DBMaster

XMLツール・ユーザーガイド

SYSCOM

CASEMaker Inc./Corporate Headquarters

1680 Civic Center Drive
Santa Clara, CA 95050, U.S.A.

Contact Information:
CASEMaker US Division
E-mail : info@casemaker.com
Europe Division
E-mail : casemaker.europe@casemaker.com
Asia Division
E-mail : casemaker.asia@casemaker.com(Taiwan)
E-mail : info@casemaker.co.jp(Japan)

www.casemaker.com
www.casemaker.com/support

Document No. 645049-231260/DBM41J-M09052003-XMLT
発行日:2003-09-05

本文には記されていない新しい機能についての説明は、CASEMakerのDBMasterをインストールしてからREADME.TXTを読んでください。

登録商標
CASEMaker、CASEMakerのロゴは、CASEMaker社の商標または登録商標です。
DBMasterは、Syscom Computer Engineering社の商標または登録商標です。
Microsoft、MS-DOS、Windows、Windows NTは、Microsoft社の商標または登録商標です。
UNIXは、The Open Groupの商標または登録商標です。
ANSIは、American National Standards Institute, Incの商標または登録商標です。

ここで使用されているその他の製品名は、その所有者の商標または登録商標で、情報として記述しているだけです。SQLは、工業用語であって、いかなる企業、企業集団、組織、組織集団の所有物でもありません。

注意事項
本書で記述されるソフトウェアは、ソフトウェアと共に提供される使用許諾書に基づきます。

保証については、ご利用の販売店にお問い合わせ下さい。販売店は、特定用途への本コンピュータ製品の商品性や適合性について、代表または保証しません。販売店は、突然の衝撃、過度の熱、冷気、湿度等の外的な要因による本コンピュータ製品へ生じたいかなる損害に対しても責任を負いません。不正な電圧や不適合なハードウェアやソフトウェアによってもたらされた損失や損害も同様です。

本書の記載情報は、その内容について十分精査していますが、その誤りについて責任を負うものではありません。本書は、事前の通知無く変更することがあります。

目次

1 はじめに

DBMasterは、データベースおよびXMLドキュメント間のデータを受け渡しをするため、Javaベースのプラットフォームから独立した2つのツールを提供します。XMLトランスファー・テンプレート・ツールおよびXMLトランスファー・マッピング・ツールは、あなたがデータベースからXMLファイルまでデータをどのようにマッピングするか指定するカスタム・テンプレートを作成することを可能にします。

一旦テンプレートを作成したならば、データベースとXMLのファイル間のデータを同期させるプロセスを自動化することを支援するためDBMasterによって提供されたAPIを利用することができます。DBMasterは、このタスクを遂行するのを助けるためにC、C++、JavaのAPI、およびストアド・プロシジャーを提供します。

2 XMLトランスファー・テンプレート・ツール

XMLトランスファー・テンプレート(XTT)ツールの目的はデータベース・データおよびXMLドキュメントの間のカスタマイズ可能なブリッジを提供することです。ブリッジは、テンプレート・ファイルであるXMLトランスファー・テンプレート(XTT)の形式をとります。XTTファイルは、どのデータベース、表およびカラムを、どのXMLエレメントとアトリビュートへマッピングするかを決定します。XTTツールの中でドラッグ・アンド・ドロップ・オペレーションを使用して、マッピングを決定します。XTTツールは、XTTシンタックスが正確であるかどうかを検証し、さらにスキーマドキュメント(XSD)あるいはドキュメント・タイプ定義(DTD)を生成するようなタスクの実行を支援します。

データを変換するXTTツールの使用は4過程プロセスです。XTT構造を作成またはインポート、SQLクエリーを備えたデータベースにXTTオブジェクトをリンク、必要ならばDTDまたはXSDのファイルを生成、そして、最後にXMLドキュメントの生成です。

通常XTTオブジェクトのリンクは、あなたが既存のXML構造をXMLファイル、XSDあるいはDTDからインポートすることを意図している場合のみ必要でしょう。同様に、あなたがデータベースに基づいたXTTを作成していれば、リンクは必要ではないでしょう。しかしながら、ハイブリッドの状況は存在するかもしれません；例えば既存のXML構造を持っているにも

かかわらず、新しいデータ用に新しいエレメントを加える必要のある場合です。

XTTファイルは、データベース・表およびカラムからXMLファイルのエレメントおよびアトリビュートへのマッピングを定義します。XTTファイルは有効なXMLドキュメントに似ているシンタックスを備えたドキュメントです。エレメントは表およびカラムを定義し、アトリビュートはSQLクエリー、アトリビュート名、アトリビュート値、およびXTTを使用して生成されたXMLドキュメントに対するエレメント値を定義します。

➲ 例：

下記は表CARDからのデータをマッピングする完全なXTTファイルです。それはカラムFIRSTNAME、LASTNAME、およびTITLEをエレメントCARDのアトリビュートとして、またカラムNUMを子エレメントとしてマッピングします：

```
<?xml version="1.0" encoding="UTF-8"?>
<xtt:template xmlns:xtt="urn:schema-dbmaster-com:xml-template">
    <root>
          <CARD xtt:query="CARD_SQL0" xtt:command="select NUM, FIRSTNAME,
LASTNAME, TITLE, BMP from SYSADM.CARD">
              <xtt:attribute name="FIRSTNAME" value="$CARD_SQL0.FIRSTNAME"/>
              <xtt:attribute name="LASTNAME" value="$CARD_SQL0.LASTNAME"/>
              <xtt:attribute name="TITLE" value="$CARD_SQL0.TITLE"/>
              <NUM xtt:textvalue="$CARD_SQL0.NUM" />
          </CARD>
    </root>
</xtt:template>
```

上記の例におけるXTTが実行された場合、次のXMLファイルが生成されます：

```
<?xml version="1.0" encoding="US-ASCII" ?>
 <root>
  <CARD FIRSTNAME="Eddie" LASTNAME="Brown" TITLE="Manager">
   <NUM>1</NUM>
  </CARD>
 </root>
```

XTTツールは、XTTファイルのスクリプト化、検証、実行のためにシンプルなユーザ・インターフェースを提供します。次のセクションはユーザ・インターフェースについて記述し、自分でXTTファイルを作成し始めることの速習を助けるために手順を提供します。

2.1 XTTツールの基礎

このセクションは、XTTツール・ユーザ・インターフェースのエレメントについて、およびデータベースにログオンする方法について記述します。

XTTツールを開きデータベースにログインする

Windowsスタート・メニューからXTTツールを開く時、データベースにログインするように自動的に促されるでしょう。あなたが情報をエクスポートしたいデータベースを選択してください。ログインするためにデータベース上のアカウントを持たなければなりません。情報を必要とするすべての表にアクセス権限のあるアカウントを必ず使用してください。

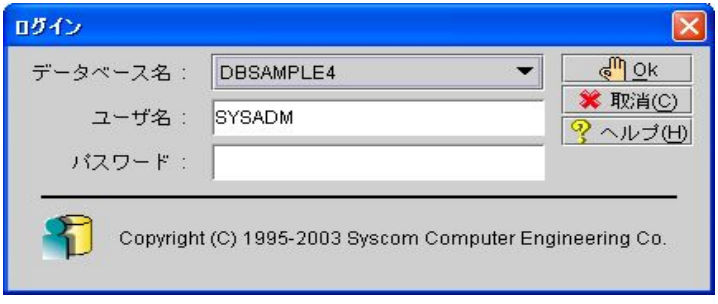


図2-1: ログイン・ダイアログ

➲ XTTツールを開きデータベースにログインする:

1. Windows**スタート・メニュー**から、**スタート>プログラム>DBMaster 4.1>XMLトランスファー・テンプレート**をクリックしてください。
2. **ログイン・**ダイアログでは、データベースを選択して、ユーザー名およびパスワードを入力してください。
3. **Ok**をクリックしてください。XTTツールは、データベース・スキーマパネル中のデータベース表を表示するでしょう。

メイン・コンソール

主要なコンソールは5つの論理的なエリアに分割することができます。以下の図2-2を参照ください。

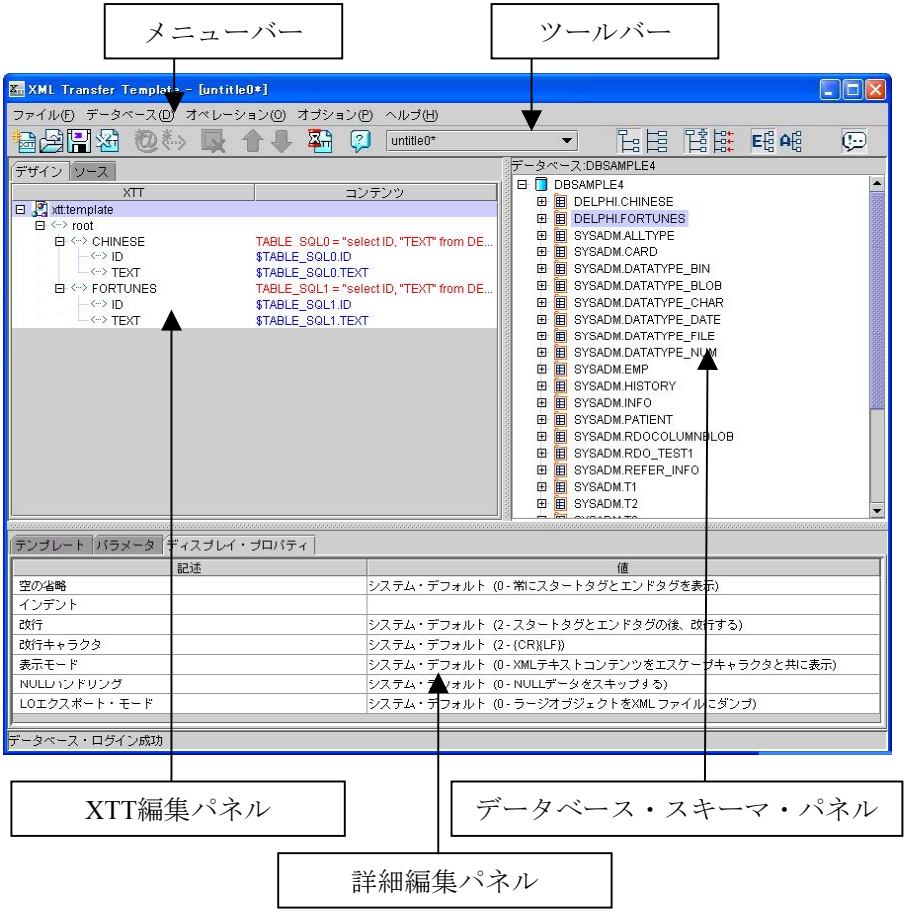


図2-2: メイン・コンソールのエレメント

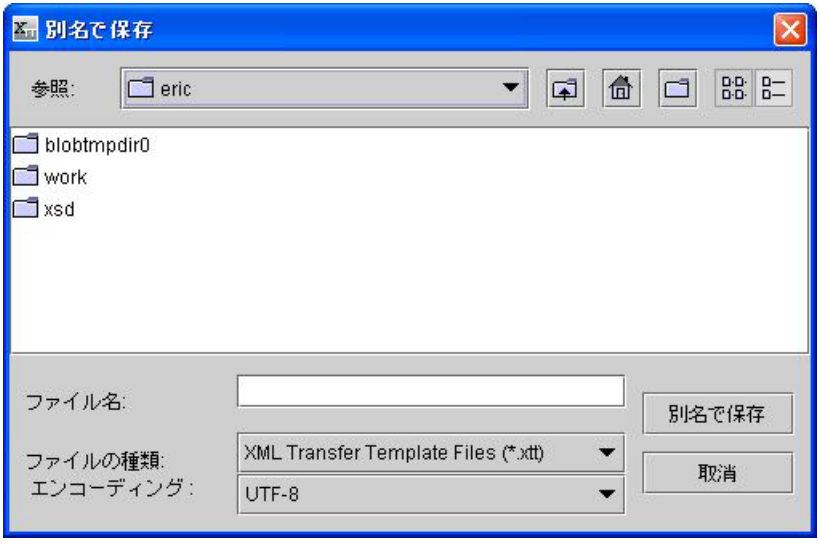
メニューバー

メニューバーは5つのメニューから成ります:**ファイル**、**データベース**、**オペレーション**、**オプション**および**ヘルプ**です。メニューアイテムは、それらを使用することができない場合、無効になります。各メニューアイテムの機能に関しては次のセクションを参照してください。

ファイル

ファイルメニューは次のアイテムから成ります：

- **新規XTT>空のXTT**：新しい空のXTTを作成します。詳しくは*空のXTTファイルを作成*を参照してください。
- **新規XTT>DTDをインポート**：DTDファイルに基づいた新しいXTTを作成します。詳しくは*DTDファイルからXTTを作成*を参照してください。
- **新規XTT>XSDをインポート**：XSDファイルに基づいた新しいXTTを作成します。詳しくは*XSDファイルからXTTを作成*を参照してください。
- **新規XTT>XMLをインポート**：XMLファイルに基づいた新しいXTTを作成します。詳しくは*XMLファイルからXTTを作成*を参照してください。
- **XTTを開く**：デフォルト・ファイル拡張フィルタとしてXTTを備えた**開く**ダイアログを開きます。
- **終了**XTTを編集するパネルにおいて現在開いているXTTを閉じます。XTTが修正された場合、確認ダイアログが、変更を保存するかどうか訊ねるでしょう。
- **保存**：XTTを編集するパネルにおいて現在開いているXTTを保存します。XTTが以前に保存されていない場合、**別名で保存**ダイアログが開くでしょう。
- **別名で保存**：デフォルト・ファイル拡張子としてXTTを備えた**別名で保存**ダイアログを開きます。



- **DTDを生成**：デフォルト・ファイル拡張子としてDTDを備えた**別名で保存**ダイアログを開きます。詳しくは*DTDを生成*を参照してください。
- **XSDを生成**：デフォルト・ファイル拡張子としてXSDを備えた**別名で保存**ダイアログを開きます。詳しくは*XSDを作成*を参照してください。
- 最近のファイル：最近に開かれたファイルを表示します。
- **終了**：XMLトランスファー・テンプレート・ツールを終了します。

データベース

データベース・メニューは次のアイテムから成ります：

- **接続**：**ログイン**ダイアログを開始します。実行しているデータベース・リストは、ドロップダウンメニューに現われます。.
- **切断**：データベース・セッションを中止します。**データベース・スキーマパネル**中の内容がクリアされます。

- **リフレッシュ**：セッションがアクティブな場合、**データベース・スキーマパネル**をリフレッシュします。

オペレーション

オペレーション・メニューは次のアイテムから成ります：

- **挿入>エレメント**：XTTオブジェクト・ツリーに新しい空のエレメントを挿入します。詳しくは新しいエレメントとアトリビュートを追加を参照してください。
- **挿入>アトリビュート**：XTTオブジェクト・ツリーに新しい空のアトリビュートを挿入します。詳しくは新しいエレメントとアトリビュートを追加を参照してください。
- **アンドゥ**：最後の修正の前の状態にXTTオブジェクト・ツリーを戻します。
- **コピー**：XTTオブジェクト・ツリーおよびすべての子孫の選択されたノードをコピーします。
- **カット**：XTTオブジェクト・ツリーおよびすべての子孫の選択されたノードをカットします。
- **貼り付け**：最後に削除されたかコピーされたXTTオブジェクト・ツリーおよびすべての子孫のノードを貼り付けます。
- **削除**：XTTオブジェクトツリーおよびすべての子孫の選択されたノードを削除します。
- **実行**：XTTファイルを実行します。詳しくはXMLデータを作成を参照してください。
- **検証**：エレメント用の変数参照が親エレメントに存在するかどうかチェックし、エレメントのSQLコマンドがデータベースにおいて有効かどうかチェックします。

オプション

オプション・メニューは次のアイテムから成ります:

- **ユーザー設定**：ユーザ設定ダイアログを開始します。
- **ツリー・オペレーション・オプション**：ツリー・オペレーション・オプション・ダイアログを開始します。

ヘルプ

ヘルプ・メニューは次のアイテムから成ります。：

- **ヘルプ**：オンライン・ヘルプを開きます。
- **ウェブサイト**：ブラウザ・ウィンドウでウェブサイトwww.dbmaker.comを開きます。
- **バージョン情報**：XMLトランスファー・テンプレートに関する情報を表示します。製造年月日、バージョン番号、CASEMaster技術サポートemailアドレスおよびwww.dbmaker.comへのリンクを含みます。

ツールバー

このセクションは、ツールバー・アイテムとそれらの同等なメニューバー・オペレーションを示します。

ファイル・オペレーション

-  空の新規XTT＝メニューバー>ファイル>新しいXTT>空のXTT
-  開く＝メニューバー>ファイル>XTTを開く
-  保存＝メニューバー>ファイル>保存
-  閉じる＝メニューバー>ファイル>終了

XTTツリー・オペレーション

-  アトリビュートオブジェクトを追加＝メニューバー>オペレーション>挿入>アトリビュート
-  エレメントオブジェクトを追加＝メニューバー>オペレーション>挿入>エレメント

-  ツリー・ノードを削除＝メニューバー>オペレーション>削除
-  上へ＝現在選択されたノードを、直前の兄弟ノードの前に移動します。エレメントは別のエレメントの前に移動することができますが、アトリビュートの前に移動することはできません。

-  下へ＝現在選択されたノードを、その直後の兄弟ノードの後に移動します。アトリビュートは別のアトリビュートの後に移動することができますが、エレメントの後に移動することはできません.

