

3

Using Single-Row Functions to Customize Output

ORACLE®

Copyright © 2007, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Describe various types of functions available in SQL
- Use character, number, and date functions in `SELECT` statements

ORACLE

3 - 2

Copyright © 2007, Oracle. All rights reserved.

Objectives

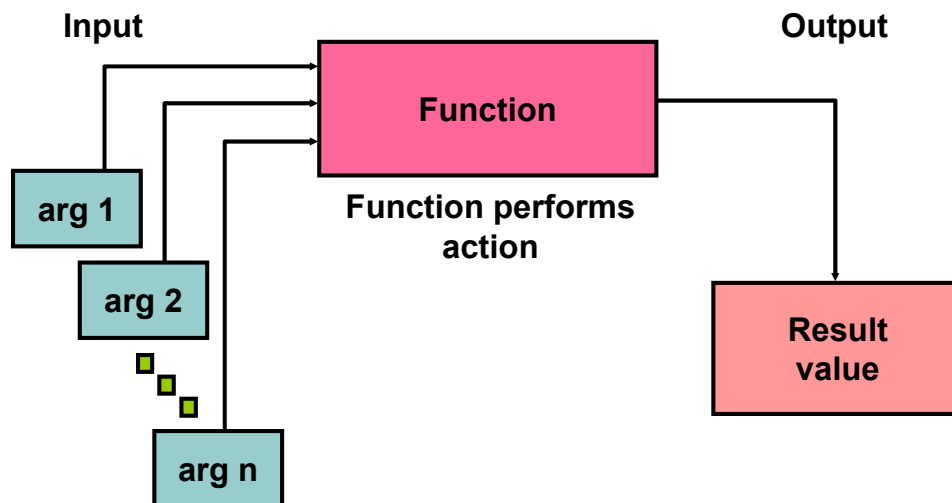
Functions make the basic query block more powerful, and they are used to manipulate data values. This is the first of two lessons that explore functions. It focuses on single-row character, number, and date functions.

Lesson Agenda

- Single-row SQL functions
- Character functions
- Number functions
- Working with dates
- Date functions

ORACLE

SQL Functions



ORACLE

3 - 4

Copyright © 2007, Oracle. All rights reserved.

SQL Functions

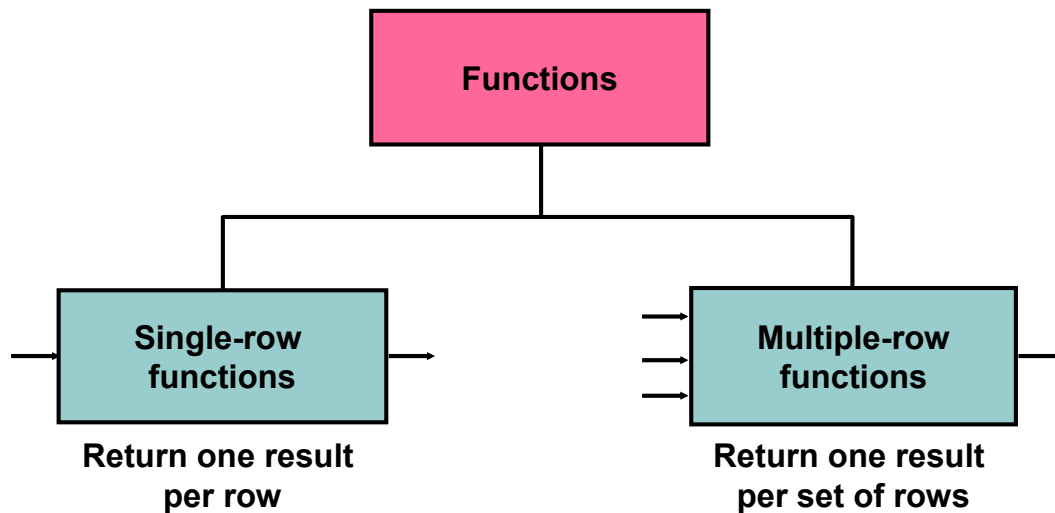
Functions are a very powerful feature of SQL. They can be used to do the following:

- Perform calculations on data
- Modify individual data items
- Manipulate output for groups of rows
- Format dates and numbers for display
- Convert column data types

SQL functions sometimes take arguments and always return a value.

Note: If you want to know whether a function is a SQL:2003 compliant function, refer to the *Oracle Compliance To Core SQL:2003* section in *Oracle Database SQL Language Reference 11g, Release 1 (11.1)*.

Two Types of SQL Functions



ORACLE

3 - 5

Copyright © 2007, Oracle. All rights reserved.

Two Types of SQL Functions

There are two types of functions:

- Single-row functions
- Multiple-row functions

Single-Row Functions

These functions operate on single rows only and return one result per row. There are different types of single-row functions. This lesson covers the following ones:

- Character
- Number
- Date
- Conversion
- General

Multiple-Row Functions

Functions can manipulate groups of rows to give one result per group of rows. These functions are also known as *group functions* (covered in lesson 5 titled “Reporting Aggregated Data Using the Group Functions”).

Note: For more information and a complete list of available functions and their syntax, see the topic, *Functions* in *Oracle Database SQL Language Reference 11g, Release 1 (11.1)*.

Single-Row Functions

Single-row functions:

- Manipulate data items
- Accept arguments and return one value
- Act on each row that is returned
- Return one result per row
- May modify the data type
- Can be nested
- Accept arguments that can be a column or an expression

```
function_name [(arg1, arg2,...)]
```

ORACLE

3 - 6

Copyright © 2007, Oracle. All rights reserved.

