Informatica®
10.2

Referência de Linguagem de Transformação

Informatica Referência de Linguagem de Transformação
10.2
Setembro 2017

Data da Publicação: 2018-10-26

Conteúdo

Prefácio. 10

Recursos da Informatica. 10

Rede da Informatica. 10

Base de Dados de Conhecimento da Informatica. 10

Documentação da Informatica. 10

Matrizes de Disponibilidade de Produto Informatica. 11

Informatica Velocity. 11

Informatica Marketplace. 11

Suporte global a clientes Informatica. 11

Capítulo 1: A Linguagem de Transformação. 12

Visão Geral da Linguagem de Transformação. 12

Componentes da Linguagem de Transformação. 12

Internacionalização e a Linguagem de Transformação. 13

Sintaxe de Expressão. 13

Componentes de Expressão. 13

Regras e Diretrizes para Sintaxe de Expressão. 14

Adicionando comentário a expressões. 15

Palavras reservadas. 16

Capítulo 2: Constantes. 17

DD_DELETE. 17

Exemplo. 17

DD_INSERT. 17

Exemplos. 18

DD_REJECT. 18

Exemplos. 18

DD_UPDATE. 19

Exemplos. 19

FALSE. 19

Exemplo. 19

NULL. 20

Trabalhando com valores Null em expressões booleanas. 20

Valores nulos em expressões de comparação. 20

Valores Null em funções de agregação. 20

Valores Null em condições de filtro. 20

Nulos com Operadores. 21

TRUE. 21

Exemplo. 21

Capítulo 3: Operadores. 22

Precedência de operador. 22
Operadores complexos. 23
Operador de subscrito. 24
Operador de ponto. 25
Operadores complexos para tipos de dados aninhados. 27
Operadores aritméticos. 31
Operadores de string. 32
Nulls. 32
Exemplo. 32
Operadores de comparação. 33
Operadores lógicos. 34
Nulls. 34

Capítulo 4: Variáveis. 35

Variáveis internas 35
SYSDATE. 35
Variáveis Locais. 35

Capítulo 5: Datas. 36

Visão geral das datas. 36
Tipo de Dados de Data/Hora. 36
Dia juliano, dia juliano modificado e calendário gregoriano. 37
Data no ano 2000. 37
Datas em bancos de dados relacionais. 39
Datas em arquivos planos. 39
Formato de data padrão. 39
Strings de Formato de Data. 40
Strings de formato TO_CHAR. 41
Exemplos. 43
Strings de formato TO_DATE e IS_DATE. 44
Regras e diretrizes para strings de formato de data. 46
Exemplo. 46
Noções básicas dos cálculos aritméticos de data. 48

Capítulo 6: Funções. 49

Categorias de função. 49
Funções de agregação. 49
Funções de agregação e Nulls. 51
Funções de caractere. 52
Funções complexas. 52
Funções de conversão. 53

Funções de Limpeza de Dados. 53
Funções de data. 54
Funções de codificação. 54
Funções financeiras. 55
Funções numéricas. 55
Funções científicas. 56
Funções especiais. 56
Funções de string. 56
Funções de teste. 56
Funções de janela. 57
ABORT. 57
ABS. 58
ADD_TO_DATE. 59
AES_DECRYPT. 61
AES_ENCRYPT. 62
ANY. 63
ARRAY. 65
ASCII. 65
AVG. 66
CAST. 68
CEIL. 69
CHOOSE. 69
CHR. 70
CHRCODE. 71
COLLECT_LIST. 72
COMPRESS. 73
CONCAT. 74
CONCAT_ARRAY. 75
CONVERT_BASE. 76
COS. 77
COSH. 78
COUNT. 78
CRC32. 81
CREATE_TIMESTAMP_TZ. 82
CUME. 82
DATE_COMPARE. 84
DATE_DIFF. 85
DEC_BASE64. 88
DECODE. 88
DECOMPRESS. 91
ENC_BASE64. 91
ERROR. 92

EXP. . . . 93
FIRST. . . . 94
FLOOR. . . . 95
FV 96
GET_DATE_PART. . . . 97
GET_TIMEZONE. . . . 99
GET_TIMESTAMP. . . . 100
GREATEST. . . . 101
IIF. . . . 102
IN. . . . 105
INDEXOF. . . . 106
INITCAP. . . . 107
INSTR. . . . 108
ISNULL. . . . 111
IS_DATE. . . . 113
IS_NUMBER. . . . 115
IS_SPACES. . . . 117
LAG. . . . 118
LAST. . . . 119
LAST_DAY. . . . 120
LEAD. . . . 122
LEAST. . . . 123
LENGTH. . . . 124
LN. . . . 125
LOG. . . . 126
LOWER. . . . 127
LPAD. . . . 128
LTRIM. . . . 130
MAKE_DATE_TIME. . . . 131
MAX (Datas). . . . 132
MAX (Números). . . . 133
MAX (String). . . . 135
MD5. . . . 136
MEDIAN. . . . 137
METAPHONE. . . . 138
MIN (Datas). . . . 142
MIN (Números). . . . 143
MIN (String). . . . 144
MOD. . . . 145
MOVINGAVG. . . . 147
MOVINGSUM. . . . 148
NPER. . . . 149

PERCENTILE. 150
PMT. 152
POWER. 153
PV. 154
RAND. 155
RATE. 156
REG_EXTRACT. 156
REG_MATCH. 159
REG_REPLACE. 160
REPLACECHR. 161
REPLACESTR. 164
RESPEC. 167
REVERSE. 168
ROUND (Datas). 169
ROUND (Números). 173
RPAD. 176
RTRIM. 177
SET_DATE_PART. 178
SIGN. 181
SIN. 182
SINH. 183
SIZE. 184
SOUNDEX. 185
SQL_LIKE. 186
SQRT. 187
STDDEV. 188
STRUCT. 190
STRUCT_AS. 191
SUBSTR. 192
SUM. 195
SYSTIMESTAMP. 196
TAN. 197
TANH. 198
TO_BIGINT. 199
TO_CHAR (Datas). 201
TO_CHAR (Números). 205
TO_DATE. 206
TO_DECIMAL. 210
TO_DECIMAL38. 211
TO_FLOAT. 212
TO_INTEGER. 213
TO_TIMESTAMP_TZ. 215

TRUNC (Datas). 216
TRUNC (Números). 219
UPPER. 221
UUID4. 222
UUID_UNPARSE. 222
VARIANCE. 223

Índice. 225

Prefácio

A *Informatica Developer* foi escrita para desenvolvedores responsáveis pela criação de mapeamentos. A *Referência de Linguagem de Transformação do Informatica Developer* presume que você conhece o SQL, conceitos de banco de dados relacional e os requisitos de interface dos aplicativos com suporte.

Recursos da Informatica

Rede da Informatica

A Rede da Informatica hospeda o Suporte Global a Clientes da Informatica, a Base de Dados de Conhecimento da Informatica e outros recursos de produtos. Para acessar a Rede da Informatica, visite https://network.informatica.com.

Como membro, você pode:

- Acessar todos os seus recursos Informatica em um só lugar.
- Pesquisar a Base de Dados de Conhecimento em busca de recursos de produtos, incluindo documentações, perguntas frequentes e práticas recomendadas.
- Visualizar informações sobre disponibilidade de produtos.
- Revisar seus casos de suporte.
- Encontrar a sua Rede de Grupo de Usuários da Informatica local e colaborar com seus colegas.

Base de Dados de Conhecimento da Informatica

Use a Base de Dados de Conhecimento da Informatica para pesquisar a Rede da Informatica em busca de recursos de produtos, como documentações, artigos de instruções, práticas recomendadas e PAMs.

Para acessar a Base de Dados de Conhecimento, visite https://kb.informatica.com. Em caso de dúvidas, comentários ou ideias sobre a Base de Dados de Conhecimento, entre em contato com a equipe da Base de Dados de Conhecimento da Informatica em KB_Feedback@informatica.com.

Documentação da Informatica

Para obter a documentação mais recente do seu produto, navegue pela Base de Dados de Conhecimento da Informatica
em .https://kb.informatica.com/_layouts/ProductDocumentation/Page/ProductDocumentSearch.aspx

Em caso de dúvidas, comentários ou ideias sobre esta documentação, entre em contato com a equipe de Documentação da Informatica pelo e-mail infa_documentation@informatica.com.

Matrizes de Disponibilidade de Produto Informatica

As Matrizes de Disponibilidade de Produto (PAMs) indicam as versões dos sistemas operacionais, os bancos de dados e outros tipos de fontes e destinos de dados com os quais uma versão de produto é compatível. Se você for membro da Rede da Informatica, poderá acessar PAMs em https://network.informatica.com/community/informatica-network/product-availability-matrices.

Informatica Velocity

O Informatica Velocity é uma coleção de dicas e práticas recomendadas desenvolvidas pelos Serviços Profissionais da Informatica. Desenvolvido com base na experiência no mundo real de centenas de projetos de gerenciamento de dados, o Informatica Velocity representa o conhecimento coletivo de nossos consultores, que trabalharam com organizações de todo o mundo para planejar, desenvolver, implantar e manter soluções de gerenciamento de dados bem-sucedidas.

Se você for membro da Rede da Informatica, poderá acessar os recursos do Informatica Velocity em http://velocity.informatica.com.

Se você tiver dúvidas, comentários ou ideias sobre o Informatica Velocity, entre em contato com os Serviços Profissionais da Informatica em ips@informatica.com.

Informatica Marketplace

O Informatica Marketplace é um fórum onde você pode encontrar soluções que aumentam, ampliam ou aprimoram suas implementações da Informatica. Aproveitando qualquer uma das centenas de soluções fornecidas por desenvolvedores e parceiros da Informatica, você pode melhorar sua produtividade e agilizar o tempo de implementação nos seus projetos. Você pode acessar o Informatica Marketplace através do link https://marketplace.informatica.com.

Suporte global a clientes Informatica

Você pode entrar em contato com um Centro de Suporte Global por telefone ou via Suporte Online na Rede da Informatica.

Para descobrir o número de telefone local do Suporte Global a Clientes da Informatica, visite o site da Informatica no seguinte link:
http://www.informatica.com/us/services-and-training/support-services/global-support-centers.

Se você for membro da Rede da Informatica, poderá usar o Suporte Online em http://network.informatica.com.

CAPÍTULO 1

A Linguagem de Transformação

Este capítulo inclui os seguintes tópicos:

- Visão Geral da Linguagem de Transformação, 12
- Sintaxe de Expressão, 13
- Adicionando comentário a expressões, 15
- Palavras reservadas, 16

Visão Geral da Linguagem de Transformação

O Informatica Developer fornece uma linguagem de transformação que inclui funções como SQL para transformar dados de origem. Use essas funções para gravar expressões.

As expressões modificam dados ou testam se os dados correspondem às condições. Por exemplo, você pode usar a função AVG para calcular o salário médio de todos os funcionários, ou a função SUM para calcular as vendas totais de uma filial específica.

Você pode criar uma expressão simples que contém apenas uma porta, como ORDERS, ou um literal numérico, como 10. Além disso, você pode gravar expressões complexas que incluam funções aninhadas dentro de funções ou combinar portas diferentes usando os operadores de linguagem de transformação.

Componentes da Linguagem de Transformação

A linguagem de transformação inclui os seguintes componentes para criar expressões de transformação simples ou compostas:

- **Funções.** Mais de 100 funções como SQL permitem que você altere os dados em um mapeamento.
- **Operadores.** Use os operadores de transformação para criar expressões de transformação para executar computações matemáticas, combinar dados ou comparar dados.
- **Constantes.** Use constantes internas para consultar valores que permanecem constantes, como TRUE.
- **Parâmetros de mapeamento.** Crie parâmetros para usar em um mapeamento ou mapplet e consultar valores que permaneçam constantes durante a execução do mapeamento ou mapplet, como a taxa de imposto sobre vendas de imóveis.
- **Variáveis locais e internas.** Use variáveis internas para gravar expressões que consultam valores que variam, como a data do sistema. Você também pode criar variáveis locais em transformações.
- **Valores de retorno.** Você também pode gravar expressões que incluam transformações de pesquisa dos valores retornados.

Internacionalização e a Linguagem de Transformação

As funções de linguagem de transformação podem tratar dados de caractere no modo de movimento de dados ASCII ou Unicode. Use o modo Unicode para tratar dados de caractere *multibyte*. Os valores retornados das seguintes funções e transformações dependem da página de código do Data Integration Service e do modo de movimento de dados:

- INITCAP
- LOWER
- UPPER
- MIN (Data)
- MIN (Número)
- MIN (String)
- MAX (Data)
- MAX (Número)
- MAX (String)
- Qualquer função que usa instruções condicionais para comparar strings, como IIF e DECODE

MIN e MAX também retornam valores com base na ordem de classificação associada à página de código do Data Integration Service.

Ao validar uma expressão inválida no Editor de Expressão, uma caixa de diálogo exibe a expressão com um indicador de erro, ">>>>". Esse indicador aparece à esquerda de e aponta para a parte da expressão que contém o erro. Por exemplo, se a expressão a = b + c contiver um erro em c, a mensagem de erro exibirá:

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

As funções de linguagem de transformação que avaliam dados de caractere são controladas por caractere, não por bytes. Por exemplo, a função LENGTH retorna o número de caracteres em uma string, não o número de bytes. A função LOWER retorna uma string em minúsculas com base na página de código do Data Integration Service.

Sintaxe de Expressão

Embora a linguagem de transformação seja baseada no SQL padrão, há diferenças entre as duas linguagens. Por exemplo, SQL oferece suporte às palavras-chaves ALL e DISTINCT para funções de agregação, mas a linguagem de transformação não. Por outro lado, a linguagem de transformação oferece suporte a uma condição de filtro opcional para funções de agregação, mas o SQL não.

Você pode criar uma expressão tão simples quanto uma porta (como ORDERS) ou um literal numérico (como 10). Além disso, pode gravar expressões complexas que incluam funções aninhadas dentro de funções ou combinar colunas diferentes usando os operadores de linguagem de transformação.

Componentes de Expressão

As expressões podem consistir em qualquer combinação dos seguintes componentes:

- Portas (entrada, entrada/saída, variável)
- Literais de string, literais numéricos
- Constantes

- Funções
- Variáveis locais e internas
- Parâmetros de mapeamento
- Operadores
- Valores retornados

Portas e valores retornados

Quando você grava uma expressão que inclui uma porta ou um valor retornado de uma transformação não conectada, use os qualificadores de referência na seguinte tabela:

Qualificador de Referência	Descrição
:LKP	Requerido quando você cria uma expressão que inclui o valor retornado de uma transformação de Pesquisa não conectada. A sintaxe geral é: `:LKP.lookup_transformation(argument1, argument2, ...)` Os argumentos são as portas locais usadas na condição de pesquisa. A ordem deve corresponder à ordem das portas na transformação. Os tipos de dados das portas locais devem corresponder ao tipo de dados das portas de Pesquisa usadas na condição de pesquisa.

Literais de String e Numéricos

Você pode incluir literais numéricos ou de string.

Certifique-se de colocar literais de string entre aspas simples. Por exemplo:

```
'Alice Davis'
```

Os literais de string diferenciam letras maiúsculas e minúsculas e podem conter qualquer caracteres, exceto aspas simples. Por exemplo, a seguinte string não é permitida:

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

Para retornar uma string que contém um aspas simples, use a função CHR:

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

Não use aspas simples com literais numéricos. Apenas digite o número que deseja incluir. Por exemplo:

```
.05
```

ou

```
$$Sales_Tax
```

Regras e Diretrizes para Sintaxe de Expressão

Use as seguintes regras e diretrizes ao gravar expressões:

- Você não pode incluir funções de agregação aninhadas e de nível único em uma transformação de Agregador.
- Se você precisar criar funções aninhadas e de nível único, crie transformações de Agregador separadas.
- Não é possível usar strings em expressões numéricas.

  Por exemplo, a expressão `1 + '1'` não é válida porque você só pode executar adição em tipos de dados numéricos. Não é possível adicionar um inteiro e uma string.

- Você não pode usar strings como parâmetros numéricos.

  Por exemplo, a expressão `SUBSTR(TEXT_VAL, '1', 10)` não é válida porque a função SUBSTR requer um valor inteiro, não uma string, como a posição inicial.
- Você não poderá misturar tipos de dados quando usar operadores de comparação.

  Por exemplo, a expressão `123.4 = '123.4'` não é válida porque ela compara um valor decimal com uma string.
- Você pode passar um valor de uma porta, número ou string literal, transformação de pesquisa ou os resultados da expressão.
- Use a guia de portas no Editor de Expressão para inserir um nome de porta em uma expressão. Se você renomear uma porta em uma transformação conectada, a ferramenta Desenvolvedor propagará a alteração de nome às expressões na transformação.
- Separe cada argumento em uma função com uma vírgula.
- Com exceção dos literais, a linguagem de transformação não faz distinção de maiúsculas e minúsculas.
- Com exceção dos literais, a ferramenta Desenvolvedor e o Data Integration Service ignoram espaços.
- Os dois-pontos (:), a vírgula (,) e o ponto final (.) têm significado especial e devem ser usados apenas para especificar a sintaxe.
- O Data Integration Service trata traços (-) como operadores de subtração.
- Se você passar um valor literal para uma função, coloque as strings literais entre aspas simples. Não use aspas para números literais. O Data Integration Service trata qualquer valor de string entre aspas simples como string de caracteres.
- Quando você passa um parâmetro de mapeamento para uma função em uma expressão, não use aspas para designá-lo.
- Não use aspas para designar portas.
- Você pode aninhar várias funções dentro de uma expressão, exceto funções de agregação, que permitem apenas uma função de agregação aninhada. O Data Integration Service avalia a expressão a partir da função mais interna.

Adicionando comentário a expressões

A linguagem de transformação fornece dois especificadores de comentário para permitir que você insira comentários em expressões:

- Dois traços, como em:

  ```
  -- These are comments
  ```
- Duas barras, como em:

  ```
  // These are comments
  ```

O Data Integration Service ignora o texto em linhas precedidas por esses dois especificadores de comentário. Por exemplo, se quiser concatenar duas strings, você poderá inserir a seguinte expressão com comentários no meio da expressão:

```
 -- 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
```

O Data Integration Service ignora os comentário e avalia a expressão desta forma:

```
FIRST_NAME || LAST_NAME
```

Você não pode continuar um comentário em uma nova linha:

```
-- 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
```

Nesse caso, a ferramenta Desenvolvedor não valida a expressão, pois a última linha não é uma expressão válida.

Se não quiser incorporar comentários, você poderá adicioná-los clicando em Comentário no Editor de Expressão.

Palavras reservadas

Algumas palavras-chave na linguagem de transformação, como constantes, operadores e variáveis internas, são reservadas para funções específicas. Elas incluem:

- :INFA
- :LKP
- :MCR
- :TYPE
- AND
- DD_DELETE
- DD_INSERT
- DD_REJECT
- DD_UPDATE
- FALSE
- NOT
- NULL
- OR
- PROC_RESULT
- SPOUTPUT
- SYSDATE
- TRUE

Nota: Não é possível usar uma palavra reservada para denominar uma porta ou variável local. Você só pode usar palavras reservadas em expressões de transformação. Palavras reservadas têm significados predefinidos em expressões.

CAPÍTULO 2

Constantes

Este capítulo inclui os seguintes tópicos:

- DD_DELETE, 17
- DD_INSERT, 17
- DD_REJECT, 18
- DD_UPDATE, 19
- FALSE, 19
- NULL, 20
- TRUE, 21

DD_DELETE

Sinaliza registros para exclusão em uma expressão de estratégia de atualização. DD_DELETE é equivalente ao literal inteiro 2.

Nota: Use a constante DD_DELETE somente na transformação da Estratégia de atualização. Use DD_DELETE em vez do literal inteiro 2 para facilitar expressões numéricas complexas de solução de problemas.

Exemplo

A seguinte expressão marca os itens com um número de ID 1001 para exclusão, e todos os itens para inserção:

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

Esta expressão de estratégia de atualização usa literais numéricos para produzir o mesmo resultado:

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

Nota: A expressão que usa constantes é mais fácil de ler do que a expressão que usa literais numéricos.

DD_INSERT

Sinaliza registros para inserção em uma expressão de estratégia de atualização. DD_INSERT é equivalente ao literal inteiro 0.

Nota: Use a constante DD_INSERT somente na transformação da Estratégia de Atualização. Use DD_INSERT em vez do literal inteiro 0 para facilitar expressões numéricas complexas de solução de problemas.

Exemplos

O exemplo a seguir modifica um mapeamento que calcula as vendas mensais por vendedor e, dessa forma, você pode examinar as vendas de apenas um vendedor.

A seguinte expressão de estratégia de atualização sinaliza as vendas de um funcionário para inserção e rejeita todas as outras informações:

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

Esta expressão de estratégia de atualização usa literais numéricos para produzir o mesmo resultado:

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

Sugestão: A expressão que usa constantes é mais fácil de ler do que a expressão que usa literais numéricos.

DD_REJECT

Sinaliza registros para rejeição em uma expressão de estratégia de atualização. DD_REJECT é equivalente ao literal inteiro 3.

Nota: Use a constante DD_REJECT somente na transformação da Estratégia de Atualização. Use DD_REJECT em vez do literal inteiro 3 para facilitar expressões numéricas complexas de solução de problemas.

Use DD_REJECT para filtrar ou validar dados. Se você sinalizar um registro como rejeitado, o Data Integration Service irá ignorá-lo e gravá-lo no arquivo rejeitado da sessão.

Exemplos

Os exemplos a seguir modificam um mapeamento que calcula as vendas no mês atual, assim ele inclui apenas valores positivos.

Essa expressão de estratégia de atualização sinaliza registros menos de 0 para rejeitar e todos os outros para inserir:

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

Essa expressão usa literais numéricos para produzir o mesmo resultado:

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

A expressão que usa constantes é mais fácil de ler do que a expressão que usa literais numéricos.

O exemplo a seguir controlado por dados usa DD_REJECT e IS_SPACES para evitar a gravação de espaços em uma coluna de caracteres em uma tabela de destino. Essa expressão sinaliza registros que consistem totalmente em espaços para rejeitar e sinaliza todos os outros para inserir:

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

DD_UPDATE

Sinaliza registros para atualização em uma expressão de estratégia de atualização. DD_UPDATE é equivalente ao literal inteiro 1.

Nota: Use a constante DD_UPDATE somente na transformação da Estratégia de Atualização. Use DD_UPDATE em vez do literal inteiro 1 para facilitar expressões numéricas complexas de solução de problemas.

Exemplos

Os exemplos a seguir modificam um mapeamento que calcula as vendas do mês atual. O mapeamento carrega as vendas de um funcionário.

Essa expressão sinaliza registros para Alex como atualizações e sinaliza todos os outros para rejeição:

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

Essa expressão usa literais numéricos para produzir o mesmo resultado, sinalizando as vendas de Alex para atualização (1) e sinalizando todos os outros registros de venda para rejeição (3):

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

A expressão que usa constantes é mais fácil de ler do que a expressão que usa literais numéricos.

A expressão de estratégia de atualização a seguir usa SYSDATE para encontrar apenas os pedidos que foram entregues nos dois últimos dias e sinalizá-los para inserção. Usando DATE_DIFF, a expressão subtrai DATE_SHIPPED da data do sistema, retornando a diferença entre as duas datas. Como DATE_DIFF retorna um valor duplo, a expressão usa TRUNC para truncar a diferença. Então, ela compara o resultado ao literal inteiro 2. Se o resultado for maior que 2, a expressão sinalizará os registros para rejeição. Se o resultado for 2 ou menor, ela sinalizará os resultados para atualização. Caso contrário, ele os sinaliza para rejeição:

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

FALSE

Esclarece uma expressão condicional. FALSE é equivalente ao inteiro 0.

Exemplo

O exemplo a seguir usa FALSE em uma expressão DECODE para retornar valores baseados nos resultados de uma comparação. Isso será útil se você quiser realizar várias pesquisas baseadas em um único valor de pesquisa:

```
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 um valor é desconhecido ou indefinido. NULL não é equivalente a um espaço em branco ou string vazia (em colunas de caractere) ou 0 (em colunas numéricas).

Embora você possa gravar expressões que retornam nulos, qualquer coluna que tenha a restrição NOT NULL ou PRIMARY KEY não aceitará nulos. Por isso, se o Data Integration Servicetentar gravar um valor nulo em uma coluna com uma dessas restrições, o banco de dados rejeitará a linha e o Data Integration Service irá gravá-lo no arquivo rejeitado. Certifique-se de considerar nulos ao criar transformações.

As funções podem tratar os nulos de forma diferente. Se você passar um valor nulo para uma função, ela poderá retornar 0 ou NULL, ou poderá ignorar valores nulos.

Tópicos Relacionados:

- "Funções" na página 49

Trabalhando com valores Null em expressões booleanas

As expressões que combinam um valor nulo com uma expressão booleana produzem resultados compatíveis com ANSI. Por exemplo, o Data Integration Service produz estes resultados:

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

Valores nulos em expressões de comparação

Quando você usa um valor nulo em uma expressão que contenha um operador de comparação, o Data Integration Service produz um valor nulo. Para verificar os valores nulos nas colunas, você deve usar ISNULL() em expressões de comparação.

Para retornar linhas que não contenham valores nulos, use a função ISNULL, em vez da constante `!=`. Por exemplo, use `NOT ISNULL(Field_A)`.

A seguinte expressão resulta em um valor nulo e a transformação de filtro não retorna nenhuma linha: `Field_A!=NULL`.

Você também pode configurar a transformação de pesquisa para tratar valores nulos como altos ou baixos em operações de comparação. Use a propriedade Ordenamento Nulo na origem da pesquisa para configurar como o Data Integration Service trata valores nulos em expressões de comparação na transformação de pesquisa.

Valores Null em funções de agregação

O Data Integration Service trata valores nulos como nulos em funções de agregação. Se você passar uma porta ou um grupo completo de valores nulos, a função retornará NULL.

Valores Null em condições de filtro

Se uma condição de filtro for avaliada como NULL, a função não selecionará o registro. Se a condição de filtro for avaliada como NULL para todos os registros na porta selecionada, a função de agregação retornará NULL (exceto COUNT, que retornará 0). Você pode usar condições de filtro com funções de agregação e as funções CUME, MOVINGAVG e MOVINGSUM.

Nulos com Operadores

Qualquer expressão que use operadores (exceto o operador de string ||) e contenha um valor nulo é sempre avaliado como NULL. Por exemplo, a seguinte expressão é avaliada como NULL:

```
8 * 10 - NULL
```

Para testar nulos, use a função ISNULL.

TRUE

Retorna um valor baseado no resultado de uma comparação. TRUE é equivalente ao inteiro 1.

Exemplo

O exemplo a seguir usa TRUE em uma expressão DECODE para retornar valores baseados nos resultados de uma comparação. Isso será útil se você quiser realizar várias pesquisas baseadas em um único valor de pesquisa:

```
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 inclui os seguintes tópicos:

- Precedência de operador, 22
- Operadores complexos, 23
- Operadores aritméticos, 31
- Operadores de string, 32
- Operadores de comparação, 33
- Operadores lógicos, 34

Precedência de operador

A linguagem de transformação oferece suporte ao uso de vários operadores e ao uso de operadores dentro de expressões aninhadas.

Se você gravar uma expressão que inclui vários operadores, o Data Integration Service avaliará a expressão na seguinte ordem:

1. Operadores complexos
2. Operadores aritméticos
3. Operadores de string
4. Operadores de comparação
5. Operadores lógicos

O Data Integration Service avalia os operadores na ordem em que aparecem na tabela a seguir. Ele avalia os operadores em uma expressão com precedência igual a todos os operadores da esquerda para a direita.

A seguinte tabela lista a precedência de todos os operadores de linguagem de transformação:

Operador	Significado
`[ ], .`	Subscrito, ponto.
( )	Parênteses.
+, -, NOT	Adição e subtração unários e o operador lógico NOT.
*, /, %	Multiplicação, divisão, módulo.

Operador	Significado
+, -	Adição, subtração.
||	Concatenar.
<, <=, >, >=	Menor, menor ou igual, maior, maior ou igual.
=, <>, !=, ^=	Igual, diferente, diferente, diferente.
AND	Operador lógico AND, usado ao especificar condições.
OR	Operador lógico OR, usado ao especificar condições.

A linguagem de transformação também oferece suporte ao uso de operadores dentro de expressões aninhadas. Quando as expressões contêm parênteses, o Data Integration Service avalia operações dentro de parênteses antes de outras operações. As operações nos parênteses mais internos são avaliadas primeiro.

Por exemplo, dependendo de como você aninha as operações, a equação 8 + 5 - 2 * 8 retorna valores diferentes:

Equação	Valor de Retorno
8 + 5 - 2 * 8	-3
8 + (5 - 2) * 8	32

Operadores complexos

Use operadores complexos para acessar elementos em um tipo de dados complexo. É possível acessar elementos em um tipo de dados de array ou struct.

Você pode usar operadores complexos em mapeamentos que são executados no mecanismo Spark.

A seguinte tabela lista os operadores complexos na linguagem de transformação:

Operador	Significado
[ ]	Operador de subscrito. Use um operador de subscrito para acessar um ou mais elementos em um array.
.	Operador de ponto. Use um operador de ponto para acessar um elemento em um struct. Você também pode usar um operador de ponto em um array de structs para acessar elementos em cada struct.

Quando você usa um operador de ponto em um array de structs, ele retorna os elementos do mesmo nome dentro de cada struct como um array. Para acessar elementos em um array ou struct aninhado, você pode usar uma combinação de operadores complexos.

Operador de subscrito

Use um operador de subscrito para acessar um ou mais elementos em um array. Você pode acessar um elemento específico ou um intervalo de elementos em um array.

Sintaxe

Para acessar um elemento específico em um array, use a seguinte sintaxe:

```
array[ index ]
```

Para acessar um intervalo de elementos em um array, use a seguinte sintaxe:

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

A seguinte tabela descreve os argumentos na sintaxe:

Argumento	Descrição
array	Array. O array do qual você deseja acessar um ou mais elementos. É possível inserir qualquer expressão de transformação válida que seja avaliada como um array.
index	Número inteiro. A posição do elemento que você deseja acessar. Por exemplo, um índice de 0 indica o primeiro elemento em um array.
start_index	Número inteiro. O índice inicial em um intervalo de elementos que você deseja acessar. O operador de subscrito inclui o elemento que o índice inicial representa.
end_index	Número inteiro. O índice final em um intervalo de elementos que você deseja acessar. O operador de subscrito exclui o elemento que o índice final representa.

Você pode usar uma expressão para o índice que retorna um valor de número inteiro . Se a expressão retornar um valor negativo, o índice será considerado 0.

Se o índice especificado for maior que o tamanho do array menos 1, o índice acessará o elemento final no array.

Valor de retorno

Se você especificar um índice, a expressão retornará o elemento no array. O tipo de retorno é o mesmo que o tipo de dados do elemento do array especificado.

Se você especificar dois índices separados por uma vírgula, como `[i,j]`, a expressão retornará um array dos elementos de `i` para `j-1`. Se `i` for maior que `j` ou o tamanho do array, a expressão retornará um array vazio. A configuração de tipo do subarray que a expressão retorna é a mesma que a configuração de tipo do array especificado.

Nulos

Se o índice no subscrito for maior que o tamanho do array, o operador de subscrito retornará um valor NULL.

Se o índice for NULL, o operador de subscrito retornará um valor NULL. Se você especificar vários índices, como `[i,j]` e ou `i` ou `j` for NULL, a expressão retornará NULL.

Se o arrr for NULL, o operador de subscrito retornará um valor NULL.

Exemplos

Você tem o seguinte array com elementos de string:

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

As expressões a seguir usam um operador de subscrito para acessar elementos de string do array:

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

Operador de ponto

Use um operador de ponto para acessar um elemento em um struct. Você também pode usar um operador de ponto em um array de structs para acessar elementos de cada struct no array.

Sintaxe

Para acessar um elemento em um struct, use a seguinte sintaxe:

```
struct.element
```

Para acessar um elemento em um array de structs, use a seguinte sintaxe:

```
array_of_structs.element
```

A seguinte tabela descreve os argumentos na sintaxe:

Argumento	Descrição
struct	Struct. O struct do qual você deseja acessar um elemento. É possível inserir qualquer expressão de transformação válida que seja avaliada como um struct.
array_of_structs	Array com elementos de Struct. O array do qual você deseja acessar elementos em cada struct. É possível inserir qualquer expressão de transformação válida que seja avaliada como um array.
element	O nome do elemento de struct que você deseja acessar.

Valor de retorno

Se você usar o operador de ponto em um struct, a expressão retornará o elemento nesse struct. O tipo de retorno é o mesmo que o tipo de dados do elemento do struct especificado.

Se você usar o operador de ponto em um array de structs, a expressão retornará um array que contém o elemento especificado em cada struct.

Nulos

Se o elemento no struct tiver um valor NULL, a expressão retornará NULL.

Se o struct for NULL, a expressão retornará NULL.

Exemplos

Você tem o seguinte struct:

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

As expressões a seguir usam um operador de ponto para acessar elementos no struct:

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

Você também pode usar um operador de ponto para acessar elementos em um array de structs.

Por exemplo, você tem o seguinte array com três elementos do tipo struct, e cada struct tem três 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
 }

]
```

As expressões a seguir usam um operador de ponto para acessar os elementos de string em cada struct do array:

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 complexos para tipos de dados aninhados

Um tipo de dados aninhado contém elementos de tipos de dados complexos. Use uma combinação de operadores complexos para acessar elementos em tipos de dados aninhados.

Quando um array ou struct contiver elementos do tipo array ou struct, use uma combinação de operadores complexos para acessar os elementos. É possível acessar elementos em arrays multidimensionais, arrays com elementos struct, structs com elementos array e structs com elementos struct.

Array multidimensional

Um array multidimensional é um array de arrays, que pode ter até cinco níveis de aninhamento. É possível usar operadores de subscrito para acessar arrays em qualquer nível ou elementos específicos em um array no nível mais interno.

Você pode usar operadores de subscrito para retornar os seguintes valores:

- Um elemento específico em um array no nível mais interno.
- Um ou mais arrays em qualquer nível.
- Um subconjunto de um ou mais arrays em qualquer nível.

Para acessar um elemento específico em um array no nível mais interno, você usa mais de um operador de subscrito. O número de dimensões em um array multidimensional determina o número de operadores de subscrito a serem usados. Cada operador de subscrito deve conter um valor de índice. O tipo de dados do valor de retorno é o mesmo que o dos elementos no array.

Por exemplo, em um array bidimensional, você usa dois operadores de subscrito. O primeiro operador de subscrito determina qual array unidimensional deve ser acessada. O segundo operador de subscrito determina qual elemento deve ser acessado dentro do array.

O seguinte array bidimensional contém três arrays e cada array contém elementos do tipo String:

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

As expressões a seguir usam dois operadores de subscrito para acessar um elemento específico de cada array unidimensional dentro de `menu_array`:

Input Value	RETURN VALUE
`menu_array[0][1]`	`'coffee'`
`menu_array[2][3]`	`'chipotle'`
`menu_array[1][2]`	`NULL`

As expressões a seguir usam um único operador de subscrito para retornar arrays unidimensionais em `menu_array`:

Input Value	RETURN VALUE
`menu_array[0]`	`['milk','coffee','tea','chai']`

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

As expressões a seguir usam dois operadores de subscrito para retornar um subconjunto de arrays dentro de `menu_array`:

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

Array com elementos de struct

Um array com elementos de struct é um array de structs. Use uma combinação de operadores de subscrito e pontos para acessar um elemento em um struct que esteja dentro de um array.

Para acessar um elemento em um struct dentro de um array, use um operador de subscrito seguido por um operador de ponto. Você também pode inverter a ordem dos operadores. Os valores de retorno são os mesmos, independentemente da ordem dos operadores. Com base na ordem dos operadores complexos, o elemento é acessado da seguinte maneira:

Você usa um operador de subscrito seguido por um operador de ponto.

O operador de subscrito primeiro acessa o elemento indexado no array e retorna um struct. Em seguida, o operador de ponto acessa um elemento dentro do struct.

Você usa um operador de ponto seguido por um operador de subscrito.

O operador de ponto localiza elementos com o mesmo nome em cada um dos structs e retorna um array. Em seguida, o operador de subscrito acessa um elemento dentro do array.

Por exemplo, você tem o seguinte array, `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
}
]
```

As expressões a seguir usam um operador de subscrito seguido por um operador de ponto no array `employee_info_array`:

Input Value	RETURN VALUE
`employee_info_array[0].name`	`'Derrick'`
`employee_info_array[1].city`	`'Redwood City'`
`employee_info_array[2].state`	`NULL`

Quando você usa um operador de ponto primeiro, este retorna um array com elementos do mesmo nome de cada struct. Por exemplo, as expressões a seguir mostram o valor de retorno quando você usa um operador de ponto:

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]`

Em seguida, o operador de subscrito acessa um elemento no array retornado. As expressões a seguir usam um operador de ponto seguido por um operador de subscrito:

Input Value	RETURN VALUE
`employee_info_array.name[0]`	`'Derrick'`
`employee_info_array.city[1]`	`'Redwood City'`
`employee_info_array.state[2]`	`NULL`

Observe que os valores de retorno são os mesmos, independentemente de você usar um operador de subscrito ou um operador de ponto primeiro. Por exemplo, as expressões `employee_info_array[0].name` e `employee_info_array.name[0]` têm o mesmo valor de retorno `'Derrick'`.

Struct com elementos de Array

Para acessar os elementos em um array que está dentro de um struct, use um operador de ponto seguido por um operador de subscrito. O operador de ponto primeiro acessa o elemento de array especificado em um struct. Em seguida, o operador de subscrito acessa os elementos no array com base no valor de índice.

Por exemplo, você tem o seguinte struct com os elementos de array `drinks, sandwiches` e `salads`.

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

Quando você usa a expressão `menu_struct.drinks[0]`, o operador de ponto primeiro acessa o elemento de array `drinks`. Em seguida, o operador de subscrito acessa o elemento na posição 0 do array `drinks`: `['milk','coffee','tea','chai']`. O valor de retorno é `milk`.

As expressões a seguir usam um operador de ponto seguido por um operador de subscrito para acessar elementos dos arrays no struct `menu_struct`:

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

Struct com elementos de Struct

Uma struct que contém um ou mais níveis de structs é um struct aninhado. Você pode usar operadores de ponto para acessar structs em qualquer nível ou elementos específicos em um struct no nível mais interno.

Você pode usar operadores de ponto para retornar os seguintes valores:

- Um elemento especificado em um struct no nível mais interno.
- Um ou mais structs em qualquer nível.

Para acessar um elemento específico em um struct no nível mais interno, você usa mais de um operador de ponto. O número de níveis em um struct aninhado determina o número de operadores de ponto a serem usados. O tipo de dados do valor de retorno é o mesmo que o tipo de dados do elemento no struct. Por exemplo, em um struct aninhado de dois níveis, você usa dois operadores de ponto. O primeiro operador de ponto acessa o elemento de struct filho especificado em um struct pai. Em seguida, o segundo operador de ponto acessa elementos no struct filho.

