

Gestiunea unui Magazin de Muzica

Facultatea de Matematica si Informatica

Baze de Date

Grupa: 141

Rachieriu Gheorghe Gabriel

Contents

1 Descrierea modelului real	3
1.1 Utilitatea sistemului de baza de date	3
2 Prezentarea constrangerilor	3
3 Descrierea entitatilor	3
4 Descrierea relatiilor	4
4.1 Relatiile de tip One-To-Many si Many-To-Many	4
4.2 Relatia de tip 3	4
5 Descrierea atributelor	5
6 Realizarea diagramei entitate-relatie	11
7 Realizarea diagramei conceptuale	12
8 Scheme relationale	12
9 Realizarea normalizarii FN1-FN3	13
9.1 Forma normala 1 - FN1	13
9.2 Forma normala 2 - FN2	13
9.3 Forma normala 3 - FN3	14
10 Crearea secentelor utilizate in inserarea inregistrarilor	15
10.1 CASA DE DISCURI	15
10.2 PRODUS	15
10.3 GEN	15
10.4 ARTIST	15
10.5 FURNIZOR	15
10.6 FORMAT MEDIA	15
10.7 FUNCTIE	15
10.8 ANGAJAT	16
10.9 CUMPARATOR	16
10.10STOC	16
10.11ACHIZITIE	16
10.12MELODIE	16
11 Creearea tabelelor SQL si inserarea datelor	16
11.1 CASA DE DISCURI	16
11.2 PRODUS	17
11.3 GEN	17
11.4 ALBUM	17
11.5 MERCHANDISE	18
11.6 ARTIST	18
11.7 FURNIZOR	19
11.8 FORMAT MEDIA	19
11.9 FUNCTIE	20
11.10ANGAJAT	20
11.11CUMPARATOR	20
11.12STOC	21
11.13ACHIZITIE	21
11.14MELODIE	22

12 Cereri SQL	24
12.1 Cererea nr. 1 - Cererea din feedback	24
12.2 Cererea nr. 2 - Cereri sincronizate cu 3 tabele	25
12.3 Cererea nr. 3 - Subcereri nesincronizare in FROM	26
12.4 Cererea nr.4 - Grupari cu subcereri in HAVING	27
12.5 Cererea nr. 5 - NVL, DECODE, Sortari	28
12.6 Cererea nr. 6 - Clauza WITH	29
13 Cereri de actualizare si suprimare a datelor	30
13.1 Cererea nr.1	30
13.2 Cererea nr. 2	31
13.3 Cererea nr. 3	32
14 Tabele view si operatii LMD	33
15 Cereri SQL folosind operatiile, outer join, division si top-n	37
15.1 Outer join	37
15.2 Division	38
15.3 Analiza Top-n	39
16 Optimizari	39
16.1 Cererea	39
16.2 Cererea SQL neoptimizata	39
16.3 Cererea SQL optimizata	40
16.4 Expresii algebrice	40
16.5 Optimizare	40
16.6 Arbore algebric	41
17 Realizarea normalizarii BCNF, FN4, FN5 si aplicarea denormalizarii	41
17.1 Forma normala Boyce-Codd	41
17.2 Formala normala 4 - FN4	42
17.3 Forma normala 5 -FN5	43
17.4 Aplicarea denormalizarii	45
18 Exemplificare Isolation Levels	45
18.1 Read Committed	45
18.2 Serializable	45
18.3 Read only	46
19 Migrarea la o baza de date de tip NoSQL	46
19.1 Prezentarea structurii bazei de date NoSQL	46
19.2 Prezentare comenzi pentru crearea bazei de date	47
19.3 Prezentarea comenzilor pentru inserare	48

1 Descrierea modelului real

Pe zi ce trece, toate afacerile, activitatile și acțiunile iau ampoare. Același fenomen se poate observa și în ceea ce privește domeniul muzical. Produsele muzicale devin tot mai diverse și sunt prezentate în atât de multe forme, iar numărul de producători este într-o creștere continuă și, tocmai de aceea, crearea unei baze de date este necesară pentru retinerea acestor informații. Acest proiect a fost creat cu scopul de a gestiona toate datele necesare organizării activității unui magazin de muzica, numit "Rachi Records".

Totodată, am ales aceasta temă deoarece vreau să îmi aprofundez cunoștințele în domeniul bazelor de date, cu precadere în sistemul de gestiune al bazelor de date Oracle. O baza de date bine structurată pentru "Rachi Records" ar permite gestionarea eficientă a stocului de muzica, urmarirea vanzarilor, analiza preferințelor clientilor și automatizarea proceselor operaționale. Astfel, magazinul poate oferi o experiență mai bună clientilor și poate crește eficiența afacerii în ansamblu.

1.1 Utilitatea sistemului de baza de date

Proiectul "Gestiunea unui magazin de muzica" este o baza de date care monitorizează vânzarea de produse din stocul unui magazin, acestea fiind achiziționate de la diferiți furnizori. Scopul principal al acesteia este de a furniza informații precise și actualizate.

Cu ajutorul bazei de date, sunt înregistrate următoarele informații: detalii despre clienti, comenzi plasate de aceștia, produsele achiziționate de clienti, locația fizică a produselor în magazin, detalii despre angajații fiecarei locații și informații despre furnizorii de produse. Astfel, aceasta baza de date asigură o gestionare eficientă a activităților magazinului de muzica și contribuie la îmbunătățirea experienței clientilor.

2 Prezentarea constrangerilor

- Pretul de baza al unui produs îl este adăugat "adaos" în funcție de forma lui media.
- O casă de discuri face mai multe albume sau niciunul.
- Un album are mai multe genuri muzicale și cel puțin unul.
- Un produs, fie el Album sau Merchandise, este asociat mai multor melodii și cel puțin una.
- Mai mulți artiști pot înregistra o singură melodie.
- Un album este disponibil în mai multe formate media și cel puțin unul.
- Un produs apare în mai multe stocuri sau în niciunul.
- Un furnizor trimite mai multe stocuri, sau niciunul.
- Un Angajat are o singură funcție și cel puțin una.
- O comandă este realizată de cel puțin un angajat.
- Un Cumpărător a efectuat mai multe comenzi, și cel puțin una.
- O achiziție are un singur produs, cantitatea acestuia, clientul și angajatul care se ocupă de comandă.
- Un angajat se ocupă de mai multe comenzi sau de niciuna.
- Un client are mai multe achiziții, sau niciuna, fiind înscris în baza de date fără a cumpăra ceva anterior.
- Un produs poate apărea în mai multe achiziții, sau în niciuna.

3 Descrierea entităților

- Pentru fiecare casă de discuri se cunosc id-ul casei, numele casei și adresa
- Pentru fiecare produs de tip Merchandise se cunoaște id-ul, tipul, numele, și optional marimea.
- Pentru fiecare album se cunosc id-ul albumului, denumirea albumului, data apariției, lungimea și pretul de baza.

- Pentru fiecare furnizor se cunosc id-ul furnizorului si denumirea furnizorului
- Pentru fiecare format media se cunosc id-ul formatului si denumirea si adaosul.
- Pentru fiecare artist se cunoaste id-ul artistului, numarul, numele artistului, prenumele artistului si grupul muzical din care face parte.
- Pentru fiecare client se cunosc id-ul clientului, numele si prenumele, numarul de telefon, adresa si localitatea.
- Pentru fiecare angajat se cunosc id-ul angajatului, numele si prenumele, numarul de telefon, adresa, data angajarii si salariu.
- Pentru fiecare angajat se cunosc id-ul angajatului, numele si prenumele, numarul de telefon, adresa, data angajarii si salariu.
- Pentru fiecare functie se cunoaste id-ul functiei, denumirea, salariul minim si maxim.
- Pentru fiecare gen muzical se cunosc id-ul genului si numele.

4 Descrierea relatiilor

4.1 Relatiile de tip One-To-Many si Many-To-Many

Relatie	Cardinalitate	Observatii
Producere	Casa De Discuri - Album: One to Many	-
Exista in	Album - Format Media: One to Many	-
Are/Inregistreaza	Produs - Artist: Many to Many	-
Distribuie	Furnizor - Produs: Many to Many	-
Apartine	Produs - Gen: One to Many	-
Are o	Angajat - Functie: One to Many	-

4.2 Relatia de tip 3

Relatia de tip 3 dintre entitatatile **PRODUS**, **CUMPARATOR** si **ANGAJAT** defineste procesul prin care un client cumpara produse, iar tranzactia este gestionata de un angajat. Fiecare achizitie este asociata cu un singur angajat, care o procezeaza, dar un angajat poate gestiona mai multe achizitii. De asemenea, un client poate face mai multe achizitii sau poate exista in sistem fara a fi cumparat nimic. Aceasta relatia ajuta la urmarirea istoricului de cumparaturi si la gestionarea vanzarilor.

5 Descrierea atributelor

1. ALBUM

Atribut	Tip de Date	Constrangeri	Valori Posibile	Valori Impligate	Observatii
id_produs	NUMBER(3,0)	PK, FK	101, 102, 103	-	Identificator unic
nume_album	VARCHAR2(100)	-	GNX, SOS, Swimming	-	Numele albumului ca text
lungime	CHAR(5)	-	45:30, 38:22, 72:15	-	Format MM:SS
id_casa_discuri	NUMBER(3,0)	FK	1, 2, 3	-	Legat de casa de discuri
id_gen	NUMBER(3,0)	FK	1, 2, 3	-	Genul muzical al albumului

2. MERCHANDISE

Atribut	Tip de Date	Constrangeri	Valori Posibile	Valori Impligate	Observatii
id_produs	NUMBER(3,0)	PK, FK	201, 202, 203	-	Identificator unic
tip_merch	VARCHAR2(100)	-	T-shirt, Hoodie, Poster, Cap	-	Tipul produsului
marime_merch	VARCHAR2(4)	-	S, M, L, XL, XXL	NULL	Obligatoriu doar pentru imbracaminte

3. PRODUS

Atribut	Tip de Date	Constrangeri	Valori Posibile	Valori Impligate	Observatii
id_produs	NUMBER(3,0)	PK	101, 201, 301	-	Cheia primara a produsului
pret	NUMBER(6,2)	-	99.99, 149.99, 24.99	-	Pret de baza in RON
data_lansare	DATE	-	2024-02-05, 2023-11-12, 2022-08-30	SYSDATE	Data lansarii produsului

4. GEN

Atribut	Tip de Date	Constrangeri	Valori Posibile	Valori Impligate	Observatii
id_gen	NUMBER(3,0)	PK	1, 2, 3, 4	-	Identificator unic
nume_gen	VARCHAR2(100)	-	Rock, Pop, Jazz, Metal, Hip-Hop	-	Numele genului muzical

5. ARTIST

Atribut	Tip de Date	Constrangeri	Valori Posibile	Valori im-plicite	Observatii
id_artist	NUMBER(3,0)	PK	401, 402, 403	-	Identificator unic
prenume	VARCHAR2(100)	-	John, Maria, Alex, Laura	-	Prenumele artistului
nume_familie	VARCHAR2(100)	-	Doe, Smith, Johnson, Popescu	-	Numele de familie
trupa	VARCHAR2(100)	-	The Rockers, Metallica, Arctic Monkeys	NULL	Poate fi NULL pentru artiști solo

6. FORMAT MEDIA

Atribut	Tip de Date	Constrangeri	Valori Posibile	Valori im-plicite	Observatii
id_format_media	NUMBER(3,0)	PK	1, 2, 3, 4	-	Identificator unic
nume_format_media	VARCHAR2(100)	-	Vinyl, CD, Digital, Cassette	-	Numele formatului media
id_produs	NUMBER(3,0)	FK	101, 102, 103	-	Legat de produsul asociat
procent_adaugat	NUMBER(3,0)	-	15, 5, 0, 10	0	Procentul adăugat la pret

7. FURNIZOR

Atribut	Tip de Date	Constrangeri	Valori Posibile	Valori im-plicite	Observatii
id_furnizor	NUMBER(3,0)	PK	1, 2, 3	-	Identificator unic
nume_furnizor	VARCHAR2(100)	-	Music Warehouse, Vinyl Plus, Global Distribution	-	Numele furnizorului
numar_telefon	CHAR(10)	-	0712345678, 0723456789, 0734567890	-	Format romanesc
adresa	VARCHAR2(100)	-	Str. Muzicii 10, Bd. Distributiei 25, Aleea Furnizorilor 5	-	Adresa furnizorului

8. FUNCTIE

Atribut	Tip de Date	Constrangeri	Valori Posibile	Valori im-plicite	Observatii
id_functie	NUMBER(3,0)	PK	1, 2, 3, 4	-	Identificator unic
nume_functie	VARCHAR2(100)	-	Manager, Sales Associate, Cashier, Inventory Specialist	-	Numele functiei
salariu_min	NUMBER(6,2)	-	2500.00, 3000.00, 4000.00	-	Salariul minim in RON
salariu_max	NUMBER(6,2)	-	5000.00, 7000.00, 10000.00	-	Salariul maxim in RON

9. ANGAJAT

Atribut	Tip de Date	Constrangeri	Valori Posibile	Valori im-plicite	Obersvatii
id_angajat	NUMBER(3,0)	PK	1, 2, 3, 4	-	Identificator unic
id_functie	NUMBER(3,0)	FK	1, 2, 3, 4	-	Legat de functia angajatului
nume_familie	VARCHAR2(100)	-	Ionescu, Popescu, Popa, Smith	-	Numele de familie
prenume	VARCHAR2(100)	-	Ion, Maria, Ana, John	-	Prenumele angajatului
numar_telefon	CHAR(11)	-	0712345678, 0723456789, 0734567890	-	Format romanesc
data_angajare	DATE	-	2020-05-15, 2022-03-10, 2023-08-22	SYSDATE	Data angajarii
salariu	NUMBER(6,2)	-	3500.00, 4200.00, 6000.00	-	Salariul curent

10. CUMPARATOR

Atribut	Tip de Date	Constrangeri	Valori Posibile	Valori im-plicite	Obersvatii
id_cumparator	NUMBER(3,0)	PK	1, 2, 3, 4	-	Identificator unic
nume_familie	VARCHAR2(100)	-	Georgescu, Vasilescu, Brown, Johnson	-	Numele de familie
prenume	VARCHAR2(100)	-	Andrei, Elena, Robert, Sarah	-	Prenumele clientului
adresa	VARCHAR2(300)	-	Str. Primaverii 10, Bd. Tineretului 25, Main Street 42	-	Adresa completa
oras	VARCHAR2(100)	-	Bucureşti, Cluj-Napoca, Iaşi, London, New York	-	Oraşul clientului

11. CASA DE DISCURI

Atribut	Tip de Date	Constrangeri	Valori Posibile	Valori im-plicite	Obersvatii
id_casa_discuri	NUMBER(3,0)	PK	1, 2, 3, 4	-	Identificator unic
nume_casa	VARCHAR2(100)	-	Universal Music, Sony Music, Warner Records	-	Numele casei de discuri
email	VARCHAR2(100)	-	contact@universal.com, info@sony.com, support@warner.com	-	Email de contact

12. ACHIZITIE

Atribut	Tip de Date	Constrangeri	Valori Posibile	Valori im-plicite	Obersvatiile
id_achizitie	NUMBER(3,0)	PK	1001, 1002, 1003	-	Identificator unic
id_cumparator	NUMBER(3,0)	FK	1, 2, 3, 4	-	Legat de clientul care cumpara
id_angajat	NUMBER(3,0)	FK	1, 2, 3, 4	-	Legat de angajatul care proceseaza
id_produs	NUMBER(3,0)	FK	101, 201, 301	-	Legat de produsul cumparat
data_achizitie	DATE	-	2024-03-15, 2024-02-28, 2024-01-10	SYSDATE	Data achizitiei
metoda_plata	VARCHAR2(100)	-	Card, Cash, PayPal, Transfer bancar	'Card'	Metoda de plata
status	VARCHAR2(100)	-	Pending, Completed, Cancelled, Delivered	'Pending'	Statusul comenzii
cantitate	NUMBER(3,0)	-	1, 2, 3, 5, 10	1	Cantitatea cumparata

