Built-in SQL Functions

Chapter 5

Type of Functions

- Character Functions
  - returning character values
  - returning numeric values
- Numeric Functions
- Date Functions
- Conversion Functions
- Group Functions
- Error Reporting
- Other Functions

Character Functions
Returning Character Values

- CHR
- CONCAT
- INITCAP

Examples

- SELECT INITCAP('the soap') "Capitals" FROM DUAL;

Capitals

The Soap

- SELECT CONCAT(CONCAT(last_name, "'s job category is '), job_id) "Job" FROM employees WHERE employee_id = 152;

Job

Hall's job category is SA_REP

Character Functions
Returning Character Values

- LOWER
- LPAD
- LTRIM
- NLS_INITCAP

Examples

- SHOW LPAD('Page 1',15,'*.')

..*.*.*Page 1

- SELECT NLS_INITCAP ('ijsland') "InitCap" FROM DUAL;

InitCap

------- Ijsland

Example

- SELECT product_name, LTRIM(product_name, 'Monitor ') "Short Name" FROM products WHERE product_name LIKE 'Monitor%';

PRODUCT_NAME	Short Name
Monitor 17/HR	17/HR
Monitor 17/HR/F	17/HR/F
Monitor 17/SD	17/SD
Monitor 19/SD	19/SD
Monitor 19/SD/M	19/SD/M
Monitor 21/D	21/D
Monitor 21/HR	21/HR

Character Functions
Returning Character Values

- NLS_LOWER
- NLS_UPPER
- NLSSORT
- REPLACE
- RPAD

Examples

- SHOW RPAD('Morrison',12,'ab')

Morrisonabab

- SELECT REPLACE('JACK and JUE','J','BL') "Changes" FROM DUAL;

Changes

BLACK and BLUE

Character Functions
Returning Character Values

- RTRIM
- SOUNDEX
- SUBSTR
- SUBSTRB
- TRANSLATE
- UPPER

Examples

- SHOW RTRIM('Last Wordxxyxy','xy')

Last Word

- SELECT SUBSTR('ABCDEFG',3,4) "Substring" FROM DUAL;

Substring

CDEF

- SELECT TRANSLATE('SQL*Plus User''s Guide', ' */''', '___') FROM DUAL;

TRANSLATE(' 'SQL*Plus User''s Guide

SQL_Plus_Users_Guide

Character Functions
Returning Numeric Values

- ASCII
- INSTR
- INSTRB
- LENGTH
- LENGTHB

Examples

- SHOW INSTR('Corporate Floor','or', 3, 2)

14

- SELECT LENGTH('CANDIDE') "Length in characters" FROM DUAL;

Length in characters

7

Numeric Functions

- ABS
- ACOS
- ASIN
- ATAN
- ATAN2

Numeric Functions

- CEIL
- COS
- COSH
- EXP
- FLOOR
- LN

Examples

- SELECT order_total, CEIL(order_total) FROM orders WHERE order_id = 2434;

ORDER_TOTAL CEIL(ORDER_TOTAL)

---------- ----------

268651.8 268652

- SELECT FLOOR(15.7) "Floor" FROM DUAL;

Floor

15

Numeric Functions

- LOG
- MOD
- POWER
- ROUND
- SIGN
- SIN

Examples

- SELECT MOD(11,4) "Modulus" FROM DUAL;

Modulus

3

- SELECT ROUND(15.193,1) "Round" FROM DUAL;

Round

15.2

Numeric Functions

- SINH
- SQRT
- TAN
- TANH
- TRUNC

Example

- SELECT TRUNC(15.79,1) "Truncate" FROM DUAL;

Truncate

15.7

Date Functions

- ADD_MONTHS
- LAST_DAY
- MONTHS_BETWEEN
- NEW_TIME
- NEXT_DAY
- ROUND
- SYSDATE
- TRUNC

Examples

- SELECT MONTHS_BETWEEN (TO_DATE('02-02-1995','MM-DD-YYYY'), TO_DATE('01-01-1995','MM-DD-YYYY')) "Months" FROM DUAL;

