

## Anexo: Funciones ODBC

---

### Resumen

---

La interfase ODBC nos provee una gran cantidad de funciones muy utiles que en general son poco conocidas.

Para llamar a una funcion ODBC la sintaxis es:

```
SELECT {fn NOMBRE_FUNCION(PARAMETROS)} FROM ...
```

Por ejemplo:

```
SELECT NAME, {fn TIMESTAMPADD(SQL_TSI_YEAR,  
1, HIRE_DATE)} FROM EMPLOYEES
```

La ventaja adicional del uso de estas funciones es que son independientes del motor SQL.

Por lo tanto:

```
SELECT {fn curdate() }
```

```
SELECT {fn curtime() }
```

Van a funcionar igual en todos los motores, en vez de usar la sintaxis especifica para pedir la fecha en cada uno de ellos.

Debajo de la funcion se indica con que version de ODBC son compatibles. Esto es importante ya que cada driver ODBC indica cual es la version que soporta.

### Funciones String

---

#### **ASCII(string\_exp)**

(ODBC 1.0) Returns the ASCII code value of the leftmost character of string\_exp as an integer.

#### **BIT\_LENGTH(string\_exp)**

(ODBC 3.0) Returns the length in bits of the string expression.

#### **CHAR(code)**

(ODBC 1.0) Returns the character that has the ASCII code value specified by code. The value of code should be between 0 and 255; otherwise, the return value is data source-dependent.

#### **CHAR\_LENGTH(string\_exp)**

(ODBC 3.0) Returns the length in characters of the string expression, if the string expression is of a character data type; otherwise, returns the length in bytes of the string expression (the smallest integer not less than the number of bits divided by 8). (This function is the same as the CHARACTER\_LENGTH function.)

**CHARACTER\_LENGTH(string\_exp)**

(ODBC 3.0) Returns the length in characters of the string expression, if the string expression is of a character data type; otherwise, returns the length in bytes of the string expression (the smallest integer not less than the number of bits divided by 8). (This function is the same as the CHAR\_LENGTH function.)

**CONCAT(string\_exp1, string\_exp2)**

(ODBC 1.0) Returns a character string that is the result of concatenating string\_exp2 to string\_exp1. The resulting string is DBMS-dependent. For example, if the column represented by string\_exp1 contained a NULL value, DB2 would return NULL but SQL Server would return the non-NULL string.

**DIFFERENCE(string\_exp1, string\_exp2)**

(ODBC 2.0) Returns an integer value that indicates the difference between the values returned by the SOUNDEX function for string\_exp1 and string\_exp2.

**INSERT(string\_exp1, start, length, string\_exp2)**

(ODBC 1.0) Returns a character string where length characters have been deleted from string\_exp1, beginning at start, and where string\_exp2 has been inserted into string\_exp, beginning at start.

**LCASE(string\_exp)**

(ODBC 1.0) Returns a string equal to that in string\_exp, with all uppercase characters converted to lowercase.

**LEFT(string\_exp, count)**

(ODBC 1.0) Returns the leftmost count characters of string\_exp.

**LENGTH(string\_exp)**

(ODBC 1.0) Returns the number of characters in string\_exp, excluding trailing blanks.

**LOCATE(string\_exp1, string\_exp2[, start])**

(ODBC 1.0) Returns the starting position of the first occurrence of string\_exp1 within string\_exp2. The search for the first occurrence of string\_exp1 begins with the first character position in string\_exp2 unless the optional argument, start, is specified. If start is specified, the search begins with the character position indicated by the value of start. The first character position in string\_exp2 is indicated by the value 1. If string\_exp1 is not found within string\_exp2, the value 0 is returned.

If an application can call the LOCATE scalar function with the string\_exp1, string\_exp2, and start arguments, the driver returns SQL\_FN\_STR\_LOCATE when SQLGetInfo is called with an

Option of SQL\_STRING\_FUNCTIONS. If the application can call the LOCATE scalar function with only the string\_exp1 and string\_exp2 arguments, the driver returns SQL\_FN\_STR\_LOCATE\_2 when SQLGetInfo is called with an Option of SQL\_STRING\_FUNCTIONS. Drivers that support calling the LOCATE function with either two or three arguments return both SQL\_FN\_STR\_LOCATE and SQL\_FN\_STR\_LOCATE\_2.

