

# **Proiect Securitatea Bazelor De Date**

**Mocică Răzvan-Cristian**

**Grupa 505**

## Cuprins

Proiect Securitatea Bazelor De Date .....	1
Mocică Răzvan-Cristian .....	1
Grupa 505 .....	1
1. Prezentarea modelului și a regulilor sale .....	1
1.1. Constrângeri pentru implementare .....	2
1.2. Diagrama Entitate Relație .....	3
1.3. Diagrama Conceptuală .....	4
1.4. Regulile de securitate .....	5
2. Procesele aplicației .....	6
2.1. Matricea Proces-Utilizator .....	7
2.2. Matricea Entitate-Proces .....	8
2.3. Matricea Entitate-Utilizator .....	9
3. Gestiunea Utilizatorilor și a Resurselor Computaționale .....	10
3.1. Configurarea Utilizatorilor și a Schemei .....	10
3.2. Memorie alocată pentru categoriile de utilizatori .....	11
3.3. Profile .....	11
3.3.1. Plan de consum .....	11
3.3.2. Crearea Efectivă .....	12
3.4. Permisii Admin .....	13
4. Creare Bazei .....	14
4.1. Crearea Schemei Admin .....	14
4.1.1. Criptare în Schema Admin .....	15
4.2. Crearea Schemei Antrenor .....	16
4.2.1. Criptare în Schema Antrenor .....	17
5. Obiect dependent .....	19
6. Audit .....	21
6.1. Audit Standard .....	21
6.2. Triggeri de Auditare .....	23
6.3. FGA .....	24

7. Contextul aplicației .....	26
7.1. VPD .....	27
8. SQL injection .....	29
8.1. Procedura Vulnerabilă .....	29
8.2. Procedura repartă .....	31
9. Mascarea datelor .....	32
9.1. Export .....	33
9.2. Import .....	34
10. Codul SQL al aplicației .....	37
10.1. Admin .....	37
10.1.2. bro_admin_audit.sql .....	42
10.1.3. bro_admin_create_tables.sql .....	45
10.1.4. bro_admin_criptare.sql .....	83
10.1.5. bro_admin_mask.sql .....	85
10.1.6. bro_admin_programs_view.sql .....	90
10.1.7. bro_admin_update_echipament_fals.sql .....	91
10.2. Antrenor .....	91
10.2.1. bro_antrenor_insert.sql .....	91
10.2.2. bro_antrenor1_cript_show.sql .....	96
10.3. Client .....	101
10.3.1. bro_client1_select_cript.sql .....	101
10.3.2. bro_client1_sql_injection.sql .....	102
10.4.1. bro_import.sql .....	103
10.5. Manager .....	103
10.5.1. bro_manager_filiala1_context.sql .....	103
10.7. SYS .....	104
10.7.1. sys_admin_antrenor_privilege.sql .....	104
10.7.2. sys_audit_1.sql .....	105
10.7.3. sys_audit_2.sql .....	109
10.7.4. sys_context.sql .....	111

10.7.5. sys_mask.sql .....	113
10.7.6. sys_users_1.sql .....	114
11. Codul CMD .....	131
11.1. import_mask_person.cmd .....	131
11.2. mask_person.cmd .....	131
11.3. seed_antrenor.cmd .....	132
12. Link repository .....	132

## 1. Prezentarea modelului și a regulilor sale

Proiectul implementează gestiunea mai multor filiale dintr-un lanț de săli de fitness. Abonamentul unui client este valabil în toate filialele, iar, de asemenea, clienții pot cumpăra suplimente nutritive de la recepția tuturor filialelor (nu se iau în calcul alte tranzacții pe care clientul le face la recepție (e.g. cumpără apă)) contorizăm doar comenzile efective de suplimente nutritive ale clientului). Modalitatea de plată a serviciilor și comenzilor nu este reținută în baza de date. Fiecare filială va avea angajați, aceștia putând fi antrenori sau recepționiști. Totodată, se rețin echipamentele pentru fiecare sediu, iar fiecare angajat lucrează doar într-un singur sediu. Echipamentele și suplimentele nutritive vor avea neapărat cel puțin un furnizor.

Clienții pot avea abonamente lunare, trimestriale, bianuale, anuale sau extinse. Despre clienți se vor înregistra numele, prenumele, vârsta, un email, dacă este student sau nu și posibil unul sau mai multe numere de telefon. De asemenea, un client va urma numai un unic program de antrenament de un anumit tip.

Fiecare filială a sălii are un anumit număr de angajați (antrenori sau recepționiști). Un antrenor poate avea unul sau mai multe programe de antrenament și este obligat să-și ateste studiile. Recepționiștii se vor ocupa de comenzi și vor avea fie program complet fie cu normă redusă. Pentru fiecare angajat se va ține minte numele, prenumele, vârsta, posibil unul sau mai multe numere de telefon, un email, data angajării și salariul (în lei). În fiecare filială va fi un unic manager.

Echipamentele sportive aparțin unei singure filiale, iar numele efectiv al acestora nu depinde de furnizor (e.g. o presă furnizată de X se va numi tot presă dacă este furnizată și de Y). Se vor ține minte data instalării echipamentelor, ultima revizie (data primei revizii va coincide cu data instalării) și numele.

Suplimentele se pot comanda doar de la recepție și numele lor nu depinde de furnizor (e.g. proteina furnizată de X se va numi tot proteină dacă este furnizată și de Y). Se vor ține minte numele, o descriere, kaloriile (pe 100g), prețul și furnizorul.

Pentru fiecare filială se păstrează angajații și echipamentele. De asemenea, se vor reține data de înființare, adresa și numele. Pentru furnizori se vor salva adresa, codul fiscal și numele.

## **1.1. Constrângeri pentru implementare**

Fiecare client poate avea un singur abonament

Un client nu poate fi și angajat.

Abonamentul unui client este unic pentru toate filialele.

Tipurile de abonamente sunt: lunar, trimestrial, bianual, anual, extins.

Suplimentele se țin pe toată firma, nu pe filiale.

Un client trebuie să urmeze numai un program de antrenament de un anumit tip.

Tipurile de programe sunt: Mass, Body, Recovery.

Emailul este unic pentru fiecare persoană.

Un angajat poate lucra doar la o singură filială.

Suplimentele nutritive/Echipamentele pot avea același id pentru furnizori diferiți.

Nu se ține minte metoda de plată pentru abonamente și comenzi

Se rețin doar comenzile efective de suplimente de la recepție ( de exemplu nu se rețin plățile pentru apa ).

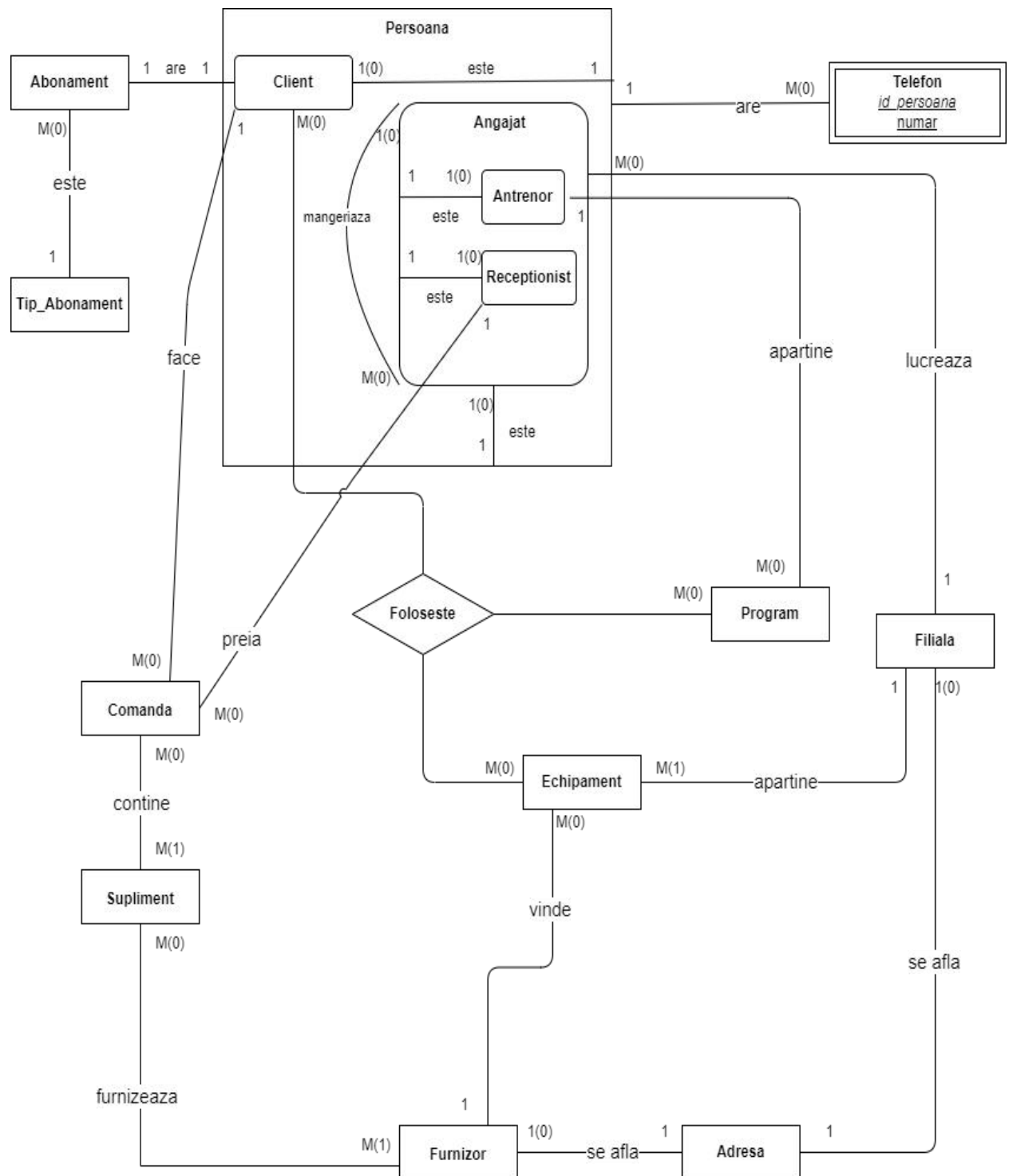
O aprovizionare a firmei se va face cu un singur supliment o dată.

Nu se înregistrează stocuri de suplimente.

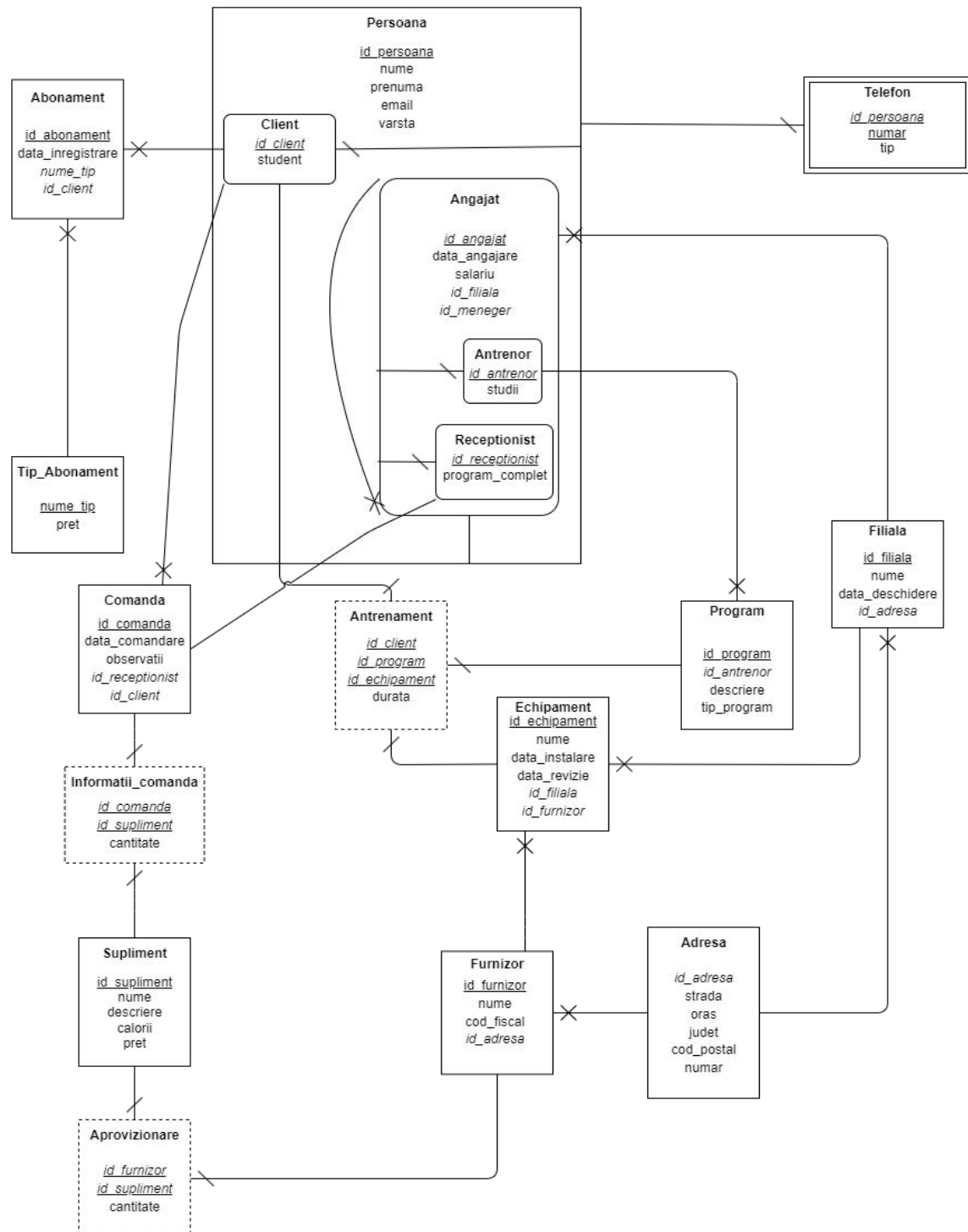
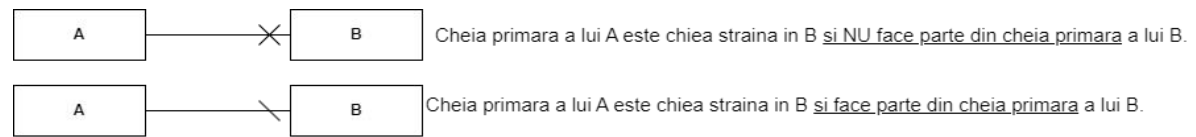
Filialele trebuie să aibă adresele diferite între ele.

Furnizorii trebuie să aibă adresele diferite între ei.

## 1.2. Diagrama Entitate Relație



### 1.3. Diagrama Conceptuală





Persoana(id\_persoana, nume, prenume, email, varsta)

Client(id\_client, student) {id\_client este și cheie străină către tabelul Persoana}

Angajat(id\_angajat, data\_angajare, salariu, id\_filiala, id\_meneger) {id\_angajat este și cheie străină către tabelul Persoana, id\_meneger este cheie străină tot către tabelul Angajat}

Antrenor(id\_antrenor, studii) {id\_antrenor este și cheie străină către tabelul Angajat}

Receptionist(id\_receptionist, program\_complet) {id\_receptionist este și cheie străină către tabelul Angajat}

Telefon(id\_persoana, numar, tip)

Abonament(id\_abonament, data\_inregistrare, nume\_tip, id\_client)

Tip\_Abonament(nume\_tip, pret)

Filiala(id\_filiala, nume, data\_deschidere, id\_adresa)

Adresa(id\_adresa, strada, oras, judet, cod\_postal, numar)

Furnizor(id\_furnizor, nume, cod\_fiscal, id\_adresa)

Program(id\_program, id\_antrenor, descriere, tip\_program)

Echipament(id\_echipament, nume, data\_instalare, data\_revizie, id\_filiala, id\_furnizor)

Antrenament(id\_client, id\_program, id\_echipament, durata)

Comanda(id\_comanda, data\_comandare, observatii, id\_receptionist, id\_client)

Informatii\_Comanda(id\_comanda, id\_supliment, cantitate)

Supliment(id\_supliment, nume, descriere, calorii, pret)

Aprovizionare(id\_furnizor, id\_supliment, cantitate)

## 1.4. Regulile de securitate

În cadrul proiectului se vor cripta datele antrenamentelor fiecărui client, astfel încât doar el și antrenorul său să poată vedea informațiile. De asemenea, se vor realiza două VPD-uri (Virtual Private Database) pentru a asigura că managerii de filială pot face

operații DML doar pe echipamentele din filiala lor și că pot vedea doar istoricul echipamentelor care sunt sau au fost în acea filială. De asemenea, se vor respecta permisiunile din matricea entitate-utilizator și se vor da cote de memorie conform necesităților de stocare.

## **2. Proceele aplicației**

Proceele care pot fi inițiate în cadrul bazei sunt:

1. Configurarea angajați
2. Vizualizarea antrenorilor dintr-o filială
3. Vizualizarea recepționiștilor dintr-o filială
4. Vizualizarea programelor de antrenament pentru un anumit antrenor
5. Delegarea/Revocarea de manager pentru o filială
6. Vizualizarea caracteristicilor unei filiale
7. Vizualizarea echipamentelor dintr-o filială
8. Managementul clienților
9. Configurarea abonamentului pentru un client
10. Configurarea programelor pentru un antrenor
11. Configurarea antrenamentelor pentru un antrenor
12. Verificarea validității abonamentului pentru un client
13. Vizualizarea clienților
14. Vizualizarea antrenamentelor unui client
15. Vizualizarea antrenamentelor unui antrenor
16. Crearea unei comenzi
17. Vizualizarea suplimentelor puse la vânzare
18. Vizualizarea tuturor comenzilor
19. Vizualizarea comenzilor pentru un client

20. Vizualizarea aprovizionărilor suplimente
21. Gestionarea aprovizionărilor suplimente
22. Gestionarea echipamentelor dintr-o filială
23. Vizualizarea tuturor antrenamentelor dintr-o filială
24. Matricea Proces-Utilizator
25. În cadrul proiectului se disting șase categorii distincte de utilizatori:
26. Admin – cel care gestionează aplicația
27. Antrenor – angajatul de tip antrenor din cadrul unei săli
28. Receptionist – angajatul de tip recepționist din cadrul unei săli
29. Manager Filială – managerul unei singure filiale
30. Client – client pe întregul lanț de săli
31. Public General

## 2.1. Matricea Proces-Utilizator

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23
<b>Admin</b>	X	X	X	X	X	X	X						X				X	X	X	X	X		
<b>Antrenor</b>		X		X		X	X			X	X			X	X		X						
<b>Receptionist</b>		X		X		X	X	X	X			X	X			X	X			X			
<b>Manager Filiala</b>		X	X	X		X	X						X	X	X		X			X		X	X
<b>Client</b>		X		X		X	X							X			X						
<b>Public General</b>		X		X		X	X										X						

## 2.2. Matricea Entitate-Proces

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23
Persoana	I,U	S	S	S				I,U	S	S	S	S	S	S	S	S		S	S				S
Client								I,U	S		S	S	S	S	S	S		S	S				S
Angajat	I,U	S	S	S	I,U					S	S			S	S	S		S	S				S
Antrenor	I,U	S		S						S	S			S	S								S
Receptionist	I,U		S													S		S	S				
Telefon	I,U	S	S					I,U															
Abonament									I,U,D			S											
Tip_Abonament									S			S											
Filiala		S	S			S	S															S	S
Adresa						S	S													S			
Furnizor																				S	S	S	
Program				S						I,U,D	S			S	S								S
Echipament							S				S			S	S							I,U,D	S
Antrenament											I,U			S	S								S
Comanda																I		S	S				
Informatii_Comanda																I		S	S				
Supliment																S	S	S	S	S	I,U		
Aprovizionare																				S	I,U		

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

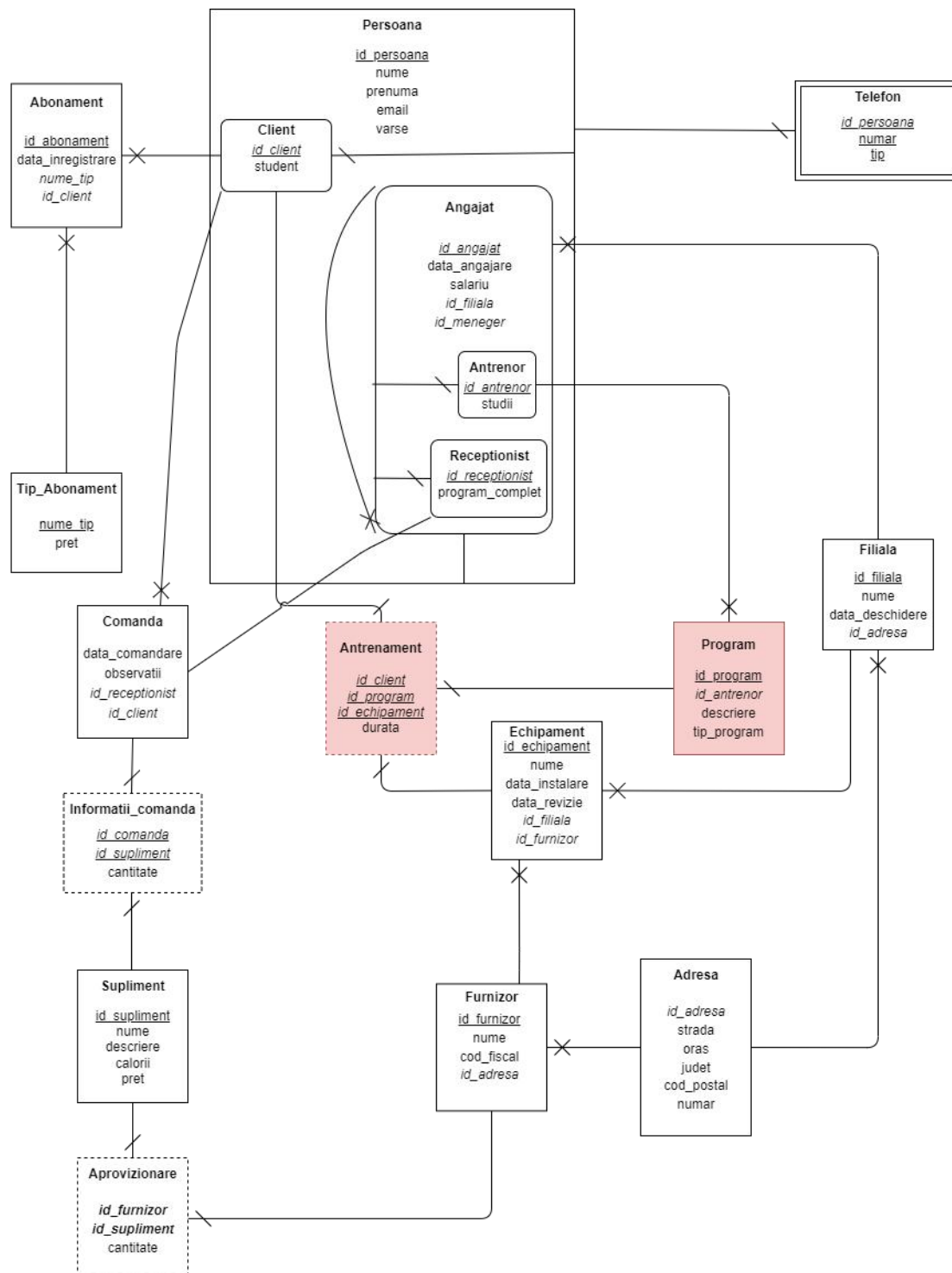
## 2.3. Matricea Entitate-Utilizator

	Admin	Antrenor	Receptionist	Manager Filiala	Client	Public General
Persoana	I,U,S	S	S,I,U	S	S	S
Client		S	I,U,S	S	S	
Angajat	I,U,S	S	S	S	S	S
Antrenor	I,U,S	S	S	S	S	S
Receptionist	I,U,S			S		
Telefon	I,U,S	S	S,I,U	S	S	S
Abonament			I,U,D,S			
Tip_Abonament			S			
Filiala	S	S	S	S	S	S
Adresa	S	S	S	S	S	S
Furnizor	S		S	S		
Program	S	S,I,U,D	S	S	S	S
Echipament	S	S	S	S,I,U,D	S	S
Antrenament		I,U,S		S	S	
Comanda	S		I			
Informatii_Comanda	S		I			
Supliment	S,I,U	S	S	S	S	S
Aprovizionare	S,I,U		S	S		

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

### 3. Gestiunea Utilizatorilor și a Resurselor Computaționale

#### 3.1. Configurarea Utilizatorilor și a Schemei



Din cadrul diagramei conceptuale, toate tabelele vor fi create în schema adminului, mai puțin **Program** și **Antrenament** care vor fi create separat de fiecare antrenor în schema lui proprie.

### 3.2. Memorie alocată pentru categoriile de utilizatori

- Deoarece majoritatea bazei de date va fi creată în schema adminului, acesta va avea la dispoziție 500MB.
- Fiecare antrenor va avea la dispoziție 10MB pentru a crea obiecte.
- Întrucât restul utilizatorilor nu vor crea obiecte, aceștia nu vor avea memorie alocată.

### 3.3. Profile

În cadrul aplicației, parolele trebuie să conțină minim un „\_”. Această condiție va fi verificată în cadrul fiecărui profil utilizând opțiunea *password\_verify\_function*.

Pentru admin se permit mai multe sesiuni, sunt permise 15 minute de idle per sesiune, parola trebuie schimbată o dată la 90 de zile, sunt permise 5 greșeli consecutive ale credentialelor și are dreptul la 6 minute maxime de CPU time per comandă.

Pentru publicul general se permit 6 sesiuni, sunt permise 5 minute de idle per sesiune, un maxim de 20 de minute per sesiune și dreptul la 1 minut maxim de CPU time per comandă.

Pentru restul tipurilor de utilizatori se permite doar o sesiune, sunt permise 15 minute de idle, parola trebuie schimbată o dată la 90 de zile, sunt permise 5 greșeli consecutive ale credentialelor și are dreptul la 2 minute maxime de CPU time per comandă.

#### 3.3.1. Plan de consum

Pentru planul de consum, când CPU-ul ajunge la 100%, în cadrul aplicației există următoarea regulă:

Admin: 30%

Public General: 5%

Clienți: 10%

Manageri: 15%

Recepționiști: 15%

Antrenori: 20%

Other Groups: 5%

### 3.3.2. Crearea Efectivă

Crearea utilizatorilor, mai puțin a adminului, a fost realizată utilizând un pachet custom utilitar. Codul pentru acest pachet și configurarea inițială a utilizatorilor cu profile și resource group este prezentă în fișierul *sys\_users\_1.sql*. Mai jos sunt câteva outputuri din rularea acestui cod:

```
Function PASSWORD_VERIFY_FUNCTION_STANDALONE compiled
```

```
Package BRO_USER_UTILS compiled
```

```
Package Body BRO_USER_UTILS compiled
```

```
SELECT DISTINCT u.username, u.profile, p.group_or_subplan, p.mgmt_p1, p.plan
FROM dba_users u JOIN dba_rsrc_plan_directives p
ON u.initial_rsrc_consumer_group=p.group_or_subplan
WHERE username LIKE upper('bro_%');
```

USERNAME	PROFILE	GROUP_OR_SUBPLAN	MGMT_P1	PLAN
BRO ANTRENOR6	BRO PROFILE ANTRENOR	BRO RG ANTRENOR	20 P	BRO
BRO ANTRENOR1	BRO PROFILE ANTRENOR	BRO RG ANTRENOR	20 P	BRO
BRO RECEPTIONIST10	BRO PROFILE RECEPTIONIST	BRO RG RECEPTIONIST	15 P	BRO
BRO CLIENT3	BRO PROFILE CLIENT	BRO RG CLIENT	10 P	BRO
BRO CLIENT8	BRO PROFILE CLIENT	BRO RG CLIENT	10 P	BRO
BRO CLIENT10	BRO PROFILE CLIENT	BRO RG CLIENT	10 P	BRO
BRO CLIENT4	BRO PROFILE CLIENT	BRO RG CLIENT	10 P	BRO
BRO RECEPTIONIST1	BRO PROFILE RECEPTIONIST	BRO RG RECEPTIONIST	15 P	BRO
BRO RECEPTIONIST4	BRO PROFILE RECEPTIONIST	BRO RG RECEPTIONIST	15 P	BRO
BRO RECEPTIONIST3	BRO PROFILE RECEPTIONIST	BRO RG RECEPTIONIST	15 P	BRO
BRO CLIENT1	BRO PROFILE CLIENT	BRO RG CLIENT	10 P	BRO
BRO CLIENT9	BRO PROFILE CLIENT	BRO RG CLIENT	10 P	BRO



### 3.4. Permisii Admin

Tot în acest fișier inițial de configurare sunt prezente și drepturile adminului.

```
GRANT CREATE SESSION
    TO bro_admin;
GRANT CREATE ANY TABLE
    TO bro_admin;
GRANT CREATE ANY VIEW
    TO bro_admin;
GRANT CREATE ANY TRIGGER
    TO bro_admin;
GRANT CREATE ANY PROCEDURE
    TO bro_admin;
GRANT CREATE ANY SEQUENCE
    TO bro_admin;
GRANT CREATE ANY INDEX
    TO bro_admin;
GRANT CREATE ANY TYPE
    TO bro_admin;
GRANT CREATE TYPE
    TO bro_admin;
--Pt proceduri
GRANT EXECUTE
    ON dbms_crypto
    TO bro_admin
    WITH GRANT OPTION;
--Pt generated by default on null as identity la create in
antrenor
GRANT SELECT ANY SEQUENCE
    TO bro_admin;
GRANT EXECUTE
    ON get_users_by_suffix
    TO bro_admin;
```

Deoarece schemele antrenorilor se vor crea utilizând scriptul din fișierul *bro\_admin\_antrenor\_seed.sql*, adminul are nevoie de permisiuni speciale pentru a putea rula scriptul de creare cu succes (create/select any).