O exemplo a seguir usa um struct `employee_info_struct` que contém dois structs filho `home_address_info` e `department_info`:

```
employee_info_struct{

emp_name: 'Derrick'

home_address_info{
    city: 'New York'
    state: NULL

department_info{
    NULL
}

}
```

As expressões a seguir usam operadores de ponto para acessar elementos do struct `employee_info_struct`:

Input Value	RETURN VALUE
`employee_info_struct.emp_name`	`'Derrick'`

Input Value	RETURN VALUE

employee_info_struct.home_address_info	{
	city: 'New York'
	state: NULL
	}

employee_info_struct.department_info	NULL
employee_info_struct.home_address_info.city	'New York'
employee_info_struct.home_address_info.state	NULL

Operadores aritméticos

Use operadores aritméticos para realizar cálculos matemáticos em dados numéricos.

A seguinte tabela lista os operadores aritméticos na ordem de precedência na linguagem de transformação:

Operador	Significado
+, -	Adição e subtração unárias. Adição unária indica um valor positivo. Subtração unária indica um valor negativo.
*, /, %	Multiplicação, divisão, módulo. Um módulo é o resto depois de dividir dois inteiros. Por exemplo, 13 % 2 = 1 porque 13 dividido por 2 é igual a 6 com um resto igual a 1.
+, -	Adição, subtração. O operador de adição (+) não concatena strings. Para concatenar strings, use o operador de string ||. Para realizar uma operação aritmética em valores de data, use as funções de data.

Se você realizar uma operação aritmética em um valor nulo, a função retornará NULL.

Quando você usa operadores aritméticos em uma expressão, todos os operandos na expressão devem ser numéricos. Por exemplo, a expressão `1+'1'` não é válida porque ela adiciona um inteiro a uma string. A expressão `1,23+4/2` é válida porque todos os operandos são numéricos.

Nota: A linguagem de transformação fornece funções de data internas que permitem realizar uma operação aritmética em valores de data/hora.

Tópicos Relacionados:

- "Noções básicas dos cálculos aritméticos de data" na página 48

Operadores de string

Use o operador de string || para concatenar duas strings. O operador || converte operandos de qualquer tipo de dados (exceto Binários) em tipos de dados de string antes da concatenação:

Valor de entrada	Valor de Retorno
'alpha' || 'betical'	alfabético
'alpha' || 2	alpha2
'alpha' || NULL	alpha

O operador || inclui espaços em branco à direita e à esquerda. Use as funções LTRIM e RTRIM para excluir espaços em branco à direita e à esquerda antes de concatenar duas strings.

Nulls

O operador || ignora valores nulos. No entanto, se os dois valores forem NULL, o operador || retornará NULL.

Exemplo

O exemplo a seguir mostra uma expressão que concatena os nomes e os sobrenomes dos funcionários em duas colunas. Essa expressão remove os espaços do final do nome e do início do sobrenome, concatena um espaço para o final de cada nome e, em seguida, concatena o sobrenome:

```
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: Você também pode usar a função CONCAT para concatenar dois valores da string. No entanto, o operador || produz os mesmos resultados em menos tempo.

Operadores de comparação

Use operadores de comparação para comparar strings de caracteres ou numéricas, manipular dados e retornar um valor TRUE (1) ou FALSE (0).

A seguinte tabela lista os operadores de comparação na linguagem de transformação:

Operador	Significado
=	Igual a.
>	Maior que.
<	Menor que.
>=	Maior que ou igual a.
<=	Menor que ou igual a.
<>	Diferente de.
!=	Diferente de.
^=	Diferente de.

Use os operadores maior que (>) e menor que (<) para comparar valores numéricos ou retorne uma variedade de linhas baseada na ordem de classificação de uma chave primária em uma porta específica.

Ao usar operadores de comparação em uma expressão, os operandos devem ter o mesmo tipo de dados. Por exemplo, a expressão 123.4 > '123' não é válida porque a expressão compara um decimal com uma string. As expressões 123.4 > 123 e 'a' != 'b' são válidas porque os operandos têm o mesmo tipo de dados.

Se você comparar um valor a um valor nulo, o resultado será NULL.

Se uma condição de filtro for avaliada como NULL, o Serviço de Integração retornará NULL.

Comparação de tipos de dados complexos

Você pode usar os operadores igual a (=) e diferente de (!=) para comparar tipos de dados complexos, como matrizes ou estruturas.

Para que duas matrizes sejam equivalentes, as seguintes condições devem ser aplicadas:

- Os elementos de matriz devem ser do mesmo tipo de dados.
- As matrizes devem ter o mesmo tamanho.
- A entrada em cada índice deve ser a mesma.

Por exemplo, você tem as seguintes matrizes:

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

Você pode fazer a seguinte comparação:

```
A = B
```

RETURN VALUE: TRUE (1)
Ambas as matrizes são do mesmo tamanho e a entrada em cada um dos índices é a mesma de modo que `A[0]=B[0], A[1]=B[1] e A[2]=B[2].`

Quando você comparar duas estruturas, as estruturas serão equivalentes se atenderem as seguintes condições:

- Os elementos de estrutura correspondentes devem ser do mesmo tipo de dados.
- As estruturas devem ter os mesmos dados.

Se essas condições forem atendidas, as duas estruturas serão equivalentes, mesmo se os elementos de estruturas tiverem nomes diferentes.

Por exemplo, você tem as seguintes estruturas:

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

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

Você pode fazer a seguinte comparação:

```
struct1 = struct2
```

RETURN VALUE: TRUE (1)

Operadores lógicos

Use operadores lógicos para manipular dados numéricos. As expressões que retornam um valor numérico são avaliadas como TRUE para valores diferentes de 0, FALSE para 0 e NULL para NULL.

A seguinte tabela lista os operadores lógicos na linguagem de transformação:

Operador	Significado
NOT	Nega o resultado de uma expressão. Por exemplo, se uma expressão for avaliada como TRUE, o operador NOT retornará FALSE. Se uma expressão for avaliada como FALSE, NOT retornará TRUE.
AND	Reúne duas condições e retorna TRUE quando ambas as condições são TRUE. Retorna FALSE quando uma condição não é verdadeira.
OR	Conecta duas condições e retorna TRUE quando alguma condição é avaliada como TRUE. Retorna FALSE quando as duas condições não são verdadeiras.

Nulls

As expressões que combinam um valor nulo com uma expressão booleana produzem resultados compatíveis com ANSI. Por exemplo, o Data Integration Service produz estes resultados:

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

CAPÍTULO 4

Variáveis

Este capítulo inclui os seguintes tópicos:

- Variáveis internas , 35
- Variáveis Locais, 35

Variáveis internas

A linguagem da transformação fornece a variável interna SYSDATE que retorna a data do sistema. Você pode usar SYSDATE em uma expressão. Por exemplo, você pode usar SYSDATE em uma função DATE_DIFF.

SYSDATE

SYSDATE retorna a data e hora atuais em segundos no nó que processa dados de cada linha que passa pela transformação. A variável SYSDATE é armazenada como um valor de tipo de dados de data/hora da transformação.

Exemplo

A expressão a seguir usa SYSDATE para encontrar os pedidos que foram entregues nos dois últimos dias e sinalizá-los para inserção. Usando DATE_DIFF, o Data Integration Service subtrai DATE_SHIPPED da data do sistema, retornando a diferença entre as duas datas. Como DATE_DIFF retorna um valor duplo, a expressão trunca a diferença. Então, ela compara o resultado ao literal inteiro 2. Se o resultado for maior que 2, a expressão sinalizará as linhas para rejeição. Se o resultado for 2 ou menor, ela os sinalizará para inserção.

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

Variáveis Locais

Se você usar variáveis locais em um mapeamento, use-as em qualquer expressão de transformação no mapeamento. Por exemplo, se você usar um cálculo de imposto complexo em um mapeamento, poderá gravar a expressão uma vez e designá-la como uma variável. Isso aumenta o desempenho, já que o Data Integration Service realiza o cálculo apenas uma vez.

As variáveis locais são úteis quando usadas com expressões de procedimento armazenadas para capturar vários valores retornados.

CAPÍTULO 5

Datas

Este capítulo inclui os seguintes tópicos:

- Visão geral das datas, 36
- Strings de Formato de Data, 40
- Strings de formato TO_CHAR, 41
- Strings de formato TO_DATE e IS_DATE, 44
- Noções básicas dos cálculos aritméticos de data, 48

Visão geral das datas

A linguagem de transformação fornece um conjunto de funções de data e variáveis internas de data para executar transformações em datas. Com as funções de data, você pode arredondar, truncar ou comparar datas, extrair uma parte de uma data ou realizar operação aritmética em uma data. Você pode passar qualquer valor com um tipo de dados de data para uma função de data.

Use variáveis de data para capturar a data atual no nó que hospeda o Data Integration Service.

A linguagem de transformação também fornece os seguintes conjuntos de strings de formato:

- **Strings de formato de data.** Use com funções de data para especificar as partes de uma data.
- **Strings de formato TO_CHAR.** Use para especificar o formato da string retornada.
- **Strings de formato TO_DATE e IS_DATE.** Use para especificar o formato de uma string que você deseja converter em uma data ou em um teste.

Tipo de Dados de Data/Hora

A Informatica usa tipos de dados genéricos para transformar dados de origens diferentes. Esses tipos de dados de transformação incluem um tipo de dados Data/Hora que oferece suporte a valores de data e hora até o nanossegundo. A Informatica armazena datas internamente em formato binário.

As funções de data aceitam valores de data e hora apenas. Para passar uma string para uma função de data, primeiro use TO_DATE para convertê-la em um valor de data e hora. Por exemplo, a seguinte expressão converte uma porta de string em valores de data e hora e, em seguida, adiciona um mês à cada data:

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

Você pode usar datas entre 1 d.C. e 9999 d.C. no sistema do calendário gregoriano.

Dia juliano, dia juliano modificado e calendário gregoriano

Só é possível usar datas no sistema do calendário gregoriano. As datas no calendário juliano são chamadas *datas* julianas e não têm suporte na Informatica. Esse termo não deve ser confundido com *dia* juliano ou dia juliano modificado.

Você pode manipular formatos MJD (dia juliano modificado) usando a string de formato J. O MJD de uma determinada data é o número de dias para essa data desde 1º de janeiro de 4713 A.C. 00:00:00 (meia-noite). Por definição, MJD inclui um componente de hora expresso como um decimal, que representa alguma fração de 24 horas. A string de formato J não converte esse componente de hora.

Por exemplo, a seguinte expressão TO_DATE converte strings na porta SHIP_DATE_MJD_STRING em valores de data no formato de data padrão:

```
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

Como a string de formato J não inclui a parte de tempo de uma data, os valores retornados têm a hora definida como 00:00:00.000000000.

Você também pode usar a string de formato J em expressões TO_CHAR. Por exemplo, use a string de formato J em uma expressão TO_CHART para converter valores de data em valores MJD expressos como strings. Por exemplo:

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

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

Nota: O Data Integration Service ignora a parte de hora da data em expressões TO_CHART.

Data no ano 2000

Todas as funções de data da linguagem de transformação oferecem suporte ao ano 2000. O Informatica Developer oferece suporte a datas entre 1 d.C. e 9999 d.C.

String de formato RR

A linguagem de transformação fornece a string de formato RR para converter strings com anos de dois dígitos em datas. Usando TO_DATE e a string de formato RR, você pode converter uma string no formato

MM/DD/RR em uma data. A string de formato RR converte dados de forma diferente dependendo do ano atual.

- **Ano atual entre 0 e 49.** Se o ano atual estiver entre 0 e 49 (como 2003) e o ano da string de origem estiver entre 0 e 49, o Data Integration Service retornará o século atual mais o ano de dois dígitos da string de origem. Se o ano da string de origem estiver entre 50 e 99, o Serviço de Integração retornará o século anterior mais o ano de dois dígitos da string de origem.
- **Ano atual entre 50 e 99.** Se o ano atual estiver entre 50 e 99 (como 1998) e o ano da string de origem estiver entre 0 e 49, o Data Integration Service retornará o século seguinte mais o ano de dois dígitos da string de origem. Se o ano da string de origem estiver entre 50 e 99, o Data Integration Service retornará o século atual mais o ano de dois dígitos especificado.

A seguinte tabela resume como a string de formato RR converte em datas:

Ano atual	Ano de origem	Retorno da string de formato RR
0 - 49	0 - 49	Século atual
0 - 49	50 - 99	Século anterior
50 - 99	0 - 49	Século seguinte
50 - 99	50 - 99	Século atual

Exemplo

A seguinte expressão produz os mesmos valores retornados para qualquer ano entre 1950 e 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

Diferença entre as strings de formato YY e RR

O Informatica Developer fornece ainda uma string de formato YY. Ambas as strings de formato RR e YY especificam anos de dois dígitos. As strings de formato YY e RR produzem resultados idênticos quando usadas com todas as funções de data, exceto TO_DATE. Em expressões TO_DATE, RR e YY produzem resultados diferentes.

A seguinte tabela mostra os diferentes resultados que cada string de formato retorna:

String	Ano atual	TO_DATE(String, 'MM/DD/RR')	TO_DATE(String, '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

Em datas no ano 2000 e seguintes, a string de formato YY produz resultados menos significativos do que a string de formato RR. Use a string de formato RR em datas no século vinte e um.

Datas em bancos de dados relacionais

Em geral, as datas armazenadas em bancos de dados relacionais contêm um valor de data e hora. A data inclui o mês, o dia e o ano, e a hora inclui as horas, os minutos, os segundos e os sub-segundos. Você pode passar os dados de data e hora a qualquer uma das funções de data.

Datas em arquivos planos

Use a função TO_DATE para converter strings em valores de data e hora. Você também pode usar IS_DATE para verificar se uma string é uma data válida antes de convertê-la com TO_DATE. As funções de data de linguagem de transformação aceitam apenas valores de data. Para passar uma string para uma função de data, você deverá primeiro usar a função TO_DATE para convertê-la em um tipo de dados de Data/Hora de transformação.

Formato de data padrão

O Data Integration Service usa um formato de data padrão para armazenar e manipular strings que representem datas. Para especificar o formato de data padrão, insira um formato de data no atributo String de Formato DateTime na configuração do visualizador de dados. Por padrão, o formato de data é MM/DD/AAAA HH24:MI:SS.US.

Como a Informatica armazena datas no formato binário, o Data Integration Service usa o formato de data padrão quando você executa as seguintes ações:

- **Converter uma data em string conectando uma porta de data/hora a uma porta de string.** O Data Integration Service converte a data em uma string no formato de data definido na configuração do visualizador de dados.
- **Converter uma string em data conectando uma porta de string a uma porta de data/hora.** O Data Integration Service espera que os valores de string estejam no formato de data definido pela configuração do visualizador de dados. Se um valor de entrada não corresponder a esse formato ou for uma data inválida, o Data Integration Service ignorará a linha. Se a string estiver nesse formato, o Data Integration Service converterá a string em um valor de data.
- **Usar TO_CHAR(data, [format_string]) para converter datas em strings.** Se você omitir a string de formato, o Data Integration Service retornará a string no formato de data definido na configuração do visualizador de dados. Se você especificar uma string de formato, o Data Integration Service retornará uma string no formato especificado.
- **Usar TO_DATE(data, [format_string]) para converter strings em datas.** Se você omitir a string de formato, o Data Integration Service esperará uma string no formato de data definido na configuração do visualizador de dados. Se você especificar uma string de formato, o Data Integration Service esperará uma string no formato especificado.

Por padrão, o formato de data MM/DD/AAAA HH24:MI:SS.US. consiste em:

- Mês (Janeiro = 01, setembro = 09)
- Dia (do mês)
- Ano (expresso em quatro dígitos, como 1998)
- Hora (no formato 24 horas, por exemplo, 12:00:00AM = 0, 1:00:00AM = 1, 12:00:00PM = 12, 11:00:00PM = 23)
- Minutos

- Segundos
- Microssegundos

Strings de Formato de Data

Você pode avaliar as datas de entrada que usam uma combinação de strings de formato e funções de data. As strings de formato de data não são internacionalizadas e devem ser inseridas em formatos predefinidos conforme listado na tabela a seguir.

A seguinte tabela resume as strings de formato para especificar uma parte de uma data:

String de Formato	Descrição
D, DD, DDD, DAY, DY, J	Dias (01 - 31). Use qualquer uma dessas strings de formato para especificar a parte do dia completa de uma data. Por exemplo, se você passar 12-APR-1997 para uma função de data, o uso de qualquer uma dessas strings de formato especificará 12.
HH, HH12, HH24	Hora do dia (0 - 23), onde 0 é 12 AM (meia-noite). Use qualquer um desses formatos para especificar a parte da hora completa de uma data. Por exemplo, se você passar a data 12-APR-1997 2:01:32 PM, use HH, HH12 ou HH24 para especificar a parte de hora da data.
MI	Minutos (0 - 59).
MM, MON, MONTH	Mês (01 - 12). Use qualquer uma dessas strings de formato para especificar a parte do mês completa de uma data. Por exemplo, se você passar 12-APR-1997 para uma função de data, use MM, MON ou MONTH para especificar APR.
MS	Milissegundos (0 - 999).
NS	Nanossegundos (0 - 999999999).
SS, SSSS	Segundos (0 - 59).
US	Microssegundos (0 - 999999).
Y, YY, YYY, YYYY, RR	Parte do ano da data (0001 a 9999). Use qualquer uma dessas strings de formato para especificar a parte do ano completa de uma data. Por exemplo, se você passar 12-APR-1997 para uma função de data, use Y, YY, YYY ou YYYY para especificar 1997.

Nota: A string de formato não faz distinção entre maiúsculas e minúsculas. Ela deve estar sempre entre aspas simples.

A seguinte tabela descreve as funções de data que usam strings de formato para avaliar datas de entrada:

Função	Descrição
ADD_TO_DATE	A parte da data que você deseja alterar.
DATE_DIFF	A parte da data para calcular a diferença entre duas datas.

Função	Descrição
GET_DATE_PART	A parte da data que você deseja retornar. Essa função retorna um valor inteiro baseado no formato de data padrão.
IS_DATE	A data que você deseja verificar.
ROUND	A parte da data que você deseja arredondar.
SET_DATE_PART	A parte da data que você deseja alterar.
SYSTIMESTAMP	A precisão do carimbo de data e hora.
TO_CHAR (Datas)	A string de caracteres.
TO_DATE	A string de caracteres.
TRUNC (Datas)	A parte da data que você deseja truncar.

Strings de formato TO_CHAR

A função TO_CHAR converte um tipo de dados Data/Hora em uma string com o formato que você especificar. Você pode converter a data completa ou uma parte da data em uma string. Você pode usar TO-CHART para converter datas em strings, alterando o formato para fins de geração de relatório.

TO_CHAR é geralmente usado quando o destino é um arquivo plano ou um banco de dados que não oferece suporte a um tipo de dados Data/Hora.

A seguinte tabela resume as strings de formato para datas na função TO_CHAR:

String de Formato	Descrição
AM, A.M. PM, P.M.	Indicador de meridiano. Use qualquer um destas strings de formato para especificar horas AM e PM. AM e PM retornam os mesmos valores que A.M. e P.M.
D	Dia da semana (1 - 7), em que domingo é igual a 1.
DAY	Nome do dia com até nove caracteres (por exemplo, Quarta).
DD	Dia do mês (01 - 31).
DDD	Dia do ano (001 - 366, incluindo anos bissextos).
DY	Nome abreviado com três caracteres para um dia (por exemplo, Qua).
HH, HH12	Hora do dia (01 - 12).
HH24	Hora do dia (00 - 23), em que 00 é 12AM (meia-noite).

String de Formato	Descrição
J	Dia juliano modificado. Converte a data do calendário em uma string equivalente ao seu valor como dia juliano modificado, calculado a partir de 1° de janeiro de 4713 00:00:00 a.C., que ignora o componente de hora da data. Por exemplo, a expressão TO_CHAR(SHIP_DATE, 'J') converte 31 de dezembro de 1999 23:59:59 na string 2451544.
MI	Minutos (00 - 59).
MM	Mês (01 - 12).
MONTH	Nome do mês com até nove caracteres (por exemplo, janeiro).
MON	Nome abreviado com três caracteres para um mês (por exemplo, jan).
MS	Milissegundos (0 - 999).
NS	Nanossegundos (0 - 999999999).
Trim.	Trimestre do ano (1 - 4), em que janeiro a março é igual a 1.
RR	Os dois últimos dígitos de um ano. A função remove os dígitos à esquerda. Por exemplo, se você usar 'RR' e passar o ano 1997, TO_CHAR retornará 97. Quando usado com TO_CHAR, 'RR' produz os mesmos resultados que 'YY' e pode ser trocado por 'YY'. No entanto, quando usado com TO_DATE, 'RR' calcula o século apropriado mais próximo e fornece os dois primeiros dígitos do ano.
SS	Segundos (00 - 59).
SSSSS	Segundos desde a meia-noite (00000 - 86399). Quando você usa SSSSS em uma expressão TO_CHAR, o Data Integration Service avalia apenas a hora de uma data. Por exemplo, a expressão TO_CHAR(SHIP_DATE, 'MM/DD/YYYY SSSSS') converte 12/31/1999 01:02:03 em 12/31/1999 03723.
US	Microssegundos (0 - 999999).
Ano	O último dígito de um ano. A função remove os dígitos à esquerda. Por exemplo, se você usar 'Y' e passar o ano 1997, TO_CHAR retornará 7.
YY	Os dois últimos dígitos de um ano. A função remove os dígitos à esquerda. Por exemplo, se você usar 'YY' e passar o ano 1997, TO_CHAR retornará 97.
YYY	Os três últimos dígitos de um ano. A função remove os dígitos à esquerda. Por exemplo, se você usar 'YYY' e passar o ano 1997, TO_CHAR retornará 997.
YYYY	Toda a parte do ano da data. Por exemplo, se você usar 'YYYY' e passar o ano 1997, TO_CHAR retornará 1997.
W	Semana do mês (1 - 5), em que a semana 1 começa no primeiro dia do mês e termina no sétimo, a semana 2 começa no oitavo dia e termina no décimo quarto dia. Por exemplo, Fev 1 designa a primeira semana de fevereiro.
WW	Semana do ano (01 - 53), em que a semana 01 começa em 1º de janeiro e termina em 7 de janeiro, a semana 2 começa em 8 de janeiro e termina em 14 de janeiro e assim por diante.
- / . ; :	Pontuação exibida na saída. Você pode usar esses símbolos para separar partes da data. Por exemplo, crie a seguinte expressão para separar partes da data com um ponto final: TO_CHAR(DATES, 'MM.DD.YYYY').

String de Formato	Descrição
“text”	Texto exibido na saída. Por exemplo, se você criar uma porta de saída com a expressão: TO_CHAR(DATES, ‘MM/DD/YYYY “Sales Were Up”’) e passar a data 1º de abril de 1997, a função retornará a string ‘04/01/1997 Sales Were Up’. Você pode inserir caracteres multibyte que sejam válidos na página de código de repositório.
“”	Use aspas duplas para separar strings de formato ambíguas, por exemplo D“”DDD. As aspas vazias não aparecem na saída.

Nota: A string de formato não faz distinção entre maiúsculas e minúsculas. Ela deve estar sempre entre aspas simples.

Exemplos

Os exemplos a seguir ilustram as strings de formato J, SSSSS, RR e YY. Consulte as funções individuais para obter mais exemplos.

Nota: O Data Integration Service ignora a hora da data em uma expressão TO_CHART.

String no formato J

Use a string de formato J em uma expressão TO_CHAR para converter valores de data em valores MJD expressos como strings. Por exemplo:

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

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

String de Formato SSSSS

Você também pode usar a string de formato SSSSS em uma expressão TO_CHAR. Por exemplo, a seguinte expressão converte as datas na porta SHIP_DATE em strings que representam o total de segundos desde a meia-noite:

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

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

String de formato RR

A seguinte expressão converte datas em strings no formato MM/DD/AAAA:

```
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

string de formato YY

Em expressões TO_CHAR, a string de formato YY produz os mesmos resultados que a string de formato RR. A seguinte expressão converte datas em strings no formato MM/DD/AAAA:

```
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

Strings de formato TO_DATE e IS_DATE

A função TO_DATE converte uma string com o formato especificado em um valor de data e hora. A função TO_DATE geralmente é usada para converter strings de arquivos planos em valores de data e hora. As cadeias de formato TO_DATE não são internacionalizadas e devem ser inseridas nos formatos predefinidos.

Nota: TO_DATE e IS_DATE usam o mesmo conjunto de strings de formato.

Quando criar uma expressão TO_DATE, use uma string de formato para cada parte da data na string de origem. O formato da cadeia de origem e a cadeia de formato devem corresponder. O separador de data não precisa corresponder para que a validação da data ocorra. Se alguma parte não corresponder, o Data Integration Service não converterá a cadeia e ignorará a linha. Se você omitir a string de formato, a string de origem deverá estar no formato de data especificado na configuração do visualizador de dados.

IS_DATE indica se um valor é uma data válida. Uma data válida é qualquer cadeia no formato de data especificado na configuração do visualizador de dados. Se as cadeias que você deseja testar não estiverem no formato de data especificado, use o formato das cadeias listado na tabela "Cadeias de Formato TO_DATE e IS_DATE". Se o formato de uma cadeia não corresponder ao formato especificado ou se a cadeia não representar uma data válida, a função retornará FALSE (0). Se o formato da cadeia corresponder ao formato especificado da cadeia e for uma data válida, a função retornará TRUE (1). As cadeias de formato IS_DATE não são internacionalizadas e devem ser inseridas em um dos formatos listados na seguinte tabela.

A seguinte tabela lista as cadeias de formato para as funções TO_DATE e IS_DATE:

Tabela 1. Strings de formato TO_DATE e IS_DATE

Cadeia de Formato	Descrição
AM, a.m., PM, p.m.	Indicador de meridiano. Use qualquer um dessas cadeias de formato para especificar as horas nos formatos AM e PM. AM e PM retornam os mesmos valores que a.m. e p.m.
DAY	Nome do dia, incluindo até nove caracteres (por exemplo, quarta-feira). A cadeia de formato DAY não faz distinção entre maiúsculas e minúsculas.
DD	Dia do mês (1-31).
DDD	Dia do ano (001 - 366, incluindo anos bissextos).
DY	Nome abreviado com três caracteres para um dia (por exemplo, qua.). A cadeia de formato DY não faz distinção entre maiúsculas e minúsculas.
HH, HH12	Hora do dia (1-12).
HH24	Hora do dia (0-23), na qual 0 é 12 AM (meia-noite).
J	Dia Juliano Modificado. Converta cadeias no formato MJD em valores de data. Ele ignora o componente de hora da cadeia de origem, atribuindo a todas as datas o horário 00:00:00.000000000. Por exemplo, a expressão TO_DATE('2451544', 'J') converte 2451544 em Dez 31 1999 00:00:00.000000000.
MI	Minutos (0-59).
MM	Mês (1-12).
MONTH	Nome do mês, incluindo até nove caracteres (por exemplo, agosto). Não há distinção entre maiúsculas e minúsculas.
MON	Nome abreviado com três caracteres para um mês (por exemplo, ago). Não há distinção entre maiúsculas e minúsculas.
MS	Milissegundos (0-999).
NS	Nanossegundos (0-999999999).
RR	Ano com quatro dígitos (por exemplo, 1998, 2034). Use quando as cadeias de origem incluírem anos com dois dígitos. Use com TO_DATE para converter anos com dois dígitos em anos com quatro dígitos. - Ano atual entre 50 e 99. Se o ano atual estiver entre 50 e 99 (como 1998) e o valor do ano da string de origem estiver entre 0 e 49, o Data Integration Service retornará o século seguinte mais o ano de dois dígitos na string de origem. Se o valor do ano da string de origem estiver entre 50 e 99, o Data Integration Service retornará o século atual mais o ano de dois dígitos especificado. - Ano atual entre 0 e 49. Se o ano atual estiver entre 0 e 49 (como 2003) e o ano da string de origem estiver entre 0 e 49, o Data Integration Service retornará o século atual mais o ano de dois dígitos na string de origem. Se o ano da string de origem estiver entre 50 e 99, o Data Integration Service retornará o século anterior mais o ano de dois dígitos na string de origem.
SS	Segundos (0-59).

Cadeia de Formato	Descrição
SSSSS	Segundos desde a meia-noite. Quando você usar SSSSS em uma expressão TO_DATE, o Data Integration Service avalia apenas parte da hora de uma data. Por exemplo, a expressão TO_DATE(DATE_STR, 'MM/DD/YYYY SSSSS') converte 12/31/1999 3783 em 12/31/1999 01:02:03.
US	Microssegundos (0-999999).
Y	O ano atual no nó que executa o Data Integration Service com o último dígito do ano substituído pelo valor da cadeia.
YY	O ano atual no nó que executa o Data Integration Service com os últimos dois dígitos do ano substituídos pelo valor da cadeia.
YYY	O ano atual no nó que executa o Data Integration Service com os últimos três dígitos do ano substituídos pelo valor da cadeia.
YYYY	Quatro dígitos de um ano. Não use essa cadeia de formato se estiver passando anos com dois dígitos. Use a cadeia de formato RR ou YY.

Regras e diretrizes para strings de formato de data

Use as seguintes regras e diretrizes ao trabalhar com strings de formato de data:

- O formato da cadeia TO_DATE deve corresponder à cadeia de formato. Se não corresponder, o Data Integration Service talvez retorne valores imprecisos ou ignore a linha. Por exemplo, se você passar a string '20200512', que representa 12 de maio de 2020, para TO_DATE, você deve incluir a string de formato YYYYMMDD. Se você não incluir uma cadeia de formato, o Data Integration Service esperará uma cadeia no formato de data especificado na configuração do visualizador de dados. Da mesma forma, se você passar uma cadeia que não corresponda à cadeia de formato, o Data Integration Service retornará um erro e ignorará a linha. Por exemplo, se você passar a cadeia 2020120 para TO_DATE e incluir a cadeia de formato YYYYMMDD, o Data Integration Service retornará um erro e ignorará a linha porque a cadeia não corresponde à cadeia de formato.
- A string de formato deve estar entre aspas simples.
- O Data Integration Service usa o formato de data e hora padrão especificado na sessão. O padrão é MM/DD/YYYY HH24:MI:SS.US. A string de formato não faz distinção entre maiúsculas e minúsculas.

Exemplo

O exemplo a seguir ilustra as strings de formato J, RR e SSSSS. Consulte as funções individuais para obter mais exemplos.

String no formato J

A seguinte expressão converte strings na porta SHIP_DATE_MJD_STRING em valores de data no formato de data padrão:

```
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

Como a string de formato J não inclui a parte de tempo de uma data, os valores retornados têm a hora definida como 00:00:00.000000000.

String de formato RR

A expressão a seguir converte uma string em um formato de ano de quatro dígitos. O ano atual é 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

string de formato YY

A expressão a seguir converte uma string em um formato de ano de quatro dígitos. O ano atual é 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: Na segunda linha, RR retorna o ano 2005, mas YY retorna o ano 1905.

String de Formato SSSSS

A seguinte expressão converte strings que incluem os segundos desde a meia-noite em valores de data:

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

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

Noções básicas dos cálculos aritméticos de data

A linguagem de transformação fornece funções de data internas para que você possa realizar operação aritmética em valores de data e hora da seguinte forma:

- **ADD_TO_DATE.** Adicione ou subtraia uma parte específica de uma data.
- **DATE_DIFF.** Subtraia duas datas.
- **SET_DATE_PART.** Altera uma parte de uma data.

Você não pode usar operadores aritméticos numéricos (como + ou -) para adicionar ou subtrair datas.

A linguagem de transformação reconhece anos bissextos e aceita datas entre 1º de janeiro de 0001 00:00:00.000000000 A.D. e 31 de dezembro de 9999 23:59:59.999999999 A.D.

Nota: A linguagem de transformação usa o tipo de dados Data/Hora da transformação para especificar valores de data. Você só pode usar as funções de data em valores de data e hora.

CAPÍTULO 6

Funções

Este capítulo descreve as funções na linguagem de transformação em ordem alfabética. A descrição de cada função inclui:

- Sintaxe
- Valor retornado
- Exemplo

Categorias de função

A linguagem de transformação fornece os seguintes tipos de funções:

- Agregado
- Caractere
- Complexas
- Conversão
- Limpeza de Dados
- Data
- Codificação
- Financeiro
- Numérico
- Científico
- Especial
- String
- Teste
- Variável
- Janela

Funções de agregação

As funções de agregação retornam valores de resumo para valores não nulos em portas selecionadas. Com funções de agregação, é possível:

- Calcular um único valor para todas as linhas em um grupo.

- Retornar um valor único para cada grupo em uma transformação de Agregador.
- Aplicar filtros para calcular valores de linhas específicas nas portas selecionadas.
- Usar operadores para realizar operação aritmética dentro da função.
- Calcular dois ou mais valores agregados, derivados das mesmas colunas de origem em uma única passagem.

A linguagem de transformação inclui as seguintes funções de agregação:

- ANY
- AVG
- COLLECT_LIST
- COUNT
- FIRST
- LAST
- MAX (Data)
- MAX (Número)
- MAX (String)
- MEDIAN
- MIN (Data)
- MIN (Número)
- MIN (String)
- PERCENTILE
- STDDEV
- SUM
- VARIANCE

Se você configurar a execução do Data Integration Service em modo Unicode, MIN e MAX retornarão valores de acordo com a ordem de classificação da página de código especificada na configuração do mapeamento.

Você pode usar funções agregadas em transformações de Agregador. É possível aninhar apenas uma função de agregação dentro de outra função de agregação. O Data Integration Service avalia a expressão mais interna da função de agregação e usa o resultado para avaliar a expressão externa. Você pode configurar uma transformação do Agregador que agrupa por ID e aninha duas funções de agregação da seguinte forma:

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

onde o conjunto de dados contém os seguintes valores:

ID	EARNINGS
1	32
1	45
1	100
2	65
2	75

ID	EARNINGS
2	76
3	21
3	45
3	99

O valor retornado é 186. O Data Integration Service agrupa por ID, avalia a expressão AVG e retorna três valores. Em seguida, ele adiciona os valores com a função SUM para obter o resultado.

Você também pode usar funções agregadas como funções de janela em uma transformação de Expressão. Para usar uma função de agregação como uma função de janela ao executar um mapeamento no mecanismo Spark, você deve configurar a transformação para definição de janelas. Se você usar uma função de agregação como uma função de janela, a transformação de Expressão se tornará ativa.

Funções de agregação e Nulls

Ao configurar o Data Integration Service, você pode escolher como quer tratar valores nulos em funções de agregação. O Data Integration Service pode tratar valores nulos em funções de agregação como NULL ou 0.

Por padrão, o Data Integration Service trata valores nulos como NULL em funções de agregação. Se você passar uma porta ou um grupo completo de valores nulos, a função retornará NULL. Você também poderá configurar o Data Integration Service se passar uma porta completa de valores nulos para uma função de agregação e retornar 0.

Condições do Filtro

Use uma condição de filtro para limitar as linhas retornadas em uma pesquisa.

Um filtro limita as linhas retornadas em uma pesquisa. Você pode aplicar uma condição de filtro a todas as funções de agregação e a CUME, MOVINGAVG e MOVINGSUM. A condição de filtro deve ser avaliada como TRUE, FALSE ou NULL. Se a condição de filtro for avaliada como NULL ou FALSE, o Data Integration Service não selecionará a linha.

Você pode inserir qualquer expressão de transformação válida. Por exemplo, a seguinte expressão calcula o salário médio de todos os funcionários que ganham mais de US$ 50.000:

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

Você também pode usar outros valores numéricos como a condição de filtro. Por exemplo, você pode inserir o seguinte como a sintaxe completa para a função MEDIAN, incluindo uma porta numérica:

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

Em todos os casos, o Data Integration Service arredonda valores decimais para um inteiro (por exemplo, 1,5 para 2; 1,2 para 1; 0,35 para 0) na condição de filtro. Se o valor for arredondado para 0, a condição de filtro retornará FALSE. Se você não quiser arredondar um valor, use a função TRUNC para truncar o valor em um inteiro:

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

Se você omitir a condição de filtro, a função selecionará todas as linhas na porta.

Funções de caractere

A linguagem de transformação inclui as seguintes funções de caracteres:

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

As funções de caracteres MAX, MIN, LOWER, UPPER e INITCAP usam a página de código do Data Integration Service para avaliar dados de caracteres.

Funções complexas

Uma função complexa é um tipo de função predefinida em que o valor da entrada ou o tipo de retorno é de um tipo de dados complexo, como array, map ou struct. Você pode usar funções complexas em mapeamentos que são executados no mecanismo Spark.

A linguagem de transformação inclui as seguintes funções complexas:

- ARRAY
- CAST
- COLLECT_LIST
- CONCAT_ARRAY
- RESPEC
- SIZE
- STRUCT
- STRUCT_AS

Funções de conversão

A linguagem de transformação inclui as seguintes funções de conversão:

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

Funções de Limpeza de Dados

A linguagem de transformação inclui um grupo de funções para eliminar erros de dados. Você pode concluir as seguintes tarefas com as funções de limpeza de dados:

- Testar valores de entrada.
- Converter o tipo de dados de um valor de entrada.
- Cortar valores de string.
- Substituir caracteres em uma string.
- Codificar strings.
- Combinar padrões em expressões regulares.

A linguagem de transformação inclui as seguintes funções de limpeza de dados:

- 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

Funções de data

A linguagem de transformação inclui um grupo de funções de data para arredondar, truncar ou comparar datas, extrair uma parte de uma data ou realizar operação aritmética em uma data.

Você pode passar qualquer valor com um tipo de dados de data para qualquer uma das funções de data. No entanto, se quiser passar uma string para uma função de data, você deverá primeiro usar a função TO_DATE para convertê-la em um tipo de dados de DATA/HORA de transformação.

A linguagem de transformação inclui as seguintes funções de data:

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

Várias funções de data incluem um argumento de *formato*. Você deve especificar uma das strings de formato de linguagem de transformação para esse argumento. As strings de formato de data não são internacionalizadas.

O tipo de dados de transformação de Data/Hora oferece suporte a datas com precisão de nanossegundo.

Tópicos Relacionados:

- "Strings de Formato de Data" na página 40

Funções de codificação

A linguagem de transformação inclui as seguintes funções para criptografia de dados, compactação, codificação e soma de verificação:

- AES_DECRYPT
- AES_ENCRYPT

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

Funções financeiras

A linguagem de transformação inclui as seguintes funções financeiras:

- VF
- NPER
- PMT
- VP
- RATE

Funções numéricas

A linguagem de transformação inclui as seguintes funções numéricas:

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

Funções científicas

A linguagem de transformação inclui as seguintes funções financeiras:

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

Funções especiais

A linguagem de transformação inclui as seguintes funções especiais:

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

Geralmente, você usa funções especiais em transformações de Expressão, Filtro e Estratégia de Atualização. É possível aninhar outras funções dentro de funções especiais. Além disso, é possível aninhar uma função especial em uma função de agregação.

Funções de string

A linguagem de transformação inclui as seguintes funções de string:

- CHOOSE
- INDEXOF
- MAX
- MIN
- REVERSE

Funções de teste

A linguagem de transformação inclui as seguintes funções de teste:

- ISNULL
- IS_DATE
- IS_NUMBER
- IS_SPACES

Funções de janela

A linguagem de transformação inclui um grupo de funções de janela que realizam cálculos em um conjunto de linhas que estão relacionadas à linha atual. As funções calculam um valor de retorno único para cada linha de entrada. Você pode usar funções de janela em mapeamentos que são executados no mecanismo Spark.

A linguagem de transformação inclui as seguintes funções de janela:

- LAG
- LEAD

Você pode usar funções de janela em transformações de Expressão. Se você usar uma função de janela em uma transformação de Expressão, a transformação estará ativa.

ABORT

Interrompe a execução do mapeamento e emite uma mensagem de erro especificada para o log. Quando encontra uma função ABORT, o Data Integration Service para de transformar dados na linha. Processa as linhas lidas antes da anulação do mapeamento. O Data Integration Service grava no destino até a linha anulada e reverte todos os dados não confirmados para o último ponto de confirmação.

Use ABORT para validar os dados. Geralmente, você usa ABORT em uma função IIF ou DECODE a fim de definir regras para anular uma sessão.

Use a função ABORT para os valores padrão da porta de entrada e de saída. Você pode usar ABORT em portas de entrada para impedir que valores nulos passem em uma transformação. Também é possível usar ABORT para controlar qualquer tipo de erro de transformação, inclusive as chamadas da função ERROR em uma expressão. O valor padrão substitui a função ERROR em uma expressão. Se você quiser verificar se a sessão é interrompida quando um erro ocorre, atribua ABORT como o valor padrão.

Se você usar ABORT em uma expressão de uma porta não conectada, o Data Integration Service não irá executá-la.

Sintaxe

```
ABORT( string )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
string	Requerido	String. A mensagem que você quer exibir no log quando a execução do mapeamento é interrompida. A string pode ter qualquer tamanho. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

NULL.

