\_decriptat;

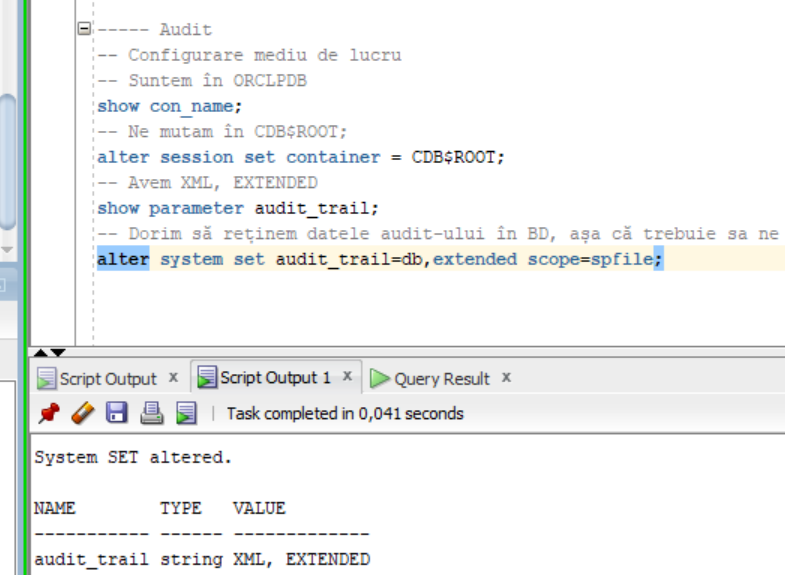
**select** \* **from** pacient;



