Palatul Copiilor Cilibia Mălina-Ioana

Specializarea Informatică Economică

Anul 2, Grupa 2

Universitatea Babeș-Bolyai, Cluj-Napoca

FACULTATEA DE ȘTIINȚE ECONOMICE ȘI GESTIUNEA AFACERILOR

Cuprins

1. **BAZA DE DATE PL/SQL............................................................................ 2**
2. Introducere.............................................................................................2
3. Baza de date existentă.............................................................................3
4. Modificările bazei de date.........................................................................8
5. Packages................................................................................................11
6. Triggers....................................................................................................25
7. **BAZA DE DATE MONGODB........................................................................27**
8. Scenariu...................................................................................................27
9. Schema bazei de date...............................................................................28
10. Crearea bazei de date...............................................................................29
11. Interogări..................................................................................................45

1. introducere

***BAZA DE DATE PL/SQL***

Pentru o gestionare mai eficientă a activitățiilor *Palatului Copiilor din Cluj-Napoca*, am dezvoltat o bază de date complexă care să integreze toate aspectele legate de organizarea și desfășurarea activitățiilor din cadrul instituției.

În cadrul Palatului, oferim tuturor copiilor cu vârste între 3 și 18 ani o gamă diversificată de cursuri și activități cu care își pot ocupa timpul liber, precum muzică, desen, sport și alte domenii creative. Fiecare copil este încadrat într-o grupă de vârstă specifică, iar costurile participării sunt calculate pe baza pachetelor de ședințe alese, existând opțiuni pentru 14 sau 28 de ședințe.

Pe lângă baza de date deja existentă, am vrut să implementăm încă câteva funcționalități absolut necesare pentru gestionarea participanțiilor și a programărilor. Astfel, în cele ce urmează, am implementat următoarele: un sistem care verifică *suprapunerile de programări* pentru a asigura o organizare eficientă, un sistem de *urmărire a statusului plăților și tarifelor*, un sistem pentru *păstrarea istoricului plăților* și un sistem pentru colectarea și *analiza feedback-ului* venit din partea participanților pentru a îmbunătăți constant calitatea serviciilor. Pe lângă aceste funcționalități, am adăugat și alte îmbunătățiri și mecanisme menite să optimizeze gestionarea activităților și să asigure o experiență completă, detaliile acestora fiind prezentate în secțiunile următoare.

Pentru asigurarea integrității datelor, cum ar fi verificarea suprapunerilor de programări și validarea sumelor plăților, au fost implementați triggeri care automatizează aceste verificări. Gestionarea operațiunilor a fost simplificată prin utilizarea procedurilor și funcțiilor stocate, iar baza de date, complet normalizată, include tabele pentru *participanți, grupe de vârstă, activități, programări, plăți, tarife și feedback*. Proiectul a fost dezvoltat pe platforma *Oracle APEX,* incluzând scripturi pentru adăugări și actualizări, oferind o soluție flexibilă și ușor de extins.

1. baza de date existentă

Pentru impelementarea noilor funcționalități, vom utiliza și extinde baza de date deja existentă din proiectul anterior. Acesta include momentan tabelele *participanți, grupe de vârstă, programări, activități, plăți* și *tarife*. Aceste tabele oferă o fundație solidă pentru dezvoltarea noilor funcționalități, asigurând continuitatea și coerența datelor.

create table grupa\_de\_varsta(

create table activitati(

cod number(3) primary key,

nume varchar2(20) not null,

nivel\_dificultate varchar2(20) not null,

durata varchar2(20) not null

);

create table programari(

numar number(3) primary key,

data\_programare DATE not null,

ora\_start number(3,1) not null,

ora\_final number(3,1) not null,

cod\_ptt2 number(5) NOT NULL,

cod\_ate number(5) NOT NULL

);

create table angajati(

cod number(5) primary key,

nume varchar2(20) not null,

prenume varchar2(30) not null,

salariu number(4) not null,

nr\_telefon number(10) not null,

experienta number(3) not null,

tip\_agt char(1) not null,

cod\_ate2 number(3) NOT NULL,

instrument varchar2(30),

echipament varchar2(30),

materiale varchar2(30)

);

grupa number(3) primary key,

varsta\_min number(2) not null,

varsta\_max number(2) not null

);

create table participanti(

cod number(5) primary key,

nume varchar2(20) not null,

prenume varchar2(30) not null,

data\_nast DATE not null,

strada varchar2(30) not null,

nr number(5) not null,

ap number(4),

nr\_tel number(10) not null,

grupa\_gvd number(3) NOT NULL

);

create table plati(

numar number(3) primary key,

data\_plata DATE not null,

suma number(5) not null,

banca varchar2(50),

tip\_pta char(1) not null,

cod\_ptt number(5) NOT NULL

);

create table tarife(

id number(3) primary key,

nr\_sedinte number(2) not null,

pret number(5) not null,

grupa\_gvd2 number(3) NOT NULL );

Constrângerile pentru cheile străine și alte reguli de validare au fost definite anterior astfel:

ALTER TABLE plati

ADD CONSTRAINT check\_tip

CHECK (tip\_pta IN ('N', 'C'));

ALTER TABLE angajati

ADD CONSTRAINT chk\_tip

CHECK (tip\_agt IN ('M', 'S', 'D'));

ALTER TABLE programari

ADD CONSTRAINT start\_inainte\_de\_final

CHECK (ora\_start <= ora\_final);

ALTER TABLE participanti

ADD CONSTRAINT chk\_nr\_tel\_lungime

CHECK (LENGTH(TO\_CHAR(nr\_tel)) = 10);

ALTER TABLE angajati

ADD CONSTRAINT chk\_nr\_telefon\_lungime

CHECK (LENGTH(TO\_CHAR(nr\_telefon)) = 10)

ALTER TABLE participanti

ADD CONSTRAINT fk\_grupa\_gvd

FOREIGN KEY (grupa\_gvd) REFERENCES grupa\_de\_varsta(grupa);

ALTER TABLE plati

ADD CONSTRAINT fk\_cod\_ptt

FOREIGN KEY (cod\_ptt) REFERENCES participanti(cod);

ALTER TABLE tarife

ADD CONSTRAINT fk\_grupa\_gvd2

FOREIGN KEY (grupa\_gvd2) REFERENCES grupa\_de\_varsta(grupa);

ALTER TABLE programari

ADD CONSTRAINT fk\_cod\_ptt2

FOREIGN KEY (cod\_ptt2) REFERENCES participanti(cod);

ALTER TABLE programari

ADD CONSTRAINT fk\_cod\_ate

FOREIGN KEY (cod\_ate) REFERENCES activitati(cod);

ALTER TABLE angajati

ADD CONSTRAINT fk\_cod\_ate2

FOREIGN KEY (cod\_ate2) REFERENCES activitati(cod);

Tabelele au fost populate cu date relevante pentru a reflecta scenariile reale:

INSERT INTO grupa\_de\_varsta VALUES('1', '3', '7');

INSERT INTO grupa\_de\_varsta VALUES('2', '7', '11');

INSERT INTO grupa\_de\_varsta VALUES('3', '12', '15');

INSERT INTO grupa\_de\_varsta VALUES('4', '16', '18');

INSERT INTO participanti VALUES('33531', 'Avram', 'Alexandru', '2-SEP-2020', 'Trandafirilor', '31', '2', '07623780432', '1');

INSERT INTO participanti VALUES('33532', 'Georgescu', 'Ioana', '1-OCT-2011', 'Aleea Pacii', '44', null, '07424908665', '3');