ABS

Retorna o valor absoluto de um valor numérico.

Sintaxe

```
ABS( numeric_value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
valor_numérico	Requerido	Tipo de dados Numérico. Passa os valores para os quais você deseja retornar os valores absolutos. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Valor numérico positivo. ABS retorna o mesmo tipo de dados que o valor numérico passou como um argumento. Se você passar um Double, ele retornará um Double. Da mesma forma, se você passar um Integer, ele retornará um Integer.

NULL se você passar um valor nulo para a função.

Nota: Se o valor retornado for Decimal com precisão superior a 15, habilite a alta precisão para garantir precisão decimal de até 28 dígitos.

Exemplo

A seguinte expressão retorna a diferença entre dois números como um valor positivo, independentemente de qual número seja maior:

```
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

Adiciona uma quantidade especificada a uma parte de um valor de data e hora e retorna uma data no mesmo formato que a data que você passa para a função. ADD_TO_DATE aceita valores inteiros positivos e negativos. Use ADD_TO_DATE para alterar as seguintes partes de uma data:

- **Ano.** Insira um inteiro positivo ou negativo no argumento *quantidade*. Use qualquer string de formato de ano: Y, YY, YYY ou YYYY. A seguinte expressão adiciona 10 anos a todas as datas na porta SHIP_DATE:

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

- **Mês.** Insira um inteiro positivo ou negativo no argumento *quantidade*. Use qualquer string de formato de ano: MM, MON, MONTH. A seguinte expressão subtrai 10 meses de cada data na porta SHIP_DATE:

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

- **Dia.** Insira um inteiro positivo ou negativo no argumento *quantidade*. Use qualquer string de formato de dia: D, DD, DDD, DY e DAY. A seguinte expressão adiciona 10 dias a todas as datas na porta SHIP_DATE:

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

- **Hora.** Insira um inteiro positivo ou negativo no argumento *quantidade*. Use qualquer string de formato de hora: HH, HH12, HH24. A seguinte expressão adiciona 14 horas a todas as datas na porta SHIP_DATE:

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

- **Minuto.** Insira um inteiro positivo ou negativo no argumento *quantidade*. Use a string de formato MI para definir o minuto. A seguinte expressão adiciona 25 minutos a todas as datas na porta SHIP_DATE:

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

- **Segundos.** Insira um inteiro positivo ou negativo no argumento *quantidade*. Use a string de formato SS para definir o segundo. A seguinte expressão adiciona 59 segundos a todas as datas na porta SHIP_DATE:

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

- **Milissegundos.** Insira um inteiro positivo ou negativo no argumento *quantidade*. Use a string de formato MS para definir os milissegundos. A seguinte expressão adiciona 125 milissegundos a todas as datas na porta SHIP_DATE:

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

- **Microssegundos.** Insira um inteiro positivo ou negativo no argumento *quantidade*. Use a string de formato US para definir os microssegundos. A seguinte expressão adiciona 2.000 microssegundos a todas as datas na porta SHIP_DATE:

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

- **Nanossegundos.** Insira um inteiro positivo ou negativo no argumento *quantidade*. Use a string de formato NS para definir os nanossegundos. A seguinte expressão adiciona 3.000.000 minutos a todas as datas na porta SHIP_DATE:

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

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
data	Requerido	Tipo de dados Data/Hora. Passa os valores que você deseja alterar. Você pode inserir qualquer expressão de transformação válida.
formato	Requerido	Uma string de formato que especifica a parte do valor de data que você deseja alterar. Coloque a string de formato entre aspas simples, por exemplo, 'mm'. A string de formato não faz distinção entre maiúsculas e minúsculas.
quantidade	Requerido	Um valor inteiro que especifica a quantidade de anos, meses, dias, horas e etc, pela qual você deseja alterar o valor de data. Você pode inserir qualquer expressão de transformação válida que avalie em um inteiro.

Valor de Retorno

Data no mesmo formato que a data que você passa para essa função.

NULL se um valor nulo for passado como um argumento para a função.

Exemplos

As expressões a seguir adicionam um mês à cada data na porta DATE_SHIPPED. Se você passar um valor que crie um dia inexistente em determinado mês, o Data Integration Service retornará o último dia do mês. Por exemplo, se você somar um mês a 31 de janeiro de 1998, o Data Integration Service retornará 28 de fevereiro de 1998.

Observe também que ADD_TO_DATE reconhece anos bissextos e adiciona um mês a Jan 29 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

As seguintes expressões subtraem 10 dias de cada data na porta 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

DATE_SHIPPED	RETURN VALUE
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

As seguintes expressões subtraem 15 horas de cada data na porta 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

Trabalhando com Datas

Use as seguintes dicas ao trabalhar com ADD_TO_DATE:

- Você pode subtrair qualquer parte da data especificando uma string de formato e tornando o argumento *quantidade* um inteiro positivo ou negativo.
- Se você passar um valor que crie um dia inexistente em determinado mês, o Data Integration Service retornará o último dia do mês. Por exemplo, se você somar um mês a 31 de janeiro de 1998, o Data Integration Service retornará 28 de fevereiro de 1998.
- Você pode aninhar TRUNC ou ROUND para manipular datas.
- Você pode aninhar TO_DATE para converter strings em datas.
- ADD_TO_DATE altera apenas uma parte da data, que você especifica. Se você modificar uma data de modo que ela altere de padrão para horário de verão, será necessário alterar a parte da hora da data.

AES_DECRYPT

Retorna dados descriptografados ao formato de string. O Data Integration Service usa o algoritmo Padrão Avançado de Criptografia (AES) com codificação de 128 bits. O algoritmo AES é um algoritmo criptográfico aprovado pelo FIPS.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
valor	Requerido	Tipo de dados Binário. Valor que você deseja descriptografar.
chave	Requerido	Tipo de dados strings. Precisão de 16 caracteres ou menos. Você pode usar variáveis de mapeamento para a chave. Use a mesma chave para descriptografar um valor que você usou para criptografá-lo.

Valor de Retorno

Valor binário descriptografado.

NULL se o valor de entrada for nulo.

Exemplo

O exemplo a seguir retorna números de previdência social descriptografados. No exemplo, o Data Integration Service deriva a chave dos primeiros três números do número de previdência social usando a função SUBSRT:

```
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

Retorna dados em um formato criptografado. O Data Integration Service usa o algoritmo Padrão Avançado de Criptografia (AES) com codificação de 128 bits. O algoritmo AES é um algoritmo criptográfico aprovado pelo FIPS.

Use essa função para impedir que dados confidenciais fiquem visíveis a todos. Por exemplo, para armazenar números de previdência social em um data warehouse, use a função AES_ENCRYPT para criptografar os números de previdência social a fim de manter a confidencialidade.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
valor	Requerido	Tipo de dados strings. Valor que você deseja criptografar.
chave	Requerido	Tipo de dados strings. Precisão de 16 caracteres ou menos. Você pode usar variáveis de mapeamento para a chave.

Valor de Retorno

Valor binário criptografado.

NULL se a entrada for um valor nulo.

Exemplo

O exemplo a seguir retorna números de previdência social criptografados. No exemplo, o Data Integration Service deriva a chave dos primeiros três números do número de previdência social usando a função SUBSRT:

```
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

Dica

Se o destino não oferecer suporte a dados binários, use AES_ENCRYPT com a função ENC_BASE64 para armazenar dados em um formato compatível com o banco de dados.

ANY

Retorna a linha any na porta selecionada. Opcionalmente, você também pode aplicar um filtro para limitar as linhas lidas pelo Serviço de Integração de Dados. É possível aninhar somente uma função de agregação em ANY.

Sintaxe

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

A tabela a seguir descreve os argumentos para esta função:

Argumento	Obrigatório / Opcional	Descrição
value	Obrigatório	Qualquer tipo de dados, exceto Binário. Transmite os valores para os quais você deseja retornar qualquer linha. Você pode inserir qualquer expressão válida de transformação.
filter_condition	Opcional	Limita as linhas na pesquisa. A condição de filtro deve ser um valor numérico ou ser avaliada como TRUE, FALSE ou NULL. Você pode inserir qualquer expressão válida de transformação.

Valor de retorno

Qualquer linha em uma porta. Retorna uma linha diferente a cada vez.

NULL se todos os valores transmitidos para a função forem NULL ou se nenhuma linha for selecionada. Por exemplo, a condição de filtro é avaliada como FALSE ou NULL em todas as linhas.

Exemplo

A seguinte expressão retorna qualquer linha na porta ITEM_NAME com um valor maior que US$ 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

ANY e tipos de dados complexos

Você pode usar ANY para retornar uma linha em uma porta complexa do tipo array ou struct.

Por exemplo, você tem o seguinte array:

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

Você pode usar a seguinte expressão para retornar qualquer linha na porta do array:

```
ANY( emp_phones )
```

RETURN VALUE: [205-128-6478, 722-515-2889]

ARRAY

Gera uma matriz com elementos baseados nos argumentos especificados.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório/ Opcional	Descrição
array_element1	Obrigatório	Qualquer tipo de dados. O elemento que você deseja adicionar ao array. Você pode inserir qualquer expressão válida de transformação.
array_element2	Opcional	Mesmo tipo de dados que array_element1.

Se você usar a função ARRAY em uma expressão de saída para uma porta de array, o tipo de dados dos argumentos da função deverá corresponder ao tipo de dados dos elementos de array especificados na configuração de tipo da porta de array.

Valor de retorno

Array.

O tipo de dados dos argumentos determina o tipo de dados dos elementos de array. Por exemplo, se você transmitir argumentos de string, a função gerará um array de elementos de string.

Exemplos

A expressão a seguir gera um array de elementos de string.

```
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

Quando o Data Integration Service usa o modo ASCII, a função ASCII retorna o valor numérico ASCII do primeiro caractere da string transmitida para a função.

Quando o Data Integration Service usa o modo Unicode, a função ASCII retorna o valor numérico Unicode do primeiro caractere da string transmitida para a função. Os valores Unicode encontram-se no intervalo de 0 a 65,535.

Você pode passar uma string de qualquer tamanho para ASCII, mas ela avalia apenas o primeiro caractere na string. Antes de passar qualquer valor de string para ASCII, você pode analisar o caractere específico que deseja converter em um valor ASCII ou Unicode. Por exemplo, você pode usar RTRIM ou outra função de manipulação de string. Se você passar um valor numérico, ASCII o converterá em uma string de caracteres e retornará o valor ASCII ou Unicode do primeiro caractere na string.

Essa função é idêntica em comportamento à função CHRCODE. Se você usar ASCII em expressões existentes, elas ainda funcionarão corretamente. No entanto, ao criar novas expressões, use a função CHRCODE em vez da função ASCII.

Sintaxe

```
ASCII ( string )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Obrigatório / Opcional	Descrição
string	Obrigatório	String de caracteres. Transmite o valor que você deseja retornar como um valor ASCII. Você pode inserir qualquer expressão válida de transformação.

Valor de Retorno

Número inteiro. O valor ASCII ou Unicode do primeiro caractere na string.

NULL se um valor transmitido para a função for NULL.

Exemplo

A seguinte expressão retorna o valor ASCII ou Unicode do primeiro caractere de cada valor na porta ITEMS:

```
ASCII( ITEMS )
```

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

AVG

Retorna a média de todos os valores em um grupo de linhas. Como opção, você pode aplicar um filtro para limitar as linhas lidas para calcular a média. Você pode aninhar apenas uma outra função de agregação dentro de AVG e a função aninhada deverá retornar um tipo de dados Numérico.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
valor_numérico	Requerido	Tipo de dados Numérico. Passa os valores para os quais você deseja calcular uma média. Você pode inserir qualquer expressão de transformação válida.
filter_condition	Opcional	Limita as linhas na pesquisa. A condição de filtro deve ser um valor numérico ou avaliada como TRUE, FALSE ou NULL. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Valor numérico.

NULL se todos os valores passados para a função forem NULL ou se nenhuma linha for selecionada. Por exemplo, a condição de filtro é avaliada como FALSE ou NULL em todas as linhas.

Nota: Se o valor retornado for Decimal com precisão superior a 15, habilite a alta precisão para garantir precisão decimal de até 28 dígitos.

Nulls

Se um valor for NULL, a função AVG ignorará a linha. No entanto, se todos os valores passados da porta forem NULL, AVG retornará NULL.

Agrupar por

AVG agrupa valores baseados em grupos por portas que você define na transformação, retornando um resultado para cada grupo.

Se não houver grupo por porta, AVG tratará todas as linhas como um grupo, retornando um valor.

Exemplo

A seguinte expressão retorna o custo médio de venda por atacado de lanternas:

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

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

RETURN VALUE: 31.66

Dica

Você pode realizar operação aritmética nos valores passados para AVG antes que a função calcule a média. Por exemplo:

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

CAST

Renomeia os elementos e altera o tipo de dados de cada elemento para o valor de struct determinado com base no tipo de dados na definição de tipo de dados complexo especificada.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório/ Opcional	Descrição
:Type.type_definition_library.type_definition	Obrigatório	A definição de tipo de dados complexo que representa o esquema dos dados de struct. Use o qualificador de referência `:Type` para referenciar a biblioteca de definições de tipo que contém a definição de tipo de dados complexo.
struct_value	Obrigatório	O valor de struct para o qual você deseja alterar o tipo de dados dos elementos de struct. É possível inserir qualquer expressão de transformação válida que seja avaliada como um struct.

O tipo de dados do valor de struct e o tipo de dados na definição de tipo de dados complexo devem ser compatíveis.

Valor de retorno

Struct.

Exemplos

A expressão a seguir altera os tipos de dados dos elementos na porta de struct h2_sales com base nos tipos de dados na definição de tipo de dados complexo h1_sales_def.

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

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

CEIL

Retorna o menor número inteiro maior que ou igual ao valor numérico passado para essa função. Por exemplo, se você passar 3.14 para CEIL, a função retornará 4. Se você passar 3.98 para CEIL, a função retornará 4. Da mesma forma, se você passar -3.17 para CEIL, a função retornará -3.

Sintaxe

```
CEIL( numeric_value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Obrigatório / Opcional	Descrição
valor_numérico	Obrigatório	Tipo de dados numérico. Você pode inserir qualquer expressão válida de transformação.

Valor de Retorno

Valor numérico.

Valor duplo se você passar um valor numérico com precisão declarada maior que 38.

NULL se um valor transmitido para a função for NULL.

Exemplo

A seguinte expressão retorna o preço arredondado para o próximo número inteiro:

```
CEIL( PRICE )
```

PRICE	RETURN VALUE
39.79	40
125.12	126
74.24	75
NULL	NULL
-100.99	-100

Sugestão: Você pode realizar operação aritmética nos valores passados para CEIL antes que CEIL retorne o próximo valor inteiro. Por exemplo, se você deseja multiplicar um valor numérico por 10 antes de calcular o menor número inteiro maior que o valor modificado, poderá gravar a função da seguinte forma:

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

CHOOSE

Escolha uma string na lista de strings com base em uma determinada posição. Especifique a posição e o valor. Se o valor corresponder à posição, o Data Integration Service irá retorná-lo.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
índice	Requerido	Tipo de dados Numérico. Insira um número com base na posição do valor que deseja corresponder.
string	Requerido	Qualquer valor de caractere.

Valor de Retorno

A string que corresponde à posição do valor de índice.

NULL se nenhuma string corresponder à posição do valor de índice.

Exemplo

A seguinte expressão retorna a string ‘flashlight’ com base em um valor de índice 2:

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

A seguinte expressão retorna NULL com base em um valor de índice 4:

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

CHOOSE retorna NULL porque a expressão não contém um quarto argumento.

CHR

Quando o Data Integration Service usa o modo ASCII, o CHR retorna o caractere ASCII correspondente ao valor numérico que você transmite para a função. Os valores ASCII encontram-se no intervalo de 0 a 255. Você pode passar qualquer inteiro para CHR, mas somente os códigos ASCII 32 a 126 são caracteres imprimíveis.

Quando o Data Integration Service usa o modo Unicode, o CHR retorna caractere Unicode correspondente ao valor numérico que você transmite para a função. Os valores Unicode encontram-se no intervalo de 0 a 65,535.

Sintaxe

```
CHR( numeric_value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Obrigatório / Opcional	Descrição
numeric_value	Obrigatório	Tipo de dados Numérico. O valor que você deseja retornar como um caractere ASCII ou Unicode. Você pode inserir qualquer expressão válida de transformação.

Valor de Retorno

Caractere ASCII ou Unicode. Uma string que contém um caractere.

NULL se um valor transmitido para a função for NULL.

Exemplo

A seguinte expressão retorna o caractere ASCII ou Unicode de cada valor numérico na porta ITEM_ID:

```
CHR( ITEM_ID )
```

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

Use a função CHR para concatenar uma aspa simples em uma string. A aspa simples é o único caractere que você não pode usar dentro de um literal de string. Considere o seguinte exemplo:

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

O valor retornado é:

```
Joan's car
```

CHRCODE

Quando o Data Integration Service usa o modo ASCII, o CHRCODE retorna o valor numérico ASCII do primeiro caractere da string transmitida para a função. Os valores ASCII encontram-se no intervalo de 0 a 255.

Quando o Data Integration Service usa o modo Unicode, o CHRCODE retorna o valor numérico Unicode do primeiro caractere da string transmitida para a função. Os valores Unicode encontram-se no intervalo de 0 a 65,535.

Normalmente, antes de passar qualquer valor de string para CHRCODE, você pode analisar o caractere específico que deseja converter em um valor ASCII ou Unicode. Por exemplo, você pode usar RTRIM ou outra função de manipulação de string. Se você passar um valor numérico, CHRCODE o converterá em uma string de caracteres e retornará o valor ASCII ou Unicode do primeiro caractere na string.

A função é idêntica em comportamento à função ASCII. Se você usar ASCII em expressões, ele ainda funcionará corretamente. No entanto, ao criar novas expressões, use a função CHRCODE em vez da função ASCII.

Sintaxe

```
CHRCODE ( string )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Obrigatório / Opcional	Descrição
string	Obrigatório	String de caracteres. Transmite os valores que você deseja retornar como valores ASCII ou Unicode. Você pode inserir qualquer expressão válida de transformação.

Valor de Retorno

Número inteiro.

NULL se um valor transmitido para a função for NULL.

Exemplo

A seguinte expressão retorna o valor ASCII ou Unicode do primeiro caractere de cada valor na porta ITEMS:

```
CHRCODE( ITEMS )
```

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

COLLECT_LIST

Retorna uma matriz com elementos baseados no argumento. O tipo de dados do argumento determina o tipo de dados da matriz. COLLECT_LIST é uma função agregada.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório/ Opcional	Descrição
value	Obrigatório	Qualquer tipo de dados. Os valores que você deseja agregar em dados hierárquicos do tipo array. Você pode inserir qualquer expressão válida de transformação.

Valor de retorno

Array.

Agrupar por

COLLECT_LIST agrupa valores com base em portas Agrupar por que você define na transformação, retornando um único resultado para cada grupo.

Se não houver uma porta Agrupar por, COLLECT_LIST tratará todas as linhas como um grupo, retornando um único valor.

Exemplos

A expressão a seguir retorna um array com os elementos no PRODUCT_NAME.

```
COLLECT_LIST(PRODUCT_NAME)
```

```
PRODUCT_NAME

Flashlight

Compass

Pressure Gauge

Vest

RETURN VALUE: [Flashlight,Compass,Pressure Gauge,Vest]
```

COMPRESS

Compacta dados usando o algoritmo de compactação zlib 1.2.1. Use a função COMPRESS antes de enviar grandes quantidades de dados por meio de uma rede de longa distância.

Sintaxe

```
COMPRESS( value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
valor	Requerido	Tipo de dados strings. Dados que você deseja compactar.

Valor de Retorno

Valor binário compactado do valor de entrada.

NULL se a entrada for um valor nulo.

Exemplo

Sua organização tiver um serviço de pedido on-line. Você deseja receber dados de pedido do cliente compactados em uma rede de longa distância. A origem contém uma linha com 10 MB. Você pode compactar os dados nessa linha usando COMPRESS. Ao compactar os dados, você reduz a quantidade de dados que o Data Integration Service grava na rede. Como resultado, o desempenho da sessão pode aumentar.

CONCAT

Concatena duas strings. CONCAT converte todos os dados em texto antes de concatenar as strings. Como alternativa, use o operador de string || para concatenar strings. O uso do operador de string || em vez de CONCAT aumenta o desempenho do Data Integration Service.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
first_string	Requerido	Qualquer tipo de dados, exceto Binário. A primeira parte da string que você deseja concatenar. Você pode inserir qualquer expressão de transformação válida.
second_string	Requerido	Qualquer tipo de dados, exceto Binário. A segunda parte da string que você deseja concatenar. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

String.

NULL se ambos os valores de string forem NULL.

Nulls

Quando uma das strings é NULL, a função CONCAT ignora-a e retorna a outra string.

Se ambas as strings forem NULL, CONCAT retornará NULL.

Exemplo

A seguinte expressão concatena os nomes nas portas FIRST_NAME e 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 não adiciona espaços a string separadas. Se adicionar um espaço entre duas strings, você poderá gravar uma expressão com duas funções CONCAT aninhadas. Por exemplo, a seguinte expressão primeiro concatena um espaço no final do nome e depois concatena o sobrenome:

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

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

Use as funções CHR e CONCAT para concatenar uma aspa simples em uma string. A aspa simples é o único caractere que você não pode usar dentro de um literal de string. Considere o seguinte exemplo:

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

O valor retornado é:

```
Joan's car
```

CONCAT_ARRAY

Concatena elementos de cadeia em um array com base em um separador que você especifica e retorna uma string.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório/ Opcional	Descrição
''	Obrigatório	Cada elemento de string é separado pelo separador especificado. Por exemplo, ', ' separa os valores com uma vírgula.
array	Obrigatório	Uma matriz com elementos do tipo string. O array que você deseja concatenar.

Valor de retorno

String

Nulos

Se um dos elementos de string for NULL, CONCAT_ARRAY o ignorará e retornará a outra string.

Se todos os elementos de string forem NULL, CONCAT_ARRAY retornará uma string vazia.

Exemplos

A expressão a seguir concatena os elementos de string no array.

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

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

CONVERT_BASE

Converte uma string numérica não negativa de um valor base em outro valor base.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
valor	Obrigatório	Tipo de dados de string. Valor que você deseja converter de uma base em outra. O máximo é 9.233.372.036.854.775.806.
source_base	Obrigatório	Tipo de dados Numérico. Valor base atual dos dados que você deseja converter. A base mínima é 2. A base máxima é 36.
dest_base	Obrigatório	Tipo de dados Numérico. Valor base no qual você deseja converter os dados. A base mínima é 2. A base máxima é 36.

Valor de Retorno

Valor numérico.

Exemplo

O seguinte exemplo converte 2222 do valor base decimal 10 no valor base binário 2:

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

O Data Integration Service retorna 100010101110.

COS

Retorna o cosseno de um valor numérico (expresso em radianos).

Sintaxe

```
COS( numeric_value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
valor_numérico	Requerido	Tipo de dados Numérico. Dados numéricos expressos em radianos (graus multiplicados por pi divididos por 180). Passa os valores para os quais você deseja calcular um cosseno. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Valor duplo.

NULL se um valor transmitido para a função for NULL.

Exemplo

A seguinte expressão retorna o co-seno de todos os valores na porta Graus:

```
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

Sugestão: Você pode realizar operação aritmética nos valores passados para COS antes que a função calcule o cosseno. Por exemplo, você pode converter os valores na porta em radianos antes de calcular o co-seno, da seguinte forma:

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

COSH

Retorna o cosseno hiperbólico de um valor numérico (expresso em radianos).

Sintaxe

```
COSH( numeric_value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
valor_numérico	Requerido	Tipo de dados Numérico. Dados numéricos expressos em radianos (graus multiplicados por pi divididos por 180). Passa os valores para os quais você deseja calcular o cosseno hiperbólico. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Valor duplo.

NULL se um valor transmitido para a função for NULL.

Exemplo

A seguinte expressão retorna o cosseno hiperbólico para os valores na porta Ângulos:

```
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

Sugestão: Você pode realizar operação aritmética nos valores passados para COSH antes que a função calcule o cosseno hiperbólico. Por exemplo:

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

COUNT

Retorna o número de linhas que têm valores não nulos em um grupo. Como opção, você pode incluir o argumento de asterisco (*) para contar todos os valores de entrada em uma transformação. Você pode

aninhar somente uma função de agregação dentro de COUNT. Você pode aplicar uma condição para filtrar linhas antes de contá-las.

Sintaxe

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

ou

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
value	Obrigatório	Qualquer tipo de dados, exceto Binário. Passa os valores que você deseja contar. Você pode inserir qualquer expressão válida de transformação.
*	Opcional	Use para contar *todas as linhas* em uma transformação.
filter_condition	Opcional	Limita as linhas na pesquisa. A condição de filtro deve ser um valor numérico ou ser avaliada como TRUE, FALSE ou NULL. Você pode inserir qualquer expressão válida de transformação.

Valor de retorno

Número inteiro.

0 se todos os valores passados para essa função forem NULL (a menos que você inclua o argumento de asterisco).

Nulls

Se todos os valores forem NULL, a função retornará 0.

Se você aplicar o argumento de asterisco, essa função contará todas as linhas, independentemente se uma coluna em uma linha que contém um valor nulo.

Se você aplicar o argumento de *valor*, essa função ignorará colunas com valores nulos.

Agrupar por

COUNT agrupa valores baseados em grupos por portas que você define na transformação, retornando um resultado para cada grupo. Se não houver grupo por porta, COUNT tratará todas as linhas como um grupo, retornando um valor.

Exemplos

A seguinte expressão conta os itens com menos de 5 quantidades em estoque, excluindo valores nulos:

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

ITEM_NAME	IN_STOCK
Flashlight	10
NULL	2
Compass	NULL

ITEM_NAME	IN_STOCK
Regulator System	5
Safety Knife	8
Halogen Flashlight	1

RETURN VALUE: 1

Neste exemplo, a função contou a lanterna Halogen, mas não o item NULL. A função conta todas as linhas em uma transformação, incluindo valores nulos, conforme ilustrado no seguinte exemplo:

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

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

RETURN VALUE: 2

Neste exemplo, a função conta o item NULL e a Lanterna Halogen. Se você incluir o argumento de asterisco, mas não usar um filtro, a função contará todas as linhas que passam na transformação. Por exemplo:

```
COUNT( * )
```

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

RETURN VALUE: 6

COUNT e tipos de dados complexos

Você pode usar COUNT para contar o número de linhas em uma porta complexa do tipo array ou struct.

Por exemplo, você tem o seguinte array:

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

Você pode usar a seguinte expressão para contar o número de linhas na porta de array:

```
COUNT( emp_phones )
```

RETURN VALUE: 4

CRC32

Retorna um valor CRC32 (verificação de redundância cíclica) de 32 bits. Use CRC32 para encontrar erros de transmissão de dados. Você também poderá usar CRC32 se desejar verificar se os dados armazenados em um arquivo foram modificados.

Se você usar CRC32 para realizar uma verificação de redundância em dados nos modos ASCII e Unicode, talvez o Data Integration Service gere resultados diferentes para o mesmo valor de entrada. Se você usar CRC32 para executar uma verificação de redundância em dados em diferentes sistemas operacionais, o Data Integration Service poderá gerar resultados diferentes no mesmo valor de entrada.

Nota: CRC32 pode retornar a mesma saída para strings de entrada diferentes. Se você quiser gerar chaves em um mapeamento, use a transformação de Gerador de Sequência. Se você usar CRC32 para gerar chaves em um mapeamento, poderá receber resultados inesperados.

Sintaxe

```
CRC32( value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Obrigatório / Opcional	Descrição
value	Obrigatório	Tipo de dados Cadeia ou Binário. Transmite os valores nos quais você deseja realizar uma verificação de redundância. O valor de entrada faz distinção entre maiúsculas e minúsculas. A distinção entre maiúsculas e minúsculas do valor de entrada afeta o valor retornado. Por exemplo, CRC32(informatica) e CRC32 (Informatica) retornam valores diferentes.

Valor de Retorno

Valor inteiro de 32 bits.

Exemplo

Você deseja ler dados de uma origem em uma rede de longa distância. Você deseja verificar se os dados foram modificados durante a transmissão. Você pode computar a soma de verificação para os dados no arquivo e armazená-los junto com o arquivo. Quando os dados de origem são lidos, o Data Integration Service usa o CRC32 para computar a soma de verificação e compará-la ao valor armazenado. Se os dois valores forem iguais, os dados não foram modificados.

CREATE_TIMESTAMP_TZ

Cria um tipo de dados de Registro de Data/Hora com Fuso Horário com base nos valores de registro de data/hora e fuso horário.

A porta de saída deve ser timestampWithTZ para expressões CREATE_TIMESTAMP_TZ.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
timestamp_value	Obrigatório	Tipo de dados Data/Hora. Você pode inserir qualquer expressão válida de transformação.
timezone_value	Obrigatório	Deve ser um tipo de dados de string. A string deve ser uma string de caracteres. Transmite os valores que você deseja criar para fuso horário. Você pode inserir uma expressão de transformação válida, conforme definido no arquivo de fuso horário presente na localização de instalação.

Valor de Retorno

Retorna um tipo de dados de registro de data/hora com fuso horário.

NULL se a entrada for um valor nulo.

Exemplo

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

Retorna um total de execução. Um total de execução significa que CUME retorna um total sempre que ele adiciona um valor. Você pode adicionar uma condição para filtrar linhas do conjunto de linhas antes de calcular o total de execução.

Use CUME e funções semelhantes (como MOVINGAVG e MOVINGSUM) para simplificar a geração de relatórios calculando os valores de execução.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
valor_numérico	Obrigatório	Tipo de dados Numérico. Passa os valores para os quais você deseja calcular um total de execução. Você pode inserir qualquer expressão válida de transformação. Você pode criar uma expressão aninhada para calcular um total de execução com base nos resultados da função desde que o resultado seja um valor numérico.
filter_condition	Opcional	Limita as linhas na pesquisa. A condição de filtro deve ser um valor numérico ou avaliada como TRUE, FALSE ou NULL. Você pode inserir qualquer expressão válida de transformação.

Valor de Retorno

Valor numérico.

NULL se todos os valores passados para a função forem NULL ou se nenhuma linha for selecionada (por exemplo, a condição de filtro avaliada como FALSE ou NULL em todas as linhas).

Nota: Se o valor retornado for Decimal com precisão superior a 15, habilite a alta precisão para garantir precisão decimal de até 28 dígitos.

Nulls

Se um valor for NULL, CUME retornará o total de execução para a linha anterior. No entanto, se todos os valores na porta selecionada forem NULL, CUME retornará NULL.

Exemplos

O seguinte conjunto de linhas de amostra pode resultar do uso da função CUME:

```
CUME( PERSONAL_SALES )
```

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

Da mesma forma, você pode adicionar valores antes de calcular um total de execução:

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

CA_SALES	OR_SALES	RETURN VALUE
40000	10000	50000

CA_SALES	OR_SALES	RETURN VALUE
80000	50000	180000
40000	2000	222000
60000	NULL	222000
NULL	NULL	222000
50000	3000	275000

DATE_COMPARE

Retorna um inteiro que indica qual das duas datas é anterior. DATE_COMPARE retorna um valor inteiro, em vez de um valor de data.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
data1	Requerido	Tipo de dados Data/Hora. A primeira data que você deseja comparar. Você pode inserir qualquer expressão de transformação válida desde que ela seja avaliada como uma data.
data2	Requerido	Tipo de dados Data/Hora. A segunda data que você deseja comparar. Você pode inserir qualquer expressão de transformação válida desde que ela seja avaliada como uma data.

Valor de Retorno

-1 se a primeira data for anterior.

0 se as duas datas forem iguais.

1 se a segunda data for anterior.

NULL se um dos valores de data for NULL.

Exemplo

A seguinte expressão compara cada data nas portas DATE_PROMISED e DATE_SHIPPED e retorna um inteiro que indica qual é a data anterior:

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

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

DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
Feb 1 1997	Feb 1 1997	0
Dec 22 1997	Dec 15 1997	1
Feb 29 1996	Apr 12 1996	-1 *(Leap year)*
NULL	Jan 6 1997	NULL
Jan 13 1997	NULL	NULL

DATE_DIFF

Retorna o período de tempo entre duas datas. Você pode solicitar que o formato seja anos, meses, dias, horas, minutos, segundos, milissegundos, microssegundos ou nanossegundos. O Data Integration Service subtrai a segunda data da primeira e retorna a diferença.

O Data Integration Service calcula a função DATE_DIFF com base no número de meses, em vez do número de dias. Ele calcula as diferenças de data para meses parciais com os dias selecionados em cada mês. Para calcular a diferença de data para o mês parcial, o Data Integration Service adiciona os dias usados no mês. Em seguida, ele divide o valor com o número total de dias no mês selecionado.

O Data Integration Service fornece um valor diferente para o mesmo período no ano bissexto e no ano não bissexto. A diferença ocorre quando fevereiro faz parte da função DATE_DIFF. O DATE_DIFF divide os dias com 29 de fevereiro para um ano bissexto e 28 se não for um ano bissexto.

Por exemplo, você deseja calcular o número de meses de 13 de setembro a 19 de fevereiro. Em um período de ano bissexto, a função DATE_DIFF calcula o mês de fevereiro como 19/29 meses ou 0.655 meses. Em um período de ano não bissexto, a função DATE_DIFF calcula o mês de fevereiro como 19/28 meses ou 0.678 meses. O Data Integration Service da mesma forma calcula a diferença em datas para o restante dos meses e a função DATE _ DIFF retorna o valor totalizado para o período especificado.

Nota: Alguns bancos de dados podem usar um algoritmo diferente para calcular a diferença em datas.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
data1	Requerido	Tipo de dados Data/Hora. Passa os valores para a primeira data que você deseja comparar. Você pode inserir qualquer expressão de transformação válida.
data2	Requerido	Tipo de dados Data/Hora. Passa os valores para a segunda data que você deseja comparar. Você pode inserir qualquer expressão de transformação válida.
formato	Requerido	String de formato que especifica a medida de data ou hora. Você pode especificar anos, meses, dias, horas, minutos, segundos, milissegundos, microssegundos ou nanossegundos. Você pode especificar somente uma parte da data, como 'mm'. Coloque as strings de formato entre aspas simples. A string de formato não faz distinção entre maiúsculas e minúsculas. Por exemplo, a string de formato 'mm' é a mesma que 'MM', 'Mm' ou 'mM'.

Valor de Retorno

Valor duplo. Se *data1* for posterior a *data2*, o valor retornado será um número positivo. Se *data1* for anterior a *data2*, o valor retornado será um número negativo.

0 se as datas forem iguais.

NULL se um (ou ambos) dos valores de data for NULL.

Exemplos

As seguintes expressões retornam o número de horas entre as portas DATE_PROMISED e 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

As expressões a seguir retornam o número de dias entre as portas DATE_PROMISED e 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

As seguintes expressões retornam o número de meses entre as portas DATE_PROMISED e 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

As seguintes expressões retornam o número de anos entre as portas DATE_PROMISED e 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
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

As seguintes expressões retornam o número de meses entre as portas DATE_PROMISED e 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 um valor codificado de base 64 e retorna uma string com a representação de dados binários dos dados. Se você codificar dados usando ENC_BASE64 e quiser decodificá-los usando DEC_BASE64, deverá executar o mapeamento usando o mesmo modo de movimento de dados. Caso contrário, a saída dos dados decodificados pode ser diferente da dos dados originais.

Sintaxe

```
DEC_BASE64( value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Obrigatório/ Opcional	Descrição
valor	Obrigatório	Tipo de dados de string. Dados que você deseja decodificar.

Valor de Retorno

Valor binário decodificado.

NULL se a entrada for um valor nulo.

Os valores de retorno serão diferentes se o mapeamento for executado no modo Unicode, e não no modo ASCII.

DECODE

Pesquisa uma porta para um valor que você especifica. Se a função localizar o valor, ela retornará um valor de resultados, que é definido por você. Você pode criar um número ilimitado de pesquisas dentro de uma função DECODE.

Se usar DECODE para pesquisar um valor em uma porta de string, você poderá cortar os espaços em branco à direita com a função RTRIM ou incluir os espaços em branco na string de pesquisa.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
valor	Requerido	Qualquer tipo de dados, exceto Binário. Passa os valores que você deseja pesquisar. Você pode inserir qualquer expressão de transformação válida.
pesquisar	Requerido	Qualquer valor com o mesmo tipo de dados que o argumento de valor. Passa os valores que você deseja pesquisar. O valor da pesquisa deve corresponder ao argumento de valor. Você não pode pesquisar uma parte de um valor. Além disso, o valor da pesquisa faz distinção entre maiúsculas e minúsculas. Por exemplo, se você quiser pesquisar a string 'Halogen Flashlight' em uma porta específica, será necessário inserir 'Halogen Flashlight, não apenas 'Halogen'. Se você inserir 'Halogen', a pesquisa não encontrará um valor correspondente. Você pode inserir qualquer expressão de transformação válida.
resultado	Requerido	Qualquer tipo de dados, exceto Binário. O valor a ser retornado se a pesquisa encontrar um valor correspondente. Você pode inserir qualquer expressão de transformação válida.
padrão	Opcional	Qualquer tipo de dados, exceto Binário. O valor a ser retornado se a pesquisa não encontrar um valor correspondente. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

First_result se a pesquisa encontrar um valor correspondente.