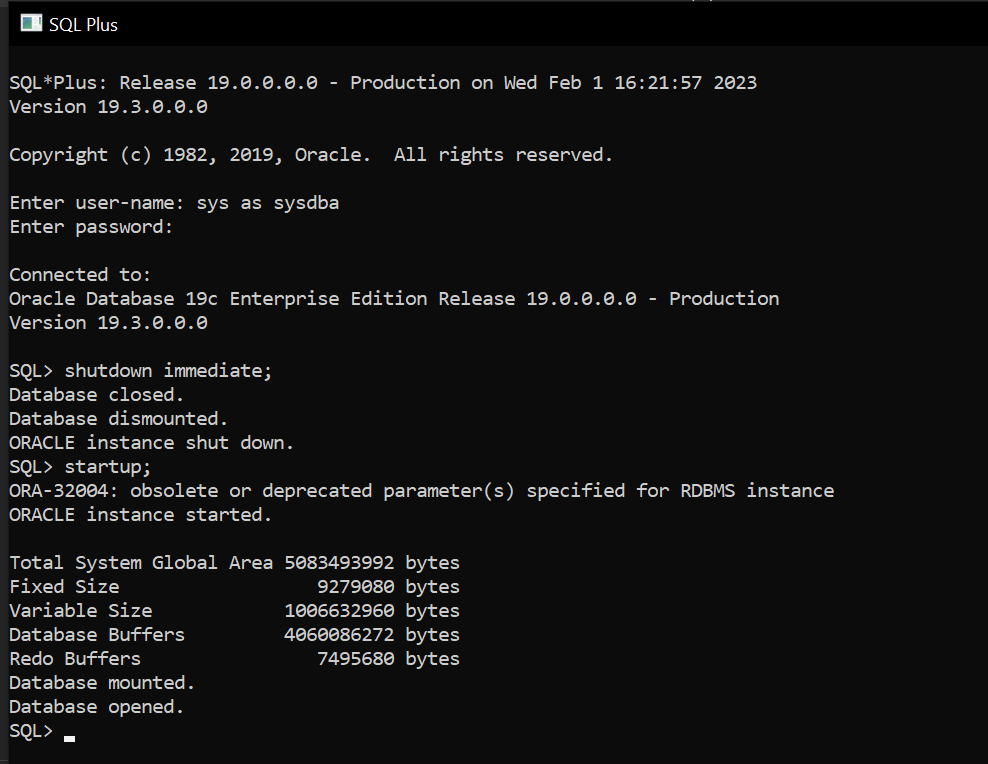


# **5. Audit**

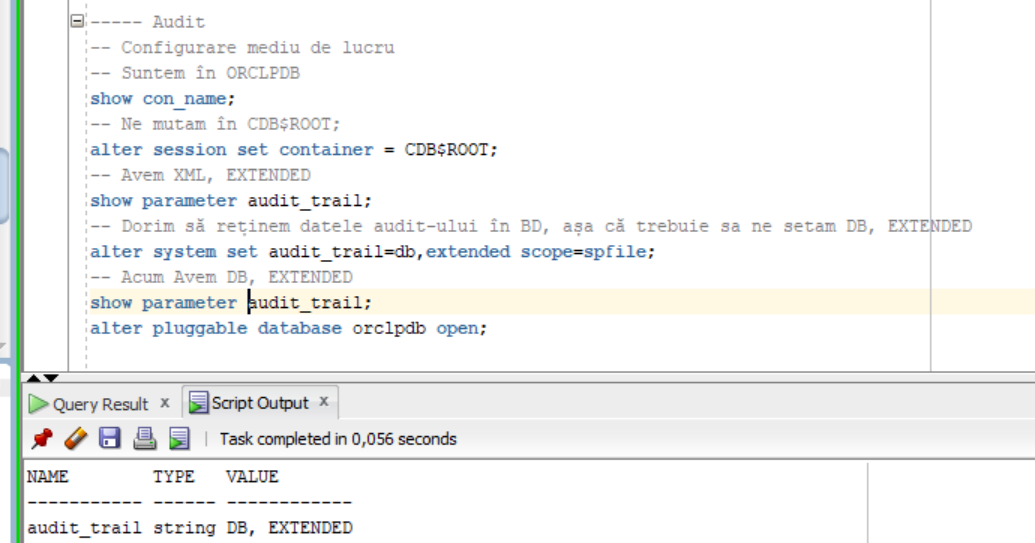
Înainte de a ne apuca propriu-zis de cerințe, trebuie să pregătim spațiul de lucru. Lucrăm în ***sys\_local***.



Pentru ca modificările noastre să se aplice, trebuie să restartăm instanța de Oracle. Intrăm în SQL Plus și introducem următoarele comenzi:



Ne reconectăm la baza de date și dacă testăm, acum vom avea DB\_EXTENDED.



## *5.1. Audit Standard*

Folosind acest tip de audit vom monitoriza istoricul pacientului: vor fi înregistrate operațiile de INSERT, UPDATE, DELETE pe tabelele **CONSULTAȚIE** și **REȚETĂ**.

În ***clinica\_admin*** rulăm:

-- Audit Standard

audit **insert**, **delete**, **update** **on** consultatie;

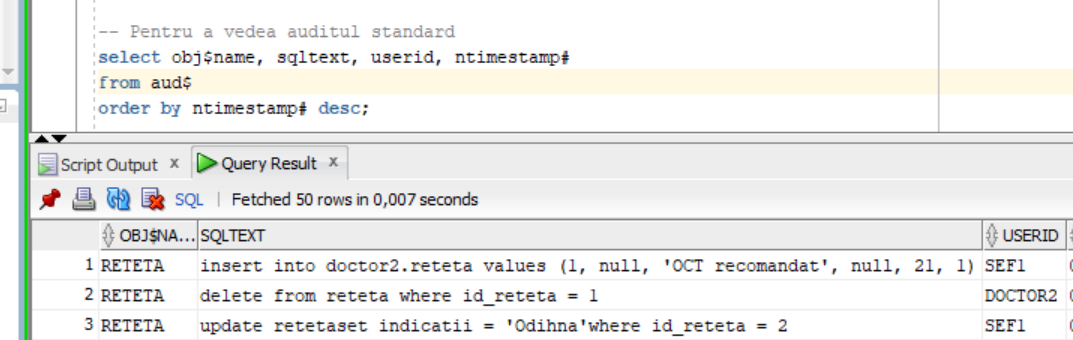
audit **insert**, **delete**, **update** **on** sef1.reteta;

audit **insert**, **delete**, **update** **on** doctor2.reteta;

audit **insert**, **delete**, **update** **on** doctor3.reteta;

audit **insert**, **delete**, **update** **on** doctor4.reteta;

Pentru a vedea operațiile auditate, în ***sys\_local*** rulăm:



Iar pentru a închide auditul standard, din ***clinica\_admin*** rulăm:

-- Pentru a oprit auditul standard

noaudit **insert**, **delete**, **update** **on** consultatie;

noaudit **insert**, **delete**, **update** **on** sef1.reteta;

noaudit **insert**, **delete**, **update** **on** doctor2.reteta;

noaudit **insert**, **delete**, **update** **on** doctor3.reteta;

noaudit **insert**, **delete**, **update** **on** doctor4.reteta;

## *5.2. Triggers de Auditare*

Vom monitoriza operațiile de INSERT și UPDATE pe tabelele **DOCTOR** și **SECȚIE**. În ***sys\_local*** avem următoarele comenzi:

--- Trigger Audit

-- Creare tabel audit

**create** sequence secventa\_audit **start** **with** **1** **increment** **by** **1**;

-- drop table audit\_istoric;

**create** **table** audit\_istoric (

id\_audit number,

user\_ varchar2(**30**),

session\_ number(**10**),

host\_ varchar2(**100**),

time\_ date,

nume varchar2(**20**),

**constraint** audit\_pk **primary** **key** (id\_audit)

);

-- Configurare Trigger (INSERT și UPDATE pe DOCTOR și SECȚIE)

**create** **or** **replace** **trigger** audit\_trigger\_doctor

**after** **insert** **or** **update** **on** clinica\_admin.doctor

**for** **each** **row**

**begin**

**insert** **into** audit\_istoric **values** (secventa\_audit.nextval, sys\_context('userenv', 'session\_user'),

sys\_context('userenv', 'sessionid'), sys\_context('userenv', 'host'),

sysdate, 'DOCTOR');

**end**;

/

**create** **or** **replace** **trigger** audit\_trigger\_sectie

**after** **insert** **or** **update** **on** clinica\_admin.sectie

**for** **each** **row**

**begin**

**insert** **into** audit\_istoric **values** (secventa\_audit.nextval, sys\_context('userenv', 'session\_user'),

sys\_context('userenv', 'sessionid'), sys\_context('userenv', 'host'),

sysdate, 'SECTIE');

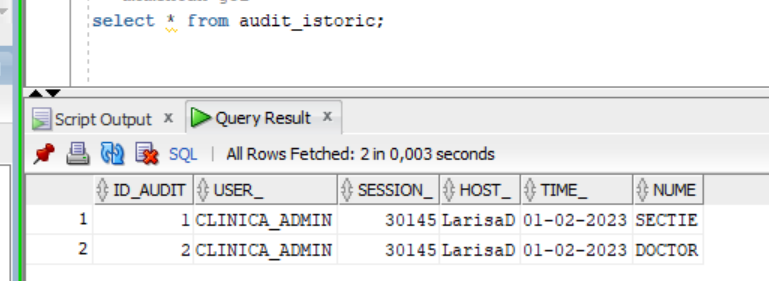
**end**;

/

-- momentan gol

**select** \* **from** audit\_istoric;

Iar în ***clinica\_admin*** putem să facem câteva operații care să activeze trigger-ul de auditare:



## *5.3. Politici de Auditare*

Vom lua în considerare operațiile de INSERT pe tabelul **PROCEDURA**. Din ***sys\_local***, creăm procedura de audit:

--- Politica Audit

-- Handler

**create** **or** **replace** **procedure** audit\_handler(object\_schema varchar2, object\_name varchar2, policy\_name varchar2) **as**

**begin**

dbms\_output.put\_line('Modififcare Procedura');

**end**;

/

-- Politica

**create** **or** **replace** **procedure** proc\_audit **as**

**begin**

dbms\_fga.add\_policy(

object\_schema => 'CLINICA\_ADMIN',

object\_name => 'PROCEDURA',

policy\_name => 'politica\_procedura',

enable => **false**,

statement\_types => 'INSERT',

handler\_module => 'AUDIT\_HANDLER'

);

**end**;

/

**execute** proc\_audit;

-- Trebuie activata mai întâi

**begin**

dbms\_fga.enable\_policy(

object\_schema => 'CLINICA\_ADMIN',

object\_name => 'PROCEDURA',

policy\_name => 'politica\_procedura'

);

