



Informatica®

10.4.0

Referencia del lenguaje de transformación

Este software y la documentación se proporcionan exclusivamente en virtud de un acuerdo de licencia independiente que contiene restricciones de uso y divulgación. Ninguna parte de este documento puede ser reproducida o transmitida de cualquier forma o manera (electrónica, fotocopia, grabación o mediante otros métodos) sin el consentimiento previo de Informatica LLC.

Informatica y el logotipo de Informatica son marcas comerciales o marcas comerciales registradas de Informatica LLC en Estados Unidos y en las diversas jurisdicciones de todo el mundo. La lista actual de marcas comerciales de Informatica está disponible en Internet en <https://www.informatica.com/trademarks.html>. Otros nombres de productos y empresas pueden ser nombres o marcas comerciales de sus respectivos titulares.

Las bases de datos, el software y los programas de DERECHOS DEL GOBIERNO DE LOS ESTADOS UNIDOS, y la documentación e información técnica relacionadas entregadas a los clientes del Gobierno de los Estados Unidos constituyen "software informático comercial" o "datos técnicos comerciales" de acuerdo con el Reglamento de Adquisición Federal y las regulaciones complementarias específicas del organismo que correspondan. Como tales, el uso, la duplicación, la divulgación, la modificación y la adaptación están sujetos a las restricciones y los términos de licencia establecidos en el contrato gubernamental aplicable, y hasta donde sea aplicable en función de los términos del contrato gubernamental, a los derechos adicionales establecidos en FAR 52.227-19, Licencia de Software Informático Comercial.

Las partes de este software o la documentación están sujetas a derechos de autor de terceros. Se incluyen con el producto los avisos obligatorios de terceros.

La información contenida en esta documentación está sujeta a cambios sin previo aviso. Si encuentra algún problema en esta documentación, escríbanos a infa_documentation@informatica.com para notificarnoslo.

Los productos de Informatica gozan de garantía en función de los términos y condiciones de los acuerdos conforme a los cuales se proporcionen. INFORMATICA PROPORCIONA LA INFORMACIÓN DE ESTE DOCUMENTO "TAL CUAL" SIN GARANTÍA DE NINGÚN TIPO, EXPRESA O IMPLÍCITA, INCLUIDAS LAS GARANTÍAS DE COMERCIALIZACIÓN, ADAPTACIÓN A UN FIN PARTICULAR Y CUALQUIER GARANTÍA O CONDICIÓN DE NO INCUMPLIMIENTO.

Tabla de contenido

Prefacio	9
Recursos de Informatica	9
Informatica Network.	9
Base de conocimiento de Informatica.	9
Documentación de Informatica	9
Matrices de disponibilidad de producto de Informatica.	10
Informatica Velocity.	10
Catálogo de soluciones de Informatica.	10
Servicio internacional de atención al cliente de Informatica.	10
 Capítulo 1: Lenguaje de transformación.....	 11
Resumen del idioma de transformación.	11
Componentes del idioma de transformación.	11
Internacionalización y el idioma de transformación.	12
Sintaxis de expresiones.	12
Componentes de expresiones.	13
Reglas y pautas sobre la sintaxis de expresiones.	14
Agregar comentarios a las expresiones.	15
Palabras reservadas.	15
 Capítulo 2: Constantes.....	 17
DD_DELETE.	17
Ejemplo.	17
DD_INSERT.	17
Ejemplos.	18
DD_REJECT.	18
Ejemplos.	18
DD_UPDATE.	19
Ejemplos.	19
FALSE.	19
Ejemplo.	19
NULL.	20
Modo de trabajo con valores nulos en expresiones booleanas.	20
Valores nulos en expresiones de comparación.	20
Valores nulos en funciones de agregado.	20
Valores nulos en condiciones de filtro.	20
Valores nulos con operadores.	21
TRUE.	21
Ejemplo.	21

Capítulo 3: Operadores.....	22
Precedencia del operador.	22
Operadores complejos.	23
Operador de subíndice.	24
Operador de punto.	25
Operadores complejos para tipos de datos anidados.	27
Operadores aritméticos.	31
Operadores de cadena.	32
Valores nulos.	32
Ejemplo.	33
Operadores de comparación.	33
Operadores lógicos.	34
Valores nulos.	35
 Capítulo 4: Variables.....	 36
Variables integradas.	36
SYSDATE.	36
Variables locales.	36
 Capítulo 5: Fechas.....	 37
Introducción a las fechas.	37
Tipo de datos de fecha y hora.	37
Día juliano, día juliano modificado y calendario gregoriano.	38
Fechas del año 2000.	38
Fechas de bases de datos relacionales.	40
Fechas de archivos sin formato.	40
Formato de fecha predeterminado.	40
Cadenas de formato de fecha.	41
Cadenas de formato TO_CHAR.	42
Ejemplos.	44
Cadenas de formato TO_DATE e IS_DATE.	45
Normas y directrices para cadenas con formato de fecha.	47
Ejemplo.	47
Descripción de operaciones aritméticas con fechas.	49
 Capítulo 6: Funciones.....	 50
Categorías de funciones.	50
Funciones de agregado.	50
Funciones de agregado y valores nulos.	52
Funciones de caracteres.	52
Funciones complejas.	53
Funciones de conversión.	54

Funciones de limpieza de datos.	54
Funciones de fecha.	55
Funciones de codificación.	55
Funciones financieras.	56
Funciones numéricas.	56
Funciones científicas.	57
Funciones especiales.	57
Funciones de cadena.	57
Funciones de prueba.	57
Funciones de ventana.	58
ABORT.	58
ABS.	59
ADD_TO_DATE.	60
AES_DECRYPT.	63
AES_ENCRYPT.	63
ANY.	64
ARRAY.	66
ASCII.	66
AVG.	68
CAST.	69
CEIL.	70
CHOOSE.	71
CHR.	72
CHRCODE.	73
COLLECT_LIST.	74
COLLECT_MAP.	74
COMPRESS.	75
CONCAT.	76
CONCAT_ARRAY.	78
CONVERT_BASE.	78
COS.	79
COSH.	80
COUNT.	81
CRC32.	84
CREATE_TIMESTAMP_TZ.	84
CUME.	85
DATE_COMPARE.	87
DATE_DIFF.	88
DEC_BASE64.	91
DECODE.	92
DECOMPRESS.	94
ENC_BASE64.	94

ERROR.	95
EXP.	96
FIRST.	97
FLOOR.	99
FV.	100
GET_DATE_PART.	101
GET_TIMEZONE.	102
GET_TIMESTAMP.	103
GREATEST.	104
IIF.	105
IN.	108
INDEXOF.	109
INITCAP.	110
INSTR.	111
ISNULL.	114
IS_DATE.	116
IS_NUMBER.	118
IS_SPACES.	120
LAG.	121
LAST.	122
LAST_DAY.	123
LEAD.	125
LEAST.	126
LENGTH.	127
LN.	128
LOG.	129
LOWER.	130
LPAD.	131
LTRIM.	133
MAKE_DATE_TIME.	134
MAP.	135
MAP_FROM_ARRAYS.	136
MAP_KEYS.	137
MAP_VALUES.	138
MAX (Fechas).	139
MAX (Números).	140
MAX (Cadena).	141
MD5.	143
MEDIAN.	143
METAPHONE.	145
MIN (Fechas).	148
MIN (Números).	149

MIN (Cadena).	151
MOD.	152
MOVINGAVG.	153
MOVINGSUM.	155
NPER.	156
PARSE_JSON.	157
PARSE_XML.	158
PERCENTILE.	160
PMT.	162
POWER.	163
PV.	164
RAND.	164
RATE.	165
REG_EXTRACT.	166
REG_MATCH.	168
REG_REPLACE.	170
REPLACECHR.	171
REPLACESTR.	174
RESPEC.	177
REVERSE.	178
ROUND (Fechas).	179
ROUND (Números).	183
RPAD.	186
RTRIM.	187
SET_DATE_PART.	188
SIGN.	191
SIN.	192
SINH.	193
SIZE.	194
SOUNDEX.	195
SQL_LIKE.	197
SQRT.	198
STDDEV.	199
STRUCT.	201
STRUCT_AS.	202
SUBSTR.	203
SUM.	205
SYSTIMESTAMP.	206
TAN.	208
TANH.	208
TIME_RANGE.	209
TO_BIGINT.	210

TO_CHAR (Fechas).	212
TO_CHAR (Números).	217
TO_DATE.	218
TO_DECIMAL.	222
TO_DECIMAL38.	223
TO_FLOAT.	224
TO_INTEGER.	225
TO_TIMESTAMP_TZ.	227
TRUNC (Fechas).	228
TRUNC (Números).	231
UPPER.	233
UUID4.	234
UUID_UNPARSE.	234
VARIANCE.	235
Índice.	237

Prefacio

Consulte la *Referencia del lenguaje de transformación de Informatica® Developer* para conocer el lenguaje de transformación en Developer tool. Obtenga información sobre cómo usar constantes, operadores, variables, fechas y funciones para transformar los datos de origen.

Recursos de Informatica

Informatica proporciona una variedad de recursos de productos a través de Informatica Network y otros portales en línea. Use los recursos para sacar el mayor provecho de los productos y las soluciones de Informatica y aprender de otros expertos en la materia y usuarios de Informatica.

Informatica Network

Informatica Network es la puerta de entrada a muchos recursos, entre ellos, la base de conocimientos de Informatica y el servicio internacional de atención al cliente de Informatica. Para entrar en Informatica Network, visite <https://network.informatica.com>.

Como miembro de Informatica Network, tiene las siguientes opciones:

- Buscar recursos de productos en la base de conocimientos
- Ver la información de disponibilidad del producto
- Crear y revisar casos de soporte
- Buscar su red de grupos de usuarios de Informatica locales y colaborar con sus pares

Base de conocimiento de Informatica

Use la base de conocimientos de Informatica para encontrar recursos de productos como artículos prácticos, procedimientos recomendados, tutoriales de video y respuestas a preguntas frecuentes.

Para buscar en la base de conocimiento, visite <https://search.informatica.com>. Si tiene preguntas, comentarios o ideas relacionadas con la base de conocimiento de Informatica, póngase en contacto con el equipo de la base de conocimiento de Informatica en KB_Feedback@informatica.com.

Documentación de Informatica

Use el portal de documentación de Informatica para recorrer una extensa biblioteca de documentación para las versiones de productos actuales y recientes. Para recorrer el portal de documentación, visite <https://docs.informatica.com>.

Si tiene preguntas, comentarios o ideas acerca de la documentación de los productos, póngase en contacto con el equipo de la documentación de Informatica en infa_documentation@informatica.com.

Matrices de disponibilidad de producto de Informatica

Las matrices de disponibilidad de producto (PAM, Product Availability Matrixes) indican las versiones de sistemas operativos, bases de datos y otros tipos de orígenes y destinos de datos admitidos por la versión de un producto. Puede recorrer las PAM de Informatica en <https://network.informatica.com/community/informatica-network/product-availability-matrices>.

Informatica Velocity

Informatica Velocity es una colección de consejos y procedimientos recomendados desarrollados por los servicios profesionales de Informatica que se basan en experiencias reales de cientos de proyectos de administración de datos. Informatica Velocity representa el conocimiento colectivo de los consultores de Informatica que trabajan con organizaciones de todo el mundo para planificar, desarrollar, implementar y dar mantenimiento a soluciones de administración de datos exitosas.

Puede encontrar recursos de Informatica Velocity en <http://velocity.informatica.com>. Si tiene alguna pregunta, comentario o idea acerca de Informatica Velocity, póngase en contacto con los servicios profesionales de Informatica en ips@informatica.com.

Catálogo de soluciones de Informatica

El catálogo de soluciones de Informatica es un foro donde puede buscar soluciones que aumenten, amplíen o mejoren sus implementaciones de Informatica. Aproveche cualquiera de los cientos de soluciones de socios y desarrolladores de Informatica que se encuentran en el catálogo para mejorar su productividad y acelerar la implementación de los proyectos. Puede encontrar el catálogo de soluciones de Informatica en <https://marketplace.informatica.com>.

Servicio internacional de atención al cliente de Informatica

Puede ponerse en contacto con un centro de atención global por teléfono o a través del Informatica Network.

Para encontrar el número de teléfono local del servicio internacional de atención al cliente de Informatica, visite el sitio web de Informatica en el siguiente vínculo:

<https://www.informatica.com/services-and-training/customer-success-services/contact-us.html>.

Para encontrar recursos de soporte en línea en Informatica Network, visite <https://network.informatica.com> y seleccione la opción eSupport.

CAPÍTULO 1

Lenguaje de transformación

Este capítulo incluye los siguientes temas:

- [Resumen del idioma de transformación, 11](#)
- [Sintaxis de expresiones, 12](#)
- [Agregar comentarios a las expresiones, 15](#)
- [Palabras reservadas, 15](#)

Resumen del idioma de transformación

Informatica Developer proporciona un idioma de transformación que incluye funciones de tipo SQL para transformar los datos de origen. Utilice estas funciones para escribir expresiones.

Las expresiones modifican datos o comprueban si los datos coinciden con las condiciones. Por ejemplo, podría utilizar la función AVG para calcular el salario medio de todos los empleados, o la función SUM para calcular las ventas totales de una rama específica.

Puede crear una expresión simple que sólo contenga un puerto, como ORDERS, o un literal numérico como 10. También puede escribir expresiones complejas que incluyan funciones anidadas dentro de funciones, o combinar los diferentes puertos que utilizan los operadores del idioma de transformación.

Componentes del idioma de transformación

El idioma de transformación incluye los siguientes componentes para crear expresiones de transformación simples o complejas:

- **Funciones.** Más de 100 funciones de tipo SQL le permiten cambiar los datos en una asignación.
- **Operadores.** Use los operadores de transformación para crear expresiones de transformación para realizar cálculos matemáticos, combinar o comparar datos.
- **Constantes.** Utilice las constantes integradas para hacer referencia a valores que permanecen constantes como TRUE.
- **Parámetros de asignación.** Crea parámetros de asignación para su uso dentro de una asignación o un mapplet para hacer referencia a valores que permanecen constantes durante una ejecución de asignación o de mapplet como, por ejemplo, un tipo de impuesto sobre las ventas.
- **Variables incorporadas y locales.** Utilice variables integradas para escribir expresiones que hagan referencia a valores que cambian como, por ejemplo, la fecha del sistema. También puede crear variables locales en transformaciones.

- **Valores de retorno.** También puede escribir expresiones que incluyan las transformaciones de búsqueda de valores de retorno.

Internacionalización y el idioma de transformación

Las funciones del idioma de transformación pueden manejar datos de caracteres en modo de movimiento de datos ASCII o Unicode. Utilice el modo Unicode para manejar los datos de caracteres *multibyte*. Los valores de retorno de las siguientes funciones y transformaciones dependen de la página de códigos del Servicio de integración de datos y del modo de movimiento de datos:

- INITCAP
- LOWER
- UPPER
- MIN (Fecha)
- MIN (Número)
- MIN (Cadena)
- MAX (Fecha)
- MAX (Número)
- MAX (Cadena)
- Cualquier función que use sentencias condicionales para comparar cadenas, tales como IIF y DECODE

MIN y MAX también devuelven valores basados en el criterio de ordenación asociado a la página de códigos del Servicio de integración de datos.

Cuando se valida una expresión no válida en el Editor de expresiones, un cuadro de diálogo muestra la expresión con un indicador de error, ">>>>". Este indicador aparece a la izquierda y señala la parte de la expresión que contiene el error. Por ejemplo, si la expresión `a = b + c` contiene un error en la `c`, se muestra el mensaje de error:

```
a = b + >>>> c
```

Las funciones del idioma de transformación que evalúan los datos de caracteres están orientadas a caracteres, no orientadas a bytes. Por ejemplo, la función LENGTH devuelve el número de caracteres de una cadena, no el número de bytes. La función LOWER devuelve una cadena en minúsculas en función de la página de códigos del Servicio de integración de datos.

Sintaxis de expresiones

Aunque el lenguaje de transformación se basa en el lenguaje SQL estándar, hay una serie de diferencias entre estos dos lenguajes. Por ejemplo, SQL admite las palabras clave ALL y DISTINCT para las funciones de agregado, pero el lenguaje de transformación no admite estas palabras clave. Además, el lenguaje de transformación admite una condición de filtro opcional para las funciones de agregado, lo cual no ocurre en SQL.

Puede crear una expresión tan simple como un puerto (por ejemplo, ORDERS) o un literal numérico (por ejemplo, 10). Asimismo, puede escribir expresiones complejas que incluyan funciones anidadas en otras funciones o combinar varias columnas mediante operadores del lenguaje de transformación.

Componentes de expresiones

Las expresiones pueden estar compuestas por cualquier combinación de los siguientes componentes:

- Puertos (entrada, entrada/salida, variable)
- Literales de cadena, literales numéricos
- Constantes
- Funciones
- Variables integradas y locales
- Parámetros de asignación
- Operadores
- Valores devueltos

Puertos y valores de retorno

Si escribe una expresión que incluye un puerto o un valor de retorno de una transformación no conectada, use los calificadores de referencia de la siguiente tabla:

Calificador de referencia	Descripción
:LKP	<p>Obligatorio cuando se crea una expresión que incluye el valor de retorno de una transformación de búsqueda no conectada. La sintaxis general es:</p> <pre>:LKP.lookup_transformation(argument1, argument2, ...)</pre> <p>Los argumentos son los puertos locales usados en la condición de búsqueda. El orden debe ser el mismo que el orden de los puertos de la transformación. Los tipos de datos para los puertos locales tienen que coincidir con el tipo de datos de los puertos de búsqueda empleados en la condición de búsqueda.</p>

Literales de cadena y literales numéricos

Puede incluir literales numéricos o de cadena.

Asegúrese de incluir los literales de cadena entre comillas simples. Por ejemplo:

```
'Alice Davis'
```

Los literales de cadena incluyen la distinción entre mayúsculas y minúsculas, y pueden contener cualquier carácter excepto comillas simples. Por ejemplo, no se permite la siguiente cadena:

```
'Joan's car'
```

Para devolver una cadena que contenga comillas simples, use la función CHR:

```
'Joan' || CHR(39) || 's car'
```

No use comillas simples con literales numéricos. Especifique solamente el número que desee incluir. Por ejemplo:

```
.05
```

O bien:

```
$$Sales_Tax
```

Reglas y pautas sobre la sintaxis de expresiones

Utilice las siguientes reglas y pautas para escribir expresiones:

- No se pueden incluir simultáneamente expresiones agregadas de un nivel y agregadas en una transformación de agregación.
- Si necesita crear tanto funciones de un nivel como anidadas, cree transformaciones de agregación separadas.
- No se pueden usar cadenas en expresiones numéricas.

Por ejemplo, la expresión `1 + '1'` no es válida porque solamente se puede realizar la suma en tipos de datos numéricos. No se puede añadir un entero y una cadena.

- No se pueden usar cadenas como parámetros numéricos.

Por ejemplo, la expresión `SUBSTR(TEXT_VAL, '1', 10)` no es válida porque la función `SUBSTR` requiere un valor entero, no una cadena, como posición inicial.

- No se pueden mezclar tipos de datos cuando se usan operadores de comparación.

Por ejemplo, la expresión `123,4 = '123,4'` no es válida porque compara un valor decimal con una cadena.

- Se puede pasar un valor de un puerto, una cadena o un número literales, una transformación de búsqueda o los resultados de otra expresión.
- Utilice la pestaña Puertos del Editor de expresiones para introducir un nombre de puerto en una expresión. Si cambia el nombre de un puerto en una transformación conectada, la herramienta Developer propaga el cambio de nombre en las expresiones de la transformación.
- Separe cada argumento de una función con una coma.
- Exceptuando los literales, el lenguaje de transformación no distingue entre mayúsculas y minúsculas.
- Exceptuando los literales, la herramienta Developer y el Servicio de Integración de Datos omiten los espacios.
- El signo de dos puntos (:), la coma (,) y el punto (.) tienen un significado especial y solamente deben usarse para especificar la sintaxis.
- El Servicio de integración de datos trata el guión (-) como operador de resta.
- Si se pasa un valor literal a una función, delimite las cadenas literales con comillas simples. No utilice comillas para números literales. El Servicio de integración de datos trata cualquier valor de cadena delimitado por comillas simples como una cadena de caracteres.
- Cuando se pasa un parámetro de asignación a una función dentro de una expresión, no utilice comillas para designar parámetros de asignación.
- No utilice comillas para designar puertos.
- Se pueden anidar múltiples funciones dentro de una expresión exceptuando funciones agregadas, que solamente permiten una función agregada anidada. El Servicio de integración de datos evalúa la expresión a partir de la función más interna.

Agregar comentarios a las expresiones

El idioma de transformación proporciona dos especificadores de comentarios para permitir insertar comentarios en las expresiones:

- Dos guiones, como en:

```
-- These are comments
```

- Dos barras, como en:

```
// These are comments
```

El Servicio de integración de datos ignora todo el texto de la línea a la que preceden estos dos especificadores de comentarios. Por ejemplo, si desea concatenar dos cadenas, puede escribir la siguiente expresión con los comentarios en el medio de la expresión:

```
-- This expression concatenates first and last names for customers:  
FIRST_NAME -- First names from the CUST table  
|| // Concat symbol  
LAST_NAME // Last names from the CUST table  
// Joe Smith Aug 18 1998
```

El Servicio de integración de datos ignora los comentarios y evalúa la expresión de la siguiente manera:

```
FIRST_NAME || LAST_NAME
```

Un comentario no puede continuar en una nueva línea:

```
-- This expression concatenates first and last names for customers:  
FIRST_NAME -- First names from the CUST table  
|| // Concat symbol  
LAST_NAME // Last names from the CUST table  
Joe Smith Aug 18 1998
```

En este caso, la herramienta Developer no valida la expresión, ya que la última línea no es una expresión válida.

Si no desea integrar comentarios, puede agregarlos haciendo clic en Comentario en el Editor de expresiones.

Palabras reservadas

Existen palabras en el idioma de transformación, tales como constantes, operadores y variables integradas, que están reservadas para funciones específicas. Algunas de ellas son:

- :INFA
- :LKP
- :MCR
- :TYPE
- AND
- DD_DELETE
- DD_INSERT
- DD_REJECT
- DD_UPDATE
- FALSE
- NOT

- NULL
- OR
- PROC_RESULT
- SPOUTPUT
- SYSDATE
- TRUE

Nota: No se puede utilizar una palabra reservada para denominar un puerto o una variable local. Sólo se pueden utilizar palabras reservadas dentro de las expresiones de transformación. Las palabras reservadas tienen significados predefinidos en las expresiones.

CAPÍTULO 2

Constantes

Este capítulo incluye los siguientes temas:

- [DD_DELETE, 17](#)
- [DD_INSERT, 17](#)
- [DD_REJECT, 18](#)
- [DD_UPDATE, 19](#)
- [FALSE, 19](#)
- [NULL, 20](#)
- [TRUE, 21](#)

DD_DELETE

Marca los registros para su eliminación en una expresión de estrategia de actualización. DD_DELETE equivale al literal entero 2.

Nota: Use la constante DD_DELETE solamente en la transformación de estrategia de actualización. Use DD_DELETE en lugar del literal entero 2 para facilitar la solución de problemas de expresiones numéricas complejas.

Ejemplo

La siguiente expresión marca los elementos con el número de ID 1001 para su eliminación y marca el resto de los elementos para su inserción:

```
IIF( ITEM_ID = 1001, DD_DELETE, DD_INSERT )
```

Esta expresión de estrategia de actualización usa literales numéricos para generar el mismo resultado:

```
IIF( ITEM_ID = 1001, 2, 0 )
```

Nota: La expresión que usa constantes es más fácil de leer que la expresión que usa literales numéricos.

DD_INSERT

Marca los registros para su inserción en una expresión de estrategia de actualización. DD_INSERT equivale al literal entero 0.

Nota: Use la constante DD_INSERT solamente en la transformación de estrategia de actualización. Use DD_INSERT en lugar del literal entero 0 para facilitar la solución de problemas de expresiones numéricas complejas.

Ejemplos

En los siguientes ejemplos, se modifica una asignación que calcula las ventas mensuales por comercial para poder examinar el volumen de ventas de un solo comercial.

La siguiente expresión de estrategia de actualización marca las ventas de un empleado para su inserción y rechaza el resto de los elementos:

```
IIF( EMPLOYEENAME = 'Alex', DD_INSERT, DD_REJECT )
```

Esta expresión de estrategia de actualización usa literales numéricos para generar el mismo resultado:

```
IIF( EMPLOYEENAME = 'Alex', 0, 3 )
```

Sugerencia: La expresión que usa constantes es más fácil de leer que la expresión que usa literales numéricos.

DD_REJECT

Marca los registros para su rechazo en una expresión de estrategia de actualización. DD_REJECT equivale al literal entero 3.

Nota: Use la constante DD_REJECT solamente en la transformación de estrategia de actualización. Use DD_REJECT en lugar del literal entero 3 para facilitar la solución de problemas de expresiones numéricas complejas.

Use DD_REJECT para filtrar o validar datos. Si marca un registro como rechazado, el Servicio de integración de datos omite el registro y lo escribe en el archivo de rechazos de la sesión.

Ejemplos

En los siguientes ejemplos, se modifica una asignación que calcula las ventas del mes actual y, por consiguiente, solamente se incluyen valores positivos.

Esta expresión de estrategia de actualización marca los registros con un valor inferior a 0 para su rechazo y marca el resto de los registros para su inserción:

```
IIF( SALES > 0, DD_INSERT, DD_REJECT )
```

Esta expresión usa literales numéricos para generar el mismo resultado:

```
IIF( SALES > 0, 0, 3 )
```

La expresión que usa constantes es más fácil de leer que la expresión que usa literales numéricos.

En el siguiente ejemplo controlado por datos se usan DD_REJECT y IS_SPACES para evitar los espacios de escritura en una columna de caracteres de una tabla de destino. Esta expresión marca los registros compuestos por espacios en su totalidad para su rechazo y marca el resto de los registros para su inserción:

```
IIF( IS_SPACES( CUST_NAMES ), DD_REJECT, DD_INSERT )
```

DD_UPDATE

Marca los registros para su actualización en una expresión de estrategia de actualización. DD_UPDATE equivale al literal entero 1.

Nota: Use la constante DD_UPDATE solamente en la transformación de estrategia de actualización. Use DD_UPDATE en lugar del literal entero 1 para facilitar la solución de problemas de expresiones numéricas complejas.

Ejemplos

Los siguientes ejemplos modifican una asignación que calcula las ventas del mes en curso. La asignación carga las ventas de un empleado.

Esta expresión señala los registros de Alex como actualizaciones y señala todos los demás para su rechazo:

```
IIF( EMPLOYEEENAME = 'Alex', DD_UPDATE, DD_REJECT )
```

Esta expresión utiliza literales numéricos para producir el mismo resultado, señalando las ventas de Alex para su actualización (1) y señalando el resto de registro de ventas para su rechazo (3):

```
IIF( EMPLOYEEENAME = 'Alex', 1, 3 )
```

La expresión que usa constantes es más fácil de leer que la expresión que usa literales numéricos.

La siguiente expresión de estrategia de actualización utiliza SYSDATE para encontrar sólo aquellos pedidos que se han distribuido en los últimos dos días y los señala para su inserción. Mediante DATE_DIFF, la expresión resta DATE_SHIPPED de la fecha del sistema y devuelve la diferencia entre las dos fechas. Debido a que DATE_DIFF devuelve un valor doble, la expresión utiliza TRUNC para truncar la diferencia. A continuación, compara el resultado con el literal entero 2. Si el resultado es superior a 2, la expresión marca los registros para su rechazo. Si el resultado es 2 o menor, señala los registros para su actualización. En caso contrario, los señala para su rechazo:

```
IIF( TRUNC( DATE_DIFF( SYSDATE, ORDERS_DATE_SHIPPED, 'DD' ), 0 ) > 2, DD_REJECT, DD_UPDATE )
```

FALSE

Aclara una expresión condicional. FALSE equivale al entero 0.

Ejemplo

En el siguiente ejemplo, se usa FALSE en una expresión DECODE para devolver valores basados en los resultados de una comparación. Esto es útil para realizar búsquedas múltiples basadas en un solo valor de búsqueda:

```
DECODE( FALSE,
Var1 = 22, 'Variable 1 was 22!',
Var2 = 49, 'Variable 2 was 49!',
Var1 < 23, 'Variable 1 was less than 23.',
Var2 > 30, 'Variable 2 was more than 30.',
'Variables were out of desired ranges.')
```

NULL

Indica que un valor es desconocido o no está definido. NULL no equivale a una cadena en blanco o vacía (para columnas de caracteres) ni a 0 (para columnas de valores numéricos).

Aunque puede escribir expresiones que devuelvan valores nulos, las columnas que incluyan la restricción NOT NULL o PRIMARY KEY no admitirán valores nulos. Por consiguiente, si el Servicio de integración de datos intenta escribir un valor nulo en una columna con una de estas restricciones, la base de datos rechaza la fila y el Servicio de integración de datos la escribe para rechazar el archivo. Asegúrese de tener en cuenta los valores nulos al crear las transformaciones.

Las funciones pueden tratar los valores nulos de distintas formas. Si pasa un valor nulo a una función, es posible que devuelva 0 o NULL, o es posible que omita los valores nulos.

TEMAS RELACIONADOS

- [“Funciones” en la página 50](#)

Modo de trabajo con valores nulos en expresiones booleanas

Las expresiones que combinan un valor nulo con una expresión booleana generan resultados compatibles con ANSI. Por ejemplo, el Servicio de integración de datos genera los siguientes resultados:

- NULL AND TRUE = NULL
- NULL AND FALSE = FALSE

Valores nulos en expresiones de comparación

Cuando se utiliza un valor nulo en una expresión que contiene un operador de comparación, el Servicio de integración de datos genera un valor nulo. Para comprobar si existen valores nulos en las columnas, debe utilizar ISNULL() en expresiones de comparación.

Para devolver filas que no contienen valores nulos, utilice la función ISNULL en lugar de la constante !=. Por ejemplo, utilice NOT ISNULL(Field_A).

La siguiente expresión tiene como resultado un valor nulo y la transformación de filtro no devuelve ningún fila: Field_A!=NULL.

También puede configurar la transformación de búsqueda para tratar los valores nulos como altos o bajos en las operaciones de comparación. Utilice la propiedad Orden nulo en el origen de búsqueda para configurar la manera en la que el Servicio de integración de datos gestiona los valores nulos en las expresiones de comparación de la transformación de búsqueda.

Valores nulos en funciones de agregado

El Servicio de integración de datos trata los valores nulos como nulos en las funciones de agregado. Si pasa un puerto completo o un grupo de valores nulos, la función devuelve NULL.

Valores nulos en condiciones de filtro

Si una condición de filtro tiene como resultado NULL, la función no selecciona el registro. Si la condición de filtro tiene como resultado NULL para todos los registros del puerto seleccionado, la función de agregado devuelve NULL (excepto en el caso de COUNT, que devuelve 0). Puede utilizar condiciones de filtro con funciones de agregado y las funciones CUME, MOVINGAVG y MOVINGSUM.

Valores nulos con operadores

Todas las expresiones que usan operadores (excepto el operador de cadena ||) y contienen un valor nulo siempre tienen como resultado NULL. Por ejemplo, la siguiente expresión tiene como resultado NULL:

```
8 * 10 - NULL
```

Para comprobar los valores nulos, use la función ISNULL.

TRUE

Devuelve un valor basado en el resultado de una comparación. TRUE equivale al entero 1.

Ejemplo

En el siguiente ejemplo, se usa TRUE en una expresión DECODE para devolver valores basados en los resultados de una comparación. Esto es útil para realizar búsquedas múltiples basadas en un solo valor de búsqueda:

```
DECODE( TRUE,  
Var1 = 22, 'Variable 1 was 22!',  
Var2 = 49, 'Variable 2 was 49!',  
Var1 < 23, 'Variable 1 was less than 23.',  
Var2 > 30, 'Variable 2 was more than 30.',  
'Variables were out of desired ranges.')
```

CAPÍTULO 3

Operadores

Este capítulo incluye los siguientes temas:

- [Precedencia del operador, 22](#)
- [Operadores complejos, 23](#)
- [Operadores aritméticos, 31](#)
- [Operadores de cadena, 32](#)
- [Operadores de comparación, 33](#)
- [Operadores lógicos, 34](#)

Precedencia del operador

El idioma de transformación es compatible con el uso de varios operadores y de operadores dentro de expresiones anidadas.

Si escribe una expresión que contiene varios operadores, el Servicio de integración de datos evalúa la expresión en el orden siguiente:

1. Operadores complejos
2. Operadores aritméticos
3. Operadores de cadena
4. Operadores de comparación
5. Operadores lógicos

El Servicio de integración de datos evalúa los operadores en el orden en que aparecen en la tabla siguiente. Evalúa operadores de una expresión con la misma precedencia para todos los operadores de izquierda a derecha.

La tabla siguiente enumera la precedencia para todos los operadores de idioma de transformación:

Operador	Significado
[], .	Subíndice, punto.
()	Paréntesis.
+, -, NOT	Unarios más y menos y el operador lógico NOT.

Operador	Significado
*, /, %	Multiplicación, división, módulo.
+, -	Suma, resta.
	Concatenar.
<, <=, >, >=	Menor que, menor que o igual a, mayor que, mayor que o igual a.
=, <>, !=, ^=	Igual a, distinto de, distinto de, distinto de.
AND	Operador lógico AND, utilizado cuando se especifican condiciones.
OR	Operador lógico OR, utilizado cuando se especifican condiciones.

El idioma de transformación también es compatible con el uso de operadores en expresiones anidadas. Cuando las expresiones contienen paréntesis, el Servicio de integración de datos evalúa las operaciones que están dentro de los paréntesis antes que las operaciones que están fuera de ellos. Las operaciones que están dentro de los paréntesis más interiores son las que primero se evalúan.

Por ejemplo, dependiendo de cómo se aniden las operaciones, la ecuación $8 + 5 - 2 * 8$ dará valores diferentes:

Ecuación	Valor de devolución
$8 + 5 - 2 * 8$	-3
$8 + (5 - 2) * 8$	32

Operadores complejos

Utilice operadores complejos para acceder a elementos en un tipo de datos complejos. Puede acceder a elementos de un tipo de datos de matriz, asignación o estructura.

Se pueden utilizar operadores complejos en las asignaciones que se ejecuten en el motor Spark.

En la siguiente tabla se incluyen los operadores complejos del lenguaje de transformación:

Operador	Significado
[]	Operador de subíndice. El operador de subíndice se utiliza para acceder a uno o varios elementos de una matriz. También puede utilizar un operador de subíndice para acceder al valor correspondiente a una clave indicada en un par clave-valor de una asignación.
.	Operador de punto. El operador de punto se utiliza para acceder a un elemento de una estructura. También se puede utilizar un operador de punto en una matriz de estructuras para acceder a los elementos de cada estructura.

Cuando se utiliza un operador de punto en una matriz de estructuras, devuelve los elementos del mismo nombre dentro de cada estructura como una matriz. Para acceder a los elementos de una matriz o estructura anidada, se puede utilizar una combinación de operadores complejos.

Operador de subíndice

Utilice un operador de subíndice para acceder a elementos de una matriz o una asignación. Puede acceder a un elemento específico o a un intervalo de elementos de una matriz. Puede acceder al valor correspondiente a una clave indicada en un par clave-valor de una asignación.

Sintaxis

Para tener acceso a un elemento específico de una matriz, utilice la siguiente sintaxis:

```
array[ index ]
```

Para tener acceso a un intervalo de elementos de una matriz, utilice la siguiente sintaxis:

```
array[ start_index , end_index ]
```

Para tener acceso al valor correspondiente a una clave indicada en una asignación, utilice la siguiente sintaxis:

```
map[ key ]
```

En la tabla siguiente se describen los argumentos de la sintaxis:

Argumento	Descripción
array	Matriz. La matriz desde la que se desea acceder a uno o varios elementos. Puede introducir cualquier expresión de transformación válida que dé como resultado una matriz.
index	Entero. La posición del elemento al que desea acceder. Por ejemplo, un índice de 0 indica el primer elemento de una matriz.
start_index	Entero. El inicio de índice en un intervalo de elementos a los que desea acceder. El operador de subíndice incluye el elemento que representa el inicio de índice.
end_index	Entero. El fin de índice en un intervalo de elementos a los que desea acceder. El operador de subíndice excluye el elemento que representa el fin de índice.
asignación	Asignación. La asignación desde la que desea recuperar el valor correspondiente a una clave.
clave	Tipo de datos de la clave. El elemento de la clave para el que desea recuperar el valor. Puede introducir cualquier expresión de transformación válida que dé como resultado un valor de clave de los datos de asignación.

Puede utilizar una expresión para el índice que devuelve un valor entero. Si la expresión devuelve un valor negativo, se considera que el índice es 0.

Si el índice especificado es mayor que el tamaño de la matriz menos 1, el índice accede al elemento final de la matriz.

Valor de devolución

Si especifica un índice, la expresión devuelve el elemento de la matriz. El tipo de devolución es el mismo que el tipo de datos del elemento de la matriz especificada.

Si especifica dos índices separados por una coma, como `[i, j]`, la expresión devuelve una matriz de los elementos de `i` a `j-1`. Si `i` es mayor que `j` o el tamaño de la matriz, la expresión devuelve una matriz vacía.

La configuración de tipo de la matriz secundaria que devuelve la expresión es la misma que la configuración de tipo de la matriz especificada.

Si especifica una clave, la expresión devuelve el valor asociado a la clave en la asignación. El tipo de devolución es el mismo que el tipo de datos del valor de la asignación especificada.

Nulos

Si el índice en el subíndice es mayor que el tamaño de la matriz, el operador de subíndice devuelve un valor NULL.

Si el índice es NULL, el operador de subíndice devuelve un valor NULL. Si especifica varios índices, como `[i,j]`, y `i` o `j` es NULL, la expresión devuelve NULL.

Si la matriz es NULL, el operador de subíndice devuelve un valor NULL.

Si la clave no existe en la asignación, el operador de subíndice devuelve un valor NULL.

Ejemplos

Tiene la siguiente matriz con elementos de cadena:

```
drinks = ['milk', 'coffee', 'tea', 'chai']
```

Las siguientes expresiones utilizan un operador de subíndice para tener acceso a los elementos de cadena de la matriz:

Input Value	RETURN VALUE
<code>drinks[0]</code>	<code>'milk'</code>
<code>drinks[2]</code>	<code>'tea'</code>
<code>drinks[NULL]</code>	<code>NULL</code>
<code>drinks[1,3]</code>	<code>['coffee','tea']</code>
<code>drinks[2,NULL]</code>	<code>NULL</code>
<code>drinks[3,1]</code>	<code>[]</code>

Tiene la siguiente asignación con elementos clave-valor del tipo cadena-cadena:

```
country_currency = ['England' -> 'Pound', 'France' -> 'Euro', 'Japan' -> 'Yen', 'USA' -> 'Dollar']
```

Input Value	RETURN VALUE
<code>country_currency ['Japan']</code>	<code>'Yen'</code>
<code>country_currency ['India']</code>	<code>NULL</code>
<code>country_currency ['England']</code>	<code>'Pound'</code>

Operador de punto

El operador de punto se utiliza para acceder a un elemento de una estructura. También se puede utilizar un operador de punto en una matriz de estructuras para acceder a los elementos de cada estructura de la matriz.

Sintaxis

Para acceder a un elemento de una estructura, utilice la sintaxis siguiente:

```
struct.element
```

Para acceder a un elemento de una matriz de estructuras, utilice la sintaxis siguiente:

```
array_of_structs.element
```

En la tabla siguiente se describen los argumentos de la sintaxis:

Argumento	Descripción
struct	Estructura. La estructura desde la que desea acceder a un elemento. Se puede introducir cualquier expresión de transformación válida que dé como resultado una estructura.
array_of_structs	Matriz con elementos de estructura. La matriz desde la que desea acceder a los elementos de cada estructura. Puede introducir cualquier expresión de transformación válida que dé como resultado una matriz.
element	El nombre del elemento de estructura al que desea acceder.

Valor de devolución

Si utiliza el operador de punto en una estructura, la expresión devuelve el elemento de la estructura. El tipo de devolución es el mismo que el tipo de datos del elemento de la estructura especificada.

Si utiliza el operador de punto en una matriz de estructuras, la expresión devuelve una matriz que contiene el elemento especificado de cada estructura.

Nulos

Si el elemento de la estructura tiene un valor NULL, la expresión devuelve NULL.

Si la estructura es NULL, la expresión devuelve NULL.

Ejemplos

Tiene la siguiente estructura:

```
location{
  street: NULL
  city : 'NEWYORK'
  state: 'NY'
  zip : 12345
}
```

Las siguientes expresiones utilizan un operador de punto para acceder a los elementos de la estructura:

Input Value	RETURN VALUE
location.street	NULL
location.city	'NEWYORK'
location.state	'NY'
location.zip	12345

También puede utilizar un operador de punto para acceder a los elementos de una matriz de estructuras.

Por ejemplo, tiene la siguiente matriz con tres elementos de tipo estructura y cada estructura tiene tres elementos:

```
employee_info_array = [
    derrick_struct{
        name: 'Derrick'
        city: NULL
        state: 'NY'
    },
    kevin_struct{
        name: 'Kevin'
        city: 'Redwood City'
        state: 'CA'
    },
    lauren_struct{
        name: 'Lauren'
        city: 'Woodcliff Lake'
        state: NULL
    }
]
```

Las siguientes expresiones utilizan un operador de punto para acceder a los elementos de cadena de cada estructura de la matriz:

Input Value	RETURN VALUE
employee_info_array.name	['Derrick', 'Kevin', 'Lauren']
employee_info_array.city	[NULL, 'Redwood City', 'Woodcliff Lake']
employee_info_array.state	['NY', 'CA', NULL]

Operadores complejos para tipos de datos anidados

Un tipo de datos anidados contiene elementos de tipos de datos complejos. Utilice una combinación de operadores complejos para acceder a elementos en tipos de datos anidados.

Cuando una matriz o una estructura contiene elementos de tipo matriz o estructura, utilice una combinación de operadores complejos para acceder a los elementos. Puede acceder a los elementos de matrices multidimensionales, matrices con elementos de estructura, estructuras con elementos de matriz y estructuras con elementos de estructura.

Matriz multidimensional

Una matriz multidimensional es una matriz de matrices que puede tener hasta cinco niveles de anidación. Con los operadores de subíndice puede acceder a matrices de cualquier nivel o a elementos concretos de una matriz situada en el nivel más interno.

Los operadores de subíndice se pueden usar para devolver los siguientes valores:

- Un elemento específico en una matriz situada en el nivel más interno.
- Una o varias matrices en cualquier nivel.
- Un subconjunto de una o varias matrices en cualquier nivel.

Para tener acceso a un elemento específico de una matriz situada en el nivel más interno, se utiliza más de un operador de subíndice. El número de dimensiones en una matriz multidimensional determina el número de

operadores de subíndice que se usarán. Cada operador de subíndice debe contener un valor de índice. El tipo de datos del valor de devolución es el mismo que el tipo de datos de los elementos de la matriz.

Por ejemplo, se utilizan dos operadores de subíndice en una matriz de dos dimensiones. El primer operador de subíndice determina a qué matriz unidimensional se va a acceder. El segundo operador de subíndice determina a qué elemento se va a acceder dentro de la matriz.

La siguiente matriz de dos dimensiones contiene tres matrices y cada matriz contiene elementos del tipo cadena:

```
menu_array = [  
    ['milk','coffee','tea','chai'],  
    ['ham','turkey',NULL],  
    ['caesar','cobb','greek','chipotle']  
]
```

Las siguientes expresiones utilizan dos operadores de subíndice para tener acceso a un elemento específico de cada matriz unidimensional dentro de `menu_array`:

Input Value	RETURN VALUE
<code>menu_array[0][1]</code>	'coffee'
<code>menu_array[2][3]</code>	'chipotle'
<code>menu_array[1][2]</code>	NULL

Las siguientes expresiones utilizan un único operador de subíndice para devolver matrices unidimensionales en `menu_array`:

Input Value	RETURN VALUE
<code>menu_array[0]</code>	['milk','coffee','tea','chai']
<code>menu_array[0,2]</code>	[['milk','coffee','tea','chai'], ['ham','turkey',NULL]]
<code>menu_array[1,0]</code>	[]
<code>menu_array[NULL,2]</code>	NULL

Las siguientes expresiones utilizan dos operadores de subíndice para devolver un subconjunto de matrices dentro de `menu_array`:

Input Value	RETURN VALUE
<code>menu_array[0][0,2]</code>	['milk','coffee']
<code>menu_array[2][0,3]</code>	['caesar','cobb','greek']
<code>menu_array[0,2][0,3]</code>	[['milk','coffee','tea'], ['ham','turkey',NULL]]

Matriz con elementos de estructura

Una matriz con elementos de estructura es una matriz de estructuras. Utilice una combinación de operadores de subíndice y de punto para acceder a un elemento de una estructura que se encuentra dentro de una matriz.

Para acceder a un elemento de una estructura dentro de una matriz, utilice un operador de subíndice seguido de un operador de punto. También puede invertir el orden de los operadores. Los valores de devolución no varían independientemente del orden de los operadores. Basándose en el orden de los operadores complejos, se accede al elemento de la siguiente manera:

Se utiliza un operador de subíndice seguido de un operador de punto.

El operador de subíndice accede en primer lugar al elemento indexado de la matriz y devuelve una estructura. A continuación, el operador de punto accede a un elemento dentro de la estructura.

Se utiliza un operador de punto seguido por un operador de subíndice.

El operador de punto localiza elementos que tengan el mismo nombre de cada una de las estructuras y devuelve una matriz. A continuación, el operador de subíndice accede a un elemento dentro de la matriz.

Por ejemplo, tiene la siguiente matriz: `employee_info_array`:

```
employee_info_array = [  
  derrick_struct{  
    name: 'Derrick'  
    city: NULL  
    state: 'NY'  
  },  
  kevin_struct{  
    name: 'Kevin'  
    city: 'Redwood City'  
    state: 'CA'  
  },  
  lauren_struct{  
    name: 'Lauren'  
    city: 'Woodcliff Lake'  
    state: NULL  
  }  
]
```

Las siguientes expresiones utilizan un operador de subíndice seguido de un operador de punto en la matriz `employee_info_array`:

Input Value	RETURN VALUE
<code>employee_info_array[0].name</code>	'Derrick'
<code>employee_info_array[1].city</code>	'Redwood City'
<code>employee_info_array[2].state</code>	NULL

Cuando se utiliza un operador de punto primero, el operador de punto devuelve una matriz con elementos del mismo nombre de cada estructura. Por ejemplo, las siguientes expresiones muestran el valor de devolución cuando se utiliza un operador de punto:

Input Value	RETURN VALUE
<code>employee_info_array.name</code>	['Derrick', 'Kevin', 'Lauren']

Input Value	RETURN VALUE
<code>employee_info_array.city</code>	<code>[NULL, 'Redwood City', 'Woodcliff Lake']</code>
<code>employee_info_array.state</code>	<code>['NY', 'CA', NULL]</code>

A continuación, el operador de subíndice accede a un elemento de la matriz devuelta. Las siguientes expresiones utilizan un operador de punto seguido de un operador de subíndice:

Input Value	RETURN VALUE
<code>employee_info_array.name[0]</code>	<code>'Derrick'</code>
<code>employee_info_array.city[1]</code>	<code>'Redwood City'</code>
<code>employee_info_array.state[2]</code>	<code>NULL</code>

Tenga en cuenta que los valores de devolución no cambian si se utiliza un operador de subíndice o un operador de punto en primer lugar. Por ejemplo, las expresiones `employee_info_array[0].name` y `employee_info_array.name[0]` tienen el mismo valor de devolución `'Derrick'`.

Estructura con elementos de matriz

Para acceder a los elementos de una matriz que se encuentra dentro de una estructura, utilice un operador de punto seguido de un operador de subíndice. El operador de punto accede en primer lugar al elemento de matriz especificado de una estructura. A continuación, el operador de subíndice accede a los elementos de la matriz en función del valor de índice.

Por ejemplo, tiene la siguiente estructura con los elementos de matriz bebidas, sándwiches y ensaladas (`drinks`, `sandwiches`, y `salads`).

```
menu_struct{
  drinks: ['milk', 'coffee', 'tea', 'chai']
  sandwiches: ['ham', 'turkey', NULL]
  salads: ['caesar', 'cobb', 'greek', 'chipotle']
}
```

Cuando utiliza la expresión `menu_struct.drinks[0]`, el operador de punto accede en primer lugar al elemento de matriz `drinks`. A continuación, el operador de subíndice accede al elemento en la posición 0 de la matriz `drinks: ['milk', 'coffee', 'tea', 'chai']`. El valor de devolución es `milk`.

Las siguientes expresiones utilizan un operador de punto seguido de un operador de subíndice para acceder a los elementos de las matrices de la estructura `menu_struct`:

Input Value	RETURN VALUE
<code>menu_struct.drinks[1]</code>	<code>'coffee'</code>
<code>menu_struct.sandwiches[2]</code>	<code>NULL</code>
<code>menu_struct.salads[3]</code>	<code>'chipotle'</code>
<code>menu_struct.drinks[0,3]</code>	<code>['milk', 'coffee', 'tea']</code>

Estructura con elementos de estructura

Una estructura que contiene uno o varios niveles de estructuras es una estructura anidada. Con los operadores de punto puede acceder a estructuras de cualquier nivel o a elementos concretos de una estructura situada en el nivel más interno.

Los operadores de punto se pueden usar para devolver los siguientes valores:

- Un elemento especificado de una estructura situada en el nivel más interno.
- Una o varias estructuras en cualquier nivel.

Para tener acceso a un elemento específico de una estructura situada en el nivel más interno, se utiliza más de un operador de punto. El número de niveles en una estructura anidada determina el número de operadores de punto que se usarán. El tipo de datos del valor de devolución es el mismo que el tipo de datos del elemento de la estructura. Por ejemplo, en una estructura anidada de dos niveles, se utilizan dos operadores de punto. El primer operador de punto accede al elemento de estructura secundario especificado en una estructura principal. Después, el segundo operador de punto accede a los elementos de la estructura secundaria.

El siguiente ejemplo utiliza una estructura `employee_info_struct` que contiene dos estructuras secundarias, `home_address_info` y `department_info`:

```
employee_info_struct{
    emp_name: 'Derrick'
    home_address_info{
        city: 'New York'
        state: NULL
    }
    department_info{
        NULL
    }
}
```

Las siguientes expresiones utilizan operadores de punto para acceder a los elementos de la estructura `employee_info_struct`:

Input Value	RETURN VALUE
<code>employee_info_struct.emp_name</code>	'Derrick'
<code>employee_info_struct.home_address_info</code>	{ city: 'New York' state: NULL }
<code>employee_info_struct.department_info</code>	NULL
<code>employee_info_struct.home_address_info.city</code>	'New York'
<code>employee_info_struct.home_address_info.state</code>	NULL

Operadores aritméticos

Utilice operadores aritméticos para realizar cálculos matemáticos en datos numéricos.

La siguiente tabla enumera los operadores aritméticos en orden de precedencia en el idioma de transformación:

Operador	Significado
+, -	Unarios más y menos. El unario más indica un valor positivo. El unario menos indica un valor negativo.
*, /, %	Multipliación, división, módulo. Un módulo es el resto después de dividir dos números enteros. Por ejemplo, $13 \% 2 = 1$, ya que 13 dividido por 2 es igual a 6 con un resto igual a 1.
+, -	Suma, resta. El operador de suma (+) no concatena cadenas. Para concatenar cadenas, utilice el operador de cadena . Para realizar operaciones aritméticas con valores de fecha, utilice las funciones de fecha.

Si se realizan operaciones aritméticas con un valor nulo, la función devuelve NULL.

Cuando se utilizan operadores aritméticos en una expresión, todos los operandos de la expresión deben ser numéricos. Por ejemplo, la expresión `1 + '1'` no es válida, ya que agrega un número entero a una cadena. La expresión `1,23 + 4 / 2` es válida porque todos los operandos son numéricos.

Nota: El idioma de transformación ofrece funciones integradas de fecha que permiten realizar operaciones aritméticas en valores de fecha y hora.

