

# **Proiect Securitatea Bazelor De Date**

**Mocică Răzvan-Cristian**

**Grupa 505**

## Cuprins

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 aplicatiei .....	37
10.1. Admin .....	37
10.1.2. bro_admin_audit.sql .....	49
10.1.3. bro_admin_create_tables.sql .....	54
10.1.4. bro_admin_criptare.sql .....	136
10.1.5. bro_admin_mask.sql .....	141
10.1.6. bro_admin_programs_view.sql .....	151
10.1.7. bro_admin_update_echipament_fals.sql .....	153
10.2 Antrenor .....	154
10.2.1. bro_antrenor_insert.sql .....	154
10.2.2. bro_antrenor1_cript_show.sql .....	165
10.3. Client .....	174
10.3.1. bro_client1_select_cript.sql .....	174
10.3.2 bro_client1_sql_injection.sql .....	176
10.4.1. bro_import.sql .....	177
10.5 Manager .....	178
10.5.1. bro_manager_filiala1_context.sql .....	178
10.7. SYS .....	180
10.7.1. sys_admin_antrenor_privilege.sql .....	180
10.7.2. sys_audit_1.sql .....	181
10.7.3. sys_audit_2.sql .....	189
10.7.4 sys_context.sql .....	193
10.7.5. sys_mask.sql .....	199
10.7.6. sys_users_1.sql .....	200
11. Codul CMD .....	238

11.1. import_mask_person.cmd .....	238
11.2. mask_person.cmd .....	238
11.3. seed_antrenor.cmd .....	239
12. Link repository .....	239

## 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 ).

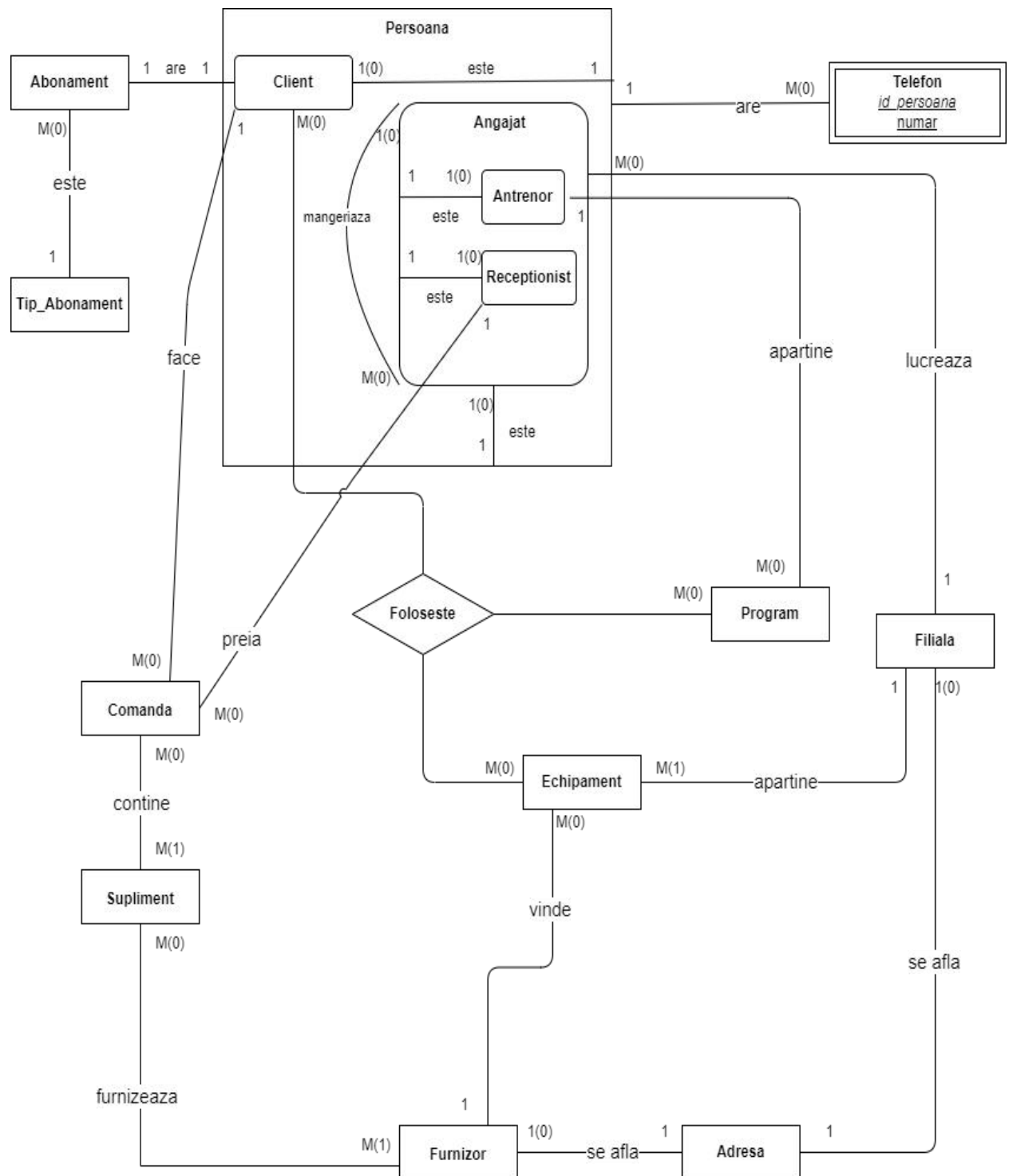
O aprovizionare a firmei se va face cu un singur supliment o dată.

