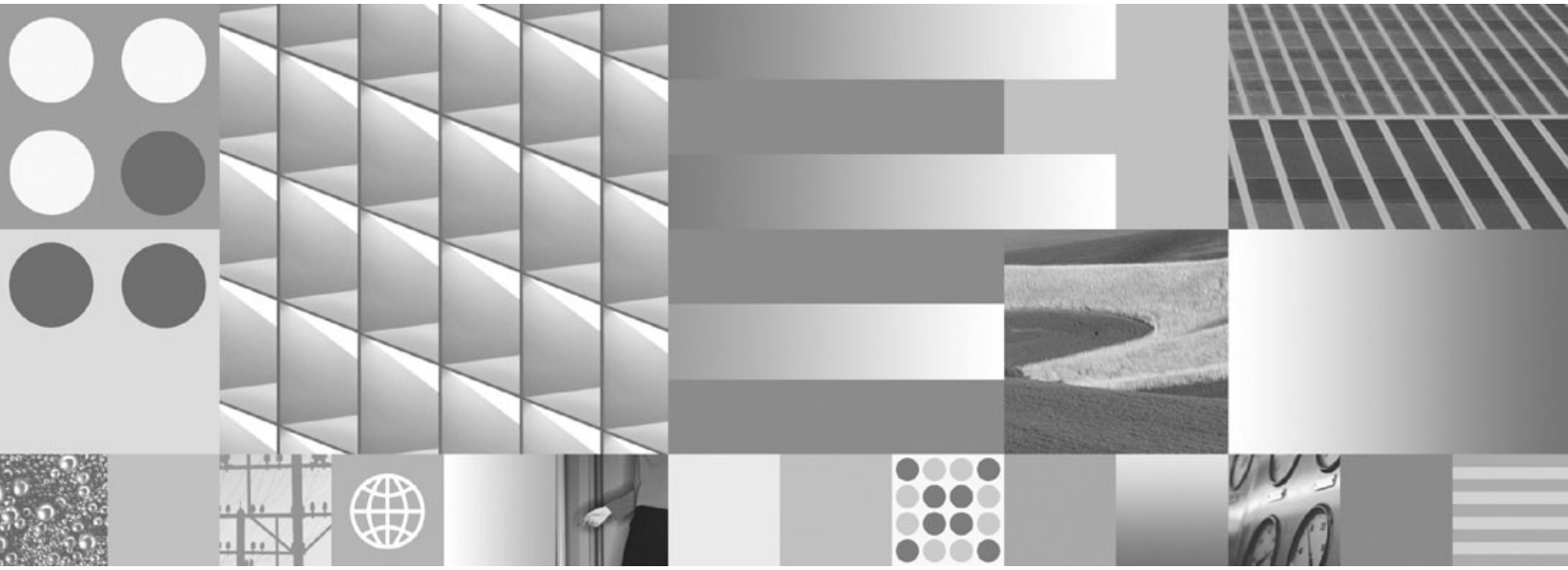
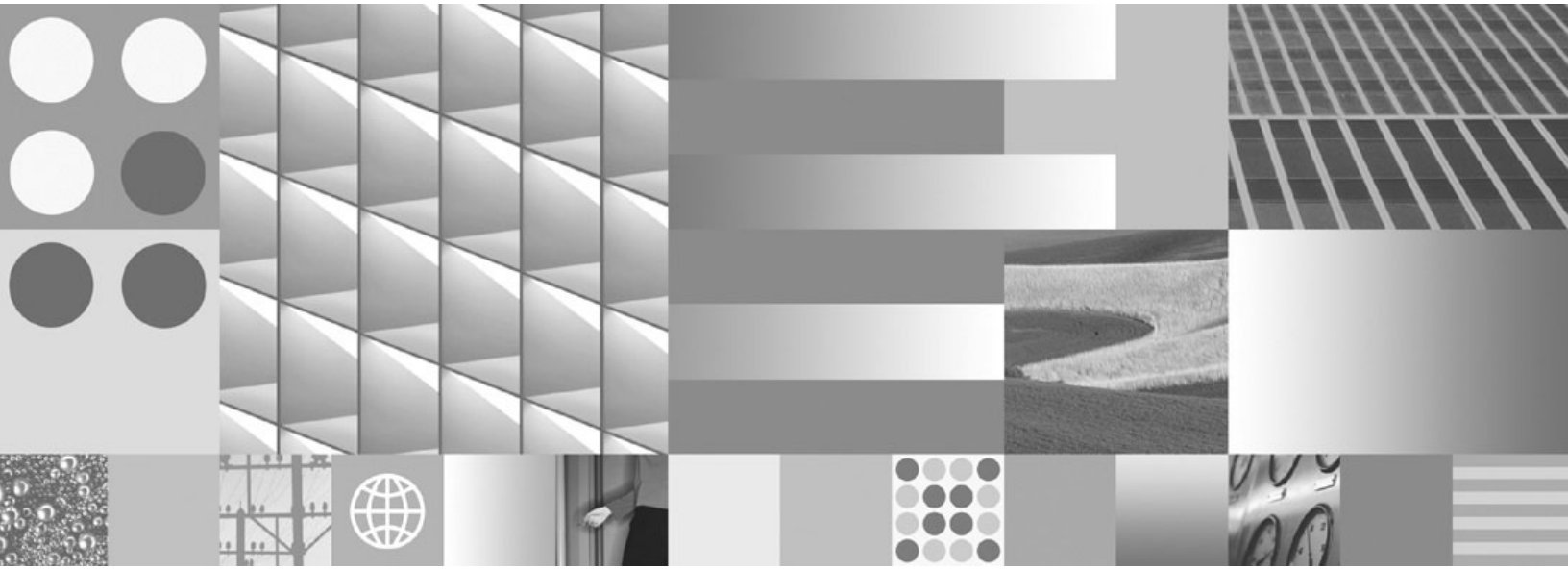
DB2 Version 9.5
for Linux, UNIX, and Windows



XQuery リファレンス

SC88-4448-01
(英文原典：SC23-5872-01)

DB2 Version 9.5
for Linux, UNIX, and Windows



XQuery リファレンス

SC88-4448-01
(英文原典：SC23-5872-01)

ご注意

本書および本書で紹介する製品をご使用になる前に、241 ページの『付録 B. 特記事項』に記載されている情報をお読みください。

当版に関する特記事項

本書には、IBM の専有情報が含まれています。その情報は、使用許諾条件に基づき提供され、著作権により保護されています。本書に記載される情報には、いかなる製品の保証も含まれていません。また、本書で提供されるいかなる記述も、製品保証として解釈すべきではありません。

IBM 資料は、オンラインでご注文いただくことも、ご自分の国または地域の IBM 担当員を通してお求めいただくこともできます。

- オンラインで資料を注文するには、www.ibm.com/shop/publications/order にある IBM Publications Center をご利用ください。
- ご自分の国または地域の IBM 担当員を見つけるには、www.ibm.com/planetwide にある IBM Directory of Worldwide Contacts をお調べください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

IBM 発行のマニュアルに関する情報のページ

`http://www.ibm.com/jp/manuals/`

こちらから、日本語版および英語版のオンライン・ライブラリーをご利用いただけます。また、マニュアルに関するご意見やご感想を、上記ページよりお送りください。今後の参考にさせていただきます。

(URL は、変更になる場合があります)

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原　典： SC23-5872-01
DB2 Version 9.5 for Linux, UNIX, and Windows
XQuery Reference

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2008.3

© Copyright International Business Machines Corporation 2006, 2008. All rights reserved.

目次

本書について vii

第 1 章 DB2 XQuery の概念 1

XQuery の概要. 1
XQuery と SQL の比較. 2
XQuery 関数を使用した DB2 データの検索 3
XQuery および XPath のデータ・モデル 5
シーケンスおよび項目 5
原子値 6
ノード階層 6
ノードのプロパティー 8
ノードの種類 8
ノードの文書順序 11
ノード ID 11
ノードの型付き値およびストリング値 11
XDM のシリアライゼーション 12
XML ネーム・スペースと QName 13
修飾名 (QName) 13
静的に既知のネーム・スペース 14
言語規則 15
大/小文字の区別 15
空白 15
コメント 16
XQuery に関する詳細情報の検索先 17

第 2 章 タイプ・システム 19

タイプ階層 19
カテゴリー別のタイプ 20
組み込みデータ・タイプのコンストラクター関数 . . 25
タイプ・キャスト 27
anyAtomicType タイプ 29
anySimpleType タイプ 29
anyType タイプ 30
anyURI タイプ 30
base64Binary タイプ 30
ブール・データ・タイプ 30
byte タイプ 30
date タイプ 31
dateTime タイプ 31
dayTimeDuration タイプ 33
decimal タイプ 34
double タイプ. 34
duration タイプ 35
ENTITY タイプ 36
float タイプ 36
gDay タイプ 37
gMonth タイプ 37
gMonthDay タイプ 38
gYear タイプ 38
gYearMonth タイプ 39
hexBinary タイプ 39
ID タイプ 39
IDREF タイプ 39
int タイプ 40
integer タイプ 40
language タイプ 40
long タイプ 40
Name タイプ 40
NCName タイプ 41
negativeInteger タイプ 41
NMTOKEN タイプ 41
nonNegativeInteger タイプ. 41
nonPositiveInteger タイプ 41
normalizedString タイプ 42
NOTATION タイプ. 42
positiveInteger タイプ 42
QName タイプ 42
short タイプ 43
string タイプ 43
time タイプ 43
token タイプ 44
unsignedByte タイプ 44
unsignedInt タイプ 44
unsignedLong タイプ 44
unsignedShort タイプ 44
untyped タイプ 44
untypedAtomic タイプ 45
yearMonthDuration タイプ. 45

第 3 章 プロローグ 47

バージョン宣言 47
境界スペース宣言 48
構成宣言 49
Copy-namespaces 宣言 49
デフォルトのエレメント/タイプのネーム・スペース宣言. 50
デフォルトの関数ネーム・スペース宣言. 51
空の順序宣言. 52
順序付けモード宣言 52
ネーム・スペース宣言. 53

第 4 章 式 55

式の評価および処理 55
動的コンテキストおよびフォーカス 55
優先順位 55
XQuery 式の結果の順序 56
原子化 58
サブタイプ置換 59
タイプのプロモーション 59
有効なブール値 60
基本式 61
リテラル 61

© Copyright IBM Corp. 2006, 2008

変数参照 63
括弧で囲んだ式 64
コンテキスト・アイテム式 64
関数呼び出し 64
パス式 65
パス式の構文 66
軸ステップ 67
パス式の省略構文 71
述部 73
シーケンス式 74
シーケンスを構成する式 74
フィルター式 75
ノードのシーケンスを結合するための式 76
算術式 77
比較式 80
値比較 80
一般比較 82
ノード比較 84
論理式 85
コンストラクター 86
コンストラクターの括弧で囲まれた式 87
直接エレメント・コンストラクター 88
計算エレメント・コンストラクター 96
計算属性コンストラクター 97
文書ノード・コンストラクター 98
テキスト・ノード・コンストラクター 99
処理命令コンストラクター 100
コメント・コンストラクター 101
FLWOR 式 102
FLWOR 式の構文 102
for 節および **let** 節 103
where 節 107
order by 節 108
return 節 110
FLWOR の例 111
条件式 114
量化式 115
キャスト式 117
キャスト可能な式 118
変換式と更新式 120
変換式での更新式の使用 120
変換式 123
基本更新式 126

第 5 章 組み込み関数 137
カテゴリー別の関数 137
adjust-date-to-timezone 関数 143
adjust-dateTime-to-timezone 関数 145
adjust-time-to-timezone 関数 147
abs 関数 149
avg 関数 150
boolean 関数 151
ceiling 関数 152
codepoints-to-string 関数 152
compare 関数 153
concat 関数 154
contains 関数 154
count 関数 155
current-date 関数 155
current-dateTime 関数 156
current-time 関数 156
data 関数 157
dateTime 関数 157
day-from-date 関数 158
day-from-dateTime 関数 159
days-from-duration 関数 159
deep-equal 関数 160
default-collation 関数 162
distinct-values 関数 162
empty 関数 163
ends-with 関数 163
exactly-one 関数 164
exists 関数 165
false 関数 165
floor 関数 165
hours-from-dateTime 関数 166
hours-from-duration 関数 167
hours-from-time 関数 168
implicit-timezone 関数 168
in-scope-prefixes 関数 169
index-of 関数 169
insert-before 関数 170
last 関数 171
local-name 関数 171
local-name-from-QName 関数 172
lower-case 関数 173
matches 関数 174
max 関数 175
min 関数 176
minutes-from-dateTime 関数 178
minutes-from-duration 関数 178
minutes-from-time 関数 179
month-from-date 関数 179
month-from-dateTime 関数 180
months-from-duration 関数 180
name 関数 181
namespace-uri 関数 182
namespace-uri-for-prefix 関数 183
namespace-uri-from-QName 関数 184
node-name 関数 184
normalize-space 関数 185
normalize-unicode 関数 185
not 関数 186
number 関数 187
one-or-more 関数 188
position 関数 188
QName 関数 189
remove 関数 189
replace 関数 190
resolve-QName 関数 192
reverse 関数 193
root 関数 193

round 関数 194
round-half-to-even 関数 195
seconds-from-dateTime 関数 196
seconds-from-duration 関数 197
seconds-from-time 関数 198
sqlquery 関数 198
starts-with 関数 201
string 関数 202
string-join 関数 203
string-length 関数 203
string-to-codepoints 関数 204
subsequence 関数 205
substring 関数 205
substring-after 関数 206
substring-before 関数 207
sum 関数 208
timezone-from-date 関数 209
timezone-from-dateTime 関数 210
timezone-from-time 関数 210
tokenize 関数 211
translate 関数 212
true 関数 214
unordered 関数 214
upper-case 関数 214
xmlcolumn 関数 216
year-from-date 関数 217
year-from-dateTime 関数 217
years-from-duration 関数 218
zero-or-one 関数 219

第 6 章 正規表現 221

第 7 章 制限 229
XQuery タイプの制限 229
サイズ制限 230

付録 A. DB2 技術情報の概説 231
DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式) 232
DB2 の印刷資料の注文方法 234
コマンド行プロセッサーから SQL 状態ヘルプを表示する 235
異なるバージョンの DB2 インフォメーション・センターへのアクセス 235
DB2 インフォメーション・センターでの希望する言語でのトピックの表示 236
コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの更新 236
DB2 チュートリアル 239
DB2 トラブルシューティング情報 239
ご利用条件 240

付録 B. 特記事項 241

索引 245

本書について

「XQuery リファレンス」では、DB2® データベースが XML データを処理するために使用する XQuery 言語について説明します。本書には、XQuery の概念、データ・タイプ、言語エレメント、XQuery 定義の関数、および DB2 定義の関数についての情報が記載されています。さらに、このリファレンスには DB2 XQuery のサイズ制限および XQuery データ・タイプの制限に関する情報も含まれています。

© Copyright IBM Corp. 2006, 2008

第 1 章 DB2 XQuery の概念

以下のトピックでは、XQuery についての基本的な概念を紹介し、XQuery が DB2 データベースでどのように機能するかを説明します。

XQuery の概要

XQuery は、XML データを照会および変更するための特定の要件を満たすように、World Wide Web Consortium (W3C) によって設計された機能プログラミング言語です。

予測可能で、規則的な構造を持つリレーショナル・データとは異なり、XML データは非常に柔軟性があります。XML データは多くの場合、予測不能であり、散在しており、自己記述タイプです。

XML データの構造は予測不能であるため、 XML データに対して実行する必要のある照会は、多くの場合、典型的なリレーショナル照会とは異なります。XQuery 言語は、このような操作を実行するために必要な柔軟性を提供します。例えば、以下のような操作を実行するには、XQuery 言語を使用しなければならない場合があります。

- 階層のどのレベルにあるかが不明なオブジェクトの XML データを検索する。
- データに対して構造変換を実行する (例えば、階層を逆転させたい場合など)。
- タイプが混合した結果を戻す。
- 既存の XML データを更新する。

XQuery 照会のコンポーネント

XQuery では、式が照会の主要ビルディング・ブロックとなります。式はネスト可能で、照会の本体を形成します。照会には、その本体の前にプロローグも含めることができます。*プロローグ* には、照会の処理環境を定義する一連の宣言が含まれます。*照会本体* は、照会の結果を定義する式で構成されます。この式は、演算子またはキーワードを使用して結合される複数の XQuery 式で構成できます。

2 ページの図 1は、典型的な照会の構造を示しています。この例では、プロローグに以下の 2 つの宣言が含まれます。照会を処理するために使用する XQuery 構文のバージョンを指定するバージョン宣言、および接頭部のないエレメント名およびタイプ名に使用するネーム・スペース URI を指定するデフォルト・ネーム・スペース宣言。照会本体には、 `price_list` エレメントを構成する式が含まれます。`price_list` エレメントの内容は、価格によって降順にソートされる `product` エレメントのリストです。

© IBM Corporation 2006, 2008

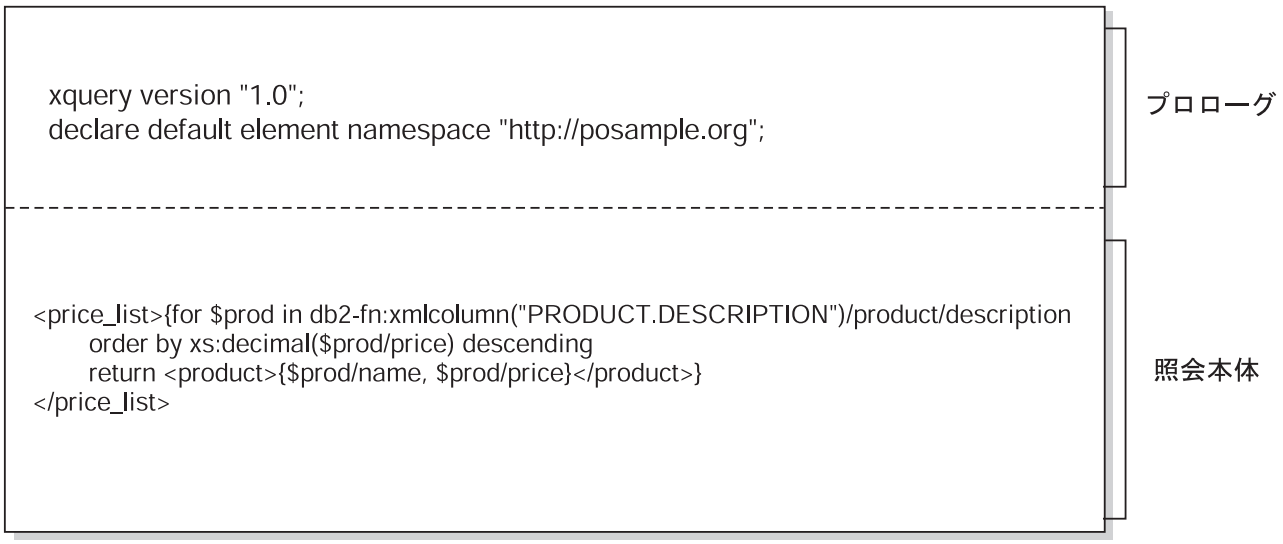


図 1. XQuery における典型的な照会の構造

XQuery と SQL の比較

DB2 データベースでは、SQL、XQuery、または SQL と XQuery の組み合わせを使用して、表の列への整形式 XMLデータの保管、およびデータベースからの XML データの検索をサポートします。両方の言語が基本照会言語としてサポートされています。また、どちらの言語も他の言語を呼び出すための関数を提供します。

XQuery

XQuery を直接呼び出す照会は、キーワード XQUERY で始まります。このキーワードは XQuery が使用されていることを示し、 XQuery 言語に適用される、大/小文字を区別する規則を、 DB2 サーバーが使用する必要があることを示します。エラー処理は、XQuery 式を処理するために使用されるインターフェースに基づきます。XQuery エラーは、SQL エラーが報告されるのと同じ方法で、SQLCODE および SQLSTATE を使用して報告されます。XQuery 式の処理で警告は戻されません。XQuery では、 DB2 表とビューから XML データを抽出する関数を呼び出すことでデータを取得します。XQuery は、SQL 照会から呼び出すこともできます。この場合、SQL 照会では、バインド変数の形式で XML データを XQuery に渡すことができます。XQuery は、XML データの処理や、エレメントおよび属性などの新規 XML オブジェクトの構成のために、さまざまな式をサポートします。XQuery のプログラミング・インターフェースでは、SQL に類似した機能を提供して照会を準備し、照会結果を取得します。

SQL SQL により、XML データ・タイプの値を定義し、インスタンス化することができます。整形式 XML 文書を含むストリングは、XML 値に解析し、状況に応じて XML スキーマに対して妥当性検査し、表に挿入または表で更新することができます。または、他のリレーショナル・データを XML 値に変換する SQL コンストラクター関数を使用することで、XML 値を構成することもできます。XQuery を使用することで、XML データを照会することも、XML データをリレーショナル表に変換して SQL 照会で使用する

こともできます。データは、XML 値をストリング・データにシリアライズ化するだけでなく、SQL タイプと XML タイプの間でキャストすることもできます。

SQL/XML では、以下の関数と述部を指定して、SQL から XQuery を呼び出します。

XMLQUERY
XMLQUERY は、引数として XQuery 式を使用し、XML シーケンスを戻すスカラー関数です。この関数には、 SQL 値を XQuery 変数として XQuery 式に渡すために使用できるオプション・パラメーターが含まれています。XMLQUERY によって戻される XML 値は、SQL 照会のコンテキスト内でさらに処理することができます。

XMLTABLE
XMLTABLE は、XQuery 式を使用して XML データから SQL の表を生成する表関数で、この表は SQL でさらに処理することができます。

XMLEXISTS
XMLEXISTS は、XQuery 式が 1 つ以上の項目のシーケンスを戻すか (空のシーケンスではないか) どうかを判別する SQL 述部です。

XQuery 関数を使用した DB2 データの検索

XQuery では、照会において以下の関数のいずれかを呼び出して、DB2 データベースから入力 XML データを取得できます。db2-fn:sqlquery および db2-fn:xmlcolumn。

関数 db2-fn:xmlcolumn は、XML 列全体を検索します。一方 db2-fn:sqlquery は、SQL 全選択に基づく XML 値を検索します。

db2-fn:xmlcolumn
db2-fn:xmlcolumn 関数は、表またはビュー内の XML 列を識別するストリング・リテラル引数を使用し、その列にある XML 値のシーケンスを戻します。この関数の引数は、大/小文字を区別します。ストリング・リテラル引数は、修飾された XML タイプの列名にする必要があります。この関数により、検索条件を適用しなくても、XML データの列全体を抽出することができます。

以下の例において、照会は db2-fn:xmlcolumn 関数を使用して、BUSINESS.ORDERS 表の PURCHASE_ORDER 列内のすべての購入注文を取得します。続いてこの照会は、この入力データを操作して、これらの購入注文の配送先住所から市区町村を抽出します。照会の結果は、注文商品が配送されるすべての市区町村のリストです。

```
db2-fn:xmlcolumn('BUSINESS.ORDERS.PURCHASE_ORDER')/shipping_address/city
```

db2-fn:sqlquery
db2-fn:sqlquery 関数は、fullselect を表すストリング引数を使用し、fullselect によって戻される XML 値の連結である XML シーケンスを戻します。fullselect では、単一列の結果セットを指定する必要があり、列はデータ・タイプが XML である必要があります。fullselect を指定することにより、

SQL の機能を使用して XML データを XQuery に提供できます。関数は、パラメーターを使用した SQL ステートメントへの値の受け渡しをサポートします。

以下の例では、BUSINESS.ORDERS という表に PURCHASE_ORDER という XML 列が含まれています。この例の照会は、db2-fn:sqlquery 関数を使用して SQL を呼び出して、配送日付が 2005 年 6 月 15 日であるすべての購入注文を取得します。続いて、この照会は、この入力データを操作して、これらの購入注文の配送先住所から市区町村を抽出します。照会の結果は、6 月 15 日に注文商品が配送されるすべての市区町村のリストです。

```
db2-fn:sqlquery("
SELECT purchase_order FROM business.orders
WHERE ship_date = '2005-06-15' ")/shipping_address/city
```

重要: db2-fn:sqlquery 関数または db2-fn:xmlcolumn 関数によって戻される XML シーケンスには、原子値およびノードを含む任意の XML 値を含めることができます。これらの関数は、必ずしも整形式の文書のシーケンスを戻すとは限りません。例えば、関数が、XML タイプのインスタンスとして、単一の原子値 (36 など) を戻す場合があります。

SQL および XQuery には、名前の大/小文字の区別に関して異なる規則があります。db2-fn:sqlquery 関数および db2-fn:xmlcolumn 関数の使用時には、これらの差異に注意する必要があります。

SQL は大/小文字を区別する言語ではない

デフォルトでは、SQL ステートメントで使用されるすべての通常 ID は、自動的に大文字に変換されます。このため、SQL の表および列の名前は、上記の例における BUSINESS.ORDERS および PURCHASE_ORDER のように、通例は大文字の名前です。SQL ステートメントでは、 business.orders および purchase_order のように小文字の名前を使用してこれらの列を参照できますが、これらは SQL ステートメントの処理中に自動的に大文字に変換されます。(名前を二重引用符で囲むことにより、SQL で*区切り ID* と呼ばれる大/小文字を区別する名前を作成することもできます。)

XQuery は大/小文字を区別する言語です

XQuery は、小文字の名前を大文字に変換しません。この違いにより、XQuery および SQL の同時使用時に混乱が生じることがあります。db2-fn:sqlquery に渡されるストリングは、SQL 照会として解釈され、SQL パーサーによって構文解析されて、これによりすべての名前が大文字に変換されます。このため、 db2-fn:sqlquery の例では、表名 business.orders、および列名 purchase_order および ship_date を、大文字または小文字のいずれでも表示できます。ただし、db2-fn:xmlcolumn のオペランドは、SQL 照会ではありません。オペランドは、列名を表す、大/小文字を区別する XQuery ストリング・リテラルです。列の実際の名前は BUSINESS.ORDERS.PURCHASE_ORDER であるため、db2-fn:xmlcolumn のオペランドに、この名前を大文字で指定する必要があります。

XQuery および XPath のデータ・モデル

XQuery 式は、XQuery および XPath のデータ・モデル (XDM) のインスタンスに対して作動し、データ・モデルのインスタンスを戻します。XDM では、1 つ以上の XML 文書またはフラグメントの要約表記を使用します。データ・モデルは、中間計算時に使用される値を含め、XQuery における式のすべての暗黙的値を定義します。

XML データの XDM への構文解析、およびスキーマに対するデータの妥当性検査は、データが XQuery で処理される前に行われます。データ・モデルの生成時に、入力 XML 文書は解析され、XDM のインスタンスに変換されます。文書は、妥当性検査の有無にかかわらず構文解析できます。

XDM は、原子値およびノードのシーケンスで説明されます。

シーケンスおよび項目

XQuery および XPath データ・モデル (XDM) のインスタンスはシーケンスです。シーケンス は、 0 個以上の項目の順序付けられたコレクションです。*項目* は、原子値またはノードです。

シーケンスには、ノード、原子値、またはノードと原子値を混合させたものを含めることができます。例えば、以下のリストの各項目はシーケンスです。

- `36`
- `<dog/>`
- `(2, 3, 4)`
- `(36, <dog/>, "cat")`
- `()`

リスト内の項目に加えて、DB2 データベース内の XML 列に保管されている XML 文書もシーケンスです。

この例では、XQuery でシーケンスを構成するために使用する構文と整合した、シーケンスを表すための記法を使用しています。

- シーケンス内の各項目は、コンマで区切ります。
- シーケンス全体は括弧で囲みます。
- 一対の空の括弧は、空のシーケンスを表します。
- 単独で表示される単一の項目は、1 つの項目を含むシーケンスと等価です。

  例えば、シーケンス (36) と原子値 36 との間に区別はありません。

シーケンスはネストできません。2 つのシーケンスが結合されると、結果は常にノードと原子値のフラット化されたシーケンスです。例えば、シーケンス (2, 3) をシーケンス (3, 5, 6) に付加した結果は、単一のシーケンス (3, 5, 6, 2, 3) です。これらのシーケンスを結合しても、ネストされたシーケンスは存在し得ないため、シーケンス (3, 5, 6, (2, 3)) は作成されません。

0 個の項目を含むシーケンスを空のシーケンス と呼びます。空のシーケンスは、欠落した、または不明の情報を表すために使用できます。

原子値

原子値 とは、XML スキーマで定義されている組み込み原子タイプの 1 つのインスタンスです。これらのデータ・タイプには、ストリング、整数、10 進数、日付、その他の原子タイプが含まれています。これらのタイプは、それ以上分割できないため、原子として記述されます。

ノードとは異なり、原子値には ID がありません。原子値の各インスタンス (整数 7 など) は、その値の他のすべてのインスタンスと同一です。

以下は、原子値の作成方法の例です。

- 原子化と呼ばれるプロセスを使用して、ノードから抽出する。原子化は、原子値のシーケンスが必要な式で使用されます。
- 数値リテラルまたはストリング・リテラルとして指定する。リテラルは、XQuery によって原子値として解釈されます。例えば、以下のリテラルは、原子値として解釈されます。
  - ″this is a string″ (xs:string タイプです)
  - 45 (xs:integer タイプです)
  - 1.44 (xs:decimal タイプです)
- コンストラクター関数で計算する。例えば、以下のコンストラクター関数は、ストリング ″2005-01-01″ から xs:date タイプの値を作成します。

  ```
  xs:date("2005-01-01")
  ```
- 組み込み関数 fn:true() および fn:false() によって戻す。これらの関数は、ブール値 (`true` および `false`) を戻します。これらの値は、リテラルで表すことはできません。
- 算術式や論理式など、さまざまな種類の式を使用して戻す。

ノード階層

シーケンスのノードは、1 つのルート・ノードおよびルート・ノードから直接的または間接的に到達可能なすべてのノードで構成される、 1 つ以上の*階層* (またはツリー) を形成します。すべてのノードは必ず 1 つの階層に属し、すべての階層には必ず 1 つのルート・ノードがあります。DB2 は、文書、エレメント、属性、テキスト、処理命令、およびコメントの 6 種類のノードをサポートします。

以下の XML 文書 products.xml には、 `products` というルート・エレメントが含まれ、このエレメントに `product` エレメントが含まれます。各 `product` エレメントは、`pid` (製品 ID) という属性と、 `description` という子エレメントを持ちます。`description` エレメントには、 `name` および `price` という子エレメントが含まれます。

```
<products>
  <product xmlns="http://posample.org" pid="10">
    <description>
      <name>Fleece jacket</name>
      <price>19.99</price>
    </description>
  </product>
  <product xmlns="http://posample.org" pid="11">
    <description>
      <name>Nylon pants</name>
```

```
      <price>9.99</price>
    </description>
  </product>
</products>
```

図 2 は、 products.xml の単純化されたデータ・モデルを示しています。この図には、文書ノード (D)、エレメント・ノード (E)、属性ノード (A)、およびテキスト・ノード (T) が含まれています。

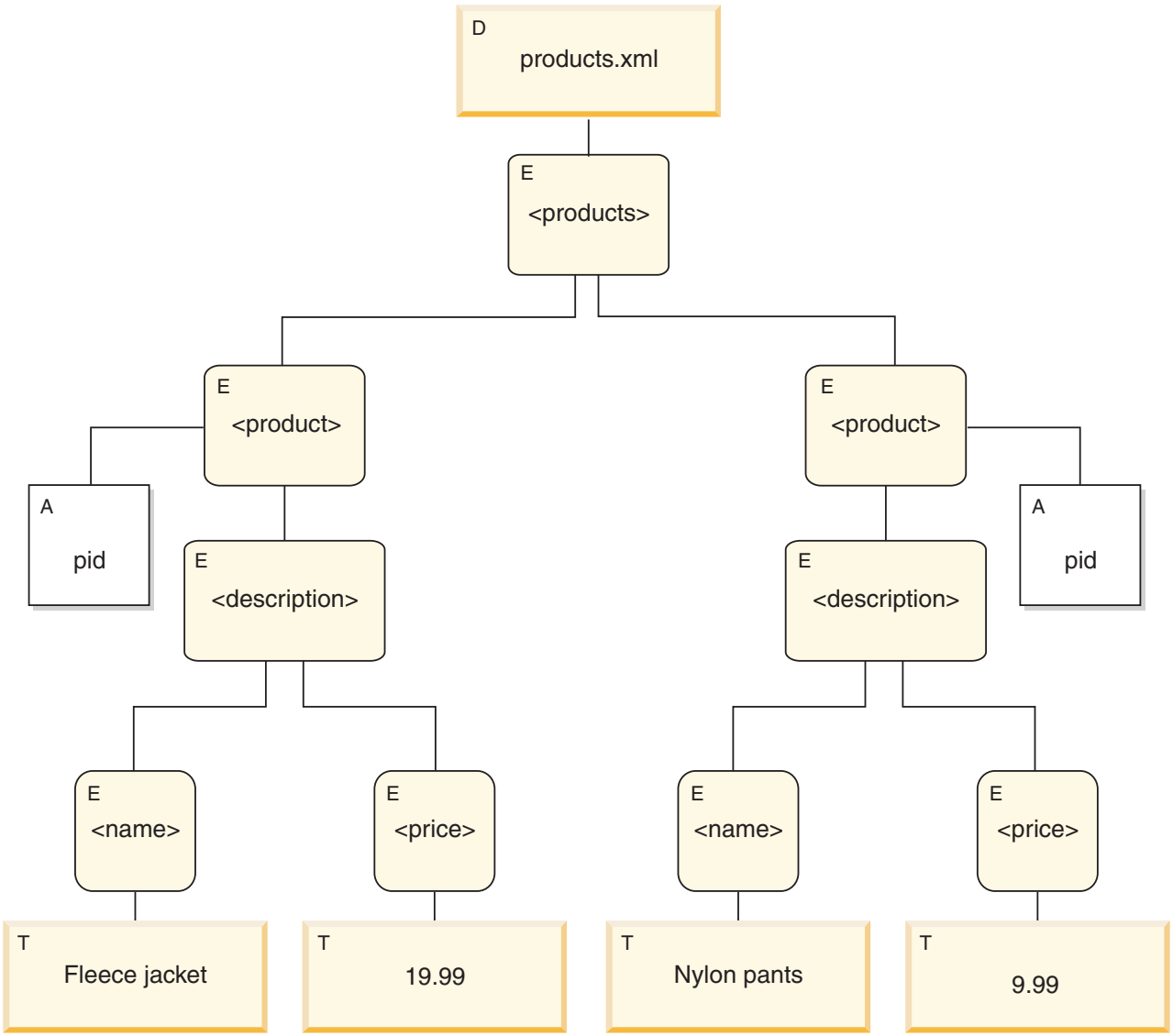


図 2. 文書 products.xml のデータ・モデル図

この例が示しているように、ノードは他のノードを子として持つことができ、このようにして 1 つ以上の*ノード階層* が形成されます。この例では、エレメント `product` は `products` の子です。エレメント `description` は `product` の子です。エレメント `name` および `price` は、エレメント `description` の子です。値 `Fleece Jacket` を持つテキスト・ノードは、エレメント `name` の子であり、テキスト・ノード `19.99` は、エレメント `price` の子です。

ノードのプロパティー

各ノードは、そのノードの特性を説明するプロパティー を持ちます。例えば、ノードのプロパティーには、ノードの名前、子、親、属性、およびそのノードを説明するその他の情報が含まれます。ノードの種類により、特定のノードがどのプロパティーを持つかが決まります。

ノードは、以下の 1 つ以上のプロパティーを持つことができます。

node-name
QName として表現されるノードの名前。

parent 現行ノードの親であるノード。

type-name
ノードの動的 (ランタイム) タイプ (タイプ・アノテーション とも言う)。

children
現行ノードの子であるノードのシーケンス。

attributes
現行ノードに属する属性ノードのセット。

string-value
ノードから抽出可能なストリング値。

typed-value
ノードから抽出可能な 0 個以上の原子値のシーケンス。

in-scope namespaces
ノードに関連付けられた範囲内のネーム・スペース。

content
ノードの内容。

ノードの種類

DB2 は、文書、エレメント、属性、テキスト、処理命令、およびコメントの 6 種類のノードをサポートします。

文書ノード

文書ノードは、XML 文書をカプセル化します。

文書ノードは、0 個以上の子を持つことができます。子は、エレメント・ノード、処理命令ノード、コメント・ノード、およびテキスト・ノードを含むことができます。

文書ノードのストリング値は、その派生したすべてのテキスト・ノードを文書順序で連結したコンテンツと同じです。ストリング値のタイプは xs:string です。文書ノードの型付き値は、型付き値が xdt:untypedAtomic タイプであることを除き、ストリング値と同じです。

文書ノードには、以下のノード・プロパティーがあります。

- children (空の場合もある)
- string-value
- typed-value

文書ノードは、計算コンストラクターを使用することで、XQuery 式で構成できます。文書ノードのシーケンスも、db2-fn:xmlcolumn 関数により戻すことができます。

エレメント・ノード

エレメント・ノードは、XML エレメントをカプセル化します。

エレメントは、 0 または 1 個の親、および 0 個以上の子を持つことができます。子は、エレメント・ノード、処理命令ノード、コメント・ノード、およびテキスト・ノードを含むことができます。文書ノードおよび属性ノードは、エレメント・ノードの子になることはありません。ただし、エレメント・ノードは、その属性の親であると認識されます。エレメント・ノードの属性は、固有の QName を持つ必要があります。

エレメント・ノードには、以下のノード・プロパティーがあります。

- node-name
- parent (空の場合もある)
- type-name
- children (空の場合もある)
- attributes (空の場合もある)
- string-value
- typed-value
- in-scope-namespaces

エレメント・ノードは、直接コンストラクターまたは計算コンストラクターを使用することで、XQuery 式で構成できます。

エレメント・ノードの type-name プロパティーは、エレメント・ノードの型付き値とストリング値との関係を示します。例えば、エレメント・ノードが type-name プロパティー xs:decimal とストリング値「47.5」を持つ場合、型付き値は 10 進数値 47.5 になります。エレメント・ノードの type-name プロパティーが xdt:untyped の場合、エレメントの型付き値は、そのストリング値と等しく、xdt:untypedAtomic タイプになります。

属性ノード

属性ノードは、XML 属性を意味します。

属性ノードは、0 または 1 個の親を持つことができます。属性を所有するエレメント・ノードは、属性ノードが親エレメントの子でない場合でも、その親であると考えられます。

属性ノードには、以下のノード・プロパティーがあります。

- node-name
- parent (空の場合もある)
- type-name
- string-value
- typed-value

属性ノードは、直接コンストラクターまたは計算コンストラクターを使用することで XQuery 式で構成できます。

属性ノードの type-name プロパティーは、属性ノードの型付き値とストリング値との関係を示します。例えば、属性ノードが type-name プロパティー xs:decimal とストリング値「47.5」を持つ場合、その型付き値は 10 進数値 47.5 になります。

テキスト・ノード

テキスト・ノードは、XML の文字内容をカプセル化します。

テキスト・ノードは、0 または 1 個の親を持つことができます。1 つの文書の子であるテキスト・ノード、またはエレメント・ノードが、隣接する兄弟として出現することはありません。文書ノードまたはエレメント・ノードが構成されると、隣接するテキスト・ノードの兄弟が結合されて単一のテキスト・ノードになります。結果のテキスト・ノードが空の場合は廃棄されます。

テキスト・ノードには、以下のノード・プロパティーがあります。

- content (空の場合もある)
- parent (空の場合もある)

テキスト・ノードは、XQuery 式において、計算コンストラクターによって、または直接エレメント・コンストラクターのアクションによって構成することができます。

処理命令ノード

処理命令ノードは、XML 処理命令をカプセル化します。

処理命令ノードは、0 または 1 個の親を持つことができます。処理命令の内容に、ストリング ?> を含めることはできません。処理命令のターゲットは、NCName にする必要があります。ターゲットは、命令の送信先であるアプリケーションを識別するために使用されます。

処理命令ノードには、以下のノード・プロパティーがあります。

- target
- content
- parent (空の場合もある)

処理命令ノードは、XQuery 式で、直接コンストラクターまたは計算コンストラクターを使用して構成することができます。

コメント・ノード

コメント・ノードは、XML コメントをカプセル化します。

コメント・ノードは、0 または 1 個の親を持つことができます。コメント・ノードのコンテンツには、ストリング「--」(2 つのハイフン) を含めることも、最後の文字としてハイフン文字 (-) を含めることもできません。

コメント・ノードは、以下のプロパティーを持ちます。

- content

- parent (空の場合もある)

コメント・ノードは、直接コンストラクターまたは計算コンストラクターを使用することで、XQuery 式内に構成できます。

ノードの文書順序

階層のすべてのノードは、*文書順序* と呼ばれる順序に従います。この順序では、各ノードはその子の前に表示されます。文書順序は、ノードの階層がシリアライズされた XML で示される場合にノードが表示される順序に対応します。

階層内のノードは、以下の順序で表示されます。

- ルート・ノードが 1 番目のノードです。
- エレメント・ノードは、その子の前に表示されます。
- 属性ノードは関連付けられているエレメント・ノードの直後に表示されます。属性ノードの相対順序は任意ですが、この順序は照会処理時も変わりません。
- 兄弟の相対順序は、ノード階層内の順序で決まります。
- ノードの子と子孫は、ノードの後にある兄弟の前に表示されます。

ノード ID

各ノードにはユニークな ID があります。2 つのノードは、その名前や値が同じでも区別できます。一方、原子値には ID はありません。

ノード ID は、ID-type 属性とは異なります。XML 文書内のエレメントには、文書の作成者が ID-type 属性を指定することができます。一方ノード ID は、システムによってすべてのノードに自動的に割り当てられますが、ユーザーにとって直接的には不可視です。

ノード ID は、以下の種類の式を処理するために使用されます。

- ノード比較。**is** 演算子は、2 つのノードが同一 ID を持っているかどうか判別するためにノード ID を使用します。
- パス式。パス式は、重複ノードを除去するためにノード ID を使用します。
- シーケンス式。**union** 演算子、**intersect** 演算子、または **except** 演算子は、重複ノードを除去するためにノード ID を使用します。

ノードの型付き値およびストリング値

各ノードは、*型付き値* および*ストリング値* の両方を持っています。これらの 2 つのノード・プロパティーは、特定の XQuery 操作 (原子化など) および関数 (fn:data、fn:string、および fn:deep-equal など) の定義で使用されます。

表 1. ノードのストリング値および型付き値

ノードの種類	ストリング値	型付き値
文書	文書の順序におけるそのすべての子孫テキスト・ノードを連結した内容である xs:string タイプのインスタンス。	文書の順序におけるそのすべての子孫テキスト・ノードを連結した内容である、xdt:untypedAtomic タイプのインスタンス。

表 1. ノードのストリング値および型付き値 (続き)

ノードの種類	ストリング値	型付き値
妥当性検査済み文書内のエレメント	• 妥当性検査により、単純タイプ (xs:decimal など) または単純な内容を持つタイプ (内容が xs:decimal である「temperature」タイプなど) がエレメント・ノードに割り当てられる場合、ストリング値は、オリジナルの XML 文書内のエレメントの値を表すストリングです。 • 妥当性検査により、混合した内容 (テキスト・エレメントと子エレメントの両方) を持つことが可能なタイプがエレメント・ノードに割り当てられる場合、ストリング値は、文書の順序におけるそのすべての子孫テキスト・ノードを連結した内容である、 xs:string タイプのインスタンスです。 • 妥当性検査により、内容を持つことができない (テキスト・エレメントも子エレメントも不可) タイプがエレメント・ノードに割り当てられる場合、エレメントのストリング値は空のストリングです。 • 妥当性検査により、子エレメントのみを含むことができる (テキストは不可) タイプがエレメント・ノードに割り当てられる場合、エレメントのストリング値は、文書の順序におけるそのすべての子孫テキスト・ノードを連結したストリング値で構成されます。	• 妥当性検査により、単純タイプ (xs:decimal など) または単純な内容を持つタイプ (内容が xs:decimal である「temperature」タイプなど) がエレメント・ノードに割り当てられる場合、型付き値は、ストリング値を、妥当性検査処理により割り当てられた単純タイプにキャストした結果です (xs:decimal など)。 • 妥当性検査により、混合した内容 (テキスト・エレメントと子エレメントの両方) を持つことが可能なタイプがエレメント・ノードに割り当てられる場合、型付き値は、文書の順序におけるそのすべての子孫テキスト・ノードを連結した内容である、 xdt:untypedAtomic タイプのインスタンスです。 • 妥当性検査により、内容を持つことができない (テキスト・エレメントも子エレメントも不可) タイプがエレメント・ノードに割り当てられる場合、型付き値は空のシーケンスです。 • 妥当性検査により、子エレメントのみを含むことができる (テキストは不可) タイプがエレメント・ノードに割り当てられる場合、エレメントは型付き値を持ちません。また、その型付き値を (例えば fn:data 関数によって) 抽出しようとすると、結果はエラーになります。
妥当性検査されていない文書内のエレメント	文書の順序におけるそのすべての子孫テキスト・ノードを連結した内容である、xs:string タイプのインスタンス。	文書の順序におけるそのすべての子孫テキスト・ノードを連結した内容である、xdt:untypedAtomic タイプのインスタンス。
妥当性検査済み文書内の属性	元の XML 文書内の属性値を表す xs:string タイプのインスタンス。	ストリング値を、妥当性検査時に属性に割り当てられたタイプにキャストした結果。例えば、属性が xs:decimal タイプを持つと検証された場合、そのストリング値は例えばストリング「74.8」などで、その型付き値は 10 進数としての 74.8 になります。
妥当性検査されていない文書内の属性	元の XML 文書内の属性値を表す xs:string タイプのインスタンス。	元の XML 文書内の属性値を表す xdt:untypedAtomic タイプのインスタンス。
テキスト	xs:string タイプのインスタンスとしての内容。	xdt:untypedAtomic タイプのインスタンスとしての内容。
コメント	xs:string タイプのインスタンスとしての内容。	xs:string タイプのインスタンスとしての内容。
処理命令	xs:string タイプのインスタンスとしての内容。	xs:string タイプのインスタンスとしての内容。

XDM のシリアライゼーション

XQuery 式の結果は、XDM のインスタンスですが、シリアライゼーション と呼ばれるプロセスを使用して、XML 表記に変換できます。

シリアライゼーション中に、ノードおよび原子値 (XDM のインスタンス) のシーケンスは、XML 表記に変換されます。シリアライゼーションの結果が、必ず完全な文書表記になるとは限りません。実際、シリアライゼーションの結果が単一の原子値 (17 など) または共通の親を持たないエレメントのシーケンスとなることがあります。

XQuery では、XDM をシリアライズする関数を提供していません。XDM がどのように XML データにシリアライズされるかは、照会が実行されている環境によって異なります。例えば、CLP (コマンド行プロセッサー) は、シリアライズされた項目のシーケンスを戻します。結果内では、シリアライズされた各項目が 1 つの行として戻されます。例えば、照会 `XQUERY (1, 2, 3)` を CLP から入力すると、以下の結果が戻されます。

```
12
3
```

シリアライゼーションは、SQL/XML 関数 XMLSERIALIZE を使用しても実行できます。

XML ネーム・スペースと QName

XML ネーム・スペースは、命名の衝突を回避します。*XML* ネーム・スペース は、ネーム・スペース URI によって識別される名前のコレクションです。ネーム・スペースにより、 XQuery のエレメント、属性、データ・タイプ、および関数に使用される名前の修飾方法が提供されます。ネーム・スペース接頭部で修飾された名前が、*修飾名* (*QName*) です。

修飾名 (QName)

QName は、オプションのネーム・スペース接頭部およびローカル名で構成されます。ネーム・スペース接頭部とローカル名は、コロンで区切ります。ネーム・スペース接頭部を指定する場合、これは URI (Universal Resource Identifier) にバインドされ、URI の短縮形が提供されます。

照会の処理時に、XQuery は、QName を展開し、ネーム・スペース接頭部にバインドされている URI を解決します。展開された QName には、ネーム・スペース URI およびローカル名が含まれます。2 つの QName は、そのネーム・スペース URI とローカル名が同じである場合に等しくなります。これは、2 つの QName は、異なる接頭部を持っていても、それらが同じネーム・スペース URI にバインドされていれば一致する可能性があるということを意味します。

以下の例には QName が含まれています。

- `ns1:name`
- `ns2:name`
- `name`

この例において、`ns1` は、URI `http://posample.org` にバインドされる接頭部です。接頭部 `ns2` は、URI `http://mycompany.com` にバインドされます。デフォルトのエレメント・ネーム・スペースは、ns1 および ns2 に関連付けられる URI とは異なる別の URI です。3 つのエレメントすべてのローカル名が `name` です。

```
<ns1:name>This text is in an element named "name" that is qualified
by the prefix "ns1".</ns1:name>

<ns2:name>This text is in an element named "name" that is qualified
by the prefix "ns2".</ns2:name>

<name>This text is in an element named "name" that is in the default
element namespace.</name>
```

この例の中のエレメントは、同じローカル名 name を共用しますが、これらのエレメントが異なるネーム・スペースに存在するため、命名競合は発生しません。式の処理時に、名前 ns1:name は、ns1 にバインドされる URI を含む名前と、ローカル名 name に展開されます。同様に、名前 ns2:name は、ns2 にバインドされる URI を含む名前と、ローカル名 name に展開されます。空の接頭部を持つエレメント name は、接頭部が指定されていないため、デフォルト・エレメント・ネーム・スペースにバインドされます。URI にバインドされていない接頭部が名前に使用されている場合、エラーが戻されます。

QName (修飾名) は、W3C 勧告 *Namespaces in XML* で定義される構文に準拠します。

静的に既知のネーム・スペース

ネーム・スペース接頭部は、ネーム・スペース宣言によって URI にバインドされます。照会式における QName の変換処理を制御するこれらのネーム・スペース・バインディングのセットを、静的に既知のネーム・スペースと呼びます。静的に既知のネーム・スペースは照会式のプロパティーであり、式によって処理されるデータとは独立しています。

一部のネーム・スペース接頭部は事前に宣言されています。その他のネーム・スペース接頭部は、照会プロローグまたはエレメント・コンストラクターのどちらの宣言によっても追加できます。 DB2 XQuery には、以下の表で説明されている事前宣言されたネーム・スペース接頭部が含まれています。

表 2. DB2 XQuery の事前宣言されたネーム・スペース

接頭部	URI	説明
xml	http://www.w3.org/XML/1998/namespace	XML 予約済みネーム・スペース
xs	http://www.w3.org/2001/XMLSchema	XML スキーマ・ネーム・スペース
xsi	http://www.w3.org/2001/XMLSchema-instance	XML スキーマ・インスタンス・ネーム・スペース
fn	http://www.w3.org/2005/xpath-functions	デフォルトの関数ネーム・スペース
xdt	http://www.w3.org/2005/xpath-datatypes	XQuery のタイプのネーム・スペース
db2-fn	http://www.ibm.com/xmlns/prod/db2/functions	DB2 関数ネーム・スペース

事前宣言されたネーム・スペースに加えて、以下のようにして、静的に既知のネーム・スペースのセットを提供することができます。

- ネーム・スペース宣言またはデフォルト・ネーム・スペース宣言を使用して、照会プロローグ内で宣言します。以下のネーム・スペース宣言の例では、ネーム・スペース接頭部 ns1 が URI http://mycompany.com と関連付けられます。

  ```
  declare namespace ns1 = "http://mycompany.com";
  ```

  以下のデフォルト・エレメント/タイプ・ネーム・スペース宣言の例では、照会に含まれる接頭部のないエレメント名に URI が設定されます。

  ```
  declare default element namespace "http://posample.org";
  ```

- エレメント・コンストラクターのネーム・スペース宣言属性によって宣言します。以下の例は、構成されるエレメントの有効範囲内で接頭部 ns2 を URI http://mycompany.com にバインドするネーム・スペース宣言属性を含むエレメント・コンストラクターです。

  ```
  <ns2:price xmlns:ns2="http://mycompany.com">14.99</ns2:price>
  ```

- SQL/XML によって提供されます。SQL/XML は、以下のネーム・スペースのセットを提供します。
  - SQL/XML の事前宣言されたネーム・スペース。
  - SQL/XML コンストラクターおよびその他の SQL/XML 式で宣言されるネーム・スペース。

  SQL/XML によって提供されるネーム・スペースは、プロローグ内のネーム・スペース宣言、または後続のエレメント・コンストラクター内のネーム・スペース宣言属性によって指定変更できます。プロローグ内で宣言されたネーム・スペースは、エレメント・コンストラクター内のネーム・スペース宣言属性によって指定変更できます。

言語規則

以下のトピックでは、XQuery の言語規則について説明します。

大/小文字の区別

XQuery は大/小文字を区別する言語です。

XQuery のキーワードには小文字を使用します。予約はされていません。XQuery 式で使用する名前は、言語キーワードと同じにすることができます。

空白

空白は式の構文に含まれませんが、ほとんどの XQuery 式において、読みやすさを向上させるため空白を使用することが可能です。空白は、スペース文字 (X'20')、復帰 (X'0D')、改行 (X'0A')、およびタブ (X'09') で構成されます。

一般に、空白は、空白が保持される以下のような状況を除いては、照会において重要ではありません。

- 空白がストリング・リテラル内にある。
- 空白が、パーサーが 2 つの隣接するトークンを 1 つのトークンとして認識することを防止することにより、式を明確にしている。

- 空白がエレメント・コンストラクター内にある。プロローグ内の境界スペース宣言により、エレメント・コンストラクター内の空白を保持するかあるいは削除するかが判別されます。

例えば、以下の式には明確さのために空白が必要です。

- `name- name` は、エラーになります。パーサーは、 name- を単一の QName (修飾名) と認識し、演算子が見つからない時点でエラーを戻します。
- `name -name` は、エラーになりません。パーサーは、最初の `name` を QName として、負符号 (-) を演算子として、そして 2 番目の `name` を別の QName として認識します。
- `name-name` は、エラーになりません。ただし、QName においてハイフン (-) は有効な文字であるため、式は単一の QName として構文解析されます。
- 以下の式はすべてエラーになります。
  - `10 div3`
  - `10div3`

  これらの式では、パーサーが各トークンを別個に認識できるようにするために、空白が必要です。

コメント

コメントは、プロローグまたは照会の本文で許可されます。コメントは、照会の処理には影響しません。

コメントは、記号 `(:` および `:)` で区切られたストリングから構成されます。以下は、XQuery のコメントの例です。

```
(: A comment. You can use comments to make your code easier to understand. :)
```

DB2 XQuery でのコメントの使用には、以下のような汎用規則が適用されます。

- コメントは、無視可能な空白文字が許可されるいずれの場所でも使用できます。*無視可能な空白文字* とは、式の結果に影響しない空白文字です。
- コメントは、コンストラクターのコンテンツでは許可されません。
- コメント同士はネスト可能ですが、ネストされたコメントは、左右を区切り文字 `(:` および `:)` で囲む必要があります。

以下は、正しいコメントの例、およびエラーになるコメントの例です。

- `(: is this a comment? ::)` は、正しいコメントです。
- `(: is this a comment? ::) or an error? :)` は、エラーになります。原因は、記号 `(:` および `:)` のネストが対になっていないためです。
- `(: commenting out a (: comment :) might be confusing, but often helpful :)` は、正しいコメントです。コメント開始と終了が正しく対になっていればコメントはネストできます。
- `"this is just a string :)"` は正しい式です。
- `(: "this is just a string :)" :)` の結果はエラーになります。同様に、`"this is another string (:"` は、正しい式ですが、 `(: "this is another string (:" :)` はエラーになります。リテラル・コンテンツは、コメントのネストが対でなくても構いません。

XQuery に関する詳細情報の検索先

DB2 XQuery の基礎となる仕様の詳細については、以下のリソースを参照してください。

- **XQuery 1.0**

  World Wide Web Consortium。*XQuery 1.0: An XML Query Language*。W3C Recommendation、23 January 2007。www.w3.org/TR/2007/REC-xquery-20070123/ を参照してください。

- **XQuery 1.0 and XPath 2.0 Functions and Operators**

  World Wide Web Consortium。*XQuery 1.0 and XPath 2.0 Functions and Operators*。 W3C Recommendation、23 January 2007。www.w3.org/TR/2007/REC-xpath-functions-20070123/ を参照してください。

- **XQuery 1.0 and XPath 2.0 Data Model**

  World Wide Web Consortium。*XQuery 1.0 and XPath 2.0 Data Model*。W3C Recommendation、23 January 2007。www.w3.org/TR/2007/REC-xpath-datamodel-20070123/ を参照してください。

- **XML Query Use Cases**

  World Wide Web Consortium。*XML Query Use Cases*。W3C Working Group Note、23 March 2007。www.w3.org/TR/2007/NOTE-xquery-use-cases-20070323/ を参照してください。

- **XML Schema**

  World Wide Web Consortium。*XML Schema*、*Part 0*、*1*、および *2*。W3C Recommendation、28 October 2004。www.w3.org/TR/2004/REC-xmlschema-0-20041028/、 www.w3.org/TR/2004/REC-xmlschema-1-20041028/、および www.w3.org/TR/2004/REC-xmlschema-2-20041028/ を参照してください。

- **XML Names**

  World Wide Web Consortium。*Namespaces in XML 1.0 (Second Edition)*。W3C Recommendation、16 August 2006。www.w3.org/TR/2006/REC-xml-names-20060816/ を参照してください。

- **Updating XML**

  World Wide Web Consortium。*XQuery Update Facility*。W3C Working Draft、11 July 2006。www.w3.org/TR/2006/WD-xqupdate-20060711/ を参照してください。

第 2 章 タイプ・システム

XQuery は、さまざまな式、演算子、および関数のオペランドを期待されるタイプに準拠させなければならない、強く型付けされた言語です。 DB2 XQuery のタイプ・システムには、XML スキーマの組み込みタイプ、および XQuery の事前定義タイプが含まれます。

XML スキーマの組み込みタイプは、ネーム・スペース `http://www.w3.org/2001/XMLSchema` にあり、事前宣言されたネーム・スペース接頭部 `xs` を持ちます。組み込みスキーマ・タイプの例としては、xs:integer、xs:string、および xs:date などがあります。

XQuery の事前定義タイプは、ネーム・スペース `http://www.w3.org/2005/xpath-datatypes` にあり、事前宣言されたネーム・スペース接頭部 `xdt` を持ちます。XQuery の事前定義タイプの例としては、xdt:untypedAtomic、xdt:yearMonthDuration、および xdt:dayTimeDuration などがあります。

各データ・タイプには、字句形式があります。これは、指定されたタイプにキャストできる、またはシリアライゼーション後に指定されたタイプの値を表すために使用できるストリングです。

タイプ階層

DB2 XQuery のタイプ階層は、XQuery 式で使用可能なすべてのタイプを示します。

20 ページの図 3 の階層には、抽象基本タイプおよび派生タイプが含まれています。すべての原子タイプは、データ・タイプ xdt:anyAtomicType から派生します。各派生データ・タイプは、派生元である基本タイプに実線で接続されています。

© Copyright IBM Corp. 2006, 2008

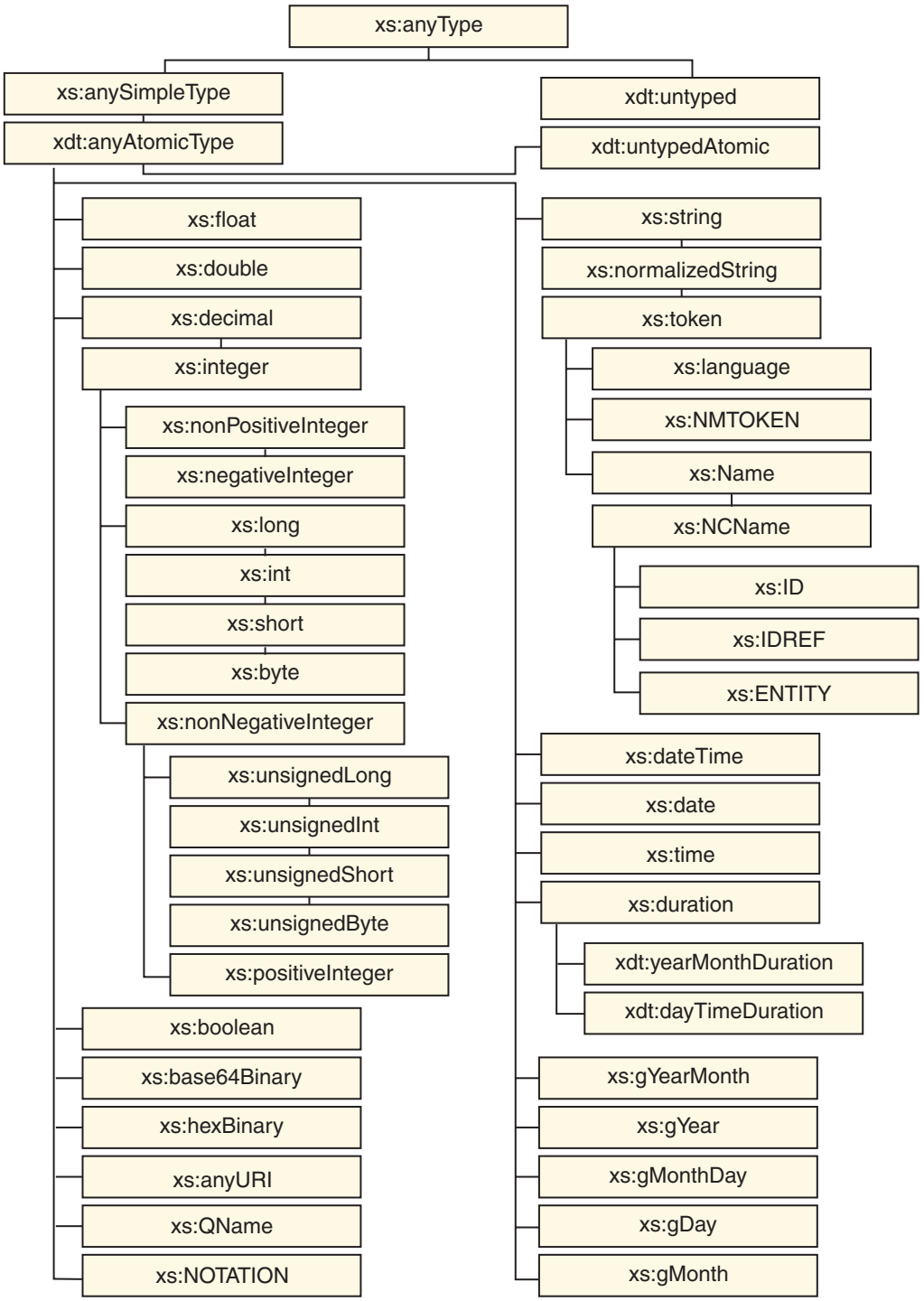


図 3. DB2 XQuery のタイプ階層

カテゴリー別のタイプ

DB2 XQuery には、次のようなタイプのカテゴリーがあります。汎用、非型付き、ストリング、数値、日付、時刻、期間、およびその他です。

汎用タイプ

表 3. 汎用タイプ

タイプ	説明
30 ページの『anyType タイプ』	データ・タイプ xs:anyType は、0 個以上のノードおよび 0 個以上の原子値のシーケンスを包含します。
29 ページの『anySimpleType タイプ』	データ・タイプ xs:anySimpleType は、任意の単純タイプを使用できるコンテキストを示します。このデータ・タイプは、すべての単純タイプの基本タイプとなります。単純タイプのインスタンスとして、原子値の任意のシーケンスが可能です。データ・タイプ xs:anyType から派生。
29 ページの『anyAtomicType タイプ』	データ・タイプ xdt:anyAtomicType は、原子タイプを使用できるコンテキストを表します。このデータ・タイプは、すべての原子タイプの基本タイプとなります。原子タイプのインスタンスは、整数、ストリング、または日付などの単一の分解不可能な値です。データ・タイプ xs:anySimpleType から派生。

非型付きデータ・タイプ

表 4. 非型付きデータ・タイプ

タイプ	説明
44 ページの『untyped タイプ』	データ・タイプ xdt:untyped は、 XML スキーマによって妥当性検査されていないノードを表します。データ・タイプ xs:anyType から派生。
45 ページの『untypedAtomic タイプ』	データ・タイプ xdt:untypedAtomic は、XML スキーマによって妥当性検査されていない原子値を表します。データ・タイプ xdt:anyAtomicType から派生。

ストリング・タイプ

表 5. ストリング・タイプ

タイプ	説明
43 ページの『string タイプ』	データ・タイプ xs:string は、文字ストリングを表します。データ・タイプ xdt:anyAtomicType から派生。
42 ページの『normalizedString タイプ』	データ・タイプ xs:normalizedString は、空白が正規化処理されたストリングを表します。データ・タイプ xs:string から派生。
44 ページの『token タイプ』	データ・タイプ xs:token は、トークン化されたストリングを表します。データ・タイプ xs:normalizedString から派生。

表 5. ストリング・タイプ (続き)

タイプ	説明
40 ページの『language タイプ』	データ・タイプ xs:language は、*RFC 3066* によって定義されている自然言語 ID を表します。データ・タイプ xs:token から派生。
41 ページの『NMTOKEN タイプ』	データ・タイプ xs:NMTOKEN は、*XML 1.0 (Third Edition)* の NMTOKEN 属性のタイプを表します。データ・タイプ xs:token から派生。
40 ページの『Name タイプ』	データ・タイプ xs:Name は、XML 名を表します。データ・タイプ xs:token から派生。
41 ページの『NCName タイプ』	データ・タイプ xs:NCName は、コロンのない XML 名を表します。データ・タイプ xs:Name から派生。
39 ページの『ID タイプ』	データ・タイプ xs:ID は、*XML 1.0 (Third Edition)* の ID 属性のタイプを表します。データ・タイプ xs:NCName から派生。
39 ページの『IDREF タイプ』	データ・タイプ xs:IDREF は、*XML 1.0 (Third Edition)* の IDREF 属性のタイプを表します。データ・タイプ xs:NCName から派生。
36 ページの『ENTITY タイプ』	データ・タイプ xs:ENTITY は、*XML 1.0 (Third Edition)* の ENTITY 属性のタイプを表します。データ・タイプ xs:NCName から派生。

