

Informatica®

10.1.1

Informatica® PowerCenter

10.1.1

Referencia del lenguaje de transformación

© Copyright Informatica LLC 2009, 2018

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

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

Hay fragmentos de este software y/o documentación que están sujetos a copyright perteneciente a terceros, incluido, entre otros: Copyright DataDirect Technologies. Todos los derechos reservados. Copyright © Sun Microsystems. Todos los derechos reservados. Copyright © RSA Security Inc. Todos los derechos reservados. Copyright © Ordinal Technology Corp. Todos los derechos reservados. Copyright © Aandacht c.v. Todos los derechos reservados. Copyright Genivia, Inc. Todos los derechos reservados. Copyright Isomorphic Software. Todos los derechos reservados. Copyright © Meta Integration Technology, Inc. Todos los derechos reservados. Copyright © Intalio. Todos los derechos reservados. Copyright © Oracle. Todos los derechos reservados. Copyright © Adobe Systems Incorporated. Todos los derechos reservados. Copyright © DataArt, Inc. Todos los derechos reservados. Copyright © ComponentSource. Todos los derechos reservados. Copyright © Microsoft Corporation. Todos los derechos reservados. Copyright © Rogue Wave Software, Inc. Todos los derechos reservados. Copyright © Teradata Corporation. Todos los derechos reservados. Copyright © Yahoo! Inc. Todos los derechos reservados. Copyright © Glyph & Cog, LLC. Todos los derechos reservados. Copyright © Thinkmap, Inc. Todos los derechos reservados. Copyright © Clearpace Software Limited. Todos los derechos reservados. Copyright © Information Builders, Inc. Todos los derechos reservados. Copyright © OSS Nokalva, Inc. Todos los derechos reservados. Copyright Edifecs, Inc. Todos los derechos reservados. Copyright Cleo Communications, Inc. Todos los derechos reservados. Copyright © International Organization for Standardization 1986. Todos los derechos reservados. Copyright © ej-technologies GmbH. Todos los derechos reservados. Copyright © Jaspersoft Corporation. Todos los derechos reservados. Copyright © International Business Machines Corporation. Todos los derechos reservados. Copyright © yWorks GmbH. Todos los derechos reservados. Copyright © Lucent Technologies. Todos los derechos reservados. Copyright © University of Toronto. Todos los derechos reservados. Copyright © Daniel Veillard. Todos los derechos reservados. Copyright © Unicode, Inc. Copyright IBM Corp. Todos los derechos reservados. Copyright © MicroQuill Software Publishing, Inc. Todos los derechos reservados. Copyright © PassMark Software Pty Ltd. Todos los derechos reservados. Copyright © LogiXML, Inc. Todos los derechos reservados. Copyright © 2003-2010 Lorenzi Davide. Todos los derechos reservados. Copyright © Red Hat, Inc. Todos los derechos reservados. Copyright © The Board of Trustees of the Leland Stanford Junior University. Todos los derechos reservados. Copyright © EMC Corporation. Todos los derechos reservados. Copyright © Flexera Software. Todos los derechos reservados. Copyright © Jinfonet Software. Todos los derechos reservados. Copyright © Apple Inc. Todos los derechos reservados. Copyright © Telerik Inc. Todos los derechos reservados. Copyright © BEA Systems. Todos los derechos reservados. Copyright © PDFlib GmbH. Todos los derechos reservados. Copyright © Orientation in Objects GmbH. Todos los derechos reservados. Copyright © Tanuki Software, Ltd. Todos los derechos reservados. Copyright © Ricebridge. Todos los derechos reservados. Copyright © Sencha, Inc. Todos los derechos reservados. Copyright © Scalable Systems, Inc. Todos los derechos reservados. Copyright © jQWidgets. Todos los derechos reservados. Copyright © Tableau Software, Inc. Todos los derechos reservados. Copyright © MaxMind, Inc. Todos los derechos reservados. Copyright © TMate Software s.r.o. Todos los derechos reservados. Copyright © MapR Technologies Inc. Todos los derechos reservados. Copyright © Amazon Corporate LLC. Todos los derechos reservados. Copyright © Highsoft. Todos los derechos reservados. Copyright © Python Software Foundation. Todos los derechos reservados. Copyright © BeOpen.com. Todos los derechos reservados. Copyright © CNRI. Todos los derechos reservados.

Este producto incluye software desarrollado por la Apache Software Foundation (<http://www.apache.org/>) y/u otro software protegido por varias versiones de la licencia Apache License ("Licencia"). Puede obtener una copia de estas licencias en <http://www.apache.org/licenses/>. A menos que las leyes aplicables lo requieran o se haya acordado por escrito, el software distribuido bajo estas licencias se distribuye "TAL CUAL", SIN GARANTÍAS NI CONDICIONES DE NINGÚN TIPO, ya sea expresas o implícitas. Consulte las licencias del idioma específico para conocer los permisos y las limitaciones que rigen según las licencias.

Este producto incluye software desarrollado por Mozilla (<http://www.mozilla.org/>), copyright del software de The JBoss Group, LLC, todos los derechos reservados; copyright del software © 1999-2006 de Bruno Lowagie y Paulo Soares y otro software protegido con licencia por el acuerdo GNU Lesser General Public License Agreement, que se puede encontrar en la dirección <http://www.gnu.org/licenses/lgpl.html>. Los materiales se facilitan gratuitamente por parte de © Informatica, "tal cual", sin garantía de ningún tipo, ya sea expresa o implícita, incluidas, entre otras, las garantías implícitas de adecuación para un propósito determinado y de validez para el comercio.

El producto incluye software ACE(TM) y TAO(TM) con copyright de Douglas C. Schmidt y su grupo de investigación de la Washington University, University of California, Irvine y Vanderbilt University, Copyright (©) 1993-2006, todos los derechos reservados.

Este producto incluye software desarrollado por el OpenSSL Project para uso en el OpenSSL Toolkit (copyright The OpenSSL Project. Todos los derechos reservados) y la redistribución de este software está sujeta a los términos especificados en <http://www.openssl.org> y <http://www.openssl.org/source/license.html>.

Este producto incluye software Curl con Copyright 1996-2013, Daniel Stenberg, <daniel@haxx.se>. Todos los derechos reservados. Los permisos y las limitaciones relativos a este software están sujetos a los términos disponibles en la dirección <http://curl.haxx.se/docs/copyright.html>. La autorización para utilizar, copiar, modificar y distribuir este software para cualquier propósito con o sin tasas se concede por el presente, siempre que el aviso de copyright anterior y este aviso de permiso aparezcan en todas las copias.

El producto incluye copyright de software 2001-2005 (©) MetaStuff, Ltd. Todos los derechos reservados. Los permisos y las limitaciones relativos a este software están sujetos a los términos disponibles en la dirección <http://www.dom4j.org/license.html>.

El producto incluye copyright de software © 2004-2007, The Dojo Foundation. Todos los derechos reservados. Los permisos y las limitaciones relativos a este software están sujetos a los términos disponibles en la dirección <http://dojotoolkit.org/license>.

Este producto incluye software ICU con copyright de International Business Machines Corporation y otros. Todos los derechos reservados. Los permisos y las limitaciones relativos a este software están sujetos a los términos disponibles en la dirección <http://source.icu-project.org/repos/icu/icu/trunk/license.html>.

Este producto incluye copyright de software © 1996-2006 Per Bothner. Todos los derechos reservados. Su derecho a utilizar estos materiales está establecido en la licencia que puede encontrarse en la dirección <http://www.gnu.org/software/kawa/Software-License.html>.

Este producto incluye software OSSP UUID con Copyright © 2002 Ralf S. Engelschall, Copyright © 2002 The OSSP Project Copyright © 2002 Cable & Wireless Deutschland. Los permisos y las limitaciones relativas a este software están sujetos a los términos disponibles en la dirección <http://www.opensource.org/licenses/mit-license.php>.

Este producto incluye software desarrollado por Boost (<http://www.boost.org/>) o protegido por la licencia de software de Boost. Los permisos y las limitaciones relativos a este software están sujetos a los términos disponibles en la dirección http://www.boost.org/LICENSE_1_0.txt.

Este producto incluye copyright de software © 1997-2007 University of Cambridge. Los permisos y las limitaciones relativos a este software están sujetos a los términos disponibles en la dirección <http://www.pcre.org/license.txt>.

Este producto incluye copyright de software © 2007 The Eclipse Foundation. Todos los derechos reservados. Los permisos y las limitaciones relativos a este software están sujetos a los términos especificados en <http://www.eclipse.org/org/documents/epl-v10.php> y <http://www.eclipse.org/org/documents/edl-v10.php>.

Este producto incluye software protegido por licencia según los términos que aparecen en <http://www.tcl.tk/software/tcltk/license.html>, <http://www.bosrup.com/web/overlib/?License>, <http://www.stlport.org/doc/license.html>, <http://asm.ow2.org/license.html>, <http://www.cryptix.org/LICENSE.TXT>, <http://hsqldb.org/web/>

hsqllicense.html, <http://httpunit.sourceforge.net/doc/license.html>, <http://jung.sourceforge.net/license.txt>, http://www.gzip.org/zlib/zlib_license.html, <http://www.openldap.org/software/release/license.html>, <http://www.libssh2.org>, <http://slf4j.org/license.html>, <http://www.sente.ch/software/OpenSourceLicense.html>, <http://fusesource.com/downloads/license-agreements/fuse-message-broker-v-5-3-license-agreement>, <http://antlr.org/license.html>, <http://aopalliance.sourceforge.net/>, <http://www.bouncycastle.org/licence.html>, <http://www.jgraph.com/jgraphdownload.html>, <http://www.jcraft.com/jsch/LICENSE.txt>, http://jotm.objectweb.org/bsd_license.html, <http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231>, <http://www.slf4j.org/license.html>, <http://nanoxml.sourceforge.net/orig/copyright.html>, <http://www.json.org/license.html>, <http://forge.ow2.org/projects/javaservice/>, <http://www.postgresql.org/about/licence.html>, <http://www.sqlite.org/copyright.html>, <http://www.tcl.tk/software/tcltk/license.html>, <http://www.jaxen.org/faq.html>, <http://www.jdom.org/docs/faq.html>, <http://www.slf4j.org/license.html>, <http://www.iodbc.org/dataspace/iodbc/wiki/iodbc/License>, <http://www.keplerproject.org/md5/license.html>, <http://www.toedter.com/en/jcalendar/license.html>, <http://www.edankert.com/bounce/index.html>, <http://www.net-snmp.org/about/license.html>, <http://www.openmdx.org/#FAQ>, http://www.php.net/license/3_01.txt, <http://srp.stanford.edu/license.txt>, <http://www.schneier.com/blowfish.html>, <http://www.jmock.org/license.html>, <http://xsom.java.net>, <http://benalman.com/about/license/>, <https://github.com/CreateJS/EaselJS/blob/master/src/easeljs/display/Bitmap.js>, <http://www.h2database.com/html/license.html#summary>, <http://jsoncpp.sourceforge.net/LICENSE>, <http://jdbc.postgresql.org/license.html>, <http://protobuf.googlecode.com/svn/trunk/src/google/protobuf/descriptor.proto>, <https://github.com/rantav/hector/blob/master/LICENSE>, <http://web.mit.edu/Kerberos/krb5-current/doc/mitK5license.html>, <http://jibx.sourceforge.net/jibx-license.html>, <https://github.com/lyokato/libgeohash/blob/master/LICENSE>, <https://github.com/hjiang/jsonxx/blob/master/LICENSE>, <https://code.google.com/p/lz4/>, <https://github.com/jedisct1/libsodium/blob/master/LICENSE>, <http://one-jar.sourceforge.net/index.php?page=documents&file=license>, <https://github.com/EsotericSoftware/kryo/blob/master/license.txt>, <http://www.scala-lang.org/license.html>, <https://github.com/tinkerpop/blueprints/blob/master/LICENSE.txt>, <http://gee.cs.oswego.edu/dl/classes/EDU/oswego/cs/dl/util/concurrent/intro.html>, <https://aws.amazon.com/asl/>, <https://github.com/twbs/bootstrap/blob/master/LICENSE>, <https://sourceforge.net/p/xmlunit/code/HEAD/tree/trunk/LICENSE.txt>, <https://github.com/documentcloud/underscore-contrib/blob/master/LICENSE> y <https://github.com/apache/hbase/blob/master/LICENSE.txt>.

Este producto incluye software desarrollado por la Academic Free License (<http://www.opensource.org/licenses/afl-3.0.php>), la Common Development and Distribution License (<http://www.opensource.org/licenses/cddl1.php>), la Common Public License (<http://www.opensource.org/licenses/cpl1.0.php>), la Sun Binary Code License Agreement Supplemental License Terms, la BSD License (<http://www.opensource.org/licenses/bsd-license.php>), la nueva BSD License (<http://opensource.org/licenses/BSD-3-Clause>), la MIT License (<http://www.opensource.org/licenses/mit-license.php>), la Artistic License (<http://www.opensource.org/licenses/artistic-license-1.0>) y la Initial Developer's Public License Version 1.0 (<http://www.firebirdsql.org/en/initial-developer-s-public-license-version-1-0/>).

Este producto incluye copyright de software © 2003-2006 Joe Walnes, 2006-2007 XStream Committers. Todos los derechos reservados. Los permisos y las limitaciones relativos a este software están sujetos a los términos disponibles en la dirección <http://xstream.codehaus.org/license.html>. Este producto incluye software desarrollado por Indiana University Extreme! Lab. Para obtener más información, visite <http://www.extreme.indiana.edu/>.

Este producto incluye software Copyright © 2013 Frank Balluffi y Markus Moeller. Todos los derechos reservados. Los permisos y las limitaciones relativas a este software están sujetos a los términos de la licencia MIT.

AVISOS

Este producto de Informática (el "Software") incluye ciertos controladores (los "Controladores DataDirect") de DataDirect Technologies, una empresa operativa de Progress Software Corporation ("DataDirect") que están sujetos a los términos y condiciones siguientes:

1. LOS CONTROLADORES DATADIRECT SE PROPORCIONAN "TAL CUAL" SIN GARANTÍA DE NINGÚN TIPO, YA SEA EXPRESA O IMPLÍCITA, INCLUIDAS, ENTRE OTRAS, LAS GARANTÍAS IMPLÍCITAS DE NO INCUMPLIMIENTO, DE ADECUACIÓN PARA UN PROPÓSITO DETERMINADO Y DE VALIDEZ PARA EL COMERCIO.
2. EN NINGÚN CASO DATADIRECT NI SUS PROVEEDORES DE TERCEROS SERÁN RESPONSABLES ANTE EL USUARIO FINAL POR NINGÚN DAÑO DIRECTO, INDIRECTO, FORTUITO, ESPECIAL, CONSECUENTE, NI DE NINGÚN OTRO TIPO, RESULTANTE DEL USO DE LOS CONTROLADORES ODBC, INDEPENDIENTEMENTE DE SI SE HA AVISADO O NO DE LOS POSIBLES DAÑOS POR ADELANTADO. ESTAS LIMITACIONES SE APLICAN A TODAS LAS DEMANDAS JUDICIALES, INCLUIDAS, ENTRE OTRAS, AQUELLAS POR INCUMPLIMIENTO DE CONTRATO, INCUMPLIMIENTO DE LA GARANTÍA, NEGLIGENCIA, RESPONSABILIDAD ESTRICTA, TERGIVERSACIÓN Y OTROS AGRAVIOS.

La información contenida en esta documentación está sujeta a cambios sin previo aviso. Si encuentra algún problema en esta documentación, infórmenos por escrito a Informática LLC 2100 Seaport Blvd. Redwood City, CA 94063.

INFORMATICA LLC PROPORCIONA LA INFORMACIÓN DE ESTE DOCUMENTO "TAL CUAL" SIN GARANTÍA DE NINGÚN TIPO, EXPRESA O IMPLÍCITA, INCLUIDAS LAS GARANTÍAS DE COMERCIALIZACIÓN, ADAPTACIÓN A UN FIN PARTICULAR Y CUALQUIER GARANTÍA O CONDICIÓN DE NO INCUMPLIMIENTO.

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

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

Hay fragmentos de este software y/o documentación que están sujetas a copyright perteneciente a terceros, incluido, entre otros: Copyright DataDirect Technologies. Todos los derechos reservados. Copyright © Sun Microsystems. Todos los derechos reservados. Copyright © RSA Security Inc. Todos los derechos reservados. Copyright © Ordinal Technology Corp. Todos los derechos reservados. Copyright © Aandacht c.v. Todos los derechos reservados. Copyright Genivia, Inc. Todos los derechos reservados. Copyright Isomorphic Software. Todos los derechos reservados. Copyright © Meta Integration Technology, Inc. Todos los derechos reservados. Copyright © Intalio. Todos los derechos reservados. Copyright © Oracle. Todos los derechos reservados. Copyright © Adobe Systems Incorporated. Todos los derechos reservados. Copyright © DataArt, Inc. Todos los derechos reservados. Copyright © ComponentSource. Todos los derechos reservados. Copyright © Microsoft Corporation. Todos los derechos reservados. Copyright © Rogue Wave Software, Inc. Todos los derechos reservados. Copyright © Teradata Corporation. Todos los derechos reservados. Copyright © Yahoo! Inc. Todos los derechos reservados. Copyright © Glyph & Cog, LLC. Todos los derechos reservados. Copyright © Thinkmap, Inc. Todos los derechos reservados. Copyright © Clearpace Software Limited. Todos los derechos reservados. Copyright © Information Builders, Inc. Todos los derechos reservados. Copyright © OSS Nokalva, Inc. Todos los derechos reservados. Copyright Edifecs, Inc. Todos los derechos reservados. Copyright Cleo Communications, Inc. Todos los derechos reservados. Copyright © International Organization for Standardization 1986. Todos los derechos reservados. Copyright © ej-technologies GmbH. Todos los derechos reservados. Copyright © Jaspersoft Corporation. Todos los derechos reservados. Copyright © International Business Machines Corporation. Todos los derechos reservados. Copyright © yWorks GmbH. Todos los derechos reservados. Copyright © Lucent Technologies. Todos los derechos reservados. Copyright © University of Toronto. Todos los derechos reservados. Copyright © Daniel Veillard. Todos los derechos reservados. Copyright © Unicode, Inc. Copyright IBM Corp. Todos los derechos reservados. Copyright © MicroQuill Software Publishing, Inc. Todos los derechos reservados. Copyright © PassMark Software Pty Ltd. Todos los derechos reservados. Copyright © LogiXML, Inc. Todos los derechos reservados. Copyright © 2003-2010 Lorenzi Davide. Todos los derechos reservados. Copyright © Red Hat, Inc. Todos los derechos reservados. Copyright © The Board of Trustees of the Leland Stanford Junior University. Todos los derechos reservados. Copyright © EMC Corporation. Todos los derechos reservados. Copyright © Flexera Software. Todos los derechos reservados. Copyright © Jinfonet Software. Todos los derechos reservados. Copyright © Apple Inc. Todos los derechos reservados. Copyright © Telerik Inc. Todos los derechos reservados. Copyright © BEA Systems. Todos los derechos reservados. Copyright © PDFlib GmbH. Todos los derechos reservados. Copyright © Orientation in Objects GmbH. Todos los derechos reservados. Copyright © Tanuki Software, Ltd. Todos los derechos reservados. Copyright © Ricebridge. Todos los derechos reservados. Copyright © Sencha, Inc. Todos los derechos reservados. Copyright © Scalable Systems, Inc. Todos los derechos reservados. Copyright © jQWidgets. Todos los derechos reservados. Copyright © Tableau Software, Inc. Todos los derechos reservados. Copyright © MaxMind, Inc. Todos los derechos reservados. Copyright © Tmate Software s.r.o. Todos los derechos reservados. Copyright © MapR Technologies Inc. Todos los derechos reservados. Copyright © Amazon Corporate LLC. Todos los derechos reservados. Copyright © Highsoft. Todos los derechos reservados. Copyright © Python Software Foundation. Todos los derechos reservados. Copyright © BeOpen.com. Todos los derechos reservados. Copyright © CNRI. Todos los derechos reservados.

Este producto incluye software desarrollado por la Apache Software Foundation (<http://www.apache.org/>) y/u otro software protegido por varias versiones de la licencia Apache License ("Licencia"). Puede obtener una copia de estas licencias en <http://www.apache.org/licenses/>. A menos que las leyes aplicables lo requieran o se haya acordado por escrito, el software distribuido bajo estas licencias se distribuye "TAL CUAL", SIN GARANTÍAS NI CONDICIONES DE NINGÚN TIPO, ya sea expresas o implícitas. Consulte las licencias del idioma específico para conocer los permisos y las limitaciones que rigen según las licencias.

Este producto incluye software desarrollado por Mozilla (<http://www.mozilla.org/>), copyright del software de The JBoss Group, LLC, todos los derechos reservados; copyright del software © 1999-2006 de Bruno Lowagie y Paulo Soares y otro software protegido por licencia por el acuerdo GNU Lesser General Public License Agreement, que se puede encontrar en la dirección <http://www.gnu.org/licenses/lgpl.html>. Los materiales se facilitan gratuitamente por parte de Informática, "tal cual", sin garantía de ningún tipo, ya sea expresa o implícita, incluidas, entre otras, las garantías implícitas de adecuación para un propósito determinado y de validez para el comercio.

El producto incluye software ACE(TM) y TAO(TM) con copyright de Douglas C. Schmidt y su grupo de investigación de la Washington University, University of California, Irvine y Vanderbilt University, Copyright (©) 1993-2006, todos los derechos reservados.

Este producto incluye software desarrollado por el OpenSSL Project para uso en el OpenSSL Toolkit (copyright The OpenSSL Project. Todos los derechos reservados) y la redistribución de este software está sujeta a los términos especificados en <http://www.openssl.org> y <http://www.openssl.org/source/license.html>.

Este producto incluye software Curl con Copyright 1996-2013, Daniel Stenberg, <daniel@haxx.se>. Todos los derechos reservados. Los permisos y las limitaciones relativos a este software están sujetos a los términos disponibles en la dirección <http://curl.haxx.se/docs/copyright.html>. La autorización para utilizar, copiar, modificar y distribuir este software para cualquier propósito con o sin tasas se concede por el presente, siempre que el aviso de copyright anterior y este aviso de permiso aparezcan en todas las copias.

El producto incluye copyright de software 2001-2005 (©) MetaStuff, Ltd. Todos los derechos reservados. Los permisos y las limitaciones relativos a este software están sujetos a los términos disponibles en la dirección <http://www.dom4j.org/license.html>.

Este producto incluye copyright de software © 1996-2006 Per Bothner. Todos los derechos reservados. Su derecho a utilizar estos materiales está establecido en la licencia que puede encontrarse en la dirección <http://www.gnu.org/software/kawa/Software-License.html>.

Este producto incluye software OSSP UUID con Copyright © 2002 Ralf S. Engelschall, Copyright © 2002 The OSSP Project Copyright © 2002 Cable & Wireless Deutschland. Los permisos y las limitaciones relativas a este software están sujetos a los términos disponibles en la dirección <http://www.opensource.org/licenses/mit-license.php>.

Este producto incluye software desarrollado por Boost (<http://www.boost.org/>) o protegido por la licencia de software de Boost. Los permisos y las limitaciones relativos a este software están sujetos a los términos disponibles en la dirección http://www.boost.org/LICENSE_1_0.txt.

Este producto incluye copyright de software © 1997-2007 University of Cambridge. Los permisos y las limitaciones relativos a este software están sujetos a los términos disponibles en la dirección <http://www.pcre.org/license.txt>.

Este producto incluye copyright de software © 2007 The Eclipse Foundation. Todos los derechos reservados. Los permisos y las limitaciones relativos a este software están sujetos a los términos especificados en <http://www.eclipse.org/org/documents/epl-v10.php> y <http://www.eclipse.org/org/documents/edl-v10.php>.

Este producto incluye software protegido por licencia según los términos que aparecen en <http://www.tcl.tk/software/tcltk/license.html>, <http://www.bosrup.com/web/overlib?License>, <http://www.stlport.org/doc/license.html>, <http://asm.ow2.org/license.html>, <http://www.cryptix.org/LICENSE.TXT>, <http://hsqldb.org/web/hsqLicense.html>, <http://httpunit.sourceforge.net/doc/license.html>, <http://jung.sourceforge.net/license.txt>, http://www.gzip.org/zlib/zlib_license.html, <http://www.openldap.org/software/release/license.html>, <http://www.libssh2.org>, <http://slf4j.org/license.html>, <http://www.sente.ch/software/OpenSourceLicense.html>, <http://fusesource.com/downloads/license-agreements/fuse-message-broker-v-5-3-license-agreement>, <http://antlr.org/license.html>, <http://aopalliance.sourceforge.net/>, <http://www.bouncycastle.org/licence.html>, <http://www.jgraph.com/jgraphdownload.html>, <http://www.jcraft.com/jsch/LICENSE.txt>, http://jotm.objectweb.org/bsd_license.html, <http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231>, <http://www.slf4j.org/license.html>, <http://nanoxml.sourceforge.net/orig/copyright.html>, <http://www.json.org/license.html>, <http://forge.ow2.org/projects/javaxservice/>, <http://www.postgresql.org/about/licence.html>, <http://www.sqlite.org/copyright.html>, <http://www.tcl.tk/software/tcltk/license.html>, <http://www.jaxen.org/faq.html>, <http://www.jdom.org/docs/faq.html>, <http://www.slf4j.org/license.html>, <http://www.iodbc.org/dataspace/iodbc/wiki/IODBC/License>, <http://www.keplerproject.org/md5/license.html>, <http://www.toedter.com/en/jcalendar/license.html>, <http://www.edankert.com/bounce/index.html>, <http://www.net-snmp.org/about/license.html>, <http://www.openmdx.org/FAQ>, http://www.php.net/license/3_01.txt, <http://srp.stanford.edu/license.txt>, <http://www.schneier.com/blowfish.html>, <http://www.jmock.org/license.html>, <http://xsom.java.net>, <http://benalman.com/about/license/>, <https://github.com/CreateJS/EaselJS/blob/master/src/easeljs/display/Bitmap.js>, <http://www.h2database.com/html/license.html#summary>, <http://jsoncpp.sourceforge.net/LICENSE>, <http://jdbc.postgresql.org/license.html>, <http://protobuf.googlecode.com/svn/trunk/src/google/protobuf/descriptor.proto>, <https://github.com/rantav/hector/blob/master/LICENSE>, <http://web.mit.edu/Kerberos/krb5-current/doc/mitK5license.html>, <http://jibx.sourceforge.net/jibx-license.html>, <https://github.com/lyokato/libgeohash/blob/master/LICENSE>, <https://code.google.com/p/lz4/>, <https://github.com/jedisct1/libsodium/blob/master/LICENSE>, <http://one-jar.sourceforge.net/index.php?page=documents&file=license>, <https://github.com/EsotericSoftware/kryo/blob/master/license.txt>, <http://www.scala-lang.org/license.html>, <https://github.com/tinkerpop/blueprints/blob/master/LICENSE.txt>, <http://gee.cs.oswego.edu/dl/classes/EDU/oswego/cs/dl/util/concurrent/intro.html>, <https://aws.amazon.com/asl/>, <https://github.com/twbs/bootstrap/blob/master/LICENSE>, <https://sourceforge.net/p/xmlunit/code/HEAD/tree/trunk/LICENSE.txt>, <https://github.com/documentcloud/underscore-contrib/blob/master/LICENSE> y <https://github.com/apache/hbase/blob/master/LICENSE.txt>.

Este producto incluye software desarrollado por la Academic Free License (<http://www.opensource.org/licenses/afl-3.0.php>), la Common Development and Distribution License (<http://www.opensource.org/licenses/cddl1.php>), la Common Public License (<http://www.opensource.org/licenses/cpl1.0.php>), la Sun Binary Code License Agreement Supplemental License Terms, la BSD License (<http://www.opensource.org/licenses/bsd-license.php>), la nueva BSD License (<http://opensource.org/licenses/BSD-3-Clause>), la MIT License (<http://www.opensource.org/licenses/mit-license.php>), la Artistic License (<http://www.opensource.org/licenses/artistic-license-1.0>) y la Initial Developer's Public License Version 1.0 (<http://www.firebirdsql.org/en/initial-developer-s-public-license-version-1-0/>).

Este producto incluye copyright de software © 2003-2006 Joe Walnes, 2006-2007 XStream Committers. Todos los derechos reservados. Los permisos y las limitaciones relativos a este software están sujetos a los términos disponibles en la dirección <http://xstream.codehaus.org/license.html>. Este producto incluye software desarrollado por Indiana University Extreme! Lab. Para obtener más información, visite <http://www.extreme.indiana.edu/>.

Este producto incluye software Copyright © 2013 Frank Balluffi y Markus Moeller. Todos los derechos reservados. Los permisos y las limitaciones relativas a este software están sujetos a los términos de la licencia MIT.

Consulte las patentes en <https://www.informatica.com/legal/patents.html>.

EXENCIÓN DE RESPONSABILIDAD: Informática LLC proporciona esta documentación "tal cual" sin garantía de ningún tipo, ya sea expresa o implícita, incluidas, entre otras, las garantías implícitas de no incumplimiento, de adecuación para un propósito determinado y de validez para el comercio. Informática LLC no garantiza que este software o esta documentación estén libres de errores. La información proporcionada en este software o en esta documentación puede contener imprecisiones técnicas o errores tipográficos. La información de este software y esta documentación está sujeta a cambios en cualquier momento sin previo aviso.

AVISOS

Este producto de Informatica (el "Software") incluye ciertos controladores (los "Controladores DataDirect") de DataDirect Technologies, una empresa operativa de Progress Software Corporation ("DataDirect") que están sujetos a los términos y condiciones siguientes:

1. LOS CONTROLADORES DATADIRECT SE PROPORCIONAN "TAL CUAL" SIN GARANTÍA DE NINGÚN TIPO, YA SEA EXPRESA O IMPLÍCITA, INCLUIDAS, ENTRE OTRAS, LAS GARANTÍAS IMPLÍCITAS DE NO INCUMPLIMIENTO, DE ADECUACIÓN PARA UN PROPÓSITO DETERMINADO Y DE VALIDEZ PARA EL COMERCIO.
2. EN NINGÚN CASO DATADIRECT NI SUS PROVEEDORES DE TERCEROS SERÁN RESPONSABLES ANTE EL USUARIO FINAL POR NINGÚN DAÑO DIRECTO, INDIRECTO, FORTUITO, ESPECIAL, CONSECUENTE, NI DE NINGÚN OTRO TIPO, RESULTANTE DEL USO DE LOS CONTROLADORES ODBC, INDEPENDIENTEMENTE DE SI SE HA AVISADO O NO DE LOS POSIBLES DAÑOS POR ADELANTADO. ESTAS LIMITACIONES SE APLICAN A TODAS LAS DEMANDAS JUDICIALES, INCLUIDAS, ENTRE OTRAS, AQUELLAS POR INCUMPLIMIENTO DE CONTRATO, INCUMPLIMIENTO DE LA GARANTÍA, NEGLIGENCIA, RESPONSABILIDAD Estricta, TERGIVERSACIÓN Y OTROS AGRAVIOS.

La información contenida en esta documentación está sujeta a cambios sin previo aviso. Si encuentra algún problema en esta documentación, infórmenos por escrito a Informatica LLC 2100 Seaport Blvd. Redwood City, CA 94063.

INFORMATICA LLC PROPORCIONA LA INFORMACIÓN DE ESTE DOCUMENTO "TAL CUAL" SIN GARANTÍA DE NINGÚN TIPO, EXPRESA O IMPLÍCITA, INCLUIDAS LAS GARANTÍAS DE COMERCIALIZACIÓN, ADAPTACIÓN A UN FIN PARTICULAR Y CUALQUIER GARANTÍA O CONDICIÓN DE NO INCUMPLIMIENTO.

Fecha de publicación: 2018-07-05

Tabla de contenido

Prefacio	13
Documentación de Informatica	13
Informatica Network.	13
Base de conocimiento de Informatica.	13
Documentación de Informatica	13
Matrices de disponibilidad de productos de Informatica.	14
Informatica Velocity.	14
Catálogo de soluciones de Informatica.	14
Servicio internacional de atención al cliente de Informatica.	14
 Capítulo 1: Lenguaje de transformación.....	15
Resumen del idioma de transformación.	15
Componentes del idioma de transformación.	15
Internacionalización y el idioma de transformación.	16
Sintaxis de expresiones.	17
Componentes de expresiones.	17
Reglas y pautas sobre la sintaxis de expresiones.	19
Agregar comentarios a las expresiones.	20
Palabras reservadas.	20
 Capítulo 2: Constantes.....	22
DD_DELETE.	22
Ejemplo.	22
DD_INSERT.	23
Ejemplos.	23
DD_REJECT.	23
Ejemplos.	24
DD_UPDATE.	24
Ejemplos.	24
FALSE.	25
Ejemplo.	25
NULL.	25
Modo de trabajo con valores nulos en expresiones booleanas.	25
Valores nulos en expresiones de comparación.	25
Valores nulos en funciones de agregado.	26
Valores nulos en condiciones de filtro.	26
Valores nulos con operadores.	27
TRUE.	27
Ejemplo.	27

Capítulo 3: Operadores.....	28
Precedencia del operador.	28
Operadores aritméticos.	29
Operadores de cadena.	30
Valores nulos.	30
Ejemplo.	30
Operadores de comparación.	31
Operadores lógicos.	31
Valores nulos.	32
 Capítulo 4: Variables.....	 33
Variables integradas.	33
\$PM<SourceName>@TableName, \$PM<TargetName>@TableName.	35
\$PMFolderName.	36
\$PMIntegrationServiceName.	36
\$PMMappingName.	36
\$PMRepositoryServiceName.	36
\$PMRepositoryUserName.	36
\$PMSessionName.	36
\$PMSessionRunMode.	37
\$PMWorkflowName.	37
\$PMWorkflowRunId.	37
\$PMWorkflowRunInstanceName.	37
SESSSTARTTIME.	37
SYSDATE.	38
WORKFLOWSTARTTIME.	38
Variables de control de transacción.	39
Variables locales.	39
 Capítulo 5: Fechas.....	 40
Introducción a las fechas.	40
Tipo de datos de fecha y hora.	40
Día juliano, día juliano modificado y calendario gregoriano.	41
Fechas del año 2000.	41
Fechas de bases de datos relacionales.	43
Fechas de archivos sin formato.	43
Formato de fecha predeterminado.	43
Cadenas de formato de fecha.	44
Cadenas de formato TO_CHAR.	46
Ejemplos.	48
Cadenas de formato TO_DATE e IS_DATE.	49
Normas y directrices para cadenas con formato de fecha.	51

Ejemplo.	51
Descripción de operaciones aritméticas con fechas.	53
Capítulo 6: Funciones.	54
Categorías de funciones.	54
Funciones agregadas.	54
Funciones de agregado y valores nulos.	56
Funciones de caracteres.	56
Funciones de conversión.	57
Funciones de limpieza de datos.	57
Funciones de fecha.	58
Funciones de codificación.	59
Funciones financieras.	59
Funciones numéricas.	59
Funciones científicas.	60
Funciones especiales.	60
Funciones de cadena.	61
Funciones de prueba.	61
Funciones de variables.	61
ABORT.	62
ABS.	62
ADD_TO_DATE.	64
AES_DECRYPT.	67
AES_ENCRYPT.	67
ANY.	68
ASCII.	69
AVG.	70
CEIL.	72
CHOOSE.	73
CHR.	74
CHRCODE.	75
COMPRESS.	76
CONCAT.	76
CONVERT_BASE.	78
COS.	78
COSH.	79
COUNT.	80
CRC32.	83
CREATE_TIMESTAMP_TZ.	83
CUME.	84
DATE_COMPARE.	86
DATE_DIFF.	87
DEC_BASE64.	90

DECODE.	91
DECOMPRESS.	93
ENC_BASE64.	94
ERROR.	94
EXP.	96
FIRST.	97
FLOOR.	98
FV.	99
GET_DATE_PART.	100
GET_TIMEZONE.	102
GET_TIMESTAMP.	103
GREATEST.	104
IIF.	105
IN.	108
INDEXOF.	109
INITCAP.	110
INSTR.	111
ISNULL.	114
IS_DATE.	115
IS_NUMBER.	118
IS_SPACES.	120
LAST.	121
LAST_DAY.	122
LEAST.	123
LENGTH.	124
LN.	125
LOG.	126
LOOKUP.	127
LOWER.	128
LPAD.	129
LTRIM.	130
MAKE_DATE_TIME.	132
MAX (Fechas).	133
MAX (Números).	134
MAX (Cadena).	135
MD5.	137
MEDIAN.	138
METAPHONE.	139
MIN (Fechas).	143
MIN (Números).	144
MIN (Cadena).	145
MOD.	147

MOVINGAVG.	148
MOVINGSUM.	150
NPER.	151
PERCENTILE.	152
PMT.	154
POWER.	155
PV.	156
RAND.	156
RATE.	157
REG_EXTRACT.	158
REG_MATCH.	161
REG_REPLACE.	162
REPLACECHR.	163
REPLACESTR.	166
REVERSE.	169
ROUND (Fechas).	170
ROUND (Números).	174
RPAD.	177
RTRIM.	178
SETCOUNTVARIABLE.	179
SET_DATE_PART.	181
SETMAXVARIABLE.	184
SETMINVARIABLE.	185
SETVARIABLE.	187
SIGN.	189
SIN.	190
SINH.	191
SOUNDEX.	192
SQL_LIKE.	194
SQRT.	195
STDDEV.	196
SUBSTR.	198
SUM.	200
SYSTIMESTAMP.	201
TAN.	203
TANH.	203
TO_BIGINT.	204
TO_CHAR (Fechas).	206
TO_CHAR (Números).	211
TO_DATE.	213
TO_DECIMAL.	216
TO_DECIMAL38.	218

TO_FLOAT.	219
TO_INTEGER.	220
TO_TIMESTAMP_TZ.	222
TRUNC (Fechas).	224
TRUNC (Números).	227
UPPER.	228
UUID4.	229
UUID_UNPARSE.	230
VARIANCE.	230

Capítulo 7: Creación de funciones personalizadas..... 232

Introducción a la creación de funciones personalizadas.	232
Pasos para la creación de funciones personalizadas.	233
Instalación de funciones personalizadas.	233
Paso 1. Obtención de atributos de ID de repositorio.	233
Paso 2. Creación de un archivo de encabezado.	234
Paso 3. Creación de un archivo de implementación.	236
Paso 4. Compilación de módulos.	245
Generación del módulo en Windows.	246
Compilación de un módulo en UNIX.	247
Paso 5. Creación de un archivo de complemento de repositorio.	247
El elemento PLUGIN.	248
El elemento FUNCTION_GROUP.	248
Determinación de un espacio de nombres.	249
Elemento FUNCTION.	249
Elemento LIBRARY.	250
Archivo XML de complemento de ejemplo.	251
Paso 6. Prueba de funciones personalizadas.	251
Validación del archivo de complemento de repositorio.	251
Comprobación de la exactitud de las funciones.	252
Instalación de funciones personalizadas.	252
Paso 1. Copia de bibliotecas de funciones personalizadas en PowerCenter.	252
Paso 2. Registro del complemento.	253
Creación de expresiones con funciones personalizadas.	253

Capítulo 8: Referencia de API de funciones personalizadas..... 254

Introducción a la referencia de API de funciones personalizadas.	254
API comunes.	254
Identificador de validación.	255
Función de validación de interfaz de usuario.	255
Estructura de INFA_EXPR_OPD_METADATA.	257
Función de obtención de versión de complemento.	258
API en tiempo de ejecución.	259

Funciones de nivel de módulo.	259
Funciones de nivel de función.	261
Funciones de nivel de instancia de función.	262
Índice.	265

Prefacio

La *Referencia del lenguaje de transformación de Informatica Developer* se ha redactado para los desarrolladores responsables de la creación de asignaciones. En la *Referencia del lenguaje de transformación de Informatica Developer* se da por supuesto que tiene conocimientos sobre SQL, los conceptos de las bases de datos relacionales y los requisitos de la interfaz para las aplicaciones auxiliares.

Documentación de Informatica

Informatica Network

Informatica Network incluye el servicio internacional de atención al cliente de Informatica, la base de conocimiento de Informatica y otros recursos de producto. Para acceder a Informatica Network, visite <https://network.informatica.com>.

Un miembro puede:

- Acceder a todos sus recursos de Informatica en un solo lugar.
- Busque recursos de producto, como documentación, preguntas frecuentes y mejores prácticas en la base de conocimiento.
- Vea la información de disponibilidad del producto.
- Revisar los casos de asistencia.
- Buscar su red de grupos de usuarios de Informatica locales y colaborar con sus iguales.

Base de conocimiento de Informatica

Utilice la base de conocimiento de Informatica para buscar recursos de producto como documentación, artículos de procedimientos, mejores prácticas y PAM en la red de Informatica.

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

Documentación de Informatica

Para obtener la documentación más reciente del producto, consulte la base de conocimiento de Informatica en https://kb.informatica.com/_layouts/ProductDocumentation/Page/ProductDocumentSearch.aspx.

Si tiene preguntas, comentarios o ideas relacionadas con esta documentación, póngase en contacto con el equipo de documentación de Informatica enviando un correo electrónico a infa_documentation@informatica.com.

Matrices de disponibilidad de productos de Informatica

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

Informatica Velocity

Informatica Velocity es un conjunto de sugerencias y mejores prácticas desarrollado por los servicios profesionales de Informatica. Desarrollado a partir de la experiencia real de cientos de proyectos de administración de datos, Informatica Velocity representa el conocimiento conjunto de nuestros asesores, los cuales han trabajado con organizaciones de todo el mundo para planificar, desarrollar, implementar y mantener con éxito soluciones de administración de datos.

Si es miembro de la red de Informatica, puede acceder a los recursos de Informatica Velocity en <http://velocity.informatica.com>.

Si tiene alguna pregunta, comentario o idea acerca de Informatica Velocity, póngase en contacto con los servicios Profesionales de Informatica en ips@informatica.com.

Catálogo de soluciones de Informatica

El Catálogo de soluciones de Informatica es un foro donde puede buscar soluciones que aumenten, amplíen o mejoren sus implementaciones de Informatica. Al aprovechar cualquiera de los cientos de soluciones de los desarrolladores y los socios de Informatica, puede mejorar la productividad y acelerar el tiempo de implementación en los proyectos. Puede acceder al Catálogo de soluciones de Informatica en <https://marketplace.informatica.com>.

Servicio internacional de atención al cliente de Informatica

Puede ponerse en contacto con un centro de atención global por teléfono o a través del soporte en línea en la red de Informatica.

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

<http://www.informatica.com/us/services-and-training/support-services/global-support-centers>.

Si es miembro de la red de Informatica, puede utilizar el soporte en línea en <http://network.informatica.com>.

CAPÍTULO 1

Lenguaje de transformación

Este capítulo incluye los siguientes temas:

- [Resumen del idioma de transformación, 15](#)
- [Sintaxis de expresiones, 17](#)
- [Agregar comentarios a las expresiones, 20](#)
- [Palabras reservadas, 20](#)

Resumen del idioma de transformación

PowerCenter® proporciona un idioma de transformación que incluye funciones de tipo SQL para transformar los datos de origen. Utilice estas funciones para escribir expresiones y crear funciones llamadas funciones definidas por el usuario.

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

Las funciones definidas por el usuario reutilizan la lógica de la expresión y construyen expresiones complejas. Puede incluirlas en otras funciones definidas por el usuario o en expresiones. Las funciones definidas por el usuario siguen las mismas pautas que las expresiones. Utilizan la misma sintaxis y pueden utilizar los mismos componentes de idioma de transformación.

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

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

Componentes del idioma de transformación

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

- **Funciones.** Más de 100 funciones de tipo SQL le permiten cambiar los datos en una asignación.
- **Operadores.** Use los operadores de transformación para crear expresiones de transformación para realizar cálculos matemáticos, combinar o comparar datos.
- **Constantes.** Utilice las constantes integradas para hacer referencia a valores que permanecen constantes como TRUE.

- **Parámetros de asignación y variables.** Crea parámetros de asignación para su uso dentro de una asignación o un mapplet para hacer referencia a valores que permanecen constantes durante una sesión como, por ejemplo, un tipo de impuesto sobre las ventas. Crea las variables de asignación en mapplets o asignaciones para escribir expresiones que hagan referencia a valores que cambian de una sesión a otra.
- **Parámetros de asignación.** Crea parámetros de asignación para su uso dentro de una asignación o un mapplet para hacer referencia a valores que permanecen constantes durante una ejecución de asignación o de mapplet como, por ejemplo, un tipo de impuesto sobre las ventas.
- **Variables de flujo de trabajo.** Crea variables de flujo de trabajo para su uso dentro de flujos de trabajo para escribir expresiones que hagan referencia a valores que cambian de un flujo de trabajo a otro.
- **Variables incorporadas y locales.** Utilice variables integradas para escribir expresiones que hagan referencia a valores que cambian como, por ejemplo, la fecha del sistema. También puede crear variables locales en transformaciones.
- **Valores de retorno.** También puede escribir expresiones que incluyan las transformaciones de búsqueda de valores de retorno.

Internacionalización y el idioma de transformación

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

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

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

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

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

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

Sintaxis de expresiones

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

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

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

Componentes de expresiones

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

- Puertos (entrada, entrada/salida, variable)
- Literales de cadena, literales numéricos
- Constantes
- Funciones
- Variables integradas y locales
- Parámetros de asignación y variables de asignación
- Parámetros de asignación
- Variables de flujo de trabajo predefinidas
- Variables de flujo de trabajo definidas por el usuario
- Operadores
- Valores devueltos

Puertos y valores de retorno

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

Calificador de referencia	Descripción
:EXT	Obligatorio cuando se crea una expresión que incluye un valor de retorno de una transformación de procedimiento externo. La sintaxis general es: :EXT.external_procedure_transformation(argument1, argument2, ...)
:LKP	Obligatorio cuando se crea una expresión que incluye el valor de retorno de una transformación de búsqueda no conectada. La sintaxis general es: :LKP.lookup_transformation(argument1, argument2, ...) Los argumentos son los puertos locales usados en la condición de búsqueda. El orden debe ser el mismo que el orden de los puertos de la transformación. Los tipos de datos para los puertos locales tienen que coincidir con el tipo de datos de los puertos de búsqueda empleados en la condición de búsqueda.
:SD	Opcional (expresiones PowerMart 3.5 solamente). Califica un puerto de tabla de origen en una expresión. La sintaxis general es: :SD.source_table.column_name
:SEQ	Obligatorio cuando se crea una expresión que incluye un puerto en una transformación de generador de secuencias. La sintaxis general es: :SEQ.sequence_generator_transformation.CURRVAL
:SP	Obligatorio cuando se escribe una expresión que incluye el valor de retorno de una transformación de procedimiento almacenado no conectada. La sintaxis general es: :SP.stored_procedure_transformation(argument1, argument2, [, PROC_RESULT]) Los argumentos tienen que coincidir con los argumentos de la transformación de procedimiento almacenado no conectada.
:TD	Obligatorio cuando se hace referencia a una tabla de destino en una función LOOKUP PowerMart 3.5. La sintaxis general es: LOOKUP(:TD.SALES.ITEM_NAME, :TD.SALES.ITEM_ID, 10, :TD.SALES.PRICE, 15.99)

Literales de cadena y literales numéricos

Puede incluir literales numéricos o de cadena.

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

```
'Alice Davis'
```

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

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

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

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

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

`.05`

O bien:

`$$Sales_Tax`

Reglas y pautas sobre la sintaxis de expresiones

Utilice las siguientes reglas y pautas para escribir expresiones:

- No se pueden incluir simultáneamente expresiones agregadas de un nivel y agregadas en una transformación de agregación.
- Si necesita crear tanto funciones de un nivel como anidadas, cree transformaciones de agregación separadas.
- No se pueden usar cadenas en expresiones numéricas.
Por ejemplo, la expresión `1 + '1'` no es válida porque solamente se puede realizar la suma en tipos de datos numéricos. No se puede añadir un entero y una cadena.
- No se pueden usar cadenas como parámetros numéricos.
Por ejemplo, la expresión `SUBSTR(TEXT_VAL, '1', 10)` no es válida porque la función `SUBSTR` requiere un valor entero, no una cadena, como posición inicial.
- No se pueden mezclar tipos de datos cuando se usan operadores de comparación.
Por ejemplo, la expresión `123,4 = '123,4'` no es válida porque compara un valor decimal con una cadena.
- Se puede pasar un valor desde un puerto, literal de cadena o número, variable, transformación Lookup, transformación Stored Procedure, transformación External Procedure o los resultados de otra expresión.
- Se puede pasar un valor de un puerto, una cadena o un número literales, una transformación de búsqueda o los resultados de otra expresión.
- Utilice la pestaña Puertos del Editor de expresiones para introducir un nombre de puerto en una expresión. Si cambia el nombre de un puerto en una transformación conectada, Designer propaga el cambio de nombre en las expresiones de la transformación.
- Utilice la pestaña Puertos del Editor de expresiones para introducir un nombre de puerto en una expresión. Si cambia el nombre de un puerto en una transformación conectada, la herramienta Developer propaga el cambio de nombre en las expresiones de la transformación.
- Separe cada argumento de una función con una coma.
- Exceptuando los literales, el lenguaje de transformación no distingue entre mayúsculas y minúsculas.
- Exceptuando los literales, Designer y el Servicio de Integración PowerCenter omiten los espacios.
- Exceptuando los literales, la herramienta Developer y el Servicio de Integración de Datos omiten los espacios.
- El signo de dos puntos (:), la coma (,) y el punto (.) tienen un significado especial y solamente deben usarse para especificar la sintaxis.
- El Servicio de integración de datos trata el guión (-) como operador de resta.
- Si se pasa un valor literal a una función, delimite las cadenas literales con comillas simples. No utilice comillas para números literales. El Servicio de integración de datos trata cualquier valor de cadena delimitado por comillas simples como una cadena de caracteres.
- Cuando se pasa un parámetro de asignación o una variable a una función dentro de una expresión, no utilice comillas para designar parámetros de asignación o variables de flujo de trabajo.

- Cuando se pasa un parámetro de asignación a una función dentro de una expresión, no utilice comillas para designar parámetros de asignación.
- No utilice comillas para designar puertos.
- Se pueden anidar múltiples funciones dentro de una expresión exceptuando funciones agregadas, que solamente permiten una función agregada anidada. El Servicio de integración de datos evalúa la expresión a partir de la función más interna.

Agregar comentarios a las expresiones

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

- Dos guiones, como en:

```
-- These are comments
```

- Dos barras, como en:

```
// These are comments
```

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

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

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

```
FIRST_NAME || LAST_NAME
```

Un comentario no puede continuar en una nueva línea:

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

En este caso, Designer y Workflow Manager no validan la expresión, ya que la última línea no es una expresión válida.

