



Informatica®

9.6.1 HotFix 2

# Référence du langage de transformation

## Informatica Référence du langage de transformation

9.6.1 HotFix 2

Janvier 2015

© Copyright Informatica LLC 1998, 2018

Ce logiciel et sa documentation contiennent des informations appartenant à Informatica Corporation, protégées par la loi sur le droit d'auteur et fournies dans le cadre d'un accord de licence prévoyant des restrictions d'utilisation et de divulgation. Toute ingénierie inverse du logiciel est interdite. Il est interdit de reproduire ou transmettre sous quelque forme et par quelque moyen que ce soit (électronique, photocopie, enregistrement ou autre) tout ou partie de ce document sans le consentement préalable d'Informatica Corporation. Ce logiciel peut être protégé par des brevets américains et/ou internationaux, ainsi que par d'autres brevets en attente.

L'utilisation, la duplication ou la divulgation du Logiciel par le gouvernement américain est sujette aux restrictions décrites dans l'accord de licence applicable du logiciel conformément aux documents DFARS 227.7202-1(a) et 227.7702-3(a) (1995), DFARS 252.227-7013<sup>©</sup>(1)(ii) (OCT 1988), FAR 12.212(a) (1995), FAR 52.227-19 ou FAR 52.227-14 (ALT III) le cas échéant.

Les informations dans ce produit ou cette documentation sont sujettes à modification sans préavis. Si vous rencontrez des problèmes dans ce produit ou la documentation, veuillez nous en informer par écrit.

Informatica, Informatica Platform, Informatica Data Services, PowerCenter, PowerCenterRT, PowerCenter Connect, PowerCenter Data Analyzer, PowerExchange, PowerMart, Metadata Manager, Informatica Data Quality, Informatica Data Explorer, Informatica B2B Data Transformation, Informatica B2B Data Exchange Informatica On Demand, Informatica Identity Resolution, Informatica Application Information Lifecycle Management, Informatica Complex Event Processing, Ultra Messaging et Informatica Master Data Management sont des marques de commerce ou des marques déposées d'Informatica Corporation aux États-Unis et dans d'autres juridictions du monde. Tous les autres noms de société ou de produit peuvent être des marques de commerce ou des marques déposées de leurs détenteurs respectifs.

Des portions de ce logiciel et/ou de la documentation sont sujettes au copyright détenu par des tierces parties, dont Copyright DataDirect Technologies. Tous droits réservés. Copyright © Sun Microsystems. Tous droits réservés. Copyright © RSA Security Inc. Tous droits réservés. Copyright © Ordinal Technology Corp. Tous droits réservés. Copyright © Aandacht c.v. Tous droits réservés. Copyright Genivia, Inc. Tous droits réservés. Copyright Isomorphic Software. Tous droits réservés. Copyright © Meta Integration Technology, Inc. Tous droits réservés. Copyright © Intalio. Tous droits réservés. Copyright © Oracle. Tous droits réservés. Copyright © Adobe Systems Incorporated. Tous droits réservés. Copyright © DataArt, Inc. Tous droits réservés. Copyright © ComponentSource. Tous droits réservés. Copyright © Microsoft Corporation. Tous droits réservés. Copyright © Rogue Wave Software, Inc. Tous droits réservés. Copyright © Teradata Corporation. Tous droits réservés. Copyright © Yahoo! Inc. Tous droits réservés. Copyright © Glyph & Cog, LLC. Tous droits réservés. Copyright © Thinkmap, Inc. Tous droits réservés. Copyright © Clearpace Software Limited. Tous droits réservés. Copyright © Information Builders, Inc. Tous droits réservés. Copyright © OSS Nokalva, Inc. Tous droits réservés. Copyright Edifecs, Inc. Tous droits réservés. Copyright Cleo Communications, Inc. Tous droits réservés. Copyright © International Organization for Standardization 1986. Tous droits réservés. Copyright © ej-technologies GmbH. Tous droits réservés. Copyright © JasperSoft Corporation. Tous droits réservés. Copyright © International Business Machines Corporation. Tous droits réservés. Copyright © yWorks GmbH. Tous droits réservés. Copyright © Lucent Technologies. Tous droits réservés. Copyright © Université de Toronto. Tous droits réservés. Copyright © Daniel Veillard. Tous droits réservés. Copyright © Unicode, Inc. Copyright IBM Corp. Tous droits réservés. Copyright © MicroQuill Software Publishing, Inc. Tous droits réservés. Copyright © PassMark Software Pty Ltd. Tous droits réservés. Copyright © LogiXML, Inc. Tous droits réservés. Copyright © 2003-2010 Lorenzi Davide. Tous droits réservés. Copyright © Red Hat, Inc. Tous droits réservés. Copyright © The Board of Trustees of the Leland Stanford Junior University. Tous droits réservés. Copyright © EMC Corporation. Tous droits réservés. Copyright © Flexera Software. Tous droits réservés. Copyright © Jinfonet Software. Tous droits réservés. Copyright © Apple Inc. Tous droits réservés. Copyright © Telerik Inc. Tous droits réservés. Copyright © BEA Systems. Tous droits réservés. Copyright © PDFlib GmbH. Tous droits réservés. Copyright © Orientation in Objects GmbH. Tous droits réservés. Copyright © Tanuki Software, Ltd. Tous droits réservés. Copyright © Ricebridge. Tous droits réservés. Copyright © Sencha, Inc. Tous droits réservés. Copyright © Scalable Systems, Inc. Tous droits réservés. Copyright © jQWidgets. Tous droits réservés.

Ce produit inclut des logiciels développés par Apache Software Foundation (<http://www.apache.org/>), et/ou d'autres logiciels sous licence et sous diverses versions Apache License (la « Licence »). Vous pouvez obtenir une copie de ces licences à l'adresse suivante : <http://www.apache.org/licenses/>. Sauf dispositions contraires de la loi en vigueur ou accord écrit, le logiciel distribué sous cette licence est livré « EN L'ÉTAT », SANS GARANTIE NI CONDITION D'AUCUNE SORTE, expresse ou implicite. Se reporter aux Licences pour la langue spécifique régissant les droits et limitations dans le cadre des Licences.

Ce produit inclut des logiciels développés par Mozilla (<http://www.mozilla.org/>), copyright de logiciel The JBoss Group, LLC, tous droits réservés ; copyright de logiciel © 1999-2006 de Bruno Lowagie et Paulo Soares et d'autres logiciels sous licence et sous diverses versions du GNU Lesser General Public License Agreement, accessible sur <http://www.gnu.org/licenses/lgpl.html>. Les matériaux sont fournis gratuitement par Informatica, « en l'état », sans garantie d'aucune sorte, expresse ou implicite, notamment les garanties implicites de conformité légale et d'usage normal.

Le produit inclut les logiciels ACE(TM) et TAO(TM), copyright Douglas C. Schmidt et son groupe de recherche à Washington University, University of California, Irvine, et Vanderbilt University, Copyright (©) 1993-2006, tous droits réservés.

Ce produit inclut des logiciels développés par OpenSSL Project pour une utilisation dans OpenSSL Toolkit (copyright The OpenSSL Project. Tous droits réservés) et la redistribution de ce logiciel est sujette aux termes publiés sur <http://www.openssl.org> et <http://www.openssl.org/source/license.html>.

Ce produit inclut le logiciel Curl, copyright 1996-2013, Daniel Stenberg, <daniel@haxx.se>. Tous Droits Réservés. Les autorisations et limitations concernant ce logiciel sont sujettes aux conditions publiées sur <http://curl.haxx.se/docs/copyright.html>. L'autorisation d'utiliser, copier, modifier et distribuer ce logiciel à toute fin, avec ou sans rémunération, est accordée par les présentes, à la condition que la notification de copyright ci-dessus et cette notification d'autorisation apparaissent dans toutes les copies.

Le produit inclut des logiciels sous copyright 2001-2005 (©) MetaStuff, Ltd. Tous droits réservés. Les autorisations et limitations concernant ce logiciel sont sujettes aux conditions publiées sur <http://www.dom4j.org/license.html>.

Le produit inclut des logiciels sous copyright © 2004-2007, The Dojo Foundation. Tous Droits Réservés. Les autorisations et limitations concernant ce logiciel sont sujettes aux conditions publiées sur <http://dojotoolkit.org/license>.

Ce produit inclut le logiciel ICU sous copyright de International Business Machines Corporation et autres. Tous Droits Réservés. Les autorisations et limitations concernant ce logiciel sont sujettes aux conditions publiées sur <http://source.icu-project.org/repos/icu/icu/trunk/license.html>.

Ce produit inclut des logiciels sous copyright © 1996-2006 Per Bothner. Tous Droits Réservés. Votre droit à utiliser de tels matériels est défini dans la licence qui peut être consultée sur <http://www.gnu.org/software/kawa/Software-License.html>.

Ce produit inclut le logiciel OSSP UUID sous copyright © 2002 Ralf S. Engelschall, copyright © 2002 The OSSP Project Copyright © 2002 Cable & Wireless Deutschland. Les autorisations et limitations concernant ce logiciel sont sujettes aux conditions publiées sur <http://www.opensource.org/licenses/mit-license.php>.

Ce produit inclut des logiciels développés par Boost (<http://www.boost.org/>) ou sous licence de logiciel Boost. Les autorisations et limitations concernant ce logiciel sont sujettes aux conditions publiées sur [http://www.boost.org/LICENSE\\_1\\_0.txt](http://www.boost.org/LICENSE_1_0.txt).

Ce produit inclut des logiciels sous copyright © 1997-2007 University of Cambridge. Les autorisations et limitations concernant ce logiciel sont sujettes aux conditions publiées sur <http://www.pcre.org/license.txt>.

Ce produit inclut des logiciels sous copyright © 2007 The Eclipse Foundation. Tous Droits Réservés. Les autorisations et limitations concernant ce logiciel sont sujettes aux conditions publiées sur <http://www.eclipse.org/org/documents/epl-v10.php> et <http://www.eclipse.org/org/documents/edl-v10.php>.

Ce produit comprend des logiciels sous licence dont les conditions se trouvent aux adresses : <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/hsqldbLicense.html>, <http://httpunit.sourceforge.net/doc/license.html>, <http://jung.sourceforge.net/license.txt>, [http://www.gzip.org/zlib/zlib\\_license.html](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://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://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>; et <https://code.google.com/p/lz4/>.

Ce produit inclut un logiciel sous licence Academic Free License (<http://www.opensource.org/licenses/afl-3.0.php>), licence Common Development Distribution License (<http://www.opensource.org/licenses/cddl1.php>) licence Common Public License (<http://www.opensource.org/licenses/cpl1.0.php>), licence Sun Binary Code License Agreement Supplemental License Terms, licence BSD (<http://www.opensource.org/licenses/bsd-license.php>), le nouvelle licence BSD License (<http://opensource.org/licenses/BSD-3-Clause>), la licence MIT (<http://www.opensource.org/licenses/mit-license.php>), la licence Artistic License (<http://www.opensource.org/licenses/artistic-license-1.0>) et la licence publique du développeur initial Version 1.0 (<http://www.firebirdsql.org/en/initial-developer-s-public-license-version-1-0/>).

Ce produit inclut des logiciels sous copyright © 2003-2006 Joe Walnes, 2006-2007 XStream Committers. Tous Droits Réservés. Les autorisations et limitations concernant ce logiciel sont sujettes aux conditions publiées sur <http://xstream.codehaus.org/license.html>. Ce produit inclut des logiciels développés par Indiana University Extreme! Lab. Pour plus d'informations, veuillez vous rendre sur <http://www.extreme.indiana.edu/>.

Ce produit inclut des logiciels sous copyright © 2013 Frank Balluffi et Markus Moeller. Tous droits réservés. Les autorisations et limitations concernant ce logiciel sont sujettes aux conditions de la licence MIT.

Ce logiciel est protégé par des brevets américains (5,794,246; 6,014,670; 6,016,501; 6,029,178; 6,032,158; 6,035,307; 6,044,374; 6,092,086; 6,208,990; 6,339,775; 6,640,226; 6,789,096; 6,823,373; 6,850,947; 6,895,471; 7,117,215; 7,162,643; 7,243,110; 7,254,590; 7,281,001; 7,421,458; 7,496,588; 7,523,121; 7,584,422; 7,676,516; 7,720,842; 7,721,270; 7,774,791; 8,065,266; 8,150,803; 8,166,048; 8,166,071; 8,200,622; 8,224,873; 8,271,477; 8,327,419; 8,386,435; 8,392,460; 8,453,159; 8,458,230; 8,707,336; 8,886,617 et RE44,478), des brevets internationaux ainsi que par d'autres brevets en attente.

**EXCLUSION DE RESPONSABILITÉ** : Informatica Corporation fournit cette documentation « en l'état », sans garantie d'aucune sorte, explicite ou implicite, notamment les garanties implicites de non-infraction, de conformité légale ou d'usage normal. Informatica Corporation ne garantit pas que ce logiciel et cette documentation sont exempts d'erreurs. Les informations fournies dans ce logiciel ou cette documentation peuvent inclure des inexactitudes techniques ou des erreurs typographiques. Les informations contenues dans ce logiciel et sa documentation sont sujettes à modification à tout moment sans préavis.

#### AVIS

Ce produit Informatica (le « Logiciel ») inclut certains pilotes (les « Pilotes DataDirect ») de DataDirect Technologies, une société de Progress Software Corporation (« DataDirect ») qui sont sujets aux conditions suivantes :

1. LES PILOTES DATADIRECT SONT FOURNIS « EN L'ÉTAT », SANS GARANTIE D'AUCUNE SORTE, EXPRESSE OU IMPLICITE, NOTAMMENT LES GARANTIES IMPLICITES DE CONFORMITÉ LÉGALE, D'USAGE NORMAL ET DE NON-INFRACTION.
2. DATADIRECT OU SES FOURNISSEURS TIERS NE POURRONT EN AUCUN CAS ÊTRE TENUS RESPONSABLES ENVERS LE CLIENT UTILISATEUR FINAL DE TOUT DOMMAGE DIRECT, ACCESSOIRE, INDIRECT, SPÉCIAL, CONSÉCUTIF OU AUTRE RÉSULTANT DE L'UTILISATION DES PILOTES ODBC, QU'ILS SOIENT INFORMÉS OU NON À L'AVANCE DE LA POSSIBILITÉ DE TELS DOMMAGES. CES LIMITATIONS S'APPLIQUENT À TOUTES LES CAUSES D'ACTION, NOTAMMENT TOUTE INFRACTION AU CONTRAT, INFRACTION À LA GARANTIE, NÉGLIGENCE, RESPONSABILITÉ STRICTE, REPRÉSENTATION INCORRECTE ET AUTRES TORTS.

Date de publication: 2018-10-31

# Sommaire

<b>Préface.....</b>	<b>9</b>
Ressources Informatica.....	9
Portail Mon support Informatica.....	9
Documentation Informatica.....	9
Matrices de disponibilité de produit Informatica.....	9
Site Web Informatica.....	10
Bibliothèque de procédures Informatica.....	10
Base de connaissances Informatica.....	10
Canal YouTube du support Informatica.....	10
Informatica Marketplace.....	10
Informatica Velocity.....	10
Support client international Informatica.....	11
<b>Chapitre 1: Le langage de transformation.....</b>	<b>12</b>
Présentation du langage de transformation.....	12
Composants du langage de transformation.....	12
Internationalisation et le langage de transformation.....	13
Syntaxe d'expression.....	13
Composants de l'expression.....	14
Règles et directives pour la syntaxe d'expression.....	15
Ajout de commentaires aux expressions.....	15
Mots réservés.....	16
<b>Chapitre 2: Constantes.....</b>	<b>18</b>
DD_DELETE.....	18
Exemple.....	18
DD_INSERT.....	19
Exemples.....	19
DD_REJECT.....	19
Exemples.....	19
DD_UPDATE.....	20
Exemples.....	20
FALSE.....	20
Exemple.....	21
NUL.....	21
Utilisation des valeurs nulles dans des expressions booléennes.....	21
Utilisation des valeurs nulles dans des expressions de comparaison.....	21
Valeurs nulles dans des fonctions Agrégation.....	21
Valeurs nulles dans des conditions de filtre.....	22
Valeurs nulles avec les opérateurs.....	22

TRUE. . . . .	22
Exemple. . . . .	22
<b>Chapitre 3: Opérateurs. . . . .</b>	<b>23</b>
Priorité des opérateurs. . . . .	23
Opérateurs arithmétiques. . . . .	24
Opérateurs de chaîne. . . . .	25
Valeurs nulles. . . . .	25
Exemple. . . . .	25
Opérateurs de comparaison. . . . .	26
Opérateurs logiques. . . . .	26
Valeurs nulles. . . . .	27
<b>Chapitre 4: Variables. . . . .</b>	<b>28</b>
Variables intégrées. . . . .	28
SYSDATE. . . . .	28
Variables locales. . . . .	28
<b>Chapitre 5: Dates. . . . .</b>	<b>30</b>
Présentation des dates. . . . .	30
Type de données Date/Heure. . . . .	30
Date ordinale, date ordinale modifiée et calendrier grégorien. . . . .	31
Dates en l'an 2000. . . . .	31
Dates dans des bases de données relationnelles. . . . .	33
Dates dans des fichiers plats. . . . .	33
Format de date par défaut. . . . .	33
Chaînes de format de date. . . . .	34
Chaînes de format TO_CHAR. . . . .	35
Exemples. . . . .	37
Les chaînes de format TO_DATE et IS_DATE. . . . .	38
Règles et directives pour les chaînes de format de date. . . . .	40
Exemple. . . . .	41
Comprendre l'arithmétique de date. . . . .	42
<b>Chapitre 6: Fonctions. . . . .</b>	<b>43</b>
Catégories de fonctions. . . . .	43
Fonctions d'agrégation. . . . .	43
Fonctions Agrégation et valeurs nulles. . . . .	45
Fonctions de caractères. . . . .	45
Fonctions de conversion. . . . .	46
Fonctions de nettoyage des données. . . . .	46
Fonctions de date. . . . .	47
Fonctions de codage. . . . .	48

Fonctions financières. . . . .	48
Fonctions numériques. . . . .	48
Fonctions scientifiques. . . . .	49
Fonctions spéciales. . . . .	49
Fonctions de chaîne. . . . .	50
Fonctions de test. . . . .	50
ABANDONNER. . . . .	50
ABS. . . . .	51
ADD_TO_DATE. . . . .	52
AES_DECRYPT. . . . .	55
AES_ENCRYPT. . . . .	55
ANY. . . . .	56
ASCII. . . . .	57
AVG. . . . .	58
CEIL. . . . .	60
CHOOSE. . . . .	61
CHR. . . . .	62
CHRCODE. . . . .	63
COMPRESS. . . . .	64
CONCAT. . . . .	64
CONVERT_BASE. . . . .	66
COS. . . . .	66
COSH. . . . .	67
COUNT. . . . .	68
CRC32. . . . .	71
CUME. . . . .	71
DATE_COMPARE. . . . .	73
DATE_DIFF. . . . .	74
DEC_BASE64. . . . .	77
DECODE. . . . .	77
DECOMPRESS. . . . .	80
ENC_BASE64. . . . .	80
ERREUR. . . . .	81
EXP. . . . .	82
FIRST. . . . .	83
FLOOR. . . . .	84
FV. . . . .	85
GET_DATE_PART. . . . .	86
GREATEST. . . . .	88
IIF. . . . .	89
IN. . . . .	92
INDEXOF. . . . .	93

INITCAP. . . . .	94
INSTR. . . . .	95
ISNULL. . . . .	98
IS_DATE. . . . .	99
IS_NUMBER. . . . .	101
IS_SPACES. . . . .	103
LAST. . . . .	104
LAST_DAY. . . . .	105
LEAST. . . . .	107
LONGUEUR. . . . .	108
LN. . . . .	109
JOURNAL. . . . .	109
LOWER. . . . .	110
LPAD. . . . .	111
LTRIM. . . . .	113
MAKE_DATE_TIME. . . . .	114
MAX (Dates). . . . .	115
MAX (Nombres). . . . .	116
MAX (Chaîne). . . . .	118
MD5. . . . .	119
MEDIAN. . . . .	120
METAPHONE. . . . .	121
MIN (Dates). . . . .	125
MIN (Nombres). . . . .	126
MIN (Chaîne). . . . .	127
MOD. . . . .	129
MOVINGAVG. . . . .	130
MOVINGSUM. . . . .	132
NPER. . . . .	133
PERCENTILE. . . . .	134
PMT. . . . .	136
POWER. . . . .	136
PV. . . . .	137
RAND. . . . .	138
RATE. . . . .	139
REG_EXTRACT. . . . .	140
REG_MATCH. . . . .	142
REG_REPLACE. . . . .	144
REPLACECHR. . . . .	145
REPLACESTR. . . . .	148
REVERSE. . . . .	151
ROUND (Dates). . . . .	152

ROUND (Nombres) . . . . .	156
RPAD . . . . .	159
RTRIM . . . . .	160
SET_DATE_PART . . . . .	161
SIGN . . . . .	164
SIN . . . . .	165
SINH . . . . .	166
SOUNDEX . . . . .	167
SQRT . . . . .	169
STDDEV . . . . .	170
SUBSTR . . . . .	172
SUM . . . . .	174
SYSTIMESTAMP . . . . .	175
TAN . . . . .	177
TANH . . . . .	177
TO_BIGINT . . . . .	178
TO_CHAR (Dates) . . . . .	180
TO_CHAR (Nombres) . . . . .	185
TO_DATE . . . . .	186
TO_DECIMAL . . . . .	190
TO_FLOAT . . . . .	191
TO_INTEGER . . . . .	192
TRUNC (Dates) . . . . .	193
TRUNC (Nombres) . . . . .	196
UPPER . . . . .	198
UUID4 . . . . .	199
UUID_UNPARSE . . . . .	199
VARIANCE . . . . .	200
<b>Index . . . . .</b>	<b>202</b>

# Préface

La *Informatica Developer PowerCenter* est écrite pour les développeurs responsables de la construction des mappages. La *Informatica Developer PowerCenter* considère que vous connaissez SQL, les concepts des bases de données relationnelles et les exigences d'interface pour vos applications de support.

## Ressources Informatica

### Portail Mon support Informatica

En tant que client Informatica, vous pouvez accéder au portail Mon support Informatica sur <http://mysupport.informatica.com>.

Ce site contient des informations sur les produits et les groupes d'utilisateurs, des bulletins d'information, un lien vers le système de gestion des dossiers d'assistance à la clientèle d'Informatica (ATLAS), une bibliothèque de procédures Informatica, une base de connaissances Informatica, ainsi que la documentation nécessaire sur les produits Informatica et l'accès à sa communauté d'utilisateurs.

### Documentation Informatica

L'équipe Documentation d'Informatica s'efforce de fournir une documentation précise et utilisable. N'hésitez pas à contacter l'équipe Documentation d'Informatica par courriel à l'adresse [infa\\_documentation@informatica.com](mailto:infa_documentation@informatica.com) pour lui faire part de vos questions, commentaires ou suggestions concernant cette documentation. Ces commentaires et suggestions nous permettront d'améliorer notre documentation. Veuillez préciser si vous acceptez d'être contacté au sujet de ces commentaires.

L'équipe Documentation met à jour la documentation chaque fois que nécessaire. Pour obtenir la toute dernière version de la documentation concernant votre produit, consultez la Documentation de produit sur <http://mysupport.informatica.com>.

### Matrices de disponibilité de produit Informatica

Les matrices de disponibilité de produit (PAM) indiquent les versions des systèmes d'exploitation, les bases de données et les autres types de sources et cibles de données pris en charge par une version d'un produit. Vous pouvez consulter les PAM sur le portail Mon Support Informatica à l'adresse <https://mysupport.informatica.com/community/my-support/product-availability-matrices>.

## Site Web Informatica

Vous pouvez accéder au site Web d'entreprise Informatica sur <http://www.informatica.com>. Le site contient des informations sur Informatica, son expertise, les événements à venir et les bureaux de vente. Vous y trouverez aussi des informations sur ses produits et ses partenaires. Les rubriques de service du site fournissent des informations importantes sur le support technique, la formation et l'éducation, ainsi que les services d'implémentation.

## Bibliothèque de procédures Informatica

En tant que client Informatica, vous avez accès à la bibliothèque de procédures Informatica sur <http://mysupport.informatica.com>. La bibliothèque de procédures Informatica est une collection de ressources destinée à vous familiariser avec les produits Informatica et leurs fonctionnalités. Elle regroupe des articles et des démonstrations interactives qui permettent de résoudre des problèmes courants et de comparer les fonctionnalités et les comportements, et qui vous guident lors de la réalisation de tâches concrètes spécifiques.

## Base de connaissances Informatica

En tant que client Informatica, vous avez accès à la base de connaissances Informatica sur <http://mysupport.informatica.com>. Utilisez la base de connaissances pour rechercher des solutions documentées aux problèmes techniques connus concernant les produits Informatica. Vous y trouverez également la réponse aux questions les plus fréquentes, des livres blancs et des conseils techniques. N'hésitez pas à contacter l'équipe Base de connaissances Informatica par courriel à l'adresse [KB\\_Feedback@informatica.com](mailto:KB_Feedback@informatica.com) pour lui faire part de vos questions, commentaires et suggestions concernant la base de connaissances.

## Canal YouTube du support Informatica

Vous pouvez accéder au canal YouTube du support Informatica sur <http://www.youtube.com/user/INFASupport>. Le canal YouTube du support Informatica contient des vidéos concernant les solutions qui vous guident dans l'exécution de tâches spécifiques. Si vous avez des questions, commentaires ou suggestions concernant le canal YouTube du support Informatica, contactez l'équipe de support YouTube par courriel à l'adresse [supportvideos@informatica.com](mailto:supportvideos@informatica.com) ou envoyez un tweet à @INFASupport.

## Informatica Marketplace

Informatica Marketplace est un forum où développeurs et partenaires peuvent partager des solutions qui permettent d'augmenter, d'étendre ou d'améliorer les implémentations d'intégration de données. En tirant profit des centaines de solutions disponibles sur Marketplace, vous pouvez améliorer votre productivité et accélérer le temps d'implémentation de vos projets. Vous pouvez accéder à Informatica Marketplace à l'adresse <http://www.informaticamarketplace.com>.

## Informatica Velocity

Vous pouvez accéder à Informatica Velocity à l'adresse <http://mysupport.informatica.com>. Développé à partir de l'expérience concrète de centaines de projets de gestion de données, Informatica Velocity représente le savoir collectif de nos consultants, qui ont travaillé avec des entreprises du monde entier pour planifier, développer, déployer et tenir à jour des solutions de gestion des données efficaces. Si vous avez des questions, des commentaires et des suggestions sur Informatica Velocity, contactez le support des services professionnels Informatica à l'adresse [ips@informatica.com](mailto:ips@informatica.com).

## Support client international Informatica

Vous pouvez contacter un centre de support client par téléphone ou via l'assistance en ligne.

L'assistance en ligne requiert un nom d'utilisateur et un mot de passe. Vous pouvez demander un nom d'utilisateur et un mot de passe sur <http://mysupport.informatica.com>.

Les numéros de téléphone du support client international Informatica sont disponibles sur le site Web Informatica à l'adresse

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

# CHAPITRE 1

## Le langage de transformation

Ce chapitre comprend les rubriques suivantes :

- [Présentation du langage de transformation, 12](#)
- [Syntaxe d'expression, 13](#)
- [Ajout de commentaires aux expressions, 15](#)
- [Mots réservés, 16](#)

### Présentation du langage de transformation

Informatica Developer fournit un langage de transformation comprenant des fonctions semblables à SQL pour transformer les données source. Utilisez ces fonctions pour écrire des expressions.

Les expressions modifient les données ou testent si les données remplissent les conditions. Par exemple, vous pouvez utiliser la fonction AVG pour calculer le salaire moyen de tous les employés ou la fonction SUM pour calculer le total des ventes pour une branche spécifique.

Vous pouvez créer une expression simple qui contient uniquement un port, par exemple ORDERS, ou un littéral numérique, tel que 10. Vous pouvez également écrire des expressions complexes comprenant des fonctions imbriquées dans d'autres fonctions ou combinant des ports à l'aide des opérateurs du langage de transformation.

### Composants du langage de transformation

Le langage de transformation comprend les composants suivants pour créer des expressions de transformation simples ou complexes :

- **Fonctions.** Plus de 100 fonctions similaires à SQL vous permettent de modifier les données dans un mappage.
- **Opérateurs.** Utilisez des opérateurs de transformation pour créer des expressions de transformation permettant d'exécuter des calculs mathématiques, de combiner des données ou de comparer des données.
- **Constantes.** Utilisez des constantes intégrées pour référencer des valeurs demeurant constantes, par exemple TRUE.
- **Paramètres de mappage.** Créez des paramètres de mappage à utiliser dans un mappage ou un mapplet pour référencer des valeurs qui demeurent constantes tout au long de l'exécution du mappage ou du mapplet, par exemple le taux de TVA d'un Etat.

- **Variables intégrées et locales.** Utilisez des variables intégrées pour écrire des expressions qui référencent des valeurs changeantes, par exemple la date système. Vous pouvez également créer des variables locales dans les transformations.
- **Valeurs de retour.** Vous pouvez aussi écrire des expressions qui comprennent les valeurs de retour des transformations Recherche.

## Internationalisation et le langage de transformation

Les fonctions du langage de transformation peuvent gérer des données de caractères en mode mouvement de données ASCII ou Unicode. Utilisez le mode Unicode pour gérer des données de caractères *multioctets*. Les valeurs de retour des fonctions et transformations suivantes dépendent de la page de code du Service d'intégration de données et du mode de mouvement de données :

- INITCAP
- LOWER
- UPPER
- MIN (Date)
- MIN (Nombre)
- MIN (Chaîne)
- MAX (Date)
- MAX (Nombre)
- MAX (Chaîne)
- Toute fonction utilisant les instructions conditionnelles pour comparer des chaînes, par exemple IIF et DECODE

MIN et MAX renvoient aussi des valeurs en fonction de l'ordre de tri associé à la page de code du Service d'intégration de données.

Lorsque vous validez une expression non valide dans l'Éditeur d'expressions, une boîte de dialogue affiche l'expression avec un indicateur d'erreur, ">>>>". Cet indicateur apparaît à gauche de l'expression et pointe vers la partie de l'expression contenant l'erreur. Par exemple, si l'expression `a = b + c` contient une erreur à `c`, le message d'erreur affiche :

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

Les fonctions du langage de transformation qui évaluent les données de caractères sont orientées caractère, pas orientées octet. Par exemple, la fonction LENGTH renvoie le nombre de caractères dans une chaîne et non le nombre d'octets. La fonction LOWER renvoie une chaîne en minuscule en fonction de la page de code du Service d'intégration de données.

## Syntaxe d'expression

Bien que le langage de transformation soit basé sur le SQL standard, il existe des différences entre les deux langages. Par exemple, SQL prend en charge les mots-clés ALL et DISTINCT pour les fonctions Agrégation contrairement au langage de transformation. En revanche, le langage de transformation prend en charge une condition de filtre facultative pour les fonctions Agrégation contrairement à SQL.

Vous pouvez créer une expression aussi simple qu'un port (comme ORDERS) ou un littéral numérique (comme 10). Vous pouvez également écrire des expressions complexes comprenant des fonctions

imbriquées dans d'autres fonctions ou combinant différentes colonnes à l'aide des opérateurs du langage de transformation.

## Composants de l'expression

Les expressions peuvent comporter n'importe quelle combinaison des composants suivants :

- Ports (entrée, entrée/sortie, variable)
- Littéraux de chaîne, littéraux numériques
- Constantes
- Fonctions
- Variables locales et intégrées
- Paramètres de mappage
- Opérateurs
- Valeurs de retour

### Ports et valeurs de retour

Lorsque vous écrivez une expression comprenant un port ou la valeur de retour d'une transformation non connectée, utilisez les qualificatifs de référence du tableau suivant :

Qualificateur de référence	Description
:LKP	Requis lorsque vous créez une expression comprenant la valeur de retour d'une transformation Recherche non connectée. La syntaxe générale est :  <code>:LKP.lookup_transformation(argument1, argument2, ...)</code>  Les arguments sont les ports locaux utilisés dans la condition de recherche. L'ordre doit correspondre à l'ordre des ports dans la transformation. Les types de données pour les ports locaux doivent correspondre aux types de données des ports de recherche utilisés dans la condition de recherche.

### Littéraux de chaîne et littéraux numériques

Vous pouvez inclure des littéraux numériques ou des littéraux de chaîne.

Veillez à placer les littéraux de chaîne entre guillemets simples. Par exemple :

```
'Alice Davis'
```

Les littéraux de chaîne sont sensibles à la casse et peuvent contenir n'importe quel caractère à l'exception d'un guillemet simple. Par exemple, la chaîne suivante n'est pas autorisée :

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

Pour renvoyer une chaîne contenant un guillemet simple, utilisez la fonction CHR :

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

N'utilisez pas de guillemets simples avec les littéraux numériques. Entrez uniquement le nombre que vous voulez inclure. Par exemple :

```
.05
```

ou

```
$$Sales_Tax
```

## Règles et directives pour la syntaxe d'expression

Utilisez les règles et directives suivantes lorsque vous écrivez des expressions :

- Vous ne pouvez pas inclure à la fois des fonctions à un seul niveau et des fonctions imbriquées dans une transformation Agrégation.
- Si vous devez créer à la fois des fonctions à un seul niveau et des fonctions imbriquées, créez des transformations Agrégation séparées.
- Vous ne pouvez pas utiliser les chaînes dans des expressions numériques.  
Par exemple, l'expression `1 + '1'` n'est pas valide, car vous ne pouvez exécuter une addition qu'avec des types de données numériques. Vous ne pouvez pas ajouter un entier et une chaîne.
- Vous ne pouvez pas utiliser des chaînes comme paramètres numériques.  
Par exemple, l'expression `SUBSTR(TEXT_VAL, '1', 10)` n'est pas valide, car la fonction SUBSTR requiert une valeur de type entier et non une chaîne pour la position de démarrage.
- Vous ne pouvez pas mélanger les types de données quand vous utilisez des opérateurs de comparaison.  
Par exemple, l'expression `123.4 = '123.4'` n'est pas valide, car elle compare une valeur décimale à une chaîne.
- Vous pouvez transmettre une valeur venant d'un port, d'une chaîne littérale ou d'un nombre, d'une transformation Recherche ou les résultats d'une autre expression.
- Utilisez l'onglet Ports dans l'Editeur d'expressions pour entrer un nom de port dans une expression. Si vous renommez un port dans une transformation connectée, l'outil Developer propage le changement de nom dans les expressions de la transformation.
- Séparez chaque argument dans une fonction par une virgule.
- A l'exception des littéraux, le langage de transformation n'est pas sensible à la casse.
- A l'exception des littéraux, l'outil Developer et le service d'intégration de données ignorent les espaces.
- Les deux-points (:), la virgule (,) et le point (.) ont une signification spéciale et doivent seulement être utilisés pour spécifier la syntaxe.
- Le Service d'intégration de données traite le tiret (-) comme un opérateur de soustraction.
- Si vous transmettez une valeur littérale à une fonction, placez les chaînes littérales entre guillemets simples. N'utilisez pas de guillemets pour les nombres littéraux. Le Service d'intégration de données traite toute valeur de chaîne entre guillemets simples comme une chaîne de caractères.
- Lorsque vous transmettez un paramètre de mappage à une fonction dans une expression, n'utilisez pas de guillemets pour désigner les paramètres de mappage.
- N'utilisez pas de guillemets pour désigner les ports.
- Vous pouvez imbriquer plusieurs fonctions dans une expression à l'exception des fonctions Agrégation qui n'autorisent qu'une seule fonction d'agrégation imbriquée. Le Service d'intégration de données évalue l'expression en commençant par la fonction la plus interne.

## Ajout de commentaires aux expressions

Le langage de transformation fournit deux indicateurs de commentaire vous permettant d'insérer des commentaires dans les expressions :

- Le double tiret, comme dans :  
`-- These are comments`

- Le double slash, comme dans :

```
// These are comments
```

Le Service d'intégration de données ignore l'intégralité du texte précédé par ces deux indicateurs de commentaire sur une ligne. Par exemple, si vous voulez concaténer deux chaînes, vous pouvez entrer l'expression suivante avec des commentaires à l'intérieur de l'expression :

```
-- 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
```

Le Service d'intégration de données ignore les commentaires et évalue l'expression comme suit :

```
FIRST_NAME || LAST_NAME
```

Vous ne pouvez pas poursuivre un commentaire sur une nouvelle ligne :

```
-- 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
```

Dans ce cas, l'outil Developer ne valide pas l'expression, car la dernière ligne n'est pas une expression valide.

Si vous ne voulez pas de commentaires incorporés, vous pouvez les ajouter en cliquant sur Commentaire dans l'Editeur d'expressions.

## Mots réservés

Certains mots-clés du langage de transformation, tels que les constantes, les opérateurs et les variables intégrées sont réservés pour des fonctions spécifiques. Ils comprennent :

- :INFA
- :LKP
- :MCR
- AND
- DD\_DELETE
- DD\_INSERT
- DD\_REJECT
- DD\_UPDATE
- FALSE
- NOT
- NUL
- OU
- PROC\_RESULT
- SPOUTPUT
- SYSDATE
- TRUE

**Remarque:** Vous ne pouvez pas utiliser un mot réservé pour nommer un port ou une variable locale. Vous ne pouvez utiliser des mots réservés que dans les expressions de transformation. Les mots réservés ont des significations prédéfinies dans les expressions.

# CHAPITRE 2

## Constantes

Ce chapitre comprend les rubriques suivantes :

- [DD\\_DELETE, 18](#)
- [DD\\_INSERT, 19](#)
- [DD\\_REJECT, 19](#)
- [DD\\_UPDATE, 20](#)
- [FALSE, 20](#)
- [NUL, 21](#)
- [TRUE, 22](#)

## DD\_DELETE

Marque les enregistrements à supprimer dans une expression de stratégie de mise à jour. DD\_DELETE est équivalent au littéral entier 2.

**Remarque:** Utilisez la constante DD\_DELETE uniquement dans la transformation Stratégie de mise à jour. Utilisez DD\_DELETE au lieu du littéral entier 2 pour faciliter le débogage des expressions numériques complexes.

### Exemple

L'expression suivante identifie pour la suppression les éléments ayant un numéro d'identifiant de 1001 et identifie tous les autres éléments pour l'insertion :

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

Cette expression de stratégie de mise à jour utilise des littéraux numériques pour produire le même résultat :

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

**Remarque:** L'expression utilisant des constantes est plus simple à lire que l'expression utilisant des littéraux numériques.

# DD\_INSERT

Identifie les enregistrements pour l'insertion dans une expression de stratégie de mise à jour. DD\_INSERT équivaut au littéral entier 0.

**Remarque:** Utilisez la constante DD\_INSERT uniquement dans la transformation Stratégie de mise à jour. Utilisez DD\_INSERT au lieu du littéral entier 0 pour faciliter le dépannage des expressions numériques complexes.

## Exemples

Les exemples suivants modifient un mappage qui calcule les ventes mensuelles par vendeur pour vous permettre d'examiner les ventes effectuées par un seul vendeur.

L'expression de stratégie de mise à jour identifie les ventes d'un employé pour l'insertion et rejette tous les autres éléments :

```
IIF( EMPLOYEE.NAME = 'Alex', DD_INSERT, DD_REJECT )
```

Cette expression de stratégie de mise à jour utilise des littéraux numériques pour produire le même résultat :

```
IIF( EMPLOYEE.NAME = 'Alex', 0, 3 )
```

**Astuce:** L'expression utilisant des constantes est plus simple à lire que l'expression utilisant des littéraux numériques.

# DD\_REJECT

Identifie les enregistrements pour le rejet dans une expression de stratégie de mise à jour. DD\_REJECT équivaut au littéral entier 3.

**Remarque:** Utilisez la constante DD\_REJECT uniquement dans la transformation Stratégie de mise à jour. Utilisez DD\_REJECT au lieu du littéral entier 3 pour faciliter le dépannage des expressions numériques complexes.

Utilisez DD\_REJECT pour filtrer ou valider les données. Si vous marquez un enregistrement comme rejeté, le Service d'intégration de données ignore l'enregistrement et l'écrit dans le fichier de rejet de la session.

## Exemples

Les exemples suivants modifient un mappage qui calcule les ventes du mois en cours pour qu'il ne comprenne que des valeurs positives.

Cette expression de stratégie de mise à jour identifie les enregistrements inférieurs à 0 pour le rejet et tous les autres pour l'insertion :

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

Cette expression utilise des littéraux numériques pour produire le même résultat :

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

L'expression utilisant des constantes est plus simple à lire que l'expression utilisant des littéraux numériques.

L'exemple orienté données ci-dessous utilise DD\_REJECT et IS\_SPACES pour éviter l'écriture d'espaces dans une colonne de caractères d'une table cible. Cette expression marque les enregistrements qui sont entièrement constitués d'espaces pour le rejet et marque tous les autres pour l'insertion :

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

## DD\_UPDATE

Marque les enregistrements pour la mise à jour dans une expression de stratégie de mise à jour. DD\_UPDATE équivaut au littéral entier 1.

**Remarque:** Utilisez la constante DD\_UPDATE uniquement dans la transformation Stratégie de mise à jour. Utilisez DD\_UPDATE au lieu du littéral entier 1 pour faciliter le débogage des expressions numériques complexes.

## Exemples

Les exemples suivants modifient un mappage qui calcule les ventes du mois en cours. Le mappage charge les ventes d'un employé.