数値タイプ

表 6. 数値タイプ

タイプ	説明
34 ページの『decimal タイプ』	データ・タイプ xs:decimal は、10 進数で表現可能な実数のサブセットを表します。データ・タイプ xdt:anyAtomicType から派生。
36 ページの『float タイプ』	データ・タイプ xs:float は、IEEE 単精度 32 ビット浮動小数点タイプを基盤にしています。データ・タイプ xdt:anyAtomicType から派生。
34 ページの『double タイプ』	データ・タイプ xs:double は、IEEE 倍精度 64 ビット浮動小数点タイプを基盤にしています。データ・タイプ xdt:anyAtomicType から派生。
40 ページの『int タイプ』	データ・タイプ xs:int は、2147483647 以下、-2147483648 以上の整数を表します。データ・タイプ xs:long から派生。
41 ページの『nonPositiveInteger タイプ』	データ・タイプ xs:nonPositiveInteger は、ゼロ以下の整数を表します。データ・タイプ xs:integer から派生。

表 6. 数値タイプ (続き)

タイプ	説明
41 ページの『negativeInteger タイプ』	データ・タイプ xs:negativeInteger は、ゼロ未満の整数を表します。データ・タイプ xs:nonPositiveInteger から派生。
41 ページの『nonNegativeInteger タイプ』	データ・タイプ xs:nonNegativeInteger は、ゼロ以上の整数を表します。データ・タイプ xs:integer から派生。
40 ページの『long タイプ』	データ・タイプ xs:long は、9223372036854775807 以下、-9223372036854775808 以上の整数を表します。データ・タイプ xs:integer から派生。
40 ページの『integer タイプ』	データ・タイプ xs:integer は、9223372036854775807 以下、-9223372036854775808 以上の数値を表します。データ・タイプ xs:decimal から派生。
43 ページの『short タイプ』	データ・タイプ xs:short は、32767 以下、および -32768 以上の整数を表します。データ・タイプ xs:int から派生。
30 ページの『byte タイプ』	データ・タイプ xs:byte は、127 以下、 -128 以上の整数を表します。データ・タイプ xs:short から派生。
44 ページの『unsignedLong タイプ』	データ・タイプ xs:unsignedLong は、9223372036854775807 以下の符号なし整数を表します。データ・タイプ xs:nonNegativeInteger から派生。
44 ページの『unsignedInt タイプ』	データ・タイプ xs:unsignedInt は、4294967295 以下の符号なし整数を表します。データ・タイプ xs:unsignedLong から派生。
44 ページの『unsignedShort タイプ』	データ・タイプ xs:unsignedShort は、65535 以下の符号なし整数を表します。データ・タイプ xs:unsignedInt から派生。
44 ページの『unsignedByte タイプ』	データ・タイプ xs:unsignedByte は、 255 以下の符号なし整数を表します。データ・タイプ xs:unsignedShort から派生。
42 ページの『positiveInteger タイプ』	データ・タイプ xs:positiveInteger は、1 以上の正の整数を表します。データ・タイプ xs:nonNegativeInteger から派生。

日付タイプ、時間タイプ、および期間タイプ

表 7. 日付タイプ、時間タイプ、および期間タイプ

タイプ	説明
35 ページの『duration タイプ』	データ・タイプ xs:duration は、グレゴリオ暦の年、月、日、時、分、および秒コンポーネントで表される時間を表します。データ・タイプ xdt:anyAtomicType から派生。

表 7. 日付タイプ、時間タイプ、および期間タイプ (続き)

タイプ	説明
45 ページの『yearMonthDuration タイプ』	データ・タイプ xdt:yearMonthDuration は、グレゴリオ暦の年および月コンポーネントで表される時間を表します。データ・タイプ xs:duration から派生。
33 ページの『dayTimeDuration タイプ』	データ・タイプ xdt:dayTimeDuration は、日数、時間数、分数、および秒数コンポーネントで表される時間を表します。データ・タイプ xs:duration から派生。
31 ページの『dateTime タイプ』	データ・タイプ xs:dateTime は、以下のプロパティーを持つインスタンスを表します。整数値として表される年、月、日、時、および分プロパティー、10 進数値として表される秒プロパティー、およびオプションの時間帯標識です。データ・タイプ xdt:anyAtomicType から派生。
31 ページの『date タイプ』	データ・タイプ xs:date は、指定された日の最初の瞬間から始まる正確に 1 日の間隔を表します。データ・タイプ xs:date は、整数値として表される年、月、および日プロパティーと、オプションの時間帯標識で構成されます。データ・タイプ xdt:anyAtomicType から派生。
43 ページの『time タイプ』	データ・タイプ xs:time は、毎日定期的に繰り返されるある一瞬の時刻を表します。データ・タイプ xdt:anyAtomicType から派生。
39 ページの『gYearMonth タイプ』	データ・タイプ xs:gYearMonth は、特定のグレゴリオ暦の年の特定のグレゴリオ暦の月を表します。グレゴリオ暦の月は、 *ISO 8601* で定義されています。データ・タイプ xdt:anyAtomicType から派生。
38 ページの『gYear タイプ』	データ・タイプ xs:gYear は、グレゴリオ暦の年を表します。グレゴリオ暦の年は、*ISO 8601* で定義されています。データ・タイプ xdt:anyAtomicType から派生。
38 ページの『gMonthDay タイプ』	データ・タイプ xs:gMonthDay は、定期的に繰り返されるグレゴリオ暦の日付を表します。グレゴリオ暦の日付は *ISO 8601* で定義されています。データ・タイプ xdt:anyAtomicType から派生。
37 ページの『gDay タイプ』	データ・タイプ xs:gDay は、定期的に繰り返されるグレゴリオ暦の日を表します。グレゴリオ暦の日は、*ISO 8601* で定義されています。データ・タイプ xdt:anyAtomicType から派生。

表 7. 日付タイプ、時間タイプ、および期間タイプ (続き)

タイプ	説明
37 ページの『gMonth タイプ』	データ・タイプ xs:gMonth は、毎年定期的に繰り返されるグレゴリオ暦の月を表します。グレゴリオ暦の月は、 *ISO 8601* で定義されています。データ・タイプ xdt:anyAtomicType から派生。

その他のデータ・タイプ

表 8. その他のデータ・タイプ

タイプ	説明
30 ページの『ブール・データ・タイプ』	データ・タイプ xs:boolean は、数学上の 2 値論理 (true または false) の概念をサポートします。データ・タイプ xdt:anyAtomicType から派生。
30 ページの『anyURI タイプ』	データ・タイプ xs:anyURI は、Uniform Resource Identifier (URI) を表します。データ・タイプ xdt:anyAtomicType から派生。
42 ページの『QName タイプ』	データ・タイプ xs:QName は、XML 修飾名 (QName) を表します。QName には、オプションのネーム・スペース接頭部、XML ネーム・スペースを識別する URI、およびローカル部分 (NCName) が含まれます。データ・タイプ xdt:anyAtomicType から派生。
42 ページの『NOTATION タイプ』	データ・タイプ xs:NOTATION は、 *XML 1.0 (Third Edition)* の NOTATION 属性タイプを表します。データ・タイプ xdt:anyAtomicType から派生。
39 ページの『hexBinary タイプ』	データ・タイプ xs:hexBinary は、16 進数にエンコードされたバイナリー・データを表します。データ・タイプ xdt:anyAtomicType から派生。
30 ページの『base64Binary タイプ』	データ・タイプ xs:base64Binary は、Base64 でエンコードされたバイナリー・データを表します。データ・タイプ xdt:anyAtomicType から派生。

組み込みデータ・タイプのコンストラクター関数

コンストラクター関数 は、特定の原子タイプのインスタンスを、異なる原子タイプのインスタンスに変換します。XML スキーマで定義されている各組み込み原子タイプについて、暗黙的に定義されたコンストラクター関数が存在します。データ・タイプ xdt:untypedAtomic、および 2 つの派生データ・タイプ xdt:yearMonthDuration および xdt:dayTimeDuration についても、コンストラクター関数が存在します。

xs:NOTATION、xs:anyType、xs:anySimpleType、または xdt:anyAtomicType について使用可能なコンストラクター関数はありません。

組み込みタイプのすべてのコンストラクター関数は、以下の汎用構文を共用します。

注: コンストラクター関数 *type-name*(*value*) のセマンティクスは、式 (*value* cast as *type-name*?) と等価になるように定義されています。

type-name
ターゲット・データ・タイプの QName。

value
ターゲット・データ・タイプのインスタンスとして構成される値。この値に原子化が適用されます。原子化の結果が空のシーケンスである場合、空のシーケンスが戻されます。原子化の結果が複数項目のシーケンスの場合、エラーが発生します。その他の場合、結果の原子値がターゲット・タイプにキャストされます。どのタイプを別のどのタイプにキャストできるかについては、 27 ページの『タイプ・キャスト』を参照してください。

例えば、以下の図は、XML スキーマ・データ・タイプ xs:unsignedInt のコンストラクター関数の構文を表しています。

このコンストラクター関数に渡すことのできる値は、ターゲット・データ・タイプに有効にキャストできる原子値です。例えば、この関数を以下のように呼び出すと、同じ結果の xs:unsignedInt 値 12 が戻されます。

```
xs:unsignedInt(12)
xs:unsignedInt("12")
```

最初の例では、数値リテラル 12 がコンストラクター関数に渡されます。このリテラルには小数点が含まれないため、これは xs:integer として構文解析され、xs:integer 値は xs:unsignedInt タイプにキャストされます。2 番目の例では、ストリング・リテラル ″12″ がコンストラクター関数に渡されます。ストリング・リテラルは xs:string として構文解析され、 xs:string 値は xs:unsignedInt タイプにキャストされます。

コンストラクター関数は、ノードを使用してその引数として呼び出すこともできます。この場合、DB2 XQuery は、ノードを原子化してその型付き値を抽出し、その値を使用してコンストラクターを呼び出します。コンストラクターに渡された値をターゲット・データ・タイプにキャストできない場合は、エラーが戻されます。

xs:QName のコンストラクター関数は、その引数として必ずストリング・リテラルを使用しなければならない点で、コンストラクター関数の一般構文とは異なります。

値をデータ・タイプにキャストする際には、キャスト可能な式を使用して、値をデータ・タイプにキャストできるかどうかテストできます。

タイプ・キャスト

xdt:untypedAtomic、xs:integer、xs:duration の 2 つの派生タイプ (xdt:yearMonthDuration および xdt:dayTimeDuration)、および XML スキーマで定義されている 19 個のプリミティブ・タイプの間でのタイプ変換がサポートされています。タイプ変換は、キャスト式およびタイプ・コンストラクターで使用されます。

サポートされているタイプ変換は、以下の表のとおりです。各表の左側にタイプ変換のソースとなるプリミティブ・タイプを、上部にタイプ変換のターゲットとなるプリミティブ・タイプを示しています。最初の表には xdt:untypedAtomic から xs:dateTime までのターゲットが記載され、2 番目の表には xs:time から xs:NOTATION までのターゲットが記載されています。

表内のセルには、以下の 3 つの文字のうち 1 つが記載されます。

Y はい。ソース・タイプからターゲット・タイプへの値の変換がサポートされていることを示します。

N いいえ。ソース・タイプからターゲット・タイプへの値の変換がサポートされていないことを示します。

M 場合による。ソース・タイプからターゲット・タイプへの値の変換が、値によって成功する場合と失敗する場合があることを示します。

xs:anySimpleType または xdt:anyAtomicType は、キャスト先としてもキャスト元としてもサポートされません。

サポートされていないキャストが試行されると、エラーが戻されます。

表 9. プリミティブ・タイプ・キャスト (1) (xdt:untypedAtomic から xs:dateTime までのターゲット)

ソース・データ・タイプ	ターゲット・データ・タイプ									
	uA	**string**	**float**	**double**	**decimal**	**integer**	**dur**	**yMD**	**dTD**	**dT**
uA	Y	Y	M	M	M	M	M	M	M	M
string	Y	Y	M	M	M	M	M	M	M	M
float	Y	Y	Y	Y	M	M	N	N	N	N
double	Y	Y	M	Y	M	M	N	N	N	N
decimal	Y	Y	Y	Y	Y	M	N	N	N	N
integer	Y	Y	Y	Y	Y	Y	N	N	N	N
dur	Y	Y	N	N	N	N	Y	Y	Y	N
yMD	Y	Y	N	N	N	N	Y	Y	N	N
dTD	Y	Y	N	N	N	N	Y	N	Y	N
dT	Y	Y	N	N	N	N	N	N	N	Y
time	Y	Y	N	N	N	N	N	N	N	N
date	Y	Y	N	N	N	N	N	N	N	Y
gYM	Y	Y	N	N	N	N	N	N	N	N
gYr	Y	Y	N	N	N	N	N	N	N	N
gMD	Y	Y	N	N	N	N	N	N	N	N

表 9. プリミティブ・タイプ・キャスト (1) (xdt:untypedAtomic から xs:dateTime までのターゲット) (続き)

ソース・データ・タイプ	ターゲット・データ・タイプ									
	uA	string	float	double	decimal	integer	dur	yMD	dTD	dT
gDay	Y	Y	N	N	N	N	N	N	N	N
gMon	Y	Y	N	N	N	N	N	N	N	N
bool	Y	Y	Y	Y	Y	Y	N	N	N	N
b64	Y	Y	N	N	N	N	N	N	N	N
hxB	Y	Y	N	N	N	N	N	N	N	N
aURI	Y	Y	N	N	N	N	N	N	N	N
QN	Y	Y	N	N	N	N	N	N	N	N
NOT	Y	Y	N	N	N	N	N	N	N	N

表 10. プリミティブ・タイプ・キャスト (2) (xs:time から xs:NOTATION までのターゲット)

ソース・データ・タイプ	ターゲット・データ・タイプ												
	time	date	gYM	gYr	gMD	gDay	gMon	bool	b64	hxB	aURI	QN	NOT
uA	M	M	M	M	M	M	M	M	M	M	M	N	N
string	M	M	M	M	M	M	M	M	M	M	M	M	M
float	N	N	N	N	N	N	N	Y	N	N	N	N	N
double	N	N	N	N	N	N	N	Y	N	N	N	N	N
decimal	N	N	N	N	N	N	N	Y	N	N	N	N	N
integer	N	N	N	N	N	N	N	Y	N	N	N	N	N
dur	N	N	N	N	N	N	N	N	N	N	N	N	N
yMD	N	N	N	N	N	N	N	N	N	N	N	N	N
dTD	N	N	N	N	N	N	N	N	N	N	N	N	N
dT	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N
time	Y	N	N	N	N	N	N	N	N	N	N	N	N
date	N	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N
gYM	N	N	Y	N	N	N	N	N	N	N	N	N	N
gYr	N	N	N	Y	N	N	N	N	N	N	N	N	N
gMD	N	N	N	N	Y	N	N	N	N	N	N	N	N
gDay	N	N	N	N	N	Y	N	N	N	N	N	N	N
gMon	N	N	N	N	N	N	Y	N	N	N	N	N	N
bool	N	N	N	N	N	N	N	Y	N	N	N	N	N
b64	N	N	N	N	N	N	N	N	Y	Y	N	N	N
hxB	N	N	N	N	N	N	N	N	Y	Y	N	N	N
aURI	N	N	N	N	N	N	N	N	N	N	Y	N	N
QN	N	N	N	N	N	N	N	N	N	N	N	N	N
NOT	N	N	N	N	N	N	N	N	N	N	N	N	M

列と行に記載されている短いコードは、以下のタイプを示します。

- uA = xdt:untypedAtomic

- string = xs:string
- float = xs:float
- double = xs:double
- decimal = xs:decimal
- integer = xs:integer
- dur = xs:duration
- yMD = xdt:yearMonthDuration
- dTD = xdt:dayTimeDuration
- dT = xs:dateTime
- time = xs:time
- date = xs:date
- gYM = xs:gYearMonth
- gYr = xs:gYear
- gMD = xs:gMonthDay
- gDay = xs:gDay
- gMon = xs:gMonth
- bool = xs:boolean
- b64 = xs:base64Binary
- hxB = xs:hexBinary
- aURI = xs:anyURI
- QN = xs:QName
- NOT = xs:NOTATION

anyAtomicType タイプ

データ・タイプ xdt:anyAtomicType は、原子タイプを使用できるコンテキストを表します。このデータ・タイプは、すべての原子タイプの基本タイプとなります。原子タイプのインスタンスは、整数、ストリング、または日付などの単一の分解不可能な値です。データ・タイプ xs:anySimpleType から派生。

データ・タイプ xdt:anyAtomicType は制約されない字句形式を持ちます。

データ・タイプ xdt:anyAtomicType からのキャストも、xdt:anyAtomicType へのキャストもサポートされません。

anySimpleType タイプ

データ・タイプ xs:anySimpleType は、任意の単純タイプを使用できるコンテキストを示します。このデータ・タイプは、すべての単純タイプの基本タイプとなります。単純タイプのインスタンスとして、原子値の任意のシーケンスが可能です。データ・タイプ xs:anyType から派生。

データ・タイプ xs:anySimpleType は制約されない字句形式を持ちます。

データ・タイプ xs:anySimpleType からのキャストも、xs:anySimpleType へのキャストもサポートされません。

anyType タイプ

データ・タイプ xs:anyType は、0 個以上のノードおよび 0 個以上の原子値のシーケンスを包含します。

anyURI タイプ

データ・タイプ xs:anyURI は、Uniform Resource Identifier (URI) を表します。データ・タイプ xdt:anyAtomicType から派生。

データ・タイプ xs:anyURI の字句形式は、*RFC 2396* で定義され、改訂が *RFC 2732* で示されているストリング (合法 URI) です。この値にはスペースを使用しないでください。使用する場合は、「%20」とエンコードしてください。

base64Binary タイプ

データ・タイプ xs:base64Binary は、Base64 でエンコードされたバイナリー・データを表します。データ・タイプ xdt:anyAtomicType から派生。

Base64 エンコード・バイナリー・データでは、バイナリー・ストリーム全体が Base64 アルファベットを使用してエンコードされます。Base64 アルファベットについては、*RFC 2045* を参照してください。

xs:base64Binary の字句形式は、*RFC 2045* で定義されている Base64 アルファベットの 65 文字に制限されています。有効な文字には、a から z、A から Z、0 から 9、正符号 (+)、スラッシュ (/)、等号 (=)、および *XML 1.0 (Third Edition)* に空白文字として定義されている文字が含まれます。それ以外の文字は使用できません。

ブール・データ・タイプ

データ・タイプ xs:boolean は、数学上の 2 値論理 (true または false) の概念をサポートします。データ・タイプ xdt:anyAtomicType から派生。

データ・タイプ xs:boolean の字句形式は、true、false、1、および 0 に制限されます。

byte タイプ

データ・タイプ xs:byte は、127 以下、 -128 以上の整数を表します。データ・タイプ xs:short から派生。

xs:byte の字句形式は、オプションの符号と、それに続く 10 進数の有限長シーケンスです。符号を省略した場合は、正符号 (+) が想定されます。このデータ・タイプの有効な例としては、`-1`、`0`、`126`、`+100` があります。

date タイプ

データ・タイプ xs:date は、指定された日の最初の瞬間から始まる正確に 1 日の間隔を表します。データ・タイプ xs:date は、整数値として表される年、月、および日プロパティーと、オプションの時間帯標識で構成されます。データ・タイプ xdt:anyAtomicType から派生。

xs:date タイプの時間帯値は、時間帯で設定されているように 1 日の開始の瞬間を追跡します。 1 日の開始の瞬間は、00:00:00 に始まり、その 1 日は 24:00:00 (この時間は含まない) まで続きます。この 24:00:00 は、次の日の最初の瞬間でもあります。例えば、日付 2002-10-10+13:00 の最初の瞬間は、値 2002-10-10T00:00:00+13:00 です。この値は、2002-10-09T11:00:00Z と同じであり、2002-10-09-11:00 の最初の瞬間でもあります。よって、値 2002-10-10+13:00 と 2002-10-09-11:00 は同じ間隔を示します。

xs:date の字句形式は、*yyyy-mm-ddzzzzzz* 形式の有限長シーケンスです。負の日付は使用できません。以下の省略形はこの形式を説明しています。

yyyy
: 年を示す 4 桁の数表示。有効な値は、0001 から 9999 です。正符号 (+) は許可されていません。

mm
: 月を示す 2 桁の数表示。

dd 日を示す 2 桁の数表示。

zzzzzz
: オプション。存在する場合、時間帯を示します。このプロパティーの形式について詳しくは、 32 ページの『時間帯標識』を参照してください。

dateTime タイプ

データ・タイプ xs:dateTime は、以下のプロパティーを持つインスタンスを表します。整数値として表される年、月、日、時、および分プロパティー、10 進数値として表される秒プロパティー、およびオプションの時間帯標識です。データ・タイプ xdt:anyAtomicType から派生。

xs:dateTime の有効な字句表記は、明示的な時間帯を意味しないことがあります。明示的な時間帯を意味しない表記の場合、UTC (協定世界時。グリニッジ標準時とも呼ばれます) の暗黙的時間帯が使用されます。数値として表される各プロパティーは、次に高いプロパティーによって設定された間隔の最大値に制限されます。例えば、日付値は、32 になることはできません。また、月が 02 で年が 2002 (2002 年 2 月) の場合は、29 の日付値もありません。

xs:dateTime の字句形式は、*yyyy-mm-dd*T*hh*:*mm*:*ss*.*ssssssszzzzzz* 形式の有限長シーケンスです。負の日付は使用できません。以下の省略形はこの形式を説明しています。

yyyy
: 年を示す 4 桁の数表示。有効な値は、0001 から 9999 です。正符号 (+) は許可されていません。

- 日付部の部分間の区切り記号です。

mm
月を示す 2 桁の数表示。

dd 日を示す 2 桁の数表示。

T 後に時刻が続くことを示す区切り記号。

hh 時間を示す 2 桁の数表示。24 の値は、表記される分と秒がゼロの場合にのみ許可されます。時刻 24:00:00 を含む照会は、次の日の 00:00:00 として処理されます。

: 時刻部の各部分間の区切り記号。

mm
分を示す 2 桁の数表示。

ss すべての秒を示す 2 桁の数表示。

.ssssss
オプション。存在する場合、秒を小数で示す 1 から 6 桁の数表示。

zzzzzz
オプション。存在する場合、時間帯を示します。このプロパティーの形式について詳しくは、『時間帯標識』を参照してください。

例えば、以下の形式は、米国の東部標準時 (EST) の 2005 年 10 月 10 日正午を示します。

```
2005-10-10T12:00:00-05:00
```

この時刻は、UTC では `2002-10-10T17:00:00Z` と表記されます。

時間帯標識

時間帯標識の字句形式は、以下のいずれかの形式を含むストリングです。

- `hh:mm` の前にある正符号 (+) または負符号 (-) は、次の省略形が使用されます。

  hh 時間を示す 2 桁の数表示 (必要に応じて先行ゼロが付きます)。現在、正式に既定された時間帯には、24 時間を超える期間はありません。これにより、時間プロパティーに値 24 が許可されるのは、分のプロパティーの値がゼロである場合のみです。

  mm
  分を示す 2 桁の数表示。時間プロパティーが 24 の場合、分のプロパティーの値はゼロでなければなりません。

  + 指定された時刻インスタンスが、*hh* 時 *mm* 分、UTC 時刻より前の時間帯にあることを示します。

  - 指定された時刻インスタンスが *hh* 時 *mm* 分、UTC 時刻より後の時間帯にあることを示します。
- リテラル `Z` は、UTC の時刻を示します (`Z` は、ズールー時を示しますが、これは UTC と同じです)。時間帯に Z を指定することは、`+00:00` または `-00:00` を指定することと同じです。

dayTimeDuration タイプ

データ・タイプ xdt:dayTimeDuration は、日数、時間数、分数、および秒数コンポーネントで表される時間を表します。データ・タイプ xs:duration から派生。

このデータ・タイプで表すことができる範囲は、
-P83333333333333Y3M11574074074DT1H46M39.999999S から
P83333333333333Y3M11574074074DT1H46M39.999999S です (または
-999999999999999 カ月 -999999999999999.999999 秒から 999999999999999 カ月
999999999999999.999999 秒)。

xdt:dayTimeDuration の字句形式は、*ISO 8601* 形式を省略した形式の `PnDTnHnMnS` です。以下の省略形はこの形式を説明しています。

P 期間指定子。

***n*D** *n* は日数を示す符号なしの整数です。

T 日時の分離文字。

***n*H**
n は時間数を示す符号なしの整数です。

***n*M**
n は分数を示す符号なしの整数です。

***n*S** *n* は秒数を示す符号なしの 10 進数です。小数点を指定する場合、小数秒を示す 1 から 6 桁の数字が小数点の後に必要です。

例えば、以下の形式は、3 日間、10 時間と 30 分間を示します。

```
P3DT10H30M
```

以下の形式は、マイナス 120 日間を示します。

```
-P120D
```

先行するオプションの負符号 (-) は、負の期間を示します。符号を省略すると、正の期間と認識されます。

この形式の精度を低減し、表記を切り捨てることが可能ですが、以下の要件に準拠する必要があります。

- 任意の式の日数、時間数、分数、秒数がゼロである場合、その数値と対応する指定子は省略できます。ただし、最低 1 個の数値および指定子が存在する必要があります。
- 2 番目の部分には、小数部を使用できます。
- 時間項目がなければ指定子 `T` を省略する必要があり、`T` を省略する必要があるのはその場合に限ります。指定子 `P` は常に必要です。

例えば、以下の形式は使用可能です。

```
P13D
PT47H
P3DT2H
-PT35.89S
P4DT251M
```

P-134D の形式は使用できませんが、-P1347D の形式は使用できます。

DB2 は、xdt:dayTimeDuration 値を正規化された形式で保管します。正規化された形式では、構成要素の秒と分は 60 より小さく、時間は 24 より小さくなります。各 60 秒が 1 分に、各 60 分が 1 時間に、および各 24 時間が 1 日に変換されます。例えば、以下の XQuery 式は dayTimeDuration が 63 日、55 時間、および 81 秒に指定されているコンストラクター関数を呼び出します。

```
xquery
xdt:dayTimeDuration("P63DT55H81S")
```

この期間では、55 時間は 2 日と 7 時間に、81 秒は 1 分と 21 秒にそれぞれ変換されます。この式は、正規化された dayTimeDuration 値 P65DT7H1M21S を戻します。

decimal タイプ

データ・タイプ xs:decimal は、10 進数で表現可能な実数のサブセットを表します。データ・タイプ xdt:anyAtomicType から派生。

xs:decimal の字句形式は、ピリオドにより小数部標識として区切られた小数桁数の有限長シーケンスです。状況に応じて符号を前に置くことが許可されています。符号を省略した場合は、正符号 (+) が想定されます。前後にゼロを置くことは任意です。小数部分がゼロの場合、ピリオドおよびその後に置くゼロは省略できます。次の数値は、このデータ・タイプの有効な例です。

```
-1.23
12678967.543233
+100000.00
210
```

double タイプ

データ・タイプ xs:double は、IEEE 倍精度 64 ビット浮動小数点タイプを基盤にしています。データ・タイプ xdt:anyAtomicType から派生。

xs:double の基本値のスペースは、 -1.7976931348623158e+308 から -2.2250738585072014e-308 までの範囲、および +2.2250738585072014e-308 から +1.7976931348623158e+308 までの範囲の値で構成されます。xs:double の値スペースには、特殊値 (正の無限大、負の無限大、正のゼロ、負のゼロ、および非数値 (NaN)) も含まれます。

xs:double の字句形式は、小数部ですが、状況に応じて後ろに文字 E または e、その後に指数が続きます。指数は整数でなければなりません。小数部は、10 進数でなければなりません。指数および小数部の表記は、xs:integer および xs:decimal の字句規則に従う必要があります。続く E または e および指数が省略されると、指数値は 0 と想定されます。

ゼロの字句形式には、正符号または負符号を使用できます。リテラル -1E4、1267.43233E12、12.78e-2、12、-0、および 0 はこのデータ・タイプの有効な例です。

特殊値 (正の無限大、負の無限大、および非数値) の字句形式は、それぞれ INF、-INF、および NaN です。正の無限大の字句形式には、正符号は使用できません。

ヒント: 特殊値 INF、 -INF、および NaN のリテラルはありません。xs:double タイプ・コンストラクターを使用することで、ストリングから値 INF、-INF、および NaN を構成します。例: `xs:double("INF")`。

duration タイプ

データ・タイプ xs:duration は、グレゴリオ暦の年、月、日、時、分、および秒コンポーネントで表される時間を表します。データ・タイプ xdt:anyAtomicType から派生。

このデータ・タイプで表すことができる範囲は、
-P83333333333333Y3M11574074074DT1H46M39.999999S から
P83333333333333Y3M11574074074DT1H46M39.999999S です (または
-999999999999999 カ月 -999999999999999.999999 秒から 999999999999999 カ月
999999999999999.999999 秒)。

xs:duration の字句形式は、*ISO 8601* 拡張フォーマット `PnYnMnDTnHnMnS` です。以下の省略形は、拡張フォーマットを説明しています。

P 期間指定子。

*n***Y** *n* は年数を示す符号なしの整数です。

*n***M**
n は月数を示す符号なしの整数です。

*n***D** *n* は日数を示す符号なしの整数です。

T 日時の分離文字。

*n***H**
n は時間数を示す符号なしの整数です。

*n***M**
n は分数を示す符号なしの整数です。

*n***S** *n* は秒数を示す符号なしの 10 進数です。小数点を指定する場合、小数秒を示す 1 から 6 桁の数字が小数点の後に必要です。

例えば、次の形式は、1 年と 2 カ月 3 日 10 時間 30 分の期間を示します。

```
P1Y2M3DT10H30M
```

以下の形式は、マイナス 120 日間を示します。

```
-P120D
```

先行するオプションの負符号 (-) は、負の期間を示します。符号を省略すると、正の期間と認識されます。

この形式の精度を低減し、表記を切り捨てることが可能ですが、以下の要件に準拠する必要があります。

- 式の年数、月数、日数、時間数、分数、または秒数がゼロである場合、数値とそれに対応する指定子は省略できます。ただし、最低 1 個の数値および指定子が存在する必要があります。
- 2 番目の部分には、小数部を使用できます。
- 時間項目がなければ指定子 T を省略する必要があり、T を省略する必要があるのはその場合に限ります。
- 指定子 P は常に必要です。

例えば、以下の形式は使用可能です。

```
P1347Y
P1347M
P1Y2MT2H
P0Y1347M
P0Y1347M0D
```

P1Y2MT の形式は、時間項目がないため、使用できません。形式 P-1347M は使用できませんが、形式 -P1347M は使用できます。

DB2 は、xs:duration 値を正規化された形式で保管します。正規化された形式では、構成要素の秒と分は 60 より小さく、時間は 24 より小さく、月は 12 より小さくなります。各 60 秒が 1 分に、各 60 分が 1 時間に、各 24 時間が 1 日に、および各 12 カ月が 1 年に各々変換されます。例えば、以下の XQuery 式は期間が 2 カ月、63 日、55 時間、および 91 分に指定されているコンストラクター関数を呼び出します。

```
xquery
xs:duration("P2M63DT55H91M")
```

この期間では、55 時間は 2 日と 7 時間に、91 分は 1 時間と 31 分にそれぞれ変換されます。この式は、正規化された期間値 P2M65DT8H31M を戻します。

ENTITY タイプ

データ・タイプ xs:ENTITY は、*XML 1.0 (Third Edition)* の ENTITY 属性のタイプを表します。データ・タイプ xs:NCName から派生。

xs:ENTITY の字句形式は、コロンを含まない XML 名です (NCName)。

float タイプ

データ・タイプ xs:float は、IEEE 単精度 32 ビット浮動小数点タイプを基盤にしています。データ・タイプ xdt:anyAtomicType から派生。

xs:float の基本値のスペースは、 -3.4028234663852886e+38 から -1.1754943508222875e-38 までの範囲、および +1.1754943508222875e-38 から +3.4028234663852886e+38 までの範囲の値で構成されます。xs:float の値スペースには、特殊値 (正の無限大、負の無限大、正のゼロ、負のゼロ、非数値 (NaN)) も含まれます。

xs:float の字句形式は、小数部ですが、状況に応じて後ろに文字 E または e、その後に指数が続きます。指数は整数でなければなりません。小数部は、10 進数でなけ

ればなりません。指数および小数部の表記は、xs:integer および xs:decimal の字句規則に従う必要があります。続く E または e および指数が省略されると、指数値は 0 と想定されます。

ゼロの字句形式には、正符号または負符号を使用できます。リテラル -1E4、1267.43233E12、12.78e-2、12、-0、および 0 はこのデータ・タイプの有効な例です。

特殊値 (正の無限大、負の無限大、および非数値) の字句形式は、それぞれ INF、-INF、および NaN です。正の無限大の字句形式には、正符号は使用できません。

ヒント: 特殊値 INF、 -INF、および NaN のリテラルはありません。xs:float タイプ・コンストラクターを使用することで、ストリングから値 INF、-INF、および NaN を構成します。例: xs:float("INF")。

gDay タイプ

データ・タイプ xs:gDay は、定期的に繰り返されるグレゴリオ暦の日を表します。グレゴリオ暦の日は、*ISO 8601* で定義されています。データ・タイプ xdt:anyAtomicType から派生。

このデータ・タイプは、月の中の特定の日を表します。例えば、給料日が各月の 15 日であることを示すためにこのデータ・タイプを使用できます。

xs:gDay の字句形式は *---ddzzzzzz* で、これは、月または年プロパティーを含まない、xs:date の省略表記です。先行する符号は使用できません。他のどのような形式も使用できません。以下の省略形はこの形式を説明しています。

dd 日を示す 2 桁の数表示。

zzzzzz
オプション。存在する場合、時間帯を示します。このプロパティーの形式について詳しくは、32 ページの『時間帯標識』を参照してください。

例えば、以下の形式は月の 16 日を示します。これは毎月繰り返し発生する日です。

```
---16
```

gMonth タイプ

データ・タイプ xs:gMonth は、毎年定期的に繰り返されるグレゴリオ暦の月を表します。グレゴリオ暦の月は、 *ISO 8601* で定義されています。データ・タイプ xdt:anyAtomicType から派生。

このデータ・タイプは、年の中の特定の月を表します。例えば、12 月にクリスマスのお祝いをすることを示すためにこのデータ・タイプを使用できます。

xs:gMonth の字句形式は *--mmzzzzzz* で、これは年または日のプロパティーを含まない、xs:date の省略表記です。先行する符号は使用できません。他のどのような形式も使用できません。以下の省略形はこの形式を説明しています。

mm
月を示す 2 桁の数表示。

zzzzzz
オプション。存在する場合、時間帯を示します。このプロパティーの形式について詳しくは、 32 ページの『時間帯標識』を参照してください。

例えば、以下の形式は 12 月を示します。これは毎年繰り返し発生する特定の月です。

```
--12
```

gMonthDay タイプ

データ・タイプ xs:gMonthDay は、定期的に繰り返されるグレゴリオ暦の日付を表します。グレゴリオ暦の日付は *ISO 8601* で定義されています。データ・タイプ xdt:anyAtomicType から派生。

このデータ・タイプは、年の中の特定の日を表します。例えば、毎年 4 月 16 日の誕生日を示すためにこのデータ・タイプを使用できます。

xs:gMonthDay の字句形式は *--mm-ddzzzzzz* で、これは年プロパティーを含まない、xs:date の省略表記です。先行する符号は使用できません。他のどのような形式も使用できません。以下の省略形はこの形式を説明しています。

mm
月を示す 2 桁の数表示。

dd 日を示す 2 桁の数表示。

zzzzzz
オプション。存在する場合、時間帯を示します。このプロパティーの形式について詳しくは、 32 ページの『時間帯標識』を参照してください。

例えば、以下の形式は 4 月 16 日を示します。これは毎年繰り返し発生する特定の日です。

```
--04-16
```

gYear タイプ

データ・タイプ xs:gYear は、グレゴリオ暦の年を表します。グレゴリオ暦の年は、*ISO 8601* で定義されています。データ・タイプ xdt:anyAtomicType から派生。

xs:gYear の字句形式は、*yyyyzzzzzz* です。この形式は、月、日、または時刻プロパティーを含まない、xs:dateTime の省略表記です。負の日付は使用できません。以下の省略形はこの形式を説明しています。

yyyy
年を示す 4 桁の数表示。有効な値は、0001 から 9999 です。正符号 (+) は許可されていません。

zzzzzz
オプション。存在する場合、時間帯を示します。このプロパティーの形式について詳しくは、 32 ページの『時間帯標識』を参照してください。

例えば、以下の形式はグレゴリオ暦 2005 年を表します。2005。

gYearMonth タイプ

データ・タイプ xs:gYearMonth は、特定のグレゴリオ暦の年の特定のグレゴリオ暦の月を表します。グレゴリオ暦の月は、 *ISO 8601* で定義されています。データ・タイプ xdt:anyAtomicType から派生。

xs:gYearMonth の字句形式は、*yyyy-mmzzzzzz* です。この形式は、時刻プロパティーを含まない、xs:dateTime の省略表記です。負の日付は使用できません。以下の省略形はこの形式を説明しています。

yyyy
年を示す 4 桁の数表示。有効な値は、0001 から 9999 です。正符号 (+) は許可されていません。

mm
月を示す 2 桁の数表示。

zzzzzz
オプション。存在する場合、時間帯を示します。このプロパティーの形式について詳しくは、 32 ページの『時間帯標識』を参照してください。

例えば、オプションの時間帯標識を含まない以下の形式は、2005 年の 10 月という月を示します。

```
2005-10
```

hexBinary タイプ

データ・タイプ xs:hexBinary は、16 進数にエンコードされたバイナリー・データを表します。データ・タイプ xdt:anyAtomicType から派生。

xs:hexBinary の字句形式は、各 2 進オクテットが 2 つの 16 進数で表された文字のシーケンスです。例えば、次の形式は、2 進表記が 111110110111 である 16 ビット整数 4023 の 16 進エンコードです。0FB7。

ID タイプ

データ・タイプ xs:ID は、*XML 1.0 (Third Edition)* の ID 属性のタイプを表します。データ・タイプ xs:NCName から派生。

xs:ID の字句形式は、コロンを含まない XML 名です (NCName)。

IDREF タイプ

データ・タイプ xs:IDREF は、*XML 1.0 (Third Edition)* の IDREF 属性のタイプを表します。データ・タイプ xs:NCName から派生。

xs:IDREF の字句形式は、コロンを含まない XML 名です (NCName)。

int タイプ

データ・タイプ xs:int は、2147483647 以下、-2147483648 以上の整数を表します。データ・タイプ xs:long から派生。

xs:int の字句形式は、オプションの符号があり、それに 10 進数の有限長シーケンスが続きます。符号を省略した場合は、正符号 (+) が想定されます。次の数値は、このデータ・タイプの有効な例です。`-1`、`0`、 `126789675`、`+100000`。

integer タイプ

データ・タイプ xs:integer は、9223372036854775807 以下、-9223372036854775808 以上の数値を表します。データ・タイプ xs:decimal から派生。

xs:integer の字句形式は、先行するオプションの符号付き 10 進数の有限長シーケンスです。符号を省略した場合は、正符号 (+) が想定されます。次の数値は、このデータ・タイプの有効な例です。`-1`、`0`、 `12678967543233`、`+100000`。

language タイプ

データ・タイプ xs:language は、*RFC 3066* によって定義されている自然言語 ID を表します。データ・タイプ xs:token から派生。

xs:language の字句形式は、ハイフンで接続されたタグのストリングで構成されます。各タグには、最大 8 文字含まれます。最初のタグには英字のみを含めることができ、後続のタグには英字および数字を含めることができます。例えば、値 en-US は、米国で使用される英語を表します。ストリングは、パターン `[a-zA-Z]{1,8}(-[a-zA-Z0-9]{1,8})*` に準拠します。

long タイプ

データ・タイプ xs:long は、 9223372036854775807 以下、-9223372036854775808 以上の整数を表します。データ・タイプ xs:integer から派生。

xs:long の字句形式は、オプションの符号と、それに続く 10 進数の有限長シーケンスです。符号を省略した場合は、正符号 (+) が想定されます。次の数値は、このデータ・タイプの有効な例です。`-1`、`0`、 `12678967543233`、`+100000`。

Name タイプ

データ・タイプ xs:Name は、XML 名を表します。データ・タイプ xs:token から派生。

xs:Name の字句形式は、 *XML 1.0 (Third Edition)* の Name のプロダクションに一致するストリングです。

NCName タイプ

データ・タイプ xs:NCName は、コロンのない XML 名を表します。データ・タイプ xs:Name から派生。

xs:NCName の字句形式は、コロンを含まない XML 名です。

negativeInteger タイプ

データ・タイプ xs:negativeInteger は、ゼロ未満の整数を表します。データ・タイプ xs:nonPositiveInteger から派生。

xs:negativeInteger の字句形式は、負符号 (-) と、それに続く 10 進数の有限長シーケンスです。このデータ・タイプで表すことのできる範囲は、-9223372036854775808 から -1 です。次の数値は、このデータ・タイプの有効な例です。`-1`、 `-12678967543233`、`-100000`。

NMTOKEN タイプ

データ・タイプ xs:NMTOKEN は、*XML 1.0 (Third Edition)* の NMTOKEN 属性のタイプを表します。データ・タイプ xs:token から派生。

xs:NMTOKEN の字句形式は、 *XML 1.0 (Third Edition)* の Nmtoken のプロダクションに一致するストリングです。

nonNegativeInteger タイプ

データ・タイプ xs:nonNegativeInteger は、ゼロ以上の整数を表します。データ・タイプ xs:integer から派生。

xs:nonNegativeInteger の字句形式は、オプションの符号と、それに続く 10 進数の有限長シーケンスです。符号を省略した場合は、正符号 (+) が想定されます。ゼロを示す字句形式については、符号は正符号 (+) または負符号 (-) のいずれも使用できます。その他のすべての字句形式において、符号がある場合は正符号 (+) にする必要があります。このデータ・タイプで表すことのできる範囲は、0 から +9223372036854775807 です。次の数値は、このデータ・タイプの有効な例です。`1`、`0`、 `12678967543233`、`+100000`。

nonPositiveInteger タイプ

データ・タイプ xs:nonPositiveInteger は、ゼロ以下の整数を表します。データ・タイプ xs:integer から派生。

xs:nonPositiveInteger の字句形式は、先行するオプションの符号と、それに続く 10 進数の有限長シーケンスです。ゼロを示す字句形式については、符号は負符号 (-) を付加するか省略できます。その他のすべての字句形式においては、負符号 (-) が必要です。このデータ・タイプによって表現できる範囲は、-9223372036854775808 から 0 です。次の数値は、このデータ・タイプの有効な例です。`-1`、`0`、`-12678967543233`、`-100000`。

normalizedString タイプ

データ・タイプ xs:normalizedString は、空白が正規化処理されたストリングを表します。データ・タイプ xs:string から派生。

xs:normalizedString の字句形式は、復帰 (X'0D') 文字、改行 (X'0A') 文字、またはタブ (X'09') 文字を含まないストリングです。

NOTATION タイプ

データ・タイプ xs:NOTATION は、 *XML 1.0 (Third Edition)* の NOTATION 属性タイプを表します。データ・タイプ xdt:anyAtomicType から派生。

xs:NOTATION タイプの字句形式は、 xs:QName タイプの字句形式です。

positiveInteger タイプ

データ・タイプ xs:positiveInteger は、1 以上の正の整数を表します。データ・タイプ xs:nonNegativeInteger から派生。

xs:positiveInteger の字句形式は、オプションの正符号 (+) と、それに続く 10 進数の有限長シーケンスです。このデータ・タイプで表すことのできる範囲は、 +1 から +9223372036854775807 です。次の数値は、このデータ・タイプの有効な例です。`1`、`12678967543233`、`+100000`。

QName タイプ

データ・タイプ xs:QName は、XML 修飾名 (QName) を表します。QName には、オプションのネーム・スペース接頭部、XML ネーム・スペースを識別する URI、およびローカル部分 (NCName) が含まれます。データ・タイプ xdt:anyAtomicType から派生。

データ・タイプ xs:QName の字句形式は、以下の形式のストリングです。 *`prefix:localName`*。以下の省略形はこの形式を説明しています。

prefix
: オプション。ネーム・スペース接頭部。ネーム・スペース接頭部は、ネーム・スペース宣言により、 URI 参照にバインドする必要があります。接頭部は、ネーム・スペース名のプレースホルダーとしてのみ機能します。接頭部が指定されていない場合、デフォルト・エレメント/タイプ・ネーム・スペースの URI が使用されます。

localName
: 修飾名のローカル部分である NCName。 NCName は、コロンのない XML 名です。

例えば、以下のストリングは、接頭部を含む QName の有効な字句形式です。

```
ns1:emp
```

short タイプ

データ・タイプ xs:short は、32767 以下、および -32768 以上の整数を表します。データ・タイプ xs:int から派生。

xs:short の字句形式は、オプションの符号と、それに続く 10 進数の有限長シーケンスです。符号を省略した場合は、正符号 (+) が想定されます。次の数値は、このデータ・タイプの有効な例です。`-1`、`0`、 `12678`、`+10000`。

string タイプ

データ・タイプ xs:string は、文字ストリングを表します。データ・タイプ xdt:anyAtomicType から派生。

xs:string の字句形式は、 XML に使用できる文字の範囲に含まれる任意の文字を含むことのできる文字のシーケンスです。

time タイプ

データ・タイプ xs:time は、毎日定期的に繰り返されるある一瞬の時刻を表します。データ・タイプ xdt:anyAtomicType から派生。

xs:time の字句形式は、*hh*:*mm*:*ss*.*ssssssszzzzzz* です。この形式は、年、日、または月プロパティーを含まない、xs:dateTime の省略表記です。以下の省略形はこの形式を説明しています。

hh 時間を示す 2 桁の数表示。24 の値は、表記される分と秒がゼロの場合にのみ許可されます。時刻 24:00:00 を含む照会は、次の日の 00:00:00 として処理されます。

: 時刻部の各部分間の区切り記号。

mm
分を示す 2 桁の数表示。

ss すべての秒を示す 2 桁の数表示。

.*ssssss*
オプション。存在する場合、秒を小数で示す 1 から 6 桁の数表示。

zzzzzz
オプション。存在する場合、時間帯を示します。このプロパティーの形式について詳しくは、 32 ページの『時間帯標識』を参照してください。

例えば、 (オプションの時間帯標識を含む) 以下の形式は、東部標準時午後 1 時 20 分 (協定世界時 (UTC) より 5 時間早い) を表します。

```
13:20:00-05:00
```

token タイプ

データ・タイプ xs:token は、トークン化されたストリングを表します。データ・タイプ xs:normalizedString から派生。

xs:token の字句形式は、以下の文字を含まないストリングです。

- 復帰 (X'0D')
- 改行 (X'0A')
- タブ (X'09')
- 先行または後続するスペース (X'20')
- 2 つ以上のスペースの内部シーケンス

unsignedByte タイプ

データ・タイプ xs:unsignedByte は、 255 以下の符号なし整数を表します。データ・タイプ xs:unsignedShort から派生。

xs:unsignedByte の字句形式は、10 進数の有限長シーケンスです。次の数値は、このデータ・タイプの有効な例です。0、126、および 100。

unsignedInt タイプ

データ・タイプ xs:unsignedInt は、4294967295 以下の符号なし整数を表します。データ・タイプ xs:unsignedLong から派生。

xs:unsignedInt の字句形式は、10 進数の有限長シーケンスです。次の数値は、このデータ・タイプの有効な例です。`0`、`1267896754`、`100000`。

unsignedLong タイプ

データ・タイプ xs:unsignedLong は、9223372036854775807 以下の符号なし整数を表します。データ・タイプ xs:nonNegativeInteger から派生。

xs:unsignedLong の字句形式は、10 進数の有限長シーケンスです。次の数値は、このデータ・タイプの有効な例です。`0`、`12678967543233`、`100000`。

unsignedShort タイプ

データ・タイプ xs:unsignedShort は、65535 以下の符号なし整数を表します。データ・タイプ xs:unsignedInt から派生。

xs:unsignedShort の字句形式は、10 進数の有限長シーケンスです。次の数値は、このデータ・タイプの有効な例です。`0`、`12678`、`10000`。

untyped タイプ

データ・タイプ xdt:untyped は、 XML スキーマによって妥当性検査されていないノードを表します。データ・タイプ xs:anyType から派生。

エレメント・ノードが xdt:untyped としてアノテーション付けされている場合、そのすべての子孫エレメント・ノードも xdt:untyped としてアノテーション付けされます。

untypedAtomic タイプ

データ・タイプ xdt:untypedAtomic は、 XML スキーマによって妥当性検査されていない原子値を表します。データ・タイプ xdt:anyAtomicType から派生。

データ・タイプ xdt:untypedAtomic の字句形式に制約はありません。

yearMonthDuration タイプ

データ・タイプ xdt:yearMonthDuration は、グレゴリオ暦の年および月コンポーネントで表される時間を表します。データ・タイプ xs:duration から派生。

このデータ・タイプで表すことができる範囲は、 -P83333333333333Y3M から P83333333333333Y3M です (または -999999999999999 カ月から 999999999999999 カ月)。

xdt:yearMonthDuration の字句形式は、 P*n*Y*n*M で、これは *ISO 8601* 形式の省略形式です。以下の省略形はこの形式を説明しています。

***n*Y** *n* は年数を示す符号なしの整数です。

***n*M**
n は月数を示す符号なしの整数です。

先行するオプションの負符号 (-) は、負の期間を示します。符号を省略すると、正の期間と認識されます。

例えば、以下の形式は、1 年 2 カ月の期間を示します。

```
P1Y2M
```

以下の形式は、負の 13 カ月の期間を示します。

```
-P13M
```

この形式の精度を低減し、表記を切り捨てることが可能ですが、以下の要件に準拠する必要があります。

- 指定子 P は常に必要です。
- 式の中の年または月の数が 0 の場合、その数と対応する指定子は省略できます。ただし、少なくとも 1 つの数およびその指定子 (Y または M) が必要です。

例えば、以下の形式は使用可能です。

```
P1347Y
P1347M
```

形式 P-1347M は使用できませんが、形式 -P1347M は使用できます。形式 P24YM および PY43M は使用できません。これは、Y には少なくとも 1 つの先行する数字が必要で、M には 1 つの先行する数字が必要であるためです。

DB2 は、xdt:yearMonthDuration 値を正規化された形式で保管します。正規化された形式では、月コンポーネントは 12 未満です。各 12 カ月が 1 年に変換されます。例えば、以下に示す XQuery 式は yearMonthDuration として 20 年と 30 カ月を指定するコンストラクター関数を呼び出します。

```
xquery
xdt:yearMonthDuration("P20Y30M")
```

この期間では、30 カ月は 2 年と 6 カ月に変換されます。この式は、正規化された yearMonthDuration 値 P22Y6M を戻します。

第 3 章 プロローグ

プロローグ は、照会の処理環境を定義する一連の宣言です。プロローグ内の各宣言の後にはセミコロン (;) が付きます。プロローグは照会のオプション部分です。プロローグを使用せずに、照会本体で有効な照会を構成できます。

プロローグには、オプションのバージョン宣言、ネーム・スペース宣言、および照会の処理に影響を与えるプロパティーの値を設定するオプションの宣言であるセッターが含まれます。

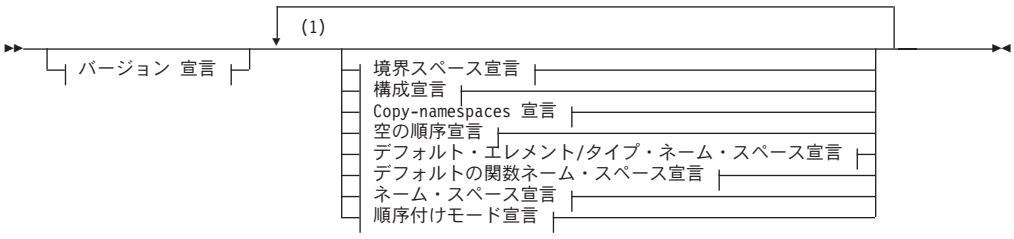
DB2 XQuery では、照会の処理方法を変更するために使用できる、境界スペース宣言をサポートしています。他にプロローグを構成するものとして、ネーム・スペース宣言およびデフォルト・ネーム・スペース宣言があります。

DB2 XQuery では、以下のセッターもサポートします。ただし、 DB2 XQuery でサポートされるのは、いずれの場合も 1 つのオプションのみであるため、これらによって処理環境が変更されることはありません。

- 構成宣言
- Copy-namespaces 宣言
- 空の順序宣言
- 順序付けモード宣言

バージョン宣言を含める場合は、プロローグの最初に指定する必要があります。セッターおよびその他の宣言は、バージョン宣言の後であればプロローグに任意の順序で指定できます。

構文



注:

1 ネーム・スペース宣言を除き、各宣言は 1 回のみ指定できます。

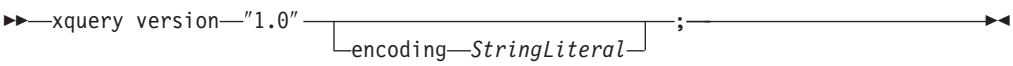
バージョン宣言

バージョン宣言は、照会の処理に必要な XQuery の構文およびセマンティクスのバージョンを示すために照会の最初に行われます。バージョン宣言にはエンコード宣言を含めることができますが、DB2 XQuery はエンコード宣言を無視します。

© IBM Corporation 2006, 2008

バージョン宣言を行う場合は、プロローグの先頭で行う必要があります。DB2 XQuery でサポートされるバージョンは「1.0」のみです。

構文



1.0 照会を処理するために、XQuery の構文およびセマンティクスのバージョン 1.0 が必要であることを指定します。

StringLiteral
エンコード名を表すストリング・リテラルを指定します。 *StringLiteral* の値は無視されるため、エンコード宣言を指定しても照会に影響はありません。DB2 XQuery では常にエンコードが UTF-8 であると想定されます。

例

以下のバージョン宣言は、照会を、XQuery バージョン 1.0 をサポートするインプリメンテーションで処理する必要があることを示しています。

```
xquery version "1.0";
```

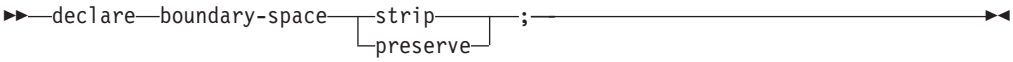
境界スペース宣言

照会プロローグ内の境界スペース宣言は、照会の境界スペース・ポリシーを設定します。*境界スペース・ポリシー* は、エレメント・コンストラクターによる境界空白の処理方法を制御します。*境界空白* には、エレメント・コンストラクター内のタグや括弧で囲んだ式の間の境界に単独で存在するすべての空白文字が含まれます。

境界スペース・ポリシーにより、エレメントの構成時に境界空白が保持されるか、または削除されるかを指定することができます。境界スペース宣言が指定されていない場合、デフォルトの動作として、エレメント構成時に境界空白が削除されます。

プロローグには、 1 つの照会について 1 つの境界スペース宣言のみを含めることができます。

構文



strip
エレメントの構成時に境界空白が削除されることを指定します。

preserve
エレメントの構成時に境界空白が保持されることを指定します。

例

以下の境界スペース宣言は、エレメントの構成時に境界空白が保持されることを指定しています。

```
declare boundary-space preserve;
```

構成宣言

照会プロローグの構成宣言は、照会の構成モードを設定します。構成モードは、新しく構成されたノードのコンテンツを形成するためにコピーされるエレメント・ノードおよび属性ノードへの、タイプのアノテーションの割り当て方法を制御します。

DB2 XQuery では、構成されたエレメント・ノードの構成モードは、常に **preserve** です。構成モードが **preserve** の場合、構成されたエレメントのコピー済み属性および子孫は、元のタイプを保持します。

preserve 以外の値を指定する構成宣言は、エラーになります。プロローグに含めることができるのは、1 つの照会について 1 つの構成宣言のみです。

構文

preserve
構成されたエレメントのコピー済み属性および子孫が元のタイプを保持することを指定します。

例

以下の構成宣言は有効ですが、エレメント構造のデフォルトの動作は変更しません。

```
declare construction preserve;
```

Copy-namespaces 宣言

Copy-namespaces モードでは、既存のエレメント・ノードがエレメント・コンストラクターによってコピーされるときに割り当てられるネーム・スペースのバインディングを制御します。

DB2 XQuery では、copy-namespaces モードは、常に **preserve** および **inherit** です。 **preserve** を設定すると、元のエレメントの範囲内にあるすべてのネーム・スペースが新規コピーに保持されるよう指定されます。デフォルトのネーム・スペースは、他のすべてのネーム・スペースのバインディングと同様に取り扱われます。つまり、コピーしたノードはそのデフォルトのネーム・スペースまたはデフォルトのネーム・スペースが欠落した状態を保持します。**inherit** を設定すると、コピーしたノードが、構成されたノードの範囲内にあるネーム・スペースを継承するよう指定されます。競合が発生した場合は、元のノードから保持されたネーム・スペースのバインディングが優先されます。

preserve および **inherit** 以外の値を指定する Copy-namespaces 宣言はエラーになります。プロローグに含めることができるのは、1 つの照会について 1 つの Copy-namespaces 宣言のみです。

構文

preserve
元のエレメントの範囲内にあるすべてのネーム・スペースを新規コピーに保持するよう指定します。

inherit
コピーしたノードが、構成されたノードの範囲内にあるネーム・スペースを継承するよう指定します。

例

以下の copy-namespaces 宣言は有効ですが、エレメント構造のデフォルトの動作は変更しません。

```
declare copy-namespaces preserve, inherit;
```

デフォルトのエレメント/タイプのネーム・スペース宣言

照会プロローグのデフォルトのエレメント/タイプのネーム・スペース宣言は、エレメント名およびタイプ名の接頭部なしの QNames (修飾名) に使用するネーム・スペースを指定します。

照会プロローグには、1 つのデフォルト・エレメント/タイプ・ネーム・スペース宣言のみを含めることができます。この宣言は、宣言が直接エレメント・コンストラクターのネーム・スペース宣言属性でオーバーライドされない限り、宣言される照会全体の有効範囲に属します。デフォルトのエレメント/タイプのネーム・スペースが宣言されない場合、接頭部なしのエレメント名およびタイプ名は、ネーム・スペースには属しません。

デフォルトのエレメント/タイプのネーム・スペースは、非修飾属性名には適用されません。接頭部なしの属性名および変数名はネーム・スペースには属しません。

構文

element
宣言がデフォルトのエレメント/タイプのネーム・スペース宣言であることを指定します。

URILiteral
ネーム・スペースの URI を示すストリング・リテラルを指定します。ストリング・リテラルは、有効な URI またはゼロ長ストリングでなければなりません。デフォルトのエレメント/タイプのネーム・スペース宣言のストリング・リテラルがゼロ長のストリングである場合、接頭部なしのエレメント名およびタイプ名はネーム・スペースに属しません。

例

以下の宣言では、エレメント名およびタイプ名のデフォルトのネーム・スペースが URI http://posample.org と関連付けられているネーム・スペースであることを指定します。

```
declare default element namespace "http://posample.org";
<name>Snow boots</name>
```

この例の照会が実行されると、ネーム・スペース URI http://posample.org に関連付けられたネーム・スペースに、新たに作成されたノード (name というエレメント・ノード) が追加されます。

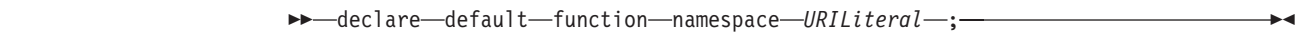
デフォルトの関数ネーム・スペース宣言

照会プロローグのデフォルトの関数ネーム・スペース宣言では、関数呼び出しの接頭部が付かない関数に使用されるネーム・スペース URI を指定します。

照会プロローグには、1 つのデフォルトの関数ネーム・スペース宣言のみを含めることができます。デフォルトの関数ネーム・スペースが宣言されない場合、デフォルトの関数ネーム・スペースは、XPath 関数および XQuery 関数のネーム・スペースです (http://www.w3.org/2005/xpath-functions)。デフォルトの関数ネーム・スペースを宣言する場合、接頭部を指定せずにデフォルトの関数ネーム・スペースで任意の関数を呼び出すことができます。

DB2 XQuery では、接頭部なし関数呼び出しのローカル名がデフォルトの関数ネーム・スペースの関数と一致しない場合、エラーが戻ります。

構文



function
宣言は、デフォルトの関数ネーム・スペース宣言であることを指定します。

URILiteral
ネーム・スペースの URI を示すストリング・リテラルを指定します。ストリング・リテラルは、有効な URI またはゼロ長ストリングでなければなりません。すべての関数が何らかのネーム・スペースに属するため、デフォルトの関数ネーム・スペース宣言のストリング・リテラルがゼロ長ストリングの場合、すべての関数の呼び出しで、接頭部付きの関数名を使用する必要があります。

例

以下の宣言では、デフォルトの関数ネーム・スペースが URI http://www.ibm.com/xmlns/prod/db2/functions と関連付けられていることを指定します。

```
declare default function namespace "http://www.ibm.com/xmlns/prod/db2/functions";
```

この例の照会の本文では、関数名に接頭部を含めずにデフォルトの関数ネーム・スペースの任意の関数を参照できます。このデフォルトの関数ネーム・スペースには、関数 xmlcolumn が含まれているため、db2-fn:xmlcolumn('T1.MYDOC') と入力する代わりに、xmlcolumn('T1.MYDOC') と入力できます。ただし、この例のデフォ

ルトの関数ネーム・スペースは、XQuery 関数のネーム・スペースと関連付けられていないため、XQuery の組み込み関数を呼び出すときは、接頭部を指定する必要が生じます。例えば、`current-date()` と入力するのではなく、`fn:current-date()` と入力する必要があります。

空の順序宣言

照会プロローグの空の順序宣言は、FLWOR 式の **order by** 節が処理されるときに、空のシーケンスまたは NaN 値が最大値または最小値のいずれとして解釈されるのかを制御します。

DB2 XQuery では、空のシーケンスは、FLWOR 式の **order by** 節の処理時に常に最大値として解釈されます。NaN 値は、空のシーケンス以外のすべての値よりも大きいと解釈されます。この設定を無効にすることはできません。**empty greatest** 以外の値を指定する空の順序宣言は、エラーになります。照会プロローグに含めることができるのは、1 つの照会について 1 つの空の順序宣言のみです。

構文

```
►►—declare—default—order—empty——greatest——;—————————————————————►◄
```

greatest
FLWOR 式の **order by** 節の処理時に、空のシーケンスが常に最大値として解釈されることを指定します。NaN 値は、空のシーケンス以外のすべての値よりも大きいと解釈されます。

例

以下は、有効な空の順序宣言です。

```
declare default order empty greatest;
```

順序付けモード宣言

照会プロローグ内の*順序付けモード宣言* により、照会の順序付けモードが設定されます。*順序付けモード*は、照会結果におけるノードの順序付けを定義します。

DB2 XQuery では、*XQuery 1.0: An XML Query Language* で定義されているような順序付けモードがサポートされていないため、順序付けモード宣言がある場合は、unordered を指定する必要があります。DB2 XQuery における照会結果の順序を規定する規則については、56 ページの『XQuery 式の結果の順序』を参照してください。

照会プロローグには、1 つの順序付けモード宣言のみを含めることができます。unordered 以外の値を指定した順序付けモード宣言の結果はエラーとなります。

構文

```
►►—declare—ordering—unordered—;—————————————————————————————►◄
```

unordered

XQuery 1.0: An XML Query Language の順序付けモードの規則が無効であることを指定します。DB2 XQuery における照会結果の順序を規定する規則については、 56 ページの『XQuery 式の結果の順序』を参照してください。

例

以下の宣言は有効ですが、 DB2 XQuery では非順序モードのみがサポートされるため、順序付けのデフォルトの動作は変更されません。

```
declare ordering unordered;
```

ネーム・スペース宣言

照会プロローグにおけるネーム・スペース宣言では、ネーム・スペース接頭部を宣言し、その接頭部をネーム・スペース URI と関連付けます。接頭部とネーム・スペース URI との間の関連付けを、ネーム・スペース・バインディング と呼びます。ネーム・スペース宣言でバインドされるネーム・スペースは、静的に既知のネーム・スペースに追加されます。*静的に既知*のネーム・スペース は、照会の処理時にネーム・スペース接頭部を解決するために使用できるすべてのネーム・スペース・バインディングで構成されます。

ネーム・スペース宣言は、宣言が直接エレメント・コンストラクターのネーム・スペース宣言属性で指定変更されない限り、それが宣言される照会全体を通して有効範囲内です。照会プロローグにおいて同じネーム・スペース接頭部を複数回宣言すると、結果はエラーになります。

構文

prefix

URILiteral によって指定する URI にバインドするネーム・スペース接頭部を指定します。ネーム・スペース接頭部は、エレメント、属性、データ・タイプ、または関数のネーム・スペースを識別するために、修飾名 (QName) で使用されます。

接頭部 `xmlns` および `xml` は予約済みであり、ネーム・スペース宣言で接頭部として指定することはできません。

URILiteral

接頭部をバインドする URI を指定します。*URILiteral* は、有効な URI を含む、長さがゼロでないリテラル・ストリングにする必要があります。

例

以下の照会には、ネーム・スペース接頭部 `ns1` を宣言し、これをネーム・スペース URI `http://posample.org` に関連付けるネーム・スペース宣言が含まれています。

```
declare namespace ns1 = "http://posample.org";
<ns1:name>Thermal gloves</ns1:name>
```

この例の照会が実行されると、ネーム・スペース URI http://posample.org に関連付けられたネーム・スペースに、新たに作成されたノード (name というエレメント・ノード) が追加されます。

事前宣言されたネーム・スペース接頭部

XQuery には、各照会の処理前に静的に既知のネーム・スペースに存在する、いくつかの事前宣言されたネーム・スペース接頭部があります。事前宣言された接頭部はすべて明示宣言することなく使用できます。DB2 XQuery の事前宣言されたネーム・スペース接頭部には、以下の表に示すような接頭部と URI のペアが含まれます。

表 11. DB2 XQuery の事前宣言されたネーム・スペース

接頭部	URI	説明
xml	http://www.w3.org/XML/1998/namespace	XML 予約済みネーム・スペース
xs	http://www.w3.org/2001/XMLSchema	XML スキーマ・ネーム・スペース
xsi	http://www.w3.org/2001/XMLSchema-instance	XML スキーマ・インスタンス・ネーム・スペース
fn	http://www.w3.org/2005/xpath-functions	デフォルトの関数ネーム・スペース
xdt	http://www.w3.org/2005/xpath-datatypes	XQuery のタイプのネーム・スペース
db2-fn	http://www.ibm.com/xmlns/prod/db2/functions	DB2 関数ネーム・スペース

照会プロローグでネーム・スペース宣言を指定することにより、事前宣言されたネーム・スペース接頭部を指定変更できます。ただし、接頭部 xml に関連付けられた URI は指定変更できません。

第 4 章 式

式は、照会の基本的なビルディング・ブロックです。式は、単独で使用することも、他の式と組み合わせて複合照会を形成することもできます。DB2 XQuery では、XML データを使用するために複数の種類の式をサポートします。