開かれたファイル

- 

  コンボボックス・ディスプレイはすべてのXTTファイル名を開きます。ファイルが編集された場合、ファイル名の後に星印(*)があるでしょう。あなたが異なるファイル名を選ぶ時、XTT編集パネルは再びロードし新しく選択されたファイルのXTTツリーを表示するでしょう。

オペレーション・オプション

-  トランスファーをラン＝メニューバー>オペレーション>実行
-  ヘルプ＝メニューバー>ヘルプ>ヘルプ
-  子として挿入＝(ドラッグ・アンド・ドロップ・オペレーションの場合)エレメントを子として挿入します。
-  同階層として挿入＝(ドラッグ・アンド・ドロップ・オペレーションの場合)エレメントを同階層として挿入します。
-  新規ノードを追加＝(ドラッグ・アンド・ドロップ・オペレーションの場合) 新しいノードを加えます。
-  ソース構造によるリンク＝(ドラッグ・アンド・ドロップ・オペレーションの場合) ソース構造によってリンクします。

- すべてエレメントとして＝選択されたエレメントの下に新しいエレメントを加えます(**挿入オペレーション**ルールに従います)。
- すべてアトリビュートとして＝選択されたエレメント内に新しいアトリビュートを加えます。

カスタマイズ・ダイアログを表示＝オペレーション・オプション—表またはビューに対するドラッグ・アンド・ドロップ・オペレーション実行時に選択された場合、**カスタマイズ**・ダイアログを表示します。

XTT編集パネル

XTT編集パネルは、2つのビューから成ります：デザイン・ビューおよびソース・ビューです。

デザイン | ソース

```
<?xml version="1.0" encoding="UTF-8"?>
<xtt:template xmlns:xtt="urn:schema-dbmaker-com:xml-template">
<xtt:parameter name="teacher_id" default="10000"/>
  <root>
    <DATATYPE_CHAR xtt:query="DATATYPE_CHAR_SQL0" xtt:command=
      <xtt:attribute name="C1" value="$DATATYPE_CHAR_SQL0.C1"/>
      <C2 xtt:textvalue="$DATATYPE_CHAR_SQL0.C2" />
    </DATATYPE_CHAR>
  </root>
</xtt:template>
```

XTT編集パネルのソース・ビュー

デザイン・ビューはXTTオブジェクト・ツリーを含んでおり、XTTが作成されるか開かれる場合、デフォルトによって表示されます。XTTオブジェクト・ツリーはXTTファイル自体(それはウェルフォームドなXMLドキュメント)の論理的な表現です。XTTオブジェクト・ツリーは5つのエレメント型から成り、詳細は表2-1に記述されています。

XTTオブジェクト・タイプ	ツリー・ノード名	コンテンツの説明
<xtt:template>	xtt:template	(なし)
<root>	root	(なし)
<xtt:attribute>	アトリビュート「name」の値	アトリビュート「value」の値
クエリーとテキスト値のないユーザ定義エレメント	エレメントのタグ名	(なし)
クエリーがあるユーザ定義エレメント	エレメントのタグ名	アトリビュート「xtt:query」の値 + '=' + アトリビュート「xtt:command」の値
テキスト値があるユーザ定義エレメント	エレメントのタグ名	アトリビュート「textvalue」の値'

表2-1: XTTオブジェクト・ツリーのエレメント型.

デザイン・ビューに現われる5つのXTTエレメント型は、表2-2の中で例証されます。

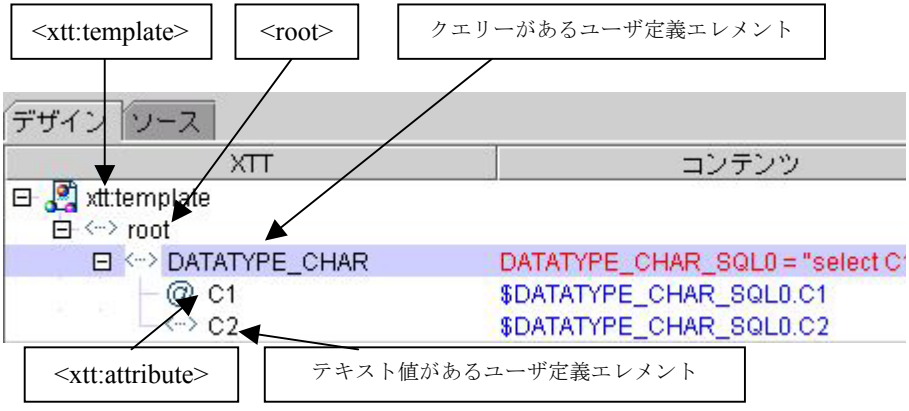


表2-2: XTTオブジェクト・ツリーのエレメント型のデザイン・ビュー

XTTツリー・オブジェクトを選択するには、その上で左クリックします。それは青色でハイライト表示されるでしょう。デザイン・ビュー上で右クリックすれば、ハイライトされたオブジェクトに応じたポップ・アップメニューが現われるでしょう。オブジェクト上で右クリックしても、それは選択されないでしょう。表 2-3は、異なる選択されたエレメントに対するポップ・アップメニュー内容を要約します。

ノード・タイプ	ポップアップ・メニュー
<xtt:template>	N/A

アトリビュート	● 「エレメントへの変更」-選択されたアトリビュートノードは、同じ名前を備えたエレメントと取り替えられるでしょう。**value**アトリビュート中のデータは、新しいエレメントの**textvalue**アトリビュートにコピーされるでしょう。 ● カット—ctrl-Xと同じ ● コピー—ctrl-Cと同じ ● 削除—ツリーから現在の選択されたノードを取り除く

ユーザ定義エレメント	● エレメント—選択されたエレメントの下に新しいエレメントを作成します。新しいエレメントが兄弟もしくは子として挿入されるかどうかは、**子として挿入**あるいは**同階層として挿入**オプションのどちらが選択されたかによって決定します。 ● アトリビュート—選択されたエレメントのための新しいアトリビュートを作成します。 ● 「アトリビュートへの変更—オブジェクトがサブノード(アトリビュートあるいはエレメントのいずれか)を持っていない場合のみ、提供されます。選択されたエレメントは、同じ名前を備えたアトリビュートと取り替えられるでしょう。**textvalue**の中のデータは、新しいアトリビュートの**value**アトリビュートにコピーされるでしょう。 ● カット—ctrl-Xと同じ ● コピー—ctrl-Cと同じ ● 貼付け—ctrl-Vと同じ ● 削除—ツリーから現在の選択されたノードを取り除く

表 2-3: デザイン・ビューのXTT編集パネルにおいて利用可能なポップアップ・メニュー

ソース・タブは、XTTファイルのソース・コードを表示します。

データベース・スキーマ・パネル

データベース・スキーマ・パネルは、スキーマ・ツリーとして接続しているデータベースの表/ビューを表示します。表/ビュー・ノードの拡張によって、それはそのサブノードとして各カラムを示すでしょう。スキーマ・

ツリーのノードはXTTツリー・パネルにドラッグすることができます。その後、それは選択されたセッティングに応じて、新しいノードを加えるか、あるいはXTTツリーへスキーマ情報をリンクするでしょう。

詳細編集パネル

詳細編集パネルは、XTTツリーの選択されたノードの詳細なプロパティを表示します。編集パネル中のテキスト・フィールドの総則は、テキスト・フィールドの外側もしくはエリアの外側にあるオブジェクトが選択されている場合、値がセットされるということです。例えば、次のフィールドが選択されていたか、XTTツリー選択が変更される後、名前フィールドはセットされるでしょう。変更はXTTツリー上で見ることができます。

XTT:TEMPLATE

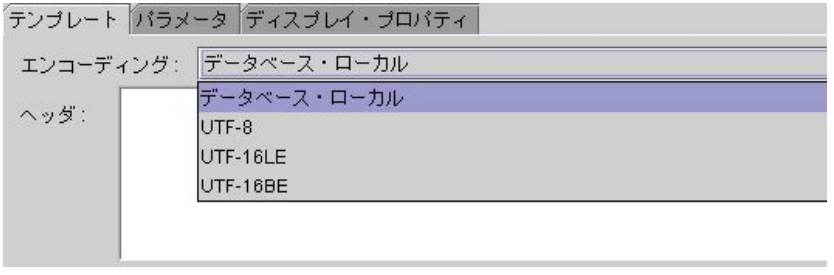
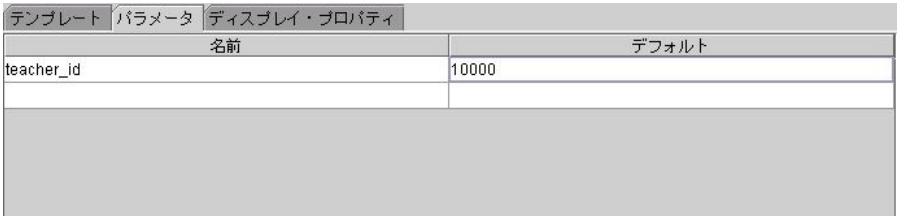


図 2-3: 詳細編集パネルのエンコーディング・メニュー

エンコーディング – エンコーディング・メニューは、任意のXML出力ファイルのためのテキスト・エンコーディングを指定します。選択はデータベース・ローカル(データベースで設定されたテキスト・エンコーディング)、UTF-8、UTF-16LEおよびUTF-16BEです。デフォルトはデータベース・ローカルです。データベース・ローカルがBig5で、エンコーディング・セッティングがデータベース・ローカルである場合、XML出力はBig5でエンコーディングされるでしょう。

ヘッダ—ヘッダ．ボックスは情報、例えばスキーマファイル、もしくは出力XMLファイルの中で適用可能なXSLを加える場所です。XTTエンジン

は、<?xml version=”1.0”…?>の後にこのブロック中の内容を出力XMLファイルへ書き出すでしょう。このヘッダー・ブロックの中へは、必ず有効なXMLコンテンツを入力してください。



パラメータ—望む限りの数のパラメーターを加えてもかまいません。パラメータ名はユニークでなくてはなりません。選択された列を削除するには「削除キー」を押してください。最後の空の列は削除することができません。パラメータ・デフォルトは空でもかまいません。

出力XMLファイルに利用可能なディスプレイ・セッティングはいくつかあります。各セッティングに対するデフォルト値は「システム・デフォルト」です。他の方法で指定されない限り、ディスプレイ・セッティングはすべてのXTTオブジェクトに適用されるでしょう。

空の省略—

0—常にスタートタグとエンドタグを表示—コンテンツが空の場合でも、常にエレメントの開始と終了のタグを表示します。

1—子エレメントがない場合、エンドタグを省略—サブエレメントが生成されない場合に、終了タグの略した形式を使用します。例えば
<Department id=”1001” />

2—エレメントが空の場合、スタートタグとエンドタグを隠す—エレメントのコンテンツが空の場合、つまりテキスト、アトリビュート、子エレメントがすべてない場合、開始と終了のタグを隠します。エレメントがアトリビュートのみを持っている場合は、タイプ1の省略を使用するでしょう。

インデント—テキストエディターに表示されるソース・ドキュメント中で各サブレベルに適用されるインデントスペースの数を指定します。例えば、数が2である場合、ルートのスタート・タグはつのインデントスペースになるでしょう。また、ルートのサブノードは4つのインデントスペースになるでしょう。

改行—

0—改行しない。

1—エンドタグの後、改行する。.

2—スタートタグとエンドタグの後、改行する。

改行キャラクター—改行として加えられる文字.

0—{CR}

1—{LF}

2—{CR}{LF}.

表示モード—テキスト・データを表示する方法。一般にテキスト文字「<」はテキスト・コンテンツの中で「lt;」に取り替えらます。しかし、それがCDATAセクションで囲まれているか、またはテキスト・コンテンツがそれ自身既にXMLフラグメントでありXMLコンテンツの一部として出力XMLへ加えることができる場合、「<」を使用することができます。テキスト・データを表示するために3つの異なる方法があります。

0—XMLコンテンツをエスケープキャラクタと共に表示。

1—CDATAセクションを使用してテキスト値を囲む。

2—コンテンツをネイティブXMLテキストに。

NULLハンドリング—NULLデータを扱う方法を指定します。

0—NULLデータをスキップする。—もしそれがアトリビュートである場合は、何もしません。

1—空エレメント・コンテンツを表示。—例えば<NAME></NAME>。

2—‘NULL’テキストを表示。—例えば<NAME>NULL</NAME>

LOエクスポートモード—ラージ・オブジェクトを扱う方法を指定します。.

