

```
CREATE OR REPLACE PACKAGE PKG_CSV
IS
```

```
/*
 * Valori di default
 */
DEFAULT_SEPARATOR          CONSTANT CHAR(1) := ',';
DEFAULT_DECIMAL_SEPARATOR CONSTANT CHAR(1) := ',';
DEFAULT_MASK_DATE         CONSTANT VARCHAR2(50) := 'DD/MM/YYYY HH24.MI:SS';
DEFAULT_MASK_TIMESTAMP    CONSTANT VARCHAR2(50) := 'DD/MM/YYYY HH24.MI:SS.FF6';
DEFAULT_QUOTE              CONSTANT CHAR(1) := NULL;
DEFAULT_TRIM               CONSTANT VARCHAR2(10) := NULL;
DEFAULT_SHOW_HEADER       BOOLEAN := TRUE;
DEFAULT_SHOW_NULLS        BOOLEAN := FALSE;
DEFAULT_END_WITH_SEPARATOR BOOLEAN := FALSE;
DEFAULT_LINE_BREAK_STYLE  VARCHAR2(5) := 'win';
```

```
/*
 * Principali tipi di dati
 */
TYPE_VARCHAR2  CONSTANT PLS_INTEGER := 1;
TYPE_NUMBER    CONSTANT PLS_INTEGER := 2;
TYPE_DATE      CONSTANT PLS_INTEGER := 12;
TYPE_TIMESTAMP CONSTANT PLS_INTEGER := 181;
TYPE_YMININTERVAL CONSTANT PLS_INTEGER := 182;
TYPE_DSINTERVAL CONSTANT PLS_INTEGER := 183;
TYPE_CHAR      CONSTANT PLS_INTEGER := 96;
TYPE_CLOB      CONSTANT PLS_INTEGER := 112;
TYPE_BLOB      CONSTANT PLS_INTEGER := 113;
TYPE_BFILE     CONSTANT PLS_INTEGER := 114;
TYPE_LONG      CONSTANT PLS_INTEGER := 8;
TYPE_RAW       CONSTANT PLS_INTEGER := 23;
TYPE_ROWID     CONSTANT PLS_INTEGER := 11;
```

```
/*
 * Altre costanti
 */
```

```
CRLF_WIN    CONSTANT CHAR(2) := CHR(13) || CHR(10);
CRLF_UNIX   CONSTANT CHAR(1) := CHR(10);
```

```
/*
 * Scrittura su disco di un file CSV.
 *
 * i_folder --> nome logico della directory di output
 * i_filename --> nome del file di export
 * i_query --> source query dei dati da esportare
 * i_separator --> separatore di campo da adottare
 * i_decimal_separator --> separatore per i numeri decimali
 * i_mask_date --> formattazione delle date
 * i_mask_timestamp --> formattazione dei timestamp
 * i_quote --> carattere di wrap dei campi (ammessi " o ')
 * i_trim --> elimina gli spazi nei campi (ammessi 'leading', 'trailing', 'both')
 * i_show_header --> include il nome delle colonne
 * i_show_nulls --> scrive 'null' in luogo di un null value
 * i_end_with_separator --> include il separatore dopo l'ultimo campo
 * i_line_break_style --> stile del separatore di linea (ammessi 'win' o 'unix')
 */
```

```
PROCEDURE Write_File(
    i_folder VARCHAR2,
    i_file_name VARCHAR2,
    i_query VARCHAR2,
    i_separator CHAR DEFAULT DEFAULT_SEPARATOR,
    i_decimal_separator CHAR DEFAULT DEFAULT_DECIMAL_SEPARATOR,
    i_mask_date VARCHAR2 DEFAULT DEFAULT_MASK_DATE,
    i_mask_timestamp VARCHAR2 DEFAULT DEFAULT_MASK_TIMESTAMP,
    i_quote CHAR DEFAULT DEFAULT_QUOTE,
    i_trim VARCHAR2 DEFAULT DEFAULT_TRIM,
    i_show_header BOOLEAN DEFAULT DEFAULT_SHOW_HEADER,
    i_show_nulls BOOLEAN DEFAULT DEFAULT_SHOW_NULLS,
    i_end_with_separator BOOLEAN DEFAULT DEFAULT_END_WITH_SEPARATOR,
    i_line_break_style VARCHAR2 DEFAULT DEFAULT_LINE_BREAK_STYLE
);
```

```
END;
/
```

```
CREATE OR REPLACE PACKAGE BODY PKG_CSV
IS
```

```
/*
 * Scrittura su disco di un file CSV.
 *
 * i_folder --> nome logico della directory di output
 * i_filename --> nome del file di export
 * i_query --> source query dei dati da esportare
 * i_separator --> separatore di campo da adottare
 * i_decimal_separator --> separatore per i numeri decimali
 * i_mask_date --> formattazione delle date
 * i_mask_timestamp --> formattazione dei timestamp
 * i_quote --> carattere di wrap dei campi (ammessi " o ')
 * i_trim --> elimina gli spazi nei campi (ammessi 'leading', 'trailing', 'both')
 * i_show_header --> include il nome delle colonne
 * i_show_nulls --> scrive 'null' in luogo di un null value
 * i_end_with_separator --> include il separatore dopo l'ultimo campo
 * i_line_break_style --> stile del separatore di linea (ammessi 'win' o 'unix')
 */
```

```
PROCEDURE Write_File(
    i_folder VARCHAR2,
    i_file_name VARCHAR2,
    i_query VARCHAR2,
    i_separator CHAR DEFAULT DEFAULT_SEPARATOR,
    i_decimal_separator CHAR DEFAULT DEFAULT_DECIMAL_SEPARATOR,
    i_mask_date VARCHAR2 DEFAULT DEFAULT_MASK_DATE,
    i_mask_timestamp VARCHAR2 DEFAULT DEFAULT_MASK_TIMESTAMP,
    i_quote CHAR DEFAULT DEFAULT_QUOTE,
    i_trim VARCHAR2 DEFAULT DEFAULT_TRIM,
    i_show_header BOOLEAN DEFAULT DEFAULT_SHOW_HEADER,
    i_show_nulls BOOLEAN DEFAULT DEFAULT_SHOW_NULLS,
    i_end_with_separator BOOLEAN DEFAULT DEFAULT_END_WITH_SEPARATOR,
    i_line_break_style VARCHAR2 DEFAULT DEFAULT_LINE_BREAK_STYLE
)
```

```
IS
```

```
-- Handler del file
```

```

v_file UTL_FILE.FILE_TYPE;
-- Handler della query
v_cursor NUMBER;
v_return NUMBER;
-- Handlers delle colonne
c_columns DBMS_SQL.DESCR_TAB2;
v_num_columns PLS_INTEGER;
-- Handler della riga
v_row VARCHAR2(32767);
-- Handlers dei campi
v_field VARCHAR2(4000);
v_field_varchar2 VARCHAR2(4000);
v_field_timestamp TIMESTAMP;
v_field_date DATE;
v_field_number NUMBER;
v_local_decimal_separator CHAR(1);
-- Formattazione dei campi
v_separator CHAR(1) := TRIM(i_separator);
v_decimal_separator CHAR(1) := TRIM(i_decimal_separator);
v_mask_date VARCHAR2(50) := TRIM(i_mask_date);
v_mask_timestamp VARCHAR2(50) := TRIM(i_mask_timestamp);
v_quote CHAR(1) := TRIM(i_quote);
v_trim VARCHAR2(50) := LOWER(TRIM(i_trim));
v_show_header BOOLEAN := i_show_header;
v_show_nulls BOOLEAN := i_show_nulls;
v_end_with_separator BOOLEAN := i_end_with_separator;
v_line_break_style VARCHAR2(50) := LOWER(TRIM(i_line_break_style));