TEMAS RELACIONADOS

- [“Descripción de operaciones aritméticas con fechas” en la página 49](#)

Operadores de cadena

Use el operador de cadena || para concatenar dos cadenas. El operador || convierte los operandos de cualquier tipo de datos (excepto el tipo de datos Binary) en tipos de datos String antes de la concatenación:

Valor de entrada	Valor devuelto
'alpha' 'betical'	alphabetical
'alpha' 2	alpha2
'alpha' NULL	alpha

El operador || incluye espacios en blanco iniciales y finales. Use las funciones LTRIM y RTRIM para recortar los espacios en blanco iniciales y finales antes de concatenar dos cadenas.

Valores nulos

El operador || omite los valores nulos. No obstante, si ambos valores son NULL, el operador || devuelve NULL.

Ejemplo

En el siguiente ejemplo, se muestra una expresión que concatena los nombres y los apellidos de los empleados de dos columnas. Esta expresión elimina los espacios del final del nombre y el inicio del apellido, concatena un espacio con el final de cada nombre y, a continuación, concatena el apellido:

```
LTRIM( RTRIM( EMP_FIRST ) || ' ' || LTRIM( EMP_LAST ) )
```

EMP_FIRST	EMP_LAST	RETURN VALUE
' Alfred'	' Rice '	Alfred Rice
' Bernice'	' Kersins'	Bernice Kersins
NULL	' Proud'	Proud
' Curt'	NULL	Curt
NULL	NULL	NULL

Nota: Además, puede usar la función CONCAT para concatenar dos valores de cadena. No obstante, el operador || genera los mismos resultados en menos tiempo.

Operadores de comparación

Utilice los operadores de comparación para comparar cadenas de caracteres o numéricas, manipular los datos y devolver un valor TRUE (1) o FALSE (0).

La siguiente tabla enumera los operadores de comparación en el idioma de transformación:

Operador	Significado
=	Igual a.
>	Mayor que.
<	Menor que.
>=	Mayor o igual que.
<=	Menor o igual que.
<>	Distinto de.
!=	Distinto de.
^=	Distinto de.

Utilice los operadores mayor que (>) y menor que (<) para comparar valores numéricos o devolver un intervalo de filas en función del orden de clasificación de una clave principal en un puerto determinado.

Cuando se utilizan operadores de comparación en una expresión, los operandos deben ser del mismo tipo de datos. Por ejemplo, la expresión 123,4 > '123' no es válida porque se compara un número decimal con una

cadena. Las expresiones `123,4 > 123` y `'a' != 'b'` son válidas porque los operandos son del mismo tipo de datos.

Si se compara un valor con un valor nulo, el resultado es `NULL`.

Si una condición de filtro se evalúa con `NULL`, el servicio de integración devuelve `NULL`.

Comparación de tipos de datos complejos

Puede utilizar los operadores igual a (`=`) y distinto de (`!=`) para comparar tipos de datos complejos, como matrices o estructuras.

Para que dos matrices sean equivalentes, deben aplicarse las siguientes condiciones:

- Los elementos de matriz deben ser del mismo tipo de datos.
- Las matrices deben tener el mismo tamaño.
- La entrada en cada índice debe ser la misma.

Supongamos que tiene las siguientes matrices:

```
A = [1, 2, 3]
B = [1, 2, 3]
```

Puede hacer la siguiente comparación:

```
A = B
```

RETURN VALUE: `TRUE (1)`

Ambas matrices tienen el mismo tamaño y la entrada en cada índice es la misma, tanto que `A[0]=B[0]`, `A[1]=B[1]`, and `A[2]=B[2]`.

Cuando se comparan dos estructuras, las estructuras son equivalentes si cumplen las siguientes condiciones:

- Los elementos de estructura correspondientes deben ser del mismo tipo de datos.
- Las estructuras deben tener los mismos datos.

Si se cumplen estas condiciones, las dos estructuras son equivalentes incluso si los elementos de la estructura tienen nombres diferentes.

Supongamos que tiene las siguientes estructuras:

```
struct1 {
  name:'Paul'
  zip:10004
}

struct2 {
  firstname:'Paul'
  zip1:10004
}
```

Puede hacer la siguiente comparación:

```
struct1 = struct2
```

RETURN VALUE: `TRUE (1)`

Operadores lógicos

Use operadores lógicos para manipular datos numéricos. Las expresiones que devuelven un valor numérico tienen como resultado `TRUE` para los valores distintos de 0, `FALSE` para 0 y `NULL` para `NULL`.

En la siguiente tabla, se incluyen los operadores lógicos del lenguaje de transformación:

Operador	Significado
NOT	Niega el resultado de una expresión. Por ejemplo, si una expresión tiene como resultado TRUE, el operador NOT devuelve FALSE. Si una expresión tiene como resultado FALSE, NOT devuelve TRUE.
AND	Une dos condiciones y devuelve TRUE si ambas condiciones tienen como resultado TRUE. Se devuelve FALSE si una de las condiciones no es TRUE.
OR	Conecta dos condiciones y devuelve TRUE si ambas condiciones tienen como resultado TRUE. Se devuelve FALSE si ninguna de las condiciones es TRUE.

Valores nulos

Las expresiones que combinan un valor nulo con una expresión booleana generan resultados compatibles con ANSI. Por ejemplo, el Servicio de integración de datos genera los siguientes resultados:

- NULL AND TRUE = NULL
- NULL AND FALSE = FALSE

CAPÍTULO 4

Variables

Este capítulo incluye los siguientes temas:

- [Variables integradas, 36](#)
- [Variables locales, 36](#)

Variables integradas

El lenguaje de transformación ofrece la variable integrada SYSDATE, que devuelve la fecha del sistema. Puede utilizar SYSDATE en una expresión. Por ejemplo, puede utilizar SYSDATE en una función DATE_DIFF.

SYSDATE

SYSDATE devuelve la fecha y hora actuales con una precisión de segundos en el nodo que procesa los datos para cada fila pasada mediante la transformación. SYSDATE se almacena como un valor de tipo de datos de fecha y hora de transformación.

Ejemplo

La siguiente expresión usa SYSDATE para buscar pedidos enviados en los dos últimos días y marcarlos para su inserción. Mediante DATE_DIFF, el Servicio de integración de datos resta DATE_SHIPPED de la fecha del sistema y devuelve la diferencia entre las dos fechas. Dado que DATE_DIFF devuelve un valor doble, la expresión trunca la diferencia. A continuación, compara el resultado con el literal entero 2. Si el resultado es superior a 2, la expresión marca las filas para su rechazo. Si el resultado es 2 o inferior, la expresión marca las filas para su inserción.

```
IIF( TRUNC( DATE_DIFF( SYSDATE, DATE_SHIPPED, 'DD' ),  
0 ) > 2, DD_REJECT, DD_INSERT
```

Variables locales

Si usa variables locales en una asignación, debe hacerlo en cualquier expresión de transformación de la asignación. Por ejemplo, si usa un cálculo de impuestos complejo en una asignación, es posible que desee escribir la expresión una vez y designarla como una variable. Esto aumenta el rendimiento, ya que el Servicio de integración de datos sólo realiza el cálculo una vez.

Las variables locales son útiles si se usan con expresiones de procedimientos almacenados para capturar varios valores devueltos.

CAPÍTULO 5

Fechas

Este capítulo incluye los siguientes temas:

- [Introducción a las fechas, 37](#)
- [Cadenas de formato de fecha, 41](#)
- [Cadenas de formato TO_CHAR, 42](#)
- [Cadenas de formato TO_DATE e IS_DATE, 45](#)
- [Descripción de operaciones aritméticas con fechas, 49](#)

Introducción a las fechas

El lenguaje de transformación proporciona un conjunto de funciones de fecha y variables de fecha integradas para realizar transformaciones en las fechas. Las funciones de fecha permiten redondear, truncar o comparar fechas, extraer una parte de una fecha o realizar una operación aritmética con una fecha. Puede pasar cualquier valor con un tipo de datos de fecha a una función de fecha.

Use variables de fecha para capturar la fecha actual en el nodo que hospeda el Servicio de integración de datos.

El lenguaje de transformación proporciona además los siguientes conjuntos de cadenas de formato:

- **Cadenas de formato de fecha.** Úselas con las funciones de fecha para especificar las partes de una fecha.
- **Cadenas de formato TO_CHAR.** Úselas para especificar el formato de la cadena devuelta.
- **Cadenas de formato TO_DATE e IS_DATE.** Úselas para especificar el formato de la cadena que desee convertir en una fecha o probar.

Tipo de datos de fecha y hora

Informatica usa tipos de datos genéricos para transformar datos de diversos orígenes. Estos tipos de datos de transformación incluyen un tipo de datos de fecha y hora que admite valores de fecha y hora de hasta nanosegundos. Informatica almacena las fechas internamente en formato binario.

Las funciones de fecha admiten solamente valores de fecha y hora. Para pasar una cadena a una función de fecha, primero debe usar TO_DATE para convertirla en un valor de fecha y hora. Por ejemplo, la siguiente expresión convierte un puerto de cadenas en valores de fecha y hora y, a continuación, añade un mes a cada fecha:

```
ADD_TO_DATE( TO_DATE( STRING_PORT, 'MM/DD/RR'), 'MM', 1 )
```

Puede usar fechas comprendidas entre el 1 d. C. y el 9999 d. C. según el calendario gregoriano.

Día juliano, día juliano modificado y calendario gregoriano

Sólo puede utilizar fechas del sistema del calendario gregoriano. Las fechas del calendario juliano se llaman *fechas julianas* y no se admiten en Informatica. Este término no debe confundirse con *día juliano* o con el calendario juliano modificado.

Puede manipular formatos de calendario juliano modificado (MJD) utilizando el formato de cadenas J. El MJD para una fecha dada es el número de días hasta esa fecha desde el 1 de enero de 4713 a. C. a las 00:00:00 (medianoche). Por definición, MJD incluye un componente de hora expresado como un decimal, que representa una fracción de 24 horas. La cadena de formato J no convierte este componente de hora.

Por ejemplo, la expresión TO_DATE siguiente convierte las cadenas del puerto SHIP_DATE_MJD_STRING en valores de fecha en el formato de fecha predeterminado:

```
TO_DATE (SHIP_DATE_MJD_STR, 'J')
```

SHIP_DATE_MJD_STR	RETURN_VALUE
2451544	Dec 31 1999 00:00:00.000000000
2415021	Jan 1 1900 00:00:00.000000000

SHIP_DATE_MJD_STR	RETURN_VALUE
2451544	Dec 31 1999 00:00:00.000000000
2415021	Jan 1 1900 00:00:00.000000000

Debido a que la cadena de formato J no incluye la parte de hora de una fecha, los valores de retorno tienen la hora establecida en 00:00:00,000000000.

También puede utilizar la cadena de formato J en expresiones TO_CHAR. Por ejemplo, utilice la cadena de formato J en una expresión TO_CHAR para convertir valores de fecha en valores MJD expresados en forma de cadenas. Por ejemplo:

```
TO_CHAR(SHIP_DATE, 'J')
```

SHIP_DATE	RETURN_VALUE
Dec 31 1999 23:59:59	2451544
Jan 1 1900 01:02:03	2415021

Nota: El Servicio de integración de datos ignora la parte de hora de la fecha en una expresión TO_CHAR.

Fechas del año 2000

Todas las funciones de fecha del lenguaje de transformación admiten el año 2000. Informatica Developer admite fechas entre el 1 d. C. y el 9999 d. C.

Cadena de formato RR

El idioma de transformación proporciona la cadena de formato RR para convertir cadenas con años de dos dígitos a fechas Usando TO_DATE y la cadena de formato RR, se puede convertir una cadena en el formato

MM/DD/RR a una fecha. La cadena de formato RR convierte los datos de manera diferente en función del año actual.

- **Año actual entre 0 y 49.** Si el año actual está entre 0 y 49 (como 2003) y el año de la cadena de origen está entre 0 y 49, el Servicio de integración de datos devuelve el siglo actual más el año en dos dígitos de la cadena de origen. Si el año de la cadena de origen está entre 50 y 99, el servicio de integración devuelve el siglo anterior más el año en dos dígitos de la cadena de origen.
- **Año actual entre 50 y 99.** Si el año actual está entre 50 y 99 (como 1998) y el año de la cadena de origen está entre 0 y 49, el Servicio de integración de datos devuelve el siglo siguiente más el año en dos dígitos de la cadena de origen. Si el año de la cadena de origen está entre 50 y 99, el Servicio de integración de datos devuelve el siglo actual más el año en dos dígitos especificado.

En la tabla siguiente, se resume cómo la cadena de formato RR convierte a fechas:

Año actual	Año de origen	La cadena de formato RR devuelve
0-49	0-49	Siglo actual
0-49	50-99	Siglo anterior
50-99	0-49	Siglo siguiente
50-99	50-99	Siglo actual

Ejemplo

La siguiente expresión devuelve los mismos valores para cualquier año entre 1950 y 2049:

```
TO_DATE( ORDER_DATE, 'MM/DD/RR' )
```

ORDER_DATE	RETURN_VALUE
'04/12/98'	04/12/1998 00:00:00.000000000
'11/09/01'	11/09/2001 00:00:00.000000000

Diferencia entre las cadenas de formato YY y RR

Informatica Developer proporciona también una cadena de formato YY. Las cadenas de formato RR e YY especifican años de dos dígitos. Las cadenas de formato YY y RR generan resultados idénticos con todas las funciones de fecha excepto TO_DATE. En el caso de las expresiones TO_DATE, RR e YY generan resultados diferentes.

En la siguiente tabla, se muestran los distintos resultados devueltos por cada cadena de formato:

Cadena	Año actual	TO_DATE (cadena, 'MM/DD/RR')	TO_DATE (cadena, 'MM/DD/YY')
04/12/98	1998	04/12/1998 00:00:00.000000000	04/12/1998 00:00:00.000000000
11/09/01	1998	11/09/2001 00:00:00.000000000	11/09/1901 00:00:00.000000000
04/12/98	2003	04/12/1998 00:00:00.000000000	04/12/2098 00:00:00.000000000
11/09/01	2003	11/09/2001 00:00:00.000000000	11/09/2001 00:00:00.000000000

En el caso de las fechas a partir del año 2000, la cadena de formato YY genera menos resultados relevantes que la cadena de formato RR. Use la cadena de formato RR para las fechas del siglo veintiuno.

Fechas de bases de datos relacionales

En general, las fechas almacenadas en bases de datos relacionales contienen un valor de fecha y hora. La fecha incluye el mes, el día y el año, y la hora puede incluir horas, minutos, segundos y subsegundos. Puede pasar datos de fecha y hora a cualquier función de fecha.

Fechas de archivos sin formato

Use la función TO_DATE para convertir cadenas en valores de fecha y hora. Además, puede usar IS_DATE para comprobar si una cadena es una fecha válida antes de convertirla con TO_DATE. Las funciones de fecha del lenguaje de transformación admiten solamente valores de fecha. Para pasar una cadena a una función de fecha, primero debe usar la función TO_DATE para convertirla en un tipo de datos de fecha y hora de transformación.

Formato de fecha predeterminado

El Servicio de integración de datos utiliza un formato predeterminado de fecha para almacenar y manipular las cadenas que representan fechas. Para especificar el formato predeterminado de fecha, indique un formato de fecha en el atributo Cadena de formato de fecha y hora en la configuración del visor de datos. El formato de fecha predeterminado es MM/DD/YYYY HH24:MI:SS.US.

Dado que Informatica almacena los datos en formato binario, el Servicio de integración de datos utiliza el formato de fecha predeterminado cuando efectúa las siguientes acciones:

- **Convierte una fecha en una cadena al conectar un puerto de fecha/hora a un puerto de cadena.** El Servicio de integración de datos convierte la fecha en una cadena en el formato de fecha definido en la configuración del visor de datos.
- **Convierte una cadena en una fecha al conectar un puerto de cadena con un puerto de fecha/hora.** El Servicio de integración de datos presupone que los valores de la cadena están en el formato de fecha definido por la configuración del visor de datos. Si un valor de entrada no coincide con este formato, o si es una fecha no válida, el Servicio de integración de datos omite la fila. Si la cadena está en este formato, el Servicio de integración de datos convierte la cadena en un valor de fecha.
- **Utilice TO_CHAR(date, [format_string]) para convertir fechas en cadenas.** Si omite la cadena de formato, el Servicio de integración de datos devuelve la cadena en el formato de fecha definido en la configuración del visor de datos. Si especifica una cadena de formato, el Servicio de integración de datos devuelve una cadena en el formato especificado.
- **Utilice TO_DATE(date, [format_string]) para convertir cadenas en fechas.** Si omite la cadena de formato, el Servicio de integración de datos presupone que la cadena está en el formato de fecha definido en la configuración del visor de datos. Si especifica una cadena de formato, el Servicio de integración de datos presupone que la cadena está en el formato especificado.

El formato predeterminado de fecha MM/DD/YYYY HH24:MI:SS.US se compone de:

- Mes (enero = 01, setiembre = 09)
- Día (del mes)
- Año (expresado en cuatro dígitos, como 1998)
- Hora (en formato de 24 horas, por ejemplo, 12:00:00 AM = 0, 1:00:00 AM = 1, 12:00:00 PM = 12, 11:00:00 PM = 23)
- Minutos

- Segundos
- Microsegundos

Cadenas de formato de fecha

Se pueden evaluar fechas de entradas mediante una combinación de cadenas de formato y funciones de fecha. Las cadenas de formato de fecha no están internacionalizadas y deben introducirse en formatos predefinidos, tal como aparecen en la siguiente tabla.

La siguiente tabla resume las cadenas de formato para especificar una parte de una fecha:

Cadena de formato	Descripción
D, DD, DDD, DAY, DY, J	Días (01-31). Utilice cualquiera de estas cadenas de formato para especificar la porción de un día completo para una fecha. Por ejemplo, si se pasa 12-APR-1997 a una función de fecha, utilice cualquier de estas cadenas de formato para especificar 12.
HH, HH12, HH24	Hora del día (0-23), donde 0 son las 12 AM (medianoche). Utilice cualquiera de estas cadenas de formato para especificar la porción de una hora completa de una fecha. Por ejemplo, si se pasa la fecha 12-APR-1997 2:01:32 PM, utilice HH, HH12 o HH24 para especificar la porción de hora de la fecha.
MI	Minutos (0-59).
MM, MON, MONTH	Mes (01-12). Utilice cualquiera de estas cadenas de formato para especificar la porción de un mes completo para una fecha. Por ejemplo, si se pasa 12-APR-1997 a una función de fecha, utilice MM, MON o MONTH para especificar APR.
MS	Milisegundos (0-999).
NS	Nanosegundos (0-999999999).
SS, SSSS	Segundos (0-59).
US	Microsegundos (0-999999).
Y, YY, YYYY, YYYY, RR	Porción del año de una fecha (de 0001 a 9999). Utilice cualquiera de estas cadenas de formato para especificar la porción de un año completo para una fecha. Por ejemplo, si se pasa 12-APR-1997 a una función de fecha, utilice Y, YY, YYYY o YYYY para especificar 1997.

Nota: La cadena de formato no distingue entre mayúsculas y minúsculas. En todos los casos debe delimitarse mediante comillas simples.

La siguiente tabla describe funciones de fecha que utilizan cadenas de formato de fecha para evaluar fechas de entrada:

Función	Descripción
ADD_TO_DATE	La parte de la fecha que se desea cambiar.
DATE_DIFF	La parte de la fecha que se utiliza para calcular la diferencia entre dos fechas.

Función	Descripción
GET_DATE_PART	La parte de la fecha que se desea devolver. Esta función devuelve un valor entero basado en el formato de fecha predeterminado.
IS_DATE	La fecha que se desea comprobar.
ROUND	La parte de la fecha que se desea redondear.
SET_DATE_PART	La parte de la fecha que se desea cambiar.
SYSTIMESTAMP	La precisión de la marca de hora.
TO_CHAR (Fechas)	La cadena de carácter.
TO_DATE	La cadena de carácter.
TRUNC (Fechas)	La segunda parte de la cadena que se desea trincar.

Cadenas de formato TO_CHAR

La función TO_CHAR convierte un tipo de datos de fecha y hora en una cadena con el formato que especifique. Puede convertir en cadena toda la fecha o una parte de la fecha. Puede utilizar TO_CHAR para convertir fechas en cadenas, cambiando así el formato para crear informes.

TO_CHAR se utiliza generalmente cuando el destino es un archivo sin formato o una base de datos que no admite un tipo de datos de fecha y hora.

La siguiente tabla resume las cadenas de formato para las fechas en la función TO_CHAR:

Cadena de formato	Descripción
AM, A.M., PM, P.M.	Indicador de meridiano. Utilice una de estas cadenas de formato para especificar horas AM y PM. AM y PM devuelven los mismos valores que A.M. y P.M.
D	Día de la semana (1-7), donde el domingo es 1.
DAY	Nombre del día con una longitud de hasta nueve caracteres (por ejemplo, miércoles).
DD	Día del mes (01-31).
DDD	Día del año (001-366, incluidos años bisiestos).
DY	Nombre de un día abreviado en tres caracteres (por ejemplo, Mié).
HH, HH12	Hora del día (01-12).
HH24	Hora de día (00-23), donde 00 es 12 AM (medianoche).

Cadena de formato	Descripción
J	Fecha juliana modificada. Convierte la fecha del calendario en una cadena equivalente a su valor en la fecha juliana modificada, que se calcula a partir del 1 de enero 4713 00:00:00 a. C. Omite el componente de hora de la fecha. Por ejemplo, la expresión TO_CHAR(SHIP_DATE, 'J') convierte la fecha 31 de diciembre de 1999, 23:59:59 en la cadena 2451544.
MI	Minutos (00-59).
MM	Mes (01-12).
MONTH	Nombre del mes con una longitud máxima de hasta nueve caracteres (por ejemplo, enero).
MON	Nombre del mes abreviado en tres caracteres (por ejemplo, Ene).
MS	Milisegundos (0-999).
NS	Nanosegundos (0-999999999).
Q	Trimestre el año (1-4), donde de enero a marzo es igual a 1.
RR	Los últimos dos dígitos de un año. La función quita los dígitos del principio. Por ejemplo, si utiliza 'RR' y transfiere el año 1997, TO_CHAR devuelve 97. Cuando se utiliza con TO_CHAR, 'RR' genera los mismos resultados, ya que se puede intercambiar con 'YY.' Sin embargo, cuando se utiliza con TO_DATE, 'RR' calcula el siglo más próximo que corresponda y proporciona los dos primeros dígitos del año.
SS	Segundos (00-59).
SSSSS	Segundos hasta medianoche (00000 - 86399). Cuando utiliza SSSSS en una expresión TO_CHAR, el Servicio de integración de datos solo verifica la parte de hora de una fecha. Por ejemplo, la expresión TO_CHAR(SHIP_DATE, 'MM/DD/YYYY SSSSS') convierte 12/31/1999 01:02:03 en 12/31/1999 03723.
US	Microsegundos (0-999999).
Y	El último dígito de un año. La función quita los dígitos del principio. Por ejemplo, si utiliza 'Y' y transfiere el año 1997, TO_CHAR devuelve 7.
YY	Los últimos dos dígitos de un año. La función quita los dígitos del principio. Por ejemplo, si utiliza 'YY' y transfiere el año 1997, TO_CHAR devuelve 97.
YYY	Los últimos tres dígitos de un año. La función quita los dígitos del principio. Por ejemplo, si utiliza 'YYY' y transfiere el año 1997, TO_CHAR devuelve 997.
YYYY	Toda la parte de año de una fecha. Por ejemplo, si utiliza 'YYY' y transfiere el año 1997, TO_CHAR devuelve 1997.
W	Semana del mes (1-5) , donde la semana 1 empieza en el primer día del mes y finaliza en el séptimo, y la semana 2 se inicia en el octavo día y finaliza en el catorceavo. Por ejemplo, 1 febrero designa la primera semana de febrero.
WW	Semana del año (01-53), donde la semana 01 se inicia el 1 de enero y finaliza el 7, y la semana 2 se inicia el 8 y finaliza el 14 de enero, y así sucesivamente.

Cadena de formato	Descripción
- / . ; :	La puntuación que se muestra en la salida. Puede utilizar estos símbolos para separar las partes de una fecha. Crea, por ejemplo, la siguiente expresión para separar las partes de una fecha con un punto: TO_CHAR(DATES, 'MM.DD.YYYY').
"text"	El texto que se muestra en la salida. Si crea, por ejemplo, un puerto de salida con la expresión: TO_CHAR(DATES, 'MM/DD/YYYY "Ventas en aumento"') y transfiere la fecha 1 de abril de 1997, la función devuelve la cadena '04/01/1997 Ventas en aumento'. Puede indicar los caracteres multibyte que sean válidos en la página de códigos del repositorio.
""	Utilice las comillas dobles para separar cadenas de formato ambiguas, por ejemplo, D""DDD. Las comillas vacías no aparecen en la salida.

Nota: La cadena de formato no distingue entre mayúsculas y minúsculas. Debe encerrarse siempre entre comillas simples.

Ejemplos

En los siguientes ejemplos, se describen las cadenas de formato J, SSSSS, RR e YY. Vea cada función para obtener más ejemplos.

Nota: El Servicio de integración de datos omite la parte correspondiente a la hora de la fecha en una expresión TO_CHAR.

Cadena de formato J

Use la cadena de formato J en una expresión TO_CHAR para convertir los valores de datos en valores MJD expresados como cadenas. Por ejemplo:

```
TO_CHAR(SHIP_DATE, 'J')
```

SHIP_DATE	RETURN_VALUE
Dec 31 1999 23:59:59	2451544
Jan 1 1900 01:02:03	2415021

Cadena de formato SSSSS

Puede usar la cadena de formato SSSSS en una expresión TO_CHAR. Por ejemplo, la siguiente expresión convierte las fechas del puerto SHIP_DATE en cadenas que representan el número total de segundos desde la medianoche:

```
TO_CHAR( SHIP_DATE, 'SSSSS')
```

SHIP_DATE	RETURN_VALUE
12/31/1999 01:02:03	3723
09/15/1996 23:59:59	86399

Cadena de formato RR

La siguiente expresión convierte las fechas en cadenas con el formato MM/DD/YY:

```
TO_CHAR( SHIP_DATE, 'MM/DD/RR')
```

SHIP_DATE	RETURN_VALUE
12/31/1999 01:02:03	12/31/99
09/15/1996 23:59:59	09/15/96
05/17/2003 12:13:14	05/17/03

Cadena de formato YY

En el caso de las expresiones TO_CHAR, la cadena de formato YY genera los mismos resultados que la cadena de formato RR. La siguiente expresión convierte las fechas en cadenas con el formato MM/DD/YY:

```
TO_CHAR( SHIP_DATE, 'MM/DD/YY')
```

SHIP_DATE	RETURN_VALUE
12/31/1999 01:02:03	12/31/99
09/15/1996 23:59:59	09/15/96
05/17/2003 12:13:14	05/17/03

Cadenas de formato TO_DATE e IS_DATE

La función TO_DATE convierte una cadena con el formato especificado en un valor de fecha y hora. TO_DATE se suele usar para convertir cadenas de archivos sin formato en valores de fecha y hora. Las cadenas de formato TO_DATE no están internacionalizadas y deben introducirse en los formatos predefinidos.

Nota: En el caso de TO_DATE e IS_DATE, se usa el mismo conjunto de cadenas de formato.

Para crear una expresión TO_DATE, use una cadena de formato para cada parte de la fecha en la cadena de origen. El formato de la cadena de origen y la cadena de formato deben coincidir. No es necesario que el separador de fecha coincida para que la fecha se valide. Si alguna parte no coincide, el Servicio de integración de datos no convierte la cadena y omite la fila. Si omite la cadena de formato, la cadena de origen debe tener el formato de fecha especificado en la configuración del visor de datos.

IS_DATE indica si un valor es una fecha válida. Una fecha válida es cualquier cadena con el formato de fecha especificado en la configuración del visor de datos. Si las cadenas que desea probar no tienen el formato de fecha especificado, utilice el formato de las cadenas que aparecen en la tabla "Cadenas de formato TO_DATE e IS_DATE". Si el formato de una cadena no coincide con el formato especificado o si la cadena no representa una fecha válida, la función devuelve FALSE (0). Si el formato de la cadena coincide con el formato especificado de la cadena y es una fecha válida, la función devuelve TRUE (1). Las cadenas de formato IS_DATE no están internacionalizadas y deben introducirse con uno de los formatos enumerados en la siguiente tabla.

En la siguiente tabla se enumeran las cadenas de formato para las funciones TO_DATE e IS_DATE:

Tabla 1. Cadenas de formato TO_DATE e IS_DATE

Cadena de formato	Descripción
AM, a.m., PM, p.m.	Indicador de hora del meridiano. Utilice una de estas cadenas de formato para especificar horas a. m. y p. m. AM y PM devuelven los mismos valores que a.m. y p.m.
DAY	Nombre del día. Puede tener nueve caracteres como máximo (por ejemplo, miércoles). La cadena de formato DAY no distingue mayúsculas de minúsculas.
DD	Día del mes (1-31).
DDD	Día del año (001-366, incluidos años bisiestos).
DY	Nombre de un día abreviado en tres caracteres (por ejemplo, Mié). La cadena de formato DY no distingue mayúsculas de minúsculas.
HH, HH12	Hora del día (1-12).
HH24	Hora del día (0-23), donde 0 es 12 a. m. (medianoche).
J	Fecha juliana modificada. Convierte las cadenas con el formato MJD en valores de fecha. Omite el componente de hora de la cadena de origen y asigna a todas las fechas la hora 00:00:00.000000000. Por ejemplo, la expresión TO_DATE('2451544', 'J') convierte 2451544 en Dic 31 1999 00:00:00.000000000.
MI	Minutos (0-59).
MM	Mes (1-12).
MONTH	Nombre del mes. Puede tener nueve caracteres como máximo (por ejemplo, agosto). No distingue mayúsculas de minúsculas.
MON	Nombre de tres caracteres abreviado para el mes (por ejemplo, Ago). No distingue mayúsculas de minúsculas.
MS	Milisegundos (0-999).
NS	Nanosegundos (0-999999999).
RR	Año en cuatro dígitos (por ejemplo, 1998, 2034). Se usa si las cadenas de origen incluyen años de dos dígitos. Se usa con TO_DATE para convertir los años de dos dígitos en años de cuatro dígitos. <ul style="list-style-type: none"> - Año actual entre 50 y 99. Si el año actual está comprendido entre 50 y 99 (por ejemplo, 1998) y el valor de año de la cadena de origen está comprendido entre 0 y 49, el Servicio de integración de datos devuelve el siglo siguiente más el año de dos dígitos de la cadena de origen. Si el valor de año de la cadena de origen está comprendido entre 50 y 99, el Servicio de integración de datos devuelve el siglo actual más el año de dos dígitos especificado. - Año actual entre 0 y 49. Si el año actual está comprendido entre 0 y 49 (por ejemplo, 2003) y el año de la cadena de origen está comprendido entre 0 y 49, el Servicio de integración de datos devuelve el siglo actual más el año de dos dígitos de la cadena de origen. Si el año de la cadena de origen está comprendido entre 50 y 99, el Servicio de integración de datos devuelve el siglo anterior más el año de dos dígitos de la cadena de origen.
SS	Segundos (0-59).

Cadena de formato	Descripción
SSSSS	Segundos desde la medianoche. Si usa SSSSS en una expresión TO_DATE, el Servicio de integración de datos solamente evalúa la parte correspondiente a la hora de una fecha. Por ejemplo, la expresión TO_DATE(DATE_STR, 'MM/DD/YYYY SSSSS') convierte 12/31/1999 3783 en 12/31/1999 01:02:03.
US	Microsegundos (0-999999).
Y	Año actual en el nodo que ejecuta el Servicio de integración de datos, donde el último dígito del año se reemplaza por el valor de cadena.
YY	Año actual en el nodo que ejecuta el Servicio de integración de datos, donde los dos últimos dígitos del año se reemplazan por el valor de cadena.
YYY	Año actual en el nodo que ejecuta el Servicio de integración de datos, donde los tres últimos dígitos del año se reemplazan por el valor de cadena.
YYYY	Año en cuatro dígitos. No use esta cadena de formato si va a pasar años de dos dígitos. Use las cadenas de formato RR o YY en su lugar.

Normas y directrices para cadenas con formato de fecha

Utilice las siguientes normas y directrices cuando trabaje con cadenas con formato de fecha:

- El formato de la cadena TO_DATE debe coincidir con la cadena de formato. Si no es así, el Servicio de integración de datos podría devolver valores incorrectos o saltarse la fila. Por ejemplo, si se pasa la cadena '20200512', que representa el 12 de mayo de 2020, a TO_DATE deberá incluir la cadena de formato YYYYMMDD. Si no se incluye una cadena de formato, el Servicio de integración de datos espera la cadena con el formato de fecha especificado en la configuración del visor de datos. Del mismo modo, si se pasa una cadena que no coincide con la cadena de formato, el Servicio de integración de datos devuelve un error y se salta la fila. Por ejemplo, si se pasa la cadena de 2020120 a TO_DATE y se incluye la cadena de formato YYYYMMDD, el Servicio de integración de datos devuelve un error y se salta la fila porque la cadena no coincide con la cadena de formato.
- La cadena de formato debe estar entre comillas simples.
- El Servicio de integración de datos utiliza el formato de fecha y hora predeterminado que se especifica en la sesión. El predeterminado es MM/DD/YYYY HH24:MI:SS.US. La cadena de formato no distingue entre mayúsculas y minúsculas.

Ejemplo

En los siguientes ejemplos, se describen las cadenas de formato J, RR y SSSSS. Vea cada función para obtener más ejemplos.

Cadena de formato J

La siguiente expresión convierte las cadenas del puerto SHIP_DATE_MJD_STRING en valores de fecha con el formato de fecha predeterminado:

```
TO_DATE (SHIP_DATE_MJD_STR, 'J')
```

SHIP_DATE_MJD_STR	RETURN_VALUE
2451544	Dec 31 1999 00:00:00.000000000
2415021	Jan 1 1900 00:00:00.000000000

Dado que la cadena de formato J no incluye la parte correspondiente a la hora de la fecha, los valores devueltos incluyen la hora establecida en 00:00:00.000000000.

Cadena de formato RR

La siguiente expresión convierte una cadena en un formato de año de cuatro dígitos. El año actual es 1998:

```
TO_DATE ( DATE_STR, 'MM/DD/RR')
```

DATE_STR	RETURN VALUE
04/01/98	04/01/1998 00:00:00.000000000
08/17/05	08/17/2005 00:00:00.000000000

Cadena de formato YY

La siguiente expresión convierte una cadena en un formato de año de cuatro dígitos. El año actual es 1998:

```
TO_DATE ( DATE_STR, 'MM/DD/YY')
```

DATE_STR	RETURN VALUE
04/01/98	04/01/1998 00:00:00.000000000
08/17/05	08/17/1905 00:00:00.000000000

Nota: Para la segunda fila, RR devuelve el año 2005, pero YY devuelve el año 1905.

Cadena de formato SSSS

La siguiente expresión convierte las cadenas que incluyen los segundos desde la medianoche en valores de fecha:

```
TO_DATE ( DATE_STR, 'MM/DD/YYYY SSSS')
```

DATE_STR	RETURN_VALUE
12/31/1999 3783	12/31/1999 01:02:03.000000000
09/15/1996 86399	09/15/1996 23:59:59.000000000

Descripción de operaciones aritméticas con fechas

El lenguaje de transformación proporciona funciones de fecha integradas para poder realizar operaciones aritméticas con valores de fecha y hora del modo siguiente:

- **ADD_TO_DATE.** Permite sumar o restar una parte específica de una fecha.
- **DATE_DIFF.** Permite restar dos fechas.
- **SET_DATE_PART.** Permite cambiar una parte de una fecha.

No se pueden usar operadores aritméticos (por ejemplo, + o -) para sumar o restar fechas.

El lenguaje de transformación reconoce los años bisiestos y admite fechas entre el 1 de enero, 0001 00:00:00.000000000 d. C., y el 31 de diciembre, 9999 23:59:59.999999999 d. C.

Nota: El lenguaje de transformación usa el tipo de datos de fecha y hora de transformación para especificar valores de fecha. Las funciones de fecha solamente se pueden usar con los valores de fecha y hora.

CAPÍTULO 6

Funciones

En este capítulo se incluye información sobre la compatibilidad de funciones en el lenguaje de transformación.

Categorías de funciones

El lenguaje de transformación incluye los siguientes tipos de funciones:

- Agregar
- Carácter
- Compleja
- Conversión
- Limpieza de datos
- Fecha
- Codificación
- Financiera
- Numérica
- Científica
- Especial
- Cadena
- Prueba
- Ventana

Funciones de agregado

Las funciones de agregado devuelven valores de resumen para valores no nulos en los puertos seleccionados. Mediante la funciones de agregado se puede:

- Calcular un valor individual para todas las filas de un grupo.
- Devolver un valor individual para cada grupo en una transformación de agregación.
- Aplicar filtros para calcular valores para filas específicas en los puertos seleccionados.
- Utilizar operadores para realizar operaciones aritméticas en la función.
- Calcular dos o más valores agregados derivados de las mismas columnas origen en un pase individual.

El lenguaje de la transformación incluye las siguientes funciones de agregado:

- ANY
- AVG
- COLLECT_LIST
- COLLECT_MAP
- COUNT
- FIRST
- LAST
- MAX (Fecha)
- MAX (Número)
- MAX (Cadena)
- MEDIAN
- MIN (Fecha)
- MIN (Número)
- MIN (Cadena)
- PERCENTILE
- STDDEV
- SUM
- VARIANCE

Si configura el Servicio de integración de datos para que se ejecute en modo Unicode, MIN y MAX devuelven valores según el orden de clasificación de la página de códigos que se haya especificado en la configuración de la asignación.

Las funciones de agregado se pueden usar en las transformaciones de agregación. Solamente se puede anidar una única función de agregación dentro de otra función de agregación. El Servicio de integración de datos evalúa la expresión de función de agregado más interna y utiliza el resultado para evaluar la expresión de función de agregado exterior. Se puede configurar una transformación de agregación que agrupe por ID y que anide dos funciones de agregado del siguiente modo:

```
SUM( AVG( earnings ) )
```

cuando el conjunto de datos contiene los siguientes valores:

ID	EARNINGS
1	32
1	45
1	100
2	65
2	75
2	76
3	21

ID	EARNINGS
3	45
3	99

El valor de devolución es 186. El Servicio de integración de datos agrupa por ID, evalúa la expresión AVG y devuelve tres valores. Luego añade los valores con la función SUM para obtener el resultado.

También puede utilizar funciones de agregado como funciones de ventana en una transformación de expresión. Para utilizar una función de agregado como una función de ventana cuando se ejecuta una asignación en el motor Spark, debe configurar la transformación para la administración de ventanas. Si utiliza una función de agregado como una función de ventana, se activa la transformación de expresión.

Funciones de agregado y valores nulos

Durante la configuración del Servicio de integración de datos, puede elegir cómo tratar los valores nulos de las funciones de agregado. Puede determinar que el Servicio de integración de datos trate los valores nulos de las funciones de agregado como NULL o 0.

De forma predeterminada, el Servicio de integración de datos trata los valores nulos como NULL en las funciones de agregado. Si pasa un puerto completo o un grupo de valores nulos, la función devuelve NULL. No obstante, puede configurar el Servicio de integración de datos si pasa un puerto completo de valores nulos a una función de agregado para que devuelva 0.

Condiciones de filtro

Utilice una condición de filtro para limitar las filas devueltas en una búsqueda.

Un filtro limita las filas devueltas en una búsqueda. Puede aplicar una condición de filtro a todas las funciones agregadas y a CUME, MOVINGAVG y MOVINGSUM. La condición de filtro da como resultado valores TRUE, FALSE o NULL. Si la condición de filtro da como resultado NULL o FALSE, el Servicio de integración de datos no selecciona la fila.

Se puede introducir cualquier expresión válida de transformación. Por ejemplo, la siguiente expresión calcula el salario promedio de todos los empleados que ganan más de 50.000 \$:

```
MEDIAN( SALARY, SALARY > 50000 )
```

También puede utilizar otros valores numéricos como la condición de filtro. Por ejemplo, puede escribir lo siguiente como la sintaxis completa de la función MEDIAN, incluyendo un puerto numérico:

```
MEDIAN( PRICE, QUANTITY > 0 )
```

En todos los casos, el Servicio de integración de datos redondea un valor decimal a un entero (por ejemplo, 1,5 a 2, 1,2 a 1, 0,35 a 0) para la condición de filtro. Si el valor se redondea a 0, la condición de filtro devuelve FALSE. Si no desea redondear un valor, utilice la función TRUNC para truncar el valor a un entero:

```
MEDIAN( PRICE, TRUNC( QUANTITY ) > 0 )
```

Si se omite la condición de filtro, la función selecciona todas las filas del puerto.

Funciones de caracteres

El lenguaje de transformación incluye las siguientes funciones de caracteres:

- ASCII

- CHR
- CHRCODE
- CONCAT
- INITCAP
- INSTR
- LENGTH
- LOWER
- LPAD
- LTRIM
- METAPHONE
- REPLACECHR
- REPLACESTR
- RPAD
- RTRIM
- SOUNDEX
- SUBSTR
- UPPER

Las funciones de caracteres MAX, MIN, LOWER, UPPER e INITCAP usan la página de códigos del Servicio de integración de datos para evaluar los datos de caracteres.

Funciones complejas

Una función compleja es un tipo de función predefinida en la que el valor de la entrada o el tipo de devolución es de un tipo de datos complejos, como una matriz, una asignación o una estructura. Se pueden utilizar funciones complejas en las asignaciones que se ejecuten en el motor Spark.

El lenguaje de la transformación incluye las siguientes funciones complejas:

- ARRAY
- CAST
- COLLECT_LIST
- COLLECT_MAP
- CONCAT_ARRAY
- MAP
- MAP_FROM_ARRAYS
- MAP_KEYS
- MAP_VALUES
- PARSE_JSON
- PARSE_XML
- RESPEC
- SIZE
- STRUCT
- STRUCT_AS

Funciones de conversión

El lenguaje de transformación incluye las siguientes funciones de conversión:

- TO_BIGINT
- TO_CHAR (número)
- TO_DATE
- TO_DECIMAL
- TO_FLOAT
- TO_INTEGER

Funciones de limpieza de datos

El lenguaje de transformación incluye un grupo de funciones para eliminar errores de los datos. Puede realizar las siguientes tareas con las funciones de limpieza de datos:

- Probar los valores de entrada.
- Convertir el tipo de datos de un valor de entrada.
- Recortar valores de cadena.
- Reemplazar caracteres de una cadena.
- Codificar cadenas.
- Buscar patrones de coincidencia en expresiones regulares.

El lenguaje de transformación incluye las siguientes funciones de limpieza de datos:

- GREATEST
- IN
- INSTR
- IS_DATE
- IS_NUMBER
- IS_SPACES
- ISNULL
- LEAST
- LTRIM
- METAPHONE
- REG_EXTRACT
- REG_MATCH
- REG_REPLACE
- REPLACECHR
- REPLACESTR
- RTRIM
- SQL_LIKE
- SOUNDEX
- SUBSTR
- TO_BIGINT

- TO_CHAR
- TO_DATE
- TO_DECIMAL
- TO_FLOAT
- TO_INTEGER

Funciones de fecha

El lenguaje de transformación incluye un grupo de funciones de fecha para redondear, truncar o comparar fechas, extraer una parte de una fecha o realizar una operación aritmética con una fecha.

Puede pasar cualquier valor con un tipo de datos de fecha a cualquier función de fecha. No obstante, si desea pasar una cadena a una función de fecha, primero debe usar la función TO_DATE para convertirla en un tipo de datos de fecha y hora de transformación.

El lenguaje de transformación incluye las siguientes funciones de fecha:

- ADD_TO_DATE
- DATE_COMPARE
- DATE_DIFF
- GET_DATE_PART
- IS_DATE
- LAST_DAY
- MAKE_DATE_TIME
- MAX
- MIN
- ROUND (fecha)
- SET_DATE_PART
- SYSTIMESTAMP
- TO_CHAR (fecha)
- TIME_RANGE
- TRUNC (fecha)

Algunas funciones de fecha incluyen un argumento *format*. Debe especificar una de las cadenas de formato del lenguaje de transformación para este argumento. Las cadenas de formato de fecha no están internacionalizadas.

El tipo de datos de transformación de fecha y hora admite fechas con una precisión de nanosegundos.

TEMAS RELACIONADOS

- [“Cadenas de formato de fecha” en la página 41](#)

Funciones de codificación

El lenguaje de transformación incluye las siguientes funciones para el cifrado de datos, la compresión, la codificación y la suma de comprobación:

- AES_DECRYPT

- AES_ENCRYPT
- COMPRESS
- CRC32
- DEC_BASE64
- DECOMPRESS
- ENC_BASE64
- MD5

Funciones financieras

El lenguaje de transformación incluye las siguientes funciones financieras:

- FV
- NPER
- PMT
- PV
- RATE

Funciones numéricas

El lenguaje de transformación incluye las siguientes funciones numéricas:

- ABS
- CEIL
- CONV
- CUME
- EXP
- FLOOR
- LN
- LOG
- MAX
- MIN
- MOD
- MOVINGAVG
- MOVINGSUM
- POWER
- RAND
- ROUND
- SIGN
- SQRT
- TRUNC

Funciones científicas

El lenguaje de transformación incluye las siguientes funciones científicas:

- COS
- COSH
- SIN
- SINH
- TAN
- TANH

Funciones especiales

El lenguaje de transformación incluye las siguientes funciones especiales:

- ABORT
- DECODE
- ERROR
- IIF
- LOOKUP
- UUID4
- UUID_UNPARSE

Normalmente, las funciones especiales se usan en las transformaciones de expresión, filtro y estrategia de actualización. Puede anidar otras funciones en las funciones especiales. Además, puede anidar una función especial en una función de agregado.

Funciones de cadena

El lenguaje de transformación incluye las siguientes funciones de cadena:

- CHOOSE
- INDEXOF
- MAX
- MIN
- REVERSE

Funciones de prueba

El lenguaje de transformación incluye las siguientes funciones de prueba:

- ISNULL
- IS_DATE
- IS_NUMBER
- IS_SPACES

Funciones de ventana

El lenguaje de transformación incluye un grupo de funciones de ventana que realiza cálculos en un conjunto de filas relacionadas con la fila actual. Las funciones calculan un único valor de devolución para cada fila de entrada. Se pueden utilizar funciones complejas en las asignaciones que se ejecuten en el motor Spark.

El lenguaje de transformación incluye las siguientes funciones de ventana:

- LAG
- LEAD

Las funciones de ventana se pueden usar en las transformaciones de expresión. Si utiliza una función de ventana en una transformación de expresión, se activará la transformación.

ABORT

Detiene la asignación y emite un mensaje de error especificado al registro. Cuando el Servicio de integración de datos detecta una función ABORT, deja de transformar datos en esa fila. Procesa todas las filas leídas antes de que se cancele la ejecución de asignación. El Servicio de integración de datos graba en el destino hasta la fila cancelada y luego acumula todos los datos no confirmados hasta el último punto de confirmación.

Utilice ABORT para validar los datos. Generalmente se usa ABORT dentro de una función IIF o DECODE para establecer las reglas para cancelar una sesión.

Utilice la función ABORT tanto para los valores predeterminados de puerto de entrada y de salida. Puede usar ABORT en puertos de entrada para evitar que los valores nulos pasen a una transformación. También se puede usar ABORT para manejar cualquier tipo de error de transformación, incluidas las llamadas de la función ERROR dentro de una expresión. El valor predeterminado anula la función ERROR en una expresión. Si desea asegurarse que una sesión se detiene cuando hay un error, asigne ABORT como valor predeterminado.

Si utiliza ABORT en una expresión para un puerto no conectado, el Servicio de integración de datos no ejecuta la función ABORT.

sintaxis

```
ABORT( string )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio/Opcional	Descripción
<i>string</i>	Obligatorio	Cadena. El mensaje que desea que aparezca en el archivo de registro después de que se detenga la ejecución de asignación. La cadena puede tener una longitud cualquiera. Se puede especificar cualquier expresión de transformación válida.

Valor de retorno

NULL.

ABS

Devuelve el valor absoluto de un valor numérico.

Sintaxis

```
ABS( numeric_value )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio/ Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. Pasa los valores para los que desea devolver valores absolutos. Se puede especificar cualquier expresión de transformación válida.

Valor devuelto

Valor numérico positivo. ABS devuelve un tipo de datos coincidente con el valor numérico pasado como argumento. Si se pasa un valor doble, se devuelve un valor doble. Asimismo, si se pasa un valor entero, se devuelve un valor entero.

Se devuelve NULL si se pasa un valor nulo a la función.

Nota: Si el valor devuelto es decimal con una precisión superior a 15, puede habilitar la alta precisión para garantizar una precisión decimal de hasta 38 dígitos.

