

Funcții SQL. Cereri multi-relație (introducere)

I. [Funcții SQL]

- Funcțiile SQL sunt predefinite în sistemul *Oracle* și pot fi utilizate în instrucțiuni SQL. Ele nu trebuie confundate cu funcțiile definite de utilizator, scrise în *PL/SQL*.
- Dacă o funcție SQL este apelată cu un argument având un alt tip de date decât cel așteptat, sistemul convertește implicit argumentul înainte să evalueze funcția.
- De obicei, dacă o funcție SQL este apelată cu un argument *null*, ea returnează valoarea *null*. Funcțiile care nu urmează această regulă sunt *CONCAT*, *NVL* și *REPLACE*.

Funcțiile SQL pot fi clasificate în următoarele categorii:

- **Funcții *single-row***;
- **Funcții *multiple-row*** (funcții agregat);

1. Funcțiile *single row* returnează câte o singură linie rezultat pentru fiecare linie a tabelului sau vizualizării interogate. Aceste funcții pot apărea în:

- listele de expresii din clauza **SELECT**
- clauzele **WHERE**, **START WITH**, **CONNECT BY** și **HAVING**.

În ceea ce privește tipul argumentelor asupra cărora operează și al rezultatelor furnizate, funcțiile *single row* pot fi clasificate în categorii corespunzătoare.

- ❑ **Funcțiile de conversie** cele mai importante sunt:

Funcție	Descriere	Exemplu conversie
TO_CHAR	convertește (sau formatează) un număr sau o dată calendaristică în șir de caractere	<i>TO_CHAR(7) = '7'</i> <i>TO_CHAR(-7) = '-7'</i> <i>TO_CHAR(SYSDATE, 'DD/MM/YYYY') = '02/03/2020'</i>
TO_DATE	convertește (sau formatează) un număr sau un șir de caractere în dată calendaristică	<i>TO_DATE('18-MAR-2019','dd-mon-yyyy')</i>
TO_NUMBER	convertește (sau formatează) un șir de caractere în număr	<i>TO_NUMBER(' -25.789', '99,999') = -25.789</i>

Obs: Există două tipuri de conversii:

- **implicite**, realizate de sistem atunci când este necesar;
- **explicite**, indicate de utilizator prin intermediul funcțiilor de conversie.

Conversiile **implicite** asigurate de server-ul *Oracle* sunt:

- de la *VARCHAR2* sau *CHAR* la *NUMBER*;
- de la *VARCHAR2* sau *CHAR* la *DATE*;
- de la *NUMBER* la *VARCHAR2* sau *CHAR*;
- de la *DATE* la *VARCHAR2* sau *CHAR*.

□ Dintre **funcțiile pentru prelucrarea caracterelor** amintim:

Funcție	Descriere	Exemplu
<i>LENGTH(string)</i>	întoarce lungimea șirului de caractere <i>string</i>	<i>LENGTH('Informatica')=11</i>
<i>SUBSTR(string, start [,n])</i>	întoarce subșirul lui <i>string</i> care începe pe poziția <i>start</i> și are lungimea <i>n</i> ; dacă <i>n</i> nu este specificat, subșirul se termină la sfârșitul lui <i>string</i> ;	<i>SUBSTR('Informatica', 1, 4) = 'Info'</i> <i>SUBSTR('Informatica',6) = 'matica'</i> <i>SUBSTR('Informatica', -5) = 'matica'</i> (ultimele 5 caractere)
<i>LTRIM(string [, 'chars'])</i>	șterge din stânga șirului <i>string</i> orice caracter care apare în <i>chars</i> , până la găsirea primului caracter care nu este în <i>chars</i> ; în cazul în care <i>chars</i> nu este specificat, se șterg spațiile libere din stânga lui <i>string</i> ;	<i>LTRIM (' info') = 'info'</i>
<i>RTRIM(string [, 'chars'])</i>	este similar funcției <i>LTRIM</i> , cu excepția faptului că ștergerea se face la dreapta șirului de caractere;	<i>RTRIM ('infoXXXX', 'X') = 'info'</i>
<i>TRIM (LEADING TRAILING BOTH chars FROM expresie)</i>	elimină caracterele specificate (<i>chars</i>) de la începutul (<i>leading</i>) , sfârșitul (<i>trailing</i>) sau din ambele părți, dintr-o expresie caracter dată.	<i>TRIM (LEADING 'X' FROM 'XXXInfoXXX') = 'InfoXXX'</i> <i>TRIM (TRAILING 'X' FROM 'XXXInfoXXX') = 'XXXInfo'</i> <i>TRIM (BOTH 'X' FROM 'XXXInfoXXX') = 'Info'</i> <i>TRIM (BOTH FROM ' Info ') = 'Info'</i>