Permisia Select Any Sequence este necesară deoarece tabelele din schema antrenorilor vor avea cheile primare generate folosind autoincrementul din Oracle. Astfel, adminul are nevoie de permisia de selectare a acestei secvențe generate la crearea tabelului:

```
id_program number(*, 0) generated BY DEFAULT ON NULL AS  
identity CONSTRAINT pk_program PRIMARY KEY
```

## 4. Creare Bazei

### 4.1. Crearea Schemei Admin

În cadrul adminului, adiacent tabelelor din schema conceptuală, se vor implementa la început încă două tabele, anume unul este de logging al erorilor care apar în operațiile DML pe tabelele sau view-urile de persoane ale schemei sale, iar celălalt este unul de mapare a unui utilizator creat cu un cont al bazei de date, astfel asigurând că există un cont înainte de a insera într-un tabel al bazei. Pentru popularea celui de-al doilea tabel este nevoie de legătura dintre admin și useri și, de aceea, există permisia de a executa funcția *get\_users\_by\_suffix* din sys către admin. Scriptul inițial al bazei admin, alături de inserarea unor date, este prezent în fișierul *bro\_admin\_create\_tables.sql*.

Structura tabelului de mapări de conturi este următoarea:

```
CREATE TABLE account_mapping(  
    id_persoana    NUMBER(*,0)    CONSTRAINT    pk_account_mapping  
    PRIMARY KEY,  
    username VARCHAR2(128) UNIQUE  
);
```

Acesta are cheia primară cea din tabela persoană, la momentul respectiv, care are contul și va ține minte și username-ul asociat acesteia. Inserările persoanelor în baza de date se vor face folosind view-urile specifice fiecărui tip, iar în trigger-ii “instead

of' se va apela funcția *insert\_into\_account\_mapping*, care asigură că există conturi ale bazei de date pentru tipul de persoană dorit a fi inserat. De exemplu, pentru clienți, dacă am utilizat toate conturile și dorim să mai adăugăm un client, vom primi eroarea:

```

1944 INSERT INTO client_extins (
1945     nume,
1946     prenume,
1947     email,
1948     varsta,
1949     student
1950 ) VALUES ( 'Nume',
1951             'Prenume',
1952             'ceva@yahoo.com',
1953             19,
1954             'Y' );
1955

```

Script Output x Query Result x

Task completed in 0.036 seconds

```

'Prenume',
'ceva@yahoo.com',
19,
'Y' )

```

Error at Command Line : 1,944 Column : 13

Error report -

SQL Error: ORA-20010: ORA-20010: ORA-20020: ORA-20020: No available account for the suffix CLIENT code: -20020

ORA-06512: at "BRO\_ADMIN.LOGGER\_UTILS", line 13

ORA-06512: at "BRO\_ADMIN.CLIENT\_EXTINS\_INSERT", line 23

ORA-04088: error during execution of trigger 'BRO\_ADMIN.CLIENT\_EXTINS\_INSERT'

#### 4.1.1. Criptare în Schema Admin

Întrucât scriptul de seed pentru antrenori include și partea de criptare corespunzătoare lor, înainte de a rula acel script voi prezenta criptarea.

Așadar, în cadrul aplicației, criptarea se va face pentru clienți la nivelul antrenamentelor unui antrenor, astfel încât un client să-și poată vedea doar propriile antrenamente. Pentru aceasta, în schema admin vom crea mai multe obiecte, scriptul asociat acestora se găsește în fișierul *bro\_admin\_criptare.sql*.

În acest script este prezent un tabel numit *chei\_client* care, pentru fiecare client, păstrează modul de criptare și cheia asociată. La inserarea în tabelul de chei se va folosi algoritmul AES128, iar paddingul va fi ales random pentru fiecare client, dintre PKCS5 și PADZERO, iar, tot random pentru fiecare client, se va alege și chainingul. După ce adminul a inserat pentru fiecare *id\_client* în *chei\_client*, vom avea:

	ID_CLIENT	MOD_OP	CHEIE
1	18	12550	8F2733D7DBCCA894A294E57DB814CA53
2	19	5126	0255F764CDC58509616077872B91A144
3	20	13062	412A221E306D56709DBFF155049B5E60
4	21	13318	34FBA0C0D1BBC7A176B669EA42798B5F
5	22	5126	CE10B23C313B59E0338313BF61AD9CC1
6	25	12550	6F39A75EBBEF0E0F5F8A6FBE33A88322
7	26	5126	980B986CFBD902C7341D684CFA3474C1
8	27	13318	4573644FE187B7D7B6525900C59E379C

Pentru ca un client să-și ia cheia unică, tot în cadrul scriptului este creată o funcție *get\_client\_key*, care, pentru username-ul luat din contextul *userenv*, va întoarce cheia de criptare asociată.

## 4.2. Crearea Schemei Antrenor

Acum, pentru a ilustra criptarea, vom rula scriptul din fișierul *bro\_antrenor\_seed.sql*. Pentru a ușura inserarea, în cadrul acestui fișier schema este dată ca un argument, de exemplu: **create table &&user\_name..program**

Tot în cadrul seed-ului se vor da antrenorilor permisiunile necesare creării obiectelor. De asemenea, pentru a rula scriptul se va folosi fișierul *seed\_antrenor.cmd*. La rularea acestui fișier se va introduce numele antrenorului:

```
SQL*Plus: Release 19.0.0.0.0 - Production on Mon Jan 13 14:47:30 2025
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle. All rights reserved.

Last Successful login time: Mon Jan 13 2025 14:23:25 +02:00

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

Enter value for user_name: bro_antrenor1
```

```
Type created.

old 1: create or replace function &user_name..fetch_decrypted_client_data (
new 1: create or replace function bro_antrenor3.fetch_decrypted_client_data (
old 5: ) return &user_name..decrypted_client_table
new 5: ) return bro_antrenor3.decrypted_client_table
old 61:         from &user_name..client_antrenament ca
new 61:         from bro_antrenor3.client_antrenament ca

Function created.

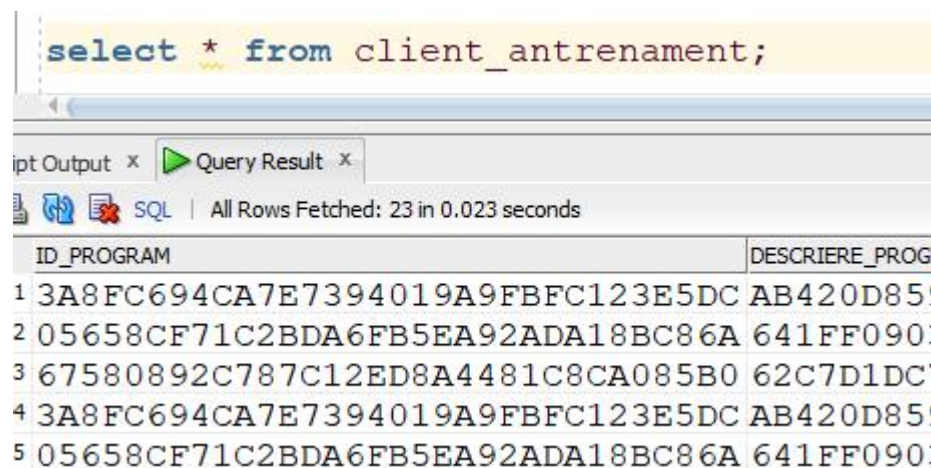
Commit complete.
```

### 4.2.1. Criptare în Schema Antrenor

Întrucât antrenorii trebuie să aibă acces la cheile de criptare ale clienților, li se va da acces la tabelul de chei, iar, totodată, scriptul creează și un view care va arăta datele antrenamentelor pentru clienți într-un mod criptat cu cheile individuale. Pentru a regăsi datele și a le valida, sunt create mai multe funcții și obiecte utilitare pe care un client le poate folosi.

Pentru a putea insera date vom rula din antrenor1 scriptul *bro\_antrenor\_insert.sql*.

Pentru a vedea datele criptate vom apela:



The screenshot shows a SQL query window with the query `select * from client_antrenament;` and its results. The results are displayed in a table with two columns: `ID_PROGRAM` and `DESCRIERE_PROG`. The data is encrypted using a key-based function. The first row shows the ID `3A8FC694CA7E7394019A9FBFC123E5DC` and the description `AB420D859`. The second row shows the ID `05658CF71C2BDA6FB5EA92ADA18BC86A` and the description `641FF0903`. The third row shows the ID `67580892C787C12ED8A4481C8CA085B0` and the description `62C7D1DC7`. The fourth row shows the ID `3A8FC694CA7E7394019A9FBFC123E5DC` and the description `AB420D859`. The fifth row shows the ID `05658CF71C2BDA6FB5EA92ADA18BC86A` and the description `641FF0903`.

ID_PROGRAM	DESCRIERE_PROG
1 3A8FC694CA7E7394019A9FBFC123E5DC	AB420D859
2 05658CF71C2BDA6FB5EA92ADA18BC86A	641FF0903
3 67580892C787C12ED8A4481C8CA085B0	62C7D1DC7
4 3A8FC694CA7E7394019A9FBFC123E5DC	AB420D859
5 05658CF71C2BDA6FB5EA92ADA18BC86A	641FF0903

De exemplu, dacă dorim să decriptăm din tabel totul, dar doar pentru clientul cu id-ul 18, outputul va arăta:

```

49 with c as (select mod_op,cheie from bro_admin.chei_client where id_client=18)
50 select decrypt_string(ca.id_program,c.mod_op,c.cheie) as id_program,
51 decrypt_string(ca.descriere_program,c.mod_op,c.cheie) as descriere_program,
52 decrypt_string(ca.tip_program,c.mod_op,c.cheie) as tip_program,
53 decrypt_string(ca.durata_antrenament,c.mod_op,c.cheie) as durata_antrenament,
54 decrypt_string(ca.id_echipament,c.mod_op,c.cheie) as id_echipament,
55 decrypt_string(ca.nume_echipament,c.mod_op,c.cheie) as nume_echipament,
56 decrypt_string(ca.data_instalare_echipament,c.mod_op,c.cheie) as data_instalare_echipament,
57 decrypt_string(ca.data_revizii_echipament,c.mod_op,c.cheie) as data_revizii_echipament,
58 decrypt_string(ca.id_filiala,c.mod_op,c.cheie) as id_filiala,
59 decrypt_string(ca.id_client,c.mod_op,c.cheie) as id_client,
60 checksum
61 from client_antrenament ca,c ;

```

ID_PROGRAM	DESCRIERE_PROGRAM	TIP_PROGRAM	DURATA_ANTRENAMENT	ID_ECHIPAMENT	NUME_ECHIPAMENT	DATA_INSTALARE_ECHIPAMENT	DATA_REVIZIE_ECHIPAMENT	ID_FILIALA	ID_CLIENT	CHECKSUM
1	Push, Pull Legs Light, for beginners	MASS	20	1	Leg Press	20-MAY-20	20-MAY-21	1	18	0BE2EBB12625080897E
2	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	7DAFE248E25885F5FC1
3	Full Body Variant Light	CARDIO	11	1	Leg Press	20-MAY-20	20-MAY-21	1	18	C4F6C9DAB2166A5D74C
4	Push, Pull Legs Light, for beginners	MASS	11	2	Chest Press	20-JUN-21	20-JUN-21	1	18	83544EC55033DEA93D5
5	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	DC1458748F034DB1E2A
6	Full Body Variant Light	CARDIO	10	2	Chest Press	20-JUN-21	20-JUN-21	1	18	FB222D264164BA5765E
7	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	A8E3C4F13A3CF1607D3
8	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	6693B65A2837C513993
9	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	88612721435FE4B4E05
10	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	8CDBB2874D69EB6EC5F
11	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	C4A969F2BA2656B331E
12	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	6F6E92D05543A7A8802
13	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	9516FD1C2E3A77DF60A
14	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	8AF0A3B767C7387085E
15	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	52D04916794B6C063FF
16	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	8656D411665F0C8284E
17	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	138005AFC87D533A3E
18	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	6B8B8E29E1F599F72BE
19	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	9C2FF44A3E3A8E3B0C5
20	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	FD1B9602D83F8570CAE
21	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	AIAC08D9439A9904414
22	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	6659C9B285E1C2061F5
23	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	8543AD595745B31577E

“Not allowed” este pus manual în caz de eroare de decriptare. Însă, dacă Oracle nu emite o eroare, el decide că a decriptat, deși, după cum se poate observa, doar pentru clientul cu id-ul 18 este descifrabil outputul. De asemenea, în scriptul de seed este prezentă și o funcție care returnează automat după id numai liniile decriptate corect:

```

=SELECT *
FROM TABLE(fetch_decrypted_client_data(
  (SELECT mod_op FROM bro_admin.chei_client WHERE id_client = 18),
  (SELECT cheie FROM bro_admin.chei_client WHERE id_client = 18),
  18
));

```

ID_PROGRAM	DESCRIERE_PROGRAM	TIP_PROGRAM	DURATA_ANTRENAMENT	ID_ECHIPAMENT	NUME_ECHIPAMENT	DATA_INSTALARE_ECHIPAMENT	DATA_REVIZIE_ECHIPAMENT	ID_FILIALA	ID_CLIENT	CHECKSUM
1	Push, Pull Legs Light, for beginners	MASS	20	1	Leg Press	20-MAY-20	20-MAY-21	1	18	5AA001D9A8B5FC5708D98B3E368FE90
2	Push, Pull Legs Light, for beginners	MASS	11	2	Chest Press	20-JUN-21	20-JUN-21	1	18	D9A0072B90783A7CDD499B974F7135E0
3	Full Body Variant Light	CARDIO	10	2	Chest Press	20-JUN-21	20-JUN-21	1	18	D5FB4661AE81EC6240FFB9B176FA97
4	Full Body Variant Light	CARDIO	11	1	Leg Press	20-MAY-20	20-MAY-21	1	18	A83E3CF683F178F56F199494344EBSA0

Funcția întoarce și un checksum pentru a valida datele în sine ulterior, dacă se dorește. Aceste două selecturi sunt prezente în fișierul *bro\_antrenor1\_cript\_show.sql*.

În acest moment, avem 1 antrenor cu o schemă creată și cu date, și adminul cu schema creată și date în ea. În continuare, vom da și restul permisiunilor necesare utilizatorilor, conform matricii entitate-utilizator. Scriptul asociat este în fișierul *sys\_users\_2.sql*. Acest script conține un rol de bază, adică cel pentru publicul general, și celelalte roluri care derivă din acesta, plus alte permisiuni individuale necesare.

Pentru a ilustra criptarea în conexiunea lui client1 vom rula ultima comandă din fișierul *bro\_client1\_select\_cript.sql*



```

26 SELECT ant.*,cs.*,
27       case when ant.checksum = cs.cur_cs then 'ok' else 'not ok' end as cs_v
28 FROM
29       (SELECT bro_admin.get_client_key() AS client_key
30        FROM dual)
31       user_key,
32       LATERAL (SELECT *FROM TABLE(
33         bro_antrenor1.fetch_decrypted_client_data(
34           p_mod_op=>user_key.client_key.mod_op,
35           p_cheie=>user_key.client_key.cheie,
36           p_id_client=>user_key.client_key.id_client))) ant,
37       lateral (
38 select bro_antrenor1.hash_checksum(SYS.ODCIVARCHAR2LIST(ant.id_program,ant.descriere_program,ant.tip_program,
39 ant.durata_antrenament,ant.id_echipament,ant.num_echipament,
40 ant.data_instalare_echipament,ant.data_revizie_echipament,ant.id_filiala,
41 user_key.client_key.cheie)) as cur_cs from dual
42 ) cs;
43

```

ID_PROGRAM	DESCRIERE_PROGRAM	TIP_PROGRAM	DURATA_ANTRENAMENT	ID_ECHIPAMENT	NUM_ECHIPAMENT	DATA_INSTALARE_ECHIPAMENT	DATA_REVIZIE_ECHIPAMENT	ID_FILIALA	ID_CLIENT	CHECKSUM
11	Push, Full Legs Light, for beginners	MASS	20	1	Leg Press	20-MAY-20	20-MAY-21	1	18	5AA001D9A885FCE5708D98E3E398FE90 5
11	Push, Full Legs Light, for beginners	MASS	11	2	Chest Press	20-JUN-21	20-JUN-21	1	18	D9AD072B90783A7CDD499B97487135B0 D
14	Full Body Variant Light	CARDIO	10	2	Chest Press	20-JUN-21	20-JUN-21	1	18	D5FB4661AE81EC662404FEB981768A97 D
44	Full Body Variant Light	CARDIO	11	1	Leg Press	20-MAY-20	20-MAY-21	1	18	A83E3CF683F178F56F199494344EB5A0 A

## 5. Obiect dependent

După ce am rulat scriptul *bro\_antrenor\_insert.sql* în schema mai multor antrenori, în cadrul sys vom rula *sys\_admin\_antrenor\_privilege.sql*. Acest script conține o procedură care oferă adminului permisiunea de select with grant option pe tabela **program** doar pentru schemele antrenorilor care au această tabelă. Avem nevoie de acest grant option, întrucât pentru a face mai ușoară selectarea tuturor antrenamentelor în admin vom crea un view care unește toate aceste programe. Scriptul pentru acest view se găsește în fișierul *bro\_admin\_programs\_view.sql*:

```

51     else
52         dbms_output.put_line('No valid program tables found. View not created. ');
53     end if;
54 end bro_admin_programs_view;
55 /
56 exec bro_admin_programs_view;

```

Procedure BRO\_ADMIN\_PROGRAMS\_VIEW compiled

PL/SQL procedure successfully completed.

```

58 select *
59 from programs_view;

```

ANTRENOR	ID_PROGRAM	DESCRIERE	TIP_PROGRAM
1 BRO ANTRENOR2	1	Push, Pull Legs Light, for beginners	MASS
2 BRO ANTRENOR2	2	Push, Pull Legs Medium	MASS
3 BRO ANTRENOR2	3	Push, Pull Legs Hard	MASS
4 BRO ANTRENOR2	4	Full Body Variant Light	CARDIO
5 BRO ANTRENOR2	5	Body Recovery Variant Light	RECOVERY
6 BRO ANTRENOR2	6	Body Bluster Variant Blusting	RECOVERY
7 BRO ANTRENOR2	7	Cardio Workout for Weight Loss	CARDIO
8 BRO ANTRENOR2	8	Cardio workout for beginners	CARDIO
9 BRO ANTRENOR2	9	Cardio workout for older adults	CARDIO
10 BRO ANTRENOR1	1	Push, Pull Legs Light, for beginners	MASS
11 BRO ANTRENOR1	2	Push, Pull Legs Medium	MASS
12 BRO ANTRENOR1	3	Push, Pull Legs Hard	MASS
13 BRO ANTRENOR1	4	Full Body Variant Light	CARDIO
14 BRO ANTRENOR1	5	Body Recovery Variant Light	RECOVERY
15 BRO ANTRENOR1	6	Body Bluster Variant Blusting	RECOVERY
16 BRO ANTRENOR1	7	Cardio Workout for Weight Loss	CARDIO
17 BRO ANTRENOR1	8	Cardio workout for beginners	CARDIO
18 BRO ANTRENOR1	9	Cardio workout for older adults	CARDIO

Deoarece, vom dori ca oricine să poată accesa acest view, tot în cadrul acestui script vom oferi grant select din admin pentru rolul de bază. (*Obs:* deși adminul nu vede dba roles, el poate da grant pe un rol existent, deoarece are implicit with grant option obiectele din schema sa). Dacă ne conectăm în contul de public general vom vedea că se poate selecta viewul:

```

1 select * from bro_admin.programs_view;
2

```

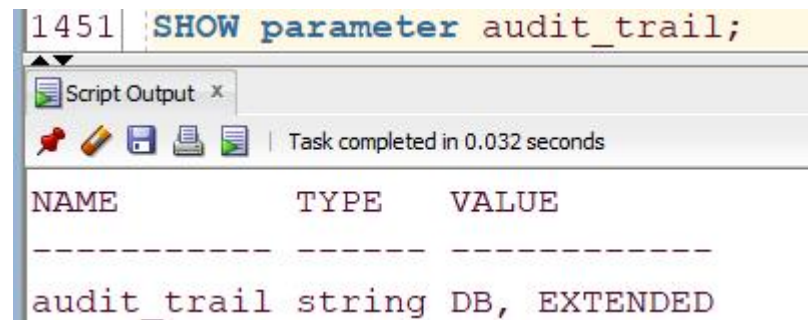
ANTRENOR	ID_PROGRAM	DESCRIERE	TIP_PROGRAM
1 BRO ANTRENOR2	1	Push, Pull Legs Light, for beginners	MASS
2 BRO ANTRENOR2	2	Push, Pull Legs Medium	MASS
3 BRO ANTRENOR2	3	Push, Pull Legs Hard	MASS
4 BRO ANTRENOR2	4	Full Body Variant Light	CARDIO
5 BRO ANTRENOR2	5	Body Recovery Variant Light	RECOVERY
6 BRO ANTRENOR2	6	Body Bluster Variant Blusting	RECOVERY
7 BRO ANTRENOR2	7	Cardio Workout for Weight Loss	CARDIO
8 BRO ANTRENOR2	8	Cardio workout for beginners	CARDIO
9 BRO ANTRENOR2	9	Cardio workout for older adults	CARDIO
10 BRO ANTRENOR1	1	Push, Pull Legs Light, for beginners	MASS
11 BRO ANTRENOR1	2	Push, Pull Legs Medium	MASS
12 BRO ANTRENOR1	3	Push, Pull Legs Hard	MASS
13 BRO ANTRENOR1	4	Full Body Variant Light	CARDIO
14 BRO ANTRENOR1	5	Body Recovery Variant Light	RECOVERY
15 BRO ANTRENOR1	6	Body Bluster Variant Blusting	RECOVERY
16 BRO ANTRENOR1	7	Cardio Workout for Weight Loss	CARDIO
17 BRO ANTRENOR1	8	Cardio workout for beginners	CARDIO
18 BRO ANTRENOR1	9	Cardio workout for older adults	CARDIO



## 6. Audit

### 6.1. Audit Standard

În cadrul proiectului vom avea auditul standard de forma **db,extended**:



```
1451 | SHOW parameter audit_trail;
```

Script Output x

Task completed in 0.032 seconds

NAME	TYPE	VALUE
audit_trail	string	DB, EXTENDED

Vom audita astfel:

1. *bro\_admin.client\_extins*: inserările și actualizările nereușite
2. *bro\_admin.echipament*: inserările, acutalizările și ștergerile
3. *bro\_admin.account\_mapping*: inserările, acutalizările și ștergerile
4. *bro\_admin.supliment* : inserările și actualizarile

Întrucât ținem auditările în baza de date, tabelul poate crește foarte mult, iar pentru a nu pierde datele vom crea în sys o procedură care va salva datele în format json într-un director, iar apoi, pentru a nu fi nevoie să se pună alarmă când se dorește salvarea, vom asocia unor joburi această procedură. Scriptul asociat se află în fișierul *sys\_audit\_1.sql*.

Pentru a distinge fișierele ușor, la salvare acestea vor avea numele: `audit_json_bro_<owner_object>_<object_name>_to_char(sysdate,'YYYYMMDD_HH24MISS').json`

De exemplu, dacă în admin updatăm 'fals' echipamentele de mai multe ori (i.e. facem `where data_revizie=data_revizie`) și apoi rulăm procedura de export, vom avea:

```

5  select *
6  from sys.aud$
7  where obj$name = upper('echipament');

```

Script Output x Query Result x						
SQL   All Rows Fetched: 20 in 0.009 seconds						
	SESSIONID	ENTRYID	STATEMENT	TIMESTAMP#	USERID	USERHOST
1	41585	34	33 (null)	BRO ADMIN	Galma-ROG	
2	41585	36	33 (null)	BRO ADMIN	Galma-ROG	
3	41585	38	33 (null)	BRO ADMIN	Galma-ROG	
4	41585	40	33 (null)	BRO ADMIN	Galma-ROG	
5	41585	6	33 (null)	BRO ADMIN	Galma-ROG	
6	41585	8	33 (null)	BRO ADMIN	Galma-ROG	
7	41585	10	33 (null)	BRO ADMIN	Galma-ROG	

```

1458  exec save_audit_to_json('echipament');
1459

```

Script Output x Query Result x		
Task completed in 0.056 seconds		
NAME	TYPE	VALUE
-----		
audit_trail	string	DB, EXTENDED
1 row deleted.		
PL/SQL procedure successfully completed.		

Iar în File Explorer avem fișierul:

audit\_json\_bro\_BRO\_ADMIN\_echipament\_20250113\_145747.json

Pentru a vedea joburile se va rula:

```
190 SELECT * FROM dba_scheduler_jobs
191 WHERE job_name LIKE upper('save_audit_to_json_job_%')
192 ORDER BY job_name;
```

	OWNER	JOB_NAME	JOB_SUBNAME	JOB_STYLE	JOB_CREATOR
1	SYS	SAVE AUDIT TO JSON JOB ACCOUNT MAPPING	(null)	REGULAR	SYS
2	SYS	SAVE AUDIT TO JSON JOB CLIENT EXTINS	(null)	REGULAR	SYS
3	SYS	SAVE AUDIT TO JSON JOB ECHIPAMENT	(null)	REGULAR	SYS
4	SYS	SAVE AUDIT TO JSON JOB SUPLIMENT	(null)	REGULAR	SYS

## 6.2. Triggeri de Auditare

Pentru triggerii de audit, în schema admin vom crea un tabel care va păstra operațiile DML pentru tabelul **echipament**. În acest audit se va memora atât valoarea veche, cât și cea nouă pentru datele inserate/modificate/șterse în cadrul unei comenzi. Scriptul asociat poate fi găsit în fișierul *bro\_admin\_audit.sql*.

Pentru a ilustra triggerul de audit, vom rula din admin scriptul de update fals, apoi vom selecta din tabelul de audit:

```
select * from audit_echipament;
```

ID_AUDIT	LOG_TIME	OPERATION_TYPE	PERFORMED_BY	ID_ECHIPAMENT	OLD_VALUES	NEW_VALUES	SUMMARY_MESSAGE
81820-DEC-24	03.35.48.6700000000	PM UPDATE	BRO ADMIN	10	[BRO ADMIN.T ECHIPAMENT]	[BRO ADMIN.T ECHIPAMENT]	UPDATE with another 12
81920-DEC-24	03.35.48.6700000000	PM UPDATE	BRO ADMIN	11	[BRO ADMIN.T ECHIPAMENT]	[BRO ADMIN.T ECHIPAMENT]	UPDATE with another 12
82020-DEC-24	03.35.48.6700000000	PM UPDATE	BRO ADMIN	12	[BRO ADMIN.T ECHIPAMENT]	[BRO ADMIN.T ECHIPAMENT]	UPDATE with another 12
82120-DEC-24	03.35.48.6700000000	PM UPDATE	BRO ADMIN	1	[BRO ADMIN.T ECHIPAMENT]	[BRO ADMIN.T ECHIPAMENT]	UPDATE with another 12
82220-DEC-24	03.35.48.6710000000	PM UPDATE	BRO ADMIN	2	[BRO ADMIN.T ECHIPAMENT]	[BRO ADMIN.T ECHIPAMENT]	UPDATE with another 12

```
16 select a.old_values.data_revizie, a.new_values.data_revizie from audit_echipament a;
```

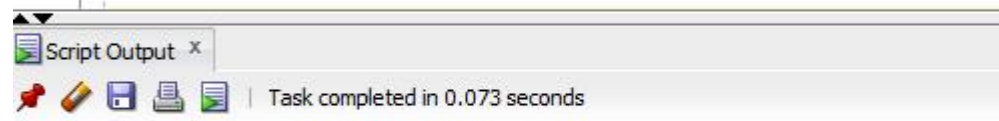
```
17
```

	OLD_VALUES.DATA_REVIZIE	NEW_VALUES.DATA_REVIZIE
1	01-JUN-22	01-JUN-22
2	01-JUN-23	01-JUN-23
3	01-SEP-23	01-SEP-23
4	20-MAY-21	20-MAY-21
5	20-JUN-21	20-JUN-21
6	01-JAN-21	01-JAN-21
7	28-APR-21	28-APR-21

### 6.3. FGA

Pentru FGA vom audita din nou tabela **echipament** pentru a vedea schimbările din coloana **data\_revizie**. De asemenea, vom salva pe disk logurile asociate. Această salvare va fi făcută în cadrul handlerului. Scriptul asociat poate fi găsit în *sys\_audit\_2.sql*.

```
64 |
65 | exec echipament_revizie_audit();
66 |
```



Directory FGADUMP\_DIR created.