INSERT INTO participanti VALUES('33533', 'Neagu', 'Sofia', '22-NOV-2013', 'Bulevardul Pacii', '3', null, '07129854308', '2');

INSERT INTO participanti VALUES('33534', 'Nistor', 'Carla', '22-JUN-2012', 'Intratrea Florilor', '67', null, '07677834669', '3');

INSERT INTO participanti VALUES('33535', 'Petrescu', 'Andrei', '9-JAN-2006', 'Calea Victoriei', '34', '44', '07990667452', '4');

INSERT INTO participanti VALUES('33536', 'Popa', 'Bogdan', '25-DEC-2013', 'Drumul Soartelui', '22', null, '07223471190', '2');

INSERT INTO participanti VALUES('33537', 'Preda', 'Ana', '6-MAR-2013', 'Splaiul Independentei', '45', null, '07226692357', '4');

INSERT INTO participanti VALUES('33538', 'Stan', 'Alexia', '18-SEP-2007', 'Piata Unirii', '6', '5', '07121234557', '4');

INSERT INTO participanti VALUES('33539', 'Stanciu', 'Eric', '15-MAR-2021', 'Aleea Frunzelor', '33', null, '07670098904', '1');

INSERT INTO participanti VALUES('33540', 'Ungureanu', 'Victor', '2-OCT-2006', 'Mihai Viteazu', '10', '18', '07434356678', '4');

INSERT INTO tarife VALUES('1', '14', '100', '1');

INSERT INTO tarife VALUES('2', '28', '200', '1');

INSERT INTO tarife VALUES('3', '14', '150', '2');

INSERT INTO tarife VALUES('4', '28', '300', '2');

INSERT INTO tarife VALUES('5', '14', '200', '3');

INSERT INTO tarife VALUES('6', '28', '400', '3');

INSERT INTO tarife VALUES('7', '14', '250', '4');

INSERT INTO tarife VALUES('8', '28', '500', '4');

INSERT INTO plati VALUES('100', '12-FEB-2024', '100', null, 'N', '33531');

INSERT INTO plati VALUES('101', '3-JAN-2024', '400', null, 'N', '33532');

INSERT INTO plati VALUES('102', '25-APR-2024', '300', 'Banca Transilvania', 'C', '33533');

INSERT INTO plati VALUES('103', '15-FEB-2024', '400', 'BRD', 'C', '33534');

INSERT INTO plati VALUES('104', '29-FEB-2024', '250', null, 'N', '33535');

INSERT INTO plati VALUES('105', '5-JAN-2024', '150', null, 'N', '33536');

INSERT INTO plati VALUES('106', '2-MAY-2024', '300', null, 'N', '33537');

INSERT INTO plati VALUES('107', '19-MAR-2024', '500', 'BRD', 'C', '33538');

INSERT INTO plati VALUES('108', '20-FEB-2024', '200', null, 'N', '33539');

INSERT INTO plati VALUES('109', '1-JAN-2024', '500', null, 'N', '33540');

INSERT INTO activitati VALUES('121', 'Curs de pian', 'avansat', '2 ore');

INSERT INTO activitati VALUES('122', 'Curs de chitara', 'incepator', '2 ore');

INSERT INTO activitati VALUES('123', 'Curs de canto', 'incepator', '2 ore');

INSERT INTO activitati VALUES('124', 'Curs de baschet', 'incepator', '2 ore');

INSERT INTO activitati VALUES('125', 'Curs de fotbal', 'incepator', '2 ore');

INSERT INTO activitati VALUES('126', 'Curs de gimnastica', 'mediu', '2 ore');

INSERT INTO activitati VALUES('127', 'Curs de inot', 'incepator', '2 ore');

INSERT INTO activitati VALUES('128', 'Curs de pictura', 'incepator', '2 ore');

INSERT INTO activitati VALUES('129', 'Curs de desen grafic', 'avansat', '2 ore');

INSERT INTO activitati VALUES('130', 'Curs pictura sticla', 'avansat', '2 ore');

INSERT INTO activitati VALUES('131', 'Curs de dans', 'incepator', '2 ore');

INSERT INTO programari VALUES('1', '11-APR-2024', '10', '12', '33531', '121');

INSERT INTO programari VALUES('2', '11-APR-2024', '12', '14', '33532', '124');

INSERT INTO programari VALUES('3', '11-APR-2024', '15', '17', '33533', '126');

INSERT INTO programari VALUES('4', '12-APR-2024', '9', '11', '33534', '129');

INSERT INTO programari VALUES('5', '12-APR-2024', '12', '14', '33535', '130');

INSERT INTO programari VALUES('6', '15-APR-2024', '15', '17', '33536', '123');

INSERT INTO programari VALUES('7', '15-APR-2024', '17', '19', '33537', '128');

INSERT INTO angajati VALUES('22340', 'Chiriac', 'Adrian', '3500', '07237759073', '8', 'M', '121', 'pian', null, null);

INSERT INTO angajati VALUES('22341', 'Ciobanu', 'Cristina', '3000', '07267895543', '7', 'M', '122', 'chitara', null, null);

INSERT INTO angajati VALUES('22342', 'Constantin', 'Andrei', '2500', '07099754876', '5', 'M', '123', 'microfon', null, null);

INSERT INTO angajati VALUES('22343', 'Duma', 'Mariana', '3500', '07998760098', '8', 'S', '124', null, 'minge baschet', null);

INSERT INTO angajati VALUES('22344', 'Dumitru', 'Carmen', '4000', '07443877659', '10', 'S', '125', null, 'minge fotbal', null);

INSERT INTO angajati VALUES('22345', 'Iliescu', 'Cristian', '2500', '07990876693', '5', 'S', '126', null, 'saltea', null);

INSERT INTO angajati VALUES('22346', 'Lungu', 'Dorin', '3000', '07554800987', '6', 'S', '127', null, 'pluta', null);

INSERT INTO angajati VALUES('22347', 'Munteanu', 'Vlad', '4000', '07454432760', '10', 'D', '128', null, null, 'vopsea');

INSERT INTO angajati VALUES('22348', 'Nistor', 'Sergiu', '4500', '07441184812', '15', 'D', '129', null, null, 'crioane');

INSERT INTO angajati VALUES('22349', 'Stanescu', 'Radu', '2000', '07998650097', '2', 'D', '130', null, null, 'sticla');

INSERT INTO angajati VALUES('22350', 'Stoica', 'Oana', '2500', '07995411254', '2', 'S', '131', null, 'costume', null);

1. modificările bazei de date

Pentru a pregăti baza de date pentru operațiile in PL/SQL, am efectuat câteva modificări de structură asupra tabelelor exitente și am adăugat încă câteva tabele. Aceste ajustări au fost necesare pentru a permite implementarea noilor funcționalități.

1. **Adăugarea coloanei durata\_totala în tabelul programări**

* pentru a înregistra durata fiecărei programări și permite calculul timpilor totali alocați, am adăugat o nouă coloană, durata\_totala în tabelul programari

ALTER TABLE programari

O imagine care conține captură de ecran, text, software, Software multimedia

Descriere generată automatADD durata\_totala NUMBER(5,2);

1. **Adăugarea coloanei status în tabelul plăți**

* Pentru a urmări starea fiecărei plăți și a face diferența între plățile efectuate și cele restante, am adăugat o coloana *status*, cu valoarea implicită *Neachitat*, în cadrul tabelului *plati*.