式の評価および処理

いくつかの操作は、式の処理によく含められます。これらの操作には、ノードからの原子値の抽出、期待されるタイプの値を取得するためのタイプ・プロモーションおよびサブタイプ置換の使用、およびシーケンスのブール値の計算が含まれます。

DB2 XQuery では、更新式は変換式の **modify** 節内でのみ使用できます。XQuery 変換式、および更新式の処理については、123 ページの『変換式』および 120 ページの『変換式での更新式の使用』を参照してください。

動的コンテキストおよびフォーカス

式の動的コンテキストは、式を評価する際に使用可能な情報です。コンテキスト・アイテム、コンテキストの位置、およびコンテキスト・サイズで構成されるフォーカスは、動的コンテキストの重要な部分です。

フォーカスは、DB2 XQuery によるシーケンス内の各アイテムの処理時に変更されます。フォーカスは、以下の情報で構成されます。

コンテキスト・アイテム
現在処理中の原子値またはノード。コンテキスト・アイテムは、単一のドット (.) で構成されるコンテキスト・アイテム式によって検索できます。

コンテキストの位置
現在処理中のシーケンス内のコンテキスト・アイテムの位置。コンテキスト・アイテムは、fn:position() 関数によって検索できます。

コンテキスト・サイズ
現在処理中のシーケンス内のアイテム数。コンテキスト・サイズは、fn:last() 関数によって検索できます。

優先順位

XQuery 文法では、演算子および式の間の組み込み優先順位が定義されています。低い優先順位を持つ式をより高い優先順位を持つ式のオペランドとして使用する場合、低い優先順位を持つ式を括弧で囲む必要があります。

以下の表は、XQuery の演算子および式を、その優先順位が低いものから高いものへ順にリストしています。結合順序の列は、同じ優先順位の演算子または式が適用される順序を示しています。

© IBM Corporation 2006, 2008

表 12. DB2 XQuery における優先順位

演算子または式	結合順序
, (コンマ)	左から右
:= (代入)	右から左
FLWOR、some、every、if	左から右
or	左から右
and	左から右
eq、ne、lt、le、gt、ge、=、!=、<、<=、>、>=、is、<<、>>	左から右
to	左から右
+, -	左から右
*、div、idiv、mod	左から右
union、|	左から右
intersect、except	左から右
castable	左から右
cast	左から右
- (単項)、+ (単項)	右から左
?	左から右
/, //	左から右
[ ], ( ), {}	左から右

XQuery 式の結果の順序

DB2 XQuery を使用する場合、シーケンスを決定論的順序で戻す種類の XQuery 式と、非決定論的順序で戻す種類の XQuery 式があります。

以下の種類の式は、シーケンスを決定論的順序で戻します。

- 明示的な **order by** 節を含む FLWOR 式は、結果を指定された順序で戻します。例えば、以下の式は、 product エレメントのシーケンスを price の昇順で戻します。

```
for $p in /product
order by $p/price
return $p
```

- エレメントを追加し明示的な位置キーワードを使用する更新式は、指定した位置にエレメントが追加された結果を戻します。例えば、以下の更新式は、**as first into** キーワードを使用し、エレメント `<status>current</status>` を customerinfo エレメント内の最初の項目エレメントとして挿入します。

```
xquery declare default element namespace 'http://posample.org';
transform
copy $mycust := db2-fn:sqlquery('select info from customer where cid = 1001')
modify
  do insert <status>current</status> as first into $mycust/customerinfo
return $mycust
```

- **union** 演算子、**intersect** 演算子、または **except** 演算子を使用してシーケンスを結合する式は、結果を文書の順序で戻します。
- 以下の条件を満たすパス式は、結果を文書の順序で戻します。
  - フォワード軸ステップのみを含むパス式。

– (関数呼び出しまたは変数参照の結果のように、) パス式の起点が単一ノード内にある。
– パス式に、複数の述部を含むステップがない。
– パス式に、fn:position 関数呼び出しまたは fn:last 関数呼び出しが含まれない。

以下の例は、結果を文書の順序で戻すパス式です (変数 *$bib* が単一エレメントにバインドされていると想定します)。

```
$bib/book[title eq "War and Peace"]/chapter
```

- **to** 演算子を含む式である範囲式は、整数のシーケンスを昇順で戻します。例: `15 to 25`。
- すべてのオペランドが決定論的順序のシーケンスである場合のコンマ演算子を含む式は、そのオペランドの順序で結果を戻します。例えば、以下の式はシーケンス (5, 10, 15, 16, 17, 18, 19, 20, 25) を戻します。

  ```
  (5, 10, 15 to 20, 25)
  ```

- 結果を決定論的順序で戻すオペランド式を含むその他の式は、結果を決定論的順序で戻します。例えば、変数 *$pub* が単一エレメントにバインドされていると想定すると、以下の条件式は順序付けされた結果を戻します。これは、then 節および else 節内のパス式が順序付けされた結果を戻すためです。

  ```
  if ($pub/type eq "journal")
    then $pub/editor
    else $pub/author
  ```

前のリストにリストされていない式が複数の項目を戻す場合、シーケンス内の項目の順序は非決定論的になります。

表 13. XQuery 式の結果の順序付けのまとめ

式の種類	決定論的順序付けの条件	結果の順序付け	例
FLWOR	明示的な **order by** 節	**order by** 節によって決定される	以下の式は、product エレメントのシーケンスを price の昇順で戻します。 `for $p in /product order by $p/price return $p`
更新式	エレメントを追加するときの位置を指定するキーワードの使用	更新式のキーワードによって決定される	キーワード **as last into** を使用してエレメントを挿入すると、そのエレメントは指定したノードの最後の子として追加されます。
union 演算子、**intersect** 演算子、または **except** 演算子を使用した式	なし	文書順序	`$managers union $students`

表 13. XQuery 式の結果の順序付けのまとめ (続き)

式の種類	決定論的順序付けの条件	結果の順序付け	例
パス式	• フォワード軸ステップのみを含むパス式。 • (関数呼び出しまたは変数参照の結果のように、) パス式の起点が単一ノード内にある。 • パス式に、複数の述部を含むステップがない。 • パス式に、fn:position 関数呼び出しまたは fn:last 関数呼び出しが含まれない。	文書順序	以下の例は、結果を文書の順序で戻すパス式です (変数 *$bib* が単一エレメントにバインドされていると想定します)。 `$bib/book [title eq "War and Peace"] /chapter`
to 演算子を含む式である範囲式	なし	昇順の整数のシーケンス	`15 to 25`
コンマ演算子を含む式	すべてのオペランドが決定論的順序のシーケンスである	結果はオペランドの順序で戻される	`(5, 10, 15 to 20, 25)`
その他の式	結果をすべて決定論的順序で戻すオペランド式	ネストされた式の結果の順序付けによって決定される	変数 *$pub* が単一エレメントにバインドされていると想定すると、以下の条件式は順序付けされた結果を戻します。これは、then 節および else 節内のパス式が順序付けされた結果を戻すためです。 `if ($pub/type eq "journal") then $pub/editor else $pub/author`

注: 決定論的順序を持たないシーケンスに定位置述部が適用される場合、結果は非決定論的になります。これは、シーケンス内の任意の項目を選択できることを意味します。

原子化

原子化 とは、項目のシーケンスを原子値のシーケンスに変換するプロセスです。原子化は、原子値のシーケンスが必要な式で使用されます。

シーケンス内の各項目が、以下のルールを適用して原子値に変換されます。

- 項目が原子値の場合、原子値が戻されます。
- 項目がノードの場合、その型付き値が戻されます。*ノードの型付き値* は、ノードから抽出できるゼロ個以上の原子値のシーケンスです。ノードが型付き値を持たない場合、エラーが戻されます。

シーケンスの暗黙的な原子化により、シーケンスに対する明示的な fn:data 関数の呼び出しと同じ結果が作成されます。

例えば、以下のシーケンスには、ノードと原子値の組み合わせが含まれます。

```
("Some text",<anElement xsi:type="string">More text</anElement>,
<anotherElement xsi:type="decimal">1.23</anotherElement>,1001)
```

このシーケンスに原子化を適用すると、以下の原子値のシーケンスになります。

```
("Some text", "More text", 1.23, 1001)
```

以下の XQuery は、原子化を使用して項目を原子値に変換します。

- 算術式
- 比較式
- 期待されるタイプが原子タイプである引数を使用した関数の呼び出し
- キャスト式
- 多様なノードのコンストラクター式
- FLWOR 式の **order by** 節
- タイプ・コンストラクター関数

サブタイプ置換

サブタイプ置換 は、期待されるタイプから派生した動的タイプを持つ値を使用するための方法です。

サブタイプ置換により、値の実際のタイプは変更されません。例えば、xs:decimal 値が期待される場所で xs:integer 値が使用された場合、値はそのタイプを xs:integer として保持します。

以下の例では、 fn:compare 関数が、xs:string 値を xs:NCName 値と比較します。

```
fn:compare("product", xs:NCName("product"))
```

戻り値は 0 で、これは引数が等しいとされることを意味します。 fn:compare 関数は、xs:string タイプの引数を期待しますが、 xs:NCNAME タイプが xs:string から派生しているため、xs:NCNAME タイプの値を使用してこの関数を呼び出すことができます。

サブタイプ置換は、式に、期待されるタイプから派生した値が渡される場合に必ず使用されます。

タイプのプロモーション

タイプのプロモーション は、原子値を、そのオリジナルのタイプから式によって期待されるタイプに変換する処理です。 XQuery は、関数呼び出し、**order by** 節、および数値またはストリングのオペランドを受け入れる演算子の評価時に、タイプのプロモーションを使用します。

XQuery では、以下のタイプのプロモーションが可能です。

数値タイプのプロモーション:

xs:float タイプ (または xs:float から制限によって派生した任意のタイプ) の値は、 xs:double タイプにプロモートできます。結果は、オリジナルの値と同じ xs:double 値です。

xs:decimal タイプ (または xs:decimal から制限によって派生した任意のタイプ) の値は、 xs:float または xs:double のいずれかのタイプにプロモートできます。このプロモーションの結果は、オリジナルの値を必要なタイプにキャストすることによって作成されます。この種類のプロモーションは、精度の喪失を発生させる可能性があります。
以下の例では、xs:double 値 13.54e-2 および xs:decimal 値 100 を含むシーケンスが、xs:double タイプの値を戻す fn:sum 関数に渡されます。

```
fn:sum(xs:double(13.54e-2), xs:decimal(100))
```

URI タイプのプロモーション:

xs:anyURI タイプ (または xs:anyURI から制限によって派生した任意のタイプ) の値は、 xs:string タイプにプロモートできます。このプロモーションの結果は、オリジナルの値を xs:string タイプにキャストすることによって作成されます。

以下の例では、URI 値が期待される xs:string タイプにプロモートされ、関数が 18 を戻します。

```
fn:string-length(xs:anyURI("http://example.com"))
```

タイプのプロモーションとサブタイプ置換は以下の点で異なることに注意してください。

- タイプのプロモーションの場合、原子値は、そのオリジナルのタイプから式によって期待されるタイプに実際に変換されます。
- サブタイプ置換の場合、特定のタイプを期待する式を、そのタイプから派生した値を使用して呼び出すことができます。ただし、値はそのオリジナルのタイプを保持します。

有効なブール値

シーケンスの*有効なブール値 (EBV)* は、ブール値が必要な式の処理時に暗黙的に計算されます。値の EBV は、値に fn:boolean 関数を適用することで決定されます。

以下の表は、値の特定のタイプに対して戻される EBV を説明しています。

表 14. XQuery において値の特定のタイプに戻される EBV

値の説明	戻される EBV
空のシーケンス	false
1 番目の項目がノードのシーケンス	true
xs:boolean タイプの単一値 (または xs:boolean から派生したタイプの単一値)	false - xs:boolean 値が false の場合
	true - xs:boolean 値が true の場合
xs:string タイプまたは xdt:untypedAtomic タイプの単一値 (またはこの 2 つのタイプのいずれかから派生したタイプの単一値)	false - 値の長さがゼロの場合
	true - 値がゼロより大きい長さの場合

表 14. XQuery において値の特定のタイプに戻される EBV (続き)

値の説明	戻される EBV
任意の数値の単一値 (または数値タイプから派生した単一値)	false - 値が NaN または数的にゼロと等しい場合
	true - 値が数的にゼロと等しくない場合
その他すべての値	エラー

注: 少なくとも 1 個のノードおよび原子値を含むシーケンスの有効なブール値は、順序が予測不能な照会では非決定論的になります。

シーケンスの有効なブール値は、以下の式のタイプが処理されるときに暗黙的に計算されます。

- 論理式 (**and**、**or**)
- fn:not 関数
- FLWOR 式の **where** 節
- `a[b]` など、述部の特定のタイプ
- 条件式 (**if**)
- 量化式 (**some**、**every**)

基本式

基本式は、言語の基本プリミティブです。これには、リテラル、変数参照、括弧で囲んだ式、コンテキスト・アイテム式、コンストラクター、および関数呼び出しが含まれます。

リテラル

リテラル は、原子値の直接構文表記です。DB2 XQuery は、数値リテラルおよびストリング・リテラルの 2 種類のリテラルをサポートします。

数値リテラル は、xs:integer タイプ、xs:decimal タイプ、または xs:double タイプの原子値です。

- 小数点 (.) および e または E の文字を含まない数値リテラルは、 xs:integer タイプの原子値です。例えば、`12` は数値リテラルです。
- 小数点 (.) を含み、e または E の文字を含まない数値リテラルは、 xs:decimal タイプの原子値です。例えば、`12.5` は数値リテラルです。
- e または E の文字を含む数値リテラルは、xs:double タイプの原子値です。例えば、`125E2` は数値リテラルです。

数値リテラルの値は、XML スキーマの規則に従って解釈されます。

ストリング・リテラル は、区切り文字の単一引用符 (') または二重引用符 (") に囲まれた、xs:string タイプの原子値です。ストリング・リテラルには、事前定義されたエンティティー参照および文字参照を含めることができます。例えば、以下のストリングは有効なストリング・リテラルです。

```
"12.5"
"He said, ""Let it be."""
'She said: "Why should I?"'
"Ben &amp;amp; Jerry&amp;apos;s"
"&amp;#8364;65.50" (: denotes the string €65.50 :)
```

ヒント: 単一引用符で区切られたストリング・リテラル内に単一引用符を含めるには、2 つの連続する単一引用符を指定します。同様に、二重引用符で区切られたストリング・リテラル内に二重引用符を含めるには、2 つの連続する二重引用符を指定します。

ストリング・リテラル内では、行の終わりは *XML 1.0 (Third Edition)* の規則に従って正規化されます。復帰 (X'0D') およびそれに続く改行 (X'0A') を含む 2 文字シーケンスは、単一の改行 (X'0A') に変換されます。後続の改行 (X'0A') がない復帰 (X'0D') は、単一の改行 (X'0A') に変換されます。

インスタンス化する値にリテラル表記がない場合は、コンストラクター関数または組み込み関数を使用して値を戻すことができます。以下の関数およびコンストラクターは、リテラル表記を持たない値を戻します。

- 組み込み関数 fn:true() および fn:false() は、それぞれブール値 `true` および `false` を戻します。これらの値は、コンストラクター関数 xs:boolean(″false″) および xs:boolean(″true″) によっても戻すことができます。
- コンストラクター関数 `xs:date("2005-04-16")` は、そのタイプが xs:date で、その値が日付 2005 年 4 月 16 日を表す項目を戻します。
- コンストラクター関数 `xdt:dayTimeDuration("PT4H")` は、そのタイプが xdt:dayTimeDuration で、その値が 4 時間という期間を表す項目を戻します。
- コンストラクター関数 `xs:float("NaN")` は、特殊な浮動小数点値「非数 (Not a Number)」を戻します。
- コンストラクター関数 `xs:double("INF")` は、特殊な倍精度値「正の無限大」を戻します。

定義済みエンティティー参照

*定義済みエンティティー参照*は、 DB2 XQuery において特定の構文的重要性を持つ文字を表す、文字の短いシーケンスです。定義済みエンティティー参照は、アンパーサンド (&) で開始され、セミコロン (;) で終了します。ストリング・リテラルの処理時に、各定義済みエンティティー参照が、それが表す文字に置き換えられます。

以下の表に、DB2 XQuery が認識する定義済みエンティティー参照をリストします。

表 15. DB2 XQuery における定義済みエンティティー参照

エンティティー参照	表される文字
&lt;	<
&gt;	>
&amp;	&
&quot;	″
&apos;	'

文字参照

文字参照 は、10 進数コード・ポイントまたは 16 進数コード・ポイントで識別される Unicode 文字の XML 形式のリファレンスです。

文字参照は、`&#x` または `&#` のいずれかで開始し、セミコロン (;) で終了します。文字参照が `&#x` で開始している場合、終了のセミコロン (;) の前の数字と文字は、*ISO/IEC 10646* 規格の文字のコード・ポイントの 16 進表記を示します。文字参照が `&#` で開始している場合、終了のセミコロン (;) の前の数字は、文字のコード・ポイントの 10 進表記を示します。

例

文字参照 `&#8364;` または `&#x20AC;` は、ユーロ記号 (€) を示します。

変数参照

変数参照は、ドル記号 ($) が先頭に付加された NCName です。照会の評価時に、各変数参照は、変数にバインドされている値に解決されます。すべての変数参照は、参照の時点で、有効範囲内変数に含まれる名前と一致している必要があります。

変数は、以下のようにして有効範囲内変数に追加されます。

- 変数は、ホスト言語環境、SQL/XML、XMLQUERY 関数、 XMLTABLE 関数、または XMLEXISTS 述部によって有効範囲内変数に追加できます。SQL/XML によって追加される変数は、 XQuery 式において同じ変数の別のバインディングによって変数が指定変更されない限り、照会全体について有効範囲内です。
- XQuery 式によって変数を値にバインドできます。変数をバインドできる式は、FLWOR 式および量化式です。関数呼び出しも、関数本体を実行する前に、関数の仮パラメーターに値をバインドします。XQuery 式によってバインドされる変数は、それがバインドされた式全体を通して有効範囲内です。

  FLWOR 式では、変数名を複数回宣言できません。例えば、DB2 XQuery では以下の式はサポートされません。

```
for $i in (1, 2)
for $i in ("a", "b")
return $i
```

変数参照が有効範囲内の 2 つ以上の変数バインディングに一致した場合、参照は内部バインディング (有効範囲がより小さいバインディング) を参照します。

ヒント: コードを読みやすいものにするには、照会内の変数に固有の名前を使用します。

例

以下の例では、FLWOR 式が、変数 `$seq` をシーケンス (10, 20, 30) にバインドします。

```
let $seq := (10, 20, 30)
return $seq[2];
```

戻り値は 20 です。

括弧で囲んだ式

複数の演算子を含む式において評価を特定の順序で実行するために、括弧を使用できます。

例えば、式 `(2 + 4) * 5` は、30 に評価されます。これは、括弧で囲んだ式 `(2 + 4)` が先に評価され、その結果が 5 で乗算されるためです。括弧がない場合、式 `2 + 4 * 5` は、22 に評価されます。これは、乗算演算子の方が加算演算子より優先順位が高いためです。

空の括弧は、空のシーケンスを表します。

コンテキスト・アイテム式

コンテキスト・アイテム式は、1 個のピリオド文字 (.) から構成されます。コンテキスト・アイテム式は、現在処理中の項目 (コンテキスト・アイテム) に評価されます。コンテキスト・アイテムは、ノードまたは原子値のどちらでも構いません。コンテキスト・アイテムは、パス式および述部式でのみ定義されます。

例

以下の例は、範囲式 `1 to 100` で戻されるシーケンスのすべての項目に係数演算子を呼び出すコンテキスト・アイテム式を含みます。

```
(1 to 100)[. mod 5 eq 0]
```

この例の結果は、1 から 100 までの数字で、5 で均等に割り切れる整数のシーケンスになります。

関数呼び出し

関数呼び出しは、QName と、その後に続く括弧で囲んだ 0 個以上の式のリスト (これを引数と呼びます) で構成されます。DB2 XQuery は、組み込み XQuery 関数および DB2 固有の関数の呼び出しをサポートします。

組み込み XQuery 関数は、ネーム・スペース `http://www.w3.org/2005/xpath-functions` にあり、これは接頭部 `fn` にバインドされます。 DB2 固有の関数は、ネーム・スペース `http://www.ibm.com/xmlns/prod/db2/functions` にあり、これは接頭部 `db2-fn` にバインドされます。関数呼び出し内の QName にネーム・スペース接頭部がない場合、関数はデフォルトの関数ネーム・スペース内になければなりません。照会プロローグ内のデフォルトの関数宣言によってネーム・スペースがオーバーライドされていない限り、デフォルトの関数ネーム・スペースは、組み込み XQuery 関数のネーム・スペースです (接頭部 `fn` にバインドされます)。

重要: 関数呼び出しの引数はコンマで分離されるため、括弧を使用して、最上位のコンマ演算子を含む引数式を囲む必要があります。

以下のステップは、DB2 XQuery が関数を評価するために使用するプロセスを説明しています。

1. DB2 XQuery は、関数呼び出しにおいて引数として渡される各式を評価し、各式の値を戻します。

2. 各引数について戻される値が、その引数に期待されるデータ・タイプに変換されます。引数に原子値または原子値のシーケンスが期待される場合は、DB2 XQuery は以下の規則を使用して、引数の値をその期待されるタイプに変換します。
   a. 指定された値に原子化が適用されます。この結果は、原子値のシーケンスです。
   b. 原子値のシーケンス内の xdt:untypedAtomic タイプの各項目は、期待される原子タイプにキャストされます。数値引数が期待される組み込み関数の場合、xdt:untypedAtomic タイプの引数は xs:double にキャストされます。
   c. 原子値のシーケンス内の期待される原子タイプにプロモート可能なすべての数値項目に、数値タイプ・プロモーションが適用されます。数値項目には、xs:integer タイプ (または xs:integer タイプから派生したタイプ)、xs:decimal タイプ、xs:float タイプ、または xs:double タイプの項目が含まれます。
   d. 期待されるタイプが xs:string である場合、原子値のシーケンス内の xs:anyURI タイプまたは xs:anyURI タイプから派生したタイプの各項目は、xs:string にプロモートされます。
3. 関数は、その引数の変換済みの値を使用して評価されます。関数呼び出しの結果は、関数の宣言済みの戻り値のタイプのインスタンスまたはエラーのいずれかです。

例

ストリング引数を使用した関数呼び出し: 以下の関数呼び出しは xs:string タイプの引数を取り、すべての文字が大文字である xs:string タイプの値を戻します。

```
fn:upper-case($ns1_customerinfo/ns1:addr/@country)
```

この例では、fn:upper-case 関数に渡される引数はパス式です。関数の呼び出し時は、パス式が評価され、結果のノード・シーケンスが原子化されます。シーケンス内の各原子値は、期待されるタイプ xs:string にキャストされます。関数は評価され、xs:string タイプの原子値のシーケンスを戻します。

シーケンス引数を使用した関数呼び出し: 以下の関数は、単一引数として、シーケンス (1, 2, 3) を取ります。

```
fn:max((1, 2, 3))
```

関数 fn:max には、原子値のシーケンスである単一引数が期待されるため、ネストされた括弧が必要です。戻り値は 3 です。

パス式

パス式 は、XML ツリー内のノードを識別します。 DB2 XQuery におけるパス式は、XPath 2.0 の構文に基づきます。

パス式は、スラッシュ (/) またはダブルスラッシュ (//) 文字で区切られた 1 つ以上のステップで構成されます。パス式は、ステップ、またはスラッシュ文字またはダブルスラッシュ文字のどちらからでも開始できます。最終ステップの前の各ステップでは、後続のステップのコンテキスト・ノードとして使用されるノードのシーケンスが生成されます。

最初のステップでは、多くの場合、ノードまたはノードのシーケンスを戻す関数呼び出しまたは変数参照を使用して、パスの開始点を指定します。最初の「/」は、パスが、コンテキスト・ノードを含むツリーのルート・ノードから開始されることを示します。最初の「//」は、パスが、コンテキスト・ノードを含むツリーのルート・ノードと、そのルート・ノードのすべての子孫で構成される初期ノード・シーケンスから開始されることを示します。

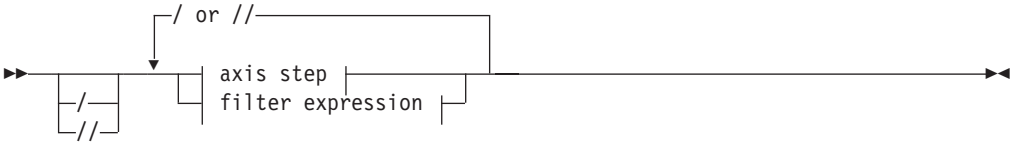
各ステップが、前のステップにより生成された各コンテキスト・ノードに対して 1 回、繰り返し実行されます。これらの反復実行の結果が結合され、後続のステップのためのコンテキスト・ノードのシーケンスが形成されます。ノード ID に基づいて、この結合シーケンスから重複ノードが除去されます。

パス式の値は、パスの最終ステップの結果である、結合された項目のシーケンスです。この値は、ノードのシーケンスまたは原子値のシーケンスのいずれかの可能性があります。パス内の各ステップは後続のステップのコンテキスト・ノードを提供するため、パスの最終ステップは、原子値のシーケンスを戻すことができる唯一のステップです。ノードと原子値が混合したものを戻すパス式の結果はエラーになります。

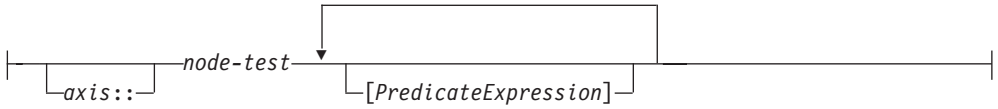
パス式の結果であるノード・シーケンスは、特定の順序であることが保証されません。パス式がいつ順序付けされた結果を戻すかについて理解するには、XQuery 式の結果の順序について説明しているトピックを参照してください。

パス式の構文

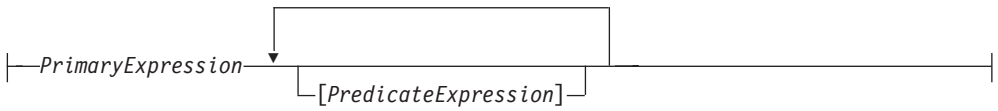
パス式の各ステップは、軸ステップまたはフィルター式のいずれかです。*軸ステップ* は、指定された軸によってコンテキスト・ノードから到達できるノードのシーケンスを戻します。*フィルター式* は、基本式と、それに続く 0 個以上の述部で構成されます。



axis step:



filter expression:



/ 最初のスラッシュ文字 (*/*) は、パスが、コンテキスト・ノードを含むツリーのル

ート・ノード (文書ノードでなければなりません) から開始されることを示します。パス式内のスラッシュ文字は、ステップを分離します。

// 最初のダブルスラッシュ文字 (//) は、パスが、コンテキスト・ノードを含むツリーのルート・ノード (文書ノードでなければなりません) と、そのルート・ノードのすべての子孫で構成される初期ノード・シーケンスから開始されることを示します。ステップ間のダブルスラッシュ文字の意味を理解するには、省略構文に関するトピックを参照してください。

axis
XML 文書またはフラグメントの移動方向。サポートされる軸のリストには、child、descendant、attribute、self、descendant-or-self、および parent などが含まれます。これらの軸の中には、省略構文を使用して表すことができるものもあります。

node-test
軸ステップによって選択される各ノードについて true でなければならない条件。このテストは、ノードの名前に基づいてノードを選択する名前テスト、またはノードの種類に基づいてノードを選択する種類テストのいずれかです。

PrimaryExpression
基本式。

PredicateExpression
シーケンスの項目が保持されているか、廃棄されているかを判別する式。

例

以下の例は、2 つの述部を含む軸ステップを示しています。このステップは、`secretary` 子エレメントおよび `assistant` 子エレメントの両方を持つ、コンテキスト・ノードの子であるすべての `employee` を選択します。

```
child::employee[secretary][assistant]
```

以下の例では、パス式内のステップとしてフィルター式を使用しています。この式は、指定された book 内で複数の footnote を含むすべての chapter または appendix を戻します。

```
$book/(chapter | appendix)[fn:count(footnote)> 1]
```

軸ステップ

*軸*ステップは、軸、ノード・テスト、述部の 3 つの部分から構成されます。オプションの*軸* は、移動の方向を指定します。*ノード・テスト*は、ノードの選択に使用する基準を指定します。ゼロ以上の*述部* は、ステップで戻されるシーケンスをフィルタリングします。軸ステップの結果は、常にゼロ以上のノードのシーケンスになります。

軸ステップは、フォワード・ステップまたはリバース・ステップに分けられます。*フォワード・ステップ* は、コンテキスト・ノードから開始し、文書の順序で XML ツリーを移動します。*リバース・ステップ* は、コンテキスト・ノードから開始し、文書の順序の逆で XML ツリーを移動します。コンテキストの項目がノードでない場合、式の結果はエラーになります。

軸ステップの非省略構文は、二重コロンで分離されている軸名およびノード・テストと、その後に続くゼロ以上の述部から構成されます。軸式の構文は、軸を省略し、省略表現を使用することで短縮できます。

以下の例では、child が軸の名前、para がこの軸上で選択されるエレメント・ノードの名前です。

```
child::para
```

この例の軸ステップでは、コンテキスト・ノードの子であるすべての para エレメントを選択します。

軸

軸 は、XML 文書内を移動する方向を指定する軸ステップの一部です。

軸は、フォワード軸またはリバース軸のいずれかに分類できます。*フォワード軸* には、コンテキスト・ノード、および文書の順序においてコンテキスト・ノードの後にあるノードが含まれます。*リバース軸* には、コンテキスト・ノード、および文書の順序においてコンテキスト・ノードの前にあるノードが含まれます。

以下の表では、 DB2 XQuery でサポートされる軸を説明します。

表 16. DB2 XQuery でサポートされる軸

軸	説明	方向	注
child	コンテキスト・ノードの子を戻します。	順方向 (フォワード)	子を持つノードは、文書ノードおよびエレメント・ノードのみです。コンテキスト・ノードがそれ以外の種類のノードである場合、またはコンテキスト・ノードが、子を持たない文書ノードまたはエレメント・ノードである場合、child 軸は空のシーケンスです。エレメント・ノード、処理命令ノード、コメント・ノード、またはテキスト・ノードを、文書ノードまたはエレメント・ノードの子とすることができます。属性ノードおよび文書ノードは子として表示されることはありません。
descendant	コンテキスト・ノードの子孫 (子、子の子など) を戻します。	順方向 (フォワード)	
属性	コンテキスト・ノードの属性を戻します。	順方向 (フォワード)	この軸は、コンテキスト・ノードがエレメント・ノードでない場合、空です。
self	コンテキスト・ノードのみを戻します。	順方向 (フォワード)	
descendant-or-self	コンテキスト・ノードおよびコンテキスト・ノードの子孫を戻します。	順方向 (フォワード)	
parent	コンテキスト・ノードの親を戻します。コンテキスト・ノードが親を持たない場合、空のシーケンスを戻します。	逆方向 (リバース)	属性ノードがエレメント・ノードの子になることはありませんが、エレメント・ノードは属性ノードの親となることができます。

軸ステップがノードのシーケンスを選択すると、各ノードには、そのシーケンスの位置に対応するコンテキストの位置が割り当てられます。軸がフォワード軸の場

合、コンテキストの位置は、文書の順序で 1 からノードに割り当てられます。軸がリバース軸の場合、文書の逆順で 1 からノードに割り当てられます。コンテキストの位置を割り当てると、その位置を指定することで、シーケンスからノードを選択できます。

ノード・テスト

ノード・テスト は、軸ステップにより選択される各ノードについて true でなければならない条件です。ノード・テストは、名前テストまたは種類テストのいずれかとして表現できます。*名前テスト* は、ノードの名前に基づいてノードを選択します。*種類テスト* は、ノードの種類に基づいてノードを選択します。

名前テスト

名前テストは、QName またはワイルドカードで構成されます。軸ステップで名前テストが指定されると、ステップは、QName またはワイルドカードに一致する指定された軸上のノードを選択します。名前テストが属性軸で指定される場合、ステップは名前テストに一致する任意の属性を選択します。他のすべての軸において、ステップは名前テストに一致する任意のエレメントを選択します。 QName は、ノードの拡張 QName が、名前テストで指定される拡張 QName に (コード・ポイント・ベースで) 等しい場合に一致します。2 つの拡張 QName は、そのネーム・スペース URI が等しく、ローカル名も等しい場合に (ネーム・スペース接頭部が等しくなくても) 等しくなります。

重要: 名前テストで指定される接頭部は、式の静的に既知のネーム・スペースの 1 つに対応する必要があります。属性軸について実行される名前テストの場合、接頭部なしの QName にはネーム・スペース URI はありません。 他のすべての軸について実行される名前テストの場合、接頭部なしの QName にはデフォルト・エレメント/タイプ・ネーム・スペースのネーム・スペース URI があります。

以下の表は、DB2 XQuery でサポートされる名前テストを説明しています。

表 17. DB2 XQuery でサポートされる名前テスト

テスト	説明	例
QName	その QName が、指定された QName に等しい (指定された軸上の) 任意のノードに一致します。軸が属性軸の場合、このテストは属性ノードに一致します。他のすべての軸の場合、このテストはエレメント・ノードに一致します。	式 `child::para` において、名前テスト `para` は、子軸上のすべての `para` エレメントを選択します。
*	指定された軸上のすべてのノードに一致します。軸が属性軸の場合、このテストはすべての属性ノードに一致します。他のすべての軸の場合、このテストはすべてのエレメント・ノードに一致します。	式 `child::*` において、名前テスト `*` は、子軸上のすべてのエレメントに一致します。

表 17. DB2 XQuery でサポートされる名前テスト (続き)

テスト	説明	例
NCName:*	QName の接頭部部分を表す *NCName* を指定します。この名前テストは、そのネーム・スペース URI が、接頭部がバインドされるネーム・スペース URI に一致する (指定された軸上の) すべてのノードに一致します。軸が属性軸の場合、このテストは属性ノードに一致します。他のすべての軸の場合、このテストはエレメント・ノードに一致します。	式 `child::ns1:*` において、名前テスト `ns1:*` は、接頭部 `ns1` にバインドされているネーム・スペースに関連付けられた子軸上のすべてのエレメントに一致します。
*:*NCName*	QName のローカル部分を表す *NCName* を指定します。この名前テストは、そのローカル名が *NCName* に等しい (指定された軸上の) 任意のノードに一致します。軸が属性軸の場合、このテストは属性ノードに一致します。他のすべての軸の場合、このテストはエレメント・ノードに一致します。	式 `child::*:customerinfo` において、名前テスト `*:customerinfo` は、エレメント名に関連付けられたネーム・スペースには関係なく、ローカル名 `customerinfo` を持つ、子軸上のすべてのエレメントに一致します。

種類テスト

軸ステップで種類テストが指定されると、ステップは、種類テストに一致する指定された軸上のノードのみを選択します。以下の表は、DB2 XQuery でサポートされる種類テストを説明しています。

表 18. DB2 XQuery でサポートされる種類テスト

テスト	説明	例
node()	指定された軸上の任意のノードに一致します。	式 `child::node()` において、種類テスト `node()` は、子軸上の任意のノードを選択します。
text()	指定された軸上の任意のテキスト・ノードに一致します。	式 `child::text()` において、種類テスト `text()` は、子軸上の任意のテキスト・ノードを選択します。
comment()	指定された軸上の任意のコメント・ノードに一致します。	式 `child::comment()` において、種類テスト `comment()` は、子軸上の任意のコメント・ノードを選択します。
processing-instruction()	指定された軸上の任意の処理命令ノードに一致します。	式 `child::processing-instruction()` において、種類テスト `processing-instruction()` は、子軸上の任意の処理命令ノードを選択します。

表 18. DB2 XQuery でサポートされる種類テスト (続き)

テスト	説明	例
element() または element(*)	指定された軸上の任意のエレメント・ノードに一致します。	式 `child::element()` において、種類テスト `element()` は、子軸上の任意のエレメント・ノードを選択します。式 `child::element(*)` において、種類テスト `element(*)` は、子軸上の任意のエレメント・ノードを選択します。
attribute() または attribute(*)	指定された軸上の任意の属性ノードに一致します。	式 `child::attribute()` において、種類テスト `attribute()` は、子軸上の任意の属性ノードを選択します。式 `child::attribute(*)` において、種類テスト `attribute(*)` は、子軸上の任意の属性ノードを選択します。
document-node()	指定された軸上の任意の文書ノードに一致します。	式 `self::document-node()` において、種類テスト `document-node()` は、コンテキスト・ノードである文書ノードを選択します。

パス式の省略構文

XQuery には、パス式の軸を示すための省略構文があります。

以下の表では、パス式で許可されている省略形を説明します。

表 19. パス式の省略構文

省略構文	説明	例
軸の指定なし	ノード・テストとして軸ステップが `attribute()` を指定したとき以外は、`child::` の省略形です。軸ステップが属性テストを指定するときは、省略される軸は、`attribute::` の省略形です。	パス式 `section/para` は、`child::section/child::para` の省略形です。パス式 `section/attribute()` は、`child::section/attribute::attribute()` の省略形です。
@	`attribute::` の省略形です。	パス式 `section/@id` は、`child::section/attribute::id` の省略形です。

表 19. パス式の省略構文 (続き)

省略構文	説明	例
//	この省略形はパス式の先頭にあるとき以外は、/descendant-or-self::node()/ の省略形です。 この省略表記がパス式の先頭に存在すると、その軸ステップでは、コンテキスト・ノードがあるツリーのルートを含む最初のノード・シーケンス、およびそのルートの下位にあるすべてのノードが選択されます。この式では、ルート・ノードが文書ノードでない場合、エラーが戻されます。	パス式 div1//para は、child::div1/descendant-or-self::node() /child::para の省略形です。
..	parent::node() の省略形です。	パス式 ../title は、parent::node()/child::title の省略形です。

省略構文および非省略構文の例

以下の表は、省略構文および非省略構文の例です。

表 20. 非省略構文および省略構文

非省略構文	省略構文	結果
child::para	para	コンテキスト・ノードの子である para エレメントを選択します。
child::*	*	コンテキスト・ノードの子であるすべてのエレメントを選択します。
child::text()	text()	コンテキスト・ノードの子であるすべてのテキスト・ノードを選択します。
child::node()	node()	コンテキスト・ノードのすべての子を選択します。この式では、属性ノードは戻されません。属性はノードの子とは認識されないためです。
attribute::name	@name	コンテキスト・ノードの name 属性を選択します。
attribute::*	@*	コンテキスト・ノードのすべての属性を選択します。
child::para[fn:position() = 1]	para[1]	コンテキスト・ノードの子である最初の para エレメントを選択します。
child::para[fn:position() = fn:last()]	para[fn:last()]	コンテキスト・ノードの子である最後の para エレメントを選択します。

表 20. 非省略構文および省略構文 (続き)

非省略構文	省略構文	結果
`/child::book/child::chapter[fn:position() = 5] /child::section[fn:position() = 2]`	`/book/chapter[5]/section[2]`	親がコンテキスト・ノードを含む文書ノードである book の 5 番目の chapter の、2 番目の section を選択します。
`child::para[attribute::type="warning"]`	`para[@type="warning"]`	値が warning のタイプ属性を持つコンテキスト・ノードのすべての子 para を選択します。
`child::para[attribute::type='warning'] [fn:position() = 5]`	`para[@type="warning"][5]`	値が warning のタイプ属性を持つコンテキスト・ノードの 5 番目の子 para を選択します。
`child::para[fn:position() = 5] [attribute::type="warning"]`	`para[5][@type="warning"]`	子が値 warning のタイプ属性を持つ場合、コンテキスト・ノードの 5 番目の子 para を選択します。
`child::chapter[child::title='Introduction']`	`chapter[title="Introduction"]`	型付き値がストリング Introduction と同じである 1 つ以上の子 title を持つコンテキスト・ノードの子 chapter を選択します。
`child::chapter[child::title]`	`chapter[title]`	1 つ以上の子 title を持つコンテキスト・ノードの子 chapter を選択します。

述部

述部 は、修飾する項目を保持することにより、シーケンスをフィルタリングします。述部は、大括弧 ([]) で囲まれた式 (述部式と呼びます) で構成されます。述部式は、選択された項目をコンテキスト・アイテムとして使用し、シーケンス内の各項目について 1 回評価されます。述部式の各評価により、*述部真理値* と呼ばれる xs:boolean 値が戻されます。述部真理値が true である項目は保持され、述部真理値が false である項目は廃棄されます。

述部真理値を判別するために、以下の規則が使用されます。

- 述部式が非数値を戻す場合、述部真理値は、述部式の有効なブール値です。
- 述部式が数値を戻す場合、述部真理値は、シーケンス内での位置がその数値に等しい項目についてのみ true です。その他の項目については、述部真理値は false です。このような種類の述部を、*数値述部* または*定位置述部* と呼びます。例えば、式 `$products[5]` において、数値述部 `[5]` は、変数 `$products` にバインドされたシーケンス内の 5 番目の項目のみを保持します。

重要: 数値述部によってシーケンスから選択される項目は、シーケンスに決定論的順序がある場合にのみ決定論的となります。

ヒント: 述部の動作は、述部式が数値を戻すかどうかに依存しますが、述部式を見てもこのことが明白でないこともあります。 `[fn:boolean(PredicateExpression)]` のように、 fn:boolean 関数を使用して、述部が必ず有効なブール値を使用するようにすることができます。あるいは、`[fn:position() eq PredicateExpression]` のように、fn:position 関数を使用することにより、述部が必ず定位置述部のように動作するように指定できます。

以下の例には、述部が含まれています。

- `chapter[2]` は、コンテキスト・ノードの子である 2 番目の `chapter` エレメントを選択します。
- `descendant::toy[@color = "Red"]` は、 `toy` という名前で、カラー属性の値が「Red」になっているエレメントである、コンテキスト・ノードのすべての子孫を選択します。
- `employee[secretary][assistant]`は、 secretary 子エレメントと assistant 子エレメントの両方を持つ、コンテキスト・ノードの子であるすべての employee を選択します。
- `(<cat />, <dog />, 47, <zebra />)[2]` は、エレメント `<dog />` を戻します。

シーケンス式

シーケンス式は、項目のシーケンスを構成、フィルタリング、および結合します。シーケンスはネストされません。例えば、値 1、(2, 3)、および ( ) を単一のシーケンスに結合した結果は、シーケンス (1, 2, 3) です。

シーケンスを構成する式

シーケンスは、コンマ演算子または範囲式のいずれかを使用して構成できます。

コンマ演算子

コンマ演算子を使用してシーケンスを構成するには、コンマで区切られた複数のオペランド (式) を指定します。シーケンス式が評価されるときは、コンマ演算子がそのオペランドを 1 つずつ評価し、結果のシーケンスを 1 つの結果シーケンスに順番に連結します。例えば、以下の式は、5 つの整数を含む 1 つのシーケンスになります。

```
(15, 1, 3, 5, 7)
```

シーケンスには、重複する原子値およびノードを含めることができます。ただし、シーケンスを別のシーケンスの項目にすることはできません。複数の入力シーケンスを連結して新規シーケンスが作成される場合、その新規シーケンスには入力シーケンスのすべての項目が含まれ、シーケンスの長さは入力シーケンスを合計した長さになります。

以下の式では、コンマ演算子を使用して、シーケンスを構成します。

- この式では、長さがそれぞれ 1、2、ゼロ、および 2 の 4 つのシーケンスを長さ 5 の 1 つのシーケンスに結合します。この式の結果は、シーケンス 10, 1, 2, 3, 4 になります。

  ```
  (10, (1, 2), (), (3, 4))
  ```

- この式の結果は、コンテキスト・ノードの子であるすべての salary エレメント、その後にそのコンテキスト・ノードの子である bonus エレメントが続くシーケンスになります。

  ```
  (salary, bonus)
  ```

- 変数 $price を値 10.50 にバインドすると想定すると、この式の結果は、シーケンス 10.50, 10.50 になります。

  ```
  ($price, $price)
  ```

範囲式

範囲式は、連続した整数のシーケンスを構成します。範囲式は、**to** 演算子で区切られた 2 つのオペランド (式) から構成されます。各オペランドの値は、xs:integer タイプの値に変換可能である必要があります。いずれかのオペランドが空のシーケンスである場合、または最初のオペランドから派生した整数が 2 番目のオペランドから派生した整数よりも大きい場合、範囲式の結果は空のシーケンスになります。それ以外の場合、結果には、2 つのオペランドから派生した 2 つの整数、その 2 つの整数の間に含まれるすべての整数が、小さい数値から順に記載されたシーケンスです。例えば、以下の範囲式は、シーケンス 1, 2, 3, 4 に評価されます。

```
(1 to 4)
```

以下の例では、範囲式を使用してシーケンスを構成します。

- 以下の例では、1 つのオペランドとして範囲式を使用してシーケンスを構成します。シーケンス式は、シーケンス 10, 1, 2, 3, 4 に評価されます。

  ```
  (10, 1 to 4)
  ```

- 以下の例では、単一整数 10 を含む長さ 1 のシーケンスを構成します。

  ```
  10 to 10
  ```

- この例の結果は、長さゼロのシーケンスになります。

  ```
  15 to 10
  ```

- 以下の例では、fn:reverse 関数を使用して、大きい数値から順に 6 個の整数のシーケンスを構成します。このシーケンス式は、シーケンス 15, 14, 13, 12, 11, 10 に評価されます。

  ```
  fn:reverse(10 to 15)
  ```

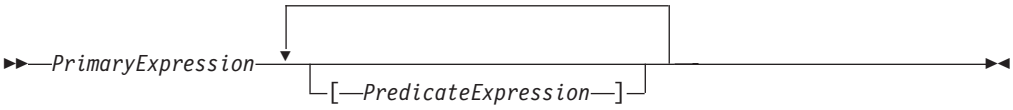
フィルター式

フィルター式は、基本式と、それに続く 0 個以上の述部で構成されます。述部が存在する場合、基本式によって戻されるシーケンスをフィルタリングします。

フィルター式の結果は、基本式で戻される true の述部真理値を持つすべての項目から構成されます。述部が指定されていない場合、結果はそのまま基本式の結果になります。結果のシーケンスに含まれる項目は、基本式で戻される項目と同じ順序に

なります。述部の評価時に、各項目は、その述部によってフィルタリングされているシーケンス内の位置を示すコンテキストの位置になります。1 番目のコンテキストの位置は、1 です。

構文



PrimaryExpression
基本式。

PredicateExpression
シーケンスの項目が保持されているか、廃棄されているかを判別する式。

例

以下の例では、フィルター式を使用して、フィルタリングされたシーケンスを戻します。

- 製品のシーケンスを変数にバインドすると、この式は、100 より高い価格の製品のみを戻します。

  ```
  $products[price gt 100]
  ```

- 以下の式では、述部を持つ範囲式を使用して、5 で割り切れる 1 から 100 までのすべての整数をリストします。範囲式は、括弧で囲まれているため、基本式として処理されます。

  ```
  (1 to 100)[. mod 5 eq 0]
  ```

- 以下の式の結果は、整数の 5 になります。

  ```
  (1 to 21)[5]
  ```

- 以下の式は、フィルター式をパス式の 1 つのステップとして使用します。式は、変数 $book にバインドされるブック内の最後の章 (chapter) または付録 (appendix) を戻します。

  ```
  $book/(chapter | appendix)[fn:last()]
  ```

ノードのシーケンスを結合するための式

DB2 XQuery では、ノードのシーケンスを結合するための演算子が提供されています。これらの演算子には、**union**、**intersect**、および **except** があります。

以下の表は、ノードのシーケンスを結合するために使用可能な演算子を説明しています。

表 21. ノードのシーケンスを結合するための XQuery 演算子

演算子	説明
union または |	オペランドとして 2 つのノード・シーケンスを取り、どちらかのオペランドに現れるすべてのノードを含むシーケンスを戻します。 **union** キーワードと | 文字は等価です。

表 21. ノードのシーケンスを結合するための XQuery 演算子 (続き)

演算子	説明
intersect	オペランドとして 2 つのノード・シーケンスを取り、両方のオペランドに現れるすべてのノードを含むシーケンスを戻します。
except	オペランドとして 2 つのノード・シーケンスを取り、最初のオペランドには現れるが、2 番目のオペランドには現れないすべてのノードを含むシーケンスを戻します。

これらのすべての演算子は、ノード ID に基づいて、結果のシーケンスから重複ノードを除去します。結果のシーケンスは文書の順序で戻されます。

union、**intersect**、または **except** のオペランドは、ノードのみを含むシーケンスに解決される必要があります。オペランドにノードではない項目が含まれている場合、エラーが戻されます。

このトピックで説明されている演算子のほかに、DB2 XQuery には、シーケンスの項目またはサブシーケンスへの索引付きアクセス (fn:index-of)、シーケンスの項目の索引付き挿入または除去 (fn:insert-before および fn:remove)、およびシーケンスからの重複項目の除去 (fn:distinct-values) のための関数が提供されています。

例

これらの例では、変数 `$managers` が、管理者である従業員を表す従業員ノードのセットにバインドされ、変数 `$students` が、学生である従業員を表す従業員ノードのセットにバインドされていることを想定しています。

以下の式はすべて、演算子を使用してノードのシーケンスを結合している有効な例です。

- `$managers union $students` は、管理者または学生のいずれかである従業員を表すノードのセットを戻します。
- `$managers intersect $students` は、管理者であり学生でもある従業員を表すノードのセットを戻します。
- `$managers except $students` は、管理者であるが学生ではない従業員を表すノードのセットを戻します。

算術式

算術式では、加算、減算、乗算、除算、モジュラスなどの演算を実行します。

次の表では、算術演算子について説明し、演算子優先順位の高いものから低いものの順にリストしています。単項演算子の優先順位は、2 項演算子よりも高くなります。ただし、括弧を使用した場合は、2 項演算子の計算が先に行われます。

表 22. XQuery の算術演算子

演算子	目的	結合順序
- (単項)、+ (単項)	オペランド値を反対の符号の値にする、オペランド値を維持する	右から左

表 22. XQuery の算術演算子 (続き)

演算子	目的	結合順序
*、 div、 idiv、 mod	乗算、除算、整数除算、係数	左から右
+, -	加算、減算	左から右

注: 減算演算子の前には空白文字が必要です。空白文字がないと、演算子が前のトークンの一部であると解釈されます。例えば、`a-b` は名前と解釈されますが、`a - b` や `a -b` は、算術演算と解釈されます。

算術式の結果は、数値、空のシーケンス、またはエラーのいずれかです。算術式の値が計算されるときは、各オペランドが原子化され (原子値に変換され)、次の規則が適用されます。

- 原子化されたオペランドが空のシーケンスである場合、算術式の結果も空のシーケンスです。
- 原子化されたオペランドが複数の値を含むシーケンスである場合、エラーが戻されます。
- 原子化されたオペランドが非型付き原子値 (xdt:untypedAtomic) である場合、値は xs:double にキャストされます。キャストに失敗すると、エラーが戻されます。

評価後のオペランドのタイプが算術演算子に対して有効な組み合わせである場合、その演算子は原子化されたオペランドに適用され、この演算の結果は、原子値またはエラーになります (ゼロ除算の結果のエラーなど)。オペランドのタイプが算術演算子に有効な組み合わせでない場合、エラーが戻されます。

79 ページの表 23 は、算術演算子に有効なタイプの組み合わせを示します。この表では、文字 A は式の第 1 オペランドを、文字 B は第 2 オペランドを示します。数値という語は、xs:integer タイプ、xs:decimal タイプ、xs:float タイプ、xs:double タイプ、またはこのいずれかから派生したタイプを示します。演算子の結果タイプが数値としてリストされている場合、結果タイプは、すべてのオペランドをサブタイプ置換およびタイプ・プロモーションによって変換できるタイプで、順序付けリストの先頭にあるタイプ (xs:integer、xs:decimal、xs:float、xs:double) になります。

表 23. 算術式のオペランドに有効なタイプ

演算子とオペランド	オペランド A タイプ	オペランド B タイプ	結果タイプ
A + B	数値	数値	数値
	xs:date	xdt:yearMonthDuration	xs:date
	xdt:yearMonthDuration	xs:date	xs:date
	xs:date	xdt:dayTimeDuration	xs:date
	xdt:dayTimeDuration	xs:date	xs:date
	xs:time	xdt:dayTimeDuration	xs:time
	xdt:dayTimeDuration	xs:time	xs:time
	xs:dateTime	xdt:yearMonthDuration	xs:dateTime
	xdt:yearMonthDuration	xs:dateTime	xs:dateTime
	xs:dateTime	xdt:dayTimeDuration	xs:dateTime
	xdt:dayTimeDuration	xs:dateTime	xs:dateTime
	xdt:yearMonthDuration	xdt:yearMonthDuration	xdt:yearMonthDuration
	xdt:dayTimeDuration	xdt:dayTimeDuration	xdt:dayTimeDuration
A - B	数値	数値	数値
	xs:date	xs:date	xdt:dayTimeDuration
	xs:date	xdt:yearMonthDuration	xs:date
	xs:date	xdt:dayTimeDuration	xs:date
	xs:time	xs:time	xdt:dayTimeDuration
	xs:time	xdt:dayTimeDuration	xs:time
	xs:dateTime	xs:dateTime	xdt:dayTimeDuration
	xs:dateTime	xdt:yearMonthDuration	xs:dateTime
	xs:dateTime	xdt:dayTimeDuration	xs:dateTime
	xdt:yearMonthDuration	xdt:yearMonthDuration	xdt:yearMonthDuration
	xdt:dayTimeDuration	xdt:dayTimeDuration	xdt:dayTimeDuration
A * B	数値	数値	数値
	xdt:yearMonthDuration	数値	xdt:yearMonthDuration
	数値	xdt:yearMonthDuration	xdt:yearMonthDuration
	xdt:dayTimeDuration	数値	xdt:dayTimeDuration
	数値	xdt:dayTimeDuration	xdt:dayTimeDuration
A idiv B	数値	数値	xs:integer
A div B	数値	数値	数値 (両方のオペランドが xs:integer の場合は xs:decimal)
	xdt:yearMonthDuration	数値	xdt:yearMonthDuration
	xdt:dayTimeDuration	数値	xdt:dayTimeDuration
	xdt:yearMonthDuration	xdt:yearMonthDuration	xs:decimal
	xdt:dayTimeDuration	xdt:dayTimeDuration	xs:decimal
A mod B	数値	数値	数値

例

- 以下の最初の式は、xs:decimal 値 -1.5 を戻し、2 番目の式は xs:integer 値 -1 を戻します。

```
-3 div 2
-3 idiv 2
```

- 以下の式では、2 つの日付値の減算の結果は、xdt:dayTimeDuration タイプの値になります。

```
$emp/hiredate - $emp/birthdate
```

- 以下の例は、減算演算子と、変数名 `unit-price` および `unit-discount` で使用されるハイフンの違いを示しています。

```
$unit-price - $unit-discount
```

比較式

比較式では、2 つの値を比較します。XQuery では、3 種類の比較式 (値比較、一般比較、およびノード比較) を提供しています。

値比較

値比較は、2 つの原子値を比較します。値比較演算子には、 **eq**、**ne**、**lt**、**le**、**gt**、および **ge** があります。

以下の表は、これらの演算子を説明しています。

表 24. XQuery における値比較演算子

演算子	目的
eq	1 番目の値が 2 番目の値と等しい場合に、true を戻します。
ne	1 番目の値が 2 番目の値と等しくない場合に、true を戻します。
lt	1 番目の値が 2 番目の値より小さい場合に、true を戻します。
le	1 番目の値が 2 番目の値より小さいか等しい場合に、true を戻します。
gt	1 番目の値が 2 番目の値より大きい場合に、true を戻します。
ge	1 番目の値が 2 番目の値より大きいか等しい場合に、true を戻します。

2 つの値は、それらが同じタイプである場合、または一方のオペランドのタイプが他方のオペランドのタイプのサブタイプである場合に比較できます。数値タイプ (xs:float タイプ、xs:integer タイプ、xs:decimal タイプ、xs:double タイプ、およびこれらのタイプから派生したタイプ) の 2 つのオペランドは比較できます。また、xs:string および xs:anyURI の値も比較できます。

特殊値: xs:float 値および xs:double 値の場合、正のゼロと負のゼロは等しいとされます。 INF は INF と等しく、-INF は -INF と等しくなります。NaN は NaN 自身と等しくありません。正の無限大は、他のすべての非 NaN 値より大きくなります。負の無限大は、他のすべての非 NaN 値より小さくなります。NaN **ne** NaN は true で、NaN 値を含むその他のすべての比較は false です。xs:QName タイプの 2 つの値は、そのネーム・スペース URI が等しく、ローカル名も等しい場合に等しいと見なされます (ネーム・スペース接頭部は重要ではありません)。

値比較の結果は、ブール値、空のシーケンス、またはエラーのいずれかです。値比較の評価時には、各オペランドが原子化 (原子値に変換) され、以下の規則が適用されます。

- 原子化されたオペランドのいずれかが空のシーケンスの場合、値比較の結果は空のシーケンスです。
- 原子化されたオペランドのいずれかが複数の値を含むシーケンスである場合、エラーが戻されます。
- 原子化されたオペランドのいずれかが非型付き原子値 (xdt:untypedAtomic) の場合、その値は xs:string にキャストされます。

  xdt:untypedAtomic タイプの値を xs:string にキャストすると、値比較が推移的になります。これに対して一般比較は、非型付きデータをキャストするために異なる規則に従い、そのため推移的にはなりません。値比較の推移性は、それによりタイプ変換時に精度が失われる可能性があるという欠点も併せ持ちます。例えば、わずかに異なる 2 つの xs:integer 値が両方とも同じ xs:float 値と等しいと見なされる可能性があります。これは、xs:float の精度が xs:integer より低いためです。
- 評価後のオペランドのタイプが演算子にとって有効な組み合わせである場合、原子化されたオペランドに演算子が適用され、比較の結果は true または false のいずれかになります。オペランドのタイプが比較演算子に有効な組み合わせでない場合、エラーが戻されます。

以下のタイプは、**eq** または **ne** 演算子を使用して比較できます。グレゴリオ・タイプという用語は、タイプ xs:gYearMonth、xs:gYear、xs:gMonthDay、xs:gDay、および xs:gMonth を指します。 2 つのグレゴリオ・タイプのオペランドを受け入れる 2 項演算子の場合、両方のオペランドが同じタイプを持っている必要があります (例えば、一方のオペランドが xs:gDay タイプの場合、他方のオペランドも xs:gDay タイプでなければなりません)。数値という用語は、xs:integer タイプ、xs:decimal タイプ、xs:float タイプ、xs:double タイプ、およびこのいずれかから派生したタイプを指します。数値を含む比較においては、サブタイプ置換および数値タイプのプロモーションが使用されて、オペランドは、すべてのオペランドが変換可能な順序付けリスト (xs:integer、xs:decimal、xs:float、xs:double) の最初のタイプに変換されます。

- 数値
- xs:boolean
- xs:string
- xs:date
- xs:time
- xs:dateTime
- xs:duration
- xdt:yearMonthDuration
- xdt:dayTimeDuration
- グレゴリオ・タイプ
- xs:hexBinary
- xs:base64Binary

- xs:QName
- xs:NOTATION

以下のタイプは、**gt**、**lt**、**ge**、および **le** 演算子を使用して比較できます。数値タイプという用語は、タイプ xs:integer、xs:decimal、xs:float、および xs:double を指します。数値を含む比較においては、サブタイプ置換および数値タイプのプロモーションが使用されて、オペランドは、すべてのオペランドが変換可能な順序付けリスト (xs:integer、xs:decimal、xs:float、xs:double) の最初のタイプに変換されます。

- 数値
- xs:boolean
- xs:string
- xs:date
- xs:time
- xs:dateTime
- xdt:yearMonthDuration
- xdt:dayTimeDuration

例

- 以下の比較は、式 $book/author によって戻されるノードを原子化します。この比較は、原子化の結果が、xs:string または xdt:untypedAtomic のインスタンスとしての値「Kennedy」である場合にのみ true です。原子化の結果が複数の値を含むシーケンスの場合、エラーが戻されます。

  ```
  $book1/author eq "Kennedy"
  ```

- 以下のパス式には、重量が 100 より大きい製品を選択する述部が含まれています。重量のサブエレメントを持たない製品については、述部の値は空のシーケンスになり、製品は選択されません。

  ```
  //product[weight gt 100]
  ```

- 以下の比較は、いずれの場合も、構成される 2 つのノードが異なる ID または名前を持っていても、原子化後に同じ値を持つため、 true になります。

  ```
  <a>5</a> eq <a>5</a>
  <a>5</a> eq <b>5</b>
  ```

一般比較

一般比較は、任意の長さの 2 つのシーケンスを比較して、最初のシーケンスの少なくとも 1 つの項目と 2 番目のシーケンスの少なくとも 1 つの項目が、指定された比較を満たしているかどうかを判別します。一般比較演算子には、
=、!=、<、<=、>、および >= があります。

以下の表は、これらの演算子を説明しています。

表 25. XQuery における値比較演算子

演算子	目的
=	1 番目のシーケンス内のいずれかの値が 2 番目のシーケンス内のいずれかの値と等しい場合に、 true を戻します。
!=	1 番目のシーケンス内のいずれかの値が 2 番目のシーケンス内のいずれかの値と等しくない場合に、true を戻します。

表 25. XQuery における値比較演算子 (続き)

演算子	目的
<	1 番目のシーケンス内のいずれかの値が 2 番目のシーケンス内のいずれかの値より小さい場合に、true を戻します。
<=	1 番目のシーケンス内のいずれかの値が 2 番目のシーケンス内のいずれかの値より小さいか等しい場合に、true を戻します。
>	1 番目のシーケンス内のいずれかの値が 2 番目のシーケンス内のいずれかの値より大きい場合に、true を戻します。
>=	1 番目のシーケンス内のいずれかの値が 2 番目のシーケンス内のいずれかの値より大きいか等しい場合に、true を戻します。

一般比較の結果は、ブール値またはエラーのいずれかです。一般比較の評価時には、各オペランドが原子化 (原子値のシーケンスに変換) されます。最初のシーケンスが 2 番目のシーケンスと比較され、最初のシーケンスの少なくとも 1 つの項目と 2 番目のシーケンスの少なくとも 1 つの項目が、指定された比較を満たしているかどうかが判別されます。個々の原子値の比較時には、以下の規則が適用されます。

- 一方の原子値が xdt:untypedAtomic のインスタンスであり、他方の原子値が数値タイプのインスタンスである場合、非型付き値が xs:double タイプにキャストされます。
- 一方の原子値が xdt:untypedAtomic のインスタンスであり、他方の原子値が xdt:untypedAtomic または xs:string のインスタンスである場合、xdt:untypedAtomic 値は xs:string タイプにキャストされます。
- 一方の原子値が xdt:untypedAtomic のインスタンスであり、他方の原子値が xs:string、xdt:untypedAtomic、または任意の数値タイプのインスタンスでない場合、xdt:untypedAtomic 値は他方の値のタイプにキャストされます。

タイプが前述のようにキャストされた後、値比較演算子 **eq**、**ne**、**lt**、**le**、**gt**、または **ge** の 1 つを使用して原子値が比較されます。

比較の結果は、原子値のペア (一方は第 1 オペランドのシーケンスに、他方は第 2 オペランドのシーケンスに) があり、その比較が true である場合は true です。

注: エラーが発生した場合、一般比較の結果はブール値またはエラーのいずれかになる可能性があります。例えば、比較 `(1, 2) = (2, "Hello")` は、`2 eq 2` が true であるため true を戻すか、値 1 を値 ″Hello″ と比較できないためエラーを戻す可能性があります。

ヒント: 項目ベースで 2 つのシーケンスを比較するには、XQuery 関数 fn:deep-equal を使用します。

例

- 以下の比較は、`$book1` の一部の author サブエレメントの型付き値が、 xs:string または xdt:untypedAtomic のインスタンスとして ″Kennedy″ である場合に true です。

```
$book1/author = "Kennedy"
```

- 以下の例には、3 つの一般比較が含まれています。最初の 2 つの比較の値は true、3 つ目の比較の値は false です。この例では、一般比較が推移的でないという事実を説明しています。

```
(1, 2) = (2, 3)
(2, 3) = (3, 4)
(1, 2) = (3, 4)
```

- 以下の例には 2 つの一般比較が含まれており、どちらも true です。この例では、= 演算子および != 演算子が相互に逆ではないという事実を説明しています。

```
(1, 2) = (2, 3)
(1, 2) != (2, 3)
```

- 以下の例では、変数 $a、$b、および $c が、タイプ・アノテーション xdt:untypedAtomic を持つエレメント・ノードにバインドされます。エレメント・ノードには、それぞれストリング値「1」、「2」、および「2.0」が含まれます。この例において、以下の式では、$b および $c にバインドされる値 (「2」および「2.0」) がストリングとして比較されるため、false が戻されます。

```
($a, $b) = ($c, 3.0)
```

  ただし、以下の式では、$b にバインドされる値 (「2」) および値 2.0 が数値として比較されるため、true が戻されます。

```
($a, $b) = ($c, 2.0)
```

ノード比較

ノード比較は、2 つのノードを比較します。ノードが同一 ID を共用しているかどうか、または文書の順序において一方のノードが他方のノードより前になるか後になるかを判別するために、ノードを比較することができます。

以下の表は、XQuery で使用可能なノード比較演算子を説明しています。

表 26. XQuery におけるノード比較演算子

演算子	目的
is	比較される 2 つのノードが同じ ID を持つ場合に、 true を戻します。
<<	文書の順序で第 1 オペランド・ノードが第 2 オペランド・ノードに先行する場合に、true を戻します。
>>	文書の順序で第 1 オペランド・ノードが第 2 オペランド・ノードの後に続く場合に、true を戻します。

ノード比較の結果は、ブール値、空のシーケンス、またはエラーのいずれかです。ノード比較の結果は、以下の規則によって定義されます。

- 各オペランドは、単一ノードまたは空のシーケンスのいずれかでなければなりません。そうでない場合、エラーが戻されます。
- 一方のオペランドが空のシーケンスの場合、比較の結果は空のシーケンスです。
- **is** 演算子を使用する比較は、比較される 2 つのノードが同一の ID を持っている場合に true となります。そうでない場合、比較は false です。
- << 演算子を使用する比較は、文書の順序において、左方オペランドのノードが右方オペランドのノードに先行する場合に true を戻します。そうでない場合、比較は false を戻します。

- >> 演算子を使用する比較は、文書の順序において、左方オペランドのノードが右方オペランドのノードの後に続く場合に true を戻します。そうでない場合、比較は false を戻します。