Ejemplo

La siguiente expresión devuelve la diferencia entre dos números como un valor positivo con independencia de qué número sea mayor:

```
ABS( PRICE - COST )
```

PRICE	COST	RETURN VALUE
250	150	100
52	48	4
169.95	69.95	100
59.95	NULL	NULL
70	30	40
430	330	100
100	200	100

ADD_TO_DATE

Añade una cantidad específica a una parte de un valor datetime y devuelve una fecha con el mismo formato que la fecha que se le pasa a la función. ADD_TO_DATE acepta valores enteros positivos y negativos. Utilice ADD_TO_DATE para cambiar las siguientes partes de una fecha:

- **Año.** Especifique un entero positivo o negativo en el argumento *amount* . Utilice una de las siguientes cadenas de formato de año: Y, YY, YYYY o YYYY. La siguiente expresión añade 10 años a todas las fechas del puerto SHIP_DATE:

```
ADD_TO_DATE ( SHIP_DATE, 'YY', 10 )
```

- **Mes.** Especifique un entero positivo o negativo en el argumento *amount* . Utilice una de las siguientes cadenas de formato de mes: MM, MON, MONTH. La siguiente expresión resta 10 meses a cada fecha del puerto SHIP_DATE:

```
ADD_TO_DATE( SHIP_DATE, 'MONTH', -10 )
```

- **Día.** Especifique un entero positivo o negativo en el argumento *amount* . Utilice una de las siguientes cadenas de formato de día: D, DD, DDD, DY y DAY. La siguiente expresión añade 10 días a todas las fechas del puerto SHIP_DATE:

```
ADD_TO_DATE( SHIP_DATE, 'DD', 10 )
```

- **Hora.** Especifique un entero positivo o negativo en el argumento *amount* . Utilice una de las siguientes cadenas de formato de hora: HH, HH12, HH24. La siguiente expresión añade 14 horas a todas las fechas del puerto SHIP_DATE:

```
ADD_TO_DATE( SHIP_DATE, 'HH', 14 )
```

- **Minuto.** Especifique un entero positivo o negativo en el argumento *amount* . Utilice la cadena de formato MI para definir el minuto. La siguiente expresión añade 25 minutos a todas las fechas del puerto SHIP_DATE:

```
ADD_TO_DATE( SHIP_DATE, 'MI', 25 )
```

- **Segundos.** Especifique un entero positivo o negativo en el argumento *amount* . Utilice la cadena de formato SS para definir el segundo. La siguiente expresión añade 59 segundos a todas las fechas del puerto SHIP_DATE:

```
ADD_TO_DATE( SHIP_DATE, 'SS', 59 )
```

- **Milisegundos.** Especifique un entero positivo o negativo en el argumento *amount* . Utilice la cadena de formato MS para definir los milisegundos. La siguiente expresión añade 125 milisegundos a todas las fechas del puerto SHIP_DATE:

```
ADD_TO_DATE( SHIP_DATE, 'MS', 125 )
```

- **Microsegundos.** Especifique un entero positivo o negativo en el argumento *amount* . Utilice la cadena de formato US para definir los microsegundos. La siguiente expresión añade 2.000 microsegundos a todas las fechas del puerto SHIP_DATE:

```
ADD_TO_DATE( SHIP_DATE, 'US', 2000 )
```

- **Nanosegundos.** Especifique un entero positivo o negativo en el argumento *amount* . Utilice la cadena de formato NS para definir los nanosegundos. La siguiente expresión añade 3.000.000 de nanosegundos a todas las fechas del puerto SHIP_DATE:

```
ADD_TO_DATE( SHIP_DATE, 'NS', 3000000 )
```

Sintaxis

```
ADD_TO_DATE( date, format, amount )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>date</i>	Obligatorio	Tipo de dato de fecha y hora. Transfiere los valores que desea cambiar. Puede especificar cualquier expresión de transformación válida.
<i>format</i>	Obligatorio	Una cadena de formato que especifica la parte del valor de fecha que desea cambiar. Encierre entre comillas simples la cadena de formato, por ejemplo, 'mm'. La cadena de formato no distingue entre mayúsculas y minúsculas.
<i>amount</i>	Obligatorio	Un entero que especifica el número de años, meses, días, horas, etc. por los que desea cambiar el valor de fecha. Puede especificar cualquier expresión de transformación válida que se verifique en un entero.

Valor de devolución

La fecha en el mismo formato que la fecha que transfirió a esta función.

NULL si se transfiere un valor nulo como un argumento de la función.

Ejemplos

Las siguientes expresiones añaden todas un mes a todas las fechas del puerto DATE_SHIPPED: Si transfiere un valor que crea un día que no existe en un determinado mes, el Servicio de integración de datos devuelve el último día del mes. Por ejemplo, si añade un mes al 31 de enero de 1998, el Servicio de integración de datos devuelve 28 de febrero de 1998.

Recuerde además que ADD_TO_DATE reconoce los años bisiestos y que añade un mes al 29 de enero de 2000:

```
ADD_TO_DATE( DATE_SHIPPED, 'MM', 1 )
ADD_TO_DATE( DATE_SHIPPED, 'MON', 1 )
ADD_TO_DATE( DATE_SHIPPED, 'MONTH', 1 )
```

DATE_SHIPPED	RETURN VALUE
Jan 12 1998 12:00:30AM	Feb 12 1998 12:00:30AM
Jan 31 1998 6:24:45PM	Feb 28 1998 6:24:45PM
Jan 29 2000 5:32:12AM	Feb 29 2000 5:32:12AM (Leap Year)
Oct 9 1998 2:30:12PM	Nov 9 1998 2:30:12PM
NULL	NULL

Las siguientes expresiones restan 10 días a cada fecha del puerto DATE_SHIPPED:

```
ADD_TO_DATE( DATE_SHIPPED, 'D', -10 )
ADD_TO_DATE( DATE_SHIPPED, 'DD', -10 )
ADD_TO_DATE( DATE_SHIPPED, 'DDD', -10 )
```

```
ADD_TO_DATE( DATE_SHIPPED, 'DY', -10 )
ADD_TO_DATE( DATE_SHIPPED, 'DAY', -10 )
```

DATE_SHIPPED	RETURN VALUE
Jan 1 1997 12:00:30AM	Dec 22 1996 12:00AM
Jan 31 1997 6:24:45PM	Jan 21 1997 6:24:45PM
Mar 9 1996 5:32:12AM	Feb 29 1996 5:32:12AM (Leap Year)
Oct 9 1997 2:30:12PM	Sep 30 1997 2:30:12PM
Mar 3 1996 5:12:20AM	Feb 22 1996 5:12:20AM
NULL	NULL

Las siguientes expresiones restan 15 horas a cada fecha del puerto DATE_SHIPPED:

```
ADD_TO_DATE( DATE_SHIPPED, 'HH', -15 )
ADD_TO_DATE( DATE_SHIPPED, 'HH12', -15 )
ADD_TO_DATE( DATE_SHIPPED, 'HH24', -15 )
```

DATE_SHIPPED	RETURN VALUE
Jan 1 1997 12:00:30AM	Dec 31 1996 9:00:30AM
Jan 31 1997 6:24:45PM	Jan 31 1997 3:24:45AM
Oct 9 1997 2:30:12PM	Oct 8 1997 11:30:12PM
Mar 3 1996 5:12:20AM	Mar 2 1996 2:12:20PM
Mar 1 1996 5:32:12AM	Feb 29 1996 2:32:12PM (Leap Year)
NULL	NULL

Cómo trabajar con fechas

Utilice los siguientes consejos cuando trabaje con ADD_TO_DATE:

- Puede agregar o restar cualquier parte de la fecha si especifica una cadena de formato y configura el argumento *amount* como un entero positivo o negativo.
- Si transfiere un valor que crea un día que no existe en un determinado mes, el Servicio de integración de datos devuelve el último día del mes. Por ejemplo, si añade un mes al 31 de enero de 1998, el Servicio de integración de datos devuelve 28 de febrero de 1998.
- Puede anidar las funciones TRUNC y ROUND para manipular fechas.
- Puede anidar TO_DATE para convertir cadenas en fechas.
- ADD_TO_DATE cambia solo una parte de la fecha, la que usted especifique. Si modifica una fecha de manera que cambia del horario estándar al horario de verano, debe cambiar la parte de hora de la fecha.

AES_DECRYPT

Devuelve datos descifrados en formato de cadena. El Servicio de integración de datos usa el algoritmo del estándar de cifrado avanzado (AES, Advanced Encryption Standard) con una codificación de 128 bits. El algoritmo AES es un algoritmo de cifrado aprobado por FIPS.

Sintaxis

```
AES_DECRYPT ( value, key )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>value</i>	Obligatorio	Tipo de datos Binary. Es el valor que se va a descifrar.
<i>key</i>	Obligatorio	Tipo de datos String. Precisión de 16 caracteres o menos. Puede usar variables de asignación para la clave. Para descifrar un valor, use la misma clave que ha usado para cifrarlo.

Valor devuelto

Valor binario descifrado.

Se devuelve NULL si el valor de entrada es un valor nulo.

Ejemplo

En el siguiente ejemplo, se devuelven números de la seguridad social descifrados. En este ejemplo, el Servicio de integración de datos obtiene la clave de los tres primeros números del número de la seguridad social mediante la función SUBSTR:

```
AES_DECRYPT( SSN_ENCRYPT, SUBSTR( SSN,1,3 ) )
```

SSN_ENCRYPT	DECRYPTED VALUE
07FB945926849D2B1641E708C85E4390	832-17-1672
9153ACAB89D65A4B81AD2ABF151B099D	832-92-4731
AF6B5E4E39F974B3F3FB0F22320CC60B	832-46-7552
992D6A5D91E7F59D03B940A4B1CBBCBE	832-53-6194
992D6A5D91E7F59D03B940A4B1CBBCBE	832-81-9528

AES_ENCRYPT

Devuelve los datos en formato cifrado. El Servicio de integración de datos utiliza el algoritmo estándar de cifrado avanzado (AES) con codificación de 128 bits. El algoritmo AES es un algoritmo criptográfico aprobado por FIPS.

Utilice esta función para evitar que los datos confidenciales sean visibles para otras personas. Por ejemplo, para almacenar los números de seguridad social en un almacén de datos, utilice la función AES_ENCRYPT para cifrar los números de seguridad social y mantener la confidencialidad.

Sintaxis

```
AES_ENCRYPT ( value, key )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>value</i>	Obligatorio	Tipo de datos String. Valor que desea cifrar.
<i>key</i>	Obligatorio	Tipo de datos String. Precisión de 16 caracteres o menos. Puede utilizar variables de asignación para la clave.

Valor de retorno

Valor binario cifrado.

NULL si la entrada es un valor nulo.

Ejemplo

El ejemplo siguiente devuelve los valores cifrados de números de la seguridad social. En este ejemplo, el Servicio de integración de datos obtiene la clave de los tres primeros números del número de la seguridad social mediante la función SUBSTR:

```
AES_ENCRYPT( SSN, SUBSTR( SSN,1,3 ))
```

SSN	ENCRYPTED VALUE
832-17-1672	07FB945926849D2B1641E708C85E4390
832-92-4731	9153ACAB89D65A4B81AD2ABF151B099D
832-46-7552	AF6B5E4E39F974B3F3FB0F22320CC60B
832-53-6194	992D6A5D91E7F59D03B940A4B1CBBCBE
832-81-9528	992D6A5D91E7F59D03B940A4B1CBBCBE

Consejo

Si el destino no admite datos binarios, utilice AES_ENCRYPT con la función ENC_BASE64 para almacenar los datos en un formato compatible con la base de datos.

ANY

Devuelve cualquier fila del puerto seleccionado. Opcionalmente, se puede aplicar un filtro para limitar el número de filas que lee el servicio de integración de datos. Dentro de ANY solamente se puede anidar una función agregada adicional.

Sintaxis

```
ANY( value [, filter_condition ] )
```

En la siguiente tabla se describen los argumentos de esta función:

Argumento	Obligatorio / Opcional	Descripción
<i>value</i>	Obligatorio	Cualquier tipo de datos excepto Binario. Pasa los valores para los que desea devolver cualquier fila. Se puede especificar cualquier expresión de transformación válida.
<i>filter_condition</i>	Opcional	Limita las filas en la búsqueda. La condición de filtro debe ser un valor numérico o dar un valor TRUE, FALSE o NULL. Se puede especificar cualquier expresión de transformación válida.

Valor devuelto

Cualquier fila de un puerto. Devuelve una fila diferente cada vez.

NULL si todos los valores pasados a la función son NULL o si no se ha seleccionado ninguna fila. Por ejemplo, la condición de filtro da un valor FALSE o NULL para todas las filas.

Ejemplo

La expresión siguiente devuelve cualquier fila del puerto ITEM_NAME con un precio mayor de 10,00 \$:

```
ANY( ITEM_NAME, ITEM_PRICE > 10 )
```

ITEM_NAME	ITEM_PRICE
Flashlight	35.00
Navigation Compass	8.05
Regulator System	150.00
Flashlight	29.00
Depth/Pressure Gauge	88.00
Vest	31.00

RETURN VALUE:Flashlight

Tipos de datos complejos y ANY

ANY se puede utilizar para devolver una fila en un puerto complejo de tipo matriz o estructura.

Supongamos que tiene la siguiente matriz:

```
emp_phones =  
[205-128-6478, 722-515-2889]  
[107-081-0961, 718-051-8116]  
[344-894-6463, 861-411-8361]  
[107-031-0961, NULL]
```

Puede utilizar la siguiente expresión para devolver cualquier fila del puerto de matriz:

```
ANY( emp_phones )
```

VALOR DE DEVOLUCIÓN: [205-128-6478, 722-515-2889]

ARRAY

Genera una matriz con elementos basada en los argumentos especificados.

Sintaxis

```
ARRAY (array_element1 as any [, array_element2] ...)
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio/ Opcional	Descripción
array_element1	Obligatorio	Cualquier tipo de datos. El elemento que desea añadir a la matriz. Se puede especificar cualquier expresión de transformación válida.
array_element2	Opcional	El mismo tipo de datos que array_element1.

Si utiliza la función ARRAY en una expresión de salida para un puerto de matriz, el tipo de datos de los argumentos de la función debe coincidir con el tipo de datos de los elementos de matriz especificados en la configuración de tipos para el puerto de matriz.

Valor de devolución

Matriz.

El tipo de datos de los argumentos determina el tipo de datos de los elementos de matriz. Por ejemplo, si pasa argumentos de cadena, la función genera una matriz de elementos de cadena.

Ejemplos

La siguiente expresión genera una matriz de elementos de cadena.

```
ARRAY (work_phone, home_phone)
```

work_phone	home_phone	RETURN VALUE
205-128-6478	722-515-2889	[205-128-6478,722-515-2889]
107-081-0961	718-051-8116	[107-081-0961,718-051-8116]
344-894-6463	861-411-8361	[344-894-6463,861-411-8361]
107-031-0961	NULL	[107-031-0961,NULL]

ASCII

Cuando el Servicio de integración de datos utiliza el modo ASCII, la función ASCII devuelve el valor numérico ASCII del primer carácter de la cadena pasada a la función.

Cuando el Servicio de integración de datos utiliza el modo Unicode, la función ASCII devuelve el valor numérico Unicode del primer carácter de la cadena pasada a la función. Los valores Unicode se encuentran en el rango comprendido entre 0 y 65.535.

Se puede pasar una cadena de cualquier tamaño a ASCII, aunque solamente evalúa el primer carácter de la cadena. Antes de pasar cualquier valor de cadena a ASCII, se puede analizar el carácter específico que se desea convertir en un valor ASCII o Unicode. Por ejemplo, se puede usar RTRIM u otra función de manipulación de cadenas. Si se pasa un valor numérico, ASCII lo convierte en una cadena de caracteres y devuelve el valor ASCII o Unicode del primer carácter de la cadena.

Esta función tiene un comportamiento idéntico al de la función CHRCODE. Si se utiliza ASCII en expresiones existentes, estas seguirán funcionando correctamente. Sin embargo, cuando se creen expresiones nuevas, utilice la función CHRCODE en lugar de la función ASCII.

Sintaxis

```
ASCII ( string )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>string</i>	Obligatorio	Cadena de caracteres. Pasa el valor que se desea devolver como valor ASCII. Se puede especificar cualquier expresión de transformación válida.

Valor devuelto

Entero. El valor ASCII o Unicode del primer carácter de la cadena.

NULL si el valor pasado a la función es NULL.

Ejemplo

La siguiente expresión devuelve el valor ASCII o Unicode para el primer carácter de cada valor en el puerto ITEMS:

```
ASCII( ITEMS )
```

ITEMS	RETURN VALUE
Flashlight	70
Compass	67
Safety Knife	83
Depth/Pressure Gauge	68
Regulator System	82

AVG

Devuelve el promedio de todos los valores de un grupo de filas. Opcionalmente, se puede aplicar un filtro para limitar el número de filas que se leen para calcular el promedio. Dentro de AVG solamente se puede anidar una función agregada y la función anidada debe devolver un tipo de datos Numérico.

Sintaxis

```
AVG( numeric_value [, filter_condition ] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento:	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. Pasa los valores para los cuales desea calcular una media. Se puede especificar cualquier expresión de transformación válida.
<i>filter_condition</i>	Opcional	Limita las filas en la búsqueda. La condición de filtro debe ser un valor numérico o dar un valor TRUE, FALSE o NULL. Se puede especificar cualquier expresión de transformación válida.

Valor de retorno

Valor numérico.

NULL si todos los valores pasados a la función son NULL o si no se ha seleccionado ninguna fila. Por ejemplo, la condición de filtro da un valor FALSE o NULL para todas las filas.

Nota: Si el valor devuelto es decimal con una precisión superior a 15, puede habilitar la alta precisión para garantizar una precisión decimal de hasta 38 dígitos.

Nulos

Si un valor es NULL, AVG omite la fila. Sin embargo, si todos los valores pasados desde el puerto son NULL, AVG devuelve NULL.

Agrupar por

AVG agrupa valores por grupos según los puertos definidos en la transformación, devolviendo un resultado para cada grupo.

Si no hay un grupo por puerto, AVG trata todas las filas como un solo grupo, devolviendo un valor.

Ejemplo

La siguiente expresión devuelve el coste mayorista promedio de las linternas:

```
AVG( WHOLESALE_COST, ITEM_NAME='Flashlight' )
```

ITEM_NAME	WHOLESALE_COST
Flashlight	35.00
Navigation Compass	8.05
Regulator System	150.00
Flashlight	29.00

ITEM_NAME	WHOLESALE_COST
Depth/Pressure Gauge	88.00
Flashlight	31.00

RETURN VALUE: 31.66

Consejo

Se pueden realizar operaciones aritméticas con los valores pasados a AVG antes de que la función calcule el promedio. Por ejemplo:

```
AVG( QTY * PRICE - DISCOUNT )
```

CAST

Cambia el nombre de los elementos y modifica el tipo de datos de cada elemento para el valor de estructura dado basándose en los tipos de datos de la definición de tipo de datos complejos especificada.

Sintaxis

```
CAST (:Type.type_definition_library.type_definition, struct_value)
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio/Opcional	Descripción
:Type.type_definition_library.type_definition	Obligatorio	La definición de tipo de datos complejos que representa el esquema de los datos de estructura. Utilice el calificador de referencia :Type para hacer referencia a la biblioteca de definiciones de tipos que contiene la definición de tipo de datos complejos.
struct_value	Obligatorio	El valor de estructura para el que desea cambiar el tipo de datos de los elementos de estructura. Se puede introducir cualquier expresión de transformación válida que dé como resultado una estructura.

El tipo de datos del valor de estructura y el tipo de datos de la definición de tipo de datos complejos deben ser compatibles.

Valor de devolución

Estructura.

Ejemplos

La siguiente expresión cambia los tipos de datos de los elementos del puerto de estructura h2_sales en función de los tipos de datos de la definición de tipo de datos complejos h1_sales_def.

```
CAST (:Type.type_definition_library.h1_sales_def, h2_sales)
```

h1_sales_def	h2_sales	RETURN VALUE
{ q1_total : bigint q2_total : double }	{ q3_total : int q4_total : int }	{ q1_total : bigint q2_total : double }

CEIL

Devuelve el entero más pequeño mayor o igual al valor numérico que se ha pasado a esta función. Por ejemplo, si se pasa 3,14 a CEIL, la función devuelve 4. Si se pasa 3,98 a CEIL, la función devuelve 4. Del mismo modo, si se pasa -3,17 a CEIL, la función devuelve -3.

Sintaxis

```
CEIL( numeric_value )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio/Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. Se puede especificar cualquier expresión de transformación válida.

Valor de devolución

Valor numérico.

Valor doble si se pasa un valor numérico con precisión declarada mayor que 38.

NULL si el valor pasado a la función es NULL.

Ejemplo

La expresión siguiente devuelve el precio redondeado al entero más próximo:

```
CEIL( PRICE )
```

PRICE	RETURN VALUE
39.79	40
125.12	126
74.24	75

PRICE	RETURN VALUE
NULL	NULL
-100.99	-100

Sugerencia: Se pueden realizar operaciones aritméticas con los valores pasados a CEIL antes de que CEIL devuelva el valor entero más próximo. Por ejemplo, si quiere multiplicar un valor numérico por 10 antes de calcular el entero más pequeño mayor que el valor modificado, debería escribir la función de la siguiente manera:

```
CEIL( PRICE * 10 )
```

CHOOSE

Permite elegir una cadena de una lista de cadenas en función de una posición determinada. Debe especificar la posición y el valor. Si el valor coincide con la posición, el Servicio de integración de datos devuelve el valor.

Sintaxis

```
CHOOSE( index, string1 [, string2, ..., stringN] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>index</i>	Obligatorio	Tipo de datos numérico. Especifique un número según la posición del valor con el que desea establecer la coincidencia.
<i>string</i>	Obligatorio	Cualquier valor de carácter.

Valor devuelto

Cadena coincidente con la posición del valor de índice.

Se devuelve NULL si ninguna cadena coincide con la posición del valor de índice.

Ejemplo

La siguiente expresión devuelve la cadena 'flashlight' según un valor de índice de 2:

```
CHOOSE( 2, 'knife', 'flashlight', 'diving hood' )
```

La siguiente expresión devuelve NULL según un valor de índice de 4:

```
CHOOSE( 4, 'knife', 'flashlight', 'diving hood' )
```

CHOOSE devuelve NULL porque la expresión no contiene un cuarto argumento.

CHR

Cuando el Servicio de integración de datos utiliza el modo ASCII, CHR devuelve el carácter ASCII correspondiente al valor numérico que se pasa a esta función. Los valores ASCII se encuentran en el rango comprendido entre 0 y 255. Se puede pasar cualquier número entero a CHR, pero sólo los códigos ASCII del 32 al 126 son caracteres imprimibles.

Cuando el Servicio de integración de datos utiliza el modo Unicode, CHR devuelve el carácter Unicode correspondiente al valor numérico que se pasa a esta función. Los valores Unicode se encuentran en el rango comprendido entre 0 y 65.535.

Sintaxis

```
CHR( numeric_value )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. El valor que desea devolver como carácter ASCII o Unicode. Se puede especificar cualquier expresión de transformación válida.

Valor devuelto

Carácter ASCII o Unicode. Una cadena que contiene un carácter.

NULL si el valor pasado a la función es NULL.

Ejemplo

La siguiente expresión devuelve el carácter ASCII o Unicode para cada valor numérico en el puerto ITEM_ID:

```
CHR( ITEM_ID )
```

ITEM_ID	RETURN VALUE
65	A
122	z
NULL	NULL
88	X
100	d
71	G

Utilice la función CHR para concatenar una comilla simple a una cadena. La comilla simple es el único carácter que no se puede utilizar dentro de una cadena literal. Examine el siguiente ejemplo:

```
'Joan' || CHR(39) || 's car'
```

El valor devuelto es:

```
Joan's car
```


CHRCODE

Cuando el Servicio de integración de datos utiliza el modo ASCII, CHRCODE devuelve el valor numérico ASCII del primer carácter de la cadena pasada a la función. Los valores ASCII se encuentran en el rango comprendido entre 0 y 255.

Cuando el Servicio de integración de datos utiliza el modo Unicode, CHRCODE devuelve el valor numérico Unicode del primer carácter de la cadena pasada a la función. Los valores Unicode se encuentran en el rango comprendido entre 0 y 65.535.

Normalmente, antes de pasar cualquier valor de cadena a CHRCODE, se puede analizar el carácter específico que se desea convertir en un valor ASCII o Unicode. Por ejemplo, se puede usar RTRIM u otra función de manipulación de cadenas. Si se pasa un valor numérico, CHRCODE lo convierte en una cadena de caracteres y devuelve el valor ASCII o Unicode del primer carácter de la cadena.

Esta función tiene un comportamiento idéntico al de la función ASCII. Si se utiliza ASCII en expresiones, estas seguirán funcionando correctamente. Sin embargo, cuando se creen expresiones nuevas, utilice la función CHRCODE en lugar de la función ASCII.

Sintaxis

```
CHRCODE ( string )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>string</i>	Obligatorio	Cadena de caracteres. Pasa los valores que se desean devolver como valores ASCII o Unicode. Se puede especificar cualquier expresión de transformación válida.

Valor devuelto

Entero.

NULL si el valor pasado a la función es NULL.

Ejemplo

La siguiente expresión devuelve el valor ASCII o Unicode para el primer carácter de cada valor en el puerto ITEMS:

```
CHRCODE( ITEMS )
```

ITEMS	RETURN VALUE
Flashlight	70
Compass	67
Safety Knife	83
Depth/Pressure Gauge	68
Regulator System	82

COLLECT_LIST

Devuelve una matriz con elementos basada en el argumento. El tipo de datos del argumento determina el tipo de datos de la matriz. COLLECT_LIST es una función agregada.

Sintaxis

```
COLLECT_LIST(value as any)
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio/Opcional	Descripción
value	Obligatorio	Cualquier tipo de datos. Los valores que desea agregar en un dato jerárquico del tipo matriz. Se puede especificar cualquier expresión de transformación válida.

Valor devuelto

Matriz.

Agrupar por

COLLECT_LIST agrupa los valores en función de los grupos por puerto que haya definido en la transformación y devuelve un resultado por cada grupo.

Si no hay ningún grupo por puerto, COLLECT_LIST trata todas las filas como un grupo y devuelve un valor.

Ejemplos

La siguiente expresión devuelve una matriz con los elementos en PRODUCT_NAME.

```
COLLECT_LIST(PRODUCT_NAME)
```

PRODUCT_NAME

Flashlight

Compass

Pressure Gauge

Vest

VALOR DE DEVOLUCIÓN: [Flashlight,Compass,Pressure Gauge,Vest]

COLLECT_MAP

Devuelve una asignación con elementos basados en los argumentos especificados.

Sintaxis

```
COLLECT_MAP(map_key as ANY, map_value as ANY)
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio/ Opcional	Descripción
map_key	Obligatorio	Cualquier tipo de datos primitivo. Los elementos que desea agregar como claves de una asignación de tipos de datos jerárquicos. Se puede especificar cualquier expresión de transformación válida.
map_value	Obligatorio	Cualquier tipo de datos primitivo o complejo. Los elementos que desea agregar como valores de una asignación de tipos de datos jerárquicos. Se puede especificar cualquier expresión de transformación válida.

Valor de devolución

Asignación.

Agrupar por

COLLECT_MAP agrupa los valores en función de los grupos por puertos que haya definido en la transformación y devuelve un resultado por cada grupo.

Si no hay ningún grupo por puerto, COLLECT_MAP trata todas las filas como un grupo y devuelve un valor.

Ejemplos

La siguiente expresión devuelve una asignación con los elementos de PRODUCT_ID como claves y los elementos de PRODUCT_NAME como valores.

```
COLLECT_MAP (PRODUCT_ID, PRODUCT_NAME)
```

PRODUCT_ID	PRODUCT_NAME
34890	Flashlight
12754	Compass
54028	Pressure Gauge
81203	Vest

VALOR DE DEVOLUCIÓN:

```
[34890 -> Flashlight, 12754 -> Compass, 54028 -> Pressure Gauge, 81203 -> Vest]
```

COMPRESS

Permite comprimir datos mediante el algoritmo de compresión zlib 1.2.1. Use la función COMPRESS antes de enviar grandes cantidades de datos en una red de área extensa.

Sintaxis

```
COMPRESS( value )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio/ Opcional	Descripción
<i>valor</i>	Obligatorio	Tipo de datos String. Datos que se van a comprimir.

Valor devuelto

Valor binario comprimido del valor de entrada.

NULL si la entrada es un valor nulo.

Ejemplo

Su organización tiene un servicio de pedidos en línea. En este caso, es posible que desee enviar los datos de los pedidos de los clientes en una red de área extensa. Los datos de origen contienen una fila de 10 MB. Puede comprimir los datos de esta fila mediante la función COMPRESS. Al comprimir los datos, se reduce la cantidad de datos que el Servicio de integración de datos escribe en la red. El resultado es un aumento del rendimiento.

CONCAT

Concatena dos cadenas. CONCAT convierte todos los datos en texto antes de concatenar las cadenas. De forma alternativa, utilice el operador de cadena || para concatenar cadenas. El uso del operador de cadena || en lugar de CONCAT mejora el rendimiento del Servicio de integración de datos.

Sintaxis

```
CONCAT( first_string, second_string )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>first_string</i>	Obligatorio	Cualquier tipo de datos excepto Binario. La primera parte de la cadena que se desea concatenar. Se puede especificar cualquier expresión de transformación válida.
<i>second_string</i>	Obligatorio	Cualquier tipo de datos excepto Binario. La segunda parte de la cadena que se desea concatenar. Se puede especificar cualquier expresión de transformación válida.

Valor de retorno

Cadena.

NULL si ambos valores de cadena son NULL.

Nulos

Si una de las cadenas es NULL, CONCAT la omite y devuelve la otra cadena.

Si ambas cadenas son NULL, CONCAT devuelve NULL.

Ejemplo

La siguiente expresión concatena los nombres en los puertos FIRST_NAME y LAST_NAME:

```
CONCAT( FIRST_NAME, LAST_NAME )
```

FIRST_NAME	LAST_NAME	RETURN VALUE
John	Baer	JohnBaer
NULL	Campbell	Campbell
Bobbi	Apperley	BobbiApperley
Jason	Wood	JasonWood
Dan	Covington	DanCovington
Greg	NULL	Greg
NULL	NULL	NULL
100	200	100200

CONCAT no añade espacios a cadenas individuales. Si desea añadir un espacio entre dos cadenas, puede escribir una expresión con dos funciones CONCAT anidadas. Por ejemplo, la siguiente expresión concatena un espacio al final del nombre y luego concatena el apellido:

```
CONCAT( CONCAT( FIRST_NAME, ' ' ), LAST_NAME )
```

FIRST_NAME	LAST_NAME	RETURN VALUE
John	Baer	John Baer
NULL	Campbell	Campbell <i>(includes leading blank)</i>
Bobbi	Apperley	Bobbi Apperley
Jason	Wood	Jason Wood
Dan	Covington	Dan Covington
Greg	NULL	Greg
NULL	NULL	NULL

Use las funciones CHR y CONCAT para concatenar una comilla simple en una cadena. La comilla simple es el único carácter que no se puede usar dentro de un literal de cadena. Vea el siguiente ejemplo:

```
CONCAT( 'Joan', CONCAT( CHR(39), 's car' ) )
```

El valor de retorno es:

```
Joan's car
```

CONCAT_ARRAY

Concatena elementos de cadena en una matriz basada en el separador que especifique y devuelve una cadena.

Sintaxis

```
CONCAT_ARRAY(' ', array)
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio/Opcional	Descripción
' '	Obligatorio	Cada elemento de cadena está separado por el separador que especifique. Por ejemplo, "," separa los valores con una coma.
array	Obligatorio	Una matriz con elementos de tipo de cadena. La matriz que desea concatenar.

Valor de devolución

Cadena

Nulos

Si uno de los elementos de cadena es NULL, CONCAT_ARRAY lo ignora y devuelve la otra cadena.

Si todos los elementos de cadena son NULL, CONCAT_ARRAY devuelve una cadena vacía.

Ejemplos

La siguiente expresión concatena los elementos de cadena de la matriz.

```
CONCAT_ARRAY( ':', Name )
```

Name	RETURN VALUE
['John', 'Baer']	'John:Baer'
['Bobbi', 'Apperley']	'Bobbi:Apperley'
['Jason', '']	'Jason:'
['Greg', NULL]	'Greg'
[NULL, NULL]	''

CONVERT_BASE

Convierte una cadena numérica no negativa de un valor base a otro valor base.

Sintaxis

```
CONVERT_BASE( value, source_base, dest_base )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>valor</i>	Obligatorio	Tipo de datos String. Valor que se va a convertir de una base a otra base. El valor máximo es 9.233.372.036.854.775.806.
<i>source_base</i>	Obligatorio	Tipo de datos numérico. Valor base actual de los datos que se van a convertir. La base mínima es 2. La base máxima es 36.
<i>dest_base</i>	Obligatorio	Tipo de datos numérico. Valor base al que se van a convertir los datos. La base mínima es 2. La base máxima es 36.

Valor devuelto

Valor numérico.

Ejemplo

En el siguiente ejemplo se convierte 2222 del valor base decimal 10 al valor base binario 2:

```
CONVERT_BASE( "2222", 10, 2 )
```

El Servicio de integración de datos devuelve 100010101110.

COS

Devuelve el coseno de un valor numérico (expresado en radianes).

Sintaxis

```
COS( numeric_value )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. Datos numéricos expresados en radianes (grados multiplicados por pi y divididos entre 180). Pasa los valores para los que se va a calcular el coseno. Puede especificar cualquier expresión de transformación válida.

Valor devuelto

Valor doble.

Se devuelve NULL si el valor pasado a la función es NULL.

Ejemplo

La siguiente expresión devuelve el coseno para todos los valores del puerto Degrees:

```
COS( DEGREES * 3.14159265359 / 180 )
```

DEGREES	RETURN VALUE
0	1.0
90	0.0
70	0.342020143325593
30	0.866025403784421
5	0.996194698091745
18	0.951056516295147
89	0.0174524064371813
NULL	NULL

Sugerencia: Puede realizar una operación aritmética con los valores pasados a COS antes de que la función calcule el coseno. Por ejemplo, puede convertir los valores del puerto en radianes antes de calcular el coseno del modo siguiente:

```
COS( ARCS * 3.14159265359 / 180 )
```

COSH

Devuelve el coseno hiperbólico de un valor numérico (expresado en radianes).

Sintaxis

```
COSH( numeric_value )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. Datos numéricos expresados en radianes (grados multiplicados por pi y divididos entre 180). Pasa los valores para los que se va a calcular el coseno hiperbólico. Puede especificar cualquier expresión de transformación válida.

Valor devuelto

Valor doble.

Se devuelve NULL si el valor pasado a la función es NULL.

Ejemplo

La siguiente expresión devuelve el coseno hiperbólico para los valores del puerto Angles:

```
COSH( ANGLES )
```

ANGLES	RETURN VALUE
1.0	1.54308063481524
2.897	9.0874465864177
3.66	19.4435376920294
5.45	116.381231106176
0	1.0
0.345	1.06010513656773
NULL	NULL

Sugerencia: Puede realizar una operación aritmética con los valores pasados a COSH antes de que la función calcule el coseno hiperbólico. Por ejemplo:

```
COSH( MEASURES.ARCS / 360 )
```

COUNT

Devuelve el número de filas que tienen valores no nulos en un grupo. Opcionalmente, se puede incluir el argumento asterisco (*) para contar todos los valores de entrada de una transformación. Dentro de COUNT solamente se puede anidar una función agregada adicional. Se puede aplicar una condición para filtrar filas antes de contarlas.

Sintaxis

```
COUNT( value [, filter_condition] )
```

o

```
COUNT( * [, filter_condition] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>value</i>	Obligatorio	Cualquier tipo de datos a excepción de los binarios. Pasa los valores que se desea contar. Se puede especificar cualquier expresión de transformación válida.
*	Opcional	Se usa para contar <i>todas las filas</i> de una transformación.
<i>filter_condition</i>	Opcional	Limita las filas en la búsqueda. La condición de filtro debe ser un valor numérico o dar un valor TRUE, FALSE o NULL. Se puede especificar cualquier expresión de transformación válida.

Valor devuelto

Entero.

0 si todos los valores pasados a esta función son NULL (salvo que se incluya el argumento de asterisco).

Nulos

Si todos los valores son NULL, la función devuelve 0.

Si se aplica el argumento de asterisco, esta función cuenta todas las filas, independientemente de que una columna de una fila contenga un valor nulo.

Si se aplica el argumento *value*, esta función omite las columnas con valores nulos.

Agrupar por

COUNT agrupa valores por grupos según los puertos definidos en la transformación, devolviendo un resultado para cada grupo. Si no hay un grupo por puerto, COUNT trata todas las filas como un solo grupo, devolviendo un valor.

Ejemplos

La siguiente expresión cuenta los artículos con menos de 5 unidades en stock, excluyendo los valores nulos:

```
COUNT( ITEM_NAME, IN_STOCK < 5 )
```

ITEM_NAME	IN_STOCK
Flashlight	10
NULL	2
Compass	NULL
Regulator System	5
Safety Knife	8
Halogen Flashlight	1
RETURN VALUE: 1	

En este ejemplo, la función ha contado la linterna Halogen, pero no el artículo NULL. La función cuenta todas las filas de una transformación, incluidos los valores nulos, tal como se ilustra en el siguiente ejemplo:

```
COUNT( *, QTY < 5 )
```

ITEM_NAME	QTY
Flashlight	10
NULL	2
Compass	NULL
Regulator System	5
Safety Knife	8
Halogen Flashlight	1

RETURN VALUE: 2

En este ejemplo, la función cuenta el artículo NULL y la linterna Halogen. Si se incluye el argumento de asterisco, pero no usa un filtro, la función cuenta todas las filas que se pasan a la transformación. Por ejemplo:

```
COUNT( * )
```

ITEM_NAME	QTY
Flashlight	10
NULL	2
Compass	NULL
Regulator System	5
Safety Knife	8
Halogen Flashlight	1

RETURN VALUE: 6

Tipos de datos complejos y COUNT

COUNT se puede utilizar para contar el número de filas en un puerto complejo de tipo matriz o estructura.

Supongamos que tiene la siguiente matriz:

```
emp_phones =  
[205-128-6478, 722-515-2889]  
[107-081-0961, 718-051-8116]  
[344-894-6463, 861-411-8361]  
[107-031-0961, NULL]
```

Puede utilizar la siguiente expresión para contar el número de filas en el puerto de matriz:

```
COUNT( emp_phones )
```

VALOR DE DEVOLUCIÓN: 4

CRC32

Devuelve un valor de comprobación de redundancia cíclica de 32 bits (CRC32). Utilice CRC32 para encontrar errores de transmisión de datos. También puede utilizar CRC32 si desea comprobar que los datos almacenados en un archivo no se han modificado.

Si utiliza CRC32 para realizar una comprobación de redundancia de datos en modo ASCII y modo Unicode, el Servicio de integración de datos podría generar resultados diferentes para el mismo valor de entrada. Si utiliza CRC32 para realizar una comprobación de redundancia de datos en diferentes sistemas operativos, el Servicio de integración de datos podría generar resultados diferentes para el mismo valor de entrada.

Nota: CRC32 puede devolver el mismo resultado para cadenas de entrada diferentes. Si desea generar claves en una asignación, utilice una transformación de generador de secuencia. Si utiliza CRC32 para generar claves en una asignación, es posible que obtenga resultados inesperados.

Sintaxis

```
CRC32( value )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>valor</i>	Obligatorio	Tipo de datos String o Binary. Pasa los valores sobre los que desea realizar una comprobación de redundancia. El valor de entrada distingue entre mayúsculas y minúsculas. La distinción entre mayúsculas y minúsculas del valor de entrada afecta al valor de retorno. Por ejemplo, CRC32 (informatica) y CRC32 (Informatica) devuelven valores diferentes.

Valor de retorno

Valor entero de 32 bits.

Ejemplo

Desea leer datos de un origen en una red de área amplia. Quiere saber si los datos han sido modificados durante la transmisión. Puede calcular la suma de comprobación de los datos del archivo y guardarla junto con el archivo. Cuando se leen los datos del origen, el Servicio de integración de datos puede usar CRC32 para calcular la suma de comprobación y compararla con el valor almacenado. Si los dos valores son iguales, los datos no se han modificado.

CREATE_TIMESTAMP_TZ

Cree una marca de tiempo con el tipo de datos Zona horaria a partir de los valores de marca de tiempo y zona horaria.

El puerto de salida debe ser timestampWithTZ para las expresiones CREATE_TIMESTAMP_TZ.

Sintaxis

```
CREATE_TIMESTAMP_TZ (timestamp_value, timezone_value)
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>timestamp_value</i>	Obligatorio	Tipo de dato de fecha y hora. Se puede especificar cualquier expresión de transformación válida.
<i>timezone_value</i>	Obligatorio	Debe ser un tipo de datos de cadena. La cadena debe ser una cadena de caracteres. Pasa los valores que desea crear para la zona horaria. Puede introducir cualquier expresión de transformación válida como se define en el archivo de zona horaria presente en la ubicación de instalación.

Valor devuelto

Devuelve una marca de tiempo con el tipo de datos de zona horaria.

NULL si la entrada es un valor nulo.

Ejemplo

INPUT VALUE	RETURN VALUE
1947-08-05 10:45:00.221111000 AM, 'America/Los_Angeles'	'1947-08-05 10:45:00.221111000 AM America/Los_Angeles'
1947-08-05 10:45:00.221111000 AM, '-08:00'	'1947-08-05 10:45:00.221111000 AM -08:00'

CUME

Devuelve un total de ejecución. Un total de ejecución significa que CUME devuelve un total cada vez que añade un valor. Puede añadir una condición para filtrar filas de un conjunto de filas antes de calcular el total de ejecución.

Utilice CUME y funciones similares (como MOVINGAVG y MOVINGSUM) para simplificar la producción de informes mediante el cálculo de valores de ejecución.

Sintaxis

```
CUME( numeric_value [, filter_condition] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. Pasa los valores para los que desea calcular un total de ejecución. Se puede especificar cualquier expresión de transformación válida. Se puede crear una expresión anidada para calcular un total de ejecución basado en los resultados de la función siempre y cuando el resultado sea un valor numérico.
<i>filter_condition</i>	Opcional	Limita las filas en la búsqueda. La condición de filtro debe ser un valor numérico o dar un valor TRUE, FALSE o NULL. Se puede especificar cualquier expresión de transformación válida.

Valor de devolución

Valor numérico.

NULL si todos los valores pasados a la función son NULL, o si no se ha seleccionado ninguna fila (por ejemplo, la condición del filtro da un valor FALSE o NULL para todas las filas).

Nota: Si el valor devuelto es decimal con una precisión superior a 15, puede habilitar la alta precisión para garantizar una precisión decimal de hasta 38 dígitos.

Nulos

Si un valor es NULL, CUME devuelve el total de ejecución para la fila anterior. Sin embargo, si todos los valores en el puerto son NULL, CUME devuelve NULL.

Ejemplos

El siguiente conjunto de filas de muestra puede ser el resultado de usar la función CUME:

```
CUME ( PERSONAL_SALES )
```

PERSONAL_SALES	RETURN VALUE
40000	40000
80000	120000
40000	160000
60000	220000
NULL	220000
50000	270000

Del mismo modo, se pueden agregar valores antes de calcular un total de ejecución:

```
CUME ( CA_SALES + OR_SALES )
```

CA_SALES	OR_SALES	RETURN VALUE
40000	10000	50000
80000	50000	180000

CA_SALES	OR_SALES	RETURN VALUE
40000	2000	222000
60000	NULL	222000
NULL	NULL	222000
50000	3000	275000

DATE_COMPARE

Devuelve un entero que indica cuál de las fechas es más temprana. DATE_COMPARE devuelve un valor entero en vez de un valor de fecha.

Sintaxis

```
DATE_COMPARE( date1, date2 )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>date1</i>	Obligatorio	Tipo de datos de fecha y hora. Es la primera fecha que se va a comparar. Puede especificar cualquier expresión de transformación válida siempre que se realice una evaluación en relación con una fecha.
<i>date2</i>	Obligatorio	Tipo de datos de fecha y hora. Es la segunda fecha que se va a comparar. Puede especificar cualquier expresión de transformación válida siempre que se realice una evaluación en relación con una fecha.

Valor devuelto

Se devuelve -1 si la primera fecha es anterior.

Se devuelve 0 si las dos fechas coinciden.

Se devuelve 1 si la segunda fecha es anterior.

Se devuelve NULL si uno de los valores de fecha es NULL.

Ejemplo

La siguiente expresión compara cada fecha de los puertos DATE_PROMISED y DATE_SHIPPED, y devuelve un entero para indicar qué fecha es anterior:

```
DATE_COMPARE( DATE_PROMISED, DATE_SHIPPED )
```

DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
Jan 1 1997	Jan 13 1997	-1
Feb 1 1997	Feb 1 1997	0

DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
Dec 22 1997	Dec 15 1997	1
Feb 29 1996	Apr 12 1996	-1 <i>(Leap year)</i>
NULL	Jan 6 1997	NULL
Jan 13 1997	NULL	NULL

DATE_DIFF

Devuelve el intervalo de tiempo entre dos fechas. Se puede especificar que el formato se exprese en años, meses, días, horas, minutos, segundos, milisegundos, microsegundos o nanosegundos. El Servicio de integración de datos resta la segunda fecha de la primera fecha y devuelve la diferencia.

El Servicio de integración de datos calcula la función DATE_DIFF en función del número de meses en lugar del número de días. Calcula las diferencias de fecha para meses parciales con los días seleccionados en cada mes. Para calcular la diferencia de fecha para el mes parcial, el Servicio de integración de datos añade los días utilizados durante el mes. A continuación, divide el valor por el número total de días del mes seleccionado.

El Servicio de integración de datos presenta un valor distinto para el mismo período del año bisiesto y un año no bisiesto. La diferencia se da cuando febrero forma parte de la función DATE_DIFF. DATE_DIFF divide los días de febrero entre 29 si es un año bisiesto y 28 si no es un año bisiesto.

Por ejemplo, desea calcular el número de meses que hay desde el 13 de septiembre al 19 de febrero. En un período de año bisiesto, la función DATE_DIFF calcula el mes de febrero como 19/29 meses o 0,655 meses. En un período de año no bisiesto, la función DATE_DIFF calcula el mes de febrero como 19/28 meses o 0,678 meses. El Servicio de integración de datos, de forma similar, calcula la diferencia en las fechas para el resto de meses y la función DATE_DIFF devuelve el valor total del período especificado.

Nota: Algunas bases de datos pueden utilizar algoritmos diferentes para calcular la diferencia de fechas.

Sintaxis

```
DATE_DIFF( date1, date2, format )
```


En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>date1</i>	Obligatorio	Tipo de datos Fecha/Hora. Pasa los valores para la primera fecha que se desea comparar. Se puede especificar cualquier expresión de transformación válida.
<i>date2</i>	Obligatorio	Tipo de datos Fecha/Hora. Pasa los valores para la segunda fecha que se desea comparar. Se puede especificar cualquier expresión de transformación válida.
<i>format</i>	Obligatorio	Cadena de formato que especifica la medición de fecha u hora. Se pueden especificar años, meses, días, horas, minutos, segundos, milisegundos, microsegundos o nanosegundos. Se puede especificar solamente una parte de la fecha, como 'mm'. Delimite las cadenas de formato mediante comillas simples. La cadena de formato no distingue entre mayúsculas y minúsculas. Por ejemplo, la cadena de formato 'mm' es la misma que 'MM', 'Mm' o 'mM'.

Valor de retorno

Valor doble. Si *date1* es posterior a *date2*, el valor de retorno es un número positivo. Si *date1* es anterior a *date2*, el valor de retorno es un número negativo.

0 si las fechas son las mismas.

NULL si uno (o ambos) de los valores de fecha es NULL.

Ejemplos

Las siguientes expresiones devuelven el número de horas entre los puertos DATE_PROMISED y DATE_SHIPPED:

```
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'HH' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'HH12' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'HH24' )
```

DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
Jan 1 1997 12:00:00AM	Mar 29 1997 12:00:00PM	-2100
Mar 29 1997 12:00:00PM	Jan 1 1997 12:00:00AM	2100
NULL	Dec 10 1997 5:55:10PM	NULL
Dec 10 1997 5:55:10PM	NULL	NULL
Jun 3 1997 1:13:46PM	Aug 23 1996 4:20:16PM	6812.89166666667
Feb 19 2004 12:00:00PM	Feb 19 2005 12:00:00PM	-8784

Las siguientes expresiones devuelven el número de días entre los puertos DATE_PROMISED y DATE_SHIPPED:

```
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'D' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'DD' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'DDD' )
```

```
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'DY' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'DAY' )
```

DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
Jan 1 1997 12:00:00AM	Mar 29 1997 12:00:00PM	-87.5
Mar 29 1997 12:00:00PM	Jan 1 1997 12:00:00AM	87.5
NULL	Dec 10 1997 5:55:10PM	NULL
Dec 10 1997 5:55:10PM	NULL	NULL
Jun 3 1997 1:13:46PM	Aug 23 1996 4:20:16PM	283.870486111111
Feb 19 2004 12:00:00PM	Feb 19 2005 12:00:00PM	-366

Las siguientes expresiones devuelven el número de meses entre los puertos DATE_PROMISED y DATE_SHIPPED:

```
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'MM' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'MON' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'MONTH' )
```

DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
Jan 1 1997 12:00:00AM	Mar 29 1997 12:00:00PM	-2.91935483870968
Mar 29 1997 12:00:00PM	Jan 1 1997 12:00:00AM	2.91935483870968
NULL	Dec 10 1997 5:55:10PM	NULL
Dec 10 1997 5:55:10PM	NULL	NULL
Jun 3 1997 1:13:46PM	Aug 23 1996 4:20:16PM	9.3290162037037
Feb 19 2004 12:00:00PM	Feb 19 2005 12:00:00PM	-12

Las siguientes expresiones devuelven el número de años entre los puertos DATE_PROMISED y DATE_SHIPPED:

```
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'Y' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'YY' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'YYY' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'YYYY' )
```

DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
Jan 1 1997 12:00:00AM	Mar 29 1997 12:00:00PM	-0.24327956989247
Mar 29 1997 12:00:00PM	Jan 1 1997 12:00:00AM	0.24327956989247
NULL	Dec 10 1997 5:55:10PM	NULL
Dec 10 1997 5:55:10PM	NULL	NULL

DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
Jun 3 1997 1:13:46PM	Aug 23 1996 4:20:16PM	0.77741801697531
Feb 19 2004 12:00:00PM	Feb 19 2005 12:00:00PM	-1

Las siguientes expresiones devuelven el número de meses entre los puertos DATE_PROMISED y DATE_SHIPPED:

```
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'MM' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'MON' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'MONTH' )
```

DATE_PROMISED	DATE_SHIPPED	LEAP YEAR VALUE (in Months)	NON-LEAP YEAR VALUE (in Months)
Sept 13	Feb 19	-5.237931034	-5.260714286
NULL	Feb 19	NULL	N/A
Sept 13	NULL	NULL	N/A

DEC_BASE64

Decodifica un valor codificado de base 64 y devuelve una cadena con la representación de los datos binarios de los datos. Si codifica los datos mediante ENC_BASE64 y desea decodificar los datos utilizando DEC_BASE64, debe ejecutar la asignación utilizando el mismo modo de movimiento de datos. De lo contrario, la salida de los datos decodificados puede diferir de los datos originales.

Sintaxis

```
DEC_BASE64( value )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio/ Opcional	Descripción
<i>valor</i>	Obligatorio	Tipo de datos String. Datos que desea decodificar.

Valor devuelto

Valor binario decodificado.

NULL si la entrada es un valor nulo.

Los valores devueltos difieren si se ejecuta la asignación en modo Unicode o en modo ASCII.

DECODE

Busca en un puerto un valor que usted especifique. Si la función encuentra el valor, devuelve el valor resultado, que usted define. Puede construir un número ilimitado de búsquedas dentro de una función DECODE.

Si utiliza DECODE para buscar un valor en un puerto de cadena, puede recortar los espacios en blanco al final con la función RTRIM o incluirlos en la cadena de búsqueda.

Sintaxis

```
DECODE( value, first_search, first_result [, second_search, second_result]...[,default] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>value</i>	Obligatorio	Cualquier tipo de datos a excepción de los binarios. Transfiere los valores que desea buscar. Puede especificar cualquier expresión de transformación válida.
<i>search</i>	Obligatorio	Cualquier valor con el mismo tipo de datos que el argumento value. Transfiere los valores para los que desea efectuar la búsqueda. El valor de búsqueda debe coincidir con el argumento value. No puede buscar una parte de un valor. El valor de búsqueda distingue también entre mayúsculas y minúsculas. Por ejemplo, si desea buscar la cadena 'linterna halógena' en un determinado puerto, debe especificar 'linterna halógena' y no 'Halógena'. Si escribe 'Halógena' la búsqueda no hallará ningún valor coincidente. Puede especificar cualquier expresión de transformación válida.
<i>result</i>	Obligatorio	Cualquier tipo de datos a excepción de los binarios. El valor que desea devolver si la búsqueda encuentra un valor coincidente. Puede especificar cualquier expresión de transformación válida.
<i>default</i>	Opcional	Cualquier tipo de datos a excepción de los binarios. El valor que desea devolver si la búsqueda no encuentra un valor coincidente. Puede especificar cualquier expresión de transformación válida.

Valor de devolución

First_result si la búsqueda encuentra un valor coincidente.

El valor predeterminado si la búsqueda no encuentra un valor coincidente.

NULL si omite el argumento default y la búsqueda no encuentra un valor coincidente.

Incluso si se cumplen varias condiciones, el Servicio de integración de datos devuelve el primer resultado coincidente.

Si los datos contienen caracteres multibyte y la expresión DECODE compara los datos de la cadena, el valor de devolución depende de la página de códigos y del modo de movimiento de datos del Servicio de integración de datos.

DECODE y los tipos de datos

Cuando utiliza DECODE, el tipo de datos del valor de devolución siempre es el mismo que el del resultado pero más preciso.

Supongamos que tiene la siguiente expresión:

```
DECODE ( CONST NAME
         'Five', 5,
         'Pythagoras', 1.414213562,
         'Archimedes', 3.141592654,
         'Pi', 3.141592654 )
```

Los valores de devolución de esta expresión son 5, 1,414213562, y 3,141592654. El primer resultado es un entero, y los otros resultados, decimales. El tipo de datos decimal es más preciso que el entero. Esta expresión siempre graba el resultado como un decimal.

Cuando ejecuta una asignación con un mayor grado de precisión, y hay un resultado que como mínimo es Doble, el tipo de datos del valor de devolución es Doble.

No puede crear una función DECODE con valores de devolución de cadena y numéricos.

La expresión siguiente, por ejemplo, no es válida:

```
DECODE ( CONST NAME
         'Five', 5,
         'Pythagoras', '1.414213562',
         'Archimedes', '3.141592654',
         'Pi', 3.141592654 )
```

Cuando valida la expresión anterior, recibe el siguiente mensaje de error:

```
Function cannot resolve operands of ambiguously mismatching datatypes.
```

Ejemplos

Puede utilizar DECODE en una expresión que busque un determinado ITEM_ID y que devuelva el ITEM_NAME:

```
DECODE( ITEM_ID, 10, 'Flashlight',
        14, 'Regulator',
        20, 'Knife',
        40, 'Tank',
        'NONE' )
```

ITEM_ID	RETURN VALUE
10	Flashlight
14	Regulator
17	NONE
20	Knife
25	NONE
NULL	NONE
40	Tank

DECODE devuelve el valor predeterminado NONE para los artículos 17 y 25 porque los valores de búsqueda no coinciden con ITEM_ID. DECODE también devuelve NONE para ITEM_ID NULL.