ALTER TABLE plati

ADD status VARCHAR2(20) DEFAULT 'Neachitat';

O imagine care conține captură de ecran, text

Descriere generată automat

1. **Crearea tabelului feedback**

* Pentru a colecta și analiza părerile participanților cu privire la activități, am creat un tabel *feedback*, care stochează comentariile, data oferirii feedback-ului și legătura cu participanții și activitățile aferente. De asemenea am adăugat și câteva populări pentru început.

create table feedback (

id\_feedback number(5) primary key,

id\_participant number(5) not null,

id\_activitate number(3) not null,

comentariu varchar2(500),

data\_feedback date not null

);

ALTER TABLE feedback

ADD CONSTRAINT fk\_feedback\_participant

FOREIGN KEY (id\_participant)

REFERENCES participanti(cod);

ALTER TABLE feedback

ADD CONSTRAINT fk\_feedback\_activitate

FOREIGN KEY (id\_activitate)

REFERENCES activitati(cod);

INSERT INTO feedback VALUES (1, 33531, 121, 'Excelent', SYSDATE);

INSERT INTO feedback VALUES (2, 33533, 123, 'Foarte bine', SYSDATE);

O imagine care conține text, captură de ecran, software

Descriere generată automatINSERT INTO feedback VALUES (3, 33535, 126, 'Bun', SYSDATE);

1. **Crearea tabelului istoric\_pret**

* Pentru a monitoriza schimbările de status ale plăților și a păstra un jurnal al modificărilor, am creat un tabel istoric\_plati, care înregistrează informații despre starea veche și cea nouă a plăților, împreună cu data modificării.

create table istoric\_plati (

id\_plata number(5),

status\_vechi varchar2(20),

status\_nou varchar2(20),

data\_modificare date DEFAULT SYSDATE

);

ALTER TABLE istoric\_plati

ADD CONSTRAINT fk\_istoric\_plati

FOREIGN KEY (id\_plata)

REFERENCES plati(numar);

O imagine care conține text, captură de ecran

Descriere generată automat

1. packages

Pentru a gestiona eficient operațiunile din cadrul bazei de date, am creat pachetul palatul\_copiilor\_package. Acesta include proceduri și funcții care automatizează procese esențiale, oferind astfel o soluție centralizată și organizată pentru administrarea datelor.

1. **Specificațiile pachetului**

* Specificațiile includ declarația unei variabile publice utilizate pentru mesaje personalizate, proceduri pentru adăugarea programărilor și gestionarea duratei acestora, actualizarea statusului plăților, precum și analiza feedback-ului. De asemenea, pachetul conține funcții pentru numărarea programărilor și calcularea detaliilor tarifare, asigurând o administrare eficientă și centralizată a operațiunilor.

create or replace PACKAGE palatul\_copiilor\_package IS

variabila\_publica CONSTANT VARCHAR2(50) := 'Palatul Copiilor - Management';

PROCEDURE adauga\_programare(

p\_numar IN programari.numar%TYPE,

p\_cod\_participant IN programari.cod\_ptt2%TYPE,

p\_cod\_activitate IN programari.cod\_ate%TYPE,

p\_data IN programari.data\_programare%TYPE,

p\_ora\_start IN programari.ora\_start%TYPE,

p\_ora\_final IN programari.ora\_final%TYPE

);

PROCEDURE seteaza\_durata\_totala;

FUNCTION numar\_programari RETURN NUMBER;

PROCEDURE actualizeaza\_status\_plata(

p\_id\_plata IN plati.numar%TYPE,

p\_status\_nou IN plati.status%TYPE

);

PROCEDURE afiseaza\_feedback\_detalii;

PROCEDURE adauga\_grupa\_varsta(

p\_varsta\_min IN grupa\_de\_varsta.varsta\_min%TYPE,

p\_varsta\_max IN grupa\_de\_varsta.varsta\_max%TYPE

);

FUNCTION total\_si\_detalii\_tarife RETURN NUMBER;

END palatul\_copiilor\_package;

1. **Corpul pachetului**
   1. Funcția **numar\_programari** este o funcție publică destinată să returneze și să afișeze numărul total de programări înregistrate în baza de date. Aceasta utilizează o interogare SQL pentru a calcula valoarea totală și afișează rezultatul într-un mesaj personalizat, folosind variabila publică definită în pachet. Funcția este utilă pentru monitorizarea rapidă a volumului programărilor gestionate.

create or replace PACKAGE BODY palatul\_copiilor\_package IS

-- functie publica: returneaza si afiseaza numarul total de programari

FUNCTION numar\_programari RETURN NUMBER IS

v\_total NUMBER;

BEGIN

SELECT COUNT(\*) INTO v\_total FROM programari;

DBMS\_OUTPUT.PUT\_LINE(variabila\_publica || ': Total programari - ' || v\_total);

RETURN v\_total;

END numar\_programari;

O imagine care conține text, captură de ecran, software, Software multimedia

Descriere generată automat

* 1. Funcția **calculeaza\_durata\_totala** este o funcție privată utilizată pentru a calcula durata totală a unei programări, pe baza orei de început și de sfârșit primite ca parametrii. Aceasta returnează valoarea calculată, fiind esențială pentru automatizarea și corectitudinea datelor în cadrul operațiunilor de gestionare a programărilor. Această funcție am create-o pentru a o putea utiliza ulterior în procedurile publice.

-- functie privata: calculeaza durata totala si returneaza valoarea

FUNCTION calculeaza\_durata\_totala(

p\_ora\_start IN programari.ora\_start%TYPE,

p\_ora\_final IN programari.ora\_final%TYPE

) RETURN programari.durata\_totala%TYPE IS

v\_durata programari.durata\_totala%TYPE;

BEGIN

v\_durata := p\_ora\_final - p\_ora\_start;

RETURN v\_durata;

END calculeaza\_durata\_totala;

* 1. Procedura **adauga\_programare** este o procedură publică destinată adăugării unei noi programări în baza de date. Aceasta calculează durata totală folosind funcția privată calculeaza\_durata\_totala și asigură înregistrarea corectă a datelor, gestionând eventualele erori prin blocuri de excepții.

-- procedura publica: adauga o programare

PROCEDURE adauga\_programare(

p\_numar IN programari.numar%TYPE,

p\_cod\_participant IN programari.cod\_ptt2%TYPE,

p\_cod\_activitate IN programari.cod\_ate%TYPE,

p\_data IN programari.data\_programare%TYPE,

p\_ora\_start IN programari.ora\_start%TYPE,

p\_ora\_final IN programari.ora\_final%TYPE

) IS

v\_durata programari.durata\_totala%TYPE;

BEGIN

-- calculam durata totala apeland functia privata

v\_durata := calculeaza\_durata\_totala(p\_ora\_start, p\_ora\_final);

INSERT INTO programari (

numar, cod\_ptt2, cod\_ate, data\_programare, ora\_start, ora\_final, durata\_totala

) VALUES (

p\_numar, p\_cod\_participant, p\_cod\_activitate, p\_data, p\_ora\_start, p\_ora\_final, v\_durata

);

DBMS\_OUTPUT.PUT\_LINE(variabila\_publica ||': Programare adaugata cu succes pentru participantul ' || p\_cod\_participant);

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: Programarea cu numarul ' || p\_numar || ' există deja.');

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: Valorile introduse sunt invalide.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare necunoscuta: ' || SQLERRM);

O imagine care conține text, captură de ecran, software, Software multimedia

Descriere generată automat END adauga\_programare;

* 1. Procedura **seteaza\_durata\_totala** este o procedură publică utilizată pentru actualizarea duratei totale a tuturor programărilor din tab, având în vedere că nu am populat aceasta coloană a tabelului. Aceasta folosește un cursor explicit pentru a parcurge fiecare programare și calculează durata totală apelând funcția privată calculeaza\_durata\_totala, asigurând actualizarea corectă a datelor.