例

- 以下の比較は、左方オペランドと右方オペランドの両方が全く同じ単一ノードであると評価された場合にのみ true となります。

  ```
  /books/book[isbn="1558604820"] is /books/book[call="QA76.9 C3845"]
  ```

- 以下の比較は、構成されたノードがそれぞれ独自の ID を持っているため、 false となります。

  ```
  <a>5</a> is <a>5</a>
  ```

- 以下の比較は、文書の順序において、左方オペランドによって識別されるノードが、右方オペランドによって識別されるノードより前に存在する場合にのみ true となります。

  ```
  /transactions/purchase[parcel="28-451"] << /transactions/sale[parcel="33-870"]
  ```

論理式

論理式は、演算子 **and** および **or** を使用して、ブール値 (true または false) を計算します。

以下の表では、これらの演算子について説明し、演算子優先順位の高いものから低いものの順にリストしています。

表 27. XQuery における論理式演算子

演算子	目的
および	両方の式が true の場合に true を戻す。
または	一方または両方の式が true の場合に true を戻す。

論理式の結果は、ブール値 (true または false) またはエラーのいずれかです。論理式の評価時に、各オペランドの有効ブール値 (EBV) が決定されます。続いてオペランドの EBV に演算子が適用され、結果がブール値かエラーのいずれかになります。オペランドの EBV がエラーの場合、論理式の結果がエラーになる可能性があります。以下の表に、オペランドの EBV に基づいて論理式から戻される結果を示します。

表 28. オペランドの EBV に基づく論理式の結果

オペランド 1 の EBV	演算子	オペランド 2 の EBV	結果
true	および	true	true
true	および	false	false
false	および	true	false
false	および	false	false
true	および	エラー	エラー
エラー	および	true	エラー
false	および	エラー	false またはエラー

表 28. オペランドの EBV に基づく論理式の結果 (続き)

オペランド 1 の EBV	演算子	オペランド 2 の EBV	結果
エラー	および	false	false またはエラー
エラー	および	エラー	エラー
true	または	true	true
false	または	false	false
true	または	false	true
false	または	true	true
true	または	エラー	true またはエラー
エラー	または	true	true またはエラー
false	または	エラー	エラー
エラー	または	false	エラー
エラー	または	エラー	エラー

ヒント: 論理式に加えて、XQuery では、パラメーターとして一般的なシーケンスを取り、ブール値を戻す fn:not という名前の関数が提供されています。

例

- 以下の式は true を戻します。

```
1 eq 1 and 2 eq 2
1 eq 1 or 2 eq 3
```

- 以下の式は、false またはエラーのいずれかを戻す可能性があります。

```
1 eq 2 and 3 idiv 0 = 1
```

- 以下の式は、true またはエラーのいずれかを戻す可能性があります。

```
1 eq 1 or 3 idiv 0 = 1
```

- 以下の式はエラーを戻します。

```
1 eq 1 and 3 idiv 0 = 1
```

コンストラクター

コンストラクターは、照会内に XML 構造を作成します。XQuery は、エレメント・ノード、属性ノード、文書ノード、テキスト・ノード、処理命令ノード、およびコメント・ノードを作成するためのコンストラクターを提供します。XQuery には、直接コンストラクターと計算コンストラクターの 2 種類のコンストラクターがあります。

直接コンストラクター は、XML に類似した表記を使用して照会内に XML 構造を作成します。XQuery には、エレメント・ノード (属性ノード、テキスト・ノード、およびネストされたエレメント・ノードが考えられる)、処理命令ノード、およびコメント・ノードを作成するための直接コンストラクターがあります。例えば、以下のコンストラクターでは、属性といくつかのネストされたエレメントを含む book エレメントを作成します。

```
<book isbn="isbn-0060229357">
  <title>Harold and the Purple Crayon</title>
  <author>
```

```
    <first>Crockett</first>
    <last>Johnson</last>
  </author>
</book>
```

計算コンストラクター では、括弧で囲まれた式に基づいた表記を使用して、照会内に XML 構造を作成します。計算コンストラクターは、作成されるノードのタイプを識別するキーワードで開始し、その後にノード名 (適用される場合)、ノードのコンテンツを計算する括弧で囲まれた式が続きます。XQuery には、エレメント・ノード、属性ノード、文書ノード、テキスト・ノード、処理命令ノード、およびコメント・ノードを作成するための計算コンストラクターがあります。例えば、以下の照会には、前述の直接コンストラクターと同じ結果を生成する計算コンストラクターが含まれます。

```
element book {
  attribute isbn {"isbn-0060229357" },
  element title { "Harold and the Purple Crayon"},
  element author {
    element first { "Crockett" },
    element last {"Johnson" }
  }
}
```

コンストラクターの括弧で囲まれた式

括弧で囲まれた式は、コンストラクターで使用され、エレメント・コンテンツおよび属性コンテンツに計算値を提供します。これらの式は、コンストラクターが処理されるときに評価され、その値で置換されます。括弧で囲まれた式は中括弧 ({}) で囲まれ、リテラル・テキストと区別されます。

括弧で囲まれた式は、以下のコンストラクターで使用して計算値を生成できます。

- 直接エレメント・コンストラクター:
  - 直接エレメント・コンストラクターの開始タグの属性値には、括弧で囲まれた式を使用できます。
  - 直接エレメント・コンストラクターのコンテンツには、構成されたノードのコンテンツおよび属性の両方を計算する括弧で囲まれた式を使用できます。
- 計算コンストラクター:
  - 括弧で囲まれた式は、ノードのコンテンツを生成するのに使用できます。

例えば、以下の直接エレメント・コンストラクターには、括弧で囲まれた式が使用されています。

```
<example>
  <p> Here is a query. </p>
  <eg> $b/title </eg>
  <p> Here is the result of the query. </p>
  <eg>{ $b/title }</eg>
</example>
```

このコンストラクターが評価されるときは、以下の結果が生成されると考えられます (読みやすくするため、この例では空白文字を追加しています)。

```
<example>
  <p> Here is a query. </p>
  <eg> $b/title </eg>
  <p> Here is the result of the query. </p>
  <eg><title>Harold and the Purple Crayon</title></eg>
</example>
```

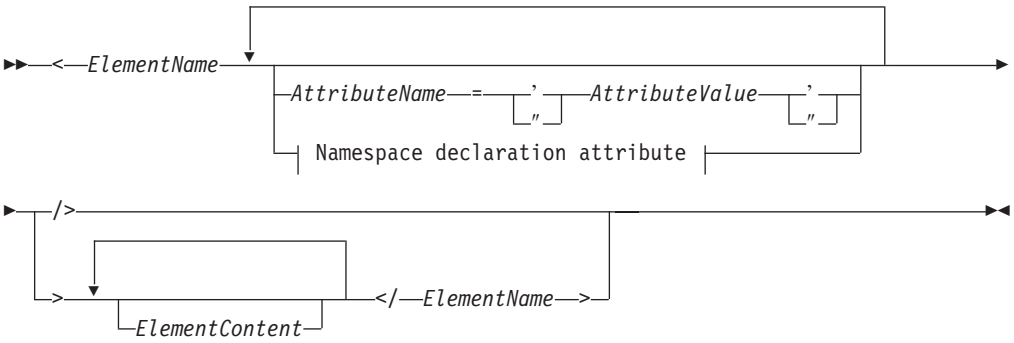
ヒント: エレメントまたは属性のコンテンツ内に通常文字として中括弧を使用するには、同じ中括弧を 2 つ使用するか、文字参照を使用することができます。例えば、1 組の {{ を使用して文字 { を表すことができます。同様に、1 組の }} を使用して文字 } を表すことができます。または、文字参照 { および } を使用して、中括弧文字を表すこともできます。1 個の左中括弧 ({) は、括弧で囲まれた式の区切り文字の開始点として解釈されます。左中括弧と一致しない 1 個の右中括弧 (}) はエラーになります。

直接エレメント・コンストラクター

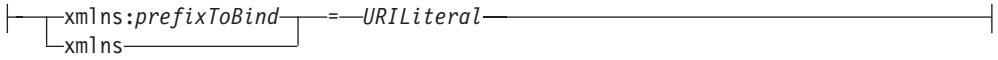
直接エレメント・コンストラクターは、XML に類似した表記を使用してエレメント・ノードを作成します。構成されたノードは、単純エレメントでも、属性、テキスト内容、およびネストされたエレメントを含む複合エレメントでも構いません。

直接エレメント・コンストラクターが作成する結果は、独自ノード ID を持つ新しいエレメント・ノードです。新規エレメント・ノードのすべての属性ノードおよび派生ノードも、それぞれ独自の ID を持つ新規ノードです。

構文



Namespace declaration attribute:



ElementName

構成するエレメントの名前を示す QName。最後のタグの *ElementName* に使用される名前は、対応する開始タグで使用される名前と、接頭部も含めて (含まれない場合も) 完全に一致する必要があります。*ElementName* にネーム・スペース接頭部が含まれている場合、その接頭部は、静的に既知のネーム・スペースを使用することで、ネーム・スペース URI に解決されます。*ElementName* にネーム・スペース接頭部がない場合、その名前はデフォルトのエレメント/タイプの

ネーム・スペースで暗黙的に修飾されます。 *ElementName* の評価により拡張された QName は、構成されたエレメント・ノードの名前になります。

AttributeName
構成する属性の名前を示す QName。 *AttributeName* にネーム・スペース接頭部が含まれている場合、その接頭部は静的に既知のネーム・スペースを使用することで、ネーム・スペース URI に解決されます。*AttributeName* にネーム・スペース接頭部がない場合、その属性はネーム・スペースに属しません。
AttributeName の評価により拡張された QName は、構成された属性ノードの名前になります。各属性の拡張された QName は固有である必要があります。固有でない場合、式の結果がエラーになります。

直接エレメント・コンストラクターの各属性は、独自のノード ID を持つ新規の属性ノードを作成します。新規属性ノードの親は、構成されたエレメント・ノードです。新規属性ノードは、xdt:untypedAtomic のタイプ・アノテーションが指定されます。

AttributeValue
属性の値を指定する文字ストリング。属性値には、エレメント・コンストラクターが処理されるときに評価され、その値で置換される括弧で囲んだ式 (中括弧で囲んだ式) を使用できます。事前定義されたエンティティー参照や文字参照も有効で、示される文字で置換されます。以下の表は、*AttributeValue* 内で有効な特殊文字をリストしていますが、これらの文字は、二重文字またはエンティティー参照で示される必要があります。

表 29. 属性値の特殊文字の表記

文字	属性値に必要な表記
{	2 個の左中括弧 ({{)
}	2 個の右中括弧 (}})
<	&lt;
&	&amp;
″	" または 2 個の二重引用符 (″″)
’	' または 2 個の単一引用符 (’’)

xmlns
ネーム・スペース宣言属性で始まる単語。QName で接頭部として指定されると、**xmlns** は、*prefixToBind* の値が *URILiteral* で指定される URI にバインドされることを示します。このネーム・スペースのバインディングは、このコンストラクター式、およびその式内でネストされているすべての式の静的に既知のネーム・スペースに追加されます (バインディングが、ネストされたネーム・スペース宣言の属性でオーバーライドされる場合を除く)。以下の例では、ネーム・スペース宣言の属性 `xmlns:metric = "http://example.org/metric/units"` が接頭部 `metric` をネーム・スペース `http://example.org/metric/units` にバインドします。

接頭部なしの完全 QName として指定される場合、**xmlns** は、デフォルトのエレメント/タイプのネーム・スペースが *URILiteral* の値に設定されることを示します。このデフォルトのエレメント/タイプのネーム・スペースは、このコンストラクター式、およびそのコンストラクター式内でネストされているすべての式に有効です (宣言が、ネストされたネーム・スペース宣言の属性でオーバーライ

ドされる場合を除く)。以下の例では、ネーム・スペース宣言属性 xmlns = "http://example.org/animals" はデフォルトのエレメント/タイプのネーム・スペースを http://example.org/animals に設定します。

prefixToBind
URILiteral に指定されている URI にバインドされる接頭部。 *prefixToBind* の値を、xml または xmlns にすることはできません。このいずれの値を指定しても、エラーになります。

URILiteral
URI を示すストリング・リテラル (単一引用符または二重引用符で囲まれたゼロ個以上の文字のシーケンス)。ストリング・リテラルの値は有効な URI である必要があります。*URILiteral* の値は、ネーム・スペース宣言の属性がデフォルトのエレメント/タイプのネーム・スペースを設定するのに使用されている場合にのみ、ゼロ長ストリングにできます。それ以外の場合、*URILiteral* にゼロ長ストリングを指定するとエラーになります。

ElementContent
直接エレメント・コンストラクターのコンテンツ。コンテンツは、コンストラクターの開始タグと終了タグの間にあるすべてのものから構成されます。エレメント・コンストラクター内で境界スペースがどのように処理されるかは、プロローグの境界スペース宣言によって制御されます。結果のコンテンツ・シーケンスは、コンテンツ・エンティティーを連結したものです。括弧で囲まれた式の結果のテキストを含む、結果の隣接するテキスト文字は、1 個のテキスト・ノードにマージされます。結果の属性ノードは、結果のコンテンツ・シーケンスにおいて他のどのコンテンツよりも前になります。

ElementContent は、以下のコンテンツのいずれかで構成されます。

- **テキスト文字**。テキスト文字はテキスト・ノードを作成し、隣接するテキスト・ノードは 1 個のテキスト・ノードにマージされます。文字シーケンス内の行の末尾は、XML 1.0 に指定されている行末処理規則に従って正規化されます。以下の表は、*ElementContent* で有効な特殊文字のリストです。これらの文字は二重文字またはエンティティー参照で示される必要があります。

表 30. エレメント・コンテンツの特殊文字の表記

文字	エレメント・コンテンツに必要な表記
{	2 個の左中括弧 ({{)
}	2 個の右中括弧 (}})
<	&lt;
&	&amp;

- **ネストされた直接コンストラクター**
- **CDataSections**。CDataSections は、構文 <![CDATA[*contents*]]> を使用して指定されます。この *contents* は、一連の文字で構成されます。*contents* に指定される文字には、< や & などの特殊文字が含まれ、区切り文字としてではなく、リテラル文字として取り扱われます。シーケンス]]> は、CDataSection を終了するため、*contents* 内では使用できません。
- **文字参照および定義済みのエンティティー参照**。処理中、定義済みのエンティティー参照および文字参照は、参照されたストリングに展開されます。

- **括弧で囲まれた式**。括弧で囲まれた式とは、中括弧で囲まれた XQuery 式のことです。例えば、{5 + 7} などは括弧で囲まれた式です。括弧で囲まれた式の値は、ノードと原子値の任意のシーケンスで構いません。括弧で囲まれた式は、直接エレメント・コンストラクターのコンテンツ内で使用し、構成されたノードのコンテンツおよび属性の両方を計算することができます。括弧で囲まれた式から戻されたそれぞれのノードごとに、元のタイプのアノテーションを保持するノードとそのすべての子孫から新規コピーが作成されます。*ElementContent* から戻された属性ノードは、結果のコンテンツ・シーケンスの最初の位置にきます。これらの属性ノードは、構成されたエレメントの属性になります。*ElementContent* から戻されたエレメント・ノード、コンテンツ・ノード、または処理命令ノードは、新しく構成されたノードの子になります。*ElementContent* から戻された原子値は、ストリングに変換されてテキスト・ノードに保管されます。このノードは、構成されたノードの子になります。隣接するテキスト・ノードは、1 つのテキスト・ノードにマージされます。

例

- 以下の直接エレメント・コンストラクターは、ブック・エレメントを作成します。ブック・エレメントは、属性ノード、複数のネストされたエレメント・ノード、および複数のテキスト・ノードを含む複合コンテンツを含みます。

```
<book isbn="isbn-0060229357">
    <title>Harold and the Purple Crayon</title>
    <author>
        <first>Crockett</first>
        <last>Johnson</last>
    </author>
</book>
```

- 以下の例では、エレメント・コンテンツが直接エレメント・コンストラクターでどのように処理されるかを示します。
  - 以下の式では、1 つの子、値「1」を含むテキスト・ノードを持つエレメント・ノードを構成します。

    ```
    <a>{1}</a>
    ```

  - 以下の式では、1 つの子、値「1 2 3」を含むテキスト・ノードを持つエレメント・ノードを構成します。

    ```
    <a>{1, 2, 3}</a>
    ```

  - 以下の式では、1 つの子、値「123」を含むテキスト・ノードを持つエレメント・ノードを構成します。

    ```
    <c>{1}{2}{3}</c>
    ```

  - 以下の式では、1 つの子、値「1 2 3」を含むテキスト・ノードを持つエレメント・ノードを構成します。

    ```
    <b>{1, "2", "3"}</b>
    ```

  - 以下の式では、1 つの子、値「I saw 8 cats」を含むテキスト・ノードを持つエレメント・ノードを構成します。:

    ```
    <fact>I saw 8 cats.</fact>
    ```

  - 以下の式では、1 つの子、値「I saw 8 cats」を含むテキスト・ノードを持つエレメント・ノードを構成します。

    ```
    <fact>I saw {5 + 3} cats.</fact>
    ```

– 以下の式では、「I saw」を含むテキスト・ノード、howmany という名前の子エレメント・ノード、および「cats」を含むテキスト・ノードという 3 つの子を構成します。 子エレメント・ノードは、値「8」を含むテキスト・ノードである単一の子を持ちます。

```
<fact>I saw <howmany>{5 + 3}</howmany> cats.</fact>
```

ネーム・スペース宣言属性

ネーム・スペース宣言属性は、直接エレメント・コンストラクターの開始タグで指定します。ネーム・スペース宣言属性は、以下の 2 つの目的で使用されます。ネーム・スペース接頭部を URI にバインドするため、および構成されるエレメント・ノードおよびその属性と子孫のデフォルト・エレメント/タイプ・ネーム・スペースを設定するためです。

構文的には、ネーム・スペース宣言属性の形式は直接エレメント・コンストラクター内の属性の形式と同じです。属性を名前と値で指定します。属性名は定数 QName です。属性値は、有効な URI を表すストリング・リテラルです。

ネーム・スペース宣言属性により、属性ノードは作成されません。

重要: 1 つの直接エレメント・コンストラクター内の各ネーム・スペース宣言属性の名前はユニークにする必要があります。そうでないと式の結果はエラーになります。

ネーム・スペース接頭部の URI へのバインディング

QName が接頭部 xmlns で始まり、後にローカル名が続く場合、宣言を使用してネーム・スペース接頭部 (ローカル名として指定) を URI (属性値として指定) にバインドします。例えば、ネーム・スペース宣言属性 xmlns:metric = "http://example.org/metric/units" は、接頭部 metric をネーム・スペース http://example.org/metric/units にバインドします。

ネーム・スペース宣言属性が処理されると、接頭部および URI がコンストラクター式の静的に既知のネーム・スペースに追加され、指定された接頭部の既存のバインディングが、新規のバインディングにより指定変更されます。接頭部および URI は、ネーム・スペース・バインディングとして、構成されるエレメントの範囲内のネーム・スペースにも追加されます。

例えば、以下のエレメント・コンストラクターでは、ネーム・スペース宣言属性が使用され、ネーム・スペース接頭部の metric と english がバインドされます。

```
<box xmlns:metric = "http://example.org/metric/units"
xmlns:english = "http://example.org/english/units">
  <height> <metric:meters>3</metric:meters> </height>
  <width> <english:feet>6</english:feet> </width>
  <depth> <english:inches>18</english:inches> </depth>
</box>
```

デフォルト・エレメント/タイプ・ネーム・スペースの設定

QName が xmlns で接頭部がない場合、宣言が使用されてデフォルト・エレメント/タイプ・ネーム・スペースが設定されます。例えば、ネーム・スペース宣言属性 xmlns = "http://example.org/animals" は、デフォルト・エレメント/タイプ・ネーム・スペースを http://example.org/animals に設定します。

ネーム・スペース宣言属性が処理されると、属性の値がネーム・スペース URI として解釈されます。この URI は、コンストラクター式のデフォルト・エレメント/タイプ・ネーム・スペースを指定し、既存のデフォルトが新規指定により指定変更されます。URI は、構成されるエレメントの範囲内のネーム・スペースにも (接頭部なしで) 追加され、接頭部のない既存のネーム・スペース・バインディングが、新規指定により指定変更されます。ネーム・スペース URI がゼロ長ストリングの場合、コンストラクター式のデフォルト・エレメント/タイプ・ネーム・スペースは、「none」に設定されます。

例えば、以下の直接エレメント・コンストラクターでは、ネーム・スペース宣言属性により、デフォルトのエレメント/タイプのネーム・スペースが `http://example.org/animals` に設定されます。

```
<cat xmlns = "http://example.org/animals">
  <breed>Persian</breed>
</cat>
```

直接エレメント・コンストラクター内の境界空白

直接エレメント・コンストラクター内において、*境界空白* は、内容の開始または終了のいずれか、直接コンストラクター、または括弧で囲んだ式によって両端で区切られる、連続する空白文字のシーケンスです。例えば、境界空白は、コンストラクターの内容において、直接コンストラクターの終了タグをネストされたエレメントの開始タグと分離するために使用されます。

以下の図は、境界空白を強調表示させた直接エレメント・コンストラクターの例です。

```
<product> ↵
  <description> { "  enclosed expression  " } </description> ↵
</product>
```

この例の境界空白には、以下の文字があります。 `product` エレメントおよび `description` エレメントの開始タグの間にある 1 つの改行文字と 4 つのスペース文字。`description` エレメントの開始タグと括弧で囲んだ式との間にある 4 つのスペース文字。括弧で囲んだ式と `description` エレメントの終了タグとの間にある 4 つのスペース文字。および `description` エレメントの終了タグの後にある 1 つの改行文字。

境界空白には、以下の種類の空白は含まれません。

- 括弧で囲んだ式によって生成される空白
- 文字参照 (例えば、`&#x20;`) または CDataSection によって生成される文字
- 文字参照または CDataSection に隣接する空白文字

境界スペース・ポリシーは、エレメント・コンストラクターによって境界空白が保持されるかどうかを制御します。このポリシーは、照会プロローグの境界スペース宣言によって指定されます。境界スペース宣言で **strip** が指定されている場合、境界空白は廃棄されます。境界スペース宣言で **preserve** が指定されている場合、境界

空白は保持されます。境界スペース宣言が指定されていない場合、デフォルトの動作として、エレメント構成時に境界空白が削除されます。

例

- 以下の例において、構成される cat エレメント・ノードは、 breed および color という 2 つの子エレメント・ノードを持ちます。

```
<cat>
  <breed>{$b}</breed>
  <color>{$c}</color>
</cat>
```

  境界スペース・ポリシーはデフォルトで **strip** であるため、子エレメントの周囲の空白は、エレメント・コンストラクターによって削除されます。

- 以下の例において、境界スペース・ポリシーは **strip** です。 この例は、<a>abc と同じ意味になります。

```
declare boundary-space strip;
<a> {"abc"} </a>
```

- ただし、以下の例においては、境界スペース・ポリシーは **preserve** です。この例は、<a>abc と同じ意味になります。

```
declare boundary-space preserve;
<a> {"abc"} </a>
```

  境界スペース・ポリシーが **preserve** であるため、括弧で囲んだ式の前後にあるスペースは、エレメント・コンストラクターによって保持されます。

- 以下の例において、z の周囲にある空白は、境界空白ではありません。この空白は常に保持され、この例は <a> z abc と同じ意味になります。

```
<a> z {"abc"}</a>
```

- 以下の例において、文字参照によって生成される空白文字および隣接する空白文字は、境界スペース・ポリシーに関係なく保持されます。この例は、<a>abc と同じ意味になります。

```
<a>     &#x20;{"abc"}</a>
```

- 以下の例においては、括弧で囲んだ式の中の空白は、境界スペース・ポリシーに関係なく保持されます。これは、括弧で囲んだ式によって生成される空白が境界空白と見なされることはないためです。この例は、<a> と同じ意味になります。

```
<a>{" "}</a>
```

  括弧で囲んだ式の中の 2 つのスペースは、エレメント・コンストラクターによって保持され、結果の中の開始タグと終了タグの間に表示されます。

構成されるエレメントの範囲内のネーム・スペース

構成されるエレメント・ノードは、ネーム・スペース・バインディングのセットで構成される範囲内のネーム・スペース・プロパティーを持ちます。各ネーム・スペース・バインディングは、ネーム・スペース接頭部を URI に関連付けます。ネーム・スペース・バインディングは、エレメントの有効範囲内の QName を解釈するために使用可能なネーム・スペース接頭部のセットを定義します。

重要: このトピックについて理解するには、以下の概念の差異を理解する必要があります。

静的に既知のネーム・スペース

静的に既知のネーム・スペース は、式のプロパティーです。このプロパティーは、XQuery が、式の処理中にネーム・スペース接頭部を解決するために使用するネーム・スペース・バインディングのセットを示します。これらのバインディングは、照会結果の一部ではありません。

範囲内のネーム・スペース

範囲内のネーム・スペース は、エレメント・ノードのプロパティーです。このプロパティーは、エレメントおよびその内容が処理されるときに、XQuery の外部のアプリケーションが使用可能なネーム・スペース・バインディングのセットを示します。これらのバインディングは、外部のアプリケーションで使用可能になるよう、照会結果の一部としてシリアライズされます。

構成されるエレメントの範囲内のネーム・スペースには、以下の方法で作成されるすべてのネーム・スペース・バインディングが組み込まれます。

- **ネーム・スペース宣言属性により明示的に作成。**ネーム・スペース・バインディングが、以下のコンストラクターで宣言される各ネーム・スペース宣言属性に対して作成されます。
  - 現行エレメント・コンストラクター。
  - 括弧で囲まれた直接エレメント・コンストラクター (ネーム・スペース宣言属性が、現行エレメント・コンストラクターまたは中間コンストラクターによって指定変更されない場合)。
- **システムにより自動的に作成。**以下の状況でネーム・スペース・バインディングが作成されます。
  - 接頭部 `xml` をネーム・スペース URI `http://www.w3.org/XML/1998/namespace` にバインドするため。このバインディングは、構成されるすべてのエレメントに対して作成されます。
  - 構成されるエレメントの名前またはその属性の名前の中で使用される各ネーム・スペースのため (エレメントの範囲内のネーム・スペース内にまだネーム・スペース・バインディングが存在しない場合)。ノードの名前に接頭部が組み込まれる場合、ネーム・スペース・バインディングでその接頭部が使用されます。名前に接頭部がない場合、空の接頭部に対してバインディングが作成されます。同じ接頭部の 2 つの異なるバインディングを必要とする競合が発生した場合、ノード名に使用される接頭部が任意の接頭部に変更され、その任意の接頭部に対してネーム・スペース・バインディングが作成されます。

    要確認: QName に使用される接頭部は、有効な URI に解決される必要があります。解決されない場合、その接頭部に対するバインディングをエレメントの範囲内のネーム・スペースに追加できません。QName が解決できない場合、式の結果はエラーとなります。

例

以下の照会は、ネーム・スペース宣言を含むプロローグ、および直接エレメント・コンストラクターを含む本体を組み込みます。

```
declare namespace p="http://example.com/ns/p";
declare namespace q="http://example.com/ns/q";
declare namespace f="http://example.com/ns/f";
<p:newElement q:b="{f:func(2)}" xmlns:r="http://example.com/ns/r"/>
```

プロローグ内のネーム・スペース宣言により、式の静的に既知のネーム・スペースに、ネーム・スペース・バインディングが追加されます。ただし、構成されるエレメントの範囲内のネーム・スペースにネーム・スペース・バインディングが追加されるのは、コンストラクター内の QName がこれらのネーム・スペースを使用する場合のみです。このため、 `p:newElement` の範囲内のネーム・スペースは、以下のネーム・スペース・バインディングで構成されます。

- `p = "http://example.com/ns/p"` - このネーム・スペース・バインディングは、接頭部 `p` が QName `p:newElement` 内にあるため、範囲内のネーム・スペースに追加されます。
- `q = "http://example.com/ns/q"` - このネーム・スペース・バインディングは、接頭部 `q` が属性 QName `q:b` 内にあるため、範囲内のネーム・スペースに追加されます。
- `r = "http://example.com/ns/r"` - このネーム・スペース・バインディングは、ネーム・スペース宣言属性で定義されているため、範囲内のネーム・スペースに追加されます。
- `xml = "http://www.w3.org/XML/1998/namespace"` - このネーム・スペース・バインディングは、構成されるすべてのエレメント・ノードに対して定義されるため、範囲内のネーム・スペースに追加されます。

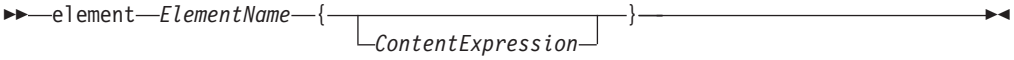
ネーム・スペース `f="http://example.com/ns/f"` に対するバインディングは、範囲内のネーム・スペースに追加されないことに注意してください。これは、(`q:b` という名前の属性の内容に `f:func(2)` が含まれていても) エレメント・コンストラクターが、接頭部 `f` を使用するエレメントまたは属性名を組み込んでいないためです。このため、このネーム・スペース・バインディングは、静的に既知のネーム・スペースに存在し、照会の処理時に使用可能であっても、照会結果には含まれません。

計算エレメント・コンストラクター

計算エレメント・コンストラクターは、括弧で囲まれた式に基づいてノードのコンテンツを計算するエレメント・ノードを作成します。

計算エレメント・コンストラクターが作成する結果は、独自ノード ID を持つ新しいエレメント・ノードになります。既存のノードのコピーであっても、新規エレメント・ノードのすべての属性ノードおよび派生ノードも、独自の ID を持つ新しいノードです。

構文



element
エレメント・ノードが構成されることを示すキーワード。

ElementName
構成するエレメントの QName。*ElementName* にネーム・スペース接頭部が含ま

れている場合、その接頭部は、静的に既知のネーム・スペースを使用することで、ネーム・スペース URI に解決されます。*ElementName* にネーム・スペース接頭部がない場合、その名前はデフォルトのエレメント/タイプのネーム・スペースで暗黙的に修飾されます。 *ElementName* の評価により拡張された QName は、構成されたエレメント・ノードの名前になります。

ContentExpression
構成されたエレメント・ノードのコンテンツを生成する式。*ContentExpression* の値は、任意のノードおよび原子値のシーケンスを使用できます。*ContentExpression* は、構成されたノードのコンテンツおよび属性の両方を計算するのに使用できます。*ContentExpression* によって戻される各ノードの場合、ノードおよびその子孫のすべてから新規コピーが作成され、その元のタイプのアノテーションを保持します。*ContentExpression* によって戻されるすべての属性ノードは、ノード・シーケンスの最初の位置 (他のノードの前) になければなりません。これらの属性ノードは、構成されたエレメントの属性になります。*ContentExpression* によって戻されるすべてのエレメント・ノード、コンテンツ・ノード、または処理命令ノードは、新しく構成されたノードの子になります。*ContentExpression* によって戻されるすべての原子値はストリングに変換され、構成されたノードの子になるテキスト・ノードに保管されます。隣接するテキスト・ノードは、1 つのテキスト・ノードにマージされます。

例

以下の式では、計算エレメント・コンストラクターは、既存のエレメントを修正したコピーを作成します。変数 `$e` は、数値コンテンツを持つエレメントにバインドされるものとします。このコンストラクターは、`$e` と同じ属性を持ち、`$e` コンテンツの 2 倍の数値コンテンツを持つ新規エレメント `length` を作成します。

```
element length {$e/@*, 2 * fn:data($e)}
```

この例では、変数 `$e` が式 `let $e := <length units="inches">{5}</length>` にバインドされる場合、上記のサンプル式の結果はエレメント `<length units="inches">10</length>` になります。

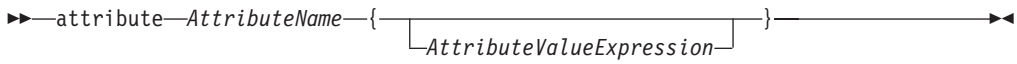
計算属性コンストラクター

計算属性コンストラクターは、括弧で囲まれた式に基づいて属性値が計算される属性ノードを作成します。

計算属性コンストラクターが作成する結果は、独自ノード ID を持つ新しい属性ノードになります。

注: 属性ノードを直接構成するには、直接エレメント・コンストラクターでその属性を宣言します。

構文



attribute
属性ノードが構成されることを示すキーワード。

AttributeName
構成する属性の QName。*AttributeName* にネーム・スペース接頭部が含まれている場合、その接頭部は静的に既知のネーム・スペースを使用することで、ネーム・スペース URI に解決されます。*AttributeName* にネーム・スペース接頭部がない場合、その属性はネーム・スペースに属しません。*AttributeName* の評価により拡張された QName は、構成された属性ノードの名前になります。エレメントの各属性の拡張された QName は固有である必要があり、固有でない場合は式の結果がエラーになります。

AttributeValueExpression
属性ノードの値を生成する式。処理中、*AttributeValueExpression* の結果に原子化が適用され、結果のシーケンスの各原子値はストリングにキャストされます。キャストの結果の個々のストリングは、スペース文字を間に挿入して連結されます。連結されたストリングは、構成された属性ノードの値となります。

例

以下の計算属性コンストラクターは、属性名 size、値「7」の属性を構成します。

```
attribute size {4 + 3}
```

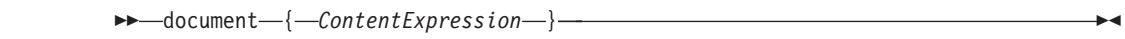
文書ノード・コンストラクター

すべての文書ノード・コンストラクターは、計算コンストラクターです。文書ノード・コンストラクターは、ノードのコンテンツが括弧で囲まれた式に基づいて計算される文書ノードを作成します。文書ノード・コンストラクターは、照会の結果が完全な文書である場合に有用です。

文書ノード・コンストラクターが作成する結果は、独自ノード ID を持つ新しい文書ノードです。

重要: 構成された文書ノードでは、妥当性検査は実行されません。XQuery 文書ノード・コンストラクターは、XML 文書の構造を管理する XML 1.0 規則を適用しません。例えば、文書ノードは、エレメント・ノードである正確に 1 個の子を持つ必要はありません。

構文



document
文書ノードが構成されることを示すキーワード。

ContentExpression
構成された文書ノードのコンテンツを生成する式。*ContentExpression* の値は、属性ノードの場合を除いてノードおよび原子値のシーケンスで構いません。コンテンツ・シーケンスの属性ノードは、エラーになります。コンテンツ・シーケンスの文書ノードは、その子に置換されます。*ContentExpression* によって戻される各ノードの場合、ノードおよびその子孫のすべてから新規コピーが作成され、その元のタイプのアノテーションを保持します。コンテンツ式によって戻された

原子値は、ストリングに変換され、テキスト・ノードに保管されます。これは、構成された文書ノードの子になります。隣接するテキスト・ノードは、1 つのテキスト・ノードにマージされます。

例

以下の文書ノード・コンストラクターには、customer-list という名前のルート・エレメントを含む XML 文書を戻すコンテンツ式が含まれます。

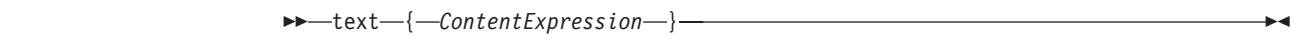
```
document
{
<customer-list>
   {db2-fn:xmlcolumn('MYSCHEMA.CUSTOMER.INFO')/ns1:customerinfo/name}
</customer-list>
}
```

テキスト・ノード・コンストラクター

テキスト・ノード・コンストラクターは、すべて計算コンストラクターです。テキスト・ノード・コンストラクターはテキスト・ノードを作成し、そのノードの内容は括弧で囲んだ式に基づいて計算されます。

テキスト・ノード・コンストラクターが作成する結果は、独自ノード ID を持つ新しいテキスト・ノードです。

構文



text
テキスト・ノードが構成されることを示すキーワード。

ContentExpression
構成されるテキスト・ノードの内容を生成する式。処理時に、*ContentExpression* の結果に原子化が適用され、結果のシーケンス内の各原子値がストリングにキャストされます。キャストの結果の個々のストリングは、スペース文字を間に挿入して連結されます。連結されたストリングは、構成されるテキスト・ノードの内容になります。原子化の結果が空のシーケンスの場合、テキスト・ノードは構成されません。

注: テキスト・ノード・コンストラクターは、ゼロ長ストリングを含むテキスト・ノードを構成するために使用できます。ただし、このテキスト・ノードが、構成されたエレメントまたはドキュメント・ノードの内容に使用される場合、テキスト・ノードは削除されるか、別のテキスト・ノードにマージされます。

例

以下のコンストラクターは、ストリング ″Hello″ を含むテキスト・ノードを作成します。

```
text {"Hello"}
```

処理命令コンストラクター

処理命令コンストラクターは、処理命令ノードを作成します。XQuery では、処理命令ノードを作成するための直接コンストラクターと計算コンストラクターの両方が提供されています。

構成されるノードには、以下のノード・プロパティーがあります。

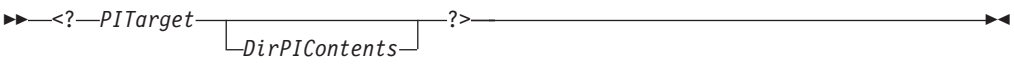
target プロパティー
処理命令の送信先であるアプリケーションを示します。

content プロパティー
処理命令の内容を指定します。

直接処理命令コンストラクター

直接処理命令コンストラクターは、XML に類似した表記を使用して、処理命令ノードを作成します。

構文



PITarget
処理命令の送信先となる処理アプリケーションの名前を示す NCName。処理命令の PI ターゲットは、大文字、小文字のいずれの組み合わせでも、「XML」という文字で構成することはできません。

DirPIContents
処理命令のコンテンツを指定する一連の文字。処理命令のコンテンツには、ストリング ?>を含めることはできません。

例

以下のコンストラクターは、処理命令ノードを作成します。

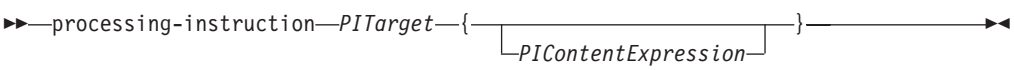
```
<?format role="output" ?>
```

計算処理命令コンストラクター

計算処理命令コンストラクターは、括弧で囲まれた式に基づいてノードのコンテンツが計算される処理命令ノードを作成します。

計算処理命令コンストラクターが作成する結果は、独自のノード ID を持つ新しい処理命令ノードになります。

構文



processing-instruction
処理命令ノードが構成されることを示すキーワード。

PITarget
処理命令の送信先となる処理アプリケーションの名前を示す NCName。この名前は、*Namespaces in XML* で指定されている NCName の形式に適合する必要があります。

PIContentExpression
処理命令ノードのコンテンツを生成する式。処理中、*PIContentExpression* の結果に原子化が適用され、結果のシーケンスの各原子値はストリングにキャストされます。キャストの結果の個々のストリングは、スペース文字を間に挿入して連結されます。先頭の空白文字は削除され、連結されたストリングが処理命令ノードのコンテンツになります。原子化の結果が空のシーケンスである場合、シーケンスはゼロ長ストリングに置換されます。コンテンツ・シーケンスには、ストリング「?>」を含めることはできません。

例

以下の計算コンストラクターは、処理命令 <?audio-output beep?> を作成します。

```
processing-instruction audio-output {"beep"}
```

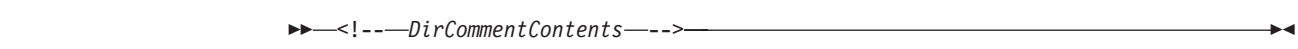
コメント・コンストラクター

コメント・コンストラクターは、コメント・ノードを作成します。XQuery は、コメント・ノードを作成するために、直接コンストラクターと計算コンストラクターの両方を提供します。

直接コメント・コンストラクター

直接コメント・コンストラクターは、XML に類似した表記を使用してコメント・ノードを作成します。

構文



DirCommentContents
コメントのコンテンツを指定する一連の文字。コメントのコンテンツは、ハイフンを 2 個連続して使用したり、ハイフンで終了させたりすることはできません。

例

以下のコンストラクターはコメント・ノードを作成します。

```
<!-- This is an XML comment. -->
```

計算コメント・コンストラクター

計算コメント・コンストラクターは、括弧で囲まれた式に基づいてノードのコンテンツが計算されるコメント・ノードを作成します。

計算コメント・コンストラクターが作成する結果は、独自のノード ID を持つ新しいコメント・ノードになります。

構文

comment
コメント・ノードが構成されることを示すキーワード。

CommentContents
コメントの内容を生成する式。処理中に、*CommentContents* の結果に原子化が適用され、原子化されたシーケンスの各原子値がストリングにキャストされます。キャストの結果の個々のストリングは、スペース文字を間に挿入して連結されます。原子化の結果が空のシーケンスである場合、シーケンスはゼロ長ストリングに置換されます。コンテンツ・シーケンスには、ハイフンを 2 個連続させて使用したり、ハイフンで終了させたりすることはできません。

例

以下の計算コンストラクターは、コメント `<!--Houston, we have a problem.-->` を作成します。

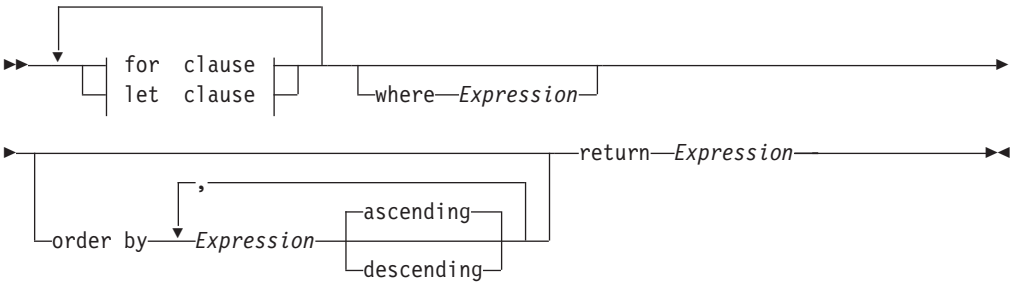
```
let $homebase := "Houston"
return comment {fn:concat($homebase, ", we have a problem.")}
```

FLWOR 式

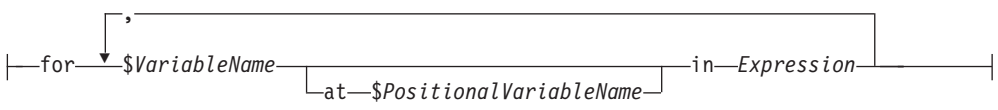
FLWOR 式は、シーケンスを繰り返し、変数を中間結果にバインドします。FLWOR 式は、2 つ以上の文書を結合して計算する、データを再構成する、および結果をソートするのに有用です。

FLWOR 式の構文

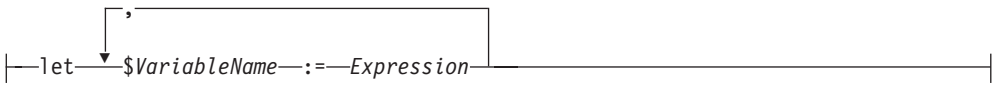
FLWOR 式は、**for** 節、**let** 節、 **where** 節、**order by** 節、および **return** 節から構成されます。一部の節はオプションです。



for clause:



let clause:



for
for 節を開始するためのキーワード。**for** 節は、*Expression* の結果を繰り返して、*Expression* で戻される各項目に *VariableName* をバインドします。

let **let** 節を開始するためのキーワード。**let** 節は、*VariableName* を *Expression* の結果全体にバインドします。

VariableName
Expression の結果にバインドする変数の名前。

PositionalVariableName
for 節の繰り返しごとにバインドされる項目の、入力ストリーム内の位置にバインドされるオプションの変数名。

Expression
任意の XQuery 式。式に最上位のコンマ演算子が含まれる場合、式を括弧で囲む必要があります。

where
where 節を開始するためのキーワード。**where** 節は、**for** 節および **let** 節で生成される変数バインディングのタプルをフィルタリングします。

order by
order by 節を開始するためのキーワード。**order by** 節は、**return** 節によって値が処理される順序を指定します。

ascending
順序付けのキーが昇順で処理されることを指定します。

descending
順序付けのキーが降順で処理されることを指定します。

return
return 節を開始するためのキーワード。 **return** 節の式は、**for** 節、**let** 節、**where** 節、および **order by** 節で生成されるバインド済み変数の各タプルについて 1 回評価されます。**return** 節に非更新式が含まれる場合、FLWOR 式は非更新式となります。 return 節のすべての評価結果は 1 つのシーケンスに連結され、それが FLWOR 式の結果となります。

return 節に更新式が含まれる場合、FLWOR 式は更新式となります。更新 FLWOR 式は、変換式の **modify** 節内に指定する必要があります。更新 FLWOR 式の結果は、更新のリストです。これが含まれる変換式は、更新を、この変換式の **modify** 節内の他の更新式によって戻される他の更新とマージした後に実行します。

for 節および let 節

FLWOR 式内の **for** 節または **let** 節は、1 つ以上の変数を、FLWOR 式の他の節で使用される値にバインドします。

for 節

for 節は、式の結果を反復処理し、シーケンス内の各項目に変数をバインドします。

最も単純なタイプの **for** 節には、1 つの変数および関連する式が含まれます。以下の例では、**for** 節には、$i という変数と、シーケンス (1, 2, 3) を構成する式が含まれます。

```
for $i in (1, 2, 3)
return <output>{$i}</output>
```

for 節の評価時に、3 つの変数バインディングが作成されます (シーケンス内の各項目に対して 1 つのバインディング):

```
$i = 1
$i = 2
$i = 3
```

例の中の **return** 節は、各バインディングに対して 1 回実行されます。この式は、以下の出力のような結果になります。

```
<output>1</output>
<output>2</output>
<output>3</output>
```

for 節には複数の変数を使用可能で、それぞれの変数が式の結果にバインドされます。以下の例では、 **for** 節に 2 つの変数 $a および $b、およびシーケンス 1 2 および 4 5 を構成する式が使用されています。

```
for $a in (1, 2), $b in (4, 5)
return <output>{$a, $b}</output>
```

for 節の評価時に、値の各組み合わせに対して変数バインディングのタプルが作成されます。この結果、変数バインディングの 4 つのタプルが作成されます。

```
($a = 1, $b = 4)
($a = 2, $b = 4)
($a = 1, $b = 5)
($a = 2, $b = 5)
```

例の中の **return** 節は、バインディングの各タプルに対して 1 回実行されます。この式は、以下の出力のような結果になります。

```
<output>1 4</output>
<output>2 4</output>
<output>1 5</output>
<output>2 5</output>
```

バインディング式が空のシーケンスに評価されると、**for** バインディングが生成されず、反復は実行されません。以下の例のバインディング・シーケンスは空シーケンスとして評価され、反復は実行されません。 **return** 節のノード・シーケンスは戻されません。

```
for $node in (<a test = "b" />, <a test = "c" />, <a test = "d" /> )[@test = "1"]
return
  <test>
    Sample return response
  </test>
```

for 節の定位置変数

for 節でバインドされる各変数は、同時にバインドされる、関連する定位置変数を持つことができます。定位置変数の名前には、キーワード **at** が先行します。1 つのシーケンス内の複数の項目について 1 つの変数が反復処理されるとき、定位置変数は、シーケンス内のこれらの項目の位置を表す整数について、1 から開始して反復処理されます。

以下の例では、**for** 節には、`$cat` という変数と、シーケンス `("Persian", "Calico", "Siamese")` を構成する式が含まれます。この節には、さらに定位置変数 `$i` が含まれ、属性コンストラクターでこれが参照されて、`order` 属性の値が計算されます。

```
for $cat at $i in ("Persian", "Calico", "Siamese")
return <cat order = "{$i}"> { $cat } </cat>
```

for 節の評価時に、変数バインディングの 3 つのタプルが作成され、各タプルに定位置変数のバインディングが含まれます。

```
($i = 1, $cat = "Persian")
($i = 2, $cat = "Calico")
($i = 3, $cat = "Siamese")
```

例の中の **return** 節は、バインディングの各タプルに対して 1 回実行されます。この式は、以下の出力のような結果になります。

```
<cat order = "1">Persian</cat>
<cat order = "2">Calico</cat>
<cat order = "3">Siamese</cat>
```

各出力エレメントに order 属性が含まれますが、`order by $i` のように、 FLWOR 式に **order by** 節が含まれない限り、出力ストリーム内のエレメントの実際の順序は保証されません。定位置変数は、出力シーケンスではなく、入力シーケンス内の値の順序位置を表します。

let 節

let 節は、変数を式の結果全体にバインドします。let 節は、反復処理を実行しません。

最も単純なタイプの **let** 節には、1 つの変数および関連する式が含まれます。以下の例では、**let** 節には、`$j` という変数と、シーケンス `(1, 2, 3)` を構成する式が含まれます。

```
let $j := (1, 2, 3)
return <output>{$j}</output>
```

let 節の評価時に、式を評価した結果のシーケンス全体について単一のバインディングが作成されます。

```
$j = 1 2 3
```

例の中の **return** 節は、1 回実行されます。この式は、以下の出力のような結果になります。

```
<output>1 2 3</output>
```

let 節には、複数の変数を含めることができます。ただし、**for** 節とは異なり、**let** 節は、各変数をその関連する式の結果にバインドし、反復処理は行いません。以下の例では、**let** 節に 2 つの変数 $a および $b、およびシーケンス 1 2 および 4 5 を構成する式が含まれています。

```
let $a := (1, 2), $b := (4, 5)
return <output>{$a, $b}</output>
```

let 節の評価時に、変数バインディングの 1 つのタプルが作成されます。

```
($a = 1 2, $b = 4 5)
```

例の中の **return** 節は、タプルに対して 1 回実行されます。この式は、以下の出力のような結果になります。

```
<output>1 2 4 5</output>
```

バインディング式が空のシーケンスに評価されると、空のシーケンスを含む let バインディングが作成されます。

同一式内の for 節および let 節

FLWOR 式に **for** 節および **let** 節の両方が含まれる場合、 **let** 節によって生成される変数バインディングが、 **for** 節によって生成される変数バインディングに追加されます。

以下の例では、**for** 節には、$a という変数と、シーケンス (1, 2, 3) を構成する式が含まれます。**let** 節には、$b という変数と、シーケンス (4, 5, 6) を構成する式が含まれます。

```
for $a in (1, 2, 3)
let $b := (4, 5, 6)
return <output>{$a, $b}</output>
```

この例の **for** 節および **let** 節の結果は、バインディングの 3 つのタプルです。タプルの数は、 **for** 節によって決定されます。

```
($a = 1, $b = 4 5 6)
($a = 2, $b = 4 5 6)
($a = 3, $b = 4 5 6)
```

例の中の **return** 節は、バインディングの各タプルに対して 1 回実行されます。この式は、以下の出力のような結果になります。

```
<output>1 4 5 6</output>
<output>2 4 5 6</output>
<output>3 4 5 6</output>
```

for 節と let 節の比較

for 節と **let** 節はどちらも変数をバインドしますが、変数のバインド方法が異なります。

以下の表は、類似した **for** 節および **let** 節を含む FLWOR 式によって戻される結果を比較する例を示しています。

表 31. FLWOR 式内の **for** *節と* **let** *節の比較*

照会の説明	FLWOR 式	結果

単一の変数をバインドする	`for $i in ("a", "b", "c")`
	`return <output>{$i}</output>`
		`<output>a</output>`
		`<output>b</output>`
		`<output>c</output>`

	`let $i := ("a", "b", "c")` `return <output>{$i}</output>`	`<output>a b c</output>`

複数の変数をバインドする	`for $i in ("a", "b"), $j in ("c", "d")`
	`return <output>{$i, $j}</output>`
		`<output>a c</output>`
		`<output>b c</output>`
		`<output>a d</output>`
		`<output>b d</output>`

	`let $i := ("a", "b"), $j := ("c", "d")` `return <output>{$i, $j}</output>`	`<output>a b c d</output>`

注: この表の式には **order by** 節が含まれていないため、出力されるエレメントの順序は非決定論的です。

for 節および let 節における変数の有効範囲

for 節または **let** 節でバインドされる変数は、変数バインディングの後に指定される FLWOR 式のすべてのサブ式について有効範囲内です。これは、**for** 節または **let** 節は、同じ節内のそれより前の節または前のバインディングでバインドされる変数を参照できることを意味します。

以下の例では、 FLWOR 式に以下の節が含まれています。

- 変数 `$orders` をバインドする **let** 節。
- `$orders` を参照し、変数 `$i` をバインドする **for** 節。
- `$orders` および `$i` の両方を参照し、変数 `$c` をバインドするもう 1 つの **let** 節。

この例では、注文のセット内のすべての固有の項目番号を検索し、各固有の項目番号に対する注文数を戻します。

```
let $orders := db2-fn:xmlcolumn("ORDERS.XMLORDER")
for $i in fn:distinct-values($orders/order/itemno)
let $c := fn:count($orders/order[itemno = $i])
return
<ordercount>
  <itemno> {$i} </itemno>
  <count> {$c} </count>
</ordercount>
```

重要: FLWOR 式の **for** 節および **let** 節は、同じ変数名を複数回バインドすることはできません。

where 節

FLWOR 式内の **where** 節は、**for** 節および **let** 節によって生成される変数バインディングのタプルをフィルタリングします。

where 節は、変数バインディングの各タプルに適用される条件を指定します。条件が true の場合 (すなわち、式の結果が有効なブール値 true の場合)、タプルが保持され、**return** 節の実行時にそのバインディングが使用されます。そうでない場合、タプルは廃棄されます。

以下の例において、 **for** 節は、変数 $x および $y を数値のシーケンスにバインドします。

```
for $x in (1.5, 2.6, 1.9), $y in (.5, 1.6, 1.7)
where ((fn:floor($x) eq 1) and (fn:floor($y) eq 1))
return <output>{$x, $y}</output>
```

for 節の評価時に、変数バインディングのタプルが 9 個作成されます。

```
($x = 1.5, $y = .5)
($x = 2.6, $y = .5)
($x = 1.9, $y = .5)
($x = 1.5, $y = 1.6)
($x = 2.6, $y = 1.6)
($x = 1.9, $y = 1.6)
($x = 1.5, $y = 1.7)
($x = 2.6, $y = 1.7)
($x = 1.9, $y = 1.7)
```

where 節がこれらのタプルをフィルタリングし、以下のタプルが保持されます。

```
($x = 1.5, $y = 1.6)
($x = 1.9, $y = 1.6)
($x = 1.5, $y = 1.7)
($x = 1.9, $y = 1.7)
```

return 節が残った各タプルに対して 1 回実行され、式の結果は以下の出力のようになります。

```
<output>1.5 1.6</output>
<output>1.9 1.6</output>
<output>1.5 1.7</output>
<output>1.9 1.7</output>
```

この例の式には **order by** 節が含まれていないため、出力されるエレメントの順序は非決定論的です。

order by 節

FLWOR 式内の **order by** 節は、**return** 節によって値が処理される順序を指定します。**order by** 節がない場合、 FLWOR 式の結果は非決定論的順序で戻されます。

order by 節には、1 つ以上の順序付けの指定が含まれます。順序付けの指定は、**where** 節によってフィルタリングされた後に保持される変数バインディングのタプルを再配列するために使用されます。結果の順序により、**return** 節が評価される順序が決まります。

順序付けの指定はそれぞれ、評価されて順序付けキーを作成する式、および順序付けキーに対してソート順序 (昇順または降順) を指定する順序修飾子で構成されます。2 つのタプルの相対順序は、それらの順序付けキーの値を左から右の方向に比較することにより決定されます。

以下の例では、FLWOR 式に、製品をその価格に基づいて降順にソートする **order by** 節が含まれています。

```
<price_list>{
    for $prod in db2-fn:xmlcolumn('PRODUCT.DESCRIPTION')/product/description
    order by xs:decimal($prod/price) descending
    return
    <product>{$prod/name, $prod/price}</product>}
</price_list>
```

order by 節の処理時に、順序付け指定の式が、**for** 節によって生成される各タプルについて評価されます。最初のタプルについて、式 `xs:decimal($prod/price)` によって戻される値は 9.99 です。続いてこの式が次のタプルについて評価され、式は値 19.99 を戻します。順序付けの指定で、項目を降順にソートすることが示されているため、価格が 19.99 の製品は、価格が 9.99 の製品より前にソートされます。このソート・プロセスは、すべてのタプルが再配列されるまで継続されます。続いて **return** 節が、再配列されたタプル・ストリーム内の各タプルについて 1 回実行されます。

SAMPLE データベースの表 PRODUCT.DESCRIPTION に対して実行すると、この例の照会は以下の結果を戻します。

```
<price_list>
    <product>
        <name>Snow Shovel, Super Deluxe 26"</name>
        <price>49.99</price>
    </product>
    <product>
        <name>Snow Shovel, Deluxe 24"</name>
        <price>19.99</price></product>
    <product>
        <name>Snow Shovel, Basic 22"</name>
        <price>9.99</price>
    </product>
    <product>
        <name>Ice Scraper, Windshield 4" Wide</name>
        <price>3.99</price>
    </product>
</price_list>
```

この例では、順序付け指定内の式により、`price` エレメントの値から、 xs:decimal 値が構成されます。XML スキーマ内の `price` エレメントのタイプ・アノテーションが xs:string であるため、このタイプ変換が必要です。この変換がないと、結果において、数値による順序付けではなく、ストリングによる順序付けが使用されます。

順序付けキーの値の動的タイプが xdt:untypedAtomic の場合は、順序付けキーの比較の規則において、非型付き原子データはストリングとして処理することが指示されているため、明示的タイプ変換も必要になります。

ヒント: FLWOR 式で **order by** 節を使用して、照会での値の順序付けを指定することができます。順序付けをしなければ、反復処理は必要ありません。例えば、以下のパス式は、顧客 ID (Cid) が 1000 より大きい `customerinfo` エレメントのリストを戻します。

```
db2-fn:xmlcolumn('CUSTOMER.INFO')/customerinfo[@Cid > "1000"]
```

しかし、これらの項目を顧客名の昇順で戻すには、次のような **order by** 節を含む FLWOR 式を指定する必要があります。

```
for $custinfo in db2-fn:xmlcolumn('CUSTOMER.INFO')/customerinfo
where ($custinfo/@Cid > "1000")
order by $custinfo/name ascending
return $custinfo
```

順序付けキーは、出力の一部である必要はありません。 以下の照会では、価格の降順で製品名のリストが作成されますが、出力に価格は含まれません。

```
for $prod in db2-fn:xmlcolumn('PRODUCT.DESCRIPTION')/product
order by $prod/description/price descending
return $prod/name
```

順序付け指定の比較の規則

順序付けの指定を評価し、比較する処理は、以下の規則に基づきます。

- 順序付け指定内の式が評価され、結果に原子化が適用されます。原子化の結果は、単一の原子値または空のシーケンスのいずれかでなければなりません。そうでない場合、エラーが戻されます。順序付け指定を評価した結果を、順序付けキーと呼びます。
- 順序付けキーのタイプが xdt:untypedAtomic の場合、そのキーは xs:string タイプにキャストされます。非型付き値を常にストリングとして扱うことにより、ソートされるすべての値のタイプに関する完全な知識がなくてもソート処理を開始することができます。
- 順序付けの指定によって生成される値がすべて同じタイプではない場合、これらの値 (キー) は、サブタイプ置換またはタイプのプロモーションによって共通のタイプに変換されます。キーは、**gt** 演算子をサポートする最も共通性の低いタイプに変換することにより比較されます。例えば、順序付けの指定により、xs:anyURI 値および xs:string 値の両方を含むキーのリストが生成された場合、キーは、xs:string タイプの **gt** 演算子を使用して比較されます。所与の順序付けの指定により生成された順序付けキーに、**gt** 演算子をサポートする共通のタイプがない場合、結果はエラーとなります。
- 順序付けキーの値は、バインド済み変数のタプルを return 節に渡して実行する順序を決定するために使用されます。タプルの順序付けは、以下の規則を使用して、順序付けキーを左から右の方向に比較することにより決定されます。
  - ソート順が昇順の場合、他のタプルより大きい順序付けキーを持つタプルが、それらのタプルより後にソートされます。
  - ソート順が降順の場合、他のタプルより大きい順序付けキーを持つタプルが、それらのタプルより前にソートされます。

  順序付けキーの「より大きい」関係は、以下のように定義されます。
  - 空のシーケンスは、他のどの値よりも大きいものとして扱われます。
  - NaN は、空のシーケンスを除く他のどの値よりも大きいものとして解釈されます。
  - ある値が他の値と比較されて **gt** 演算子が true を戻す場合、その値は他の値より大きいものとして扱われます。
  - `+0.0 gt -0.0` および `-0.0 gt +0.0` は両方とも false であるため、特別浮動小数点値の正のゼロまたは負のゼロは、どちらも他方より大きいものとして扱われません。

  注: 順序付けキーが空であるタプルは、 **ascending** オプションが指定されている場合 (デフォルトの設定) は出力ストリームの最後に、**descending** オプションが指定されている場合は出力ストリームの先頭に出力されます。

return 節

return 節は、FLWOR 式の結果を生成します。

return 節は、 FLWOR 式のその他の節によって生成される変数バインディングの各タプルに対して 1 回評価されます。**return** 節によるバインド済み変数のタプルの処理順序は、 FLWOR 式に **order by** 節が含まれていない限り非決定論的です。

return 節内の式が非更新式である場合、**return** 節のすべての評価結果は連結されて非更新 FLWOR 式の結果が形成されます。

return 節内の式が更新式である場合、**return** 節のすべての評価の結果は更新のリストになります。 FLWOR 式を含む変換式は、更新を、この変換式の **modify** 節内の他の更新式によって戻される更新とマージした後に実行します。

ヒント: return 節では、最上位のコンマ演算子を含む式を括弧を使用して囲みます。 FLWOR 式はコンマ演算子より優先順位が高いため、最上位のコンマ演算子を含む式は、括弧を使用しないとエラーまたは予期しない結果になる可能性があります。

FLWOR の例

これらの例は、FLWOR 式を使用して、完全な照会において、結合、グループ化、集約を実行する方法を示します。

XML データを結合する FLWOR 式

次の照会は SAMPLE データベースの PRODUCT および PURCHASEORDER テーブルの XML データを結合して、2005 年の購入注文で注文された製品の名前をリストします。

製品文書および PurchaseOrder 文書内のエレメントはともに同じネーム・スペースに存在するため、照会はデフォルトのネーム・スペースを宣言することで開始され、照会のエレメント名は接頭部を必要としません。**for** 節は PURCHASEORDER.PORDER 列を繰り返します。具体的には、「2005」で始まる OrderDate 属性値を持つ購入注文となります。各購入注文について、**let** 節は *$parts* 変数に部品 ID 値 (partid) を割り当てます。続いて、**return** 節は、購入注文に含まれる製品名をリストします。

```
declare default element namespace 'http://posample.org';
for $po in db2-fn:xmlcolumn('PURCHASEORDER.PORDER')
  /PurchaseOrder[fn:starts-with(@OrderDate, "2005")]
let $parts := $po/item/partid
return
<ProductList PoNum = "{$po/@PoNum}">
    { db2-fn:xmlcolumn('PRODUCT.DESCRIPTION')
    /product[@pid = $parts]/description/name }
</ProductList>
```

この照会は以下の結果を戻します。

```
<ProductList xmlns="http://posample.org" PoNum="5001">
    <name>Snow Shovel, Deluxe 24 inch</name>
    <name>Snow Shovel, Super Deluxe 26 inch</name>
    <name>Ice Scraper, Windshield 4 inch</name>
</ProductList>
<ProductList xmlns="http://posample.org" PoNum="5003">
    <name>Snow Shovel, Basic 22 inch</name>
</ProductList>
```

```
<ProductList xmlns="http://posample.org" PoNum="5004">
    <name>Snow Shovel, Basic 22 inch</name>
    <name>Snow Shovel, Super Deluxe 26 inch</name>
</ProductList>
```

エレメントをグループ化する FLWOR 式

次の照会は、SAMPLE データベースの CUSTOMER 表内のカスタマー名を市区町村別でグループ化します。**for** 節は customerinfo 文書を繰り返して、それぞれの市区町村エレメントを変数 *$city* にバインドします。各市区町村について、**let** 節は、変数 *$cust-names* をその市区町村に存在するすべてのカスタマー名の非順序リストにバインドします。照会はそれぞれの市区町村エレメントに含まれる市区町村名、およびその市区町村に住むすべてのカスタマーのネストされた名前エレメントを戻します。

```
declare default element namespace 'http://posample.org';
for $city in fn:distinct-values(db2-fn:xmlcolumn('CUSTOMER.INFO')
    /customerinfo/addr/city)
let $cust-names := db2-fn:xmlcolumn('CUSTOMER.INFO')
    /customerinfo/name[../addr/city = $city]
order by $city
return <city>{$city, $cust-names} </city>
```

この照会は以下の結果を戻します。

```
<city xmlns="http://posample.org">Aurora
    <name>Robert Shoemaker</name>
</city>
<city xmlns="http://posample.org">Markham
    <name>Kathy Smith</name>
    <name>Jim Noodle</name>
</city>
<city xmlns="http://posample.org">Toronto
    <name>Kathy Smith</name>
    <name>Matt Foreman</name>
    <name>Larry Menard</name>
</city>
```

データを集約する FLWOR 式

次の照会は 2005 年の各購入注文によって生成された合計売上を戻し、HTML レポートを作成します。

照会はオーダー日付が 2005 年の各 PurchaseOrder エレメントを繰り返し、エレメントを **for** 節の変数 *$po* にバインドします。続いて、パス式 `$po/item/` はコンテキスト位置を PurchaseOrder エレメント内の各項目エレメントに移動します。ネストされた式 `(price * quantity)` はその項目の合計売上を判別します。fn:sum 関数は各項目の合計売上の結果シーケンスを追加します。 **let** 節は fn:sum 関数の結果を変数 *$revenue* にバインドします。 **order by** 節は結果を各購入注文の合計売上でソートします。最後に、**return** 節はレポート・テーブルに各購入注文の行を作成します。

```
declare default element namespace 'http://posample.org';
<html>
<body>
<h1>PO totals</h1>
<table>
<thead>
    <tr>
        <th>PO Number</th>
```

```
        <th>Status</th>
        <th>Revenue</th>
    </tr>
</thead>
<tbody>{
    for $po in db2-fn:xmlcolumn('PURCHASEORDER.PORDER')/
      PurchaseOrder[fn:starts-with(@OrderDate, "2005")]
    let $revenue := sum($po/item/(price * quantity))
    order by $revenue descending
    return
      <tr>
          <td>{string($po/@PoNum)}</td>
          <td>{string($po/@Status)}</td>
          <td>{$revenue}</td>
      </tr>
}
</tbody>
</table>
</body>
</html>
```

