

## 2. OPERATORI ȘI FUNCȚII

## 2.1. Operatori

Un operator este un simbol care specifică o anumită operatie executată asupra unei sau mai multe expresii operand. În SQL Server există următoarele categorii de operatori:

Operatorul de atribuire

În SQL Server există un singur operator de atribuire, reprezentat prin semnul de egalitate ( $=$ ). Operatorul de atribuire poate fi folosit numai în cadrul uneor instrucțiuni SET sau într-o clauză (instrucțione) SELECT.

Operatori per biti

Operatori pe băti acționează asupra a două expreșii de tip întreg și realizează una dintr-un mătoarele operații la nivel de băti:

Semantics	
Operator	
&	Operation AND ( $\wedge$ ) pe biti
!	Operation OR ( $\vee$ ) pe biti
$\wedge$	Operation XOR ( $\oplus$ ) pe biti

Operatori de comparare

Operatorii de comparare testeză dacă două expresii sunt sau nu egale, respectiv dacă una dintr-elle este mai mare sau nu decât cealaltă. Operatorii de comparare se pot aplica tuturor tipurilor de expresii exceptând text, ntext sau image. Operatorii de comparare acceptă de la **SQL Server** sun:

Operatori aritmetici

Operatorii aritmetici execută operații matematice asupra a două expresii

Operator	Semnificație
+	Adunare.
-	Scădere.
*	Multiplicare.
/	Divizare.
$\frac{a}{b}$	Modulo = returnează restul unei împărțiri întregi.

Obenwinkel

- Observații:**

  1. Operatorul modulo se aplică numai unor operanți de tip întreg și returnază ca rezultat tot un întreg. De exemplu:  $12 \% 5 = 2$  adică restul împărțirii întregi a lui 12 la 5.
  2. Operatorii + și - se pot folosi și pentru operații asupra unor valori de tip **datetime** și **smalldatetime**.

1. Rezultatul oricărei operații de comparare este de tip boolean, având una din valorile: TRUE, FALSE sau UNKNOWN (în cazul comparațiilor cu valori NULL).

2. Spec deosebire de alte tipuri de date din SQL Server, tipul de date boolean nu poate fi specificat ca tipul de date al unei coloane sau variabile și nu se poate returna într-o relație rezultat. Expresiile booleene se pot folosi doar în următoarele contexte: clauzele WHERE sau HAVING, expresii CASE, instrucțiuni IF și WHILE.

Operator	Semnificație
=	Egalitate.
>	Mai mare.
<	Mai mic.
>=	Mai mare sau egal.
<=	Mai mic sau egal.
$\neq$	Diferit.
is	Diferit (nu este prevăzut în standardul SQL '92).
is not	Nu mai mic decât (nu este prevăzut în standardul SQL '92).
is similar to	Nu mai mare decât (nu este prevăzut în standardul SQL '92).
is not similar to	Nu mai mic decât (nu este prevăzut în standardul SQL '92).

### Observation

1. Rezultatul oricărui operări de comparare este de tip boolean, având una din valorile: TRUE, FALSE sau UNKNOWN (în cazul comparărilor cu valori NULL).
  2. Spie deosebire de alte tipuri de date din SQL Server, tipul de date boolean nu poate fi specificat ca tipul de date al unei coloane sau variabile și nu se poate returna într-o relație rezultat. Expresiile booleene se pot folosi doar în următoarele contexte: clauzele WHERE sau HAVING, expresii CASE, instrucțiuni IF și WHILE.

3. Dacă opțiunea **SET ANSI\_NULLS** este ON, atunci orice operator de comparare care are unul sau doi operanzi cu valoarea NULL va returna valoarea UNKNOWN. Dacă opțiunea **SET ANSI\_NULLS** este OFF, atunci se aplică aceleasi reguli, exceptând operatorul de egalitate (=) care returnează TRUE dacă ambi operatori sunt NULL. Astfel, testul `NULL = NULL` returnează TRUE de către SET **ANSI\_NULLS** este OFF și UNKNOWN dacă SET **ANSI\_NULLS** este ON.

### Observație:

- Implicit, șirul vid este interpretat ca atare în operațiile de concatenare. De exemplu: 'abc' + '' + 'def' produce ca rezultat 'abcdef'.
- Șirul vid nu trebuie confundat cu valoarea NULL a unui șir. De exemplu: 'abc' + NULL + 'def' produce ca rezultat NULL, adică un șir cu valoare nedefinită.

### Operatori logici

Operatorii logici testează valoarea de adevăr a unei condiții. La fel ca și operatorii de comparație, operatorii logici returneză o valoare de tip boolean care poate fi TRUE, FALSE sau UNKNOWN.

Operator	Semnificație
ALL	TRUE dacă toate comparațiile dintr-un set sunt TRUE.
	TRUE dacă ambele expresii booleene date ca operanzi sunt TRUE, UNKNOWN dacă una din expresii este UNKNOWN și cealaltă TRUE, respectiv FALSE dacă una din expresii este FALSE.
ANY	TRUE dacă cel puțin una dintre comparațiile dintr-un set este TRUE.
BETWEEN	TRUE dacă operandul este între două valori (inclusiv).
EXISTS	TRUE dacă o subînărcare returnează cel puțin o tuplă.
IN	TRUE dacă operandul este egal cu cel puțin una dintre valorile unei liste de expresii sau a unei relații rezultat.
LIKE	TRUE dacă operandul respectă un anumit şablon.
NOT	TRUE dacă operandul este FALSE, FALSE dacă operandul este TRUE și UNKNOWN dacă operandul este UNKNOWN.
OR	TRUE dacă cel puțin una din expresiile booleene dă ca operand este TRUE, FALSE dacă ambele expresii sunt FALSE și UNKNOWN în rest.
SOME	TRUE dacă cel puțin una dintre comparațiile dintr-un set este TRUE.

