

# Centralizare date universitate

Roșca Paul-Teodor

## Cuprins

<b>1</b>	<b>Introducere</b>	<b>2</b>
<b>2</b>	<b>Tabele</b>	<b>2</b>
2.1	Studenti . . . . .	2
2.1.1	Constrângeri . . . . .	3
2.1.2	Triggere . . . . .	3
2.2	Discipline . . . . .	4
2.2.1	Constrângeri . . . . .	4
2.2.2	Triggere . . . . .	5
2.3	Discipline Studenti . . . . .	6
2.4	Prezențe . . . . .	7
2.5	Catalog . . . . .	7
2.5.1	Constrângeri . . . . .	7
2.5.2	Triggere . . . . .	8
<b>3</b>	<b>Vederi Materializate</b>	<b>9</b>
3.1	Număr Teste Student . . . . .	9
<b>4</b>	<b>Vederi</b>	<b>9</b>
4.1	Prezențe . . . . .	9
4.2	Număr prezențe . . . . .	9
4.3	Numărul testelor la o disciplină . . . . .	10
4.4	Note . . . . .	10
4.5	Note centralizate . . . . .	10
4.6	Medii discipline . . . . .	11
4.7	Discipline studenti . . . . .	11
<b>5</b>	<b>ERD</b>	<b>12</b>
<b>6</b>	<b>GitHub</b>	<b>12</b>

## 1 Introducere

Pentru acest proiect am ales să folosesc baza de date de la Oracle și anume versiunea **Oracle Database Express Edition (XE) Release 18.4.0.0.0 (18c)**.

Partea de programare am realizat-o în Java, folosind driver-ul **Oracle Database 18c (18.3) JDBC Driver** pentru conectarea la baza de date și API-ul **Swing** pentru interfața grafică.

Validarea datelor din tabele s-a realizat înainte de inserare atât la nivelul aplicației cât și la nivelul bazei de date prin intermediul *constraint-urilor* și al *trigger-urilor*.

Deoarece Oracle nu suportă tipul de date **boolean** am folosit pe post de înlocuitor tipul de date **number(1,0)** împreună cu un **check constraint**, pentru a permite doar valori de 1 și 0.

## 2 Tabele


Toate tabelele create sunt în **BCNF (3.5NF)**. Astfel acestea respectă toate dintre următoarele condiții :

- Coloanele conțin doar valori atomice
- Coloanele conțin doar același tip de date
- Coloanele dintr-un tabel au nume diferite
- Nu conțin dependențe parțiale
- Nu conțin dependențe tranzitive
- Pentru fiecare dependență  $X \rightarrow Y$ ,  $X$  este supercheie

Pentru fiecare tabelă care are o cheie primară simplă, am folosit o secvență pentru generarea acelei chei.

### 2.1 Studenți

Tabela reține informații de bază despre fiecare student înscris la universitatea respectivă. Precum CNP-ul, numele, prenumele, informațiile de contact, soldul anului universitar curent și faptul că studentul este angajat (1) sau nu (0).

PK	Name	Data Type	Size	Not Null	Default
	NR_MATRIC...	NUMBER		<input checked="" type="checkbox"/>	
	CNP	CHAR	13	<input checked="" type="checkbox"/>	
	NUME	VARCHAR2	20	<input checked="" type="checkbox"/>	
	PRENUME	VARCHAR2	20	<input checked="" type="checkbox"/>	
	NRTEL	CHAR	10	<input checked="" type="checkbox"/>	
	EMAIL	VARCHAR2	40	<input checked="" type="checkbox"/>	
	ADRESA	VARCHAR2	100	<input type="checkbox"/>	
	SOLD	NUMBER	10	<input checked="" type="checkbox"/>	0
	ANGAJAT	NUMBER	1	<input checked="" type="checkbox"/>	0

### 2.1.1 Constrângeri

CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION
1 ANGAJAT_CK	Check	ANGAJAT IN (1,0)
2 CNP_UK	Unique	(null)
3 EMAIL_CK	Check	REGEXP_LIKE (EMAIL, '^([a-zA-Z0-9...)
4 NRTEL_CK	Check	REGEXP_LIKE (NRTEL, '^07[2-7][0-...)
5 NUME_CK	Check	REGEXP_LIKE (NUME, '^([a-zăîăş...)
6 PERSOANE_PK	Primary_Key	(null)
7 PRENUME_CK	Check	REGEXP_LIKE (PRENUME, '^([a-zăî...)
8 SOLD_CK	Check	SOLD >= 0
9 SYS_C007299	Check	"CNP" IS NOT NULL
10 SYS_C007300	Check	"NUME" IS NOT NULL
11 SYS_C007301	Check	"PRENUME" IS NOT NULL
12 SYS_C007312	Check	"NRTEL" IS NOT NULL
13 SYS_C007313	Check	"EMAIL" IS NOT NULL
14 SYS_C007724	Check	"ANGAJAT" IS NOT NULL
15 SYS_C007726	Check	"SOLD" IS NOT NULL
16 SYS_C008432	Check	"NR_MATRICOL" IS NOT NULL