Single-Row Functions

Single-row functions are used to manipulate data items. They accept one or more arguments and return one value for each row that is returned by the query. An argument can be one of the following:

- User-supplied constant
- Variable value
- Column name
- Expression

Features of single-row functions include:

- Acting on each row that is returned in the query
- Returning one result per row
- Possibly returning a data value of a different type than the one that is referenced
- Possibly expecting one or more arguments
- Can be used in SELECT, WHERE, and ORDER BY clauses; can be nested

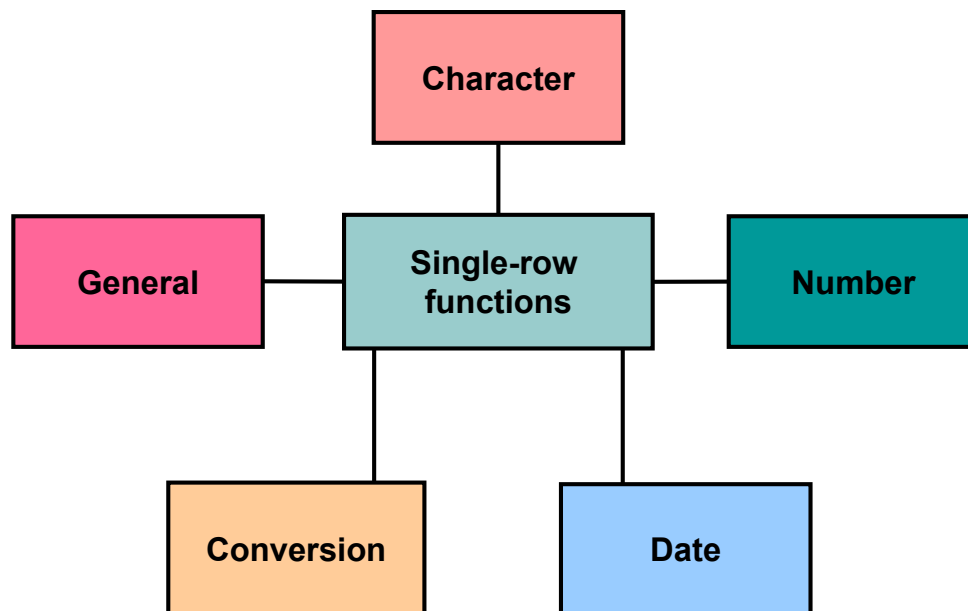
In the syntax:

function_name
arg1, arg2

is the name of the function

is any argument to be used by the function. This can be represented by a column name or expression.

Single-Row Functions



ORACLE

3 - 7

Copyright © 2007, Oracle. All rights reserved.

Single-Row Functions (continued)

This lesson covers the following single-row functions:

- **Character functions:** Accept character input and can return both character and number values
- **Number functions:** Accept numeric input and return numeric values
- **Date functions:** Operate on values of the DATE data type (All date functions return a value of the DATE data type except the MONTHS_BETWEEN function, which returns a number.)

The following single-row functions are discussed in the next lesson titled “Using Conversion Functions and Conditional Expressions”:

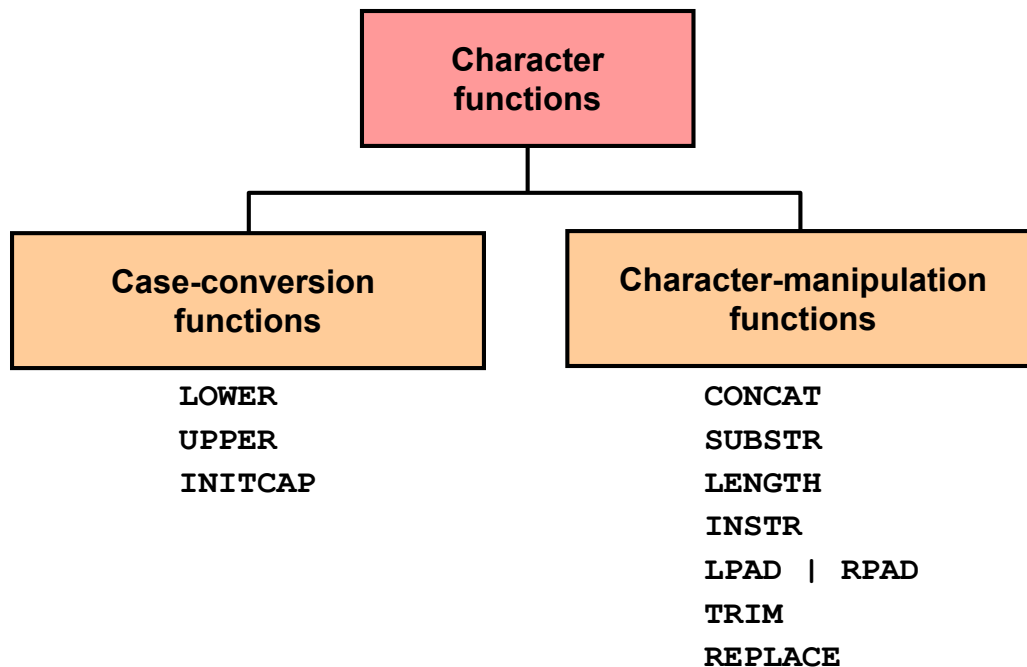
- **Conversion functions:** Convert a value from one data type to another
- **General functions:**
 - NVL
 - NVL2
 - NULLIF
 - COALESCE
 - CASE
 - DECODE

Lesson Agenda

- Single-row SQL functions
- **Character functions**
- Number functions
- Working with dates
- Date functions

ORACLE

Character Functions



ORACLE

3 - 9

Copyright © 2007, Oracle. All rights reserved.

Character Functions

Single-row character functions accept character data as input and can return both character and numeric values. Character functions can be divided into the following:

- Case-conversion functions
- Character-manipulation functions

| Function | Purpose |
|---|---|
| <code>LOWER(column expression)</code> | Converts alpha character values to lowercase |
| <code>UPPER(column expression)</code> | Converts alpha character values to uppercase |
| <code>INITCAP(column expression)</code> | Converts alpha character values to uppercase for the first letter of each word; all other letters in lowercase |
| <code>CONCAT(column1 expression1, column2 expression2)</code> | Concatenates the first character value to the second character value; equivalent to concatenation operator () |
| <code>SUBSTR(column expression, m[, n])</code> | Returns specified characters from character value starting at character position <i>m</i> , <i>n</i> characters long (If <i>m</i> is negative, the count starts from the end of the character value. If <i>n</i> is omitted, all characters to the end of the string are returned.) |

Note: The functions discussed in this lesson are only some of the available functions.