### Operatori Unari

Operatorii unari execută o operație dată doar asupra unei singure expresii de tip numeric. Operatorii unari din SQL SERVER sunt:

Operator	Semnificație
+	Valoarea numerică este pozitivă.
-	Valoarea negativă a operandului.
~	Complementul lăsat de unu al operandului (Bitwise NOT).

### Precedență operatorilor

Precedența operatorilor determină ordinea de execuție a acestora în cazul unor expresii care contin o secvență de operatori. Un operator cu precedență mai mare este executat înaintea altuia cu precedență mai mică. Precedența operatorilor din SQL Server este dată în ordine crescătoare astfel:

Operatori Unari:	+, -, ~
Operatori aritmici multiplicativi:	* / %
Operatori aritmici aditivi:	(Adunare), + (Concatenare), - (Scadere)
Operatori de comparație:	=, >, <, >=, <=, !=, >>
Operatori pe biții:	& &
Operatorul de negație logică:	NOT
Operatorul SI logic:	AND
Alți operatori logici:	ALL, ANY, BETWEEN, IN, LIKE, OR, SOME
Operatorul de atribuire:	=

Dacă doi operatori dintr-o expresie au aceeași precedență, evaluarea lor se face de la stânga la dreapta în ordinea aparținării lor în expresie. Ordinea implicită de execuție poate fi modificată prin folosirea parantezelor.

Operatorul de concatenare a șirurilor de caractere este notat cu semnul de adunare (+). Aceasta este singura operație de manipulare a șirurilor de caractere care se realizează printre-un operator, toate celelalte fiind realizate prin diferite funcții.

### Operatorul de concatenare

Operatorul de concatenare a șirurilor de caractere este notat cu semnul de adunare (+). Aceasta este singura operație de manipulare a șirurilor de caractere care se realizează printre-un operator, toate celelalte fiind realizate prin diferite funcții.

## Operatorul BETWEEN

Sintaxa operatorului BETWEEN este următoarea:

*expresie\_test [NOT] BETWEEN*

*expresie\_de\_la AND expresie\_pina\_la*

și returnează valoarea TRUE dacă *expresie\_test* are valoarea între *expresie\_de\_la* și *expresie\_pina\_la* (inclusiv) și FALSE în caz contrar (în varianta fără NOT). Prin folosirea operatorului NOT se reversează valoarea rezultatului returnat.

### Observații:

1. Operatorul BETWEEN realizează un test inclusiv. Pentru a face un test exclusiv se vor folosi operatorii > și <.
2. Dacă oricare dintre cele trei expresii operand ale operatorilor BETWEEN sau NOT BETWEEN au valoarea NULL, atunci rezultatul este UNKNOWN.

## Operatorul LIKE

Determinăm dacă un sir de caractere dat respectă sau nu un anumit şablon (pattern). Un şablon poate include atât caractere obişnuite, cât și caractere de înlocuire (wildcard). La compararea unui sir de caractere cu un şablon caracterelor obişnuite trebuie să se potrivească exact cu caracterele din sir. În schimb, caracterele de înlocuire pot fi înlocuite cu fragmente arbitrar ale sirului de caractere. Prin folosirea caracterelor de înlocuire operatorul LIKE devine mult mai flexibil decât operatorul = sau <>.

Sintaxa operatorului LIKE este următoarea:

*expresie\_sir\_caractere [NOT] LIKE şablon*

[ESCAPE *caracter\_escape*]

unde:

- expresie\_sir\_caractere* – este orice expresie care se evaluează la un sir de caractere valid în SQL Server.  
*şablon* – este şablonul cu care se compară sirul de caractere. Şablonul poate conține următoarele caractere de înlocuire:

Caracter de înlocuire	Descriere
%	Orice fragment de sir conținând zero sau mai multe caractere.
_	Un singur caracter oricare.
[]	Orice caracter dintr-un set sau interval dat.
[^]	Orice caracter care nu face parte dintr-un set sau interval dat.

- caracter\_escape* – este orice expresie validă de tip sir de caractere. *caracter\_escape* nu are valoare implicită și constă dintr-un singur caracter.

Operatorul LIKE returnează TRUE dacă *expresie\_sir\_caractere* se potrivește cu şablonul specificat.

### Observații:

1. La compararea unui sir de caractere prin LIKE toate caracterele din şablon sunt semnificative, inclusiv spațiile de la început sau sfârșit. De exemplu:

'abc' LIKE 'abc'

returnează FALSE. Totuși, aceste spații nu se iau în considerare la sirul de comparare. De exemplu oricare dintre comparațiile:

'abc' LIKE 'abc', 'abc' LIKE 'abc', 'abc' LIKE 'abc'

returnează TRUE.

