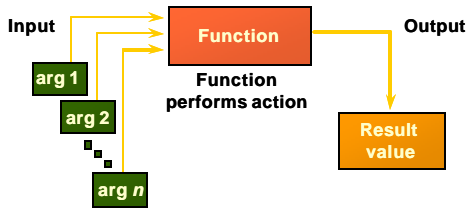


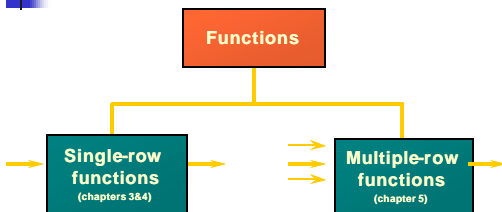
Chapter 3 Character and Number Functions

SQL Functions

- A predefined block of code that can accept inputs, perform processing, and return a result



Two Types of SQL Functions



Single-Row Functions

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

- Manipulate data
- Accept arguments and return one value
- Act on each row returned
- Return one result per row
- May modify the datatype
- Can be nested

4

Case Conversion Functions

- Convert case for character strings

Function	Result
LOWER('SQL Course')	
UPPER('SQL Course')	
INITCAP('SQL Course')	

- Egs:

```
SQL> SELECT ln, fn FROM writer WHERE ln = 'HALL'  
SQL> SELECT ln, fn FROM writer WHERE UPPER(ln) = 'HALL'
```

5

Character Manipulation Functions

Function	Result
LPAD(5000,10,'*')	
RPAD(5000,10, '+')	
LENGTH('String')	
SUBSTR('String', 3, 4)	
INSTR('String', 'r')	
INSTR('Mississippi River', 'i', 3, 4)	
CONCAT('Good', 'String')	

6

Character Function Examples

```
SQL> SELECT LPAD(ln, 20, ' ') FROM writer;

SQL> SELECT RPAD(ln, 20, '.') FROM writer;

SQL> SELECT title, LENGTH(title) FROM article;

SQL> SELECT phone, SUBSTR(phone, 2, 3) FROM writer;

SQL> SELECT title, issue
FROM article
WHERE INSTR(issue, 'MAY') <> 0;

SQL> SELECT cardnum, SUBSTR(cardnum, -4) last4,
      expires, limit
FROM ttollen.creditcard;
```

7

Character Manipulation Functions

Function	Result
LTRIM(' Tom')	
RTRIM('Tom ')	
RTRIM('852220000', '0')	
REPLACE(street_address, 'St.', 'Street')	

```
SQL> SELECT street_address
FROM student
WHERE INSTR(street_address, 'St.') > 0;

SQL> SELECT street_address,
      REPLACE(street_address, 'St.', 'Street')
FROM student
WHERE INSTR (street_address, 'St.') > 0
```

8

Concatenation Operator ||

- Combines two character strings into one result string

```
SQL> SELECT last_name || ', ' || first_name, phone
FROM instructor;
```

```
SQL> SELECT city || ', ' || state || ' ' || zip
FROM zipcode;
```

- vs CONCAT()

9

Number Functions

Function	Result
ABS(-20)	
MOD(90,60)	
ROUND(63.7352, 2)	
TRUNC(63.7352, 2)	
ROUND(63.7352, -1)	
ROUND(63.7352, -2)	
FLOOR(63.7352)	
CEIL(63.7352)	
CEIL(63.0001)	

13

Arithmetic Operators

Operator	Meaning
*	multiplication
/	division
+	addition
-	subtraction

- Shown in order of **precedence**

```
SQL> SELECT 3+4*5 FROM DUAL;
```

```
SQL> SELECT ln, amount, amount + 50
FROM writer
WHERE amount <> 0;
```

14

In-Class Practice

- Alex filled up his SUV with 21.3 gallons at odometer reading 23,557. His previous fill-up was at odometer reading 23,266. How many MPG did he get?
- Each article is to be rewritten to condense it to 65% of its original length. Show each title, original length and new length. Round calculated new lengths to the closest whole number. Entitle the column **Goal**.
- A PowerPoint presentation has 22 slides. When printed 3-to-a-page, how many pages are needed and how many slides will appear on the last page?
- Using string functions, show the title of any article containing **War**.