-- procedura publica: seteaza durata totala pentru toate programarile folosind un cursor explicit

PROCEDURE seteaza\_durata\_totala IS

CURSOR cursor\_programari IS

SELECT numar, ora\_start, ora\_final FROM programari;

v\_durata programari.durata\_totala%TYPE;

BEGIN

FOR record IN cursor\_programari LOOP

-- calculam durata totala pentru fiecare programare apeland functia privata

v\_durata := calculeaza\_durata\_totala(record.ora\_start, record.ora\_final);

UPDATE programari

SET durata\_totala = v\_durata

WHERE numar = record.numar;

DBMS\_OUTPUT.PUT\_LINE(variabila\_publica ||': Durata totala pentru programarea ' || record.numar || ' a fost actualizata la ' || v\_durata || ' ore.');

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(variabila\_publica ||': Durata totala a fost setata pentru toate programarile.');

END seteaza\_durata\_totala;

O imagine care conține captură de ecran, astronomie

Descriere generată automat

O imagine care conține text, captură de ecran, software, Software multimedia

Descriere generată automat

* 1. Procedura **insereaza\_istoric** este o procedură privată folosită pentru a adăuga înregistrări în tabelul istoric\_plati. Aceasta stochează informații despre modificările de status ale unei plăți, inclusiv statusul anterior, cel nou și data modificării, contribuind la păstrarea unei evidențe clare a actualizărilor efectuate.

-- procedura privata: insereaza in istoricul platilor

PROCEDURE insereaza\_istoric(

p\_id\_plata IN istoric\_plati.id\_plata%TYPE,

p\_status\_vechi IN istoric\_plati.status\_vechi%TYPE,

p\_status\_nou IN istoric\_plati.status\_nou%TYPE

) IS

BEGIN

INSERT INTO istoric\_plati (

id\_plata, status\_vechi, status\_nou, data\_modificare

) VALUES (

p\_id\_plata, p\_status\_vechi, p\_status\_nou, SYSDATE

);

DBMS\_OUTPUT.PUT\_LINE(variabila\_publica ||': Istoric adaugat pentru plata ' || p\_id\_plata || '.');

END insereaza\_istoric;

* 1. Procedura **actualizeaza\_status\_plata** este o procedură publică care gestionează actualizarea statusului unei plăți în tabelul plati. Prin utilizarea unui cursor explicit, aceasta obține statusul vechi al plății și îl actualizează cu cel nou. De asemenea, procedura inserează modificarea în tabelul istoric\_plati, utilizând procedura privată insereaza\_istoric, pentru a menține o evidență a schimbărilor efectuate.

-- procedura publica: actualizeaza statusul unei plati si insereaza in istoric

PROCEDURE actualizeaza\_status\_plata(

p\_id\_plata IN plati.numar%TYPE,

p\_status\_nou IN plati.status%TYPE

) IS

v\_status\_vechi plati.status%TYPE;

-- cursor explicit cu parametri

CURSOR cursor\_plata(p\_numar\_plata plati.numar%TYPE) IS

SELECT status FROM plati WHERE numar = p\_numar\_plata;

BEGIN

OPEN cursor\_plata(p\_id\_plata);

FETCH cursor\_plata INTO v\_status\_vechi;

CLOSE cursor\_plata;

UPDATE plati

SET status = p\_status\_nou

WHERE numar = p\_id\_plata;

DBMS\_OUTPUT.PUT\_LINE(variabila\_publica ||': Status actualizat pentru plata ' || p\_id\_plata || ' la ' || p\_status\_nou || '.');

insereaza\_istoric(p\_id\_plata, v\_status\_vechi, p\_status\_nou);

END actualizeaza\_status\_plata; O imagine care conține text, captură de ecran, software, Software multimedia

Descriere generată automat

O imagine care conține captură de ecran, text, număr

Descriere generată automat

* 1. Procedura **genereaza\_statistici\_feedback** este o procedură privată care compilează și afișează statistici despre feedback-ul primit de la participanți. Aceasta utilizează diverse tipuri de cursoare pentru a extrage informații relevante: un cursor implicit pentru a număra feedback-urile, un cursor explicit pentru a parcurge fiecare feedback, un cursor explicit cu parametri pentru a afișa comentariile unui participant specific și un cursor multiplu pentru iterarea peste toate feedback-urile. Procedura contribuie la monitorizarea și analiza calității serviciilor oferite.

-- procedura privata: genereaza statistici despre feedback

PROCEDURE genereaza\_statistici\_feedback IS

v\_total\_feedback NUMBER;

-- cursor explicit

CURSOR feedback\_cursor IS

SELECT id\_feedback, comentariu FROM feedback;

-- cursor explicit cu parametri

CURSOR feedback\_participant\_cursor(p\_id\_participant feedback.id\_participant%TYPE) IS

SELECT comentariu FROM feedback WHERE id\_participant = p\_id\_participant;

-- cursor multiplu

TYPE feedback\_record\_type IS RECORD (

id\_feedback feedback.id\_feedback%TYPE,

comentariu feedback.comentariu%TYPE

);

CURSOR feedback\_multiplu\_cursor IS

SELECT id\_feedback, comentariu FROM feedback;

v\_feedback\_record feedback\_record\_type;

BEGIN

-- cursor implicit: numar total de feedback-uri

SELECT COUNT(\*) INTO v\_total\_feedback FROM feedback;

DBMS\_OUTPUT.PUT\_LINE('Total feedback-uri: ' || v\_total\_feedback);

-- cursor explicit: afiseaza fiecare feedback

FOR feedback\_row IN feedback\_cursor LOOP

DBMS\_OUTPUT.PUT\_LINE('Feedback ID: ' || feedback\_row.id\_feedback || ', Comentariu: ' || feedback\_row.comentariu);

END LOOP;

-- cursor explicit cu parametri: feedback pentru un participant specific

OPEN feedback\_participant\_cursor(33531); -- exemplu de participant

LOOP

FETCH feedback\_participant\_cursor INTO v\_feedback\_record.comentariu;

EXIT WHEN feedback\_participant\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Feedback pentru participantul 33531: ' || v\_feedback\_record.comentariu);

END LOOP;

CLOSE feedback\_participant\_cursor;

-- cursor multiplu: iterare peste toate feedback-urile

OPEN feedback\_multiplu\_cursor;

LOOP

FETCH feedback\_multiplu\_cursor INTO v\_feedback\_record;

EXIT WHEN feedback\_multiplu\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Feedback ID: ' || v\_feedback\_record.id\_feedback || ', Comentariu: ' || v\_feedback\_record.comentariu);

END LOOP;

CLOSE feedback\_multiplu\_cursor;

END genereaza\_statistici\_feedback;

* 1. Procedura **afiseaza\_feedback\_detalii** este oprocedură publică utilizată pentru a afișa detalii generale despre feedback-ul primit de la participanți. Aceasta apelează procedura privată genereaza\_statistici\_feedback pentru a colecta și afișa statistici detaliate despre feedback-uri, integrând astfel informațiile într-un mod accesibil și centralizat.