2. Regula de comparare a spațiilor se poate manifesta în mod neașteptat atunci când se fac comparații cu variabile sau câmpuri având tipul de date char sau varchar. Un char de o anumită dimensiune va conține spații de completare ori de către ori sirul actual memorat este mai scurt decât dimensiunea declarată. Aceasta ar putea duce la eşuarea comparațiilor dacă se folosește un şablon având tipul de date char. O expresie de tip varchar are întotdeauna dimensiunea şirului actual pe care îl conține fără a fi completat cu spații la sfârșit. Acest fapt ar putea cauza probleme dacă se folosește un sir de caractere de comparat având tipul varchar.

### Exemple:

1. Furnizorii al căror nume începe cu litera 'F':

```
SELECT *  
FROM Furnizor  
WHERE Num LIKE 'F%'
```

2. Furnizorii al căror nume începe cu litera 'A' sau 'F':

```
SELECT *
FROM Furnizor
WHERE Nume LIKE 'A%' OR
      Nume LIKE 'F%'
```

3. Furnizorii al căror nume nu începe cu litera 'A' sau 'F':

*Varianta1:*

```
SELECT *
FROM Furnizor
WHERE Nume NOT LIKE '[AF]%'
```

*Varianta2:*

```
SELECT *
FROM Furnizor
WHERE Nume LIKE '[^AF]%'
```

#### Observație:

Pentru interogarea de mai sus cele două variante de rezolvare sunt echivalente și vor returna același rezultat. Acest lucru nu mai este adeverat în cazul săabloanelor care în partea fixă au specificat o secvență de mai multe caractere. Astfel, următoarea frază returneză furnizorii al căror nume nu începe cu secvența 'S.C.':

```
SELECT *
FROM Furnizor
WHERE Nume NOT LIKE 'S.C.%'
```

În timp ce următoarea frază elimină din rezultat și numele care încep cu 'S', 'S.' și 'S.m.d.'

```
SELECT *
FROM Furnizor
WHERE Nume LIKE '[^S][^.][^c] [^.]%'
```

Aceasta se datorează faptului că la comparațiile de siruri cu săabloane care conțin caractere de înlocuire negative **SQL Server** face evaluarea pas cu pas, pentru că un singur caracter de înlocuire la un moment dat.

4. Furnizorii al căror nume conține caracterul '\_' în a doua poziție:

```
SELECT *
FROM Furnizor
WHERE Nume LIKE '_[_]%'
```

#### Observație:

- Caracterele de înlocuire se pot folosi ca literale în săabloane prin închiderea lor între paranteze drepte. În interogarea de mai sus simbolul '\_' este folosit ca și caracter de înlocuire în prima sa apariție și ca literal în a doua.

Tabelul de mai jos conține câteva exemple de folosire a operatorului LIKE și a caracterelor de înlocuire pe post de literale:

Săbton	Semnificație
LIKE 'S%'	5%
LIKE '_%'	_n
LIKE '[a-e]ff'	a,b,c,d sau f
LIKE '[^a-e]ff'	,a,c,d sau f
LIKE '[_]'	_
LIKE 'T'	T
LIKE 'abc[_d%'	oicice incepe cu abc, d
LIKE 'abc[_def'	abcd, abce sau abef

- Interogarea 4) se poate rezolva și prin folosirea clauzei **ESCAPE** și a caracterului de escape. Orice caracter de înlocuire care urmează după un caracter escape este interpretat ca literal. Dacă alegem să folosim simbolul '\_' pe post de caracter de escape, atunci interogarea 4) se rezolvă astfel:

```
SELECT *
FROM Furnizor
WHERE Nume LIKE '_!_%' ESCAPE '!'
```

## 2.2. Funcții SQL

Limbajul de programare **Transact-SQL** oferă trei categorii de funcții:

- *funcții care returneză un set de înregistrări (rowset functions)*

ACEste funcții pot înlocui nume de tabele din frazele **SQL**.

- *funcții de agregare*
- Operează asupra unei colecții de valori, de regulă provenite din una sau mai multe coloane ale unor tabele și returnează o singură valoare agregată. Funcțiile de agregare pot fi folosite exclusiv în cadrul frazelor **SELECT** și vor fi prezentate în contextul acestora.

- *funcții scalare*
- Operează asupra unei singure valori și returnează tot o singură

valoare. O funcție scalară poate fi folosită oriunde este legală plasarea unei expresii.

## 2.2.1. Funcții scalare

Limbajul de programare Transact-SQL oferă mai multe categorii de funcții scalare dintre care cele mai comune sunt: funcțiile matematice, funcțiile care operează asupra șirurilor de caractere și funcțiile pentru manipularea valorilor de tip dată calendaristică. De asemenea, de interes sunt funcțiile sistem de tip scalar. Dintre acestea cele mai frecvent utilizate sunt: funcțiile de conversie explicită (CAST și CONVERT), funcțiile (expresii) CASE și funcțiile care tratează valorile de tip NULL (ISNULL).