0—ラージオブジェクトをXMLファイルへダンプ。それがBLOBタイプ・CLOBタイプ・データである場合はストリングを、それがBLOBタイプ・データである場合は16進数フォーマットで書き出します。

1—ラージオブジェクトを外部ファイルに保存。

ユーザ定義エレメント

下記は、XTTオブジェクトツリーのユーザ定義エレメントのプロパティを表示しています。

図2-4：詳細編集パネルのユーザ定義エレメント

名—エレメント名。それは英大小文字識別で空になりえません。

値—テキスト値。これは定数テキストおよび変数参照の両方を持つことができるテキスト表現です。例えば、"886 - $SQL1.PHONE"のようにすべての電話番号データに国コードを加えたいと思う場合、"886 –"は定数テキストで"$SQL1.PHONE"は変数参照です。XTTエンジンはエレメントのテキスト値として出力XMLファイルへ両方を連結するでしょう。

ブラウズ・ボタン—ブラウズ・ボタンは現在のレベルで利用可能なパラメータおよび変数参照をすべてリストするでしょう。変数名を選び、ブラウ

ズ・ボタンの隣のテキスト表現フィールドにそれらを挿入することができます。

図 2-5: 詳細編集パネルのユーザー定義エレメントのクエリー・ビュー

エレメントは、その中にクエリー・プロパティを埋め込むことができます。有効なXTTエレメントは両方のフィールド(名前とコマンド)が指定されているか、あるいは、空のままのどちらかです。

名—クエリー名。それはサブノードの中で変数参照として使用されます。

コマンド—SQLクエリー文。その文は結果セットを生成することが可能でなければなりません。例えば、**delete table**文をここに入力することはできません。

ディスプレイ・セッティングはxtt:templateのものに似ています。しかし、デフォルトは「テンプレート設定に従う」です。

NULLハンドリング—3つではなく2つの選択があります。「テンプレート設定に従う」を選択しNULLハンドリングが「0—NULLデータをスキップする」の場合、それは「1—空のコンテンツを表示」として扱われるでしょう。

1—空のコンテンツを表示。例えば、<NAME></NAME>.

2—「NULL」を表示。例えば、<NAME>NULL</NAME>.

アトリビュート・ノード

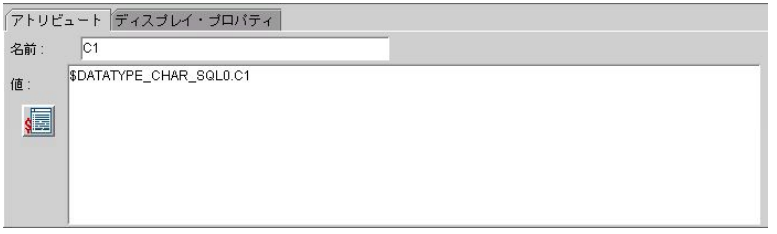


図 2-6: 詳細編集パネルでみるアトリビュート

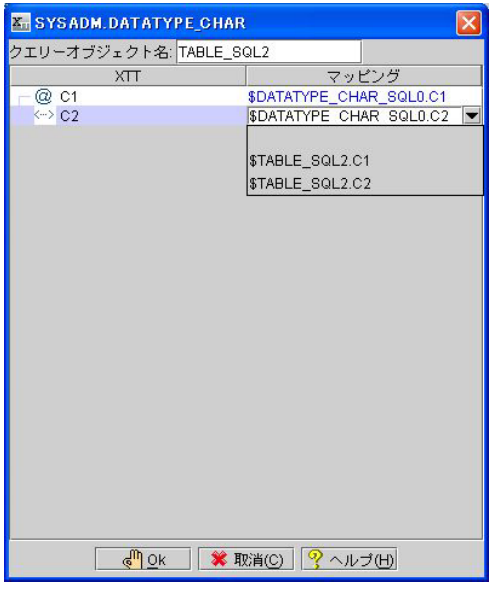
名前—アトリビュート名。名前フィールドは空になりえません。アトリビュートは同じ親エレメントに対してユニークな名前を持ってなければなりません。.

値—アトリビュート値；テキスト表現フィールド。

アトリビュート・ノードのための2つのディスプレイ・セッティングだけがあります；NULLハンドリングおよびLOモード。両方ともデフォルトは「テンプレート設定に従う」です。

カスタマイズ・ダイアログ

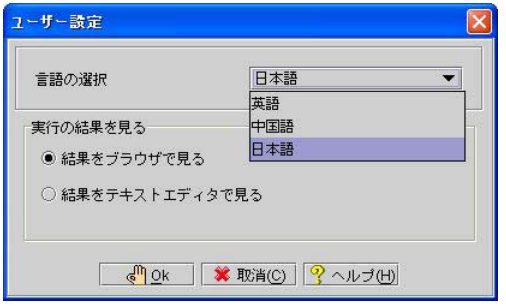
カスタマイズ・ダイアログを表示が選択されている場合、ドラッグ・アンド・ドロップを実行するとカスタマイズ・ダイアログが現れます。カスタマイズ・ダイアログの外観は、ツリー・オペレーション・オプション・ダイアログで選択されたセッティングに依存します。**ソース構造によるリンク**（リンク・モード）が選択されているか**新規ノードを追加**（追加モード）が選択されているかに依ってカスタマイズ・ダイアログは2種類のGUIのどちらかを提供します。



カスタマイズ・ダイアログ（リンク・モード）

ユーザ設定ダイアログ

ユーザ設定ダイアログはユーザ・インターフェースの言語、および実行の結果を見る方法を選択する場所です。ユーザ・インターフェース言語として英語、中国語あるいは日本語を選ぶことができます。XTTを実行した時に出力を見る方法として、あなたのデフォルトXMLブラウザ、あるいはあなたのデフォルト・テキストエディターを選択することができます。



ツリー・オペレーション・オプション・ダイアログ

ツリー・オペレーション・オプション・ダイアログは、ドラッグ・アンド・ドロップ・オペレーションの振る舞いを選択する場所です。データ・ベース・オブジェクトをエレメントあるいはアトリビュートとして追加；オブジェクトを子ノードあるいは兄弟ノードとして追加；オブジェクトを新しいツリーとして追加あるいはデータを既存のエレメントあるいはアトリビュートにリンク、を選択することができます。

2.2 新しいXTTを作成

XTTはXMLデータ・ファイル生成のためのテンプレートです。XTTは4つの方法のうちの1つで作成することができます：空のXTTファイル、DTDファイル、XSDファイル、およびXMLファイルからです。

空のXTTファイルを作成

あなたが、データベースにXMLの形で表示したいデータを持っていて、XMLデータがどのようにフォーマットされるべきかの必須条件がなければ、空のXTTファイルの作成が役立ちます。空のXTTファイルはルート.ノードからのみ成ります。空のXTTファイルを作成した後に、エレメントとアトリビュートを加えてください。エレメントとアトリビュートを加えることについてもっと知るためには、*XTTを編集*を参照してください。.

➲ 空のXTTファイルを作成する：

1. 1. XTTツールを開いて、あなたが使用したいデータ・ベースにログインしてください。
2. **ファイル>新規XTT>空のXTT**をクリックしてください。ルート・ノードがXTTオブジェクト・ツリーに現われるでしょう。

DTDファイルからXTTを作成

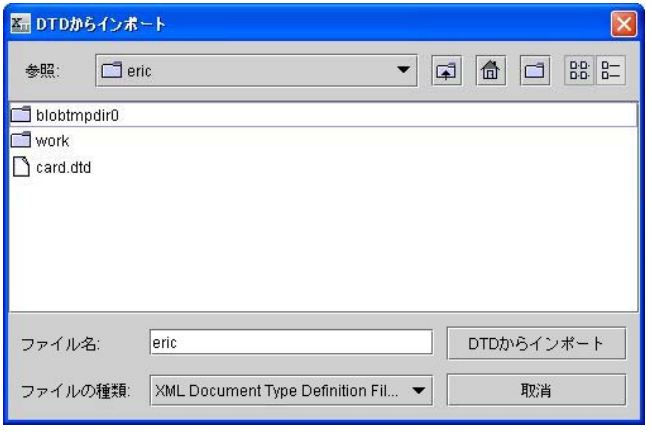
既存のドキュメント・タイプ定義(DTD)ファイルに基づいたXMLドキュメントの構造を定義したいと思うかもしれません。外部のDTDに基づく既存のXMLドキュメントを持っており、DTDをそれに順応させるデータ・ベース・データからXMLファイルを生成したければ、DTDファイルからXTTを作成することができます。

ルートエレメントをXTTのために指定してください。インポート・プロセス中に、ダイアログは、あなたがXTTのルートとしてどんな有効なエレメント定義も選ぶことを可能にします。

XTTを作成した後に、DTDの選択されたルート・ノードの下のエレメントおよびアトリビュートはすべて、XTTオブジェクト・ツリーに現われるでしょう。エレメントまたはアトリビュートの定義のどれも値を含んでいません—XTTを使用して、SQLデータをXMLファイルへ渡す方法はありません。これは、XTTオブジェクト・ツリーのエレメント・ノードの編集により遂行することができます。XTTオブジェクト・ツリーのノードを修正する方法についての詳細にはデータをエレメントとアトリビュートにマッピングを参照してください。

DTDからXTTを作成する：

1. XTTツールを開き、あなたが使用したいデータベースにログインします。
2. **ファイル>新規XTT>DTDをインポート**をクリックします。
3. **DTDをインポート**・ダイアログでは、インポートするDTDファイルを選択して**DTDをインポート**をクリックしてください



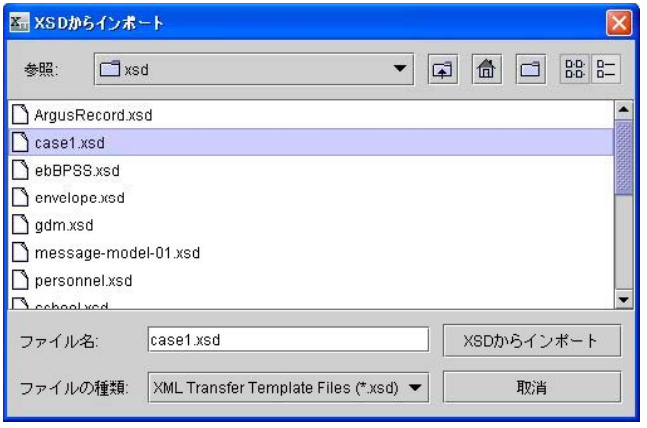
4. **DTD dom ツリーをルートとして選択**ダイアログでは、XTTオブジェクト・ツリーのルートエレメントになるエレメント定義を選択し、かつ**Ok**をクリックします。

XSDファイルからXTTを作成

XMLスキーマ定義(XSD)ドキュメント・ファイルによって定義されたXMLスキーマを基にXTTオブジェクト・ツリー構造を生成することも可能です。DTDファイルと同様に、ルートエレメントを指定する必要があります。どの有効なエレメント定義もXTTのルートとして選ぶことが可能です。XTTオブジェクト・ツリーの新しく作成されたアトリビュートあるいはエレメントのどの定義も値を持ちません；エレメントを修正し、XMLファイルにSQLデータを入れるための定義を指定してください。XTTオブジェクトツリーのノードを修正する方法についての詳細に関しては、*XTTを編集*を参照してください。

➲ XSDファイルからXTTを作成する：

1. XTTツールを開いて、あなたが使用したいデータベースにログインしてください。
2. **ファイル>新規XTT>XSDをインポート**をクリックしてください。
3. **XSDをインポート**ダイアログでは、インポートしたいXSDファイルを選択して**XSDをインポート**をクリックしてください。



4. **XSD dom ツリー・ノードをルートとして選択**ダイアログを開き、XTTオブジェクト・ツリーのルートエレメントとして定義したいエレメントを選択して、Okをクリックします。

XMLファイルからXTTを作成

XTT構造を定義したDTDまたはXSDファイルが入手不可能である状況で、既存のXML構造との一貫性を維持する必要があれば、XMLファイルからXTTを直接生成することができます。XTTツールは、XTTオブジェクト・ツリーを生成するためにあなたのXMLドキュメントの構造を解析するでしょう。DTDもしくはXSDを使用したXTT生成と、XTTを作成するこの方法間の主要な違いは、どのエレメントがXTTオブジェクトツリーのルートノードを構成するかに関して選択することができないことでしょう。

➲ XMLファイルからXTTを作成する：

1. XTTツールを開いて、あなたが使用したいデータ・ベースにログインしてください。
2. **ファイル>新規XTT>XMLをインポート**をクリックしてください。
3. **XMLをインポート**・ダイアログでは、インポートするXMLファイルを選択し**XMLをインポート**をクリックしてください。

2.3 XTTを編集

ルートノードおよび構造(XML、DTDあるいはXSDからXTTを作成する場合)をルートノードおよび構造(XML、DTDあるいはXSDからXTTを作成する場合)を作成した後、生成されたXMLファイルのためにコンテンツを提供したいと思うでしょう。空のXTTにおいては、これはほとんどデータベース・スキーマ・パネル中の表からXTTオブジェクト・ツリーへのドラッグ・アンド・ドロップ・オペレーションで行います。*表を挿入*のセクションおよび*新しいエレメントとアトリビュートを追加*の中で、新しいXTTを作成する場合に遂行する必要がある主要なタスクについて記述します。

XML、DTD、あるいはXSDファイルに基づいて作成されたXTTのために、あなたはXTTオブジェクト・ツリーへアトリビュートおよびエレメント定義へクエリー文および値を加える必要があるでしょう。これらのタスクは*データをエレメントとアトリビュートにマッピング*のセクションに記述されます。

これらのタスクは相互に排他的ではありません。また、上記のガイドラインは有効なXTTドキュメントを作成する方法を速く理解することを可能にするために提供されただけです。時には、新しいXTTの中のエレメント定義を修正することを有用に感じるかもしれません。例えば、あなたがSQL表から特定の条件を満たす値だけを抽出したい場合です。あるいは、XTTがその基づくXMLスキーマと正確に一致することを確認する必要がないとします。その場合には、ドラッグ・アンド・ドロップ・オペレーションのみを使用して、あなたのXTTオブジェクトツリーを構築することが可能です。

デザイン・ビューについて

XTT編集パネルのデザイン・ビューはXTTオブジェクトツリーを表示します。空の新規XTTは、XTTオブジェクト・ツリーがXTTテンプレート・ノードで空のルート・ノードだけを含んでいます。XML、XSDあるいはDTDから作成されたXTTオブジェクト・ツリーは、異なるルートを持つで

しょう。デザイン・ビューにおいて利用可能なオブジェクトおよび機能についての詳細な情報に関しては*XTT編集パネル*を参照してください。

表を挿入

子か兄弟として表を挿入することができます。第一に、挿入する表はルートノードの子としてでなくてはなりません；xtt:templateノードにエレメントを加えることを試みると、ツールはエラーを返すでしょう。

表がXTTオブジェクト・ツリーに加えられる前に、どのカラムをエレメントとして加えるか、どのカラムをアトリビュートとして加えるか、どのカラムを選択はするがXTTオブジェクトツリーへは加えないか、およびどのカラムを選択しないか、を指定することができます。これはカスタマイズ・ダイアログで遂行されます。XTTオブジェクト・ツリーにオブジェクトを加える場合にカスタマイズ・ダイアログを表示したい場合、**カスタマイズ・ダイアログを表示**ボタンをクリックしてください。