Character Functions (continued)

| Function | Purpose |
|--|---|
| LENGTH(<i>column expression</i>) | Returns the number of characters in the expression |
| INSTR(<i>column expression</i> , ' <i>string</i> ', [<i>m</i>], [<i>n</i>]) | Returns the numeric position of a named string. Optionally, you can provide a position <i>m</i> to start searching, and the occurrence <i>n</i> of the string. <i>m</i> and <i>n</i> default to 1, meaning start the search at the beginning of the string and report the first occurrence. |
| LPAD(<i>column expression</i> , <i>n</i> , ' <i>string</i> ') RPAD(<i>column expression</i> , <i>n</i> , ' <i>string</i> ') | Returns an expression left-padded to length of <i>n</i> characters with a character expression. Returns an expression right-padded to length of <i>n</i> characters with a character expression. |
| TRIM(<i>leading trailing both</i> , <i>trim_character</i> FROM <i>trim_source</i>) | Enables you to trim leading or trailing characters (or both) from a character string. If <i>trim_character</i> or <i>trim_source</i> is a character literal, you must enclose it in single quotation marks. This is a feature that is available in Oracle8i and later versions. |
| REPLACE(<i>text</i> , <i>search_string</i> , <i>replacement_string</i>) | Searches a text expression for a character string and, if found, replaces it with a specified replacement string |

Note: Some of the functions that are fully or partially SQL:2003 compliant are:

UPPER
LOWER
TRIM
LENGTH
SUBSTR
INSTR

Refer to the *Oracle Compliance To Core SQL:2003* section in *Oracle Database SQL Language Reference 11g, Release 1 (11.1)* for more information.

Case-Conversion Functions

These functions convert the case for character strings:

| Function | Result |
|-----------------------|------------|
| LOWER('SQL Course') | sql course |
| UPPER('SQL Course') | SQL COURSE |
| INITCAP('SQL Course') | Sql Course |

ORACLE

3 - 11

Copyright © 2007, Oracle. All rights reserved.

Case-Conversion Functions

LOWER, UPPER, and INITCAP are the three case-conversion functions.

- LOWER: Converts mixed-case or uppercase character strings to lowercase
- UPPER: Converts mixed-case or lowercase character strings to uppercase
- INITCAP: Converts the first letter of each word to uppercase and the remaining letters to lowercase

```
SELECT 'The job id for '||UPPER(last_name)||' is '  
      ||LOWER(job_id) AS "EMPLOYEE DETAILS"  
FROM   employees;
```

| | A Z | EMPLOYEE DETAILS |
|-----|-----|-----------------------------------|
| 1 | | The job id for ABEL is sa_rep |
| 2 | | The job id for DAVIES is st_clerk |
| 3 | | The job id for DE HAAN is ad_vp |
| ... | | |
| 19 | | The job id for WHALEN is ad_asst |
| 20 | | The job id for ZLOTKEY is sa_man |

Using Case-Conversion Functions

Display the employee number, name, and department number for employee Higgins:

```
SELECT employee_id, last_name, department_id
FROM employees
WHERE last_name = 'higgins';
```

0 rows selected

```
SELECT employee_id, last_name, department_id
FROM employees
WHERE LOWER(last_name) = 'higgins';
```

| | EMPLOYEE_ID | LAST_NAME | DEPARTMENT_ID |
|---|-------------|-----------|---------------|
| 1 | 205 | Higgins | 110 |

ORACLE

3 - 12

Copyright © 2007, Oracle. All rights reserved.

Using Case-Conversion Functions

The slide example displays the employee number, name, and department number of employee Higgins.

The WHERE clause of the first SQL statement specifies the employee name as `higgins`. Because all the data in the `EMPLOYEES` table is stored in proper case, the name `higgins` does not find a match in the table, and no rows are selected.

The WHERE clause of the second SQL statement specifies that the employee name in the `EMPLOYEES` table is compared to `higgins`, converting the `LAST_NAME` column to lowercase for comparison purposes. Because both names are now lowercase, a match is found and one row is selected. The WHERE clause can be rewritten in the following manner to produce the same result:

```
...WHERE last_name = 'Higgins'
```

The name in the output appears as it was stored in the database. To display the name in uppercase, use the `UPPER` function in the SELECT statement.

```
SELECT employee_id, UPPER(last_name), department_id
FROM employees
WHERE INITCAP(last_name) = 'Higgins';
```

Character-Manipulation Functions

These functions manipulate character strings:

| Function | Result |
|--------------------------------------|----------------|
| CONCAT('Hello', 'World') | HelloWorld |
| SUBSTR('HelloWorld',1,5) | Hello |
| LENGTH('HelloWorld') | 10 |
| INSTR('HelloWorld', 'W') | 6 |
| LPAD(salary,10,'*') | *****24000 |
| RPAD(salary, 10, '*') | 24000***** |
| REPLACE ('JACK and JUE','J','BL') | BLACK and BLUE |
| TRIM('H' FROM 'HelloWorld') | elloWorld |

ORACLE

3 - 13

Copyright © 2007, Oracle. All rights reserved.

Character-Manipulation Functions

CONCAT, SUBSTR, LENGTH, INSTR, LPAD, RPAD, and TRIM are the character-manipulation functions that are covered in this lesson.

- CONCAT: Joins values together (You are limited to using two parameters with CONCAT.)
- SUBSTR: Extracts a string of determined length
- LENGTH: Shows the length of a string as a numeric value
- INSTR: Finds the numeric position of a named character
- LPAD: Returns an expression left-padded to the length of *n* characters with a character expression
- RPAD: Returns an expression right-padded to the length of *n* characters with a character expression
- TRIM: Trims leading or trailing characters (or both) from a character string (If *trim_character* or *trim_source* is a character literal, you must enclose it within single quotation marks.)