Pentru validarea informațiilor *nume*, *prenume*, *email* și *numar de telefon* am folosit următoarele expresii regulate :

```
01 | REGEXP_LIKE (EMAIL, '^([a-zA-Z0-9]+(-|\.|_|_)*[a-zA-Z0-9]+)+  
02 | @([a-zA-Z0-9]+(-|\.|_|_)*[a-zA-Z0-9]+)+\.[a-z]{2,4}$');  
03 | REGEXP_LIKE (NRTEL, '^07[2-7][0-9]{7}$');  
04 | REGEXP_LIKE (NUME, '^([a-zăîăşȚA-ZĂÎĂȘȚ]{3,}-{0,1})+$');  
05 | REGEXP_LIKE (PRENUME, '^([a-zăîăşȚA-ZĂÎĂȘȚ]{3,}-[:blank:]{0,1})+$');
```

Spațiile goale reprezintă diacriticele limbii române.

### 2.1.2 Triggere

```
01 | TRIGGER PERSOANE_CNP_TRIGGER  
02 | BEFORE INSERT OR UPDATE OF CNP ON "STUDENTI"  
03 | REFERENCING OLD AS OLD NEW AS NEW  
04 | FOR EACH ROW  
05 | DECLARE  
06 |     s NUMBER(4,0) := 0;  
07 |     c CHAR(12) := '279146358279';  
08 | BEGIN  
09 |     :NEW.CNP := TRIM(:NEW.CNP);  
10 |     IF LENGTH(:NEW.CNP) != 13 THEN  
11 |         RAISE_APPLICATION_ERROR(-20666, 'CNP Invalid');  
12 |     END IF;  
13 |     FOR i IN 1..12 LOOP  
14 |         s := s + TO_NUMBER(SUBSTR(:NEW.CNP, i, 1)) * TO_NUMBER(SUBSTR(c, i, 1));  
15 |     END LOOP;  
16 |     s := MOD(s, 11);  
17 |     IF s = 10 THEN  
18 |         s := 1;  
19 |     END IF;  
20 |     IF s != TO_NUMBER(SUBSTR(:NEW.CNP, 13, 1)) THEN  
21 |         RAISE_APPLICATION_ERROR(-20666, 'CNP Invalid');  
22 |     END IF;  
23 | END;
```

Deoarece validarea CNP-ului este un proces complex, am folosit un trigger pentru a ne asigura că acest CNP este valid înainte de a-l insera.

```


01 | TRIGGER TRIM_TRIGGER
02 | BEFORE INSERT OR UPDATE OF PRENOME, NUME, NRTEL, EMAIL, ADRESA ON STUDENTI
03 | REFERENCING OLD AS OLD NEW AS NEW
04 | FOR EACH ROW
05 | BEGIN
06 |     :NEW.NUME := TRIM(UPPER(:NEW.NUME));
07 |     :NEW.PRENOME := TRIM(UPPER(:NEW.PRENOME));
08 |     :NEW.NRTEL := TRIM(:NEW.NRTEL);
09 |     :NEW.EMAIL := TRIM(:NEW.EMAIL);
10 |     :NEW.ADRESA := TRIM(:NEW.ADRESA);
11 | END;
12 |

```

De asemenea deoarece String-urile pot fi introduse cu spații înainte sau la final, trebuie să ne asigurăm că le îndepărtăm, folosind funcția *TRIM*, iar pentru consistență vom reține mereu numele cu litere mari.

## 2.2 Discipline

Această tabelă reține informațiile de bază ale unei discipline, precum numele acesteia, numărul minim de prezențe pe care un student trebuie să îl aibă la acea disciplină (care diferă dacă studentul este angajat sau nu), numărul de teste la disciplina respectivă, precum și modul de calculare al mediei, care este reprezentat prin procentul pe care îl reprezintă testele, respectiv examenul din media finală, la care se adaugă punctele bonus, dacă disciplina acceptă puncte bonus (1) sau nu, în caz contrar (0).