-- procedura publica: afiseaza detalii despre feedback si genereaza statistici

PROCEDURE afiseaza\_feedback\_detalii IS

BEGIN

DBMS\_OUTPUT.PUT\_LINE(variabila\_publica ||': --- Detalii despre feedback ---');

-- apel procedura privata

genereaza\_statistici\_feedback;

END afiseaza\_feedback\_detalii;

O imagine care conține text, captură de ecran, software, Font

Descriere generată automat

* 1. Procedura **adauga\_grupa\_varsta** este o procedură publică care permite adăugarea unei noi grupe de vârstă în baza de date. Aceasta generează automat un ID unic folosind o secvență și inserează detalii precum intervalul de vârstă în tabelul grupa\_de\_varsta. Am creat această procedură deoarece dorim extinderea activităților și pentru persoanele care au vârsta de 18-20 de ani.
* secvența folosită este următoarea

CREATE SEQUENCE seq\_grupa\_de\_varsta START WITH 5 INCREMENT BY 1;

-- procedura publica: adauga o grupa de varsta

PROCEDURE adauga\_grupa\_varsta(

p\_varsta\_min IN grupa\_de\_varsta.varsta\_min%TYPE,

p\_varsta\_max IN grupa\_de\_varsta.varsta\_max%TYPE

) IS

v\_grupa\_id grupa\_de\_varsta.grupa%TYPE;

BEGIN

-- generare ID folosind secvența

v\_grupa\_id := seq\_grupa\_de\_varsta.NEXTVAL;

INSERT INTO grupa\_de\_varsta (grupa, varsta\_min, varsta\_max)

VALUES (v\_grupa\_id, p\_varsta\_min, p\_varsta\_max);

DBMS\_OUTPUT.PUT\_LINE(variabila\_publica ||': Grupa de varsta adaugata cu ID ' || v\_grupa\_id || ', Interval: ' || p\_varsta\_min || '-' || p\_varsta\_max || ' ani.');

O imagine care conține text, captură de ecran, software, Software multimedia

Descriere generată automat END adauga\_grupa\_varsta;

O imagine care conține captură de ecran, text

Descriere generată automat

* 1. Funcția **total\_si\_detalii\_tarife** esteo funcție publică care afișează detalii despre fiecare tarif existent în baza de date și calculează suma totală a prețurilor. Utilizând un cursor multiplu, aceasta parcurge toate înregistrările din tabelul tarife, afișează informații precum ID-ul tarifului, numărul de ședințe, prețul și grupa de vârstă asociată, iar la final returnează suma totală a prețurilor.

-- functie publica: afiseaza detalii despre tarife si returneaza suma totala

FUNCTION total\_si\_detalii\_tarife RETURN NUMBER IS

--cursor multiplu

TYPE tarife\_record IS RECORD (

id\_tarif tarife.id%TYPE,

nr\_sedinte tarife.nr\_sedinte%TYPE,

pret tarife.pret%TYPE,

grupa\_gvd2 tarife.grupa\_gvd2%TYPE

);

CURSOR tarife\_cursor IS

SELECT id, nr\_sedinte, pret, grupa\_gvd2

FROM tarife;

v\_tarif tarife\_record;

v\_total\_pret NUMBER := 0;

BEGIN

OPEN tarife\_cursor;

LOOP

FETCH tarife\_cursor INTO v\_tarif;

EXIT WHEN tarife\_cursor%NOTFOUND;

v\_total\_pret := v\_total\_pret + v\_tarif.pret;

DBMS\_OUTPUT.PUT\_LINE('Tarif ID: ' || v\_tarif.id\_tarif ||

', Nr. sedinte: ' || v\_tarif.nr\_sedinte ||

', Pret: ' || v\_tarif.pret ||

', Grupa varsta: ' || v\_tarif.grupa\_gvd2);

END LOOP;

CLOSE tarife\_cursor;

RETURN v\_total\_pret;

END total\_si\_detalii\_tarife;

END palatul\_copiilor\_package;

O imagine care conține text, electronice, captură de ecran, software

Descriere generată automat

1. triggers

Pentru a asigura integritatea și validitarea datelor din baza de date, am implementat doi triggeri care verifică anumite reguli și constrângeri în momentul inserării sau actualizării datelor. Acești triggeri contribuie la prevenirea introducerii de date eronate și la menținerea unei gestionări corecte a operațiunilor.

1. Trigger **trg\_check\_programare\_suprapunere**

* Triggerul trg\_check\_programare\_suprapunere previne inserarea de programări suprapuse pentru același participant în aceeași zi. Este activat înainte de operațiunile de tip INSERT pe tabelul programari și verifică dacă intervalul orar al noii programări se suprapune cu programările existente. Dacă se detectează o suprapunere, operațiunea este blocată și se afișează un mesaj de eroare. Această validare asigură integritatea datelor și o planificare eficientă.

create or replace TRIGGER trg\_check\_programare\_suprapunere

BEFORE INSERT ON programari

FOR EACH ROW

DECLARE

v\_count NUMBER;

BEGIN

SELECT COUNT(\*)

INTO v\_count

FROM programari

WHERE cod\_ptt2 = :NEW.cod\_ptt2

AND data\_programare = :NEW.data\_programare

AND (

(:NEW.ora\_start < ora\_final AND :NEW.ora\_final > ora\_start)

);

IF v\_count > 0 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Exista deja o programare pentru acest participant in acest interval de timp.');

END IF;

END;

O imagine care conține text, captură de ecran, software, Software multimedia

Descriere generată automat

1. Trigger **trg\_check\_plati\_sume\_valide**

* Triggerul trg\_check\_plati\_suma\_valida validează și ajustează datele înainte de inserarea sau actualizarea în tabelul plati. Acesta asigură că suma plății este mai mare decât zero, blocând operațiunea și afișând un mesaj de eroare în caz contrar. De asemenea, dacă banca nu este specificată, triggerul setează automat valoarea implicită "Banca Necunoscută". Astfel, triggerul contribuie la menținerea integrității și coerenței datelor din tabelul plati.

create or replace TRIGGER trg\_check\_plati\_suma\_valida

BEFORE INSERT OR UPDATE ON plati

FOR EACH ROW

BEGIN

IF :NEW.suma <= 0 THEN

RAISE\_APPLICATION\_ERROR(

-20003,

'Eroare: Suma plății trebuie să fie mai mare decât 0.'

);

END IF;

IF :NEW.banca IS NULL THEN

:NEW.banca := 'Banca Necunoscută';

END IF;

END;

O imagine care conține text, Font, Software multimedia, software

Descriere generată automat

***BAZA DE DATE MongoDB***

1. scenariu

***BAZA DE DATE MongoDB***

Palatul Copiilor din Cluj-Napoca este o instituție dedicată educației și dezvoltării copiilor cu vârste între 3 și 18 ani, care le oferă oportunități de a participa la o gamă variată de cursuri și activități educative, recreative și sportive. Scopul principal al acestei instituții este de a sprijini dezvoltarea copiilor în domenii precum muzica, sportul și artele vizuale, într-un mediu educativ. Activitățile sunt organizate pe grupe de vârstă, iar tarifele pentru cursuri și servicii sunt stabilite în funcție de numărul de ședințe selectate de fiecare participant.

Gruparea copiilor în categorii de vârstă este esențială, deoarece tarifele pentru activitățile desfășurate sunt stabilite în funcție de acestea. Astfel, fiecare copil este încadrat într-o anumită grupă de vârstă, asigurându-ne că serviciile noastre sunt accesibile și personalizate nevoilor specifice ale fiecărui participant. Tarifele noastre depind de asemenea de numărul de ședințe ales de fiecare copil, noi având disponibile două pachete, unul de 14 ședințe și unul de 28 de ședințe (fiecare ședință având durata de 2 ore). De menționat este și faptul că participarea la cursuri se face pe bază de programare. După stabilirea datei și a orei în care copilul va participa la curs, tot ce mai trebuie făcut este ca ședința să fie achitată (oferim posibilitatea de a achita cash sau card).