Note: You can use functions such as UPPER and LOWER with ampersand substitution. For example, use UPPER('&job_title') so that the user does not have to enter the job title in a specific case.

Using the Character-Manipulation Functions

```

SELECT employee_id, CONCAT(first_name, last_name) NAME,
       job_id, LENGTH (last_name),
       INSTR(last_name, 'a') "Contains 'a'?"
FROM   employees
WHERE  SUBSTR(job_id, 4) = 'REP';
    
```

| | EMPLOYEE_ID | NAME | JOB_ID | LENGTH(LAST_NAME) | Contains 'a'? |
|---|-------------|----------------|--------|-------------------|---------------|
| 1 | 174 | EllenAbel | SA_REP | 4 | 0 |
| 2 | 176 | JonathonTaylor | SA_REP | 6 | 2 |
| 3 | 178 | KimberelyGrant | SA_REP | 5 | 3 |
| 4 | 202 | PatFay | MK_REP | 3 | 2 |

ORACLE

3 - 14

Copyright © 2007, Oracle. All rights reserved.

Using the Character-Manipulation Functions

The slide example displays employee first names and last names joined together, the length of the employee last name, and the numeric position of the letter “a” in the employee last name for all employees who have the string, REP, contained in the job ID starting at the fourth position of the job ID.

Example:

Modify the SQL statement in the slide to display the data for those employees whose last names end with the letter “n.”

```

SELECT employee_id, CONCAT(first_name, last_name) NAME,
       LENGTH (last_name), INSTR(last_name, 'a') "Contains 'a'?"
FROM   employees
WHERE  SUBSTR(last_name, -1, 1) = 'n';
    
```

| | EMPLOYEE_ID | NAME | LENGTH(LAST_NAME) | Contains 'a'? |
|---|-------------|------------------|-------------------|---------------|
| 1 | 102 | LexDe Haan | 7 | 5 |
| 2 | 200 | JenniferWhalen | 6 | 3 |
| 3 | 201 | MichaelHartstein | 9 | 2 |

Lesson Agenda

- Single-row SQL functions
- Character functions
- **Number functions**
- Working with dates
- Date Functions

ORACLE

Number Functions

- **ROUND:** Rounds value to a specified decimal
- **TRUNC:** Truncates value to a specified decimal
- **MOD:** Returns remainder of division

| Function | Result |
|-------------------|--------|
| ROUND (45.926, 2) | 45.93 |
| TRUNC (45.926, 2) | 45.92 |
| MOD (1600, 300) | 100 |

ORACLE

3 - 16

Copyright © 2007, Oracle. All rights reserved.

Number Functions

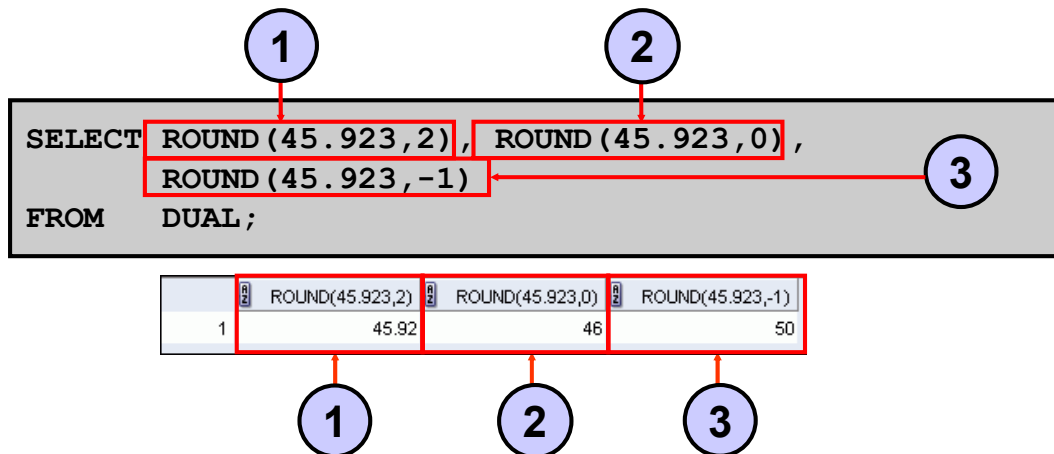
Number functions accept numeric input and return numeric values. This section describes some of the number functions.

| Function | Purpose |
|--|---|
| ROUND (<i>column</i> <i>expression</i> , <i>n</i>) | Rounds the column, expression, or value to <i>n</i> decimal places or, if <i>n</i> is omitted, no decimal places (If <i>n</i> is negative, numbers to the left of decimal point are rounded.) |
| TRUNC (<i>column</i> <i>expression</i> , <i>n</i>) | Truncates the column, expression, or value to <i>n</i> decimal places or, if <i>n</i> is omitted, <i>n</i> defaults to zero |
| MOD (<i>m</i> , <i>n</i>) | Returns the remainder of <i>m</i> divided by <i>n</i> |

Note: This list contains only some of the available number functions.

For more information, see the section on *Numeric Functions* in *Oracle Database SQL Language Reference 11g, Release 1 (11.1)*.

Using the ROUND Function



DUAL is a dummy table that you can use to view results from functions and calculations.

ORACLE

3 - 17

Copyright © 2007, Oracle. All rights reserved.