### 2.2.1.1. Funcții matematice

Funcțiile matematice realizează calculul valorii unei funcții specificate pe baza a una sau mai multe valori numerice date ca argumente și returnează tot o valoare numerică. Cele mai importante funcții matematice împreună cu sintaxa acestora sunt prezentate mai jos:

Sintaxă	Semnificație
<code>ABS(<i>expresie numerică</i>)</code>	valoarea absolută a argumentului;
<code>ACOS(<i>expresie floatantă</i>)</code>	valoarea în radiani a unghiului al căruia cosinus este egal cu argumentul;
<code>ATN(<i>expresie floatantă</i>)</code>	valoarea în radiani a unghiului al căruia sinus este egal cu argumentul;
<code>ATAN(<i>expresie floatantă</i>)</code>	valoarea în radiani a unghiului a căruia tangentă este egală cu argumentul;
<code>ATN2(<i>expresie floatantă</i>, <i>expresie floatantă</i>)</code>	valoarea în radiani a unghiului a căruia tangentă este egală cu argumentul;
<code>CEILING(<i>expresie numerică</i>)</code>	valoarea între cele două argumente, cel mai mic întreg mai mare sau egal cu argumentul;
<code>COS(<i>expresie floatantă</i>)</code>	cosinusal argumentului specificat în radiani;
<code>COT(<i>expresie floatantă</i>)</code>	cotangenta argumentului specificat în radiani;
<code>DEGREES(<i>expresie numerică</i>)</code>	valoarea în grade a argumentului specificat în radiani;
<code>EXP(<i>expresie floatantă</i>)</code>	exponentiala argumentului, cel mai mare întreg mai mic sau egal cu argumentul;
<code>FLOOR(<i>expresie numerică</i>)</code>	valoarea logaritmului natural al argumentului;
<code>LOG(<i>expresie floatantă</i>)</code>	valoarea logaritmului zecimal al argumentului;
<code>LOG10(<i>expresie floatantă</i>)</code>	valoarea constantei PI;
<code>PI()</code>	ridicare la putere;
<code>POWER(<i>expresie numerică</i>, <i>putere</i>)</code>	valoarea în radiani a argumentului specificat în grade;
<code>RADIANS(<i>expresie numerică</i>)</code>	valoare floatantă aleatoare în intervalul 0 la 1;
<code>RAND([origine])</code>	

Sintaxă	Semnificație
<code>ROUND(<i>expresie numerică</i>, <i>lungime[, funcție]</i>)</code>	valoarea argumentului rotunjită la lungimea sau precizia specificată, semnul argumentului pozitiv (+1), zero (0), negativ (-1);
<code>SIGN(<i>expresie numerică</i>)</code>	sinusul argumentului specificat în radiani;
<code>SIN(<i>expresie floatantă</i>)</code>	pătratul argumentului;
<code>SQUARE(<i>expresie floatantă</i>)</code>	rădăcina pătrată a argumentului;
<code>SQRT(<i>expresie floatantă</i>)</code>	tangenta argumentului specificat în radiani;
<code>TAN(<i>expresie floatantă</i>)</code>	tangenta argumentului specificat în radiani;

Observații:

1. *expresie\_numerică* se evaluatează la orice valoare de tip numeric exceptând tipul **bit**;
2. *expresie\_floatantă* se evaluatează la orice valoare de tip floatant (**real** sau **float**);
3. *putere* se evaluatează la orice valoare de tip numeric exceptând tipul **bit**;
4. În cazul funcțiilor ACOS și ASIN valoarea argumentului trebuie să fie în intervalul [-1,1], în caz contrar funcțiile returnează valoarea NULL și se semnalizează o eroare de domeniu;
5. Similar, în cazul funcțiilor LOG, LOG10 și SQRT, argumentul trebuie să fie pozitiv, în caz contrar funcțiile returnează valoarea NULL și se semnalizează o eroare de domeniu;
6. *lungime* este precizia de rotunjire a lui *expresie\_numerică* prin funcția ROUND. Dacă *lungime* este un număr pozitiv, atunci rotunjirea se face la numărul de zecimale indicate de acesta. Dacă *lungime* este un număr negativ atunci rotunjirea se face la numărul de poziții de la stânga punctului decimal indicat de valoarea absolută a acestuia. Dacă numărul de cifre semnificative a lui *expresie\_numerică* este mai mic decât valoarea absolută a parametrului *funcție*, atunci rezultatul returnat de funcție este zero. Parametrul *funcție* indică tipul de rotunjire care se face: valoarea 0 (implicită) semnifică rotunjirea propriu-zisă, iar o valoare diferită de 0 indică trunciere. Atât *lungime* cât și *funcție* pot fi de tip tinyint, smallint sau int.
7. Funcțiile aritmétice cum ar fi ABS, CEILING, DEGREES, FLOOR, POWER, RADIAN și SIGN, returnează o valoare care are același tip de date ca argumentul. Funcțiile trigonometrice împreună cu EXP, LOG, LOG10, SQUARE și SQRT, returneză valori floatante.