**end**;

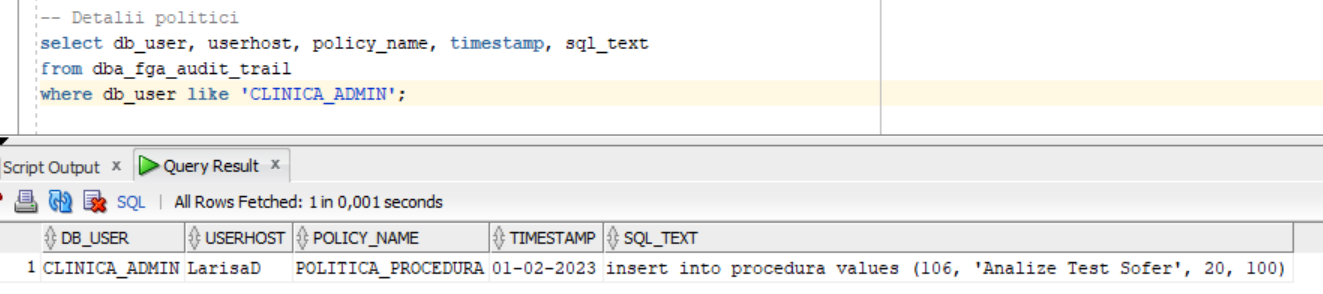
/

-- Detalii politici

**select** db\_user, userhost, policy\_name, **timestamp**, sql\_text