Cette expression marque les enregistrements pour Alex pour la mise à jour et marque tous les autres pour le rejet :

```
IIF( EMPLOYEE.NAME = 'Alex', DD_UPDATE, DD_REJECT )
```

Cette expression utilise des littéraux numériques pour produire le même résultat, en marquant les ventes d'Alex pour la mise à jour (1) et en marquant tous les autres enregistrements de ventes pour le rejet (3) :

```
IIF( EMPLOYEE.NAME = 'Alex', 1, 3 )
```

L'expression utilisant des constantes est plus simple à lire que l'expression utilisant des littéraux numériques.

L'expression de stratégie de mise à jour suivante utilise SYSDATE pour trouver uniquement les commandes qui ont été expédiées dans les deux derniers jours et les marque pour l'insertion. En utilisant DATE\_DIFF, l'expression soustrait DATE\_SHIPPED de la date système et renvoie la différence entre les deux dates. DATE\_DIFF renvoyant une valeur Double, l'expression utilise TRUNC pour tronquer la différence. Elle compare ensuite le résultat au littéral entier 2. Si le résultat est supérieur à 2, l'expression identifie les enregistrements pour le rejet. Si le résultat est inférieur ou égal à 2, elle marque les enregistrements pour la mise à jour. Sinon, elle les marque pour le rejet :

```
IIF( TRUNC( DATE_DIFF( SYSDATE, ORDERS.DATE_SHIPPED, 'DD' ), 0 ) > 2, DD_REJECT, DD_UPDATE )
```

## FALSE

Clarifie une expression conditionnelle. FALSE équivaut à l'entier 0.

## Exemple

L'exemple suivant utilise FALSE dans une expression DECODE pour renvoyer des valeurs selon les résultats d'une comparaison. Ceci est utile si vous voulez effectuer plusieurs recherches basées sur une seule valeur de recherche :

```
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.')
```

## NUL

Indique qu'une valeur est soit inconnue, soit non définie. NULL n'est pas équivalent à un blanc ou une chaîne vide (pour les colonnes de caractères) ou à 0 (pour les colonnes numériques).

Bien que vous puissiez écrire des expressions renvoyant NULL, une colonne qui a la contrainte NOT NULL ou PRIMARY KEY n'acceptera pas les valeurs nulles. Ainsi, si le Service d'intégration de données essaie d'écrire une valeur nulle dans une colonne ayant une de ces contraintes, la base de données rejettera la ligne et le Service d'intégration de données l'écrira dans le fichier de rejet. Veillez à prendre en compte les valeurs nulles lorsque vous créez les transformations.

Les fonctions peuvent gérer les valeurs nulles de différentes manières. Si vous transmettez une valeur nulle à une fonction, elle peut renvoyer 0 ou NULL, ou elle peut ignorer les valeurs nulles.

### LIENS CONNEXES :

- ["Fonctions" à la page 43](#)

## Utilisation des valeurs nulles dans des expressions booléennes

Les expressions qui combinent une valeur nulle et une expression booléenne produisent des résultats qui sont conformes à ANSI. Par exemple, le Service d'intégration de données produit les résultats suivants :

- NULL et TRUE = NULL
- NULL et FALSE = FALSE

## Utilisation des valeurs nulles dans des expressions de comparaison

Quand vous utilisez une valeur nulle dans une expression contenant un opérateur de comparaison, le Service d'intégration de données produit une valeur nulle.

## Valeurs nulles dans des fonctions Agrégation

Le Service d'intégration de données traite les valeurs nulles comme NULL dans les fonctions Agrégation. Si vous transmettez un port entier ou un groupe de valeurs nulles, la fonction renvoie NULL.

## Valeurs nulles dans des conditions de filtre

Si une condition de filtre vaut NULL, la fonction ne sélectionne pas l'enregistrement. Si la condition de filtre vaut NULL pour tous les enregistrements du port sélectionné, la fonction d'agrégation renvoie NULL (sauf pour COUNT qui renvoie 0). Vous pouvez utiliser les conditions de filtre avec les fonctions d'agrégation et les fonctions CUME, MOVINGAVG et MOVINGSUM.

## Valeurs nulles avec les opérateurs

Toute expression utilisant des opérateurs (excepté l'opérateur de chaîne ||) et contenant une valeur nulle vaut toujours NULL. Par exemple, l'expression suivante vaut NULL :

```
8 * 10 - NULL
```

Pour identifier les valeurs nulles, utilisez la fonction ISNULL.

## TRUE

Renvoie une valeur basée sur le résultat d'une comparaison. TRUE équivaut à l'entier 1.

## Exemple

L'exemple suivant utilise TRUE dans une expression DECODE pour renvoyer des valeurs selon les résultats d'une comparaison. Ceci est utile si vous voulez effectuer plusieurs recherches basées sur une seule valeur de recherche :

```
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.')
```

# CHAPITRE 3

## Opérateurs

Ce chapitre comprend les rubriques suivantes :

- [Priorité des opérateurs, 23](#)
- [Opérateurs arithmétiques, 24](#)
- [Opérateurs de chaîne, 25](#)
- [Opérateurs de comparaison, 26](#)
- [Opérateurs logiques, 26](#)

### Priorité des opérateurs

Le langage de transformation prend en charge l'utilisation de plusieurs opérateurs et l'utilisation des opérateurs dans des expressions imbriquées.

Si vous écrivez une expression qui comporte plusieurs opérateurs, le Service d'intégration de données évalue l'expression dans l'ordre suivant :

1. Opérateurs arithmétiques
2. Opérateurs de chaîne
3. Opérateurs de comparaison
4. Opérateurs logiques

Le Service d'intégration de données évalue les opérateurs selon leur ordre d'apparition dans le tableau suivant. Il évalue les opérateurs, dans une expression où tous les opérateurs ont une priorité identique, de la gauche vers la droite.

Le tableau suivant liste la priorité de tous les opérateurs du langage de transformation :

Opérateur	Signification
( )	Parenthèses.
+, -, NOT	plus unaire et moins unaire et l'opérateur logique NOT.
*, /, %	Multiplication, division, modulo.
+, -	Addition, soustraction.
	Concaténation.

Opérateur	Signification
<, <=, >, >=	Inférieur, inférieur ou égal, supérieur, supérieur ou égal.
=, <>, !=, ^=	Egal, différent, différent, différent.
AND	Opérateur logique AND, utilisé lors de la spécification de conditions.
OU	Opérateur logique OR, utilisé lors de la spécification de conditions.

Le langage de transformation prend également en charge l'utilisation d'opérateurs dans des expressions imbriquées. Quand les expressions comprennent des parenthèses, le Service d'intégration de données évalue les opérations comprises entre les parenthèses avant les opérations à l'extérieur des parenthèses. Les opérations situées dans les parenthèses les plus internes sont évaluées en premier.

Par exemple, en fonction de la façon dont vous imbriquez les opérations, l'équation  $8 + 5 - 2 * 8$  renvoie des valeurs différentes :

Equation	Valeur de retour
$8 + 5 - 2 * 8$	-3
$8 + (5 - 2) * 8$	32

## Opérateurs arithmétiques

Utilisez les opérateurs arithmétiques pour effectuer des calculs mathématiques sur des données numériques.

Le tableau suivant liste les opérateurs arithmétiques selon leur ordre de priorité dans le langage de transformation :

Opérateur	Signification
+, -	plus unaire et moins unaire. Le plus unaire indique une valeur positive. Le moins unaire indique une valeur négative.
*, /, %	Multipliation, division, modulo. Le modulo est le reste de la division de deux entiers. Par exemple, $13 \% 2 = 1$ , car 13 divisé par 2 est égal à 6, avec un reste de 1.
+, -	Addition, soustraction. L'opérateur d'addition (+) ne permet pas de concaténer des chaînes. Pour concaténer des chaînes, utilisez l'opérateur de chaîne   . Pour effectuer de l'arithmétique sur des valeurs date, utilisez les fonctions de date.

Si vous effectuez de l'arithmétique sur une valeur nulle, la fonction renvoie NULL.

Lorsque vous utilisez des opérateurs arithmétiques dans une expression, tous les opérandes de l'expression doivent être numériques. Par exemple, l'expression  $1 + '1'$  n'est pas valide, car elle ajoute un entier à une chaîne. L'expression  $1.23 + 4 / 2$  est valide, car tous les opérandes sont numériques.

**Remarque:** Le langage de transformation fournit des fonctions de date intégrées qui vous permettent d'effectuer de l'arithmétique sur des valeurs date/heure.

#### LIENS CONNEXES :

- ["Comprendre l'arithmétique de date" à la page 42](#)

## Opérateurs de chaîne

Utilisez l'opérateur de chaîne || pour concaténer deux chaînes. L'opérateur || convertit les opérandes de tout type de données (excepté binaires) en types de données Chaîne avant la concaténation :

Valeur d'entrée	Valeur de retour
'alpha'    'bétique'	alphabétique
'alpha'    2	alpha2
'alpha'    NULL	alpha

L'opérateur || inclut des blancs au début et à la fin. Utilisez les fonctions LTRIM et RTRIM pour couper les blancs au début et à la fin avant de concaténer deux chaînes.

### Valeurs nulles

L'opérateur || ignore les valeurs nulles. Cependant, si les deux valeurs sont NULL, l'opérateur || renvoie NULL.

### Exemple

L'exemple suivant montre une expression qui fait une concaténation des prénoms et des noms des employés de deux colonnes. Cette expression supprime les espaces à la fin du prénom et au début du nom de famille, concatène un espace à la fin de chaque prénom, puis concatène le nom de famille :

```
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

**Remarque:** Vous pouvez également utiliser la fonction CONCAT pour concaténer deux valeurs de chaîne. L'opérateur ||, cependant, produit les mêmes résultats en moins de temps.

# Opérateurs de comparaison

Utilisez les opérateurs de comparaison pour comparer des chaînes de caractères ou des chaînes numériques, manipuler des données et renvoyer une valeur TRUE (1) ou FALSE (0).

Le tableau suivant liste les opérateurs de comparaison du langage de transformation :

Opérateur	Signification
=	Egal à.
>	Supérieur à.
<	Inférieur à.
>=	Supérieur ou égal à.
<=	Inférieur ou égal à.
<>	Différent de.
!=	Différent de.
^=	Différent de.

Utilisez les opérateurs supérieur (>) et inférieur (<) pour comparer des valeurs numériques ou renvoyer une plage de lignes selon l'ordre de tri d'une clé primaire dans un port spécifique.

Lorsque vous utilisez les opérateurs de comparaison dans une expression, les opérandes doivent être du même type de données. Par exemple, l'expression `123.4 > '123'` n'est pas valide, car elle compare une valeur décimale à une chaîne. Les expressions `123.4 > 123` et `'a' != 'b'` sont valides, car les opérandes sont du même type de données.

Si vous comparez une valeur à une valeur nulle, le résultat est NULL.

Si une condition de filtre vaut NULL, le service d'intégration renvoie NULL.

# Opérateurs logiques

Utilisez les opérateurs logiques pour manipuler des données numériques. Les expressions qui renvoient une valeur numérique valent TRUE pour les valeurs autres que 0, FALSE pour 0 et NULL pour NULL.

Le tableau suivant liste les opérateurs logiques du langage de transformation :

Opérateur	Signification
NOT	Inverse logiquement le résultat d'une expression. Par exemple, si une expression vaut TRUE, l'opérateur NOT renvoie FALSE. Si une expression vaut FALSE, NOT renvoie TRUE.
AND	Joint deux conditions et renvoie TRUE si les deux conditions valent TRUE. Renvoie FALSE si une condition n'est pas vraie.
OU	Connecte deux conditions et renvoie TRUE si une condition vaut TRUE. Renvoie FALSE si les deux conditions ne sont pas vraies.

## Valeurs nulles

Les expressions qui combinent une valeur nulle avec une expression booléenne produisent des résultats qui sont conformes à ANSI. Par exemple, le Service d'intégration de données produit les résultats suivants :

- NULL et TRUE = NULL
- NULL et FALSE = FALSE

# CHAPITRE 4

## Variables

Ce chapitre comprend les rubriques suivantes :

- [Variables intégrées, 28](#)
- [Variables locales, 28](#)

### Variables intégrées

Le langage de transformation fournit la variable intégrée SYSDATE qui renvoie la date système. Vous pouvez utiliser SYSDATE dans une expression. Par exemple, vous pouvez utiliser SYSDATE dans une fonction DATE\_DIFF.

#### SYSDATE

SYSDATE renvoie la date et l'heure courante, jusqu'aux secondes, sur le nœud qui traite les données pour chaque ligne transmise via la transformation. SYSDATE est stockée comme une valeur de type de données de la transformation date/heure.

#### Exemple

L'expression suivante utilise SYSDATE pour trouver les commandes qui ont été expédiées dans les deux derniers jours et les marque pour l'insertion. En utilisant DATE\_DIFF, le Service d'intégration de données soustrait DATE\_SHIPPED de la date système, renvoyant la différence entre les deux dates. DATE\_DIFF renvoyant une valeur Double, l'expression tronque la différence. Elle compare ensuite le résultat au littéral entier 2. Si le résultat est supérieur à 2, l'expression marque les lignes pour le rejet. Si le résultat est inférieur ou égal à 2, elle les marque pour l'insertion.

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

### Variables locales

Si vous utilisez des variables locales dans un mappage, utilisez-les dans n'importe quelle expression de transformation du mappage. Par exemple, si vous utilisez un calcul complexe de la taxe tout au long d'un mappage, vous souhaiteriez peut-être écrire l'expression une fois et la désigner comme une variable. Ceci accroît les performances, car le Service d'intégration de données effectue le calcul une seule fois.

Les variables locales sont utiles quand elles sont utilisées avec des expressions de procédure stockée pour capturer plusieurs valeurs de retour.

# CHAPITRE 5

## Dates

Ce chapitre comprend les rubriques suivantes :

- [Présentation des dates, 30](#)
- [Chaînes de format de date, 34](#)
- [Chaînes de format TO\\_CHAR, 35](#)
- [Les chaînes de format TO\\_DATE et IS\\_DATE, 38](#)
- [Comprendre l'arithmétique de date, 42](#)

## Présentation des dates

Le langage de transformation fournit un ensemble de fonctions de date et de variables de date intégrées pour effectuer des transformations sur les dates. Avec les fonctions de date, vous pouvez arrondir, tronquer ou comparer des dates, extraire une partie d'une date ou effectuer de l'arithmétique sur une date. Vous pouvez transmettre n'importe quelle valeur ayant un type de données date à une fonction date.

Utilisez des variables de date pour capturer la date actuelle sur le nœud hébergeant le Service d'intégration de données.

Le langage de transformation fournit également les ensembles de chaînes de format suivants :

- **Chaînes de format de date.** Utilisez avec des fonctions de date pour spécifier les parties d'une date.
- **Chaînes de format TO\_CHAR.** Utilisez pour spécifier le format de la chaîne de retour.
- **Chaînes de format TO\_DATE et IS\_DATE.** Utilisez pour spécifier le format d'une chaîne que vous voulez convertir en date ou tester.

## Type de données Date/Heure

Informatica utilise des types de données génériques pour transformer les données provenant de différentes sources. Ces types de données de transformation comprennent un type de données Date/Heure qui prend en charge les valeurs date/heure allant jusqu'à la nanoseconde. Informatica stocke les dates en interne au format binaire.

Les fonctions de date acceptent uniquement des valeurs date/heure. Pour transmettre une chaîne à une fonction date, utilisez d'abord TO\_DATE pour la convertir en une valeur date/heure. Par exemple, l'expression suivante convertit un port de chaîne en valeurs date/heure puis ajoute un mois à chaque date :

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

Vous pouvez utiliser des dates comprises entre l'an 1 et 9999 apr. J.-C. dans le système calendaire grégorien.

## Date ordinale, date ordinale modifiée et calendrier grégorien

Vous pouvez utiliser des dates uniquement dans le système calendaire grégorien. Les dates dans le calendrier Ordinal sont appelées *dates* ordinales et ne sont pas prises en charge par Informatica. Ce terme ne doit pas être confondu avec le *jour* ordinal ou avec le jour ordinal modifié.

Vous pouvez manipuler les formats de jour ordinal modifié (JOM) avec la chaîne de format J. Le MJD pour une date donnée est le nombre de jours jusqu'à cette date depuis le 1er janvier de l'an 4713 av. J.-C. à 00:00:00 (minuit). Par définition, MJD inclut un élément heure exprimé en décimal, qui représente une fraction de 24 heures. La chaîne de format J ne convertit pas cet élément heure.

Par exemple, l'expression suivante TO\_DATE convertit les chaînes dans le port SHIP\_DATE\_MJD\_STRING en valeurs de date au format de date par défaut :

```
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

La chaîne de format J ne contenant pas la portion d'heure d'une date, l'heure des valeurs de retour est définie à 00:00:00.000000000.

Vous pouvez également utiliser la chaîne de format J dans des expressions TO\_CHAR. Par exemple, utilisez la chaîne de format J dans une expression TO\_CHAR pour convertir des valeurs de date en valeurs MJD exprimées en chaînes. Par exemple :

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

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

**Remarque:** Le Service d'intégration de données ignore la portion heure d'une date dans une expression TO\_CHAR.

## Dates en l'an 2000

Toutes les fonctions de date du langage de transformation prennent en charge l'année 2000. Informatica Developer prend en charge des dates comprises entre l'an 1 et 9999 apr. J.-C..

## Chaîne du format RR

Le langage de transformation fournit la chaîne de format RR pour convertir des chaînes avec des années à deux chiffres en dates. Avec TO\_DATE et la chaîne de format RR, vous pouvez convertir une chaîne de format MM/DD/RR en date. La chaîne de format RR convertit les données différemment selon l'année en cours.

- **Année en cours entre 0 et 49.** Si l'année en cours est comprise entre 0 et 49 (comme 2003) et que l'année de la chaîne source est comprise entre 0 et 49, le Service d'intégration de données renvoie le siècle actuel et les deux chiffres de l'année de la chaîne source. Si l'année de la chaîne source est comprise entre 50 et 99, le service d'intégration renvoie le siècle précédent et les deux chiffres de l'année de la chaîne source.
- **Année en cours entre 50 et 99.** Si l'année en cours est comprise entre 50 et 99 (comme 1998) et que l'année de la chaîne source est comprise entre 0 et 49, le Service d'intégration de données renvoie le siècle suivant et les deux chiffres de l'année de la chaîne source. Si l'année de la chaîne source est comprise entre 50 et 99, le Service d'intégration de données renvoie le siècle en cours et les deux chiffres de l'année spécifiée.

Le tableau suivant résume la méthode de conversion en dates de la chaîne de format RR :

Année en cours	Année source	La chaîne de format RR renvoie
0-49	0-49	Siècle en cours
0-49	50-99	Siècle précédent
50-99	0-49	Siècle suivant
50-99	50-99	Siècle en cours

## Exemple

L'expression suivante produit les mêmes valeurs de retour pour toute année en cours comprise entre 1950 et 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

## Différence entre les chaînes de format RR et YY

Informatica Developer fournit également une chaîne de format YY. Les chaînes de format RR et YY spécifient les années sur deux chiffres. Les chaînes de format YY et RR produisent des résultats identiques quand elles sont utilisées avec toutes les fonctions de date, sauf TO\_DATE. Dans les expressions TO\_DATE, RR et YY produisent des résultats différents.

Le tableau suivant affiche les différents résultats renvoyés par chaque chaîne de format :

Chaîne	Année en cours	TO_DATE(Chaîne, 'MM/DD/RR')	TO_DATE(Chaîne, 'MM/DD/YY')
04/12/98	1998	04/12/1998 00:00:00.000000000	04/12/1998 00:00:00.000000000
11/09/01	1998	11/09/2001 00:00:00.000000000	11/09/1901 00:00:00.000000000
04/12/98	2003	04/12/1998 00:00:00.000000000	04/12/2098 00:00:00.000000000
11/09/01	2003	11/09/2001 00:00:00.000000000	11/09/2001 00:00:00.000000000

Pour les dates de l'an 2000 et ultérieures, la chaîne de format YY produit moins de résultats significatifs que la chaîne de format RR. Utilisez la chaîne de format RR pour les dates du vingt et unième siècle.

## Dates dans des bases de données relationnelles

En général, les dates stockées dans les bases de données relationnelles contiennent une valeur de date et heure. La date comprend le mois, le jour et l'année ; l'heure, quant à elle, peut inclure les heures, les minutes, les secondes et les sous-secondes. Vous pouvez transmettre des données date/heure à n'importe quelle fonction de date.

## Dates dans des fichiers plats

Utilisez la fonction TO\_DATE pour convertir les chaînes en valeurs date/heure. Vous pouvez également utiliser IS\_DATE pour vérifier si une chaîne est une date valide avant de la convertir avec TO\_DATE. Les fonctions de date du langage de transformation acceptent uniquement des valeurs date. Pour transmettre une chaîne à une fonction date, vous devez d'abord utiliser la fonction TO\_DATE pour la convertir en un type de données de transformation Date/Heure.

## Format de date par défaut

Le Service d'intégration de données utilise un format de date par défaut pour stocker et manipuler des chaînes représentant des dates. Pour spécifier le format de date par défaut, entrez un format de date dans l'attribut de chaîne de format date/heure dans la configuration de la visionneuse de données. Le format de date par défaut est MM/DD/YYYY HH24:MI:SS.US.

Informatica stocke les données au format binaire ; c'est pourquoi le Service d'intégration de données utilise le format de date par défaut quand vous effectuez les actions suivantes :

- **Convertissez une date en chaîne en connectant un port date/heure à un port chaîne.** Le Service d'intégration de données convertit la date en une chaîne au format de date défini dans la configuration de la visionneuse de données.
- **Convertissez une chaîne en date en connectant un port chaîne à un port date/heure.** Le Service d'intégration de données demande à ce que les valeurs de chaîne soient au format de date défini dans la configuration de la visionneuse de données. Si une valeur d'entrée ne correspond pas à ce format ou s'il s'agit d'une date non valide, le Service d'intégration de données ignore la ligne. Si la chaîne est dans ce format, le Service d'intégration de données convertit la chaîne en valeur date.
- **Utilisez TO\_CHAR(date, [format\_string]) pour convertir des dates en chaînes.** Si vous omettez la chaîne de format, le Service d'intégration de données renvoie la chaîne dans le format de date défini dans la configuration de la visionneuse de données. Si vous spécifiez une chaîne de format, le Service d'intégration de données renvoie une chaîne dans le format spécifié.

- Utilisez **TO\_DATE(date, [format\_string])** pour convertir des chaînes en dates. Si vous omettez la chaîne de format, le Service d'intégration de données attend la chaîne au format de date défini dans la configuration de la visionneuse de données. Si vous spécifiez une chaîne de format, le Service d'intégration de données attend une chaîne dans le format spécifié.

Le format de date par défaut MM/DD/YYYY HH24:MI:SS.US est composé de :

- Mois (janvier = 01, septembre = 09)
- Jour (du mois)
- Année (exprimée en quatre chiffres, par exemple 1998)
- Heure (au format 24 heures, par exemple, 12:00:00AM = 0, 1:00:00AM = 1, 12:00:00PM = 12, 11:00:00PM = 23)
- Minutes
- Secondes
- Microsecondes

## Chaînes de format de date

Vous pouvez évaluer les dates d'entrée en utilisant une combinaison de chaînes de format et de fonctions de date. Les chaînes de format de date ne sont pas internationalisées et doivent être entrées dans les formats prédéfinis listés dans le tableau suivant :

Le tableau suivant récapitule les chaînes de format permettant de spécifier une partie d'une date :

Chaîne de format	Description
D, DD, DDD, DAY, DY, J	Jours (01-31). Utilisez n'importe laquelle de ces chaînes de format pour spécifier la portion du jour, en entier, d'une date. Par exemple, si vous transmettez 12-AVR-1997 à une fonction date, l'utilisation de n'importe laquelle de ces chaînes de format spécifie 12.
HH, HH12, HH24	Heure du jour (0-23), où 0 correspond à 0 h (minuit). Utilisez n'importe lequel de ces formats pour spécifier la portion de l'heure, en entier, d'une date. Par exemple, si vous transmettez la date 12-AVR-1997 2:01:32 PM, utilisez HH, HH12 ou HH24 pour spécifier la portion heure d'une date.
MI	Minutes (0 à 59).
MM, MON, MONTH	Mois (01-12). Utilisez n'importe laquelle de ces chaînes de format pour spécifier la portion du mois, en entier, d'une date. Par exemple, si vous transmettez 12-AVR-1997 à une fonction date, utilisez MM, MON ou MONTH pour spécifier AVR.
MS	Millisecondes (0 à 999).
NS	Nanosecondes (0 à 999999999).
SS, SSSS	Secondes (0 à 59).

Chaîne de format	Description
US	Microsecondes (0 à 999999).
Y, YY, YYY, YYYY, RR	Portion année d'une date (0001 à 9999). Utilisez n'importe laquelle de ces chaînes de format pour spécifier la portion de l'année, en entier, d'une date. Par exemple, si vous transmettez 12-AVR-1997 à une fonction date, utilisez Y, YY, YYY ou YYYY pour spécifier 1997.

**Remarque:** La chaîne de format n'est pas sensible à la casse. Elle doit toujours être entourée de guillemets simples.

Le tableau suivant décrit les fonctions de date qui utilisent des chaînes de format de date pour évaluer les dates en entrée :

Fonction	Description
ADD_TO_DATE	La partie de la date que vous voulez modifier.
DATE_DIFF	La partie de la date à utiliser pour calculer la différence entre deux dates.
GET_DATE_PART	La partie de la date que vous voulez renvoyer. Cette fonction renvoie une valeur de type entier basée sur le format de date par défaut.
IS_DATE	La date que vous voulez vérifier.
ROUND	La partie de la date que vous voulez arrondir.
SET_DATE_PART	La partie de la date que vous voulez modifier.
SYSTIMESTAMP	La précision de l'horodatage.
TO_CHAR (Dates)	La chaîne de caractères.
TO_DATE	La chaîne de caractères.
TRUNC (Dates)	La partie de la date que vous voulez tronquer.

## Chaînes de format TO\_CHAR

La fonction TO\_CHAR convertit un type de données Date/Heure en une chaîne ayant le format que vous spécifiez. Vous pouvez convertir la totalité de la date ou une partie de la date en chaîne. Vous pouvez utiliser TO\_CHAR pour convertir des dates en chaînes, en modifiant le format à des fins d'établissement de rapports.

TO\_CHAR est généralement utilisé quand la cible est un fichier plat ou une base de données ne prenant pas en charge un type de données Date/Heure.

Le tableau suivant récapitule les chaînes de format pour les dates dans la fonction TO\_CHAR :

Chaîne de format	Description
AM, A.M., PM, P.M.	Indicateur méridien. Utilisez ces chaînes de format pour indiquer les heures AM (matin) et PM (après-midi). AM et PM renvoient les mêmes valeurs que A.M. et P.M.
D	Jour de la semaine (1-7), où dimanche est égal à 1.
DAY	Nom du jour, comprenant jusqu'à neuf caractères (par exemple, dimanche).
DD	Jour du mois (01-31).
DDD	Jour de l'année (001 à 366, incluant les années bissextiles).
DY	Nom du jour abrégé (trois caractères) (par exemple, mer).
HH, HH12	Heure du jour (01-12).
HH24	Heure du jour (00 à 23), où 00 correspond à minuit (12 AM).
J	Date ordinale modifiée. Convertit la date calendaire en une chaîne équivalente à sa valeur Date ordinale modifiée, calculée à partir du 1er janvier 4713 av. J.-C. à 00:00:00. Elle ignore l'élément heure de la date. Par exemple, l'expression TO_CHAR( SHIP_DATE, 'J' ) convertit Dec 31 1999 23:59:59 en chaîne 2451544.
MI	Minutes (00-59).
MM	Mois (01-12).
MONTH	Nom de mois, comprenant jusqu'à neuf caractères (par exemple, janvier).
MON	Nom de mois abrégé à trois caractères (par exemple, Jan).
MS	Millisecondes (0 à 999).
NS	Nanosecondes (0 à 999999999).
Q	Trimestre de l'année (1-4), où janvier à mars est égal à 1.
RR	Deux derniers chiffres d'une année. La fonction supprime les premiers chiffres. Par exemple, si vous utilisez « RR » et transmettez l'année 1997, TO_CHAR renvoie 97. Utilisé avec TO_CHAR, « RR » produit les mêmes résultats que « YY » et est interchangeable avec lui. Cependant, utilisé avec TO_DATE, « RR » calcule le siècle approprié le plus proche et fournit les deux premiers chiffres de l'année.
SS	Secondes (00-59).
SSSSS	Secondes depuis minuit (00000 - 86399). Quand vous utilisez SSSSS dans une expression TO_CHAR, le Service d'intégration de données évalue uniquement la portion heure d'une date. Par exemple, l'expression TO_CHAR(SHIP_DATE, 'MM/DD/YYYY SSSSS') convertit 12/31/1999 01:02:03 en 12/31/1999 03723.
US	Microsecondes (0 à 999999).
Y	Dernier chiffre d'une année. La fonction supprime les premiers chiffres. Par exemple, si vous utilisez « Y » et transmettez l'année 1997, TO_CHAR renvoie 7.

Chaîne de format	Description
YY	Deux derniers chiffres d'une année. La fonction supprime les premiers chiffres. Par exemple, si vous utilisez « YY » et transmettez l'année 1997, TO_CHAR renvoie 97.
YYY	Trois derniers chiffres d'une année. La fonction supprime les premiers chiffres. Par exemple, si vous utilisez « YYY » et que vous transmettez l'année 1997, TO_CHAR renvoie 997.
YYYY	Portion entière de l'année d'une date. Par exemple, si vous utilisez « YYYY » et que vous transmettez l'année 1997, TO_CHAR renvoie 1997.
W	Semaine du mois (1-5), où la semaine 1 démarre le premier jour du mois et se termine au septième et où la semaine 2 démarre le huitième jour et se termine le quatorzième. Par exemple, Feb 1 indique la première semaine de février.
WW	Semaine de l'année (01-53) où la semaine 1 démarre le 1er janvier et se termine le 7 janvier, où la semaine 2 commence le 8 janvier et se termine le 14 janvier, et ainsi de suite.
- / . ; :	Ponctuation affichée dans la sortie. Vous pouvez utiliser ces symboles pour séparer les parties de date. Par exemple, vous créez l'expression suivante pour séparer les parties de date par un point : TO_CHAR( DATES, 'MM.DD.YYYY' ).
"texte"	Texte affiché dans la sortie. Par exemple, si vous créez un port de sortie avec l'expression : TO_CHAR( DATES, 'MM/DD/YYYY "Les ventes ont progressé"' ) et que vous transmettez la date Avr 1 1997, la fonction renvoie la chaîne « 04/01/1997 Les ventes ont progressé ». Vous pouvez entrer des caractères multioctets qui sont valides dans la page de code du référentiel.
""	Utilisez des guillemets pour séparer des chaînes de format ambiguës, par exemple D""DDD. Les guillemets vides ne s'affichent pas dans la sortie.

**Remarque:** La chaîne de format n'est pas sensible à la casse. Elle doit toujours être entourée de guillemets simples.

## Exemples

Les exemples suivants présentent les chaînes de format J, SSSSS, RR et YY. Consultez les fonctions individuelles pour plus d'exemples.

**Remarque:** Le Service d'intégration de données ignore la portion heure de la date dans une expression TO\_CHAR.

### Chaîne de format J

Utilisez la chaîne de format J dans une expression TO\_CHAR pour convertir des valeurs de date en valeurs MJD exprimées en chaînes. Par exemple :

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

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

## Chaîne de format SSSSS

Vous pouvez également utiliser la chaîne de format SSSSS dans une expression TO\_CHAR. Par exemple, l'expression suivante convertit les dates dans le port SHIP\_DATE en chaînes représentant le nombre total de secondes depuis minuit :

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

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

## Chaîne du format RR

L'expression suivante convertit les dates en chaînes au format 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

## Chaîne de format YY

Dans les expressions TO\_CHAR, la chaîne de format YY produit les mêmes résultats que la chaîne de format RR. L'expression suivante convertit les dates en chaînes au format 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

# Les chaînes de format TO\_DATE et IS\_DATE

La fonction TO\_DATE convertit une chaîne avec le format que vous spécifiez à une valeur date/heure. TO\_DATE est généralement utilisé pour convertir les chaînes de fichiers plats en valeurs date/heure. Les chaînes de format TO\_DATE ne sont pas internationalisées et doivent être entrées dans les formats prédéfinis.

**Remarque:** TO\_DATE et IS\_DATE utilisent le même ensemble de chaînes de format.

Lorsque vous créez une expression TO\_DATE, utilisez une chaîne de format pour chaque partie de la date dans la chaîne source. Le format de la chaîne source et la chaîne de format doivent correspondre, y compris tout séparateur de date. Si une partie ne correspond pas, le Service d'intégration de données ne convertit pas la chaîne et ignore la ligne. Si vous omettez la chaîne de format, la chaîne source doit être dans le format de date spécifié dans la configuration de la visionneuse de données.

IS\_DATE indique si une valeur est une date valide. Une date valide est toute chaîne représentant une date valide dans le format de date spécifié dans la configuration de la visionneuse de données. Si les chaînes que vous voulez tester ne sont pas dans ce format de date, utilisez les chaînes de format listées dans ["Les chaînes de format TO\\_DATE et IS\\_DATE" à la page 38](#) pour spécifier le format de date. Si une chaîne ne correspond pas à la chaîne de format spécifiée ou n'est pas une date valide, la fonction renvoie FALSE (0). Si la chaîne correspond à la chaîne de format et est une date valide, la fonction renvoie TRUE (1). Les chaînes de format IS\_DATE ne sont pas internationalisées et doivent être entrées dans les formats prédéfinis listés dans le tableau suivant :

Le tableau suivant récapitule les chaînes de format pour les fonctions TO\_DATE et IS\_DATE :

**Tableau 1. Les chaînes de format TO\_DATE et IS\_DATE**

Chaîne de format	Description
AM, a.m., PM, p.m.	Indicateur méridien. Utilisez ces chaînes de format pour indiquer les heures AM (matin) et PM (après-midi). AM et PM renvoient les mêmes valeurs que a.m. et p.m.
DAY	Nom du jour, comprenant jusqu'à neuf caractères (par exemple, dimanche). La chaîne du format DAY n'est pas sensible à la casse.
DD	Jour du mois (1 à 31).
DDD	Jour de l'année (001 à 366, incluant les années bissextiles).
DY	Nom du jour abrégé (trois caractères) (par exemple, mer). La chaîne du format DY n'est pas sensible à la casse.
HH, HH12	Heure du jour (1 à 12).
HH24	Heure du jour (0 à 23), où 0 correspond à minuit (12 AM).
J	Date ordinale modifiée. Convertissez les chaînes au format MJD en valeurs date. Il ignore l'élément heure de la chaîne source, en assignant à toutes les dates, l'heure 00:00:00.000000000. Par exemple, l'expression TO_DATE('2451544', 'J') convertit 2451544 en Dec 31 1999 00:00:00.000000000.
MI	Minutes (0 à 59).
MM	Mois (1-12).
MONTH	Nom de mois, incluant jusqu'à neuf caractères (par exemple, septembre). La casse n'est pas importante.
MON	Nom de mois abrégé (trois caractères) (par exemple, sep). La casse n'est pas importante.
MS	Millisecondes (0 à 999).
NS	Nanosecondes (0 à 999999999).

Chaîne de format	Description
RR	Année à quatre chiffres (par exemple, 1998, 2034). Utilisé lorsque les chaînes source incluent des années à deux chiffres. Utilisez avec TO_DATE pour convertir les années à deux chiffres en années à quatre chiffres. <ul style="list-style-type: none"> <li>- Année en cours entre 50 et 99. Si l'année en cours est comprise entre 50 et 99 (par exemple 1998) et que la valeur de l'année de la chaîne source est comprise entre 0 et 49, le Service d'intégration de données renvoie le siècle suivant et les deux chiffres de l'année de la chaîne source. Si la valeur de l'année de la chaîne source est comprise entre 50 et 99, le Service d'intégration de données renvoie le siècle en cours et les deux chiffres de l'année spécifiée.</li> <li>- Année en cours entre 0 et 49. Si l'année en cours est comprise entre 0 et 49 (par exemple 2003) et que l'année de la chaîne source est comprise entre 0 et 49, le Service d'intégration de données renvoie le siècle en cours et les deux chiffres de l'année de la chaîne source. Si l'année de la chaîne source est comprise entre 50 et 99, le Service d'intégration de données renvoie le siècle précédent et les deux chiffres de l'année de la chaîne source.</li> </ul>
SS	Secondes (0 à 59).
SSSSS	Secondes depuis minuit. Quand vous utilisez SSSSS dans une expression TO_DATE, le Service d'intégration de données évalue uniquement la portion heure d'une date. Par exemple, l'expression TO_DATE( DATE_STR, 'MM/DD/YYYY SSSSS') convertit 12/31/1999 3783 en 12/31/1999 01:02:03.
US	Microsecondes (0 à 999999).
Y	L'année en cours sur le nœud exécutant le Service d'intégration de données avec le dernier chiffre de l'année remplacé par la valeur de la chaîne.
YY	L'année en cours sur le nœud exécutant le Service d'intégration de données avec les deux derniers chiffres de l'année remplacés par la valeur de la chaîne.
YYY	L'année en cours sur le nœud exécutant le Service d'intégration de données avec les trois derniers chiffres de l'année remplacés par la valeur de la chaîne.
YYYY	Les quatre chiffres de l'année. N'utilisez pas cette chaîne de format si vous adoptez les années à deux chiffres. Utilisez la chaîne de format RR ou YY à la place.

## Règles et directives pour les chaînes de format de date

Utilisez les règles et directives suivantes quand vous travaillez avec des chaînes de format de date :

- Le format de la chaîne TO\_DATE doit correspondre à la chaîne de format, y compris les séparateurs de date. Sinon, le Service d'intégration de données peut renvoyer des valeurs incorrectes ou ignorer la ligne. Par exemple, si vous transmettez la chaîne « 20200512 », représentant 12 mai 2020, à TO\_DATE, vous devez inclure la chaîne de format YYYYMMDD. Si vous n'indiquez pas une chaîne de format, le Service d'intégration de données attend que la chaîne soit dans le format de date spécifié dans la session. De la même façon, si vous transmettez une chaîne qui ne correspond pas à la chaîne de format, le Service d'intégration de données renvoie une erreur et ignore la ligne. Par exemple, si vous transmettez la chaîne 2020120 à TO\_DATE et que vous incluez la chaîne de format YYYYMMDD, le Service d'intégration de données renvoie une erreur et ignore la ligne, car la chaîne ne correspond pas à la chaîne de format.
- La chaîne de format doit être entourée de guillemets simples.
- Le Service d'intégration de données utilise le format date/heure par défaut spécifié dans la session. La valeur par défaut est MM/DD/YYYY HH24:MI:SS.US. La chaîne de format n'est pas sensible à la casse.

## Exemple

Les exemples suivants illustrent les chaînes de format J, RR et SSSSS. Consultez les fonctions individuelles pour plus d'exemples.