この照会は以下の結果を戻します。

```
<html xmlns="http://posample.org">
<body>
<h1>PO totals</h1>
<table>
<thead>
    <tr>
        <th>PO Number</th>
        <th>Status</th>
        <th>Revenue</th>
    </tr>
</thead>
<tbody>
    <tr>
        <td>5004</td>
        <td>Shipped</td>
        <td>139.94</td>
    </tr>
    <tr>
        <td>5001</td>
        <td>Shipped</td>
        <td>123.96</td>
    </tr>
    <tr>
        <td>5003</td>
        <td>UnShipped</td>
        <td>9.99</td>
    </tr>
</tbody>
</table>
</body>
</html>
```

ブラウザーで表示したときには、照会は以下の表に近い形で出力されます。

PO totals

PO Number	Status	Revenue
5004	Shipped	139.94
5001	Shipped	123.96
5003	Unshipped	9.99

XML データを更新する FLWOR 式

以下の例は、DB2 SAMPLE データベースの CUSTOMER 表を使用します。CUSTOMER 表の XML 列 INFO には、顧客の住所と電話に関する情報が含まれています。

変換式は顧客情報を含む XML 文書のコピーを作成します。 **modify** 節の FLWOR 式と名前変更式は、phone というノード名のインスタンスをすべて phonenumber という名前に変更します。

```
xquery
declare default element namespace 'http://posample.org';
transform
  copy $mycust := db2-fn:sqlquery('select info from customer where cid = 1003')
  modify
    for $phone in $mycust/customerinfo/phone
    return
      do rename $phone as "phonenumber"
  return $mycust
```

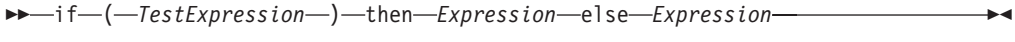
SAMPLE データベースに対して実行すると、この式は phone というノード名を phonenumber に変更し、次の結果を戻します。

```
<customerinfo xmlns="http://posample.org" Cid="1003">
  <name>Robert Shoemaker</name>
  <addr country="Canada">
    <street>1596 Baseline</street>
    <city>Aurora</city>
    <prov-state>Ontario</prov-state>
    <pcode-zip>N8X 7F8</pcode-zip>
  </addr>
  <phonenumber type="work">905-555-7258</phonenumber>
  <phonenumber type="home">416-555-2937</phonenumber>
  <phonenumber type="cell">905-555-8743</phonenumber>
  <phonenumber type="cottage">613-555-3278</phonenumber>
</customerinfo>
```

条件式

条件式では、キーワード **if**、**then**、および **else** を使用して、テスト式の値が true または false のいずれであるかに応じて 2 つの式のいずれかを評価します。

構文



if テスト式の直前に配置するキーワード。

TestExpression
評価する条件式の部分を決定する XQuery 式。

then
TestExpression の有効なブール値が true の場合、このキーワードの後の式が評価されます。テスト式の有効なブール値が false の場合、式は評価またはエラー・チェックされません。

else

TestExpression の有効なブール値が false の場合、このキーワードの後の式が評価されます。テスト式の有効なブール値が true の場合、式は評価またはエラー・チェックされません。

Expression

任意の XQuery 式。式に最上位のコンマ演算子が含まれる場合、式を括弧で囲む必要があります。

then または **else** 条件分岐のいずれかに更新式が含まれる場合、条件式は更新式となります。更新式は、変換式の **modify** 節内に指定する必要があります。

更新条件式の各分岐には、更新式または空のシーケンスのいずれかが含まれていなければなりません。テスト式の値に基づいて、**then** または **else** 節のいずれかが選択され、評価されます。条件更新式の結果は、選択された分岐によって戻される更新のリストです。これが含まれる変換式は、更新を、この変換式の **modify** 節内の他の更新式によって戻される更新とマージした後に実行します。

例

以下の例では、照会は属性 basic を含む product エレメントのリストを構成します。 basic 属性の値は、price エレメントの値が 10 未満であるかどうかに応じて条件付きで指定されます。

```
declare namespace ns1= "http://posample.org";
for $prod in db2-fn:xmlcolumn('PRODUCT.DESCRIPTION')/ns1:product/ns1:description
return (
if (xs:decimal($prod/ns1:price) < 10)
  then <product basic = "true">{fn:data($prod/ns1:name)}</product>
  else <product basic = "false">{fn:data($prod/ns1:name)}</product>)
```

この照会は以下の結果を戻します。

```
<product basic="true">Snow Shovel, Basic 22"</product>
<product basic="false">Snow Shovel, Deluxe 24"</product>
<product basic="false">Snow Shovel, Super Deluxe 26"</product>
<product basic="true">Ice Scraper, Windshield 4" Wide</product>
```

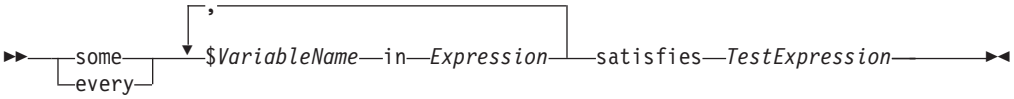
この例では、テスト式により、price エレメントの値から xs:decimal 値が構成されます。 xs:decimal 関数の使用により、強制的に 10 進数比較が行われます。

量化式

量化式は、1 つ以上のシーケンスの一部またはすべての項目が特定の条件を満たす場合に true を戻します。量化式の値は、常に true または false です。

量化式の先頭には、式が存在量化または全称量化のいずれを実行するかを示す数量詞 (**some** または **every**) を記述します。数量詞の後に、式によって戻される項目に変数をバインドする 1 つ以上の節が続きます。続いてバインド済み変数がテスト式で参照され、バインド済みの値の一部またはすべてが特定の条件を満たしているかどうかが判別されます。

構文



some
このキーワードが指定されていると、量化式は、*Expression* によって戻される少なくとも *1* つの 項目について *TestExpression* の有効ブール値が true である場合に、true を戻します。そうでない場合、量化式は false を戻します。

every
このキーワードが指定されていると、量化式は、*Expression* によって戻されるすべての 項目について *TestExpression* の有効ブール値が true である場合に、true を戻します。そうでない場合、量化式は false を戻します。

VariableName
Expression の結果の各項目にバインドされる変数の名前。量化式でバインドされる変数は、量化式において変数バインディングの後に指定されるすべてのサブ式について有効範囲内です。

Expression
任意の XQuery 式。式に最上位のコンマ演算子が含まれる場合、式を括弧で囲む必要があります。

satisfies
テスト式の直前に配置するキーワード。

TestExpression
Expression によって戻されるシーケンスの一部またはすべての項目が満たす必要のある条件を指定する XQuery 式。

注: エラーが発生した場合、量化比較の結果はブール値またはエラーのいずれかになる可能性があります。

例

- 以下の例の量化式は、SAMPLE データベースの CUSTOMER.INFO 列内のすべての顧客の住所が Canada にある場合に、true を戻します。

```
every $cust in db2-fn:xmlcolumn('CUSTOMER.INFO')/customerinfo
satisfies $cust/addr/@country = "Canada"
```

- 以下の例では、変数 a および b にバインドされる値のすべての組み合わせ (全部で 9 個の組み合わせがあります) について、各量化式がそのテスト式を評価します。

  以下の式の結果は true です。

```
some $a in (3, 5, 9), $b in (1, 3, 5)
satisfies $a * $b = 27
```

  以下の式の結果は false です。

```
every $a in (3, 5, 9), $b in (1, 3, 5)
satisfies $a * $b = 27
```

- 以下の例は、エラーがある場合、量化式の結果は決定論的でないことを説明しています。テスト式が、一方の変数バインディングについては true を、もう一方の変数バインディングについてはエラーを戻すため、この式は true またはエラーのいずれかを戻す可能性があります。

  ```
  some $a in (3, 5, "six") satisfies $a * 3 = 9
  ```

  同様に、以下の式は false またはエラーを戻す可能性があります。

  ```
  every $a in (3, 5, "six") satisfies $a * 3 = 9
  ```

キャスト式

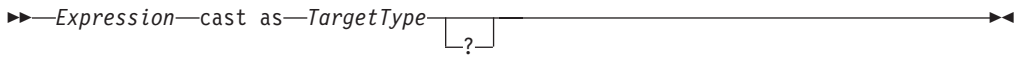
キャスト式では、既存の値に基づいて特定のタイプの新しい値を作成します。

キャスト式では、2 つのオペランド (入力式およびターゲット・タイプ) を使用します。キャスト式が評価されるときは、原子化を使用して入力式の結果が原子値または空のシーケンスに変換されます。原子化の結果が複数の原子値のシーケンスである場合、エラーが戻されます。エラーが戻されない場合、キャスト式は、入力値に基づくターゲット・タイプの新しい値の作成を試行します。入力タイプとターゲット・タイプの組み合わせによっては、キャストがサポートされていません。どのタイプを別のどのタイプにキャストできるかについては、 27 ページの『タイプ・キャスト』を参照してください。値をデータ・タイプにキャストする際には、キャスト可能な式を使用して、値をデータ・タイプにキャストできるかどうかテストできます。

空のシーケンスは、ターゲット・タイプの後に疑問符 (?) が付いた場合のみ有効な入力値です。

キャスト式のターゲット・タイプが xs:QName か、xs:QName または xs:NOTATION から派生したタイプで、さらに入力式が xs:string タイプで、リテラル・ストリングでない場合は、エラーが戻されます。

構文



Expression
1 つの原子値または空のシーケンスを戻す任意の XQuery 式。空のシーケンスは、*TargetType* の後に疑問符 (?) が付いたときに許可されます。

TargetType
Expression の値がキャストされるタイプ。*TargetType* は、事前定義されている原子 XML スキーマ・タイプの原子タイプでなければなりません。*TargetType* に対して、データ・タイプ xs:NOTATION、xdt:anyAtomicType、および xs:anySimpleType は有効なタイプではありません。

? *Expression* の結果が空のシーケンスである可能性もあることを示します。

例

以下の例では、キャスト式を使用して xs:string タイプを持つ price エレメントの値を、xs:decimal タイプにキャストします。

```
for $price in db2-fn:xmlcolumn('PRODUCT.DESCRIPTION')/product/description/price
return $price cast as xs:decimal
```

SAMPLE データベースの表 PRODUCT.DESCRIPTION に対して実行すると、この例の照会は以下の結果を戻します。

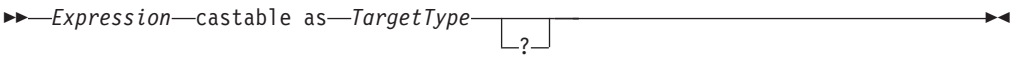
```
9.99
19.99
49.99
3.99
```

キャスト可能な式

キャスト可能な式は、特定のデータ・タイプに値をキャストできるかどうかをテストします。値をデータ・タイプにキャストできる場合、キャスト可能な式は true を戻します。キャストできない場合、式は false を戻します。

キャスト可能な式を、述部として使用して評価時のキャスト・エラーを回避できます。また、値の処理に適するタイプを選択するのに使用することもできます。どのタイプを別のどのタイプにキャストできるかについては、 27 ページの『タイプ・キャスト』を参照してください。

構文



Expression
1 つの原子値または空のシーケンスを戻す XQuery 式。

TargetType
Expression の値をキャストできるかどうかのテストに使用するタイプ。*TargetType* は、事前定義されている XML スキーマ・タイプのいずれかの原子タイプでなければなりません。*TargetType* に対して、データ・タイプ xs:NOTATION、xdt:anyAtomicType、および xs:anySimpleType は有効なタイプではありません。

? 空のシーケンスがターゲット・タイプの有効なインスタンスと見なされることを示します。*Expression* が空のシーケンスに評価される場合、? が指定されていないと、キャスト可能な式は False を戻します。

戻り値

Expression を *TargetType* にキャストできる場合、キャスト可能な式は true を戻します。キャストできない場合、式は false を戻します。

Expression の結果が空のシーケンスで、*TargetType* の後に疑問符の標識がある場合、キャスト可能な式は true を戻します。以下の例では、ターゲット・タイプ xs:integer の後に疑問符の標識があります。

```
$prod/revision castable as xs:integer?
```

キャスト可能な式の *TargetType* が xs:QName か、xs:QName または xs:NOTATION から派生したタイプで、さらに *Expression* が xs:string タイプで、リテラル・ストリングでない場合は、キャスト可能な式の戻り値は false です。

Expression の結果が複数原子値のシーケンスの場合、エラーが戻されます。

例

以下の例は、キャスト可能な式を述部として使用して、評価時のエラーを回避します。以下の例は、`@OrderDate` が有効な日付でない場合に動的エラーを回避します。

```
$po/orderID[if ( $po/@OrderDate castable as xs:date)
   then xs:date($po/@OrderDate) gt xs:date("2000-01-01")
   else false()]
```

日付の属性が 2000 年 1 月 1 日より後の有効な日付である場合のみ、述部は true で、`orderID` を戻します。それ以外の場合、述部は false で、空のシーケンスを戻します。

以下の例は、指定された値の処理に適するタイプを選択するのに、キャスト可能な式を使用します。この例は、キャスト可能な式を使用して、郵便番号を整数またはストリングとしてキャストします。

```
if ($postalcode castable as xs:integer)
   then $postalcode cast as xs:integer
   else $postalcode cast as xs:string
```

以下の例は、FLWOR **let** 節でキャスト可能な式を使用して、`$prod/mfgdate` の値をテストし、値を `$currdate` にバインドします。キャスト可能な式とキャスト式は、疑問符の標識を使用して、空のシーケンスの処理をサポートしています。

```
let $currdate := if ($prod/mfgdate castable as xs:date?)
   then $prod/mfgdate cast as xs:date?
   else "1000-01-01" cast as xs:date
```

`$prod/mfgdate` の値を xs:date としてキャストできる場合、データ・タイプにキャストされ、`$currdate` にバインドされます。`$prod/mfgdate` が空のシーケンスである場合、空のシーケンスが `$currdate` にバインドされます。`$prod/mfgdate` を xs:date としてキャストできない場合、タイプ xs:date の 1000-01-01 の値が `$currdate` にバインドされます。

以下の例は、キャスト可能な式を使用して、比較を実行する前に製品カテゴリーの値をテストします。XML 列 FEATURES.INFO で、文書にはエレメント `/prod/category` が含まれています。値は数字コードかストリング・コードです。XMLEXISTS 述部中のキャスト可能な式は、比較を実行する前に `/prod/category` の値をテストして、評価時のエラーを回避します。

```
SELECT F.PRODID FROM F FEATURES
WHERE xmlexists('$test/prod/category[ (( . castable as xs:double) and . > 100 ) or
   (( . castable as xs:string) and . > "A100" )]'
   passing F.INFO as "test")
```

戻り値は製品 ID で、カテゴリー・コードは数値 100 より大きい値か、ストリング ″A100″ より大きい値です。

変換式と更新式

DB2 XQuery を使用して既存の XML データを更新するには、変換式の **modify** 節内で更新式を使用します。

変換式での更新式の使用

DB2 XQuery の更新式は、変換式の **modify** 節で使用する必要があります。更新式は、変換式の **copy** 節によって作成される、コピーされたノードを操作します。

以下の式が更新式です。

- 削除式
- 挿入式
- 名前変更式
- 置換式
- **return** 節に更新式を含む FLWOR 式
- **then** または **else** 節に更新式を含む条件式
- すべてのオペランドが更新式か空のシーケンスの、コンマで区切られた 2 つ以上の更新式

無効な更新式の場合、DB2 XQuery はエラーを戻します。例えば、条件式の一方の分岐に更新式が含まれ、もう一方の分岐には更新式でも空のシーケンスでもないものが含まれる場合、DB2 XQuery はエラーを戻します。

変換式は、既存のノードを変更するわけではないので、更新式ではありません。変換式は、既存のノードの変更されたコピーを作成します。変換式の結果は、変換式の **modify** 節内の更新式によって作成されたノードと、既存のノードのコピーを含むことができます。

XQuery 更新操作の処理

変換式では、**modify** 節で複数の更新を指定できます。例えば、既存の値を置き換える式と、新しいエレメントを挿入する式の、2 つの更新式を **modify** 節に含めることができます。 **modify** 節に複数の更新式が含まれている場合、各更新式は単独で評価され、変換式の **copy** 節によって作成された特定のノードに関連付けられた変更操作のリストができます。

modify 節内で、更新式は他の更新式によって追加される新しいノードを変更できません。例えば、更新式が新しいエレメント・ノードを追加する場合、別の更新式は新しく作成されたそのノードのノード名を変更することはできません。

変換式の **modify** 節で指定されたすべての変更操作が収集され、以下の順序で実際に適用されます。

1. 以下の更新操作が非決定論的順序で行われます。
   - **before**、**after**、**as first**、**as last** などの順序付けキーワードを使用していない挿入操作。
   - すべての名前変更操作。

- キーワード **value of** が指定されていて、ターゲット・ノードが属性ノード、テキスト・ノード、コメント・ノード、または処理命令ノードである置換操作。

2. **before**、**after**、**as first**、**as last** などの順序付けキーワードを使用している挿入操作。
3. キーワード **value of** が指定されていない置換操作。
4. キーワード **value of** が指定されていてターゲット・ノードがエレメント・ノードである置換操作。
5. すべての削除操作。

変更操作がこの順序で適用されることによって、一連の複数の変更が決定論的結果になります。更新操作の順序によって、一連の複数の変更が決定論的結果になることが保証されることの例については、『例』にある最後の XQuery 式を参照してください。

無効な XQuery 更新操作

変換式の処理中に以下のいずれかの状態が検出された場合、DB2 XQuery はエラーを戻します。

- 同一ノードに 2 つ以上の名前変更操作が適用される。
- **value of** キーワードを使用する複数の置換操作が同じノードに適用される。
- **value of** キーワードを使用しない複数の置換操作が同じノードに適用される。
- 変換式の結果が有効な XDM インスタンスでない。

  無効な XDM インスタンスの例として、2 つの属性を持つ 1 つのエレメントが含まれていて、両方の属性の名前が同じであるインスタンスがあります。
- XDM インスタンスに、不整合なネーム・スペース・バインディングが含まれる。

  不整合なネーム・スペース・バインディングとは、例えば次のようなものです。
  - 属性ノードの QName でのネーム・スペース・バインディングが、その親エレメント・ノードでのネーム・スペース・バインディングと一致しない。
  - 同じ親を持つ 2 つの属性ノードでのネーム・スペース・バインディングが、相互に一致しない。

例

以下の例では、変換式の **copy** 節が変数 $product をエレメント・ノードのコピーにバインドし、変換式の **modify** 節が 2 つの更新式を使用して、コピーされたノードを変更します。

```
xquery
declare default element namespace "http://posample.org";
transform
copy $product := db2-fn:sqlquery(
  "select description from product where pid='100-100-01'")/product
modify(
  do replace value of $product/description/price with 349.95,
  do insert <status>Available</status> as last into $product )
return $product
```

以下の例では、SQL UPDATE ステートメント内で XQuery 変換式を使用して、CUSTOMER 表の XML データを変更します。この SQL UPDATE ステートメントは、CUSTOMER 表の行を操作します。変換式が行の INFO 列から XML 文書のコピーを作成し、そのコピーに status エレメントを追加します。UPDATE ステートメントが、行の INFO 列の文書を、変換式によって変更された文書のコピーに置き換えます。

```
UPDATE customer
SET info = xmlquery( 'declare default element namespace "http://posample.org";
   transform
   copy $newinfo := $info
   modify do insert <status>Current</status> as last into $newinfo/customerinfo
   return $newinfo' passing info as "info")
WHERE cid = 1003
```

以下の例では、DB2 SAMPLE データベースの CUSTOMER 表を使用します。CUSTOMER 表の XML 列 INFO には、顧客の住所と電話に関する情報が含まれています。

以下の例の SQL SELECT ステートメントは、CUSTOMER 表の行を操作します。変換式の **copy** 節が、INFO 列から XML 文書のコピーを作成します。削除式が、住所情報と work 以外の電話番号を文書のコピーから削除します。 **return** は、CUSTOMER 表のオリジナル文書にあるカスタマー ID 属性と country 属性を使用します。

```
SELECT XMLQUERY( 'declare default element namespace "http://posample.org";
  transform
  copy $mycust := $d
  modify
    do delete ( $mycust/customerinfo/addr,
       $mycust/customerinfo/phone[@type != "work"] )
  return
  <custinfo>
    <Cid>{data($d/customerinfo/@Cid)}</Cid>
    {$mycust/customerinfo/*}
    <country>{data($d/customerinfo/addr/@country)}</country>
  </custinfo>'
  passing INFO as "d")
FROM CUSTOMER
WHERE CID = 1003
```

SAMPLE データベースに対して実行すると、このステートメントは以下の結果を戻します。

```
<custinfo xmlns="http://posample.org">
  <Cid>1003</Cid>
  <name>Robert Shoemaker</name>
  <phone type="work">905-555-7258</phone>
  <country>Canada</country>
</custinfo>
```

以下の例の XQuery 式は、一連の複数の変更が決定論的結果になることが、更新操作の順序によって保証されることを示しています。挿入式が phone エレメントの後に status エレメントを追加し、置換式が phone エレメントを email エレメントに置き換えます。

```
xquery
declare default element namespace "http://posample.org";
let $email  := <email>jnoodle@my-email.com</email>
let $status := <status>current</status>
return
```

```
transform
copy $mycust := db2-fn:sqlquery('select info from customer where cid = 1002')
modify (
  do replace $mycust/customerinfo/phone with $email,
  do insert $status after $mycust/customerinfo/phone[@type = "work"] )
return $mycust
```

modify 節内では、置換式が挿入式の前にあります。ただし、コピーされたノード・シーケンス $mycust を更新するときは、決定論的結果になるように、置換更新操作の前に挿入更新操作が行われます。 SAMPLE データベースに対して実行すると、この式は以下の結果を戻します。

```
<customerinfo xmlns="http://posample.org" Cid="1002">
  <name>Jim Noodle</name>
  <addr country="Canada">
    <street>25 EastCreek</street>
    <city>Markham</city>
    <prov-state>Ontario</prov-state>
    <pcode-zip>N9C 3T6</pcode-zip>
  </addr>
  <email>jnoodle@my-email.com</email>
  <status>current</status>
</customerinfo>
```

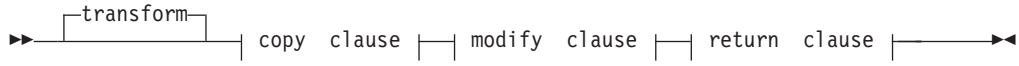
最初に置換操作が行われると、ノード・シーケンスに phone エレメントが存在しなくなるので、phone エレメントの後に status エレメントを挿入するという操作は、成り立たなくなってしまいます。

更新操作の順序については、 120 ページの『XQuery 更新操作の処理』を参照してください。

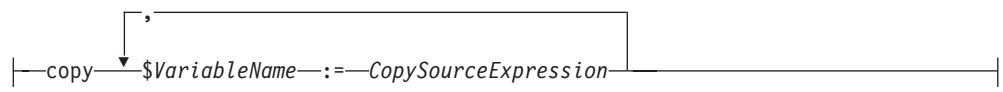
変換式

変換式は、1 つ以上のノードのコピーを作成します。変換式の **modify** 節内の更新式は、コピーされたノードを変更します。**return** 節内の式は、変換式の結果を指定します。

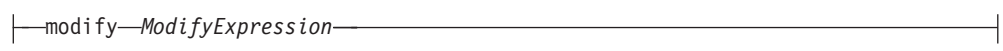
構文



copy clause:



modify clause:



return clause:

パラメーター

transform
変換式を開始するために使用できるオプションのキーワード。

copy
変換式の **copy** 節を開始するためのキーワード。 **copy** 節内の各 *VariableName* は、対応する *CopySourceExpression* によって戻されるノード・ツリーのコピーにバインドされます。

VariableName
CopySourceExpression によって戻されるノード・ツリーのコピーにバインドする変数の名前。

CopySourceExpression
更新式以外の XQuery 式。式は、単一のノードと、ノード・ツリーと呼ばれるその子孫 (ある場合) を一緒に戻さなければなりません。

式に最上位のコンマ演算子が含まれる場合、式を括弧で囲む必要があります。*CopySourceExpression* は、あたかもエレメント・コンストラクター内の括弧で囲まれた式であるかのように評価されます。
copy 節によって作成されたノードは新しいノード ID を持ち、非型付きノードになります。例えば、xs:decimal タイプのエレメント・ノードからコピーが作成された場合、コピーされたノードは独自のノード ID を持ち、タイプは xs:anyType になります。

modify
変換式の **modify** 節を開始するためのキーワード。

ModifyExpression
更新式または空のシーケンス。式に最上位のコンマ演算子が含まれる場合、式を括弧で囲む必要があります。更新式が評価され、その結果の更新内容が、**copy** 節によって作成されたノードに適用されます。

更新式のターゲット・ノードが、その更新式を含む変換式の **copy** 節によって作成されたノードでない場合、DB2 XQuery はエラーを戻します。例えば、名前変更式が名前変更しようとするノードが、**copy** 節によって作成されたノードでない場合、DB2 XQuery はエラーを戻します。

modify 節で指定された更新は、隣接する複数のテキスト・ノードを子として持つノードになります。隣接する複数のテキスト・ノードがノードの子として存在する場合、隣接するテキスト・ノードは単一のテキスト・ノードにマージされます。結果として得られるテキスト・ノードのストリング値は、隣接するテキスト・ノードがスペース挿入なしで連結されたストリング値です。作成された子ノードがゼロ長ストリングのストリング値を持つテキスト・ノードだった場合、そのテキスト・ノードは削除されます。

return
変換式の **return** 節を開始するためのキーワード。

ReturnExpression
更新式以外の XQuery 式。式に最上位のコンマ演算子が含まれる場合、式を括弧で囲む必要があります。

return 節内の式は評価され、変換式の結果として戻されます。 **return** 節内の式は、**modify** 節内の更新式によって変更または作成されたノードにアクセスできます。

変換式の **return** 節が戻せるのは、**copy** 節によって作成されたノードのみに限定されません。*ReturnExpression* は、コピーされたノード、オリジナルのノード、構成されたノードのそれぞれの組み合わせを戻すことができます。

ReturnExpression の結果が、新しいノードやコピーされたノードなどの非型付きノードを含む XML 文書の場合は、SQL/XML 関数の XMLVALIDATE を使用して、XML 文書内のノードに特定のタイプ情報を割り当てることができます。

例

以下の例では、DB2 SAMPLE データベースの CUSTOMER 表を使用します。CUSTOMER 表の XML 列 INFO には、顧客の住所と電話に関する情報が含まれています。

以下の例では、変換式の **copy** 節が INFO 列から XML 文書のコピーを作成します。 **modify** 節内の削除式は、電話の type 属性が home でないすべての電話番号を、XML 文書から削除します。

```
xquery
declare default element namespace 'http://posample.org';
transform
   copy $mycust := db2-fn:sqlquery('select INFO from CUSTOMER where Cid = 1003')
   modify
     do delete $mycust/customerinfo/phone[@type!="home"]
   return $mycust;
```

SAMPLE データベースに対して実行すると、この式は以下の結果を戻します。

```
<customerinfo xmlns="http://posample.org" Cid="1003">
  <name>Robert Shoemaker</name>
  <addr country="Canada">
    <street>1596 Baseline</street>
    <city>Aurora</city>
    <prov-state>Ontario</prov-state>
    <pcode-zip>N8X 7F8</pcode-zip>
  </addr>
  <phone type="home">416-555-2937</phone>
</customerinfo>
```

以下の式ではオプションのキーワード **transform** を使用しません。変換式は **copy** 節で開始し、前の式と同等です。

```
xquery
declare default element namespace 'http://posample.org';
copy $mycust := db2-fn:sqlquery('select INFO from CUSTOMER where Cid = 1003')
modify
   do delete $mycust/customerinfo/phone[@type!="home"]
return $mycust;
```

以下の例では、SQL UPDATE ステートメントが CUSTOMER 表の行の XML 文書を変更して妥当性検査をします。変換式の **copy** 節が、INFO 列から XML 文書のコピーを作成します。置換式が、XML 文書のコピー内の name エレメントの値を変

更します。文書のコピーは妥当性検査されておらず、文書のコピーのノードにはタイプ情報がありません。 XMLVALIDATE 関数が文書のコピーの妥当性検査をし、ノードにタイプ情報を追加します。

```
UPDATE customer set info = XMLVALIDATE(
  XMLQUERY(' declare default element namespace "http://posample.org";
  transform
  copy $mycust := $cust
  modify
    do replace value of $mycust/customerinfo/name with "Larry Menard, Jr."
  return $mycust'
  passing info as "cust" )
ACCORDING TO XMLSCHEMA ID customer)
where cid = 1005
```

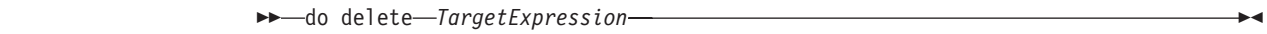
基本更新式

XQuery の 4 つの基本更新式を使用して、既存の XML データを更新するための複雑な更新式を作成できます。DB2 XQuery を使用する場合、更新式は変換式の **modify** 節内で使用します。

削除式

削除式は、ノード・シーケンスから 0 個以上のノードを削除します。

構文



do delete
削除式を開始するためのキーワード。

TargetExpression
更新式以外の XQuery 式。式に最上位のコンマ演算子が含まれる場合、式を括弧で囲む必要があります。*TargetExpression* の結果は、0 個以上のノードのシーケンスでなければなりません。各ノードの親プロパティーが空であってはなりません。

変換式が削除式を評価し、削除対象のノードから成る更新のリストを生成します。*TargetExpression* に一致するノードは、削除対象としてマークされます。*TargetExpression* のノードを削除する際、そのノードは親ノードからデタッチされます。削除されたノードとその子は、ノード・シーケンスの一部ではなくなります。

例

以下の例では、DB2 SAMPLE データベースの CUSTOMER 表を使用します。CUSTOMER 表の XML 列 INFO には、顧客の住所と電話に関する情報が含まれています。

以下の式は、住所エレメントとそのすべての子ノード、および電話の type 属性が home でないすべての電話番号を、XML 文書から削除します。

```
xquery
declare default element namespace 'http://posample.org';
transform
```

```
copy $mycust := db2-fn:sqlquery('select INFO from CUSTOMER where Cid =1003')
modify
  do delete ($mycust/customerinfo/addr, $mycust/customerinfo/phone[@type!="home"])
return $mycust
```

SAMPLE データベースに対して実行すると、この式は以下の結果を戻します。

```
<customerinfo xmlns="http://posample.org" Cid="1003">
  <name>Robert Shoemaker</name>
  <phone type="home">416-555-2937</phone>
</customerinfo>
```

以下の例は、phone エレメント・ノードから、属性値が home の type 属性を削除します。

```
xquery
declare default element namespace "http://posample.org";
transform
copy $mycust := db2-fn:sqlquery('select info from customer where cid = 1004')
modify (
  for $phone in $mycust/customerinfo//phone[@type="home"]
  return
    do delete $phone/@type )
return $mycust
```

SAMPLE データベースに対して実行すると、この式は以下の結果を戻します。

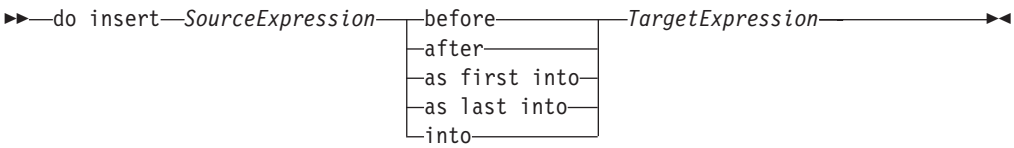
```
<customerinfo xmlns="http://posample.org" Cid="1004">
  <name>Matt Foreman</name>
  <addr country="Canada">
    <street>1596 Baseline</street>
    <city>Toronto</city>
    <prov-state>Ontario</prov-state>
    <pcode-zip>M3Z 5H9</pcode-zip>
  </addr>
  <phone type="work">905-555-4789</phone>
  <phone>416-555-3376</phone>
  <assistant><name>Gopher Runner</name>
    <phone>416-555-3426</phone>
  </assistant>
</customerinfo>
```

この式は、顧客の電話番号とアシスタントの電話番号の両方から、type 属性を削除します。

挿入式

挿入式は、ノード・シーケンス内の指定された位置に、1 つ以上のノードのコピーを挿入します。

構文



do insert
挿入式を開始するためのキーワード。

SourceExpression
更新式以外の XQuery 式。式に最上位のコンマ演算子が含まれる場合、式を括弧で囲む必要があります。式は、あたかもエレメント・コンストラクター内の括弧で囲まれた式であるかのように評価されます。 *SourceExpression* の結果は、挿入される 0 個以上のノードのシーケンスであり、これを*挿入シーケンス* と呼びます。挿入シーケンスに文書ノードが含まれる場合、挿入シーケンス内のその文書ノードは、子に置き換えられます。

挿入シーケンスの最初に属性ノードがある場合、その属性は、指定したキーワードに応じて、*TargetExpression* のノードまたはその親に追加されます。属性ノード以外のノードの次に属性ノードがある挿入シーケンスの場合、DB2 XQuery はエラーを戻します。

before
SourceExpression のノードが *TargetExpression* のノードより前に来る兄弟になることを指定するキーワード。

SourceExpression のノードは、*TargetExpression* のノードの直前に挿入されます。複数のノードが *TargetExpression* の前に挿入される場合は、非決定論的順序で挿入されますが、挿入されたノードのセットは *TargetExpression* の直前に現れます。挿入シーケンスの最初に属性ノードがある場合、属性ノードはターゲット・ノードの親の属性になります。

after
SourceExpression のノードが *TargetExpression* のノードより後に来る兄弟になることを指定するキーワード。

SourceExpression のノードは、*TargetExpression* のノードの直後に挿入されます。複数のノードが *TargetExpression* の後に挿入される場合は、非決定論的順序で挿入されますが、挿入されたノードのセットは *TargetExpression* の直後に現れます。挿入シーケンスの最初に属性ノードがある場合、属性ノードはターゲット・ノードの親の属性になります。

as first into
SourceExpression のノードが *TargetExpression* のノードの最初の子になることを指定するキーワード。

複数のノードが *TargetExpression* のノードの最初の子として挿入される場合は、非決定論的順序で挿入されますが、挿入されたノードのセットは *TargetExpression* の最初の子として現れます。挿入シーケンスの最初に属性ノードがある場合、属性ノードはターゲット・ノードの属性になります。

as last into
SourceExpression のノードが *TargetExpression* のノードの最後の子になることを指定するキーワード。

複数のノードが *TargetExpression* のノードの最後の子として挿入される場合は、非決定論的順序で挿入されますが、挿入されたノードのセットは *TargetExpression* のノードの最後の子として現れます。挿入シーケンスの最初に属性ノードがある場合、属性ノードはターゲット・ノードの属性になります。

into
SourceExpression のノードが非決定論的順序で *TargetExpression* のノードの子になることを指定するキーワード。

SourceExpression のノードは、非決定論的順序で *TargetExpression* のノードの子として挿入されます。挿入シーケンスの最初に属性ノードがある場合、属性ノードはターゲット・ノードの属性になります。

TargetExpression
更新式以外の XQuery 式。式に最上位のコンマ演算子が含まれる場合、式を括弧で囲む必要があります。*TargetExpression* の前に指定されるキーワードに応じて、以下の規則が適用されます。

- **before** または **after** を指定する場合、*TargetExpression* の結果は、エレメント、テキスト、処理命令、またはコメント・ノードである必要があり、いずれの場合も親プロパティーが空であってはなりません。 *TargetExpression* のノードの親が文書ノードのときに **before** または **after** を指定する場合は、挿入シーケンスに属性ノードを含めることはできません。
- **into**、**as first into**、または **as last into** を指定する場合、*TargetExpression* の結果は単一エレメント・ノードまたは単一文書ノードでなければなりません。
- *TargetExpression* が文書ノードのときに **into** を指定する場合は、挿入シーケンスに属性ノードを含めることはできません。

例

以下の例では、DB2 SAMPLE データベースの CUSTOMER 表を使用します。CUSTOMER 表の XML 列 INFO には、顧客の住所と電話に関する情報が含まれています。

以下の例では、変換式の **copy** 節が INFO 列から XML 文書のコピーを作成します。挿入式が、`billto` エレメントとそのすべての子を、最後の `phone` エレメントの後に挿入します。

```
xquery
declare default element namespace 'http://posample.org';
   transform
   copy $mycust := db2-fn:sqlquery('select info from customer where cid = 1004')
   modify
     do insert
       <billto country="Canada">
          <street>4441 Wagner</street>
          <city>Aurora</city>
          <prov-state>Ontario</prov-state>
          <pcode-zip>N8X 7F8</pcode-zip>
       </billto>
    after $mycust/customerinfo/phone[last()]
   return $mycust
```

SAMPLE データベースに対して実行すると、この式は以下の結果を戻します。

```
<customerinfo xmlns="http://posample.org" Cid="1004">
  <name>Matt Foreman</name>
  <addr country="Canada">
    <street>1596 Baseline</street>
    <city>Toronto</city>
    <prov-state>Ontario</prov-state>
    <pcode-zip>M3Z 5H9</pcode-zip>
  </addr>
  <phone type="work">905-555-4789</phone>
  <phone type="home">416-555-3376</phone>
  <billto country="Canada">
    <street>4441 Wagner</street>
```

```
    <city>Aurora</city>
    <prov-state>Ontario</prov-state>
    <pcode-zip>N8X 7F8</pcode-zip>
  </billto>
  <assistant>
    <name>Gopher Runner</name>
    <phone type="home">416-555-3426</phone>
  </assistant>
</customerinfo>
```

以下の例では、値が x2334 の属性 extension を、電話の type 属性が work の phone エレメントに挿入します。

```
xquery
let $phoneext :=  attribute extension { "x2334" }
return
  transform
  copy $mycust := db2-fn:sqlquery('select info from customer where cid = 1004')
  modify
    do insert  $phoneext into $mycust/*:customerinfo/*:phone[@type="work"]
  return $mycust
```

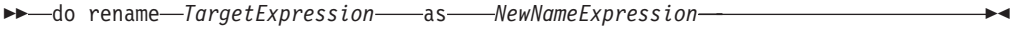
SAMPLE データベースに対して実行すると、この式は以下の結果を戻します。

```
<customerinfo xmlns="http://posample.org" Cid="1004">
  <name>Matt Foreman</name>
    <addr country="Canada">
      <street>1596 Baseline</street>
      <city>Toronto</city>
      <prov-state>Ontario</prov-state>
      <pcode-zip>M3Z 5H9</pcode-zip>
    </addr>
  <phone extension="x2334" type="work">905-555-4789</phone>
  <phone type="home">416-555-3376</phone>
  <assistant>
    <name>Gopher Runner</name>
    <phone type="home">416-555-3426</phone>
  </assistant>
</customerinfo>
```

名前変更式

名前変更式は、データ・モデル・ノードの名前プロパティーを、新しい QName に置き換えます。

構文



do rename
名前変更式を開始するためのキーワード。

TargetExpression
更新式以外の XQuery 式。*TargetExpression* の結果は、単一のエレメント・ノード、属性ノード、または処理命令ノードでなければなりません。式に最上位のコンマ演算子が含まれる場合、式を括弧で囲む必要があります。

名前変更式は、*TargetExpression* のノードにのみ影響を及ぼします。
TargetExpression のノードがエレメント・ノードである場合、式はターゲット・

ノードの属性や子に影響を及ぼしません。 *TargetExpression* のノードが属性ノードである場合、式は *TargetExpression* のノードの他の属性や親ノードの他の子孫に影響を及ぼしません。

as 新しい名前の式を開始するためのキーワード。

NewNameExpression
更新式以外の XQuery 式。*NewNameExpression* の結果は、xs:string、xs:QName、xs:untypedAtomic タイプの値、またはそのような値を原子化処理で抽出できるノードでなければなりません。式に最上位のコンマ演算子が含まれる場合、式を括弧で囲む必要があります。

結果値は QName に変換され、ネーム・スペース接頭部がある場合は静的に既知のネーム・スペースに基づいて解決されます。結果は、エラーまたは拡張された QName のどちらかになります。*TargetExpression* のノードの名前は、拡張された QName に置き換えられます。

新しい QName に含まれる接頭部が同じでも URI が *TargetExpression* のノードのスコープ内ネーム・スペースと異なる場合、DB2 XQuery はエラーを戻します。

例

以下の例では、DB2 SAMPLE データベースの CUSTOMER 表を使用します。CUSTOMER 表の XML 列 INFO には、顧客の住所と電話に関する情報が含まれています。

以下の例では、変換式の **copy** 節が INFO 列から XML 文書のコピーを作成します。この名前変更式は、`addr` エレメントの名前プロパティーを `shipto` に変更します。

```
xquery
declare default element namespace 'http://posample.org';
transform
copy $mycust := db2-fn:sqlquery('select info from customer where cid = 1000')
modify
  do rename $mycust/customerinfo/addr as "shipto"
return $mycust
```

SAMPLE データベースに対して実行すると、この式は以下の結果を戻します。

```
<customerinfo xmlns="http://posample.org" Cid="1000">
  <name>Kathy Smith</name>
  <shipto country="Canada">
    <street>5 Rosewood</street>
    <city>Toronto</city>
    <prov-state>Ontario</prov-state>
    <pcode-zip>M6W 1E6</pcode-zip>
  </shipto>
  <phone type="work">416-555-1358</phone>
</customerinfo>
```

以下の例では、変換式の **modify** 節に FLWOR 式と名前変更式が含まれています。この名前変更式は、`phone` エレメントのすべてのインスタンスの名前プロパティーを `phonenumber` に変更します。

```
xquery
declare default element namespace 'http://posample.org';
transform
copy $mycust :=  db2-fn:sqlquery('select info from customer where cid = 1003')
```

```
modify
  for $phone in $mycust/customerinfo/phone
  return
    do rename $phone as "phonenumber"
return $mycust
```

SAMPLE データベースに対して実行すると、この式は以下の結果を戻します。

```
<customerinfo xmlns="http://posample.org" Cid="1003">
  <name>Robert Shoemaker</name>
  <addr country="Canada">
    <street>1596 Baseline</street>
    <city>Aurora</city>
    <prov-state>Ontario</prov-state>
    <pcode-zip>N8X 7F8</pcode-zip>
  </addr>
  <phonenumber type="work">905-555-7258</phonenumber>
  <phonenumber type="home">416-555-2937</phonenumber>
  <phonenumber type="cell">905-555-8743</phonenumber>
  <phonenumber type="cottage">613-555-3278</phonenumber>
</customerinfo>
```

以下の例では、名前変更式は addr エレメントの属性の名前を、country から geography に変更します。

```
xquery
declare default element namespace 'http://posample.org';
transform
copy $mycust := db2-fn:sqlquery('select info from customer where cid = 1000')
modify
  do rename $mycust/customerinfo/addr/@country as "geography"
return $mycust
```

SAMPLE データベースに対して実行すると、この式は以下の結果を戻します。

```
<customerinfo xmlns="http://posample.org" Cid="1000">
  <name>Kathy Smith</name>
  <addr geography="Canada">
    <street>5 Rosewood</street>
    <city>Toronto</city>
    <prov-state>Ontario</prov-state>
    <pcode-zip>M6W 1E6</pcode-zip>
  </addr>
  <phone type="work">416-555-1358</phone>
</customerinfo>
```

以下の例では、名前変更式と fn:QName 関数を使用して、顧客の勤務先以外の電話番号のエレメントと属性の名前に、ネーム・スペース接頭部 other を追加します。接頭部 other は、URI http://otherphone.com にバインドされます。

```
xquery
declare default element namespace 'http://posample.org';
transform
copy $mycust := db2-fn:sqlquery('select info from customer where cid = 1004')
modify
  for $elem in $mycust/customerinfo/phone[@type != "work"]
  let $elemLocalName := fn:local-name($elem)
  let $newElemQName := fn:QName("http://otherphone.com", fn:concat("other:",
     $elemLocalName))
  return
    ( do rename $elem as $newElemQName,
    for $a in $elem/@* let $attrlocalname := fn:local-name($a)
    let $newAttrName := fn:QName("http://otherphone.com", fn:concat("other:",
```

```
       $attrlocalname))
    return
      do rename $a as $newAttrName )
return $mycust
```

SAMPLE データベースに対して実行すると、この式は以下の結果を戻します。

```
<customerinfo xmlns="http://posample.org" Cid="1004">
  <name>Matt Foreman</name>
  <addr country="Canada">
    <street>1596 Baseline</street>
    <city>Toronto</city>
    <prov-state>Ontario</prov-state>
    <pcode-zip>M3Z 5H9</pcode-zip>
  </addr>
  <phone type="work">905-555-4789</phone>
  <other:phone xmlns:other="http://otherphone.com" other:type="home">
    416-555-3376</other:phone>
  <assistant>
    <name>Gopher Runner</name>
    <phone type="home">416-555-3426</phone>
  </assistant>
</customerinfo>
```

以下の式を変換式の **return** 節で使用すると、デフォルト・エレメント・ネーム・スペースを使用する phone エレメント・ノードが primary ノードの子ノードとして現れ、other:phone エレメント・ノードが secondary ノードの子ノードとして現れます。

```
<phonenumbers xmlns:other="http://otherphone.com">
  <primary>
    { $mycust//phone }
  </primary>
  <secondary>
    { $mycust//other:phone }
  </secondary>
</phonenumbers>
```

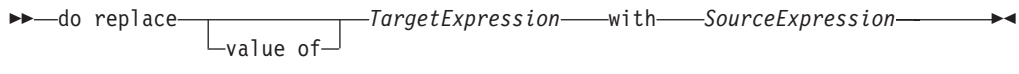
SAMPLE データベースに対して実行すると、変換式は以下の結果を戻します。

```
<phonenumbers xmlns:other="http://otherphone.com" xmlns="http://posample.org">
  <primary>
    <phone type="work">905-555-4789</phone>
    <phone type="home">416-555-3426</phone>
  </primary>
  <secondary>
    <other:phone other:type="home">416-555-3376</other:phone>
  </secondary>
</phonenumbers>
```

置換式

置換式は、既存のノードを 0 個以上のノードの新しいシーケンスに置き換えるか、ノードの値をノードの ID を保持したまま置き換えます。

構文



do replace
置換式を開始するためのキーワード。

TargetExpression
更新式以外の XQuery 式。式に最上位のコンマ演算子が含まれる場合、式を括弧で囲む必要があります。*TargetExpression* の結果は、文書ノード以外の単一ノードでなければなりません。 *TargetExpression* の結果が文書ノードの場合、DB2 XQuery はエラーを戻します。

value of キーワードを指定しない場合、*TargetExpression* の結果は、空でない親プロパティーを持つ単一ノードでなければなりません。

value of
TargetExpression のノードの値をノードの ID を保持したまま置き換えることを指定するためのキーワード。

with
ソース式を開始するためのキーワード。

SourceExpression
更新式以外の XQuery 式。式に最上位のコンマ演算子が含まれる場合、式を括弧で囲む必要があります。

value of キーワードを指定した場合、*SourceExpression* は、あたかもテキスト・ノード・コンストラクターの内容式であるかのように評価されます。
SourceExpression の結果は、単一テキスト・ノードまたは空のシーケンスです。

value of キーワードを指定しない場合、*SourceExpression* の結果はノードのシーケンスでなければなりません。 *SourceExpression* は、あたかもエレメント・コンストラクター内の括弧で囲まれた式であるかのように評価されます。
SourceExpression シーケンスに文書ノードが含まれる場合、その文書ノードは子に置き換えられます。 *SourceExpression* シーケンスは、以下のノード・タイプで構成されている必要があります。

- *TargetExpression* のノードが属性ノードである場合、置換シーケンスは 0 個以上の属性ノードで構成されている必要があります。
- *TargetExpression* のノードがエレメント・ノード、テキスト・ノード、コメント・ノード、または処理命令ノードである場合、置換シーケンスは、0 個以上のエレメント・ノード、テキスト・ノード、コメント・ノード、または処理命令ノードの組み合わせで構成されている必要があります。

value of キーワードを指定した場合は、以下の更新が生成されます。

- *TargetExpression* のノードがエレメント・ノードである場合、*TargetExpression* のノードの既存の子は、*SourceExpression* によって戻されるテキスト・ノードに置き換えられます。 *SourceExpression* が空のシーケンスを戻した場合、*TargetExpression* のノードの子プロパティーは空になります。 *TargetExpression* のノードに属性ノードが含まれる場合、その属性ノードは影響を受けません。
- *TargetExpression* のノードがエレメント・ノードでない場合、*TargetExpression* のノードのストリング値は、*SourceExpression* によって戻されるテキスト・ノードのストリング値に置き換えられます。 *SourceExpression* がテキスト・ノードを戻さない場合、*TargetExpression* のノードのストリング値は、ゼロ長ストリングに置き換えられます。

value of キーワードを指定しない場合は、以下の更新が生成されます。

- *TargetExpression* のノードが *SourceExpression* のノードに置き換えられます。*TargetExpression* のノードの親ノードは、*SourceExpression* の各ノードの親になります。*TargetExpression* のノードがあったノード階層内の位置に、*SourceExpression* のノードが置かれます。
- *TargetExpression* のノードとそのすべての属性および子孫は、ノード・シーケンスからデタッチされます。

例

以下の例では、DB2 SAMPLE データベースの CUSTOMER 表を使用します。CUSTOMER 表の XML 列 INFO には、顧客の住所と電話に関する情報が含まれています。

以下の例では、変換式の **copy** 節が INFO 列から XML 文書のコピーを作成します。この置換式は、addr エレメントとその子を置き換えます。

```
xquery
declare default element namespace 'http://posample.org';
transform
copy $mycust := db2-fn:sqlquery('select info from customer where cid = 1000')
modify
  do replace $mycust/customerinfo/addr
    with
      <addr country="Canada">
        <street>1596 14th Avenue NW</street>
        <city>Calgary</city>
        <prov-state>Alberta</prov-state>
        <pcode-zip>T2N 1M7</pcode-zip>
      </addr>
return $mycust
```

SAMPLE データベースに対して実行すると、この式は、住所情報が置き換えられた以下の結果を戻します。

```
<customerinfo xmlns="http://posample.org" Cid="1000">
  <name>Kathy Smith</name>
  <addr country="Canada">
    <street>1596 14th Avenue NW</street>
    <city>Calgary</city>
    <prov-state>Alberta</prov-state>
    <pcode-zip>T2N 1M7</pcode-zip>
  </addr>
  <phone type="work">416-555-1358</phone>
</customerinfo>
```

以下の式は、顧客の phone エレメントの type 属性の値を、home から personal に置き換えます。

```
xquery
declare default element namespace 'http://posample.org';
transform
copy $mycust := db2-fn:sqlquery('select info from customer where cid = 1004')
modify
  do replace value of $mycust/customerinfo/phone[@type="home"]/@type with "personal"
return $mycust
```

SAMPLE データベースに対して実行すると、この式は、属性値が置き換えられた以下の結果を戻します。

```
<customerinfo xmlns="http://posample.org" Cid="1004">
  <name>Matt Foreman</name>
  <addr country="Canada">
```

```
    <street>1596 Baseline</street>
    <city>Toronto</city>
    <prov-state>Ontario</prov-state>
    <pcode-zip>M3Z 5H9</pcode-zip>
  </addr>
  <phone type="work">905-555-4789</phone>
  <phone type="personal">416-555-3376</phone>
  <assistant>
    <name>Gopher Runner</name>
    <phone type="home">416-555-3426</phone>
  </assistant>
</customerinfo>
```

アシスタントの phone の属性値は、変更されていません。

第 5 章 組み込み関数

DB2 XQuery は、XML データを使用するために組み込み関数のライブラリーを提供します。これらの組み込み関数には、XQuery の定義済み関数および DB2 の定義済み関数が含まれます。

XQuery の定義済み関数

XQuery の定義済み関数は、接頭部 `fn` にバインドされるネーム・スペースにあります。このネーム・スペースは、デフォルトの関数ネーム・スペースであり、これはネーム・スペース接頭部を指定しなくても XQuery の定義済み関数を呼び出すことができることを意味します。このデフォルトの関数ネーム・スペースを、照会プロローグのデフォルトの関数ネーム・スペース宣言でオーバーライドする場合、接頭部 `fn` を使用して、XQuery の定義済み関数を呼び出す必要があります。

DB2 の定義済み関数

DB2 の定義済み関数は、db2-fn:xmlcolumn および db2-fn:sqlquery です。これらは、DB2 データベースにある XML 値にアクセスする場合に使用します。接頭部 `db2-fn` は、デフォルトの関数ネーム・スペースではないため、デフォルトのネーム・スペースを照会プロローグのデフォルトの関数ネーム・スペース宣言でオーバーライドしない限り、これらの関数を呼び出す際にネーム・スペース接頭部を使用する必要があります。

カテゴリー別の関数

関数には、以下のカテゴリー (ストリング、ブール、数値、日付と時間、シーケンス、QName、ノード、その他) があります。

ストリング関数

関数	説明
152 ページの『codepoints-to-string 関数』	fn:codepoints-to-string 関数は、Unicode コード・ポイントのシーケンスに相当するストリングを戻します。
153 ページの『compare 関数』	fn:compare 関数は、2 つのストリングを比較します。
154 ページの『concat 関数』	fn:concat 関数は、2 つ以上の原子値を連結したストリングを戻します。
154 ページの『contains 関数』	fn:contains 関数は、ストリングに、指定されたサブストリングが含まれるかどうかを判別します。サブストリングはデフォルトの照合を使用して突き合わされます。
163 ページの『ends-with 関数』	fn:ends-with 関数は、ストリングが、指定されたサブストリングで終了しているかどうかを判別します。サブストリングはデフォルトの照合を使用して突き合わされます。

© IBM Corporation 2006, 2008

関数	説明
173 ページの『lower-case 関数』	fn:lower-case 関数は、ストリングを小文字に変換します。
174 ページの『matches 関数』	fn:matches 関数は、ストリングが、指定されたパターンに一致するかどうかを判別します。
185 ページの『normalize-space 関数』	fn:normalize-space 関数は、ストリングから前後の空白文字を除去し、内部の連続した空白文字をそれぞれ単一のブランク文字に置き換えます。
185 ページの『normalize-unicode 関数』	fn:normalize-unicode 関数は、ストリングに対して Unicode 正規化を実行します。
190 ページの『replace 関数』	fn:replace 関数は、ストリング内の文字の各セットを指定されたパターンと比較し、パターンに一致する文字を別の文字のセットに置き換えます。
201 ページの『starts-with 関数』	fn:starts-with 関数は、ストリングが、指定されたサブストリングで開始しているかどうかを判別します。サブストリングはデフォルトの照合を使用して突き合わされます。
202 ページの『string 関数』	fn:string 関数は、値のストリング表記を戻します。
203 ページの『string-join 関数』	fn:string-join 関数は、分離文字で分離された項目を連結させることによって生成されるストリングを戻します。
203 ページの『string-length 関数』	fn:string-length 関数は、ストリングの長さを戻します。
204 ページの『string-to-codepoints 関数』	fn:string-to-codepoints 関数は、ストリング値に相当する Unicode コード・ポイントのシーケンスを戻します。
205 ページの『substring 関数』	fn:substring 関数は、ストリングのサブストリングを戻します。
206 ページの『substring-after 関数』	fn:substring-after 関数は、指定された検索ストリングの最初の出現の終了後にストリング内に出現するサブストリングを戻します。検索ストリングはデフォルトの照合を使用して突き合わされます。
207 ページの『substring-before 関数』	fn:substring-before 関数は、指定された検索ストリングが最初に出現する前にストリング内に出現するサブストリングを戻します。検索ストリングはデフォルトの照合を使用して突き合わされます。
211 ページの『tokenize 関数』	fn:tokenize 関数は、ストリングを切断して、サブストリングのシーケンスを作成します。
212 ページの『translate 関数』	fn:translate 関数は、ストリング内の選択された文字を置換文字に置き換えます。

関数	説明
214 ページの『upper-case 関数』	fn:upper-case 関数は、ストリングを大文字に変換します。

ブール関数

関数	説明
151 ページの『boolean 関数』	fn:boolean 関数は、シーケンスの有効なブール値を戻します。
165 ページの『false 関数』	fn:false 関数は、xs:boolean 値 false を戻します。
186 ページの『not 関数』	fn:not 関数は、シーケンスの有効なブール値が true の場合は false を、シーケンスの有効なブール値が false の場合は true を戻します。
214 ページの『true 関数』	fn:true 関数は、xs:boolean 値 true を戻します。
219 ページの『zero-or-one 関数』	fn:zero-or-one 関数は、引数に 1 つの項目が含まれるか、引数が空のシーケンスである場合に、その引数を戻します。

数値関数

関数	説明
149 ページの『abs 関数』	fn:abs 関数は、数値の絶対値を戻します。
150 ページの『avg 関数』	fn:avg 関数は、シーケンス内の値の平均を戻します。
152 ページの『ceiling 関数』	fn:ceiling 関数は、指定された数値以上の最小整数値を戻します。
165 ページの『floor 関数』	fn:floor 関数は、指定された数値以下の最大整数値を戻します。
175 ページの『max 関数』	fn:max 関数は、シーケンス内の最大値を戻します。
176 ページの『min 関数』	fn:min 関数は、シーケンス内の最小値を戻します。
187 ページの『number 関数』	fn:number 関数は、値を xs:double タイプに変換します。
194 ページの『round 関数』	fn:round 関数は、指定された数値に最も近い整数を戻します。
195 ページの『round-half-to-even 関数』	fn:round-half-to-even 関数は、指定された数値に最も近い、指定された精度の数値を戻します。
208 ページの『sum 関数』	fn:sum 関数は、シーケンス内の値の合計を戻します。

日付、時間、および期間関数

関数	説明
143 ページの『adjust-date-to-timezone 関数』	fn:adjust-time-to-timezone 関数は、xs:time 値を特定のタイム・ゾーンに調整するか、値からタイム・ゾーン・コンポーネントを除去します。
145 ページの『adjust-dateTime-to-timezone 関数』	fn:adjust-dateTime-to-timezone 関数は、xs:dateTime 値を特定のタイム・ゾーンに調整するか、値からタイム・ゾーン・コンポーネントを除去します。
147 ページの『adjust-time-to-timezone 関数』	fn:adjust-time-to-timezone 関数は、xs:time 値を特定のタイム・ゾーンに調整するか、値からタイム・ゾーン・コンポーネントを除去します。
155 ページの『current-date 関数』	fn:current-date 関数は、暗黙指定された UTC の時間帯で現在日付を戻します。
156 ページの『current-dateTime 関数』	fn:current-dateTime 関数は、暗黙指定された UTC の時間帯で現在日時を戻します。
156 ページの『current-time 関数』	fn:current-time 関数は、暗黙指定された UTC の時間帯で現在時刻を戻します。
157 ページの『dateTime 関数』	fn:dateTime 関数は、xs:date 値および xs:time 値から、xs:dateTime 値を構成します。
158 ページの『day-from-date 関数』	fn:day-from-date 関数は xs:date 値の日付コンポーネントを戻します。
159 ページの『day-from-dateTime 関数』	fn:day-from-dateTime 関数は、xs:dateTime 値の日付コンポーネントを戻します。
159 ページの『days-from-duration 関数』	fn:days-from-duration 関数は、期間の日付コンポーネントを戻します。
166 ページの『hours-from-dateTime 関数』	fn:hours-from-dateTime 関数は、xs:dateTime 値の時間コンポーネントを戻します。
167 ページの『hours-from-duration 関数』	fn:hours-from-duration 関数は、期間値の時間コンポーネントを戻します。
168 ページの『hours-from-time 関数』	fn:hours-from-time 関数は、xs:time 値の時間コンポーネントを戻します。
168 ページの『implicit-timezone 関数』	fn:implicit-timezone 関数は、暗黙指定された時間帯値 PT0S を戻します。これは xs:dayTimeDuration タイプです。値 PT0S は、UTC が暗黙指定された時間帯であることを示します。
178 ページの『minutes-from-dateTime 関数』	fn:minutes-from-dateTime 関数は、xs:dateTime 値の分コンポーネントを戻します。
178 ページの『minutes-from-duration 関数』	fn:minutes-from-duration 関数は、期間の分コンポーネントを戻します。
179 ページの『minutes-from-time 関数』	fn:minutes-from-time 関数は、xs:time 値の分コンポーネントを戻します。

関数	説明
179 ページの『month-from-date 関数』	fn:month-from-date 関数は、xs:date 値の月コンポーネントを戻します。
180 ページの『month-from-dateTime 関数』	fn:month-from-dateTime 関数は、xs:dateTime 値の月コンポーネントを戻します。
180 ページの『months-from-duration 関数』	fn:months-from-duration 関数は、期間値の月コンポーネントを戻します。
196 ページの『seconds-from-dateTime 関数』	fn:seconds-from-dateTime 関数は、xs:dateTime 値の秒コンポーネントを戻します。
197 ページの『seconds-from-duration 関数』	fn:seconds-from-duration 関数は、期間の秒コンポーネントを戻します。
198 ページの『seconds-from-time 関数』	fn:seconds-from-time 関数は、xs:time 値の秒コンポーネントを戻します。
209 ページの『timezone-from-date 関数』	fn:timezone-from-date 関数は、xs:date 値のタイム・ゾーン・コンポーネントを戻します。
210 ページの『timezone-from-dateTime 関数』	fn:timezone-from-dateTime 関数は、xs:dateTime 値のタイム・ゾーン・コンポーネントを戻します。
210 ページの『timezone-from-time 関数』	fn:timezone-from-time 関数は、xs:time 値のタイム・ゾーン・コンポーネントを戻します。
217 ページの『year-from-date 関数』	fn:year-from-date 関数は、xs:date 値の年コンポーネントを戻します。
217 ページの『year-from-dateTime 関数』	fn:year-from-dateTime 関数は、xs:dateTime 値の年コンポーネントを戻します。
218 ページの『years-from-duration 関数』	fn:years-from-duration 関数は、期間の年コンポーネントを戻します。

シーケンス関数

関数	説明
155 ページの『count 関数』	fn:count 関数は、シーケンス内の値の個数を戻します。
157 ページの『data 関数』	fn:data 関数は、入力シーケンス内のノードをその型付き値に置き換えた後の入力シーケンスを戻します。
160 ページの『deep-equal 関数』	fn:deep-equal 関数は、2 つのシーケンスを比較して、それらが値同等性の要件を満たすかどうかを判別します。
162 ページの『distinct-values 関数』	fn:distinct-values 関数は、シーケンス内で重複を取り除いた値を戻します。
163 ページの『empty 関数』	fn:empty 関数は、引数が空のシーケンスであるかどうかを示します。
164 ページの『exactly-one 関数』	fn:exactly-one 関数は、引数に正確に 1 つの項目が含まれる場合にその引数を戻します。
165 ページの『exists 関数』	fn:exists 関数は、シーケンスが空のシーケンスでないかどうかを示します。

関数	説明
171 ページの『last 関数』	fn:last 関数は、現在処理中のシーケンス内の値の個数を戻します。
169 ページの『index-of 関数』	fn:index-of 関数は、シーケンス内で項目が現れる位置を戻します。
170 ページの『insert-before 関数』	fn:insert-before 関数は、シーケンスを、別のシーケンスの指定された位置の前に挿入します。
188 ページの『one-or-more 関数』	fn:one-or-more 関数は、引数に 1 つ以上の項目が含まれる場合にその引数を戻します。
188 ページの『position 関数』	fn:position 関数は、現在処理中のシーケンス内のコンテキスト・アイテムの位置を戻します。
189 ページの『remove 関数』	fn:remove 関数は、シーケンスから項目を除去します。
193 ページの『reverse 関数』	fn:reverse 関数は、シーケンス内の項目の順序を逆にします。
205 ページの『subsequence 関数』	fn:subsequence 関数は、シーケンスのサブシーケンスを戻します。
214 ページの『unordered 関数』	fn:unordered 関数は、シーケンス内の項目を非決定論的順序で戻します。

QName 関数

関数	説明
169 ページの『in-scope-prefixes 関数』	fn:in-scope-prefixes 関数は、エレメントのすべての範囲内のネーム・スペースの接頭部のリストを戻します。
172 ページの『local-name-from-QName 関数』	fn:local-name-from-QName 関数は、xs:QName 値のローカル部分を戻します。
183 ページの『namespace-uri-for-prefix 関数』	fn:namespace-uri-for-prefix 関数は、エレメントの範囲内のネーム・スペースの接頭部と関連付けられたネーム・スペース URI を戻します。
184 ページの『namespace-uri-from-QName 関数』	fn:namespace-uri-from-QName 関数は、xs:QName 値のネーム・スペース URI の部分を戻します。
189 ページの『QName 関数』	fn:QName 関数は、ネーム・スペース URI および字句の QName (オプションの接頭部を持つ) を含むストリングから、展開名を作成します。
192 ページの『resolve-QName 関数』	fn:resolve-QName 関数は、エレメントの範囲内のネーム・スペースを使用して、ネーム・スペース接頭部をネーム・スペース URI に解決することにより、字句の QName を含むストリングを展開された QName に変換します。

ノード関数

関数	説明
171 ページの『local-name 関数』	fn:local-name 関数は、ノードのローカル名プロパティーを戻します。
181 ページの『name 関数』	fn:name 関数は、ノード名の接頭部およびローカル名の部分を戻します。
182 ページの『namespace-uri 関数』	fn:namespace-uri 関数は、ノードの修飾名のネーム・スペース URI を戻します。
184 ページの『node-name 関数』	fn:node-name 関数は、ノードの展開された QName を戻します。
193 ページの『root 関数』	fn:root 関数は、ノードが属するツリーのルート・ノードを戻します。

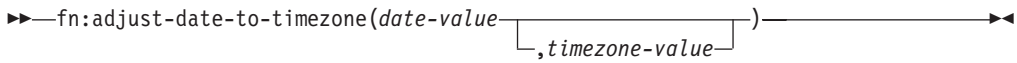
その他の関数

関数	説明
162 ページの『default-collation 関数』	fn:default-collation 関数は、データベースについて定義されているデフォルト照合を表す URI を戻します。
198 ページの『sqlquery 関数』	db2-fn:sqlquery 関数は、現在接続されている DB2 データベースで、 SQL 全選択の結果であるシーケンスを検索します。
216 ページの『xmlcolumn 関数』	db2-fn:xmlcolumn 関数は、現在接続されている DB2 データベース内の列からシーケンスを検索します。

adjust-date-to-timezone 関数

fn:adjust-date-to-timezone 関数は、xs:date 値を特定のタイム・ゾーンに調整するか、値からタイム・ゾーン・コンポーネントを除去します。