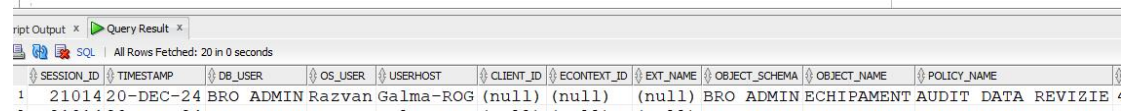
Procedure BRO\_AUDIT\_TABLESE\_HANDLER compiled

Procedure ECHIPAMENT\_REVIZIE\_AUDIT compiled

PL/SQL procedure successfully completed.

Pentru a ilustra FGA vom rula din nou update-ul fals din admin:

```
select * from dba_fga_audit_trail where policy_name='AUDIT_DATA_REVIZIE';
```



Fișierul txt de pe disk va salva, într-un mod append, momentele când auditul a fost activat:



```
FGA Triggered:
Timestamp: 2024-12-20 15:47:41
Object Schema: BRO_ADMIN
Object Name: ECHIPAMENT
Policy Name: AUDIT_DATA_REVIZIE
```

```
FGA Triggered:
Timestamp: 2024-12-20 15:47:41
Object Schema: BRO_ADMIN
Object Name: ECHIPAMENT
Policy Name: AUDIT_DATA_REVIZIE
```

```
FGA Triggered:
Timestamp: 2024-12-20 15:47:41
Object Schema: BRO_ADMIN
Object Name: ECHIPAMENT
Policy Name: AUDIT_DATA_REVIZIE
```

```
FGA Triggered:
Timestamp: 2025-01-13 15:08:47
Object Schema: BRO_ADMIN
Object Name: ECHIPAMENT
Policy Name: AUDIT_DATA_REVIZIE
```

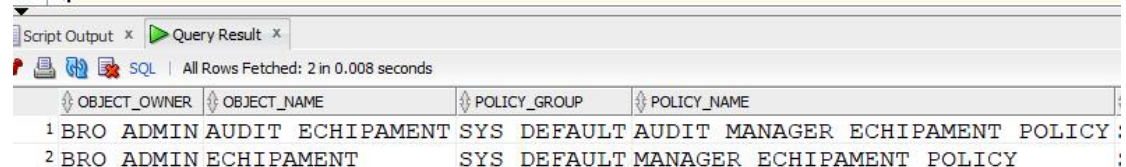
```
FGA Triggered:
Timestamp: 2025-01-13 15:08:47
Object Schema: BRO_ADMIN
Object Name: ECHIPAMENT
Policy Name: AUDIT_DATA_REVIZIE
```



## 7. Contextul aplicației

Vom crea un context care, pentru fiecare manager de filială, va extrage din username filiala asociată. Numele managerilor în baza de date este astfel: `bro_manager_filiala<id_filiala>`, de exemplu, pentru filiala 1 avem userul `bro_manager_filiala1`. De asemenea, contextul va fi folosit în două VPD-uri pentru a asigura că un manager de filială face operații DML doar pe filiala sa și poate accesa din tabelul **audit\_echipament** doar câmpurile care au legătură cu filiala sa. Scriptul asociat este în fișierul `sys_context.sql`:

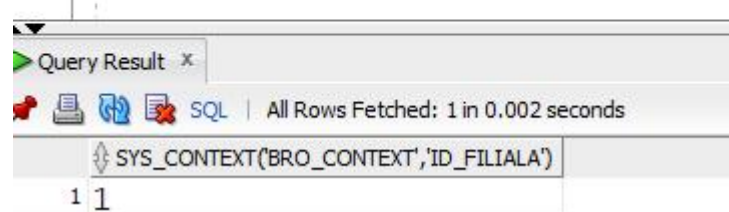
```
30 |
31 | SELECT *
32 | FROM dba_policies
33 | where object_owner like upper('bro%');
34 |
```



	OBJECT_OWNER	OBJECT_NAME	POLICY_GROUP	POLICY_NAME
1	BRO ADMIN AUDIT ECHIPAMENT	SYS	DEFAULT AUDIT MANAGER ECHIPAMENT	POLICY
2	BRO ADMIN ECHIPAMENT	SYS	DEFAULT MANAGER ECHIPAMENT	POLICY

Conectandu-ne în manager1 putem vedea contextul:

```
1 | select sys_context(
2 |     'bro_context',
3 |     'id_filiala'
4 | ) from dual;
```

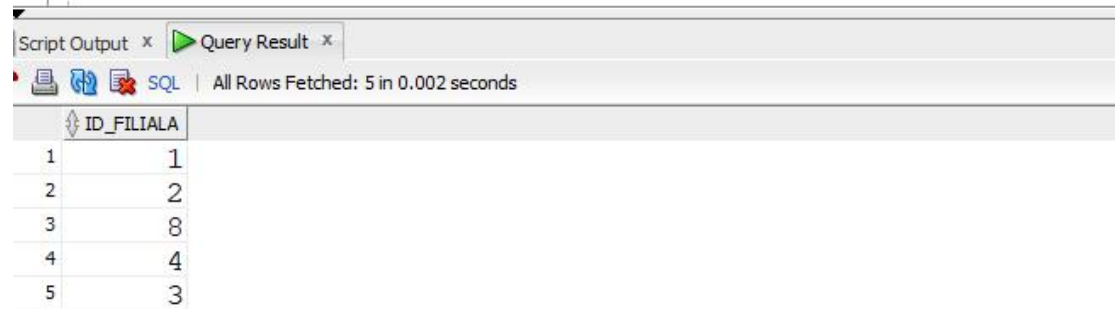


SYS_CONTEXT('BRO_CONTEXT','ID_FILIALA')
1

## 7.1. VPD

De exemplu, pentru VPD de update, dacă se va încerca update pentru echipamentele din filiala 1, se va afișa un număr de linii updatate. În schimb, dacă se încearcă update-ul pe echipamente din alte filiale, se vor afișa 0 linii updatate, fără nicio eroare, deși avem echipamente și în alte filiale. SQL-ul asociat managerului este în fișierul *bro\_manager\_filiala1\_context.sql*.

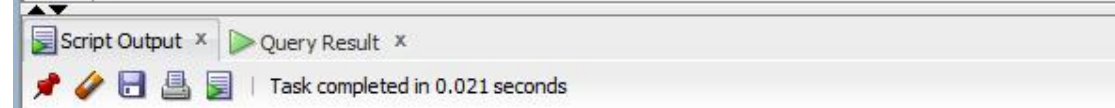
```
6 |
7 | select distinct id_filiala from bro_admin.echipament;
8 |
```



The screenshot shows the SQL Developer interface. The 'Query Result' tab is active, displaying a table with one column, 'ID\_FILIALA'. The table contains five rows of data: (1, 1), (2, 2), (3, 8), (4, 4), and (5, 3). The status bar indicates 'All Rows Fetched: 5 in 0.002 seconds'.

ID_FILIALA
1
2
8
4
3

```
8 |
9 | update bro_admin.echipament o
10 |     set
11 |         nume = (
12 |             select nume
13 |             from bro_admin.echipament i
14 |             where i.id_echipament = o.id_echipament
15 |         )
16 |     where o.id_filiala = 1;
17 |
```



The screenshot shows the SQL Developer interface. The 'Script Output' tab is active, displaying the message '2 rows updated.'. The status bar indicates 'Task completed in 0.021 seconds'.

2 rows updated.

```
18 update bro_admin.echipament o
19 set
20     nume = (
21         select nume
22         from bro_admin.echipament i
23         where i.id_echipament = o.id_echipament
24     )
25 where o.id_filiala != 1;
26
```

Script Output x Query Result x

Task completed in 0.022 seconds

0 rows updated.

```
27 update bro_admin.echipament o
28 set
29     nume = (
30         select nume
31         from bro_admin.echipament i
32         where i.id_echipament = o.id_echipament
33     );
```

Script Output x Query Result x

Task completed in 0.02 seconds

2 rows updated.

Politica de select pe tabelul de audit are rolul de a lăsa managerul unei filiale să vadă doar auditul pe echipamentele care fie au fost în filiala sa (i.e. old\_values.id\_filiala=1 aici), fie sunt (i.e. new\_values.id\_filiala=1):



```

36 select count(*)
37 from bro_admin.audit_echipament a
38 where a.old_values.id_filiala=1 or a.old_values.id_filiala=1;

```

Script Output x Query Result x

SQL | All Rows Fetched: 1 in 0.001 seconds

	COUNT(*)
1	212

```

40 select count(*)
41 from bro_admin.audit_echipament a
42 where a.old_values.id_filiala!=1 and a.old_values.id_filiala!=1;

```

Script Output x Query Result x

SQL | All Rows Fetched: 1 in 0.001 seconds

	COUNT(*)
1	0

Dacă rulăm în admin ultimul select, vom vedea că valoarea este diferită de 0:

```

1941 select count(*)
1942 from bro_admin.audit_echipament a
1943 where a.old_values.id_filiala!=1 and a.old_values.id_filiala!=1;

```

Script Output x Query Result x

SQL | All Rows Fetched: 1 in 0.002 seconds

	COUNT(*)
1	1000

## 8. SQL injection

### 8.1. Procedura Vulnerabilă

Pentru SQL injection, să presupunem că antrenor1 vrea să creeze o procedură care permite utilizatorilor să vadă un program cu echipamentele care vor fi folosite în cadrul acestuia, filtrând echipamentele după data reviziei. Scriptul este în fișierul *bro\_antrenor1\_sql\_injection.sql*. Procedura va primi doi parametri: primul, id-ul programului, iar cel de-al doilea, data reviziei echipamentelor. Partea relevantă a procedurii este:

```

'SELECT * FROM bro_admin.echipament e
NATURAL JOIN program p
WHERE p.id_program = '

```

```

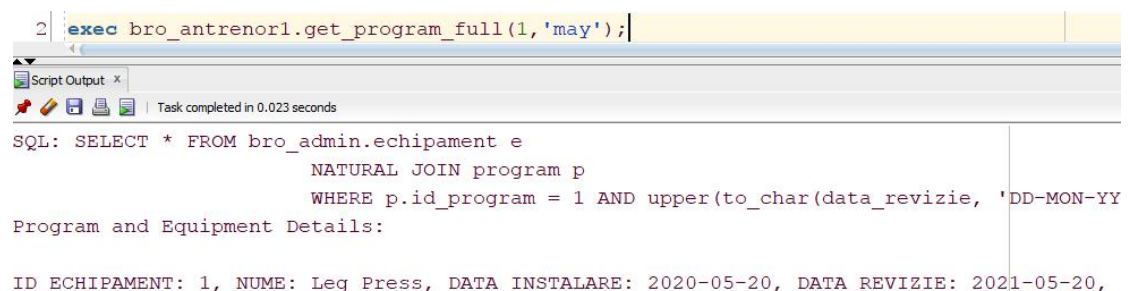
|| id_prg ||
' AND upper(to_char(data_revizie, 'DD-MON-YY')) LIKE '%'
|| upper(data_inst)||
'%'

```

După cum se poate observa, inputul este direct concatenat în selectul care va fi transmis motorului bazei de date, fără a fi sanitizat.

Pentru a putea rula procedura din antrenor, în cadrul fișierului menționat vom da drepturi de execuție lui client1. Vom rula scriptul *bro\_client1\_sql\_injection.sql* pentru a demonstra un apel onest și două apeluri menite să arate vulnerabilitățile procedurii:

Apel onest:



```

2 | exec bro_antrenor1.get_program_full(1, 'may');

```

Script Output x

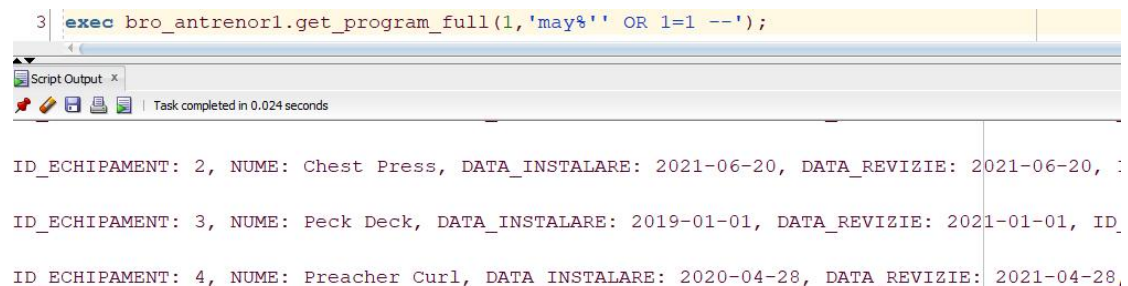
Task completed in 0.023 seconds

SQL: SELECT \* FROM bro\_admin.echipament e  
NATURAL JOIN program p  
WHERE p.id\_program = 1 AND upper(to\_char(data\_revizie, 'DD-MON-YY')) LIKE '%MAY%'

Program and Equipment Details:

ID\_ECHIPAMENT: 1, NUME: Leg Press, DATA\_INSTALARE: 2020-05-20, DATA\_REVIZIE: 2021-05-20,

Apel care întoarce toate programele cu echipamentele asociate, subminând filtrarea:



```

3 | exec bro_antrenor1.get_program_full(1, 'may%' OR 1=1 --');

```

Script Output x

Task completed in 0.024 seconds

ID\_ECHIPAMENT: 2, NUME: Chest Press, DATA\_INSTALARE: 2021-06-20, DATA\_REVIZIE: 2021-06-20, ID\_PROGRAM: 1

ID\_ECHIPAMENT: 3, NUME: Peck Deck, DATA\_INSTALARE: 2019-01-01, DATA\_REVIZIE: 2021-01-01, ID\_PROGRAM: 1

ID\_ECHIPAMENT: 4, NUME: Preacher Curl, DATA\_INSTALARE: 2020-04-28, DATA\_REVIZIE: 2021-04-28, ID\_PROGRAM: 1

Apel care întoarce toate antrenamentele, deși clientul nu are drept de select pe tabela antrenament:

```
9
10 select * from bro_antrenor1.antrenament;
11
```

Script Output x Query Result x

SQL | Executing:select \* from bro\_antrenor1.antrenament in 0 seconds

ORA-00942: table or view does not exist  
00942. 00000 - "table or view does not exist"  
\*Cause:  
\*Action:  
Error at Line: 10 Column: 29

```
14 begin
15     bro_antrenor1.get_program_full(1, 'may%' UNION SELECT ID_ECHIPAMENT, 'Injectat',
16                                     SYSDATE, SYSDATE, ID_CLIENT, DURATA, ID_PROGRAM,
17                                     'Injectat Desc', 'Tip injectat' FROM ANTRENAMENT --');
18 end;
```

Script Output x Query Result x

Task completed in 0.042 seconds

ID_ECHIPAMENT: 1,	NUME: INJECTAT,	DATA_INSTALARE: 2025-01-14,	DATA_REVIZIE: 2025-01-14,	ID_FILIALA: 18,	ID_FURNIZOR: 11,	ID_PROGRAM: 4,
ID_ECHIPAMENT: 1,	NUME: INJECTAT,	DATA_INSTALARE: 2025-01-14,	DATA_REVIZIE: 2025-01-14,	ID_FILIALA: 18,	ID_FURNIZOR: 20,	ID_PROGRAM: 1,
ID_ECHIPAMENT: 1,	NUME: INJECTAT,	DATA_INSTALARE: 2025-01-14,	DATA_REVIZIE: 2025-01-14,	ID_FILIALA: 19,	ID_FURNIZOR: 5,	ID_PROGRAM: 1,
ID_ECHIPAMENT: 1,	NUME: Leg Press,	DATA_INSTALARE: 2020-05-20,	DATA_REVIZIE: 2021-05-20,	ID_FILIALA: 1,	ID_FURNIZOR: 5,	ID_PROGRAM: 1,
ID_ECHIPAMENT: 2,	NUME: INJECTAT,	DATA_INSTALARE: 2025-01-14,	DATA_REVIZIE: 2025-01-14,	ID_FILIALA: 18,	ID_FURNIZOR: 10,	ID_PROGRAM: 4,
ID_ECHIPAMENT: 2,	NUME: INJECTAT,	DATA_INSTALARE: 2025-01-14,	DATA_REVIZIE: 2025-01-14,	ID_FILIALA: 18,	ID_FURNIZOR: 11,	ID_PROGRAM: 1,
ID_ECHIPAMENT: 2,	NUME: INJECTAT,	DATA_INSTALARE: 2025-01-14,	DATA_REVIZIE: 2025-01-14,	ID_FILIALA: 19,	ID_FURNIZOR: 10,	ID_PROGRAM: 5,
ID_ECHIPAMENT: 2,	NUME: INJECTAT,	DATA_INSTALARE: 2025-01-14,	DATA_REVIZIE: 2025-01-14,	ID_FILIALA: 19,	ID_FURNIZOR: 25,	ID_PROGRAM: 1,
ID_ECHIPAMENT: 2,	NUME: INJECTAT,	DATA_INSTALARE: 2025-01-14,	DATA_REVIZIE: 2025-01-14,	ID_FILIALA: 21,	ID_FURNIZOR: 42,	ID_PROGRAM: 4,
ID_ECHIPAMENT: 3,	NUME: INJECTAT,	DATA_INSTALARE: 2025-01-14,	DATA_REVIZIE: 2025-01-14,	ID_FILIALA: 19,	ID_FURNIZOR: 32,	ID_PROGRAM: 1,

## 8.2. Procedura repartă

Pentru a face procedura mai sigură la atacuri de tip injection, se va schimba crearea query-ului care folosește parametrii de intrare: se va înlocui simpla concatenare cu binding, astfel:

```
'SELECT
    e.id_echipament,
    e.numa,
    e.data_instalare,
    e.data_revizie,
    e.id_filiala,
    e.id_furnizor,
    p.id_program,
```

```

        p.descriere,
        p.tip_program
        FROM bro_admin.echipament e
        NATURAL JOIN program p
        WHERE p.id_program = :id_prg
        AND
        UPPER(TO_CHAR(e.data_revizie, 'DD-MON-YY'))
LIKE :data_inst'

```

Iar, apelarea sa va fi urmatoarea:

```

EXECUTE IMMEDIATE v_sql BULK COLLECT
INTO v_program_echipament USING id_prg
, '%' || upper(data_inst) || '%';

```

Dacă se va încerca oricare dintre apelurile rău intenționate, procedura nu va întoarce nicio linie, întrucât în acest moment apelantul nu mai are posibilitatea de a altera structura efectivă a stringului de interogare:

```

30 | exec bro_antrenor1.get_program_full_safe(1, 'may%' UNION SELECT ID_ECHIPAMENT, 'Injectat',
Script Output x Query Result x
Task completed in 0.032 seconds

        p.id_program,
        p.descriere,
        p.tip_program
        FROM bro_admin.echipament e
        NATURAL JOIN program p
        WHERE p.id_program = :id_prg
        AND UPPER(TO_CHAR(e.data_revizie, 'DD-MON-YY')) LIKE :data_inst
Program and Equipment Details:

PL/SQL procedure successfully completed.

```

## 9. Mascarea datelor

Pentru mascarea datelor se vor exporta persoanele din baza de date modificând astfel coloanele:

1. Valorile coloanelor numerice care nu sunt chei se vor schimba în valori care încep cu aceeași cifră și au aceeași lungime, restul de cifre vor fi random.
2. Valorile coloanelor de tip string se vor schimba astfel: prima dată se alege random dacă se va dubla lungimea stringului, după care se păstrează primul caracter, apoi se adaugă random până la noua lungime câte un caracter '\*' sau '#'.

3. Cheile își vor păstra unicitatea dar vor putea avea dimensiunea până de 5 ori mai mare.

## 9.1. Export

Pentru export se va crea în sys un nou director, iar adminul va primi permisiuni pe acesta. Pentru import se va crea un nou utilizator cu drepturi de import, ie *datapump\_imp\_full\_database*, pentru a putea remapa schema lui *bro\_admin* la schema noului utilizator. S-a ales acest model pentru a demonstra exportul și importul păstrând constrângerile inițiale ale tabelelor.

Pentru sys, fișierul asociat este *sys\_mask.sql*, în care se creează noul user, directorul și se dau drepturile asociate.

În fișierul *bro\_admin\_mask.sql* se găsește definirea pachetului care realizează maparea.

```
Package MASK_PERSON compiled
```

```
Package Body MASK_PERSON compiled
```

Comnada de realizare a mapării se află în *mask\_person.cmd*:

```
PS C:\Master\An2\seml\securitateBD\proiect\sql\final\cmd> .\mask_person.cmd

Export: Release 19.0.0.0.0 - Production on Mon Jan 13 17:37:48 2025
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved.

Connected to: Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Starting "BRO_ADMIN". "SYS_EXPORT_TABLE_01": bro_admin/*****@//localhost:1522/orclpdb tables=BRO_ADMIN.PERSOANA, BRO_ADMIN.ANGAJAT, BRO_ADMIN
.ANTRENOR, BRO_ADMIN.RECEPTIONIST, BRO_ADMIN.CLIENT remap_data=persoana.id_persoana:mask_person.mask_person_id remap_data=persoana.num:mask_per
son.mask_item remap_data=persoana.prenume:mask_person.mask_item remap_data=persoana.email:mask_person.mask_item remap_data=persoana.varsta:mask
person.mask_item remap_data=angajat.id_angajat:mask_person.mask_person_fk remap_data=angajat.salariu:mask_person.mask_item remap_data=angajat.id
_meneger:mask_person.mask_person_fk remap_data=antrenor.id_antrenor:mask_person.mask_person_fk remap_data=receptionist.id_receptionist:mask_pers
on.mask_person_fk remap_data=client.id_client:mask_person.mask_person_fk directory=MASK_DUMP parallel=8 dumpfile=mask_person.dmp logfile=mask_pe
rson.log reuse_dumpfiles=y
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type TABLE_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/REF_CONSTRAINT
Processing object type TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type TABLE_EXPORT/TABLE/STATISTICS/MARKER
. . exported "BRO_ADMIN"."ANGAJAT" 7.312 KB 17 rows
. . exported "BRO_ADMIN"."ANTRENOR" 5.679 KB 7 rows
. . exported "BRO_ADMIN"."CLIENT" 5.625 KB 10 rows
. . exported "BRO_ADMIN"."PERSOANA" 7.875 KB 27 rows
. . exported "BRO_ADMIN"."RECEPTIONIST" 5.632 KB 10 rows
Master table "BRO_ADMIN"."SYS_EXPORT_TABLE_01" successfully loaded/unloaded
*****
Dump file set for BRO_ADMIN.SYS_EXPORT_TABLE_01 is:
D:\ORACLEEE\INSTALL\ADMIN\ORCL\MASKDUMP\MASK_PERSON.DMP
Job "BRO_ADMIN"."SYS_EXPORT_TABLE_01" successfully completed at Mon Jan 13 17:37:54 2025 elapsed 0 00:00:05

Done exporting mask persoana
PS C:\Master\An2\seml\securitateBD\proiect\sql\final\cmd> |
```

*Obs:* Deși am pus tables într-o anumită ordine, Oracle ia tables alfabetic, în minunata lor documentație nu am găsit nimic. Așa că a trebuit să preinitializez la

mask\_person\_id și mask\_person\_fk cheile din persoană, dacă stateul de chei este gol. Nu cred că este un comportament normal, pe internet nu am găsit pe cineva să se plângă de acest aspect.

(Nu am folosit package body init, ie un begin, pentru a asigura că datele din persoană sunt cele din momentul exportului.)

## 9.2. Import

Pentru import în schema bro\_import, sa va rula scriptul *import\_mask\_person.cmd*:

```
PS C:\Master\An2\sem1\securitateaBD\proiect\sql\final\cmd> .\import_mask_person.cmd

Import: Release 19.0.0.0.0 - Production on Mon Jan 13 17:44:09 2025
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved.

Connected to: Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Master table "BRO_IMPORT"."SYS_IMPORT_FULL_01" successfully loaded/unloaded
Starting "BRO_IMPORT"."SYS_IMPORT_FULL_01": bro_import/*****@//localhost:1522/orclpdb remap_table=persoana:persoana_mask remap_table=angajat
:angajat_mask remap_table=antrenor:antrenor_mask remap_table=receptionist:receptionist_mask remap_table=client:client_mask remap_schema=bro_admi
n:bro_import directory=MASK_DUMP dumpfile=mask_person.dmp logfile=mask_person_import.log parallel=8 transform=disable_archive_logging:y
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
.. imported "BRO_IMPORT"."ANGAJAT_MASK" 7.312 KB 17 rows
.. imported "BRO_IMPORT"."ANTRENOR_MASK" 5.679 KB 7 rows
.. imported "BRO_IMPORT"."CLIENT_MASK" 5.625 KB 10 rows
.. imported "BRO_IMPORT"."PERSONA_MASK" 7.875 KB 27 rows
.. imported "BRO_IMPORT"."RECEPTIONIST_MASK" 5.632 KB 10 rows
Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type TABLE_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/REF_CONSTRAINT
ORA-39083: Object type REF_CONSTRAINT:"BRO_IMPORT"."FK_ANGAJAT_FILIALA" failed to create with error:
ORA-00942: table or view does not exist

Failing sql is:
ALTER TABLE "BRO_IMPORT"."ANGAJAT_MASK" ADD CONSTRAINT "FK_ANGAJAT_FILIALA" FOREIGN KEY ("ID_FILIALA") REFERENCES "BRO_IMPORT"."FILIALA" ("ID_FI
LIALA") ON DELETE CASCADE ENABLE

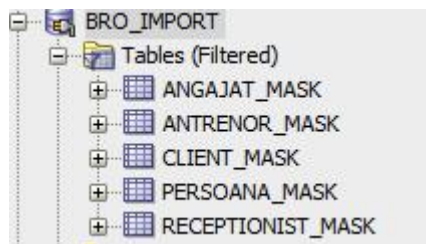
Processing object type TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/MARKER
Job "BRO_IMPORT"."SYS_IMPORT_FULL_01" completed with 1 error(s) at Mon Jan 13 17:44:16 2025 elapsed 0 00:00:06

Done importing mask persoana
PS C:\Master\An2\sem1\securitateaBD\proiect\sql\final\cmd> |
```

Întrucât am exportat doar tabelele **persoana**, **angajat**, **antrenor**, **receptionist** și **client**, nu și tabela **filiala**, și am păstrat la export constrângerile, la import vom avea o eroare care spune că nu se poate rezolva constrângerea de FK pentru tabela **filiala**. Pentru că suntem conștienți că nu am exportat acea tabelă, putem ignora această eroare, întrucât singurul lucru care se va întâmpla este că în tabela **angajat\_mask**, cea importată, nu vom mai avea acea constrângere de FK.

Dacă deschidem o conexiune cu userul bro\_import, vom putea constata crearea tabelelor, iar pentru angajat mapat nu este prezentă constrângerea de FK pe **filiala**. De asemenea, în tabele vor fi prezente datele mapate:





```

7  SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE
8  FROM USER_CONSTRAINTS
9  WHERE TABLE_NAME = upper('angajat_mask');

```

Script Output x Query Result x

SQL | All Rows Fetched: 7 in 1.185 seconds

	CONSTRAINT_NAME	CONSTRAINT_TYPE
1	FK ANGAJAT PERSOANA	R
2	FK ANGAJAT ANGAJAT	R
3	NN ANGAJAT DATA ANGJARE	C
4	SYS C009010	C
5	SYS C009011	C
6	CK ANGAJAT SALARIU	C
7	PK ANGAJAT	P

```

11 select * from persoana_mask;
12

```

Script Output x Query Result x

SQL | All Rows Fetched: 27 in 0.005 seconds

	ID_PERSOANA	NUME	PRENUME	EMAIL	VARSTA
1	15019	P#####	I##	p#####	36
2	2966	P#####	G#####	p#####	31
3	25895	I#####	A#####	i#####	21
4	1811	I##	I	i#####	21
5	44447	M**	M##	m#####	28
6	56060	A#####	A#	a#####	36
7	8507	S#####	R##	s#####	25
8	64223	D#	A###	d#####	28
9	42072	V####	M#	v#####	28
10	234148877	P***	G##	p#####	28
11	1600561812	P#	M##	p#####	21
12	1532896655	D*#####	A**	d#####	28
13	850249301	M**	I***	m#####	63
14	958675650	D#	I#	d#####	42
15	351498841	M*#####	I	m#####	21
16	411112727	I*	A*	i#####	24
17	1183066573	D#####	S***	d#####	36
18	303611716	V****	R#	v#####	21
19	1881622286	I#*	A***	i#####	18

```
select * from angajat_mask
join persoana_mask
on id_angajat=id_persoana;
```

t Output x Query Result x									
All Rows Fetched: 17 in 0.002 seconds									
ID_ANGAJAT	DATA_ANGAJARE	SALARIU	ID_FILIALA	ID_MENEGER	ID_PERSOANA	NUME	PRENUME	EMAIL	VARSTA
1811 01-FEB-15	2483	1	8507	1811 I*##	I##	i*****##*##			21
2966 01-FEB-15	2851	1	8507	2966 P*##	G##	p*****##*##			31
8507 15-APR-05	3557	1	(null)	8507 S*##	R##	s*****##*##			25
15019 11-JAN-20	1303	1	8507	15019 P*##	I*##	p*****##*##			36
25895 20-MAR-17	2616	1	8507	25895 I***	A***##	i*****##*##			21
42072 01-JUL-19	1333	1	8507	42072 V*##	M***	v*****##*##			28
44447 01-MAY-21	2364	1	8507	44447 M*##	M**	m*****##			28
56060 01-JAN-10	5680	1	8507	56060 A*##	A***	a*****##*##			36
64223 01-JUN-01	2364	1	8507	64223 D##	A***##	d*****##*##			28

SQL-ul asociat userului de import se găsește în fișierul *bro\_import.sql*.