### Chaîne de format J

L'expression suivante convertit les chaînes du port SHIP\_DATE\_MJD\_STRING en valeurs de date dans le format de date par défaut :

```
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

La chaîne de format J ne contenant pas la portion d'heure d'une date, l'heure des valeurs de retour est définie à 00:00:00.000000000.

### Chaîne du format RR

L'expression suivante convertit une chaîne en un format d'année à quatre chiffres. L'année en cours est 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

### Chaîne de format YY

L'expression suivante convertit une chaîne en un format d'année à quatre chiffres. L'année en cours est 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

**Remarque:** Pour la deuxième ligne, RR renvoie l'année 2005, mais YY renvoie l'année 1905.

## Chaîne de format SSSSS

L'expression suivante convertit les chaînes contenant les secondes à partir de minuit en valeurs de date :

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

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

## Comprendre l'arithmétique de date

Le langage de transformation fournit des fonctions de date intégrées de façon à pouvoir exécuter de l'arithmétique sur les valeurs date/heure, comme suit :

- **ADD\_TO\_DATE.** Ajoutez ou soustrayez une portion de date spécifique.
- **DATE\_DIFF.** Soustrayez deux dates.
- **SET\_DATE\_PART.** Modifiez une partie d'une date.

Vous ne pouvez pas utiliser d'opérateurs arithmétiques numériques (comme + ou -) pour ajouter ou soustraire des dates.

Le langage de transformation reconnaît les années bissextiles et accepte les dates entre Jan. 1, 0001 00:00:00.000000000 apr. J.-C. et Déc. 31, 9999 23:59:59.999999999 apr. J.-C..

**Remarque:** Le langage de transformation utilise le type de données de la transformation Date/Heure pour spécifier les valeurs de date. Vous ne pouvez utiliser les fonctions de date que sur des valeurs date/heure.

## CHAPITRE 6

# Fonctions

Ce chapitre décrit les fonctions du langage de transformation par ordre alphabétique. Chaque description de fonction inclut :

- Syntaxe
- Valeur de retour
- Exemple

## Catégories de fonctions

Le langage de transformation fournit les types de fonctions suivants :

- Agréger
- Caractère
- Conversion
- Nettoyage des données
- Date
- Codage
- Financier
- Numérique
- Scientifique
- Spécial
- Chaîne
- Test
- Variable

## Fonctions d'agrégation

Les fonctions Agrégation renvoient des valeurs de résumé pour les valeurs non nulles dans les ports sélectionnés. Les fonctions Agrégation permettent de :

- Calculer une valeur unique pour toutes les lignes d'un groupe.
- Renvoyer une valeur unique pour chaque groupe dans une transformation Agrégation.
- Appliquer des filtres pour calculer les valeurs de lignes spécifiques dans les ports sélectionnés.

- Utiliser les opérateurs pour effectuer des opérations arithmétiques dans la fonction.
- Calculer en un seul passage deux valeurs d'agrégat ou plus, dérivées des mêmes colonnes source.

Le langage de transformation comprend les fonctions Agrégation suivantes :

- ANY
- AVG
- COUNT
- FIRST
- LAST
- MAX (Date)
- MAX (Nombre)
- MAX (Chaîne)
- MEDIAN
- MIN (Date)
- MIN (Nombre)
- MIN (Chaîne)
- PERCENTILE
- STDDEV
- SUM
- VARIANCE

Si vous configurez l'exécution du Service d'intégration de données en mode Unicode, MIN et MAX renvoient des valeurs en fonction de l'ordre de tri de la page de code que vous spécifiez dans la configuration du mappage.

Utilisez les fonctions Agrégation dans les transformations Agrégation uniquement. Vous ne pouvez imbriquer qu'une seule fonction Agrégation dans une autre fonction Agrégation. Le Service d'intégration de données évalue l'expression de la fonction Agrégation la plus imbriquée et utilise le résultat pour évaluer l'expression de la fonction Agrégation externe. Vous pouvez configurer une transformation Agrégation qui effectue des regroupements par ID et imbrique deux fonctions Agrégation comme suit :

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

Lorsque l'ensemble de données contient les valeurs suivantes :

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

ID	EARNINGS
3	45
3	99

La valeur de retour est 186. Le Service d'intégration de données groupe par ID, évalue l'expression AVG et renvoie trois valeurs. Puis, il ajoute les valeurs avec la fonction SUM pour obtenir le résultat.

## Fonctions Agrégation et valeurs nulles

Lorsque vous configurez le Service d'intégration de données, vous pouvez choisir la méthode de traitement des valeurs nulles dans les fonctions Agrégation. Vous pouvez configurer le Service d'intégration de données de manière à ce qu'il traite les valeurs nulles dans les fonctions Agrégation comme NULL ou 0.

Par défaut, le Service d'intégration de données traite les valeurs nulles comme NULL dans les fonctions Agrégation. Si vous transmettez un port entier ou un groupe de valeurs nulles, la fonction renvoie NULL. Vous pouvez éventuellement configurer le Service d'intégration de données pour qu'une fonction Agrégation renvoie 0 si vous lui transmettez un port complet de valeurs nulles.

### Conditions de filtre

Utilisez une condition de filtre pour limiter les lignes renvoyées dans une recherche.

Un filtre permet de limiter les lignes renvoyées dans une recherche. Vous pouvez appliquer une condition de filtre à toutes les fonctions Agrégation et aux fonctions CUME, MOVINGAVG et MOVINGSUM. La condition de filtre doit renvoyer TRUE, FALSE ou NULL. Si la condition de filtre renvoie NULL ou FALSE, le Service d'intégration de données ne sélectionne pas la ligne.

Vous pouvez entrer l'expression de transformation valide de votre choix. Par exemple, l'expression suivante calcule le salaire médian pour tous les employés qui gagnent plus de 50 000 \$ :

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

Vous pouvez également utiliser d'autres valeurs numériques comme condition de filtre. Par exemple, vous pouvez entrer les données suivantes comme syntaxe complète pour la fonction MEDIAN, incluant un port numérique :

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

Dans tous les cas, le Service d'intégration de données arrondit une valeur décimale à un nombre entier (par exemple : 1,5 à 2, 1,2 à 1, 0,35 à 0) pour la condition de filtre. Si la valeur est arrondie à 0, la condition de filtre renvoie FALSE. Si vous ne voulez pas arrondir une valeur, utilisez la fonction TRUNC pour tronquer la valeur en un nombre entier :

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

Si vous omettez la condition de filtre, la fonction sélectionne toutes les lignes dans le port.

## Fonctions de caractères

Le langage de transformation comprend les fonctions de caractères suivantes :

- ASCII
- CHR
- CHRCODE

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

Les fonctions de caractères MAX, MIN, LOWER, UPPER et INITCAP utilisent la page de code du Service d'intégration de données pour évaluer les données de caractères.

## Fonctions de conversion

Le langage de transformation comprend les fonctions de conversion suivantes :

- TO\_BIGINT
- TO\_CHAR(Number)
- TO\_DATE
- TO\_DECIMAL
- TO\_FLOAT
- TO\_INTEGER

## Fonctions de nettoyage des données

Le langage de transformation inclut un groupe de fonctions pour éliminer les erreurs de données. Les fonctions de nettoyage des données permettent d'effectuer les tâches suivantes :

- Tester des valeurs d'entrée.
- Convertir le type de données d'une valeur d'entrée.
- Découper des valeurs de chaîne.
- Remplacer des caractères dans une chaîne.
- Coder des chaînes.
- Faire correspondre des modèles dans des expressions régulières.

Le langage de transformation comprend les fonctions de nettoyage des données suivantes :

- GREATEST
- IN

- INSTR
- IS\_DATE
- IS\_NUMBER
- IS\_SPACES
- ISNULL
- LEAST
- LTRIM
- METAPHONE
- REG\_EXTRACT
- REG\_MATCH
- REG\_REPLACE
- REPLACECHR
- REPLACESTR
- RTRIM
- SOUNDEX
- SUBSTR
- TO\_BIGINT
- TO\_CHAR
- TO\_DATE
- TO\_DECIMAL
- TO\_FLOAT
- TO\_INTEGER

## Fonctions de date

Le langage de transformation inclut un groupe de fonctions de date pour arrondir, tronquer, ou comparer des dates, extraire une partie d'une date, ou effectuer une opération arithmétique sur une date.

Vous pouvez transmettre une valeur ayant un type de données de date aux fonctions de date de votre choix. Cependant, si vous voulez transmettre une chaîne à une fonction de date, vous devez d'abord utiliser la fonction TO\_DATE pour la convertir en type de données Transformation Date/Heure.

Le langage de transformation comprend les fonctions de date suivantes :

- ADD\_TO\_DATE
- DATE\_COMPARE
- DATE\_DIFF
- GET\_DATE\_PART
- IS\_DATE
- LAST\_DAY
- MAKE\_DATE\_TIME
- MAX
- MIN
- ROUND(Date)

- SET\_DATE\_PART
- SYSTIMESTAMP
- TO\_CHAR(Date)
- TRUNC(Date)

Certaines fonctions de date incluent un argument de *format*. Vous devez spécifier une des chaînes de format du langage de transformation pour cet argument. Les chaînes de format de date ne sont pas internationalisées.

Le type de données Transformation Date/Heure prend en charge les dates avec une précision à la nanoseconde.

#### LIENS CONNEXES :

- ["Chaînes de format de date" à la page 34](#)

## Fonctions de codage

Le langage de transformation inclut les fonctions suivantes pour le cryptage de données, la compression, le codage et la somme de contrôle :

- AES\_DECRYPT
- AES\_ENCRYPT
- COMPRESS
- CRC32
- DEC\_BASE64
- DECOMPRESS
- ENC\_BASE64
- MD5

## Fonctions financières

Le langage de transformation comprend les fonctions financières suivantes :

- FV
- NPER
- PMT
- PV
- RATE

## Fonctions numériques

Le langage de transformation comprend les fonctions numériques suivantes :

- ABS
- CEIL
- CONV
- CUME
- EXP

- FLOOR
- LN
- JOURNAL
- MAX
- MIN
- MOD
- MOVINGAVG
- MOVINGSUM
- POWER
- RAND
- ROUND
- SIGN
- SQRT
- TRUNC

## Fonctions scientifiques

Le langage de transformation comprend les fonctions scientifiques suivantes :

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

## Fonctions spéciales

Le langage de transformation comprend les fonctions spéciales suivantes :

- ABANDONNER
- DECODE
- ERREUR
- IIF
- RECHERCHE
- UUID4
- UUID\_UNPARSE

Généralement, vous utilisez les fonctions spéciales dans des transformations Expression, Filtre et Stratégie de mise à jour. Vous pouvez imbriquer d'autres fonctions dans des fonctions spéciales. Vous pouvez également imbriquer une fonction spéciale dans une fonction Agrégation.

## Fonctions de chaîne

Le langage de transformation comprend les fonctions de chaîne suivantes :

- CHOOSE
- INDEXOF
- MAX
- MIN
- REVERSE

## Fonctions de test

Le langage de transformation comprend les fonctions de test suivantes :

- ISNULL
- IS\_DATE
- IS\_NUMBER
- IS\_SPACES

# ABANDONNER

Arrête l'exécution du mappage et émet un message d'erreur spécifique dans le journal. Lorsque le Service d'intégration de données détecte une fonction ABORT, il arrête la transformation des données à cette ligne. Il traite les lignes lues avant l'abandon de l'exécution du mappage. Le Service d'intégration de données écrit dans la cible jusqu'à la ligne abandonnée puis restaure toutes les données non validées au dernier point de validation.

Utilisez ABORT pour valider des données. Généralement, vous utilisez ABORT dans une fonction IIF ou DECODE pour définir les règles d'abandon d'une session.

Utilisez la fonction ABORT pour les valeurs par défaut des ports d'entrée et de sortie. Vous pouvez utiliser ABORT pour les ports d'entrée, pour empêcher la transmission des valeurs nulles à une transformation. Vous pouvez également utiliser ABORT pour gérer tout type d'erreur de transformation, y compris les appels de la fonction ERROR dans une expression. La valeur par défaut écrase la fonction ERROR dans une expression. Si vous voulez garantir l'arrêt de la session lorsqu'une erreur se produit, affectez ABORT comme valeur par défaut.

Si vous utilisez ABORT dans une expression pour un port non connecté, le Service d'intégration de données n'exécute pas la fonction ABORT.

### Syntaxe

```
ABORT( string )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire/ Facultatif	Description
<i>chaîne</i>	Obligatoire	Chaîne. Message que vous souhaitez afficher dans le journal lors de l'arrêt de l'exécution du mappage. La chaîne peut être de n'importe quelle longueur. Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

NULL.

## ABS

Renvoie la valeur absolue d'une valeur numérique.

### Syntaxe

```
ABS( numeric_value )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire/ Facultatif	Description
<i>numeric_value</i>	Obligatoire	Type de données numérique. Transmet les valeurs pour lesquelles vous voulez renvoyer les valeurs absolues. Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

Valeur numérique positive. ABS renvoie le même type de données que la valeur numérique transmise comme argument. Si vous transmettez une valeur de type Double, la fonction renvoie une valeur de type Double. De la même manière, si vous transmettez un nombre entier, elle renvoie un nombre entier.

NULL si vous transmettez une valeur nulle à la fonction.

**Remarque:** Si la valeur de retour est décimale avec une précision supérieure à 15, vous pouvez activer la précision élevée pour garantir une précision décimale jusqu'à 28 chiffres.

### Exemple

L'expression suivante renvoie la différence entre deux nombres comme une valeur positive, quel que soit le nombre le plus grand :

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

PRICE	COST	RETURN VALUE
250	150	100
52	48	4
169.95	69.95	100

PRICE	COST	RETURN VALUE
59.95	NULL	NULL
70	30	40
430	330	100
100	200	100

## ADD\_TO\_DATE

Ajoute un nombre spécifié à une partie d'une valeur date/heure et renvoie une date au même format que celle que vous transmettez à la fonction. ADD\_TO\_DATE accepte des valeurs entières positives et négatives. Utilisez ADD\_TO\_DATE pour modifier les parties suivantes d'une date :

- Année.**Entrez un entier positif ou négatif dans l'argument *quantité*. Utilisez les chaînes de format de l'année : Y, YY, YYYY, ou YYYY. L'expression suivante ajoute 10 ans à toutes les dates dans le port SHIP\_DATE :
 

```
ADD_TO_DATE ( SHIP_DATE, 'YY', 10 )
```
- Mois.**Entrez un entier positif ou négatif dans l'argument *quantité*. Utilisez les chaînes de format de mois suivantes : MM, MON, MONTH. L'expression suivante soustrait 10 mois à chaque date dans le port SHIP\_DATE :
 

```
ADD_TO_DATE( SHIP_DATE, 'MONTH', -10 )
```
- Jour.**Entrez un entier positif ou négatif dans l'argument *quantité*. Utilisez une des chaînes de format de jour : D, DD, DDD, DY et DAY. L'expression suivante ajoute 10 jours à chaque date dans le port SHIP\_DATE :
 

```
ADD_TO_DATE( SHIP_DATE, 'DD', 10 )
```
- Heure.**Entrez un entier positif ou négatif dans l'argument *quantité*. Utilisez une des chaînes de format d'heure : HH, HH12, HH24. L'expression suivante ajoute 14 heures à chaque date dans le port SHIP\_DATE :
 

```
ADD_TO_DATE( SHIP_DATE, 'HH', 14 )
```
- Minute.**Entrez un entier positif ou négatif dans l'argument *quantité*. Utilisez la chaîne de format MI pour définir les minutes. L'expression suivante ajoute 25 minutes à chaque date dans le port SHIP\_DATE :
 

```
ADD_TO_DATE( SHIP_DATE, 'MI', 25 )
```
- Secondes.**Entrez un entier positif ou négatif dans l'argument *quantité*. Utilisez la chaîne de format SS pour définir les secondes. L'expression suivante ajoute 59 secondes à chaque date dans le port SHIP\_DATE :
 

```
ADD_TO_DATE( SHIP_DATE, 'SS', 59 )
```
- Millisecondes.**Entrez un entier positif ou négatif dans l'argument *quantité*. Utilisez la chaîne de format MS pour définir les millisecondes. L'expression suivante ajoute 125 millisecondes à chaque date dans le port SHIP\_DATE :
 

```
ADD_TO_DATE( SHIP_DATE, 'MS', 125 )
```
- Microsecondes.**Entrez un entier positif ou négatif dans l'argument *quantité*. Utilisez la chaîne de format US pour définir les microsecondes. L'expression suivante ajoute 2000 microsecondes à chaque date dans le port SHIP\_DATE :
 

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

- **Nanosecondes.** Entrez un entier positif ou négatif dans l'argument *quantité*. Utilisez la chaîne de format NS pour définir les nanosecondes. L'expression suivante ajoute 3 000 000 nanosecondes à chaque date dans le port SHIP\_DATE :

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

## Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>date</i>	Obligatoire	Type de données Date/Heure. Transmet les valeurs que vous voulez modifier. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>format</i>	Obligatoire	Chaîne de format spécifiant la partie de la valeur de date que vous voulez modifier. Placez la chaîne de format entre guillemets simples, par exemple : 'mm'. La chaîne de format n'est pas sensible à la casse.
<i>quantité</i>	Obligatoire	Valeur entière spécifiant le nombre d'années, de mois, de jours, d'heures, etc. par laquelle vous voulez remplacer la valeur de date. Vous pouvez saisir toute expression de transformation valide qui renvoie un nombre entier.

## Valeur de retour

Date au même format que celle que vous avez transmise à cette fonction.

NULL si une valeur nulle est transmise comme argument à la fonction.

## Exemples

Toutes les expressions suivantes ajoutent un mois à chaque date dans le port DATE\_SHIPPED. Si vous transmettez une valeur qui crée un jour qui n'existe pas dans un mois donné, le Service d'intégration de données renvoie le dernier jour du mois. Par exemple, si vous ajoutez un mois à Jan 31 1998, le Service d'intégration de données renvoie Feb 28 1998.

Notez également que ADD\_TO\_DATE reconnaît les années bissextiles et ajoute un mois à Jan 29 2000 :

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

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

Les expressions suivantes soustraient 10 jours à chaque date dans le port 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

Les expressions suivantes soustraient 15 heures à chaque date dans le port 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

## Utilisation des dates

Suivez les conseils suivants lorsque vous travaillez avec ADD\_TO\_DATE :

- Vous pouvez ajouter ou soustraire une partie de la date en définissant une chaîne de format et en modifiant l'argument *quantité* par un nombre entier positif ou négatif.
- Si vous transmettez une valeur qui crée un jour qui n'existe pas dans un mois donné, le Service d'intégration de données renvoie le dernier jour du mois. Par exemple, si vous ajoutez un mois à Jan 31 1998, le Service d'intégration de données renvoie Feb 28 1998.
- Vous pouvez imbriquer TRUNC et ROUND pour manipuler les dates.
- Vous pouvez imbriquer TO\_DATE pour convertir des chaînes en dates.
- ADD\_TO\_DATE modifie uniquement la partie de la date que vous indiquez. Si vous modifiez une date pour passer de l'heure standard à l'heure d'été, vous devez changer la partie heure de la date.

# AES\_DECRYPT

Revoie des données décryptées vers le format de chaîne. Le Service d'intégration de données utilise l'algorithme standard de cryptage avancé (AES) avec un codage 128 bits. L'algorithme AES est un algorithme cryptographique certifié FIPS.

## Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>valeur</i>	Obligatoire	Type de données binaire. Valeur que vous voulez décrypter.
<i>clé</i>	Obligatoire	Type de données Chaîne. Précision inférieure ou égale à 16 caractères. Vous pouvez utiliser des variables de mappage pour la clé. Pour décrypter une valeur, utilisez la même clé que celle utilisée pour la crypter.

## Valeur de retour

Valeur binaire décryptée.

NULL si la valeur d'entrée est une valeur nulle.

## Exemple

L'exemple suivant renvoie des numéros de sécurité sociale décryptés. Dans cet exemple, le Service d'intégration de données dérive la clé à partir des trois premiers chiffres du numéro de sécurité sociale à l'aide de la fonction 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

Revoie des données au format crypté. Le Service d'intégration de données utilise l'algorithme standard de cryptage avancé (AES) avec un codage 128 bits. L'algorithme AES est un algorithme cryptographique certifié FIPS.

Utilisez cette fonction pour empêcher que des données sensibles soient visibles par tout le monde. Par exemple, pour stocker des numéros de sécurité sociale dans un entrepôt de données, utilisez la fonction AES\_ENCRYPT pour crypter les numéros de sécurité sociale et garantir la confidentialité.

## Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>valeur</i>	Obligatoire	Type de données Chaîne. Valeur que vous voulez crypter.
<i>clé</i>	Obligatoire	Type de données Chaîne. Précision inférieure ou égale à 16 caractères. Vous pouvez utiliser des variables de mappage pour la clé.

## Valeur de retour

Valeur binaire cryptée.

NULL si l'entrée est une valeur nulle.

## Exemple

L'exemple suivant renvoie les valeurs cryptées de numéros de sécurité sociale. Dans cet exemple, le Service d'intégration de données dérive la clé à partir des trois premiers chiffres du numéro de sécurité sociale à l'aide de la fonction 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

## Conseil

Si la cible ne prend pas en charge les données binaires, utilisez AES\_ENCRYPT avec la fonction ENC\_BASE64 pour stocker les données dans un format compatible avec la base de données.

# ANY

Renvoie n'importe quelle ligne dans le port sélectionné. Vous pouvez également appliquer un filtre pour limiter les lignes lues par le service d'intégration de données. Vous ne pouvez imbriquer qu'une seule autre fonction Agrégation dans ANY.

## Syntaxe

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

Le tableau suivant décrit les arguments de cette fonction :

Argument	Obligatoire / Facultatif	Description
<i>value</i>	Obligatoire	Tout type de données excepté Binaire. Transmet les valeurs pour lesquelles vous voulez renvoyer n'importe quelle ligne. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>filter_condition</i>	Facultatif	Limite les lignes dans la recherche. La condition de filtre doit être une valeur numérique ou renvoyer TRUE, FALSE ou NULL. Vous pouvez entrer l'expression de transformation valide de votre choix.

## Valeur de retour

N'importe quelle ligne dans un port. Renvoie une ligne différente à chaque fois.

NULL si toutes les valeurs transmises à la fonction sont NULL, ou si aucune ligne n'est sélectionnée. Par exemple, la condition de filtre renvoie FALSE ou NULL pour toutes les lignes.

## Exemple

L'expression suivante renvoie n'importe quelle ligne dans le port ITEM\_NAME avec un prix supérieure à 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

Lorsque vous configurez l'exécution du Service d'intégration de données en mode ASCII, la fonction ASCII renvoie la valeur numérique ASCII du premier caractère de la chaîne transmise à la fonction.

Lorsque vous configurez l'exécution du Service d'intégration de données en mode Unicode, la fonction ASCII renvoie la valeur numérique Unicode du premier caractère de la chaîne transmise à la fonction. Les valeurs Unicode sont comprises dans la plage 0 à 65535.

Vous pouvez convertir une chaîne, quelle que soit sa taille, en ASCII, mais elle n'évalue que le premier caractère de la chaîne. Avant de convertir une valeur de chaîne en ASCII, vous pouvez analyser les caractères spécifiques que vous voulez convertir en valeur ASCII ou Unicode. Par exemple, vous pouvez utiliser RTRIM ou une autre fonction de manipulation de chaîne. Si vous transmettez une valeur numérique, ASCII la convertit en une chaîne de caractères et renvoie la valeur ASCII ou Unicode du premier caractère de la chaîne.

Le comportement de cette fonction est identique à la fonction CHRCODE. Si vous utilisez ASCII dans des expressions existantes, elles continueront à fonctionner correctement. Cependant, lorsque vous créez de nouvelles expressions, utilisez la fonction CHRCODE au lieu de la fonction ASCII.

## Syntaxe

```
ASCII ( string )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>chaîne</i>	Obligatoire	Chaîne de caractères. Convertit la valeur que vous voulez renvoyer en valeur ASCII. Vous pouvez entrer l'expression de transformation valide de votre choix.

## Valeur de retour

Entier. Valeur ASCII ou Unicode du premier caractère de la chaîne.

NULL si une valeur transmise à la fonction est NULL.

## Exemple

L'expression suivante renvoie la valeur ASCII ou Unicode pour le premier caractère de chaque valeur dans le port ITEMS :

```
ASCII( ITEMS )
```

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

# AVG

Renvoie la moyenne de toutes les valeurs dans un groupe de lignes. Vous pouvez également appliquer un filtre de façon à limiter les lignes à lire pour calculer la moyenne. Vous pouvez imbriquer une seule autre fonction Agrégation dans AVG et la fonction imbriquée doit renvoyer un type de données numérique.

## Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Obligatoire	Type de données numérique. Convertit les valeurs pour lesquelles vous souhaitez calculer une moyenne. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>filter_condition</i>	Facultatif	Limite les lignes dans la recherche. La condition de filtre doit être une valeur numérique ou renvoyer TRUE, FALSE ou NULL. Vous pouvez entrer l'expression de transformation valide de votre choix.

## Valeur de retour

Valeur numérique.

NULL si toutes les valeurs transmises à la fonction sont NULL ou si aucune ligne n'est sélectionnée. Par exemple, la condition de filtre renvoie FALSE ou NULL pour toutes les lignes.

**Remarque:** Si la valeur de retour est décimale avec une précision supérieure à 15, vous pouvez activer la précision élevée pour garantir une précision décimale jusqu'à 28 chiffres.

## Valeurs nulles

Si une valeur est nulle, AVG ignore la ligne. Cependant, si toutes les valeurs transmises à partir du port sont NULL, AVG renvoie NULL.

## Grouper par

AVG groupe des valeurs en fonction des ports de regroupement que vous avez définis dans la transformation et renvoie un résultat pour chaque groupe.

S'il n'existe aucun port de regroupement, AVG traite toutes les lignes comme un seul groupe et renvoie une seule valeur.

## Exemple

L'expression suivante renvoie le prix de gros moyen de lampes de poche :

```
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

## Conseil

Vous pouvez effectuer des opérations arithmétiques sur les valeurs transmises à AVG avant le calcul de la moyenne par la fonction. Par exemple :

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

# CEIL

Renvoie le plus petit entier supérieur ou égal à la valeur numérique transmise à cette fonction. Par exemple, si vous transmettez 3,14 à CEIL, la fonction renvoie 4. Si vous transmettez 3,98 à CEIL, la fonction renvoie 4. De la même manière, si vous transmettez -3,17 à CEIL, la fonction renvoie -3.

## Syntaxe

```
CEIL( numeric_value )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire/ Facultatif	Description
<i>numeric_value</i>	Obligatoire	Type de données numérique. Vous pouvez entrer l'expression de transformation valide de votre choix.

## Valeur de retour

Nombre entier si vous transmettez une valeur numérique avec une précision déclarée entre 0 et 28.

Valeur Double si vous transmettez une valeur numérique avec une précision déclarée supérieure à 28.

NULL si une valeur transmise à la fonction est NULL.

## Exemple

L'expression suivante renvoie le prix arrondi au nombre entier suivant :

```
CEIL( PRICE )
```

PRICE	RETURN VALUE
39.79	40
125.12	126
74.24	75

PRICE	RETURN VALUE
NULL	NULL
-100.99	-100

**Astuce:** Vous pouvez effectuer des opérations arithmétiques sur les valeurs transmises à CEIL avant que CEIL renvoie la valeur entière suivante. Par exemple, si vous voulez multiplier une valeur numérique par 10 avant de calculer le plus petit entier inférieur à la valeur modifiée, vous pouvez écrire la fonction comme suit :

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

## CHOOSE

Choisit une chaîne parmi une liste de chaînes d'après une position donnée. Spécifiez la position et la valeur. Si la valeur correspond à la position, le Service d'intégration de données renvoie la valeur.

### Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>index</i>	Obligatoire	Type de données numérique. Entrez un nombre basé sur la position de la valeur de correspondance.
<i>chaîne</i>	Obligatoire	Toute valeur de caractère.

### Valeur de retour

Chaîne qui correspond à la position de la valeur d'index.

NULL si aucune chaîne ne correspond à la position de la valeur d'index.

### Exemple

L'expression suivante renvoie la chaîne « flashlight » en fonction d'une valeur d'index de 2 :

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

L'expression suivante renvoie NULL en fonction d'une valeur d'index de 4 :

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

CHOOSE renvoie NULL, car l'expression ne contient pas de quatrième argument.

# CHR

Lorsque vous configurez le Service d'intégration de données pour déplacer des données en mode ASCII, CHR renvoie le caractère ASCII correspondant à la valeur numérique que vous transmettez à cette fonction. Les valeurs ASCII sont comprises dans la plage 0 à 255. Vous pouvez transmettre n'importe quel nombre entier à CHR, mais seuls les codes ASCII 32 à 126 sont des caractères imprimables.

Lorsque vous configurez le Service d'intégration de données pour déplacer des données en mode Unicode, CHR renvoie le caractère Unicode correspondant à la valeur numérique que vous transmettez à cette fonction. Les valeurs Unicode sont comprises dans la plage 0 à 65535.

## Syntaxe

```
CHR( numeric_value )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Obligatoire	Type de données numérique. Valeur que vous voulez renvoyer comme caractère ASCII ou Unicode. Vous pouvez entrer l'expression de transformation valide de votre choix.

## Valeur de retour

Caractère ASCII ou Unicode. Chaîne contenant un caractère.

NULL si une valeur transmise à la fonction est NULL.

## Exemple

L'expression suivante renvoie le caractère ASCII ou Unicode pour chaque valeur numérique dans le port ITEM\_ID :

```
CHR( ITEM_ID )
```

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

Utilisez la fonction CHR pour concaténer un guillemet simple dans une chaîne. Le guillemet simple est le seul caractère que vous ne pouvez pas utiliser dans un littéral chaîne. Considérons l'exemple suivant :

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

La valeur de retour est :

```
Joan's car
```

# CHRCODE

Lorsque vous configurez l'exécution du Service d'intégration de données en mode ASCII, CHRCODE renvoie la valeur numérique ASCII du premier caractère de la chaîne transmise à la fonction. Les valeurs ASCII sont comprises dans la plage 0 à 255.

Lorsque vous configurez l'exécution du Service d'intégration de données en mode Unicode, CHRCODE renvoie la valeur numérique Unicode du premier caractère de la chaîne transmise à la fonction. Les valeurs Unicode sont comprises dans la plage 0 à 65535.

En règle générale, avant de convertir une valeur de chaîne en CHRCODE, vous devez analyser les caractères spécifiques à convertir en valeur ASCII ou Unicode. Par exemple, vous pouvez utiliser RTRIM ou une autre fonction de manipulation de chaîne. Si vous transmettez une valeur numérique, CHRCODE la convertit en une chaîne de caractères et renvoie la valeur ASCII ou Unicode du premier caractère dans la chaîne.

Le comportement de cette fonction est identique à la fonction ASCII. Si vous utilisez actuellement ASCII dans les expressions, elle continuera de fonctionner correctement. Cependant, lorsque vous créez de nouvelles expressions, utilisez la fonction CHRCODE au lieu de la fonction ASCII.

## Syntaxe

```
CHRCODE ( string )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>chaîne</i>	Obligatoire	Chaîne de caractères. Transmet les valeurs que vous voulez renvoyer comme valeurs ASCII ou Unicode. Vous pouvez entrer l'expression de transformation valide de votre choix.

## Valeur de retour

Entier.

NULL si une valeur transmise à la fonction est NULL.

## Exemple

L'expression suivante renvoie la valeur ASCII ou Unicode pour le premier caractère de chaque valeur dans le port ITEMS :

```
CHRCODE( ITEMS )
```

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

# COMPRESS

Comprime des données à l'aide de l'algorithme de compression zlib 1.2.1. Utilisez la fonction COMPRESS avant d'envoyer des quantités volumineuses de données dans un réseau étendu.

## Syntaxe

```
COMPRESS( value )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire/ Facultatif	Description
<i>value</i>	Obligatoire	Type de données Chaîne. Données que vous voulez compresser.

## Valeur de retour

Valeur binaire compressée de la valeur d'entrée.

NULL si l'entrée est une valeur nulle.

## Exemple

Votre entreprise dispose d'un service de commandes sur internet. Vous voulez envoyer les données des commandes client via un réseau étendu. La source contient une ligne de 10 Mo. Vous pouvez compresser les données dans cette ligne à l'aide de COMPRESS. Lorsque vous compressez les données, vous réduisez la quantité de données que le Service d'intégration de données écrit dans le réseau. Par conséquent, vous pouvez améliorer les performances.

# CONCAT

Permet de concaténer deux chaînes. CONCAT convertit toutes les données en texte avant de concaténer les chaînes. Vous pouvez également utiliser l'opérateur de chaîne || pour concaténer des chaînes. En utilisant l'opérateur de chaîne || au lieu de CONCAT, les performances du Service d'intégration de données s'améliorent.

## Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>first_string</i>	Obligatoire	Tout type de données excepté Binaire. Première partie de la chaîne que vous voulez concaténer. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>second_string</i>	Obligatoire	Tout type de données excepté Binaire. Deuxième partie de la chaîne à concaténer. Vous pouvez entrer l'expression de transformation valide de votre choix.

## Valeur de retour

Chaîne.

NULL si les deux valeurs de chaîne sont nulles.

## Valeurs nulles

Si une des chaînes est NULL, CONCAT l'ignore et renvoie l'autre chaîne.

Si les deux chaînes sont NULL, CONCAT renvoie NULL.

## Exemple

L'expression suivante concatène les noms dans les ports FIRST\_NAME et 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 n'ajoute pas d'espace pour séparer les chaînes. Si vous voulez ajouter un espace entre deux chaînes, vous pouvez écrire une expression contenant deux fonctions CONCAT imbriquées. Par exemple, l'expression suivante concatène d'abord un espace à la fin du prénom, puis concatène le nom de famille :

```
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

Utilisez les fonctions CHR et CONCAT pour concaténer un guillemet simple dans une chaîne. Le guillemet simple est le seul caractère que vous ne pouvez pas utiliser dans un littéral chaîne. Considérons l'exemple suivant :

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

La valeur de retour est :

```
Joan's car
```

## CONVERT\_BASE

Convertit un nombre à partir d'une valeur de base en une autre valeur de base.

### Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>value</i>	Obligatoire	Type de données Chaîne. Valeur que vous souhaitez convertir d'une base vers une autre base. La valeur maximum est de 9 233 372 036 854 775 806.
<i>source_base</i>	Obligatoire	Type de données numérique. Valeur de base actuelle des données à convertir. La base minimale est 2. La base maximale est 36.
<i>dest_base</i>	Obligatoire	Type de données numérique. Valeur de base vers laquelle vous souhaitez convertir les données. La base minimale est 2. La base maximale est 36.

### Valeur de retour

Valeur numérique.

### Exemple

L'exemple suivant convertit 2 222 à partir de la base de base décimale 10 en valeur de base binaire 2 :

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

Le Service d'intégration de données renvoie 100010101110.

## COS

Renvoie le cosinus d'une valeur numérique (exprimé en radians).

### Syntaxe

```
COS( numeric_value )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Obligatoire	Type de données numérique. Données numériques exprimées en radians (degrés multipliés par pi, divisés par 180). Transmet les valeurs pour lesquelles vous souhaitez calculer le cosinus. Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

Valeur Double.

NULL si une valeur transmise à la fonction est NULL.

### Exemple

L'expression suivante renvoie le cosinus de toutes les valeurs dans le port 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

**Astuce:** Vous pouvez effectuer des opérations arithmétiques sur les valeurs transmises à COS avant le calcul du cosinus par la fonction. Par exemple, vous pouvez convertir les valeurs dans le port en radians avant de calculer le cosinus, comme suit :

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

## COSH

Renvoie le cosinus hyperbolique d'une valeur numérique (exprimé en radians).

### Syntaxe

```
COSH( numeric_value )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Obligatoire	Type de données numérique. Données numériques exprimées en radians (degrés multipliés par pi, divisés par 180). Convertit les valeurs pour lesquelles vous souhaitez calculer le cosinus hyperbolique. Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

Valeur Double.

NULL si une valeur transmise à la fonction est NULL.

### Exemple

L'expression suivante renvoie le cosinus hyperbolique des valeurs dans le port 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

**Astuce:** Vous pouvez effectuer des opérations arithmétiques sur les valeurs transmises à COSH avant le calcul du cosinus hyperbolique par la fonction. Par exemple :

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

## COUNT

Renvoie le nombre de lignes ayant des valeurs non nulles dans un groupe. Vous pouvez également inclure l'argument astérisque (\*) pour compter toutes les valeurs d'entrée dans une transformation. Vous pouvez imbriquer une seule autre fonction Agrégation dans la fonction COUNT. Vous pouvez appliquer une condition pour filtrer des lignes avant de les compter.

### Syntaxe

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

ou

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>valeur</i>	Obligatoire	Tout type de données excepté Binaire. Convertit les valeurs que vous voulez compter. Vous pouvez entrer l'expression de transformation valide de votre choix.
*	Facultatif	Utilisez pour compter <i>toutes les lignes</i> dans une transformation.
<i>filter_condition</i>	Facultatif	Limite les lignes dans la recherche. La condition de filtre doit être une valeur numérique ou renvoyer TRUE, FALSE ou NULL. Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

Entier.

0 si toutes les valeurs transmises à cette fonction sont NULL (sauf si vous incluez l'argument astérisque).

### Valeurs nulles

Si toutes les valeurs sont nulles, la fonction renvoie 0.

Si vous appliquez l'argument astérisque, cette fonction compte toutes les lignes, sans tenir compte d'éventuelles valeurs nulles dans la colonne d'une ligne.

Si vous appliquez l'argument *valeur*, cette fonction ignore les colonnes contenant des valeurs nulles.

### Grouper par

COUNT groupe des valeurs en fonction des ports de regroupement que vous définissez dans la transformation et renvoie un résultat pour chaque groupe. S'il n'existe aucun port de regroupement, COUNT traite toutes les lignes comme un seul groupe et renvoie une seule valeur.

### Exemples

L'expression suivante compte les éléments dont la quantité en stock est inférieure à 5, en excluant les valeurs nulles :

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

ITEM_NAME	IN_STOCK
Flashlight	10
NULL	2
Compass	NULL
Regulator System	5
Safety Knife	8

ITEM_NAME	IN_STOCK
Halogen Flashlight	1

**RETURN VALUE:** 1

Dans cet exemple, la fonction a compté la lampe de poche halogène, mais pas l'élément NULL. La fonction compte toutes les lignes dans une transformation, y compris les valeurs nulles, comme l'illustre l'exemple suivant :

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

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

**RETURN VALUE:** 2

Dans cet exemple, la fonction compte l'élément NULL et la lampe de poche halogène. Si vous incluez l'argument astérisque, mais que vous n'utilisez aucun filtre, la fonction compte toutes les lignes transmises à la transformation. Par exemple :

```
COUNT( * )
```

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

**RETURN VALUE:** 6

# CRC32

Renvoie une valeur de contrôle de redondance cyclique 32 bits (CRC32). Utilisez CRC32 pour rechercher des erreurs de transmission de données. Vous pouvez également utiliser CRC32 si vous souhaitez vérifier que les données stockées dans un fichier n'ont pas été modifiées.

Si vous utilisez CRC32 pour effectuer une vérification de redondance des données en mode ASCII et en mode Unicode, le Service d'intégration de données peut générer des résultats différents pour la même valeur d'entrée. Si vous utilisez CRC32 pour effectuer un contrôle de redondance des données sur différents systèmes d'exploitation, le Service d'intégration de données peut générer des résultats différents pour la même valeur d'entrée.

**Remarque:** CRC32 peut renvoyer la même sortie pour différentes chaînes d'entrée. Si vous souhaitez générer des clés dans un mappage, utilisez une transformation Générateur de séquence. Si vous utilisez CRC32 pour générer des clés dans un mappage, il est possible que vous obteniez des résultats inattendus.

## Syntaxe

```
CRC32( value )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>value</i>	Obligatoire	Type de données Chaîne ou Binaire. Transmet les valeurs pour lesquelles vous voulez effectuer une vérification de redondance. La valeur d'entrée est sensible à la casse. La casse de la valeur d'entrée affecte la valeur de retour. Par exemple, CRC32(informatica) et CRC32 (Informatica) renvoient des valeurs différentes.

## Valeur de retour

Valeur entière 32 bits.

## Exemple

Vous voulez lire les données à partir d'une source dans un réseau étendu. Vous voulez vous assurer que les données ont été modifiées lors de la transmission. Vous pouvez calculer la somme de contrôle pour les données du fichier et la stocker également dans le fichier. Lors de la lecture des données source, le Service d'intégration de données peut utiliser CRC32 pour calculer la somme de contrôle et la comparer à la valeur stockée. Si les deux valeurs sont identiques, les données n'ont pas été modifiées.

# CUME

Renvoie un total cumulé. Un total cumulé signifie que CUME renvoie un total à chaque fois qu'il ajoute une valeur. Vous pouvez ajouter une condition pour filtrer les lignes hors de l'ensemble de lignes avant de calculer le total cumulé.

Utilisez la fonction CUME et d'autres fonctions similaires (par exemple, MOVINGAVG et MOVINGSUM) pour simplifier le rapport en calculant des valeurs cumulées.

## Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Obligatoire	Type de données numérique. Transmet les valeurs pour lesquelles vous souhaitez calculer un total cumulé. Vous pouvez entrer l'expression de transformation valide de votre choix. Vous pouvez créer une expression imbriquée pour calculer un total cumulé basé sur les résultats de la fonction dans la mesure où le résultat est une valeur numérique.
<i>filter_condition</i>	Facultatif	Limite les lignes dans la recherche. La condition de filtre doit être une valeur numérique ou renvoyer TRUE, FALSE ou NULL. Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

Valeur numérique.

NULL si toutes les valeurs passées à la fonction sont NULL, ou si aucune ligne n'est sélectionnée (par exemple, la condition de filtre renvoie FALSE ou NULL pour toutes les lignes).

**Remarque:** Si la valeur de retour est décimale avec une précision supérieure à 15, vous pouvez activer la précision élevée pour garantir une précision décimale jusqu'à 28 chiffres.

### Valeurs nulles

Si une valeur est nulle, CUME renvoie le total cumulé pour la ligne précédente. Cependant, si toutes les valeurs dans le port sélectionné sont NULL, CUME renvoie NULL.

### Exemples

L'exemple de rowset ci-dessous peut découler de l'utilisation de la fonction CUME :

```
CUME ( PERSONAL_SALES )
```

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

De la même manière, vous pouvez ajouter des valeurs avant de calculer un total cumulé :

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

CA_SALES	OR_SALES	RETURN VALUE
40000	10000	50000

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

## DATE\_COMPARE

Renvoie un entier indiquant quelle date est la première. DATE\_COMPARE renvoie une valeur d'entier et non une valeur de date.

### Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>date1</i>	Obligatoire	Type de données Date/Heure. Première date à comparer. Vous pouvez entrer une expression de transformation valide à condition qu'elle renvoie une date.
<i>date2</i>	Obligatoire	Type de données Date/Heure. Deuxième date à comparer. Vous pouvez entrer une expression de transformation valide à condition qu'elle renvoie une date.

### Valeur de retour

-1 si la première date est antérieure.

0 si les deux dates sont identiques.

1 si la seconde date est antérieure.

NULL si une des valeurs de date est NULL.

### Exemple

L'expression suivante compare chaque date dans les ports DATE\_PROMISED et DATE\_SHIPPED et renvoie un nombre entier indiquant la date antérieure :

```
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

Renvoie la durée entre deux dates. Vous pouvez demander à ce que le format soit des années, des mois, des jours, des heures, des minutes, des secondes, des millisecondes, des microsecondes ou des nanosecondes. Le Service d'intégration de données soustrait la deuxième date à la première date et renvoie la différence.