La siguiente expresión prueba varias columnas y condiciones que se han verificado, de arriba abajo, en TRUE o FALSE.

```
DECODE( TRUE,
        Var1 = 22, 'Variable 1 was 22!',
        Var2 = 49, 'Variable 2 was 49!',
        Var1 < 23, 'Variable 1 was less than 23.',
```

```
Var2 > 30, 'Variable 2 was more than 30.',
'Variables were out of desired ranges.')
```

Var1	Var2	RETURN VALUE
21	47	Variable 1 was less than 23.
22	49	Variable 1 was 22!
23	49	Variable 2 was 49!
24	27	Variables were out of desired ranges.
25	50	Variable 2 was more than 30.

DECOMPRESS

Descomprime datos utilizando el algoritmo de compresión zlib 1.2.1. Utilice la función DECOMPRESS en datos que han sido comprimidos con la función COMPRESS o con una herramienta de compresión que utilice el algoritmo zlib 1.2.1. Si la asignación que descomprime los datos utiliza un modo de movimiento de datos diferente a la asignación que ha comprimido los datos, la salida de los datos descomprimidos puede diferir de los datos originales.

Sintaxis

```
DECOMPRESS( value, precision )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio/ Opcional	Descripción
<i>value</i>	Obligatorio	Tipo de datos Binary. Datos que quiere descomprimir.
<i>precision</i>	Opcional	Tipo de datos Integer.

Valor de retorno

Valor binario descomprimido del valor de entrada.

NULL si la entrada es un valor nulo.

ENC_BASE64

Codifica datos mediante la conversión de datos binarios a datos de cadena utilizando la codificación Multipurpose Internet Mail Extensions (MIME). Codifica los datos cuando se desea almacenar datos en una base de datos o en un archivo que no permite datos binarios. También se pueden codificar los datos para pasar datos binarios mediante transformaciones en formato de cadena. Los datos codificados son aproximadamente un 33% más largos que los datos originales. Se muestra como un conjunto de caracteres aleatorios.

Sintaxis

```
ENC_BASE64( value )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio/Opcional	Descripción
<i>value</i>	Obligatorio	Tipo de datos Binary o String. Datos que desea codificar.

Valor de retorno

Valor codificado.

NULL si la entrada es un valor nulo.

ERROR

Permite al Servicio de integración de datos omitir una fila y emitir un mensaje de error que usted puede definir. El mensaje de error se visualiza en el registro. El Servicio de integración de datos no graba estas filas omitidas en el archivo de registros rechazados.

Utilice ERROR en las transformaciones de expresión para validar los datos. En general, se utiliza ERROR en una función IIF o DECODE para establecer las reglas para omitir filas.

Utilice la función ERROR para los valores predeterminados del puerto de entrada y de salida. Puede utilizar ERROR en los puertos de entrada para evitar que los valores nulos se transfieran a una transformación.

Utilice ERROR en los puertos de salida para gestionar cualquier error de transformación, incluidas las llamadas de la función ERROR dentro de una expresión. Cuando utiliza la función ERROR en una expresión y en el valor predeterminado del puerto de salida, el Servicio de integración de datos omite la fila y registra el mensaje de error de la expresión y el mensaje de error del valor predeterminado. Si quiere asegurarse de que el Servicio de integración de datos omita las filas que generan un error, asigne ERROR como valor predeterminado.

Si utiliza un valor de salida predeterminado distinto de ERROR, el valor predeterminado reemplaza la función ERROR en una expresión. Supongamos, por ejemplo, que utiliza la función ERROR en una expresión y que asigna el valor predeterminado '1234' al puerto de salida. Cada vez que el Servicio de integración de datos encuentra la función ERROR en la expresión, reemplaza el error con el valor '1234' y transfiere '1234' a la siguiente transformación. No omite la fila y no registra ningún error en el registro.

Sintaxis

```
ERROR( string )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>string</i>	Obligatorio	Valor de cadena. El mensaje que desea visualizar cuando el servicio de integración omite una fila en función de la expresión que contiene la función ERROR. La cadena no tiene límite de caracteres.

Valor de devolución

Cadena.

Ejemplo:

El siguiente ejemplo muestra cómo se hace referencia a una asignación que calcula el salario medio de los empleados de todos los departamentos de una empresa, pero que omite los valores negativos. La siguiente expresión anida la función ERROR en una expresión IIF de modo que el Servicio de integración de datos encuentra un salario negativo en el puerto Salary, omite la fila y muestra un error:

```
IIF( SALARY < 0, ERROR ('Error. Negative salary found. Row skipped.', EMP_SALARY )
```

SALARY	RETURN VALUE
10000	10000
-15000	'Error. Negative salary found. Row skipped.'
NULL	NULL
150000	150000
1005	1005

EXP

Devuelve un valor e elevado a la potencia especificada (exponente), donde $e = 2,71828183$. Por ejemplo, EXP(2) devuelve 7,38905609893065. Puede usar esta función para analizar los datos científicos y técnicos en lugar de los datos empresariales. La función EXP tiene una correspondencia recíproca con la función LN, la cual devuelve el logaritmo natural de un valor numérico.

Sintaxis

```
EXP( exponent )
```


En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>exponent</i>	Obligatorio	Tipo de datos numérico. Valor al que se va a elevar e. Exponente de la ecuación e^{valor} . Puede especificar cualquier expresión de transformación válida.

Valor devuelto

Valor doble.

Se devuelve NULL si el valor pasado como argumento a la función es NULL.

Ejemplo

La siguiente expresión usa los valores almacenados en el puerto Numbers como valor de exponente:

```
EXP ( NUMBERS )
```

NUMBERS	RETURN VALUE
10	22026.4657948067
-2	0.135335283236613
8.55	5166.754427176
NULL	NULL

FIRST

Devuelve el primer valor encontrado en un puerto o grupo. Opcionalmente, se puede aplicar un filtro para limitar el número de filas que lee el Servicio de integración de datos. Dentro de FIRST solamente se puede anidar una función agregada adicional.

Sintaxis

```
FIRST( value [, filter_condition ] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>value</i>	Obligatorio	Cualquier tipo de datos excepto Binario. Pasa los valores para los que desea devolver el primer valor. Se puede especificar cualquier expresión de transformación válida.
<i>filter_condition</i>	Opcional	Limita las filas en la búsqueda. La condición de filtro debe ser un valor numérico o dar un valor TRUE, FALSE o NULL. Se puede especificar cualquier expresión de transformación válida.

Valor de retorno

Primer valor de un grupo.

NULL si todos los valores pasados a la función son NULL, o si no se ha seleccionado ninguna fila (por ejemplo, la condición del filtro da un valor FALSE o NULL para todas las filas).

Nulos

Si un valor es NULL, FIRST omite la fila. Sin embargo, si todos los valores pasados desde el puerto son NULL, FIRST devuelve NULL.

Agrupar por

FIRST agrupa valores por grupos según los puertos definidos en la transformación, devolviendo un resultado para cada grupo.

Si no hay un grupo por puerto, FIRST trata todas las filas como un solo grupo, devolviendo un valor.

Ejemplos

La siguiente expresión devuelve el primer valor en el puerto ITEM_NAME que tiene un precio mayor que \$10.00:

```
FIRST( ITEM_NAME, ITEM_PRICE > 10 )
```

ITEM_NAME	ITEM_PRICE
Flashlight	35.00
Navigation Compass	8.05
Regulator System	150.00
Flashlight	29.00
Depth/Pressure Gauge	88.00
Flashlight	31.00

RETURN VALUE: Flashlight

La siguiente expresión devuelve el primer valor en el puerto ITEM_NAME que tiene un precio mayor que \$40.00:

```
FIRST( ITEM_NAME, ITEM_PRICE > 40 )
```

ITEM_NAME	ITEM_PRICE
Flashlight	35.00
Navigation Compass	8.05
Regulator System	150.00
Flashlight	29.00
Depth/Pressure Gauge	88.00

ITEM_NAME	ITEM_PRICE
Flashlight	31.00
RETURN VALUE: Regulator System	

FLOOR

Devuelve el mayor entero que sea menor o igual al valor numérico que se pasa a esta función. Por ejemplo, si se pasa 3,14 a FLOOR, la función devuelve 3. Si se pasa 3,98 a FLOOR, la función devuelve 3. Del mismo modo, si se pasa -3,17 a FLOOR, la función devuelve -4.

Sintaxis

```
FLOOR( numeric_value )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de dato numérico. Puede introducir cualquier expresión de transformación válida, ya que devuelve datos numéricos.

Valor de retorno

Entero si se pasa un valor numérico con precisión declarada entre 0 y 28.

Doble si se pasa un valor numérico con precisión declarada mayor de 28.

NULL si el valor pasado a la función es NULL.

Ejemplo

La siguiente expresión devuelve el mayor entero menor o igual a los valores del puerto PRICE:

```
FLOOR( PRICE )
```

PRICE	RETURN VALUE
39.79	39
125.12	125
74.24	74
NULL	NULL
-100.99	-101

Sugerencia: Se pueden realizar operaciones aritméticas de los valores que se pasen a FLOOR. Por ejemplo, para multiplicar un valor numérico por 10 y calcular el entero más grande que sea menor que el producto, podría escribir la función de la siguiente manera:

```
FLOOR( UNIT_PRICE * 10 )
```

FV

Devuelve el valor futuro de una inversión, en la que usted realiza pagos periódicos o constantes y la inversión gana un tipo de interés constante.

Sintaxis

```
FV( rate, terms, payment [, present value, type] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>rate</i>	Obligatorio	Númerico. tipo de interés ganado en cada período. Expresado como un número decimal. Divida el índice de porcentaje entre 100 para expresarlo como un número decimal. Debe ser mayor o igual a 0.
<i>terms</i>	Obligatorio	Númerico. Número de períodos o pagos. Debe ser mayor que 0.
<i>payment</i>	Obligatorio	Númerico. Importe del pago debido por período. Debe ser un número negativo.
<i>present value</i>	Opcional	Númerico. Valor actual de la inversión. Si omite este argumento, FV utiliza 0.
<i>type</i>	Opcional	Entero. Programación del pago. Introduzca 1 si el pago es al inicio del período. Introduzca 0 si el pago es al final del período. El valor predeterminado es 0. Si introduce un valor distinto de 0 ó 1, el Servicio de integración de datos trata el valor como 1.

Valor de retorno

Númerico.

Ejemplo

Usted deposita 2.000 \$ en una cuenta con un rendimiento del 9% de interés anual compuesto mensualmente (interés mensual del 9%/12, ó 0,75%). Piensa depositar 250 \$ a principios de cada mes durante los próximos 12 meses. La siguiente expresión devuelve 5.337,96 \$ como el saldo de la cuenta al final del período de 12 meses:

```
FV(0.0075, 12, -250, -2000, TRUE)
```

Notas

Para calcular la tasa de los intereses devengados en cada período, se divide la tasa anual por el número de pagos efectuados en un año. El valor del pago y el valor actual son negativos porque se trata de cantidades que usted paga.

GET_DATE_PART

Devuelve la parte especificada de una fecha como valor entero. Por tanto, si se crea una expresión que devuelva la porción del mes de la fecha y pasa una fecha como 1 de abril 1997 00:00:00, GET_DATE_PART devuelve 4.

Sintaxis

```
GET_DATE_PART( date, format )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>date</i>	Obligatorio	Tipo de datos Fecha/Hora. Se puede especificar cualquier expresión de transformación válida.
<i>format</i>	Obligatorio	Una cadena de formato que especifica la porción del valor de fecha que se desea devolver. Delimite las cadenas de formato mediante comillas simples; por ejemplo, 'mm'. La cadena de formato no distingue entre mayúsculas y minúsculas. Cada cadena de formato devuelve la parte completa de la fecha según el formato de fecha especificado en la asignación. Por ejemplo, si se pasa la fecha 1 de abril 1997 a GET_DATE_PART, todas las cadenas de formato 'Y', 'YY', 'YYY' o 'YYYY' devuelven 1997.

Valor de retorno

Entero que representa la parte especificada de la fecha.

NULL si el valor pasado a la función es NULL.

Ejemplos

Las siguientes expresiones devuelven la hora para cada fecha en el puerto DATE_SHIPPED. 12:00:00AM devuelve 0 porque el formato de fecha predeterminado está basado en el intervalo de 24 horas:

```
GET_DATE_PART( DATE_SHIPPED, 'HH' )
GET_DATE_PART( DATE_SHIPPED, 'HH12' )
GET_DATE_PART( DATE_SHIPPED, 'HH24' )
```

DATE_SHIPPED	RETURN VALUE
Mar 13 1997 12:00:00AM	0
Sep 2 1997 2:00:01AM	2
Aug 22 1997 12:00:00PM	12
June 3 1997 11:30:44PM	23
NULL	NULL

Las siguientes expresiones devuelven el día para cada fecha en el puerto DATE_SHIPPED.

```
GET_DATE_PART( DATE_SHIPPED, 'D' )
GET_DATE_PART( DATE_SHIPPED, 'DD' )
GET_DATE_PART( DATE_SHIPPED, 'DDD' )
```

```
GET_DATE_PART( DATE_SHIPPED, 'DY' )
GET_DATE_PART( DATE_SHIPPED, 'DAY' )
```

DATE_SHIPPED	RETURN VALUE
Mar 13 1997 12:00:00AM	13
June 3 1997 11:30:44PM	3
Aug 22 1997 12:00:00PM	22
NULL	NULL

Las siguientes expresiones devuelven el mes para cada fecha en el puerto DATE_SHIPPED.

```
GET_DATE_PART( DATE_SHIPPED, 'MM' )
GET_DATE_PART( DATE_SHIPPED, 'MON' )
GET_DATE_PART( DATE_SHIPPED, 'MONTH' )
```

DATE_SHIPPED	RETURN VALUE
Mar 13 1997 12:00:00AM	3
June 3 1997 11:30:44PM	6
NULL	NULL

Las siguientes expresiones devuelven el año para cada fecha en el puerto DATE_SHIPPED.

```
GET_DATE_PART( DATE_SHIPPED, 'Y' )
GET_DATE_PART( DATE_SHIPPED, 'YY' )
GET_DATE_PART( DATE_SHIPPED, 'YYY' )
GET_DATE_PART( DATE_SHIPPED, 'YYYY' )
```

DATE_SHIPPED	RETURN VALUE
Mar 13 1997 12:00:00AM	1997
June 3 1997 11:30:44PM	1997
NULL	NULL

GET_TIMEZONE

Devuelve el valor de zona horaria de una marca de tiempo dada con la columna de zona horaria.

Por ejemplo:

```
String TimeZone = GET_TIMEZONE (timestampWithTZ);
```

El puerto de salida debe ser de tipo de datos de cadena para las expresiones GET_TIMEZONE.

Sintaxis

```
GET_TIMEZONE (timestamp_with_timezone_value)
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>timestamp_with_timezone_value</i>	Obligatorio	Debe ser una marca de tiempo con tipo de datos de zona horaria. Se puede especificar cualquier expresión de transformación válida.

Valor devuelto

Devuelve un tipo de datos de cadena que contiene el nombre de la región de zona horaria o el ajuste de la zona horaria.

NULL si la entrada es un valor nulo.

Ejemplo

INPUT VALUE	RETURN VALUE
'1947-08-05 10:45:00.221111111 AM America/Los_Angeles'	'AMERICA/LOS_ANGELES'
'1947-08-05 10:45:00.221111111 AM -08:00'	'-08:00'

GET_TIMESTAMP

Devuelve el valor de fecha y hora para un tipo de entrada `timestampWithTZ`. La fecha y hora devueltas estarán en la zona horaria solicitada, que se puede proporcionar como segundo argumento. Si el valor de la zona horaria no se especifica en el segundo argumento, la función devuelve la parte de la marca de tiempo del valor `timestampWithTZ` de entrada.

Por ejemplo:

```
GET_TIMESTAMP(Timestamp with Time Zone, "+08:30")
```

El primer argumento, marca de tiempo con zona horaria tiene (+05:30) como el valor de la zona horaria. La función devuelve la marca de tiempo en la zona horaria especificada como segundo argumento (+08:30).

El puerto de salida debe ser Date/Time para las expresiones `GET_TIMESTAMP`.

Sintaxis

```
GET_TIMESTAMP (timestamp_with_timezone_value, [timezone_value])
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>timestamp_with_timezone_value</i>	Obligatorio	Debe ser una marca de tiempo con tipo de datos de zona horaria. Se puede especificar cualquier expresión de transformación válida.
<i>timezone_value</i>	Opcional	Debe ser un tipo de datos de cadena. La cadena debe ser una cadena de caracteres. Pasa los valores que se desean mostrar para la zona horaria y en base a los cuales la función puede devolver la marca de tiempo. Se puede especificar cualquier expresión de transformación válida. Si no se especifica la zona horaria, la función devuelve la parte de la marca de tiempo del primer argumento.

Valor devuelto

Devuelve el valor de marca de tiempo en la región o el ajuste de zona horaria especificado.

Si no se pasa el valor de zona horaria, la función devuelve la parte de la marca de tiempo del primer argumento.

NULL si la entrada es un valor nulo.

Ejemplo

INPUT VALUE	RETURN VALUE
'1996-01-05 10:45:00.221111111 AM America/Los_Angeles', '+05:30'	Returns the timestamp value in time zone offset of '+05:30': '1996-01-06 12:15:00.221111111 AM'
'1996-01-05 10:45:00.221111111 AM America/Los_Angeles', 'GMT'	Returns the timestamp value in the 'GMT' time zone: '1996-01-05 06:45:00.221111111 PM'
'1996-01-05 10:45:00.221111111 AM America/Los_Angeles'	As the time zone value is not passed as the second input parameter, the timestamp is returned: '1996-01-05 10:45:00.221111111 AM'

GREATEST

Devuelve el valor mayor de una lista de valores de entrada. Utilice esta función para devolver la cadena, fecha o número mayor. De forma predeterminada, la coincidencia distingue mayúsculas de minúsculas.

Sintaxis

```
GREATEST( value1, [value2, ..., valueN,] CaseFlag )
```


En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>valor</i>	Obligatorio	Cualquier tipo de datos excepto Binario. El tipo de datos debe ser compatible con otros valores. Valor que desea comparar con otros valores. Debe introducir por lo menos un argumento de valor. Si el valor es numérico y otros valores de entrada son numéricos, todos los valores usan la mayor precisión posible. Por ejemplo, si el tipo de datos de algunos valores es Entero y el de otros es Doble, el Servicio de integración de datos convierte los valores a Doble.
<i>CaseFlag</i>	Opcional	Debe ser un entero. Especifique un valor cuando el argumento de valor de entrada sea un valor de cadena. Determina si en los argumentos de esta función se distingue entre mayúsculas y minúsculas. Se puede especificar cualquier expresión de transformación válida. Cuando CaseFlag es un número distinto de 0, la función distingue entre mayúsculas y minúsculas. Cuando CaseFlag es 0, la función no distingue entre mayúsculas y minúsculas. El valor predeterminado distingue mayúsculas de minúsculas.

Valor devuelto

value1 si es el mayor de los valores de entrada, *value2* si es el mayor de los valores de entrada, y así sucesivamente.

NULL si alguno de los argumentos es nulo.

Ejemplo

La expresión siguiente devuelve la cantidad mayor de artículos pedidos:

```
GREATEST( QUANTITY1, QUANTITY2, QUANTITY3 )
```

QUANTITY1	QUANTITY2	QUANTITY3	RETURN VALUE
150	756	27	756
			NULL
5000	97	17	5000
120	1724	965	1724

IIF

Devuelve uno de los dos valores que especifique, según el resultado de una condición.

Sintaxis

```
IIF( condition, value1 [,value2] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>condition</i>	Obligatorio	La condición que desea verificar. Puede especificar cualquier expresión de transformación válida que se verifique en TRUE o FALSE.
<i>value1</i>	Obligatorio	Cualquier tipo de datos a excepción de los binarios. El valor que desea devolver si la condición es TRUE. El valor de devolución siempre es el tipo de datos que se ha especificado con este argumento. Puede especificar cualquier expresión de transformación válida, incluida otra expresión IIF.
<i>value2</i>	Opcional	Cualquier tipo de datos a excepción de los binarios. El valor que desea devolver si la condición es FALSE. Puede especificar cualquier expresión de transformación válida, incluida otra expresión IIF.

A diferencia de lo que ocurre con las funciones condicionales de algunos sistemas, la condición FALSE (*value2*) no es necesaria en la función IIF. Si omite *value2*, la función devuelve lo siguiente cuando la condición es FALSE:

- 0 si *value1* es un tipo de datos numérico.
- Cadena vacía si *value1* es un tipo de datos String.
- NULL si *value1* es un tipo de datos de fecha y hora.

Por ejemplo, la siguiente expresión no incluye una condición FALSE y *value1* es un tipo de datos String, así que el Servicio de integración de datos devuelve una cadena vacía para cada fila que se verifica en FALSE:

```
IIF( SALES > 100, EMP_NAME )
```

SALES	EMP_NAME	RETURN VALUE
150	John Smith	John Smith
50	Pierre Bleu	' ' (empty string)
120	Sally Green	Sally Green
NULL	Greg Jones	' ' (empty string)

Valor de devolución

value1 si la condición es TRUE.

value2 si la condición es FALSE.

La siguiente expresión incluye, por ejemplo, la condición FALSE NULL, así que el Servicio de integración de datos devuelve NULL para cada fila que se verifica en FALSE:

```
IIF( SALES > 100, EMP_NAME, NULL )
```

SALES	EMP_NAME	RETURN VALUE
150	John Smith	John Smith
50	Pierre Bleu	NULL

SALES	EMP_NAME	RETURN VALUE
120	Sally Green	Sally Green
NULL	Greg Jones	NULL

Si los datos contienen caracteres multibyte y el argumento condition compara datos de cadena, el valor de devolución depende de la página de códigos y del modo de movimiento de datos del Servicio de integración de datos.

IIF y los tipos de datos

Cuando utiliza IIF, el tipo de datos del valor de devolución es el mismo que el del resultado pero más preciso.

Supongamos que tiene la siguiente expresión:

```
IIF( SALES < 100, 1, .3333 )
```

El resultado TRUE (1) es un entero y el resultado FALSE (0,3333), un decimal. El tipo de datos decimal es más preciso que Integer, así que el tipo de datos del valor de devolución siempre es un decimal.

Cuando ejecuta una asignación con un mayor grado de precisión y como mínimo un resultado es Double, el tipo de datos del valor de devolución es Double.

Tipos de datos complejos e IIF

Se puede utilizar IIF para devolver una matriz o una estructura, o elementos de la matriz o estructura.

Supongamos que tiene la siguiente matriz:

```
names = ['John', 'Kevin', 'Laura']
```

Puede utilizar la siguiente expresión para devolver uno de los valores de la matriz:

```
IIF( SIZE(names) > 2, names[2], names[0] )
```

RETURN VALUE: 'Laura'

Usos especiales de IIF

Utilice instrucciones IIF anidadas para probar varias condiciones. El siguiente ejemplo prueba varias condiciones y devuelve 0 si las ventas son 0 o negativas:

```
IIF( SALES > 0, IIF( SALES < 50, SALARY1, IIF( SALES < 100, SALARY2, IIF( SALES < 200, SALARY3, BONUS))), 0 )
```

Si añade algunos comentarios, la lógica de esta instrucción será más comprensible:

```
IIF( SALES > 0,
--then test to see if sales is between 1 and 49:
  IIF( SALES < 50,

--then return SALARY1
    SALARY1,

--else test to see if sales is between 50 and 99:
    IIF( SALES < 100,

--then return
      SALARY2,

--else test to see if sales is between 100 and 199:
      IIF( SALES < 200,

--then return
        SALARY3,
```

```

--else for sales over 199, return
    BONUS)
    )
),
--else for sales less than or equal to zero, return
    0)

```

Utilice IIF en las estrategias de actualización. Por ejemplo:

```
IIF( ISNULL( ITEM_NAME ), DD_REJECT, DD_INSERT)
```

Alternativa a IIF

Utilice **“DECODE” en la página 92** en lugar de IIF en muchos casos. DECODE puede mejorar la legibilidad. El siguiente esquema muestra cómo puede utilizar DECODE en lugar de IIF con el primer ejemplo de la sección anterior:

```

DECODE( TRUE,
    SALES > 0 and SALES < 50, SALARY1,
    SALES > 49 AND SALES < 100, SALARY2,
    SALES > 99 AND SALES < 200, SALARY3,
    SALES > 199, BONUS)

```

Puede utilizar una transformación de filtro en lugar de IIF para maximizar el rendimiento.

IN

Hace coincidir los datos de entrada con una lista de valores. De forma predeterminada, la coincidencia distingue mayúsculas de minúsculas.

Sintaxis

```
IN( valueToSearch, value1, [value2, ..., valueN,] CaseFlag )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>valueToSearch</i>	Obligatorio	Puede ser un valor de cadena, fecha o numérico. Valor de entrada del que desea buscar una coincidencia dentro de una lista de valores separados por comas.
<i>valor</i>	Obligatorio	Puede ser un valor de cadena, fecha o numérico según el tipo especificado para el argumento valueToSearch. Lista de valores separados por comas que desea buscar. Los valores pueden ser los puertos de una transformación. No hay un número máximo de valores que se pueden enumerar.
<i>CaseFlag</i>	Opcional	Debe ser un entero. Especifique un valor cuando el argumento valueToSearch sea un valor de cadena. Determina si en los argumentos de esta función se distingue entre mayúsculas y minúsculas. Se puede especificar cualquier expresión de transformación válida. Cuando CaseFlag es un número distinto de 0, la función distingue entre mayúsculas y minúsculas. Cuando CaseFlag es 0, la función no distingue entre mayúsculas y minúsculas. El valor predeterminado distingue mayúsculas de minúsculas.

Valor devuelto

TRUE (1) si el valor de la entrada coincide con la lista de valores.

FALSE (0) si el valor de la entrada no coincide con la lista de valores.

NULL si la entrada es un valor nulo.

Ejemplo

La siguiente expresión determina si el valor de entrada es safety knife, chisel point knife o medium titanium knife. En los valores de entrada, las mayúsculas y las minúsculas no tienen que coincidir con las de los valores de la lista separada por comas:

```
IN( ITEM_NAME, 'Chisel Point Knife', 'Medium Titanium Knife', 'Safety Knife', 0 )
```

ITEM_NAME	RETURN VALUE
Stabilizing Vest	0 (FALSE)
Safety knife	1 (TRUE)
Medium Titanium knife	1 (TRUE)
	NULL

INDEXOF

Busca el índice de un valor de entre una lista de valores. De forma predeterminada, la coincidencia distingue mayúsculas de minúsculas.

Sintaxis

```
INDEXOF( valueToSearch, string1 [, string2, ..., stringN,] [CaseFlag] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>valueToSearch</i>	Obligatorio	Tipo de datos String. Valor que desea buscar en la lista de cadenas.
<i>string</i>	Obligatorio	Tipo de datos String. Lista de valores separados por comas en la que desea buscar. Los valores pueden estar en formato de cadena. No hay un número máximo de valores que se pueden enumerar. El valor distingue mayúsculas de minúsculas a menos que establezca CaseFlag en 0.
<i>CaseFlag</i>	Opcional	Debe ser un entero. Especifique un valor cuando el argumento valueToSearch sea un valor de cadena. Determina si en los argumentos de esta función se distingue entre mayúsculas y minúsculas. Se puede especificar cualquier expresión de transformación válida. Cuando CaseFlag es un número distinto de 0, la función distingue entre mayúsculas y minúsculas. Cuando CaseFlag es 0, la función no distingue entre mayúsculas y minúsculas.

Valor devuelto

1 si el valor de entrada coincide con *string1*, 2 si el valor de entrada coincide con *string2*, y así sucesivamente.

0 si el valor de entrada no se encuentra.

NULL si la entrada es un valor nulo.

Ejemplo

La siguiente expresión determina si los valores del puerto ITEM_NAME coinciden con la primera, con la segunda o con la tercera cadena:

```
INDEXOF( ITEM_NAME, 'diving hood', 'flashlight', 'safety knife')
```

ITEM_NAME	RETURN VALUE
Safety Knife	0
diving hood	1
Compass	0
safety knife	3
flashlight	2

Safety Knife devuelve un valor de 0, ya que las mayúsculas y minúsculas no coinciden con las del valor de entrada.

INITCAP

Se usan mayúsculas para la primera letra de cada palabra de una cadena y se usan minúsculas para el resto de las letras. Las palabras se delimitan con espacios en blanco (espacio en blanco, avance de página, nueva línea, retorno de carro, tabulación o tabulación vertical) y caracteres no alfanuméricos. Por ejemplo, si pasa la cadena '...THOMAS', la función devuelve Thomas.

Sintaxis

```
INITCAP( string )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>string</i>	Obligatorio	Cualquier tipo de datos a excepción de los binarios. Se puede especificar cualquier expresión de transformación válida.

Valor de devolución

Cadena. Si los datos contienen caracteres multibyte, el valor de devolución depende de la página de códigos y el modo de movimiento de datos del Servicio de integración de datos.

Se devuelve NULL si el valor pasado a la función es NULL.

Ejemplo

La siguiente expresión usa mayúsculas para todos los nombres del puerto FIRST_NAME:

```
INITCAP( FIRST_NAME )
```

FIRST_NAME	RETURN VALUE
ramona	Ramona
18-albert	18-Albert
NULL	NULL
?!SAM	?!Sam
THOMAS	Thomas
PierRe	Pierre

INSTR

Devuelve la posición de un conjunto de caracteres en una cadena, contando de izquierda a derecha.

Sintaxis

```
INSTR( string, search_value [,start [,occurrence [,comparison_type ]]] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio/ opcional	Descripción
<i>string</i>	Obligatorio	La cadena debe ser una cadena de caracteres. Transfiere el valor que desea evaluar. Puede especificar cualquier expresión de transformación válida. Los resultados de la expresión deben ser una cadena de caracteres. De lo contrario, INSTR convierte el valor en una cadena antes de evaluarla.
<i>search_value</i>	Obligatorio	Cualquier valor. El valor de búsqueda distingue entre mayúsculas y minúsculas. El conjunto de caracteres que desea buscar. El argumento search_value debe coincidir con una parte de la cadena. Por ejemplo, si escribe INSTR('Alfred Pope', 'Alfred Smith'), la función devuelve 0. Puede especificar cualquier expresión de transformación válida. Si desea buscar una cadena de caracteres, encierre entre comillas simples los caracteres que desea buscar, por ejemplo 'abc'.

Argumento	Obligatorio/ opcional	Descripción
<i>start</i>	Opcional	<p>Debe ser un entero. La posición en la cadena donde desea iniciar la búsqueda. Puede especificar cualquier expresión de transformación válida.</p> <p>El valor predeterminado es 1, lo que significa que INSTR inicia la búsqueda en el primer carácter de la cadena.</p> <p>Si la posición de inicio es 0, INSTR busca a partir del primer carácter de la cadena. Si la posición de inicio es un número positivo, INSTR ubica la posición de inicio contando desde el principio de la cadena. Si la posición de inicio es un número negativo, INSTR ubica la posición de inicio contando desde el final de la cadena. Si omite este argumento, la función utiliza el valor predeterminado 1.</p>
<i>occurrence</i>	Opcional	<p>Un entero positivo mayor que 0. Puede especificar cualquier expresión de transformación válida. Si el valor de búsqueda aparece más de una vez en la cadena, puede especificar la ocurrencia que desea buscar. Por ejemplo, debe indicar 2 si desea buscar la segunda ocurrencia a partir de la posición inicial.</p> <p>Si omite este argumento, la función utiliza el valor predeterminado 1, lo que significa que INSTR busca la primera ocurrencia del valor de búsqueda. Si transfiere un decimal, el Servicio de integración de datos lo redondea al valor entero más próximo. Si transfiere un entero negativo o 0, la sesión genera un error.</p>
<i>comparison_type</i>	Opcional	<p>El tipo de comparación de la cadena, ya sea lingüística o binaria, cuando el Servicio de integración de datos se ejecuta en modo Unicode. Cuando el Servicio de integración de datos se ejecuta en modo ASCII, el tipo de comparación siempre es binaria.</p> <p>Las comparaciones lingüísticas tienen en cuenta las reglas de agrupación específicas de cada idioma, mientras que las comparaciones binarias efectúan comparaciones por bit. Por ejemplo, el carácter ß del alemán coincide con la cadena "ss" en una comparación lingüística, pero no en una comparación binaria. Las comparaciones binarias son más rápidas que las comparaciones lingüísticas.</p> <p>Debe ser un entero, ya sea 0 ó 1:</p> <ul style="list-style-type: none"> - 0: INSTR efectúa una comparación lingüística de las cadenas. - 1: INSTR efectúa una comparación binaria de las cadenas. <p>El valor predeterminado es 0.</p>

Valor de devolución

Un entero si la búsqueda es satisfactoria. El entero representa la posición del primer carácter en el argumento *search_value*, contando de izquierda a derecha.

0 si la búsqueda no es satisfactoria.

NULL si el valor que se ha transferido a la función es NULL.

Ejemplos

La siguiente expresión devuelve la posición de la primera ocurrencia de la letra 'a', empezando por el principio de cada nombre de la compañía. Dado que el argumento *search_value* distingue entre mayúsculas y minúsculas, la función omite la letra 'A' de 'Blue Fin Aqua Center', y devuelve la posición para la 'a' de 'Aqua':

```
INSTR( COMPANY, 'a' )
```

COMPANY	RETURN VALUE
Blue Fin Aqua Center	13
Maco Shark Shop	2
Scuba Gear	5
Frank's Dive Shop	3
VIP Diving Club	0

La siguiente expresión devuelve la posición de la segunda ocurrencia de la letra 'a', empezando por el principio de cada nombre de la compañía. Dado que el argumento *search_value* distingue entre mayúsculas y minúsculas, la función omite la letra 'A' de 'Blue Fin Aqua Center', y devuelve 0:

```
INSTR( COMPANY, 'a', 1, 2 )
```

COMPANY	RETURN VALUE
Blue Fin Aqua Center	0
Maco Shark Shop	8
Scuba Gear	9
Frank's Dive Shop	0
VIP Diving Club	0

La siguiente expresión devuelve la posición de la segunda ocurrencia de la letra 'a' de cada nombre de compañía, empezando por el último carácter del nombre de la compañía. Dado que el argumento *search_value* distingue entre mayúsculas y minúsculas, la función omite la letra 'A' de 'Blue Fin Aqua Center', y devuelve 0:

```
INSTR( COMPANY, 'a', -1, 2 )
```

COMPANY	RETURN VALUE
Blue Fin Aqua Center	0
Maco Shark Shop	2
Scuba Gear	5

COMPANY	RETURN VALUE
Frank's Dive Shop	0
VIP Diving Club	0

La siguiente expresión devuelve la posición del primer carácter de la cadena 'Blue Fin Aqua Center' (empezando por el último carácter del nombre de la compañía):

```
INSTR( COMPANY, 'Blue Fin Aqua Center', -1, 1 )
```

COMPANY	RETURN VALUE
Blue Fin Aqua Center	1
Maco Shark Shop	0
Scuba Gear	0
Frank's Dive Shop	0
VIP Diving Club	0

Uso de INSTR anidadas

Puede anidar la función INSTR con otras funciones para llevar a cabo tareas más complejas.

La siguiente expresión evalúa una cadena empezando desde el final. La expresión busca el último espacio (el más a la derecha) de la cadena y devuelve todos los caracteres a su izquierda:

```
SUBSTR( CUST_NAME, 1, INSTR( CUST_NAME, ' ', -1, 1 ) )
```

CUST_NAME	RETURN VALUE
PATRICIA JONES	PATRICIA
MARY ELLEN SHAH	MARY ELLEN

La siguiente expresión quita el carácter '#' de una cadena:

```
SUBSTR( CUST_ID, 1, INSTR(CUST_ID, '#')-1 ) || SUBSTR( CUST_ID, INSTR(CUST_ID, '#')+1 )
```

CUST_ID	RETURN VALUE
ID#33	ID33
#A3577	A3577
SS #712403399	SS 712403399

ISNULL

Devuelve si un valor es NULL. ISNULL evalúa una cadena vacía como FALSE.

Nota: Para comprobar las cadenas vacías, use LENGTH.

Sintaxis

```
ISNULL( value )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
valor	Obligatorio	Cualquier tipo de datos a excepción de los binarios. Pasa las filas que se van a evaluar. Se puede especificar cualquier expresión de transformación válida.

Valor devuelto

Se devuelve TRUE (1) si el valor es NULL.

Se devuelve FALSE (0) si el valor no es NULL.

Ejemplo

La siguiente expresión comprueba los valores nulos de la tabla de elementos:

```
ISNULL( ITEM_NAME )
```

ITEM_NAME	RETURN VALUE
Flashlight	0 (FALSE)
NULL	1 (TRUE)
Regulator system	0 (FALSE)
' '	0 (FALSE) <i>Empty string is not NULL</i>

Tipos de datos complejos e ISNULL

Puede utilizar ISNULL para comprobar si una matriz o una estructura tiene un valor nulo.

Las siguientes expresiones verifican los valores NULL en los siguientes tipos de datos complejos:

Complex Data Type	Input Value	RETURN VALUE
NULL_array = NULL	ISNULL(NULL_array)	1 (TRUE)
NULL_struct = NULL	ISNULL(NULL_struct)	1 (TRUE)
num_array = [1, 2, 3]	ISNULL(num_array)	0 (FALSE)
num_array = [1, NULL, 3]	ISNULL(num_array)	0 (FALSE)
num_struct{ number: int rank: int }	ISNULL(num_struct)	0 (FALSE)

IS_DATE

Devuelve si el valor de una cadena es una fecha válida. Una fecha válida es una cadena expresada en la parte de fecha del formato de fecha especificado en la sesión. Si la cadena que desea probar no está en este formato de fecha, utilice la cadena de formato TO_DATE para especificar el formato de fecha. Si las cadenas transferidas a IS_DATE no coinciden con la cadena de formato especificada, la función devuelve FALSE (0). Si las cadenas coinciden con la cadena de formato, la función devuelve TRUE (1).

IS_DATE verifica las cadenas y devuelve un valor entero.

El puerto de salida de una expresión IS_DATE debe ser un tipo de datos String o numérico.

Puede utilizar IS_DATE para probar o filtrar datos en un archivo sin formato antes de grabarlos en el destino.

Con IS_DATE, utilice la cadena de formato RR en lugar de YY. En la mayoría de casos, las dos cadenas de formato devuelven los mismos valores, pero hay algunos casos únicos en los que YY devuelve resultados incorrectos. Por ejemplo, la expresión IS_DATE('02/29/00', 'YY') se calcula de forma interna como IS_DATE(02/29/1900 00:00:00), que devuelve un valor FALSE. El Servicio de integración de datos, sin embargo, calcula la expresión IS_DATE('02/29/00', 'RR') como IS_DATE(02/29/2000 00:00:00), que devuelve el valor TRUE. En el primer caso, el año 1900 no es un año bisiesto, así que no hay 29 de febrero.

Nota: IS_DATE utiliza las mismas cadenas de formato que TO_DATE.

Sintaxis

```
IS_DATE( value [,format] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>value</i>	Obligatorio	Debe ser un tipo de datos String. Transfiere las filas que desea verificar. Puede especificar cualquier expresión de transformación válida.
<i>format</i>	Opcional	Especifique una cadena de formato TO_DATE válida. La cadena de formato debe coincidir con las partes del argumento <i>string</i> . Si transfiere por ejemplo, 'Mar 15 1997 12:43:10AM', debe utilizar la cadena de formato 'MON DD YYYY HH12:MI:SSAM'. Si omite la cadena de formato, el valor de la cadena debe estar en el formato de fecha especificado en la configuración de la asignación.

Valor de devolución

TRUE (1) si la fila es una fecha válida.

FALSE (0) si la fila no es una fecha válida.

NULL si un valor en la expresión es NULL o si la cadena de formato es NULL.

advertencia: El formato de la cadena IS_DATE debe coincidir con la cadena de formato, incluidos los separadores de fecha. Si no coincide, el Servicio de integración de datos puede devolver valores inexactos u omitir el registro.

Ejemplos

La siguiente expresión verifica si hay fechas válidas en el puerto INVOICE_DATE:

```
IS_DATE( INVOICE_DATE )
```

Esta expresión devuelve datos similares a los siguientes:

INVOICE_DATE	RETURN VALUE
NULL	NULL
'180'	0 (FALSE)
'04/01/98'	0 (FALSE)
'04/01/1998 00:12:15.7008'	1 (TRUE)
'02/31/1998 12:13:55.9204'	0 (FALSE) (February does not have 31 days)
'John Smith'	0 (FALSE)

La siguiente expresión IS_DATE especifica una cadena de formato de 'YYYY/MM/DD':

```
IS_DATE( INVOICE_DATE, 'YYYY/MM/DD' )
```

Si el valor de la cadena no coincide con este formato, IS_DATE devuelve FALSE:

INVOICE_DATE	RETURN VALUE
NULL	NULL
'180'	0 (FALSE)
'04/01/98'	0 (FALSE)
'1998/01/12'	1 (TRUE)
'1998/11/21 00:00:13'	0 (FALSE)
'1998/02/31'	0 (FALSE) (February does not have 31 days)
'John Smith'	0 (FALSE)

El siguiente ejemplo muestra cómo se debe utilizar IS_DATE para probar los datos antes de utilizar TO_DATE para convertir las cadenas en fechas. Esta expresión verifica los valores del puerto INVOICE_DATE y convierte las fechas válidas en un valor de fecha. Si el valor no es una fecha válida, el Servicio de integración de datos devuelve ERROR y omite la fila.

Este ejemplo devuelve un valor de fecha/hora. Por tanto, el puerto de salida para la expresión debe ser Fecha/Hora:

```
IIF( IS_DATE ( INVOICE_DATE, 'YYYY/MM/DD' ), TO_DATE( INVOICE_DATE ), ERROR('Not a valid date' ) )
```

INVOICE_DATE	RETURN VALUE
NULL	NULL
'180'	'Not a valid date'
'04/01/98'	'Not a valid date'
'1998/01/12'	1998/01/12

INVOICE_DATE	RETURN VALUE
'1998/11/21 00:00:13'	'Not a valid date'
'1998/02/31'	'Not a valid date'
'John Smith'	'Not a valid date'

IS_NUMBER

Devuelve si una cadena es un número válido. Un número válido se compone de las siguientes partes:

- Espacio opcional antes del número
- Signo opcional (+/-)
- Uno o más dígitos con un punto decimal opcional
- Notación científica opcional, como la letra 'e' o 'E' (y la letra 'd' o 'D' en Windows), seguido por un signo opcional (+/-), seguido de uno o más dígitos
- Espacio en blanco opcional después del número

Los números siguientes son válidos:

```
' 100 '
' +100 '
'-100'
'-3.45e+32'
'+3.45E-32'
'+3.45d+32' (Windows only)
'+3.45D-32' (Windows only)
'.6804'
```

El puerto de salida para una expresión IS_NUMBER debe ser un tipo de datos String o numérico.

Puede usar IS_NUMBER para probar o filtrar los datos de un archivo sin formato antes de escribirlos en un destino.

Sintaxis

```
IS_NUMBER( value )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>value</i>	Obligatorio	Debe ser un tipo de datos String. Pasa las filas que desea evaluar. Se puede introducir cualquier expresión de transformación válida.

Valor de retorno

TRUE (1) si la fila es un número válido.

FALSE (0) si la fila no es un número válido.

NULL si un valor de la expresión es NULL.

Ejemplos

La expresión siguiente comprueba si el puerto ITEM_PRICE tiene números válidos:

```
IS_NUMBER( ITEM_PRICE )
```

ITEM_PRICE	RETURN VALUE
'123.00'	1 (True)
'-3.45e+3'	1 (True)
'-3.45D-3'	1 (True - Windows only)
'-3.45d-3'	0 (False - UNIX only)
'3.45E-'	0 (False) <i>Incomplete number</i>
' '	0 (False) <i>Consists entirely of blanks</i>
''	0 (False) <i>Empty string</i>
'+123abc'	0 (False)
' 123'	1 (True) <i>Leading white blanks</i>
'123 '	1 (True) <i>Trailing white blanks</i>
'ABC'	0 (False)
'-ABC'	0 (False)
NULL	NULL

Utilice IS_NUMBER para probar los datos antes de utilizar una de las funciones de conversión numéricas, como TO_FLOAT. Por ejemplo, la expresión siguiente comprueba los valores en el puerto ITEM_PRICE y convierte cada número válido en un valor de coma flotante de precisión doble. Si el valor no es un número válido, el Servicio de integración de datos devuelve 0,00:

```
IIF( IS_NUMBER ( ITEM_PRICE ), TO_FLOAT( ITEM_PRICE ), 0.00 )
```

ITEM_PRICE	RETURN VALUE
'123.00'	123
'-3.45e+3'	-3450
'3.45E-3'	0.00345
' '	0.00 <i>Consists entirely of blanks</i>
''	0.00 <i>Empty string</i>
'+123abc'	0.00
' ' 123ABC'	0.00
'ABC'	0.00

ITEM_PRICE	RETURN VALUE
'-ABC'	0.00
NULL	NULL

IS_SPACES

Devuelve si el valor de una cadena se compone únicamente de espacios. Un espacio es un espacio en blanco, un salto de página, un salto de línea, un retorno de carro, un tabulador o un tabulador vertical.

IS_SPACES devuelve FALSE en un cadena vacía porque no contiene espacios. Para comprobar una cadena vacía, use LENGTH.

Sintaxis

```
IS_SPACES( value )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>value</i>	Obligatorio	Debe ser un tipo de datos String. Pasa las filas que desea evaluar. Se puede introducir cualquier expresión de transformación válida.

Valor de retorno

TRUE (1) si la fila se compone únicamente de espacios.

FALSE (0) si la fila contiene datos.

NULL si un valor de la expresión es NULL.

Ejemplo

La expresión siguiente comprueba si las filas del puerto ITEM_NAME se componen únicamente de espacios:

```
IS_SPACES( ITEM_NAME )
```

ITEM_NAME	RETURN VALUE
Flashlight	0 (False)
	1 (True)
Regulator system	0 (False)
NULL	NULL
' '	0 (FALSE) (Empty string does not contain spaces.)

Sugerencia: Utilice IS_SPACES para evitar la escritura de espacios en una columna de caracteres de una tabla de destino. Por ejemplo, si tiene una transformación que escribe nombres de clientes en una columna

CHAR(5) de longitud fija de una tabla de destino, podría escribir '00000' en lugar de espacios. Puede crear una expresión similar a la siguiente:

```
IIF( IS_SPACES( CUST_NAMES ), '00000', CUST_NAMES )
```

LAG

Devuelve el valor que es un número de desplazamiento de filas antes de la fila actual en una transformación de expresión. Utilice esta función para comparar los valores de la fila actual con los valores de una fila anterior cuando ejecute una asignación en el motor Spark en el entorno de Hadoop.

Aparece un valor lag antes de la fila actual en un conjunto de datos.

Cuando se utiliza LAG en una transformación, se debe configurar la transformación para la administración de ventanas. Las propiedades de administración de ventanas definen cómo se particionan y se ordenan los datos.

Sintaxis

```
LAG ( column_name, offset, default )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio/ Opcional	Descripción
<i>column_name</i>	Obligatorio	La expresión o la columna de destino en la que opera la función.
<i>offset</i>	Obligatorio	Tipo de datos enteros. El número de filas antes de la fila actual a partir de las que se obtiene un valor.
<i>default</i>	Opcional	El valor predeterminado que se devolverá en caso de que el desplazamiento esté fuera de los límites de la partición o de la tabla. El valor predeterminado es NULL.

Valor de retorno

El tipo de datos del *column_name* especificado.

Default si el valor de devolución está fuera de los límites de la partición especificada.

NULL si se omite *default* o se establece en NULL.

Ejemplos

La siguiente expresión devuelve la fecha en la que se realizó el pedido anterior:

```
LAG ( ORDER_DATE, 1, NULL )
```

ORDER_DATE	ORDER_ID	RETURN VALUE
2017/09/25	1	NULL

ORDER_DATE	ORDER_ID	RETURN VALUE
2017/09/26	2	2017/09/25
2017/09/27	3	2017/09/26
2017/09/28	4	2017/09/27
2017/09/29	5	2017/09/28
2017/09/30	6	2017/09/29

El valor lag de la primera fila está fuera de la partición, por lo que la función ha devuelto el valor predeterminado de NULL.

En el siguiente ejemplo, su organización recibe pings de GPS de vehículos que incluyen ID de viajes y eventos y una marca de tiempo. Desea calcular la diferencia de tiempo entre cada ping.

La siguiente expresión calcula la diferencia de tiempo entre la fila actual y la fila anterior para dos viajes independientes:

```
DATE_DIFF( EVENT_TIME, LAG ( EVENT_TIME, 1, NULL ), 'ss' )
```

Particiona los datos por ID de viaje y ordena por ID de evento.

TRIP_ID	EVENT_ID	EVENT_TIME	RETURN VALUE
101	1	2017-05-03 12:00:00	NULL
101	2	2017-05-03 12:00:34	34
101	3	2017-05-03 12:02:00	86
102	1	2017-05-03 12:00:00	NULL
102	2	2017-05-03 12:01:56	116
102	3	2017-05-03 12:02:00	4

Los valores lag de la primera y la cuarta fila están fuera de la partición especificada, por lo que la función ha devuelto dos valores NULL predeterminados.

LAST

Devuelve la última fila del puerto seleccionado. Si lo desea, puede aplicar un filtro para limitar las filas que lee el Servicio de integración de datos. Sólo puede anidar otra función de agregado con LAST.

Sintaxis

```
LAST( value [, filter_condition ] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>value</i>	Obligatorio	Cualquier tipo de dato, excepto binario. Pasa los valores para los que desea devolver la última fila. Se puede introducir cualquier expresión de transformación válida.
<i>filter_condition</i>	Opcional	Limita las filas de la búsqueda. La condición de filtro debe ser un valor numérico o dar como resultado TRUE, FALSE o NULL. Se puede introducir cualquier expresión de transformación válida.

Valor de retorno

Última fila de un puerto.

NULL si todos los valores pasados a la función son NULL o si no hay filas seleccionadas (por ejemplo, si la condición de filtro da como resultado FALSE o NULL para todas las filas).

Ejemplo

La expresión siguiente devuelve la última fila del puerto ITEMS_NAME con un precio mayor de 10,00 \$:

```
LAST( ITEM_NAME, ITEM_PRICE > 10 )
```

ITEM_NAME	ITEM_PRICE
Flashlight	35.00
Navigation Compass	8.05
Regulator System	150.00
Flashlight	29.00
Depth/Pressure Gauge	88.00
Vest	31.00
RETURN VALUE: Vest	

LAST_DAY

Devuelve la fecha del último día del mes para cada fecha en un puerto.

Sintaxis

```
LAST_DAY( date )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>date</i>	Obligatorio	Tipo de datos de fecha/hora. Pasa las fechas de las que desea devolver el último día del mes. Puede introducir cualquier expresión de transformación válida que dé como resultado una fecha.

Valor de retorno

Fecha. El último día del mes para el valor de la fecha que se pase a esta función.

NULL si un valor del puerto seleccionado es NULL.

Nulo

Si uno de los valores es NULL, LAST_DAY ignora la fila. Sin embargo, si todos los valores pasados desde el puerto son NULL, LAST_DAY devuelve NULL.

Agrupar por

LAST_DAY agrupa los valores en base a los puertos de agrupación que se hayan definido en la transformación, devolviendo un resultado para cada grupo. Si no hay un puerto de agrupación, LAST_DAY trata todas las filas como un solo grupo, devolviendo un solo valor.

Ejemplos

La siguiente expresión devuelve el último día del mes para cada fecha del puerto ORDER_DATE.

```
LAST_DAY( ORDER_DATE )
```

ORDER_DATE	RETURN VALUE
Apr 1 1998 12:00:00AM	Apr 30 1998 12:00:00AM
Jan 6 1998 12:00:00AM	Jan 31 1998 12:00:00AM
Feb 2 1996 12:00:00AM	Feb 29 1996 12:00:00AM (Leap year)
NULL	NULL
Jul 31 1998 12:00:00AM	Jul 31 1998 12:00:00AM

Puede anidar TO_DATE para convertir los valores de cadena en una fecha. TO_DATE siempre incluye la información de la hora. Si pasa una cadena que no tiene un valor de hora, la fecha devuelta incluye la hora 00:00:00.

El siguiente ejemplo devuelve el último día del mes para cada fecha de pedido en el mismo formato que la cadena.