## 10. Codul SQL al aplicatiei

### 10.1. Admin

#### 10.1.1 bro\_admin\_antrenor\_seed.sql

```
-- Seed pentru schema bro_antrenor rulat de catre bro_admin
GRANT SELECT,REFERENCES ON bro_admin.antrenor TO &&user_name;
GRANT REFERENCES,SELECT ON bro_admin.echipament TO &&user_name;
GRANT REFERENCES ON bro_admin.client TO &&user_name;
GRANT REFERENCES,SELECT ON bro_admin.chei_client TO &&user_name;
GRANT EXECUTE ON dbms_crypto TO &&user_name;

CREATE TABLE &&user_name..PROGRAM (
  ID_PROGRAM  NUMBER(*,0)
    GENERATED BY DEFAULT ON NULL AS IDENTITY
    CONSTRAINT PK_PROGRAM PRIMARY KEY,
  DESCRIERE   VARCHAR2(255),
  TIP_PROGRAM VARCHAR2(20)
    CONSTRAINT CK_PROGRAM_TIP_PROGRAM NOT NULL
  CHECK ( TIP_PROGRAM IN ( 'CARDIO',
                           'MASS',
                           'RECOVERY' ) )
);

CREATE TABLE &&USER_NAME..ANTRENAMENT (
  ID_CLIENT    NUMBER(*,0),
  ID_PROGRAM   NUMBER(*,0),
  ID_ECHIPAMENT NUMBER(*,0),
  DURATA       NUMBER(3)
    CONSTRAINT NN_ANTRENAMENT_DURATA NOT NULL,
  CONSTRAINT PK_ANTRENAMENT PRIMARY KEY ( ID_CLIENT,
                                           ID_PROGRAM,
                                           ID_ECHIPAMENT ),
  CONSTRAINT FK_ANTRENAMENT_CLIENT FOREIGN KEY ( ID_CLIENT )
    REFERENCES BRO_ADMIN.CLIENT ( ID_CLIENT ),
  CONSTRAINT FK_ANTRENAMENT_PROGRAM FOREIGN KEY ( ID_PROGRAM )
    REFERENCES &&USER_NAME..PROGRAM ( ID_PROGRAM ),
  CONSTRAINT FK_ANTRENAMENT_ECHIPAMENT FOREIGN KEY ( ID_ECHIPAMENT )
    REFERENCES BRO_ADMIN.ECHIPAMENT ( ID_ECHIPAMENT )
);

CREATE OR REPLACE FUNCTION &&USER_NAME..CRIPT_STRING (
  ORG   VARCHAR2,
  MOD_OP PLS_INTEGER,
  CHEIE RAW
) RETURN RAW IS
BEGIN
  RETURN DBMS_CRYPTO.ENCRYPT(
    UTL_I18N.STRING_TO_RAW(
      ORG,
```

```

        'AL32UTF8'
    ),
    MOD_OP,
    CHEIE
);
END;
/

CREATE OR REPLACE FUNCTION &&USER_NAME..DECRYPT_STRING (
    CRIPT RAW,
    MOD_OP PLS_INTEGER,
    CHEIE RAW
) RETURN VARCHAR2 IS
    RESULT VARCHAR2(4000);
BEGIN
    RESULT := UTL_I18N.RAW_TO_CHAR(
        DBMS_CRYPTO.DECRYPT(
            CRIPT,
            MOD_OP,
            CHEIE
        ),
        'AL32UTF8'
    );

    IF RESULT IS NULL
    OR LENGTH(RESULT) = 0 THEN
        RETURN 'Not allowed';
    END IF;
    RETURN RESULT;
EXCEPTION
    WHEN OTHERS THEN
        RETURN 'Not allowed';
END;
/

CREATE OR REPLACE FUNCTION &&USER_NAME..HASH_CHECKSUM (
    INPUT_ARRAY SYS.ODCIVARCHAR2LIST
) RETURN RAW IS
    CONCATENATED_STRING VARCHAR2(4000);
BEGIN
    CONCATENATED_STRING := '';
    FOR I IN 1..INPUT_ARRAY.COUNT LOOP
        CONCATENATED_STRING := CONCATENATED_STRING || INPUT_ARRAY(I);
    END LOOP;

    RETURN DBMS_CRYPTO.HASH(
        UTL_I18N.STRING_TO_RAW(
            CONCATENATED_STRING,
            'AL32UTF8'
        ),
        DBMS_CRYPTO.HASH_MD5
    );
END;

```

/

```
CREATE OR REPLACE VIEW &&USER_NAME..CLIENT_ANTRENAMENT AS
SELECT CRIPT_STRING(
    P.ID_PROGRAM,
    C.MOD_OP,
    C.CHEIE
) AS ID_PROGRAM,
    CRIPT_STRING(
        P.DESCRIERE,
        C.MOD_OP,
        C.CHEIE
    ) AS DESCRIERE_PROGRAM,
    CRIPT_STRING(
        P.TIP_PROGRAM,
        C.MOD_OP,
        C.CHEIE
    ) AS TIP_PROGRAM,
    CRIPT_STRING(
        A.DURATA,
        C.MOD_OP,
        C.CHEIE
    ) AS DURATA_ANTRENAMENT,
    CRIPT_STRING(
        E.ID_ECHIPAMENT,
        C.MOD_OP,
        C.CHEIE
    ) AS ID_ECHIPAMENT,
    CRIPT_STRING(
        E.NUME,
        C.MOD_OP,
        C.CHEIE
    ) AS NUME_ECHIPAMENT,
    CRIPT_STRING(
        E.DATA_INSTALARE,
        C.MOD_OP,
        C.CHEIE
    ) AS DATA_INSTALARE_ECHIPAMENT,
    CRIPT_STRING(
        E.DATA_REVIZIE,
        C.MOD_OP,
        C.CHEIE
    ) AS DATA_REVIZIE_ECHIPAMENT,
    CRIPT_STRING(
        E.ID_FILIALA,
        C.MOD_OP,
        C.CHEIE
    ) AS ID_FILIALA,
    CRIPT_STRING(
        A.ID_CLIENT,
        C.MOD_OP,
        C.CHEIE
    ) AS ID_CLIENT,
```

```

        HASH_CHECKSUM(SYS.ODCIVARCHAR2LIST(
            P.ID_PROGRAM,
            P.DESCRIERE,
            P.TIP_PROGRAM,
            A.DURATA,
            E.ID_ECHIPAMENT,
            E.NUME,
            E.DATA_INSTALARE,
            E.DATA_REVIZIE,
            E.ID_FILIALA,
            C.CHEIE
        )) AS CHECKSUM
    FROM &&USER_NAME..PROGRAM P
    JOIN &&USER_NAME..ANTRENAMENT A
    ON P.ID_PROGRAM = A.ID_PROGRAM
    JOIN BRO_ADMIN.ECHIPAMENT E
    ON A.ID_ECHIPAMENT = E.ID_ECHIPAMENT
    JOIN BRO_ADMIN.CHEI_CLIENT C
    ON C.ID_CLIENT = A.ID_CLIENT;

CREATE OR REPLACE FUNCTION &&USER_NAME..NUMBER_TO_RAW (
    N NUMBER
) RETURN RAW IS
BEGIN
    RETURN HEXTORAW(TO_CHAR(
        N,
        'FM0X'
    ));
END;
/

CREATE OR REPLACE TYPE &&USER_NAME..DECRYPTED_CLIENT_RECORD AS OBJECT (
    ID_PROGRAM          VARCHAR2(100),
    DESCRIERE_PROGRAM   VARCHAR2(2500),
    TIP_PROGRAM          VARCHAR2(100),
    DURATA_ANTRENAMENT  VARCHAR2(50),
    ID_ECHIPAMENT        VARCHAR2(100),
    NUME_ECHIPAMENT      VARCHAR2(255),
    DATA_INSTALARE_ECHIPAMENT VARCHAR2(50),
    DATA_REVIZIE_ECHIPAMENT VARCHAR2(50),
    ID_FILIALA           VARCHAR2(100),
    ID_CLIENT            VARCHAR2(100),
    CHECKSUM             RAW(16)
);
/

CREATE OR REPLACE TYPE &&USER_NAME..DECRYPTED_CLIENT_TABLE AS
    TABLE OF &&USER_NAME..DECRYPTED_CLIENT_RECORD;
/

CREATE OR REPLACE FUNCTION &&USER_NAME..FETCH_DECRYPTED_CLIENT_DATA (
    P_MOD_OP    NUMBER,

```

```

P_CHEIE      RAW,
P_ID_CLIENT  NUMBER
) RETURN &&USER_NAME..DECRYPTED_CLIENT_TABLE
PIPELINED
IS
BEGIN
FOR R IN (
SELECT DECRYPT_STRING(
CA.ID_PROGRAM,
P_MOD_OP,
P_CHEIE
) AS ID_PROGRAM,
DECRYPT_STRING(
CA.DESCRIERE_PROGRAM,
P_MOD_OP,
P_CHEIE
) AS DESCRIERE_PROGRAM,
DECRYPT_STRING(
CA.TIP_PROGRAM,
P_MOD_OP,
P_CHEIE
) AS TIP_PROGRAM,
DECRYPT_STRING(
CA.DURATA_ANTRENAMENT,
P_MOD_OP,
P_CHEIE
) AS DURATA_ANTRENAMENT,
DECRYPT_STRING(
CA.ID_ECHIPAMENT,
P_MOD_OP,
P_CHEIE
) AS ID_ECHIPAMENT,
DECRYPT_STRING(
CA.NUME_ECHIPAMENT,
P_MOD_OP,
P_CHEIE
) AS NUME_ECHIPAMENT,
DECRYPT_STRING(
CA.DATA_INSTALARE_ECHIPAMENT,
P_MOD_OP,
P_CHEIE
) AS DATA_INSTALARE_ECHIPAMENT,
DECRYPT_STRING(
CA.DATA_REVIZIE_ECHIPAMENT,
P_MOD_OP,
P_CHEIE
) AS DATA_REVIZIE_ECHIPAMENT,
DECRYPT_STRING(
CA.ID_FILIALA,
P_MOD_OP,
P_CHEIE
) AS ID_FILIALA,

```

```

        DECRYPT_STRING(
            CA.ID_CLIENT,
            P_MOD_OP,
            P_CHEIE
        ) AS ID_CLIENT,
        CHECKSUM
    FROM &&USER_NAME..CLIENT_ANTRENAMENT CA
    WHERE REGEXP_LIKE ( DECRYPT_STRING(
        CA.ID_CLIENT,
        P_MOD_OP,
        P_CHEIE
    ),
        '^\\d+$' )
) LOOP
    PIPE ROW ( decrypted_client_record(
        r.id_program,
        r.descriere_program,
        r.tip_program,
        r.durata_antrenament,
        r.id_echipament,
        r.nume_echipament,
        r.data_instalare_echipament,
        r.data_revizie_echipament,
        r.id_filiala,
        r.id_client,
        r.checksum
    ) );
END LOOP;
RETURN;
end;
/

COMMIT;

EXIT;

```

### 10.1.2. bro\_admin\_audit.sql

```

CREATE OR REPLACE TYPE t_echipament AS OBJECT (
    id_echipament INT,
    nume          VARCHAR2(40),
    data_instalare DATE,
    data_revizie  DATE,
    id_filiala    INT,
    id_furnizor   INT
);
/

CREATE TABLE audit_echipament (
    id_audit      INT
    GENERATED BY DEFAULT AS IDENTITY

```

```

PRIMARY KEY,
log_time          TIMESTAMP DEFAULT systimestamp,
operation_type    VARCHAR2(15),
performed_by     VARCHAR2(128),
id_echipament    INT,
old_values        t_echipament,
new_values        t_echipament,
summary_message   VARCHAR2(2500)
);

CREATE OR REPLACE TRIGGER audit_echipament_trg FOR
INSERT OR UPDATE OR DELETE ON echipament
COMPOUND TRIGGER
    TYPE t_row_change IS RECORD (
        operation_type VARCHAR2(15),
        id_echipament   INT,
        old_values       t_echipament,
        new_values       t_echipament
    );
    TYPE t_change_table IS
        TABLE OF t_row_change INDEX BY PLS_INTEGER;
    g_changes  t_change_table;
    g_count_op INT := 0;
    BEFORE EACH ROW IS BEGIN
        IF inserting THEN
            g_count_op := g_count_op + 1;
            g_changes(g_changes.count + 1) := t_row_change(
                'INSERT',
                :new.id_echipament,
                NULL,
                t_echipament(
                    :new.id_echipament,
                    :new.nume,
                    :new.data_instalare,
                    :new.data_revizie,
                    :new.id_filiala,
                    :new.id_furnizor
                )
            );
        ELSIF updating THEN
            g_count_op := g_count_op + 1;
            g_changes(g_changes.count + 1) := t_row_change(
                'UPDATE',
                :new.id_echipament,
                t_echipament(
                    :old.id_echipament,
                    :old.nume,
                    :old.data_instalare,
                    :old.data_revizie,
                    :old.id_filiala,
                    :old.id_furnizor
                ),
            ),

```

```

        t_echipament(
            :new.id_echipament,
            :new.nume,
            :new.data_instalare,
            :new.data_revizie,
            :new.id_filiala,
            :new.id_furnizor
        )
    );
ELSIF deleting THEN
    g_count_op := g_count_op + 1;
    g_changes(g_changes.count + 1) := t_row_change(
        'DELETE',
        :old.id_echipament,
        t_echipament(
            :old.id_echipament,
            :old.nume,
            :old.data_instalare,
            :old.data_revizie,
            :old.id_filiala,
            :old.id_furnizor
        ),
        NULL
    );
END IF;
END BEFORE EACH ROW;
AFTER STATEMENT IS BEGIN
    FOR i IN 1..g_changes.count LOOP
        INSERT INTO audit_echipament (
            operation_type,
            performed_by,
            id_echipament,
            old_values,
            new_values,
            summary_message
        ) VALUES ( g_changes(i).operation_type,
                    user,
                    g_changes(i).id_echipament,
                    g_changes(i).old_values,
                    g_changes(i).new_values,
                    upper(g_changes(i).operation_type)
                    || ' with another '
                    || g_count_op );
    END LOOP;
END AFTER STATEMENT;
END;
/

SELECT * FROM audit_echipament;

SELECT a.old_values.data_revizie, a.new_values.data_revizie FROM
audit_echipament a;

```



```
GRANT SELECT,UPDATE ON audit_echipament TO bro_manager_filiala1;

SELECT COUNT(*)
FROM audit_echipament a
WHERE a.old_values.id_filiala!=1 AND a.new_values.id_filiala!=1;
```

### 10.1.3. bro\_admin\_create\_tables.sql

```
-- Crearea tabelelor si inserarea datelor initiale in schema bro_admin
```

```
SET SERVEROUTPUT ON;
```

```
CREATE TABLE persoana(
    id_persoana NUMBER(*,0) CONSTRAINT pk_persoana PRIMARY KEY,
    nume VARCHAR2(20) CONSTRAINT nn_persoana_nume NOT NULL,
    prenume VARCHAR2(30) CONSTRAINT nn_persoana_prenume NOT NULL,
    email VARCHAR2(30) CONSTRAINT nn_u_persoana_email NOT NULL UNIQUE,
    varsta NUMBER(3,0) CONSTRAINT nn_persoana_varsta NOT NULL
);

CREATE TABLE telefon(
    tip VARCHAR2(20) CONSTRAINT nn_telefon_tip NOT NULL ,
    numar VARCHAR2(20) CONSTRAINT nn_telefon_numar NOT NULL,
    id_persoana NUMBER(*,0) CONSTRAINT fk_telefon_persoana REFERENCES
persoana(id_persoana) ON DELETE CASCADE,
    CONSTRAINT pk_telefon PRIMARY KEY (id_persoana, numar)
);

CREATE TABLE client(
    id_client NUMBER(*,0) CONSTRAINT pk_client PRIMARY KEY ,
    student VARCHAR2(1) DEFAULT 'N' CONSTRAINT ck_client_student CHECK (student
IN('Y','N')) ,
    CONSTRAINT fk_client_persoana FOREIGN KEY (id_client)
REFERENCES persoana(id_persoana) ON DELETE CASCADE
);

CREATE TABLE adresa(
    id_adresa NUMBER(*,0) CONSTRAINT pk_adresa PRIMARY KEY,
    strada VARCHAR2(40) CONSTRAINT nn_adresa_strada NOT NULL,
    oras VARCHAR2(20) CONSTRAINT nn_adresa_oras NOT NULL,
    judet VARCHAR2(20) CONSTRAINT nn_adresa_judet NOT NULL,
    cod_postal NUMBER(10,0) CONSTRAINT nn_adresa_cod_postal NOT NULL,
    numar NUMBER(4,0) CONSTRAINT nn_adresa_numar NOT NULL
);

CREATE TABLE filiala (
    id_filiala NUMBER(*,0) CONSTRAINT pk_filiala PRIMARY KEY,
    nume VARCHAR2(40) CONSTRAINT nn_filiala_nume NOT NULL,
    data_deschidere DATE CONSTRAINT nn_filiala_data_deschidere NOT NULL,
    id_adresa NUMBER(*,0) CONSTRAINT fk_filiala_adresa REFERENCES
adresa(id_adresa) NOT NULL UNIQUE
);
```

```

CREATE TABLE angajat(
    id_angajat NUMBER(*,0) CONSTRAINT pk_angajat PRIMARY KEY,
    data_angajare DATE CONSTRAINT nn_angajat_data_angjare NOT NULL,
    salariu NUMBER(20,2) CONSTRAINT ck_angajat_salariu CHECK (salariu > 0) NOT
NULL,
    id_filiala NUMBER(*,0) CONSTRAINT fk_angajat_filiala REFERENCES
filiala(id_filiala) ON DELETE CASCADE NOT NULL ,
    id_meneger NUMBER(*,0) CONSTRAINT fk_angajat_angajat REFERENCES
angajat(id_angajat),
    CONSTRAINT fk_angajat_persoana FOREIGN KEY (id_angajat)
REFERENCES persoana(id_persoana) ON DELETE CASCADE
);

CREATE TABLE receptionist(
    id_receptionist NUMBER(*,0) CONSTRAINT pk_receptionist PRIMARY KEY,
    program_complet VARCHAR2(1) CONSTRAINT ck_receptionist_program_complet
CHECK(program_complet IN ('Y','N')),
    CONSTRAINT fk_receptionist_angajat FOREIGN KEY (id_receptionist)
REFERENCES angajat(id_angajat) ON DELETE CASCADE
);

CREATE TABLE antrenor(
    id_antrenor NUMBER(*,0) CONSTRAINT pk_antrenor PRIMARY KEY,
    studii VARCHAR2(40) CONSTRAINT nn_antrenor_studii NOT NULL,
    CONSTRAINT fk_antrenor_angajat FOREIGN KEY (id_antrenor)
REFERENCES angajat(id_angajat) ON DELETE CASCADE
);

CREATE TABLE furnizor(
    id_furnizor NUMBER(*,0) CONSTRAINT pk_furnizor PRIMARY KEY,
    nume VARCHAR2(40) CONSTRAINT nn_furnizor_nume NOT NULL,
    cod_fiscal NUMBER(10,0) CONSTRAINT ck_furnizor_cod_fiscal NOT NULL UNIQUE,
    id_adresa NUMBER(*,0) CONSTRAINT fk_furnizor_adresa REFERENCES
adresa(id_adresa) NOT NULL UNIQUE
);

CREATE TABLE echipament(
    id_echipament NUMBER(*,0) CONSTRAINT pk_echipament PRIMARY KEY,
    nume VARCHAR2(40) CONSTRAINT nn_echipament_nume NOT NULL,
    data_instalare DATE CONSTRAINT nn_echipament_data_instalare NOT NULL,
    data_revizie DATE CONSTRAINT nn_echipament_data_revizie NOT NULL ,
    id_filiala NUMBER(*,0) CONSTRAINT fk_echipament_filiala REFERENCES
filiala(id_filiala) NOT NULL,
    id_furnizor NUMBER(*,0) CONSTRAINT fk_echipament_furnizor REFERENCES
furnizor(id_furnizor) NOT NULL,
    CONSTRAINT ck_echipament_instalare_revizie CHECK(data_instalare <=
data_revizie)
);

CREATE TABLE tip_abonament (
    nume_tip VARCHAR2(40) CONSTRAINT pk_tip_abonament PRIMARY KEY CHECK
(nume_tip IN ( 'lunar', 'trimestrial', 'bianual' , 'anual', 'extins')),
    pret NUMBER(8,2) CONSTRAINT ck_tip_abonament_pret NOT NULL UNIQUE

```

```

);

CREATE TABLE abonament(
    id_abonament NUMBER(*,0) CONSTRAINT pk_abonament PRIMARY KEY,
    nume_tip VARCHAR2(40) CONSTRAINT fk_abonament_tip_abonament REFERENCES
tip_abonament(nume_tip) NOT NULL,
    id_client NUMBER(*,0) CONSTRAINT fk_abonament_client REFERENCES
client(id_client) NOT NULL UNIQUE,
    data_inregistrare DATE CONSTRAINT nn_abonament_data_intregistrare NOT NULL
);

CREATE TABLE comanda(
    id_comanda NUMBER(*,0) CONSTRAINT pk_comanda PRIMARY KEY,
    id_receptionist NUMBER(*,0) CONSTRAINT fk_comanda_receptionist REFERENCES
receptionist(id_receptionist) NOT NULL,
    id_client NUMBER(*,0) CONSTRAINT fk_comanda_client REFERENCES
client(id_client) NOT NULL,
    data_comandare DATE CONSTRAINT nn_comanda_data_comandare NOT NULL,
    observatii VARCHAR2(255)
);

CREATE TABLE supliment (
    id_supliment NUMBER(*,0) CONSTRAINT pk_supliment PRIMARY KEY,
    nume VARCHAR2(50) CONSTRAINT nn_supliment_nume NOT NULL,
    descriere VARCHAR2(255),
    calorii NUMBER(10,4)CONSTRAINT nn_suplimen_calorii NOT NULL,
    pret NUMBER(10,4) CONSTRAINT nn_supliment_pret NOT NULL
);

CREATE TABLE aprovizionare (
    id_furnizor NUMBER(*,0),
    id_supliment NUMBER(*,0),
    cantitate NUMBER(4) CONSTRAINT ck_aprovizionare_cantitate CHECK(cantitate >
0) NOT NULL,
    CONSTRAINT pk_aprovizionare PRIMARY KEY (id_furnizor,id_supliment),
    CONSTRAINT fk_aprovizionare_furnizor FOREIGN KEY (id_furnizor) REFERENCES
furnizor(id_furnizor),
    CONSTRAINT fk_aprovizionare_supliment FOREIGN KEY (id_supliment) REFERENCES
supliment(id_supliment)
);

CREATE TABLE informatii_comanda (
    id_comanda NUMBER(*,0),
    id_supliment NUMBER(*,0),
    cantitate NUMBER(4) CONSTRAINT ck_ic_cantitate CHECK(cantitate > 0) NOT
NULL,
    CONSTRAINT pk_informatii_comanda PRIMARY KEY (id_comanda,id_supliment),
    CONSTRAINT fk_ic_comanda FOREIGN KEY (id_comanda) REFERENCES
comanda(id_comanda),
    CONSTRAINT fk_ic_supliment FOREIGN KEY (id_supliment) REFERENCES
supliment(id_supliment)
);

```

```

CREATE TABLE account_mapping(
    id_persoana NUMBER(*,0) CONSTRAINT pk_account_mapping PRIMARY KEY,
    username VARCHAR2(128) UNIQUE
);

COMMIT;

CREATE TABLE logger(
    id_logger NUMBER(*,0) CONSTRAINT pk_logger PRIMARY KEY,
    message VARCHAR2(255),
    message_type VARCHAR2(1) CONSTRAINT ck_logger_message_type CHECK
(message_type IN('E','W','I')),
    created_by VARCHAR2(40) CONSTRAINT nn_logger_created_by NOT NULL,
    created_at TIMESTAMP CONSTRAINT nn_logger_created_at NOT NULL
);

CREATE OR REPLACE PACKAGE logger_utils IS
    PROCEDURE logger_entry(mesaj VARCHAR2,tip_mesaj VARCHAR2, cod NUMBER);
    PROCEDURE logger_entry(mesaj VARCHAR2,tip_mesaj VARCHAR2);
END logger_utils;
/

-- PRAGMA AUTONOMOUS_TRANSACTION este necesara, deoarece functia
-- RAISE_APPLICATION_ERROR opreste tranzactia originala, ceea ce impiedica
-- inserarea in Logger. In acest caz folosirea acesteia nu conduce
-- la probleme pentru ca nu folosim date
-- din noua tranzactie in cea originala.
CREATE OR REPLACE PACKAGE BODY logger_utils IS
    PROCEDURE logger_entry(mesaj VARCHAR2,tip_mesaj VARCHAR2, cod NUMBER) IS
        PRAGMA autonomous_transaction;
    BEGIN
        INSERT INTO logger(message, message_type,created_by, created_at)
        VALUES (substr(mesaj,1,255), tip_mesaj,user,
TO_DATE(to_char(sysdate, 'DD-MON-YYYY HH24:MI:SS'), 'DD-MON-YYYY HH24:MI:SS'));
        COMMIT;
        raise_application_error(cod,mesaj);
        dbms_output.put_line(cod || ' : ' ||mesaj);
        EXCEPTION
        WHEN OTHERS THEN
            ROLLBACK;
            raise_application_error(sqlcode,sqlerrm);
    END logger_entry;
    PROCEDURE logger_entry(mesaj VARCHAR2,tip_mesaj VARCHAR2) IS
        PRAGMA autonomous_transaction;
    BEGIN
        dbms_output.put_line(tip_mesaj || ' : ' ||mesaj);
        INSERT INTO logger(message, message_type,created_by,
created_at) VALUES
        (substr(mesaj,1,255), tip_mesaj,user, TO_DATE(to_char(sysdate, 'DD-
MON-YYYY HH24:MI:SS'), 'DD-MON-YYYY HH24:MI:SS'));
        COMMIT;
    END;

```

```

END logger_utils;
/

CREATE OR REPLACE PACKAGE sequence_utils IS
    PROCEDURE create_sequence(p_seq_name IN VARCHAR2);
    PROCEDURE create_sequence_trigger (p_tbl_name IN VARCHAR2);
END sequence_utils;
/

CREATE OR REPLACE PACKAGE BODY sequence_utils IS

    PROCEDURE create_sequence(p_seq_name IN VARCHAR2) IS
        seq_count INT;
        seq_name VARCHAR2(128);
    BEGIN
        --          dbms_output.put_line(p_seq_name);
        seq_name:=dbms_assert.simple_sql_name(p_seq_name);
        --          dbms_output.put_line(seq_name);
        SELECT COUNT(*) INTO seq_count FROM user_sequences WHERE
sequence_name = upper(seq_name);
        IF seq_count > 0 THEN
            EXECUTE IMMEDIATE 'DROP SEQUENCE ' || seq_name;
        END IF;
        EXECUTE IMMEDIATE 'CREATE SEQUENCE ' || seq_name || ' START WITH 1
INCREMENT BY 1';
    EXCEPTION
        WHEN OTHERS THEN
            logger_utils.logger_entry(sqlerrm,'E',sqlcode);
    END create_sequence;

    PROCEDURE create_sequence_trigger (p_tbl_name IN VARCHAR2) IS
        count_tables NUMBER;
        v_id_count INT;
        no_id EXCEPTION;
        table_not_found EXCEPTION;
        tbl_name VARCHAR2(128);
    BEGIN
        dbms_output.put_line(p_tbl_name);
        tbl_name:=dbms_assert.simple_sql_name(p_tbl_name);
        dbms_output.put_line(tbl_name);
        SELECT COUNT(*)
        INTO count_tables
        FROM all_tables
        WHERE table_name = upper(tbl_name);

        IF count_tables = 0 THEN
            RAISE table_not_found;
        END IF;

        EXECUTE IMMEDIATE
            'SELECT COUNT(*) FROM all_tab_columns WHERE upper(table_name) =
upper('' || tbl_name ||