LPAD (string, length [, 'chars'])	adaugă <i>chars</i> la stânga șirului de caractere <i>string</i> până când lungimea noului șir devine <i>length</i> ; în cazul în care <i>chars</i> nu este specificat, atunci se adaugă spații libere la stânga lui <i>string</i> ;	LPAD (LOWER('iNfO'),6) = ' info'
RPAD (string, length [, 'chars'])	este similar funcției LPAD , dar adăugarea de caractere se face la dreapta șirului;	RPAD (LOWER('iNfO'), 6, 'X') = 'infoXX'
REPLACE (string1, string2 [, string3])	întoarce <i>string1</i> cu toate aparițiile lui <i>string2</i> înlocuite prin <i>string3</i> ; dacă <i>string3</i> nu este specificat, atunci toate aparițiile lui <i>string2</i> sunt șterse;	REPLACE ('\$b\$bb','\$','a') = 'ababb' REPLACE ('\$b\$bb','\$b','ad') = 'adadb' REPLACE ('\$a\$aa','\$') = 'aaa'
UPPER (string), LOWER (string)	transformă toate literele șirului de caractere <i>string</i> în majuscule, respectiv minuscule;	LOWER ('iNfO') = 'info' UPPER ('iNfO') = 'INFO'
INITCAP (string)	transformă primul caracter al șirului în majusculă, restul caracterelor fiind transformate în minuscule;	INITCAP ('iNfO') = 'Info'
INSTR (string, 'chars' [, start [, n]])	caută în <i>string</i> , începând de la poziția <i>start</i> , a <i>n</i> -a apariție a secvenței <i>chars</i> și întoarce poziția respectivă; dacă <i>start</i> nu este specificat, căutarea se face de la începutul șirului; dacă <i>n</i> nu este specificat, se caută prima apariție a secvenței <i>chars</i> ;	INSTR (LOWER('AbC aBcDe'), 'ab', 5, 2) = 0 INSTR (LOWER('AbCdE aBcDe'), 'ab', 5) = 7
ASCII (char)	furnizează codul ASCII al primului caracter al unui șir	ASCII ('alfa') = ASCII ('a') = 97
CHR (num)	întoarce caracterul corespunzător codului ASCII specificat	CHR (97)= 'a'
CONCAT (string1, string2)	realizează concatenarea a două șiruri de caractere	CONCAT ('In', 'fo') = 'Info'
TRANSLATE (string, source, destination)	fiecare caracter care apare în șirurile de caractere <i>string</i> și <i>source</i> este transformat în caracterul corespunzător (aflat pe aceeași poziție ca și în <i>source</i>) din șirul de caractere <i>destination</i>	TRANSLATE ('\$a\$aa','\$','b') = 'babaa' TRANSLATE ('\$a\$aaa','\$a','bc') = 'bcbccc'

Obs: Testarea funcțiilor prezentate se face astfel :

SELECT apel_funcție FROM dual;

Astfel că vom omite comanda *SELECT* și vom da numai apelul funcției și rezultatul returnat.