Valor padrão se a pesquisa não encontrar um valor correspondente.

NULL se você omitir o argumento padrão e a pesquisa não encontrar um valor correspondente.

Mesmo se várias condições forem atendidas, o Data Integration Service retornará o primeiro resultado correspondente.

Se os dados contiverem caracteres multibyte e a expressão DECODE comparar dados de string, o valor retornado dependerá da página de código e do modo de movimento de dados do Data Integration Service.

DECODE e Tipos de Dados

Quando você usar DECODE, o tipo de dados do valor retornado será sempre o mesmo tipo de dados do resultado com a maior precisão.

Por exemplo, você tem a seguinte expressão:

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

Os valores retornados nessa expressão são 5, 1,414213562 e 3,141592654. O primeiro resultado é um Inteiro e os outros resultados são Decimais. O tipo de dados Decimal tem uma precisão maior que Integer. Essa expressão sempre grava o resultado como um Decimal.

Quando você executa um mapeamento no modo de alta precisão, se pelo menos um resultado for Double, o tipo de dados do valor retornado será Double.

Não é possível criar uma função DECODE com os valores retornados numéricos e de string.

Por exemplo, a seguinte expressão é inválida:

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

Ao validar a expressão acima, você recebe esta mensagem de erro:

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

Exemplos

Você pode usar DECODE em uma expressão que pesquisa um ITEM_ID específico e retorna o 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 retorna o valor padrão de NONE nos itens 17 e 25 porque os valores de pesquisa não corresponderam ao ITEM_ID. Além disso, DECODE retorna NONE para o ITEM_ID NULL.

A seguinte expressão testa várias colunas e condições, avaliadas na ordem de cima para baixo de TRUE ou 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!

Var1	Var2	RETURN VALUE
24	27	Variables were out of desired ranges.
25	50	Variable 2 was more than 30.

DECOMPRESS

Descompacta dados usando o algoritmo de compactação zlib 1.2.1. Use a função DECOMPRESS em dados compactados com a função COMPRESS ou uma ferramenta de compactação que use o algoritmo zlib 1.2.1. Se o mapeamento que descompacta os dados usar um modo de movimento de dados diferente da sessão que os compactou, a saída dos dados descompactados poderá ser diferente da dos originais.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
valor	Requerido	Tipo de dados Binário. Dados que você deseja descompactar.
precisão	Opcional	Tipo de dados Inteiro.

Valor de Retorno

Valor binário descompactado do valor de entrada.

NULL se a entrada for um valor nulo.

ENC_BASE64

Codifica dados ao converter dados binários em dados de string usando a codificação MIME. Codifica dados quando você deseja armazenar dados em um banco de dados ou arquivo que não permite dados binários. Você também pode codificar dados para passar dados binários por transformações em formato de string. Os dados codificados são aproximadamente 33% maiores que os dados originais. Eles são exibidos como um conjunto de caracteres aleatórios.

Sintaxe

```
ENC_BASE64( value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
valor	Requerido	Tipo de dados Binário ou String. Dados que você deseja codificar.

Valor de Retorno

Valor codificado.

NULL se a entrada for um valor nulo.

ERROR

Faz com que o Data Integration Service ignore uma linha e emita uma mensagem de erro definida por você. A mensagem de erro é exibida no log. O Data Integration Service não grava essas linhas ignoradas no arquivo rejeitado.

Use ERROR nas transformações de Expressão para validar dados. Geralmente, você usa ERROR com uma função IIF ou DECODE a fim de definir regras para ignorar linhas.

Use a função ERROR para os valores padrão da porta de entrada e de saída. Você pode usar ERROR em portas de entrada para impedir que valores nulos passem em uma transformação.

Use ERROR em portas de saída para tratar qualquer tipo de erro de transformação, incluindo as chamadas da função ERROR em uma expressão. Quando você usa a função ERROR em uma expressão e no valor padrão da porta de saída, o Data Integration Service ignora a linha e registra as mensagens de erro da expressão e do valor padrão. Para ter certeza de que o Data Integration Service ignorará as linhas que geram erro, atribua ERROR como valor padrão.

Se você usar um valor padrão de saída diferente de ERROR, o valor padrão substituirá a função ERROR em uma expressão. Por exemplo, use a função ERROR em uma expressão e atribua o valor padrão, '1234', à porta de saída. Sempre que encontra a função ERROR na expressão, o Data Integration Service substitui o erro pelo valor '1234' e passa '1234' à transformação seguinte. Ele não ignora a linha e não registra um erro no log.

Sintaxe

```
ERROR( string )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
string	Requerido	Valor de string. A mensagem que você deseja exibir quando o Serviço de Integração ignora uma linha com base na expressão que contém a função ERROR. A string pode ter qualquer tamanho.

Valor de Retorno

String.

Exemplo

O exemplo a seguir mostra como consultar um mapeamento que calcula o salário médio de funcionários em todos os departamentos da organização, mas ignorar valores negativos. A seguinte expressão aninha a função ERROR em uma expressão IIF, de modo que se encontrar um salário negativo na porta Salário, o Data Integration Service ignorará a linha e exibirá um erro:

```
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

Retorna e elevado à potência especificada (expoente), onde e=2,71828183. Por exemplo, EXP(2) retorna 7,38905609893065. Você pode usar essa função para analisar dados científicos e técnicos, em vez de dados comerciais. EXP é o oposto da função LN, que retorna o logaritmo natural de um valor numérico.

Sintaxe

```
EXP( exponent )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
expoente	Requerido	Tipo de dados Numérico. O valor ao qual você deseja elevar e. O expoente na equação e^*valor*. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Valor duplo.

NULL se um valor NULL for passado como um argumento para a função.

Exemplo

A seguinte expressão usa os valores armazenados na porta Números como o valor do expoente:

```
EXP( NUMBERS )
```

NUMBERS	RETURN VALUE
10	22026.4657948067

NUMBERS	RETURN VALUE
-2	0.135335283236613
8.55	5166.754427176
NULL	NULL

FIRST

Retorna o primeiro valor encontrado em uma porta ou grupo. Você também pode aplicar um filtro para limitar as linhas lidas pelo Data Integration Service. Você pode aninhar somente uma função de agregação dentro de FIRST.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
valor	Requerido	Qualquer tipo de dados, exceto Binário. Passa os valores para os quais você deseja retornar o primeiro valor. Você pode inserir qualquer expressão de transformação válida.
filter_condition	Opcional	Limita as linhas na pesquisa. A condição de filtro deve ser um valor numérico ou avaliada como TRUE, FALSE ou NULL. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Primeiro valor em um grupo.

NULL se todos os valores passados para a função forem NULL ou se nenhuma linha for selecionada (por exemplo, a condição de filtro avaliada como FALSE ou NULL em todas as linhas).

Nulls

Se um valor for NULL, FIRST ignorará a linha. No entanto, se todos os valores passados da porta forem NULL, FIRST retornará NULL.

Agrupar por

FIRST agrupa valores baseados em grupos por portas que você define na transformação, retornando um resultado para cada grupo.

Se não houver grupo por porta, FIRST tratará todas as linhas como um grupo, retornando um valor.

Exemplos

A seguinte expressão retorna a primeira linha na porta ITEM_NAME com um preço maior que US$ 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

A seguinte expressão retorna a primeira linha na porta ITEM_NAME com um preço maior que US$ 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
Flashlight	31.00

RETURN VALUE: Regulator System

FLOOR

Retorna o maior inteiro menor ou igual ao valor numérico que você passa para essa função. Por exemplo, se você passar 3,14 para FLOOR, a função retornará 3. Se você passar 3,98 para FLOOR, a função retornará 3. Da mesma forma, se você passar -3,17 para FLOOR, a função retornará -4.

Sintaxe

```
FLOOR( numeric_value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
valor_numérico	Requerido	Tipo de dados Numérico. Você pode inserir qualquer expressão de transformação válida desde que ela seja avaliada como dados numéricos.

Valor de Retorno

Integer se você passar um valor numérico com precisão declarada entre 0 e 28.

Double se você passar um valor numérico com precisão declarada maior que 28.

NULL se um valor transmitido para a função for NULL.

Exemplo

A seguinte expressão retorna o maior número inteiro menor ou igual aos valores na porta PRICE:

```
FLOOR( PRICE )
```

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

Sugestão: Você pode realizar operação aritmética nos valores que passa para FLOOR. Por exemplo, para multiplicar um valor numérico por 10 e calcular o maior inteiro que é menor que o produto, você poderá gravar a função da seguinte forma:

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

FV

Retorna o valor futuro de um investimento, em que você faz pagamentos periódicos e constantes e o investimento rende uma taxa constante de juros.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
taxa	Requerido	Numérico. Taxa de juros ganhos em cada período. Expresso como um número decimal. Divida a taxa percentual por 100 para expressá-la como um número decimal. Deve ser maior ou igual a 0.
prazo	Requerido	Numérico. Número de períodos ou pagamentos. Deve ser maior que 0.
pagamento	Requerido	Numérico. Montante de pagamento devido por período. Deve ser um número negativo.
valor atual	Opcional	Numérico. Valor atual do investimento. Se você omitir este argumento, FV usará 0.
tipo	Opcional	Número inteiro. Momento do pagamento. Insira 1 se o pagamento for no início do período. Insira 0 se o pagamento for no final do período. O padrão é 0. Se você inserir um valor diferente de 0 ou 1, o Data Integration Service tratará o valor como 1.

Valor de Retorno

Numérico.

Exemplo

Você deposita US$ 2.000 em uma conta que rende 9% ao ano de juros mensais compostos (juros mensais de 9%/12 ou 0,75%). Você planeja depositar US$ 250 no início de cada mês pelos próximos 12 meses. A seguinte expressão retornará US$ 5.337,96 como o saldo da conta no final de 12 meses:

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

Observações

Para calcular a taxa de juros ganhos em cada período, divida a taxa anual pelo número de pagamentos feitos em um ano. O valor do pagamento e o valor atual são negativos porque indicam montantes pagos.

GET_DATE_PART

Retorna a parte especificada de uma data como um valor inteiro. Por isso, se você criar uma expressão que retorna a parte do mês da data e passa uma data, como Apr 1 1997 00:00:00, GET_DATE_PART retornará 4.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
data	Requerido	Tipo de dados Data/Hora. Você pode inserir qualquer expressão de transformação válida.
formato	Requerido	Uma string de formato que especifica a parte do valor de data que você deseja retornar. Coloque as strings de formato entre aspas simples, por exemplo, 'mm'. A string de formato não faz distinção entre maiúsculas e minúsculas. Cada string de formato retorna a data completa com base no formato de data especificado no mapeamento. Por exemplo, se você passar a data Apr 1 1997 to GET_DATE_PART, as strings de formato 'Y', 'YY', 'YYY' ou 'YYYY' retornarão 1997.

Valor de Retorno

Inteiro que representa a parte especificada da data.

NULL se um valor transmitido para a função for NULL.

Exemplos

As seguintes expressões retornam a hora de cada data na porta DATE_SHIPPED. 12:00:00AM retorna 0 porque o formato de data padrão é baseado no 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

As seguintes expressões retornam o dia de cada data na porta 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

DATE_SHIPPED	RETURN VALUE
Aug 22 1997 12:00:00PM	22
NULL	NULL

As seguintes expressões retornam o mês de cada data na porta 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

A seguinte expressão retorna o ano de cada data na porta 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

Retorna o valor de fuso horário de determinada coluna de registro de data/hora com fuso horário.

Por exemplo:

```
String TimeZone = GET_TIMEZONE (timestampWithTZ);
```

A porta de saída deve ser do tipo de dados de String para expressões GET_TIMEZONE.

Sintaxe

```
GET_TIMEZONE (timestamp_with_timezone_value)
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
timestamp_with_timezone_value	Obrigatório	Deve ser um tipo de dados de registro de data/hora com fuso horário. Você pode inserir qualquer expressão válida de transformação.

Valor de Retorno

Retorna um tipo de dados de String que contém o nome da região de fuso horário ou a diferença de fuso horário.

NULL se a entrada for um valor nulo.

Exemplo

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

Retorna o valor de data e hora para um tipo de entrada timestampWithTZ. A data e hora retornadas estarão no fuso horário solicitado, que pode ser apresentado como o segundo argumento. Se o valor de fuso horário não for especificado no segundo argumento, a função retornará a parte do registro de data/hora do valor de entrada timestampWithTZ.

Por exemplo:

```
GET_TIMESTAMP(Registro de data/hora com Fuso Horário,“+08:30”)
```

O primeiro argumento, registro de data/hora com fuso horário, tem (+05:30) como o valor de fuso horário. A função retorna o registro de data/hora no fuso horário especificado como o segundo argumento, (+08:30).

A porta de saída deve ser data e hora para expressões GET_TIMESTAMP.

Sintaxe

```
GET_TIMESTAMP (timestamp_with_timezone_value, [timezone_value])
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
timestamp_with_timezone_value	Obrigatório	Deve ser um tipo de dados de registro de data/hora com fuso horário. Você pode inserir qualquer expressão válida de transformação.
timezone_value	Opcional	Deve ser um tipo de dados de string. A string deve ser uma string de caracteres. Transmite os valores que você deseja exibir para o fuso horário que será usado como base para a função retornar o registro de data/hora. Você pode inserir qualquer expressão válida de transformação. Se você não especificar o fuso horário, a função retornará a parte do registro de data/hora do primeiro argumento.

Valor de Retorno

Retorna o valor de registro de data/hora na diferença de fuso horário ou região especificada.

Se o valor de fuso horário não for transmitido, a função retornará a parte do registro de data/hora do primeiro argumento.

NULL se a entrada for um valor nulo.

Exemplo

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

Retorna o maior valor de uma lista de valores de entrada. Use essa função para retornar a maior string, data ou número. Por padrão, a correspondência faz distinção entre maiúsculas e minúsculas.

Sintaxe

```
GREATEST( value1, [value2, ..., valueN,] CaseFlag )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
valor	Obrigatório	Qualquer tipo de dados, exceto Binário. O tipo de dados deve ser compatível com outros valores. O valor que você deseja comparar com outros valores. Você deve inserir pelo menos um argumento de valor. Se o valor for numérico, e outros valores de entrada forem numéricos, todos os valores usarão a maior precisão possível. Por exemplo, se alguns valores forem do tipo de dados Número Inteiro, e outros forem do tipo de dados Duplo, o Data Integration Service os converterá em Duplo.
CaseFlag	Opcional	Deve ser um número inteiro. Especifique um valor quando o argumento de valor de entrada for um valor de string. Determina se os argumentos nessa função fazem distinção entre maiúsculas e minúsculas. Você pode inserir qualquer expressão válida de transformação. Quando CaseFlag é um número diferente de 0, a função faz distinção entre maiúsculas e minúsculas. Quando CaseFlag é 0, a função não faz distinção entre maiúsculas e minúsculas. O padrão faz distinção entre maiúsculas e minúsculas.

Valor de Retorno

valor1 se for o maior dos valores de entrada, *valor2* se for o maior dos valores de entrada, e assim por diante.

NULL se um dos argumentos for NULL.

Exemplo

A seguinte expressão retorna a maior quantidade de itens ordenados:

```
GREATEST( QUANTITY1, QUANTITY2, QUANTITY3 )
```

QUANTITIY1	QUANTITY2	QUANTITY3	RETURN VALUE
150	756	27	756
			NULL
5000	97	17	5000
120	1724	965	1724

IIF

Retorna um dos dois valores especificados, com base nos resultados de uma condição.

Sintaxe

```
IIF( condition, value1 [,value2] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
condição	Obrigatório	A condição que você deseja avaliar. Você pode inserir qualquer expressão de transformação válida que avalie em TRUE ou FALSE.
value1	Obrigatório	Qualquer tipo de dados, exceto Binário. O valor a ser retornado se a condição for TRUE. O valor retornado é sempre o tipo de dados especificado por esse argumento. Você pode inserir qualquer expressão de transformação válida, incluindo outra expressão IIF.
value2	Opcional	Qualquer tipo de dados, exceto Binário. O valor a ser retornado se a condição for FALSE. Você pode inserir qualquer expressão de transformação válida, incluindo outra expressão IIF.

Diferente das funções condicionais em alguns sistemas, a condição FALSE (*valor2*) na função IIF não é requerida. Se você omitir *valor2*, a função retornará o seguinte quando a condição for FALSE:

- 0 se *valor1* for um tipo de dados Numérico.
- String vazia se *valor1* for um tipo de dados String.
- NULL se *valor1* for um tipo de dados Data/Hora.

Por exemplo, a expressão a seguir não inclui uma condição FALSE e *value1* é um tipo de dados de string, de modo que o Data Integration Service retorna uma string vazia em cada linha avaliada como 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 Retorno

valor1 se a condição for TRUE.

valor2 se a condição for FALSE.

Por exemplo, a expressão a seguir inclui a condição FALSE NULL, de modo que o Data Integration Service retorna NULL em cada linha avaliada como 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

Se os dados contiverem caracteres multibyte e o argumento da condição comparar dados de string, o valor retornado dependerá da página de código e do modo de movimento de dados do Data Integration Service.

IIF e tipos de dados

Quando você usar IIF, o tipo de dados do valor retornado será o mesmo tipo de dados do resultado com a maior precisão.

Por exemplo, você tem a seguinte expressão:

```
IIF( SALES < 100, 1, .3333 )
```

O resultado TRUE (1) é um inteiro e o resultado FALSE (.3333) é um decimal. O tipo de dados Decimal tem precisão maior que Integer, portanto o tipo de dados do valor retornado é sempre um Decimal.

Se você executar uma sessão no modo de alta precisão e pelo menos um resultado for Double, o tipo de dados do valor retornado será Double.

Tipos de dados complexos e IIF

Você pode usar IIF para retornar uma matriz ou uma estrutura, ou elementos da matriz ou estrutura.

Por exemplo, você tem a seguinte matriz:

```
names = ['John', 'Kevin', 'Laura']
```

Você pode usar a seguinte expressão para retornar um dos valores na matriz:

```
IIF( SIZE(names) > 2, names[2], names[0] )
```

RETURN VALUE: 'Laura'

Usos Especiais de IIF

Use declarações IIF aninhadas para testar várias condições. O seguinte exemplo testará várias condições e retornará 0 se as vendas forem 0 ou negativas:

```
IIF( SALES > 0, IIF( SALES < 50, SALARY1, IIF( SALES < 100, SALARY2, IIF( SALES < 200,
SALARY3, BONUS))), 0 )
```

Você pode tornar essa lógica mais legível ao adicionar comentários:

```
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)
```

Use IIF em estratégias de atualização. Por exemplo:

```
IIF( ISNULL( ITEM_NAME ), DD_REJECT, DD_INSERT)
```

Alternativa para IIF

Use "DECODE" na página 88, em vez de IIF em muitos casos. DECODE pode melhorar a legibilidade. O seguinte exemplo mostra como usar DECODE, em vez de IIF, usando o primeiro exemplo da seção anterior:

```
DECODE( TRUE,
   SALES > 0 and SALES < 50, SALARY1,
   SALES > 49 AND SALES < 100, SALARY2,
   SALES > 99 AND SALES < 200, SALARY3,
   SALES > 199, BONUS)
```

Geralmente se pode usar uma transformação de Filtro em vez de IIF, para maximizar o desempenho.

IN

Faz correspondência dos dados de entrada com uma lista de valores. Por padrão, a correspondência faz distinção entre maiúsculas e minúsculas.

Sintaxe

```
IN( valueToSearch, value1, [value2, ..., valueN,] CaseFlag )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
valueToSearch	Obrigatório	Pode ser uma string, uma data ou um valor numérico. O valor de entrada que você deseja corresponder a uma lista de valores separada por vírgula.
valor	Obrigatório	Pode ser uma string, uma data ou um valor numérico, dependendo do tipo especificado para o argumento valueToSearch. Lista de valores separada por vírgula que deseja pesquisar. Os valores podem ser portas em uma transformação. Não há um número máximo de valores que você pode listar.
CaseFlag	Opcional	Deve ser um número inteiro. Especifique um valor quando o argumento valueToSearch for um valor de string. Determina se os argumentos nessa função fazem distinção entre maiúsculas e minúsculas. Você pode inserir qualquer expressão válida de transformação. Quando CaseFlag é um número diferente de 0, a função faz distinção entre maiúsculas e minúsculas. Quando CaseFlag é 0, a função não faz distinção entre maiúsculas e minúsculas. O padrão faz distinção entre maiúsculas e minúsculas.

Valor de Retorno

TRUE (1) se o valor de entrada corresponder à lista de valores.

FALSE (0) se o valor de entrada não corresponder à lista de valores.

NULL se a entrada for um valor nulo.

Exemplo

A expressão a seguir determina se o valor de entrada é uma faca de segurança, um cinzel ou uma faca de titânio média. Não é necessário que os valores de entrada tenham o mesmo padrão de maiúsculas e minúsculas que os valores da lista separada por vírgula:

```
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

Encontra o índice de um valor entre uma lista de valores. Por padrão, a correspondência faz distinção entre maiúsculas e minúsculas.

Sintaxe

```
INDEXOF( valueToSearch, string1 [, string2, ..., stringN,] [CaseFlag] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
valueToSearch	Obrigatório	Tipo de dados de string. Valor que você deseja pesquisar na lista de strings.
string	Obrigatório	Tipo de dados de string. Lista separada por vírgula de valores nos quais você deseja pesquisar. Os valores podem estar no formato de string. Não há um número máximo de valores que você pode listar. O valor faz distinção entre maiúsculas e minúsculas, a menos que você defina CaseFlag como 0.
CaseFlag	Opcional	Deve ser um número inteiro. Especifique um valor quando o argumento valueToSearch for um valor de string. Determina se os argumentos nessa função fazem distinção entre maiúsculas e minúsculas. Você pode inserir qualquer expressão válida de transformação. Quando CaseFlag é um número diferente de 0, a função faz distinção entre maiúsculas e minúsculas. Quando CaseFlag é 0, a função não faz distinção entre maiúsculas e minúsculas.

Valor de Retorno

1 se o valor de entrada corresponder a *string1*, 2 se o valor de entrada corresponder a *string2* e assim por diante.

0 se o valor de entrada não for encontrado.

NULL se a entrada for um valor nulo.

Exemplo

A seguinte expressão determina se os valores da porta ITEM_NAME correspondem à primeira, segunda ou terceira string:

```
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

Faca de Segurança retorna um valor 0 porque ela não corresponde à distinção entre maiúsculas e minúsculas do valor de entrada.

INITCAP

Coloca em maiúscula a primeira letra de cada palavra de uma string e converte todas as outras letras em minúscula. As palavras são delimitadas por espaços em branco (um espaço em branco, avanço de página, nova linha, retorno de carro, tabulação ou tabulação vertical) e caracteres que não são alfanuméricos. Por exemplo, se você passar a string '…THOMAS', a função retornará Thomas.

Sintaxe

```
INITCAP( string )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Obrigatório / Opcional	Descrição
string	Obrigatório	Qualquer tipo de dados, exceto Binário. Você pode inserir qualquer expressão válida de transformação.

Valor de Retorno

String. Se os dados contiverem caracteres multibyte, o valor retornado dependerá da página de código e do modo de movimento de dados do Data Integration Service.

NULL se um valor transmitido para a função for NULL.

Exemplo

A expressão a seguir coloca em letra maiúscula todos os nomes contidos na porta FIRST_NAME.

```
INITCAP( FIRST_NAME )
```

FIRST_NAME	RETURN VALUE
ramona	Ramona
18-albert	18-Albert
NULL	NULL
?!SAM	?!Sam
THOMAS	Thomas
PierRe	Pierre

INSTR

Retorna a posição de um conjunto de caracteres em uma string, contando da esquerda para a direita.

Sintaxe

```
INSTR( string, search_value [,start [,occurrence [,comparison_type ]]] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório/ Opcional	Descrição
string	Requerido	A string deve ser uma string de caracteres. Passa o valor que você deseja avaliar. Você pode inserir qualquer expressão de transformação válida. Os resultados da expressão deve ser uma string de caracteres. Se não, INSTR converte o valor em uma string antes de avaliá-la.
search_value	Requerido	Qualquer valor. O valor pesquisado faz distinção entre maiúsculas e minúsculas. O conjunto de caracteres que você deseja pesquisar. O valor_pesquisado deve corresponder a uma parte da string. Por exemplo, se você gravar `INSTR('Alfred Pope', 'Alfred Smith')` a função retornará 0. Você pode inserir qualquer expressão de transformação válida. Se você quiser pesquisar uma string de caractere, coloque os caracteres que deseja pesquisar em aspas simples, por exemplo, 'abc'.

Argumento	Obrigatório/ Opcional	Descrição
início	Opcional	Deve ser um valor inteiro. A posição na string onde você deseja começar a pesquisa. Você pode inserir qualquer expressão de transformação válida. O padrão é 1, ou seja, INSTR começa a pesquisa no primeiro caractere da string. Se a posição inicial for 0, INSTR pesquisará a partir do primeiro caractere da string. Se a posição inicial for um número positivo, o INSTR localizará a posição inicial contando a partir do começo da strings. Se a posição inicial for um número negativo, o INSTR localizará a posição inicial contando a partir do final da string. Se você omitir esse argumento, a função usará o valor padrão de 1.
ocorrência	Opcional	Um número inteiro positivo maior que 0. Você pode inserir qualquer expressão de transformação válida. Se o valor pesquisado aparecer mais de uma vez na string, você poderá especificar qual ocorrência deseja pesquisar. Por exemplo, digite 2 para pesquisar a segunda ocorrência a partir da posição inicial. Se você omitir esse argumento, a função usará o valor padrão 1, ou seja, INSTR pesquisará a primeira ocorrência do valor pesquisado. Se você passar um decimal, o Data Integration Service o arredondará para o valor inteiro mais próximo. Se você passar um inteiro negativo ou 0, a sessão falhará.
comparison_type	Opcional	O tipo de comparação de string pode ser linguístico ou binário quando o Data Integration Service é executado no modo Unicode. Quando o Data Integration Service é executado no modo ASCII, o tipo de comparação é sempre binário. As comparações linguísticas consideram regras de colação específicas ao idioma, enquanto que as comparações binárias executam correspondência de bit a bit. Por exemplo, o caractere sharp s em alemão corresponde à string “ss” em uma comparação linguística, mas não em uma comparação binária. As comparações binárias são executadas mais rapidamente do que as comparações linguísticas. Deve ser um valor inteiro, 0 ou 1. - 0: INSTR executa uma comparação de strings linguística. - 1: INSTR executa uma comparação de strings binária. O padrão é 0.

Valor de Retorno

Inteiro se a pesquisa tiver êxito. Inteiro representa a posição do primeiro caractere no *valor_pesquisado*, contando da esquerda para a direita.

0 se a pesquisa não tiver êxito.

NULL se um valor transmitido para a função for NULL.

Exemplos

A seguinte expressão retorna a posição da primeira ocorrência da letra 'a', começando do início de cada nome de empresa. Como o argumento *search_value* faz distinção entre maiúsculas e minúsculas, ele ignora o ‘A’ em ‘Blue Fin Aqua Center’ e retorna a posição para o ‘a’ em ‘Aqua’:

```
INSTR( COMPANY, 'a' )
```

COMPANY	RETURN VALUE
Blue Fin Aqua Center	13
Maco Shark Shop	2

COMPANY	RETURN VALUE
Scuba Gear	5
Frank's Dive Shop	3
VIP Diving Club	0

A seguinte expressão retorna a posição da segunda ocorrência da letra 'a', começando do início de cada nome de empresa. Como o argumento *search_value* faz distinção entre maiúsculas e minúsculas, ele ignora o ‘A’ em ‘Blue Fin Aqua Center’ e retorna 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

A expressão a seguir retorna a posição da segunda ocorrência da letra ‘a’ em cada nome de empresa, começando a partir do último caractere no nome de empresa. Como o argumento *search_value* faz distinção entre maiúsculas e minúsculas, ele ignora o ‘A’ em ‘Blue Fin Aqua Center’ e retorna 0:

```
INSTR( COMPANY, 'a', -1, 2 )
```

COMPANY	RETURN VALUE
Blue Fin Aqua Center	0
Maco Shark Shop	2
Scuba Gear	5
Frank's Dive Shop	0
VIP Diving Club	0

A seguinte expressão retorna a posição do primeiro caractere na string ‘Blue Fin Aqua Center’ (começando a partir do último caractere no nome da empresa):

```
INSTR( COMPANY, 'Blue Fin Aqua Center', -1, 1 )
```

COMPANY	RETURN VALUE
Blue Fin Aqua Center	1
Maco Shark Shop	0

COMPANY	RETURN VALUE
Scuba Gear	0
Frank's Dive Shop	0
VIP Diving Club	0

Usando INSTR aninhado

Você pode aninhar a função INSTR em outras funções para realizar tarefas mais complexas.

A expressão a seguir avalia uma string, começando a partir do final da string. A expressão encontra o último espaço (extrema direita) na string e, em seguida, retorna todos os caracteres à sua esquerda:

```
SUBSTR( CUST_NAME,1,INSTR( CUST_NAME,' ' ,-1,1 ))
```

CUST_NAME	RETURN VALUE
PATRICIA JONES	PATRICIA
MARY ELLEN SHAH	MARY ELLEN

A seguinte expressão remove o caractere '#' de uma string:

```
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

Retorna se um valor é NULL. ISNULL avalia uma cadeia vazia como FALSE.

Nota: Para testar strings vazias, use LENGTH.

Sintaxe

```
ISNULL( value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Obrigatório / Opcional	Descrição
valor	Obrigatório	Qualquer tipo de dados, exceto Binário. Passa as linhas que você deseja avaliar. Você pode inserir qualquer expressão válida de transformação.

Valor de Retorno

TRUE (1) se o valor for NULL.

FALSE (0) se o valor não for NULL.

Exemplo

O seguinte exemplo verifica valores nulos na tabela de itens:

```
ISNULL( ITEM_NAME )
```

ITEM_NAME	RETURN VALUE
Flashlight	0 (FALSE)
NULL	1 (TRUE)
Regulator system	0 (FALSE)
''	0 (FALSE) *Empty string is not NULL*

Tipos de dados complexos e ISNULL

Você pode usar o ISNULL para verificar se uma matriz ou uma estrutura tem um valor nulo.

As expressões a seguir verificam os valores nulos nos seguintes tipos de dados complexos:

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

Retorna se um valor de string é uma data válida. Uma data válida é qualquer string na parte de data do formato de data e hora especificado na sessão. Se a string que você deseja testar não estiver nesse formato de data, use a string de formato TO_DATE para especificar o formato de data. Se as strings passadas para IS_DATE não corresponderem à string de formato especificada, a função retornará FALSE (0). Se as strings corresponderem à string de formato, a função retornará TRUE (1).

IS_DATE avalia as strings e retorna um valor inteiro.

A porta de saída de uma expressão IS_DATE deve ser um tipo de dados Numérico ou de String.

Você pode usar IS_DATE para testar ou filtrar dados em um arquivo plano antes de gravá-lo em um destino.

Use a string de formato RR com IS_DATE, em vez da string de formato YY. Na maioria dos casos, as duas strings de formato retornam os mesmos valores, mas há casos exclusivos em que YY retorna resultados incorretos. Por exemplo,a expressão IS_DATE('02/29/00', 'YY') é computada internamente como IS_DATE(02/29/1900 00:00:00), que retorna false. No entanto, o Data Integration Service computa a expressão IS_DATE('02/29/00', 'RR') como IS_DATE(02/29/2000 00:00:00), que retorna TRUE. No primeiro caso, o ano 1900 não é um ano bissexto, portanto não existe a data de 29 de fevereiro.

Nota: IS_DATE usa as mesmas strings de formato que TO_DATE.

Sintaxe

```
IS_DATE( value [,format] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
valor	Requerido	Deve ser um tipo de dados de string. Passa as linhas que você deseja avaliar. Você pode inserir qualquer expressão de transformação válida.
formato	Opcional	Insira uma string de formato TO_DATE válida. A string de formato deve corresponder às partes do argumento *string*. Por exemplo, se você passar a string 'Mar 15 1997 12:43:10AM', será necessário usar a string de formato 'MON DD YYYY HH12:MI:SSAM'. Se você omitir a string de formato, o valor da string deverá estar no formato de data especificado na configuração do mapeamento.

Valor de Retorno

TRUE (1) se a linha for um número válido.

FALSE (0) se a linha não for uma data válida.

NULL se um valor na expressão for NULL ou se a string de formato for NULL.

Aviso: O formato da string IS_DATE deve corresponder à string de formato, incluindo quaisquer separadores de data. Se não, o Data Integration Service talvez retorne valores imprecisos ou ignore o registro.

Exemplos

A seguinte expressão verifica as datas válidas na porta INVOICE_DATE:

```
IS_DATE( INVOICE_DATE )
```

Essa expressão retorna dados semelhantes ao seguinte:

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)

A seguinte expressão IS_DATE especifica uma string de formato igual a 'YYYY/MM/DD':

```
IS_DATE( INVOICE_DATE, 'YYYY/MM/DD' )
```

Se o valor da string não corresponder a esse formato, IS_DATE retornará 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)

O exemplo a seguir mostra como usar IS_DATE para testar dados antes de usar TO_DATE para converter as strings em datas. Essa expressão verifica os valores na porta INVOICE_DATE e converte cada data válida em um valor de data. Se o valor não for uma data válida, o Data Integration Service retornará ERROR e ignorará a linha.

Este exemplo retorna um valor de Data/Hora. Portanto, a porta de saída da expressão precisa ser Data/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

Retorna se uma string for um número válido. Um número válido consiste nas seguintes partes:

- Espaço opcional antes do número
- Sinal opcional (+/-)
- Um ou mais dígitos com um ponto decimal opcional
- Notação científica opcional, como a letra ‘e’ ou ‘E’ (e a letra ‘d’ ou ‘D’ no Windows) seguida por um sinal opcional (+/-), seguida por um ou mais dígitos
- Espaço em branco opcional que segue o número

Os seguintes números são todos válidos:

```
' 100  '
'   +100'
'-100'
'-3.45e+32'
'+3.45E-32'
'+3.45d+32' (Windows only)
'+3.45D-32' (Windows only)
'.6804'
```

A porta de saída de uma expressão IS_NUMBER deve ser um tipo de dados Numérico ou de String.

Você pode usar IS_NUMBER para testar ou filtrar dados em um arquivo plano antes de gravá-lo em um destino.

Sintaxe

```
IS_NUMBER( value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
valor	Requerido	Deve ser um tipo de dados de String. Passa as linhas que você deseja avaliar. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

TRUE (1) se a linha for um número válido.

FALSE (0) se a linha não for um número válido.

NULL se um valor na expressão for NULL.

Exemplos

A seguinte expressão verifica a porta ITEM_PRICE para obter 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) *Incomplete number*
' '	0 (False) *Consists entirely of blanks*
''	0 (False) *Empty string*
'+123abc'	0 (False)
' 123'	1 (True) *Leading white blanks*
'123 '	1 (True) *Trailing white blanks*
'ABC'	0 (False)
'-ABC'	0 (False)
NULL	NULL

Use IS_NUMBER para testar dados antes de usar uma das funções de conversão numérica, como TO_FLOAT. Por exemplo, a expressão a seguir verifica os valores na porta ITEM_PRICE e converte cada número válido em um valor de ponto flutuante de precisão dupla. Se o valor não for um número válido, o Data Integration Service retornará 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 *Consists entirely of blanks*
''	0.00 *Empty string*
'+123abc'	0.00
' 123ABC'	0.00
'ABC'	0.00

ITEM_PRICE	RETURN VALUE
'-ABC'	0.00
NULL	NULL

IS_SPACES

Retorna se um valor de string consistir totalmente em espaços. Um espaço é um espaço em branco, uma alimentação de forma, uma nova linha, um retorno de carro, uma guia ou uma guia vertical.

IS_SPACES avalia uma string vazia como FALSE porque não há espaços. Para testar uma string vazia, use LENGTH.

Sintaxe

```
IS_SPACES( value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
valor	Requerido	Deve ser um tipo de dados de string. Passa as linhas que você deseja avaliar. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

TRUE (1) se a linha consistir totalmente em espaços.

FALSE (0) se a linha contiver dados.

NULL se um valor na expressão for NULL.

Exemplo

A seguinte expressão verifica a porta ITEM_NAME em linhas que consistem totalmente em espaços:

```
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.*)

Sugestão: Use IS_SPACES para evitar a gravação de espaços em uma coluna de caracteres em uma tabela de destino. Por exemplo, se tiver uma transformação que grava nomes de cliente em uma coluna CHAR(5) de

tamanho fixo em uma tabela de destino, você poderá gravar '00000' em vez de espaços. Você criará uma expressão semelhante ao seguinte:

```
IIF( IS_SPACES( CUST_NAMES ), '00000', CUST_NAMES )
```

LAG

Retorna o valor, que é um número de deslocamento de linhas antes da linha atual em uma transformação de Expressão. Use essa função para comparar valores na linha atual com valores em uma linha anterior quando você executar um mapeamento no mecanismo Spark do ambiente Hadoop.

Um valor de atraso é exibido antes da linha atual em um conjunto de dados.

Ao usar LAG em uma transformação, você deve configurar a transformação para definição de janelas. Propriedades de definição de janelas definem como os dados são particionados e ordenados.

Sintaxe

```
LAG ( column_name, offset, default )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório/ Opcional	Descrição
column_name	Obrigatório	A coluna ou expressão de destino em que a função opera.
offset	Obrigatório	Tipo de dados Integer. O número de linhas antes da linha atual do qual obter um valor.
default	Opcional	O valor padrão a ser retornado em caso de deslocamento está fora dos limites da partição ou tabela. O padrão é NULL.

Valor de retorno

O tipo de dados do *column_name* especificado.

Padrão se o valor de retorno estiver fora dos limites da partição especificada.

NULL se *padrão* for omitido ou definido como NULL.

Exemplos

A expressão a seguir retorna a data em que o pedido anterior foi efetuado:

```
LAG ( ORDER_DATE, 1, NULL )
```

ORDER_DATE	ORDER_ID	RETURN VALUE
2017/09/25	1	NULL
2017/09/26	2	2017/09/25
2017/09/27	3	2017/09/26

ORDER_DATE	ORDER_ID	RETURN VALUE
2017/09/28	4	2017/09/27
2017/09/29	5	2017/09/28
2017/09/30	6	2017/09/29

O valor de atraso da primeira linha está fora da partição e, portanto, a função retornou o valor padrão de NULL.

No exemplo a seguir, sua organização recebe pings de GPS de veículos que incluem IDs de viagem e eventos e um registro de data/hora. Você deseja calcular a diferença de tempo entre cada ping.