**LTRIM(string\_exp)**

(ODBC 1.0) Returns the characters of string\_exp, with leading blanks removed.

**OCTET\_LENGTH(string\_exp)**

(ODBC 3.0) Returns the length in bytes of the string expression. The result is the smallest integer not less than the number of bits divided by 8.

**POSITION(character\_exp IN character\_exp)**

(ODBC 3.0) Returns the position of the first character expression in the second character expression. The result is an exact numeric with an implementation-defined precision and a scale of 0.

**REPEAT(string\_exp, count)**

(ODBC 1.0) Returns a character string composed of string\_exp repeated count times.

**REPLACE(string\_exp1, string\_exp2, string\_exp3)**

(ODBC 1.0) Search string\_exp1 for occurrences of string\_exp2, and replace with string\_exp3.

**RIGHT(string\_exp, count)**

(ODBC 1.0) Returns the rightmost count characters of string\_exp.

**RTRIM(string\_exp)**

(ODBC 1.0) Returns the characters of string\_exp with trailing blanks removed.

**SOUNDEX(string\_exp)**

(ODBC 2.0) Returns a data source-dependent character string representing the sound of the words in string\_exp. For example, SQL Server returns a 4-digit SOUNDEX code; Oracle returns a phonetic representation of each word.

**SPACE(count) (ODBC 2.0)**

Returns a character string consisting of count spaces.

**SUBSTRING(string\_exp, start, length)**

(ODBC 1.0) Returns a character string that is derived from string\_exp, beginning at the character position specified by start for length characters.

**UCASE(string\_exp)**

(ODBC 1.0) Returns a string equal to that in string\_exp, with all lowercase characters converted to uppercase.

---

**Funciones Numericas**

---

**ABS(numeric\_exp)**

(ODBC 1.0) Returns the absolute value of numeric\_exp.

**ACOS(float\_exp)**

(ODBC 1.0) Returns the arccosine of float\_exp as an angle, expressed in radians.

**ASIN(float\_exp)**

(ODBC 1.0) Returns the arcsine of float\_exp as an angle, expressed in radians.

**ATAN(float\_exp)**

(ODBC 1.0) Returns the arctangent of float\_exp as an angle, expressed in radians.

**ATAN2(float\_exp1, float\_exp2)**

(ODBC 2.0) Returns the arctangent of the x and y coordinates, specified by float\_exp1 and float\_exp2, respectively, as an angle, expressed in radians.

**CEILING(numeric\_exp)**

(ODBC 1.0) Returns the smallest integer greater than or equal to numeric\_exp. The return value is of the same data type as the input parameter.

**COS(float\_exp)**

(ODBC 1.0) Returns the cosine of float\_exp, where float\_exp is an angle expressed in radians.

**COT(float\_exp)**

(ODBC 1.0) Returns the cotangent of float\_exp, where float\_exp is an angle expressed in radians.

**DEGREES(numeric\_exp)**

(ODBC 2.0) Returns the number of degrees converted from numeric\_exp radians.

**EXP(float\_exp)**

(ODBC 1.0) Returns the exponential value of float\_exp.

**FLOOR(numeric\_exp)**

(ODBC 1.0) Returns the largest integer less than or equal to numeric\_exp. The return value is of the same data type as the input parameter.

**LOG(float\_exp)**

(ODBC 1.0) Returns the natural logarithm of float\_exp.

**LOG10(float\_exp)**

(ODBC 2.0) Returns the base 10 logarithm of float\_exp.

**MOD(integer\_exp1, integer\_exp2)**

(ODBC 1.0) Returns the remainder (modulus) of integer\_exp1 divided by integer\_exp2.

**PI( )**

(ODBC 1.0) Returns the constant value of pi as a floating-point value.