❑ **Funcțiile aritmetice** *single-row* pot opera asupra:

- unei singure valori, și aceste funcții sunt: *ABS* (valoarea absolută), *CEIL* (partea întreagă superioară), *FLOOR* (partea întreagă inferioară), *ROUND* (rotunjire cu un număr specificat de zecimale), *TRUNC* (trunchiere cu un număr specificat de zecimale), *EXP* (ridicarea la putere a lui *e*), *LN* (logaritm natural), *LOG* (logaritm într-o bază specificată), *MOD* (restul împărțirii a două numere specificate), *POWER* (ridicarea la putere), *SIGN* (semnul unui număr), *COS* (cosinus), *COSH* (cosinus hiperbolic), *SIN* (sinus), *SINH* (sinus hiperbolic), *SQRT* (rădăcina pătrată), *TAN* (tangent), *TANH* (tangent hiperbolic);
- unei liste de valori, iar acestea sunt funcțiile *LEAST* și *GREATEST*, care întorc cea mai mică, respectiv cea mai mare valoare a unei liste de expresii.

❑ **Funcțiile pentru prelucrarea datelor calendaristice sunt:**

Funcție	Descriere	Exemplu
<i>SYSDATE</i>	Întoarce data și timpul curent	<i>SELECT SYSDATE FROM dual;</i> (de revăzut utilizarea acestei funcții împreună cu <i>TO_CHAR</i> în cadrul laboratorului 1)
<i>ADD_MONTHS(expr_date, nr_luni)</i>	Întoarce data care este după <i>nr_luni</i> luni de la data <i>expr_date</i> ;	<i>ADD_MONTHS('06-MAR-2020', 3) = '06-JUN-2020'.</i>
<i>NEXT_DAY(expr_date, day)</i>	Întoarce următoarea dată după data <i>expr_date</i> , a cărei zi a săptămânii este cea specificată prin șirul de caractere <i>day</i>	<i>NEXT_DAY('02-MAR-2020', 'Monday') = '09-MAR-2020'</i>
<i>LAST_DAY(expr_date)</i>	Întoarce data corespunzătoare ultimei zile a lunii din care data <i>expr_date</i> face parte	<i>LAST_DAY('02-MAR-2020') = '31-MAR-2020'</i>
<i>MONTHS_BETWEEN(expr_date2, expr_date1)</i>	Întoarce numărul de luni dintre cele două date calendaristice specificate. Data cea mai recentă trebuie specificată în primul argument, altfel rezultatul este negativ.	<i>MONTHS_BETWEEN('06-MAR-2017', '10-OCT-2015') = 16.8709677</i> <i>MONTHS_BETWEEN('10-OCT-2015', '06-MAR-2017') = -16.8709677</i>

<i>TRUNC(expr_date)</i>	întoarce data <i>expr_date</i> , dar cu timpul setat la ora 12:00 AM (miezul nopții);	<i>TO_CHAR(TRUNC(SYSDATE), 'dd/mm/yy HH24:MI') = '02/03/20 00:00'</i>
<i>ROUND(expr_date)</i>	dacă data <i>expr_date</i> este înainte de miezul zilei, întoarce data <i>d</i> cu timpul setat la ora 12:00 AM; altfel, este returnată data corespunzătoare zilei următoare, cu timpul setat la ora 12:00 AM;	<i>TO_CHAR(ROUND(SYSDATE), 'dd/mm/yy hh24:mi am') = '02/03/20 00:00 AM'</i>
<i>LEAST(d1, d2, ..., dn), GREATEST(d1, d2, ..., dn)</i>	dintr-o listă de date calendaristice, funcțiile întorc prima, respectiv ultima dată în ordine cronologică	<i>LEAST(SYSDATE, SYSDATE + 3, SYSDATE - 5) = SYSDATE-5</i> <i>GREATEST(SYSDATE, SYSDATE + 3, SYSDATE - 5) = SYSDATE + 3</i>

Operațiile care se pot efectua asupra datelor calendaristice sunt următoarele:

Operație	Tipul de date al rezultatului	Descriere
<i>expr_date</i> +/- <i>expr_number</i>	<i>Date</i>	Scade/adună un număr de zile dintr-o / la o dată. Numărul de zile poate sa nu fie întreg (putem adăuga, de exemplu, un număr de minute sau de ore).
<i>expr_date1</i> – <i>expr_date2</i>	<i>Number</i>	Întoarce numărul de zile dintre două date calendaristice. Data <i>expr_date1</i> trebuie să fie mai recentă decât <i>expr_date2</i> , altfel rezultatul este negativ.