```
LAST_DAY( TO_DATE( ORDER_DATE, 'DD-MON-YY' ) )
```

ORDER_DATE	RETURN VALUE
'18-NOV-98'	Nov 30 1998 00:00:00
'28-APR-98'	Apr 30 1998 00:00:00

ORDER_DATE	RETURN_VALUE
NULL	NULL
'18-FEB-96'	Feb 29 1996 00:00:00 (<i>Leap year</i>)

LEAD

Devuelve el valor que es un número de desplazamiento de filas tras la fila actual en una transformación de expresión. Utilice esta función para comparar los valores de la fila actual con los valores de una fila futura cuando ejecute una asignación en el motor Spark en el entorno de Hadoop.

Aparece un valor lead después de la fila actual en un conjunto de datos.

Nota: Cuando se utiliza LEAD en una transformación, se debe configurar la transformación para la administración de ventanas. Las propiedades de administración de ventanas definen cómo se particionan y se ordenan los datos.

Sintaxis

```
LEAD ( column_name, offset, default )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio/ Opcional	Descripción
<i>column_name</i>	Obligatorio	La expresión o la columna de destino en la que opera la función.
<i>offset</i>	Obligatorio	Tipo de datos enteros. El número de filas tras la fila actual a partir de las que se obtiene un valor.
<i>default</i>	Opcional	El valor predeterminado que se devolverá en caso de que el desplazamiento esté fuera de los límites de la partición o de la tabla. El valor predeterminado es NULL.

Valor de retorno

El tipo de datos del *column_name* especificado.

Default si el valor de devolución está fuera de los límites de la partición especificada.

NULL si se omite *default* o se establece en NULL.

Ejemplos

La siguiente expresión devuelve, para cada empleado, la fecha en la que se contrató al siguiente empleado:

```
LEAD ( HIRE_DATE, 1, NULL )
```

EMPLOYEE	HIRE_DATE	RETURN VALUE
Hynes	2012/12/07	2014/05/18
Williams	2014/05/18	2015/07/24
Pritchard	2015/07/24	2015/12/24
Snyder	2015/12/24	2016/11/15
Troy	2016/11/15	2017/08/10
Randolph	2017/08/10	NULL

No hay ningún valor lead disponible para la última fila, por lo que la función ha devuelto el valor predeterminado NULL.

La siguiente expresión devuelve la diferencia en valores de cuota de ventas que existe entre el primer trimestre y el tercer trimestre de dos años naturales:

```
LEAD ( Sales_Quota, 2, 0 ) - Sales_Quota
```

Los datos se particionan por año y se ordenan por trimestre.

YEAR	QUARTER	SALES_QUOTA	QUOTA_DIFF
2016	1	300	7700
2016	2	7000	0
2016	3	8000	0
2017	1	5000	4000
2017	2	400	0
2017	3	9000	0

Los valores lead del segundo y el tercer trimestre están fuera de la partición especificada, por lo que la función ha devuelto el valor "0".

LEAST

Devuelve el valor menor de una lista de valores de entrada. De forma predeterminada, la coincidencia distingue mayúsculas de minúsculas.

Sintaxis

```
LEAST( value1, [value2, ..., valueN,] CaseFlag )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>valor</i>	Obligatorio	Cualquier tipo de datos a excepción de los binarios. El tipo de datos debe ser compatible con otros valores. Valor que desea comparar con otros valores. Debe introducir por lo menos un argumento de valor. Si el valor es numérico y otros valores de entrada son de otros tipos de datos numéricos, todos los valores usan la mayor precisión posible. Por ejemplo, si el tipo de datos de algunos valores es Entero y el de otros es Doble, el Servicio de integración de datos convierte los valores en Doble.
<i>CaseFlag</i>	Opcional	Debe ser un entero. Especifique un valor cuando el argumento de valor de entrada sea un valor de cadena. Determina si en los argumentos de esta función se distingue entre mayúsculas y minúsculas. Se puede especificar cualquier expresión de transformación válida. Cuando CaseFlag es un número distinto de 0, la función distingue entre mayúsculas y minúsculas. Cuando CaseFlag es 0, la función no distingue entre mayúsculas y minúsculas. El valor predeterminado distingue mayúsculas de minúsculas.

Valor devuelto

value1 si es el menor de los valores de entrada, *value2* si es el menor de los valores de entrada, y así sucesivamente.

NULL si alguno de los argumentos es nulo.

Ejemplo

La expresión siguiente devuelve la cantidad menor de los artículos pedidos:

```
LEAST( QUANTITY1, QUANTITY2, QUANTITY3 )
```

QUANTITY1	QUANTITY2	QUANTITY3	RETURN VALUE
150	756	27	27
			NULL
5000	97	17	17
120	1724	965	120

LENGTH

Devuelve el número de caracteres de una cadena, incluidos los espacios en blanco finales.

Sintaxis

```
LENGTH( string )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>string</i>	Obligatorio	Tipo de datos String. Cadenas que se van a evaluar. Puede especificar cualquier expresión de transformación válida.

Valor devuelto

Entero que representa la longitud de la cadena.

Se devuelve NULL si el valor pasado a la función es NULL.

Ejemplo

La siguiente expresión devuelve la longitud del nombre de cada cliente:

```
LENGTH( CUSTOMER_NAME )
```

CUSTOMER_NAME	RETURN VALUE
Bernice Davis	13
NULL	NULL
John Baer	9
Greg Brown	10

Sugerencias para LENGTH

Use LENGTH para comprobar las condiciones de las cadenas vacías. Si desea buscar campos en los que no se incluye el nombre del cliente, use una expresión como la siguiente:

```
IIF( LENGTH( CUSTOMER_NAME ) = 0, 'EMPTY STRING' )
```

Para comprobar los campos nulos, use ISNULL. Para comprobar los espacios, use IS_SPACES.

LN

Devuelve el logaritmo natural de un valor numérico. Por ejemplo, LN(3) devuelve 1,098612. Esta función se suele usar para analizar los datos científicos en lugar de los datos empresariales.

Esta función tiene una correspondencia recíproca con la función EXP.

Sintaxis

```
LN( numeric_value )
```


En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. Debe ser un número positivo mayor que 0. Pasa los valores para los que se va a calcular el logaritmo natural. Puede especificar cualquier expresión de transformación válida.

Valor devuelto

Valor doble.

Se devuelve NULL si el valor pasado a la función es NULL.

Ejemplo

La siguiente expresión devuelve el logaritmo natural para todos los valores del puerto NUMBERS:

```
LN( NUMBERS )
```

NUMBERS	RETURN VALUE
10	2.302585092994
125	4.828313737302
0.96	-0.04082199452026
NULL	NULL
-90	Error. (The Integration Service does not write row.)
0	Error. (The Integration Service does not write row.)

Nota: El Servicio de integración de datos muestra un error y no escribe la fila si pasa un número negativo o 0. El valor de *numeric_value* debe ser un número positivo mayor que 0.

LOG

Devuelve el logaritmo de un valor numérico. Muy a menudo, se utiliza esta función para analizar los datos científicos en lugar de los datos empresariales.

Sintaxis

```
LOG( base, exponent )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>base</i>	Obligatorio	La base del logaritmo. Debe ser un valor positivo numérico distinto de 0 ó 1. Cualquier expresión de transformación válida cuyo resultado es un número positivo distinto de 0 ó 1.
<i>exponent</i>	Obligatorio	El exponente del logaritmo. Debe ser un valor numérico positivo mayor que 0. Cualquier expresión de transformación válida cuyo resultado sea un número positivo mayor que 0.

Valor de retorno

Valor doble.

NULL si el valor pasado a la función es NULL.

Ejemplo

La siguiente expresión devuelve el logaritmo de todos los valores del puerto NUMBERS:

```
LOG( BASE, EXPONENT )
```

BASE	EXPONENT	RETURN VALUE
15	1	0
.09	10	-0.956244644696599
NULL	18	NULL
35.78	NULL	NULL
-9	18	Error. (Servicio de integración de datos does not write the row.)
0	5	Error. (Servicio de integración de datos does not write the row.)
10	-2	Error. (Servicio de integración de datos does not write the row.)

El Servicio de integración de datos muestra un error y no escribe la fila si se pasa un número negativo, 0 ó 1 como valor base o si se pasa un valor negativo para el exponente.

LOWER

Convierte a minúsculas los caracteres de cadenas en mayúsculas.

Sintaxis

```
LOWER( string )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>cadena</i>	Obligatorio	Cualquier valor de cadena. El argumento pasa los valores de cadena que se van a devolver en minúsculas. Puede especificar cualquier expresión de transformación válida que tenga como resultado una cadena.

Valor devuelto

Cadena de caracteres en minúsculas. Si los datos contienen caracteres multibyte, el valor devuelto depende de la página de códigos y el modo de movimiento de datos del servicio de integración.

NULL si un valor del puerto seleccionado es NULL.

Ejemplo

La siguiente expresión devuelve todos los nombres en minúsculas:

```
LOWER( FIRST_NAME )
```

FIRST_NAME	RETURN VALUE
antonia	antonia
NULL	NULL
THOMAS	thomas
PierRe	pierre
BERNICE	bernice

LPAD

Añade un conjunto de espacios o caracteres al inicio de una cadena para convertirla a una longitud especificada.

Sintaxis

```
LPAD( first_string, length [,second_string] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>first_string</i>	Obligatorio	Puede ser una cadena de carácter. Las cadenas que se desea cambiar. Se puede especificar cualquier expresión de transformación válida.
<i>length</i>	Obligatorio	Debe ser un literal entero positivo. Este argumento especifica la longitud que desea que tenga cada cadena.
<i>second_string</i>	Opcional	Puede ser cualquier valor de cadena. Los caracteres que desea anexas a la izquierda de los valores <i>first_string</i> . Se puede especificar cualquier expresión de transformación válida. Se puede introducir un literal de cadena específico. Sin embargo, delimite los caracteres que desea añadir al inicio de la cadena mediante comillas simples, como 'abc'. Este argumento distingue entre mayúsculas y minúsculas. Si se omite <i>second_string</i> , la función rellena el inicio de la primera cadena con espacios.

Valor de devolución

Cadena de la longitud especificada.

NULL si un valor pasado a la función es NULL o si *length* es un número negativo.

Ejemplos

La siguiente expresión estandariza números a seis dígitos añadiendo ceros al inicio de estos:

```
LPAD( PART_NUM, 6, '0')
```

PART_NUM	RETURN VALUE
702	000702
1	000001
0553	000553
484834	484834

LPAD cuenta la longitud de izquierda a derecha. Si la primera cadena es más larga que la longitud, LPAD trunca la cadena de derecha a izquierda. Por ejemplo, LPAD('alphabetical', 5, 'x') devuelve la cadena 'alpha'.

Si la segunda cadena es más larga que el número total de caracteres necesarios para devolver la longitud especificada, LPAD utiliza una porción de la segunda cadena:

```
LPAD( ITEM_NAME, 16, '*.*.*' )
```

ITEM_NAME	RETURN VALUE
Flashlight	*.*.*.Flashlight
Compass	*.*.*.*.Compass
Regulator System	Regulator System
Safety Knife	*.*.*Safety Knife

LTRIM

Quita espacios en blanco o caracteres del inicio de una cadena. Puede utilizar LTRIM con IIF o DECODE en una transformación de expresión o estrategia de actualización para evitar espacios en la tabla de destino.

Si no especifica un parámetro *trim_set* en la expresión:

- En UNICODE, LTRIM quita los espacios de uno y dos bytes del principio de una cadena.
- En ASCII, LTRIM quita solamente los espacios de un byte.

Si utiliza LTRIM para quitar los caracteres de una cadena, LTRIM compara, carácter por carácter, el argumento *trim_set* con cada carácter del argumento *string*, empezando por la derecha de la cadena. Si el carácter de la cadena coincide con cualquier carácter de *trim_set*, LTRIM lo quita. LTRIM sigue la comparación y quita los caracteres hasta que no puede encontrar un carácter coincidente en *trim_set*. A continuación, devuelve la cadena, que ya no incluye los caracteres coincidentes.

Sintaxis

```
LTRIM( string [, trim_set] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumentos	Obligatorio / Opcional	Descripción
<i>string</i>	Obligatorio	Cualquier valor de cadena. Transfiere las cadenas que desea modificar. Puede especificar cualquier expresión de transformación válida. Utilice operadores para efectuar las comparaciones o para concatenar las cadenas antes de quitar los caracteres del principio de una cadena.
<i>trim_set</i>	Opcional	Cualquier valor de cadena. Transfiere los caracteres que desea quitar del principio de la primera cadena. Puede especificar cualquier expresión de transformación válida. También puede especificar una cadena de caracteres. Sin embargo, debe encerrar entre comillas simples los caracteres que desea quitar del principio de la cadena, por ejemplo, 'abc'. Si omite la segunda cadena, la función quita los espacios en blanco del principio de la cadena. LTRIM distingue entre mayúsculas y minúsculas. Si desea quitar, por ejemplo, el carácter 'A' de la cadena 'Alfredo', debe especificar 'A' y no 'a'.

Valor de devolución

Cadena. Se quitan los valores de cadena con los caracteres especificados en el argumento *trim_set*.

NULL si el valor que se ha transferido a la función es NULL. Si *trim_set* es NULL, la función devuelve NULL.

Ejemplo:

La siguiente expresión quita los caracteres 'S' y '.' de las cadenas del puerto LAST_NAME:

```
LTRIM( LAST_NAME, 'S.')
```

LAST_NAME	RETURN VALUE
Nelson	Nelson
Osborne	Osborne
NULL	NULL

LAST_NAME	RETURN VALUE
S. MacDonald	MacDonald
Sawyer	awyer
H. Bender	H. Bender
Steadman	teadman

LTRIM quita 'S.' de S. MacDonald y la 'S' de Sawyer y Steadman, pero no el punto de H. Bender. El motivo es que LTRIM busca, carácter por carácter, el conjunto de caracteres especificado en el argumento *trim_set*. Si el primer carácter de la cadena coincide con el primer carácter de *trim_set*, LTRIM lo quita. LTRIM se centra a continuación en el segundo carácter de la cadena. Si coincide con el segundo carácter en *trim_set*, LTRIM lo quita, y así sucesivamente. Cuando el primer carácter de la cadena no coincide con el carácter correspondiente de *trim_set*, LTRIM devuelve la cadena y verifica la siguiente fila.

En el ejemplo de H. Bender, H no coincide con ningún carácter del argumento *trim_set*, así que LTRIM devuelve la cadena en el puerto LAST_NAME y prosigue con la siguiente fila.

Consejos para LTRIM

Utilice RTRIM y LTRIM con || o CONCAT para quitar los espacios en blanco al principio o al final después de concatenar dos cadenas.

Si anida LTRIM, también puede quitar varios conjuntos de caracteres. Por ejemplo, si desea quitar los espacios en blanco al principio y el carácter 'T' de una columna de nombres, puede crear una expresión similar a la siguiente:

```
LTRIM( LTRIM( NAMES ), 'T' )
```

MAKE_DATE_TIME

Devuelve la fecha y la hora en función de los valores de entrada.

Sintaxis

```
MAKE_DATE_TIME( year, month, day, hour, minute, second, nanosecond )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>year</i>	Obligatorio	Tipo de dato numérico. Número entero positivo de 4 dígitos. Si se pasa a esta función un año de 2 dígitos, el Servicio de integración de datos devuelve "00" como los dos primeros dígitos del año.
<i>month</i>	Obligatorio	Tipo de dato numérico. Número entero positivo entre 1 y 12 (enero = 1 y diciembre = 12).
<i>day</i>	Obligatorio	Tipo de dato numérico. Número entero positivo entre 1 y 31 (a excepción de los meses que tienen menos de 31 días: febrero, abril, junio, septiembre y noviembre).

Argumento	Obligatorio / Opcional	Descripción
<i>hour</i>	Opcional	Tipo de dato numérico. Número entero positivo entre 0 y 24 (donde 0=12 a.m., 12=12 p.m., y 24=12 a.m.).
<i>minute</i>	Opcional	Tipo de dato numérico. Entero positivo comprendido entre 0 y 59.
<i>second</i>	Opcional	Tipo de dato numérico. Entero positivo comprendido entre 0 y 59.
nanosegundo	Opcional	Tipo de dato numérico. Número entero positivo entre 0 y 999.999.999.

Valor de retorno

Fecha como MM/DD/YYYY HH24:MI:SS. Devuelve un valor nulo si no se pasa un año, un mes o un día a la función.

Ejemplo

La expresión siguiente crea una fecha y hora de los puertos de entrada:

```
MAKE_DATE_TIME( SALE_YEAR, SALE_MONTH, SALE_DAY, SALE_HOUR, SALE_MIN, SALE_SEC )
```

SALE_YR	SALE_MTH	SALE_DAY	SALE_HR	SALE_MIN	SALE_SEC	RETURN VALUE
2002	10	27	8	36	22	10/27/2002 08:36:22
2000	6	15	15	17		06/15/2000 15:17:00
2003	1	3		22	45	01/03/2003 00:22:45
04	3	30	12	5	10	03/30/0004 12:05:10
99	12	12	5		16	12/12/0099 05:00:16

MAP

Genera una asignación con elementos basados en el par clave-valor especificado.

Sintaxis

```
MAP(map_key1 as any, map_value1 as any [, map_key2, map_value2]...)
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio/Opcional	Descripción
map_key1	Obligatorio	Cualquier tipo de datos primitivo. Un elemento que desea añadir como una clave de los datos de asignación. Se puede especificar cualquier expresión de transformación válida.
map_value1	Obligatorio	Cualquier tipo de datos primitivo o complejo. Un elemento que desea añadir como un valor de la clave de los datos de asignación. Se puede especificar cualquier expresión de transformación válida.

Si utiliza la función MAP en una expresión de salida para un puerto de asignación, el tipo de datos de los argumentos de la función debe coincidir con el tipo de datos de los elementos de la asignación especificados en la configuración de tipos para el puerto de asignación. map_key no puede ser nulo.

Valor de devolución

Asignación.

El tipo de datos de los argumentos determina el tipo de datos de los elementos de asignación. Por ejemplo, si pasa argumentos de entero como clave y argumentos de estructura como valor, la función genera datos de asignación con un par clave-valor de tipos entero y estructura.

Ejemplos

La siguiente expresión genera una asignación de elementos de enteros y de cadena.

```
MAP(emp_id, emp_name)
```

emp_id	emp_name	RETURN VALUE
45781	'Laura'	[45781 -> 'Lauren']
78345	'Derrick'	[78345 -> 'Derrick']
87289	'Kevin'	[87289 -> 'Kevin']
30912		[30912 -> NULL]

MAP_FROM_ARRAYS

Genera una asignación a partir de las matrices de claves y valores especificadas.

Sintaxis

```
MAP_FROM_ARRAYS(map_keys as ARRAY, map_values as ARRAY)
```


En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio/Opcional	Descripción
map_keys	Obligatorio	Tipo de matriz con elementos de tipo de datos primitivo. Matriz de elementos que desea añadir como claves de los datos de asignación. Se puede especificar cualquier expresión de transformación válida.
map_values	Obligatorio	Tipo de matriz con elementos de cualquier tipo de datos. Matriz de elementos que desea añadir como valores de claves de los datos de asignación. Se puede especificar cualquier expresión de transformación válida.

El número de elementos de la matriz map_keys y la matriz map_values debe ser el mismo. Si utiliza la función MAP en una expresión de salida para un puerto de asignación, el tipo de datos de los argumentos de la función debe coincidir con el tipo de datos de los elementos de la asignación especificados en la configuración de tipos para el puerto de asignación.

Valor de devolución

Asignación.

El tipo de datos de los argumentos determina el tipo de datos de los elementos de asignación. Por ejemplo, si pasa argumentos de entero como clave y argumentos de estructura como valor, la función genera datos de asignación con un par clave-valor de tipos entero y estructura.

Ejemplos

La siguiente expresión genera una asignación a partir de los elementos de la matriz del tipo cadena.

```
MAP_FROM_ARRAYS(cust_name, cust_phone)
```

cust_type	cust_name	cust_phone	RETURN VALUE
silver	[Adams, Clark]	[205-128-6478, 722-515-2889]	[Adams -> 205-128-6478, Clark -> 722-515-2889]
gold	[Baker, Davis]	[107-081-0961, 718-051-8116]	[Baker -> 107-081-0961, Davis -> 718-051-8116]
platinum	[Evans, Hills]	[344-894-6463, 861-411-8361]	[Evans -> 344-894-6463, Hills -> 861-411-8361]

MAP_KEYS

Devuelve una matriz de elementos clave para la asignación especificada.

Sintaxis

```
MAP_KEYS(map as MAP)
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio/Opcional	Descripción
map	Obligatorio	Tipo de datos de asignación. Asignación con elementos del par clave-valor a partir de los cuales desea recuperar las claves. Se puede especificar cualquier expresión de transformación válida.

Valor de devolución

Matriz.

El tipo de datos de las claves determina el tipo de datos de los elementos de matriz. Por ejemplo, si la clave es del tipo cadena, la función genera datos de matriz con elementos de cadena.

Devuelve -1 si la clave de asignación es nula.

Devuelve NULL si la asignación es nula.

Ejemplos

La siguiente expresión genera una matriz con elementos del tipo cadena.

```
MAP_KEYS(stock_sellprice)
```

stock_sellprice	RETURN VALUE
[AAPL -> 150.45, GOOGL -> 1150.96]	[AAPL, GOOGL]
[AMZN -> 1400.54, TSLA -> 339.63]	[AMZN, TSLA]

MAP_VALUES

Devuelve una matriz de elementos de valor para la asignación especificada.

Sintaxis

```
MAP_KEYS(map as MAP)
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio/Opcional	Descripción
map	Obligatorio	Tipo de datos de asignación. Asignación con elementos del par clave-valor a partir de los cuales desea recuperar los valores. Se puede especificar cualquier expresión de transformación válida.

Valor de devolución

Matriz.

El tipo de datos de los valores de la asignación determina el tipo de datos de los elementos de matriz. Por ejemplo, si el valor es de tipo entero, la función genera datos de matriz con elementos de enteros.

Devuelve -1 si el valor de asignación es nulo.

Devuelve NULL si la asignación es nula.

Ejemplos

La siguiente expresión genera una matriz con elementos del tipo cadena.

```
MAP_VALUES(stock_sellprice)
```

stock_sellprice	RETURN VALUE
[AAPL -> 150.45, GOOGL -> 1150.96]	[150.45, 1150.96]
[AMZN -> 1400.54, TSLA -> 339.63]	[1400.54, 339.63]

MAX (Fechas)

Devuelve la última fecha encontrada en un puerto o grupo. Puede aplicar un filtro para limitar las filas de la búsqueda. Sólo puede anidar una función de agregados dentro de MAX.

También puede utilizar MAX para devolver el valor numérico más grande o el mayor valor de cadena en un puerto o en un grupo.

Sintaxis

```
MAX( date [, filter_condition] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>date</i>	Obligatorio	Tipo de datos de fecha/hora. Pasa la fecha para la que desea devolver la fecha máxima. Puede introducir cualquier expresión de transformación válida.
<i>filter_condition</i>	Opcional	Limita las filas de la búsqueda. La condición de filtro debe ser un valor numérico o dar como resultado TRUE, FALSE o NULL. Puede introducir cualquier expresión de transformación válida.

Valor de retorno

Fecha.

NULL si todos los valores pasados a la función son NULL o si no hay filas seleccionadas (por ejemplo, si la condición de filtro da como resultado FALSE o NULL para todas las filas).

Ejemplo

Puede devolver la fecha máxima de un puerto o un grupo. La siguiente expresión devuelve la fecha máxima del pedido de linternas:

```
MAX( ORDERDATE, ITEM_NAME='Flashlight' )
```

ITEM_NAME	ORDER_DATE
Flashlight	Apr 20 1998
Regulator System	May 15 1998
Flashlight	Sep 21 1998
Diving Hood	Aug 18 1998
Flashlight	NULL

MAX (Números)

Devuelve el valor máximo encontrado dentro de un puerto o grupo. Se puede aplicar un filtro para limitar el número de filas en la búsqueda. Dentro de MAX solamente se puede anidar una función agregada adicional. También puede usar MAX para devolver la fecha más reciente o el valor de cadena más alto en un puerto o grupo.

Sintaxis

```
MAX( numeric_value [, filter_condition] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. Pasa los valores numéricos para los que desea devolver un valor numérico máximo. Se puede especificar cualquier expresión de transformación válida.
<i>filter_condition</i>	Opcional	Limita las filas en la búsqueda. La condición de filtro debe ser un valor numérico o dar un valor TRUE, FALSE o NULL. Se puede especificar cualquier expresión de transformación válida.

Valor de retorno

Valor numérico.

NULL si todos los valores pasados a la función son NULL, o si no se ha seleccionado ninguna fila (por ejemplo, la condición del filtro da un valor FALSE o NULL para todas las filas).

Nulos

Si un valor es NULL, MAX lo omite. Sin embargo, si todos los valores pasados desde el puerto son NULL, MAX devuelve NULL.

Agrupar por

MAX agrupa valores por grupos según los puertos definidos en la transformación, devolviendo un resultado para cada grupo.

Si no hay un grupo por puerto, MAX trata todas las filas como un solo grupo, devolviendo un valor.

Ejemplo

La primera expresión devuelve el precio máximo de las linternas:

```
MAX( PRICE, ITEM_NAME='Flashlight' )
```

ITEM_NAME	PRICE
Flashlight	10.00
Regulator System	360.00
Flashlight	55.00
Diving Hood	79.00
Halogen Flashlight	162.00
Flashlight	85.00
Flashlight	NULL
RETURN VALUE: 85.00	

MAX (Cadena)

Devuelve el valor de cadena más alto contenido en un puerto o grupo. Se puede aplicar un filtro para limitar el número de filas en la búsqueda. Dentro de MAX solamente se puede anidar una función agregada adicional.

Nota: La función MAX utiliza el mismo orden de clasificación que la transformación Sorter. Sin embargo, la función MAX distingue entre mayúsculas y minúsculas y la transformación Sorter puede no distinguir entre mayúsculas y minúsculas.

También puede usar MAX para devolver la fecha más reciente o el valor numérico más alto en un puerto o grupo.

Sintaxis

```
MAX( string [, filter_condition] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>string</i>	Obligatorio	Tipo de datos String. Pasa los valores de cadena para los que desea devolver un valor de cadena máximo. Se puede especificar cualquier expresión de transformación válida.
<i>filter_condition</i>	Opcional	Limita las filas en la búsqueda. La condición de filtro debe ser un valor numérico o dar un valor TRUE, FALSE o NULL. Se puede especificar cualquier expresión de transformación válida.

Valor de retorno

Cadena.

NULL si todos los valores pasados a la función son NULL, o si no se ha seleccionado ninguna fila (por ejemplo, la condición del filtro da un valor FALSE o NULL para todas las filas).

Nulos

Si un valor es NULL, MAX lo omite. Sin embargo, si todos los valores pasados desde el puerto son NULL, MAX devuelve NULL.

Agrupar por

MAX agrupa valores por grupos según los puertos definidos en la transformación, devolviendo un resultado para cada grupo.

Si no hay un grupo por puerto, MAX trata todas las filas como un solo grupo, devolviendo un valor.

Ejemplo:

La siguiente expresión devuelve el nombre de artículo máximo del fabricante ID 104:

```
MAX( ITEM_NAME, MANUFACTURER_ID='104' )
```

MANUFACTURER_ID	ITEM_NAME
101	First Stage Regulator
102	Electronic Console
104	Flashlight
104	Battery (9 volt)
104	Rope (20 ft)
104	60.6 cu ft Tank
107	75.4 cu ft Tank
108	Wristband Thermometer

RETURN VALUE: Rope (20 ft)

MD5

Calcula la suma de comprobación del valor de entrada. La función utiliza el algoritmo Message-Digest 5 (MD5). MD5 es una función hash criptográfica unidireccional con un valor de hash de 128 bits. Permite concluir que los valores de entrada son diferentes cuando las sumas de comprobación de los valores de entrada son diferentes. Utilice MD5 para comprobar la integridad de los datos.

Sintaxis

```
MD5( value )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>value</i>	Obligatorio	Tipo de datos String o Binary. Valor para el que se desea calcular la suma de comprobación. La distinción entre mayúsculas y minúsculas del valor de entrada afecta al valor de retorno. Por ejemplo, MD5(informatica) y MD5(Informatica) devuelven valores diferentes.

Valor de retorno

Cadena única de 32 caracteres de dígitos hexadecimales 0-9 y a-f.

NULL si la entrada es un valor nulo.

Ejemplo

Se desea grabar datos modificados en una base de datos. Utilice MD5 para generar valores de suma de comprobación para las filas de datos que se leen de un origen. Cuando ejecute una asignación, compare los valores de suma de comprobación generados con los nuevos valores de suma de comprobación. Luego, grabe en el destino las filas que tienen valores de suma de comprobación actualizados. Se puede concluir que un valor de suma de comprobación actualizado indica que los datos han sido modificados.

Consejo

Puede utilizar el valor de retorno como clave hash.

MEDIAN

Devuelve la mediana de todos los valores de un puerto seleccionado.

Si hay un número par de valores en el puerto, la mediana es el promedio de los dos valores centrales de una serie de valores colocados en orden en una línea de números. Si en el puerto hay un número de valores impar, la mediana es el número central.

Dentro de MEDIAN solamente se puede anidar una función agregada adicional y la función anidada debe devolver un tipo de datos Numérico.

El Servicio de integración de datos lee todas las filas de datos para realizar el cálculo de la mediana. El proceso de lectura de filas de datos para realizar el cálculo puede afectar al rendimiento. Opcionalmente, se puede aplicar un filtro para limitar el número de filas que se leen para calcular la mediana.

Sintaxis

```
MEDIAN( numeric_value [, filter_condition ] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. Pasa los valores para los que desea calcular una mediana. Se puede especificar cualquier expresión de transformación válida.
<i>filter_condition</i>	Opcional	Limita las filas en la búsqueda. La condición de filtro debe ser un valor numérico o dar un valor TRUE, FALSE o NULL. Se puede especificar cualquier expresión de transformación válida.

Valor de retorno

Valor numérico.

NULL si todos los valores pasados a la función son NULL o si no se ha seleccionado ninguna fila. Por ejemplo, la condición de filtro da un valor FALSE o NULL para todas las filas.

Nota: Si el valor devuelto es decimal con una precisión superior a 15, puede habilitar la alta precisión para garantizar una precisión decimal de hasta 38 dígitos.

Nulos

Si un valor es NULL, MEDIAN omite la fila. Sin embargo, si todos los valores pasados desde el puerto son NULL, MEDIAN devuelve NULL.

Agrupar por

MEDIAN agrupa valores por grupos según los puertos definidos en la transformación, devolviendo un resultado para cada grupo.

Si no hay un grupo por puerto, MEDIAN trata todas las filas como un solo grupo, devolviendo un valor.

Ejemplo

Para calcular la mediana del salario de todos los departamentos se crea una transformación de agregación agrupada por departamentos con un puerto que especifique la siguiente expresión:

```
MEDIAN( SALARY )
```

La siguiente expresión devuelve el valor de la mediana de los pedidos de arneses estabilizadores:

```
MEDIAN( SALES, ITEM = 'Stabilizing Vest' )
```

ITEM	SALES
Flashlight	85
Stabilizing Vest	504
Stabilizing Vest	36
Safety Knife	5
Medium Titanium Knife	150

ITEM	SALES
Tank	NULL
Stabilizing Vest	441
Chisel Point Knife	60
Stabilizing Vest	NULL
Stabilizing Vest	1044
Wrist Band Thermometer	110
RETURN VALUE: 472.5	

METAPHONE

Cifra los valores de las cadenas. Puede especificar la longitud de la cadena que desea cifrar.

METAPHONE cifra los caracteres del alfabeto inglés (A-Z). Cifra las letras tanto mayúsculas como minúsculas en mayúsculas.

METAPHONE cifra los caracteres según la siguiente lista de reglas:

- Omite las vocales (A, E, I, O y U) a menos que una de ellas sea el primer carácter de la cadena de entrada. METAPHONE('CAR') devuelve 'KR' y METAPHONE('AAR') devuelve 'AR'.
- Utiliza directrices de codificación especiales.

La tabla siguiente enumera las directrices de codificación METAPHONE:

Entrada	Devuelve	Condición	Ejemplo
B	- n/a	- detrás de M	- METAPHONE ('Lamb') devuelve LM.
B	- B	- en el resto de casos	- METAPHONE ('Box') devuelve BKS.
C	- X	- cuando va seguido de IA o H	- METAPHONE ('Facial') devuelve FXL.
C	- S	- seguido de I, E o Y	- METAPHONE ('Fence') devuelve FNS.
C	- n/a	- detrás de S, y seguido de I, E o Y	- METAPHONE ('Scene') devuelve SN.
C	- K	- en el resto de casos	- METAPHONE ('Cool') devuelve KL.
D	- J	- cuando va seguido de GE, GY o GI	- METAPHONE ('Dodge') devuelve TJ.
D	- T	- en el resto de casos	- METAPHONE ('David') devuelve TFT.
F	- F	- en todos los casos	- METAPHONE ('FOX') devuelve FKS.

Entrada	Devuelve	Condición	Ejemplo
G	- F	- cuando va seguido de H y el primer carácter en la cadena de entrada no es B, D o H	- METAPHONE ('Tough') devuelve TF.
G	- n/a	- cuando va seguido de H y el primer carácter en la cadena de entrada es B, D o H	- METAPHONE ('Hugh') devuelve HF.
G	- J	- cuando va seguido de I, E o Y y no se repite	- METAPHONE ('Magic') devuelve MJK.
G	- K	- en el resto de casos	- METAPHONE ('GUN') devuelve KN.
H	- H	- cuando no va detrás de C, G, P, S o T y va seguido de A, E, I o U	- METAPHONE ('DHAT') devuelve THT.
H	- n/a	- en el resto de casos	- METAPHONE ('Chain') devuelve XN.
J	- J	- en todos los casos	- METAPHONE ('Jen') devuelve JN.
K	- n/a - K	- detrás de C - en el resto de casos	- METAPHONE ('Ckim') devuelve KM. - METAPHONE ('Kim') devuelve KM.
L	- L	- en todos los casos	- METAPHONE ('Laura') devuelve LR.
M	- M	- en todos los casos	- METAPHONE ('Maggi') devuelve MK.
N	- N	- en todos los casos	- METAPHONE ('Nancy') devuelve NNS.
P	- F	- cuando va seguido de H	- METAPHONE ('Phone') devuelve FN.
P	- P	- en el resto de casos	- METAPHONE ('Pip') devuelve PP.
Q	- K	- en todos los casos	- METAPHONE ('Queen') devuelve KN.
R	- R	- en todos los casos	- METAPHONE ('Ray') devuelve R.
S	- X	- cuando va seguido de H, IO, IA o CHW	- METAPHONE ('Cash') devuelve KX.
S	- S	- en el resto de casos	- METAPHONE ('Sing') devuelve SNK.
T	- X	- cuando va seguido de IA o IO	- METAPHONE ('Patio') devuelve PX.
T	- 0 ¹	- cuando va seguido de H	- METAPHONE ('Thor') devuelve 0R.
T	- n/a	- cuando va seguido de CH	- METAPHONE ('Glitch') devuelve KLTX.
T	- T	- en el resto de casos	- METAPHONE ('Tim') devuelve TM.
V	- F	- en todos los casos	- METAPHONE ('Vin') devuelve FN.
W	- W	- cuando va seguido de A, E, I, O o U	- METAPHONE ('Wang') devuelve WNK.
W	- n/a	- en el resto de casos	- METAPHONE ('When') devuelve HN.

Entrada	Devuelve	Condición	Ejemplo
X	- KS	- en todos los casos	- METAPHONE ('Six') devuelve SKS.
S	- Y	- cuando va seguido de A, E, I, O o U	- METAPHONE ('Yang') devuelve YNK.
Y	- n/a	- en el resto de casos	- METAPHONE ('Bobby') devuelve BB.
Z	- S	- en todos los casos	- METAPHONE ('Zack') devuelve SK.

1. El entero 0.

- Omite el carácter inicial y cifra la cadena restante si los dos primeros caracteres de la cadena de entrada tienen uno de los siguientes valores:
 - **KN**. Por ejemplo, METAPHONE('KNOT') devuelve 'NT'.
 - **GN**. Por ejemplo, METAPHONE('GNOB') devuelve 'NB'.
 - **PN**. Por ejemplo, METAPHONE('PNRX') devuelve 'NRKS'.
 - **AE**. Por ejemplo, METAPHONE('AERL') devuelve 'ERL'.
- Si en la cadena de entrada hay un carácter diferente de "C" que se repite más de una vez, se cifra solamente la primera ocurrencia. Por ejemplo, METAPHONE('BBOX') devuelve 'BKS' y METAPHONE('CCOX') devuelve 'KKKS'.

Sintaxis

```
METAPHONE( string [,length] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>string</i>	Obligatorio	Debe ser una cadena de caracteres. Transfiere el valor que desea cifrar. El primer carácter debe ser un carácter del alfabeto inglés (A-Z). Se puede especificar cualquier expresión de transformación válida. Omite los caracteres no alfabéticos de <i>string</i> .
<i>length</i>	Opcional	Debe ser un entero mayor que 0. Especifica el número de caracteres de <i>string</i> que desea cifrar. Se puede especificar cualquier expresión de transformación válida. Cuando <i>length</i> es 0 o un valor mayor que la longitud de <i>string</i> , se cifra toda la cadena de entrada. El valor predeterminado es 0.

Valor devuelto

Cadena.

NULL si una de las siguientes condiciones es verdadera:

- Todos los valores transferidos a la función son NULL.
- No hay ningún carácter de *string* que sea una letra del alfabeto inglés.
- *string* está vacío.

Ejemplos

La siguiente expresión cifra los dos primeros caracteres en el puerto EMPLOYEE_NAME en una cadena:

```
METAPHONE( EMPLOYEE_NAME, 2 )
```

Employee_Name	Return Value
John	JH
*@#	NULL
P\$%oc&&KMNL	PK

La siguiente expresión cifra los cuatro primeros caracteres en el puerto EMPLOYEE_NAME en una cadena:

```
METAPHONE( EMPLOYEE_NAME, 4 )
```

Employee_Name	Return Value
John	JHN
1ABC	ABK
*@#	NULL
P\$%oc&&KMNL	PKKM

MIN (Fechas)

Devuelve la fecha más temprana encontrada en un puerto o grupo. Puede aplicar un filtro para limitar las filas de la búsqueda. Sólo puede anidar otra función de agregado con MIN y la función anidada debe devolver un tipo de datos de fecha.

También puede utilizar MIN para devolver el valor numérico más pequeño o el menor valor de cadena en un puerto o en un grupo.

Sintaxis

```
MIN( date [, filter_condition] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>date</i>	Obligatorio	Tipo de datos de fecha/hora. Pasa los valores para los que desea devolver el valor mínimo. Se puede introducir cualquier expresión de transformación válida.
<i>filter_condition</i>	Opcional	Limita las filas de la búsqueda. La condición de filtro debe ser un valor numérico o dar como resultado TRUE, FALSE o NULL. Se puede introducir cualquier expresión de transformación válida.

Valor de retorno

Fecha si el argumento *value* es una fecha.

NULL si todos los valores pasados a la función son NULL o si no hay filas seleccionadas (por ejemplo, si la condición de filtro da como resultado FALSE o NULL para todas las filas).

Nulos

Si un solo valor es NULL, MIN lo ignora. Sin embargo, si todos los valores pasados desde el puerto son NULL, MIN devuelve NULL.

Agrupar por

MIN agrupa los valores en base a los puertos de agrupación que se hayan definido en la transformación, devolviendo un resultado para cada grupo.

Si no hay un puerto de agrupación, MIN trata a todas las filas como un solo grupo, devolviendo un solo valor.

Ejemplo

La siguiente expresión devuelve la fecha más antigua del pedido de linternas:

```
MIN( ORDER_DATE, ITEM_NAME='Flashlight' )
```

ITEM_NAME	ORDER_DATE
Flashlight	Apr 20 1998
Regulator System	May 15 1998
Flashlight	Sep 21 1998
Diving Hood	Aug 18 1998
Halogen Flashlight	Feb 1 1998
Flashlight	Oct 10 1998
Flashlight	NULL

RETURN VALUE: Feb 1 1998

MIN (Números)

Devuelve el valor mínimo encontrado dentro de un puerto o grupo. Se puede aplicar un filtro para limitar el número de filas en la búsqueda. Dentro de MIN solamente se puede anidar una función agregada adicional y la función anidada debe devolver un tipo de datos numérico.

También puede usar MIN para devolver la fecha más reciente o el valor de cadena más bajo en un puerto o grupo.

Sintaxis

```
MIN( numeric_value [, filter_condition] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipos de datos numéricos. Pasa los valores para los que desea devolver el valor mínimo. Se puede especificar cualquier expresión de transformación válida.
<i>filter_condition</i>	Opcional	Limita las filas en la búsqueda. La condición de filtro debe ser un valor numérico o dar un valor TRUE, FALSE o NULL. Se puede especificar cualquier expresión de transformación válida.

Valor de retorno

Valor numérico.

NULL si todos los valores pasados a la función son NULL, o si no se ha seleccionado ninguna fila (por ejemplo, la condición del filtro da un valor FALSE o NULL para todas las filas).

Nota: Si el valor devuelto es decimal con una precisión superior a 15, puede habilitar la alta precisión para garantizar una precisión decimal de hasta 38 dígitos.

Nulos

Si un valor es NULL, MIN lo omite. Sin embargo, si todos los valores pasados desde el puerto son NULL, MIN devuelve NULL.

Agrupar por

MIN agrupa valores por grupos según los puertos definidos en la transformación, devolviendo un resultado para cada grupo.

Si no hay un grupo por puerto, MIN trata todas las filas como un solo grupo, devolviendo un valor.

Ejemplo

La primera expresión devuelve el precio mínimo de las linternas:

```
MIN ( PRICE, ITEM_NAME='Flashlight' )
```

ITEM_NAME	PRICE
Flashlight	10.00
Regulator System	360.00
Flashlight	55.00
Diving Hood	79.00
Halogen Flashlight	162.00
Flashlight	85.00
Flashlight	NULL
RETURN VALUE: 10.00	

MIN (Cadena)

Devuelve el valor de cadena mínimo encontrado dentro de un puerto o grupo. Se puede aplicar un filtro para limitar el número de filas en la búsqueda. Dentro de MIN solamente se puede anidar una función agregada adicional y la función anidada debe devolver un tipo de datos String.

Nota: La función MIN utiliza el mismo orden de clasificación que la transformación Sorter. Sin embargo, la función MIN distingue entre mayúsculas y minúsculas y la transformación Sorter puede no distinguir entre mayúsculas y minúsculas.

También puede usar MIN para devolver la fecha más reciente o el valor numérico más bajo en un puerto o grupo.

Sintaxis

```
MIN( string [, filter_condition] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>string</i>	Obligatorio	Tipo de datos String. Pasa los valores para los que desea devolver el valor mínimo. Se puede especificar cualquier expresión de transformación válida.
<i>filter_condition</i>	Opcional	Limita las filas en la búsqueda. La condición de filtro debe ser un valor numérico o dar un valor TRUE, FALSE o NULL. Se puede especificar cualquier expresión de transformación válida.

Valor de retorno

Valor de cadena.

NULL si todos los valores pasados a la función son NULL, o si no se ha seleccionado ninguna fila (por ejemplo, la condición del filtro da un valor FALSE o NULL para todas las filas).

Nulos

Si un valor es NULL, MIN lo omite. Sin embargo, si todos los valores pasados desde el puerto son NULL, MIN devuelve NULL.

Agrupar por

MIN agrupa valores por grupos según los puertos definidos en la transformación, devolviendo un resultado para cada grupo.

Si no hay un grupo por puerto, MIN trata todas las filas como un solo grupo, devolviendo un valor.

Ejemplo

La siguiente expresión devuelve el nombre de artículo mínimo del fabricante ID 104:

```
MIN ( ITEM_NAME, MANUFACTURER_ID='104' )
```

MANUFACTURER_ID	ITEM_NAME
101	First Stage Regulator
102	Electronic Console

MANUFACTURER_ID	ITEM_NAME
104	Flashlight
104	Battery (9 volt)
104	Rope (20 ft)
104	60.6 cu ft Tank
107	75.4 cu ft Tank
108	Wristband Thermometer

RETURN VALUE: 60.6 cu ft Tank

MOD

Devuelve el resto de una división. Por ejemplo, `MOD(8,5)` devuelve 3.

Sintaxis

`MOD(numeric_value, divisor)`

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de dato numérico. Los valores que desea dividir. Puede introducir cualquier expresión de transformación válida.
<i>divisor</i>	Obligatorio	El valor numérico por el que desea dividir. El divisor no puede ser 0.

Valor de retorno

Valor numérico del tipo de datos que se pasa a la función. El resto del valor numérico dividido por el divisor.

NULL si el valor pasado a la función es NULL.

Ejemplos

La expresión siguiente devuelve el módulo de los valores del puerto PRICE divididos por los valores del puerto QTY:

```
MOD( PRICE, QTY )
```

PRICE	QTY	RETURN VALUE
10.00	2	0
12.00	5	2
9.00	2	1
15.00	3	0
NULL	3	NULL
20.00	NULL	NULL
25.00	0	Error. Integration Service does not write row.

La última fila (25, 0) provocó un error porque no se puede dividir por 0. Para evitar la división por 0, puede crear una expresión similar a la siguiente que devuelve el módulo del precio dividido por la cantidad sólo si la cantidad no es 0. Si la cantidad es 0, la función devuelve NULL:

```
MOD( PRICE, IIF( QTY = 0, NULL, QTY ) )
```

PRICE	QTY	RETURN VALUE
10.00	2	0
12.00	5	2
9.00	2	1
15.00	3	0
NULL	3	NULL
20.00	NULL	NULL
25.00	0	NULL

La última fila (25, 0) produce un valor NULL en lugar de un error porque la función IIF reemplaza NULL con el 0 en el puerto QTY.

MOVINGAVG

Devuelve el promedio (fila a fila) de un conjunto especificado de filas. Opcionalmente, puede aplicar una condición para filtrar filas antes de calcular la media móvil.

Sintaxis

```
MOVINGAVG( numeric_value, rowset [, filter_condition] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. El valor para el cual desea calcular una media móvil. Se puede especificar cualquier expresión de transformación válida.
<i>rowset</i>	Obligatorio	Debe ser un literal entero positivo mayor que 0. Define el conjunto de filas para el cual desea calcular la media móvil. Por ejemplo, si desea calcular una media móvil para una columna de datos, en grupos de cinco filas, puede escribir una expresión como: <code>MOVINGAVG(SALES, 5)</code> .
<i>filter_condition</i>	Opcional	Limita las filas en la búsqueda. La condición de filtro debe ser un valor numérico o dar un valor TRUE, FALSE o NULL. Se puede especificar cualquier expresión de transformación válida.

Valor de retorno

Valor numérico.

NULL si todos los valores pasados a la función son NULL, o si no se ha seleccionado ninguna fila (por ejemplo, la condición del filtro da un valor FALSE o NULL para todas las filas).

Nota: Si el valor devuelto es decimal con una precisión superior a 15, puede habilitar la alta precisión para garantizar una precisión decimal de hasta 38 dígitos.

Nulos

MOVINGAVG omite los valores nulos cuando calcula la media móvil. Sin embargo, si todos los valores son NULL, la función devuelve NULL.

Ejemplo

La siguiente expresión devuelve el pedido promedio de un arnés estabilizador, basado en las primeras cinco filas del puerto Ventas y, a continuación, devuelve el promedio de las últimas cinco filas leídas:

```
MOVINGAVG( SALES, 5 )
```

ROW_NO	SALES	RETURN VALUE
1	600	NULL
2	504	NULL
3	36	NULL
4	100	NULL
5	550	358
6	39	245.8
7	490	243

La función devuelve el promedio para un conjunto de cinco filas: 358 basado en las filas 1 a 5, 245,8 basado en las filas 2 a 6 y 243 basado en las filas 3 a 7.

MOVINGSUM

Devuelve la suma (fila a fila) de un conjunto especificado de filas.

Opcionalmente, puede aplicar una condición para filtrar filas antes de calcular la suma móvil.

Sintaxis

```
MOVINGSUM( numeric_value, rowset [, filter_condition] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. El valor para el cual desea calcular una suma móvil. Se puede especificar cualquier expresión de transformación válida.
<i>rowset</i>	Obligatorio	Debe ser un literal entero positivo mayor que 0. Define el conjunto de filas para el cual desea calcular la suma móvil. Por ejemplo, si desea calcular una suma móvil para una columna de datos, en grupos de cinco filas, puede escribir una expresión como: <code>MOVINGSUM(SALES, 5)</code>
<i>filter_condition</i>	Opcional	Limita las filas en la búsqueda. La condición de filtro debe ser un valor numérico o dar un valor TRUE, FALSE o NULL. Se puede especificar cualquier expresión de transformación válida.

Valor de retorno

Valor numérico.

NULL si todos los valores pasados a la función son NULL, o si la función no selecciona ninguna fila (por ejemplo, la condición del filtro da un valor FALSE o NULL para todas las filas).

Nota: Si el valor devuelto es decimal con una precisión superior a 15, puede habilitar la alta precisión para garantizar una precisión decimal de hasta 38 dígitos.

Nulos

MOVINGSUM omite los valores nulos cuando calcula la suma móvil. Sin embargo, si todos los valores son NULL, la función devuelve NULL.

Ejemplo

La siguiente expresión devuelve la suma de pedidos de un arnés estabilizador, basado en las primeras cinco filas del puerto Ventas y, a continuación, devuelve el promedio de las últimas cinco filas leídas:

```
MOVINGSUM( SALES, 5 )
```

ROW_NO	SALES	RETURN VALUE
1	600	NULL

ROW_NO	SALES	RETURN VALUE
2	504	NULL
3	36	NULL
4	100	NULL
5	550	1790
6	39	1229
7	490	1215

La función devuelve la suma para un conjunto de cinco filas: 1790 basado en las filas 1 a 5, 1229 basado en las filas 2 a 6 y 1215 basado en las filas 3 a 7.

NPER

Devuelve el número de períodos para una inversión basada en un tipo de interés constante y periódico, pagos constantes.

Sintaxis

`NPER(rate, present value, payment [, future value, type])`

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>rate</i>	Obligatorio	Numérico. Tipo de interés ganado en cada período. Se expresa como número decimal. Divida el tipo por 100 para expresarlo como número decimal. Debe ser mayor o igual que 0.
<i>present_value</i>	Obligatorio	Numérico. Importe de la cantidad global que vale una serie de pagos futuros.
<i>payment</i>	Obligatorio	Numérico. Importe del pago debido por período. Debe ser un número negativo.
<i>future_value</i>	Opcional	Numérico. Balance monetario que se desea conseguir después de que se haya realizado el último pago. Si omite este valor, NPER utiliza 0.
<i>type</i>	Opcional	Booleano. Programación del pago. Introduzca 1 si el pago se efectúa al inicio del período. Introduzca 0 si el pago se efectúa al final del período. El valor predeterminado es 0. Si introduce un valor distinto de 0 o 1, el Servicio de integración de datos trata el valor como 1.

Valor de retorno

Numérico.

Ejemplo

El valor actual de una inversión es de 500 \$. Cada pago es de 2000 \$ y el valor futuro de la inversión es de 20 000 \$. La siguiente expresión devuelve 9 como el número de períodos para el que se necesita hacer los pagos:

```
NPER ( 0.015, -500, -2000, 20000, TRUE )
```

Notas

Para calcular el tipo de interés obtenido en cada período, divida el tipo de interés anual por el número de pagos efectuados en un año. Por ejemplo, si se realizan pagos mensuales a un tipo de interés anual del 15%, el valor del argumento Tipo de interés es 15% dividido entre 12. Si se realizan pagos anuales, el valor del argumento Tipo de interés es 15%.

El valor del pago y el valor actual son negativos porque estos son los importes que se pagan.

PARSE_JSON

Analiza los datos jerárquicos de JSON en un tipo de datos de cadena y genera un struct. El esquema del struct se basa en la definición especificada de tipo de datos complejo que se pasa en el argumento. Esta función es adecuada cuando se reciben datos jerárquicos en un nivel intermedio de la asignación y hay que analizar los datos para un procesamiento de nivel inferior.

Sintaxis

```
PARSE_JSON(upstream_string, :Type.type_definition_library.type_definition)
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio/Opcional	Descripción
upstream_string	Obligatorio	Campo de cadena de origen, en un nivel intermedio de la asignación, que contiene datos jerárquicos en formato JSON.
:Type.type_definition_library.type_definition	Obligatorio	La definición de tipo de datos complejos que representa el esquema de los datos del struct. Utilice el calificador de referencia :Type para hacer referencia a la biblioteca de definiciones de tipos que contiene la definición de tipo de datos complejos.

