



Бази даних та інформаційні системи

Лекція 8. Функції для одного рядка

СумДУ, каф. КН
2020

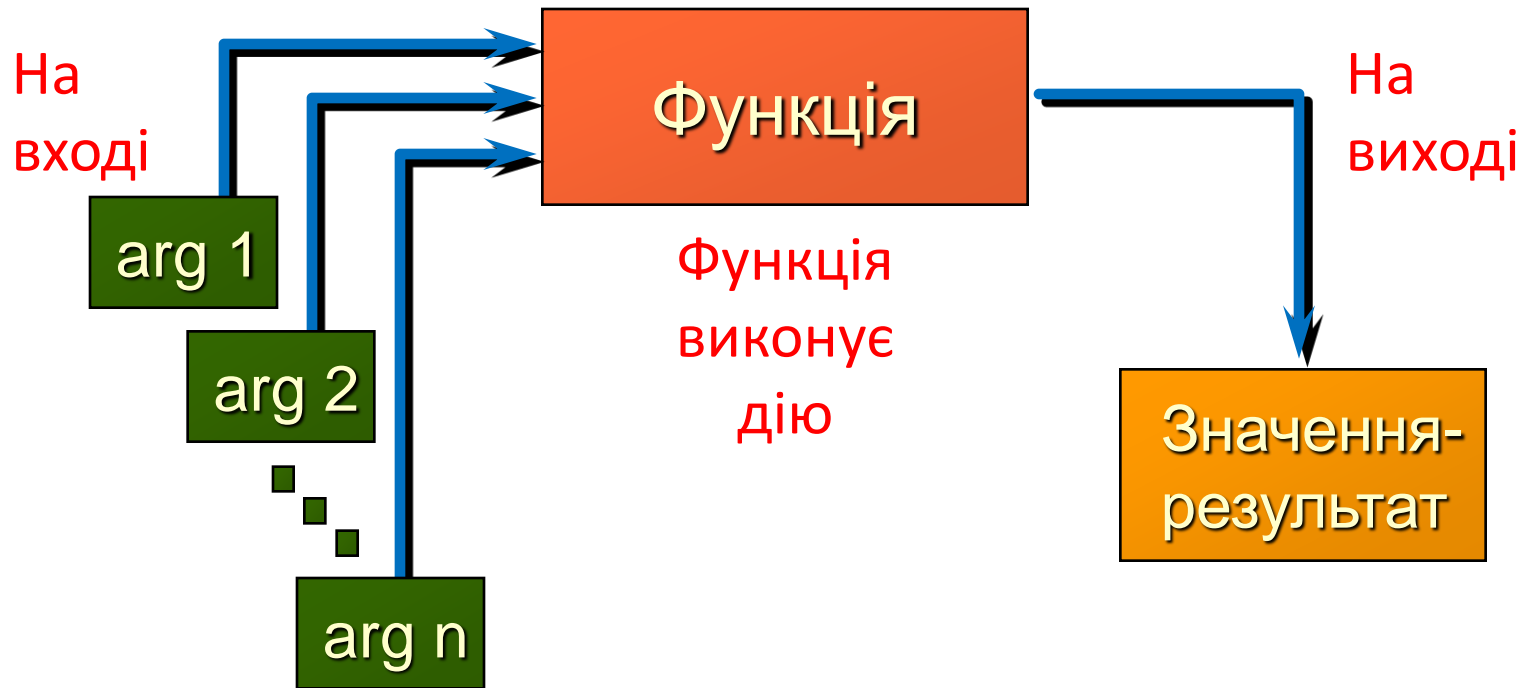
Задачі заняття

Після завершення заняття ви маєте вміти і знати наступне:

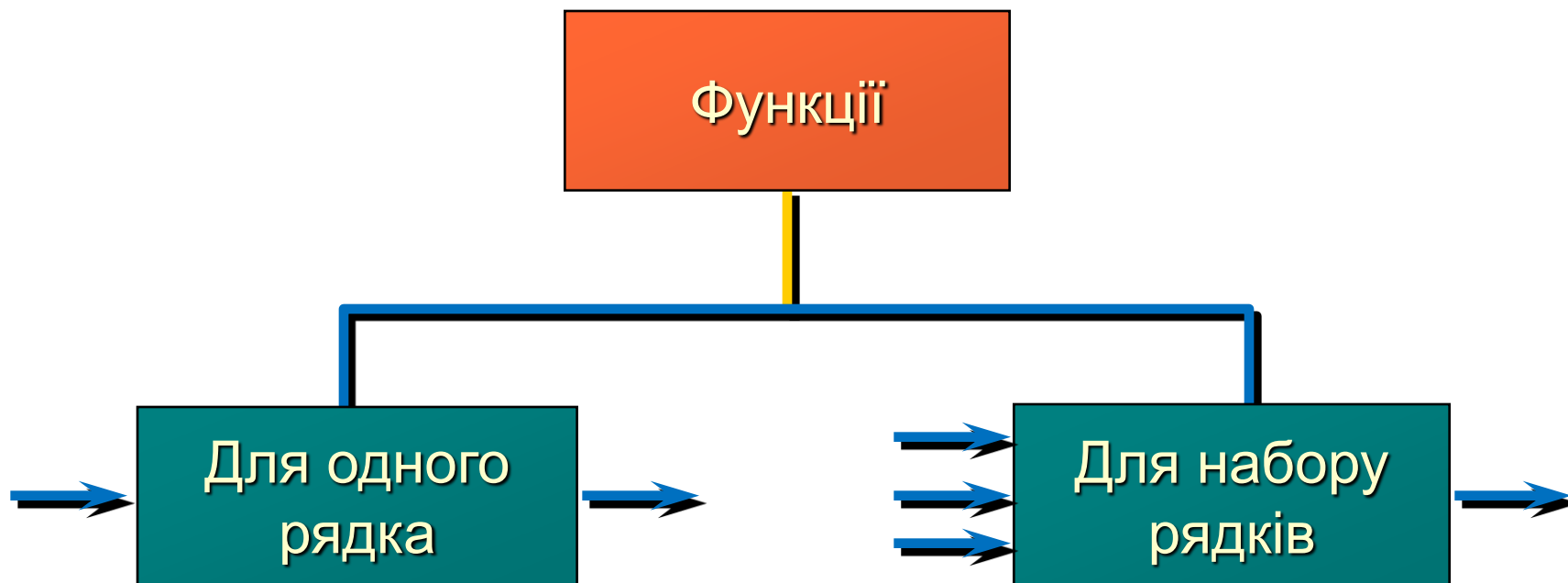
- ▶ Розрізняти види функцій, що використовуються в SQL;
- ▶ Використовувати функції роботи з символьними, числовими і темпоральними даними в виразах SELECT;
- ▶ Використовувати функції перетворення типів.