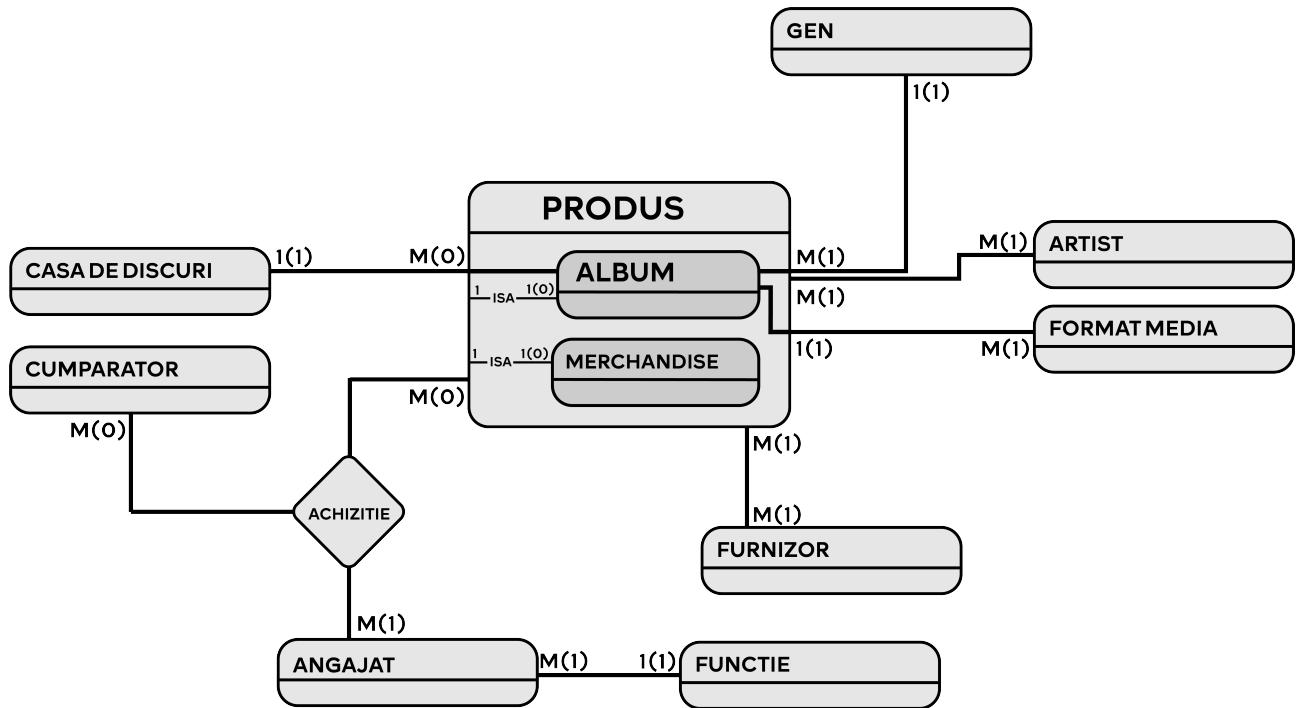
13. STOC

Atribut	Tip de Date	Constrangeri	Valori Posibile	Valori im-plicite	Obersvatiile
id_stoc	NUMBER(3,0)	PK	501, 502, 503	-	Identificator unic
id_produs	NUMBER(3,0)	FK	101, 201, 301	-	Legat de produsul in stoc
id_furnizor	NUMBER(3,0)	FK	1, 2, 3	-	Legat de furnizorul produsului
cantitate	NUMBER(3,0)	-	10, 25, 50, 100	0	Cantitatea disponibila

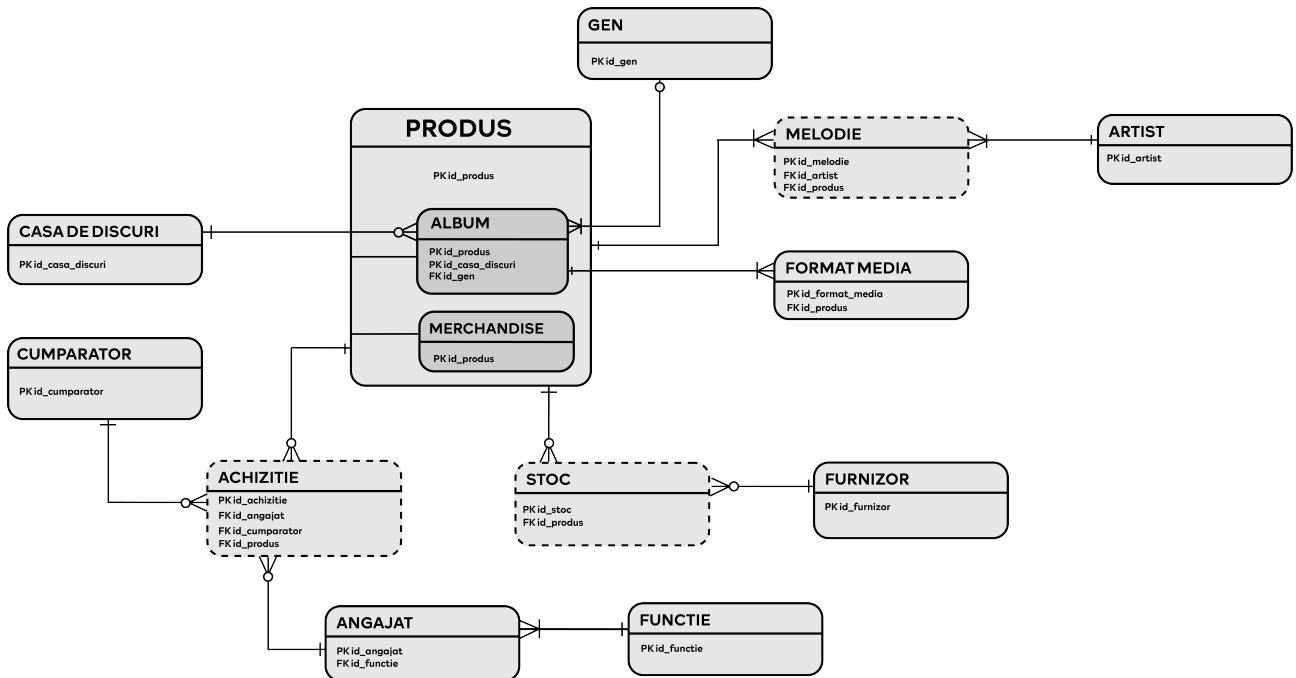
14. MELODIE

Atribut	Tip de Date	Constrangeri	Valori Posibile	Valori im-plicite	Obersvatiile
id_melodie	NUMBER(3,0)	PK	601, 602, 603	-	Identifier unic
id_artist	NUMBER(3,0)	FK	401, 402, 403	-	Legat de artistul melodiei
id_produs	NUMBER(3,0)	FK	101, 102, 103	-	Legat de produsul (albумul)
titlu	VARCHAR2(100)	-	Summer Vibes, Rock Forever, Soul of Mine	-	Titlul melodiei
lungime	VARCHAR2(100)	-	3:45, 4:20, 2:55, 6:10	-	Durata melodiei (MM:SS)

6 Realizarea diagramei entitate-relatie



7 Realizarea diagramei conceptuale



8 Scheme relationale

Schemele relattionale corespunzătoare diagramei conceptuale sunt următoarele:

ALBUM(id_produs#, nume_album, lungime, id_casa_discuri, id_gen)

MERCHANDISE(id_produs#, tip_merch, marime_merch)

PRODUS(id_produs#, pret, data_lansare)

GEN(id_gen#, nume_gen)

ARTIST(id_artist#, prenume, nume_familie, trupa)

FORMAT MEDIA(id_format_media#, nume_format_media, id_produs, procent_adaugat)

FURNIZOR(id_furnizor#, nume_furnizor, numar_telefon, adresa)

FUNCTIONIE(id_functie#, nume_functie, salariu_min, salariu_max)

ANGAJAT(id_angajat#, id_functie, nume_familie, prenume, numar_telefon, data_angajare, salariu)

CUMPARATOR(id_cumparator#, nume_familie, prenume, adresa, oras)

CASA DE DISCURI(id_casa_discuri#, nume_casa, email)

ACHIZITIE(id_achizitie#, id_cumparator, id_angajat, id_produs, data_achizitie, metoda_plata, status, cantitate)

STOC(id_stoc#, id_produs, id_furnizor, cantitate)

MELODIE(id_melodie#, id_artist, id_produs, titlu, lungime)

9 Realizarea normalizarii FN1-FN3

9.1 Forma normala 1 - FN1

Forma normala 1 (FN1) impune ca toate valorile stocate intr-un camp al unei baze de date sa fie atomice, adica sa contina o singura valoare, nu liste sau grupuri de date. Totodata, fiecare rand din tabel trebuie sa poata fi identificat in mod unic, prin intermediul unei chei primare.

In modelul implementat, cerintele acestei forme normale sunt respectate: nu exista campuri care contin mai multe valori, iar fiecare inregistrare este identificabila in mod clar cu ajutorul unei chei primare.

Pentru a ilustra aceasta etapa de normalizare, se poate analiza urmatorul exemplu:

ARTIST	MELODIE
Artist1	ML1, ML2, ML3
Artist2	ML2, ML4
Artist3	ML3

Table 1: Exemplu non-FN1.

ARTIST	MELODIE
Artist1	ML1
Artist1	ML2
Artist1	ML3
Artist2	ML2
Artist2	ML4
Artist3	ML3

Table 2: Exemplu FN1.

9.2 Forma normala 2 - FN2

O relatie se afla in a doua forma normala (FN2) doar daca este deja in forma normala 1 (FN1) si daca fiecare atribut care nu face parte din cheia primara depinde de intreaga cheie, nu doar de o parte a acesteia.

Mai exact, FN2 presupune ca toate atributurile non-cheie dintr-o tabela sa fie dependente functionale de cheia primara in intregime, nu parțial.

In cazul modelului implementat, sunt indeplinite conditiile formei normale a doua: relatiile sunt in FN1, iar toate atributurile care nu fac parte din cheile primare sunt dependente in mod complet de acestea.

Pentru a ilustra procesul de normalizare, vom analiza exemplul urmator:

id_artist#	prenume	id_produs#	pret
A1	Kendrick	P1	6.99
A2	West	P1	6.99
A1	Kendrick	P2	15.99
A3	Miller	P3	10.49

Table 3: Exemplu non-FN2.

Observam ca avem urmatoarele dependente:

$\{id_artist\} \rightarrow \{prenume\}$. id_artist determina functional numele.
 $\{id_produs\} \rightarrow \{pret\}$

id_artist#	id_produs#	pret
A1	P1	6.99
A2	P1	6.99
A1	P2	15.99
A3	P3	10.49

Table 4: Exemplu FN2.

id_artist#	prenume
A1	Kendrick
A2	West
A3	Miller

Table 5: Exemplu FN2.

9.3 Forma normala 3 - FN3

Asemanator, o entitate se găsește în a treia forma(FN3) normală dacă și numai dacă se găsește în a doua forma normală(FN2) și în plus niciun atribut care nu este parte a UID-ului nu depinde de un alt atribut non-UID. Cu alte cuvinte, nu se acceptă dependențe tranzitive, adică un atribut să depindă de UID în mod indirect.

FN3 cere ca orice atribut non-cheie să depindă doar de cheia primă, de întreaga cheie și exclusiv de aceasta, fără intermedieri prin alte attribute.

În cadrul modelului implementat, se respectă toate cerințele formei normale a treia: toate attributele care nu sunt chei sunt dependente direct de cheia primă.

Pentru a ilustra această normalizare, vom analiza exemplul următor:

id_artist#	id_produs#	pret
A1	P1	6.99
A2	P1	6.99
A1	P2	15.99
A3	P3	10.49

Table 6: Exemplu non-FN3.

Pentru a aduce relația R în forma normală 3 (FN3), se elimină dependențele functionale tranzitive, astfel încât toate attributele non-cheie să depindă direct și exclusiv de cheia primă.

R1(id_produs#, pret)

R2(id_artist#, id_produs#)

id_produs#	pret
P1	6.99
P2	15.99
P3	10.49

Table 7: Exemplu FN3.

id_artist#	id_produs#
A1	P1
A2	P1
A1	P2
A3	P3

Table 8: Exemplu FN3.

10 Crearea seventelor utilizate in inserarea inregistrarilor

10.1 CASA DE DISCURI

```
CREATE SEQUENCE SEQ_CASA_DE_DISCURI
    INCREMENT BY 1
    START WITH 1
    MAXVALUE 1000
    NOCYCLE;
```

10.2 PRODUS

```
CREATE SEQUENCE SEQ_PRODUS
    INCREMENT BY 1
    START WITH 1
    MAXVALUE 1000
    NOCYCLE;
```

10.3 GEN

```
CREATE SEQUENCE SEQ_GEN
    INCREMENT BY 1
    START WITH 1
    MAXVALUE 1000
    NOCYCLE;
```

10.4 ARTIST

```
CREATE SEQUENCE SEQ_ARTIST
    INCREMENT BY 1
    START WITH 1
    MAXVALUE 1000
    NOCYCLE;
```

10.5 FURNIZOR

```
CREATE SEQUENCE SEQ_FURNIZOR
    INCREMENT BY 1
    START WITH 1
    MAXVALUE 1000
    NOCYCLE;
```

10.6 FORMAT MEDIA

```
CREATE SEQUENCE SEQ_FORMAT_MEDIA
    INCREMENT BY 1
    START WITH 1
    MAXVALUE 1000
    NOCYCLE;
```

10.7 FUNCTIE

```
CREATE SEQUENCE SEQ_FUNCTIE
    INCREMENT BY 1
    START WITH 1
    MAXVALUE 1000
    NOCYCLE;
```

10.8 ANGAJAT

```
CREATE SEQUENCE SEQ_ANGAJAT
    INCREMENT BY 1
    START WITH 1
    MAXVALUE 1000
    NOCYCLE;
```

10.9 CUMPARATOR

```
CREATE SEQUENCE SEQ_CUMPARATOR
    INCREMENT BY 1
    START WITH 1
    MAXVALUE 1000
    NOCYCLE;
```

10.10 STOC

```
CREATE SEQUENCE SEQ_STOC
    INCREMENT BY 1
    START WITH 1
    MAXVALUE 1000
    NOCYCLE;
```

10.11 ACHIZITIE

```
CREATE SEQUENCE SEQ_ACHIZITIE
    INCREMENT BY 1
    START WITH 1
    MAXVALUE 1000
    NOCYCLE;
```

10.12 MELODIE

```
CREATE SEQUENCE SEQ_MELODIE
    INCREMENT BY 1
    START WITH 1
    MAXVALUE 1000
    NOCYCLE;
```

11 Creearea tabelelor SQL si inserarea datelor

11.1 CASA DE DISCURI

```
CREATE TABLE CASA_DE_DISCURI (
    id_casa_discuri NUMBER(3,0) CONSTRAINT pk_casa_de_discuri PRIMARY KEY,
    nume_casa VARCHAR2(100),
    email VARCHAR2(100)
);

INSERT INTO CASA_DE_DISCURI VALUES (SEQ_CASA_DE_DISCURI.NEXTVAL, 'Universal Music',
'contact@universalmusic.com');
INSERT INTO CASA_DE_DISCURI VALUES (SEQ_CASA_DE_DISCURI.NEXTVAL, 'Sony Music',
'info@sonymusic.com');
INSERT INTO CASA_DE_DISCURI VALUES (SEQ_CASA_DE_DISCURI.NEXTVAL, 'Republic Records',
'office@republicrecords.com');
INSERT INTO CASA_DE_DISCURI VALUES (SEQ_CASA_DE_DISCURI.NEXTVAL, 'Atlantic Records',
'info@atlanticrecords.com');
INSERT INTO CASA_DE_DISCURI VALUES (SEQ_CASA_DE_DISCURI.NEXTVAL, 'Electrecord',
```

```
'contact@electrecord.ro');
INSERT INTO CASA_DE_DISCURI VALUES (SEQ_CASA_DE_DISCURI.NEXTVAL, 'Def Jam Recordings',
'contact@defjam.com');
INSERT INTO CASA_DE_DISCURI VALUES (SEQ_CASA_DE_DISCURI.NEXTVAL, 'Interscope Records',
'info@interscope.com');

COMMIT;
```