Using the ROUND Function

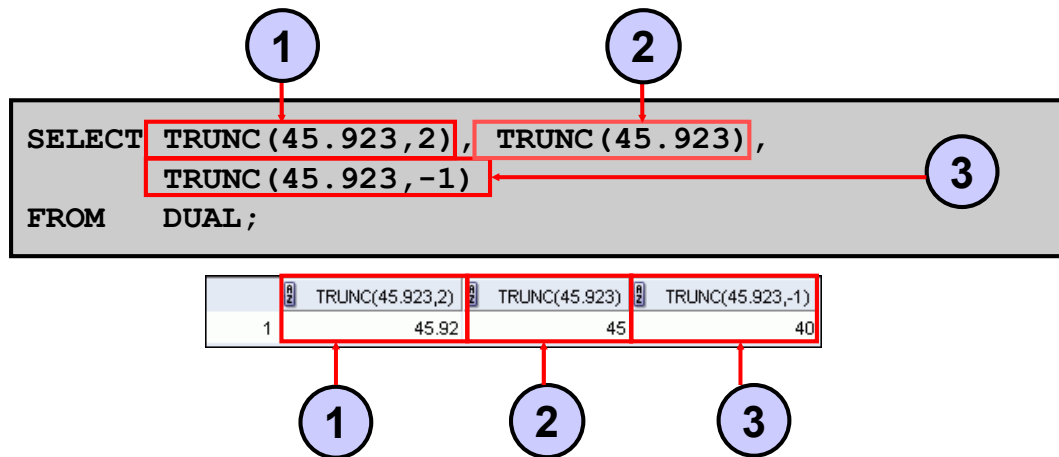
The ROUND function rounds the column, expression, or value to *n* decimal places. If the second argument is 0 or is missing, the value is rounded to zero decimal places. If the second argument is 2, the value is rounded to two decimal places. Conversely, if the second argument is -2, the value is rounded to two decimal places to the left (rounded to the nearest unit of 100).

The ROUND function can also be used with date functions. You will see examples later in this lesson.

DUAL Table

The DUAL table is owned by the user SYS and can be accessed by all users. It contains one column, DUMMY, and one row with the value X. The DUAL table is useful when you want to return a value only once (for example, the value of a constant, pseudocolumn, or expression that is not derived from a table with user data). The DUAL table is generally used for completeness of the SELECT clause syntax, because both SELECT and FROM clauses are mandatory, and several calculations do not need to select from the actual tables.

Using the TRUNC Function



ORACLE

3 - 18

Copyright © 2007, Oracle. All rights reserved.

Using the TRUNC Function

The TRUNC function truncates the column, expression, or value to n decimal places.

The TRUNC function works with arguments similar to those of the ROUND function. If the second argument is 0 or is missing, the value is truncated to zero decimal places. If the second argument is 2, the value is truncated to two decimal places. Conversely, if the second argument is -2, the value is truncated to two decimal places to the left. If the second argument is -1, the value is truncated to one decimal place to the left.

Like the ROUND function, the TRUNC function can be used with date functions.

Using the MOD Function

For all employees with the job title of Sales Representative, calculate the remainder of the salary after it is divided by 5,000.

```
SELECT last_name, salary, MOD(salary, 5000)
FROM   employees
WHERE  job_id = 'SA_REP';
```

| | LAST_NAME | SALARY | MOD(SALARY,5000) |
|---|-----------|--------|------------------|
| 1 | Abel | 11000 | 1000 |
| 2 | Taylor | 8600 | 3600 |
| 3 | Grant | 7000 | 2000 |

ORACLE

Using the MOD Function

The MOD function finds the remainder of the first argument divided by the second argument. The slide example calculates the remainder of the salary after dividing it by 5,000 for all employees whose job ID is SA_REP.

Note: The MOD function is often used to determine whether a value is odd or even.

Lesson Agenda

- Single-row SQL functions
- Character functions
- Number functions
- **Working with dates**
- Date functions

ORACLE

Working with Dates

- The Oracle database stores dates in an internal numeric format: century, year, month, day, hours, minutes, and seconds.
- The default date display format is DD-MON-RR.
 - Enables you to store 21st-century dates in the 20th century by specifying only the last two digits of the year
 - Enables you to store 20th-century dates in the 21st century in the same way

```
SELECT last_name, hire_date
FROM   employees
WHERE  hire_date < '01-FEB-88';
```

| | LAST_NAME | HIRE_DATE |
|---|-----------|-----------|
| 1 | King | 17-JUN-87 |
| 2 | Whalen | 17-SEP-87 |

ORACLE

3 - 21

Copyright © 2007, Oracle. All rights reserved.

Working with Dates

The Oracle database stores dates in an internal numeric format, representing the century, year, month, day, hours, minutes, and seconds.

The default display and input format for any date is DD-MON-RR. Valid Oracle dates are between January 1, 4712 B.C., and December 31, 9999 A.D.

In the example in the slide, the `HIRE_DATE` column output is displayed in the default format DD-MON-RR. However, dates are not stored in the database in this format. All the components of the date and time are stored. So, although a `HIRE_DATE` such as 17-JUN-87 is displayed as day, month, and year, there is also *time* and *century* information associated with the date. The complete data might be June 17, 1987, 5:10:43 PM.

RR Date Format

| Current Year | Specified Date | RR Format | YY Format |
|--------------|----------------|-----------|-----------|
| 1995 | 27-OCT-95 | 1995 | 1995 |
| 1995 | 27-OCT-17 | 2017 | 1917 |
| 2001 | 27-OCT-17 | 2017 | 2017 |
| 2001 | 27-OCT-95 | 1995 | 2095 |

| | | If the specified two-digit year is: | |
|--|-------|---|--|
| | | 0–49 | 50–99 |
| If two digits of the current year are: | 0–49 | The return date is in the current century | The return date is in the century before the current one |
| | 50–99 | The return date is in the century after the current one | The return date is in the current century |

ORACLE

3 - 22

Copyright © 2007, Oracle. All rights reserved.

RR Date Format

The **RR** date format is similar to the **YY** element, but you can use it to specify different centuries. Use the **RR** date format element instead of **YY** so that the century of the return value varies according to the specified two-digit year and the last two digits of the current year. The table in the slide summarizes the behavior of the **RR** element.

| Current Year | Given Date | Interpreted (RR) | Interpreted (YY) |
|--------------|------------|------------------|------------------|
| 1994 | 27-OCT-95 | 1995 | 1995 |
| 1994 | 27-OCT-17 | 2017 | 1917 |
| 2001 | 27-OCT-17 | 2017 | 2017 |

Oracle Date Format

This data is stored internally as follows:

| CENTURY | YEAR | MONTH | DAY | HOUR | MINUTE | SECOND |
|---------|------|-------|-----|------|--------|--------|
| 19 | 87 | 06 | 17 | 17 | 10 | 43 |

Centuries and the Year 2000

When a record with a date column is inserted into a table, the *century* information is picked up from the `SYSDATE` function. However, when the date column is displayed on the screen, the century component is not displayed (by default).