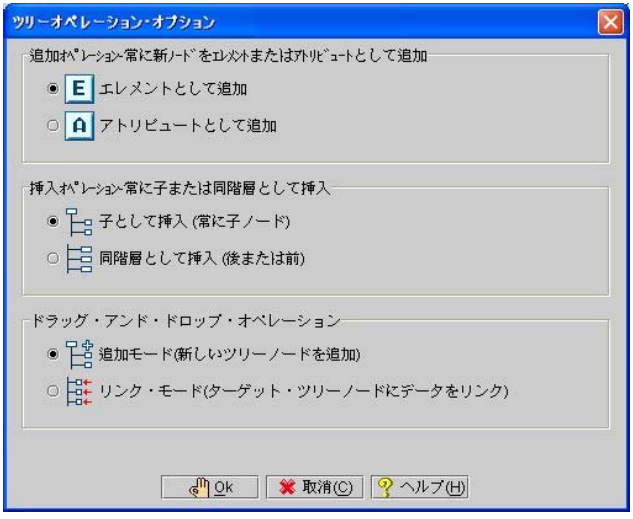
表を挿入する場合、それがXTTオブジェクト・ツリーへドラッグ・アンド・ドロップされた後、カスタマイズ・ダイアログが開き、クエリー・オブジェクト名とデータベース・オブジェクトの構造が表示されるでしょう。デフォルトとして、親および子オブジェクトの両方はエレメントとして挿入されます。子オブジェクトは、アトリビュートとして挿入するか、アトリビュート・オブジェクトあるいはエレメント・オブジェクトとしてそれらを挿入せずに、隠れデータベース・オブジェクトとして選択するか、あるいはデータベース・オブジェクトを選択しないよう指定することができます。

あなたが第1の表を挿入した後、次の表は、第1の表の子あるいは兄弟として加えることができます。表は、XTTオブジェクト・ツリーの中では常にエレメントとして表現されなければなりません。もしあなたが既存のファイルとしてXTTを作成したのならば、挿入するどんな表も既存のXTTエレメントの中の子あるいは兄弟となることが可能です。

➲ 子ノードとして表を挿入する：

1. オプション>ツリー・オペレーション・オプションをクリックしてください。

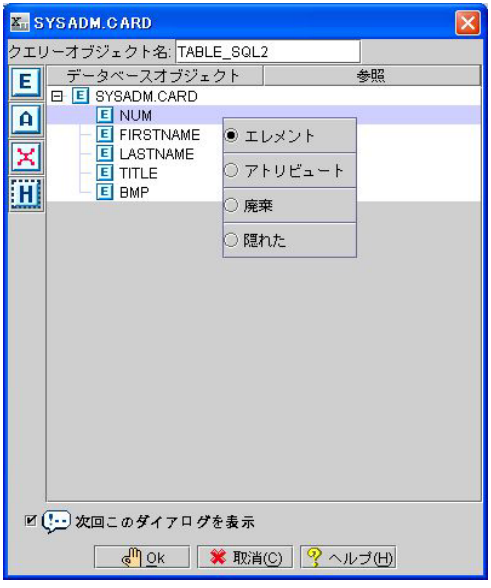
2. ツリー・オペレーション・オプション・ダイアログでは、**エレメントとして追加、子として挿入、追加モード**をクリックします。



3. データベース・スキーマ・パネルから表を選びます。

4. データベース・スキーマ・パネルから新しいノードの親になるXTTオブジェクト・ツリー・ノードまで表をドラッグしてください。

5. カスタマイズ・ダイアログでは、XTTに含みたくないカラムを選択して、**ノード挿入をキャンセル**をクリックしてください。SQLコマンドの中で選択したいがXMLファイルにはエクスポートしたくないカラムを選択して**隠れノードを挿入**をクリックしてください。アトリビュートとして加えたいカラムを選択して、**アトリビュート・オブジェクトを追加**をクリックしてください。エレメントとして加えたいカラムを選択して、**エレメント・オブジェクトを追加**をクリックしてください。



カスタマイズ・ダイアログ（追加モード）

6. **OK**をクリックしてください。表はXTTオブジェクトツリーに新しいエレメントとして現われるでしょう。カラムはそれらがカスタマイズ・ダイアログでどのように選択されたかに依存して、エレメントとアトリビュートとして現われるでしょう。

新しいエレメントとアトリビュートを追加

希望のデータ構造を作るためにXTTオブジェクト・ツリーにエレメントまたはアトリビュートを加える必要があるかもしれません。下記の手順で、空のエレメントおよびアトリビュートを加える方法について簡潔に記述します。エレメントとアトリビュートを定義しどうやってXTTオブジェクト・ツリーのデータベースから空のエレメントまでデータをマッピングするかについての情報としてご覧ください。

➲ 新しいエレメントあるいはアトリビュートを加える：

1. 新しいエレメントかアトリビュートを加えたいエレメントを選択してください。

2. 選択されたエレメントとあなたの新しいオブジェクトとの関係を指定してください：新しいオブジェクトを選択されたエレメントの兄弟にする場合、**同階層として挿入**をクリックしてください。新しいオブジェクトを選択されたエレメントの子にする場合、**子として挿入**をクリックしてください。

3. アトリビュートを加えるには、**アトリビュート・オブジェクトを追加**をクリックします。新しいエレメントを加えるには、**エレメント・オブジェクトを追加**をクリックします。

4. **詳細編集パネル**の名前ボックスに新しいオブジェクトの名前を入力して、Enterを押してください。

データをエレメントとアトリビュートにマッピング

エレメントはデータと関連付ける必要があります。データとエレメント・オブジェクトを関連させるために、親エレメント・オブジェクトはSQLクエリーを含んでいなければなりません。SQLクエリーは、子エレメント・オブジェクトにマッピングする表およびカラムを選択しなければいけません。SQLクエリーに親エレメントを関連させた後に、カラムに子エレメントを関連させなければなりません。

ツリー・オペレーション・オプションのソース構造によるリンクを使用して、エレメントとSQLクエリーを関連付けしてください。SQLクエリーにエレメントを関連させる場合、カスタマイズ・ダイアログを使用して選択されたSQLクエリーのカラムに子エレメントあるいはアトリビュートを関連させてください。XTTオブジェクト・ツリーの親エレメントにデータベース・スキーマ・パネルから表をドラッグすると、カスタマイズ・ダイアログは開くでしょう。

ソース構造によってエレメント・オブジェクトをリンクする時、カスタマイズ・ダイアログはクエリー・オブジェクト名および2つのカラム：XTTオブジェクト・カラムとマッピング・カラムを表示するでしょう。XTTオブジェクト・カラムは子エレメント、および表をドラッグしたエレメントのアトリビュートをすべて表示します。マッピング・カラムはどんな既存のコンテンツも表示し、マッピングしたいカラムのコンテンツを選択する

場所です。XTTオブジェクトに対応するSQLデータ・ソースを選ぶためにマッピング・カラムの列をクリックしてください。

➲ 空のXTTエレメントオブジェクトにデータをマッピングする：

1. データ・ベース・スキーマ・パネル中の表をクリックしてください。
2. 2. XTTオブジェクト・ツリーのエレメント・オブジェクトに表を引きずってください。表をドラッグしたエレメントは親エレメントになるでしょう。
3. **カスタマイズ・ダイアログ**では、各エレメントおよびアトリビュートオブジェクトのためにあなたが実行したいマッピングを選択してください。
   - **a)** データベース・データをマッピングしたいXTTオブジェクトに対応するマッピング・カラムをクリックしてください。
   - **b)** プルダウンメニューからXTTオブジェクトに見合う値を選んでください。値はデータベース中のカラム・データに相当します。それは、SQLクエリー名、1つのドット、そしてカラム名、のフォーマットで現われます。
4. データベース・カラムへマッピングしたいすべてのXTTオブジェクト・ノードに対するマッピング値を選択する作業を完成した場合、OKをクリックしてください。

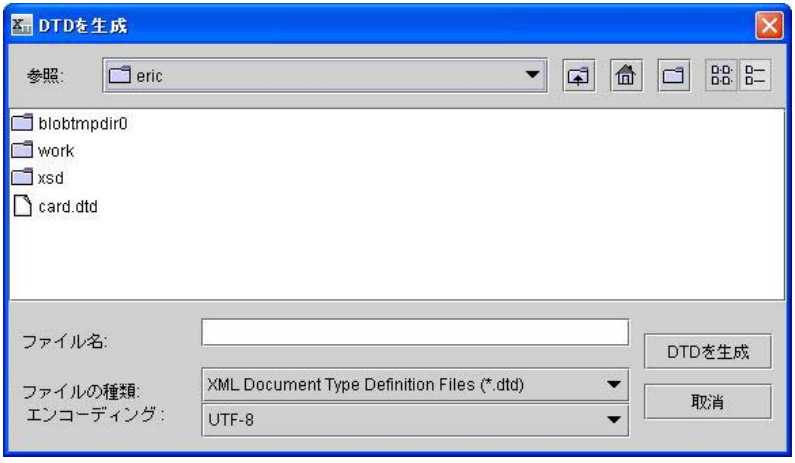
XTTとして保存

XTTオブジェクト・ツリーの編集を完成した後、XTTファイルを保存するべきです。保存されたXTTファイルはその後修正のために再び呼び出すか、あるいは自動的にデータベース・データをXMLファイルへ渡すためにストアド・プロシジャーの中で呼び出すことができます。

XTTクリックを保存するために、**保存**アイコン、あるいはメニューバーから**ファイル>保存**をクリックしてください。

2.4 DTDを生成

生成されたXMLファイルの構造について記述するために、ドキュメント・タイプ定義(DTD)ファイルを作成する必要があるとあなたの開発必要条件が要求する場合、XMLトランスファー・テンプレート・ツールの**DTDを生成**機能を使用することができます。



DTDファイル構造はソースXTTの構造と一致します。

➲ 例：

次のXTTファイルが与えられたとします：

```
<?xml version="1.0" encoding="US-ASCII"?>
<xtt:template xmlns:xtt="urn:schema-dbmaster-com:xml-template">
    <root>
        <CARD xtt:query="CARD_SQL0" xtt:command="select NUM, FIRSTNAME,
LASTNAME, TITLE, BMP from SYSADM.CARD">
            <NUM xtt:textvalue="$CARD_SQL0.NUM" />
            <FIRSTNAME xtt:textvalue="$CARD_SQL0.FIRSTNAME" />
            <LASTNAME xtt:textvalue="$CARD_SQL0.LASTNAME" />
            <TITLE xtt:textvalue="$CARD_SQL0.TITLE" />
            <BMP xtt:textvalue="$CARD_SQL0.BMP" />
        </CARD>
    </root>
</xtt:template>
```

生成されるDTDファイルは次のとおりです：

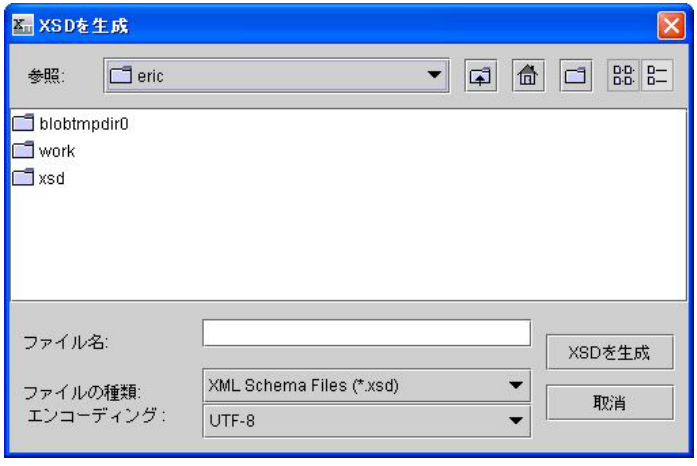
```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT root (CARD*) >
<!ELEMENT CARD (NUM, FIRSTNAME, LASTNAME, TITLE, BMP) >
<!ELEMENT NUM (#PCDATA) >
<!ELEMENT FIRSTNAME (#PCDATA) >
<!ELEMENT LASTNAME (#PCDATA) >
<!ELEMENT TITLE (#PCDATA) >
<!ELEMENT BMP (#PCDATA) >
```

➲ XTTオブジェクト・ツリーからDTDファイルを生成する

1. あなたがDTDに変換したいXTTが開いていることを確認してください。
2. **ファイル> DTDを生成**をクリックしてください。
3. **DTDを生成**ダイアログでは、出力パスを選択して、ファイル名とエンコーディングのタイプを指定してください。可能なエンコーディング・タイプは以下の通りです：**UTF-8、UTF-6LE、UTF-16BE**およびデータベース・ローカルです。
4. **DTDを生成**をクリックすると、DTDファイルは選択されたフォルダーの中に作成されるでしょう。

2.5 XSDを作成

XTTファイルにおいて代表された論理的な構造からのXMLスキーマファイルを生成したいと思うかもしれません。いくつかのツールは、XMLデータを解析することができることをスキーマ・ファイルに要求するかもしれません。スキーマ・ファイル構造はXTTオブジェクト・ツリー構造と一致しています。下記の例において、小さなXTTファイルはスキーマ・ファイルを生成するために使用されます。



➲ 例

オブジェクト・ツリー構造を備えた次のXTTファイルを与えられたとします。

```
<?xml version="1.0" encoding="US-ASCII"?>
<xtt:template xmlns:xtt="urn:schema-dbmaster-com:xml-template">
    <root>
         <CARD xtt:query="CARD_SQL0" xtt:command="select NUM, FIRSTNAME,
LASTNAME, TITLE, BMP from SYSADM.CARD">
              <xtt:attribute name="NUM" value="$CARD_SQL0.NUM"/>
              <xtt:attribute name="FIRSTNAME" value="$CARD_SQL0.FIRSTNAME"/>
              <xtt:attribute name="LASTNAME" value="$CARD_SQL0.LASTNAME"/>
              <xtt:attribute name="TITLE" value="$CARD_SQL0.TITLE"/>
              <BMP xtt:textvalue="$CARD_SQL0.BMP" />
         </CARD>
     </root>
```

```
</xtt:template>
```

結果のスキーマ・ファイル構造は次のとおりでしょう：

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="root">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="CARD" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="CARD">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="BMP" />
    </xsd:sequence>
    <xsd:attribute name="NUM" type="xsd:int"/>
    <xsd:attribute name="FIRSTNAME" type="xsd:string"/>
    <xsd:attribute name="LASTNAME" type="xsd:string"/>
    <xsd:attribute name="TITLE" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="BMP" type="xsd:string"/>
</xsd:schema>
```

➲ XTTオブジェクト・ツリーからのXSDファイルを生成する：

1. あなたがXSDに変換したいXTTが開いていることを確認してください。
2. **ファイル> XSDを生成**をクリックしてください;してください。
3. **XSDを生成**ダイアログでは、出力パスを選択して、ファイル名とエンコーディング・タイプを指定してください。可能なエンコーディング・タイプは以下の通りです：**UTF-8、UTF-6LE、UTF-16BE**およびデータベース・ローカル。
4. **XSDを生成**をクリックすると、XSDファイルは選択されたフォルダーの中に作成されるでしょう。

2.6 XMLデータを作成

必要なトランスファー・テンプレートおよびオプションのDTDを作成するか、XSDファイル後に、あなたは、データベースに格納されたデータを使用して、XMLファイルを生成する準備ができています。

トランスファーの実行機能はXTTをテストするのに最も役立ちます。XTTを作成した後、XMLドキュメント・オンデマンドを生成し、かつデータをあなたのXMLアプリケーションへ渡すためにそれを使用することができます。

次の例は完成したXTTテンプレート・ファイルおよび対応する出力ファイルを示します。

```
<?xml version="1.0" encoding="US-ASCII"?>
<xtt:template xmlns:xtt="urn:schema-dbmaster-com:xml-template">
    <root>
          <CARD xtt:query="CARD SQL0" xtt:command="select NUM, FIRSTNAME,
LASTNAME, TITLE, BMP from SYSADM.CARD">
              <NUM xtt:textvalue="$CARD_SQL0.NUM" />
              <FIRSTNAME xtt:textvalue="$CARD SQL0.FIRSTNAME" />
              <LASTNAME xtt:textvalue="$CARD_SQL0.LASTNAME" />
              <TITLE xtt:textvalue="$CARD_SQL0.TITLE" />
              <BMP xtt:textvalue="$CARD_SQL0.BMP" />
          </CARD>
      </root>
</xtt:template>
```

前のXTTファイル用の対応する出力ファイル：