Nu se înregistrează stocuri de suplimente.

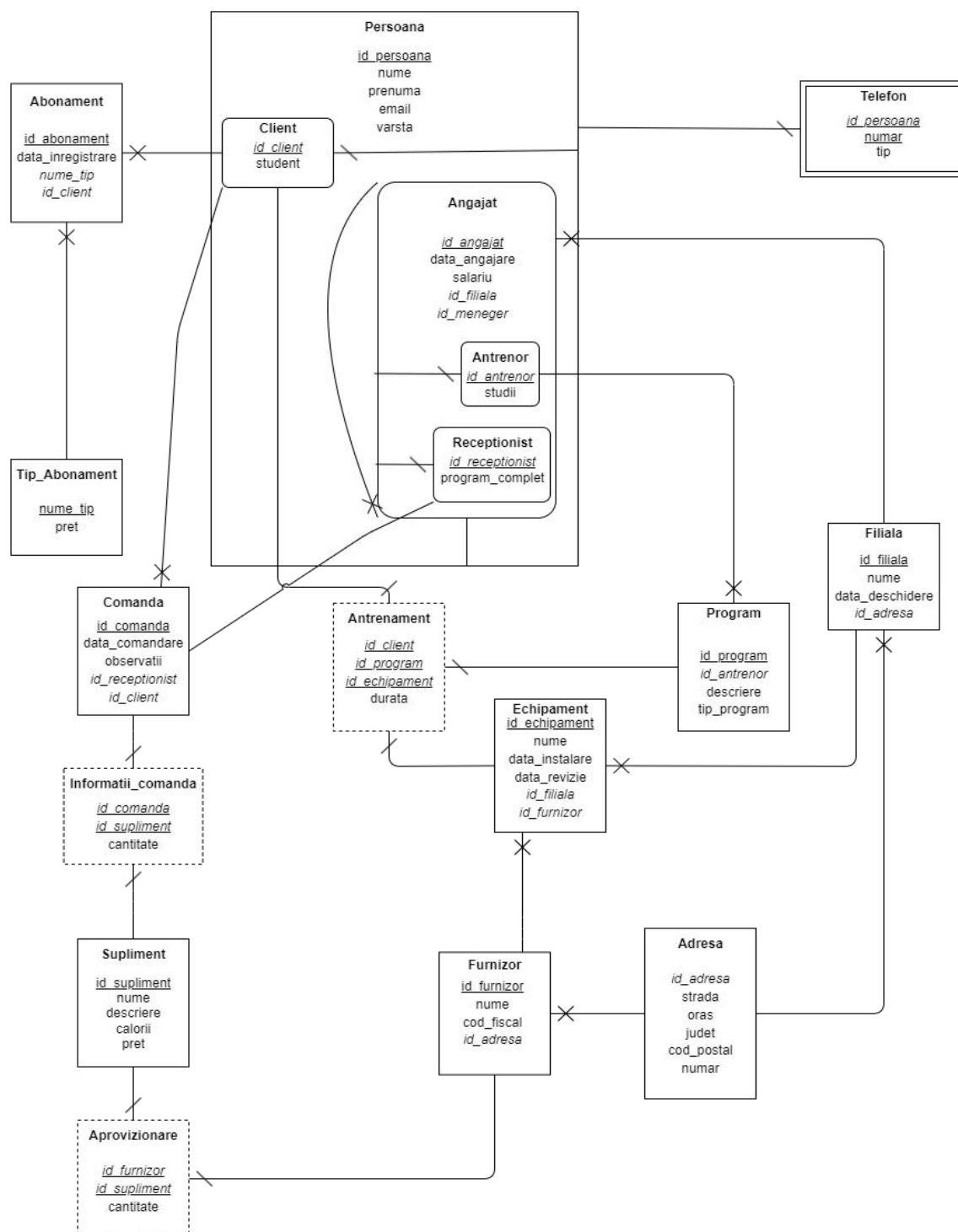
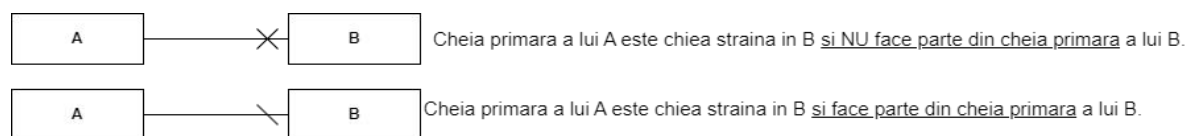
Filialele trebuie să aibă adresele diferite între ele.