```

```

        ''') AND upper(column_name) = upper('id_' || tbl_name || ''')'
INTO v_id_count;

    IF v_id_count=0 THEN
        RAISE no_id;
    END IF;
    create_sequence(tbl_name || '_seq');

    EXECUTE IMMEDIATE 'CREATE OR REPLACE TRIGGER ' || tbl_name ||
        '_trigger BEFORE INSERT ON ' || tbl_name ||
        ' FOR EACH ROW BEGIN SELECT '
||tbl_name||'_seq.NEXTVAL INTO :NEW.id_'||
        lower(tbl_name)||' FROM dual; END;';

    EXCEPTION
        WHEN no_id THEN
            logger_utils.logger_entry('Column named id_'|| tbl_name || '
does not exist in ' ||tbl_name,'E',-20006);
        WHEN table_not_found THEN
            logger_utils.logger_entry('Table '|| tbl_name || ' does not
exist.','E',-20007);
        WHEN OTHERS THEN
            logger_utils.logger_entry( sqlerrm || ' code: ' ||
sqlcode,'E',-20010);
    END create_sequence_trigger;

END sequence_utils;
/
TRUNCATE TABLE logger;
EXEC sequence_utils.create_sequence_trigger('Logger');

CREATE OR REPLACE PROCEDURE insert_into_account_mapping(
    id_persoana NUMBER,acc_suff VARCHAR2
) IS
    v_user VARCHAR2(128);
BEGIN
    SELECT column_value INTO v_user FROM (
        SELECT column_value,TO_NUMBER(regexp_substr(column_value, '[0-9]+$')) AS
numeric_part
        FROM TABLE(sys.get_users_by_suffix(acc_suff))
        WHERE NOT EXISTS(SELECT 1 FROM account_mapping WHERE
username=upper(column_value))
        ORDER BY numeric_part ASC)
    WHERE ROWNUM =1 ;
    dbms_output.put_line('user ' ||v_user);
    INSERT INTO account_mapping VALUES (id_persoana,upper(v_user));
    EXCEPTION
        WHEN OTHERS THEN
            logger_utils.logger_entry( 'No available account for the suffix '||
acc_suff,'E',-20020);
END;
/
CREATE OR REPLACE PACKAGE global_constants IS
    persoana_seq CONSTANT VARCHAR2(20) := 'PERSOANA_SEQ_GLOBAL';

```

```

END global_constants;
/
COMMIT;

-- multiple vizualizari si triggere de tipul instead of pentru a usura
inserarea
CREATE OR REPLACE VIEW client_extins AS(
SELECT c.id_client, p.nume, p.prenume,p.email,p.varsta, c.student
FROM persoana p JOIN client c ON c.id_client = p.id_persoana
);

CREATE OR REPLACE TRIGGER client_extins_insert INSTEAD OF INSERT ON
client_extins
FOR EACH ROW
DECLARE
    seq_count NUMBER;
    seq_not_found EXCEPTION;
    id_nr persoana.id_persoana%TYPE;
BEGIN
    SELECT COUNT(*)
    INTO seq_count
    FROM user_sequences
    WHERE sequence_name = global_constants.persoana_seq;
    IF seq_count = 0 THEN
        RAISE seq_not_found;
    END IF;
    EXECUTE IMMEDIATE 'SELECT ' || global_constants.persoana_seq ||
'.NEXTVAL FROM dual' INTO id_nr;
    INSERT INTO persoana(id_persoana,nume,prenume,email,varsta) VALUES
(id_nr, :new.nume, :new.prenume, :new.email, :new.varsta);
    INSERT INTO client VALUES
(id_nr,:new.student);
    insert_into_account_mapping(id_nr,'CLIENT');
EXCEPTION
    WHEN seq_not_found THEN
        logger_utils.logger_entry('Secventa pentru persoana nu
exista.','E',-20005);
    WHEN OTHERS THEN
        logger_utils.logger_entry( sqlerrm || ' code: ' ||
sqlcode,'E',-20010);
END;
/

CREATE OR REPLACE VIEW angajat_extins AS (
    SELECT a.id_angajat,p.nume, p.prenume,p.email,p.varsta,
        a.data_angajare, a.salariu, a.id_filiala, a.id_meneger
    FROM persoana p JOIN angajat a ON p.id_persoana = a.id_angajat
);

CREATE OR REPLACE TRIGGER angajat_extins_insert INSTEAD OF INSERT ON
angajat_extins
FOR EACH ROW
BEGIN

```



```

        INSERT INTO persoana(id_persoana,nume,prenume,email,varsta) VALUES
        (:new.id_angajat, :new.num, :new.prenume, :new.email, :new.varsta);
        INSERT INTO angajat(id_angajat, data_angajare, salariu, id_filiala,
id_meneger) VALUES
        (:new.id_angajat,:new.data_angajare, :new.salariu, :new.id_filiala,
        :new.id_meneger);
    EXCEPTION
    WHEN OTHERS THEN
        logger_utils.logger_entry( sqlerrm || ' code: ' ||
sqlcode,'E',-20010);
    END;
/

CREATE OR REPLACE VIEW antrenor_extins AS (
    SELECT ant.id_antrenor,a.num, a.prenume,a.email,a.varsta,
        a.data_angajare, a.salariu, a.id_filiala, a.id_meneger,ant.studii
    FROM antrenor ant JOIN angajat_extins a ON ant.id_antrenor = a.id_angajat
);

CREATE OR REPLACE TRIGGER antrenor_extins_insert INSTEAD OF INSERT ON
antrenor_extins
FOR EACH ROW
DECLARE
    seq_count NUMBER;
    seq_not_found EXCEPTION;
    id_nr persoana.id_persoana%TYPE;
    id_men persoana.id_persoana%TYPE := NULL;
BEGIN
    SELECT COUNT(*)
    INTO seq_count
    FROM user_sequences
    WHERE sequence_name = global_constants.persoana_seq;
    IF seq_count = 0 THEN
        RAISE seq_not_found;
    END IF;

    EXECUTE IMMEDIATE 'SELECT ' || global_constants.persoana_seq ||
'.NEXTVAL FROM dual' INTO id_nr;

    IF :new.id_meneger IS NOT NULL THEN
        id_men:=:new.id_meneger;
    END IF;

    INSERT INTO angajat_extins(id_angajat,nume, prenume,email,varsta,
        data_angajare, salariu, id_filiala, id_meneger) VALUES
        (id_nr, :new.num, :new.prenume,:new.email,:new.varsta,
        :new.data_angajare, :new.salariu,:new.id_filiala, id_men);
    INSERT INTO antrenor VALUES
        (id_nr,:new.studii);
    insert_into_account_mapping(id_nr,'ANTRENOR');
    EXCEPTION
    WHEN seq_not_found THEN

```

```

        logger_utils.logger_entry('Secventa pentru persoana nu
exista.','E',-20005);
    WHEN OTHERS THEN
        dbms_output.put_line(sqlerrm);
        logger_utils.logger_entry( sqlerrm || ' code: ' ||
sqlcode,'E',-20010);
    END;
/

CREATE OR REPLACE VIEW receptionist_extins AS (
    SELECT r.id_receptionist,a.nume, a.prenume,a.email,a.varsta,
        a.data_angajare, a.salariu, a.id_filiala,
a.id_meneger,r.program_complet
    FROM receptionist r JOIN angajat_extins a ON r.id_receptionist =
a.id_angajat
);
/
CREATE OR REPLACE TRIGGER receptionist_extins_insert INSTEAD OF INSERT ON
receptionist_extins
    FOR EACH ROW
    DECLARE
        seq_count NUMBER;
        seq_not_found EXCEPTION;
        id_nr persoana.id_persoana%TYPE;
        id_men persoana.id_persoana%TYPE := NULL;
    BEGIN
        SELECT COUNT(*)
        INTO seq_count
        FROM user_sequences
        WHERE sequence_name = global_constants.persoana_seq;
        IF seq_count = 0 THEN
            RAISE seq_not_found;
        END IF;

        EXECUTE IMMEDIATE 'SELECT ' || global_constants.persoana_seq ||
'.NEXTVAL FROM dual' INTO id_nr;

        IF :new.id_meneger IS NOT NULL THEN
            id_men:=:new.id_meneger;
        END IF;

        INSERT INTO angajat_extins(id_angajat,nume, prenume,email,varsta,
            data_angajare, salariu, id_filiala, id_meneger) VALUES
(id_nr, :new.nume, :new.prenume,:new.email,:new.varsta,
            :new.data_angajare, :new.salariu,:new.id_filiala, id_men);
        INSERT INTO receptionist(id_receptionist,program_complet) VALUES
(id_nr,:new.program_complet);
        insert_into_account_mapping(id_nr,'RECEPTIONIST');
    EXCEPTION
        WHEN seq_not_found THEN
            logger_utils.logger_entry('Secventa pentru persoana nu
exista.','E',-20005);
    WHEN OTHERS THEN

```

```

        logger_utils.logger_entry( sqlerrm || ' code: ' ||
sqlcode,'E',-20010);
    END;

/

EXEC sequence_utils.create_sequence_trigger('Adresa');

INSERT INTO adresa (
    strada,
    numar,
    oras,
    judet,
    cod_postal
) VALUES ( 'Bd. Lujerului',
    33,
    'Bucuresti',
    'Bucuresti',
    '405985' );
INSERT INTO adresa (
    strada,
    numar,
    oras,
    judet,
    cod_postal
) VALUES ( 'Bd. Tineretului',
    21,
    'Bucuresti',
    'Bucuresti',
    '582155' );
INSERT INTO adresa (
    strada,
    numar,
    oras,
    judet,
    cod_postal
) VALUES ( 'Bd. Bucuresti',
    11,
    'Brasov',
    'Brasov',
    '123456' );
INSERT INTO adresa (
    strada,
    numar,
    oras,
    judet,
    cod_postal
) VALUES ( 'Bd. Republicii',
    3,
    'Ploiesti',
    'Prahova',
    55231 );
INSERT INTO adresa (

```

```

        strada,
        numar,
        oras,
        judet,
        cod_postal
) VALUES ( 'Str Parangului',
            100,
            'Craiova',
            'Dolj',
            7742101 );
INSERT INTO adresa (
        strada,
        numar,
        oras,
        judet,
        cod_postal
) VALUES ( 'Matei Basarab',
            18,
            'Bucuresti',
            'Bucuresti',
            665842 );
INSERT INTO adresa (
        strada,
        numar,
        oras,
        judet,
        cod_postal
) VALUES ( 'Unirii',
            33,
            'Bucuresti',
            'Bucuresti',
            868605 );
INSERT INTO adresa (
        strada,
        numar,
        oras,
        judet,
        cod_postal
) VALUES ( 'Mihai Bravu',
            22,
            'Bucuresti',
            'Bucuresti',
            78592 );
INSERT INTO adresa (
        strada,
        numar,
        oras,
        judet,
        cod_postal
) VALUES ( 'Frigului',
            77,
            'Brasov',

```

```

        'Brasov',
        888801 );
INSERT INTO adresa (
    strada,
    numar,
    oras,
    judet,
    cod_postal
) VALUES ( 'Calea Traian',
    99,
    'Craiova',
    'Dolj',
    224402 );
INSERT INTO adresa (
    strada,
    numar,
    oras,
    judet,
    cod_postal
) VALUES ( 'Calea Serban Voda',
    232,
    'Bucuresti',
    'Bucuresti',
    40578 );
INSERT INTO adresa (
    strada,
    numar,
    oras,
    judet,
    cod_postal
) VALUES ( 'Viilor',
    12,
    'Bucuresti',
    'Bucuresti',
    232454 );
INSERT INTO adresa (
    strada,
    numar,
    oras,
    judet,
    cod_postal
) VALUES ( 'Alea Tomis',
    36,
    'Arad',
    'Arad',
    111454 );
INSERT INTO adresa (
    strada,
    numar,
    oras,
    judet,
    cod_postal

```

```

) VALUES ( 'Anastasie Panu',
            56,
            'Iasi',
            'Iasi',
            999454 );
INSERT INTO adresa (
    strada,
    numar,
    oras,
    judet,
    cod_postal
) VALUES ( 'Aleea Tomis',
            1,
            'Dej',
            'Cluj',
            123454 );
INSERT INTO adresa (
    strada,
    numar,
    oras,
    judet,
    cod_postal
) VALUES ( 'Tiberiu Popoviciu ',
            22,
            'Cluj',
            'Cluj',
            538454 );

EXEC sequence_utils.create_sequence_trigger('Filiala');

INSERT INTO filiala (
    nume,
    data_deschidere,
    id_adresa
) VALUES ( 'Lujerului',
            TO_DATE('21-JAN-2014','DD-MON-YYYY'),
            1 );
INSERT INTO filiala (
    nume,
    data_deschidere,
    id_adresa
) VALUES ( 'Tineretului',
            TO_DATE('21-FEB-2000','DD-MON-YYYY'),
            2 );
INSERT INTO filiala (
    nume,
    data_deschidere,
    id_adresa
) VALUES ( 'Brasov',
            TO_DATE('14-FEB-2010','DD-MON-YYYY'),
            3 );
INSERT INTO filiala (
    nume,

```

```

        data_deschidere,
        id_adresa
    ) VALUES ( 'Ploiesti',
                TO_DATE('11-DEC-1999','DD-MON-YYYY'),
                4 );
INSERT INTO filiala (
    nume,
    data_deschidere,
    id_adresa
) VALUES ( 'Craiova',
            TO_DATE('01-NOV-1995','DD-MON-YYYY'),
            5 );
INSERT INTO filiala (
    nume,
    data_deschidere,
    id_adresa
) VALUES ( 'Filiala Sector 4',
            TO_DATE('01-FEB-1999','DD-MON-YYYY'),
            11 );
INSERT INTO filiala (
    nume,
    data_deschidere,
    id_adresa
) VALUES ( 'Filiala Sector 3',
            TO_DATE('15-MAR-2005','DD-MON-YYYY'),
            6 );
INSERT INTO filiala (
    nume,
    data_deschidere,
    id_adresa
) VALUES ( 'Sediul Unirii',
            TO_DATE('01-MAY-2000','DD-MON-YYYY'),
            7 );
INSERT INTO filiala (
    nume,
    data_deschidere,
    id_adresa
) VALUES ( 'Filiala Viilor',
            TO_DATE('01-APR-2012','DD-MON-YYYY'),
            12 );

EXEC sequence_utils.create_sequence(global_constants.persoana_seq);

SELECT *
FROM account_mapping;

INSERT INTO antrenor_extins (
    nume,
    prenume,
    email,
    varsta,
    data_angajare,
    salariu,

```



```

        id_filiala,
        studii,
        id_meneger
) VALUES ( 'Popescu',
            'Ion',
            'popescuI@yahoo.com',
            30,
            TO_DATE('11-JAN-2020','DD-MON-YYYY'),
            1500,
            1,
            'Liceul Sportiv 1 Bucuresti',
            NULL );
INSERT INTO antrenor_extins (
    nume,
    prenume,
    email,
    varsta,
    data_angajare,
    salariu,
    id_filiala,
    studii,
    id_meneger
) VALUES ( 'Popescu',
            'George',
            'popescuG@yahoo.com',
            31,
            TO_DATE('01-FEB-2015','DD-MON-YYYY'),
            2100,
            1,
            'Liceul Sportiv Breaza',
            NULL );
INSERT INTO antrenor_extins (
    nume,
    prenume,
    email,
    varsta,
    data_angajare,
    salariu,
    id_filiala,
    studii,
    id_meneger
) VALUES ( 'Ionescu',
            'Andrei',
            'ionescuA@yahoo.com',
            21,
            TO_DATE('20-MAR-2017','DD-MON-YYYY'),
            2200,
            1,
            'Facultate Kinetoterapie',
            NULL );
INSERT INTO antrenor_extins (
    nume,

```

```

    prenume,
    email,
    varsta,
    data_angajare,
    salariu,
    id_filiala,
    studii,
    id_meneger
) VALUES ( 'Ionescu',
            'Ion',
            'ionescuI@yahoo.com',
            21,
            TO_DATE('01-FEB-2015', 'DD-MON-YYYY'),
            2000,
            1,
            'IEFS',
            NULL );

INSERT INTO antrenor_extins (
    nume,
    prenume,
    email,
    varsta,
    data_angajare,
    salariu,
    id_filiala,
    studii,
    id_meneger
) VALUES ( 'Mihai',
            'Marcel',
            'mihaimarcel@yahoo.com',
            22,
            TO_DATE('01-MAY-2021', 'DD-MON-YYYY'),
            2600,
            1,
            'Facultate Kinetoterapie',
            NULL );

INSERT INTO antrenor_extins (
    nume,
    prenume,
    email,
    varsta,
    data_angajare,
    salariu,
    id_filiala,
    studii,
    id_meneger
) VALUES ( 'Aioanei',
            'Andrei',
            'aioaneiandrei@yahoo.com',
            30,
            TO_DATE('01-JAN-2010', 'DD-MON-YYYY'),
            5000,

```

```

        1,
        'IEFS',
        NULL );
INSERT INTO antrenor_extins (
    nume,
    prenume,
    email,
    varsta,
    data_angajare,
    salariu,
    id_filiala,
    studii,
    id_meneger
) VALUES ( 'Stancioiu',
            'Razvan',
            'stancioiurazvan@yahoo.com',
            28,
            TO_DATE('15-APR-2005', 'DD-MON-YYYY'),
            3500,
            1,
            'Curs FRCF',
            NULL );

SELECT *
FROM antrenor_extins;

```

```

INSERT INTO receptionist_extins (
    nume,
    prenume,
    email,
    varsta,
    data_angajare,
    salariu,
    id_filiala,
    program_complet,
    id_meneger
) VALUES ( 'Dinca',
            'Antoaneta',
            'dincaa@yahoo.com',
            22,
            TO_DATE('01-JUN-2001', 'DD-MON-YYYY'),
            2600,
            1,
            'Y',
            NULL );
INSERT INTO receptionist_extins (
    nume,
    prenume,
    email,
    varsta,
    data_angajare,
    salariu,

```

```

        id_filiala,
        program_complet,
        id_meneger
) VALUES ( 'Vasilescu',
            'Marcel',
            'vasilescum@yahoo.com',
            22,
            TO_DATE('01-JUL-2019','DD-MON-YYYY'),
            1300,
            1,
            'N',
            NULL );
INSERT INTO receptionist_extins (
    nume,
    prenume,
    email,
    varsta,
    data_angajare,
    salariu,
    id_filiala,
    program_complet,
    id_meneger
) VALUES ( 'Popescu',
            'George',
            'popescug@yahoo.com',
            22,
            TO_DATE('01-JAN-2018','DD-MON-YYYY'),
            2300,
            1,
            'Y',
            NULL );
INSERT INTO receptionist_extins (
    nume,
    prenume,
    email,
    varsta,
    data_angajare,
    salariu,
    id_filiala,
    program_complet,
    id_meneger
) VALUES ( 'Preda',
            'Marina',
            'predam@yahoo.com',
            27,
            TO_DATE('01-FEB-2017','DD-MON-YYYY'),
            2500,
            1,
            'Y',
            NULL );
INSERT INTO receptionist_extins (
    nume,

```

```

    prenume,
    email,
    varsta,
    data_angajare,
    salariu,
    id_filiala,
    program_complet,
    id_meneger
) VALUES ( 'Dumitrescu',
            'Anca',
            'dumitrescua@yahoo.com',
            22,
            TO_DATE('01-MAR-2015','DD-MON-YYYY'),
            2750,
            1,
            'Y',
            NULL );

INSERT INTO receptionist_extins (
    nume,
    prenume,
    email,
    varsta,
    data_angajare,
    salariu,
    id_filiala,
    program_complet,
    id_meneger
) VALUES ( 'Marinica',
            'Ion',
            'marinicaion@yahoo.com',
            60,
            TO_DATE('01-JUN-2016','DD-MON-YYYY'),
            1700,
            1,
            'N',
            NULL );

INSERT INTO receptionist_extins (
    nume,
    prenume,
    email,
    varsta,
    data_angajare,
    salariu,
    id_filiala,
    program_complet,
    id_meneger
) VALUES ( 'Dinca',
            'Ion',
            'dincaion@yahoo.com',
            45,
            TO_DATE('01-APR-2021','DD-MON-YYYY'),
            1700,

```

```

        9,
        'N',
        NULL );
INSERT INTO receptionist_extins (
    nume,
    prenume,
    email,
    varsta,
    data_angajare,
    salariu,
    id_filiala,
    program_complet,
    id_meneger
) VALUES ( 'Marinescu',
            'Ion',
            'marinescuion@yahoo.com',
            23,
            TO_DATE('01-JUN-2022','DD-MON-YYYY'),
            2900,
            9,
            'Y',
            NULL );
INSERT INTO receptionist_extins (
    nume,
    prenume,
    email,
    varsta,
    data_angajare,
    salariu,
    id_filiala,
    program_complet,
    id_meneger
) VALUES ( 'Ignat',
            'Ana',
            'ignatana@yahoo.com',
            20,
            TO_DATE('01-FEB-2023','DD-MON-YYYY'),
            2600,
            5,
            'Y',
            NULL );
INSERT INTO receptionist_extins (
    nume,
    prenume,
    email,
    varsta,
    data_angajare,
    salariu,
    id_filiala,
    program_complet,
    id_meneger
) VALUES ( 'Dancescu',

```

```

        'Sorin',
        'dancescusorin@yahoo.com',
        35,
        TO_DATE('01-FEB-2020','DD-MON-YYYY'),
        3000,
        4,
        'Y',
        NULL );

SELECT *
  FROM account_mapping;

UPDATE angajat
  SET
    id_meneger = NULL
 WHERE id_angajat = 7;
UPDATE angajat
  SET
    id_meneger = 7
 WHERE id_angajat != 7
    AND id_filiala = 1;
UPDATE angajat
  SET
    id_meneger = 14
 WHERE id_angajat = 15;
SELECT *
  FROM angajat;

INSERT INTO client_extins (
  nume,
  prenume,
  email,
  varsta,
  student
) VALUES ( 'Vasilescu',
            'Razvan',
            'vasilescurazvan@yahoo.com',
            21,
            'N' );

INSERT INTO client_extins (
  nume,
  prenume,
  email,
  varsta,
  student
) VALUES ( 'Ionescu',
            'Andrei',
            'ionescua@yahoo.com',
            19,
            'Y' );

INSERT INTO client_extins (
  nume,
  prenume,

```



```

        email,
        varsta,
        student
    ) VALUES ( 'Tanasescu',
                'Ion',
                'tanasescui@yahoo.com',
                19,
                'Y' );
INSERT INTO client_extins (
    nume,
    prenume,
    email,
    varsta,
    student
) VALUES ( 'Ionescu',
            'Vasile',
            'ionescuv@yahoo.com',
            32,
            'N' );
INSERT INTO client_extins (
    nume,
    prenume,
    email,
    varsta,
    student
) VALUES ( 'Tanasescu',
            'Anca',
            'tanasescua@yahoo.com',
            50,
            'N' );
INSERT INTO client_extins (
    nume,
    prenume,
    email,
    varsta,
    student
) VALUES ( 'Marinecu',
            'Vlad',
            'vladutz@yahoo.com',
            27,
            'N' );
INSERT INTO client_extins (
    nume,
    prenume,
    email,
    varsta,
    student
) VALUES ( 'Dobrescu',
            'Marcel',
            'dorescu_mar@yahoo.com',
            37,
            'N' );

```

```

INSERT INTO client_extins (
    nume,
    prenume,
    email,
    varsta,
    student
) VALUES ( 'Marinica',
            'Stefan',
            'marinicastefan@yahoo.com',
            35,
            'N' );

INSERT INTO client_extins (
    nume,
    prenume,
    email,
    varsta,
    student
) VALUES ( 'Marinica',
            'Bogdan',
            'marinicabogdan@yahoo.com',
            22,
            'Y' );

INSERT INTO client_extins (
    nume,
    prenume,
    email,
    varsta,
    student
) VALUES ( 'Stefanescu',
            'Ana',
            'stefanescuana@yahoo.com',
            19,
            'Y' );

SELECT *
FROM account_mapping;

EXEC sequence_utils.create_sequence_trigger('Furnizor');

INSERT INTO furnizor (
    nume,
    cod_fiscal,
    id_adresa
) VALUES ( 'MyProtein',
            '8859692',
            1 );

INSERT INTO furnizor (
    nume,
    cod_fiscal,
    id_adresa
) VALUES ( 'Gym Beam',
            '9859692',

```

```

        2 );
INSERT INTO furnizor (
    nume,
    cod_fiscal,
    id_adresa
) VALUES ( 'Redis',
            '7859692',
            3 );
INSERT INTO furnizor (
    nume,
    cod_fiscal,
    id_adresa
) VALUES ( 'Decathlon',
            '1859692',
            4 );
INSERT INTO furnizor (
    nume,
    cod_fiscal,
    id_adresa
) VALUES ( 'Vexio',
            '9959692',
            5 );
INSERT INTO furnizor (
    nume,
    cod_fiscal,
    id_adresa
) VALUES ( 'BEWIT',
            '9059692',
            13 );
INSERT INTO furnizor (
    nume,
    cod_fiscal,
    id_adresa
) VALUES ( 'BODY NEWLINE CONCEPT',
            '48393052',
            14 );
INSERT INTO furnizor (
    nume,
    cod_fiscal,
    id_adresa
) VALUES ( 'Pro Nutrition',
            '12420890',
            15 );
INSERT INTO furnizor (
    nume,
    cod_fiscal,
    id_adresa
) VALUES ( 'Arena Systems',
            '32120890',
            16 );

EXEC sequence_utils.create_sequence_trigger('Supliment');

```

```

INSERT INTO supliment (
    nume,
    descriere,
    calorii,
    pret
) VALUES ( 'Whey Protein',
            'Zer premium cu 21 g de proteine per portie.',
            '430',
            '100' );

INSERT INTO supliment (
    nume,
    descriere,
    calorii,
    pret
) VALUES ( 'Izolat proteic din soia',
            'O alegere excelenta pentru vegetarieni si vegani.',
            300,
            150 );

INSERT INTO supliment (
    nume,
    descriere,
    calorii,
    pret
) VALUES ( 'Vitafiber',
            'Derivat din amidon de porumb nemodificat genetic.',
            150,
            210 );

INSERT INTO supliment (
    nume,
    descriere,
    calorii,
    pret
) VALUES ( 'Unt de arahide',
            'Amestec pudra cu 70% mai putine grasimi.',
            300,
            90 );

INSERT INTO supliment (
    nume,
    descriere,
    calorii,
    pret
) VALUES ( 'Impact Diet Lean',
            'Amestec fibre sub forma de fructo-oligozaharide.',
            250,
            200 );

INSERT INTO supliment (
    nume,
    descriere,
    calorii,
    pret
) VALUES ( 'Muscle Mass - pachet premium',

```

```

        'Pachet complet: gainer de top + preworkout Complete Workout +
formula pe baza de creatina.',
        1000,
        334 );
INSERT INTO supliment (
    nume,
    descriere,
    calorii,
    pret
) VALUES ( 'X-plode plicuri',
        'Imbunatateste performanta fizica, regenerarea si volumizarea
celulelor musculare.',
        80,
        56 );
INSERT INTO supliment (
    nume,
    descriere,
    calorii,
    pret
) VALUES ( 'Essential Amino Acids',
        'Con?ine un mix de 8 aminoacizi esen?iali.',
        30,
        54 );
INSERT INTO supliment (
    nume,
    descriere,
    calorii,
    pret
) VALUES ( 'Jeleuri cu arom? de otet de cidru de mere',
        'Ajuta la protejarea celulelor impotriva stresului oxidativ.',
        10,
        79 );
INSERT INTO supliment (
    nume,
    descriere,
    calorii,
    pret
) VALUES ( 'Jeleuri pre-antrenament',
        'Un mod simplu de a va pregati mintal si fizic pentru fiecare
antrenament.',
        15,
        129 );

INSERT INTO aprovizionare (
    id_furnizor,
    id_supliment,
    cantitate
) VALUES ( 1,
        1,
        10 );
INSERT INTO aprovizionare (
    id_furnizor,
    id_supliment,

```

```

        cantitate
    ) VALUES ( 3,
                2,
                10 );
INSERT INTO aprovizionare (
    id_furnizor,
    id_supliment,
    cantitate
) VALUES ( 1,
            3,
            20 );
INSERT INTO aprovizionare (
    id_furnizor,
    id_supliment,
    cantitate
) VALUES ( 3,
            4,
            50 );
INSERT INTO aprovizionare (
    id_furnizor,
    id_supliment,
    cantitate
) VALUES ( 1,
            5,
            15 );
INSERT INTO aprovizionare (
    id_furnizor,
    id_supliment,
    cantitate
) VALUES ( 2,
            1,
            10 );
INSERT INTO aprovizionare (
    id_furnizor,
    id_supliment,
    cantitate
) VALUES ( 2,
            5,
            20 );
INSERT INTO aprovizionare (
    id_furnizor,
    id_supliment,
    cantitate
) VALUES ( 2,
            3,
            90 );
INSERT INTO aprovizionare (
    id_furnizor,
    id_supliment,
    cantitate
) VALUES ( 3,
            3,