PK	Name	Data Type	Size	Not Null	Default
	ID	NUMBER		<input checked="" type="checkbox"/>	
	NUME_DISCIPLINA	VARCHAR2	20	<input checked="" type="checkbox"/>	
	PREZENTE_MINIME_NANG	NUMBER		<input checked="" type="checkbox"/>	10
	PREZENTE_MINIME_ANG	NUMBER		<input checked="" type="checkbox"/>	7
	NR_TESTE	NUMBER		<input checked="" type="checkbox"/>	0
	PROCENT_TESTE	NUMBER		<input checked="" type="checkbox"/>	
	PROCENT_EXAMEN	NUMBER		<input checked="" type="checkbox"/>	
	BONUS	NUMBER	1	<input checked="" type="checkbox"/>	0

### 2.2.1 Constrângeri

Pentru acest tabel constrângerile sunt mai simple. Prin acestea ne asigurăm că : procentele sunt corecte, numele este unul valid pentru o disciplină (sau un acronim de disciplină) și numărul de prezențe minime este corect.

```

01 | REGEXP_LIKE(NUME_DISCIPLINA, '^[a-zA-Z]+[a-zA-Z0-9]*$');
02 | "PREZENTE_MINIME_NANG" >= 0 AND "PREZENTE_MINIME_NANG" <= 14;
03 | PROCENT_TESTE + PROCENT_EXAMEN = 100;

```

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION
1	BONUS_CHK	Check	BONUS IN (0,1)
2	DISCIPLINE_NUME_DISCIPLINA_UK	Unique	(null)
3	DISCIPLINE_PK	Primary_Key	(null)
4	NR_TESTE_CHK	Check	NR_TESTE >= 0
5	NUME_DISCIPLINA_CHK	Check	REGEXP_LIKE(NUME_DISCIPLINA, '^...)
6	PREZENTE_MINIME_CHK	Check	"PREZENTE_MINIME_NANG">=0 AND "...
7	PROCENT_CHK	Check	PROCENT_TESTE + PROCENT_EXAMEN ...
8	PROCENT_EXAMEN_CHK	Check	PROCENT_EXAMEN BETWEEN 0 AND 100
9	PROCENT_TESTE_CHK	Check	PROCENT_TESTE BETWEEN 0 AND 100
10	SYS_C007479	Check	"NUME_DISCIPLINA" IS NOT NULL
11	SYS_C007480	Check	"PREZENTE_MINIME_NANG" IS NOT NULL
12	SYS_C007481	Check	"NR_TESTE" IS NOT NULL
13	SYS_C007482	Check	"PROCENT_TESTE" IS NOT NULL
14	SYS_C007483	Check	"PROCENT_EXAMEN" IS NOT NULL
15	SYS_C007484	Check	"BONUS" IS NOT NULL
16	SYS_C007877	Check	"PREZENTE_MINIME_ANG" IS NOT NULL
17	SYS_C008443	Check	"ID" IS NOT NULL

## 2.2.2 Triggere

```

01 | TRIGGER DISCIPLINA_STERGERE_BONUS_TRIGGER
02 | AFTER UPDATE OF BONUS ON DISCIPLINE
03 | REFERENCING OLD AS OLD NEW AS NEW
04 | FOR EACH ROW
05 | BEGIN
06 |     IF (:NEW.BONUS = 0) THEN
07 |         DELETE FROM CATALOG_UNIVERSITATE
08 |         WHERE ID_D = :NEW.ID AND TIP = 'B';
09 |     END IF;
10 | END;

```

Acest trigger se asigură că dacă o disciplină nu mai suportă puncte bonus, punctele bonus deja acordate la acea disciplină să fie șterse din catalog.

```

01 | TRIGGER DISCIPLINE_PREZENTE_TRIGGER
02 | BEFORE INSERT OR UPDATE OF PREZENTE_MINIME_ANG, PREZENTE_MINIME_NANG
03 | ON DISCIPLINE
04 | REFERENCING OLD AS OLD NEW AS NEW
05 | FOR EACH ROW
06 | BEGIN
07 |     :NEW.PREZENTE_MINIME_ANG := ROUND(:NEW.PREZENTE_MINIME_NANG * 7 / 10);
08 | END;