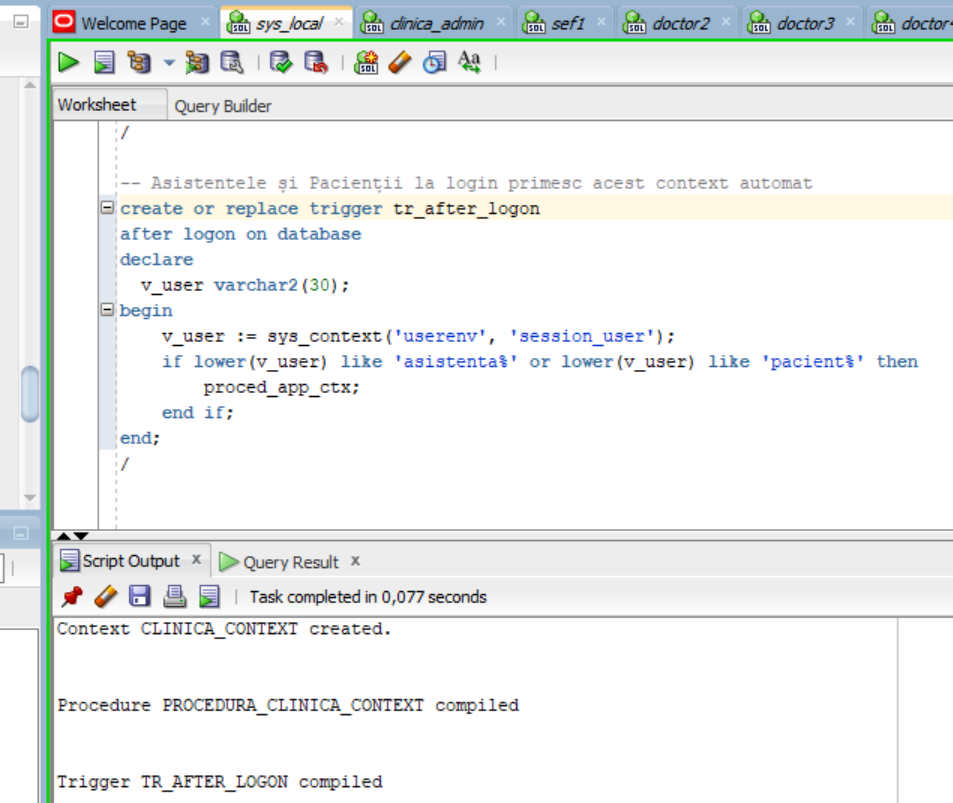
**from** dba\_fga\_audit\_trail

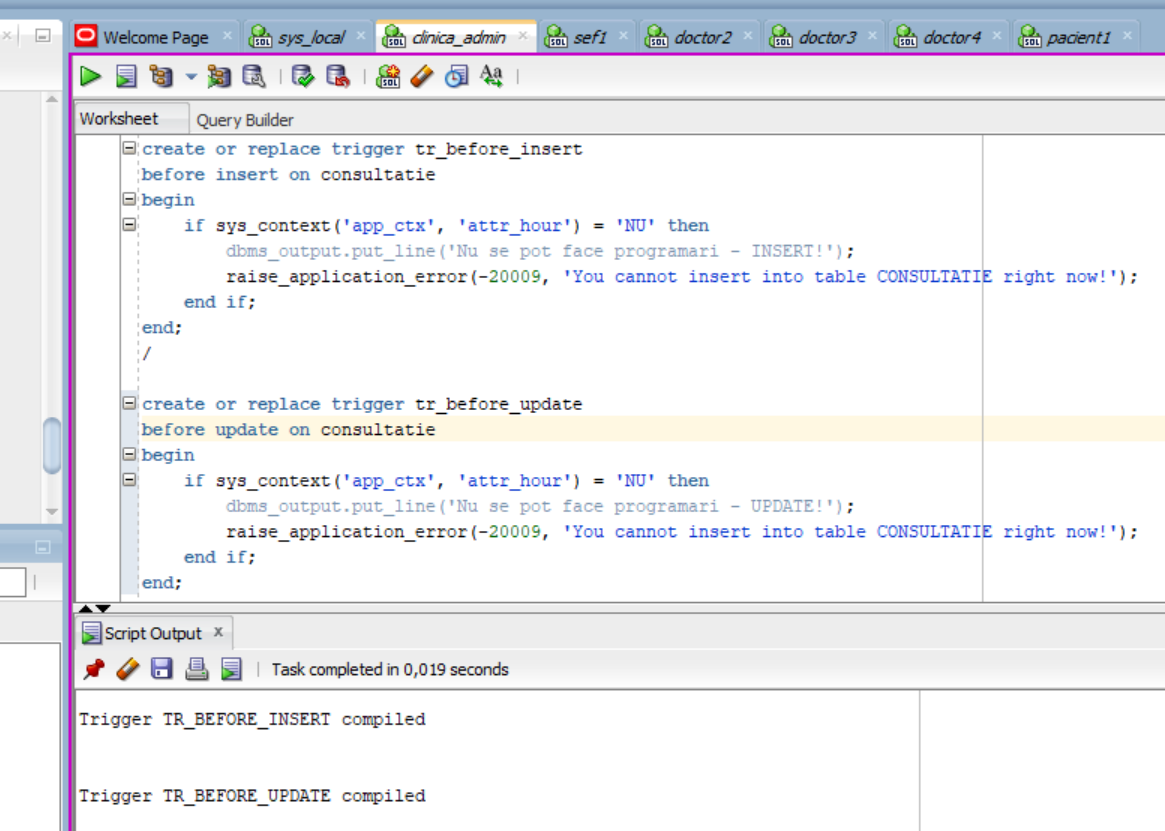
**where** db\_user **like** 'CLINICA\_ADMIN';

