Baze de Date
Laborator 2

Funcţii *SQL.* Cereri multi-relaţie (introducere)

I. [Funcţii SQL]

- Funcţiile *SQL* sunt predefinite în sistemul *Oracle* şi pot fi utilizate în instrucţiuni *SQL*. Ele nu trebuie confundate cu funcţiile definite de utilizator, scrise în *PL/SQL*.

- Dacă o funcţie *SQL* este apelată cu un argument având un alt tip de date decât cel aşteptat, sistemul converteşte implicit argumentul înainte să evalueze funcţia.

- De obicei, dacă o funcţie *SQL* este apelată cu un argument *null*, ea returnează valoarea *null*. Funcţiile care nu urmează această regulă sunt *CONCAT*, *NVL* şi *REPLACE*.

Funcţiile *SQL* pot fi clasificate în următoarele categorii:

- Funcţii *single-row;*
- Funcţii *multiple-row* (funcţii agregat);

1. **Funcţiile *single row*** returnează câte o singură linie rezultat pentru fiecare linie a tabelului sau vizualizării interogate. Aceste funcţii pot apărea în:
   - listele de expresii din clauza ***SELECT***
   - clauzele ***WHERE***, *START WITH*, *CONNECT BY* şi ***HAVING***.

   În ceea ce priveşte tipul argumentelor asupra cărora operează şi al rezultatelor furnizate, funcţiile *single row* pot fi clasificate în categorii corespunzătoare.

- **Funcţiile de conversie** cele mai importante sunt:

Funcţie	Descriere	Exemplu conversie
TO_CHAR	converteşte (sau formatează) un număr sau o dată calendaristică în şir de caractere	*TO_CHAR(7) = ' 7'* *TO_CHAR(-7) = '-7'* *TO_CHAR(SYSDATE, 'DD/MM/YYYY') = ' 06/03/2023'*
TO_DATE	converteşte (sau formatează) un număr sau un şir de caractere în dată calendaristică	*TO_DATE('22-FEB-2023','dd-mon-yyyy')*
TO_NUMBER	converteşte (sau formatează) un şir de caractere în număr	*TO_NUMBER ('-25.789', 99.999) = -25.789*

Obs: Există două tipuri de conversii:

- ***implicite***, realizate de sistem atunci când este necesar;
- ***explicite***, indicate de utilizator prin intermediul funcţiilor 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 **funcţiile pentru prelucrarea caracterelor** amintim:

Funcţie	Descriere	Exemplu
LENGTH(string)	întoarce lungimea şirului de caractere *string*	*LENGTH('Informatica')=11*
***SUBSTR**(string, start [,n])*	întoarce subşirul lui *string* care începe pe poziţia *start* şi are lungimea *n*; dacă *n* nu este specificat, subşirul se termină la sfârşitul lui *string*;	*SUBSTR('Informatica', 1, 4) = 'Info'* *SUBSTR('Informatica',6)= 'matica'* *SUBSTR('Informatica', -5) = 'atica'* (ultimele 5 caractere)
LTRIM(string [,'chars'])	şterge din stânga şirului *string* orice caracter care apare în *chars*, până la găsirea primului caracter care nu este în *chars*; în cazul în care *chars* nu este specificat, se şterg spaţiile libere din stânga lui *string*;	*LTRIM (' info') = 'info'*
RTRIM(string [,'chars'])	este similar funcţiei *LTRIM*, cu excepţia faptului că ştergerea se face la dreapta şirului de caractere;	*RTRIM ('infoXXXX', 'X') = 'info'*
TRIM (LEADING | TRAILING | BOTH chars FROM expresie)	elimină caracterele specificate (*chars*) de la începutul (*leading*) , sfârşitul (*trailing*) sau din ambele părţi, dintr-o expresie caracter dată.	*TRIM (LEADING 'X' FROM 'XXXInfoXXX') = 'InfoXXX'* *TRIM (TRAILING 'X' FROM 'XXXInfoXXX') = 'XXXInfo'* *TRIM (BOTH 'X' FROM 'XXXInfoXXX') = 'Info'* *TRIM (BOTH FROM ' Info ') = 'Info'*