**POWER(numeric\_exp, integer\_exp)**

(ODBC 2.0) Returns the value of numeric\_exp to the power of integer\_exp.

**RADIANS(numeric\_exp)**

(ODBC 2.0) Returns the number of radians converted from numeric\_exp degrees.

**RAND([integer\_exp])**

(ODBC 1.0) Returns a random floating-point value using integer\_exp as the optional seed value.

**ROUND(numeric\_exp, integer\_exp)**

(ODBC 2.0) Returns numeric\_exp rounded to integer\_exp places right of the decimal point. If integer\_exp is negative, numeric\_exp is rounded to |integer\_exp| places to the left of the decimal point.

**SIGN(numeric\_exp)**

(ODBC 1.0) Returns an indicator of the sign of numeric\_exp. If numeric\_exp is less than zero, -1 is returned. If numeric\_exp equals zero, 0 is returned. If numeric\_exp is greater than zero, 1 is returned.

**SIN(float\_exp)**

(ODBC 1.0) Returns the sine of float\_exp, where float\_exp is an angle expressed in radians.

**SQRT(float\_exp)**

(ODBC 1.0) Returns the square root of float\_exp.

**TAN(float\_exp)**

(ODBC 1.0) Returns the tangent of float\_exp, where float\_exp is an angle expressed in radians.

**TRUNCATE(numeric\_exp, integer\_exp)**

(ODBC 2.0) Returns numeric\_exp truncated to integer\_exp places right of the decimal point. If integer\_exp is negative, numeric\_exp is truncated to |integer\_exp| places to the left of the decimal point.

---

**Funciones de Fecha y Hora**

---

**CURRENT\_DATE( )**

(ODBC 3.0) Returns the current date.

**CURRENT\_TIME[(time-precision)]**

(ODBC 3.0) Returns the current local time. The time-precision argument determines the seconds precision of the returned value.

**CURRENT\_TIMESTAMP**

[(timestamp-precision)]

(ODBC 3.0) Returns the current local date and local time as a timestamp value. The timestamp-precision argument determines the seconds precision of the returned timestamp.

**CURDATE( )**

(ODBC 1.0) Returns the current date.

**CURTIME( )**

(ODBC 1.0) Returns the current local time.

**DAYNAME(date\_exp)**

(ODBC 2.0) Returns a character string containing the data source-specific name of the day (for example, Sunday through Saturday or Sun. through Sat. for a data source that uses English, or Sonntag through Samstag for a data source that uses German) for the day portion of date\_exp.

**DAYOFMONTH(date\_exp)**

(ODBC 1.0) Returns the day of the month based on the month field in date\_exp as an integer value in the range of 1–31.

**DAYOFWEEK(date\_exp)**

(ODBC 1.0) Returns the day of the week based on the week field in date\_exp as an integer value in the range of 1–7, where 1 represents Sunday.

**DAYOFYEAR(date\_exp)**

(ODBC 1.0) Returns the day of the year based on the year field in date\_exp as an integer value in the range of 1–366.

**EXTRACT(extract-field FROM extract-source)**

(ODBC 3.0) Returns the extract-field portion of the extract-source. The extract-source argument is a datetime or interval expression. The extract-field argument can be one of the following keywords:

YEAR  
MONTH  
DAY  
HOUR  
MINUTE  
SECOND

The precision of the returned value is implementation-defined. The scale is 0 unless SECOND is specified, in which case the scale is not less than the fractional seconds precision of the extract-source field.

**HOUR(time\_exp)**

(ODBC 1.0) Returns the hour based on the hour field in time\_exp as an integer value in the range of 0–23.

**MINUTE(time\_exp)**

(ODBC 1.0) Returns the minute based on the minute field in time\_exp as an integer value in the range of 0–59.

**MONTH(date\_exp)**

(ODBC 1.0) Returns the month based on the month field in date\_exp as an integer value in the range of 1–12.

**MONTHNAME(date\_exp)**

(ODBC 2.0) Returns a character string containing the data source-specific name of the month (for example, January through December or Jan. through Dec. for a data source that uses English, or Januar through Dezember for a data source that uses German) for the month portion of date\_exp.