The `DATE` data type always stores year information as a four-digit number internally: two digits for the century and two digits for the year. For example, the Oracle database stores the year as 1987 or 2004, and not just as 87 or 04.

Using the SYSDATE Function

SYSDATE is a function that returns:

- Date
- Time

```
SELECT sysdate  
FROM dual;
```

| SYSDATE |
|-------------|
| 1 31-MAY-07 |

ORACLE

Using the SYSDATE Function

SYSDATE is a date function that returns the current database server date and time. You can use SYSDATE just as you would use any other column name. For example, you can display the current date by selecting SYSDATE from a table. It is customary to select SYSDATE from a dummy table called DUAL.

Note: SYSDATE returns the current date and time set for the operating system on which the database resides. Hence, if you are in a place in Australia and connected to a remote database in a location in the United States (US), sysdate function will return the US date and time. In that case, you can use the CURRENT_DATE function that returns the current date in the session time zone.

The CURRENT_DATE function and other related time zone functions are discussed in detail in the course titled *Oracle Database 11g: SQL Fundamentals II*.

Arithmetic with Dates

- Add or subtract a number to or from a date for a resultant date value.
- Subtract two dates to find the number of days between those dates.
- Add hours to a date by dividing the number of hours by 24.

ORACLE

3 - 25

Copyright © 2007, Oracle. All rights reserved.

Arithmetic with Dates

Because the database stores dates as numbers, you can perform calculations using arithmetic operators such as addition and subtraction. You can add and subtract number constants as well as dates.

You can perform the following operations:

| Operation | Result | Description |
|------------------|----------------|--|
| date + number | Date | Adds a number of days to a date |
| date – number | Date | Subtracts a number of days from a date |
| date – date | Number of days | Subtracts one date from another |
| date + number/24 | Date | Adds a number of hours to a date |

Using Arithmetic Operators with Dates

```
SELECT last_name, (SYSDATE-hire_date)/7 AS WEEKS  
FROM employees  
WHERE department_id = 90;
```

| | LAST_NAME | WEEKS |
|---|-----------|-------------------------------------|
| 1 | King | 1041.168239087301587301587301587302 |
| 2 | Kochhar | 923.025381944444444444444444444444 |
| 3 | De Haan | 750.168239087301587301587301587302 |

ORACLE

3 - 26

Copyright © 2007, Oracle. All rights reserved.

Using Arithmetic Operators with Dates

The example in the slide displays the last name and the number of weeks employed for all employees in department 90. It subtracts the date on which the employee was hired from the current date (SYSDATE) and divides the result by 7 to calculate the number of weeks that a worker has been employed.

Note: SYSDATE is a SQL function that returns the current date and time. Your results may differ depending on the date and time set for the operating system of your local database when you run the SQL query.

If a more current date is subtracted from an older date, the difference is a negative number.

Lesson Agenda

- Single-row SQL functions
- Character functions
- Number functions
- Working with dates
- **Date functions**

ORACLE

Date-Manipulation Functions

| Function | Result |
|----------------|------------------------------------|
| MONTHS_BETWEEN | Number of months between two dates |
| ADD_MONTHS | Add calendar months to date |
| NEXT_DAY | Next day of the date specified |
| LAST_DAY | Last day of the month |
| ROUND | Round date |
| TRUNC | Truncate date |

ORACLE

3 - 28

Copyright © 2007, Oracle. All rights reserved.

Date-Manipulation Functions

Date functions operate on Oracle dates. All date functions return a value of the DATE data type except MONTHS_BETWEEN, which returns a numeric value.

- MONTHS_BETWEEN(*date1*, *date2*): Finds the number of months between *date1* and *date2*. The result can be positive or negative. If *date1* is later than *date2*, the result is positive; if *date1* is earlier than *date2*, the result is negative. The noninteger part of the result represents a portion of the month.
- ADD_MONTHS(*date*, *n*): Adds *n* number of calendar months to *date*. The value of *n* must be an integer and can be negative.
- NEXT_DAY(*date*, '*char*'): Finds the date of the next specified day of the week ('*char*') following *date*. The value of *char* may be a number representing a day or a character string.
- LAST_DAY(*date*): Finds the date of the last day of the month that contains *date*.

The above list is a subset of the available date functions. ROUND and TRUNC number functions can also be used to manipulate the date values as shown below:

- ROUND(*date*[, '*fmt*']): Returns *date* rounded to the unit that is specified by the format model *fmt*. If the format model *fmt* is omitted, *date* is rounded to the nearest day.
- TRUNC(*date*[, '*fmt*']): Returns *date* with the time portion of the day truncated to the unit that is specified by the format model *fmt*. If the format model *fmt* is omitted, *date* is truncated to the nearest day.

The format models are covered in detail in the next lesson titled “Using Conversion Functions and Conditional Expressions.”

Using Date Functions

| Function | Result |
|--|-------------|
| MONTHS_BETWEEN ('01-SEP-95', '11-JAN-94') | 19.6774194 |
| ADD_MONTHS ('31-JAN-96', 1) | '29-FEB-96' |
| NEXT_DAY ('01-SEP-95', 'FRIDAY') | '08-SEP-95' |
| LAST_DAY ('01-FEB-95') | '28-FEB-95' |

ORACLE

3 - 29

Copyright © 2007, Oracle. All rights reserved.

Using Date Functions

In the slide example, the ADD_MONTHS function adds one month to the supplied date value, “31-JAN-96” and returns “29-FEB-96.” The function recognizes the year 1996 as the leap year and hence returns the last day of the February month. If you change the input date value to “31-JAN-95,” the function returns “28-FEB-95.”