Функції SQL



Два види функцій SQL



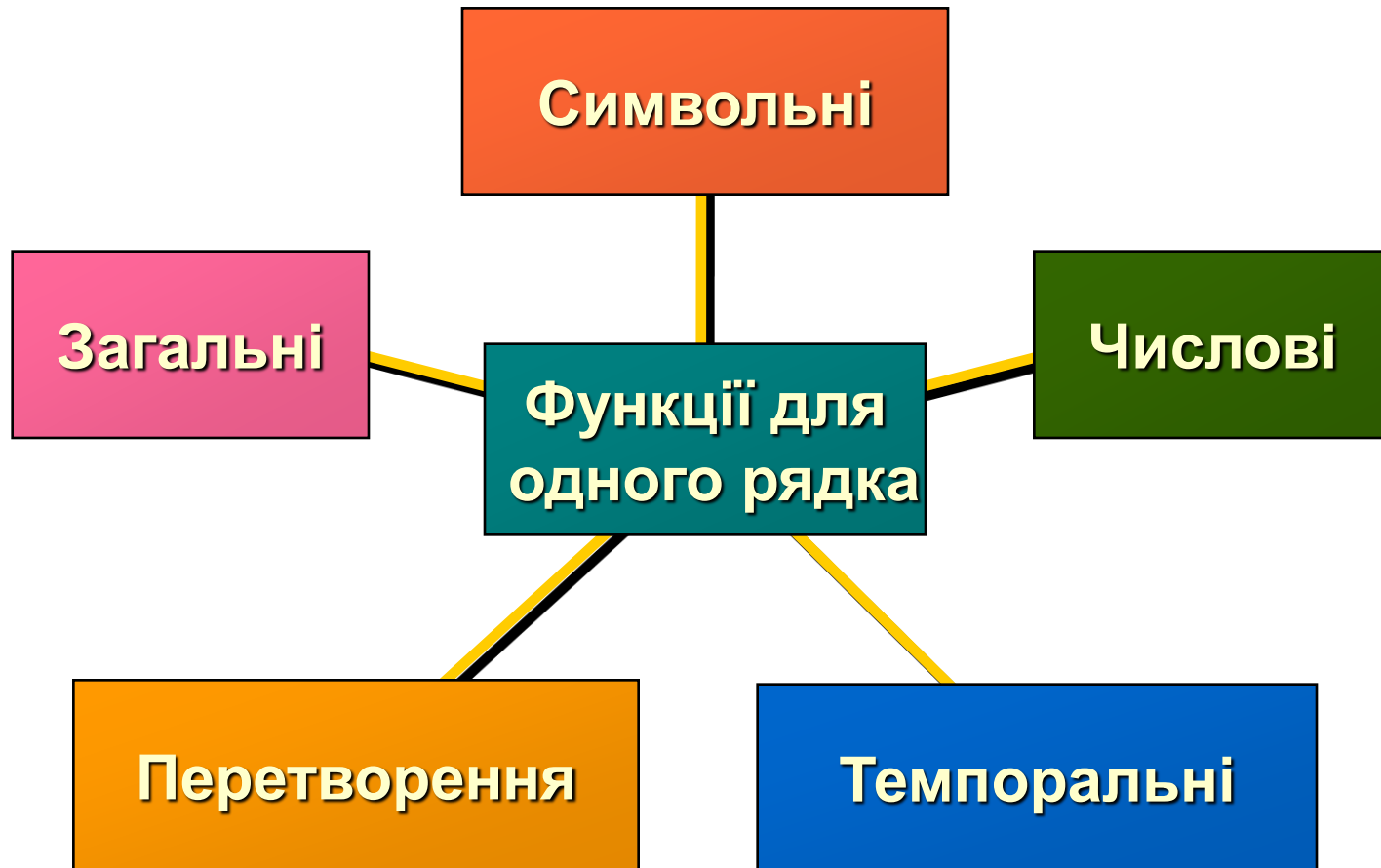
Функції для одного рядка

- ▶ Маніпулюють даними;
- ▶ Допускають аргументи і повертають одне значення;
- ▶ Діють на кожен повернений рядок;
- ▶ Повертають одне значення для одного рядка;
- ▶ Можуть змінювати тип даних;
- ▶ Можуть бути вкладеними.

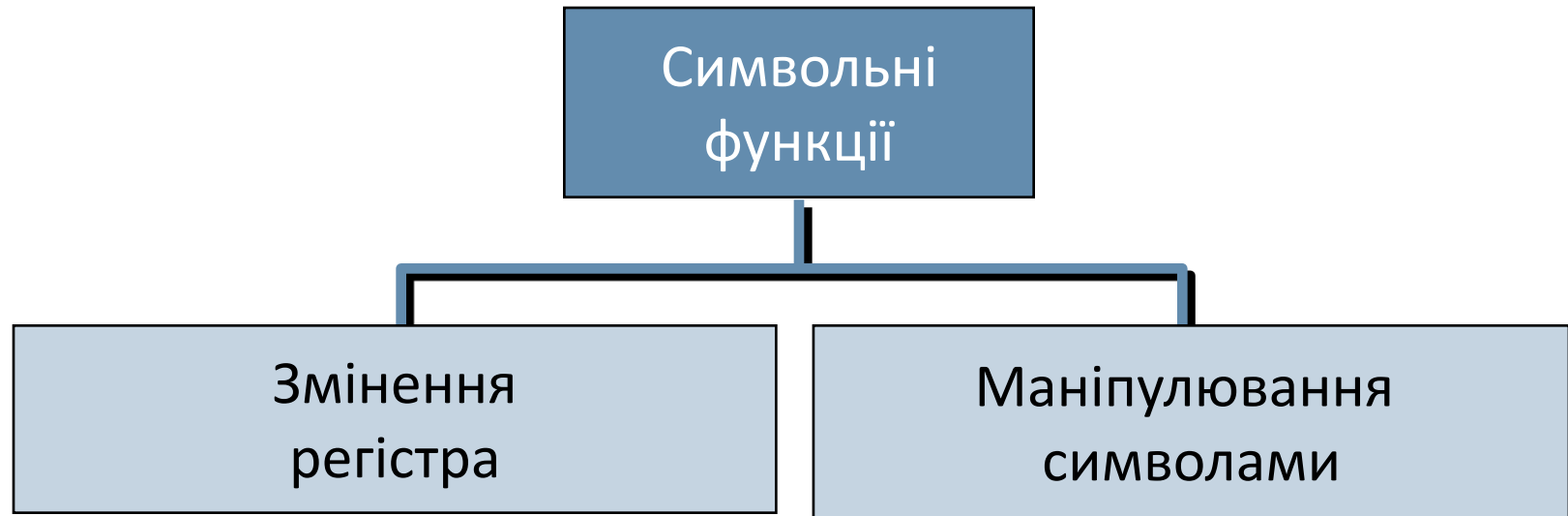
```
function_name (column|expression, [arg1, arg2, ...])
```



Функції для одного рядка



Символьні функції



LOWER
UPPER
INITCAP

CONCAT
SUBSTR
LENGTH
INSTR
LPAD
RPAD



Функції зміни регістру

Функція	Результат
LOWER(' SQL Course ')	sql course
UPPER(' SQL Course ')	SQL COURSE
INITCAP(' SQL course ')	Sql Course



Використання функцій зміни регістру

Виведення номера службовця, імені та номера департаменту для службовця Blake.

```
SQL> SELECT empno, ename, deptno
2 FROM emp
3 WHERE ename = 'blake';
no rows selected
```

```
SQL> SELECT empno, ename, deptno
2 FROM emp
3 WHERE LOWER(ename) = 'blake';
```

