Tutoriat 5

DECODE; CASE; DIVISION;

DECODE si CASE

DECODE si CASE reprezinta o functie, repsectiv o structura cu utilitati asemenatoare. DECODE verifica cu ce ramura este egala o expreise data si intoarce o valoare pentru prima ramura cu care este egala expresia. CASE functioneaza asemanator instructiunii „switch” din C++ sau Python. Ea primeste tot o expresie insa, diferit fata de CASE, in loc sa verifice numai egalitatea pentru a intra pe o ramura, aceasta poate verifica valoarea de adevar a unei secvente de cod. Sintaxa este :

```
1.  DECODE(expr, val1, output1, val2, output2, ..., valN, outputN, default);
2.
3.  -- pentru egalitatea cu valori (echivalent cu DECODE)
4.  CASE expr WHEN val1 THEN output1
5.            WHEN val2 THEN output2
6.            ...
7.            WHEN valN THEN outputN
8.            ELSE default END;
9.
10. --pentru valoarea de adevar a unei expresii
11. CASE WHEN expr1 THEN output1
12.      WHEN expr2 THEN output2
13.      ...
14.      WHEN exprN THEN outputN
15.      ELSE default END;
```

Putem observa ca DECODE primeste ca parametrii expresia ce trebuie egalata si apoi perechi de forma: valoare, output intors daca expresia este egala cu valoarea. La final functia primeste valoarea default ce este intoarsa in cazul in care expresia nu este egala cu nici o valoare.

Pentru instructiunea CASE exista 2 cazuri: primul caz, ce este echivalent cu DECODE, si al doilea in care, in loc sa primeasca o expresie si valori cu care trebuie egalata, primeste mai multe expresii si intoarce output-ul primei expresii adevarate. Daca nici-o expresie nu este adevarata atunci intoarce valoarea default.

Pentru a exemplifica DECODE si primul caz de CASE o sa afisam din tabelul EMPLOYEES toti angajatii ce lucreaza ca si: „IT_PROG” drept ”Programator”, „SA_REP” drept „Reprezentant”, „SA_MAN” drept „Manager” , altfel „Angajat”.

```sql
SELECT first_name, DECODE(job_id, 'IT_PROG', 'Programator', 'SA_REP', 'Repreze
ntant',  'SA_MAN', 'Manager', 'Angajat')
FROM employees;

SELECT first_name, CASE job_id WHEN 'IT_PROG' THEN 'Programator'
                               WHEN 'SA_REP' THEN 'Reprezentant'
                               WHEN 'SA_MAN' THEN 'Managet'
                               ELSE 'Angajat' END
FROM employees;
```

Ambele interogari sunt echivalente. In ambele cazuri am verificat numele job-ului si am afisat denumirea ceruta.

LAST_NAME	DECODE(...)
Donald	Angajat
Douglas	Angajat
Alexander	Programator
Peter	Reprezentant
John	Manager
...	...

In continuare, pentru cazul 2 din CASE sa afisam toti angajatii cu salariile impartite astfel: daca angajatul castiga mai putin de 5000 sa afisam „salariu mic”, daca castiga intre 5000 si 10000 sa afisam „salariu mediu” si daca castiga mai mult de 10000 sa afisam „salariu mare”:

```sql
SELECT first_name, CASE WHEN salary < 5000 THEN 'salariu mic'
                        WHEN salary < 10000 THEN 'salariu mediu'
                        ELSE 'salariu mare' END
FROM employees;
```

Pentru acest exercitiu am verificat daca angajatul are salariu mic sau mediu, toti angajatii care nu indelplinesc cerintele fiind automat consideraiti angajati cu salariu mare .

LAST_NAME	CASE...
Donald	salariu mic
Douglas	salariu mic
Alexander	salariu mediu
Peter	salariu mare
John	salariu mediu
...	...

