1. TEORIE

Operatori pe mulțimi

Operatorii pe mulțimi combină rezultatele obținute din două sau mai multe interogări. Cererile care conțin operatori pe mulțimi se numesc ***cereri compuse***.

Există patru operatori pe mulțimi:

1. **UNION** - returnează toate liniile selectate de două cereri, eliminând duplicatele
   - nu ignoră valorile *null*
   - coloanele selectate trebuie să corespundă ca număr şi tip de date

```
SELECT nume_col1, nume_col2, ..., nume_coln FROM nume_tabel_1
UNION
SELECT nume_col1, nume_col2, ..., nume_coln FROM nume_tabel_2;
```

2. **UNION ALL** - echivalent cu *UNION*, dar fără a elimina duplicatele
   - în cererile asupra cărora se aplică *UNION ALL* nu poate fi utilizat cuvântul cheie *DISTINCT*

```
SELECT nume_col1, nume_col2, ..., nume_coln FROM nume_tabel_1
UNION ALL
SELECT nume_col1, nume_col2, ..., nume_coln FROM nume_tabel_2;
```

3. **INTERSECT** - returnează toate liniile comune cererilor asupra cărora se aplică
   - nu ignoră valorile *null*
   - coloanele selectate trebuie să corespundă ca număr şi tip de date

```
SELECT nume_col1, nume_col2, ..., nume_coln FROM nume_tabel_1
INTERSECT
SELECT nume_col1, nume_col2, ..., nume_coln FROM nume_tabel_2;
```

4. **MINUS** - determină liniile returnate de prima cerere care nu apar în rezultatul celei de-a doua cereri
   - este necesar ca toate coloanele din clauza *WHERE* să se afle şi în clauza *SELECT*
   - coloanele selectate trebuie să corespundă ca număr şi tip de date

```
SELECT nume_col1, nume_col2, ..., nume_coln FROM nume_tabel_1
MINUS
SELECT nume_col1, nume_col2, ..., nume_coln FROM nume_tabel_2;
```

Observații:

- Toți operatorii pe mulțimi au aceeaşi precedență. Dacă o instrucțiune *SQL* conține mai mulți operatori pe mulțimi, *server*-ul *Oracle* evaluează cererea de la stânga la dreapta (sau de sus în jos). Pentru a schimba această ordine de evaluare, se pot utiliza paranteze.
- În mod implicit, pentru toți operatorii cu excepția lui *UNION ALL*, rezultatul este ordonat crescător după valorile primei coloane din clauza *SELECT*.
- Pentru o cerere care utilizează operatori pe mulțimi, cu excepția lui *UNION ALL*, *server*-ul *Oracle* elimină liniile duplicat.

Funcţii SQL

➔ Sunt funcţii predefinite in sistemul Oracle.
➔ Dacă sunt apelate cu:
- ❖ un argument având un alt tip de date decât cel aşteptat, sistemul converteşte implicit argumentul.
- ❖ un argument *null*, returnează valoarea *null*.
  Excepţii: *CONCAT*, *NVL* şi *REPLACE*.

➔ Pot fi clasificate în:
1. Funcţii *single-row*
2. Funcţii *multiple-row*

1. Funcţiile *single row*

➔ returnează câte o singură linie rezultat pentru fiecare linie a tabelului sau vizualizării interogate

1.1. Funcţii de conversie

TO_CHAR(value, [format]) - converteşte un număr / o dată calendaristică în şir de caractere

```
TO_CHAR(7) = '7'
TO_CHAR(876.53, '$999.9') = '$876.5'
TO_CHAR(SYSDATE, 'DD/MM/YYYY') = '01/03/2017'
```

TO_DATE(value, [format]) - converteşte un număr / un şir de caractere în dată calendaristică; dacă nu este specificat formatul sirul de caractere trebuie sa aiba formatul default.

```
TO_DATE('January.12.2008','Month.DD.YYYY') => 12-JAN-2008
TO_DATE('12-JAN-2008') => 12-JAN-2008
```

TO_NUMBER(value, [format]) - converteşte (sau formatează) un şir de caractere în număr. Formatul trebuie sa corespundă şirului de caractere.

```
TO_NUMBER('-1210.73')  = -1210.73
TO_NUMBER('1210.73', '9999.99') = 1210.73
```

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*.

1.2. Funcţii pentru prelucrarea caracterelor

LENGTH(string) - întoarce lungimea şirului de caractere primit ca parametru

```
LENGTH('Ana') = 3
```

SUBSTR(string, start, [n]) - întoarce subşirul lui *string* care începe pe poziția *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'
SUBSTR('Informatica', -5, 2) = 'at'
```

LTRIM(string, ['chars']) / RTRIM(string, ['chars']) - şterge din stânga/dreapta 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 spațiile libere din stânga / dreapta lui *string*;

```
LTRIM('    Informatica') = 'Informatica'
LTRIM('aaaaInformatica', 'aIn') = 'formatica'

RTRIM('Informatica    ') = 'Informatica'
RTRIM('Informaticaaaaa', 'cai') = 'Informat'
```

TRIM (LEADING | TRAILING | BOTH chars FROM expresie) - elimină caracterele specificate (*chars*) de la începutul (*leading*) , sfârşitul (*trailing*) sau din ambele părți, 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']) / RPAD(string, length [,'chars']) - adaugă *chars* la stânga/dreapta şirului de caractere *string* până când lungimea noului şir devine *length*; în cazul în care *chars* nu este specificat, atunci se adaugă spații libere la stânga lui *string*;