EMPNO	ENAME	DEPTNO
7698	BLAKE	30

Функції маніпулювання символами

Маніпулюють символьними рядками

Функція	Результат
CONCAT('Good', 'String')	GoodString
SUBSTR('String', 1, 3)	Str
LENGTH('String')	6
INSTR('String', 'r')	3
LPAD(sal, 10, '*')	*****5000
RPAD(sal, 10, '*')	5000*****
TRIM(' KING ')	KING



Використання функцій маніпулювання символами

```
SQL> SELECT ename, CONCAT (ename, job), LENGTH (ename),  
2 INSTR (ename, 'A')  
3 FROM emp  
4 WHERE SUBSTR (job, 1, 5) = 'SALES';
```

ENAME	CONCAT (ENAME, JOB)	LENGTH (ENAME)	INSTR (ENAME, 'A')
-----	-----	-----	-----
MARTIN	MARTINSALESMAN	6	2
ALLEN	ALLENSALESMAN	5	1
TURNER	TURNERSALESMAN	6	0
WARD	WARDSALESMAN	4	2

INSTR

INSTR повертає **n**-е входження підрядка у рядок.

```
INSTR (string, substring [, start_position [, nth_appearance]])
```

nth_appearance є **n**-м входженням підрядку.

<pre>SELECT loc, INSTR(loc, 'O'), INSTR(loc, 'O', 1, 2), INSTR(loc, 'O', -1, 2) FROM dept;</pre>			
LOC	<pre>INSTR(LOC, 'O')</pre>	<pre>INSTR(LOC, 'O', 1, 2)</pre>	<pre>INSTR(LOC, 'O', -1, 2)</pre>
NEW_YORK	6	0	0
DALLAS	0	0	0
CHICAGO	7	0	0
BOSTON	2	5	2
HONKONG	2	5	2
HONKONG	2	5	2
NEW_YORK	6	0	0

SUBSTR

Виділення підрядка з рядку.

```
SUBSTR( string, start_position, [ length ] )
```

Якщо start_position є негативним числом, то функція SUBSTR починає з кінця рядка і рахує в зворотному напрямку.

```
SELECT SUBSTR('This is a test', 6, 2)    a,  
       SUBSTR('This is a test', 6)      b,  
       SUBSTR('This is a test', -9, 4)   c  
FROM DUAL;
```

A	B	C
--	-----	----
is	is a test	is a

LPAD

- ▶ **LPAD** додає з лівої частини рядка певний набір символів (коли `string1` не `null`).

```
LPAD(string1, padded_length, [ pad_string ])
```

Повертає string значення.

```
SQL> SELECT LPAD('lpad', 8, '0') FROM DUAL;
```

--Результат: 0000lpad



Функція TRIM

- ▶ Видаляє всі зазначені символи з початку або кінця рядка.

```
TRIM([ [LEADING | TRAILING | BOTH] trim_character FROM] string1)
```

(з початку, з кінця, з обох сторін символного рядка)

- ▶ Якщо не визначим параметри, функція **TRIM** видалить пробіли з початку і з кінця рядка.

```
select trim(loc) FROM dept;
```



Використання функції TRIM


```
SELECT TRIM('  KING  ') ex_1,  
       TRIM(' ' FROM ' KING ') ex_2,  
       TRIM(LEADING '0' FROM '000KING123') ex_3,  
       TRIM(TRAILING '1' FROM 'KING1') ex_4,  
       TRIM(BOTH '1' FROM '123KING111') ex_5  
FROM DUAL;
```

EX_1	EX_2	EX_3	EX_4	EX_5
----	----	----	----	----
KING	KING	KING123	KING	23KING




Числові функції


- **ROUND** - округлює значення до зазначеного виду

ROUND (45.926, 2)  **45.93**


- **TRUNC** - відсікає значення до зазначеного виду

TRUNC (45.926, 2)  **45.92**

- **MOD** - повертає залишок від ділення

MOD (1600, 300)  **100**

- **ABS** - повертає абсолютне значення числа

▶ **ABS (-23.6)**  **23,6**

Використання функції ROUND

```
SQL> SELECT ROUND (45.923,2) , ROUND (45.923,0) ,  
2          ROUND (45.923,-1)  
3 FROM DUAL;
```

ROUND (45.923,2)	ROUND (45.923,0)	ROUND (45.923,-1)
----- 45.92	----- 46	----- 50



Використання функції TRUNC

```
SQL> SELECT TRUNC (45.923, 2) , TRUNC (45.923) ,  
2          TRUNC (45.923, -1)  
3 FROM DUAL;
```