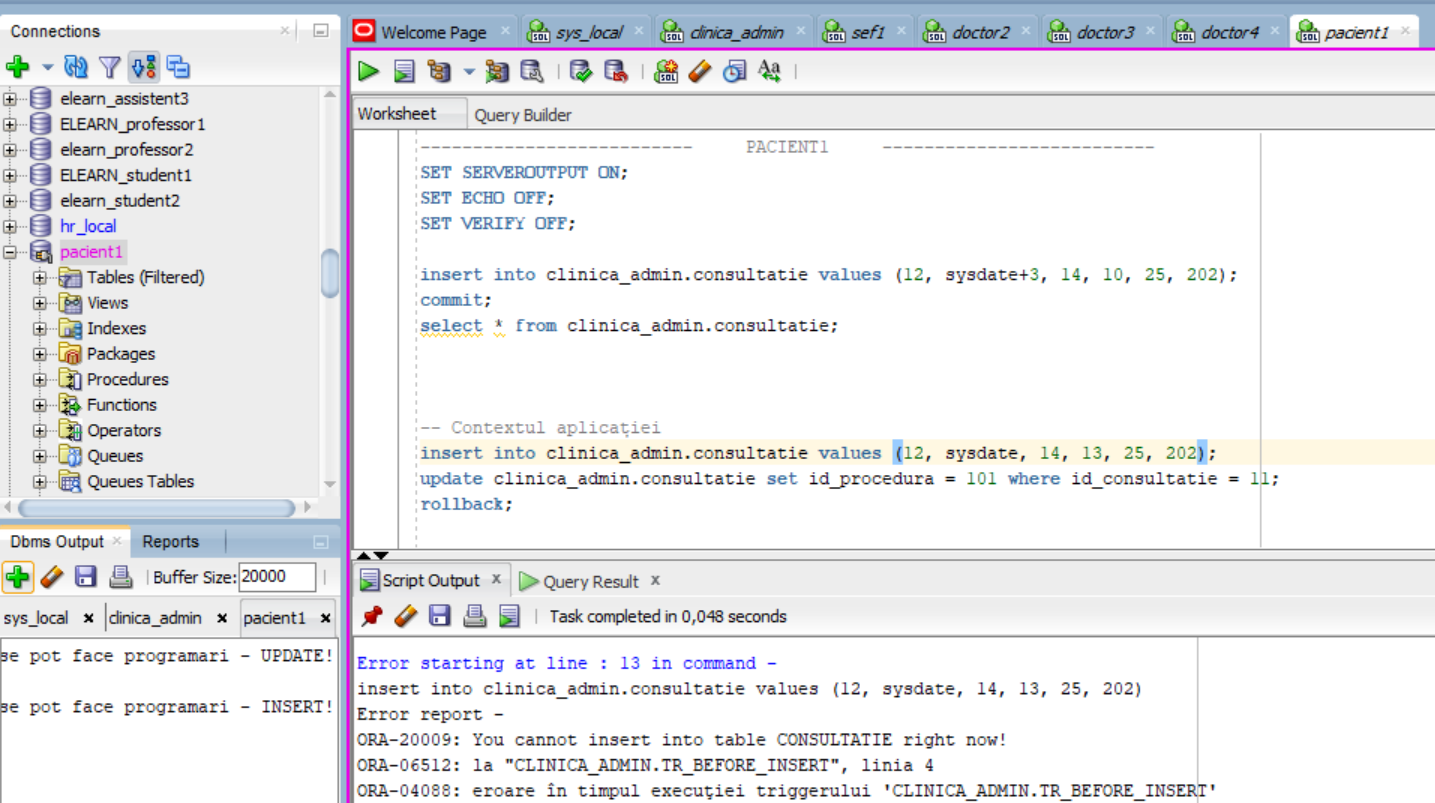


# **6. Aplicațiile pe baza de date și securitatea datelor**

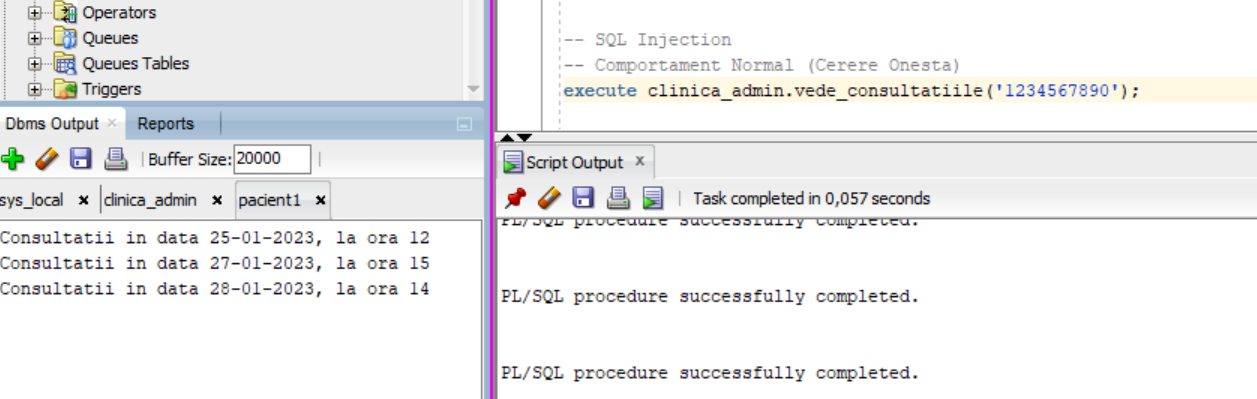
Pentru Contextul Aplicației, similar nu laboratorul, nu vom lăsa asistentele și pacienții să programeze consultații în afara orele de lucru (8.00 – 20.00).