```

Prezențele minime pentru studenții angajați le calculăm automat, la toate disciplinele acestea reprezentând 70% din numărul de prezențe pe care trebuie să le aibă un student neangajat.

```

01 | TRIGGER NUME_DISCIPLINA_TRIGGER
02 | BEFORE INSERT OR UPDATE OF NUME_DISCIPLINA ON DISCIPLINE
03 | REFERENCING OLD AS OLD NEW AS NEW
04 | FOR EACH ROW
05 | BEGIN
06 |     :NEW.NUME_DISCIPLINA := TRIM(UPPER(:NEW.NUME_DISCIPLINA));
07 | END;

```

Pentru consistență ne asigurăm că numele disciplinei este mereu cu litere mari și că nu avem spații goale înaintea sau la finalul acestuia.

```

01 | TRIGGER DISCIPLINA_STERGERE_TESTE_TRIGGER
02 | BEFORE UPDATE OF NR_TESTE ON DISCIPLINE
03 | REFERENCING OLD AS OLD NEW AS NEW
04 | FOR EACH ROW
05 | DECLARE
06 | type vector is table of number;
07 | nrmSD vector;
08 | nrTesteStudent number;
09 | difNrT number;
10 | BEGIN
11 |     IF (:NEW.NR_TESTE = 0) THEN
12 |         :NEW.PROCENT_TESTE := 0;
13 |         :NEW.PROCENT_EXAMEN := 100;
14 |     END IF;
15 |     IF (:NEW.NR_TESTE < :OLD.NR_TESTE) THEN
16 |         SELECT NR_MATRICOL
17 |         BULK COLLECT
18 |         INTO nrmSD
19 |         FROM DISCIPLINE_STUDENTI
20 |         WHERE ID_D = :NEW.ID;
21 |
22 |         FOR i IN 1..nrmSD.count LOOP
23 |             SELECT NR_TESTE
24 |             INTO nrTesteStudent
25 |             FROM NUMAR_TESTE_STUDENT_DISCIPLINA_VIEW
26 |             WHERE ID_D = :NEW.ID AND NR_MATRICOL = nrmSD(i);
27 |             difNrT := nrTesteStudent - :NEW.NR_TESTE;
28 |             IF (difNrT > 0) THEN
29 |                 DELETE FROM CATALOG_UNIVERSITATE
30 |                 WHERE ID_NOTA IN
31 |                 (
32 |                     SELECT ID_NOTA
33 |                     FROM
34 |                     (
35 |                         SELECT *
36 |                         FROM CATALOG_UNIVERSITATE
37 |                         WHERE NR_MATRICOL = nrmSD(i) AND ID_D = :NEW.ID
38 |                         AND TIP = 'T'
39 |                         ORDER BY DATA DESC
40 |                     )
41 |                     WHERE rownum <= difNrT
42 |                 );
43 |             END IF;
44 |         END LOOP;
45 |     END IF;
46 | END;

```

Acest trigger este unul mai complex și are 2 roluri.



În primul rând se asigură că dacă modificăm numărul de teste al disciplinei la 0, procentul acestora din medie devine 0 și cel al examenului devine 100.

În al doilea rând prin modificarea numărului de teste putem ajunge la o problemă dacă numărul nou de teste este mai mic decât cel inițial. În acel caz putem avea studenți care au deja prea multe teste, iar pentru a rezolva problema ștergem ultimele note primite de acei stundeți la teste, pentru ca numărul lor total de teste să coincidă cu cel al disciplinei.

## 2.3 Discipline Studenți


Această tabelă este una de joncțiune și are rolul de a mapa relația *many-to-many* dintre **Studenți** și **Discipline**. Astfel cheia ei primară este alcătuită din cheile primare ale celorlalte doua tabele, care

constituie chei străine.

PK	Name	Data Type	Size	Not Null	Default
	NR_MATRIC...	NUMBER		<input checked="" type="checkbox"/>	
	ID_D	NUMBER		<input checked="" type="checkbox"/>	


## 2.4 Prezențe

Tabela de prezențe reține pe fiecare rând câte o prezență a unui student la o disciplină. Constrângerile sunt doar cele de nullificare și cele de cheie străină pentru id-ul disciplinei și numărul matricol al studentului.


PK	Name	Data Type	Size	Not Null	Default
	ID_PREZ	NUMBER		<input checked="" type="checkbox"/>	
	DATA	DATE		<input checked="" type="checkbox"/>	
	NR_MATRIC...	NUMBER		<input checked="" type="checkbox"/>	
	ID_D	NUMBER		<input checked="" type="checkbox"/>	

## 2.5 Catalog

Pentru reținerea tuturor notelor studenților folosim această tabelă, în care se găsesc atât notele de la teste și examene, cât și punctele bonus acordate elevilor la o disciplină. Aceste tipuri diferite de note sunt distinse folosind coloana **TIP** care poate conține doar caracterele *T*, *E* sau *B*.