TRUNC (45.923, 2)	TRUNC (45.923)	TRUNC (45.923, -1)	
-----	-----	-----	
45.92	45	40	



Використання функції MOD

Обчислення залишку від ділення зарплати на комісійні для всіх працівників, які працюють продавцями.

```
SQL> SELECT  ename, sal, comm, MOD(sal, comm)
2  FROM      emp
3  WHERE      job = 'SALESMAN';
```

ENAME	SAL	COMM	MOD (SAL, COMM)
MARTIN	1250	1400	1250
ALLEN	1600	300	100
TURNER	1500	0	1500
WARD	1500	50.5	35.5

Робота з датами

- ▶ Oracle зберігає дати у внутрішньому числовому форматі: століття, рік, місяць, день, година, хвилина, секунда.
- ▶ Формат дати за замовчуванням має вигляд DD-MON-YY.
- ▶ Функція SYSDATE повертає дату і час.



Арифметика з датами

- ▶ Результатом додавання числа до дати або віднімання числа від дати є дата.
- ▶ Віднімання однієї дати з іншого має результатом число днів між цими датами.
- ▶ Для додавання годин до дати розділіть їх кількість на 24.



Використання арифметичних операторів при роботі з датами

Скільки тижнів відпрацювали співробітники департаменту 10?

```
SQL> SELECT  ename, (SYSDATE-hiredate)/7 WEEKS  
2    FROM    emp  
3    WHERE    deptno = 10;
```

ENAME	WEEKS
-----	-----
KING	830.93709
CLARK	853.93709
MILLER	821.36566



Функції для дат

Функція	Опис
MONTH_BETWEEN	Кількість місяців між двома датами
ADD_MONTHS	Додавання календарних місяців до дати
NEXT_DAY	Наступний день відносно дати
LAST_DAY	Останній день місяця
ROUND	Округлює дату
TRUNC	Округлює дату в меншу сторону
EXTRACT	Вилучає значення з дати або значення інтервалу



Використання функцій для дат

MONTHS_BETWEEN ('13-FEB-2020','21-MAY-2020')

➡ -3,2580645

ADD_MONTHS ('13-FEB-2020',6)

➡ '13-AUG-20'

NEXT_DAY ('13-FEB-2020', 'monday')

➡ '17-FEB-20'


LAST_DAY('13-FEB-2020')


➡ '29-FEB-20'



Використання функцій для дат

EXTRACT (**YEAR** FROM DATE '2019-08-22')  2019

EXTRACT (**MONTH** FROM DATE '2019-08-22')  8

EXTRACT (**DAY** FROM DATE '2019-08-22')  22

Ви можете отримати тільки **YEAR**, **MONTH**, і **DAY** з дати

```
SELECT ename, EXTRACT (YEAR FROM hiredate) years  
FROM emp  
ORDER BY years DESC;
```