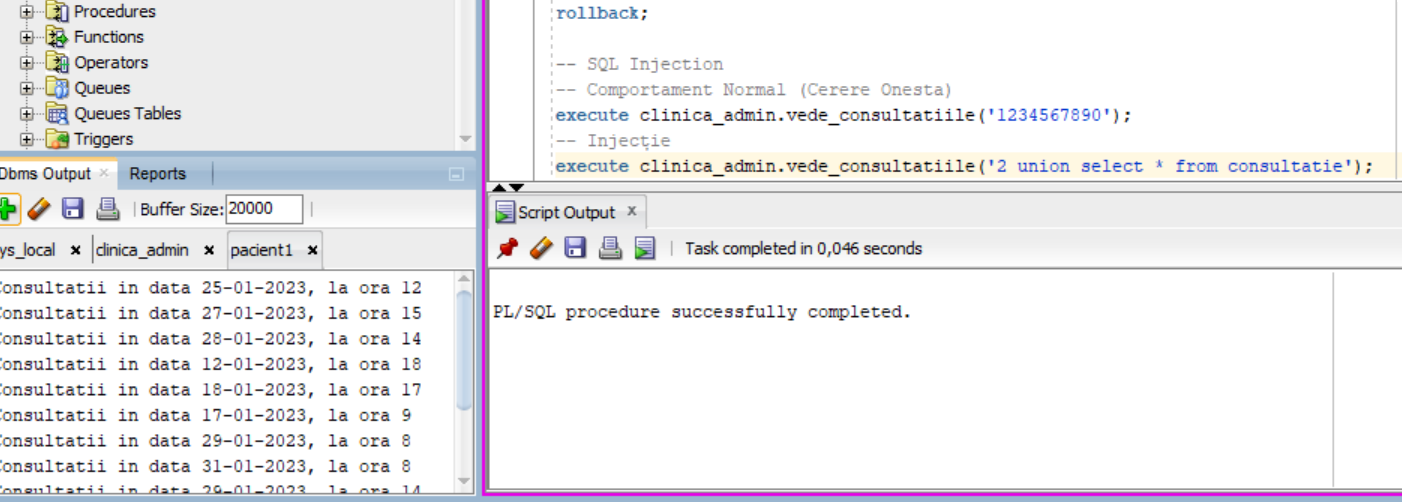






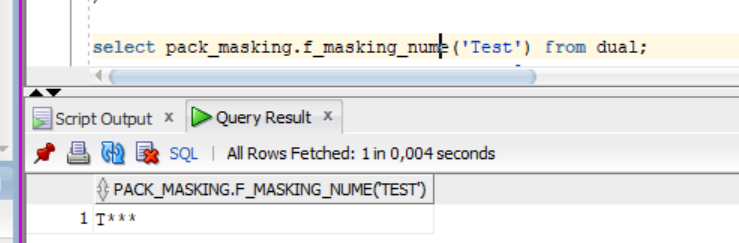
Pentru SQL injection vom crea o funcție astfel încât un pacient să poată vedea toate consultațiile, inclusiv cele la care nu ar avea acces.

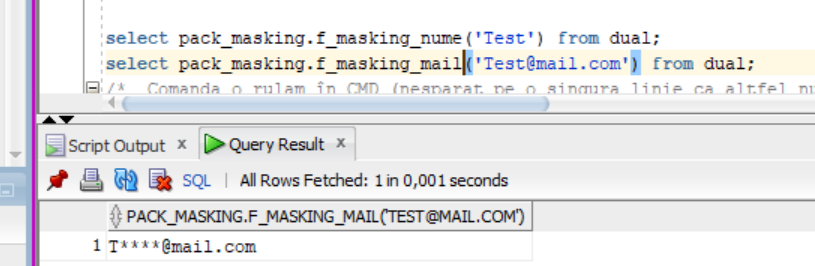




# **7. Mascarea Datelor**

Vom masca numele pacienților și mail-ul acestora.





Apoi vom exporta datele.