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

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

Palabras reservadas

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

- :EXT

- :INFA
- :LKP
- :MCR
- :SD
- :SEQ
- :SP
- :TD
- AND
- DD_DELETE
- DD_INSERT
- DD_REJECT
- DD_UPDATE
- FALSE
- NOT
- NULL
- OR
- PROC_RESULT
- SESSSTARTTIME
- SPOUTPUT
- SYSDATE
- TRUE
- WORKFLOWSTARTTIME

Las siguientes palabras están reservadas para expresiones de flujo de trabajo:

- ABORTED
- DISABLED
- FAILED
- NOTSTARTED
- STARTED
- STOPPED
- SUCCEEDED

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

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

CAPÍTULO 2

Constantes

Este capítulo incluye los siguientes temas:

- [DD_DELETE, 22](#)
- [DD_INSERT, 23](#)
- [DD_REJECT, 23](#)
- [DD_UPDATE, 24](#)
- [FALSE, 25](#)
- [NULL, 25](#)
- [TRUE, 27](#)

DD_DELETE

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

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

Al ejecutar un flujo de trabajo, seleccione la estrategia de actualización controlada por datos para eliminar los registros de un destino en función de este indicador.

Ejemplo

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

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

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

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

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

DD_INSERT

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

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

Al ejecutar un flujo de trabajo, seleccione la estrategia de actualización controlada por datos para escribir los registros en un destino en función de este indicador.

Ejemplos

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

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

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

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

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

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

La siguiente expresión de estrategia de actualización usa SESSSTARTTIME para buscar solamente los pedidos enviados en los dos últimos días y marcarlos para su inserción. Mediante DATE_DIFF, la expresión resta DATE_SHIPPED de la fecha del sistema y devuelve la diferencia entre las dos fechas. Debido a que DATE_DIFF devuelve un valor doble, la expresión utiliza TRUNC para truncar la diferencia. A continuación, compara el resultado con el literal entero 2. Si el resultado es superior a 2, la expresión marca los registros para su rechazo. Si el resultado es 2 o inferior, la expresión marca los registros para su inserción.

```
IIF( TRUNC( DATE_DIFF( SESSSTARTTIME, ORDERS_DATE_SHIPPED, 'DD' ), 0 ) > 2, DD_REJECT, DD_INSERT )
```

DD_REJECT

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

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

Al ejecutar un flujo de trabajo, seleccione la estrategia de actualización controlada por datos para rechazar los registros de un destino en función de este indicador.

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

Ejemplos

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

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

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

Esta expresión usa literales numéricos para generar el mismo resultado:

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

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

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

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

DD_UPDATE

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

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

Al ejecutar un flujo de trabajo, seleccione la estrategia de actualización controlada por datos para escribir los registros en un destino en función de este indicador.

Ejemplos

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

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

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

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

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

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

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

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


FALSE

Aclara una expresión condicional. FALSE equivale al entero 0.

Ejemplo

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

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

NULL

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

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

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

TEMAS RELACIONADOS

- [“Funciones” en la página 54](#)

Modo de trabajo con valores nulos en expresiones booleanas

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

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

Valores nulos en expresiones de comparación

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

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

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

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

Cuando se utiliza un valor nulo en una expresión que contiene un operador de comparación, el Servicio de integración de datos genera un valor nulo. Sin embargo, también puede configurar el Servicio de integración de datos para tratar los valores nulos como altos o bajos en las operaciones de comparación.

Utilice la propiedad Tratamiento de nulos en operadores de comparación como para configurar el modo en el que el Servicio de integración de datos gestiona los valores nulos en expresiones de comparación.

Esta propiedad de configuración del Servicio de integración de datos afecta al comportamiento de los siguientes operadores de comparación en expresiones:

`=, !=, ^=, <>, >, >=, <, <=`

Por ejemplo, considere las siguientes expresiones:

```
NULL > 1
NULL = NULL
```

La tabla siguiente describe cómo el Servicio de integración de datos evalúa las expresiones:

Expresión	Tratamiento de nulos en operadores de comparación como		
	NULL	HIGH	LOW
NULL > 1	NULL	TRUE	FALSE
NULL = NULL	NULL	TRUE	TRUE

Valores nulos en funciones de agregado

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

El Servicio de integración de datos trata los valores nulos como nulos en las funciones de agregado. Si pasa un puerto completo o un grupo de valores nulos, la función devuelve NULL. No obstante, durante la configuración del servicio de integración de PowerCenter, puede elegir cómo tratar los valores nulos de las funciones de agregado. Puede determinar que el servicio de integración de PowerCenter trate los valores nulos como 0 o NULL en las funciones de agregado.

Valores nulos en condiciones de filtro

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

Valores nulos con operadores

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

```
8 * 10 - NULL
```

Para comprobar los valores nulos, use la función ISNULL.

TRUE

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

Ejemplo

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

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

CAPÍTULO 3

Operadores

Este capítulo incluye los siguientes temas:

- [Precedencia del operador, 28](#)
- [Operadores aritméticos, 29](#)
- [Operadores de cadena, 30](#)
- [Operadores de comparación, 31](#)
- [Operadores lógicos, 31](#)

Precedencia del operador

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

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

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

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

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

Operador	Significado
()	Paréntesis.
+, -, NOT	Unarios más y menos y el operador lógico NOT.
*, /, %	Multiplicación, división, módulo.
+, -	Suma, resta.
	Concatenar.

Operador	Significado
<, <=, >, >=	Menor que, menor que o igual a, mayor que, mayor que o igual a.
=, <>, !=, ^=	Igual a, distinto de, distinto de, distinto de.
AND	Operador lógico AND, utilizado cuando se especifican condiciones.
OR	Operador lógico OR, utilizado cuando se especifican condiciones.

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

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

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

Operadores aritméticos

Utilice operadores aritméticos para realizar cálculos matemáticos en datos numéricos.

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

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

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

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

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

TEMAS RELACIONADOS

- [“Descripción de operaciones aritméticas con fechas” en la página 53](#)

Operadores de cadena

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

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

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

Valores nulos

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

Ejemplo

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

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

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

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

Operadores de comparación

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

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

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

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

Cuando se utilizan operadores de comparación en una expresión, los operandos deben ser del mismo tipo de datos. Por ejemplo, la expresión `123,4 > '123'` no es válida porque se compara un número decimal con una cadena. Las expresiones `123,4 > 123` y `'a' != 'b'` son válidas porque los operandos son del mismo tipo de datos.

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

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

Operadores lógicos

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

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

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

Valores nulos

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

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

CAPÍTULO 4

Variables

Este capítulo incluye los siguientes temas:

- [Variables integradas, 33](#)
- [Variables de control de transacción, 39](#)
- [Variables locales, 39](#)

Variables integradas

El lenguaje de transformación ofrece variables integradas. Estas variables pueden devolver información en tiempo de ejecución o información del sistema. Las variables en tiempo de ejecución devuelven información como el nombre de la tabla de origen y de destino, el nombre de la carpeta, el modo de ejecución de la sesión y el nombre de la instancia que ejecuta el flujo de trabajo. Las variables del sistema devuelven la hora de inicio de la sesión, la fecha del sistema y la hora de inicio del flujo de trabajo.

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

En Designer o en el Administrador del flujo de trabajo, puede utilizar variables integradas en expresiones. Puede utilizar, por ejemplo, la variable de sistema SYSDATE en una función DATE_DIFF. Puede utilizar variables en tiempo de ejecución en expresiones y en campos de entrada que acepten variables de asignación o de flujo de trabajo. Por ejemplo, puede utilizar la variable en tiempo de ejecución \$PMWorkflowRunInstanceName como parte de un nombre de archivo de salida de destino. El Servicio de integración de datos establece los valores de las variables integradas. No puede definir valores para las variables integradas en un flujo de trabajo o en un archivo de parámetros de sesión.

Puede utilizar variables integradas en las expresiones. Puede utilizar, por ejemplo, la variable de sistema SYSDATE en una función DATE_DIFF.

Las siguientes variables integradas ofrecen información en tiempo de ejecución:

- \$PM<SourceName>@TableName, \$PM<TargetName>@TableName
- \$PMFolderName
- \$PMIntegrationServiceName
- \$PMMappingName
- \$PMRepositoryServiceName
- \$PMRepositoryUserName
- \$PMSessionName
- \$PMSessionRunMode

- \$PMWorkflowName
- \$PMWorkflowRunId
- \$PMWorkflowRunInstanceName

Las siguientes variables integradas ofrecen información del sistema:

- \$\$\$SessStartTime
- SESSSTARTTIME
- SYSDATE
- WORKFLOWSTARTTIME

La siguiente tabla describe dónde se utilizan las variables integradas en Designer y en el Administrador del flujo de trabajo.

Nombre de variable	Designer	Administrador del flujo de trabajo
\$PM<SourceName>@TableN ame, \$PM<TargetName>@TableN ame,	<ul style="list-style-type: none"> - Expresiones - Campos de entrada que aceptan variables de asignación 	<ul style="list-style-type: none"> - Campos de entrada que aceptan variables de asignación
\$PMFolderName	<ul style="list-style-type: none"> - Expresiones - Campos de entrada que aceptan variables de asignación - Campos de entrada que aceptan variables de flujo de trabajo 	<ul style="list-style-type: none"> - Expresiones - Campos de entrada que aceptan variables de asignación - Campos de entrada que aceptan variables de flujo de trabajo
\$PMIntegrationServiceName	<ul style="list-style-type: none"> - Expresiones - Campos de entrada que aceptan variables de asignación - Campos de entrada que aceptan variables de flujo de trabajo 	<ul style="list-style-type: none"> - Expresiones - Campos de entrada que aceptan variables de asignación - Campos de entrada que aceptan variables de flujo de trabajo
\$PMMappingName	<ul style="list-style-type: none"> - Expresiones - Campos de entrada que aceptan variables de asignación 	<ul style="list-style-type: none"> - Campos de entrada que aceptan variables de asignación
\$PMRepositoryServiceName	<ul style="list-style-type: none"> - Expresiones - Campos de entrada que aceptan variables de asignación - Campos de entrada que aceptan variables de flujo de trabajo 	<ul style="list-style-type: none"> - Expresiones - Campos de entrada que aceptan variables de asignación - Campos de entrada que aceptan variables de flujo de trabajo
\$PMRepositoryUserName	<ul style="list-style-type: none"> - Expresiones - Campos de entrada que aceptan variables de asignación - Campos de entrada que aceptan variables de flujo de trabajo 	<ul style="list-style-type: none"> - Expresiones - Campos de entrada que aceptan variables de asignación - Campos de entrada que aceptan variables de flujo de trabajo
\$PMSessionName	<ul style="list-style-type: none"> - Expresiones - Campos de entrada que aceptan variables de asignación 	<ul style="list-style-type: none"> - Campos de entrada que aceptan variables de asignación
\$PMSessionRunMode	<ul style="list-style-type: none"> - Expresiones - Campos de entrada que aceptan variables de asignación 	<ul style="list-style-type: none"> - Campos de entrada que aceptan variables de asignación

Nombre de variable	Designer	Administrador del flujo de trabajo
\$PMWorkflowName	<ul style="list-style-type: none"> - Expresiones - Campos de entrada que aceptan variables de asignación - Campos de entrada que aceptan variables de flujo de trabajo 	<ul style="list-style-type: none"> - Expresiones - Campos de entrada que aceptan variables de asignación - Campos de entrada que aceptan variables de flujo de trabajo
\$PMWorkflowRunId	<ul style="list-style-type: none"> - Expresiones - Campos de entrada que aceptan variables de asignación - Campos de entrada que aceptan variables de flujo de trabajo 	<ul style="list-style-type: none"> - Expresiones - Campos de entrada que aceptan variables de asignación - Campos de entrada que aceptan variables de flujo de trabajo
\$PMWorkflowRunInstanceName	<ul style="list-style-type: none"> - Expresiones - Campos de entrada que aceptan variables de asignación - Campos de entrada que aceptan variables de flujo de trabajo 	<ul style="list-style-type: none"> - Expresiones - Campos de entrada que aceptan variables de asignación - Campos de entrada que aceptan variables de flujo de trabajo
\$\$\$SessStartTime	<ul style="list-style-type: none"> - Condiciones de filtro de asignación o mapplet - Uniones definidas por el usuario - Reemplazos de SQL 	<ul style="list-style-type: none"> - Condiciones de filtro de asignación o mapplet - Uniones definidas por el usuario - Reemplazos de SQL
SESSSTARTTIME	- Expresiones	[n/a]
SYSDATE	- Expresiones	- Expresiones
WORKFLOWSTARTTIME	[n/a]	- Expresiones

\$PM<SourceName>@TableName, \$PM<TargetName>@TableName

\$PM<SourceName>@TableName y \$PM<TargetName>@TableName devuelven los nombres de las tablas de origen y de destino de instancias de origen y destino relacionales como valores de cadena. Utilice estas variables con cualquier función que acepte tipos de datos String.

El nombre de la variable depende del nombre de instancia de origen o de destino. Por ejemplo, para una instancia de origen llamada "Customers", el nombre de la variable incorporada es \$PMCustomers@TableName. Si el origen o destino relacional es parte de un mapplet dentro de una asignación, el nombre de la variable incorporada incluye el nombre del mapplet:

- \$PM<MappletName>.<SourceName>@TableName
- \$PM<MappletName>.<TargetName>@TableName

Utilice \$PM<SourceName>@TableName y \$PM<TargetName>@TableName en una asignación o en un mapplet. Por ejemplo, en una asignación que contiene varios orígenes relacionales, puede utilizar \$PM<SourceName>@TableName en el puerto de salida de una transformación de expresiones para escribir el nombre de la tabla de origen para cada fila en el destino. También puede utilizar estas variables en los campos de entrada que acepten variables de asignación.

\$PMFolderName

\$PMFolderName devuelve el nombre de la carpeta del repositorio como un valor de cadena. Puede usar \$PMFolderName con cualquier función que acepte tipos de datos String.

Puede usar \$PMFolderName en una asignación, en un mapplet, en vínculos de flujo de trabajo o en tareas de flujo de trabajo, como las tareas de asignación y decisión. Además, puede usar \$PMFolderName en campos de entrada que acepten variables de asignación o flujo de trabajo.

\$PMIntegrationServiceName

\$PMIntegrationServiceName devuelve el nombre del Servicio de integración de datos que ejecuta la sesión. Puede usar \$PMIntegrationServiceName con cualquier función que acepte tipos de datos String.

\$PMIntegrationServiceName devuelve el nombre del Servicio de integración de datos como un valor de cadena.

Puede usar \$PMIntegrationServiceName en una asignación, en un mapplet, en vínculos de flujo de trabajo o en tareas de flujo de trabajo, como las tareas de asignación y decisión. Además, puede usar \$PMIntegrationServiceName en campos de entrada que acepten variables de asignación o flujo de trabajo.

\$PMMappingName

\$PMMappingName devuelve el nombre de asignación como un valor de cadena. Puede usar \$PMMappingName con cualquier función que acepte tipos de datos String.

Puede usar \$PMMappingName en una asignación o un mapplet. Además, puede usar \$PMMappingName en campos de entrada que acepten variables de asignación.

\$PMRepositoryServiceName

\$PMRepositoryServiceName devuelve el nombre del Servicio de repositorio de modelos como un valor de cadena. Puede usar \$PMRepositoryServiceName con cualquier función que acepte tipos de datos String.

Puede usar \$PMRepositoryServiceName en una asignación, en un mapplet, en vínculos de flujo de trabajo o en tareas de flujo de trabajo, como las tareas de asignación y decisión. Además, puede usar \$PMRepositoryServiceName en campos de entrada que acepten variables de asignación o flujo de trabajo.

\$PMRepositoryUserName

\$PMRepositoryUserName devuelve el nombre del usuario del repositorio que ejecuta la sesión. Puede usar \$PMRepositoryUserName con cualquier función que acepte tipos de datos String. \$PMRepositoryUserName devuelve el nombre del usuario del repositorio como un valor de cadena.

Puede usar \$PMRepositoryUserName en una asignación, en un mapplet, en vínculos de flujo de trabajo o en tareas de flujo de trabajo, como las tareas de asignación y decisión. Además, puede usar \$PMRepositoryUserName en campos de entrada que acepten variables de asignación o flujo de trabajo.

\$PMSessionName

\$PMSessionName devuelve el nombre de sesión como un valor de cadena. Puede usar \$PMSessionName con cualquier función que acepte tipos de datos String.

Puede usar \$PMSessionName en una asignación o un mapplet. Además, puede usar \$PMSessionName en campos de entrada que acepten variables de asignación.

\$PMSessionRunMode

\$PMSessionRunMode devuelve el modo de ejecución de sesión, *normal* o *recuperación*, como un valor de cadena. Puede usar \$PMSessionRunMode con cualquier función que acepte tipos de datos String.

Puede usar \$PMSessionRunMode en una asignación o un mapplet. Además, puede usar \$PMSessionRunMode en campos de entrada que acepten variables de asignación.

\$PMWorkflowName

\$PMWorkflowName devuelve el nombre del flujo de trabajo como un valor de cadena. Puede usar \$PMWorkflowName con cualquier función que acepte tipos de datos String.

Puede usar \$PMWorkflowName en una asignación, en un mapplet, en vínculos de flujo de trabajo o en tareas de flujo de trabajo, como las tareas de asignación y decisión. Además, puede usar \$PMWorkflowName en campos de entrada que acepten variables de asignación o flujo de trabajo.

\$PMWorkflowRunId

Cada ejecución de flujo de trabajo tiene un ID de ejecución único. \$PMWorkflowRunId devuelve el ID de ejecución de flujo de trabajo como un valor de cadena. Puede usar \$PMWorkflowRunId con cualquier función que acepte tipos de datos String.

Puede usar \$PMWorkflowRunId en una asignación, en un mapplet, en vínculos de flujo de trabajo o en tareas de flujo de trabajo, como las tareas de asignación y decisión. Además, puede usar \$PMWorkflowRunId en campos de entrada que acepten variables de asignación o flujo de trabajo. Por ejemplo, puede configurar un flujo de trabajo para ejecutarlo simultáneamente con el mismo nombre de instancia y realizar un seguimiento del estado de cada ejecución del flujo de trabajo con una aplicación de otros fabricantes. Puede usar \$PMWorkflowRunId en un comando de shell posterior a la sesión para pasar el ID de ejecución a la aplicación.

\$PMWorkflowRunInstanceName

\$PMWorkflowRunInstanceName devuelve el nombre de instancia de ejecución de flujo de trabajo como un valor de cadena. Puede usar \$PMWorkflowRunInstanceName con cualquier función que acepte tipos de datos String.

Puede usar \$PMWorkflowRunInstanceName en una asignación, en un mapplet, en vínculos de flujo de trabajo o en tareas de flujo de trabajo, como las tareas de asignación y decisión. Además, puede usar \$PMWorkflowRunInstanceName en campos de entrada que acepten variables de asignación o flujo de trabajo. Por ejemplo, en el caso de un flujo de trabajo simultáneo con nombres de instancia únicos, para crear archivos de destino únicos para cada instancia de ejecución, establezca el nombre del archivo de salida de destino en las propiedades de la sesión en "OutFile_\$PMWorkflowRunInstanceName.txt".

O bien, puede usar un comando de shell posterior a la sesión para crear un archivo indicador usado por una tarea de espera de eventos predefinida. En el comando de shell que genera el archivo indicador, use \$PMWorkflowRunInstanceName en el nombre del archivo indicador para asegurarse de que una instancia de ejecución de flujo de trabajo no elimine un archivo indicador necesario para otra instancia de ejecución de flujo de trabajo.

SESSSTARTTIME

SESSSTARTTIME devuelve el valor de fecha y hora actual en el nodo que ejecuta la sesión una vez que el Servicio de integración inicializa la sesión. Puede usar SESSSTARTTIME con cualquier función que acepte

tipos de datos de fecha/hora de transformación. `SESSSTARTTIME` se almacena como un valor de tipo de datos de fecha/hora de transformación.

Puede usar `SESSSTARTTIME` en una asignación o un mapplet. Solamente puede hacer referencia a `SESSSTARTTIME` en el lenguaje de expresión.

Ejemplo

La siguiente expresión usa `$$$SessStartTime` en la condición de filtro de origen de un calificador de origen para realizar una extracción incremental. La expresión especifica un intervalo de fechas de todos los días de la semana antes de que el Servicio de integración de datos inicialice la sesión. La expresión usa la función `DATE_DIFF` para buscar la diferencia de número de días entre el valor `ORDER_DATE` y `$$$SessStartTime`. Si la diferencia entre las dos fechas es inferior o igual a siete días, el Servicio de integración de datos extrae la fila del origen:

```
DATE_DIFF(DAY, ORDER_DATE, '$$$SessStartTime') <= 7
```

SYSDATE

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

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

Para capturar una fecha estática del sistema, use la variable `SESSSTARTTIME` en lugar de `SYSDATE`.

Ejemplo

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

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

WORKFLOWSTARTTIME

`WORKFLOWSTARTTIME` devuelve la fecha y hora actuales en el nodo que hospeda el servicio de integración cuando el Servicio de integración de datos inicializa el flujo de trabajo. Puede usar `WORKFLOWSTARTTIME` con cualquier función que admita tipos de datos de fecha y hora de transformación. `WORKFLOWSTARTTIME` se almacena como un valor de tipo de datos de fecha y hora de transformación.

Puede usar `WORKFLOWSTARTTIME` en vínculos y tareas de flujo de trabajo, como las tareas de asignación y decisión. Solamente puede hacer referencia a `WORKFLOWSTARTTIME` en el lenguaje de expresión.

Ejemplo

La siguiente expresión usa `WORKFLOWSTARTTIME` para mostrar el número de minutos entre la hora de inicio del flujo de trabajo y la hora de inicio de una tarea del flujo de trabajo. Mediante la función `DATE_DIFF`

de SQL, el Servicio de integración de datos resta la hora de inicio de la tarea de WORKFLOWSTARTTIME y devuelve el resultado como un número de días:

```
DATE_DIFF(WORKFLOWSTARTTIME, $$EmployeeData.StartTime, 'MI')
```

Variables de control de transacción

Las variables de control de transacción definen las condiciones para transacciones de confirmación o rollback durante el procesamiento de filas de la base de datos. Estas variables se usan en expresiones de control de transacciones que se construyen en el Editor de expresiones. Las expresiones de control de transacciones utilizan la función IIF para comprobar cada fila con respecto a una condición. En función del valor de retorno de la condición, el Servicio de integración de datos efectúa un commit, un rollback o no hace ningún cambio de transacción para la fila.

El siguiente ejemplo utiliza variables de control de transacción para determinar dónde procesar una fila:

```
IIF (NEWTRAN=1, TC_COMMIT_BEFORE, TC_CONTINUE_TRANSACTION)
```

Si NEWTRAN=1, la variable TC_COMMIT_BEFORE hace que se produzca una confirmación antes de procesarse la fila actual. De lo contrario, la variable TC_CONTINUE_TRANSACTION obliga a que se procese la fila en la transacción actual.

Utilice las siguientes variables en el Editor de expresiones cuando se cree una expresión de control de transacciones:

- **TC_CONTINUE_TRANSACTION.** El Servicio de integración de datos no realiza ningún cambio de transacción para la fila actual. Este es el valor predeterminado de la variable de control de transacciones.
- **TC_COMMIT_BEFORE.** El Servicio de integración de datos hace una confirmación para la transacción, inicia una transacción nueva y escribe la fila actual en el destino. La fila actual está en la transacción nueva.
- **TC_COMMIT_AFTER.** El Servicio de integración de datos escribe la fila actual en el destino, hace un commit de la transacción e inicia una nueva transacción. La fila actual está en la transacción para la que se ha efectuado la confirmación.
- **TC_ROLLBACK_BEFORE.** El Servicio de integración de datos hace un rollback para la transacción, inicia una transacción nueva y escribe la fila actual en el destino. La fila actual está en la transacción nueva.

Variables locales

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

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

CAPÍTULO 5

Fechas

Este capítulo incluye los siguientes temas:

- [Introducción a las fechas, 40](#)
- [Cadenas de formato de fecha, 44](#)
- [Cadenas de formato TO_CHAR, 46](#)
- [Cadenas de formato TO_DATE e IS_DATE, 49](#)
- [Descripción de operaciones aritméticas con fechas, 53](#)

Introducción a las fechas

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

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

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

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

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

Tipo de datos de fecha y hora

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

Las funciones de fecha admiten solamente valores de fecha y hora. Para pasar una cadena a una función de fecha, primero debe usar TO_DATE para convertirla en un valor de fecha y hora. Por ejemplo, la siguiente

expresión convierte un puerto de cadenas en valores de fecha y hora y, a continuación, añade un mes a cada fecha:

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

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

Día juliano, día juliano modificado y calendario gregoriano

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

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

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

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

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

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

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

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

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

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

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

Fechas del año 2000

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

Cadena de formato RR

El idioma de transformación proporciona la cadena de formato RR para convertir cadenas con años de dos dígitos a fechas Usando TO_DATE y la cadena de formato RR, se puede convertir una cadena en el formato MM/DD/RR a una fecha. La cadena de formato RR convierte los datos de manera diferente en función del año actual.

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

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

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

Ejemplo

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

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

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

Diferencia entre las cadenas de formato YY y RR

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

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

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

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

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

Fechas de bases de datos relacionales

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

Fechas de archivos sin formato

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

Formato de fecha predeterminado

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

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

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

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

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

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

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

Cadenas de formato de fecha

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

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

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

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

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

Función	Descripción
ADD_TO_DATE	La parte de la fecha que se desea cambiar.
DATE_DIFF	La parte de la fecha que se utiliza para calcular la diferencia entre dos fechas.
GET_DATE_PART	La parte de la fecha que se desea devolver. Esta función devuelve un valor entero basado en el formato de fecha predeterminado.
IS_DATE	La fecha que se desea comprobar.
ROUND	La parte de la fecha que se desea redondear.
SET_DATE_PART	La parte de la fecha que se desea cambiar.
SYSTIMESTAMP	La precisión de la marca de hora.
TO_CHAR (Fechas)	La cadena de carácter.

Función	Descripción
TO_DATE	La cadena de carácter.
TRUNC (Fechas)	La segunda parte de la cadena que se desea trincar.

Cadenas de formato TO_CHAR

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

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

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

Cadena de formato	Descripción
AM, A.M., PM, P.M.	Indicador de meridiano. Utilice una de estas cadenas de formato para especificar horas AM y PM. AM y PM devuelven los mismos valores que A.M. y P.M.
D	Día de la semana (1-7), donde el domingo es 1.
DAY	Nombre del día con una longitud de hasta nueve caracteres (por ejemplo, miércoles).
DD	Día del mes (01-31).
DDD	Día del año (001-366, incluidos años bisiestos).
DY	Nombre de un día abreviado en tres caracteres (por ejemplo, Mié).
HH, HH12	Hora del día (01-12).
HH24	Hora de día (00-23), donde 00 es 12 AM (medianoche).
J	Fecha juliana modificada. Convierte la fecha del calendario en una cadena equivalente a su valor en la fecha juliana modificada, que se calcula a partir del 1 de enero 4713 00:00:00 a. C. Omite el componente de hora de la fecha. Por ejemplo, la expresión TO_CHAR(SHIP_DATE, 'J') convierte la fecha 31 de diciembre de 1999, 23:59:59 en la cadena 2451544.
MI	Minutos (00-59).
MM	Mes (01-12).
MONTH	Nombre del mes con una longitud máxima de hasta nueve caracteres (por ejemplo, enero).
MON	Nombre del mes abreviado en tres caracteres (por ejemplo, Ene).
MS	Milisegundos (0-999).

Cadena de formato	Descripción
NS	Nanosegundos (0-999999999).
Q	Trimestre el año (1-4), donde de enero a marzo es igual a 1.
RR	Los últimos dos dígitos de un año. La función quita los dígitos del principio. Por ejemplo, si utiliza 'RR' y transfiere el año 1997, TO_CHAR devuelve 97. Cuando se utiliza con TO_CHAR, 'RR' genera los mismos resultados, ya que se puede intercambiar con 'YY.' Sin embargo, cuando se utiliza con TO_DATE, 'RR' calcula el siglo más próximo que corresponda y proporciona los dos primeros dígitos del año.
SS	Segundos (00-59).
SSSSS	Segundos hasta medianoche (00000 - 86399). Cuando utiliza SSSSS en una expresión TO_CHAR, el Servicio de integración de datos solo verifica la parte de hora de una fecha. Por ejemplo, la expresión TO_CHAR(SHIP_DATE, 'MM/DD/YYYY SSSSS') convierte 12/31/1999 01:02:03 en 12/31/1999 03723.
US	Microsegundos (0-999999).
Y	El último dígito de un año. La función quita los dígitos del principio. Por ejemplo, si utiliza 'Y' y transfiere el año 1997, TO_CHAR devuelve 7.
YY	Los últimos dos dígitos de un año. La función quita los dígitos del principio. Por ejemplo, si utiliza 'YY' y transfiere el año 1997, TO_CHAR devuelve 97.
YYY	Los últimos tres dígitos de un año. La función quita los dígitos del principio. Por ejemplo, si utiliza 'YYY' y transfiere el año 1997, TO_CHAR devuelve 997.
YYYY	Toda la parte de año de una fecha. Por ejemplo, si utiliza 'YYY' y transfiere el año 1997, TO_CHAR devuelve 1997.
W	Semana del mes (1-5) , donde la semana 1 empieza en el primer día del mes y finaliza en el séptimo, y la semana 2 se inicia en el octavo día y finaliza en el catorceavo. Por ejemplo, 1 febrero designa la primera semana de febrero.
WW	Semana del año (01-53), donde la semana 01 se inicia el 1 de enero y finaliza el 7, y la semana 2 se inicia el 8 y finaliza el 14 de enero, y así sucesivamente.
- / . ; :	La puntuación que se muestra en la salida. Puede utilizar estos símbolos para separar las partes de una fecha. Crea, por ejemplo, la siguiente expresión para separar las partes de una fecha con un punto: TO_CHAR(DATES, 'MM.DD.YYYY').
"text"	El texto que se muestra en la salida. Si crea, por ejemplo, un puerto de salida con la expresión: TO_CHAR(DATES, 'MM/DD/YYYY "Ventas en aumento"') y transfiere la fecha 1 de abril de 1997, la función devuelve la cadena '04/01/1997 Ventas en aumento'. Puede indicar los caracteres multibyte que sean válidos en la página de códigos del repositorio.
""	Utilice las comillas dobles para separar cadenas de formato ambiguas, por ejemplo, D""DDD. Las comillas vacías no aparecen en la salida.

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

Ejemplos

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

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

Cadena de formato J

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

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

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

Cadena de formato SSSSS

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

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

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

Cadena de formato RR

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

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

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

Cadena de formato YY

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

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

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

Cadenas de formato TO_DATE e IS_DATE

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

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

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

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

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

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

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

Tabla 1. Cadenas de formato TO_DATE e IS_DATE

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

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

Normas y directrices para cadenas con formato de fecha

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

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

Ejemplo

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

Cadena de formato J

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

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

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

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

Cadena de formato RR

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

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

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

Cadena de formato YY

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

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

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

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

Cadena de formato SSSS

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

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

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

Descripción de operaciones aritméticas con fechas

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

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

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

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

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

CAPÍTULO 6

Funciones

En este capítulo, se describen las funciones del lenguaje de transformación por orden alfabético. La descripción de cada función incluye lo siguiente:

- Sintaxis
- Valor devuelto
- Ejemplo

Categorías de funciones

El lenguaje de transformación incluye los siguientes tipos de funciones:

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

Funciones agregadas

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

- Calcular un valor individual para todas las filas de un grupo.
- Devolver un valor individual para cada grupo en una transformación de agregación.
- Aplicar filtros para calcular valores para filas específicas en los puertos seleccionados.

- Utilizar operadores para realizar operaciones aritméticas en la función.
- Calcular dos o más valores agregados derivados de las mismas columnas origen en un pase individual.

El idioma de la transformación incluye las siguientes funciones agregadas:

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

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

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

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

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

cuando el conjunto de datos contiene los siguientes valores:

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

ID	EARNINGS
2	76
3	21
3	45
3	99

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

Funciones de agregado y valores nulos

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

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

Condiciones de filtro

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

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

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

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

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

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

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

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

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

Funciones de caracteres

El lenguaje de transformación incluye las siguientes funciones de caracteres:

- ASCII
- CHR

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

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

Funciones de conversión

El lenguaje de transformación incluye las siguientes funciones de conversión:

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

Funciones de limpieza de datos

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

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

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

- GREATEST

- IN
- INSTR
- IS_DATE
- IS_NUMBER
- IS_SPACES
- ISNULL
- LEAST
- LTRIM
- METAPHONE
- REG_EXTRACT
- REG_MATCH
- REG_REPLACE
- REPLACECHR
- REPLACESTR
- RTRIM
- SQL_LIKE
- SOUNDEX
- SUBSTR
- TO_BIGINT
- TO_CHAR
- TO_DATE
- TO_DECIMAL
- TO_FLOAT
- TO_INTEGER

Funciones de fecha

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

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

El lenguaje de transformación incluye las siguientes funciones de fecha:

- `ADD_TO_DATE`
- `DATE_COMPARE`
- `DATE_DIFF`
- `GET_DATE_PART`
- `IS_DATE`
- `LAST_DAY`
- `MAKE_DATE_TIME`
- `MAX`

- MIN
- ROUND (fecha)
- SET_DATE_PART
- SYSTIMESTAMP
- TO_CHAR (fecha)
- TRUNC (fecha)

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

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

TEMAS RELACIONADOS

- [“Cadenas de formato de fecha” en la página 44](#)

Funciones de codificación

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

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

Funciones financieras

El lenguaje de transformación incluye las siguientes funciones financieras:

- FV
- NPER
- PMT
- PV
- RATE

Funciones numéricas

El lenguaje de transformación incluye las siguientes funciones numéricas:

- ABS
- CEIL
- CONV

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

Funciones científicas

El lenguaje de transformación incluye las siguientes funciones científicas:

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

Funciones especiales

El lenguaje de transformación incluye las siguientes funciones especiales:

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

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

Funciones de cadena

El lenguaje de transformación incluye las siguientes funciones de cadena:

- CHOOSE
- INDEXOF
- MAX
- MIN
- REVERSE

Funciones de prueba

El lenguaje de transformación incluye las siguientes funciones de prueba:

- ISNULL
- IS_DATE
- IS_NUMBER
- IS_SPACES

Funciones de variables

El lenguaje de transformación incluye un grupo de funciones de variables para actualizar el valor actual de una variable de asignación durante el transcurso de la sesión. Cuando se ejecuta un flujo de trabajo, el Servicio de Integración PowerCenter evalúa el valor inicial y actual de una variable al inicio de la sesión según el valor final de la variable de la última sesión ejecutada. Utilice las siguientes funciones de variables:

- SetCountVariable
- SetMaxVariable
- SetMinVariable
- SetVariable

Utilice distintas funciones de variable con una variable basada en el tipo de agregación de la variable.

Cuando se usan variables de asignación en sesiones con múltiples particiones, utilice funciones de variable para determinar el valor final de la variable para cada partición. Al final de la sesión, el Servicio de Integración PowerCenter lleva a cabo la función de agregación en todas las particiones para determinar un valor final que guardar en el repositorio. Salvo que se haya anulado, utiliza el valor guardado como el valor inicial de la variable para la siguiente vez que se utilice esta sesión.

Por ejemplo, utilice SetMinVariable para establecer una variable con el valor mínimo evaluado. El Servicio de Integración PowerCenter calcula el valor mínimo actual para la variable de cada partición. Luego, al final de la sesión, busca el valor mínimo actual en todas las particiones y guarda dicho valor en el repositorio.

Utilice SetVariable solamente una vez para cada variable de asignación de una conducción. Cuando se crean múltiples particiones en una conducción, el Servicio de Integración PowerCenter utiliza múltiples procesos para procesar dicha conducción. Si se utiliza esta función más de una vez para la misma variable, el valor actual de una variable de asignación puede dar resultados indeterminados.

ABORT

Detiene la sesión y emite un mensaje de error especificado al archivo de registro de la sesión. Cuando el Servicio de integración de datos detecta una función ABORT, deja de transformar datos en esa fila. Procesa todas las filas leídas antes de que se cancele la sesión y las carga según el intervalo de confirmación de origen o destino y el tamaño de bloque de búfer definido para la sesión. El Servicio de integración de datos graba en el destino hasta la fila cancelada y luego acumula todos los datos no confirmados hasta el último punto de confirmación. Se puede realizar una recuperación de la sesión después de la acumulación.

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

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

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

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

Nota: El Servicio de integración de datos maneja de formas distintas la función ABORT y el comando Abort que se lanza desde el Workflow Manager.

sintaxis

```
ABORT( string )
```

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

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

Valor de retorno

NULL.

ABS

Devuelve el valor absoluto de un valor numérico.

Sintaxis

```
ABS( numeric_value )
```

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

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

Valor devuelto

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

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

Nota: Si el valor devuelto es un decimal con una precisión superior a 15, puede habilitar el modo de alta precisión para asegurarse de obtener una precisión decimal de hasta 28 dígitos.

Ejemplo

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

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

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

ADD_TO_DATE

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Sintaxis

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


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

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

Valor de devolución

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

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

Ejemplos

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

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

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

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

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

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

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

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

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

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

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

Cómo trabajar con fechas

Utilice los siguientes consejos cuando trabaje con ADD_TO_DATE:

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

AES_DECRYPT

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

Sintaxis

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

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

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

Valor devuelto

Valor binario descifrado.

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

Ejemplo

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

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

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

AES_ENCRYPT

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

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

Sintaxis

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

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

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

Valor de retorno

Valor binario cifrado.

NULL si la entrada es un valor nulo.

Ejemplo

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

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

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

Consejo

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

ANY

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

Sintaxis

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

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

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

Valor de retorno

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

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

Ejemplo

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

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

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

ASCII

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

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

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

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

Sintaxis

```
ASCII ( string )
```

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

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

Valor devuelto

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

NULL si el valor pasado a la función es NULL.

Ejemplo

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

```
ASCII( ITEMS )
```

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

AVG

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

Sintaxis

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

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

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

Valor de retorno

Valor numérico.

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

Nota: Si el valor devuelto es un decimal con una precisión superior a 15, puede habilitar el modo de alta precisión para asegurarse de obtener una precisión decimal de hasta 28 dígitos.

Nulos

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

Nota: De forma predeterminada, el Servicio de integración de datos trata los valores nulos como NULL en las funciones agregadas. Si pasa un puerto o grupo de valores nulos completo, la función devuelve NULL. Sin embargo, cuando se configura el Servicio de integración de datos, se puede indicar la forma en que se desea manejar los valores nulos en las funciones agregadas. En las funciones agregadas se pueden tratar los valores nulos como 0 o como NULL.

Agrupar por

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

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

Ejemplo

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

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

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

ITEM_NAME	WHOLESALE_COST
Flashlight	31.00

RETURN VALUE: 31.66

Consejo

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

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

CEIL

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

Sintaxis

```
CEIL( numeric_value )
```

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

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

Valor de devolución

Valor numérico.

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

Valor doble si se pasa un valor numérico con precisión declarada mayor que 28.

NULL si el valor pasado a la función es NULL.

Ejemplo

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

```
CEIL( PRICE )
```

PRICE	RETURN VALUE
39.79	40
125.12	126
74.24	75

PRICE	RETURN VALUE
NULL	NULL
-100.99	-100

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

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

CHOOSE

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

Sintaxis

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

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

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

Valor devuelto

Cadena coincidente con la posición del valor de índice.

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

Ejemplo

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

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

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

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

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

CHR

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

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

Sintaxis

```
CHR( numeric_value )
```

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

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

Valor devuelto

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

NULL si el valor pasado a la función es NULL.

Ejemplo

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

```
CHR( ITEM_ID )
```

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

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

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

El valor devuelto es:

```
Joan's car
```

CHRCODE

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

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

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

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

Sintaxis

```
CHRCODE ( string )
```

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

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

Valor devuelto

Entero.

NULL si el valor pasado a la función es NULL.

Ejemplo

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

```
CHRCODE( ITEMS )
```

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

COMPRESS

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

Sintaxis

```
COMPRESS( value )
```

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

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

Valor devuelto

Valor binario comprimido del valor de entrada.

NULL si la entrada es un valor nulo.

Ejemplo

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

CONCAT

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

Sintaxis

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

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

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

Valor de retorno

Cadena.

NULL si ambos valores de cadena son NULL.

Nulos

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

Si ambas cadenas son NULL, CONCAT devuelve NULL.

Ejemplo

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

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

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

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

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

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

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

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

El valor de retorno es:

```
Joan's car
```

CONVERT_BASE

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

Sintaxis

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

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

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

Valor devuelto

Valor numérico.

Ejemplo

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

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

El Servicio de integración de datos devuelve 100010101110.

COS

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

Sintaxis

```
COS( numeric_value )
```

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

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

Valor devuelto

Valor doble.

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

Ejemplo

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

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

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

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

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

COSH

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

Sintaxis

```
COSH( numeric_value )
```

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

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

Valor devuelto

Valor doble.

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

Ejemplo

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

```
COSH ( ANGLES )
```

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

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

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

COUNT

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

Sintaxis

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


0

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

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

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

Valor de retorno

Entero.

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

Nulos

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

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

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

Nota: De forma predeterminada, el Servicio de integración de datos trata los valores nulos como NULL en las funciones agregadas. Si pasa un puerto o grupo de valores nulos completo, la función devuelve NULL. Sin embargo, cuando se configura el Servicio de integración de datos, se puede indicar la forma en que se desea manejar los valores nulos en las funciones agregadas. En las funciones agregadas se pueden tratar los valores nulos como 0 o como NULL.

Agrupar por

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

Ejemplos

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

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

ITEM_NAME	IN_STOCK
Flashlight	10
NULL	2
Compass	NULL
Regulator System	5

ITEM_NAME	IN_STOCK
Safety Knife	8
Halogen Flashlight	1
RETURN VALUE: 1	

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

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

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

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

```
COUNT( * )
```

ITEM_NAME	QTY
Flashlight	10
NULL	2
Compass	NULL
Regulator System	5
Safety Knife	8
Halogen Flashlight	1
RETURN VALUE: 6	

CRC32

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

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

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

Sintaxis

```
CRC32( value )
```

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

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

Valor de retorno

Valor entero de 32 bits.

Ejemplo

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

CREATE_TIMESTAMP_TZ

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

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

Sintaxis

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

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

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

Valor devuelto

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

NULL si la entrada es un valor nulo.

Ejemplo

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

CUME

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

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

Sintaxis

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

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

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

Valor de retorno

Valor numérico.

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

Nota: Si el valor devuelto es un decimal con una precisión superior a 15, puede habilitar el modo de alta precisión para asegurarse de obtener una precisión decimal de hasta 28 dígitos.

Nulos

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

Ejemplos

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

```
CUME ( PERSONAL_SALES )
```

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

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

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

CA_SALES	OR_SALES	RETURN VALUE
40000	10000	50000
80000	50000	180000

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

DATE_COMPARE

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

Sintaxis

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

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

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

Valor devuelto

Se devuelve -1 si la primera fecha es anterior.

Se devuelve 0 si las dos fechas coinciden.

Se devuelve 1 si la segunda fecha es anterior.

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

Ejemplo

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

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

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

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

DATE_DIFF

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

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

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

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

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

Sintaxis

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

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

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

Valor de retorno

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

0 si las fechas son las mismas.

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

Ejemplos

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

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

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

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

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



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

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

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

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

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

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

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

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

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

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

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

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

DEC_BASE64

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

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

Sintaxis

```
DEC_BASE64( value )
```

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

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

Valor devuelto

Valor binario decodificado.

NULL si la entrada es un valor nulo.

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

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

Ejemplo

Usted ha codificado los identificadores de mensaje de WebSphere MQ y los ha escrito en un archivo sin formato durante un flujo de trabajo. Quiere leer los datos del origen del archivo sin formato, incluidos los identificadores de mensaje de WebSphere MQ. Puede utilizar DEC_BASE64 para decodificar los identificadores y convertirlos en sus valores binarios originales.

DECODE

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

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

Sintaxis

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

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

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

Valor de devolución

First_result si la búsqueda encuentra un valor coincidente.

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

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

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

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

DECODE y los tipos de datos

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

Supongamos que tiene la siguiente expresión:

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

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

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

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

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

La expresión siguiente, por ejemplo, no es válida:

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

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

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

Ejemplos

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

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

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

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

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

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

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

DECOMPRESS

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

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

Sintaxis

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

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

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

Valor de retorno

Valor binario descomprimido del valor de entrada.

NULL si la entrada es un valor nulo.

Ejemplo

Su organización cuenta con un servicio de pedidos en línea. Ha recibido datos de pedidos de un cliente comprimidos a través de una red de área amplia. Desea leer los datos utilizando PowerCenter y cargarlos a un almacén de datos. Puede descomprimir cada fila de datos utilizando DECOMPRESS para la fila. El Servicio de integración de datos podrá cargar entonces los datos descomprimidos en el destino.

ENC_BASE64

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

Sintaxis

```
ENC_BASE64( value )
```

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

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

Valor de retorno

Valor codificado.

NULL si la entrada es un valor nulo.

Ejemplo

Desea leer mensajes de WebSphere MQ y escribir los datos en un destino del archivo sin formato. Quiere incluir el identificador de mensaje de WebSphere MQ como parte de los datos de destino. Sin embargo, el campo MsgID es binario y el destino del archivo sin formato no es compatible con datos binarios. Utilice ENC_BASE64 para codificar el MsgID antes de que el Servicio de integración de datos escriba los datos en el destino.

ERROR

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

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

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

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

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

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

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

Sintaxis

```
ERROR( string )
```

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

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

Valor de devolución

Cadena.

Ejemplo:

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

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

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

SALARY	RETURN VALUE
150000	150000
1005	1005

EXP

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

Sintaxis

`EXP (exponent)`

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

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

Valor devuelto

Valor doble.

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

Ejemplo

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

`EXP (NUMBERS)`

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

FIRST

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

Sintaxis

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

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

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

Valor de retorno

Primer valor de un grupo.

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

Nulos

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

Nota: De forma predeterminada, el Servicio de integración de datos trata los valores nulos como NULL en las funciones agregadas. Si pasa un puerto o grupo de valores nulos completo, la función devuelve NULL. Sin embargo, cuando se configura el Servicio de integración de datos, se puede indicar la forma en que se desea manejar los valores nulos en las funciones agregadas. En las funciones agregadas se pueden tratar los valores nulos como 0 o como NULL.

Agrupar por

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

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

Ejemplos

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

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

ITEM_NAME	ITEM_PRICE
Flashlight	35.00

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

RETURN VALUE: Flashlight

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

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

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

RETURN VALUE: Regulator System

FLOOR

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

Sintaxis

```
FLOOR( numeric_value )
```

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

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

Valor de retorno

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

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

NULL si el valor pasado a la función es NULL.

Ejemplo

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

```
FLOOR( PRICE )
```

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

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

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

FV

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

Sintaxis

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

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

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

Valor de retorno

Númerico.

Ejemplo

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

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

Notas

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

GET_DATE_PART

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

Sintaxis

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

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

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

Valor de retorno

Entero que representa la parte especificada de la fecha.

NULL si el valor pasado a la función es NULL.

Ejemplos

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

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

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

Las siguientes expresiones devuelven el día para cada fecha en el puerto DATE_SHIPPED.

```
GET_DATE_PART( DATE_SHIPPED, 'D' )
GET_DATE_PART( DATE_SHIPPED, 'DD' )
GET_DATE_PART( DATE_SHIPPED, 'DDD' )
```

```
GET_DATE_PART( DATE_SHIPPED, 'DY' )
GET_DATE_PART( DATE_SHIPPED, 'DAY' )
```

DATE_SHIPPED	RETURN VALUE
Mar 13 1997 12:00:00AM	13
June 3 1997 11:30:44PM	3
Aug 22 1997 12:00:00PM	22
NULL	NULL

Las siguientes expresiones devuelven el mes para cada fecha en el puerto DATE_SHIPPED.

```
GET_DATE_PART( DATE_SHIPPED, 'MM' )
GET_DATE_PART( DATE_SHIPPED, 'MON' )
GET_DATE_PART( DATE_SHIPPED, 'MONTH' )
```

DATE_SHIPPED	RETURN VALUE
Mar 13 1997 12:00:00AM	3
June 3 1997 11:30:44PM	6
NULL	NULL

Las siguientes expresiones devuelven el año para cada fecha en el puerto DATE_SHIPPED.

```
GET_DATE_PART( DATE_SHIPPED, 'Y' )
GET_DATE_PART( DATE_SHIPPED, 'YY' )
GET_DATE_PART( DATE_SHIPPED, 'YYY' )
GET_DATE_PART( DATE_SHIPPED, 'YYYY' )
```

DATE_SHIPPED	RETURN VALUE
Mar 13 1997 12:00:00AM	1997
June 3 1997 11:30:44PM	1997
NULL	NULL

GET_TIMEZONE

Devuelve el valor de zona horaria de una marca de tiempo dada con la columna de zona horaria.

Por ejemplo:

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

El puerto de salida debe ser de tipo de datos de cadena para las expresiones GET_TIMEZONE.