For example, display the employee number, hire date, number of months employed, six-month review date, first Friday after hire date, and the last day of the hire month for all employees who have been employed for fewer than 100 months.

```
SELECT employee_id, hire_date,
       MONTHS_BETWEEN (SYSDATE, hire_date) TENURE,
       ADD_MONTHS (hire_date, 6) REVIEW,
       NEXT_DAY (hire_date, 'FRIDAY'), LAST_DAY(hire_date)
FROM   employees
WHERE  MONTHS_BETWEEN (SYSDATE, hire_date) < 100;
```

| | EMPLOYEE_ID | HIRE_DATE | TENURE | REVIEW | NEXT_DAY(HIRE_DATE,FRIDAY) | LAST_DAY(HIRE_DATE) |
|---|-------------|-----------|---------------|-----------|----------------------------|---------------------|
| 1 | 124 | 16-NOV-99 | 91.1099600... | 16-MAY-00 | 19-NOV-99 | 30-NOV-99 |
| 2 | 149 | 29-JAN-00 | 88.6906052... | 29-JUL-00 | 04-FEB-00 | 31-JAN-00 |
| 3 | 178 | 24-MAY-99 | 96.8518955... | 24-NOV-99 | 28-MAY-99 | 31-MAY-99 |
| 4 | 99999 | 07-JUN-99 | 96.4002826... | 07-DEC-99 | 11-JUN-99 | 30-JUN-99 |
| 5 | 113 | 11-JUN-07 | 0.25824335... | 11-DEC-07 | 15-JUN-07 | 30-JUN-07 |

Using ROUND and TRUNC Functions with Dates

Assume `SYSDATE = '25-JUL-03'`:

| Function | Result |
|--------------------------------------|-----------|
| <code>ROUND(SYSDATE, 'MONTH')</code> | 01-AUG-03 |
| <code>ROUND(SYSDATE, 'YEAR')</code> | 01-JAN-04 |
| <code>TRUNC(SYSDATE, 'MONTH')</code> | 01-JUL-03 |
| <code>TRUNC(SYSDATE, 'YEAR')</code> | 01-JAN-03 |

ORACLE

3 - 30

Copyright © 2007, Oracle. All rights reserved.

Using ROUND and TRUNC Functions with Dates

The `ROUND` and `TRUNC` functions can be used for number and date values. When used with dates, these functions round or truncate to the specified format model. Therefore, you can round dates to the nearest year or month. If the format model is month, dates 1-15 result in the first day of the current month. Dates 16-31 result in the first day of the next month. If the format model is year, months 1-6 result in January 1 of the current year. Months 7-12 result in January 1 of the next year.

Example:

Compare the hire dates for all employees who started in 1997. Display the employee number, hire date, and starting month using the `ROUND` and `TRUNC` functions.

```
SELECT employee_id, hire_date,  
       ROUND(hire_date, 'MONTH'), TRUNC(hire_date, 'MONTH')  
FROM   employees  
WHERE  hire_date LIKE '%97';
```

| | EMPLOYEE_ID | HIRE_DATE | ROUND(HIRE_DATE,'MONTH') | TRUNC(HIRE_DATE,'MONTH') |
|---|-------------|-----------|--------------------------|--------------------------|
| 1 | 142 | 29-JAN-97 | 01-FEB-97 | 01-JAN-97 |
| 2 | 202 | 17-AUG-97 | 01-SEP-97 | 01-AUG-97 |

Summary

In this lesson, you should have learned how to:

- Perform calculations on data using functions
- Modify individual data items using functions

ORACLE

3 - 31

Copyright © 2007, Oracle. All rights reserved.

Summary

Single-row functions can be nested to any level. Single-row functions can manipulate the following:

- Character data: LOWER, UPPER, INITCAP, CONCAT, SUBSTR, INSTR, LENGTH
- Number data: ROUND, TRUNC, MOD
- Date values: SYSDATE, MONTHS_BETWEEN, ADD_MONTHS, NEXT_DAY, LAST_DAY

Remember the following:

- Date values can also use arithmetic operators.
- ROUND and TRUNC functions can also be used with date values.

SYSDATE and DUAL

SYSDATE is a date function that returns the current date and time. It is customary to select SYSDATE from a dummy table called DUAL.

Practice 3: Overview

This practice covers the following topics:

- Writing a query that displays the current date
- Creating queries that require the use of numeric, character, and date functions
- Performing calculations of years and months of service for an employee

ORACLE

3 - 32

Copyright © 2007, Oracle. All rights reserved.

Practice 3: Overview

This practice provides a variety of exercises using different functions that are available for character, number, and date data types.

Practice 3

Part 1

1. Write a query to display the system date. Label the column as Date.

Note: If your database is remotely located in a different time zone, the output will be the date for the operating system on which the database resides.

| | Date |
|---|-----------|
| 1 | 31-MAY-07 |

2. The HR department needs a report to display the employee number, last name, salary, and salary increased by 15.5% (expressed as a whole number) for each employee. Label the column New Salary. Save your SQL statement in a file named lab_03_02.sql.
3. Run your query in the lab_03_02.sql file.

| | EMPLOYEE_ID | LAST_NAME | SALARY | New Salary |
|----|-------------|-----------|--------|------------|
| 1 | 100 | King | 24000 | 27720 |
| 2 | 101 | Kochhar | 17000 | 19635 |
| 3 | 102 | De Haan | 17000 | 19635 |
| 4 | 103 | Hunold | 9000 | 10395 |
| 5 | 104 | Ernst | 6000 | 6930 |
| 6 | 107 | Lorentz | 4200 | 4851 |
| 7 | 124 | Mourgos | 5800 | 6699 |
| 8 | 141 | Rajs | 3500 | 4043 |
| 9 | 142 | Davies | 3100 | 3581 |
| 10 | 143 | Matos | 2600 | 3003 |

...