11.2 PRODUS

```
CREATE TABLE PRODUS (
    id_produs NUMBER(3,0) CONSTRAINT pk_produs PRIMARY KEY,
    pret NUMBER(6,2),
    data_lansare DATE DEFAULT SYSDATE
);

INSERT INTO PRODUS VALUES (SEQ_PRODUS.NEXTVAL, 49.99, TO_DATE('15/03/2022', 'DD/MM/YYYY'));
INSERT INTO PRODUS VALUES (SEQ_PRODUS.NEXTVAL, 49.99, TO_DATE('15/03/2022', 'DD/MM/YYYY'));
INSERT INTO PRODUS VALUES (SEQ_PRODUS.NEXTVAL, 39.99, TO_DATE('22/05/2022', 'DD/MM/YYYY'));
INSERT INTO PRODUS VALUES (SEQ_PRODUS.NEXTVAL, 29.99, TO_DATE('10/07/2022', 'DD/MM/YYYY'));
INSERT INTO PRODUS VALUES (SEQ_PRODUS.NEXTVAL, 19.99, TO_DATE('30/09/2022', 'DD/MM/YYYY'));
INSERT INTO PRODUS VALUES (SEQ_PRODUS.NEXTVAL, 59.99, TO_DATE('12/12/2022', 'DD/MM/YYYY'));
INSERT INTO PRODUS VALUES (SEQ_PRODUS.NEXTVAL, 34.99, TO_DATE('25/01/2023', 'DD/MM/YYYY'));
INSERT INTO PRODUS VALUES (SEQ_PRODUS.NEXTVAL, 44.99, TO_DATE('18/03/2023', 'DD/MM/YYYY'));
INSERT INTO PRODUS VALUES (SEQ_PRODUS.NEXTVAL, 24.99, TO_DATE('05/05/2023', 'DD/MM/YYYY'));
INSERT INTO PRODUS VALUES (SEQ_PRODUS.NEXTVAL, 64.99, TO_DATE('28/06/2023', 'DD/MM/YYYY'));
INSERT INTO PRODUS VALUES (SEQ_PRODUS.NEXTVAL, 54.99, TO_DATE('15/08/2023', 'DD/MM/YYYY'));
INSERT INTO PRODUS VALUES (SEQ_PRODUS.NEXTVAL, 52.99, TO_DATE('15/09/2023', 'DD/MM/YYYY'));
INSERT INTO PRODUS VALUES (SEQ_PRODUS.NEXTVAL, 47.99, TO_DATE('25/10/2023', 'DD/MM/YYYY'));
INSERT INTO PRODUS VALUES (SEQ_PRODUS.NEXTVAL, 22.99, TO_DATE('08/11/2023', 'DD/MM/YYYY'));
INSERT INTO PRODUS VALUES (SEQ_PRODUS.NEXTVAL, 37.99, TO_DATE('01/12/2023', 'DD/MM/YYYY'));
INSERT INTO PRODUS VALUES (SEQ_PRODUS.NEXTVAL, 42.99, TO_DATE('15/01/2024', 'DD/MM/YYYY'));

COMMIT;
```

11.3 GEN

```
CREATE TABLE GEN (
    id_gen NUMBER(3,0) CONSTRAINT pk_gen PRIMARY KEY,
    nume_gen VARCHAR2(100)
);

INSERT INTO GEN VALUES (SEQ_GEN.NEXTVAL, 'RnB');
INSERT INTO GEN VALUES (SEQ_GEN.NEXTVAL, 'Rap');
INSERT INTO GEN VALUES (SEQ_GEN.NEXTVAL, 'Pop');
INSERT INTO GEN VALUES (SEQ_GEN.NEXTVAL, 'Rap-RNB');
INSERT INTO GEN VALUES (SEQ_GEN.NEXTVAL, 'Folk');
INSERT INTO GEN VALUES (SEQ_GEN.NEXTVAL, 'Hip-Hop');
INSERT INTO GEN VALUES (SEQ_GEN.NEXTVAL, 'Alternative');
INSERT INTO GEN VALUES (SEQ_GEN.NEXTVAL, 'Indie');
INSERT INTO GEN VALUES (SEQ_GEN.NEXTVAL, 'Electronic');
INSERT INTO GEN VALUES (SEQ_GEN.NEXTVAL, 'Alternative Pop');

COMMIT;
```

11.4 ALBUM

```
CREATE TABLE ALBUM (
    id_produs NUMBER(3,0) CONSTRAINT pk_album PRIMARY KEY,
```

```

nume_album VARCHAR2(100),
lungime CHAR(5),
id_casa_discuri NUMBER(3,0),
id_gen NUMBER(3,0),
CONSTRAINT fk_album_produs FOREIGN KEY (id_produs) REFERENCES PRODUS(id_produs),
CONSTRAINT fk_album_casa FOREIGN KEY (id_casa_discuri) REFERENCES CASA_DE_DISCURI(id_casa_discuri),
CONSTRAINT fk_album_gen FOREIGN KEY (id_gen) REFERENCES GEN(id_gen)
);

INSERT INTO ALBUM VALUES (1, 'Swimming', '49:07', 1, 1);
INSERT INTO ALBUM VALUES (2, 'GNX', '42:11', 2, 2);
INSERT INTO ALBUM VALUES (3, 'Short n Sweet', '42:19', 3, 3);
INSERT INTO ALBUM VALUES (4, 'HUMBLE', '43:00', 4, 4);
INSERT INTO ALBUM VALUES (5, 'folklore', '45:30', 5, 5);
INSERT INTO ALBUM VALUES (11, 'the album', '30:45', 1, 8);
INSERT INTO ALBUM VALUES (12, 'Blonde', '60:08', 3, 7);
INSERT INTO ALBUM VALUES (13, 'DAMN.', '54:54', 4, 2);
INSERT INTO ALBUM VALUES (15, 'After Hours', '56:19', 6, 9);
INSERT INTO ALBUM VALUES (16, 'Happier Than Ever', '56:15', 7, 10);

COMMIT;

```

11.5 MERCANDISE

```

CREATE TABLE MERCANDISE (
    id_produs NUMBER(3,0) CONSTRAINT pk_merchandise PRIMARY KEY,
    tip_merch VARCHAR2(100),
    marime_merch VARCHAR2(4),
    CONSTRAINT fk_merchandise_produs FOREIGN KEY (id_produs) REFERENCES PRODUS(id_produs)
);

INSERT INTO MERCANDISE VALUES (6, 'T-Shirt', 'M');
INSERT INTO MERCANDISE VALUES (7, 'Poster', NULL);
INSERT INTO MERCANDISE VALUES (8, 'Cap', 'S');
INSERT INTO MERCANDISE VALUES (9, 'Mug', NULL);
INSERT INTO MERCANDISE VALUES (10, 'Hoodie', 'XL');
INSERT INTO MERCANDISE VALUES (14, 'Vinyl Stand', NULL);

COMMIT;

```

11.6 ARTIST

```

CREATE TABLE ARTIST (
    id_artist NUMBER(3,0) CONSTRAINT pk_artist PRIMARY KEY,
    prenume VARCHAR2(100),
    nume_familie VARCHAR2(100),
    trupa VARCHAR2(100)
);

INSERT INTO ARTIST VALUES (SEQ_ARTIST.NEXTVAL, 'Mac', 'Miller', NULL);
INSERT INTO ARTIST VALUES (SEQ_ARTIST.NEXTVAL, 'Kendrick', 'Lamar', NULL);
INSERT INTO ARTIST VALUES (SEQ_ARTIST.NEXTVAL, 'Sabrina', 'Carpenter', NULL);
INSERT INTO ARTIST VALUES (SEQ_ARTIST.NEXTVAL, 'Taylor', 'Swift', NULL);
INSERT INTO ARTIST VALUES (SEQ_ARTIST.NEXTVAL, 'boygenius', 'boygenius', 'band');
INSERT INTO ARTIST VALUES (SEQ_ARTIST.NEXTVAL, 'Rihanna', NULL, NULL);
INSERT INTO ARTIST VALUES (SEQ_ARTIST.NEXTVAL, 'Frank', 'Ocean', NULL);
INSERT INTO ARTIST VALUES (SEQ_ARTIST.NEXTVAL, 'Billie', 'Eilish', NULL);
INSERT INTO ARTIST VALUES (SEQ_ARTIST.NEXTVAL, 'The Weeknd', NULL, NULL);
INSERT INTO ARTIST VALUES (SEQ_ARTIST.NEXTVAL, 'SZA', NULL, NULL);

```

COMMIT;

11.7 FURNIZOR

```
CREATE TABLE FURNIZOR (
    id_furnizor NUMBER(3,0) CONSTRAINT pk_furnizor PRIMARY KEY,
    nume_furnizor VARCHAR2(100),
    numar_telefon CHAR(10),
    adresa VARCHAR2(100)
);

INSERT INTO FURNIZOR VALUES (SEQ_FURNIZOR.NEXTVAL, 'Music Distribution SRL', '07456123890',
'Str. Muzicii nr. 15, Bucureşti');
INSERT INTO FURNIZOR VALUES (SEQ_FURNIZOR.NEXTVAL, 'Vinyl Factory', '07345678912',
'Calea Victoriei nr. 78, Bucureşti');
INSERT INTO FURNIZOR VALUES (SEQ_FURNIZOR.NEXTVAL, 'Merch Production', '07234567891',
'Bd. Unirii nr. 45, Bucureşti');
INSERT INTO FURNIZOR VALUES (SEQ_FURNIZOR.NEXTVAL, 'Global Logistics', '07123456789',
'Str. Industriilor nr. 12, Cluj-Napoca');
INSERT INTO FURNIZOR VALUES (SEQ_FURNIZOR.NEXTVAL, 'Record Distributor', '07987654321',
'Str. Republicii nr. 36, Timişoara');
INSERT INTO FURNIZOR VALUES (SEQ_FURNIZOR.NEXTVAL, 'Audio Tech SRL', '07765432109',
'Str. Aviației nr. 24, Bucureşti');
INSERT INTO FURNIZOR VALUES (SEQ_FURNIZOR.NEXTVAL, 'Music Warehouse', '07654321098',
'Calea Floreasca nr. 55, Bucureşti');
```

COMMIT;

11.8 FORMAT MEDIA

```
CREATE TABLE FORMAT_MEDIA (
    id_format_media NUMBER(3,0) CONSTRAINT pk_format_media PRIMARY KEY,
    nume_format_media VARCHAR2(100),
    id_produs NUMBER(3,0),
    procent_adaugat NUMBER(3,0) DEFAULT 0,
    CONSTRAINT fk_format_media_produs FOREIGN KEY (id_produs) REFERENCES PRODUS(id_produs)
);

INSERT INTO FORMAT_MEDIA VALUES (SEQ_FORMAT_MEDIA.NEXTVAL, 'CD', 1, 10);
INSERT INTO FORMAT_MEDIA VALUES (SEQ_FORMAT_MEDIA.NEXTVAL, 'Vinyl', 1, 25);
INSERT INTO FORMAT_MEDIA VALUES (SEQ_FORMAT_MEDIA.NEXTVAL, 'Cassette', 1, 25);
INSERT INTO FORMAT_MEDIA VALUES (SEQ_FORMAT_MEDIA.NEXTVAL, 'Digital', 1, 25);
INSERT INTO FORMAT_MEDIA VALUES (SEQ_FORMAT_MEDIA.NEXTVAL, 'Vinyl', 2, 20);
INSERT INTO FORMAT_MEDIA VALUES (SEQ_FORMAT_MEDIA.NEXTVAL, 'Digital', 3, 0);
INSERT INTO FORMAT_MEDIA VALUES (SEQ_FORMAT_MEDIA.NEXTVAL, 'CD', 3, 0);
INSERT INTO FORMAT_MEDIA VALUES (SEQ_FORMAT_MEDIA.NEXTVAL, 'Cassette', 4, 5);
INSERT INTO FORMAT_MEDIA VALUES (SEQ_FORMAT_MEDIA.NEXTVAL, 'Digital', 5, 0);
INSERT INTO FORMAT_MEDIA VALUES (SEQ_FORMAT_MEDIA.NEXTVAL, 'Vinyl', 5, 25);
INSERT INTO FORMAT_MEDIA VALUES (SEQ_FORMAT_MEDIA.NEXTVAL, 'Cassette', 5, 10);
INSERT INTO FORMAT_MEDIA VALUES (SEQ_FORMAT_MEDIA.NEXTVAL, 'Digital', 11, 0);
INSERT INTO FORMAT_MEDIA VALUES (SEQ_FORMAT_MEDIA.NEXTVAL, 'Digital', 12, 0);
INSERT INTO FORMAT_MEDIA VALUES (SEQ_FORMAT_MEDIA.NEXTVAL, 'Vinyl', 12, 25);
INSERT INTO FORMAT_MEDIA VALUES (SEQ_FORMAT_MEDIA.NEXTVAL, 'Cassette', 13, 10);
INSERT INTO FORMAT_MEDIA VALUES (SEQ_FORMAT_MEDIA.NEXTVAL, 'Digital', 15, 0);
INSERT INTO FORMAT_MEDIA VALUES (SEQ_FORMAT_MEDIA.NEXTVAL, 'CD', 15, 15);
INSERT INTO FORMAT_MEDIA VALUES (SEQ_FORMAT_MEDIA.NEXTVAL, 'Digital', 16, 0);
INSERT INTO FORMAT_MEDIA VALUES (SEQ_FORMAT_MEDIA.NEXTVAL, 'Vinyl 7"', 16, 20);
```

COMMIT;

11.9 FUNCTIE

```
CREATE TABLE FUNCTIE (
    id_functie NUMBER(3,0) CONSTRAINT pk_functie PRIMARY KEY,
    nume_functie VARCHAR2(100),
    salariu_min NUMBER(6,2),
    salariu_max NUMBER(6,2)
);

INSERT INTO FUNCTIE VALUES (SEQ_FUNCTIE.NEXTVAL, 'Manager', 5000, 8000);
INSERT INTO FUNCTIE VALUES (SEQ_FUNCTIE.NEXTVAL, 'Sales Representative', 3000, 5000);
INSERT INTO FUNCTIE VALUES (SEQ_FUNCTIE.NEXTVAL, 'Cashier', 2500, 3500);
INSERT INTO FUNCTIE VALUES (SEQ_FUNCTIE.NEXTVAL, 'Marketing Specialist', 4000, 6000);
INSERT INTO FUNCTIE VALUES (SEQ_FUNCTIE.NEXTVAL, 'Store Assistant', 2000, 3000);
INSERT INTO FUNCTIE VALUES (SEQ_FUNCTIE.NEXTVAL, 'IT Support', 3500, 5500);
INSERT INTO FUNCTIE VALUES (SEQ_FUNCTIE.NEXTVAL, 'Inventory Manager', 4500, 6500);
```

COMMIT;

11.10 ANGAJAT

```
CREATE TABLE ANGAJAT (
    id_angajat NUMBER(3,0) CONSTRAINT pk_angajat PRIMARY KEY,
    id_functie NUMBER(3,0),
    nume_familie VARCHAR2(100),
    prenume VARCHAR2(100),
    numar_telefon CHAR(10),
    data_angajare DATE DEFAULT SYSDATE,
    salariu NUMBER(6,2),
    CONSTRAINT fk_angajat_functie FOREIGN KEY (id_functie) REFERENCES FUNCTIE(id_functie)
);

INSERT INTO ANGAJAT VALUES (SEQ_ANGAJAT.NEXTVAL, 1, 'Popescu', 'Maria', '07123456789',
TO_DATE('15/01/2020', 'DD/MM/YYYY'), 6500);
INSERT INTO ANGAJAT VALUES (SEQ_ANGAJAT.NEXTVAL, 2, 'Ionescu', 'Andrei', '07234567890',
TO_DATE('10/03/2020', 'DD/MM/YYYY'), 4000);
INSERT INTO ANGAJAT VALUES (SEQ_ANGAJAT.NEXTVAL, 3, 'Popa', 'Elena', '07345678901',
TO_DATE('22/05/2021', 'DD/MM/YYYY'), 3000);
INSERT INTO ANGAJAT VALUES (SEQ_ANGAJAT.NEXTVAL, 4, 'Georgescu', 'Mihai', '07456789012',
TO_DATE('17/08/2021', 'DD/MM/YYYY'), 5000);
INSERT INTO ANGAJAT VALUES (SEQ_ANGAJAT.NEXTVAL, 5, 'Stanciu', 'Cristina', '07567890123',
TO_DATE('05/12/2021', 'DD/MM/YYYY'), 2500);
INSERT INTO ANGAJAT VALUES (SEQ_ANGAJAT.NEXTVAL, 6, 'Dumitrache', 'Alexandru', '07678901234',
TO_DATE('10/02/2022', 'DD/MM/YYYY'), 4500);
INSERT INTO ANGAJAT VALUES (SEQ_ANGAJAT.NEXTVAL, 7, 'Marinescu', 'Diana', '07789012345',
TO_DATE('18/04/2022', 'DD/MM/YYYY'), 5500);
```