Sintaxis

```
GET_TIMEZONE (timestamp_with_timezone_value)
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>timestamp_with_timezone_value</i>	Obligatorio	Debe ser una marca de tiempo con tipo de datos de zona horaria. Se puede especificar cualquier expresión de transformación válida.

Valor devuelto

Devuelve un tipo de datos de cadena que contiene el nombre de la región de zona horaria o el ajuste de la zona horaria.

NULL si la entrada es un valor nulo.

Ejemplo

INPUT VALUE	RETURN VALUE
'1947-08-05 10:45:00.221111111 AM America/Los_Angeles'	'AMERICA/LOS_ANGELES'
'1947-08-05 10:45:00.221111111 AM -08:00'	'-08:00'

GET_TIMESTAMP

Devuelve el valor de fecha y hora para un tipo de entrada `timestampWithTZ`. La fecha y hora devueltas estarán en la zona horaria solicitada, que se puede proporcionar como segundo argumento. Si el valor de la zona horaria no se especifica en el segundo argumento, la función devuelve la parte de la marca de tiempo del valor `timestampWithTZ` de entrada.

Por ejemplo:

```
GET_TIMESTAMP(Timestamp with Time Zone, "+08:30")
```

El primer argumento, marca de tiempo con zona horaria tiene (+05:30) como el valor de la zona horaria. La función devuelve la marca de tiempo en la zona horaria especificada como segundo argumento (+08:30).

El puerto de salida debe ser Date/Time para las expresiones `GET_TIMESTAMP`.

Sintaxis

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

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

Argumento	Obligatorio / Opcional	Descripción
<i>timestamp_with_timezone_value</i>	Obligatorio	Debe ser una marca de tiempo con tipo de datos de zona horaria. Se puede especificar cualquier expresión de transformación válida.
<i>timezone_value</i>	Opcional	Debe ser un tipo de datos de cadena. La cadena debe ser una cadena de caracteres. Pasa los valores que se desean mostrar para la zona horaria y en base a los cuales la función puede devolver la marca de tiempo. Se puede especificar cualquier expresión de transformación válida. Si no se especifica la zona horaria, la función devuelve la parte de la marca de tiempo del primer argumento.

Valor devuelto

Devuelve el valor de marca de tiempo en la región o el ajuste de zona horaria especificado.

Si no se pasa el valor de zona horaria, la función devuelve la parte de la marca de tiempo del primer argumento.

NULL si la entrada es un valor nulo.

Ejemplo

INPUT VALUE	RETURN VALUE
'1996-01-05 10:45:00.221111111 AM America/Los_Angeles', '+05:30'	Returns the timestamp value in time zone offset of '+05:30': '1996-01-06 12:15:00.221111111 AM'
'1996-01-05 10:45:00.221111111 AM America/Los_Angeles', 'GMT'	Returns the timestamp value in the 'GMT' time zone: '1996-01-05 06:45:00.221111111 PM'
'1996-01-05 10:45:00.221111111 AM America/Los_Angeles'	As the time zone value is not passed as the second input parameter, the timestamp is returned: '1996-01-05 10:45:00.221111111 AM'

GREATEST

Devuelve el valor mayor de una lista de valores de entrada. Utilice esta función para devolver la cadena, fecha o número mayor. De forma predeterminada, la coincidencia distingue mayúsculas de minúsculas.

Sintaxis

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


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

Argumento	Obligatorio / Opcional	Descripción
<i>valor</i>	Obligatorio	Cualquier tipo de datos excepto Binario. El tipo de datos debe ser compatible con otros valores. Valor que desea comparar con otros valores. Debe introducir por lo menos un argumento de valor. Si el valor es numérico y otros valores de entrada son numéricos, todos los valores usan la mayor precisión posible. Por ejemplo, si el tipo de datos de algunos valores es Entero y el de otros es Doble, el Servicio de integración de datos convierte los valores a Doble.
<i>CaseFlag</i>	Opcional	Debe ser un entero. Especifique un valor cuando el argumento de valor de entrada sea un valor de cadena. Determina si en los argumentos de esta función se distingue entre mayúsculas y minúsculas. Se puede especificar cualquier expresión de transformación válida. Cuando CaseFlag es un número distinto de 0, la función distingue entre mayúsculas y minúsculas. Cuando CaseFlag es 0, la función no distingue entre mayúsculas y minúsculas. El valor predeterminado distingue mayúsculas de minúsculas.

Valor devuelto

value1 si es el mayor de los valores de entrada, *value2* si es el mayor de los valores de entrada, y así sucesivamente.

NULL si alguno de los argumentos es nulo.

Ejemplo

La expresión siguiente devuelve la cantidad mayor de artículos pedidos:

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

QUANTITY1	QUANTITY2	QUANTITY3	RETURN VALUE
150	756	27	756
			NULL
5000	97	17	5000
120	1724	965	1724

IIF

Devuelve uno de los dos valores que especifique, según el resultado de una condición.

Sintaxis

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

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

Argumento	Obligatorio / Opcional	Descripción
<i>condition</i>	Obligatorio	La condición que desea verificar. Puede especificar cualquier expresión de transformación válida que se verifique en TRUE o FALSE.
<i>value1</i>	Obligatorio	Cualquier tipo de datos a excepción de Binary. El valor que desea devolver si la condición es TRUE. El valor de devolución siempre es el tipo de datos que se ha especificado con este argumento. Puede especificar cualquier expresión de transformación válida, incluida otra expresión IIF.
<i>value2</i>	Opcional	Cualquier tipo de datos a excepción de Binary. El valor que desea devolver si la condición es FALSE. Puede especificar cualquier expresión de transformación válida, incluida otra expresión IIF.

A diferencia de lo que ocurre con las funciones condicionales de algunos sistemas, la condición FALSE (*value2*) no es necesaria en la función IIF. Si omite *value2*, la función devuelve lo siguiente cuando la condición es FALSE:

- 0 si *value1* es un tipo de datos numérico.
- Cadena vacía si *value1* es un tipo de datos String.
- NULL si *value1* es un tipo de datos de fecha y hora.

Por ejemplo, la siguiente expresión no incluye una condición FALSE y *value1* es un tipo de datos String, así que el Servicio de integración de datos devuelve una cadena vacía para cada fila que se verifica en FALSE:

```
IIF( SALES > 100, EMP_NAME )
```

SALES	EMP_NAME	RETURN VALUE
150	John Smith	John Smith
50	Pierre Bleu	' ' (empty string)
120	Sally Green	Sally Green
NULL	Greg Jones	' ' (empty string)

Valor de devolución

value1 si la condición es TRUE.

value2 si la condición es FALSE.

La siguiente expresión incluye, por ejemplo, la condición FALSE NULL, así que el Servicio de integración de datos devuelve NULL para cada fila que se verifica en FALSE:

```
IIF( SALES > 100, EMP_NAME, NULL )
```

SALES	EMP_NAME	RETURN VALUE
150	John Smith	John Smith
50	Pierre Bleu	NULL

SALES	EMP_NAME	RETURN VALUE
120	Sally Green	Sally Green
NULL	Greg Jones	NULL

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

IIF y los tipos de datos

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

Supongamos que tiene la siguiente expresión:

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

El resultado TRUE (1) es un entero y el resultado FALSE (0,3333), un decimal. El tipo de datos decimal es más preciso que Integer, así que el tipo de datos del valor de devolución siempre es un decimal.

Cuando ejecuta una sesión con un mayor grado de precisión y como mínimo un resultado es Double, el tipo de datos del valor de devolución es Double.

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

Usos especiales de IIF

Utilice instrucciones IIF anidadas para probar varias condiciones. El siguiente ejemplo prueba varias condiciones y devuelve 0 si las ventas son 0 o negativas:

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

Si añade algunos comentarios, la lógica de esta instrucción será más comprensible:

```
IIF( SALES > 0,
--then test to see if sales is between 1 and 49:
  IIF( SALES < 50,

--then return SALARY1
    SALARY1,

--else test to see if sales is between 50 and 99:
    IIF( SALES < 100,

--then return
      SALARY2,

--else test to see if sales is between 100 and 199:
      IIF( SALES < 200,

--then return
        SALARY3,

--else for sales over 199, return
        BONUS)
    )
  ),
--else for sales less than or equal to zero, return
  0)
```

Utilice IIF en las estrategias de actualización. Por ejemplo:

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

Alternativa a IIF

Utilice [“DECODE” en la página 91](#) en lugar de IIF en muchos casos. DECODE puede mejorar la legibilidad. El siguiente esquema muestra cómo puede utilizar DECODE en lugar de IIF con el primer ejemplo de la sección anterior:

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

Puede utilizar una transformación de filtro en lugar de IIF para maximizar el rendimiento de la sesión.

Puede utilizar una transformación de filtro en lugar de IIF para maximizar el rendimiento.

IN

Hace coincidir los datos de entrada con una lista de valores. De forma predeterminada, la coincidencia distingue mayúsculas de minúsculas.

Sintaxis

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

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

Argumento	Obligatorio / Opcional	Descripción
<i>valueToSearch</i>	Obligatorio	Puede ser un valor de cadena, fecha o numérico. Valor de entrada del que desea buscar una coincidencia dentro de una lista de valores separados por comas.
<i>valor</i>	Obligatorio	Puede ser un valor de cadena, fecha o numérico según el tipo especificado para el argumento valueToSearch. Lista de valores separados por comas que desea buscar. Los valores pueden ser los puertos de una transformación. No hay un número máximo de valores que se pueden enumerar.
<i>CaseFlag</i>	Opcional	Debe ser un entero. Debe ser un entero o NULL. Especifique un valor cuando el argumento valueToSearch sea un valor de cadena. Determina si en los argumentos de esta función se distingue entre mayúsculas y minúsculas. Se puede especificar cualquier expresión de transformación válida. Cuando CaseFlag es un número distinto de 0, la función distingue entre mayúsculas y minúsculas. Cuando CaseFlag es 0, la función no distingue entre mayúsculas y minúsculas. Cuando CaseFlag es 0, la función no distingue entre mayúsculas y minúsculas. Cuando CaseFlag es un valor nulo, la función devuelve NULL si no coincide con los argumentos en la función. De lo contrario, CaseFlag devuelve 1 si coincide con el argumento de la función. El valor predeterminado distingue mayúsculas de minúsculas.

Valor devuelto

TRUE (1) si el valor de la entrada coincide con la lista de valores.

FALSE (0) si el valor de la entrada no coincide con la lista de valores.

NULL si la entrada es un valor nulo.

Ejemplo

La siguiente expresión determina si el valor de entrada es safety knife, chisel point knife o medium titanium knife. En los valores de entrada, las mayúsculas y las minúsculas no tienen que coincidir con las de los valores de la lista separada por comas:

```
IN( ITEM_NAME, 'Chisel Point Knife', 'Medium Titanium Knife', 'Safety Knife', 0 )
```

ITEM_NAME	RETURN VALUE
Stabilizing Vest	0 (FALSE)
Safety knife	1 (TRUE)
Medium Titanium knife	1 (TRUE)
	NULL

INDEXOF

Busca el índice de un valor de entre una lista de valores. De forma predeterminada, la coincidencia distingue mayúsculas de minúsculas.

Sintaxis

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

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

Argumento	Obligatorio / Opcional	Descripción
<i>valueToSearch</i>	Obligatorio	Tipo de datos String. Valor que desea buscar en la lista de cadenas.
<i>string</i>	Obligatorio	Tipo de datos String. Lista de valores separados por comas en la que desea buscar. Los valores pueden estar en formato de cadena. No hay un número máximo de valores que se pueden enumerar. El valor distingue mayúsculas de minúsculas a menos que establezca CaseFlag en 0.
<i>CaseFlag</i>	Opcional	Debe ser un entero. Especifique un valor cuando el argumento valueToSearch sea un valor de cadena. Determina si en los argumentos de esta función se distingue entre mayúsculas y minúsculas. Se puede especificar cualquier expresión de transformación válida. Cuando CaseFlag es un número distinto de 0, la función distingue entre mayúsculas y minúsculas. Cuando CaseFlag es 0, la función no distingue entre mayúsculas y minúsculas.

Valor devuelto

1 si el valor de entrada coincide con *string1*, 2 si el valor de entrada coincide con *string2*, y así sucesivamente.

0 si el valor de entrada no se encuentra.

NULL si la entrada es un valor nulo.

Ejemplo

La siguiente expresión determina si los valores del puerto ITEM_NAME coinciden con la primera, con la segunda o con la tercera cadena:

```
INDEXOF( ITEM_NAME, 'diving hood', 'flashlight', 'safety knife')
```

ITEM_NAME	RETURN VALUE
Safety Knife	0
diving hood	1
Compass	0
safety knife	3
flashlight	2

Safety Knife devuelve un valor de 0, ya que las mayúsculas y minúsculas no coinciden con las del valor de entrada.

INITCAP

Se usan mayúsculas para la primera letra de cada palabra de una cadena y se usan minúsculas para el resto de las letras. Las palabras se delimitan con espacios en blanco (espacio en blanco, avance de página, nueva línea, retorno de carro, tabulación o tabulación vertical) y caracteres no alfanuméricos. Por ejemplo, si pasa la cadena '...THOMAS', la función devuelve Thomas.

Sintaxis

```
INITCAP( string )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>string</i>	Obligatorio	Cualquier tipo de datos a excepción de los binarios. Se puede especificar cualquier expresión de transformación válida.

Valor de devolución

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

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

Ejemplo

La siguiente expresión usa mayúsculas para todos los nombres del puerto FIRST_NAME:

```
INITCAP( FIRST_NAME )
```

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

INSTR

Devuelve la posición de un conjunto de caracteres en una cadena, contando de izquierda a derecha.

Sintaxis

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

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

Argumento	Obligatorio/ opcional	Descripción
<i>string</i>	Obligatorio	La cadena debe ser una cadena de caracteres. Transfiere el valor que desea evaluar. Puede especificar cualquier expresión de transformación válida. Los resultados de la expresión deben ser una cadena de caracteres. De lo contrario, INSTR convierte el valor en una cadena antes de evaluarla.
<i>search_value</i>	Obligatorio	Cualquier valor. El valor de búsqueda distingue entre mayúsculas y minúsculas. El conjunto de caracteres que desea buscar. El argumento search_value debe coincidir con una parte de la cadena. Por ejemplo, si escribe INSTR('Alfred Pope', 'Alfred Smith'), la función devuelve 0. Puede especificar cualquier expresión de transformación válida. Si desea buscar una cadena de caracteres, encierre entre comillas simples los caracteres que desea buscar, por ejemplo 'abc'.

Argumento	Obligatorio/ opcional	Descripción
<i>start</i>	Opcional	<p>Debe ser un entero. La posición en la cadena donde desea iniciar la búsqueda. Puede especificar cualquier expresión de transformación válida.</p> <p>El valor predeterminado es 1, lo que significa que INSTR inicia la búsqueda en el primer carácter de la cadena.</p> <p>Si la posición de inicio es 0, INSTR busca a partir del primer carácter de la cadena. Si la posición de inicio es un número positivo, INSTR ubica la posición de inicio contando desde el principio de la cadena. Si la posición de inicio es un número negativo, INSTR ubica la posición de inicio contando desde el final de la cadena. Si omite este argumento, la función utiliza el valor predeterminado 1.</p>
<i>occurrence</i>	Opcional	<p>Un entero positivo mayor que 0. Puede especificar cualquier expresión de transformación válida. Si el valor de búsqueda aparece más de una vez en la cadena, puede especificar la ocurrencia que desea buscar. Por ejemplo, debe indicar 2 si desea buscar la segunda ocurrencia a partir de la posición inicial.</p> <p>Si omite este argumento, la función utiliza el valor predeterminado 1, lo que significa que INSTR busca la primera ocurrencia del valor de búsqueda. Si transfiere un decimal, el Servicio de integración de datos lo redondea al valor entero más próximo. Si transfiere un entero negativo o 0, la sesión genera un error.</p>
<i>comparison_type</i>	Opcional	<p>El tipo de comparación de la cadena, ya sea lingüística o binaria, cuando el Servicio de integración de datos se ejecuta en modo Unicode. Cuando el Servicio de integración de datos se ejecuta en modo ASCII, el tipo de comparación siempre es binaria.</p> <p>Las comparaciones lingüísticas tienen en cuenta las reglas de agrupación específicas de cada idioma, mientras que las comparaciones binarias efectúan comparaciones por bit. Por ejemplo, el carácter ß del alemán coincide con la cadena "ss" en una comparación lingüística, pero no en una comparación binaria. Las comparaciones binarias son más rápidas que las comparaciones lingüísticas.</p> <p>Debe ser un entero, ya sea 0 ó 1:</p> <ul style="list-style-type: none"> - 0: INSTR efectúa una comparación lingüística de las cadenas. - 1: INSTR efectúa una comparación binaria de las cadenas. <p>El valor predeterminado es 0.</p> <p>Si introduce 0, el orden de clasificación de la sesión no debe ser binario.</p>

Valor de devolución

Un entero si la búsqueda es satisfactoria. El entero representa la posición del primer carácter en el argumento *search_value*, contando de izquierda a derecha.

0 si la búsqueda no es satisfactoria.

NULL si el valor que se ha transferido a la función es NULL.

Ejemplos

La siguiente expresión devuelve la posición de la primera ocurrencia de la letra 'a', empezando por el principio de cada nombre de la compañía. Dado que el argumento *search_value* distingue entre mayúsculas y minúsculas, la función omite la letra 'A' de 'Blue Fin Aqua Center', y devuelve la posición para la 'a' de 'Aqua':

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

COMPANY	RETURN VALUE
Blue Fin Aqua Center	13
Maco Shark Shop	2
Scuba Gear	5
Frank's Dive Shop	3
VIP Diving Club	0

La siguiente expresión devuelve la posición de la segunda ocurrencia de la letra 'a', empezando por el principio de cada nombre de la compañía. Dado que el argumento *search_value* distingue entre mayúsculas y minúsculas, la función omite la letra 'A' de 'Blue Fin Aqua Center', y devuelve 0:

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

COMPANY	RETURN VALUE
Blue Fin Aqua Center	0
Maco Shark Shop	8
Scuba Gear	9
Frank's Dive Shop	0
VIP Diving Club	0

La siguiente expresión devuelve la posición de la segunda ocurrencia de la letra 'a' de cada nombre de compañía, empezando por el último carácter del nombre de la compañía. Dado que el argumento *search_value* distingue entre mayúsculas y minúsculas, la función omite la letra 'A' de 'Blue Fin Aqua Center', y devuelve 0:

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

COMPANY	RETURN VALUE
Blue Fin Aqua Center	0
Maco Shark Shop	2
Scuba Gear	5

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

La siguiente expresión devuelve la posición del primer carácter de la cadena 'Blue Fin Aqua Center' (empezando por el último carácter del nombre de la compañía):

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

COMPANY	RETURN VALUE
Blue Fin Aqua Center	1
Maco Shark Shop	0
Scuba Gear	0
Frank's Dive Shop	0
VIP Diving Club	0

Uso de INSTR anidadas

Puede anidar la función INSTR con otras funciones para llevar a cabo tareas más complejas.

La siguiente expresión evalúa una cadena empezando desde el final. La expresión busca el último espacio (el más a la derecha) de la cadena y devuelve todos los caracteres a su izquierda:

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

CUST_NAME	RETURN VALUE
PATRICIA JONES	PATRICIA
MARY ELLEN SHAH	MARY ELLEN

La siguiente expresión quita el carácter '#' de una cadena:

```
SUBSTR( CUST_ID, 1, INSTR(CUST_ID, '#')-1 ) || SUBSTR( CUST_ID, INSTR(CUST_ID, '#')+1 )
```

CUST_ID	RETURN VALUE
ID#33	ID33
#A3577	A3577
SS #712403399	SS 712403399

ISNULL

Devuelve si un valor es NULL. ISNULL evalúa una cadena vacía como FALSE.

Nota: Para comprobar las cadenas vacías, use LENGTH.

Sintaxis

```
ISNULL( value )
```

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

Argumento	Obligatorio / Opcional	Descripción
value	Obligatorio	Cualquier tipo de datos excepto el tipo de datos binario. Pasa las filas que se van a evaluar. Puede especificar cualquier expresión de transformación válida.

Valor devuelto

Se devuelve TRUE (1) si el valor es NULL.

Se devuelve FALSE (0) si el valor no es NULL.

Ejemplo

La siguiente expresión comprueba los valores nulos de la tabla de elementos:

```
ISNULL( ITEM_NAME )
```

ITEM_NAME	RETURN VALUE
Flashlight	0 (FALSE)
NULL	1 (TRUE)
Regulator system	0 (FALSE)
' '	0 (FALSE) <i>Empty string is not NULL</i>

IS_DATE

Devuelve si el valor de una cadena es una fecha válida. Una fecha válida es una cadena expresada en la parte de fecha del formato de fecha especificado en la sesión. Si la cadena que desea probar no está en este formato de fecha, utilice la cadena de formato TO_DATE para especificar el formato de fecha. Si las cadenas transferidas a IS_DATE no coinciden con la cadena de formato especificada, la función devuelve FALSE (0). Si las cadenas coinciden con la cadena de formato, la función devuelve TRUE (1).

IS_DATE verifica las cadenas y devuelve un valor entero.

El puerto de salida de una expresión IS_DATE debe ser un tipo de datos String o numérico.

Puede utilizar IS_DATE para probar o filtrar datos en un archivo sin formato antes de grabarlos en el destino.

Con IS_DATE, utilice la cadena de formato RR en lugar de YY. En la mayoría de casos, las dos cadenas de formato devuelven los mismos valores, pero hay algunos casos únicos en los que YY devuelve resultados incorrectos. Por ejemplo, la expresión IS_DATE('02/29/00', 'YY') se calcula de forma interna como IS_DATE(02/29/1900 00:00:00), que devuelve un valor FALSE. El Servicio de integración de datos , sin

embargo, calcula la expresión `IS_DATE('02/29/00', 'RR')` como `IS_DATE(02/29/2000 00:00:00)`, que devuelve el valor `TRUE`. En el primer caso, el año 1900 no es un año bisiesto, así que no hay 29 de febrero.

Nota: `IS_DATE` utiliza las mismas cadenas de formato que `TO_DATE`.

Sintaxis

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

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

Argumento	Obligatorio / Opcional	Descripción
<i>value</i>	Obligatorio	Debe ser un tipo de datos <code>String</code> . Transfiere las filas que desea verificar. Puede especificar cualquier expresión de transformación válida.
<i>format</i>	Opcional	<p>Especifique una cadena de formato <code>TO_DATE</code> válida. La cadena de formato debe coincidir con las partes del argumento <i>string</i> . Si transfiere por ejemplo, 'Mar 15 1997 12:43:10AM', debe utilizar la cadena de formato 'MON DD YYYY HH12:MI:SSAM'. Si omite la cadena de formato, el valor de la cadena debe estar en el formato de fecha especificado en la sesión.</p> <p>Especifique una cadena de formato <code>TO_DATE</code> válida. La cadena de formato debe coincidir con las partes del argumento <i>string</i> . Si transfiere por ejemplo, 'Mar 15 1997 12:43:10AM', debe utilizar la cadena de formato 'MON DD YYYY HH12:MI:SSAM'. Si omite la cadena de formato, el valor de la cadena debe estar en el formato de fecha especificado en la configuración de la asignación.</p>

Valor de devolución

`TRUE` (1) si la fila es una fecha válida.

`FALSE` (0) si la fila no es una fecha válida.

`NULL` si un valor en la expresión es `NULL` o si la cadena de formato es `NULL`.

advertencia: El formato de la cadena `IS_DATE` debe coincidir con la cadena de formato, incluidos los separadores de fecha. Si no coincide, el Servicio de integración de datos puede devolver valores inexactos u omitir el registro.

Ejemplos

La siguiente expresión verifica si hay fechas válidas en el puerto `INVOICE_DATE`:

```
IS_DATE( INVOICE_DATE )
```

Esta expresión devuelve datos similares a los siguientes:

INVOICE_DATE	RETURN VALUE
<code>NULL</code>	<code>NULL</code>
<code>'180'</code>	<code>0 (FALSE)</code>
<code>'04/01/98'</code>	<code>0 (FALSE)</code>
<code>'04/01/1998 00:12:15.7008'</code>	<code>1 (TRUE)</code>

INVOICE_DATE	RETURN VALUE
'02/31/1998 12:13:55.9204'	0 (FALSE) <i>(February does not have 31 days)</i>
'John Smith'	0 (FALSE)

La siguiente expresión IS_DATE especifica una cadena de formato de 'YYYY/MM/DD':

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

Si el valor de la cadena no coincide con este formato, IS_DATE devuelve FALSE:

INVOICE_DATE	RETURN VALUE
NULL	NULL
'180'	0 (FALSE)
'04/01/98'	0 (FALSE)
'1998/01/12'	1 (TRUE)
'1998/11/21 00:00:13'	0 (FALSE)
'1998/02/31'	0 (FALSE) <i>(February does not have 31 days)</i>
'John Smith'	0 (FALSE)

El siguiente ejemplo muestra cómo se debe utilizar IS_DATE para probar los datos antes de utilizar TO_DATE para convertir las cadenas en fechas. Esta expresión verifica los valores del puerto INVOICE_DATE y convierte las fechas válidas en un valor de fecha. Si el valor no es una fecha válida, el Servicio de integración de datos devuelve ERROR y omite la fila.

Este ejemplo devuelve un valor de fecha/hora. Por tanto, el puerto de salida para la expresión debe ser Fecha/Hora:

```
IIF( IS_DATE ( INVOICE_DATE, 'YYYY/MM/DD' ), TO_DATE( INVOICE_DATE ), ERROR('Not a valid date' ) )
```

INVOICE_DATE	RETURN VALUE
NULL	NULL
'180'	'Not a valid date'
'04/01/98'	'Not a valid date'
'1998/01/12'	1998/01/12
'1998/11/21 00:00:13'	'Not a valid date'
'1998/02/31'	'Not a valid date'
'John Smith'	'Not a valid date'

IS_NUMBER

Devuelve si una cadena es un número válido. Un número válido se compone de las siguientes partes:

- Espacio opcional antes del número
- Signo opcional (+/-)
- Uno o más dígitos con un punto decimal opcional
- Notación científica opcional, como la letra 'e' o 'E' (y la letra 'd' o 'D' en Windows), seguido por un signo opcional (+/-), seguido de uno o más dígitos
- Espacio en blanco opcional después del número

Los números siguientes son válidos:

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

El puerto de salida para una expresión IS_NUMBER debe ser un tipo de datos String o numérico.

Puede usar IS_NUMBER para probar o filtrar los datos de un archivo sin formato antes de escribirlos en un destino.

Sintaxis

```
IS_NUMBER( value )
```

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

Argumento	Obligatorio / Opcional	Descripción
value	Obligatorio	Debe ser un tipo de datos String. Pasa las filas que desea evaluar. Se puede introducir cualquier expresión de transformación válida.

Valor de retorno

TRUE (1) si la fila es un número válido.

FALSE (0) si la fila no es un número válido.

NULL si un valor de la expresión es NULL.

Ejemplos

La expresión siguiente comprueba si el puerto ITEM_PRICE tiene números válidos:

```
IS_NUMBER( ITEM_PRICE )
```

ITEM_PRICE	RETURN VALUE
'123.00 '	1 (True)
'-3.45e+3 '	1 (True)

ITEM_PRICE	RETURN VALUE
'-3.45D-3'	1 (True - Windows only)
'-3.45d-3'	0 (False - UNIX only)
'3.45E-'	0 (False) <i>Incomplete number</i>
' '	0 (False) <i>Consists entirely of blanks</i>
''	0 (False) <i>Empty string</i>
'+123abc'	0 (False)
' 123'	1 (True) <i>Leading white blanks</i>
'123 '	1 (True) <i>Trailing white blanks</i>
'ABC'	0 (False)
'-ABC'	0 (False)
NULL	NULL

Utilice IS_NUMBER para probar los datos antes de utilizar una de las funciones de conversión numéricas, como TO_FLOAT. Por ejemplo, la expresión siguiente comprueba los valores en el puerto ITEM_PRICE y convierte cada número válido en un valor de coma flotante de precisión doble. Si el valor no es un número válido, el Servicio de integración de datos devuelve 0,00:

```
IIF( IS_NUMBER ( ITEM_PRICE ), TO_FLOAT( ITEM_PRICE ), 0.00 )
```

ITEM_PRICE	RETURN VALUE
'123.00'	123
'-3.45e+3'	-3450
'3.45E-3'	0.00345
' '	0.00 <i>Consists entirely of blanks</i>
''	0.00 <i>Empty string</i>
'+123abc'	0.00
' ' 123ABC'	0.00
'ABC'	0.00
'-ABC'	0.00
NULL	NULL

IS_SPACES

Devuelve si el valor de una cadena se compone únicamente de espacios. Un espacio es un espacio en blanco, un salto de página, un salto de línea, un retorno de carro, un tabulador o un tabulador vertical.

IS_SPACES devuelve FALSE en una cadena vacía porque no contiene espacios. Para comprobar una cadena vacía, use LENGTH.

Sintaxis

```
IS_SPACES( value )
```

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

Argumento	Obligatorio / Opcional	Descripción
value	Obligatorio	Debe ser un tipo de datos String. Pasa las filas que desea evaluar. Se puede introducir cualquier expresión de transformación válida.

Valor de retorno

TRUE (1) si la fila se compone únicamente de espacios.

FALSE (0) si la fila contiene datos.

NULL si un valor de la expresión es NULL.

Ejemplo

La expresión siguiente comprueba si las filas del puerto ITEM_NAME se componen únicamente de espacios:

```
IS_SPACES( ITEM_NAME )
```

ITEM_NAME	RETURN VALUE
Flashlight	0 (False)
	1 (True)
Regulator system	0 (False)
NULL	NULL
' '	0 (FALSE) (Empty string does not contain spaces.)

Sugerencia: Utilice IS_SPACES para evitar la escritura de espacios en una columna de caracteres de una tabla de destino. Por ejemplo, si tiene una transformación que escribe nombres de clientes en una columna CHAR(5) de longitud fija de una tabla de destino, podría escribir '00000' en lugar de espacios. Puede crear una expresión similar a la siguiente:

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


LAST

Devuelve la última fila del puerto seleccionado. Si lo desea, puede aplicar un filtro para limitar las filas que lee el Servicio de integración de datos. Sólo puede anidar otra función de agregado con LAST.

Sintaxis

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

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

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

Valor de retorno

Última fila de un puerto.

NULL si todos los valores pasados a la función son NULL o si no hay filas seleccionadas (por ejemplo, si la condición de filtro da como resultado FALSE o NULL para todas las filas).

Nota: De forma predeterminada, el Servicio de integración de datos trata los valores nulos como NULL en las funciones agregadas. Si pasa un puerto o grupo de valores nulos completo, la función devuelve NULL. Sin embargo, cuando se configura el Servicio de integración de datos, se puede indicar la forma en que se desea manejar los valores nulos en las funciones agregadas. En las funciones agregadas se pueden tratar los valores nulos como 0 o como NULL.

Ejemplo

La expresión siguiente devuelve la última fila del puerto ITEMS_NAME con un precio mayor de 10,00 \$:

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

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

LAST_DAY

Devuelve la fecha del último día del mes para cada fecha en un puerto.

Sintaxis

```
LAST_DAY( date )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>date</i>	Obligatorio	Tipo de datos de fecha/hora. Pasa las fechas de las que desea devolver el último día del mes. Puede introducir cualquier expresión de transformación válida que dé como resultado una fecha.

Valor de retorno

Fecha. El último día del mes para el valor de la fecha que se pase a esta función.

NULL si un valor del puerto seleccionado es NULL.

Nulo

Si uno de los valores es NULL, LAST_DAY ignora la fila. Sin embargo, si todos los valores pasados desde el puerto son NULL, LAST_DAY devuelve NULL.

Agrupar por

LAST_DAY agrupa los valores en base a los puertos de agrupación que se hayan definido en la transformación, devolviendo un resultado para cada grupo. Si no hay un puerto de agrupación, LAST_DAY trata todas las filas como un solo grupo, devolviendo un solo valor.

Ejemplos

La siguiente expresión devuelve el último día del mes para cada fecha del puerto ORDER_DATE.

```
LAST_DAY( ORDER_DATE )
```

ORDER_DATE	RETURN VALUE
Apr 1 1998 12:00:00AM	Apr 30 1998 12:00:00AM
Jan 6 1998 12:00:00AM	Jan 31 1998 12:00:00AM
Feb 2 1996 12:00:00AM	Feb 29 1996 12:00:00AM (Leap year)
NULL	NULL
Jul 31 1998 12:00:00AM	Jul 31 1998 12:00:00AM

Puede anidar TO_DATE para convertir los valores de cadena en una fecha. TO_DATE siempre incluye la información de la hora. Si pasa una cadena que no tiene un valor de hora, la fecha devuelta incluye la hora 00:00:00.

El siguiente ejemplo devuelve el último día del mes para cada fecha de pedido en el mismo formato que la cadena.

```
LAST_DAY( TO_DATE( ORDER_DATE, 'DD-MON-YY' ) )
```

ORDER_DATE	RETURN VALUE
'18-NOV-98'	Nov 30 1998 00:00:00
'28-APR-98'	Apr 30 1998 00:00:00
NULL	NULL
'18-FEB-96'	Feb 29 1996 00:00:00 (<i>Leap year</i>)

LEAST

Devuelve el valor menor de una lista de valores de entrada. De forma predeterminada, la coincidencia distingue mayúsculas de minúsculas.

Sintaxis

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

```
LEAST( value1, [value2, ..., valueN,] )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>valor</i>	Obligatorio	Cualquier tipo de datos a excepción de los binarios. El tipo de datos debe ser compatible con otros valores. Valor que desea comparar con otros valores. Debe introducir por lo menos un argumento de valor. Si el valor es numérico y otros valores de entrada son de otros tipos de datos numéricos, todos los valores usan la mayor precisión posible. Por ejemplo, si el tipo de datos de algunos valores es Entero y el de otros es Doble, el Servicio de integración de datos convierte los valores en Doble.
<i>CaseFlag</i>	Opcional	Debe ser un entero. Especifique un valor cuando el argumento de valor de entrada sea un valor de cadena. Determina si en los argumentos de esta función se distingue entre mayúsculas y minúsculas. Se puede especificar cualquier expresión de transformación válida. Cuando CaseFlag es un número distinto de 0, la función distingue entre mayúsculas y minúsculas. Cuando CaseFlag es 0, la función no distingue entre mayúsculas y minúsculas. El valor predeterminado distingue mayúsculas de minúsculas.

Valor devuelto

value1 si es el menor de los valores de entrada, *value2* si es el menor de los valores de entrada, y así sucesivamente.

NULL si alguno de los argumentos es nulo.

Ejemplo

La expresión siguiente devuelve la cantidad menor de los artículos pedidos:

```
LEAST( QUANTITY1, QUANTITY2, QUANTITY3 )
```

QUANTITY1	QUANTITY2	QUANTITY3	RETURN VALUE
150	756	27	27
			NULL
5000	97	17	17
120	1724	965	120

LENGTH

Devuelve el número de caracteres de una cadena, incluidos los espacios en blanco finales.

Sintaxis

```
LENGTH( string )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>string</i>	Obligatorio	Tipo de datos String. Cadenas que se van a evaluar. Puede especificar cualquier expresión de transformación válida.

Valor devuelto

Entero que representa la longitud de la cadena.

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

Ejemplo

La siguiente expresión devuelve la longitud del nombre de cada cliente:

```
LENGTH( CUSTOMER_NAME )
```

CUSTOMER_NAME	RETURN VALUE
Bernice Davis	13
NULL	NULL
John Baer	9
Greg Brown	10

Sugerencias para LENGTH

Use LENGTH para comprobar las condiciones de las cadenas vacías. Si desea buscar campos en los que no se incluye el nombre del cliente, use una expresión como la siguiente:

```
IIF( LENGTH( CUSTOMER_NAME ) = 0, 'EMPTY STRING' )
```

Para comprobar los campos nulos, use ISNULL. Para comprobar los espacios, use IS_SPACES.

LN

Devuelve el logaritmo natural de un valor numérico. Por ejemplo, LN(3) devuelve 1,098612. Esta función se suele usar para analizar los datos científicos en lugar de los datos empresariales.

Esta función tiene una correspondencia recíproca con la función EXP.

Sintaxis

```
LN( numeric_value )
```

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

Argumento	Obligatorio / Opcional	Descripción
numeric_value	Obligatorio	Tipo de datos numérico. Debe ser un número positivo mayor que 0. Pasa los valores para los que se va a calcular el logaritmo natural. Puede especificar cualquier expresión de transformación válida.

Valor devuelto

Valor doble.

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

Ejemplo

La siguiente expresión devuelve el logaritmo natural para todos los valores del puerto NUMBERS:

```
LN( NUMBERS )
```

NUMBERS	RETURN VALUE
10	2.302585092994
125	4.828313737302
0.96	-0.04082199452026
NULL	NULL
-90	Error. (The Integration Service does not write row.)
0	Error. (The Integration Service does not write row.)

Nota: El Servicio de integración de datos muestra un error y no escribe la fila si pasa un número negativo o 0. El valor de *numeric_value* debe ser un número positivo mayor que 0.

LOG

Devuelve el logaritmo de un valor numérico. Muy a menudo, se utiliza esta función para analizar los datos científicos en lugar de los datos empresariales.

Sintaxis

`LOG(base, exponent)`

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

Argumento	Obligatorio / Opcional	Descripción
<i>base</i>	Obligatorio	La base del logaritmo. Debe ser un valor positivo numérico distinto de 0 ó 1. Cualquier expresión de transformación válida cuyo resultado es un número positivo distinto de 0 ó 1.
<i>exponent</i>	Obligatorio	El exponente del logaritmo. Debe ser un valor numérico positivo mayor que 0. Cualquier expresión de transformación válida cuyo resultado sea un número positivo mayor que 0.

Valor de retorno

Valor doble.

NULL si el valor pasado a la función es NULL.

Ejemplo

La siguiente expresión devuelve el logaritmo de todos los valores del puerto NUMBERS:

`LOG(BASE, EXPONENT)`

BASE	EXPONENT	RETURN VALUE
15	1	0
.09	10	-0.956244644696599
NULL	18	NULL
35.78	NULL	NULL
-9	18	Error. (Servicio de integración de datos does not write the row.)
0	5	Error. (Servicio de integración de datos does not write the row.)
10	-2	Error. (Servicio de integración de datos does not write the row.)

El Servicio de integración de datos muestra un error y no escribe la fila si se pasa un número negativo, 0 ó 1 como valor base o si se pasa un valor negativo para el exponente.

LOOKUP

Busca un valor en una columna de origen de búsqueda.

La función LOOKUP compara los datos de un origen de búsqueda con un valor que usted especifique. Cuando el Servicio de integración de datos encuentra el valor de búsqueda en la tabla de búsqueda, devuelve el valor desde una columna especificada en la misma fila de la tabla de búsqueda.

Cuando crea una sesión que se basa en una asignación que utiliza la función LOOKUP, debe especificar las conexiones de la base de datos para \$Source Connection Value y \$Target Connection Value en las propiedades de la sesión. Para validar una función de búsqueda en una transformación de expresión, asegúrese de que la definición de búsqueda esté en la asignación.

Nota: Esta función no es compatible con los mapplets.

Uso de transformación de búsqueda o de la función LOOKUP

Utilice la *transformación* de búsqueda en lugar de la *función* LOOKUP para buscar valores en las asignaciones de PowerCenter. Si utiliza la función LOOKUP en una asignación, debe habilitar la opción de caché de búsqueda para compatibilidad 3.5 en las propiedades de la sesión. Esta opción existe explícitamente para los usuarios de PowerMart 3.5 que desean continuar utilizando la función LOOKUP, en lugar de crear transformaciones de búsqueda. Si desea más información, consulte "Transformación de búsqueda" en la *Guía de transformación de PowerCenter*.

En una función LOOKUP puede definir varias búsquedas para una tabla de búsqueda. Para devolver el valor de búsqueda, cada búsqueda debe encontrar, sin embargo, un valor coincidente.

Sintaxis

```
LOOKUP( result, search1, value1 [, search2, value2]... )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>result</i>	Obligatorio	Cualquier tipo de datos a excepción de los binarios. Debe ser un puerto de salida en la misma tabla de búsqueda que la búsqueda. Especifica el valor de devolución si la búsqueda coincide con el valor. Delante de este argumento siempre debe haber el calificador de referencia :TD.
<i>search1</i>	Obligatorio	El tipo de datos debe coincidir con <i>value1</i> . Debe ser un puerto de salida en la misma tabla de búsqueda que el resultado. Especifica los valores que desea que coincidan con el valor. Delante de este argumento siempre debe haber el calificador de referencia :TD.
<i>value1</i>	Obligatorio	Cualquier tipo de datos a excepción de los binarios. Debe coincidir con el tipo de datos de <i>search1</i> . Los valores que desea buscar en la columna de origen de búsqueda especificados en <i>search1</i> . Puede especificar cualquier expresión de transformación válida.

Valor de devolución

Result si todas las búsquedas encuentran valores coincidentes. Si el Servicio de integración de datos encuentra valores coincidentes, devuelve el resultado desde la misma fila que el argumento *search1*.

NULL si la búsqueda no encuentra ningún valor coincidente.

Error si la búsqueda encuentra más de un valor coincidente.

Ejemplo:

La siguiente expresión busca en el origen de búsqueda :TD.SALES un determinado ID de artículo y precio, y devuelve el nombre del artículo si ambas búsquedas encuentran una coincidencia.

```
LOOKUP( :TD.SALES.ITEM_NAME, :TD.SALES.ITEM_ID, 10, :TD.SALES.PRICE, 15.99 )
```

ITEM_NAME	ITEM_ID	PRICE
Regulator	5	100.00
Flashlight	10	15.99
Halogen Flashlight	15	15.99
NULL	20	15.99

RETURN VALUE: Flashlight

Consejos para LOOKUP

Cuando compara valores char y varchar, la función LOOKUP solo devuelve un resultado si las dos filas coinciden. Esto significa que deben coincidir el valor y la longitud de cada fila. Si el origen de búsqueda es un valor char de relleno de una determinada longitud y la búsqueda es un valor varchar, debe utilizar la función RTRIM para recortar los espacios en blanco al final del origen de búsqueda para que los valores coincidan con la búsqueda:

```
LOOKUP(:TD.ORDERS.PRICE, :TD.ORDERS.ITEM, RTRIM( ORDERS.ITEM, ' '))
```

Utilice el calificador de referencia :TD en los argumentos *result* y *search* de una función LOOKUP:

```
LOOKUP(:TD.ORDERS.ITEM, :TD.ORDERS.PRICE, ORDERS.PRICE, :TD.ORDERS.QTY, ORDERS.QTY)
```

LOWER

Convierte a minúsculas los caracteres de cadenas en mayúsculas.

Sintaxis

```
LOWER( string )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>cadena</i>	Obligatorio	Cualquier valor de cadena. El argumento pasa los valores de cadena que se van a devolver en minúsculas. Puede especificar cualquier expresión de transformación válida que tenga como resultado una cadena.

Valor devuelto

Cadena de caracteres en minúsculas. Si los datos contienen caracteres multibyte, el valor devuelto depende de la página de códigos y el modo de movimiento de datos del servicio de integración.

NULL si un valor del puerto seleccionado es NULL.

Ejemplo

La siguiente expresión devuelve todos los nombres en minúsculas:

```
LOWER( FIRST_NAME )
```

FIRST_NAME	RETURN VALUE
antonia	antonia
NULL	NULL
THOMAS	thomas
PierRe	pierre
BERNICE	bernice

LPAD

Añade un conjunto de espacios o caracteres al inicio de una cadena para convertirla a una longitud especificada.

Sintaxis

```
LPAD( first_string, length [,second_string] )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>first_string</i>	Obligatorio	Puede ser una cadena de carácter. Las cadenas que se desea cambiar. Se puede especificar cualquier expresión de transformación válida.
<i>length</i>	Obligatorio	Debe ser un literal entero positivo. Este argumento especifica la longitud que desea que tenga cada cadena.
<i>second_string</i>	Opcional	Puede ser cualquier valor de cadena. Los caracteres que desea anexas a la izquierda de los valores <i>first_string</i> . Se puede especificar cualquier expresión de transformación válida. Se puede introducir un literal de cadena específico. Sin embargo, delimite los caracteres que desea añadir al inicio de la cadena mediante comillas simples, como 'abc'. Este argumento distingue entre mayúsculas y minúsculas. Si se omite <i>second_string</i> , la función rellena el inicio de la primera cadena con espacios.

Valor de devolución

Cadena de la longitud especificada.

NULL si un valor pasado a la función es NULL o si *length* es un número negativo.

Ejemplos

La siguiente expresión estandariza números a seis dígitos añadiendo ceros al inicio de estos:

```
LPAD( PART_NUM, 6, '0')
```

PART_NUM	RETURN VALUE
702	000702
1	000001
0553	000553
484834	484834

LPAD cuenta la longitud de izquierda a derecha. Si la primera cadena es más larga que la longitud, LPAD trunca la cadena de derecha a izquierda. Por ejemplo, LPAD('alphabetical', 5, 'x') devuelve la cadena 'alpha'.

Si la segunda cadena es más larga que el número total de caracteres necesarios para devolver la longitud especificada, LPAD utiliza una porción de la segunda cadena:

```
LPAD( ITEM_NAME, 16, '*.*.*' )
```

ITEM_NAME	RETURN VALUE
Flashlight	*.*.*.Flashlight
Compass	*.*.*.*.Compass
Regulator System	Regulator System
Safety Knife	*.*.*Safety Knife

LTRIM

Quita espacios en blanco o caracteres del inicio de una cadena. Puede utilizar LTRIM con IIF o DECODE en una transformación de expresión o estrategia de actualización para evitar espacios en la tabla de destino.

Si no especifica un parámetro *trim_set* en la expresión:

- En UNICODE, LTRIM quita los espacios de uno y dos bytes del principio de una cadena.
- En ASCII, LTRIM quita solamente los espacios de un byte.

Si utiliza LTRIM para quitar los caracteres de una cadena, LTRIM compara, carácter por carácter, el argumento *trim_set* con cada carácter del argumento *string*, empezando por la derecha de la cadena. Si el carácter de la cadena coincide con cualquier carácter de *trim_set*, LTRIM lo quita. LTRIM sigue la comparación y quita los caracteres hasta que no puede encontrar un carácter coincidente en *trim_set*. A continuación, devuelve la cadena, que ya no incluye los caracteres coincidentes.

Sintaxis

```
LTRIM( string [, trim_set] )
```

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

Argumentos	Obligatorio / Opcional	Descripción
<i>string</i>	Obligatorio	Cualquier valor de cadena. Transfiere las cadenas que desea modificar. Puede especificar cualquier expresión de transformación válida. Utilice operadores para efectuar las comparaciones o para concatenar las cadenas antes de quitar los caracteres del principio de una cadena.
<i>trim_set</i>	Opcional	Cualquier valor de cadena. Transfiere los caracteres que desea quitar del principio de la primera cadena. Puede especificar cualquier expresión de transformación válida. También puede especificar una cadena de caracteres. Sin embargo, debe encerrar entre comillas simples los caracteres que desea quitar del principio de la cadena, por ejemplo, 'abc'. Si omite la segunda cadena, la función quita los espacios en blanco del principio de la cadena. LTRIM distingue entre mayúsculas y minúsculas. Si desea quitar, por ejemplo, el carácter 'A' de la cadena 'Alfredo', debe especificar 'A' y no 'a'.

Valor de devolución

Cadena. Se quitan los valores de cadena con los caracteres especificados en el argumento *trim_set*.

NULL si el valor que se ha transferido a la función es NULL. Si *trim_set* es NULL, la función devuelve NULL.

Ejemplo:

La siguiente expresión quita los caracteres 'S' y '.' de las cadenas del puerto LAST_NAME:

```
LTRIM( LAST_NAME, 'S.')
```

LAST_NAME	RETURN VALUE
Nelson	Nelson
Osborne	Osborne
NULL	NULL
S. MacDonald	MacDonald
Sawyer	awyer
H. Bender	H. Bender
Steadman	teadman

LTRIM quita 'S.' de S. MacDonald y la 'S' de Sawyer y Steadman, pero no el punto de H. Bender. El motivo es que LTRIM busca, carácter por carácter, el conjunto de caracteres especificado en el argumento *trim_set*. Si el primer carácter de la cadena coincide con el primer carácter de *trim_set*, LTRIM lo quita. LTRIM se centra a continuación en el segundo carácter de la cadena. Si coincide con el segundo carácter en *trim_set*, LTRIM lo quita, y así sucesivamente. Cuando el primer carácter de la cadena no coincide con el carácter correspondiente de *trim_set*, LTRIM devuelve la cadena y verifica la siguiente fila.

En el ejemplo de H. Bender, H no coincide con ningún carácter del argumento *trim_set*, así que LTRIM devuelve la cadena en el puerto LAST_NAME y prosigue con la siguiente fila.

Consejos para LTRIM

Utilice RTRIM y LTRIM con || o CONCAT para quitar los espacios en blanco al principio o al final después de concatenar dos cadenas.

Si anida LTRIM, también puede quitar varios conjuntos de caracteres. Por ejemplo, si desea quitar los espacios en blanco al principio y el carácter 'T' de una columna de nombres, puede crear una expresión similar a la siguiente:

```
LTRIM( LTRIM( NAMES ), 'T' )
```

MAKE_DATE_TIME

Devuelve la fecha y la hora en función de los valores de entrada.

Sintaxis

```
MAKE_DATE_TIME( year, month, day, hour, minute, second, nanosecond )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>year</i>	Obligatorio	Tipo de dato numérico. Número entero positivo de 4 dígitos. Si se pasa a esta función un año de 2 dígitos, el Servicio de integración de datos devuelve "00" como los dos primeros dígitos del año.
<i>month</i>	Obligatorio	Tipo de dato numérico. Número entero positivo entre 1 y 12 (enero = 1 y diciembre = 12).
<i>day</i>	Obligatorio	Tipo de dato numérico. Número entero positivo entre 1 y 31 (a excepción de los meses que tienen menos de 31 días: febrero, abril, junio, septiembre y noviembre).
<i>hour</i>	Opcional	Tipo de dato numérico. Número entero positivo entre 0 y 24 (donde 0=12 a.m., 12=12 p.m., y 24=12 a.m.).
<i>minute</i>	Opcional	Tipo de dato numérico. Entero positivo comprendido entre 0 y 59.
<i>second</i>	Opcional	Tipo de dato numérico. Entero positivo comprendido entre 0 y 59.
<i>nanosegundo</i>	Opcional	Tipo de dato numérico. Número entero positivo entre 0 y 999.999.999.

Valor de retorno

Fecha como MM/DD/YYYY HH24:MI:SS. Devuelve un valor nulo si no se pasa un año, un mes o un día a la función.

Ejemplo

La expresión siguiente crea una fecha y hora de los puertos de entrada:

```
MAKE_DATE_TIME( SALE_YEAR, SALE_MONTH, SALE_DAY, SALE_HOUR, SALE_MIN, SALE_SEC )
```

SALE_YR	SALE_MTH	SALE_DAY	SALE_HR	SALE_MIN	SALE_SEC	RETURN VALUE
2002	10	27	8	36	22	10/27/2002 08:36:22
2000	6	15	15	17		06/15/2000 15:17:00
2003	1	3		22	45	01/03/2003 00:22:45
04	3	30	12	5	10	03/30/0004 12:05:10
99	12	12	5		16	12/12/0099 05:00:16

MAX (Fechas)

Devuelve la última fecha encontrada en un puerto o grupo. Puede aplicar un filtro para limitar las filas de la búsqueda. Sólo puede anidar una función de agregados dentro de MAX.

También puede utilizar MAX para devolver el valor numérico más grande o el mayor valor de cadena en un puerto o en un grupo.

Sintaxis

```
MAX( date [, filter_condition] )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>date</i>	Obligatorio	Tipo de datos de fecha/hora. Pasa la fecha para la que desea devolver la fecha máxima. Puede introducir cualquier expresión de transformación válida.
<i>filter_condition</i>	Opcional	Limita las filas de la búsqueda. La condición de filtro debe ser un valor numérico o dar como resultado TRUE, FALSE o NULL. Puede introducir cualquier expresión de transformación válida.

Valor de retorno

Fecha.

NULL si todos los valores pasados a la función son NULL o si no hay filas seleccionadas (por ejemplo, si la condición de filtro da como resultado FALSE o NULL para todas las filas).

Ejemplo

Puede devolver la fecha máxima de un puerto o un grupo. La siguiente expresión devuelve la fecha máxima del pedido de linternas:

```
MAX( ORDERDATE, ITEM_NAME='Flashlight' )
```

ITEM_NAME	ORDER_DATE
Flashlight	Apr 20 1998
Regulator System	May 15 1998
Flashlight	Sep 21 1998
Diving Hood	Aug 18 1998
Flashlight	NULL

MAX (Números)

Devuelve el valor máximo encontrado dentro de un puerto o grupo. Se puede aplicar un filtro para limitar el número de filas en la búsqueda. Dentro de MAX solamente se puede anidar una función agregada adicional. También puede usar MAX para devolver la fecha más reciente o el valor de cadena más alto en un puerto o grupo.

Sintaxis

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

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

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

Valor de retorno

Valor numérico.

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

Nota: Si el valor devuelto es un decimal con una precisión superior a 15, puede habilitar el modo de alta precisión para asegurarse de obtener una precisión decimal de hasta 28 dígitos.