| | | | | |
|----|-----|---------|-------|-------|
| 19 | 205 | Higgins | 12000 | 13860 |
| 20 | 206 | Gietz | 8300 | 9587 |

4. Modify your query lab_03_02.sql to add a column that subtracts the old salary from the new salary. Label the column Increase. Save the contents of the file as lab_03_04.sql. Run the revised query.



| | EMPLOYEE_ID | LAST_NAME | SALARY | New Salary | Increase |
|---|-------------|-----------|--------|------------|----------|
| 1 | 100 | King | 24000 | 27720 | 3720 |
| 2 | 101 | Kochhar | 17000 | 19635 | 2635 |
| 3 | 102 | De Haan | 17000 | 19635 | 2635 |
| 4 | 103 | Hunold | 9000 | 10395 | 1395 |
| 5 | 104 | Ernst | 6000 | 6930 | 930 |

...



| | | | | | |
|----|-----|-------|------|------|------|
| 20 | 206 | Gietz | 8300 | 9587 | 1287 |
|----|-----|-------|------|------|------|

Practice 3 (continued)

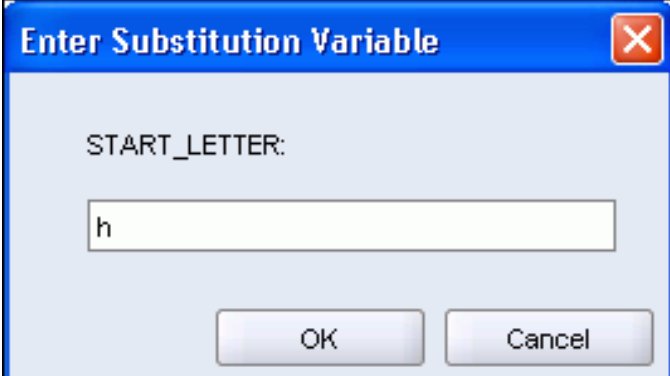
5. Write a query that displays the last name (with the first letter in uppercase and all the other letters in lowercase) and the length of the last name for all employees whose name starts with the letters “J,” “A,” or “M.” Give each column an appropriate label. Sort the results by the employees’ last names.

| |  Name |  Length |
|---|--|--|
| 1 | Abel | 4 |
| 2 | Matos | 5 |
| 3 | Mourgos | 7 |



Rewrite the query so that the user is prompted to enter a letter that the last name starts with. For example, if the user enters “H” (capitalized) when prompted for a letter, then the output should show all employees whose last name starts with the letter “H.”

| |  Name |  Length |
|---|--|--|
| 1 | Hartstein | 9 |
| 2 | Higgins | 7 |
| 3 | Hunold | 6 |

Modify the query such that the case of the entered letter does not affect the output. The entered letter must be capitalized before being processed by the `SELECT` query.



The dialog box titled "Enter Substitution Variable" has a blue header bar with a red close button. The main area is light blue and contains the label "START_LETTER:" followed by a text input field containing the lowercase letter "h". At the bottom, there are two buttons: "OK" and "Cancel".

| |  Name |  Length |
|---|--|--|
| 1 | Hartstein | 9 |
| 2 | Higgins | 7 |
| 3 | Hunold | 6 |

Practice 3 (continued)

- The HR department wants to find the duration of employment for each employee. For each employee, display the last name and calculate the number of months between today and the date on which the employee was hired. Label the column as MONTHS_WORKED. Order your results by the number of months employed. Round the number of months up to the closest whole number.

Note: Because this query depends on the date when it was executed, the values in the MONTHS_WORKED column will differ for you.

| | A Z LAST_NAME | A Z MONTHS_WORKED |
|----|---------------|-------------------|
| 1 | Zlotkey | 88 |
| 2 | Mourgos | 90 |
| 3 | Grant | 96 |
| 4 | Lorentz | 100 |
| 5 | Vargas | 107 |
| 6 | Taylor | 110 |
| 7 | Matos | 111 |
| 8 | Fay | 117 |
| 9 | Davies | 124 |
| 10 | Abel | 133 |
| 11 | Hartstein | 135 |
| 12 | Rajs | 139 |
| 13 | Higgins | 156 |
| 14 | Gietz | 156 |
| 15 | De Haan | 173 |
| 16 | Ernst | 192 |
| 17 | Hunold | 209 |
| 18 | Kochhar | 212 |
| 19 | Whalen | 236 |
| 20 | King | 239 |

Practice 3 (continued)

If you have time, complete the following exercises:

7. Create a query to display the last name and salary for all employees. Format the salary to be 15 characters long, left-padded with the \$ symbol. Label the column as SALARY.

| | LAST_NAME | SALARY |
|---|-----------|-------------------------|
| 1 | King | \$\$\$\$\$\$\$\$\$24000 |
| 2 | Kochhar | \$\$\$\$\$\$\$\$\$17000 |

...

| | | |
|----|-------|------------------------|
| 20 | Gietz | \$\$\$\$\$\$\$\$\$8300 |
|----|-------|------------------------|

8. Create a query that displays the first eight characters of the employees' last names and indicates the amounts of their salaries with asterisks. Each asterisk signifies a thousand dollars. Sort the data in descending order of salary. Label the column as EMPLOYEES_AND_THEIR_SALARIES.

| | EMPLOYEES_AND_THEIR_SALARIES |
|---|------------------------------|
| 1 | King ***** |
| 2 | Kochhar ***** |
| 3 | De Haan ***** |
| 4 | Hartstei ***** |
| 5 | Higgins ***** |

...

| | |
|----|-----------|
| 19 | Matos ** |
| 20 | Vargas ** |

9. Create a query to display the last name and the number of weeks employed for all employees in department 90. Label the number of weeks column as TENURE. Truncate the number of weeks value to 0 decimal places. Show the records in descending order of the employee's tenure.
Note: The TENURE value will differ as it depends on the date on which you run the query.

| | LAST_NAME | TENURE |
|---|-----------|--------|
| 1 | King | 1041 |
| 2 | Kochhar | 923 |
| 3 | De Haan | 750 |