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

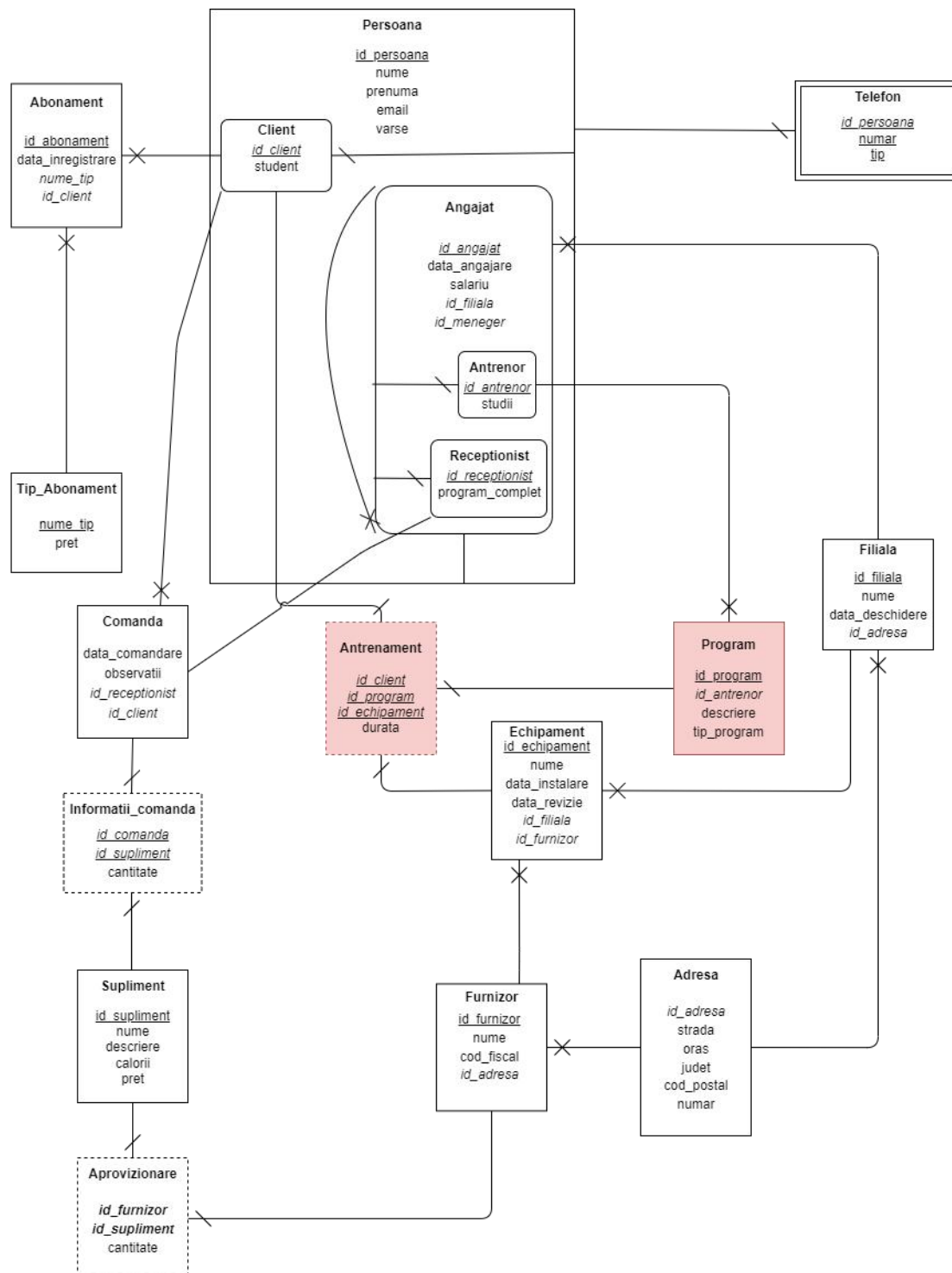
## 2.3. Matricea Entitate-Utilizator

	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.

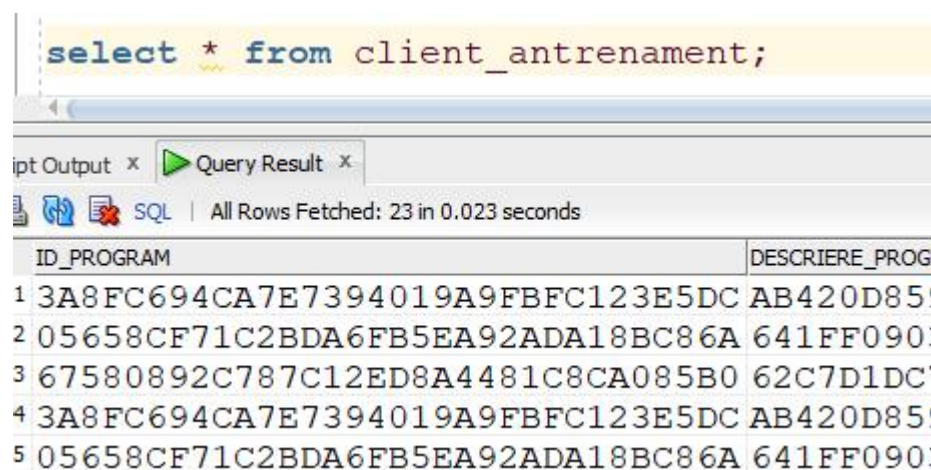
Commit complete.
```

### 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 hexadecimal strings.

	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_REVIZII_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	8CDBDB274D69EB6EC5F
11	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	C4A969FDBA2656B331E
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	138005AFCF87D533A3E
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	9C2FF44A3B3A8E3B0C5
20	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed	FD1B9062D83FB570CAE
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_REVIZII_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	5AA001D9A8B5FCB570B99B3B368FE90
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	5AA001D9A885FCE5708D9BE3E39EF905
11	Push, Full Legs Light, for beginners	MASS	11	2	Chest Press	20-JUN-21	20-JUN-21	1	18	D9AD072B90783A7CDD499B97487135E0D
14	Full Body Variant Light	CARDIO	10	2	Chest Press	20-JUN-21	20-JUN-21	1	18	D5FB4661AE81EC662404FEB981768A97D
44	Full Body Variant Light	CARDIO	11	1	Leg Press	20-MAY-20	20-MAY-21	1	18	A83E3CF683F178F56F199494344EB5A0A

## 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