PK	Name	Data Type	Size	Not Null
	ID_NOTA	NUMBER		<input checked="" type="checkbox"/>
	DATA	DATE		<input checked="" type="checkbox"/>
	NR_MATRIC...	NUMBER		<input checked="" type="checkbox"/>
	ID_D	NUMBER		<input checked="" type="checkbox"/>
	PUNCTAJ	NUMBER	4	<input checked="" type="checkbox"/>
	TIP	CHAR	1	<input checked="" type="checkbox"/>

### 2.5.1 Constrângeri

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION
1	CATALOG_UNIVERSITATE_FK1	Foreign_Key	(null)
2	CATALOG_UNIVERSITATE_PK	Primary_Key	(null)
3	CATALOG_UNIVERSITATE_PUNCTAJ...	Check	PUNCTAJ BETWEEN 0 AND 10
4	CATALOG_UNIVERSITATE_UK	Unique	(null)
5	SYS_C008166	Check	"DATA" IS NOT NULL
6	SYS_C008167	Check	"PUNCTAJ" IS NOT NULL
7	SYS_C008168	Check	"TIP" IS NOT NULL
8	SYS_C008171	Check	"NR_MATRICOL" IS NOT NULL
9	SYS_C008172	Check	"ID_D" IS NOT NULL
10	SYS_C008495	Check	"ID_NOTA" IS NOT NULL
11	TP_PUNCTAJ_CHK	Check	TIP IN ('E','T','B')

Constrângerile acestei tabelă sunt simple și la obiect

### 2.5.2 Triggere

```
01 | TRIGGER CATALOG_VALIDARE_NOTA_TRIGGER
02 | BEFORE INSERT OR UPDATE OF DATA, ID_D, NR_MATRICOL, PUNCTAJ, TIP ON
    CATALOG_UNIVERSITATE
03 | REFERENCING OLD AS OLD NEW AS NEW
04 | FOR EACH ROW
05 | DECLARE
06 | bonusDisciplina number(1,0);
07 | examene number;
08 | nrTesteActual number;
09 | nrTesteMax number;
10 | BEGIN
11 |     IF (:NEW.DATA > SYSDATE) THEN
12 |         RAISE_APPLICATION_ERROR(-20666, 'Data notei nu poate fi in viitor');
13 |     END IF;
14 |     :NEW.TIP := TRIM (:NEW.TIP);
15 |     IF (:NEW.TIP = 'B') THEN
16 |         SELECT BONUS
17 |         INTO bonusDisciplina
18 |         FROM DISCIPLINE
19 |         WHERE ID = :NEW.ID_D;
20 |         IF (bonusDisciplina = 0) THEN
21 |             RAISE_APPLICATION_ERROR(-20666,
22 |                                     'Disciplina nu accepta puncte bonus');
23 |         END IF;
24 |     ELSIF (:NEW.TIP = 'E') THEN
25 |         SELECT COUNT(*)
26 |         INTO examene
27 |         FROM CATALOG_UNIVERSITATE
28 |         WHERE NR_MATRICOL = :NEW.NR_MATRICOL AND ID_D = :NEW.ID_D AND TIP = 'E';
29 |         IF (examene > 0) THEN
30 |             RAISE_APPLICATION_ERROR(-20666,
31 |                                     'Studentul are deja o nota la acest examen!');
32 |         END IF;
33 |     ELSIF (:NEW.TIP = 'T') THEN
34 |
35 |         SELECT NR_TESTE
36 |         INTO nrTesteMax
37 |         FROM DISCIPLINE
38 |         WHERE ID = :NEW.ID_D;
39 |
40 |         SELECT NR_TESTE
41 |         INTO nrTesteActual
42 |         FROM NUMAR_TESTE_STUDENT_DISCIPLINA_VIEW
43 |         WHERE NR_MATRICOL = :NEW.NR_MATRICOL AND ID_D = :NEW.ID_D;
44 |         IF (nrTesteActual = nrTesteMax) THEN
45 |             RAISE_APPLICATION_ERROR(-20666,
46 |                                     'Studentul deja are inregistrare toate notele la materia respectiva');
47 |         END IF;
48 |
49 |     END IF;
50 | END;
```

Acesta este principalul trigger al tabelii, care se asigură că nota introdusă în tabelă este validă, oricare ar fi tipul acesteia și că data notei nu depășește data sistemului.