構文



date-value

調整対象の日付値。

date-value は xs:date タイプであるか、空のシーケンスです。

timezone-value

date-value が調整されるタイム・ゾーンを表す期間です。

timezone-value は空のシーケンスにするか、-PT14H から PT14H の範囲内の xdt:dayTimeDuration タイプの単一値にすることができます。この値は分数

(整数) であり、秒のコンポーネントにすることはできません。
timezone-value が指定されていない場合、デフォルト値は PT0H で、これは UTC を表します。

戻り値

戻り値は、指定されるパラメーターに応じて、xs:date タイプの値か、空のシーケンスになります。*date-value* が空のシーケンスではない場合、戻り値は xs:date タイプです。戻される可能性のある戻り値を以下の表に示します。

表 32. fn:adjust-date-to-timezone の入力値と戻り値のタイプ

date-value	*timezone-value*	**戻り値**
タイム・ゾーン・コンポーネントを含む *date-value*	明示的な値、または値の指定なし (期間 PT0H)	*timezone-value* によって表されるタイム・ゾーンに調整される *date-value*。
タイム・ゾーン・コンポーネントを含む *date-value*	空のシーケンス	タイム・ゾーン・コンポーネントを含まない *date-value*。
タイム・ゾーン・コンポーネントを含まない *date-value*	明示的な値、または値の指定なし (期間 PT0H)	タイム・ゾーン・コンポーネントを含む *date-value*。タイム・ゾーン・コンポーネントは、*timezone-value* によって表されるタイム・ゾーンです。日付コンポーネントはタイム・ゾーンに調整されません。
タイム・ゾーン・コンポーネントを含まない *date-value*	空のシーケンス	*date-value*。
空のシーケンス	明示的な値、空のシーケンス、または値の指定なし	空のシーケンス。

date-value を異なるタイム・ゾーンに調整する場合、*date-value* は時間コンポーネント 00:00:00 を持つ dateTime として扱われます。戻り値には、*timezone-value* によって表されるタイム・ゾーン・コンポーネントが含まれます。以下の関数は、調整された日付の値を計算します。

```
xs:date(fn:adjust-dateTime-to-timezone(xs:dateTime(date-value),timezone-value))
```

例

以下の例では、変数 `$tz` は -10 時間の期間であり、`xdt:dayTimeDuration("-PT10H")` と定義されます。

以下の関数は、UTC+1 タイム・ゾーンの 2002 年 5 月 7 日の日付値を調整します。関数は *timezone-value* -PT10H を指定します。

```
fn:adjust-date-to-timezone(xs:date("2002-05-07+01:00"), $tz)
```

戻される日付値は 2002-05-06-10:00 です。日付は UTC-10 タイム・ゾーンに調整されます。

以下の関数は、タイム・ゾーン・コンポーネントを持たない、2002 年 3 月 7 日の日付値にタイム・ゾーン・コンポーネントを追加します。関数は *timezone-value* -PT10H を指定します。

```
fn:adjust-date-to-timezone(xs:date("2002-03-07"), $tz)
```

戻り値は 2002-03-07-10:00 です。日付値にタイム・ゾーン・コンポーネントが追加されます。

以下の関数は、UTC-7 タイム・ゾーンの 2002 年 2 月 9 日の日付値を調整します。*timezone-value* が指定されていない場合、関数はデフォルトの *timezone-value* PT0H を使用します。

```
fn:adjust-date-to-timezone(xs:date("2002-02-09-07:00"))
```

戻される日付は 2002-02-09Z であり、日付は UTC に調整されます。

以下に示す関数は、UTC-7 タイム・ゾーンの 2002 年 5 月 7 日の日付値からタイム・ゾーン・コンポーネントを除去します。*timezone-value* は空のシーケンスです。

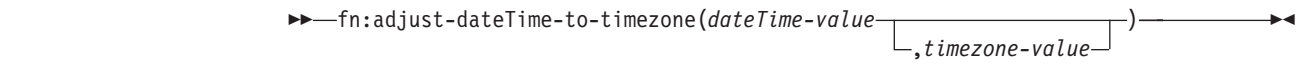
```
fn:adjust-date-to-timezone(xs:date("2002-05-07-07:00"), ())
```

戻り値は 2002-05-07 です。

adjust-dateTime-to-timezone 関数

fn:adjust-dateTime-to-timezone 関数は、xs:dateTime 値を特定のタイム・ゾーンに調整するか、値からタイム・ゾーン・コンポーネントを除去します。

構文



dateTime-value

調整対象の dateTime 値。

dateTime-value は xs:dateTime タイプであるか、空のシーケンスです。

timezone-value

dateTime-value が調整されるタイム・ゾーンを表す期間です。

timezone-value は空のシーケンスにするか、-PT14H から PT14H の範囲内の xdt:dayTimeDuration タイプの単一値にすることができます。この値は分数 (整数) であり、秒のコンポーネントにすることはできません。
timezone-value が指定されていない場合、デフォルト値は PT0H で、これは UTC を表します。

戻り値

戻り値は、入力値のタイプに応じて xs:dateTime タイプの値か、空のシーケンスになります。*dateTime-value* が空のシーケンスではない場合、戻り値は xs:dateTime タイプです。戻される可能性のある戻り値を以下の表に示します。

表 33. fn:adjust-dateTime-to-timezone の入力値と戻り値のタイプ

dateTime-value	*timezone-value*	**戻り値**
タイム・ゾーン・コンポーネントを含む *dateTime-value*	明示的な値、または値の指定なし (期間 PT0H)	*dateTime-value* は、*timezone-value* が表すタイム・ゾーンに調整されます。戻り値には、*timezone-value* によって表されるタイム・ゾーン・コンポーネントが含まれます。
タイム・ゾーン・コンポーネントを含む *dateTime-value*	空のシーケンス	タイム・ゾーン・コンポーネントを含まない *dateTime-value*。
タイム・ゾーン・コンポーネントを含まない *dateTime-value*	明示的な値、または値の指定なし (期間 PT0H)	タイム・ゾーン・コンポーネントを含む *dateTime-value*。タイム・ゾーン・コンポーネントは、*timezone-value* によって表されるタイム・ゾーンです。日付コンポーネントと時間コンポーネントはタイム・ゾーンに調整されません。
タイム・ゾーン・コンポーネントを含まない *dateTime-value*	空のシーケンス	*dateTime-value*。
空のシーケンス	明示的な値、空のシーケンス、または値の指定なし	空のシーケンス。

例

以下の例では、変数 `$tz` は -10 時間の期間であり、`xdt:dayTimeDuration("-PT10H")` と定義されます。

以下の関数は、UTC-7 タイム・ゾーンの 2002 年 3 月 7 日、午前 10 時の dateTime を、*timezone-value* で指定されている -PT10H のタイム・ゾーンに調整します。

```
fn:adjust-dateTime-to-timezone(xs:dateTime("2002-03-07T10:00:00-07:00"), $tz)
```

戻される dateTime は 2002-03-07T07:00:00-10:00 です。

以下の関数は、2002 年 3 月 7 日の午前 10 時の dateTime 値を調整します。*dateTime-value* にはタイム・ゾーン・コンポーネントは含まれず、関数は *timezone-value* -PT10H を指定します。

```
fn:adjust-dateTime-to-timezone(xs:dateTime("2002-03-07T10:00:00"), $tz)
```

戻される dateTime は 2002-03-07T10:00:00-10:00 です。

以下の関数では、UTC-7 タイム・ゾーンの2006 年 6 月 4 日、午前 10 時の dateTime 値を調整します。 *timezone-value* が指定されていない場合、関数はデフォルトのタイム・ゾーン値 PT0H を使用します。

```
fn:adjust-dateTime-to-timezone(xs:dateTime("2006-06-04T10:00:00-07:00"))
```

戻される dateTime は 2006-06-04T17:00:00Z であり、dateTime は UTC に調整されます。

以下に示す関数は、UTC-7 タイム・ゾーンの 2002 年 3 月 7 日、午前 10 時の dateTime 値からタイム・ゾーン・コンポーネントを除去します。*timezone-value* 値は空のシーケンスです。

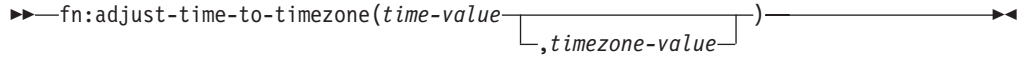
```
fn:adjust-dateTime-to-timezone(xs:dateTime("2002-03-07T10:00:00-07:00"), ())
```

戻される dateTime は 2002-03-07T10:00:00 です。

adjust-time-to-timezone 関数

fn:adjust-time-to-timezone 関数は、xs:time 値を特定のタイム・ゾーンに調整するか、値からタイム・ゾーン・コンポーネントを除去します。

構文



time-value

調整する時間値。

time-value は xs:time タイプであるか、空のシーケンスです。

timezone-value

time-value が調整されるタイム・ゾーンを表す期間です。

timezone-value は空のシーケンスにするか、-PT14H から PT14H の範囲内の xdt:dayTimeDuration タイプの単一値にすることができます。この値は分数 (整数) であり、秒のコンポーネントにすることはできません。
timezone-value が指定されていない場合、デフォルト値は PT0H で、これは UTC を表します。

戻り値

戻り値は、指定されるパラメーターに応じて、xs:time タイプの値か、空のシーケンスになります。*time-value* が空のシーケンスではない場合、戻り値は xs:time タイプです。戻される可能性のある戻り値を以下の表に示します。

表 34. fn:adjust-time-to-timezone の入力値と戻り値のタイプ

date-value	*timezone-value*	**戻り値**
タイム・ゾーン・コンポーネントを含む *time-value*	明示的な値、または値の指定なし (期間 PT0H)	*timezone-value* によって表されるタイム・ゾーンに調整される *time-value*。戻り値には、*timezone-value* によって表されるタイム・ゾーン・コンポーネントが含まれます。タイム・ゾーンの調整が夜中の 12 時をまたぐ場合、日付の変更は無視されます。
タイム・ゾーン・コンポーネントを含む *time-value*	空のシーケンス	タイム・ゾーン・コンポーネントを含まない *time-value*。
タイム・ゾーン・コンポーネントを含まない *time-value*	明示的な値、または値の指定なし (期間 PT0H)	タイム・ゾーン・コンポーネントを含む *time-value*。タイム・ゾーン・コンポーネントは、*timezone-value* によって表されるタイム・ゾーンです。時刻コンポーネントはタイム・ゾーンに調整されません。
タイム・ゾーン・コンポーネントを含まない *time-value*	空のシーケンス	*time-value*。
空のシーケンス	明示的な値、空のシーケンス、または値の指定なし	空のシーケンス。

例

以下の例では、変数 `$tz` は -10 時間の期間であり、`xdt:dayTimeDuration("-PT10H")` と定義されます。

以下の関数は、UTC-7 タイム・ゾーンの午前 10 時の時刻値を調整し、*timezone-value* -PT10H を指定します。

```
fn:adjust-time-to-timezone(xs:time("10:00:00-07:00"), $tz)
```

戻り値は 7:00:00-10:00 です。時刻は、期間 -PT10H によって表されるタイム・ゾーンに調整されます。

以下の関数では、午後 1 時の時刻値を調整します。時刻値にはタイム・ゾーン・コンポーネントが含まれていません。

```
fn:adjust-time-to-timezone(xs:time("13:00:00"), $tz)
```

戻り値は 13:00:00-10:00 です。時刻には、期間 -PT10H によって表されるタイム・ゾーン・コンポーネントが含まれます。

以下の関数は、UTC-7 タイム・ゾーンの午前 10 時の時刻値を調整します。関数は *timezone-value* を指定せず、デフォルト値 PT0H を使用します。

```
fn:adjust-time-to-timezone(xs:time("10:00:00-07:00"))
```

戻り値は 17:00:00Z であり、時間は UTC に調整されます。

以下の関数は、UTC-7 タイム・ゾーンの午前 8 時の時刻値からタイム・ゾーン・コンポーネントを除去します。*timezone-value* は空のシーケンスです。

```
fn:adjust-time-to-timezone(xs:time("08:00:00-07:00"), ())
```

戻り値は 8:00:00 です。

以下の例は、2 つの時刻を比較しています。タイム・ゾーンの調整が夜中の 12 時をまたぐ場合、日付の変更が生じます。しかし、fn:adjust-time-to-timezone では日付の変更は無視します。

```
fn:adjust-time-to-timezone(xs:time("01:00:00+14:00"), $tz)
   eq xs:time("01:00:00-10:00")
```

戻り値は true です。

abs 関数

fn:abs 関数は、数値の絶対値を戻します。

構文

numeric-value

原子値または空のシーケンス。

numeric-value が原子値である場合、以下のいずれかのタイプを持ちます。

- xs:float
- xs:double
- xs:decimal
- xs:integer
- 上記のいずれかのタイプから派生したタイプ
- xdt:untypedAtomic

numeric-value が xdt:untypedAtomic タイプである場合、xs:double 値に変換されます。

戻り値

numeric-value が空のシーケンスでない場合、戻り値は、*numeric-value* の絶対値です。

numeric-value が空のシーケンスの場合、fn:abs は空のシーケンスを戻します。

戻り値のデータ・タイプは、*numeric-value* のデータ・タイプによって異なります。

- *numeric-value* が xs:float、xs:double、xs:decimal、または xs:integer である場合、戻される値は *numeric-value* と同じタイプになります。
- *numeric-value* が xs:float、xs:double、xs:decimal、または xs:integer から派生したデータ・タイプを持つ場合、戻される値は *numeric-value* の直接の親のデータ・タイプです。

- *numeric-value* が xdt:untypedAtomic タイプを持つ場合、戻される値は、xs:double タイプを持ちます。

例

以下の関数は、-10.5 の絶対値を戻します。

```
fn:abs(-10.5)
```

戻り値は、10.5 です。

avg 関数

fn:avg 関数は、シーケンス内の値の平均を戻します。

構文

sequence-expression

以下の原子タイプのいずれかの項目を含むシーケンス、または空のシーケンス。

- xs:float
- xs:double
- xs:decimal
- xs:integer
- xdt:untypedAtomic
- xdt:dayTimeDuration
- xdt:yearMonthDuration
- 上記のいずれかのタイプから派生したタイプ

xdt:untypedAtomic タイプの入力項目は、xs:double にキャストされます。このキャスト後、入力シーケンスに含まれるすべての項目は、プロモーションまたはサブタイプの置換によって共通のタイプに変換可能である必要があります。平均は、この共通タイプで計算されます。例えば、入力シーケンスに xs:decimal から派生した money タイプ、および xs:float から派生した stockprice タイプの項目が含まれている場合、平均は xs:float タイプで計算されます。

戻り値

sequence-expression が空のシーケンスでない場合、戻り値は、*sequence-expression* の値の平均です。戻り値のデータ・タイプは、*sequence-expression* の項目のデータ・タイプ、または *sequence-expression* の項目がプロモートされるデータ・タイプと同じです。

sequence-expression が空のシーケンスである場合、空のシーケンスが戻されます。

例

以下の関数は、シーケンス (5, 1.0E2, 40.5) の平均を戻します。

```
fn:avg((5, 1.0E2, 40.5))
```

値は xs:double タイプにプロモートされます。関数は、xs:double 値 4.85E1 を戻し、これは「48.5」としてシリアライズされます。

boolean 関数

fn:boolean 関数は、シーケンスの有効なブール値を戻します。

構文

sequence-expression
任意のタイプの項目を含むシーケンス、または空のシーケンス。

戻り値

戻される有効なブール値 (EBV) は、*sequence-expression* の値によって異なります。

表 35. XQuery において値の特定のタイプに戻される EBV

値の説明	戻される EBV
空のシーケンス	false
1 番目の項目がノードのシーケンス	true
xs:boolean タイプの単一値 (または xs:boolean から派生したタイプの単一値)	false - xs:boolean 値が false の場合
	true - xs:boolean 値が true の場合
xs:string タイプまたは xdt:untypedAtomic タイプの単一値 (またはこの 2 つのタイプのいずれかから派生したタイプの単一値)	false - 値の長さがゼロの場合
	true - 値がゼロより大きい長さの場合
任意の数値の単一値 (または数値タイプから派生した単一値)	false - 値が NaN または数的にゼロと等しい場合
	true - 値が数的にゼロと等しくない場合
その他すべての値	エラー

注: 少なくとも 1 個のノードおよび原子値を含むシーケンスの有効なブール値は、順序が予測不能な照会では非決定論的になります。

例

単一の数値の引数を使用した例: 以下の関数は有効なブール値 0 を戻します。

```
fn:boolean(0)
```

戻り値は false です。

複数項目のシーケンスの引数を使用した例: 次の関数は有効なブール値 (<a/>, 0, <b/>) を戻します。

```
fn:boolean((<a/>, 0, <b/>))
```

戻り値は true です。

ceiling 関数

fn:ceiling 関数は、指定された数値以上の最小整数値を戻します。

構文

numeric-value

原子値または空のシーケンス。

numeric-value が原子値である場合、以下のいずれかのタイプを持ちます。

- xs:float
- xs:double
- xs:decimal
- xs:integer
- xdt:untypedAtomic
- 上記のいずれかのタイプから派生したタイプ

numeric-value が xdt:untypedAtomic タイプである場合、xs:double 値に変換されます。

戻り値

numeric-value が空のシーケンスでない場合、戻り値は *numeric-value* 以上の最小の整数値です。戻り値のデータ・タイプは、*numeric-value* のデータ・タイプによって異なります。

- *numeric-value* が xs:float、xs:double、xs:decimal、または xs:integer である場合、戻される値は *numeric-value* と同じタイプになります。
- *numeric-value* が xs:float、xs:double、xs:decimal、または xs:integer から派生したデータ・タイプを持つ場合、戻される値は *numeric-value* の直接の親のデータ・タイプです。

numeric-value が空のシーケンスである場合、戻り値は空のシーケンスです。

例

正の引数を使用した例: 以下の関数は、上限値 0.5 を戻します。

```
fn:ceiling(0.5)
```

戻り値は 1 です。

負の引数を使用した例: 以下の関数は、上限値 (-1.2) を戻します。

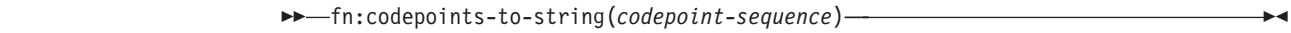
```
fn:ceiling(-1.2)
```

戻り値は -1 です。

codepoints-to-string 関数

fn:codepoints-to-string 関数は、Unicode コード・ポイントのシーケンスに相当するストリングを戻します。

構文



codepoint-sequence
Unicode コード・ポイントに対応する整数のシーケンス、または空のシーケンス。

戻り値

codepoint-sequence が空のシーケンスでない場合、戻り値は *codepoint-sequence* の項目と同等の文字の連結であるストリングです。*codepoint-sequence* の項目が有効な Unicode コード・ポイントでない場合、エラーが戻されます。

codepoint-sequence が空のシーケンスである場合、戻り値はゼロ長ストリングです。

例

以下の関数は、UTF-8 コード・ポイントのシーケンス (88,81,117,101,114,121) と同等の文字を戻します。

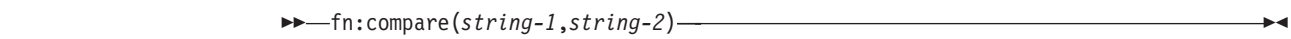
```
fn:codepoints-to-string((88,81,117,101,114,121))
```

戻り値は、'XQuery' です。

compare 関数

fn:compare 関数は、2 つのストリングを比較します。

構文



string-1 , *string-2*
比較される xs:string 値。

戻り値

string-1 および *string-2* が空のシーケンスでない場合、以下のいずれかの値が戻されます。

-1 *string-1* が *string-2* 未満の場合。

0 *string-1* が *string-2* と等しい場合。

1 *string-1* が *string-2* より大きい場合。

ゼロ長の場合も含めて同じ長さを持ち、デフォルトの照合に従って対応するすべての文字が等しい場合、 *string-1* と *string-2* は等しくなります。

string-1 と *string-2* が等しくない場合、その関係 (より大きい値を持つ方) は、ストリングの左端から等しくない文字の最初の組を比較することで判別されます。この比較は、デフォルトの照合に従って行われます。

string-1 が *string-2* より長く、*string-2* のすべての文字が *string-1* の先行文字と等しい場合、*string-1* は *string-2* よりも大きくなります。

string-1 または *string-2* が空のシーケンスである場合、空のシーケンスが戻されます。

例

以下の関数は、デフォルトの照合を使用して 'ABC' と 'ABD' を比較します。

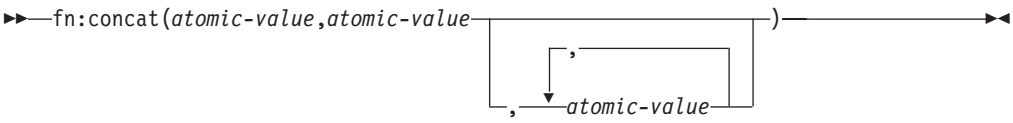
```
fn:compare('ABC', 'ABD')
```

'ABC' は 'ABD' よりも小さいです。戻り値は -1 です。

concat 関数

fn:concat 関数は、2 つ以上の原子値を連結したストリングを戻します。

構文



atomic-value
: 原子値または空のシーケンス。引数が空のシーケンスの場合、引数は、ゼロ長のストリングとして処理されます。*atomic-value* が xs:string 値でない場合、値が連結される前に xs:string にキャストされます。

戻り値

すべての *atomic-value* の引数が空のシーケンスである場合、戻り値はゼロ長ストリングです。空のシーケンスでない場合、戻り値は *atomic-value* 引数をストリングにキャストした結果の xs:string 値の連結です。

例

以下の関数は、ストリング 'ABC'、'ABD'、空のシーケンス、および 'ABE' を連結します。

```
fn:concat('ABC', 'ABD', (), 'ABE')
```

戻り値は、'ABCABDABE' です。

contains 関数

fn:contains 関数は、ストリングに、指定されたサブストリングが含まれるかどうかを判別します。サブストリングはデフォルトの照合を使用して突き合わされます。

構文

string　*substring* を検索するストリング。

string は、xs:string データ・タイプであるか、空のシーケンスです。*string* が空のシーケンスの場合、*string* はゼロ長ストリングに設定されます。

substring

string 内を検索するサブストリング。

substring は、xs:string データ・タイプであるか、空のシーケンスです。

戻り値

戻り値は、以下の条件のいずれかが満たされた場合、xs:boolean 値 true になります。

- *substring* が *string* 内にある。
- *substring* が空のシーケンスまたはゼロ長ストリングである。

それ以外の場合、戻り値は false です。

例

以下の関数は、ストリング 'Test literal' にストリング 'lite' が含まれるかどうかを判別します。

```
fn:contains('Test literal','lite')
```

戻り値は true です。

count 関数

fn:count 関数は、シーケンス内の値の個数を戻します。

構文

sequence-expression

任意のタイプの項目を含むシーケンスまたは空のシーケンス。

戻り値

sequence-expression が空のシーケンスでない場合、*sequence-expression* の値の数が戻されます。*sequence-expression* が空のシーケンスの場合、0 が戻されます。

例

以下の関数では、シーケンス (5, 1.0E2, 40.5) に含まれる項目の数が戻されます。

```
fn:count((5, 1.0E2, 40.5))
```

戻り値は 3 です。

current-date 関数

fn:current-date 関数は、暗黙指定された UTC の時間帯で現在日付を戻します。

構文

戻り値

戻り値は、現在日付である xs:date 値です。

例

以下の関数は現在日付を戻します。

```
fn:current-date()
```

この関数が 2005 年 12 月 2 日に呼び出された場合、戻り値は 2005-12-02Z です。

current-dateTime 関数

fn:current-dateTime 関数は、暗黙指定された UTC の時間帯で現在日時を戻します。

構文

戻り値

戻り値は、現在の日付と時刻の xs:dateTime 値です。

例

以下の関数は、現在の日付と時刻を戻します。

```
fn:current-dateTime()
```

この関数がトロント (時間帯 -PT5H) で 2005 年 12 月 2 日 6:25 に呼び出された場合、戻り値は 2005-12-02T011:25:30.864001Z です。

current-time 関数

fn:current-time 関数は、暗黙指定された UTC の時間帯で現在時刻を戻します。

構文

戻り値

戻り値は、現在時刻の xs:time の値です。

例

以下の関数は現在時刻を戻します。

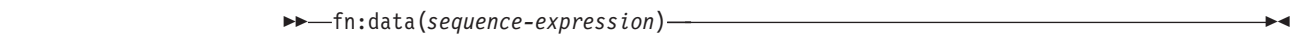
```
fn:current-time()
```

この関数がグリニッジ標準時 6:31 に呼び出された場合、戻り値は 06:31:35.519001Z です。

data 関数

fn:data 関数は、入力シーケンス内のノードをその型付き値に置き換えた後の入力シーケンスを戻します。

構文



sequence-expression
任意のシーケンス (空のシーケンスを含む)。

戻り値

sequence-expression が空のシーケンスである場合、戻り値は空のシーケンスです。

sequence-expression が単一の原子値の場合、戻り値は *sequence-expression* です。

sequence-expression が単一のノードである場合、戻り値は *sequence-expression* の型付き値です。

sequence-expression が複数項目のシーケンスの場合、原子値のシーケンスが *sequence-expression* の項目から戻されます。 *sequence-expression* の各原子値は変更されません。 *sequence-expression* の各ノードは、ゼロ以上の原子値のシーケンスである型付き値によって置換されます。

例

以下の関数は、シーケンス (<x xsi:type=″string″>ABC</x>,<y xsi:type=″decimal″>1.23</y>) に属する原子値を含むシーケンスを戻します。

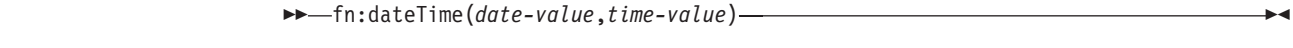
```
fn:data((<x xsi:type="string">ABC</x>,<y xsi:type="decimal">1.23</y>))
```

戻り値は (″ABC″,1.23) です。

dateTime 関数

fn:dateTime 関数は、xs:date 値および xs:time 値から、xs:dateTime 値を構成します。

構文



date-value
xs:date 値。

time-value
xs:time 値。

戻り値

戻り値は *date-value* と等しい日付コンポーネント、および *time-value* と等しい時間コンポーネントを持つ xs:dateTime 値です。結果の時間帯は、以下のように計算されます。

- どちらの引数も時間帯を持たない場合、結果も時間帯を持ちません。
- 引数のどちらか一方のみが時間帯を持つ場合、または両方の引数が同じ時間帯を持つ場合、結果はこの時間帯を持ちます。
- 2 つの引数が異なる時間帯を持つ場合、エラーが戻されます。

例

以下の関数は、xs:date 値および xs:time 値の xs:dateTime 値を戻します。

```
fn:dateTime((xs:date("2005-04-16")), (xs:time("12:30:59")))
```

戻り値は xs:dateTime 値 2005-04-16T12:30:59 です。

day-from-date 関数

fn:day-from-date 関数は xs:date 値の日付コンポーネントを戻します。

構文

date-value
日付コンポーネントが取り出される日付値。

date-value は xs:date タイプであるか、空のシーケンスです。

戻り値

date-value は xs:date タイプであり、戻り値は xs:integer タイプであり、値の範囲は 1 から 31 です。値は *date-value* の日付コンポーネントです。

date-value が空のシーケンスである場合、戻り値は空のシーケンスです。

例

以下の関数は、2005 年 6 月 1 日の日付値の日付コンポーネントを戻します。

```
fn:day-from-date(xs:date("2005-06-01"))
```

戻り値は 1 です。

day-from-dateTime 関数

fn:day-from-dateTime 関数は、xs:dateTime 値の日付コンポーネントを戻します。

構文

```
►►—fn:day-from-dateTime(dateTime-value)———————————————————————►◄
```

dateTime-value
日付コンポーネントが取り出される dateTime 値。

dateTime-value は xs:dateTime タイプであるか、空のシーケンスです。

戻り値

dateTime-value は xs:dateTime タイプであり、戻り値は xs:integer タイプであり、値の範囲は 1 から 31 です。値は、*dateTime-value* の日付コンポーネントです。

dateTime-value が空のシーケンスである場合、戻り値は空のシーケンスです。

例

以下の関数は、UTC+4 タイム・ゾーンの 2005 年 1 月 31 日、午前 8 時の dateTime 値の日付コンポーネントを戻します。

```
fn:day-from-dateTime(xs:dateTime("2005-01-31T20:00:00+04:00"))
```

戻り値は 31 です。

days-from-duration 関数

fn:days-from-duration 関数は、期間の日付コンポーネントを戻します。

構文

```
►►—fn:days-from-duration(duration-value)———————————————————————►◄
```

duration-value
日付コンポーネントが取り出される期間値。

duration-value は空のシーケンスであるか、タイプが xdt:dayTimeDuration、xs:duration、または xdt:yearMonthDuration のいずれかである値です。

戻り値

戻り値は、*duration-value* のタイプによって異なります。

- *duration-value* のタイプが xdt:dayTimeDuration または xs:duration である場合、戻り値は xs:integer タイプであり、xdt:dayTimeDuration としてキャストされる *duration-value* の日付コンポーネントです。 *duration-value* が負の数である場合、戻り値も負の数です。
- *duration-value* がタイプ xdt:yearMonthDuration である場合、戻り値は 0 です。
- *duration-value* が空のシーケンスである場合、戻り値は空のシーケンスです。

xdt:dayTimeDuration としてキャストされる *duration-value* の日付コンポーネントは、(*S* idiv 86400) として計算される日数 (整数) です。値 *S* は、年および月コンポーネントを除去するための、xdt:dayTimeDuration としてキャストされる *duration-value* の総秒数です。

例

この関数では、-10 日と 0 時間の期間の日付コンポーネントを戻します。

```
fn:days-from-duration(xdt:dayTimeDuration("-P10DT00H"))
```

戻り値は -10 です。

この関数では、3 日と 55 時間の期間の日付コンポーネントを戻します。

```
fn:days-from-duration(xdt:dayTimeDuration("P3DT55H"))
```

戻り値は 5 です。期間の総日数を計算する場合、55 時間は 2 日と 7 時間に変換されます。この期間は P5D7H と等しく、これは 5 日の日付コンポーネントを持ちます。

deep-equal 関数

fn:deep-equal 関数は、2 つのシーケンスを比較して、それらが値同等性の要件を満たすかどうかを判別します。

構文

sequence-1, *sequence-2*
比較対象のシーケンス。各シーケンスの項目は、任意のタイプの原子値またはノードです。

戻り値

sequence-1 と *sequence-2* が完全に一致する場合、戻り値は xs:boolean の値 true です。それ以外の場合、戻り値は false です。

sequence-1 と *sequence-2* が空のシーケンスである場合、これらは完全に一致します。

両方のシーケンスが空でない場合、以下の両方の要件を満たすと、この 2 つのシーケンスは完全に一致することになります。

- *sequence-1* の項目数が *sequence-2* の項目数と等しい。
- *sequence-1* (*item-1*) の各項目が、*sequence-2* (*item-2*) の対応する項目について完全一致の条件を満たしている。以下のいずれかの条件を満たす場合、 *item-1* および *item-2* は完全に一致することになります。
  - *item-1* および *item-2* はともに原子値であり、次の条件のいずれかを満たします。
    - 式 *item-1* eq *item-2* が true を戻す。

- *item-1* および *item-2* の両方とも xs:float タイプまたは xs:double タイプであり、値 NaN を持つ。

– *item-1* および *item-2* は、両方とも同じ種類のノードであり、以下の表の完全一致の条件を満たす。

表 36. シーケンスのノードの完全一致

item-1 および item-2 両方のノードの種類	**完全一致の条件**
文書	item-1 のテキストおよびエレメントの子のシーケンスが、item-2 のテキストおよびエレメントの子のシーケンスと完全に一致する。
エレメント	以下の条件すべてが true である必要があります。 • *item-1* および *item-2* が同じ名前を持つ。つまり、両方のネーム・スペース URI が一致し、ローカル名も一致することを意味します。ネーム・スペース接頭部は無視されます。名前の突き合わせは、バイナリー比較を使用して行われます。 • *item-1* および *item-2* の属性の数が同じであり、*item-1* の各属性が *item-2* の属性と完全に一致する。 • 以下のいずれかの条件が true である。 – 両方のノードが、混合コンテンツ (テキストおよび子のエレメントの両方) を許可するタイプで未検証または検証済みのいずれかであり、*item-1* のテキストおよびエレメントの子のシーケンスが *item-2* のテキストおよびエレメントの子のシーケンスと完全に一致する。 – 両方のノードが単純タイプ (xs:decimal) または単純コンテンツ (コンテンツが xs:decimal の「temperature」タイプなど) を持つタイプで検証され、*item-1* の型付き値が *item-2* の型付き値と完全に一致する。 – 両方のノードがコンテンツなし (テキストも子のエレメントもなし) を許可するタイプで検証される。 – 両方のノードが子エレメントのみ (テキストなし) を許可するタイプで検証され、*item-1* のそれぞれの子エレメントが *item-2* の対応する子エレメントと完全に一致する。
属性	以下の条件すべてが true である必要があります。 • *item-1* および *item-2* が同じ名前を持つ。つまり、両方のネーム・スペース URI が一致し、ローカル名も一致することを意味します。ネーム・スペース接頭部は無視されます。名前の突き合わせは、バイナリー比較を使用して行われます。 • *item-1* の型付き値が、*item-2* の型付き値と完全に一致する。
テキスト	デフォルトの照合を使用して eq 演算子でストリングとして比較した場合に、コンテンツ・プロパティー値が等しい。
コメント	デフォルトの照合を使用して eq 演算子でストリングとして比較した場合に、コンテンツ・プロパティー値が等しい。
処理命令	以下の条件すべてが true である必要があります。 • *item-1* および *item-2* が同じ名前を持つ。 • デフォルトの照合を使用して eq 演算子でストリングとして比較した場合に、コンテンツ・プロパティー値が等しい。

例

以下の関数は、シーケンス (1,'ABC') と (1,'ABCD') が完全一致であるかどうか比較します。ストリングの比較は、デフォルトの相関を使用します。

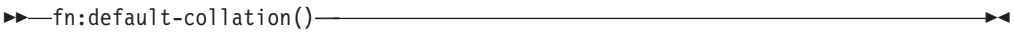
```
fn:deep-equal((1,'ABC'), (1,'ABCD'))
```

戻り値は false です。

default-collation 関数

fn:default-collation 関数は、データベースについて定義されているデフォルト照合を表す URI を戻します。

構文



戻り値

戻り値は xs:anyURI タイプで、データベースの照合を指定します。

例

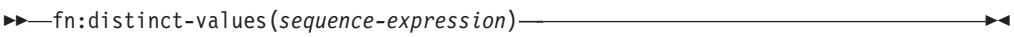
DB2 データベースは、照合 UCA400_NO を指定して作成されます。このデータベースに対して照会を実行すると、以下の関数は `http://www.ibm.com/xmlns/prod/db2/sql/collations?name=UCA400_NO` を戻します。

```
fn:default-collation()
```

distinct-values 関数

fn:distinct-values 関数は、シーケンス内で重複を取り除いた値を戻します。

構文



sequence-expression
原子値のシーケンス、または空のシーケンス。

戻り値

sequence-expression が空のシーケンスでない場合、戻り値は、*sequence-expression* 内の別個の値を含むシーケンスです。2 つの値 *value1* と *value2* は、デフォルトの照合を使用した *value1* eq *value2* が false の場合、別個のものです。eq 演算子が 2 つの値に対して定義されていない場合、これらの値は別個のものと見なされます。

xdt:untypedAtomic タイプの値は、値の比較の前に xs:string タイプの値に変換されます。

xs:float 値および xs:double 値の場合、*sequence-expression* に複数の NaN 値が含まれる場合、単一の NaN 値が戻されます。

xs:dateTime 値、xs:date 値、または xs:time 値の場合、それらの値の比較の前に、時間帯の相違が調整されます。値に時間帯が指定されていない場合、暗黙的な時間帯 (UTC) が使用されます。

sequence-expression が空のシーケンスである場合、空のシーケンスが戻されます。

入力シーケンスの 2 つの値が *eq* 演算子により等しくても、タイプが異なる場合、値のいずれか一方 (両方ではない) が結果のシーケンスに表示されることがあります。結果のシーケンスは、入力シーケンスの順序を保持しない場合があります。

例

以下の関数は、シーケンスのノードを原子化した後にシーケンスの別個の値を戻します。

```
fn:distinct-values((1, 'a', 1.0, 'A', <greeting>Hello</greeting>))
```

戻り値は (1, 'a', 'A', 'Hello') または (1.0, 'A', 'a', 'Hello') になります。

empty 関数

fn:empty 関数は、引数が空のシーケンスであるかどうかを示します。

構文

item 任意のデータ・タイプの式、または空のシーケンス。

戻り値

戻り値は、*item* が空のシーケンスである場合 true です。それ以外の場合、戻り値は false です。

例

以下の例では、empty 関数を使用して変数 $seq 内のシーケンスが空のシーケンスかどうかを判別します。

```
let $seq := (5, 10)
return fn:empty($seq)
```

戻り値は false です。

ends-with 関数

fn:ends-with 関数は、ストリングが、指定されたサブストリングで終了しているかどうかを判別します。サブストリングはデフォルトの照合を使用して突き合わされます。

構文

string　*substring* を検索するストリング。

string は、xs:string データ・タイプであるか、空のシーケンスです。*string* が空のシーケンスである場合、*string* はゼロ長ストリングに設定されます。

substring

string の最後で検索するサブストリング。

substring は、xs:string データ・タイプであるか、空のシーケンスです。

戻り値

戻り値は、以下の条件のいずれかが満たされた場合、xs:boolean 値 true になります。

- *substring* が *string* の最後に出現する。
- *substring* が空のシーケンスまたはゼロ長ストリングである。

それ以外の場合、戻り値は false です。

例

以下の関数は、ストリング 'Test literal' がストリング 'literal' で終了するかどうかを判別します。

```
fn:ends-with('Test literal','literal')
```

戻り値は true です。

exactly-one 関数

fn:exactly-one 関数は、引数に正確に 1 つの項目が含まれる場合にその引数を戻します。

構文

sequence-expression

任意のシーケンス (空のシーケンスを含む)。

戻り値

sequence-expression に正確に 1 個の項目が含まれる場合、*sequence-expression* が戻されます。それ以外の場合は、エラーが戻されます。

例

以下の例では、exactly-one 関数を使用して変数 $seq 内のシーケンスが正確に 1 個の項目を含むかどうかを判別します。

```
let $seq := 5
return fn:exactly-one($seq)
```

値 5 が戻されます。

exists 関数

fn:exists 関数は、シーケンスが空のシーケンスでないかどうかを示します。

構文

sequence-expression
任意のデータ・タイプのシーケンスまたは空のシーケンス。

戻り値

戻り値は、*sequence-expression* が空のシーケンスでない場合 true です。それ以外の場合、戻り値は false です。

例

以下の例では、exists 関数を使用して、変数 $seq 内のシーケンスが空のシーケンスでないかどうかを判別します。

```
let $seq := (5, 10)
return fn:exists($seq)
```

値 true が戻されます。

false 関数

fn:false 関数は、xs:boolean 値 false を戻します。

構文

戻り値

戻り値は、xs:boolean 値の false です。

例

false 関数を使用して値 false を戻します。

```
fn:false()
```

値 false が戻されます。

floor 関数

fn:floor 関数は、指定された数値以下の最大整数値を戻します。

構文

numeric-value
原子値または空のシーケンス。

numeric-value が原子値である場合、以下のいずれかのタイプを持ちます。
- xs:float
- xs:double
- xs:decimal
- xs:integer
- xdt:untypedAtomic
- 上記のいずれかのタイプから派生したタイプ

numeric-value が xdt:untypedAtomic タイプである場合、 xs:double 値に変換されます。

戻り値

numeric-value が空のシーケンスでない場合、戻り値は *numeric-value* より小さい最大整数です。戻り値のデータ・タイプは、*numeric-value* のデータ・タイプによって異なります。

- *numeric-value* が xs:float、xs:double、xs:decimal、または xs:integer である場合、戻される値は *numeric-value* と同じタイプになります。
- *numeric-value* が xs:float、xs:double、xs:decimal、または xs:integer から派生したデータ・タイプを持つ場合、戻される値は *numeric-value* の直接の親のデータ・タイプです。

numeric-value が空のシーケンスである場合、戻り値は空のシーケンスです。

例

正の引数を使用した例: 以下の関数は floor 値 0.5 を戻します。

```
fn:floor(0.5)
```

戻り値は 0 です。

負の引数を使用した例: 以下の関数は floor 値 (-1.2) を戻します。

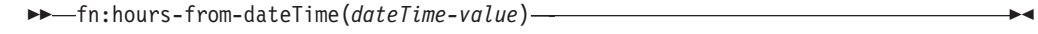
```
fn:floor(-1.2)
```

戻り値は -2 です。

hours-from-dateTime 関数

fn:hours-from-dateTime 関数は、xs:dateTime 値の時間コンポーネントを戻します。

構文



dateTime-value
時間コンポーネントが取り出される dateTime 値。

dateTime-value は xs:dateTime タイプであるか、空のシーケンスです。

戻り値

dateTime-value は xs:dateTime タイプであり、戻り値は xs:integer タイプであり、値の範囲は 0 から 23 です。値は、*dateTime-value* の時間コンポーネントです。

dateTime-value が空のシーケンスである場合、戻り値は空のシーケンスです。

例

以下の関数は、UTC-8 タイム・ゾーンの 2005 年 1 月 31 日、午後 2 時の dateTime 値の時間コンポーネントを戻します。

```
fn:hours-from-dateTime(xs:dateTime("2005-01-31T14:00:00-08:00"))
```

戻り値は 14 です。

hours-from-duration 関数

fn:hours-from-duration 関数は、期間値の時間コンポーネントを戻します。

構文

duration-value
時間コンポーネントが取り出される期間値。

duration-value は空のシーケンスであるか、タイプが xdt:dayTimeDuration、xs:duration、または xdt:yearMonthDuration のいずれかである値です。

戻り値

戻り値は、*duration-value* のタイプによって異なります。

- *duration-value* のタイプが xdt:dayTimeDuration または xs:duration である場合、戻り値は xs:integer タイプであり、値の範囲は -23 から 23 です。値は、xdt:dayTimeDuration としてキャストされる *duration-value* の時間コンポーネントです。*duration-value* が負の数である場合、この値も負の数になります。
- *duration-value* が xdt:yearMonthDuration タイプである場合、戻り値は xs:integer タイプであり、0 です。
- *duration-value* が空のシーケンスである場合、戻り値は空のシーケンスです。

xdt:dayTimeDuration としてキャストされる *duration-value* の時間コンポーネントは、`((S mod 86400) idiv 3600)` として計算される時間数 (整数) です。値 *S* は、日および月コンポーネントを除去するための、xdt:dayTimeDuration としてキャストされる *duration-value* の総秒数です。1 日を秒数の値で表すと 86400 であり、1 時間を秒数の値で表すと 3600 です。

例

以下の関数は、期間が 126 時間の時間コンポーネントを戻します。

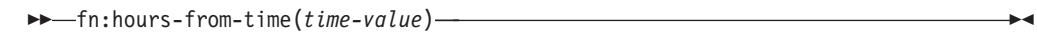
```
fn:hours-from-duration(xdt:dayTimeDuration("PT126H"))
```

戻り値は 6 です。期間の総時間数を計算する場合、126 時間は 5 日と 6 時間に変換されます。この期間は P5DT6H と等しく、これは 6 時間の時間コンポーネントを持ちます。

hours-from-time 関数

fn:hours-from-time 関数は、xs:time 値の時間コンポーネントを戻します。

構文



time-value
時間コンポーネントが取り出される時刻値。

time-value は xs:time タイプであるか、空のシーケンスです。

戻り値

time-value が空のシーケンスではない場合、戻り値は xs:integer タイプであり、値の範囲は 0 から 23 です。値は、*time-value* の時間コンポーネントです。

time-value が空のシーケンスである場合、戻り値は空のシーケンスです。

例

以下の関数は、UTC-8 タイム・ゾーンの午前 9 時 30 分の時刻値の時間コンポーネントを戻します。

```
fn:hours-from-time(xs:time("09:30:00-08:00"))
```

戻り値は 9 です。

implicit-timezone 関数

fn:implicit-timezone 関数は、暗黙指定された時間帯値 PT0S を戻します。これは xs:dayTimeDuration タイプです。値 PT0S は、UTC が暗黙指定された時間帯であることを示します。

構文

戻り値

戻り値は、xs:dayTimeDuration タイプで表された UTC である PT0S です。

例

以下の関数は xdt:dayTimeDuration(″PT0S″) を戻します。

```
fn:implicit-timezone()
```

in-scope-prefixes 関数

fn:in-scope-prefixes 関数は、エレメントのすべての範囲内のネーム・スペースの接頭部のリストを戻します。

構文

element
範囲内のネーム・スペースの接頭部が取得されるエレメント。

戻り値

戻り値は、 *element* のすべての範囲内のネーム・スペースの接頭部である xs:NCName 値のシーケンスです。デフォルトのネーム・スペースが *element* の範囲内にある場合、デフォルトのネーム・スペース接頭部のシーケンス項目は、ゼロ長ストリングです。ネーム・スペース「xml」は常に 1 つのエレメントの範囲内のネーム・スペースに含まれます。

例

以下の照会は、エレメント emp の範囲内のネーム・スペースの接頭部のシーケンスを (NCName として) 戻します。

```
declare namespace d="http://www.mycompany.com";
let $department := document {
    <comp:dept xmlns:comp="http://www.mycompany.com" id="A07">
        <comp:emp id="31201" />
    </comp:dept> }
return fn:in-scope-prefixes($department/d:dept/d:emp)
```

戻り値は (″xml″, ″comp″) です。必ずしもこの順序であるとは限りません。

index-of 関数

fn:index-of 関数は、シーケンス内で項目が現れる位置を戻します。

構文

sequence-expression
原子タイプの任意のシーケンスまたは空のシーケンス。

search-value
sequence-expression で検索する値。

戻り値

戻り値は、デフォルトの照合を使用して **eq** 演算子の規則を使用して比較したときに *search-value* と一致する *sequence-expression* の項目の位置を示す xs:integer 値のシーケンスです。該当するタイプに対して **eq** 演算子が定義されていないために比較できない項目は、*search-value* に一致しないと見なされるため、位置は戻されません。シーケンス内の最初の項目が、位置 1 になります。

この関数は、*search-value* が *sequence-expression* のいずれの項目とも一致しない場合、または *sequence-expression* が空のシーケンスである場合に空のシーケンスを戻します。

例

以下の関数は、シーケンス内の 'ABC' の位置を戻します。

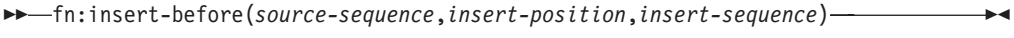
```
fn:index-of(('ABC','DEF','ABC','123'), 'ABC')
```

戻り値はシーケンス (1,3) です。

insert-before 関数

fn:insert-before 関数は、シーケンスを、別のシーケンスの指定された位置の前に挿入します。

構文



source-sequence
: シーケンスが挿入されるシーケンス。

  source-sequence は任意のデータ・タイプの項目のシーケンス、または空のシーケンスです。

insert-position
: シーケンスがその前に挿入される *source-sequence* 内の位置。*insert-position* は xs:integer タイプです。*insert-position*<=0 の場合、*insert-position* は 1 に設定されます。*insert-position* が *source-sequence* 内の項目数より大きい場合、*insert-position* は *source-sequence* 内の項目数より 1 大きい数に設定されます。

insert-sequence
: *source-sequence* に挿入されるシーケンス。

  insert-sequence は、任意のデータ・タイプの項目のシーケンス、または空のシーケンスです。

戻り値

source-sequence が空のシーケンスでない場合:

- *insert-sequence* が空のシーケンスでない場合、戻り値は以下の項目の、ここに示すとおりの順序のシーケンスです。
  - 項目 *insert-position* の前にある *source-sequence* の項目

- *insert-sequence* の項目
- 項目 *insert-position* にある *source-sequence* の項目
- 項目 *insert-position* の後にある *source-sequence* の項目。

- *insert-sequence* が空のシーケンスである場合、戻り値は *source-sequence* です。

source-sequence が空のシーケンスである場合:

- *insert-sequence* が空のシーケンスでない場合、戻り値は *insert-sequence* です。
- *insert-sequence* が空のシーケンスである場合、戻り値は空のシーケンスです。

例

以下の関数は、シーケンス (1,2,3,7) の 4 番目の位置の前にシーケンス (4,5,6) を挿入した結果のシーケンスを戻します。

```
fn:insert-before((1,2,3,7),4,(4,5,6))
```

戻り値は (1,2,3,4,5,6,7) です。

last 関数

fn:last 関数は、現在処理中のシーケンス内の値の個数を戻します。

構文

戻り値

現在処理中のシーケンスが空のシーケンスでない場合、戻り値はシーケンス内の値の数です。現在処理中のシーケンスが空のシーケンスである場合、戻り値は空のシーケンスです。

例

以下の例は、述部式として関数を使用して、現行シーケンス内の最後の項目を戻します。

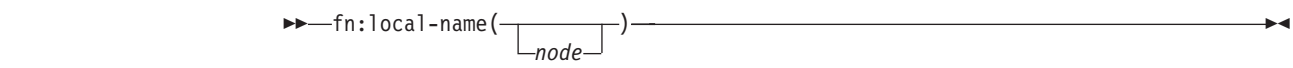
```
(<a/>, <b/>, <c/>)[fn:last()]
```

戻り値は <c/> です。

local-name 関数

fn:local-name 関数は、ノードのローカル名プロパティーを戻します。

構文



node ローカル名が取得されるノード。*node* が指定されていない場合、fn:local-name は、現行のコンテキスト・ノードについて評価されます。

戻り値

戻り値は、*node* が指定されているかどうか、および *node* の値によって異なります。

- *node* が指定されていない場合、コンテキスト・ノードのローカル名が戻されます。
- *node* が以下のいずれかの条件と一致する場合、ゼロ長ストリングが戻されます。
  - *node* が空のシーケンスである。
  - *node* がエレメント・ノード、属性ノード、または処理命令ノードではない。
- *node* が以下のいずれかの条件と一致する場合、エラーが戻されます。
  - *node* が未定義である。
  - *node* がノードでない。
- それ以外の場合、*node* の拡張名のローカル名部が含まれる xs:string 値が戻されます。

例

以下の関数は、ノード emp のローカル名を戻します。

```
declare namespace a="http://posample.org";
fn:local-name(<a:b/>)
```

戻り値は `b` です。

local-name-from-QName 関数

fn:local-name-from-QName 関数は、 xs:QName 値のローカル部分を戻します。

構文

qualified-name
ローカル部分が取得される修飾名。

qualified-name は、xs:QName データ・タイプであるか、空のシーケンスです。

戻り値

qualified-name が空のシーケンスでない場合、戻される値は、*qualified-name* のローカル部である xs:NCName 値です。*qualified-name* が空のシーケンスである場合、空のシーケンスが戻されます。

例

以下の関数は、修飾名のローカル部を戻します。

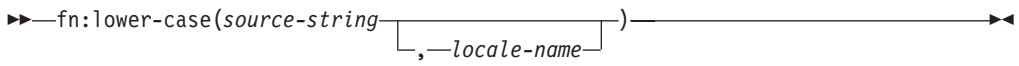
```
fn:local-name-from-QName(fn:QName("http://www.mycompany.com/", "ns:employee"))
```

戻り値は ″employee″ です。

lower-case 関数

fn:lower-case 関数は、ストリングを小文字に変換します。

構文



source-string
小文字に変換されるストリング。

source-string は xs:string タイプであるか、空のシーケンスです。

locale-name
小文字の操作に使用するロケールを含むストリング。

locale-name は xs:string タイプであるか、空のシーケンスです。*locale-name* が空のシーケンスでない場合は、*locale-name* の値に大文字と小文字の区別はなく、有効なロケールまたは長さゼロのストリングでなければなりません。

戻り値

source-string が空のシーケンスでない場合、戻り値は、各文字が対応する小文字に変換された *source-string* です。*locale-name* が指定されていないか、空のシーケンスか、または長さゼロのストリングの場合は、Unicode 規格で定義されている小文字の規則が使用されます。それ以外の場合は、指定されたロケールの小文字の規則が使用されます。対応する小文字がない文字はすべて、元の形式で戻り値に含まれます。

source-string が空のシーケンスである場合、戻り値はゼロ長ストリングです。

例

以下の関数は、ストリング ″Wireless Router TB2561″ を小文字に変換します。

```
fn:lower-case("Wireless Router TB2561")
```

戻り値は ″wireless router tb2561″ です。

以下の関数は、トルコ語のロケール tr_TR を指定し、文字 ″I″ と数値による文字参照 İ (ドットが上に付いたローマ字の大文字 I の文字参照) を変換します。

```
fn:lower-case("I&#x130;", "tr_TR")
```

戻り値は、ı (ドットのないローマ字の小文字 i) と文字 ″i″ の 2 つの文字で構成されます。トルコ語のロケールの場合、文字 ″I″ は ı (ドットのないローマ字の小文字 i) で表される文字に変換され、İ (ドットが上に付いたローマ字の大文字 I) は ″i″ に変換されます。

以下の関数は、ロケールを指定せず、Unicode 規格で定義されている規則を使用して文字 ″I″ を小文字に変換します。

```
fn:lower-case("I")
```

戻り値は文字 ″i″ です。

matches 関数

fn:matches 関数は、ストリングが、指定されたパターンに一致するかどうかを判別します。

構文

source-string
パターンと比較されるストリング。

source-string は xs:string 値または空のシーケンスです。

pattern *source-string* と比較される正規表現。正規表現は、検索パターン内のストリングまたはストリングのグループを定義する、文字、ワイルドカード、および演算子のセットです。

pattern は xs:string 値です。

flags *pattern* と *source-string* のマッチングを制御する、以下の値のいずれかを含む xs:string 値。

s ドット (.) がすべての文字と一致することを示します。

s フラグが指定されていない場合、ドット (.) は、改行文字 (X'0A') 以外のすべての文字と一致します。

m 脱字記号 (^) が行の開始 (改行文字の後の位置) と一致し、ドル記号 ($) が行の最後 (改行文字の前の位置) と一致することを示します。

m フラグが指定されていない場合、脱字記号 (^) はストリングの開始と一致し、ドル記号 ($) はストリングの最後と一致します。

i マッチングにおいて大/小文字を区別しないことを示します。

i フラグが指定されていない場合、大/小文字を区別してマッチングが行われます。

x *pattern* 内の空白文字が無視されることを示します。

x フラグが指定されていない場合、空白文字を考慮してマッチングします。

戻り値

source-string が空のシーケンスでない場合、戻り値は *source-string* が *pattern* と一致する場合 true になります。戻り値は、*source-string* が *pattern* と一致しない場合 false になります。

pattern にストリングまたは行の開始文字の脱字記号 (^)、あるいはストリングまたは行の終了文字のドル記号 ($) が含まれていない場合、*source-string* のいずれかのサブストリングが *pattern* と一致すると、*source-string* と *pattern* が一致することに

なります。*pattern* にストリングまたは行の開始文字の脱字記号 (^) が含まれている場合、*source-string* が *source-string* の開始または *source-string* の行の開始から *pattern* と一致する場合にのみ *source-string* と *pattern* は一致します。*pattern* にストリングまたは行の終了文字のドル記号 ($) が含まれている場合、*source-string* が *source-string* の最後または *source-string* の行の最後で *pattern* と一致する場合にのみ *source-string* と *pattern* は一致します。m フラグは、マッチングがストリングの最初から行われるのか、行の最初から行われるのかを判別します。

source-string が空のシーケンスである場合、戻り値は false です。

例

ストリング内の任意のサブストリングとパターンのマッチングの例: 以下の関数は、文字 ″ac″ または ″bd″ がストリング ″abbcacadbdcd″ 内の任意の場所にあるかどうかを判別します。

```
fn:matches("abbcacadbdcd","(ac)|(bd)")
```

戻り値は true です。

パターンとストリング全体のマッチングの例: 以下の関数は、文字 ″ac″ または ″bd″ がストリング ″bd″ と一致するかどうかを判別します。

```
fn:matches("bd","^(ac)|(bd)$")
```

戻り値は true です。

パターンをマッチングするときにスペースと大文字小文字の違いを無視する例: 以下の関数は、i および x フラグを使用して、ストリング ″abc1234″ がパターン ″ABC 1234″ と一致するかどうかを判別するときに大文字小文字の違いとスペースを無視します。

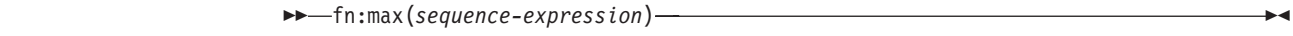
```
fn:matches("abc1234","ABC 1234", "ix")
```

戻り値は true です。

max 関数

fn:max 関数は、シーケンス内の最大値を戻します。

構文



sequence-expression

以下の原子タイプのいずれかの項目を含むシーケンス、または空のシーケンス。

- xs:float
- xs:double
- xs:decimal
- xs:integer
- xs:string
- xs:date

- xs:time
- xs:dateTime
- xdt:untypedAtomic
- xdt:dayTimeDuration
- xdt:yearMonthDuration
- 上記のいずれかのタイプから派生したタイプ

xdt:untypedAtomic タイプの入力項目は、xs:double にキャストされます。このキャスト後、入力シーケンス内のすべての項目は、プロモーションまたはサブタイプの置換によって、**ge** 演算子をサポートする共通のタイプに変換できる必要があります。最大値はこの共通のタイプで計算されます。例えば、入力シーケンスに xs:decimal から派生した money タイプの項目と xs:float から派生した stockprice タイプの項目が含まれる場合、最大値は xs:float タイプで計算されます。

日付値、時間値、または日時値は、比較される前に共通の時間帯に調整されます。明示的な時間帯コンポーネントを持たない日時値は、暗黙的な時間帯 (UTC) を使用します。

ストリング値はデフォルトの照合を使用して比較されます。

戻り値

sequence-expression が空のシーケンスでない場合、戻り値は *sequence-expression* 内の最大値です。戻り値のデータ・タイプは、*sequence-expression* 内の項目のデータ・タイプか、*sequence-expression* 内の項目がプロモートされる共通のデータ・タイプと同じになります。

sequence-expression が空のシーケンスである場合、空のシーケンスが戻されます。シーケンスに値 NaN が含まれる場合、NaN が戻されます。

例

以下の関数は、シーケンス (500, 1.0E2, 40.5) の最大値を戻します。

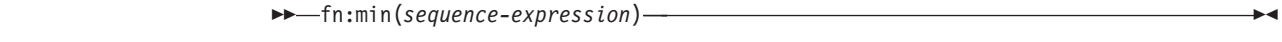
```
fn:max((500, 1.0E2, 40.5))
```

値は xs:double タイプにプロモートされます。関数は xs:double 値 5.0E2 を戻し、これは「500」としてシリアライズされます。

min 関数

fn:min 関数は、シーケンス内の最小値を戻します。

構文



sequence-expression

以下の原子タイプのいずれかの項目を含むシーケンス、または空のシーケンス。

- xs:float
- xs:double

- xs:decimal
- xs:integer
- xs:string
- xs:date
- xs:time
- xs:dateTime
- xdt:untypedAtomic
- xdt:dayTimeDuration
- xdt:yearMonthDuration
- 上記のいずれかのタイプから派生したタイプ

xdt:untypedAtomic タイプの入力項目は、xs:double にキャストされます。このキャスト後、入力シーケンス内のすべての項目は、プロモーションまたはサブタイプの置換によって **le** 演算子をサポートする共通のタイプに変換できる必要があります。最小値はこの共通のタイプで計算されます。例えば、入力シーケンスに xs:decimal から派生した money タイプの項目と、xs:float から派生した stockprice タイプの項目が含まれている場合、最小値は xs:float タイプで計算されます。

日付値、時間値、または日時値は、比較される前に共通の時間帯に調整されます。明示的な時間帯コンポーネントを持たない日時値は、暗黙的な時間帯 (UTC) を使用します。

ストリング値はデフォルトの照合を使用して比較されます。

戻り値

sequence-expression が空のシーケンスでない場合、戻り値は *sequence-expression* 内の最小値です。戻り値のデータ・タイプは、*sequence-expression* 内の項目のデータ・タイプか、*sequence-expression* 内の項目がプロモートされる共通のデータ・タイプと同じになります。

sequence-expression が空のシーケンスである場合、空のシーケンスが戻されます。シーケンスに値 NaN が含まれる場合、NaN が戻されます。

例

数値の引数を使用した例: 以下の関数は、シーケンス (500, 1.0E2, 40.5) の最小値を戻します。

```
fn:min((500, 1.0E2, 40.5))
```

値は xs:double タイプにプロモートされます。関数は、xs:double 値 4.05E1 を戻し、これは「40.5」としてシリアライズされます。

ストリングの引数を使用した例: 以下の関数は、デフォルトの照合を使用してシーケンス (″x″, ″y″, ″Z″) の最小値を戻します。デフォルトの照合では、英大文字の前に英小文字がソートされるものとします。

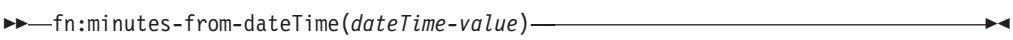
```
fn:min(("x", "y", "Z"))
```

戻り値は ″x″ です。

minutes-from-dateTime 関数

fn:minutes-from-dateTime 関数は、xs:dateTime 値の分コンポーネントを戻します。

構文



dateTime-value
分コンポーネントが取り出される dateTime 値。

dateTime-value は xs:dateTime タイプであるか、空のシーケンスです。

戻り値

dateTime-value は xs:dateTime タイプであり、戻り値は xs:integer タイプであり、値の範囲は 0 から 59 です。値は、*dateTime-value* の分コンポーネントです。

dateTime-value が空のシーケンスである場合、戻り値は空のシーケンスです。

例

以下の関数は、UTC-8 タイム・ゾーンの 2005 年 1 月 23 日、午前 9 時 42 分の dateTime 値の分コンポーネントを戻します。

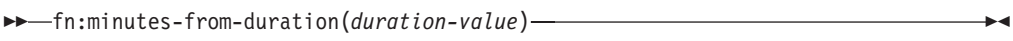
```
fn:minutes-from-dateTime(xs:dateTime("2005-01-23T09:42:00-08:00"))
```

戻り値は 42 です。

minutes-from-duration 関数

fn:minutes-from-duration 関数は、期間の分コンポーネントを戻します。

構文



duration-value
分コンポーネントが取り出される期間値。

duration-value は空のシーケンスであるか、タイプが xdt:dayTimeDuration、xs:duration、または xdt:yearMonthDuration のいずれかである値です。

戻り値

戻り値は、*duration-value* のタイプによって異なります。

- *duration-value* のタイプが xdt:dayTimeDuration または xs:duration である場合、戻り値は xs:integer タイプであり、値の範囲は -59 から 59 です。値は、xdt:dayTimeDuration としてキャストされる *duration-value* の分コンポーネントです。*duration-value* が負の数である場合、この値も負の数になります。
- *duration-value* がタイプ xdt:yearMonthDuration である場合、戻り値は 0 です。
- *duration-value* が空のシーケンスである場合、戻り値は空のシーケンスです。

xdt:dayTimeDuration としてキャストされる *duration-value* の分コンポーネントは、((S mod 3600) idiv 60) として計算される分数 (整数) です。値 *S* は、年および月コンポーネントを除去するための、xdt:dayTimeDuration としてキャストされる *duration-value* の総秒数です。

例

以下の関数は、期間が 2 日と 16 時間 93 分の分コンポーネントを戻します。

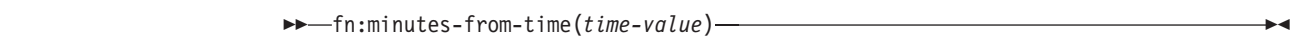
```
fn:minutes-from-duration(xdt:dayTimeDuration("P2DT16H93M"))
```

戻り値は 33 です。この期間の総分数を計算する場合、93 分は 1 時間と 33 分に変換されます。この期間は分コンポーネント 33 分を持つ、P2DT17H33M と等しいと言えます。

minutes-from-time 関数

fn:minutes-from-time 関数は、xs:time 値の分コンポーネントを戻します。

構文



time-value
分コンポーネントが取り出される時刻値。

time-value は xs:time タイプであるか、空のシーケンスです。

戻り値

time-value は xs:time タイプであり、戻り値は xs:integer タイプであり、値の範囲は 0 から 59 です。値は、*time-value* の分コンポーネントです。

time-value が空のシーケンスである場合、戻り値は空のシーケンスです。

例

以下の関数は、UTC-8 タイム・ゾーンの午前 8 時 59 分の時刻値の分コンポーネントを戻します。

```
fn:minutes-from-time(xs:time("08:59:00-08:00"))
```

戻り値は 59 です。

month-from-date 関数

fn:month-from-date 関数は、xs:date 値の月コンポーネントを戻します。

構文

date-value
月コンポーネントが取り出される日付値。

date-value は xs:date タイプであるか、空のシーケンスです。

戻り値

date-value は xs:date タイプであり、戻り値は xs:integer タイプであり、値の範囲は 1 から 12 です。値は *date-value* の月コンポーネントです。

date-value が空のシーケンスである場合、戻り値は空のシーケンスです。

例

以下の関数は、2005 年 12 月 1 日の日付値の月コンポーネントを戻します。

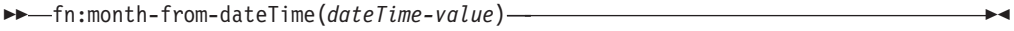
```
fn:month-from-date(xs:date("2005-12-01"))
```

戻り値は 12 です。

month-from-dateTime 関数

fn:month-from-dateTime 関数は、xs:dateTime 値の月コンポーネントを戻します。

構文



dateTime-value
月コンポーネントが取り出される dateTime 値。

dateTime-value は xs:dateTime タイプであるか、空のシーケンスです。

戻り値

dateTime-value は xs:dateTime タイプであり、戻り値は xs:integer タイプであり、値の範囲は 1 から 12 です。値は、*dateTime-value* の月コンポーネントです。

dateTime-value が空のシーケンスである場合、戻り値は空のシーケンスです。

例

以下の関数は、UTC-8 タイム・ゾーンの 2005 年 10 月 31 日の午前 8 時 15 分の dateTime 値の月コンポーネントを戻します。

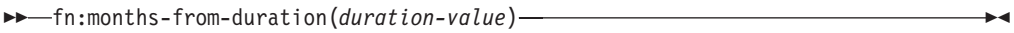
```
fn:month-from-dateTime(xs:dateTime("2005-10-31T08:15:00-08:00"))
```