A expressão a seguir calcula a diferença de tempo entre a linha atual e a linha anterior para duas viagens separadas:

```
DATE_DIFF( EVENT_TIME, LAG ( EVENT_TIME, 1, NULL ), 'ss' )
```

Você particiona os dados por ID de viagem e 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

Os valores de atraso da primeira e da quarta linhas estão fora da partição especificada e, portanto, a função retornou dois valores NULL padrão.

LAST

Retorna a última linha na porta selecionada. Você também pode aplicar um filtro para limitar as linhas lidas pelo Data Integration Service. Você pode aninhar somente uma função de agregação dentro de LAST.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
valor	Requerido	Qualquer tipo de dados, exceto Binário. Passa os valores para os quais você deseja retornar a última linha. Você pode inserir qualquer expressão de transformação válida.
filter_condition	Opcional	Limita as linhas na pesquisa. A condição de filtro deve ser um valor numérico ou avaliada como TRUE, FALSE ou NULL. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Última linha em uma porta.

NULL se todos os valores passados para a função forem NULL ou se nenhuma linha for selecionada (por exemplo, a condição de filtro avaliada como FALSE ou NULL em todas as linhas).

Exemplo

A seguinte expressão retorna a última linha na porta ITEMS_NAME com um preço maior que US$ 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

Retorna a data do último dia do mês de cada data em uma porta.

Sintaxe

```
LAST_DAY( date )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
data	Requerido	Tipo de dados Data/Hora. Passa as datas para as quais você deseja retornar o último dia do mês. Você pode inserir qualquer expressão de transformação válida que avalie em uma data.

Valor de Retorno

Data. O último dia do mês para o valor de data que você passa para essa função.

NULL se um valor na porta selecionada for NULL.

Nulo

Se um valor for NULL, LAST_DAY ignorará a linha. No entanto, se todos os valores passados da porta são NULL, LAST_DAY retornará NULL.

Agrupar por

LAST_DAY agrupa valores baseados em portas agrupar por que você define na transformação, retornando um resultado para cada grupo. Se não houver grupo por porta, LAST_DAY tratará todas as linhas como um grupo, retornando um valor.

Exemplos

A seguinte expressão retorna o último dia do mês de cada data na porta 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

Você pode aninhar TO_DATE para converter valores de string em uma data. TO_DATE sempre inclui informações de hora. Se você passar uma string que não tem um valor de hora, a data retornada incluirá a hora 00:00:00.

O seguinte exemplo retorna o último dia do mês de cada data do pedido no mesmo formato que a string:

```
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 *(Leap year)*

LEAD

Retorna o valor, que é um número de deslocamento de linhas depois da linha atual em uma transformação de Expressão. Use essa função para comparar valores na linha atual com valores em uma linha futura quando você executar um mapeamento no mecanismo Spark do ambiente Hadoop.

Um valor de líder aparece após a linha atual em um conjunto de dados.

Nota: Ao usar LEAD em uma transformação, você deve configurar a transformação para definição de janelas. Propriedades de definição de janelas definem como os dados são particionados e ordenados.

Sintaxe

```
LEAD ( column_name, offset, default )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório/ Opcional	Descrição
column_name	Obrigatório	A coluna ou expressão de destino em que a função opera.
offset	Obrigatório	Tipo de dados Integer. O número de linhas após a linha atual do qual obter um valor.
default	Opcional	O valor padrão a ser retornado em caso de deslocamento está fora dos limites da partição ou tabela. O padrão é NULL.

Valor de retorno

O tipo de dados do *column_name* especificado.

Padrão se o valor de retorno estiver fora dos limites da partição especificada.

NULL se *padrão* for omitido ou definido como NULL.

Exemplos

A expressão a seguir retorna, para cada funcionário, a data em que o próximo funcionário foi contratado:

```
LEAD ( HIRE_DATE, 1, NULL )
```

EMPLOYEE	HIRE_DATE	RETURN VALUE
Hynes	2012/12/07	2014/05/18

EMPLOYEE	HIRE_DATE	RETURN VALUE
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

Não há nenhum valor de líder disponível para a última linha e, portanto, a função retornou o valor padrão de NULL.

A expressão a seguir retorna a diferença nos valores de cotas de vendas entre o primeiro trimestre e o terceiro trimestre de dois anos civis:

```
LEAD ( Sales_Quota, 2, 0 ) - Sales_Quota
```

Você particiona os dados por ano e ordena 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

Os valores de líder do segundo e do terceiro trimestres estão fora da partição especificada e, portanto, a função retornou um valor de "0".

LEAST

Retorna o menor valor de uma lista de valores de entrada. Por padrão, a correspondência faz distinção entre maiúsculas e minúsculas.

Sintaxe

```
LEAST( value1, [value2, ..., valueN,] CaseFlag )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
valor	Obrigatório	Qualquer tipo de dados, exceto Binário. O tipo de dados deve ser compatível com outros valores. O valor que você deseja comparar com outros valores. Você deve inserir pelo menos um argumento de valor. Se o valor for Numérico, e outros valores de entrada forem de outros tipos de dados numéricos, todos os valores usarão a maior precisão possível. Por exemplo, se alguns valores forem do tipo de dados Número Inteiro, e outros forem do tipo de dados Duplo, o Data Integration Service os converterá em Duplo.
CaseFlag	Opcional	Deve ser um número inteiro. Especifique um valor quando o argumento de valor de entrada for um valor de string. Determina se os argumentos nessa função fazem distinção entre maiúsculas e minúsculas. Você pode inserir qualquer expressão válida de transformação. Quando CaseFlag é um número diferente de 0, a função faz distinção entre maiúsculas e minúsculas. Quando CaseFlag é 0, a função não faz distinção entre maiúsculas e minúsculas. O padrão faz distinção entre maiúsculas e minúsculas.

Valor de Retorno

value1 se for o maior dos valores de entrada, *value2* se for o menor dos valores de entrada, e assim por diante.

NULL se um dos argumentos for NULL.

Exemplo

A seguinte expressão retorna a menor quantidade de itens ordenados:

```
LEAST( QUANTITY1, QUANTITY2, QUANTITY3 )
```

QUANTITIY1	QUANTITY2	QUANTITY3	RETURN VALUE
150	756	27	27
			NULL
5000	97	17	17
120	1724	965	120

LENGTH

Retorna o número de caracteres em uma string, incluindo espaços em branco à direita.

Sintaxe

```
LENGTH( string )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
string	Requerido	Tipo de dados strings. As strings que você deseja avaliar. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Inteiro que representa o tamanho da string.

NULL se um valor transmitido para a função for NULL.

Exemplo

A seguinte expressão retorna o tamanho de cada nome de cliente:

```
LENGTH( CUSTOMER_NAME )
```

CUSTOMER_NAME	RETURN VALUE
Bernice Davis	13
NULL	NULL
John Baer	9
Greg Brown	10

Dicas para LENGTH

Use LENGTH para testar as condições de string vazia. Se você quiser encontrar campos nos quais o nome do cliente está vazio, use uma expressão, como:

```
IIF( LENGTH( CUSTOMER_NAME ) = 0, 'EMPTY STRING' )
```

Para testar um campo nulo, use ISNULL. Para testar espaços, use IS_SPACES.

LN

Retorna o logaritmo natural de um valor numérico. Por exemplo, `LN(3)` retorna `1.098612`. Geralmente, você usa essa função para analisar dados científicos, em vez de dados comerciais.

Essa função é o oposto da função EXP.

Sintaxe

```
LN( numeric_value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
valor_numérico	Requerido	Tipo de dados Numérico. Deve ser um número positivo, maior que 0. Passa os valores para os quais você deseja calcular o logaritmo natural. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Valor duplo.

NULL se um valor transmitido para a função for NULL.

Exemplo

A seguinte expressão retorna o logaritmo natural de todos os valores na porta 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: O Data Integration Service exibe um erro e não grava a linha quando você passa um número negativo ou 0. O *valor_numérico* deve ser um número positivo maior que 0.

LOG

Retorna o logaritmo de um valor numérico. Esta função é mais usada para a análise de dados científicos e técnicos, em vez de dados comerciais.

Sintaxe

```
LOG( base, exponent )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
base	Requerido	A base do logaritmo. Deve ser um valor numérico positivo diferente de 0 ou 1. Qualquer expressão de transformação válida avaliada como um número positivo diferente de 0 ou 1.
expoente	Requerido	O expoente do logaritmo. Deve ser um valor numérico positivo maior que 0. Qualquer expressão de transformação válida avaliada como um número positivo maior que 0.

Valor de Retorno

Valor duplo.

NULL se um valor transmitido para a função for NULL.

Exemplo

A seguinte expressão retorna o logaritmo de todos os valores na porta 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. (Data Integration Service does not write the row.)*
0	5	*Error. (Data Integration Service does not write the row.)*
10	-2	*Error. (Data Integration Service does not write the row.)*

O Data Integration Service exibirá um erro e não gravará a linha se você passar 0, 1 ou um número negativo como valor base ou um valor negativo para o expoente.

LOWER

Converte os caracteres maiúsculos de uma string para minúsculo.

Sintaxe

```
LOWER( string )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
string	Requerido	Qualquer valor de string. O argumento transmite os valores da string que você deseja retornar como letra minúscula. Você pode inserir qualquer expressão de transformação válida que avalie em uma string.

Valor de Retorno

A string de caracteres minúsculos. Se os dados contiverem caracteres multibyte, o valor retornado dependerá da página de código e do modo de movimento de dados do Serviço de Integração.

NULL se um valor na porta selecionada for NULL.

Exemplo

A seguinte expressão retorna todos os nomes para letra minúscula:

```
LOWER( FIRST_NAME )
```

FIRST_NAME	RETURN VALUE
antonia	antonia
NULL	NULL
THOMAS	thomas
PierRe	pierre
BERNICE	bernice

LPAD

Adiciona um conjunto de espaços em branco ou caracteres ao início de uma string para definir a string como um tamanho especificado.

Sintaxe

```
LPAD( first_string, length [,second_string] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
first_string	Obrigatório	Pode ser uma cadeia de caracteres. As cadeias que você deseja alterar. Você pode inserir qualquer expressão válida de transformação.
tamanho	Obrigatório	Deve ser um número literal inteiro positivo. Este argumento especifica o tamanho que você deseja que cada cadeia tenha.
second_string	Opcional	Pode ser qualquer valor de cadeia. Os caracteres que você deseja anexar ao lado esquerdo dos valores *first_string*. Você pode inserir qualquer expressão válida de transformação. Você pode inserir um literal de cadeia específico. No entanto, anexe os caracteres que você deseja adicionar ao início da cadeia entre aspas simples, como em 'abc'. Esse argumento faz distinção entre maiúsculas e minúsculas. Se você omitir *second_string*, a função preencherá o início da primeira cadeia com espaços em branco.

Valor de Retorno

String do tamanho especificado.

NULL se um valor passado para a função for NULL ou se *length* for um número negativo.

Exemplos

A expressão a seguir padroniza números em seis dígitos preenchendo-os com zeros à esquerda.

```
LPAD( PART_NUM, 6, '0')
```

PART_NUM	RETURN VALUE
702	000702
1	000001
0553	000553
484834	484834

LPAD conta o tamanho da esquerda para a direita. Se a primeira string for maior que o tamanho, LPAD truncará a string da direita para a esquerda. Por exemplo, LPAD('alphabetical', 5, 'x') retornará a string 'alpha'.

Se a segunda string for maior que os caracteres totais necessários para retornar o tamanho especificado, LPAD usará uma parte da segunda string:

```
LPAD( ITEM_NAME, 16, '*..*' )
```

ITEM_NAME	RETURN VALUE
Flashlight	*..**.Flashlight
Compass	*..**..**Compass
Regulator System	Regulator System
Safety Knife	*..*Safety Knife

LTRIM

Remove espaços em branco ou caracteres do início de uma string. Você pode usar LTRIM com IIF ou DECODE em uma transformações de Expressão ou Estratégia de Atualização para evitar espaços em uma tabela de destino.

Se você não especificar um parâmetro *trim_set* na expressão:

- No modo UNICODE, LTRIM remove espaços de byte único e duplo do início de uma string.
- No modo ASCII, LTRIM remove apenas espaços de byte único.

Se você usar LTRIM para remover caracteres de uma string, LTRIM compara o *trim_set* a cada caractere no argumento *string*, caractere-por-caractere, a partir do lado esquerdo da string. Se o caractere na string corresponder a qualquer caractere no *trim_set*, LTRIM o removerá. LTRIM continua a comparar e remover caracteres até que ele não consiga encontrar um caractere correspondente no *trim_set*. Em seguida, ele retorna a string, que não inclui caracteres correspondentes.

Sintaxe

```
LTRIM( string [, trim_set] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumentos	Requerido/ Opcional	Descrição
string	Requerido	Qualquer valor de string. Passa as strings que você deseja modificar. Você pode inserir qualquer expressão de transformação válida. Use operadores para executar comparações ou concatenar strings antes de remover caracteres a partir do início de uma string.
trim_set	Opcional	Qualquer valor de string. Passa os caracteres que você deseja remover a partir do início da primeira string. Você pode inserir qualquer expressão de transformação válida. Você também pode inserir uma string de caracteres. No entanto, você deve anexar os caracteres que deseja remover do início da string entre aspas simples, por exemplo, 'abc'. Se você omitir a segunda string, a função removerá todos os espaços em branco do início da string. LTRIM diferencia maiúsculas de minúsculas. Por exemplo, se você quiser remover o caractere 'A' da string 'Alfredo', insira 'A', não 'a'.

Valor de Retorno

String. Os valores de string com os caracteres especificados no argumento *trim_set* removido.

NULL se um valor transmitido para a função for NULL. Se o *trim_set* for NULL, a função retornará NULL.

Exemplo

A seguinte expressão remove os caracteres 'S' e '.' das strings na porta LAST_NAME:

```
LTRIM( LAST_NAME, 'S.')
```

LAST_NAME	RETURN VALUE
Nelson	Nelson
Osborne	Osborne

LAST_NAME	RETURN VALUE
NULL	NULL
S. MacDonald	MacDonald
Sawyer	awyer
H. Bender	H. Bender
Steadman	teadman

LTRIM remove ‘S.’ de S. MacDonald e o ‘S’ de Sawyer e Steadman, mas não o ponto final de H. Bender. Isso porque LTRIM pesquisa, caractere-por-caractere, o conjunto de caracteres que você especifica no argumento *trim_set*. Se o primeiro caractere na string corresponder ao primeiro caractere em *trim_set*, LTRIM o removerá. Em seguida, LTRIM pesquisa o segundo caractere na string. Se ele corresponder ao segundo caractere em *trim_set*, LTRIM o removerá, e assim por diante. Quando o primeiro caractere na string não combinar com o caractere correspondente em *trim_set*, LTRIM retornará a string e avaliará a linha seguinte.

No exemplo de H. Bender, H não corresponde ao caractere no argumento *trim_set*, então LTRIM retorna a string na porta LAST_NAME e transfere para a linha seguinte.

Dicas para LTRIM

Use RTRIM e LTRIM com || ou CONCAT para remover espaços em branco à direita e à esquerda depois de concatenar duas strings.

Você também pode remover vários conjuntos de caracteres ao aninhar LTRIM. Por exemplo, se quiser remover espaços em branco à esquerda e o caractere 'T' de uma coluna de nomes, você poderá criar uma expressão semelhante a seguinte:

```
LTRIM( LTRIM( NAMES ), 'T' )
```

MAKE_DATE_TIME

Retorna a data e a hora baseadas nos valores de entrada.

Sintaxe

```
MAKE_DATE_TIME( year, month, day, hour, minute, second, nanosecond )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
ano	Requerido	Tipo de dados Numérico. Inteiro positivo de 4 dígitos. Se você passar à função um ano de 2 dígitos, o Data Integration Service retornará "00" como os dois primeiros dígitos do ano.
mês	Requerido	Tipo de dados Numérico. Inteiro positivo entre 1 e 12 (janeiro = 1 a dezembro = 12).

Argumento	Requerido/ Opcional	Descrição
dia	Requerido	Tipo de dados Numérico. Inteiro positivo entre 1 e 31 (com exceção dos meses que têm menos de 31 dias: fevereiro, abril, junho, setembro e novembro).
hora	Opcional	Tipo de dados Numérico. Inteiro positivo entre 0 e 24 (onde 0 = 12AM, 12 = 12PM e 24 = 12AM).
minuto	Opcional	Tipo de dados Numérico. Inteiro positivo entre 0 e 59.
segundo	Opcional	Tipo de dados Numérico. Inteiro positivo entre 0 e 59.
nanossegundo	Opcional	Tipo de dados Numérico. Inteiro positivo entre 0 e 999,999,999.

Valor de Retorno

Data como MM/DD/YYYY HH24:MI:SS. Retornará um valor nulo se você não passar para a função um ano, mês ou dia.

Exemplo

A seguinte expressão cria uma data e uma hora nas portas 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

MAX (Datas)

Retorna a data mais recente encontrada em uma porta ou grupo. Você pode aplicar um filtro para limitar as linhas na pesquisa. Você pode aninhar somente uma função de agregação dentro de MAX.

Você também pode usar MAX para retornar o maior valor numérico ou o valor de string mais alto em uma porta ou grupo.

Sintaxe

```
MAX( date [, filter_condition] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
data	Requerido	Tipo de dados Data/Hora. Passa a data para a qual você deseja retornar uma data máxima. Você pode inserir qualquer expressão de transformação válida.
filter_condition	Opcional	Limita as linhas na pesquisa. A condição de filtro deve ser um valor numérico ou avaliada como TRUE, FALSE ou NULL. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Data.

NULL se todos os valores passados para a função forem NULL ou se nenhuma linha for selecionada (por exemplo, a condição de filtro avaliada como FALSE ou NULL em todas as linhas).

Exemplo

Você pode retornar a data máxima para uma porta ou um grupo. A seguinte expressão retorna a data de pedido máxima para lanternas:

```
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)

Retorna o valor numérico máximo encontrado em uma porta ou grupo. Você pode aplicar um filtro para limitar as linhas na pesquisa. Você pode aninhar somente uma função de agregação dentro de MAX. Você também pode usar MAX para retornar a data mais recente ou o maior valor de string em uma porta ou grupo.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
valor_numérico	Requerido	Tipo de dados Numérico. Passa os valores numéricos para os quais você deseja retornar um valor numérico máximo. Você pode inserir qualquer expressão de transformação válida.
filter_condition	Opcional	Limita as linhas na pesquisa. A condição de filtro deve ser um valor numérico ou avaliada como TRUE, FALSE ou NULL. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Valor numérico.

NULL se todos os valores passados para a função forem NULL ou se nenhuma linha for selecionada (por exemplo, a condição de filtro avaliada como FALSE ou NULL em todas as linhas).

Nulls

Se um valor for NULL, MAX irá ignorá-lo. No entanto, se todos os valores passados da porta forem NULL, MAX retornará NULL.

Agrupar por

MAX agrupa valores baseados em grupos por portas que você define na transformação, retornando um resultado para cada grupo.

Se não houver grupo por porta, MAX tratará todas as linhas como um grupo, retornando um valor.

Exemplo

A primeira expressão retorna o preço máximo de lanternas:

```
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 (String)

Retorna o valor de string mais alto encontrado em uma porta ou grupo. Você pode aplicar um filtro para limitar as linhas na pesquisa. Você pode aninhar somente uma função de agregação dentro de MAX.

Nota: A função MAX usa a mesma ordem de classificação que a transformação de Classificador. No entanto, a função MAX faz distinção entre maiúsculas e minúsculas, e a transformação de Classificador pode não fazer distinção entre maiúsculas e minúsculas.

Você também pode usar MAX para retornar a data mais recente ou o maior valor numérico em uma porta ou grupo.

Sintaxe

```
MAX( string [, filter_condition] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
string	Requerido	Tipo de dados strings. Passa os valores de string para os quais você deseja retornar um valor de string máximo. Você pode inserir qualquer expressão de transformação válida.
filter_condition	Opcional	Limita as linhas na pesquisa. A condição de filtro deve ser um valor numérico ou avaliada como TRUE, FALSE ou NULL. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

String.

NULL se todos os valores passados para a função forem NULL ou se nenhuma linha for selecionada (por exemplo, a condição de filtro avaliada como FALSE ou NULL em todas as linhas).

Nulls

Se um valor for NULL, MAX irá ignorá-lo. No entanto, se todos os valores passados da porta forem NULL, MAX retornará NULL.

Agrupar por

MAX agrupa valores baseados em grupos por portas que você define na transformação, retornando um resultado para cada grupo.

Se não houver grupo por porta, MAX tratará todas as linhas como um grupo, retornando um valor.

Exemplo

A seguinte expressão retorna o nome de item máximo da ID do fabricante 104:

```
MAX( 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: Rope (20 ft)

MD5

Calcula a soma de verificação do valor de entrada. A função usa o algoritmo Message-Digest 5 (MD5). MD5 é uma função de hash criptográfica unidirecional com um valor de hash de 128 bits. Você pode concluir que os valores de entrada são diferentes quando as somas de verificação dos valores de entrada são diferentes. Use MD5 para verificar a integridade de dados.

Sintaxe

```
MD5( value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
valor	Requerido	Tipo de dados String ou Binary. Valor para o qual você deseja calcular a soma de verificação. A diferenciação entre maiúscula e minúscula do valor de entrada afeta o valor retornado. Por exemplo, MD5(informatica) e MD5 (Informatica) retornam valores diferentes.

Valor de Retorno

String exclusiva de 32 caracteres de dígitos hexadecimais de 0 a 9 e de a a f.

NULL se a entrada for um valor nulo.

Exemplo

Você deseja gravar dados alterados em um banco de dados. Use MD5 para gerar valores de soma de verificação em linhas de dados lidas em uma origem. Ao executar uma sessão, compare os valores de soma de verificação gerados anteriormente com os novos. Em seguida, grave as linhas com os valores de soma de verificação atualizados no destino. Você pode concluir que um valor de soma de verificação atualizado indica que os dados foram alterados.

Dica

Você pode usar o valor retornado como uma chave hash.

MEDIAN

Retorna o valor mediano de todos os valores em uma porta selecionada.

Se houver um número igual de valores na porta, o valor mediano será a média dos dois valores do meio quando todos os valores forem colocados em ordem em uma linha de número. Se houver um número diferente de valores na porta, o valor mediano será o número do meio.

Você pode aninhar apenas uma outra função de agregação dentro de MEDIAN, e a função aninhada deverá retornar um tipo de dados Numérico.

O Data Integration Service lê todas as linhas de dados para realizar o cálculo da mediana. O processo de leitura de linhas de dados para realizar o cálculo pode afetar o desempenho. Como opção, você pode aplicar um filtro para limitar as linhas lidas para calcular o valor mediano.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
valor_numérico	Requerido	Tipo de dados Numérico. Passa os valores para os quais você deseja calcular um valor mediano. Você pode inserir qualquer expressão de transformação válida.
filter_condition	Opcional	Limita as linhas na pesquisa. A condição de filtro deve ser um valor numérico ou avaliada como TRUE, FALSE ou NULL. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Valor numérico.

NULL se todos os valores passados para a função forem NULL ou se nenhuma linha for selecionada. Por exemplo, a condição de filtro é avaliada como FALSE ou NULL em todas as linhas.

Nota: Se o valor retornado for Decimal com precisão superior a 15, habilite a alta precisão para garantir precisão decimal de até 28 dígitos.

Nulls

Se um valor for NULL, MEDIAN ignorará a linha. No entanto, se todos os valores passados da porta forem NULL, MEDIAN retornará NULL.

Agrupar por

MEDIAN agrupa valores baseados em portas agrupar por que você define na transformação, retornando um resultado para cada grupo.

Se não houver porta agrupar por, MEDIAN tratará todas as linhas como um grupo, retornando um valor.

Exemplo

Para calcular o salário mediano de todos os departamentos, crie uma transformação de Agregador agrupada por departamentos com uma porta que especifica a seguinte expressão:

```
MEDIAN( SALARY )
```

A seguinte expressão retorna o valor mediano de pedidos de estabilização de benefícios proporcionais diferidos:

```
MEDIAN( SALES, ITEM = 'Stabilizing Vest' )
```

ITEM	SALES
Flashlight	85
Stabilizing Vest	504
Stabilizing Vest	36
Safety Knife	5
Medium Titanium Knife	150
Tank	NULL
Stabilizing Vest	441
Chisel Point Knife	60
Stabilizing Vest	NULL
Stabilizing Vest	1044
Wrist Band Thermometer	110

RETURN VALUE: 472.5

METAPHONE

Codifica valores de cadeia. Você pode especificar o tamanho da cadeia a ser codificada.

METAPHONE codifica caracteres do alfabeto do idioma inglês (A-Z). Ele codifica as letras maiúsculas e minúsculas em maiúsculas.

METAPHONE codifica caracteres de acordo com a seguinte lista de regras:

- Ignora vogais (A, E, I, O e U), a menos que uma delas seja o primeiro caractere da string de entrada. METAPHONE('CAR') retorna 'KR' e METAPHONE('AAR') retorna 'AR'.
- Usa diretrizes de codificação especiais.

A seguinte tabela lista as diretrizes de codificação de METAPHONE:

Entrada	Retornos	Condição	Exemplo
B	- n/d	- quando segue M	- METAPHONE ('Lamb') retorna LM.
B	- B	- em todos os outros casos	- METAPHONE ('Box') retorna BKS.
C	- X	- quando seguido de IA ou H	- METAPHONE ('Facial') retorna FXL.

Entrada	Retornos	Condição	Exemplo
C	- S	- quando seguido de I, E ou Y	- METAPHONE (‘Fence’) retorna FNS.
C	- n/d	- quando segue S e é seguido por I, E ou Y	- METAPHONE (‘Scene’) retorna SN.
C	- K	- em todos os outros casos	- METAPHONE (‘Cool’) retorna KL.
D	- J	- quando seguido por GE, GY ou GI	- METAPHONE (‘Dodge’) retorna TJ.
D	- T	- em todos os outros casos	- METAPHONE (‘David’) retorna TFT.
F	- F	- em todos os casos	- METAPHONE (‘FOX’) retorna FKS.
G	- F	- quando seguido por H e o primeiro caractere na string de entrada não é B, D ou H	- METAPHONE (‘Tough’) retorna TF.
G	- n/d	- quando seguido por H e o primeiro caractere na string de entrada não é B, D ou H	- METAPHONE (‘Hugh’) retorna HF.
G	- J	- quando seguido por I, E ou Y e não repete	- METAPHONE (‘Magic’) retorna MJK.
G	- K	- em todos os outros casos	- METAPHONE(‘GUN’) retorna KN.
H	- H	- quando não segue C, G, P, S ou T e é seguido por A, E, I ou U	- METAPHONE (‘DHAT’) retorna THT.
H	- n/d	- em todos os outros casos	- METAPHONE (‘Chain’) retorna XN.
J	- J	- em todos os casos	- METAPHONE (‘Jen’) retorna JN.
K	- n/d - K	- quando segue C - em todos os outros casos	- METAPHONE (‘Ckim’) retorna KM. - METAPHONE (‘Kim’) retorna KM.
L	- L	- em todos os casos	- METAPHONE (‘Laura’) retorna LR.
M	- M	- em todos os casos	- METAPHONE (‘Maggi’) retorna MK.
N	- N	- em todos os casos	- METAPHONE (‘Nancy’) retorna NNS.
P	- F	- quando seguido por H	- METAPHONE (‘Phone’) retorna FN.
P	- P	- em todos os outros casos	- METAPHONE (‘Pip’) retorna PP.
Q	- K	- em todos os casos	- METAPHONE (‘Queen’) retorna KN.
R	- R	- em todos os casos	- METAPHONE (‘Ray’) retorna R.
S	- X	- quando seguido por H, IO, IA ou CHW	- METAPHONE (‘Cash’) retorna KX.
S	- S	- em todos os outros casos	- METAPHONE (‘Sing’) retorna SNK.
T	- X	- quando seguido por IA ou IO	- METAPHONE (‘Patio’) retorna PX.
T	- 0[1]	- quando seguido por H	- METAPHONE (‘Thor’) retorna 0R.

Entrada	Retornos	Condição	Exemplo
T	- n/d	- quando seguido por CH	- METAPHONE (‘Glitch’) retorna KLTX.
T	- T	- em todos os outros casos	- METAPHINE (‘Tim’) retorna TM.
V	- F	- em todos os casos	- METAPHONE (‘Vin’) retorna FN.
W	- W	- quando seguido por A, E, I, O ou U	- METAPHONE (‘Wang’) retorna WNK.
W	- n/d	- em todos os outros casos	- METAPHONE (‘When’) retorna HN.
X	- KS	- em todos os casos	- METAPHONE (‘Six’) retorna SKS.
Y	- Y	- quando seguido por A, E, I, O ou U	- METAPHONE (‘Yang’) retorna YNK.
Y	- n/d	- em todos os outros casos	- METAPHONE (‘Bobby’) retorna BB.
Z	- S	- em todos os casos	- METAPHONE (‘Zack’) retorna SK.

1. O inteiro 0.

- Ignorará o caractere inicial e codificará a string restante se os dois primeiros caracteres da string de entrada tiverem um dos seguintes valores:
  - **KN.** Por exemplo, METAPHONE(‘KNOT’) retorna ‘NT’.
  - **GN.** Por exemplo, METAPHONE(‘GNOB’) retorna ‘NB’.
  - **PN.** Por exemplo, METAPHONE(‘PNRX’) retorna ‘NRKS’.
  - **AE.** Por exemplo, METAPHONE(‘AERL’) retorna ‘ERL’.
- Se um caractere diferente de "C" ocorrer mais de uma vez na string de entrada, codificará apenas a primeira ocorrência. Por exemplo, METAPHONE(‘BBOX’) retorna ‘BKS’ e METAPHONE(‘CCOX’) retorna ‘KKKS’.

Sintaxe

```
METAPHONE( string [,length] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
string	Obrigatório	Deve ser uma cadeia de caracteres. Passa o valor que você deseja codificar. O primeiro caractere deve ser um caractere do alfabeto do idioma inglês (A-Z). Você pode inserir qualquer expressão válida de transformação. Ignora qualquer caractere não alfabético na *string*.
length	Opcional	Deve ser um inteiro maior que 0. Especifica o número de caracteres na *string* que você deseja codificar. Você pode inserir qualquer expressão válida de transformação. Quando o *length* é 0 ou um valor maior que o tamanho da *string*, codifica toda a string de entrada. O padrão é 0.

Valor de Retorno

Cadeia.

NULL se uma das seguintes condições for verdadeira:

- Todos os valores passados para a função são NULL.
- Nenhum caractere na *string* é uma letra do alfabeto inglês.
- *string* está vazia.

Exemplos

A seguinte expressão codifica os dois primeiros caracteres na porta EMPLOYEE_NAME para uma string:

```
METAPHONE( EMPLOYEE_NAME, 2 )
```

Employee_Name	Return Value
John	JH
*@#$	NULL
P$%%oc&&KMNL	PK

A seguinte expressão codifica os quatro primeiros caracteres na porta EMPLOYEE_NAME para uma string:

```
METAPHONE( EMPLOYEE_NAME, 4 )
```

Employee_Name	Return Value
John	JHN
1ABC	ABK
*@#$	NULL
P$%%oc&&KMNL	PKKM

MIN (Datas)

Retorna a data mais antiga encontrada em uma porta ou grupo. Você pode aplicar um filtro para limitar as linhas na pesquisa. Você pode aninhar apenas uma outra função de agregação dentro de MIN, e a função aninhada deverá retornar um tipo de dados de data.

Você também pode usar MIN para retornar o menor valor numérico ou o valor de string mais baixo em uma porta ou grupo.

Sintaxe

```
MIN( date [, filter_condition] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
data	Requerido	Tipo de dados Data/Hora. Passa os valores para os quais você deseja retornar o valor mínimo. Você pode inserir qualquer expressão de transformação válida.
filter_condition	Opcional	Limita as linhas na pesquisa. A condição de filtro deve ser um valor numérico ou avaliada como TRUE, FALSE ou NULL. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Data se o argumento de *valor* for uma data.

NULL se todos os valores passados para a função forem NULL ou se nenhuma linha for selecionada (por exemplo, a condição de filtro avaliada como FALSE ou NULL em todas as linhas).

Nulls

Se um único valor for NULL, MIN irá ignorá-lo. No entanto, se todos os valores passados da porta forem NULL, MIN retornará NULL.

Agrupar por

MIN agrupa valores baseados em grupos por portas que você define na transformação, retornando um resultado para cada grupo.

Se não houver grupo por porta, MIN tratará todas as linhas como um grupo, retornando um valor.

Exemplo

A seguinte expressão retorna a data de pedido mais antiga para lanternas:

```
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

ITEM_NAME	ORDER_DATE
Halogen Flashlight	Feb 1 1998
Flashlight	Oct 10 1998
Flashlight	NULL

RETURN VALUE: Feb 1 1998

MIN (Números)

Retorna o menor valor numérico encontrado em uma porta ou grupo. Você pode aplicar um filtro para limitar as linhas na pesquisa. Você pode aninhar apenas uma outra função de agregação dentro de MIN, e a função aninhada deverá retornar um tipo de dados numérico.

Você também pode usar MIN para retornar a data mais recente ou o menor valor de string em uma porta ou grupo.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
valor_numérico	Requerido	Tipos de dados numéricos. Passa os valores para os quais você deseja retornar o valor mínimo. Você pode inserir qualquer expressão de transformação válida.
filter_condition	Opcional	Limita as linhas na pesquisa. A condição de filtro deve ser um valor numérico ou avaliada como TRUE, FALSE ou NULL. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Valor numérico.

NULL se todos os valores passados para a função forem NULL ou se nenhuma linha for selecionada (por exemplo, a condição de filtro avaliada como FALSE ou NULL em todas as linhas).

Nota: Se o valor retornado for Decimal com precisão superior a 15, habilite a alta precisão para garantir precisão decimal de até 28 dígitos.

Nulls

Se um único valor for NULL, MIN irá ignorá-lo. No entanto, se todos os valores passados da porta forem NULL, MIN retornará NULL.

Agrupar por

MIN agrupa valores baseados em grupos por portas que você define na transformação, retornando um resultado para cada grupo.

Se não houver grupo por porta, MIN tratará todas as linhas como um grupo, retornando um valor.

Exemplo

A seguinte expressão retorna o preço mínimo de lanternas:

```
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 (String)

Retorna o valor de string mais baixo encontrado em uma porta ou grupo. Você pode aplicar um filtro para limitar as linhas na pesquisa. Você pode aninhar apenas uma outra função de agregação dentro de MIN e a função aninhada deverá retornar um tipo de dados de string.

Nota: A função MIN usa a mesma ordem de classificação que a transformação de Classificador. No entanto, a função MIN faz distinção a maiúsculas e minúsculas, mas a transformação de Classificador podem não fazer distinção a maiúsculas e minúsculas.

Você também pode usar MIN para retornar a data mais recente ou o valor numérico mínimo em uma porta ou grupo.

Sintaxe

```
MIN( string [, filter_condition] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
string	Requerido	Tipo de dados strings. Passa os valores para os quais você deseja retornar o valor mínimo. Você pode inserir qualquer expressão de transformação válida.
filter_condition	Opcional	Limita as linhas na pesquisa. A condição de filtro deve ser um valor numérico ou avaliada como TRUE, FALSE ou NULL. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Valor de string.

NULL se todos os valores passados para a função forem NULL ou se nenhuma linha for selecionada (por exemplo, a condição de filtro avaliada como FALSE ou NULL em todas as linhas).

Nulls

Se um único valor for NULL, MIN irá ignorá-lo. No entanto, se todos os valores passados da porta forem NULL, MIN retornará NULL.

Agrupar por

MIN agrupa valores baseados em grupos por portas que você define na transformação, retornando um resultado para cada grupo.

Se não houver grupo por porta, MIN tratará todas as linhas como um grupo, retornando um valor.

Exemplo

A seguinte expressão retorna o nome de item mínimo da ID do fabricante 104:

```
MIN ( 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: 60.6 cu ft Tank

MOD

Retorna o restante de um cálculo de divisão. Por exemplo, `MOD(8,5)` retorna 3.

Sintaxe

```
MOD( numeric_value, divisor )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
valor_numérico	Requerido	Tipo de dados Numérico. Os valores que deseja dividir. Você pode inserir qualquer expressão de transformação válida.
divisor	Requerido	O valor numérico que você deseja dividir. O divisor não pode ser 0.

Valor de Retorno

O valor numérico do tipo de dados que você passa para a função. O restante do valor numérico dividido pelo divisor.

NULL se um valor transmitido para a função for NULL.

Exemplos

A seguinte expressão retorna o módulo dos valores na porta PRICE dividido pelos valores na porta 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.*

A última linha (25, 0) produziu um erro porque você não pode dividir por 0. Para evitar dividir por 0, você pode criar uma expressão semelhante ao seguinte, que retorna o módulo de Preço dividido pela Quantidade somente se a quantidade não for 0. Se a quantidade for 0, a função retornará 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

PRICE	QTY	RETURN VALUE
20.00	NULL	NULL
25.00	0	NULL

A última linha (25, 0) produziu um NULL em vez de um erro porque a função IIF substitui NULL pelo 0 na porta QTY.

MOVINGAVG

Retorna a média (linha por linha) de um conjunto especificado de valores. Você também pode aplicar uma condição para filtrar linhas antes de calcular a média móvel.

Sintaxe

```
MOVINGAVG( numeric_value, rowset [, filter_condition] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
valor_numérico	Requerido	Tipo de dados Numérico. Os valores para os quais você deseja calcular uma média móvel. Você pode inserir qualquer expressão de transformação válida.
conjunto de linhas	Requerido	Deve ser um literal inteiro positivo maior que 0. Define o conjunto de linhas para o qual você deseja calcular a média móvel. Por exemplo, se quiser calcular uma média móvel de uma coluna de dados, cinco linhas por vez, você poderá gravar uma expressão, como: `MOVINGAVG(SALES, 5)`.
filter_condition	Opcional	Limita as linhas na pesquisa. A condição de filtro deve ser um valor numérico ou avaliada como TRUE, FALSE ou NULL. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Valor numérico.

NULL se todos os valores passados para a função forem NULL ou se nenhuma linha for selecionada (por exemplo, a condição de filtro avaliada como FALSE ou NULL em todas as linhas).

Nota: Se o valor retornado for Decimal com precisão superior a 15, habilite a alta precisão para garantir precisão decimal de até 28 dígitos.

Nulls

MOVINGAVG ignora valores nulos ao calcular a média móvel. No entanto, se todos os valores forem NULL, a função retornará NULL.

Exemplo

A seguinte expressão retorna a ordem média de um Stabilizing Vest, com base nas primeiras cinco linhas na porta Vendas e então retorna a média das últimas cinco linhas lidas:

```
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

A função retorna a média de um conjunto de cinco linhas: 358 baseados nas linhas de 1 a 5, 245.8 baseados nas linhas de 2 a 6 e 243 baseados nas linhas de 3 a 7.

