Universitatea Tehnică a Moldovei

Facultatea Calculatoare Informatică și Microelectronică  
Departamentul Ingineria Software și Automatică

**RAPORT**

**Lucrarea de laborator nr. 9**

**la****Baze de date**

A realizat: st. gr. SI-211 Chihai Adrian

A verificat: Rodica Bulai

**Chișinău – 2023**

**Tema: Transact-sql – instrucțiuni procedurale**

**Noțiuni teoretice:**

SQL standardizat (ANSI SQL) este un limbaj de interogare pentru sistemele de gestiune a bazelor de date obiect-relationale și nu are puterea unui limbaj de programare. Nu permite utilizarea de variabile, structuri de control al fluxului, bucle și alte elemente caracteristice programarii. Deci, nu e de mirare ca SQL este un limbaj de interogiri, nu un limbaj de programare.

Astfel, SQL este instrumentul ideal pentru a lucra cu bazele de date. SQL este un limbaj neprocedural cu format liber, deoarece precizeaza ce rezultate sunt necesare, și nu procedura prin care se obtin acestea.

Atunci cand se dorește realizarea unei aplicatii complete pentru gestionarea unei baze de date relationale, este necesar sa se utilizeze un instrument, care ar sustine atat capacitatea de interogare SQL, cat și calitatile limbajelor de programare traditionale. Transact-SQL este limbajul de programare pe care Microsoft SQL Server ii ofera pentru a extinde standardul SQL cu alte tipuri de instructiuni și elemente proprii limbajelor de programare.

*Variabile globale ale serverului:*

* @@ERROR Contine numarul celei mai recente erori Transact-SQL.
* @@IDENTITY Contine valoarea campului IDENTITY al ultimei inregistrari inserate.
* @@VERSION Contine informatii in legatura cu sistemul și compilarea curenta a serverului instalat.
* @@VERSION Contine informatii in legatura cu sistemul și compilarea curenta a serverului instalat.
* @@VERSION Contine informatii in legatura cu sistemul și compilarea curenta a serverului instalat.
* @@MAX CONNECTIONS Returneaza numarul maxim de conexiuni permise.

*Structuri de control ale fluxului:*

Expresia CASE are urmatoarea sintaxa generala:

CASE <expresie\_intrare>

WHEN <expresie\_comparare> THEN <expresie\_rezultate>

[WHEN <expresie\_comparare> THEN <expresie\_rezultate> ... ]

[ELSE <expresie\_rezultate\_alternativ>]

END

*Structura alternativă IF ...ELSE:*

IF <expresie\_boolean>

{ <instructiune> | <bloc\_instructiuni> }

[ ELSE

{ <instructiune> | <bloc\_instructiuni> } ]

*Structura repetitivă WHILE:*

WHILE <expresie\_boolean>

{ <instructiune> | <bloc\_instructiuni> | BREAK | CONTINUE }

* **BREAK** Se realizeaza ieșirea forțata din WHILE Toate instructiunile situate dupa cuvantul-cheie END, care marcheaza sfârșitul buclei WHILE, sunt executate.
* **CONTINUE** Cauzeaza repornirea buclei WHILE, ignorand toate instructiunile care urmeaza dupa cuvantul CONTINUE.
* **RETURN** Oprirea forțată a unui proces și returnarea unui set de valori.

Functia **COALESCE** returneaza prima expresie nenula din lista de expresii ce constituie parametrii acestei functii. Sintaxa functiei este urmatoarea: COALESCE ( , ... ) unde se returneaza , daca aceasta este nenula, se returneaza ill cazul In care prima expresie este nula, dar a doua e nenula. Expresia este returnata in cazul in care aceasta este nenula, dar toate expresiile precedente sunt nule. Toate expresiile trebuie sa fie de acelși tip. Functia data permite inlocuirea valorii NULL cu o alta valoare.

Functia **ISNULL** din Transact-SQL este folositii pentru convertirea unei expresii nule in altii valoare.

Dimpotriva, functia **NULLIF**, valabila m Transact-SQL, compara doua expresii. Daca acestea sunt egale, functia returneaza valoarea NULL. Daca nu sunt egale, functia returneaza prima expresie. Nu este posibila specificarea literalului NULL pentru prima expresie.