```

```

        70 );
INSERT INTO aprovizionare (
    id_furnizor,
    id_supliment,
    cantitate
) VALUES ( 8,
            6,
            5 );
INSERT INTO aprovizionare (
    id_furnizor,
    id_supliment,
    cantitate
) VALUES ( 8,
            7,
            15 );
INSERT INTO aprovizionare (
    id_furnizor,
    id_supliment,
    cantitate
) VALUES ( 8,
            8,
            20 );
INSERT INTO aprovizionare (
    id_furnizor,
    id_supliment,
    cantitate
) VALUES ( 8,
            9,
            15 );
INSERT INTO aprovizionare (
    id_furnizor,
    id_supliment,
    cantitate
) VALUES ( 8,
            10,
            20 );

EXEC sequence_utils.create_sequence_trigger('Echipament');

INSERT INTO echipament (
    nume,
    data_instalare,
    data_reviziei,
    id_filiala,
    id_furnizor
) VALUES ( 'Leg Press',
            TO_DATE('20-MAY-2020','DD-MON-YYYY'),
            TO_DATE('20-MAY-2021','DD-MON-YYYY'),
            1,
            5 );
INSERT INTO echipament (
    nume,
    data_instalare,

```

```

    data_revizie,
    id_filiala,
    id_furnizor
) VALUES ( 'Chest Press',
            TO_DATE('20-JUN-2021', 'DD-MON-YYYY'),
            TO_DATE('20-JUN-2021', 'DD-MON-YYYY'),
            1,
            5 );
INSERT INTO echipament (
    nume,
    data_instalare,
    data_revizie,
    id_filiala,
    id_furnizor
) VALUES ( 'Peck Deck',
            TO_DATE('01-JAN-2019', 'DD-MON-YYYY'),
            TO_DATE('01-JAN-2021', 'DD-MON-YYYY'),
            2,
            4 );
INSERT INTO echipament (
    nume,
    data_instalare,
    data_revizie,
    id_filiala,
    id_furnizor
) VALUES ( 'Preacher Curl',
            TO_DATE('28-APR-2020', 'DD-MON-YYYY'),
            TO_DATE('28-APR-2021', 'DD-MON-YYYY'),
            3,
            3 );
INSERT INTO echipament (
    nume,
    data_instalare,
    data_revizie,
    id_filiala,
    id_furnizor
) VALUES ( 'Calves Raises',
            TO_DATE('20-APR-2021', 'DD-MON-YYYY'),
            TO_DATE('20-APR-2022', 'DD-MON-YYYY'),
            4,
            3 );
INSERT INTO echipament (
    nume,
    data_instalare,
    data_revizie,
    id_filiala,
    id_furnizor
) VALUES ( 'Lateral Raises',
            TO_DATE('10-APR-2021', 'DD-MON-YYYY'),
            TO_DATE('10-APR-2022', 'DD-MON-YYYY'),
            4,
            5 );

```



```

INSERT INTO echipament (
    nume,
    data_instalare,
    data_revizie,
    id_filiala,
    id_furnizor
) VALUES ( 'Frontal Raises',
    TO_DATE('20-MAR-2020', 'DD-MON-YYYY'),
    TO_DATE('20-MAR-2022', 'DD-MON-YYYY'),
    4,
    5 );

INSERT INTO echipament (
    nume,
    data_instalare,
    data_revizie,
    id_filiala,
    id_furnizor
) VALUES ( 'Sistem de catarare cu prindere pe perete',
    TO_DATE('30-SEP-2022', 'DD-MON-YYYY'),
    TO_DATE('30-SEP-2023', 'DD-MON-YYYY'),
    8,
    9 );

INSERT INTO echipament (
    nume,
    data_instalare,
    data_revizie,
    id_filiala,
    id_furnizor
) VALUES ( 'Semisfera de echilibru cu manere',
    TO_DATE('30-JUN-2022', 'DD-MON-YYYY'),
    TO_DATE('30-JUN-2023', 'DD-MON-YYYY'),
    8,
    9 );

INSERT INTO echipament (
    nume,
    data_instalare,
    data_revizie,
    id_filiala,
    id_furnizor
) VALUES ( 'Banca de gimnastica tip Pivetta',
    TO_DATE('01-JUN-2021', 'DD-MON-YYYY'),
    TO_DATE('01-JUN-2022', 'DD-MON-YYYY'),
    3,
    9 );

INSERT INTO echipament (
    nume,
    data_instalare,
    data_revizie,
    id_filiala,
    id_furnizor
) VALUES ( 'Coarda sarituri cu maner din lemn',
    TO_DATE('01-JAN-2022', 'DD-MON-YYYY'),

```

```

        TO_DATE('01-JUN-2023','DD-MON-YYYY'),
        3,
        9 );
INSERT INTO echipament (
    nume,
    data_instalare,
    data_revizie,
    id_filiala,
    id_furnizor
) VALUES ( 'Plan propioceptiv rotativ',
    TO_DATE('01-JUN-2023','DD-MON-YYYY'),
    TO_DATE('01-SEP-2023','DD-MON-YYYY'),
    3,
    9 );

EXEC sequence_utils.create_sequence_trigger('Comanda');

INSERT INTO comanda (
    id_client,
    id_receptionist,
    observatii,
    data_comandare
) VALUES ( 18,
    8,
    'Urgenta',
    TO_DATE('22-FEB-2022','DD-MON-YYYY') );
INSERT INTO comanda (
    id_client,
    id_receptionist,
    observatii,
    data_comandare
) VALUES ( 18,
    8,
    'Preluare dupa ora 17',
    TO_DATE('11-MAR-2022','DD-MON-YYYY') );
INSERT INTO comanda (
    id_client,
    id_receptionist,
    observatii,
    data_comandare
) VALUES ( 20,
    9,
    NULL,
    TO_DATE('01-APR-2022','DD-MON-YYYY') );
INSERT INTO comanda (
    id_client,
    id_receptionist,
    observatii,
    data_comandare
) VALUES ( 20,
    9,
    NULL,
    TO_DATE('02-APR-2022','DD-MON-YYYY') );

```

```

INSERT INTO comanda (
    id_client,
    id_receptionist,
    observatii,
    data_comandare
) VALUES ( 22,
            9,
            NULL,
            TO_DATE('22-APR-2022','DD-MON-YYYY') );

INSERT INTO comanda (
    id_client,
    id_receptionist,
    observatii,
    data_comandare
) VALUES ( 26,
            16,
            'In curs de achitare',
            TO_DATE('01-SEP-2023','DD-MON-YYYY') );

INSERT INTO comanda (
    id_client,
    id_receptionist,
    observatii,
    data_comandare
) VALUES ( 27,
            17,
            'Platita',
            TO_DATE('01-OCT-2023','DD-MON-YYYY') );

INSERT INTO comanda (
    id_client,
    id_receptionist,
    observatii,
    data_comandare
) VALUES ( 27,
            10,
            'Platita',
            TO_DATE('11-OCT-2023','DD-MON-YYYY') );

INSERT INTO comanda (
    id_client,
    id_receptionist,
    observatii,
    data_comandare
) VALUES ( 27,
            11,
            'Platita',
            TO_DATE('21-OCT-2023','DD-MON-YYYY') );

INSERT INTO comanda (
    id_client,
    id_receptionist,
    observatii,
    data_comandare
) VALUES ( 27,
            12,

```

```

        'Platita',
        TO_DATE('22-OCT-2023', 'DD-MON-YYYY') );
INSERT INTO comanda (
    id_client,
    id_receptionist,
    observatii,
    data_comandare
) VALUES ( 27,
    13,
    'Platita',
    TO_DATE('22-SEP-2022', 'DD-MON-YYYY') );

INSERT INTO tip_abonament (
    nume_tip,
    pret
) VALUES ( 'lunar',
    100 );
INSERT INTO tip_abonament (
    nume_tip,
    pret
) VALUES ( 'trimestrial',
    280 );
INSERT INTO tip_abonament (
    nume_tip,
    pret
) VALUES ( 'bianual',
    550 );
INSERT INTO tip_abonament (
    nume_tip,
    pret
) VALUES ( 'anual',
    800 );
INSERT INTO tip_abonament (
    nume_tip,
    pret
) VALUES ( 'extins',
    1500 );

EXEC sequence_utils.create_sequence_trigger('Abonament');
INSERT INTO abonament (
    nume_tip,
    id_client,
    data_inregistrare
) VALUES ( 'lunar',
    18,
    '01-APR-22' );
INSERT INTO abonament (
    nume_tip,
    id_client,
    data_inregistrare
) VALUES ( 'trimestrial',
    19,
    '01-APR-21' );

```

```

INSERT INTO abonament (
    nume_tip,
    id_client,
    data_inregistrare
) VALUES ( 'bianual',
            20,
            '01-FEB-22' );
INSERT INTO abonament (
    nume_tip,
    id_client,
    data_inregistrare
) VALUES ( 'extins',
            21,
            '01-SEP-21' );
INSERT INTO abonament (
    nume_tip,
    id_client,
    data_inregistrare
) VALUES ( 'anual',
            22,
            '01-NOV-20' );
INSERT INTO abonament (
    nume_tip,
    id_client,
    data_inregistrare
) VALUES ( 'anual',
            23,
            '01-NOV-22' );
INSERT INTO abonament (
    nume_tip,
    id_client,
    data_inregistrare
) VALUES ( 'anual',
            25,
            '01-DEC-22' );
INSERT INTO abonament (
    nume_tip,
    id_client,
    data_inregistrare
) VALUES ( 'bianual',
            26,
            '15-JUL-23' );
INSERT INTO abonament (
    nume_tip,
    id_client,
    data_inregistrare
) VALUES ( 'extins',
            27,
            '01-JAN-22' );

DECLARE
    nr NUMBER;
BEGIN

```

```

FOR i IN 1..22 LOOP
    SELECT round(dbms_random.value(
        1000000000,
        9999999999
    ))
    INTO nr
    FROM dual;
    IF i <= 10 THEN
        INSERT INTO telefon (
            tip,
            numar,
            id_persoana
        ) VALUES ( 'serviciu',
                    nr,
                    i );
    ELSE
        INSERT INTO telefon (
            tip,
            numar,
            id_persoana
        ) VALUES ( 'personal',
                    nr,
                    i );
    END IF;
END LOOP;
END;
/

INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 1,
            1,
            2 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 1,
            2,
            1 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 1,
            3,
            4 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate

```

```

) VALUES ( 1,
            4,
            3 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 1,
            5,
            7 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 2,
            1,
            2 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 3,
            1,
            2 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 3,
            2,
            1 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 3,
            4,
            1 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 3,
            5,
            5 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 6,
            4,
            3 );

```

```

INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 6,
            3,
            1 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 6,
            7,
            1 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 6,
            8,
            2 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 6,
            2,
            1 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 7,
            1,
            4 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 7,
            3,
            2 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 7,
            7,
            1 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,

```



```

        cantitate
    ) VALUES ( 7,
                8,
                5 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 7,
            9,
            2 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 7,
            10,
            1 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 8,
            7,
            1 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 9,
            8,
            5 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 10,
            9,
            2 );
INSERT INTO informatii_comanda (
    id_comanda,
    id_supliment,
    cantitate
) VALUES ( 11,
            10,
            1 );

COMMIT;

```

### 10.1.4. bro\_admin\_criptare.sql

```
--- Criptare in schema lui bro_admin
CREATE OR REPLACE FUNCTION select_random_from_nr_list (
    input_list sys.odcinumberlist
) RETURN NUMBER IS
    selected_value NUMBER;
BEGIN
    SELECT input_list(trunc(dbms_random.value(
        1,
        input_list.count + 1
    )))
    INTO selected_value
    FROM dual;

    RETURN selected_value;
END;
/

CREATE TABLE chei_client (
    id_client NUMBER
    CONSTRAINT fk_client_chei
    REFERENCES client ( id_client )
    PRIMARY KEY,
    mod_op INT NOT NULL,
    cheie RAW(16) NOT NULL
);

CREATE OR REPLACE PROCEDURE insert_into_chei_client (
    p_client NUMBER
) IS
    mod_op_value NUMBER;
    v_nr SMALLINT;
    v_cheie RAW(16) := dbms_crypto.randombytes(16);
BEGIN
    SELECT COUNT(*)
    INTO v_nr
    FROM chei_client c
    WHERE c.id_client = p_client;
    IF v_nr > 0 THEN
        raise_application_error(
            -20022,
            'The client is already in the table keys. Client ' || p_client
        );
    END IF;
    mod_op_value := dbms_crypto.encrypt_aes128 +
    bro_admin.select_random_from_nr_list(sys.odcinumberlist(
        dbms_crypto.pad_pkcs5,
        dbms_crypto.pad_zero
    )) + bro_admin.select_random_from_nr_list(sys.odcinumberlist(
        dbms_crypto.chain_cbc,
```

```

        dbms_crypto.chain_cfb,
        dbms_crypto.chain_ecb,
        dbms_crypto.chain_ofb
    ));

    INSERT INTO chei_client (
        id_client,
        mod_op,
        cheie
    ) VALUES ( p_client,
                mod_op_value,
                v_cheie );
END;
/

exec insert_into_chei_client(18);
exec insert_into_chei_client(18);
exec insert_into_chei_client(19);
exec insert_into_chei_client(20);
exec insert_into_chei_client(21);
exec insert_into_chei_client(22);
exec insert_into_chei_client(25);
exec insert_into_chei_client(26);
exec insert_into_chei_client(27);

SELECT *
FROM chei_client;
COMMIT;

CREATE OR REPLACE TYPE chei_client_object AS OBJECT (
    id_client NUMBER,
    mod_op     INT,
    cheie      RAW(16)
);
/

CREATE OR REPLACE FUNCTION get_client_key RETURN chei_client_object IS
    res chei_client_object;
BEGIN
    SELECT chei_client_object(
        c.id_client,
        c.mod_op,
        c.cheie
    )
    INTO res
    FROM account_mapping a
    JOIN chei_client c
    ON a.id_persoana = c.id_client
    WHERE username = sys_context(
        'userenv',
        'session_user'
    );

```

```

    RETURN res;
EXCEPTION
    WHEN no_data_found THEN
        raise_application_error(
            -20023,
            'Client with username '
            || sys_context(
                'userenv',
                'session_user'
            )
            || ' does not have a cript key'
        );
END;
/

```

### 10.1.5. bro\_admin\_mask.sql

```

CREATE OR REPLACE PACKAGE mask_person IS
    FUNCTION mask_item (
        item VARCHAR2
    ) RETURN VARCHAR2;
    FUNCTION mask_item (
        item NUMBER
    ) RETURN NUMBER;
    FUNCTION mask_person_id (
        item NUMBER
    ) RETURN NUMBER;
    FUNCTION mask_person_fk (
        item NUMBER
    ) RETURN NUMBER;
    PROCEDURE empty_person_ids;
END;
/
CREATE OR REPLACE PACKAGE BODY mask_person IS
    TYPE id_rec IS RECORD (
        old_value INT,
        new_value INT
    );
    TYPE new_key_rec IS RECORD (
        new_val INT,
        new_max INT,
        new_min INT
    );
    TYPE ids_tbl IS
        TABLE OF id_rec INDEX BY PLS_INTEGER;
    person_ids ids_tbl;
    FUNCTION find_person_by_value (
        val INT,
        raise_empty BOOLEAN := FALSE,
        use_old_val BOOLEAN := TRUE
    ) RETURN INT IS

```

```

BEGIN
    dbms_output.put_line(person_ids.count);
    FOR i IN 1..person_ids.count LOOP
        IF use_old_val THEN
            IF person_ids(i).old_value = val THEN
                RETURN person_ids(i).new_value;
            END IF;
        ELSE
            IF person_ids(i).new_value = val THEN
                RETURN person_ids(i).old_value;
            END IF;
        END IF;
    END LOOP;
    IF raise_empty THEN
        raise_application_error(
            -20020,
            'Id-ul '
            || val
            || ' nu este prezent in baza originala'
        );
    ELSE
        RETURN NULL;
    END IF;
END;

```

```

FUNCTION generate_new_key (
    val          INT,
    max_len_factor INT := 1
) RETURN new_key_rec IS
    len          INT := length(to_char(val));
    new_max_len INT := len * max_len_factor;
    new_key      new_key_rec;

```

```

BEGIN
    IF new_max_len > 38 THEN
        new_max_len := 38; -- mx nr id
    END IF;
    new_key.new_min := TO_NUMBER ( rpad(
        substr(
            to_char(val),
            1,
            1
        ),
        len,
        '0'
    ) );
    new_key.new_max := TO_NUMBER ( rpad(
        substr(
            to_char(val),
            1,
            1
        ),
        len,
        new_max_len,

```

```

        '9'
    ) ); -- to not have colission as often
    dbms_random.seed(val => val);
    new_key.new_val := round(
        dbms_random.value(
            low => new_key.new_min,
            high => new_key.new_max
        ),
        0
    );
    RETURN new_key;
END;

FUNCTION append_to_person_list (
    val INT
) RETURN INT IS
    rec      INT := find_person_by_value(val);
    pers_cnt INT := person_ids.count + 1;
    new_key  INT;
BEGIN
    IF rec IS NOT NULL THEN
        RETURN rec;
    END IF;
    new_key := generate_new_key(
        val,
        5
    ).new_val;
    WHILE ( find_person_by_value(
        val      => new_key,
        use_old_val => FALSE
    ) IS NOT NULL
    OR find_person_by_value(
        val      => new_key,
        use_old_val => TRUE
    ) IS NOT NULL ) LOOP
        new_key := generate_new_key(
            val,
            5
        ).new_val; -- no colisions
    END LOOP;
    person_ids(pers_cnt).old_value := val;
    person_ids(pers_cnt).new_value := new_key;
    RETURN new_key;
END;

-- ne trebuie tampanie asta crunta pt ca oracle exporta alfabetic
-- nu stiu dc face asta, dar asa face
PROCEDURE load_person_ids IS
    TYPE id_list_type IS TABLE OF persoana.id_persoana%TYPE;
    id_list id_list_type;
    dummy_val INT;
BEGIN
    IF person_ids.count = 0 THEN

```

```

        SELECT id_persoana
        BULK COLLECT INTO id_list
        FROM persoana;
        FOR i IN 1 .. id_list.count LOOP
            dummy_val := append_to_person_list(id_list(i));
        END LOOP;

dbms_output.put_line('IDs have been initialized in the table type.');
```

```

END IF;
END;

--public
FUNCTION mask_person_id (
    item NUMBER
) RETURN NUMBER IS
BEGIN
    load_person_ids();
--    return append_to_person_list(item);
    RETURN find_person_by_value(
        val      => item,
        raise_empty => TRUE
    );
END;

FUNCTION mask_item (
    item VARCHAR2
) RETURN VARCHAR2 IS
    masked_item VARCHAR2(30);
    new_length  NUMBER;
    random_char CHAR(1);
BEGIN

    IF dbms_random.value(
        0,
        1
    ) > 0.5 THEN
        new_length := length(item) * 2;
    ELSE
        new_length := length(item);
    END IF;

    IF new_length > 30 THEN
        new_length := 30; --max string length
    END IF;
    masked_item := substr(
        item,
        1,
        1
    );
    FOR i IN 2..new_length LOOP

```

```

        IF dbms_random.value(
            0,
            1
        ) > 0.5 THEN
            random_char :=
                CASE
                    WHEN dbms_random.value(
                        0,
                        1
                    ) > 0.5 THEN
                        '*'
                    ELSE '#'
                END;
            masked_item := masked_item || random_char;
        END IF;
    END LOOP;
    RETURN masked_item;
END;

FUNCTION mask_item (
    item NUMBER
) RETURN NUMBER IS
BEGIN
    RETURN generate_new_key(item).new_val;
END;

FUNCTION mask_person_fk (
    item NUMBER
) RETURN NUMBER IS
BEGIN
    IF item IS NULL THEN
        RETURN NULL;
    END IF;
    load_person_ids();
    RETURN find_person_by_value(
        val => item,
        raise_empty => TRUE
    );
END;

PROCEDURE empty_person_ids IS
BEGIN
    person_ids.DELETE;
END;

END;
/

```



## 10.1.6. bro\_admin\_programs\_view.sql

```
set serveroutput on;
DECLARE
    users sys.global_user_table;
BEGIN
    users := sys.get_users_by_suffix('ANTRENOR');
    FOR i IN 1..users.count LOOP
        dbms_output.put_line(users(i));
    END LOOP;
END;
/
CREATE OR REPLACE PROCEDURE bro_admin_programs_view IS
    v_user sys.global_user_table := sys.get_users_by_suffix('ANTRENOR');
    v_sql CLOB := 'CREATE OR REPLACE VIEW programs_view AS ';
    v_first BOOLEAN := TRUE;
BEGIN
    FOR i IN 1..v_user.count LOOP
        BEGIN
            -- check existence of program table for antrenor
            EXECUTE IMMEDIATE 'SELECT 1 FROM '
                                || v_user(i)
                                || '.program WHERE ROWNUM = 1';
            IF v_first THEN
                v_sql := v_sql
                    || 'SELECT '
                    || v_user(i)
                    || '' AS antrenor, id_program, descriere, tip_program
FROM '
                    || v_user(i)
                    || '.program';
                v_first := FALSE;
            ELSE
                v_sql := v_sql
                    || ' UNION ALL SELECT '
                    || v_user(i)
                    || '' AS antrenor, id_program, descriere, tip_program
FROM '
                    || v_user(i)
                    || '.program';
            END IF;
        END LOOP;
    EXCEPTION
        WHEN OTHERS THEN
            -- skip if table its not in antrenor
            dbms_output.put_line('Skipping '
                                || v_user(i)
                                || '.program as it does not exist.');
```

```
END;
END LOOP;
dbms_output.put_line(v_sql);
IF NOT v_first THEN
```

```

        EXECUTE IMMEDIATE v_sql;
        dbms_output.put_line('View programs_view created successfully.');
```

ELSE

```

        dbms_output.put_line('No valid program tables found. View not created.');
```

END IF;

```

END bro_admin_programs_view;
/
exec bro_admin_programs_view;

SELECT *
  FROM programs_view;

-- poate da grant la role, desi nu vede rolurile
--SELECT role FROM dba_roles WHERE role = 'R_BRO_PUBLIC_GENERAL';

GRANT SELECT ON bro_admin.programs_view TO r_bro_public_general;
```

### 10.1.7. bro\_admin\_update\_echipament\_fals.sql

```

BEGIN
  FOR i IN 1..20 LOOP
    UPDATE echipament o
      SET
        data_reviziei = (
          SELECT data_reviziei
            FROM echipament i
           WHERE i.id_echipament = o.id_echipament
        );
    COMMIT;
  END LOOP;
END;
/
```

## 10.2 Antrenor

### 10.2.1. bro\_antrenor\_insert.sql

```

-- Inserare date in tabelele din baza de date pentru antrenor
INSERT INTO program (
  descriere,
  tip_program
) VALUES ( 'Push, Pull Legs Light, for beginners',
            'MASS' );
INSERT INTO program (
  descriere,
  tip_program
) VALUES ( 'Push, Pull Legs Medium',
```

```

        'MASS' );
INSERT INTO program (
    descriere,
    tip_program
) VALUES ( 'Push, Pull Legs Hard',
    'MASS' );
INSERT INTO program (
    descriere,
    tip_program
) VALUES ( 'Full Body Variant Light',
    'CARDIO' );
INSERT INTO program (
    descriere,
    tip_program
) VALUES ( 'Body Recovery Variant Light',
    'RECOVERY' );
INSERT INTO program (
    descriere,
    tip_program
) VALUES ( 'Body Bluster Variant Blusting',
    'RECOVERY' );
INSERT INTO program (
    descriere,
    tip_program
) VALUES ( 'Cardio Workout for Weight Loss',
    'CARDIO' );
INSERT INTO program (
    descriere,
    tip_program
) VALUES ( 'Cardio workout for beginners',
    'CARDIO' );
INSERT INTO program (
    descriere,
    tip_program
) VALUES ( 'Cardio workout for older adults',
    'CARDIO' );

INSERT INTO antrenament (
    durata,
    id_program,
    id_echipament,
    id_client
) VALUES ( 20,
    1,
    1,
    18 );
INSERT INTO antrenament (
    durata,
    id_program,
    id_echipament,
    id_client
) VALUES ( 11,

```

```

        1,
        2,
        18 );
INSERT INTO antrenament (
    durata,
    id_program,
    id_echipament,
    id_client
) VALUES ( 11,
            4,
            1,
            18 );
INSERT INTO antrenament (
    durata,
    id_program,
    id_echipament,
    id_client
) VALUES ( 10,
            4,
            2,
            18 );
INSERT INTO antrenament (
    durata,
    id_program,
    id_echipament,
    id_client
) VALUES ( 5,
            1,
            1,
            19 );
INSERT INTO antrenament (
    durata,
    id_program,
    id_echipament,
    id_client
) VALUES ( 25,
            1,
            2,
            19 );
INSERT INTO antrenament (
    durata,
    id_program,
    id_echipament,
    id_client
) VALUES ( 32,
            1,
            3,
            19 );
INSERT INTO antrenament (
    durata,
    id_program,
    id_echipament,

```

```

        id_client
    ) VALUES ( 10,
                5,
                2,
                19 );
INSERT INTO antrenament (
    durata,
    id_program,
    id_echipament,
    id_client
) VALUES ( 5,
            4,
            3,
            20 );
INSERT INTO antrenament (
    durata,
    id_program,
    id_echipament,
    id_client
) VALUES ( 25,
            4,
            4,
            20 );
INSERT INTO antrenament (
    durata,
    id_program,
    id_echipament,
    id_client
) VALUES ( 12,
            4,
            5,
            21 );
INSERT INTO antrenament (
    durata,
    id_program,
    id_echipament,
    id_client
) VALUES ( 42,
            4,
            2,
            21 );
INSERT INTO antrenament (
    durata,
    id_program,
    id_echipament,
    id_client
) VALUES ( 20,
            4,
            5,
            22 );
INSERT INTO antrenament (
    durata,

```

```

        id_program,
        id_equipement,
        id_client
    ) VALUES ( 10,
                4,
                4,
                22 );
INSERT INTO antrenament (
    durata,
    id_program,
    id_equipement,
    id_client
) VALUES ( 10,
            7,
            8,
            25 );
INSERT INTO antrenament (
    durata,
    id_program,
    id_equipement,
    id_client
) VALUES ( 20,
            7,
            9,
            25 );
INSERT INTO antrenament (
    durata,
    id_program,
    id_equipement,
    id_client
) VALUES ( 10,
            7,
            10,
            25 );
INSERT INTO antrenament (
    durata,
    id_program,
    id_equipement,
    id_client
) VALUES ( 10,
            8,
            3,
            26 );
INSERT INTO antrenament (
    durata,
    id_program,
    id_equipement,
    id_client
) VALUES ( 20,
            8,
            4,
            19 );

```

```

INSERT INTO antrenament (
    durata,
    id_program,
    id_echipament,
    id_client
) VALUES ( 10,
            8,
            5,
            19 );

INSERT INTO antrenament (
    durata,
    id_program,
    id_echipament,
    id_client
) VALUES ( 10,
            8,
            12,
            27 );

INSERT INTO antrenament (
    durata,
    id_program,
    id_echipament,
    id_client
) VALUES ( 20,
            8,
            9,
            27 );

INSERT INTO antrenament (
    durata,
    id_program,
    id_echipament,
    id_client
) VALUES ( 10,
            8,
            11,
            27 );

COMMIT;

```

### 10.2.2. bro\_antrenor1\_cript\_show.sql

```

WITH c AS (
    SELECT mod_op,
           cheie
    FROM bro_admin.chei_client
    WHERE id_client = 18
)
SELECT decript_string(
    ca.id_program,
    c.mod_op,
    c.cheie
) AS id_program,

```

```

        decrypt_string(
            ca.descriere_program,
            c.mod_op,
            c.cheie
        ) AS descriere_program,
        decrypt_string(
            ca.tip_program,
            c.mod_op,
            c.cheie
        ) AS tip_program,
        decrypt_string(
            ca.durata_antrenament,
            c.mod_op,
            c.cheie
        ) AS durata_antrenament,
        decrypt_string(
            ca.id_echipament,
            c.mod_op,
            c.cheie
        ) AS id_echipament,
        decrypt_string(
            ca.nume_echipament,
            c.mod_op,
            c.cheie
        ) AS nume_echipament,
        decrypt_string(
            ca.data_instalare_echipament,
            c.mod_op,
            c.cheie
        ) AS data_instalare_echipament,
        decrypt_string(
            ca.data_revizie_echipament,
            c.mod_op,
            c.cheie
        ) AS data_revizie_echipament,
        decrypt_string(
            ca.id_filiala,
            c.mod_op,
            c.cheie
        ) AS id_filiala,
        decrypt_string(
            ca.id_client,
            c.mod_op,
            c.cheie
        ) AS id_client,
        checksum
FROM client_antrenament ca,
c;

SELECT *
FROM TABLE ( fetch_decrypted_client_data(
    (

```



```

        SELECT mod_op
          FROM bro_admin.chei_client
         WHERE id_client = 18
      ),
      (
        SELECT cheie
          FROM bro_admin.chei_client
         WHERE id_client = 18
      ),
      18
    ) );