MOVINGSUM

Retorna a soma (linha-por-linha) de um conjunto especificado de linhas.

Como opção, você pode aplicar uma condição para filtrar linhas antes de calcular a soma móvel.

Sintaxe

```
MOVINGSUM( numeric_value, rowset [, filter_condition] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
valor_numérico	Requerido	Tipo de dados Numérico. Os valores para os quais você deseja calcular uma soma móvel. Você pode inserir qualquer expressão de transformação válida.
conjunto de linhas	Requerido	Deve ser um literal inteiro positivo maior que 0. Define o conjunto de linhas para o qual você deseja calcular a soma móvel. Por exemplo, se quiser calcular uma soma móvel de uma coluna de dados, cinco linhas por vez, você poderá gravar uma expressão como: `MOVINGSUM( SALES, 5 )`
filter_condition	Opcional	Limita as linhas na pesquisa. A condição de filtro deve ser um valor numérico ou avaliada como TRUE, FALSE ou NULL. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Valor numérico.

NULL se todos os valores passados para a função forem NULL ou se a função não selecionar nenhuma linha (por exemplo, a condição de filtro avaliada como FALSE ou NULL em todas as linhas).

Nota: Se o valor retornado for Decimal com precisão superior a 15, habilite a alta precisão para garantir precisão decimal de até 28 dígitos.

Nulls

MOVINGSUM ignora valores nulos ao calcular a soma móvel. No entanto, se todos os valores forem NULL, a função retornará NULL.

Exemplo

A seguinte expressão retorna a soma de ordens de um Stabilizing Vest, com base nas primeiras cinco linhas na porta Vendas e então retorna a média das últimas cinco linhas lidas:

```
MOVINGSUM( SALES, 5 )
```

ROW_NO	SALES	RETURN VALUE
1	600	NULL
2	504	NULL
3	36	NULL
4	100	NULL
5	550	1790
6	39	1229
7	490	1215

A função retorna a soma de um conjunto de cinco linhas: 1790 baseados nas linhas de 1 a 5, 1229 baseados nas linhas de 2 a 6 e 1215 baseados nas linhas de 3 a 7.

NPER

Retorna o número de períodos para um investimento baseado em uma taxa constante de juros e em pagamentos periódicos e constantes.

Sintaxe

```
NPER( rate, present value, payment [, future value, type] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
taxa	Requerido	Numérico. Taxa de juros ganhos em cada período. Expresso como um número decimal. Divida a taxa por 100 para expressá-la como um número decimal. Deve ser maior ou igual a 0.
valor atual	Requerido	Numérico. Montante da soma global que uma série de pagamentos futuros vale.
pagamento	Requerido	Numérico. Montante de pagamento devido por período. Deve ser um número negativo.
valor futuro	Opcional	Numérico. Saldo de caixa que você deseja conseguir depois que o último pagamento for feito. Se você omitir esse valor, NPER usará 0.
tipo	Opcional	Booleano. Momento do pagamento. Insira 1 se o pagamento for no início do período. Insira 0 se o pagamento for no final do período. O padrão é 0. Se você inserir um valor diferente de 0 ou 1, o Data Integration Service tratará o valor como 1.

Valor de Retorno

Numérico.

Exemplo

O valor atual de um investimento é US$ 500. Cada pagamento é de US$ 2000 e o valor futuro do investimento é de US$ 20.000. A seguinte expressão retorna 9 como o número de períodos em que você precisa fazer os pagamentos:

```
NPER ( 0.015, -500, -2000, 20000, TRUE )
```

Observações

Para calcular a taxa de juros ganhos em cada período, divida a taxa anual pelo número de pagamentos feitos no ano. Por exemplo, se você fizer pagamentos mensais a uma taxa de juros de 15% ao ano, o valor do argumento Taxa será 15% dividido por 12. Se fizer pagamentos anuais, o valor do argumento Taxa será 15%.

O valor do pagamento e o valor atual são negativos porque indicam montantes pagos.

PERCENTILE

Calcula o valor que se encaixa em determinado percentil em um grupo de números. Você pode aninhar apenas uma outra função de agregação dentro de PERCENTILE, e a função aninhada deverá retornar um tipo de dados Numérico.

O Data Integration Service lê todas as linhas de dados para realizar o cálculo do percentil. O processo de leitura de linhas para realizar o cálculo pode afetar o desempenho. Como opção, você pode aplicar um filtro para limitar as linhas lidas para calcular o percentil.

Sintaxe

```
PERCENTILE( numeric_value, percentile [, filter_condition ] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
valor_numérico	Requerido	Tipo de dados Numérico. Passa os valores para os quais você deseja calcular um percentil. Você pode inserir qualquer expressão de transformação válida.
percentil	Requerido	Inteiro entre 0 e 100, inclusive. Passa o percentil que você deseja calcular. Você pode inserir qualquer expressão de transformação válida. Se você passar um número fora do intervalo de 0 a 100, o Data Integration Service exibirá um erro e não gravará a linha.
filter_condition	Opcional	Limita as linhas na pesquisa. A condição de filtro deve ser um valor numérico ou avaliada como TRUE, FALSE ou NULL. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Valor numérico.

NULL se todos os valores passados para a função forem NULL ou se nenhuma linha for selecionada (por exemplo, a condição de filtro avaliada como FALSE ou NULL em todas as linhas).

Nota: Se o valor retornado for Decimal com precisão superior a 15, habilite a alta precisão para garantir precisão decimal de até 28 dígitos.

Nulls

Se um valor for NULL, PERCENTILE ignorará a linha. Porém, se todos os valores em um grupo forem NULL, PERCENTILE retornará NULL.

Agrupar por

PERCENTILE agrupa valores baseados em grupos por portas que você define na transformação, retornando um resultado para cada grupo.

Se não houver grupo por porta, PERCENTILE tratará todas as linhas como um grupo, retornando um valor.

Exemplo

O Data Integration Service calcula um percentil usando a seguinte lógica:

$$I = \frac{(x + 1) \times \text{percentil}}{100}$$

Use as seguintes diretrizes para essa equação:

- *x* é o número de elementos no grupo de valores para o qual você está calculando um percentil.
- Se *i* < 1, PERCENTILE retornará o valor do primeiro elemento na lista.
- Se *i* for um valor inteiro, PERCENTILE retornará o valor do elemento *i*th na lista.
- Caso contrário, PERCENTILE retornará o valor de *n*:

$$n = [\lfloor i \rfloor \text{th?element} \times (\lceil i \rceil - i)] + [\lceil i \rceil \text{th?element} \times (i - \lfloor i \rfloor)]$$

A seguinte expressão retorna o salário que se encaixa no 75º percentil de salários mais altos que US $ 50.000:

```
PERCENTILE( SALARY, 75, SALARY > 50000 )
```

SALARY
125000.0
27900.0
100000.0
NULL
55000.0
9000.0
85000.0
86000.0
48000.0
99000.0

RETURN VALUE: 106250.0

PMT

Retorna o pagamento para um empréstimo baseado em pagamentos constantes e taxa de juros constante.

Sintaxe

```
PMT( rate, terms, present value[, future value, type] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
taxa	Requerido	Numérico. Taxa de juros do empréstimo para cada período. Expresso como um número decimal. Divida a taxa por 100 para expressá-la como um número decimal. Deve ser maior ou igual a 0.
prazo	Requerido	Numérico. Número de períodos ou pagamentos. Deve ser maior que 0.
valor atual	Requerido	Numérico. Princípio para o empréstimo.

Argumento	Requerido/ Opcional	Descrição
valor futuro	Opcional	Numérico. Saldo de caixa que você deseja conseguir após o último pagamento. Se você omitir esse valor, PMT usará 0.
tipo	Opcional	Booleano. Momento do pagamento. Insira 1 se o pagamento for no início do período. Insira 0 se o pagamento for no final do período. O padrão é 0. Se você inserir um valor diferente de 0 ou 1, o Data Integration Service tratará o valor como 1.

Valor de Retorno

Numérico.

Exemplo

A seguinte expressão retorna -2.111,64 como o montante do pagamento mensal de um empréstimo:

```
PMT( 0.01, 10, 20000 )
```

Observações

Para calcular a taxa de juros ganhos em cada período, divida a taxa anual pelo número de pagamentos feitos em um ano. Por exemplo, se você fizer pagamentos mensais a uma taxa de juros anual de 15%, a taxa será 15%/12. Se você fizer pagamentos anuais, a taxa será de 15%.

O valor do pagamento é negativo porque indica montantes pagos.

POWER

Retorna um valor elevado ao expoente que você passa para a função.

Sintaxe

```
POWER( base, exponent )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
base	Requerido	Valor numérico. Esse argumento é o valor base. Você pode inserir qualquer expressão de transformação válida. Quando o valor base é negativo, o expoente deve ser um inteiro.
expoente	Requerido	Valor numérico. Esse argumento é o valor expoente. Você pode inserir qualquer expressão de transformação válida. Quando o valor base é negativo, o expoente deve ser um inteiro. Nesse caso, a função arredonda todos os valores decimais para o inteiro mais próximo antes de retornar um valor.

Valor de Retorno

Valor duplo.

NULL se você passar um valor nulo para a função.

Exemplo

A seguinte expressão retorna os valores na porta Números elevados aos valores na porta Expoente:

```
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

O valor -3,0 elevado a 6 retorna os mesmos resultados que -3,0 elevado a 5,5. Se a base for negativa, o expoente deverá ser um inteiro. Caso contrário, o Data Integration Service arredondará o expoente para o valor inteiro mais próximo.

PV

Retorna o valor presente de um investimento.

Sintaxe

```
PV( rate, terms, payment [, future value, type] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
taxa	Requerido	Numérico. Taxa de juros ganhos em cada período. Expressa um número decimal. Divida a taxa por 100 para expressá-la como um número decimal. Deve ser maior ou igual a 0.
prazo	Requerido	Numérico. Número de períodos ou pagamentos. Deve ser maior que 0.
pagamentos	Requerido	Numérico. Montante de pagamento devido por período. Deve ser um número negativo.

Argumento	Requerido/ Opcional	Descrição
valor futuro	Opcional	Numérico. Saldo de caixa após o último pagamento. Se você omitir esse valor, PV usará 0.
tipos	Opcional	Booleano. Momento do pagamento. Insira 1 se o pagamento for no início do período. Insira 0 se o pagamento for no final do período. O padrão é 0. Se você inserir um valor diferente de 0 ou 1, o Data Integration Service tratará o valor como 1.

Valor de Retorno

Numérico.

Exemplo

A seguinte expressão retornará 12.524,43 como o montante que você deve depositar na conta hoje para ter um valor futuro de US$ 20.000 em um ano se você também depositar US$ 500 no início de cada período:

```
PV( 0.0075, 12, -500, 20000, TRUE )
```

RAND

Retorna um número aleatório entre 0 e 1. Isso é útil para cenários de probabilidade.

Sintaxe

```
RAND( seed )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
semente	Opcional	Numérico. Valor inicial para que o Serviço de Integração gere o número aleatório. O valor deve ser uma constante Se você não inserir uma semente, o Data Integration Service usará a hora do sistema atual para derivar os números de segundos desde 1º de janeiro de 1971. Ele usa esse valor como a semente.

Valor de Retorno

Numérico.

Na mesma semente, o Data Integration Service gera a mesma sequência de números.

Exemplo

A seguinte expressão pode retornar um valor de 0.417022004702574:

```
RAND (1)
```

RATE

Retorna a taxa de juros que um título ganha por período.

Sintaxe

```
RATE( terms, payment, present value[, future value, type] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
prazo	Requerido	Numérico. Número de períodos ou pagamentos. Deve ser maior que 0.
pagamentos	Requerido	Numérico. Montante de pagamento devido por período. Deve ser um número negativo.
valor atual	Requerido	Numérico. Montante da soma global que uma série de pagamentos futuros vale agora.
valor futuro	Opcional	Numérico. Saldo de caixa que você deseja conseguir após o último pagamento. Por exemplo, o valor futuro de um empréstimo é 0. Se você omitir este argumento, RATE usará 0.
tipos	Opcional	Booleano. Momento do pagamento. Insira 1 se o pagamento for no início do período. Insira 0 se o pagamento for no final do período. O padrão é 0. Se você inserir um valor diferente de 0 ou 1, o Data Integration Service tratará o valor como 1.

Valor de Retorno

Numérico.

Exemplo

A seguinte expressão retorna 0,0077 como a taxa de juros mensal de um empréstimo:

```
RATE( 48, -500, 20000 )
```

Para calcular a taxa de juros anual do empréstimo, multiplique 0.0077 por 12. A taxa de juros anual é 0.0924 ou 9.24%.

REG_EXTRACT

Extrai sub-padrões de uma expressão regular dentro de um valor de entrada. Por exemplo, de um padrão de expressão regular para um nome completo, você pode extrair o primeiro nome ou sobrenome.

Nota: Use a função REG_REPLACE para substituir um padrão de caractere em uma string com outro padrão de caractere.

Sintaxe

```
REG_EXTRACT( subject, 'pattern', subPatternNum )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
assunto	Obrigatório	Tipo de dados de cadeia. Passa o valor que você deseja comparar em relação ao padrão de expressão regular.
padrão	Obrigatório	Tipo de dados de cadeia. Padrão de expressão regular que você deseja corresponder. É necessário usar a sintaxe de expressão regular compatível com perl. Coloque o padrão entre aspas simples. Coloque cada subpadrão entre parênteses.
subPatternNum	Opcional	Valor inteiro. Número de subpadrão da expressão regular que você deseja corresponder. Use as diretrizes a seguir para determinar o número de subpadrão. - nenhum valor ou 1. Extrai o primeiro sub-padrão da expressão regular. - 2. Extrai o segundo sub-padrão da expressão regular. - n. Extrai o *n*th sub-padrão de expressão regular. O padrão é 1.

Usando sintaxe de Expressão Regular Compatível com perl

Você deve usar a sintaxe de expressão regular compatível com perl com as funções REG_EXTRACT, REG_MATCH e REG_REPLACE.

A seguinte tabela fornece diretrizes de sintaxe de expressão regular compatíveis com perl:

Sintaxe	Descrição
. (um ponto)	Faz a correspondência de qualquer caractere.
[a-z]	Corresponde a uma instância de um caractere em letra minúscula. Por exemplo, [a-z] corresponde a ab. Use [A-Z] para corresponder a caracteres em letra maiúscula.
\d	Faz a correspondência da instância de qualquer dígito de 0 a 9.
\s	Faz a correspondência de um caractere de espaço em branco.
\w	Corresponde a um caractere alfanumérico, inclusive o caractere sublinhado (_)
()	Agrupa uma expressão. Por exemplo, os parênteses em (\d-\d-\d\d) agrupam a expressão \d\d-\d\d, que encontra qualquer grupo de dois números seguido de um hífen e de qualquer grupo de dois números, como em 12-34.
{}	Faz a correspondência do número de caracteres. Por exemplo, \d{3} corresponde a qualquer grupo de três números, como 650 ou 510. Ou, [a-z]{2} corresponde a qualquer grupo de duas letras, como CA ou NY.
?	Faz a correspondência do caractere ou grupo de caracteres precedente nenhuma ou uma vez. Por exemplo, \d{3}(-{d{4})? faz a correspondência de qualquer grupo de três números, que pode ser seguido de um hífen ou de qualquer grupo de quatro números.

Sintaxe	Descrição
* (um asterisco)	Faz a correspondência de nenhuma ou mais instâncias dos valores que seguem o asterisco. Por exemplo, *0 é qualquer valor que preceda um 0.
+	Faz a correspondência de uma ou mais instâncias dos valores que seguem o sinal de adição. Por exemplo, \w+ é qualquer valor que siga um caractere alfanumérico.

Por exemplo, a seguinte expressão regular encontra códigos postais norte-americanos de 5 dígitos, como 93930, e códigos postais de 9 dígitos, como 93930-5407:

```
\d{5}(-\d{4})?
```

\d{5} refere-se a qualquer grupo de cinco números, como 93930. Os parênteses em torno de -\d{4} agrupam esse segmento da expressão. O hífen representa o hífen de um código de 9 dígitos, como em 93930-5407. \d{4} refere-se a qualquer grupo de quatro números, como 5407. O ponto de interrogação determina que o hífen e os quatro últimos dígitos são opcionais ou podem aparecer uma vez.

Convertendo Sintaxe COBOL em Sintaxe de Expressão Regular Compatível com Perl

Se estiver familiarizado com a sintaxe COBOL, você poderá usar as informações a seguir para gravar expressões regulares compatíveis com perl.

A seguinte tabela mostra exemplos de sintaxe COBOL e seus equivalentes perl:

Sintaxe COBOL	Sintaxe perl	Descrição
9	\d	Faz a correspondência da instância de qualquer dígito de 0 a 9.
9999	\d\d\d\d ou \d{4}	Corresponde a quaisquer quatro dígitos de 0 a 9, como em 1234 ou 5936.
x	[a-z]	Faz a correspondência da instância de uma letra.
9xx9	\d[a-z][a-z]\d	Corresponde a qualquer número seguido de duas letras e outro número, como em 1ab2.

Converter Sintaxe SQL em Sintaxe de Expressão Regular compatível com perl

Se estiver familiarizado com a sintaxe SQL, você poderá usar as informações a seguir para gravar expressões regulares compatíveis com perl.

A seguinte tabela mostra exemplos de sintaxe SQL e seus equivalentes perl:

Sintaxe SQL	Sintaxe perl	Descrição
%	. *	Corresponde a qualquer cadeia.
A%	A.*	Combina a letra "A" seguida por qualquer cadeia, como em Astro.
_	. (um ponto)	Faz a correspondência de qualquer caractere.
A_	A.	Combina "A" seguida por qualquer caractere, como AZ.

Valor de Retorno

Retorna o valor do sub-padrão *n*th que é parte do valor de entrada. O sub-padrão *nth* é baseado no valor que você especifica para subPatternNum.

NULL se a entrada for um valor nulo ou se o padrão for nulo.

Exemplo

Você pode usar REG_EXTRACT em uma expressão para extrair nomes do meio de uma expressão regular que corresponde ao primeiro nome, ao nome do meio e ao sobrenome. Por exemplo, a seguinte expressão retorna o nome do meio de uma expressão 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

Retorna se um valor corresponde a um padrão de expressão regular. Isso permite que você valide padrões de dados, como IDs, números de telefone, CEPs e nomes de estado.

Nota: Use a função REG_REPLACE para substituir um padrão de caractere em uma string com um novo padrão de caractere.

Sintaxe

```
REG_MATCH( subject, pattern )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
assunto	Requerido	Tipo de dados strings. Passa o valor que você deseja comparar em relação ao padrão de expressão regular.
padrão	Requerido	Tipo de dados strings. Padrão de expressão regular que você deseja corresponder. É necessário usar a sintaxe de expressão regular compatível com perl. Coloque o padrão entre aspas simples. Para obter mais informações, consulte "REG_EXTRACT" na página 156.

Valor de Retorno

TRUE se os dados corresponderem ao padrão.

FALSE se os dados não corresponderem ao padrão.

NULL se a entrada for um valor nulo ou se o padrão for NULL.

Exemplo

Você pode usar REG_MATCH em uma expressão para validar números de telefone. Por exemplo, a seguinte expressão corresponde a um número de telefone de 10 dígitos em relação ao padrão e retorna um valor Booleano baseado na correspondência:

```
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

Dica

Você também pode usar REG_MATCH para as seguintes tarefas:

- Para verificar se um valor corresponde a um padrão. Esse uso é semelhante à função SQL LIKE.
- Para verificar se os valores são caracteres. Esse uso é semelhante à função SQL IS_CHAR.

Para verificar se um valor corresponde a um padrão, use um ponto final (.) e um asterisco (*) com a função REG_MATCH em uma expressão. Um ponto final corresponde a qualquer caractere. Um asterisco corresponde a 0 ou mais instâncias de valores que são fornecidos após ele.

Por exemplo, use a seguinte expressão para encontrar números de conta que comecem com 1835:

```
REG_MATCH(ACCOUNT_NUMBER, '1835.*')
```

Para verificar se os valores são caracteres, use uma função REG_MATCH com a expressão regular [a-zA-Z]+. a-z corresponde a todos os caracteres minúsculos. A-Z corresponde a todos os caracteres maiúsculos. O sinal de adição (+) indica que deve haver pelo menos um caractere.

Por exemplo, use a seguinte expressão para verificar se uma lista de sobrenomes contém apenas caracteres:

```
REG_MATCH(LAST_NAME, '[a-zA-Z]+')
```

REG_REPLACE

Substitui caracteres de uma string por caracteres de outro padrão. Por padrão, REG_REPLACE pesquisa a string de entrada do padrão de caractere especificado e substitui todas as ocorrências pelo padrão de substituição. Você também pode indicar o número de ocorrências do padrão que deseja substituir na string.

Sintaxe

```
REG_REPLACE( subject, pattern, replace, numReplacements )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
assunto	Requerido	Tipo de dados strings. Passa as strings que você deseja pesquisar.
padrão	Requerido	Tipo de dados strings. Passa a string de caracteres a ser substituída. É necessário usar a sintaxe de expressão regular compatível com perl. Coloque o padrão entre aspas simples. Para obter mais informações, consulte "REG_EXTRACT" na página 156.
substituir	Requerido	Tipo de dados strings. Passa a nova string de caracteres.
numReplacements	Opcional	Tipo de dados Numérico. Especifica o número de ocorrências que você deseja substituir. Se você omitir essa opção, REG_REPLACE substituirá todas as ocorrências da string de caracteres.

Valor de Retorno

String

Exemplo

A seguinte expressão remove espaços adicionais dos dados do nome do Funcionário para cada linha da porta 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

Substitui caracteres em uma cadeia por um único caractere ou nenhum caractere. REPLACECHR pesquisa a cadeia de entrada para os caracteres que você especifica e substitui todas as ocorrências de todos os caracteres pelo novo caractere especificado.

Sintaxe

```
REPLACECHR( CaseFlag, InputString, OldCharSet, NewChar )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
CaseFlag	Obrigatório	Deve ser um inteiro. Determina se os argumentos nessa função fazem distinção entre maiúsculas e minúsculas. Você pode inserir qualquer expressão válida de transformação. Quando *CaseFlag* é um número diferente de 0, a função faz distinção entre maiúsculas e minúsculas. Quando *CaseFlag* é um valor nulo ou 0, a função não faz distinção entre maiúsculas e minúsculas.
InputString	Obrigatório	Deve ser uma cadeia de caracteres. Passa as cadeias que você deseja pesquisar. Você pode inserir qualquer expressão válida de transformação. Se você passar um valor numérico, a função o converterá em uma cadeia de caracteres. Se *InputString* for NULL, REPLACECHR retornará NULL.
OldCharSet	Obrigatório	Deve ser uma cadeia de caracteres. Os caracteres que você deseja substituir. Você pode inserir um ou mais caracteres. Você pode inserir qualquer expressão válida de transformação. Você também pode inserir um literal de texto entre aspas simples, por exemplo, 'abc'. Se você passar um valor numérico, a função o converterá em uma cadeia de caracteres. Se *OldCharSet* for NULL ou vazio, REPLACECHR retornará *InputString*.
NewChar	Obrigatório	Deve ser uma cadeia de caracteres. Você pode inserir um caractere, uma cadeia vazia ou NULL. Você pode inserir qualquer expressão válida de transformação. Se *NewChar* for NULL ou vazio, REPLACECHR removerá todas as ocorrências de todos os caracteres em *OldCharSet* em *InputString*. Se *NewChar* contiver mais de um caractere, REPLACECHR usará o primeiro caractere para substituir *OldCharSet*.

Valor de Retorno

String.

String vazia se REPLACECHR remover todos os caracteres em *InputString*.

NULL se *InputString* for NULL.

InputString se *OldCharSet* for NULL ou vazio.

Exemplos

A seguinte expressão remove as aspas duplas dos dados de log da Web de cada linha na porta 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

A seguinte expressão remove vários caracteres de cada linha na porta 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
[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

A seguinte expressão altera parte do valor do código do cliente em cada linha na porta CUSTOMER_CODE:

```
REPLACECHR ( 1, CUSTOMER_CODE, 'A', 'M' )
```

CUSTOMER_CODE	RETURN VALUE
ABA	MBM
abA	abM
BBC	BBC
ACC	MCC
NULL	NULL

A seguinte expressão altera parte do valor do código do cliente em cada linha na porta CUSTOMER_CODE:

```
REPLACECHR ( 0, CUSTOMER_CODE, 'A', 'M' )
```

CUSTOMER_CODE	RETURN VALUE
ABA	MBM
abA	MbM
BBC	BBC
ACC	MCC

A seguinte expressão altera parte do valor do código do cliente em cada linha na porta CUSTOMER_CODE:

```
REPLACECHR ( 1, CUSTOMER_CODE, 'A', NULL )
```

CUSTOMER_CODE	RETURN VALUE
ABA	B
BBC	BBC
ACC	CC

CUSTOMER_CODE	RETURN VALUE
AAA	*[empty string]*
aaa	aaa
NULL	NULL

A seguinte expressão remove vários números de cada linha na porta INPUT:

```
REPLACECHR ( 1, INPUT, '14', NULL )
```

INPUT	RETURN VALUE
12345	235
4141	NULL
111115	5
NULL	NULL

Quando quiser usar uma aspa simples (') em *OldCharSet* ou *NewChar*, você deve usar a função CHR. A aspa simples é o único caractere que não pode ser usado dentro de um literal de string.

A seguinte expressão remove vários caracteres, incluindo a aspa simples, de cada linha na porta INPUT:

```
REPLACECHR (1, INPUT, CHR(39), NULL )
```

INPUT	RETURN VALUE
'Tom Smith' 'Laura Jones'	Tom Smith Laura Jones
Tom's	Toms
NULL	NULL

REPLACESTR

Substitui caracteres em uma string por um único caractere, vários caracteres ou nenhum caractere. REPLACESTR pesquisa a string de entrada para todas as strings que você especifica e as substitui pela nova string especificada.

Sintaxe

```
REPLACESTR ( CaseFlag, InputString, OldString1, [OldString2, ... OldStringN,] NewString )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
CaseFlag	Requerido	Deve ser um inteiro. Determina se os argumentos nessa função fazem distinção entre maiúsculas e minúsculas. Você pode inserir qualquer expressão de transformação válida. Quando *CaseFlag* é um número diferente de 0, a função faz distinção entre maiúsculas e minúsculas. Quando *CaseFlag* é um valor nulo, a função não faz distinção entre maiúsculas e minúsculas.
InputString	Requerido	Deve ser uma string de caractere. Passa os valores que você deseja pesquisar. Você pode inserir qualquer expressão de transformação válida. Se você passar um valor numérico, a função o converterá em uma string de caractere. Se *InputString* estiver como NULL, REPLACESTR retornará o valor NULL.
OldString	Requerido	Deve ser uma string de caractere. A string que você deseja substituir. Você deve inserir pelo menos um argumento *OldString*. Você pode inserir um ou mais caracteres por argumento *OldString*. Você pode inserir qualquer expressão de transformação válida. Você também pode inserir um literal de texto anexado entre aspas simples, por exemplo, 'abc'. Se você passar um valor numérico, a função o converterá em uma string de caractere. Quando REPLACESTR contém vários argumentos *OldString* e um ou mais argumentos *OldString* é NULL ou vazio, REPLACESTR ignora o argumento *OldString*. Quando todos os argumentos *OldString* são NULL ou vazio, REPLACESTR retorna *InputString*. A função substitui os caracteres nos argumentos *OldString* na ordem que eles aparecem na função. Por exemplo, se você inserir vários argumentos *OldString*, o primeiro argumento *OldString* terá precedência sobre o segundo argumento *OldString* e o segundo argumento *OldString* terá precedência sobre o terceiro argumento *OldString*. Quando REPLACESTR substituir uma string, ele coloca o cursor depois dos caracteres substituídos em *InputString* antes de pesquisar a próxima correspondência.
NewString	Requerido	Deve ser uma string de caractere. Você pode inserir um caractere, vários caracteres, uma string vazia ou NULL. Você pode inserir qualquer expressão de transformação válida. Se *NewString* for NULL ou vazio, REPLACESTR removerá todas as ocorrências de *OldString* em *InputString*.

Valor de Retorno

String.

String vazia se REPLACESTR remover todos os caracteres em *InputString*.

NULL se *InputString* for NULL.

InputString se todos os argumentos *OldString* forem NULL ou vazios.

Exemplos

A seguinte expressão remove as aspas duplas e as duas strings de texto diferentes dos dados de log da Web de cada linha na porta 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

A seguinte expressão altera o título de determinados valores de cada linha na porta TITLE:

```
REPLACESTR ( 1, TITLE, 'rs.', 'iss', 's.' )
```

TITLE	RETURN VALUE
Mrs.	Ms.
Miss	Ms.
Mr.	Mr.
MRS.	MRS.

A seguinte expressão altera o título de determinados valores de cada linha na porta TITLE:

```
REPLACESTR ( 0, TITLE, 'rs.', 'iss', 's.' )
```

TITLE	RETURN VALUE
Mrs.	Ms.
MRS.	Ms.

A seguinte expressão mostra como a função REPLACESTR substitui vários argumentos `OldString` de cada linha na porta INPUT:

```
REPLACESTR ( 1, INPUT, 'ab', 'bc', '*' )
```

INPUT	RETURN VALUE
abc	*c
abbc	**
abbbbc	*bb*
bc	*

A seguinte expressão mostra como a função REPLACESTR substitui vários argumentos `OldString` de cada linha na porta INPUT:

```
REPLACESTR ( 1, INPUT, 'ab', 'bc', 'b' )
```

INPUT	RETURN VALUE
ab	b
bc	b
abc	bc
abbc	bb
abbcc	bbc

Quando quiser usar uma aspa simples (') em *OldString* ou *NewString*, você deve usar a função CHR. Use as funções CHR e CONCAT para concatenar uma aspa simples em uma string. A aspa simples é o único caractere que não pode ser usado dentro de um literal de string. Considere o seguinte exemplo:

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

O valor retornado é:

```
Joan's car
```

A seguinte expressão altera uma string que inclui a aspa simples, de cada linha na porta 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

Renomeia cada elemento do valor de estrutura determinado com base nos nomes dos elementos na definição de tipo de dados complexo especificada.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório/ Opcional	Descrição
:Type.type_definition_library.type_definition	Obrigatório	A definição de tipo de dados complexo que representa o esquema dos dados de struct. Use o qualificador de referência `:Type` para referenciar a biblioteca de definições de tipo que contém a definição de tipo de dados complexo.
struct_value	Obrigatório	O valor de struct para o qual você deseja alterar os nomes de elementos. É possível inserir qualquer expressão de transformação válida que seja avaliada como um struct.

O tipo de dados de cada elemento na definição de tipo de dados complexo deve corresponder ao tipo de dados do elemento correspondente do struct.

Valor de retorno

Struct.

Exemplos

A expressão a seguir altera os nomes dos elementos na porta de struct h2_sales com base nos nomes na definição de tipo de dados complexo 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

Reverte a string de entrada.

Sintaxe

```
REVERSE( string )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
string	Requerido	Qualquer valor de caractere. Valor que você deseja reverter.

Valor de Retorno

String. Reversão do valor de entrada.

Exemplo

A seguinte expressão reverte os números do código do cliente:

```
REVERSE( CUSTOMER_CODE )
```

CUSTOMER_CODE	RETURN VALUE
0001	1000
0002	2000
0003	3000
0004	4000

ROUND (Datas)

Arredonda uma parte de uma data. Você também pode usar ROUND para arredondar números.

Essa função pode arredondar as seguintes partes de uma data:

Ano

Arredonda a parte do ano de uma data com base no mês.

Mês

Arredonda a parte do mês de uma data com base no dia do mês.

Dia

Arredonda a parte do dia da data com base na hora.

Hora

Arredonda a parte da hora da data com base nos minutos da hora.

Minuto

Arredonda a parte do minuto da data com base nos segundos.

Segundo

Arredonda a parte do segundo da data com base nos milissegundos.

Milissegundo

Arredonda a parte do milissegundo da data com base nos microssegundos.

Microssegundo

Arredonda a parte do microssegundo da data com base nos nanossegundos.

A seguinte tabela mostra as condições usadas pela expressão ROUND e os valores retornados:

Condição	Expressão	Valor de Retorno
Mês entre janeiro e junho, a função retorna 1º de janeiro do mesmo ano e define a hora como 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
Mês entre julho e dezembro, a função retorna 1º de janeiro do próximo ano e define a hora como 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
Dia do mês entre 1 e 15, a função retorna o primeiro dia do mês de entrada e define a hora como 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
Dia do mês entre 16 e o último dia do mês, a função retorna o primeiro dia do próximo mês e define a hora como 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
Hora entre 00:00:00 (12 a.m.) e 11:59:59 a.m., a função retorna a data atual e define a hora como 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
Hora 12:00:00 (12 p.m.) ou superior, a função arredonda a data para o próximo dia e define a hora como 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
Parte do minuto da hora entre 0 e 29 minutos, a função retorna a hora atual e define os minutos, segundos, millissegundos e nanossegundos como 0.	`ROUND(TO_DATE('04/01/1998 11:29:35','MM/DD/YYYY HH24:MI:SS'),'HH')`	04/01/1998 11:00:00.000000000
Parte do minuto da hora 30 ou superior, a função retorna a próxima hora e define os minutos, segundos, millissegundos e nanossegundos como 0.	`ROUND(TO_DATE('04/01/1998 13:39:00','MM/DD/YYYY HH24:MI:SS'),'HH')`	04/01/1998 14:00:00.000000000
Hora entre 0 e 29 segundos, a função retorna o minuto atual e define segundos, milissegundos e nanossegundos como 0.	`ROUND(TO_DATE('05/22/1998 10:15:29','MM/DD/YYYY HH24:MI:SS'),'MI')`	05/22/1998 10:15:00.000000000
Hora entre 30 e 59 segundos, a função retorna o próximo minuto e define segundos, milissegundos e nanossegundos como 0.	`ROUND(TO_DATE('05/22/1998 10:15:30','MM/DD/YYYY HH24:MI:SS'),'MI')`	05/22/1998 10:16:00.000000000
Hora entre 0 e 499 milissegundos, a função retorna o segundo atual e define os milissegundos como 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
Hora entre 500 e 999 milissegundos, a função retorna o próximo segundo e define os milissegundos como 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
Hora entre 0 e 499 microssegundos, a função retorna o milissegundo atual e define os microssegundos como 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

Condição	Expressão	Valor de Retorno
Hora entre 500 e 999 microssegundos, a função retorna o próximo milissegundo e define os microssegundos como 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
Hora entre 0 e 499 nanossegundos, a função retorna o microssegundo atual e define os nanossegundos como 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
Hora entre 500 e 999 nanossegundos, a função retorna o próximo microssegundo e define os nanossegundos como 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

Sintaxe

```
ROUND( date [,format] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
data	Obrigatório	Tipo de dados Data/Hora. Você pode aninhar TO_DATE para converter cadeias em datas antes do arredondamento.
formato	Opcional	Insira uma cadeia de formato válida. Essa é a parte da data que você deseja arredondar. Você pode arredondar apenas uma parte da data. Se você omitir a cadeia de formato, a função arredondará a data para o dia mais próximo.

Valor de Retorno

Data com a parte especificada arredondada. ROUND retorna uma data no mesmo formato que a data de origem. Você pode conectar os resultados dessa função a qualquer porta com um tipo de dados de data e hora.

NULL se você passar um valor nulo para a função.

Exemplos

As seguintes expressões arredondam a parte do ano de dados na porta 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
Apr 19 1998 1:31:20PM	Jan 1 1998 12:00:00.000000000AM

DATE_SHIPPED	RETURN VALUE
Dec 20 1998 3:29:55PM	Jan 1 1999 12:00:00.000000000AM
NULL	NULL

As seguintes expressões arredondam a parte do mês de cada data na porta 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

As seguintes expressões arredondam a parte do dia de cada data na porta 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

As seguintes expressões arredondam a parte da hora de cada data na porta 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

As seguintes expressões arredondam a parte do minuto de cada data na porta 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)

Arredonda números para um número especificado de dígitos ou casas decimais. Você também pode usar ROUND para arredondar datas.

Sintaxe

```
ROUND( numeric_value [, precision] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
valor_numérico	Obrigatório	Tipo de dados Numérico. Você pode inserir qualquer expressão válida de transformação. Use operadores para realizar operações aritméticas antes de arredondar os valores.
precisão	Opcional	Inteiro positivo ou negativo. Se você inserir uma *precisão* positiva, a função arredondará para esse número de casas decimais. Por exemplo, ROUND(12,99, 1) retornará 13,0 e ROUND(15,44, 1) retornará 15,4. Se você inserir uma *precisão* negativa, a função arredondará esse número de dígitos para a esquerda do ponto decimal, retornando um inteiro. Por exemplo, ROUND(12,99, -1) retornará 10 e ROUND(15.99, -1) retornará 20. Se você inserir uma *precisão* decimal, a função arredondará para o inteiro mais próximo antes de avaliar a expressão. Por exemplo, ROUND(12,99, 0,8) retornará 13,0 porque a função arredonda 0,8 para 1 e, em seguida, avalia a expressão. Se você omitir o argumento de *precisão*, a função arredondará para o inteiro mais próximo, truncando a parte decimal do número. Por exemplo, ROUND(12,99) retorna 13.

Valor de Retorno

Valor numérico.

Se um dos argumentos for NULL, ROUND retornará NULL.

Nota: Se o valor retornado for Decimal com precisão superior a 15, habilite a alta precisão para garantir precisão decimal de até 28 dígitos.

Exemplos

A expressão a seguir retorna os valores da porta Preço arredondados para três casas decimais.

```
ROUND( PRICE, 3 )
```

PRICE	RETURN VALUE
12.9936	12.994
15.9949	15.995
-18.8678	-18.868
56.9561	56.956
NULL	NULL

Você pode arredondar dígitos à esquerda da vírgula decimal passando um inteiro negativo no argumento *precisão*:

```
ROUND( PRICE, -2 )
```

PRICE	RETURN VALUE
13242.99	13200.0

PRICE	RETURN VALUE
1435.99	1400.0
-108.95	-100.0
NULL	NULL

Se você passar um valor decimal no argumento *precisão*, o Data Integration Service o arredondará para o inteiro mais próximo antes de avaliar a expressão:

```
ROUND( PRICE, 0.8 )
```

PRICE	RETURN VALUE
12.99	13.0
56.34	56.3
NULL	NULL

Se você omitir o argumento *precisão*, a função arredondará para o inteiro mais próximo:

```
ROUND( PRICE )
```

PRICE	RETURN VALUE
12.99	13.0
-15.99	-16.0
-18.99	-19.0
56.95	57.0
NULL	NULL

Dica

Você também pode usar ROUND para definir explicitamente a precisão de valores calculados e obter os resultados esperados. Quando executado em modo de baixa precisão, o Data Integration Service truncará o resultado dos cálculos se a precisão do valor exceder 15 dígitos. Por exemplo, convém processar a seguinte expressão em modo de baixa precisão:

```
7/3 * 3 = 7
```

Nesse caso, o Data Integration Service avalia o lado esquerdo da expressão como 6,999999999999999 porque o resultado da primeira operação de divisão é truncado. O Data Integration Service avalia toda a expressão como FALSE. Esse pode não ser o resultado esperado.