❑ **Funcții diverse:**

Funcție	Descriere	Exemplu
DECODE (value, if1, then1, if2, then2, ... , ifN, thenN, else)	returnează <i>then1</i> dacă <i>value</i> este egală cu <i>if1</i> , <i>then2</i> dacă <i>value</i> este egală cu <i>if2</i> etc.; dacă <i>value</i> nu este egală cu nici una din valorile <i>if</i> , atunci funcția întoarce valoarea <i>else</i> ;	<i>DECODE</i> ('a', 'a', 'b', 'c') = 'b' -> (dacă primul parametru 'a' este egal cu al doilea returnează 'b', altfel ret 'c') <i>DECODE</i> ('b', 'a', 'b', 'c') = 'c' <i>DECODE</i> ('c', 'a', 'b', 'c') = 'c'
NVL (expr_1, expr_2)	dacă <i>expr_1</i> este <i>NULL</i> , întoarce <i>expr_2</i> ; altfel, întoarce <i>expr_1</i> . Tipurile celor două expresii trebuie să fie compatibile sau <i>expr_2</i> să poată fi convertit implicit la <i>expr_1</i>	<i>NVL</i> (<i>NULL</i> , 1) = 1 <i>NVL</i> (2, 1) = 2 <i>NVL</i> ('a', 1) = a -- conversie implicită <i>NVL</i> (1, 'a') -- ??
NVL2 (expr_1, expr_2, expr_3)	dacă <i>expr_1</i> este <i>NOT NULL</i> , întoarce <i>expr_2</i> , altfel întoarce <i>expr_3</i>	<i>NVL2</i> (1, 2, 3) = 2 <i>NVL2</i> (<i>NULL</i> , 1, 2) = 2
NULLIF (expr_1, expr_2)	Dacă <i>expr_1</i> = <i>expr_2</i> atunci funcția returnează <i>NULL</i> , altfel returnează expresia <i>expr_1</i> . Echivalent cu <i>CASE WHEN expr1 = expr2 THEN NULL ELSE expr1 END</i> ;	<i>NULLIF</i> (1, 2) = 1 <i>NULLIF</i> (1, 1) = <i>NULL</i>
COALESCE (expr_1, expr_2, ... , expr_n)	Returnează prima expresie <i>NOT NULL</i> din lista de argumente;	<i>COALESCE</i> (<i>NULL</i> , <i>NULL</i> , 1, 2, <i>NULL</i>) = 1
UID, USER	întorc <i>ID</i> -ul, respectiv <i>username</i> -ul utilizatorului <i>ORACLE</i> curent;	<i>SELECT USER</i> <i>FROM dual</i> ;
VSIZE (expr)	întoarce numărul de octeți ai unei expresii de tip <i>DATE</i> , <i>NUMBER</i> sau <i>VARCHAR2</i> ;	<i>SELECT VSIZE(salary)</i> <i>FROM employees</i> <i>WHERE employee_id=200</i> ;

Utilizarea funcției **DECODE** este echivalentă cu utilizarea clauzei **CASE** (într-o comandă SQL). O formă a acestei clauze este:

<pre> CASE <i>expr</i> WHEN <i>expr_1</i> THEN <i>valoare_1</i> [WHEN <i>expr_2</i> THEN <i>valoare_2</i> ... WHEN <i>expr_n</i> THEN <i>valoare_n</i>] [ELSE <i>valoare</i>] END </pre>	<p>În funcție de valoarea expresiei <i>expr</i> returnează <i>valoare_i</i> corespunzătoare primei clauze WHEN .. THEN pentru care <i>expr</i> = <i>expresie_i</i>; dacă nu corespunde cu nici o clauză WHEN atunci returnează valoarea din ELSE. Nu se poate specifica NULL pentru toate valorile de returnat. Toate valorile trebuie să aibă același tip de date.</p>
--	---

2. Funcțiile multiple-row (agregat) pot fi utilizate pentru a returna informația corespunzătoare fiecăruia dinte grupurile obținute în urma divizării liniilor tabelului cu ajutorul clauzei GROUP BY. Ele pot apărea în clauzele **SELECT**, **ORDER BY** și **HAVING**. Server-ul *Oracle* aplică aceste funcții fiecărui grup de linii și returnează un singur rezultat pentru fiecare mulțime.