```
  <?xml version="1.0" encoding="US-ASCII" ?>
- <root>
- <CARD>
  <NUM>1</NUM>
  <FIRSTNAME>Eddie</FIRSTNAME>
  <LASTNAME>Chang</LASTNAME>
  <TITLE>Manager</TITLE>
  <BMP>lobdir5\blobfile0.tmp</BMP>
  </CARD>
- <CARD>
  <NUM>2</NUM>
  <FIRSTNAME>Hook</FIRSTNAME>
  <LASTNAME>Hu</LASTNAME>
  <TITLE>Software Engineer</TITLE>
  <BMP>lobdir5\blobfile1.tmp</BMP>
  </CARD>
```

```
  </CARD>
- <CARD>
  <NUM>7</NUM>
  <FIRSTNAME>Oscar</FIRSTNAME>
  <LASTNAME>Tseng</LASTNAME>
  <TITLE>Software Engineer</TITLE>
  <BMP>lobdir5\blobfile6.tmp</BMP>
  </CARD>
- <CARD>
  <NUM>8</NUM>
  <FIRSTNAME>Jerry</FIRSTNAME>
  <LASTNAME>Liu</LASTNAME>
  <TITLE>Manager</TITLE>
  <BMP>lobdir5\blobfile7.tmp</BMP>
  </CARD>
  </root>
```

3 XMLトランスファー・マッピング・ツール

XMLトランスファー・マッピング(XTM)ツールは、あなたがXSL変換を使用して、XMLデータをデータベースへ渡すことを可能にします。XMLトランスファー・マッピング・ツールは3つの部分から成ります：XMLスキーマ部分(ソース・データとして使用しているXMLファイルのスキーマを表示)；SQLデータベース部分(データ・ベース・表を表示)；XTM部分(XMLスキーマからデータベース・表へのマッピングを表示)です。

ツールの使用は基本的な5ステップで要約されます：ソースXMLあるいはXSDファイルを開いてソースXMLスキーマを作成、データベースに接続、データベース・表およびソースXMLスキーマのエレメント／アトリビュートマッピングからXTM構造を生成、、最後に、XSLファイルとしてXTM構造を保存することです。

一旦XSLファイルが作成されれば、それはデータベース・データへのソース・スキーマに一致するすべてのXMLファイルを変換するために使用することができます。.

3.1 XTM ツールの基礎

XTTツールと違い、XTMツールは起動時にデータベースへの接続を要求しません。あなたがXTMファイルを作成するか開く場合、ツールはデータベース接続を作成し、*XMLスキーマ・ツリー*に記述されている、データベース・スキーマ・ツリーを表示するために使用します。

本章はXTMツールの主要なGUIについて説明します。

メイン・コンソール

XTMツールの主要なコンソールは、表3-1の中で例証されるような5つの主なエリアへ分類することができます。

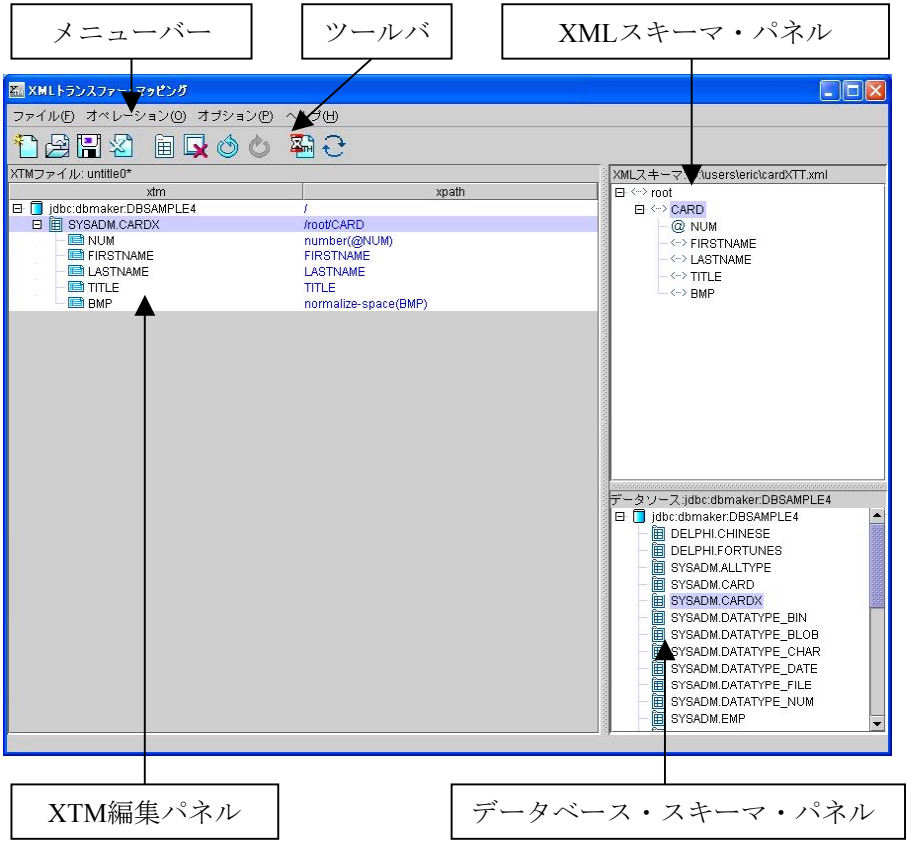


表3-1: XTMメイン・コンソールのエレメント

メニューバー

メニューバーは4つのメニューから成ります：**ファイル**、**オペレーション**、**オプション**および**サポート**です。メニューアイテムはそれらを使用することができない場合、無効になります。各メニューアイテムの機能に関しては次のセクションを参照してください。

ファイルメニューは次のアイテムから成ります：

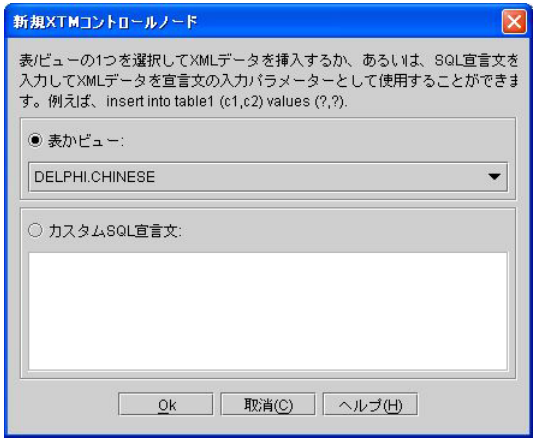
- **新規**：ソース・スキーマおよびデータベース情報を入力するようにあなたに促す**新規XTM**ダイアログを開きます。
- **開く**：デフォルト・ファイル拡張子フィルタとして.XSLを備えた**開く**ダイアログを開きます。
- **保存**：XTM編集パネルにおいて現在開かれているXTMをXSLファイルとして保存します。XTMが以前に保存されていない場合、**別名で保存**ダイアログが開くでしょう。
- **別名で保存**：デフォルト・ファイル拡張子として.XSLを備えた**別名で保存**ダイアログを開きます。
- **閉じる**：XTM編集パネルにおいて現在開いているXTMを閉じます。XTMが修正された場合、確認ダイアログが、変更を保存するかどうか訊ねるでしょう。
- **最近のファイル**：現在のセッションで開かれたXSLファイルのリスト
- **終了**：XTMツールを終了します。現在のXTMが保存されていない場合、デフォルト・ファイル拡張子として.XSLを備えた**別名で保存**ダイアログを開きます。

オペレーション

オペレーション・メニューは次のアイテムから成ります：

- **アンドゥ**：最後の修正の直前の状態にXTTオブジェクト．ツリーを戻します。
- **リドゥ**：最後に実行されたアクションを再び実行します。

- **挿入**：新規XTMコントロール・ノードを挿入します。**新規XTMコントロールノード**・ダイアログを開きます。

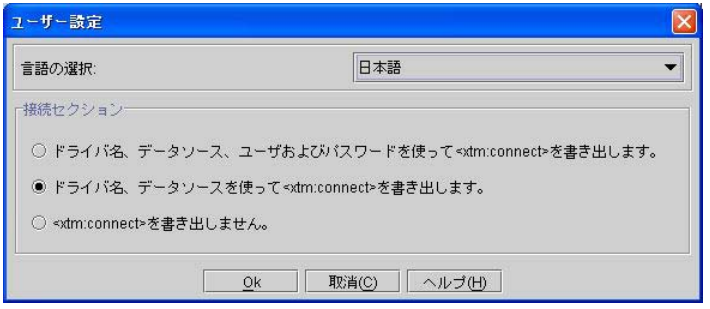


- **削除**：XTMオブジェクト．ツリーおよびすべての子孫の選択されたノードを削除します。
- **実行**：**XTMを実行**ダイアログを開きます。XTMダイアログは、直ちにXTMを実行しデータベースへデータを送るかあるいは後の使用の為SQLスクリプトを生成するかのオプションを提供します。
- **リフレッシュ**：データベース．スキーマをリフレッシュするためにデータベースをクエリーします。

オプション

オプション・メニューは次のアイテムから成ります：

- **ユーザー設定**：UIの表示言語の設定と、XSLファイルのconnectセクションのシンタックスを設定することを可能にする**ユーザー設定**ダイアログを開きます。



- **JDBC ドライバ**：他のデータ・ソースに接続することを可能にする**JDBCドライバ**・ダイアログを開きます。

ヘルプ

ヘルプ・メニューは次のアイテムから成ります。

- **ヘルプ**：オンライン・ヘルプを開きます。
- **ウェブサイト**：ウェブサイトwww.dbmaker.comにブラウザー・ウィンドウを開きます。
- **バージョン情報**：XMLトランスファー・マッピングに関する情報を表示します。製造年月日、バージョン番号、CASEMaster技術サポートemailアドレスおよびwww.dbmaker.comへのリンクを含みます。

ツールバー

- 新規ファイル　メニューバー>ファイル>新規
- ファイルを開く　メニューバー>ファイル>開く
- ファイルを保存　メニューバー>ファイル>保存
- ファイルを閉じる　メニューバー>ファイル>終了
- 新しい表／宣言文ノードを追加　メニューバー>オペレーション>挿入
- ノードの削除　メニューバー>オペレーション>削除

-  アンドゥ　メニューバー>オペレーション>アンドゥ
-  リドゥ　メニューバー>オペレーション>リドゥ
-  実行　メニューバー>オペレーション>実行
-  データベースをリフレッシュ　メニューバー>オペレーション>リフレッシュ

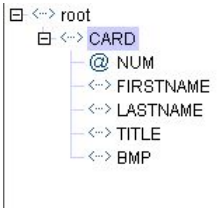
XTMオブジェクト・ツリー

XTMオブジェクト・ツリーはXTMファイルの構造の論理的な表現です。それは2カラムを含んでいます：XTMカラムおよびxpathカラムです。XTMカラムは、データベース・スキーマ・ツリーから挿入されたオブジェクトの視覚的なツリー表現を含んでいます。xpathカラムは、XMLスキーマ・ツリーに対応する位置に取り組むためのパスを含んでいます。

xtm	xpath
⊟ jdbc:dbmaker:DBSAMPLE4	/
⊟ SYSADM.CARDX	/root/CARD
NUM	number(@NUM)
FIRSTNAME	FIRSTNAME
LASTNAME	LASTNAME
TITLE	TITLE
BMP	normalize-space(BMP)

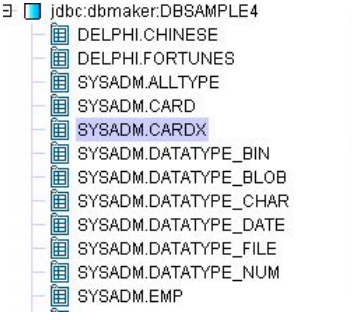
XMLスキーマ・ツリー

XMLスキーマ・ツリーは、XTMがアドレスを作るXML、DTDあるいはXSDファイルのスキーマの視覚的な表現を提供します。XTMが作成されており、XSLとして保存する後、XMLスキーマ・ツリーに表わされたスキーマに一致するどんなXMLファイルはデータソースとして使用することができます。



データベース・スキーマ・ツリー

データベース・スキーマ・ツリーは、選択されたデータベース内のデータベース表の論理的な表現を表示します。



3.2 XTMを生成

XTMの作成はデータソースを選択しデータベースに接続することを要求します。

データソースはXML、XSDあるいはDTDファイルでなくてはなりません。ソース・ファイル中のエレメントのうちの1つをルートに選ばなければなりません。あなたがXTMファイルを作成した後、子エレメントおよび選択されたエレメントのアトリビュートだけがXMLスキーマパネルにおいて可視できるでしょう。他のエレメントあるいはアトリビュートを含みたければ、新しいXTMファイルを作成する必要があるでしょう。

データベースに接続することは、データベースのドライバーおよび接続を要求します。DBMasterデータベースのための標準のドライバーはdbmaster.sql. JdbcOdbcDriverです。ドライバーに関してより詳細には、*新しいJDBCドライバーを加える*を参照してください。データベースへの接続は、データベースが開始していて、TCP/IPの上のコミュニケーションのチャンネルが開いていることを必要とします。また、あなたが表示したい表上の権限を備えたアカウント用のユーザー名およびパスワードが利用可能である必要があります。.

➲ 新しいXTMを作成する：

1. メニューバーから**ファイル>新規**を選択してください；**新規XTMファイル・**ダイアログが開きます。

2. **ファイルパス**・ボックスでは、ファイルパスをタイプするか、あるいはソース・スキーマ・ファイル(XML、XSDあるいはDTD)を見つけるためにブラウズ・ボタンを押してください。

3. ルート・エレメント・ボックスで示されたものと異なるルート・エレメントを選択するためには、右へのブラウズ・ボタンをクリックしてください。**ルート・エレメント選択**ダイアログが現われます。

4. ツリーのルートにしたいエレメントを選択してください。ノードを拡張し、かつその子エレメントを見るためにツリーのエレメントをダブルクリックしてください。

5. Okをクリックしてください。

6. データベース・ボックスでは、メニューからJDBCドライバーおよびデータ・ソースを選んでください。
7. データベース上のアカウントのユーザー名およびパスワードを入力してください。
8. Okをクリックしてください。選択されたXMLスキーマおよびデータベース表は、XMLスキーマ・パネルおよびデータベース・スキーマ・パネルにそれぞれ現われるでしょう。

➲ 既存のXTMを開く：

1. メニューバーから**ファイル>開く**を選択してください。
2. ファイル・チューザを使用して開きたいXSLファイルを選択してください。
3. ファイルパス・ボックスでは、ファイルパスを入力するか、あるいはソース・スキーマ・ファイル(XML、XSDあるいはDTD)を見つけるためにブラウズ・ボタンを選択してください。
4. ルート・エレメント・ボックスの中で示されたものと異なるルート・エレメントを選択するためには、右のブラウズ・ボタンをクリックしてください。**ルート・エレメント選択**ダイアログが現われます。
5. ツリーのルートにしたいエレメントを選択してください。ノードを拡張し、かつその子エレメントを見るためにツリーのエレメントをダブルクリックしてください。
6. Okをクリックしてください。
7. **データベース**・ボックスでは、メニューからJDBCドライバーおよびデータ・ソースを選んでください。
8. データ・ベース上のアカウントのユーザー名およびパスワードを入力してください。
9. **Ok**をクリックしてください。選択されたXMLスキーマおよびデータベース表は、XMLスキーマ・パネルおよびデータベース・スキーマ・パネルにそれぞれ現われるでしょう。

新しいJDBCドライバーを加える

標準装備されているものと異なるJDBCドライバーを使用することが必要であるかもしれません。専用のJDBCドライバが存在する限り、異なるベンダーによって提供されるデータベースに接続することも可能です。XTMツールはオプション・メニューを通じてこの機能を提供します。新しいドライバを加えるためにXTMファイルを開く必要はありません。

さらに、あなたが追加したドライバを編集したり削除することが可能です。デフォルト・ドライバは削除、編集することはできません。

➲ 新しいJDBCドライバーを加える

1. メニューバーから**オプション> JDBCドライバ**を選択してください。**JDBCドライバ・**ウィンドウが開きます。

2. **追加**をクリックします。**JDBCドライバ詳細**ダイアログが開きます。
Click **Add**. The **JDBC driver detail** dialog opens.

3. データソースのクラス名および接頭辞を適切なテキストボックスに入力してください。

4. クラス・パスを入力するか、あるいはドライバの位置へ導くべきブラウズ・ボタンをクリックしてください。
5. Okをクリックしてください。
6. 新しいドライバーはJDBCドライバーのリストに現われるでしょう。Okをクリックしてください。

3.3 XTMオブジェクト・ノードにxpath文をマッピングする

新しいXTMを作成した後に、次のステップは、XMLスキーマおよびデータベース表の中でノード間のマッピングを作成することです。最初に、XTMオブジェクト・ツリーにデータベース表を加える必要があるでしょう。次に、データが捜し出される場所でXMLスキーマ中のアドレスから適切にマッピングするXMLスキーマ・パスを加える必要があるでしょう。

XTMオブジェクト・ツリーの構造を作るためには、XTMオブジェクト・ツリーの希望の位置へデータベースからの表をドラッグ・アンド・ドロップしてください。第1のノードはルート・ノード上にドロップしなければなりません；後のノードは、ルート・ノード、あるいは表を表わす他のノードにドロップすることができます。表はカラムを表わすノードにドロップすることができません。あなたが表をドロップする時、それは、可視のカラムのすべてを備えたXTMカラムに現われるでしょう。

XTMオブジェクト・ツリー構造が完成した後、XTMオブジェクト・ツリーの希望のノード上にXMLスキーマ・パネルからのノードをドラッグ・アンド・ドロップしてください。文は、XMLスキーマ中のエレメントかアトリビュートアドレスのxpath表現に相当するxpathカラムに現われるでしょう。更に、数もしくはバイナリのデータがタイプする場合、データ特性はxpath文に先行するでしょう。データ特性文は文字データ・タイプに先行しないでしょう。xpathデータ特性および対応するデータ・タイプは表3-1の中で要約されます。

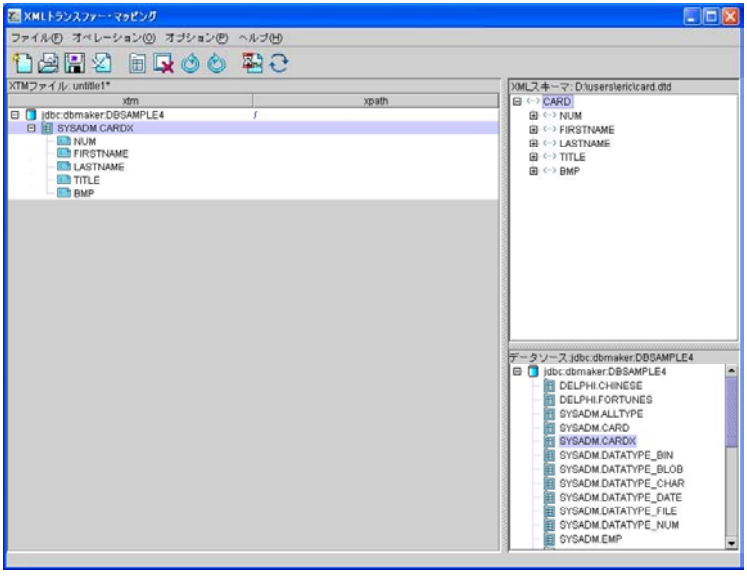
XPATHデータ特性	SQLデータ・タイプ
文字データ	Char
	Varchar
	Longvarchar
	Longvarbinary

XPATHデータ特性	SQLデータ・タイプ
	File
	Nchar
	Nvarchar
	Nclob
数値データ	Serial
	Smallint
	Int
	Float
	Double
	Decimal
標準スペース(バイナリ)データ	Binary
	Date
	Time
	Timestamp

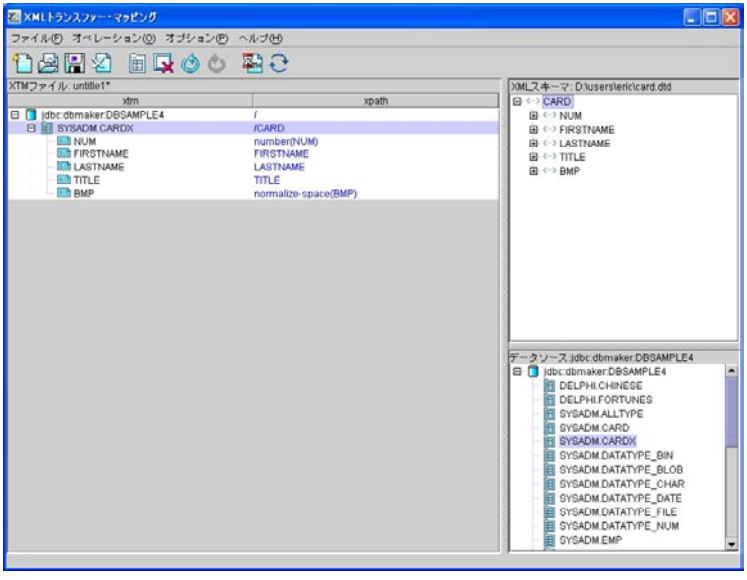
表3-1: xpathデータ特性および対応するSQLデータ・タイプ

➲ XTMの構造を建造する：

1. 新しいXTMを作成するか、あるいは既存のXTMを開いてください：
2. データベース・スキーマ・パネルからデータをインポートしたいと望む表をXTM編集パネルのxtmカラムまでドラッグ・アンド・ドロップします。



3. あなたがデータをマッピングしたいXTMノードへのエレメントかアトリビュートをドラッグ・アンド・ドロップしてください。Xpath文はxpathカラムに現われます。.



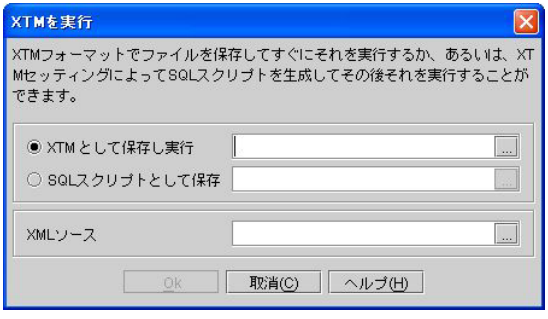
4. 終わった時、ファイル>保存をクリックしてください。

5. ブラウズ・ボタンであなたがXTMを保存したいフォルダを指定し、ファイル名を入力してください。ファイルは、拡張子.XSL.で自動的に保存されるでしょう。

3.4 XTMを実行

XTMを作成し、XSLファイルとしてそれを保存した後に、XTMの実行によりデータをソースXMLファイルからデータベースへ渡すことができます。XTMを実行する場合、XSLファイルとしてデータを保存しXTMを実行することに決めるか、あるいはSQLスクリプトとして変換を保存することができます。SQLスクリプトとしての保存はデータベースへデータを入力しないでしょう。データベースへデータを入力するためにはスクリプトを実行しなければなりません。

データ・トランスファーAPIあるいはストアド・プロシジャーを使用して、データ・トランスファーを自動化しようとしていれば、XSLファイルとして変換を保存するべきです。



SQLスクリプトとしてXTMを保存

SQLスクリプトとしてXTMを保存することは、あなたがXTMファイルをデータベース上で実行するのと同じオペレーションを実行するSQLスクリプトを作成することを可能にします。唯一の違いはSQLコマンドをスクリプト・ファイルに保存する点です。この方法は、実際にはデータベースにデータを保存しません。このオペレーションを実行する前に必ずXTMをXSLファイルとして保存してください。さもなければ、XMLデータあるいは変換をSQLスクリプト以外の形式でしか保存することがきなくなります。

➲ XTMを実行し、出力をSQLスクリプトとして保存する：

1. **オペレーション>実行**をクリックしてください。**XTMを実行**ウィンドウが現われるでしょう。
2. **SQLスクリプトとして保存**をクリックしてください。
3. **SQLスクリプトとして保存**ボックスで、XSLファイル用にフルパスおよびファイル名を入力するか、あるいはブラウズ・ボタンのクリックによりファイルおよびパスを選択してください。
4. **ソースXML**ボックスでは、XMLファイルがデータをインポートするべきフルパスおよびファイル名を入力するか、あるいはブラウズ・ボタンのクリックによりファイルおよびパスを選択してください。
5. **OK**をクリックしてください。XTMツールはSQLスクリプトを作成するでしょう。dmSQLプロンプトから、あるいはJDBAツールを使用することにより、データベースへデータを挿入するSQLスクリプトを実行することができます。

➲ 例：

SQLスクリプト出力：