```
LPAD('iNfO',6) = '  iNfO'
RPAD('info', 6, 'X')  = 'infoXX'
```

REPLACE(string1, string2 [,string3]) - întoarce *string1* cu toate aparițiile lui *string2* înlocuite prin *string3*; dacă *string3* nu este specificat, atunci toate aparițiile 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 poziția *start*, a *n*-a apariție a secvenței *chars* şi întoarce poziția respectivă; dacă *start* nu este specificat, căutarea se face de la începutul şirului; dacă *n* nu este specificat, se caută prima apariție a secvenței *chars*;

```
INSTR(LOWER('AbC aBcDe'), 'ab', 5, 2) = 0
INSTR(LOWER('AbC aBcDe'), 'ab', 5, 1) = 5
INSTR(LOWER('AbCdE aBcDe'), 'ab', 5) = 7
```

TRANSLATE(string, source, destination) - returnează *string* cu toate apariţiile fiecărui caracter din *source* înlocuite cu caracterul corespunzător din *destination (*aflat pe aceeași poziție)

```
TRANSLATE('$a$aa','$','b') = 'babaa'
TRANSLATE('$a$aaa','$a','bc')  = 'bcbccc'
```

1.3. Funcţii aritmetice

1.3.1. Care operează asupra unei singure valori

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
MOD - restul împărțirii a două numere specificate

1.3.2. Care operează asupra unei liste de valori

LEAST / GREATEST - cea mai mică, respectiv cea mai mare valoare a unei liste de expresii (valabilă şi pentru date calendaristice)

1.4. Funcţii pentru prelucrarea datelor calendaristice

ADD_MONTHS(expr_date, nr_luni) - întoarce data care este după *nr_luni* luni de la data *expr_date*

```
ADD_MONTHS('02-MAR-2016', 3) = '02-JUN-2016'
```

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('02-MAR-2016', 'Monday') = '07-MAR-2007'
```

LAST_DAY(expr_date) - întoarce data corespunzătoare ultimei zile a lunii din care data *expr_date* face parte

```
LAST_DAY('02-MAR-2016') = '31-MAR-2016'
```

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('02-DEC-2014','10-OCT-2011') = 37.7419355
MONTHS_BETWEEN('10-OCT-2011', '02-DEC-2014') = -37.7419355
```

Operațiile care se pot efectua asupra datelor calendaristice sunt următoarele:

Operație	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.

1.5. Funcţii diverse

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 funcția întoarce valoarea

```
DECODE('a', 'a', 'b', 'c') = 'b'
DECODE('b', 'a', 'b', 'c') = 'c'
```

Utilizarea funcției *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 funcție 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.

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') -- eroare, nu are loc conversia implicită
```

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 funcția 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
```

2. **Funcţiile *multiple row*** (laboratorul următor)

---

2. EXERCIŢII

[Operatori pe mulţimi]

1. Se cer codurile departamentelor al căror nume conține şirul “re” sau în care lucrează angajați având codul job-ului “SA_REP”. Rezolvaţi problema folosind două metode (UNION şi JOIN). Afişati rezultatele în ordinea default de la UNION şi verificaţi că aţi obţinut acelaşi lucru folosind operatori pe mulţimi. Ce se întâmplă dacă înlocuim *UNION* cu *UNION ALL*?

2. Să se obţină codurile departamentelor în care nu lucreaza nimeni.

3. Se cer codurile departamentelor al căror nume conține şirul “re” şi în care lucrează angajați având codul job-ului “HR_REP”.

4. Să se afişeze numele şi prenumele angajaților reunite cu id-ul jobului şi data angajării.

5. Să se afişeze numele şi prenumele angajaților reunite cu numele departamentelor.

[Funcții pe şiruri de caractere]

6. 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>*. Etichetati coloana “Salariu ideal”.

7. Scrieți 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 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* şi funcția *SUBSTR*).

8. Să se afişeze, pentru angajații 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.

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

[Funcții aritmetice]

10. Să se afişeze detalii despre salariații care au lucrat un număr întreg de săptămâni până la data curentă.

    Obs: Soluția necesită rotunjirea diferenței celor două date calendaristice. De ce este necesar acest lucru?

11. 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. Etichetați ultimele două coloane “Salariu nou”, respectiv “Numar sute”. Se vor lua în considerare salariații al căror salariu nu este divizibil cu 1000.

12. Să se listeze numele şi data angajării salariaților care câştigă comision. Să se eticheteze coloanele „Nume angajat”, „Data angajarii”. Utilizați funcția *RPAD* pentru a vă asigura că data angajării are lungimea de 20 de caractere.

[Funcții şi operații cu date calendaristice]

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

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

15. a) Să se afişeze data de peste 12 ore.

    b) Să se afişeze data de peste 5 minute

16. 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. Etichetați această coloană “Negociere”.

17. Pentru fiecare angajat să se afişeze numele şi 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.

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

18. Să se afişeze numele, data angajării şi ziua săptămânii în care a început lucrul fiecare salariat. Etichetați coloana “Zi”. Ordonați rezultatul după ziua săptămânii, începând cu Luni.

[Funcții diverse]

19. Să se afişeze numele angajaților şi comisionul. Dacă un angajat nu câştigă comision, să se scrie “Fara comision”. Etichetați coloana “Comision”.

20. Să se listeze numele, salariul şi comisionul tuturor angajaților al căror venit lunar (salariu + valoare comision) depăşeşte 10000.

[Instrucțiunea CASE, comanda DECODE]

21. Să se afişeze numele, codul job-ului, salariul şi o coloană care să arate salariul după mărire. Se presupune că pentru IT_PROG are loc o mărire de 20%, pentru SA_REP creşterea este de 25%, iar pentru SA_MAN are loc o mărire de 35%. Pentru ceilalți angajați nu se acordă mărire. Să se denumească coloana "Salariu renegociat". Să se rezolve exercitiul în 2 moduri: folosind CASE şi DECODE.