Para obter o resultado esperado, use ROUND para arredondar o resultado truncado do lado esquerdo da expressão para o resultado esperado. O Data Integration Service avalia a seguinte expressão como TRUE:

```
ROUND(7/3 * 3) = 7
```

RPAD

Converte uma string em um tamanho especificado ao adicionar espaços em branco ou caracteres ao final da string.

Sintaxe

```
RPAD( first_string, length [,second_string] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
first_string	Requerido	Qualquer valor de string. As strings que você deseja alterar. Você pode inserir qualquer expressão de transformação válida.
length	Requerido	Deve ser um número literal inteiro positivo. Especifica o tamanho que você deseja que cada string tenha.
second_string	Opcional	Qualquer valor de string. Passa a string que você deseja anexar ao lado direito dos valores *first_string*. Anexe os caracteres que você deseja adicionar ao final da string entre aspas simples, por exemplo, 'abc'. Esse argumento faz distinção entre maiúsculas e minúsculas. Se você omitir a segunda string, a função preencherá o final da primeira string com espaços em branco.

Valor de Retorno

String do tamanho especificado.

NULL se um valor passado para a função for NULL ou se o tamanho for um número negativo.

Exemplos

A expressão a seguir retorna o nome de item com um tamanho de 16 caracteres, anexando a string '.' ao final de cada nome de item:

```
RPAD( ITEM_NAME, 16, '.')
```

ITEM_NAME	RETURN VALUE
Flashlight	Flashlight......
Compass	Compass.........
Regulator System	Regulator System
Safety Knife	Safety Knife....

RPAD conta o tamanho da esquerda para a direita. Então, se a primeira string for maior que o tamanho, RPAD truncará a string da direita para a esquerda. Por exemplo, RPAD('alphabetical', 5, 'x') retornará a string 'alpha'. RPAD usa uma parte parcial da *second_string* quando necessário.

A expressão a seguir retorna o nome de item com um tamanho de 16 caracteres, anexando a string '*.*' ao final de cada nome de item:

```
RPAD( ITEM_NAME, 16, '*..*' )
```

ITEM_NAME	RETURN VALUE
Flashlight	Flashlight*..**.
Compass	Compass*..**..**
Regulator System	Regulator System
Safety Knife	Safety Knife*..*

RTRIM

Remove espaços em branco ou caracteres do final de uma string.

Se você não especificar um parâmetro *trim_set* na expressão:

- No modo UNICODE, RTRIM remove espaços de byte único e duplo do final de uma string.
- No modo ASCII, RTRIM remove apenas espaços de byte único.

Se você usar RTRIM para remover caracteres de uma string, RTRIM compara o *trim_set* a cada caractere no argumento *string*, caractere-por-caractere, a partir do lado direito da string. Se o caractere na string corresponder a qualquer caractere no *trim_set*, RTRIM o removerá. RTRIM continua a comparar e remover caracteres até que ele não consiga encontrar um caractere correspondente no *trim_set*. Ele retorna a string sem os caracteres correspondentes.

Sintaxe

```
RTRIM( string [, trim_set] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
string	Requerido	Qualquer valor de string. Passa os valores que você deseja cortar. Você pode inserir qualquer expressão de transformação válida. Use operadores para executar comparações ou concatenar string antes de remover espaços em branco a partir do final de uma string.
trim_set	Opcional	Qualquer valor de string. Passa os caracteres que você deseja remover a partir do final da string. Você também pode inserir um literal de texto. No entanto, você deve anexar os caracteres que deseja remover do final da string entre aspas simples, por exemplo, 'abc'. Se você omitir a segunda string, a função removerá espaços em branco no final da primeira string. RTRIM diferencia maiúsculas de minúsculas.

Valor de Retorno

String. Os valores de string com os caracteres especificados no argumento *trim_set* removido.

NULL se um valor transmitido para a função for NULL.

Exemplo

A seguinte expressão remove os caracteres ‘re’ das strings na porta 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 removerá ‘e’ da Página mesmo se ‘r’ for o primeiro caractere no *trim_set*. Isso porque RTRIM pesquisa, caractere-por-caractere, o conjunto de caracteres que você especifica no argumento *trim_set*. Se o último caractere na string corresponder ao primeiro caractere no *trim_set*, RTRIM o removerá. Porém, se o último caractere na string não corresponder, RTRIM comparará o segundo caractere no *trim_set*. Se o segundo do último caractere na string corresponder ao segundo caractere no *trim_set*, RTRIM o removerá. Quando o caractere na string não corresponder ao *trim_set*, RTRIM retornará a string e avaliará a próxima linha.

No último exemplo, o último caractere em Nelson não corresponde a nenhum caractere no argumento *trim_set*, assim RTRIM retorna a string 'Nelson' e avalia a próxima linha.

Dicas para RTRIM

Use RTRIM e LTRIM com || ou CONCAT para remover espaços em branco à direita e à esquerda depois de concatenar duas strings.

Você também pode remover vários conjuntos de caracteres ao aninhar RTRIM. Por exemplo, se quiser remover espaços em branco à esquerda e o caractere 't' do final de cada string em uma coluna de nomes, você poderá criar uma expressão semelhante ao seguinte:

```
RTRIM( RTRIM( NAMES ), 't' )
```

SET_DATE_PART

Define uma parte de um valor de Data/Hora para um valor que você especifica. Com SET_DATE_PART, você pode alterar as seguintes partes de uma data:

- **Ano.** Altere o ano inserindo um inteiro positivo no argumento de *valor*. Use qualquer string de formato de ano: Y, YY, YYY ou YYYY para definir o ano. Por exemplo, a seguinte expressão muda o ano para 2001 em todas as datas na porta SHIP_DATE:

  ```
  SET_DATE_PART( SHIP_DATE, 'YY', 2001 )
  ```

- **Mês.** Altere o mês inserindo um inteiro positivo entre 1 e 12 (Janeiro = 1 e Dezembro = 12) no argumento *valor*. Use qualquer string de formato de ano: MM, MON, MONTH para definir o mês. Por exemplo, a seguinte expressão muda o mês para outubro em todas as datas na porta SHIP_DATE:

  ```
  SET_DATE_PART( SHIP_DATE, 'MONTH', 10 )
  ```

- **Dia.** Altere o dia inserindo um inteiro positivo entre 1 e 31 (com exceção dos meses que têm menos de 31 dias: fevereiro, abril, junho, setembro e novembro) no argumento *value*. Use qualquer uma das strings de formato de mês (D, DD, DDD, DY e DAY) para definir o dia. Por exemplo, a seguinte expressão muda o dia para 10 em todas as datas na porta SHIP_DATE:

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

- **Hora.** Altere a hora inserindo um inteiro positivo entre 0 e 24 (em que 0 = 12AM, 12 = 12PM e 24 = 12AM) no argumento *valor*. Use qualquer uma das strings de formato de hora (HH, HH12, HH24) para definir a hora. Por exemplo, a seguinte expressão muda a hora para 14:00:00 (ou 2:00:00PM) em todas as datas na porta SHIP_DATE:

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

- **Minuto.** Altere os minutos inserindo um inteiro positivo entre 0 e 59 no argumento *valor*. Use a string de formato MI para definir o minuto. Por exemplo, a seguinte expressão muda o minuto para 25 em todas as datas na porta SHIP_DATE:

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

- **segundos.** Altere os segundos inserindo um inteiro positivo entre 0 e 59 no argumento *valor*. Use a string de formato SS para definir o segundo. Por exemplo, a seguinte expressão muda o segundo para 59 em todas as datas na porta SHIP_DATE:

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

- **Milissegundos**. Altere os milissegundos inserindo um inteiro positivo entre 0 e 999 no argumento *valor*. Use a string de formato MS para definir os milissegundos. Por exemplo, a seguinte expressão muda os milissegundos para 125 em todas as datas na porta SHIP_DATE:

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

- **Microssegundos.** Altere os microssegundos inserindo um inteiro positivo entre 1000 e 999999 no argumento *value*. Use a string de formato US para definir os microssegundos. Por exemplo, a seguinte expressão muda os microssegundos para 12555 em todas as datas na porta SHIP_DATE:

  ```
  SET_DATE_PART( SHIP_DATE, 'US', 12555 )
  ```

- **Nanossegundos.** Altere os segundos inserindo um inteiro positivo entre 1000000 e 999999999 no argumento *valor*. Use a string de formato NS para definir os nanossegundos. Por exemplo, a seguinte expressão muda os nanossegundos para `12555555` em todas as datas na porta SHIP_DATE:

  ```
  SET_DATE_PART( SHIP_DATE, 'NS', 12555555 )
  ```

Sintaxe

```
SET_DATE_PART( date, format, value )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
data	Requerido	Tipo de dados Data/Hora. A data que você deseja modificar. Você pode inserir qualquer expressão de transformação válida.
formato	Requerido	String de formato que especifica a porção da data a ser alterada. A string de formato não faz distinção entre maiúsculas e minúsculas.
valor	Requerido	Um valor inteiro positivo atribuído à parte especificada da data. O inteiro deve ser um valor válido da parte da data que você deseja alterar. Se você inserir um valor inadequado, como 30 de fevereiro, a sessão falhará.

Valor de Retorno

Data no mesmo formato que a data de origem com a parte especificada alterada.

NULL se um valor transmitido para a função for NULL.

Exemplos

As seguintes expressões alteram a hora para 4PM em cada data na porta 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

As seguintes expressões alteram o mês para Junho para as datas na porta DATE_PROMISED: O Data Integration Service exibe um erro quando você tenta criar uma data que não existe, como alterar 31 de março para 31 de junho:

```
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	*Error. Integration Service doesn't write row.*

DATE_PROMISED	RETURN VALUE
Dec 12 1997 8:07:33AM	Jun 12 1997 8:07:33AM
NULL	NULL

As seguintes expressões alteram o ano para 2000 para as datas na porta 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

Dica

Se quiser alterar várias partes de uma data simultaneamente, você poderá aninhar várias funções SET_DATE_PART dentro do argumento *data*. Por exemplo, você pode gravar a seguinte expressão para alterar todas as datas na porta DATE_ENTERED para 1º de julho de 1998:

```
SET_DATE_PART( SET_DATE_PART( SET_DATE_PART( DATE_ENTERED, 'YYYY', 1998),MM', 7), 'DD',
1)
```

SIGN

Retorna se um valor numérico é positivo, negativo ou 0.

Sintaxe

```
SIGN( numeric_value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
valor_numérico	Requerido	Valor numérico. Passa os valores que você deseja avaliar. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

-1 para valores negativos.

0 para 0.

1 para valores positivos.

NULL se NULL.

Exemplo

A seguinte expressão determina se a porta SALES inclui algum valor negativo:

```
SIGN( SALES )
```

SALES	RETURN VALUE
100	1
-25.99	-1
0	0
NULL	NULL

SIN

Retorna o seno de um valor numérico (expresso em radianos).

Sintaxe

```
SIN( numeric_value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
valor_numérico	Requerido	Tipo de dados Numérico. Dados numéricos expressos em radianos (graus multiplicados por pi divididos por 180). Passa os valores para os quais você deseja calcular o seno. Você pode inserir qualquer expressão de transformação válida. Você também pode usar operadores para converter um valor numérico em radianos ou realizar uma operação aritmética dentro do cálculo SIN.

Valor de Retorno

Valor duplo.

NULL se um valor transmitido para a função for NULL.

Exemplo

A seguinte expressão converte os valores na porta Graus em radianos e, em seguida, calcula o seno de cada radiano:

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

DEGREES	RETURN VALUE
0	0
90	1

DEGREES	RETURN VALUE
70	0.939692620785936
30	0.50000000000003
5	0.0871557427476639
89	0.999847695156393
NULL	NULL

Você pode realizar operação aritmética nos valores passados para SIN antes que a função calcule o seno. Por exemplo:

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

SINH

Retorna o seno hiperbólico de um valor numérico.

Sintaxe

```
SINH( numeric_value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
valor_numérico	Requerido	Tipo de dados Numérico. Dados numéricos expressos em radianos (graus multiplicados por pi divididos por 180). Passa os valores para os quais você deseja calcular o seno hiperbólico. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Valor duplo.

NULL se um valor transmitido para a função for NULL.

Exemplo

A seguinte expressão retorna o seno hiperbólico para os valores na porta Ângulos:

```
SINH( ANGLES )
```

ANGLES	RETURN VALUE
1.0	1.1752011936438
2.897	9.03225804884884
3.66	19.4178051793031

ANGLES	RETURN VALUE
5.45	116.376934801486
0	0.0
0.345	0.35188478309993
NULL	NULL

Dica

Você pode realizar operação aritmética nos valores passados para SINH antes que a função calcule o seno hiperbólico. Por exemplo:

```
SINH( MEASURES.ARCS / 180 )
```

SIZE

Retorna o tamanho do array.

Sintaxe

```
SIZE(array_value)
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório/Opcional	Descrição
array_value	Obrigatório	Tipo de dados array. O array para o qual você deseja determinar o tamanho.

Valor de retorno

Int.

Retornará -1 se o array for NULL.

Exemplos

A expressão a seguir retorna o tamanho do array 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

SOUNDEX

Codifica um valor de string em uma string de quatro caracteres.

SOUNDEX funciona em caracteres no alfabeto em inglês (A-Z). Ele usa o primeiro caractere da string de entrada como o primeiro caractere no valor retornado e codifica as três consoantes exclusivas remanescentes como números.

SOUNDEX codifica caracteres de acordo com a seguinte lista de regras:

- Usa o primeiro caractere na *string* como o primeiro caractere no valor retornado e o codifica em letra maiúscula. Por exemplo, SOUNDEX('John') e SOUNDEX('john') retornam 'J500'.
- Codifica as três primeiras consoantes exclusivas após o primeiro caractere na *string* e ignora o restante. Por exemplo, SOUNDEX('JohnRB') e SOUNDEX('JohnRBCD') retornam 'J561'.
- Atribui um código único a consoantes que soam de forma semelhante.

  A seguinte tabela lista diretrizes de codificação SOUNDEX para consoantes:

  Tabela 2. Diretrizes de Codificação SOUNDEX para consoantes

Código	Consoante
1	B, P, F, V
2	C, S, G, J, K, Q, X, Z
3	D, T
4	L
5	M, N
6	R

- Ignora os caracteres A, E, I, O, U, H e W, a menos que um deles seja o primeiro caracteres na *string*. Por exemplo, SOUNDEX('A123') retorna 'A000' e SOUNDEX('MAeiouhwC') retorna 'M200'.
- Se *string* produz menos de quatro caracteres, SOUNDEX preenche a string resultante com zeros. Por exemplo, SOUNDEX('J') retorna 'J000'.
- Se a *string* contiver um conjunto de consoantes consecutivas que usam o mesmo código listado em "SOUNDEX" na página 185, SOUNDEX codificará a primeira ocorrência e ignorará as ocorrências remanescentes no conjunto. Por exemplo, SOUNDEX('AbbpdMN') retorna 'A135'.
- Ignora números na *string*. Por exemplo, SOUNDEX('Joh12n') e SOUNDEX('1John') retornam 'J500'.
- Retornará NULL se *string* for NULL ou se todos os caracteres na *string* não forem letras do alfabeto em inglês.

Sintaxe

```
SOUNDEX( string )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Obrigatório / Opcional	Descrição
string	Obrigatório	String de caracteres. Passa o valor de string que você deseja codificar. Você pode inserir qualquer expressão válida de transformação.

Valor de Retorno

Cadeia.

NULL se uma das seguintes condições for verdadeira:

- Se o valor passado para a função for NULL.
- Nenhum caractere na *string* é uma letra do alfabeto inglês.
- *string* está vazia.

Exemplo

A seguinte expressão codifica os valores na porta EMPLOYEE_NAME:

```
SOUNDEX( EMPLOYEE_NAME )
```

EMPLOYEE_NAME	RETURN VALUE
John	J500
William	W450
jane	J500
joh12n	J500
1abc	A120
NULL	NULL

SQL_LIKE

Retorna se um valor corresponde a um padrão de expressão regular. Isso permite que você valide padrões de dados, como IDs, números de telefone, CEPs e nomes de estados.

Sintaxe

```
SQL_LIKE(subject, pattern, escape character)
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório/ Opcional	Descrição
subject	Obrigatório	Tipo de dados de string. Transmite o valor que você deseja comparar com a expressão regular. Coloque o valor entre aspas simples.
pattern	Obrigatório	Tipo de dados de string. Expressão regular que você deseja corresponder. Coloque o padrão entre aspas simples.
escape character	Opcional	Tipo de dados de string. A função SQL_LIKE suporta o sinal de porcentagem (%) e sublinhado (_) como caracteres de escape. Coloque o caractere de escape entre aspas simples.

Valor de Retorno

TRUE se os dados corresponderem ao padrão.

FALSE se os dados não corresponderem ao padrão.

NULL se a entrada for um valor nulo ou se o padrão for NULL.

Exemplo

Você pode usar SQL_LIKE em uma expressão para localizar os nomes que correspondem a um padrão. Por exemplo, a seguinte expressão corresponde aos nomes para o padrão "A_#%" com o caractere de escape '#':

```
SQL_LIKE(ENAME, 'A_#%', '#')
```

ENAME	Valor
SMITH	FALSE
AX%	TRUE
MILLER	FALSE
A%	FALSE
JONES	FALSE
BLAKE	FALSE
A%l	FALSE

SQRT

Retorna a raiz quadrada de um valor numérico não negativo.

Sintaxe

```
SQRT( numeric_value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
valor_numérico	Requerido	Valor numérico positivo. Passa os valores para os quais você deseja calcular uma raiz quadrada. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Valor duplo.

NULL se um valor transmitido para a função for NULL.

Exemplo

A seguinte expressão retorna a raiz quadrada dos valores na porta Números:

```
SQRT( NUMBERS )
```

NUMBERS	RETURN VALUE
100	10
-100	*Error. Data Integration Service does not write row.*
NULL	NULL
60.54	7.78074546557076

O valor -100 resulta em erro, pois a função SQRT só avalia valores numéricos positivos. Se você passar um valor negativo ou de caractere, o Data Integration Service exibirá um Erro de Avaliação de Transformação e não gravará a linha.

Você pode realizar operação aritmética nos valores passados para SQRT antes que a função calcule a raiz quadrada.

STDDEV

Retorna o desvio padrão dos valores numéricos que você passa para essa função. STDDEV é usado para analisar dados estatísticos. Você pode aninhar apenas uma outra função de agregação dentro de STDDEV, e a função aninhada deverá retornar um tipo de dados Numérico.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
valor_numérico	Requerido	Tipos de dados numéricos. Essa função passa os valores para os quais você deseja calcular um desvio padrão ou os resultados de uma função. Você pode inserir qualquer expressão de transformação válida. Você pode usar operadores para calcular a média de valores em portas diferentes.
filter_condition	Opcional	Limita as linhas na pesquisa. A condição de filtro deve ser um valor numérico ou avaliada como TRUE, FALSE ou NULL. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Valor numérico.

NULL se todos os valores passados para a função forem NULL ou se nenhuma linha for selecionada (por exemplo, a condição de filtro avaliada como FALSE ou NULL em todas as linhas).

Nota: Se o valor retornado for Decimal com precisão superior a 15, habilite a alta precisão para garantir precisão decimal de até 28 dígitos.

Nulls

Se um único valor for NULL, STDDEV irá ignorá-lo. No entanto, se todos os valores forem NULL, STDDEV retornará NULL.

Agrupar por

STDDEV agrupa valores baseados em grupos por portas que você define na transformação, retornando um resultado para cada grupo.

Se não houver grupo por porta, STDDEV tratará todas as linhas como um grupo, retornando um valor.

Exemplos

A seguinte expressão calcula o desvio padrão de todas as linhas com valores superiores a US$ 2.000 na porta 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

A função não inclui os valores 1010.90 e 153.88 no cálculo porque a *condição_de_filtro* especifica as vendas com valores superiores a US$ 2.000.

A seguinte expressão calcula o desvio padrão de todas as linhas na porta SALES:

```
STDDEV(SALES)
```

```
SALES

2198.0

2198.0

2198.0

2198.0

RETURN VALUE: 0
```

O valor retornado é 0 porque todas as linhas contêm o mesmo número (não existe desvio padrão). Se não existir nenhum desvio padrão, o valor retornado será 0.

STRUCT

Gera uma estrutura com nomes de elementos e tipos de dados baseados nos argumentos especificados.

Sintaxe

```
STRUCT(element_name1, value1 as any [, element_name2, value2 as any] ...)
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório/Opcional	Descrição
element_name1	Obrigatório	O nome do elemento de struct.
value1	Obrigatório	Qualquer tipo de dados. O valor do elemento de struct.

Se você usar a função STRUCT em uma expressão de saída para uma porta de struct, os argumentos da função deverão corresponder ao tipo de dados dos elementos na definição de tipo de dados complexo.

Valor de retorno

Struct.

Exemplos

A expressão a seguir gera um struct.

```
STRUCT(city , 'New York', state, 'NY')
```

RETURN VALUE

{
city:New York
state:NY
}

A expressão a seguir gera um struct para uma porta de saída de struct com uma definição de tipo de dados complexo **adrs_typedef**:

```
STRUCT(city, cust_city, state, cust_state)
```

A definição de tipo de dados complexo **adrs_typedef** é especificada na biblioteca de definições de tipo da seguinte maneira:

```
adrs_typedef{
    city : string
    state : string
}
```

cust_city	cust_state	RETURN VALUE

NEWYORK	NY	{
		city:NEWYORK
		state:NY
		}

REDWOOD CITY	CA	{
		city:REDWOOD CITY
		state:CA
		}

STRUCT_AS

Gera uma estrutura com um esquema com base na definição de tipo de dados complexo especificada e nos valores que você transmite como argumentos.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório/ Opcional	Descrição
:Type.type_definition_library.type_definition	Obrigatório	A definição de tipo de dados complexo que representa o esquema dos dados de struct. Use o qualificador de referência `:Type` para referenciar a biblioteca de definições de tipo que contém a definição de tipo de dados complexo.
struct_value	Obrigatório	Valor para cada elemento na definição de tipos de dados complexos, separado por vírgula.

Valor de retorno

Struct.

Exemplos

A expressão a seguir gera um struct com base na definição de tipo de dados complexo especificada h1_address_def, com os valores que você transmite como argumentos para os elementos de struct.

```
STRUCT_AS (:Type.type_definition_library.h1_address_def, City, State, ZIP)
```

A definição de tipo de dados complexo **h1_address_def** é especificada na biblioteca de definições de tipo da seguinte maneira:

```
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

Retorna uma parte de uma string. A função SUBSTR conta todos os caracteres, inclusive espaços em branco, iniciando pelo começo da string.

Sintaxe

```
SUBSTR( string, start [,length] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
string	Requerido	Deve ser uma string de caractere. Passa os valores que você deseja pesquisar. Você pode inserir qualquer expressão de transformação válida. Se você passar um valor numérico, a função o converterá em uma string de caractere.
início	Requerido	Deve ser um inteiro. A posição na string onde você deseja começar a contagem. Você pode inserir qualquer expressão de transformação válida. Se a posição inicial for um número positivo, SUBSTR localizará a posição inicial contando a partir do começo da strings. Se a posição inicial for um número negativo, SUBSTR localizará a posição inicial contando a partir do final da string. Se a posição inicial for 0, SUBSTR pesquisará a partir do primeiro caractere da string.
length	Opcional	Deve ser um inteiro maior que 0. O número de caracteres que você deseja que SUBSTR retorne. Você pode inserir qualquer expressão de transformação válida. Se você omitir o argumento length, SUBSTR retornará todos os caracteres da posição inicial ao fim da strings. Se você transmitir um inteiro negativo ou 0, a função retornará uma string vazia. Se você passar um decimal, a função o arredondará para o valor inteiro mais próximo.

Valor de Retorno

String.

String vazia se você passar um valor de comprimento negativo ou 0.

NULL se um valor transmitido para a função for NULL.

Exemplos

As seguintes expressões retornam o código de área de cada linha na porta Telefone:

```
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

As seguintes expressões retornam o número de telefone sem o código de área de cada linha na porta Telefone:

```
SUBSTR( PHONE, 5, 8 )
```

PHONE	RETURN VALUE
808-555-0269	555-0269
809-555-3915	555-3915
357-687-6708	687-6708
NULL	NULL

Você também pode passar um valor inicial negativo para retornar o número de telefone de cada linha na porta Telefone. A expressão ainda lê a string de origem da esquerda para a direita ao retornar o resultado do argumento *comprimento*:

```
SUBSTR( PHONE, -8, 3 )
```

PHONE	RETURN VALUE
808-555-0269	555
809-555-3915	555
357-687-6708	687
NULL	NULL

Você pode aninhar INSTR no argumento *início* ou *comprimento* para pesquisar uma string específica e retornar sua posição.

A expressão a seguir avalia uma string, começando a partir do final da string. A expressão encontra o último espaço (extrema direita) na string e, em seguida, retorna todos os caracteres precedentes:

```
SUBSTR( CUST_NAME,1,INSTR( CUST_NAME,' ' ,-1,1 ) - 1 )
```

CUST_NAME	RETURN VALUE
PATRICIA JONES	PATRICIA
MARY ELLEN SHAH	MARY ELLEN

A seguinte expressão remove o caractere '#' de uma string:

```
SUBSTR( CUST_ID, 1, INSTR(CUST_ID, '#')-1 ) || SUBSTR( CUST_ID, INSTR(CUST_ID, '#')+1 )
```

Quando o argumento *length* for maior que a string, SUBSTR retornará todos os caracteres da posição inicial até o fim da string. Considere o seguinte exemplo:

```
SUBSTR('abcd', 2, 8)
```

O valor retornado é ‘bcd’. Compare esse resultado ao seguinte exemplo:

```
SUBSTR('abcd', -2, 8)
```

O valor retornado é ‘cd’.

SUM

Retorna a soma de todos os valores na porta selecionada. Como opção, você pode aplicar um filtro para limitar as linhas lidas para calcular o total. Você pode aninhar apenas uma outra função de agregação dentro de SUM, e a função aninhada deverá retornar um tipo de dados Numérico.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
valor_numérico	Requerido	Tipo de dados Numérico. Passa os valores que você deseja adicionar. Você pode inserir qualquer expressão de transformação válida. Você pode usar operadores para adicionar valores em portas diferentes.
filter_condition	Opcional	Limita as linhas na pesquisa. A condição de filtro deve ser um valor numérico ou avaliada como TRUE, FALSE ou NULL. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Valor numérico.

NULL se todos os valores passados para a função forem NULL ou se nenhuma linha for selecionada (por exemplo, a condição de filtro avaliada como FALSE ou NULL em todas as linhas).

Nota: Se o valor retornado for Decimal com precisão superior a 15, habilite a alta precisão para garantir precisão decimal de até 28 dígitos.

Nulls

Se um único valor for NULL, SUM irá ignorá-lo. No entanto, se todos os valores passados da porta forem NULL, SUM retornará NULL.

Agrupar por

SUM agrupa valores baseados em grupos por portas que você define na transformação, retornando um resultado para cada grupo.

Se não houver grupo por porta, SUM tratará todas as linhas como um grupo, retornando um valor.

Exemplo

A seguinte expressão retorna a soma de todos os valores maiores que 2000 na porta Vendas:

```
SUM( SALES, SALES > 2000 )
```

SALES
2500.0
1900.0
1200.0
NULL

```
SALES

3458.0

4519.0

RETURN VALUE: 10477.0
```

Dica

Você pode realizar operação aritmética nos valores passados para SUM antes que a função calcule o total. Por exemplo:

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

SYSTIMESTAMP

Retorna a data e hora atuais do nó que hospeda o Data Integration Service com precisão de nanossegundos. A precisão para a qual você exibe a data e a hora depende da plataforma.

O valor retornado da função varia, dependendo de como você configura o argumento:

- Quando você configura o argumento de SYSTIMESTAMP como variável, o Data Integration Service avalia a função para cada linha na transformação.
- Quando você configura o argumento de SYSTIMESTAMP como constante, o Data Integration Service avalia a função uma vez e retém o valor para cada linha na transformação.

Sintaxe

```
SYSTIMESTAMP( [format] )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
formato	Opcional	Precisão para a qual você deseja recuperar o carimbo de data/hora. É possível especificar a precisão para segundos (SS), milissegundos (MS), microssegundos (US) ou nanossegundos (NS). Coloque a string de formato entre aspas simples. A string de formato não faz distinção entre maiúsculas e minúsculas. Por exemplo, para exibir a data e a hora da precisão de milissegundos, use a seguinte sintaxe: SYSTIMESTAMP('MS'). A precisão padrão é microssegundos (US).

Valor de Retorno

Carimbo de data/hora. Retorna a data e a hora à precisão específica.

Exemplos

A sua organização possui um serviço de pedido on-line e processa dados em tempo real. Você pode usar a função SYSTIMESTAMP para gerar uma chave primária de cada transação no banco de dados de destino.

Crie uma transformação de Expressão com as seguintes portas e 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)

Na execução, o Data Integration Service gera a hora do sistema referente a cada linha com precisão de microssegundos:

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

Retorna a tangente de um valor numérico (expresso em radianos).

Sintaxe

```
TAN( numeric_value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido / Opcional	Descrição
valor_numérico	Requerido	Tipo de dados Numérico. Dados numéricos expressos em radianos (graus multiplicados por pi divididos por 180). Passa os valores numéricos para os quais você deseja calcular a tangente. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Valor duplo.

NULL se um valor transmitido para a função for NULL.

Exemplo

A seguinte expressão retorna a tangente para todos os valores na porta Graus:

```
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

Retorna a tangente hiperbólica do valor numérico passado para essa função.

Sintaxe

```
TANH( numeric_value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
valor_numérico	Requerido	Tipo de dados Numérico. Dados numéricos expressos em radianos (graus multiplicados por pi divididos por 180). Passa os valores numéricos para os quais você deseja calcular a tangente hiperbólica. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Valor duplo.

NULL se um valor transmitido para a função for NULL.

Exemplo

A seguinte expressão retorna a tangente hiperbólica para os valores na porta Ângulos:

```
TANH( ANGLES )
```

ANGLES	RETURN VALUE
1.0	0.761594155955765

ANGLES	RETURN VALUE
2.897	0.993926947790665
3.66	0.998676551914886
5.45	0.999963084213409
0	0.0
0.345	0.331933853503641
NULL	NULL

Dica

Você pode realizar operação aritmética nos valores passados para TANH antes que a função calcule a tangente hiperbólica. Por exemplo:

```
TANH( ARCS / 360 )
```

TO_BIGINT

Converte uma string ou um valor numérico em um valor bigint. A sintaxe TO_BIGINT contém um argumento opcional que você pode optar por arredondar o número para o inteiro mais próximo ou truncar a parte decimal. TO_BIGINT ignora espaços em branco à esquerda.

Sintaxe

```
TO_BIGINT( value [, flag] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
valor	Obrigatório	String ou tipo de dados numérico. Passa o valor que você deseja converter em um valor bigint. Você pode inserir qualquer expressão de transformação válida.
flag	Opcional	Especifica se trunca ou arredonda a parte decimal. O flag deve ser um inteiro ou as constantes TRUE ou FALSE. TO_BIGINT trunca a parte decimal quando o flag é TRUE ou um número diferente de 0. TO_BIGINT arredondará o valor para o inteiro mais próximo se o flag for FALSE ou 0 ou se você omitir esse argumento. O flag não é definido por padrão.

Valor de Retorno

Bigint.

NULL se o valor passado para a função for NULL.

Se o valor transmitido para a função contiver dados que não são válidos para um valor bigint, o Serviço de Integração de Dados marcará a linha como uma linha de erro ou reprovará o mapeamento.

Exemplos

As expressões a seguir usam valores da porta 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'	*Error. Integration Service skips this row.*
' 176201123435.87'	176201123435
'-7245176201123435.2'	-7245176201123435
'-7245176201123435.23'	-7245176201123435
-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'	*Error. Integration Service skips this row.*
' 176201123435.87'	176201123436
'-7245176201123435.6789'	-7245176201123436
'-7245176201123435.23'	-7245176201123435
-9223372036854775806.9	-9223372036854775807
9223372036854775806.9	9223372036854775807

TO_CHAR (Datas)

Converte datas em strings de caracteres. TO_CHAR também converte valores numéricos em strings. Você pode converter a data em qualquer formato usando as strings de formato TO_CHAR.

TO_CHAR (data [,formato]) converte um tipo de dados ou valor interno de data, Registro de data/hora, Registro de data/hora com Fuso Horário ou Registro de data/hora com Fuso Horário Local em um valor do tipo de dados de string especificado pela string de formato.

Sintaxe

```
TO_CHAR( date [,format] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
data	Obrigatório	Tipo de dados Data/Hora. Transmite os valores de data que você deseja converter em strings de caracteres. Você pode inserir qualquer expressão válida de transformação.
formato	Opcional	Insira uma string de formato TO_CHAR válida. A string de formato define o formato do valor de retorno, não o formato dos valores no argumento de data. Se você omitir a string de formato, a função retornará uma string com base no formato de data especificado na configuração do mapeamento.

Valor de Retorno

String.

NULL se um valor transmitido para a função for NULL.

Exemplos

A seguinte expressão converte as datas na porta DATE_PROMISED em texto no 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

Se você omitir o argumento *formato*, TO_CHAR retornará uma string no formato de data especificado na configuração do mapeamento, por padrão, 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'

DATE_PROMISED	RETURN VALUE
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

As seguintes expressões retornam o dia da semana para cada data em uma porta:

```
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

A seguinte expressão retorna o dia do mês para cada data em uma porta:

```
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

A seguinte expressão retorna o dia do ano para cada data em uma porta:

```
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

As seguintes expressões retornam a hora do dia para cada data em uma porta:

```
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

A seguinte expressão converte valores de dados em valores MJD expressos como strings:

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

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

A seguinte expressão converte datas em strings no formato MM/DD/AAAA:

```
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

Você também pode usar a string de formato SSSSS em uma expressão TO_CHAR. Por exemplo, a seguinte expressão converte as datas na porta SHIP_DATE em strings que representam o total de segundos desde a meia-noite:

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

SHIP_DATE	RETURN_VALUE
12/31/1999 01:02:03AM	3783
09/15/1996 03:59:59PM	86399

Em expressões TO_CHAR, a string de formato YY produz os mesmos resultados que a string de formato RR.

A seguinte expressão converte datas em strings no formato MM/DD/AAAA:

```
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

A seguinte expressão retorna a semana do mês para data em uma porta:

```
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

A seguinte expressão retorna a semana do ano para data em uma porta:

```
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

Dica

Você pode combinar TO_CHAR e TO_DATE para converter um valor numérico de um mês no valor de texto de um mês usando uma função, como:

```
TO_CHAR( TO_DATE( numeric_month, 'MM' ), 'MONTH' )
```

TO_CHAR (Números)

Converte valores numéricos em strings de texto. TO_CHAR também converte datas em strings.

TO_CHAR converte os valores duplos em cadeias de texto da seguinte maneira:

- Converte os valores duplos de até 16 dígitos em cadeias e fornece uma precisão de até 15 dígitos. Se você transferir um número com mais de 15 dígitos, TO_CHAR arredondará o número com base no décimo sexto número e retornará a representação de cadeia do número em notação científica. Por exemplo, o valor duplo 1234567890123456 é convertido no o valor de cadeia '1.23456789012346e + 015'.
- Retorna a notação decimal de números nos intervalos (-1e16,-1e-16] e [1e-16, 1e16). TO_CHAR retorna a notação científica de números fora desses intervalos. Por exemplo, o valor duplo 10842764968208837340 é convertido no valor de cadeia '1.08427649682088e + 019'.

TO_CHAR converte os valores decimais em cadeias de texto da seguinte maneira:

- No modo de alta precisão, TO_CHAR converte os valores decimais de até 38 dígitos em cadeias. Se você transferir um valor decimal com mais de 38 dígitos, TO_CHAR retornará a notação científica para números maiores de 38 dígitos.
- No modo de baixa precisão, TO_CHAR trata os valores decimais como valores duplos.

Sintaxe

```
TO_CHAR( numeric_value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Obrigatório / Opcional	Descrição
numeric_value	Obrigatório	Tipo de dados numérico. O valor numérico que você deseja converter em uma cadeia. Você pode inserir qualquer expressão válida de transformação.

Valor de Retorno

String.

NULL se um valor transmitido para a função for NULL.

Exemplo de Conversão de Duplos

A seguinte expressão converte os valores duplos na porta SALES em cadeias:

```
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

Exemplo de Conversão de Decimal

A seguinte expressão converte os valores decimais na porta SALES em cadeias no modo de alta precisão:

```
TO_CHAR( SALES )
```

SALES	RETURN VALUE
2378964536789761	'2378964536789761'
12345678901234567890123456789	'12345678901234567890123456789'
1.2345789454696493458761234567	'1.2345789454696493458761234567'
0.99999999999999999999999999	'0.99999999999999999999999999'
1234567890123456789012345679	'1234567890123456789012345679'
2345678899223345667845893456712346523	'2345678899223345667845893456712346523'
42345678901234567890123456799123456789	'4.23456789012346e+038'

(maior que 38)

TO_DATE

Converte uma string de caracteres em um tipo de dados Data/Hora. Use as strings de formato TO_DATE para especificar o formato das strings de origem.

A porta de saída deve ser Data/Hora para expressões TO_DATE.

Se você estiver convertendo anos de dois dígitos com TO_DATE, use a string de formato RR ou YY. Não use a string de formato YYYY.

Sintaxe

```
TO_DATE( string [, format] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
string	Requerido	Deve ser um tipo de dados de string. Passa os valores que você deseja converter em datas. Você pode inserir qualquer expressão de transformação válida.
formato	Opcional	Insira uma string de formato TO_DATE válida. A string de formato deve corresponder às partes do argumento *string*. Por exemplo, se você passar a string 'Mar 15 1998 12:43:10AM', será necessário usar a string de formato 'MON DD YYYY HH12:MI:SSAM'. Se você omitir a string de formato, o valor da string deverá estar no formato de data especificado na sessão.

Valor de Retorno

Data.

TO_DATE sempre retorna uma data e uma hora. Se você passar uma string que não tem um valor de hora, a data retornada incluirá a hora 00:00:00.000000000. Você pode mapear os resultados dessa função para qualquer coluna de destino que tenha um tipo de dados de data e hora. Se a precisão da coluna de destino for menor que nanossegundos, o Data Integration Service truncará o valor datetime para corresponder à precisão da coluna de destino ao gravar valores datetime no destino.

NULL se você passar um valor nulo para essa função.

Aviso: O formato da string TO_DATE deve corresponder à string de formato, incluindo quaisquer separadores de data. Se não, o Data Integration Service talvez retorne valores imprecisos ou ignore o registro.