```

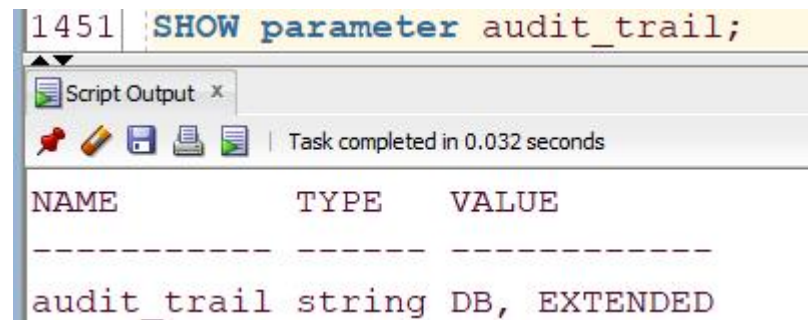
ANTRENOR	ID_PROGRAM	DESCRIERE	TIP_PROGRAM
1 BRO ANTRENOR2	1	Push, Pull Legs Light, for beginners	MASS
2 BRO ANTRENOR2	2	Push, Pull Legs Medium	MASS
3 BRO ANTRENOR2	3	Push, Pull Legs Hard	MASS
4 BRO ANTRENOR2	4	Full Body Variant Light	CARDIO
5 BRO ANTRENOR2	5	Body Recovery Variant Light	RECOVERY
6 BRO 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 salavare 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:

```

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

```

Script Output x Query Result x

SQL | All Rows Fetched: 20 in 0.012 seconds

	SESSIONID	ENTRYID	STATEMENT	TIMESTAMP#	USERID	USERHOST	
1	21010	18	73 (null)	BRO ADMIN Galma-ROG u			
2	21010	19	73 (null)	BRO ADMIN Galma-ROG u			
3	21010	20	73 (null)	BRO ADMIN Galma-ROG u			
4	21010	21	73 (null)	BRO ADMIN Galma-ROG u			
5	21010	2	73 (null)	BRO ADMIN Galma-ROG u			
6	21010	3	73 (null)	BRO ADMIN Galma-ROG u			
7	21010	4	73 (null)	BRO ADMIN Galma-ROG u			

```

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

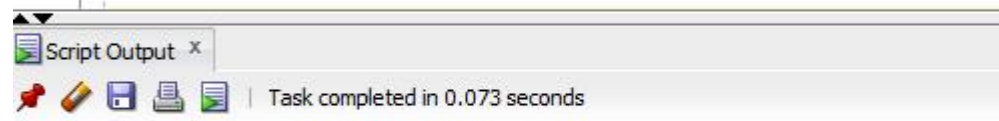
```
16 select a.old_values.data_revizie, a.new_values.data_revizie from audit_echipament a;
```

	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.

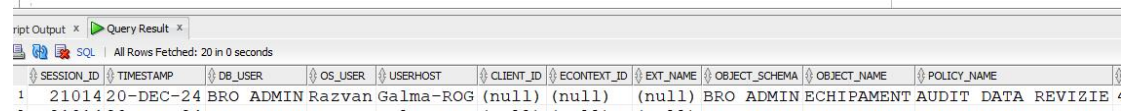
Procedure BRO\_AUDIT\_TABLESE\_HANDLER compiled

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
```

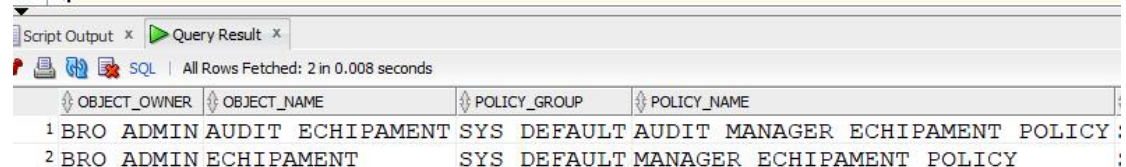
```
FGA Triggered:
Timestamp: 2025-01-13 15:08:47
Object Schema: BRO_ADMIN
Object Name: ECHIPAMENT
Policy Name: AUDIT_DATA_REVIZIE
```