Nulos

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

Nota: De forma predeterminada, el Servicio de integración de datos trata los valores nulos como NULL en las funciones agregadas. Si pasa un puerto o grupo de valores nulos completo, la función devuelve NULL. Sin embargo, cuando se configura el Servicio de integración de datos, se puede indicar la forma en que se desea manejar los valores nulos en las funciones agregadas. En las funciones agregadas se pueden tratar los valores nulos como 0 o como NULL.

Agrupar por

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

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

Ejemplo

La primera expresión devuelve el precio máximo de las linternas:

```
MAX( PRICE, ITEM_NAME='Flashlight' )
```

ITEM_NAME	PRICE
Flashlight	10.00
Regulator System	360.00
Flashlight	55.00
Diving Hood	79.00
Halogen Flashlight	162.00
Flashlight	85.00
Flashlight	NULL
RETURN VALUE: 85.00	

MAX (Cadena)

Devuelve el valor de cadena más alto contenido en un puerto o grupo. Se puede aplicar un filtro para limitar el número de filas en la búsqueda. Dentro de MAX solamente se puede anidar una función agregada adicional.

Nota: La función MAX utiliza el mismo orden de clasificación que la transformación Sorter. Sin embargo, la función MAX distingue entre mayúsculas y minúsculas y la transformación Sorter puede no distinguir entre mayúsculas y minúsculas.

También puede usar MAX para devolver la fecha más reciente o el valor numérico más alto en un puerto o grupo.

Sintaxis

```
MAX( string [, filter_condition] )
```

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

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

Valor de retorno

Cadena.

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

Nulos

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

Nota: De forma predeterminada, el Servicio de integración de datos trata los valores nulos como NULL en las funciones agregadas. Si pasa un puerto o grupo de valores nulos completo, la función devuelve NULL. Sin embargo, cuando se configura el Servicio de integración de datos, se puede indicar la forma en que se desea manejar los valores nulos en las funciones agregadas. En las funciones agregadas se pueden tratar los valores nulos como 0 o como NULL.

Agrupar por

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

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

Ejemplo:

La siguiente expresión devuelve el nombre de artículo máximo del fabricante ID 104:

```
MAX( ITEM_NAME, MANUFACTURER_ID='104' )
```

MANUFACTURER_ID	ITEM_NAME
101	First Stage Regulator
102	Electronic Console
104	Flashlight
104	Battery (9 volt)
104	Rope (20 ft)
104	60.6 cu ft Tank
107	75.4 cu ft Tank

MANUFACTURER_ID	ITEM_NAME
108	Wristband Thermometer

RETURN VALUE: Rope (20 ft)

MD5

Calcula la suma de comprobación del valor de entrada. La función utiliza el algoritmo Message-Digest 5 (MD5). MD5 es una función hash criptográfica unidireccional con un valor de hash de 128 bits. Permite concluir que los valores de entrada son diferentes cuando las sumas de comprobación de los valores de entrada son diferentes. Utilice MD5 para comprobar la integridad de los datos.

Sintaxis

`MD5(value)`

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

Argumento	Obligatorio / Opcional	Descripción
<i>value</i>	Obligatorio	Tipo de datos String o Binary. Valor para el que se desea calcular la suma de comprobación. La distinción entre mayúsculas y minúsculas del valor de entrada afecta al valor de retorno. Por ejemplo, MD5(informatica) y MD5(Informatica) devuelven valores diferentes.

Valor de retorno

Cadena única de 32 caracteres de dígitos hexadecimales 0-9 y a-f.

NULL si la entrada es un valor nulo.

Ejemplo

Se desea grabar datos modificados en una base de datos. Utilice MD5 para generar valores de suma de comprobación para las filas de datos que se leen de un origen. Cuando ejecute una sesión, compare los valores de suma de comprobación generados con los nuevos valores de suma de comprobación. Luego, grabe en el destino las filas que tienen valores de suma de comprobación actualizados. Se puede concluir que un valor de suma de comprobación actualizado indica que los datos han sido modificados.

Se desea grabar datos modificados en una base de datos. Utilice MD5 para generar valores de suma de comprobación para las filas de datos que se leen de un origen. Cuando ejecute una asignación, compare los valores de suma de comprobación generados con los nuevos valores de suma de comprobación. Luego, grabe en el destino las filas que tienen valores de suma de comprobación actualizados. Se puede concluir que un valor de suma de comprobación actualizado indica que los datos han sido modificados.

Consejo

Puede utilizar el valor de retorno como clave hash.

MEDIAN

Devuelve la mediana de todos los valores de un puerto seleccionado.

Si hay un número par de valores en el puerto, la mediana es el promedio de los dos valores centrales de una serie de valores colocados en orden en una línea de números. Si en el puerto hay un número de valores impar, la mediana es el número central.

Dentro de MEDIAN solamente se puede anidar una función agregada adicional y la función anidada debe devolver un tipo de datos Numérico.

El Servicio de integración de datos lee todas las filas de datos para realizar el cálculo de la mediana. El proceso de lectura de filas de datos para realizar el cálculo puede afectar al rendimiento. Opcionalmente, se puede aplicar un filtro para limitar el número de filas que se leen para calcular la mediana.

Sintaxis

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

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

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

Valor de retorno

Valor numérico.

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

Nota: Si el valor devuelto es un decimal con una precisión superior a 15, puede habilitar el modo de alta precisión para asegurarse de obtener una precisión decimal de hasta 28 dígitos.

Nulos

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

Nota: De forma predeterminada, el Servicio de integración de datos trata los valores nulos como NULL en las funciones agregadas. Si pasa un puerto o grupo de valores nulos completo, la función devuelve NULL. Sin embargo, cuando se configura el Servicio de integración de datos, se puede indicar la forma en que se desea manejar los valores nulos en las funciones agregadas. En las funciones agregadas se pueden tratar los valores nulos como 0 o como NULL.

Agrupar por

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

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

Ejemplo

Para calcular la mediana del salario de todos los departamentos se crea una transformación de agregación agrupada por departamentos con un puerto que especifique la siguiente expresión:

```
MEDIAN( SALARY )
```

La siguiente expresión devuelve el valor de la mediana de los pedidos de arneses estabilizadores:

```
MEDIAN( SALES, ITEM = 'Stabilizing Vest' )
```

ITEM	SALES
Flashlight	85
Stabilizing Vest	504
Stabilizing Vest	36
Safety Knife	5
Medium Titanium Knife	150
Tank	NULL
Stabilizing Vest	441
Chisel Point Knife	60
Stabilizing Vest	NULL
Stabilizing Vest	1044
Wrist Band Thermometer	110
RETURN VALUE: 472.5	

METAPHONE

Cifra los valores de las cadenas. Puede especificar la longitud de la cadena que desea cifrar.

METAPHONE cifra los caracteres del alfabeto inglés (A-Z). Cifra las letras tanto mayúsculas como minúsculas en mayúsculas.

METAPHONE cifra los caracteres según la siguiente lista de reglas:

- Omite las vocales (A, E, I, O y U) a menos que una de ellas sea el primer carácter de la cadena de entrada. METAPHONE('CAR') devuelve 'KR' y METAPHONE('AAR') devuelve 'AR'.
- Utiliza directrices de codificación especiales.

La tabla siguiente enumera las directrices de codificación METAPHONE:

Entrada	Devuelve	Condición	Ejemplo
B	- n/a	- detrás de M	- METAPHONE ('Lamb') devuelve LM.
B	- B	- en el resto de casos	- METAPHONE ('Box') devuelve BKS.
C	- X	- cuando va seguido de IA o H	- METAPHONE ('Facial') devuelve FXL.
C	- S	- seguido de I, E o Y	- METAPHONE ('Fence') devuelve FNS.
C	- n/a	- detrás de S, y seguido de I, E o Y	- METAPHONE ('Scene') devuelve SN.
C	- K	- en el resto de casos	- METAPHONE ('Cool') devuelve KL.
D	- J	- cuando va seguido de GE, GY o GI	- METAPHONE ('Dodge') devuelve TJ.
D	- T	- en el resto de casos	- METAPHONE ('David') devuelve TFT.
F	- F	- en todos los casos	- METAPHONE ('FOX') devuelve FKS.
G	- F	- cuando va seguido de H y el primer carácter en la cadena de entrada no es B, D o H	- METAPHONE ('Tough') devuelve TF.
G	- n/a	- cuando va seguido de H y el primer carácter en la cadena de entrada es B, D o H	- METAPHONE ('Hugh') devuelve HF.
G	- J	- cuando va seguido de I, E o Y y no se repite	- METAPHONE ('Magic') devuelve MJK.
G	- K	- en el resto de casos	- METAPHONE ('GUN') devuelve KN.
H	- H	- cuando no va detrás de C, G, P, S o T y va seguido de A, E, I o U	- METAPHONE ('DHAT') devuelve THT.
H	- n/a	- en el resto de casos	- METAPHONE ('Chain') devuelve XN.
J	- J	- en todos los casos	- METAPHONE ('Jen') devuelve JN.
K	- n/a - K	- detrás de C - en el resto de casos	- METAPHONE ('Ckim') devuelve KM. - METAPHONE ('Kim') devuelve KM.
L	- L	- en todos los casos	- METAPHONE ('Laura') devuelve LR.
M	- M	- en todos los casos	- METAPHONE ('Maggi') devuelve MK.
N	- N	- en todos los casos	- METAPHONE ('Nancy') devuelve NNS.
P	- F	- cuando va seguido de H	- METAPHONE ('Phone') devuelve FN.
P	- P	- en el resto de casos	- METAPHONE ('Pip') devuelve PP.
Q	- K	- en todos los casos	- METAPHONE ('Queen') devuelve KN.
R	- R	- en todos los casos	- METAPHONE ('Ray') devuelve R.

Entrada	Devuelve	Condición	Ejemplo
S	- X	- cuando va seguido de H, IO, IA o CHW	- METAPHONE ('Cash') devuelve KX.
S	- S	- en el resto de casos	- METAPHONE ('Sing') devuelve SNK.
T	- X	- cuando va seguido de IA o IO	- METAPHONE ('Patio') devuelve PX.
T	- 0 ¹	- cuando va seguido de H	- METAPHONE ('Thor') devuelve 0R.
T	- n/a	- cuando va seguido de CH	- METAPHONE ('Glitch') devuelve KLTX.
T	- T	- en el resto de casos	- METAPHONE ('Tim') devuelve TM.
V	- F	- en todos los casos	- METAPHONE ('Vin') devuelve FN.
W	- W	- cuando va seguido de A, E, I, O o U	- METAPHONE ('Wang') devuelve WNK.
W	- n/a	- en el resto de casos	- METAPHONE ('When') devuelve HN.
X	- KS	- en todos los casos	- METAPHONE ('Six') devuelve SKS.
Y	- Y	- cuando va seguido de A, E, I, O o U	- METAPHONE ('Yang') devuelve YNK.
Y	- n/a	- en el resto de casos	- METAPHONE ('Bobby') devuelve BB.
Z	- S	- en todos los casos	- METAPHONE ('Zack') devuelve SK.

¹. El entero 0.

- Omite el carácter inicial y cifra la cadena restante si los dos primeros caracteres de la cadena de entrada tienen uno de los siguientes valores:
 - **KN**. Por ejemplo, METAPHONE('KNOT') devuelve 'NT'.
 - **GN**. Por ejemplo, METAPHONE('GNOB') devuelve 'NB'.
 - **PN**. Por ejemplo, METAPHONE('PNRX') devuelve 'NRKS'.
 - **AE**. Por ejemplo, METAPHONE('AERL') devuelve 'ERL'.
- Si en la cadena de entrada hay un carácter diferente de "C" que se repite más de una vez, se cifra solamente la primera ocurrencia. Por ejemplo, METAPHONE('BBOX') devuelve 'BKS' y METAPHONE('CCOX') devuelve 'KKKS'.

Sintaxis

```
METAPHONE( string [,length] )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>string</i>	Obligatorio	Debe ser una cadena de caracteres. Transfiere el valor que desea cifrar. El primer carácter debe ser un carácter del alfabeto inglés (A-Z). Se puede especificar cualquier expresión de transformación válida. Omite los caracteres no alfabéticos de <i>string</i> .
<i>length</i>	Opcional	Debe ser un entero mayor que 0. Especifica el número de caracteres de <i>string</i> que desea cifrar. Se puede especificar cualquier expresión de transformación válida. Cuando <i>length</i> es 0 o un valor mayor que la longitud de <i>string</i> , se cifra toda la cadena de entrada. El valor predeterminado es 0.

Valor de retorno

Cadena.

NULL si una de las siguientes condiciones es verdadera:

- Todos los valores transferidos a la función son NULL.
- No hay ningún carácter de *string* que sea una letra del alfabeto inglés.
- *string* está vacío.

Ejemplos

La siguiente expresión cifra los dos primeros caracteres en el puerto EMPLOYEE_NAME en una cadena:

```
METAPHONE( EMPLOYEE_NAME, 2 )
```

Employee_Name	Return Value
John	JH
*@#\$	NULL
P\$%%oC&&KMNL	PK

La siguiente expresión cifra los cuatro primeros caracteres en el puerto EMPLOYEE_NAME en una cadena:

```
METAPHONE( EMPLOYEE_NAME, 4 )
```

Employee_Name	Return Value
John	JHN
1ABC	ABK
*@#\$	NULL
P\$%%oC&&KMNL	PKKM

MIN (Fechas)

Devuelve la fecha más temprana encontrada en un puerto o grupo. Puede aplicar un filtro para limitar las filas de la búsqueda. Sólo puede anidar otra función de agregado con MIN y la función anidada debe devolver un tipo de datos de fecha.

También puede utilizar MIN para devolver el valor numérico más pequeño o el menor valor de cadena en un puerto o en un grupo.

Sintaxis

```
MIN( date [, filter_condition] )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>date</i>	Obligatorio	Tipo de datos de fecha/hora. Pasa los valores para los que desea devolver el valor mínimo. Se puede introducir cualquier expresión de transformación válida.
<i>filter_condition</i>	Opcional	Limita las filas de la búsqueda. La condición de filtro debe ser un valor numérico o dar como resultado TRUE, FALSE o NULL. Se puede introducir cualquier expresión de transformación válida.

Valor de retorno

Fecha si el argumento *value* es una fecha.

NULL si todos los valores pasados a la función son NULL o si no hay filas seleccionadas (por ejemplo, si la condición de filtro da como resultado FALSE o NULL para todas las filas).

Nulos

Si un solo valor es NULL, MIN lo ignora. Sin embargo, si todos los valores pasados desde el puerto son NULL, MIN devuelve NULL.

Agrupar por

MIN agrupa los valores en base a los puertos de agrupación que se hayan definido en la transformación, devolviendo un resultado para cada grupo.

Si no hay un puerto de agrupación, MIN trata a todas las filas como un solo grupo, devolviendo un solo valor.

Ejemplo

La siguiente expresión devuelve la fecha más antigua del pedido de linternas:

```
MIN( ORDER_DATE, ITEM_NAME='Flashlight' )
```

ITEM_NAME	ORDER_DATE
Flashlight	Apr 20 1998
Regulator System	May 15 1998
Flashlight	Sep 21 1998
Diving Hood	Aug 18 1998

ITEM_NAME	ORDER_DATE
Halogen Flashlight	Feb 1 1998
Flashlight	Oct 10 1998
Flashlight	NULL
RETURN VALUE: Feb 1 1998	

MIN (Números)

Devuelve el valor mínimo encontrado dentro de un puerto o grupo. Se puede aplicar un filtro para limitar el número de filas en la búsqueda. Dentro de MIN solamente se puede anidar una función agregada adicional y la función anidada debe devolver un tipo de datos numérico.

También puede usar MIN para devolver la fecha más reciente o el valor de cadena más bajo en un puerto o grupo.

Sintaxis

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

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

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

Valor de retorno

Valor numérico.

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

Nota: Si el valor devuelto es un decimal con una precisión superior a 15, puede habilitar el modo de alta precisión para asegurarse de obtener una precisión decimal de hasta 28 dígitos.

Nulos

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

Nota: De forma predeterminada, el Servicio de integración de datos trata los valores nulos como NULL en las funciones agregadas. Si pasa un puerto o grupo de valores nulos completo, la función devuelve NULL. Sin embargo, cuando se configura el Servicio de integración de datos, se puede indicar la forma en que se desea

manejar los valores nulos en las funciones agregadas. En las funciones agregadas se pueden tratar los valores nulos como 0 o como NULL.

Agrupar por

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

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

Ejemplo

La primera expresión devuelve el precio mínimo de las linternas:

```
MIN ( PRICE, ITEM_NAME='Flashlight' )
```

ITEM_NAME	PRICE
Flashlight	10.00
Regulator System	360.00
Flashlight	55.00
Diving Hood	79.00
Halogen Flashlight	162.00
Flashlight	85.00
Flashlight	NULL
RETURN VALUE: 10.00	

MIN (Cadena)

Devuelve el valor de cadena mínimo encontrado dentro de un puerto o grupo. Se puede aplicar un filtro para limitar el número de filas en la búsqueda. Dentro de MIN solamente se puede anidar una función agregada adicional y la función anidada debe devolver un tipo de datos String.

Nota: La función MIN utiliza el mismo orden de clasificación que la transformación Sorter. Sin embargo, la función MIN distingue entre mayúsculas y minúsculas y la transformación Sorter puede no distinguir entre mayúsculas y minúsculas.

También puede usar MIN para devolver la fecha más reciente o el valor numérico más bajo en un puerto o grupo.

Sintaxis

```
MIN( string [, filter_condition] )
```

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

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

Valor de retorno

Valor de cadena.

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

Nulos

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

Nota: De forma predeterminada, el Servicio de integración de datos trata los valores nulos como NULL en las funciones agregadas. Si pasa un puerto o grupo de valores nulos completo, la función devuelve NULL. Sin embargo, cuando se configura el Servicio de integración de datos, se puede indicar la forma en que se desea manejar los valores nulos en las funciones agregadas. En las funciones agregadas se pueden tratar los valores nulos como 0 o como NULL.

Agrupar por

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

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

Ejemplo

La siguiente expresión devuelve el nombre de artículo mínimo del fabricante ID 104:

```
MIN ( ITEM_NAME, MANUFACTURER_ID='104' )
```

MANUFACTURER_ID	ITEM_NAME
101	First Stage Regulator
102	Electronic Console
104	Flashlight
104	Battery (9 volt)
104	Rope (20 ft)
104	60.6 cu ft Tank
107	75.4 cu ft Tank

MANUFACTURER_ID	ITEM_NAME
108	Wristband Thermometer

RETURN VALUE: 60.6 cu ft Tank

MOD

Devuelve el resto de una división. Por ejemplo, MOD(8,5) devuelve 3.

Sintaxis

```
MOD( numeric_value, divisor )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de dato numérico. Los valores que desea dividir. Puede introducir cualquier expresión de transformación válida.
<i>divisor</i>	Obligatorio	El valor numérico por el que desea dividir. El divisor no puede ser 0.

Valor de retorno

Valor numérico del tipo de datos que se pasa a la función. El resto del valor numérico dividido por el divisor.

NULL si el valor pasado a la función es NULL.

Ejemplos

La expresión siguiente devuelve el módulo de los valores del puerto PRICE divididos por los valores del puerto QTY:

```
MOD( PRICE, QTY )
```

PRICE	QTY	RETURN VALUE
10.00	2	0
12.00	5	2
9.00	2	1
15.00	3	0
NULL	3	NULL

PRICE	QTY	RETURN VALUE
20.00	NULL	NULL
25.00	0	<i>Error. Integration Service does not write row.</i>

La última fila (25, 0) provocó un error porque no se puede dividir por 0. Para evitar la división por 0, puede crear una expresión similar a la siguiente que devuelve el módulo del precio dividido por la cantidad sólo si la cantidad no es 0. Si la cantidad es 0, la función devuelve NULL:

```
MOD( PRICE, IIF( QTY = 0, NULL, QTY ) )
```

PRICE	QTY	RETURN VALUE
10.00	2	0
12.00	5	2
9.00	2	1
15.00	3	0
NULL	3	NULL
20.00	NULL	NULL
25.00	0	NULL

La última fila (25, 0) produce un valor NULL en lugar de un error porque la función IIF reemplaza NULL con el 0 en el puerto QTY.

MOVINGAVG

Devuelve el promedio (fila a fila) de un conjunto especificado de filas. Opcionalmente, puede aplicar una condición para filtrar filas antes de calcular la media móvil.

Sintaxis

```
MOVINGAVG( numeric_value, rowset [, filter_condition] )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. El valor para el cual desea calcular una media móvil. Se puede especificar cualquier expresión de transformación válida.
<i>rowset</i>	Obligatorio	Debe ser un literal entero positivo mayor que 0. Define el conjunto de filas para el cual desea calcular la media móvil. Por ejemplo, si desea calcular una media móvil para una columna de datos, en grupos de cinco filas, puede escribir una expresión como: <code>MOVINGAVG (SALES, 5)</code> .
<i>filter_condition</i>	Opcional	Limita las filas en la búsqueda. La condición de filtro debe ser un valor numérico o dar un valor TRUE, FALSE o NULL. Se puede especificar cualquier expresión de transformación válida.

Valor de retorno

Valor numérico.

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

Nota: Si el valor devuelto es un decimal con una precisión superior a 15, puede habilitar el modo de alta precisión para asegurarse de obtener una precisión decimal de hasta 28 dígitos.

Nulos

MOVINGAVG omite los valores nulos cuando calcula la media móvil. Sin embargo, si todos los valores son NULL, la función devuelve NULL.

Ejemplo

La siguiente expresión devuelve el pedido promedio de un arnés estabilizador, basado en las primeras cinco filas del puerto Ventas y, a continuación, devuelve el promedio de las últimas cinco filas leídas:

```
MOVINGAVG ( SALES, 5 )
```

ROW_NO	SALES	RETURN VALUE
1	600	NULL
2	504	NULL
3	36	NULL
4	100	NULL
5	550	358
6	39	245.8
7	490	243

La función devuelve el promedio para un conjunto de cinco filas: 358 basado en las filas 1 a 5, 245,8 basado en las filas 2 a 6 y 243 basado en las filas 3 a 7.

MOVINGSUM

Devuelve la suma (fila a fila) de un conjunto especificado de filas.

Opcionalmente, puede aplicar una condición para filtrar filas antes de calcular la suma móvil.

Sintaxis

```
MOVINGSUM( numeric_value, rowset [, filter_condition] )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. El valor para el cual desea calcular una suma móvil. Se puede especificar cualquier expresión de transformación válida.
<i>rowset</i>	Obligatorio	Debe ser un literal entero positivo mayor que 0. Define el conjunto de filas para el cual desea calcular la suma móvil. Por ejemplo, si desea calcular una suma móvil para una columna de datos, en grupos de cinco filas, puede escribir una expresión como: <code>MOVINGSUM(SALES, 5)</code>
<i>filter_condition</i>	Opcional	Limita las filas en la búsqueda. La condición de filtro debe ser un valor numérico o dar un valor TRUE, FALSE o NULL. Se puede especificar cualquier expresión de transformación válida.

Valor de retorno

Valor numérico.

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

Nota: Si el valor devuelto es un decimal con una precisión superior a 15, puede habilitar el modo de alta precisión para asegurarse de obtener una precisión decimal de hasta 28 dígitos.

Nulos

MOVINGSUM omite los valores nulos cuando calcula la suma móvil. Sin embargo, si todos los valores son NULL, la función devuelve NULL.

Ejemplo

La siguiente expresión devuelve la suma de pedidos de un arnés estabilizador, basado en las primeras cinco filas del puerto Ventas y, a continuación, devuelve el promedio de las últimas cinco filas leídas:

```
MOVINGSUM( SALES, 5 )
```

ROW_NO	SALES	RETURN VALUE
1	600	NULL
2	504	NULL
3	36	NULL
4	100	NULL

ROW_NO	SALES	RETURN VALUE
5	550	1790
6	39	1229
7	490	1215

La función devuelve la suma para un conjunto de cinco filas: 1790 basado en las filas 1 a 5, 1229 basado en las filas 2 a 6 y 1215 basado en las filas 3 a 7.

NPER

Devuelve el número de períodos para una inversión basada en un tipo de interés constante y periódico, pagos constantes.

Sintaxis

`NPER(rate, present value, payment [, future value, type])`

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

Argumento	Obligatorio / Opcional	Descripción
<i>rate</i>	Obligatorio	Númérico. Tipo de interés ganado en cada período. Se expresa como número decimal. Divida el tipo por 100 para expresarlo como número decimal. Debe ser mayor o igual que 0.
<i>present_value</i>	Obligatorio	Númérico. Importe de la cantidad global que vale una serie de pagos futuros.
<i>payment</i>	Obligatorio	Númérico. Importe del pago debido por período. Debe ser un número negativo.
<i>future_value</i>	Opcional	Númérico. Balance monetario que se desea conseguir después de que se haya realizado el último pago. Si omite este valor, NPER utiliza 0.
<i>type</i>	Opcional	Booleano. Programación del pago. Introduzca 1 si el pago se efectúa al inicio del período. Introduzca 0 si el pago se efectúa al final del período. El valor predeterminado es 0. Si introduce un valor distinto de 0 o 1, el Servicio de integración de datos trata el valor como 1.

Valor de retorno

Númérico.

Ejemplo

El valor actual de una inversión es de 500 \$. Cada pago es de 2000 \$ y el valor futuro de la inversión es de 20 000 \$. La siguiente expresión devuelve 9 como el número de períodos para el que se necesita hacer los pagos:

`NPER (0.015, -500, -2000, 20000, TRUE)`

Notas

Para calcular el tipo de interés obtenido en cada período, divida el tipo de interés anual por el número de pagos efectuados en un año. Por ejemplo, si se realizan pagos mensuales a un tipo de interés anual del 15%, el valor del argumento Tipo de interés es 15% dividido entre 12. Si se realizan pagos anuales, el valor del argumento Tipo de interés es 15%.

El valor del pago y el valor actual son negativos porque estos son los importes que se pagan.

PERCENTILE

Calcula el valor que cae en un percentil dado en un grupo de números. Solo puede anidar otra función de agregado más dentro de PERCENTILE, y la función anidada debe devolver un tipo de dato numérico.

El Servicio de integración de datos lee todas las filas de datos para efectuar el cálculo del percentil. El proceso de lectura de filas para efectuar el cálculo puede tener efectos en el rendimiento. Puede optar por aplicar un filtro para limitar las filas que se deben leer en el cálculo del percentil.

Sintaxis

```
PERCENTILE( numeric_value, percentile [, filter_condition ] )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de dato numérico. Transfiere los valores para los que desea calcular un percentil. Puede especificar cualquier expresión de transformación válida.
<i>percentile</i>	Obligatorio	Entero comprendido entre 0 y 100, ambos inclusive. Transfiere el percentil que desea calcular. Puede especificar cualquier expresión de transformación válida. Si se transfiere un número fuera del intervalo entre 0 y 100, el Servicio de integración de datos muestra un error y no graba la fila.
<i>filter_condition</i>	Opcional	Limita las filas de la búsqueda. La condición de filtro debe ser un valor numérico o debe verificarse como TRUE, FALSE o NULL. Puede especificar cualquier expresión de transformación válida.

Valor de devolución

Valor numérico.

NULL si todos los valores transferidos a la función son NULL, o si no se ha seleccionado ninguna fila (por ejemplo, la condición de filtro se verifica como FALSE o NULL para todas las filas).

Nota: Si el valor devuelto es un decimal con una precisión superior a 15, puede habilitar el modo de alta precisión para asegurarse de obtener una precisión decimal de hasta 28 dígitos.

Nulos

Si un valor es NULL, PERCENTILE omite la fila. Sin embargo, si todos los valores de un grupo son NULL, PERCENTILE devuelve NULL.

Nota: De forma predeterminada, el Servicio de integración de datos trata los valores nulos como NULL en las funciones agregadas. Si pasa un puerto o grupo de valores nulos completo, la función devuelve NULL. Sin embargo, cuando se configura el Servicio de integración de datos, se puede indicar la forma en que se desea manejar los valores nulos en las funciones agregadas. En las funciones agregadas se pueden tratar los valores nulos como 0 o como NULL.

Agrupar por

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

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

Ejemplo:

El Servicio de integración de datos calcula un percentil con la siguiente lógica:

$$i = \frac{(x + 1) \times \text{percentile}}{100}$$

Utilice las siguientes directrices para esta ecuación:

- x es el número de elementos en el grupo de valores para los que está calculando un percentil.
- Si $i < 1$, PERCENTILE devuelve el valor del primer elemento de la lista.
- Si i es un valor entero, PERCENTILE devuelve el valor del *enésimo* elemento de la lista.
- De no ser así, PERCENTILE devuelve el valor de n :

$$n = \lceil i \rceil^{\text{th}} \text{element} \times (\lceil i \rceil - i) + \lfloor i \rfloor^{\text{th}} \text{element} \times (i - \lfloor i \rfloor)$$

La siguiente expresión devuelve el salario que se halla en el 75º percentil de los salarios superiores a 50.000 dólares:

```
PERCENTILE( SALARY, 75, SALARY > 50000 )
```

SALARY

125000.0

27900.0

100000.0

NULL

55000.0

9000.0

85000.0

86000.0

48000.0

SALARY

99000.0

RETURN VALUE: 106250.0

PMT

Devuelve el pago de un préstamo basado en los pagos constantes y un tipo de interés constante.

Sintaxis

`PMT(rate, terms, present value[, future value, type])`

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

Argumento	Obligatorio / Opcional	Descripción
<i>rate</i>	Obligatorio	Numérico. Tipo de interés del préstamo para cada período. Expresado como un número decimal. Divida el índice entre 100 para expresarlo como un número decimal. Debe ser mayor o igual a 0.
<i>terms</i>	Obligatorio	Numérico. Número de períodos o pagos. Debe ser mayor que 0.
<i>present value</i>	Obligatorio	Numérico. Principio para el préstamo.
<i>future value</i>	Opcional	Numérico. Saldo monetario que desea conseguir después del último pago. Si omite este valor, PMT utiliza 0.
<i>type</i>	Opcional	Booleano. Programación del pago. Introduzca 1 si el pago es al inicio del período. Introduzca 0 si el pago es al final del período. El valor predeterminado es 0. Si introduce un valor distinto de 0 ó 1, el Servicio de integración de datos trata el valor como 1.

Valor de retorno

Numérico.

Ejemplo

La expresión siguiente devuelve -2111,64 como importe de pago mensual de un préstamo.

`PMT(0.01, 10, 20000)`

Notas

Para calcular el tipo de los intereses devengados en cada período, se divide el tipo anual por el número de pagos efectuados en un año. Por ejemplo, si realiza pagos mensuales a un tipo de interés anual del 15%, el tipo será 15%/12. Si realiza pagos anuales, el tipo es del 15%.

El valor del pago es negativo porque se trata de cantidades que usted paga.

POWER

Devuelve un valor elevado al exponente que se pase a la función.

Sintaxis

```
POWER( base, exponent )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>base</i>	Obligatorio	Valor numérico. Este argumento es el valor de base. Puede introducir cualquier expresión de transformación válida. Si el valor de base es negativo, el exponente tiene que ser un número entero.
<i>exponent</i>	Obligatorio	Valor numérico. Este argumento es el valor del exponente. Puede introducir cualquier expresión de transformación válida. Si el valor de base es negativo, el exponente tiene que ser un número entero. En este caso, la función redondea los valores decimales al número entero más cercano antes de devolver un valor.

Valor de retorno

Valor doble.

NULL si pasa un valor nulo a la función.

Ejemplo

La expresión siguiente devuelve los valores del puerto Números elevados a los valores del puerto Exponente:

```
POWER( NUMBERS, EXPONENT )
```

NUMBERS	EXPONENT	RETURN VALUE
10.0	2.0	100
3.5	6.0	1838.265625
3.5	5.5	982.594307804838
NULL	2.0	NULL
10.0	NULL	NULL
-3.0	-6.0	0.00137174211248285
3.0	-6.0	0.00137174211248285
-3.0	6.0	729.0
-3.0	5.5	729.0

El valor -3,0 elevado a 6 devuelve los mismos resultados que -3,0 elevado a 5,5. Si la base es negativa, el exponente tiene que ser un número entero. De lo contrario, el Servicio de integración de datos redondea el exponente al valor entero más cercano.

PV

Devuelve el valor actual de una inversión.

Sintaxis

```
PV( rate, terms, payment [, future value, type] )
```

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

Argumento	Obligatorio / Opcional	Descripción
rate	Obligatorio	Númerico. Tipo de interés ganado en cada período Se expresa como un número decimal. Divida el índice entre 100 para expresarlo como un número decimal. Debe ser mayor o igual a 0.
terms	Obligatorio	Númerico. Número de períodos o pagos. Debe ser mayor que 0.
payments	Obligatorio	Númerico. Importe del pago debido por período. Debe ser un número negativo.
future value	Opcional	Númerico. Balance monetario después del último pago. Si omite este valor, PV utiliza 0.
types	Opcional	Booleano. Programación del pago. Introduzca 1 si el pago es al inicio del período. Introduzca 0 si el pago es al final del período. El valor predeterminado es 0. Si introduce un valor distinto de 0 o 1, el Servicio de integración de datos trata el valor como 1.

Valor de retorno

Númerico.

Ejemplo

La expresión siguiente devuelve 12.524,43 como importe que debe depositar en la cuenta hoy para tener un valor futuro de 20.000 \$ en un año si también deposita 500 \$ al comienzo de cada período:

```
PV( 0.0075, 12, -500, 20000, TRUE )
```

RAND

Devuelve un número aleatorio entre 0 y 1. Esto es útil en los escenarios de probabilidad.

Sintaxis

```
RAND( seed )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>seed</i>	Opcional	Numérico. Valor inicial con el que el servicio de integración genera el número aleatorio. El valor debe ser una constante. Si no especifica ningún valor de inicialización, el Servicio de integración de datos usa la hora del sistema actual para obtener el número de segundos desde el 1 de enero de 1971. Este valor se usa como valor de inicialización.

Valor devuelto

Numérico.

Para el mismo valor de inicialización, el Servicio de integración de datos genera la misma secuencia de números.

Ejemplo

La siguiente expresión puede devolver un valor de 0,417022004702574:

```
RAND (1)
```

RATE

Devuelve el tipo de interés obtenido por período por una garantía.

Sintaxis

```
RATE( terms, payment, present value[, future value, type] )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>terms</i>	Obligatorio	Numérico. Número de períodos o pagos. Debe ser mayor que 0.
<i>payments</i>	Obligatorio	Numérico. Importe del pago debido por período. Debe ser un número negativo.
<i>present value</i>	Obligatorio	Numérico. Importe de la cantidad global al que equivale actualmente una serie de pagos futuros.
<i>future value</i>	Opcional	Numérico. Saldo monetario que desea conseguir después del último pago. Por ejemplo, el valor futuro de un préstamo es 0. Si omite este argumento, RATE utiliza 0.
<i>types</i>	Opcional	Booleano. Programación del pago. Introduzca 1 si el pago es al inicio del período. Introduzca 0 si el pago es al final del período. El valor predeterminado es 0. Si introduce un valor distinto de 0 ó 1, el Servicio de integración de datos trata el valor como 1.

Valor de retorno

Numérico.

Ejemplo

La expresión siguiente devuelve 0,0077 como tipo de interés mensual de un préstamo.

```
RATE( 48, -500, 20000 )
```

Para calcular el tipo de interés anual del préstamo, multiplique 0,0077 por 12. El tipo de interés anual es 0,0924 o 9,24%.

REG_EXTRACT

Extrae subpatrones de una expresión regular en un valor de entrada. Puede extraer, por ejemplo, de un patrón de expresión regular de un nombre completo, el nombre o el apellido.

Nota: Utilice la función REG_REPLACE para reemplazar un patrón de caracteres en una cadena con otro patrón de caracteres.

Sintaxis

```
REG_EXTRACT( subject, 'pattern', subPatternNum, match_from_start )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>subject</i>	Obligatorio	Tipo de datos String. Transfiere el valor que desea comparar con el patrón de la expresión regular.
<i>pattern</i>	Obligatorio	Tipo de datos String. Patrón de expresión regular con el que se desea coincidir. Debe utilizar una sintaxis de expresiones regulares compatible con perl. Encierre el patrón entre comillas sencillas. Encierre los subpatrones entre paréntesis.
<i>subPatternNum</i>	Opcional	Valor entero. Número del subpatrón de la expresión regular con el que desea coincidir. Utilice las siguientes directrices para determinar el número del subpatrón: <ul style="list-style-type: none">- ningún valor o 1. Extrae el primer subpatrón de la expresión regular.- 2. Extrae el segundo subpatrón de la expresión regular.- n. Extrae el <i>enésimo</i> subpatrón de la expresión regular. El valor predeterminado es 1.
<i>match_from_start</i>	Opcional	Valor numérico. Devuelve la subcadena si se encuentra una coincidencia desde el inicio de la cadena. Utilice las siguientes directrices para determinar la coincidencia desde el valor inicial: <ul style="list-style-type: none">- 0. Hace coincidir el patrón con la cadena subject desde el índice inicial o cualquier índice.- Distinto de cero. Hace coincidir el patrón con la cadena subject desde el índice inicial.

Uso de la sintaxis de expresión regular compatible con perl

Debe utilizar la sintaxis de expresión regular compatible con perl con las funciones REG_EXTRACT, REG_MATCH y REG_REPLACE.

La siguiente tabla ofrece las directrices de la sintaxis de expresión regular compatible con perl.

Sintaxis	Descripción
.	Coincide con cualquier carácter.
[a-z]	Coincide con una instancia de un carácter en minúsculas. Por ejemplo, [a-z] coincide con ab. Utilice [A-Z] para que coincidan caracteres en mayúsculas.
\d	Coincide con una instancia de cualquier dígito de 0-9.
\s	Coincide con un carácter de espacio en blanco.
\w	Coincide con un carácter alfanumérico, incluido el subrayado (_)
()	Agrupar una expresión. Por ejemplo, los paréntesis en (\d-\d-\d\d) agrupan la expresión \d\d-\d\d, que busca todos los grupos de dos números seguidos de un guión y dos números más, como 12-34.
{}	Coincide con el número de caracteres. Por ejemplo, \d{3} coincide con tres números, como 650 ó 510. O, [a-z]{2} coincide con dos letras, como CA o NY.
?	Coincide con el carácter precedente o con un grupo de caracteres ninguna o una vez. Por ejemplo, \d{3}(-\d{4})? coincide con tres números que pueden ir seguidos de un guión y otros cuatro números.
*	Coincide con ninguna o más instancias de los valores que siguen al asterisco. Por ejemplo, *0 es un valor que precede a 0.
+	Coincide con una o más instancias de los valores que siguen al signo más. Por ejemplo, \w+ es cualquier valor que sigue a un carácter alfanumérico.

Por ejemplo, la siguiente expresión regular busca códigos postales de EE. UU. de cinco dígitos, como 93930 y códigos postales de 9 dígitos, como 93930-5407:

```
\d{5}(-\d{4})?
```

\d{5} hace referencia a cinco números, como 93930. Los paréntesis que rodean a -\d{4} agrupan este segmento de la expresión. El guión representa el guión de un código postal de 9 dígitos, como 93930-5407. \d{4} hace referencia a cuatro números, como 5407. El signo de interrogación indica que el guión y los últimos cuatro dígitos son opcionales o que pueden aparecer una vez.

Conversión de la sintaxis de COBOL a la sintaxis de expresión regular compatible con perl.

Si está familiarizado con la sintaxis de COBOL, puede utilizar la siguiente información para escribir expresiones regulares compatibles con perl.

La siguiente tabla muestra ejemplos de la sintaxis de COBOL y de sus equivalentes en perl:

Sintaxis de COBOL	Sintaxis de perl	Descripción
9	\d	Coincide con una instancia de cualquier dígito de 0-9.
9999	\d\d\d\d o \d{4}	Coincide con cuatro dígitos entre 0-9, como 1234 ó 5936.
x	[a-z]	Coincide con una instancia de una letra.
9xx9	\d[a-z][a-z]\d	Coincide con cualquier número seguido por dos letras y otro número, como 1ab2.

Conversión de la sintaxis de SQL a la sintaxis de expresión regular compatible con perl

Si está familiarizado con la sintaxis de SQL, puede utilizar la siguiente información para escribir expresiones regulares compatibles con perl.

La siguiente tabla muestra algunos ejemplos de la sintaxis de SQL y de sus equivalentes en perl:

Sintaxis de SQL	Sintaxis de perl	Descripción
%	. *	Coincide con cualquier cadena.
A%	A. *	Coincide con la letra "A" seguida de cualquier cadena, como Area (en inglés).
_	. (un punto)	Coincide con cualquier carácter.
A_	A.	Coincide con "A" seguida de cualquier carácter, como AZ.

Valor de devolución

Devuelve el valor del *enésimo* subpatrón que forma parte del valor de entrada. El *enésimo* subpatrón se basa en el valor que especifique para subPatternNum.

NULL si la entrada es un valor nulo o si el patrón es nulo.

Ejemplo

Puede utilizar REG_EXTRACT en una expresión para extraer de una expresión regular segundos nombres que coincidan con el nombre, el segundo nombre y el apellido. Por ejemplo, la siguiente expresión devuelve el segundo nombre de una expresión regular:

```
REG_EXTRACT( Employee_Name, '(\w+)\s+(\w+)\s+(\w+)', 2)
```

Employee_Name	Return Value
Stephen Graham Smith	Graham
Juan Carlos Fernando	Carlos

REG_MATCH

Devuelve si un valor coincide con un patrón de expresión regular. Permite validar patrones de datos como números de identificación, números de teléfono, códigos postales y nombres de estado.

Nota: Utilice la función REG_REPLACE para sustituir un patrón de caracteres en una cadena con un nuevo patrón de caracteres.

Sintaxis

```
REG_MATCH( subject, pattern )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>subject</i>	Obligatorio	Tipo de datos String. Pasa el valor que se desea hacer coincidir respecto al patrón de la expresión regular.
<i>pattern</i>	Obligatorio	Tipo de datos String. Patrón de expresión regular con el que se desea coincidir. Debe utilizar una sintaxis de expresiones regulares compatible con perl. Delimite el patrón mediante comillas simples. Para más información, véase "REG_EXTRACT" en la página 158 .

Valor de retorno

TRUE si los datos coinciden con el patrón.

FALSE si los datos no coinciden con el patrón.

NULL si la entrada es un valor nulo o si el patrón es NULL.

Ejemplo

Puede usar REG_MATCH en una expresión para validar números de teléfono. Por ejemplo, la siguiente expresión hace coincidir un número de teléfono de 10 dígitos respecto al patrón y devuelve un valor booleano basado en la coincidencia:

```
REG_MATCH (Phone_Number, '(\d\d\d-\d\d\d-\d\d\d\d) ' )
```

Phone_Number	Return Value
408-555-1212	TRUE
	NULL
510-555-1212	TRUE
92 555 51212	FALSE
650-555-1212	TRUE
415-555-1212	TRUE
831 555 12123	FALSE

Consejo

También puede usar REG_MATCH para las siguientes tareas:

- Para verificar que un valor coincide con un patrón. Se utiliza de forma similar a la función SQL LIKE.
- Para verificar que los valores son caracteres. Se utiliza de forma similar a la función SQL IS_CHAR.

Para verificar que un valor coincide con un patrón, utilice un punto (.) y un asterisco (*) con la función REG_MATCH en una expresión. Un punto coincide con un carácter cualquiera. Un asterisco coincide con 0 o más instancias de valores que le suceden.

Por ejemplo, utilice la siguiente expresión para encontrar números de cuenta que empiezan por 1835:

```
REG_MATCH (ACCOUNT_NUMBER, '1835.*')
```

Para verificar que los valores son caracteres, utilice una función REG_MATCH con la expresión regular [a-zA-Z]+. a-z hace coincidir todos los caracteres en minúscula. A-Z hace coincidir todos los caracteres en mayúscula. El signo más (+) indica que debe haber un carácter como mínimo.

Por ejemplo, utilice la siguiente expresión para comprobar que una lista de apellidos solamente contiene caracteres.

```
REG_MATCH (LAST_NAME, '[a-zA-Z]+')
```

REG_REPLACE

Reemplaza caracteres en una cadena con otro patrón de caracteres. De forma predeterminada, REG_REPLACE busca en la cadena de entrada el patrón de caracteres que se haya especificado y reemplaza todas las apariciones con el patrón de reemplazo. También puede indicar el número de apariciones del patrón que desea reemplazar en la cadena.

Sintaxis

```
REG_REPLACE( subject, pattern, replace, numReplacements )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>subject</i>	Obligatorio	Tipo de datos String. Pasa la cadena que desea buscar.
<i>pattern</i>	Obligatorio	Tipo de datos String. Pasa la cadena de carácter que se quiere reemplazar. Debe utilizar la sintaxis de expresión regular compatible con Perl. Incluya el patrón entre comillas simples. Para obtener más información, consulte “REG_EXTRACT” en la página 158 .
<i>replace</i>	Obligatorio	Tipo de datos String. Pasa la cadena del nuevo carácter.
<i>numReplacements</i>	Opcional	Tipo de datos numérico. Especifica el número de apariciones que desea reemplazar. Si omite esta opción, REG_REPLACE reemplazará todas las apariciones de la cadena de caracteres.

Valor de retorno

Cadena

Ejemplo

La expresión siguiente elimina espacios adicionales de los datos del nombre del empleado para cada fila del puerto `Employee_name`:

```
REG_REPLACE( Employee_Name, '\s+', ' ' )
```

Employee_Name	RETURN VALUE
Adam Smith	Adam Smith
Greg Sanders	Greg Sanders
Sarah Fe	Sarah Fe
Sam Cooper	Sam Cooper

REPLACECHR

Sustituye caracteres de una cadena por un carácter único o por ningún carácter. REPLACECHR busca en la cadena de entrada los caracteres especificados y sustituye todas las ocurrencias de todos los caracteres con el nuevo carácter especificado.

Sintaxis

```
REPLACECHR( CaseFlag, InputString, OldCharSet, NewChar )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>CaseFlag</i>	Obligatorio	El valor debe ser un entero. Determina si en los argumentos de esta función se distingue entre mayúsculas y minúsculas. Se puede especificar cualquier expresión de transformación válida. Cuando <i>CaseFlag</i> es un número distinto de 0, la función distingue entre mayúsculas y minúsculas. Cuando <i>CaseFlag</i> es un valor nulo o es 0, la función no distingue entre mayúsculas y minúsculas.
<i>InputString</i>	Obligatorio	Debe ser una cadena de caracteres. Pasa la cadena que se desea buscar. Se puede especificar cualquier expresión de transformación válida. Si se pasa un valor numérico, la función la convierte en una cadena de caracteres. Si <i>InputString</i> es NULL, REPLACECHR devuelve NULL.

Argumento	Obligatorio / Opcional	Descripción
<i>OldCharSet</i>	Obligatorio	Debe ser una cadena de caracteres. Los caracteres que se desea sustituir. Se puede introducir uno o más caracteres. Se puede especificar cualquier expresión de transformación válida. También puede especificar un literal de texto encerrado entre comillas simples, por ejemplo, 'abc'. Si se pasa un valor numérico, la función la convierte en una cadena de caracteres. Si <i>OldCharSet</i> es NULL o está vacío, REPLACECHR devuelve <i>InputString</i> .
<i>NewChar</i>	Obligatorio	Debe ser una cadena de caracteres. Puede introducir un carácter, una cadena vacía o NULL. Se puede especificar cualquier expresión de transformación válida. Si <i>NewChar</i> es NULL o está vacío, REPLACECHR elimina todas las ocurrencias de todos los caracteres de <i>OldCharSet</i> en <i>InputString</i> . Si <i>NewChar</i> contiene más de un carácter, REPLACECHR utiliza el primer carácter para sustituir <i>OldCharSet</i> .

Valor de devolución

Cadena.

Cadena vacía si REPLACECHR elimina todos los caracteres de *InputString*.

NULL si *InputString* es NULL.

InputString si *OldCharSet* es NULL o está vacío.

Ejemplos

La siguiente expresión elimina las comillas dobles de los datos de registro de web para cada fila en el puerto WEBLOG:

```
REPLACECHR( 0, WEBLOG, '"', NULL )
```

WEBLOG	RETURN VALUE
"GET /news/index.html HTTP/1.1"	GET /news/index.html HTTP/1.1
"GET /companyinfo/index.html HTTP/1.1"	GET /companyinfo/index.html HTTP/1.1
GET /companyinfo/index.html HTTP/1.1	GET /companyinfo/index.html HTTP/1.1
NULL	NULL

La siguiente expresión elimina múltiples caracteres para cada fila en el puerto WEBLOG:

```
REPLACECHR ( 1, WEBLOG, ']['', NULL )
```

WEBLOG	RETURN VALUE
[29/Oct/2001:14:13:50 -0700]	29/Oct/2001:14:13:50 -0700
[31/Oct/2000:19:45:46 -0700] "GET /news/index.html HTTP/1.1"	31/Oct/2000:19:45:46 -0700 GET /news/index.html HTTP/1.1

WEBLOG	RETURN VALUE
[01/Nov/2000:10:51:31 -0700] "GET /news/index.html HTTP/1.1"	01/Nov/2000:10:51:31 -0700 GET /news/index.html HTTP/1.1
NULL	NULL

La siguiente expresión cambia parte del valor del código de cliente para cada fila en el puerto CUSTOMER_CODE:

```
REPLACECHR ( 1, CUSTOMER_CODE, 'A', 'M' )
```

CUSTOMER_CODE	RETURN VALUE
ABA	MBM
abA	abM
BBC	BBC
ACC	MCC
NULL	NULL

La siguiente expresión cambia parte del valor del código de cliente para cada fila en el puerto CUSTOMER_CODE:

```
REPLACECHR ( 0, CUSTOMER_CODE, 'A', 'M' )
```

CUSTOMER_CODE	RETURN VALUE
ABA	MBM
abA	MbM
BBC	BBC
ACC	MCC

La siguiente expresión cambia parte del valor del código de cliente para cada fila en el puerto CUSTOMER_CODE:

```
REPLACECHR ( 1, CUSTOMER_CODE, 'A', NULL )
```

CUSTOMER_CODE	RETURN VALUE
ABA	B
BBC	BBC
ACC	CC
AAA	[empty string]

CUSTOMER_CODE	RETURN VALUE
aaa	aaa
NULL	NULL

La siguiente expresión elimina múltiples caracteres para cada fila en el puerto INPUT:

```
REPLACECHR ( 1, INPUT, '14', NULL )
```

INPUT	RETURN VALUE
12345	235
4141	NULL
111115	5
NULL	NULL

Cuando se desea usar una comilla simple (') en *OldCharSet* o *NewChar*, debe utilizarse la función CHR. La comilla simple es el único carácter que no se puede usar dentro de un literal de cadena.

La siguiente expresión elimina múltiples caracteres, incluidas las comillas simples, para cada fila en el puerto INPUT:

```
REPLACECHR (1, INPUT, CHR(39), NULL )
```

INPUT	RETURN VALUE
'Tom Smith' 'Laura Jones'	Tom Smith Laura Jones
Tom's	Toms
NULL	NULL

REPLACESTR

Sustituye caracteres de una cadena por un carácter único, por múltiples caracteres o por ningún carácter. REPLACESTR busca la cadena de entrada en todas las cadenas que indique y la sustituye por la nueva cadena que especifique.

Sintaxis

```
REPLACESTR ( CaseFlag, InputString, OldString1, [OldString2, ... OldStringN,] NewString )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>CaseFlag</i>	Obligatorio	<p>Debe ser un entero. Determina si en los argumentos de esta función se distingue entre mayúsculas y minúsculas. Puede especificar cualquier expresión de transformación válida.</p> <p>Cuando <i>CaseFlag</i> es un número distinto de 0, la función distingue entre mayúsculas y minúsculas.</p> <p>Cuando <i>CaseFlag</i> es un valor nulo o 0, la función no distingue entre mayúsculas y minúsculas.</p>
<i>InputString</i>	Obligatorio	<p>Debe ser una cadena de caracteres. Transfiere las cadenas que desea buscar. Puede especificar cualquier expresión de transformación válida. Si transfiere un valor numérico, la función lo convierte en una cadena de caracteres.</p> <p>Si <i>InputString</i> es NULL, REPLACESTR devuelve NULL.</p>
<i>OldString</i>	Obligatorio	<p>Debe ser una cadena de caracteres. La cadena que desea sustituir. Debe especificar como mínimo, un argumento <i>OldString</i>. Para cada argumento <i>OldString</i>, puede especificar uno o más caracteres. Puede especificar cualquier expresión de transformación válida. También puede especificar un literal de texto encerrado entre comillas simples, por ejemplo, 'abc'.</p> <p>Si transfiere un valor numérico, la función lo convierte en una cadena de caracteres.</p> <p>Cuando REPLACESTR contiene varios argumentos <i>OldString</i> y uno o más argumentos <i>OldString</i> son NULL o están vacíos, REPLACESTR no considera el argumento <i>OldString</i>. Cuando todos los argumentos <i>OldString</i> son NULL o están vacíos, REPLACESTR devuelve <i>InputString</i>.</p> <p>La función sustituye los caracteres de los argumentos <i>OldString</i> en el orden en el que aparecen en la función. Por ejemplo, si sustituye varios argumentos <i>OldString</i>, el primer argumento <i>OldString</i> tiene prioridad sobre el segundo argumento <i>OldString</i>, y el segundo argumento <i>OldString</i> tiene prioridad sobre el tercer argumento <i>OldString</i>. Cuando REPLACESTR sustituye una cadena, sitúa el cursor tras los caracteres sustituidos en <i>InputString</i> antes de buscar la siguiente coincidencia.</p>
<i>NewString</i>	Obligatorio	<p>Debe ser una cadena de caracteres. Puede especificar un carácter, varios caracteres, una cadena vacía o NULL. Puede especificar cualquier expresión de transformación válida.</p> <p>Si <i>NewString</i> es NULL o está vacía, REPLACESTR quita todas las ocurrencias de <i>OldString</i> en <i>InputString</i>.</p>

Valor de devolución

Cadena.