ENAME	YEARS
ADAMS	2013
SCOTT	2012
MILLER	2012
JONES	2011
MARTIN	2011
ALLEN	2011
TURNER	2011
JAMES	2011

Використання функцій для дат

```
ROUND( date, [ format ] )
```

ROUND('25-JUL-12','MONTH') → 01-AUG-12

ROUND('25-JUL-12','YEAR') → 01-JAN-13

```
TRUNC ( date, [ format ] )
```

TRUNC('25-JUL-12','MONTH') → 01-JUL-12

TRUNC('25-JUL-12','YEAR') → 01-JAN-12



Функції перетворення типів даних

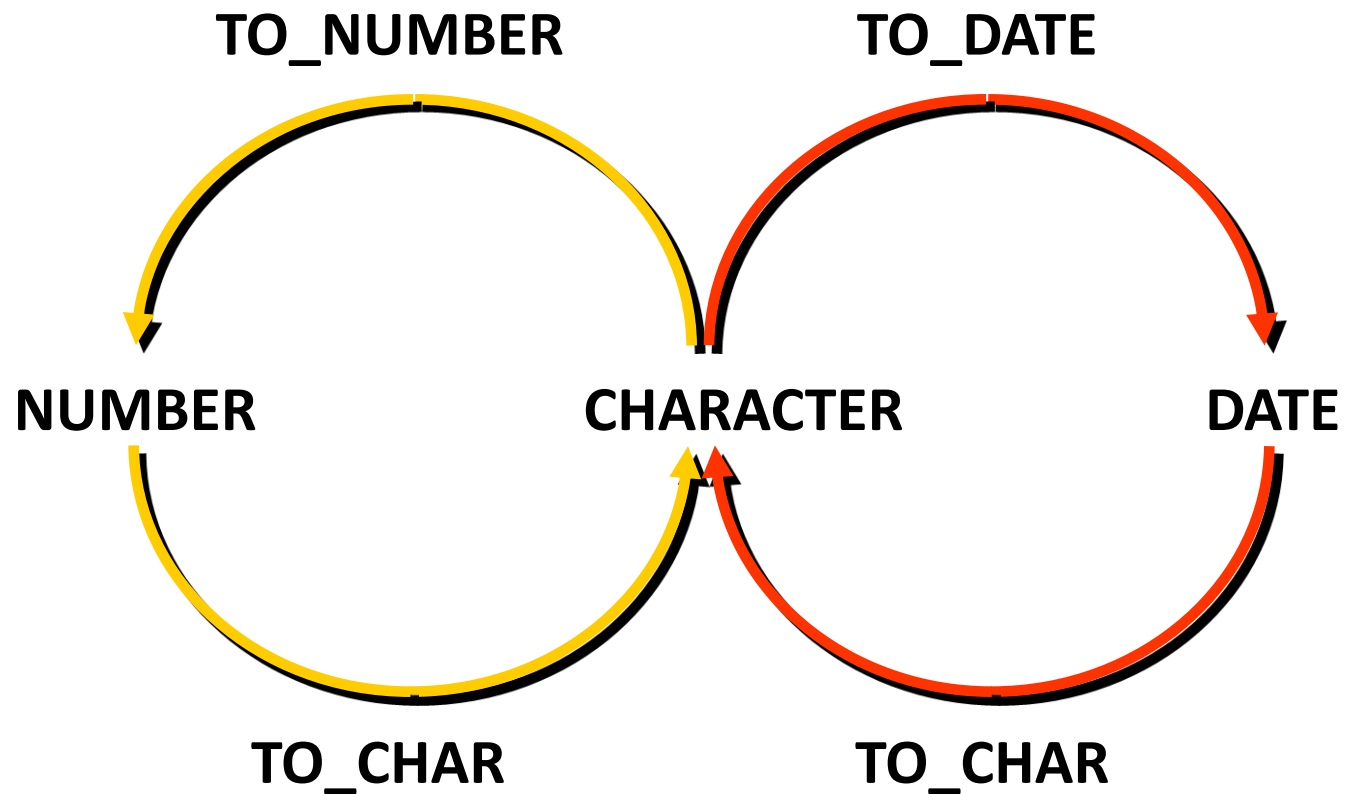


Неявне приведення типів

Сервер Oracle може автоматично виконати неявне перетворення типу даних у виразі.

	CHAR	VARCHAR2	NCHAR	NVARCHAR2	DATE	DATETIME/ INTERVAL	NUMBER	BINARY_FLOAT	BINARY_DOUBLE	LONG	RAW	ROWID	CLOB	BLOB	NCLOB
CHAR	-	X	X	X	X	X	X	X	X	X	X	-	X	X	X
VARCHAR2	X	-	X	X	X	X	X	X	X	X	X	X	X	-	X
NCHAR	X	X	-	X	X	X	X	X	X	X	X	X	X	-	X
NVARCHAR2	X	X	X	-	X	X	X	X	X	X	X	X	X	-	X
DATE	X	X	X	X	-	-	-	-	-	-	-	-	-	-	-
DATETIME/ INTERVAL	X	X	X	X	-	-	-	-	-	X	-	-	-	-	-
NUMBER	X	X	X	X	-	-	-	X	X	-	-	-	-	-	-
BINARY_FLOAT	X	X	X	X	-	-	X	-	X	-	-	-	-	-	-
BINARY_DOUBLE	X	X	X	X	-	-	X	X	-	-	-	-	-	-	-
LONG	X	X	X	X	-	X ¹	-	-	-	-	X	-	X	-	X
RAW	X	X	X	X	-	-	-	-	-	X	-	-	-	X	-
ROWID	-	X	X	X	-	-	-	-	-	-	-	-	-	-	-
CLOB	X	X	X	X	-	-	-	-	-	X	-	-	-	-	X
BLOB	-	-	-	-	-	-	-	-	-	-	X	-	-	-	-
NCLOB	X	X	X	X	-	-	-	-	-	X	-	-	X	-	-

Явне приведення типів



Функція TO_CHAR для дат

```
TO_CHAR(date, 'fmt') 
```

Вимоги до формату:

- ▶ Укладається в одинарні лапки;
- ▶ Чутливий до регістру;
- ▶ Може включати коректні елементи форматування;
- ▶ Має елемент **fm**, щоб видалити додані пробіли або прибрати початкові нулі;
- ▶ Відокремлений комою від дати.