Pentru o gestionare mai eficientă a activităților și serviciilor oferite și pentru a facilita monitorizarea programărilor și plăților, am decis să dezvoltăm o bază de date care să susțină operațiunile zilnice ale Palatului Copiilor. Aceasta va sprijini procesul de înregistrare al participanților, gestionarea programărilor pentru activități, monitorizarea plăților și gestionarea angajaților implicați în desfășurarea activităților.

În cadrul acestui proiect, am adaptat baza de date inițială creată în Oracle pentru a corespunde cerințelor unui sistem bazat pe MongoDB, doarece acesta oferă mai multă flexibilitate datorită structurii sale bazate pe documente imbricate. Astfel, am creat colecțiile participant, programare și angajat, fiecare având o structură care reflectă cerințele aplicației, înlocuind tabelele relaționale cu documente imbricate.

Schema bazei de date a fost optimizată pentru MongoDB. Am utilizat referințe între colecții, folosind ObjectId și identificatoare personalizate pentru a conecta documentele între ele, eliminând astfel necesitatea tabelelor de legături din sistemul relațional. Baza de date a fost populată cu datele necesare pentru testare și va permite gestionarea eficientă a tuturor operațiunilor în cadrul Palatului Copiilor. De asemenea, am realizat modificări structurale în baza de date, efectuând actualizări și ștergeri de documente pentru a păstra datele corecte și actualizate.

La final, am realizat interogări pentru a verifica integritatea datelor. Aceste interogări au fost efectuate în MongoDB Compass, unde am vizualizat documentele și am confirmat că structura bazei de date este funcțională și se aliniază cerințelor proiectului.

1. **schema bazei de date**

În cadrul acestui proiect, am reproiectat baza de date creată inițial in Oracle pentru o adapta la cerințele unui sistem bazat pe MongoDB. Astfel, am reconfigurat tabelele relaționale existente într-o structură de colecții imbricate.

Schema bazei de date în Oracle includea mai multe tabele, fiecare având rolul său foarte important în gestionarea Palatului. Tabelele incluse în schema inițială sunt:

* *grupa\_de\_varsta* - stochează intervalele de vârstă și le asociază cu diferite tarife.
* *participanti* - conține informații despre participanți (inclusiv grupa de vârstă).
* *tarife* - stochează tarifele pentru fiecare grupă de vârstă.
* *plati* - ține evidența plăților efectuate de participanți.
* *programari* - gestionează programările pentru cursuri și activități
* *angajati* - conține informații despre angajații (inclusiv activitățile de care se ocupă)
* *activitati* - conține informațiile despre toate activitățile

În MongoDB, am ajustat schema astfel încât fiecare colecție să conțină informațiile esențiale pentru domeniul specific al aplicației. Am decis să folosesc colecții imbricate în acest sens. Am redus complexitatea relațiilor între tabele, profitând de documentele imbricate. Astfel am creat următoarele colecții:

1. **colecția participant**

* în MongoDB, am creat colecția participant pentru a stoca informațiile legate de participanți, iar pentru a simplifica interogările, am refăcut informațiile legate de grupa de vârstă și tarifele într-un document imbricat. Astfel, în colecția participant, fiecare participant include acum un câmp tarife, care este un array de obiecte ce conțin informațiile de tarif specifice grupei de vârstă. În plus, am păstrat un câmp adresa care este și el un obiect imbricat, ce conține detalii precum strada, numărul, apartamentul și telefonul.

1. **colecția programare**

* în schema inițială, activitățile și programările erau gestionate într-un tabel separat. În MongoDB, am ales să utilizez un model de date imbricate în colecția programare, unde informațiile referitoare la plățile unui participant sunt stocate direct ca un obiect în interiorul documentului de programare. În loc să am mai multe tabele de legături, am introdus câmpuri de tip participant\_id (referință la colecția participant) și am păstrat activitățile în cadrul aceluiași document. De asemenea, am folosit ObjectId pentru a crea legături între colecțiile relevante, fără a necesită seturi complexe de tabele de legături.

1. **colecția angajat**

* în loc de a păstra informațiile despre angajați într-un tabel separat, am ales să creez o colecție angajat în MongoDB, care include un câmp activități sub formă de array. Acest câmp conține obiecte imbricate care descriu fiecare activitate de care angajatul este responsabil, incluzând numele activității, nivelul de dificultate și durata activității.

1. **crearea bazei de date**

În continuare, voi prezenta cum am creat baza de date utilizând terminalul CMD și cum am verificat comenzile folosind Mongo Compass.

1. **Crearea bazei de date**

* am folosit comanda use palatul\_copiilor pentru a crea baza de date palatul\_copiilor

1. **Crearea colecției participant**

* am folosit comanda db.createCollection("participant") pentru a crea prima colecție
* am inserat un document exemplu (drept test), care să conțină informații despre un participant. Pentru acestă comandă am folosit *ObjectId generat automat* pentru identificarea unică a documentului în colecție. Documentul include un obiect imbricat pentru tarife și adresă, care conțin informații suplimentare legate de participant.

db.participant.insertOne({

"nume": "Ion Popescu",

"prenume": "Ion",

"data\_nasterii": new Date("2008-09-25"),

"grupa\_gvd": 3,

"tarife": [

{

"nr\_sedinte": 14,

"pret": 150,

"grupa\_gvd": 3

}

],

"adresa": {

"strada": "Strada Mare",

"numar": "10",

"apartament": "5",

"telefon": "0712345678"

}

});

* am folosit comanda db.participant.find().pretty() pentru a verifica dacă datele au fost corect adăugate pentru participant

O imagine care conține text, captură de ecran

Descriere generată automat

* în continuare am inserat mai multe documente în colecția participant folosind comanda insertMany() pentru a putea insera cu ușurință mai multe documente dintr-o singura comanda. Fiecărui participant i-am generat un ObjectId automat.

db.participant.insertMany([

{

"nume": "Avram",

"prenume": "Alexandru",

"data\_nasterii": new Date("2020-09-02"),

"grupa\_gvd": 1,

"tarife": [ { "nr\_sedinte": 14, "pret": 100, "grupa\_gvd": 1 } ],

"adresa": { "strada": "Trandafirilor", "numar": "31", "apartament": "2", "telefon": "07623780432" }

},

{

"nume": "Georgescu",

"prenume": "Ioana",

"data\_nasterii": new Date("2011-10-01"),

"grupa\_gvd": 3,

"tarife": [ { "nr\_sedinte": 28, "pret": 200, "grupa\_gvd": 3 } ],

"adresa": { "strada": "Aleea Pacii", "numar": "44", "telefon": "07424908665" }

},

{

"nume": "Neagu",

"prenume": "Sofia",

"data\_nasterii": new Date("2013-11-22"),

"grupa\_gvd": 2,

"tarife": [ { "nr\_sedinte": 14, "pret": 150, "grupa\_gvd": 2 } ],

"adresa": { "strada": "Bulevardul Pacii", "numar": "3", "telefon": "07129854308" }

},

{

"nume": "Nistor",

"prenume": "Carla",

"data\_nasterii": new Date("2012-06-22"),

"grupa\_gvd": 3,

"tarife": [ { "nr\_sedinte": 28, "pret": 300, "grupa\_gvd": 3 } ],

"adresa": { "strada": "Intratrea Florilor", "numar": "67", "telefon": "07677834669" }