Dintre funcțiile grup definite în sistemul *Oracle*, se pot enumera: **AVG**, **SUM**, **MAX**, **MIN**, **COUNT**, **STDDEV**, **VARIANCE** etc. Tipurile de date ale argumentelor funcțiilor grup pot fi **CHAR**, **VARCHAR2**, **NUMBER** sau **DATE**. Funcțiile **AVG**, **SUM**, **STDDEV** și **VARIANCE** operează numai asupra valorilor numerice. Funcțiile **MAX** și **MIN** pot opera asupra valorilor numerice, caracter sau dată calendaristică.

Toate funcțiile grup, cu excepția lui COUNT(*), ignoră valorile null. **COUNT(expresie)** returnează numărul de linii pentru care expresia dată nu are valoarea *null*. Funcția **COUNT** returnează un număr mai mare sau egal cu zero și nu întoarce niciodată valoarea *null*.

Când este utilizată clauza **GROUP BY**, server-ul sortează implicit mulțimea rezultată în ordinea crescătoare a valorilor coloanelor după care se realizează gruparea.

III. [Exerciții]

[Funcții pe șiruri de caractere]

1. Scrieți o cerere care are următorul rezultat pentru fiecare angajat:

<prenume angajat> <nume angajat> castiga <salariu> lunar dar doreste <salariu de 3 ori mai mare>. Etichetați coloana **“Salariu ideal”**. Pentru concatenare, utilizați atât funcția **CONCAT** cât și operatorul **“||”**.

```

SELECT CONCAT(STR1, STR2) || ' castiga ' || salary || ... "Salariu ideal"
FROM employees;

```


2. Scrieți o cerere prin care să se afișeze **prenumele salariatului** cu **prima litera majusculă** și **toate celelalte litere minuscule**, **numele acestuia cu majuscule și lungimea numelui**, pentru angajații al căror nume începe cu J sau M sau care au a treia literă din nume A. Rezultatul va fi ordonat descrescător după lungimea numelui. Se vor eticheta coloanele corespunzător. Se cer 2 soluții (cu operatorul *LIKE* și funcția *SUBSTR*).
3. Să se afișeze, pentru angajații cu prenumele „Steven”, **codul** și **numele** acestora, precum și **codul departamentului** în care lucrează. Căutarea trebuie să **nu** fie *case-sensitive*, iar eventualele *blank*-uri care preced sau urmează numelui trebuie ignorate.

```
SELECT ...  
FROM ...  
WHERE LTRIM(RTRIM(UPPER(first_name)))='STEVEN';
```

sau

```
SELECT ...  
FROM ...  
WHERE TRIM(BOTH FROM UPPER(first_name))='STEVEN';
```

4. Să se afișeze pentru toți angajații al căror nume se termină cu litera 'e', codul, numele, lungimea numelui și poziția din nume în care apare prima data litera 'A'. Utilizați *alias*-uri corespunzătoare pentru coloane.

[Funcții aritmetice]

5. Să se afișeze detalii despre salariații care au lucrat un **număr întreg de săptămâni** până la data curentă.

Obs: Soluția necesită rotunjirea diferenței celor două date calendaristice.

6. Să se afișeze **codul salariatului**, **numele**, **salariul**, **salariul mărit cu 15%**, exprimat cu două zecimale și **numărul de sute** al salariului nou rotunjit la 2 zecimale. Etichetați ultimele două coloane „Salariu nou”, respectiv „Numar sute”. Se vor lua în considerare salariații al căror salariu **nu este divizibil** cu 1000.
7. Să se listeze **numele** și **data angajării** salariaților **care câștigă comision**. Să se eticheteze coloanele „Nume angajat”, „Data angajării”. Utilizați funcția **RPAD** pentru a determina ca data angajării să aibă lungimea de 20 de caractere.

```
SELECT last_name, ____  
FROM employees  
WHERE ____;
```