## 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`:

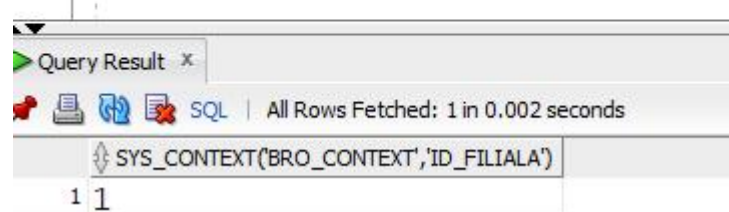
```
30 |
31 | SELECT *
32 | FROM dba_policies
33 | where object_owner like upper('bro%');
34 |
```



	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;
```

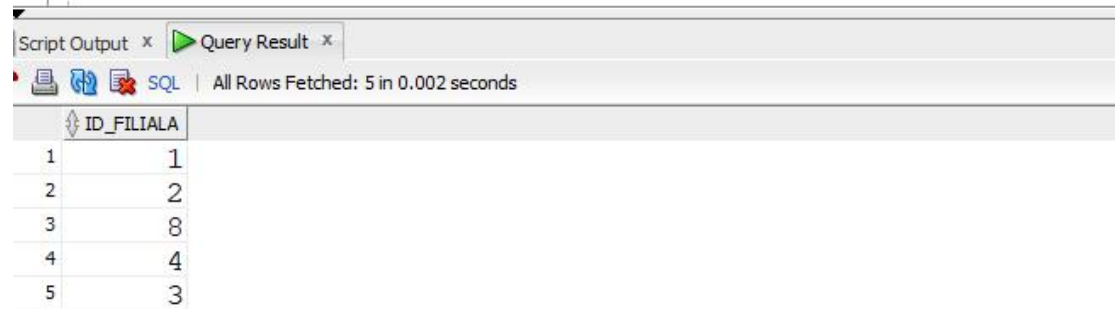


SYS_CONTEXT('BRO_CONTEXT','ID_FILIALA')
1

## 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*.

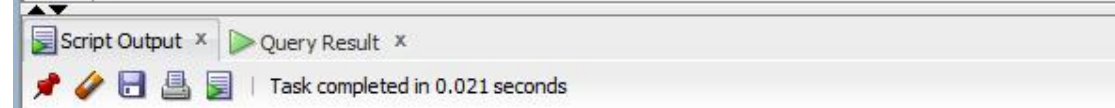
```
6 |
7 | select distinct id_filiala from bro_admin.echipament;
8 |
```



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 'Query Result' 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 = '

```

```

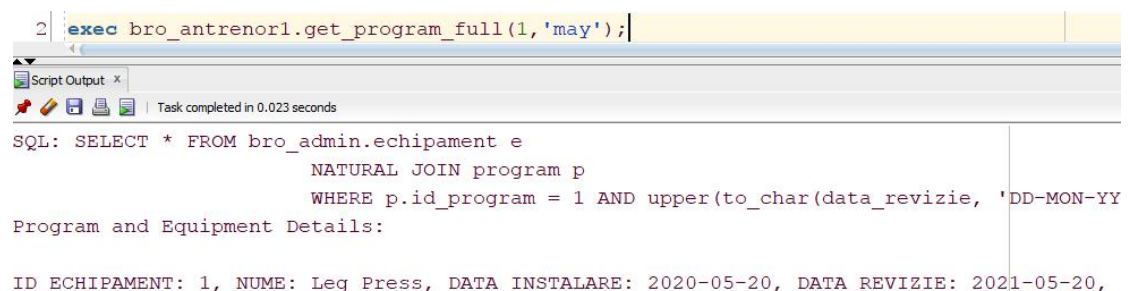
|| id_prg ||
' AND upper(to_char(data_revizie, 'DD-MON-YY')) LIKE '%'
|| upper(data_inst)||
'%'