```

```
BEGIN
```

```

/*
 * Gestione dei parametri di input a fronte di NULL o di valori non ammissibili
 * erroneamente passati dall'utente
 */
v_separator := NVL(v_separator, DEFAULT_SEPARATOR);
v_decimal_separator := NVL(v_decimal_separator, DEFAULT_DECIMAL_SEPARATOR);
v_mask_date := NVL(v_mask_date, DEFAULT_MASK_DATE);
v_mask_timestamp := NVL(v_mask_timestamp, DEFAULT_MASK_TIMESTAMP);
CASE WHEN
    v_quote IS NULL OR v_quote NOT IN ('"', '''') THEN v_quote := DEFAULT_QUOTE;
ELSE NULL;
END CASE;
CASE WHEN

```

```

        v_trim IS NULL OR v_trim NOT IN ('leading', 'trailing', 'both') THEN v_trim := DEFAULT_TRIM;
        ELSE NULL;
    END CASE;
    v_show_header := NVL(v_show_header, DEFAULT_SHOW_HEADER);
    v_show_nulls := NVL(v_show_nulls, DEFAULT_SHOW_NULLS);
    v_end_with_separator := NVL(v_end_with_separator, DEFAULT_END_WITH_SEPARATOR);
    CASE WHEN
        v_line_break_style IS NULL OR v_line_break_style NOT IN ('win', 'unix') THEN v_line_break_style :=
            DEFAULT_LINE_BREAK_STYLE;
        ELSE NULL;
    END CASE;

    dbms_output.put_line('libe break = ' || v_line_break_style);

    v_file := UTL_FILE.FOpen(i_folder, i_file_name, 'W');

    /*
     * Lettura header colonne
     */
    v_cursor := DBMS_SQL.Open_Cursor;
    DBMS_SQL.Parse(v_cursor, i_query, DBMS_SQL.NATIVE);
    DBMS_SQL.Describe_Columns2(v_cursor, v_num_columns, c_columns);

    v_row := '';
    FOR i IN c_columns.FIRST .. c_columns.LAST
    LOOP
        v_field := c_columns(i).COL_NAME;
        -- Quoting?
        IF i_quote IS NOT NULL THEN
            v_field := i_quote || v_field || i_quote;
        END IF;
        -- Concatenazione
        v_row := v_row || v_field || i_separator;
    END LOOP;
    -- Separatore finale?
    IF NOT i_end_with_separator THEN
        v_row := TRIM(TRAILING i_separator FROM v_row);
    END IF;
    -- Scrittura header?
    IF v_show_header THEN
        IF v_line_break_style = 'win' THEN
            UTL_FILE.Put(v_file, v_row || CRLF_WIN);
        ELSE