Le Service d'intégration de données calcule la fonction DATE\_DIFF en fonction du nombre de mois au lieu du nombre de jours. Il calcule les différences de date pour les mois incomplets avec les jours sélectionnés dans chaque mois. Pour calculer la différence de date pour le mois incomplet, le Service d'intégration de données ajoute les jours utilisés dans le mois. Ensuite, il divise la valeur par le nombre total de jours dans le mois sélectionné.

Le Service d'intégration de données indique une valeur différente pour la même période dans une année bissextile et une année non bissextile. La différence se produit lorsque le mois de février est inclus dans la fonction DATE\_DIFF. La fonction DATE\_DIFF divise les jours par 29 pour le mois de février dans une année bissextile et 28 s'il ne s'agit pas d'une année bissextile.

Par exemple, vous souhaitez calculer le nombre de mois écoulés entre le 13 septembre et le 19 février. Dans une période d'année bissextile, la fonction DATE\_DIFF calcule le mois de février comme 19/29 mois ou 0,655 mois. Dans une période d'année non bissextile, la fonction DATE\_DIFF calcule le mois de février comme 19/28 mois ou 0,678 mois. De la même façon, le Service d'intégration de données calcule la différence entre les dates pour les mois restants et la fonction DATE\_DIFF renvoie la valeur spécifiée pour la période.

**Remarque:** Certaines bases de données peuvent utiliser un algorithme différent pour calculer la différence entre les dates.

### Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>date1</i>	Obligatoire	Type de données Date/Heure. Transmet les valeurs pour la première date à comparer. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>date2</i>	Obligatoire	Type de données Date/Heure. Transmet les valeurs pour la deuxième date que vous voulez comparer. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>format</i>	Obligatoire	Chaîne de format spécifiant la mesure de la date ou de l'heure. Vous pouvez spécifier des années, des mois, des jours, des heures, des minutes, des secondes, des millisecondes, des microsecondes ou des nanosecondes. Vous pouvez spécifier uniquement une partie de la date, par exemple 'mm'. Placez les chaînes de format entre guillemets simples. La chaîne de format n'est pas sensible à la casse. Par exemple, la chaîne de format 'mm' est identique à 'MM', 'Mm' ou 'mM'.

### Valeur de retour

Valeur Double. Si la *date1* est ultérieure à la *date2*, la valeur de retour sera un nombre positif. Si la *date1* est antérieure à la *date2*, la valeur de retour sera un nombre négatif.

0 si les dates sont identiques.

NULL si une ou les deux valeurs de date sont nulles.

### Exemples

Les expressions suivantes renvoient le nombre d'heures entre les ports DATE\_PROMISED et 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

Les expressions suivantes renvoient le nombre de jours entre les ports DATE\_PROMISED et 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

Les expressions suivantes renvoient le nombre de mois entre les ports DATE\_PROMISED et 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

Les expressions suivantes renvoient le nombre d'années entre les ports DATE\_PROMISED et DATE\_SHIPPED :

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

DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
Jan 1 1997 12:00:00AM	Mar 29 1997 12:00:00PM	-0.24327956989247
Mar 29 1997 12:00:00PM	Jan 1 1997 12:00:00AM	0.24327956989247
NULL	Dec 10 1997 5:55:10PM	NULL
Dec 10 1997 5:55:10PM	NULL	NULL
Jun 3 1997 1:13:46PM	Aug 23 1996 4:20:16PM	0.77741801697531
Feb 19 2004 12:00:00PM	Feb 19 2005 12:00:00PM	-1

Les expressions suivantes renvoient le nombre de mois entre les ports DATE\_PROMISED et 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

Décode une valeur codée en base 64 et renvoie une chaîne avec la représentation de données binaires des données. Si vous codez des données à l'aide de ENC\_BASE64 et que vous voulez décoder des données à l'aide de DEC\_BASE64, vous devez exécuter le mappage avec le même mode de mouvement de données. Dans le cas contraire, la sortie des données décodées peut être différente des données d'origine.

### Syntaxe

```
DEC_BASE64( value )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire/ Facultatif	Description
<i>value</i>	Obligatoire	Type de données Chaîne. Données que vous voulez décoder.

### Valeur de retour

Valeur décodée binaire.

NULL si l'entrée est une valeur nulle.

Les valeurs de retour diffèrent selon l'exécution du mappage en mode Unicode ou en mode ASCII.

## DECODE

Recherche dans un port une valeur que vous avez spécifiée. Si la fonction trouve la valeur, elle renvoie une valeur de résultat, que vous définissez. Vous pouvez créer un nombre de recherches illimité à l'intérieur d'une fonction DECODE.

Si vous utilisez DECODE pour rechercher une valeur dans un port de chaîne, vous pouvez couper les espaces de fin à l'aide de la fonction RTRIM ou les inclure dans la chaîne de recherche.

## Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>valeur</i>	Obligatoire	Tout type de données excepté Binaire. Transmet les valeurs que vous voulez rechercher. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>recherche</i>	Obligatoire	Toute valeur dont le type de données est identique à l'argument de la valeur. Transmet les valeurs que vous voulez rechercher. La valeur de recherche doit correspondre à l'argument de valeur. Vous ne pouvez pas rechercher une partie d'une valeur. En outre, la valeur de recherche est sensible à la casse.  Par exemple, si vous voulez rechercher la chaîne « Halogen Flashlight » dans un port spécifique, vous devez saisir « Halogen Flashlight », pas uniquement « Halogen ». Si vous entrez 'Halogen', la recherche ne trouve aucune valeur correspondante. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>résultat</i>	Obligatoire	Tout type de données excepté Binaire. Valeur que vous voulez renvoyer si la recherche trouve une valeur correspondante. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>valeur par défaut</i>	Facultatif	Tout type de données excepté Binaire. Valeur à renvoyer si la recherche ne trouve pas de valeur correspondante. Vous pouvez entrer l'expression de transformation valide de votre choix.

## Valeur de retour

*First\_result* si la recherche trouve une valeur correspondante.

Valeur par défaut si la recherche ne trouve aucune valeur correspondante.

NULL si vous omettez l'argument par défaut et que la recherche ne trouve aucune valeur correspondante.

Même si plusieurs conditions sont satisfaites, le Service d'intégration de données renvoie le premier résultat correspondant.

Si les données contiennent des caractères multioctets et que l'expression DECODE compare les données de chaîne, la valeur de retour dépend de la page de code et du mode de mouvement de données du Service d'intégration de données.

## fonction DECODE et types de données

Lorsque vous utilisez la fonction DECODE, le type de données de la valeur de retour est toujours identique à celui du résultat dont la précision est la plus élevée.

Par exemple, prenons l'expression suivante :

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

Les valeurs de retour dans cette expression sont 5, 1.414213562 et 3.141592654. Le premier résultat est un nombre entier et les autres sont des décimaux. La précision du type de données décimal est supérieure à celle d'un nombre entier. Cette expression écrit toujours le résultat sous forme de décimal.

Lorsque vous exécutez un mappage en mode précision élevée, si au moins un résultat est de type double, le type de données de la valeur de retour sera double.

Vous ne pouvez pas créer de fonction DECODE avec les valeurs de retour de chaîne et numériques.

Par exemple, l'expression suivante n'est pas valide :

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

Lorsque vous validez l'expression ci-dessus, vous recevez le message d'erreur suivant :

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

## Exemples

Vous pouvez utiliser DECODE dans une expression qui recherche un ITEM\_ID spécifique et renvoie 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 renvoie la valeur par défaut NONE pour les éléments 17 et 25, car les valeurs de recherche ne correspondent pas à ITEM\_ID. En outre, DECODE renvoie NONE pour ITEM\_ID NULL.

L'expression suivante teste plusieurs colonnes et conditions, évaluées de haut en bas sur TRUE ou FALSE :

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

Var1	Var2	RETURN VALUE
21	47	Variable 1 was less than 23.
22	49	Variable 1 was 22!
23	49	Variable 2 was 49!

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

## DECOMPRESS

Décompresse les données à l'aide de l'algorithme de compression zlib 1.2.1. Utilisez la fonction DECOMPRESS sur les données qui ont été compressées avec la fonction COMPRESS ou un outil de compression utilisant l'algorithme zlib 1.2.1. Si le mappage qui décompresse les données utilise un mode de mouvement de données différent du mappage qui a compressé les données, la sortie des données décompressées peut être différente de celle des données d'origine.

### Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire/ Facultatif	Description
<i>valeur</i>	Obligatoire	Type de données binaire. Données que vous voulez décompresser.
<i>précision</i>	Facultatif	Type de données Nombre entier.

### Valeur de retour

Valeur binaire décompressée de la valeur d'entrée.

NULL si l'entrée est une valeur nulle.

## ENC\_BASE64

Code les données en convertissant les données binaires en données de chaînes à l'aide du codage Multipurpose Internet Mail Extensions (MIME). Codent des données que vous voulez stocker dans une base de données ou un fichier qui n'autorise pas les données binaires. Vous pouvez également encoder des données pour transmettre des données binaires via des transformations au format de chaîne. Les données encodées sont approximativement 33 % plus longues que les données d'origine. Elles s'affichent sous forme de jeu de caractères aléatoires.

### Syntaxe

```
ENC_BASE64( value )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire/ Facultatif	Description
<i>valeur</i>	Obligatoire	Type de données binaire ou de chaîne. Données que vous voulez encoder.

### Valeur de retour

Valeur encodée.

NULL si l'entrée est une valeur nulle.

## ERREUR

Entraîne le Service d'intégration de données à ignorer une ligne et émettre un message d'erreur, que vous définissez. Le message d'erreur s'affiche dans le journal. Le Service d'intégration de données n'écrit pas les lignes ignorées dans le fichier de rejet.

Utilisez la fonction ERROR dans des transformations Expression pour valider des données. En règle générale, utilisez ERROR dans une fonction IIF ou DECODE afin de définir des règles pour ignorer des lignes.

Utilisez la fonction ERROR pour les valeurs par défaut de ports d'entrée et de sortie. Vous pouvez utiliser ERROR de sorte que les ports d'entrée ne transmettent pas les valeurs nulles dans une transformation.

Utilisez ERROR pour des ports de sortie de sorte à gérer tout type d'erreur de transformation, y compris des appels de la fonction ERROR dans une expression. Lorsque vous utilisez la fonction ERROR dans une expression et dans la valeur par défaut du port de sortie, le Service d'intégration de données ignore la ligne et journalise le message d'erreur de l'expression et celui de la valeur par défaut. Si vous voulez vous assurer que le Service d'intégration de données ignore les lignes qui produisent une erreur, affectez ERROR comme valeur par défaut.

Si vous utilisez une valeur par défaut de sortie autre que ERROR, la valeur par défaut écrase la fonction ERROR dans une expression. Par exemple, utilisez la fonction ERROR dans une expression et attribuez la valeur par défaut, « 1234 », au port de sortie. Chaque fois que le Service d'intégration de données rencontre la fonction ERROR dans l'expression, il remplace l'erreur par la valeur « 1234 » et transmet « 1234 » à la transformation suivante. Il n'ignore pas la ligne et ne journalise pas d'erreur dans le journal.

### Syntaxe

```
ERROR( string )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>chaîne</i>	Obligatoire	Valeur de chaîne. Message vous souhaitez afficher lorsque le service d'intégration ignore une ligne en fonction de l'expression contenant la fonction ERROR. La chaîne peut être de n'importe quelle longueur.

### Valeur de retour

Chaîne.

## Exemple

L'exemple suivant illustre le référencement d'un mappage qui calcule le salaire moyen des employés dans tous les services de l'entreprise, en ignorant les valeurs négatives. L'expression suivante imbrique la fonction ERROR dans une expression IIF de sorte que si le Service d'intégration de données trouve un salaire négatif dans le port Salary, il ignore la ligne et affiche une erreur :

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

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

## EXP

Renvoie une valeur  $e$  élevée à la puissance spécifiée (exposant), où  $e = 2,71828183$ . Par exemple,  $\text{EXP}(2)$  renvoie 7,38905609893065. Vous pouvez utiliser cette fonction pour analyser des données scientifiques et techniques au lieu de données métier. EXP est la valeur réciproque de la fonction LN, qui renvoie le logarithme népérien d'une valeur numérique.

### Syntaxe

```
EXP( exponent )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>exposant</i>	Obligatoire	Type de données numérique. Valeur que vous voulez élever. Il s'agit de l'exposant dans l'équation $e^{\text{valeur}}$ . Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

Valeur Double.

NULL si une valeur transmise comme argument pour la fonction est nulle.

## Exemple

L'expression suivante utilise les valeurs stockées dans le port Numbers comme valeur exposant :

```
EXP ( NUMBERS )
```

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

## FIRST

Revoit la première valeur trouvée dans un port ou un groupe. Vous pouvez également appliquer un filtre pour limiter les lignes lues par le Service d'intégration de données. Vous pouvez imbriquer une seule autre fonction Agrégation dans la fonction FIRST.

### Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>value</i>	Obligatoire	Tout type de données excepté Binaire. Transmet les valeurs pour lesquelles vous voulez renvoyer la première valeur. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>filter_condition</i>	Facultatif	Limite les lignes dans la recherche. La condition de filtre doit être une valeur numérique ou renvoyer TRUE, FALSE ou NULL. Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

Première valeur dans un groupe.

NULL si toutes les valeurs transmises à la fonction sont NULL, ou si aucune ligne n'est sélectionnée (par exemple : la condition de filtre renvoie FALSE ou NULL pour toutes les lignes).

### Valeurs nulles

Si une valeur est nulle, FIRST ignore la ligne. Cependant, si toutes les valeurs transmises à partir du port sont NULL, FIRST renvoie NULL.

### Grouper par

FIRST groupe des valeurs en fonction du groupement par ports que vous définissez dans la transformation et renvoie un résultat pour chaque groupe.

S'il n'existe aucun port de regroupement, FIRST traite toutes les lignes comme un seul groupe et renvoie une seule valeur.

## Exemples

L'expression suivante renvoie la première valeur dans le port ITEM\_NAME et dont le montant est supérieur à 10,00 \$ :

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

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

**RETURN VALUE:** Flashlight

L'expression suivante renvoie la première valeur dans le port ITEM\_NAME et dont le montant est supérieur à 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

Renvoie le plus grand nombre entier inférieur ou égal à la valeur numérique passée à cette fonction. Par exemple, si vous passez 3,14 à FLOOR, la fonction renvoie 3. Si vous passez 3,98 à FLOOR, la fonction renvoie 3. De la même manière, si vous passez -3,17 à FLOOR, la fonction renvoie -4.

### Syntaxe

```
FLOOR( numeric_value )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Obligatoire	Type de données numérique. Vous pouvez saisir une expression de transformation valide dans la mesure où elle renvoie des données numériques.

### Valeur de retour

Nombre entier si vous transmettez une valeur numérique avec une précision déclarée entre 0 et 28.

Double si vous transmettez une valeur numérique dont la précision déclarée est supérieure à 28.

NULL si une valeur transmise à la fonction est NULL.

### Exemple

L'expression suivante renvoie le plus grand nombre entier inférieur ou égal aux valeurs dans le port PRICE :

```
FLOOR( PRICE )
```

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

**Astuce:** Vous pouvez effectuer des opérations arithmétiques dans les valeurs que vous passez à FLOOR. Par exemple, pour multiplier une valeur numérique par 10 et calculer le plus grand nombre entier inférieur au produit, vous pouvez écrire la fonction comme suit :

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

## FV

Renvoie la valeur future d'un investissement, lorsque vous effectuez des paiements constants et réguliers et que l'investissement bénéficie d'un taux d'intérêt constant.

### Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>taux</i>	Obligatoire	Numérique Taux d'intérêt appliqué à chaque période Exprimé sous forme de nombre décimal. Divisez le pourcentage par 100 pour l'exprimer sous forme de nombre décimal. Doit être supérieur ou égal à 0.
<i>termes</i>	Obligatoire	Numérique Nombre de périodes ou paiements. Doit être supérieur à 0.
<i>paiement</i>	Obligatoire	Numérique Montant de paiement dû par période. Doit être un nombre négatif
<i>valeur actuelle</i>	Facultatif	Numérique Valeur actuelle de l'investissement Si vous omettez cet argument, FV utilise 0.
<i>type</i>	Facultatif	Entier. Moment du paiement. Entrez 1 si le paiement est effectué en début de période. Entrez 0 si le paiement est effectué en fin de période. Par défaut 0. Si vous entrez une valeur différente de 0 ou 1, le Service d'intégration de données traite la valeur comme 1.

## Valeur de retour

Numérique

## Exemple

Vous déposez 2000 \$ dans un compte qui bénéficie d'un taux d'intérêt annuel de 9 % calculé mensuellement (taux d'intérêt mensuel de 9 % / 12, soit 0,75 %). Vous prévoyez de déposer 250 \$ au début de chaque mois pour les 12 prochains mois. L'expression suivante renvoie 5,337.96 \$ comme solde de compte à l'issue des 12 mois :

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

## Remarques

Pour calculer le taux d'intérêt appliqué à chaque période, divisez le taux annuel par le nombre de paiements effectués dans l'année. La valeur de paiement et la valeur actuelle sont négatives, car il s'agit de montants que vous payez.

# GET\_DATE\_PART

Renvoie la partie spécifique d'une date comme valeur entière. Par conséquent, si vous créez une expression qui renvoie la partie du mois de la date et que vous transmettez une date telle que Apr 1 1997 00:00:00, GET\_DATE\_PART renvoie 4.

## Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>date</i>	Obligatoire	Type de données Date/Heure. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>format</i>	Obligatoire	Chaîne de format spécifiant la partie de la valeur de date à renvoyer. Placez des chaînes de format entre guillemets simples, par exemple : 'mm'. La chaîne de format n'est pas sensible à la casse. Chaque chaîne de format renvoie la partie complète de la date en fonction du format de date spécifié dans le mappage.  Par exemple, si vous transmettez la date Apr 1 1997 à GET_DATE_PART, toutes les chaînes de format Y, YY, YYY, ou YYYY renvoient 1997.

### Valeur de retour

Nombre entier représentant la partie spécifique de la date.

NULL si une valeur transmise à la fonction est NULL.

### Exemples

Les expressions suivantes renvoient l'heure pour chaque date dans le port DATE\_SHIPPED. 12:00:00AM renvoie 0, car le format de date par défaut est basé sur un intervalle de 24 heures :

```
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

Les expressions suivantes renvoient le jour pour chaque date dans le port DATE\_SHIPPED :

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

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

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

Les expressions suivantes renvoient le mois pour chaque date dans le port 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

L'expression suivante renvoie l'année pour chaque date dans le port 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

## GREATEST

Renvoie la plus grande valeur d'une liste de valeurs d'entrée. Utilisez cette fonction pour renvoyer la chaîne, date ou nombre le plus grand. Par défaut, la correspondance est sensible à la casse.

### Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>valeur</i>	Obligatoire	Tout type de données excepté Binaire. Le type de données doit être compatible avec d'autres valeurs. Valeur à comparer à d'autres valeurs. Vous devez entrer au moins un argument de valeur.  Si la valeur est numérique et que d'autres valeurs d'entrée le sont également, toutes les valeurs utilisent la précision la plus élevée possible. Par exemple, si certaines valeurs sont de type nombre entier et d'autres de type double, le Service d'intégration de données convertit les valeurs en type double.
<i>CaseFlag</i>	Facultatif	Doit être un nombre entier. Détermine si les arguments de cette fonction sont sensibles à la casse. Vous pouvez entrer l'expression de transformation valide de votre choix.  Si <i>CaseFlag</i> est un nombre différent de 0, la fonction est sensible à la casse. Si <i>CaseFlag</i> est une valeur nulle ou 0, la fonction n'est pas sensible à la casse.

### Valeur de retour

*value1* s'il s'agit de la plus grande valeur d'entrée, *value2* s'il s'agit de la plus grande valeur d'entrée, etc.

NULL si l'un des arguments est nul.

### Exemple

L'expression suivante renvoie le plus grand nombre d'éléments commandés :

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

QUANTITY1	QUANTITY2	QUANTITY3	RETURN VALUE
150	756	27	756
			NULL
5000	97	17	5000
120	1724	965	1724

## IIF

Renvoie l'une des deux valeurs spécifiées, en fonction des résultats d'une condition.

### Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>condition</i>	Obligatoire	Condition que vous souhaitez évaluer. Vous pouvez saisir une expression de transformation valide qui renvoie TRUE ou FALSE.
<i>value1</i>	Obligatoire	Tout type de données excepté Binaire. Valeur que vous voulez renvoyer si la condition est TRUE. La valeur de retour est toujours le type de données spécifié par cet argument. Vous pouvez saisir une expression de transformation valide, y compris une autre expression IIF.
<i>value2</i>	Facultatif	Tout type de données excepté Binaire. Valeur que vous voulez renvoyer si la condition est FALSE. Vous pouvez saisir une expression de transformation valide, y compris une autre expression IIF.

Contrairement aux fonctions conditionnelles de certains systèmes, la condition FALSE (*value2*) dans la fonction IIF n'est pas requise. Si vous omettez *value2*, la fonction renvoie les éléments suivants lorsque la condition est FALSE :

- 0 si *value1* est un type de données numérique.
- Chaîne vide si *value1* est un type de données de chaîne.
- NULL si *value1* est un type de données Date/Heure.

Par exemple, l'expression suivante n'inclut pas de condition FALSE et *value1* est un type de données de chaîne de sorte que le Service d'intégration de données renvoie une chaîne vide pour chaque ligne qui renvoie FALSE :

```
IIF( SALES > 100, EMP_NAME )
```

SALES	EMP_NAME	RETURN VALUE
150	John Smith	John Smith
50	Pierre Bleu	' ' ( <i>empty string</i> )
120	Sally Green	Sally Green
NULL	Greg Jones	' ' ( <i>empty string</i> )

## Valeur de retour

*value1* si la condition est TRUE.

*value2* si la condition est FALSE.

Par exemple, l'expression suivante inclut la condition FALSE NULL de sorte que le Service d'intégration de données renvoie NULL pour chaque ligne qui renvoie FALSE :

```
IIF( SALES > 100, EMP_NAME, NULL )
```

SALES	EMP_NAME	RETURN VALUE
150	John Smith	John Smith

SALES	EMP_NAME	RETURN VALUE
50	Pierre Bleu	NULL
120	Sally Green	Sally Green
NULL	Greg Jones	NULL

Si les données contiennent des caractères multioctets et que l'argument de condition compare les données de chaîne, la valeur de retour dépendra de la page de code et du mode mouvement de données du Service d'intégration de données.

## IIF et types de données

Lorsque vous utilisez IIF, le type de données de la valeur de retour est identique à celui du résultat dont la précision est la plus élevée.

Par exemple, prenons l'expression suivante :

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

Le résultat TRUE (1) est un nombre entier et le résultat FALSE (.3333) est un nombre décimal. La précision du type de données décimal est plus élevée que celle du type nombre entier, le type de données de la valeur de retour est donc toujours décimal.

Lorsque vous exécutez un mappage en mode précision élevée et qu'au moins un résultat est de type double, le type de données de la valeur de retour sera double.

## Utilisations spéciales de IIF

Utilisez des instructions IIF imbriquées pour tester plusieurs conditions. L'exemple suivant teste différentes conditions et renvoie 0 si la valeur des ventes est de 0 ou négative :

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

Vous pouvez rendre cette logique plus accessible en lecture en ajoutant des commentaires :

```
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)
```

Utilisez IIF dans des stratégies de mise à jour. Par exemple :

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

### Alternative à l'IIF

Utilisez "DECODE" à la page 77 au lieu d'IIF dans de nombreux cas. DECODE peut améliorer la lisibilité. Voici un exemple d'utilisation de DECODE au lieu d'IIF à l'aide du premier exemple de la section précédente :

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

La plupart du temps, vous pouvez utiliser une transformation filtre au lieu d'IIF pour optimiser des performances.

## IN

Les correspondances entrent des données dans une liste de valeurs. Par défaut, la correspondance est sensible à la casse.

### Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>valueToSearch</i>	Obligatoire	Peut être une chaîne, une date ou une valeur numérique. La valeur d'entrée que vous voulez faire correspondre avec une liste de valeurs séparées par des virgules.
<i>valeur</i>	Obligatoire	Peut être une chaîne, une date ou une valeur numérique. Liste de valeurs, séparées par des virgules, que vous voulez rechercher. Les valeurs peuvent être des ports d'une transformation. Il n'existe pas de nombre maximum de valeurs que vous pouvez répertorier.
<i>CaseFlag</i>	Facultatif	Doit être un nombre entier. Détermine si les arguments de cette fonction sont sensibles à la casse. Vous pouvez entrer l'expression de transformation valide de votre choix. Si <i>CaseFlag</i> est un nombre différent de 0, la fonction est sensible à la casse. Si <i>CaseFlag</i> est une valeur nulle ou 0, la fonction n'est pas sensible à la casse.

### Valeur de retour

TRUE (1) si la valeur d'entrée correspond à la liste de valeurs.

FALSE (0) si la valeur d'entrée ne correspond pas à la liste de valeurs.

NULL si l'entrée est une valeur nulle.

## Exemple

L'expression suivante détermine si la valeur d'entrée est un couteau de sécurité, un couteau à pointe biseautée, ou un couteau en titane de taille moyenne. Il n'est pas nécessaire que la casse des valeurs d'entrée corresponde à celle des valeurs dans la liste séparée par des virgules :

```
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

Trouve l'index d'une valeur parmi la liste de valeurs. Par défaut, la correspondance est sensible à la casse.

## Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>valueToSearch</i>	Obligatoire	Type de données Chaîne. Valeur que vous voulez rechercher dans la liste de chaînes.
<i>chaîne</i>	Obligatoire	Type de données Chaîne. Liste de valeurs, séparées par des virgules, avec laquelle vous voulez effectuer la correspondance. Les valeurs peuvent être au format de chaîne. Il n'existe pas de nombre maximum de valeurs que vous pouvez répertorier. La valeur est sensible à la casse, excepté si vous définissez <i>CaseFlag</i> sur 0.
<i>CaseFlag</i>	Facultatif	Doit être un nombre entier. Détermine si les arguments de cette fonction sont sensibles à la casse. Vous pouvez entrer l'expression de transformation valide de votre choix. Si <i>CaseFlag</i> est différent de 0 ou n'est pas spécifié, la fonction est sensible à la casse. Si <i>CaseFlag</i> est une valeur nulle ou 0, la fonction n'est pas sensible à la casse.

## Valeur de retour

1 si la valeur d'entrée correspond à *string1*, 2 si la valeur d'entrée correspond à *string2*, etc.

0 si la valeur d'entrée est introuvable.

NULL si l'entrée est une valeur nulle.

## Exemple

L'expression suivante détermine si les valeurs provenant du port ITEM\_NAME correspondent à la première, seconde, ou troisième chaîne :

```
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 (Couteau de sécurité) renvoie une valeur de 0, car cela ne correspond pas à la casse de la valeur d'entrée.

# INITCAP

Met en majuscule la première lettre de chaque mot d'une chaîne et convertit toutes les autres lettres en minuscule. Les mots sont délimités par un espace blanc (espace vide, saut de ligne, nouvelle ligne, retour chariot, tabulation, ou tabulation verticale) et des caractères non alphanumériques. Par exemple, si vous passez la chaîne «... THOMAS », la fonction renvoie Thomas.

## Syntaxe

```
INITCAP( string )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>string</i>	Requis	Tout type de données excepté Binaire. Vous pouvez entrer l'expression de transformation valide de votre choix.

## Valeur de retour

Chaîne. Si les données contiennent des caractères multioctets, la valeur de retour dépend de la page de code et du mode mouvement de données du Service d'intégration de données.

NULL si une valeur transmise à la fonction est NULL.

## Exemple

L'expression suivante met en majuscule tous les noms dans le port FIRST\_NAME :

```
INITCAP( FIRST_NAME )
```

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

## INSTR

Renvoie la position d'un jeu de caractères défini dans une chaîne, en comptant de gauche à droite.

### Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire/ Facultatif	Description
<i>chaîne</i>	Obligatoire	Il doit s'agir d'une chaîne de caractères. Transmet la valeur que vous voulez évaluer. Vous pouvez entrer l'expression de transformation valide de votre choix. Les résultats de l'expression doivent être une chaîne de caractères. Dans le cas contraire, INSTR convertit la valeur en chaîne avant de l'évaluer.
<i>search_value</i>	Obligatoire	Aucune valeur. La valeur de recherche est sensible à la casse. Jeu de caractères que vous voulez rechercher. La valeur search_value doit correspondre à une partie de la chaîne. Par exemple, si vous écrivez INSTR('Alfred Pope', 'Alfred Smith'), la fonction renvoie 0.  Vous pouvez entrer l'expression de transformation valide de votre choix. Si vous voulez rechercher une chaîne de caractères, placez les caractères de recherche entre guillemets simples, par exemple 'abc'.

Argument	Obligatoire/ Facultatif	Description
<i>début</i>	Facultatif	<p>Doit être une valeur entière. Position dans la chaîne à partir de laquelle vous voulez démarrer la recherche. Vous pouvez entrer l'expression de transformation valide de votre choix.</p> <p>La valeur par défaut est 1 et signifie que INSTR démarre la recherche au premier caractère dans la chaîne.</p> <p>Si la position de départ est 0, INSTR effectue la recherche à partir du premier caractère dans la chaîne. Si la position de départ est un nombre positif, INSTR recherche la position de départ en comptant à partir du début de la chaîne. Si la position de départ est un nombre négatif, INSTR recherche la position de départ en comptant à partir de la fin de la chaîne. Si vous omettez cet argument, la fonction utilise la valeur par défaut 1.</p>
<i>occurrence</i>	Facultatif	<p>Nombre entier positif supérieur à 0. Vous pouvez entrer l'expression de transformation valide de votre choix. Si la valeur de recherche apparaît plusieurs fois dans la chaîne, vous pouvez spécifier l'occurrence que vous voulez rechercher. Par exemple, entrez 2 pour rechercher la deuxième occurrence à partir de la position de départ.</p> <p>Si vous omettez cet argument, la fonction utilise la valeur par défaut 1 ; en d'autres termes, INSTR recherche la première occurrence de la valeur de recherche. Si vous transmettez une valeur décimale, le Service d'intégration de données l'arrondit à la valeur entière la plus proche. Si vous transmettez un nombre entier négatif ou 0, la session échoue.</p>
<i>comparison_type</i>	Facultatif	<p>Type de comparaison de chaîne, linguistique ou binaire, lorsque le Service d'intégration de données est exécuté en mode Unicode. Lorsque le Service d'intégration de données est exécuté en mode ASCII, le type de comparaison est toujours binaire.</p> <p>Les comparaisons linguistiques tiennent compte des règles de collation spécifiques à la langue, tandis que les comparaisons binaires effectuent une correspondance de bits. Par exemple, le caractère allemand dur s correspond à la chaîne « ss » dans une comparaison linguistique, mais non dans une comparaison binaire. Les comparaisons binaires s'exécutent plus rapidement que les comparaisons linguistiques.</p> <p>Doit être une valeur entière, 0 ou 1 :</p> <ul style="list-style-type: none"> <li>- 0: INSTR effectue une comparaison linguistique de chaîne.</li> <li>- 1: INSTR effectue une comparaison binaire de chaîne.</li> </ul> <p>Par défaut 0.</p>

### Valeur de retour

Nombre entier si la recherche est réussie. Un nombre entier représente la position du premier caractère dans la *search\_value*, en comptant de gauche à droite.

0 si la recherche échoue.

NULL si une valeur transmise à la fonction est NULL.

## Exemples

L'expression suivante renvoie la position de la première occurrence de la lettre « a », en commençant au début de chaque nom de société. L'argument *search\_value* étant sensible à la casse, il ignore les « A » dans « Blue Fin Aqua Center » et renvoie la position du « 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

L'expression suivante renvoie la position de la deuxième occurrence de la lettre « a », en partant du début de chaque nom de société. L'argument *search\_value* étant sensible à la casse, il ignore les « A » dans « Blue Fin Aqua Center » et renvoie 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

L'expression suivante renvoie la position de la deuxième occurrence de la lettre « a » dans chaque nom de société, en commençant par le dernier caractère du nom de la société. L'argument *search\_value* étant sensible à la casse, il ignore les « A » dans « Blue Fin Aqua Center » et renvoie 0 :

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

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

L'expression suivante renvoie la position du premier caractère dans la chaîne « Blue Fin Aqua Center », en commençant par le dernier caractère du nom de la société :

```
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

### Utilisation de la fonction INSTR imbriquée

Vous pouvez imbriquer la fonction INSTR dans d'autres fonctions pour réaliser des tâches plus complexes.

L'expression suivante évalue une chaîne, en commençant par la fin de cette chaîne. L'expression recherche le dernier espace (le plus à droite) dans la chaîne et renvoie tous les caractères situés à sa gauche :

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

CUST_NAME	RETURN VALUE
PATRICIA JONES	PATRICIA
MARY ELLEN SHAH	MARY ELLEN

L'expression suivante supprime le caractère « # » d'une chaîne :

```
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

Renvoie une valeur indiquant si une valeur est NULL. ISNULL évalue une chaîne vide comme FALSE.

**Remarque:** Pour tester des chaînes vides, utilisez LENGTH.

### Syntaxe

```
ISNULL( value )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>valeur</i>	Obligatoire	Tout type de données excepté Binaire. Transmet les lignes à évaluer. Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

TRUE (1) si la valeur est nulle.

FALSE (0) si la valeur n'est pas nulle.

### Exemple

Dans l'exemple suivant, les valeurs nulles sont recherchées dans la table d'éléments :

```
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

Indique si une valeur de chaîne est une date valide. Une date valide est n'importe quelle chaîne dans la partie de date au format date/heure spécifié dans la session. Si la chaîne que vous voulez tester n'est pas au format de date, utilisez la chaîne de format TO\_DATE pour spécifier le format de date. Si les chaînes transmises à IS\_DATE ne correspondent pas à la chaîne de format spécifiée, la fonction renvoie FALSE (0). Si les chaînes correspondent à la chaîne de format, la fonction renvoie TRUE (1).

IS\_DATE évalue les chaînes et renvoie une valeur entière.

Le port de sortie d'une expression IS\_DATE doit être un type de données chaîne ou numérique.

Vous pouvez utiliser IS\_DATE pour tester ou filtrer les données dans un fichier plat avant de les écrire dans une cible.

Utilisez la chaîne de format RR avec IS\_DATE au lieu de la chaîne de format YY. Dans la plupart des cas, les deux chaînes de format renvoient les mêmes valeurs, mais il existe certains cas exceptionnels où YY renvoie des résultats incorrects. Par exemple, l'expression IS\_DATE('02/29/00', 'YY') est calculée en interne comme IS\_DATE(02/29/1900 00:00:00), qui renvoie false. Cependant, le Service d'intégration de données calcule l'expression IS\_DATE('02/29/00', 'RR') comme IS\_DATE(02/29/2000 00:00:00), qui renvoie TRUE. Dans le premier cas, l'année 1900 n'est pas bissextile, il n'existe donc pas de 29 février.

**Remarque:** IS\_DATE utilise les mêmes chaînes de format que TO\_DATE.

## Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>valeur</i>	Obligatoire	Doit être un type de données String. Transmet les lignes à évaluer. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>format</i>	Facultatif	Entrez une chaîne de format TO_DATE valide. La chaîne de format doit correspondre aux parties de l'argument <i>chaîne</i> . Par exemple, si vous passez la chaîne 'Mar 15 1997 12:43:10AM', vous devez utiliser la chaîne de format 'MON DD YYYY HH12:MI:SSAM'. Si vous omettez la chaîne de format, la valeur de chaîne doit être au format de date spécifié dans la configuration du mappage.

## Valeur de retour

TRUE (1) si la ligne est une date valide.

FALSE (0) si la ligne n'est pas une date valide.

NULL si une valeur dans l'expression est nulle ou si la chaîne de format est nulle.

**Avertissement:** Le format de la chaîne IS\_DATE doit correspondre à la chaîne de format, y compris les séparateurs de date. Si ce n'est pas le cas, le Service d'intégration de données peut renvoyer des valeurs non précises ou ignorer l'enregistrement.

## Exemples

L'expression suivante recherche les dates valides pour le port INVOICE\_DATE :

```
IS_DATE( INVOICE_DATE )
```

Cette expression renvoie des données similaires aux suivantes :

INVOICE_DATE	RETURN VALUE
NULL	NULL
'180'	0 (FALSE)
'04/01/98'	0 (FALSE)
'04/01/1998 00:12:15.7008'	1 (TRUE)
'02/31/1998 12:13:55.9204'	0 (FALSE) ( <i>February does not have 31 days</i> )
'John Smith'	0 (FALSE)

L'expression IS\_DATE suivante spécifie une chaîne de format « YYYY/MM/DD » :

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

Si la valeur de chaîne ne correspond pas à ce format, IS\_DATE renvoie FALSE :

INVOICE_DATE	RETURN VALUE
NULL	NULL
'180'	0 (FALSE)
'04/01/98'	0 (FALSE)
'1998/01/12'	1 (TRUE)
'1998/11/21 00:00:13'	0 (FALSE)
'1998/02/31'	0 (FALSE) (February does not have 31 days)
'John Smith'	0 (FALSE)

L'exemple suivant illustre l'utilisation de IS\_DATE pour tester des données avant d'utiliser TO\_DATE pour convertir les chaînes en dates. Cette expression vérifie les valeurs dans le port INVOICE\_DATE et convertit chaque date valide en valeur de date. Si la valeur n'est pas une date valide, le Service d'intégration de données renvoie ERROR et ignore la ligne.

Cet exemple renvoie une valeur Date/Heure. Par conséquent, le port de sortie pour l'expression doit être Date/Heure :

```
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

Indique si une chaîne est un nombre valide. Un nombre valide est composé des parties suivantes :

- Espace facultatif avant le nombre
- Signe (+/-) facultatif
- Un ou plusieurs chiffres avec un point décimal facultatif

- Notation scientifique facultative, comme la lettre « e » ou « E » (et la lettre « d » ou « D » sous Windows) suivie d'un signe (+/-) facultatif, suivi d'un ou plusieurs chiffres
- Espace facultatif suivant le nombre

Les nombres suivants sont tous valides :

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

Le port de sortie pour une expression IS\_NUMBER doit être un type de données de chaîne ou numérique.

Vous pouvez utiliser IS\_NUMBER pour tester ou filtrer les données dans un fichier simple avant d'écrire dans une cible.

## Syntaxe

```
IS_NUMBER( value )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
valueur	Obligatoire	Doit être un type de données String. Transmet les lignes à évaluer. Vous pouvez entrer l'expression de transformation valide de votre choix.

## Valeur de retour

TRUE (1) si la ligne est un nombre valide.

FALSE (0) si la ligne n'est pas un nombre valide.

NULL si une valeur dans l'expression est nulle.

## Exemples

L'expression suivante vérifie les nombres valides du port ITEM\_PRICE :

```
IS_NUMBER( ITEM_PRICE )
```

ITEM_PRICE	RETURN VALUE
'123.00'	1 (True)
'-3.45e+3'	1 (True)
'-3.45D-3'	1 (True - Windows only)
'-3.45d-3'	0 (False - UNIX only)
'3.45E-'	0 (False) <i>Incomplete number</i>
' '	0 (False) <i>Consists entirely of blanks</i>
''	0 (False) <i>Empty string</i>

ITEM_PRICE	RETURN VALUE
'+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

Utilisez IS\_NUMBER pour tester des données avant d'utiliser l'une des fonctions de conversion numérique, telles que TO\_FLOAT. Par exemple, l'expression suivante vérifie les valeurs dans le port ITEM\_PRICE et convertit chaque nombre valide en une valeur à virgule flottante double précision. Si la valeur n'est pas un nombre valide, le Service d'intégration de données renvoie 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

Indique si une valeur de chaîne est constituée uniquement d'espaces. Il peut s'agir d'un espace vide, un saut de ligne, une nouvelle ligne, un retour chariot, une tabulation ou une tabulation verticale.

IS\_SPACES évalue une chaîne vide comme FALSE, car il n'existe pas d'espace. Pour tester une chaîne vide, utilisez LENGTH.

### Syntaxe

```
IS_SPACES( value )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>valeur</i>	Obligatoire	Doit être un type de données String. Transmet les lignes à évaluer. Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

TRUE (1) si la ligne est constituée uniquement d'espaces.

FALSE (0) si la ligne contient des données.

NULL si une valeur dans l'expression est nulle.

### Exemple

L'expression suivante recherche des lignes constituées uniquement d'espaces dans le port ITEM\_NAME :

```
IS_SPACES( ITEM_NAME )
```

ITEM_NAME	RETURN VALUE
Flashlight	0 (False)
	1 (True)
Regulator system	0 (False)
NULL	NULL
' '	0 (FALSE) ( <i>Empty string does not contain spaces.</i> )

**Astuce:** Utilisez IS\_SPACES pour éviter l'écriture d'espaces dans une colonne de caractères dans une table cible. Par exemple, si vous disposez d'une transformation qui écrit des noms de clients dans une colonne CHAR(5) de longueur fixe dans une table cible, vous pouvez écrire « 00000 » au lieu d'espaces. Vous devrez alors créer une expression comme suit :

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

## LAST

Renvoie la dernière ligne dans le port sélectionné. Vous pouvez également appliquer un filtre pour limiter les lignes lues par le Service d'intégration de données. Vous pouvez imbriquer une seule autre fonction Agrégation dans la fonction LAST.

### Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>valeur</i>	Obligatoire	Tout type de données excepté Binaire. Transmet les valeurs pour lesquelles vous voulez renvoyer la dernière ligne. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>filter_condition</i>	Facultatif	Limite les lignes dans la recherche. La condition de filtre doit être une valeur numérique ou renvoyer TRUE, FALSE ou NULL. Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

Dernière ligne dans un port.

NULL si toutes les valeurs passées à la fonction sont NULL, ou si aucune ligne n'est sélectionnée (par exemple, la condition de filtre renvoie FALSE ou NULL pour toutes les lignes).

### Exemple

L'expression suivante renvoie la dernière ligne dans le port ITEMS\_NAME dont le montant est supérieur à 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

Renvoie la date du dernier jour du mois pour chaque date dans un port.

### Syntaxe

```
LAST_DAY( date )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>date</i>	Obligatoire	Type de données Date/Heure. Transmet les dates pour lesquelles vous voulez renvoyer le dernier jour du mois. Vous pouvez entrer une expression de transformation valide qui renvoie une date.

### Valeur de retour

Date. Dernier jour du mois pour cette valeur de date que vous transmettez à cette fonction.

NULL si une valeur dans le port sélectionné est nulle.

### Null

Si une valeur est nulle, LAST\_DAY ignore la ligne. Cependant, si toutes les valeurs transmises à partir du port sont NULL, LAST\_DAY renvoie NULL.

### Grouper par

LAST\_DAY groupe des valeurs en fonction du groupement par ports que vous définissez dans la transformation et renvoie un résultat pour chaque groupe. S'il n'existe aucun port de regroupement, LAST\_DAY traite toutes les lignes comme un seul groupe et renvoie une valeur.

### Exemples

L'expression suivante renvoie le dernier jour du mois pour chaque date dans le port 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

Vous pouvez imbriquer TO\_DATE pour convertir des valeurs de chaîne en date. TO\_DATE inclut toujours des informations d'heure. Si vous transmettez une chaîne qui ne contient pas de valeur d'heure, la date renvoyée inclura l'heure 00:00:00.

L'exemple suivant renvoie le dernier jour du mois pour chaque date de commande au même format que la chaîne :

```
LAST_DAY( TO_DATE( ORDER_DATE, 'DD-MON-YY' ) )
```

ORDER_DATE	RETURN VALUE
'18-NOV-98'	Nov 30 1998 00:00:00
'28-APR-98'	Apr 30 1998 00:00:00

ORDER_DATE	RETURN VALUE
NULL	NULL
'18-FEB-96'	Feb 29 1996 00:00:00 ( <i>Leap year</i> )

## LEAST

Renvoie la plus petite valeur depuis une liste de valeurs d'entrée. Par défaut, la correspondance est sensible à la casse.

### Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>valeur</i>	Obligatoire	Tout type de données excepté Binaire. Le type de données doit être compatible avec d'autres valeurs. Valeur à comparer à d'autres valeurs. Vous devez entrer au moins un argument de valeur.  Si la valeur est numérique et que d'autres valeurs d'entrée sont d'autres types de données numériques, toutes les valeurs utilisent la précision la plus élevée possible. Par exemple, si des valeurs sont du type de données nombre entier et que d'autres sont de type double, le Service d'intégration de données convertit les valeurs en type double.
<i>CaseFlag</i>	Facultatif	Doit être un nombre entier. Détermine si les arguments de cette fonction sont sensibles à la casse. Vous pouvez entrer l'expression de transformation valide de votre choix.  Si <i>CaseFlag</i> est un nombre différent de 0, la fonction est sensible à la casse. Si <i>CaseFlag</i> est une valeur nulle ou 0, la fonction n'est pas sensible à la casse.

### Valeur de retour

*valeur1* s'il s'agit de la plus faible valeur d'entrée, *valeur2* s'il s'agit de la plus faible valeur d'entrée, etc.

NULL si l'un des arguments est nul.

### Exemple

L'expression suivante renvoie le plus petit nombre d'éléments commandés :

```
LEAST( QUANTITY1, QUANTITY2, QUANTITY3 )
```

QUANTITY1	QUANTITY2	QUANTITY3	RETURN VALUE
150	756	27	27
			NULL

QUANTITY1	QUANTITY2	QUANTITY3	RETURN VALUE
5000	97	17	17
120	1724	965	120

## LONGUEUR

Renvoie le nombre de caractères dans une chaîne, y compris les espaces blancs de fin.

### Syntaxe

```
LENGTH( string )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>chaîne</i>	Obligatoire	Type de données Chaîne. Chaînes à évaluer. Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

Nombre entier représentant la longueur de la chaîne.

NULL si une valeur transmise à la fonction est NULL.

### Exemple

L'expression suivante renvoie la longueur de chaque nom de client :

```
LENGTH( CUSTOMER_NAME )
```

CUSTOMER_NAME	RETURN VALUE
Bernice Davis	13
NULL	NULL
John Baer	9
Greg Brown	10

### Conseils d'utilisation de la fonction LENGTH

Utilisez LENGTH pour tester des conditions de chaîne vide. Si vous voulez rechercher des champs pour lesquels le nom du client n'est pas rempli, utilisez une expression comme suit :

```
IIF( LENGTH( CUSTOMER_NAME ) = 0, 'EMPTY STRING' )
```

Pour tester un champ NULL, utilisez ISNULL. Pour tester des espaces, utilisez IS\_SPACES.

# LN

Renvoie le logarithme népérien d'une valeur numérique. Par exemple, LN(3) renvoie 1.098612. En règle générale, vous utilisez cette fonction pour analyser des données scientifiques au lieu de données métier.

Cette fonction est la réciproque de la fonction EXP.

## Syntaxe

```
LN( numeric_value )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Obligatoire	Type de données numérique. Le nombre doit être positif et supérieur à 0. Transmet les valeurs pour lesquelles vous souhaitez calculer le logarithme népérien. Vous pouvez entrer l'expression de transformation valide de votre choix.

## Valeur de retour

Valeur Double.

NULL si une valeur transmise à la fonction est NULL.

## Exemple

L'expression suivante renvoie le logarithme népérien pour toutes les valeurs dans le port NUMBERS :

```
LN( NUMBERS )
```

NUMBERS	RETURN VALUE
10	2.302585092994
125	4.828313737302
0.96	-0.04082199452026
NULL	NULL
-90	<i>Error. (The Integration Service does not write row.)</i>
0	<i>Error. (The Integration Service does not write row.)</i>

**Remarque:** Le Service d'intégration de données affiche une erreur et n'écrit pas la ligne si vous transmettez un nombre négatif ou 0. La *numeric\_value* doit être un nombre positif supérieur à 0.

# JOURNAL

Renvoie le logarithme d'une valeur numérique. Plus fréquemment, vous devez utiliser cette fonction pour analyser les données scientifiques au lieu de données métier.

## Syntaxe

```
LOG( base, exponent )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>base</i>	Obligatoire	Base du logarithme. Doit être une valeur numérique positive autre que 0 ou 1. Toute transformation Expression valide qui renvoie un nombre positif autre que 0 ou 1.
<i>exposant</i>	Obligatoire	Exposant du logarithme. Doit être une valeur numérique positive supérieure à 0. Toute transformation Expression valide qui renvoie un nombre positif supérieur à 0.

## Valeur de retour

Valeur Double.

NULL si une valeur transmise à la fonction est NULL.

## Exemple

L'expression suivante renvoie le logarithme pour toutes les valeurs dans le port NUMBERS :

```
LOG( BASE, EXPONENT )
```

BASE	EXPONENT	RETURN VALUE
15	1	0
.09	10	-0.956244644696599
NULL	18	NULL
35.78	NULL	NULL
-9	18	<i>Error. (Service d'intégration de données does not write the row.)</i>
0	5	<i>Error. (Service d'intégration de données does not write the row.)</i>
10	-2	<i>Error. (Service d'intégration de données does not write the row.)</i>

Le Service d'intégration de données affiche une erreur et n'écrit pas la ligne si vous transmettez un nombre négatif, 0 ou 1 comme valeur de base, ou si vous transmettez une valeur négative pour l'exposant.

# LOWER

Convertit les caractères majuscules de la chaîne en minuscule.

## Syntaxe

```
LOWER( string )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>chaîne</i>	Obligatoire	Toute valeur de chaîne. L'argument transmet les valeurs de chaîne que vous voulez renvoyer en minuscule. Vous pouvez entrer une expression de transformation valide qui renvoie une chaîne.

### Valeur de retour

Chaîne de caractères minuscules. Si les données contiennent des caractères multioctets, la valeur de retour dépend de la page de code et du mode mouvement de données du service d'intégration.

NULL si une valeur dans le port sélectionné est nulle.

### Exemple

L'expression suivante renvoie tous les prénoms en minuscule :

```
LOWER( FIRST_NAME )
```

FIRST_NAME	RETURN VALUE
antonia	antonia
NULL	NULL
THOMAS	thomas
PierRe	pierre
BERNICE	bernice

## LPAD

Ajoute un ensemble d'espaces ou de caractères au début d'une chaîne pour la définir sur une longueur spécifique.

### Syntaxe

```
LPAD( first_string, length [,second_string] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>first_string</i>	Requis	Peut être une chaîne de caractères. Chaînes à modifier. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>length</i>	Requis	Doit être un littéral entier positif. Cet argument spécifie la longueur que vous voulez appliquer à chaque chaîne.
<i>second_string</i>	Facultatif	Peut être une valeur de chaîne. Caractères que vous souhaitez ajouter à gauche des valeurs <i>first_string</i> . Vous pouvez entrer l'expression de transformation valide de votre choix. Vous pouvez entrer un littéral de chaîne spécifique. Cependant, placez les caractères à ajouter en début de chaîne entre guillemets simples, par exemple : 'abc'. Cet argument est sensible à la casse. Si vous omettez <i>second_string</i> , la fonction ajoute des espaces vides au début de la première chaîne.

### Valeur de retour

Chaîne de la longueur spécifiée.

NULL si une valeur transmise à la fonction est NULL ou si la *longueur* est un nombre négatif.

### Exemples

L'expression suivante normalise les nombres à six chiffres en les faisant précéder de zéros :

```
LPAD( PART_NUM, 6, '0')
```

PART_NUM	RETURN VALUE
702	000702
1	000001
0553	000553
484834	484834

LPAD compte la longueur de gauche à droite. Si la première chaîne est supérieure à la longueur, LPAD tronque la chaîne de droite à gauche. Par exemple, LPAD('alphabetical', 5, 'x') renvoie la chaîne 'alpha'.

Si la deuxième chaîne est plus longue que le nombre total de caractères requis pour renvoyer la longueur spécifiée, LPAD utilise une partie de la deuxième chaîne :

```
LPAD( ITEM_NAME, 16, '*..*' )
```

ITEM_NAME	RETURN VALUE
Flashlight	*..**.Flashlight
Compass	*..**..**Compass
Regulator System	Regulator System
Safety Knife	*..*Safety Knife

# LTRIM

Supprime les espaces ou les caractères du début d'une chaîne. Vous pouvez utiliser LTRIM avec les fonctions IIF ou DECODE dans une transformation Expression ou Stratégie de mise à jour pour éviter les espaces dans une table cible.

Si vous ne spécifiez aucun paramètre *trim\_set* dans l'expression :

- En mode Unicode, LTRIM supprime les espaces à octet simple et à deux octets au début d'une chaîne.
- En mode ASCII, LTRIM supprime seulement les espaces à octet simple.

Si vous utilisez LTRIM pour supprimer des caractères d'une chaîne, LTRIM compare *trim\_set* à chaque caractère dans l'argument de *chaîne* l'un après l'autre, en commençant du côté gauche de la chaîne. Si le caractère dans la chaîne correspond à caractère dans *trim\_set*, LTRIM le supprime. LTRIM continue de comparer et de supprimer des caractères tant qu'il trouve des caractères correspondants dans *trim\_set*. Puis, il renvoie la chaîne qui n'inclut aucun caractère correspondant.

## Syntaxe

```
LTRIM( string [, trim_set] )
```

Le tableau suivant décrit les arguments de cette commande:

Arguments	Obligatoire / Facultatif	Description
<i>chaîne</i>	Obligatoire	Toute valeur de chaîne. Transmet les chaînes que vous souhaitez modifier. Vous pouvez entrer l'expression de transformation valide de votre choix. Utilisez des opérateurs pour effectuer des comparaisons ou concaténer des chaînes avant de supprimer de caractères au début d'une chaîne.
<i>trim_set</i>	Facultatif	Toute valeur de chaîne. Transmet les caractères que vous voulez supprimer au début de la première chaîne. Vous pouvez entrer l'expression de transformation valide de votre choix. Vous pouvez également saisir une chaîne de caractères. Cependant, vous devez placer les caractères que vous voulez supprimer au début de la chaîne entre guillemets simples, par exemple : 'abc'. Si vous omettez la deuxième chaîne, la fonction supprimera tous les espaces au début de la chaîne.  LTRIM est sensible à la casse. Par exemple, si vous voulez supprimer le caractère « A » de la chaîne « Alfredo », saisissez « A », non « a ».

## Valeur de retour

Chaîne. Valeurs de chaîne dont les caractères spécifiés dans l'argument *trim\_set* ont été retirés.

NULL si une valeur transmise à la fonction est NULL. Si *trim\_set* est NULL, la fonction renvoie NULL.

## Exemple

L'expression suivante supprime les caractères « S » et « . » des chaînes dans le port LAST\_NAME :

```
LTRIM( LAST_NAME, 'S.')
```

LAST_NAME	RETURN VALUE
Nelson	Nelson
Osborne	Osborne

LAST_NAME	RETURN VALUE
NULL	NULL
S. MacDonald	MacDonald
Sawyer	awyer
H. Bender	H. Bender
Steadman	teadman

LTRIM supprime « S. » dans S. MacDonald et « S » dans Sawyer et Steadman, mais non le point dans H. Bender. Cela est dû au fait que LTRIM recherche, un caractère après l'autre, dans le jeu de caractères que vous spécifiez dans l'argument *trim\_set*. Si le premier caractère dans la chaîne correspond au premier caractère dans *trim\_set*, LTRIM le supprime. Puis, LTRIM consulte le deuxième caractère dans la chaîne. Si elle correspond au deuxième caractère dans *trim\_set*, LTRIM le supprime et ainsi de suite. Lorsque le premier caractère dans la chaîne ne correspond pas au caractère *trim\_set*, LTRIM renvoie la chaîne et évalue la ligne suivante.

Dans l'exemple de H. Bender, H ne correspond pas au caractère dans l'argument *trim\_set* ; LTRIM renvoie donc la chaîne dans le port LAST\_NAME et passe à la ligne suivante.

### Conseils d'utilisation de la fonction LTRIM

Utilisez RTRIM et LTRIM avec || ou CONCAT pour supprimer les espaces de début et de fin après la concaténation de deux chaînes.

Vous pouvez également supprimer plusieurs jeux de caractères en imbriquant LTRIM. Par exemple, si vous voulez retirer les espaces au début et le caractère 'T' d'une colonne de noms, vous pouvez créer une expression comme suit :

```
LTRIM( LTRIM( NAMES ), 'T' )
```

## MAKE\_DATE\_TIME

Renvoie la date et l'heure à partir des valeurs d'entrée.

### Syntaxe

```
MAKE_DATE_TIME( year, month, day, hour, minute, second, nanosecond )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>année</i>	Obligatoire	Type de données numérique. Nombre entier positif de 4 chiffres. Si vous passez à cette fonction une année à 2 chiffres, le Service d'intégration de données renvoie « 00 » pour les deux premiers chiffres de l'année.
<i>mois</i>	Obligatoire	Type de données numérique. Nombre entier positif entre 1 et 12 (janvier = 1 et décembre = 12).
<i>jour</i>	Obligatoire	Type de données numérique. Nombre entier positif entre 1 et 31 (excepté pour les mois inférieurs à 31 jours : Février, avril, juin, septembre et novembre).
<i>heure</i>	Facultatif	Type de données numérique. Nombre entier positif compris entre 0 et 24 (où 0 = minuit, 12 = midi et 24 = minuit).
<i>minute</i>	Facultatif	Type de données numérique. Nombre entier positif compris entre 0 et 59.
<i>seconde</i>	Facultatif	Type de données numérique. Nombre entier positif compris entre 0 et 59.
<i>nanoseconde</i>	Facultatif	Type de données numérique. Nombre entier positif compris entre 0 et 999,999,999.

### Valeur de retour

Date sous la forme : MM/DD/YYYY HH24:MI:SS. Renvoie une valeur nulle si vous ne passez une année, un mois, ou un jour à la fonction.

### Exemple

L'expression suivante crée une date et une heure à partir des ports d'entrée :

```
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 (Dates)

Renvoie la dernière date détectée dans un port ou un groupe. Vous pouvez appliquer un filtre pour limiter les lignes dans la recherche. Vous ne pouvez imbriquer qu'une seule autre fonction Agrégation dans MAX.

Vous pouvez également utiliser MAX pour renvoyer la plus grande valeur numérique ou la valeur de chaîne la plus élevée dans un port ou un groupe.

## Syntaxe

```
MAX( date [, filter_condition] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>date</i>	Obligatoire	Type de données Date/Heure. Transmet la date pour laquelle vous voulez renvoyer une date maximum. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>filter_condition</i>	Facultatif	Limite les lignes dans la recherche. La condition de filtre doit être une valeur numérique ou renvoyer TRUE, FALSE ou NULL. Vous pouvez entrer l'expression de transformation valide de votre choix.

## Valeur de retour

Date.

NULL si toutes les valeurs passées à la fonction sont NULL, ou si aucune ligne n'est sélectionnée (par exemple, la condition de filtre renvoie FALSE ou NULL pour toutes les lignes).

## Exemple

Vous pouvez renvoyer la date maximum pour un port ou un groupe. L'expression suivante renvoie la date maximum de commande de lampes de poche :

```
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 (Nombres)

Renvoie la valeur numérique maximale détectée dans un port ou un groupe. Vous pouvez appliquer un filtre pour limiter les lignes dans la recherche. Vous ne pouvez imbriquer qu'une seule autre fonction Agrégation dans MAX. Vous pouvez également utiliser MAX pour renvoyer la dernière date ou la valeur de chaîne la plus élevée dans un port ou un groupe.

## Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Obligatoire	Type de données numérique. Transmet les valeurs numériques pour lesquelles vous voulez renvoyer une valeur numérique maximum. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>filter_condition</i>	Facultatif	Limite les lignes dans la recherche. La condition de filtre doit être une valeur numérique ou renvoyer TRUE, FALSE ou NULL. Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

Valeur numérique.

NULL si toutes les valeurs transmises à la fonction sont NULL, ou si aucune ligne n'est sélectionnée (par exemple : la condition de filtre renvoie FALSE ou NULL pour toutes les lignes).

### Valeurs nulles

Si une valeur est nulle, MAX l'ignore. Cependant, si toutes les valeurs transmises à partir du port sont NULL, MAX renvoie NULL.

### Grouper par

MAX groupe des valeurs en fonction des ports de regroupement que vous définissez dans la transformation et renvoie un seul résultat pour chaque groupe.

S'il n'existe aucun port de regroupement, MAX traite toutes les lignes comme un groupe et renvoie une seule valeur.

### Exemple

La première expression renvoie le prix maximum des lampes de poche :

```
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 (Chaîne)

Renvoie la valeur de type chaîne la plus élevée dans un port ou un groupe. Vous pouvez appliquer un filtre pour limiter les lignes dans la recherche. Vous ne pouvez imbriquer qu'une seule autre fonction Agrégation dans MAX.

**Remarque:** La fonction MAX utilise le même ordre de tri que celui de la transformation Trieur. Cependant, la fonction MAX est sensible à la casse et la transformation Trieur peut ne pas l'être.

Vous pouvez également utiliser MAX pour renvoyer la dernière date ou la plus grande valeur numérique dans un port ou un groupe.

## Syntaxe

```
MAX( string [, filter_condition] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>chaîne</i>	Obligatoire	Type de données Chaîne. Transmet les valeurs de chaîne pour lesquelles vous voulez renvoyer une valeur de chaîne maximum. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>filter_condition</i>	Facultatif	Limite les lignes dans la recherche. La condition de filtre doit être une valeur numérique ou renvoyer TRUE, FALSE ou NULL. Vous pouvez entrer l'expression de transformation valide de votre choix.

## Valeur de retour

Chaîne.

NULL si toutes les valeurs passées à la fonction sont NULL, ou si aucune ligne n'est sélectionnée (par exemple, la condition de filtre renvoie FALSE ou NULL pour toutes les lignes).

## Valeurs nulles

Si une valeur est nulle, MAX l'ignore. Cependant, si toutes les valeurs transmises à partir du port sont NULL, MAX renvoie NULL.

## Grouper par

MAX groupe des valeurs en fonction des ports de regroupement que vous définissez dans la transformation et renvoie un seul résultat pour chaque groupe.

S'il n'existe aucun port de regroupement, MAX traite toutes les lignes comme un groupe et renvoie une seule valeur.

## Exemple

L'expression suivante renvoie le nom de l'élément maximum pour le fabricant ID 104 :

```
MAX( ITEM_NAME, MANUFACTURER_ID='104' )
```

MANUFACTURER_ID	ITEM_NAME
101	First Stage Regulator

MANUFACTURER_ID	ITEM_NAME
102	Electronic Console
104	Flashlight
104	Battery (9 volt)
104	Rope (20 ft)
104	60.6 cu ft Tank
107	75.4 cu ft Tank
108	Wristband Thermometer

**RETURN VALUE:** Rope (20 ft)

## MD5

Calcule la somme de contrôle de la valeur d'entrée. La fonction utilise l'algorithme Message-Digest 5 (MD5). MD5 est une fonction de hachage cryptographique unilatérale avec une valeur de hachage de 128 bits. Vous pouvez conclure que les valeurs d'entrée sont différentes lorsque les sommes de contrôles des valeurs d'entrée sont différentes. Utilisez MD5 pour vérifier l'intégrité des données.

### Syntaxe

MD5( *value* )

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>value</i>	Obligatoire	Type de données Chaîne ou Binaire. Valeur pour laquelle vous souhaitez calculer la somme de contrôle. La casse de la valeur d'entrée affecte la valeur de retour. Par exemple, MD5(informatica) et MD5(Informatica) renvoient des valeurs différentes.

### Valeur de retour

Chaîne unique de 32 caractères de chiffres hexadécimaux 0-9 et a-f.

NULL si l'entrée est une valeur nulle.

### Exemple

Vous voulez écrire les données modifiées dans une base de données. Utilisez MD5 pour générer des valeurs de somme de contrôle pour les lignes de données lues à partir d'une source. Lorsque vous exécutez un mappage, comparez les valeurs de somme de contrôle précédemment générées aux nouvelles valeurs de somme de contrôle. Puis, écrivez les lignes avec des valeurs de somme de contrôle mises à jour dans la

cible. Vous pouvez conclure qu'une valeur de somme de contrôle mise à jour indique que les données ont été modifiées.

### Conseil

Vous pouvez utiliser la valeur de retour comme clé de hachage.

## MEDIAN

Revoit la médiane de toutes les valeurs dans un port sélectionné.

S'il existe un nombre pair de valeurs dans le port, la médiane est la moyenne des deux valeurs centrales lorsque toutes les valeurs sont placées de manière : dans une ligne de nombres. S'il existe un nombre impair de valeurs dans le port, la médiane est le nombre central.

Vous ne pouvez imbriquer qu'une seule autre fonction Agrégation à la fonction MEDIAN et la fonction imbriquée doit renvoyer un type de données numérique.

Le Service d'intégration de données lit toutes les lignes de données pour effectuer le calcul de la médiane. Le processus de lecture des lignes de données pour effectuer le calcul peut affecter les performances. Vous pouvez également appliquer un filtre pour limiter les lignes que vous lisez pour calculer la médiane.

### Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Obligatoire	Type de données numérique. Transmet les valeurs pour lesquelles vous souhaitez calculer une médiane. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>filter_condition</i>	Facultatif	Limite les lignes dans la recherche. La condition de filtre doit être une valeur numérique ou renvoyer TRUE, FALSE ou NULL. Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

Valeur numérique.

NULL si toutes les valeurs transmises à la fonction sont NULL, ou si aucune ligne n'est sélectionnée. Par exemple, la condition de filtre renvoie FALSE ou NULL pour toutes les lignes.

**Remarque:** Si la valeur de retour est décimale avec une précision supérieure à 15, vous pouvez activer la précision élevée pour garantir une précision décimale jusqu'à 28 chiffres.

### Valeurs nulles

Si une valeur est nulle, MEDIAN ignore la ligne. Cependant, si toutes les valeurs transmises à partir du port sont NULL, MEDIAN renvoie NULL.

## Grouper par

MEDIAN groupe des valeurs en fonction des ports de regroupement que vous définissez dans la transformation et renvoie un résultat pour chaque groupe.

S'il n'existe aucun port de regroupement, MEDIAN traite toutes les lignes comme un seul groupe et renvoie une seule valeur.

## Exemple

Pour calculer le salaire médian de tous les services, créez une transformation Agrégation groupée par service avec un port spécifiant l'expression suivante :

```
MEDIAN( SALARY )
```

L'expression suivante renvoie la valeur médiane des commandes de gilets de stabilisation :

```
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

Encode des valeurs de chaîne. Vous pouvez spécifier la longueur de la chaîne que vous voulez encoder.

METAPHONE encode les caractères de l'alphabet de la langue anglaise (A-Z). Il encode les lettres majuscules et minuscules en majuscule.

METAPHONE encode les caractères selon la liste de règles suivante :

- Ignore les voyelles (A, E, I, O et U), sauf si l'une d'elles est le premier caractère de la chaîne d'entrée. METAPHONE('CAR') renvoie « KR » et METAPHONE('AAR') renvoie « AR ».
- Utilise des instructions de codage spécial.

Le tableau suivant répertorie les instructions du codage METAPHONE :

Entrée	Renvoi	Condition	Exemple
B	- n/a	- lorsqu'il suit M	- METAPHONE ('Lamb') renvoie LM.
B	- B	- dans tous les autres cas	- METAPHONE ('Box') renvoie BKS.
C	- X	- quand suivi de IA ou H	- METAPHONE ('Facial') renvoie FXL.
C	- S	- quand suivi de I, E, ou Y	- METAPHONE ('Fence') renvoie FNS.
C	- n/a	- Lorsqu'il suit S et est suivi de I, E, ou Y	- METAPHONE ('Scene') renvoie SN.
C	- K	- dans tous les autres cas	- METAPHONE ('Cool') renvoie KL.
D	- J	- quand suivi de GE, GY, ou GI	- METAPHONE ('Dodge') renvoie TJ.
D	- T	- dans tous les autres cas	- METAPHONE ('David') renvoie TFT.
F	- F	- dans tous les cas	- METAPHONE ('FOX') renvoie FKS.
G	- F	- quand suivi de H et que le premier caractère dans la chaîne d'entrée n'est pas B, D ou H	- METAPHONE ('Tough') renvoie TF.
G	- n/a	- quand suivi de H et que le premier caractère dans la chaîne d'entrée est B, D ou H	- METAPHONE ('Hugh') renvoie HF.
G	- J	- quand suivi de I, E ou Y et n'est pas répété	- METAPHONE ('Magic') renvoie MJK.
G	- K	- dans tous les autres cas	- METAPHONE('GUN') renvoie KN.
H	- H	- lorsqu'il ne suit pas C, G, P, S ou T et est suivi de A, E, I ou U	- METAPHONE ('DHAT') renvoie THT.
H	- n/a	- dans tous les autres cas	- METAPHONE ('Chain') renvoie XN.
J	- J	- dans tous les cas	- METAPHONE ('Jen') renvoie JN.
K	- n/a - K	- lorsqu'il suit C - dans tous les autres cas	- METAPHONE ('Ckim') renvoie KM. - METAPHONE ('Kim') renvoie KM.
L	- L	- dans tous les cas	- METAPHONE ('Laura') renvoie LR.
M	- M	- dans tous les cas	- METAPHONE ('Maggi') renvoie MK.
N	- N	- dans tous les cas	- METAPHONE ('Nancy') renvoie NNS.
P	- F	- Lorsque suivi de H	- METAPHONE ('Phone') renvoie FN.
P	- P	- dans tous les autres cas	- METAPHONE ('Pip') renvoie PP.
Q	- K	- dans tous les cas	- METAPHONE ('Queen') renvoie KN.
R	- R	- dans tous les cas	- METAPHONE ('Ray') renvoie R.
S	- X	- quand suivi de H, IO, IA ou CHW	- METAPHONE ('Cash') renvoie KX.

Entrée	Renvoi	Condition	Exemple
S	- S	- dans tous les autres cas	- METAPHONE ('Sing') renvoie SNK.
T	- X	- quand suivi de IA ou IO	- METAPHONE ('Patio') renvoie PX.
T	- O <sup>1</sup>	- Lorsque suivi de H	- METAPHONE ('Thor') renvoie OR.
T	- n/a	- quand suivi de CH	- METAPHONE ('Glitch') renvoie KLTX.
T	- T	- dans tous les autres cas	- METAPHINE ('Tim') renvoie TM.
V	- F	- dans tous les cas	- METAPHONE ('Vin') renvoie FN.
W	- W	- quand suivi de A, E, I, O ou U	- METAPHONE ('Wang') renvoie WNK.
W	- n/a	- dans tous les autres cas	- METAPHONE ('When') renvoie HN.
X	- KS	- dans tous les cas	- METAPHONE ('Six') renvoie SKS.
Y	- Y	- quand suivi de A, E, I, O ou U	- METAPHONE ('Yang') renvoie YNK.
Y	- n/a	- dans tous les autres cas	- METAPHONE ('Bobby') renvoie BB.
Z	- S	- dans tous les cas	- METAPHONE ('Zack') renvoie SK.

<sup>1</sup>. Nombre entier 0.

- Ignore le caractère initial et encode la chaîne restante si les deux premiers caractères de la chaîne d'entrée ont une des valeurs suivantes :
  - **KN**. Par exemple, METAPHONE('KNOT') renvoie NT.
  - **GN**. Par exemple, METAPHONE('GNOB') renvoie NB.
  - **PN**. Par exemple, METAPHONE('PNRX') renvoie NRKS.
  - **AE**. Par exemple, METAPHONE('AERL') renvoie ERL.
- Si un caractère autre que "C" apparaît plusieurs fois dans la chaîne d'entrée, seule la première occurrence sera codée. Par exemple, METAPHONE('BBOX') renvoie BKS et METAPHONE('CCOX') renvoie KKKS.

## Syntaxe

```
METAPHONE( string [,length] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>chaîne</i>	Obligatoire	Doit être une chaîne de caractères. Transmet la valeur que vous voulez encoder. Le premier caractère doit être un caractère de l'alphabet anglais (A-Z). Vous pouvez entrer l'expression de transformation valide de votre choix. Ignore les caractères non alphabétiques dans la <i>chaîne</i> .
<i>longueur</i>	Facultatif	Doit être un nombre entier supérieur à 0. Spécifie le nombre de caractères dans la <i>chaîne</i> que vous voulez encoder. Vous pouvez entrer l'expression de transformation valide de votre choix. Lorsque la <i>longueur</i> est égale à 0 ou est supérieure à la longueur de , la chaîne d'entrée entière sera encodée. Par défaut 0.

## Valeur de retour

Chaîne.

NULL si l'une des conditions suivantes est vraie :

- Toutes les valeurs transmises à la fonction sont nulles.
- Aucun caractère dans la *chaîne* n'est une lettre de l'alphabet anglais.
- La *chaîne* est vide.

## Exemples

L'expression suivante encode les deux premiers caractères du port EMPLOYEE\_NAME dans une chaîne :

```
METAPHONE ( EMPLOYEE_NAME, 2 )
```

Employee_Name	Return Value
John	JH
*@# \$	NULL
P\$%%oc&&KMNL	PK

L'expression suivante encode les quatre premiers caractères du port EMPLOYEE\_NAME dans une chaîne :

```
METAPHONE ( EMPLOYEE_NAME, 4 )
```

Employee_Name	Return Value
John	JHN
1ABC	ABK
*@# \$	NULL
P\$%%oc&&KMNL	PKKM

# MIN (Dates)

Renvoie la date la plus antérieure dans un port ou un groupe. Vous pouvez appliquer un filtre pour limiter les lignes dans la recherche. Vous ne pouvez imbriquer qu'une seule autre fonction Agrégation dans MIN et la fonction imbriquée doit renvoyer une donnée de type date.

Vous pouvez également utiliser MIN pour renvoyer la plus petite valeur numérique ou la valeur de chaîne la plus faible dans un port ou un groupe.

## Syntaxe

```
MIN( date [, filter_condition] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>date</i>	Obligatoire	Type de données Date/Heure. Transmet les valeurs pour lesquelles vous voulez renvoyer la valeur minimum. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>filter_condition</i>	Facultatif	Limite les lignes dans la recherche. La condition de filtre doit être une valeur numérique ou renvoyer TRUE, FALSE ou NULL. Vous pouvez entrer l'expression de transformation valide de votre choix.

## Valeur de retour

Date si l'argument de *valeur* est une date.

NULL si toutes les valeurs passées à la fonction sont NULL, ou si aucune ligne n'est sélectionnée (par exemple, la condition de filtre renvoie FALSE ou NULL pour toutes les lignes).

## Valeurs nulles

Si une seule valeur est nulle, MIN l'ignore. Cependant, si toutes les valeurs transmises à partir du port sont NULL, MIN renvoie NULL.

## Grouper par

MIN groupe les valeurs en fonction des ports de regroupement que vous définissez dans la transformation et renvoie un résultat pour chaque groupe.

S'il n'existe aucun port de regroupement, MIN traite toutes les lignes comme un seul groupe et renvoie une seule valeur.

## Exemple

L'expression suivante renvoie la date de commande de lampes de poche la plus antérieure :

```
MIN( ORDER_DATE, ITEM_NAME='Flashlight' )
```

ITEM_NAME	ORDER_DATE
Flashlight	Apr 20 1998
Regulator System	May 15 1998
Flashlight	Sep 21 1998

ITEM_NAME	ORDER_DATE
Diving Hood	Aug 18 1998
Halogen Flashlight	Feb 1 1998
Flashlight	Oct 10 1998
Flashlight	NULL

**RETURN VALUE:** Feb 1 1998

## MIN (Nombres)

Revoie la plus petite valeur détectée dans un port ou un groupe. Vous pouvez appliquer un filtre pour limiter les lignes dans la recherche. Vous ne pouvez imbriquer qu'une seule autre fonction Agrégation dans MIN et la fonction imbriquée doit renvoyer un type de données numérique.

Vous pouvez également utiliser MIN pour renvoyer la date la plus récente ou la valeur de chaîne la plus faible dans un port ou un groupe.

### Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Obligatoire	Types de données numérique. Transmet les valeurs pour lesquelles vous voulez renvoyer la valeur minimum. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>filter_condition</i>	Facultatif	Limite les lignes dans la recherche. La condition de filtre doit être une valeur numérique ou renvoyer TRUE, FALSE ou NULL. Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

Valeur numérique.

NULL si toutes les valeurs passées à la fonction sont NULL, ou si aucune ligne n'est sélectionnée (par exemple, la condition de filtre renvoie FALSE ou NULL pour toutes les lignes).

**Remarque:** Si la valeur de retour est décimale avec une précision supérieure à 15, vous pouvez activer la précision élevée pour garantir une précision décimale jusqu'à 28 chiffres.

### Valeurs nulles

Si une seule valeur est nulle, MIN l'ignore. Cependant, si toutes les valeurs transmises à partir du port sont NULL, MIN renvoie NULL.

## Grouper par

MIN groupe les valeurs en fonction des ports de regroupement que vous définissez dans la transformation et renvoie un résultat pour chaque groupe.

S'il n'existe aucun port de regroupement, MIN traite toutes les lignes comme un seul groupe et renvoie une seule valeur.

## Exemple

L'expression suivante renvoie le prix minimum des lampes de poche :

```
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 (Chaîne)

Renvoie la valeur chaîne la plus faible trouvée dans un port ou un groupe. Vous pouvez appliquer un filtre pour limiter les lignes dans la recherche. Vous ne pouvez imbriquer qu'une seule autre fonction Agrégation dans MIN et la fonction imbriquée doit renvoyer une donnée de type chaîne.

**Remarque:** La fonction MIN utilise le même ordre de tri que la transformation Trieur. Cependant, la fonction MIN est sensible à la casse, mais la transformation Trieur peut ne pas l'être.

Vous pouvez également utiliser MIN pour renvoyer la dernière date ou la valeur numérique minimum dans un port ou un groupe.

## Syntaxe

```
MIN( string [, filter_condition] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>chaîne</i>	Obligatoire	Type de données Chaîne. Transmet les valeurs pour lesquelles vous voulez renvoyer la valeur minimum. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>filter_condition</i>	Facultatif	Limite les lignes dans la recherche. La condition de filtre doit être une valeur numérique ou renvoyer TRUE, FALSE ou NULL. Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

Valeur de chaîne.

NULL si toutes les valeurs passées à la fonction sont NULL, ou si aucune ligne n'est sélectionnée (par exemple, la condition de filtre renvoie FALSE ou NULL pour toutes les lignes).

### Valeurs nulles

Si une seule valeur est nulle, MIN l'ignore. Cependant, si toutes les valeurs transmises à partir du port sont NULL, MIN renvoie NULL.

### Grouper par

MIN groupe les valeurs en fonction des ports de regroupement que vous définissez dans la transformation et renvoie un résultat pour chaque groupe.

S'il n'existe aucun port de regroupement, MIN traite toutes les lignes comme un seul groupe et renvoie une seule valeur.

### Exemple

L'expression suivante renvoie le nom de l'élément minimum pour le fabricant 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
108	Wristband Thermometer

MANUFACTURER\_ID

ITEM\_NAME

RETURN VALUE: 60.6 cu ft Tank

## MOD

Revoie le reste d'un calcul de division. Par exemple, MOD (8,5) renvoie 3.

### Syntaxe

```
MOD( numeric_value, divisor )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Obligatoire	Type de données numérique. Valeurs que vous voulez diviser. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>diviseur</i>	Obligatoire	Valeur numérique avec laquelle effectuer la division. Le diviseur ne peut pas être 0.

### Valeur de retour

Valeur numérique du type de données que vous passez à la fonction. Reste de la valeur numérique divisée par le diviseur.

NULL si une valeur transmise à la fonction est NULL.

### Exemples

L'expression suivante renvoie le modulo des valeurs dans le port PRICE divisé par les valeurs dans le port 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 dernière ligne (25, 0) a généré une erreur, car vous ne pouvez diviser par 0. Pour éviter de diviser par 0, vous pouvez créer une expression comme suit, qui renvoie le modulo du prix divisé par la quantité uniquement si la quantité n'est pas égale à 0. Si la quantité est 0, la fonction renvoie 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 dernière ligne (25, 0) a généré une valeur nulle au lieu d'une erreur, car la fonction IIF remplace la valeur nulle par 0 dans le port QTY.

## MOVINGAVG

Renvoie la moyenne (ligne par ligne) d'un ensemble de lignes spécifique. En option, vous pouvez appliquer une condition pour filtrer les lignes avant de calculer la moyenne mobile.

### Syntaxe

```
MOVINGAVG( numeric_value, rowset [, filter_condition] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Obligatoire	Type de données numérique. Valeurs pour lesquelles vous souhaitez calculer une moyenne mobile. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>rowset</i>	Obligatoire	Doit être un littéral positif supérieur à 0. Définit l'ensemble de lignes pour lequel vous souhaitez calculer la moyenne mobile. Par exemple, si vous souhaitez calculer une moyenne mobile pour une colonne de données, cinq lignes à la fois, vous pouvez écrire une expression comme suit : <code>MOVINGAVG (SALES, 5)</code> .
<i>filter_condition</i>	Facultatif	Limite les lignes dans la recherche. La condition de filtre doit être une valeur numérique ou renvoyer TRUE, FALSE ou NULL. Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

Valeur numérique.

NULL si toutes les valeurs transmises à la fonction sont NULL, ou si aucune ligne n'est sélectionnée (par exemple : la condition de filtre renvoie FALSE ou NULL pour toutes les lignes).

**Remarque:** Si la valeur de retour est décimale avec une précision supérieure à 15, vous pouvez activer la précision élevée pour garantir une précision décimale jusqu'à 28 chiffres.

### Valeurs nulles

MOVINGAVG ignore les valeurs nulles lors du calcul de la moyenne mobile. Cependant, si toutes les valeurs sont NULL, la fonction renvoie NULL.

### Exemple

L'expression suivante renvoie les commandes moyennes de ventes de stabilisation, en fonction des cinq premières lignes dans le port Sales, puis renvoie la moyenne pour les cinq dernières lignes lues :

```
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 fonction renvoie la moyenne pour un ensemble de cinq lignes : 358 basé sur les lignes 1 à 5, 245.8 basé sur les lignes 2 à 6 et 243 basé sur les lignes 3 à 7.

# MOVINGSUM

Renvoie la somme (ligne par ligne) d'un ensemble de lignes spécifique.

Vous pouvez également appliquer une condition pour filtrer les lignes avant de calculer le total mobile.