15

Nulls

- Nulls cause problems in calculations
 - the result will be Null if any of its inputs are Null

```
SQL> desc scott.emp
SQL> SELECT ename, sal, comm, sal+comm AS Total FROM scott.emp;
ENAME      SAL      COMM      TOTAL
-----
SMITH      800      300      1900
ALLEN      1600     500      2100
WARD       1250     500      1750
JONES      2975     500      3475
```

```
SQL> SELECT ename, sal, comm, sal+NVL(comm,0) AS Total FROM scott.emp;
ENAME      SAL      COMM      TOTAL
-----
SMITH      800      300      1100
ALLEN      1600     500      2100
WARD       1250     500      1750
JONES      2975     500      3475
```

16

NVL Function

```
NVL(input_expression, substitute_expression)
```

- Replaces a NULL value with a default value
- If *input_expression* is null, NVL returns *substitute_expression*
- If *input_expression* has a value, NVL returns *input_expression*
- Datatypes must be compatible
 - NVL(hiredate,'01-JAN-97') --both are dates
 - NVL(phone, 'Not on file') --both are text
 - NVL(amount, 0) --both are numeric

```
SQL> SELECT ln, fn, NVL(contact, 'No supervisor') Contact
FROM writer;
```

17

COALESCE Function

```
COALESCE(input_expr, subst_expr1 [, subst_expr2], [...])
```

- Searches a list of replacement values until a non-null value is available to return
- Datatypes must be compatible

```
SQL> SELECT last_name, first_name,
COALESCE(work_phone, cell_phone, home_phone)
FROM employee;
```

18

NVL2 Function

```
NVL2(input_expr, not_null_subst_expr, null_subst_expr)
```

- Returns one of two values, depending on whether `input_expr` is NULL
 - If `input_expr` has a value, NVL2 returns `not_null_subst_expr`
 - If `input_expr` is null, NVL2 returns `null_subst_expr`
- Datatypes of last 2 arguments must be compatible

```
SQL> SELECT ln, fn, phone, NVL2(phone, 'On file', 'Not on file') "Status"
       FROM writer
-----
ln          fn          phone          Status
-----
Lawton      Pat          (917) 361-8181 On file
Waldeck     Kristine     (210) 783-5415 On file
Dox         Kelly
...
Hissner     Tonya       Not on file
```

NULLIF Function

"null if equal"

```
NULLIF(input_expr1, input_expr2)
```

- Used to compare two values and display a result
- Compares two input expressions
 - If `input_expr1 = input_expr2`, NULLIF returns NULL
 - If `input_expr1 ≠ input_expr2`, NULLIF returns `input_expr1`

```
SQL> SELECT student_id, last_name, created_date, modified_date,
       NULLIF(created_date, modified_date)
       FROM student;
```

```
SQL> SELECT student_id, last_name, created_date, modified_date
       FROM student
       WHERE NULLIF(created_date, modified_date) IS NOT NULL;
```

20

DECODE Function

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

- Used to display substitute values
- Provides if...then...else logic

```
SQL> SELECT ln, DECODE(freelancer, 'Y', 'Freelancer',
       'N', 'Staff Writer',
       'Unknown') AS Status
       FROM writer;
```

21

Searched Case Expression

```
CASE WHEN condition THEN return_expr  
  [WHEN condition THEN return_expr]  
  [ELSE else_expr]  
END
```

- Specifies a **set** of alternate return values
- Conditions are evaluated sequentially; 1st true one is returned
- Conditions can include ranges, boolean operators
- Can be used in SELECT, WHERE, or ORDER BY clauses
- Datatypes of return values must be compatible

```
SQL> SELECT ename, sal, CASE WHEN sal <1000 THEN 'grunt'  
                        WHEN sal <=3000 THEN 'drone'  
                        WHEN sal IS NULL THEN 'unknown'  
                        ELSE 'honcho'  
                        END AS Status  
FROM scott.emp
```

22