### Exemple:

```

SELECT ABS(-1)           - rezultat 1
SELECT ABS(-1.0)          - rezultat 1.0
SELECT ACOS(-1)           - rezultat 3.1415926535897931
SELECT DEGREES(ACOS(-1))   - rezultat 180.0

SELECT POWER(2, 0.5)       - rezultat 1
SELECT POWER(2, 0, 0.5)     - rezultat 1.4
SELECT POWER(2, 0.0000, 0.5) - rezultat 1.4142135623730952
SELECT POWER(2, 0.0000000000000000, 0.5) - rezultat 1.4142135623730951

SELECT SQRT(2)            - rezultat 1.4142135623730951

SELECT ROUND(748.58, -1)    - rezultat 750.00
SELECT ROUND(748.58, -2)    - rezultat 700.00
SELECT ROUND(748.58, -3)    - rezultat 1000.00
SELECT ROUND(748.58, -4)    - rezultat 0
SELECT ROUND(123.4565, 2)   - rezultat 123.4600
SELECT ROUND(123.4565, 2, 0) - rezultat 123.4600
SELECT ROUND(123.4565, 2, 1) - rezultat 123.4500
SELECT TAN(PI() / 4)        - rezultat 0.9999999999999998

```

### 2.2.1.2. Funcții care operează asupra șirurilor de caractere

Functiile scalare pe șiruri de caractere operează, de regulă, însupra unui șir de caractere dat ca argument și returnează fie un șir de caractere, fie o valoare numerică.

Cele mai importante funcții pe șiruri de caractere împreună cu sintaxa acestora și valoarea returnată sunt prezentate mai jos:

Sintaxa	Semnificație
LEN( <i>expresie_caracter</i> )	dimensiunea egală cu valoarea parametrului <i>lungime</i> , lungimea în caractere a șirului <i>expresie_caracter</i> .
LOWER( <i>expresie_caracter</i> )	exclusiv eventuală caractere spații de la sfârșitul șirului; expresia rezultată în urma transformării în litere mici a tuturor caracterelor din <i>expresie_caracter</i> .
LTRIM( <i>expresie_caracter</i> )	șirul de caractere rezultat prin eliminarea spațiilor din stânga lui <i>expresie_caracter</i> .
NCHAR( <i>expresie_numerică_UNICOD</i> )	caracterul UNICOD cu al căru cod este <i>expresie_numerică_UNICOD</i> .
PATINDEX('%' <i>pattern%</i> ', <i>expresie_caracter</i> )	poziția de start a primei aparitii a lui <i>pattern</i> în <i>expresie_caracter</i> sau 0 în caz contrar.
REPLACE( <i>'expresie_sir1'</i> , <i>'expresie_sir2'</i> , <i>'expresie_sir3'</i> )	înllocuiesc în <i>expresie_sir3</i> toate aparițiile lui <i>expresie_sir2</i> din <i>expresie_sir1</i> se mențineaza șirul de caractere dat ca prim argument întrucât de delimitare (implicit '!' și ' ') nu specifică prin al doilea argument ca fiind una din: ' ', ',', '.', ') punctu a forma un sir UNICOD valid ca identificator în SQL Server.
REPLICATE( <i>expresie_caracter</i> , <i>intreg</i> )	repetă <i>expresie_caracter</i> de un număr de ori specificat prin <i>intreg</i> .
REVERSE( <i>expresie_caracter</i> )	reversează șirul de caractere dat ca parametru.
RIGHT( <i>expresie_caracter_lungime</i> )	segmentul din dreapta lui <i>expresie_caracter</i> având dimensiunea egală cu valoarea parametrului <i>lungime</i> .
RTRIM( <i>expresie_caracter</i> )	șirul de caractere rezultat prin eliminarea spațiilor din dreapta lui <i>expresie_caracter</i> .
SOUNDEX( <i>expresie_caracter</i> )	returnează un cod SOUNDEX de 4 caractere folosind similaritatea dintre două șiruri.
SPACE( <i>lungime</i> )	șir de caractere format dintr-un număr de <i>lungime</i> spații.
STR( <i>expresie_floating_lungime</i> [, <i>zecimale</i> ])	șirul rezultat prin înlocuirea cu segmentul din <i>expresie_caracter2</i> a segmentului din <i>expresie_caracter1</i> care începe din poziția <i>start</i> și are <i>lungime</i> caractere;
STUFF( <i>expresie_caracter1</i> , <i>start</i> , <i>lungime</i> , <i>expresie_caracter2</i> )	subșirul din <i>expresie_caracter1</i> care începe din poziția <i>start</i> și are <i>lungime</i> caractere;
SUBSTRING( <i>expresie_caracter</i> , <i>start</i> , <i>lungime</i> )	poziția <i>start</i> și are <i>lungime</i> caractere;
UNICODE( <i>expresie_caracter</i> )	codul UNICOD al celui mai din stânga caracter din argumentului;
UPPER( <i>expresie_caracter</i> )	expresia rezultată în urma transformării în litere mari a tuturor caracterelor din <i>expresie_caracter</i> .

### Observații:

- i. *expresie\_numerică\_ASCII* se evaluează la un întreg între 0 și 255, în caz contrar funcția CHAR returnează valoarea NULL;

2. expresie numerică UNICOD se evaluează la un întreg între 0 și 65535, în caz contrar funcția **NCHAR** returnează valoarea NULL;
3. expresie str(1,2,3) pot fi de tip caracter sau binar; funcția **REPLACE** întoarce o valoare de tip binar dacă cel puțin una dintre argumentele sale este de tip binar.