```

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

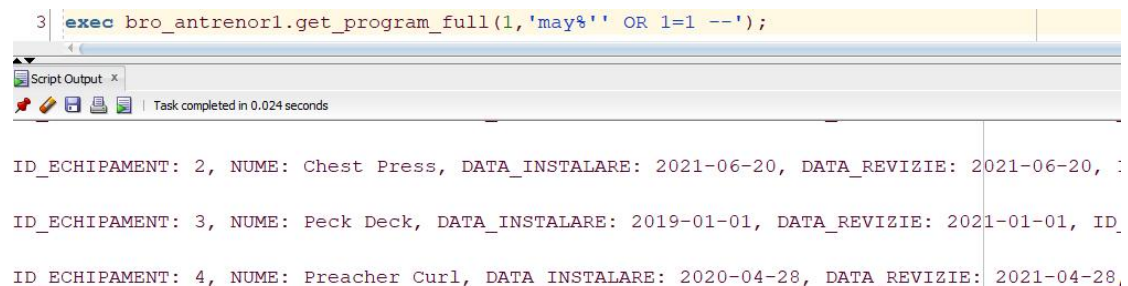
Task completed in 0.023 seconds

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 '%' || upper(data\_inst)|| '%'

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

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

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'

```

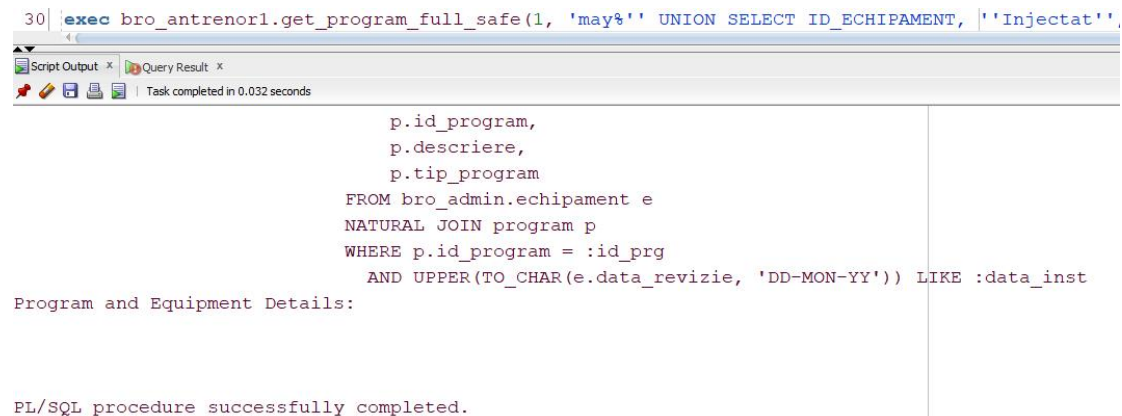
Iar, apelarea sa va fi urmatoarea:

```

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_admin: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

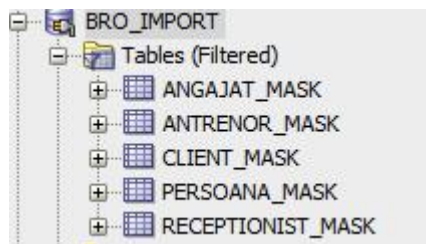
Processing object type TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/MARKER
Job "BRO_IMPORT"."SYS_IMPORT_FULL_01" completed with 1 error(s) at Mon Jan 13 17:44:16 2025 elapsed 0 00:00:06

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_inl8n.raw_to_char(

        dbms_crypto.decrypt(

            cript,

            mod_op,

            cheie

        ),

        'AL32UTF8'

    );

end;

```

```

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**

```

SET SERVEROUTPUT ON;

-- Crearea tabelelor si inserarea datelor initiale in schema bro_admin

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_supliment_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(regex_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.num, 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.nume, :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.nume, 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.nume, :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 ( 'Anastasiu 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',

        'popescul@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',

```

```

        'aioaneiaandrei@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_revizie,
```

```
    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(
```

```
            10000000000,
```

```
            99999999999
```

```
        ))
```

```
        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;
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
    ),
    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 tamperie 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;

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_revizie = (

            select data_revizie
```

```

        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_echipament,
    id_client
) values ( 10,
    4,
    4,
    22 );

insert into antrenament (
    durata,
    id_program,
    id_echipament,
    id_client
) values ( 10,
    7,
    8,
    25 );