**Sarcini practice:**

1. Completati urmatorul cod pentru a afișa eel mai mare numar dintre cele trei numere prezentate.

DO $$

DECLARE

n1 INT;

n2 INT;

n3 INT;

mai\_mare INT;

BEGIN

n1 := floor(random() \* 60);

n2 := floor(random() \* 60);

n3 := floor(random() \* 60);

IF n1 >= n2 AND n1 >= n3 THEN

mai\_mare := n1;

ELSIF n2 >= n1 AND n2 >= n3 THEN

mai\_mare := n2;

ELSE

mai\_mare := n3;

END IF;

RAISE NOTICE 'n1 = %', n1;

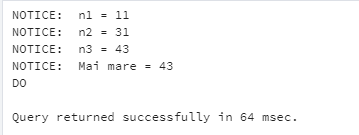
RAISE NOTICE 'n2 = %', n2;

RAISE NOTICE 'n3 = %', n3;

RAISE NOTICE 'Mai mare = %', mai\_mare;

END $$;

Rezultat:



1. Afișati primele zece date (numele, prenumele studentului) in functie de valoarea notei (cu exceptia notelor 6 și 8) a studentului la primul test al disciplinei Baze de date , folosind structura de altemativa IF. .. ELSE. Sa se foloseasca variabilele.

DO $$

DECLARE

nume VARCHAR(20);

prenume VARCHAR(20);

nota INT;

count INT := 0;

index INT := 0;

BEGIN

WHILE index < 10 LOOP

SELECT s.nume\_student, s.prenume\_student, sr.nota

INTO nume, prenume, nota

FROM studenti\_reusita sr

JOIN discipline d ON d.id\_disciplina = sr.id\_disciplina

JOIN studenti s ON s.id\_student = sr.id\_student

WHERE sr.tip\_evaluare = 'Testul 1' AND s.id\_student = count + 100;

IF nota <> 6 AND nota <> 8 THEN

RAISE NOTICE '% %', nume, prenume;

index := index + 1;

END IF;

count := count + 1;

END LOOP;

END $$;

Rezultat:

1. Rezolvati aceeși sarcina, 1, apeland la structura selectiva CASE.

DO $$

DECLARE

n1 INT;

n2 INT;

n3 INT;

mai\_mare INT;

BEGIN

n1 := floor(random() \* 60);

n2 := floor(random() \* 60);

n3 := floor(random() \* 60);

mai\_mare :=

CASE

WHEN n1 >= n2 AND n1 >= n3 THEN n1

WHEN n2 >= n1 AND n2 >= n3 THEN n2

WHEN n3 >= n1 AND n3 >= n2 THEN n3

END;

RAISE NOTICE 'n1 = %', n1;

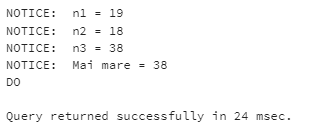
RAISE NOTICE 'n2 = %', n2;

RAISE NOTICE 'n3 = %', n3;

RAISE NOTICE 'Mai mare = %', mai\_mare;

END $$;

Rezultat:



1. Modificati exercitiile din sarcinile 1 și 2 pentru a include procesarea erorilor cu TRY și CATCH, și RAISERRROR.

DO $$

DECLARE

n1 INT := 60;

n2 INT := 60;

n3 INT := 60;

mai\_mare INT := 0;

BEGIN

mai\_mare :=

CASE

WHEN n1 > n2 AND n1 > n3 THEN n1

WHEN n2 > n1 AND n2 > n3 THEN n2

WHEN n3 > n1 AND n3 > n2 THEN n3

ELSE 0

END;

IF mai\_mare = 0 THEN

RAISE EXCEPTION 'Toate numerele sunt egale';

END IF;

RAISE NOTICE 'n1 = %', n1;

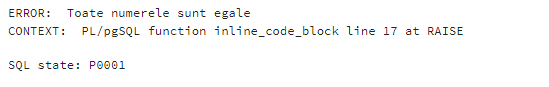
RAISE NOTICE 'n2 = %', n2;

RAISE NOTICE 'n3 = %', n3;

RAISE NOTICE 'Mai mare = %', mai\_mare;

END $$;

Rezultat:



**Concluzie:**

În această lucrare de laborator am aplicat cunoștințele despre scrierea și executarea interogărilor în scrierea instrucțiunilor procedurale în Postgresql. Avantajul folosirii acestui limbaj este nu doar de a declara variabile și a stoca anumite date, dar și posibilitatea controlului fluxului de execuție al unui proces.