## Syntaxe

```
MOVINGSUM( numeric_value, rowset [, filter_condition] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Obligatoire	Type de données numérique. Valeurs pour lesquelles vous souhaitez calculer un total mobile. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>rowset</i>	Obligatoire	Doit être un littéral positif supérieur à 0. Définit l'ensemble de lignes pour lequel vous souhaitez calculer le total mobile. Par exemple, si vous souhaitez calculer un total mobile pour une colonne de données, cinq lignes à la fois, vous pouvez écrire une expression comme suit : <code>MOVINGSUM( SALES, 5 )</code>
<i>filter_condition</i>	Facultatif	Limite les lignes dans la recherche. La condition de filtre doit être une valeur numérique ou renvoyer TRUE, FALSE ou NULL. Vous pouvez entrer l'expression de transformation valide de votre choix.

## Valeur de retour

Valeur numérique.

NULL si toutes les valeurs transmises à la fonction sont NULL, ou si la fonction ne sélectionne aucune ligne (par exemple, la condition de filtre renvoie FALSE ou NULL pour toutes les lignes).

**Remarque:** Si la valeur de retour est décimale avec une précision supérieure à 15, vous pouvez activer la précision élevée pour garantir une précision décimale jusqu'à 28 chiffres.

## Valeurs nulles

MOVINGSUM ignore les valeurs nulles lors du calcul du total mobile. Cependant, si toutes les valeurs sont NULL, la fonction renvoie NULL.

## Exemple

L'expression suivante renvoie la somme des commandes de gilets de stabilisation, basée sur les cinq premières lignes dans le port Sales, puis renvoie la moyenne pour les cinq dernières lignes lues :

```
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 fonction renvoie la somme pour un ensemble de cinq lignes : 1790 basé sur les lignes 1 à 5, 1229 basé sur les lignes 2 à 6 et 1215 basé sur les lignes 3 à 7.

## NPER

Renvoie le nombre de périodes pour un investissement d'après un taux d'intérêt constant et des paiements constants et réguliers.

### Syntaxe

`NPER( rate, present value, payment [, future value, type] )`

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>taux</i>	Obligatoire	Numérique Taux d'intérêt appliqué à chaque période Exprimé sous forme de nombre décimal. Divisez le taux par 100 pour l'exprimer en nombre décimal. Doit être supérieur ou égal à 0.
<i>valeur actuelle</i>	Obligatoire	Numérique Montant forfaitaire que représente actuellement une série de paiements futurs.
<i>paiement</i>	Obligatoire	Numérique Montant de paiement dû par période. Doit être un nombre négatif.
<i>Valeur future</i>	Facultatif	Numérique Solde que vous souhaitez atteindre après le dernier paiement. Si vous omettez cette valeur, NPER utilise 0.
<i>type</i>	Facultatif	Booléen. Moment du paiement. Entrez 1 si le paiement est effectué en début de période. Entrez 0 si le paiement est effectué en fin de période. Par défaut 0. Si vous entrez une valeur différente de 0 ou 1, le Service d'intégration de données traite la valeur comme 1.

### Valeur de retour

Numérique

### Exemple

La valeur actuelle d'un investissement est de 500 \$. Chaque paiement est de 2 000 \$ et la valeur future de l'investissement sera de 20 000 \$. L'expression suivante renvoie 9 comme nombre de périodes pour lesquelles vous devez effectuer les paiements :

`NPER ( 0.015, -500, -2000, 20000, TRUE )`

## Remarques

Pour calculer le taux d'intérêt appliqué à chaque période, divisez le taux annuel par le nombre total de paiements effectués dans une année. Par exemple, si vous effectuez des paiements mensuels à un taux d'intérêt annuel de 15 %, la valeur de l'argument de taux sera 15 % divisé par 12. Si vous effectuez des paiements annuels, la valeur de l'argument Taux sera de 15 %.

La valeur de paiement et la valeur actuelle sont négatives, car il s'agit de montants que vous payez.

# PERCENTILE

Calcule la valeur correspondant à un centile donné dans un groupe de nombres. Vous pouvez imbriquer une seule autre fonction Agrégation dans PERCENTILE et la fonction imbriquée doit renvoyer un type de données numérique.

Le Service d'intégration de données lit toutes les lignes de données pour effectuer le calcul du centile. Le processus de lecture des lignes pour effectuer le calcul peut affecter les performances. Vous pouvez également appliquer un filtre pour limiter les lignes que vous lisez pour calculer le centile.

## Syntaxe

```
PERCENTILE( numeric_value, percentile [, filter_condition ] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Obligatoire	Type de données numérique. Transmet les valeurs pour lesquelles vous souhaitez calculer un centile. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>centile</i>	Obligatoire	Nombre entier compris entre 0 et 100, inclus. Transmet le centile que vous souhaitez calculer. Vous pouvez entrer l'expression de transformation valide de votre choix. Si vous transmettez un nombre hors de la plage de 0 à 100, le Service d'intégration de données affiche une erreur et n'écrit pas la ligne.
<i>filter_condition</i>	Facultatif	Limite les lignes dans la recherche. La condition de filtre doit être une valeur numérique ou renvoyer TRUE, FALSE ou NULL. Vous pouvez entrer l'expression de transformation valide de votre choix.

## Valeur de retour

Valeur numérique.

NULL si toutes les valeurs passées à la fonction sont NULL, ou si aucune ligne n'est sélectionnée (par exemple, la condition de filtre renvoie FALSE ou NULL pour toutes les lignes).

**Remarque:** Si la valeur de retour est décimale avec une précision supérieure à 15, vous pouvez activer la précision élevée pour garantir une précision décimale jusqu'à 28 chiffres.

## Valeurs nulles

Si une valeur est nulle, PERCENTILE ignore la ligne. Cependant, si toutes les valeurs dans un groupe sont NULL, PERCENTILE renvoie NULL.

## Grouper par

PERCENTILE groupe des valeurs en fonction du regroupement par ports que vous définissez dans la transformation et renvoie un seul résultat pour chaque groupe.

S'il n'existe aucun port de regroupement, PERCENTILE traite toutes les lignes comme un seul groupe et renvoie une seule valeur.

## Exemple

Le Service d'intégration de données calcule un centile à l'aide de la logique suivante :

$$i = \frac{(x + 1) \times \text{percentile}}{100}$$

Utilisez les directives suivantes pour cette équation :

- X est le nombre d'éléments dans le groupe de valeurs pour lesquelles vous calculez un centile.
- Si  $i < 1$ , PERCENTILE renvoie la valeur du premier élément dans la liste.
- Si  $i$  est une valeur entière, PERCENTILE renvoie la valeur du  $i$ ème élément dans la liste.
- Dans le cas contraire, PERCENTILE renvoie la valeur de  $n$ :

$$n = [\text{[i ]th?element} \times (\lceil i \rceil - i)] + [\text{[i ]th?element} \times (i - \lfloor i \rfloor)]$$

L'expression suivante renvoie le salaire correspondant au 75e centile des salaires supérieurs à 50 000 \$ :

```
PERCENTILE( SALARY, 75, SALARY > 50000 )
```

### **SALARY**

125000.0

27900.0

100000.0

NULL

55000.0

9000.0

85000.0

86000.0

48000.0

99000.0

**RETURN VALUE:** 106250.0

# PMT

Renvoie le paiement pour un prêt d'après des paiements constants et un taux d'intérêt constant.

## Syntaxe

```
PMT( rate, terms, present value[, future value, type] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
taux	Obligatoire	Numérique Taux d'intérêt du prêt pour chaque période. Exprimé sous forme de nombre décimal. Divisez le taux par 100 pour l'exprimer en nombre décimal. Doit être supérieur ou égal à 0.
termes	Obligatoire	Numérique Nombre de périodes ou paiements. Doit être supérieur à 0.
valeur actuelle	Obligatoire	Numérique Objet du prêt.
Valeur future	Facultatif	Numérique Solde que vous voulez atteindre après le dernier paiement. Si vous omettez cette valeur, PMT utilise 0.
type	Facultatif	Booléen. Moment du paiement. Entrez 1 si le paiement est effectué en début de période. Entrez 0 si le paiement est effectué en fin de période. Par défaut 0. Si vous entrez une valeur différente de 0 ou 1, le Service d'intégration de données traite la valeur comme 1.

## Valeur de retour

Numérique

## Exemple

L'expression suivante renvoie -2111.64 comme montant du paiement mensuel d'un prêt :

```
PMT( 0.01, 10, 20000 )
```

## Remarques

Pour calculer le taux d'intérêt appliqué à chaque période, divisez le taux annuel par le nombre de paiements effectués dans l'année. Par exemple, si vous effectuez des paiements mensuels à un taux d'intérêt annuel de 15 %, le taux sera de 15 % / 12. Si vous effectuez des paiements annuels, le taux sera de 15 %.

La valeur de paiement est négative, car il s'agit de montants que vous payez.

# POWER

Renvoie une valeur élevée à l'exposant que vous passez à la fonction.

## Syntaxe

```
POWER( base, exponent )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>base</i>	Obligatoire	Valeur numérique. Cet argument est la valeur de base. Vous pouvez entrer l'expression de transformation valide de votre choix. Si la valeur de base est négative, l'exposant doit être un nombre entier.
<i>exposant</i>	Obligatoire	Valeur numérique. Cet argument est la valeur exposant. Vous pouvez entrer l'expression de transformation valide de votre choix. Si la valeur de base est négative, l'exposant doit être un nombre entier. Dans ce cas, la fonction arrondit les valeurs décimales à l'entier le plus proche avant de renvoyer une valeur.

### Valeur de retour

Valeur Double.

NULL si vous transmettez une valeur nulle à la fonction.

### Exemple

L'expression suivante renvoie les valeurs dans le port Numbers élevées aux valeurs dans le port Exponent :

```
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

La valeur -3.0 élevée à 6 renvoie les mêmes résultats que -3.0 élevée à 5.5. Si la base est négative, l'exposant doit être un nombre entier. Dans le cas contraire, le Service d'intégration de données arrondit l'exposant à la valeur entière la plus proche.

## PV

Renvoie la valeur actuelle d'un investissement.

## Syntaxe

```
PV( rate, terms, payment [, future value, type] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
taux	Obligatoire	Numérique Taux d'intérêt appliqué à chaque période Exprime sous forme de nombre décimal. Divisez le taux par 100 pour l'exprimer en nombre décimal. Doit être supérieur ou égal au 0.
termes	Obligatoire	Numérique Nombre de périodes ou de paiements. Doit être supérieur à 0.
paiements	Obligatoire	Numérique Montant de paiement dû par période. Doit être un nombre négatif.
Valeur future	Facultatif	Numérique Solde après le dernier paiement. Si vous omettez cette valeur, PV utilise 0.
types	Facultatif	Booléen. Moment du paiement. Entrez 1 si le paiement est effectué en début de période. Entrez 0 si le paiement est effectué en fin de période. Par défaut 0. Si vous entrez une valeur différente de 0 ou 1, le Service d'intégration de données traite la valeur comme 1.

## Valeur de retour

Numérique

## Exemple

L'expression suivante renvoie 12,524.43 comme montant que vous devez déposer dans le compte aujourd'hui pour obtenir une valeur future de 20 000 \$ en un an si vous déposez également 500 \$ au début de chaque période:

```
PV( 0.0075, 12, -500, 20000, TRUE )
```

# RAND

Renvoie un nombre aléatoire entre 0 et 1. Cette fonction est utile dans des scénarios de probabilité.

## Syntaxe

```
RAND( seed )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>valeur de départ aléatoire</i>	Facultatif	Numérique Valeur de départ pour la génération d'un nombre aléatoire par le service d'intégration. La valeur doit être une constante. Si vous ne saisissez aucune valeur de départ aléatoire, le Service d'intégration de données utilise l'heure système actuelle pour extraire le nombre de secondes à partir du 1er janvier 1971. Il utilise cette valeur comme valeur de départ aléatoire.

## Valeur de retour

Numérique

Pour la même valeur de départ aléatoire, le Service d'intégration de données génère la même séquence de nombres.

## Exemple

L'expression suivante peut renvoyer une valeur de 0.417022004702574 :

```
RAND (1)
```

# RATE

Renvoie le taux d'intérêt reçu par période pour une obligation.

## Syntaxe

```
RATE( terms, payment, present value[, future value, type] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
termes	Obligatoire	Numérique Nombre de périodes ou paiements. Doit être supérieur à 0.
paiements	Obligatoire	Numérique Montant de paiement dû par période. Doit être un nombre négatif.
valeur actuelle	Obligatoire	Numérique Montant forfaitaire que représente actuellement une série de paiements futurs.
Valeur future	Facultatif	Numérique Solde que vous voulez atteindre après le dernier paiement. Par exemple, la valeur future d'un prêt est de 0. Si vous omettez cette valeur, RATE utilise 0.
types	Facultatif	Booléen. Moment du paiement. Entrez 1 si le paiement est effectué en début de période. Entrez 0 si le paiement est effectué en fin de période. Par défaut 0. Si vous entrez une valeur différente de 0 ou 1, le Service d'intégration de données traite la valeur comme 1.

## Valeur de retour

Numérique

## Exemple

L'expression suivante renvoie 0.0077 comme taux d'intérêt mensuel d'un prêt :

```
RATE( 48, -500, 20000 )
```

Pour calculer le taux d'intérêt annuel du prêt, multipliez 0.0077 par 12. Le taux d'intérêt annuel est de 0.0924 ou 9.24 %.

# REG\_EXTRACT

Extrait des sous-tendances d'une expression régulière dans une valeur d'entrée. Par exemple, à partir d'un modèle d'expression régulière pour un nom complet, vous pouvez extraire le prénom ou le nom de famille.

**Remarque:** Utilisez la fonction REG\_REPLACE pour remplacer le modèle de caractère d'une chaîne par un autre modèle de caractère.

## Syntaxe

```
REG_EXTRACT( subject, 'pattern', subPatternNum )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>subject</i>	Requis	Type de données de chaîne. Transmet la valeur à comparer avec le modèle d'expression régulière.
<i>pattern</i>	Requis	Type de données de chaîne. Modèle d'expression régulière avec lequel établir une correspondance. Vous devez utiliser une syntaxe d'expression régulière compatible perl. Placez le modèle entre guillemets simples. Placez chaque sous-modèle entre parenthèses.
<i>subPatternNum</i>	Facultatif	Valeur entière. Nombre de sous-modèles de l'expression régulière à faire correspondre. Utilisez les directives suivantes pour définir le nombre de sous-modèles : <ul style="list-style-type: none"><li>- Aucune valeur ou 1. Extrait le premier sous-modèle d'expression régulière.</li><li>- 2. Extrait le deuxième sous-modèle d'expression régulière.</li><li>- n. Extrait le nième sous-modèle d'expression régulière.</li></ul> La valeur par défaut est 1.

## Utilisation de la syntaxe d'expression régulière compatible perl

Vous devez utiliser la syntaxe d'expression régulière compatible perl avec les fonctions REG\_EXTRACT, REG\_MATCH et REG\_REPLACE.

Le tableau suivant fournit les instructions relatives à la syntaxe d'expression régulière compatible perl :

Syntaxe	Description
.	Fait correspondre l'un des caractères.
[a-z]	Fait correspondre une instance d'un caractère en minuscule. Par exemple, [a-z] correspond à ab. Utilisez [A-Z] pour faire correspondre des caractères en majuscule.
\d	Fait correspondre une instance d'un chiffre de 0 à 9.
\s	Fait correspondre une espace.
\w	Fait correspondre un caractère alphanumérique, y compris le soulignement (_).
()	Groupe une expression. Par exemple, les parenthèses dans (\d-\d-\d\d) groupent l'expression \d \d-\d\d, qui recherche deux nombres suivis d'un trait d'union, puis de deux nombres (12-34, par exemple).
{}	Fait correspondre le nombre de caractères. Par exemple, \d{3} fait correspondre trois chiffres (650 ou 510, par exemple). De même, [a-z]{2} fait correspondre deux lettres (CA ou NY, par exemple).
?	Fait correspondre ou non le caractère ou le groupe de caractères précédent. Par exemple, \d{3}(-{d{4}})? fait correspondre trois chiffres, qui peuvent être suivi par un trait d'union et quatre chiffres.
*	Fait correspondre aucune ou plusieurs instances des valeurs qui suivent l'astérisque. Par exemple, *0 correspond à toute valeur précédant un 0.
+	Fait correspondre une ou plusieurs instances des valeurs qui suivent le signe plus. Par exemple, \w+ correspond à toute valeur suivant un caractère alphanumérique.

Par exemple, l'expression régulière recherche des codes postaux des Etats-Unis à 5 chiffres, par exemple 93930 et des codes postaux à 9 chiffres, par exemple 93930-5407 :

```
\d{5}(-\d{4})?
```

\d{5} fait référence à cinq chiffres, par exemple 93930. Les parenthèses autour de -\d{4} groupe ce segment de l'expression. Le trait d'union représente le trait d'union dans un code postal à 9 chiffres, comme dans 93930-5407. \d{4} fait référence à quatre chiffres, par exemple 5407. Le point d'interrogation indique que le trait d'union et quatre derniers chiffres sont facultatifs ou peuvent apparaître une seule fois.

### Conversion de la syntaxe COBOL en syntaxe d'expression régulière compatible perl

Si vous connaissez la syntaxe COBOL, vous pouvez utiliser les informations suivantes pour écrire des expressions régulières compatibles perl.

Le tableau suivant présente des exemples de syntaxe COBOL et leurs équivalents perl :

Syntaxe COBOL	Syntaxe perl	Description
9	\d	Fait correspondre une instance d'un chiffre de 0 à 9.
9999	\d\d\d\d ou \d{4}	Fait correspondre quatre chiffres de 0 à 9 (1234 ou 5936, par exemple).

Syntaxe COBOL	Syntaxe perl	Description
Q	[a-z]	Fait correspondre une instance d'une lettre.
9xx9	\d[a-z][a-z]\d	Fait correspondre tout chiffre suivi de deux lettres et d'un autre chiffre (1ab2, par exemple).

### Conversion de la syntaxe SQL en syntaxe d'expression régulière compatible perl

Si vous connaissez la syntaxe SQL, vous pouvez utiliser les informations suivantes pour écrire des expressions régulières compatibles perl.

Le tableau suivant présente des exemples de syntaxe SQL et leurs équivalents perl :

Syntaxe SQL	Syntaxe perl	Description
%	.*	Fait correspondre toute chaîne.
A%	A.*	Fait correspondre la lettre « A » suivie d'une chaîne (« Année », par exemple).
_	.(point)	Fait correspondre l'un des caractères.
A_	A.	Fait correspondre « A » suivi d'un caractère (AZ, par exemple).

### Valeur de retour

Renvoie la valeur du *n*ième sous-modèle dans la valeur d'entrée. Le *n*-ième sous-modèle est basé sur la valeur que vous spécifiez pour subPatternNum.

NULL si l'entrée est une valeur nulle ou si le modèle est nul.

### Exemple

Vous pouvez utiliser REG\_EXTRACT dans une expression pour extraire les deuxièmes prénoms d'une expression régulière qui fait correspondre les prénoms, deuxièmes prénoms et noms de famille. Par exemple, l'expression suivante renvoie le deuxième prénom d'une expression régulière :

```
REG_EXTRACT( Employee_Name, '(\w+)\s+(\w+)\s+(\w+)', 2)
```

Employee_Name	Return Value
Stephen Graham Smith	Graham
Juan Carlos Fernando	Carlos

## REG\_MATCH

Indique si une valeur correspond ou non à un modèle d'expression régulière. Cela permet de valider des modèles de données, tels que des ID, des numéros de téléphone, des codes postaux et des noms d'état.

**Remarque:** Utilisez la fonction REG\_REPLACE pour remplacer un modèle de caractère dans une chaîne par un nouveau modèle de caractère.

## Syntaxe

```
REG_MATCH( subject, pattern )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i> sujet </i>	Obligatoire	Type de données Chaîne. Transmet la valeur à faire correspondre au modèle d'expression régulière.
<i> modèle </i>	Obligatoire	Type de données Chaîne. Modèle d'expression régulière avec lequel vous voulez établir une correspondance. Vous devez utiliser une syntaxe d'expression régulière compatible perl. Placez le modèle entre guillemets simples. Pour plus d'informations, consultez <a href="#">"REG_EXTRACT" à la page 140</a> .

## Valeur de retour

TRUE si les données correspondent au modèle.

FALSE si les données ne correspondent pas au modèle.

NULL si l'entrée est une valeur nulle ou si le modèle est NULL.

## Exemple

Vous pouvez utiliser REG\_MATCH dans une expression pour valider des numéros de téléphone. Par exemple, l'expression suivante correspond à un numéro de téléphone à 10 chiffres dans le modèle et renvoie une valeur booléenne basée sur la correspondance :

```
REG_MATCH (Phone_Number, '(\d\d\d-\d\d\d-\d\d\d\d)')
```

Phone_Number	Return Value
408-555-1212	TRUE
510-555-1212	TRUE
92 555 51212	FALSE
650-555-1212	TRUE
415-555-1212	TRUE
831 555 12123	FALSE

## Conseil

Vous pouvez également utiliser REG\_MATCH pour réaliser les tâches suivantes :

- Permet de vérifier qu'une valeur correspond à un modèle. Cette utilisation est similaire à la fonction SQL LIKE.
- Permet de vérifier que les valeurs sont des caractères. Cette utilisation est similaire à la fonction SQL IS\_CHAR.

Pour vérifier qu'une valeur correspond à un modèle, utilisez un point (.) et un astérisque (\*) avec la fonction REG\_MATCH dans une expression. Un point correspond à tout caractère. Un astérisque correspond à 0 ou plusieurs instances de valeurs qui le suivent.

Par exemple, utilisez l'expression suivante pour rechercher des numéros de compte qui commencent par 1835 :

```
REG_MATCH (ACCOUNT_NUMBER, '1835.*')
```

Pour vérifier que les valeurs sont des caractères, utilisez la fonction REG\_MATCH avec l'expression régulière [a-zA-Z]+. a-z fait correspondre tous les caractères minuscules. A-Z fait correspondre tous les caractères majuscules. Le signe plus (+) indique qu'au moins un caractère doit figurer.

Par exemple, utilisez l'expression suivante pour vérifier qu'une liste de noms de famille contient uniquement des caractères :

```
REG_MATCH (LAST_NAME, '[a-zA-Z]+')
```

## REG\_REPLACE

Remplace les caractères d'une chaîne par un autre modèle de caractère. Par défaut, REG\_REPLACE recherche la chaîne d'entrée pour le modèle de caractères que vous indiquez et remplace toutes les occurrences par le modèle de remplacement. Vous pouvez également indiquer le nombre d'occurrences du modèle que vous voulez remplacer dans la chaîne.

### Syntaxe

```
REG_REPLACE( subject, pattern, replace, numReplacements )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i> sujet </i>	Obligatoire	Type de données Chaîne. Transmet la chaîne que vous voulez rechercher.
<i> modèle </i>	Obligatoire	Type de données Chaîne. Transmet la chaîne de caractères à remplacer. Vous devez utiliser une syntaxe d'expression régulière compatible perl. Placez le modèle entre guillemets simples. Pour plus d'informations, consultez <a href="#">"REG_EXTRACT" à la page 140.</a>
<i> remplacer </i>	Obligatoire	Type de données Chaîne. Transmet la nouvelle chaîne de caractères.
<i> numReplacements </i>	Facultatif	Type de données numérique. Spécifie le nombre d'occurrences à remplacer. Si vous omettez cette option, REG_REPLACE remplacera toutes les occurrences de la chaîne de caractères.

### Valeur de retour

Chaîne

## Exemple

L'expression suivante supprime les espaces supplémentaires dans les données de noms d'employés pour chaque ligne du port `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

Remplace certains caractères d'une chaîne par un seul ou aucun caractère. `REPLACECHR` recherche dans la chaîne d'entrée les caractères spécifiés et remplace toutes les occurrences de l'ensemble de ces caractères par le nouveau caractère spécifié.

## Syntaxe

```
REPLACECHR( CaseFlag, InputString, OldCharSet, NewChar )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>CaseFlag</i>	Requis	Doit être un nombre entier. Détermine si les arguments de cette fonction sont sensibles à la casse. Vous pouvez entrer l'expression de transformation valide de votre choix. Si <i>CaseFlag</i> est un nombre différent de 0, la fonction est sensible à la casse. Si <i>CaseFlag</i> est une valeur nulle ou 0, la fonction n'est pas sensible à la casse.
<i>InputString</i>	Requis	Doit être une chaîne de caractères. Transmet la chaîne à rechercher. Vous pouvez entrer l'expression de transformation valide de votre choix. Si vous transmettez une valeur numérique, la fonction la convertit en une chaîne de caractères. Si la valeur de <i>InputString</i> est NULL, <code>REPLACECHR</code> renvoie NULL.

Argument	Obligatoire / Facultatif	Description
<i>OldCharSet</i>	Requis	Doit être une chaîne de caractères. Caractères à remplacer. Vous pouvez entrer un ou plusieurs caractères. Vous pouvez entrer l'expression de transformation valide de votre choix. Vous pouvez également entrer un littéral de texte entre guillemets simples ('abc', par exemple). Si vous transmettez une valeur numérique, la fonction la convertit en une chaîne de caractères. Si la valeur de <i>OldCharSet</i> est NULL ou vide, REPLACECHR renvoie <i>InputString</i> .
<i>NewChar</i>	Requis	Doit être une chaîne de caractères. Vous pouvez entrer un caractère, une chaîne vide, ou une valeur NULL. Vous pouvez entrer l'expression de transformation valide de votre choix. Si la valeur de <i>NewChar</i> est NULL ou vide, REPLACECHR supprime l'ensemble des occurrences de tous les caractères de <i>OldCharSet</i> dans <i>InputString</i> . Si <i>NewChar</i> contient plusieurs caractères, REPLACECHR utilise le premier caractère pour remplacer <i>OldCharSet</i> .

## Valeur de retour

Chaîne.

Chaîne vide si REPLACECHR supprime tous les caractères dans *InputString*.

NULL si *InputString* est NULL.

*InputString* si *OldCharSet* est NULL ou vide.

## Exemples

L'expression suivante supprime les guillemets doubles des données du journal web pour chaque ligne dans le port 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

L'expression suivante supprime plusieurs caractères pour chaque ligne dans le port 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

L'expression suivante modifie une partie de la valeur du code client pour chaque ligne dans le port CUSTOMER\_CODE :

```
REPLACECHR ( 1, CUSTOMER_CODE, 'A', 'M' )
```

CUSTOMER_CODE	RETURN VALUE
ABA	MBM
abA	abM
BBC	BBC
ACC	MCC
NULL	NULL

L'expression suivante modifie une partie de la valeur du code client pour chaque ligne dans le port CUSTOMER\_CODE :

```
REPLACECHR ( 0, CUSTOMER_CODE, 'A', 'M' )
```

CUSTOMER_CODE	RETURN VALUE
ABA	MBM
abA	MbM
BBC	BBC
ACC	MCC

L'expression suivante modifie une partie de la valeur du code client pour chaque ligne dans le port 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

L'expression suivante supprime les nombres multiples pour chaque ligne dans le port INPUT :

```
REPLACECHR ( 1, INPUT, '14', NULL )
```

INPUT	RETURN VALUE
12345	235
4141	NULL
111115	5
NULL	NULL

Lorsque vous voulez utiliser un guillemet simple (') dans *OldCharSet* ou *NewChar*, vous devez utiliser la fonction CHR. Le guillemet simple est le seul caractère qui ne peut être utilisé dans un littéral chaîne.

L'expression suivante supprime plusieurs caractères, y compris le guillemet simple, pour chaque ligne dans le port INPUT :

```
REPLACECHR (1, INPUT, CHR(39), NULL )
```

INPUT	RETURN VALUE
'Tom Smith' 'Laura Jones'	Tom Smith Laura Jones
Tom's	Toms
NULL	NULL

## REPLACESTR

Remplace des caractères d'une chaîne par un seul, aucun ou plusieurs caractères. REPLACESTR recherche la chaîne d'entrée pour toutes les chaînes que vous indiquez et les remplace par la nouvelle chaîne indiquée.

### Syntaxe

```
REPLACESTR ( CaseFlag, InputString, OldString1, [OldString2, ... OldStringN,] NewString )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>CaseFlag</i>	Obligatoire	Doit être un nombre entier. Détermine si les arguments de cette fonction sont sensibles à la casse. Vous pouvez entrer l'expression de transformation valide de votre choix. Si <i>CaseFlag</i> est un nombre différent de 0, la fonction est sensible à la casse. Si <i>CaseFlag</i> est une valeur nulle ou 0, la fonction n'est pas sensible à la casse.
<i>InputString</i>	Obligatoire	Doit être une chaîne de caractères. Transmet les chaînes à rechercher. Vous pouvez entrer l'expression de transformation valide de votre choix. Si vous transmettez une valeur numérique, la fonction la convertit en une chaîne de caractères. Si <i>InputString</i> est nulle, REPLACESTR renvoie NULL.
<i>OldString</i>	Obligatoire	Doit être une chaîne de caractères. Chaîne à remplacer. Vous devez entrer au moins un argument <i>OldString</i> . Vous pouvez entrer un ou plusieurs caractères par argument <i>OldString</i> . Vous pouvez entrer l'expression de transformation valide de votre choix. Vous pouvez également saisir un littéral de texte placé entre guillemets simples, par exemple, 'abc'. Si vous transmettez une valeur numérique, la fonction la convertit en une chaîne de caractères. Si REPLACESTR contient plusieurs arguments <i>OldString</i> et qu'un ou plusieurs arguments <i>OldString</i> sont NULL ou vides, REPLACESTR ignore l'argument <i>OldString</i> . Lorsque tous les arguments <i>OldString</i> sont nuls ou vides, REPLACESTR renvoie <i>InputString</i> . La fonction remplace les caractères dans les arguments <i>OldString</i> dans l'ordre d'apparition dans la fonction. Par exemple, si vous entrez plusieurs arguments <i>OldString</i> , le premier argument <i>OldString</i> est prioritaire sur le deuxième argument <i>OldString</i> et le deuxième argument <i>OldString</i> est prioritaire sur le troisième argument <i>OldString</i> . Lorsque REPLACESTR remplace une chaîne, il place le curseur après les caractères remplacés dans <i>InputString</i> avant de rechercher la correspondance suivante.
<i>NewString</i>	Obligatoire	Doit être une chaîne de caractères. Vous pouvez entrer un ou plusieurs caractères, une chaîne vide, ou NULL. Vous pouvez entrer l'expression de transformation valide de votre choix. Si <i>NewString</i> est NULL ou vide, REPLACESTR supprime toutes les occurrences de <i>OldString</i> dans <i>InputString</i> .

### Valeur de retour

Chaîne.

Chaîne vide si REPLACESTR supprime tous les caractères dans *InputString*.

NULL si *InputString* est NULL.

*InputString* si tous les arguments *OldString* sont NULL ou vides.

## Exemples

L'expression suivante supprime les guillemets doubles et deux chaînes de texte différentes des données du journal web pour chaque ligne dans le port 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

L'expression modifie le titre de certaines valeurs pour chaque ligne dans le port TITLE :

```
REPLACESTR ( 1, TITLE, 'rs.', 'iss', 's.' )
```

TITLE	RETURN VALUE
Mrs.	Ms.
Miss	Ms.
Mr.	Mr.
MRS.	MRS.

L'expression modifie le titre de certaines valeurs pour chaque ligne dans le port TITLE :

```
REPLACESTR ( 0, TITLE, 'rs.', 'iss', 's.' )
```

TITLE	RETURN VALUE
Mrs.	Ms.
MRS.	Ms.

L'expression suivante montre comment la fonction REPLACESTR remplace plusieurs arguments OldString pour chaque ligne dans le port INPUT :

```
REPLACESTR ( 1, INPUT, 'ab', 'bc', '*' )
```

INPUT	RETURN VALUE
abc	*c
abbc	**
abbbbc	*bb*
bc	*

L'expression suivante montre comment la fonction REPLACESTR remplace plusieurs arguments *OldString* pour chaque ligne dans le port INPUT :

```
REPLACESTR ( 1, INPUT, 'ab', 'bc', 'b' )
```

INPUT	RETURN VALUE
ab	b
bc	b
abc	bc
abbc	bb
abbcc	bbc

Lorsque vous voulez utiliser un guillemet simple (') dans *OldString* ou *NewString*, vous devez utiliser la fonction CHR. Utilisez les fonctions CHR et CONCAT pour concaténer un guillemet simple dans une chaîne. Le guillemet simple est le seul caractère qui ne peut être utilisé dans un littéral chaîne. Considérons l'exemple suivant :

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

La valeur de retour est :

```
Joan's car
```

L'expression suivante modifie une chaîne qui comprend le guillemet simple, pour chaque ligne dans le port 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

Inverse la chaîne d'entrée.

### Syntaxe

```
REVERSE( string )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire/ Facultatif	Description
<i>chaîne</i>	Obligatoire	Toute valeur de caractère. Valeur que vous voulez inverser.

### Valeur de retour

Chaîne. Inversement de la valeur d'entrée.

### Exemple

L'expression suivante inverse les chiffres du code client :

```
REVERSE ( CUSTOMER_CODE )
```

CUSTOMER_CODE	RETURN VALUE
0001	1000
0002	2000
0003	3000
0004	4000

## ROUND (Dates)

Arrondit une partie d'une date. Vous pouvez également utiliser ROUND pour arrondir des nombres.

Cette fonction permet d'arrondir les parties d'une date :

#### Année

Arrondit la partie année d'une date en fonction du mois.

#### Mois

Arrondit la partie mois d'une date en fonction du jour du mois.

#### Jour

Arrondit la partie jour de la date en fonction de l'heure.

#### Heure

Arrondit la partie heure de la date en fonction des minutes de l'heure.

#### Minute

Arrondit la partie minute de la date en fonction des secondes.

#### Seconde

Arrondit la deuxième partie de la date en fonction des millisecondes.

#### Milliseconde

Arrondit la partie milliseconde de la date en fonction des microsecondes.

## Microseconde

Arrondit la partie microseconde de la date en fonction des nanosecondes.

Le tableau suivant indique les conditions utilisées par l'expression ROUND et les valeurs de retour :

Condition	Expression	Valeur de retour
Pour les mois compris entre janvier et juin, la fonction renvoie 1er janvier de la même année et définit l'heure sur 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
Pour les mois compris entre juillet et décembre, la fonction renvoie 1er janvier de l'année suivante et définit l'heure sur 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
Pour les jours du mois compris entre le 1er et le 15, la fonction renvoie la date du premier jour du mois entré et définit l'heure sur 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
Pour les jours du mois compris entre le 16 et le dernier jour du mois, la fonction renvoie le premier jour du mois suivant et définit l'heure sur 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
Pour une heure comprise entre 00:00:00 (minuit) et 11:59:59, la fonction renvoie la date en cours et définit l'heure sur 00:00:00.000000000 (minuit).	ROUND(TO_DATE('06/13/1998 2:30:45', 'MM/DD/YYYY HH24:MI:SS'), 'DD')	06/13/1998 00:00:00.000000000
Pour une heure égale ou ultérieure à 12:00:00 (midi), la fonction arrondit la date au jour suivant et définit l'heure sur 00:00:00.000000000 (minuit).	ROUND(TO_DATE('06/13/1998 22:30:45', 'MM/DD/YYYY HH24:MI:SS'), 'DD')	06/14/1998 00:00:00.000000000
Pour la partie des minutes comprises entre 0 et 29 minutes dans l'heure, la fonction renvoie l'heure en cours et définit les minutes, les secondes, les millisecondes et les nanosecondes sur 0.	ROUND(TO_DATE('04/01/1998 11:29:35', 'MM/DD/YYYY HH24:MI:SS'), 'HH')	04/01/1998 11:00:00.000000000
Pour la partie des minutes à partir de 30, la fonction renvoie l'heure suivante et définit les minutes, les secondes, les millisecondes et les nanosecondes sur 0.	ROUND(TO_DATE('04/01/1998 13:39:00', 'MM/DD/YYYY HH24:MI:SS'), 'HH')	04/01/1998 14:00:00.000000000
Pour la partie de l'heure comprise entre 0 et 29 secondes, la fonction renvoie la minute en cours et définit les secondes, les millisecondes et les nanosecondes sur 0.	ROUND(TO_DATE('05/22/1998 10:15:29', 'MM/DD/YYYY HH24:MI:SS'), 'MI')	05/22/1998 10:15:00.000000000
Pour la partie de l'heure comprise entre 30 et 59 secondes, la fonction renvoie la minute suivante et définit les secondes, les millisecondes et les nanosecondes sur 0.	ROUND(TO_DATE('05/22/1998 10:15:30', 'MM/DD/YYYY HH24:MI:SS'), 'MI')	05/22/1998 10:16:00.000000000

Condition	Expression	Valeur de retour
Pour la partie de l'heure comprise entre 0 et 499 millisecondes, la fonction renvoie la seconde en cours et définit les millisecondes sur 0.	<code>ROUND(TO_DATE('05/22/1998 10:15:29.499', 'MM/DD/YYYY HH24:MI:SS.MS'), 'SS')</code>	05/22/1998 10:15:29.000000000
Pour la partie de l'heure comprise entre 500 et 999 millisecondes, la fonction renvoie la seconde suivante et définit les millisecondes sur 0.	<code>ROUND(TO_DATE('05/22/1998 10:15:29.500', 'MM/DD/YYYY HH24:MI:SS.MS'), 'SS')</code>	05/22/1998 10:15:30.000000000
Pour l'heure comprise entre 0 et 499 microsecondes, la fonction renvoie la milliseconde en cours et définit les microsecondes sur 0.	<code>ROUND(TO_DATE('05/22/1998 10:15:29.498125', 'MM/DD/YYYY HH24:MI:SS.US'), 'MS')</code>	05/22/1998 10:15:29.498000000
Pour l'heure comprise entre 500 et 999 microsecondes, la fonction renvoie la milliseconde suivante et définit les microsecondes sur 0.	<code>ROUND(TO_DATE('05/22/1998 10:15:29.498785', 'MM/DD/YYYY HH24:MI:SS.US'), 'MS')</code>	05/22/1998 10:15:29.499000000
Pour la partie de l'heure comprise entre 0 et 499 nanosecondes, la fonction renvoie la microseconde en cours et définit les nanosecondes sur 0.	<code>ROUND(TO_DATE('05/22/1998 10:15:29.498125345', 'MM/DD/YYYY HH24:MI:SS.NS'), 'US')</code>	05/22/1998 10:15:29.498125000
Pour la partie de l'heure comprise entre 500 et 999 nanosecondes, la fonction renvoie la microseconde suivante et définit les nanosecondes sur 0.	<code>ROUND(TO_DATE('05/22/1998 10:15:29.498125876', 'MM/DD/YYYY HH24:MI:SS.NS'), 'US')</code>	05/22/1998 10:15:29.498126000

## Syntaxe

```
ROUND( date [,format] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>date</i>	Requis	Type de données Date/Heure. Vous pouvez imbriquer TO_DATE pour convertir les chaînes en dates avant d'effectuer l'arrondi.
<i>format</i>	Facultatif	Entrez une chaîne de format valide. Il s'agit de la partie de la date à arrondir. Seule une partie de la date peut être arrondie. Si vous omettez la chaîne de format, la fonction arrondit la date au jour le plus proche.

## Valeur de retour

Date avec la partie spécifiée arrondie. ROUND renvoie une date au même format que la date source. Vous pouvez lier les résultats de cette fonction à un port dont le type de données est Date/Heure.

NULL si vous transmettez une valeur nulle à la fonction.

## Exemples

Les expressions suivantes permettent d'arrondir la partie année des dates dans le port DATE\_SHIPPED :

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

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 1 1998 12:00:00.000000000AM
Apr 19 1998 1:31:20PM	Jan 1 1998 12:00:00.000000000AM
Dec 20 1998 3:29:55PM	Jan 1 1999 12:00:00.000000000AM
NULL	NULL

Les expressions suivantes permettent d'arrondir la partie mois de chaque date dans le port 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

Les expressions suivantes permettent d'arrondir la partie jour de chaque date dans le port 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

Les expressions permettent d'arrondir la partie heure de chaque date dans le port 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
Dec 31 1998 11:59:59PM	Jan 1 1999 12:00:00.000000000AM
NULL	NULL

L'expression suivante permet d'arrondir la partie minute de chaque date dans le port 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 (Nombres)

Arrondit les nombres à un nombre de chiffres ou de décimales spécifique. Vous pouvez également utiliser ROUND pour arrondir des dates.

### Syntaxe

```
ROUND( numeric_value [, precision] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Requis	Type de données numérique. Vous pouvez entrer l'expression de transformation valide de votre choix. Utilisez des opérateurs pour effectuer des opérations arithmétiques avant d'arrondir les valeurs.
<i>précision</i>	Facultatif	<p>Nombre entier positif ou négatif. Si vous entrez une <i>précision</i> positive, la fonction arrondit ce nombre aux décimales. Par exemple, ROUND(12.99, 1) renvoie 13.0 et ROUND(15.44, 1) renvoie 15.4.</p> <p>Si vous entrez une <i>précision</i> négative, la fonction arrondit ce nombre de chiffres à gauche de la virgule et renvoie un nombre entier. Par exemple, ROUND(12.99, -1) renvoie 10 et ROUND(15.99, -1) renvoie 20.</p> <p>Si vous entrez une <i>précision</i> décimale, la fonction arrondit à l'entier le plus proche avant d'évaluer l'expression. Par exemple, ROUND(12.99, 0.8) renvoie 13.0, car la fonction arrondit 0.8 à 1, puis évalue l'expression.</p> <p>Si vous omettez l'argument <i>précision</i>, la fonction arrondit au nombre entier le plus proche, en tronquant la partie décimale du nombre. Par exemple, ROUND(12.99) renvoie 13.</p>

## Valeur de retour

Valeur numérique.

Si l'un des arguments est NULL, ROUND renvoie NULL.

**Remarque:** Si la valeur de retour est décimale avec une précision supérieure à 15, vous pouvez activer la précision élevée pour garantir une précision décimale jusqu'à 28 chiffres.

## Exemples

L'expression suivante renvoie les valeurs du port Price arrondies à trois décimales.