```

```

        UTL_FILE.Put(v_file, v_row || CRLF_UNIX);
    END IF;
    UTL_FILE.FFlush(v_file);
END IF;

/*
 * Binding colonne del cursore con le variabili
 */
FOR i IN 1 .. v_num_columns
LOOP
    CASE c_columns(i).COL_TYPE
        WHEN TYPE_DATE THEN
            -- Data/Ora
            DBMS_SQL.Define_Column(v_cursor, i, v_field_date);
        WHEN TYPE_TIMESTAMP THEN
            -- Timestamp
            DBMS_SQL.Define_Column(v_cursor, i, v_field_timestamp);
        ELSE
            -- Altro
            DBMS_SQL.Define_Column(v_cursor, i, v_field_varchar2, 4000);
        END CASE;
    END LOOP;

SELECT SUBSTR(VALUE, 1, 1)
INTO v_local_decimal_separator
FROM V$NLS_PARAMETERS
WHERE PARAMETER = 'NLS_NUMERIC_CHARACTERS';

/*
 * Lettura dei records
 */
v_return := DBMS_SQL.Execute(v_cursor);
LOOP
    v_return := DBMS_SQL.Fetch_Rows(v_cursor);
    EXIT WHEN v_return = 0;
    v_row := '';
    FOR i IN 1 .. v_num_columns
    LOOP
        -- Formattazione
        CASE c_columns(i).COL_TYPE
            WHEN TYPE_DATE THEN
                -- Data/Ora
                DBMS_SQL.Column_Value(v_cursor, i, v_field_date);

```

```

        v_field := TO_CHAR(v_field_date, v_mask_date);
    WHEN TYPE_TIMESTAMP THEN
        -- Timestamp
        DBMS_SQL.Column_Value(v_cursor, i, v_field_timestamp);
        v_field := TO_CHAR(v_field_timestamp, v_mask_timestamp);
    WHEN TYPE_NUMBER THEN
        -- Numero
        DBMS_SQL.Column_Value(v_cursor, i, v_field_varchar2);
        v_field := REPLACE(v_field_varchar2, v_local_decimal_separator, v_decimal_separator);
        ELSE
        -- Altro
        DBMS_SQL.Column_Value(v_cursor, i, v_field);
    END CASE;
    -- NULL handling
    IF i_show_nulls AND v_field IS NULL THEN
        v_field := 'null';
    END IF;
    -- Trim?
    CASE LOWER(TRIM(i_trim))
        WHEN 'leading' THEN
            v_field := TRIM(LEADING ' ' FROM v_field);
        WHEN 'trailing' THEN
            v_field := TRIM(TRAILING ' ' FROM v_field);
        WHEN 'both' THEN
            v_field := TRIM(BOTH ' ' FROM v_field);
        ELSE
            NULL;
    END CASE;
    -- Quoting?
    IF i_quote IS NOT NULL THEN
        v_field := i_quote || v_field || i_quote;
    END IF;
    -- Concatenazione
    v_row := v_row || v_field || i_separator;
END LOOP;
-- Separatore finale?
IF NOT i_end_with_separator THEN
    v_row := TRIM(TRAILING i_separator FROM v_row);
END IF;
-- Scrittura su disco!
IF v_line_break_style = 'win' THEN
    UTL_FILE.Put(v_file, v_row || CRLF_WIN);
ELSE

```

```
        UTL_FILE.Put(v_file, v_row || CRLF_UNIX);
    END IF;
    UTL_FILE.FFlush(v_file);
END LOOP;
```

```
DBMS_SQL.Close_Cursor(v_cursor);
UTL_FILE.FClose(v_file);
```

```
EXCEPTION
```

```
    WHEN OTHERS THEN
```

```
        IF DBMS_SQL.Is_Open(v_cursor) THEN
            DBMS_SQL.Close_Cursor(v_cursor);
            UTL_FILE.FClose(v_file);
        END IF;
        RAISE;
```

```
END Write_File;
```

```
END;
```

```
/
```