Valor de retorno

Struct.

Ejemplos

La siguiente expresión genera un struct a partir de la definición de tipo de datos complejo especificada en **customer_def**.

```
PARSE_JSON(upstream_string, :Type.type_definition_library.customer_def)
```

En el siguiente ejemplo se muestran datos JSON en los campos de nivel superior asignados a un tipo de datos de cadena:

```
{"customer" : "ABC Co", "contract" : "1010", "city" : "Phoenix", "saleamt" : "150000.00"}
{"customer" : "Data Inc", "contract" : "1111", "city" : "Portland", "saleamt" : "20000.00"}
```

La definición de tipo de datos complejo **customer_def** se define en la biblioteca de definiciones de tipos del siguiente modo:

```
customer_def{
  customer : string
  contract : int
  city : string
  saleamt : int
}
```

En la siguiente tabla se muestra cómo la función PARSE_JSON en la expresión de la asignación analiza la cadena de nivel superior mediante **customer_def** y devuelve un struct:

customer	contract	city	saleamt	RETURN VALUE
ABC Co	1010	Phoenix	150000.00	{ customer:ABC Co contract:1010 city:Phoenix saleamt:150000.00 }
Data Inc	1111	Portland	20000.00	{ customer:Data Inc contract:1111 city:Portland saleamt:20000.00 }

PARSE_XML

Analiza los datos jerárquicos de XML en un tipo de datos de cadena y genera un struct. El esquema del struct se basa en la definición especificada de tipo de datos complejo que se pasa en el argumento. Esta función es adecuada cuando se reciben datos jerárquicos en un nivel intermedio de la asignación y hay que analizar los datos para un procesamiento de nivel inferior.

Sintaxis

```
PARSE_XML(upstream_string, :Type.type_definition_library.type_definition)
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio/ Opcional	Descripción
upstream_string	Obligatorio	Campo de cadena de origen, en un nivel intermedio de la asignación, que contiene datos jerárquicos en formato XML.
:Type.type_definition_library.type_definition	Obligatorio	La definición de tipo de datos complejos que representa el esquema de los datos del struct. Utilice el calificador de referencia :Type para hacer referencia a la biblioteca de definiciones de tipos que contiene la definición de tipo de datos complejos.

Valor de retorno

Struct.

Ejemplos

La siguiente expresión genera un struct a partir de la definición de tipo de datos complejo especificada en **customer_def**.

```
PARSE_XML(upstream_string,:Type.type_definition_library.customer_def)
```

En el siguiente ejemplo se muestran datos XML en los campos de nivel superior asignados a un tipo de datos de cadena:

```
<Customers>
  <Customer>
    <CustName>ABC Co</CustName>
    <Contract>1010</Contract>
    <City>Phoenix</City>
    <SaleAmt>150000.00</SaleAmt>
  </Customer>
  <Customer>
    <CustName>Data Inc</CustName>
    <Contract>1111</Contract>
    <City>Portland</City>
    <SaleAmt>20000.00</SaleAmt>
  </Customer>
</Customers>
```

La definición de tipo de datos complejo **customer_def** se define en la biblioteca de definiciones de tipos del siguiente modo:

```
customer_def{
  CustName : string
  Contract : int
  City : string
  SaleAmt : int
}
```

En la siguiente tabla se muestra cómo la función `PARSE_XML` en la expresión de la asignación analiza la cadena de nivel superior mediante `customer_def` y devuelve un struct:

CustName	Contract	City	SaleAmt	RETURN VALUE
ABC Co	1010	Phoenix	150000.00	{ CustName:ABC Co Contract:1010 City:Phoenix SaleAmt:150000.00 }
Data Inc	1111	Portland	20000.00	{ CustName:Data Inc Contract:1111 City:Portland SaleAmt:20000.00 }

PERCENTILE

Calcula el valor que cae en un percentil dado en un grupo de números. Solo puede anidar otra función de agregado más dentro de `PERCENTILE`, y la función anidada debe devolver un tipo de dato numérico.

El Servicio de integración de datos lee todas las filas de datos para efectuar el cálculo del percentil. El proceso de lectura de filas para efectuar el cálculo puede tener efectos en el rendimiento. Puede optar por aplicar un filtro para limitar las filas que se deben leer en el cálculo del percentil.

Sintaxis

```
PERCENTILE( numeric_value, percentile [, filter_condition ] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de dato numérico. Transfiere los valores para los que desea calcular un percentil. Puede especificar cualquier expresión de transformación válida.
<i>percentile</i>	Obligatorio	Entero comprendido entre 0 y 100, ambos inclusive. Transfiere el percentil que desea calcular. Puede especificar cualquier expresión de transformación válida. Si se transfiere un número fuera del intervalo entre 0 y 100, el Servicio de integración de datos muestra un error y no graba la fila.
<i>filter_condition</i>	Opcional	Limita las filas de la búsqueda. La condición de filtro debe ser un valor numérico o debe verificarse como TRUE, FALSE o NULL. Puede especificar cualquier expresión de transformación válida.

Valor de devolución

Valor numérico.

NULL si todos los valores transferidos a la función son NULL, o si no se ha seleccionado ninguna fila (por ejemplo, la condición de filtro se verifica como FALSE o NULL para todas las filas).

Nota: Si el valor devuelto es decimal con una precisión superior a 15, puede habilitar la alta precisión para garantizar una precisión decimal de hasta 38 dígitos.

Nulos

Si un valor es NULL, PERCENTILE omite la fila. Sin embargo, si todos los valores de un grupo son NULL, PERCENTILE devuelve NULL.

Agrupar por

PERCENTILE agrupa los valores en función de los grupos por puerto que haya definido en la transformación y devuelve un resultado por cada grupo.

Si no hay ningún grupo por puerto, PERCENTILE trata todas las filas como un grupo y devuelve un valor.

Ejemplo:

El Servicio de integración de datos calcula un percentil con la siguiente lógica:

$$i = \frac{(x + 1) \times \text{percentile}}{100}$$

Utilice las siguientes directrices para esta ecuación:

- x es el número de elementos en el grupo de valores para los que está calculando un percentil.
- Si $i < 1$, PERCENTILE devuelve el valor del primer elemento de la lista.
- Si i es un valor entero, PERCENTILE devuelve el valor del *enésimo* elemento de la lista.
- De no ser así, PERCENTILE devuelve el valor de n :

$$n = \lceil i \rceil^{\text{th}} \text{element} \times (\lceil i \rceil - i) + \lfloor i \rfloor^{\text{th}} \text{element} \times (i - \lfloor i \rfloor)$$

La siguiente expresión devuelve el salario que se halla en el 75º percentil de los salarios superiores a 50.000 dólares:

```
PERCENTILE( SALARY, 75, SALARY > 50000 )
```

SALARY

125000.0

27900.0

100000.0

NULL

55000.0

9000.0

85000.0

86000.0

48000.0

SALARY

99000.0

RETURN VALUE: 106250.0

PMT

Devuelve el pago de un préstamo basado en los pagos constantes y un tipo de interés constante.

Sintaxis

`PMT(rate, terms, present value[, future value, type])`

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>rate</i>	Obligatorio	Numérico. Tipo de interés del préstamo para cada período. Expresado como un número decimal. Divida el índice entre 100 para expresarlo como un número decimal. Debe ser mayor o igual a 0.
<i>terms</i>	Obligatorio	Numérico. Número de períodos o pagos. Debe ser mayor que 0.
<i>present value</i>	Obligatorio	Numérico. Principio para el préstamo.
<i>future value</i>	Opcional	Numérico. Saldo monetario que desea conseguir después del último pago. Si omite este valor, PMT utiliza 0.
<i>type</i>	Opcional	Booleano. Programación del pago. Introduzca 1 si el pago es al inicio del período. Introduzca 0 si el pago es al final del período. El valor predeterminado es 0. Si introduce un valor distinto de 0 ó 1, el Servicio de integración de datos trata el valor como 1.

Valor de retorno

Numérico.

Ejemplo

La expresión siguiente devuelve -2111,64 como importe de pago mensual de un préstamo.

`PMT(0.01, 10, 20000)`

Notas

Para calcular el tipo de los intereses devengados en cada período, se divide el tipo anual por el número de pagos efectuados en un año. Por ejemplo, si realiza pagos mensuales a un tipo de interés anual del 15%, el tipo será 15%/12. Si realiza pagos anuales, el tipo es del 15%.

El valor del pago es negativo porque se trata de cantidades que usted paga.

POWER

Devuelve un valor elevado al exponente que se pase a la función.

Sintaxis

`POWER(base, exponent)`

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>base</i>	Obligatorio	Valor numérico. Este argumento es el valor de base. Puede introducir cualquier expresión de transformación válida. Si el valor de base es negativo, el exponente tiene que ser un número entero.
<i>exponent</i>	Obligatorio	Valor numérico. Este argumento es el valor del exponente. Puede introducir cualquier expresión de transformación válida. Si el valor de base es negativo, el exponente tiene que ser un número entero. En este caso, la función redondea los valores decimales al número entero más cercano antes de devolver un valor.

Valor de retorno

Valor doble.

NULL si pasa un valor nulo a la función.

Ejemplo

La expresión siguiente devuelve los valores del puerto Números elevados a los valores del puerto Exponente:

`POWER(NUMBERS, EXPONENT)`

NUMBERS	EXPONENT	RETURN VALUE
10.0	2.0	100
3.5	6.0	1838.265625
3.5	5.5	982.594307804838
NULL	2.0	NULL
10.0	NULL	NULL
-3.0	-6.0	0.00137174211248285
3.0	-6.0	0.00137174211248285
-3.0	6.0	729.0
-3.0	5.5	729.0

El valor -3,0 elevado a 6 devuelve los mismos resultados que -3,0 elevado a 5,5. Si la base es negativa, el exponente tiene que ser un número entero. De lo contrario, el Servicio de integración de datos redondea el exponente al valor entero más cercano.

PV

Devuelve el valor actual de una inversión.

Sintaxis

```
PV( rate, terms, payment [, future value, type] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
rate	Obligatorio	Númerico. Tipo de interés ganado en cada período Se expresa como un número decimal. Divida el índice entre 100 para expresarlo como un número decimal. Debe ser mayor o igual a 0.
terms	Obligatorio	Númerico. Número de períodos o pagos. Debe ser mayor que 0.
payments	Obligatorio	Númerico. Importe del pago debido por período. Debe ser un número negativo.
future value	Opcional	Númerico. Balance monetario después del último pago. Si omite este valor, PV utiliza 0.
types	Opcional	Booleano. Programación del pago. Introduzca 1 si el pago es al inicio del período. Introduzca 0 si el pago es al final del período. El valor predeterminado es 0. Si introduce un valor distinto de 0 o 1, el Servicio de integración de datos trata el valor como 1.

Valor de retorno

Númerico.

Ejemplo

La expresión siguiente devuelve 12.524,43 como importe que debe depositar en la cuenta hoy para tener un valor futuro de 20.000 \$ en un año si también deposita 500 \$ al comienzo de cada período:

```
PV( 0.0075, 12, -500, 20000, TRUE )
```

RAND

Devuelve un número aleatorio entre 0 y 1. Esto es útil en los escenarios de probabilidad.

Sintaxis

```
RAND( seed )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>seed</i>	Opcional	Numérico. Valor inicial con el que el servicio de integración genera el número aleatorio. El valor debe ser una constante. Si no especifica ningún valor de inicialización, el Servicio de integración de datos usa la hora del sistema actual para obtener el número de segundos desde el 1 de enero de 1971. Este valor se usa como valor de inicialización.

Valor devuelto

Numérico.

Para el mismo valor de inicialización, el Servicio de integración de datos genera la misma secuencia de números.

Ejemplo

La siguiente expresión puede devolver un valor de 0,417022004702574:

```
RAND (1)
```

RATE

Devuelve el tipo de interés obtenido por período por una garantía.

Sintaxis

```
RATE( terms, payment, present value[, future value, type] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>terms</i>	Obligatorio	Numérico. Número de períodos o pagos. Debe ser mayor que 0.
<i>payments</i>	Obligatorio	Numérico. Importe del pago debido por período. Debe ser un número negativo.
<i>present value</i>	Obligatorio	Numérico. Importe de la cantidad global al que equivale actualmente una serie de pagos futuros.
<i>future value</i>	Opcional	Numérico. Saldo monetario que desea conseguir después del último pago. Por ejemplo, el valor futuro de un préstamo es 0. Si omite este argumento, RATE utiliza 0.
<i>types</i>	Opcional	Booleano. Programación del pago. Introduzca 1 si el pago es al inicio del período. Introduzca 0 si el pago es al final del período. El valor predeterminado es 0. Si introduce un valor distinto de 0 ó 1, el Servicio de integración de datos trata el valor como 1.

Valor de retorno

Numérico.

Ejemplo

La expresión siguiente devuelve 0,0077 como tipo de interés mensual de un préstamo.

```
RATE( 48, -500, 20000 )
```

Para calcular el tipo de interés anual del préstamo, multiplique 0,0077 por 12. El tipo de interés anual es 0,0924 o 9,24%.

REG_EXTRACT

Extrae subpatrones de una expresión regular en un valor de entrada. Puede extraer, por ejemplo, de un patrón de expresión regular de un nombre completo, el nombre o el apellido.

Nota: Utilice la función REG_REPLACE para reemplazar un patrón de caracteres en una cadena con otro patrón de caracteres.

Sintaxis

```
REG_EXTRACT( subject, 'pattern', subPatternNum )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>subject</i>	Obligatorio	Tipo de datos String. Transfiere el valor que desea comparar con el patrón de la expresión regular.
<i>pattern</i>	Obligatorio	Tipo de datos String. Patrón de expresión regular con el que se desea coincidir. Debe utilizar una sintaxis de expresiones regulares compatible con perl. Encierre el patrón entre comillas sencillas. Encierre los subpatrones entre paréntesis.
<i>subPatternNum</i>	Opcional	Valor entero. Número del subpatrón de la expresión regular con el que desea coincidir. Utilice las siguientes directrices para determinar el número del subpatrón: <ul style="list-style-type: none">- ningún valor o 1. Extrae el primer subpatrón de la expresión regular.- 2. Extrae el segundo subpatrón de la expresión regular.- n. Extrae el <i>enésimo</i> subpatrón de la expresión regular. El valor predeterminado es 1.

Uso de la sintaxis de expresión regular compatible con perl

Debe utilizar la sintaxis de expresión regular compatible con perl con las funciones REG_EXTRACT, REG_MATCH y REG_REPLACE.

La siguiente tabla ofrece las directrices de la sintaxis de expresión regular compatible con perl.

Sintaxis	Descripción
.	(un punto) Coincide con cualquier carácter.
[a-z]	Coincide con una instancia de un carácter en minúsculas. Por ejemplo, [a-z] coincide con ab. Utilice [A-Z] para que coincidan caracteres en mayúsculas.
\d	Coincide con una instancia de cualquier dígito de 0-9.
\s	Coincide con un carácter de espacio en blanco.
\w	Coincide con un carácter alfanumérico, incluido el subrayado (_)
()	Agrupar una expresión. Por ejemplo, los paréntesis en (\d-\d-\d\d) agrupan la expresión \d\d-\d\d, que busca todos los grupos de dos números seguidos de un guión y dos números más, como 12-34.
{}	Coincide con el número de caracteres. Por ejemplo, \d{3} coincide con tres números, como 650 ó 510. O, [a-z]{2} coincide con dos letras, como CA o NY.
?	Coincide con el carácter precedente o con un grupo de caracteres ninguna o una vez. Por ejemplo, \d{3}(-\d{4})? coincide con tres números que pueden ir seguidos de un guión y otros cuatro números.
*	(un asterisco) Coincide con ninguna o más instancias de los valores que siguen al asterisco. Por ejemplo, *0 es un valor que precede a 0.
+	Coincide con una o más instancias de los valores que siguen al signo más. Por ejemplo, \w+ es cualquier valor que sigue a un carácter alfanumérico.

Por ejemplo, la siguiente expresión regular busca códigos postales de EE. UU. de cinco dígitos, como 93930 y códigos postales de 9 dígitos, como 93930-5407:

```
\d{5}(-\d{4})?
```

\d{5} hace referencia a cinco números, como 93930. Los paréntesis que rodean a -\d{4} agrupan este segmento de la expresión. El guión representa el guión de un código postal de 9 dígitos, como 93930-5407. \d{4} hace referencia a cuatro números, como 5407. El signo de interrogación indica que el guión y los últimos cuatro dígitos son opcionales o que pueden aparecer una vez.

Conversión de la sintaxis de COBOL a la sintaxis de expresión regular compatible con perl.

Si está familiarizado con la sintaxis de COBOL, puede utilizar la siguiente información para escribir expresiones regulares compatibles con perl.

La siguiente tabla muestra ejemplos de la sintaxis de COBOL y de sus equivalentes en perl:

Sintaxis de COBOL	Sintaxis de perl	Descripción
9	\d	Coincide con una instancia de cualquier dígito de 0-9.
9999	\d\d\d\d o \d{4}	Coincide con cuatro dígitos entre 0-9, como 1234 ó 5936.

Sintaxis de COBOL	Sintaxis de perl	Descripción
x	[a-z]	Coincide con una instancia de una letra.
9xx9	\d[a-z][a-z]\d	Coincide con cualquier número seguido por dos letras y otro número, como 1ab2.

Conversión de la sintaxis de SQL a la sintaxis de expresión regular compatible con perl

Si está familiarizado con la sintaxis de SQL, puede utilizar la siguiente información para escribir expresiones regulares compatibles con perl.

La siguiente tabla muestra algunos ejemplos de la sintaxis de SQL y de sus equivalentes en perl:

Sintaxis de SQL	Sintaxis de perl	Descripción
%	. *	Coincide con cualquier cadena.
A%	A. *	Coincide con la letra "A" seguida de cualquier cadena, como Area (en inglés).
_	. (un punto)	Coincide con cualquier carácter.
A_	A.	Coincide con "A" seguida de cualquier carácter, como AZ.

Valor de devolución

Devuelve el valor del *enésimo* subpatrón que forma parte del valor de entrada. El *enésimo* subpatrón se basa en el valor que especifique para subPatternNum.

NULL si la entrada es un valor nulo o si el patrón es nulo.

Ejemplo

Puede utilizar REG_EXTRACT en una expresión para extraer de una expresión regular segundos nombres que coincidan con el nombre, el segundo nombre y el apellido. Por ejemplo, la siguiente expresión devuelve el segundo nombre de una expresión regular:

```
REG_EXTRACT( Employee_Name, '(\w+)\s+(\w+)\s+(\w+)', 2)
```

Employee_Name	Return Value
Stephen Graham Smith	Graham
Juan Carlos Fernando	Carlos

REG_MATCH

Devuelve si un valor coincide con un patrón de expresión regular. Permite validar patrones de datos como números de identificación, números de teléfono, códigos postales y nombres de estado.

Nota: Utilice la función REG_REPLACE para sustituir un patrón de caracteres en una cadena con un nuevo patrón de caracteres.

Sintaxis

```
REG_MATCH( subject, pattern )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>subject</i>	Obligatorio	Tipo de datos String. Pasa el valor que se desea hacer coincidir respecto al patrón de la expresión regular.
<i>pattern</i>	Obligatorio	Tipo de datos String. Patrón de expresión regular con el que se desea coincidir. Debe utilizar una sintaxis de expresiones regulares compatible con perl. Delimite el patrón mediante comillas simples. Para más información, véase "REG_EXTRACT" en la página 166 .

Valor de retorno

TRUE si los datos coinciden con el patrón.

FALSE si los datos no coinciden con el patrón.

NULL si la entrada es un valor nulo o si el patrón es NULL.

Ejemplo

Puede usar REG_MATCH en una expresión para validar números de teléfono. Por ejemplo, la siguiente expresión hace coincidir un número de teléfono de 10 dígitos respecto al patrón y devuelve un valor booleano basado en la coincidencia:

```
REG_MATCH (Phone_Number, '(\d\d\d\d\d\d\d\d\d\d) ' )
```

Phone_Number	Return Value
408-555-1212	TRUE
	NULL
510-555-1212	TRUE
92 555 51212	FALSE
650-555-1212	TRUE
415-555-1212	TRUE
831 555 12123	FALSE

Consejo

También puede usar REG_MATCH para las siguientes tareas:

- Para verificar que un valor coincide con un patrón. Se utiliza de forma similar a la función SQL LIKE.
- Para verificar que los valores son caracteres. Se utiliza de forma similar a la función SQL IS_CHAR.

Para verificar que un valor coincide con un patrón, utilice un punto (.) y un asterisco (*) con la función REG_MATCH en una expresión. Un punto coincide con un carácter cualquiera. Un asterisco coincide con 0 o más instancias de valores que le suceden.

Por ejemplo, utilice la siguiente expresión para encontrar números de cuenta que empiezan por 1835:

```
REG_MATCH (ACCOUNT_NUMBER, '1835.*')
```

Para verificar que los valores son caracteres, utilice una función REG_MATCH con la expresión regular [a-zA-Z]+. a-z hace coincidir todos los caracteres en minúscula. A-Z hace coincidir todos los caracteres en mayúscula. El signo más (+) indica que debe haber un carácter como mínimo.

Por ejemplo, utilice la siguiente expresión para comprobar que una lista de apellidos solamente contiene caracteres.

```
REG_MATCH (LAST_NAME, '[a-zA-Z]+')
```

REG_REPLACE

Reemplaza caracteres en una cadena con otro patrón de caracteres. De forma predeterminada, REG_REPLACE busca en la cadena de entrada el patrón de caracteres que se haya especificado y reemplaza todas las apariciones con el patrón de reemplazo. También puede indicar el número de apariciones del patrón que desea reemplazar en la cadena.

Sintaxis

```
REG_REPLACE( subject, pattern, replace, numReplacements )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>subject</i>	Obligatorio	Tipo de datos String. Pasa la cadena que desea buscar.
<i>pattern</i>	Obligatorio	Tipo de datos String. Pasa la cadena de carácter que se quiere reemplazar. Debe utilizar la sintaxis de expresión regular compatible con Perl. Incluya el patrón entre comillas simples. Para obtener más información, consulte "REG_EXTRACT" en la página 166 .
<i>replace</i>	Obligatorio	Tipo de datos String. Pasa la cadena del nuevo carácter.
<i>numReplacements</i>	Opcional	Tipo de datos numérico. Especifica el número de apariciones que desea reemplazar. Si omite esta opción, REG_REPLACE reemplazará todas las apariciones de la cadena de caracteres.

Valor de retorno

Cadena

Ejemplo

La expresión siguiente elimina espacios adicionales de los datos del nombre del empleado para cada fila del puerto `Employee_name`:

```
REG_REPLACE( Employee_Name, '\s+', ' ' )
```

Employee_Name	RETURN VALUE
Adam Smith	Adam Smith
Greg Sanders	Greg Sanders
Sarah Fe	Sarah Fe
Sam Cooper	Sam Cooper

REPLACECHR

Sustituye caracteres de una cadena por un carácter único o por ningún carácter. REPLACECHR busca en la cadena de entrada los caracteres especificados y sustituye todas las ocurrencias de todos los caracteres con el nuevo carácter especificado.

Sintaxis

```
REPLACECHR( CaseFlag, InputString, OldCharSet, NewChar )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>CaseFlag</i>	Obligatorio	El valor debe ser un entero. Determina si en los argumentos de esta función se distingue entre mayúsculas y minúsculas. Se puede especificar cualquier expresión de transformación válida. Cuando <i>CaseFlag</i> es un número distinto de 0, la función distingue entre mayúsculas y minúsculas. Cuando <i>CaseFlag</i> es un valor nulo o es 0, la función no distingue entre mayúsculas y minúsculas.
<i>InputString</i>	Obligatorio	Debe ser una cadena de caracteres. Pasa la cadena que se desea buscar. Se puede especificar cualquier expresión de transformación válida. Si se pasa un valor numérico, la función la convierte en una cadena de caracteres. Si <i>InputString</i> es NULL, REPLACECHR devuelve NULL.

Argumento	Obligatorio / Opcional	Descripción
<i>OldCharSet</i>	Obligatorio	Debe ser una cadena de caracteres. Los caracteres que se desea sustituir. Se puede introducir uno o más caracteres. Se puede especificar cualquier expresión de transformación válida. También puede especificar un literal de texto encerrado entre comillas simples, por ejemplo, 'abc'. Si se pasa un valor numérico, la función la convierte en una cadena de caracteres. Si <i>OldCharSet</i> es NULL o está vacío, REPLACECHR devuelve <i>InputString</i> .
<i>NewChar</i>	Obligatorio	Debe ser una cadena de caracteres. Puede introducir un carácter, una cadena vacía o NULL. Se puede especificar cualquier expresión de transformación válida. Si <i>NewChar</i> es NULL o está vacío, REPLACECHR elimina todas las ocurrencias de todos los caracteres de <i>OldCharSet</i> en <i>InputString</i> . Si <i>NewChar</i> contiene más de un carácter, REPLACECHR utiliza el primer carácter para sustituir <i>OldCharSet</i> .

Valor de devolución

Cadena.

Cadena vacía si REPLACECHR elimina todos los caracteres de *InputString*.

NULL si *InputString* es NULL.

InputString si *OldCharSet* es NULL o está vacío.

Ejemplos

La siguiente expresión elimina las comillas dobles de los datos de registro de web para cada fila en el puerto WEBLOG:

```
REPLACECHR( 0, WEBLOG, '"', NULL )
```

WEBLOG	RETURN VALUE
"GET /news/index.html HTTP/1.1"	GET /news/index.html HTTP/1.1
"GET /companyinfo/index.html HTTP/1.1"	GET /companyinfo/index.html HTTP/1.1
GET /companyinfo/index.html HTTP/1.1	GET /companyinfo/index.html HTTP/1.1
NULL	NULL

La siguiente expresión elimina múltiples caracteres para cada fila en el puerto WEBLOG:

```
REPLACECHR ( 1, WEBLOG, ']['"', NULL )
```

WEBLOG	RETURN VALUE
[29/Oct/2001:14:13:50 -0700]	29/Oct/2001:14:13:50 -0700
[31/Oct/2000:19:45:46 -0700] "GET /news/index.html HTTP/1.1"	31/Oct/2000:19:45:46 -0700 GET /news/index.html HTTP/1.1

WEBLOG	RETURN VALUE
[01/Nov/2000:10:51:31 -0700] "GET /news/index.html HTTP/1.1"	01/Nov/2000:10:51:31 -0700 GET /news/index.html HTTP/1.1
NULL	NULL

La siguiente expresión cambia parte del valor del código de cliente para cada fila en el puerto CUSTOMER_CODE:

```
REPLACECHR ( 1, CUSTOMER_CODE, 'A', 'M' )
```

CUSTOMER_CODE	RETURN VALUE
ABA	MBM
abA	abM
BBC	BBC
ACC	MCC
NULL	NULL

La siguiente expresión cambia parte del valor del código de cliente para cada fila en el puerto CUSTOMER_CODE:

```
REPLACECHR ( 0, CUSTOMER_CODE, 'A', 'M' )
```

CUSTOMER_CODE	RETURN VALUE
ABA	MBM
abA	MbM
BBC	BBC
ACC	MCC

La siguiente expresión cambia parte del valor del código de cliente para cada fila en el puerto CUSTOMER_CODE:

```
REPLACECHR ( 1, CUSTOMER_CODE, 'A', NULL )
```

CUSTOMER_CODE	RETURN VALUE
ABA	B
BBC	BBC
ACC	CC
AAA	[empty string]

CUSTOMER_CODE	RETURN VALUE
aaa	aaa
NULL	NULL

La siguiente expresión elimina múltiples caracteres para cada fila en el puerto INPUT:

```
REPLACECHR ( 1, INPUT, '14', NULL )
```

INPUT	RETURN VALUE
12345	235
4141	NULL
111115	5
NULL	NULL

Cuando se desea usar una comilla simple (') en *OldCharSet* o *NewChar*, debe utilizarse la función CHR. La comilla simple es el único carácter que no se puede usar dentro de un literal de cadena.

La siguiente expresión elimina múltiples caracteres, incluidas las comillas simples, para cada fila en el puerto INPUT:

```
REPLACECHR (1, INPUT, CHR(39), NULL )
```

INPUT	RETURN VALUE
'Tom Smith' 'Laura Jones'	Tom Smith Laura Jones
Tom's	Toms
NULL	NULL

REPLACETR

Sustituye caracteres de una cadena por un carácter único, por múltiples caracteres o por ningún carácter. REPLACETR busca la cadena de entrada en todas las cadenas que indique y la sustituye por la nueva cadena que especifique.

Sintaxis

```
REPLACETR ( CaseFlag, InputString, OldString1, [OldString2, ... OldStringN,] NewString )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>CaseFlag</i>	Obligatorio	<p>Debe ser un entero. Determina si en los argumentos de esta función se distingue entre mayúsculas y minúsculas. Puede especificar cualquier expresión de transformación válida.</p> <p>Cuando <i>CaseFlag</i> es un número distinto de 0, la función distingue entre mayúsculas y minúsculas.</p> <p>Cuando <i>CaseFlag</i> es un valor nulo o 0, la función no distingue entre mayúsculas y minúsculas.</p>
<i>InputString</i>	Obligatorio	<p>Debe ser una cadena de caracteres. Transfiere las cadenas que desea buscar. Puede especificar cualquier expresión de transformación válida. Si transfiere un valor numérico, la función lo convierte en una cadena de caracteres.</p> <p>Si <i>InputString</i> es NULL, REPLACESTR devuelve NULL.</p>
<i>OldString</i>	Obligatorio	<p>Debe ser una cadena de caracteres. La cadena que desea sustituir. Debe especificar como mínimo, un argumento <i>OldString</i>. Para cada argumento <i>OldString</i>, puede especificar uno o más caracteres. Puede especificar cualquier expresión de transformación válida. También puede especificar un literal de texto encerrado entre comillas simples, por ejemplo, 'abc'.</p> <p>Si transfiere un valor numérico, la función lo convierte en una cadena de caracteres.</p> <p>Cuando REPLACESTR contiene varios argumentos <i>OldString</i> y uno o más argumentos <i>OldString</i> son NULL o están vacíos, REPLACESTR no considera el argumento <i>OldString</i>. Cuando todos los argumentos <i>OldString</i> son NULL o están vacíos, REPLACESTR devuelve <i>InputString</i>.</p> <p>La función sustituye los caracteres de los argumentos <i>OldString</i> en el orden en el que aparecen en la función. Por ejemplo, si sustituye varios argumentos <i>OldString</i>, el primer argumento <i>OldString</i> tiene prioridad sobre el segundo argumento <i>OldString</i>, y el segundo argumento <i>OldString</i> tiene prioridad sobre el tercer argumento <i>OldString</i>. Cuando REPLACESTR sustituye una cadena, sitúa el cursor tras los caracteres sustituidos en <i>InputString</i> antes de buscar la siguiente coincidencia.</p>
<i>NewString</i>	Obligatorio	<p>Debe ser una cadena de caracteres. Puede especificar un carácter, varios caracteres, una cadena vacía o NULL. Puede especificar cualquier expresión de transformación válida.</p> <p>Si <i>NewString</i> es NULL o está vacía, REPLACESTR quita todas las ocurrencias de <i>OldString</i> en <i>InputString</i>.</p>

Valor de devolución

Cadena.

Cadena vacía si REPLACESTR quita todos los caracteres de *InputString*.

NULL si *InputString* es NULL.

InputString si todos los argumentos *OldString* son NULL o están vacíos.

Ejemplos

La siguiente expresión quita las comillas dobles y dos cadenas de texto diferentes de los datos del registro web de cada fila del puerto WEBLOG:

```
REPLACESTR ( 1, WEBLOG, '"', 'GET ', ' HTTP/1.1', NULL )
```

WEBLOG	RETURN VALUE
"GET /news/index.html HTTP/1.1"	/news/index.html
"GET /companyinfo/index.html HTTP/1.1"	/companyinfo/index.html
GET /companyinfo/index.html	/companyinfo/index.html
GET	[empty string]
NULL	NULL

La siguiente expresión cambia el título para determinados valores de cada fila del puerto TITLE:

```
REPLACESTR ( 1, TITLE, 'rs.', 'iss', 's.' )
```

TITLE	RETURN VALUE
Mrs.	Ms.
Miss	Ms.
Mr.	Mr.
MRS.	MRS.

La siguiente expresión cambia el título para determinados valores de cada fila del puerto TITLE:

```
REPLACESTR ( 0, TITLE, 'rs.', 'iss', 's.' )
```

TITLE	RETURN VALUE
Mrs.	Ms.
MRS.	Ms.

La siguiente expresión muestra cómo la función REPLACESTR sustituye varios argumentos `OldString` de cada fila en el puerto INPUT:

```
REPLACESTR ( 1, INPUT, 'ab', 'bc', '*' )
```

INPUT	RETURN VALUE
abc	*c
abbc	**
abbbbc	*bb*
bc	*

La siguiente expresión muestra cómo la función REPLACESTR sustituye varios argumentos *OldString* de cada fila en el puerto INPUT:

```
REPLACESTR ( 1, INPUT, 'ab', 'bc', 'b' )
```

INPUT	RETURN VALUE
ab	b
bc	b
abc	bc
abbc	bb
abbcc	bbc

Cuando quiera utilizar comillas simples (') en *OldString* o *NewString*, debe utilizar la función CHR. Utilice las funciones CHR y CONCAT para concatenar unas comillas simples en una cadena. Las comillas simples son el único carácter que no se puede utilizar dentro de un literal de cadena. Observe el siguiente ejemplo:

```
CONCAT( 'Joan', CONCAT( CHR(39), 's car' ) )
```

El valor de devolución es:

```
Joan's car
```

La siguiente expresión cambia una cadena que incluye unas comillas simples de las filas del puerto INPUT:

```
REPLACESTR ( 1, INPUT, CONCAT('it', CONCAT(CHR(39), 's' )), 'its' )
```

INPUT	RETURN VALUE
it's	its
mit's	mits
mits	mits
mits'	mits'

RESPEC

Cambia el nombre de cada elemento en el valor de estructura dado basándose en los nombres de los elementos en la definición de tipo de datos complejos especificada.

Sintaxis

```
RESPEC (:Type.type_definition_library.type_definition, struct_value)
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio/ Opcional	Descripción
:Type.type_definition_library.type_definition	Obligatorio	La definición de tipo de datos complejos que representa el esquema de los datos de estructura. Utilice el calificador de referencia :Type para hacer referencia a la biblioteca de definiciones de tipos que contiene la definición de tipo de datos complejos.
struct_value	Obligatorio	El valor de estructura para el que desea cambiar los nombres de elemento. Se puede introducir cualquier expresión de transformación válida que dé como resultado una estructura.

El tipo de datos de cada elemento de la definición de tipo de datos complejos debe coincidir con el tipo de datos del elemento correspondiente de la estructura.

Valor de devolución

Estructura.

Ejemplos

La siguiente expresión cambia los nombres de los elementos del puerto de estructura h2_sales en función de los tipos de datos de la definición de tipo de datos complejos h1_sales_def.

```
RESPEC(:Type.type_definition_library.h2_sales_def, h2_sales)
```

h2_sales_def	h2_sales	RETURN VALUE
{ q1_sales : int q2_sales : bigint }	{ q3_total : int q4_total : bigint }	{ q1_sales : int q2_sales : bigint }

REVERSE

Invierte la cadena de entrada.

Sintaxis

```
REVERSE( string )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio/ Opcional	Descripción
string	Obligatorio	Cualquier valor de carácter. Es el valor que se va a invertir.

Valor devuelto

Cadena. Inversión del valor de entrada.

Ejemplo

La siguiente expresión invierte los números del código de cliente:

```
REVERSE ( CUSTOMER_CODE )
```

CUSTOMER_CODE	RETURN VALUE
0001	1000
0002	2000
0003	3000
0004	4000

ROUND (Fechas)

Redondea una parte de una fecha. También puede utilizar ROUND para redondear números.

Esta función puede redondear las siguientes partes de una fecha:

Año

Redondea la parte del año de una fecha en función del mes.

Mes

Redondea la parte del mes de una fecha en función del día del mes.

Día

Redondea la parte del día de una fecha en función de la hora.

Hora

Redondea la parte de la hora de una fecha en función de los minutos de una hora.

Minuto

Redondea la parte del minuto de una fecha en función de los segundos.

Segundo

Redondea la parte del segundo de una fecha en función de los milisegundos.

Milisegundo

Redondea la parte del milisegundo de una fecha en función de los microsegundos.

Microsegundo

Redondea la parte del microsegundo de una fecha en función de los nanosegundos.

La siguiente tabla muestra las condiciones que utiliza la expresión ROUND y los valores de devolución:

Condición	Expresión	Valor de devolución
Un mes entre enero y junio, la función devuelve 1 de enero del mismo año y establece la hora en 00:00:00.000000000.	ROUND(TO_DATE('04/16/1998 8:24:19', 'MM/DD/YYYY HH24:MI:SS'), 'YY')	01/01/1998 00:00:00.000000000
Un mes entre julio y diciembre, la función devuelve 1 de enero del año siguiente y establece la hora en 00:00:00.000000000.	ROUND(TO_DATE('07/30/1998 2:30:55', 'MM/DD/YYYY HH24:MI:SS'), 'YY')	01/01/1999 00:00:00.000000000
Un día del mes entre 1 y 15, la función devuelve el primer día del mes de entrada y establece la hora en 00:00:00.000000000.	ROUND(TO_DATE('04/15/1998 8:24:19', 'MM/DD/YYYY HH24:MI:SS'), 'MM')	04/01/1998 00:00:00.000000000
Un día del mes entre el 16 y el último día del mes, la función devuelve el primer día del mes siguiente y establece la hora en 00:00:00.000000000.	ROUND(TO_DATE('05/22/1998 10:15:29', 'MM/DD/YYYY HH24:MI:SS'), 'MM')	06/01/1998 00:00:00.000000000
La hora entre 00:00:00 (12 a.m.) y 11:59:59 a.m., la función devuelve la fecha actual y establece la hora en 00:00:00.000000000 (12 a.m.).	ROUND(TO_DATE('06/13/1998 2:30:45', 'MM/DD/YYYY HH24:MI:SS'), 'DD')	06/13/1998 00:00:00.000000000
La hora 12:00:00 (12 p.m.) o más tarde, la función redondea la fecha hasta el día siguiente y establece la hora en 00:00:00.000000000 (12 a.m.).	ROUND(TO_DATE('06/13/1998 22:30:45', 'MM/DD/YYYY HH24:MI:SS'), 'DD')	06/14/1998 00:00:00.000000000
La parte de minuto de la hora entre 0 y 29 minutos, la función devuelve la hora actual y establece los minutos, los segundos, los milisegundos y los nanosegundos en 0.	ROUND(TO_DATE('04/01/1998 11:29:35', 'MM/DD/YYYY HH24:MI:SS'), 'HH')	04/01/1998 11:00:00.000000000
La parte de minuto de la hora entre 30 o más, la función devuelve la hora siguiente y establece los minutos, los segundos, los milisegundos y los nanosegundos en 0.	ROUND(TO_DATE('04/01/1998 13:39:00', 'MM/DD/YYYY HH24:MI:SS'), 'HH')	04/01/1998 14:00:00.000000000
La hora entre 0 y 29 segundos, la función devuelve el minuto actual y establece los segundos, los milisegundos y los nanosegundos en 0.	ROUND(TO_DATE('05/22/1998 10:15:29', 'MM/DD/YYYY HH24:MI:SS'), 'MI')	05/22/1998 10:15:00.000000000
La hora entre 30 y 59 segundos, la función devuelve el minuto siguiente y establece los segundos, los milisegundos, y los nanosegundos en 0.	ROUND(TO_DATE('05/22/1998 10:15:30', 'MM/DD/YYYY HH24:MI:SS'), 'MI')	05/22/1998 10:16:00.000000000
La hora entre los 0 y 499 milisegundos, la función devuelve el segundo actual y establece los milisegundos en 0.	ROUND(TO_DATE('05/22/1998 10:15:29.499', 'MM/DD/YYYY HH24:MI:SS.MS'), 'SS')	05/22/1998 10:15:29.000000000
La hora entre los 500 y 999 milisegundos, la función devuelve el segundo siguiente y establece los milisegundos en 0.	ROUND(TO_DATE('05/22/1998 10:15:29.500', 'MM/DD/YYYY HH24:MI:SS.MS'), 'SS')	05/22/1998 10:15:30.000000000

Condición	Expresión	Valor de devolución
La hora entre los 0 y 499 microsegundos, la función devuelve el milisegundo actual y establece los microsegundos en 0.	ROUND(TO_DATE('05/22/1998 10:15:29.498125','MM/DD/YYYY HH24:MI:SS.US'),'MS')	05/22/1998 10:15:29.498000000
La hora entre los 500 y 999 microsegundos, la función devuelve el milisegundo siguiente y establece los microsegundos en 0.	ROUND(TO_DATE('05/22/1998 10:15:29.498785','MM/DD/YYYY HH24:MI:SS.US'),'MS')	05/22/1998 10:15:29.499000000
La hora entre los 0 y 499 nanosegundos, la función devuelve el microsegundo actual y establece los nanosegundos en 0.	ROUND(TO_DATE('05/22/1998 10:15:29.498125345','MM/DD/YYYY HH24:MI:SS.NS'),'US')	05/22/1998 10:15:29.498125000
La hora entre los 500 y 999 nanosegundos, la función devuelve el microsegundo siguiente y establece los nanosegundos en 0.	ROUND(TO_DATE('05/22/1998 10:15:29.498125876','MM/DD/YYYY HH24:MI:SS.NS'),'US')	05/22/1998 10:15:29.498126000

Sintaxis

```
ROUND( date [,format] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>date</i>	Obligatorio	Tipo de dato de fecha y hora. Puede anidar TO_DATE para convertir cadenas en fechas antes de redondear.
<i>format</i>	Opcional	Especifique una cadena de formato válida. Es la parte de la fecha que desea redondear. Solamente se puede redondear una parte de la fecha. Si omite la cadena de formato, la función redondea la fecha a la fecha más cercana.

Valor de devolución

Fecha con la parte especificada redondeada. ROUND devuelve una fecha en el mismo formato que la fecha de origen. Puede vincular los resultados de esta función a cualquier puerto con un tipo de datos de fecha y hora.

NULL si transfiere un valor nulo a la función.

Ejemplos

Las siguientes expresiones redondean la parte del año de las fechas del puerto DATE_SHIPPED:

```
ROUND( DATE_SHIPPED, 'Y' )
ROUND( DATE_SHIPPED, 'YY' )
ROUND( DATE_SHIPPED, 'YYY' )
ROUND( DATE_SHIPPED, 'YYYY' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 1 1998 12:00:00.000000000AM

DATE_SHIPPED	RETURN VALUE
Apr 19 1998 1:31:20PM	Jan 1 1998 12:00:00.000000000AM
Dec 20 1998 3:29:55PM	Jan 1 1999 12:00:00.000000000AM
NULL	NULL

Las siguientes expresiones redondean la parte del mes de las fechas del puerto DATE_SHIPPED:

```
ROUND( DATE_SHIPPED, 'MM' )
ROUND( DATE_SHIPPED, 'MON' )
ROUND( DATE_SHIPPED, 'MONTH' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 1 1998 12:00:00.000000000AM
Apr 19 1998 1:31:20PM	May 1 1998 12:00:00.000000000AM
Dec 20 1998 3:29:55PM	Jan 1 1999 12:00:00.000000000AM
NULL	NULL

Las siguientes expresiones redondean la parte del día de las fechas del puerto DATE_SHIPPED:

```
ROUND( DATE_SHIPPED, 'D' )
ROUND( DATE_SHIPPED, 'DD' )
ROUND( DATE_SHIPPED, 'DDD' )
ROUND( DATE_SHIPPED, 'DY' )
ROUND( DATE_SHIPPED, 'DAY' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 15 1998 12:00:00.000000000AM
Apr 19 1998 1:31:20PM	Apr 20 1998 12:00:00.000000000AM
Dec 20 1998 3:29:55PM	Dec 21 1998 12:00:00.000000000AM
Dec 31 1998 11:59:59PM	Jan 1 1999 12:00:00.000000000AM
NULL	NULL

Las siguientes expresiones redondean la parte de la hora de las fechas del puerto DATE_SHIPPED:

```
ROUND( DATE_SHIPPED, 'HH' )
ROUND( DATE_SHIPPED, 'HH12' )
ROUND( DATE_SHIPPED, 'HH24' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:31AM	Jan 15 1998 2:00:00.000000000AM
Apr 19 1998 1:31:20PM	Apr 19 1998 2:00:00.000000000PM
Dec 20 1998 3:29:55PM	Dec 20 1998 3:00:00.000000000PM

DATE_SHIPPED	RETURN VALUE
Dec 31 1998 11:59:59PM	Jan 1 1999 12:00:00.000000000AM
NULL	NULL

Las siguientes expresiones redondean la parte del minuto de las fechas del puerto DATE_SHIPPED:

```
ROUND( DATE_SHIPPED, 'MI' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 15 1998 2:11:00.000000000AM
Apr 19 1998 1:31:20PM	Apr 19 1998 1:31:00.000000000PM
Dec 20 1998 3:29:55PM	Dec 20 1998 3:30:00.000000000PM
Dec 31 1998 11:59:59PM	Jan 1 1999 12:00:00.000000000AM
NULL	NULL

ROUND (Números)

Redondea números hasta un número especificado de dígitos o decimales. También se puede usar ROUND para redondear fechas.

Sintaxis

```
ROUND( numeric_value [, precision] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. Se puede especificar cualquier expresión de transformación válida. Utilice operadores para realizar operaciones aritméticas antes de redondear los valores.
<i>precision</i>	Opcional	<p>Entero positivo o negativo. Si introduce un valor de <i>precision</i> positivo, la función redondea a este número de posiciones decimales. Por ejemplo, ROUND(12.99, 1) devuelve 13.0 y ROUND(15.44, 1) devuelve 15.4.</p> <p>Si se introduce un valor de <i>precision</i> negativo, la función redondea este número a la izquierda del punto decimal, devolviendo un entero. Por ejemplo, ROUND(12.99, -1) devuelve 10 y ROUND(15.99, -1) devuelve 20.</p> <p>Si introduce un valor de <i>precision</i> decimal, la función redondea al entero más próximo antes de evaluar la expresión. Por ejemplo, ROUND(12.99, 0.8) devuelve 13.0 porque la función redondea 0.8 a 1 y luego evalúa la expresión.</p> <p>Si omite el argumento <i>precision</i>, la función redondea al entero más próximo, truncando la porción decimal del número. Por ejemplo, ROUND(12.99) devuelve 13.</p>

Valor de devolución

Valor numérico.

Si uno de los argumentos es NULL, ROUND devuelve NULL.

Nota: Si el valor devuelto es decimal con una precisión superior a 15, puede habilitar la alta precisión para garantizar una precisión decimal de hasta 38 dígitos.

Ejemplos

La siguiente expresión devuelve los valores en el puerto Price redondeados a tres posiciones decimales:

```
ROUND( PRICE, 3 )
```

PRICE	RETURN VALUE
12.9936	12.994
15.9949	15.995
-18.8678	-18.868
56.9561	56.956
NULL	NULL

Se pueden redondear dígitos a la izquierda del punto decimal pasando un entero negativo en el argumento *precision*:

```
ROUND( PRICE, -2 )
```

PRICE	RETURN VALUE
13242.99	13200.0

PRICE	RETURN VALUE
1435.99	1400.0
-108.95	-100.0
NULL	NULL

Si se pasa un valor decimal en el argumento *precision*, el Servicio de integración de datos lo redondea al entero más próximo antes de evaluar la expresión:

```
ROUND( PRICE, 0.8 )
```

PRICE	RETURN VALUE
12.99	13.0
56.34	56.3
NULL	NULL

Si omite el argumento *precision*, la función efectúa el redondeo al entero más próximo:

```
ROUND( PRICE )
```

PRICE	RETURN VALUE
12.99	13.0
-15.99	-16.0
-18.99	-19.0
56.95	57.0
NULL	NULL

Consejo

También se puede usar ROUND para establecer de forma explícita la precisión de los valores calculados y obtener resultados esperados. Cuando el Servicio de integración de datos se ejecuta en modo de precisión baja, trunca el resultado de los cálculos cuando la precisión del valor excede 15 dígitos. Por ejemplo, quizá desee procesar la siguiente expresión en modo de precisión baja:

```
7/3 * 3 = 7
```

En este caso, el Servicio de integración de datos evalúa el lado izquierdo de la expresión como 6,999999999999999 porque trunca el resultado de la primera operación de división. El Servicio de integración de datos evalúa la expresión completa como FALSE. Es posible que este no sea el resultado esperado.

Para lograr el resultado esperado, utilice ROUND para redondear el resultado truncado de la izquierda de la expresión al resultado esperado. El Servicio de integración de datos evalúa la siguiente expresión como TRUE:

```
ROUND(7/3 * 3) = 7
```

RPAD

Convierte una cadena a una longitud especificada añadiendo espacios en blanco o caracteres al final de la cadena.

Sintaxis

```
RPAD( first_string, length [,second_string] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>first_string</i>	Obligatorio	Cualquier valor de cadena. Cadenas que desea cambiar. Puede introducir cualquier expresión de transformación válida.
<i>length</i>	Obligatorio	Debe ser un literal entero positivo. Especifica la longitud que desea que tenga cada cadena.
<i>second_string</i>	Opcional	Cualquier valor de cadena. Pase la cadena que desea añadir a la derecha de los valores <i>first_string</i> . Incluya los caracteres que desea añadir al final de la cadena entre comillas simples, por ejemplo, 'abc'. Este argumento distingue entre mayúsculas y minúsculas. Si se omite la segunda cadena, la función rellena el final de la primera cadena con espacios en blanco.

Valor de retorno

Cadena de la longitud especificada.

NULL si el valor pasado a la función es NULL o si la longitud es un número negativo.

Ejemplos

La expresión siguiente devuelve el nombre del elemento con una longitud de 16 caracteres, añadiendo la cadena '.' al final de cada nombre de elemento:

```
RPAD( ITEM_NAME, 16, '.')
```

ITEM_NAME	RETURN VALUE
Flashlight	Flashlight.....
Compass	Compass.....
Regulator System	Regulator System
Safety Knife	Safety Knife....

RPAD cuenta la longitud de izquierda a derecha. Por lo tanto, si la primera cadena es más larga que la longitud, RPAD trunca la cadena de derecha a izquierda. Por ejemplo, RPAD ('alfabético', 5 'x') devolverá la cadena 'alfab'. RPAD utiliza una parte de la *second_string* si es necesario.

La expresión siguiente devuelve el nombre del elemento con una longitud de 16 caracteres, añadiendo la cadena '*' al final de cada nombre de elemento:

```
RPAD( ITEM_NAME, 16, '*' )
```

ITEM_NAME	RETURN VALUE
Flashlight	Flashlight*.*.*.
Compass	Compass*.*.*.*.*
Regulator System	Regulator System
Safety Knife	Safety Knife*.*

RTRIM

Quita espacios en blanco o caracteres del final de una cadena.

Si no especifica un parámetro *trim_set* en la expresión:

- En UNICODE, RTRIM quita los espacios de uno y dos bytes del final de una cadena
- En ASCII, RTRIM quita solamente los espacios de un byte.