Cadena vacía si REPLACESTR quita todos los caracteres de *InputString*.

NULL si *InputString* es NULL.

InputString si todos los argumentos *OldString* son NULL o están vacíos.

Ejemplos

La siguiente expresión quita las comillas dobles y dos cadenas de texto diferentes de los datos del registro web de cada fila del puerto WEBLOG:

```
REPLACESTR ( 1, WEBLOG, '"', 'GET ', ' HTTP/1.1', NULL )
```

WEBLOG	RETURN VALUE
"GET /news/index.html HTTP/1.1"	/news/index.html
"GET /companyinfo/index.html HTTP/1.1"	/companyinfo/index.html
GET /companyinfo/index.html	/companyinfo/index.html
GET	[empty string]
NULL	NULL

La siguiente expresión cambia el título para determinados valores de cada fila del puerto TITLE:

```
REPLACESTR ( 1, TITLE, 'rs.', 'iss', 's.' )
```

TITLE	RETURN VALUE
Mrs.	Ms.
Miss	Ms.
Mr.	Mr.
MRS.	MRS.

La siguiente expresión cambia el título para determinados valores de cada fila del puerto TITLE:

```
REPLACESTR ( 0, TITLE, 'rs.', 'iss', 's.' )
```

TITLE	RETURN VALUE
Mrs.	Ms.
MRS.	Ms.

La siguiente expresión muestra cómo la función REPLACESTR sustituye varios argumentos `OldString` de cada fila en el puerto INPUT:

```
REPLACESTR ( 1, INPUT, 'ab', 'bc', '*' )
```

INPUT	RETURN VALUE
abc	*c
abbc	**
abbbbc	*bb*
bc	*

La siguiente expresión muestra cómo la función REPLACESTR sustituye varios argumentos *OldString* de cada fila en el puerto INPUT:

```
REPLACESTR ( 1, INPUT, 'ab', 'bc', 'b' )
```

INPUT	RETURN VALUE
ab	b
bc	b
abc	bc
abbc	bb
abbcc	bbc

Cuando quiera utilizar comillas simples (') en *OldString* o *NewString*, debe utilizar la función CHR. Utilice las funciones CHR y CONCAT para concatenar unas comillas simples en una cadena. Las comillas simples son el único carácter que no se puede utilizar dentro de un literal de cadena. Observe el siguiente ejemplo:

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

El valor de devolución es:

```
Joan's car
```

La siguiente expresión cambia una cadena que incluye unas comillas simples de las filas del puerto INPUT:

```
REPLACESTR ( 1, INPUT, CONCAT('it', CONCAT(CHR(39), 's' )), 'its' )
```

INPUT	RETURN VALUE
it's	its
mit's	mits
mits	mits
mits'	mits'

REVERSE

Invierte la cadena de entrada.

Sintaxis

```
REVERSE( string )
```

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

Argumento	Obligatorio/Opcional	Descripción
<i>string</i>	Obligatorio	Cualquier valor de carácter. Es el valor que se va a invertir.

Valor devuelto

Cadena. Inversión del valor de entrada.

Ejemplo

La siguiente expresión invierte los números del código de cliente:

```
REVERSE ( CUSTOMER_CODE )
```

CUSTOMER_CODE	RETURN VALUE
0001	1000
0002	2000
0003	3000
0004	4000

ROUND (Fechas)

Redondea una parte de una fecha. También puede utilizar ROUND para redondear números.

Esta función puede redondear las siguientes partes de una fecha:

Año

Redondea la parte del año de una fecha en función del mes.

Mes

Redondea la parte del mes de una fecha en función del día del mes.

Día

Redondea la parte del día de una fecha en función de la hora.

Hora

Redondea la parte de la hora de una fecha en función de los minutos de una hora.

Minuto

Redondea la parte del minuto de una fecha en función de los segundos.

Segundo

Redondea la parte del segundo de una fecha en función de los milisegundos.

Milisegundo

Redondea la parte del milisegundo de una fecha en función de los microsegundos.

Microsegundo

Redondea la parte del microsegundo de una fecha en función de los nanosegundos.

La siguiente tabla muestra las condiciones que utiliza la expresión ROUND y los valores de devolución:

Condición	Expresión	Valor de devolución
Un mes entre enero y junio, la función devuelve 1 de enero del mismo año y establece la hora en 00:00:00.000000000.	ROUND(TO_DATE('04/16/1998 8:24:19', 'MM/DD/YYYY HH24:MI:SS'), 'YY')	01/01/1998 00:00:00.000000000
Un mes entre julio y diciembre, la función devuelve 1 de enero del año siguiente y establece la hora en 00:00:00.000000000.	ROUND(TO_DATE('07/30/1998 2:30:55', 'MM/DD/YYYY HH24:MI:SS'), 'YY')	01/01/1999 00:00:00.000000000
Un día del mes entre 1 y 15, la función devuelve el primer día del mes de entrada y establece la hora en 00:00:00.000000000.	ROUND(TO_DATE('04/15/1998 8:24:19', 'MM/DD/YYYY HH24:MI:SS'), 'MM')	04/01/1998 00:00:00.000000000
Un día del mes entre el 16 y el último día del mes, la función devuelve el primer día del mes siguiente y establece la hora en 00:00:00.000000000.	ROUND(TO_DATE('05/22/1998 10:15:29', 'MM/DD/YYYY HH24:MI:SS'), 'MM')	06/01/1998 00:00:00.000000000
La hora entre 00:00:00 (12 a.m.) y 11:59:59 a.m., la función devuelve la fecha actual y establece la hora en 00:00:00.000000000 (12 a.m.).	ROUND(TO_DATE('06/13/1998 2:30:45', 'MM/DD/YYYY HH24:MI:SS'), 'DD')	06/13/1998 00:00:00.000000000
La hora 12:00:00 (12 p.m.) o más tarde, la función redondea la fecha hasta el día siguiente y establece la hora en 00:00:00.000000000 (12 a.m.).	ROUND(TO_DATE('06/13/1998 22:30:45', 'MM/DD/YYYY HH24:MI:SS'), 'DD')	06/14/1998 00:00:00.000000000
La parte de minuto de la hora entre 0 y 29 minutos, la función devuelve la hora actual y establece los minutos, los segundos, los milisegundos y los nanosegundos en 0.	ROUND(TO_DATE('04/01/1998 11:29:35', 'MM/DD/YYYY HH24:MI:SS'), 'HH')	04/01/1998 11:00:00.000000000
La parte de minuto de la hora entre 30 o más, la función devuelve la hora siguiente y establece los minutos, los segundos, los milisegundos y los nanosegundos en 0.	ROUND(TO_DATE('04/01/1998 13:39:00', 'MM/DD/YYYY HH24:MI:SS'), 'HH')	04/01/1998 14:00:00.000000000
La hora entre 0 y 29 segundos, la función devuelve el minuto actual y establece los segundos, los milisegundos y los nanosegundos en 0.	ROUND(TO_DATE('05/22/1998 10:15:29', 'MM/DD/YYYY HH24:MI:SS'), 'MI')	05/22/1998 10:15:00.000000000
La hora entre 30 y 59 segundos, la función devuelve el minuto siguiente y establece los segundos, los milisegundos, y los nanosegundos en 0.	ROUND(TO_DATE('05/22/1998 10:15:30', 'MM/DD/YYYY HH24:MI:SS'), 'MI')	05/22/1998 10:16:00.000000000
La hora entre los 0 y 499 milisegundos, la función devuelve el segundo actual y establece los milisegundos en 0.	ROUND(TO_DATE('05/22/1998 10:15:29.499', 'MM/DD/YYYY HH24:MI:SS.MS'), 'SS')	05/22/1998 10:15:29.000000000
La hora entre los 500 y 999 milisegundos, la función devuelve el segundo siguiente y establece los milisegundos en 0.	ROUND(TO_DATE('05/22/1998 10:15:29.500', 'MM/DD/YYYY HH24:MI:SS.MS'), 'SS')	05/22/1998 10:15:30.000000000

Condición	Expresión	Valor de devolución
La hora entre los 0 y 499 microsegundos, la función devuelve el milisegundo actual y establece los microsegundos en 0.	ROUND(TO_DATE('05/22/1998 10:15:29.498125','MM/DD/YYYY HH24:MI:SS.US'),'MS')	05/22/1998 10:15:29.498000000
La hora entre los 500 y 999 microsegundos, la función devuelve el milisegundo siguiente y establece los microsegundos en 0.	ROUND(TO_DATE('05/22/1998 10:15:29.498785','MM/DD/YYYY HH24:MI:SS.US'),'MS')	05/22/1998 10:15:29.499000000
La hora entre los 0 y 499 nanosegundos, la función devuelve el microsegundo actual y establece los nanosegundos en 0.	ROUND(TO_DATE('05/22/1998 10:15:29.498125345','MM/DD/YYYY HH24:MI:SS.NS'),'US')	05/22/1998 10:15:29.498125000
La hora entre los 500 y 999 nanosegundos, la función devuelve el microsegundo siguiente y establece los nanosegundos en 0.	ROUND(TO_DATE('05/22/1998 10:15:29.498125876','MM/DD/YYYY HH24:MI:SS.NS'),'US')	05/22/1998 10:15:29.498126000

Sintaxis

```
ROUND( date [,format] )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>date</i>	Obligatorio	Tipo de dato de fecha y hora. Puede anidar TO_DATE para convertir cadenas en fechas antes de redondear.
<i>format</i>	Opcional	Especifique una cadena de formato válida. Es la parte de la fecha que desea redondear. Solamente se puede redondear una parte de la fecha. Si omite la cadena de formato, la función redondea la fecha a la fecha más cercana.

Valor de devolución

Fecha con la parte especificada redondeada. ROUND devuelve una fecha en el mismo formato que la fecha de origen. Puede vincular los resultados de esta función a cualquier puerto con un tipo de datos de fecha y hora.

NULL si transfiere un valor nulo a la función.

Ejemplos

Las siguientes expresiones redondean la parte del año de las fechas del puerto DATE_SHIPPED:

```
ROUND( DATE_SHIPPED, 'Y' )
ROUND( DATE_SHIPPED, 'YY' )
ROUND( DATE_SHIPPED, 'YYY' )
ROUND( DATE_SHIPPED, 'YYYY' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 1 1998 12:00:00.000000000AM

DATE_SHIPPED	RETURN VALUE
Apr 19 1998 1:31:20PM	Jan 1 1998 12:00:00.000000000AM
Dec 20 1998 3:29:55PM	Jan 1 1999 12:00:00.000000000AM
NULL	NULL

Las siguientes expresiones redondean la parte del mes de las fechas del puerto DATE_SHIPPED:

```
ROUND( DATE_SHIPPED, 'MM' )
ROUND( DATE_SHIPPED, 'MON' )
ROUND( DATE_SHIPPED, 'MONTH' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 1 1998 12:00:00.000000000AM
Apr 19 1998 1:31:20PM	May 1 1998 12:00:00.000000000AM
Dec 20 1998 3:29:55PM	Jan 1 1999 12:00:00.000000000AM
NULL	NULL

Las siguientes expresiones redondean la parte del día de las fechas del puerto DATE_SHIPPED:

```
ROUND( DATE_SHIPPED, 'D' )
ROUND( DATE_SHIPPED, 'DD' )
ROUND( DATE_SHIPPED, 'DDD' )
ROUND( DATE_SHIPPED, 'DY' )
ROUND( DATE_SHIPPED, 'DAY' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 15 1998 12:00:00.000000000AM
Apr 19 1998 1:31:20PM	Apr 20 1998 12:00:00.000000000AM
Dec 20 1998 3:29:55PM	Dec 21 1998 12:00:00.000000000AM
Dec 31 1998 11:59:59PM	Jan 1 1999 12:00:00.000000000AM
NULL	NULL

Las siguientes expresiones redondean la parte de la hora de las fechas del puerto DATE_SHIPPED:

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

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:31AM	Jan 15 1998 2:00:00.000000000AM
Apr 19 1998 1:31:20PM	Apr 19 1998 2:00:00.000000000PM
Dec 20 1998 3:29:55PM	Dec 20 1998 3:00:00.000000000PM

DATE_SHIPPED	RETURN VALUE
Dec 31 1998 11:59:59PM	Jan 1 1999 12:00:00.000000000AM
NULL	NULL

Las siguientes expresiones redondean la parte del minuto de las fechas del puerto DATE_SHIPPED:

```
ROUND( DATE_SHIPPED, 'MI' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 15 1998 2:11:00.000000000AM
Apr 19 1998 1:31:20PM	Apr 19 1998 1:31:00.000000000PM
Dec 20 1998 3:29:55PM	Dec 20 1998 3:30:00.000000000PM
Dec 31 1998 11:59:59PM	Jan 1 1999 12:00:00.000000000AM
NULL	NULL

ROUND (Números)

Redondea números hasta un número especificado de dígitos o decimales. También se puede usar ROUND para redondear fechas.

Sintaxis

```
ROUND( numeric_value [, precision] )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. Se puede especificar cualquier expresión de transformación válida. Utilice operadores para realizar operaciones aritméticas antes de redondear los valores.
<i>precision</i>	Opcional	<p>Entero positivo o negativo. Si introduce un valor de <i>precision</i> positivo, la función redondea a este número de posiciones decimales. Por ejemplo, ROUND(12.99, 1) devuelve 13.0 y ROUND(15.44, 1) devuelve 15.4.</p> <p>Si se introduce un valor de <i>precision</i> negativo, la función redondea este número a la izquierda del punto decimal, devolviendo un entero. Por ejemplo, ROUND(12.99, -1) devuelve 10 y ROUND(15.99, -1) devuelve 20.</p> <p>Si introduce un valor de <i>precision</i> decimal, la función redondea al entero más próximo antes de evaluar la expresión. Por ejemplo, ROUND(12.99, 0.8) devuelve 13.0 porque la función redondea 0.8 a 1 y luego evalúa la expresión.</p> <p>Si omite el argumento <i>precision</i>, la función redondea al entero más próximo, truncando la porción decimal del número. Por ejemplo, ROUND(12.99) devuelve 13.</p>

Valor de devolución

Valor numérico.

Si uno de los argumentos es NULL, ROUND devuelve NULL.

Nota: Si el valor devuelto es un decimal con una precisión superior a 15, puede habilitar el modo de alta precisión para asegurarse de obtener una precisión decimal de hasta 28 dígitos.

Ejemplos

La siguiente expresión devuelve los valores en el puerto Price redondeados a tres posiciones decimales:

```
ROUND( PRICE, 3 )
```

PRICE	RETURN VALUE
12.9936	12.994
15.9949	15.995
-18.8678	-18.868
56.9561	56.956
NULL	NULL

Se pueden redondear dígitos a la izquierda del punto decimal pasando un entero negativo en el argumento *precision*:

```
ROUND( PRICE, -2 )
```

PRICE	RETURN VALUE
13242.99	13200.0

PRICE	RETURN VALUE
1435.99	1400.0
-108.95	-100.0
NULL	NULL

Si se pasa un valor decimal en el argumento *precision*, el Servicio de integración de datos lo redondea al entero más próximo antes de evaluar la expresión:

```
ROUND( PRICE, 0.8 )
```

PRICE	RETURN VALUE
12.99	13.0
56.34	56.3
NULL	NULL

Si omite el argumento *precision*, la función efectúa el redondeo al entero más próximo:

```
ROUND( PRICE )
```

PRICE	RETURN VALUE
12.99	13.0
-15.99	-16.0
-18.99	-19.0
56.95	57.0
NULL	NULL

Consejo

También se puede usar ROUND para establecer de forma explícita la precisión de los valores calculados y obtener resultados esperados. Cuando el Servicio de integración de datos se ejecuta en modo de precisión baja, trunca el resultado de los cálculos cuando la precisión del valor excede 15 dígitos. Por ejemplo, quizá desee procesar la siguiente expresión en modo de precisión baja:

```
7/3 * 3 = 7
```

En este caso, el Servicio de integración de datos evalúa el lado izquierdo de la expresión como 6,999999999999999 porque trunca el resultado de la primera operación de división. El Servicio de integración de datos evalúa la expresión completa como FALSE. Es posible que este no sea el resultado esperado.

Para lograr el resultado esperado, utilice ROUND para redondear el resultado truncado de la izquierda de la expresión al resultado esperado. El Servicio de integración de datos evalúa la siguiente expresión como TRUE:

```
ROUND(7/3 * 3) = 7
```


RPAD

Convierte una cadena a una longitud especificada añadiendo espacios en blanco o caracteres al final de la cadena.

Sintaxis

```
RPAD( first_string, length [,second_string] )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>first_string</i>	Obligatorio	Cualquier valor de cadena. Cadenas que desea cambiar. Puede introducir cualquier expresión de transformación válida.
<i>length</i>	Obligatorio	Debe ser un literal entero positivo. Especifica la longitud que desea que tenga cada cadena.
<i>second_string</i>	Opcional	Cualquier valor de cadena. Pase la cadena que desea añadir a la derecha de los valores <i>first_string</i> . Incluya los caracteres que desea añadir al final de la cadena entre comillas simples, por ejemplo, 'abc'. Este argumento distingue entre mayúsculas y minúsculas. Si se omite la segunda cadena, la función rellena el final de la primera cadena con espacios en blanco.

Valor de retorno

Cadena de la longitud especificada.

NULL si el valor pasado a la función es NULL o si la longitud es un número negativo.

Ejemplos

La expresión siguiente devuelve el nombre del elemento con una longitud de 16 caracteres, añadiendo la cadena '.' al final de cada nombre de elemento:

```
RPAD( ITEM_NAME, 16, '.')
```

ITEM_NAME	RETURN VALUE
Flashlight	Flashlight.....
Compass	Compass.....
Regulator System	Regulator System
Safety Knife	Safety Knife....

RPAD cuenta la longitud de izquierda a derecha. Por lo tanto, si la primera cadena es más larga que la longitud, RPAD trunca la cadena de derecha a izquierda. Por ejemplo, RPAD ('alfabético', 5 'x') devolverá la cadena 'alfab'. RPAD utiliza una parte de la *second_string* si es necesario.

La expresión siguiente devuelve el nombre del elemento con una longitud de 16 caracteres, añadiendo la cadena '*' al final de cada nombre de elemento:

```
RPAD( ITEM_NAME, 16, '*' )
```

ITEM_NAME	RETURN VALUE
Flashlight	Flashlight*.*.*.
Compass	Compass*.*.*.*.*
Regulator System	Regulator System
Safety Knife	Safety Knife*.*

RTRIM

Quita espacios en blanco o caracteres del final de una cadena.

Si no especifica un parámetro *trim_set* en la expresión:

- En UNICODE, RTRIM quita los espacios de uno y dos bytes del final de una cadena
- En ASCII, RTRIM quita solamente los espacios de un byte.

Si utiliza RTRIM para quitar los caracteres de una cadena, RTRIM compara, carácter por carácter, el argumento *trim_set* con cada carácter del argumento *string*, empezando por la derecha de la cadena. Si el carácter de la cadena coincide con cualquier carácter de *trim_set*, RTRIM lo quita. RTRIM sigue la comparación y quita los caracteres hasta que no puede encontrar un carácter coincidente en *trim_set*. Devuelve la cadena sin los caracteres coincidentes.

Sintaxis

```
RTRIM( string [, trim_set] )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>string</i>	Obligatorio	Cualquier valor de cadena. Transfiere los valores que desea recortar. Puede especificar cualquier expresión de transformación válida. Utilice operadores para efectuar las comparaciones o para concatenar las cadenas antes de quitar los espacios en blanco del final de una cadena.
<i>trim_set</i>	Opcional	Cualquier valor de cadena. Transfiere los caracteres que desea quitar del final de la cadena. También puede especificar un literal de texto. Sin embargo, debe encerrar entre comillas simples los caracteres que desea quitar del final de la cadena, por ejemplo, 'abc'. Si omite la segunda cadena, la función quita los espacios en blanco del final de la primera cadena. RTRIM distingue entre mayúsculas y minúsculas.

Valor de devolución

Cadena. Se quitan los valores de cadena con los caracteres especificados en el argumento *trim_set*.

NULL si el valor que se ha transferido a la función es NULL.

Ejemplo:

La siguiente expresión quita los caracteres 're' de las cadenas del puerto LAST_NAME:

```
RTRIM( LAST_NAME, 're')
```

LAST_NAME	RETURN VALUE
Nelson	Nelson
Page	Pag
Osborne	Osborn
NULL	NULL
Sawyer	Sawyer
H. Bender	H. Bend
Steadman	Steadman

RTRIM quita 'e' del término Page aunque 'r' sea el primer carácter del argumento *trim_set*. El motivo es que RTRIM busca, carácter por carácter, el conjunto de caracteres especificado en el argumento *trim_set*. Si el último carácter de la cadena coincide con el primer carácter de *trim_set*, RTRIM lo quita. Sin embargo, si el último carácter de la cadena no coincide, RTRIM compara el segundo carácter del argumento *trim_set*. Si el segundo carácter por el final de la cadena coincide con el segundo carácter del argumento *trim_set*, RTRIM lo quita, y así sucesivamente. Cuando el carácter de la cadena no coincide con *trim_set*, RTRIM devuelve la cadena y verifica la siguiente fila.

En el último ejemplo, el último carácter del término Nelson no coincide con ninguno de los caracteres del argumento *trim_set*, así que RTRIM devuelve la cadena 'Nelson' y analiza la siguiente fila.

Consejos para RTRIM

Utilice RTRIM y LTRIM con || o CONCAT para quitar los espacios en blanco al principio o al final después de concatenar dos cadenas.

Si anida RTRIM, también puede quitar varios conjuntos de caracteres. Por ejemplo, si desea quitar los espacios en blanco al final y el carácter 't' del final de cada cadena en una columna de nombres, puede crear una expresión similar a la siguiente:

```
RTRIM( RTRIM( NAMES ), 't' )
```

SETCOUNTVARIABLE

Cuenta las filas evaluadas por la función, e incrementa el valor actual de una variable de asignación basándose en el tipo de fila actual. Aumenta el valor actual en uno para cada fila marcada para inserción. Disminuye el valor actual en uno para cada fila marcada para eliminación. Mantiene el valor actual igual para cada fila marcada para actualización o rechazo. Devuelve el nuevo valor actual.

Cuando termina una sesión con éxito, el Servicio de integración de datos guarda el último valor actual en el repositorio. Cuando se utiliza con una sesión que contiene varias particiones, el Servicio de integración de datos genera valores actuales diferentes para cada partición. Al final de la sesión, determina el recuento

total para todas las particiones y guarda el total en el repositorio. A menos que se reemplace, utiliza el valor guardado como valor inicial de la variable para la próxima vez que utilice esta sesión.

Utilice la función SETCOUNTVARIABLE solo una vez para cada variable de asignación en un canal. El Servicio de integración de datos procesa las funciones de variable a medida que las encuentra en la asignación. El orden en el que el Servicio de integración de datos encuentra las funciones de variable en la asignación puede que no sea el mismo cada vez que se ejecuta la sesión. Esta diferencia puede generar resultados incoherentes cuando se utiliza la misma función de variable varias veces en una asignación.

Utilice SETCOUNTVARIABLE con variables de asignación con un tipo de agregación de recuento. Utilice SETCOUNTVARIABLE en las siguientes transformaciones:

- Expresión
- Filtro
- Enrutador
- Estrategia de actualización

El Servicio de integración de datos no guarda el valor final de una variable de asignación en el repositorio cuando se da una de las siguientes condiciones:

- La sesión no se puede completar.
- La sesión se ha configurado para una carga de prueba.
- La sesión es una sesión de depuración.
- La sesión se ejecuta en modo de depuración y se ha configurado para descartar la salida de la sesión.

Sintaxis

```
SETCOUNTVARIABLE( $$Variable )
```

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

Argumento	Obligatorio / Opcional	Descripción
\$\$Variable	Obligatorio	Nombre de la variable de asignación que desea configurar. Utilice variables de asignación con un tipo de agregación de recuento.

Valor de devolución

El valor actual de la variable.

Ejemplo:

Tiene una asignación que actualiza una tabla de dimensión que cambia lentamente y que contiene información del distribuidor. La siguiente expresión hace un recuento del número de distribuidores actuales con la variable de asignación \$\$CurrentDistributors y devuelve el valor al puerto CUR_DIST. Aumenta el recuento por uno por cada fila insertada, disminuye el recuento para cada fila eliminada y mantiene el recuento igual para todas las filas actualizadas o rechazadas. El valor inicial de \$\$CurrentDistributors la última vez que se ejecutó la sesión es de 23.

```
SETCOUNTVARIABLE ($$CurrentDistributors)
```

(row marked for...)	DIST_ID	DISTRIBUTOR	CUR_DIST
(update)	000015	MSD Inc.	23

(row marked for...)	DIST_ID	DISTRIBUTOR	CUR_DIST
(insert)	000024	Darkroom Co.	24
(insert)	000025	Howard's Supply	25
(update)	000003	JNR Ltd.	25
(delete)	000024	Darkroom Co.	24
(insert)	000026	Supply.com	25

Al final de la sesión, el Servicio de integración de datos guarda '25' en el repositorio como valor actual para \$
\$CurrentDistributors. La próxima vez que se ejecuta la sesión, el Servicio de integración de PowerCenter
verifica el valor inicial de \$\$CurrentDistributors en '25'.

El Servicio de integración de datos guarda el mismo valor de \$\$CurrentDistributors en el repositorio para las
sesiones con varias particiones y también para las sesiones con una única partición.

SET_DATE_PART

Establece una parte del valor Fecha/Hora en el valor que usted especifique. Con SET_DATE_PART, puede
cambiar las siguientes partes de una fecha:

- **Año.** Puede cambiar el año si especifica un entero positivo en el argumento *value*. Utilice una de las
siguientes cadenas de formato de año: Y, YY, YYYY o YYYY para establecer el año. La siguiente expresión,
por ejemplo, cambia el año a 2001 para todas las fechas del puerto SHIP_DATE:

```
SET_DATE_PART( SHIP_DATE, 'YY', 2001 )
```

- **Mes.** Puede cambiar el mes si especifica un entero positivo entre 1 y 12 (enero=1 y diciembre=12) en el
argumento *value*. Utilice una de las siguientes cadenas de formato de mes: MM, MON, MONTH para
establecer el mes. La siguiente expresión, por ejemplo, cambia el mes a octubre para todas las fechas del
puerto SHIP_DATE:

```
SET_DATE_PART( SHIP_DATE, 'MONTH', 10 )
```

- **Día.** Puede cambiar el día si especifica un entero positivo entre el 1 y el 31 (excepto en los meses con
menos de 31 días: febrero, abril, junio, setiembre y noviembre) en el argumento *value*. Utilice una de las
siguientes cadenas de formato de mes (D, DD, DDD, DY y DAY) para establecer el día. La siguiente
expresión, por ejemplo, cambia el día a 10 para todas las fechas del puerto SHIP_DATE:

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

- **Hora.** Puede cambiar la hora si especifica un entero positivo entre 0 y 24 (donde 0=12 AM 12=12 PM y 24
=12 AM) en el argumento *value*. Utilice una de las cadenas de formato de hora (HH, HH12, HH24) para
establecer la hora. La siguiente expresión, por ejemplo, cambia la hora a 14:00:00 (o 2:00:00 PM) para
todas las fechas del puerto SHIP_DATE:

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

- **Minuto.** Puede cambiar los minutos si especifica un entero positivo entre 0 y 59 en el argumento *value*.
Utilice la cadena de formato MI para establecer el minuto. La siguiente expresión, por ejemplo, cambia el
minuto a 25 para todas las fechas del puerto SHIP_DATE:

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

- **Segundos.** Puede cambiar los segundos si especifica un entero positivo entre 0 y 59 en el argumento *value*. Utilice la cadena de formato SS para establecer el segundo. La siguiente expresión, por ejemplo, cambia el segundo a 59 para todas las fechas del puerto SHIP_DATE:

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

- **Milisegundos.** Puede cambiar los milisegundos si especifica un entero positivo entre 0 y 999 en el argumento *value*. Utilice la cadena de formato MS para establecer los milisegundos. La siguiente expresión, por ejemplo, cambia el milisegundo a 125 para todas las fechas del puerto SHIP_DATE:

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

- **Microsegundos.** Puede cambiar los microsegundos si especifica un entero positivo entre 1000 y 999999 en el argumento *value*. Utilice la cadena de formato US para establecer los microsegundos. La siguiente expresión, por ejemplo, cambia los microsegundos a 12555 para todas las fechas del puerto SHIP_DATE:

```
SET_DATE_PART( SHIP_DATE, 'US', 12555 )
```

- **Nanosegundos.** Puede cambiar los nanosegundos si especifica un entero positivo entre 1000000 y 999999999 en el argumento *value*. Utilice la cadena de formato NS para establecer los nanosegundos. La siguiente expresión, por ejemplo, cambia los nanosegundos a 12555555 para todas las fechas del puerto SHIP_DATE:

```
SET_DATE_PART( SHIP_DATE, 'NS', 12555555 )
```

Sintaxis

```
SET_DATE_PART( date, format, value )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>date</i>	Obligatorio	Tipo de dato de fecha y hora. La fecha que desea modificar. Puede especificar cualquier expresión de transformación válida.
<i>format</i>	Obligatorio	Cadena de formato que especifica la porción de la fecha que se va a modificar. La cadena de formato no distingue entre mayúsculas y minúsculas.
<i>value</i>	Obligatorio	Un valor entero positivo que se va a asignar a la porción especificada de la fecha. El entero debe ser un valor válido para la parte de la fecha que desea cambiar. Si especifica un valor incorrecto como 30 de febrero, se produce un error.

Valor de devolución

La fecha en el mismo formato que la fecha de origen con la parte que se ha especificado cambiada.

NULL si el valor que se ha transferido a la función es NULL.

Ejemplos

Las siguientes expresiones cambian la hora a 4 PM para las fechas del puerto DATE_PROMISED:

```
SET_DATE_PART( DATE_PROMISED, 'HH', 16 )
SET_DATE_PART( DATE_PROMISED, 'HH12', 16 )
SET_DATE_PART( DATE_PROMISED, 'HH24', 16 )
```

DATE_PROMISED	RETURN VALUE
Jan 1 1997 12:15:56AM	Jan 1 1997 4:15:56PM

DATE_PROMISED	RETURN VALUE
Feb 13 1997 2:30:01AM	Feb 13 1997 4:30:01PM
Mar 31 1997 5:10:15PM	Mar 31 1997 4:10:15PM
Dec 12 1997 8:07:33AM	Dec 12 1997 4:07:33PM
NULL	NULL

Las siguientes expresiones cambian el mes a junio para las fechas del puerto DATE_PROMISED: El Servicio de integración de datos muestra un error cuando intenta crear una fecha que no existe, como cuando cambia de 31 de marzo a 31 de junio:

```
SET_DATE_PART( DATE_PROMISED, 'MM', 6 )
SET_DATE_PART( DATE_PROMISED, 'MON', 6 )
SET_DATE_PART( DATE_PROMISED, 'MONTH', 6 )
```

DATE_PROMISED	RETURN VALUE
Jan 1 1997 12:15:56AM	Jun 1 1997 12:15:56AM
Feb 13 1997 2:30:01AM	Jun 13 1997 2:30:01AM
Mar 31 1997 5:10:15PM	<i>Error. Integration Service doesn't write row.</i>
Dec 12 1997 8:07:33AM	Jun 12 1997 8:07:33AM
NULL	NULL

Las siguientes expresiones cambian el año a 2000 para las fechas del puerto DATE_PROMISED:

```
SET_DATE_PART( DATE_PROMISED, 'Y', 2000 )
SET_DATE_PART( DATE_PROMISED, 'YY', 2000 )
SET_DATE_PART( DATE_PROMISED, 'YYY', 2000 )
SET_DATE_PART( DATE_PROMISED, 'YYYY', 2000 )
```

DATE_PROMISED	RETURN VALUE
Jan 1 1997 12:15:56AM	Jan 1 2000 12:15:56AM
Feb 13 1997 2:30:01AM	Feb 13 2000 2:30:01AM
Mar 31 1997 5:10:15PM	Mar 31 2000 5:10:15PM
Dec 12 1997 8:07:33AM	Dec 12 2000 4:07:33PM
NULL	NULL

Consejo

Si desea cambiar varias partes de una fecha a la vez, puede anidar varias funciones SET_DATE_PART dentro del argumento *date*. Puede escribir, por ejemplo, la siguiente expresión para cambiar todas las fechas del puerto DATE_ENTERED a 1 de julio de 1998:

```
SET_DATE_PART( SET_DATE_PART( SET_DATE_PART( DATE_ENTERED, 'YYYY', 1998), 'MM', 7), 'DD', 1)
```

SETMAXVARIABLE

Establece el valor actual de una variable de asignación en el mayor de dos valores: el valor actual de la variable o el valor que usted especifique. Devuelve el nuevo valor actual. La función se ejecuta únicamente si una fila está marcada para inserción. SETMAXVARIABLE omite el resto de tipos de filas y el valor actual permanece inalterado.

Al final de una sesión correcta, el Servicio de integración de datos guarda el valor actual final en el repositorio. Cuando se utiliza con una sesión que contiene varias particiones, el Servicio de integración de datos genera diferentes valores actuales para cada partición. Al final de la sesión, guarda el mayor valor actual de todas las particiones en el repositorio. A menos que se anule, utiliza el valor guardado como el valor inicial de la variable para la ejecución de la siguiente sesión.

Cuando se utiliza con una variable de asignación de cadena, SETMAXVARIABLE devuelve la cadena mayor según el criterio de ordenación seleccionado para la sesión.

Utilice la función SETMAXVARIABLE una sola vez para cada variable de asignación en un canal. El Servicio de integración de datos procesa las funciones de variables a medida que las encuentra en la asignación. El orden en el que el Servicio de integración de datos encuentra las funciones de variables en la asignación podría no ser el mismo para cada ejecución de sesión. Esto puede provocar resultados inconsistentes cuando utiliza la misma función de variable varias veces en una asignación.

Utilice SETMAXVARIABLE con variables de asignación con un tipo máximo de agregación. Utilice SETMAXVARIABLE en las siguientes transformaciones:

- Expresión
- Filtro
- Enrutador
- Estrategia de actualización

El Servicio de integración de datos no guarda el valor final de una variable de asignación en el repositorio cuando se cumple alguna de las siguientes condiciones:

- La sesión no se puede completar.
- La sesión está configurada para una carga de prueba.
- La sesión es una sesión de depuración.
- La sesión se ejecuta en modo de depuración y se configura para descartar la salida de la sesión.

Sintaxis

```
SETMAXVARIABLE( $$Variable, value )
```

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

Argumento	Obligatorio / Opcional	Descripción
<code>\$\$Variable</code>	Obligatorio	Nombre de la variable de asignación que desea configurar. Utilice las variables de asignación con el tipo máximo de agregación.
<code>value</code>	Obligatorio	El valor que desea que compare el Servicio de integración de datos con el valor actual de la variable. Puede introducir cualquier expresión de transformación válida que dé como resultado un tipo de datos compatible con el tipo de datos de la variable.

Valor de retorno

El más alto de los dos valores: el valor actual de la variable o el valor que ha especificado. El valor de retorno es el nuevo valor actual de la variable.

Cuando *value* es NULL, el Servicio de integración de datos devuelve el valor actual de \$\$Variable.

Ejemplos

La expresión siguiente compara el número de artículos adquiridos en cada transacción con una variable de asignación \$\$MaxItems. Establece \$\$MaxItems en el mayor de los dos valores y devuelve el número más alto históricamente de los artículos adquiridos en una sola transacción al puerto MAX_ITEMS. El valor inicial de \$\$MaxItems de la ejecución de sesión anterior es 22.

```
SETMAXVARIABLE ($$MAXITEMS, ITEMS)
```

TRANSACTION	ITEMS	MAX_ITEMS
0100002	12	22
0100003	5	22
0100004	18	22
0100005	35	35
0100006	5	35
0100007	14	35

Al final de la sesión, el Servicio de integración de datos guarda '35' en el repositorio como el valor máximo actual de \$\$MaxItems. La siguiente vez que se ejecuta la sesión, el Servicio de integración de datos da '35' como valor inicial de \$\$MaxItems.

Si la misma sesión contiene tres particiones, el Servicio de integración de datos devuelve \$\$MaxItems para cada partición. A continuación, guarda el mayor valor en el repositorio. Por ejemplo, el último valor evaluado por \$\$MaxItems en cada partición es el siguiente:

Partition	Final Current Value for \$\$MaxItems
Partition 1	35
Partition 2	23
Partition 3	22

SETMINVARIABLE

Establece el valor actual de una variable de asignación en el menor de dos valores: el valor actual de la variable o el valor que usted especifique. Devuelve el nuevo valor actual. La función SETMINVARIABLE se ejecuta únicamente si una fila está marcada para inserción. SETMINVARIABLE omite el resto de tipos de filas y el valor actual permanece inalterado.

Cuando termina una sesión con éxito, el Servicio de integración de datos guarda el valor final actual en el repositorio. Cuando se utiliza con una sesión que contiene varias particiones, el Servicio de integración de datos genera valores actuales diferentes para cada partición. Al final de la sesión, guarda el valor actual más bajo de todas las particiones en el repositorio. A menos que se reemplace, utiliza el valor guardado como valor inicial de la variable para la próxima vez que se ejecute la sesión.

Cuando se utiliza con una variable de asignación de cadena, SETMINVARIABLE devuelve la cadena más baja en función del orden de clasificación seleccionado para la sesión.

Utilice la función SETMINVARIABLE solo una vez para cada variable de asignación en un canal. El Servicio de integración de datos procesa las funciones de variable a medida que las encuentra en la asignación. El orden en el que el Servicio de integración de datos encuentra las funciones de variable en la asignación puede que no sea el mismo para cada ejecución de sesión. Esta diferencia puede generar resultados incoherentes cuando se utiliza la misma función de variable varias veces en una asignación.

Utilice SETMINVARIABLE con variables de asignación con un tipo de agregación mínima. Utilice SETMINVARIABLE en las siguientes transformaciones:

- Expresión
- Filtro
- Enrutador
- Estrategia de actualización

El Servicio de integración de datos no guarda el valor final de una variable de asignación en el repositorio cuando se da una de las siguientes condiciones:

- La sesión no se puede completar.
- La sesión se ha configurado para una carga de prueba.
- La sesión es una sesión de depuración.
- La sesión se ejecuta en modo de depuración y se ha configurado para descartar la salida de la sesión.

Sintaxis

```
SETMINVARIABLE( $$Variable, value )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>\$\$Variable</i>	Obligatorio	Nombre de la variable de asignación que desea configurar. Debe utilizarlo con las variables de asignación con un tipo de agregación mínimo.
<i>value</i>	Obligatorio	El valor que desea que el Servicio de integración de datos compare con el valor actual de la variable. Puede especificar cualquier expresión de transformación válida que verifique un tipo de datos compatible con el tipo de datos de la variable.

Valor de devolución

El menor de dos valores: el valor actual de la variable o el valor que ha especificado. El valor de devolución es el nuevo valor actual de la variable.

Cuando *value* es NULL, el Servicio de integración de datos devuelve el valor actual de *\$\$Variable*.

Ejemplo:

La siguiente expresión compara el precio de un artículo con una variable de asignación \$\$MinPrice. Configura \$\$MinPrice en el menor valor de los dos valores y devuelve el precio del artículo históricamente más bajo al puerto MIN_PRICE. El valor inicial de \$\$MinPrice en la ejecución de sesión anterior es de 22,50.

```
SETMINVARIABLE ($$MinPrice, PRICE)
```

DATE	PRICE	MIN_PRICE
05/01/2000 09:00:00	23.50	22.50
05/01/2000 10:00:00	27.00	22.50
05/01/2000 11:00:00	26.75	22.50
05/01/2000 12:00:00	25.25	22.50
05/01/2000 13:00:00	22.00	22.00
05/01/2000 14:00:00	22.75	22.00
05/01/2000 15:00:00	23.00	22.00
05/01/2000 16:00:00	24.25	22.00
05/01/2000 17:00:00	24.00	22.00

Al final de la sesión, el Servicio de integración de datos guarda 22,00 en el repositorio como valor mínimo actual para \$\$MinPrice. La próxima vez que se ejecuta la sesión, el Servicio de integración de datos verifica el valor inicial de \$\$MinPrice en 22,00.

Si la misma sesión contiene tres particiones, el Servicio de integración de datos verifica \$\$MinPrice para cada partición. A continuación, guarda el valor más pequeño en el repositorio. Por ejemplo, el último valor evaluado para \$\$MinPrice en cada partición es el que se indica a continuación:

Partition	Final Current Value for \$\$MinPrice
Partition 1	22.00
Partition 2	22.50
Partition 3	22.50

SETVARIABLE

Establece el valor actual de una variable de asignación en un valor que usted especifique. Devuelve el valor especificado. La función SETVARIABLE se ejecuta únicamente si una fila está marcada para inserción o actualización. SETVARIABLE omite el resto de tipos de filas y el valor actual permanece inalterado.

Cuando termina una sesión con éxito, el Servicio de integración de datos compara el valor final actual de la variable con el valor inicial de la variable. En función del tipo de agregado de la variable, guarda un valor final actual en el repositorio. A menos que se reemplace, utiliza el valor guardado como valor inicial de la variable para la próxima vez que se ejecute la sesión.

Utilice la función SETVARIABLE solo una vez para cada variable de asignación en un canal. El Servicio de integración de datos procesa las funciones de variable a medida que las encuentra en la asignación. El orden en el que el Servicio de integración de datos encuentra las funciones de variable en la asignación puede que no sea el mismo cada vez que se ejecuta la sesión. Esta diferencia puede generar resultados incoherentes cuando se utiliza la misma función de variable varias veces en una asignación.

Utilice SETVARIABLE en las siguientes transformaciones:

- Expresión
- Filtro
- Enrutador
- Estrategia de actualización

El Servicio de integración de datos no guarda el valor final de una variable de asignación en el repositorio cuando se da una de las siguientes condiciones:

- La sesión no se puede completar.
- La sesión se ha configurado para una carga de prueba.
- La sesión es una sesión de depuración.
- La sesión se ejecuta en modo de depuración y se ha configurado para descartar la salida de la sesión.

Sintaxis

```
SETVARIABLE( $$Variable, value )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>\$\$Variable</i>	Obligatorio	Nombre de la variable de asignación que desea configurar. Debe utilizarlo con las variables de asignación con un tipo de agregación mínimo/máximo.
<i>value</i>	Obligatorio	El valor con el que desea configurar el valor actual de la variable. Puede especificar cualquier expresión de transformación válida que verifique un tipo de datos compatible con el tipo de datos de la variable.

Valor de devolución

El valor actual de la variable.

Cuando *value* es NULL, el Servicio de integración de datos devuelve el valor actual de *\$\$Variable*.

Ejemplos

La siguiente expresión configura una variable de asignación \$\$Time con la fecha del sistema en el mismo momento en que el Servicio de integración de datos verifica la fila y devuelve la fecha del sistema al puerto SET_\$\$TIME:

```
SETVARIABLE ( $$Time, SYSDATE )
```

TRANSACTION	TOTAL	SET_\$\$TIME
0100002	534.23	10/10/2000 01:34:33
0100003	699.01	10/10/2000 01:34:34

TRANSACTION	TOTAL	SET_\$\$TIME
0100004	97.50	10/10/2000 01:34:35
0100005	116.43	10/10/2000 01:34:36
0100006	323.95	10/10/2000 01:34:37

Al final de la sesión, el Servicio de integración de datos guarda 10/10/2000 01:34:37 en el repositorio como último valor actual verificado para \$\$Time. La próxima vez que se ejecuta la sesión, el Servicio de integración de datos verifica todas las referencias de \$\$Time en 10/10/2000 01:34:37.

La siguiente expresión configura la variable de asignación \$\$Timestamp en las marcas horarias asociadas a la fila y devuelve la marca horaria al puerto SET_\$\$TIMESTAMP:

```
SETVARIABLE ($$Time, TIMESTAMP)
```

TRANSACTION	TIMESTAMP	TOTAL	SET_\$\$TIMESTAMP
0100002	10/01/2000 12:01:01	534.23	10/01/2000 12:01:01
0100003	10/01/2000 12:10:22	699.01	10/01/2000 12:10:22
0100004	10/01/2000 12:16:45	97.50	10/01/2000 12:16:45
0100005	10/01/2000 12:23:10	116.43	10/01/2000 12:23:10
0100006	10/01/2000 12:40:31	323.95	10/01/2000 12:40:31

Al final de la sesión, el Servicio de integración de datos guarda 10/01/2000 12:40:31 en el repositorio como último valor actual verificado para \$\$Timestamp.

La próxima vez que se ejecuta la sesión, el Servicio de integración de datos verifica el valor inicial de \$\$Timestamp en 10/01/2000 12:40:31.

SIGN

Se devuelve si un valor numérico es positivo, negativo o 0.

Sintaxis

```
SIGN( numeric_value )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Valor numérico. Pasa los valores que se van a evaluar. Puede especificar cualquier expresión de transformación válida.

Valor devuelto

Se devuelve -1 para los valores negativos.

Se devuelve 0 para 0.

Se devuelve 1 para los valores positivos.

Se devuelve NULL si el valor es NULL.

Ejemplo

La siguiente expresión determina si el puerto SALES incluye valores negativos:

```
SIGN( SALES )
```

SALES	RETURN VALUE
100	1
-25.99	-1
0	0
NULL	NULL

SIN

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

Sintaxis

```
SIN( numeric_value )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. Datos numéricos expresados en radianes (grados multiplicados por pi y divididos entre 180). Pasa los valores para los que se va a calcular el seno. Puede especificar cualquier expresión de transformación válida. Además, puede usar operadores para convertir un valor numérico en radianes o realizar una operación aritmética durante el cálculo de SIN.

Valor devuelto

Valor doble.

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

Ejemplo

La siguiente expresión convierte los valores del puerto Degrees en radianes y, a continuación, calcula el seno para cada radián:

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

DEGREES	RETURN VALUE
0	0
90	1
70	0.939692620785936
30	0.500000000000003
5	0.0871557427476639
89	0.999847695156393
NULL	NULL

Puede realizar una operación aritmética con los valores pasados a SIN antes de que la función calcule el seno. Por ejemplo:

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

SINH

Devuelve el seno hiperbólico del valor numérico.

Sintaxis

```
SINH( numeric_value )
```

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

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

Valor devuelto

Valor doble.

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

Ejemplo

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

```
SINH( ANGLES )
```

ANGLES	RETURN VALUE
1.0	1.1752011936438
2.897	9.03225804884884
3.66	19.4178051793031
5.45	116.376934801486
0	0.0
0.345	0.35188478309993
NULL	NULL

Sugerencia

Puede realizar una operación aritmética con los valores pasados a SINH antes de que la función calcule el seno. Por ejemplo:

```
SINH( MEASURES.ARCS / 180 )
```

SOUNDEX

Cifra el valor de una cadena en una cadena de cuatro caracteres.

SOUNDEX funciona con caracteres del alfabeto inglés (A-Z). Utiliza el primer carácter de la cadena de entrada como el primer carácter del valor de retorno y codifica las tres consonantes restantes únicas como números.

SOUNDEX codifica caracteres de acuerdo a la siguiente lista de reglas:

- Utiliza el primer carácter de *string* como el primer carácter del valor de retorno y lo codifica en caja mayúscula. Por ejemplo, tanto SOUNDEX('John') como SOUNDEX('john') devuelven 'J500'.
- Codifica las tres primeras consonantes únicas que siguen al primer carácter en *string* y omite el resto. Por ejemplo, tanto SOUNDEX('JohnRB') como SOUNDEX('JohnRBCD') devuelven 'J561'.
- Asigna un código individual a consonantes que suenan de forma parecida.

La siguiente tabla incluye las pautas de codificación de SOUNDEX para consonantes:

Tabla 2. Pautas de codificación SOUNDEX para consonantes

Code	Consonant
1	B, P, F, V
2	C, S, G, J, K, Q, X, Z
3	D, T
4	L
5	M, N
6	R

- Omite los caracteres A, E, I, O, U, H y W excepto en los casos en que uno de ellos sea el primero en *string*. Por ejemplo, SOUNDEX('A123') devuelve 'A000' y SOUNDEX('MAeiouhwC') devuelve 'M000'.
- Si *string* produce menos de cuatro caracteres, SOUNDEX rellena la cadena resultante con ceros. Por ejemplo, SOUNDEX('J') devuelve 'J000'.
- Si *string* contiene un conjunto de consonantes consecutivas que utilizan el mismo código que aparece en ["SOUNDEX" en la página 192](#), SOUNDEX codifica la primera ocurrencia y omite el resto de ocurrencias del conjunto. Por ejemplo, SOUNDEX('AbbpdMN') devuelve 'A135'.
- Omite los números en *string*. Por ejemplo, tanto SOUNDEX('Joh12n') como SOUNDEX('1John') devuelven 'J500'.
- Devuelve NULL si *string* es NULL o si todos los caracteres en *string* no son letras del alfabeto inglés.

Sintaxis

```
SOUNDEX( string )
```

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

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

Valor de retorno

Cadena.

NULL si una de las siguientes condiciones es verdadera:

- Si el valor pasado a la función es NULL.
- Ningún carácter en *string* es una letra del alfabeto inglés.
- *string* está vacío.

Ejemplo

La siguiente expresión codifica los valores en el puerto EMPLOYEE_NAME:

```
SOUNDEX ( EMPLOYEE_NAME )
```

EMPLOYEE_NAME	RETURN VALUE
John	J500
William	W450
jane	J500
joh12n	J500
1abc	A120
NULL	NULL

SQL_LIKE

Devuelve si un valor coincide con un patrón de expresión regular. Permite validar patrones de datos como números de identificación, números de teléfono, códigos postales y nombres de estado.

Sintaxis

```
SQL_LIKE(subject, pattern, escape character)
```

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

Argumento	Obligatorio/ Opcional	Descripción
subject	Obligatorio	Tipo de datos de cadena. Pasa el valor que se desea hacer coincidir con la expresión regular. Encierre el valor entre comillas sencillas.
patrón	Obligatorio	Tipo de datos de cadena. Expresión regular con la que se desea coincidir. Encierre el patrón entre comillas sencillas.
carácter de escape	Opcional	Tipo de datos de cadena. La función SQL_LIKE admite como caracteres de escape el signo de porcentaje (%) y el signo de subrayado (_). Encierre el carácter de escape entre comillas sencillas.

Valor devuelto

TRUE si los datos coinciden con el patrón.

FALSE si los datos no coinciden con el patrón.

NULL si la entrada es un valor nulo o si el patrón es NULL.

Ejemplo

Puede utilizar SQL_LIKE en una expresión para buscar nombres que coincidan con un patrón. Por ejemplo, la siguiente expresión hace coincidir nombres que tengan el patrón "A_%" con el carácter de escape '#':

```
SQL_LIKE(ENAME, 'A_#%', '#')
```

ENAME	Valor
SMITH	FALSE
AX%	TRUE
MILLER	FALSE
A%	FALSE
JONES	FALSE
BLAKE	FALSE
A%l	FALSE

SQRT

Devuelve la raíz cuadrada de un valor numérico no negativo.

Sintaxis

```
SQRT( numeric_value )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Valor numérico positivo. Pasa los valores para los que se va a calcular la raíz cuadrada. Puede especificar cualquier expresión de transformación válida.