[Funcții și operații cu date calendaristice]

8. Să se afișeze **data** (numele lunii, ziua, anul, ora, minutul și secunda) **de peste 30 zile**.

```
SELECT TO_CHAR(SYSDATE+30, 'MONTH DD YYYY HH24:MI:SS') "Data"  
FROM DUAL;
```

9. Să se afișeze **numărul de zile** rămase până la sfârșitul anului.

10. a) Să se afișeze **data** de peste 12 ore.

```
SELECT TO_CHAR(SYSDATE+12/24, 'DD/MM HH24:MI:SS') "Data"  
FROM DUAL;
```

b) Să se afișeze data de peste 5 minute

Obs: Cât reprezintă 5 minute dintr-o zi?

11. Să se afișeze **numele** și **prenumele** angajatului (într-o singură coloană), **data angajării** și **data negocierii salariului**, care este prima zi de Luni după 6 luni de serviciu. Etichetați această coloană "Negociere".

```
SELECT concat(last_name, first_name), hire_date,  
NEXT_DAY(ADD_MONTHS(hire_date, 6), 'monday') "Negociere"  
FROM employees;
```

12. Pentru fiecare angajat să se afișeze **numele** și **numărul de luni** de la data angajării. Etichetați coloana "Luni lucrate". Să se ordoneze rezultatul după numărul de luni lucrate. Se va rotunji numărul de luni la cel mai apropiat număr întreg.

```
SELECT last_name, months_between(sysdate, hire_date) "Luni lucrate"  
FROM employees;  
ORDER BY MONTHS_BETWEEN(SYSDATE, hire_date);
```

Sau:

```
ORDER BY "Luni lucrate";
```

Sau:

```
ORDER BY 2;
```

Obs: În clauza **ORDER BY**, precizarea criteriului de ordonare se poate realiza și prin indicarea *alias*-urilor coloanelor sau a pozițiilor acestora în clauza **SELECT**.

[Funcții diverse]

13. Să se afișeze **numele** angajaților și **comisionul**. Dacă un angajat nu câștigă comision, să se scrie "Fara comision". Etichetați coloana "Comision".
14. Să se listeze **numele**, **salariul** și **comisionul** tuturor angajaților al căror venit lunar (salariu + valoare comision) depășește 10 000.

```
SELECT last_name, salary, commission_pct
FROM employees
WHERE _____;
```

[Instrucțiunea CASE, comanda DECODE]

15. Să se afișeze **numele**, **codul funcției**, **salariul** și o coloana care să arate **salariul după mărire**. Se știe că pentru **IT_PROG** are loc o mărire de 10%, pentru **ST_CLERK** 15%, iar pentru **SA_REP** o mărire de 20%. Pentru ceilalți angajați nu se acordă mărire. Să se denumească coloana "Salariu renegociat"

```
SELECT last_name, job_id, salary,
       DECODE(job_id,
              'IT_PROG', salary*1.1,
              ...
              salary) "Salariu renegociat"
FROM employees;
```

Sau

```
SELECT last_name, job_id, salary,
       CASE job_id WHEN 'IT_PROG' THEN salary* 1.1
              ...
              ELSE salary
       END "Salariu renegociat"
FROM employees;
```

II. [Join]

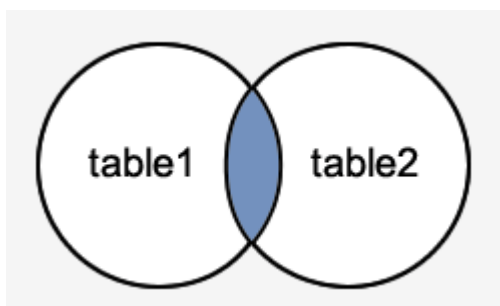
Join-ul este operația de regăsire a datelor din două sau mai multe tabele, pe baza valorilor comune ale unor coloane. De obicei, aceste coloane reprezintă **cheia primară**, respectiv **cheia externă** a tabelelor.