戻り値は 10 です。

months-from-duration 関数

fn:months-from-duration 関数は、期間値の月コンポーネントを戻します。

構文



duration-value
月コンポーネントが取り出される期間値。

duration-value は空のシーケンスであるか、タイプが xdt:dayTimeDuration、xs:duration、または xdt:yearMonthDuration のいずれかである値です。

戻り値

戻り値は、*duration-value* のタイプによって異なります。

- *duration-value* のタイプが xs:duration または xdt:yearMonthDuration である場合、戻り値は xs:integer であり、値の範囲は -11 から 11 です。値は、xdt:yearMonthDuration としてキャストされる *duration-value* の月コンポーネントです。*duration-value* が負の数である場合、この値も負の数になります。
- *duration-value* がタイプ xdt:dayTimeDuration である場合、戻り値は 0 です。
- *duration-value* が空のシーケンスである場合、戻り値は空のシーケンスです。

xdt:yearMonthDuration としてキャストされる *duration-value* の月コンポーネントは、*duration-value* の総月数を 12 で除算して残る月数 (整数) です。

例

以下の関数は、期間が 20 年と 5 カ月の月コンポーネントを戻します。

```
fn:months-from-duration(xs:duration("P20Y5M"))
```

戻り値は 5 です。

以下の関数は、yearMonthDuration が -9 年と -13 カ月の月コンポーネントを戻します。

```
fn:months-from-duration(xdt:yearMonthDuration("-P9Y13M"))
```

戻り値は -1 です。期間の総月数を計算する場合、-13 カ月は -1 年と -1 カ月に変換されます。この期間は -P10Y1M と等しく、これは -1 カ月の月コンポーネントを持ちます。

以下の関数は、期間が 14 年 11 カ月 40 日 13 時間の月コンポーネントを戻します。

```
xquery fn:months-from-duration(xs:duration("P14Y11M40DT13H"))
```

戻り値は 11 です。

name 関数

fn:name 関数は、ノード名の接頭部およびローカル名の部分を戻します。

構文

node 名前が取得されるノードの修飾名。*node* が指定されていない場合、fn:name は現行のコンテキスト・ノードについて評価されます。

戻り値

戻り値は、*node* の値によって異なります。

- *node* が以下のいずれかの条件と一致する場合、ゼロ長ストリングが戻されます。
  - *node* が空のシーケンスである。
  - *node* がエレメント・ノード、属性ノード、または処理命令ノードではない。
- *node* が以下のいずれかの条件と一致する場合、エラーが戻されます。
  - *node* が未定義である。
  - *node* がノードでない。
- それ以外の場合、*node* の接頭部 (ある場合) およびローカル名が含まれる xs:string 値が戻されます。

例

以下の照会は、値 ″comp:emp″ を戻します。

```
declare namespace d="http://www.mycompany.com";
let $department := document {
    <comp:dept xmlns:comp="http://www.mycompany.com" id="A07">
        <comp:emp id="31201" />
    </comp:dept> }
return fn:name($department/d:dept/d:emp)
```

以下の照会も、値 ″comp:emp″ を戻します。

```
declare namespace d="http://www.mycompany.com";
let $department := document {
    <comp:dept xmlns:comp="http://www.mycompany.com" id="A07">
        <comp:emp id="31201" />
    </comp:dept> }
return $department/d:dept/d:emp/fn:name()
```

namespace-uri 関数

fn:namespace-uri 関数は、ノードの修飾名のネーム・スペース URI を戻します。

構文

node ネーム・スペース URI が取得されるノードの修飾名。 *node* が指定されていない場合、fn:namespace-uri は、現行のコンテキスト・ノードについて評価されます。

戻り値

戻り値は、*node* の値によって異なります。

- *node* が以下のいずれかの条件と一致する場合、ゼロ長ストリングが戻されます。
  - *node* が空のシーケンスである。
  - *node* がエレメント・ノードまたは属性ノードではない。
  - *node* はエレメント・ノードまたは属性ノードであるが、*node* の拡張された修飾名がネーム・スペースにない。

- *node* が以下のいずれかの条件と一致する場合、エラーが戻されます。
  - *node* が未定義である。
  - *node* がノードでない。
- それ以外の場合、*node* の拡張名のネーム・スペース URI が含まれる xs:string 値が戻されます。

例

以下の照会は、値 ″http://www.mycompany.com″ を戻します。

```
declare namespace d="http://www.mycompany.com";
let $department := document {
    <comp:dept xmlns:comp="http://www.mycompany.com" id="A07">
        <comp:emp id="31201" />
    </comp:dept> }
return fn:namespace-uri($department/d:dept/d:emp)
```

以下の照会も、値 ″http://www.mycompany.com″ を戻します。

```
declare namespace d="http://www.mycompany.com";
let $department := document {
    <comp:dept xmlns:comp="http://www.mycompany.com" id="A07">
        <comp:emp id="31201" />
    </comp:dept> }
return $department/d:dept/d:emp/fn:namespace-uri()
```

namespace-uri-for-prefix 関数

fn:namespace-uri-for-prefix 関数は、エレメントの範囲内のネーム・スペースの接頭部と関連付けられたネーム・スペース URI を戻します。

構文

```
►►—fn:namespace-uri-for-prefix(prefix,element)———————————————————►◄
```

prefix ネーム・スペースが戻される接頭部。

prefix はゼロ長にすることができる xs:string データ・タイプであるか、空のシーケンスです。

element

prefix にバインドされた範囲内のネーム・スペースを持つエレメント。

戻り値

戻り値は、*prefix* の値によって異なります。

- *element* が、接頭部値が *prefix* の値と一致する範囲内のネーム・スペースを持つ場合、そのネーム・スペースのネーム・スペース URI が戻されます。
- *element* が、接頭部値が *prefix* の値と一致する範囲内のネーム・スペースを持たない場合、空のシーケンスが戻されます。
- *prefix* がゼロ長ストリングである場合、または空のシーケンスである場合、デフォルトのネーム・スペースであるネーム・スペース URI が戻されます。

例

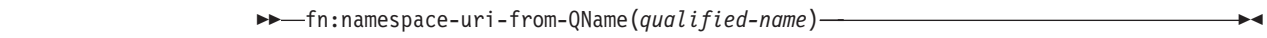
以下の照会は、値 ″http://www.mycompany.com″ を戻します。

```
declare namespace d="http://www.mycompany.com";
let $department := document {
    <comp:dept xmlns:comp="http://www.mycompany.com" id="A07">
        <comp:emp id="31201" />
    </comp:dept> }
return fn:namespace-uri-for-prefix("comp", $department/d:dept/d:emp)
```

namespace-uri-from-QName 関数

fn:namespace-uri-from-QName 関数は、 xs:QName 値のネーム・スペース URI の部分を戻します。

構文



qualified-name
ネーム・スペース URI 部が取得される修飾名。

qualified-name は xs:QName データ・タイプであるか、空のシーケンスです。

戻り値

qualified-name が空のシーケンスでない場合、戻される値は *qualified-name* のネーム・スペース URI 部である xs:string 値です。*qualified-name* がネーム・スペースにない場合、ゼロ長ストリングが戻されます。*qualified-name* が空のシーケンスである場合、空のシーケンスが戻されます。

例

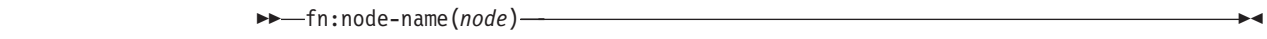
以下の関数は、ストリング値 ″http://www.mycompany.com″ を戻します。

```
fn:namespace-uri-from-QName(fn:QName("http://www.mycompany.com", "comp:employee"))
```

node-name 関数

fn:node-name 関数は、ノードの展開された QName を戻します。

構文



node 拡張名が取得されるノード。

戻り値

戻り値は、*node* の拡張 QName を含む xs:QName 値です。*node* が空のシーケンスである場合、空のシーケンスが戻されます。

例

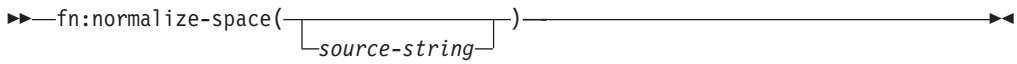
以下の照会は、 URI http://www.mycompany.com および字句 QName comp:emp に対応する拡張 QName を戻します。

```
declare namespace d="http://www.mycompany.com";
let $department := document {
    <comp:dept xmlns:comp="http://www.mycompany.com" id="A07">
        <comp:emp id="31201" />
    </comp:dept> }
return fn:node-name($department/d:dept/d:emp)
```

normalize-space 関数

fn:normalize-space 関数は、ストリングから前後の空白文字を除去し、内部の連続した空白文字をそれぞれ単一のブランク文字に置き換えます。

構文



source-string

空白文字が正規化されるストリング。

source-string は xs:string 値または空のシーケンスです。

source-string が指定されていない場合、fn:normalize-space の引数は、fn:string 関数を使用して xs:string 値に変換される現行のコンテキスト・アイテムです。

戻り値

戻り値は、以下の操作が *source-string* に対して実行された結果の xs:string 値です。

- 前後の空白文字を削除する。
- 1 つ以上の連続する空白文字の各内部シーケンスがシングル・スペース (X'20') 文字で置換される。

空白文字はスペース (X'20')、タブ (X'09')、改行 (X'0A')、および復帰 (X'0D') です。

source-string が空のシーケンスである場合、ゼロ長ストリングが戻されます。

例

以下の関数は、余分な空白文字をストリング "a b c d " から削除します。

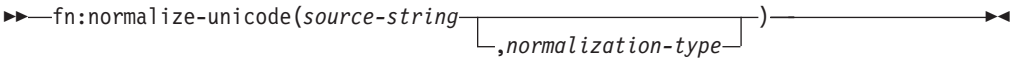
```
fn:normalize-space(" a b c d " )
```

戻り値は "a b c d" です。

normalize-unicode 関数

fn:normalize-unicode 関数は、ストリングに対して Unicode 正規化を実行します。

構文



source-string
Unicode 正規化が実行される値。

source-string は xs:string 値または空のシーケンスです。

normalization-type
実行される Unicode 正規化のタイプを示す xs:string 値。可能な値は以下のとおりです。

NFC Unicode 正規化形式 C。*normalization-type* が指定されていない場合、NFC 正規化が実行されます。

NFD Unicode 正規化形式 D。

NFKC Unicode 正規化形式 KC。

NFKD Unicode 正規化形式 KD。

ゼロ長ストリングが指定された場合、正規化は実行されません。

戻り値

source-string が空のシーケンスでない場合、戻り値は *normalization-type* で指定された Unicode 正規化が *source-string* に対して実行された結果の xs:string 値です。*normalization-type* が指定されていない場合は、*source-string* に対して Unicode 正規化形式 C (NFC) が実行されます。Unicode 正規化については、「*Character Model for the World Wide Web 1.0*」で説明されています。

source-string が空のシーケンスである場合、ゼロ長ストリングが戻されます。

例

以下の関数は、ストリング ″ṃ″ (ラテン語の小文字 m の下にドットが付いたもの) に対して Unicode 正規化形式 C を実行します。

```
fn:normalize-unicode("&#x6d;&#x323;","NFC")
```

戻り値は数値文字参照 &x1e43; で表現される UTF-8 文字 (ラテン語の小文字 m の下にドットが付いたもの) です。

以下の例は、正規化された Unicode を 10 進数のコード・ポイントに変換します。

```
fn:string-to-codepoints(fn:normalize-unicode("&#x6d;&#x323;", "NFC"))
```

戻り値は 7747 です。

not 関数

fn:not 関数は、シーケンスの有効なブール値が true の場合は false を、シーケンスの有効なブール値が false の場合は true を戻します。

構文

sequence-expression
任意のタイプの項目を含むシーケンス、または空のシーケンス。

戻り値

sequence-expression が空のシーケンスでない場合、戻される値は、シーケンスの有効なブール値が false である場合 true になります。戻り値は、シーケンスの有効なブール値が true である場合 false になります。

sequence-expression が空のシーケンスである場合、戻り値は true です。

例

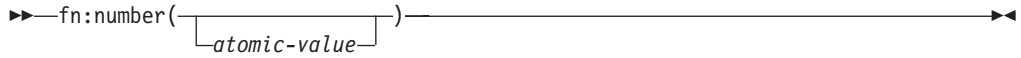
以下の関数は、ノードの有効なブール値が true であるため false を戻します。

```
fn:not(<employee />)
```

number 関数

fn:number 関数は、値を xs:double タイプに変換します。

構文



atomic-value
原子値または空のシーケンス。*atomic-value* が指定されていない場合、fn:number は現行のコンテキスト・アイテムについて評価されます。

戻り値

atomic-value が空のシーケンスでない場合、戻り値は xs:double として *atomic-value* をキャストした結果です。*atomic-value* を xs:double タイプにキャストできない場合、NaN が戻されます。

numeric-value が空のシーケンスである場合、NaN が戻されます。

例

xs:decimal 値を xs:double に変換する例: 以下の関数は xs:decimal 値 2.75 を xs:double に変換します。

```
fn:number(2.75)
```

戻り値は 2.75E0 です。

xs:boolean 値を xs:double に変換する例: 以下の関数は、ブール値 false() を xs:double に変換します。

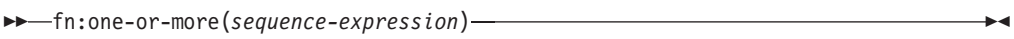
```
fn:number(false())
```

戻り値は 0.0E0 です。

one-or-more 関数

fn:one-or-more 関数は、引数に 1 つ以上の項目が含まれる場合にその引数を戻します。

構文



sequence-expression
任意のシーケンス (空のシーケンスを含む)。

戻り値

sequence-expression が 1 個以上の項目を含む場合、*sequence-expression* が戻されます。1 個以上の項目が含まれない場合、エラーが戻されます。

例

以下の例では、fn:one-or-more 関数を使用して、変数 $seq のシーケンスが 1 個以上の項目を含むかどうかを判別します。

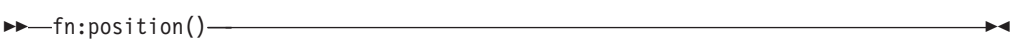
```
let $seq := (5,10)
return fn:one-or-more($seq)
```

(5,10) が戻されます。

position 関数

fn:position 関数は、現在処理中のシーケンス内のコンテキスト・アイテムの位置を戻します。

構文



戻り値

戻り値は、現在処理中のシーケンス内のコンテキスト・アイテムの位置を示す xs:integer 値です。コンテキスト・アイテムが未定義である場合、エラーが戻されます。position 関数は、コンテキスト・アイテムを含むシーケンスが決定論的順序になっている場合にのみ決定論的な結果を戻します。通常、position 関数は述部で使用されます。

例

以下の式では、position 関数は 10 個の項目のあるシーケンスの各項目に対して呼び出されます。 position 関数は、シーケンスにおけるその項目の位置を項目ごとに戻します。述部 `position() eq 5` は、シーケンス内の 5 番目の項目についてのみ true です。

```
(11 to 20)[position() eq 5]
```

式の戻り値は 15 です。

QName 関数

fn:QName 関数は、ネーム・スペース URI および字句の QName (オプションの接頭部を持つ) を含むストリングから、展開名を作成します。

構文

URI 拡張名のネーム・スペース部分。

URI は xs:string タイプであるか、空のストリングまたは空のシーケンスです。

QName

xs:QName タイプの正しい字句形式の値。

QName は xs:string タイプです。

戻り値

戻り値は、*URI* で指定されたネーム・スペース URI、接頭部、および *QName* で指定されたローカル名を持つ拡張名の xs:QName です。

fn:QName 関数は、*QName* のネーム・スペース接頭部を *URI* の値と関連付けます。*QName* がネーム・スペース接頭部を持つ場合、*URI* をゼロ長ストリングまたは空のシーケンスとすることはできません。*QName* がローカル名のみを持ち、接頭部がない場合は、*URI* をゼロ長ストリングまたは空のシーケンスにすることができます。

例

以下の関数では、ネーム・スペース URI および字句 QName を含むストリングが指定されており、xs:QName タイプの値が戻されます。

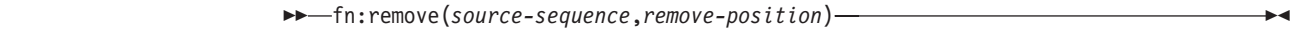
```
fn:QName("http://www.mycompany.com", "comp:employee")
```

戻り値は、ネーム・スペース URI ″http://www.mycompany.com″、接頭部 ″comp″、およびローカル名 ″employee″ を持つ xs:QName 値になります。

remove 関数

fn:remove 関数は、シーケンスから項目を除去します。

構文



source-sequence

項目が削除されるシーケンス。

source-sequence は任意のデータ・タイプの項目のシーケンス、または空のシーケンスです。

remove-position

source-sequence における削除対象の項目の位置。*remove-position* は xs:integer タイプです。

戻り値

source-sequence が空のシーケンスでない場合:

- *remove-position* が 1 より小さい場合、または *source-sequence* の長さより大きい場合、戻り値は *source-sequence* です。
- *remove-position* が 1 以上、*source-sequence* の長さ以下の場合、戻り値は以下の項目の (記載順序どおりの) シーケンスです。
  - 項目 *remove-position* の前にある *source-sequence* の項目。
  - 項目 *remove-position* の後にある *source-sequence* の項目。
- *source-sequence* が空のシーケンスである場合、戻り値は空のシーケンスです。

例

以下の関数は、シーケンス (1,2,4,7) の 3 番目の位置にある項目を削除した結果のシーケンスを戻します。

```
fn:remove((1,2,4,7),3)
```

戻り値は (1,2,7) です。

replace 関数

fn:replace 関数は、ストリング内の文字の各セットを指定されたパターンと比較し、パターンに一致する文字を別の文字のセットに置き換えます。

構文

source-string

置換対象の文字を含むストリング。

source-string は xs:string 値または空のシーケンスです。

pattern *source-string* と比較される正規表現。正規表現は、検索パターン内のストリングまたはストリングのグループを定義する、文字、ワイルドカード、および演算子のセットです。

pattern は xs:string 値です。

replacement-string

source-string の *pattern* と一致する文字を置換する文字を含むストリング。

replacement-string は xs:string 値です。

replacement-string には、変数 $0 から $9 を含めることができます。$0 は *pattern* のストリング全体を示します。変数 $1 から $9 は、*pattern* の 9

つまで指定できる副次式 (括弧で囲まれている) の 1 つを示します。($1 は最初の副次式、$2 は 2 番目の副次式 (以下同様) を示します。)

replacement-string でリテラルのドル記号 ($) を使用するには、ストリング ″¥$″ を使用します。*replacement-string* でリテラルのバックスラッシュ (¥) を使用するには、ストリング ″¥¥″ を使用します。

flags *pattern* と *source-string* のマッチングを制御する、以下のいずれかの値を含むことができる xs:string 値。

s ドット (.) がすべての文字を置換することを示します。

s フラグが指定されていない場合、ドット (.) は改行文字 (X'0A') 以外のすべての文字を置換します。

m 脱字記号 (^) が行の開始 (改行文字の後の位置) を置換し、ドル記号 ($) が行の最後 (改行文字の前の位置) を置換することを示します。

m フラグが指定されていない場合、脱字記号 (^) はストリングの開始を置換し、ドル記号 ($) はストリングの最後を置換します。

i マッチングにおいて大/小文字を区別しないことを示します。

i フラグが指定されていない場合、大/小文字を区別してマッチングが行われます。

x *pattern* 内の空白文字が無視されることを示します。

x フラグが指定されていない場合、空白文字を考慮してマッチングします。

戻り値

source-string が空のシーケンスでない場合、戻り値は、以下の操作が *source-string* に対して実行されたときの結果のストリングです。

- *source-string* が、*pattern* に一致する文字について検索される。*pattern* に複数の文字の代替セットが含まれる場合に、 *source-string* 内の文字に一致する *pattern* 内の最初の文字のセットがマッチング・パターンとして認識される。
- *pattern* と一致する *source-string* の各文字セットが、*replacement-string* に置換される。*replacement-string* に変数 $0 から $9 のいずれかが含まれる場合、その変数に対応する、*pattern* の副次式と一致する *source-string* のサブストリングが *replacement-string* 内の変数を置換します。その後、変更された *replacement-string* が *source-string* に挿入されます。副次式の数以上の変数があるために、変数に対応する副次式が *pattern* にない場合、または副次式に一致するものが*source-string* にない場合、*replacement-string* 内の変数がゼロ長ストリングに置き換えられます。

pattern が *source-string* で検出されない場合は、エラーが戻されます。

source-string が空のシーケンスである場合、ゼロ長ストリングが戻されます。

例

サブストリングを別のサブストリングで置換する例: 以下の関数は、ストリング ″abbcacadbdcd″ 内の ″a″ のインスタンスをすべて ″ba″ に置換します。

```
fn:replace("abbcacadbdcd","a","ba")
```

戻り値は ″babbcbacbadbdcd″ です。

サブストリングを、変数を持つ置き換えストリングを使用して置換する例: 以下の関数は、″abbcacadbdcd″ の ″a″ とその次の文字を、″a″ の次の文字 2 つのインスタンスで置換します。

```
fn:replace("abbcacadbdcd","a(.)","$1$1")
```

戻り値は ″bbbcccddbdcd″ です。

resolve-QName 関数

fn:resolve-QName 関数は、エレメントの範囲内のネーム・スペースを使用して、ネーム・スペース接頭部をネーム・スペース URI に解決することにより、字句の QName を含むストリングを展開された QName に変換します。

構文

qualified-name
: 修飾名の形式のストリング。

  qualified-name は xs:string データ・タイプであるか、空のシーケンスです。

element-for-namespace
: *qualified-name* の範囲内のネーム・スペースを提供するエレメント。

  element-for-namespace はエレメント・ノードです。

戻り値

qualified-name が空のシーケンスでない場合、戻り値は、以下のように構成される拡張名です。

- 拡張された QName の接頭部とローカル名は *qualified-name* から取られます。
- *qualified-name* に接頭部があり、その接頭部が *element-for-namespace* の範囲内のネーム・スペースの接頭部と一致する場合、この接頭部がバインドされるネーム・スペース URI が戻り値のネーム・スペース URI になります。
- *qualified-name* に接頭部がなく、デフォルトのネーム・スペース URI が *element-for-namespace* の範囲内のネーム・スペースで定義されている場合、このデフォルトのネーム・スペース URI が戻り値のネーム・スペース URI になります。
- *qualified-name* に接頭部がなく、デフォルトのネーム・スペース URI が *element-for-namespace* の範囲内のネーム・スペースに定義されていない場合、戻り値にネーム・スペース URI は含まれません。
- *qualified-name* の接頭部と *element-for-namespace* の範囲内のネーム・スペース内のネーム・スペース接頭部が一致しない場合、または *qualified-name* が有効な修飾名の形式ではない場合、エラーが戻されます。

qualified-name が空のシーケンスである場合、空のシーケンスが戻されます。

例

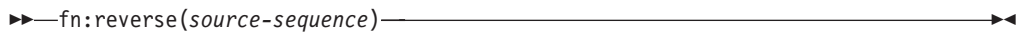
以下の照会は、URI http://www.mycompany.com および字句 QName comp:dept に対応する拡張 QName を戻します。

```
declare namespace d="http://www.mycompany.com";
let $department := document {
<comp:dept xmlns:comp="http://www.mycompany.com" id="A07">
  <comp:emp id="31201" />
</comp:dept> }
return fn:resolve-QName("comp:dept", $department/d:dept/d:emp)
```

reverse 関数

fn:reverse 関数は、シーケンス内の項目の順序を逆にします。

構文



source-sequence
逆順に並べ替えられるシーケンス。

source-sequence は任意のデータ・タイプの項目のシーケンス、または空のシーケンスです。

戻り値

source-sequence が空のシーケンスでない場合、戻り値は、*source-sequence* の項目を逆順に含むシーケンスです。

source-sequence が空のシーケンスである場合、空のシーケンスが戻されます。

例

以下の関数は、シーケンス (1,2,3,7) の項目を逆順で戻します。

```
fn:reverse((1,2,3,7))
```

戻り値は (7,3,2,1) です。

root 関数

fn:root 関数は、ノードが属するツリーのルート・ノードを戻します。

構文

node ノードまたは空のシーケンス。*node* のデフォルト値は、コンテキスト・ノードです。

戻り値

node が空のシーケンスでない場合、戻り値は *node* が属するツリーのルート・ノードです。 *node* がツリーのルート・ノードである場合、戻り値は *node* です。

node が空のシーケンスである場合、戻り値は空のシーケンスです。

例

いくつかの XQuery 変数が以下のように定義されているとします。

```
let $f := <first>Laura</first>
let $e := <emp> {$f} <last>Brown</last> </emp>
let $doc := document {<emps>{$e}</emps>}
```

エレメントのルート・ノードを戻す例: 以下の関数は、last という名前のエレメントのルート・ノードを戻します。

```
fn:root($e/last)
```

戻り値は <emp><first>Laura</first><last>Brown</last></emp> です。

文書のルート・ノードを戻す例: 以下の関数は、変数 `$doc` にバインドされた文書のルート・ノードを戻します。

```
fn:root($doc)
```

戻り値は文書ノードです。

round 関数

fn:round 関数は、指定された数値に最も近い整数を戻します。

構文

numeric-value

原子値または空のシーケンス。

numeric-value が原子値である場合、以下のいずれかのタイプを持ちます。

- xs:float
- xs:double
- xs:decimal
- xs:integer
- xdt:untypedAtomic
- 上記のいずれかのタイプから派生したタイプ

numeric-value が xdt:untypedAtomic タイプである場合、xs:double 値に変換されます。

戻り値

numeric-value が空のシーケンスでない場合、戻り値は *numeric-value* に最も近い整数です。つまり、fn:round(*numeric-value*) は fn:floor(*numeric-value*+0.5) と等価です。戻り値のデータ・タイプは、*numeric-value* のデータ・タイプによって異なります。

- *numeric-value* が xs:float、xs:double、xs:decimal、または xs:integer である場合、戻される値は *numeric-value* と同じタイプになります。
- *numeric-value* が xs:float、xs:double、xs:decimal、または xs:integer から派生したデータ・タイプを持つ場合、戻される値は *numeric-value* の直接の親のデータ・タイプです。

numeric-value が空のシーケンスである場合、戻り値は空のシーケンスです。

例

正の引数を使用した例: 以下の関数は、0.5 の丸め値を戻します。

```
fn:round(0.5)
```

戻り値は 1 です。

負の引数を使用した例: 以下の関数は、(-1.5) の丸め値を戻します。

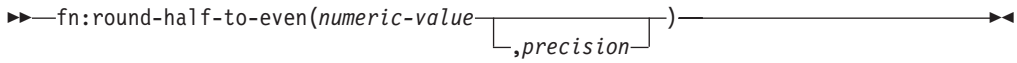
```
fn:round(-1.5)
```

戻り値は -1 です。

round-half-to-even 関数

fn:round-half-to-even 関数は、指定された数値に最も近い、指定された精度の数値を戻します。

構文



numeric-value

原子値または空のシーケンス。

numeric-value が原子値である場合、以下のいずれかのタイプを持ちます。

- xs:float
- xs:double
- xs:decimal
- xs:integer
- xdt:untypedAtomic
- 上記のいずれかのタイプから派生したタイプ

numeric-value が xdt:untypedAtomic タイプである場合、xs:double 値に変換されます。

precision

numeric-value が丸められる小数点の右側の桁数。*precision* は xs:integer 値です。*precision* のデフォルト値は 0 です。

戻り値

numeric-value が空のシーケンスではなく、*precision* が 0 であるか指定されていない場合、戻り値は *numeric-value* に最も近い整数です。*numeric-value* が 2 つの整数のどちらにも等しく近い場合、戻り値は偶数の方になります。

numeric-value が空のシーケンスではなく、*precision* が 0 ではない場合、戻り値は、小数点の右側に *precision* 桁ある、*numeric-value* に最も近い数値になります。*numeric-value* が 2 つの値のどちらにも等しく近い場合、戻り値は最下位桁が偶数の値になります。

戻り値のデータ・タイプは、*numeric-value* のデータ・タイプによって異なります。

- *numeric-value* が xs:float、xs:double、xs:decimal、または xs:integer である場合、戻される値は *numeric-value* と同じタイプになります。
- *numeric-value* が xs:float、xs:double、xs:decimal、または xs:integer から派生したデータ・タイプを持つ場合、戻される値は *numeric-value* の直接の親のデータ・タイプです。

numeric-value が空のシーケンスである場合、戻り値は空のシーケンスです。

例

精度の引数を使用しない例: 以下の関数は、0.5 の丸め値を戻します。

```
fn:round-half-to-even(0.5)
```

戻り値は 0 です。

ゼロ以外の精度引数を使用した例: 以下の関数は、1.5432 を小数点以下 2 桁に丸めた値を戻します。

```
fn:round-half-to-even(1.5432,2)
```

戻り値は 1.54 です。

負の精度を使用した例: 以下の関数は 35600 を戻します。

```
fn:round-half-to-even(35612.25, -2)
```

seconds-from-dateTime 関数

fn:seconds-from-dateTime 関数は、xs:dateTime 値の秒コンポーネントを戻します。

構文

dateTime-value

秒コンポーネントが取り出される dateTime 値。

dateTime-value は xs:dateTime タイプであるか、空のシーケンスです。

戻り値

dateTime-value は xs:dateTime タイプであり、戻り値は xs:decimal タイプであり、値は 0 以上、60 未満です。値は、*dateTime-value* の秒コンポーネント (小数部を含む) です。

dateTime-value が空のシーケンスである場合、戻り値は空のシーケンスです。

例

以下の関数は、UTC-8 タイム・ゾーンの 2005 年 2 月 8 日、午後 2 時 16 分 23 秒の dateTime 値の秒コンポーネントを戻します。

```
fn:seconds-from-dateTime(xs:dateTime("2005-02-08T14:16:23-08:00"))
```

戻り値は 23 です。

以下の関数は、UTC タイム・ゾーンの 2005 年 6 月 23 日、午前 9 時 16 分 20.43 秒の dateTime 値の秒コンポーネントを戻します。

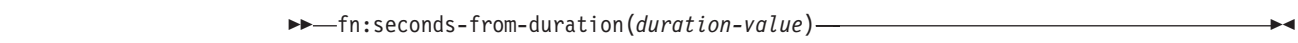
```
fn:seconds-from-dateTime(xs:dateTime("2005-06-23T09:16:23.43Z"))
```

戻り値は 20.43 です。

seconds-from-duration 関数

fn:seconds-from-duration 関数は、期間の秒コンポーネントを戻します。

構文



duration-value
秒コンポーネントが取り出される期間値。

duration-value は空のシーケンスであるか、タイプが xdt:dayTimeDuration、xs:duration、または xdt:yearMonthDuration のいずれかである値です。

戻り値

戻り値は、*duration-value* のタイプによって異なります。

- *duration-value* のタイプが xdt:dayTimeDuration または xs:duration である場合、戻り値は xs:decimal タイプであり、値の範囲は -60 より大きく、60 未満です。値は、xdt:dayTimeDuration としてキャストされる *duration-value* の秒コンポーネント (小数部を含む) です。*duration-value* が負の数である場合、この値も負の数になります。
- *duration-value* が xdt:yearMonthDuration タイプである場合、戻り値は xs:integer タイプであり、0 です。
- *duration-value* が空のシーケンスである場合、戻り値は空のシーケンスです。

xdt:dayTimeDuration としてキャストされる *duration-value* の秒コンポーネント (小数部を含む) は、(*S* mod 60) として計算されます。値 *S* は、年および月コンポー

ネントを除去するための、xdt:dayTimeDuration としてキャストされる *duration-value* の総秒数 (小数部を含む) です。

例

以下の関数は、期間が 150.5 秒の期間の秒コンポーネントを戻します。

```
fn:seconds-from-duration(xdt:dayTimeDuration("PT150.5S"))
```

戻り値は 30.5 です。期間の総秒数を計算する場合、150.5 秒は 2 分と 30.5 秒に変換されます。この期間は PT2M30.5S と等しく、30.5 秒の秒コンポーネントを持ちます。

seconds-from-time 関数

fn:seconds-from-time 関数は、xs:time 値の秒コンポーネントを戻します。

構文

time-value
秒コンポーネントが取り出される時刻値。

time-value は xs:time タイプであるか、空のシーケンスです。

戻り値

time-value は xs:time タイプであり、戻り値は xs:decimal タイプであり、値は 0 以上、60 未満です。値は、*time-value* の秒コンポーネント (小数部を含む) です。

time-value が空のシーケンスである場合、戻り値は空のシーケンスです。

例

以下の関数は、UTC-8 タイム・ゾーンの午前 8 時 59 分 59 秒の時刻値の秒コンポーネントを戻します。

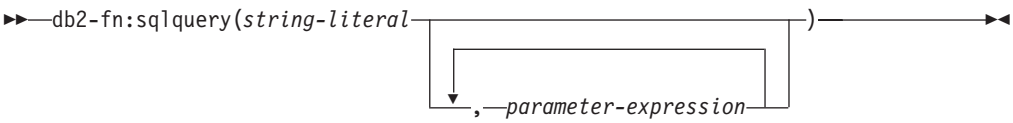
```
fn:seconds-from-time(xs:time("08:59:59-08:00"))
```

戻り値は 59 です。

sqlquery 関数

db2-fn:sqlquery 関数は、現在接続されている DB2 データベースで、 SQL 全選択の結果であるシーケンスを検索します。

構文



string-literal

fullselect を含みます。fullselect では、単一列の結果セットを指定する必要があり、その列には XML タイプが含まれる必要があります。fullselect の有効範囲は、ネストされた SQL 照会ではなく新規 SQL 照会の有効範囲です。

fullselect に単一引用符が含まれる場合 (例えば、ストリング定数の周囲) は、関数の引数を二重引用符で囲みます。例:

```
"select c1 from t1 where c2 = 'Hello'"
```

fullselect に二重引用符が含まれる場合 (例えば、区切り ID の周囲) は、関数の引数を単一引用符で囲みます。例:

```
'select c1 from "t1" where c2 = 47'
```

fullselect に単一引用符と二重引用符の両方が含まれる場合は、関数の引数を単一引用符で囲み、内部の各単一引用符は 2 つの連続する単一引用符で表します。例:

```
'select c1 from "t1" where c2 = ''Hello'''
```

fullselect には、db2-fn:sqlquery 関数呼び出しで指定された各 *parameter-expression* からの結果値を参照するために、PARAMETER 関数の呼び出しを含めることができます。 PARAMETER 関数呼び出しは、fullselect の実行時に、対応する *parameter-expression* の結果値で置き換えられます。

parameter-expression

値を戻す XQuery 式。各 *parameter-expression* の結果値は、SQL 全選択内で、指定する SQL 関数 PARAMETER と整数値の引数によって参照することができます。整数値は、*parameter-expression* の索引であり、db2-fn:sqlquery 関数呼び出しでのその位置による索引です。有効な整数値の範囲は、1 から、関数呼び出しでの *parameter-expression* の総数までです。例えば、*string-literal* 引数内の SQL 全選択に `PARAMETER(1)` および `PARAMETER(2)` が組み込まれている場合は、関数呼び出しで 2 つの XQuery *parameter-expression* 引数が指定されていなければなりません。`PARAMETER(1)` は最初の *parameter-expression* 引数の結果を参照し、`PARAMETER(2)` は 2 番目の *parameter-expression* 引数の結果を参照します。

SQL 全選択の処理中、各 PARAMETER 関数呼び出しは、db2-fn:sqlquery 関数呼び出しにおける対応する *parameter-expression* の結果値で置き換えられます。 *parameter-expression* は、SQL 全選択内で何回参照されていたとしても、それぞれ 1 回ずつ評価されます。

XMLCAST の規則では、対応する *parameter-expression* の結果データ・タイプは PARAMETER 関数の結果タイプにキャストできなければなりません。そうでないと、エラーが戻されます。

PARAMETER 関数の結果タイプは、SQL 全選択内のパラメーター・マーカーと同様に判別されます。例えば、パラメーター・マーカーは、他のコンテキストでは疑問符 (?) によって示されます。PARAMETER 関数の結果タイプを判別できない場合、エラーが戻されます。

ヒント: 演算で非型付きパラメーター・マーカーを使用できない場合、CAST 指定または XMLCAST 指定を使用してタイプを指定することができ

ます。例えば、PARAMETER(1) を DOUBLE タイプにキャストするには、CAST(PARAMETER(1) as double) という CAST 指定を使用します。

戻り値

戻り値は、 *string-literal* 内の fullselect の結果であるシーケンスです。fullselect は、許可およびネーム解決に関する通常の動的 SQL の規則に従って SQL ステートメントとして処理されます。 fullselect に PARAMETER 関数の呼び出しが含まれる場合、それらは fullselect の評価時に、対応する *parameter-expression* 引数の XQuery 式の結果値と置き換えられます。 fullselect によって戻される XML 値が連結され、関数の結果が形成されます。NULL 値を含む行は、結果シーケンスに影響を与えません。fullselect により行が戻されない場合、または NULL 値のみが戻される場合は、関数の結果は空のシーケンスになります。

db2-fn:sqlquery 関数によって戻されるシーケンス内の項目数が、fullselect によって戻される行数と異なる場合があります。これは、これらの行の中に NULL 値や、複数の項目からなるシーケンスが含まれる場合があるためです。

例

XML 文書のシーケンスを戻す fullselect の例: 以下の例は、表 PRODUCT から同じ文書のシーケンスを戻すいくつかの関数呼び出しを示しています。文書は、列 DESCRIPTION にあります。

以下のすべての関数により、同じ結果が作成されます。

```
db2-fn:sqlquery('select description from product')
db2-fn:sqlquery('SELECT DESCRIPTION FROM PRODUCT')
db2-fn:sqlquery('select "DESCRIPTION" from "PRODUCT"')
```

単一の XML 文書を戻す fullselect の例: 以下の例は、表 PRODUCT 内の単一の文書であるシーケンスを戻します。文書は列 DESCRIPTION にあり、列 PID の「100-103-01」の値によって識別されます。

以下のすべての関数により、同じ結果が作成されます。

```
db2-fn:sqlquery('select Description from Product where pID=''100-103-01''')
db2-fn:sqlquery("select description from product where pid='100-103-01'")
db2-fn:sqlquery("select ""DESCRIPTION"" from product where pid='100-103-01'")
```

2 つの PARAMETER 関数呼び出しと 1 つの式を使った fullselect の例: 以下の例は、販売促進期間中に販売されたすべての部品に関する購入 ID、部品 ID、および購入日を戻します。

```
xquery
declare default element namespace "http://posample.org";
for $po in db2-fn:xmlcolumn('PURCHASEORDER.PORDER')/PurchaseOrder,
  $item in $po/item/partid
for $p in db2-fn:sqlquery(
  "select description
  from product
  where promostart < parameter(1)
  and  promoend  >  parameter(1)",
  $po/@OrderDate )
where $p//@pid = $item
return
<RESULT>
```

```
  <PoNum>{data($po/@PoNum)}</PoNum>
  <PartID>{data($item)} </PartID>
  <PoDate>{data($po/@OrderDate)}</PoDate>
</RESULT>
```

db2-fn:sqlquery 関数の処理中、parameter(1) への参照の両方が、注文日の属性の値 $po/@OrderDate を戻します。

2 つの PARAMETER 関数呼び出しと 2 つの式を使った fullselect の例: 以下の例は、DB2 SAMPLE データベースの PURCHASEORDER 表を使用します。XQuery 式は、注文日が 2006 年 4 月 4 日より前になっている未配送の購入注文を検索し、各購入注文の個々の部品番号をリストします。

```
xquery
declare default element namespace 'http://posample.org' ;
let $status := ( "Unshipped" ), $date := ( "2006-04-04" )
for $myorders in db2-fn:sqlquery(
   "select porder from purchaseorder
   where status = parameter(1)
   and orderdate < parameter(2)",
   $status, $date )
return
<LateOrder>
  <PoNum>
  {data($myorders/PurchaseOrder/@PoNum)}
  </PoNum>
  <PoDate>
  {data($myorders/PurchaseOrder/@OrderDate)}
  </PoDate>
  <Items>
     {for $itemID in distinct-values( $myorders/PurchaseOrder/item/partid )
     return
     <PartID>
     {$itemID}
     </PartID>}
   </Items>
</LateOrder>
```

db2-fn:sqlquery 関数の処理中、parameter(1) の参照は式 $status の結果値を戻し、parameter(2) の参照は式 $date の結果値を戻します。

SAMPLE データベースに対して実行すると、この式は以下の結果を戻します。

```
<LateOrder xmlns="http://posample.org">
  <PoNum>5000</PoNum>
  <PoDate>2006-02-18</PoDate>
  <Items>
    <PartID>100-100-01</PartID>
    <PartID>100-103-01</PartID>
  </Items>
</LateOrder>
```

starts-with 関数

fn:starts-with 関数は、ストリングが、指定されたサブストリングで開始しているかどうかを判別します。サブストリングはデフォルトの照合を使用して突き合わされます。

構文

string *substring* を検索するストリング。

string は、xs:string データ・タイプであるか、空のシーケンスです。*string* が空のシーケンスの場合、*string* はゼロ長ストリングに設定されます。

substring

string の始まりの部分を検索するためのサブストリング。

substring は、xs:string データ・タイプであるか、空のシーケンスです。

戻り値

戻り値は、以下の条件のいずれかが満たされた場合、xs:boolean 値 true になります。

- *substring* が *string* の先頭に出現する。
- *substring* が空のシーケンスまたはゼロ長ストリングである。

それ以外の場合、戻り値は false です。

例

以下の関数は、ストリング 'Test literal' がストリング 'lite' で始まっているかどうかを判別します。

```
fn:starts-with('Test literal','lite')
```

戻り値は false です。

string 関数

fn:string 関数は、値のストリング表記を戻します。

構文

value ストリングとして表される値。

value は、ノードまたは原子値、あるいは空のシーケンスです。

value が指定されていない場合、fn:string は、現行コンテキストの項目について評価されます。現行コンテキストの項目が未定義の場合は、エラーが戻されます。

戻り値

value が空のシーケンスでない場合:

- *value* がノードの場合、戻り値はノードのストリング値です。

- *value* が原子値の場合、戻り値は *value* を xs:string タイプにキャストした結果です。

value が空のシーケンスの場合、結果はゼロ長ストリングです。

例

以下の関数は、123 のストリング表記を戻します。

```
fn:string(xs:integer(123))
```

戻り値は '123' です。

string-join 関数

fn:string-join 関数は、分離文字で分離された項目を連結させることによって生成されるストリングを戻します。

構文

sequence
ストリングを形成するために連結される項目のシーケンス。

sequence は、 xs:string 値の任意のシーケンス、または空のシーケンスです。

separator
結果のストリングの *sequence* からの項目間に挿入される区切り文字。

separator のデータ・タイプは xs:string です。

戻り値

戻り値は、*sequence* 内の項目が *separator* で区切られて連結されたストリングです。*separator* がゼロ長ストリングである場合、*sequence* 内の項目は区切り記号なしで連結されます。*sequence* が空のシーケンスである場合、ゼロ長ストリングが戻されます。

例

以下の関数は、シーケンス ("I", "made", "a", "sentence!") 内の項目を連結し、区切り記号として空白文字を使用した結果のストリングを戻します。

```
fn:string-join(("I" , "made", "a", "sentence!"), " ")
```

戻り値は、ストリング "I made a sentence!" です。

string-length 関数

fn:string-length 関数は、ストリングの長さを戻します。

構文

source-string
その長さが戻されるストリング。

source-string は、xs:string データ・タイプであるか、空のシーケンスです。

戻り値

source-string が空のシーケンスでない場合、戻り値は文字列の中の *source-string* の長さです。xFFFF より上のコード・ポイントは、サロゲート・ペアと呼ばれる 2 つの 16 ビット値を使用し、ストリングの長さにおいて 1 文字としてカウントされます。*source-string* は、 xs:integer 値です。

source-string が空のシーケンスの場合、戻り値は 0 です。

例

以下の関数は、ストリング 'Test literal' の長さを戻します。

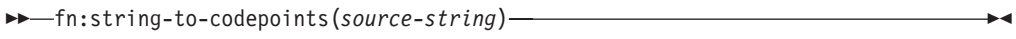
```
fn:string-length('Test literal')
```

戻り値は 12 です。

string-to-codepoints 関数

fn:string-to-codepoints 関数は、ストリング値に相当する Unicode コード・ポイントのシーケンスを戻します。

構文



source-string
各文字の Unicode コード・ポイントが戻されるストリング値、または空のシーケンス。

戻り値

source-string が空のシーケンスではなく、長さがゼロでない場合、戻り値は、*source-string* の文字のコード・ポイントを示す xs:integer 値のシーケンスです。

source-string が空のシーケンスであるかまたは長さがゼロの場合、戻り値は空のシーケンスです。

例

以下の関数は、ストリング 'XQuery' の文字を示すコード・ポイントのシーケンスを戻します。

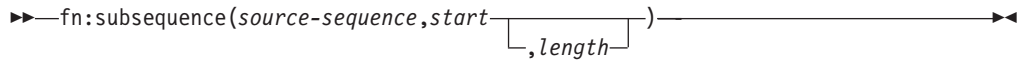
```
fn:string-to-codepoints("XQuery")
```

戻り値は (88,81,117,101,114,121) です。

subsequence 関数

fn:subsequence 関数は、シーケンスのサブシーケンスを戻します。

構文



source-sequence
: サブシーケンスが取得されるシーケンス。

  source-sequence は、任意のシーケンス (空のシーケンスも含む) です。

start
: サブシーケンスの *source-sequence* での開始位置。*source-sequence* の最初の位置は 1 です。*start*<=0 の場合、*start* は 1 に設定されます。

  start は xs:double タイプです。

length
: サブシーケンス内の項目数。*length* のデフォルトは、*source-sequence* の項目数です。*start*+*length*-1 が *source-sequence* の長さより大きい場合、*length* は (*source-sequence* の長さ)-*start*+1 に設定されます。

  length は xs:double タイプです。

戻り値

source-sequence が空のシーケンスでない場合、戻り値は、*start* の位置で始まり、*length* 個の項目を含む *source-sequence* のサブシーケンスです。

source-sequence が空のシーケンスである場合、空のシーケンスが戻されます。

例

以下の関数は、シーケンス ('T','e','s','t',' ','s','e','q','u','e','n','c','e') の 6 番目の項目から開始する 3 つの項目を戻します。

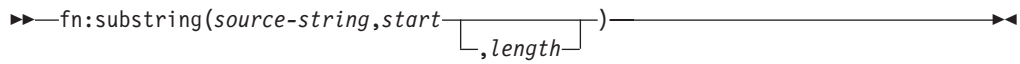
```
fn:subsequence(('T','e','s','t',' ','s','e','q','u','e','n','c','e'),6,3)
```

戻り値は ('s','e','q') です。

substring 関数

fn:substring 関数は、ストリングのサブストリングを戻します。

構文



source-string
: サブストリングが取得されるストリング。

  source-string は、xs:string データ・タイプであるか、空のシーケンスです。

start サブストリングの *source-string* の開始文字の位置。*source-string* の最初の位置は 1 です。*start*<=0 の場合、*start* は 1 に設定されます。サロゲート・ペアと呼ばれる 2 つの 16 ビット値を使用する xFFFF より上のコード・ポイントは、1 文字としてカウントされます。

start は xs:double タイプです。

length サブストリングの長さ (文字数)。*length* のデフォルトは、*source-string* の長さです。*start*+*length*-1 が *source-string* の長さよりも大きい場合、*length* は (*source-string* の長さ)-*start*+1 に設定されます。 xFFFF より上のコード・ポイントは、サロゲート・ペアと呼ばれる 2 つの 16 ビット値を使用し、ストリングの長さにおいて 1 文字としてカウントされます。

length は xs:double タイプです。

戻り値

source-string が空のシーケンスでない場合、戻り値は文字位置 *start* で開始し、*length* 個の文字を含む *source-string* のサブストリングです。 *source-string* が空のシーケンスである場合、結果はゼロ長ストリングです。

例

以下の関数は、ストリング 'Test literal' の 6 番目の文字で開始する 7 文字を戻します。

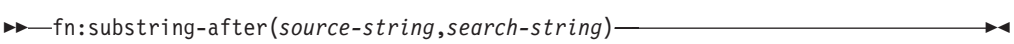
```
fn:substring('Test literal',6,7)
```

戻り値は 'literal' です。

substring-after 関数

fn:substring-after 関数は、指定された検索ストリングの最初の出現の終了後にストリング内に出現するサブストリングを戻します。検索ストリングはデフォルトの照合を使用して突き合わされます。

構文



source-string

サブストリングが取得されるストリング。

source-string は、xs:string データ・タイプであるか、空のシーケンスです。*source-string* が空のシーケンスである場合、*source-string* はゼロ長ストリングに設定されます。

search-string

source-string での最初の出現を検索される検索対象ストリング。

search-string は xs:string データ・タイプであるか、空のシーケンスです。

戻り値

source-string が空のシーケンスまたはゼロ長ストリングでない場合:

- *source-string* の長さが *n* であり、*m*<*n* であると仮定します。 *search-string* が *source-string* で検出され、*source-string* での *search-string* の最初の出現の最後が *m* の位置である場合、戻り値は、*m*+1 の位置から始まり、*source-string* の *n* の位置で終わるサブストリングです。
- *source-string* の長さが *n* であるとします。*search-string* が *source-string* で検出され、*source-string* での *search-string* の最初の出現の最後が *n* の位置である場合、戻り値はゼロ長ストリングです。
- *search-string* が空のシーケンスまたはゼロ長ストリングである場合、戻り値は *source-string* です。
- *search-string* が *source-string* で検出されない場合、戻り値はゼロ長ストリングです。

source-string が空のシーケンスまたはゼロ長ストリングである場合、戻り値はゼロ長ストリングです。

例

以下の関数は、デフォルトの照合を使用して、ストリング 'DEFABCD' で 'ABC' の後の文字を検索します。

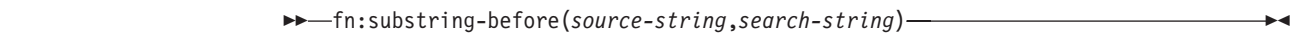
```
fn:substring-after('DEFABCD', 'ABC')
```

戻り値は 'D' です。

substring-before 関数

fn:substring-before 関数は、指定された検索ストリングが最初に出現する前にストリング内に出現するサブストリングを戻します。検索ストリングはデフォルトの照合を使用して突き合わされます。

構文



source-string
サブストリングが取得されるストリング。

source-string は、xs:string データ・タイプであるか、空のシーケンスです。*source-string* が空のシーケンスである場合、*source-string* はゼロ長ストリングに設定されます。

search-string
source-string での最初の出現を検索される検索対象ストリング。

search-string は xs:string データ・タイプであるか、空のシーケンスです。

戻り値

source-string が空のシーケンスまたはゼロ長ストリングでない場合:

- *search-string* が *source-string* の *m* の位置で検出され、*m*>1 である場合、戻り値は *source-string* の 1 の位置から始まり *m* の位置で終わるサブストリングです。

- *search-string* が *source-string* の 1 の位置で検出された場合、戻り値はゼロ長ストリングです。
- *search-string* が空のシーケンスまたはゼロ長ストリングである場合、戻り値はゼロ長ストリングです。
- *search-string* が *source-string* で検出されない場合、戻り値はゼロ長ストリングです。

source-string が空のシーケンスまたはゼロ長ストリングである場合、戻り値はゼロ長ストリングです。

例

以下の関数は、デフォルトの照合を使用して、ストリング 'DEFABCD' の 'ABC' の前にある文字を検索します。

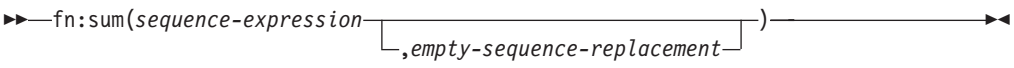
```
fn:substring-before('DEFABCD', 'ABC')
```

戻り値は 'DEF' です。

sum 関数

fn:sum 関数は、シーケンス内の値の合計を戻します。

構文



sequence-expression

以下の原子タイプのいずれかの項目を含むシーケンス、または空のシーケンス。

- xs:float
- xs:double
- xs:decimal
- xs:integer
- xdt:untypedAtomic
- xdt:dayTimeDuration
- xdt:yearMonthDuration
- 上記のいずれかのタイプから派生したタイプ

xdt:untypedAtomic タイプの入力項目は、xs:double にキャストされます。このキャスト後、入力シーケンスに含まれるすべての項目は、プロモーションまたはサブタイプの置換によって共通のタイプに変換可能である必要があります。合計はこの共通のタイプで計算されます。例えば、入力シーケンスに money タイプ (xs:decimal から派生) および stockprice タイプ (xs:float から派生) の項目が含まれる場合、合計は xs:float タイプで計算されます。

empty-sequence-replacement

sequence-expression が空のシーケンスである場合に戻される値。

empty-sequence-replacement は、*sequence-expression* についてリストされているデータ・タイプの 1 つにできます。

戻り値

sequence-expression が空のシーケンスでない場合、戻り値は *sequence-expression* 内の値の合計です。戻り値のデータ・タイプは、*sequence-expression* の項目のデータ・タイプ、または *sequence-expression* の項目がプロモートされるデータ・タイプと同じです。

sequence-expression が空のシーケンスで、*empty-sequence-replacement* が指定されていない場合、fn:sum は 0.0E0 を戻します。*sequence-expression* が空のシーケンスで、 *empty-sequence-replacement* が指定されている場合、fn:sum は *empty-sequence-replacement* を戻します。

例

以下の関数は、シーケンス (500, 1.0E2, 40.5) の合計を戻します。

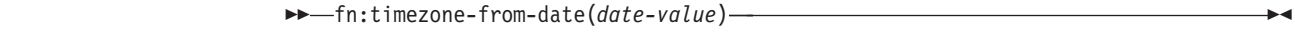
```
fn:sum((500, 1.0E2, 40.5))
```

値は xs:double タイプにプロモートされます。関数は、 xs:double 値 6.405E2 を戻し、これは「640.5」としてシリアライズされます。

timezone-from-date 関数

fn:timezone-from-date 関数は、xs:date 値のタイム・ゾーン・コンポーネントを戻します。

構文



date-value
: タイム・ゾーン・コンポーネントが取り出される日付値。

  date-value は xs:date タイプであるか、空のシーケンスです。

戻り値

date-value が xs:date タイプで、明示的なタイム・ゾーン・コンポーネントを含んでいる場合、戻り値は xdt:dayTimeDuration であり、値の範囲は -PT14H 時間から PT14H 時間です。値は、UTC タイム・ゾーンからの *date-value* タイム・ゾーン・コンポーネントの偏差です。

date-value が明示的なタイム・ゾーン・コンポーネントを持っていないか、空のシーケンスである場合、戻り値は空のシーケンスです。

例

以下の関数は、UTC-8 タイム・ゾーンの 2007 年 3 月 13 日の日付値のタイム・ゾーン・コンポーネントを戻します。

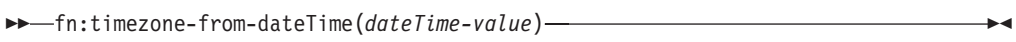
```
fn:timezone-from-date(xs:date("2007-03-13-08:00"))
```

戻り値は -PT8H です。

timezone-from-dateTime 関数

fn:timezone-from-dateTime 関数は、xs:dateTime 値のタイム・ゾーン・コンポーネントを戻します。

構文



dateTime-value
タイム・ゾーン・コンポーネントが取り出される dateTime 値。

dateTime-value は xs:dateTime タイプであるか、空のシーケンスです。

戻り値

dateTime-value が xs:dateTime タイプで、明示的なタイム・ゾーン・コンポーネントを含んでいる場合、戻り値は xdt:dayTimeDuration であり、値の範囲は -PT14H から PT14H です。値は、UTC タイム・ゾーンからの *dateTime-value* タイム・ゾーン・コンポーネントの偏差です。

dateTime-value が明示的なタイム・ゾーン・コンポーネントを持っていないか、空のシーケンスである場合、戻り値は空のシーケンスです。

例

以下の関数は、UTC-8 タイム・ゾーンの 2005 年 10 月 30 日、午前 7 時 30 分の dateTime 値のタイム・ゾーン・コンポーネントを戻します。

```
fn:timezone-from-dateTime(xs:dateTime("2005-10-30T07:30:00-08:00"))
```

戻り値は -PT8H です。

以下の関数は、UTC+10:30 タイム・ゾーンの 2005 年 1 月 1 日、午後 2 時 30 分の dateTime 値のタイム・ゾーン・コンポーネントを戻します。

```
fn:timezone-from-dateTime(xs:dateTime("2005-01-01T14:30:00+10:30"))
```

戻り値は PT10H30M です。

timezone-from-time 関数

fn:timezone-from-time 関数は、xs:time 値のタイム・ゾーン・コンポーネントを戻します。

構文

time-value
タイム・ゾーン・コンポーネントが取り出される時刻値。

time-value は xs:time タイプであるか、空のシーケンスです。

戻り値

time-value が xs:time タイプで、明示的なタイム・ゾーン・コンポーネントを含んでいる場合、戻り値は xdt:dayTimeDuration であり、値の範囲は -PT14H から PT14H です。値は、UTC タイム・ゾーンからの *time-value* タイム・ゾーン・コンポーネントの偏差です。

time-value が明示的なタイム・ゾーン・コンポーネントを持っていないか、空のシーケンスである場合、戻り値は空のシーケンスです。

例

以下の関数は、UTC-5 タイム・ゾーンの正午 12 時の時刻値のタイム・ゾーン・コンポーネントを戻します。

```
fn:timezone-from-time(xs:time("12:00:00-05:00"))
```

戻り値は -PT5H です。

以下の関数では、午後 1 時の時刻値はタイム・ゾーン・コンポーネントを持っていません。

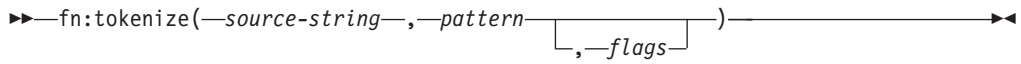
```
fn:timezone-from-time(xs:time("13:00:00"))
```

戻り値は空のシーケンスです。

tokenize 関数

fn:tokenize 関数は、ストリングを切断して、サブストリングのシーケンスを作成します。

構文



source-string
サブストリングのシーケンスに分割されるストリング。

source-string は xs:string 値または空のシーケンスです。

pattern *source-string* 内のサブストリング間の区切り文字。

pattern は、正規表現を含む xs:string 値です。正規表現は、検索パターン内のストリングまたはストリングのグループを定義する、文字、ワイルドカード、および演算子のセットです。

flags *source-string* 内の文字に *pattern* を一致させる方法を制御する以下のいずれかの値を使用可能な xs:string 値:

s 正規表現の中のドット (.) が、改行文字 (X'0A') を含むすべての文字と一致することを示します。

s フラグが指定されていない場合、ドット (.) は、改行文字 (X'0A') を除くすべての文字と一致します。

m 脱字記号 (^) を行の始まり (改行文字の後の位置) に、およびドル記号 ($) を行の終わり (改行文字の前の位置) に一致させることを指示します。

m フラグが指定されていない場合、脱字記号 (^) はストリングの開始と一致し、ドル記号 ($) はストリングの最後と一致します。

i マッチングにおいて大/小文字を区別しないことを示します。

i フラグが指定されていない場合、大/小文字を区別してマッチングが行われます。

x *pattern* 内の空白文字が無視されることを示します。

x フラグが指定されていない場合、空白文字を考慮してマッチングします。

戻り値

source-string が空のシーケンスまたはゼロ長ストリングでない場合、戻り値は、*source-string* に対して以下の操作が実行された場合の結果であるシーケンスです。

- *source-string* が、*pattern* に一致する文字について検索される。
- *pattern* に複数の文字の代替セットが含まれる場合に、 *source-string* 内の文字に一致する *pattern* 内の最初の文字のセットがマッチング・パターンとして認識される。
- *pattern* に一致しない文字の各セットが、結果シーケンス内の項目になる。
- *pattern* が *source-string* の始まりの文字と一致する場合、戻されるシーケンス内の最初の項目はゼロ長ストリングである。
- *source-string* 内に *pattern* について 2 つの連続する一致が検出される場合、ゼロ長ストリングがシーケンスに追加される。
- *pattern* が *source-string* の終わりの文字と一致する場合、戻されるシーケンス内の最後の項目はゼロ長ストリングである。

pattern が *source-string* で検出されない場合は、エラーが戻されます。

source-string が空のシーケンスまたはゼロ長ストリングの場合、結果は空のシーケンスです。

例

以下の関数を実行すると、ストリング ″Tokenize this sentence, please.″ からシーケンスが作成されます。″¥s+″ は、1 つ以上の空白文字を示す正規表現です。

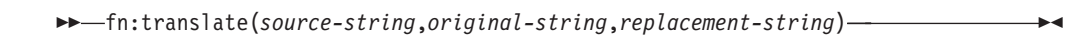
```
fn:tokenize("Tokenize this sentence, please.", "¥s+")
```

戻り値は、シーケンス (″Tokenize″, ″this″, ″sentence,″, ″please.″) です。

translate 関数

fn:translate 関数は、ストリング内の選択された文字を置換文字に置き換えます。

構文



source-string
その中に含まれる文字が変換されるストリング。

source-string は、xs:string データ・タイプであるか、空のシーケンスです。

original-string
変換可能な文字を含むストリング。

original-string は、xs:string タイプです。

replacement-string
original-string 内の文字を置き換える文字を含むストリング。

replacement-string は、xs:string タイプです。

replacement-string が *original-string* より長い場合、*replacement-string* 内の追加の文字は無視されます。

戻り値

source-string が空のシーケンスでない場合、戻り値は、以下の操作が実行された場合の結果である xs:string 値です。

- *original-string* に含まれる *source-string* 内の各文字については、*source-string* 内の文字を、*original-string* 内の文字と同じ位置にある *replacement-string* 内の文字で置き換えます。文字はバイナリー比較を使用して突き合わされます。

  original-string が *replacement-string* より長い場合は、*original-string* に含まれる *source-string* 内の各文字を削除しますが、*original-string* 内の文字位置は、*replacement-string* 内の位置に対応しません。

  original-string 内に 1 つの文字が複数回使用される場合は、*original-string* 内でその文字が最初に使用される位置により、使用される *replacement-string* 内の文字が決定されます。
- *original-string* に含まれない *source-string* 内の各文字については、文字をそのままの状態にします。

source-string が空のシーケンスである場合、ゼロ長ストリングが戻されます。

例

以下の関数は、ストリング 'Test literal' 内の e を o に、l を m に置き換えた結果を戻します。

```
fn:translate('Test literal','el','om')
```

戻り値は 'Tost mitoram' です。

以下の関数は、ストリング・リテラル 'Another test literal' 内の A を B に、t を f に、e を i に、および r を m に置き換えた結果のストリングを戻します。

```
fn:translate('Another test literal', 'Ater', 'Bfim')
```

戻り値は 'Bnofhim fisf lifimal' です。

true 関数

fn:true 関数は、xs:boolean 値 true を戻します。

構文

戻り値

戻り値は xs:boolean 値の true です。

例

true 関数を使用して、値 true を戻します。

```
fn:true()
```

値 true が戻されます。

unordered 関数

fn:unordered 関数は、シーケンス内の項目を非決定論的順序で戻します。

構文

sequence-expression
任意のシーケンス (空のシーケンスを含む)。

戻り値

戻り値は、非決定論的順序の *sequence-expression* 内の項目です。これは、照会オプティマイザーによる、シーケンス内の項目の順序に依存しないアクセス・パスの選択を支援します。

例

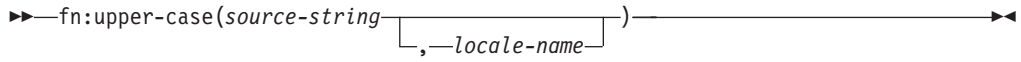
以下の関数は、シーケンス (1,2,3) 内の項目を非決定論的順序で戻します。

```
fn:unordered((1,2,3))
```

upper-case 関数

fn:upper-case 関数は、ストリングを大文字に変換します。

構文



source-string
大文字に変換されるストリング。

source-string は、xs:string データ・タイプであるか、空のシーケンスです。

locale-name
大文字の操作に使用するロケールを含むストリング。

locale-name は xs:string タイプであるか、空のシーケンスです。*locale-name* が空のシーケンスでない場合は、*locale-name* の値に大文字と小文字の区別はなく、有効なロケールまたは長さゼロのストリングでなければなりません。

戻り値

source-string が空のシーケンスでない場合、戻り値は、各文字が対応する大文字に変換された *source-string* です。*locale-name* が指定されていないか、空のシーケンスか、または長さゼロのストリングの場合は、Unicode 規格で定義されている大文字の規則が使用されます。それ以外の場合は、指定されたロケールの大文字の規則が使用されます。対応する大文字がないすべての文字は、元の形式で戻り値に含まれます。

source-string が空のシーケンスである場合、戻り値はゼロ長ストリングです。

例

以下の関数を実行すると、ストリング 'Test literal 1' が大文字に変換されます。

```
fn:upper-case('Test literal 1')
```

戻り値は 'TEST LITERAL 1' です。

以下の関数は、トルコ語のロケール tr_TR を指定し、文字 ″i″ と数値による文字参照 ı (ドットのないローマ字の小文字 i の文字参照) を変換します。

```
fn:upper-case("i&#x131;", "tr_TR")
```

戻り値は、İ (ドットが上に付いたローマ字の大文字 I) と文字 ″I″ の 2 つの文字で構成されます。トルコ語のロケールの場合、文字 ″i″ は İ (ドットが上に付いたローマ字の大文字 I) で表される文字に変換され、ı (ドットのないローマ字の小文字 i) は文字 ″I″ に変換されます。

以下の関数は、ロケールを指定せず、Unicode 規格で定義されている規則を使用して 2 つの文字を大文字に変換します。

```
fn:upper-case("&#x131;i")
```

この関数は文字 ″II″ を戻します。 fn:upper-case は小文字 ı と文字 ″i″ の両方を大文字 ″I″ に変換します。

xmlcolumn 関数

db2-fn:xmlcolumn 関数は、現在接続されている DB2 データベース内の列からシーケンスを検索します。

構文

string-literal
: シーケンスの検索先の列の名前を指定します。列名は、表名、ビュー名、または別名で修飾する必要があり、 XML タイプの列を参照する必要があります。 SQL スキーマ名はオプションです。SQL スキーマ名を指定しない場合、 CURRENT SCHEMA 特殊レジスターが表またはビューの暗黙的な修飾子として使用されます。*string-literal* は、大/小文字を区別します。*string-literal* には、データベース内の列名を識別する正確な文字を使用する必要があります。