```

```
insert into antrenament (
```

```
    durata,
```

```
    id_program,
```

```
    id_echipament,
```

```
    id_client
```

```
) values ( 20,
```

```
    7,
```

```
    9,
```

```
    25 );
```

```
insert into antrenament (
```

```
    durata,
```

```
    id_program,
```

```
    id_echipament,
```

```
    id_client
```

```
) values ( 10,
```

```
    7,
```

```
    10,
```

```
    25 );
```

```
insert into antrenament (
```

```
    durata,
```

```
    id_program,
```

```
    id_echipament,
```

```
    id_client
```

```
) values ( 10,
```

```

        8,
        3,
        26 );

insert into antrenament (
    durata,
    id_program,
    id_echipament,
    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 decrypt_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,
```

```

decript_string(
    ca.durata_antrenament,
    c.mod_op,
    c.cheie
) as durata_antrenament,
decript_string(
    ca.id_echipament,
    c.mod_op,
    c.cheie
) as id_echipament,
decript_string(
    ca.nume_echipament,
    c.mod_op,
    c.cheie
) as nume_echipament,
decript_string(
    ca.data_instalare_echipament,
    c.mod_op,
    c.cheie
) as data_instalare_echipament,
decript_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.nume,

        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 ||
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') ||
to_char(v_program_echipament(i).data_instalare, 'YYYY-MM-DD') ||
        ', DATA_REVIZIE: ' || to_char(v_program_echipament(i).data_revizie, 'YYYY-MM-DD') ||
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.desc

riere_program,ant.tip_program,

      ant.durata_antrenament,ant.id_echipament,ant.nume_echipament,

ant.data_instalare_echipament,ant.data_revizie_echipament,ant.id_filiala,

      user_key.client_key.cheie)) as cur_cs from dual

) cs;

select * from bro_admin.programs_view;

```

### 10.3.2 bro\_client1\_sql\_injection.sql

```
set serveroutput on;
```

```
-- apel onest
```

```
exec bro_antrenor1.get_program_full(1,'may');
```

```
--apel care intoarce toate programele cu echipamente, subminand filtrarea
```

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

```
--apel care intoarce toate antrenamentele, desi clientul nu are drept de select pe tabela antrenament
```

```
select * from bro_antrenor1.antrenament;
```

```
exec      bro_antrenor1.get_program_full(1,      'may%'      UNION      SELECT  
ID_ECHIPAMENT, "Injectat", SYSDATE, SYSDATE, ID_CLIENT, DURATA,  
ID_PROGRAM, "Injectat Desc", "Tip injectat" FROM ANTRENAMENT --');
```

```
begin
```

```
      bro_antrenor1.get_program_full(1, 'may%' UNION SELECT ID_ECHIPAMENT,  
"Injectat",
```

```
      SYSDATE,      SYSDATE,      ID_CLIENT,      DURATA,  
ID_PROGRAM,
```

```
      "Injectat Desc", "Tip injectat" FROM ANTRENAMENT --');
```

```
end;
```

```
/
```

-- repetam cu safe

-- apel onest

exec bro\_antrenor1.get\_program\_full\_safe(1,'may');

--apel care intoarce toate programele cu echipamente, subminand filtrarea

exec bro\_antrenor1.get\_program\_full\_safe(1,'may%' OR 1=1 --');

exec bro\_antrenor1.get\_program\_full\_safe(1, 'may%' UNION SELECT  
ID\_ECHIPAMENT, "Injectat", SYSDATE, SYSDATE, ID\_CLIENT, DURATA,  
ID\_PROGRAM, "Injectat Desc", "Tip injectat" FROM ANTRENAMENT --');

begin

bro\_antrenor1.get\_program\_full\_safe(1, 'may%' UNION SELECT  
ID\_ECHIPAMENT, "Injectat",  
SYSDATE, SYSDATE, ID\_CLIENT, DURATA,  
ID\_PROGRAM,  
"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;

--creare 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 utiliztar 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,
    ovveride  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_]+$'

    ) 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;

```

```

        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)

                || ' - '

                || 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,

    ovveride 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 utilizaotrii 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_pl,

    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;

```

### **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.numa: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_antrenor_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>