LPAD(string, length [,'chars'])	adaugă *chars* la stânga şirului de caractere *string* până când lungimea noului şir devine *length*; în cazul în care *chars* nu este specificat, atunci se adaugă spaţii libere la stânga lui *string*;	*LPAD (LOWER('iNfO'),6) = ' info'*
RPAD(string, length [,'chars'])	este similar funcţiei *LPAD*, dar adăugarea de caractere se face la dreapta şirului;	*RPAD (LOWER('InfO'), 6, 'X') = 'infoXX'*
REPLACE(string1, string2 [,string3])	întoarce *string1* cu toate apariţiile lui *string2* înlocuite prin *string3*; dacă *string3* nu este specificat, atunci toate apariţiile lui *string2* sunt şterse;	*REPLACE ('$b$bb','$','a') = 'ababb'* *REPLACE ('$b$bb','$b','ad') = 'adadb'* *REPLACE ('$a$aa','$') = 'aaa'*
***UPPER**(string),* ***LOWER**(string)*	transformă toate literele şirului de caractere *string* în majuscule, respectiv minuscule;	*LOWER ('InFo') = 'info'* *UPPER ('iNfO') = 'INFO'*
***INITCAP**(string)*	transformă primul caracter al şirului în majusculă, restul caracterelor fiind transformate în minuscule;	*INITCAP ('iNfO') = 'Info'*
INSTR*(string, 'chars' [,start [,n]])*	caută în *string*, începând de de la poziţia *start*, a *n*-a apariţie a secvenţei *chars* şi întoarce poziţia respectivă; dacă *start* nu este specificat, căutarea se face de la începutul şirului; dacă *n* nu este specificat, se caută prima apariţie a secvenţei *chars*;	*INSTR (LOWER('AbC aBcDe'), 'ab', 5, 2) = 0* *INSTR (LOWER('AbCdE aBcDe'), 'ab', 5) = 7*
ASCII(char)	furnizează codul *ASCII* al primului caracter al unui şir	*ASCII ('alfa') = ASCII ('a') = 97*
CHR(num)	întoarce caracterul corespunzător codului *ASCII* specificat	*CHR(97)= 'a'*
***CONCAT**(string1, string2)*	realizează concatenarea a două şiruri de caractere	*CONCAT ('In', 'fo') = 'Info'*
TRANSLATE(string, source, destination)	fiecare caracter care apare în şirurile de caractere *string* şi *source* este transformat în caracterul corespunzător (aflat pe aceeaşi poziţie ca şi în *source*) din şirul de caractere *destination*	*TRANSLATE('$a$aa','$','b') = 'babaa'* *TRANSLATE('$a$aaa','$a','bc') = 'bcbccc'*

Obs: Testarea funcţiilor prezentate se face astfel :

SELECT apel_functie FROM dual;

Astfel că vom omite comanda *SELECT* şi vom da numai apelul funcţiei şi rezultatul returnat.

- **Funcţiile aritmetice** *single-row* pot opera asupra:
  - unei singure valori, şi aceste funcţii 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 împărţirii 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 funcţiile *LEAST* şi *GREATEST*, care întorc cea mai mică, respectiv cea mai mare valoare a unei liste de expresii.

- **Funcţiile pentru prelucrarea datelor calendaristice sunt:**

Funcţie	Descriere	Exemplu
SYSDATE	întoarce data şi timpul curent	*SELECT SYSDATE FROM dual;*
ADD_MONTHS(expr_date, nr_luni)	întoarce data care este după *nr_luni* luni de la data *expr_date*;	*ADD_MONTHS('06-MAR-2023', 3)*= '06-JUN-2023'.
NEXT_DAY(expr_date, day)	întoarce următoarea dată după data *expr_date*, a cărei zi a săptămânii este cea specificată prin şirul de caractere *day*	*NEXT_DAY('07-MAR-2023', 'Monday')= '13-MAR-2023'*
LAST_DAY(expr_date)	întoarce data corespunzătoare ultimei zile a lunii din care data *expr_date* face parte	*LAST_DAY('07-MAR-2023') = '31-MAR-2023'*
MONTHS_BETWEEN(expr_date2, expr_date1)	întoarce numărul de luni dintre cele două date calendaristice specificate. Data cea mai recentă trebuie specificată în primul argument, altfel rezultatul este negativ.	*MONTHS_BETWEEN('10-FEB-2020', '07-MAR-2023')= --36.90322* *MONTHS_BETWEEN('07-MAR-2023', '10-FEB-2020')= -36.90322*