Months

1.03225806

- SELECT NEW_TIME(TO_DATE('11-10-99 01:23:45', 'MM-DD-YY HH24:MI:SS'), 'AST', 'PST') "New Date and Time" FROM DUAL;

New Date and Time

09-NOV-1999 21:23:45

Example

- SELECT SYSDATE, LAST_DAY(SYSDATE) "Last", LAST_DAY(SYSDATE) - SYSDATE "Days Left" FROM DUAL;

```
SYSDATE    Last Days      Left
---------  ---------      ----------
30-MAY-01  31-MAY-01  1
```

Conversion Functions

- CHARTOROWID
- CONVERT
- HEXTORAW
- RAWTOHEX
- ROWIDTOCHAR

Conversion Functions

- TO_CHAR
- TO_DATE
- TO_LABEL
- TO_MULTI_BYTE
- TO_NUMBER
- TO_SINGLE_BYTE

Examples

- SELECT TO_DATE('January 15, 1989, 11:00 A.M.', 'Month dd, YYYY, HH:MI A.M.', 'NLS_DATE_LANGUAGE = American') FROM DUAL;

TO_DATE('

15-JAN-89

Group Functions

- AVG
- COUNT
- GLB
- LUB

Group Functions

- MAX
- MIN
- STDDEV
- SUM
- VARIANCE

Error Reporting Functions

- SQLCODE
- SQLERRM

Other Functions

- BFILENAME
- DECODE
- DUMP
- GREATEST
- GREATEST_LB
- LEAST

Other Functions

- LEAST_LB
- NVL
- UID
- USER
- USERENV
- VSIZE

Agenda

- Stored Procedures
- Functions
- Parameters
- Calling Stored Procedures & Functions
- Examples

Stored Procedures

- Named PL/SQL blocks that
  - Are stored in the database
  - May have formal parameters
  - Can return more than one value to the calling program
  - Can be called from
    - within other PL/SQL blocks as a PL/SQL statement by itself
    - SQL> prompt

PL/SQL Block vs. Stored Procedures

Anonymous PL/SQL Block

```
DECLARE

    -- variable declaration
BEGIN
    -- required executable
EXCEPTION
    -- exception handling
END;
/
```

Stored Procedure

```
CREATE OR REPLACE PROCEDURE X
    [(formal_parameters)] AS[IS]
    -- variable declaration
BEGIN
    -- required executable
EXCEPTION
    -- exception handling
END X;
/
```

Parameters

- Parameters are optional
- MUST be given a data type, but must NOT be given a size
- Parameters have 3 modes
  - IN
    - Read-only within procedure/function
    - Default mode (if mode is not explicitly specified)
  - OUT
    - Has an initial value of NULL within the procedure/function
    - Ignores any values that the actual parameters have when the procedure/function is called
    - Can read from and write to
  - IN OUT
    - Value of actual parameters are passed into procedure/function
    - Can read from and write to

Stored Procedure with Parameters

```
CREATE OR REPLACE PROCEDURE X (
    p_Parameter1 IN VARCHAR2,
    p_Parameter2 IN NUMBER,
    p_Parameter3 OUT VARCHAR2,
    p_Parameter4 OUT NOCOPY NUMBER,
    p_Parameter5 IN OUT NUMBER DEFAULT 1) AS

    -- variable declaration
BEGIN
    -- required executable
EXCEPTION
    -- exception handling
END X;
/
```

```
set serveroutput on
CREATE OR REPLACE PROCEDURE BoatReservations(p_Color IN VARCHAR2) AS

        CURSOR c_Reservations IS
                SELECT s.sname, r.day, r.bid
                FROM Sailor s, Reserve r, Boat b
                WHERE r.sid = s.sid
                        AND r.bid = b.bid
                        AND b.color = p_Color;
        v_Reservation       c_Reservations%ROWTYPE;
BEGIN

        OPEN c_Reservations;

        FETCH c_Reservations INTO v_Reservation;

        WHILE c_Reservations%FOUND LOOP
                DBMS_OUTPUT.PUT_LINE(v_Reservation.sname||' '||v_Reservation.day||'
'||v_Reservation.bid);
                FETCH c_Reservations INTO v_Reservation;
        END LOOP;
        CLOSE c_Reservations;
END BoatReservations;
/
```