```
ROUND( PRICE, 3 )
```

PRICE	RETURN VALUE
12.9936	12.994
15.9949	15.995
-18.8678	-18.868
56.9561	56.956
NULL	NULL

Vous pouvez arrondir des chiffres à gauche de la décimale en transmettant un nombre entier négatif dans l'argument de *précision* :

```
ROUND( PRICE, -2 )
```

PRICE	RETURN VALUE
13242.99	13200.0

PRICE	RETURN VALUE
1435.99	1400.0
-108.95	-100.0
NULL	NULL

Si vous transmettez une valeur décimale dans l'argument de *précision*, le Service d'intégration de données l'arrondit au nombre entier le plus proche avant d'évaluer l'expression:

```
ROUND( PRICE, 0.8 )
```

PRICE	RETURN VALUE
12.99	13.0
56.34	56.3
NULL	NULL

Si vous omettez l'argument de *précision*, la fonction arrondit à l'entier le plus proche :

```
ROUND( PRICE )
```

PRICE	RETURN VALUE
12.99	13.0
-15.99	-16.0
-18.99	-19.0
56.95	57.0
NULL	NULL

## Conseil

Vous pouvez également utiliser ROUND pour définir explicitement la précision des valeurs calculées et obtenir les résultats attendus. Lorsque le Service d'intégration de données est exécuté en mode précision faible, il tronque le résultat des calculs si la précision de la valeur est supérieure à 15 chiffres. Par exemple, vous pouvez traiter l'expression suivante en mode de précision faible :

```
7/3 * 3 = 7
```

Dans ce cas, le Service d'intégration de données évalue le côté gauche de l'expression sous la forme 6.999999999999999, car il tronque le résultat de la première opération de division. Le Service d'intégration de données évalue l'expression complète comme FALSE. Ceci n'est peut-être pas le résultat que vous attendiez.

Pour obtenir le résultat attendu, utilisez ROUND pour arrondir le résultat tronqué du côté gauche de l'expression au résultat attendu. Le Service d'intégration de données évalue l'expression suivante comme TRUE :

```
ROUND(7/3 * 3) = 7
```

# RPAD

Convertit une chaîne à une longueur spécifiée en ajoutant des espaces ou des caractères à la fin de la chaîne.

## Syntaxe

```
RPAD( first_string, length [,second_string] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>first_string</i>	Obligatoire	Toute valeur de chaîne. Chaînes à modifier. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>longueur</i>	Obligatoire	Doit être un littéral entier positif. Spécifie la longueur que vous voulez attribuer à chaque chaîne.
<i>second_string</i>	Facultatif	Toute valeur de chaîne. Transmet la chaîne que vous voulez ajouter à droite des valeurs <i>first_string</i> . Placez les caractères que vous souhaitez ajouter à la fin de la chaîne entre guillemets simples, par exemple, 'abc'. Cet argument est sensible à la casse.  Si vous omettez la deuxième chaîne, la fonction remplit la fin de la première chaîne avec des espaces.

## Valeur de retour

Chaîne de la longueur spécifiée.

NULL si une valeur transmise à la fonction est NULL ou si la longueur est un nombre négatif.

## Exemples

L'expression suivante renvoie le nom de l'élément dont la longueur est de 16 caractères et adjoint la chaîne '.' à la fin de chaque nom d'élément :

```
RPAD( ITEM_NAME, 16, '.')
```

ITEM_NAME	RETURN VALUE
Flashlight	Flashlight.....
Compass	Compass.....
Regulator System	Regulator System
Safety Knife	Safety Knife....

RPAD compte la longueur de gauche à droite. Par conséquent, si la première chaîne est supérieure à la longueur, RPAD tronque la chaîne de droite à gauche. Par exemple, RPAD('alphabetical', 5, 'x') renverra la chaîne 'alpha'. RPAD utilise une partie de la *second\_string* si nécessaire.

L'expression suivante renvoie le nom de l'élément dont la longueur est de 16 caractères, en ajoutant la chaîne '\*..\*' à la fin de chaque nom d'élément :

```
RPAD( ITEM_NAME, 16, '*..*' )
```

ITEM_NAME	RETURN VALUE
Flashlight	Flashlight*..**.
Compass	Compass*..**..**
Regulator System	Regulator System
Safety Knife	Safety Knife*..*

## RTRIM

Supprime des espaces ou caractères de la fin d'une chaîne.

Si vous ne spécifiez aucun paramètre *trim\_set* dans l'expression :

- En mode Unicode, RTRIM supprime les espaces à octet simple ou à deux octets à la fin d'une chaîne.
- En mode ASCII, RTRIM supprime uniquement les espaces à octet simple.

Si vous utilisez RTRIM pour supprimer des caractères d'une chaîne, RTRIM compare *trim\_set* à chaque caractère dans l'argument *chaîne*, un caractère après l'autre, en commençant par le côté droit de la chaîne. Si le caractère dans la chaîne correspond à un caractère dans *trim\_set*, RTRIM le supprime. RTRIM continue de comparer et de supprimer des caractères tant qu'il trouve des caractères correspondants dans *trim\_set*. Il renvoie la chaîne sans caractères correspondants.

### Syntaxe

```
RTRIM( string [, trim_set] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>chaîne</i>	Obligatoire	Toute valeur de chaîne. Transmet les valeurs que vous voulez couper. Vous pouvez entrer l'expression de transformation valide de votre choix. Utilisez des opérateurs pour effectuer des comparaisons ou concaténer des chaînes avant de supprimer des espaces à la fin d'une chaîne.
<i>trim_set</i>	Facultatif	Toute valeur de chaîne. Transmet des caractères que vous voulez supprimer de la fin de la chaîne. Vous pouvez également saisir un littéral de texte. Cependant, vous devez placer les caractères que vous voulez supprimer de la fin de la chaîne entre guillemets simples, par exemple, 'abc'. Si vous omettez la deuxième chaîne, la fonction supprime les espaces de la fin de la première chaîne. RTRIM est sensible à la casse.

### Valeur de retour

Chaîne. Valeurs de chaîne dont les caractères spécifiés dans l'argument *trim\_set* ont été retirés.

NULL si une valeur transmise à la fonction est NULL.

## Exemple

L'expression suivante supprime les caractères 're' des chaînes dans le port LAST\_NAME :

```
RTRIM( LAST_NAME, 're')
```

LAST_NAME	RETURN VALUE
Nelson	Nelson
Page	Pag
Osborne	Osborn
NULL	NULL
Sawyer	Sawy
H. Bender	H. Bend
Steadman	Steadman

RTRIM supprime 'e' de la page, même si 'r' est le premier caractère dans *trim\_set*. Cela est dû au fait que RTRIM recherche, un caractère après l'autre, le jeu de caractères que vous spécifiez dans l'argument *trim\_set*. Si le dernier caractère dans la chaîne correspond au premier caractère dans *trim\_set*, RTRIM le supprime. Si toutefois le dernier caractère dans la chaîne ne correspond pas, RTRIM compare le deuxième caractère dans *trim\_set*. Si le deuxième caractère à partir du dernier dans la chaîne correspond au deuxième caractère dans *trim\_set*, RTRIM le supprime et ainsi de suite. Lorsque le caractère dans la chaîne ne correspond à *trim\_set*, RTRIM renvoie la chaîne et évalue la ligne suivante.

Dans le dernier exemple, le dernier caractère dans Nelson ne correspond à aucun caractère dans l'argument *trim\_set* ; RTRIM renvoie donc la chaîne 'Nelson' et évalue la ligne suivante.

## Conseils d'utilisation de RTRIM

Utilisez RTRIM et LTRIM avec || ou CONCAT pour supprimer les espaces de début et de fin après la concaténation de deux chaînes.

Vous pouvez également supprimer plusieurs jeux de caractères en imbriquant RTRIM. Par exemple, si vous voulez supprimer des espaces de fin et le caractère « t » de la fin de chaque chaîne dans une colonne de noms, vous pouvez créer une expression comme suit :

```
RTRIM( RTRIM( NAMES ), 't' )
```

# SET\_DATE\_PART

Définit une partie d'une valeur Date/Heure sur une valeur que vous indiquez. SET\_DATE\_PART permet de changer les parties suivantes d'une date :

- **Année.** Modifiez l'année en saisissant un entier positif dans l'argument de *valeur*. Utilisez les chaînes de format de l'année : Y, YY, YYY, ou YYYY pour définir l'année. Par exemple, l'expression suivante modifie l'année par 2001 pour toutes les dates dans le port SHIP\_DATE :

```
SET_DATE_PART( SHIP_DATE, 'YY', 2001 )
```

- **Mois.** Changez le mois en saisissant un entier positif entre 1 et 12 (janvier = 1 et décembre = 12) dans l'argument de *valeur*. Utilisez les chaînes de format de mois suivantes : MM, MON, MONTH pour définir le mois. Par exemple, l'expression suivante modifie le mois par octobre pour toutes les dates dans le port SHIP\_DATE :

```
SET_DATE_PART( SHIP_DATE, 'MONTH', 10 )
```

- **Jour.** Changez le jour en saisissant un nombre entier positif entre 1 et 31 (excepté pour les mois inférieurs à 31 jours : Février, avril, juin, septembre et novembre) dans l'argument de *valeur*. Utilisez les chaînes de format de jour (D, DD, DDD, DY et DAY) pour définir le jour. Par exemple, l'expression suivante modifie le jour par 10 pour toutes les dates dans le port SHIP\_DATE :

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

- **Heure.** Modifiez l'heure en saisissant un entier positif entre 0 et 24 (où 0 = minuit, 12 = midi et 24 =minuit) dans l'argument de *valeur*. Utilisez les chaînes de format d'heure (HH, HH12, HH24) pour définir l'heure. Par exemple, l'expression suivante modifie l'heure par 14:00:00 (ou 2:00:00PM) pour toutes les dates dans le port SHIP\_DATE :

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

- **Minute.** Modifiez les minutes en saisissant un entier positif entre 0 et 59 dans l'argument de *valeur*. Utilisez la chaîne de format MI pour définir les minutes. Par exemple, l'expression suivante modifie la minute par 25 pour toutes les dates dans le port SHIP\_DATE :

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

- **Secondes.** Modifiez les secondes en saisissant un entier positif entre 0 et 59 dans l'argument de *valeur*. Utilisez la chaîne de format SS pour définir les secondes. Par exemple, l'expression suivante modifie les secondes par 59 pour toutes les dates dans le port SHIP\_DATE :

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

- **Millisecondes.** Modifiez les millisecondes en saisissant un entier positif entre 0 et 999 dans l'argument de *valeur*. Utilisez la chaîne de format MS pour définir les millisecondes. Par exemple, l'expression suivante modifie les millisecondes par 125 pour toutes les dates dans le port SHIP\_DATE :

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

- **Microsecondes.** Modifiez les microsecondes en saisissant un entier positif entre 1000 et 999999 dans l'argument de *valeur*. Utilisez la chaîne de format US pour définir les microsecondes. Par exemple, l'expression suivante modifie les microsecondes par 12555 pour toutes les dates dans le port SHIP\_DATE :

```
SET_DATE_PART( SHIP_DATE, 'US', 12555 )
```

- **Nanosecondes.** Modifiez les nanosecondes en saisissant un entier positif entre 1000000 et 999999999 dans l'argument de *valeur*. Utilisez la chaîne de format NS pour définir les nanosecondes. Par exemple, l'expression suivante modifie les nanosecondes par 12555555 pour toutes les dates dans le port SHIP\_DATE :

```
SET_DATE_PART( SHIP_DATE, 'NS', 12555555 )
```

## Syntaxe

```
SET_DATE_PART( date, format, value )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>date</i>	Obligatoire	Type de données Date/Heure. Date à modifier. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>format</i>	Obligatoire	Chaîne de format spécifiant la partie de la date à modifier. La chaîne de format n'est pas sensible à la casse.
<i>valeur</i>	Obligatoire	Valeur entière positive attribuée à la partie de la date spécifiée. Le nombre entier doit être une valeur valide pour la partie de la date à modifier. Si vous entrez une valeur incorrecte comme le 30 février, la session échouera.

### Valeur de retour

Date au même format que la date source contenant la partie spécifiée modifiée.

NULL si une valeur transmise à la fonction est NULL.

### Exemples

Les expressions suivantes modifient l'heure par 4PM pour chaque date dans le port DATE\_PROMISED :

```
SET_DATE_PART( DATE_PROMISED, 'HH', 16 )
SET_DATE_PART( DATE_PROMISED, 'HH12', 16 )
SET_DATE_PART( DATE_PROMISED, 'HH24', 16 )
```

DATE_PROMISED	RETURN VALUE
Jan 1 1997 12:15:56AM	Jan 1 1997 4:15:56PM
Feb 13 1997 2:30:01AM	Feb 13 1997 4:30:01PM
Mar 31 1997 5:10:15PM	Mar 31 1997 4:10:15PM
Dec 12 1997 8:07:33AM	Dec 12 1997 4:07:33PM
NULL	NULL

Les expressions suivantes modifient le mois par juin pour les dates dans le port DATE\_PROMISED. Le Service d'intégration de données affiche une erreur lorsque vous tentez de créer une date qui n'existe pas, par exemple en modifiant 31 mars par 31 juin :

```
SET_DATE_PART( DATE_PROMISED, 'MM', 6 )
SET_DATE_PART( DATE_PROMISED, 'MON', 6 )
SET_DATE_PART( DATE_PROMISED, 'MONTH', 6 )
```

DATE_PROMISED	RETURN VALUE
Jan 1 1997 12:15:56AM	Jun 1 1997 12:15:56AM
Feb 13 1997 2:30:01AM	Jun 13 1997 2:30:01AM
Mar 31 1997 5:10:15PM	<i>Error. Integration Service doesn't write row.</i>

DATE_PROMISED	RETURN VALUE
Dec 12 1997 8:07:33AM	Jun 12 1997 8:07:33AM
NULL	NULL

Les expressions suivantes modifient l'année par 2000 pour les dates dans le port 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

## Conseil

Si vous voulez modifier plusieurs parties de date en une seule fois, vous pouvez imbriquer plusieurs fonctions SET\_DATE\_PART dans l'argument de *date*. Par exemple, vous pouvez écrire l'expression suivante pour mettre toutes les dates dans le port DATE\_ENTERED au 1er juillet 1998 :

```
SET_DATE_PART( SET_DATE_PART( SET_DATE_PART( DATE_ENTERED, 'YYYY', 1998), 'MM', 7), 'DD', 1)
```

# SIGN

Indique si une valeur numérique est positive, négative ou égale à 0.

## Syntaxe

```
SIGN( numeric_value )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Obligatoire	Valeur numérique. Transmet les valeurs que vous voulez évaluer. Vous pouvez entrer l'expression de transformation valide de votre choix.

## Valeur de retour

-1 pour les valeurs négatives.

0 pour 0.

1 pour les valeurs positives.

NULL si NULL.

## Exemple

L'expression suivante détermine si le port SALES inclut des valeurs négatives :

```
SIGN( SALES )
```

SALES	RETURN VALUE
100	1
-25.99	-1
0	0
NULL	NULL

# SIN

Renvoie le sinus d'une valeur numérique (exprimé en radians).

## Syntaxe

```
SIN( numeric_value )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Obligatoire	Type de données numérique. Données numériques exprimées en radians (degrés multipliés par pi, divisés par 180). Transmet les valeurs pour lesquelles vous souhaitez calculer le sinus. Vous pouvez entrer l'expression de transformation valide de votre choix. Vous pouvez également utiliser des opérateurs pour convertir une valeur numérique en radians ou effectuer des opérations arithmétiques dans le calcul SIN.

## Valeur de retour

Valeur Double.

NULL si une valeur transmise à la fonction est NULL.

## Exemple

L'expression suivante convertit les valeurs dans le port Degrees en radians puis calcule le sinus pour chaque radian :

```
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

Vous pouvez effectuer des opérations arithmétiques dans les valeurs transmises à SIN avant que la fonction calcule le sinus. Par exemple :

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

# SINH

Renvoie le sinus hyperbolique de la valeur numérique.

## Syntaxe

```
SINH( numeric_value )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Obligatoire	Type de données numérique. Données numériques exprimées en radians (degrés multipliés par pi, divisés par 180). Transmet les valeurs pour lesquelles vous souhaitez calculer le sinus hyperbolique. Vous pouvez entrer l'expression de transformation valide de votre choix.

## Valeur de retour

Valeur Double.

NULL si une valeur transmise à la fonction est NULL.

## Exemple

L'expression suivante renvoie le sinus hyperbolique pour les valeurs dans le port 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

## Conseil

Vous pouvez effectuer des opérations arithmétiques dans les valeurs transmises à SIN avant que la fonction calcule le sinus hyperbolique. Par exemple :

```
SINH ( MEASURES.ARCS / 180 )
```

# SOUNDEX

Code une valeur de chaîne en une chaîne de quatre caractères.

SOUNDEX fonctionne avec les caractères de l'alphabet anglais (A-Z). Cette fonction utilise le premier caractère de la chaîne d'entrée comme premier caractère dans la valeur de retour et code les trois seules consonnes restantes sous forme de chiffres.

SOUNDEX code les caractères conformément à la liste de règles suivantes :

- Utilise le premier caractère dans la *chaîne* comme premier caractère dans la valeur de retour et le code en majuscule. Par exemple, SOUNDEX('John') et SOUNDEX('john') renvoient tous deux 'J500'.
- Code les trois premières seules consonnes suivant le premier caractère dans la *chaîne* et ignore le reste. Par exemple, SOUNDEX('JohnRB') et SOUNDEX('JohnRBCD') renvoient tous deux 'J561'.
- Attribue un code unique aux consonnes de prononciation similaire.

Le tableau suivant présente les instructions de codage SOUNDEX pour les consonnes :

**Tableau 2. Instructions de codage SOUNDEX pour les consonnes**

Code	Consonne
1	B, P, F, V
2	C, S, G, J, K, Q, X, Z
3	D, T
4	L
5	M, N
6	R

- Ignore les caractères A, E, I, O, U, H et W, sauf si l'un d'eux est le premier caractère dans la *chaîne*. Par exemple, SOUNDEX('A123') renvoie 'A000' et SOUNDEX('MAeiouhWC') renvoie 'M000'.
- Si la *chaîne* produit moins de quatre caractères, SOUNDEX remplit la chaîne obtenue par des zéros. Par exemple, SOUNDEX('J') renvoie 'J000'.
- Si la *chaîne* contient un ensemble de consonnes consécutives qui utilisent le même code indiqué dans ["SOUNDEX" à la page 167](#), SOUNDEX code la première occurrence et ignore les autres occurrences dans l'ensemble. Par exemple, SOUNDEX('AbbpdMN') renvoie 'A135'.
- Ignore les numéros dans la *chaîne*. Par exemple, les deux SOUNDEX('Joh12n') et SOUNDEX('1John') renvoient 'J500'.
- Renvoie NULL si la *chaîne* est NULL ou si tous les caractères dans la *chaîne* ne sont pas des lettres de l'anglais alphabet.

## Syntaxe

```
SOUNDEX( string )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>chaîne</i>	Obligatoire	Chaîne de caractères. Transmet la valeur de chaîne que vous voulez coder. Vous pouvez entrer l'expression de transformation valide de votre choix.

## Valeur de retour

Chaîne.

NULL si l'une des conditions suivantes est vraie :

- Si la valeur transmise à la fonction est NULL.
- Aucun caractère dans la *chaîne* n'est une lettre de l'alphabet anglais.
- La *chaîne* est vide.

## Exemple

L'expression suivante code les valeurs dans le port EMPLOYEE\_NAME :

```
SOUNDEX ( EMPLOYEE_NAME )
```

EMPLOYEE_NAME	RETURN VALUE
John	J500
William	W450
jane	J500
joh12n	J500
1abc	A120
NULL	NULL

## SQRT

Renvoie la racine carrée d'une valeur numérique non négative.

### Syntaxe

```
SQRT( numeric_value )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Obligatoire	Valeur numérique positive. Transmet les valeurs pour lesquelles vous souhaitez calculer une racine carrée. Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

Valeur Double.

NULL si une valeur transmise à la fonction est NULL.

### Exemple

L'expression suivante renvoie la racine carrée pour les valeurs dans le port Numbers :

```
SQRT ( NUMBERS )
```

NUMBERS	RETURN VALUE
100	10
-100	<i>Error. Service d'intégration de données does not write row.</i>

NUMBERS	RETURN VALUE
NULL	NULL
60.54	7.78074546557076

La valeur -100 entraîne une erreur, car la fonction SQRT évalue uniquement les valeurs numériques positives. Si vous transmettez une valeur négative ou de caractère, le Service d'intégration de données affiche une erreur d'évaluation de transformation et n'écrit pas la ligne.

Vous pouvez effectuer des opérations arithmétiques dans les valeurs transmises à SQRT avant que le calcul de la racine carrée par la fonction.

## STDDEV

Renvoie l'écart-type des valeurs numériques que vous passez à cette fonction. STDDEV permet d'analyser des données statistiques. Vous pouvez imbriquer une seule autre fonction Agrégation dans STDDEV et la fonction imbriquée doit renvoyer un type de données numérique.

### Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Obligatoire	Types de données numérique. Cette fonction transmet les valeurs pour lesquelles vous souhaitez calculer un écart-type ou les résultats d'une fonction. Vous pouvez entrer l'expression de transformation valide de votre choix. Vous pouvez utiliser des opérateurs pour calculer la moyenne des valeurs dans plusieurs ports.
<i>filter_condition</i>	Facultatif	Limite les lignes dans la recherche. La condition de filtre doit être une valeur numérique ou renvoyer TRUE, FALSE ou NULL. Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

Valeur numérique.

NULL si toutes les valeurs transmises à la fonction sont NULL, ou si aucune ligne n'est sélectionnée (par exemple : la condition de filtre renvoie FALSE ou NULL pour toutes les lignes).

**Remarque:** Si la valeur de retour est décimale avec une précision supérieure à 15, vous pouvez activer la précision élevée pour garantir une précision décimale jusqu'à 28 chiffres.

### Valeurs nulles

Si une seule valeur est nulle, STDDEV l'ignore. Cependant, si toutes les valeurs sont NULL, STDDEV renvoie NULL.

## Grouper par

STDDEV groupe des valeurs en fonction du groupement par ports que vous définissez dans la transformation et renvoie un résultat pour chaque groupe.

S'il n'existe aucun port de regroupement, STDDEV traite toutes les lignes comme un seul groupe et renvoie une valeur.

## Exemples

L'expression suivante calcule l'écart-type de toutes les lignes supérieures à 2000.00 \$ dans le port 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 fonction n'inclut pas les valeurs 1010.90 et 153.88 dans le calcul, car la *filter\_condition* spécifie des ventes supérieures à 2000 \$.

L'expression suivante calcule l'écart-type de toutes les lignes dans le port SALES :

```
STDDEV(SALES)
```

### SALES

2198.0

2198.0

2198.0

2198.0

**RETURN VALUE:** 0

La valeur de retour est de 0, car chaque ligne contient le même nombre (aucun écart-type n'existe). S'il n'existe aucun écart-type, la valeur de retour est de 0.

# SUBSTR

Renvoie une portion d'une chaîne. SUBSTR compte tous les caractères, y compris blancs, en partant du début de la chaîne.

## Syntaxe

```
SUBSTR( string, start [,length] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>chaîne</i>	Obligatoire	Doit être une chaîne de caractères. Transmet les chaînes à rechercher. Vous pouvez entrer l'expression de transformation valide de votre choix. Si vous transmettez une valeur numérique, la fonction la convertit en une chaîne de caractères.
<i>début</i>	Obligatoire	Doit être un nombre entier. Position dans la chaîne où vous voulez démarrer le compte. Vous pouvez entrer l'expression de transformation valide de votre choix. Si la position de départ est un nombre positif, SUBSTR recherche la position de départ par en comptant à partir du début de la chaîne. Si la position de départ est un nombre négatif, SUBSTR recherche la position de départ par en comptant à partir de la fin de la chaîne. Si la position de départ est 0, SUBSTR recherche à partir du premier caractère dans la chaîne.
<i>longueur</i>	Facultatif	Doit être un nombre entier supérieur à 0. Nombre de caractères que vous voulez que SUBSTR renvoie. Vous pouvez entrer l'expression de transformation valide de votre choix. Si vous omettez l'argument de longueur, SUBSTR renvoie tous les caractères compris entre la position de départ et la fin de la chaîne. Si vous transmettez un entier négatif ou 0, la fonction renvoie une chaîne vide. Si vous transmettez un nombre décimal, la fonction arrondit à la valeur entière la plus proche.

## Valeur de retour

Chaîne.

Chaîne vide si vous transmettez une valeur de longueur négative ou 0.

NULL si une valeur transmise à la fonction est NULL.

## Exemples

Les expressions suivantes renvoient l'indicatif régional pour chaque ligne dans le port 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

Les expressions suivantes renvoient le numéro de téléphone sans indicatif régional pour chaque ligne dans le port 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

Vous pouvez également transmettre une valeur de départ négative pour renvoyer le numéro de téléphone pour chaque ligne dans le port Phone. L'expression continue de lire la chaîne source de gauche à droite lors du renvoi du résultat de l'argument *longueur* :

SUBSTR( PHONE, -8, 3 )

PHONE	RETURN VALUE
808-555-0269	555
809-555-3915	555
357-687-6708	687
NULL	NULL

Vous pouvez imbriquer INSTR dans l'argument de *démarrage* ou de *longueur* pour rechercher une chaîne spécifique et renvoyer sa position.

L'expression suivante évalue une chaîne, en commençant par la fin de cette chaîne. L'expression recherche le dernier espace (le plus à droite) dans la chaîne et renvoie tous les caractères qui le précèdent :

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

CUST_NAME	RETURN VALUE
PATRICIA JONES	PATRICIA
MARY ELLEN SHAH	MARY ELLEN

L'expression suivante supprime le caractère « # » d'une chaîne :

```
SUBSTR( CUST_ID, 1, INSTR(CUST_ID, '#')-1 ) || SUBSTR( CUST_ID, INSTR(CUST_ID, '#')+1 )
```

Lorsque l'argument de *longueur* est plus long que la chaîne, SUBSTR renvoie tous les caractères compris entre la position de départ et la fin de la chaîne. Considérons l'exemple suivant :

```
SUBSTR('abcd', 2, 8)
```

La valeur de retour est 'bcd'. Comparez ce résultat à l'exemple suivant :

```
SUBSTR('abcd', -2, 8)
```

La valeur de retour est 'cd'.

## SUM

Renvoie la somme de toutes les valeurs dans le port sélectionné. Vous également pouvez appliquer un filtre pour limiter les lignes que vous lisez pour calculer le total. Vous pouvez imbriquer une seule autre fonction Agrégation dans SUM et la fonction imbriquée doit renvoyer un type de données numérique.

### Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Obligatoire	Type de données numérique. Transmet les valeurs que vous voulez ajouter. Vous pouvez entrer l'expression de transformation valide de votre choix. Vous pouvez utiliser des opérateurs pour ajouter des valeurs dans plusieurs ports.
<i>filter_condition</i>	Facultatif	Limite les lignes dans la recherche. La condition de filtre doit être une valeur numérique ou renvoyer TRUE, FALSE ou NULL. Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

Valeur numérique.

NULL si toutes les valeurs transmises à la fonction sont NULL, ou si aucune ligne n'est sélectionnée (par exemple : la condition de filtre renvoie FALSE ou NULL pour toutes les lignes).

**Remarque:** Si la valeur de retour est décimale avec une précision supérieure à 15, vous pouvez activer la précision élevée pour garantir une précision décimale jusqu'à 28 chiffres.

### Valeurs nulles

Si une seule valeur est nulle, SUM l'ignore. Cependant, si toutes les valeurs transmises à partir du port sont NULL, SUM renvoie NULL.

### Grouper par

SUM regroupe des valeurs en fonction du groupement par ports que vous définissez dans la transformation et renvoie un résultat pour chaque groupe.

S'il n'existe aucun port de groupement, SUM traite toutes les lignes comme un groupe et renvoie une valeur.

### Exemple

L'expression suivante renvoie la somme de toutes les valeurs supérieures à 2000 dans le port Sales :

```
SUM( SALES, SALES > 2000 )
```

#### SALES

2500.0

1900.0

1200.0

NULL

3458.0

4519.0

**RETURN VALUE:** 10477.0

### Conseil

Vous pouvez effectuer des opérations arithmétiques dans les valeurs transmises à SUM avant le calcul du total par la fonction. Par exemple :

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

## SYSTIMESTAMP

Renvoie la date et l'heure actuelle du nœud hébergeant le Service d'intégration de données avec une précision à la nanoseconde. La précision à laquelle vous affichez la date et l'heure dépend de la plate-forme.

La valeur de retour de la fonction varie en fonction du type de configuration de l'argument :

- Lorsque vous configurez l'argument SYSTIMESTAMP comme une variable, le Service d'intégration de données évalue la fonction pour chaque ligne dans la transformation.
- Lorsque vous configurez l'argument SYSTIMESTAMP comme une constante, le Service d'intégration de données évalue la fonction une fois et conserve la valeur pour chaque ligne dans la transformation.

## Syntaxe

```
SYSTIMESTAMP( [format] )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>format</i>	Facultatif	Précision à laquelle vous souhaitez récupérer l'horodatage. Vous pouvez spécifier la précision en secondes (SS), millisecondes (MS), microsecondes (US), ou nanosecondes (NS). Placez la chaîne de format entre des guillemets simples. La chaîne de format n'est pas sensible à la casse. Par exemple, pour afficher la date et l'heure avec une précision des millisecondes, utilisez la syntaxe suivante : SYSTIMESTAMP('MS'). Par défaut, la valeur de la précision est en microsecondes (US).

## Valeur de retour

Horodatage Renvoie la date et l'heure spécifiée pour la précision.

## Exemples

Votre organisation dispose d'un service de commandes en ligne et traite les données en temps réel. Vous pouvez utiliser la fonction SYSTIMESTAMP pour générer une clé primaire pour chaque transaction dans la base de données cible.

Créez une expression de transformation avec les ports et les valeurs suivants :

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 )

Lors de l'exécution, le Service d'intégration de données génère l'heure système avec une précision en microsecondes pour chaque ligne :

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

Revoie la tangente d'une valeur numérique (exprimée en radians).

## Syntaxe

```
TAN( numeric_value )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire/ Facultatif	Description
<i>numeric_value</i>	Obligatoire	Type de données numérique. Données numériques exprimées en radians (degrés multipliés par pi, divisés par 180). Transmet les valeurs numériques pour lesquelles vous souhaitez calculer la tangente. Vous pouvez entrer l'expression de transformation valide de votre choix.

## Valeur de retour

Valeur Double.

NULL si une valeur transmise à la fonction est NULL.

## Exemple

L'expression suivante renvoie la tangente pour toutes les valeurs dans le port 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

Revoie la tangente hyperbolique de la valeur numérique passée à cette fonction.

## Syntaxe

```
TANH( numeric_value )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Obligatoire	Type de données numérique. Données numériques exprimées en radians (degrés multipliés par pi, divisés par 180). Transmet les valeurs numériques pour lesquelles vous souhaitez calculer la tangente hyperbolique. Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

Valeur Double.

NULL si une valeur transmise à la fonction est NULL.

### Exemple

L'expression suivante renvoie la tangente hyperbolique pour les valeurs dans le port 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

### Conseil

Vous pouvez effectuer des opérations arithmétiques dans les valeurs transmises à TANH avant le calcul de la tangente hyperbolique par la fonction. Par exemple :

```
TANH ( ARCS / 360 )
```

## TO\_BIGINT

Convertit une chaîne ou une valeur numérique en valeur de type bigint. La syntaxe TO\_BIGINT comporte un argument facultatif que vous pouvez choisir pour arrondir le nombre à l'entier le plus proche ou tronquer la partie décimale. TO\_BIGINT ignore les espaces au début.

### Syntaxe

```
TO_BIGINT( value [, flag] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>valeur</i>	Obligatoire	Type de données de chaîne ou numérique. Transmet la valeur à convertir en valeur de type bigint. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>indicateur</i>	Facultatif	Spécifie si la partie décimale doit être tronquée ou arrondie. L'indicateur doit être un littéral entier ou les constantes TRUE ou FALSE. TO_BIGINT tronque la partie décimale lorsque l'indicateur est TRUE ou un nombre différent de 0. TO_BIGINT arrondit la valeur à l'entier le plus proche si l'indicateur est FALSE ou 0, ou si vous omettez cet argument. L'indicateur n'est pas défini par défaut.

### Valeur de retour

Bigint.

NULL si la valeur transmise à la fonction est NULL.

Si la valeur passée à la fonction contient des données qui ne sont pas valides pour une valeur bigint, le service d'intégration de données marque la ligne comme une ligne d'erreur ou fait échouer le mappage.

### Exemples

Les expressions suivantes utilisent des valeurs du port IN\_TAX :

```
TO_BIGINT( IN_TAX, TRUE )
```

IN_TAX	RETURN VALUE
'7245176201123435.6789'	7245176201123435
'7245176201123435.2'	7245176201123435
'7245176201123435.2.48'	7245176201123435
NULL	NULL
'A12.3Grove'	<i>Error. Integration Service skips this row.</i>
' 176201123435.87'	176201123435
'-7245176201123435.2'	-7245176201123435
'-7245176201123435.23'	-7245176201123435

IN_TAX	RETURN VALUE
-9223372036854775806.9	-9223372036854775806
9223372036854775806.9	9223372036854775806
TO_BIGINT( IN_TAX )	
IN_TAX	RETURN VALUE
'7245176201123435.6789'	7245176201123436
'7245176201123435.2'	7245176201123435
'7245176201123435.348'	7245176201123435
NULL	NULL
'A12.3Grove'	<i>Error. Integration Service skips this row.</i>
' 176201123435.87'	176201123436
'-7245176201123435.6789'	-7245176201123436
'-7245176201123435.23'	-7245176201123435
-9223372036854775806.9	-9223372036854775807
9223372036854775806.9	9223372036854775807

## TO\_CHAR (Dates)

Convertit les dates en chaînes de caractère. TO\_CHAR convertit également des valeurs numériques en chaînes. Vous pouvez convertir la date dans le format de votre choix à l'aide de chaînes de format TO\_CHAR.

### Syntaxe

```
TO_CHAR( date [,format] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>date</i>	Obligatoire	Type de données Date/Heure. Transmet les valeurs de date à convertir en chaînes de caractères. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>format</i>	Facultatif	Entrez une chaîne de format TO_CHAR valide. La chaîne de format définit le format de la valeur de retour, pas le format pour les valeurs dans l'argument de date. Si vous omettez la chaîne de format, la fonction renvoie une chaîne basée sur le format de date spécifié dans la configuration du mappage.

### Valeur de retour

Chaîne.

NULL si une valeur transmise à la fonction est NULL.

### Exemples

L'expression suivante convertit les dates dans le port DATE\_PROMISED en texte au format 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 vous omettez l'argument de *format*, TO\_CHAR renvoie une chaîne au format de date spécifié dans la configuration du mappage, par défaut : MM/DD/YYYY HH24:MI:SS.US :

```
TO_CHAR( DATE_PROMISED )
```

DATE_PROMISED	RETURN VALUE
Apr 1 1998 12:00:10AM	'04/01/1998 00:00:10.000000'
Feb 22 1998 01:31:10PM	'02/22/1998 13:31:10.000000'
Oct 24 1998 02:12:30PM	'10/24/1998 14:12:30.000000'
NULL	NULL

Les expressions suivantes renvoient le jour de la semaine pour chaque date dans un port :

```
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

L'expression suivante renvoie le jour du mois pour chaque date dans un port :

```
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

L'expression suivante renvoie le jour de l'année pour chaque date dans un port :

```
TO_CHAR( DATE_PROMISED, 'DDD' )
```

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10AM	'091'
02-22-1997 01:31:10PM	'053'
10-24-1997 02:12:30PM	'297'
NULL	NULL

Les expressions suivantes renvoient l'heure du jour pour chaque date dans un port :

```
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

L'expression suivante convertit les valeurs de date en valeurs MJD exprimées sous forme de chaînes :

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

SHIP_DATE	RETURN VALUE
Dec 31 1999 03:59:59PM	2451544
Jan 1 1900 01:02:03AM	2415021

L'expression suivante convertit les dates en chaînes au format MM/DD/YY :

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

SHIP_DATE	RETURN VALUE
12/31/1999 01:02:03AM	12/31/99
09/15/1996 03:59:59PM	09/15/96
05/17/2003 12:13:14AM	05/17/03

Vous pouvez également utiliser la chaîne de format SSSSS dans une expression TO\_CHAR. Par exemple, l'expression suivante convertit les dates dans le port SHIP\_DATE en chaînes représentant le nombre total de secondes depuis minuit :

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

SHIP_DATE	RETURN_VALUE
12/31/1999 01:02:03AM	3783
09/15/1996 03:59:59PM	86399

Dans les expressions TO\_CHAR, la chaîne de format YY produit les mêmes résultats que la chaîne de format RR.

L'expression suivante convertit les dates en chaînes au format 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

L'expression suivante renvoie la semaine du mois pour chaque date dans un port :

```
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

L'expression suivante renvoie la semaine de l'année pour chaque date dans un port :

```
TO_CHAR( DATE_PROMISED, 'WW' )
```

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10PM	'18'
02-22-1997 01:31:10AM	'08'
10-24-1997 02:12:30AM	'43'
NULL	NULL

## Conseil

Vous pouvez associer TO\_CHAR et TO\_DATE pour convertir une valeur numérique pour un mois en valeur de texte pour un mois à l'aide d'une fonction telle que la suivante :

```
TO_CHAR( TO_DATE( numeric_month, 'MM' ), 'MONTH' )
```

# TO\_CHAR (Nombres)

Convertit des valeurs numériques en chaînes de texte. TO\_CHAR convertit également des dates en chaînes.

TO\_CHAR convertit des valeurs doubles en chaînes de texte comme suit :

- Convertit des valeurs doubles contenant jusqu'à 16 chiffres en chaînes et fournit une précision allant jusqu'à 15 chiffres. Si vous transmettez un nombre comportant plus de 15 chiffres, l'expression TO\_CHAR l'arrondit en fonction du seizième chiffre et renvoie la représentation de chaîne du nombre en notation scientifique. Par exemple, la valeur double 1234567890123456 est convertie en valeur de chaîne « 1,23456789012346e + 015 ».
- Renvoie une notation décimale pour des nombres compris dans les plages (-1e16, -1e-16] et [1e-16, 1e16). L'expression TO\_CHAR renvoie une notation scientifique pour les nombres hors de ces plages. Par exemple, la valeur double 10842764968208837340 est convertie en valeur de chaîne « 1,08427649682088e + 019 ».

L'expression TO\_CHAR convertit des valeurs décimales en chaînes de texte comme suit :

- En mode de précision élevée, l'expression TO\_CHAR convertit les valeurs décimales jusqu'à 28 chiffres en chaînes. Si vous transmettez une valeur décimale de plus de 28 chiffres, l'expression TO\_CHAR renvoie une notation scientifique pour les nombres de plus de 28 chiffres.
- En mode de précision faible, l'expression TO\_CHAR traite les valeurs décimales comme des valeurs doubles.

## Syntaxe

```
TO_CHAR( numeric_value )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Requis	Type de données numériques. Valeur numérique à convertir en chaîne. Vous pouvez entrer l'expression de transformation valide de votre choix.

## Valeur de retour

Chaîne.

NULL si une valeur transmise à la fonction est NULL.

## Exemple de conversion double

L'expression suivante convertit les valeurs doubles du port SALES en chaînes :

```
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

## Exemple de conversion décimale

En mode de précision élevée, l'expression suivante convertit les valeurs décimales du port SALES en chaînes :

```
TO_CHAR( SALES )
```

SALES	RETURN VALUE
2378964536789761	'2378964536789761'
1234567890123456789012345679	'1234567890123456789012345679'
1.234578945469649345876123456	'1.234578945469649345876123456'
0.999999999999999999999999999999	'0.999999999999999999999999999999'
12345678901234567890123456799 (greater than 28)	'1.23456789012346e+028'

# TO\_DATE

Convertit une chaîne de caractères en type de donnée Date/Heure. Utilisez des chaînes de format TO\_DATE pour spécifier le format des chaînes source.

Le port de sortie doit être Date/Heure pour des expressions TO\_DATE.

Si vous convertissez des années à deux chiffres à l'aide de TO\_DATE, utilisez la chaîne de format RR ou YY. N'utilisez pas la chaîne de format YYYY.