TRUNC(expr_date)	întoarce data *expr_date*, dar cu timpul setat la ora 12:00 AM (miezul nopţii);	*TO_CHAR (TRUNC(SYSDATE), 'dd/mm/yy HH24:MI') = '07/03/23 00:00'* *TRUNC (22.2) = 22* *TRUNC (22.9) = 22*
***ROUND**(expr_date)*	dacă data *expr_date* este înainte de miezul zilei, întoarce data *d* cu timpul setat la ora 12:00 AM; altfel, este returnată data corespunzătoare zilei următoare, cu timpul setat la ora 12:00 AM;	TO_CHAR(ROUND(SYSDATE*), 'dd/mm/yy hh24:mi am') =* '*07/03/23 00:00 AM'* *ROUND (22.2) = 22* *ROUND (22.9) = 23*
LEAST(d1, d2, …, dn), GREATEST(d1, d2, …, dn)	dintr-o listă de date calendaristice, funcţiile întorc prima, respectiv ultima dată în ordine cronologică	*LEAST(SYSDATE, SYSDATE + 3, SYSDATE – 5) = SYSDATE-5* *GREATEST(SYSDATE, SYSDATE + 3, SYSDATE – 5) = SYSDATE + 3*

Operaţiile care se pot efectua asupra datelor calendaristice sunt următoarele:

Operaţie	**Tipul de date al rezultatului**	**Descriere**
expr_date -/+ expr_number	*Date*	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).
expr_date1 – expr_date2	*Number*	Intoarce numărul de zile dintre două date calendaristice. Data *expr_date1* trebuie să fie mai recentă decât *expr_date2*, altfel rezultatul este negativ.

- **Funcţii diverse:**

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

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

CASE *expr* *WHEN expr_1 THEN* *valoare_1* *[WHEN expr_2 THEN* *valoare_2* *...* *WHEN expr_n THEN* *valoare_n]* *[ELSE valoare]* *END*	În funcţie de valoarea expresiei *expr* returnează *valoare_i* corespunzătoare primei clauze ***WHEN .. THEN*** pentru care *expr = expresie_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 acelaşi tip de date.

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

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

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

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

III. [Exerciţii]

[Funcţii pe şiruri de caractere]

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

   <prenume angajat> <*nume angajat*> *castiga* <*salariu*> *lunar dar doreste* <*salariu de 3 ori mai mare*>. Etichetati coloana **“Salariu ideal”**. Pentru concatenare, utilizaţi atât funcţia ***CONCAT*** cât şi operatorul **“||”.**

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

2. Scrieţi o cerere prin care să se afişeze **prenumele salariatului** cu **prima litera majusculă** şi **toate celelalte litere minuscule**, **numele acestuia cu majuscule şi lungimea numelui**, pentru angajaţii 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 soluţii (cu operatorul *LIKE* şi funcţia *SUBSTR*).

3. Să se afişeze, pentru angajaţii cu prenumele „Steven”, **codul** şi **numele** acestora, precum şi **codul departamentului** în care lucrează. Căutarea trebuie să **nu** fie *case-sensitive*, iar eventualele *blank*-uri care preced sau urmează numelui trebuie ignorate.

--Varianta 1:

```
SELECT employee_id, last_name, department_id
FROM employees
WHERE LTRIM(RTRIM(UPPER(first_name)))='STEVEN';
```

--Varianta 2:

```
SELECT employee_id, last_name, department_id
FROM employees
WHERE TRIM(BOTH FROM UPPER(first_name))='STEVEN';
```

4. Să se afişeze pentru toţi angajaţii al căror nume se termină cu litera 'e', codul, numele, lungimea numelui şi poziţia din nume în care apare prima data litera 'A'. Utilizaţi *alias*-uri corespunzătoare pentru coloane.