COMMIT;

11.11 CUMPARATOR

```
CREATE TABLE CUMPARATOR (
    id_cumparator NUMBER(3,0) CONSTRAINT pk_cumparator PRIMARY KEY,
    nume_familie VARCHAR2(100),
    prenume VARCHAR2(100),
    adresa VARCHAR2(300),
    oras VARCHAR2(100)
```

```

);

INSERT INTO CUMPARATOR VALUES (SEQ_CUMPARATOR.NEXTVAL, 'Dumitrescu', 'Alexandru',
'Str. Libertatii nr. 10, Bucureşti', 'Bucureşti');
INSERT INTO CUMPARATOR VALUES (SEQ_CUMPARATOR.NEXTVAL, 'Radu', 'Ioana',
'Str. Unirii nr. 25, Cluj-Napoca', 'Cluj-Napoca');
INSERT INTO CUMPARATOR VALUES (SEQ_CUMPARATOR.NEXTVAL, 'Stoica', 'Gabriel',
'Bd. Independentei nr. 15, Iaşi', 'Iaşi');
INSERT INTO CUMPARATOR VALUES (SEQ_CUMPARATOR.NEXTVAL, 'Munteanu', 'Diana',
'Str. Mihai Viteazul nr. 8, Timişoara', 'Timişoara');
INSERT INTO CUMPARATOR VALUES (SEQ_CUMPARATOR.NEXTVAL, 'Dinu', 'Bogdan',
'Str. Primaverii nr. 12, Braşov', 'Braşov');
INSERT INTO CUMPARATOR VALUES (SEQ_CUMPARATOR.NEXTVAL, 'Vasile', 'Andreea',
'Str. Florilor nr. 5, Constanţa', 'Constanţa');
INSERT INTO CUMPARATOR VALUES (SEQ_CUMPARATOR.NEXTVAL, 'Preda', 'Robert',
'Str. Tudor Vladimirescu nr. 18, Craiova', 'Craiova');
INSERT INTO CUMPARATOR VALUES (SEQ_CUMPARATOR.NEXTVAL, 'Manole', 'Cristina',
'Str. Mioritei nr. 7, Sibiu', 'Sibiu');

COMMIT;

```

11.12 STOC

```

CREATE TABLE STOC (
    id_stoc NUMBER(3,0) CONSTRAINT pk_stoc PRIMARY KEY,
    id_produs NUMBER(3,0),
    id_furnizor NUMBER(3,0),
    cantitate NUMBER(3,0) DEFAULT 0,
    CONSTRAINT fk_stoc_produs FOREIGN KEY (id_produs) REFERENCES PRODUS(id_produs),
    CONSTRAINT fk_stoc_furnizor FOREIGN KEY (id_furnizor) REFERENCES FURNIZOR(id_furnizor)
);

INSERT INTO STOC VALUES (SEQ_STOC.NEXTVAL, 1, 1, 50);
INSERT INTO STOC VALUES (SEQ_STOC.NEXTVAL, 1, 3, 50);
INSERT INTO STOC VALUES (SEQ_STOC.NEXTVAL, 2, 2, 30);
INSERT INTO STOC VALUES (SEQ_STOC.NEXTVAL, 2, 7, 30);
INSERT INTO STOC VALUES (SEQ_STOC.NEXTVAL, 2, 5, 70);
INSERT INTO STOC VALUES (SEQ_STOC.NEXTVAL, 3, 1, 100);
INSERT INTO STOC VALUES (SEQ_STOC.NEXTVAL, 4, 2, 20);
INSERT INTO STOC VALUES (SEQ_STOC.NEXTVAL, 5, 1, 40);
INSERT INTO STOC VALUES (SEQ_STOC.NEXTVAL, 6, 3, 75);
INSERT INTO STOC VALUES (SEQ_STOC.NEXTVAL, 7, 3, 60);
INSERT INTO STOC VALUES (SEQ_STOC.NEXTVAL, 8, 3, 90);
INSERT INTO STOC VALUES (SEQ_STOC.NEXTVAL, 9, 3, 45);
INSERT INTO STOC VALUES (SEQ_STOC.NEXTVAL, 10, 3, 30);
INSERT INTO STOC VALUES (SEQ_STOC.NEXTVAL, 11, 1, 25);
INSERT INTO STOC VALUES (SEQ_STOC.NEXTVAL, 12, 2, 15);
INSERT INTO STOC VALUES (SEQ_STOC.NEXTVAL, 13, 1, 35);
INSERT INTO STOC VALUES (SEQ_STOC.NEXTVAL, 14, 3, 50);
INSERT INTO STOC VALUES (SEQ_STOC.NEXTVAL, 15, 2, 40);
INSERT INTO STOC VALUES (SEQ_STOC.NEXTVAL, 15, 4, 60);
INSERT INTO STOC VALUES (SEQ_STOC.NEXTVAL, 15, 5, 70);
INSERT INTO STOC VALUES (SEQ_STOC.NEXTVAL, 16, 1, 30);

COMMIT;

```

11.13 ACHIZITIE

```
CREATE TABLE ACHIZITIE (
```

```

id_achizitie NUMBER(3,0) CONSTRAINT pk_achizitie PRIMARY KEY,
id_cumparator NUMBER(3,0),
id_angajat NUMBER(3,0),
id_produs NUMBER(3,0),
data_achizitie DATE DEFAULT SYSDATE,
metoda_plata VARCHAR2(100) DEFAULT 'Card',
status VARCHAR2(100) DEFAULT 'Pending',
cantitate NUMBER(3,0) DEFAULT 1,
CONSTRAINT fk_achizitie_cumparator FOREIGN KEY (id_cumparator) REFERENCES CUMPARATOR(id_cumparator)
CONSTRAINT fk_achizitie_angajat FOREIGN KEY (id_angajat) REFERENCES ANGAJAT(id_angajat),
CONSTRAINT fk_achizitie_produs FOREIGN KEY (id_produs) REFERENCES PRODUS(id_produs)
);

INSERT INTO ACHIZITIE VALUES (SEQ_ACHIZITIE.NEXTVAL, 1, 2, 1, TO_DATE('20/04/2023', 'DD/MM/YYYY'), 'Card', 'Completed', 1);
INSERT INTO ACHIZITIE VALUES (SEQ_ACHIZITIE.NEXTVAL, 2, 2, 3, TO_DATE('25/04/2023', 'DD/MM/YYYY'), 'Cash', 'Completed', 1);
INSERT INTO ACHIZITIE VALUES (SEQ_ACHIZITIE.NEXTVAL, 3, 3, 6, TO_DATE('30/04/2023', 'DD/MM/YYYY'), 'Card', 'Completed', 2);
INSERT INTO ACHIZITIE VALUES (SEQ_ACHIZITIE.NEXTVAL, 4, 2, 2, TO_DATE('05/05/2023', 'DD/MM/YYYY'), 'Card', 'Pending', 1);
INSERT INTO ACHIZITIE VALUES (SEQ_ACHIZITIE.NEXTVAL, 5, 3, 8, TO_DATE('10/05/2023', 'DD/MM/YYYY'), 'Cash', 'Pending', 3);
INSERT INTO ACHIZITIE VALUES (SEQ_ACHIZITIE.NEXTVAL, 1, 2, 5, TO_DATE('15/05/2023', 'DD/MM/YYYY'), 'Card', 'Completed', 1);
INSERT INTO ACHIZITIE VALUES (SEQ_ACHIZITIE.NEXTVAL, 2, 3, 10, TO_DATE('20/05/2023', 'DD/MM/YYYY'), 'Cash', 'Cancelled', 1);
INSERT INTO ACHIZITIE VALUES (SEQ_ACHIZITIE.NEXTVAL, 3, 2, 4, TO_DATE('25/05/2023', 'DD/MM/YYYY'), 'Card', 'Completed', 1);
INSERT INTO ACHIZITIE VALUES (SEQ_ACHIZITIE.NEXTVAL, 6, 4, 12, TO_DATE('02/06/2023', 'DD/MM/YYYY'), 'Card', 'Completed', 1);
INSERT INTO ACHIZITIE VALUES (SEQ_ACHIZITIE.NEXTVAL, 7, 5, 13, TO_DATE('10/06/2023', 'DD/MM/YYYY'), 'Cash', 'Completed', 2);
INSERT INTO ACHIZITIE VALUES (SEQ_ACHIZITIE.NEXTVAL, 8, 2, 15, TO_DATE('18/06/2023', 'DD/MM/YYYY'), 'Card', 'Pending', 1);
INSERT INTO ACHIZITIE VALUES (SEQ_ACHIZITIE.NEXTVAL, 4, 3, 11, TO_DATE('25/06/2023', 'DD/MM/YYYY'), 'Cash', 'Completed', 1);

COMMIT;

```

11.14 MELODIE

```

CREATE TABLE MELODIE (
    id_melodie NUMBER(3,0) CONSTRAINT pk_melodie PRIMARY KEY,
    id_artist NUMBER(3,0),
    id_produs NUMBER(3,0),
    titlu VARCHAR2(100),
    lungime VARCHAR2(100),
    CONSTRAINT fk_melodie_artist FOREIGN KEY (id_artist) REFERENCES ARTIST(id_artist),
    CONSTRAINT fk_melodie_produs FOREIGN KEY (id_produs) REFERENCES PRODUS(id_produs)
);

INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 1, 1, 'Come Back to Earth', '2:41');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 1, 1, 'Hurt Feelings', '4:05');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 1, 1, 'What's the Use?', '4:48');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 1, 1, 'Perfecto', '3:35');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 1, 1, 'Self Care', '5:45');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 1, 1, 'Wings', '4:10');

```

```

INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 1, 1, 'Ladders', '4:47');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 1, 1, 'Small Worlds', '4:31');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 1, 1, 'Conversation Pt. 1', '3:30');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 1, 1, 'Dunno', '3:57');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 1, 1, 'Jet Fuel', '5:45');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 1, 1, '2009', '5:47');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 1, 1, 'So It Goes', '5:12');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 2, 'wacked out murals', '5:17');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 2, 'squabble up', '2:37');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 10, 2, 'luther (with sza)', '2:57');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 2, 'luther (with sza)', '2:57');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 2, 'man at the garden', '3:53');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 2, 'hey now (feat. dody6)', '3:37');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 2, 'reincarnated', '4:35');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 2, 'tv off (feat. hefty gunplay)', '3:40');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 2, 'dodger blue (feat. wallie the...)', '2:11');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 2, 'peekaboo (feat. azchike)', '2:35');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 2, 'heart pt. 6', '4:52');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 3, 3, 'Taste', '4:54');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 3, 3, 'Please Please Please', '5:57');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 3, 3, 'Good Graces', '3:05');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 3, 3, 'Sharpest Tool', '3:38');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 3, 3, 'Coincidence', '2:44');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 3, 3, 'Bed Chem', '2:51');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 3, 3, 'Espresso', '2:55');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 3, 3, 'Dumb n Poetic', '2:13');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 3, 3, 'Slim Pickins', '2:32');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 3, 3, 'Juno', '3:43');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 3, 3, 'Lie to girls', '3:22');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 4, 'BLOOD.', '1:58');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 4, 'DNA', '3:05');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 4, 'YAH.', '2:40');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 4, 'ELEMENT.', '3:28');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 4, 'FEEL.', '3:34');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 4, 'LOYALTY.', '3:47');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 6, 4, 'LOYALTY.', '3:47');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 4, 'PRIDE.', '4:35');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 4, 'HUMBLE', '6:22');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 4, 'LUST.', '5:07');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 4, 'LOVE.', '3:33');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 4, 'XXX.', '4:14');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 4, 'FEAR.', '7:40');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 4, 'GOD.', '4:08');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 4, 'DUCKWORTH.', '4:08');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 4, 5, 'cardigan', '3:59');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 4, 5, 'august', '4:21');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 4, 5, 'exile (featuring Bon Iver)', '4:45');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 4, 5, 'my tears ricochet', '4:15');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 4, 5, 'mirrorball', '3:29');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 4, 5, 'seven', '3:28');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 4, 5, 'this is me trying', '3:15');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 4, 5, 'illicit affairs', '3:10');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 4, 5, 'betty', '4:54');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 4, 5, 'peace', '5:23');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 4, 5, 'hoax', '3:40');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 5, 11, 'two', '3:12');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 5, 11, 'three', '2:57');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 5, 11, 'four', '3:22');

```

```

INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 5, 11, 'five', '4:01');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 5, 11, 'six', '3:44');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 5, 11, 'seven', '3:18');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 5, 11, 'eight', '2:55');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 7, 12, 'Pink + White', '3:04');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 7, 12, 'Solo', '4:17');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 7, 12, 'Self Control', '4:09');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 7, 12, 'Nights', '5:07');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 7, 12, 'White Ferrari', '4:08');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 7, 12, 'Godspeed', '2:57');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 7, 12, 'Futura Free', '9:24');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 13, 'BLOOD.', '1:58');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 13, 'DNA.', '3:05');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 13, 'ELEMENT.', '3:28');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 13, 'FEEL.', '3:34');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 13, 'HUMBLE.', '2:57');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 13, 'XXX.', '4:14');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 2, 13, 'FEAR.', '7:40');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 9, 15, 'After Hours', '6:01');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 9, 15, 'Heartless', '3:18');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 9, 15, 'In Your Eyes', '3:57');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 9, 15, 'Too Late', '3:59');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 9, 15, 'Hardest To Love', '3:31');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 9, 15, 'Scared To Live', '3:11');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 9, 15, 'Snowchild', '4:07');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 9, 15, 'Escape From LA', '5:55');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 9, 15, 'Faith', '4:43');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 8, 16, 'Therefore I Am', '2:54');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 8, 16, 'Your Power', '4:05');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 8, 16, 'Lost Cause', '3:32');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 8, 16, 'my future', '3:30');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 8, 16, 'Oxytocin', '3:30');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 8, 16, 'GOLDWING', '2:31');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 8, 16, 'Billie Bossa Nova', '3:16');
INSERT INTO MELODIE VALUES (SEQ_MELODIE.NEXTVAL, 8, 16, 'Getting Older', '4:04');

COMMIT;

```

12 Cereri SQL

12.1 Cererea nr. 1 - Cererea din feedback

Care sunt albumele care se pot achizitiona in cel putin doua formate diferite, si care sunt aduse de cel putin doi furnizori. In aceasta cerere am folosit 2 subinterrogari. Prima returneaza albumele ce se gasesc in cel putin 2 formaturi media, grupandu-le dupa id_product si numarand cate randuri distincte are fiecare. A doua functioneaza asemanator, dar cu datele din tabela STOCK.

```

SELECT a.id_produs, a.nume_album
FROM ALBUM a
WHERE a.id_produs IN (
    SELECT m.id_produs
    FROM FORMAT_MEDIA m
    GROUP BY m.id_produs
    HAVING COUNT(DISTINCT m.id_format_media) >= 2
)
AND a.id_produs IN (
    SELECT s.id_produs
    FROM STOC s

```

```

        GROUP BY s.id_produs
        HAVING COUNT(DISTINCT s.id_furnizor) >= 2
);

```

The screenshot shows a SQL worksheet interface with multiple tabs at the top: 'Gestiunea_Unui_Records_Store', 'clear.sql', 'create_tables.sql', 'insert_in_tables.sql', 'queries.sql' (which is the active tab), and 'update_delete.sql'. Below the tabs is a toolbar with icons for running, saving, and zooming. The main area is divided into two panes: 'Worksheet' and 'Query Builder'. The 'Worksheet' pane contains the following SQL code:

```

--1
SELECT a.id_produs, a.numere_album
FROM ALBUM a
WHERE a.id_produs IN (
    SELECT m.id_produs
    FROM FORMAT_MEDIA m
    GROUP BY m.id_produs
    HAVING COUNT(DISTINCT m.id_format_media) >= 2
)
AND a.id_produs IN (
    SELECT s.id_produs
    FROM STOC s
    GROUP BY s.id_produs
    HAVING COUNT(DISTINCT s.id_furnizor) >= 2
)

```

The 'Query Result' pane shows the output of the query:

ID_PRODUS	NUMERE_ALBUM
1	1 Swimming
2	15 After Hours

12.2 Cererea nr. 2 - Cereri sincronizate cu 3 tabele

Sa se afiseze numele albumului, pretul si genul muzical pentru toate albumele care au un pret mai mare sau egal cu pretul mediu al albumelor de același gen, unde media este calculată exclusiv pentru albumele interpretate de artiști care fac parte dintr-o trupa. Dacă pentru un anumit gen nu există astfel de albume, se vor afișa toate albumele din genul respectiv.