},

{

"nume": "Petrescu",

"prenume": "Andrei",

"data\_nasterii": new Date("2006-01-09"),

"grupa\_gvd": 4,

"tarife": [ { "nr\_sedinte": 14, "pret": 200, "grupa\_gvd": 4 } ],

"adresa": { "strada": "Calea Victoriei", "numar": "34", "telefon": "07990667452" }

},

{

"nume": "Popa",

"prenume": "Bogdan",

"data\_nasterii": new Date("2013-12-25"),

"grupa\_gvd": 2,

"tarife": [ { "nr\_sedinte": 14, "pret": 150, "grupa\_gvd": 2 } ],

"adresa": { "strada": "Drumul Soartelui", "numar": "22", "telefon": "07223471190" }

}

]);

O imagine care conține text, captură de ecran, afișaj, Software multimedia

Descriere generată automat

* deoarece primul participant a fost adăugat doar drept test, am ales să îl șterg din baza de date folosind următoarea comandă:

db.participant.deleteOne({ "\_id": ObjectId("677282ca2c81cb47310d8190") });

O imagine care conține text, captură de ecran, Font

Descriere generată automat

* O imagine care conține text, captură de ecran, software, Pictogramă computer

  Descriere generată automatcolecția finală pentru participanți testată in Mongo Compass:

O imagine care conține text, captură de ecran, software, Pictogramă computer

Descriere generată automat

O imagine care conține text, captură de ecran, software, număr

Descriere generată automat

1. **Crearea colecției programare**

* am folosit comanda db.createCollection("programare") pentru a crea cea de a doua colecție
* am inserat un document exemplu în colecția programare utilizând comanda insertOne(). Acest document reprezintă o programare a unui participant la o activitate, conținând câmpuri esențiale cum ar fi id (un identificator personalizat care utilizează o valoare unică), data programării, ora de start si final, id-ul participantului (referința către participantul asociat), activitate și plată (care e un obiect imbricat care conține detalii despre plata efectuată cum ar fi data plății, suma, tipul și banca)

db.programare.insertOne({

"\_id": "programare\_001",

"data\_programare": new Date("2024-05-10T10:00:00Z"),

"ora\_start": 10,

"ora\_final": 12,

"participant\_id": ObjectId("605c72ef1532071f3d2e68f4"),

"activitate": "pian",

"plata": {

"data\_plata": new Date("2024-01-15"),

"suma": 150,

"tip\_pta": "N",

"banca": null

}

});

* am folosit comanda db.programare.find().pretty(); pentru a verifica dacă datele au fost corect adăugate

O imagine care conține text, captură de ecran, software

Descriere generată automat

* am inserat mai multe programări in colecția programare dupa modelul prezentat mai sus folosind comanda insertMany(). La participant\_id am adăugat id-urile generate anterior pentru fiecare participant, astfel realizând legătura dintre cele doua colecții

db.programare.insertMany([

{

"\_id": "programare\_002",

"data\_programare": new Date("2024-05-10T10:00:00Z"),

"ora\_start": 10,

"ora\_final": 12,

"participant\_id": ObjectId("67728cae2c81cb47310d8191"),

"activitate": "Curs de pian",

"plata": { "data\_plata": new Date("2024-01-15"), "suma": 150, "tip\_pta": "N", "banca": null }

},

{

"\_id": "programare\_003",

"data\_programare": new Date("2024-05-11T12:00:00Z"),

"ora\_start": 12,

"ora\_final": 14,

"participant\_id": ObjectId("67728cae2c81cb47310d8192"),

"activitate": "Curs de chitara",

"plata": { "data\_plata": new Date("2024-01-16"), "suma": 200, "tip\_pta": "N", "banca": null }

},

{

"\_id": "programare\_004",

"data\_programare": new Date("2024-05-12T15:00:00Z"),

"ora\_start": 15,

"ora\_final": 17,

"participant\_id": ObjectId("67728cae2c81cb47310d8193"),

"activitate": "Curs de canto",

"plata": { "data\_plata": new Date("2024-01-17"), "suma": 250, "tip\_pta": "C", "banca": "Banca Transilvania" }

},

{

"\_id": "programare\_005",

"data\_programare": new Date("2024-05-13T09:00:00Z"),

"ora\_start": 9,

"ora\_final": 11,

"participant\_id": ObjectId("67728cae2c81cb47310d8194"),

"activitate": "Curs de baschet",

"plata": { "data\_plata": new Date("2024-01-18"), "suma": 300, "tip\_pta": "C", "banca": "BRD" }

},

{

"\_id": "programare\_006",

"data\_programare": new Date("2024-05-14T12:00:00Z"),

"ora\_start": 12,

"ora\_final": 14,

"participant\_id": ObjectId("67728cae2c81cb47310d8195"),

"activitate": "Curs de fotbal",

"plata": { "data\_plata": new Date("2024-01-19"), "suma": 350, "tip\_pta": "N", "banca": null }

},

{

"\_id": "programare\_007",

"data\_programare": new Date("2024-05-15T15:00:00Z"),

"ora\_start": 15,

"ora\_final": 17,

"participant\_id": ObjectId("67728cae2c81cb47310d8196"),

"activitate": "Curs de gimnastica",

"plata": { "data\_plata": new Date("2024-01-20"), "suma": 400, "tip\_pta": "N", "banca": null }

}

]);

O imagine care conține text, captură de ecran, afișaj, software

Descriere generată automat

* am ales sa folosesc o operație de ștergere pentru prima programare deoarece a fost adăugata doar drept test

db.programare.deleteOne({ "\_id": "programare\_001" });

O imagine care conține text, captură de ecran, Font

Descriere generată automat

* în continuare, voi realiza o actualizare pentru colecția programari pentru a include informații suplimentare despre activitățile fiecărei programări. Fiecare document din colecția programare a fost modificat pentru a include un obiect imbricat activitate, care cuprinde numele activității, nivelul de dificultate si durata activității.

db.programare.updateOne({ "\_id": "programare\_002" },

{ $set: { "activitate": { "activitate": "Curs de pian", "nivel\_dificultate": "Incepator", "durata": "2 ore" } } });

db.programare.updateOne({ "\_id": "programare\_003" },

{ $set: { "activitate": { "activitate": "Curs de chitara", "nivel\_dificultate": "Incepator", "durata": "2 ore" } } });

db.programare.updateOne({ "\_id": "programare\_004" },

{ $set: { "activitate": { "activitate": "Curs de canto", "nivel\_dificultate": "Incepator", "durata": "2 ore" } } });

db.programare.updateOne({ "\_id": "programare\_005" },

{ $set: { "activitate": { "activitate": "Curs de baschet", "nivel\_dificultate": "Incepator", "durata": "2 ore" } } });

db.programare.updateOne({ "\_id": "programare\_006" },

{ $set: { "activitate": { "activitate": "Curs de fotbal", "nivel\_dificultate": "Incepator", "durata": "2 ore" } } });

db.programare.updateOne({ "\_id": "programare\_007" },

{ $set: { "activitate": { "activitate": "Curs de gimnastica", "nivel\_dificultate": "Mediu", "durata": "2 ore" } } });

O imagine care conține text, captură de ecran

Descriere generată automat

* colecția finală pentru programări testată in Mongo Compass:

O imagine care conține text, captură de ecran, software, Pictogramă computer

Descriere generată automat

O imagine care conține text, captură de ecran, software, număr

Descriere generată automat

O imagine care conține text, captură de ecran, software, Pictogramă computer

Descriere generată automat

1. **Crearea colecției angajat**

* am creat ultima colecție folosind comanda db.createCollection("programare")
* am inserat un document exemplu in care conține numele si prenumele angajatului, experiența, salariul și numărul de telefon ale acestuia, dar și un array de obiecte, fiecare obiect descriind activitățile pe care angajatul le desfășoară, cu informații despre numele activității, nivelul de dificultate și durata activității. De asemenea, id-ul este un identificator personalizat pentru angajat, stabilit ca "angajat\_001"

db.angajat.insertOne({

"\_id": "angajat\_001",

"nume": "Cristina Ciobanu",

"prenume": "Cristina",

"experienta": 7,

"salariu": 3000,

"telefon": "07267895543",

"activitati": [

{

"activitate": "Curs de chitara",

"nivel\_dificultate": "Incepator",

"durata": "2 ore"

},

{

"activitate": "Curs de pian",

"nivel\_dificultate": "Avansat",

"durata": "2 ore"

}

]

});

* O imagine care conține text, captură de ecran, Software multimedia, software

  Descriere generată automatverificarea am efectuat-o folosind comanda db.angajat.find().pretty();
* am inserat mai multe documente în colecția angajat folosind comanda insertMany(), care permite adăugarea simultană a mai multor angajați. Fiecare document conține informații detaliate despre angajat, cum ar fi numele, experiența, salariul, telefonul și activitățile desfășurate. Activitățile sunt structurate ca un array, incluzând informații despre nivelul de dificultate și durata fiecărei activități.

db.angajat.insertMany([

{

"\_id": "angajat\_002",

"nume": " Ciobanu",

"prenume": "Cristina",

"experienta": 7,

"salariu": 3000,

"telefon": "07267895543",

"activitati": [

{"activitate": "Curs de chitara", "nivel\_dificultate": "Incepator", "durata": "2 ore"},

{"activitate": "Curs de pian", "nivel\_dificultate": "Avansat", "durata": "2 ore"}]},

{

"\_id": "angajat\_003",

"nume": " Chiriac",

"prenume": "Adrian",

"experienta": 8,

"salariu": 3500,

"telefon": "07237759073",

"activitati": [

{"activitate": "Curs de pian", "nivel\_dificultate": "Avansat", "durata": "2 ore"}] },

{

"\_id": "angajat\_004",

"nume": " Lungu",

"prenume": "Dorin",

"experienta": 6,

"salariu": 3000,

"telefon": "07267895543",

"activitati": [

{"activitate": "Curs de inot", "nivel\_dificultate": "Incepator", "durata": "2 ore"} ]},

{

"\_id": "angajat\_005",

"nume": " Constantin",

"prenume": "Andrei",

"experienta": 5,

"salariu": 2500,

"telefon": "07099754876",

"activitati": [

{"activitate": "Curs de canto", "nivel\_dificultate": "Incepator", "durata": "2 ore"}]},

{

"\_id": "angajat\_006",

"nume": " Duma",

"prenume": "Mariana",

"experienta": 8,

"salariu": 3500,

"telefon": "07998760098",

"activitati": [

{"activitate": "Curs de baschet", "nivel\_dificultate": "Incepator", "durata": "2 ore"}]},

{

"\_id": "angajat\_007",

"nume": " Dumitru",

"prenume": "Carmen",

"experienta": 10,

"salariu": 4000,

"telefon": "07443877659",

"activitati": [

{"activitate": "Curs de fotbal", "nivel\_dificultate": "Incepator", "durata": "2 ore"}

]}

]);

O imagine care conține text, captură de ecran, afișaj, Software multimedia

Descriere generată automat

* voi realiza și pentru această colecție operația de ștergere pentru primul angajat, care a fost adăugat drept test.

db.angajat.deleteOne({ "\_id": "angajat\_001" });

O imagine care conține text, captură de ecran, Font

Descriere generată automat

* colecția finală pentru angajat testată in Mongo Compass:

O imagine care conține text, captură de ecran, software, Pictogramă computer

Descriere generată automat

O imagine care conține text, captură de ecran, software, Pictogramă computer

Descriere generată automat

O imagine care conține text, captură de ecran, software, număr

Descriere generată automat

1. **interogări**

Realizarea interogărilor am început-o prin a crea 2 **index-uri** în colecțiile angajat și participant.

Pentru colecția angajat, am creat un index care permite sortarea angajaților după experiență și salariu.

db.angajat.createIndex({ "experienta": 1, "salariu": -1 });

Pentru colecția participant, am creat un index care permite sortarea participanțiilor după grupa de vârstă și strada.

db.participant.createIndex({ "grupa\_gvd": 1, "adresa.strada": 1 });

În continuare, voi prezenta câteva interogări realizate pentru baza de date creată anterior, realizate pentru a căuta informații specifice în baza noastră de date.

1. Găsirea programărilor pentru un anumit participant, sortate după data și ora la care au loc

**Enunț:** Găsiți programările pentru un participant specificat, sortate în ordine crescătoare după data și ora de start.

db.programare.find({ "participant\_id": ObjectId("67728cae2c81cb47310d8191") }).sort({ "data\_programare": 1, "ora\_start": 1 }).pretty();

O imagine care conține text, captură de ecran, software, Software multimedia

Descriere generată automat

1. Găsirea angajaților cu cel puțin 7 ani experiență, sortați după salariu

**Enunț:** Găsiți angajații care au o experiență de cel puțin 7 ani și sortați-i în funcție de salariu, în ordine descrescătoare.

O imagine care conține text, captură de ecran, afișaj, software

Descriere generată automatdb.angajat.find({ "experienta": { $gte: 7 } }).sort({"salariu":-1}).pretty();

1. Găsirea numărului de angajați implicați în fiecare activitate, grupat pe nivelul de dificultate

**Enunț:** Aflați câți angajați sunt implicați în fiecare activitate, grupându-i pe nivelul de dificultate (începător, mediu, avansat).

db.angajat.aggregate([

{

$unwind: "$activitati"

},

{

$group: {

\_id: { activitate: "$activitati.activitate", nivel\_dificultate: "$activitati.nivel\_dificultate" },

angajati\_count: { $sum: 1 }

}

},

{

$sort: { "angajati\_count": -1 }

}])

O imagine care conține text, captură de ecran, software

Descriere generată automat

* Această interogare utilizează framework-ul de agregare pentru a obține numărul de angajați implicați în fiecare activitate, grupându-i pe nivelul de dificultate. Se folosește $unwind pentru a desface array-ul de activități și $group pentru a agrega datele.

1. Sortarea participanților după grupă și stradă

**Enunț:** Obțineți o listă a tuturor participanților, sortată crescător în funcție de grupa de vârstă și în cadrul grupelor, după strada acestora.

db.participant.find().sort({ "grupa\_gvd": 1, "adresa.strada": 1 }).pretty();

O imagine care conține text, captură de ecran

Descriere generată automat

1. Calcularea vârstei medii a participanților și numărul acestora, pe grupă de vârstă

**Enunț:** Calculați vârsta medie a participanților pentru fiecare grupă de vârstă și numărul acestora, rotunjind vârsta medie la 0 zecimale.

db.participant.aggregate([

{

$group: {

\_id: "$grupa\_gvd",

numar\_participanti: { $sum: 1 },

varsta\_medie: {

$avg: {

$divide: [

{ $subtract: [new Date(), "$data\_nasterii"] },

31536000000

]

}

}

}

},

{

$project: {

\_id: 1,

numar\_participanti: 1,

varsta\_medie: { $round: ["$varsta\_medie", 0] }

}

},

{

$sort: { numar\_participanti: -1 }

}

O imagine care conține text, captură de ecran, Font

Descriere generată automat]);