Valor devuelto

Valor doble.

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

Ejemplo

La siguiente expresión devuelve la raíz cuadrada de los valores del puerto Numbers:

```
SQRT( NUMBERS )
```

NUMBERS	RETURN VALUE
100	10

NUMBERS	RETURN VALUE
-100	Error. Servicio de integración de datos does not write row.
NULL	NULL
60.54	7.78074546557076

El valor -100 genera un error porque la función SQRT solamente evalúa valores numéricos positivos. Si pasa un valor negativo o un valor de carácter, el Servicio de integración de datos muestra un error de evaluación de transformación y no escribe la fila.

Puede realizar una operación aritmética con los valores pasados a SQRT antes de que la función calcule la raíz cuadrada.

STDDEV

Devuelve la desviación estándar de los valores numéricos que se pasan a esta función. STDDEV se utiliza para analizar datos estadísticos. Dentro de STDDEV solamente se puede anidar una función agregada adicional y la función anidada debe devolver un tipo de datos Numérico.

Sintaxis

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

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

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipos de datos numéricos. Esta función pasa los valores para los que se desea calcular una desviación estándar o los resultados de una función. Se puede especificar cualquier expresión de transformación válida. Se pueden usar operadores para obtener el promedio de valores en distintos puertos.
<i>filter_condition</i>	Opcional	Limita las filas en la búsqueda. La condición de filtro debe ser un valor numérico o dar un valor TRUE, FALSE o NULL. Se puede especificar cualquier expresión de transformación válida.

Valor de retorno

Valor numérico.

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

Nota: Si el valor devuelto es un decimal con una precisión superior a 15, puede habilitar el modo de alta precisión para asegurarse de obtener una precisión decimal de hasta 28 dígitos.

Nulos

Si un valor es NULL, STDDEV lo omite. Sin embargo, si todos los valores son NULL, STDDEV devuelve NULL.

Nota: De forma predeterminada, el Servicio de integración de datos trata los valores nulos como NULL en las funciones agregadas. Si pasa un puerto o grupo de valores nulos completo, la función devuelve NULL. Sin embargo, cuando se configura el Servicio de integración de datos, se puede indicar la forma en que se desea manejar los valores nulos en las funciones agregadas. En las funciones agregadas se pueden tratar los valores nulos como 0 o como NULL.

Agrupar por

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

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

Ejemplos

La siguiente expresión calcula la desviación estándar para todas las filas superiores a \$2000,00 en el puerto TOTAL_SALES:

```
STDDEV( SALES, SALES > 2000.00 )
```

SALES

2198.0

1010.90

2256.0

153.88

3001.0

NULL

8953.0

RETURN VALUE: 3254.60361129688

La función no incluye los valores 1010,90 y 153,88 en el cálculo porque *filter_condition* especifica ventas superiores a \$2.000.

La siguiente expresión calcula la desviación estándar para todas las filas en el puerto SALES:

```
STDDEV(SALES)
```

SALES

2198.0

2198.0

2198.0

2198.0

RETURN VALUE: 0

El valor de retorno es 0 porque cada fila contiene el mismo número (no hay ninguna desviación estándar). Si no hay ninguna desviación estándar, el valor de retorno es 0.

SUBSTR

Devuelve una porción de una cadena. SUBSTR cuenta todos los caracteres, incluidos los espacios en blanco, comenzando por el inicio de la cadena.

Sintaxis

```
SUBSTR( string, start [,length] )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>string</i>	Obligatorio	Debe ser una cadena de carácter. Pasa las cadenas que se desea buscar. Se puede especificar cualquier expresión de transformación válida. Si se pasa un valor numérico, la función la convierte en una cadena de caracteres.
<i>start</i>	Obligatorio	El valor debe ser un entero. La posición en la cadena donde se desea iniciar el recuento. Se puede especificar cualquier expresión de transformación válida. Si la posición inicial es un número positivo, SUBSTR localiza la posición inicial contando a partir del principio de la cadena. Si la posición inicial es un número positivo, SUBSTR localiza la posición inicial contando a partir del principio de la cadena. Si la posición inicial es 0, SUBSTR busca a partir del primer carácter de la cadena.
<i>length</i>	Opcional	Debe ser un entero mayor que 0. El número de caracteres que se desea que devuelva SUBSTR. Se puede especificar cualquier expresión de transformación válida. Si se omite el argumento de longitud, SUBSTR devuelve todos los caracteres a a partir de la posición inicial hasta el final de la cadena. Si se pasa un entero negativo o 0, la función devuelve una cadena vacía. Si se pasa un decimal, la función lo redondea al valor entero más próximo.

Valor de retorno

Cadena.

Cadena vacía si se pasa un valor negativo o de longitud 0.

NULL si el valor pasado a la función es NULL.

Ejemplos

Las siguientes expresiones devuelven el código de área para cada fila en el puerto Phone:

```
SUBSTR( PHONE, 0, 3 )
```

PHONE	RETURN VALUE
809-555-0269	809

PHONE	RETURN VALUE
-------	--------------

357-687-6708	357
NULL	NULL

SUBSTR(PHONE, 1, 3)

PHONE	RETURN VALUE
-------	--------------

809-555-3915	809
357-687-6708	357
NULL	NULL

Las siguientes expresiones devuelven el número de teléfono sin el código de área para cada fila en el puerto Phone:

SUBSTR(PHONE, 5, 8)

PHONE	RETURN VALUE
-------	--------------

808-555-0269	555-0269
809-555-3915	555-3915
357-687-6708	687-6708
NULL	NULL

También se puede pasar un valor inicial negativo para devolver el número de teléfono para cada fila en el puerto Phone: La expresión todavía lee la cadena origen de izquierda a derecha cuando devuelve el resultado del argumento *length*:

SUBSTR(PHONE, -8, 3)

PHONE	RETURN VALUE
-------	--------------

808-555-0269	555
809-555-3915	555
357-687-6708	687
NULL	NULL

Se puede anidar INSTR en el argumento *start* o *length* para buscar una cadena específica y devolver su posición.

La siguiente expresión evalúa una cadena, comenzando por el final de la misma. La expresión encuentra el último espacio (situado en el extremo derecho) de la cadena y luego devuelve todos los caracteres que lo preceden:

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

CUST_NAME	RETURN VALUE
PATRICIA JONES	PATRICIA
MARY ELLEN SHAH	MARY ELLEN

La siguiente expresión elimina el carácter '#' de una cadena:

```
SUBSTR( CUST_ID, 1, INSTR(CUST_ID, '#')-1 ) || SUBSTR( CUST_ID, INSTR(CUST_ID, '#')+1 )
```

Cuando el argumento *length* es más largo que la cadena, SUBSTR devuelve todos los caracteres desde la posición inicial hasta el final de la cadena. Vea el siguiente ejemplo:

```
SUBSTR('abcd', 2, 8)
```

El valor de retorno es 'bcd'. Compare este resultado con el siguiente ejemplo:

```
SUBSTR('abcd', -2, 8)
```

El valor de retorno es 'cd'.

SUM

Devuelve la suma de todos los valores de un puerto seleccionado. Opcionalmente, se puede aplicar un filtro para limitar el número de filas que se leen para calcular el total. Dentro de SUM solamente se puede anidar una función agregada adicional y la función anidada debe devolver un tipo de datos Numérico.

Sintaxis

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

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

Argumento:	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. Pasa los valores que se desea sumar. Se puede especificar cualquier expresión de transformación válida. Se pueden usar operadores para sumar valores en distintos puertos.
<i>filter_condition</i>	Opcional	Limita las filas en la búsqueda. La condición de filtro debe ser un valor numérico o dar un valor TRUE, FALSE o NULL. Se puede especificar cualquier expresión de transformación válida.

Valor de retorno

Valor numérico.

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

Nota: Si el valor devuelto es un decimal con una precisión superior a 15, puede habilitar el modo de alta precisión para asegurarse de obtener una precisión decimal de hasta 28 dígitos.

Nulos

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

Nota: De forma predeterminada, el Servicio de integración de datos trata los valores nulos como NULL en las funciones agregadas. Si pasa un puerto o grupo de valores nulos completo, la función devuelve NULL. Sin embargo, cuando se configura el Servicio de integración de datos, se puede indicar la forma en que se desea manejar los valores nulos en las funciones agregadas. En las funciones agregadas se pueden tratar los valores nulos como 0 o como NULL.

Agrupar por

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

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

Ejemplo

La siguiente expresión devuelve la suma de todos los valores superiores a 2000 en el puerto Sales:

```
SUM( SALES, SALES > 2000 )
```

SALES

2500.0

1900.0

1200.0

NULL

3458.0

4519.0

RETURN VALUE: 10477.0

Consejo

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

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

SYSTIMESTAMP

Devuelve la fecha y la hora actual del nodo que aloja el Servicio de integración de datos con precisión de nanosegundos. La precisión con que mostrará la fecha y hora depende de la plataforma.

El valor de retorno de la función varía en función de cómo configure el argumento:

- Al configurar el argumento de SYSTIMESTAMP como una variable, el Servicio de integración de datos evalúa la función para cada fila de la transformación.
- Al configurar el argumento de SYSTIMESTAMP como una constante, el Servicio de integración de datos evalúa la función una vez y guarda el valor para cada fila de la transformación.

Sintaxis

```
SYSTIMESTAMP( [format] )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>format</i>	Opcional	Precisión con la que desea recuperar la marca de hora. Puede especificar una precisión de hasta segundos (SS), milisegundos (MS), microsegundos (US), o nanosegundos (NS). Escriba la cadena de formato entre comillas simples. La cadena de formato no distingue entre mayúsculas y minúsculas. Por ejemplo, para mostrar la fecha y la hora con precisión de milisegundos, utilice la siguiente sintaxis: SYSTIMESTAMP('MS'). La precisión predeterminada es microsegundos (US).

Valor de retorno

Marca de hora. Devuelve la fecha y la hora con la precisión especificada.

Ejemplos

Su organización cuenta con un servicio de pedidos en línea y procesa datos en tiempo real. Puede utilizar la función SYSTIMESTAMP para generar una clave principal para cada transacción en la base de datos de destino.

Genere una transformación de expresión con los siguientes puertos y valores:

Port Name	Port Type	Expression
Customer_Name	Input	n/a
Order_Qty	Input	n/a
Time_Counter	Variable	'US'
Transaction_Id	Output	SYSTIMESTAMP (Time_Counter)

En tiempo de ejecución, el Servicio de integración de datos genera la hora del sistema con precisión de microsegundos para cada fila:

Customer_Name	Order_Qty	Transaction_Id
Vani Deed	14	07/06/2007 18:00:30.701015000
Kalia Crop	3	07/06/2007 18:00:30.701029000
Vani Deed	6	07/06/2007 18:00:30.701039000
Harry Spoon	32	07/06/2007 18:00:30.701048000

TAN

Devuelve la tangente de un valor numérico (expresado en radianes).

Sintaxis

```
TAN( numeric_value )
```

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

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

Valor devuelto

Valor doble.

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

Ejemplo

La siguiente expresión devuelve la tangente para todos los valores del puerto Degrees:

```
TAN( DEGREES * 3.14159 / 180 )
```

DEGREES	RETURN VALUE
70	2.74747741945531
50	1.19175359259435
30	0.577350269189672
5	0.0874886635259298
18	0.324919696232929
89	57.2899616310952
NULL	NULL

TANH

Devuelve la tangente hiperbólica del valor numérico pasado a esta función.

Sintaxis

```
TANH( numeric_value )
```

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

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

Valor devuelto

Valor doble.

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

Ejemplo

La siguiente expresión devuelve la tangente hiperbólica para los valores del puerto Angles:

```
TANH ( ANGLES )
```

ANGLES	RETURN VALUE
1.0	0.761594155955765
2.897	0.993926947790665
3.66	0.998676551914886
5.45	0.999963084213409
0	0.0
0.345	0.331933853503641
NULL	NULL

Sugerencia

Puede realizar una operación aritmética con los valores pasados a TANH antes de que la función calcule la tangente hiperbólica. Por ejemplo:

```
TANH ( ARCS / 360 )
```

TO_BIGINT

Convierte una cadena o valor numérico a un valor bigint. La sintaxis de TO_BIGINT contiene un argumento opcional que puede elegir para redondear el número al entero más cercano o truncar la parte decimal. TO_BIGINT ignora los espacios a la izquierda.

Sintaxis

```
TO_BIGINT( value [, flag] )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>value</i>	Obligatorio	Tipo de dato de cadena o numérico. Pasa el valor que desea convertir en un valor bigint. Puede introducir cualquier expresión de transformación válida.
<i>flag</i>	Opcional	<p>Marca que especifica si trunca o redondear la parte decimal. La marca debe ser un literal entero o las constantes TRUE o FALSE.</p> <p>TO_BIGINT trunca la parte decimal cuando la marca es TRUE o un número distinto de 0.</p> <p>TO_BIGINT redondea el valor al entero más cercano si la marca es FALSE o 0, o si se omite este argumento.</p> <p>La marca no está definida de forma predeterminada.</p>

Valor de retorno

Bigint.

NULL si el valor pasado a la función es NULL.

0 si el valor pasado a la función contiene caracteres alfanuméricos.

Si el valor que se pasa a la función contiene datos que no son válidos para un valor bigint, el servicio de integración de datos marca la fila como fila de error o no puede realizar la asignación.

Ejemplos

Las expresiones siguientes utilizan valores del puerto IN_TAX:

```
TO_BIGINT( IN_TAX, TRUE )
```

IN_TAX	RETURN VALUE
'7245176201123435.6789'	7245176201123435
'7245176201123435.2'	7245176201123435
'7245176201123435.2.48'	7245176201123435
NULL	NULL
'A12.3Grove'	0
'A12.3Grove'	<i>Error. Integration Service skips this row.</i>
' 176201123435.87'	176201123435
'-7245176201123435.2'	-7245176201123435
'-7245176201123435.23'	-7245176201123435

IN_TAX	RETURN VALUE
-9223372036854775806.9	-9223372036854775806
9223372036854775806.9	9223372036854775806
TO_BIGINT(IN_TAX)	
IN_TAX	RETURN VALUE
'7245176201123435.6789'	7245176201123436
'7245176201123435.2'	7245176201123435
'7245176201123435.348'	7245176201123435
NULL	NULL
'A12.3Grove'	0
'A12.3Grove'	<i>Error. Integration Service skips this row.</i>
' 176201123435.87'	176201123436
'-7245176201123435.6789'	-7245176201123436
'-7245176201123435.23'	-7245176201123435
-9223372036854775806.9	-9223372036854775807
9223372036854775806.9	9223372036854775807

TO_CHAR (Fechas)

Convierte fechas en cadenas de caracteres. TO_CHAR convierte también valores numéricos en cadenas. Puede convertir la fecha en cualquier formato mediante las cadenas de formato TO_CHAR.

TO_CHAR (fecha [,formato]) convierte un tipo de datos o un valor interno con un tipo de datos de fecha, marca de tiempo, marca de tiempo con zona horaria o marca de tiempo con zona horaria local en un valor de un tipo de datos de cadena especificado por la cadena de formato.

Sintaxis

```
TO_CHAR( date [,format] )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>date</i>	Obligatorio	Tipo de dato de fecha y hora. Pasa los valores de fecha que desea convertir en cadenas de caracteres. Se puede especificar cualquier expresión de transformación válida.
<i>format</i>	Opcional	<p>Especifique una cadena de formato TO_CHAR válida. La cadena de formato define el formato del valor devuelto, no el formato de los valores del argumento date. Si omite la cadena de formato, la función devuelve una cadena que se basa en el formato de fecha especificado en la sesión.</p> <p>Especifique una cadena de formato TO_CHAR válida. La cadena de formato define el formato del valor devuelto, no el formato de los valores del argumento date. Si omite la cadena de formato, la función devuelve una cadena que se basa en el formato de fecha especificado en la configuración de la asignación.</p>

Valor devuelto

Cadena.

NULL si el valor pasado a la función es NULL.

Ejemplos

La siguiente expresión convierte las fechas del puerto DATE_PROMISED en texto en el formato MON DD YYYY:

```
TO_CHAR( DATE_PROMISED, 'MON DD YYYY' )
```

DATE_PROMISED	RETURN VALUE
Apr 1 1998 12:00:10AM	'Apr 01 1998'
Feb 22 1998 01:31:10PM	'Feb 22 1998'
Oct 24 1998 02:12:30PM	'Oct 24 1998'
NULL	NULL

Si omite el argumento *format*, TO_CHAR devuelve una cadena en el formato de fecha especificado en la sesión, que es, como valor predeterminado, MM/DD/YYYY HH24:MI:SS.US:

Si omite el argumento *format*, TO_CHAR devuelve una cadena en el formato de fecha especificado en la configuración de asignación, que es, como valor predeterminado, MM/DD/YYYY HH24:MI:SS.US:

```
TO_CHAR( DATE_PROMISED )
```

DATE_PROMISED	RETURN VALUE
Apr 1 1998 12:00:10AM	'04/01/1998 00:00:10.000000'
Feb 22 1998 01:31:10PM	'02/22/1998 13:31:10.000000'

DATE_PROMISED	RETURN VALUE
Oct 24 1998 02:12:30PM	'10/24/1998 14:12:30.000000'
NULL	NULL

Las siguientes expresiones devuelven el día de la semana para cada fecha de un puerto:

`TO_CHAR(DATE_PROMISED, 'D')`

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10AM	'3'
02-22-1997 01:31:10PM	'7'
10-24-1997 02:12:30PM	'6'
NULL	NULL

`TO_CHAR(DATE_PROMISED, 'DAY')`

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10AM	'Tuesday'
02-22-1997 01:31:10PM	'Saturday'
10-24-1997 02:12:30PM	'Friday'
NULL	NULL

La siguiente expresión devuelve el día del mes para cada fecha de un puerto:

`TO_CHAR(DATE_PROMISED, 'DD')`

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10AM	'01'
02-22-1997 01:31:10PM	'22'
10-24-1997 02:12:30PM	'24'
NULL	NULL

La siguiente expresión devuelve el día del año para cada fecha de un puerto:

`TO_CHAR(DATE_PROMISED, 'DDD')`

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10AM	'091'

DATE_PROMISED	RETURN VALUE
02-22-1997 01:31:10PM	'053'
10-24-1997 02:12:30PM	'297'
NULL	NULL

Las siguientes expresiones devuelven la hora del día para cada fecha de un puerto:

```
TO_CHAR( DATE_PROMISED, 'HH' )
TO_CHAR( DATE_PROMISED, 'HH12' )
```

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10AM	'12'
02-22-1997 01:31:10PM	'01'
10-24-1997 02:12:30PM	'02'
NULL	NULL

```
TO_CHAR( DATE_PROMISED, 'HH24' )
```

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10AM	'00'
02-22-1997 01:31:10PM	'13'
10-24-1997 11:12:30PM	'23'
NULL	NULL

La siguiente expresión convierte los valores de fecha en valores MJD expresados como cadenas:

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

SHIP_DATE	RETURN VALUE
Dec 31 1999 03:59:59PM	2451544
Jan 1 1900 01:02:03AM	2415021

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

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

SHIP_DATE	RETURN VALUE
12/31/1999 01:02:03AM	12/31/99

SHIP_DATE	RETURN_VALUE
09/15/1996 03:59:59PM	09/15/96
05/17/2003 12:13:14AM	05/17/03

También puede utilizar la cadena de formato SSSSS en una expresión TO_CHAR. Por ejemplo, la siguiente expresión convierte las fechas del puerto SHIP_DATE en cadenas que representan los segundos totales hasta medianoche:

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

SHIP_DATE	RETURN_VALUE
12/31/1999 01:02:03AM	3783
09/15/1996 03:59:59PM	86399

En las expresiones TO_CHAR, la cadena de formato YY produce los mismos resultados que la cadena de formato RR.

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

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

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

La siguiente expresión devuelve la semana del mes para cada fecha de un puerto:

```
TO_CHAR( DATE_PROMISED, 'W' )
```

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10AM	'01'
02-22-1997 01:31:10AM	'04'
10-24-1997 02:12:30PM	'04'
NULL	NULL

La siguiente expresión devuelve la semana del año para cada fecha de un puerto:

```
TO_CHAR( DATE_PROMISED, 'WW' )
```

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10PM	'18'

DATE_PROMISED	RETURN VALUE
02-22-1997 01:31:10AM	'08'
10-24-1997 02:12:30AM	'43'
NULL	NULL

Consejo

Puede combinar TO_CHAR y TO_DATE para convertir un valor numérico de un mes en el valor de texto de un mes con una función como:

```
TO_CHAR( TO_DATE( numeric_month, 'MM' ), 'MONTH' )
```

TO_CHAR (Números)

Convierte valores numéricos a cadenas de texto. TO_CHAR también convierte fechas a cadenas.

TO_CHAR convierte valores dobles en cadenas de texto de la siguiente forma:

- Convierte los valores dobles de hasta 16 dígitos en cadenas y ofrece una precisión de hasta 15 dígitos. Si se pasa un número con más de 15 dígitos, TO_CHAR redondea el número en función del decimosexto dígito y devuelve la representación de cadena del número en notación científica. Por ejemplo, el valor doble 1234567890123456 se convierte en el valor de cadena "1.23456789012346e+015".
- Devuelve la notación decimal para los números en los intervalos (-1e16, -1e-16] y [1e-16, 1e16). TO_CHAR devuelve la notación científica para los números fuera de estos intervalos. Por ejemplo, el valor doble 10842764968208837340 se convierte en el valor de cadena "1.08427649682088e+019".

TO_CHAR convierte valores decimales en cadenas de texto de la siguiente forma:

- En el modo de alta precisión, TO_CHAR convierte valores decimales de hasta 38 dígitos en cadenas. Si se pasa un valor decimal con más de 38 dígitos, TO_CHAR devuelve la notación científica para los números con más de 38 dígitos.
- En el modo de alta precisión, TO_CHAR convierte valores decimales de hasta 28 dígitos en cadenas. Si se pasa un valor decimal con más de 28 dígitos, TO_CHAR devuelve la notación científica para los números con más de 28 dígitos.
- En el modo de baja precisión, TO_CHAR trata los valores decimales como valores dobles.

Sintaxis

```
TO_CHAR( numeric_value )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. El valor numérico que desea convertir en una cadena. Se puede especificar cualquier expresión de transformación válida.

Valor de devolución

Cadena.

NULL si el valor pasado a la función es NULL.

Ejemplo de conversión de doble

La siguiente expresión convierte los valores dobles en el puerto SALES en cadenas:

```
TO_CHAR( SALES )
```

SALES	RETURN VALUE
1010.99	'1010.99'
-15.62567	'-15.62567'
10842764968208837340	'1.08427649682088e+019' (rounded based on the 16th digit and returns the value in scientific notation)
236789034569723	'236789034569723'
0	'0'
33.15	'33.15'
NULL	NULL

Ejemplo de conversión de decimal

La siguiente expresión convierte los valores decimales en el puerto SALES en cadenas en modo de alta precisión:

```
TO_CHAR( SALES )
```

SALES	RETURN VALUE
2378964536789761	'2378964536789761'
1234567890123456789012345679	'1234567890123456789012345679'
1.234578945469649345876123456	'1.234578945469649345876123456'
0.999999999999999999999999999999	'0.999999999999999999999999999999'
12345678901234567890123456799	'12345678901234567890123456799'
23456788992233456678458934567123465239	'23456788992233456678458934567123465239'
423456789012345678901234567991234567899 (mayor que 38)	'4.23456789012346e+038'

SALES	RETURN VALUE
2378964536789761	'2378964536789761'
1234567890123456789012345679	'1234567890123456789012345679'

Ejemplos

La siguiente expresión devuelve valores de fecha para las cadenas en el puerto DATE_PROMISED. TO_DATE siempre devuelve una fecha y una hora. Si se pasa una cadena que no tiene un valor de hora, la fecha devuelta siempre incluye la hora 00:00:00.000000000. Si ejecuta una sesión en el siglo veinte, el siglo será el 19. En este ejemplo, el año actual en el nodo que ejecuta el Servicio de integración de datos es 1998. El formato de datetime para la columna destino es MON DD YY HH24:MI SS, por lo que el Servicio de integración de datos trunca el valor datetime a segundos cuando lo escribe en el destino:

La siguiente expresión devuelve valores de fecha para las cadenas en el puerto DATE_PROMISED. TO_DATE siempre devuelve una fecha y una hora. Si se pasa una cadena que no tiene un valor de hora, la fecha devuelta siempre incluye la hora 00:00:00.000000000. Si ejecuta una asignación en el siglo XX, el siglo será el 19. En este ejemplo, el año actual en el nodo que ejecuta el Servicio de integración de datos es 1998. El formato de datetime para la columna destino es MON DD YY HH24:MI SS, por lo que el Servicio de integración de datos trunca el valor datetime a segundos cuando lo escribe en el destino:

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

DATE_PROMISED	RETURN VALUE
'01/22/98'	Jan 22 1998 00:00:00
'05/03/98'	May 3 1998 00:00:00
'11/10/98'	Nov 10 1998 00:00:00
'10/19/98'	Oct 19 1998 00:00:00
NULL	NULL

La siguiente expresión devuelve valores de fecha y hora para las cadenas en el puerto DATE_PROMISED. Si se pasa una cadena que no tiene un valor de hora, el Servicio de integración de datos devuelve un error. Si ejecuta una sesión en el siglo XX, el siglo será el 19. El año actual en el nodo que ejecuta el Servicio de integración de datos es 1998.

La siguiente expresión devuelve valores de fecha y hora para las cadenas en el puerto DATE_PROMISED. Si se pasa una cadena que no tiene un valor de hora, el Servicio de integración de datos devuelve un error. Si ejecuta una asignación en el siglo XX, el siglo será el 19. El año actual en el nodo que ejecuta el Servicio de integración de datos es 1998.

```
TO_DATE ( DATE_PROMISED, 'MON DD YYYY HH12:MI:SSAM' )
```

DATE_PROMISED	RETURN VALUE
'Jan 22 1998 02:14:56PM'	Jan 22 1998 02:14:56PM
'Mar 15 1998 11:11:11AM'	Mar 15 1998 11:11:11AM
'Jun 18 1998 10:10:10PM'	Jun 18 1998 10:10:10PM
'October 19 1998'	<i>Error. Integration Service skips this row.</i>
NULL	NULL

La siguiente expresión convierte cadenas en el puerto SHIP_DATE_MJD_STRING en valores de fecha:

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

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

Debido a que la cadena de formato J no incluye la porción de hora de una fecha, los valores de retorno tienen la hora establecida en 00:00:00.000000000.

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

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

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

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

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

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

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

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

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

DATE_STR	RETURN VALUE
'04/01/8'	04/01/1998 00:00:00.000000000
'08/17/5'	08/17/1995 00:00:00.000000000

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

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

DATE_STR	RETURN VALUE
'04/01/998'	04/01/1998 00:00:00.000000000
'08/17/995'	08/17/1995 00:00:00.000000000

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

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

DATE_STR	RETURN_VALUE
'12/31/1999 3783'	12/31/1999 01:02:03
'09/15/1996 86399'	09/15/1996 23:59:59

Si el destino acepta formatos de fecha diferentes, use TO_DATE y IS_DATE con la función DECODE para probar formatos aceptables. Por ejemplo:

```
DECODE( TRUE,

--test first format
  IS_DATE( CLOSE_DATE, 'MM/DD/YYYY HH24:MI:SS' ),

--if true, convert to date
  TO_DATE( CLOSE_DATE, 'MM/DD/YYYY HH24:MI:SS' ),

--test second format; if true, convert to date
  IS_DATE( CLOSE_DATE, 'MM/DD/YYYY' ), TO_DATE( CLOSE_DATE, 'MM/DD/YYYY' ),

--test third format; if true, convert to date
  IS_DATE( CLOSE_DATE, 'MON DD YYYY' ), TO_DATE( CLOSE_DATE, 'MON DD YYYY' ),

--if none of the above
  ERROR( 'NOT A VALID DATE' ) )
```

Puede combinar TO_CHAR y TO_DATE para convertir un valor numérico para un mes en el valor de texto de un mes utilizando una función como:

```
TO_CHAR( TO_DATE( numeric_month, 'MM' ), 'MONTH' )
```

TEMAS RELACIONADOS

- [“Normas y directrices para cadenas con formato de fecha” en la página 51](#)

TO_DECIMAL

Convierte una cadena o valor numérico en un valor decimal. TO_DECIMAL ignora los espacios a la izquierda.

Sintaxis

```
TO_DECIMAL( value [, scale] )
```


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

Argumento	Obligatorio / Opcional	Descripción
<i>valor</i>	Obligatorio	Debe ser un tipo de datos de cadena o numérico. Pasa los valores que se desean convertir en valores decimales. Se puede especificar cualquier expresión de transformación válida.
<i>scale</i>	Opcional	Tiene que ser un literal entero entre 0 y 28, ambos incluidos. Especifica el número de dígitos permitidos después del punto decimal. Si se omite este argumento, la función devuelve un valor con la misma escala que el valor de entrada.

Valor devuelto

Decimal de precisión y escala entre 0 y 28, ambos incluidos.

NULL si el valor pasado a la función es NULL.

Si la cadena contiene un carácter no numérico, convierte la parte numérica de la cadena hasta el primer carácter no numérico.

Si el primer carácter numérico es no numérico, devuelve 0.

Si el valor que se pasa a la función contiene datos que no son válidos para un valor decimal, el Servicio de integración de datos marca la fila como fila de error.

Nota: Si el valor devuelto es un decimal con una precisión superior a 15, puede habilitar el modo de alta precisión para asegurarse de obtener una precisión decimal de hasta 28 dígitos.

Ejemplo

Esta expresión utiliza valores del puerto IN_TAX. IN_TAX es un tipo de datos de cadena con precisión de 44 dígitos. RETURN VALUE es un tipo de datos de decimal con una precisión de 28 y una escala de 3:

```
TO_DECIMAL( IN_TAX, 3 )
```

IN_TAX	RETURN VALUE
'15.6789'	15.679
'60.2'	60.200
'118.348'	118.348
NULL	NULL
'A12.3Grove'	0
'711A1'	711
'1234567890.123'	1234567890.123
'123456789012345678901234567890.123'	Error. Integration Service skips this row.

IN_TAX	RETURN VALUE
'1234567890123456789012345678901234567890.123'	Error. Integration Service skips this row.

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.
'711A1'	Error. Integration Service skips this row.
'1234567890.123'	1234567890.123
'123456789012345678901234567890.123'	Error. Integration Service skips this row.
'1234567890123456789012345678901234567890.123'	Error. Integration Service skips this row.

TO_DECIMAL38

Convierte una cadena o valor numérico en un valor decimal. TO_DECIMAL38 ignora los espacios en blanco a la izquierda.

Sintaxis

```
TO_DECIMAL38( value [, scale] )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>valor</i>	Obligatorio	Debe ser un tipo de datos de cadena o numérico. Pasa los valores que desea convertir en valores decimales. Se puede especificar cualquier expresión de transformación válida.
<i>scale</i>	Opcional	Tiene que ser un literal entero entre 0 y 38, ambos incluidos. Especifica el número de dígitos permitidos después del punto decimal. Si se omite este argumento, la función devuelve un valor con la misma escala que el valor de entrada.

Valor devuelto

Decimal de precisión y escala entre 0 y 38, ambos incluidos.

NULL si el valor pasado a la función es NULL.

Si el valor que se pasa a la función contiene datos que no son válidos para un valor decimal, el Servicio de integración de datos marca la fila como fila de error. Por ejemplo, si pasa `TO_DECIMAL38("1234567890123456789012345678901234567890.12")`, el Servicio de integración de datos rechaza la fila.

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

Ejemplo

Esta expresión utiliza valores del puerto `IN_TAX`. `IN_TAX` es un tipo de datos de cadena con precisión de 44 dígitos. `RETURN VALUE` es un tipo de datos de decimal con una precisión de 38 y una escala de 3:

```
TO_DECIMAL38( IN_TAX, 3 )
```

IN_TAX	RETURN VALUE
'15.6789'	15.679
'60.2'	60.200
'118.348'	118.348
NULL	NULL
'A12.3Grove'	<i>Error. Integration Service skips this row.</i>
'1234567890.123'	1234567890.123
'123456789012345678901234567890.123'	123456789012345678901234567890.123
'1234567890123456789012345678901234567890.123'	<i>Error. Integration Service skips this row.</i>
'711A1'	<i>Error. Integration Service skips this row.</i>

TO_FLOAT

Convierte una cadena o un valor numérico en un número de coma flotante de doble precisión (tipo de datos Double). `TO_FLOAT` omite los espacios en blanco.

Sintaxis

```
TO_FLOAT( value )
```

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

Argumento	Obligatorio / Opcional	Descripción
<i>value</i>	Obligatorio	Debe ser un tipo de dato cadena o numérico. Pasa los valores que se van a convertir en valores dobles. Puede introducir cualquier expresión de transformación válida.

Valor de retorno

Valor doble.

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

Se devuelve 0 si el valor del puerto está en blanco o es un carácter no numérico.

Si el valor que se pasa a la función contiene datos que no son válidos para un valor float, el servicio de integración de datos marca la fila como fila de error o no puede realizar la asignación.

Ejemplo

Esta expresión usa valores del puerto IN_TAX:

```
TO_FLOAT( IN_TAX )
```

IN_TAX	RETURN VALUE
'15.6789'	15.6789
'60.2'	60.2
'118.348'	118.348
NULL	NULL
'A12.3Grove'	0
'A12.3Grove'	<i>Error. Integration Service skips this row.</i>

TO_INTEGER

Convierte una cadena o valor numérico a un número entero. La sintaxis de TO_INTEGER contiene un argumento opcional que puede elegir para redondear el número al entero más cercano o truncar la parte decimal. TO_INTEGER ignora los espacios a la izquierda.

Sintaxis

```
TO_INTEGER( value [, flag] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>value</i>	Obligatorio	Tipo de dato de cadena o numérico. Pasa el valor que desea convertir a un número entero. Puede introducir cualquier expresión de transformación válida.
<i>flag</i>	Opcional	Marca que especifica si trunca o redondear la parte decimal. La marca debe ser un literal entero o las constantes TRUE o FALSE. TO_INTEGER trunca la parte decimal cuando la marca es TRUE o un número distinto de 0. TO_INTEGER redondea el valor al entero más cercano si la marca es FALSE o 0, o si se omite este argumento.

Valor de retorno

Entero.

NULL si el valor pasado a la función es NULL.

0 si el valor pasado a la función contiene caracteres alfanuméricos.

Si el valor que se pasa a la función contiene datos que no son válidos para un valor integer, el servicio de integración de datos marca la fila como fila de error o no puede realizar la asignación.

Ejemplos

Las expresiones siguientes utilizan valores del puerto IN_TAX. El Servicio de integración de datos muestra un error cuando la conversión provoca un desbordamiento numérico:

```
TO_INTEGER( IN_TAX, TRUE )
```

IN_TAX	RETURN VALUE
'15.6789'	15
'60.2'	60
'118.348'	118
'5,000,000,000'	<i>Error. Integration Service skips this row.</i>
NULL	NULL
'A12.3Grove'	0
'A12.3Grove'	<i>Error. Integration Service skips this row.</i>
' 123.87'	123

IN_TAX	RETURN VALUE
'-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'	<i>Error. Integration Service skips this row.</i>
NULL	NULL
'A12.3Grove'	0
'A12.3Grove'	<i>Error. Integration Service skips this row.</i>
' 123.87'	124
'-15.6789'	-16
'-15.23'	-15

TO_TIMESTAMP_TZ

Convierte una cadena en marca de tiempo con valor de zona horaria. La función devuelve el tipo de datos de marca de tiempo con zona horaria. Utilice las cadenas de formato TO_TIMESTAMP_TZ para especificar el formato de las cadenas origen.

Sintaxis

```
TO_TIMESTAMP_TZ ( String , [format] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>String</i>	Obligatorio	Debe ser un tipo de datos de cadena. Pasa los valores que desea convertir en marca de tiempo con zona horaria. Se puede especificar cualquier expresión de transformación válida. La cadena debe ser una cadena de caracteres.
<i>format</i>	Opcional	Especifique una cadena de formato TO_TIMESTAMP_TZ válida. La cadena de formato debe coincidir con las partes del argumento string. Por ejemplo, si se pasa la cadena "Mar 15 1997 12:43:10AM", debe utilizar la cadena de formato "MON DD YYYY HH12:MI:SSAM TZR". Si no especifica la cadena de formato, la función utiliza el formato de fecha y hora predeterminado en el cuadro de diálogo Ejecutar configuraciones.

Valor devuelto

Devuelve una marca de tiempo con el tipo de datos de zona horaria.

NULL si la entrada es un valor nulo.

Si el valor que se pasa a la función contiene datos que no son válidos para un valor de marca de tiempo con zona horaria, el Servicio de integración de datos marca la fila como fila de error o no realiza la asignación.

Ejemplo

INPUT VALUE	RETURN VALUE
'1947-08-05 10:45:00.221111000 AM America/Los_Angeles', 'YYYY-MM-DD HH:MI:SS.NS AM TZR'	Devuelve un tipo de datos de marca de tiempo con zona horaria con los siguientes datos: '1947-08-05 10:45:00.221111000 AM AMERICA/LOS_ANGELES'
'1947-08-05 10:45:00.221111000 AM America/Los_Angeles', 'YYYY-MM-DD HH:MI:SS.NS AM'	Devuelve un tipo de datos de marca de tiempo con zona horaria sin especificar la región de zona horaria en el formato de región de zona horaria: '1947-08-05 10:45:00.221111000 AM AMERICA/LOS_ANGELES'
'1947-08-05 10:45:00.221111000 AM America/Los_Angeles'	Devuelve un tipo de datos de marca de tiempo con zona horaria sin especificar el formato de marca de tiempo con zona horaria. '1947-08-05 10:45:00.221111000 AM AMERICA/LOS_ANGELES'
	El formato de fecha y hora predeterminado del cuadro de diálogo Ejecutar configuraciones se utiliza cuando el formato no se especifica en el nivel de función. Formato de fecha y hora predeterminado: "YYYY-MM-DD HH:MI:SS.NS AM TZR"
'1947-08-05 10:45:00.221111000 AM America/Los_Angeles', 'MM-DD-YYYY HH:MI:SS.NS AM'	Si los datos de una marca de tiempo con zona horaria no coinciden con el formato dado, aparece el siguiente error: Process row failed for function [TO_TIMESTAMP_TZ]: Failed to convert the string to timestamp with time zone value. Verify that the specified date format string is valid. Verify that the timestamp with time zone string used in the first argument is compatible with the specified date format.

TRUNC (Fechas)

Además, trunca fechas hasta un año, mes, día, hora, minuto, segundo, milisegundo o microsegundo determinados. También se puede usar TRUNC para truncar números.

Se pueden truncar las siguientes partes de fechas:

- **Año.** Si se trunca la parte del año de la fecha, la función devuelve Ene 1 del año introducido, con la hora fijada en 00:00:00.000000000. Por ejemplo, la siguiente expresión devuelve 1/1/1997 00:00:00.000000000:
`TRUNC(12/1/1997 3:10:15, 'YY')`
- **Mes.** Si se trunca la parte del mes de la fecha, la función devuelve el primer día del mes, con la hora fijada en 00:00:00.000000000. Por ejemplo, la siguiente expresión devuelve 01/04/1997 00:00:00.000000000:
`TRUNC(4/15/1997 12:15:00, 'MM')`
- **Día.** Si se trunca la parte del día de la fecha, la función devuelve la fecha, con la hora fijada en 00:00:00.000000000. Por ejemplo, la siguiente expresión devuelve 13/06/1997 00:00:00.000000000:
`TRUNC(6/13/1997 2:30:45, 'DD')`
- **Hora.** Si se trunca la parte de la hora de una fecha, la función devuelve la fecha, con los minutos, segundos y subsegundos fijados en 0. Por ejemplo, la siguiente expresión devuelve 01/04/1997 11:00:00.000000000:
`TRUNC(4/1/1997 11:29:35, 'HH')`
- **Minuto.** Si se trunca la parte del minuto de una fecha, la función devuelve la fecha, con los segundos y los subsegundos fijados en 0. Por ejemplo, la siguiente expresión devuelve 22/05/1997 10:15:00.000000000:
`TRUNC(5/22/1997 10:15:29, 'MI')`
- **Segundo.** Si se trunca la segunda porción de una fecha, la función devuelve la fecha, con los milisegundos fijados en 0. Por ejemplo, la siguiente expresión devuelve 22/05/1997 10:15:29.000000000:
`TRUNC(5/22/1997 10:15:29.135, 'SS')`
- **Milisegundo.** Si se trunca la porción de milisegundos de una fecha, la función devuelve la fecha, con los microsegundos fijados en 0. Por ejemplo, la siguiente expresión devuelve 22/05/1997 10:15:30.135000000:
`TRUNC(5/22/1997 10:15:30.135235, 'MS')`
- **Microsegundo.** Si se trunca la porción de microsegundos de una fecha, la función devuelve la fecha, con los nanosegundos fijados en 0. Por ejemplo, la siguiente expresión devuelve 22/05/1997 10:15:30.135235000:
`TRUNC(5/22/1997 10:15:29.135235478, 'US')`

Sintaxis

```
TRUNC( date [,format] )
```


En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>date</i>	Obligatorio	Tipo de datos Fecha/Hora. Los valores de fecha que se desea trunca. Se puede introducir cualquier expresión de transformación válida que dé como valor una fecha.
<i>format</i>	Opcional	Introduzca una cadena de formato válida. La cadena de formato no es sensible a mayúsculas y minúsculas. Si se omite la cadena de formato, la función trunca la porción de la hora, fijándola en 00:00:00.000000000.

Valor de retorno

Fecha.

NULL si el valor pasado a la función es NULL.

Ejemplos

Las siguientes expresiones devuelven la porción del años de las fechas en el puerto DATE_SHIPPED:

```
TRUNC( DATE_SHIPPED, 'Y' )
TRUNC( DATE_SHIPPED, 'YY' )
TRUNC( DATE_SHIPPED, 'YYY' )
TRUNC( DATE_SHIPPED, 'YYYY' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 1 1998 12:00:00.000000000
Apr 19 1998 1:31:20PM	Jan 1 1998 12:00:00.000000000
Jun 20 1998 3:50:04AM	Jan 1 1998 12:00:00.000000000
Dec 20 1998 3:29:55PM	Jan 1 1998 12:00:00.000000000
NULL	NULL

Las siguientes expresiones devuelven la porción del mes de las fechas en el puerto DATE_SHIPPED:

```
TRUNC( DATE_SHIPPED, 'MM' )
TRUNC( DATE_SHIPPED, 'MON' )
TRUNC( DATE_SHIPPED, 'MONTH' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 1 1998 12:00:00.000000000AM
Apr 19 1998 1:31:20PM	Apr 1 1998 12:00:00.000000000AM
Jun 20 1998 3:50:04AM	Jun 1 1998 12:00:00.000000000AM
Dec 20 1998 3:29:55PM	Dec 1 1998 12:00:00.000000000AM
NULL	NULL

Las siguientes expresiones devuelven la porción del día de las fechas en el puerto DATE_SHIPPED:

```
TRUNC( DATE_SHIPPED, 'D' )
TRUNC( DATE_SHIPPED, 'DD' )
TRUNC( DATE_SHIPPED, 'DDD' )
TRUNC( DATE_SHIPPED, 'DY' )
TRUNC( DATE_SHIPPED, 'DAY' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 15 1998 12:00:00.000000000AM
Apr 19 1998 1:31:20PM	Apr 19 1998 12:00:00.000000000AM
Jun 20 1998 3:50:04AM	Jun 20 1998 12:00:00.000000000AM
Dec 20 1998 3:29:55PM	Dec 20 1998 12:00:00.000000000AM
Dec 31 1998 11:59:59PM	Dec 31 1998 12:00:00.000000000AM
NULL	NULL

Las siguientes expresiones devuelven la porción de la hora de las fechas en el puerto DATE_SHIPPED:

```
TRUNC( DATE_SHIPPED, 'HH' )
TRUNC( DATE_SHIPPED, 'HH12' )
TRUNC( DATE_SHIPPED, 'HH24' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:31AM	Jan 15 1998 2:00:00.000000000AM
Apr 19 1998 1:31:20PM	Apr 19 1998 1:00:00.000000000PM
Jun 20 1998 3:50:04AM	Jun 20 1998 3:00:00.000000000AM
Dec 20 1998 3:29:55PM	Dec 20 1998 3:00:00.000000000PM
Dec 31 1998 11:59:59PM	Dec 31 1998 11:00:00.000000000AM
NULL	NULL

Las siguientes expresiones devuelven la porción del minuto de las fechas en el puerto DATE_SHIPPED:

```
TRUNC( DATE_SHIPPED, 'MI' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 15 1998 2:10:00.000000000AM
Apr 19 1998 1:31:20PM	Apr 19 1998 1:31:00.000000000PM
Jun 20 1998 3:50:04AM	Jun 20 1998 3:50:00.000000000AM
Dec 20 1998 3:29:55PM	Dec 20 1998 3:29:00.000000000PM
Dec 31 1998 11:59:59PM	Dec 31 1998 11:59:00.000000000PM
NULL	NULL

NULL si uno de los argumentos es NULL.

Nota: Si el valor devuelto es un decimal con una precisión superior a 15, puede habilitar el modo de alta precisión para asegurarse de obtener una precisión decimal de hasta 28 dígitos.

Ejemplos

La expresión siguiente trunca los valores del puerto Precio:

```
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
1235.99	1230.0
NULL	NULL

```
TRUNC( PRICE )
```

PRICE	RETURN VALUE
12.99	12.0
-18.99	-18.0
56.95	56.0
15.99	15.0
NULL	NULL

UPPER

Convierte a mayúsculas los caracteres de cadenas en minúsculas.