Exemplos

A expressão a seguir retorna valores de data para as strings na porta DATE_PROMISED. TO_DATE sempre retorna uma data e hora. Se você passar uma string que não tenha valor de hora, a data retornada incluirá a hora 00:00:00.000000000. Se você executar um mapeamento no século vinte, o século será 19. Nesse exemplo, o ano atual no nó que executa o Data Integration Service é 1998. O formato datetime da coluna de destino é MON DD YY HH24:MI SS, e assim o Data Integration Service trunca o valor datetime em segundos ao gravar no 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

A expressão a seguir retorna valores de data para as strings na porta DATE_PROMISED. Se você passar uma string que não tenha valor de hora, o Data Integration Service retornará um erro. Se você executar um mapeamento no século vinte, o século será 19. O ano atual no nó que executa o Data Integration Service é 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'	*Error. Integration Service skips this row.*
NULL	NULL

A seguinte expressão converte strings na porta SHIP_DATE_MJD_STRING em valores de data:

```
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

Como a string de formato J não inclui a parte de tempo de uma data, os valores retornados têm a hora definida como 00:00:00.000000000.

A expressão a seguir converte uma string em um formato de ano de quatro dígitos. O ano atual é 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

A expressão a seguir converte uma string em um formato de ano de quatro dígitos. O ano atual é 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: Na segunda linha, RR retorna o ano 2005 e YY retorna o ano 1905.

A expressão a seguir converte uma string em um formato de ano de quatro dígitos. O ano atual é 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

A expressão a seguir converte uma string em um formato de ano de quatro dígitos. O ano atual é 1998:

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

DATE_STR	RETURN VALUE
'04/01/998'	04/01/1998 00:00:00.000000000
'08/17/995'	08/17/1995 00:00:00.000000000

A seguinte expressão converte strings que incluem os segundos desde a meia-noite em valores de data:

```
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

Se o destino aceitar formatos de data diferentes, use TO_DATE e IS_DATE com a função DECODE para testar formatos aceitáveis. Por exemplo:

```
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') )
```

Você pode combinar TO_CHAR e TO_DATE para converter um valor numérico de um mês no valor de texto de um mês usando uma função, como:

```
TO_CHAR( TO_DATE( numeric_month, 'MM' ), 'MONTH' )
```

Tópicos Relacionados:

- "Regras e diretrizes para strings de formato de data" na página 46

TO_DECIMAL

Converte uma string ou valor numérico em um valor decimal. TO_DECIMAL ignora espaços em branco à esquerda.

Sintaxe

```
TO_DECIMAL( value [, scale] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
valor	Obrigatório	Deve ser uma string ou um tipo de dados numéricos. Transmite os valores que você deseja converter em valores decimais. Você pode inserir qualquer expressão válida de transformação.
escala	Opcional	Deve ser um número literal inteiro entre 0 e 28, inclusive. Especifica o número de dígitos permitidos após uma vírgula decimal. Se você omitir esse argumento, a função retornará um valor com a mesma escala que o valor de entrada.

Valor de Retorno

Decimal da precisão e escala entre 0 e 28, inclusive.

NULL se um valor transmitido para a função for NULL.

Se o valor transmitido para a função contiver dados inválidos para um valor decimal, o Serviço de Integração de Dados marcará a linha como uma linha de erro.

Nota: Se o valor retornado for Decimal com precisão superior a 15, habilite a alta precisão para garantir precisão decimal de até 28 dígitos.

Exemplo

Esta expressão usa valores da porta IN_TAX. IN_TAX é um tipo de dados de String com precisão de 44 dígitos. RETURN VALUE é um tipo de dados Decimal com precisão de 28 e 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

IN_TAX	RETURN VALUE
'A12.3Grove'	*Error. Integration Service skips this row.*
'711A1'	*Error. Integration Service skips this row.*
'1234567890.123'	1234567890.123
'12345678901234567890123456789012345678901234567890.123'	Error. Integration Service skips this row.
'12345678901234567890123456789012345678901234567890.123	Error. Integration Service skips this row.

TO_DECIMAL38

Converte uma string ou valor numérico em um valor decimal. TO_DECIMAL38 ignora espaços em branco à esquerda.

Sintaxe

```
TO_DECIMAL38( value [, scale] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
valor	Obrigatório	Deve ser um tipo de dados de string ou numérico. Transfere os valores que você deseja converter em valores decimais. Você pode inserir qualquer expressão válida de transformação.
escala	Opcional	Deve ser um número inteiro literal entre 0 e 38, inclusive. Especifica o número de dígitos permitidos após uma vírgula decimal. Se você omitir esse argumento, a função retornará um valor com a mesma escala que o valor de entrada.

Valor de Retorno

Decimal da precisão e escala entre 0 e 38, inclusive.

NULL se um valor transmitido para a função for NULL.

Se o valor transmitido para a função contiver dados inválidos para um valor decimal, o Serviço de Integração de Dados marcará a linha como uma linha de erro. Por exemplo, se você transmitir `TO_DECIMAL38("12345678901234567890123456789012345678901234567890.12")`, o Serviço de Integração de Dados rejeitará a linha.

Nota: Se o valor de retorno for Decimal com precisão superior a 15, você poderá ativar a alta precisão para garantir precisão decimal de até 38 dígitos.

Exemplo

Esta expressão usa valores da porta IN_TAX. IN_TAX é um tipo de dados de String com precisão de 44 dígitos. RETURN VALUE é um tipo de dados Decimal com precisão de 38 e 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'	*Error. Integration Service skips this row.*
'1234567890.123'	1234567890.123
'12345678901234567890123456789.123'	12345678901234567890123456789.123
'123456789012345678901234567890123456789.123'	*Error. Integration Service skips this row.*
'711A1'	*Error. Integration Service skips this row.*

TO_FLOAT

Converte uma string ou valor numérico em um número de ponto flutuante de precisão dupla (o tipo de dados Duplo). TO_FLOAT ignora espaços em branco à esquerda.

Sintaxe

```
TO_FLOAT( value )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Obrigatório / Opcional	Descrição
valor	Obrigatório	Deve ser uma string ou tipo de dados numéricos. Passa os valores que você deseja converter em valores duplos. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Valor duplo.

NULL se um valor transmitido para esta função for NULL.

Se o valor transmitido para a função contiver dados que não são válidos para um valor flutuante, o Serviço de Integração de Dados marcará a linha como uma linha de erro ou reprovará o mapeamento.

Exemplo

Esta expressão usa valores da porta 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'	*Error. Integration Service skips this row.*

TO_INTEGER

Converte uma string ou valor numérico em um inteiro. A sintaxe TO_INTEGER contém um argumento opcional que você pode optar por arredondar o número para o inteiro mais próximo ou truncar a parte decimal. TO_INTEGER ignora espaços em branco à esquerda.

Sintaxe

```
TO_INTEGER( value [, flag] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
valor	Obrigatório	String ou tipo de dados numérico. Passa o valor que você deseja converter em um inteiro. Você pode inserir qualquer expressão de transformação válida.
flag	Opcional	Especifica se trunca ou arredonda a parte decimal. O flag deve ser um inteiro ou as constantes TRUE ou FALSE. TO_INTEGER trunca a parte decimal quando o flag é TRUE ou um número diferente de 0. TO_INTEGER arredondará o valor para o inteiro mais próximo se o flag for FALSE ou 0 ou se você omitir esse argumento.

Valor de Retorno

Número inteiro.

NULL se o valor passado para a função for NULL.

Se o valor transmitido para a função contiver dados que não são válidos para um valor de inteiro, o Serviço de Integração de Dados marcará a linha como uma linha de erro ou reprovará o mapeamento.

Exemplos

As expressões a seguir usam valores da porta IN_TAX. O Data Integration Service exibe um erro quando a conversão causa uma sobrecarga numérica:

```
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

Converte uma string em um valor de Registro de data/hora com Fuso Horário. A função retorna o tipo de dados de Registro de data/hora com Fuso Horário. Use as strings de formato TO_TIMESTAMP_TZ para especificar o formato das strings de origem.

Sintaxe

```
TO_TIMESTAMP_TZ ( String , [format] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Obrigatório / Opcional	Descrição
String	Obrigatório	Deve ser um tipo de dados de string. Transmite os valores que você deseja converter em Registro de data/hora com Fuso Horário. Você pode inserir qualquer expressão válida de transformação. A string deve ser uma string de caracteres.
formato	Opcional	Insira uma string válida de formato TO_TIMESTAMP_TZ. A string de formato deve corresponder às partes do argumento string. Por exemplo, se você transmitir a string "Mar 15 1997 12:43:10AM ASIA/CALCUTTA", deverá usar a string de formato "MON DD YYYY HH12:MI:SSAM TZR". Se você não especificar a string de formato, a função usará o formato padrão de data e hora na caixa de diálogo Executar Configurações.

Valor de Retorno

Retorna um tipo de dados de registro de data/hora com fuso horário.

NULL se a entrada for um valor nulo.

Se o valor transmitido para a função contiver dados inválidos para um valor de registro de data/hora com fuso horário, o Serviço de Integração de Dados marcará a linha como uma linha de erro e emitirá uma falha no mapeamento.

Exemplo

INPUT VALUE	RETURN VALUE
‘1947-08-05 10:45:00.221111000 AM America/Los_Angeles’, 'YYYY-MM-DD HH:MI:SS.NS AM TZR'	Retorna um tipo de dados de registro de data/hora com fuso horário com os seguintes dados: ‘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'	Retorna um tipo de dados de registro de data/hora com fuso horário, mesmo sem especificar a região de fuso horário no formato da região de fuso horário: ‘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’	Retorna um tipo de dados de registro de data/hora com fuso horário, mesmo sem especificar o formato de registro de data/hora com fuso horário. '1947-08-05 10:45:00.221111000 AM AMERICA/LOS_ANGELES' O formato de data e hora padrão na caixa de diálogo Executar Configurações é usado quando o formato não é especificado no nível da função. Formato de data e hora padrão: 'YYYY-MM-DD HH:MI:SS.NS AM TZR'
‘1947-08-05 10:45:00.221111000 AM America/Los_Angeles’, ’MM-DD-YYYY HH:MI:SS.NS AM'	Se um registro de data/hora com fuso horário não corresponder ao formato especificado, o seguinte erro aparecerá: 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 (Datas)

Trunca datas para um ano, mês, dia, hora, minuto, segundo, milissegundo ou microssegundo específico. Você também pode usar TRUNC para truncar números.

Você pode truncar as seguintes partes de data:

- **Ano.** Se você truncar a parte do ano da data, a função retornará Jan 1 do ano de entrada com a hora definida como 00:00:00.000000000. Por exemplo, a seguinte expressão retorna 1/1/1997 00:00:00.000000000:

  ```
  TRUNC(12/1/1997 3:10:15, 'YY')
  ```

- **Mês.** Se você truncar a parte do mês de uma data, a função retornará o primeiro dia do mês com a hora definida como 00:00:00.000000000. Por exemplo, a seguinte expressão retorna 01.04.97 00:00:00.000000000:

  ```
  TRUNC(4/15/1997 12:15:00, 'MM')
  ```

- **Dia.** Se você truncar a parte do dia de uma data, a função retornará a data com a hora definida como 00:00:00.000000000. Por exemplo, a seguinte expressão retorna 13.06.97 00:00:00.000000000:

  ```
  TRUNC(6/13/1997 2:30:45, 'DD')
  ```

- **Hora.** Se você truncar a parte da hora de uma data, a função retornará a data com os minutos, segundos e sub-segundos definidos como 0. Por exemplo, a seguinte expressão retorna 01.04.97 11:00:00.000000000:

  ```
  TRUNC(4/1/1997 11:29:35, 'HH')
  ```

- **Minuto.** Se você truncar a parte do minuto de uma data, a função retornará a data com os segundos e sub-segundos definidos como 0. Por exemplo, a seguinte expressão retorna 22.05.97 10:15:00.000000000:

  ```
  TRUNC(5/22/1997 10:15:29, 'MI')
  ```

- **Segundo.** Se você truncar a segunda parte de uma data, a função retornará a data com os milissegundos definidos como 0. Por exemplo, a seguinte expressão retorna 22.05.97 10:15:29.000000000:

  ```
  TRUNC(5/22/1997 10:15:29.135, 'SS')
  ```

- **Milissegundo.** Se você truncar a parte de milissegundo de uma data, a função retornará a data com os microssegundos definidos como 0. Por exemplo, a seguinte expressão retorna 22.05.97 10:15:30.135000000:

```
TRUNC(5/22/1997 10:15:30.135235, 'MS')
```

- **Microssegundo.** Se você truncar a parte de microssegundo de uma data, a função retornará a data com os nanossegundos definidos como 0. Por exemplo, a seguinte expressão retorna 22.05.97 10:15:30.135235000:

```
TRUNC(5/22/1997 10:15:29.135235478, 'US')
```

Sintaxe

```
TRUNC( date [,format] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
data	Requerido	Tipo de dados Data/Hora. Os valores de data que você deseja truncar. Você pode inserir qualquer expressão de transformação válida que avalie em uma data.
formato	Opcional	Insira uma string de formato válida. A string de formato não faz distinção entre maiúsculas e minúsculas. Se você omitir a string de formato, a função truncará a parte da hora da data, definindo-a como 00:00:00.000000000.

Valor de Retorno

Data.

NULL se um valor transmitido para a função for NULL.

Exemplos

As seguintes expressões truncam a parte do ano de dados na porta 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

As seguintes expressões truncam a parte do mês de cada data na porta 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
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

As seguintes expressões truncam a parte do dia de cada data na porta 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

As seguintes expressões truncam a parte da hora de cada data na porta 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

As seguintes expressões truncam a parte do minuto de cada data na porta 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 os números em um dígito específico. Você também pode usar TRUNC para truncar datas.

Sintaxe

```
TRUNC( numeric_value [, precision] )
```

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido/ Opcional	Descrição
valor_numérico	Requerido	Tipo de dados Numérico. Passa os valores que você deseja truncar. Você pode inserir qualquer expressão de transformação válida que avalie em um tipo de dados Numérico.
precisão	Opcional	Pode ser um inteiro positivo ou negativo. Você pode inserir qualquer expressão de transformação válida que avalie em um inteiro. O inteiro especifica o número de dígitos para truncar.

Se a *precisão* for um inteiro positivo, TRUNC retornará o *valor_numérico* com o número de casas decimais especificadas pela *precisão*. Se a *precisão* for um inteiro negativo, TRUNC alterará os dígitos especificados à esquerda do ponto decimal para zeros. Se você omitir o argumento de *precisão*, TRUNC truncará a parte decimal do *valor_numérico* e retornará um inteiro.

Se você passar um valor de *precisão* decimal, o Data Integration Service arredondará *valor_numérico* para o inteiro mais próximo antes de avaliar a expressão.

Quando você executar um mapeamento no modo de alta precisão, use a função ROUND antes do truncamento.

Por exemplo, suponha que a expressão a seguir é usada para truncar os valores na porta QTY:

```
TRUNC ( QTY / 15 )
```

Quando o valor para QTY = 15000000, a sessão retorna o valor 999999. O resultado esperado é 1000000.

No tempo de execução, o Data Integration Service avalia a parte constante da expressão e, em seguida, a parte variável.

Na expressão acima, QTY é o valor de variável e (1/15) é o valor constante.

Quando QTY = 15000000, a expressão é avaliada como segue:

```
TRUNC ( 15000000 * (1/15)
TRUNC ( 15000000 * (1/15)
= TRUNC ( 15000000 * 0.0666666666666666)
= TRUNC ( 15000000 * 0.0666666666666666)
= TRUNC ( 999999.99999999)
= 999999
```

Se você usar a função ROUND antes do truncamento, a expressão é avaliada como segue:

```
TRUNC (ROUND (QTY/15, .99999999999999999999999999)).
```

Valor de Retorno

Valor numérico.

NULL se um dos argumentos for NULL.

Nota: Se o valor retornado for Decimal com precisão superior a 15, habilite a alta precisão para garantir precisão decimal de até 28 dígitos.

Exemplos

As seguintes expressões truncam os valores no porta Preço:

```
TRUNC( PRICE, 3 )
```

PRICE	RETURN VALUE
12.9995	12.999
-18.8652	-18.865
56.9563	56.956
15.9928	15.992
NULL	NULL

```
TRUNC( PRICE, -1 )
```

PRICE	RETURN VALUE
12.99	10.0
-187.86	-180.0
56.95	50.0

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

Converte os caracteres minúsculos de uma string em maiúsculos.

Sintaxe

```
UPPER( string )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
string	Requerido	Tipo de dados strings. Passa os valores que você deseja alterar para texto em letras maiúsculas. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

String de letra maiúscula. Se os dados contiverem caracteres multibyte, o valor retornado dependerá da página de código e do modo de movimento de dados do Data Integration Service.

NULL se um valor transmitido para a função for NULL.

Exemplo

A seguinte expressão altera todos os nomes na porta FIRST_NAME para letra maiúscula:

```
UPPER( FIRST_NAME )
```

FIRST_NAME	RETURN VALUE
Ramona	RAMONA

FIRST_NAME	RETURN VALUE
NULL	NULL
THOMAS	THOMAS
PierRe	PIERRE
Bernice	BERNICE

UUID4

Retorna um valor binário de 16 bytes gerado aleatoriamente, que está de acordo com a variante 4 da especificação de UUID descrita na RFC 4122. UUID4 não tem um argumento.

Sintaxe

```
UUID4()
```

Valor de Retorno

Binário.

UUID4 nunca retorna um valor nulo ou um erro.

UUID_UNPARSE

Converte um valor binário de 16 bytes em uma representação de string de 36 caracteres, conforme especificado na RFC 4122.

Sintaxe

```
UUID_UNPARSE( binary )
```

A tabela a seguir descreve o argumento para este comando:

Argumento	Requerido/ Opcional	Descrição
binário	Obrigatório	Tipo de dados Binário. Qualquer valor binário de 16 bytes que você deseja converter em uma string de 36 caracteres.

Valor de Retorno

String de 36 caracteres.

Retorna nulo quando o argumento é nulo, e um erro quando o argumento não é um valor binário de 16 bytes.

Exemplo

A seguinte expressão pode retornar um valor de 6948DF80-14BD-4E04-8842-7668D9C001F5:

```
UUID_UNPARSE(UUID4())
```

VARIANCE

Retorna a variação de um valor que você passa para ela. VARIANCE é usado para analisar dados estatísticos. Você pode aninhar apenas uma outra função de agregação dentro de VARIANCE, e a função aninhada deverá retornar um tipo de dados Numérico.

Sintaxe

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

A tabela a seguir descreve os argumentos para este comando:

Argumento	Requerido / Opcional	Descrição
valor_numérico	Obrigatório	Tipo de dados Numérico. Passa os valores para os quais você deseja calcular uma variação. Você pode inserir qualquer expressão de transformação válida.
filter_condition	Opcional	Limita as linhas na pesquisa. A condição de filtro deve ser um valor numérico ou avaliada como TRUE, FALSE ou NULL. Você pode inserir qualquer expressão de transformação válida.

Valor de Retorno

Valor duplo.

NULL se todos os valores passados para a função forem NULL ou se nenhuma linha for selecionada (por exemplo, a *filter_condition* avaliada como FALSE ou NULL em todas as linhas).

Nulls

Se um único valor for NULL, VARIANCE irá ignorá-lo. No entanto, se todos os valores passados para a função forem NULL ou se nenhuma linha for selecionada, VARIANCE retornará NULL.

Agrupar por

VARIANCE agrupa valores baseados em grupos por portas que você define na transformação, retornando um resultado para cada grupo.

Se não houver grupo por porta, VARIANCE tratará todas as linhas como um grupo, retornando um valor.

Exemplo

A seguinte expressão calcula a variação de todas as linhas na porta TOTAL_SALES:

```
VARIANCE( TOTAL_SALES )
```

TOTAL_SALES
2198.0
2256.0
3001.0
NULL

```
TOTAL_SALES
```

```
8953.0
```

RETURN VALUE: 10592444.6666667

ÍNDICE

A

algoritmo Padrão Avançado de Criptografia
  descrição 61, 62
alta precisão
  ABS 58
  AVG 66
  CEIL 69
  CUME 82
  EXP 93
  função ABS 58
  função AVG 66
  Função CREATE_TIMESTAMP_TZ 82
  função CUME 82
  Função GET_TIMESTAMP 100
  Função GET_TIMEZONE 99
  função MAX 133
  função MEDIAN 137
  função MIN 143
  função MOVINGAVG 147
  função MOVINGSUM 148
  função PERCENTILE 150
  função ROUND 173
  função STDDEV 188
  função SUM 195
  função TO_DECIMAL 210
  Função TO_DECIMAL38 211
  Função TO_TIMESTAMP_TZ 215
  função TRUNC 219
  LOG 126
  MAX (números) 133
  MEDIAN 137
  MIN (números) 143
  MOD 145
  MOVINGAVG 147
  MOVINGSUM 148
  operadores aritméticos 31
  PERCENTILE 150
  POWER 153
  ROUND (números) 173
  SIGN 181
  SIN 182
  SUM 195
AND
  palavra reservada 16
ano 2000
  datas 37
aritméticos
  valores de data e hora 48
arquivos simples
  datas 39
array
  gerando 65, 72
arredondando
  datas 169
  números 173
ASCII
  convertendo caracteres em valores ASCII 65
  convertendo em valores Unicode 71
  convertendo valores ASCII 70
  função CHR 70
aspas
  inserindo simples usando função CHR 14
aspas simples em literais de string
  função CHR 70
  usando funções CHR e CONCAT 74
atualizações de linguagem de transformação
  expressões booleanas 20
  expressões de comparação 20

B

bancos de dados relacionais
  datas 39
bigint
  convertendo valores em 199

C

cálculo de divisão
  retornando restante 145
Calendário gregoriano
  em funções de data 37
calendários
  tipos de dados com suporte 37
capitalização
  strings 107, 127, 221
caracteres
  adicionando a strings 128, 176
  capitalização 107, 127, 221
  caracteres ASCII 65, 70
  caracteres Unicode 65, 70, 71
  codificando 138, 185
  contando 192
  removendo de strings 130, 177
  retornando número 124
  substituindo um 161
  substituindo vários 164
codificando
  caracteres 138, 185
  função ENC_BASE64 91
comentários
  adicionando a expressões 15
compactação
  compactação de dados 73
  descompactando dados 91
componentes da linguagem de transformação
  visão geral 12
concatenando
  strings 32, 74

condições de filtro
  funções de agregação 51
  valores nulos 20
constante DD_DELETE
  atualizar exemplo de estratégia 17
  descrição 17
  palavra reservada 16
constante DD_INSERT
  atualizar exemplo de estratégia 17
  descrição 17
  palavra reservada 16
constante DD_REJECT
  atualizar exemplo de estratégia 18
  descrição 18
  palavra reservada 16
constante DD_UPDATE
  atualizar exemplo de estratégia 19
  descrição 19
  palavra reservada 16
constante FALSE
  descrição 19
  palavra reservada 16
constante NULL
  descrição 20
  palavra reservada 16
constante TRUE
  descrição 21
  palavra reservada 16
constantes
  DD_INSERT 17
  DD_REJECT 18
  DD_UPDATE 19
  descrição 12
  FALSE 19
  NULL 20
  TRUE 21
conversão de string
  datas 37
convertendo
  strings de data 37
cosseno
  calculando 77
  calculando cosseno hiperbólico 78
criptografia
  função AES_ENCRYPT 62
  usando o algoritmo Padrão Avançado de Criptografia 62

D

dados hierárquicos
  acessando elementos 23
  gerando 65, 72, 190, 191
datas
  ano 2000 37
  arquivos simples 39
  arredondando 169
  bancos de dados relacionais 39
  convertendo em strings de caracteres 201
  formato de data e hora padrão 39
  funções 54
  Juliano 37
  Juliano modificado 37
  realizando operação aritmética 48
  strings de formato 40
  truncando 216
  visão geral 36
Datas julianas
  em funções de data 37
decodificação
  função AES_DECRYPT 61
  função DEC_BASE64 88
desvio padrão
  retornando 188
dia Juliano
  string de formato 41, 44
Dia Juliano Modificado
  string de formato 41, 44

E

espaços
  evitando em linhas 117
  removendo com DD_REJECT 18
estratégia de atualização
  exemplo de DD_DELETE 17
  exemplo de DD_INSERT 17
  exemplo de DD_REJECT 18
  exemplo de DD_UPDATE 19
expressões
  adicionando comentários 15
  condicional 19
  sintaxe 13
  usando operadores 22
  visão geral 12
expressões aninhadas
  operadores 22
expressões de transformação
  restrições nulas 20
  visão geral 12

F

formato
  da string de caracteres para data 206
  de data em string de caracteres 201
formato de data e hora padrão
  definindo 39
função ABORT
  descrição 57
função ABS
  descrição 58
função ADD_TO_DATE
  descrição 59
função AES_DECRYPT
  descrição 61
função AES_ENCRYPT
  descrição 62
Função ANY
  descrição 63
Função ARRAY
  descrição 65
função ASCII
  descrição 65
função AVG
  descrição 66
função CAST
  descrição 68
função CEIL
  descrição 69
função CHOOSE
  descrição 69

função CHR
descrição 70
inserindo aspas simples 14, 70
função CHRCODE
descrição 71
função COLLECT_LIST
descrição 72
função COMPRESS
descrição 73
função CONCAT
descrição 74
inserindo aspas simples usando 74
Função CONCAT_ARRAY
descrição 75
função CONVERT_BASE
descrição 76
função COS
descrição 77
função COSH
descrição 78
função COUNT
descrição 78
função CRC32
descrição 81
Função CREATE_TIMESTAMP_TZ
descrição 82
função CUME
descrição 82
função DATE_COMPARE
descrição 84
função DATE_DIFF
descrição 85
função DEC_BASE64
descrição 88
função DECODE
descrição 88
internacionalização 13
função DECOMPRESS
descrição 91
função ENC_BASE64
descrição 91
função ERROR
descrição 92
valor padrão 92
função EXP
descrição 93
função FIRST
descrição 94
função FLOOR
descrição 95
função FLOOR (expressões)
descrição 95
função FV
descrição 96
função GET_DATE_PART
descrição 97
Função GET_TIMESTAMP
descrição 100
Função GET_TIMEZONE
descrição 99
função GREATEST
descrição 101
função IIF
descrição 102
internacionalização 13
função IN
descrição 105
função INDEXOF
descrição 106
função INITCAP
descrição 107
internacionalização 13
função INSTR
descrição 108
função IS_DATE
descrição 113
strings de formato 44
função IS_NUMBER
descrição 115
função IS_SPACES
descrição 117
função ISNULL
descrição 111
função LAG
descrição 118
função LAST
descrição 119
função LAST_DAY
descrição 120
função LEAD
descrição 122
função LEAST
descrição 123
função LENGTH
descrição 124
teste de string vazia 124
função LN
descrição 125
função LOG
descrição 126
função LOWER
descrição 127
internacionalização 13
função LPAD
descrição 128
função LTRIM
descrição 130
função MAKE_DATE_TIME
descrição 131
função MAX (datas)
descrição 132
internacionalização 13
função MAX (números)
descrição 133
internacionalização 13
Função MAX (string)
descrição 135
função MD5
descrição 136
função MEDIAN
descrição 137
função MIN (datas)
descrição 142
internacionalização 13
função MIN (números)
descrição 143, 144
internacionalização 13
função MOD
descrição 145
função MOVINGAVG
descrição 147
função MOVINGSUM
descrição 148
função NPER
descrição 149

função PERCENTILE
descrição 150
função PMT
descrição 152
função POWER
descrição 153
função PV
descrição 154
função RAND
descrição 155
função RATE
descrição 156
função REG_EXTRACT
descrição 156
usando sintaxe perl 156
função REG_MATCH
descrição 159
usando sintaxe perl 156
função REG_REPLACE
descrição 160
função REPLACECHR
descrição 161
função REPLACESTR
descrição 164
função RESPEC
descrição 167
função REVERSE
descrição 168
função ROUND (datas)
descrição 169
processando subssegundos 169
Função ROUND (números)
descrição 173
função RPAD
descrição 176
função RTRIM
descrição 177
função SET_DATE_PART
descrição 178
função SIGN
descrição 181
função SIN
descrição 182
função SINH
descrição 183
função SIZE
descrição 184
função SOUNDEX
descrição 185
função SQL IS_CHAR
usando REG_MATCH 159
função SQL LIKE
usando REG_MATCH 159
Função SQL_LIKE
descrição 186
função SQRT
descrição 187
função STDDEV
descrição 188
função STRUCT
descrição 190
função STRUCT_AS
descrição 191
função SUBSTR
descrição 192
função SUM
descrição 195
função SYSTIMESTAMP
descrição 196
função TAN
descrição 197
função TANH
descrição 198
Função TO__TIMESTAMP_TZ
descrição 215
função TO_CHAR (datas)
descrição 201
exemplos 43
strings de formato 41
Função TO_CHAR (números)
descrição 205
função TO_DATE
descrição 206
exemplos 46
strings de formato 44
função TO_DECIMAL
descrição 210
Função TO_DECIMAL38
descrição 211
função TO_FLOAT
descrição 212
função TO_INTEGER
descrição 213
função TRUNC (datas)
descrição 216
processando subssegundos 216
função TRUNC (números)
descrição 219
função UPPER
descrição 221
internacionalização 13
Função UUID_UNPARSE
descrição 222
Função UUID4
descrição 222
função VARIANCE
descrição 223
funções
agregação 49
caractere 52
categorias 49
científico 56
codificando 54
complexas 52
conversão 53
data 54
descrição 12
especial 56
financeiras 55
internacionalização 13
janela 57
limpeza de dados 53
numérico 55
string 56
teste 56
funções científicas
COS 77
COSH 78
descrição 56
SIN 182
SINH 183
TAN 197
TANH 198
funções complexas
ARRAY 65

funções complexas *()*
CAST 68
COLLECT_LIST 72
CONCAT_ARRAY 75
descrição 52
RESPEC 167
SIZE 184
STRUCT 190
STRUCT_AS 191
funções de agregação
ANY 63
AVG 66
COUNT 78
descrição 49
FIRST 94
LAST 119
MAX (datas) 132
MAX (números) 133
MAX (string) 135
MEDIAN 137
MIN (datas) 142
MIN (números) 143, 144
PERCENTILE 150
STDDEV 188
SUM 195
valores nulos 20, 51
VARIANCE 223
funções de caractere
ASCII 65
CHR 70
CHRCODE 71
função CONCAT 74
INITCAP 107
INSTR 108
LENGTH 124
lista de 52
LOWER 127
LPAD 128
LTRIM 130
METAPHONE 138
REG_EXTRACT 156
REG_MATCH 159
REG_REPLACE 160
REPLACECHR 161
REPLACESTR 164
RPAD 176
RTRIM 177
SOUNDEX 185
SUBSTR 192
UPPER 221
funções de codificação
AES_DECRYPT 61
AES_ENCRYPT 62
COMPRESS 73
CRC32 81
DEC_BASE64 88
DECOMPRESS 91
descrição 54
ENC_BASE64 91
MD5 136
funções de conversão
CREATE_TIMESTAMP_TZ 82
descrição 53
GET_TIMESTAMP 100
GET_TIMEZONE 99
TO_CHAR (datas) 201
TO_CHAR (números) 205
TO_DATE 206
funções de conversão *()*
TO_DECIMAL 210
TO_DECIMAL38 211
TO_FLOAT 212
TO_INTEGER 213
TO_TIMESTAMP_TZ 215
funções de data
ADD_TO_DATE 59
DATE_COMPARE 84
DATE_DIFF 85
GET_DATE_PART 97
LAST_DAY 120
MAKE_DATE_TIME 131
MAX (datas) 132
MIN (datas) 142
ROUND 169
SET_DATE_PART 178
SYSTIMESTAMP 196
TRUNC (Datas) 216
funções de janela
descrição 57
LAG 118
LEAD 122
funções de limpeza de dados
descrição 53
GREATEST 101
IN 105
LEAST 123
funções de string
CHOOSE 69
descrição 56
INDEXOF 106
REVERSE 168
funções de teste
descrição 56
IS_DATE 113
IS_NUMBER 115
IS_SPACES 117
ISNULL 111
funções especiais
ABORT 57
DECODE 88
descrição 56
ERROR 92
IIF 102
funções financeiras
descrição 55
função FV 96
função NPER 149
função PMT 152
função PV 154
função RATE 156
funções numéricas
ABS 58
CEIL 69
CONVERT_BASE 76
CUME 82
descrição 55
EXP 93
FLOOR 95
LN 125
LOG 126
MOD 145
MOVINGAVG 147
MOVINGSUM 148
POWER 153
RAND 155
ROUND (números) 173

funções numéricas *()*
SIGN 181
SQRT 187
TRUNC (números) 219

H

hiperbólica
função de cosseno 78
função de seno 183
função de tangente 198

I

ignorando
linhas 92
inteiros
convertendo valores em 213
internacionalização
expressão inválida 13
funções afetadas 13
ordem de classificação 13

L

:qualificador de referência LKP
descrição 14
palavra reservada 16
letras maiúsculas e minúsculas
convertendo em letras maiúsculas 221
linguagem de transformação
comparada ao SQL 13
operadores 22
palavras reservadas 16
linhas
contando 78
evitando espaços 117
ignorando 92
retornando a linha any 63
retornando a soma 148
retornando média 147
retornando primeira linha 94
retornando última linha 119
total de execução 82
literais
aspas simples em 70, 74
requerimento de aspas simples 14
literais de string
aspas simples em 70, 74
requerimento de aspas simples 14
logaritmo
retornando 125, 126

M

médias
funções de agregação para determinar 66
retornando 147
mês
retornando último dia 120
METAPHONE
descrição 138
mínimo
valor, retornando 142

N

NOT
palavra reservada 16
números
arredondando 173
truncando 219

O

operador de ponto
descrição 25
para tipos de dados complexos 23
usando para acessar dados 25
operador de subscrito
descrição 24
para tipos de dados complexos 23
usando para acessar dados 24
operadores
aritméticos 31
complexas 23
descrição 12
operadores de comparação 33
operadores de string 32
operadores lógicos 34
usando strings em aritmética 31
usando strings em comparação 33
valores nulos 21
operadores aritméticos
descrição 31
usando para converter dados 31
usando strings em expressões 31
operadores complexos
acessando tipos de dados aninhados 27
descrição 23
para array com elementos de struct 28
para arrays multidimensionais 27
para struct com elementos de array 29
para struct com elementos de struct 30
para tipos de dados aninhados 27
usando para acessar dados 23
operadores de comparação
descrição 33
usando strings em expressões 33
operadores de ponto
para struct com elementos de struct 30
para tipo de dados aninhado 27
operadores de string
descrição 32
operadores de subscrito
para arrays multidimensionais 27
para tipo de dados aninhado 27
operadores lógicos
descrição 34
OR
palavra reservada 16
ordem de classificação
internacionalização 13

P

palavras reservadas
lista 16
parâmetros de mapeamento
definição 12

portas
sintaxe 14
precedência de operador
expressões 22

Q

:qualificador de referência INFA
palavra reservada 16
:qualificador de referência MCR
palavra reservada 16
qualificador de referência:TYPE
palavra reservada 16
qualificadores de referência
descrição 14

R

raiz quadrada
retornando 187
restrição de chave primária
valores nulos 20

S

seno
retornando 182, 183
Serviço de Integração de Dados
tratando nulos em expressões de comparação 20
sintaxe
expressão 13
portas 14
regras gerais 14
valores retornados 14
sintaxe COBOL
convertendo para uma sintaxe perl 156
sintaxe de expressão regular compatível com perl
usando em uma função REG_EXTRACT 156
usando em uma função REG_MATCH 156
Sintaxe SQL
convertendo para uma sintaxe perl 156
soma
retornando 148, 195
SPOUTPUT
palavra reservada 16
string de formato RR
descrição 37
diferença entre YY e RR 38
usando com IS_DATE 47
usando com TO_CHAR 44
usando com TO_DATE 47
string de formato SSSSS
usando com IS_DATE 47
usando com TO_CHAR 43
usando com TO_DATE 47
string de formato YY
diferença entre RR e YY 38
usando com IS_DATE 47
usando com TO_CHAR 44
usando com TO_DATE 47
String no formato J
usando com IS_DATE 47
usando com TO_CHAR 43
usando com TO_DATE 47
strings
adicionando caracteres 128
adicionando espaços em branco 128
capitalização 107, 127, 221
concatenando 32, 74
conjunto de caracteres 108
convertendo datas em caracteres 201
convertendo strings de caracteres em datas 206
convertendo tamanho 176
convertendo valores numéricos em strings de texto 205
número de caracteres 124
removendo caracteres 130
removendo espaços em branco 130
removendo espaços em branco e caracteres 177
retornando a parte 192
substituindo um caractere 161
substituindo vários caracteres 164
strings de caracteres
convertendo de datas 201
convertendo em datas 206
strings de formato
correspondência 46
datas 40
definição 36
dia Juliano 41, 44
Dia Juliano Modificado 41, 44
função IS_DATE 44
função TO_CHAR 41
função TO_DATE 44
strings de texto
convertendo valores numéricos 205
strings vazias
testando 124
struct
gerando 190, 191
subssegundos
processando na função ROUND (datas) 169
processando na função TRUNC (datas) 216

T

tamanho
array 184
tangente
retornando 197, 198
tipos de dados
Data/Hora 36
total de execução
retornando 82
Transformação de filtro
usando a função ISNULL 111
truncando
datas 216
números 219

U

Unicode
convertendo caracteres em valores Unicode 65
convertendo em valores ASCII 71
convertendo valores Unicode 70

V

valores absolutos
  obtendo 58
valores de data e hora
  adicionando 59
valores de expoente
  calculando 93
  retornando 153
valores de precisão duplos
  números de ponto flutuante 212
valores de string
  retornando máximo 135
  retornando mínimo 144
valores decimais
  convertendo 82, 99, 100, 210, 211, 215
valores negativos
  SIGN 181
valores nulos
  condições de filtro 20
  em expressões de comparação 20
  funções de agregação 20, 51
  ISNULL 111
  operador de string 32
  operadores 21
  operadores lógicos 34
  verificando 111
valores numéricos
  convertendo em strings de texto 205
  retornando co-seno 77
  retornando cosseno hiperbólico de 78
  retornando desvio padrão 188
  retornando logaritmos 125, 126
  retornando mínimo 143
  retornando raiz quadrada 187
valores numéricos *()*
  retornando seno 182
  retornando seno hiperbólico 183
  retornando tangente 197
  retornando tangente hiperbólica 198
  retornando valor absoluto 58
  SIGN 181
valores padrão
  função ERROR 92
valores positivos
  SIGN 181
valores retornados
  descrição 12
  sintaxe 14
várias pesquisas
  exemplo de constante TRUE 21
variáveis
  SYSDATE 35
  variáveis internas 35
variáveis de mapeamento
  variáveis internas 35
variáveis de sistema 35
variáveis internas
  descrição 35
variáveis locais
  descrição 12
variável PROC_RESULT
  palavra reservada 16
variável SESSSTARTTIME
  usando em funções de data 48
variável SYSDATE
  descrição 35
  palavra reservada 16
  usando em expressões 35