Si utiliza RTRIM para quitar los caracteres de una cadena, RTRIM compara, carácter por carácter, el argumento *trim_set* con cada carácter del argumento *string*, empezando por la derecha de la cadena. Si el carácter de la cadena coincide con cualquier carácter de *trim_set*, RTRIM lo quita. RTRIM sigue la comparación y quita los caracteres hasta que no puede encontrar un carácter coincidente en *trim_set*. Devuelve la cadena sin los caracteres coincidentes.

Sintaxis

```
RTRIM( string [, trim_set] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>string</i>	Obligatorio	Cualquier valor de cadena. Transfiere los valores que desea recortar. Puede especificar cualquier expresión de transformación válida. Utilice operadores para efectuar las comparaciones o para concatenar las cadenas antes de quitar los espacios en blanco del final de una cadena.
<i>trim_set</i>	Opcional	Cualquier valor de cadena. Transfiere los caracteres que desea quitar del final de la cadena. También puede especificar un literal de texto. Sin embargo, debe encerrar entre comillas simples los caracteres que desea quitar del final de la cadena, por ejemplo, 'abc'. Si omite la segunda cadena, la función quita los espacios en blanco del final de la primera cadena. RTRIM distingue entre mayúsculas y minúsculas.

Valor de devolución

Cadena. Se quitan los valores de cadena con los caracteres especificados en el argumento *trim_set*.

NULL si el valor que se ha transferido a la función es NULL.

Ejemplo:

La siguiente expresión quita los caracteres 're' de las cadenas del puerto LAST_NAME:

```
RTRIM( LAST_NAME, 're')
```

LAST_NAME	RETURN VALUE
Nelson	Nelson
Page	Pag
Osborne	Osborn
NULL	NULL
Sawyer	Sawy
H. Bender	H. Bend
Steadman	Steadman

RTRIM quita 'e' del término Page aunque 'r' sea el primer carácter del argumento *trim_set*. El motivo es que RTRIM busca, carácter por carácter, el conjunto de caracteres especificado en el argumento *trim_set*. Si el último carácter de la cadena coincide con el primer carácter de *trim_set*, RTRIM lo quita. Sin embargo, si el último carácter de la cadena no coincide, RTRIM compara el segundo carácter del argumento *trim_set*. Si el segundo carácter por el final de la cadena coincide con el segundo carácter del argumento *trim_set*, RTRIM lo quita, y así sucesivamente. Cuando el carácter de la cadena no coincide con *trim_set*, RTRIM devuelve la cadena y verifica la siguiente fila.

En el último ejemplo, el último carácter del término Nelson no coincide con ninguno de los caracteres del argumento *trim_set*, así que RTRIM devuelve la cadena 'Nelson' y analiza la siguiente fila.

Consejos para RTRIM

Utilice RTRIM y LTRIM con || o CONCAT para quitar los espacios en blanco al principio o al final después de concatenar dos cadenas.

Si anida RTRIM, también puede quitar varios conjuntos de caracteres. Por ejemplo, si desea quitar los espacios en blanco al final y el carácter 't' del final de cada cadena en una columna de nombres, puede crear una expresión similar a la siguiente:

```
RTRIM( RTRIM( NAMES ), 't' )
```

SET_DATE_PART

Establece una parte del valor Fecha/Hora en el valor que usted especifique. Con SET_DATE_PART, puede cambiar las siguientes partes de una fecha:

- **Año.** Puede cambiar el año si especifica un entero positivo en el argumento *value*. Utilice una de las siguientes cadenas de formato de año: Y, YY, YYYY o YYYY para establecer el año. La siguiente expresión, por ejemplo, cambia el año a 2001 para todas las fechas del puerto SHIP_DATE:

```
SET_DATE_PART( SHIP_DATE, 'YY', 2001 )
```

- **Mes.** Puede cambiar el mes si especifica un entero positivo entre 1 y 12 (enero=1 y diciembre=12) en el argumento *value*. Utilice una de las siguientes cadenas de formato de mes: MM, MON, MONTH para establecer el mes. La siguiente expresión, por ejemplo, cambia el mes a octubre para todas las fechas del puerto SHIP_DATE:

```
SET_DATE_PART( SHIP_DATE, 'MONTH', 10 )
```

- **Día.** Puede cambiar el día si especifica un entero positivo entre el 1 y el 31 (excepto en los meses con menos de 31 días: febrero, abril, junio, setiembre y noviembre) en el argumento *value*. Utilice una de las siguientes cadenas de formato de mes (D, DD, DDD, DY y DAY) para establecer el día. La siguiente expresión, por ejemplo, cambia el día a 10 para todas las fechas del puerto SHIP_DATE:

```
SET_DATE_PART( SHIP_DATE, 'DD', 10 )
```

- **Hora.** Puede cambiar la hora si especifica un entero positivo entre 0 y 24 (donde 0=12 AM 12=12 PM y 24 =12 AM) en el argumento *value*. Utilice una de las cadenas de formato de hora (HH, HH12, HH24) para establecer la hora. La siguiente expresión, por ejemplo, cambia la hora a 14:00:00 (o 2:00:00 PM) para todas las fechas del puerto SHIP_DATE:

```
SET_DATE_PART( SHIP_DATE, 'HH', 14 )
```

- **Minuto.** Puede cambiar los minutos si especifica un entero positivo entre 0 y 59 en el argumento *value*. Utilice la cadena de formato MI para establecer el minuto. La siguiente expresión, por ejemplo, cambia el minuto a 25 para todas las fechas del puerto SHIP_DATE:

```
SET_DATE_PART( SHIP_DATE, 'MI', 25 )
```

- **Segundos.** Puede cambiar los segundos si especifica un entero positivo entre 0 y 59 en el argumento *value*. Utilice la cadena de formato SS para establecer el segundo. La siguiente expresión, por ejemplo, cambia el segundo a 59 para todas las fechas del puerto SHIP_DATE:

```
SET_DATE_PART( SHIP_DATE, 'SS', 59 )
```

- **Milisegundos.** Puede cambiar los milisegundos si especifica un entero positivo entre 0 y 999 en el argumento *value*. Utilice la cadena de formato MS para establecer los milisegundos. La siguiente expresión, por ejemplo, cambia el milisegundo a 125 para todas las fechas del puerto SHIP_DATE:

```
SET_DATE_PART( SHIP_DATE, 'MS', 125 )
```

- **Microsegundos.** Puede cambiar los microsegundos si especifica un entero positivo entre 1000 y 999999 en el argumento *value*. Utilice la cadena de formato US para establecer los microsegundos. La siguiente expresión, por ejemplo, cambia los microsegundos a 12555 para todas las fechas del puerto SHIP_DATE:

```
SET_DATE_PART( SHIP_DATE, 'US', 12555 )
```

- **Nanosegundos.** Puede cambiar los nanosegundos si especifica un entero positivo entre 1000000 y 999999999 en el argumento *value*. Utilice la cadena de formato NS para establecer los nanosegundos. La siguiente expresión, por ejemplo, cambia los nanosegundos a 12555555 para todas las fechas del puerto SHIP_DATE:

```
SET_DATE_PART( SHIP_DATE, 'NS', 12555555 )
```

Sintaxis

```
SET_DATE_PART( date, format, value )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>date</i>	Obligatorio	Tipo de dato de fecha y hora. La fecha que desea modificar. Puede especificar cualquier expresión de transformación válida.
<i>format</i>	Obligatorio	Cadena de formato que especifica la porción de la fecha que se va a modificar. La cadena de formato no distingue entre mayúsculas y minúsculas.
<i>value</i>	Obligatorio	Un valor entero positivo que se va a asignar a la porción especificada de la fecha. El entero debe ser un valor válido para la parte de la fecha que desea cambiar. Si especifica un valor incorrecto como 30 de febrero, se produce un error.

Valor de devolución

La fecha en el mismo formato que la fecha de origen con la parte que se ha especificado cambiada.

NULL si el valor que se ha transferido a la función es NULL.

Ejemplos

Las siguientes expresiones cambian la hora a 4 PM para las fechas del puerto DATE_PROMISED:

```
SET_DATE_PART( DATE_PROMISED, 'HH', 16 )
SET_DATE_PART( DATE_PROMISED, 'HH12', 16 )
SET_DATE_PART( DATE_PROMISED, 'HH24', 16 )
```

DATE_PROMISED	RETURN VALUE
Jan 1 1997 12:15:56AM	Jan 1 1997 4:15:56PM
Feb 13 1997 2:30:01AM	Feb 13 1997 4:30:01PM
Mar 31 1997 5:10:15PM	Mar 31 1997 4:10:15PM
Dec 12 1997 8:07:33AM	Dec 12 1997 4:07:33PM
NULL	NULL

Las siguientes expresiones cambian el mes a junio para las fechas del puerto DATE_PROMISED: El Servicio de integración de datos muestra un error cuando intenta crear una fecha que no existe, como cuando cambia de 31 de marzo a 31 de junio:

```
SET_DATE_PART( DATE_PROMISED, 'MM', 6 )
SET_DATE_PART( DATE_PROMISED, 'MON', 6 )
SET_DATE_PART( DATE_PROMISED, 'MONTH', 6 )
```

DATE_PROMISED	RETURN VALUE
Jan 1 1997 12:15:56AM	Jun 1 1997 12:15:56AM
Feb 13 1997 2:30:01AM	Jun 13 1997 2:30:01AM
Mar 31 1997 5:10:15PM	<i>Error. Integration Service doesn't write row.</i>

DATE_PROMISED	RETURN VALUE
Dec 12 1997 8:07:33AM	Jun 12 1997 8:07:33AM
NULL	NULL

Las siguientes expresiones cambian el año a 2000 para las fechas del puerto DATE_PROMISED:

```
SET_DATE_PART( DATE_PROMISED, 'Y', 2000 )
SET_DATE_PART( DATE_PROMISED, 'YY', 2000 )
SET_DATE_PART( DATE_PROMISED, 'YYY', 2000 )
SET_DATE_PART( DATE_PROMISED, 'YYYY', 2000 )
```

DATE_PROMISED	RETURN VALUE
Jan 1 1997 12:15:56AM	Jan 1 2000 12:15:56AM
Feb 13 1997 2:30:01AM	Feb 13 2000 2:30:01AM
Mar 31 1997 5:10:15PM	Mar 31 2000 5:10:15PM
Dec 12 1997 8:07:33AM	Dec 12 2000 4:07:33PM
NULL	NULL

Consejo

Si desea cambiar varias partes de una fecha a la vez, puede anidar varias funciones SET_DATE_PART dentro del argumento *date*. Puede escribir, por ejemplo, la siguiente expresión para cambiar todas las fechas del puerto DATE_ENTERED a 1 de julio de 1998:

```
SET_DATE_PART( SET_DATE_PART( SET_DATE_PART( DATE_ENTERED, 'YYYY', 1998), 'MM', 7), 'DD', 1)
```

SIGN

Se devuelve si un valor numérico es positivo, negativo o 0.

Sintaxis

```
SIGN( numeric_value )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Valor numérico. Pasa los valores que se van a evaluar. Puede especificar cualquier expresión de transformación válida.

Valor devuelto

Se devuelve -1 para los valores negativos.

Se devuelve 0 para 0.

Se devuelve 1 para los valores positivos.

Se devuelve NULL si el valor es NULL.

Ejemplo

La siguiente expresión determina si el puerto SALES incluye valores negativos:

```
SIGN( SALES )
```

SALES	RETURN VALUE
100	1
-25.99	-1
0	0
NULL	NULL

SIN

Devuelve el seno de un valor numérico (expresado en radianes).

Sintaxis

```
SIN( numeric_value )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. Datos numéricos expresados en radianes (grados multiplicados por pi y divididos entre 180). Pasa los valores para los que se va a calcular el seno. Puede especificar cualquier expresión de transformación válida. Además, puede usar operadores para convertir un valor numérico en radianes o realizar una operación aritmética durante el cálculo de SIN.

Valor devuelto

Valor doble.

Se devuelve NULL si el valor pasado a la función es NULL.

Ejemplo

La siguiente expresión convierte los valores del puerto Degrees en radianes y, a continuación, calcula el seno para cada radián:

```
SIN( DEGREES * 3.14159265359 / 180 )
```

DEGREES	RETURN VALUE
0	0
90	1
70	0.939692620785936
30	0.500000000000003
5	0.0871557427476639
89	0.999847695156393
NULL	NULL

Puede realizar una operación aritmética con los valores pasados a SIN antes de que la función calcule el seno. Por ejemplo:

```
SIN( ARCS * 3.14159265359 / 180 )
```

SINH

Devuelve el seno hiperbólico del valor numérico.

Sintaxis

```
SINH( numeric_value )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. Datos numéricos expresados en radianes (grados multiplicados por pi y divididos entre 180). Pasa los valores para los que se va a calcular el seno hiperbólico. Puede especificar cualquier expresión de transformación válida.

Valor devuelto

Valor doble.

Se devuelve NULL si el valor pasado a la función es NULL.

Ejemplo

La siguiente expresión devuelve el seno hiperbólico para los valores del puerto Angles:

```
SINH( ANGLES )
```

ANGLES	RETURN VALUE
1.0	1.1752011936438
2.897	9.03225804884884
3.66	19.4178051793031
5.45	116.376934801486
0	0.0
0.345	0.35188478309993
NULL	NULL

Sugerencia

Puede realizar una operación aritmética con los valores pasados a SINH antes de que la función calcule el seno. Por ejemplo:

```
SINH( MEASURES.ARCS / 180 )
```

SIZE

Devuelve el tamaño de la matriz o la asignación especificadas.

Sintaxis

```
SIZE(value)
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio/Opcional	Descripción
value	Obligatorio	Tipo de datos de matriz o asignación. La matriz o la asignación para la que desea determinar el tamaño. Si especifica una matriz, la función devuelve el número de elementos de la matriz. Si especifica una asignación, la función devuelve el número de pares clave-valor de la asignación.

Valor de devolución

Int.

Devuelve -1 si la matriz o la asignación es NULL.

Ejemplos

La siguiente expresión devuelve el tamaño de la matriz ITEM_NAME.

```
SIZE(ITEM_NAME)
```

ITEM_NAME	RETURN VALUE
['apples', 'bananas', 'oranges']	3
['milk', 'coffee', 'tea', 'chai']	4
['cookie', 'cake']	2
['croissant', NULL]	2
NULL	-1

La siguiente expresión devuelve el tamaño de la asignación SHIPMENT_DETAILS.

```
SIZE(SHIPMENT_DETAILS)
```

SHIPMENT_DETAILS	RETURN VALUE
[CA -> CNTR345, TX -> T234]	2
[AZ -> CNTR123, AL -> CNTR730, CA -> CNTR345]	3
[FL -> CNTR208, NULL]	2
NULL	-1

SOUNDEX

Cifra el valor de una cadena en una cadena de cuatro caracteres.

SOUNDEX funciona con caracteres del alfabeto inglés (A-Z). Utiliza el primer carácter de la cadena de entrada como el primer carácter del valor de retorno y codifica las tres consonantes restantes únicas como números.

SOUNDEX codifica caracteres de acuerdo a la siguiente lista de reglas:

- Utiliza el primer carácter de *string* como el primer carácter del valor de retorno y lo codifica en caja mayúscula. Por ejemplo, tanto SOUNDEX('John') como SOUNDEX('john') devuelven 'J500'.
- Codifica las tres primeras consonantes únicas que siguen al primer carácter en *string* y omite el resto. Por ejemplo, tanto SOUNDEX('JohnRB') como SOUNDEX('JohnRBCD') devuelven 'J561'.
- Asigna un código individual a consonantes que suenan de forma parecida.

La siguiente tabla incluye las pautas de codificación de SOUNDEX para consonantes:

Tabla 2. Pautas de codificación SOUNDEX para consonantes

Código	Consonant
1	B, P, F, V
2	C, S, G, J, K, Q, X, Z
3	D, T
4	L
5	M, N
6	R

- Omite los caracteres A, E, I, O, U, H y W excepto en los casos en que uno de ellos sea el primero en *string*. Por ejemplo, SOUNDEX('A123') devuelve 'A000' y SOUNDEX('MAeiouhwC') devuelve 'M200'.
- Si *string* produce menos de cuatro caracteres, SOUNDEX rellena la cadena resultante con ceros. Por ejemplo, SOUNDEX('J') devuelve 'J000'.
- Si *string* contiene un conjunto de consonantes consecutivas que utilizan el mismo código que aparece en ["SOUNDEX" en la página 195](#), SOUNDEX codifica la primera ocurrencia y omite el resto de ocurrencias del conjunto. Por ejemplo, SOUNDEX('AbbpdMN') devuelve 'A135'.
- Omite los números en *string*. Por ejemplo, tanto SOUNDEX('Joh12n') como SOUNDEX('1John') devuelven 'J500'.
- Devuelve NULL si *string* es NULL o si todos los caracteres en *string* no son letras del alfabeto inglés.

Sintaxis

```
SOUNDEX( string )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>string</i>	Obligatorio	Cadena de caracteres. Pasa el valor de cadena que se desea codificar. Se puede especificar cualquier expresión de transformación válida.

Valor de devolución

Cadena.

NULL si una de las siguientes condiciones es verdadera:

- Si el valor pasado a la función es NULL.
- Ningún carácter en *string* es una letra del alfabeto inglés.
- *string* está vacío.

Ejemplo

La siguiente expresión codifica los valores en el puerto EMPLOYEE_NAME:

```
SOUNDEX ( EMPLOYEE_NAME )
```

EMPLOYEE_NAME	RETURN VALUE
John	J500
William	W450
jane	J500
joh12n	J500
1abc	A120
NULL	NULL

SQL_LIKE

Devuelve si un valor coincide con un patrón de expresión regular. Permite validar patrones de datos como números de identificación, números de teléfono, códigos postales y nombres de estado.

Sintaxis

```
SQL_LIKE(subject, pattern, escape character)
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio/ Opcional	Descripción
subject	Obligatorio	Tipo de datos de cadena. Pasa el valor que se desea hacer coincidir con la expresión regular. Encierre el valor entre comillas sencillas.
patrón	Obligatorio	Tipo de datos de cadena. Expresión regular con la que se desea coincidir. Encierre el patrón entre comillas sencillas.
carácter de escape	Opcional	Tipo de datos de cadena. La función SQL_LIKE admite como caracteres de escape el signo de porcentaje (%) y el signo de subrayado (_). Encierre el carácter de escape entre comillas sencillas.

Valor devuelto

TRUE si los datos coinciden con el patrón.

FALSE si los datos no coinciden con el patrón.

NULL si la entrada es un valor nulo o si el patrón es NULL.

Ejemplo

Puede utilizar SQL_LIKE en una expresión para buscar nombres que coincidan con un patrón. Por ejemplo, la siguiente expresión hace coincidir nombres que tengan el patrón "A_%" con el carácter de escape '#':

```
SQL_LIKE(ENAME, 'A_#%', '#')
```

ENAME	Valor
SMITH	FALSE
AX%	TRUE
MILLER	FALSE
A%	FALSE
JONES	FALSE
BLAKE	FALSE
A%l	FALSE

SQRT

Devuelve la raíz cuadrada de un valor numérico no negativo.

Sintaxis

```
SQRT( numeric_value )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Valor numérico positivo. Pasa los valores para los que se va a calcular la raíz cuadrada. Puede especificar cualquier expresión de transformación válida.

Valor devuelto

Valor doble.

Se devuelve NULL si el valor pasado a la función es NULL.

Ejemplo

La siguiente expresión devuelve la raíz cuadrada de los valores del puerto Numbers:

```
SQRT( NUMBERS )
```

NUMBERS	RETURN VALUE
100	10

NUMBERS	RETURN VALUE
-100	Error. Servicio de integración de datos does not write row.
NULL	NULL
60.54	7.78074546557076

El valor -100 genera un error porque la función SQRT solamente evalúa valores numéricos positivos. Si pasa un valor negativo o un valor de carácter, el Servicio de integración de datos muestra un error de evaluación de transformación y no escribe la fila.

Puede realizar una operación aritmética con los valores pasados a SQRT antes de que la función calcule la raíz cuadrada.

STDDEV

Devuelve la desviación estándar de los valores numéricos que se pasan a esta función. STDDEV se utiliza para analizar datos estadísticos. Dentro de STDDEV solamente se puede anidar una función agregada adicional y la función anidada debe devolver un tipo de datos Numérico.

Sintaxis

```
STDDEV( numeric_value [,filter_condition] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipos de datos numéricos. Esta función pasa los valores para los que se desea calcular una desviación estándar o los resultados de una función. Se puede especificar cualquier expresión de transformación válida. Se pueden usar operadores para obtener el promedio de valores en distintos puertos.
<i>filter_condition</i>	Opcional	Limita las filas en la búsqueda. La condición de filtro debe ser un valor numérico o dar un valor TRUE, FALSE o NULL. Se puede especificar cualquier expresión de transformación válida.

Valor de retorno

Valor numérico.

NULL si todos los valores pasados a la función son NULL, o si no se ha seleccionado ninguna fila (por ejemplo, la condición del filtro da un valor FALSE o NULL para todas las filas).

Nota: Si el valor devuelto es decimal con una precisión superior a 15, puede habilitar la alta precisión para garantizar una precisión decimal de hasta 38 dígitos.

Nulos

Si un valor es NULL, STDDEV lo omite. Sin embargo, si todos los valores son NULL, STDDEV devuelve NULL.

Agrupar por

STDDEV agrupa valores por grupos según los puertos definidos en la transformación, devolviendo un resultado para cada grupo.

Si no hay un grupo por puerto, STDDEV trata todas las filas como un solo grupo, devolviendo un valor.

Ejemplos

La siguiente expresión calcula la desviación estándar para todas las filas superiores a \$2000,00 en el puerto TOTAL_SALES:

```
STDDEV( SALES, SALES > 2000.00 )
```

SALES

2198.0

1010.90

2256.0

153.88

3001.0

NULL

8953.0

RETURN VALUE: 3254.60361129688

La función no incluye los valores 1010,90 y 153,88 en el cálculo porque *filter_condition* especifica ventas superiores a \$2.000.

La siguiente expresión calcula la desviación estándar para todas las filas en el puerto SALES:

```
STDDEV(SALES)
```

SALES

2198.0

2198.0

2198.0

2198.0

RETURN VALUE: 0

El valor de retorno es 0 porque cada fila contiene el mismo número (no hay ninguna desviación estándar). Si no hay ninguna desviación estándar, el valor de retorno es 0.

STRUCT

Genera una estructura con nombres de elementos y tipos de datos basados en los argumentos especificados.

Sintaxis

```
STRUCT(element_name1, value1 as any [, element_name2, value2 as any] ...)
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio/Opcional	Descripción
element_name1	Obligatorio	El nombre del elemento de estructura.
value1	Obligatorio	Cualquier tipo de datos. El valor del elemento de estructura.

Si utiliza la función STRUCT en una expresión de salida para un puerto de estructura, los argumentos de función deben coincidir con el tipo de datos de los elementos de la definición de tipo de datos complejos.

Valor de devolución

Estructura.

Ejemplos

La siguiente expresión genera una estructura.

```
STRUCT(city , 'New York', state, 'NY')
```

RETURN VALUE

```
{
  city:New York
  state:NY
}
```

La siguiente expresión genera una estructura para un puerto de salida de estructura con una definición de tipo de datos complejos **adrs_typedef**:

```
STRUCT(city, cust_city, state, cust_state)
```

La definición de tipo de datos complejos **adrs_typedef** se define como sigue en la biblioteca de definiciones de tipo:

```
adrs_typedef{
  city : string
```

```

    state : string
}

```

cust_city	cust_state	RETURN VALUE
NEWYORK	NY	<pre> { city:NEWYORK state:NY } </pre>
REDWOOD CITY	CA	<pre> { city:REDWOOD CITY state:CA } </pre>

STRUCT_AS

Genera una estructura con un esquema basado en la definición de tipo de datos complejos especificada y los valores que se pasan como argumento.

Sintaxis

```
STRUCT_AS (:Type.type_definition_library.type_definition, struct_value)
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio/Opcional	Descripción
:Type.type_definition_library.type_definition	Obligatorio	La definición de tipo de datos complejos que representa el esquema de los datos de estructura. Utilice el calificador de referencia :Type para hacer referencia a la biblioteca de definiciones de tipos que contiene la definición de tipo de datos complejos.
struct_value	Obligatorio	Valor para cada elemento de la definición de tipo de datos complejos separado por coma.

Valor de devolución

Estructura.

Ejemplos

La siguiente expresión genera una estructura basada en la definición de tipo de datos complejos especificada h1_address_def con los valores que se pasan como argumentos para los elementos de estructura.

```
STRUCT_AS (:Type.type_definition_library.h1_address_def, City, State, ZIP)
```

La definición de tipo de datos complejos **h1_address_def** se define como sigue en la biblioteca de definiciones de tipo:

```

h1_address_def{
  city : string
  state : string
}

```

```

    zip : int
}

```

city	state	zip	RETURN VALUE
NEWYORK	NY	12345	{ city:NEWYORK state:NY zip:12345 }
REDWOOD CITY	CA	23452	{ city:REDWOOD CITY state:CA zip:23452 }

SUBSTR

Devuelve una porción de una cadena. SUBSTR cuenta todos los caracteres, incluidos los espacios en blanco, comenzando por el inicio de la cadena.

Sintaxis

```
SUBSTR( string, start [,length] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>string</i>	Obligatorio	Debe ser una cadena de carácter. Pasa las cadenas que se desea buscar. Se puede especificar cualquier expresión de transformación válida. Si se pasa un valor numérico, la función la convierte en una cadena de caracteres.
<i>start</i>	Obligatorio	El valor debe ser un entero. La posición en la cadena donde se desea iniciar el recuento. Se puede especificar cualquier expresión de transformación válida. Si la posición inicial es un número positivo, SUBSTR localiza la posición inicial contando a partir del principio de la cadena. Si la posición inicial es un número negativo, SUBSTR localiza la posición inicial contando a partir del principio de la cadena. Si la posición inicial es 0, SUBSTR busca a partir del primer carácter de la cadena.
<i>length</i>	Opcional	Debe ser un entero mayor que 0. El número de caracteres que se desea que devuelva SUBSTR. Se puede especificar cualquier expresión de transformación válida. Si se omite el argumento de longitud, SUBSTR devuelve todos los caracteres a a partir de la posición inicial hasta el final de la cadena. Si se pasa un entero negativo o 0, la función devuelve una cadena vacía. Si se pasa un decimal, la función lo redondea al valor entero más próximo.

Valor de retorno

Cadena.

Cadena vacía si se pasa un valor negativo o de longitud 0.

NULL si el valor pasado a la función es NULL.

Ejemplos

Las siguientes expresiones devuelven el código de área para cada fila en el puerto Phone:

```
SUBSTR( PHONE, 0, 3 )
```

PHONE	RETURN VALUE
809-555-0269	809
357-687-6708	357
NULL	NULL

```
SUBSTR( PHONE, 1, 3 )
```

PHONE	RETURN VALUE
809-555-3915	809
357-687-6708	357
NULL	NULL

Las siguientes expresiones devuelven el número de teléfono sin el código de área para cada fila en el puerto Phone:

```
SUBSTR( PHONE, 5, 8 )
```

PHONE	RETURN VALUE
808-555-0269	555-0269
809-555-3915	555-3915
357-687-6708	687-6708
NULL	NULL

También se puede pasar un valor inicial negativo para devolver el número de teléfono para cada fila en el puerto Phone: La expresión todavía lee la cadena origen de izquierda a derecha cuando devuelve el resultado del argumento *length*:

```
SUBSTR( PHONE, -8, 3 )
```

PHONE	RETURN VALUE
808-555-0269	555
809-555-3915	555
357-687-6708	687
NULL	NULL

Se puede anidar INSTR en el argumento *start* o *length* para buscar una cadena específica y devolver su posición.

La siguiente expresión evalúa una cadena, comenzando por el final de la misma. La expresión encuentra el último espacio (situado en el extremo derecho) de la cadena y luego devuelve todos los caracteres que lo preceden:

```
SUBSTR( CUST_NAME,1,INSTR( CUST_NAME,' ' ,-1,1 ) - 1 )
```

CUST_NAME	RETURN VALUE
PATRICIA JONES	PATRICIA
MARY ELLEN SHAH	MARY ELLEN

La siguiente expresión elimina el carácter '#' de una cadena:

```
SUBSTR( CUST_ID, 1, INSTR(CUST_ID, '#')-1 ) || SUBSTR( CUST_ID, INSTR(CUST_ID, '#')+1 )
```

Cuando el argumento *length* es más largo que la cadena, SUBSTR devuelve todos los caracteres desde la posición inicial hasta el final de la cadena. Vea el siguiente ejemplo:

```
SUBSTR('abcd', 2, 8)
```

El valor de retorno es 'bcd'. Compare este resultado con el siguiente ejemplo:

```
SUBSTR('abcd', -2, 8)
```

El valor de retorno es 'cd'.

SUM

Devuelve la suma de todos los valores de un puerto seleccionado. Opcionalmente, se puede aplicar un filtro para limitar el número de filas que se leen para calcular el total. Dentro de SUM solamente se puede anidar una función agregada adicional y la función anidada debe devolver un tipo de datos Numérico.

Sintaxis

```
SUM( numeric_value [, filter_condition ] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento:	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. Pasa los valores que se desea sumar. Se puede especificar cualquier expresión de transformación válida. Se pueden usar operadores para sumar valores en distintos puertos.
<i>filter_condition</i>	Opcional	Limita las filas en la búsqueda. La condición de filtro debe ser un valor numérico o dar un valor TRUE, FALSE o NULL. Se puede especificar cualquier expresión de transformación válida.

Valor de retorno

Valor numérico.

NULL si todos los valores pasados a la función son NULL, o si no se ha seleccionado ninguna fila (por ejemplo, la condición del filtro da un valor FALSE o NULL para todas las filas).

Nota: Si el valor devuelto es decimal con una precisión superior a 15, puede habilitar la alta precisión para garantizar una precisión decimal de hasta 38 dígitos.

Nulos

Si un valor es NULL, SUM lo omite. Sin embargo, si todos los valores pasados desde el puerto son NULL, SUM devuelve NULL.

Agrupar por

SUM agrupa valores por grupos según los puertos definidos en la transformación, devolviendo un resultado para cada grupo.

Si no hay un grupo por puerto, SUM trata todas las filas como un solo grupo, devolviendo un valor.

Ejemplo

La siguiente expresión devuelve la suma de todos los valores superiores a 2000 en el puerto Sales:

```
SUM( SALES, SALES > 2000 )
```

SALES

2500.0

1900.0

1200.0

NULL

3458.0

4519.0

RETURN VALUE: 10477.0

Consejo

Se pueden realizar operaciones aritméticas con los valores pasados a SUM antes de que la función calcule el total. Por ejemplo:

```
SUM( QTY * PRICE - DISCOUNT )
```

SYSTIMESTAMP

Devuelve la fecha y la hora actual del nodo que aloja el Servicio de integración de datos con precisión de nanosegundos. La precisión con que mostrará la fecha y hora depende de la plataforma.

El valor de retorno de la función varía en función de cómo configure el argumento:

- Al configurar el argumento de SYSTIMESTAMP como una variable, el Servicio de integración de datos evalúa la función para cada fila de la transformación.
- Al configurar el argumento de SYSTIMESTAMP como una constante, el Servicio de integración de datos evalúa la función una vez y guarda el valor para cada fila de la transformación.

Sintaxis

```
SYSTIMESTAMP ( [format] )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>format</i>	Opcional	Precisión con la que desea recuperar la marca de hora. Puede especificar una precisión de hasta segundos (SS), milisegundos (MS), microsegundos (US), o nanosegundos (NS). Escriba la cadena de formato entre comillas simples. La cadena de formato no distingue entre mayúsculas y minúsculas. Por ejemplo, para mostrar la fecha y la hora con precisión de milisegundos, utilice la siguiente sintaxis: SYSTIMESTAMP('MS'). La precisión predeterminada es microsegundos (US).

Valor de retorno

Marca de hora. Devuelve la fecha y la hora con la precisión especificada.

Ejemplos

Su organización cuenta con un servicio de pedidos en línea y procesa datos en tiempo real. Puede utilizar la función SYSTIMESTAMP para generar una clave principal para cada transacción en la base de datos de destino.

Genere una transformación de expresión con los siguientes puertos y valores:

Port Name	Port Type	Expression
Customer_Name	Input	n/a
Order_Qty	Input	n/a
Time_Counter	Variable	'US'
Transaction_Id	Output	SYSTIMESTAMP (Time_Counter)

En tiempo de ejecución, el Servicio de integración de datos genera la hora del sistema con precisión de microsegundos para cada fila:

Customer_Name	Order_Qty	Transaction_Id
Vani Deed	14	07/06/2007 18:00:30.701015000
Kalia Crop	3	07/06/2007 18:00:30.701029000
Vani Deed	6	07/06/2007 18:00:30.701039000
Harry Spoon	32	07/06/2007 18:00:30.701048000

TAN

Devuelve la tangente de un valor numérico (expresado en radianes).

Sintaxis

```
TAN( numeric_value )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio/ Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. Datos numéricos expresados en radianes (grados multiplicados por pi y divididos entre 180). Pasa los valores numéricos para los que se va a calcular la tangente. Puede especificar cualquier expresión de transformación válida.

Valor devuelto

Valor doble.

Se devuelve NULL si el valor pasado a la función es NULL.

Ejemplo

La siguiente expresión devuelve la tangente para todos los valores del puerto Degrees:

```
TAN( DEGREES * 3.14159 / 180 )
```

DEGREES	RETURN VALUE
70	2.74747741945531
50	1.19175359259435
30	0.577350269189672
5	0.0874886635259298
18	0.324919696232929
89	57.2899616310952
NULL	NULL

TANH

Devuelve la tangente hiperbólica del valor numérico pasado a esta función.

Sintaxis

```
TANH( numeric_value )
```


En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. Datos numéricos expresados en radianes (grados multiplicados por pi y divididos entre 180). Pasa los valores numéricos para los que se va a calcular la tangente hiperbólica. Puede especificar cualquier expresión de transformación válida.

Valor devuelto

Valor doble.

Se devuelve NULL si el valor pasado a la función es NULL.

Ejemplo

La siguiente expresión devuelve la tangente hiperbólica para los valores del puerto Angles:

```
TANH ( ANGLES )
```

ANGLES	RETURN VALUE
1.0	0.761594155955765
2.897	0.993926947790665
3.66	0.998676551914886
5.45	0.999963084213409
0	0.0
0.345	0.331933853503641
NULL	NULL

Sugerencia

Puede realizar una operación aritmética con los valores pasados a TANH antes de que la función calcule la tangente hiperbólica. Por ejemplo:

```
TANH ( ARCS / 360 )
```

TIME_RANGE

Determina el intervalo de tiempo para los eventos de transmisión a los que se va a unir.

La función `TIME_RANGE` se aplica únicamente para una transformación de combinación en una asignación de transmisión.

Sintaxis

```
TIME_RANGE (EventTime1,EventTime2,Format,Interval)
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio/ Opcional	Descripción
EventTime1	Obligatorio	Tipo de datos Date. La hora a la que se genera un evento de transmisión en el puerto principal de una transformación de combinación.
EventTime2	Obligatorio	Tipo de datos Date. La hora a la que se genera un evento de transmisión en el puerto de detalles de una transformación de combinación.
Formatear	Obligatorio	Una cadena de formato que especifica la parte del valor de hora del evento que desea cambiar. Escriba la cadena de formato entre comillas simples. Por ejemplo, "Segundos". La cadena de formato no distingue mayúsculas de minúsculas. El argumento de formato acepta los siguientes valores: <ul style="list-style-type: none">- Años- Meses- Semanas- Días- Horas- Minutos- Segundos- Milisegundos- Microsegundos
Intervalo	Obligatorio	Un valor entero para el valor de hora del evento que quiere cambiar según el formato.

Valor de retorno

NULL si transfiere un valor nulo a la función.

Ejemplo

El siguiente ejemplo devuelve la expresión de intervalo de tiempo para la transformación de combinación:

```
TIME_RANGE(EventTime1,EventTime2,'Second',4)
```

VALOR DE DEVOLUCIÓN:

```
(EventTime1.<=(EventTime2).&&(EventTime2.<=(EventTime1.+(expr("INTERVAL 4 SECONDS")))))
```

TO_BIGINT

Convierte una cadena o valor numérico a un valor bigint. La sintaxis de TO_BIGINT contiene un argumento opcional que puede elegir para redondear el número al entero más cercano o truncar la parte decimal.

TO_BIGINT ignora los espacios a la izquierda.

Sintaxis

```
TO_BIGINT( value [, flag] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>value</i>	Obligatorio	Tipo de dato de cadena o numérico. Pasa el valor que desea convertir en un valor bigint. Puede introducir cualquier expresión de transformación válida.
<i>flag</i>	Opcional	<p>Marca que especifica si trunca o redondear la parte decimal. La marca debe ser un literal entero o las constantes TRUE o FALSE.</p> <p>TO_BIGINT trunca la parte decimal cuando la marca es TRUE o un número distinto de 0.</p> <p>TO_BIGINT redondea el valor al entero más cercano si la marca es FALSE o 0, o si se omite este argumento.</p> <p>La marca no está definida de forma predeterminada.</p>

Valor de retorno

Bigint.

NULL si el valor pasado a la función es NULL.

Si el valor que se pasa a la función contiene datos que no son válidos para un valor bigint, el servicio de integración de datos marca la fila como fila de error o no puede realizar la asignación.

Ejemplos

Las expresiones siguientes utilizan valores del puerto IN_TAX:

```
TO_BIGINT( IN_TAX, TRUE )
```

IN_TAX	RETURN VALUE
'7245176201123435.6789'	7245176201123435
'7245176201123435.2'	7245176201123435
'7245176201123435.2.48'	7245176201123435
NULL	NULL
'A12.3Grove'	<i>Error. Integration Service skips this row.</i>
' 176201123435.87'	176201123435
'-7245176201123435.2'	-7245176201123435
'-7245176201123435.23'	-7245176201123435

IN_TAX	RETURN VALUE
-9223372036854775806.9	-9223372036854775806
9223372036854775806.9	9223372036854775806
TO_BIGINT(IN_TAX)	
IN_TAX	RETURN VALUE
'7245176201123435.6789'	7245176201123436
'7245176201123435.2'	7245176201123435
'7245176201123435.348'	7245176201123435
NULL	NULL
'A12.3Grove'	<i>Error. Integration Service skips this row.</i>
' 176201123435.87'	176201123436
'-7245176201123435.6789'	-7245176201123436
'-7245176201123435.23'	-7245176201123435
-9223372036854775806.9	-9223372036854775807
9223372036854775806.9	9223372036854775807

TO_CHAR (Fechas)

Convierte fechas en cadenas de caracteres. TO_CHAR convierte también valores numéricos en cadenas. Puede convertir la fecha en cualquier formato mediante las cadenas de formato TO_CHAR.

TO_CHAR (fecha [,formato]) convierte un tipo de datos o un valor interno con un tipo de datos de fecha, marca de tiempo, marca de tiempo con zona horaria o marca de tiempo con zona horaria local en un valor de un tipo de datos de cadena especificado por la cadena de formato.

Sintaxis

```
TO_CHAR( date [,format] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>date</i>	Obligatorio	Tipo de dato de fecha y hora. Pasa los valores de fecha que desea convertir en cadenas de caracteres. Se puede especificar cualquier expresión de transformación válida.
<i>format</i>	Opcional	Especifique una cadena de formato TO_CHAR válida. La cadena de formato define el formato del valor devuelto, no el formato de los valores del argumento date. Si omite la cadena de formato, la función devuelve una cadena que se basa en el formato de fecha especificado en la configuración de la asignación.

Valor devuelto

Cadena.

NULL si el valor pasado a la función es NULL.

Ejemplos

La siguiente expresión convierte las fechas del puerto DATE_PROMISED en texto en el formato MON DD YYYY:

```
TO_CHAR( DATE_PROMISED, 'MON DD YYYY' )
```

DATE_PROMISED	RETURN VALUE
Apr 1 1998 12:00:10AM	'Apr 01 1998'
Feb 22 1998 01:31:10PM	'Feb 22 1998'
Oct 24 1998 02:12:30PM	'Oct 24 1998'
NULL	NULL

Si omite el argumento *format*, TO_CHAR devuelve una cadena en el formato de fecha especificado en la configuración de asignación, que es, como valor predeterminado, MM/DD/YYYY HH24:MI:SS.US:

```
TO_CHAR( DATE_PROMISED )
```

DATE_PROMISED	RETURN VALUE
Apr 1 1998 12:00:10AM	'04/01/1998 00:00:10.000000'
Feb 22 1998 01:31:10PM	'02/22/1998 13:31:10.000000'
Oct 24 1998 02:12:30PM	'10/24/1998 14:12:30.000000'
NULL	NULL

Las siguientes expresiones devuelven el día de la semana para cada fecha de un puerto:

```
TO_CHAR( DATE_PROMISED, 'D' )
```

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10AM	'3'
02-22-1997 01:31:10PM	'7'
10-24-1997 02:12:30PM	'6'
NULL	NULL

```
TO_CHAR( DATE_PROMISED, 'DAY' )
```

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10AM	'Tuesday'
02-22-1997 01:31:10PM	'Saturday'
10-24-1997 02:12:30PM	'Friday'
NULL	NULL

La siguiente expresión devuelve el día del mes para cada fecha de un puerto:

```
TO_CHAR( DATE_PROMISED, 'DD' )
```

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10AM	'01'
02-22-1997 01:31:10PM	'22'
10-24-1997 02:12:30PM	'24'
NULL	NULL

La siguiente expresión devuelve el día del año para cada fecha de un puerto:

```
TO_CHAR( DATE_PROMISED, 'DDD' )
```

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10AM	'091'
02-22-1997 01:31:10PM	'053'
10-24-1997 02:12:30PM	'297'
NULL	NULL

Las siguientes expresiones devuelven la hora del día para cada fecha de un puerto:

```
TO_CHAR( DATE_PROMISED, 'HH' )
TO_CHAR( DATE_PROMISED, 'HH12' )
```

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10AM	'12'
02-22-1997 01:31:10PM	'01'
10-24-1997 02:12:30PM	'02'
NULL	NULL

```
TO_CHAR( DATE_PROMISED, 'HH24' )
```

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10AM	'00'
02-22-1997 01:31:10PM	'13'
10-24-1997 11:12:30PM	'23'
NULL	NULL

La siguiente expresión convierte los valores de fecha en valores MJD expresados como cadenas:

```
TO_CHAR( SHIP_DATE, 'J' )
```

SHIP_DATE	RETURN VALUE
Dec 31 1999 03:59:59PM	2451544
Jan 1 1900 01:02:03AM	2415021

La siguiente expresión convierte las fechas en cadenas en el formato MM/DD/YY;

```
TO_CHAR( SHIP_DATE, 'MM/DD/RR' )
```

SHIP_DATE	RETURN VALUE
12/31/1999 01:02:03AM	12/31/99
09/15/1996 03:59:59PM	09/15/96
05/17/2003 12:13:14AM	05/17/03

También puede utilizar la cadena de formato SSSSS en una expresión TO_CHAR. Por ejemplo, la siguiente expresión convierte las fechas del puerto SHIP_DATE en cadenas que representan los segundos totales hasta medianoche:

```
TO_CHAR( SHIP_DATE, 'SSSSS')
```

SHIP_DATE	RETURN_VALUE
12/31/1999 01:02:03AM	3783
09/15/1996 03:59:59PM	86399

En las expresiones TO_CHAR, la cadena de formato YY produce los mismos resultados que la cadena de formato RR.

La siguiente expresión convierte las fechas en cadenas en el formato MM/DD/YY;

```
TO_CHAR( SHIP_DATE, 'MM/DD/YY')
```

SHIP_DATE	RETURN_VALUE
12/31/1999 01:02:03AM	12/31/99
09/15/1996 03:59:59PM	09/15/96
05/17/2003 12:13:14AM	05/17/03

La siguiente expresión devuelve la semana del mes para cada fecha de un puerto:

```
TO_CHAR( DATE_PROMISED, 'W' )
```

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10AM	'01'
02-22-1997 01:31:10AM	'04'
10-24-1997 02:12:30PM	'04'
NULL	NULL

La siguiente expresión devuelve la semana del año para cada fecha de un puerto:

```
TO_CHAR( DATE_PROMISED, 'WW' )
```

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10PM	'18'
02-22-1997 01:31:10AM	'08'
10-24-1997 02:12:30AM	'43'
NULL	NULL

Consejo

Puede combinar TO_CHAR y TO_DATE para convertir un valor numérico de un mes en el valor de texto de un mes con una función como:

```
TO_CHAR( TO_DATE( numeric_month, 'MM' ), 'MONTH' )
```

TO_CHAR (Números)

Convierte valores numéricos a cadenas de texto. TO_CHAR también convierte fechas a cadenas.

TO_CHAR convierte valores dobles en cadenas de texto de la siguiente forma:

- Convierte los valores dobles de hasta 16 dígitos en cadenas y ofrece una precisión de hasta 15 dígitos. Si se pasa un número con más de 15 dígitos, TO_CHAR redondea el número en función del decimosexto dígito y devuelve la representación de cadena del número en notación científica. Por ejemplo, el valor doble 1234567890123456 se convierte en el valor de cadena "1.23456789012346e+015".
- Devuelve la notación decimal para los números en los intervalos (-1e16, -1e-16] y [1e-16, 1e16). TO_CHAR devuelve la notación científica para los números fuera de estos intervalos. Por ejemplo, el valor doble 10842764968208837340 se convierte en el valor de cadena "1.08427649682088e+019".

TO_CHAR convierte valores decimales en cadenas de texto de la siguiente forma:

- En el modo de alta precisión, TO_CHAR convierte valores decimales de hasta 38 dígitos en cadenas. Si se pasa un valor decimal con más de 38 dígitos, TO_CHAR devuelve la notación científica para los números con más de 38 dígitos.
- En el modo de baja precisión, TO_CHAR trata los valores decimales como valores dobles.

Sintaxis

```
TO_CHAR( numeric_value )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. El valor numérico que desea convertir en una cadena. Se puede especificar cualquier expresión de transformación válida.

Valor de devolución

Cadena.

NULL si el valor pasado a la función es NULL.

Ejemplo de conversión de doble

La siguiente expresión convierte los valores dobles en el puerto SALES en cadenas:

```
TO_CHAR( SALES )
```

SALES	RETURN VALUE
1010.99	'1010.99'
-15.62567	'-15.62567'
10842764968208837340	'1.08427649682088e+019' (rounded based on the 16th digit and returns the value in scientific notation)
236789034569723	'236789034569723'
0	'0'
33.15	'33.15'
NULL	NULL

Ejemplo de conversión de decimal

La siguiente expresión convierte los valores decimales en el puerto SALES en cadenas en modo de alta precisión:

```
TO_CHAR( SALES )
```

SALES	RETURN VALUE
2378964536789761	'2378964536789761'
1234567890123456789012345679	'1234567890123456789012345679'
1.234578945469649345876123456	'1.234578945469649345876123456'
0.999999999999999999999999999999	'0.999999999999999999999999999999'
12345678901234567890123456799	'12345678901234567890123456799'
23456788992233456678458934567123465239	'23456788992233456678458934567123465239'
423456789012345678901234567991234567899 (mayor que 38)	'4.23456789012346e+038'

TO_DATE

Convierte una cadena de caracteres en un tipo de datos Date/Time. Utilice las cadenas de formato TO_DATE para especificar el formato de las cadenas origen.

El puerto de salida debe ser Date/Time para expresiones TO_DATE.

Si desea convertir años de dos dígitos con TO_DATE, utilice la cadena de formato RR o YY. No utilice la cadena de formato YYYY.

Sintaxis

```
TO_DATE( string [, format] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>string</i>	Obligatorio	Debe ser un tipo de datos cadena. Pasa los valores que se desea convertir en fechas. Se puede especificar cualquier expresión de transformación válida.
<i>format</i>	Opcional	Introduzca una cadena de formato TO_DATE válida. La cadena de formato debe coincidir con las partes del argumento <i>string</i> . Por ejemplo, si se pasa la cadena 'Mar 15 1998 12:43:10AM', debe utilizar la cadena de formato 'MON DD YYYY HH12:MI:SSAM'. Si se omite la cadena de formato, el valor de la cadena debe estar en el formato de fecha especificado en la sesión.

Valor de retorno

Fecha.

TO_DATE siempre devuelve una fecha y una hora. Si se pasa una cadena que no tiene un valor de hora, la fecha devuelta siempre incluye la hora 00:00:00.000000000. Se pueden asignar los resultados de esta función a cualquier columna destino con un tipo de datos datetime. Si la precisión de la columna destino es menor de nanosegundos, el Servicio de integración de datos trunca el valor datetime para que coincida con la precisión de la columna destino cuando escribe valores datetime en el destino.

NULL si se pasa un valor nulo a esta función.

Advertencia: El formato de la cadena TO_DATE debe coincidir con la cadena de formato, incluido cualquier separador de fecha. De lo contrario, el Servicio de integración de datos puede devolver valores inexactos u omitir el registro.

Ejemplos

La siguiente expresión devuelve valores de fecha para las cadenas en el puerto DATE_PROMISED. TO_DATE siempre devuelve una fecha y una hora. Si se pasa una cadena que no tiene un valor de hora, la fecha devuelta siempre incluye la hora 00:00:00.000000000. Si ejecuta una asignación en el siglo XX, el siglo será el 19. En este ejemplo, el año actual en el nodo que ejecuta el Servicio de integración de datos es 1998. El formato de datetime para la columna destino es MON DD YY HH24:MI SS, por lo que el Servicio de integración de datos trunca el valor datetime a segundos cuando lo escribe en el destino:

```
TO_DATE( DATE_PROMISED, 'MM/DD/YY' )
```

DATE_PROMISED	RETURN VALUE
'01/22/98'	Jan 22 1998 00:00:00
'05/03/98'	May 3 1998 00:00:00
'11/10/98'	Nov 10 1998 00:00:00
'10/19/98'	Oct 19 1998 00:00:00
NULL	NULL

La siguiente expresión devuelve valores de fecha y hora para las cadenas en el puerto DATE_PROMISED. Si se pasa una cadena que no tiene un valor de hora, el Servicio de integración de datos devuelve un error. Si ejecuta una asignación en el siglo XX, el siglo será el 19. El año actual en el nodo que ejecuta el Servicio de integración de datos es 1998.