#### Exemplu:

```

SELECT ASCII('A')           - rezultat 65
SELECT CHAR(65)             - rezultat 'A'

SELECT UNICODE('A')          - rezultat 197
SELECT NCHAR(197)            - rezultat 'A'

SELECT LEFT('ananas', 3)     - rezultat 'ana'
SELECT RIGHT('ananas', 3)    - rezultat 'nas'
SELECT SUBSTRING('ananas', 3, 3) - rezultat 'ana'

SELECT STR(123.456, 7, 3)    - rezultat '123.456'
SELECT STUFF('mere', 1, 1, 'p') - rezultat 'pere'
SELECT STUFF('mere', 2, 1, 'u') - rezultat 'mure'

SELECT LEN('struguri')      - rezultat 8
SELECT LOWER('ABCDEFGH')     - rezultat 'abcdefghijklm'
SELECT UPPER('abcdefghijklm') - rezultat 'ABCDEFGHIJKLM'

SELECT LTRIM('abc')          - rezultat 'abc'
SELECT RTRIM('abc')          - rezultat 'abc'

SELECT PATINDEX('%abc%', 'abcabcabc') - rezultat 1
SELECT PATINDEX('abc%', 'abcabcabc') - rezultat 1
SELECT PATINDEX('%abc', 'abcabcabc') - rezultat 7

```

```

SELECT PATINDEX('abc', 'abcabcabc') - rezultat 0
SELECT REPLACE('abcabcabc', 'abc', 'a') - rezultat 'aaa'
SELECT QUOTENAME('abc def') - rezultat [abc def]
SELECT REPLICATE('abc', 3) - rezultat 'abcabcabc'
SELECT REPLICATE(' ', 3) - rezultat ' '
SELECT SPACE(3) - rezultat ' '
SELECT REVERSE('abc') - rezultat 'cba'

```

### 2.2.1.3. Funcții pentru manipularea valorilor de tip dată calendaristică

ACESTE FUNCȚII scalare acționează asupra unor valori de tip dată calendaristică și returnează o valoare care poate fi de tip: sir de caractere, numeric sau dată calendaristică.

FUNCȚIILE pentru manipularea valorilor de tip dată calendaristică împreună cu sintaxa acestora și valoarea returnată sunt prezentate mai jos:

Sintaxă	Semnificație
DATEADD(element_dată, număr, dată)	data rezultată prin adăugarea la data calendaristică specificată a unui interval de timp specificat prin parametrii element dată și număr.
DATEDIFF(element_dată, dată_inicială, dată_finală)	calculează diferența dintre datele calendaristice dată_inicială și dată_finală exprimată în intervale de timp de tipul element_dată;
DATENAME(element_dată, dată)	numele elementului de dată specificat prin parametrul element_dată;
DATEPART(element_dată, dată)	returnează un întreg reprezentând valoarea zilei elementului de dată specificat prin parametrul element_dată;
DAY(dată)	returnează un întreg reprezentând valoarea lunii din lună a datei calendaristice furnizată ca parametru;
GETDATE()	returnează data curentă din sistem în format standard;
MONTH(dată)	returnează un întreg reprezentând valoarea lunii din an a datei calendaristice furnizată ca parametru;
YEAR(dată)	returnează un întreg reprezentând valoarea anului pentru data calendaristică furnizată ca parametru.

## Observații:

1. *element\_dată* este un parametru care specifică o anumită parte a datei calendaristice. Tabelul de mai jos indică părțile unei date calendaristice împreună cu abrevierile ce le corespund:

Partea datei	Abreviere
anul	yy sau yyyy
trimestrul	qq sau q
luna	mm sau m
ziua din an	dy sau Y
ziua din lună	dd sau d
ziua din săptămână	wd
săptămâna	dw
ora	hh
minutul	mi sau n
secunda	ss sau s
millisecunda	ns

2. *dată\_initială și dată\_finală* sunt expresii care se evaluatează la valori de tip dată calendaristică (*datetime* sau *smalldatetime*);

Exemple:

```

SELECT DATEADD(YY, 1, '01/21/2001')           - rezultat 2002-01-21 00:00:00.000
SELECT DATEADD(ss, 1, '01/21/2001')            - rezultat 2001-01-21 00:00:01.000

SELECT DATEDIFF(YY, '01/01/2000', '01/01/2001') - rezultat 1
SELECT DATEDIFF(QQ, '01/01/2000', '01/01/2001') - rezultat 4
SELECT DATEDIFF(mm, '01/01/2000', '01/01/2001') - rezultat 12
SELECT DATEDIFF(dy, '01/01/2000', '01/01/2001') - rezultat 366
SELECT DATEDIFF(hh, '01/01/2000', '01/01/2001') - rezultat 8784
...
SELECT DATEDIFF(day, '01/01/2000', getdate())    - numărul de zile
                                                    - curse de la 1 ianuarie 2000 și
                                                    - până la data curentă;

SELECT DATENAME(mm, '01/22/2001')
SELECT DATENAME(dw, '01/22/2001')

SELECT DATEPART(mm, '01/22/2001')
SELECT DATEPART(dw, '01/22/2001')

SELECT DAY('01/22/2001')
SELECT MONTH('01/22/2001')
SELECT YEAR('01/22/2001')

```

## 2.2.1.4. Funcții sistem (conversii)

Dintre funcțiile sistem de tip scalar prezentă în acest paragraf doar funcțiile de conversie. Funcțiile CASE și cele pentru tratarea valorilor NULL vor fi prezentate ulterior.

În SQL Server, sunt posibile două niveluri de conversii între tipurile de date:

- Când data dintr-un obiect este mutată, comparată sau combinată cu o dată dintr-un alt obiect - data poate fi convertită din tipul de date al unui obiect în tipul de date al celuilalt obiect.
- Când data dintr-o coloană rezultată dintr-o tranzacție SQL, codul returnat sau parametrul de ieșire al unei proceduri stocate este mutat într-o variabilă program, acesta trebuie convertit la tipul de date al variabilei.