DIVISION

Division reprezinta un operator binar ce are ca scop sa verificarea ca, daca un element de un tip este in relatie cu mai multe elemnte de alt tip, in toate aceste relatii sa se regaseasca acelasi atribut. In alte cuvinte daca o linie dintr-un tabel este legata de mai multe linii din alt tabel, toate aceste linii de care este legata trebuie sa indeplinesca o conditie stabilita.

Un exemplu simplu ar fi relatia ce exista intre proiecte si angajati. Fiecare angajat este legat de mai multe proiecte. O operatie de divison pe aceasta este sa afisam toti angajatii ce lucreaza la toate proeictele cu buget de 10000.

Pentru rezolvarea acestei probleme exista 3 metode prin care putem simula division in SQL:

Metoda 1: Folosind dubla negatie

```
1.  SELECT DISTINCT employee_id
2.  FROM works_on a
3.  WHERE NOT EXISTS
4.        (SELECT 1
5.         FROM project p
6.         WHERE budget=10000
7.         AND NOT EXISTS
8.               (SELECT 'x'
9.                FROM works_on b
10.               WHERE p.project_id=b.project_id
11.               AND b.employee_id=a.employee_id));
```

Pentru aceasta metoda voi verifica pentru fiecare angajat ca nu exista nici-un proiect cu buget de 10000 la care el nu lucreaza, daca lista proiectelor de buget 10000 la care el nu lucreaza este vida atunci il voi afisa. Pentru aceasta se foloseste operatorul NOT EXISTS, in prima subcerere, pentru a extrage toate proiectele cu buget de 10000 la care angajatul NU lucreaza (astfel avand un tabel ce are raspunsul cerintei negate) si apoi, in cererea principala, pentru a verifica ca aceasta lista este vida (neg negarea si astfel ajung la raspunsul corect).

Metoda 2: Folosind COUNT

```
1. SELECT employee_id
2. FROM works_on
3. WHERE project_id IN (SELECT project_id
4.                      FROM project
5.                      WHERE budget=10000)
6. GROUP BY employee_id
7. HAVING COUNT(project_id)= (SELECT COUNT(*)
8.                            FROM project
9.                            WHERE budget=10000);
```

Pentru aceasta metoda voi alege toti angazatii ce lucreaza la macar un proiect cu buget de 10000 (liniile 3-5), voi grupa tabelul dupa acesti angajati si apoi voi numara la cate proiecte de buget 10000 lucreaza fiecare. In final(liniile 7-9) numar toate proectele cu buget 10000 si verific daca angajatul lucreaza la toate, adica daca numarul de proiecte cu buget mare la care lucreaza este egal cu numarul de proeicte cu buget mare total.

Metoda 3: Folosind operatii pe multimi (operatorul MINUS)

```
1. SELECT employee_id
2. FROM works_on
3.
4. MINUS
5.
6. SELECT employee_id
7. FROM (SELECT employee_id, project_id
8.       FROM (SELECT DISTINCT employee_id FROM works_on) t1,
9.            (SELECT project_id FROM project WHERE budget=10000) t2
10.      MINUS
11.      SELECT employee_id, project_id FROM works_on) t3;
```

Pentru acesta metoda selectez toate combinatiile posibile dintre toti angajatii(tabelul t1) si toate proiectele cu buget 10000(tabelul t2) pe liniile 7-8. Din acest tabel elimin toate combinatiile de angajati si proiecte ce exista deja in tabel(MINUS-ul de la linia 10). Dupa aceasta, in tabel se afla toti angajatii ce nu lucreaza la proiecte cu buget mare. Selectez toti angajatii intr-un alt tabel (linia 1) si scad tabelul creat anterior. Rezultatul este un tabel cu angajatii din care s-au eliminat toti angajatii ce au proiecte cu buget mare la care nu lucreaza.

Exercitii se pot gasi pe Drive -> BD 2018-2019 -> Laborator5-> Exercitiile 26 – 23

-> Laborator6-> Exercitiile 1 - 14