Aceasta interogare este sincronizata deoarece subcerere nu poate functiona independent. Interrogarea principală parcurge albumele pe randuri, iar la fiecare pas, ii trimite subcererii o valoarea **id-ului genului albumului**.

Subcererarea folosește aceasta valoare pentru a calcula media de pretului doar pentru acel gen. Se repetă pentru fiecare album, înseamnând că subcererarea este executată de mai multe ori.

```

SELECT a.numere_album, p.pret, g.numere_gen
FROM ALBUM a
JOIN PRODUS p ON a.id_produs = p.id_produs
JOIN GEN g ON a.id_gen = g.id_gen
WHERE p.pret >= NVL(
    SELECT AVG(p2.pret)
    FROM ALBUM a2
    JOIN PRODUS p2 ON a2.id_produs = p2.id_produs
    JOIN MELODIE m2 ON a2.id_produs = m2.id_produs
    JOIN ARTIST ar2 ON m2.id_artist = ar2.id_artist
    WHERE a2.id_gen = g.id_gen AND ar2.trupa IS NOT NULL ), 0);

```

The screenshot shows the Oracle SQL Developer interface. In the top window (Worksheet), there is a complex SQL query. The bottom window (Query Result) displays the output of the query, which is a table with three columns: NUME_ALBUM, PRET, and NUME_GEN. The data is as follows:

NUME_ALBUM	PRET	NUME_GEN
1 Swimming	49.99	RnB
2 GNX	49.99	Rap
3 DAMN.	47.99	Rap
4 Short n Sweet	39.99	Pop
5 HUMBLE	29.99	Rap-RNB
6 folklore	19.99	Folk
7 blonde - special edition	52.99	Alternative
8 the album	54.99	Indie
9 After Hours	37.99	Electronic

12.3 Cererea nr. 3 - Subcereri nesincronizare in FROM

Afisati numele formatului media, numarul de albume care se gasesc in respectivul format media, si media pretului tuturor albumelor din respectivul format media dupa adaugarea procentului la pretul de baza.

```

SELECT fm.nume_format_media,
       COUNT(a.id_produs) AS numar_albume,
       ROUND(AVG(p.pret * (1 + fm.procent_adaugat/100)), 2) AS pret_cu_adaos
FROM ALBUM a
JOIN PRODUS p ON a.id_produs = p.id_produs
JOIN (
       SELECT id_produs, nume_format_media, procent_adaugat
       FROM FORMAT_MEDIA
       WHERE procent_adaugat > 0
) fm ON fm.id_produs = a.id_produs
GROUP BY fm.nume_format_media
ORDER BY pret_cu_adaos DESC;
    
```

The screenshot shows a SQL Server Management Studio window. The top bar has tabs for 'Gestiunea_Unui_Records_Store' and several other scripts like 'dear.sql', 'create_tables.sql', etc. The main area is a 'Worksheet' tab with a 'Query Builder' interface. The query itself is:

```
--3
SELECT fm.nume_format_media,
       COUNT(a.id_produs) AS numar_albume,
       ROUND(AVG(p.pret * (1 + fm.procent_adaugat/100)), 2) AS pret_cu_adao
FROM ALBUM a
JOIN PRODUS p ON a.id_produs = p.id_produs
JOIN (
    SELECT id_produs, nume_format_media, procent_adaugat
    FROM FORMAT_MEDIA
    WHERE procent_adaugat > 0
) fm ON fm.id_produs = a.id_produs
GROUP BY fm.nume_format_media
ORDER BY pret_cu_adao DESC;
```

Below the code, the 'Query Result' tab is open, showing the following data:

NUME_FORMAT_MEDIA	NUMAR_ALBUME	PRET CU ADAOS
1 Digital	1	62.49
2 Vinyl	4	53.43
3 Vinyl 7"	1	51.59
4 CD	2	49.34
5 Cassette	4	42.19

12.4 Cererea nr.4 - Grupari cu subcereri in HAVING

Afisati genurile muzicale care au **numarul de melodii mai mare decat media numarului de melodii per album**. Pentru fiecare gen muzical afisat se cere si **numarul de albume distinste**, **numarul total de melodii si pretul mediu al albumelor pentru fiecare gen**. Rezultatele vor fi afisate descrescator dupa numarul de melodii.

Am folosit **ROUND(AVG(p.price), 2)** pentru a afisa media pretului rotunjita la 2 zecimale. In ultimele randuri, folosim o subcerere care intoarce o tabela cu numarul de albume de pe fiecare album. Acestei noi tabele ii aplicam functia **AVG(numar_melodii)** pentru a afla numarul mediu de melodii al albumelor. Acest rezultat este comparat in clauza **HAVING** cu **COUNT(s.id_song)**, conform cerintei.

```
SELECT g.nume_gen,
       COUNT(DISTINCT a.id_produs) AS numar_albume,
       COUNT(mel.id_melodie) AS numar_melodii,
       ROUND(AVG(p.pret), 2) AS pret_mediul
FROM GEN g
JOIN ALBUM a ON g.id_gen = a.id_gen
JOIN MELODIE mel ON mel.id_produs = a.id_produs
JOIN PRODUS p ON a.id_produs = p.id_produs
GROUP BY g.nume_gen
HAVING COUNT(mel.id_melodie) > (
    SELECT AVG(numar_melodii_per_album)
    FROM (
        SELECT COUNT(id_melodie) AS numar_melodii_per_album
        FROM MELODIE
        GROUP BY id_produs
    )
)
ORDER BY numar_melodii DESC;
```

```

...ge Gestiunea_Unui_Records_Store dear.sql create_tables.sql insert_in_tables.sql queries.sql update_delete.sql
SQL Worksheet History
Worksheet Query Builder
--4
SELECT g.num_gen,
       COUNT(DISTINCT a.id_produs) AS numar_albume,
       COUNT(mel.id_melodie) AS numar_melodii,
       ROUND(AVG(p.pret), 2) AS pret_mediul
FROM GEN g
JOIN ALBUM a ON g.id_gen = a.id_gen |
JOIN MELODIE mel ON mel.id_produs = a.id_produs
JOIN PRODUS p ON a.id_produs = p.id_produs
GROUP BY g.num_gen
HAVING COUNT(mel.id_melodie) > (
    SELECT AVG(numar_melodii_per_album)
)

```

Script Output | All Rows Fetched: 5 in 0.271 seconds

NUM_GEN	NUMAR_ALBUME	NUMAR_MELODII	PRET_MEDIU
1 Rap	2	18	49.21
2 Rap-RNB	1	15	29.99
3 RnB	1	13	49.99
4 Pop	1	11	39.99
5 Folk	1	11	19.99

12.5 Cererea nr. 5 - NVL, DECODE, Sortari

Afisati profilul unui cumparator, care cuprinde numele complet, orasul acestuia, numarul de achizitii, numarul de produse cumparate, suma totala cheltuita, data ultimei achizitii, si catalogatii in functie de numarul de albume cumparate(Nu merchandise!) astfel: 0: "Client nou", mai mare de 2: Ascultator ocazional, restul: Ascultator Casual. Soratati descrescator in functie de numarul de achizitii, in caz de egalitate, sortati crescator dupa numarul de bani cheltuiti.

Pentru a crea un profil ce contine numele complet pe o coloana, am concatenat `last_name` cu `first_name` folosindu-ma de "|||". Vrem sa calculam suma totala cheltuita pe site, iar pentru a evita cazurile NULL, folosim `NVL(SUM(p.quantity * pur.price), 0)`, care inlocuieste NULL cu 0. Asemanator, in cazul in care cumparatorul nu are nicio achizitie, `NVL(MAX(TO_CHAR(p.purchase_date, 'YYYY-MM-DD')), 'Client nou')`, inlocuieste o data eronata cu "Client nou". `DECODE` ne ajuta sa asociem un "tip_client" cumparatorului in functie de numarul de albume cumparate. Gruparea se face pe fiecare client, dupa cele 3 coloane care ne intereseaza din client, (nume + prenume + oras), ca sa putem calcula totaluri individuale. Conform enuntului, sortarea se face dupa 2 criterii, folosind doi parametrii in clauza `ORDER BY`.

```

SELECT
    c.nume_familie || ', ' || c.prenume AS nume_client,
    c.oras,
    COUNT(ach.id_achizitie) AS numar_achizitii,
    SUM(ach.cantitate) AS total_articole,
    NVL(SUM(ach.cantitate * prod.pret), 0) AS total_cheltuit,
    NVL(MAX(TO_CHAR(ach.data_achizitie, 'YYYY-MM-DD')), 'Nicio achizitie') AS ultima_achizitie,
    DECODE(COUNT(DISTINCT alb.id_produs), 0, 'Client nou',
           CASE WHEN COUNT(DISTINCT alb.id_produs) > 1 THEN 'Ascultator frecvent'
                 ELSE 'Ascultator ocazional' END) AS tip_client,
    COUNT(DISTINCT alb.id_produs)
FROM CUMPARATOR c
LEFT JOIN ACHIZITIE ach ON c.id_cumparator = ach.id_cumparator

```

```
LEFT JOIN PRODUS prod ON ach.id_produs = prod.id_produs
LEFT JOIN ALBUM alb ON prod.id_produs = alb.id_produs
GROUP BY c.nume_familie, c.prenume, c.oras
ORDER BY numar_achizitii DESC, total_cheltuit ASC;
```

The screenshot shows the Oracle SQL Developer interface. In the top-left corner, there's a toolbar with various icons. Below it, the 'Worksheet' tab is selected. The main area contains a SQL query:

```
ORDER BY numar_melodii DESC;
--5
SELECT
    c.nume_familie || ', ' || c.prenume AS nume_client,
    c.oras,
    COUNT(ach.id_achizitie) AS numar_achizitii,
    SUM(ach.cantitate) AS total_articole,
    NVL(SUM(ach.cantitate * prod.pret), 0) AS total_cheltuit,
    NVL(MAX(TO_CHAR(ach.data_achizitie, 'YYYY-MM-DD')), 'Nicio achizitie') AS ultima_achizitie,
    DECODE(COUNT(DISTINCT alb.id_produs), 0, 'Client nou',
           CASE WHEN COUNT(DISTINCT alb.id_produs) > 1 THEN 'Afcultator frecvent'
                ELSE 'Afcultator ocazional' END) AS tip_client,
    COUNT(DISTINCT alb.id_produs)
```

Below the code, the 'Query Result' tab is selected, showing the following data:

NUME_CLIENT	ORAS	NUMAR_ACHIZITII	TOTAL_ARTICOLE	TOTAL_CHELTUIT	ULTIMA_ACHIZITIE	TIPI_CLIENT
1 Radu, Ioana	Cluj-Napoca	3	3	154.97	2025-06-03	Afcultator frecvent
2 Stoica, Gabriel	Iași	3	5	249.95	2025-06-03	Afcultator frecvent
3 Dumitrescu, Alexandru	București	2	2	69.98	2023-05-15	Afcultator frecvent
4 Munteanu, Diana	Timișoara	2	2	104.98	2023-06-25	Afcultator frecvent
5 Manole, Cristina	Sibiu	1	1	37.99	2023-06-18	Afcultator ocazional
6 Vasile, Andreea	Constanța	1	1	52.99	2023-06-02	Afcultator ocazional
7 Preda, Robert	Craiova	1	2	95.98	2023-06-10	Afcultator ocazional
8 Dinu, Bogdan	Brașov	1	3	134.97	2023-05-10	Client nou

12.6 Cererea nr. 6 - Clauza WITH

Afișați numele complet al angajatilor care au procesat comenzi, denumirea functiei lor, valoarea totala a comenziilor procesate, categoria de activitate in functie de valoarea vanzarilor și luna in care au fost angajati(Mai mare de 500: Activ, Mai mare de 1000: Performant, altfel: Slab activ). Afisati si salariul total primit de la firma de la angajarea acestuia. Ordonati des crescator dupa valoarea vanzarilor și limitati rezultatul la primii 15 angajati. Clauza **WITH** ne realizeaza un tabel temporar cu valoarea totala a produselor din comenziile aflate sub evidenta fiecarui angajat pentru a fi accesate anterior. Salariul angajatului de cand este angajat in firma este calculat cu diferite functii de calcul cu date precum **MONTHS_BETWEEN**(calculeaza numarul de luni dintre 2 date) si **SYSDATE**(returneaza data actuala la runtime). Subcererea din **WHERE** verifica daca angajatul luat in calcul, s-a ocupat de cel putin o comanda.

```
WITH total_vanzari_per_angajat AS (
    SELECT
        ach.id_angajat,
        SUM(prod.pret * ach.cantitate) AS valoare_totala
    FROM ACHIZITIE ach
    JOIN PRODUS prod ON ach.id_produs = prod.id_produs
    GROUP BY ach.id_angajat
)
SELECT *
FROM (
    SELECT
        ang.prenume || ' ' || NVL(ang.nume_familie, 'anonim') AS nume_complet,
```

```

f.numa_functie,
ROUND(tva.valoare_totala, 2) AS valoare_totala_vanzari,
CASE
    WHEN tva.valoare_totala > 1000 THEN 'Performant'
    WHEN tva.valoare_totala > 500 THEN 'Activ'
    ELSE 'Slab activ'
END AS categorie_angajat,
TO_CHAR(ang.data_angajare, 'Mon YYYY', 'NLS_DATE_LANGUAGE=Romanian') AS angajat_din_luna,
ROUND(ang.salariu * MONTHS_BETWEEN(SYSDATE, ang.data_angajare), 2) AS salariu_estimat_total
FROM ANGAJAT ang
JOIN FUNCTIE f ON ang.id_functie = f.id_functie
JOIN total_vanzari_per_angajat tva ON ang.id_angajat = tva.id_angajat
WHERE EXISTS (
    SELECT 1
    FROM ACHIZITIE ach_check
    WHERE ach_check.id_angajat = ang.id_angajat
)
ORDER BY tva.valoare_totala DESC
)
WHERE ROWNUM <= 15;

```

The screenshot shows the Oracle SQL Developer interface. The top navigation bar has tabs for 'clear.sql', 'create_tables.sql', 'insert_in_tables.sql', 'queries.sql' (which is currently selected), and 'update_delete.sql'. Below the tabs is a toolbar with various icons. The main area is divided into two panes: 'Worksheet' (Query Builder) and 'Script Output'. The 'Worksheet' pane contains the SQL code provided above. The 'Script Output' pane shows the results of the query, which is a table with six columns: NUME_COMPLET, NUME_FUNCTIE, VALOARE_TOTALA_VANZARI, CATEGORIE_ANAJAT, ANGAJAT_DIN_LUNA, and SALARIU_ESTIMAT_TOTAL. The data is as follows:

NUME_COMPLET	NUME_FUNCTIE	VALOARE_TOTALA_VANZARI	CATEGORIE_ANAJAT	ANGAJAT_DIN_LUNA	SALARIU_ESTIMAT_TOTAL
1 Elena Popa	Cashier	374.93	Slab activ	Mai 2021	143497.98
2 Andrei Ionescu	Sales Representative	227.94	Slab activ	Mar 2020	248879.02
3 Cristina Stanciu	Store Assistant	95.98	Slab activ	Dec 2021	103452.61
4 Mihai Georgescu	Marketing Specialist	52.99	Slab activ	Aug 2021	224969.74

13 Cereri de actualizare si suprimare a datelor

13.1 Cererea nr.1

Sa se mareasca pretul tuturor albumelor care au unul dintre genurile muzicale "Pop" cu 20%.

```

UPDATE PRODUS
SET pret = pret * 1.2
WHERE id_produs IN (

```

```

SELECT alb.id_produs
FROM ALBUM alb
JOIN GEN g ON alb.id_gen = g.id_gen
WHERE UPPER(g.numar_gen) = 'POP'
);

```

The screenshot shows a SQL worksheet window with the following content:

```

--1
UPDATE PRODUS
SET pret = pret * 1.2
WHERE id_produs IN (
    SELECT alb.id_produs
    FROM ALBUM alb
    JOIN GEN g ON alb.id_gen = g.id_gen
    WHERE UPPER(g.numar_gen) = 'POP'
);

--2
UPDATE ANGAJAT
SET salariu = salariu * 0.75
WHERE id_functie IN (
    SELECT id_functie
    FROM FUNCTIE
    WHERE numar_functie = 'IT Support'
);

```

Script Output:

```

1 row updated.