Există două categorii de conversii:

- Conversiile implicite care sunt transparente pentru utilizator. SQL Server convertește automat datele dint-un tip în altul ori de către ori este posibil. De exemplu dacă **smallint** este comparat cu un **int**, atunci **smallint** este convertit implicit la **int** înaintea comparației.
- Conversiile explicate sunt realizate de către utilizator prin folosirea funcțiilor CAST sau CONVERT.

### Funcții CAST și CONVERT

Realizează conversia explicită a unei expresii de un anumit tip în alt tip. CAST și CONVERT sunt similare în funcționare, fiind diferite ca sintaxă. Funcția CAST este compatibilă cu standardul SQL'92 și este preferată (acolo unde este posibil) față de CONVERT care este specifică SQL Server.

Sintaxa:

Utilizând CAST:

```

CAST(expresie AS tip_data)

```

Utilizând CONVERT:

```

CONVERT(tip_data [(lungime)], expresie [,stil])

```

unde:

*expresie* - orice expresie SQL Server validă.

*tip\_data* - tipul de date întărit furnizat de sistem. Tipurile de date utilizator nu pot fi folosite în conversii.

*lungime* - parametru optional al tipurilor de date **nchar**, **nvarchar**, **char**, **varchar**, **binary** sau **varbinary**.

*stil* - este stilul formatului de dată calendaristică folosit la conversia **datetime** sau **smalldatetime** la un tip caracter (tipurile **nchar**, **nvarchar**, **char**, **varchar**, **nchar**, sau **nvarchar**), sau formatul unui sir când se convertește **float**, **real**, **money** sau **smallmoney** la un tip caracter.

În tabela de mai jos primele două coloane din stânga reprezintă valorile pe care le poate avea parametrul *stil* în cazul conversiei de la **datetime** sau **smalldatetime** la un tip caracter.

Fără secol (YY)	Cu secol (YYYY)	Cu secol (YY)	Standard	Intrare/iesire**
-	0 or 100 (*)	Implicit	mon dd yyyy hh:miAM (PM)	mon dd/yy
1	101	USA	mm/dd/yy	yy-mm-dd
2	102	ANSI	dd/mm/yy	dd/mm/yy
3	103	British/French	dd.mm.yy	yy-mm-dd
4	104	German	dd.mm.yyyy	dd-mm-yy
5	105	Italian	-	dd mon yy
6	106	-	-	mon dd,yy
7	107	-	-	hh:nn:ss
8	108	-	-	mon dd yyyy hh:mi:ssAM (PM)
-	9 or 109 (*)	Implicit + milisecunde	mon dd yyyy hh:mi:ss.mmmAM (PM)	mon-dd-yy
10	110	USA	yy/mm/dd	yy/mm/dd
11	111	JAPAN	yy/mm/dd	yyymmdd
12	112	ISO	yyymmdd	yy-mm-dd
-	13, 113 (*)	Europa implicit + milisecunde	dd mon yyyy hh:mm:ss.mmm(24h)	hh:mi:ss.mmm(24h)
14	114	-	-	yyyy-mm-dd hh:mi:ss(24h)
-	20, 120 (*)	ODBC canonic	yyyy-mm-dd hh:mi:ss.mmm(24h)	yyyy-mm-dd hh:mi:ss.mmm(24h)
-	21, 121 (*)	ODBC canonic +milisecunde	yyyy-mm-dd hh:mi:ss.mmm(24h)	yyyy-mm-dd hh:mi:ss.mmm(24h)

\* Valorile implicite (*stil* 0 sau 100, 9 sau 109, 13 sau 113, 20 sau 120 și 21 sau 121) returnează întotdeauna secolul (yyyy).

\*\* Intrare când se convertește la **datetime**; iesire când se convertește la o dată de tip caracter.

2. Când se convertește un **smalldatetime** la un tip caracter, stilurile ce includ secunde sau milisecunde vor avea zero în aceste pozitii.

Tabelul următor indică natura conversiei din **float** sau **real** la un tip caracter în funcție de parametrul *stil*:

Valoare	Iesire
0 (implicit)	6 cifre maxim. Utilizată în notatiile sătunifice, când corespunde.
1	Intotdeauna 8 cifre. Intotdeauna folosit în notatiile sătunifice.
2	Intotdeauna 16 cifre. Intotdeauna folosit în notatiile sătunifice.

În tabelul următor coloana din stânga reprezintă valorile parametrului *stil* în cazul conversiei **money** sau **smallmoney** la un tip caracter:

Valoare	Iesire
0 (implicit)	Fără virgulă la fiecare trei cifre din stânga punctului zecimal și cu două cifre la dreapta punctului zecimal. De ex., 4235,98
1	Virgula după fiecare trei cifre din stânga punctului zecimal și cu două cifre după punctul zecimal. De exemplu, 3,510,92.
2	Fără virgulă la fiecare trei cifre din stânga punctului zecimal și patru cifre la dreapta punctului zecimal. De ex., 4,235,9819.