[Funcţii aritmetice]

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

Obs: Soluția necesită rotunjirea diferenței celor două date calendaristice.

```
SELECT *
FROM employees
WHERE ___  ;
```

6. Să se afişeze **codul salariatului**, **numele**, **salariul**, **salariul mărit cu 15%,** exprimat cu două zecimale **şi numărul de sute** al salariului nou rotunjit la 2 zecimale. Etichetaţi ultimele două coloane “Salariu nou”, respectiv “Numar sute”. Se vor lua în considerare salariaţii al căror salariu **nu este divizibil** cu 1000.

```
SELECT employee_id, last_name, salary,
      round(salary + 0.15 * salary, 2)  "Salariu Nou",
      round((salary + 0.15 * salary) / 100, 2)  "Numar sute"
FROM employees
WHERE MOD(salary, 1000) != 0;
```

7. Să se listeze **numele** şi **data angajării** salariaţilor **care câştigă comision**. Să se eticheteze coloanele „Nume angajat”, „Data angajarii”. Utilizaţi funcţia ***RPAD*** pentru a determina ca data angajării să aibă lungimea de 20 de caractere.

```
SELECT last_name  AS "Nume angajat" , RPAD(to_char(hire_date),20,'X')  "Data angajarii"
FROM employees
WHERE  commission_pct IS NOT NULL;
```

[Funcţii şi operaţii cu date calendaristice]

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

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

9. Să se afişeze **numărul de zile** rămase până la sfârşitul anului.

```
SELECT to_date('31-12-2023','dd-mm-yyyy') - sysdate
FROM dual;
```

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

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

b) Să se afişeze data de peste 5 minute

***Obs**:* Cât reprezintă 5 minute dintr-o zi?

11. Să se afişeze **numele** şi **prenumele** angajatului (într-o singură coloană), **data angajării** şi **data negocierii salariului**, care este prima zi de Luni după 6 luni de serviciu. Etichetaţi 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 afişeze **numele** şi **numărul de luni** de la data angajării. Etichetaţi coloana “Luni lucrate”. Să se ordoneze rezultatul după numărul de luni lucrate. Se va rotunji numărul de luni la cel mai apropiat număr întreg.

```
-- prima varianta de ordonare

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

-- a doua varianta de ordonare

SELECT last_name, round(months_between(sysdate, hire_date)) "Luni lucrate"
FROM employees
ORDER BY 2;

-- a treia varianta de ordonare

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

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

[Funcţii diverse]

13. Să se afişeze **numele** angajaţilor şi **comisionul**. Dacă un angajat nu câştigă comision, să se scrie “Fara comision”. Etichetaţi coloana “Comision”.

14. Să se listeze **numele**, **salariul** şi **comisionul** tuturor angajaţilor al căror venit lunar (salariu + valoare comision) depăşeşte 10 000.

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

[Instrucţiunea CASE, comanda DECODE]

15. Să se afişeze **numele**, **codul functiei**, **salariul** şi o coloana care să arate **salariul după mărire**. Se ştie că pentru **IT_PROG** are loc o mărire de 10%, pentru **ST_CLERK** 15%, iar pentru **SA_REP** o mărire de 20%. Pentru ceilalti angajati nu se acordă mărire. Să se denumească coloana "Salariu renegociat".

```
--CASE VARIANTA 1:
SELECT last_name, job_id, salary,
   CASE job_id WHEN 'IT_PROG' THEN salary * 1.1
            WHEN 'ST_CLERK' THEN salary * 1.15
            WHEN 'SA_REP' THEN salary * 1.2
            ELSE salary
   END "Salariu renegociat"
FROM employees;
```

```
-- CASE VARIANTA 2:
SELECT last_name, job_id, salary,
   CASE WHEN job_id = 'IT_PROG' THEN salary * 1.1
        WHEN job_id ='ST_CLERK' THEN salary * 1.15
        WHEN job_id ='SA_REP' THEN salary*1.2
        ELSE salary
   END "Salariu renegociat"
FROM employees;
```