```

13.2 Cererea nr. 2

Sa se reduca salariul angajatilor din departamentul de "IT Support" cu 20%.

```

UPDATE ANGAJAT
SET salariu = salariu * 0.75
WHERE id_functie IN (
    SELECT id_functie
    FROM FUNCTIE
    WHERE numar_functie = 'IT Support'
);

```

```

...ge Gestiunea_Unui_Records_Store dear.sql create_tables.sql insert_in_tables.sql queries.sql update_delete.sql
SQL Worksheet History
Worksheet Query Builder
JOIN GEN g ON alb.id_gen = g.id_gen
WHERE UPPER(g.numere_gen) = 'POP'
);
--2
UPDATE ANGAJAT
SET salariu = salariu * 0.75
WHERE id_functie IN (
    SELECT id_functie
    FROM FUNCTIE
    WHERE nume_functie = 'IT Support'
);
--3
--a)
UPDATE STOC s

```

Script Output | Task completed in 0.117 seconds

```

1 row updated.

1 row updated.

```

13.3 Cererea nr. 3

Sa se reactualizeze stocurile de produse in functie comenziile prezente in 'Pending', urmand ca statusul acestora sa fie schimbat in 'Completed'. In cazul in care un stoc are 0 produse(Este epuizat), stergeti definitiv stocul respectiv.

Subcererea din **SET** ne calculeaza numarul de produse vandute aflate doar in comenziile ce asteapta sa fie trimise ("Pending") si actualizeaza doar un singur stoc de acest produs.In a treia subcerere, folosind **SELECT MIN(s2.id_stoc)** si metoda in care sunt adaugate date in tabele cu sechete, ne asiguram ca scadem cantitatea din cel mai vechi stoc de acel produs, pentru a respecta principiul **first in first out**. Ulterior, toate comenziile "Pending" sunt transformate in "Completed", iar in cazul in care un stoc este epuizat complet si nu mai este nevoie de el, este sters cu total din tabela **STOCK**.

```

--a)
UPDATE STOC s
SET s.cantitate = s.cantitate - (
    SELECT SUM(ach.cantitate)
    FROM ACHIZITIE ach
    WHERE ach.status = 'Pending'
    AND ach.id_produs = s.id_produs
)
WHERE EXISTS (
    SELECT 1
    FROM ACHIZITIE ach
    WHERE ach.status = 'Pending'
    AND ach.id_produs = s.id_produs
)
AND s.id_stoc = (
    SELECT MIN(s2.id_stoc)
    FROM STOC s2
    WHERE s2.id_produs = s.id_produs
)

```

```
);
UPDATE ACHIZITIE
SET status = 'Completed'
WHERE status = 'Pending';
--b)
DELETE FROM STOC
WHERE cantitate <= 0;
```

The screenshot shows the SSMS interface with two main panes. The top pane is the 'Worksheet' tab, which contains the SQL code provided above. The bottom pane is the 'Script Output' tab, which displays the results of the executed query. The output shows:

```
1 row updated.

0 rows updated.

0 rows updated.

0 rows deleted.
```

14 Tabele view si operatii LMD

Generati o tabela view care sa afiseze detalii despre toatele produsele de tip album, calculand acestora **pretul final de vanzare raportat la formatul media in care se gasesc**.

```
CREATE VIEW ALBUMEDETALIATE AS
SELECT
    a.id_produs,
    a.nume_album,
    g.nume_gen,
    f.nume_format_media,
    p.pret AS pret_baza,
    f.procent_adaugat,
    (p.pret * (1 + f.procent_adaugat / 100)) AS pret_final_calculat
FROM ALBUM a
JOIN PRODUS p ON a.id_produs = p.id_produs
JOIN GEN g ON a.id_gen = g.id_gen
JOIN FORMAT_MEDIA f ON a.id_produs = f.id_produs
WHERE p.data_lansare >= TO_DATE ('2015-01-01', 'YYYY-MM-DD');
```

The screenshot shows the Oracle SQL Developer interface. In the top tab bar, there are several tabs: Welcome Page, maomor, Gestiunea_Unui_Record_Store.sql, Gestiunea_Unui_Records_Store, and view.sql. The view.sql tab is active.

In the main workspace, under the 'Worksheet' tab, the following SQL code is displayed:

```

CREATE VIEW ALBUMEDETALIATE AS
SELECT
    a.id_produs,
    a.numere_album,
    g.numere_gen,
    f.numere_format_media,
    p.pret AS pret_baza,
    f.procent_adaugat,
    (p.pret * (1 + f.procent_adaugat / 100)) AS pret_final_calculat
FROM ALBUM a
JOIN PRODUS p ON a.id_produs = p.id_produs
JOIN GEN g ON a.id_gen = g.id_gen
JOIN FORMAT_MEDIA f ON a.id_produs = f.id_produs
WHERE p.data_lansare >= TO_DATE ('2015-01-01', 'YYYY-MM-DD');

```

Below the code, the 'Script Output' tab shows the execution results:

```

SQL Error: ORA-01779: cannot modify a column which maps to a non key-preserved table
https://docs.oracle.com/error-help/db/ora-01779/01779. 00000 - "cannot modify a column which maps to a non key-preserved table"
*Cause: An attempt was made to insert or update columns of a join view which
map to a non-key-preserved table.
>Action: Modify the underlying base tables directly.

More Details :
https://docs.oracle.com/error-help/db/ora-01779/

View ALBUMEDETALIATE created.

```

The screenshot shows the Oracle SQL Developer interface with the ALBUMEDETALIATE view selected in the top tab bar. The main workspace displays the data from the view in a grid format:

ID_PRODUS	NUME_ALBUM	NUME_GEN	NUME_FORMAT_MEDIA	PRET_BAZA	PROCENT_ADAUGAT	PRET_FINAL_CALCULAT
1	1 Swimming	RnB	CD	49.99	10	54.989
2	1 Swimming	RnB	Vinyl	49.99	25	62.4875
3	1 Swimming	RnB	Cassette	49.99	25	62.4875
4	1 Swimming	RnB	Digital	49.99	25	62.4875
5	2 GNX	Rap	Vinyl	49.99	20	59.988
6	3 Short n Sweet	Pop	Digital	39.99	0	39.99
7	3 Short n Sweet	Pop	CD	39.99	0	39.99
8	4 HUMBLE	Rap-RNB	Cassette	29.99	5	31.4895
9	5 folklore	Folk	Digital	19.99	0	19.99
10	5 folklore	Folk	Vinyl	19.99	25	24.9875
11	11 the album	Indie	Digital	54.99	0	54.99
13	12 Blonde	Alternative	Digital	52.99	0	52.99
14	12 Blonde	Alternative	Vinyl	52.99	25	66.2375
15	13 DAMN.	Rap	Cassette	47.99	10	52.789
16	15 After Hours	Electronic	Digital	37.99	0	37.99
17	15 After Hours	Electronic	CD	37.99	15	43.6885
18	16 Happier Than Ever	Alternative Pop	Digital	42.99	0	42.99
19	16 Happier Than Ever	Alternative Pop	Vinyl 7"	42.99	20	51.588

O operatie LMD nepermisa ar fi una care modifica o coloana a tabelului view derivat, ale carei valori sunt rezultatul unei expresii dependente de coloanele tabelului de baza. Un exemplu ar fi acesta:

```

UPDATE ALBUMEDETALIATE
SET pret_final_calculat = 75.00

```

```
WHERE id_producator = 12;
```

The screenshot shows a SQL developer interface with multiple tabs at the top: Welcome Page, maomor, Gestiunea_Unui_Record_Store.sql, Gestiunea_Unui_Records_Store, view.sql, ALBUMEDETALIATE, and Gestiunea_Unui_Records_Store. The Worksheet tab is active, displaying the following SQL code:

```

WHERE p.data_lansare >= TO_DATE ('2015-01-01', 'YYYY-MM-DD');

--Operatie LMD nepermisa
UPDATE ALBUMEDETALIATE
SET nume_album = 'blonde - special edition'
WHERE id_producator = 12;

--sau
UPDATE ALBUMEDETALIATE
SET pret_final_calculat = 75.00
WHERE id_producator = 12;

--Operatie LMD Permisă
UPDATE ALBUM
SET nume_album = 'blonde - special edition' --

```

The bottom pane, Script Output, shows the error message:

```

Error at Command Line : 23 Column : 5
Error report -
SQL Error: ORA-01733: virtual column not allowed here

https://docs.oracle.com/error-help/db/ora-01733/01733. 00000 - "virtual column not allowed here"
*Cause: An attempt was made to use an INSERT, UPDATE, or DELETE
        statement on an expression in a view.
*Action: INSERT, UPDATE, or DELETE data in the base tables,
        instead of the view.

More Details :
https://docs.oracle.com/error-help/db/ora-01733/

```

Un alt exemplu de operatie LMD nepermisa ar fi modificarea unei coloane din tabelul derivat care mosteneste din tabelul de baza o coloana cu id-uri unice, dar se multiplica in urma operatiei JOIN. In contextul unui view, unicitatea nu mai este data de id si este creata ambiutate.

```

UPDATE ALBUMEDETALIATE
SET nume_album = 'blonde - special edition'
WHERE id_producator = 12;

```

The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, several tabs are open: Welcome Page, maomor, Gestiunea_Unui_Record_Store.sql, Gestiunea_Unui_Records_Store, view.sql, ALBUMEDETALIATE, and Gestiunea_Unui_Records_Store. The main area has two tabs: Worksheet and Query Builder. The Worksheet tab contains the following SQL code:

```

JOIN FORMAT_MEDIA f ON a.id_produs = f.id_produs
WHERE p.data_lansare >= TO_DATE ('2015-01-01', 'YYYY-MM-DD');

--Operatie LMD nepermisa
UPDATE ALBUMEDETALIATE
SET nume_album = 'blonde - special edition'
WHERE id_produs = 12;

--sau
UPDATE ALBUMEDETALIATE
SET pret_final_calculat = 75.00
WHERE id_produs = 12;

--Operatie LMD Permisa
UPDATE ALBUM

```

In the Script Output tab, the results of the execution are shown:

```

Task completed in 0.384 seconds
WHERE id_produs = 12
Error at Command Line : 18 Column : 5
Error report -
SQL Error: ORA-01779: cannot modify a column which maps to a non key-preserved table

https://docs.oracle.com/error-help/db/ora-01779/01779. 00000 - "cannot modify a column which maps to a non key-preserved table"
*Cause: An attempt was made to insert or update columns of a join view which
map to a non-key-preserved table.
>Action: Modify the underlying base tables directly.

More Details :
https://docs.oracle.com/error-help/db/ora-01779/

```

Rezolvarea acestei probleme este realizarea operatiei LMD direct pe clasele de baza ale view-ului.

```

UPDATE ALBUM
SET nume_album = 'blonde - special edition'
WHERE id_produs = 12;

```

The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, several tabs are open: Welcome Page, maomor, Gestiunea_Unui_Record_Store.sql, Gestiunea_Unui_Records_Store, view.sql, ALBUMEDETALIATE, and Gestiunea_Unui_Records_Store. The main area has two tabs: Worksheet and Query Builder. The Worksheet tab contains the following SQL code:

```
--Operatie LMD nepermisa
UPDATE ALBUMEDETALIATE
SET nume_album = 'blonde - special edition'
WHERE id_producator = 12;

--sau
UPDATE ALBUMEDETALIATE
SET pret_final_calculat = 75.00
WHERE id_producator = 12;

--Operatie LMD permisa
UPDATE ALBUM
SET nume_album = 'blonde - special edition' --
WHERE id_producator = 12;
```

The bottom part of the screenshot shows the Script Output tab with the following error message:

```
SQL Error: ORA-01779: cannot modify a column which maps to a non key-preserved table
https://docs.oracle.com/error-help/db/ora-01779/00000 - "cannot modify a column which maps to a non key-preserved table"
*Cause: An attempt was made to insert or update columns of a join view which
map to a non-key-preserved table.
*Action: Modify the underlying base tables directly.

More Details :
https://docs.oracle.com/error-help/db/ora-01779/
1 row updated.
```

15 Cereri SQL folosind operatiile, outer join, division si top-n

15.1 Outer join

Afișați toate articolele de merchandise. Pentru fiecare, includeti prețul de bază, tipul de merchandise și marimea (**care poate fi nula**). Dacă articolul este în stoc, afișați și numele furnizorului pentru acel stoc. Dacă un articol de merchandise nu este în stoc sau stocul respectiv nu are un furnizor specificat, articolul trebuie totuși listat, cu informațiile despre furnizor goale (NULL).

The screenshot shows a SQL worksheet interface with multiple tabs at the top: ...or, Gestiu..._Record_Store.sql, Gestiu..._Records_Store, view.sql, ALBUMDETALIATE, outerjoinindivisiontopn.sql, and Gestiu..._Records_Store. The main area is titled 'Worksheet' and contains a 'Query Builder' tab. The query code is:

```
-- Outer join
SELECT
    m.id_produs,
    p.pret AS pret_baza,
    m.tip_merch,
    m.marime_merch,
    a.id_stoc,
    f.nume_furnizor
FROM
    MERCHANDISE m
JOIN PRODUS p ON m.id_produs = m.id_produs
LEFT OUTER JOIN STOC s ON m.id_produs = s.id_produs
LEFT OUTER JOIN FURNIZOR f ON s.id_furnizor = f.id_furnizor;
```

Below the code is a 'Script Output' window showing the results of the query:

ID_PRODUS	PRET_BAZA	TIP_MERCH	MARI
6	59.99	T-Shirt	M
7	34.99	Poster	S
8	44.99	Cap	XL
9	24.99	Mug	CL
10	64.99	Hoodie	
14	22.99	Vinyl Stand	

6 rows selected.

15.2 Division

Gasiti artiștii care au înregistrat melodii pe **toate** albumele produse de casa de discuri 'Universal Music'.