```
INSERT INTO DELPHI.CHINESE (ID,TEXT) VALUES (?,?);
1,'lobdir1\clobfile0.txt';
2,'lobdir1\clobfile1.txt';
3,'lobdir1\clobfile2.txt';
4,'lobdir1\clobfile3.txt';
5,'lobdir1\clobfile4.txt';
…
```

XSLファイルとしてXTMを保存し実行

XSLファイルとしてXTMを保存および実行することはXTM APIまたはストアド・プロシジャーと同じオペレーションを実行します。XTMファイルの実行は、それを自動化する前にあなたが変換におけるエラーを見つけることを可能にし、与えられた変換に関してそれが希望の結果を生むことを保証するため、出力をテストすることを可能にします。

➲ XTMを実行し、かつ出力をXSLファイルとして保存します：

1. **オペレーション>実行**をクリックしてください。**XTMを実行**ウィンドウが現われるでしょう。

2. **XTMとして保存し実行**をクリックしてください。
3. **XTMとして保存し実行**ボックスでは、XSLファイル用にフルパスおよびファイル名を入力するか、あるいはブラウズ・ボタンのクリックによりファイルおよびパスを選択してください。
4. **ソースXML**ボックスでは、XMLファイルがデータをインポートするべきフルパスおよびファイル名を入力するか、あるいはブラウズ・ボタンのクリックによりファイルおよびパスを選択してください。
5. **OK**をクリックしてください。XTMツールはXSLファイルを作成し、データベース中の選択された表に新しいデータを加えるでしょう。

➲ 例：

XSL出力：

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0"
xmlns:xtm="dbmaster.xml.xtm.XMLTransferMap"
extension-element-prefixes="xtm">
<xsl:template match="/">
<xtm:connect driver="dbmaster.sql.JdbcOdbcDriver" datasource="jdbc:
dbmaster:DBSAMPLE4">
<xtm:table owner="DELPHI" name="CHINESE" select="/root/CHINESE">
<xtm:column name="ID" select="number(@ID)"/>
<xtm:column name="TEXT" select="@TEXT"/>
</xtm:table>
</xtm:connect>
</xsl:template>
</xsl:stylesheet>
```

4 XTM API 関数

XTMファイルを作成した後、あなたがトランスファー・プロセスを自動化する準備ができました。XTM APIは、あなたがプロセスを自動化することを可能にします。DBMasterは4つのXTMトランスファーAPIを提供します；C++の中のXTM API、Cの中のXTM API、JavaのXTM APIおよびXTM APIストアド・プロシジャーです。

XTM APIは、いくつかの共通オブジェクトをプロセス用に要求します：

- ソースXMLファイル—XTMのためのデータソースはXMLファイルの中で維持されます。.
- XTMファイル—このファイルは、ソースXMLからデータをいかに抽出し、データベースへ保存するかの方法を指定します。.

さらに、あなたがAPIをどのように実装するかに依り、次のプロパティが要求されるかもしれません：

- データベース接続—オプションとしてJDBC/ODBC接続をXTMエンジンへ渡すことが可能です。もし指定されれば、エンジンはこの接続を使用し、XTMファイルの「xtm:connect」エレメントを無視するでしょう。いくつかのベンダーによって提供されるJDBCドライバーのうちの一つを使用して、JDBC接続を確立することができます。しかし、ODBC接続はDbmaster ODBC接続ハンドルを使用しなければなりません。
- ユーザー名—データベース接続用のユーザー名。ある場合には、ユーザー名を「xtm:connect」に加えたいと思わないかもしれません。その

場合は、XTMファイルをプロセスする時にユーザー名が渡すことも可能です。もし指定されれば、このユーザー名が、XTMファイル中の「xtm:connect」エレメントの代わりに使用されるでしょう。

- パスワード—データベースへのログイン用パスワードは、ユーザー名のように、API引き数でXTMエンジンへ渡すことが可能です。

XTMプロセスのデータベース接続のための優先ルールは以下の通りです：:

1. **setConnection**メソッドの中で渡された接続の詳細。.
2. **setUser/setPassword**メソッドの中で渡されたユーザ/パスワードとの接続。
3. XTMファイルの<xtm:connect> で指定された接続情報。

プロパティがrunメソッドを呼ぶ前に二度以上セットされた場合、最後の入力値が使用されるでしょう。

4.1 C++のXTM API

XTMクラスは、以前に記述されるような必要なプロパティをセットするパブリック メソッドおよび実行関数を提供します。

パブリック メソッド:

コンストラクタとデコンストラクタ

シンタックス

```
XTM()
```

XTMオブジェクトを初期化します。

```
~XTM()
```

XTMオブジェクトを破壊します。

SETCONNECTION

データベース接続をセットします。このデータベース接続はDbmaster ODBCデータベース接続でなければなりません。

シンタックス

```
int setConnection( HDBC hdbc );
```

引き数	I入力/出力	説明
Hdbc	入力	データベース接続ハンドル (null不可).