```

### 10.2.3. bro\_antrenor1\_sql\_injection.sql

```

CREATE OR REPLACE PROCEDURE get_program_full (
  id_prg      NUMBER,
  data_inst   VARCHAR2
) IS
  TYPE program_echipament_rec IS RECORD (
    id_echipament bro_admin.echipament.id_echipament%TYPE,
    nume           bro_admin.echipament.nume%TYPE,
    data_instalare bro_admin.echipament.data_instalare%TYPE,
    data_revizie   bro_admin.echipament.data_revizie%TYPE,
    id_filiala     bro_admin.echipament.id_filiala%TYPE,
    id_furnizor    bro_admin.echipament.id_furnizor%TYPE,
    id_program     program.id_program%TYPE,
    descriere      program.descriere%TYPE,
    tip_program    program.tip_program%TYPE
  );
  TYPE program_echipament_tab IS
    TABLE OF program_echipament_rec;
  v_program_echipament program_echipament_tab;
  v_sql                 VARCHAR2(2500) := 'SELECT * FROM bro_admin.echipament e
    NATURAL JOIN program p
    WHERE p.id_program = '
    || id_prg
    || ' AND upper(to_char(data_revizie, ''DD-MON-YY''))
    LIKE ''%'
    || upper(data_inst)
    || '%''';
BEGIN
  dbms_output.put_line('SQL: ' || v_sql);
  EXECUTE IMMEDIATE v_sql
  BULK COLLECT
    INTO v_program_echipament;
  dbms_output.put_line('Program and Equipment Details:');
  dbms_output.new_line();
  FOR i IN 1..v_program_echipament.count LOOP
    dbms_output.put_line('ID_ECHIPAMENT: '
      || v_program_echipament(i).id_echipament
      || ', NUME: '

```

```

        || v_program_echipament(i).nume
        || ', DATA_INSTALARE: '
        || to_char(
v_program_echipament(i).data_instalare,
'YYYY-MM-DD'
)
        || ', DATA_REVIZIE: '
        || to_char(
v_program_echipament(i).data_revizie,
'YYYY-MM-DD'
)
        || ', ID_FILIALA: '
        || v_program_echipament(i).id_filiala
        || ', ID_FURNIZOR: '
        || v_program_echipament(i).id_furnizor
        || ', ID_PROGRAM: '
        || v_program_echipament(i).id_program
        || ', DESCRIERE: '
        || v_program_echipament(i).descriere
        || ', TIP_PROGRAM: '
        || v_program_echipament(i).tip_program);
dbms_output.new_line();
END LOOP;
EXCEPTION
    WHEN no_data_found THEN
        dbms_output.put_line('No data found for the specified program ID: ' ||
id_prg);
    WHEN OTHERS THEN
        dbms_output.put_line('An error occurred running get_program_full: ' ||
sqlerrm);
END;
/

GRANT EXECUTE ON get_program_full TO bro_client1;
/
-- sql injection fix

CREATE OR REPLACE PROCEDURE get_program_full_safe (
    id_prg    NUMBER,
    data_inst VARCHAR2
) IS
    TYPE program_echipament_rec IS RECORD (
        id_echipament bro_admin.echipament.id_echipament%TYPE,
        nume          bro_admin.echipament.nume%TYPE,
        data_instalare bro_admin.echipament.data_instalare%TYPE,
        data_revizie  bro_admin.echipament.data_revizie%TYPE,
        id_filiala    bro_admin.echipament.id_filiala%TYPE,
        id_furnizor   bro_admin.echipament.id_furnizor%TYPE,
        id_program    program.id_program%TYPE,
        descriere     program.descriere%TYPE,
        tip_program   program.tip_program%TYPE
    );
    TYPE program_echipament_tab IS TABLE OF program_echipament_rec;

```

```

v_program_echipament program_echipament_tab;
v_sql VARCHAR2(2500) := 'SELECT
                                e.id_echipament,
                                e.num_e,
                                e.data_instalare,
                                e.data_revizie,
                                e.id_filiala,
                                e.id_furnizor,
                                p.id_program,
                                p.descriere,
                                p.tip_program
                        FROM bro_admin.echipament e
                        NATURAL JOIN program p
                        WHERE p.id_program = :id_prg
                        AND UPPER(TO_CHAR(e.data_revizie, 'DD-MON-YY'))
LIKE :data_inst';
BEGIN
    dbms_output.put_line('Executing SQL: ' || v_sql);

    EXECUTE IMMEDIATE v_sql BULK COLLECT
        INTO v_program_echipament
        USING id_prg, '%' || upper(data_inst) || '%';

    dbms_output.put_line('Program and Equipment Details:');
    dbms_output.new_line();

    FOR i IN 1 .. v_program_echipament.count LOOP
        dbms_output.put_line('ID_ECHIPAMENT: ' ||
v_program_echipament(i).id_echipament ||
                                ', NUME: ' || v_program_echipament(i).num_e ||
                                ', DATA_INSTALARE: ' ||
to_char(v_program_echipament(i).data_instalare, 'YYYY-MM-DD') ||
                                ', DATA_REVIZIE: ' ||
to_char(v_program_echipament(i).data_revizie, 'YYYY-MM-DD') ||
                                ', ID_FILIALA: ' ||
v_program_echipament(i).id_filiala ||
                                ', ID_FURNIZOR: ' ||
v_program_echipament(i).id_furnizor ||
                                ', ID_PROGRAM: ' ||
v_program_echipament(i).id_program ||
                                ', DESCRIERE: ' || v_program_echipament(i).descriere
||
                                ', TIP_PROGRAM: ' ||
v_program_echipament(i).tip_program);
        dbms_output.new_line();
    END LOOP;

    EXCEPTION
        WHEN no_data_found THEN
            dbms_output.put_line('No data found for the specified program ID: ' ||
id_prg);
        WHEN OTHERS THEN

```

```

        dbms_output.put_line('An error occurred running get_program_full: ' ||
sqlerrm);
END;
/

```

```
GRANT EXECUTE ON get_program_full_safe TO bro_client1;
```

## 10.3. Client

### 10.3.1. bro\_client1\_select\_cript.sql

```
SELECT bro_admin.get_client_key
FROM dual;
```

```
SELECT ( bro_admin.get_client_key() ).id_client AS id_client,
       ( bro_admin.get_client_key() ).mod_op AS mod_op,
       ( bro_admin.get_client_key() ).cheie AS cheie
FROM dual;
```

```
WITH user_key AS (
    SELECT bro_admin.get_client_key() AS client_key
    FROM dual
)
SELECT ( user_key.client_key ).id_client AS id_client,
       ( user_key.client_key ).mod_op AS mod_op,
       ( user_key.client_key ).cheie AS cheie
FROM user_key;
```

```
WITH user_key AS (
    SELECT bro_admin.get_client_key() AS client_key
    FROM dual
)

```

```
SELECT ant.*,cs.*,
       CASE WHEN ant.checksum = cs.cur_cs THEN 'ok' ELSE 'not ok' END AS cs_v
FROM
    (SELECT bro_admin.get_client_key() AS client_key
    FROM dual)
    user_key,
LATERAL (SELECT *FROM TABLE(
    bro_antrenor1.fetch_decrypted_client_data(
        p_mod_op=>user_key.client_key.mod_op,
        p_cheie=>user_key.client_key.cheie,
        p_id_client=>user_key.client_key.id_client))) ant,
LATERAL (
SELECT
bro_antrenor1.hash_checksum(sys.odcivarchar2list(ant.id_program,ant.descriere_p
rogram,ant.tip_program,
                                ant.durata_antrenament,ant.id_echipament,ant.num_echip
ament,
```



```

                                ''Injectat Desc'', ''Tip injectat'' FROM
ANTRENAMENT --');
END;
/

```

## 10.4. Import

### 10.4.1. bro\_import.sql

```

SELECT constraint_name, constraint_type
FROM user_constraints
WHERE table_name = upper('angajat_mask');

SELECT * FROM persoana_mask;

SELECT * FROM angajat_mask
JOIN persoana_mask
ON id_angajat=id_persoana;

```

## 10.5 Manager

### 10.5.1. bro\_manager\_filiala1\_context.sql

```

SELECT sys_context(
    'bro_context',
    'id_filiala'
) FROM dual;

SELECT DISTINCT id_filiala FROM bro_admin.echipament;

UPDATE bro_admin.echipament o
SET
    nume = (
        SELECT nume
        FROM bro_admin.echipament i
        WHERE i.id_echipament = o.id_echipament
    )
WHERE o.id_filiala = 1;

UPDATE bro_admin.echipament o
SET
    nume = (
        SELECT nume
        FROM bro_admin.echipament i
        WHERE i.id_echipament = o.id_echipament
    )
WHERE o.id_filiala != 1;

```

```

UPDATE bro_admin.echipament o
SET
  nume = (
    SELECT nume
    FROM bro_admin.echipament i
    WHERE i.id_echipament = o.id_echipament
  );

```

```

SELECT COUNT(*)
FROM bro_admin.audit_echipament a
WHERE a.old_values.id_filiala=1 OR a.old_values.id_filiala=1;

```

```

SELECT COUNT(*)
FROM bro_admin.audit_echipament a
WHERE a.old_values.id_filiala!=1 AND a.old_values.id_filiala!=1;

```

## 10.7. SYS

### 10.7.1. sys\_admin\_antrenor\_privilege.sql

```

-- dupa ce admin a create tabelul pt antrenori
ALTER SESSION SET container = orclpdb;
CREATE OR REPLACE PROCEDURE bro_admin_programs_privileges IS
  v_user global_user_table := get_users_by_suffix('ANTRENOR');
  v_first BOOLEAN := TRUE;
BEGIN
  FOR i IN 1..v_user.count LOOP
    dbms_output.put_line(v_user(i));
    BEGIN
      EXECUTE IMMEDIATE 'SELECT 1 FROM '
        || v_user(i)
        || '.program WHERE ROWNUM = 1';
      EXECUTE IMMEDIATE 'grant select on '
        || v_user(i)
        || '.program to bro_admin with grant option';
      dbms_output.put_line('Giving select privileges on '
        || v_user(i)
        || '.program to bro_admin.');
```

```

      IF v_first THEN
        v_first := FALSE;
      END IF;
    EXCEPTION
      WHEN OTHERS THEN
        dbms_output.put_line('Skipping '
          || v_user(i)
          || '.program as it does not exist.');
```

```

  END;
END LOOP;

```

```

        IF NOT v_first THEN
            dbms_output.put_line('Giving select privileges on program tables to
bro_admin.');
```

```

        ELSE
            dbms_output.put_line('No valid program tables found');
```

```

        END IF;
END bro_admin_programs_privileges;
/

set SERVEROUTPUT ON;

exec bro_admin_programs_privileges;
```

## 10.7.2. sys\_audit\_1.sql

```

ALTER SESSION SET container = orclpdb;

SHOW parameter audit_trail;

AUDIT INSERT,UPDATE ON bro_admin.client_extins BY ACCESS WHENEVER NOT
SUCCESSFUL;
AUDIT INSERT,UPDATE,DELETE ON bro_admin.echipament;
AUDIT INSERT,UPDATE,DELETE ON bro_admin.account_mapping;
AUDIT INSERT,UPDATE ON bro_admin.supliment;

CREATE OR REPLACE DIRECTORY json_dir AS 'D:\ORACLEEE\INSTALL\ADMIN\ORCL\MYDUMP';

-- Multumiri deosebite primului raspuns de aici si nu documentatiei oracle care
este oribila
-- https://stackoverflow.com/questions/50417586/write-a-clob-to-file-in-
oracle
CREATE OR REPLACE PROCEDURE convert_clob_2_file (
    p_filename IN VARCHAR2,
    p_dir      IN VARCHAR2,
    p_clob     IN CLOB
) AS
    v_lob_image_id NUMBER;
    v_clob         CLOB := p_clob;
    v_buffer       RAW(32767);
    c_buffer       VARCHAR2(32767);
    v_buffer_size  BINARY_INTEGER;
    v_amount       BINARY_INTEGER;
    v_pos          NUMBER(38) := 1;
    v_clob_size    INTEGER;
    v_out_file     utl_file.file_type;
BEGIN
    v_pos := 1;
    v_clob_size := dbms_lob.getlength(v_clob);
    v_buffer_size := 32767;
    v_amount := v_buffer_size;
    IF ( dbms_lob.isopen(v_clob) = 0 ) THEN
        dbms_lob.open(
            v_clob,
```



```

        dbms_lob.lob_readonly
    );
END IF;
v_out_file := utl_file.fopen(
    p_dir,
    p_filename,
    'WB',
    max_linesize => 32767
);
WHILE v_amount >= v_buffer_size LOOP
    dbms_lob.read(
        v_clob,
        v_amount,
        v_pos,
        c_buffer
    );
    v_buffer := utl_raw.cast_to_raw(c_buffer);
    v_pos := v_pos + v_amount;
    utl_file.put_raw(
        v_out_file,
        v_buffer,
        TRUE
    );
    utl_file.fflush(v_out_file);
END LOOP;
utl_file.fflush(v_out_file);
utl_file.fclose(v_out_file);
IF ( dbms_lob.isopen(v_clob) = 1 ) THEN
    dbms_lob.close(v_clob);
END IF;
EXCEPTION
    WHEN OTHERS THEN
        IF ( dbms_lob.isopen(v_clob) = 1 ) THEN
            dbms_lob.close(v_clob);
        END IF;
        RAISE;
END;
/

CREATE OR REPLACE PROCEDURE save_audit_to_json (
    p_obj_name VARCHAR2
) IS
    obj_name   VARCHAR2(128) := dbms_assert.simple_sql_name(p_obj_name);
    a_file     utl_file.file_type;
    a_json     CLOB;
    a_filename VARCHAR2(1000);
BEGIN
    FOR owner_rec IN (
        SELECT DISTINCT obj$creator
        FROM sys.aud$
        WHERE obj$name = upper(obj_name)
    ) LOOP
        a_filename := 'audit_json_bro_'

```

```

        || owner_rec.obj$creator
        || '-'
        || obj_name
        || '-'
        || to_char(
sysdate,
'YYYYMMDD_HH24MISS'
)
        || '.json';

SELECT to_clob(JSON_ARRAYAGG(
  JSON_OBJECT(
    KEY 'sessionId' VALUE sessionid,
    KEY 'userId' VALUE userid,
    KEY 'entryId' VALUE entryid,
    KEY 'userhost' VALUE userhost,
    KEY 'returncode' VALUE returncode,
    KEY 'timestamp' VALUE to_char(
      ntimestamp#,
      'yyyy-mm-dd hh24:mi:ss'
    ),
    KEY 'sqltext' VALUE sqltext,
    KEY 'objectName' VALUE obj_name,
    KEY 'objectOwner' VALUE owner_rec.obj$creator
  )
  RETURNING CLOB)
RETURNING CLOB))
  INTO a_json
  FROM sys.aud$
 WHERE obj$name = upper(obj_name)
    AND obj$creator = upper(owner_rec.obj$creator);

convert_clob_2_file(
  p_clob      => a_json,
  p_dir       => 'JSON_DIR',
  p_filename => a_filename
);
EXECUTE IMMEDIATE 'delete from SYS.AUD$ where obj$name = upper(:1) and
obj$creator = upper(:2)'
  USING obj_name,
  owner_rec.obj$creator;
END LOOP;
COMMIT;
EXCEPTION
  WHEN OTHERS THEN
    dbms_output.put_line('Error occurred: ' || sqlerrm);
  RAISE;
END;
/

exec save_audit_to_json('client_extins');
exec save_audit_to_json('echipament');
```

```

exec save_audit_to_json('account_mapping');

-- create audit jobs
DECLARE
  job_names odcivarchar2list := odcivarchar2list(
    'client_extins',
    'echipament',
    'account_mapping',
    'supliment'
  );
BEGIN
  FOR i IN 1..job_names.count LOOP
    dbms_scheduler.create_job(
      job_name      => 'SAVE_AUDIT_TO_JSON_JOB_' || upper(job_names(i)),
      job_type      => 'PLSQL_BLOCK',
      job_action    => 'BEGIN save_audit_to_json(''
                        || upper(job_names(i))
                        || ''); END;',
      start_date    => systimestamp,
      repeat_interval => 'FREQ=DAILY; BYHOUR=17; BYMINUTE=22; BYSECOND=0',
      enabled       => TRUE
    );
  END LOOP;
END;
/

-- drop audit jobs
-- declare
--   job_names odcivarchar2list;
-- begin
--   select job_name
--   bulk collect
--   into job_names
--   from user_scheduler_jobs
--   where job_name like 'SAVE_AUDIT_TO_JSON_JOB_%';
--   for i in 1..job_names.count loop
--     dbms_scheduler.drop_job(job_name => upper(job_names(i)));
--   end loop;

-- end;
-- /

-- noaudit all on bro_admin.client_extins;
-- noaudit all on bro_admin.echipament;
-- noaudit all on bro_admin.account_mapping;
-- noaudit all on bro_admin.supliment;

SELECT * FROM dba_audit_trail
WHERE username LIKE upper('bro%');

SELECT * FROM dba_scheduler_jobs
WHERE job_name LIKE upper('save_audit_to_json_job_%')
ORDER BY job_name;

```

### 10.7.3. sys\_audit\_2.sql

```
ALTER SESSION SET container = orclpdb;

CREATE OR REPLACE DIRECTORY fgadump_dir AS
'D:\OracleEE\install\admin\orcl\fgadump';

-- salvare intr-un fisier txt a logurilor
CREATE OR REPLACE PROCEDURE bro_audit_tablese_handler (
    object_schema VARCHAR2,
    object_name    VARCHAR2,
    policy_name    VARCHAR2
) IS
    fga_file      utl_file.file_type;
    log_message   VARCHAR2(5000);
BEGIN
    log_message := 'FGA Triggered:'
                  || chr(10)
                  || 'Timestamp: '
                  || to_char(
                      sysdate,
                      'YYYY-MM-DD HH24:MI:SS'
                  )
                  || chr(10)
                  || 'Object Schema: '
                  || object_schema
                  || chr(10)
                  || 'Object Name: '
                  || object_name
                  || chr(10)
                  || 'Policy Name: '
                  || policy_name
                  || chr(10);
    fga_file := utl_file.fopen(
        'FGADUMP_DIR',
        policy_name || '.txt',
        'a'
    );
    utl_file.put_line(
        fga_file,
        log_message
    );
    utl_file.fflush(fga_file);
    utl_file.fclose(fga_file);
EXCEPTION
    WHEN OTHERS THEN
        IF utl_file.is_open(fga_file) THEN
            utl_file.fclose(fga_file);
        END IF;
        RAISE;
END;
/
```

```

CREATE OR REPLACE PROCEDURE echipament_revizie_audit IS
BEGIN
    dbms_fga.add_policy(
        object_schema => 'BRO_ADMIN',
        object_name   => 'ECHIPAMENT',
        policy_name    => 'AUDIT_DATA_REVIZIE',
        audit_condition => 'data_revizie IS NOT NULL',
        audit_column   => 'data_revizie',
        handler_schema => 'SYS',
        handler_module  => 'bro_audit_tablese_handler',
        enable          => TRUE,
        statement_types => 'insert,update'
    );
END;
/

exec echipament_revizie_audit();

SELECT *
FROM dba_fga_audit_trail
WHERE policy_name = 'AUDIT_DATA_REVIZIE';

BEGIN
    dbms_fga.enable_policy(
        object_schema => 'BRO_ADMIN',
        object_name   => 'ECHIPAMENT',
        policy_name    => 'AUDIT_DATA_REVIZIE'
    );
END;
/

-- begin
--     dbms_fga.disable_policy(
--         object_schema => 'BRO_ADMIN',
--         object_name   => 'ECHIPAMENT',
--         policy_name    => 'AUDIT_DATA_REVIZIE'
--     );
-- end;
-- /

-- begin
--     dbms_fga.drop_policy(
--         object_schema => 'BRO_ADMIN',
--         object_name   => 'ECHIPAMENT',
--         policy_name    => 'AUDIT_DATA_REVIZIE'
--     );
-- end;
-- /

```

## 10.7.4 sys\_context.sql

```
ALTER SESSION SET container = orclpdb;
--create contextului
CREATE OR REPLACE PROCEDURE bro_context_proc IS
BEGIN
    IF regexp_like(
        upper(user),
        '^BRO_ADMIN'
    ) THEN
        dbms_session.set_context(
            'bro_context',
            'id_filiala',
            -1
        ); -- pentru a lasa adminul in pace
    ELSIF regexp_like(
        upper(user),
        '^BRO_MANAGER_FILIALA[0-9]+$'
    ) THEN
        dbms_session.set_context(
            'bro_context',
            'id_filiala',
            regexp_substr(
                user,
                'BRO_MANAGER_FILIALA([0-9]+)',
                1,
                1,
                NULL,
                1
            )
        );
    END IF;
END;
/
DROP CONTEXT bro_context;
CREATE CONTEXT bro_context USING bro_context_proc;

--trigger pt a popula contextul
CREATE OR REPLACE TRIGGER manager_id_filiala_trg
AFTER LOGON ON DATABASE BEGIN
    bro_context_proc();
END;
/

-- fiecare manager face dml pe filiala lui
CREATE OR REPLACE FUNCTION manager_echipament_management_policy (
    schema_name IN VARCHAR2,
    table_name IN VARCHAR2
) RETURN VARCHAR2 IS
    v_id_filiala INT := sys_context(
        'bro_context',
        'id_filiala'
```

```

    );
BEGIN
    IF v_id_filiala = -1 THEN
        RETURN ''; -- pentru a lasa adminul in pace
    ELSE
        RETURN 'id_filiala = ' || v_id_filiala;
    END IF;
END;
/
BEGIN
    dbms_ols.add_policy(
        object_schema => 'BRO_ADMIN',
        object_name    => 'ECHIPAMENT',
        policy_name     => 'MANAGER_ECHIPAMENT_POLICY',
        function_schema => 'SYS',
        policy_function => 'manager_echipament_management_policy',
        statement_types => 'INSERT, UPDATE, DELETE',
        update_check    => TRUE,
        enable           => TRUE
    );
END;
/

-- begin
--     dbms_ols.drop_policy(
--         object_schema => 'BRO_ADMIN',
--         object_name    => 'ECHIPAMENT',
--         policy_name     => 'MANAGER_ECHIPAMENT_POLICY'
--     );
-- end;
-- /

-- fiecare manager vede istoric cu leagatura la filiala lui
CREATE OR REPLACE FUNCTION audit_echipament_policy (
    schema_name IN VARCHAR2,
    table_name  IN VARCHAR2
) RETURN VARCHAR2 AS
    v_id_filiala INT := sys_context(
        'bro_context',
        'id_filiala'
    );
BEGIN
    IF upper(user) = upper(schema_name) THEN
        RETURN ''; -- pentru a lasa adminul in pace
    ELSE
        RETURN '
            (TREAT(old_values AS bro_admin.t_echipament).id_filiala = '
                || v_id_filiala
                || ' OR TREAT(new_values AS bro_admin.t_echipament).id_filiala = '
                || v_id_filiala
                || ')';
    END IF;
END audit_echipament_policy;

```

```

/

BEGIN
    dbms_ols.add_policy(
        object_schema => 'bro_admin',
        object_name    => 'audit_echipament',
        policy_name     => 'AUDIT_MANAGER_ECHIPAMENT_POLICY',
        function_schema => 'SYS',
        policy_function  => 'audit_echipament_policy',
        statement_types  => 'SELECT',
        update_check     => TRUE,
        enable           => TRUE
    );
END;
/

-- begin
--     dbms_ols.drop_policy(
--         object_schema => 'BRO_ADMIN',
--         object_name    => 'audit_echipament',
--         policy_name     => 'AUDIT_MANAGER_ECHIPAMENT_POLICY'
--     );
-- end;
-- /

SELECT *
FROM dba_policies
WHERE object_owner LIKE upper('bro%');

```

### 10.7.5. sys\_mask.sql

```

CREATE OR REPLACE DIRECTORY mask_dump AS
'D:\OracleEE\install\admin\orcl\maskdump';
GRANT READ,WRITE ON DIRECTORY mask_dump TO bro_admin;
--pt import ca sa nu avem coliziuni la import
CREATE USER bro_import IDENTIFIED BY bro_import;
GRANT
    CREATE SESSION
TO bro_import;
GRANT
    CREATE TABLE,
    CREATE SEQUENCE
TO bro_import;
ALTER USER bro_import
    QUOTA 20M ON users;
GRANT datapump_imp_full_database TO bro_import;

```



## 10.7.6. sys\_users\_1.sql

```
-- Conexiune sys pentru crearea utilizatorilor

ALTER SESSION SET container = "orclpdb";
ALTER DATABASE orclpdb OPEN;

-- Functie ce asigura ca parolele contin cel putin un _
-- Trebuie sa fie standalone pt ca oracle nu vrea din pachet
CREATE OR REPLACE FUNCTION password_verify_function_standalone (
    username      VARCHAR2,
    new_password  VARCHAR2,
    old_password  VARCHAR2
) RETURN BOOLEAN IS
BEGIN
    IF instr(
        new_password,
        '_'
    ) = 0 THEN
        raise_application_error(
            -20003,
            'Password must contain at least one underscore (_) character.'
        );
    ELSE
        RETURN FALSE;
    END IF;
END;
/

--Pachet utilizat pentru gestiunea utilizatorilor
CREATE OR REPLACE PACKAGE bro_user_utils IS
    antrenor_suffix CONSTANT VARCHAR2(40) := 'ANTRENOR';
    receptionist_suffix CONSTANT VARCHAR2(40) := 'RECEPTIONIST';
    manager_suffix CONSTANT VARCHAR2(40) := 'MANAGER_FILIALA';
    client_suffix CONSTANT VARCHAR2(40) := 'CLIENT';
    public_suffix CONSTANT VARCHAR2(40) := 'PUBLIC_GENERAL';
    admin_suffix CONSTANT VARCHAR2(40) := 'ADMIN';
    bro_tablespace CONSTANT VARCHAR2(40) := 'USERS';
    antrenor_quota CONSTANT VARCHAR2(40) := '10M';
    bro_profile_public_general CONSTANT VARCHAR2(40) :=
'BRO_PROFILE_PUBLIC_GENERAL';
    bro_profile_antrenor CONSTANT VARCHAR2(40) := 'BRO_PROFILE_ANTRENOR';
    bro_profile_receptionist CONSTANT VARCHAR2(40) := 'BRO_PROFILE_RECEPTIONIST';
    bro_profile_manager CONSTANT VARCHAR2(40) := 'BRO_PROFILE_MANAGER';
    bro_profile_client CONSTANT VARCHAR2(40) := 'BRO_PROFILE_CLIENT';
    bro_plan CONSTANT VARCHAR2(10) := 'P_BRO';
    bro_role CONSTANT VARCHAR2(10) := 'R_BRO_';
    TYPE user_names IS
        TABLE OF VARCHAR2(128) INDEX BY PLS_INTEGER;
    TYPE suffix_names IS
        TABLE OF VARCHAR2(40) INDEX BY PLS_INTEGER;
    FUNCTION get_suffixes RETURN suffix_names;
```

```

FUNCTION get_users (
    suffix VARCHAR2
) RETURN user_names;

PROCEDURE create_user_by_suffix (
    suffix          VARCHAR2,
    password_expire BOOLEAN := FALSE
);

PROCEDURE create_user (
    user_name       VARCHAR2,
    password_expire BOOLEAN := FALSE
);

PROCEDURE alter_quota (
    user_name VARCHAR2
);

PROCEDURE alter_quota_all (
    suffix VARCHAR2
);

FUNCTION password_verify_function (
    username      VARCHAR2,
    new_password  VARCHAR2,
    old_password  VARCHAR2
) RETURN BOOLEAN;

PROCEDURE create_profile (
    suffix VARCHAR2
);

PROCEDURE assign_profile (
    user_name VARCHAR2
);

PROCEDURE assign_profile_all (
    suffix VARCHAR2
);

PROCEDURE configure_user_by_suffix (
    suffix          VARCHAR2,
    create_user     BOOLEAN := TRUE
);

PROCEDURE create_bro_plan_rg;

PROCEDURE clear_bro_plan_rg;

PROCEDURE create_role (
    suffix  VARCHAR2,
    override BOOLEAN := FALSE
);

PROCEDURE assign_role (

```

```

        suffix VARCHAR2
    );
END bro_user_utils;
/

CREATE OR REPLACE PACKAGE BODY bro_user_utils IS

    PROCEDURE is_valid_suffix (
        suffix VARCHAR2
    ) IS
    BEGIN
        IF suffix NOT IN ( antrenor_suffix,
                           receptionist_suffix,
                           manager_suffix,
                           client_suffix,
                           public_suffix ) THEN
            raise_application_error(
                -20002,
                'Invalid SUFFIX provided. Must be one of: ANTRENOR, RECEPTIONIST,
MANAGER_FILIALA, CLIENT, PUBLIC_GENERAL.'
            );
        END IF;
    END;

    FUNCTION is_valid_user_name (
        user_name VARCHAR2
    ) RETURN VARCHAR2 IS
        v_suffix VARCHAR2(40);
    BEGIN
        IF NOT regexp_like(
            lower(user_name),
            '^bro_[a-z0-9_]+$',
            'i'
        ) THEN
            raise_application_error(
                -20005,
                'Invalid user name provided. Must start with "bro_" and contain
only letters, numbers, and underscores (case-insensitive).'
            );
        END IF;

        v_suffix := regexp_replace(
            upper(substr(
                user_name,
                5
            )),
            '[0-9]',
            ''
        );
        is_valid_suffix(v_suffix);
        RETURN v_suffix;
    END;