Astfel, dacă nota introdusă este :

- **un bonus**, trebuie să verificăm dacă disciplina acceptă puncte bonus.
- **un test**, trebuie să verificăm dacă studentul nu a atins deja numărul de teste la disciplina respectivă.
- **un examen**, trebuie să verificăm dacă studentul nu are deja o notă la examenul acelei discipline.



## 3 Vederi Materializate

### 3.1 Număr Teste Student

#### *NUMAR\_TESTE\_STUDENT\_MVIEW*

```
01 | SELECT NR_MATRICOL, ID_D, COUNT(*) NUMAR_TESTE_DISCIPLINA
02 | FROM CATALOG_UNIVERSITATE
03 | WHERE TIP = 'T'
04 | GROUP BY NR_MATRICOL, ID_D
```

Această vedere calculează și reține pentru fiecare student numărul de teste pe care acesta le-a dat la o disciplină. Avem nevoie ca această vedere să fie materializată, pentru că este folosită indirect în triggerul de la 2.5.2 .

## 4 Vederi

Deoarece unele informații sunt accesate mai des în aplicație și necesită un query complex, am creat multiple vederi care să ne ofere aceste informații în formatul dorit.

### 4.1 Prezențe

#### *PREZENTE\_VIEW*

```
01 |
02 | SELECT P.ID_PREZ, P.DATA, P.NR_MATRICOL, S.CNP, P.ID_D, D.NUME_DISCIPLINA
03 | FROM PREZENTE P, STUDENTII S, DISCIPLINE D
04 | WHERE S.NR_MATRICOL = P.NR_MATRICOL AND P.ID_D = D.ID
05 | ORDER BY P.DATA DESC
```

Pentru utilizatorul aplicației cheile primare nu sunt neapărat foarte utile, așa că pentru prezențe am adăugat vederii și informații precum CNP-ul studentului și numele disciplinei, iar pentru a oferi datelor o ordine, le sortăm descrescător după data prezenței.

### 4.2 Număr prezențe

#### *NUMAR\_PREZENTE\_VIEW*

```
01 | SELECT NR_MATRICOL, ID_D, COUNT(*) PREZENTE_STUDENT
02 | FROM PREZENTE
03 | GROUP BY NR_MATRICOL, ID_D
```

Această vedere calculează numărul de prezențe al studenților din tabelul 2.4 nefiind utilă în mod direct utilizatorului, ci mai degrabă fiind folosită în modelarea altor vederi.

### 4.3 Numărul testelor la o disciplină

#### *NUMAR\_TESTE\_STUDENT\_DISCIPLINA\_VIEW*

```
01 | SELECT DS.NR_MATRICOL , DS.ID_D , NVL(NTSMV.NUMAR_TESTE_DISCIPLINA ,0)
02 | FROM DISCIPLINE_STUDENTI DS
03 | LEFT JOIN NUMAR_TESTE_STUDENT_MVIEW NTSMV
04 | ON DS.NR_MATRICOL = NTSMV.NR_MATRICOL AND DS.ID_D = NTSMV.ID_D
```

Vederea are rol de tabelă auxiliară pentru validarea notelor din triggerul 2.5.2 și se asigură că fiecare pereche student-disciplină are un număr de prezențe atribuit (0 în cazul în care nu apar prezențe pentru perechea respectivă în tabela de prezențe)

### 4.4 Note

#### *NOTE\_VIEW*

```
01 | SELECT CU.ID_NOTA , CU.NR_MATRICOL , S.CNP , CU.DATA ,
02 |       D.NUME_DISCIPLINA , CU.PUNCTAJ , CU.TIP
03 | FROM CATALOG_UNIVERSITATE CU , DISCIPLINE D , STUDENTI S
04 | WHERE CU.ID_D = D.ID AND CU.NR_MATRICOL = S.NR_MATRICOL
05 | ORDER BY CU.DATA DESC
```

Pentru utilizatorul aplicației cheile primare nu sunt neapărat foarte utile, așa că pentru note am adăugat vederii și informații precum CNP-ul studentului și numele disciplinei, iar pentru a oferi datelor o ordine, le sortăm descrescător după data prezenței.