戻り値	説明
SUCCESS	setConnectionとHdbcはSUCCESSあるいは次のエラー・コードを返すでしょう。
ERR_XTM_INV_API	JVM のロード失敗Java クラスdbmaster.xml.XTM が見つかりません dbmaster.xml.XTM のインスタンス作成に失敗 Javaクラスdbmaster.sql.JdbcOdbcConnectionが見つかりません SetConnectionの呼び出し失敗
ERR_XTM_PROCESS	可能な失敗の理由: dbmaster.sql.JdbcOdbcConnectionのインスタンス初期化に失敗

SETUSER

データベース接続のためにユーザをセットします。

シンタックス

```
int setUser ( char *user );
```

引き数	入力/出力	説明
User	入力	ユーザ名

戻り値	説明
SUCCESS	setUserはSUCCESSあるいは次のエラー・コードを返すでしょう。
ERR_XTM_INV_API	可能な失敗の理由: Java仮想マシンのロード失敗 JavaクラスBMaster.xml.XTMが見つかりません dbmaster.xml.XTM のインスタンス初期化に失敗 SetUserの呼び出し失敗

SETPASSWORD

データベース接続のためにユーザのパスワードをセットします。

シンタックス

```
int setPassword( char *password);
```

引き数	入力/出力	説明
password	入力	パスワード

戻り値	説明
SUCCESS	setPasswordはSUCCESSあるいは次のエラー・コードを返すでしょう。
ERR_XTM_INV_API	可能な失敗の理由: JVMのロード失敗 JavaクラスBMaster.xml.XTMが見つかりません dbmaster.xml.XTMのインスタンス初期化に失敗 setPassword の呼び出し失敗

GETERROR

run()、setConnection()、setUser()あるいはsetPassword()からのリターン・コードが、SUCCESSでない場合、エラー・メッセージを得るこのメソッドを使用してください。エラーに遭遇しない場合、このメソッドはNULLを返すでしょう。

シンタックス

```
const char * getError();
```

RUN

run()メソッドはXTMコンテンツの変換を実行します。XTMファイルは実はそれ自身XSLファイルです。XTMエンジンはアパッチXalan XSLTプロセッサーの拡張です。エンジンは、XTMファイル中の仕様によってソースXMLファイルを変換するでしょう。

シンタックス

```
int run ( const char * XtmFileName,
          const char * XmlFileName);
```

引き数	入力/出力	説明
XtmFileName	入力	XTMファイル名 (null不可).
XmlFileName	入力	ソースXMLファイル名 (null不可).

戻り値	説明
SUCCESS	エンジンは、XTM仕様にもとづいてソースXMLファイル処理を成功しました。
ERR_XTM_PROCESS	XTMプロセス・エラーの可能な理由は次の通り: XSTLプロセス・エラー—XTMあるいはソースXMLファイルが有効なXMLではありません。あるいは、XTMが有効なXSLファイルではありません。 XTMシンタックス・エラー

SQL Error	SQLコマンドでエラー発生.

例:

次のコードセグメントは、C++の中でのXTM APIの実装を示します。

```
/* an XTM object */
XTM transfer;
const char *error = NULL;
long rc = 0;

/* start transformation */
rc = transfer.run("c:\\temp\\abc.xsl","c:\\temp\\abc.xml");

/* check if process fails.
   if so, getError to get a detailed error message. */
if( rc != 0 )
{
error = transfer.getError();
printf("%s\n",error);
}
```

または

```
/* an XTM object */
XTM transfer;
const char *error = NULL;
long rc = 0;

/* set database connection,
XTM engine will use this connection,
rather than the xtm:connection specification in XTM file. */
rc = transfer.setConnection(hdbc);

/* start transfermation */
rc = transfer.run("c:\\temp\\abc.xsl","c:\\temp\\abc.xml");

/* check if process fails.
   if so, getError to get the detail error message. */
if( rc != 0 )
{
error = transfer.getError();
printf("%s\n",error);
}
```

4.2 CのXTM API

要約

XTM関数はソースXMLファイルからデータベースへコンテンツをトランスファーします。

シンタックス:

```
int Xtm(                char *xtmfile,
                        char *xmlfile,
                        char * msgBuffer,
                        int szMsgBuffer)

int XtmUP(              char * user,
                        char * password,
                        char * xtmfile,
                        char * xmlfile,
                        char * msgBuffer,
                        int szMsgBuffer)

int XtmH(               HDBC hdbc,
                        char * xtmfile,
                        char * xmlfile,
                        char * msgBuffer,
                        int szMsgBuffer)
```

3つの関数はすべてXTMをプロセスし、ソースXMLファイルからデータを検索し、データベースを更新するでしょう。3つの関数はすべてファイル名用の引き数でXTMファイル名およびソースXMLファイル名を指定する必要があります。関数XtmUPは、データ・ベース接続用のユーザー名およびパスワードを指定する必要があります。関数XtmHはデータベース接続を指定する必要があります。

引き数 (XTM用)

引き数	入力/出力	説明
XtmFileName	入力	XTMファイル名。有効なファイルパスでなくてはなりません。
XmlFileName	入力	ソースXMLファイル名(null不可)。
msgBuffer	出力	エラー・メッセージ用バッファーへのポインター。出力結果はnull終止。
szMsgBuffer	入力	msgBufferのサイズ

引き数 (XTMUP用)

引き数	入力/出力	説明
user	入力	ユーザ名:データベース接続を確立するために使われます。
password	入力	パスワード: データベース接続を確立するために使われます。
XtmFileName	入力	XTMファイル名。有効なファイルパスでなくてはなりません。
XmlFileName	入力	ソースXMLファイル名(null不可)。
msgBuffer	出力	エラー・メッセージ用バッファーへのポインター。出力結果はnull終止。
szMsgBuffer	入力	msgBufferのサイズ。

引き数 (XTMH用)

引き数	入力/出力	説明
Hdbc	入力	データベース接続ハンドル、null不可。

XtmFileName	入力	XTMファイル名。有効なファイルパスでなくてはなりません。
XmlFileName	入力	ソースXMLファイル名は、nullであってはなりません。
msgBuffer	出力	エラー・メッセージ用バッファーへのポインター。出力結果はnull終止。
szMsgBuffer	入力	msgBuffer.のサイズ

戻り値	説明
SUCCESS	エンジンは、XTM仕様にもとづいてソースXMLファイルの処理に成功しました。
ERR_XTM_PROCESS	XTMプロセス・エラーの可能な理由は次の通り: XSTLプロセス・エラー—XTMあるいはソースXMLファイルは有効なXMLではありません。あるいは、XTMは有効なXSLファイルではありません。XTMシンタックス・エラー。
SQL Error	SQLコマンドでエラー発生。.

例：

```
/* pass user name and password to XTM API. */
rc = XtmUP( user, password, xtm, xml, message, sizeof(message));

/* check if process fails.
   if so,message should contain detailed error information. */
if( rc != 0 )
{
  if( message != NULL )
{
    printf("%s\n\n",message);
  }
}
```

または

```
/* pass hdbc (odbc database connection handle ) to XTM API */
rc = XtmH(hdbc, xtm, xml,  message, sizeof(message));

/* check if process fails.
   if so,message should contain detailed error information. */
if( rc != 0 )
{
  if( message != NULL )
{
    printf("%s\n\n",message);
  }
}
```

4.3 JavaのXTM API

XTMプロセッサはさらにJavaクラスdbmaster.xml.XTMを提供します。

パブリック メソッド:

コンストラクタ:

XTMオブジェクトを初期化します。

シンタックス

```
XTM( )
```

SETCONNECTION

データベース接続をセットします。この関数はjava.sql.Connectionオブジェクトを引き数として受理します。その接続は、Dbmaster JDBCドライバーあるいは他のベンダーからのJDBCドライバーによって確立することができます。

シンタックス

```
void setConnection( Connection conn );
```

引き数	入力/出力	説明
conn	入力	クラスjava.sql.Connectionオブジェクト (null不可)

SETUSER

データベース接続のためにユーザをセットします。

シンタックス

```
void setUser ( String user);
```

引き数	入力/出力	説明
user	入力	ユーザ名

SETPASSWORD

データベース接続のためにユーザのパスワードをセットします。

シンタックス

```
void setPassword ( String password );
```

引き数	入力/出力	説明
password	入力	データベース接続用のパスワード

SETPARAMETER

XSLプロセッサー用のパラメーターをセットします。パラメーター値はXTMプロセス中に利用可能になります。シンタックスの中で***$parameter_name***としてそれを参照することができます。

シンタックス

```
void setParameters ( String name, Object value );
```

引き数	入力/出力	説明
name	入力	パラメーター名。パラメーター名は2つのストリングから成る資格のある名で、括弧({})で囲まれた名前空間URI、続いてローカル名です。名前がnullのURLを持っている場合、ストリングは、ローカル名のみを含んでいます。アプリケーションは、その名前の第1のキャラクタが「{}文字かどうか確かめるテストすることにより、、無効でないURIを安全にチェックすることができます。例えば、もしURIおよびローカル名が <xyz:foo xmlns:xyz="http://xyz.foo.com/yada/baz.html"/>で定義されたエレメントから得られるならば、資格のある名前は "{http://xyz.foo.com/yada/baz.html}foo"でしょう。接頭辞が使用されないことに注意してください。
value	入力	パラメーター値。

GETERROR

run()からのリターン・コードが、SUCCESSでない場合、エラー・メッセージを得るこのメソッドを使用してください。エラーに遭遇しない場合、このメソッドはNULLを返すでしょう。

シンタックス

```
String getError();
```

RUN

run()メソッドはXTMコンテンツの変換を実行します。XTMファイルは実はそれ自身XSLファイルです。XTMエンジンはアパッチXalan XSLTプロセッサーの拡張です。エンジンは、XTMファイル中の仕様によってソースXMLファイルを変換するでしょう。

シンタックス

```
int run ( String xtmfile,
          String xmlfile);
```

引き数	入力/出力	説明
xtmfile	入力	XTMファイル名。null不可.
xmlfile	入力	ソースXMLファイル名。null不可.

戻り値	説明
SUCCESS	エンジンは、XTM仕様にもとづいてソースXMLファイル処理を成功しました。

戻り値	説明
ERR_XTM_PROCESS	XTMプロセス・エラーの可能な理由は次の通り: XSTLプロセス・エラー—XTMあるいはソースXMLファイルが有効なXMLではありません。あるいは、XTMが有効なXSLファイルではありません。 XTMシンタックス・エラー
SQL Error	SQLコマンドでエラー発生

例:

```
/* create new XTM object */
XTM transfer = new XTM();

/* start transformation */
int rc = transfer.run(xtm, xml);

/* check if process fails.
   if so, getError for error detail. */
if ( rc != 0 )
{
  System.out.println(transfer.getError()+"\n");
}
```

4.4 XTM ストアド・プロシジャー

XTMストアド・プロシジャーは、C++、CおよびJava APIに似ているXTM変換関数を提供します。主要な違いは

- XTMとソースXMLファイル名のみを受理します。
- 現在のデータ・ベース接続のみを使用します。.
- 指定されたファイル名がサーバー・サイト上に存在します。.

ストアド・プロシジャー定義:

シンタックス

```
CREATE PROCEDURE XTM(
  VARCHAR(257)  XTMFILE     INPUT,
  VARCHAR(257)  XMLFILE     INPUT );
```

引き数	入力/出力	説明
XTMFILE	入力	XTMファイル名
XMLFILE	入力	ソースXMLファイル名

権限

このストアド・プロシジャーの所有者はSYSTEMです。ストアド・プロシジャーの権限はストアド・プロシジャーを実行するユーザと同じです。例えば、XTMファイルにユーザに挿入の権限がない表へのinsert文がある場合、SQLエラーが返されるでしょう。

例:

```
dmSQL> call XTM('c:\temp\case1.xsl ','c:\temp\case1.xml');
```

5 XTT API 関数

XTT APIは、データベースからファイルあるいはメモリ中のある位置へのXMLデータ出力を自動化するためにDBMasterによって提供されます。APIはC、C++、Javaで、そして、さらにストアド・プロシジャーの形式で提供されます。XTTプロセスのためには、いくつかのプロパティが必要になります:

- データベース接続—接続ストリングあるいは接続されているハンドルのいずれか。.
- XTT —XTTファイル名あるいはメモリ・バッファー中のXTTコンテンツのいずれか。
- 出力XML— XMLファイル名あるいはアロケートされた出力XMLコンテンツ用のバッファーのいずれか。
- パラメーター—テンプレート・ファイルに対するパラメーター値を備えたストリング。ひとつのストリングは複数ペアのパラメーター名および値を含むことが可能。各ペアはコンマによって分離してください。等号の左側はパラメーター名、右側はその値です。値はダブルクォートあるいはシングルクォートによって囲んでください。例えば "student_id='10012'、class_id='103'"のように。

5.1 C++のXTT API

XTTクラスは、前のセクションに記述されたような必要なプロパティおよび実行関数をセットするパブリック・メソッドを提供します。

パブリック メソッド:

コンストラクタとデコンストラクタ

シンタックス

```
XTT()
```

XTT オブジェクトを初期化します。

```
~XTT()
```

XTTオブジェクトを破壊します。.

SETDATABASECONNECTION

データベース接続をセットします。すでに接続ストリングを使用した呼び出しによって確立された接続がある場合、それが接続の前に切断されるでしょう。さもなければ、オブジェクトが消滅する時、接続は切断されるでしょう。つまりXTTが適切に消滅した場合、データソースは自動的に切断されるのです。また、接続は*setDatabaseConnection((HDBC)NULL).*を呼び出しすことによって強制的に切断することができます。

シンタックス

```
int setDatabaseConnection( const HDBC hdbc );
```

あらかじめ接続しているodbc接続ハンドルとのデータベース接続をセットします。

引き数	入力/出力	説明
Hdbc	入力	データベース接続ハンドル (null不可

```
int setDatabaseConnection( const char * connectionStr );
```

接続ストリングを使用してデータベース接続をセットし、データベースに接続して接続ハンドルを維持します。

引き数	入力/出力	説明
connectionStr	入力	データソース接続に使用することができる接続ストリング。(null不可).

戻り値	説明
SUCCESS	Hdbcを使ったデータベース接続は常にSUCCESSを返すでしょう。接続ストリングを使ったデータ・ベース接続の場合、それが成功裡に接続した場合に、SUCCESSを返すでしょう。.
SQLConnect Error	接続ストリングを使ってデータ・ベース接続をセットする場合にエラーが生じれば、このエラー・メッセージを返すでしょう。

SETXTT

run()の前に必要な入力XTTをセットします。

シンタックス

```
void setXtt( const char *xttFileName );
```

XTTファイル名をセットします。

引き数	入力/出力	説明
xttFileName	入力	XTTファイル名

```
void setXtt( const char * xttValuePtr,
             int szXttValue );
```

XTTコンテンツをセットします。

引き数	入力/出力	説明

引き数	入力/出力	説明
xttValuePtr	入力	XTTコンテンツを備えたバッファーへのポインター。
SzXttValue	入力	xttValuePtrの長さ

SETOUTPUTXML

出力XMLをセットします。

シンタックス

```
void setOutputXml( const char *xmlFileName );
```

出力XMLファイル名をセットします。

引き数	入力/出力	説明
xmlFileName	入力	出力XMLファイル名(null不可)。結果XMLはファイルに書き込まれるでしょう。

```
void setOutputXml( char * xttValuePtr,
                   int szXttValue );
```

出力コンテンツのためにメモリ・バッファーをセットします。出力バッファーはnull終止です。

引き数	入力/出力	説明
xmlValuePtr	出力	出力XMLコンテンツ用のバッファーへのポインター。出力結果はnull終止です。
szXmlValue	入力	xmlValuePtrバッファーのサイズ。

SETPARAMETERS

テンプレート・ファイルに対するパラメーター値をセットします。例えば、XTTファイルに定義された2つのパラメーターがある場合:

```
<xtt:template xmlns:xtt="urn:schema-dbmaster-com:xml-template" encoding="UTF-8">
<xtt:parameter name="student_id" default="10001"/>
<xtt:parameter name="class_id" default="201"/>
...
```

```
</xtt:template>
```

setParametersメソッドは、あなたが各パラメーターの値を設定することを可能にします。引き数はストリング(それは数ペアのパラメーター名および値を含んでいるかもしれない)です。各ペアはコンマによって分離されます。等号の左側のテキストはパラメーターの名前、右側は値です。例えば上記の例においてXTTファイル用にパラメーターをセットするには：

```
setParameters("student_id='10012',class_id='103'").
```

シンタックス

```
void setParameters ( const char *parameters );
```

引き数	入力/出力	説明
parameters	入力	テンプレート・ファイル用のパラメーター： フォーマット： *[param_name="value" [,param_name="value"]...]* 注:値はシングルクォートまたはダブルクォートで囲むことができます。.