```

```

FUNCTION get_profile_name (
    suffix VARCHAR2
) RETURN VARCHAR2 IS
    v_profile_name VARCHAR2(40);
BEGIN
    is_valid_suffix(suffix);
    CASE
        WHEN suffix = antrenor_suffix THEN
            v_profile_name := bro_profile_antrenor;
        WHEN suffix = receptionist_suffix THEN
            v_profile_name := bro_profile_receptionist;
        WHEN suffix = manager_suffix THEN
            v_profile_name := bro_profile_manager;
        WHEN suffix = client_suffix THEN
            v_profile_name := bro_profile_client;
        WHEN suffix = public_suffix THEN
            v_profile_name := bro_profile_public_general;
    END CASE;

    RETURN v_profile_name;
END get_profile_name;

PROCEDURE drop_profile (
    v_profile_name VARCHAR2
) IS
    v_cnt INT;
BEGIN
    SELECT COUNT(DISTINCT profile)
    INTO v_cnt
    FROM dba_profiles
    WHERE profile = upper(v_profile_name);
    IF v_cnt > 1 THEN
        raise_application_error(
            -20004,
            'Impossible situation occurred'
        );
    ELSIF v_cnt = 1 THEN
        EXECUTE IMMEDIATE 'DROP PROFILE '
            || v_profile_name
            || ' CASCADE';
    END IF;
END;

FUNCTION get_users (
    suffix VARCHAR2
) RETURN user_names IS
    v_names user_names;
BEGIN
    is_valid_suffix(suffix);
    SELECT username
    BULK COLLECT
    INTO v_names
    FROM dba_users

```

```

        WHERE username LIKE upper('bro_'
                                || suffix
                                || '%');

    RETURN v_names;
EXCEPTION
    WHEN no_data_found THEN
        dbms_output.put_line('No users found for suffix: ' || suffix);
        raise_application_error(
            -20006,
            'No users found for suffix: ' || suffix
        );
END;

FUNCTION get_user_by_suffix (
    suffix      VARCHAR2,
    next_user   BOOLEAN := FALSE
) RETURN VARCHAR2 IS
    v_suffix_cnt INT;
    v_user_name  VARCHAR2(128);
BEGIN
    SELECT COUNT(*)
        INTO v_suffix_cnt
        FROM dba_users
        WHERE username LIKE upper('bro_'
                                || suffix
                                || '%');

    IF next_user THEN
        v_suffix_cnt := v_suffix_cnt + 1;
    END IF;
    dbms_output.put_line('V_SUFFIX_CNT: ' || v_suffix_cnt);
    v_user_name := upper('bro_'
                        || suffix
                        || v_suffix_cnt);

    RETURN v_user_name;
END;

--public
FUNCTION get_suffixes RETURN suffix_names IS
    v_suffixes suffix_names;
BEGIN
    v_suffixes(1) := antrenor_suffix;
    v_suffixes(2) := receptionist_suffix;
    v_suffixes(3) := manager_suffix;
    v_suffixes(4) := client_suffix;
    v_suffixes(5) := public_suffix;
    RETURN v_suffixes;
END;

PROCEDURE create_user (
    user_name      VARCHAR2,
    password_expire BOOLEAN := FALSE
) IS

```

```

user_count          SMALLINT;
stmt                VARCHAR2(200);
sanitized_user_name VARCHAR2(128);
BEGIN
    sanitized_user_name := dbms_assert.simple_sql_name(user_name);
    SELECT COUNT(*)
      INTO user_count
    FROM dba_users
   WHERE username = upper(sanitized_user_name);
    IF user_count > 1 THEN
        raise_application_error(
            -20001,
            'Impossible situation occurred'
        );
    ELSIF user_count = 1 THEN
        EXECUTE IMMEDIATE 'DROP USER "'
                        || upper(sanitized_user_name)
                        || '" CASCADE';
    ELSE
        EXECUTE IMMEDIATE 'CREATE USER "'
                        || upper(sanitized_user_name)
                        || '" IDENTIFIED BY "'
                        || lower(sanitized_user_name)
                        || '"';
        IF password_expire THEN
            EXECUTE IMMEDIATE 'ALTER USER "'
                            || upper(sanitized_user_name)
                            || '" PASSWORD EXPIRE';
        END IF;

        EXECUTE IMMEDIATE 'GRANT CREATE SESSION TO "'
                        || upper(sanitized_user_name)
                        || '"';
    END IF;

    COMMIT;
END;

PROCEDURE create_user_by_suffix (
    suffix          VARCHAR2,
    password_expire BOOLEAN := FALSE
) IS
    v_user_name VARCHAR2(128);
BEGIN
    v_user_name := get_user_by_suffix(
        suffix,
        TRUE
    );
    create_user(
        v_user_name,
        password_expire
    );
END;

```

```

PROCEDURE alter_quota (
    user_name VARCHAR2
) IS
    v_suffix VARCHAR2(40);
    v_quota VARCHAR2(40);
BEGIN
    v_suffix := is_valid_user_name(user_name);
    IF v_suffix = antrenor_suffix THEN
        v_quota := antrenor_quota;
    ELSE
        v_quota := '0M';
    END IF;

    EXECUTE IMMEDIATE 'ALTER USER '
        || user_name
        || ' QUOTA '
        || v_quota
        || ' ON '
        || bro_tablespace;
    EXECUTE IMMEDIATE 'ALTER USER '
        || user_name
        || ' DEFAULT TABLESPACE '
        || bro_tablespace;

    COMMIT;
END;

PROCEDURE alter_quota_all (
    suffix VARCHAR2
) IS
    v_names user_names;
    v_quota VARCHAR2(40);
BEGIN
    v_names := get_users(suffix);
    dbms_output.put_line('V_NAMES.COUNT: ' || v_names.count);
    FOR i IN 1..v_names.count LOOP
        alter_quota(v_names(i));
    END LOOP;

    COMMIT;
END;

FUNCTION password_verify_function (
    username VARCHAR2,
    new_password VARCHAR2,
    old_password VARCHAR2
) RETURN BOOLEAN IS
BEGIN
    IF instr(
        new_password,
        '-'
    ) = 0 THEN
        raise_application_error(

```

```

        -20003,
        'Password must contain at least one underscore (_) character.'
    );
ELSE
    RETURN FALSE;
END IF;
END;

PROCEDURE create_profile (
    suffix VARCHAR2
) IS
    v_cnt                SMALLINT;
    v_profile_name        VARCHAR2(40);
    v_password_verify_function CONSTANT VARCHAR2(80) := '
PASSWORD_VERIFY_FUNCTION PASSWORD_VERIFY_FUNCTION_STANDALONE';
BEGIN
    v_profile_name := get_profile_name(suffix);
    drop_profile(v_profile_name);
    CASE v_profile_name
        WHEN bro_profile_public_general THEN
            EXECUTE IMMEDIATE 'CREATE PROFILE '
                                || v_profile_name
                                || ' LIMIT SESSIONS_PER_USER 6 IDLE_TIME 5
CONNECT_TIME 20 CPU_PER_CALL 6000 ';
        ELSE
            EXECUTE IMMEDIATE 'CREATE PROFILE '
                                || v_profile_name
                                || ' LIMIT SESSIONS_PER_USER 1 IDLE_TIME 15
PASSWORD_LIFE_TIME 90 FAILED_LOGIN_ATTEMPTS 5 '
                                || 'CPU_PER_CALL 12000 '
                                || v_password_verify_function;
    END CASE;
END;

PROCEDURE assign_profile (
    user_name VARCHAR2
) IS
    v_suffix        VARCHAR2(40);
    v_profile_name  VARCHAR2(40);
BEGIN
    v_suffix := is_valid_user_name(user_name);
    v_profile_name := get_profile_name(v_suffix);
    EXECUTE IMMEDIATE 'ALTER USER '
                        || user_name
                        || ' PROFILE '
                        || v_profile_name;
END;

PROCEDURE assign_profile_all (
    suffix VARCHAR2
) IS
    v_names        user_names;
    v_profile_name VARCHAR2(40);

```



```

BEGIN
    v_names := get_users(suffix);
    FOR i IN 1..v_names.count LOOP
        assign_profile(v_names(i));
    END LOOP;
END;

PROCEDURE configure_user_by_suffix (
    suffix          VARCHAR2,
    create_user     BOOLEAN := TRUE
) IS
    v_user_name     VARCHAR2(128);
BEGIN
    IF create_user THEN
        create_user_by_suffix(suffix);
        COMMIT;
    END IF;
    v_user_name := get_user_by_suffix(suffix);
    dbms_output.put_line('V_USER_NAME: ' || v_user_name);
    alter_quota(v_user_name);
    assign_profile(v_user_name);
    COMMIT;
END;

PROCEDURE create_bro_plan_rg IS
    v_suffixes          suffix_names := get_suffixes;
    v_user_names        user_names;
    v_exists_default_group SMALLINT;
    v_rg_prefix         VARCHAR2(40) := 'BRO_RG_';
BEGIN
    v_suffixes(v_suffixes.count + 1) := admin_suffix;
    dbms_resource_manager.create_pending_area();
    dbms_resource_manager.create_plan(
        plan                => bro_plan,
        comment             => 'Consumption plan for BRO users',
        active_sess_pool_mth => 'ACTIVE_SESS_POOL_ABSOLUTE',
        parallel_degree_limit_mth => 'PARALLEL_DEGREE_LIMIT_ABSOLUTE',
        queueing_mth        => 'FIFO_TIMEOUT',
        mgmt_mth            => 'EMPHASIS',
        sub_plan            => FALSE
    );
    FOR i IN 1..v_suffixes.count LOOP
        dbms_resource_manager.create_consumer_group(
            consumer_group => v_rg_prefix || v_suffixes(i),
            comment        => 'Consumer group for BRO '
                        || v_suffixes(i)
                        || ' users'
        );
    END LOOP;

    SELECT COUNT(*)
    INTO v_exists_default_group
    FROM dba_rsrc_consumer_groups

```

```

WHERE consumer_group = 'OTHER_GROUPS';
IF v_exists_default_group = 0 THEN
    dbms_resource_manager.create_consumer_group(
        consumer_group => 'OTHER_GROUPS',
        comment         => 'Default consumer group for all other users'
    );
END IF;

FOR i IN 1..v_suffixes.count LOOP
    IF ( v_suffixes(i) = admin_suffix ) THEN
        dbms_resource_manager.set_consumer_group_mapping(
            attribute     => dbms_resource_manager.oracle_user,
            value         => 'BRO_ADMIN',
            consumer_group => v_rg_prefix || v_suffixes(i)
        );
    ELSE
        v_user_names := get_users(v_suffixes(i));
        dbms_output.put_line('V_USER_NAMES.COUNT: ' || v_user_names.count);
        FOR j IN 1..v_user_names.count LOOP
            dbms_resource_manager.set_consumer_group_mapping(
                attribute     => dbms_resource_manager.oracle_user,
                value         => v_user_names(j),
                consumer_group => v_rg_prefix || v_suffixes(i)
            );
        END LOOP;
    END IF;
END LOOP;

-- doar mgmt_p1 ca nu avem subplanuri
dbms_resource_manager.create_plan_directive(
    plan             => bro_plan,
    group_or_subplan => 'OTHER_GROUPS',
    comment          => 'Default consumer group for all other users',
    mgmt_p1          => 5
);
dbms_resource_manager.create_plan_directive(
    plan             => bro_plan,
    group_or_subplan => v_rg_prefix || antrenor_suffix,
    comment          => 'Consumer group for BRO ANTRENOR users',
    mgmt_p1          => 20
);
dbms_resource_manager.create_plan_directive(
    plan             => bro_plan,
    group_or_subplan => v_rg_prefix || receptionist_suffix,
    comment          => 'Consumer group for BRO RECEPTIONIST users',
    mgmt_p1          => 15
);
dbms_resource_manager.create_plan_directive(
    plan             => bro_plan,
    group_or_subplan => v_rg_prefix || manager_suffix,
    comment          => 'Consumer group for BRO MANAGER users',
    mgmt_p1          => 15
);

```

```

);
dbms_resource_manager.create_plan_directive(
    plan          => bro_plan,
    group_or_subplan => v_rg_prefix || client_suffix,
    comment       => 'Consumer group for BRO CLIENT users',
    mgmt_p1       => 10
);
dbms_resource_manager.create_plan_directive(
    plan          => bro_plan,
    group_or_subplan => v_rg_prefix || public_suffix,
    comment       => 'Consumer group for BRO PUBLIC users',
    mgmt_p1       => 5
);
dbms_resource_manager.create_plan_directive(
    plan          => bro_plan,
    group_or_subplan => v_rg_prefix || admin_suffix,
    comment       => 'Consumer group for BRO ADMIN users',
    mgmt_p1       => 30
);
dbms_resource_manager.validate_pending_area();
dbms_resource_manager.submit_pending_area();
COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('Error: ' || sqlerrm);
        dbms_resource_manager.clear_pending_area();
        COMMIT;
END;

PROCEDURE clear_bro_plan_rg IS
    v_suffixes          suffix_names := get_suffixes;
    v_exists_default_group SMALLINT;
    v_rg_prefix          VARCHAR2(40) := 'BRO_RG_';
BEGIN
    v_suffixes(v_suffixes.count + 1) := admin_suffix;
    dbms_resource_manager.create_pending_area();
    BEGIN
        dbms_resource_manager.delete_plan(plan => bro_plan);
    EXCEPTION
        WHEN OTHERS THEN
            dbms_output.put_line('Error deleting plan: ' || sqlerrm);
    END;

    FOR i IN 1..v_suffixes.count LOOP
        BEGIN
            dbms_resource_manager.delete_consumer_group(consumer_group =>
v_rg_prefix || v_suffixes(i));
        EXCEPTION
            WHEN OTHERS THEN
                dbms_output.put_line('Error deleting consumer group: '
|| v_rg_prefix
|| v_suffixes(i)
|| ' - ');
        END;
    END LOOP;
END;

```

```

|| sqlerrm);

END;
END LOOP;

SELECT COUNT(*)
  INTO v_exists_default_group
  FROM dba_rsrc_consumer_groups
 WHERE consumer_group = 'OTHER_GROUPS';
IF v_exists_default_group > 0 THEN
  BEGIN
    dbms_resource_manager.delete_consumer_group(consumer_group =>
'OTHER_GROUPS');
  EXCEPTION
    WHEN OTHERS THEN
      dbms_output.put_line('Error deleting OTHER_GROUPS: ' || sqlerrm);
  END;
END IF;

dbms_resource_manager.validate_pending_area();
dbms_resource_manager.submit_pending_area();
dbms_output.put_line('All objects created by CREATE_BRO_PLAN_RG have been
cleared.');
```

```

  COMMIT;
EXCEPTION
  WHEN OTHERS THEN
    dbms_output.put_line('Error: ' || sqlerrm);
    dbms_resource_manager.clear_pending_area();
    COMMIT;
END;

PROCEDURE create_role (
  suffix  VARCHAR2,
  override BOOLEAN := FALSE
) IS
  v_cnt      SMALLINT;
  v_role_name VARCHAR2(40);
BEGIN
  is_valid_suffix(suffix);
  v_role_name := dbms_assert.simple_sql_name(upper(bro_role || suffix));
  SELECT COUNT(*)
    INTO v_cnt
    FROM dba_roles
   WHERE role = upper(v_role_name);
  dbms_output.put_line('NOT IMPLEMENTED');
```

```

END;

PROCEDURE assign_role (
  suffix VARCHAR2
) IS
  v_role_name VARCHAR2(40);
  v_users      user_names;
  v_cnt        SMALLINT;
```

```

BEGIN
    v_role_name := dbms_assert.simple_sql_name(upper(bro_role || suffix));
    v_users := get_users(suffix);
    SELECT COUNT(*)
    INTO v_cnt
    FROM dba_roles
    WHERE role = upper(v_role_name);
    dbms_output.put_line('NOT IMPLEMENTED');

END;
END bro_user_utils;
/

-- Table si functie care intoarce toti utilizatorii creati
-- pentru a permite adimnului access doar la cei
-- din cadrul aplicatiei noastre
CREATE OR REPLACE TYPE global_user_table AS
    TABLE OF VARCHAR2(128);
/

CREATE OR REPLACE FUNCTION get_users_by_suffix (
    suffix VARCHAR2
) RETURN global_user_table IS
    result_cursor      SYS_REFCURSOR;
    user_indexed_table bro_user_utils.user_names;
    user_nested_table  global_user_table := global_user_table();
BEGIN
    user_indexed_table := bro_user_utils.get_users(suffix);
    FOR i IN user_indexed_table.first..user_indexed_table.last LOOP
        user_nested_table.extend;
        user_nested_table(user_nested_table.count) := user_indexed_table(i);
    END LOOP;
    RETURN user_nested_table;
END;
/

CREATE USER bro_admin IDENTIFIED BY bro_admin
    PASSWORD EXPIRE;

GRANT
    CREATE SESSION
TO bro_admin;
GRANT
    CREATE ANY TABLE
TO bro_admin;
GRANT
    CREATE ANY VIEW
TO bro_admin;
GRANT
    CREATE ANY TRIGGER
TO bro_admin;
GRANT
    CREATE ANY PROCEDURE

```

```

TO bro_admin;

GRANT
    CREATE ANY SEQUENCE
TO bro_admin;

GRANT
    CREATE ANY INDEX
TO bro_admin;

GRANT
    CREATE ANY TYPE
TO bro_admin;

GRANT
    CREATE TYPE
TO bro_admin;
GRANT EXECUTE ON global_user_table TO bro_admin;

-- Pt proceduri
GRANT EXECUTE ON dbms_crypto TO bro_admin WITH GRANT OPTION;
--Pt generated by default on null as identity la create in antrenor
GRANT
    SELECT ANY SEQUENCE
TO bro_admin;
GRANT EXECUTE ON get_users_by_suffix TO bro_admin;

--Profilul adminului
CREATE PROFILE bro_profile_admin LIMIT
    IDLE_TIME 15
    PASSWORD_LIFE_TIME 90
    FAILED_LOGIN_ATTEMPTS 5
    CPU_PER_CALL 36000
    PASSWORD_VERIFY_FUNCTION password_verify_function_standalone;

ALTER USER bro_admin
    PROFILE bro_profile_admin;

ALTER USER bro_admin
    QUOTA 500M ON users;

-- Creare profilurilor pentru restul de utilizatori
exec BRO_USER_UTILS.CREATE_PROFILE(BRO_USER_UTILS.PUBLIC_SUFFIX);
exec BRO_USER_UTILS.CREATE_PROFILE(BRO_USER_UTILS.ANTRENOR_SUFFIX);
exec BRO_USER_UTILS.CREATE_PROFILE(BRO_USER_UTILS.RECEPTIONIST_SUFFIX);
exec BRO_USER_UTILS.CREATE_PROFILE(BRO_USER_UTILS.MANAGER_SUFFIX);
exec BRO_USER_UTILS.CREATE_PROFILE(BRO_USER_UTILS.CLIENT_SUFFIX);

-- public general
exec BRO_USER_UTILS.CONFIGURE_USER_BY_SUFFIX(BRO_USER_UTILS.PUBLIC_SUFFIX,
TRUE);
--manager filiala
exec BRO_USER_UTILS.CONFIGURE_USER_BY_SUFFIX(BRO_USER_UTILS.MANAGER_SUFFIX,
TRUE);

```

```

--antrenor
BEGIN
    FOR i IN 1..7 LOOP
        bro_user_utils.configure_user_by_suffix(
            bro_user_utils.antrenor_suffix,
            TRUE
        );
    END LOOP;
END;
/
--receptionist
BEGIN
    FOR i IN 1..10 LOOP
        bro_user_utils.configure_user_by_suffix(
            bro_user_utils.receptionist_suffix,
            TRUE
        );
    END LOOP;
END;
/
--client
BEGIN
    FOR i IN 1..10 LOOP
        bro_user_utils.configure_user_by_suffix(
            bro_user_utils.client_suffix,
            TRUE
        );
    END LOOP;
END;
/

-- planul de resurse
exec BRO_USER_UTILS.CREATE_BRO_PLAN_RG;

SELECT username,
       initial_rsrc_consumer_group
FROM   dba_users
WHERE  username LIKE upper('bro_%');

SELECT DISTINCT u.username, u.profile, p.group_or_subplan, p.mgmt_p1, p.plan
FROM   dba_users u JOIN dba_rsrc_plan_directives p
ON     u.initial_rsrc_consumer_group=p.group_or_subplan
WHERE  username LIKE upper('bro_%');
SELECT DISTINCT
       u.username,
       u.profile,
       p.group_or_subplan,
       p.mgmt_p1,
       p.plan,
       r.granted_role
FROM   dba_users u
LEFT JOIN dba_rsrc_plan_directives p

```

```

        ON u.initial_rsrc_consumer_group = p.group_or_subplan
LEFT JOIN dba_role_privs r
        ON u.username = r.grantee
WHERE
    u.username LIKE upper('bro_%')
ORDER BY
    u.username, r.granted_role;

-- Restul de privilegii si roluri vor fi data dupa ce admin creeaza tot
-- inclusiv in antrenori etc

```

### 10.7.7 sys\_users\_2.sql

```

--SYS
ALTER SESSION SET container = orclpdb;
-- Dupa ce bro_admin a create tabele si a si inserat datele
-- Dupa ce bro_admin a creat tabele si a si inserat datele pt bro_antrenor1..n
CREATE ROLE r_bro_public_general;
GRANT
    CREATE SESSION
TO r_bro_public_general;
GRANT SELECT ON bro_admin.antrenor_extins TO r_bro_public_general;
GRANT SELECT ON bro_admin.filiala TO r_bro_public_general;
GRANT SELECT ON bro_admin.adresa TO r_bro_public_general;
GRANT SELECT ON bro_admin.supliment TO r_bro_public_general;

GRANT SELECT ON bro_admin.echipament TO r_bro_public_general;
GRANT SELECT ON bro_antrenor1.program TO r_bro_public_general;
GRANT SELECT ON bro_antrenor2.program TO r_bro_public_general;
GRANT SELECT ON bro_antrenor3.program TO r_bro_public_general;
GRANT r_bro_public_general TO bro_public_general1;

-- roluri
-- antrenor
CREATE ROLE r_bro_antrenor;
GRANT r_bro_public_general TO r_bro_antrenor;
GRANT SELECT ON bro_admin.client_extins TO r_bro_antrenor;
GRANT SELECT ON bro_admin.telefon TO r_bro_antrenor;
GRANT SELECT ON bro_admin.antrenor TO r_bro_antrenor;
GRANT SELECT ON bro_admin.chei_client TO r_bro_antrenor;
GRANT EXECUTE ON bro_admin.select_random_from_nr_list TO r_bro_antrenor;
GRANT
    CREATE TABLE,
    CREATE VIEW,
    CREATE SEQUENCE,
    CREATE PROCEDURE,
    CREATE TYPE
TO r_bro_antrenor;
GRANT EXECUTE ON dbms_crypto TO r_bro_antrenor;

```



```

GRANT r_bro_antrenor TO bro_antrenor1;
GRANT r_bro_antrenor TO bro_antrenor2;
GRANT r_bro_antrenor TO bro_antrenor3;

-- client
CREATE ROLE r_bro_client;
GRANT r_bro_public_general TO r_bro_client;
GRANT SELECT ON bro_admin.client_extins TO r_bro_client;
GRANT EXECUTE ON bro_admin.get_client_key TO r_bro_client;

-- fiecare antrenor da la clientii sai
GRANT EXECUTE ON bro_antrenor1.fetch_decrypted_client_data TO bro_client1;
GRANT EXECUTE ON bro_antrenor1.hash_checksum TO bro_client1;
GRANT EXECUTE ON bro_antrenor1.fetch_decrypted_client_data TO bro_client2;
GRANT EXECUTE ON bro_antrenor1.hash_checksum TO bro_client2;
GRANT EXECUTE ON bro_antrenor1.fetch_decrypted_client_data TO bro_client3;
GRANT EXECUTE ON bro_antrenor1.hash_checksum TO bro_client3;
--programul e public

GRANT r_bro_client TO bro_client1;
GRANT r_bro_client TO bro_client2;
GRANT r_bro_client TO bro_client3;

--receptionist
CREATE ROLE r_bro_receptionist;
GRANT r_bro_public_general TO r_bro_receptionist;
GRANT SELECT,INSERT,UPDATE ON bro_admin.client_extins TO r_bro_receptionist;
GRANT SELECT,INSERT,UPDATE ON bro_admin.telefon TO r_bro_receptionist;
GRANT SELECT,INSERT,UPDATE ON bro_admin.abonament TO r_bro_receptionist;
GRANT SELECT ON bro_admin.tip_abonament TO r_bro_receptionist;
GRANT SELECT ON bro_admin.furnizor TO r_bro_receptionist;
GRANT INSERT ON bro_admin.comanda TO r_bro_receptionist;
GRANT INSERT ON bro_admin.informatii_comanda TO r_bro_receptionist;
GRANT SELECT ON bro_admin.aprovizionare TO r_bro_receptionist;
--programul e public
GRANT r_bro_receptionist TO bro_receptionist1;
GRANT r_bro_receptionist TO bro_receptionist2;
GRANT r_bro_receptionist TO bro_receptionist3;

--manager filiala
CREATE ROLE r_bro_manager_filiala;
GRANT r_bro_public_general TO r_bro_manager_filiala;
GRANT SELECT ON bro_admin.receptionist_extins TO r_bro_manager_filiala;
GRANT SELECT ON bro_admin.client_extins TO r_bro_manager_filiala;
GRANT SELECT ON bro_admin.furnizor TO r_bro_manager_filiala;

GRANT SELECT ON bro_antrenor1.program TO r_bro_manager_filiala;
--pt fiecare antrenor in filiala sa
GRANT SELECT ON bro_antrenor1.antrenament TO r_bro_manager_filiala;
GRANT SELECT ON bro_antrenor2.antrenament TO r_bro_manager_filiala;
GRANT SELECT ON bro_antrenor3.antrenament TO r_bro_manager_filiala;

```

```
GRANT SELECT,UPDATE,INSERT,DELETE ON bro_admin.echipament TO  
r_bro_manager_filiala;
```

```
GRANT SELECT ON bro_admin.aprovizionare TO r_bro_manager_filiala;  
GRANT r_bro_manager_filiala TO bro_manager_filiala1;
```

## 11. Codul CMD

### 11.1. import\_mask\_person.cmd

```
@echo off  
impdp bro_import/bro_import@//localhost:1522/orclpdb ^  
remap_table=persoana:persoana_mask ^  
remap_table=angajat:angajat_mask ^  
remap_table=antrenor:antrenor_mask ^  
remap_table=receptionist:receptionist_mask ^  
remap_table=client:client_mask ^  
remap_schema=bro_admin:bro_import ^  
directory=MASK_DUMP ^  
dumpfile=mask_person.dmp ^  
logfile=mask_person_import.log ^  
parallel=8 ^  
transform=disable_archive_logging:y  
echo Done importing mask persoana  
exit /b 0
```

### 11.2. mask\_person.cmd

```
@echo off  
expdp bro_admin/bro_admin@//localhost:1522/orclpdb ^  
tables=BRO_ADMIN.PERSOANA, BRO_ADMIN.ANGAJAT, BRO_ADMIN.ANTRENOR,  
BRO_ADMIN.RECEPTIONIST, BRO_ADMIN.CLIENT ^  
remap_data=persoana.id_persoana:mask_person.mask_person_id ^  
remap_data=persoana.nume:mask_person.mask_item ^  
remap_data=persoana.prenume:mask_person.mask_item ^  
remap_data=persoana.email:mask_person.mask_item ^  
remap_data=persoana.varsta:mask_person.mask_item ^  
remap_data=angajat.id_angajat:mask_person.mask_person_fk ^  
remap_data=angajat.salariu:mask_person.mask_item ^  
remap_data=angajat.id_meneger:mask_person.mask_person_fk ^  
remap_data=antrenor.id_antrenor:mask_person.mask_person_fk ^  
remap_data=receptionist.id_receptionist:mask_person.mask_person_fk ^  
remap_data=client.id_client:mask_person.mask_person_fk ^  
directory=MASK_DUMP parallel=8 dumpfile=mask_person.dmp logfile=mask_person.log  
reuse_dumpfiles=y  
echo Done exporting mask persoana  
exit /b 0
```

### 11.3. seed\_antrenor.cmd

```
@echo off

set
"SQL_SCRIPT_PATH=C:\Master\An2\sem1\securitateaBD\proiect\sql\bro_admin_antreno
r_seed.sql"
sqlplus bro_admin/bro_admin@//localhost:1522/orclpdb @"SQL_SCRIPT_PATH%
echo Done seeding antrenor table
exit /b 0
```

## 12. Link repository

Proiectul de posate găsi pe github: <https://github.com/MocicaRazvan/sbdProiect>