The screenshot shows a SQL worksheet interface with multiple tabs at the top: Worksheet, Query Builder, and others. The main area is titled 'Worksheet' and contains a 'Query Builder' tab. The query code is:

```
--Division
SELECT
    a.id_artist,
    a.prenume,
    a.nume_familie
FROM
    ARTIST a
WHERE NOT EXISTS (
    SELECT al.id_produs
    FROM ALBUM al
    JOIN CASA_DE_DISCURI c ON al.id_casa_discuri = c.id_casa_discuri
    WHERE c.nume_casa = 'Universal Music'
    AND NOT EXISTS (
        SELECT m.id_melodie
        FROM MELODIE m
        WHERE m.id_produs = al.id_produs
    )
);
```

Below the code is a 'Script Output' window showing the results of the query:

ID_ARTIST	PRENUME	NUME_FAMILIE
1	5 boygenius	boygenius

All Rows Fetched: 1 in 0.066 seconds

15.3 Analiza Top-n

Afișați primii 5 angajați cu cele mai mari salarii.

```

SQL Worksheet History
Worksheet Query Builder
SELECT
    prenume, nume_familie,
    nume_functie,
    salariu
  FROM (
    SELECT
        a.prenume , a.nume_familie,
        f.nume_functie,
        a.salariu
      FROM ANGAJAT a
      JOIN FUNCTIE f ON a.id_functie = f.id_functie
     ORDER BY a.salariu DESC
  )
 WHERE ROWNUM <= 5;
  
```

Script Output | Query Result | Query Result 1

All Rows Fetched: 5 in 0.115 seconds

	PRENUME	NUME_FAMILY	NUME_FUNCTIE	SALARIU
1	Maria	Popescu	Manager	6500
2	Diana	Marinescu	Inventory Manager	5500
3	Mihai	Georgescu	Marketing Specialist	5000
4	Alexandru	Dumitrache	IT Support	4500
5	Andrei	Ionescu	Sales Representative	4000

16 Optimizari

16.1 Cererea

Afișați pretul de baza și numele albumelor lui 'Kendrick Lamar' care sunt produse de 'Atlantic Records'.

16.2 Cererea SQL neoptimizata

```

SELECT DISTINCT
    a.nume_album,
    p.pret
  FROM
    ARTIST ar,
    MELODIE m,
    ALBUM a,
    PRODUS p,
    CASA_DE_DISCURI c
 WHERE
    ar.id_artist = m.id_artist AND
    m.id_produs = a.id_produs AND
    a.id_produs = p.id_produs AND
    a.id_casa_discuri = c.id_casa_discuri AND
    ar.prenume = 'Kendrick' AND ar.nume_familie = 'Lamar' AND c.nume_casa = 'Atlantic Records';
  
```

16.3 Cererea SQL optimizata

```

SELECT DISTINCT
    a.numere_album,
    p.pret
FROM ARTIST ar
JOIN MELODIE m ON ar.id_artist = m.id_artist
JOIN ALBUM a ON m.id_produs = a.id_produs
JOIN PRODUS p ON a.id_produs = p.id_produs
JOIN CASA_DE_DISCURI cd ON a.id_casa_discuri = cd.id_casa_discuri
WHERE ar.prenume = 'Kendrick' AND ar.numere_familie = 'Lamar' AND cd.numere_casa = 'Atlantic Records';

```

16.4 Expresii algebrice

```

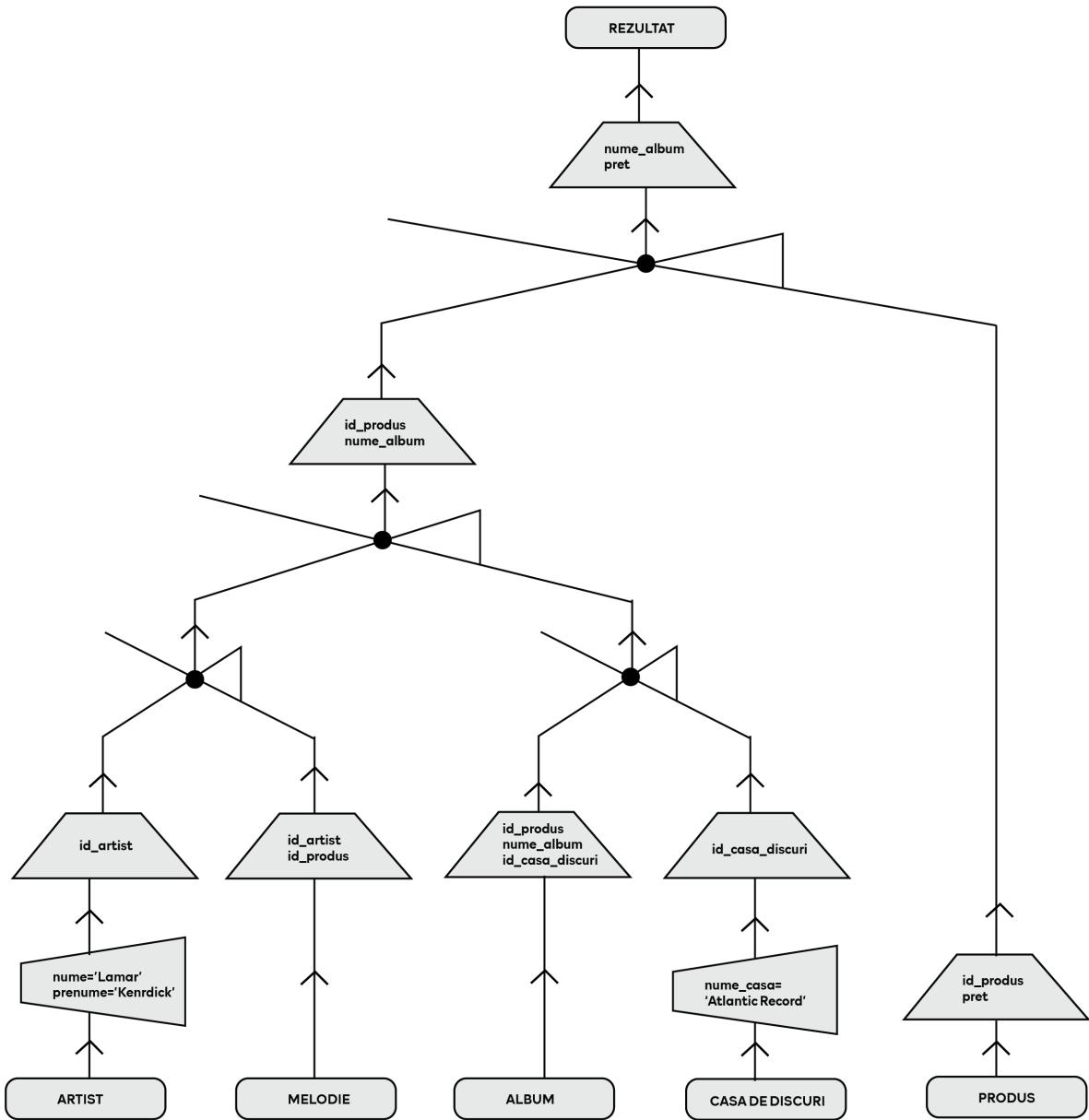
R1 = SELECT(ARTIST, prenume='Kendrick' AND numere_familie='Lamar')
      -Eliminam partile de care nu avem nevoie
R2 = PROJECT(R1, id_artist)
R3 = SELECT(CASA_DE_DISCURI, numere_casa='Atlantic Records')
R4 = PROJECT(R3, id_casa_discuri)
R5 = PROJECT(MELODIE, id_artist, id_produs)
R6 = JOIN(R2, R5, R2.id_artist = R5.id_artist)
R7 = PROJECT(R6, id_produs)
R8 = PROJECT(ALBUM, id_produs, numere_album, id_casa_discuri)
R9 = JOIN(R8, R4, R8.id_casa_discuri = R4.id_casa_discuri)
R10 = PROJECT(R9, id_produs, numere_album)
R11 = JOIN(R7, R10, R7.id_produs = R10.id_produs)
R12 = PROJECT(PRODUS, id_produs, pret)
R13 = JOIN(R11, R12, R11.id_produs = R12.id_produs)
Rezulta = PROJECT(R13, numere_album, pret)

```

16.5 Optimizare

1. Selectiile se executa cat mai devreme pentru a scoate din context elementele care nu sunt necesare.
2. Produsul cartezian a fost inlocuit cu JOIN.
3. Proiectiile se executa la inceput.

16.6 Arboare algebric



17 Realizarea normalizarii BCNF, FN4, FN5 si aplicarea denormalizarii

17.1 Forma normala Boyce-Codd

Determinantul este un atribut sau o multime de attribute neredundante, care constituie un identificator unic pentru alt atribut sau alta multime de attribute ale unei relatii date.

Intuitiv, o relatie R este in forma normala Boyce-Codd daca si numai daca fiecare determinant este o cheie candidata.

Formal, o relatie R este in forma normala Boyce-Codd daca si numai daca pentru orice dependenta functionala non-triviala $X \rightarrow Y$, X este o cheie candidata a lui R.

Fie urmatoare relatie: **R(id_melodie#, id_artist#, nume_familie_artist, prenume_artist, id_producus_album, nume_album, titlu_melodie)** - Un artist are mai multe melodii pe un album.

$\{id_melodie\#} \rightarrow \{id_artist, nume_familie_artist, prenume_artist, id_produs_album, nume_album, titlu_melodie\}$

$\{id_artist\} \rightarrow \{nume_familie_artist, prenume_artist\}$

$\{id_produs_album\} \rightarrow \{nume_album\}$

Aplicand regulile de descompunere BCNF optinem:

id_melodie#	id_artist	nume_familie_artist	prenume_artist	id_produs_album	nume_album	titlu_melodie
M1	A1	Miller	Mac	AL1	Swimming	Come Back to Earth
M2	A1	Miller	Mac	AL1	Swimming	Hurt feelings
M3	A2	Kendrick	Lamar	AL2	DAMN.	DNA.
M4	A1	Miller	Mac	AL3	Circles	Circles

Table 9: Exemplu non-BCNF

ARTIST (id_artist, nume_familie_artist, prenume_artist)

ALBUM (id_producus_album, nume_album)

MELODIE(id_melodie, id_artist, id_producus_album, titlu_melodie)

id_artist#	nume_familie_artist	prenume_artist
A1	Miller	Mac
A2	Kendrick	Lamar

Table 10: ARTIST

id_producus_album#	nume_album
AL1	Swimming
AL2	DAMN.
AL3	Circles

Table 11: ALBUM

id_melodie#	id_artist#	id_producus_album#	titlu_melodie
M1	A1	AL1	Come back to earth
M2	A1	AL1	Hurt feelings
M3	A2	AL2	DNA.
M4	A1	AL3	Circles

Table 12: MELODIE

17.2 Formala normala 4 - FN4

FN4 elimina redundantele datorate relatiilor m:n, adica dependentele multivaloare.

Intuitiv, o relatie R este in forma normala 4 (FN4) daca si numai daca relata este in BCNF si nu contine dependente multivaloare independente (adica, daca o cheie candidata A determina un set de valori B si un set de valori C, iar seturile B si C sunt independente unul de celalalt, atunci A, B, si C nu ar trebui sa fie in aceeași tabela care are cheia A).

Fie urmatorul exemplu: un album poate avea mai multi artiști care au contribuit. Independent, același album poate fi disponibil în mai multe formate media.

Există urmatoarea relație: **R(ID_ALBUM#, ID_ARTIST#, ID_FORMAT_MEDIA#)**. În acest exemplu avem urmatoarele dependente multivaloare independente, unde ID_ALBUM# este determinantul:

$ID_ALBUM\# \rightarrow ID_ARTIST\#$ (Un album este asociat cu un set de artiști colaboratori)

$ID_ALBUM\# \rightarrow ID_FORMAT_MEDIA\#$ (Un album este asociat cu un set de formate media în care este disponibil)

id_album#	id_album#	id_format_- media#
A1	AR1	F1
A1	AR1	F2
A1	AR2	F2
A2	AR2	F2
A2	AR3	F2
A2	AR3	F3

Table 13: Forma non-FN4

Relația R se va descompune astfel pentru a aduce în FN4:

R1(ID_ALBUM#, ID_ARTIST_COLAB#)

R2(ID_ALBUM#, ID_FORMAT_DISPONIBIL#)

id_album#	id_artist#
A1	AR1
A1	AR2
A2	AR3

Table 14: R1

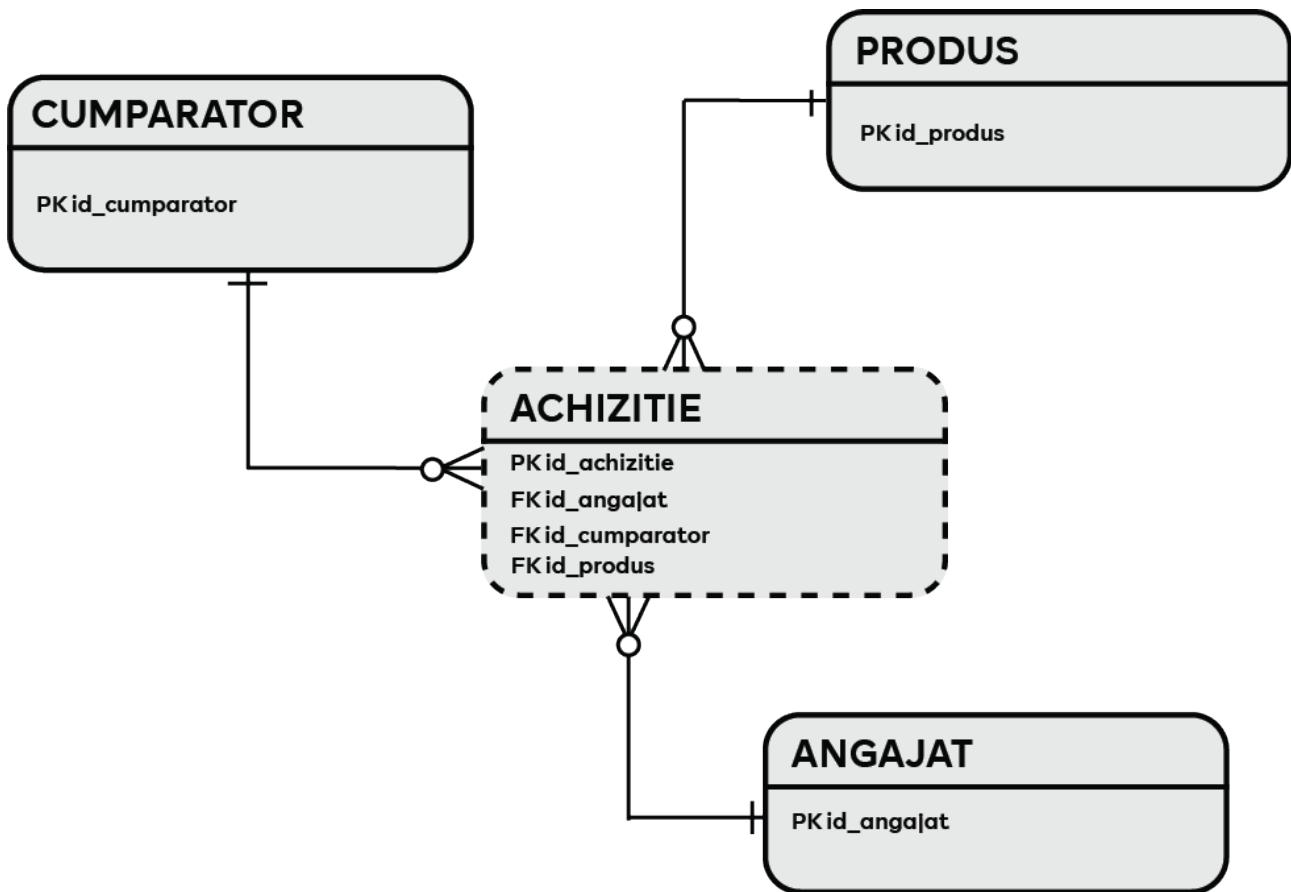
id_album#	id_format_me- dia#
A1	F1
A1	F2
A2	F1
A2	F3

Table 15: R2

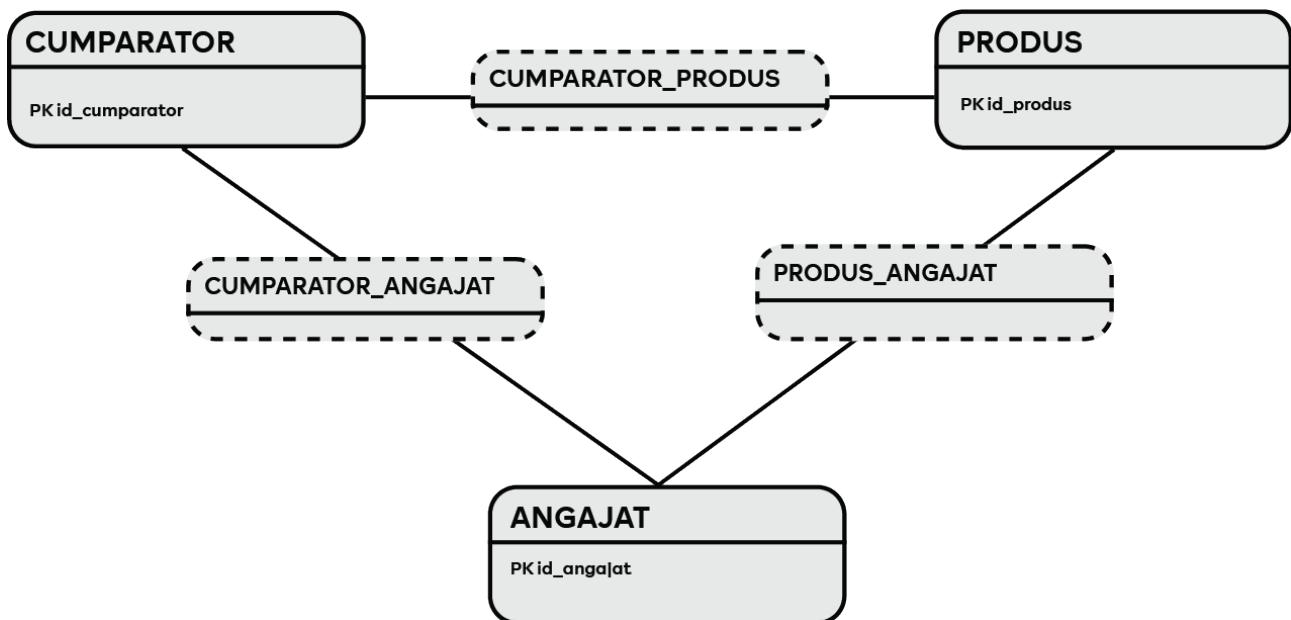
17.3 Forma normală 5 -FN5

FN5 își propune eliminarea redundanțelor care apar în relații ce implica dependențe m:n. Intuitiv, o relație R este în forma normală 5 dacă și numai dacă este deja în FN4 și nu conține dependențe ciclice.