SETGENXMLHEADER

XMLヘッダー<?xml...?>を生成してxml出力を書き出す場合、このパラメーターをセットしてください。。デフォルト値はtrueです。もしfalseであれば、ヘッダーは書き出されないでしょう。

シンタックス

```
void setGenXMLHeader( bool stat );
```

引き数	入力/出力	説明
stat	入力	ヘッダーを備えたxmlを出力するかどうかの、Boolean条件。.

GETERROR

run ()かsetDatabaseConnection()からのリターン・コードが、SUCCESSでない場合は、エラー・メッセージを得るこのメソッドを使用してください。エラーがない場合、このメソッドはNULLを返すでしょう。

シンタックス

```
const char * getError();
```

RUN

Run()メソッドは、指定されたXTTファイル名あるいはメモリから、XTTコンテンツを抽出して内部プロセスができるXTTオブジェクト・モデルへロードするでしょう。XTTの中で設定された指示に従ってプロセスされて、指定された出力XMLファイルあるいは出力メモリのいずれかに出力されます。もし成功裡に実行されれば、リターン・コードは0でしょう。エラーが生じる場合は、エラー・メッセージを得るgetError()メソッドを使用してください。

2番目と3番目のメソッドはプロパティをセットするために対応するsetterメソッドを呼び出し、そして次にrun()を呼び出します。setXtt()が再び呼び出されるか、XTTオブジェクトが破壊されるまで、XTTオブジェクト・モデルはキャッシュされます。

シンタックス

```
int run ( );
int run ( const char * connectionStr,
          const char *xttFileName,
          const char *xmlFileName );
```

引き数	入力/出力	説明
connectionStr	入力	データソース(NULL不可)に接続するために使用することができる接続ストリング。
xttFileName	入力	XTTファイル名
xmlFileName	入力	出力XMLファイル名。それはNULL不可です。生成されたXMLは、ファイルに書き込まれるでしょう。

```
int run ( const char * connectionStr,
          const char *xttValuePtr,
          int szXttValue,
          char *xmlValuePtr,
          int szXmlValue);
```

引き数	入力/出力	説明
connectionStr	入力	データソース(NULL不可)に接続するために使用することができる接続ストリング。
xttValuePtr	入力	XTTコンテンツを備えたバッファーへのポインター。
szXttValue	入力	xttValuePtrの長さ
xmlValuePtr	出力	出力XMLコンテンツ用のバッファーへのポインター。出力結果はNULL終止になります。
szXmlValue	入力	xmlValuePtrバッファーのサイズ。

戻り値	説明
SUCCESS	変換が成功しました。
ERR_XTT_INV_ARG	XTTエンジンがプロセスするために必要な引き数/プロパティが見当たりません。例えば、XTTファイル名かXTTコンテンツがセットされていません。
ERR_XTT_XERCES_PARSER	apache xercesパーサによって返されたエラー。XTTは有効なXMLフォーマットではありません。
ERR_XTT_INV_SYNTAX	無効のXTTシンタックス。可能な原因： 認識されたxttエレメントあるいはアトリビュートはありません xtt:templateの中でユーザに定義されたエレメントではありません。 <xtt:attribute>が、その親の中の任意のユー

	ザ定義エレメントの後に宣言されています。 <xtt:parameter>が<xtt:template>の下のユーザ定義エレメントの後に宣言されます。 要求されたアトリビュートが見当たりません。
ERR_XTT_PROCESS	XTTプロセス・エラー: フォルダーまたはファイルの作成に失敗しました。 無効の変数参照。
SQL Error	SQLエラーが処理中に生じました。

例：

```
/* an object of XTT */
XTT transfer;

/* set database connection pre-connected odbc connection */
transfer.setDatabaseConnection(hdbc);

/* set input XTT filename */
transfer.setXtt("c:\\temp\\abc.xtt");

/* set output XML filename */
transfer.setOutputXml("c:\\temp\\abc.xml");

/* start transformation */
int rc = transfer.run();

/* check if process fails.
   if so, getError to get the detail error message. */
if ( rc != 0 )
{
  printf("%d %s\n", rc, transfer.getError());
}
```

もしくは

```
/* an object of XTT */
XTT transfer;

/* transfer with specified arguments
- database connection string
- input XTT filename
- output XML filename
*/
int rc = transfer.run( "DSN=DBSAMPLE;UID=SYSADM;PWD=x123",
"c:\\temp\\abc.xtt",
"c:\\temp\\abc.xml");
```

5.2 CのXTT API

XTTは、XTTファイル中の定義によってデータベースのデータをトランスファーします。

シンタックス

```
int Xtt(                    const HDBC hdbc,
                            const char *xttFileName,
                            const char *parameters,
                            const char *xmlFileName,
                            char * msgBuffer,
                            int szMsgBuffer )

int XttMem (                const HDBC hdbc,
                            const char *xttValuePtr,
                            int szXttValue,
                            const char *parameters,
                            char *xmlValuePtr,
                            int szXmlValue,
                            unsigned char * msgBuffer,
                            int szMsgBuffer )
```

両方の関数はXTTをプロセスし、データベースからデータを抽出し、XML出力にそれを書くでしょう。関数XttMemがメモリにソース・コンテンツおよび出力バッファーを持っている一方、関数XttはXTTソースおよびXML出力ファイル名を行っています。

引き数 (Xtt用) :

引き数	入力/出力	説明
hdbc	入力	データベース接続ハンドル(NULL不可)。
xttFileName	入力	XTTファイル名:有効なファイルパスでなくてはなりません。
parameters	入力	テンプレート・ファイル用のパラメーター フォーマット： [param_name="value"

引き数	入力/出力	説明
		[,param_name="value"]...]
xmlFileName	入力	出力XMLファイル名。それはNULL不可です。生成されるXMLは、ファイルに書き込まれるでしょう。
msgBuffer	出力	エラー・メッセージ用のバッファーへのポインター。出力結果はNULL終止です。
szMsgBuffer	入力	msgBufferのサイズ。.

引き数 (XttMem用) :

引き数	入力/出力	説明
hdbc	入力	データベース接続ハンドル、NULL不可。
xttValuePtr	入力	XTTコンテンツを備えたバッファーへのポインター。
szXttValue	入力	xttValuePtrの長さ
parameters	入力	テンプレート・ファイル用のパラメーター フォーマット： [param_name="value" [,param_name="value"]...]
outputXmlPtr	出力	出力XMLコンテンツ用のバッファーへのポインター。出力結果はNULL終止になります。
szOutputXml	入力	outputXmlPtrバッファーのサイズ。
msgBuffer	出力	エラー・メッセージ用のバッファーへのポインター。出力結果はNULL終止になります。
szMsgBuffer	入力	msgBufferの長さ。

戻り値	説明
SUCCESS	変換が成功しました。
ERR_XTT_INV_ARG	XTTエンジンのプロセスに対する要求された引き数/プロパティが見当たりません。例えば、XTTファイル名かXTTコンテンツがセットされていません。
ERR_XTT_XERCES_PARSER	apache xercesパーサによって返されたエラー。XTTは有効なXMLフォーマットではありません。
ERR_XTT_INV_SYNTAX	無効のXTTシンタックス。可能な原因： 認識されたxttエレメントあるいはアトリビュートはありません xtt:templateの中でユーザに定義されたエレメントではありません。 <xtt:attribute>が、その親の中の任意のユーザ定義エレメントの後に宣言されています。 <xtt:parameter>が<xtt:template>の下のユーザ定義エレメントの後に宣言されます。 要求されたアトリビュートが見当たりません。
ERR_XTT_PROCESS	XTTプロセス・エラー： フォルダーまたはファイルの作成に失敗しました。 無効の変数参照。
SQL Error	SQLエラーが処理中に生じました。

5.3 JavaのXTT API

パブリック メソッド:

CONSTRUCTOR AND DESTRUCTOR:

```
XTT( )
```

Constructs a XTT object.

```
finalize ( )
```

Before destroying a XTT object.

SETDATABASECONNECTION

この機能はデータベース接続をセットします。接続ストリングのメソッドを使用して、以前の呼び出しによって確立されていたどんな接続も、最初に切断されるでしょう。その接続は*setDatabaseConnection((String)NULL).* を呼び出すことによって強制的に切断することができます。

シンタックス

```
int setDatabaseConnection( JdbcOdbcConnection conn );
```

引き数	入力/出力	説明
conn	入力	Dbmaster jdbc接続オブジェクト (NULL不可)。.

```
int setDatabaseConnection( String connStr );
```

引き数	入力/出力	説明
connStr	入力	データソースに接続可能な接続ストリング(NULL不可)

戻り値	
SUCCESS	hdbcとのデータベース接続のセットは常にSUCCESSを返すでしょう。接続ストリングとのデータベース接続をセットする場合、それは指定された接続を使用して接続しようとし、成功した場合SUCCESSを返すでしょう。
SQLConnect Error	接続ストリングとのデータ・ベース接続のセットが失敗する場合、関数はこのストリングを返すでしょう。

SETXTT

XTTファイル名かコンテンツをセットします。

シンタックス

```
void setXtt( String xttFileName );
```

引き数	入力/出力	説明
xttFileName	入力	XTTファイル名

```
void setXtt( byte [] xttValue );
```

引き数	入力/出力	説明
xttValue	入力	XTTコンテンツを備えたバイト配列。

SETOUTPUTXML

出力コンテンツのために出力XMLファイル名あるいはメモリ・バッファーをセットします。

シンタックス

```
void setOutputXml( String xmlFileName );
```

引き数	入力/出力	説明
xmlFileName	入力	出力XMLファイル名。

```
void setOutputXml( byte [] xmlValue );
```

The output buffer is null terminated.

引き数	入力/出力	説明
xmlValue	入力	出力XMLコンテンツを備えたバイト配列。

SETPARAMETERS

テンプレート・ファイルに対するパラメーター値をセットします。

シンタックス

```
void setParameters ( String parameters );
```

引き数	入力/出力	説明
parameters	入力	テンプレート・ファイル用のパラメーター フォーマット： [param_name="value" [,param_name="value"]...]

GETERROR

run()からのリターン・コードがSUCCESSでない場合は、エラー・メッセージを得るこのメソッドを使用してください。エラーが返されない場合、このメソッドはNULLを返すでしょう。

シンタックス

```
String getError();
```

RUN

XML変換を始めます。.

シンタックス

```
int run ( );
int run (  String connectionStr,
           String xttFileName,
           String outputFileName );
```

引き数	入力/出力	説明
connectionStr	入力	データソース(NULL不可)に接続するために使用することができる接続ストリング。
xttFileName	入力	XTTファイル名
xmlFileName	入力	出力XMLファイル名(NULL不可)。生じるXMLは、ファイルに書き込まれるでしょう。

```
int run (  String connectionStr,
           byte [] xttValuePtr,
           byte [] xmlValuePtr);
```

引き数	入力/出力	説明
ConnectionStr	入力	データソース(NULL不可)に接続するために使用することができる接続ストリング。
XttValue	入力	XTTコンテンツを備えたバイト配列。
XmlValue	出力	出力XMLコンテンツためのバイト配列。出力結果はNULL終止になります。

戻り値	説明
SUCCESS	変換成功.
ERR_XTT_INV_ARG	XTTエンジンのプロセスに対する必要な引き数/プロパティが見当たりません。例えば、XTTファイル名かXTTコンテンツがセットされていません。
ERR_XTT_XERCES_PARSER	apache xercesパーサーによって返されたエラー。XTTは有効なXMLフォーマットではありません。
ERR_XTT_INV_SYNTAX	無効なXTTシンタックスです。可能な原因： 認識できるxttエレメントかアトリビュートはありません。 xtt:template elementの中でユーザ定義されたエレメントではありません。 <xtt:attribute>その親の中の任意のユーザ定義エレメントの後に宣言されます。 <xtt:parameter>が、<xtt:template>の下のユーザ定義エレメントの後に宣言されています。

	要求されたアトリビュートが見つかりません。
ERR_XTT_PROCESS	XTTプロセス・エラー： フォルダーまたはファイルの作成に失敗しました。 無効な変数参照。
SQL Error	SQLエラーが処理中に生じました。

例:

```
    String xtt = "c:\\temp\\case1.xtt";
    String xml = "c:\\temp\\case1.xml";
    String error = "c:\\temp\\case1.log";
    String param = null;
    String dbname = "DBSAMPLE";
    String user = "SYSADM";
    String password = "";
    try
    {
      /* connect to DBMaster database via dbmaster jdbc bridge */
      Class.forName("dbmaster.sql.JdbcOdbcDriver");
      System.setProperty("DM_DRIVER_MODE", "CLIENT_SERVER");
      System.setProperty("DM_CONNECT_MODE", "CONNECT_DB");
      Connection conn = DriverManager.getConnection("jdbc:dbmaster:"+dbname,
user, password);
      /* new XTT object */
      XTT transfer = new XTT();

      /* set database connection with previous connected Jdbc connection. */
          transfer.setDatabaseConnection((dbmaster.sql.JdbcOdbcConnection)conn);

      /* set XTT filename */
          transfer.setXtt(xtt);

      /* set output XML filename */
          transfer.setOutputXml(xml);

      /* set parameter values for XTT template file */
          transfer.setParameters(param);

      /* start XML transformation */
      int rc = transfer.run();

          if( rc != 0 )
      {
        /* print out error message */
            System.out.println(transfer.getError());
          }
      conn.close();
    }
    catch( ClassNotFoundException e )
    {
      e.printStackTrace();
    }
    catch( SQLException sqle )
    {
      sqle.printStackTrace();
    }
```

5.4 XTTストアド・プロシジャー

XTTストアド・プロシジャーはC、C++およびJavaのAPIに似ているXML変換関数を提供します。主な違いは：

- XTTと出力XMLファイル名のみを受理します。
- 指定されたファイル名がサーバー・サイト上に存在します。

ストアド・プロシジャー定義:

```
CREATE PROCEDURE XTT(
  VARCHAR(257)  XTTFILE              INPUT,
  VARCHAR(257)  OUTPUTFILE           INPUT,
  VARCHAR(257)  PARAMETERS    INPUT );
```

引き数	入力/出力	説明
XTTFILE	入力	XTTファイル名
OUTPUTFILE	入力	出力XMLファイル名
PARAMETERS	入力	XTTテンプレート用のパラメーター値

権限

このストアド・プロシジャーの所有者はSYSTEMです。ストアド・プロシジャーの権限はストアド・プロシジャーを実行するユーザと同じです。例えば、XTMファイルにある表へのクエリーがある場合、現在ログオンしているユーザがその表をselectする権限を持っていない場合、SQLエラーが返されるでしょう。

例:

```
dmSQL> call XTT('d:\document\test\xtt\case1.xml','c:\temp\case1.xml',NULL);

dmSQL> call XTT('d:\document\test\xtt\case1.xml',NULL,NULL);
ERROR (5612): [Dbmaster] xtt invalid argument :
   xml filename or xml destination buffer is required.
```