Елементи формату дати

ФОРМАТ	ЗНАЧЕННЯ
YYYY	Записаний цифрами рік
YEAR	Рядковий запис року
MM	Номер місяця (2 цифри)
MONTH	Повна назва місяця
DD	День місяця (1 - 31)
MON	Скорочена назва місяця
DDD	День року (1 - 366)
D	День тижня (1-7)
DY	Скорочена назва дня тижня (3 символи)
DAY	День тижня



Елементи формату дати

ФОРМАТ	ЗНАЧЕННЯ
AM (або A.M.)	Показник часу до полудня
PM (або P.M.)	Показник часу після полудня
HH	Година дня (1-12)
HH24	Година дня (0-23)
MI	Хвилини (0-59)
SS	Секунди (0-59)



Елементи формату дати

- ▶ Виведення часу дня:

HH24:MI:SS AM	15:45:32 PM
----------------------	--------------------

- ▶ Рядки можна додати, уклавши їх у лапки:

DD "of" MONTH	12 of OCTOBER
----------------------	----------------------

- ▶ суфікси числівників:

ddspth	fourteenth
---------------	-------------------



Використання функції TO_CHAR для дати

```
SQL> SELECT ename,  
2          TO_CHAR(hiredate, 'fmDD Month YYYY') HIREDATE  
3 FROM      emp;
```

ENAME	HIREDATE
KING	17 November 1981
BLAKE	1 May 1981
CLARK	9 June 1981
JONES	2 April 1981
MARTIN	28 September 1981
ALLEN	20 February 1981
...	

14 rows selected.

Використання функції TO_CHAR для дати

Поточний час:

```
SELECT ( TO_CHAR(SYSDATE, 'HH24:MI:SS AM')) "CurrentTime"  
FROM DUAL;
```

CurrentTime
19:43:53 PM

Поточний день тижня:

```
SELECT ( TO_CHAR(SYSDATE, 'Day')) "CurrentDay" FROM DUAL;
```

CurrentDay
wednesday



Функція TO_CHAR для чисел

```
TO_CHAR(number, 'fmt') 
```

Використовуйте формати функції TO_CHAR для отримання символьного представлення чисел.

Елемент	Результат
9	являє цифру
0	виводить нуль
\$	Знак долара
L	Місцевий знак валюти
.	десяткова крапка
,	роздільник тисяч



Використання функції TO_CHAR з числами

```
SQL> SELECT TO_CHAR(sal, '$99,999') SALARY  
2 FROM emp  
3 WHERE ename = 'SCOTT';
```

SALARY

\$3,000



Функції TO_NUMBER і TO_DATE

Перетворення символьного рядка до числа проводиться за допомогою функції **TO_NUMBER**.

```
TO_NUMBER(char)
```

```
SQL> SELECT TO_NUMBER('123.4567') FROM DUAL;
```

Перетворення символьного рядка до дати проводиться за допомогою функції **TO_DATE**.

```
TO_DATE(char[, 'fmt'])
```

```
SQL> SELECT TO_DATE('072219', 'MMDDYY') FROM DUAL;
```

```
TO_DATE('
-----
22-JUL-19
```



Формат дати RR

Поточний рік	Вказана дата	Формат RR	Формат YY
2013	27.10.95	95	1995
2013	27.10.17	17	2017
2055	27.10.17	17	2117
2055	27.10.95	95	2095

		Вказаний двома цифрами рік	
		0-49	50-99
Дві цифри поточного року	0-49	Повертається дата поточного століття	Повертається дата попереднього століття
	50-99	Повертається дата наступного століття	Повертається дата поточного століття

Робота з **NULL**

Для роботи з **null**-значеннями існують функції:

- ▶ **NVL** (**expr1**, **expr2**)
- ▶ **NVL2** (**expr1**, **expr2**, **expr3**)
- ▶ **NULLIF** (**expr1**, **expr2**)
- ▶ **COALESCE** (**expr1**, **expr2**, ..., **exprn**)



Функція NVL

- ▶ Перетворює **null** в реальне значення.
- ▶ Може використовуватися для дат, символьних і числових даних.
- ▶ Вирази у функції **NVL** повинні мати однаковий тип даних.

NVL(comm, 0)

NVL(hiredate, '01-JAN-97')

NVL(job, 'No Job Yet')



Використання функції NVL

```
SQL> SELECT ename, sal, comm, (sal*12)+NVL(comm,0)
2 FROM emp;
```

ENAME	SAL	COMM	(SAL*12)+NVL(COMM,0)
KING	5000		60000
BLAKE	2850		34200
CLARK	2450		29400
JONES	2975		35700
MARTIN	1250	1400	16400
ALLEN	1600	300	19500
...			

14 rows selected.

Функція NVL2

```
NVL2(string1, value_if_NOT_null, value_if_null)
```

Дозволяє замінювати значення, коли зустрічається **null**-значення, а також коли зустрічається **НЕ null**-значення.

```
SELECT  ename, salary, comm,  
        NVL2(comm, 'SAL+COMM', 'SAL') income  
FROM    emp WHERE deptno IN (50, 80);
```

ENAME	SAL	COMM	INCOME
BLAKE	2850	(null)	SAL
MARTIN	1250	1400	SAL+COMM
ALLEN	1600	300	SAL+COMM
TURNER	1500	0	SAL+COMM
JAMES	950	(null)	SAL
WARD	1250	500	SAL+COMM

6 rows selected.