## Syntaxe

```
TO_DATE( string [, format] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>chaîne</i>	Obligatoire	Doit être un type de données String. Transmet les valeurs que vous voulez convertir en dates. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>format</i>	Facultatif	Entrez une chaîne de format TO_DATE valide. La chaîne de format doit correspondre aux parties de l'argument <i>chaîne</i> . Par exemple, si vous passez la chaîne 'Mar 15 1998 12:43:10AM', vous devez utiliser la chaîne de format 'MON DD YYYY HH12:MI:SSAM'. Si vous omettez la chaîne de format, la valeur de chaîne doit être dans le format de date spécifié dans la session.

## Valeur de retour

Date.

TO\_DATE renvoie toujours une date et une heure. Si vous transmettez une chaîne qui n'a pas de valeur d'heure, la date renvoyée inclura toujours l'heure 00:00:00.000000000. Vous pouvez mapper les résultats de cette fonction vers une colonne cible de votre choix avec un type de données Date/Heure. Si la précision de la colonne cible est inférieure aux nanosecondes, le Service d'intégration de données tronque la valeur date/heure pour faire correspondre la précision de la colonne cible lors de l'écriture des valeurs de date/heure dans la cible.

NULL si vous transmettez une valeur nulle à cette fonction.

**Avertissement:** Le format de la chaîne TO\_DATE doit correspondre à la chaîne de format, y compris les séparateurs de date. Si ce n'est pas le cas, le Service d'intégration de données peut renvoyer des valeurs non précises ou ignorer l'enregistrement.

## Exemples

L'expression suivante renvoie des valeurs date pour les chaînes dans le port DATE\_PROMISED. TO\_DATE renvoie toujours une date et une heure. Si vous transmettez une chaîne qui n'a pas de valeur d'heure, la date renvoyée inclura toujours l'heure 00:00:00.000000000. Si vous exécutez un mappage au vingtième siècle, il s'agira du 19e siècle. Dans cet exemple, l'année en cours sur le nœud exécutant le Service d'intégration de données est 1998. Le format date/heure pour la colonne cible est MON DD YY HH24:MI SS ; le Service d'intégration de données tronque donc la valeur date/heure en secondes lors de l'écriture dans la cible :

```
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

L'expression suivante renvoie les valeurs de date et heure pour les chaînes dans le port DATE\_PROMISED. Si vous transmettez une chaîne qui n'a pas de valeur d'heure, le Service d'intégration de données renvoie une

erreur. Si vous exécutez un mappage au vingtième siècle, il s'agira du 19e siècle. L'année en cours sur le nœud exécutant le Service d'intégration de données est 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

L'expression suivante convertit les chaînes dans le port SHIP\_DATE\_MJD\_STRING en valeurs de date :

```
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

La chaîne de format J ne contenant pas la portion d'heure d'une date, l'heure des valeurs de retour est définie à 00:00:00.000000000.

L'expression suivante convertit une chaîne en un format d'année à quatre chiffres. L'année en cours est 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

L'expression suivante convertit une chaîne en un format d'année à quatre chiffres. L'année en cours est 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

**Remarque:** Pour la deuxième ligne, RR renvoie l'année 2005 et YY renvoie l'année 1905.

L'expression suivante convertit une chaîne en un format d'année à quatre chiffres. L'année en cours est 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

L'expression suivante convertit une chaîne en un format d'année à quatre chiffres. L'année en cours est 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

L'expression suivante convertit les chaînes comprenant les secondes à partir de minuit en valeurs de date :

```
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 la cible accepte différents formats de date, utilisez TO\_DATE et IS\_DATE avec la fonction DECODE pour évaluer les formats acceptables. Par exemple :

```
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' ) )
```

Vous pouvez associer TO\_CHAR et TO\_DATE pour convertir une valeur numérique pour un mois en valeur de texte pour un mois à l'aide d'une fonction telle que la suivante :

```
TO_CHAR( TO_DATE( numeric_month, 'MM' ), 'MONTH' )
```

## LIENS CONNEXES :

- ["Règles et directives pour les chaînes de format de date" à la page 40](#)

# TO\_DECIMAL

Convertit une chaîne ou une valeur numérique en valeur décimale. TO\_DECIMAL ignore les espaces du début.

## Syntaxe

```
TO_DECIMAL( value [, scale] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>valeur</i>	Obligatoire	Doit être un type de données chaîne ou numérique. Transmet les valeurs que vous voulez convertir en décimales. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>échelle</i>	Facultatif	Doit être un littéral entier compris entre 0 et 28, inclus. Spécifie le nombre de chiffres autorisés après le point décimal. Si vous omettez cet argument, la fonction renvoie une valeur ayant la même échelle que la valeur d'entrée.

## Valeur de retour

Décimal dont la précision et l'échelle sont compris entre 0 et 28, inclus.

NULL si une valeur transmise à la fonction est NULL.

Si la valeur passée à la fonction contient des données qui ne sont pas valides pour une valeur décimal, le service d'intégration de données marque la ligne comme une ligne d'erreur ou fait échouer le mappage.

**Remarque:** Si la valeur de retour est décimale avec une précision supérieure à 15, vous pouvez activer la précision élevée pour garantir une précision décimale jusqu'à 28 chiffres.

## Exemple

Cette expression utilise les valeurs du port IN\_TAX. Le type de données est décimal avec une précision de 10 et une échelle de 3 :

```
TO_DECIMAL( IN_TAX, 3 )
```

IN_TAX	RETURN VALUE
'15.6789'	15.679
'60.2'	60.200
'118.348'	118.348
NULL	NULL

IN_TAX	RETURN VALUE
'A12.3Grove'	<i>Error. Integration Service skips this row.</i>
'711A1'	711

## TO\_FLOAT

Convertit une chaîne ou une valeur numérique en nombre à virgule flottante de précision double (type de données double). TO\_FLOAT ignore les espaces du début.

### Syntaxe

```
TO_FLOAT( value )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>valeur</i>	Obligatoire	Doit être un type de données chaîne ou numérique. Transmet les valeurs que vous voulez convertir en valeurs doubles. Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

Valeur Double.

NULL si une valeur transmise à cette fonction est NULL.

Si la valeur passée à la fonction contient des données qui ne sont pas valides pour une valeur float, le service d'intégration de données marque la ligne comme une ligne d'erreur ou fait échouer le mappage.

### Exemple

Cette expression utilise les valeurs du port IN\_TAX :

```
TO_FLOAT( IN_TAX )
```

IN_TAX	RETURN VALUE
'15.6789'	15.6789
'60.2'	60.2
'118.348'	118.348
NULL	NULL
'A12.3Grove'	<i>Error. Integration Service skips this row.</i>

# TO\_INTEGER

Convertit une chaîne ou une valeur numérique en nombre entier. La syntaxe TO\_INTEGER contient un argument facultatif que vous pouvez choisir pour arrondir le nombre à l'entier le plus proche ou tronquer la partie décimale. TO\_INTEGER ignore les espaces du début.

## Syntaxe

```
TO_INTEGER( value [, flag] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>valeur</i>	Obligatoire	Type de données de chaîne ou numérique. Transmet la valeur à convertir en nombre entier. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>indicateur</i>	Facultatif	Spécifie si la partie décimale doit être tronquée ou arrondie. L'indicateur doit être un littéral entier ou les constantes TRUE ou FALSE. TO_INTEGER tronque la partie décimale lorsque l'indicateur est TRUE ou un nombre autre que 0. TO_INTEGER arrondit la valeur à l'entier le plus proche si l'indicateur est FALSE, 0 ou si vous omettez cet argument.

## Valeur de retour

Entier.

NULL si la valeur transmise à la fonction est NULL.

Si la valeur passée à la fonction contient des données qui ne sont pas valides pour une valeur integer, le service d'intégration de données marque la ligne comme une ligne d'erreur ou fait échouer le mappage.

## Exemples

Les expressions suivantes utilisent des valeurs du port IN\_TAX. Le Service d'intégration de données affiche une erreur lorsque la conversion entraîne un dépassement numérique :

```
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'	<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'	<i>Error. Integration Service skips this row.</i>
' 123.87'	124
'-15.6789'	-16
'-15.23'	-15

## TRUNC (Dates)

Tronque des dates à une année, un mois, un jour, une heure, une minute, une seconde, une milliseconde ou une microseconde spécifique. Vous pouvez également utiliser TRUNC pour tronquer des chiffres.

Vous pouvez tronquer les parties de date suivantes :

- Année.** Si vous tronquez la partie année de la date, la fonction renvoie JAN 1 de l'année d'entrée avec l'heure définie sur 00:00:00.000000000. Par exemple, l'expression suivante renvoie 1/1/1997 00:00:00.000000000 :
 

```
TRUNC(12/1/1997 3:10:15, 'YY')
```
- Mois.** Si vous tronquez la partie mois d'une date, la fonction renvoie le premier jour du mois avec l'heure définie sur 00:00:00.000000000. Par exemple, l'expression suivante renvoie 01/04/1997 00:00:00.000000000 :
 

```
TRUNC(4/15/1997 12:15:00, 'MM')
```
- Jour.** Si vous tronquez la partie jour d'une date, la fonction renvoie la date avec l'heure définie sur 00:00:00.000000000. Par exemple, l'expression suivante renvoie 13/06/1997 00:00:00.000000000 :
 

```
TRUNC(6/13/1997 2:30:45, 'DD')
```
- Heure.** Si vous tronquez la partie heure d'une date, la fonction renvoie la date avec les minutes, les secondes et les sous-secondes définies sur 0. Par exemple, l'expression suivante renvoie 01/04/1997 11:00:00.000000000 :
 

```
TRUNC(4/1/1997 11:29:35, 'HH')
```

- **Minute.** Si vous tronquez la partie minute d'une date, la fonction renvoie la date avec les secondes et les sous-secondes définies sur 0. Par exemple, l'expression suivante renvoie 22/05/1997 10:15:00.000000000 :  

```
TRUNC(5/22/1997 10:15:29, 'MI')
```
- **Seconde.** Si vous tronquez la deuxième partie d'une date, la fonction renvoie la date avec les millisecondes définies sur 0. Par exemple, l'expression suivante renvoie 22/05/1997 10:15:29.000000000 :  

```
TRUNC(5/22/1997 10:15:29.135, 'SS')
```
- **Milliseconde.** Si vous tronquez la partie milliseconde d'une date, la fonction renvoie la date avec les microsecondes définies sur 0. Par exemple, l'expression suivante renvoie 22/05/1997 10:15:30.135000000 :  

```
TRUNC(5/22/1997 10:15:30.135235, 'MS')
```
- **Microseconde.** Si vous tronquez la partie microseconde d'une date, la fonction renvoie la date avec les nanosecondes définies sur 0. Par exemple, l'expression suivante renvoie 22/05/1997 10:15:30.135235000 :  

```
TRUNC(5/22/1997 10:15:29.135235478, 'US')
```

## Syntaxe

```
TRUNC( date [,format] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>date</i>	Obligatoire	Type de données Date/Heure. Valeurs de date que vous voulez tronquer. Vous pouvez entrer une expression de transformation valide qui renvoie une date.
<i>format</i>	Facultatif	Entrez une chaîne de format valide. La chaîne de format n'est pas sensible à la casse. Si vous omettez la chaîne de format, la fonction tronque la partie heure de la date et la définit sur 00:00:00.000000000.

## Valeur de retour

Date.

NULL si une valeur transmise à la fonction est NULL.

## Exemples

Les expressions suivantes tronquent la partie année des dates dans le port 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

DATE_SHIPPED	RETURN VALUE
Dec 20 1998 3:29:55PM	Jan 1 1998 12:00:00.000000000
NULL	NULL

Les expressions suivantes tronquent la partie mois de chaque date dans le port 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

Les expressions suivantes tronquent la partie jour de chaque date dans le port 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

Les expressions suivantes tronquent la partie heure de chaque date dans le port 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

DATE_SHIPPED	RETURN VALUE
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

L'expression suivante tronque la partie minute de chaque date dans le port DATE\_SHIPPED :

```
TRUNC( DATE_SHIPPED, 'MI' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 15 1998 2:10:00.000000000AM
Apr 19 1998 1:31:20PM	Apr 19 1998 1:31:00.000000000PM
Jun 20 1998 3:50:04AM	Jun 20 1998 3:50:00.000000000AM
Dec 20 1998 3:29:55PM	Dec 20 1998 3:29:00.000000000PM
Dec 31 1998 11:59:59PM	Dec 31 1998 11:59:00.000000000PM
NULL	NULL

## TRUNC (Nombres)

Tronque des nombres à un chiffre spécifique. Vous pouvez également utiliser TRUNC pour tronquer des dates.

### Syntaxe

```
TRUNC( numeric_value [, precision] )
```

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire / Facultatif	Description
<i>numeric_value</i>	Obligatoire	Type de données numérique. Transmet les valeurs que vous voulez tronquer. Vous pouvez entrer n'importe quelle expression de transformation valide qui renvoie un type de données numérique.
<i>précision</i>	Facultatif	Peut être un entier positif ou négatif. Vous pouvez saisir toute expression de transformation valide qui renvoie un nombre entier. Le nombre entier spécifie le nombre de chiffres à tronquer.

Si la *précision* est un entier positif, TRUNC renvoie *numeric\_value* avec le nombre de décimales spécifié dans la *précision*. Si la *précision* est un entier négatif, TRUNC remplace les chiffres spécifiés à gauche du point



PRICE	RETURN VALUE
-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

Convertit les caractères minuscules de la chaîne en majuscule.

### Syntaxe

`UPPER( string )`

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>chaîne</i>	Obligatoire	Type de données Chaîne. Transmet les valeurs que vous voulez modifier en texte en majuscule. Vous pouvez entrer l'expression de transformation valide de votre choix.

### Valeur de retour

Chaîne en majuscule. Si les données contiennent des caractères multioctets, la valeur de retour dépend de la page de code et du mode mouvement de données du Service d'intégration de données.

NULL si une valeur transmise à la fonction est NULL.

## Exemple

L'expression suivante change tous les noms dans le port FIRST\_NAME en majuscule :

```
UPPER( FIRST_NAME )
```

FIRST_NAME	RETURN VALUE
Ramona	RAMONA
NULL	NULL
THOMAS	THOMAS
PierRe	PIERRE
Bernice	BERNICE

## UUID4

Revoie une valeur binaire de 16 octets générée aléatoirement, conforme à la variante 4 de la spécification UUID décrite dans la norme RFC 4122. UUID4 ne prend pas d'argument.

### Syntaxe

```
UUID4()
```

### Valeur de retour

Binaire.

UUID4 ne renvoie jamais de valeur Null ou d'erreur.

## UUID\_UNPARSE

Convertit une valeur binaire de 16 octets en représentation de chaîne de 36 caractères, comme indiqué dans la norme RFC 4122.

### Syntaxe

```
UUID_UNPARSE( binary )
```

Le tableau suivant décrit l'argument de cette commande :

Argument	Obligatoire / Facultatif	Description
<i>binnaire</i>	Obligatoire	Type de données Binary. Toute valeur binaire de 16 octets que vous souhaitez convertir en chaîne de 36 caractères.

## Valeur de retour

Chaîne de 36 caractères.

Renvoie une valeur Null si l'argument est Null et renvoie une erreur si l'argument n'est pas une valeur binaire de 16 octets.

## Exemple

L'expression suivante peut renvoyer une valeur de 6948DF80-14BD-4E04-8842-7668D9C001F5 :

```
UUID_UNPARSE (UUID4 ( ) )
```

# VARIANCE

Renvoie la variance d'une valeur que vous passez à la fonction. VARIANCE est utilisée pour analyser des données statistiques. Vous ne pouvez imbriquer qu'une seule autre fonction Agrégation dans VARIANCE et la fonction imbriquée doit renvoyer un type de données numérique.

## Syntaxe

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

Le tableau suivant décrit les arguments de cette commande:

Argument	Obligatoire/ Facultatif	Description
<i>numeric_value</i>	Obligatoire	Type de données numérique. Transmet les valeurs pour lesquelles vous souhaitez calculer une variance. Vous pouvez entrer l'expression de transformation valide de votre choix.
<i>filter_condition</i>	Facultatif	Limite les lignes dans la recherche. La condition de filtre doit être une valeur numérique ou renvoyer TRUE, FALSE ou NULL. Vous pouvez entrer l'expression de transformation valide de votre choix.

## Valeur de retour

Valeur Double.

NULL si toutes les valeurs transmises à la fonction sont NULL, ou si aucune ligne n'est sélectionnée (par exemple, *filter\_condition* renvoie FALSE ou NULL pour toutes les lignes).

## Valeurs nulles

Si une seule valeur est nulle, VARIANCE l'ignore. Cependant, si toutes les valeurs transmises à la fonction sont NULL, ou si aucune ligne n'est sélectionnée, VARIANCE renvoie NULL.

## Grouper par

VARIANCE groupe les valeurs en fonction de ports de regroupement que vous définissez dans la transformation et renvoie un résultat pour chaque groupe.

S'il n'existe aucun port de regroupement, VARIANCE traite toutes les lignes comme un seul groupe et renvoie une seule valeur.

## Exemple

L'expression suivante calcule la variance de toutes les lignes dans le port TOTAL\_SALES :

```
VARIANCE( TOTAL_SALES )
```

```
TOTAL_SALES
```

```
2198.0
```

```
2256.0
```

```
3001.0
```

```
NULL
```

```
8953.0
```

```
RETURN VALUE: 10592444.6666667
```

# INDEX

## A

Algorithme standard de cryptage avancé  
description [55](#)  
AND  
mot réservé [16](#)  
année 2000  
dates [31](#)  
arithmétique  
valeurs date/heure [42](#)  
arrondi  
dates [152](#)  
numéros [156](#)  
ASCII  
conversion de caractères en valeurs ASCII [57](#)  
conversion de valeurs ASCII [62](#)  
conversion en valeurs Unicode [63](#)  
fonction CHR [62](#)

## B

bases de données relationnelles  
dates [33](#)  
bigint  
conversion de valeurs en [178](#)

## C

calcul de division  
renvoi du reste [129](#)  
calendrier grégorien  
dans les fonctions de date [31](#)  
calendriers  
types de date pris en charge [31](#)  
capitalisation  
chaînes [94](#), [110](#), [198](#)  
caractères  
ajout de chaînes [111](#), [159](#)  
capitalisation [94](#), [110](#), [198](#)  
caractères ASCII [57](#), [62](#)  
Caractères Unicode [57](#), [62](#), [63](#)  
codage [121](#), [167](#)  
compte [172](#)  
remplacement d'un [145](#)  
remplacement de plusieurs [148](#)  
renvoi du nombre [108](#)  
suppression dans des chaînes [113](#), [160](#)  
casse  
conversion en majuscule [198](#)  
chaîne de format J  
à utiliser avec TO\_CHAR [37](#)  
utilisation avec IS\_DATE [41](#)  
utilisation avec TO\_DATE [41](#)

chaîne de format SSSSS  
à utiliser avec TO\_CHAR [38](#)  
utilisation avec IS\_DATE [42](#)  
utilisation avec TO\_DATE [42](#)  
chaîne de format YY  
à utiliser avec TO\_CHAR [38](#)  
différence entre RR et YY [32](#)  
utilisation avec IS\_DATE [41](#)  
utilisation avec TO\_DATE [41](#)  
chaîne du format RR  
à utiliser avec TO\_CHAR [38](#)  
description [32](#)  
différence entre YY et RR [32](#)  
utilisation avec IS\_DATE [41](#)  
utilisation avec TO\_DATE [41](#)  
chaînes  
ajout d'espaces blancs [111](#)  
ajout de caractères [111](#)  
capitalisation [94](#), [110](#), [198](#)  
concaténation [25](#), [64](#)  
conversion de chaînes de caractères en dates [186](#)  
conversion de dates en caractères [180](#)  
conversion de longueur [159](#)  
conversion de valeurs numériques en chaînes de texte [185](#)  
jeu de caractères [95](#)  
nombre de caractères [108](#)  
remplacement d'un caractère [145](#)  
remplacement de plusieurs caractères [148](#)  
renvoi d'une partie [172](#)  
suppression d'espaces [113](#)  
suppression de caractères [113](#)  
suppression des espaces et des caractères [160](#)  
chaînes de caractères  
conversion à partir de dates [180](#)  
conversion en dates [186](#)  
chaînes de format  
correspondance [40](#)  
Date ordinale [35](#), [38](#)  
Date ordinale modifiée [35](#), [38](#)  
dates [34](#)  
définition [30](#)  
fonction IS\_DATE [38](#)  
fonction TO\_CHAR [35](#)  
fonction TO\_DATE [38](#)  
chaînes de texte  
conversion de valeurs numériques [185](#)  
chaînes vides  
test de [108](#)  
codage  
caractères [121](#), [167](#)  
fonction ENC\_BASE64 [80](#)  
commentaires  
ajout aux expressions [15](#)  
composants du langage de transformation  
présentation [12](#)

- compression
  - compression de données [64](#)
  - décompression des données [80](#)
- concaténation
  - chaînes [25](#), [64](#)
- conditions de filtre
  - fonctions agrégation [45](#)
  - valeurs nulles [22](#)
- constante DD\_DELETE
  - description [18](#)
  - exemple de stratégie de mise à jour [18](#)
  - mot réservé [16](#)
- constante DD\_INSERT
  - description [19](#)
  - exemple de stratégie de mise à jour [19](#)
  - mot réservé [16](#)
- constante DD\_REJECT
  - description [19](#)
  - exemple de stratégie de mise à jour [19](#)
  - mot réservé [16](#)
- constante DD\_UPDATE
  - description [20](#)
  - exemple de stratégie de mise à jour [20](#)
  - mot réservé [16](#)
- constante FALSE
  - description [20](#)
  - mot réservé [16](#)
- constante NULL
  - description [21](#)
  - mot réservé [16](#)
- constante TRUE
  - description [22](#)
  - mot réservé [16](#)
- constantes
  - DD\_INSERT [19](#)
  - DD\_REJECT [19](#)
  - DD\_UPDATE [20](#)
  - description [12](#)
  - FALSE [20](#)
  - NUL [21](#)
  - TRUE [22](#)
- contrainte de clé primaire
  - valeurs nulles [21](#)
- conversion
  - chaînes de date [32](#)
- conversion de chaîne
  - dates [32](#)
- cosinus
  - calcul [66](#)
  - calcul du cosinus hyperbolique [67](#)
- cryptage
  - fonction AES\_ENCRYPT [55](#)
  - utilisation de l'algorithme standard de cryptage avancé [55](#)

## D

- Date ordinaire
  - chaîne de format [35](#), [38](#)
- Date ordinaire modifiée
  - chaîne de format [35](#), [38](#)
- dates
  - année 2000 [31](#)
  - arrondi [152](#)
  - bases de données relationnelles [33](#)
  - chaînes de format [34](#)
  - conversion de chaînes de caractères [180](#)
  - exécution de l'arithmétique [42](#)

- dates (*a continué*)
  - fichiers plats [33](#)
  - fonctions [47](#)
  - format date/heure par défaut [33](#)
  - Ordinal modifié [31](#)
  - Ordinal [31](#)
  - présentation [30](#)
  - troncation [193](#)
- dates ordinales
  - dans les fonctions de date [31](#)
- décodage
  - fonction DEC\_BASE64 [77](#)
- décryptage
  - fonction AES\_DECRYPT [55](#)

## E

- écart-type
  - retour [170](#)
- espaces
  - éviter dans les lignes [103](#)
  - suppression avec DD\_REJECT [19](#)
- expressions
  - ajout de commentaires [15](#)
  - conditionnelles [20](#)
  - présentation [12](#)
  - syntaxe [13](#)
  - utilisation des opérateurs [23](#)
- expressions de transformation
  - contraintes Null [21](#)
  - présentation [12](#)
- expressions imbriquées
  - opérateurs [23](#)

## F

- fichiers plats
  - dates [33](#)
- fonction ABORT
  - description [50](#)
- fonction ABS
  - description [51](#)
- fonction ADD\_TO\_DATE
  - description [52](#)
- fonction AES\_DECRYPT
  - description [55](#)
- fonction AES\_ENCRYPT
  - description [55](#)
- Fonction ANY
  - description [56](#)
- Fonction ASCII
  - description [57](#)
- fonction AVG
  - description [58](#)
- fonction CEIL
  - description [60](#)
- fonction CHOOSE
  - description [61](#)
- fonction CHR
  - description [62](#)
  - insertion de guillemets simples [14](#), [62](#)
- fonction CHRCODE
  - description [63](#)
- Fonction COMPRESS
  - description [64](#)

fonction CONCAT  
   description [64](#)  
   insertion de guillemets simples à l'aide de [64](#)

fonction CONVERT\_BASE  
   description [66](#)

fonction COS  
   description [66](#)

fonction COSH  
   description [67](#)

fonction COUNT  
   description [68](#)

fonction CRC32  
   description [71](#)

fonction CUME  
   description [71](#)

fonction DATE\_COMPARE  
   description [73](#)

fonction DATE\_DIFF  
   description [74](#)

fonction DEC\_BASE64  
   description [77](#)

fonction DECODE  
   description [77](#)  
   internationalisation [13](#)

Fonction DECOMPRESS  
   description [80](#)

fonction ENC\_BASE64  
   description [80](#)

fonction ERROR  
   description [81](#)  
   valeur par défaut [81](#)

fonction EXP  
   description [82](#)

fonction FIRST  
   description [83](#)

fonction FLOOR  
   description [84](#)

fonction FLOOR (expressions)  
   description [84](#)

fonction FV  
   description [85](#)

fonction GET\_DATE\_PART  
   description [86](#)

fonction GREATEST  
   description [88](#)

fonction IIF  
   description [89](#)  
   internationalisation [13](#)

fonction IN  
   description [92](#)

fonction INDEXOF  
   description [93](#)

fonction INITCAP  
   description [94](#)  
   internationalisation [13](#)

fonction INSTR  
   description [95](#)

fonction IS\_DATE  
   chaînes de format [38](#)  
   description [99](#)

fonction IS\_NUMBER  
   description [101](#)

fonction IS\_SPACES  
   description [103](#)

fonction ISNULL  
   description [98](#)

fonction LAST  
   description [104](#)

fonction LAST\_DAY  
   description [105](#)

fonction LEAST  
   description [107](#)

fonction LENGTH  
   description [108](#)  
   test de chaîne vide [108](#)

fonction LN  
   description [109](#)

fonction LOG  
   description [109](#)

fonction LOWER  
   description [110](#)  
   internationalisation [13](#)

fonction LPAD  
   description [111](#)

fonction LTRIM  
   description [113](#)

fonction MAKE\_DATE\_TIME  
   description [114](#)

fonction MAX (chaîne)  
   description [118](#)

fonction MAX (dates)  
   description [115](#)  
   internationalisation [13](#)

fonction MAX (nombres)  
   description [116](#)  
   internationalisation [13](#)

fonction MD5  
   description [119](#)

fonction MEDIAN  
   description [120](#)

fonction MIN (dates)  
   description [125](#)  
   internationalisation [13](#)

fonction MIN (nombres)  
   description [126](#), [127](#)  
   internationalisation [13](#)

fonction MOD  
   description [129](#)

fonction MOVINGAVG  
   description [130](#)

fonction MOVINGSUM  
   description [132](#)

fonction NPER  
   description [133](#)

fonction PERCENTILE  
   description [134](#)

fonction PMT  
   description [136](#)

Fonction POWER  
   description [136](#)

fonction PV  
   description [137](#)

fonction RAND  
   description [138](#)

fonction RATE  
   description [139](#)

fonction REG\_EXTRACT  
   description [140](#)  
   utilisation de la syntaxe perl [140](#)

fonction REG\_MATCH  
   description [142](#)  
   utilisation de la syntaxe perl [140](#)

fonction REG\_REPLACE  
   description [144](#)

fonction REPLACECHR  
   description [145](#)

fonction REPLACESTR  
   description [148](#)  
 fonction REVERSE  
   description [151](#)  
 fonction ROUND (dates)  
   description [152](#)  
   traitement des sous-secondes [152](#)  
 fonction ROUND (nombres)  
   description [156](#)  
 fonction RPAD  
   description [159](#)  
 fonction RTRIM  
   description [160](#)  
 fonction SET\_DATE\_PART  
   description [161](#)  
 fonction SIGN  
   description [164](#)  
 fonction SIN  
   description [165](#)  
 fonction SINH  
   description [166](#)  
 fonction SOUNDEX  
   description [167](#)  
 fonction SQL IS\_CHAR  
   utilisation de REG\_MATCH [142](#)  
 fonction SQL LIKE  
   utilisation de REG\_MATCH [142](#)  
 fonction SQRT  
   description [169](#)  
 fonction STDDEV  
   description [170](#)  
 fonction SUBSTR  
   description [172](#)  
 fonction SUM  
   description [174](#)  
 fonction SYSTIMESTAMP  
   description [175](#)  
 fonction TAN  
   description [177](#)  
 fonction TANH  
   description [177](#)  
 fonction TO\_CHAR (dates)  
   chaînes de format [35](#)  
   description [180](#)  
   exemples [37](#)  
 fonction TO\_CHAR (nombres)  
   description [185](#)  
 fonction TO\_DATE  
   chaînes de format [38](#)  
   description [186](#)  
   exemples [41](#)  
 fonction TO\_DECIMAL  
   description [190](#)  
 fonction TO\_FLOAT  
   description [191](#)  
 fonction TO\_INTEGER  
   description [192](#)  
 fonction TRUNC (dates)  
   description [193](#)  
   traitement des sous-secondes [193](#)  
 fonction TRUNC (nombres)  
   description [196](#)  
 fonction UPPER  
   description [198](#)  
   internationalisation [13](#)  
 fonction UUID\_UNPARSE  
   description [199](#)

fonction UUID4  
   description [199](#)  
 fonction VARIANCE  
   description [200](#)  
 fonctions  
   agrèger [43](#)  
   caractère [45](#)  
   catégories [43](#)  
   chaîne [50](#)  
   codage [48](#)  
   conversion [46](#)  
   date [47](#)  
   description [12](#)  
   financières [48](#)  
   internationalisation [13](#)  
   nettoyage des données [46](#)  
   numérique [48](#)  
   scientifique [49](#)  
   spécial [49](#)  
   test [50](#)  
 fonctions agrégation  
   AVG [58](#)  
   COUNT [68](#)  
   description [43](#)  
   FIRST [83](#)  
   LAST [104](#)  
   MAX (chaîne) [118](#)  
   MAX (dates) [115](#)  
   MAX (nombres) [116](#)  
   MEDIAN [120](#)  
   MIN (dates) [125](#)  
   MIN (nombres) [126](#), [127](#)  
   PERCENTILE [134](#)  
   STDDEV [170](#)  
   SUM [174](#)  
   valeurs nulles [21](#), [45](#)  
 fonctions d'agrégation  
   ANY [56](#)  
   VARIANCE [200](#)  
 fonctions de caractères  
   ASCII [57](#)  
   CHR [62](#)  
   CHRCODE [63](#)  
   fonction CONCAT [64](#)  
   INITCAP [94](#)  
   INSTR [95](#)  
   liste de [45](#)  
   LONGUEUR [108](#)  
   LOWER [110](#)  
   LPAD [111](#)  
   LTRIM [113](#)  
   METAPHONE [121](#)  
   REG\_EXTRACT [140](#)  
   REG\_MATCH [142](#)  
   REG\_REPLACE [144](#)  
   REPLACECHR [145](#)  
   REPLACESTR [148](#)  
   RPAD [159](#)  
   RTRIM [160](#)  
   SOUNDEX [167](#)  
   SUBSTR [172](#)  
   UPPER [198](#)  
 fonctions de chaîne  
   CHOOSE [61](#)  
   description [50](#)  
   INDEXOF [93](#)  
   REVERSE [151](#)

#### fonctions de codage

AES\_DECRYPT [55](#)  
AES\_ENCRYPT [55](#)  
COMPRESS [64](#)  
CRC32 [71](#)  
DEC\_BASE64 [77](#)  
DECOMPRESS [80](#)  
description [48](#)  
ENC\_BASE64 [80](#)  
MD5 [119](#)

#### fonctions de conversion

description [46](#)  
TO\_CHAR (dates) [180](#)  
TO\_CHAR (nombres) [185](#)  
TO\_DATE [186](#)  
TO\_DECIMAL [190](#)  
TO\_FLOAT [191](#)  
TO\_INTEGER [192](#)

#### fonctions de date

ADD\_TO\_DATE [52](#)  
DATE\_COMPARE [73](#)  
DATE\_DIFF [74](#)  
GET\_DATE\_PART [86](#)  
LAST\_DAY [105](#)  
MAKE\_DATE\_TIME [114](#)  
MAX (dates) [115](#)  
MIN (dates) [125](#)  
ROUND [152](#)  
SET\_DATE\_PART [161](#)  
SYSTIMESTAMP [175](#)  
TRUNC (Dates) [193](#)

#### fonctions de nettoyage des données

description [46](#)  
GREATEST [88](#)  
IN [92](#)  
LEAST [107](#)

#### fonctions de test

description [50](#)  
IS\_DATE [99](#)  
IS\_NUMBER [101](#)  
IS\_SPACES [103](#)  
ISNULL [98](#)

#### fonctions financières

description [48](#)  
fonction FV [85](#)  
fonction NPER [133](#)  
fonction PMT [136](#)  
fonction PV [137](#)  
fonction RATE [139](#)

#### fonctions numériques

ABS [51](#)  
CEIL [60](#)  
CONVERT\_BASE [66](#)  
CUME [71](#)  
description [48](#)  
EXP [82](#)  
FLOOR [84](#)  
JOURNAL [109](#)  
LN [109](#)  
MOD [129](#)  
MOVINGAVG [130](#)  
MOVINGSUM [132](#)  
POWER [136](#)  
RAND [138](#)  
ROUND (nombres) [156](#)  
SIGN [164](#)  
SQRT [169](#)  
TRUNC (nombres) [196](#)

#### fonctions scientifiques

COS [66](#)  
COSH [67](#)  
description [49](#)  
SIN [165](#)  
SINH [166](#)  
TAN [177](#)  
TANH [177](#)

#### fonctions spéciales

ABANDONNER [50](#)  
DECODE [77](#)  
description [49](#)  
ERREUR [81](#)  
IIF [89](#)

#### format

à partir de la chaîne de caractères en date [186](#)  
à partir de la date en chaîne de caractères [180](#)  
format date/heure par défaut  
paramètre [33](#)

## G

#### guillemets

insertion unique avec la fonction CHR [14](#)  
guillemets simples dans des littéraux de chaîne  
fonction CHR [62](#)  
utilisation des fonctions CHR et CONCAT [64](#)

## H

#### hyperbolique

fonction cosinus [67](#)  
fonction sinus [166](#)  
fonction tangente [177](#)

## I

#### ignorer

lignes [81](#)

#### internationalisation

expression non valide [13](#)  
fonctions affectées [13](#)  
ordre de tri [13](#)

## L

#### qualificateur de référence :LKP

description [14](#)  
mot réservé [16](#)

#### langage de transformation

en comparaison à SQL [13](#)  
mots réservés [16](#)  
opérateurs [23](#)

#### lignes

compte [68](#)  
éviter les espaces [103](#)  
ignorer [81](#)  
renvoi de la dernière ligne [104](#)  
renvoi de la moyenne [130](#)  
renvoi de la première ligne [83](#)  
renvoi de la somme [132](#)  
renvoie n'importe quelle ligne [56](#)  
total cumulé [71](#)

littéraux  
  guillemets simples dans [62](#), [64](#)  
  spécification guillemets simples [14](#)  
littéraux de chaîne  
  guillemets simples dans [62](#), [64](#)  
  spécification guillemets simples [14](#)  
logarithme  
  retour [109](#)

## M

METAPHONE  
  description [121](#)  
minimum  
  valeur, retour [125](#)  
mises à jour du langage de transformation  
  expressions booléennes [21](#)  
  expressions de comparaison [21](#)  
mois  
  renvoi du dernier jour [105](#)  
mots réservés  
  list [16](#)  
moyennes  
  fonctions Agrégation pour déterminer [58](#)  
  retour [130](#)

## N

nombres entiers  
  conversion de valeurs en [192](#)  
NOT  
  mot réservé [16](#)  
numéros  
  arrondi [156](#)  
  troncation [196](#)

## O

opérateurs  
  arithmétique [24](#)  
  description [12](#)  
  opérateurs de chaîne [25](#)  
  opérateurs de comparaison [26](#)  
  opérateurs logiques [26](#)  
  utilisation de chaînes dans l'arithmétique [24](#)  
  utilisation de chaînes dans une comparaison [26](#)  
  valeurs nulles [22](#)  
opérateurs arithmétiques  
  description [24](#)  
  utilisation de chaînes dans les expressions [24](#)  
  utilisation pour la conversion de données [24](#)  
opérateurs de chaîne  
  description [25](#)  
opérateurs de comparaison  
  description [26](#)  
  utilisation de chaînes dans les expressions [26](#)  
opérateurs logiques  
  description [26](#)  
ordre de tri  
  internationalisation [13](#)  
OU  
  mot réservé [16](#)

## P

paramètres de mappage  
  définition [12](#)  
ports  
  syntaxe [14](#)  
précision élevée  
  ABS [51](#)  
  AVG [58](#)  
  CEIL [60](#)  
  CUME [71](#)  
  EXP [82](#)  
  fonction ABS [51](#)  
  fonction AVG [58](#)  
  fonction CUME [71](#)  
  fonction MAX [116](#)  
  fonction MEDIAN [120](#)  
  fonction MIN [126](#)  
  fonction MOVINGAVG [130](#)  
  fonction MOVINGSUM [132](#)  
  fonction PERCENTILE [134](#)  
  Fonction ROUND [156](#)  
  fonction STDDEV [170](#)  
  fonction SUM [174](#)  
  fonction TO\_DECIMAL [190](#)  
  fonction TRUNC [196](#)  
  JOURNAL [109](#)  
  MAX (nombres) [116](#)  
  MEDIAN [120](#)  
  MIN (nombres) [126](#)  
  MOD [129](#)  
  MOVINGAVG [130](#)  
  MOVINGSUM [132](#)  
  opérateurs arithmétiques [24](#)  
  PERCENTILE [134](#)  
  POWER [136](#)  
  ROUND (nombres) [156](#)  
  SIGN [164](#)  
  SIN [165](#)  
  SUM [174](#)  
priorité des opérateurs  
  expressions [23](#)

## Q

qualificateur de référence :INFA  
  mot réservé [16](#)  
qualificateur de référence :MCR  
  mot réservé [16](#)  
qualificateurs de référence  
  description [14](#)

## R

racine carrée  
  retour [169](#)  
recherches multiples  
  exemple de constante TRUE [22](#)

## S

Service d'intégration de données  
  gestion des valeurs nulles dans les expressions de comparaison [21](#)  
sinus  
  retour [165](#), [166](#)

- somme
  - retour [132](#), [174](#)
- sous-secondes
  - traitement dans la fonction ROUND (dates) [152](#)
  - traitement dans la fonction TRUNC (dates) [193](#)
- SPOUTPUT
  - mot réservé [16](#)
- stratégie de mise à jour
  - Exemple DD\_DELETE [18](#)
  - Exemple DD\_INSERT [19](#)
  - Exemple DD\_REJECT [19](#)
  - Exemple DD\_UPDATE [20](#)
- syntaxe
  - expression [13](#)
  - ports [14](#)
  - règles générales [15](#)
  - valeurs de retour [14](#)
- Syntaxe COBOL
  - conversion en syntaxe perl [140](#)
- Syntaxe d'expression régulière compatible perl
  - Utilisation dans une fonction REG\_MATCH [140](#)
  - utilisation de la fonction REG\_EXTRACT [140](#)
- Syntaxe SQL
  - conversion en syntaxe perl [140](#)

## T

- tangente
  - retour [177](#)
- total cumulé
  - retour [71](#)
- Transformation filtre
  - utilisation de la fonction ISNULL [98](#)
- troncation
  - dates [193](#)
  - numéros [196](#)
- types de données
  - Date/Heure [30](#)

## U

- Unicode
  - conversion de caractères en valeurs Unicode [57](#)
  - conversion de valeurs Unicode [62](#)
  - conversion en valeurs ASCII [63](#)

## V

- valeurs absolues
  - obtention [51](#)
- Valeurs d'exposant
  - calcul [82](#)
  - retour [136](#)
- valeurs date/heure
  - ajout [52](#)

- valeurs de précision double
  - nombres à virgule flottante [191](#)
- valeurs de retour
  - description [12](#)
  - syntaxe [14](#)
- valeurs de type chaîne
  - renvoi du maximum [118](#)
  - renvoi du minimum [127](#)
- valeurs décimales
  - conversion [190](#)
- valeurs négatives
  - SIGN [164](#)
- valeurs nulles
  - conditions de filtre [22](#)
  - dans les expressions de comparaison [21](#)
  - fonctions agrégation [21](#), [45](#)
  - ISNULL [98](#)
  - opérateur de chaîne [25](#)
  - opérateurs [22](#)
  - opérateurs logiques [27](#)
  - recherche [98](#)
- valeurs numériques
  - conversion en chaînes de texte [185](#)
  - renvoi d'une valeur absolue [51](#)
  - renvoi de l'écart-type [170](#)
  - renvoi de la tangente [177](#)
  - renvoi de la tangente hyperbolique [177](#)
  - renvoi de logarithmes [109](#)
  - renvoi du cosinus [66](#)
  - renvoi du cosinus hyperbolique de [67](#)
  - renvoi du minimum [126](#)
  - renvoi du sinus [165](#)
  - renvoi du sinus hyperbolique [166](#)
  - renvoi la racine carrée [169](#)
  - SIGN [164](#)
- Valeurs par défaut
  - fonction ERROR [81](#)
- valeurs positives
  - SIGN [164](#)
- variable PROC\_RESULT
  - mot réservé [16](#)
- variable SESSSTARTTIME
  - utilisation dans des fonctions de date [42](#)
- variable SYSDATE
  - description [28](#)
  - mot réservé [16](#)
  - utilisation dans les expressions [28](#)
- variables
  - SYSDATE [28](#)
  - variables intégrées [28](#)
- variables de mappage
  - variables intégrées [28](#)
- variables intégrées
  - description [28](#)
- variables locales
  - description [12](#)
- variables système [28](#)