### 4.5 Note centralizate

#### *NOTE\_CENTRALIZATE\_VIEW*

```
01 | SELECT DS.NR_MATRICOL , DS.ID_D , ROUND(NVL(TESTE.MEDIE_TESTE ,0))
02 |       , ROUND(NVL(EXAMEN.NOTA_EXAMEN ,0)) , NVL(BONUS.PUNCTE_BONUS ,0)
03 | FROM DISCIPLINE_STUDENTI DS
04 | LEFT JOIN
05 | (
06 |     SELECT ST.NR_MATRICOL , ST.ID_D , ST.SUMA_TESTE/D.NR_TESTE MEDIE_TESTE
07 |     FROM
08 |     (
09 |         SELECT CU.NR_MATRICOL , CU.ID_D , SUM(ROUND(CU.PUNCTAJ)) SUMA_TESTE
10 |         FROM CATALOG_UNIVERSITATE CU
11 |         WHERE CU.TIP = 'T'
12 |         GROUP BY CU.NR_MATRICOL , CU.ID_D
13 |     ) ST
14 |     , DISCIPLINE D
15 |     WHERE D.ID = ST.ID_D
16 | ) TESTE
17 | ON DS.NR_MATRICOL = TESTE.NR_MATRICOL AND DS.ID_D = TESTE.ID_D
18 | LEFT JOIN
19 | (
20 |     SELECT NR_MATRICOL , ID_D , PUNCTAJ , NOTA_EXAMEN
21 |     FROM CATALOG_UNIVERSITATE
22 |     WHERE TIP = 'E'
23 | ) EXAMEN
```

```

24 |      ON DS.NR_MATRICOL = EXAMEN.NR_MATRICOL AND DS.ID_D = EXAMEN.ID_D
25 |      LEFT JOIN
26 |      (
27 |          SELECT NR_MATRICOL, ID_D, SUM(PUNCTAJ) PUNCTE_BONUS
28 |          FROM CATALOG_UNIVERSITATE
29 |          WHERE TIP = 'B'
30 |          GROUP BY NR_MATRICOL, ID_D
31 |      ) BONUS
32 |      ON DS.NR_MATRICOL = BONUS.NR_MATRICOL AND DS.ID_D = BONUS.ID_D

```

Rolul acestei vederi este de a pune la un loc și a prelucra toate notele studenților la disciplinele la care sunt înscriși. Astfel calculăm media testelor, extragem nota de examen și însumăm punctele bonus pentru fiecare înscriere student-disciplină.

## 4.6 Medii discipline

### *MEDII\_STUDENT\_VIEW*

```

01 | SELECT NCV.NR_MATRICOL, NCV.ID_D,
02 | CASE
03 | WHEN ROUND(NCV.MEDIE_TESTE*(D.PROCENT_TESTE/100)
04 |      +NCV.NOTA_EXAMEN*(D.PROCENT_EXAMEN/100)+NCV.PUNCTE_BONUS) < 10
05 | THEN ROUND(NCV.MEDIE_TESTE*(D.PROCENT_TESTE/100)
06 |      +NCV.NOTA_EXAMEN*(D.PROCENT_EXAMEN/100)+NCV.PUNCTE_BONUS)
07 | ELSE 10
08 | END AS MEDIE_DISCIPLINA
09 | FROM NOTE_CENTRALIZATE_VIEW NCV
10 | JOIN DISCIPLINE D
11 | ON D.ID = NCV.ID_D

```

Această vedere calculează mediile finale ale studenților la fiecare disciplină la care sunt înscriși după formula :  $MediaTestelor * ProcentTeste + NotaExamen * ProcentExamen + PuncteBonus$