Функція NULLIF

```
NULLIF( expr1, expr2 )
```

Функція **NULLIF** порівнює `expr1` і `expr2`.
Якщо `expr1` і `expr2` рівні, повертає `NULL`.
В іншому випадку, вона повертає `expr1`.

```
SELECT  ename,  
        NULLIF(empno, mgr) result  
FROM    emp;
```



Функція COALESCE

```
COALESCE ( expr1, expr2, ..., expr_n )
```

- ▶ На відміну від **NVL** функція **COALESCE** може приймати безліч альтернативних значень.
- ▶ Функція **COALESCE** повертає перший аргумент, що не є **null**.
- ▶ Все вирази у функції **COALESCE** повинні мати однаковий тип даних.
- ▶ Еквівалентна умовному оператору **IF-THEN-ELSE**.



Використання COALESCE

```
SELECT ename, empno,  
       COALESCE (TO_CHAR(comm), TO_CHAR(mgr),  
                'No commission and no manager')  
FROM emp;
```

ENAME	EMPNO	EXAMPLE_COALESCE
KING	7839	No commission and no manager
BLAKE	7698	7839
CLARK	7782	7839
JONES	7566	7839
MARTIN	7654	1400
ALLEN	7499	300
TURNER	7844	0
JAMES	7900	7698
WARD	7521	500
FORD	7902	7566
SMITH	7369	7902
SCOTT	7788	7566
ADAMS	7876	7788
MILLER	7934	7782
JACKIE CHAN	8000	1700
JET LI	8001	600
BRUCE LEE	8002	8000
DR NO	8003	7839

18 rows selected.

CASE

Працює за принципом **IF-THEN-ELSE**

```
CASE expr WHEN search1 THEN return_expr_1  
    [ WHEN search2 THEN return_expr_2  
      ...  
      WHEN search3 THEN return_expr_N]  
    [ ELSE else_expr]  
END;
```

```
CASE WHEN condition_expr1 THEN return_expr_1  
    [ WHEN condition_expr2 THEN return_expr_2  
      ...  
      WHEN condition_exprN THEN return_expr_N]  
    [ ELSE else_expr]  
END;
```



Приклади

```
SELECT job, sal,  
        CASE job WHEN 'ANALYST' THEN SAL*1.1  
            WHEN 'CLERK' THEN SAL*1.15  
            WHEN 'MANAGER' THEN SAL*1.20  
            ELSE SAL END RR  
  
FROM emp;
```

```
SELECT job, sal,  
        CASE WHEN job='ANALYST' THEN SAL*1.1  
            WHEN job LIKE 'C%' THEN SAL*1.15  
            WHEN sal < 1000 THEN SAL*1.20  
            ELSE sal END RR  
  
FROM emp;
```



Функція **DECODE**

Полегшує виконання умовних запитів, діючи подібно виразам **CASE** або **IF-THEN-ELSE**

```
DECODE(col/expression, search1, result1  
      [, search2, result2, ...,]  
      [, default])
```



Використання функції DECODE

```
SQL> SELECT job, sal,  
2          DECODE(job, 'ANALYST', SAL*1.1,  
3                    'CLERK', SAL*1.15,  
4                    'MANAGER', SAL*1.20,  
5                    SAL)  
6          REVISED_SALARY  
7 FROM emp;
```

JOB	SAL	REVISED_SALARY
PRESIDENT	5000	5000
MANAGER	2850	3420
MANAGER	2450	2940
...		

14 rows selected.

Вкладені функції

- Рівень вкладеності функцій для одного рядка може бути довільним.
- Вкладені функції обчислюються починаючи з найглибшого рівня до самого зовнішнього.

```
F3 (F2 (F1 (col, arg1) , arg2) , arg3)
```

Step 1 = Result 1

Step 2 = Result 2

Step 3 = Result 3









Вкладені функції

```
SQL> SELECT  ename ,  
2          NVL (TO_CHAR (mgr) , 'No Manager' )  
3 FROM      emp  
4 WHERE     mgr IS NULL;
```

ENAME	NVL (TO_CHAR (MGR) , 'NOMANAGER')
-----	-----
KING	No Manager



Які з тверджень є справедливими для функцій одного рядка?

-  1. Приймають аргументи і повертають один результат для кожного аргументу.
-  2. Повертають один результат для кожного рядка.
-  3. Повертають один результат для групи рядків.
-  4. Не можуть змінювати тип даних.
-  5. Можуть бути вкладені одна в одну.
-  6. Аргументом може бути стовпець або вираз.



Висновки

Використовуйте функції для:

- ▶ виконання операцій над даними;
- ▶ модифікування окремих значень;
- ▶ управління виведенням даних для груп рядків;
- ▶ зміни форматів дат;
- ▶ перетворення типів даних стовпців.