戻り値

戻り値は、*string-literal* で指定された列内の非 NULL の XML 値を連結したシーケンスです。表またはビューに行がない場合、db2-fn:xmlcolumn は空のシーケンスを戻します。

db2-fn:xmlcolumn 関数によって戻されるシーケンス内の項目数が、指定された表またはビュー内の行数と異なる場合があります。これは、これらの行の中に NULL 値や、複数の項目からなるシーケンスが含まれる場合があるためです。

db2-fn:xmlcolumn 関数は db2-fn:sqlquery 関数に関連付けられ、両方の関数が同じ結果を生成する場合があります。ただし、2 つの関数の引数は、大/小文字の区別の点で異なります。 db2-fn:xmlcolumn 関数内の引数は、XQuery によって処理されるため、大/小文字を区別します。 DB2 データベースにおける表名および列名はデフォルトで大文字であるため、db2-fn:xmlcolumn の引数は通常大文字です。
db2-fn:sqlquery 関数の引数は SQL によって処理されるため、ID が自動的に大文字に変換されます。

以下の関数呼び出しは等価であり、同じ結果を戻します。

```
db2-fn:xmlcolumn('SQLSCHEMA.TABLENAME.COLNAME')
db2-fn:sqlquery('select colname from sqlschema.tablename')
```

例

文書のシーケンスを戻す例: 以下の関数は、この例の場合 SQL スキーマ SAMPLE に含まれる PRODUCT という名前の表内の XML 列 DESCRIPTION に保管される XML 文書のシーケンスを戻します。

```
db2-fn:xmlcolumn('SAMPLE.PRODUCT.DESCRIPTION')
```

暗黙的な SQL スキーマを使用する例: 以下の例では、DB2 データベースの CURRENT SCHEMA 特殊レジスターが SAMPLE に設定され、このため関数は前の例と同じ結果を戻します。

```
db2-fn:xmlcolumn('PRODUCT.DESCRIPTION')
```

SQL 区切り ID を使用する例: 以下の関数は、表 ″Student″ の列 ″Thesis″ に保管される文書のシーケンスを戻します (この表が現在 CURRENT SCHEMA に割り当てられているスキーマ内にあると想定します)。表名および列名に小文字が含まれるため、 db2-fn:xmlcolumn 関数のストリング・リテラル引数でこれらを指定するには以下の 2 つの方法があります。

- SQL 区切り ID として (二重引用符で囲んで) 指定:

```
db2-fn:xmlcolumn('"Student"."Thesis"')
```

- SQL 区切り ID であることを指示せずにストリングとして指定:

```
db2-fn:xmlcolumn('Student.Thesis')
```

これに対して、同じ表および列の情報を db2-fn:sqlquery 関数で使用する場合は、以下のように SQL 区切り ID を使用する必要があります。

```
db2-fn:sqlquery('select "Thesis" from "Student"')
```

year-from-date 関数

fn:year-from-date 関数は、xs:date 値の年コンポーネントを戻します。

構文

date-value
年コンポーネントが取り出される日付値。

date-value は xs:date タイプであるか、空のシーケンスです。

戻り値

date-value は xs:date タイプであり、戻り値は xs:integer タイプです。値は *date-value* の年コンポーネントであり、負の値にはなりません。

date-value が空のシーケンスである場合、戻り値は空のシーケンスです。

例

以下の関数は、2005 年 10 月 29 日の日付値の年コンポーネントを戻します。

```
fn:year-from-date(xs:date("2005-10-29"))
```

戻り値は 2005 です。

year-from-dateTime 関数

fn:year-from-dateTime 関数は、xs:dateTime 値の年コンポーネントを戻します。

構文

dateTime-value
年コンポーネントが取り出される dateTime 値。

dateTime-value は xs:dateTime タイプであるか、空のシーケンスです。

戻り値

dateTime-value が xs:dateTime タイプである場合、戻り値は xs:integer タイプです。値は *dateTime-value* の年コンポーネントであり、負の値にはなりません。

dateTime-value が空のシーケンスである場合、戻り値は空のシーケンスです。

例

以下の関数は、UTC-8 タイム・ゾーンの 2005 年 10 月 29 日、午前 8 時の dateTime 値の年コンポーネントを戻します。

```
fn:year-from-dateTime(xs:dateTime("2005-10-29T08:00:00-08:00"))
```

戻り値は 2005 です。

years-from-duration 関数

fn:years-from-duration 関数は、期間の年コンポーネントを戻します。

構文

duration-value
年コンポーネントが取り出される期間値。

duration-value は空のシーケンスであるか、タイプが xdt:dayTimeDuration、xs:duration、または xdt:yearMonthDuration のいずれかである値です。

戻り値

戻り値は、*duration-value* のタイプによって異なります。

- *duration-value* のタイプが xdt:yearMonthDuration または xs:duration である場合、戻り値は xs:integer タイプです。値は、xdt:yearMonthDuration としてキャストされる *duration-value* の年コンポーネントです。 *duration-value* が負の数である場合、この値も負の数になります。
- *duration-value* が xs:dayTimeDuration タイプである場合、戻り値は xs:integer タイプであり、0 です。
- *duration-value* が空のシーケンスである場合、戻り値は空のシーケンスです。

xdt:yearMonthDuration としてキャストされる *duration-value* の年コンポーネントは、xdt:yearMonthDuration としてキャストされる *duration-value* の総月数を 12 で除算することによって判別される年数 (整数) です。

例

以下の関数は、期間が -4 年 -11 カ月 -320 日の年コンポーネントを戻します。

```
fn:years-from-duration(xs:duration("-P4Y11M320D"))
```

戻り値は -4 です。

以下の関数は、9 年 13 カ月の期間の年コンポーネントを戻します。

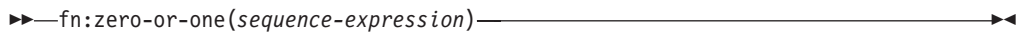
```
fn:years-from-duration(xdt:yearMonthDuration("P9Y13M"))
```

戻り値は 10 です。期間の総年数を計算する場合、13 カ月は 1 年と 1 カ月に変換されます。この期間は年コンポーネント 10 年を持つ、P10Y1M と等しいと言えます。

zero-or-one 関数

fn:zero-or-one 関数は、引数に 1 つの項目が含まれるか、引数が空のシーケンスである場合に、その引数を戻します。

構文



sequence-expression
任意のシーケンス (空のシーケンスを含む)。

戻り値

sequence-expression が 1 個の項目を含む場合、または空のシーケンスである場合、*sequence-expression* が戻されます。それ以外の場合は、エラーが戻されます。

例

以下の例では、fn:zero-or-one 関数を使用して、変数 $seq のシーケンスが 1 個以下の項目を含むかどうかを判別します。

```
let $seq := (5,10)
return fn:zero-or-one($seq)
```

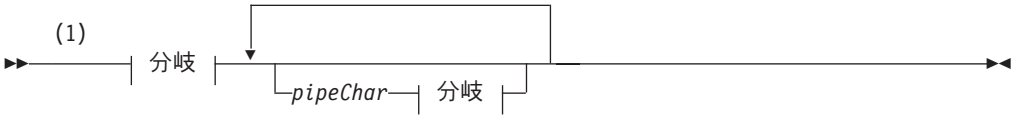
この場合、シーケンスに 2 個の項目が含まれているため、エラーが戻されます。

第 6 章 正規表現

正規表現とは、ストリングをマッチングおよび操作するためのパターンとしての動作する文字シーケンスです。正規表現は、以下の XQuery 関数で使用されます: fn:matches、fn:replace、および fn:tokenize。 DB2 XQuery 正規表現サポートは、W3C Recommendation *XML Schema Part 2: Datatypes Second Edition* で定義された XML スキーマの正規表現サポートと W3C Recommendation *XQuery 1.0 and XPath 2.0 Functions and Operators* により定義された拡張子を基にしています。

構文

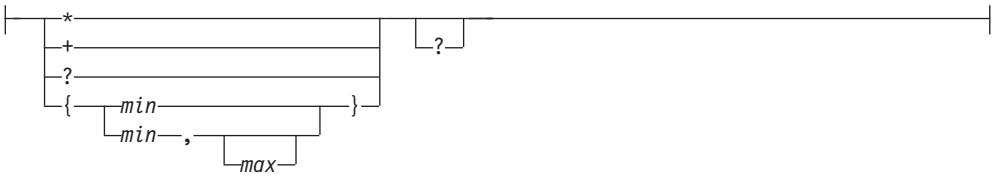
正規表現



分岐:

Atom:

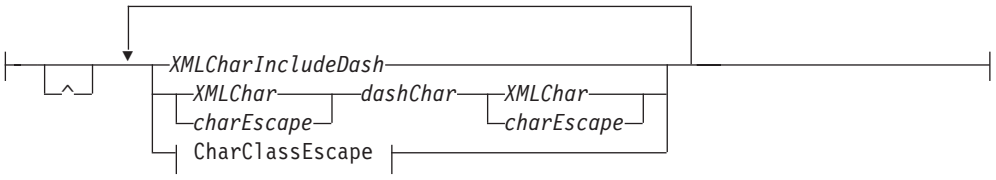
Quantifier:



CharClassExpression:

© IBM Corporation 2006, 2008

CharGroup:



CharClassEscape:

注:

1 正規表現の構文は、パターン文字としての特定の意味を持つ空白文字以外の空白文字を含まないストリング・リテラルの内容を示します。すべての形式の空白文字が許可されているため、構文エレメント間のスペース、または内容は考慮されません。

正規表現

正規表現には、1 つ以上の分岐が含まれます。分岐は、パイプ (|) で区切られており、それぞれの分岐が代替パターンであることを示します。

pipeChar
パイプ文字 (|) は、正規表現内の代替分岐を区切ります。

分岐
分岐は 0 個以上のアトムで構成されており、それぞれのアトムがオプションの数量詞を許可します。

Atom

アトムは、正規文字、文字クラス式、文字クラス・エスケープ、または括弧で囲まれた正規表現です。

normalCharacter
メタ文字の 1 つではない、有効な XML 文字はすべて 225 ページの表 37にあります。

^ 分岐の先頭で使用される場合、脱字記号 (^) は、ストリングの先頭から一致する必要のあるパターンを示します。

**$** 分岐の終わりで使用されている場合、ドル記号 ($) はストリングの終わりから一致する必要のあるパターンを示します。

Quantifier

数量詞は、正規表現内のアトムの繰り返しを示します。デフォルトでは、数量詞はターゲット・ストリングに可能なかぎり多くマッチングする、*欲張り (greedy) アルゴリズム*として知られるアルゴリズムを使用します。例えば、正規表現 `'A.*A'` は、'ABACADA' というストリング全体とマッチングします。これは、指定された外部文字 'A' に挟まれたサブストリングがすべての文字とすべての出現回数にマッチングするためです。デフォルトの欲張りアルゴリズムは、数量詞の後に疑問符 (?) 文字を指定することで変更できます。疑問符 (?) は、*控えめ (reluctant) アルゴリズム*を使用したパターン・マッチングを示します。これは、ターゲット・ストリング内の次に短い、正規表現の条件を満たすサブストリングにマッチング (左から右へ) していきます。例えば、正規表現 `'A.*?A'` は、'ABACADA' というストリング全体にマッチングせずに、サブストリング 'ABA' および 'ADA' にマッチングします。控えめアルゴリズムを使用して正規表現にマッチングしたサブストリングの文字は、以降のマッチングでは考慮されません。前例で 'ACA' がマッチングしたと判断されないのは、このためです。控えめアルゴリズムは、fn:replace 関数と使用すると一番役立ちます。これは、マッチングおよび置換が左から右へ処理されるためです。

例えば、ユーザーが欲張りアルゴリズムを関数 `fn:replace("nonsensical","n(.*)s","mus")` 内に使用した場合、″n″ で始まり、″s″ で終了する文字ストリングを、ストリング ″mus″ で置換するため、戻り値は 'musical' になります。サブストリング ″nons″ および ″ns″ を含む元のストリングで、さらに次のマッチングのパターンが左から右へのスキャンで一致している場合でも、欲張りアルゴリズムはこれらのマッチングを作動しません。これは、より長いエンクロージング・マッチングが検索されたためです。

関数 `fn:replace("nonsensical","n(.*?)s","mus")` 内の同一ストリングに、控えめアルゴリズムを使用した場合は、結果が異なります。戻り値は、″musemusical″ です。この場合、ストリング内で 2 つの置換が実行されます。最初のマッチングは ″nons″ を ″mus″ で置換し、2 番目のマッチングは ″ns″ を ″mus″ で置換します。

別の例としては、欲張りアルゴリズムを使用して何文字でも含むことができる文字 A を文字 X (関数 `fn:replace("AbrAcAdAbrA","A(.*)A","X$1X")`で同一の文字を含む) に置換すると、戻り値は ″XbrAcAdAbrX″ になります。サブストリング ″AbrA″ および ″AdA″ を含む元のストリングで、さらに次のマッチングのパターンが左から右へのスキャンで一致している場合でも、欲張りアルゴリズムはこれらのマッチングを作動しません。これは、より長いエンクロージング・マッチングが検索されたためです。

関数 `fn:replace("AbrAcAdAbrA","A(.*?)A","X$1X")` 内の同一ストリングに、控えめアルゴリズムを使用した場合は、結果が異なります。戻り値は ″XbrXcXdXbrA″ です。この場合、ストリング内で 2 つの置換が実行されます。最初の置換は ″AbrA″ に実行され、二番目の置換は ″AdA″ に実行されます。ストリング内の最後の ″A″ は、置換されません。これは、控えめアルゴリズムがストリング内のその他すべてのマッチングにおいて先行する ″A″ 文字を使用しているためです。元のストリング内の文字 ″A″ で始まり、終了するその他のサブストリング

(″AcA″、″AcAdA″、 ″AdAbrA″ および ″AbrA″ など) は、考慮されません。控えめアルゴリズムは、この文字がパターンのマッチングに加わった後で既に使用されていると見なすためです。

***** アトムに 0 回以上マッチングする。数量詞 {0,} に等しい。

+ アトムに 1 回以上マッチングする。数量詞 {1,} に等しい。

? アトムに 0 回、または 1 回マッチングする。数量詞 {0,1} に等しい。以下の別の数量詞では、欲張りアルゴリズムの代わりに控えめアルゴリズムを使用することが示されています。

min
少なくとも *min* 回、アトムにマッチングする。*min* は、正の整数である必要があります。

- {*min*} は、正確に *min* 回アトムにマッチングします。
- {*min*,} は、少なくとも *min* 回アトムにマッチングします。

max
最大でも *max* 回以下の回数、アトムにマッチングします。*max* は、正の整数で *min* よりも大きいか等しい必要があります。

- {0,*max*} は、*min* 回以下の回数のアトムにマッチングします。
- {0,0} は空ストリングにのみマッチングします。

CharGroup

^ CharGroup の残りの部分で定義されている文字セットの補数を示します。

dashChar
ダッシュ文字 (-) は、文字範囲内で外部文字を定義する 2 つの文字を区切ります。フォーム s-e の文字範囲は、s よりも大きいか等しく、 e よりも小さいか等しい UCS2 コード・ポイントのセットです。以下に例を示します:

- s は、バックスラッシュ文字 (¥) ではありません。
- s が CharGroup 内で最初の文字の場合、脱字文字 (^) ではありません。
- e は、バックスラッシュ文字 (¥)、または左大括弧文字 ([) ではありません。
- e のコード・ポイントは、s のコード・ポイントよりも大きくなります。

XMLCharIncludeDash
有効な XML 文字セットからの単一文字は、バックスラッシュ (¥) および大括弧 ([]) を除きますが、ダッシュ (-) は含みます。ダッシュ文字は、CharGroup の先頭、または終わりでのみ有効です。 CharGroup の先頭の脱字文字 (^) は、グループの補数を示します。先頭以外のグループ内のどこでも、脱字文字は脱字文字にのみマッチングします。 *XMLCharIncludeDash* には、正規表現 [^¥#5B#5D] によってマッチングされたすべての文字を含めることができます。

XMLChar
有効な XML 文字セットからの単一文字は、バックスラッシュ (¥)、ブラケット ([])、およびダッシュ (-) を除きます。ダッシュ文字は、CharGroup の先頭、または終わりでのみ有効です。 CharGroup の先頭の脱字文字 (^) は、グループの補数を示します。先頭以外のグループ内のどこでも、脱字文字は脱字文字にのみマッチングします。 *XMLChar* には、正規表現 [^¥#2D#5B#5D] によってマッチングされたすべての文字を含めることができます。

charEscape

単一メタキャラクター、改行文字、リターン文字、またはタブ文字が後に続くバックスラッシュです。ユーザーは、これらにマッチングさせるため、正規表現の表 37内の文字を拡張する必要があります。

表 37. 有効なメタキャラクター・エスケープ

文字エスケープ	表される文字	説明
¥n	#x0A	改行
¥r	#x0D	リターン
¥t	#x09	タブ
¥¥	¥	バックスラッシュ
¥|	|	パイプ
¥.	.	ピリオド
¥-	-	ダッシュ
¥^	^	脱字
¥?	?	疑問符 (?)
¥$	$	ドル記号
¥*	*	アスタリスク
¥+	+	正符号
¥{	{	左中括弧
¥}	}	右中括弧
¥(	(	左括弧
¥)	)	右括弧
¥[	[	左大括弧
¥]	]	右大括弧

CharClassEscape

. ピリオド文字 (.) は、改行、および戻り文字を除くすべての文字に一致します。ピリオド文字は、式 [^¥n¥r] に相当します。

¥*nonZeroDigit*

副次式によってマッチングされたストリングに一致する戻り参照 (これは括弧で囲まれています) を正規表現の *nonZeroDigit* の位置に指定します。*nonZeroDigit* は、1 から 9 の間である必要があります。最初の 9 個の副次式は参照できます。

注: 将来の上位互換性では、戻り参照の後に 1 桁の文字が続く場合、戻り参照を括弧で囲みます。例えば、桁数 3 が続く最初の副次式に対する戻り参照は、現在両方の結果が同じでも、/13 ではなく、(/1)3 と表される必要があります。

¥P{**Is***blockName*}

ユニコード・コード・ポイントの範囲の補数を指定します。*XML Schema Part 2: Datatypes Second Edition* にリストされているように、範囲は *blockName* によって指定されます。

¥p{Is*blockName***}**
ユニコード・コード・ポイントの特定の範囲内に、文字を指定します。*XML Schema Part 2: Datatypes Second Edition* にリストされているように、範囲は *blockName* によって指定されます。

charEscape
単一メタキャラクター、改行文字、リターン文字、またはタブ文字が後に続くバックスラッシュです。ユーザーは、これらにマッチングさせるため、正規表現の 225 ページの表 37内の文字を拡張する必要があります。

multiCharEscape
バックスラッシュの後に単一文字が続きます。その文字は、表 38 に示すように、一般的に使用される文字セットを正規表現内で表すために使用されます。

表 38. 複数文字エスケープ

複数文字エスケープ	同一の正規表現	説明
¥s	[#x20¥t¥n¥r]	スペース、タブ、改行、または戻り文字。
¥S	[^¥s]	スペース、タブ、改行、または戻り文字を除くすべての文字。
¥i	なし	XML 名内で最初の文字として許可されている文字セット。
¥I	[^¥i]	XML 名内で最初の文字として許可されている文字セットに含まれない。
¥c	なし	XML 名内で許可されている文字セット。
¥C	[^¥c]	XML 名内で許可されている文字セットに含まれない。
¥d	¥p{Nd}	小数桁数。
¥D	[^¥d]	小数桁数ではない。
¥w	[#x0000-#x10FFFF]-[¥p{P}¥p{Z}¥p{C}]	以下の *charProperty* カテゴリーを含むワード文字: 文字、マーク、シンボル、および数値。
¥W	[^¥w]	以下の *charProperty* カテゴリーを含む非ワード文字: 句読点、区切り文字、およびその他。

¥p{*charProperty***}**
カテゴリー内の文字を指定する。カテゴリーは、 227 ページの表 39にリストされています。

¥P{*charProperty***}**
文字カテゴリーの補数を指定します。カテゴリーは、 227 ページの表 39にリストされています。

表 39. サポートされている charProperty 値

カテゴリー	*charProperty*	**説明**
文字	L	すべての文字
	Lu	大文字
	Ll	小文字
	Lt	タイトル文字
	Lm	修飾子
	Lo	その他
マーク	M	すべてのマーク
	Mn	スペーシングなし
	Mc	スペーシング結合
	Me	括弧で囲まれた
数値	N	すべての数値
	Nd	小数桁数
	Nl	文字
	なし	その他
句読点	P	すべての句読点
	Pc	コネクター
	Pd	ダッシュ
	Ps	オープン
	Pe	クローズ
	Pi	初期の引用符 (使用法によって Ps または Pe のように動作する)
	Pf	最終の引用符 (使用法によって Ps または Pe のように動作する)
	Po	その他
区切り文字	Z	すべての区切り文字
	Zs	スペース
	Zl	行
	Zp	パラグラフ
シンボル	S	すべてのシンボル
	Sm	算術
	Sc	通貨
	Sk	修飾子
	So	その他
その他	C	すべてのその他
	Cc	制御
	Cf	フォーマット
	Co	専用
	Cn	割り当てない

注: 正規表現はバイナリー比較を使用して突き合わされます。デフォルトの照合は使用されません。

第 7 章 制限

DB2 XQuery にはサイズおよびデータ・タイプに制限があります。

XQuery タイプの制限

このトピックでは、特定の DB2 XQuery タイプに使用可能な値の範囲を示します。

表 40. XQuery 数値タイプの制限

データ・タイプ	説明	制限値
xs:float	最小値	-3.4028234663852886e+38
	最大値	+3.4028234663852886e+38
	正の最小値	+1.1754943508222875e-38
	負の最大値	-1.1754943508222875e-38
xs:double	最小値	-1.7976931348623158e+308
	最大値	+1.7976931348623158e+308
	正の最小値	+2.2250738585072014e-308
	負の最大値	+2.2250738585072014e-308
xs:decimal	10 進数の精度の最大値	31 桁
xs:integer	最小値	-9 223 372 036 854 775 808
	最大値	+9 223 372 036 854 775 807
xs:nonPositiveInteger	最小値	-9 223 372 036 854 775 808
	最大値	0
xs:negativeInteger	最小値	-9 223 372 036 854 775 808
	最大値	-1
xs:long	最小値	-9 223 372 036 854 775 808
	最大値	9 223 372 036 854 775 807
xs:int	最小値	-2 147 483 648
	最大値	+2 147 483 647
xs:short	最小値	-32 768
	最大値	+32 767
xs:byte	最小値	-128
	最大値	+127
xs:nonNegativeInteger	最小値	0
	最大値	+9 223 372 036 854 775 807
xs:unsignedLong	最小値	0
	最大値	+9 223 372 036 854 775 807
xs:unsignedInt	最小値	0
	最大値	4 294 967 295
xs:unsignedShort	最小値	0
	最大値	+65 535

© Copyright IBM Corp. 2006, 2008

表 40. XQuery 数値タイプの制限 (続き)

データ・タイプ	説明	制限値
xs:unsignedByte	最小値	0
	最大値	+255
xs:positiveInteger	最小値	+1
	最大値	+9 223 372 036 854 775 807

表 41. XQuery 日付タイプ、時間タイプ、および期間タイプの制限

データ・タイプ	説明	制限値
xs:duration	最小値	-P83333333333333Y3M11574074074DT1H46M39.999999S
	最大値	P83333333333333Y3M11574074074DT1H46M39.999999S
xdt:yearMonthDuration	最小値	-P83333333333333Y3M
	最大値	P83333333333333Y3M
xdt:dayTimeDuration	最小値	-P11574074074DT1H46M39.999999S
	最大値	P11574074074DT1H46M39.999999S
xs:dateTime[1、 2]	最小値	0001-01-01T00:00:00.000000Z
	最大値	9999-12-31T23:59:59.999999Z
xs:date[1]	最小値	0001-01-01Z
	最大値	9999-12-31Z
xs:time[2]	最小値	00:00:00Z
	最大値	23:59:59Z
xs:gDay[1]	最小値	01Z
	最大値	31Z
xs:gMonth[1]	最小値	01Z
	最大値	12Z
xs:gYear[1]	最小値	0001Z
	最大値	9999Z
xs:gYearMonth[1]	最小値	0001-01Z
	最大値	9999-12Z
xs:gMonthDay[1]	最小値	01-01Z
	最大値	12-31Z

注: DB2 XQuery では、負の日付をサポートしません。

サイズ制限

DB2 XQuery には、ストリング・リテラルおよび照会のサイズ制限があります。

ストリング・リテラルのサイズ制限は、32672 バイトです。

照会の長さにおけるサイズ制限は、2097152 バイトです。

付録 A. DB2 技術情報の概説

DB2 技術情報は、以下のツールと方法を介して利用できます。

- DB2 インフォメーション・センター
  - トピック (タスク、概念、およびリファレンス・トピック)
  - DB2 ツールのヘルプ
  - サンプル・プログラム
  - チュートリアル
- DB2 資料
  - PDF ファイル (ダウンロード可能)
  - PDF ファイル (DB2 PDF DVD に含まれる)
  - 印刷資料
- コマンド行ヘルプ
  - コマンド・ヘルプ
  - メッセージ・ヘルプ

注: DB2 インフォメーション・センターのトピックは、PDF やハードコピー資料よりも頻繁に更新されます。最新の情報を入手するには、資料の更新が発行されたときにそれをインストールするか、ibm.com® にある DB2 インフォメーション・センターを参照してください。

技術資料、ホワイト・ペーパー、IBM® Redbooks® 資料などのその他の DB2 技術情報には、オンライン (ibm.com) でアクセスできます。 DB2 Information Management ソフトウェア・ライブラリー・サイト (http://www.ibm.com/software/data/sw-library/) にアクセスしてください。

資料についてのフィードバック

DB2 の資料についてのお客様からの貴重なご意見をお待ちしています。DB2 の資料を改善するための提案については、db2docs@ca.ibm.com まで E メールを送信してください。DB2 の資料チームは、お客様からのフィードバックすべてに目を通しますが、直接お客様に返答することはありません。お客様が関心をお持ちの内容について、可能な限り具体的な例を提供してください。特定のトピックまたはヘルプ・ファイルについてのフィードバックを提供する場合は、そのトピック・タイトルおよび URL を含めてください。

DB2 お客様サポートに連絡する場合には、この E メール・アドレスを使用しないでください。資料を参照しても、DB2 の技術的な問題が解決しない場合は、お近くの IBM サービス・センターにお問い合わせください。

© IBM Corporation 1993, 2008

DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式)

以下の表は、DB2 ライブラリーについて説明しています。DB2 ライブラリーに関する詳細な説明については、www.ibm.com/shop/publications/order にある IBM Publications Center にアクセスしてください。英語の DB2 バージョン 9.5 のマニュアル (PDF 形式) とその翻訳版は、 www.ibm.com/support/docview.wss?rs=71&uid=swg2700947 からダウンロードできます。

この表には印刷資料が入手可能かどうかが示されていますが、国または地域によっては入手できない場合があります。

資料番号は、資料が更新される度に大きくなります。資料を参照する際は、以下にリストされている最新版であることを確認してください。

注: DB2 インフォメーション・センターは、PDF やハードコピー資料よりも頻繁に更新されます。

表 42. DB2 の技術情報

資料名	資料番号	印刷資料が入手可能かどうか
管理 API リファレンス	SC88-4431-01	入手可能
管理ルーチンおよびビュー	SC88-4435-01	入手不可
コール・レベル・インターフェース ガイドおよびリファレンス 第 1 巻	SC88-4433-01	入手可能
コール・レベル・インターフェース ガイドおよびリファレンス 第 2 巻	SC88-4434-01	入手可能
コマンド・リファレンス	SC88-4432-01	入手可能
データ移動ユーティリティー ガイドおよびリファレンス	SC88-4421-01	入手可能
データ・リカバリーと高可用性 ガイドおよびリファレンス	SC88-4423-01	入手可能
データ・サーバー、データベース、およびデータベース・オブジェクトのガイド	SC88-4259-01	入手可能
データベース・セキュリティー・ガイド	SC88-4418-01	入手可能
ADO.NET および OLE DB アプリケーションの開発	SC88-4425-01	入手可能
組み込み SQL アプリケーションの開発	SC88-4426-01	入手可能
Java アプリケーションの開発	SC88-4427-01	入手可能
Perl および PHP アプリケーションの開発	SC88-4428-01	入手不可
SQL および外部ルーチンの開発	SC88-4429-01	入手可能
データベース・アプリケーション開発の基礎	GC88-4430-01	入手可能

表 42. DB2 の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能かどうか
DB2 インストールおよび管理概説 (Linux および Windows 版)	GC88-4439-01	入手可能
国際化対応ガイド	SC88-4420-01	入手可能
メッセージ・リファレンス 第 1 巻	GI88-4109-00	入手不可
メッセージ・リファレンス 第 2 巻	GI88-4110-00	入手不可
マイグレーション・ガイド	GC88-4438-01	入手可能
Net Search Extender 管理およびユーザーズ・ガイド	SC88-4630-01	入手可能
パーティションおよびクラスタリングのガイド	SC88-4419-01	入手可能
Query Patroller 管理およびユーザーズ・ガイド	SC88-4611-00	入手可能
IBM データ・サーバー・クライアント機能 概説およびインストール	GC88-4441-01	入手不可
DB2 サーバー機能 概説およびインストール	GC88-4440-01	入手可能
Spatial Extender and Geodetic Data Management Feature ユーザーズ・ガイドおよびリファレンス	SC88-4629-01	入手可能
SQL リファレンス 第 1 巻	SC88-4436-01	入手可能
SQL リファレンス 第 2 巻	SC88-4437-01	入手可能
システム・モニター ガイドおよびリファレンス	SC88-4422-01	入手可能
問題判別ガイド	GI88-4108-01	入手不可
データベース・パフォーマンスのチューニング	SC88-4417-01	入手可能
Visual Explain チュートリアル	SC88-4449-00	入手不可
新機能	SC88-4445-01	入手可能
ワークロード・マネージャー ガイドおよびリファレンス	SC88-4446-01	入手可能
pureXML ガイド	SC88-4447-01	入手可能
XQuery リファレンス	SC88-4448-01	入手不可

表 43. DB2 Connect 固有の技術情報

資料名	資料番号	印刷資料が入手可能かどうか
DB2 Connect Personal Edition 概説およびインストール	GC88-4443-01	入手可能

表 43. DB2 Connect 固有の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能かどうか
DB2 Connect サーバー機能 概説およびインストール	GC88-4444-01	入手可能
DB2 Connect ユーザーズ・ガイド	SC88-4442-01	入手可能

表 44. Information Integration の技術情報

資料名	資料番号	印刷資料が入手可能かどうか
Information Integration: フェデレーテッド・システム 管理ガイド	SC88-4166-01	入手可能
Information Integration: レプリケーションおよびイベント・パブリッシングのための ASNCLP プログラム・リファレンス	SC88-4167-02	入手可能
Information Integration: フェデレーテッド・データ・ソース構成ガイド	SC88-4185-01	入手不可
Information Integration: SQL レプリケーション ガイドおよびリファレンス	SC88-4168-01	入手可能
Information Integration: レプリケーションとイベント・パブリッシング 概説	GC88-4187-01	入手可能

DB2 の印刷資料の注文方法

DB2 の印刷資料が必要な場合、オンラインで購入することができますが、すべての国および地域で購入できるわけではありません。 DB2 の印刷資料については、IBM 営業担当員にお問い合わせください。 *DB2 PDF ドキュメンテーション* DVD の一部のソフトコピー・ブックは、印刷資料では入手できないことに留意してください。例えば、「*DB2 メッセージ・リファレンス*」はどちらの巻も印刷資料としては入手できません。

DB2 PDF ドキュメンテーション DVD で利用できる DB2 の印刷資料の大半は、IBM に有償で注文することができます。国または地域によっては、資料を IBM Publications Center からオンラインで注文することもできます。お客様の国または地域でオンライン注文が利用できない場合、DB2 の印刷資料については、IBM 営業担当員にお問い合わせください。 DB2 PDF ドキュメンテーション DVD に収録されている資料の中には、印刷資料として提供されていないものもあります。

注: 最新で完全な DB2 資料は、DB2 インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2luw/v9r5) で参照することができます。

DB2 の印刷資料は以下の方法で注文することができます。

- 日本 IBM 発行のマニュアルはインターネット経由でご購入いただけます。詳しくは http://www.ibm.com/shop/publications/order の「ご注文について」をご覧ください。資料の注文情報にアクセスするには、お客様の国、地域、または言語を選択してください。その後、各ロケーションにおける注文についての指示に従ってください。
- DB2 の印刷資料を IBM 営業担当員に注文するには、以下のようにします。
  1. 以下の Web サイトのいずれかから、営業担当員の連絡先情報を見つけてください。
     - IBM Directory of world wide contacts (www.ibm.com/planetwide)
     - IBM Publications Web サイト (http://www.ibm.com/shop/publications/order) 国、地域、または言語を選択し、お客様の所在地に該当する Publications ホーム・ページにアクセスしてください。このページから、「このサイトについて」のリンクにアクセスしてください。
  2. 電話をご利用の場合は、DB2 資料の注文であることをご指定ください。
  3. 担当者に、注文する資料のタイトルと資料番号をお伝えください。タイトルと資料番号は、232 ページの『DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式)』でご確認いただけます。

コマンド行プロセッサーから SQL 状態ヘルプを表示する

DB2 は、SQL ステートメントの結果の原因になったと考えられる条件の SQLSTATE 値を戻します。 SQLSTATE ヘルプは、SQL 状態および SQL 状態クラス・コードの意味を説明します。

SQL 状態ヘルプを呼び出すには、コマンド行プロセッサーを開いて以下のように入力します。

```
? sqlstate or ? class code
```

ここで、*sqlstate* は有効な 5 桁の SQL 状態を、*class code* は SQL 状態の最初の 2 桁を表します。

例えば、`? 08003` を指定すると SQL 状態 08003 のヘルプが表示され、 `? 08` を指定するとクラス・コード 08 のヘルプが表示されます。

異なるバージョンの DB2 インフォメーション・センターへのアクセス

DB2 バージョン 9.5 のトピックを扱っている DB2 インフォメーション・センターの URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/です。

DB2 バージョン 9 のトピックを扱っている DB2 インフォメーション・センターの URL は http://publib.boulder.ibm.com/infocenter/db2luw/v9/です。

DB2 バージョン 8 のトピックについては、バージョン 8 のインフォメーション・センターの URL http://publib.boulder.ibm.com/infocenter/db2luw/v8/にアクセスしてください。

DB2 インフォメーション・センターでの希望する言語でのトピックの表示

DB2 インフォメーション・センターでは、ブラウザーの設定で指定した言語でのトピックの表示が試みられます。トピックがその指定言語に翻訳されていない場合は、 DB2 インフォメーション・センターでは英語でトピックが表示されます。

- Internet Explorer Web ブラウザーで、指定どおりの言語でトピックを表示するには、以下のようにします。
  1. Internet Explorer の「**ツール**」 -> 「**インターネット オプション**」 -> 「**言語...**」ボタンをクリックします。「言語の優先順位」ウィンドウがオープンします。
  2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
     - リストに新しい言語を追加するには、「**追加...**」ボタンをクリックします。

       注: 言語を追加しても、特定の言語でトピックを表示するのに必要なフォントがコンピューターに備えられているとはかぎりません。
     - リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで「**上に移動**」ボタンをクリックします。
  3. ブラウザー・キャッシュを消去してから、ページを最新表示します。希望する言語で DB2 インフォメーション・センターが表示されます。
- Firefox または Mozilla Web ブラウザーの場合に、希望する言語でトピックを表示するには、以下のようにします。
  1. 「**ツール**」 -> 「**オプション**」 -> 「**詳細**」 ダイアログの「**言語**」セクションにあるボタンを選択します。「設定」ウィンドウに「言語」パネルが表示されます。
  2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
     - リストに新しい言語を追加するには、「**追加...**」ボタンをクリックしてから、「言語を追加」ウィンドウで言語を選択します。
     - リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで「**上に移動**」ボタンをクリックします。
  3. ブラウザー・キャッシュを消去してから、ページを最新表示します。希望する言語で DB2 インフォメーション・センターが表示されます。

ブラウザーとオペレーティング・システムの組み合わせによっては、オペレーティング・システムの地域の設定も希望のロケールと言語に変更しなければならない場合があります。

コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの更新

DB2 インフォメーション・センターをローカルにインストールしている場合は、IBM から資料の更新を入手してインストールすることができます。

ローカルにインストールされた DB2 インフォメーション・センターを更新するには、以下のことを行う必要があります。

1. コンピューター上の DB2 インフォメーション・センターを停止し、インフォメーション・センターをスタンドアロン・モードで再始動します。インフォメーション・センターをスタンドアロン・モードで実行すると、ネットワーク上の他のユーザーがそのインフォメーション・センターにアクセスできなくなります。これで、更新を適用できるようになります。非管理者および非 root の DB2 インフォメーション・センターは常にスタンドアロン・モードで実行されます。を参照してください。
2. 「更新」機能を使用することにより、どんな更新が利用できるかを確認します。インストールする更新がある場合は、「更新」機能を使用してそれを入手およびインストールできます。

   注: ご使用の環境において、インターネットに接続されていないマシンに DB2 インフォメーション・センターの更新をインストールする必要がある場合は、インターネットに接続されていて DB2 インフォメーション・センターがインストールされているマシンを使用して、更新サイトをローカル・ファイル・システムにミラーリングする必要があります。ネットワーク上の多数のユーザーが資料の更新をインストールする場合にも、更新サイトをローカルにミラーリングして、更新サイト用のプロキシーを作成することにより、個々のユーザーが更新を実行するのに要する時間を短縮できます。
   更新パッケージが入手可能な場合、「更新」機能を使用してパッケージを入手します。ただし、「更新」機能は、スタンドアロン・モードでのみ使用できます。
3. スタンドアロンのインフォメーション・センターを停止し、コンピューター上の DB2 インフォメーション・センターを再開します。

注: Windows® Vista の場合、下記のコマンドは管理者として実行する必要があります。完全な管理者特権でコマンド・プロンプトまたはグラフィカル・ツールを起動するには、ショートカットを右クリックしてから、 **「管理者として実行」**を選択します。

コンピューターまたはイントラネット・サーバーにインストール済みの DB2 インフォメーション・センターを更新するには、以下のようにします。

1. DB2 インフォメーション・センターを停止します。
   - Windows では、**「スタート」** → **「コントロール パネル」** → **「管理ツール」** → **「サービス」**をクリックします。次に、**「DB2 インフォメーション・センター」**サービスを右クリックして**「停止」**を選択します。
   - Linux® では、以下のコマンドを入力します。

     ```
     /etc/init.d/db2icdv95 stop
     ```
2. インフォメーション・センターをスタンドアロン・モードで開始します。
   - Windows の場合:
     a. コマンド・ウィンドウを開きます。
     b. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは <Program Files>¥IBM¥DB2 Information Center¥Version 9.5 ディレクトリーにインストールされています (<Program Files> は「Program Files」ディレクトリーのロケーション)。

c. インストール・ディレクトリーから doc¥bin ディレクトリーにナビゲートします。

d. 次のように help_start.bat ファイルを実行します。

```
help_start.bat
```

- Linux の場合:

a. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは /opt/ibm/db2ic/V9.5 ディレクトリーにインストールされています。

b. インストール・ディレクトリーから doc/bin ディレクトリーにナビゲートします。

c. 次のように help_start スクリプトを実行します。

```
help_start
```

システムのデフォルト Web ブラウザーが起動し、スタンドアロンのインフォメーション・センターが表示されます。

3. 「**更新**」ボタン () をクリックします。インフォメーション・センターの右側のパネルで、「**更新の検索 (Find Updates)**」をクリックします。 既存の文書に対する更新のリストが表示されます。
4. インストール・プロセスを開始するには、インストールする更新をチェックして選択し、「**更新のインストール**」をクリックします。
5. インストール・プロセスが完了したら、「**完了**」をクリックします。
6. 次のようにして、スタンドアロンのインフォメーション・センターを停止します。
   - Windows の場合は、インストール・ディレクトリーの doc¥bin ディレクトリーにナビゲートしてから、次のように help_end.bat ファイルを実行します。

     ```
     help_end.bat
     ```

     注: help_end バッチ・ファイルには、help_start バッチ・ファイルを使用して開始したプロセスを安全に終了するのに必要なコマンドが含まれています。help_start.bat は、Ctrl-C や他の方法を使用して終了しないでください。
   - Linux の場合は、インストール・ディレクトリーの doc/bin ディレクトリーにナビゲートしてから、次のように help_end スクリプトを実行します。

     ```
     help_end
     ```

     注: help_end スクリプトには、help_start スクリプトを使用して開始したプロセスを安全に終了するのに必要なコマンドが含まれています。他の方法を使用して、help_start スクリプトを終了しないでください。
7. DB2 インフォメーション・センターを再開します。
   - Windows では、「**スタート**」 → 「**コントロール パネル**」 → 「**管理ツール**」 → 「**サービス**」をクリックします。次に、「**DB2 インフォメーション・センター**」サービスを右クリックして「**開始**」を選択します。
   - Linux では、以下のコマンドを入力します。

     ```
     /etc/init.d/db2icdv95 start
     ```

更新された DB2 インフォメーション・センターに、更新された新しいトピックが表示されます。

DB2 チュートリアル

DB2 チュートリアルは、DB2 製品のさまざまな機能について学習するのを支援します。この演習をとおして段階的に学習することができます。

はじめに

インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2help/) から、このチュートリアルの XHTML 版を表示できます。

演習の中で、サンプル・データまたはサンプル・コードを使用する場合があります。個々のタスクの前提条件については、チュートリアルを参照してください。

DB2 チュートリアル

チュートリアルを表示するには、タイトルをクリックします。

「*pureXML ガイド* 」の『**pureXML™**』
: XML データを保管し、ネイティブ XML データ・ストアに対して基本的な操作を実行できるように、DB2 データベースをセットアップします。

「*Visual Explain チュートリアル*」の『**Visual Explain**』
: Visual Explain を使用して、パフォーマンスを向上させるために SQL ステートメントを分析し、最適化し、調整します。

DB2 トラブルシューティング情報

DB2 製品を使用する際に役立つ、トラブルシューティングおよび問題判別に関する広範囲な情報を利用できます。

DB2 ドキュメンテーション
: トラブルシューティング情報は、DB2 問題判別ガイド、または DB2 インフォメーション・センターの「サポートおよびトラブルシューティング」セクションにあります。ここには、DB2 診断ツールおよびユーティリティーを使用して、問題を切り分けて識別する方法、最も頻繁に起こる幾つかの問題に対するソリューションについての情報、および DB2 製品を使用する際に発生する可能性のある問題の解決方法についての他のアドバイスがあります。

DB2 Technical Support の Web サイト
: 現在問題が発生していて、考えられる原因とソリューションを検索したい場合は、DB2 Technical Support の Web サイトを参照してください。
Technical Support サイトには、最新の DB2 資料、TechNotes、プログラム診断依頼書 (APAR またはバグ修正)、フィックスパック、およびその他のリソースへのリンクが用意されています。この知識ベースを活用して、問題に対する有効なソリューションを探し出すことができます。

  DB2 Technical Support の Web サイト (http://www.ibm.com/software/data/db2/udb/support.html) にアクセスしてください。

ご利用条件

これらの資料は、以下の条件に同意していただける場合に限りご使用いただけます。

個人使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。ただし、IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することはできません。

商業的使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することができます。 ただし、IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入関連法規を含む、すべての関連法規を遵守するものとします。

IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態で提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任なしで提供されます。

付録 B. 特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-8711
東京都港区六本木 3-2-12
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書は、IBM 以外の Web サイトおよびリソースへのリンクまたは参照を含む場合があります。 IBM は、本書より参照もしくはアクセスできる、または本書からリンクされた IBM 以外の Web サイトもしくは第三者のリソースに対して一切の責任を負いません。 IBM 以外の Web サイトにリンクが張られていることにより IBM が当該 Web サイトを推奨するものではなく、またその内容、使用もしくはサイトの所有者について IBM が責任を負うことを意味するものではありません。 また、IBM は、お客様が IBM Web サイトから第三者の存在を知ることになった場合にも (もしくは、IBM Web サイトから第三者へのリンクを使用した場合にも)、お客様と第三者との間のいかなる取引に対しても一切責任を負いません。 従って、お客様は、IBM が上記の外部サイトまたはリソースの利用について責任を負うものではなく、また、外部サイトまたはリソースからアクセス可能なコンテンツ、サービス、

© IBM Corporation 1993, 2008

製品、またはその他の資料一切に対して IBM が責任を負うものではないことを承諾し、同意するものとします。 第三者により提供されるソフトウェアには、そのソフトウェアと共に提供される固有の使用条件が適用されます。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

© (*お客様の会社名*) (*西暦年*). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. _*年を入れる*_. All rights reserved.

商標

以下は、International Business Machines Corporation の米国およびその他の国における商標です。

pureXML
ibm.com
IBM
Redbooks
DB2

以下は、それぞれ各社の商標または登録商標です。

- Linux は、Linus Torvalds の米国およびその他の国における商標です。
- Windows は、Microsoft Corporation の米国およびその他の国における商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

値、原子 6
値のキャストの検査 (DB2 XQuery) 118
値のキャストのテスト (DB2 XQuery) 118
値比較 80
一般比較 82
エレメント
  計算コンストラクター 96
  直接コンストラクター 88
  範囲内のネーム・スペース 94
エレメント・ノード 9
演算子
  優先順位 55
エンティティー参照 62

[カ行]

階層、 ノード 11
階層、データ・タイプ 19
拡張 QName
  説明 13
  変換 192
括弧、操作の優先順位 55
括弧で囲まれた式
  コンストラクターの 87
括弧で囲んだ式 64
空のシーケンス、順序 52
空の順序宣言 52
関数
  DB2 XQuery
    カウント 155
    カテゴリー 137
    シーケンス関数のリスト 141
    時間関数のリスト 140
    数値関数のリスト 139
    ストリング 202
    ストリング関数のリスト 137
    その他の関数のリスト 143
    データ 157
    ノード関数のリスト 143
    日付、時間、および期間関数のリスト 140
    ブール 151
    ブール関数、リスト 139
    リスト 137
    abs 149
関数 *(続き)*
  DB2 XQuery *(続き)*
    adjust-dateTime-to-timezone 145
    adjust-date-to-timezone 143
    adjust-time-to-timezone 147
    avg 150
    ceiling 152
    codepoints-to-string 153
    compare 153
    concat 154
    contains 155
    current-date 156
    current-dateTime 156
    current-time 156
    dateTime 158
    days-from-duration 159
    day-from-date 158
    day-from-dateTime 159
    deep-equal 160
    default-collation 162
    distinct-values 162
    empty 163
    ends-with 164
    exactly-one 164
    exists 165
    false 165
    floor 166
    hours-from-dateTime 166
    hours-from-duration 167
    hours-from-time 168
    implicit-timezone 168
    index-of 169
    insert-before 170
    in-scope-prefixes 169
    last 171
    local-name 171
    local-name-from-QName 172
    lower-case 173
    matches 174
    max 175
    min 176
    minutes-from-dateTime 178
    minutes-from-duration 178
    minutes-from-time 179
    months-from-duration 180
    month-from-date 179
    month-from-dateTime 180
    name 181
    namespace-uri 182
    namespace-uri-for-prefix 183
    namespace-uri-from-QName 184

© Copyright IBM Corp. 2006, 2008

関数 *(続き)*
DB2 XQuery *(続き)*
node-name 184
normalize-space 185
normalize-unicode 186
not 187
number 187
one-or-more 188
position 188
QName 189
QName 関数のリスト 142
remove 189
replace 190
resolve-QName 192
reverse 193
root 193
round 194
round-half-to-even 195
seconds-from-dateTime 196
seconds-from-duration 197
seconds-from-time 198
sqlquery 199
starts-with 202
string-join 203
string-length 204
string-to-codepoints 204
subsequence 205
substring 205
substring-after 206
substring-before 207
sum 208
timezone-from-date 209
timezone-from-dateTime 210
timezone-from-time 210
tokenize 211
translate 213
true 214
unordered 214
upper-case 215
xmlcolumn 216
years-from-duration 218
year-from-date 217
year-from-dateTime 217
zero-or-one 219
関数呼び出し 64
期間タイプのリスト 23
基本式 61
キャスト、 データ・タイプ 27
キャスト可能な式 118
キャスト式 117
境界空白
宣言 48
直接エレメント・コンストラクター内の 93
境界スペース宣言 48
空白
境界 48
空白 *(続き)*
説明 15
直接エレメント・コンストラクター内の 93
組み込み データ・タイプ、のコンストラクター 26
組み込み関数 137
計算コンストラクター
エレメント 96
処理 命令 100
説明 86
属性
説明 97
comment 102
計算属性コンストラクター 97
結果
式の順序 56
結果の順序 56
原子 タイプ 19
原子値 6
原子化 58
更新
DB2 インフォメーション・センター 237
更新式 120
結合 120
更新式の結合 120
構成宣言 49
構文
省略 71
変換式 123
FLWOR 式 102
コメント
計算コンストラクター 102
構成、概要 101
照会言語、説明 16
直接コンストラクター 101
コメント・ノード 10
ご利用条件
資料の使用 240
コンストラクター
括弧で囲まれた式 87
組み込みタイプ 26
計算 エレメント 96
計算 コメント 102
計算 処理命令 100
計算 属性 97
処理 命令 100
属性
説明 97
直接 エレメント
説明 88
直接 コメント 101
直接 処理命令 100
テキスト・ ノード 99
ネーム・スペース宣言属性 92
範囲内のネーム・スペース 94
文書 ノード 98
XML 86

コンテキスト・アイテム式　64

［サ行］

削除式　126
サブタイプ 置換　59
算術式　77
シーケンス
　空　52
　原子化　58
　構成　74
　説明　5
　ノード
　　結合　76
　有効なブール値　60
シーケンス関数のリスト　141
シーケンス式　74
シーケンス内の 項目　5
時間関数のリスト　140
時間帯、暗黙的な　168
時間タイプのリスト　23
式
　基本
　　エンティティー参照　62
　　概要　61
　　括弧で囲んだ　64
　　関数呼び出し　64
　　コンテキスト・アイテム　64
　　変数 参照　63
　　文字参照　63
　　リテラル　61
　キャスト可能　118
　結果の順序　56
　原子化　58
　コンストラクター
　　計算 エレメント　96
　　計算 コメント　102
　　計算 処理命令　100
　　計算 属性　97
　　処理 命令　100
　　説明　86
　　直接 エレメント　88
　　直接 コメント　101
　　直接 処理命令　100
　　テキスト・ ノード　99
　　ネーム・スペース宣言属性　92
　　範囲内のネーム・スペース　94
　　文書 ノード　98
　コンストラクターの 括弧で囲まれた　87
　削除　126
　サブタイプ 置換　59
　算術　77
　シーケンス　74
　シーケンスの 構成　74
　述部　73
　条件　114
式 *(続き)*
　処理　55
　挿入　127
　タイプの プロモーション　60
　動的コンテキスト　55
　名前変更　130
　ノードの シーケンスの結合　76
　パス
　　構文　66
　　省略構文　71
　　説明　65
　範囲　75
　比較
　　値　80
　　一般　82
　　概要　80
　　ノード　84
　フィルター　75
　フォーカス　55
　変換
　　構文　123
　　copy 節　124
　　modify 節　124
　　return 節　124
　有効なブール値　60
　優先順位　55
　量化　115
　論理　85
　cast　117
　FLWOR
　　概要　102
　　構文　102
　　例　111
　　for 節　104
　　for 節 および let 節、変数の有効範囲　107
　　for 節および let 節、 概要　104
　　for 節および let 節、 比較　106
　　for 節および let 節の 両方　106
　　let 節　105
　　order by 節　108
　　return 節　111
　　where 節　107
　replace　133
　XML データ更新時のエラー　121
　XML データの更新　120
式の コンテキスト　55
式の評価　55
式のフォーカス　55
軸
　省略構文　71
　パス式の　68
軸ステップ
　ノード・テスト　69
　パス式の　67
修飾名 (QName)
　概要　13

修飾名 (QName) *(続き)*
拡張、変換 192
述部
式 73
種類テスト 70
順序付けモード宣言 52
仕様
XQuery 17
照会
構造 1
照会言語
コメント 16
大/小文字の区別 15
XML データ 2
条件式 114
省略構文 71
処理順序 108
処理の 順序 108
処理命令ノード
構成 100
説明 10
シリアライゼーション
XML データ 13
資料
印刷 232
注文 234
概要 231
使用に関するご利用条件 240
PDF 232
数値関数のリスト 139
数値述部 73
数値タイプのリスト 22
数値リテラル 61
ストリング ・タイプのリスト 21
ストリング関数のリスト 137
ストリング・リテラル 61
正規化された期間形式
dayTimeDuration タイプ 33
duration タイプ 35
yearMonthDuration タイプ 45
正規表現
構文 221
説明 221
制限
サイズ 230
XQuery タイプ 229
静的に既知のネーム・スペース 94
セッター、 プロローグ 47
宣言
空の 順序 52
境界スペース 48
構成 49
順序付けモード 52
デフォルトのエレメント/タイプの ネーム・スペース宣言 50
デフォルトの関数 ネーム・スペース 51
宣言 *(続き)*
ネーム・スペース 53
プロローグ 47
copy-namespaces 49
version 48
挿入式 127
属性
計算コンストラクター
説明 97
構成 97
ネーム・スペース 宣言 92
属性ノード 9

[タ行]

タイプ
データ・タイプを参照 21
タイプ階層 19
タイプの プロモーション 60
タイプ・ キャスト 27
大/小文字の区別、照会言語 15
置換式 133
チュートリアル
トラブルシューティング 239
問題判別 239
Visual Explain 239
直接コンストラクター
エレメント
説明 88
エレメント内の空白 93
処理 命令 100
説明 86
comment 101
データ・ タイプ階層 19
データ・ モデル
XQuery および XPath 5
データ・タイプ
値のキャストのテスト (DB2 XQuery) 118
概要 19
カテゴリー 21
キャスト 27
組み込み、のコンストラクター 26
数値のリスト 22
ストリング。のリスト 21
制限 229
その他のリスト 25
置換 59
汎用のリスト 21
非型付きのリスト 21
日付、時間、および期間のリスト 23
プロモーション 60
リスト 21
xdt: 45
xdt:anyAtomicType 29
xdt:dayTimeDuration 33
xdt:untyped 45

データ・タイプ *(続き)*
xdt:untypedAtomic 45
xs:anySimpleType 29
xs:anyType 30
xs:anyURI 30
xs:base64Binary 30
xs:boolean 30
xs:byte 30
xs:date 31
xs:dateTime 31
xs:decimal 34
xs:double 34
xs:duration 35
xs:ENTITY 36
xs:float 36
xs:gDay 37
xs:gMonth 37
xs:gMonthDay 38
xs:gYear 38
xs:gYearMonth 39
xs:hexBinary 39
xs:ID 39
xs:IDREF 39
xs:int 40
xs:integer 40
xs:language 40
xs:long 40
xs:Name 40
xs:NCName 41
xs:negativeInteger 41
xs:NMTOKEN 41
xs:nonNegativeInteger 41
xs:nonPositiveInteger 41
xs:normalizedString 42
xs:NOTATION 42
xs:positiveInteger 42
xs:QName 42
xs:short 43
xs:string 43
xs:time 43
xs:token 44
xs:unsignedByte 44
xs:unsignedInt 44
xs:unsignedLong 44
xs:unsignedShort 44
定位置述部 73
テキスト・ノード
構成 99
説明 10
デフォルトのエレメント/タイプの ネーム・スペース宣言 50
デフォルトの関数 ネーム・スペース宣言 51
動的コンテキスト、式 55
特記事項 241
トラブルシューティング
オンライン情報 239
チュートリアル 239

［ナ行］

名前 テスト 69
名前変更式 130
ネーム・スペース 13
関数、 デフォルト 51
接頭部の バインディング 92
宣言 53
デフォルトの エレメント/タイプ 50, 92
デフォルトの 設定 92
範囲内の 94
ネーム・スペース宣言 53
ネーム・スペース宣言属性 92
ノード
エレメント 9
階層 11
概要 6, 8
型付き値 11
シーケンスの 結合 76
処理 命令
構成 100
説明 10
ストリング 値 11
属性 9
重複 11
比較 84
プロパティー 8
文書
構成 98
説明 8
comment
計算コンストラクター 102
構成、概要 101
説明 10
直接コンストラクター 101
identity 11
text
構成 99
説明 10
ノード、削除 126
ノード、追加 127
ノードおよびノード値、置換 133
ノードおよびノード値の置換 133
ノードの ストリング値 11
ノードの ID 11
ノードの型付き値 11
ノードの削除 126
ノードの除去 126
ノードの挿入 127
ノードの追加 127
ノードの名前変更 130
ノード名、変更 130
ノード名の変更 130
ノード・テスト 69

［ハ行］

バージョン宣言　48
パス式
　構文　66
　軸ステップ　67
　省略構文および 非省略構文　71
　説明　65
範囲 式　75
範囲内のネーム・スペース　94
汎用 タイプのリスト　21
比較式
　値　80
　一般　82
　概要　80
　ノード　84
非型付きデータ・タイプのリスト　21
日付および時間関数のリスト　140
日付タイプのリスト　23
ブール関数　151
ブール関数、リスト　139
ブール・データ・タイプ　30
フィルター式　75
フォワード軸　68
プリミティブ・タイプのキャスト　27
プロローグ
　空の順序宣言　52
　境界スペース宣言　48
　構成宣言　49
　構文　47
　順序付けモード宣言　52
　デフォルトのエレメント/タイプの ネーム・スペース宣言　50
　デフォルトの関数 ネーム・スペース宣言　51
　ネーム・スペース宣言　53
　バージョン宣言　48
　copy-namespace 宣言　49
文書順序　11
文書ノード
　構成　98
　説明　8
ヘルプ
　言語の構成　236
　SQL ステートメント　235
変換式
　構文　123
　copy 節　124
　modify 節　124
　return 節　124
変数
　参照　63
　for 節および let 節における 有効範囲　107
　for 節の定位置　105

［マ行］

文字参照　63
問題判別
　チュートリアル　239
　利用できる情報　239

［ヤ行］

有効なブール値　60
優先順位
　演算子 と式　55

［ラ行］

リソース
　XQuery　17
リテラル　61
リバース軸　68
量化式　115
論理式　85

A

abs 関数　149
adjust-dateTime-to-timezone 関数　145
adjust-date-to-timezone 関数　143
adjust-time-to-timezone 関数　147
and 演算子　85
anyAtomicType タイプ　29
anySimpleType タイプ　29
anyType タイプ　30
anyURI タイプ　30
attribute 軸　68
avg 関数　150

B

base64Binary タイプ　30
byte タイプ　30

C

ceiling 関数　152
child 軸　68
codepoints-to-string 関数　153
compare 関数　153
concat 関数　154
contains 関数　155
copy 節　124
copy-namespace 宣言　49
count 関数　155
current-date 関数　156
current-dateTime 関数　156
current-time 関数　156

D

data 関数 157
date タイプ 31
dateTime 関数 158
dateTime タイプ 31
days-from-duration 関数 159
dayTimeDuration タイプ 33
  正規化された形式 33
day-from-date 関数 158
day-from-dateTime 関数 159
DB2 XQuery 関数
  カウント 155
  関数のリスト 140
  シーケンス関数のリスト 141
  数値関数のリスト 139
  ストリング 202
  ストリング関数のリスト 137
  その他の関数のリスト 143
  データ 157
  ノード関数のリスト 143
  日付、時間、および期間関数のリスト 140
  ブール 151
  ブール関数、リスト 139
  abs 149
  adjust-dateTime-to-timezone 145
  adjust-date-to-timezone 143
  adjust-time-to-timezone 147
  avg 150
  ceiling 152
  codepoints-to-string 153
  compare 153
  concat 154
  contains 155
  current-date 156
  current-dateTime 156
  current-time 156
  dateTime 158
  days-from-duration 159
  day-from-date 158
  day-from-dateTime 159
  deep-equal 160
  default-collation 162
  distinct-values 162
  empty 163
  ends-with 164
  exactly-one 164
  exists 165
  false 165
  floor 166
  hours-from-dateTime 166
  hours-from-duration 167
  hours-from-time 168
  implicit-timezone 関数 168
  index-of 169
  insert-before 170
DB2 XQuery 関数 *(続き)*
  in-scope-prefixes 169
  last 171
  local-name 171
  local-name-from-QName 172
  lower-case 173
  matches 174
  max 175
  min 176
  minutes-from-dateTime 178
  minutes-from-duration 178
  minutes-from-time 179
  months-from-duration 180
  month-from-date 179
  month-from-dateTime 180
  name 181
  namespace-uri 182
  namespace-uri-for-prefix 183
  namespace-uri-from-QName 184
  node-name 184
  normalize-space 185
  normalize-unicode 186
  not 187
  number 187
  one-or-more 188
  position 188
  QName 189
  QName 関数のリスト 142
  remove 189
  replace 190
  resolve-QName 192
  reverse 193
  root 193
  round 194
  round-half-to-even 195
  seconds-from-dateTime 196
  seconds-from-duration 197
  seconds-from-time 198
  sqlquery 3, 199
  starts-with 202
  string-join 203
  string-length 204
  string-to-codepoints 204
  subsequence 205
  substring 205
  substring-after 206
  substring-before 207
  sum 208
  timezone-from-date 209
  timezone-from-dateTime 210
  timezone-from-time 210
  tokenize 211
  translate 213
  true 214
  unordered 214
  upper-case 215

DB2 XQuery 関数 *(続き)*
xmlcolumn 3, 216
years-from-duration 218
year-from-date 217
year-from-dateTime 217
zero-or-one 219
DB2 XQuery、概要 1
DB2 XQuery、XML データの更新 120
DB2 インフォメーション・センター
言語 236
更新 237
バージョン 235
別の言語で表示する 236
DB2 資料の印刷方法 234
DB2 の定義済み関数 137
decimal タイプ 34
deep-equal 関数 160
default-collation 関数 162
descendant 軸 68
descendant-or-self 軸 68
distinct-values 関数 162
double タイプ 34
duration タイプ 35
正規化された形式 35

E

empty 関数 163
ends-with 関数 164
ENTITY タイプ 36
exactly-one 関数 164
exists 関数 165

F

false 関数 165
float タイプ 36
floor 関数 166
FLWOR 式
概要 102
構文 102
例 111
for 節 104
for 節および let 節
概要 104
同一式内の 106
比較 106
変数の有効範囲 107
let 節
説明 105
order by 節 108
return 節 111
where 節 107
for 節
説明 104

G

gDay タイプ 37
gMonth タイプ 37
gMonthDay タイプ 38
gYear タイプ 38
gYearMonth タイプ 39

H

hexBinary タイプ 39
hours-from-dateTime 関数 166
hours-from-duration 関数 167
hours-from-time 関数 168

I

ID タイプ 39
IDREF タイプ 39
if-then-else 式
説明 114
implicit-timezone 関数 168
index-of 関数 169
insert-before 関数 170
int タイプ 40
integer タイプ 40
in-scope-prefixes 関数 169

L

language タイプ 40
last 関数 171
let 節
説明 105
local-name 関数 171
local-name-from-QName 関数 172
long タイプ 40
lower-case 関数 173

M

matches 関数 174
max 関数 175
min 関数 176
minutes-from-dateTime 関数 178
minutes-from-duration 関数 178
minutes-from-time 関数 179
modify 節 124
months-from-duration 関数 180
month-from-date 関数 179
month-from-dateTime 関数 180

N

name 関数 181
Name タイプ 40
namespace-uri 関数 182
namespace-uri-for-prefix 関数 183
namespace-uri-from-QName 関数 184
NCName タイプ 41
negativeInteger タイプ 41
NMTOKEN タイプ 41
node-name 関数 184
nonNegativeInteger タイプ 41
nonPositiveInteger タイプ 41
normalizedString タイプ 42
normalize-space 関数 185
normalize-unicode 関数 186
not 関数 187
NOTATION タイプ 42
number 関数 187

O

one-or-more 関数 188
or 演算子 85
order by 節 108

P

parent 軸 68
position 関数 188
positiveInteger タイプ 42

Q

QName 関数 189
QName (修飾 名) 13
  概要 13
  拡張、変換 192
QName タイプ 42

R

remove 関数 189
replace 関数 190
resolve-QName 関数 192
return 節 111, 124
reverse 関数 193
root 関数 193
round 関数 194
round-half-to-even 関数 195

S

seconds-from-dateTime 関数 196
seconds-from-duration 関数 197
seconds-from-time 関数 198
self 軸 68
short タイプ 43
SQL ステートメント
  ヘルプを表示する 235
sqlquery 関数 3, 199
starts-with 関数 202
string 関数 202
string タイプ 43
string-join 関数 203
string-length 関数 204
string-to-codepoints 関数 204
subsequence 関数 205
substring 関数 205
substring-after 関数 206
substring-before 関数 207
sum 関数 208

T

time タイプ 43
timezone-from-date 関数 209
timezone-from-dateTime 関数 210
timezone-from-time 関数 210
token タイプ 44
tokenize 関数 211
translate 関数 213
true 関数 214

U

Unicode 文字 63
unordered 関数 214
unsignedByte タイプ 44
unsignedInt タイプ 44
unsignedLong タイプ 44
unsignedShort タイプ 44
untyped タイプ 45
untypedAtomic タイプ 45
upper-case 関数 215
URI
  ネーム・スペース 接頭部のバインディング 92

V

Visual Explain
  チュートリアル 239

W

where 節
  説明 107

X

XDM。XQuery および XPath のデータ・モデルを参照 5
XML データ
  シリアライズ 13
  DB2 データベースでの 照会 2
xmlcolumn 関数 3, 216
XMLEXISTS 関数 3
XMLQUERY 関数 3
XMLTABLE 関数 3
XQuery
  概要 1
  基本概念 1
  言語規則 15
  更新式 120, 126
  更新式の結合 120
  サイズおよびデータ・タイプの制限 229
  式 55
  静的に既知のネーム・スペース 14
  リソース 17
  SQL からの呼び出し 3
XQuery および XPath のデータ・モデル 5
XQuery 更新
  エラー 121
XQuery 更新でのエラー 121
XQuery の定義済み関数 137
XQuery リファレンスの概要 i
XQuery を使用した XML データの更新 120

Y

yearMonthDuration タイプ 45
  正規化された形式 45
years-from-duration 関数 218
year-from-date 関数 217
year-from-dateTime 関数 217

Z

zero-or-one 関数 219

Printed in Japan

SC88-4448-01

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12

Spine information:

DB2 Version 9.5 for Linux, UNIX, and Windows

XQuery リファレンス