Să considerăm urmatorul exemplu: un CUMPARATOR poate achiziționa mai multe PRODUSE, iar fiecare achiziție este gestionată de un ANGAJAT. Aceasta reprezintă o interacțiune centrală în magazinul de muzică, materializată prin tabela ACHIZITIE.



Relatia de tip 3 precedenta poate fi considerata echivalenta cu trei relatii de tip 2, mai ales daca am presupune ca o achizitie are loc doar daca exista deja anumite legaturi intre perechile de entitati.



Deoarece relatiile binare formeaza un ciclu, la efectuarea tuturor JOIN-urilor dintre ele, se vor genera date identice cu cele din relatia de tip 3. Pentru ca o relatia sa fie considerata FN5, ea trebuie sa indeplineasca conditiile FN4 si, in plus, sa nu prezinte dependiente ciclice.

Astfel, se constata ca cele trei relatii de tip 2 alcataiesc dependente ciclice. Prin urmare, aceste relatii de tip 2 nu ar respecta cerintele FN5 daca ar duce la redundanta care ar putea fi evitata prin utilizarea relatiei de

tip 3.

17.4 Aplicarea denormalizarii

Denormalizarea este un proces de optimizare a performantei unei baze de date prin introducerea intentionata a redundantei intr-o baza de date normalizata. Scopul este reducerea numarului de JOIN-uri necesare pentru a rezolva o interogare, prin includerea unor date in avans.

Se considera urmatorul exemplu: Avem tabela ALBUM (care contine nume_album) si tabela PRODUS (care contine pret, legat de ALBUM prin id_produc). Pentru a afisa frecvent numele albumului impreuna cu pretul sau, este necesar un JOIN intre ALBUM si PRODUS.

In acest caz, daca interogarea care combina nume_album cu pret este executata des, iar preturile albumelor nu se modifica des, dar performanta obtinuta prin JOIN nu este considerata optima pentru numarul mare de citiri, se poate aplica denormalizarea. Acest proces presupune adaugarea atributului pret direct in tabela ALBUM. Astfel, s-ar elibera nevoia de JOIN pentru aceasta interogare specifica, deoarece nu mai este la fel de eficient ca atributul pret sa fie accesat constant printr-un JOIN daca este necesar impreuna cu nume_album.

18 Exemplificare Isolation Levels

Intr-un sistem de gestiune al bazelor de date, nivelurile de izolare a tranzactiilor sunt modul in care operatiile dintr-o tranzactie sunt izolate de alte operatii din tranzactii concurente.

Cu alte cuvinte, definesc cum si cand schimbarile facute de o tranzactie sunt vizibile catre alte tranzactii, asigurand consistenta datelor.

Nivelul de izolare al unei tranzactii se defineste dupa urmatoarele fenomene:

1. Dirty Read

Un Dirty Read presupune situatia in care o tranzactie citeste date care nu au fost committed.

2. Non Repeatable read

Un Non Repeatable read apare cand o tranzactie citeste o valoare de 2 ori si primeste o valoare diferita de fiecare data.

3. Phantom Read

Un Phantom Read apare cand 2 interogari sunt executate, dar liniile folosite sunt diferite.

Cele 3 standarde pe care le-am folosit pentru nivelurile de izolare sunt:

1. Read Committed

2. Serializable

3. Read Only

18.1 Read Committed

De exemplu, un angajat vrea sa proceseze o comanda asa ca verifica si actualizeaza stocul, dar in acelasi timp un alt angajat vrea sa faca acelasi lucru.

Cu **Read Committed** fiecare SELECT vede doar datele confirmate.

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
UPDATE STOC
SET cantitate = cantitate - 1
WHERE id_produc = 1;
INSERT INTO ACHIZITIE (id_achizitie, id_cumparator, id_angajat, id_produc, cantitate)
VALUES (SEQ_ACHIZITIE.NEXTVAL, 2, 1, 1, 1);
COMMIT;
```

18.2 Serializable

Fie situatia in care mai multi clienti doresc sa cumpere acelasi produs in acelasi timp. Folosind **Serializable**, baza de date va trata fiecare tranzactie unua dupa alta, nu in paralel. Acest isolation level previne consistenta stocului, in cazul in care acesta este pe terminante.

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
UPDATE STOC
SET cantitate = cantitate - 2
```

```

WHERE id_produs = 1;
INSERT INTO ACHIZITIE (
id_achizitie, id_cumparator, id_angajat, id_produs, cantitate)
VALUES (SEQ_ACHIZITIE.NEXTVAL, 3, 2, 1, 2);
COMMIT;

```

18.3 Read only

Fie situatia in care managerul magazinului doreste sa vada care produse au pretul mai mare de 50 de lei pentru a face un raport. In acest caz, se doreste ca acesta doar sa citeasca, nu sa modifice.

```

SET TRANSACTION READ ONLY;
SELECT id_produs, pret
FROM PRODUS
WHERE pret > 50;
COMMIT;

```

19 Migrarea la o baza de date de tip NoSQL

Proiectul Gestiunea unui Magazin de Muzica "Rachi Store" a fost conceput folosind un model relational folosind limbajul SQL si sistemul Oracle. Acesta este un sistem cu relatii riguroase, totusi in anumite scenarii o astfel de abordare poate deveni greu de intretinut.

De exemplu, un produs poate fi un album sau un tricou, poate avea sau nu o marime, poate sa fie in mai multe formate media, iar informatiile nu sunt mereu standard. Asta inseamna ca daca vreau sa adaug ceva nou sau sa modific structura, trebuie sa umblu in toata schema SQL, sa rescriu tabele, sa refac chei etc. E rigid.

In plus, pentru a vedea detalii despre un album (cum ar fi genul, casa de discuri, formatele in care e disponibil), trebuia sa fac 3-4 JOIN-uri. In MongoDB, pot sa pun totul intr-un singur document. E mai simplu, mai flexibil si functioneaza bine pentru proiecte in care datele pot varia mult.

19.1 Prezentarea structurii bazei de date NoSQL

Pentru a nu mai avea join-uri si produse carteziene ca intr-o baza de date relationala am ales sa combin tabelele ce erau separate in 3 colectii dupa cum urmeaza:

Album

- Album
- Gen
- Casa_De_Discuri
- Format_Media
- Melodie
- Artist
- Stoc
- Furnizor

Merchandise

- Merchandise
- Stoc
- Furnizor

Achizitie

- Achizitie
- Cumparator
- Angajator
- Functie
- Produs

19.2 Prezentare comenzi pentru crearea bazei de date

```
client = MongoClient("mongodb://localhost:27017/")
db = client["RecordStore"]
if "Album" not in db.list_collection_names():
    db.create_collection("Album")
if "Merchandise" not in db.list_collection_names():
    db.create_collection("Merchandise")
if "Achizitie" not in db.list_collection_names():
    db.create_collection("Achizitie")
```

The screenshot shows a user interface for managing database connections. At the top, there's a title 'CONNECTIONS (2)' next to a close button (X), a plus sign (+) for adding new connections, and three dots for more options. Below this is a search bar labeled 'Search connections' with a magnifying glass icon. The main area lists two connections:

- A star icon followed by 'Local MongoDB'.
- A star icon followed by 'MagazinDeMuzica', which is expanded to show its internal structure:
 - A folder icon followed by 'RecordStore', which is further expanded to show three subfolders: 'Achizitie', 'Album', and 'Merchandise'.
 - A folder icon followed by 'admin'.
 - A folder icon followed by 'config'.
 - A folder icon followed by 'local'.
 - A folder icon followed by 'magazin'.

19.3 Prezentarea comezilor pentru inserare

Pentru inserarea datelor am ales sa fac un meniu interactiv CRUD in terminal. Aceasta citeste optiuni de la tastatura iar in final le adauga in baza de date.

```
while True:  
    print("\n===== MENIU =====")
```

```

print("1. Introduceti un album")
print("2. Introduceti un produs merchandise")
print("3. Introduceti o achizitie")
print("0. Iesire")
opt = input("Optiune: ")

if opt == "0":
    print("Exit.")
    break

elif opt == "1":
    nume_album = input("Nume album: ")
    pret = float(input("Pret: "))
    data_lansare = input("Data lansare (YYYY-MM-DD): ")
    lungime = input("Durata album (ex: 49:07): ")

    gen_nume = input("Gen muzical: ")
    casa_nume = input("Nume casa discuri: ")
    casa_email = input("Email casa discuri: ")

    formate_disponibile = []
    while True:
        nume_format = input("Format media (CD, Vinyl etc) [ENTER pt stop]: ")
        if nume_format == "":
            break
        procent = int(input("Procent adaugat: "))
        formate_disponibile.append({"nume": nume_format, "procent_adaugat": procent})

    melodii = []
    while True:
        titlu = input("Titlu melodie [ENTER pt stop]: ")
        if titlu == "":
            break
        lungime_melodie = input("Durata melodie: ")
        prenume = input("Prenume artist: ")
        nume_familie = input("Nume familie artist: ")
        trupa = input("Trupa (ENTER daca nu are): ")
        melodii.append({
            "titlu": titlu,
            "lungime": lungime_melodie,
            "artist": {
                "prenume": prenume,
                "nume_familie": nume_familie,
                "trupa": trupa if trupa else None
            }
        })

    album = {
        "nume_album": nume_album,
        "pret": pret,
        "data_lansare": data_lansare,
        "lungime": lungime,
        "gen": {"nume": gen_nume},
        "casa_discuri": {"nume": casa_nume, "email": casa_email},
        "formate_disponibile": formate_disponibile,
        "melodii": melodii
    }

```

```

albums.insert_one(album)
print("Album adaugat.")

elif opt == "2":
    nume = input("Nume produs: ")
    tip = input("Tip produs (Tricou, Poster etc): ")
    marime = input("Marime (ENTER daca nu are): ")
    pret = float(input("Pret: "))
    data_lansare = input("Data lansare (YYYY-MM-DD): ")

    stoc = []
    while True:
        cantitate = input("Cantitate pe stoc [ENTER pt stop]: ")
        if cantitate == "":
            break
        cantitate = int(cantitate)
        furnizor_nume = input("Nume furnizor: ")
        furnizor_telefon = input("Telefon furnizor: ")
        furnizor_adresa = input("Adresa furnizor: ")
        stoc.append({
            "cantitate": cantitate,
            "furnizor": {
                "nume": furnizor_nume,
                "telefon": furnizor_telefon,
                "adresa": furnizor_adresa
            }
        })
    )

    produs = {
        "nume": nume,
        "tip": tip,
        "marime": marime if marime else None,
        "pret": pret,
        "data_lansare": data_lansare,
        "stoc": stoc
    }

    merch.insert_one(produs)
    print("Merchandise adaugat.")

elif opt == "3":
    data = input("Data achizitie (YYYY-MM-DD): ")
    metoda_plata = input("Metoda plata (Cash, Card etc): ")
    status = input("Status (Pending, Completed): ")
    cantitate = int(input("Cantitate: "))

    cumparator_nume = input("Nume cumparator: ")
    cumparator_prenume = input("Prenume cumparator: ")
    adresa = input("Adresa cumparator: ")
    oras = input("Oras: ")

    angajat_nume = input("Nume angajat: ")
    angajat_prenume = input("Prenume angajat: ")
    functie_nume = input("Functie: ")
    salariu_min = float(input("Salariu minim functie: "))
    salariu_max = float(input("Salariu maxim functie: "))
    telefon = input("Telefon angajat: ")
    data_angajare = input("Data angajare (YYYY-MM-DD): ")

```

```

salariu = float(input("Salariu actual: "))

tip_produs = input("Tip produs (Album sau Merchandise): ")
nume_produs = input("Nume produs: ")
pret_produs = float(input("Pret produs: "))

achizitie = {
    "data": data,
    "metoda_plata": metoda_plata,
    "status": status,
    "cantitate": cantitate,
    "cumparator": {
        "nume": cumparator_nume,
        "prenume": cumparator_prenume,
        "adresa": adresa,
        "oras": oras
    },
    "angajat": {
        "nume": angajat_nume,
        "prenume": angajat_prenume,
        "functie": {
            "nume": functie_nume,
            "salariu_min": salariu_min,
            "salariu_max": salariu_max
        },
        "telefon": telefon,
        "data_angajare": data_angajare,
        "salariu": salariu
    },
    "produs": {
        "tip": tip_produs,
        "nume": nume_produs,
        "pret": pret_produs
    }
}

achizitii.insert_one(achizitie)
print("Achizitie adaugata.")

else:
    print("Optiune invalida.")

```

Introducere Albums

Type a query: { field: 'value' } or [Generate query](#)

[ADD DATA](#) [EXPORT DATA](#) [UPDATE](#) [DELETE](#)

```

_id: ObjectId('6849ed6ae410306dec877ec3')
nume_album : "GNX"
pret : 20
data_lansare : "2024-06-10"
lungime : "45:35"
gen : Object
  nume : "Rap"
casa_discuri : Object
  nume : "Republic Records"
  email : "\rep@gmail.com"
formate_disponibile : Array (1)
  0: Object
    nume : "CD"
    procent_adaugat : 20
melodii : Array (1)
  0: Object
    titlu : "dasda"
    lungime : "23"
    artist : Object
      prenume : "vxcds"
      nume_familie : "lamar"
      trupa : null
  
```

Introducere Merchandise

MagazinDeMuzica > RecordStore > Merchandise

[Open MongoDB shell](#)

Documents 1 Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or [Generate query](#)

[ADD DATA](#) [EXPORT DATA](#) [UPDATE](#) [DELETE](#)

```

_id: ObjectId('6849ecb4e410306dec877ec2')
nume : "Tricou"
tip : "Tricou"
marime : "M"
pret : 34
data_lansare : "2005-06-12"
stoc : Array (1)
  0: Object
  
```