Observații:

- Conversiile implicate nu sunt suportate de către tipurile text și **image**. Se pot însă convertești explicit date de tip text la tipul caracter și date de tip **image** la **binary** sau **varbinary**, dar lungimea maximă care poate fi specificată este 8000.
- Dacă se încearcă o conversie care nu este posibilă (de exemplu o expresie de tip caracter care include litera la **int**), SQL Server generează un mesaj de eroare.
- La conversia expresiilor de tip caracter sau binar (**char**, **nchar**, **nvarchar**, **varchar**, **binary** sau **varbinary**) la o expresie de un tip diferit, data poate fi trunchiată, tipărîță parțial sau poate fi returnată o eroare.

Exemple:

```
SELECT CAST('123' AS char(3))           - rezultat '123'
SELECT CONVERT(char(3) ,123)             - rezultat '123'
SELECT CAST('123' AS char(2))           - rezultat *
SELECT CAST('123' AS char(4))           - rezultat '123'
SELECT CAST('123' AS varchar(4))         - rezultat '123'

SELECT CAST(GETDATE() AS varchar(12))      - rezultat 'Jul 13 2001'
SELECT CONVERT(varchar(12) ,GETDATE(0 ,3))   - rezultat '13/07/01'
SELECT CONVERT(varchar(12) ,GETDATE(0 ,103)) - rezultat '13/07/2001'
```

1. Implicit, SQL Server interpretează cele două cifre din an considerând anul 2049 ca an de referință. Aceasta înseamnă că anul 49 e interpretat ca 2049, iar anul 50 e interpretat ca 1950.

2. Când se convertește un **smalldatetime** la un tip caracter, stilurile ce includ secunde sau milisecunde vor avea zero în aceste pozitii.

Observații:

- Implicit, SQL Server interpretează cele două cifre din an considerând anul 2049 ca an de referință. Aceasta înseamnă că anul 49 e interpretat ca 2049, iar anul 50 e interpretat ca 1950.

## 2.4. Exerciții și probleme

1. Să se indice rezultatul returnat de următoarele operații SELECT:

```
SELECT POWER(1,0,5)
SELECT SQRT(9)

SELECT ROUND(748,-1)
SELECT ROUND(48,58,1)
SELECT ROUND(78,34,1)
SELECT ROUND(701,-2)
SELECT ROUND(12,12,2)

SELECT CHARINDEX('aa','aaaaaaaa')
SELECT CHARINDEX('aa','aaaaaaaa',4)

SELECT LEFT('abcdef',2)
SELECT RIGHT('abcdef',4)
SELECT SUBSTRING('abcdef',2,3)

SELECT REPLACE('abcaabcabc','a','abc')

SELECT CAST('123' AS char(2))

SELECT CAST('07/13/01' AS datetime)
SELECT CONVERT(varchar(12),CAST('07/06/01' AS datetime),1)
SELECT CONVERT(varchar(12),CAST('07/06/01' AS datetime),101)
SELECT CONVERT(datetime,'07/06/01',1),6)
SELECT CONVERT(varchar(12),CONVERT(datetime,'07/06/01',1),6)
SELECT CONVERT(varchar(12),CONVERT(datetime,'07/06/01',3),6)
```

## 3. TABELE ȘI INDECȘI

1. Să se indice rezultatul returnat de următoarele operații SELECT:

Tabelele sunt cele mai importante obiecte dintr-o bază de date relatională. Toate datele sunt stocate în tabele organizate sub formă de linii și coloane. Fiecare tabelă corespunde unei relații din schema relatională a bazei de date. Linile tabelei corespund tupelor relației, iar coloanele corespund atribuțiilor acestia.

Proiecțarea unei baze de date presupune, în primul rând, stabilirea colecției de tabele și a descrierii tabelelor împreună cu toate proprietățile asociate. Înainte de crearea unei tabele noi trebuie luate o serie de decizii de proiectare referitoare la:

- tipurile de date care vor fi stocate în tabele;
- coloanele tabelei și tipul de date (eventual lungimea) pentru fiecare coloană;
- coloanele care (nu)acceptă valori NULL;
- constrângeri, reguli, valori implicate;
- indecși necesari, coloanele care formează cheia primară ori care sunt parte a unei chei străine.

Odată proiecția baza de date se poate trece la crearea tabelelor în care vor fi stocate datele. Stocarea datelor se face în tabelele permanente care sunt grupate în una sau mai multe fișiere ale bazei de date. Pe lângă tabelele permanente, în SQL Server, există posibilitatea de crea tabele temporare. Acestea sunt similare tabelelor permanente, cu excepția faptului că sunt stocate în baza de date sistem *tempdb* de unde sunt șterse în mod automat de îndată ce nu mai sunt folosite. Tabelele temporare pot fi locale sau globale. Numele unei tabele temporare locale începe cu simbolul #, ea este vizibilă numai pentru utilizatorul curent și este distrusă când acesta se deconectează de la SQL Server. Numele tabelelor temporare globale încep cu ##, sunt vizibile pentru toți utilizatorii (care au drepturi corespunzătoare!) și sunt păstrate în sistem până la deconectarea ultimului utilizator care le folosește.

### 3.1. Instrucțiunea CREATE TABLE

Creează o nouă tabelă.

Sintaxa (simplificată):

```
CREATE TABLE [nume_baza_date].[proprietar].[proprietar] nume_tabel
( { <definiri_coloana> |<constrainte_tabela> } [,n ] )
```