```
-- DECODE:
SELECT last_name, job_id, salary,
   DECODE(job_id, 'IT_PROG', salary * 1.1,
                  'ST_CLERK', salary * 1.15,
                  'SA_REP', salary * 1.2,
                   salary ) "Salariu renegociat"
FROM employees;
```

II. [Join]

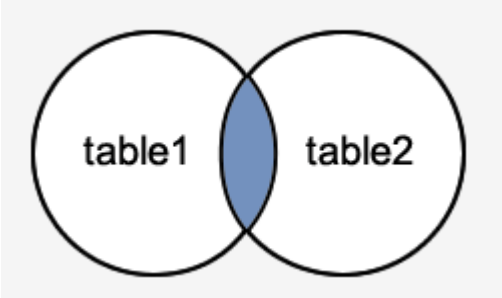
Join-ul este operaţia de regăsire a datelor din două sau mai multe tabele, pe baza valorilor comune ale unor coloane. De obicei, aceste coloane reprezintă **cheia primară**, respectiv **cheia externă** a tabelelor.

Condiţia de *join* se poate scrie în clauza *WHERE* a instrucţiunii *SELECT*. Într-o instrucţiune *SELECT* care uneşte tabele prin **operaţia de *join***, se recomandă ca numele coloanelor să fie precedate de numele sau alias-urile tabelelor pentru claritate şi pentru îmbunătăţirea timpului de acces la baza de date. Dacă acelaşi 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 puţin ***n* – 1 condiţii de *join***.

***Inner join* (equijoin, join simplu)** – corespunde situaţiei în care valorile de pe coloanele ce apar în condiţia de *join* trebuie să fie egale. (**returnează toate randurile din mai multe tabele – din tabelele specificate in 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 afişeze **codul angajatilor** şi **numele departamentului** pentru toţi angajaţii.

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 operaţia de join între tabelele ***employees*** şi ***departments***, pe baza coloanei comune ***department_id***. Observaţi utilizarea *alias*-urilor.

Ce se întâmplă dacă eliminăm condiţia de *join*?

Obs: Numele sau *alias*-urile tabelelor sunt obligatorii în dreptul coloanelor care au acelaşi 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 afişeze **numele angajatului**, **numele departamentului** şi **id-ul locației** pentru toţi angajaţii care câştigă comision.

```
SELECT ____, _____, _____
FROM   ____, _____
WHERE_____ AND  commission_pct _____ ;
```

19. Să se afişeze **numele**, **titlul job-ului** şi **denumirea departamentului** pentru toţi angajaţii care lucrează în Oxford (coloana - city).

20. Să se afişeze **codul angajatului** şi **numele** acestuia, împreună cu **numele** şi **codul şefului** 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 operaţia de *self*-join (inner join al tabelului cu el însuşi).

21. Să se modifice cererea anterioară pentru a afişa toţi salariaţii, inclusiv cei care nu au şef.

22. Scrieţi o cerere care afişează **numele angajatului, codul departamentului** în care acesta lucrează şi **numele colegilor** săi de departament. Se vor eticheta coloanele corespunzător.

23. Creaţi o cerere prin care să se afişeze **numele angajatilor**, **codul job-ului**, **titlul job-ului**, **numele departamentului** şi **salariul** angajaţilor. Se vor include și angajații al căror departament nu este cunoscut.

```
SELECT last_name, j.job_id, job_title, department_name, salary
FROM employees e, departments d, jobs j
WHERE e.department_id = d.department_id (+)
        AND  j.job_id = e.job_id;
```

24. Să se afişeze **numele** şi **data angajării** pentru salariaţii care au fost angajaţi după *Gates*.

```
SELECT ang.last_name NumeAng, ang.hire_date DataAng,
       gates.last_name NumeGates, gates.hire_date DataGates
FROM employees ang, employees gates
WHERE ____ ;
```

25. Să se afişeze **numele** salariatului şi **data angajării** împreună cu **numele** şi **data angajării şefului direct** pentru salariaţii care au fost angajaţi **înaintea** şefilor lor. Se vor eticheta coloanele Angajat, Data_ang, Manager si Data_mgr.

```
SELECT ang.last_name Angajat, ang.hire_date Data_Ang, m.last_name Manager,
m.hire_date Data_mgr
FROM employees ang, employees m
WHERE ang.manager_id = m.employee_id AND ang.hire_date < m.hire_date;
```