Functions

- Named PL/SQL blocks that
  - Are stored in the database
  - May have formal parameters
  - <u>MUST</u> use the keyword **RETURN** to return only one value
    - RETURN passes control back to the calling program
    - Required for functions
  - Can be called from
    - within other PL/SQL blocks as part of an expression
    - SQL> prompt

Stored Procedures vs. Functions

Stored Procedure

```
CREATE OR REPLACE PROCEDURE X
    [(parameters)] AS

    -- variable declaration
BEGIN
    -- required executable



EXCEPTION
    -- exception handling
END X;
/
```

Function

```
CREATE OR REPLACE FUNCTION X
    [(formal_parameters)] RETURN
    return_type IS[AS]
    -- variable declaration
BEGIN
    -- required executable

    -- required RETURN statement
    RETURN Z;
EXCEPTION
    -- exception handling
END X;
/
```

```
CREATE OR REPLACE  FUNCTION NextBusinessDate1 (p_Date DATE)
RETURN DATE IS

        -- Variable that will contain the day that corresponds to the date parameter
        v_CurrentDay            VARCHAR2(9);

        -- Variable that will contain the computed date of the next business day
        v_NextDate              DATE;
BEGIN


        /*First, determine the corresponding name of the day for the date parameter.  It will be used
later
        to determine the number of days by which the date should be incremented.*/
        v_CurrentDay := UPPER(TRIM(TO_CHAR(p_Date, 'DAY')));


        /*Based upon the name of the day and the business rule, calculate the next business date*/
        IF v_CurrentDay = 'FRIDAY' THEN
                v_NextDate := p_Date + 3;
        ELSIF v_CurrentDay = 'SATURDAY' THEN
                v_NextDate := p_Date + 2;
        ELSE
                v_NextDate := p_Date + 1;
        END IF;


        -- Now, return the computed next business date to the calling program
        RETURN v_NextDate;

END NextBusinessDate1;
/
```

TRIM and TO_CHAR functions

- TRIM(string)

  Removes leading and trailing blanks

- TO_CHAR(date, ‘format’)

  See Table 5-4 for a list of valid formats

  The date field in the reservation table has been populated, but the weekday field is NULL.

  Write a query to populate the weekday field with the name of the day that corresponds to the date specified in the date field.

  UPDATE reservation SET weekday = TRIM(TO_CHAR(date, ‘DAY’));

NOTE: The ‘DAY’ format returns the name of the day with blanks padded on the right such that the length is 9 characters.

Parameters

- May be passed by value or by reference
  - IN → by default, passed by reference
  - OUT → by default, passed by value
  - IN OUT → by default, passed by value

  - Passing by reference results in faster performance

- NOCOPY
  - A *compiler hint* to pass OUT & IN OUT parameters by reference
  - Cannot use NOCOPY with IN parameters
  - Ex:
    - (P_outParameter IN OUT **NOCOPY** VARCHAR2) IS

Parameters

- Formal parameters can have default values
  - Formal parameters with default values must appear as the last items in the parameter list
- When calling a stored procedure or function, the actual arguments can be passed by positional or named notation

Calling Stored Procedures & Functions

- With Parameters
  - Stored Procedure from SQL> prompt
    - CALL X(v_Variable1, …., v_VariableN);
      OR CALL X(p_Parameter1 => v_Variable1,…);
    - **EXEC X(v_Variable1,….,v_VariableN);**
  - Stored Procedure from within PL/SQL block
    - EXECUTE IMMEDIATE 'CALL X(……..)'; OR
      X(v_Variable1,….,v_VariableN);

Function

    - Used in an expression
      - SELECT ElapsedDays('01-JAN-1999') FROM dual;
- Without Parameters
  - If the stored procedure (or function) does not have parameters, then do not use parentheses to define or call the stored procedure (or function)