```
TO_DATE( DATE_PROMISED, 'MON DD YYYY HH12:MI:SSAM' )
```

DATE_PROMISED	RETURN VALUE
'Jan 22 1998 02:14:56PM'	Jan 22 1998 02:14:56PM
'Mar 15 1998 11:11:11AM'	Mar 15 1998 11:11:11AM
'Jun 18 1998 10:10:10PM'	Jun 18 1998 10:10:10PM
'October 19 1998'	<i>Error. Integration Service skips this row.</i>
NULL	NULL

La siguiente expresión convierte cadenas en el puerto SHIP_DATE_MJD_STRING en valores de fecha:

```
TO_DATE (SHIP_DATE_MJD_STR, 'J')
```

SHIP_DATE_MJD_STR	RETURN VALUE
'2451544'	Dec 31 1999 00:00:00.000000000
'2415021'	Jan 1 1900 00:00:00.000000000

Debido a que la cadena de formato J no incluye la porción de hora de una fecha, los valores de retorno tienen la hora establecida en 00:00:00.000000000.

La siguiente expresión convierte una cadena en un formato de año de cuatro dígitos. El año actual es 1998:

```
TO_DATE( DATE_STR, 'MM/DD/RR')
```

DATE_STR	RETURN VALUE
'04/01/98'	04/01/1998 00:00:00.000000000
'08/17/05'	08/17/2005 00:00:00.000000000

La siguiente expresión convierte una cadena en un formato de año de cuatro dígitos. El año actual es 1998:

```
TO_DATE( DATE_STR, 'MM/DD/YY')
```

DATE_STR	RETURN VALUE
'04/01/98'	04/01/1998 00:00:00.000000000
'08/17/05'	08/17/1905 00:00:00.000000000

Nota: Para la segunda fila, RR devuelve el año 2005 e YY devuelve el año 1905.

La siguiente expresión convierte una cadena en un formato de año de cuatro dígitos. El año actual es 1998:

```
TO_DATE( DATE_STR, 'MM/DD/Y')
```

DATE_STR	RETURN VALUE
'04/01/8'	04/01/1998 00:00:00.000000000
'08/17/5'	08/17/1995 00:00:00.000000000

La siguiente expresión convierte una cadena en un formato de año de cuatro dígitos. El año actual es 1998:

```
TO_DATE( DATE_STR, 'MM/DD/YYYY')
```

DATE_STR	RETURN VALUE
'04/01/998'	04/01/1998 00:00:00.000000000
'08/17/995'	08/17/1995 00:00:00.000000000

La siguiente expresión convierte cadenas que incluyen los segundos desde medianoche en valores de fecha:

```
TO_DATE( DATE_STR, 'MM/DD/YYYY SSSSS')
```

DATE_STR	RETURN_VALUE
'12/31/1999 3783'	12/31/1999 01:02:03
'09/15/1996 86399'	09/15/1996 23:59:59

Si el destino acepta formatos de fecha diferentes, use TO_DATE y IS_DATE con la función DECODE para probar formatos aceptables. Por ejemplo:

```
DECODE( TRUE,
  --test first format
  IS_DATE( CLOSE_DATE, 'MM/DD/YYYY HH24:MI:SS' ),
  --if true, convert to date
  TO_DATE( CLOSE_DATE, 'MM/DD/YYYY HH24:MI:SS' ),
  --test second format; if true, convert to date
  IS_DATE( CLOSE_DATE, 'MM/DD/YYYY' ), TO_DATE( CLOSE_DATE, 'MM/DD/YYYY' ),
  --test third format; if true, convert to date
  IS_DATE( CLOSE_DATE, 'MON DD YYYY' ), TO_DATE( CLOSE_DATE, 'MON DD YYYY' ),
  --if none of the above
  ERROR( 'NOT A VALID DATE' ) )
```

Puede combinar TO_CHAR y TO_DATE para convertir un valor numérico para un mes en el valor de texto de un mes utilizando una función como:

```
TO_CHAR( TO_DATE( numeric_month, 'MM' ), 'MONTH' )
```

TEMAS RELACIONADOS

- [“Normas y directrices para cadenas con formato de fecha” en la página 47](#)

TO_DECIMAL

Convierte una cadena o valor numérico en un valor decimal. TO_DECIMAL ignora los espacios a la izquierda.

Sintaxis

```
TO_DECIMAL( value [, scale] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>valor</i>	Obligatorio	Debe ser un tipo de datos de cadena o numérico. Pasa los valores que se desean convertir en valores decimales. Se puede especificar cualquier expresión de transformación válida.
<i>scale</i>	Opcional	Tiene que ser un literal entero entre 0 y 28, ambos incluidos. Especifica el número de dígitos permitidos después del punto decimal. Si se omite este argumento, la función devuelve un valor con la misma escala que el valor de entrada.

Valor devuelto

Decimal de precisión y escala entre 0 y 28, ambos incluidos.

NULL si el valor pasado a la función es NULL.

Si el valor que se pasa a la función contiene datos que no son válidos para un valor decimal, el servicio de integración de datos marca la fila como fila de error.

Nota: Si el valor devuelto es decimal con una precisión superior a 15, puede habilitar alta precisión para garantizar una precisión decimal de hasta 28 dígitos.

Ejemplo

Esta expresión utiliza valores del puerto IN_TAX. IN_TAX es un tipo de datos de cadena con precisión de 44 dígitos. RETURN VALUE es un tipo de datos de decimal con una precisión de 28 y una escala de 3:

```
TO_DECIMAL( IN_TAX, 3 )
```

IN_TAX	RETURN VALUE
'15.6789'	15.679
'60.2'	60.200
'118.348'	118.348
NULL	NULL
'A12.3Grove'	<i>Error. Integration Service skips this row.</i>

IN_TAX	RETURN VALUE
'711A1'	Error. Integration Service skips this row.
'1234567890.123'	1234567890.123
'123456789012345678901234567890.123'	Error. Integration Service skips this row.
'1234567890123456789012345678901234567890.123'	Error. Integration Service skips this row.

TO_DECIMAL38

Convierte una cadena o valor numérico en un valor decimal. TO_DECIMAL38 ignora los espacios en blanco a la izquierda.

Sintaxis

```
TO_DECIMAL38( value [, scale] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>valor</i>	Obligatorio	Debe ser un tipo de datos de cadena o numérico. Pasa los valores que desea convertir en valores decimales. Se puede especificar cualquier expresión de transformación válida.
<i>scale</i>	Opcional	Tiene que ser un literal entero entre 0 y 38, ambos incluidos. Especifica el número de dígitos permitidos después del punto decimal. Si se omite este argumento, la función devuelve un valor con la misma escala que el valor de entrada.

Valor devuelto

Decimal de precisión y escala entre 0 y 38, ambos incluidos.

NULL si el valor pasado a la función es NULL.

Si el valor que se pasa a la función contiene datos que no son válidos para un valor decimal, el Servicio de integración de datos marca la fila como fila de error. Por ejemplo, si pasa

`TO_DECIMAL38 ("1234567890123456789012345678901234567890.12")`, el Servicio de integración de datos rechaza la fila.

Nota: Si el valor devuelto es decimal con una precisión superior a 15, puede habilitar la alta precisión para garantizar una precisión decimal de hasta 38 dígitos.

Ejemplo

Esta expresión utiliza valores del puerto IN_TAX. IN_TAX es un tipo de datos de cadena con precisión de 44 dígitos. RETURN VALUE es un tipo de datos de decimal con una precisión de 38 y una escala de 3:

```
TO_DECIMAL38( IN_TAX, 3 )
```

IN_TAX	RETURN VALUE
'15.6789'	15.679
'60.2'	60.200
'118.348'	118.348
NULL	NULL
'A12.3Grove'	<i>Error. Integration Service skips this row.</i>
'1234567890.123'	1234567890.123
'123456789012345678901234567890.123'	123456789012345678901234567890.123
'1234567890123456789012345678901234567890.123'	<i>Error. Integration Service skips this row.</i>
'711A1'	<i>Error. Integration Service skips this row.</i>

TO_FLOAT

Convierte una cadena o un valor numérico en un número de coma flotante de doble precisión (tipo de datos Double). TO_FLOAT omite los espacios en blanco.

Sintaxis

```
TO_FLOAT( value )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
value	Obligatorio	Debe ser un tipo de dato cadena o numérico. Pasa los valores que se van a convertir en valores dobles. Puede introducir cualquier expresión de transformación válida.

Valor de retorno

Valor doble.

Se devuelve NULL si el valor pasado a esta función es NULL.

Si el valor que se pasa a la función contiene datos que no son válidos para un valor float, el servicio de integración de datos marca la fila como fila de error o no puede realizar la asignación.

Ejemplo

Esta expresión usa valores del puerto IN_TAX:

```
TO_FLOAT( IN_TAX )
```

IN_TAX	RETURN VALUE
'15.6789'	15.6789
'60.2'	60.2
'118.348'	118.348
NULL	NULL
'A12.3Grove'	<i>Error. Integration Service skips this row.</i>

TO_INTEGER

Convierte una cadena o valor numérico a un número entero. La sintaxis de TO_INTEGER contiene un argumento opcional que puede elegir para redondear el número al entero más cercano o truncar la parte decimal. TO_INTEGER ignora los espacios a la izquierda.

Sintaxis

```
TO_INTEGER( value [, flag] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>value</i>	Obligatorio	Tipo de dato de cadena o numérico. Pasa el valor que desea convertir a un número entero. Puede introducir cualquier expresión de transformación válida.
<i>flag</i>	Opcional	Marca que especifica si truncar o redondear la parte decimal. La marca debe ser un literal entero o las constantes TRUE o FALSE. TO_INTEGER trunca la parte decimal cuando la marca es TRUE o un número distinto de 0. TO_INTEGER redondea el valor al entero más cercano si la marca es FALSE o 0, o si se omite este argumento.

Valor de retorno

Entero.

NULL si el valor pasado a la función es NULL.

Si el valor que se pasa a la función contiene datos que no son válidos para un valor integer, el servicio de integración de datos marca la fila como fila de error o no puede realizar la asignación.

Ejemplos

Las expresiones siguientes utilizan valores del puerto IN_TAX. El Servicio de integración de datos muestra un error cuando la conversión provoca un desbordamiento numérico:

```
TO_INTEGER( IN_TAX, TRUE )
```

IN_TAX	RETURN VALUE
'15.6789'	15
'60.2'	60
'118.348'	118
'5,000,000,000'	Error. Integration Service skips this row.
NULL	NULL
'A12.3Grove'	Error. Integration Service skips this row.
' 123.87'	123
'-15.6789'	-15
'-15.23'	-15

```
TO_INTEGER( IN_TAX, FALSE)
```

IN_TAX	RETURN VALUE
'15.6789'	16
'60.2'	60
'118.348'	118
'5,000,000,000'	Error. Integration Service skips this row.
NULL	NULL
'A12.3Grove'	Error. Integration Service skips this row.
' 123.87'	124
'-15.6789'	-16
'-15.23'	-15

TO_TIMESTAMP_TZ

Convierte una cadena en marca de tiempo con valor de zona horaria. La función devuelve el tipo de datos de marca de tiempo con zona horaria. Utilice las cadenas de formato TO_TIMESTAMP_TZ para especificar el formato de las cadenas origen.

Sintaxis

```
TO_TIMESTAMP_TZ ( String , [format] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>String</i>	Obligatorio	Debe ser un tipo de datos de cadena. Pasa los valores que desea convertir en marca de tiempo con zona horaria. Se puede especificar cualquier expresión de transformación válida. La cadena debe ser una cadena de caracteres.
<i>format</i>	Opcional	Especifique una cadena de formato TO_TIMESTAMP_TZ válida. La cadena de formato debe coincidir con las partes del argumento string. Por ejemplo, si se pasa la cadena "Mar 15 1997 12:43:10AM", debe utilizar la cadena de formato "MON DD YYYY HH12:MI:SSAM TZR". Si no especifica la cadena de formato, la función utiliza el formato de fecha y hora predeterminado en el cuadro de diálogo Ejecutar configuraciones.

Valor devuelto

Devuelve una marca de tiempo con el tipo de datos de zona horaria.

NULL si la entrada es un valor nulo.

Si el valor que se pasa a la función contiene datos que no son válidos para un valor de marca de tiempo con zona horaria, el Servicio de integración de datos marca la fila como fila de error o no realiza la asignación.

Ejemplo

INPUT VALUE	RETURN VALUE
'1947-08-05 10:45:00.221111000 AM America/Los_Angeles', 'YYYY-MM-DD HH:MI:SS.NS AM TZR'	Devuelve un tipo de datos de marca de tiempo con zona horaria con los siguientes datos: '1947-08-05 10:45:00.221111000 AM AMERICA/LOS_ANGELES'
'1947-08-05 10:45:00.221111000 AM America/Los_Angeles', 'YYYY-MM-DD HH:MI:SS.NS AM'	Devuelve un tipo de datos de marca de tiempo con zona horaria sin especificar la región de zona horaria en el formato de región de zona horaria: '1947-08-05 10:45:00.221111000 AM AMERICA/LOS_ANGELES'

INPUT VALUE	RETURN VALUE
'1947-08-05 10:45:00.221111000 AM America/Los_Angeles'	Devuelve un tipo de datos de marca de tiempo con zona horaria sin especificar el formato de marca de tiempo con zona horaria. '1947-08-05 10:45:00.221111000 AM AMERICA/LOS_ANGELES' El formato de fecha y hora predeterminado del cuadro de diálogo Ejecutar configuraciones se utiliza cuando el formato no se especifica en el nivel de función. Formato de fecha y hora predeterminado: "YYYY-MM-DD HH:MI:SS.NS AM TZR" Si los datos de una marca de tiempo con zona horaria no coinciden con el formato dado, aparece el siguiente error:
'1947-08-05 10:45:00.221111000 AM America/Los_Angeles', 'MM-DD-YYYY HH:MI:SS.NS AM'	Process row failed for function [TO_TIMESTAMP_TZ]: Failed to convert the string to timestamp with time zone value. Verify that the specified date format string is valid. Verify that the timestamp with time zone string used in the first argument is compatible with the specified date format.

TRUNC (Fechas)

Además, trunca fechas hasta un año, mes, día, hora, minuto, segundo, milisegundo o microsegundo determinados. También se puede usar TRUNC para truncar números.

Se pueden truncar las siguientes partes de fechas:

- **Año.** Si se trunca la parte del año de la fecha, la función devuelve Ene 1 del año introducido, con la hora fijada en 00:00:00.000000000. Por ejemplo, la siguiente expresión devuelve 1/1/1997 00:00:00.000000000:
`TRUNC(12/1/1997 3:10:15, 'YY')`
- **Mes.** Si se trunca la parte del mes de la fecha, la función devuelve el primer día del mes, con la hora fijada en 00:00:00.000000000. Por ejemplo, la siguiente expresión devuelve 01/04/1997 00:00:00.000000000:
`TRUNC(4/15/1997 12:15:00, 'MM')`
- **Día.** Si se trunca la parte del día de la fecha, la función devuelve la fecha, con la hora fijada en 00:00:00.000000000. Por ejemplo, la siguiente expresión devuelve 13/06/1997 00:00:00.000000000:
`TRUNC(6/13/1997 2:30:45, 'DD')`
- **Hora.** Si se trunca la parte de la hora de una fecha, la función devuelve la fecha, con los minutos, segundos y subsegundos fijados en 0. Por ejemplo, la siguiente expresión devuelve 01/04/1997 11:00:00.000000000:
`TRUNC(4/1/1997 11:29:35, 'HH')`
- **Minuto.** Si se trunca la parte del minuto de una fecha, la función devuelve la fecha, con los segundos y los subsegundos fijados en 0. Por ejemplo, la siguiente expresión devuelve 22/05/1997 10:15:00.000000000:
`TRUNC(5/22/1997 10:15:29, 'MI')`
- **Segundo.** Si se trunca la segunda porción de una fecha, la función devuelve la fecha, con los milisegundos fijados en 0. Por ejemplo, la siguiente expresión devuelve 22/05/1997 10:15:29.000000000:
`TRUNC(5/22/1997 10:15:29.135, 'SS')`
- **Milisegundo.** Si se trunca la porción de milisegundos de una fecha, la función devuelve la fecha, con los microsegundos fijados en 0. Por ejemplo, la siguiente expresión devuelve 22/05/1997 10:15:30.135000000:
`TRUNC(5/22/1997 10:15:30.135235, 'MS')`

- **Microsegundo.** Si se trunca la porción de microsegundos de una fecha, la función devuelve la fecha, con los nanosegundos fijados en 0. Por ejemplo, la siguiente expresión devuelve 22/05/1997 10:15:30.135235000:

```
TRUNC(5/22/1997 10:15:29.135235478, 'US')
```

Sintaxis

```
TRUNC( date [,format] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>date</i>	Obligatorio	Tipo de datos Fecha/Hora. Los valores de fecha que se desea truncar. Se puede introducir cualquier expresión de transformación válida que dé como valor una fecha.
<i>format</i>	Opcional	Introduzca una cadena de formato válida. La cadena de formato no es sensible a mayúsculas y minúsculas. Si se omite la cadena de formato, la función trunca la porción de la hora, fijándola en 00:00:00.000000000.

Valor de retorno

Fecha.

NULL si el valor pasado a la función es NULL.

Ejemplos

Las siguientes expresiones devuelven la porción del años de las fechas en el puerto DATE_SHIPPED:

```
TRUNC( DATE_SHIPPED, 'Y' )
TRUNC( DATE_SHIPPED, 'YY' )
TRUNC( DATE_SHIPPED, 'YYY' )
TRUNC( DATE_SHIPPED, 'YYYY' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 1 1998 12:00:00.000000000
Apr 19 1998 1:31:20PM	Jan 1 1998 12:00:00.000000000
Jun 20 1998 3:50:04AM	Jan 1 1998 12:00:00.000000000
Dec 20 1998 3:29:55PM	Jan 1 1998 12:00:00.000000000
NULL	NULL

Las siguientes expresiones devuelven la porción del mes de las fechas en el puerto DATE_SHIPPED:

```
TRUNC( DATE_SHIPPED, 'MM' )
TRUNC( DATE_SHIPPED, 'MON' )
TRUNC( DATE_SHIPPED, 'MONTH' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 1 1998 12:00:00.000000000AM

DATE_SHIPPED	RETURN VALUE
Apr 19 1998 1:31:20PM	Apr 1 1998 12:00:00.000000000AM
Jun 20 1998 3:50:04AM	Jun 1 1998 12:00:00.000000000AM
Dec 20 1998 3:29:55PM	Dec 1 1998 12:00:00.000000000AM
NULL	NULL

Las siguientes expresiones devuelven la porción del día de las fechas en el puerto DATE_SHIPPED:

```
TRUNC( DATE_SHIPPED, 'D' )
TRUNC( DATE_SHIPPED, 'DD' )
TRUNC( DATE_SHIPPED, 'DDD' )
TRUNC( DATE_SHIPPED, 'DY' )
TRUNC( DATE_SHIPPED, 'DAY' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 15 1998 12:00:00.000000000AM
Apr 19 1998 1:31:20PM	Apr 19 1998 12:00:00.000000000AM
Jun 20 1998 3:50:04AM	Jun 20 1998 12:00:00.000000000AM
Dec 20 1998 3:29:55PM	Dec 20 1998 12:00:00.000000000AM
Dec 31 1998 11:59:59PM	Dec 31 1998 12:00:00.000000000AM
NULL	NULL

Las siguientes expresiones devuelven la porción de la hora de las fechas en el puerto DATE_SHIPPED:

```
TRUNC( DATE_SHIPPED, 'HH' )
TRUNC( DATE_SHIPPED, 'HH12' )
TRUNC( DATE_SHIPPED, 'HH24' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:31AM	Jan 15 1998 2:00:00.000000000AM
Apr 19 1998 1:31:20PM	Apr 19 1998 1:00:00.000000000PM
Jun 20 1998 3:50:04AM	Jun 20 1998 3:00:00.000000000AM
Dec 20 1998 3:29:55PM	Dec 20 1998 3:00:00.000000000PM
Dec 31 1998 11:59:59PM	Dec 31 1998 11:00:00.000000000AM
NULL	NULL

Las siguientes expresiones devuelven la porción del minuto de las fechas en el puerto DATE_SHIPPED:

```
TRUNC( DATE_SHIPPED, 'MI' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 15 1998 2:10:00.000000000AM
Apr 19 1998 1:31:20PM	Apr 19 1998 1:31:00.000000000PM
Jun 20 1998 3:50:04AM	Jun 20 1998 3:50:00.000000000AM
Dec 20 1998 3:29:55PM	Dec 20 1998 3:29:00.000000000PM
Dec 31 1998 11:59:59PM	Dec 31 1998 11:59:00.000000000PM
NULL	NULL

TRUNC (Números)

Trunca los números en un dígito específico. También puede utilizar TRUNC para truncar fechas.

Sintaxis

```
TRUNC( numeric_value [, precision] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de dato numérico. Pasa los valores que desea truncar. Puede introducir cualquier expresión de transformación válida que dé como resultado un tipo de datos numérico.
<i>precision</i>	Opcional	Puede ser un número entero positivo o negativo. Puede introducir cualquier expresión de transformación válida que dé como resultado un número entero. El número entero especifica el número de dígitos para truncar.

Si *precision* es un entero positivo, TRUNC devuelve *numeric_value* con el número de decimales especificados por *precision*. Si *precision* es un entero negativo, TRUNC cambia los dígitos especificados a la izquierda del punto decimal por ceros. Si omite el argumento *precision*, TRUNC trunca la parte decimal de *numeric_value* y devuelve un entero.

Si pasa un valor de *precision* decimal, el Servicio de integración de datos redondea *numeric_value* al entero más cercano antes de evaluar la expresión.

Cuando ejecute una asignación en el modo de alta precisión, utilice la función ROUND antes de realizar el truncamiento.

Por ejemplo, supongamos que la siguiente expresión se utiliza para truncar los valores del puerto QTY:

```
TRUNC ( QTY / 15 )
```

Cuando el valor para QTY = 15000000, la sesión devuelve el valor 999999. El resultado esperado es 1000000.

PRICE	RETURN VALUE
1235.99	1230.0
NULL	NULL
TRUNC(PRICE)	

PRICE	RETURN VALUE
12.99	12.0
-18.99	-18.0
56.95	56.0
15.99	15.0
NULL	NULL

UPPER

Convierte a mayúsculas los caracteres de cadenas en minúsculas.

Sintaxis

```
UPPER( string )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>string</i>	Obligatorio	Tipo de datos String. Pasa los valores que se van a cambiar a texto en mayúsculas. Puede especificar cualquier expresión de transformación válida.

Valor devuelto

Cadena en mayúsculas. Si los datos contienen caracteres multibyte, el valor devuelto depende de la página de códigos y el modo de movimiento de datos del Servicio de integración de datos.

Se devuelve NULL si el valor pasado a la función es NULL.

Ejemplo

La siguiente expresión cambia todos los nombres del puerto FIRST_NAME a mayúsculas:

```
UPPER( FIRST_NAME )
```

FIRST_NAME	RETURN VALUE
Ramona	RAMONA

FIRST_NAME	RETURN VALUE
NULL	NULL
THOMAS	THOMAS
PierRe	PIERRE
Bernice	BERNICE

UUID4

Devuelve un valor binario de 16 bytes generado aleatoriamente que se ajusta a la variante 4 de la especificación de UUID descrita en RFC 4122. UUID4 no toma un argumento.

Sintaxis

```
UUID4()
```

Valor devuelto

Binario.

UUID4 nunca devuelve un valor nulo o un error.

UUID_UNPARSE

Convierte un valor binario de 16 bytes en una representación de cadena de 36 caracteres como se define en RFC 4122.

Sintaxis

```
UUID_UNPARSE( binary )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>binario</i>	Obligatorio	Tipo de datos binario. Cualquier valor binario de 16 bytes que desea convertir en una cadena de 36 caracteres.

Valor devuelto

Cadena de 36 caracteres.

Devuelve un resultado nulo si el argumento es nulo y un error si el argumento no es un valor binario de 16 bytes.

Ejemplo

La siguiente expresión puede devolver el valor 6948DF80-14BD-4E04-8842-7668D9C001F5:

```
UUID_UNPARSE (UUID4 ( ) )
```

VARIANCE

Devuelve la varianza de un valor que se pasa a la función. VARIANCE se utiliza para analizar datos estadísticos. Dentro de VARIANCE solamente se puede anidar una función agregada adicional y la función anidada debe devolver un tipo de datos Numérico.

Sintaxis

```
VARIANCE( numeric_value [, filter_condition ] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio/ Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. Pasa los valores para los que desea calcular una varianza. Se puede especificar cualquier expresión de transformación válida.
<i>filter_condition</i>	Opcional	Limita las filas en la búsqueda. La condición de filtro debe ser un valor numérico o dar un valor TRUE, FALSE o NULL. Se puede especificar cualquier expresión de transformación válida.

Valor devuelto

Valor doble.

NULL si todos los valores pasados a la función son NULL, o si no se ha seleccionado ninguna fila (por ejemplo, la *filter_condition* da un valor FALSE o NULL para todas las filas).

Nulos

Si un valor es NULL, VARIANCE lo omite. Sin embargo, si todos los valores pasados a la función son NULL o si no se ha seleccionado ninguna fila, VARIANCE devuelve NULL.

Agrupar por

VARIANCE agrupa valores por grupos según los puertos definidos en la transformación, devolviendo un resultado para cada grupo.

Si no hay un grupo por puerto, VARIANCE trata todas las filas como un solo grupo, devolviendo un valor.

Ejemplo

La siguiente expresión calcula la variación para todas las filas en el puerto TOTAL_SALES:

```
VARIANCE( TOTAL_SALES )
```

TOTAL_SALES

2198.0

TOTAL_SALES

2256.0

3001.0

NULL

8953.0

RETURN VALUE: 10592444.6666667

INDICE

A

- actualizaciones de lenguaje de transformación
 - expresiones booleanas [20](#)
- actualizaciones del idioma de transformación
 - expresiones de comparación [20](#)
- algoritmo del estándar de cifrado avanzado
 - descripción [63](#)
- Algoritmo estándar de cifrado avanzado
 - descripción [63](#)
- alta precisión
 - ABS [59](#)
 - AVG [68](#)
 - CEIL [70](#)
 - CUME [85](#)
 - EXP [96](#)
 - función ABS [59](#)
 - función AVG [68](#)
 - función CREATE_TIMESTAMP_TZ [84](#)
 - Función CUME [85](#)
 - función GET_TIMESTAMP [103](#)
 - función GET_TIMEZONE [102](#)
 - Función MAX [140](#)
 - Función MEDIAN [143](#)
 - Función MIN [149](#)
 - Función MOVINGAVG [153](#)
 - Función MOVINGSUM [155](#)
 - función PERCENTILE [160](#)
 - Función ROUND [183](#)
 - Función STDDEV [199](#)
 - Función SUM [205](#)
 - función TO_TIMESTAMP_TZ [227](#)
 - función TO_DECIMAL [222](#)
 - función TO_DECIMAL38 [223](#)
 - función TRUNC [231](#)
 - LOG [129](#)
 - MAX (números) [140](#)
 - MEDIAN [143](#)
 - MIN (números) [149](#)
 - MOD [152](#)
 - MOVINGAVG [153](#)
 - MOVINGSUM [155](#)
 - operadores aritméticos [31](#)
 - PERCENTILE [160](#)
 - POWER [163](#)
 - ROUND (números) [183](#)
 - SIGN [191](#)
 - SIN [192](#)
 - SUM [205](#)
- AND
 - palabra reservada [15](#)
- año 2000
 - fechas [38](#)
- archivos sin formato
 - fechas [40](#)

- aritméticos
 - valores de fecha y hora [49](#)
- array
 - generación [137](#), [138](#)
- ASCII
 - conversión de valores ASCII [72](#)
 - convirtiendo a valores Unicode [73](#)
 - convirtiendo caracteres en valores ASCII [66](#)
 - función CHR [72](#)
- asignación
 - generación [74](#), [135](#), [136](#)

B

- bases de datos relacionales
 - fechas [40](#)
- bigint
 - conversión de valores a [210](#)
- búsquedas múltiples
 - ejemplo de constante TRUE [21](#)

C

- cadena de formato J
 - usar con IS_DATE [48](#)
 - usar con TO_CHAR [44](#)
 - usar con TO_DATE [48](#)
- cadena de formato RR
 - descripción [38](#)
 - diferencia entre YY y RR [39](#)
 - usar con IS_DATE [48](#)
 - usar con TO_CHAR [45](#)
 - usar con TO_DATE [48](#)
- cadena de formato SSSSS
 - usar con IS_DATE [48](#)
 - usar con TO_CHAR [44](#)
 - usar con TO_DATE [48](#)
- cadena de formato YY
 - diferencia entre RR e YY [39](#)
 - usar con IS_DATE [48](#)
 - usar con TO_CHAR [45](#)
 - usar con TO_DATE [48](#)
- cadenas
 - adición de caracteres [131](#)
 - adición de espacios en blanco [131](#)
 - concatenación [76](#)
 - concatenar [32](#)
 - conversión de fechas a caracteres [212](#)
 - conversión de longitud [186](#)
 - conversión de valores numéricos a cadenas de texto [217](#)
 - convirtiendo cadenas de caracteres en fechas [218](#)
 - devolución de porción [203](#)
 - eliminación de caracteres [133](#)
 - eliminación de espacios en blanco [133](#)

- cadenas (*continuado*)
 - eliminación de espacios en blanco y caracteres [187](#)
 - juego de caracteres [111](#)
 - número de caracteres [127](#)
 - sustitución de un carácter [171](#)
 - sustitución de varios caracteres [174](#)
 - uso de mayúsculas [110](#), [130](#), [233](#)
- cadenas con formato
 - coincidencia [47](#)
- cadenas de caracteres
 - conversión de fechas [212](#)
 - convirtiendo a fechas [218](#)
- cadenas de formato
 - definición [37](#)
 - día juliano [45](#)
 - día juliano modificado [45](#)
 - fecha juliana [42](#)
 - fecha juliana modificada [42](#)
 - fechas [41](#)
 - función IS_DATE [45](#)
 - función TO_CHAR [42](#)
 - función TO_DATE [45](#)
- cadenas de texto
 - conversión de valores numéricos [217](#)
- cadenas vacías
 - comprobar [127](#)
- cálculo de división
 - devolución de resto [152](#)
- calendario gregoriano
 - en funciones de fechas [38](#)
- calendarios
 - tipos de fechas admitidas [38](#)
- calificador de referencia:INFA
 - palabra reservada [15](#)
- calificador de referencia:MCR
 - palabra reservada [15](#)
- calificador de referencia:TYPE
 - palabra reservada [15](#)
- calificadores de referencia
 - descripción [13](#)
- caracteres
 - adición a cadenas [131](#)
 - agregar a cadenas [186](#)
 - caracteres ASCII [66](#), [72](#)
 - caracteres Unicode [66](#), [72](#), [73](#)
 - codificación [195](#)
 - devolver número [127](#)
 - eliminación en cadenas [187](#)
 - recuento [203](#)
 - sustitución de un [171](#)
 - uso de mayúsculas [110](#), [130](#), [233](#)
- Caracteres
 - codificación [145](#)
 - eliminación en cadenas [133](#)
 - sustitución de varios caracteres [174](#)
- cifrado
 - función AES_ENCRYPT [63](#)
 - utilización del algoritmo estándar de cifrado avanzado [63](#)
- codificación
 - caracteres [195](#)
 - Caracteres [145](#)
 - función ENC_BASE64 [94](#)
- comentarios
 - agregar a las expresiones [15](#)
- comillas
 - insertar simples mediante función CHR [13](#)
- comillas simples en literales de cadena
 - función CHR [72](#)

- comillas simples en literales de cadena (*continuado*)
 - uso de funciones CHR y CONCAT [76](#)
- componentes del idioma de transformación
 - resumen [11](#)
- compresión
 - comprimir datos [75](#)
 - descompresión de datos [94](#)
- concatenación
 - cadenas [76](#)
- concatenar
 - cadenas [32](#)
- condiciones de filtro
 - funciones de agregado [52](#)
 - valores nulos [20](#)
- constante DD_DELETE
 - descripción [17](#)
 - ejemplo de estrategia de actualización [17](#)
 - palabra reservada [15](#)
- constante DD_INSERT
 - descripción [17](#)
 - ejemplo de estrategia de actualización [17](#)
 - palabra reservada [15](#)
- constante DD_REJECT
 - descripción [18](#)
 - ejemplo de estrategia de actualización [18](#)
 - palabra reservada [15](#)
- constante DD_UPDATE
 - descripción [19](#)
 - ejemplo de estrategia de actualización [19](#)
 - palabra reservada [15](#)
- constante FALSE
 - descripción [19](#)
 - palabra reservada [15](#)
- constante NULL
 - descripción [20](#)
 - palabra reservada [15](#)
- constante TRUE
 - descripción [21](#)
 - palabra reservada [15](#)
- constantes
 - DD_INSERT [17](#)
 - DD_REJECT [18](#)
 - DD_UPDATE [19](#)
 - descripción [11](#)
 - FALSE [19](#)
 - NULL [20](#)
 - TRUE [21](#)
- conversión
 - cadenas de fechas [38](#)
- conversión de cadena
 - fechas [38](#)
- coseno
 - calcular [79](#)
 - calcular coseno hiperbólico [80](#)

D

- datos jerárquicos
 - acceder a elementos [23](#)
 - cómo analizar [157](#), [158](#)
 - cómo generar [66](#), [74](#), [157](#), [158](#), [201](#), [202](#)
 - generación [74](#), [135](#)–[138](#)
- decodificación
 - función DEC_BASE64 [91](#)
- descifrado
 - función AES_DECRYPT [63](#)

- desviación estándar
 - devolviendo [199](#)
- día juliano
 - cadena de formato [45](#)
- día juliano modificado
 - cadena de formato [45](#)

E

- enteros
 - conversión de valores a [225](#)
- espacios
 - evitar en filas [120](#)
 - quitar con DD_REJECT [18](#)
- estrategia de actualización
 - ejemplo de DD_DELETE [17](#)
 - ejemplo de DD_INSERT [17](#)
 - ejemplo de DD_REJECT [18](#)
 - ejemplo de DD_UPDATE [19](#)
- expresiones
 - agregar comentarios [15](#)
 - condicional [19](#)
 - resumen [11](#)
 - sintaxis [12](#)
 - utilización de operadores [22](#)
- expresiones anidadas
 - operadores [22](#)
- expresiones de transformación
 - restricciones de valores nulos [20](#)
 - resumen [11](#)

F

- fecha juliana
 - cadena de formato [42](#)
- fecha juliana modificada
 - cadena de formato [42](#)
- fechas
 - año 2000 [38](#)
 - archivos sin formato [40](#)
 - bases de datos relacionales [40](#)
 - cadenas de formato [41](#)
 - conversión a cadenas de caracteres [212](#)
 - formato de fecha y hora predeterminado [40](#)
 - funciones [55](#)
 - introducción [37](#)
 - Juliano [38](#)
 - juliano modificado [38](#)
 - realizar operaciones aritméticas [49](#)
 - redondeo [179](#)
 - truncando [228](#)
- fechas julianas
 - en funciones de fechas [38](#)
- filas
 - devolución de la primera fila [97](#)
 - devolución de promedio [153](#)
 - devolución de suma [155](#)
 - devolver cualquier fila [64](#)
 - devolver la última fila [122](#)
 - evitar espacios [120](#)
 - omisión [95](#)
 - recuento [81](#)
 - total de ejecución [85](#)
- formato
 - de cadena de caracteres a fecha [218](#)
 - de fecha a cadena de caracteres [212](#)

- formato de fecha y hora predeterminado
 - configuración [40](#)
- Función ABORT
 - descripción [58](#)
- función ABS
 - descripción [59](#)
- función ADD_TO_DATE
 - Descripción [60](#)
- función AES_DECRYPT
 - descripción [63](#)
- función AES_ENCRYPT
 - descripción [63](#)
- Función ANY
 - descripción [64](#)
- función ARRAY
 - descripción [66](#)
- Función ASCII
 - descripción [66](#)
- función AVG
 - descripción [68](#)
- función CAST
 - descripción [69](#)
- función CEIL
 - descripción [70](#)
- función CHOOSE
 - descripción [71](#)
- función CHR
 - descripción [72](#)
 - inserción de comillas simples [72](#)
 - insertar comillas simples [13](#)
- Función CHRCODE
 - descripción [73](#)
- función COLLECT_LIST
 - descripción [74](#)
- función COLLECT_MAP
 - descripción [74](#)
- función COMPRESS
 - descripción [75](#)
- función CONCAT
 - descripción [76](#)
 - inserción de comillas simples mediante [76](#)
- función CONCAT_ARRAY
 - descripción [78](#)
- función CONVERT_BASE
 - descripción [78](#)
- función COS
 - descripción [79](#)
- función COSH
 - descripción [80](#)
- Función COUNT
 - descripción [81](#)
- función CRC32
 - descripción [84](#)
- función CREATE_TIMESTAMP_TZ
 - descripción [84](#)
- Función CUME
 - descripción [85](#)
- función DATE_COMPARE
 - descripción [87](#)
- Función DATE_DIFF
 - descripción [88](#)
- función DEC_BASE64
 - descripción [91](#)
- función DECODE
 - Descripción [92](#)
 - internacionalización [12](#)
- función DECOMPRESS
 - descripción [94](#)

función ENC_BASE64
 descripción [94](#)
 función ERROR
 Descripción [95](#)
 valor predeterminado [95](#)
 función EXP
 descripción [96](#)
 Función FIRST
 descripción [97](#)
 función FLOOR
 descripción [99](#)
 función FLOOR (expresiones)
 descripción [99](#)
 Función FV
 descripción [100](#)
 Función GET_DATE_PART
 descripción [101](#)
 función GET_TIMESTAMP
 descripción [103](#)
 función GET_TIMEZONE
 descripción [102](#)
 función GREATEST
 descripción [104](#)
 función IIF
 Descripción [105](#)
 internacionalización [12](#)
 función IN
 descripción [108](#)
 función INDEXOF
 descripción [109](#)
 función INITCAP
 descripción [110](#)
 internacionalización [12](#)
 función INSTR
 Descripción [111](#)
 función IS_DATE
 cadenas de formato [45](#)
 Descripción [116](#)
 función IS_NUMBER
 descripción [118](#)
 función IS_SPACES
 descripción [120](#)
 función ISNULL
 descripción [114](#)
 función LAG
 descripción [121](#)
 función LAST
 descripción [122](#)
 función LAST_DAY
 descripción [123](#)
 función LEAD
 descripción [125](#)
 función LEAST
 descripción [126](#)
 función LENGTH
 comprobación de cadenas vacías [127](#)
 descripción [127](#)
 función LN
 descripción [128](#)
 función LOG
 descripción [129](#)
 función LOWER
 descripción [130](#)
 internacionalización [12](#)
 Función LPAD
 descripción [131](#)
 función LTRIM
 Descripción [133](#)
 función MAKE_DATE_TIME
 descripción [134](#)
 función MAP
 descripción [135](#)
 función MAP_FROM_ARRAYS
 descripción [136](#)
 función MAP_KEYS
 descripción [137](#)
 función MAP_VALUES
 descripción [138](#)
 Función MAX (cadena)
 descripción [141](#)
 función MAX (fechas)
 descripción [139](#)
 internacionalización [12](#)
 función MAX (números)
 internacionalización [12](#)
 Función MAX (números)
 descripción [140](#)
 Función MD5
 descripción [143](#)
 Función MEDIAN
 descripción [143](#)
 función MIN (fechas)
 descripción [148](#)
 internacionalización [12](#)
 función MIN (números)
 internacionalización [12](#)
 Función MIN (números)
 descripción [149, 151](#)
 función MOD
 descripción [152](#)
 Función MOVINGAVG
 Descripción [153](#)
 Función MOVINGSUM
 Descripción [155](#)
 Función NPER
 descripción [156](#)
 función PARSE_JSON
 descripción [157](#)
 función PARSE_XML
 descripción [158](#)
 función PERCENTILE
 Descripción [160](#)
 función PMT
 descripción [162](#)
 función POWER
 descripción [163](#)
 función PV
 descripción [164](#)
 función RAND
 descripción [164](#)
 función RATE
 descripción [165](#)
 Función REG_EXTRACT
 Descripción [166](#)
 uso de la sintaxis de perl [166](#)
 función REG_MATCH
 uso de la sintaxis de perl [166](#)
 Función REG_MATCH
 descripción [168](#)
 función REG_REPLACE
 descripción [170](#)
 Función REPLACECHR
 descripción [171](#)
 función REPLACESTR
 Descripción [174](#)

- función RESPEC
 - descripción [177](#)
- función REVERSE
 - descripción [178](#)
- función ROUND (fechas)
 - Descripción [179](#)
 - proceso de subsegundos [179](#)
- Función ROUND (números)
 - descripción [183](#)
- función RPAD
 - descripción [186](#)
- función RTRIM
 - Descripción [187](#)
- función SET_DATE_PART
 - Descripción [188](#)
- función SIGN
 - descripción [191](#)
- función SIN
 - descripción [192](#)
- función SINH
 - descripción [193](#)
- función SIZE
 - descripción [194](#)
- Función SOUNDEX
 - descripción [195](#)
- Función SQL IS_CHAR
 - uso de REG_MATCH [168](#)
- Función SQL LIKE
 - uso de REG_MATCH [168](#)
- función SQL_LIKE
 - descripción [197](#)
- función SQRT
 - descripción [198](#)
- Función STDDEV
 - descripción [199](#)
- función STRUCT
 - descripción [201](#)
- función STRUCT_AS
 - descripción [202](#)
- Función SUBSTR
 - descripción [203](#)
- Función SUM
 - descripción [205](#)
- función SYSTIMESTAMP
 - descripción [206](#)
- función TAN
 - descripción [208](#)
- función TANH
 - descripción [208](#)
- función TO__TIMESTAMP_TZ
 - descripción [227](#)
- función TO_CHAR (fechas)
 - cadenas de formato [42](#)
 - descripción [212](#)
 - ejemplos [44](#)
- función TO_CHAR (números)
 - descripción [217](#)
- función TO_DATE
 - cadenas de formato [45](#)
 - ejemplos [47](#)
- Función TO_DATE
 - descripción [218](#)
- función TO_DECIMAL
 - descripción [222](#)
- función TO_DECIMAL38
 - descripción [223](#)
- función TO_FLOAT
 - descripción [224](#)

- función TO_INTEGER
 - descripción [225](#)
- Función TRUNC (fechas)
 - descripción [228](#)
 - procesamiento de subsegundos [228](#)
- función TRUNC (números)
 - descripción [231](#)
- función UPPER
 - descripción [233](#)
 - internacionalización [12](#)
- Función UUID_UNPARSE
 - descripción [234](#)
- Función UUID4
 - descripción [234](#)
- Función VARIANCE
 - descripción [235](#)
- funciones
 - agregado [50](#)
 - cadena [57](#)
 - caracteres [52](#)
 - categorías [50](#)
 - científicas [57](#)
 - codificación [55](#)
 - complejas [53](#)
 - conversión [54](#)
 - descripción [11](#)
 - especiales [57](#)
 - fecha [55](#)
 - financieras [56](#)
 - internacionalización [12](#)
 - limpieza de datos [54](#)
 - numéricas [56](#)
 - prueba [57](#)
 - ventana [58](#)
- funciones agregadas
 - AVG [68](#)
 - COUNT [81](#)
 - FIRST [97](#)
 - MAX (cadena) [141](#)
 - MAX (números) [140](#)
 - MEDIAN [143](#)
 - MIN (números) [149, 151](#)
 - STDDEV [199](#)
 - SUM [205](#)
- funciones científicas
 - COS [79](#)
 - COSH [80](#)
 - descripción [57](#)
 - SIN [192](#)
 - SINH [193](#)
 - TAN [208](#)
 - TANH [208](#)
- funciones complejas
 - ARRAY [66](#)
 - CAST [69](#)
 - COLLECT_LIST [74](#)
 - COLLECT_MAP [74](#)
 - CONCAT_ARRAY [78](#)
 - descripción [53](#)
 - función PARSE_JSON [157](#)
 - función PARSE_XML [158](#)
 - MAP [135](#)
 - MAP_FROM_ARRAYS [136](#)
 - MAP_KEYS [137](#)
 - MAP_VALUES [138](#)
 - RESPEC [177](#)
 - SIZE [194](#)
 - STRUCT [201](#)

funciones complejas (*continuado*)

STRUCT_AS [202](#)

funciones de agregado

ANY [64](#)

descripción [50](#)

LAST [122](#)

MAX (fechas) [139](#)

MIN (fechas) [148](#)

PERCENTILE [160](#)

valores nulos [20](#), [52](#)

VARIANCE [235](#)

funciones de cadena

CHOOSE [71](#)

descripción [57](#)

INDEXOF [109](#)

REVERSE [178](#)

funciones de carácter

ASCII [66](#)

CHR [72](#)

CHRCODE [73](#)

función CONCAT [76](#)

INITCAP [110](#)

INSTR [111](#)

LPAD [131](#)

LTRIM [133](#)

REG_EXTRACT [166](#)

REG_MATCH [168](#)

REG_REPLACE [170](#)

REPLACECHR [171](#)

REPLACESTR [174](#)

RPAD [186](#)

RTRIM [187](#)

SUBSTR [203](#)

funciones de caracteres

LENGTH [127](#)

lista de [52](#)

LOWER [130](#)

METAPHONE [145](#)

SOUNDEX [195](#)

UPPER [233](#)

funciones de codificación

AES_DECRYPT [63](#)

AES_ENCRYPT [63](#)

COMPRESS [75](#)

CRC32 [84](#)

DEC_BASE64 [91](#)

DECOMPRESS [94](#)

descripción [55](#)

ENC_BASE64 [94](#)

MD5 [143](#)

funciones de conversión

CREATE_TIMESTAMP_TZ [84](#)

descripción [54](#)

GET_TIMESTAMP [103](#)

GET_TIMEZONE [102](#)

TO_CHAR (fechas) [212](#)

TO_CHAR (números) [217](#)

TO_DATE [218](#)

TO_DECIMAL [222](#)

TO_DECIMAL38 [223](#)

TO_FLOAT [224](#)

TO_INTEGER [225](#)

TO_TIMESTAMP_TZ [227](#)

funciones de fecha

ADD_TO_DATE [60](#)

DATE_COMPARE [87](#)

DATE_DIFF [88](#)

GET_DATE_PART [101](#)

funciones de fecha (*continuado*)

ROUND [179](#)

SET_DATE_PART [188](#)

TRUNC (Fechas) [228](#)

funciones de fechas

LAST_DAY [123](#)

MAKE_DATE_TIME [134](#)

MAX (fechas) [139](#)

MIN (fechas) [148](#)

SYSTIMESTAMP [206](#)

funciones de limpieza de datos

descripción [54](#)

GREATEST [104](#)

IN [108](#)

LEAST [126](#)

funciones de prueba

descripción [57](#)

IS_DATE [116](#)

IS_NUMBER [118](#)

IS_SPACES [120](#)

ISNULL [114](#)

funciones de ventana

descripción [58](#)

LAG [121](#)

LEAD [125](#)

funciones especiales

ABORT [58](#)

DECODE [92](#)

descripción [57](#)

ERROR [95](#)

IIF [105](#)

funciones financieras

descripción [56](#)

Función FV [100](#)

Función NPV [156](#)

función PMT [162](#)

función PV [164](#)

función RATE [165](#)

funciones numéricas

ABS [59](#)

CEIL [70](#)

CONVERT_BASE [78](#)

CUME [85](#)

descripción [56](#)

EXP [96](#)

FLOOR [99](#)

LN [128](#)

LOG [129](#)

MOD [152](#)

MOVINGAVG [153](#)

MOVINGSUM [155](#)

POWER [163](#)

RAND [164](#)

ROUND (números) [183](#)

SIGN [191](#)

SQRT [198](#)

TRUNC (números) [231](#)

H

hiperbólica

función de coseno [80](#)

función de seno [193](#)

función de tangente [208](#)

I

- idioma de transformación
 - operadores [22](#)
 - palabras reservadas [15](#)
- internacionalización
 - expresión no válida [12](#)
 - funciones afectadas [12](#)
 - orden de clasificación [12](#)

L

- calificador de referencia:LKP
 - descripción [13](#)
 - palabra reservada [15](#)
- lenguaje de transformación
 - comparación con SQL [12](#)
- literales
 - comillas simples en [72](#), [76](#)
 - requisito de comillas simples [13](#)
- literales de cadena
 - comillas simples en [72](#), [76](#)
 - requisito de comillas simples [13](#)
- logaritmo
 - devolver [128](#), [129](#)

M

- matriz
 - cómo generar [66](#), [74](#)
- mayúsculas y minúsculas
 - convertir a mayúsculas [233](#)
- mes
 - devolver el último día [123](#)
- METAPHONE
 - descripción [145](#)
- mínimo
 - valor, devolución [148](#)

N

- NOT
 - palabra reservada [15](#)
- números
 - redondeo [183](#)
 - truncando [231](#)

O

- omisión
 - filas [95](#)
- operador de punto
 - descripción [25](#)
 - para tipos de datos complejos [23](#)
 - utilizar para acceder a datos [25](#)
- operador de subíndice
 - para tipos de datos complejos [23](#)
 - tipo de datos de asignación [24](#)
 - tipo de datos de matriz [24](#)
 - utilizar para acceder a datos [24](#)
- operadores
 - aritméticos [31](#)
 - complejas [23](#)
 - descripción [11](#)

- operadores (*continuado*)
 - operadores de cadena [32](#)
 - operadores de comparación [33](#)
 - operadores lógicos [34](#)
 - utilización de cadenas en aritmética [31](#)
 - utilización de cadenas en comparaciones [33](#)
 - valores nulos [21](#)
- operadores aritméticos
 - descripción [31](#)
 - utilización de cadenas en expresiones [31](#)
 - utilización para convertir datos [31](#)
- operadores complejos
 - acceder a tipos de datos anidados [27](#)
 - descripción [23](#)
 - para estructura con elementos de estructura [31](#)
 - para estructura con elementos de matriz [30](#)
 - para matrices multidimensionales [27](#)
 - para matriz con elementos de estructura [29](#)
 - para tipos de datos anidados [27](#)
 - utilizar para acceder a datos [23](#)
- operadores de cadena
 - descripción [32](#)
- operadores de comparación
 - descripción [33](#)
 - utilización de cadenas en expresiones [33](#)
- operadores de punto
 - para el tipo de datos anidados [27](#)
 - para estructura con elementos de estructura [31](#)
- operadores de subíndice
 - para el tipo de datos anidados [27](#)
 - para matrices multidimensionales [27](#)
- operadores lógicos
 - descripción [34](#)
- OR
 - palabra reservada [15](#)
- orden de clasificación
 - internacionalización [12](#)

P

- palabras reservadas
 - lista [15](#)
- parámetros de asignación
 - definición [11](#)
- precedencia del operador
 - expresiones [22](#)
- promedios
 - devolviendo [153](#)
 - funciones agregadas para determinar [68](#)
- puertos
 - sintaxis [13](#)

R

- raíz cuadrada
 - devolver [198](#)
- redondeo
 - fechas [179](#)
 - números [183](#)
- restricción de clave principal
 - valores nulos [20](#)

S

seno
devolver [192](#), [193](#)
Servicio de integración de datos
manejo de valores nulos en expresiones de comparación [20](#)
sintaxis
expresión [12](#)
puertos [13](#)
reglas generales [14](#)
valores de retorno [13](#)
sintaxis de COBOL
conversión a la sintaxis de perl [166](#)
sintaxis de expresión regular compatible con perl
uso en una función REG_EXTRACT [166](#)
uso en una función REG_MATCH [166](#)
sintaxis de SQL
conversión a la sintaxis de perl [166](#)
SPOUTPUT
palabra reservada [15](#)
struct
cómo generar [157](#), [158](#), [201](#), [202](#)
subsegundos
procesamiento en la función TRUNC (fechas) [228](#)
proceso en función ROUND (fechas) [179](#)
suma
devolución [205](#)
devolviendo [155](#)

T

tamaño
asignación [194](#)
matriz [194](#)
tangente
devolver [208](#)
tipos de datos
fecha y hora [37](#)
total de ejecución
devolviendo [85](#)
Transformación de filtro
usar función ISNULL [114](#)
truncando
fechas [228](#)
números [231](#)

U

Unicode
conversión de valores Unicode [72](#)
convirtiendo a valores ASCII [73](#)
convirtiendo caracteres en valores Unicode [66](#)
uso de mayúsculas
cadenas [110](#), [130](#), [233](#)

V

valores absolutos
obtener [59](#)
valores de cadena
devolución del máximo [141](#)

valores de cadena (*continuado*)
devolución del mínimo [151](#)
valores de doble precisión
números de coma flotante [224](#)
valores de exponente
calcular [96](#)
valores de fecha/hora
adición [60](#)
valores de retorno
descripción [11](#)
sintaxis [13](#)
valores decimales
conversión [84](#), [102](#), [103](#), [222](#), [223](#), [227](#)
valores del exponente
devolver [163](#)
valores negativos
SIGN [191](#)
valores nulos
comprobar [114](#)
condiciones de filtro [20](#)
en expresiones de comparación [20](#)
funciones de agregado [20](#), [52](#)
ISNULL [114](#)
operador de cadena [32](#)
operadores [21](#)
operadores lógicos [35](#)
valores numéricos
conversión a cadenas de texto [217](#)
devolución de desviación estándar [199](#)
devolución del mínimo [149](#)
devolver coseno [79](#)
devolver coseno hiperbólico de [80](#)
devolver logaritmos [128](#), [129](#)
devolver raíz cuadrada [198](#)
devolver seno [192](#)
devolver seno hiperbólico [193](#)
devolver tangente [208](#)
devolver tangente hiperbólica [208](#)
devolver valor absoluto [59](#)
SIGN [191](#)
valores positivos
SIGN [191](#)
valores predeterminados
función ERROR [95](#)
variable PROC_RESULT
palabra reservada [15](#)
variable SESSSTARTTIME
usar en funciones de fecha [49](#)
variable SYSDATE
descripción [36](#)
palabra reservada [15](#)
usar en expresiones [36](#)
variables
SYSDATE [36](#)
variables integradas [36](#)
variables de asignación
variables integradas [36](#)
variables del sistema [36](#)
variables integradas
Descripción [36](#)
variables locales
descripción [11](#)