Sintaxis

```
UPPER( string )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>string</i>	Obligatorio	Tipo de datos String. Pasa los valores que se van a cambiar a texto en mayúsculas. Puede especificar cualquier expresión de transformación válida.

Valor devuelto

Cadena en mayúsculas. Si los datos contienen caracteres multibyte, el valor devuelto depende de la página de códigos y el modo de movimiento de datos del Servicio de integración de datos.

Se devuelve NULL si el valor pasado a la función es NULL.

Ejemplo

La siguiente expresión cambia todos los nombres del puerto FIRST_NAME a mayúsculas:

```
UPPER( FIRST_NAME )
```

FIRST_NAME	RETURN VALUE
Ramona	RAMONA
NULL	NULL
THOMAS	THOMAS
PierRe	PIERRE
Bernice	BERNICE

UUID4

Devuelve un valor binario de 16 bytes generado aleatoriamente que se ajusta a la variante 4 de la especificación de UUID descrita en RFC 4122. UUID4 no toma un argumento.

Sintaxis

```
UUID4()
```

Valor devuelto

Binario.

UUID4 nunca devuelve un valor nulo o un error.

UUID_UNPARSE

Convierte un valor binario de 16 bytes en una representación de cadena de 36 caracteres como se define en RFC 4122.

Sintaxis

```
UUID_UNPARSE( binary )
```

En la siguiente tabla se describe el argumento de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>binario</i>	Obligatorio	Tipo de datos binario. Cualquier valor binario de 16 bytes que desea convertir en una cadena de 36 caracteres.

Valor devuelto

Cadena de 36 caracteres.

Devuelve un resultado nulo si el argumento es nulo y un error si el argumento no es un valor binario de 16 bytes.

Ejemplo

La siguiente expresión puede devolver el valor 6948DF80-14BD-4E04-8842-7668D9C001F5:

```
UUID_UNPARSE(UUID4 ( ) )
```

VARIANCE

Devuelve la varianza de un valor que se pasa a la función. VARIANCE se utiliza para analizar datos estadísticos. Dentro de VARIANCE solamente se puede anidar una función agregada adicional y la función anidada debe devolver un tipo de datos Numérico.

Sintaxis

```
VARIANCE( numeric_value [, filter_condition ] )
```

En la siguiente tabla se describen los argumentos de este comando:

Argumento	Obligatorio / Opcional	Descripción
<i>numeric_value</i>	Obligatorio	Tipo de datos numérico. Pasa los valores para los que desea calcular una varianza. Se puede especificar cualquier expresión de transformación válida.
<i>filter_condition</i>	Opcional	Limita las filas en la búsqueda. La condición de filtro debe ser un valor numérico o dar un valor TRUE, FALSE o NULL. Se puede especificar cualquier expresión de transformación válida.

Valor devuelto

Valor doble.

NULL si todos los valores pasados a la función son NULL, o si no se ha seleccionado ninguna fila (por ejemplo, la *filter_condition* da un valor FALSE o NULL para todas las filas).

Nulos

Si un valor es NULL, VARIANCE lo omite. Sin embargo, si todos los valores pasados a la función son NULL o si no se ha seleccionado ninguna fila, VARIANCE devuelve NULL.

Nota: De forma predeterminada, el Servicio de integración de datos trata los valores nulos como NULL en las funciones agregadas. Si pasa un puerto o grupo de valores nulos completo, la función devuelve NULL. Sin embargo, cuando se configura el Servicio de integración de datos, se puede indicar la forma en que se desea manejar los valores nulos en las funciones agregadas. En las funciones agregadas se pueden tratar los valores nulos como 0 o como NULL.

Agrupar por

VARIANCE agrupa valores por grupos según los puertos definidos en la transformación, devolviendo un resultado para cada grupo.

Si no hay un grupo por puerto, VARIANCE trata todas las filas como un solo grupo, devolviendo un valor.

Ejemplo

La siguiente expresión calcula la variación para todas las filas en el puerto TOTAL_SALES:

```
VARIANCE( TOTAL_SALES )
```

TOTAL_SALES
2198.0
2256.0
3001.0
NULL
8953.0

RETURN VALUE: 10592444.6666667

CAPÍTULO 7

Creación de funciones personalizadas

Este capítulo incluye los siguientes temas:

- [Introducción a la creación de funciones personalizadas, 232](#)
- [Paso 1. Obtención de atributos de ID de repositorio, 233](#)
- [Paso 2. Creación de un archivo de encabezado, 234](#)
- [Paso 3. Creación de un archivo de implementación, 236](#)
- [Paso 4. Compilación de módulos, 245](#)
- [Paso 5. Creación de un archivo de complemento de repositorio, 247](#)
- [Paso 6. Prueba de funciones personalizadas, 251](#)
- [Instalación de funciones personalizadas, 252](#)
- [Creación de expresiones con funciones personalizadas, 253](#)

Introducción a la creación de funciones personalizadas

Las funciones personalizadas amplían la biblioteca de funciones de PowerCenter. Son funciones que se crean para utilizarlas en las expresiones de transformación y flujo. Puede crear funciones personalizadas fuera de PowerCenter con la API de funciones personalizadas. Para utilizar la API de funciones personalizadas, debe instalar la plataforma de desarrollo de Informatica desde el sitio web de la comunidad de Informatica Developer.

La API de funciones personalizadas utiliza el lenguaje de programación C. Puede compartir las funciones personalizadas con otros usuarios. Los usuarios pueden añadir las funciones para su repositorio y usarlas como función del lenguaje de transformación de PowerCenter.

En este capítulo se incluye una función de ejemplo que demuestra cómo crear y utilizar una función personalizada con optimización de inserciones. Con los pasos de este capítulo se crea la función ECHO. Esta función utiliza un argumento como entrada y devuelve el valor de entrada al usuario. El código de muestra para la función ECHO está en el directorio `\CustomFunctionAPI\samples\echo` de la instalación de la plataforma de desarrollo de Informatica.

También puede ver un ejemplo de función personalizada más complejo. La función personalizada `SampleLoanPayment` contiene funciones que no están disponibles al utilizar C. `SampleLoanPayment` está en

el directorio `\CustomFunctionAPI\samples\loanpayment` de la instalación de la plataforma de desarrollo de Informatica.

Pasos para la creación de funciones personalizadas

Realice los siguientes pasos para crear funciones personalizadas:

1. **Obtención de atributos de ID de repositorio.** Obtenga los atributos de ID de repositorio para incluirlos en el complemento de repositorio.
2. **Creación del archivo de encabezado.** Defina una o varias funciones personalizadas en el archivo de encabezado.
3. **Creación del archivo de implementación.** Defina una o varias funciones personalizadas en el archivo de implementación.
4. **Compilación de módulos.** Compile los módulos para crear DLL y bibliotecas compartidas.
5. **Creación de un archivo de complemento de repositorio.** Defina los metadatos para las funciones personalizadas.
6. **Prueba de funciones personalizadas.** Instale funciones personalizadas y úselas en una asignación y un flujo de trabajo para su comprobación.

Instalación de funciones personalizadas

Para usar funciones personalizadas, debe añadir las funciones al entorno de PowerCenter.

TEMAS RELACIONADOS

- [“Instalación de funciones personalizadas” en la página 252](#)

Paso 1. Obtención de atributos de ID de repositorio

Antes de desarrollar una función personalizada, deberá determinar los atributos de ID del repositorio del complemento del repositorio de la función personalizada. Utilice los atributos de ID del repositorio para identificar el complemento al definir los metadatos del complemento.

Para obtener los atributos de ID del repositorio, realice una de las siguientes tareas:

- Si va a distribuir las funciones personalizadas fuera de su organización, póngase en contacto con Informatica. Informatica asigna a cada complemento unos atributos de ID del repositorio únicos. Los atributos de ID del repositorio no son válidos si entran en conflicto con los de otro proveedor. Para obtener los atributos de ID del repositorio, visite <https://community.informatica.com/community/marketplace/repositoryidattributes> y haga clic en **Enviar**.
- Si solo utiliza funciones personalizadas en su organización, defina los atributos de ID del repositorio sin ponerse en contacto con Informatica. Si desarrolla un complemento para su organización que se utilizará con otros complementos en PowerCenter, asigne valores únicos a los atributos de ID del repositorio para cada complemento.

En la tabla siguiente se muestran los atributos XML que requieren valores únicos para definir un complemento:

Atributo de ID del repositorio	Descripción
ID del complemento	Determina el ID del complemento. Este valor se corresponde con el atributo de ID del elemento PLUGIN.
ID de proveedor	Identifica el proveedor que desarrolla el complemento. Este valor se corresponde con el atributo VENDORID del elemento PLUGIN.
ID del grupo de la función	Identifica el ID del grupo de la función. Este valor se corresponde con el atributo de ID del elemento FUNCTION_GROUP.
ID de la función	Determina el ID de la función. Este valor se corresponde con el atributo de ID del elemento FUNCTION.

Nota: Los atributos de ID del repositorio no son válidos si entran en conflicto unos con otros.

TEMAS RELACIONADOS

- [“El elemento PLUGIN” en la página 248](#)
- [“El elemento FUNCTION_GROUP” en la página 248](#)
- [“Elemento FUNCTION” en la página 249](#)

Paso 2. Creación de un archivo de encabezado

Cree un archivo de encabezado mediante C para declarar todas las funciones. Use un archivo de encabezado para una o varias funciones personalizadas.

En el siguiente ejemplo, se muestra el archivo de encabezado echo.h para la función personalizada ECHO:

```
#ifndef __ECHO_PLUGIN_HPP
#define __ECHO_PLUGIN_HPP

#if defined(WIN32)
    #if defined(SAMPLE_EXPR_EXPORTS)
        #define SAMPLE_EXPR_SPEC __declspec(dllexport)
    #else
        #define SAMPLE_EXPR_SPEC __declspec(dllimport)
    #endif
#else
    #define SAMPLE_EXPR_SPEC
#endif

// method to get description of Echo function
extern "C" SAMPLE_EXPR_SPEC IUNICHAR * getDescriptionEcho(IUNICHAR* ns, IUNICHAR*
sFuncName);

// method to get prototype of Echo function
extern "C" SAMPLE_EXPR_SPEC IUNICHAR * getPrototypeEcho(IUNICHAR* ns, IUNICHAR*
sFuncName);

// method to validate usage of Echo function
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS validateFunctionEcho(IUNICHAR* ns,
IUNICHAR* sFuncName,
                                IUINT32 numArgs, INFA_EXPR_OPD_METADATA** inputArgList,
```

```

        INFA_EXPR_OPD_METADATA* retValue);

//method to generate SQL code for pushdown optimization
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS pushdownFunctionEcho(IUNICHAR* sNameSpace,
        IUNICHAR* sFuncName,
        IUINT32 numArgs,
        INFA_EXPR_OPD_METADATA** inputArgList,
        EDatabaseType dbType,
        EPushdownMode pushdownMode,
        IUNICHAR** sGenSql);

// method to process row for Echo function
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_ROWSTATUS
processRowEcho(INFA_EXPR_FUNCTION_INSTANCE_HANDLE *fnInstance, IUNICHAR **errMsg);

// method to do module level initialization for Echo function
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS moduleInitEcho(INFA_EXPR_MODULE_HANDLE
*modHandle);

// method to do module level deinitialization for Echo function
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS moduleDeinitEcho(INFA_EXPR_MODULE_HANDLE
*modHandle);

// method to do function level initialization for Echo function
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS functionInitEcho(INFA_EXPR_FUNCTION_HANDLE
*funHandle);

// method to do function level deinitialization for Echo function
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS
functionDeinitEcho(INFA_EXPR_FUNCTION_HANDLE *funHandle);

// method to do function instance level initialization for Echo function
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS
functionInstInitEcho(INFA_EXPR_FUNCTION_INSTANCE_HANDLE *funInstHandle);

// method to do function instance level deinitialization for Echo function
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS
functionInstDeinitEcho(INFA_EXPR_FUNCTION_INSTANCE_HANDLE *funInstHandle);

/**
    These are all plugin callbacks, which have been implemented to get various module,
    function level interfaces
*/
// method to get plugin version
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS INFA_EXPR_GetPluginVersion(INFA_VERSION*
sdkVersion, INFA_VERSION* pluginVersion);

// method to delete the string allocated by this plugin. used for deleting the error
// messages
extern "C" SAMPLE_EXPR_SPEC void INFA_EXPR_DestroyString(IUNICHAR *);

// method to get validation interfaces
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS
INFA_EXPR_ValidateGetUserInterface( IUNICHAR* ns, IUNICHAR* sFuncName,
INFA_EXPR_VALIDATE_METHODS* functions);

// method to get module interfaces
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS
INFA_EXPR_ModuleGetUserInterface(INFA_EXPR_LIB_METHODS* functions);

// method to get function interfaces
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS
INFA_EXPR_FunctionGetUserInterface(IUNICHAR* nameSpaceName, IUNICHAR* functionName,
INFA_EXPR_FUNCTION_METHODS* functions);

// method to get function instance interfaces
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS
INFA_EXPR_FunctionInstanceGetUserInterface(IUNICHAR* nameSpaceName, IUNICHAR*
functionName, INFA_EXPR_FUNCTION_INSTANCE_METHODS* functions);

#endif

```

Paso 3. Creación de un archivo de implementación

El archivo de implementación contiene las definiciones de las funciones usadas para crear una función personalizada. Cree un archivo de implementación mediante C. Puede usar un archivo de implementación para una o varias funciones personalizadas. Además, puede usar un archivo de implementación para definir las funciones de validación y tiempo de ejecución de una función personalizada.

En el siguiente ejemplo, se muestra el archivo de implementación echo.c para la función personalizada ECHO:

```
/* *****  
 * ECHO function Procedure File  
 *  
 * This file contains code that creates the ECHO function, which the  
 * Integration Service calls during a workflow.  
 * ***** */  
  
/* Informatica ECHO function example developed using the Custom Function  
 * API.  
  
 * Filename: Echo.c  
  
 * An example of a custom function developed using PowerCenter  
 *  
 * The purpose of the ECHO function is to return the input value to the user.  
 *  
 */  
  
/* *****  
 * Includes  
 * ***** */  
#include <stdio.h>  
#include <string.h>  
#include "sdkexpr/exprsdk.h"  
  
#define SAMPLE_EXPR_EXPORTS  
#include "SampleExprPlugin.hpp"  
  
static IUNICHAR ECHO_STR[80];  
  
/* *****  
 * Functions  
 * ***** */  
  
/* *****  
 * Function: INFA_EXPR_GetPluginVersion  
  
Description: Defines the version of the plug-in. It must be the same as the  
Custom Function API version. Returns ISUCCESS if the plug-in version  
matches the Custom Function API version. Otherwise, returns IFailure.  
  
Input: sdkVersion - Current version of the Custom Function API.  
Output: pluginVersion - Set the version of the plug-in.  
Remarks: Custom Function API checks for compatibility between itself and the  
plug-in version.  
 * ***** */  
  
extern "C" SAMPLE_EXPR_SPEC  
INFA_EXPR_STATUS INFA_EXPR_GetPluginVersion(INFA_VERSION* sdkVersion, INFA_VERSION*  
pluginVersion)  
{  
    pluginVersion->m_major = 1;  
    pluginVersion->m_minor = 0;  
    pluginVersion->m_patch = 0;  
  
    INFA_EXPR_STATUS retStatus;
```

```

        retStatus.status = ISUCCESS;
        retStatus.errMsg = NULL;
        return retStatus;
    }

/*****
    Function: INFA_EXPR_DestroyString

Description: Destroys all strings the plug-in returns. For example, it
destroys error messages or the return value of other function calls, such
as getFunctionDescription. Returns no value.

Input: The pointer to the allocated string.
Output: N/A
Remarks: Frees the memory to avoid issues with multiple heaps.
*****/

extern "C" SAMPLE_EXPR_SPEC void INFA_EXPR_DestroyString(IUNICHAR *strToDelete)
{
    delete [] strToDelete;
}

/*****
    Function: INFA_EXPR_ValidateGetUserInterface

Description: Returns function pointers to the validation functions. Returns
ISUCCESS when the plug-in implemented the function. Returns IFailure
when the plug-in did not implement the function or another error occurred.

Input: Namespace and name of function.
Output: Functions. The plug-in needs to set various function pointers.
Remarks: Check the namespace and function name for validity. Set the various
function pointers appropriately.
*****/

extern "C" SAMPLE_EXPR_SPEC
    INFA_EXPR_STATUS INFA_EXPR_ValidateGetUserInterface(IUNICHAR* ns, IUNICHAR*
sFuncName,
                                                    INFA_EXPR_VALIDATE_METHODS* functions)
{
    INFA_EXPR_STATUS retStatus;
    retStatus.errMsg = NULL;

    // check function name is not null
    if (!sFuncName)
    {
        retStatus.status = IFailure;
        return retStatus;
    }

    // set the appropriate function pointers
    functions->validateFunction = validateFunctionEcho;
    functions->getFunctionDescription = getDescriptionEcho;
    functions->getFunctionPrototype = getPrototypeEcho;
    functions->pushdownFunction = pushdownFunctionEcho;

    retStatus.status = ISUCCESS;
    return retStatus;
}

/*****
    Function: INFA_EXPR_ModuleGetUserInterface

Description: Sets the function pointers for module-level interaction.
Returns ISUCCESS when functions pointers are set appropriately. Otherwise,
returns IFailure.

Input: N/A
Output: Functions. The plug-in needs to set various function pointers.
Remarks: Set the module init/deinit function pointers.
*****/

```

```

*****/

extern "C" SAMPLE_EXPR_SPEC
    INFA_EXPR_STATUS INFA_EXPR_ModuleGetUserInterface(INFA_EXPR_LIB_METHODS* functions)
{
    functions->module_init = moduleInitEcho;
    functions->module_deinit = moduleDeinitEcho;

    INFA_EXPR_STATUS retStatus;
    retStatus.status = ISUCCESS;
    retStatus.errMsg = NULL;
    return retStatus;
}

/*****
    Function: INFA_EXPR_FunctionGetUserInterface

Description: Sets the function pointers for function-level interaction.
PowerCenter calls this function for every custom function this library
implements. Returns ISUCCESS when The plugin implements this function and
sets the function pointers correctly. Otherwise, returns IFailure.

Input: Namespace and name of function.
Output: Functions. The plug-in needs to set function pointers for function
init/deinit.
Remarks: Set the function init/deinit function pointers.
*****/

extern "C" SAMPLE_EXPR_SPEC
    INFA_EXPR_STATUS INFA_EXPR_FunctionGetUserInterface(IUNICHAR* nameSpaceName,
                                                         IUNICHAR* functionName,
                                                         INFA_EXPR_FUNCTION_METHODS* functions)
{
    functions->function_init = functionInitEcho;
    functions->function_deinit = functionDeinitEcho;

    INFA_EXPR_STATUS retStatus;
    retStatus.status = ISUCCESS;
    retStatus.errMsg = NULL;
    return retStatus;
}

/*****
    Function: INFA_EXPR_FunctionInstanceGetUserInterface

Description: Sets the function pointers for function instance-level
interaction. PowerCenter calls this function for every custom function this
library implements. Returns ISUCCESS when The plugin implements this
function and sets the function pointers correctly. Otherwise, returns
IFailure.

Input: Namespace and name of function.
Output: Functions. The plug-in needs to set function pointers for instance
init/deinit/processrow.
Remarks: Set the function instance init/deinit/processrow function pointers.
*****/

extern "C" SAMPLE_EXPR_SPEC
    INFA_EXPR_STATUS INFA_EXPR_FunctionInstanceGetUserInterface(IUNICHAR* nameSpaceName,
                                                                IUNICHAR* functionName,
                                                                INFA_EXPR_FUNCTION_INSTANCE_METHODS* functions)
{
    functions->fnInstance_init = functionInstInitEcho;
    functions->fnInstance_processRow = processRowEcho;
    functions->fnInstance_deinit = functionInstDeinitEcho;

    INFA_EXPR_STATUS retStatus;
    retStatus.status = ISUCCESS;
    retStatus.errMsg = NULL;
    return retStatus;
}

```

```

}

/*****
Function: INFA_EXPR_getDescriptionEcho

Description: Gets the description of the ECHO function. It calls
destroyString to delete the arguments from memory when usage is complete.
The return value must be a null-terminated string.

Input: Namespace and name of function.
Output: Description of the function.
Remarks: Returns the description of function. The Custom Functions API calls
destroy string to free the allocated memory.
*****/

extern "C" SAMPLE_EXPR_SPEC
IUNICHAR * getDescriptionEcho(IUNICHAR* ns, IUNICHAR* sFuncName)
{
    static IUNICHAR *uniDesc = NULL;
    const char *description = "Echoes the input";

    if (uniDesc)
        return uniDesc;

    int i, len;
    len = strlen(description);
    uniDesc = new IUNICHAR[2*len+2];
    for (i=0; i<len; i++)
    {
        uniDesc[i] = description[i];
    }
    uniDesc[i] = 0;
    return uniDesc;
}

/*****
Function: INFA_EXPR_getPrototypeEcho

Description: Gets the arguments of the ECHO function in the Expression
Editor. It calls destroyString to delete the arguments from memory when
usage is complete. The return value must be a null-terminated string. The
function returns NULL if there is no value for the arguments.

Input: Namespace and name of the function.
Output: Prototype of the function
Remarks: Returns the prototype of function. The Custom Functions API calls
destroy string to free the allocated memory.
*****/

extern "C" SAMPLE_EXPR_SPEC
IUNICHAR * getPrototypeEcho(IUNICHAR* ns, IUNICHAR* sFuncName)
{
    static IUNICHAR *uniProt = NULL;
    const char *prototype = "Echo(x), where x can be any type, returns x";

    if (uniProt)
        return uniProt;

    int i, len;
    len = strlen(prototype);
    uniProt = new IUNICHAR[2*len+2];
    for (i=0; i<len; i++)
    {
        uniProt[i] = prototype[i];
    }
    uniProt[i] = 0;
    return uniProt;
}

/*****

```

Function: validateFunctionEcho

Description: Validates the arguments in the ECHO function. Provides the name, datatype, precision, and scale of the arguments in the ECHO function. Provides the datatype of the return value of the ECHO function. PowerCenter calls this function once for each instance of the ECHO function used in a mapping or workflow. Returns ISUCCESS when function usage is valid as per the syntax.

The ECHO function takes exactly one argument of any datatype. The return datatype is the same as the input datatype, because the function echoes the input. Otherwise, returns IFailure.

Input: Namespace and name of the function, the number of arguments being passed, and the metadata (datatype, scale, precision) of each argument.

Output: retValue. Set the metadata for return type.

Remarks: Called by the Custom Functions API to validate the usage of the function and the input argument metadata to be passed. The plug-in needs to verify the number of arguments for this function, the expected metadata for each argument, etc. The plug-in can optionally change the expected datatype of the input arguments. The plug-in needs to set the return type metadata. The plugin can specify if the return value of this function is constant, depending on whether or not all input arguments are constant.

*****/

```
extern "C" SAMPLE_EXPR_SPEC
INFA_EXPR_STATUS validateFunctionEcho(IUNICHAR* ns, IUNICHAR* sFuncName,
                                     IUINT32 numArgs,
                                     INFA_EXPR_OPD_METADATA** inputArgList,
                                     INFA_EXPR_OPD_METADATA* retValue)
{
    INFA_EXPR_STATUS exprStatus;

    // Check number of arguments.
    if (numArgs != 1)
    {
        static const char *err = "Echo function takes one argument.";
        IUNICHAR *errMsg = NULL;

        unsigned int len = strlen(err);
        errMsg = new IUNICHAR[2*len+2];
        unsigned int i;
        for (i=0; i<len; i++)
        {
            errMsg[i] = err[i];
        }
        errMsg[i] = 0;

        exprStatus.status = IFailure;
        exprStatus.errMsg = errMsg;
        return exprStatus;
    }

    // This is an echo function.
    // It returns the input value.
    retValue->datatype = inputArgList[0]->datatype;
    retValue->precision = inputArgList[0]->precision;
    retValue->scale = inputArgList[0]->scale;

    // If the input value is constant,
    // the return value is also constant.
    if (inputArgList[0]->isValueConstant)
        retValue->isValueConstant = ITRUE;
    else
        retValue->isValueConstant = IFALSE;

    exprStatus.status = ISUCCESS;
    return exprStatus;
}
```



```

/*****
Function: processRowEcho

Description: Called when an input row is available to an ECHO function
instance. The data for the input arguments of the ECHO function is bound and
accessed through fnInstance->inputOPDHandles. Set the data, length, and
indicator for the output and return ports in fnInstance->retHandle.
PowerCenter calls the function-level initialization function before calling
this function.

Returns INFA_ROWSUCCESS when the function successfully processes the row of
data. Returns INFA_ROWERROR when the function encounters an error for the row
of data. The Integration Service increments the internal error count.
Only returns this value when the data access mode is row. Returns
INFA_FATALERROR when the function encounters a fatal error for the row of
data or the block of data. The Integration Service fails the session.

Input: Function instance handle, which has the input data.
Output: return value
Remarks: The plug-in needs to get various input arguments from the function
instance handle, perform calculations, and set the return value.
*****/

extern "C" SAMPLE_EXPR_SPEC
INFA_EXPR_ROWSTATUS processRowEcho(INFA_EXPR_FUNCTION_INSTANCE_HANDLE *fnInstance,
IUNICHAR **errMsg)
{
    INFA_EXPR_OPD_RUNTIME_HANDLE* arg1 = fnInstance->inputOPDHandles[0];
    INFA_EXPR_OPD_RUNTIME_HANDLE* retHandle = fnInstance->retHandle;

    // Check if the input argument has a null indicator.
    // If yes, the return value is also null.
    if (INFA_EXPR_GetIndicator(arg1) == INFA_EXPR_NULL_DATA)
    {
        INFA_EXPR_SetIndicator(retHandle, INFA_EXPR_NULL_DATA);
        return INFA_EXPR_SUCCESS;
    }

    short sval;
    long lval;
    int ival;
    char *strval;
    IUNICHAR *ustrval;
    void *rawval;
    float fval;
    double dval;
    INFA_EXPR_DATE *infaDate = NULL;
    int len;

    // Depending on the datatype,
    // get the input argument
    // and set the same value in the return value.
    // Also, set the same indicator.
    switch (arg1->pOPDMetadata->datatype)
    {
        case eCTYPE_SHORT:
            sval = INFA_EXPR_GetShort(arg1);
            INFA_EXPR_SetShort(retHandle, sval);
            INFA_EXPR_SetIndicator(retHandle, INFA_EXPR_GetIndicator(arg1));
            break;

        case eCTYPE_LONG:
        case eCTYPE_LONG64:
            lval = INFA_EXPR_GetLong(arg1);
            INFA_EXPR_SetLong(retHandle, lval);
            INFA_EXPR_SetIndicator(retHandle, INFA_EXPR_GetIndicator(arg1));
            break;

        case eCTYPE_INT32:

```

```

        ival = INFA_EXPR_GetInt(arg1);
        INFA_EXPR_SetInt(retHandle, ival);
        INFA_EXPR_SetIndicator(retHandle, INFA_EXPR_GetIndicator(arg1));
        break;

    case eCTYPE_CHAR:
        strval = INFA_EXPR_GetString(arg1);
        len = INFA_EXPR_GetLength(arg1);
        strcpy((char *)retHandle->pUserDefinedPtr, strval);
        INFA_EXPR_SetString(retHandle, retHandle->pUserDefinedPtr);
        INFA_EXPR_SetLength(retHandle, INFA_EXPR_GetLength(arg1));
        INFA_EXPR_SetIndicator(retHandle, INFA_EXPR_GetIndicator(arg1));
        break;

    case eCTYPE_RAW:
        rawval = INFA_EXPR_GetRaw(arg1);
        len = INFA_EXPR_GetLength(arg1);
        memcpy(retHandle->pUserDefinedPtr, rawval, len);
        INFA_EXPR_SetRaw(retHandle, retHandle->pUserDefinedPtr);
        INFA_EXPR_SetLength(retHandle, len);
        INFA_EXPR_SetIndicator(retHandle, INFA_EXPR_GetIndicator(arg1));
        break;

    case eCTYPE_UNICHAR:
        ustrval = INFA_EXPR_GetUniString(arg1);
        len = INFA_EXPR_GetLength(arg1);
        memcpy(retHandle->pUserDefinedPtr, ustrval, 2*(len+1));
        INFA_EXPR_SetUniString(retHandle, retHandle->pUserDefinedPtr);
        INFA_EXPR_SetLength(retHandle, len);
        INFA_EXPR_SetIndicator(retHandle, INFA_EXPR_GetIndicator(arg1));
        break;

    case eCTYPE_TIME:
        infaDate = INFA_EXPR_GetDate(arg1);
        *((INFA_EXPR_DATE *)retHandle->pUserDefinedPtr) = *infaDate;
        INFA_EXPR_SetDate(retHandle, retHandle->pUserDefinedPtr);
        INFA_EXPR_SetIndicator(retHandle, INFA_EXPR_GetIndicator(arg1));
        break;

    case eCTYPE_FLOAT:
        fval = INFA_EXPR_GetFloat(arg1);
        INFA_EXPR_SetFloat(retHandle, fval);
        INFA_EXPR_SetIndicator(retHandle, INFA_EXPR_GetIndicator(arg1));
        break;

    case eCTYPE_DOUBLE:
        dval = INFA_EXPR_GetDouble(arg1);
        INFA_EXPR_SetDouble(retHandle, dval);
        INFA_EXPR_SetIndicator(retHandle, INFA_EXPR_GetIndicator(arg1));
        break;

    default:
        return INFA_EXPR_ROWERROR;
        break;
}
return INFA_EXPR_SUCCESS;
}

```

```

/*****
Function: moduleInitEcho

```

Description: Called once for each module to initialize any global data structure in the function. Called before calling any function-level functions. Returns ISUCCESS when module initialization is successful. Otherwise, returns IFailure.

Input: module handle

Output: status

Remarks: The plug-in can optionally implement this method for one-time initialization.

```

*****/

```

```

extern "C" SAMPLE_EXPR_SPEC
    INFA_EXPR_STATUS moduleInitEcho(INFA_EXPR_MODULE_HANDLE *modHandle)
{
    INFA_EXPR_STATUS exprStatus;

    // initialize the ECHO_STR
    const char *fnName = "Echo";
    int len = strlen(fnName);
    int i;
    for (i=0;i<len;i++)
        ECHO_STR[i] = fnName[i];

    exprStatus.status = ISUCCESS;
    return exprStatus;
}

/*****
    Function: moduleDeinitEcho

Description: Called once for each module to deinitialize any data structure
in this function. Called after all function-level interactions are complete.
Returns ISUCCESS when module deinitialization is successful. Otherwise,
returns IFailure.

Input: module handle
Output: status
Remarks: The plug-in can optionally implement this method for one-time
deinitialization.
*****/

extern "C" SAMPLE_EXPR_SPEC
    INFA_EXPR_STATUS moduleDeinitEcho(INFA_EXPR_MODULE_HANDLE *modHandle)
{
    INFA_EXPR_STATUS exprStatus;
    exprStatus.status = ISUCCESS;
    return exprStatus;
}

/*****
    Function: functionInitEcho

Description: Called once for each custom function to initialize any
structure related to the custom function. Module-level initialization
function is called before this function. Returns ISUCCESS when function
init is successful. Otherwise, returns IFailure.

Input: function handle
Output: status
Remarks: The plug-in can optionally implement this method for one-time function
initialization.
*****/

extern "C" SAMPLE_EXPR_SPEC
    INFA_EXPR_STATUS functionInitEcho(INFA_EXPR_FUNCTION_HANDLE *funHandle)
{
    INFA_EXPR_STATUS exprStatus;
    exprStatus.status = ISUCCESS;
    return exprStatus;
}

/*****
    Function: functionDeinitEcho

Description: Called once for each function level to deinitialize any
structure related to the ECHO function. Returns ISUCCESS when function deinit
is successful. Otherwise, returns IFailure.

Input: function handle

```

Output: status
 Remarks: The plug-in can optionally implement this method for one-time function deinitialization.

```

*****/

extern "C" SAMPLE_EXPR_SPEC
    INFA_EXPR_STATUS functionDeinitEcho(INFA_EXPR_FUNCTION_HANDLE *funHandle)
{
    INFA_EXPR_STATUS exprStatus;
    exprStatus.status = ISUCCESS;
    return exprStatus;
}

/*****
    Function: functionInstInitEcho
  *****/

```

Description: Called once for each custom function instance to initialize any structure related to the an instance of the ECHO function. If there are two instances of ECHO in a mapping or workflow, PowerCenter calls this function twice. PowerCenter calls the module-level initialization function before calling this function. Returns ISUCCESS when function instance initialization is successful. Otherwise, returns IFailure.

Input: function instance handle
 Output: status
 Remarks: The plug-in can optionally implement this method for one-time function instance initialization.

```

*****/

extern "C" SAMPLE_EXPR_SPEC
    INFA_EXPR_STATUS functionInstInitEcho(INFA_EXPR_FUNCTION_INSTANCE_HANDLE
    *funInstHandle)
{
    INFA_EXPR_STATUS exprStatus;
    exprStatus.status = ISUCCESS;

    INFA_EXPR_OPD_RUNTIME_HANDLE *retHandle = funInstHandle->retHandle;

    // Allocate memory depending on the datatype.
    if (retHandle->pOPDMetadata->datatype == eCTYPE_CHAR)
        retHandle->pUserDefinedPtr = new char[retHandle->pOPDMetadata->precision+1];
    else if (retHandle->pOPDMetadata->datatype == eCTYPE_UNICHAR)
        retHandle->pUserDefinedPtr = new IUNICHAR[retHandle->pOPDMetadata->precision+1];
    else if (retHandle->pOPDMetadata->datatype == eCTYPE_RAW)
        retHandle->pUserDefinedPtr = new unsigned char[retHandle->pOPDMetadata->precision];
    else if (retHandle->pOPDMetadata->datatype == eCTYPE_TIME)
        retHandle->pUserDefinedPtr = new INFA_EXPR_DATE();
    return exprStatus;
}

/*****
    Function: functionInstDeinitEcho
  *****/

```

Description: Called once for each function level during deinitialization. Can deinitialize any structure related to the ECHO function. Returns ISUCCESS when deinitialization is successful. Otherwise, returns IFailure.

Input: function instance handle
 Output: status
 Remarks: The plug-in can optionally implement this method for one-time function instance deinitialization.

```

*****/

extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS
functionInstDeinitEcho(INFA_EXPR_FUNCTION_INSTANCE_HANDLE *funInstHandle)
{
    INFA_EXPR_STATUS exprStatus;
    exprStatus.status = ISUCCESS;
    INFA_EXPR_OPD_RUNTIME_HANDLE *retHandle = funInstHandle->retHandle;

```

```

        if (retHandle->pOPDMetadata->datatype == eCTYPE_CHAR)
            delete [] (char *)retHandle->pUserDefinedPtr;
        else if (retHandle->pOPDMetadata->datatype == eCTYPE_UNICHAR)
            delete [] (IUNICHAR *)retHandle->pUserDefinedPtr;
        else if (retHandle->pOPDMetadata->datatype == eCTYPE_RAW)
            delete [] (unsigned char *)retHandle->pUserDefinedPtr;
        else if (retHandle->pOPDMetadata->datatype == eCTYPE_TIME)
            delete (INFA_EXPR_DATE *)retHandle->pUserDefinedPtr;
        return exprStatus;
    }

/*****
Function: pushdownFunctionEcho

Description: Method to generate SQL code for pushdown optimization.

Input:  Namespace and name of the function, the number of arguments being passed,
        and the metadata (datatype, scale, precision) of each argument, database type,
        Pushdown mode.
Output: Generated SQL.
Remarks: The plug-in can optionally implement this method to enable pushdown
optimization.
*****/

//method to generate SQL code for pushdown optimization
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS pushdownFunctionEcho(IUNICHAR* sNameSpace,
                                                                    IUNICHAR* sFuncName,
                                                                    IUINT32 numArgs,
                                                                    INFA_EXPR_OPD_METADATA** inputArgList,
                                                                    EDatabaseType dbType,
                                                                    EPushdownMode pushdownMode,
                                                                    IUNICHAR** sGenSql)
{
    INFA_EXPR_STATUS retStatus;
    static const char *sql_str = "{1}";

    // Construct the SQL: "{1}"
    unsigned int len = strlen(sql_str);

    IUNICHAR *pGenSql = new IUNICHAR[len+1];
    unsigned int i;

    for (i=0; i<len; i++)
    {
        pGenSql[i] = sql_str[i];
    }

    pGenSql[len] = 0;

    // Return the generated SQL
    *sGenSql = pGenSql;

    retStatus.status = ISUCCESS;
    retStatus.errMsg = NULL;
    return retStatus;
}

```

Paso 4. Compilación de módulos

Puede compilar los módulos en Windows o UNIX. Compile un módulo para cada plataforma en la que se ejecute PowerCenter. Debe compilar un módulo en Windows, ya que el cliente de PowerCenter reside en

Windows. Además, es posible que deba compilar módulos en UNIX o Linux según el nodo que hospede el servicio de integración.

En la siguiente tabla, se incluyen los nombres de los archivos de biblioteca de cada plataforma para la compilación del módulo:

Plataforma	Nombre de archivo de módulo
Windows	<module_identifier>.dll
AIX	lib<module_identifier>.a
Linux	lib<module_identifier>.so
Solaris	lib<module_identifier>.so

Debe declarar estos módulos en el archivo XML de complemento de repositorio.

TEMAS RELACIONADOS

- [“Paso 5. Creación de un archivo de complemento de repositorio” en la página 247](#)

Generación del módulo en Windows

En Windows, se puede utilizar Microsoft Visual C++ para generar el módulo.

Para generar el módulo en Windows:

1. Iniciar Visual C++.
2. Haga clic en Archivo > Nuevo.
3. En el cuadro de diálogo Nuevo, haga clic en la ficha Proyectos y seleccione la opción Win32 Dynamic-Link Library.
4. Introduzca su ubicación.

En el ejemplo Echo, introduzca el directorio siguiente:

```
<Informatica Development Platform installation directory>\CustomFunctionAPI\samples  
\echo
```

5. Especifique el nombre del proyecto:
Debe utilizar el nombre del módulo especificado para la función personalizada como el nombre del proyecto. En el ejemplo Echo, introduzca EchoDemo.
6. Haga clic en Aceptar.
Visual C++ crea un asistente para definir los componentes del proyecto.
7. En el asistente, seleccione un proyecto de DLL vacío y haga clic en Finalizar. Haga clic en Aceptar en el cuadro de diálogo Información del nuevo proyecto.
Visual C++ crea los archivos del proyecto en el directorio especificado.
8. Haga clic en Proyecto > Agregar a proyecto > Archivos.
9. Suba un nivel de directorio. Este directorio contiene el procedimiento de los archivos que ha creado. Seleccione todos los archivos .c y haga clic en Aceptar.
En el ejemplo Echo, agregue el archivo Echo.c:
10. Haga clic en Proyecto > Configuración.
11. Haga clic en la ficha C/C++, y seleccione Preprocesador en el campo Categoría.

12. En el campo Directorios Include adicionales, escriba la ruta siguiente y haga clic en Aceptar:

```
..; <Informatica Development Platform installation directory>\CustomFunctionAPI  
  \samples\echo; ...
```

13. Haga clic en Generar > Generar <nombre de módulo>.dll o presione F7 para generar el proyecto.

Visual C++ crea el archivo DLL y lo coloca en el directorio debug o release bajo el directorio del proyecto.

Compilación de un módulo en UNIX

En UNIX, puede usar cualquier compilador de C para compilar un módulo.

Para compilar un módulo en UNIX:

1. Establezca la variable de entorno INFA_HOME en el directorio de instalación del servicio de integración de PowerCenter.

Nota: Si especifica una ruta de acceso al directorio incorrecta para la variable de entorno INFA_HOME, el servicio de integración de PowerCenter no se puede iniciar.

Reinicie el nodo para aplicar los cambios.

2. Especifique un comando de la siguiente tabla para crear el proyecto:

Versión de UNIX	Comando
AIX (32 bits)	make -f makefile.aix
AIX (64 bits)	make -f makefile.aix64
Linux	make -f makefile.linux
Solaris	make -f makefile.sol

Paso 5. Creación de un archivo de complemento de repositorio

Cree un archivo XML para definir los metadatos de función de una o varias funciones personalizadas. Use la estructura del archivo DTD de complemento para crear o modificar el archivo XML de complemento. El archivo DTD de complemento, plugin.dtd, se almacena en el directorio del cliente de PowerCenter. Use cualquier herramienta que permita crear un archivo XML. Al crear el archivo de complemento de repositorio, especifique un nombre nuevo para el archivo.

En la siguiente figura, se muestra la estructura de plugin.dtd:

```
POWERMART  
├── REPOSITORY  
│   ├── PLUGIN  
│   │   ├── FUNCTION_GROUP  
│   │   │   ├── FUNCTION  
│   │   │   └── LIBRARY
```

El elemento PLUGIN

Utilice el elemento PLUGIN para definir los metadatos del plug-in que desea crear. Los atributos del elemento PLUGIN identifican de forma única los metadatos del complemento.

En la tabla siguiente, se muestran los atributos del elemento PLUGIN:

Atributo	Obligatorio/ Opcional	Descripción
NAME	Obligatorio	Nombre del complemento. El nombre del complemento se muestra en la ficha Complemento del servicio de repositorio de PowerCenter.
ID	Obligatorio	ID del complemento. Se utiliza para distinguir los complementos desarrollados con el mismo VENDORID.
VENDORNAME	Obligatorio	Nombre del proveedor. El nombre del proveedor se muestra en la ficha Complemento del servicio de repositorio de PowerCenter.
VENDORID	Obligatorio	ID de proveedor. Obtiene un ID de proveedor de Informatica si está desarrollando funciones personalizadas que se van a distribuir fuera de su organización. Para obtener más información, consulte "Paso 1. Obtención de atributos de ID de repositorio" en la página 233.
DESCRIPTION	Opcional	Descripción del complemento. La descripción del complemento se muestra en la ficha Complemento del servicio de repositorio de PowerCenter.
VERSION	Obligatorio	Versión del complemento. Se utiliza para realizar el seguimiento de las actualizaciones de los metadatos del complemento.

El elemento FUNCTION_GROUP

Utilice el elemento FUNCTION_GROUP para definir el grupo al que pertenece la función personalizada.

En la tabla siguiente, se muestran los atributos del elemento FUNCTION_GROUP:

Atributo	Obligatorio / Opcional	Descripción
NAME	Obligatorio	Nombre del grupo de función personalizada que desea definir. El nombre del grupo de la función se muestra en la ficha Complemento del servicio de repositorio de PowerCenter.
ID	Obligatorio	ID del grupo de la función. Obtiene un ID de grupo de la función de Informatica si está desarrollando funciones personalizadas que se van a distribuir fuera de su organización. Para obtener más información, consulte "Paso 1. Obtención de atributos de ID de repositorio" en la página 233. El ID del grupo de la función se muestra en la ficha Complemento del servicio de repositorio de PowerCenter.
COMPONENTVERSION	Obligatorio	Número de versión del grupo de la función. Realiza el seguimiento de las actualizaciones del elemento FUNCTION_GROUP.

Atributo	Obligatorio / Opcional	Descripción
DESCRIPTION	Opcional	Descripción del grupo de la función La descripción del grupo de la función se muestra en la ficha Complemento del servicio de repositorio de PowerCenter.
NAMESPACE	Obligatorio	<p>Espacio del nombre del grupo de la función El editor de expresiones muestra funciones personalizadas con el espacio del nombre en una carpeta separada en la ficha Funciones.</p> <p>Los espacios de nombres no distinguen entre mayúsculas y minúsculas. No puede utilizar el espacio del nombre "infa". Está reservado. Además el espacio del nombre no puede estar vacío.</p>

Determinación de un espacio de nombres

Puede elegir un espacio de nombres para todas las funciones creadas. No obstante, el espacio de nombres no puede estar en conflicto con el espacio de nombres de las funciones personalizadas desarrolladas por otros proveedores. Por lo tanto, debe elegir un espacio de nombres único. Por ejemplo, puede seleccionar un espacio de nombres relacionado con el nombre de su compañía, como el código bursátil.

Elemento FUNCTION

Use el elemento FUNCTION para definir las propiedades de la función personalizada.

En la siguiente tabla se muestran los atributos del elemento FUNCTION:

Atributo	Obligatorio / Opcional	Descripción
NAME	Obligatorio	Nombre de la función de otros fabricantes que va a definir.
ID	Obligatorio	ID del elemento FUNCTION. Identifica la función. Solicite un ID de función a Informática si va a desarrollar funciones personalizadas para distribuir las fuera de su organización. Para obtener más información, consulte "Paso 1. Obtención de atributos de ID de repositorio" en la página 233.
FUNCTION_CATEGORY	Opcional	Categoría de la función que va a definir. Use una de las siguientes categorías: <ul style="list-style-type: none"> - Carácter - Conversión - Limpieza de datos - Fecha - Numérica - Científica - Especial - Prueba El Editor de expresiones muestra la función personalizada según estas categorías.

Elemento LIBRARY

Use el elemento LIBRARY para especificar las bibliotecas compartidas compiladas para la función personalizada.

En la siguiente tabla, se muestran los atributos del elemento LIBRARY:

Atributo	Obligatorio / Opcional	Descripción
NAME	Obligatorio	Nombre de la biblioteca compartida compilada.
OSTYPE	Obligatorio	Sistema operativo para el que se compila la biblioteca compartida.
TYPE	Obligatorio	Tipo de biblioteca compartida. Especifique uno de los siguientes tipos: <ul style="list-style-type: none"> - VALIDATION. Biblioteca usada por el cliente de PowerCenter para recuperar la descripción de la función personalizada y validar la invocación de la función, como el tipo devuelto y el número de argumentos. - SERVER. Biblioteca usada por el servicio de integración de PowerCenter para ejecutar la llamada a la función.

Archivo XML de complemento de ejemplo

En el siguiente ejemplo, se muestra el archivo de complemento de repositorio que define la función personalizada ECHO:

```
<?xml version="1.0" encoding="us-ascii"?>
<!DOCTYPE POWERMART SYSTEM "plugin.dtd">
<POWERMART>
  <REPOSITORY CODEPAGE="us-ascii">
    <PLUGIN NAME="Echo" ID="506001" VENDORNAME="Informatica"
      VENDORID="1"
      DESCRIPTION="Plugin for Expressions from Informatica">
      <FUNCTION_GROUP ID="506002" NAME="INFA Function Group1"
        COMPONENTVERSION="1.0.0"
        DESCRIPTION="The functions group for my own Echo function"
        NAMESPACE="">
        <FUNCTION ID="506004" NAME="ECHO" FUNCTION_CATEGORY="Data Cleansing"/>
        <LIBRARY NAME="pmecho.dll" OSTYPE="NT" TYPE="VALIDATION"/>
        <LIBRARY NAME="llibpmecho.sl" OSTYPE="HPUX" TYPE="VALIDATION"/>
        <LIBRARY NAME="libpmecho.so" OSTYPE="SOLARIS" TYPE="VALIDATION"/>
        <LIBRARY NAME="libpmecho.so" OSTYPE="LINUX" TYPE="VALIDATION"/>
        <LIBRARY NAME="libpmecho.a" OSTYPE="AIX" TYPE="VALIDATION"/>
        <LIBRARY NAME="pmecho.dll" OSTYPE="NT" TYPE="SERVER"/>
        <LIBRARY NAME="llibpmecho.sl" OSTYPE="HPUX" TYPE="SERVER"/>
        <LIBRARY NAME="libpmecho.so" OSTYPE="SOLARIS" TYPE="SERVER"/>
        <LIBRARY NAME="libpmecho.so" OSTYPE="LINUX" TYPE="SERVER"/>
        <LIBRARY NAME="libpmecho.a" OSTYPE="AIX" TYPE="SERVER"/>
      </FUNCTION_GROUP>
    </PLUGIN>
  </REPOSITORY>
</POWERMART>
```

Paso 6. Prueba de funciones personalizadas

Puede probar las funciones personalizadas durante el desarrollo. Realice las siguientes tareas para probar las funciones personalizadas en PowerCenter:

- Valide el archivo XML de complemento de repositorio.
- Compruebe si las funciones personalizadas de una expresión generan datos exactos.

Para probar las funciones personalizadas, debe instalarlas en un entorno de PowerCenter.

Validación del archivo de complemento de repositorio

Para validar el archivo de complemento de repositorio, debe registrarlo en un repositorio de PowerCenter. Al registrar un archivo de complemento, el archivo DTD asociado plugin.dtd valida la estructura del archivo. El archivo debe coincidir con la estructura del archivo plugin.dtd asociado. El archivo plugin.dtd se encuentra en el directorio del cliente de PowerCenter.

Durante el desarrollo de funciones personalizadas, puede crear un archivo de complemento de repositorio y registrarlo antes de terminar de crear los archivos de encabezado e implementación para las funciones. Al registrar el archivo, debe añadir los metadatos de las funciones personalizadas, como el ID del complemento, el espacio de nombres y los nombres de las funciones. Esta información se almacena en el repositorio.

Una vez registrado el archivo de complemento de repositorio, puede desarrollar las funciones personalizadas. Cuando termine de desarrollar las funciones, vuelva a registrar el archivo de complemento de repositorio para actualizar los metadatos de las funciones personalizadas en el repositorio.

Una vez registrado el archivo de complemento de repositorio, puede ver los metadatos del complemento.

Visualización de detalles de metadatos del complemento

En la siguiente tabla, se indican los metadatos que Informatica Administrator muestra en la pestaña de complementos:

Atributo del servicio de repositorio	Elemento y atributo XML
Nombre	PLUGIN NAME
Nombre del proveedor	PLUGIN VENDORNAME
Descripción	PLUGIN DESCRIPTION
Nombre de grupo	FUNCTION_GROUP NAME
ID de grupo	FUNCTION_GROUP ID
Descripción del grupo	FUNCTION_GROUP DESCRIPTION

Comprobación de la exactitud de las funciones

Para comprobar la exactitud de una función personalizada, debe crear una expresión con la función e incluirla en una asignación o un flujo de trabajo. Realice los siguientes pasos para comprobar la exactitud de una función personalizada:

1. Cree datos de prueba.
2. Cree una asignación.
3. Añada la función personalizada a una expresión en la asignación.
4. Cree una asignación.
5. Ejecute el depurador (opcional). O bien, cree una sesión y un flujo de trabajo para la asignación.
6. Ejecute el flujo de trabajo.
7. Compruebe los resultados.

Instalación de funciones personalizadas

Realice los siguientes pasos para instalar funciones personalizadas:

1. Copie las bibliotecas de funciones personalizadas en el entorno de PowerCenter.
2. Registre el complemento de repositorio.

Una vez instaladas las funciones personalizadas, puede usarlas en las expresiones de transformación y flujo de trabajo.

Paso 1. Copia de bibliotecas de funciones personalizadas en PowerCenter

Copie las bibliotecas de funciones personalizadas y el archivo XML de complemento de repositorio en los directorios del cliente y el servicio de integración de PowerCenter según las instrucciones del desarrollador de las funciones personalizadas.

Si tiene una disponibilidad alta o ejecuta las sesiones en una cuadrícula, incluya las bibliotecas en una sola ubicación y defina esta ubicación como un recurso requerido.

Paso 2. Registro del complemento

Registre el archivo XML de complemento de repositorio en la herramienta de administrador.

Creación de expresiones con funciones personalizadas

Puede añadir una función personalizada a una expresión. Si especifica una función personalizada durante la creación manual de una expresión, debe añadir un prefijo a la función definida por el usuario con el espacio de nombres proporcionado por el desarrollador de la función personalizada. Al crear una expresión con el Editor de expresiones, las funciones personalizadas se muestran en la lista de todas las funciones junto con los tipos de funciones. Use las funciones personalizadas del mismo modo que cualquier otra función.

Al validar la expresión, el diseñador o el administrador de flujos de trabajo no validan la función personalizada. Solamente validan la expresión. El complemento valida la función personalizada.

CAPÍTULO 8

Referencia de API de funciones personalizadas

Este capítulo incluye los siguientes temas:

- [Introducción a la referencia de API de funciones personalizadas, 254](#)
- [API comunes, 254](#)
- [API en tiempo de ejecución, 259](#)

Introducción a la referencia de API de funciones personalizadas

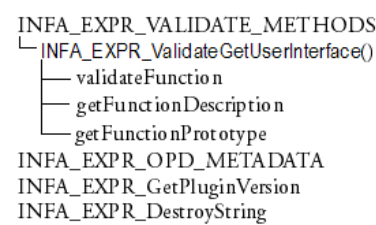
Use la API de funciones personalizadas para desarrollar funciones personalizadas que puede incluir en una expresión de transformación o flujo de trabajo. La API de funciones personalizadas es un marco para crear funciones personalizadas. Incluye API comunes y API en tiempo de ejecución. En PowerCenter, las API permiten la validación de las expresiones con funciones personalizadas y el uso de las expresiones en flujos de trabajo.

Use las API en los archivos de encabezado e implementación para desarrollar funciones personalizadas. Debe crear bibliotecas compartidas con los archivos de encabezado e implementación. Especifique las bibliotecas compartidas en el archivo de complemento de repositorio registrado en PowerCenter. Además, debe copiar las bibliotecas compartidas en el entorno de PowerCenter.

API comunes

El cliente, el servicio de integración y el servicio de repositorio de PowerCenter llaman a las API comunes para validar expresiones, eliminar devoluciones de funciones de la memoria después del uso y eliminar descripciones y prototipos de funciones de la memoria después del uso.

Las API comunes tienen la siguiente estructura:



Identificador de validación

El identificador INFA_EXPR_VALIDATE_METHODS es un identificador de validación. PowerCenter llama a INFA_EXPR_ValidateGetUserInterface para obtener los punteros de función de este identificador de validación.

Función de validación de interfaz de usuario

Cuando PowerCenter llama a INFA_EXPR_ValidateGetUserInterface, el complemento devuelve punteros de función para las funciones de validación.

Use la siguiente sintaxis:

```
INFA_EXPR_STATUS INFA_EXPR_ValidateGetUserInterface( IUNICHAR* sNamespace, IUNICHAR*
sFuncName, INFA_EXPR_VALIDATE_METHODS* functions);
```

Argumento	Tipo de datos	Entrada/ Salida	Descripción
sNamespace	IUNICHAR	Entrada	Espacio de nombres de la función personalizada.
sFuncName	IUNICHAR	Entrada	Nombre de la función personalizada.
funciones	INFA_EXPR_VALIDATE_METHODS	Salida	Punteros para las distintas funciones a las que se llama durante la validación y la generación de informes.

El tipo de datos devuelto es INFA_EXPR_STATUS. Use ISUCCESS e IFailure como valores devueltos. Si la función devuelve IFailure, el complemento no ha implementado la función o se ha generado otro tipo de error.

INFA_EXPR_ValidateGetUserInterface devuelve las siguientes funciones:

- **validateFunction.** Valida una función personalizada.
- **getFunctionDescription.** Describe una función personalizada.
- **getFunctionPrototype.** Proporciona un prototipo para una función personalizada.
- **pushdownFunction.** Genera código SQL para la optimización de inserciones.

Función de validación de la función personalizada

PowerCenter llama a validateFunction para validar los argumentos de la función personalizada. Utilice esta función para proporcionar el nombre, el tipo de datos, la precisión y la escala de los argumentos de la

función personalizada. Esta función también se utiliza para proporcionar el tipo de datos del valor de retorno de la función personalizada.

PowerCenter llama a esta función una vez para cada instancia de la función personalizada utilizada en una asignación o flujo de trabajo.

Utilice la siguiente sintaxis:

```
INFA_EXPR_STATUS *(validateFunction)(IUNICHAR* sNamespace, IUNICHAR* sFuncName, IUINT32 numArgs, INFA_EXPR_OPD_METADATA** inputArgList, INFA_EXPR_OPD_METADATA* retValue);
```

Argumento	Tipo de datos	Entrada/ Salida	Descripción
sNamespace	IUNICHAR	Entrada	Espacio del nombre de la función.
sFuncName	IUNICHAR	Entrada	Nombre de la función personalizada para validar.
numArgs	IUINT32	Entrada	Número de argumentos de la función personalizada.
inputArgList	INFA_EXPR_OPD_METADATA	Entrada	Argumentos de entrada de la función personalizada.
retValue	INFA_EXPR_OPD_METADATA	Salida	Metadatos del puerto de retorno de la función personalizada. Establezca el tipo de datos, la precisión y la escala del valor de retorno de este argumento.

El tipo de datos de retorno es `INFA_EXPR_STATUS`. Utilice `ISUCCESS` y `IFAILURE` como valor de retorno. Cuando la función devuelve `IFAILURE`, PowerCenter muestra un mensaje de error.

Función de descripción de funciones personalizadas

PowerCenter llama a `getFunctionDescription` para obtener una descripción de la función personalizada. Además, llama a `destroyString` para eliminar la descripción de la memoria después del uso.

Use la siguiente sintaxis:

```
IUNICHAR* *(getFunctionDescription)(IUNICHAR* sNamespace, IUNICHAR* sFuncName);
```

Argumento	Tipo de datos	Entrada/ Salida	Descripción
sNamespace	IUNICHAR	Entrada	Espacio de nombres de la función.
sFuncName	IUNICHAR	Entrada	Nombre de la función personalizada que debe describir el complemento.

El tipo de datos devuelto es `IUNICHAR`. El valor devuelto debe ser una cadena terminada en un valor nulo.

Función de prototipo de funciones personalizadas

PowerCenter llama a `getFunctionPrototype` para obtener los argumentos de la función personalizada en el Editor de expresiones. Además, llama a `destroyString` para eliminar los argumentos de la memoria después del uso.

Use la siguiente sintaxis:

```
IUNICHAR* *(getFunctionPrototype) (IUNICHAR* sNamespace, IUNICHAR* sFuncName);
```

Argumento	Tipo de datos	Entrada/ Salida	Descripción
sNamespace	IUNICHAR	Entrada	Espacio de nombres de la función.
sFuncName	IUNICHAR	Entrada	Nombre de la función personalizada que debe describir el complemento.

El tipo de datos devuelto es IUNICHAR. El valor devuelto debe ser una cadena terminada en un valor nulo. La función devuelve NULL si no hay ningún valor para los argumentos.

Función de inserción de funciones personalizadas

PowerCenter llama a pushdownFunction para generar código SQL para la optimización de inserciones.

Utilice la siguiente sintaxis:

```
INFA_EXPR_STATUS pushdownFunctionEcho(IUNICHAR* sNameSpace,  
                                       IUNICHAR* sFuncName,  
                                       IUINT32 numArgs,  
                                       INFA_EXPR_OPD_METADATA** inputArgList,  
                                       EDatabaseType dbType,  
                                       EPushdownMode pushdownMode,  
                                       IUNICHAR** sGenSql)
```

Argumento	Tipo de datos	Entrada/ Salida	Descripción
sNameSpace	IUNICHAR	Entrada	Espacio del nombre de la función.
sFuncName	IUNICHAR	Entrada	Nombre de la función personalizada para validar.
numArgs	IUINT32	Entrada	Número de argumentos de la función personalizada.
inputArgList	INFA_EXPR_OPD_METADATA	Entrada	Argumentos de entrada de la función personalizada.
dbType	EDatabaseType	Entrada	Tipo de base de datos.
pushdownMode	EPushdownMode	Entrada	Tipo de optimización de inserciones.
sGenSql	IUNICHAR	Salida	SQL generado por la función personalizada.

El tipo de datos de retorno es INFA_EXPR_STATUS. Utilice ISUCCESS y IFailure como valor de retorno. Cuando la función devuelve IFailure, PowerCenter muestra un mensaje de error.

Estructura de INFA_EXPR_OPD_METADATA

Esta estructura define los metadatos de los operandos de expresión, incluidos los argumentos pasados a la función y el tipo devuelto.

La estructura contiene los siguientes metadatos:

- **datatype.** Tipo de datos del argumento.
- **precision.** Precisión del argumento.
- **scale.** Escala del argumento.
- **isValueConstant.** Indica si el argumento es una constante. En ese caso, el marco evalúa el argumento una vez para cada llamada a la función. El marco usa isValueConstant para optimizar el rendimiento. Si los argumentos de entrada son constantes, el complemento puede obtener los valores de los argumentos durante la inicialización de las instancias de función para optimizar el rendimiento. En el caso de los valores de salida, el complemento establece isValueConstant en TRUE.

Función de obtención de versión de complemento

Esta función define la versión del complemento. Esta versión debe coincidir con la versión de la API de funciones personalizadas, que es 1.0.0.

Use la siguiente sintaxis:

```
INFA_EXPR_STATUS INFA_EXPR_GetPluginVersion(INFA_VERSION *sdkVersion, INFA_VERSION *pluginVersion);
```

Argumento	Tipo de datos	Entrada/ Salida	Descripción
sdkVersion	INFA_VERSION	Entrada	Versión de la API de funciones personalizadas. Use 1.0.0.
pluginVersion	INFA_VERSION	Salida	Versión del complemento que se va a crear.

El tipo de datos devuelto es INFA_EXPR_STATUS. Use ISUCCESS e IFailure como valores devueltos. Si la función devuelve IFailure, la sesión o el flujo de trabajo generan un error.

Función de destrucción de cadenas

Esta función destruye todas las cadenas devueltas por el complemento. Por ejemplo, destruye los mensajes de error o el valor devuelto de otras llamadas a funciones, como getFunctionDescription.

Use la siguiente sintaxis:

```
void *(DestroyString) (IUNICHAR* str);
```

Argumento	Tipo de datos	Entrada/ Salida	Descripción
str	IUNICHAR	Entrada	Cadena de entrada que elimina esta función.

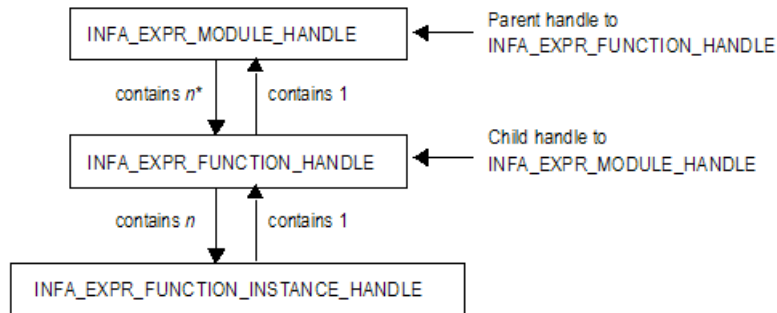
La función no devuelve ningún valor.

API en tiempo de ejecución

El servicio de integración de PowerCenter llama a las API en tiempo de ejecución durante una sesión para evaluar la expresión que contiene la función personalizada. Inicializa el complemento en los niveles del módulo, función e instancia de la función.

Cada nivel contiene un conjunto de funciones. Estas funciones están asociadas con un manejador, como INFA_EXPR_MODULE_HANDLE. El primer parámetro de estas funciones es el manejador al que afectan las funciones. Los manejadores de API de funciones personalizadas tienen una relación jerárquica entre sí. Un manejador principal tiene una relación 1:*n* con su manejador secundario.

La siguiente figura muestra los manejadores de la API de funciones personalizadas:



La siguiente tabla describe los manejadores en tiempo de ejecución:

Nombre del manejador	Descripción
INFA_EXPR_MODULE_HANDLE	Representa la biblioteca compartida o DLL. El complemento sólo puede acceder al manejador del módulo de su propia biblioteca compartida o DLL. No puede acceder al manejador del módulo de otra biblioteca compartida o DLL.
INFA_EXPR_FUNCTION_HANDLE	Representa una función personalizada dentro de la biblioteca compartida o DLL.
INFA_EXPR_FUNCTION_INSTANCE_HANDLE	Representa una instancia de función personalizada específica.

Funciones de nivel de módulo

PowerCenter llama a las funciones de nivel de módulo una vez para cada biblioteca compartida o DLL.

Función de nivel de módulo de obtención de interfaz de usuario

Esta función establece los punteros de función para la interacción en el nivel de módulo.

Use la siguiente sintaxis:

```
INFA_EXPR_STATUS INFA_EXPR_ModuleGetUserInterface(INFA_EXPR_LIB_METHODS* functions);
```

Argumento	Tipo de datos	Entrada/ Salida	Descripción
funciones	INFA_EXPR_LIB_METHODS	Salida	Funciones de obtención de interfaz de usuario de nivel de módulo.

El tipo de datos devuelto es INFA_EXPR_STATUS. Use ISUCCESS e IFailure como valores devueltos. Si la función devuelve IFailure, la sesión o el flujo de trabajo generan un error.

Esta función devuelve las siguientes funciones:

- **function_init.** Inicializa la función.
- **function_deinit.** Desinicializa la función.

Función de inicialización de nivel de módulo

PowerCenter llama a module_init una vez por cada módulo para inicializar cualquier estructura de datos global de la función. Llama a esta función antes de llamar a cualquier función de nivel de función.

Use la siguiente sintaxis:

```
INFA_EXPR_STATUS (*module_init) (INFA_EXPR_MODULE_HANDLE module);
```

Argumento	Tipo de datos	Entrada/ Salida	Descripción
módulo	INFA_EXPR_MODULE_HANDLE	Entrada	Almacena los datos que se pueden recuperar en el nivel de función.

El tipo de datos devuelto es INFA_EXPR_STATUS. Use ISUCCESS e IFailure como valores devueltos. Si la función devuelve IFailure, la sesión o el flujo de trabajo generan un error.

Función de desinicialización de nivel de módulo

PowerCenter llama a module_deinit una vez por cada módulo para desinicializar cualquier estructura de datos de esta función. Llama a esta función una vez completadas todas las interacciones de nivel de función.

Use la siguiente sintaxis:

```
INFA_EXPR_STATUS (*module_deinit) (INFA_EXPR_MODULE_HANDLE module);
```

Argumento	Tipo de datos	Entrada/ Salida	Descripción
módulo	INFA_EXPR_MODULE_HANDLE	Entrada	Identificador de nivel de módulo que el marco pasa al complemento cuando se llama a la función de inicialización de nivel de módulo.

El tipo de datos devuelto es INFA_EXPR_STATUS. Use ISUCCESS e IFailure como valores devueltos. Si la función devuelve IFailure, la sesión o el flujo de trabajo generan un error.

Funciones de nivel de función

PowerCenter llama a las funciones de nivel de función una vez para cada función personalizada y una vez para cada biblioteca compartida o DLL que proporcione los parámetros para la función personalizada.

Función de nivel de función de obtención de interfaz de usuario

Esta función establece los punteros de función para la interacción en el nivel de función. PowerCenter llama a esta función para cada función personalizada que implementa esta biblioteca.

Use la siguiente sintaxis:

```
INFA_EXPR_STATUS INFA_EXPR_FunctionGetUserInterface (IUNICHAR* nameSpaceName, IUNICHAR*
functionName, INFA_EXPR_FUNCTION_METHODS* functions);
```

Argumento	Tipo de datos	Entrada/ Salida	Descripción
nameSpaceName	IUNICHAR	Entrada	Espacio de nombres de la función.
functionName	IUNICHAR	Entrada	Nombre de la función personalizada que debe describir el complemento.
función	INFA_EXPR_FUNCTION_METHODS	Entrada	Marcador de posición para los punteros de función que se van a invocar en el nivel de instancia de función.

El tipo de datos devuelto es INFA_EXPR_STATUS. Use ISUCCESS e IFailure como valores devueltos. Si la función devuelve IFailure, la sesión o el flujo de trabajo generan un error.

Esta función devuelve las siguientes funciones:

- **function_init.** Inicializa la función.
- **function_deinit.** Desinicializa la función.

Función de inicialización de nivel de función

PowerCenter llama a function_init una vez por cada función personalizada para inicializar cualquier estructura relacionada con la función personalizada. Llama a la función de inicialización de nivel de módulo antes de llamar a esta función.

Use la siguiente sintaxis:

```
INFA_EXPR_STATUS (*function_init) (INFA_EXPR_FUNCTION_HANDLE fnInstance);
```

Argumento	Tipo de datos	Entrada/ Salida	Descripción
fnInstance	INFA_EXPR_FUNCTION_HANDLE	Entrada	Se realizan las siguientes tareas: <ul style="list-style-type: none">- Se almacenan los punteros definidos por el usuario para el marco que se va a recuperar en tiempo de ejecución o durante la desinicialización.- Se inicializan las estructuras de datos para el nivel de instancia de función.- Si el argumento de entrada es una constante, el complemento recupera el valor de esta constante y realiza el preprocesamiento necesario.

El tipo de datos devuelto es INFA_EXPR_STATUS. Use ISUCCESS e IFailure como valores devueltos. Si la función devuelve IFailure, la sesión o el flujo de trabajo generan un error.

Función de desinicialización de nivel de función

PowerCenter llama a esta función una vez por cada nivel de función para desinicializar cualquier estructura relacionada con la función personalizada.

Use la siguiente sintaxis:

```
INFA_EXPR_STATUS (*function_deinit) (INFA_EXPR_FUNCTION_HANDLE function);
```

Argumento	Tipo de datos	Entrada/ Salida	Descripción
fnInstance	INFA_EXPR_FUNCTION_HANDLE	Entrada	Identificador de nivel de función que el marco pasa a los complementos cuando se llama a la función de inicialización de nivel de instancia de función.

El tipo de datos devuelto es INFA_EXPR_STATUS. Use ISUCCESS e IFailure como valores devueltos. Si la función devuelve IFailure, la sesión o el flujo de trabajo generan un error.

Funciones de nivel de instancia de función

PowerCenter llama a estas funciones cada vez que se usa una función personalizada en una asignación o un flujo de trabajo.

Función de nivel de función de obtención de interfaz de usuario

Esta función establece los punteros de función para la interacción en el nivel de función. PowerCenter llama a esta función para cada función personalizada que implementa esta biblioteca.

Use la siguiente sintaxis:

```
INFA_EXPR_STATUS INFA_EXPR_FunctionInstanceGetUserInterface(IUNICHAR* functionName,  
INFA_EXPR_FUNCTION_INSTANCE_METHODS* functions)
```

Argumento	Tipo de datos	Entrada/ Salida	Descripción
functionName	IUNICHAR	Entrada	Espacio de nombres de la función.
funciones	INFA_EXPR_FUNCTION_INSTANC E_METHODS	Entrada	Marcador de posición para los punteros de función que se van a invocar en el nivel de instancia de función.

Esta función devuelve las siguientes funciones:

- **fnInstance_init.** Inicializa una instancia de una función personalizada.
- **fnInstance_processRow.** Procesa los datos de una instancia de una función personalizada.
- **fnInstance_deinit.** Desinicializa una instancia de una función personalizada.

Función de inicialización de nivel de instancia de función

PowerCenter llama a `fnInstance_init` una vez por cada instancia de función personalizada para inicializar cualquier estructura relacionada con la instancia de función personalizada. Si hay dos instancias de una función personalizada en una asignación o un flujo de trabajo, PowerCenter llama a esta función dos veces. PowerCenter llama a la función de inicialización de nivel de módulo antes de llamar a esta función.

Use la siguiente sintaxis:

```
INFA_EXPR_STATUS (*fnInstance_init)(INFA_EXPR_FUNCTION_INSTANCE_HANDLE fnInstance);
```

Argumento	Tipo de datos	Entrada/ Salida	Descripción
fnInstance	INFA_EXPR_FUNCTION_HANDLE	Entrada	Se realizan las siguientes tareas: <ul style="list-style-type: none">- Se almacenan los punteros definidos por el usuario para el marco que se va a recuperar en tiempo de ejecución o durante la desinicialización.- Se inicializan las estructuras de datos para el nivel de instancia de función.- Si el argumento de entrada es una constante, el complemento recupera el valor de esta constante y realiza el preprocesamiento necesario.

El tipo de datos devuelto es `INFA_EXPR_STATUS`. Use `ISUCCESS` e `IFAILURE` como valores devueltos. Si la función devuelve `IFAILURE`, la sesión o el flujo de trabajo generan un error.

Función de procesamiento de fila de instancia de función

PowerCenter llama a `fnInstance_processRow` cuando una fila de entrada está disponible para una instancia de función personalizada. Los datos de los argumentos de entrada de la función personalizada están unidos y se acceden a través de `fnInstance-inputOPDHandles`. Ajuste los datos, la longitud y el indicador de los

puertos de salida y retorno en fnInstance->retHandle. PowerCenter llama a la función de inicialización de la función de nivel antes de llamar a esta función.

Utilice la siguiente sintaxis:

```
INFA_EXPR_ROWSTATUS (*fnInstance_processRow) (INFA_EXPR_FUNCTION_INSTANCE_HANDLE fnInstance);
```

Argumento	Tipo de datos	Entrada Salida	Descripción
fnInstance	INFA_EXPR_FUNCTION_HANDLE	Entrada	Manejador del nivel de función para el que los datos están disponibles.

El tipo de dato del valor de retorno es INFA_EXPR_ROWSTATUS. Utilice los siguientes valores para el valor de retorno:

- **INFA_ROWSUCCESS.** Indica que la función ha procesado correctamente la fila de datos.
- **INFA_ROWERROR.** Indica que la función ha encontrado un error en la fila de datos. El servicio de integración de PowerCenter incrementa el contador del número de errores internos. Sólo devuelve este valor cuando el modo de acceso a datos es fila.
- **INFA_FATALERROR.** Indica que la función ha detectado un error fatal en la fila de datos o en el bloque de datos. El servicio de integración de PowerCenter interrumpe la sesión.

Función de desinicialización de nivel de instancia de función

PowerCenter llama a fnInstance_deinit una vez para cada nivel de función durante la desinicialización. Llama a esta función para desinicializar cualquier estructura relacionada con la función personalizada.

Use la siguiente sintaxis:

```
INFA_EXPR_STATUS (*fnInstance_deinit) (INFA_EXPR_FUNCTION_INSTANCE_HANDLE fnInstance);
```

Argumento	Tipo de datos	Entrada/ Salida	Descripción
fnInstance	INFA_EXPR_FUNCTION_INSTANCE_HANDLE	Entrada	Identificador de nivel de función que el marco pasa a los complementos cuando se llama a la función de inicialización de nivel de instancia de función.

El tipo de datos devuelto es INFA_EXPR_STATUS. Use ISUCCESS e IFailure como valores devueltos. Si la función devuelve IFailure, la sesión o el flujo de trabajo generan un error.

INDICE

A

- actualizaciones de lenguaje de transformación
 - expresiones booleanas [25](#)
- actualizaciones del idioma de transformación
 - expresiones de comparación [25](#)
- algoritmo del estándar de cifrado avanzado
 - descripción [67](#)
- Algoritmo estándar de cifrado avanzado
 - descripción [67](#)
- alta precisión
 - ABS [62](#)
 - AVG [70](#)
 - CEIL [72](#)
 - CUME [84](#)
 - EXP [96](#)
 - función ABS [62](#)
 - función AVG [70](#)
 - función CREATE_TIMESTAMP_TZ [83](#)
 - Función CUME [84](#)
 - función GET_TIMESTAMP [103](#)
 - función GET_TIMEZONE [102](#)
 - Función MAX [134](#)
 - Función MEDIAN [138](#)
 - Función MIN [144](#)
 - Función MOVINGAVG [148](#)
 - Función MOVINGSUM [150](#)
 - función PERCENTILE [152](#)
 - Función ROUND [174](#)
 - Función STDDEV [196](#)
 - Función SUM [200](#)
 - función TO_TIMESTAMP_TZ [222](#)
 - función TO_DECIMAL [216](#)
 - función TO_DECIMAL38 [218](#)
 - función TRUNC [227](#)
 - LOG [126](#)
 - MAX (números) [134](#)
 - MEDIAN [138](#)
 - MIN (números) [144](#)
 - MOD [147](#)
 - MOVINGAVG [148](#)
 - MOVINGSUM [150](#)
 - operadores aritméticos [29](#)
 - PERCENTILE [152](#)
 - POWER [155](#)
 - ROUND (números) [174](#)
 - SIGN [189](#)
 - SIN [190](#)
 - SUM [200](#)
- AND
 - palabra reservada [20](#)
- año 2000
 - fechas [41](#)
- archivo de encabezado
 - crear [234](#)

- archivo de implementación
 - crear [236](#)
- archivo XML de complemento
 - elemento FUNCTION [249](#)
 - elemento LIBRARY [250](#)
- archivo XML del complemento
 - elemento FUNCTION_GROUP [248](#)
 - elemento PLUGIN [248](#)
- archivos sin formato
 - fechas [43](#)
- aritméticos
 - valores de fecha y hora [53](#)
- ASCII
 - conversión de valores ASCII [74](#)
 - convirtiendo a valores Unicode [75](#)
 - convirtiendo caracteres en valores ASCII [69](#)
 - función CHR [74](#)
- atributos de ID del repositorio
 - obtención [233](#)

B

- bases de datos relacionales
 - fechas [43](#)
- bibliotecas compartidas
 - compilar para funciones personalizadas [245](#)
- bigint
 - conversión de valores a [204](#)
- búsquedas múltiples
 - ejemplo de constante TRUE [27](#)

C

- cadena de formato J
 - usar con IS_DATE [52](#)
 - usar con TO_CHAR [48](#)
 - usar con TO_DATE [52](#)
- cadena de formato RR
 - descripción [42](#)
 - diferencia entre YY y RR [42](#)
 - usar con IS_DATE [52](#)
 - usar con TO_CHAR [48](#)
 - usar con TO_DATE [52](#)
- cadena de formato SSSSS
 - usar con IS_DATE [52](#)
 - usar con TO_CHAR [48](#)
 - usar con TO_DATE [52](#)
- cadena de formato YY
 - diferencia entre RR e YY [42](#)
 - usar con IS_DATE [52](#)
 - usar con TO_CHAR [49](#)
 - usar con TO_DATE [52](#)
- cadenas
 - adición de caracteres [129](#)

cadenas (*continuado*)

- adición de espacios en blanco [129](#)
 - concatenación [76](#)
 - concatenar [30](#)
 - conversión de fechas a caracteres [206](#)
 - conversión de longitud [177](#)
 - conversión de valores numéricos a cadenas de texto [211](#)
 - convirtiendo cadenas de caracteres en fechas [213](#)
 - devolución de porción [198](#)
 - eliminación de caracteres [130](#)
 - eliminación de espacios en blanco [130](#)
 - eliminación de espacios en blanco y caracteres [178](#)
 - juego de caracteres [111](#)
 - número de caracteres [124](#)
 - sustitución de un carácter [163](#)
 - sustitución de varios caracteres [166](#)
 - uso de mayúsculas [110](#), [128](#), [228](#)
- cadenas con formato
- coincidencia [51](#)
- cadenas de caracteres
- conversión de fechas [206](#)
 - convirtiendo a fechas [213](#)
- cadenas de formato
- definición [40](#)
 - día juliano [49](#)
 - día juliano modificado [49](#)
 - fecha juliana [46](#)
 - fecha juliana modificada [46](#)
 - fechas [44](#)
 - función IS_DATE [49](#)
 - función TO_CHAR [46](#)
 - función TO_DATE [49](#)
- cadenas de texto
- conversión de valores numéricos [211](#)
- cadenas vacías
- comprobar [124](#)
- cálculo de división
- devolución de resto [147](#)
- calendario gregoriano
- en funciones de fechas [41](#)
- calendarios
- tipos de fechas admitidas [41](#)
- calificador de referencia:INFA
- palabra reservada [20](#)
- calificador de referencia:MCR
- palabra reservada [20](#)
- calificadores de referencia
- descripción [18](#)
- caracteres
- adición a cadenas [129](#)
 - agregar a cadenas [177](#)
 - caracteres ASCII [69](#), [74](#)
 - caracteres Unicode [69](#), [74](#), [75](#)
 - codificación [192](#)
 - devolver número [124](#)
 - eliminación en cadenas [178](#)
 - recuento [198](#)
 - sustitución de un [163](#)
 - uso de mayúsculas [110](#), [128](#), [228](#)
- Caracteres
- codificación [139](#)
 - eliminación en cadenas [130](#)
 - sustitución de varios caracteres [166](#)
- cifrado
- función AES_ENCRYPT [67](#)
 - utilización del algoritmo estándar de cifrado avanzado [67](#)
- codificación
- caracteres [192](#)

codificación (*continuado*)

- Caracteres [139](#)
 - función ENC_BASE64 [94](#)
- comentarios
- agregar a las expresiones [20](#)
- comillas
- insertar simples mediante función CHR [18](#)
- comillas simples en literales de cadena
- función CHR [74](#)
 - uso de funciones CHR y CONCAT [76](#)
- compilar
- módulos para funciones personalizadas [245](#)
- complemento del repositorio
- obtención de atributos de Id. del repositorio [233](#)
 - registrar [253](#)
- componentes del idioma de transformación
- resumen [15](#)
- compresión
- comprimir datos [76](#)
 - descompresión de datos [93](#)
- concatenación
- cadenas [76](#)
- concatenar
- cadenas [30](#)
- condiciones de filtro
- funciones de agregado [56](#)
 - valores nulos [26](#)
- constante DD_DELETE
- descripción [22](#)
 - ejemplo de estrategia de actualización [22](#)
 - palabra reservada [20](#)
- constante DD_INSERT
- descripción [23](#)
 - ejemplo de estrategia de actualización [23](#)
 - palabra reservada [20](#)
- constante DD_REJECT
- descripción [23](#)
 - ejemplo de estrategia de actualización [23](#)
 - palabra reservada [20](#)
- constante DD_UPDATE
- descripción [24](#)
 - ejemplo de estrategia de actualización [24](#)
 - palabra reservada [20](#)
- constante FALSE
- descripción [25](#)
 - palabra reservada [20](#)
- constante NULL
- descripción [25](#)
 - palabra reservada [20](#)
- constante TRUE
- descripción [27](#)
 - palabra reservada [20](#)
- constantes
- DD_INSERT [23](#)
 - DD_REJECT [23](#)
 - DD_UPDATE [24](#)
 - descripción [15](#)
 - FALSE [25](#)
 - NULL [25](#)
 - TRUE [27](#)
- conversión
- cadenas de fechas [42](#)
- conversión de cadena
- fechas [42](#)
- coseno
- calcular [78](#)
 - calcular coseno hiperbólico [79](#)

crear

- archivo de encabezado para funciones personalizadas [234](#)
- archivo de implementación para funciones personalizadas [236](#)
- funciones personalizadas [233](#)

D

decodificación

- función DEC_BASE64 [90](#)

descifrado

- función AES_DECRYPT [67](#)

desviación estándar

- devolviendo [196](#)

deteniendo

- sesiones [62](#)

día juliano

- cadena de formato [49](#)

día juliano modificado

- cadena de formato [49](#)

DLL

- compilar para funciones personalizadas [245](#)

E

calificador de referencia:EXT

- descripción [18](#)
- palabra reservada [20](#)

Editor de expresiones

- usar con funciones personalizadas [253](#)

elemento FUNCTION

- descripción [249](#)

elemento FUNCTION_GROUP

- descripción [248](#)

elemento LIBRARY

- descripción [250](#)

elemento PLUGIN

- descripción [248](#)

elementos

- FUNCTION [249](#)
- FUNCTION_GROUP [248](#)
- LIBRARY [250](#)
- PLUGIN [248](#)

enteros

- conversión de valores a [220](#)

espacios

- evitar en filas [120](#)
- quitar con DD_REJECT [24](#)

espacios de nombres

- elegir [249](#)

estrategia de actualización

- ejemplo de DD_DELETE [22](#)
- ejemplo de DD_INSERT [23](#)
- ejemplo de DD_REJECT [23](#)
- ejemplo de DD_UPDATE [24](#)

expresiones

- agregar comentarios [20](#)
- condicional [25](#)
- crear con funciones personalizadas [253](#)
- resumen [15](#)
- sintaxis [17](#)
- utilización de operadores [28](#)

expresiones anidadas

- operadores [28](#)

expresiones de transformación

- restricciones de valores nulos [25](#)
- resumen [15](#)

F

fecha juliana

- cadena de formato [46](#)

fecha juliana modificada

- cadena de formato [46](#)

fechas

- año 2000 [41](#)
- archivos sin formato [43](#)
- bases de datos relacionales [43](#)
- cadenas de formato [44](#)
- conversión a cadenas de caracteres [206](#)
- formato de fecha y hora predeterminado [43](#)
- funciones [58](#)
- introducción [40](#)
- Juliano [41](#)
- juliano modificado [41](#)
- realizar operaciones aritméticas [53](#)
- redondeo [170](#)
- truncando [224](#)

fechas julianas

- en funciones de fechas [41](#)

filas

- devolución de la primera fila [97](#)
- devolución de promedio [148](#)
- devolución de suma [150](#)
- devolver cualquier fila [68](#)
- devolver la última fila [121](#)
- evitar espacios [120](#)
- omisión [94](#)
- recuento [80](#)
- total de ejecución [84](#)

formato

- de cadena de caracteres a fecha [213](#)
- de fecha a cadena de caracteres [206](#)

formato de fecha y hora predeterminado

- configuración [43](#)

Función ABORT

- descripción [62](#)

función ABS

- descripción [62](#)

función ADD_TO_DATE

- Descripción [64](#)

función AES_DECRYPT

- descripción [67](#)

función AES_ENCRYPT

- descripción [67](#)

Función ANY

- descripción [68](#)

Función ASCII

- descripción [69](#)

función AVG

- descripción [70](#)

función CEIL

- descripción [72](#)

función CHOOSE

- descripción [73](#)

función CHR

- descripción [74](#)

- inserción de comillas simples [74](#)

- insertar comillas simples [18](#)

Función CHRCODE

- descripción [75](#)

función COMPRESS

- descripción [76](#)

función CONCAT

- descripción [76](#)

- inserción de comillas simples mediante [76](#)

función CONVERT_BASE
 descripción [78](#)

función COS
 descripción [78](#)

función COSH
 descripción [79](#)

Función COUNT
 descripción [80](#)

función CRC32
 descripción [83](#)

función CREATE_TIMESTAMP_TZ
 descripción [83](#)

Función CUME
 descripción [84](#)

función DATE_COMPARE
 descripción [86](#)

Función DATE_DIFF
 descripción [87](#)

Función de ejemplo ECHO
 descripción [232](#)

función de muestra
 ECHO [232](#)
 SampleLoanPayment [232](#)

Función de muestra SampleLoanPayment
 descripción [232](#)

función DEC_BASE64
 descripción [90](#)

función DECODE
 Descripción [91](#)
 internacionalización [16](#)

función DECOMPRESS
 descripción [93](#)

función ENC_BASE64
 descripción [94](#)

función ERROR
 Descripción [94](#)
 valor predeterminado [94](#)

función EXP
 descripción [96](#)

Función FIRST
 descripción [97](#)

función FLOOR
 descripción [98](#)

función FLOOR (expresiones)
 descripción [98](#)

Función FV
 descripción [99](#)

Función GET_DATE_PART
 descripción [100](#)

función GET_TIMESTAMP
 descripción [103](#)

función GET_TIMEZONE
 descripción [102](#)

función GREATEST
 descripción [104](#)

función IIF
 Descripción [105](#)
 internacionalización [16](#)

función IN
 descripción [108](#)

función INDEXOF
 descripción [109](#)

función INITCAP
 descripción [110](#)
 internacionalización [16](#)

función INSTR
 Descripción [111](#)

función IS_DATE
 cadenas de formato [49](#)
 Descripción [115](#)

función IS_NUMBER
 descripción [118](#)

función IS_SPACES
 descripción [120](#)

función ISNULL
 descripción [114](#)

función LAST
 descripción [121](#)

función LAST_DAY
 descripción [122](#)

función LEAST
 descripción [123](#)

función LENGTH
 comprobación de cadenas vacías [124](#)
 descripción [124](#)

función LN
 descripción [125](#)

función LOG
 descripción [126](#)

función LOOKUP
 Descripción [127](#)

función LOWER
 descripción [128](#)
 internacionalización [16](#)

Función LPAD
 descripción [129](#)

función LTRIM
 Descripción [130](#)

función MAKE_DATE_TIME
 descripción [132](#)

Función MAX (cadena)
 descripción [135](#)

función MAX (fechas)
 descripción [133](#)
 internacionalización [16](#)

función MAX (números)
 internacionalización [16](#)

Función MAX (números)
 descripción [134](#)

Función MD5
 descripción [137](#)

Función MEDIAN
 descripción [138](#)

función MIN (fechas)
 descripción [143](#)
 internacionalización [16](#)

función MIN (números)
 internacionalización [16](#)

Función MIN (números)
 descripción [144](#), [145](#)

función MOD
 descripción [147](#)

Función MOVINGAVG
 Descripción [148](#)

Función MOVINGSUM
 Descripción [150](#)

Función NPER
 descripción [151](#)

función PERCENTILE
 Descripción [152](#)

función PMT
 descripción [154](#)

función POWER
 descripción [155](#)

función PV
 descripción [156](#)
 función RAND
 descripción [156](#)
 función RATE
 descripción [157](#)
 Función REG_EXTRACT
 Descripción [158](#)
 uso de la sintaxis de perl [158](#)
 función REG_MATCH
 uso de la sintaxis de perl [158](#)
 Función REG_MATCH
 descripción [161](#)
 función REG_REPLACE
 descripción [162](#)
 Función REPLACECHR
 descripción [163](#)
 función REPLACESTR
 Descripción [166](#)
 función REVERSE
 descripción [169](#)
 función ROUND (fechas)
 Descripción [170](#)
 proceso de subsegundos [170](#)
 Función ROUND (números)
 descripción [174](#)
 función RPAD
 descripción [177](#)
 función RTRIM
 Descripción [178](#)
 función SET_DATE_PART
 Descripción [181](#)
 función SETCOUNTVARIABLE
 Descripción [179](#)
 función SETMAXVARIABLE
 descripción [184](#)
 función SETMINVARIABLE
 Descripción [185](#)
 función SETVARIABLE
 Descripción [187](#)
 función SIGN
 descripción [189](#)
 función SIN
 descripción [190](#)
 función SINH
 descripción [191](#)
 Función SOUNDEX
 descripción [192](#)
 Función SQL_IS_CHAR
 uso de REG_MATCH [161](#)
 Función SQL LIKE
 uso de REG_MATCH [161](#)
 función SQL_LIKE
 descripción [194](#)
 función SQRT
 descripción [195](#)
 Función STDDEV
 descripción [196](#)
 Función SUBSTR
 descripción [198](#)
 Función SUM
 descripción [200](#)
 función SYSTIMESTAMP
 descripción [201](#)
 función TAN
 descripción [203](#)
 función TANH
 descripción [203](#)
 función TO_TIMESTAMP_TZ
 descripción [222](#)
 función TO_CHAR (fechas)
 cadenas de formato [46](#)
 descripción [206](#)
 ejemplos [48](#)
 función TO_CHAR (números)
 descripción [211](#)
 función TO_DATE
 cadenas de formato [49](#)
 ejemplos [51](#)
 Función TO_DATE
 descripción [213](#)
 función TO_DECIMAL
 descripción [216](#)
 función TO_DECIMAL38
 descripción [218](#)
 función TO_FLOAT
 descripción [219](#)
 función TO_INTEGER
 descripción [220](#)
 Función TRUNC (fechas)
 descripción [224](#)
 procesamiento de subsegundos [224](#)
 función TRUNC (números)
 descripción [227](#)
 función UPPER
 descripción [228](#)
 internacionalización [16](#)
 Función UUID_UNPARSE
 descripción [230](#)
 Función UUID4
 descripción [229](#)
 Función VARIANCE
 descripción [230](#)
 funciones
 agregado [54](#)
 cadena [61](#)
 caracteres [56](#)
 categorías [54](#)
 científicas [60](#)
 codificación [59](#)
 conversión [57](#)
 descripción [15](#)
 especiales [60](#)
 fecha [58](#)
 financieras [59](#)
 internacionalización [16](#)
 limpieza de datos [57](#)
 numéricas [59](#)
 prueba [61](#)
 variable [61](#)
 funciones agregadas
 AVG [70](#)
 COUNT [80](#)
 FIRST [97](#)
 MAX (cadena) [135](#)
 MAX (números) [134](#)
 MEDIAN [138](#)
 MIN (números) [144](#), [145](#)
 STDDEV [196](#)
 SUM [200](#)
 funciones científicas
 COS [78](#)
 COSH [79](#)
 descripción [60](#)
 SIN [190](#)
 SINH [191](#)

funciones científicas (*continuado*)

TAN [203](#)
TANH [203](#)

funciones de agregado

ANY [68](#)
descripción [54](#)
LAST [121](#)
MAX (fechas) [133](#)
MIN (fechas) [143](#)
PERCENTILE [152](#)
valores nulos [26](#), [56](#)
VARIANCE [230](#)

funciones de cadena

CHOOSE [73](#)
descripción [61](#)
INDEXOF [109](#)
REVERSE [169](#)

funciones de carácter

ASCII [69](#)
CHR [74](#)
CHRCODE [75](#)
función CONCAT [76](#)
INITCAP [110](#)
INSTR [111](#)
LPAD [129](#)
LTRIM [130](#)
REG_EXTRACT [158](#)
REG_MATCH [161](#)
REG_REPLACE [162](#)
REPLACECHR [163](#)
REPLACESTR [166](#)
RPAD [177](#)
RTRIM [178](#)
SOUNDEX [192](#)
SUBSTR [198](#)

funciones de caracteres

LENGTH [124](#)
lista de [56](#)
LOWER [128](#)
METAPHONE [139](#)
UPPER [228](#)

funciones de codificación

AES_DECRYPT [67](#)
AES_ENCRYPT [67](#)
COMPRESS [76](#)
CRC32 [83](#)
DEC_BASE64 [90](#)
DECOMPRESS [93](#)
descripción [59](#)
ENC_BASE64 [94](#)
MD5 [137](#)

funciones de conversión

CREATE_TIMESTAMP_TZ [83](#)
descripción [57](#)
GET_TIMESTAMP [103](#)
GET_TIMEZONE [102](#)
TO_CHAR (fechas) [206](#)
TO_CHAR (números) [211](#)
TO_DATE [213](#)
TO_DECIMAL [216](#)
TO_DECIMAL38 [218](#)
TO_FLOAT [219](#)
TO_INTEGER [220](#)
TO_TIMESTAMP_TZ [222](#)

funciones de fecha

ADD_TO_DATE [64](#)
DATE_COMPARE [86](#)
DATE_DIFF [87](#)

funciones de fecha (*continuado*)

GET_DATE_PART [100](#)
ROUND [170](#)
SET_DATE_PART [181](#)
TRUNC (Fechas) [224](#)

funciones de fechas

LAST_DAY [122](#)
MAKE_DATE_TIME [132](#)
MAX (fechas) [133](#)
MIN (fechas) [143](#)
SYSTIMESTAMP [201](#)

funciones de limpieza de datos

descripción [57](#)
GREATEST [104](#)
IN [108](#)
LEAST [123](#)

funciones de prueba

descripción [61](#)
IS_DATE [115](#)
IS_NUMBER [118](#)
IS_SPACES [120](#)
ISNULL [114](#)

funciones de variable

SETCOUNTVARIABLE [179](#)
SETMINVARIABLE [185](#)
SETVARIABLE [187](#)

funciones de variables

con particiones múltiples [61](#)
descripción [61](#)

funciones especiales

ABORT [62](#)
DECODE [91](#)
descripción [60](#)
ERROR [94](#)
IIF [105](#)
LOOKUP [127](#)

funciones financieras

descripción [59](#)
Función FV [99](#)
Función NPV [151](#)
función PMT [154](#)
función PV [156](#)
función RATE [157](#)

funciones numéricas

ABS [62](#)
CEIL [72](#)
CONVERT_BASE [78](#)
CUME [84](#)
descripción [59](#)
EXP [96](#)
FLOOR [98](#)
LN [125](#)
LOG [126](#)
MOD [147](#)
MOVINGAVG [148](#)
MOVINGSUM [150](#)
POWER [155](#)
RAND [156](#)
ROUND (números) [174](#)
SIGN [189](#)
SQRT [195](#)
TRUNC (números) [227](#)

funciones personalizadas

compilar módulos [245](#)
crear [233](#)
crear archivo de encabezado [234](#)
crear archivo de implementación [236](#)
en el Editor de expresiones [253](#)

funciones personalizadas (*continuado*)

instalar [233](#), [252](#)

resumen [232](#)

funciones variables

SETMAXVARIABLE [184](#)

H

hiperbólica

función de coseno [79](#)

función de seno [191](#)

función de tangente [203](#)

I

idioma de transformación

operadores [28](#)

palabras reservadas [20](#)

instalar

funciones personalizadas [233](#), [252](#)

internacionalización

expresión no válida [16](#)

funciones afectadas [16](#)

orden de clasificación [16](#)

L

calificador de referencia:LKP

descripción [18](#)

palabra reservada [20](#)

lenguaje de transformación

comparación con SQL [17](#)

literales

comillas simples en [74](#), [76](#)

requisito de comillas simples [18](#)

literales de cadena

comillas simples en [74](#), [76](#)

requisito de comillas simples [18](#)

logaritmo

devolver [125](#), [126](#)

M

mayúsculas y minúsculas

convertir a mayúsculas [228](#)

mes

devolver el último día [122](#)

METAPHONE

descripción [139](#)

mínimo

valor, devolución [143](#)

módulos

compilar para funciones personalizadas [245](#)

N

NOT

palabra reservada [20](#)

números

redondeo [174](#)

truncando [227](#)

O

omisión

filas [94](#)

operadores

aritméticos [29](#)

descripción [15](#)

operadores de cadena [30](#)

operadores de comparación [31](#)

operadores lógicos [31](#)

utilización de cadenas en aritmética [29](#)

utilización de cadenas en comparaciones [31](#)

valores nulos [27](#)

operadores aritméticos

descripción [29](#)

utilización de cadenas en expresiones [29](#)

utilización para convertir datos [29](#)

operadores de cadena

descripción [30](#)

operadores de comparación

descripción [31](#)

utilización de cadenas en expresiones [31](#)

operadores lógicos

descripción [31](#)

OR

palabra reservada [20](#)

orden de clasificación

internacionalización [16](#)

P

\$PMFolderName

descripción [36](#)

\$PMIntegrationServiceName

descripción [36](#)

\$PMMappingName

descripción [36](#)

\$PMRepositoryServiceName

descripción [36](#)

\$PMRepositoryUserName

descripción [36](#)

\$PMSessionName

descripción [36](#)

\$PMSessionRunMode

descripción [37](#)

\$PMSourceName@TableName

descripción [35](#)

\$PMTargetName@TableName

descripción [35](#)

\$PMWorkflowName

descripción [37](#)

\$PMWorkflowRunId

descripción [37](#)

\$PMWorkflowRunInstanceName

descripción [37](#)

palabras reservadas

lista [20](#)

parámetros de asignación

definición [15](#)

precedencia del operador

expresiones [28](#)

promedios

devolviendo [148](#)

funciones agregadas para determinar [70](#)

puertos

sintaxis [18](#)

R

- raíz cuadrada
 - devolver [195](#)
- redondeo
 - fechas [170](#)
 - números [174](#)
- registrar
 - complemento del repositorio [253](#)
- restricción de clave principal
 - valores nulos [25](#)

S

- \$\$\$SessStartTime
 - usar en expresiones [38](#)
- calificador de referencia:SEQ
 - descripción [18](#)
 - palabra reservada [20](#)
- calificador de referencia:SD
 - descripción [18](#)
 - palabra reservada [20](#)
- calificador de referencia:SP
 - descripción [18](#)
 - palabra reservada [20](#)
- seno
 - devolver [190](#), [191](#)
- Servicio de integración de datos
 - manejo de valores nulos en expresiones de comparación [25](#)
- Servicio de integración de PowerCenter
 - manejo de valores nulos en expresiones de comparación [25](#)
- sesiones
 - deteniendo [62](#)
- sintaxis
 - expresión [17](#)
 - puertos [18](#)
 - reglas generales [19](#)
 - valores de retorno [18](#)
- sintaxis de COBOL
 - conversión a la sintaxis de perl [158](#)
- sintaxis de expresión regular compatible con perl
 - uso en una función REG_EXTRACT [158](#)
 - uso en una función REG_MATCH [158](#)
- sintaxis de SQL
 - conversión a la sintaxis de perl [158](#)
- Sistema operativo Windows
 - generación de una DLL para funciones personalizadas [246](#)
- SPOUTPUT
 - palabra reservada [20](#)
- subsegundos
 - procesamiento en la función TRUNC (fechas) [224](#)
 - proceso en función ROUND (fechas) [170](#)
- suma
 - devolución [200](#)
 - devolviendo [150](#)

T

- calificador de referencia:TD
 - descripción [18](#)
 - palabra reservada [20](#)
- tangente
 - devolver [203](#)
- tipos de datos
 - fecha y hora [40](#)

- total de ejecución
 - devolviendo [84](#)
- Transformación de filtro
 - usar función ISNULL [114](#)
- Transformación personalizada
 - funciones [254](#)
- truncando
 - fechas [224](#)
 - números [227](#)

U

- Unicode
 - conversión de valores Unicode [74](#)
 - convirtiendo a valores ASCII [75](#)
 - convirtiendo caracteres en valores Unicode [69](#)
- UNIX
 - compilar bibliotecas compartidas para funciones personalizadas [247](#)
- uso de mayúsculas
 - cadenas [110](#), [128](#), [228](#)

V

- valores absolutos
 - obtener [62](#)
- valores de cadena
 - devolución del máximo [135](#)
 - devolución del mínimo [145](#)
- valores de doble precisión
 - números de coma flotante [219](#)
- valores de exponente
 - calcular [96](#)
- valores de fecha/hora
 - adición [64](#)
- valores de retorno
 - descripción [15](#)
 - sintaxis [18](#)
- valores decimales
 - conversión [83](#), [102](#), [103](#), [216](#), [218](#), [222](#)
- valores del exponente
 - devolver [155](#)
- valores negativos
 - SIGN [189](#)
- valores nulos
 - comprobar [114](#)
 - condiciones de filtro [26](#)
 - en expresiones de comparación [25](#)
 - funciones de agregado [26](#), [56](#)
 - ISNULL [114](#)
 - operador de cadena [30](#)
 - operadores [27](#)
 - operadores lógicos [32](#)
- valores numéricos
 - conversión a cadenas de texto [211](#)
 - devolución de desviación estándar [196](#)
 - devolución del mínimo [144](#)
 - devolver coseno [78](#)
 - devolver coseno hiperbólico de [79](#)
 - devolver logaritmos [125](#), [126](#)
 - devolver raíz cuadrada [195](#)
 - devolver seno [190](#)
 - devolver seno hiperbólico [191](#)
 - devolver tangente [203](#)
 - devolver tangente hiperbólica [203](#)
 - devolver valor absoluto [62](#)

valores numéricos (*continuado*)

SIGN [189](#)

valores positivos

SIGN [189](#)

valores predeterminados

función ERROR [94](#)

variable PROC_RESULT

palabra reservada [20](#)

variable SESSSTARTTIME

descripción [37](#)

palabra reservada [20](#)

usar en funciones de fecha [53](#)

variable SYSDATE

descripción [38](#)

palabra reservada [20](#)

usar en expresiones [38](#)

Variable TC_COMMIT_AFTER

descripción [39](#)

Variable TC_COMMIT_BEFORE

descripción [39](#)

Variable TC_CONTINUE_TRANSACTION

descripción [39](#)

Variable TC_ROLLBACK_BEFORE

descripción [39](#)

variable WORKFLOWSTARTTIME

descripción [38](#)

palabra reservada [20](#)

usar en expresiones [38](#)

variables

\$PMFolderName [36](#)

\$PMIntegrationServiceName [36](#)

\$PMMappingName [36](#)

\$PMRepositoryServiceName [36](#)

variables (*continuado*)

\$PMRepositoryUserName [36](#)

\$PMSessionName [36](#)

\$PMSessionRunMode [37](#)

\$PMSourceName@TableName [35](#)

\$PMTargetName@TableName [35](#)

\$PMWorkflowName [37](#)

\$PMWorkflowRunId [37](#)

\$PMWorkflowRunInstanceName [37](#)

SESSSTARTTIME [37](#)

SYSDATE [38](#)

TC_COMMIT_AFTER [39](#)

TC_COMMIT_BEFORE [39](#)

TC_CONTINUE_TRANSACTION [39](#)

TC_ROLLBACK_BEFORE [39](#)

variables de control de transacción [39](#)

variables integradas [33](#)

WORKFLOWSTARTTIME [38](#)

variables de asignación

descripción [15](#)

variables integradas [33](#)

variables de control de transacción

descripción [39](#)

variables de flujo de trabajo

descripción [15](#)

variables del flujo de trabajo

variables integradas [33](#)

variables del sistema [33](#)

variables integradas

Descripción [33](#)

variables locales

descripción [15](#)