Condiția de *join* se poate scrie în clauza **WHERE** a instrucțiunii **SELECT**. Într-o instrucțiune **SELECT** care **unește tabele prin operația de join**, se recomandă ca numele coloanelor să fie precedate de numele sau alias-urile tabelelor pentru claritate și pentru îmbunătățirea timpului de acces la baza de date. Dacă același nume de coloană apare în mai mult de două tabele, atunci numele coloanei se prefixează **obligatoriu** cu numele sau *alias*-ul tabelului corespunzător. Pentru a realiza un *join* între ***n* tabele**, va fi nevoie de cel puțin ***n* – 1 condiții de join**.

Inner join (equijoin, join simplu) – corespunde situației în care **valorile de pe coloanele ce apar în condiția de join trebuie să fie egale**. (returnează toate randurile din mai multe tabele – din tabelele specificate în condiția de join – în care condiția de join este îndeplinită).

Structură:

```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.column = table2.column;
```



După cum se observă, **INNER JOIN** returnează rândurile (rows) pentru care condiția de join este îndeplinită.

[Exercitii Join]

16. Să se afișeze **codul angajatilor** și **numele departamentului** pentru toți angajații.

I. Condiția de Join este scrisă în clauza WHERE a instrucțiunii SELECT

```
select employee_id, department_name  
from employees e, departments d  
where e.department_id = d.department_id;
```

II. Condiția de Join este scrisă în FROM

Utilizăm ON:

```
select employee_id, department_name  
from employees e join departments d on (e.department_id = d.department_id);
```

Utilizăm USING – atunci când avem coloane cu același nume:

```
select employee_id, department_name  
from employees e join departments d using(department_id);
```

Ce observați având în vedere numărul de rânduri returnate?

Obs: Am realizat operația de join între tabelele **employees** și **departments**, pe baza coloanei comune **department_id**. Observați utilizarea *alias*-urilor. Ce se întâmplă dacă eliminăm condiția de *join*?

Obs: Numele sau *alias*-urile tabelelor sunt obligatorii în dreptul coloanelor care au același nume în mai multe tabele. Altfel, nu sunt necesare dar este recomandată utilizarea lor pentru o mai bună claritate a cererii.

17. Să se listeze **codurile** și **denumirile job-urilor** care există în departamentul 30.

18. Să se afișeze **numele angajatului**, **numele departamentului** și **id-ul locației** pentru toți angajații care câștigă comision.

```
SELECT _____, _____, _____  
FROM _____, _____  
WHERE _____ AND commission_pct _____ ;
```

19. Să se afișeze **numele**, **titlul job-ului** și **denumirea departamentului** pentru toți angajații care lucrează în Oxford.

20. Să se afișeze **codul angajatului** și **numele** acestuia, împreună cu **numele** și **codul șefului** său direct. Se vor eticheta coloanele Ang#, Angajat, Mgr#, Manager.

```
SELECT ang.employee_id Ang#, ang.last_name Angajat, sef.employee_id Mgr#,  
sef.last_name Manager  
FROM employees ang, employees sef  
WHERE ang.manager_id = sef.employee_id;
```

Obs: Am realizat operația de self-join (inner join al tabelului cu el însuși).

21. Să se modifice cererea anterioară pentru a afișa toți salariații, inclusiv cei care nu au șef.

22. Scrieți o cerere care afișează **numele angajatului**, **codul departamentului** în care acesta lucrează și **numele colegilor** săi de departament. Se vor eticheta coloanele corespunzător.

23. Creați o cerere prin care să se afișeze **numele angajaților**, **codul job-ului**, **titlul job-ului**, **numele departamentului** și **salariul** angajaților. Se vor include și angajații al căror departament nu este cunoscut.

24. Să se afișeze **numele** și **data angajării** pentru salariații care au fost angajați după Gates.

```
...  
WHERE e.hire_date > gates.hire_date AND _____;
```

25. Să se afișeze **numele** salariatului și **data angajării** împreună cu **numele** și **data angajării șefului direct** pentru salariații care au fost angajați **înaintea** șefilor lor. Se vor eticheta coloanele Angajat, Data_ang, Manager si Data_mgr.

```
WHERE ang.manager_id = m.employee_id AND ang.hire_date < m.hire_date;
```