**NOW( )**

(ODBC 1.0) Returns current date and time as a timestamp value.

**QUARTER(date\_exp)**

(ODBC 1.0) Returns the quarter in date\_exp as an integer value in the range of 1–4, where 1 represents January 1 through March 31.

**SECOND(time\_exp)**

(ODBC 1.0) Returns the second based on the second field in time\_exp as an integer value in the range of 0–59.

Returns the timestamp calculated by adding integer\_exp intervals of type interval to timestamp\_exp. Valid values of interval are the following keywords:

```
SQL_TSI_FRAC_SECOND
SQL_TSI_SECOND
SQL_TSI_MINUTE
SQL_TSI_HOUR
SQL_TSI_DAY
SQL_TSI_WEEK
SQL_TSI_MONTH
SQL_TSI_QUARTER
SQL_TSI_YEAR
```

where fractional seconds are expressed in billionths of a second. For example, the following SQL statement returns the name of each employee and his or her one-year anniversary date:

```
SELECT NAME, {fn TIMESTAMPADD(SQL_TSI_YEAR,
1, HIRE_DATE)} FROM EMPLOYEES
```

If timestamp\_exp is a time value and interval specifies days, weeks, months, quarters, or years, the date portion of timestamp\_exp is set to the current date before calculating the resulting timestamp.



If timestamp\_exp is a date value and interval specifies fractional seconds, seconds, minutes, or hours, the time portion of timestamp\_exp is set to 0 before calculating the resulting timestamp.

An application determines which intervals a data source supports by calling SQLGetInfo with the SQL\_TIMEDATE\_ADD\_INTERVALS option.

### **TIMESTAMPDIFF(interval, timestamp\_exp1, timestamp\_exp2)**

(ODBC 2.0) Returns the integer number of intervals of type interval by which timestamp\_exp2 is greater than timestamp\_exp1. Valid values of interval are the following keywords:

```
SQL_TSI_FRAC_SECOND  
SQL_TSI_SECOND  
SQL_TSI_MINUTE  
SQL_TSI_HOUR  
SQL_TSI_DAY  
SQL_TSI_WEEK  
SQL_TSI_MONTH  
SQL_TSI_QUARTER  
SQL_TSI_YEAR
```

where fractional seconds are expressed in billionths of a second. For example, the following SQL statement returns the name of each employee and the number of years he or she has been employed:

```
SELECT NAME, {fn  
TIMESTAMPDIFF(SQL_TSI_YEAR,  
{fn CURDATE()}, HIRE_DATE)}  
FROM EMPLOYEES
```

If either timestamp expression is a time value and interval specifies days, weeks, months, quarters, or years, the date portion of that timestamp is set to the current date before calculating the difference between the timestamps.

If either timestamp expression is a date value and interval specifies fractional seconds, seconds, minutes, or hours, the time portion of that timestamp is set to 0 before calculating the difference between the timestamps.

An application determines which intervals a data source supports by calling SQLGetInfo with the SQL\_TIMEDATE\_DIFF\_INTERVALS option.

**WEEK(date\_exp)**

(ODBC 1.0) Returns the week of the year based on the week field in date\_exp as an integer value in the range of 1–53.

**YEAR(date\_exp)**

(ODBC 1.0) Returns the year based on the year field in date\_exp as an integer value. The range is data source-dependent.

---

**Funciones del Sistema**

---

**DATABASE( )**

(ODBC 1.0) Returns the name of the database corresponding to the connection handle. (The name of the database is also available by calling SQLGetConnectOption with the SQL\_CURRENT\_QUALIFIER connection option.)

**IFNULL(exp, value)**

(ODBC 1.0) If exp is null, value is returned. If exp is not null, exp is returned. The possible data type or types of value must be compatible with the data type of exp.

**USER( )**

(ODBC 1.0) Returns the user name in the DBMS. (The user name is also available by way of SQLGetInfo by specifying the information type: SQL\_USER\_NAME.) This can be different than the login name.