## 4.7 Discipline studenți

### *DISCIPLINE\_STUDENTI\_VIEW*

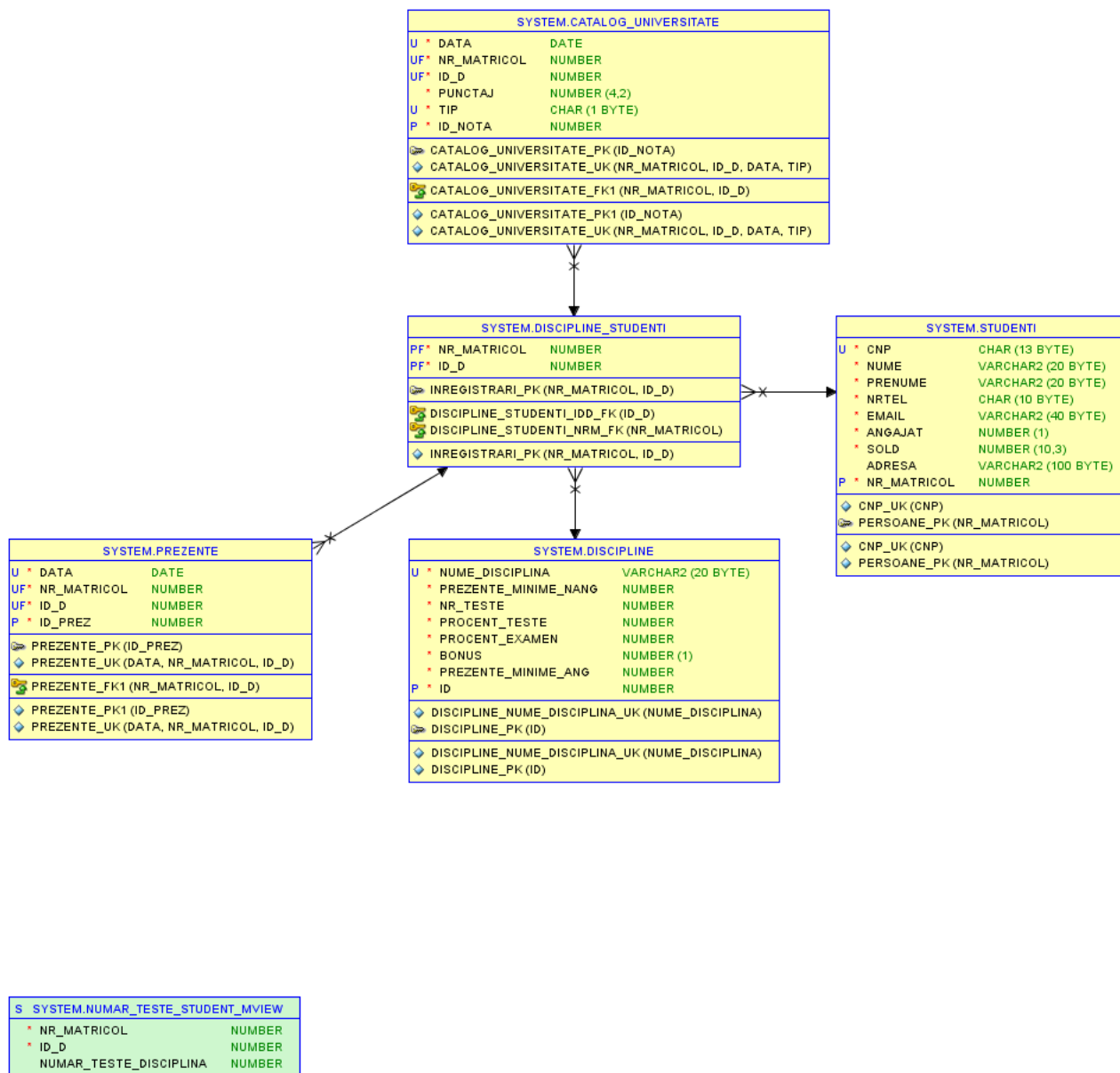
```

01 | SELECT S.NR_MATRICOL, S.CNP, S.NUME, S.PRENUME, D.ID ID_D, D.NUME_DISCIPLINA,
02 | CASE WHEN S.ANGAJAT = 1 THEN D.PREZENTE_MINIME_ANG
03 | ELSE D.PREZENTE_MINIME_NANG
04 | END AS PREZENTE_MINIME, NVL(NPV.PREZENTE_STUDENT,0), MSV.MEDIA
05 | FROM DISCIPLINE_STUDENTI DS
06 | LEFT JOIN STUDENTI S
07 |      ON S.NR_MATRICOL = DS.NR_MATRICOL
08 | LEFT JOIN DISCIPLINE D
09 |      ON D.ID = DS.ID_D
10 | LEFT JOIN NUMAR_PREZENTE_VIEW NPV
11 |      ON NPV.NR_MATRICOL = DS.NR_MATRICOL AND NPV.ID_D = DS.ID_D
12 | LEFT JOIN MEDII_STUDENT_VIEW MSV
13 |      ON MSV.NR_MATRICOL = DS.NR_MATRICOL AND MSV.ID_D = DS.ID_D

```

În această vedere punem la un loc toate informațiile relevante utilizatorului despre situațiile disciplinelor unui student, astfel includem numărul de prezențe necesar la disciplinele respective, numărul de prezențe al studentului și media studentului la acele discipline.

## 5 ERD



## 6 GitHub

Aplicația de gestiune și scriptul pentru crearea tabelelor în Oracle se găsesc aici.